Evaluation of Kalman Filtering for Network Time Keeping

Aggelos Bletsas
Massachusetts Institute of Technology
Media Laboratory
20 Ames Street, Cambridge MA 02139
aggelos@media.mit.edu

Abstract

Time information is critical for a variety of applications in distributed environments that facilitate pervasive computing and communication. This work describes and evaluates a novel Kalman filtering algorithm for end-to-end time synchronization between a client computer and a server of "true" time (e.g. a GPS source) using messages transmitted over packet switched networks, such as the Internet. The messages exchanged have the NTP format and the algorithm evaluated, is performed only at the client side. The Kalman filtering algorithm is compared to two other techniques widely used, based on linear programming and statistical averaging and the experiments involve independent consecutive measurements (gaussian case) or measurements exhibiting long-range dependence (self-similar case). Performance is evaluated according to the estimation error of frequency offset and time offset between client and server clock, the standard deviation of the estimates and the number of packets used for a specific estimation. The algorithms can exploit existing NTP infrastructure and a specific example is presented.

1 Introduction

In this paper we evaluate a novel Kalman filtering algorithm for end-to-end time synchronization between two computers exchanging messages over a packet switched network such as the Internet. The Kalman filtering algorithm is compared to two other techniques widely used, based on linear programming and statistical averaging. We are particularly interested in the calculation of *frequency offset* and *time offset* between the clock of a client pc and the clock of a remote server. The server acts as a source of "true time".

Accurately synchronized clocks enable services and provide the basis for efficient communications. Autonomous sensor array operation is facilitated by accurate time stamps [8]. Global Positioning System, as well as proposed Ultra-Wide Band urban and intra-building location systems rely on precise timing measurements. Internet performance can be evaluated from accurate measurement of the delay between various nodes in the network. Various important Internet Protocols such as TCP could benefit from accurate time keeping [9].

In this work, we are using the Network Time Protocol (NTP) [5] messages between a client and a single server in three different algorithms and evaluate their performance. However, the algorithms do not depend on the details of the NTP message format and other formats could be adopted with minor modifications. NTP is a hierarchical client-server synchronization scheme widely used and can provide for accuracies on the order of milliseconds under the assumption that there is a *reasonably short* round trip time between client and server. NTP also incorporates connections to multiple servers for increased reliability. The following discussion is about the algorithms for the local clock parameters estimation when an Internet connection is used to a single time reference server. These parameters could then be used to steer the local clock according to hardware and operating system details.

The rest of the paper is organized as follows. In Section 2 we introduce the terminology used throughout this work and proceed to formulate the problem. We present prior art and justify the selection of the three algorithms compared in this work. In Section 3 we analytically present the three algorithms based on Kalman filtering, Linear Programming and Averaging of Time Differences and in Section 4 we investigate their performance for the Gaussian Case (no dependence between successive measurements) and for the Self-Similar Case (long range dependence). We conclude in Section 5.

2 Background

2.1 Clock Basics

A *clock* $C(t)$ reporting the value of time t can be considered as a piecewise linear function, twice differentiable except on a countable set of point $P \subset R$, where P is finite:

$$C(t) : R \to R,\ C'(t) = \frac{dC(t)}{dt},\ C''(t) = \frac{dC^2(t)}{dt^2},\ t \notin P \tag{1}$$

The set P includes all the points were the clock parameters suddenly change, therefore no derivative of the function $C(t)$ can be defined. A "true" clock can be described by the identity below:

$$C(t) = t \equiv T(t) \quad and \quad ||P|| = 0 \tag{2}$$

which states that the (ideal) source of true time is a strictly linear function of time. In practice, clocks due to thermodynamic fluctuations deviate from the strictly linear case.

Using the representation $C(t)$ for a clock reading and $T(t) = t$ for true time, the following definitions are presented:

- **time offset:** the difference between the time reported from a clock and the "true" time: $C(t) - T(t) = C(t) - t$. In this paper we will refer to the time offset calculated for $t = 0$ as θ and for $t \neq 0$ as x.
- **frequency offset** (also referred as *skew*)**:** the difference in frequencies between a clock and the "true" time: $C'(t) - T'(t) = C'(t) - 1$. In this work, we will refer to frequency offset as $\phi - 1$.
- **drift:** the rate of clock frequency changes: $C''(t) - T''(t) = C''(t)$.

Typical quartz oscillators (without any type of temperature compensation) exhibit frequency offsets on the order of a few parts per million (PPM). For example a 10 PPM oscillator will introduce an uncertainty (i.e. error) of 36 msec in one hour. Cesium beam atomic clocks on the other hand, exploiting the stabilities of the quantum world perform better with uncertainties close or smaller than 1 nsec in 24 hours.

Modeling a clock as a piecewise linear function of time, with a discrete set P of discontinuities, is a reasonable step since any function can be approximated in a similar manner. The client should estimate only two parameters, namely the time and frequency offset θ, $\phi - 1$ respectively, compared to the source of true time $T(t)$, since only two parameters are needed to define a line.

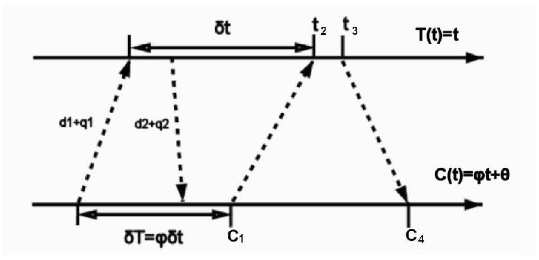

Figure 1. Exchanging timestamps between client and time server. Notice that a time difference of δt according to server clock is translated to $\phi\delta t$ according to client clock.

The parameters ϕ, θ change with a rate related to the clock drift, defined above. That rate of clock parameters shift, should determine how often the clock synchronization process should be re-run so as the new ϕ, θ be estimated. A statistical tool to quantify the rate of clock parameters change, correlated with the duration of the observation interval is the Allan variance [3] and it has been found that for most free running oscillators used in current computer systems, this change happens at intervals on the order of 1-2 hours or more [4]. This shift rate of the two clock parameters discussed is reasonable to expect since temperature that influences the stability and accuracy of oscillators changes at that rate. For an excellent review of oscillators, Allan variance, time and frequency metrology, the interested reader could refer to [3].

2.2 Problem Formulation

After the description of the clock nomenclature followed in this work, we are ready to formulate the problem. The client clock $C(t)$ is *synchronized* to a time source $T(t) = t$ when both frequency offset ϕ and time offset θ are estimated.

The client timestamps ($C(t_1)$) a UDP packet according to each own clock $C(t)$ and sends the message to a time source server which timestamps the packet upon reception and retransmission (t_2, t_3 respectively) back to the originating client (Fig. 1). The client timestamps again the message upon reception and therefore acquires a set of 4 timestamps: $(C(t_1), t_2, t_3, C(t_4))$. For convenience, we will notate $C(t_1)$ as C_1 and $C(t_2)$ as C_2 from now on. The same process can be repeated for a set of N consecutive messages. Therefore we should answer the following questions:

- What is the optimal processing of the N messages $(C_1^1, t_2^1, t_3^1, C_4^1), (C_1^2, t_2^2, t_3^2, C_4^2), \ldots, (C_1^N, t_2^N, t_3^N, C_4^N)$ so as to obtain unbiased estimates with minimum error?

- What is the cost of obtaining estimates of ϕ and θ in terms of bandwidth spent (number N, inter-departure time between packets)?
- Do the algorithms employed in the estimation of ϕ, θ impose special restrictions in the operation of client (or server) operating system (i.e. are there any major non-algorithmic modifications in the operation of existing client/time server daemons)?

The number of packets N exchanged between client and server (Fig. 1) is a crucial parameter of any algorithm eventually adopted, considering the heavy load current Internet time servers service, on the order of a $1000 - 1200$ of requests per second and increasing every year. Moreover, the inter-departure intervals of the NTP-like messages should not be very large since closely spaced packets ensure that the clock parameters are not changing during the measurement.

Finally, we need to emphasize that the queuing delay q_1 across the *forward* path (from client to server) is never constant and generally different from the queuing delay q_2 across the *reverse* path (from server to client) (Fig. 1). Moreover, since the messages are carried through UDP packets, the forward and reverse routes could be physically different and therefore the propagations delays [1] d_1, d_2 could be unequal across the forward and reverse paths.

$$d_1 + q_1 \neq d_2 + q_2 \tag{3}$$

2.3 Prior Art on Client-Server Schemes

NTP estimates the time offset using the 4 timestamps of a message, according to the following equation (Eq. 4):

$$\hat{x}_n = \frac{C_1^n - t_2^n - t_3^n + C_4^n}{2} \tag{4}$$

Since the round-trip time (rtt) is on the order of a few msecs, the contribution of the frequency skew on the total error is negligible (e.g. a 10 ppm oscillator for a 10 msec rtt exhibits 0.1 μsec which is on the order of "noise" due to the operating system) and therefore excluded from 4. The frequency offset can be estimated using several measurements of $\theta \equiv x$.

From a closer look on Eq. 4, NTP estimates are erroneous by a quantity proportional to half the difference between forward and reverse path delays (*assymetry*).

$$\begin{aligned} \hat{x}_n &= x_n + \frac{d_2^n + q_2^n - d_1^n - q_1^n}{2} \Rightarrow & (5) \\ \hat{x}_n &= x_n + w_n & (6) \end{aligned}$$

[1]Time needed for the first bit to arrive at the destination as opposed to transmission delay which is related to the speed of the link.

That is why the NTP error is upper bounded by half the round-trip time. If we make the assumption that the assymetry, depicted as "noise" w_n in Eq. 6 for the $n - th$ NTP message, is a Gaussian, zero-mean random variable, then the estimate of Eq. 4 is the Maximum Likelihood estimate, equivalent to the efficient[2] minimum variance, unbiased estimator for this particular case, according to the Gauss-Markov theorem. However, the assymetry is not always gaussian, as we will discuss in the following sections.

Line fitting techniques, based on the median slope calculated from averaged one way delay measurements [10] or linear programming [6] are alternative proposals for frequency offset estimation. The linear programming technique proposed in [6] is revisited with a slightly different derivation which provides not only for frequency offset ($\phi - 1$) estimation but also for time offset estimation (θ).

In the gaussian case, averaging N measurements from Eq. 4 can improve the estimates (decreasing the variance of the estimate) by a factor of $\sqrt{N}$. This is an idea exploited in the client-server synchronization schemes deployed by the National Institute of Standards and Technology (NIST) using dedicated phone lines [2] or the Internet [4].

A variant of this method is discussed in this work.

Finally, Kalman filtering is an attractive alternative for clock parameter estimation [1], since Kalman filters are the optimal linear estimators for the Gaussian case i.e. the linear estimators that minimize the Mean Square Error (MSE). As we will see in the next section, the problem can be formalized using the Kalman filtering notation and due to the optimality property (at least for the Gaussian case) excels over a range of recursive estimators like phased lock loops [1].

The optimality and the appealing recursive nature of Kalman filtering, the intuitive structure (as explained below) of the linear programming technique and the simplicity of the averaging technique (referred as *Averaged Time Differences (ATD)*) as well as its wide deployment, were the reasons behind the selection of the above algorithms for comparative performance evaluation.

3 The Algorithms

3.1 Kalman Filtering

The motivation behind the adoption of the Kalman filtering notation stems from a simple observation: a time interval δt according to the "true" time is translated to $\phi\,\delta t$ according to the client clock (Fig. 1). Therefore, it is sufficient for the client to send the messages at constant intervals δT measured according to the local clock and estimate the

[2]The efficient estimator when exists achieves the minimum variance of the estimate, equal to the Cramer-Rao bound.

inter-arrival intervals at the server, using the timestamps t_2^n which correspond to the "true" time. Variation of forward and reverse one-way delays jitter acts as *noise* in the estimation process.

With the above, the formulation of the problem using Kalman filtering becomes clear: the client sends the NTP packets at constant intervals δT and estimates the inter-arrival interval $s = \frac{\delta T}{\phi}$ in the presence of network jitter v, exploiting the measured inter-arrival intervals $y_n = t_2^{n+1} - t_2^n$ for $n \in [1..N]$. The measurement and state model of the Kalman filter easily follow:

$$\begin{aligned} y_n &= t_2^{n+1} - t_2^n, \text{ n=1..N-1} && (7) \\ s_n &= \frac{\delta T}{\phi} && (8) \end{aligned}$$

$$\begin{aligned} y_n &= x_n + v_n, \text{ measurement model} && (9) \\ s_{n+1} &= s_n + w_n, \text{ state model} && (10) \end{aligned}$$

$$\begin{aligned} E[v_n v_i] &= R\,\delta_{ni} && (11) \\ E[w_n w_i] &= Q\,\delta_{ni} && (12) \\ E[v_n w_i] &= 0 && (13) \\ \delta_{ni} &= \begin{cases} 1 & \text{n = i} \\ 0 & \text{n} \neq \text{i} \end{cases} \end{aligned}$$

From Fig. 1 it is easily derived for constant inter-departure intervals δT that:

$$t_2^{n+1} - t_2^n = \delta t + q_1^{n+1} - q_1^n + d_1^{n+1} - d_1^n \qquad (14)$$

Therefore, we have the following estimate for the power R of the "jitter" noise:

$$\begin{aligned} \hat{R} &= var(t_2^{n+1} - t_2^n) = var(y_n) && (15) \\ \hat{Q} &= var(C_1^{n+1} - C_1^n) && (16) \end{aligned}$$

In practice, the inter-departure times between consecutive packets from the client will not be constant, possibly due to operating system delay jitter. The timestamps C_1^n could be used to estimate the power Q of the state model noise w. Alternatively, we can treat that noise as additional measurement noise and set $Q = 0$ (that was the approach followed in this work). Nevertheless, maintaining w and Q as small as possible (i.e. keeping the inter-departure intervals constant) improved performance.

The Kalman filtering technique is a recursive scheme, therefore the estimate x_n converges to the correct value of δt after a number of messages $(C_1^j, t_2^j, t_3^j, C_4^j)$. The initial value of x was set to δT and an averaging FIR filter of 11 samples was applied to obtain the final estimate of δt. The frequency of the client clock is then obtained by:

$$\hat{\phi} = \frac{\delta T}{\hat{s}} \qquad (17)$$

For the estimation of the time offset θ we could refer to Eq. 4. However for a large number N of packets used, the duration of the experiment multiplied by the frequency skew could contribute significantly to the time offset error (e.g. 100 packets spaced 1 sec from each other correspond to an additional offset of 1 msec in a 10 ppm clock during the synchronization experiment). Therefore the estimate of the frequency offset should be used in the time offset calculation.

From Fig. 1 we have the following relationships:

$$\begin{aligned} C_1^n - \phi\, t_2^n &= \theta - \phi\,(d_1 + q_1)^n && (18) \\ C_4^n - \phi\, t_3^n &= \theta + \phi\,(d_2 + q_2)^n && (19) \end{aligned}$$

$$\begin{aligned} C_1^n - \phi\, t_2^n &\leq \theta - \phi\, d_1 && (20) \\ C_4^n - \phi\, t_3^n &\geq \theta + \phi\, d_2 && (21) \end{aligned}$$

Therefore, an estimate of θ is obtained by the following relationship:

$$\hat{\theta} = \frac{max(C_1^i - \hat{\phi}\, t_2^i) + min(C_4^j - \hat{\phi}\, t_3^j)}{2} \qquad (22)$$

The Kalman filter *"predict"* and *"update"* equations are omitted and can be found in a relevant textbook, such as [7].

3.2 Linear Programming

This line fitting technique exploits both the forward and reverse path timestamps, by estimating a "clock line" that minimizes the distance between the line and the data, leaving all the data points below the line on a (t_2, C_1) plane or above the line on a t_3, C_4 plane. The following equations describe the problem and it's solution:

3.2.1 Forward Path

$$\begin{aligned} \alpha_1 &= \phi && (23) \\ \beta_1 &= \theta - \phi\, d_1 \end{aligned}$$

$$\text{Eq. 20} \Rightarrow \alpha_1\, t_2^n + \beta_1 - C_1^n \geq 0, \forall\, n \in [1..N] \qquad (24)$$

Find α_1, β_1 that minimize

$$f(\alpha_1, \beta_1) = \sum_{n=1}^{N} (\alpha_1\, t_2^n + \beta_1 - C_1^n) \qquad (25)$$

under the constraint of Eq. 24

3.2.2 Reverse Path

$$\begin{aligned} \alpha_2 &= \phi \\ \beta_2 &= \theta + \phi\, d_2 \end{aligned} \tag{26}$$

$$\text{Eq. } 21 \Rightarrow C_4^n - \alpha_2\, t_3^n - \beta_2 \geq 0, \forall\, n \in [1..N] \tag{27}$$

Find α_2, β_2 that minimize

$$f(\alpha_2, \beta_2) = \sum_{n=1}^{N} (C_4^n - \alpha_2\, t_3^n - \beta_2) \tag{28}$$

under the constraint of Eq. 27

$$\hat{\phi} = \frac{\alpha_1 + \alpha_2}{2} \tag{29}$$

$$\hat{\theta} = \frac{\beta_1 + \beta_2}{2} \tag{30}$$

The simple and intuitive derivation above sets this technique as a strong candidate for clock parameter estimation.

3.3 Averaged Time Differences

The time offset x_n is computed according to the NTP formula (Eq. 4) and therefore this method has all the limitations discussed at the NTP section above. Differences of the time offset estimates provide estimates for the frequency offset. Particularly, clusters of 25-50 closely spaced messages are used, time offsets are computed and the results are averaged to a single data point for the time offset. Then that is used in the following formula for frequency offset estimation.

$$\begin{aligned} \hat{f}(t_{n+1}) &= \frac{x_{n+1} - x_n}{\tau} \\ \hat{f} &\equiv \hat{\phi} - 1 \end{aligned} \tag{31}$$

$$y(t_{n+1}) = \frac{y(t_n) + \alpha \hat{f}(t_{n+1})}{1 + \alpha} \tag{32}$$

The value of τ nominally should be equal to $t_{n+1} - t_n$ however this quantity cannot be measured by the client's own clock. Nevertheless for small values of the frequency offset this can be set to $C(t_{n+1}) - C(t_n)$, since that is what the client can measure. The estimated frequency offset is averaged again using an exponential filter with a time constant α that depends on the stability of the local oscillator. Then the filtered frequency offset is used in the following:

$$\hat{x}(t_{n+1}) = \hat{x}(t_n) + y(t_n)(\tau) \tag{33}$$

A variant of this method is used in this work. Frequency offsets are calculated using Eq. 31 and then filtered using the above exponential filter with $\alpha = 0.5$. The final frequency offset estimation is the mean of all the N exponentially filtered frequency offsets calculated at each epoch.

The power of this method is its simplicity. For the Gaussian case where consecutive measurements are independent from each other, an increase of samples averaged by a factor of N reduces the variance of the estimate by a factor of $\sqrt{N}$. Therefore, there is a trade-off between accuracy achieved and cost of realizing it.

4 Performance

In this section we evaluate the performance of the three algorithms in two separate cases:

- The "Gaussian case" where the queuing delay difference between two consecutive NTP messages is a gaussian random variable. Consequently, the dispersion of the packets at the server is also a gaussian random variable. In this experiment, consecutive measurements are independent.
- The "Self-Similar case" where multiple pareto connections aggregate and form cross-traffic with long-range dependence.

The estimate, the variance of the estimate and the number N of packets used at each epoch are reported. In both cases the true clock frequency offset $\phi - 1$ was +10 ppm and the time offset θ was 20 msec. The NTP messages were transmitted at intervals of 1000 msec. Each experiment was run 200 times.

4.1 The Gaussian Case

The average round-trip time was on the order of 20 msecs and consecutive measurements were independent and identically distributed. In Fig. 2, Fig. 3 we present the average estimate and the standard deviation of the estimate for the frequency and time offset respectively, as a function of the number N of packets used.

The Kalman filter performed better when the number of packets N was above the minimum number of samples needed for convergence (on the order of 25-30 packets). This experimental finding is validated by the fact that the kalman filter (at steady-state) is the optimal linear estimator in the presence of gaussian noise. The ATD technique performed slightly better than the LP technique, as expected given the properties of averaging in the presence of gaussian noise. Frequency offset estimate variance was decreased with the number of packets used, while time offset estimates were close to the real value, regardless of N. This can be justified by the fact that the algorithms presented here focus on the accurate calculation of frequency offset which was

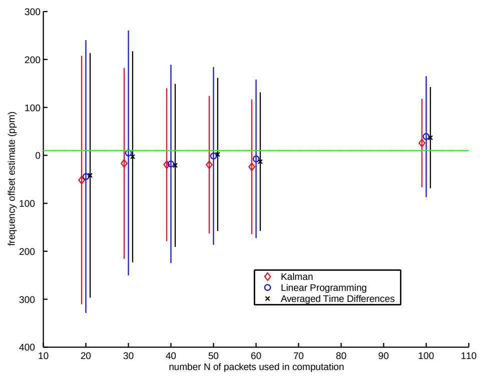

Figure 2. Frequency offset estimate and standard deviation as a function of N (number of packets used).

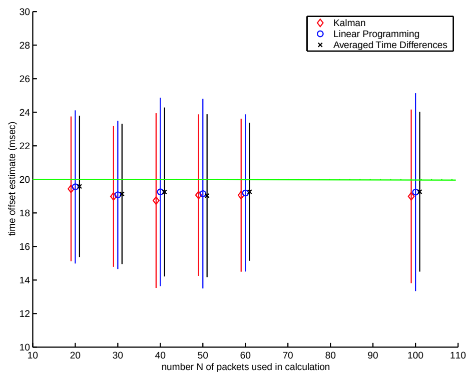

Figure 3. Time offset estimate and standard deviation as a function of N (number of packets used).

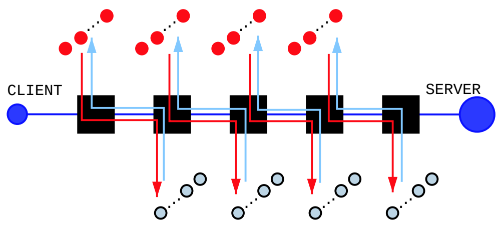

Figure 4. Simulation in ns-2 with pareto cross traffic. 14 connections per link per direction.

set at 10 ppm in this experiment. Error in the calculation of a 10 ppm quantity over a duration of 100 sec (1 packet every 1000 msec) is negligible in the calculation of time offset (using the algorithms described above) [3] and of course not visible at the time scales of Fig. 3.

4.2 The Self-Similar Case

In this section, we are investigating the performance of the three algorithms in the presence of bursty traffic. It has been shown that the aggregation of many on/off sources could form a self-similar source, exhibiting long range dependence [11]. The fact that Local Area Network traffic demonstrates chaotic (self-similar) behavior motivates the test of the three algorithms in a self-similar environment which is 'fundamentally different from the gaussian case for which the Kalman filtering and the ATD techniques are appropriate.

Fig. 4 displays the simulation setup in ns-2. The utilization of the links was 90% and the average round-trip time on the order of 40 msecs. The inter-departure time of NTP packets remains 1000 msecs.

Fig. 5 and Fig. 6 show in a sample run how well the Kalman filter "locks" onto the correct inter-arrival time δt and frequency offset value $(\phi - 1)$. Fig. 7 shows how well the ATD technique (with the exponential filter) "locks" onto the frequency offset value $(\phi - 1)$. The internal line is the filtered waveform through a low pass filter. Fig. 8 displays the one-way delay across the reverse path as a function of time. The trend of the plot is coherent with the following derivation. The "clock line" $\hat{\phi}\, t_3^n + \hat{\theta}$ with parameters estimated by the LP technique is also depicted.

$$\begin{aligned} Eq.21 &\Rightarrow && (34)\\ T_4^n - t_3^n &= (\phi - 1)\, t_3^n + \phi\,(d_2^n + q_2^n) + \theta \Rightarrow \\ T_4^n - t_3^n &\geq (\phi - 1)\, t_3^n + \phi\, d_2^n + \theta \end{aligned}$$

Fig. 9 shows the histogram of frequency offset estimates for the self-similar case, for $N = 100$ and Fig. 10 shows the performance of the three algorithms in the estimation of frequency offset, for various number N of packets at each epoch. Time offset estimation $\hat{\theta}$ resulted in errors equal to a fraction of the assymetry between the forward and reverse path, as expected.

From the above diagrams, it is deduced that the Kalman filtering technique no longer produces the best estimates with the smallest variance. The noise is no longer gaussian so Kalman filtering is not optimal and LP performs better in the presence of bursty traffic both in terms of the accuracy

[3] Time offset θ was estimated using the same algorithm for Kalman and ATD (described in the kalman fi ltering section).

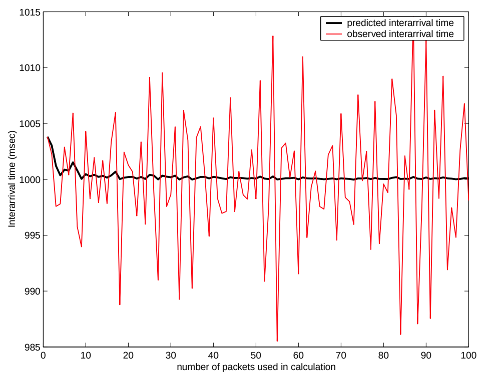

Figure 5. Predicted inter-arrival and Measured inter-arrival interval using the Kalman filter for self-similar cross traffic.

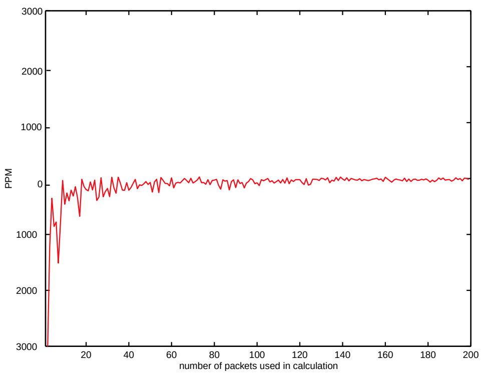

Figure 6. Estimation of frequency offset $\phi - 1$ using the Kalman filter for self-similar cross traffic.

of the estimate and its variance. For the same reason (burstiness and asymptotically long range dependence as opposed to the gaussian distribution around the mean), ATD performs inferiorly than the LP technique. All algorithms reduce the standard deviation (and therefore variance) of the estimate with increased number N of packets used, and the relation between that improvement and N is clearly not linear, as seen in Fig. 10.

4.3 Measurements

In order to emphasize the end-to-end character of the algorithms evaluated (especially for the case of Kalman filtering and LP), we modified an NTP client daemon and exchanged 100 packets at intervals of 1 sec with a stratum-0 server (connected to GPS). The time server was geograph-

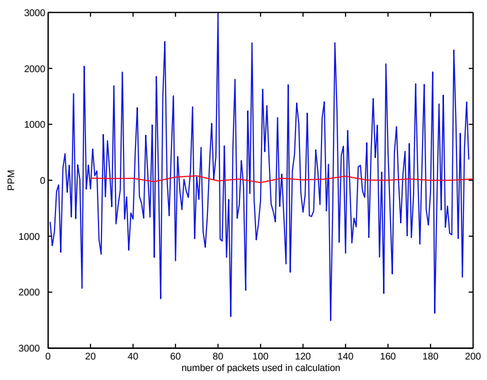

Figure 7. Estimation of frequency offset $\phi - 1$ using the ATD technique. A low pass filtering of the data is also plotted.

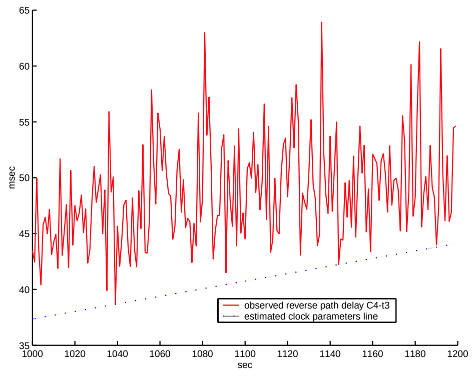

Figure 8. Delay $C_4^n - t_3^n$ from the reverse path and clock line estimation using LP for self-similar cross traffic.

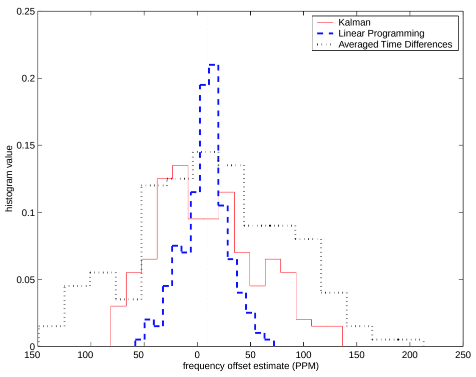

Figure 9. Histogram of the frequency offset estimates for self-similar cross traffic.

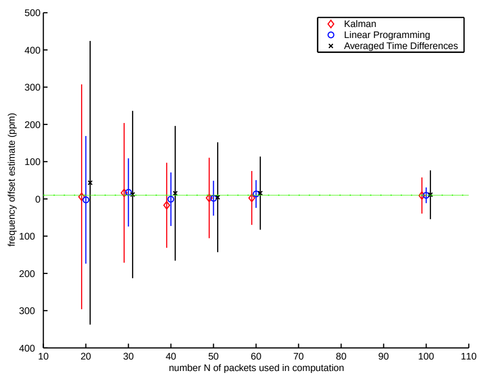

Figure 10. Frequency offset estimates for self-similar cross traffic as a function of number N of packets used in calculation.

Table 1. Frequency offset estimation using a NTP/GPS server.

	Kalman	LP	ATD
$\hat{\phi} - 1$ (PPM)	115.5	54.7	122.7

ically located at Palo Alto CA, 3,100 miles away from our client machine, with average round-trip time 85 msec, 18 hops away. We then processed the packets according to the algorithms evaluated above and the frequency offset estimation results are presented at Table 1.

An interesting idea could be averaging the two estimates calculated according to Kalman and LP since the former performed better at the Gaussian case and the latter at the Self-similar one.

5 Conclusion

The Kalman filtering technique, optimal for the Gaussian case, needed a considerable number of packets in order to converge (on the order of 20-30 packets for the formulation adopted and the experimental setup). Nevertheless, the technique performed well at the Self-similar case as well, with improved performance in terms of error and variance of the estimate when the number of packets N increased. The algorithm estimates the variance of network delay (jitter) and uses that estimate to calculate the frequency and time offset model variables. The algorithm can be applied without operation system requirements in the NTP-client daemon or any modifications in the NTP-server daemon. It could be benefitted by scheduled transmission from the client system that ensure minimum delay variance due to the operating system.

The Linear Prediction technique surpassed all the other techniques at the case of bursty traffic approximating real-word long-range dependence (chaotic) conditions even though it had inferior performance when measurements where completely independent. Its intuitive structure makes it attractive for straightforward implementation.

Finally, averaging as exploited and implemented in the Averaging Time Differences technique *(where equal intervals between measurements were used and therefore averaging differences of time was equivalent of averaging frequency skew estimates)* performed inferiorly to the LP and Kalman filtering techniques at the Self-similar case where measurements are not independent. However, its simplicity makes it attractive, especially at the cases where a small number of measurements are available or a trade-off between accuracy and cost of realizing it cannot be avoided.

All three techniques showed improvement in terms of estimate error and variance of the estimate when the number of packets N increased. From the plots, the relationship between improvement and N is not linear and therefore increased accuracy is expensive in terms of number of packets used (quadratic to N or worse).

References

[1] A. Bletsas. Time keeping in myriad networks: Theories, solutions and applications. *M.S. MIT*, June 2001.

[2] J. Levine. An algorithm to synchronize the time of a computer to universal time. *IEEE/ACM Transactions on Networking*, 3(1), February 1995.

[3] J. Levine. Introduction to time and frequency metrology. *Review of Scientifi c Instruments*, 70(6), June 1996.

[4] J. Levine. Time synchronization using the internet. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 45(2), March 1998.

[5] D. L. Mills. Network time protocol (version 3) specifi cation, implementation and analysis. *RFC 1305*, University of Delaware. March 1992.

[6] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. *18th IEEE INFOCOM*, 1999.

[7] S. J. Orfanidis. *Optimum Signal Processing, An Introduction, Second Edition.* McGraw-Hill, 1988.

[8] J. Paradiso, K.Hsiao, J. Strickon, and P. Rice. New sensor and music systems for large interactive surfaces. *Proceedings of the International Computer Music Conference (ICMC)*, August 2000.

[9] C. Parsa and J. Garcia-Luna-Aceves. Improving tcp congestion control over internets with heterogeneous transmission media. *Proceedings of IEEE ICNP 99*, October 1999.

[10] V. Paxson. Measurement and analysis of end-to-end internet dynamics. *Ph.D. University of California at Berkeley*, April 1997.

[11] M. S. Taqqu, W. Willinger, and R. Sherman. Proof of a fundamental result in self-similar traffi c modeling. *ACM Computer Communications Review*, April 1997.