Calculating Relative Clock Drifts Using IEEE 802.11 Beacons

Ram Krishna
Department of Electronics and Communication Engineering
Indian Institute Of Technology Guwahati
781039, Assam, India

Christian Hoene
Interactive Communication Systems (ICS)
Universität Tübingen
72076 Tübingen, Germany

***Abstract*—This paper proposes an approach to calculate the relative clock drift occurring between two communicating nodes in an WLAN system using IEEE 802.11 beacon frames. The IEEE 802.11 beacon frames contain a timestamp field that tells the time at which the packet was sent from the transmitting node. If a IEEE 802.11 packet is received, the WLAN card stores a timestamp that precisely depicts the time at which the packet was received. Using these timestamps, we developed an algorithm that calculates the relative clock drift between two nodes and conducted many experimental measurements using off-the-shelf WLAN cards showing the clock drifts under different circumstances. Knowing the relative clock drifts is important for two-way time-of-arrival (TOA) location tracking.**

I. Introduction

State of the art localization technology has aroused considerable research interest in the last 20 years due to a wide range of application areas. The popularity of the localization technology stems from the fact that the traditionally used outdoor localization technology GPS system is hard to be implemented in indoor and dense urban areas. As the satellite's signals are reflected and/or diffracted by building structures, the receiver can not receive clear and strong satellite signal for localization purposes in such areas. Therefore, WLAN based location tracking algorithms have been developed. The most popular of all localization technology uses the Received Signal Strength Indications (RSSI) for determining the position of WLAN nodes, which is now a mature technology.

Location tracking based on time of arrival (TOA) algorithms measures the duration of the propagation of the physical transmission signal, which travels at the speed of light through vacuum. As compared to RSSI measurements, TOA has the benefit that its measurement results scale linearly with the open-air propagation distances. In our previous work [1], we have shown that even off-the-shelf IEEE 802.11 network interface cards can conduct time of flight measurements at accuracy of few meters even though the resolution of timestamps is only 1 µs or 300 m. We utilized the immediate acknowledgements of the IEEE 802.11 protocol to measure the round trip time of packets.

Because of drifting clocks, the TOA measurements are subjected to a time varying clock quantization effect, which helps to get results below the resolution of a single TOA measurement. Because we take advantage of rounding the alternating errors of clock quantization, the clocks of different nodes must not be synchronized and clock drifts must be present.

Sometimes, WLAN cards synchronize their clocks with the clocks of other nodes. In these cases, the clock drift is in the order of ppb (as compared to the typical ± 25 ppm of normal quartzes) and the two-way TOA measurement does not work anymore. Thus, it is important to determine the clock drift to know whether the results of two-way TOA measurements are meaningful. In this paper we present an algorithm to determine the clock drift using IEEE 802.11 beacon frames.

The IEEE 802.11 standard defines various frame types that nodes use for communication purpose, as well as for managing and controlling the wireless link. Every frame has a control field that depicts the 802.11 protocol version, frame type, and various indicators, such as whether WEP (wired equivalent privacy) is on, power management is active, and so on. In addition all frames contain MAC addresses of the source and destination station (or access point), a frame sequence number, frame body and frame check sequence (used for error detection).

IEEE 802.11 data frames carry protocols and data from higher layers within the frame body. A data frame, for example, could be carrying the HTML code from a web page (complete with TCP/IP headers) that the user is viewing. Other frames, that stations use for management and control, carry specific information regarding the wireless link in the frame body. For example, a beacon's frame body contains the Service Set Identifier (SSID), timestamp, and other pertinent information regarding the access point.

The access point (AP) in a WLAN periodically sends a beacon frame to announce its presence and relay information such as timestamp, SSID, and other parameters regarding the AP to radio NICs that are within range. Radio NICs continually scan all 802.11 radio channels and listen to beacons as the basis for choosing which access point is best to associate with.

In this publication, we selected the beacon frame as the basis for determining the clock drifts that occur in the wireless LANs devices. Every AP in WLAN periodically transmits beacon frames with the broadcasting time, which is stored in a piece of the beacon frame field called "beacon timestamp". The timestamps are generated by an AP clock and record the time when the first bit of the beacon frame hits the physical layer for transmission. The time resolution of a timestamp is governed by 802.11 protocols as one microsecond. When a receiving node receives beacon frame, every beacon frame's arrival time is recorded in the form of "MAC timestamp". These two timestamps can be used to calculate the relative clock drift occurring between the communicating nodes.

II. Related Work

There has been substantial works about estimating clock drifts in various environments. In [2] the authors have tried to estimate the relative clock drifts between two communicating nodes by measuring the length of successive response packets sent by one of the two nodes. In [3] the author has defined the notion of clock drift in terms of the rate of change of the hardware clock of one node with respect to the hardware clock of the other node. Similarly in [4], the author points out the classical solution by measuring the frequency offset at the output of the preamble synchronization system (PLL for instance) used for packet reception.

III. Measurements

A. Experimental Setup

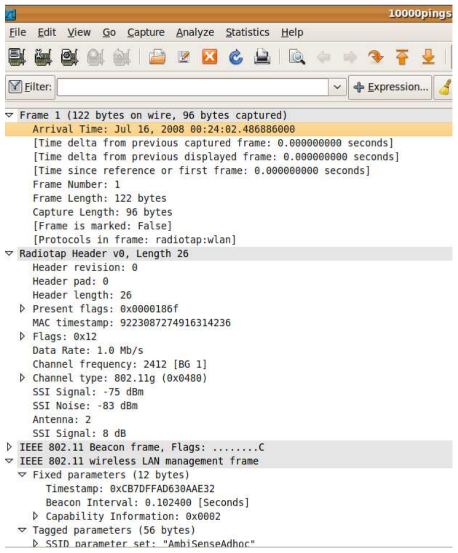
All IEEE 802.11 measurements were done at the University facility building called “Sand” in Tübingen, Germany. Using a Lenovo R51 notebook and a Netgear Dual Band Wireless PC Card WAG 511 (chipset Atheros 168c:0013), we collected with tcpdump [5] packets using the monitor mode of the WLAN card. Tcpdump is a powerful tool that allows us to sniff network packets and make some statistical analysis out of those dumps. During the experiments, the WLAN was set to the ad-hoc mode, the transmission rate was set to 1 Mbps, and the RTS/CTS mode was turned on.

Also, the notebook via another virtual interface was sending on the same WLAN card pings to an access point. Ping is a program that sends a series of packets over a network or the Internet to a specific computer in order to generate a response from that computer. The other computer responds with an acknowledgement that it received the packets. Ping estimates the round-trip time (RTT), generally in milliseconds, records any packet loss and prints a statistical summary when finished. However, throughout this publication, the pings were not of relevance. Actually, we ignored all pings and focused only on the IEEE 802.11 beacon packets, which were received at the same time.

This way we captured the network packets once for about two hours and once for about eight hours. During this period, multiple PCAP files were stored each containing either 2000 or 10000 floated pings and multiple beacon packets.

B. Data analysis

We obtained the entire network packets in the form of PCAP files. Then we exported the data in the form of text files using the "tcpdump" command. The exact tcpdump command that we used for exporting data from PCAP files was `"tcpdump -n -tt -eee -x -r source.PCAP > target.txt"` in which `"source.PCAP"` is the PCAP file whose data we want to export in the text format and



```
▽ Frame 1 (122 bytes on wire, 96 bytes captured)
    Arrival Time: Jul 16, 2008 00:24:02.486886000
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 122 bytes
    Capture Length: 96 bytes
    [Frame is marked: False]
    [Protocols in frame: radiotap:wlan]
▽ Radiotap Header v0, Length 26
    Header revision: 0
    Header pad: 0
    Header length: 26
  ▷ Present flags: 0x0000186f
    MAC timestamp: 9223087274916314236
  ▷ Flags: 0x12
    Data Rate: 1.0 Mb/s
    Channel frequency: 2412 [BG 1]
  ▷ Channel type: 802.11g (0x0480)
    SSI Signal: -75 dBm
    SSI Noise: -83 dBm
    Antenna: 2
    SSI Signal: 8 dB
▷ IEEE 802.11 Beacon frame, Flags: ........C
▽ IEEE 802.11 wireless LAN management frame
  ▽ Fixed parameters (12 bytes)
      Timestamp: 0xCB7DFFAD630AAE32
      Beacon Interval: 0.102400 [Seconds]
    ▷ Capability Information: 0x0002
  ▽ Tagged parameters (56 bytes)
    ▷ SSID parameter set: "AmbiSenseAdhoc"
```

Fig. 1. A PCAP file displayed with Wireshark.

`"target.txt"` is the target text file where we want to store the data obtained from PCAP file. Actually using this particular tcpdump command, we made sure that we have at least some important parameters present in our text file for the frames. These important parameters include the MAC addresses of the sending and receiving stations, beacon timestamps, MAC timestamps and the another important parameter called arrival time. The arrival time is a timestamp which tells the time (in microseconds) at which the beacon frame was received by the destination with respect to a reference time, where the reference is the EPOCH (1. Jan 1970). Figure 1 shows a typical PCAP file opened using Wireshark software and showing the arrival time, MAC timestamp and beacon timestamp for one transmitted beacon frame.

Having extracted all these parameters from all the PCAP files in text format we go onto calculate the clock drift. The process of clock drift calculation from the tcpdumped text files involves the following key steps:

1) For a particular MAC address, store the beacon timestamps t_i^b, MAC timestamps t_i^M and arrival times t_i^a occurring in a particular PCAP file. If multiple beacons are received having the same MAC address, this step has to be repeated for all $i \in \{1...n\}$.
2) Then subtract all the beacon timestamps from the beacon timestamp of the first beacon occurring in that PCAP file and thus it gives the modified beacon timestamp for all the beacons: $t_i^{b'} = t_i^b - t_1^b$. Similarly subtract the MAC timestamps of all the beacons from that of the first occurring beacon and thus giving us the modified MAC timestamp: $t_i^{M'} = t_i^M - t_1^M$.
3) Now we subtract the MAC timestamp corresponding to each beacon from their corresponding beacon timestamp: $d_i = t_i^{M'} - t_i^{b'}$. This gives us the difference of the timestamps at which the beacon was sent from one node and the timestamp at which it was received by another

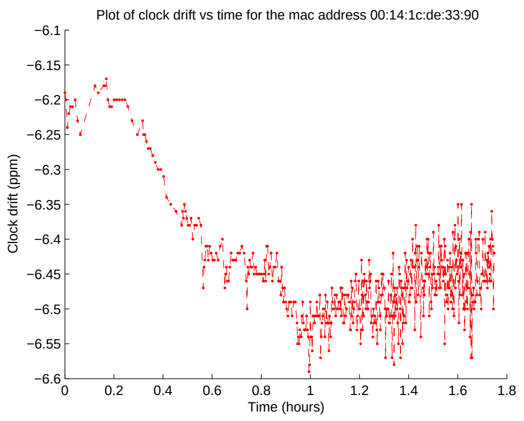


Fig. 2. Plot of clock drift for 2000 ping case for MAC address 00:14:1c:de:33:90 (Cisco)

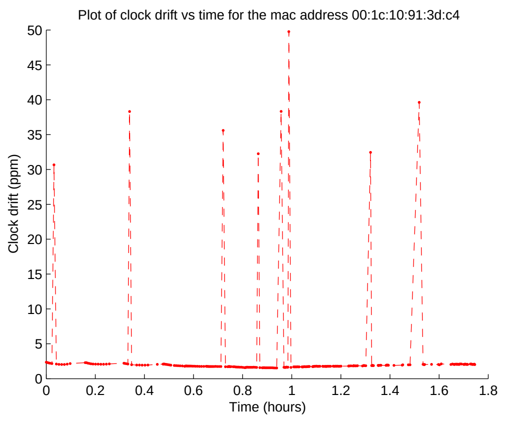


Fig. 3. Plot of clock drift for 2000 ping case for MAC address 00:1c:10:91:3d:c4 (Linksys)

node. Now we plot the values of the difference between these two timestamps d_i versus the MAC timestamp $t_i^{M'}$. Then we apply linear regression to get the best fit line for the data points obtained that way: $d = t^{M'} \cdot \beta + \epsilon$. We also calculated the degree of fitness R^2 to signify the accuracy to which the best fit line was in good agreement with the actual curve that the data points signified.

4) The slope β of the best fit line obtained in the way described above is the clock drift for our system.
5) This process was applied for all MAC addresses of a particular PCAP file and this was repeated for all of the PCAP files.

The algorithm described above is implemented in Matlab using a script which calculates the clock drift corresponding to all MAC addresses in a PCAP file. This way the clock drift for all MAC addresses of all PCAP files are obtained and the results are stored in a text file called `"table.txt"`.

C. Results

We captured a number of PCAP files from the active network using the tcpdump and analyzed the data to obtain the clock drifts for different MAC addresses. The plot of clock drifts versus arrival time (modified by subtracting all arrival times from the first arrival time in a file for a particular MAC address) were largely stable in the order of a few ppm. The plot of clock drift for the MAC address 00:14:1c:de:33:90 (a Cisco Aironet 1131 AP) for the 2000 pings case is shown in Figure 2 while that for the MAC Address 00:1c:10:91:3d:c4 (our Linksys AP) for the 2000 pings case are shown in Figure 3. Similarly the plot of clock drift for the MAC address 00:05:4e:4c:9d:7b (a Philips product) for the 10000 pings case is shown in Figure 4.

D. Discussion

The clock drifts calculated by our algorithm were pretty stable for 2000 ping packets for both observed MAC Addresses 00:14:1c:de:33:90 and 00:1c:10:91:3d:c4. The clock drift for the MAC Address 00:14:1c:de:33:90 can be examined to vary

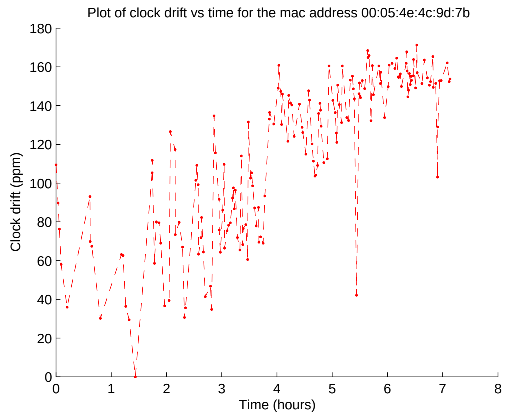


Fig. 4. Plot of clock drift for 10000 ping case for MAC Address 00:05:4e:4c:9d:7b (Philips)

between -6.15 to -6.6 ppm whereas for the MAC Address 00:1c:10:91:3d:c4 it was fairly constant at values close to 2.5-3 ppm. The clock drift for the 10000 ping case for the MAC Address 00:05:4e:4c:9d:7b was largely varying between 40-180 ppm.

IV. Measuring Clock Drifts with our Trilateration Software

Previously, we have developed an open source WLAN location tracking software called “Goodtry” [6], [7], which can trilaterate WLAN nodes by using off-the-shelf WLAN card and round-trip time-of-flight measurements. We have extended Goodtry to measure the clock drifts of neighbouring IEEE 802.11 ad-hoc nodes and access points. As described in the previous section, we collect both the timestamps: when the beacon has been sent and when it has been received. Then, a linear regression is calculated to determine the relation between sent and received timestamps. The values of slope, which ideally should be 1, helps to calculate the clock drift, which is the slope minus one. In addition, we calculate the 95% confidence of the results to know how precise the linear regression fits the measurement data.

TABLE I
CLOCK DRIFT MEASURED WITH TWO DIFFERENT WLAN CARDS.

MAC address	AR5413	BCM4318	AP hardware
000E8E0FF573	237.3	-0.0003	SparkLAN
001A70FCD290	244.0	7.13471	Linksys WRT54GL (BCM5352)
004096575809	243.9	7.09844	Linksys WRT54GL (2)
001C102FADB4	244.0	7.16844	Linksys WRT54GL (3)
001C102FAF25	243.6	7.1245	Linksys WRT54GL (4)
001C103CD5FE	244.0	7.1583	Linksys WRT54GL (5)
001C10913DC4	243.5	7.1576	Linksys WRT54GL (6)
00223F8BF62F	245.5	8.28903	Netgear WG602v4 (Broadcom 5354)
002333C2CA80	234.1	-2.83843	Cisco (Aironet 1131)
002333C2CE80	231.2	-5.78454	Cisco (2)
002333C2D520	236.3	-0.4948	Cisco (3)
002333C2D580	231.7	-5.1837	Cisco (4)
0024148A4C60	233.5	-3.3938	Cisco (5)

Using this software, we conducted multiple experiments using two different WLAN NIC cards as observing node. Again, the measurements took place in the Sand building in Tübingen, during which we used a Atheros AR5413 (168C:001B) card and a Broadcom BCM4318 card (14E4:4318). Nearby, six Linksys WRT54GL, five Cisco, one Netgear and one SparkLAN routers were placed.

The measurement data shows some interesting results. Firstly, the AR5413 chipset clock timer is much faster than what the standard limits allows (0.01%). Assumingly, it tries to achieve a higher transmission performance by overclocking. Earlier Atheros and the Broadcom chipsets did not show this behaviour and work well within the typical limits (± 25 ppm).

If we look at the clock drift of the access points, it can be seen that in case of the Cisco hardware the clock drifts differ by a few ppm. Such distribution can be expected if one considers the typical accuracy of quartzes. The Linksys WRT54GL access point (they were running all in ad hoc mode), did not show such a distribution. The clock drifts were below one ppm. We assume that the built-in Broadcom WLAN hardware adjusts its timer to the beacons (as required in the IBSS/ad-hoc mode). Thus, in these cases one hardly sees any clock drift.

Alternatively, Broadcom WLAN hardware might be having a phase-lock-loop that tunes the clock to the nodes. The Broadcom WLAN driver has a parameter called “freqtrack” which we assumed to be setting the afore mentioned frequency tracking on and off. However, we were unable to disable the frequency tracking in our experiments.

We also conducted measurements of the clock drift over longer periods of time. Figure 5 displaying a period of 15 meters shows that the relative clock drift does not remain stable. At the beginning of the measurement period, the relative clock drift is slower and it becomes faster at the end.

V. CONCLUSION

In this publication we introduce an easy and effective way to measure the relative clock drift between two WLAN nodes. Our experimental results yielded the following findings. Typically, the relative clock drift differs in the order of 10 ppm.

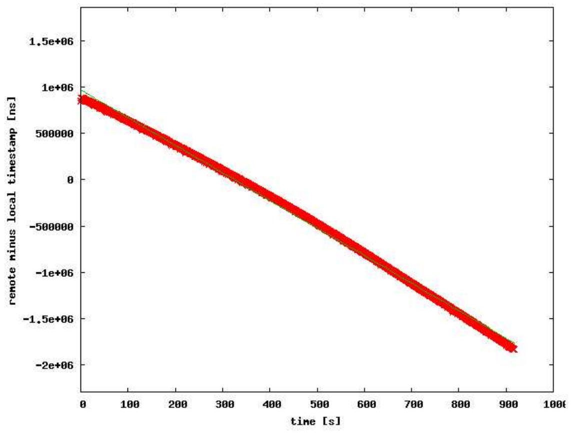


Fig. 5. Relative clock drift between a BCM4318 WLAN card and a Cisco AP over time.

This result was expected because this is the accuracy of commercial grade quartzes. Also, the relative clock drift does not remain stable and it keeps changing. Thus, the clock drift measurements has to be done continuously to adapt to varying conditions.

Some commercial WLAN cards show unexpected properties. For example, the Atheros AR5413 chipset has been overclocked and runs faster than the recommended limits of IEEE 802.11 standards. On the other hand, Broadcom chips tend to synchronize their clocks and hardly showed any clock drifts. Indeed, in those cases a software based time-of-flight measurement is not possible, because the time varying time quantisation cannot be observed anymore.

ACKNOWLEDGMENT

This work was supported by the German Academic Exchange Service (DAAD) and Landesstiftung Baden Württemberg gGmbH within the AmbiSense project of University of Tübingen. The authors would take the opportunity to thank S. D. Mandal, M. Hyder, S. König and M. Schmidt for helping us.

REFERENCES

[1] P. Bahl and V. N. Padmanabhan, “RADAR: an in-building RF-based user location and tracking system,” in *Infocom 2008*, vol. 2, Tel-Aviv, Israel, 2000, pp. 775–784.

[2] K. I. Ahmed and G. Heidari-Bateni, “Improving two-way ranging precision with phase-offset measurements,” in *IEEE Global Telecommunications Conference (GLOBECOM ’06)*, 2006, p. 1.

[3] L.-M. He, “An improved time synchronization algorithm for wireless sensor networks,” in *4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM ’08)*, Oct. 2008, pp. 1–4.

[4] N. Marechal, J.-B. Pierrot, and J.-M. Gorce, “Fine synchronization for wireless sensor networks using gossip averaging algorithms,” in *IEEE International Conference on Communications (ICC ’08)*, May 2008, pp. 4963–4967.

[5] (2008) Tcpdump/libpcap public repository. [Online]. Available: http://www.tcpdump.org/

[6] C. Hoene. (2009) Goodtry - software based WLAN trilateration. [Online]. Available: http://www.ambisense.uni-tuebingen.de/

[7] C. Hoene and J. Willmann, “Four-way TOA and software-based trilateration of IEEE 802.11 devices,” in *IEEE PIMRC*, Cannes, Sep. 2008.