Accuracy Time Synchronization for USRP

April 20, 2013

Student: Gong Zhang, graduate student, China
Email: zhangnow@gmail.com
Open source project: GNU Radio

1 Abstract

Because of crystal oscillator instability, USRP internal clock will drift away from ideal time even if it's initially perfectly tuned, which leads to out of synchronization. Clock synchronization is well studied in wireless sensor networks[1] (WSN), and it can be used with USRP in several distributed applications.

There are several ways to synchronize USRPs suggested in the synchronization application notes in official website[2]. But it's unrealistic to use a MIMO cable or add an expensive GPSDO for synchronization in some distributed applications such as measure the range. Referring to PC time is also illegal because PC uses Internet (maybe still inaccessible) and can only obtain milliseconds accuracy. Our project aims to implement a network synchronization protocol to obtain microseconds sync accuracy, which just uses USRP transmitting signal.

2 Benefits

Accuracy clock synchronization is very useful for some distributed applications especially without external input.

- Collaborative Communication[3]. It's important for relays to forward the data at the same time.
- Low Duty Cycle Network[4]. Nodes in low duty cycle network are sleeping most of the time for energy saving. And they wake up and turn on the radio chip at the same for receive and send data. It critical for them to wake up at the same time. Otherwise, missing the wake time leads to failure of data delivery.
- Frequency Hopping. Fast frequency hopping over a broad band require the receiver to tune the LO precisely when the frequency changing. This project can easily achieve network synchronization for network frequency hopping.
- Measurement[5]. We can get the propagation time after recording the time when signal is send and received. Then the distance between the sender and receiver is obtained. This is very important for localization. And the localization is the foundation for geographic routing, habitat monitoring and tactical surveillance.
- Other Applications. Such as sensor network-based countersniper system[6], Radio interferometric geolocation[7] and so on.

3 The project

3.1 Background(mainly from [1][8])

This maybe a little long, but it shows that I'm ready for all theory preparation. And this's useful for who is unfamiliar with clock synchronization.

According to the data sheet of USRP-N210,the frequency of a clock varies 2.5ppm, which means clock of different nodes can loose as much as 2.5μs in a second. The clock relationship between two USRPs (A and B) can be modeled as[1]

$$C_B(t) = \theta^{AB} + f^{AB} \cdot C_A(t), \tag{1}$$

where the parameter θ^{AB} and f^{AB} are called relative initial clock offset and relative skew between A and B. If two clock are perfectly synchronized, $\theta^{AB} = 0$ and $f^{AB} = 1$.

A typical synchronization method is as following. If USRP B needs to be synchronized to USRP A. Assuming A's clock is the ideal clock

$$C_A(t) = t. \tag{2}$$

At time t_1 and t_2, USRP A sends its current time to B. B receive them at time $T_{1,B}$ and $T_{2,B}$ referring to B's local clock. If there is no delay,

$$\begin{aligned} T_{1,B} &= \theta^{AB} + f^{AB} \cdot t_1 \\ T_{2,B} &= \theta^{AB} + f^{AB} \cdot t_2 \end{aligned}. \tag{3}$$

After rearrangement, we get

$$\begin{aligned} f^{AB} &= \frac{T_{2,B} - T_{1,B}}{t_2 - t_1} \\ \theta^{AB} &= T_{1,B} - \frac{T_{2,B} - T_{1,B}}{t_2 - t_1} \cdot t_1 \end{aligned}. \tag{4}$$

Unfortunately, in a real world wireless network, various delays affect the message delivery, making clock synchronization much more difficult than is seems to be:

- Send Time: The time spent in building the message at the application layer, including delays introduced by the operation system when processing the send request. Depending on the current processor load, it's nondeterministic and can be as high as hundreds of milliseconds.
- Access Time: The waiting time for accessing the channel after reaching the medium access control layer. It's varying from milliseconds up to seconds depending on the current network traffic.
- Transmission Time: The time for transmitting a message at the physical (PHY) layer. It's deterministic depending on the message length.
- Propagation Time: The actual time for a message to be transmitted from the sender to the

receiver in a wireless channel. It's deterministic depending on the distance between sender and receiver.

- Reception Time: The time required for receiving a message at the PHY layer, which is the same as the transmission time. It's the same as the transmission time.
- Receive Time: Time to construct and send the received message to the application layer at the receiver. And it's the same as the send time.
- Encoding/Decoding Time: The time it takes for the radio chip to encode message into electromagnetic waves or decode electromagnetic waves into message. There are deterministic.
- Interrupt Handling Time: The delay between the radio chip raising and the microcontroller responding to an interrupt. It caused by waiting for the microcontroller to finish the currently executed instruction. It depends on the microcontroller frequency.

After subtracting the deterministic delay from (3), we get

$$\begin{aligned} T_{1,B} &= \theta^{AB} + f^{AB} \cdot t_1 + e_1 \\ T_{2,B} &= \theta^{AB} + f^{AB} \cdot t_2 + e_2 \end{aligned}, \tag{5}$$

where e_1 and e_2 mean the nondeterministic delays above.

For better accuracy, B receiver N message from A and we get

$$\mathbf{T_B} = \mathbf{T_A} \cdot \boldsymbol{\eta} + \mathbf{e}, \tag{6}$$

where $\mathbf{T_B} = \begin{pmatrix} T_{1,B} \\ T_{2,B} \\ \cdots \\ T_{N,B} \end{pmatrix}, \mathbf{T_A} = \begin{pmatrix} 1 & t_1 \\ 1 & t_2 \\ .. & .. \\ 1 & t_N \end{pmatrix}, \boldsymbol{\eta} = \begin{pmatrix} \theta^{AB} \\ f^{AB} \end{pmatrix}$ and $\mathbf{e} = \begin{pmatrix} e_1 \\ e_2 \\ .. \\ e_N \end{pmatrix}$.

Using least square estimator, we can get $\boldsymbol{\eta}$ and B is synchronized to A.

$$\boldsymbol{\eta} = (\mathbf{T_A^T T_A})^{-1} \mathbf{T_A^T T_B} \tag{7}$$

Considering A can broadcast the message, **a network can also be synchronized** to A even via synchronized to one already synchronized (multi-hop).

3.2 Details

To achieve microsecond synchronization accuracy, the key points are that A record the actual time when the message leave and B record the actual time when the message is received.

According to the above, recording time in PC is infeasible because the message passing ethernet port introducing nondeterministic delay up to hundreds of millisecond. So we must perform recording time in USRP both on the sender and receiver. Therefore, it's very close to the antenna, thus the jitter caused by send/receive/access/reception time can be eliminated. And there is no interrupt handling time because of USRP processes the incomings like a stream without interrupt. And the encode/decode time is deterministic and can be compensated with some pre-test.

After several months reading the source code and mailing list, I find the TX tags functions, which could set the time when custom data are send. The thinking is creating a custom data containing the time that the data will be send. The framer is as Fig.1.

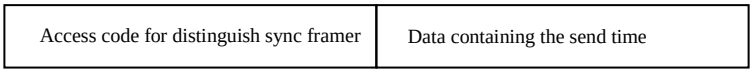

Fig.1 Synchronization Framer Format

On the receive side, when data pass DDC, the time is recorded in metadata. Using modified framer sink block, I can distinguish the synchronization framer. Then, I get the receive time of the framer from the metadata. Thus, I **get the time when the framer was send and received**. Then it's easily perform synchronization using Equation 7.

My plan is achieving microsecond synchronization accuracy, but it depend on the delay jitter between real send/receive time and TX/RX tags. I will test the delay jitter and analyze the influence on synchronization accuracy.

4 Required Deliverables

- We will provide a synchronization block in GNU Radio, which can easily synchronize multiple USRPs.
- Test the delay jitter between real send/receive time and TX/RX tags in USRP-N210 and provide report.
- Perform a real-world experiment to see the synchronization accuracy.
- Analyze the delay jitter influence on synchronization accuracy. All of the above will be included in final report with necessary pictures.

5 Project Schedule

- May 27 – Jun. 25: I will submit a report about the code architecture.
- Jun. 26 – Jul. 25: Coding synchronization block and writing related documentation.
- Jul. 26 – Aug. 6: Fixing bugs and testing the synchronization accuracy. Ensure what the accuracy can be obtained.
- Aug. 7 – Aug. 20: Testing the delay jitter in USRP-N210.
- Aug. 17 – Sep. 11: Quantitative analysis of the delay jitter's influence on synchronization accuracy, writing related documentation.
- Sep. 12 – Sep. 23: Fixing bugs and writing final documentation.

6 My advantage

I like the Open Source software model and it really changes the world. My research area includes wireless sensor network and statistical signal processing, and I'm very interested in implement some WSN application in USRP. That's really cool and may benefit others. In section 2.1, you can see my previous theory preparation and that has been done.

I’m a graduate student so I can work at least 6 days per week have at least 48 hours per week for working in this project. After finishing the project, it may become part of my paper, which has great probability publishing on academic journals. In addition, my lab has four USRP-N210s with WBX and Agilent N9010A signal analyzer, which helps the project with many convenient.

Ref:

[1] Wu Y C, Chaudhari Q, Serpedin E. Clock synchronization of wireless sensor networks[J]. Signal Processing Magazine, IEEE, 2011, 28(1): 124-138.

[2] UHD - Synchronization Application Notes.

[3] Yavatkar R. MCP: A protocol for coordination and temporal synchronization in multimedia collaborative applications[C]. Distributed Computing Systems, 1992., Proceedings of the 12th International Conference on. IEEE, 1992: 606-613.

[4] Ye W, Silva F, Heidemann J. Ultra-low duty cycle MAC with scheduled channel polling[C]. Proceedings of the 4th international conference on Embedded networked sensor systems. ACM, 2006: 321-334.

[5] Patwari N, Ash J N, Kyperountas S, et al. Locating the nodes: cooperative localization in wireless sensor networks[J]. Signal Processing Magazine, IEEE, 2005, 22(4): 54-69.

[6] Simon G, Maróti M, Lédeczi Á, et al. Sensor network-based countersniper system[C]. Proceedings of the 2nd international conference on Embedded networked sensor systems. ACM, 2004: 1-12.

[7] Maróti M, Völgyesi P, Dóra S, et al. Radio interferometric geolocation[C]. Proceedings of the 3rd international conference on Embedded networked sensor systems. ACM, 2005: 1-12.

[8] Maróti M, Kusy B, Simon G, et al. The flooding time synchronization protocol[C]. Proceedings of the 2nd international conference on Embedded networked sensor systems. ACM, 2004: 39-49.