NETWORK TIME PROTOCOL PRECISION TIMESTAMPING SERVICE

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] Embodiments of the present invention generally relate to improving the time-offset accuracy between the transmitter and receiver paths over data networks and, more specifically, to network time protocol precision timestamping service.

Description of the Related Art

[0002] In many communications applications, each element in a network has its own internal clock running independently of the other clocks in the network. Where the networks are synchronous networks, such as in many telecommunications applications and in high speed wide area networks, these clocks must be synchronized to the server. In legacy telecommunications networks, network elements utilized "TDM" links, such as T1 and SONET, which are inherently capable of carrying timing information from server to client at the physical layer. However, it is generally agreed that next generation networks will be based on a packet-switched infrastructure and, therefore, packet-based methods for transporting timing information will be used. One of the methods for synchronizing clocks in packet-switched data networks is Network Time Protocol (NTP).

[0003] Commercial NTP servers typically employ highly accurate hardware based clocks, which are disciplined to the external standards. In turn, NTP clients send carefully crafted packets to NTP servers and analyze their replies in order to determine the offset of the client clock relative to the server clock. A typical packet contains four timestamps. The timestamps are designed to precisely time the transmit and receive paths of the client/server time packet interchange so that the roundtrip delay between the endpoints and the offset of the client clock may be calculated. However, despite the high accuracy of the hardware based clock at the NTP server, if there are any inaccuracies associated with the timestamping function in the NTP server, the NTP server introduces errors into the timestamps provided to

the client. Subsequently, these errors propagate to the client clock and compromise proper correction of the client clock.

[0004] Most of the inaccuracies within the timestamps created by the NTP server are introduced by various layers of software, including device drivers, network stacks, and the increasing use of non-real-time operating systems. This is illustrated in Figure 1, conceptually demonstrating a packet timestamping flow in a typical NTP system 100, according to prior art. Most of existing NTP servers include systems such as the NTP system 100. Typically, a timestamp in a packet is supposed to reflect the instant of the arrival or departure referenced to the physical layer interface. With reference to Figure 1, the physical layer is associated with an Ethernet PHY 110. In this case, the time-of-arrival refers to the instant the first (or the designated) bit of each of the incoming packets traverses the boundary between the external environment (not shown) and the Ethernet PHY 110. Likewise, the time-of-departure is meant to be the instant when the first (or the designated) bit of each of the outgoing packets crosses the said boundary between the Ethernet PHY 110 and the external environment.

[0005] As illustrated in Figure 1, when an incoming packet 102 arrives at the NTP system 100, it passes through several entities: the physical layer associated with the Ethernet PHY 110, an Ethernet Media Access Controller (MAC) 120, operating system (O/S) services 130, an NTP application 140. Some of these entities are physical (such as the Ethernet MAC 120) and some are logical (such as the O/S services 130). A path for the incoming packet 102 is shown with arrows 112, 122, and 132. As shown with an arrow 142, the NTP application 140 requests a timestamp from a hardware clock 150 to be recorded in the incoming packet 102 only after the incoming packet 102 traverses the Ethernet PHY 110, the Ethernet MAC 120, and the O/S services 130 and actually reaches the NTP application 140 (such a timestamp is referred to herein as a "receive timestamp"). Upon receiving a request from the NTP application 140, the hardware clock 150 sends current time value as the receive timestamp, illustrated as timestamps 144. Thus, there is a

delay, with a significant random component, between the actual time-of-arrival at the NTP system 100 and the time when the NTP application 140 actually reads the hardware clock 150 to ascertain the value for the receive timestamp.

[0006] A similar situation occurs for an outgoing packet 104 leaving the NTP system 100. The NTP application 140 reads current time value from the hardware clock 150 and inserts that value as a timestamp in the appropriate field within the outgoing packet 104 (such a timestamp is referred to herein as a "transmit timestamp"). After thetransmit timestamp is inserted, the NTP application 140 launches the outgoing packet 104 on a path shown by arrows 134, 124, and 114. Again, since the outgoing packet 104 must first traverse the O/S services 130, the Ethernet MAC 120, and the Ethernet PHY 110, there is a delay, with a significant random component, between the instant when the transmit timestamp was obtained from the hardware clock 150 and the actual time-of-departure from the NTP system 100. Consequently, inaccurate receive and transmit timestamps propagate from the NTP system 100 to the client clock and set the baseline error for time keeping and rate keeping at the client clock.

[0007] As the foregoing illustrates, what is needed in the art is an improved NTP timestamping method which could provide more precise and stable measurement of the transition of the NTP packets.

SUMMARY OF THE INVENTION

[0008] One embodiment of the present invention sets forth a method packet for establishing a transmit timestamp to be inserted into an outgoing packet by a network time protocol (NTP) server. The method includes the steps of requesting current time from a hardware clock within the NTP server, and requesting a correction term from a latency estimator filter (LEF), where the correction term is an estimate of a transmission latency associated with the outgoing packet traversing various layers of software before crossing a boundary with external environment. The method also includes the steps of inserting a sum of the current time and the correction term into the outgoing packet as the transmit timestamp, placing the

outgoing packet with the transmit timestamp on an outbound queue, triggering the hardware clock to latch a time-of-departure of the outgoing packet, performing steps of an LEF algorithm to update the correction term by calculating an offset between the transmit timestamp and the latched time-of-departure to obtain more accurate transmit timestamp in the next outgoing packet, storing the updated correction term, and using the updated correction term to insert the transmit timestamp into a next outgoing packet.

[0009] One advantage of the disclosed systems and methods is that improved control of the timestamping function may be implemented in the existing NTP servers without the risk of introducing interworking incompatibilities with other servers and clients that are compliant with the current NTP standards. Importantly, the disclosed methods include steps that utilize the packet timestamping engine and the latency estimator filter in a manner that retains compatibility with the current NTP standards. In this fashion, the time-of-arrival and time-of-departure are detected and registered by hardware, thereby achieving high degree of precision, and the uncertainty associated with the device drivers, network stacks, operating system, and so on is avoided. As a result, more accurate and stable NTP server timestamps are provided to the NTP client.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] So that the manner in which the above recited features of the present invention can be understood in detail, a more particular description of the invention, briefly summarized above, may be had by reference to embodiments, some of which are illustrated in the appended drawings. It is to be noted, however, that the appended drawings illustrate only typical embodiments of this invention and are therefore not to be considered limiting of its scope, for the invention may admit to other equally effective embodiments.

[0011] Figure 1 is a conceptual diagram of a packet timestamping flow in a typical NTP system, according to prior art;

[0012] Figure 2 illustrates timestamps contained in a header of an NTP packet, according to one embodiment of the present invention;

[0013] Figure 3 is a conceptual diagram of a packet timestamping flow in a precision timestamping service (PTS) system, according to one embodiment of the present invention;

[0014] Figure 4 sets forth a flow diagram of method steps implemented by the PTS system of Figure 3, according to one embodiment of the present invention;

[0015] Figure 5 sets forth a flow diagram of method steps implemented by the latency estimator filter of Figure 3, according to one embodiment of the present invention; and

[0016] Figure 6 is a conceptual diagram of a precision NTP server configured to implement one or more aspects of the present invention.

DETAILED DESCRIPTION

[0017] Figure 2 illustrates timestamps contained in a header of an NTP packet, according to one embodiment of the present invention. As shown in Figure 2, each time measurement along a time line 200 involves four timestamps, which are defined as follows: T_1 is a timestamp representing the best estimate of the transmit originating epoch of a packet originating from a client clock 210, T_2 is a timestamp representing the best estimate of the receive termination epoch of a packet terminating at a precision NTP server clock 220, T_3 is a timestamp representing the best estimate of the transmit origination epoch of a packet originating from the precision NTP server clock 220, and T_4 is a timestamp representing the best estimate of the receive termination epoch of a packet terminating at the client clock 210. After the time measurement, these four timestamps are used to calculate the roundtrip delay between the endpoints and the offset of the client clock 210 according to the following principles:

a) Delay estimate: $\sigma = (T_4 - T_1) - (T_3 - T_2)$ (1)

b) Offset estimate: $\Theta = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$ (2)

c) Error: $\varepsilon = \phi \cdot (T_4 - T_1) + \alpha$ (3)

where $\phi(t)$ is a function that describes the skew accumulation rate of the client clock and α is a measure of uncertainty of the precision NTP server time measurement.

[0018] Since the precision NTP server is primarily responsible for providing T_2 and T_3 to the client clock 210, any inaccuracy in the values of T_2 and T_3 contribute to inaccuracies in the delay and offset estimates, which, in turn, result in an increased value of uncertainty in the time measurement process α. However, equation (3) demonstrates that the precision NTP server can also improve the performance of the time measurement circuit by reducing uncertainty in the time measurement process α. The system and method for achieving more accurate and stable time measurements from the precision NTP server are described in Figures 3 through 5.

[0019] Figure 3 is a conceptual diagram of a packet timestamping flow in a precision timestamping service (PTS) system 300, according to one embodiment of the present invention. As shown, the PTS system 300 includes, without limitation, the physical layer associated with an Ethernet PHY 310, an Ethernet MAC 320, O/S services 330, a NTP application 340, and a hardware clock 350. As previously described, the time-of-arrival (denoted as T_{rx}) of an incoming packet 302 refers to the instant the first (or the designated) bit of the incoming packet 302 traverses the boundary between the external environment (not shown) and the Ethernet PHY 310. Subsequently, the first step for the incoming packet 302 that just crossed the said boundary between the external environment and the Ethernet PHY 310 is to continue to the Ethernet MAC 320 through a receive packet channel 312. A path for the incoming packet 302 is shown with arrows 312, 322, and 332. As also previously described, the time-of-departure T_{tx} of an outgoing packet 304 refers to

the instant when the first (or the designated) bit of the outgoing packet 304 crosses the said boundary between the Ethernet PHY 310 and the external environment. The last step for the outgoing packet 304 that is about to cross the said boundary between the Ethernet PHY 310 and the external environment is to travel through a transmit packet channel 314 (a path for the outgoing packet 304 is shown with arrows 334, 324, and 314). Combined, the receive packet channel 312 and the transmit packet channel 314 are referred to herein as a "media independent interface (MII) channel."

[0020] Importantly, the PTS system 300 further includes a passive tap unit 316 and a latency estimator filter (LEF) 360. The passive tap unit 316 is a packet timestamping engine configured to monitor the network traffic flowing through the MII channel. As each packet goes by, the passive tap unit 316 analyzes the data flowing through the MII channel to determine if the incoming packet 302 is of interest to the NTP application 340 (i.e., if the incoming packet 302 is an NTP packet). If this is the case, the passive tap unit 316 is configured to trigger the hardware clock 350 to latch (i.e., temporarily record) the time-of-arrival, T_{rx}, as shown with an arrow 319. Similarly, if the passive tap unit 316 detects the outgoing packet 304 created by the NTP application 340, the passive tap unit 316 triggers the hardware clock 350 to latch the time-of-departure, T_{tx}, as shown with an arrow 318. The functionality of a packet timestamping engine such as the passive tap unit 316 that includes monitoring the network traffic flowing through the MII channel and triggering a time latch at the hardware clock 350 may be implemented either in hardware or software. An example of a hardware implementation may be found in U.S. patent 6,985,499, filed April 17, 2001, and entitled "Precise Network Time Transfer," which is incorporated herein by reference. Persons skilled in the art will recognize that, in different embodiments, the passive tap 316 and the hardware clock 350 may be disposed on/in the same module or on/in different modules.

[0021] The LEF 360 is configured to obtain the latched time-of-departure, T_{tx}, from the hardware clock 350 and implement methods steps described in Figure 5 to

calculate a correction term k. The correction term k is an estimate of the transmission latency through the precision NTP server that is intended to account for the delays experienced by the NTP packets while traveling between the NTP application 340 and the Ethernet PHY 310. After each NTP client/server packet interchange, the LEF 360 updates the value of the correction term k to maintain the best estimate for the next transmission delay. Using the latched time-of-arrival, T_{rx}, and the most recent correction term k calculated by the LEF 360, the PTS system 300 implements method steps described in Figure 4, producing more precise and stable time measurements of the timestamps T_2 and T_3.

[0022] Figure 4 sets forth a flow diagram of method steps implemented by the PTS system 300 of Figure 3, according to one embodiment of the present invention. Although the methods steps are described in conjunction with the system of Figure 3, persons skilled in the art will understand that any system that performs the methods steps, in any order, is within the scope of the present invention.

[0023] The method begins in step 402, where the PTS system 300 initializes the LEF 360 by setting the elements in the array of transmission latencies to zero. In alternative embodiments, the elements in the array of transmission latencies may be set to some other pre-determined values. In step 404, the passive tap unit 316 determines if there is a new incoming packet 302 that is an NTP packet. If there is no new NTP packet, then the method proceeds to step 405 where PTS system 300 waits until the next packet arrives. From step 405 the method returns to step 404.

[0024] If, in step 404, the passive tap unit 316 determines that a new incoming packet 302 is an NTP packet, then the method proceeds to step 406, where the passive tap unit 316 triggers the hardware clock 350 to latch the time-of-arrival T_{rx}. As the incoming packet 302 traverses the Ethernet MAC 320 and the O/S services 330, the method proceeds to step 408 where the incoming packet 302 is received at the NTP application 340. Once the NTP application 340 recognizes that the incoming packet 302 has arrived, the method proceeds to step 410, where the NTP application 340 requests the latched time-of-arrival, T_{rx}, from the hardware clock

350, as shown in Figure 3 with an arrow 342. After the NTP application 340 obtains the requested latched time-of-arrival, T_{rx}, as shown in Figure 3 with an arrow 344, the method proceeds to step 412. In step 412, the NTP application 340 inserts the timestamp T_2 with a value equal to T_{rx} into the incoming packet 302. Note that this approach of determining the timestamp T_2 is quite different from the method of operation in the typical NTP system 100 where, upon sending the timestamp trigger 142 to the hardware clock 150, the NTP application 140 receives current time value as opposed to the time value representing the instant in the past when the incoming packet 102 actually arrived.

[0025] In step 414, the NTP application 340 creates a reply packet for the incoming packet 302. As the reply packet has not yet been transmitted, no corresponding transmit information is available from the hardware clock 350. In step 416, the NTP application 340 requests the current time, $T_{current}$, from the hardware clock 350, as shown in Figure 3 with an arrow 346. After the NTP application 340 obtains the requested current time, $T_{current}$, as shown in Figure 3 with an arrow 348, the method proceeds to step 418. In step 418, the NTP application 340 requests the correction term k from the LEF 360. An arrow 362 in Figure 3 illustrates the LEF 360 sending the correction term k to the NTP application 340. In step 420, the NTP application 340 adds the correction term k, which is an estimate of the transmission latency through the precision NTP server, to the value of $T_{current}$ obtained from the hardware clock 350 and inserts the result as the timestamp T_3 into the reply packet. The timestamp T_3 is also sent to the LEF 360.

[0026] In step 422, the NTP application 340 places the reply packet with the timestamp T_3 on the outbound queue as the outgoing packet 304. As the outgoing packet 304 traverses the O/S services 330, the Ethernet MAC 320, and is detected by the passive tap unit 316, the method proceeds to step 424. In step 424, the passive tap unit 316 triggers the hardware clock 350 to latch the actual time-of-departure T_{tx}. The method then proceeds to step 426, where the hardware clock 350 supplies the value of the actual time-of-departure, T_{tx}, to the LEF 360, as shown

in Figure 3 with an arrow 352. In turn, in step 428, the LEF 360 implements method steps that are described in more detail in Figure 5 and calculates an updated value for the correction term k. The method then returns to step 404, described above.

[0027] Figure 5 sets forth a flow diagram of method steps implemented by the LEF (latency estimator filter) 360 of Figure 3, according to one embodiment of the present invention. Again, although the methods steps are described in conjunction with the system of Figure 3, persons skilled in the art will understand that any system that performs the methods steps, in any order, is within the scope of the present invention.

[0028] The LEF 360 is designed to smooth jitter in correction term update. This is accomplished by maintaining a small array of transmission latencies with the newest entry forcing the oldest entry out of the array. The array may be denoted as $\{\mu_i, \mu_{i-1}, \mu_{i-2}, \ldots, \mu_{i-N}\}$, where N+1 is the number of elements in the array. In one embodiment, the array may contain only 2 elements, in which case N is equal to 1.

[0029] The method begins in step 502, where, as described in step 402 of Figure 4, the PTS system 300 initializes the LEF 360 by setting the values of the elements in the array of transmission latencies to zero. In alternative embodiments, the variables in the array of transmission latencies may be set to some other pre-determined values. In step 504, the LEF 360 determines if there is a new timestamp T_3 provided by the NTP application 340. If there is no new timestamp T_3, then the method proceeds to step 505, where the LEF 360 waits for a certain amount of time. From step 505 the method returns to step 504.

[0030] If, in step 504, the LEF 360 determines that there is a new timestamp T_3 (as shown in Figure 3 with an arrow 349), then the method proceeds to step 506, where the LEF 360 discards the old timestamp T_3. The method then proceeds to step 508, where the LEF 360 loads the new timestamp T_3 obtained from the NTP application 340 as well as the value of the actual time-of-departure, T_{tx}, provided by the hardware clock 350. The method then proceeds to step 510.

[0031] In step 510, the LEF 360 executes the steps of an LEF algorithm to sort the array of transmission latencies and chose the minimum value as the next value of the correction term k. In one embodiment, using the variables defined above, the LEF algorithm of step 510 is as follows:

Step 1: $\mu_{i-1} = \mu_i$ (bump the array down by one)

Step 2: $\mu_i = T_{tx} - T_3 + k$ (update the transmission latency)

Step 3: $k = \min(\mu_{i-1}, \mu_i)$ (calculate the new value for the correction term)

[0032] As a result of executing the steps of the LEF algorithm, in step 512, the LEF 360 produces an updated value for the correction term k, which represents an estimate of the transmission latency through the precision NTP server. The updated correction term k is stored and subsequently used in calculating the next transmit timestamp T_3 for the reply packet, as described in Figure 4. Persons skilled in the art will recognize that alternate algorithms for calculating the best estimate of the correction term k may be utilized as well, without departing from the basic scope of the present invention. The LEF algorithm presented in this embodiment constitutes a computationally efficient operation. From step 512, the method returns to step 502, described above.

[0033] Figure 6 is a conceptual diagram of a precision NTP server 600 configured to implement one or more aspects of the present invention. As shown, the precision NTP server 600 includes, without limitation, a central processing unit (CPU) 610, a network interface 620, and the PTS system 300, which further includes the hardware clock 350. The network interface 620 communicates packets with the PTS system 300, which, in turn, communicates with the CPU 610 and implements the method steps described herein to establishes the receive and transmit timestamps T_2 and T_3.

[0034] The present invention enables a precision NTP server to

- Monitor network traffic flowing through a MII channel, calculate transmission latency through the precision NTP server, insert a receive timestamp T_2 into an incoming NTP packet based on time-of-arrival latched by a hardware clock, insert a transmit timestamp T_3 into an outgoing NTP packet based on an estimate for the transmission latency that the outgoing NTP packet is expected to experience, and update an estimate of the transmission latency after each NTP client/server packet interchange by performing the steps of

    - Detecting an incoming packet from a precision NTP client immediately after the incoming packet crosses the boundary between the external environment and the physical layer of the precision timestamping service (PTS) system;

    - Triggering a time latch at the hardware clock to temporarily record time-of-arrival of the incoming packet;

    - Retrieving the latched time-of-arrival from the hardware clock and inserting a receive timestamp T_2 equal to the latched time-of-arrival into the incoming packet;

    - Requesting current time from the hardware clock and most recent estimate of the transmission latency from a latency estimator filter and inserting a transmit timestamp T_3 equal to the sum of the current time and the most recent estimate of the transmission latency into the outgoing packet before the outgoing packet actually crosses the boundary between the physical layer of the PTS system and the external environment;

    - Detecting an outgoing packet immediately before the outgoing packet crosses the boundary between the physical layer and the external environment;

- Triggering a time latch at the hardware clock to temporarily record the time-of-departure of the outgoing packet;

- Updating the estimate of the transmission latency by calculating the offset between the software calculated transmit timestamp value and the actual measured transmit timestamp and filtering the difference to chose the minimum value of the transmission latency as the next estimate of the transmission latency;

- Provide more accurate and stable measurements of the transitions of the NTP packets both into, and out of, the precision NTP server compared to the prior art methods.

[0035] One advantage of the disclosed systems and methods is that significantly better control of the timestamping function may be implemented in the existing NTP servers without the risk of introducing interworking incompatibilities with other servers and clients that are compliant with the current NTP standards. The disclosed precision timestamping service (PTS) system includes a packet timestamping engine implemented in hardware and configured to recognize NTP packets. The disclosed PTS system further includes a latency estimator filter configured to calculate an estimate of the transmission latency through various layers of software in NTP servers. Importantly, the disclosed methods include steps that utilize the packet timestamping engine and the latency estimator filter in a manner that retains compatibility with the current NTP standards. In this fashion, the time-of-arrival and time-of-departure are detected and registered by hardware, thereby achieving high degree of precision, and the uncertainty associated with the device drivers, network stacks, operating system, and so on is avoided. As a result, more accurate and stable NTP server timestamps are provided to the NTP clients.

[0036] One embodiment of the present invention is implemented as a program product for use with a computer system. The program(s) of the program product defines functions of the embodiments (including the methods described herein) and

can be contained on a variety of computer-readable storage media. Illustrative computer-readable storage media include, but are not limited to: (i) non-writable storage media (e.g., read-only memory devices within a computer such as CD-ROM disks readable by a CD-ROM drive) on which information is permanently stored; (ii) writable storage media (e.g., floppy disks within a diskette drive or hard-disk drive) on which alterable information is stored. Such computer-readable storage media, when carrying computer-readable instructions that direct the functions of the present invention, are embodiments of the present invention.

[0037] Other media include communications media through which information is conveyed to a computer, such as through a computer or telephone network, including wireless communications networks. The latter embodiment specifically includes transmitting information to/from the Internet and other networks. Such communications media, when carrying computer-readable instructions that direct the functions of the present invention, are embodiments of the present invention.

[0038] While the foregoing is directed to embodiments of the present invention, other and further embodiments of the invention may be devised without departing from the basic scope thereof.

I Claim:

1. A method for establishing a transmit timestamp to be inserted into an outgoing packet by a network time protocol (NTP) server, the method comprising:

initializing a latency estimator filter (LEF);

creating the outgoing packet;

requesting current time from a hardware clock within the NTP server;

requesting a correction term from the LEF, wherein the correction term is an estimate of a transmission latency associated with the outgoing packet traversing various layers of software before crossing a boundary with external environment;

inserting a sum of the current time and the correction term into the outgoing packet as the transmit timestamp;

providing the transmit timestamp to the LEF;

placing the outgoing packet with the transmit timestamp on an outbound queue;

monitoring network traffic flowing through a media independent interface (MII) channel to detect the outgoing packet;

triggering the hardware clock to latch a time-of-departure of the detected outgoing packet;

providing the latched time-of-departure to the LEF;

performing steps of an LEF algorithm to update the correction term by calculating an offset between the transmit timestamp and the latched time-of-departure to obtain more accurate transmit timestamp in the next outgoing packet;

storing the updated correction term and using the updated correction term to insert the transmit timestamp into a next outgoing packet.

2. The method of claim 1, wherein the LEF is initialized either by setting elements of an array of transmission latencies equal to zero or by setting the elements of the array of transmission latencies equal to pre-determined values.

3. The method of claim 1, wherein the outgoing packet is created as a reply to an incoming packet from an NTP client.

4. The method of claim 1, wherein the LEF algorithm includes the following:

Step 1: Move elements of an array of transmission latencies down by one:

$\mu_{i-1} = \mu_i$.

Step 2: Update most recent transmission latency:

$\mu_i = T_{tx} - T_3 + k$.

Step 3: Calculate a new value for the correction term:

$k = \min(\mu_{i-1}, \mu_i)$.

5. The method of claim 1, including further steps for establishing a receive timestamp to be inserted into an incoming packet by the NTP server, the steps comprising:

monitoring network traffic flowing through the MII channel to detect an incoming packet from an NTP client;

triggering the hardware clock to latch a time-of-arrival of the detected incoming packet;

inserting the latched time-of-arrival into the incoming packet as the receive timestamp.

6. A precision timestamping service (PTS) system configured to insert a transmit timestamp into an outgoing packet, the system comprising:

a physical layer associated with an Ethernet PHY;

an Ethernet media access controller (MAC);

a media independent interface (MII) channel connecting the Ethernet PHY and the Ethernet MAC;

operating system (O/S) services;

a hardware clock configured to provide current time and to latch a time-of-departure of the outgoing packet;

a latency estimator filter (LEF) configured to be initialized and to perform steps of an LEF algorithm to store and update a correction term by calculating an offset between the transmit timestamp and a time-of-departure latched by the hardware clock and to obtain more accurate transmit timestamp in a next outgoing packet;

a network time protocol (NTP) application configured to:

create the outgoing packet,

request current time from a hardware clock,

request the correction term from the LEF, wherein the correction term is an estimate of a transmission latency associated with the outgoing packet traversing various layers of software before crossing a boundary between the physical layer and external environment,

insert a sum of the current time and the correction term into the outgoing packet as the transmit timestamp,

provide the transmit timestamp to the LEF,

place the outgoing packet with the transmit timestamp on an outbound queue,

a passive tap unit configured to:

monitor network traffic flowing through the MII channel to detect the outgoing packet,

trigger the hardware clock to latch the time-of-departure of the detected outgoing packet,

provide the latched time-of-departure to the LEF.

7. The system of claim 6, wherein the LEF is initialized either by setting elements of an array of transmission latencies equal to zero or by setting the elements of the array of transmission latencies equal to pre-determined values.

8. The system of claim 6, wherein the outgoing packet is created as a reply to an incoming packet from an NTP client.

9. The system of claim 6, wherein the LEF algorithm includes the following:

Step 1: Move elements of an array of transmission latencies down by one:

$\mu_{i-1} = \mu_i$.

Step 2: Update most recent transmission latency:

$\mu_i = T_{tx} - T_3 + k$.

Step 3: Calculate a new value for the correction term:

$k = \min(\mu_{i-1}, \mu_i)$.

10. The system of claim 6, including further steps for establishing a receive timestamp to be inserted into an incoming packet by the PTS system, the steps comprising:

monitoring network traffic flowing through the MII channel to detect an incoming packet from an NTP client;

triggering the hardware clock to latch a time-of-arrival of the detected incoming packet;

inserting the latched time-of-arrival into the incoming packet as the receive timestamp.

11. A precision network time protocol (NTP) server configured to insert a transmit timestamp into an outgoing packet, the server comprising:

a central processing unit (CPU);

a network interface;

a precision timestamping service (PTS) system that has a hardware clock configured to provide current time and to latch a time-of-departure of the outgoing packet, wherein the PTS system includes:

a physical layer associated with an Ethernet PHY,

an Ethernet media access controller (MAC),

a media independent interface (MII) channel connecting the Ethernet PHY and the Ethernet MAC,

operating system (O/S) services,

a latency estimator filter (LEF) configured to be initialized and to perform steps of an LEF algorithm to update a correction term by calculating an offset between the transmit timestamp and a time-of-departure latched by the hardware clock and to obtain more accurate transmit timestamp in a next outgoing packet;

a network time protocol (NTP) application configured to:

create the outgoing packet,

request current time from a hardware clock,

request the correction term from the LEF, wherein the correction term is an estimate of a transmission latency associated with the outgoing packet traversing various layers of software before crossing a boundary between the physical layer and external environment,

insert a sum of the current time and the correction term into the outgoing packet as the transmit timestamp,

provide the transmit timestamp to the LEF,

place the outgoing packet with the transmit timestamp on an outbound queue,

a passive tap unit configured to:

monitor network traffic flowing through the MII channel to detect the outgoing packet,

trigger the hardware clock to latch the time-of-departure of the detected outgoing packet,

provide the latched time-of-departure to the LEF.

12. The system of claim 11, wherein the LEF is initialized either by setting elements of an array of transmission latencies equal to zero or by setting the elements of the array of transmission latencies equal to pre-determined values.

13. The system of claim 11, wherein the outgoing packet is created as a reply to an incoming packet from an NTP client.

14. The system of claim 11, wherein the LEF algorithm includes the following:

Step 1: Move elements of an array of transmission latencies down by one:

$\mu_{i-1} = \mu_i$.

Step 2: Update most recent transmission latency:

$\mu_i = T_{tx} - T_3 + k$.

Step 3: Calculate a new value for the correction term:

$k = \min(\mu_{i-1}, \mu_i)$.

15. The system of claim 11, including further steps for establishing a receive timestamp to be inserted into an incoming packet by the PTS system, the steps comprising:

monitoring network traffic flowing through the MII channel to detect an incoming packet from an NTP client;

triggering the hardware clock to latch a time-of-arrival of the detected incoming packet;

inserting the latched time-of-arrival into the incoming packet as the receive timestamp.

16. A computer-readable storage medium containing instructions for controlling a network time protocol (NTP) server to perform steps of:

initializing a latency estimator filter (LEF);

creating the outgoing packet;

requesting current time from a hardware clock within the NTP server;

requesting a correction term from the LEF, wherein the correction term is an estimate of a transmission latency associated with the outgoing packet traversing various layers of software before crossing a boundary with external environment;

inserting a sum of the current time and the correction term into the outgoing packet as the transmit timestamp;

providing the transmit timestamp to the LEF;

placing the outgoing packet with the transmit timestamp on an outbound queue;

monitoring network traffic flowing through a media independent interface (MII) channel to detect the outgoing packet;

triggering the hardware clock to latch a time-of-departure of the detected outgoing packet;

providing the latched time-of-departure to the LEF;

performing steps of an LEF algorithm to update the correction term by calculating an offset between the transmit timestamp and the latched time-of-departure to obtain more accurate transmit timestamp in the next outgoing packet;

storing the updated correction term and using the updated correction term to insert the transmit timestamp into a next outgoing packet.

17. The system of claim 16, wherein the LEF is initialized either by setting elements of an array of transmission latencies equal to zero or by setting the elements of the array of transmission latencies equal to pre-determined values.

18. The system of claim 16, wherein the outgoing packet is created as a reply to an incoming packet from an NTP client.

19. The system of claim 16, wherein the LEF algorithm includes the following:

Step 1: Move elements of an array of transmission latencies down by one:

$\mu_{i-1} = \mu_i$.

Step 2: Update most recent transmission latency:

$\mu_i = T_{tx} - T_3 + k$.

Step 3: Calculate a new value for the correction term:

$k = \min(\mu_{i-1}, \mu_i)$.

20. The system of claim 16, including further steps for establishing a receive timestamp to be inserted into an incoming packet by the PTS system, the steps comprising:

monitoring network traffic flowing through the MII channel to detect an incoming packet from an NTP client;

triggering the hardware clock to latch a time-of-arrival of the detected incoming packet;

inserting the latched time-of-arrival into the incoming packet as the receive timestamp.