Implementation and Testing of a Low-Overhead Network Synchronization Protocol

Skye R. Kowalski, Timothy M. Christman, Andrew G. Klein
Department of Engineering and Design, Western Washington University
Bellingham, WA 98225
{kowalsd2, chris70, andy.klein}@wwu.edu

Mitchell W.S. Overdick, Joseph E. Canfield
PACCAR Technical Center
Mt. Vernon, WA 98273
{mitchell.overdick, joe.canfield}@paccar.com

D. Richard Brown III
Department of ECE, Worcester Polytechnic Inst.
Worcester, MA 01609
drb@wpi.edu

***Abstract*—This paper describes a radio frequency implementation of precise synchronization between two software defined radios without the use of timestamps. Contemporary synchronization protocols mostly rely on exchanging digital timestamps between nodes. The finite precision of these digital timestamps limits the degree of synchronization achievable, and the additional overhead of sending timestamps reduces data throughput. Previous experimental work demonstrates that synchronization information can instead be conveyed at the physical layer through pairwise message exchanges between nodes. By forgoing the use of digital timestamps the problem of finite precision error can be avoided and the accuracy of synchronization is only limited by fundamental bounds of delay estimation. Our implementation was successful in synchronizing two Ettus N210 Software Defined Radios to within a standard deviation of 8.18 ns.**

TABLE OF CONTENTS

1. INTRODUCTION

Recently, there has been great interest in techniques such as distributed MIMO and beamforming [1]. These techniques permit multiple devices to coordinate their communication and pool their antenna resources, thereby forming a virtual antenna array. Such distributed transmission techniques require precise synchronization between devices to permit carrier phase alignment. The past few decades have seen the development of a number of synchronization protocols such as Network Time Protocol [2], the Global Positioning System [3], and several lightweight synchronization protocols for use in resource constrained sensor networks [4]. A common theme with existing protocols is that they rely on exchanging digital timestamps between devices. Finite precision error resulting from digital time stamps inherently limits the accuracy of such synchronization protocols. Furthermore, precise synchronization requires frequent exchanging of timing information. Exchanging digital timestamps at these speeds can present a prohibitive amount of overhead to network channels between devices. To eliminate the significant issues of conventional synchronization for this application, a timestamp-free synchronization protocol was proposed that performs synchronization implicitly at the physical layer through pairwise message exchanges [5]. Theoretical analysis has shown that this leads to precise, accurate clock synchronization between two devices, and results in lighter load on a network when compared with conventional timestamp-based synchronization approaches.

In the past, this protocol has been used to synchronize various DSP platforms with mean errors well below the period of the carrier frequency. Previously, an implementation of this protocol at acoustic frequencies using a Texas Instruments TMS320C6713 DSK [6], and a subsequent implementation at radio frequency (RF) using an Ettus E310 software defined radio [7] were tested. While these experiments were encouraging, the former did not operate at radio frequencies, while the latter Ettus E310 implementation was only able to achieve 70 ns synchronization due to practical limitations in the architecture of the software defined radio (SDR). That is, the delay estimation had to be performed on the baseband signal after downconversion. It is well known from the Cramer-Rao lower bound that the achievable accuracy of delay estimators on such signals are limited when compared to performing delay-estimation on sampled RF signals [8].

This paper describes a real-time implementation of the timestamp-free network synchronization protocol implemented on an Ettus N210 software defined radio, in which the delay estimator operates directly on the sampled RF signal. This leads to synchronization performance that is significantly improved with respect to the previous RF implementation of this protocol in Overdick *et al*. [7]. Experimental results presented in this paper show that the protocol can synchronize two radios to within 8.18 ns of precision while accounting for the presence of propagation delay. This experimental study serves to demonstrate the improved synchronization performance of the timestamp-free protocol in a software radio architecture that employs RF passband signals rather than baseband signals in performing delay estimation.

978-1-5386-2014-4/18/$31.00

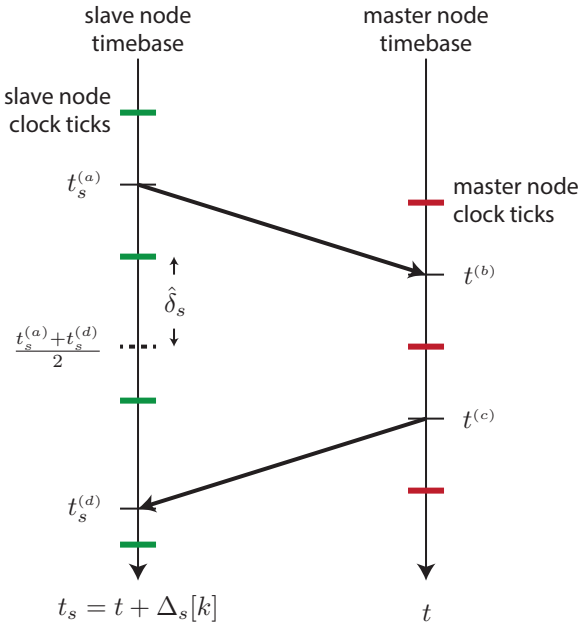


Figure 1. Timestamp-free synchronization bidirectional signal exchange.

2. Timestamp-Free Synchronization Protocol

Figure 1 shows the interactions between a slave node and the master node using the timestamp-free synchronization protocol. The time-varying clock offset at the slave node with respect to the master node is denoted as $\Delta_s[k]$ and local time at the slave node is denoted as

$$t_s = t + \Delta_s[k]$$

where t is the reference time corresponding to the local clock at the master node. The timestamp-free synchronization protocol begins with the slave node transmitting a signal to the master node at arbitrary local time $t_s^{(a)}$. The signal arrives at the master node at local time

$$t^{(b)} = t_s^{(a)} - \Delta_s[k] + \tau_s$$

where $\Delta_s[k]$ is the current clock offset of the slave node with respect to the master node and τ_s is the propagation delay between the slave node and the master node. The master node then transmits a signal back to the slave node at time $t^{(c)}$ where $t^{(c)}$ is selected such that

$$\frac{t^{(b)} + t^{(c)}}{2} \pmod{T_0} = 0 \tag{1}$$

where T_0 is master node *clock tick period.* Note that, unlike the usual sender/receiver synchronization protocol, e.g. [9], no timestamps are exchanged between the nodes. Implicit timing information is embedded in the master node's response to the slave node by selecting $t^{(c)}$ so that a local clock tick the master node is centered between $t^{(b)}$ and $t^{(c)}$. Assuming a reciprocal channel and transmitter/receiver chain, the slave node receives the reply signal from the master node at local time

$$t_s^{(d)} = t^{(c)} + \Delta_s[k] + \tau_s.$$

The slave node can then estimate its clock tick offset with respect to the master node by calculating

$$\hat{\delta}_s = \left(\frac{t_s^{(a)} + t_s^{(d)}}{2} \right)_{T_0} \tag{2}$$

where the notation $(z)_{T_0}$ corresponds to wrapping z to the interval $[-T_0/2, T_0/2)$. The offset estimate in (2) can be used directly for immediate clock offset correction at the slave node or as an input to a filtering algorithm to correct both clock offsets and drifts.

3. Delay Estimation for Bandpass RF Signals

This section provides an overview of the delay estimator used in the timestamp-free synchronization protocol. Previous experimental work that reported on the accuracy of the timestamp-free protocol on RF signals used an Ettus E310 embedded software radio [7]. Due to the front-end architecture of the E310 software radio, estimation on a baseband signal was required and made use of an approach based on quadratic interpolation. Because the Ettus N210 radios employed in this paper permit direct sampling of an RF signal, we assume the synchronization pulse is a bandpass signal of the form

$$s(t) = \cos(\Omega_0 t)u(t)$$

where $u(t)$ is a bandlimited signal such that $U(\Omega) = 0$ for all $|\Omega| \geq \Omega_0$. The synchronization pulse in discrete time can be expressed as

$$\begin{aligned} s[\ell] &= [\cos(\Omega_0 t)u(t)]_{t=\ell T_s} \\ &= \cos(\omega_0 \ell)u[\ell] \end{aligned}$$

where $\omega_0 = \Omega_0 T_s$ is the normalized carrier frequency in radians/sample.

The discrete-time observation with unknown delay τ can be expressed as

$$\begin{aligned} y[\ell] &= [\cos(\Omega_0(t-\tau))u(t-\tau)]_{t=\ell T_s} \\ &= \cos(\omega_0(\ell - \tau f_s))u(\ell T_s - \tau) \end{aligned}$$

for $\ell = 0, \ldots, L-1$ where L is the number of samples in the observation. Standard cross-correlation techniques with the template waveform $s[\ell]$ can be used to generate a quantized delay estimate $\hat{\ell} \in \mathbb{Z}$. The accuracy of this quantized delay estimate is limited, however, by the sampling rate of the delay estimator.

To refine the delay estimate, we define

$$\begin{aligned} s_i[\ell, \hat{\ell}] &= \cos(\omega_0(\ell - \hat{\ell}))u[\ell - \hat{\ell}] \\ s_q[\ell, \hat{\ell}] &= \sin(\omega_0(\ell - \hat{\ell}))u[\ell - \hat{\ell}] \end{aligned}$$

for $\ell = 0, \ldots, L-1$ where $\hat{\ell}$ is the quantized delay estimate.

We can then compute

$$
\begin{aligned}
z_i[\hat{\ell}] &= \sum_{\ell=0}^{K-1} y[\ell] s_i[\ell, \hat{\ell}] \\
&= \frac{1}{2} \sum_{\ell=0}^{K-1} \Big(\cos(\omega_0(2\ell - \tau f_s - \hat{\ell})) + \cos(\omega_0(\tau f_s - \hat{\ell})) \Big) \\
&\quad \times u(\ell T_s - \tau) u[\ell - \hat{\ell}] \\
&\approx \frac{1}{2} \cos(\omega_0(\tau f_s - \hat{\ell})) \sum_{\ell=0}^{K-1} u(\ell T_s - \tau) u[\ell - \hat{\ell}]
\end{aligned}
$$

where the approximation results from the fact that $\sum_{\ell=0}^{K-1} \cos(\omega_0(2\ell - \tau f_s - \hat{\ell})) u(\ell T_s - \tau) u[\ell - \hat{\ell}] \approx 0$. Similarly, we can calculate

$$
\begin{aligned}
z_q[\hat{\ell}] &= \sum_{\ell=0}^{K-1} y[\ell] s_q[\ell, \hat{\ell}] \\
&\approx \frac{1}{2} \sin(\omega_0(\tau f_s - \hat{\ell})) \sum_{\ell=0}^{K-1} u(\ell T_s - \tau) u[\ell - \hat{\ell}]
\end{aligned}
$$

We can then compute

$$
\begin{aligned}
\theta &= \tan^{-1} \left(\frac{z_q[\hat{\ell}]}{z_i[\hat{\ell}]} \right) \qquad (3) \\
&\approx \tan^{-1} \left(\frac{\sin(\omega_0(\tau f_s - \hat{\ell}))}{\cos(\omega_0(\tau f_s - \hat{\ell}))} \right) \\
&= \omega_0(\tau f_s - \hat{\ell}).
\end{aligned}
$$

The refined delay estimate can then be computed as

$$
\hat{\tau} = \left(\hat{\ell} + \frac{\theta}{\omega_0} \right) T_s
$$

where θ is calculated according to (3). Note that $\hat{\ell}$ represents the integer, sample-level delay, which in the sequel we term the *coarse estimate*; meanwhile, $\frac{\theta}{\omega_0}$ represents the fractional-sample delay, so we term it the *fine estimate*. Again, the coarse estimate can be be computed with standard cross-correlation techniques.

4. Passband Radio-Frequency Implementation

To demonstrate the superiority of passband over baseband signals with respect to delay estimation, the timestamp-free synchronization protocol was implemented with radio-frequency signals on Ettus N210 SDRs with wired connections between devices. Unlike the Ettus E310 SDRs which perform analog downconversion before sampling, the Ettus N210 SDRs are able to sample the incoming RF signal directly, which, according to the Cramer-Rao lower bound [8], leads to increased synchronization accuracy. To allow sufficient time for computation, a relatively low sampling frequency of 250kHz was selected for this implementation. The USRP devices operate in block fashion, where each time a receive command is issued, a block of samples is returned to the calling routine. Similarly, transmission also works in block fashion, where transmissions are scheduled in advance by passing blocks of data with a transmit command. In addition, the desired time of transmission is provided to the transmit command, and needs to be sufficiently far in the future to prevent sampling under-runs. In this implementation, we used the default block size of 1000 samples and a transmission delay of fifteen blocks.

Two Ettus USRP N210s were used to implement the protocol: one as the master node, a second as the slave node. In addition, a third N210 was used as a measurement device to determine the resulting clock offset. Each of the nodes' software was implemented in C++ using the USRP Hardware Driver (UHD) provided by Ettus, and the USRP's were connected to host computers with Core i7-6700k processors. As the N210's are part of Ettus' "networked" series of SDRs, the signal processing and logic needed to implement the synchronization protocol were run in real-time on the host computers, and not on the N210 SDR's themselves. The result of this is processing latency that is significantly larger as compared to the previously considered stand-alone embedded E310 platform which performs all computation on the device itself [7].

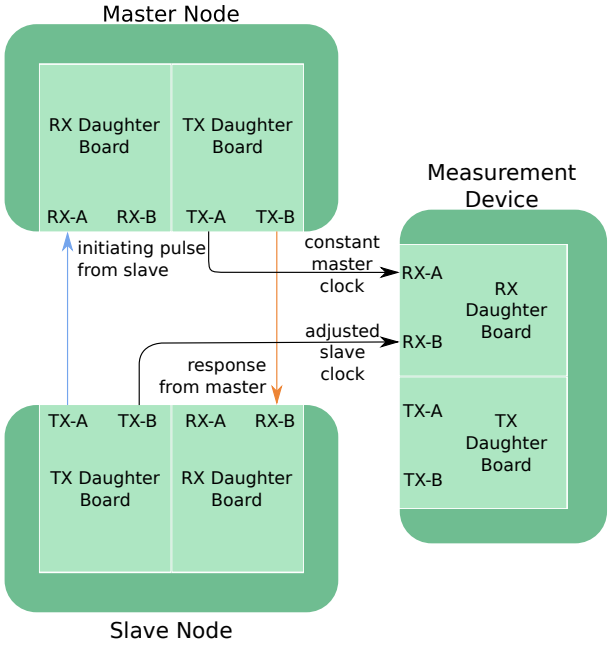


Figure 2. Diagram of the connections between the N210s used for the master node, slave node, and measurement device. Each N210 has an independent local oscillator.

A diagram of the timestamp-free synchronization test setup is shown in Fig. 2. Each of the three USRPs utilized two Ettus LF daughterboards which provide access to the 14-bit ADC and 16-bit DAC on the N210s. One daughter-board was set to receive incoming signals while the other was used to transmit. Each daughterboard contains two channels; for instance, the LFTX transmitting daughterboard has channels TX-A and TX-B, while the LFRX receiving daughterboard has channels RX-A and RX-B. The basic operation is that the slave node transmits an initial pulse which is received by the master node. Once the master node detects that it

has received a pulse, it transmits a time-reversed copy of the received pulse which is received by the slave node. In order to externally measure the accuracy of synchronization achieved, each node also transmits a clock signal on its remaining TX channel. The master node transmits its clock signal as driven by its internal oscillator while the slave node adjusts its clock in attempt to transmit simultaneously with the master node. In addition to sending the pulses to the measurement node, the pulses are displayed on the oscilloscope for qualitative evaluation and debugging purposes.

Signals used in this implementation were modulated sinc pulses as described in Section 3, with a carrier frequency equivalent to half of the Nyquist sampling rate of the master and slave nodes, with baseband bandwidth chosen to utilize a large proportion of the available spectrum while avoiding aliasing. For simplicity the same signals were used for exchanges between the master and slave nodes, and for clock pulses sent to the measurement device. Figure 3 shows a 4 ms recording of typical synchronized clock pulses transmitted by the slave and master nodes.

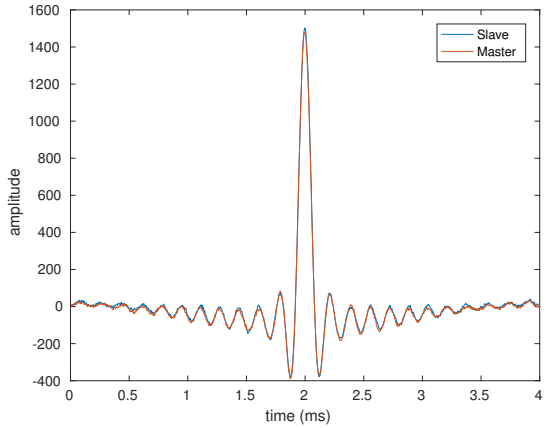


Figure 3. Example recording of clock pulses sent by master and slave nodes.

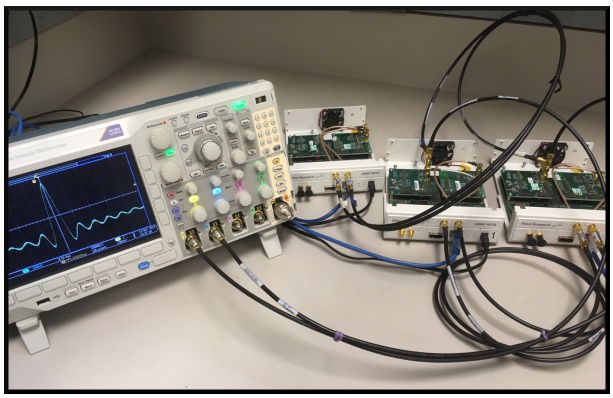


Figure 4. Experimental test setup.

A state-diagram of the master node is shown in Fig. 5. The TX-B and RX-A signal paths at the master node facilitate all communication of sync pulses to and from the slave node while TX-B facilitates the clock output. Each signal path on the master node is controlled synchronously within the software, but is displayed on the diagram asynchronously for simplicity. Cross-correlation is used to detect the presence of a pulse. When the magnitude of the cross-correlation breaks a predefined threshold, the master node schedules the time reversed transmission of the received waveform to the slave node. The transmissions at the master node are scheduled 4 blocks in advance to accommodate the latency between the host computer and the radio.

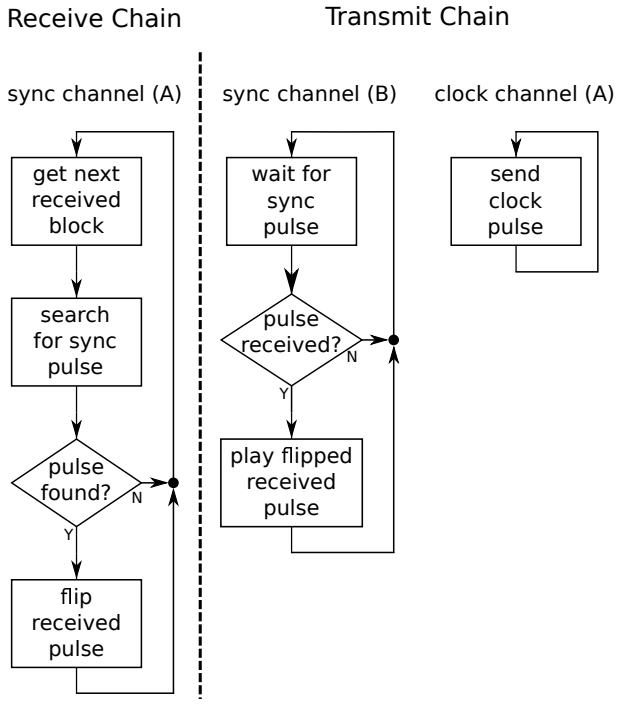


Figure 5. Flow diagram of master node operation.

A state-diagram of the slave node is shown in Fig. 6. TX-A and RX-B facilitate all communication of sync pulses to and from the master node and TX-B outputs the synchronized clock output. Because the slave node adjusts its clock based on the master's timed response, the implementation complexity of the slave node is significantly higher than the master node. When searching, the slave node performs cross-correlation on every receive block in order to locate the sample at which the peak of the sinc pulse was received. As with the master node, the slave node computes the magnitude of the cross-correlation when searching for a pulse to confirm that the threshold is surpassed. When the detection threshold is broken and a pulse is detected the slave node sets a flag for calculation, and saves both the position of the peak in the most recent block of received samples as well as the complex value resulting from the cross-correlation. The calculation step uses the peak position as the coarse delay estimate, and the fine delay estimator in equation (3) uses the peak cross-correlation value to calculate the fractional delay.

The resulting total delay (coarse and fine) is used at the slave node to calculate an estimate of the master node's clock offset and drift using a smoothing filter. The slave node maintains estimates of the master node's clock offset and drift that are updated every block, and are subsequently used to generate a new delayed pulse template which is transmitted on channel TX-B of the slave node. When a new delay offset observation is available at the slave node, the slave node's prediction of the master clock offset and drift is refined. The smoothing

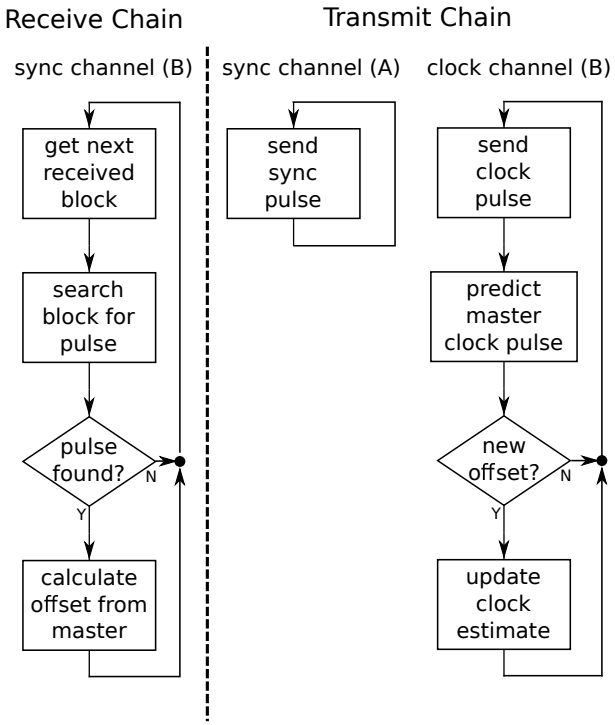


Figure 6. Flow diagram of slave node operation.

filter makes use of the standard two-state (offset and drift) clock model [10], and is implemented as a pseudo-Kalman filter with a static Kalman gain; this results in a performance penalty with respect to a true Kalman filter; however, the static gain also yields a reduction in complexity by avoiding computation of the covariance matrix that updates the gains to appropriate magnitudes as the process approaches steady-state. Instead, an optimized set of gains are experimentally chosen using the approach described in [6] and ample time is given for the system to approach steady-state.

The slave node's predictions of the clock offset and drift estimates at time k are denoted $\Delta_{k|k-1}$ and $D_{k|k-1}$, and are initially set to zero and one respectively. In each block, the pseudo-Kalman prediction is computed via

$$\begin{bmatrix}\Delta_{k|k-1}\\ D_{k|k-1}\end{bmatrix} = \begin{bmatrix}1 & T\\ 0 & 1\end{bmatrix}\begin{bmatrix}\Delta_{k-1|k-1}\\ D_{k-1|k-1}\end{bmatrix} \tag{4}$$

where T is the time-step, equal to 1000-samples in our case. We denote $z[k]$ as the new clock offset observation. When new observations arrive, we compute the innovations $\tilde{y}[k]$ which correspond to the difference between the predicted clock offset from the Kalman filter and the new clock offset observation at the time of the new clock offset observation, or

$$\tilde{y}[k] = z[k] - \Delta_{k|k-1}.$$

This innovation process is subsequently multiplied by the static Kalman filter gains K_1 and K_2, and used to adjust the slave node's prediction of the master node's clock offset and drift via the update equation

$$\begin{bmatrix}\Delta_{k|k}\\ D_{k|k}\end{bmatrix} = \begin{bmatrix}\Delta_{k|k-1}\\ D_{k|k-1}\end{bmatrix} + \begin{bmatrix}K_1\\ K_2\end{bmatrix}\tilde{y}[k]. \tag{5}$$

5. EXPERIMENTAL RESULTS

This section summarizes our experimental results in implementing the timestamp-free synchronization protocol using bandpass RF signals. First, we investigate the N210's ability to estimate offsets between signals by synthesizing clock signals with known offsets, and comparing these known offsets with the measurement device's estimate. Second, we test the N210 when measuring two signals from the same source, isolating the measurement integrity from the transmitter's induced error. Third, we report on the process of calibrating the pseudo-Kalman filter used to refine the output of the delay estimator. Finally, we report on synchronization data collected. The clock pulses sent out by the master and slave nodes were recorded by the measurement device while the offsets between these signals were calculated after runtime in MATLAB.

Due to constraints mentioned in the discussion section below, the master and slave nodes were not able to operate at a rate higher than 250 kilo-samples per second without running out of time to complete each operation before the next block. The measurement device was able to operate at 5 mega-samples per second because processing of the data recorded by the measurement device was not done in real-time, but instead was conducted in MATLAB after recording each session. Clock pulses were sent from each node to the measurement device in 1000 sample blocks (equivalent to 4 ms in duration) back-to-back. Synchronization signals exchanged between nodes were sent at a period of 15 blocks (60 ms) to allow enough time to compute cross-correlation, and to allow for transmission delays between nodes.

Characterizing the Precision of the Transmission and Measurement Device

In order to validate the results derived from sampled data it is imperative to know the precision and accuracy between the transmission and measurement device. To accomplish this a single N210 was configured to send clock pulses on both of its TX channels, but with an adjustable, intentional offset between the signals. The measurement device recorded these signals and offsets between pulses were calculated in MATLAB. These calculated offsets were compared to the true offsets as described by the test. Signals were transmitted with delay offsets ranging from ± 2 μs in increments of 400 ns, corresponding to ± 0.5 sample periods at an increment of 0.1 samples. Signals were transmitted at each offset 250 times each session, and results were averaged over 5 sessions. Figure 7 shows a plot of the error statistics of a run of this test. The standard deviation of the error over all offsets from ± 0.5 fractional samples was found to be 5.97 ns. This was a limiting factor in our experimentation as the coarseness of this measurement determined the minimum delay offset the receiver node was able to accurately detect.

Characterizing the Precision of Received Pulses with a 'T-Junction'

Following a similar model of the precision test, a radio was configured to send simultaneous pulses to the device under test (DUT) along a T-Junction connector. This way, the transmission is duplicated along two wires, effectively transmitting two synchronized pulses and eliminating potential error introduced by the transmission device. The error from

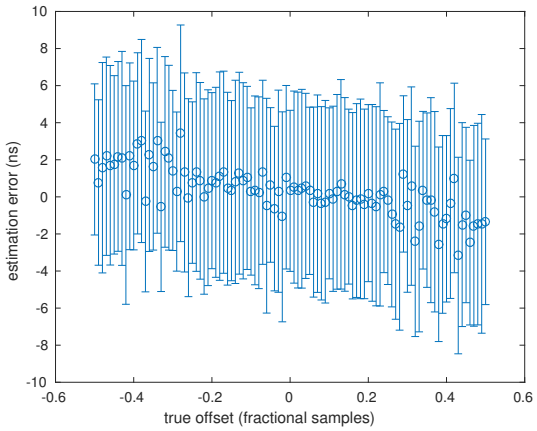


Figure 7. Statistics showing measurement device accuracy

each measurement device (labeled A,B, and C) is isolated and quantified for each radio with this process. The results of this experiment are reported in Table 1.

DUT	mean$[\tilde{y}[k]]$	std$[\tilde{y}[k]]$
A	-0.51 ns	8.54 ns
B	0.55 ns	7.16 ns
C	-0.82 ns	6.77 ns

Table 1. Summary of T-Junction precision results

Selection of Pseudo-Kalman Filter Gains

As described earlier in Section 4, and by equations (4) and (5), we implemented a simplified, pseudo-Kalman filter to smooth and correct for measurement error in delay offset estimates. Due to the use of a reduced-complexity pseudo-Kalman filter without an adaptive Kalman filter gain, it was necessary to determine static filter gains which provided the best smoothing performance and minimized prediction error. The statistics of prediction errors calculated by the slave node during runtime can serve as an indicator of performance in absence of ground truth. To take advantage of this we configured the slave node to record delays between sent and received signals calculated during runtime. We then used MATLAB post-processing to apply pseudo-Kalman filters with different gains and simulate the resulting prediction error statistics as if the data had been filtered in real time. By doing this we were able to determine which filtering coefficients minimized the resulting prediction error. Filtering coefficients were chosen to minimize error over the last minute of a 5 minute recording, allowing the filter to take advantage of prediction error memory accrued during the beginning of the session.

Synchronization Accuracy Test

To evaluate the effectiveness of the timestamp-free synchronization protocol we implemented the procedure described in the implementation section above. To control for process parameter variation across devices, the accuracy of synchronization was tested with each SDR configured to operate as each role in the protocol. A total of 6 tests were conducted with each of the three different SDRs serving alternately as the measurement device, master node, or slave node. During each test the master and slave nodes ran the protocol for 3 minutes prior to recording to allow the nodes to synchronize. The measurement device then recorded two minutes of clock signals (about 30,000 pulses) sent from each node, and this recording was used to evaluate the accuracy of synchronization between the node. Prediction error statistics and master/slave clock pulses were recorded in each test. The results of these tests are summarized in Table 2. The mean standard deviation across all configurations was 13.2 ns. In the next section, we will discuss the implication and limitations of these results.

M/S	mean$[\tilde{y}[k]]$	std$[\tilde{y}[k]]$
A/B	4.66 ns	8.18 ns
B/A	-3.85 ns	8.41 ns
C/A	2.72 ns	16.3 ns
A/C	1.77 ns	13.4 ns
B/C	-6.95 ns	15.9 ns
C/B	-3.76 ns	16.7 ns

Table 2. Summary of clock pulse synchronization results

6. DISCUSSION

This work represents a significant improvement with respect to the previous RF implementation of the protocol on Ettus E310 SDRs [7]. Results indicate a worst-case synchronization error with a standard-deviation of 16.7 ns, corresponding to about 0.1% of the period of the carrier frequency. It is worth mentioning that the inaccuracy of our measurement device has been shown to be comparable to the degree of synchronization achieved. Without the ability to measure delay offsets between signals to a finer resolution, our reported results will continue to be dominated by the measurement error in our estimation device. There were several other key factors that limited the degree of synchronization, including network latency, computational complexity, and error compounded from finite-precision effects, all of which we now explore in more detail.

In our implementation the data rate of the receiver is limited by the processing delay between the radio and the computer. In addition to this, due to the nature of the USRPs, all transmissions must be scheduled in advance. Accounting for the clock drift that occurs from the time when the transmission is scheduled to the time when it actually transmits is simple, but the need to accommodate this additional latency does add potential error to the system.

Additionally, the complexity of the computation, and requirement to operate in real-time limit the possible sampling rate of the USRP N210s. By far the most computationally intensive operation in the protocol is the repeated cross-correlations (i.e. receive filtering) that is performed at both the master and slave nodes. If cross-correlation does not finish executing before the next block of samples is ready, then samples are lost and the device reports underruns. This limits the degree of synchronization achievable as lower sampling frequencies require lower bandwidth signals which limit the degree of delay estimation accuracy as given by the Cramer-Rao Lower Bound [8].

One more item that negatively impacted synchronization performance was the sensitivity of the arctangent-based fine delay estimator in (3) when fractional delays approached a half sample. Specifically, we observed that small errors from finite precision effects sourced from the ADC created small errors in the cross-correlation, which subsequently accumulated to larger errors in the fine delay estimator.

7. CONCLUSION

This work demonstrates the synchronization accuracy and implementation challenges of implementing timestamp-free synchronization protocol at RF using sampled passband signals instead of modulated baseband signals. An implementation of the protocol using such signals is presented along with with experimental results and analysis. Experimental results show that the protocol is able to achieve synchronization to within a tenth of a percent of the period of the carrier frequency over 4 ms prediction intervals.

Future work could improve the accuracy of the timestamp-free synchronization protocol by addressing issues mentioned in Section 6. Specifically, the sampling rate of our devices is currently bottlenecked by the time it takes the CPU to compute cross-correlation between template and received waveforms. This latency could be reduced by offloading computation from the CPU of the host computer to the GPU using NVIDIA's CUDA API. Alternatively, when the development of Ettus' RFNoC technology is sufficiently mature, much of the computation (i.e. the cross-correlation) can be offloaded to the FPGA on the Ettus radios. Future work could also extend this implementation to operate in a wireless setting with multipath and additional noise, or to work with more than two synchronizing nodes. In a fully wireless, multiple-node scenario, one could investigate use of the timestamp-free network synchronization protocol in combination with consensus-based techniques [11] for operation in a distributed setting without the need for a centralized master/slave approach.

All source code for this experiment is available at —
`https://github.com/agklein1/tsfreesync-n210`

ACKNOWLEDGMENTS

This work was supported by a Research Experience for Undergraduates (REU) Supplement to National Science Foundation award CCF-1319458.

REFERENCES

[1] R. D. Preuss and D. R. B. III, "Two-way synchronization for coordinated multicell retrodirective downlink beamforming," *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5415–5427, Nov 2011.

[2] D. Mills, "Internet time synchronization: the network time protocol," *IEEE Trans. Commun.*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.

[3] W. Lewandowski, J. Azoubib, and W. Klepczynski, "GPS: primary tool for time transfer," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 163–172, Jan 1999.

[4] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Netw.*, vol. 18, no. 4, pp. 45–50, Jul.–Aug. 2004.

[5] D. R. Brown and A. G. Klein, "Precise timestamp-free network synchronization," in *Information Sciences and Systems (CISS), 2013 47th Annual Conference on*. IEEE, 2013, pp. 1–6.

[6] M. Li, S. Gvozdenovic, A. Ryan, R. David, D. R. Brown, and A. G. Klein, "A real-time implementation of precise timestamp-free network synchronization," in *2015 49th Asilomar Conference on Signals, Systems and Computers*, Nov 2015, pp. 1214–1218.

[7] M. Overdick, J. Canfield, A. G. Klein, and D. R. Brown, "A software-define radio implementation of timestamp-free netwrok synchronization," in *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 1193–1197.

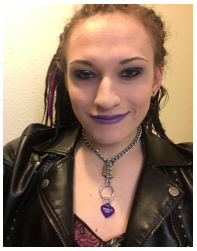
[8] A. Weiss and E. Weinstein, "Fundamental limitations in passive time delay estimation–Part I: Narrow-band systems," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 31, no. 2, pp. 472 – 486, April 1983.

[9] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings ACM SenSys 2003*. ACM New York, NY, USA, Nov. 2003, pp. 138–149.

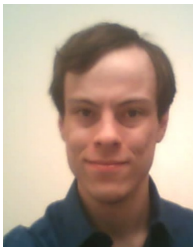
[10] L. Galleani, "A tutorial on the two-state model of the atomic clock noise," *Metrologia*, vol. 45, no. 6, p. S175, 2008.

[11] D. R. Brown, A. G. Klein, and R. Wang, "Monotonic mean-squared convergence conditions for random pairwise consensus synchronization in wireless networks," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 988–1000, Feb. 2015.

BIOGRAPHY



Skye R. Kowalski *is currently pursuing a B.S. in electrical engineering from WWU and expects to graduate in June 2018. Her current research activities include working with software-defined radios, and clock synchronization. She wishes to continue her studies in Embedded and Cyberphysical systems in application of IoT sensor networks.*



Timothy M. Christman *is currently pursuing a B.S. in electrical engineering from WWU and expects to graduate in June 2018. His current research activities include software-defined radios, synchronization, and fault line detection in power systems. He hopes to study or work with applications in machine learning in the future.*

Andrew G. Klein *received the B.S. degree from Cornell University, Ithaca, NY, USA, the M.S. degree from the University of California, Berkeley, CA, USA, and the Ph.D. degree from Cornell University, all in electrical engineering. Previously, he was an Assistant Professor with the Worcester Polytechnic Institute, Worcester, MA, USA, from 2007 to 2014, and he was a Post-Doctoral Researcher with Suplec/LSS, Paris, France, from 2006 to 2007. He joined the Department of Engineering and Design, Western Washington University, Bellingham, WA, USA, in 2014, where he is currently an Associate Professor.*

Mitchell W.S. Overdick *received his B.S. degree in Electrical Engineering at WWU in 2017, and was a member of the ASPECT Lab. Currently, he works as an embedded systems validation engineer at PACCAR in Mount Vernon, WA. His research interests include Embedded Systems, Signal Processing, Machine Learning, and Automotive.*

Joseph E. Canfield *works as an electrical hardware validation engineer at the PACCAR Technical Center in Mount Vernon, WA. His research interests include radio communications and analog audio technology.*

D. Richard Brown III *received the B.S. and M.S. degrees from the University of Connecticut in 1992 and 1996, respectively, and the Ph.D. degree from Cornell University in 2000, all in electrical engineering. From 1992 to 1997, he was with General Electric Electrical Distribution and Control. He was a Faculty Member with the Worcester Polytechnic Institute, Worcester, MA, USA, in 2000. He was a Visiting Associate Professor with Princeton University from 2007 to 2008. Since 2016, he has been with the Computing and Communication Foundations Division, National Science Foundation, as a Program Director.*