WATCH: A Distributed Clock Time Offset Estimation Tool for Software-Defined Radio Platforms

Cassie Jeng
Electrical & Systems Engineering
Washington University in St. Louis
St. Louis, Missouri, United States
jeng.c@wustl.edu

Neal Patwari
McKelvey School of Engineering
Washington University in St. Louis
St. Louis, Missouri, United States
npatwari@wustl.edu

***Abstract*—The synchronization of clocks used at different transceivers across space is of critical importance in next-generation wireless networks. Experimental testbeds must provide synchronization services so that users can test new efficient communication and accurate localization technologies; yet reconfigurability can sometimes introduce unintended asynchronicities. We introduce WATCH, a tool to monitor the time synchronization of nodes in a geographically distributed software-defined radio (SDR) testbed. WATCH estimates clock differences using cross-correlation between transmitted and received packets and considers both propagation delay and noise in the channel. We model pairwise measurements as a linear function of the difference between the nodes' local clocks from the network's global clock, as well as each node's additional delay for packet transmission. This model allows WATCH to simultaneously estimate all of the clock offsets and transmit delays. We compare using transmissions of modulated data packets and pseudo-noise (PN) code signals to experimentally test and verify the performance of WATCH on the Platform for Open Wireless Data-driven Experimental Research (POWDER), where it detected a firmware problem that led to a time synchronization error between SDR nodes on the platform.**

***Index Terms*—time synchronization, wireless networks, software-defined radio, experimental wireless testbed**

I. Introduction

Clock time synchronization is imperative for mobile and wireless networks. From improving network efficiency [1] to reducing channel overhead and computation, time synchronization is necessary for effective and reliable communication across a distributed wireless network [2]. Deployed time-division duplex (TDD) cellular protocols such as 5G New Radio (NR) and Citizens Broadband Radio Service (CBRS) require $< 3\mu$s synchronization [3]. Wireless technologies that use multiple distributed receivers to increase reliability and bandwidth rely on phase and time synchronization [4], [5]. Experimental testbeds must provide time synchronization services to geographically distributed SDRs, with better accuracy and precision than current needs, to enable testing of next-generation efficient wireless protocols.

This material is based upon work supported by the US National Science Foundation under Grants #1827940 and #2232464; and the PAWR Project Office.

However, distributed and reconfigurable platforms can have configuration, network, and device software issues that can cause synchronization problems. A means to monitor synchronization in the network allows users to verify that no significant timing errors are currently measured across nodes.

This paper describes experimentation with and development of WATCH on the Platform for Open Wireless Data-driven Experimental Research (POWDER), a large city-scale openly accessible SDR platform. To address the need for accurate network synchronization in wireless research, POWDER currently deploys two synchronization systems for its rooftop nodes and fixed endpoints that can operate in TDD mode for experimentation in the CBRS band [6]. One of these systems deploys the White Rabbit protocol which provides sub-nanosecond time synchronization accuracy across a network via ethernet and the precision time protocol (PTP) [7] to achieve on the order of 0.3 ns synchronization accuracy [6]. However, even when installed, the White Rabbit synchronization system can fail without indicating to platform engineers that there is anything wrong. The time synchronization of transceivers additionally requires correct configuration of the SDR and its compute node, which is a moving target due to software updates.

The development of WATCH, our distributed clock time offset estimation tool, uses POWDER and the Shout measurement framework [8] to investigate time synchronization accuracy in the network. The goal is to estimate clock offsets in the network by using the transmit (TX) and receive (RX) data from Shout, hopefully enabling future research with new time synchronization capabilities on POWDER and other SDR platforms. This paper details the transmission and reception of a digitally modulated packet in order to provide intuition about the time-delayed, attenuating, and noisy radio channel. The impact of the channel and the packet signal can lead to errors in the time delay estimate, which we elucidate using the example of a narrowband digitally modulated packet signal. This motivates the use of a PN signal for the purpose of time synchronization.

The delays between packet transmission and reception should only be due to propagation delays, which on POWDER are on the order of $1\mu s$. However, in the experiments reported in this paper, WATCH found larger delays, which we later diagnosed to be related to a firmware misconfiguration.

A. *Desired Functionality*

With ideal time synchronization, there should be almost zero lag beyond propagation delay between the transmitted and received packet, independent of which nodes are used. While POWDER deploys the White Rabbit protocol to time-synchronize its rooftop nodes, the network operators may sometimes be unaware about conditions which make the protocol fail. WATCH presents a solution to this challenge by providing an independent means to identify any asynchronous nodes by analyzing the offsets/lags in a short over-the-air experiment's received packets. The tool's resulting offset estimates show which nodes, if any, are significantly offset from the rest in the network, allowing researchers to either compensate or notify operators about the synchronization failures.

II. Methods

WATCH was developed by focusing on time-based localization and link timing between a transmitter and receiver, time synchronization, and error correction. These steps helped further understand the elapsed time during transmission in relation to each node's local time, algebraically express packet cross-correlation offset, and determine the level of time synchronization among nodes by analyzing collected network data. Every node in a distributed network of SDR-based nodes has an inherent transmit delay, as well as a local clock offset from the network global clock. The combination of these two unknowns makes it difficult to understand whether a measured receiver delay is due to one or the other, or both. Rather than measuring pair-wise delays between each pair of nodes, WATCH uses a more efficient protocol in which each node, in turn, transmits once while all other nodes measure the received signal [9]. WATCH then uses a linear model relating the measurements to all node clock offsets and transmission delays simultaneously. This model has more equations than unknowns if the number of nodes is sufficiently high. Then, WATCH solves for all node clock offsets and transmission delays.

A. *Experimental Motivation*

The linear model for these node transmission delays and local clock offsets was built using the cross-correlation index of a received packet with either the preamble of the original transmitted packet or the entire original packet, a method inspired by how a receiver detects and synchronizes to digitally modulated packets [11]. Our cross-correlation function originally used a locally reconstructed preamble to find the index of the maximum cross-correlation, corresponding to the received signal's lag in samples and, therefore, representative of the clock time offset of that TX-RX node pair. Fig. 1 shows the results of this function, plotting both where the preamble

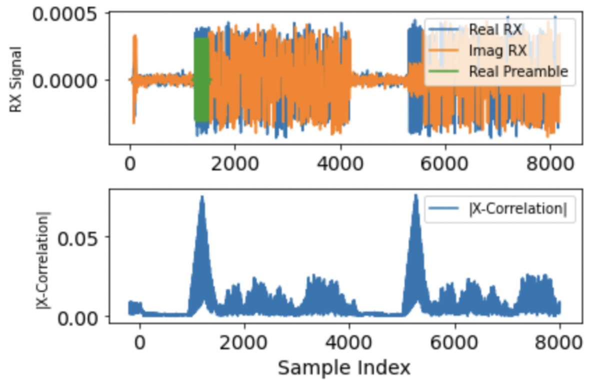


Fig. 1. Cross-correlation of received packet with the packet preamble. The top subplot depicts the received signal with the overlaid preamble at the calculated lag index. The bottom subplot depicts the resulting cross-correlation between the received signal and the preamble.

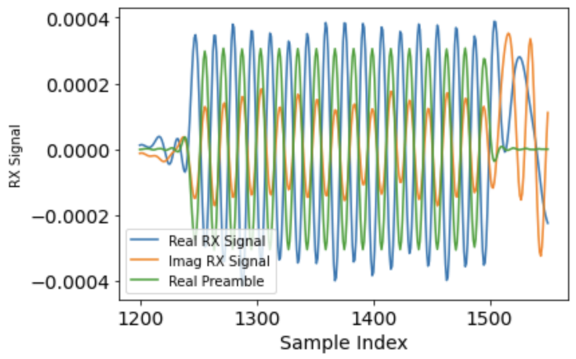


Fig. 2. Enlarged preamble match region showing, in green, the reconstructed preamble and how it aligns with the received packet.

was found and the resulting cross-correlation signal where peaks show high cross-correlation between the two signals. Fig. 2 shows an enlarged version of the green region where the reconstructed preamble matched a preamble section in the received packet.

The lags indicate an estimate for how many samples into the captured receiver samples the transmitted packet is detected. In an optimally time-synchronized network, all received packets from the same transmitter should have the same offset, within 1-2 samples, for every repeated transmission.

Our initial results did not show this low variation in estimated offset, which we identified as being related to the autocorrelation structure of the preamble in use. Initially, we had cross-correlated with the preamble signal, which resulted in multiple peaks shown in Fig. 1, representing a large range where the function might find a close match to the preamble in the received signal. Sometimes the estimated offset from the peak-finding algorithm would be of a different cross-correlation peak. We also then tested cross-correlating the raw received signal with the packet IQ sampled signal, recreated at the receiver. This worked better than cross-correlating with just the preamble, but the cross-correlation function still was

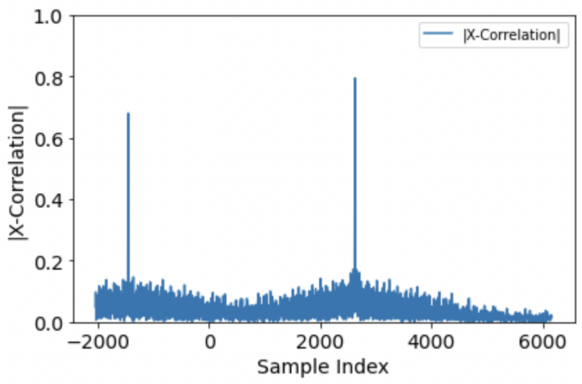


Fig. 3. PN Code absolute value of Cross-Correlation by sample showing peaks with clear maximum values.

wide, with sometimes poor performance in low SNR cases.

These sub-optimal time offset estimation results produced from running experiments with standard narrowband digitally modulated packets showed that a specially designed transmit signal is desired for optimally estimating lags. We designed new transmit signals containing only a BPSK maximum length PN code. These codes, also referred to as linear feedback shift register (LFSR) sequences, are a solution to the dichotomy of necessary randomness and pattern. A PN code *appears* random, yet its entire sequence is known and has good auto-correlation properties, fulfilling both requirements while also serving a practical purpose for digital communication systems [14]. A special characteristic of the PN code is in its sequence of runs, i.e., sequences of binary 1s and 0s. A run length is the count of 1s or 0s in that run sequence. In any PN code, the run length of 1s and 0s only differ by one bit [14]. Therefore, when a PN code is shifted by some number of bits and added to the original sequence with a modulo-2 exclusive OR (XOR) gate, the result is the original PN code shifted by some new number of bits [14]. An N-state PN code will have $2^N - 1$ bits in the generated sequence. Since WATCH uses QPSK demodulation, two unique 9-state, 511 bit PN codes were used, one for in-phase and one for quadrature, using a modified version of a simulated LFSR MATLAB function [15]. Transmitting PN codes and cross-correlating the received packet with the entire original PN code further isolated the peak areas and resulted in well-defined maximum values and clear lag index choices, as shown in Fig. 3. The time corresponding to the maximum of the cross-correlation function is referred to $\delta_{i,j}$, where i is the transmitter and j is the receiver. Note that the time is taken according to the local clock of the receiver node j.

B. Lag time vs. Node Clocks and Timing

This subsection derives the linear model that relates the measured offsets to the clock offsets and the transmission delays. We describe the message TX and RX with a prototype link timing diagram in Fig. 4. Each column refers to a clock, either the local clock of a node or the global clock. Time progresses downward in the figure; each horizontal line

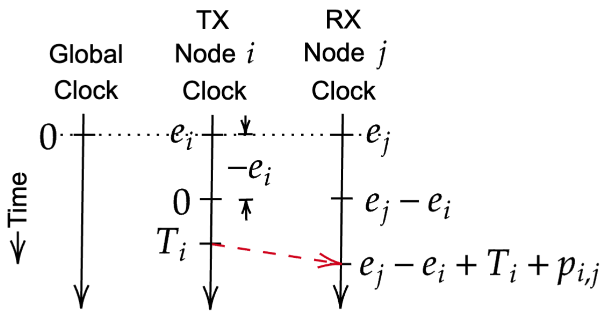


Fig. 4. The local clocks at node i and node j have their own offsets from the global clock e_i and e_j, respectively, so that at global time 0, it is time e_i at node i and time e_j at node j. Node i triggers its transmission routine at the time it believes to be time 0, which corresponds to time $e_j - e_i$ on node j's clock. There is a delay T_i before node i actually starts transmission out of its antenna. Adding propagation delay $p_{i,j}$ between nodes i and j, signal reception actually starts at time $e_j - e_i + T_i + p_{i,j}$ on node j.

represents the same time at different nodes. We refer to the link transmitter node as node i and the link receiver node as node j, and although many nodes receive the signal transmitted by node i, we show only one receiver for simplicity. Node i and j's internal clocks differ from the global clock by e_i and e_j, respectively, as shown in Fig. 4. Each transmitter has a delay between the time it is asked to transmit (local clock time 0 in Fig. 4) and when it actually emits the signal from its antenna, denoted T_i.

Using the timing diagram in Fig. 4, we can write a model for the measured offset lag time $\delta_{i,j}$.

$$\delta_{i,j} = e_j - e_i + T_i + p_{i,j} + \eta, \tag{1}$$

where $p_{i,j}$ is the propagation delay on the link (i, j), and η is noise in the measurement. The propagation delay $p_{i,j}$ is a deterministic function of the node's coordinates. Both $p_{i,j}$ and noise n in the channel are assumed to be negligible in comparison to $\delta_{i,j}$. Even when not negligible, the known $p_{i,j}$ can be subtracted out; thus in the following, we simplify by setting η and $p_{i,j}$ to zero.

Ideally, the TX and RX local clocks would be the same, as well as synchronized with the network global clock. In the timing diagram in Fig. 4, this would correlate to both $e_i = 0$ and $e_j = 0$ and would simplify (1) to $\delta = T_i$. Having $\delta = 0$, meaning ideal time synchronization, would require both the transmitter and receiver to have no delay from the global clock, $e_i = 0$ and $e_j = 0$, and for there to be no delay at the transmitter before sending the packet, $T_i = 0$.

C. Linear Matrix Equation

The timing diagram was used to create a linear matrix equation for the relationship between all nodes in the network, the experimental delays, and the estimated parameter offsets. The parameter vector θ is a list of all e_i and T_i unknowns, specifically,

$$\theta = [e_1, \ldots, e_N, T_1, \ldots, T_N]^T. \tag{2}$$

Δ is a vector of the measured δ offsets from all M measured links (in our case, $M = 42$). Using these elements, our measurement equation is,

$$\Delta = A\theta, \tag{3}$$

where A is an $M \times 2N - 1$ measurement model matrix constructed such that each row represents one of M link measurements, and N represents the number of nodes in the network.

The measurement model matrix A holds node interactions, represented as either 0, 1, or -1. The columns of matrix A represent the nodes, repeated twice. For example, column 1 and $N+1$ both correspond to node 1 in the experimental node list. A is built by row (link), by assigning a 1 to both columns corresponding to the transmitter, and a -1 to the first column corresponding to the receiver. The rest of the entries in the matrix are 0. This process sets up the left half of the matrix, which is related to the top half of the θ vector, establishing $e_i - e_j$. The right half of the A matrix, relating to the bottom half of the θ vector, defines the T_i in (1).

Because there is no available reference for the global clock time, matrix A, defined initially as a $M \times 2N$ matrix, was rank deficient. To prevent this, the first column of A was removed to use the first node as the reference point for the global clock. Therefore, all results are in relation to the first node and are assuming the first node to be synchronized with the global clock. The structure of the resulting $M \times 2N - 1$ matrix is shown in (4). The elements of the sub-matrices L_i, a $(N-1) \times (N-1)$ matrix in the first column, and J_i, a $(N-1) \times N$ matrix in the second column, are defined in (5).

$$A = \begin{bmatrix} L_1 & J_1 \\ L_2 & J_2 \\ ... & ... \\ L_N & J_N \end{bmatrix} \tag{4}$$

$$\begin{aligned} [J_i]_{k,l} &= \begin{cases} 1, & \text{if } l = i \\ 0, & o.w. \end{cases} \\ [L_i]_{k,l} &= \begin{cases} 1, & \text{if } l = i-1 \\ -1 & \text{if } l < i-1 \text{ and } k = l+1 \\ -1 & \text{if } l > i-1 \text{ and } k = l \\ 0, & o.w. \end{cases} \end{aligned} \tag{5}$$

We note that A will be rank deficient if $M < 2N - 1$. At most, the number of independent link measurements in an N node network is $N(N-1)$. This means we would not be able to estimate θ from (3) when $N < 3$.

After ensuring that A is not rank deficient, the Moore-Penrose pseudo-inverse of A, which we denote $A^\dagger$, is used to solve (3) for the unknown θ estimate vector. The pseudo-inverse is used since the A matrix is both non-square and sparse. Equation (6) shows the new, rearranged equation that allows WATCH to estimate the parameters:

$$\hat{\theta} = A^\dagger \Delta \tag{6}$$

The final $\hat{\theta}$ vector that is found using the pseudo-inverse of the node interactions matrix and the experimental offset vector gives the WATCH parameter vector estimate, containing each node's local clock offset from the global clock, e_i, and the transmission delay at each node, T_i. Our purpose in this paper is to estimate each node's e_i value, so we denote the vector $\hat{e} = [\hat{e_1}, \ldots, \hat{e_N}]^T$ as the estimates of the e_i clock offsets at each node taken from $\hat{\theta}$.

III. WATCH Experiments on POWDER

Much of the experimentation with WATCH after development was done with POWDER, an open experimental city-scale SDR platform at the University of Utah. The POWDER testbed offers, among other resources, eight rooftop base stations with SDRs, seven fixed-endpoints, white rabbit synchronization nodes at several nodes, and a framework for remote accessibility [6].

The development of WATCH builds off POWDER and its instrumental measurement framework, Shout. Shout is a suite of Python scripts that iteratively orchestrates simultaneous transmission and reception across a collection of nodes to collect wireless measurements and their associated link budget estimates [10]. This process can be conducted in any frequency band, with any transmitted signal, and any number or type of nodes, all customized by the user while designing the experiment parameters. For example, in our experiment with seven rooftop nodes, each node broadcasts the same packet to all of the other six nodes in the experimental network, sweeping the reserved frequency range in designated increments and using a fixed time and power. To gather "power-over-noise" data for each channel, Shout collects measurements with and without the active transmitter, first measuring just the channel noise and then measuring the received signal [10].

While developing WATCH, we focus on the POWDER rooftop x310 radio base stations and their associated d740 compute nodes. We use the seven available rooftop radios: Behavioral (BES), Browning, Friendship Manor (FM), Hospital, Honors, Medical Tower (SMT), and USTAR. Specific aspects of a default Shout experiment, such as the pre-loaded JSON parameters file and the `meascli.py` Shout script, were adjusted to specify the frequency range to 3450-3460 Hz, 250 kHz sampling rate, node names, and PN code packet to transmit as listed in Appendix B of [12]. Additionally, the script was modified to enable external clock using the White Rabbit (WR) time synchronization systems, an Ethernet-based innovation for sub-nanosecond synchronization in large distributed systems [16]. However, there are still synchronization performance challenges in platforms like POWDER because nodes reside in a realistic environment where they are susceptible to aspects like temperature fluctuations [16], and network, hardware, and software misconfigurations. For WATCH, packets were transmitted four times per each transmitter and receiver pair via Shout, and in the lag A matrix, $M = 42$ and $N = 7$ for all experiments. All lag data was collected using the same process, as outlined in the `README.md` document of the WATCH Repository [13].

The results, an estimation vector $\hat{\theta}$, were formulated into the linear matrix equation and node interaction matrix A to produce experimental offset calculations. Knowing the offset of each node's local clock from the local reference node can inform the researcher whether the nodes in their experiment are time-synchronized or not, and therefore, whether some additional adjustment is needed.

IV. Results

We observed two significant network behaviors from the various WATCH experiments run via Shout. First, software misconfigurations in the network SDRs or in Shout itself can cause the nodes to operate via only their internal clocks and thus behave asynchronously. When this occurred, the results from WATCH showed distinct groups of nodes with similar time synchronizations within each. An example experiment where this was detected resulted in FM, Hospital, Honors, and USTAR having similar lags to each other, while Browning, SMT, and BES had similar lags to one another but different than the first group. Enabling White Rabbit on all the nodes allowed all the TX-RX pairs to result in lags within the expected 1-2 sample offset from one another.

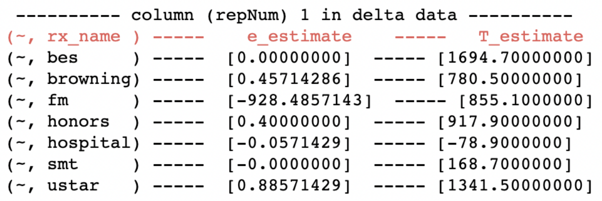
FM was the only node that, despite having WR enabled, remained differing from the other nodes by a 930 sample additional lag. A 930 sample offset in the network corresponds to a 3.72 ms difference at the nodes (corresponding to a path length of 1100 km!) which cannot be attributed to the propagation delay between nodes, and therefore must be related to the time synchronization in the network. Equation 7 uses the 930 sample difference and the 250 kHz sampling rate to calculate the time offset in milliseconds.

$$\left(\frac{930}{1}\right)\left(\frac{1 \text{ s}}{250000}\right)\left(\frac{1000 \text{ ms}}{1\text{s}}\right) = 3.72 \text{ ms} \tag{7}$$

We found that this was due to a firmware misconfiguration on the FM rooftop node that caused it to sample PPS at the clock rising edges instead of the falling edges that are tracked running the stock firmware on the other POWDER nodes. WATCH detected this misconfiguration by producing network time synchronization results showing only one node significantly offset from the expected propagation delay and channel noise lags, informing the POWDER network operations team of the differing firmware on FM.

Fig. 5 shows the output from a WATCH experiment running WR and detecting the firmware misconfiguration in FM. All other nodes are offset less than 1 sample from each other, referring to the e_i values, and FM is offset from the rest by approximately 930 samples. Since we do not know the global clock time, all resulting e_i values are in relation to a chosen reference node, BES, while still showing how synchronized the network is among nodes. Negative values in the figure refer to clocks that are running earlier than BES.

The WATCH results, node e_i values, can be used by researchers to accommodate or correct for imperfections in the network's time synchronization, as well as inform network operators that there might be time synchronization failures.



```
 ---------- column (repNum) 1 in delta data ----------
(~, rx_name ) -----     e_estimate    -----   T_estimate
(~, bes      ) -----  [0.00000000]  ----- [1694.70000000]
(~, browning) -----  [0.45714286]  ----- [780.50000000]
(~, fm       ) -----  [-928.4857143]  ----- [855.1000000]
(~, honors   ) -----  [0.40000000]  ----- [917.90000000]
(~, hospital) -----  [-0.0571429]  ----- [-78.9000000]
(~, smt      ) -----  [-0.0000000]  ----- [168.7000000]
(~, ustar    ) -----  [0.88571429]  ----- [1341.50000000]
```

Fig. 5. Lag results, in number of samples, from WATCH experiment with WR enabled on all nodes.

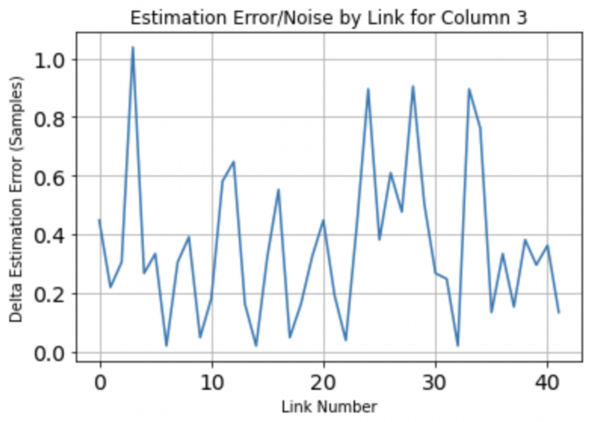


Fig. 6. Plot of direct error, $|\Delta - \hat{\Delta}|$. The x-axis shows the links, numbered to 42 and the y-axis shows the error in number of samples with a 250 kHz sampling rate.

A. Error Analysis

For all WATCH experiments, error was defined as the difference between the offsets from the measured lag data, Δ, and an estimated delta vector, $\hat{\Delta}$, found using (8), (3), and the estimated $\hat{\theta}$.

$$\hat{\Delta} = A\hat{\theta} \tag{8}$$

In-depth error analysis was performed on all experiments and included in WATCH to inform on the precision and reliability of WATCH for each data set as environmental fluctuations may lead to inconsistencies between repetitions. We analyze direct error comparison, $\|\Delta - A\hat{\theta}\|$, and root mean squared error (RMSE) to compare repetitions and estimation results.

Ideally, $\|\Delta - A\hat{\theta}\|$ should be random with no obvious pattern or major peaks. Fig. 6 shows an example resulting error plot with a RMSE of 0.4447 samples or 1.779μs. Observing these plots for each experiment, WATCH can distinguish between data from time-synchronized nodes and data from asynchronous nodes based on their y-axis errors.

The RMSE, in number of samples, is a single number over all samples for each repetition, that shows, on average, how many samples differ between the measured lags, Δ, and the estimations, $\hat{\Delta}$. If the resulting RMSE for any or all of the repetitions in an experiment is larger than approximately 1.0, this informs the researcher that error is high and there might be an issue causing results to differ from the model.

V. Related Work

WATCH contributes to research on time synchronization, experimental testbeds, and debugging of time synchronization challenges. Time synchronization issues in systems can be caused by many different problems, including degrading nodes, clock drift, and uncertainties in message delays [17]. Current corrections and techniques are implemented through either local device synchronization or network clock synchronization, both similarly and differently from WATCH.

Local device synchronization reuses parts of the existing network for time synchronization purposes and does not rely on a predetermined global clock [17]. In Time-Stamp Synchronization (TSS), a method where all nodes run asynchronously with timestamps that are only valid at their node, time gets transferred to other nodes by sending the timestamp as a part of a transmitted message [17]. Contrarily, network clock synchronization requires consistency between a global clock and all nodes [17], such as in the wireless SDR method that uses Universal Software Radio Perpiheral (USRP) generated reference PPS pulses to transmit and synchronize with surrounding USRP SDRs [18]. In this method, each RX node checks the rising edge and maximum amplitude of the received PPS to find the exact timing of the reference USRP [18].

WATCH provides a way for Shout to perform local device synchronization by measuring the offsets between nodes in the network without knowledge of global time. Time synchronization can be monitored by WATCH through iterative transmissions and receptions across all devices, as well as offset analysis upon packet reception. While introducing hardware and network clock synchronization can bring new challenges, such as a question of readily available resources, it can also simplify the required software for running the system [19]. It is, therefore, helpful to observe the advantages and disadvantages inherent to network clock synchronization, as well as to understand the workings of WATCH.

VI. Conclusion

We developed WATCH as an operational GitHub repository that uses our node interaction matrix A to model the measured time offsets observed on links across a distributed network as a function of the clock offsets and TX/RX delays. WATCH provides researchers with suggestions about the accuracy of the external clock time synchronization among their nodes, allowing readjustments to prevent time degradations from negatively effecting their results. WATCH was evaluated through experiments on POWDER, transmitting PN code packets, and analyzing error between measured offsets and estimations, but it was also designed to be applicable to other SDR platforms. Future steps include modifying WATCH to allow adjustments to values such as the number of transmission iterations and the number of nodes, therefore accommodating other platforms, as well as automating the code to consistently monitor networks.

Acknowledgment

The authors would like to thank Jie Wang, Aarti Singh, and Meles Gebreyesus Weldegebriel for introducing the useful tools in POWDER; Dr. Jim Feher and Francesca Allhoff at Washington University, and the members of the Flux Research Group at the University of Utah.

References

[1] E. Hamed, H. Rahul, and B. Partov. “Chorus: Truly Distributed Distributed-MIMO.” In *ACM SIGCOMM 2018 Conference*, Budapest, Hungary, Aug. 2018.

[2] A. Luong, P. Hillyard, A. S. Abrar, C. Che, A. Rowe, T. Schmid, and N. Patwari. “A stitch in time and frequency synchronization saves bandwidth.” *17th ACM/IEEE Intl. Conf. on Information Processing in Sensor Networks (IPSN)*, Apr. 2018.

[3] ETSI. “5G NR: Requirements for support of radio resource management (3GPP TS 38.133 version 16.4.0 Release 16).” *ETSI Standards Technical Specification*, Aug. 2020.

[4] B. Li, X. Zhu, Y. Jiang, H. Zeng, and Y. Wang. “Cooperative time synchronization and parameter estimation via broadcasting for cell-free massive MIMO networks.” *IEEE Wireless Communications and Networking Conference (WCNC)*, 2022.

[5] L. Baldesi, F. Restuccia and T. Melodia, “ChARM: NextG spectrum sharing through data-driven real-time O-RAN dynamic control.” *IEEE Conference on Computer Communications*, London, United Kingdom, 2022, pp. 240-249, doi: 10.1109/INFOCOM48880.2022.9796985.

[6] J. Breen, A. Buffmire, J. Duerig, K. Dutt, A. Ghosh, M. Hibler, D. Johnson, S. K. Kasera, E. Lewis, D. Maas, C. Martin, A. Orange, N. Patwari, D. Reading, R. Ricci, D. Schurig, L. Stoller, A. Todd, K. V. der Merwe, N. Viswanathan, K. Webb, and G. Wong. “POWDER: Platform for open wireless data-driven experimental research.” *Computer Networks*, 197, Oct. 2021.

[7] M. Lipiński, T. W lostowski, J. Serrano, and P. Alvarez. “White rabbit: A PTP application for robust sub-nanosecond synchronization.” In *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pp. 25–30, 2011.

[8] F. Mitchell, A. Baset, and K. Webb. “Shout Measurement Framework.” Flux Gitlab, Source Code, Version 3, December 2021. Online: https://gitlab.flux.utah.edu/frost/proj-radio-meas.

[9] A. Solomon Abrar, A. Luong, G. Spencer, N. Genstein, N. Patwari, and M. Minor. “Collision prediction from UWB range measurements.” arXiv:2010.04313 [eess.SP], 9 Oct. 2020.

[10] K. Webb, S. K. Kasera, N. Patwari, and J. Van der Merwe. “WiMatch: Wireless resource matchmaking.” *IEEE INFOCOM Workshop: Computer and Networking Experimental Research using Testbeds (CNERT 2021)*, May 2021.

[11] C. Jeng, N. Patwari, A. Singh, J. Wang, and M. G. Weldegebriel. “Over-the-air Narrowband QPSK Modulation and Demodulation.” Source Code, Version 0.1, January 2023. Online: https://github.com/npatwari/tx_rx_processing.

[12] Cassie Jeng. “WATCH: A distributed clock time offset estimation tool on the platform for open wireless data-driven experimental research.” 2023. Washington University in St. Louis, Masters Thesis.

[13] C. Jeng. “Offset estimation.” Source Code, Version 0.1, May 2023. Online: https://github.com/cjeng8771/offset_estimation.

[14] R. N. Mutagi. Pseudo noise sequences for engineers. *Electronics & Communication Engineering Journal*, 8(2), April 1996.

[15] N. Patwari. *ECE 5325/6325: Wireless communication systems lecture notes*, January 2017. Department of Electrical and Computer Engineering, University of Utah.

[16] J. Serrano, M. Cattin, E. Gousiou, E. van der Bij, T. W Lostowski, G. Daniluk, and M. Lipiński. “The White Rabbit Project.” *Proceedings of IBIC2013*, Oxford, UK, Sep. 2013.

[17] K. Römer, P. C. C. Blum, and L. Meier. “Time synchronization and calibration in wireless sensor networks.” In *Handbook of Sensor Networks*, 2005.

[18] W. J. Yoo, K. H. Choi, J. Lim, L. W. Kim, H. K. Lee, and H. So. “An experiment study for time synchronization utilizing USRP and GNURadio.” In *GNU Radio Conference 2017*, September 2017.

[19] M. Buevich, N. Rajagopal, and A. Rowe. “Hardware assisted clock synchronization for real-time sensor networks.” In *2013 IEEE 34th Real-Time Systems Symposium*, 2013.