Localization and Clock Synchronization Need Similar Hardware Support in Wireless LANs

Smruti Parichha and Mart Molle
Department of Computer Science and Engineering
University of California, Riverside
Riverside, CA, 92521, USA
{sparichh, mart}@cs.ucr.edu

***Abstract*— Secure localization protocols enable a group of cooperating *verifier* nodes in a wireless LAN to determine the physical location of a stranger (called the *prover*), using precision timing measurements during a carefully scripted packet exchange. After identifying a number of common features between the localization and precision clock synchronization problems, we describe a small set of additional physical-layer capabilities that — if incorporated into commercial wireless transceiver (PHY) products — could lead to significant performance improvements in both solution domains.**

I. Introduction

A. Notation

A "clock" $C(\cdot)$ is a function from event e (which must be visible to the "clock") to timestamp t, where t represents the reading of $C(\cdot)$ at the instant when e occurred. Note that we sometimes use t as a special location-independent event type, representing the instant when the ideal "universal clock" would generate timestamp value t. We say that $C_X(\cdot) \equiv C_Y(\cdot)$ if the two clocks are *synchronized* (i.e., they run at the same rate, with zero offset), and that $C_X(\cdot)||C_Y(\cdot)$ if the two clocks are *syntonized* (i.e., they run at the same rate while maintaining some fixed offset, possibly not zero).

B. Bi-lateral Timed-Echo Distance Bounding

Secure localization protocols enable the verifier(s) to impose location-based resource control policies on the prover, i.e., a coffee shop wishing to offer free wireless Internet service *to anyone sitting inside the store*, instead of a list of subscribers with pre-arranged login credentials. Therefore, simply providing a means for the prover to determine *its own* location (such as a GPS receiver) *is not a solution*, since a dishonest prover could simply claim a false position.

The basic *timed-echo distance bounding protocol* enables verifier V to upper bound its distance $D(V, P)$ from prover P – constraining P's location to a circle around V. Repeating the protocol with different verifiers can further localize P to a smaller region. The protocol, shown in Fig. 1(a), uses four timed events defined by two packet transmissions: e_1, V sends a "challenge"; e_2, the "challenge" reaches P; e_3, P sends its "response"; and e_4, the "response" reaches V. Afterwards, V finds the one-way network transit delay to P

$$\tau_{VP} = (C_V(e_4) - C_V(e_1) - \Delta_P)/2, \qquad (1)$$

and hence $D(V, P) \leq \tau_{VP} \cdot v$, where v is the signal propagation velocity.

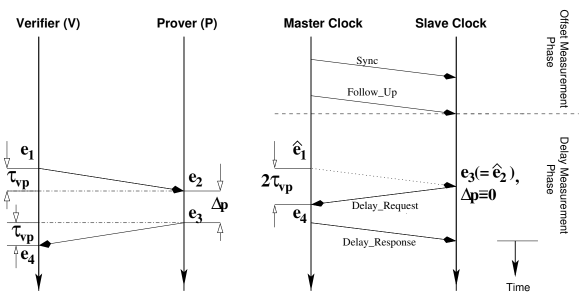

Fig. 1. Space-Time representations of: (a) the basic timed-echo distance-bounding protocol; and (b) IEEE 1588 precision clock synchronization.

The major difficulty with this approach is determining P's "response time", Δ_P, for interpretting the question, formulating an answer, and generating the response. Depending on the author, the value of Δ_P is assumed to be either "tiny" [1] or else P is expected to adjust e_3 to force Δ_P to take on "some fixed/known constant" [12]. This is a serious concern that can lead to distance fraud attacks (where a dishonest prover intentionally changes Δ_P) or low quality results (because $\Delta_P \gg \tau_{VP}$ and difficult to control).

C. Similarities to IEEE 1588 Precision Clock Synchroniztion

It is interesting to compare the IEEE 1588 precision clock sychronization algorithm to the basic timed-echo distance bounding protocol described above. During the first (offset-measurement) phase, the Slave, P, uses the periodic "Sync" and "Follow Up" messages sent by trusted Master, V, to adjust its local clock to satisfy $C_P(\cdot)||C_V(\cdot)$ with an offset lagging behind "Master time" by exactly τ_{VP}. During the second (delay-measurement) phase, P effectively *simulates the basic timed-echo protocol* using two timed events defined by one timed and one untimed packet transmission: e_3, P sends its "response" (Delay Request packet); and e_4, the "response" reaches the V. The "simulation" inserts an imaginary "challenge" ahead of the actual "response", defined by events $\hat{e}_1$, when it left V, and $\hat{e}_2$, when it reached P. Because of phase one, P knows that $C_P(\hat{e}_2) = C_V(\hat{e}_1)$

must hold. Moreover, because it is just a simulation, P sets $C_P(\hat{e}_2) = C_P(e_3)$, and hence $\Delta_P \equiv 0$. Therefore, once V sends $C_V(e_4)$ as payload of the untimed Delay Response packet, the Slave P (but *not* the Master V) knows $C_V(\hat{e}_1)$, Δ_P, $C_V(e_4)$, and can find τ_{VP}.

Notice that the IEEE 1588 protocol just provides a means for the Slave to adjust its own clock, with no provision for an outside entity (such as the Master) to verify that the Slave's clock has indeed been properly synchronized. To accomplish this stronger goal, we would need to add a *third clock-verfication phase* to the IEEE 1588 protocol, in which the Master "challenges" the Slave to send a "response" that reaches the Master at precisely its chosen time.

It is also interesting to note that the IEEE 1588 protocol replaces one precision timing requirement (i.e., knowledge of Δ_P) by another (i.e., $C_V(\cdot)||C_P(\cdot)$). And once again, *correctness* requires all parties to be trustworthy, and *accuracy* is limited by the participants' abilities to measure event times.

II. Multi-Lateral Time-Difference-of-Arrival

A. Infrastructure-Based Methods

More sophisticated localization protocols are based on time-difference-of-arrival (TDoA) measurements shared among multiple verifiers. As applied to infrastructure-based systems (including cellular 911 [3] or indoor localization with UWB radios [4]), the protocol consists of $N + 1$ timed events defined by the transmission of a single packet over an omnidirectional broadcast channel: e_3, P sends an unsolicited "response"; and $e_4^{(V_i)}$, the "response" reaches V_i, $1 \leq i \leq N$. Since the exact locations of all verifiers are assumed to be known, and $C_{V_i}(\cdot) \equiv C_{V_j}(\cdot)$ for all i, j, two verifiers can localize P to the following hyperbola

$$C_{V_i}(e_4^{(V_i)}) \cdot v - D(P, V_i) = C_{V_j}(e_4^{(V_j)}) \cdot v - D(P, V_j) \quad (2)$$

without considering e_3. Adding a third, non-collinear verifier collapses the result to a single intersection point. Thus, P can be uncooperative and/or dumb, as long as the verifiers can generate precision timestamps using synchronized clocks.

B. Smart-Repeater Based Method of Loschmidt et al.

Recently, Loschmidt et. al [8] have proposed a modification to the basic infrastructure-based TDoA protocol for utilizing commercial-off-the-shelf IEEE 802.11 hardware. They partition the fixed infrastructure into fully-functional verifiers, $V_1, \ldots, V_N$, that represent wireless access points running both phases of the IEEE 1588 protocol, and low-cost (possibly battery-powered) "smart repeaters", $R_1, \ldots, R_M$ that only run phase one (syntonization) of the 1588 protocol.

The repeaters' task is simply to transmit a duplicate "response" after a fixed, repeater-specific forwarding delay, Δ_{R_j}, as shown in Fig. 2. This increases the total number of timed events in the protocol to $(M + 1)(N + 2) - 1$ and introduces two new event types: $e_5^{(R_j)}$, R_j transits its duplicate; and $e_6^{(R_j,V_i)}$, the duplicate from R_j reaches V_i. The key observation is that V_i can determine the timestamp for repeater event $e_4^{(R_j)}$ from local event $e_6^{(?,V_i)}$ in two steps: (i) *identifying* R_j by matching $C_{V_i}(e_6^{(\cdot,V_i)}) - C_{V_i}(e_4^{(V_i)})$ with $\{\Delta_{R_j}\}$; and then (ii) *back-tracing the event to* R_j to give $C_{V_i}(e_4^{(R_j)}) = C_{V_i}(e_6^{(R_j,V_i)}) - \Delta_{R_j} - \tau_{R_j,V_i}$.

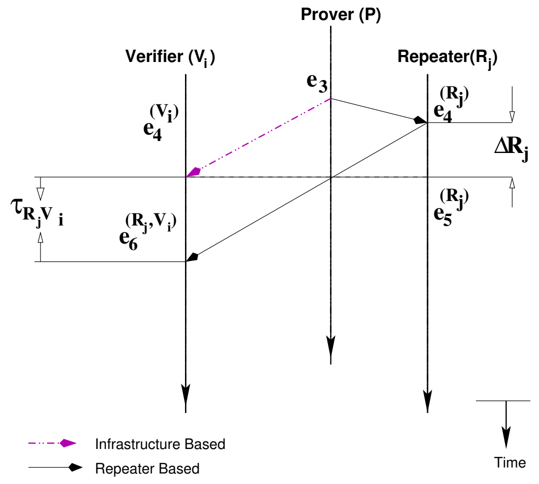

Fig. 2. The Smart-Repeater localization system of Loschmidt et al.

Obviously, Δ_{R_j} must be carefully chosen to avoid causing collisions and/or unanticipated access delays among the duplicates. Moreover, the perturbation of event timings due to the IEEE 802.11 CSMA/CA MAC protocol is a serious limitation to this approach, and may necessitate upgrading the capabilities of a repeater to support full timestamp generation with synchronized clocks. Finally, given the tight tolerances required for time stamps and forwarding delays (roughly $10ns$), it seems unlikely that the method can operate without the help of dedicated hardware support.

C. Witnessed Challenge-Response of Saha and Molle

Saha and Molle [10] have developed a novel localization protocol for omnidirectional broadcast networks, which adds $N - 1$ receive-only trusted "secondary verifiers" (or "witnesses") at known locations to the basic two-packet timed-echo distance bounding protocol, as shown in Fig. 3. Their protocol consists of $3N + 1$ timed events of the following types: e_1, V_1 sends the (only) "challenge"; $e_2^{(X)}$, the "challenge" reaches node $X \in \{P, V_1, \ldots, V_N\}$; $\hat{e}_2^{(X)}$, "end-of-challenge" reaches node $X \in \{V_2, \ldots, V_N\}$; $e_3^{(P)}$, P sends its "response"; and $e_4^{(V_i)}$, the "response" reaches V_i, $1 \leq i \leq N$. Note that $e_1 \equiv e_2^{(V_1)}$ are the same event.

Following the packet exchange, V_i knows the *packet interarrival time* in its local frame of reference,

$$A_i = C_{V_i}(e_4^{(V_i)}) - C_{V_i}(e_2^{(V_i)}). \quad (3)$$

Since the time delay between e_1 and $e_2^{(V_i)}$ is τ_{V_1,V_i}, we have

$$A_i \equiv D(V_1, P)/v_i + \Delta_P^{(i)} + D(P, V_i)/v_i - D(V_1, V_i)/v_i, \quad (4)$$

where v_i and $\Delta_P^{(i)}$, respectively, are the signal velocity and P's response time, normalized to $C_i(\cdot)$. Now suppose that

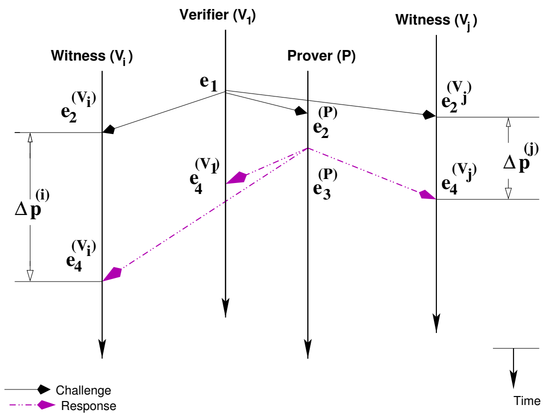

Fig. 3. Witnessed challenge-response localization by Saha and Molle.

$C_i(\cdot)||C_j(\cdot)$. In this case, $v_i \equiv v_j$ and $\Delta_P^{(i)} \equiv \Delta_P^{(j)}$, so simply forming the difference between two instances of Eq.(4) representing V_i and V_j would eliminate the unknown $\Delta_P^{(i)}$ and localize P to a hyperbola equivalent to Eq.(2):

$$A_i \cdot v_i + D(V_1, V_i) - D(P, V_i) = A_j \cdot v_j + D(V_1, V_j) - D(P, V_j). \tag{5}$$

Notice how the "challenge" serves as a common reference event among the verifiers, which relaxes the timing requirement from $C_i(\cdot) \equiv C_j(\cdot)$ (for other multi-lateral TDoA protocols) to just $C_i(\cdot)||C_j(\cdot)$ (for this approach). Moreover, we can determine the value of $v^{(i)}$ *without* assuming syntonization to the "universal clock" from a "dummy" execution of the protocol to "localize" the known position of a trusted verifier, since all distances in Eq. (5) are known. To complete the protocol, Saha and Molle proposed a novel method for setting $C_{V_i}(\cdot)||C_{V_1}(\cdot)$, which utilizes some hidden capabilities of existing Physical Layer transceivers.

In addition to its system clock, $C_X(\cdot)$, node X also has two other "clocks" integrated within its PHY transceiver logic: a *master*, $\overline{C}_X(\cdot)$, to control its *transmit logic* (e.g., a *crystal oscillator* set to the baud rate of the physical channel), and a *slave*, $\underline{C}_X(\cdot)$, to control its *receive logic* (e.g., a *phase-locked loop* that tracks the exact baud rate of each incoming packet). Their method enables all verifiers $\{V_i\}$ to both *learn* the "tick rate" for $\overline{C}_{V_1}(\cdot)$, and then *immediately use it* for their respective inter-packet time measurements.

By definition of the PHY transceiver logic, $\underline{C}_{V_i}(\cdot)||\overline{C}_{V_1}(\cdot)$ must hold while V_i is receiving the "challenge" sent by V_1, whereas $\overline{C}_{V_i}(\cdot)$ continues to run at its normal speed. Therefore, V_i can easily find the scale factor to ensure that

$$\rho_{V_1,V_i} \cdot \overline{C}_{V_i}(\cdot)||\overline{C}_{V_1}(\cdot), \tag{6}$$

as the ratio of clock increments from $e_2^{(V_i)}$ to $\hat{e}_2^{(V_i)}$, namely

$$\rho_{V_1,V_i} = \frac{\underline{C}_{V_i}(\hat{e}_2^{(V_i)}) - \underline{C}_{V_i}(e_2^{(V_i)})}{\overline{C}_{V_i}(\hat{e}_2^{(V_i)}) - \overline{C}_{V_i}(e_2^{(V_i)})} \tag{7}$$

III. Timing with Current IEEE 802.11 Hardware

A. *Synchronization Services in the IEEE 802.11 Standard*

The IEEE 802.11 Standard [5] currently provides a Time Synchronization Function (TSF), through which all stations syntonize their local MAC-layer protocol timers to the "timestamps" (actually 64-bit microsecond counter values) broadcast by the Access Point in periodic *Beacon Frames*. Separate from the TSF, the IEEE 802.11 standard also includes an optional capability called `MLME-HL-SYNC` in the MAC-layer management entity (MLME), which is intended to support application-layer time synchronization protocols.

To enable the `MLME-HL-SYNC` capability, the MAC client issues the `MLME-HL-SYNC.request` primitive to the MLME, together with a target multicast MAC address; this triggers the MLME to immediately issue the `MLME-HL-SYNC.confirm` primitive, together with a result code of either `SUCCESS` or `NOT_SUPPORTED`. If it is supported, the MLME starts searching for the next frame that contains the target multicast MAC address as its destination; when found, the MLME waits until the end of the frame and then issues the `MLME-HL-SYNC.indication` primitive to the MAC client, together with the source MAC address and sequence number from the triggering frame. Notice that the `MLME-HL-SYNC` capability handles both transmitted and received frames, in which case the `MLME-HL-SYNC.indication` primitive will coincide with either the `PHY_TXEND.confirm` or the `PHY_RXEND.indication` primitive, respectively.

Unfortunately, neither the TSF timer nor the `MLME-HL-SYNC` capability can match the precision timing requirments of applications described in Sections I and II. In particular, the specified tolerances for the TSF timer are rather loose ($\pm 0.01\%$) and in practice its accuracy will likely be substantially worse because the update mechanism does not account for variability in MAC-layer channel access delays. Moreover, the role of the MLME is strictly limited to issuing the `MLME-HL-SYNC.indication` primitive at certain end-of-packet events; the MAC client is left with the full responsibility for generating the timestamp to this event by consulting some sort of external clock.

B. *Overview of the IEEE 802.11 PHY-MAC Interface*

The 802.11 PHY consists of two sublayers: the *PMD* (i.e., the actual radio transceiver) and the *PLCP* (i.e., a set of functions for controlling and/or [re-]configuring the PMD). The PLCP-PMD boundary is somewhat vague because it was never intended to be an exposed interface, and hence is inappropriate for timestamping.

Figure 4 shows the sequence of primitives that cross the MAC-PHY interface to handle a single packet. Notice that the Preamble and PLCP Header are handled entirely by the PLCP sublayer: unlike Ethernet autonegotiation (which configures the PHY once at link startup), 802.11 nodes may need to reconfigure the PHY on a packet-by-packet basis to communicate with different nodes in the same Basic Service Area or even – for some modulation schemes – in mid packet.

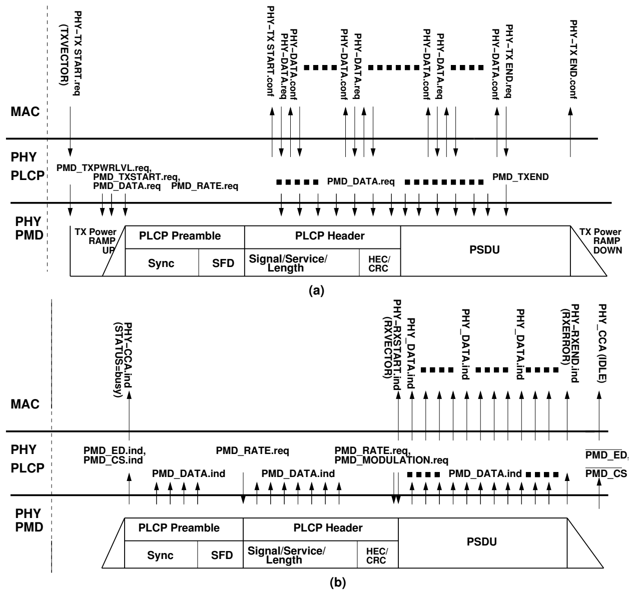

Fig. 4. Control Flow at the MAC-PHY Interface for an 802.11 Node: (a) Transmitting a Packet; (b) Receiving a Packet.

By default, the PMD is configured to receive incoming packet headers (CS/CCA state). Therefore, to initiate a packet transmission, the MAC issues the `PHY-TXSTART.request` to the PLCP, together with a parameter list including the data rate, packet length, preamble type, modulation to be used, scrambler initialization vector (if OFDM is used), and the transmit power level. Receipt of this primitive causes the PLCP to ready the PHY for this packet transmission by: (i) issuing various primitives to the PMD to configure and then power up its transmit function, (ii) generating an appropriate Preamble and PLCP Header, and (iii) then passing this stream of header data to the PMD for transmission.

Once the PMD has been configured and transmission of header data is under way, the PLCP issues the `PHY-TXSTART.confirm` primitive, telling the MAC of its readiness to accept the outgoing packet, one octet at a time, through an exchange of `PHY-DATA.req` and `PHY-DATA.confirm` primitives. After supplying the final octet of data, the MAC issues the `PHY-TXEND.request` primitive to the PLCP. Receipt of this primitive causes the PLCP to power down the PMD's transmit function and restore it to the CS/CCA state after the entire packet has been sent, then issue a `PHY-TXEND.confirm` primitive to the MAC acknowledging its completion.

Packet reception involves similar interations between the MAC and PLCP, shown in Fig. 4(b). As soon as the PMD detects a signal on the medium, the PLCP notifies the MAC by issuing the `PHY-CCA.indication` primitive with `STATUS=busy`, and then waits for the PMD receive function to synchronize with the incoming data stream. Once the PLCP has received enough of the incoming PMD data stream to detect a valid SFD and decode the parameters (including its length) from the PLCP Header, it issues a `PHY-RXSTART.indication` primitive to notify the MAC that a data packet is now arriving, and, possibly, reconfigures the PMD to a new rate and modulation scheme. Subsequently, each correctly-received octet is passed to the MAC with the `PHY-DATA.indicate` primitive. When it finds the end of the packet, the PLCP notifies the MAC by issuing the `PHY-RXEND.indicate` primitive with `RXERROR=no_error`, and reconfigures the PMD back to its default CS/CCA state. Finally, when the PMD has stopped detecting a signal, the PLCP notifies the MAC by issuing the `PHY-CCA.indication` primitive with `STATUS=idle`.

C. Using the PHY-MAC Interface to Support IEEE 1588

Recently, two experimental studies investigated hardware-assisted timestamping in commercial IEEE 802.11b hardware to support IEEE 1588 Synchronization. In both studies, the triggering events were derived from interface signals between the transceiver (PHY) and controller (MAC), using the Intersil PRISM 2.4 GHz WLAN Chip Set product family. In these Intersil products, the rising and falling edges, respectively, of the `TX-RDY` interface signal provide the `PHY-TXSTART.confirm` and `PHY-TXEND.confirm` primitives. Similarly, the rising and falling edges of the `TX-RDY` interface signal provide the `PHY-RXSTART.indicate` and `PHY-RXEND.indicate` primitives. It is interesting to note that the IEEE 1588 standard ([6], section 6.6.5) and the IEEE 802.11 `MLME-HL-SYNC` capability specify their respective timestamp reference points at opposite ends of the packet transmission: whereas 1588 uses "the beginning of the first symbol following the start of frame delimiter", the 802.11 `MLME-HL-SYNC.indication` uses the end-of-packet.

Cooklev et al. [9], [11] measured the one-way network delay between the PHY-MAC interfaces of two Cisco AIRONET series 340 wireless PC cards equipped with the Intersil HFA3861B chipset [2]. To limit the effects of jitter on the air medium (due to changing channel conditions and multipath, etc), the two radios were placed $1m$ apart with clear line-of-sight in an area known to be free of interference in the 2.4 GHz band. This configuration allowed the authors to focus on the jitter induced by the PHY circuitry.

Using an oscilloscope to capture the timing offset between signal transitions at the transmitting and receiving nodes, the authors found the two interface signals had a mean offset of $39.44\mu s$ and standard deviation of $145.6ns$ at the rising edges, compared to a mean offset of $7.35\mu s$ and standard deviation of $594ns$ at the trailing edges, and concluded that the "last-symbol-on-the-air" event is the appropriate timestamp reference point in 802.11 networks. On the other hand, it is important to recognize that these jitter measurements are orders of magnitude larger than the actual signal propagation delay ($3.3ns$) over the $1m$ air gap between the two nodes. Moreover, the spread between the minimum and maximum individual offset values in each experiment – from $39.20\mu s$ to $41.20\mu s$ at the rising edge, and from $-9.95\mu s$ to $9.64\mu s$ at the falling edge – shows how difficult it is to retrofit a timestamp reference point into pre-existing hardware.

Kannisto et al. [7] implemented a prototype for IEEE 1588 synchronization on a pair of Altera Excalibur EPXA1 embedded development boards connected to Intersil HW1151-EVAL transceivers equipped with the (slightly older) Intersil HFA3860B chipset. The ARM9 processors on each board handled the IEEE 1588 protocol, while FPGAs were used to implement the two 32-bit local second and nanosecond clocks for the 1588 protocol and generate packet timestamps triggered by the rising edge of the interface signals. The FPGAs also handled experimental data collection through serial ports connected to an external pulse generator (running at approximately 1 Hz) and a PC analyser connected to both development boards. The simultaneous arrival of a pulse to both development boards triggered their respective FPGAs to send a copy of its clocks (counters) to the PC analyser, which tracked the clock offset between the two boards over a 10 minute measurement period.

In their experiments, Kannitso et al. gave the Slave clock some initial offset and then started the IEEE 1588 synchronization protocol in both Master and Slave nodes. After discarding the first 5 minutes of "warmup" data, they calculated the average clock offset over the remainder of the measurement period. Using 10 replications of the complete experiment, they calculated the overall average offset of $1.1ns$ with a variance between replications of $3.1ns^2$.

Unfortunately, despite the remarkable accuracy of their reported results, we must point out that Kannisto's methodology provides almost no information about the measurement error in individual timestamps, since clock synchronization over a long interval is insensitive to individual timestamp errors, and the symmetric hardware configuration ensures that the timestamping errors will have similar distributions in both directions.[1]

IV. Adding Precision Timing to the 802.11 PHY

A. The Interval Counter and its External Interface

Separate from the MAC-layer TSF timer, every 802.11 PHY needs a high-precision *reference oscillator* to regulate both the transmit center frequency and symbol clock within its transmit logic. Depending on the chosen combination of modulation scheme and data rate, the specified tolerance[2] for the reference oscillator is never weaker than $\pm 25ppm$ – which is 40 times more strict than the tolerance for the MAC-layer TSF timer!

To take advantage of the PHY's existing reference oscillator, we now propose to add an "interval timer" to the PHY, i.e., a free-running counter clocked by the reference oscillator. Whenever some application at V_i needs to generate high-precision timestamps at packet-boundary events, it would use this PHY counter to emulate $\overline{C}_{V_i}(\cdot)$. Otherwise, the PHY counter logic could be disabled to reduce power consumption in the PHY, similar to the optional `MLME-HL-SYNC` capability in the current IEEE 802.11 standard.

For compatibility with the local clocks in IEEE 1588, and to provide enough resolution to represent propagation delays over distances on the order of $1m$, we will assume a 32-bit counter that runs at a nominal rate of 1 tick per nanosecond. However, it is important to recognize that it is just a simple, uncalibrated interval timer, not a full 1588-style clock, to avoid adding an unreasonable amount of complexity to the PHY. Moreover, since the IEEE 802.11 standard requires the PHY to use different oscillator frequencies for various combinations of modulation scheme and data rate, the counter won't necessarily advance in unit increments. For example, the counter might advance in fixed increments of size 50 under OFDM modulation with a 20 MHz sampling rate. More importantly, since the PHY must always use an 11 MHz chip rate for the Preamble and PLCP Header transmission (and $1/11$ is a non-terminating decimal fraction!), in this case the 11 counter-increments per microsecond would be properly rounded to 5 steps of size 91, followed by one step of size 90, and then another 5 steps of size 91.

To avoid the difficulties of attempting to control this interval timer remotely from an application program, let us further assume that the PHY counter is linked to read-only registers that automatically store its value at the most-recent start-packet or end-packet event, respectively, whenever the PHY counter is enabled. Thereafter, each stored counter value remains in its respective register until it is overwritten by events generated by the next packet. This provides the application program with a (relatively-large) window of time in which to retrieve the stored values of $\overline{C}_{V_i}(e)$ from the PHY registers, without further degrading the data due to the addition of an offset or some jitter to the retreived value.

B. Triggering Timestamps for Packet-Boundary Events

The simplest method for triggering the required timestamps would be to follow the IEEE 802.11 `MLME-HL-SYNC` capability and the experimental studies described in Section III-C in using some existing MAC-PHY interface signals. Even this naïve approach should provide better accuracy than the `MLME-HL-SYNC` capability, because the same PHY logic that issued the `MLME-HL-SYNC.indicate` primitive could simultaneously trigger a timestamp from the PHY counter, without waiting for the MAC client to respond to this primitive and generate a timestamp from another clock. However, these MAC-PHY interface signals are too far removed from events at the air-PMD interface to provide precision time stamping.

For example, restating Cooklev's results in the notation of Section II-C, we see that

$$(\overline{C}_P(\hat{e}_2^{(P)}) - \overline{C}_P(e_2^{(P)})) - (\underline{C}_V(\hat{e}_2^{(V)}) - \underline{C}_V(e_2^{(V)})) \approx 32\mu s$$

[1]Since the variance of the mean of N i.i.d. samples is $1/N$th the population variance, and $N \approx 300$ for one second sampling over a 5 minute experiment, we can use $\sqrt{3.1 \times 300} \approx 30ns$ as a crude estimate for the standard deviation of the individual clock offset samples in Kannisto's experiment – which is remarkably consistent with Cooklev's result of $145.6ns$ for the standard deviation of the individual timing offset samples.

[2]The transmit center frequency tolerance for all versions of the PHY are given as $\pm 25ppm$ except as follows. For 1 Mbps operation, the tolerance is specified as $\pm 60KHz$ on the 2.4 GHz band, which is equivalent to $\pm 25ppm$. For the 5 GHz band, the tolerance is specified as $\pm 20ppm$ for the 20 MHz and 10 MHz sampling rates, and $\pm 10ppm$ for the 5 MHz sampling rate.

holds on average. This discrepancy in the measured packet transmission time corresponds to an uncertainty of 350 bits in packet length (using a data rate of $11Mb/s$) or $96km$ in the distance between the two nodes! The major reason for this discrepancy is that the start-of-packet event at the air-PMD interface only affects the MAC-PLCP interface signals indirectly, and the offset between the two layers is inherently different for transmitters and receivers and also varies significantly between different combinations of modulation scheme and data rate.

To highlight the issue, let us define an "ideal" timing reference point for the start-of-packet event to occur when the end of the last bit from the Preamble and PLCP Header passes through the air-PMD interface. (A similar argument can be made for the end-of-packet event, but is omitted due to limited space.) From within the PMD, it should be possible (at least in theory) to determine the time of such events with uncertainties on the order of a single sampling period – although the answer may be delayed considerably to allow the PMD to carry out some off-line computations involving many samples. Nevertheless, *even if an oracle could instantly reveal the exact time of this "ideal" start-of-packet event to the PLCP*, neither the transmitter nor the receiver could change the time at which it issues the `PHY-TXSTART.confirm` or `PHY-RXSTART.indication` primitive, respectively.

The reason for this behavior – together with the fact that the IEEE 802.11 standard does not specify the exact timing of these primitives, relative to events at the air-PMD interface – should be evident from Fig. 4. At the transmitter, the `PHY-DATA.confirm` primitive must be issued *before the end* of the header error check (HEC) has been transmitted, but it could be as early as the start of the Preamble. Similarly, the `PHY-DATA.confirm` must be issued *before* any of the associated data is transmitted, and the `PHY-TXEND.confirm` must be issued *after* the transmission of last bit of the packet. Conversely, at the receiver, the `PHY-RXSTART.indication` primitive must be issued *after the end* of the HEC has been received, but it could be much later as long as the PHY has enough buffer space. Similarly, the `PHY-DATA.indicate(DATA)` and `PHY-RXEND.indicate` primitives must be issued *after* the associated data has been received.

V. Conclusion

We describe several localization algorithms and show how the witnessed time-difference-of-arrival method of Saha and Molle could be applied to a commercial Wireless LAN environment. We also show the similarity between timed packet exchanges used by localization algorithms and the IEEE 1588 Precision Clock Synchronization Protocol.

In order to estimate the distance to a node with an uncertainty of less than $10m$, a timestamping precision of about $30ns$ is required. Based on published experiments about supporting IEEE 1588 synchronization on similar hardware, and a careful review of IEEE 802.11 standards, we explain why this level of accuracy is just slightly out of reach for existing COTS hardware.

To solve this problem, we propose the addition of a simple counter and the supporting digital logic to the 802.11 PHY. The counter would be driven by the PHY's existing high-precision reference oscillator and equipped with a means for generating precise timestamps at the moment that a packet-start (or packet-end) passes through the air-PHY interface. An application program could later access the stored timestamp values simply by reading the contents a register – without degrading its value because of some uncontrolled notification and/or access delays.

References

[1] Stefan Brands and David Chaum. Distance-bounding protocols. In T. Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 344–359. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1994.

[2] Intersil Corp. Hfa3861b direct sequence spread spectrum baseband processor. Technical Report Data Sheet FN4816.2, February 2002.

[3] J. M. Zagami et al. Providing universal location services using a wireless e911 location network. In *IEEE Communications Magazine*, pages 66–71, April 1998.

[4] Robert J. Fontana, Edward Richley, and JoAnn Barney. Commercialization of an Ultra Wideband Precision Asset Location System. In *IEEE Conference on Wideband Systems and Technologies*, November 2003.

[5] IEEE. *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) And Physical Layer (PHY) Specifications*. Number 802.11-2007. Piscataway, NJ, June 2007.

[6] IEEE. *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. Number 1588-2008. Piscataway, NJ, Jan 2008.

[7] J. Kannisto, T. Vanhatupa, M. Hannikainen, and T.D Hamalainen. Software and Hardware Prototypes of the IEEE 1588 Precision Time Protocol on Wireless LAN. In *14th IEEE Workshop on Local and Metropolitan Area Networks*, September 2005.

[8] P. Loschmidt, G. Gaderer, and T. Sauter. Clock synchronization for wireless positioning of cots mobile nodes. In *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pages 64–69, October 2007.

[9] A. Pakdaman, T.Cooklev, and J.C.Eidson. Ieee 1588 over ieee 802.11b for. synchronization of wireless local. area network nodes. In *2004 Conference on IEEE 1588, Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, pages 116–127, November 2004.

[10] A. Saha and M. Molle. Localization with witnesses. In *Proc. 1st International Conference on New Technologies, Mobility and Security (NTMS 2007)*, May 2007.

[11] T.Cooklev, J.C.Eidson, and A. Pakdaman. An implementation of IEEE 1588 Over IEEE 802.11b for Synchronization of Wireless Local Area Network Nodes. *IEEE Trans. on Instrumentation and Measurement*, 56:1632–1639, 2007.

[12] Brent R. Waters and Edward W. Felten. Secure, Private Proofs of Location. Technical Report TR-667-03, Department of Computer Science, Princeton University, January 2003.