High-Performance Synchronization for Circuit Emulation in an Ethernet MAN

Ilija Hadžić and Edward S. Szurkowski

Abstract: **Ethernet is being deployed in metropolitan area networks (MANs) as a lower-cost alternative to SONET-based infrastructures. MANs are usually required to support common communication services, such as voice and frame relay, based on legacy synchronous TDM technology in addition to asynchronous packet data transport.**

This paper addresses the clock synchronization problem that arises when transporting synchronous services over an asynchronous packet infrastructure, such as Ethernet.

A novel algorithm for clock synchronization is presented combining time-stamp methods used in the network time protocol (NTP) with signal processing techniques applied to measured packet interarrival times. The algorithm achieves the frequency accuracy, stability, low drift, holdover performance, and rapid convergence required for viable emulation of TDM circuit services over Ethernet.

Index Terms: **Circuit emulation, Ethernet, metropolitan networks, SONET, synchronization, TDM.**

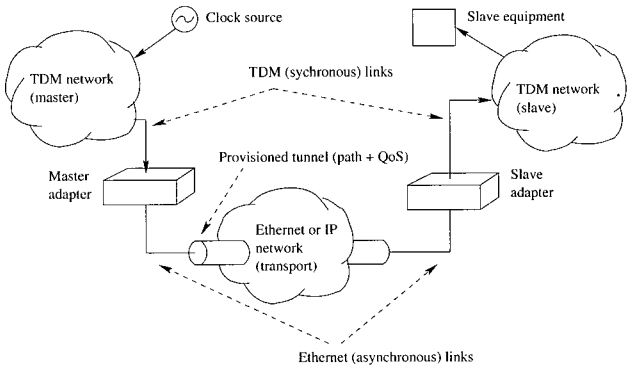

Fig. 1. General framework for clock synchronization problem.

I. INTRODUCTION

Most current metropolitan area networks (MANs) are based on SONET [1], a physical and link layer protocol designed for multiplexing and transport of constant bit rate time division multiplex (TDM) traffic over optical fiber. SONET networks require precise clock synchronization between sending and receiving stations to avoid data loss. Typically, a master clock operated by a carrier is used to provide the timing for all elements of a network. Although well suited for voice, transporting asynchronous data traffic over SONET is inefficient and often results in high equipment and operation costs [2].

To reduce data transport costs, some carriers have begun to deploy Ethernet rather than SONET in metro networks. Ethernet was originally designed for low-cost local-area data networking within enterprises [3], but the ongoing evolution of its capabilities has made it a viable alternative to technologies designed explicitly for public networks. Ethernet networks are asynchronous; that is, there is no shared or master clock driving the interconnected switches or terminals in the network.

Synchronous networks are efficient for transporting voice, which is usually carried as constant bit rate streams (circuits) between source and destination. The public switched telephone network (PSTN) is an example. The PSTN is fully synchronous; that is, all of the equipment that generates, manipulates or terminates the traffic is synchronized to a master clock. The voice equipment on a customer premises, such as a PBX, is also synchronized to the public network clock through the network access connection.

To reduce costs and complexity, end-users would generally prefer to access all public network services over a single integrated connection. These carrier services include TDM voice, data using TDM formats (e.g., frame relay), and packet data applications (e.g., Internet access or private office-to-office IP connectivity). To enable the integrated access, an Ethernet MAN must be capable of emulating legacy interfaces and functions provided by the SONET and the synchronous PSTN. The asynchronous property of Ethernet poses significant challenges to supporting clock synchronization in TDM-based services with quality equivalent to SONET networks. This paper presents a new technique to address the timing and synchronization problems that arise when providing *synchronous* TDM-based services over an *asynchronous* packet-based infrastructure.

Fig. 1 illustrates the general framework of the clock synchronization problem. A TDM network, designated as the "master," is connected via an adapter element to an asynchronous packet network. The second TDM network, designated as the "slave," is also connected to the packet network using another adapter element. A clock source is connected to the master TDM network for synchronization of its network elements. The clock source for network elements on the slave TDM network is the slave adapter, which derives its clock from the master TDM network.

The master and slave adapters connected to the TDM networks provide two key functions: (1) Encapsulation of TDM data into Ethernet frames, and (2) synchronization of the slave adapter clock to the service clock in the master TDM network. The encapsulation of the TDM data is straightforward and is beyond the scope of this paper. The remainder of this paper describes the method used by the master and slave adapters to

Manuscript received December 3, 2002; approved for publication by Gi-Hong Im, Division I Editor, September 30, 2004.

The authors are with the Bell Labs, Lucent Technologies in Murray Hill, New Jersey, USA, email: {ihadzic, szurko}@lucent.com.

1229-2370/05/$10.00

derive a clock on the slave adapter that is synchronized to the master TDM network clock with acceptable performance metrics.

The performance objective for the algorithm described herein is to create a T1 interface at the output of the Ethernet transport network that conforms to requirements set by the ITU-T G.824 [4] recommendation, as well as the Stratum-3 compatibility[1] requirements set by the ANSI T.101-1994 standard [5]. Meeting these performance requirements is generally considered to be adequate for public MANs.

Clock synchronization problem is not new and various synchronization techniques were developed as telecommunication systems evolved [6]. Existing synchronization methods, mostly designed for circuit emulation over ATM [7], can be classified based on two major criteria. Methods that require common reference clock, also called the system clock, to facilitate recovery of the service clock are known as the *synchronous* methods. In contrast, *asynchronous* methods do not require such a reference. From the perspective of how the clock indications are conveyed between the master and the slave node, methods can be explicit (e.g., timestamp based) or implicit (e.g., deduced from the packet arrivals). Synchronous residual timestamp (SRTS) method [8], widely used in ATM, is an example of a synchronous clocking scheme with explicit clock indications. It communicates the difference between the system clock and the master service clock to the slave in form of a timestamp, which is used at the slave side to reconstruct the service clock. This method dominated ATM circuit emulation deployments, mainly because of its performance and simplicity. Since almost all ATM networks were implemented over SONET, the common reference was readily available. In Ethernet, this assumption rarely true, making SRTS impractical.[2] The main property of the algorithm we construct in this paper is that it does not require a system clock, while still achieving previously stated performance objective.

An alternative to SRTS, also used in ATM, is the adaptive clocking. This method is an example of asynchronous clocking scheme with implicit clock indications. Namely, cell interarrival times are averaged over a long period, either by direct arrival time measurements or by controlling the local clock frequency to maintain constant receive buffer level. Past experience from commercial ATM deployments has shown clear superiority of SRTS method over adaptive clocking. However, the ATM community had little incentive to improve the adaptive clocking algorithms, mainly because of the availability of the system clock from underlying SONET infrastructure. From this perspective, our proposed algorithm represents an improvement to the variety of adaptive clocking algorithms that has become necessary with the proliferation of (asynchronous) Ethernet MANs.

In [9], the authors proposed to synthesize the reference clock without using centralized reference source, but instead synchronizing internal node clocks among each other and then implementing the SRTS on the top of it to extract the service clock. Clock indications are represented as tick events exchanged among neighboring nodes. To limit the delay variance, the authors have proposed to originate and terminate clock indications on each link either through SONET overheads (not applicable to Ethernet) or by using special characters in the 8B10B line code (applicable to Ethernet). Eliminating the need for centralized reference clock source would be the main benefit for Ethernet, but hop-to-hop synchronization would require augmenting each switch in the network with the proposed algorithm, which may not be practical. Our objective is to construct an algorithm that minimizes the set of requirements imposed on the Ethernet equipment inside the network. The only two requirements we impose are that the packets carrying TDM payload be assigned higher priority than the background data packets and that the number of hops between two TDM adapter elements be limited. The first requirement is a reasonable QoS capability that many carrier-class Ethernet switches already implement, while the second requirement is always satisfied in a typical MAN.

An example of an asynchronous clocking system with explicit clock indications that is architecturally suitable for a packet network has been reported in [10]. This system recovers the clock from received timestamps by assuming linear dependency between the master and the slave clock and estimating coefficients of a linear function that minimizes the mean square error over some predefined time window. The slave clock is then adjusted using the estimated linear dependency. The results reported in [10] have shown the ability to reconstruct the master clock with jitter performance that is 10 times better than the network jitter. As we shall show in Section III-B, simulating a realistic model of an Ethernet MAN results in the packet arrival jitter of $\pm$ 20 μs or more depending on the number of hops and the packet size. A factor of 10 improvement would result in the recovered clock jitter that is still too high with respect to the requirements set by the standards.

In the remaining sections of this paper, we present a novel algorithm that is asynchronous and thus suitable for TDM transport over a pure packet infrastructure with no common reference clock, but recovers the service clock with performance superior to all asynchronous methods known to us. The synchronization is performed in two steps. The first step uses implicit clock indications, namely packet interarrival times, that converges fast but has limited accuracy. In the second step, the algorithm switches to explicit clock indications (i.e., timestamps) and completes the synchronization with high accuracy.

II. THE ALGORITHM

In this section, we introduce the algorithm used to synchronize the slave clock to a master. Instead of presenting an entire system at once, we first discuss each element, and gradually construct the complete algorithm by adding new blocks as we explain their function.

A. Holdover Loop

The first component of the algorithm is a digitally controlled frequency locked loop (FLL) shown in Fig. 2, here referred to

[1]Stratum-3 requires that during the first 24 hours the clock may not drift by more than 3.7×10^{-7} relative to the last synchronized frequency, while any deviation for any reason may not be higher than 4.6 ppm of the nominal frequency.

[2]If a central office has access to common system clock from some other (legacy) network or GPS system, the use of SRTS in Ethernet MAN is a viable option. However, such an assumption not universal.

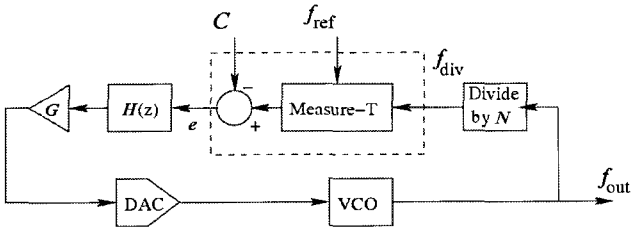

Fig. 2. The holdover loop.

as the holdover loop. It is used for two purposes: (1) To maintain the last known master frequency if the reference clock indications are lost, and (2) to enable linear control of the output frequency using an offset calculated by other components in the system. This FLL is unlike classical digital PLL circuits [11] or fractional-N frequency synthesizers [12]. Rather than determining the output frequency by multiplying the reference with control input (a non linear operation), the input to the holdover loop is an *additive* numerical offset from the nominal frequency. This design provides convenient method for linearly controlling the output frequency, which simplifies implementation of other components of the system, while still meeting performance requirements.

The system consists of a frequency divider, frequency measurement block, loop filter ($H(z)$), linear gain/attenuator (G), D/A converter (DAC) and voltage-controlled oscillator (VCO). The output signal frequency is divided by N and fed into the measurement block, labeled Measure-T in the figure. This block uses a local reference (f_{ref}) to measure the period of the divider output. A control signal (C) is subtracted from the output of the measurement block to yield the period error.

Assuming good stability of the reference frequency over a short measurement interval, the period error can be written as

$$e[n] \approx \left\lfloor \frac{N f_{\text{ref}}[n]}{f_{\text{out}}[n]} \right\rfloor - C[n]. \tag{1}$$

The units of $e[n]$ are clock ticks, where one tick represents the nominal period of a reference clock. We define the control signal as

$$C[n] = \frac{N f_{r0}}{f_0 + x[n]}, \tag{2}$$

where f_{r0} and f_0 are the nominal reference frequency and nominal VCO center frequency, respectively, which are known to the system. The signal $x[n]$ is a desired deviation from the nominal frequency. Neglecting the effects of the floor operator in (1) and combining it with (2) yields

$$e[n] = \frac{N f_{r0}\left[\frac{f_{\text{ref}}}{f_{r0}}(f_0 + x[n]) - f_{\text{out}}[n]\right]}{f_{\text{out}}[n](f_0 + x[n])}. \tag{3}$$

Since the system output is typically a small deviation from the nominal VCO center frequency f_0, the denominator of (3) can be approximated as f_0^2. Further, defining the relative error of the reference frequency as $\delta_{\text{ref}} = (f_{\text{ref}} - f_{r0})/f_{r0}$, the equation can be written as

$$e[n] = \frac{N f_{r0}(1+\delta_{\text{ref}})}{f_0^2}\left(f_0 + x[n] - \frac{1}{1+\delta_{\text{ref}}} f_{\text{out}}[n]\right). \tag{4}$$

Finally, taking a geometric expansion of $\frac{1}{1+\delta_{\text{ref}}}$ and neglecting the higher order terms yields

$$e[n] = \frac{N f_{r0}(1+\delta_{\text{ref}})}{f_0^2}\left\{x[n] - \left[f_{\text{out}}[n](1-\delta_{\text{ref}}) - f_0\right]\right\}. \tag{5}$$

Defining the system output $y[n] = f_{\text{out}}[n] - f_0$ as an *offset* from nominal VCO center frequency, we observe that the module encircled with the dashed line in Fig. 2 is a period comparator followed by a linear gain, which can be written as

$$A = G K_v \frac{N f_{r0}(1+\delta_{\text{ref}})}{f_0^2}, \tag{6}$$

where K_v is the gain of an ideal VCO and G is the linear attenuation/gain that must be designed, along with the filter transfer function $H(z)$, to provide good steady-state accuracy, fast response, and stability. For $H(z)$, we use a first-order filter with a pole at $z = 1$, which can provide zero steady-state error [13]

$$H(z) = \frac{1 - d \cdot z^{-1}}{1 - z^{-1}}. \tag{7}$$

The tunable parameters are the gain (G) and the location of the zero in the loop filter (d). We have found that $d = 0.05$ and G chosen such that the loop gain $A = 1$ are values that yield slight underdamping, but provide a good tradeoff between the response time and the overshoot. Analysis that lead us to these values is presented in Section III.

B. Open Loop Filter

The holdover loop is capable of maintaining and tracking the frequency encoded by an input signal with a precision determined by the accuracy and frequency of the reference clock. We now present the system components that determine the desired output frequency. Our algorithm uses two independent mechanisms for this purpose. The first mechanism, presented in this section, uses low-pass filtering of measured packet interarrival times. The second mechanism, which is presented in Section II-C, uses timestamps to fine-tune the frequency set by the first method.

For the first method, we observe that the master TDM network injects packets into the asynchronous network at a rate proportional to the master-clock. Therefore, packet interarrival times can convey master clock information to the slave. However, as packets pass through an asynchronous network, their measured interarrival times at the slave experience two kinds of distortion.

The first and dominant form of distortion is variable delay due to queuing. The amount and distribution of this delay depend on network load and QoS parameters. The second (and less visible) type of jitter is caused by the asynchronous design of typical equipment used in Ethernet networks. Each line card in Ethernet equipment uses an independent clock source and the relative tolerance among these clocks will be reflected in the packet interarrival times, even if no queuing delay has been experienced.

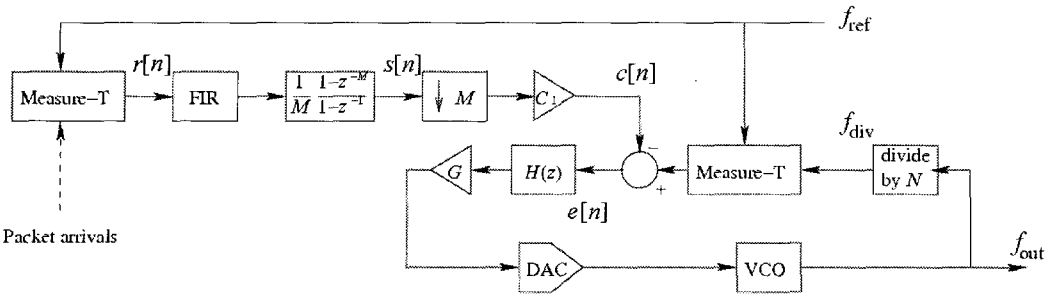

Fig. 3. The open-loop filter combined with the holdover loop.

In the analysis presented in this paper, we consider only queuing distortion and neglect the second effect.

The effects of jitter can be viewed as the signal processing equivalent of noise imposed on a DC signal. If data are transmitted over a *synchronous* network and interarrival times were plotted as a function of time, the result would be a DC signal. If packets experienced variable delay, this signal would still have the same DC component but with an *additive* zero-mean noise component. If the noise can be removed from the signal using a low-pass filter, the original DC signal (i.e., the constant interarrival time) could be extracted.

The AAL-1 adaptive clocking mechanisms have used this technique [8], but little attention has been paid to the design of a filter for best clock extraction performance. In some published work it has been (incorrectly) suggested that the filter design task is trivial because the network jitter has only the high-frequency component [14]. As we show in Section III-B, while zero mean, the noise from network jitter has very low frequency components and an extremely narrow-band filter is necessary to eliminate it.

Fig. 3 shows the system after combining the open-loop filter with the holdover loop. The interarrival times are measured using the Measure-T block described in Section II-A. The Measure-T block on the left side of the figure outputs the number of reference clock ticks that occur between two packet arrivals. The output is equivalent to the *normalized* interarrival time represented as a fixed-point unsigned number with all bits representing fractional binary digits. Therefore,

$$r[n] = \frac{T[n] f_{\text{ref}}}{2^b}, \tag{8}$$

where $T[n]$ is the interarrival time (in seconds) of n-th packet, f_{ref} is the reference clock frequency and b is the number of bits in the counter. The all-ones output corresponds to $r[n] = 2^{b-1}/2^b \approx 1$, representing the maximum measurable interarrival time. The number of bits needed to represent the measurements is function of the average interarrival time, network jitter and the reference frequency.

The measured interarrival times are passed through a low-pass filter implemented as an FIR filter, followed by a moving average filter. The two filters remove the high frequency noise, which represents the network jitter. The signal is downsampled and normalized using a constant gain C_1. The length of the moving average filter equals the downsample rate, so the output of the decimator represents the *mean* of all samples being dropped. The motivation for the moving average is to construct a downsampled signal by using the information contained in all dropped samples.

To provide proper input to the holdover loop, the constant gain C_1 must be determined, such that the output of the decimator approximates the (2). From (8), we have

$$s[n] = \frac{f_{\text{ref}}}{2^b}(T[n] * h[n]), \tag{9}$$

where $h[n]$ is the impulse response of the FIR and the moving average filter combined, and the symbol $*$ signifies convolution. Based on Fig. 7, we assume that applying a low-pass filter to a signal composed of interarrival times $T[n]$, results in an *estimate* of departure times at the master side (designated as $\tilde{T}_m$). Departure times are inversely proportional to the master clock frequency (f_m) and directly proportional to the TDM frame length L_{frame} (193 bits for T1 interface). Therefore,

$$\tilde{T}_m[n] = T[n] * h[n] \approx \frac{L_{\text{frame}}}{f_m[n]}. \tag{10}$$

The estimated master frequency is the sum of the nominal frequency and the estimated offset: $\tilde{f}_m = f_0 + \tilde{x}[n]$. Applying this definition to (10) and combining it with (2) and (9) yields

$$C[n] = \frac{N2^b}{L_{\text{frame}}} \cdot \frac{f_{r0}}{f_{\text{ref}}} s[n] \approx \frac{N2^b}{L_{\text{frame}}} s[n]. \tag{11}$$

The last approximation assumes that the local reference clock does not deviate much from its nominal value. The effects of the local reference precision are discussed in more detail in Section III-D. Finally, we conclude that the open-loop gain should be set to the following value

$$C_1 = \frac{N2^b}{L_{\text{frame}}}. \tag{12}$$

Following successful initial estimate of the master frequency using the open loop filter, the recovered clock is "fine tuned" using timestamped packets.

C. The NTP Loop

The open-loop filter alone cannot lock the clock to the master with the necessary accuracy, but it can provide an initial estimate. After this, the timestamp-based mechanism takes over and completes the locking process. We use a variant of the network time protocol (NTP) [15], [16] and its timestamps and also adopt the notation of NTP in this description.

To date, NTP has principally been used on the Internet where propagation delays and jitter are large and the clock sources (i.e., computer clocks) are operating with poor granularity, different accuracy, and potentially large initial offset [17]. Properties of the circuit emulation application in MAN environment allow for several simplifications of the standard NTP. First, the clock distribution is strictly hierarchical and the clock sources are considered reliable and accurate. This eliminates any requirement to implement the clock selection and filtering parts of the NTP algorithm (except for possible switchover to a backup clock source in case of a failure). Second, in the circuit emulation application, the goal is to synchronize clock frequency, not the absolute time value. Furthermore, if timestamps are exchanged in-band, they can be deduced from the frame sequence number because the time when the master sends the response can be made deterministic. Fourth, the propagation delay and jitter in a MAN environment are lower than on the Internet and the initial clock offset can be made very small, as determined by the open-loop filter. Therefore, the initial state for the NTP algorithm is close to the target frequency, allowing fast convergence. Finally, NTP adjusts the loop filter bandwidth dynamically [18] as the slave clock converges to the master. Since our algorithm implements two loops inside each other (the holdover loop and the NTP loop), the order of the resulting system is higher than if using NTP alone, which makes the design of an adaptive filter more difficult. Therefore, we opted for using a single-pole, non-adaptive filter in the NTP loop to avoid any undesirable effect on system stability.

The NTP loop is combined with the open-loop filter and the holdover loop to produce the complete system shown in Fig. 4. With proper selection of the time when the the NTP loop dominates the control, the output converges quickly and with high accuracy. The input for the holdover loop is a combination of the open-loop filter output and the NTP protocol. The weight factor w determines the relative contributions of the two mechanisms. Initially, with $w = 0$, there is no feedback from NTP, and the system behaves as described in Section II-B. Over time, the weight factor is gradually changed to 1, giving full control to NTP. Our implementation, uses a linear ramp for w that reaches 1 after 50 seconds, but other functions, such as raised cosine, may also be used.

When the transition period is finished, the weight factor becomes 1 and the system operates as a dual loop. The inner loop operates as described in Section II-A and the outer loop is the NTP protocol.

During the operation four timestamps are exchanged between the slave and the master. Initially the slave sends out the request for synchronization and records the time when the request has departed the slave (T_1). Other timestamps are recorded when the request reaches the master (T_2), when the response departs the master (T_3) and reaches the slave (T_4). Timestamps T_2 and T_3 are communicated to the slave as part of the response. The process repeats periodically. At each timestamp exchange, the following value is calculated

$$\theta = \frac{T_2 - T_1 + T_3 - T_4}{2}. \tag{13}$$

Assuming that at the beginning of n-th timestamp exchange the absolute time at the slave is $T_1 = x$ (in unit ticks with reference to the recovered frequency f_{out}) and that the absolute time at the master is offset by Δ from the slave, the timestamp values are as follows

$$\begin{aligned} T_1 &= x, \\ T_2 &= x + \Delta + \int_0^{t_{p1}} f_m(t)dt, \\ T_3 &= x + \Delta + \int_0^{\tau+t_{p1}} f_m(t)dt, \\ T_4 &= x + \Delta + \int_0^{\tau+t_{p1}+t_{p2}} f_{\text{out}}(t)dt, \end{aligned} \tag{14}$$

where t_{p1} and t_{p2} are the propagation times from the slave to the master and from the master to the slave, respectively.

Combining (13) and (14), followed by lengthy, but straightforward algebraic manipulations yields

$$\theta[n] = \Delta + \frac{1}{2}\int_0^{\tau} [f_m(t) - f_{\text{out}}(t)]dt + \phi_n[n], \tag{15}$$

where $\phi_n[n]$ represents all the remaining terms under the integral after the equation has been rearranged to read as (15). The first term Δ is the absolute time offset that can be made relatively small by initially synchronizing the absolute time at both sides before attempting to synchronize the frequency. Typically, the time offset can be maintained at the level comparable with the propagation delay or better [17]. The second term is the phase error, a desired component that depends on the instantaneous master and slave clock frequencies only and can be made sufficiently larger than unwanted terms by increasing the period between receiving the synchronization request and sending the response (τ). The third term is the phase noise, mostly due to asymmetric links, variable propagation times and frequency instabilities on both the slave and the master. The resulting sequence $\theta[n]$ is passed through the loop filter resulting in the input for the holdover loop.

For successful convergence, the phase error must be significantly larger than the noise component. This result can be achieved if the time between receiving the NTP request at the master side and sending an NTP response is deterministic and significantly larger than the propagation time. In a MAN environment with short delays and moderate jitter, this requirement can be easily met.

For the loop filter, we use a single-pole IIR filter of the form

$$H_1(z) = \frac{1-\alpha}{1-\alpha z^{-1}}. \tag{16}$$

Unlike the FLL in the holdover loop that requires an integrator (pole at $z = 1$) to eliminate the steady state error, the PLL model of NTP can operate without an integrator, as phase calculation (tick accumulation on the master and the slave side) already performs this function. We have empirically determined that the dual loop behaves best with parameters set to $\alpha = 0.1$ and $G_1 = 0.08$.

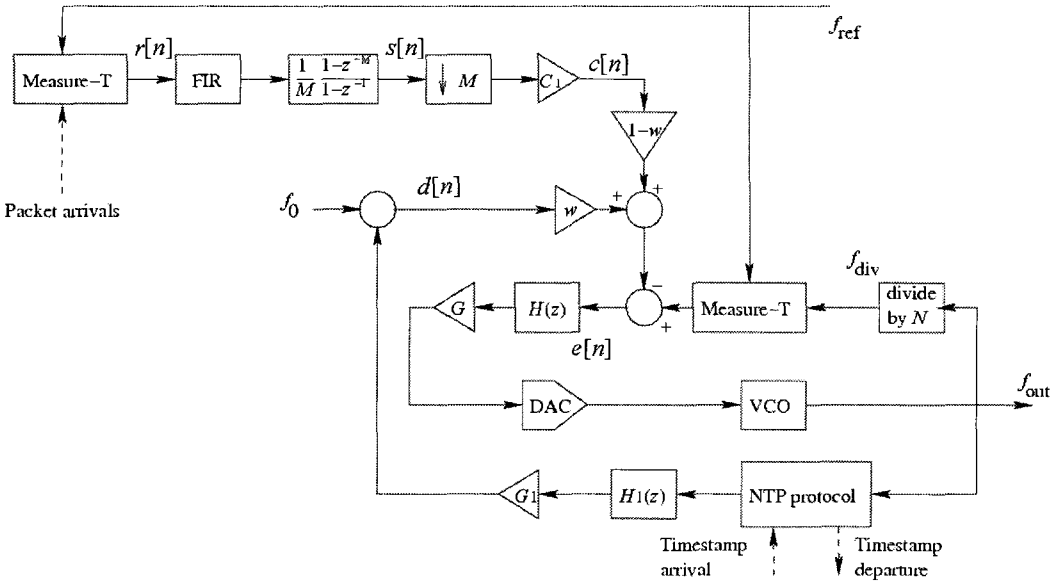

Fig. 4. The full algorithm: Hold-over loop, open loop filter, and the NTP.

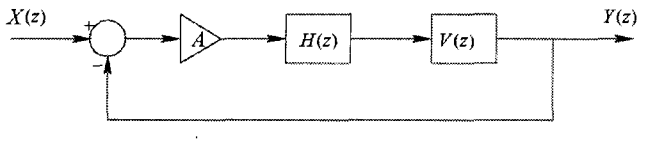

Fig. 5. A linearized model of the holdover loop.

III. SYSTEM ANALYSIS AND SIMULATIONS

To determine the parameters of the system needed to meet the requirements, we conducted simulations and conducted an analytical study. This section presents the results and outlines the lessons learned.

A. Holdover Loop Analysis

As suggested in Section II-A, the holdover loop must provide accurate steady state output, fast response and it must be stable. Starting from (5), recalling the definition of the system output $y[n] = f_{\text{out}}[n] - f_0$ and neglecting the $(1 - \delta_{\text{ref}})$ term, we conclude that the holdover loop can be approximated as the discrete-time linear system shown in Fig. 5.

The system is only a rough approximation valid in the vicinity of the operating point, but still useful enough to determine the system parameters, namely the loop gain A and the location of pole d in the loop filter. The sampling rate of such a system is variable and determined by the output frequency f_{out} and the division factor N. Therefore, only discrete domain analysis can produce meaningful results.

Function $V(z)$ is an artifact of approximations done to construct the linearized model and for the purpose of this analysis is attributed to the VCO dynamics. The goal of the analysis is to first identify the $V(z)$ and then determine the subset of space defined by variables A and d for which the system is stable. Having determined the stability region, we can vary the parameters to further optimize the system response.

Due to the nature of approximations we did to model the system, $V(z)$ cannot be represented in closed form. Instead, we model individual components of the system shown in Fig. 2 and simulate the impulse response. We then use the impulse response to identify the $V(z)$ and determine the stability region analytically. Finally, we test the system behavior at various points inside the stability region as well as at its boundaries. If we observe reasonable similarity in behavior between the simulated system and the analytical linear model, we conclude that the model is valid.

We observed, through experimentation with the simulation model, that using a two-pole, single-zero function for $V(z)$ appropriately models the system in the region of interest. Fig. 6 shows the stability region of the system for which all parameters except d and A have been fixed ($N = 1544000$, 16-bit D/A converter, $f_{\text{ref}} = 311.04$ MHz, 50 ppm VCO deviation, $V(z) = \frac{z-0.01}{(z-0.05)(z+0.042)}$). The location of the operating point at $d = 0.05$ and $A = 1$, provides sufficient margin to ensure the loop stability over wide parameter tolerance range.

B. Open Loop Filter Requirements

As indicated in Section II-B, high packet arrival jitter makes it hard to accurately estimate the master clock based on packet arrivals only. The goal of the open-loop filter is to provide the initial estimate with accuracy as high as possible without imposing unrealistic or hardly implementable requirements on the filter structure.

We designed the filter by running network simulations and constructing the power spectrum of measured packet interarrival times. At the master side, packets were generated by a simulated T1 constant bit rate (CBR) feed that is transported over a 5-hop network with selectable background traffic. We performed spectral analysis of the interarrival times using a 2048-point FFT and a 2048-point Kaiser window with $\beta = 0.25$ [19]. The result is shown in Fig. 7, which compares the power spectrum of the

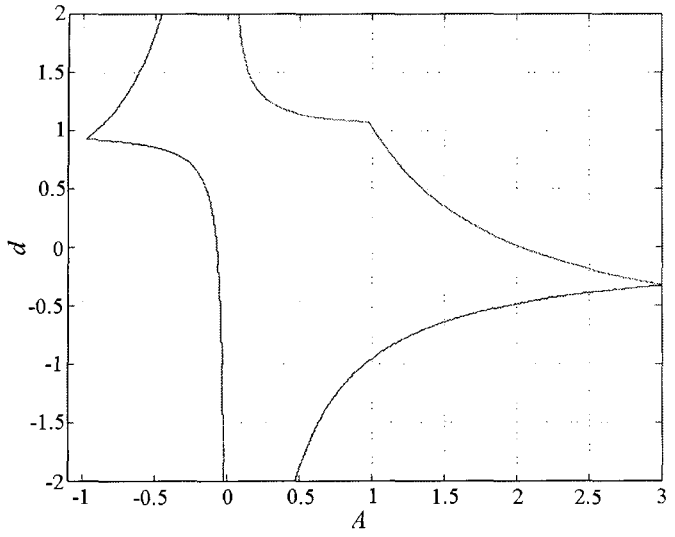

Fig. 6. Holdover loop stability region.

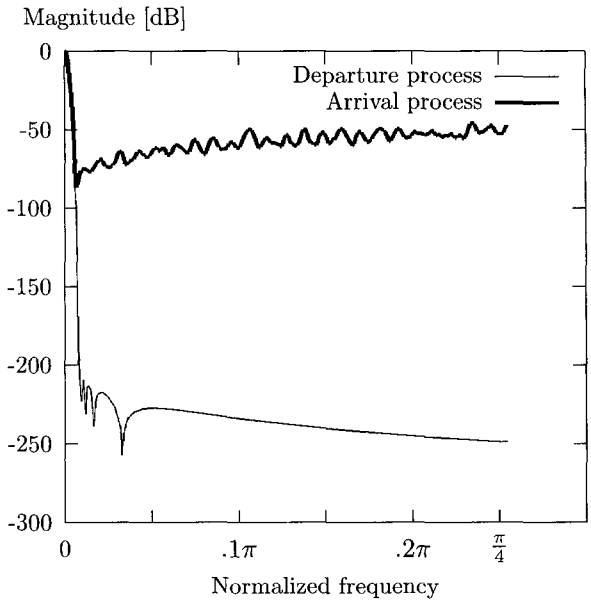

Fig. 7. Power spectrum the arrival/departure process.

arrivals at the slave side (heavy line) to the power spectrum of the departures at the master side (thin line).

We ran simulations using the `ns-2` [20] tool with the following setup: Each link in the simulation had a 1 Gb/s capacity and was 75% loaded with background traffic. The T1 source and sink were located at nodes 1 and 5, respectively. The T1 source generated a 64-byte packet (T1 frame plus overhead) every 125 μs. For the background traffic, total of 150 independent exponential sources with identical statistics were distributed evenly across the five nodes (30 per node). At each node, 20% of the background traffic was removed and 20% of new background traffic was injected into the network, to maintain a 75% load. This ensured that the background traffic at each node contained a mix of packets that had passed through only one hop and packets that had experienced queuing at multiple hops. The maximum packet size for the background traffic was 1500 bytes.

Class-based queuing (CBQ) [21] was used with two classes: T1 traffic and the background traffic. Twenty percent of the total link bandwidth was allocated for the T1 class and borrowing unused bandwidth from any class was permitted. The T1 traffic had strict priority over the background traffic. The main source of jitter in this simulation is the non-preemptive nature of queuing. Although a T1 packet arriving at the node has priority, it will not interrupt a transmission in progress. Therefore, the maximum packet size of the background traffic determines the maximum delay jitter, which is $\pm$ 20 μs.

Fig. 7 shows the noise spectrum starting at very low frequencies, implying that an extremely narrowband filter is needed to extract the original DC signal. Our experiments indicate that a 2048-tap FIR filter with cut-off frequency of $10^{-4} \cdot \pi$ is necessary to recover the clock with accuracy better than 2 ppm.

C. Performance of an Open-Loop System

Fig. 8 shows the relative error ($\frac{f_{\text{out}} - f_m}{f_m}$) of the clock synthesized by filtering the packet interarrival times. Secondary effects such as finite precision arithmetic, non-ideal oscillators, etc., have all been modeled in the simulation and set as follows: Nominal VCO center frequency and master frequency are set to 1.544 MHz (T1 link). Initial offset of the VCO is 10 ppm from the nominal frequency. The systematic drift of the VCO is 10 ppm per day.[3] The VCO deviation is 50 ppm and the VCO is linear in this region. The local reference clock is 311.04 MHz and conforms to Stratum-3 compatibility requirements. The worst case is assumed for other parameters of the reference clock including maximum allowed drift, maximum offset from the nominal frequency, etc. Sampling rate for the holdover loop is 1s (which translates to a division factor of $N = 1544000$ and a downsample rate of $M = 8000$). The FIR filter uses 16-bit values for the input signal and filter coefficients and the cut-off frequency has been set to $10^{-4} \cdot \pi$. Overflow is avoided by using a 32-bit fixed-point multiply-and-accumulate (MAC[4]) unit and ensuring that the sum of filter coefficients equals 1 (i.e., unit DC gain filter). The holdover loop filter and the gain C_1 are implemented using floating-point arithmetic and the Measure-T block in the holdover loop is implemented using 32-bit counters. The D/A converter is 16-bit. These parameters were chosen to reflect a realistic system implemented with low-cost components.

The results show that it is possible to approach the master clock frequency with accuracy better than 2 ppm using only measured packet interarrival times. Although this accuracy is not sufficient, it demonstrates the tolerance of an extremely high noise level with jitter of 20 μs compared to 125 μs nominal interarrival period (almost five orders of magnitude jitter improvement). Any residual jitter is due to noise that cannot be filtered with a practical low-pass filter using finite precision arithmetic and is removed using the timestamp-based method.

[3] Such a large drift is unlikely to occur in practice, but we used it in simulations to stress the algorithm's ability to correct for large errors.

[4] The abbreviation "MAC" is often used for both multiply-and-accumulate and medium access control for Ethernet. It should be clear from the context which is intended for a particular occurrence.

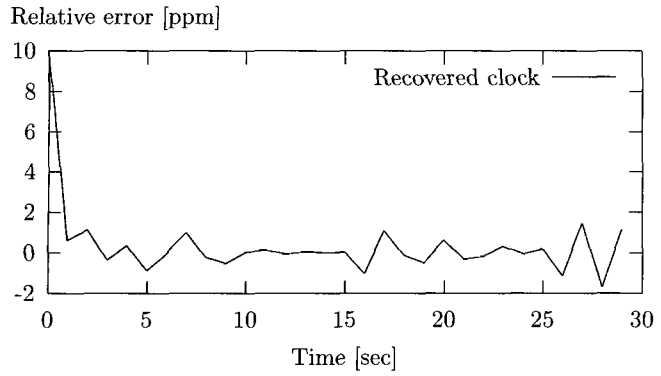

Fig. 8. Synthesized frequency using the open-loop filter and the holdover loop.

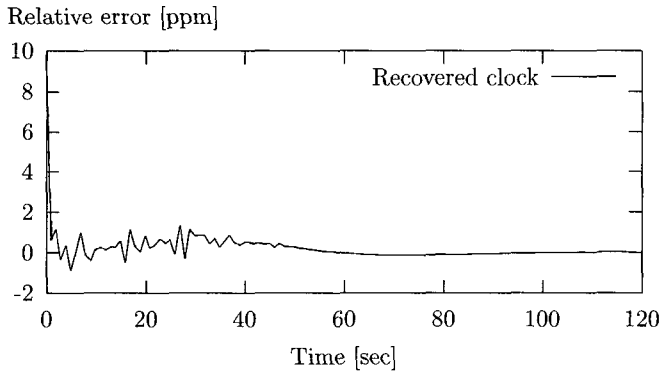

Fig. 9. Synthesized frequency using the full algorithm.

D. Effects of the Local Reference

It is important to note that using the same reference source in both Measure-T blocks shown in Fig. 3 cancels the effects of reference instability. This can be analytically demonstrated by combining (1) and (11). Neglecting the effects of rounding and noting that the estimated master frequency is by definition $\tilde{f}_m = L_{\text{frame}}/\tilde{T}_m$ yields

$$e[n] = \frac{N f_r[n]}{f_m f_{\text{out}}[n]} (\tilde{f}_m[n] - f_{\text{out}}[n]). \tag{17}$$

The equation indicates that the frequency error inside the closed-loop system does not depend on the reference frequency. The only effect of non-ideal reference frequency is modulation of the loop gain, which does not affect the system convergence as long as the stability is not compromised.

Our system is designed to remain stable for all loop gains in the interval of $0 < A < 2$, with $A = 1$ being the operating point. With such a wide stability region, a tolerance of 4.6 ppm (per Stratum-3 requirements) is safe for the system stability. The loop gain is also modulated by the output frequency as it converges to the master frequency, but this modulation is bound by the VCO deviation, which is still within the safe stability range. This modulation causes slightly slower convergence than that predicted by a linear model.

The result of (17) is also intuitive, as any increase in reference frequency will cause $r[n]$ to increase and, in turn, cause the input to the loop ($c[n]$) to increase. On the other hand, an increase in reference frequency will cause an increase in the measured period of f_{div}, which provides negative feedback in the error $e[n]$.

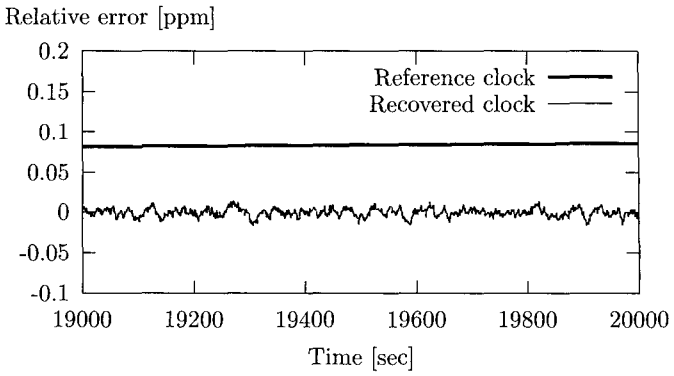

Fig. 10. Clock drift after 20000 seconds of simulation.

E. Performance of a Dual-Loop System

Fig. 9 shows the first 120 seconds of simulation, giving NTP full control after 50 seconds and reducing the maximum error to under 10^{-8}, while the average error is at 10^{-11} order of magnitude. The open-loop action can be observed in the beginning and the graph is similar to that of Fig. 8.

The result shown in the figure uses the same network setup as discussed in Section III-B. For NTP, it is assumed that a new set of NTP timestamps is exchanged immediately following the clock update at the slave side. The time between receiving the request and sending the response at the master side is 1 s.

The tick counter on the slave side counts the cycles of the f_{out} signal and does not directly use the local reference. The reference clock variation will initially cause the inner loop to modify the output frequency, but the difference will be reflected in the value of the timestamps and corrected on the next frequency update. In this way, the outer loop corrects any wander caused by the inner loop and allows the output to fully lock to the master frequency. This behavior is illustrated in Fig. 10, which shows the relative frequency error of the local reference (thick line) compared to the recovered clock.

The figure shows the state after 20000 seconds, assuming that the reference clock started with zero error and drifted over time at the rate of 0.37 ppm/day (Stratum-3). After 20000 seconds the reference clock error is slightly less than 0.1 ppm and continues to drift, while the recovered clock is still oscillating around the zero mean error.

IV. CONFORMANCE TESTS

We have presented an algorithm and simulation results that show a new method for high-accuracy synchronization. In locked mode, the maximum relative error between the local clock and the master clock was approximately 10^{-8} (0.01ppm) at any time, and 10^{-11} averaged over a long period. In this section, we discuss the performance of the algorithm compared to the performance requirements set by standards for synchronous networks [4]. Of specific interest is the clock wander perfor-

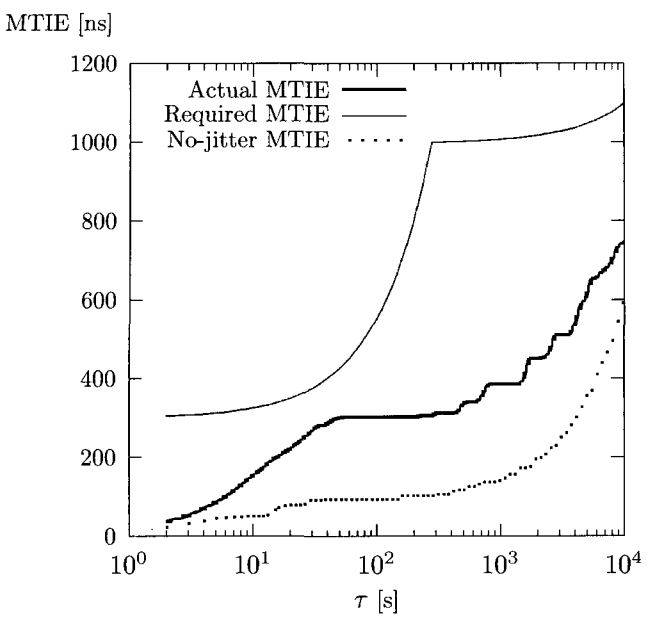

Fig. 11. Maximum time interval error (MTIE).

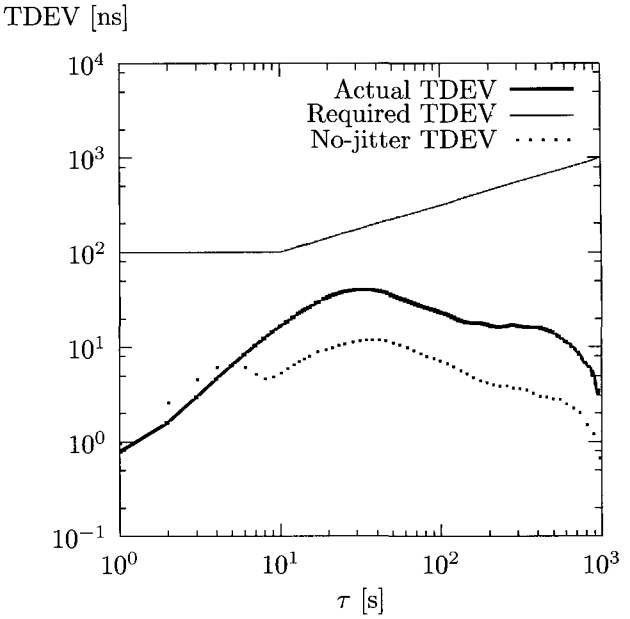

Fig. 12. Time deviation (TDEV).

mance, which is characterized by maximum time interval error (MTIE) and by time deviation (TDEV).

MTIE is a measure of buffer overflow or starvation if data are injected into the network synchronously with an ideal clock and removed from the network synchronously with the clock under test. It is defined as the maximum difference between the maximum and the minimum time error function within any time window $\tau = n\tau_0$, also referred to as the observation time over n samples where τ_0 is the sampling period

$$MTIE(n\tau_0) = \max_{1\le k\le N-n}\left[\max_{k\le i\le k+n} x_i - \min_{k\le i\le k+n} x_i\right]. \quad (18)$$

Here, the time error function $x_i = x(i\tau_0)$ is the difference between the time measured with reference to an ideal clock and the same time measured with reference to the clock under test.

The time deviation (TDEV) as a function of an integration period $\tau = n\tau_0$ is defined as

$$TDEV(n\tau_0) = \sqrt{\frac{1}{6n^2(N-3n+1)}\sum_{j=1}^{N-3n+1} A_{j,n}}$$
$$A_{j,n} = \sum_{i=j}^{j+n-1} x_{i+2n} - 2x_{i+n} + x_i, \quad (19)$$

where N is the range of integration periods over which TDEV is calculated as $n \in \{1, 2, 3, \cdots, \lfloor N/3 \rfloor\}$. Similar to MTIE, time deviation can also be interpreted as a buffer level when the data are clocked in synchronously to the reference and clocked out using the slave clock. While MTIE presented a static measure (worst case level within an observation time), TDEV provides some insight into dynamic characteristics. Lower MTIE and TDEV indicate better performance.

We have measured the MTIE and TDEV for our algorithm using observation times between 2 and 10000 s. Because the algorithm updates the VCO once per second, clock wander performance over time intervals shorter than 1 second cannot be guaranteed with the algorithm. Instead, required short-term performance is ensured by selecting the VCO device with appropriate wander specification. Low-wander VCO requirements affect only short observation times and we do not anticipate a problem in practice.

Figs. 11 and 12 compare the observed performance (thick lines) to the requirements of ITU-T G.824 [4] standard (thin lines). The graph shows that the emulated T1 interface conforms to the standard over the observation times we tested.

The dotted lines show the performance from a simulation of a zero-delay network. This curve illustrates the sensitivity of the algorithm to network delay and delay variation. The zero delay is applied only to the NTP loop, whereas the open loop filter was still subjected to the full range of delay and jitter conditions. The penalty ranges between 50% and a factor of 3, suggesting that a network with small delays and delay variation is required.

With highest queuing priority for TDM traffic, delay variation is determined by the number of hops and the packet size distribution of the background traffic. The mean delay is determined by the number of hops and geographical distance. Our simulation environment reflects a realistic metropolitan-size gigabit Ethernet network. The results suggest that TDM emulation is feasible for MANs, but its utility for core networks (e.g., a national backbone) must still be explored. Using 10 Gb/s Ethernet will improve the delay variation, yielding improved performance, but the mean delay is irreducible.

V. IMPLEMENTATION CONCERNS

An adapter supporting T1 transport over Ethernet is currently under construction in our lab and this section outlines some lessons learned from this effort. While the simulation discussed previously accurately models key parameters and includes many secondary effects (e.g., finite precision arithmetic, D/A conversion quantization, non ideal oscillators, etc.), there are additional side effects that are important in a real-world implementation.

A. Measuring the Interarrival Times

One of the foundations of the algorithm is measuring the event period with high precision (i.e., the Measure-T block). Measurements (in clock ticks) are interpreted by other blocks as a fixed-point representation of the signal. The local reference clock determines the measurement granularity, with a higher frequency giving more accurate results. The effect of using a finite frequency is analogous to quantization noise in signal processing systems. The simulations presented earlier model this effect and show that a local reference of approximately 300 MHz is sufficient to minimize the impact of quantization on performance. To avoid using a custom-built oscillator, we have chosen a 311.04 MHz reference frequency, which we achieve by using a standard 19.44 MHz Stratum-3 oscillator in cascade with an analog PLL circuit that boosts the frequency to 155.52 MHz. The measurement circuit operates on both edges of the clock, effectively achieving the desired 311.04 MHz reference.

In locked mode, the reference frequency error has two effects. First, it modulates the loop gain; second, it adds an offset to the output frequency whose relative magnitude equals the relative reference frequency error. The closed loop system ensures that none of these effects impact the accuracy of the output. They can only impact the system stability, which we prevent by ensuring that the system remains stable for all possible variations in reference frequency. In holdover mode, which is entered if the data link between the master and the slave adapter is broken or if packets are dropped for an extended period, the reference clock jitter and wander propagate to the output. The system maintains the last known master frequency by keeping the control input at the last valid update value. The choice of a Stratum-3 compatible reference ensures that the entire system is Stratum-3 compatible when operating in the holdover mode.

To maintain the accuracy of measurements, packet interarrival times must be measured close to the physical layer. Many Ethernet switching components pass packets through several independent clock domains before reaching a packet processing block where the arrival of a TDM packet can be recognized. To avoid these resynchronizations and the resulting accuracy loss, we measure the arrival times of all packets using a physical-layer circuit with byte-realignment function disabled. At the parallel interface side, we search for start-of-packet (SOP) characters and record the word in which the event appeared. We also record the amount of shift that would be necessary to realign the word. This information is sufficient to construct a pulse with a bit-granularity whose location in time is determined by the packet arrival time. Measuring the time between such pulses generates a train of interarrival time measurements for *all* traffic at the interface including corrupted and invalid packets. The measurement data are passed to a packet processor which adds them to an accumulator register until the packet header matches the pattern that designates the monitored T1 traffic. After the match, the value in the accumulator register is passed to the FIR filter and the accumulator is reset to zero.

B. Hardware/Software Partitioning

The next stage in the open-loop is a 2048-tap FIR filter, which executes on each T1 packet arrival (125 μs on the average). For 16-bit operands, the filter requires a 6 K buffer storage (2 K for filter coefficients and 4 K for samples) and a processor capable of executing one MAC operation every 122 ns.[5] The FIR filter (buffers and the MAC unit) is implemented in the same FPGA as the Measure-T block.

The moving average filter and the downsampling is also implemented in FPGA. Because the downsample rate matches the length of the moving average filter, a long buffer to store the latest M samples is not necessary. Instead, the decimator block and the moving average are merged into a single block containing an accumulator followed by a divider. The accumulator starts with a zero value and adds the incoming samples until M of them have arrived. The resulting value is divided by M, passed to the next block of the algorithm and the accumulator is reset. To avoid losing precision, a minimum of $b_{\mathrm{MAC}} + \log_2 M$ bits are used to implement the accumulator, where b_{MAC} is the width of the MAC unit output (32 bits in our implementation).

The accumulator/decimator circuit is the last stage implemented in the FPGA. The samples following this block arrive at a modest rate of 1 sample per second, which allows the rest of the algorithm to be implemented using a low-cost general purpose processor. Division by M is performed in software using floating point arithmetic. The performance requirements on the processor are modest at this point, enabling the use of floating-point arithmetic. In the holdover loop, the divide-by-N and the Measure-T blocks are implemented in FPGA, while all remaining blocks are executing on the processor. The Measure-T block is the same as the one used to measure packet interarrival times, except that a 32-bit number representation is used.

The choice of the processor and the operating system has mostly been determined by cost and availability. The prototype implementations uses a Motorola PowerQUICC II[6] processor [22] integrating a PowerPC core, communication co-processor and T1 and Ethernet interfaces. The operating system we have chosen is Linux kernel version 2.4.x. The clock extraction algorithm and other system management/housekeeping processes run on the PowerPC core in the user space. Packet encapsulation is a combination of co-processor microcode and a kernel process running on the core. The structure of the clock recovery process is simple: It reads data from the FPGA device driver and the NTP daemon, performs calculations following the data flow of Fig. 4 and writes the output to the device driver controlling the DAC.

C. VCO Selection

The clock recovery algorithm updates the VCO input once per second, on average. As shown in Section IV the algorithm can successfully meet the performance objectives for any type of offset, drift, or wander if the observation times are greater than the update rate. The only VCO requirement is that its deviation be greater than the maximum possible offset resulting from conditions during the equipment life cycle. These factors include accumulated drift at the end of equipment life, maximum frequency wander, initial center frequency offset, etc. For

[5]The coefficient symmetry that exists in low-pass, linear-phase, and FIR filters allows for storing only the first 1024 coefficients and executing the filter using 1024 MAC operations.

[6]PowerQUICC II is a trademark of Motorola, Inc.

observation times shorter than 1 second, the algorithm cannot correct errors caused by jitter and wander, and the system must rely on the VCO to stay within the required specifications.

VI. CONCLUSIONS AND FUTURE WORK

The ability to emulate synchronous services over an asynchronous network infrastructure is critical for supporting legacy telecom service using emerging MAN technologies such as Ethernet. Successful emulation allows lower-cost technologies to be used for high-bandwidth data transport while not requiring a separate infrastructure for older, but important applications.

Published work and the efforts in relevant standard organizations and consortiums, such as IETF [23] and metro Ethernet forum [24], have specified the synchronization problem, but there have been limited results leading to an adequate solution. Instead, the previous work has largely focused on packetization techniques, which we consider a straightforward engineering task rather than a research challenge. The major challenge for TDM emulation is meeting the clocking and synchronization requirements. We demonstrated that synchronization problem can be solved for services at useful speeds (T1 and E1). Improving this algorithm to meet more rigorous requirements for the OC-*X* hierarchy (thus enabling full emulation of SONET over Ethernet) is still unexplored. If achievable and cost-effective, such a capability would bring a radically new alternative for metro network deployments.

We are currently implementing a prototype system to transport multiple T1 links over an Ethernet network. The objective is to experimentally verify the correctness of the proposed algorithm beyond the simulations and to integrate the technology with other Ethernet equipment. For future work, we plan to explore (both experimentally and theoretically) the effect of different traffic patterns, network topologies and QoS to the clock recovery performance. We expect to develop further improvements to the algorithm in an effort to achieve successful SONET over Ethernet emulation. We further plan to explore the viability of implementing various TDM services, such as fractional-T1 and Centrex, in Ethernet environment. We believe this work will result in further elimination of TDM-centric equipment, reduction of network operating costs and smoother migration to fully packet-oriented MANs.

ACKNOWLEDGMENTS

The authors would like to thank Martin Carroll and Dennis Morgan as well as the anonymous reviewers for invaluable comments that contributed to better quality of this paper.

REFERENCES

[1] J. Babcock, "Sonet: A practical perspective," *Business Commun. Review*, vol. 20, no. 9, pp. 59–63, Sept. 1990.

[2] A. Chapman and H. T. Kung, "Enhancing transport networks with Internet protocols," *IEEE Commun. Mag.*, vol. 36, no. 5, pp. 100–101, May 1998.

[3] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Commun. of the ACM*, vol. 19, no. 5, pp. 395–404, July 1976.

[4] International Telecommunication Union (ITU), Telecommunication Standardization Sector, "The control of jitter and wonder within digital networks which are based on the 1544 kbits/s hierarchy," *ITU-T Recommendation G.824*, Mar. 2000.

[5] American National Standard Institute, "Synchronization interface standard," *ANSI T1.101-1994*, Feb. 1994.

[6] S. Bregni, "A historical perspective on telecommunications network synchronization," *IEEE Commun. Mag.*, vol. 36, no. 6, pp. 158–166, June 1998.

[7] International Telecommunications Union, "B-ISDN ATM adaptation layer specification: Type 1 AAL," *ITU-T I.363*, Aug. 1996.

[8] R. C. Lau and P. E. Fleischer, "Synchronous techniques for timing recovery in BISDN," *IEEE Trans. Commun.*, vol. 43, no. 234, pp. 1810–1818, Feb./Mar./Apr. 1995.

[9] C.-S. Li and Y. Ofek, "Distributed source-destination synchronization using inband clock distribution," *IEEE J. Select. Areas Commun.*, vol. 14, no. 1, pp. 153–161, Jan. 1996.

[10] R. Noro, M. Hamdi, and J. P. Hubaux, "Circuit emulation over IP networks," in *Proc. IFIP TC6 WG6.1 & WG6.4 / IEEE ComSoc TC on on Gigabit Networking 6-th International Workshop on Protocols for High Speed Networks VI*, Aug. 1999, pp. 187–201.

[11] W. C. Lindsey and C. M. Chie, "A survey of digital phase-locked loops," *Proc. IEEE*, vol. 69, no. 4, pp. 410–431, Apr. 1981.

[12] B. G. Goldberg, "Programmable fractional-N frequency synthesizer," *U.S. Patent 5,223,132*, Jan. 1992.

[13] C. L. Phillips and H. T. Nagle, *Digital Control Systems, Analysis and Design*, 2nd ed., Prentice Hall, 1990.

[14] H. M. Ahmed and M. G. Hluchyj, "ATM circuit emulation - a comparison of recent techniques," in *Proc. IEEE GLOBECOM'91*, 1991, pp. 370–374.

[15] D. L. Mills, "Improved algorithms for synchronizing computer network clocks," *IEEE/ACM Trans. Networking*, vol. 3, no. 3, pp. 245–254, June 1995.

[16] D. L. Mills, "Adaptive hybrid clock discipline algorithm for the network time protocol," *IEEE/ACM Trans. Networking*, vol. 6, no. 5, pp. 505–514, Oct. 1998.

[17] D. L. Mills, "On the accuracy and stability of clocks synchronized by the network time protocol in the Internet system," *ACM Comput. Commun. Review*, vol. 20, no. 1, pp. 65–75, Jan. 1990.

[18] D. L. Mills, "Modeling and analysis of computer network clocks," *Electrical Engineering Report 92-5-2*, University of Delaware, May 1992.

[19] A. V. Oppenheim and R. W. Shaffer, *Discrete-Time Signal Processing*, Prentice Hall, 1989.

[20] L. Breslau *et al.*, "Advances in network simulation," *Computer*, vol. 33, no. 5, pp. 59–67, May 2000.

[21] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Networking*, vol. 3, no. 4, pp. 365–386, Aug. 1995.

[22] Motorola, "PowerQUICC II integrated communications processor family," *Motorola Fact Sheet*, Rev. 3, 2001.

[23] Pseudo Wire Emulation Edge to Edge (pwe3), http://www.ietf.org/html.charters/pwe3-charter.html.

[24] Metro Ethernet Forum, http://www.metroethernetforum.org.

Ilija Hadžić received his Ph.D. degree from the University of Pennsylvania in 1999 and since then he has been with Bell Labs, Lucent Technologies in Murray Hill, New Jersey, USA. While at Penn., he was involved in DARPA funded research on Active Networks, where his main contribution was an FPGA-based platform called Programmable Protocol Processing Pipeline (P4), which was used to improve the performance of a traditional CPU-based AN nodes. During his career in Bell Labs, Ilija built many hardware and software systems that were used as platforms for research in network services, content distribution, and metro and access networks, as well as prototype systems for Lucent products. His current research interests are data networks in general, specialized hardware architectures for network processing, network processors, and applications of FPGAs in communication systems.

Edward S. Szurkowski is Director of Optical Data Systems Research at Bell Laboratories, Lucent Technologies in Murray Hill, New Jersey, USA. He is responsible for a range of research and forward-looking work programs in the areas of broadband access systems and metro optical networks supporting residential and business applications. Previously, he led programs in interactive television, electronic publishing, high-performance processors, and other networking & computing systems. Mr. Szurkowski joined Bell Labs in 1981 after receiving the Ph.D. in Electrical Engineering from the University of Delaware.