BPS ATSC 3.0 Broadcast Emission Time Stabilization System Proof-of-concept

Mark T. Corl
Triveni Digital, Inc.
Princeton, N.J., United States
mcorl@trivenidigital.com

Vladimir Anishchenko
Avateq Corp.
Markham, Ontario, Canada
vlad_a@avateq.com

Tariq Mondal
National Association of Broadcasters
Washington, D.C., United States
tmondal@nab.org

Abstract – The ATSC 3.0 NextGen TV standard defines a precise time stamp of the emission time of each broadcast frame. The time stamp, along with the transmitting antenna location, can be used to determine the distance of the receiver from the transmitter. If at least three such ATSC 3.0 transmissions are available in an area, these broadcasts can be used as a precise positioning system, called a Broadcast Positioning System (BPS) [1], which can be used as a backup system for the Global Positioning System (GPS).

However, providing a precise time stamp is technically challenging due to various delays and variability in the broadcast transmission studio processing chain. This paper describes a proof-of-concept system, developed by multiple partner companies under the direction of NAB Pilot, that provides a closed loop time stabilization system. The paper explores the techniques used to measure the broadcast signal's emission time and to compensate for the processing chain timing variability stabilizing the emission time stamp. The paper also describes the lessons learned and the barriers to further improve the timing accuracy of the BPS system.

Background

The preamble of an ATSC 3.0 frame carries the bootstrap emission timestamp. If this timestamp is accurate, a receiver can synchronize its clock using the ATSC 3.0 signal. Moreover, since the ATSC 3.0 signal can carry data, the location of the transmitting antenna can be sent with the signal as data. Armed with the precise bootstrap emission time and the location of the transmitting antenna, a receiver at a known location can maintain a very accurate clock.

A receiver can calculate its own location if it receives emission timestamps from three or more ATSC 3.0 transmitting antennas with known locations. A receiver can also improve its past time and location estimates if the past timestamping errors are reported in the data sent over the ATSC 3.0 broadcast.

This paper is excerpted from the Proceedings of the 2023 NAB Broadcast Engineering and Information Technology (BEIT) Conference, © 2023, National Association of Broadcasters, 1 M Street SE, Washington, DC 20003 USA.

Finally, location spoofing resiliency increases if an ATSC 3.0 station listens to the signals from neighboring ATSC 3.0 stations and reports the neighbor signal measurements in the data pipe.

Although BPS supports the multiple use cases mentioned above, the goal of the proof-of-concept was to demonstrate an ATSC 3.0-based timing solution with an accurate clock in a lab setting. However, it was also deemed useful to design and test the data structures and transmission so that the timing solution could easily be extended to other use cases in the future.

Proof-of-concept Prototype Description

The prototype implementation is based on multiple existing products that were enhanced to meet the proof-of-concept objectives and deliverables. These included the Triveni Digital Broadcast Gateway and the Avateq AVQ1022(1st Gen) RF Layer Monitoring Receiver. The use of existing hardware and software platforms allowed the delivery goals of the proof-of-concept to be met. The discussion following the system architecture diagram also describes the products impacted by the function described.

Figure 1 provides a logical diagram of the systems and functions developed. The numbered paragraphs that follow the diagram correspond to the callouts in the diagram.

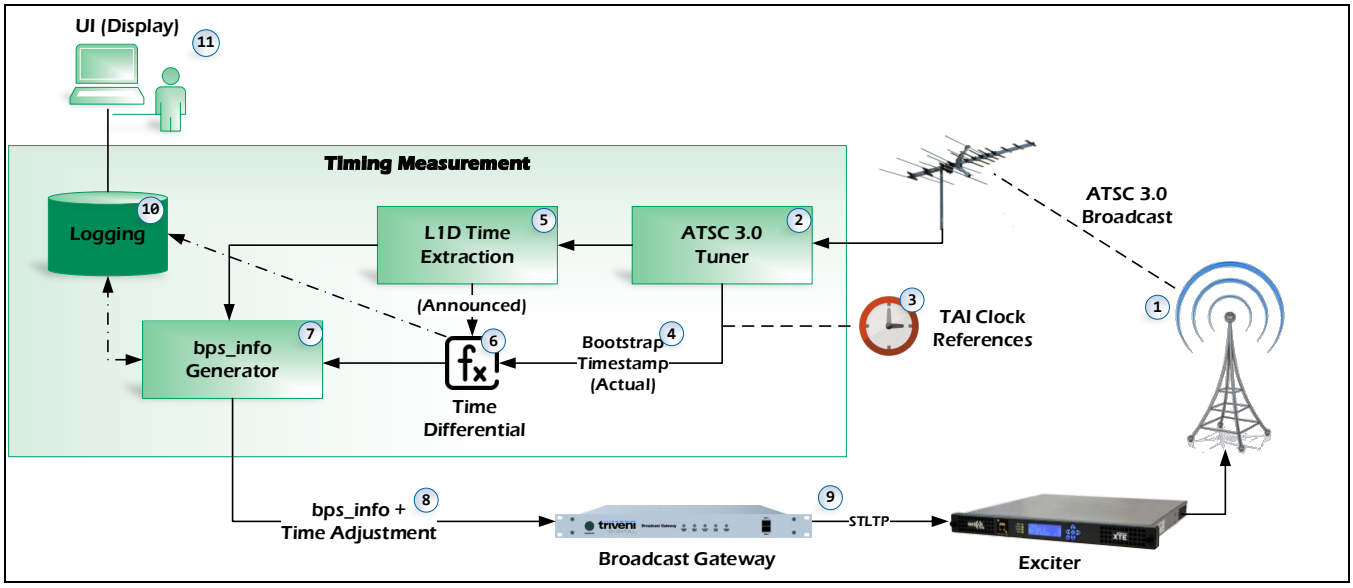

FIGURE 1: BPS PROTOTYPE SYSTEM LOGICAL DIAGRAM

1) **ATSC 3.0 transmission**. Legal transmissions must include a time stamp based on the current TAI time in the detailed preamble (L1D) according to ATSC A/322 [2]. Leap seconds are communicated in other signaling (*i.e.*, LLS). The time represented by the L1D timestamp is referred to as the "Time Information Position," or "TIP," which is the instant in time at the center of the leading edge of the first sample of the first symbol of the bootstrap.
2) **ATSC 3.0 tuner**. For the proof-of-concept, this was the tuner currently deployed in the Avateq AVQ1022(1st Gen) RF Layer Monitoring Receiver which has limited capability relative to the desired measurement accuracy. The exact measurement accuracy was determined empirically as part of the prototype development to be near 200 nsec.
3) **Reference clock**. The TAI clock reference used by the receiver is GPS.

4) **Measured time**. The receiver hardware combined with the clock reference results in the Time Information Position measurement. This represents the actual bootstrap emission time accounting for the propagation delay due to the distance from the transmission tower. The accuracy and precision of this value is an outcome of the current Avateq receiver.
5) **L1D Announced time**. The time reported in the preamble is extracted to provide the announced time in the ATSC 3.0 frame. This is created from the L1D_time_sec, L1D_time_msec, L1D_time_usec, and L1D_time_nsec fields in the preamble.
6) **Time differential**. The difference between the measured time and the announced time is calculated. This value is input to a smoothing function to avoid jitter. The filters and methods used to evaluate the overall loop performance are described in detail later in this paper. The resultant values are sent to the bps_info Generator as well as logged.
7) **`bps_info` data generation**. The bps_info Generator uses the latest time differential information along with preset location values to create a bps_info data structure. This data structure is logged and sent to the Broadcast Gateway as an LLS user-defined table. User-defined LLS tables are described in ATSC A/331 [3].
8) **Time adjustment**. The time adjustment information is sent to the Broadcast Gateway through an extension to the products standard control APIs. The time adjustment can be associated with individual transmitters. The bps_info data structure packet is sent to the Broadcast Gateway through using the current gateway interfaces for ingesting LLS.
9) **Broadcast gateway**. The Broadcast Gateway uses the time adjustment information to adjust the emission time in the next available frame for the transmitter. The bps_info data is inserted into a PLP dedicated to the purpose. Note that due to frame buffering within the ATSC 3.0 pipeline, time adjustments will be delayed by multiple frame times.
10) **Log**. The Logging function collects information from the various generation functions and makes it available to the user through the UI interface. The data supplied is also exportable as CSV to allow the data to be ingested into Excel for further evaluation.
11) **User interface**. A user interface (UI) is provided via a web interface (HTTP) to allow logging information to be displayed as well as to allow the system to be controlled. Control connections are not shown to reduce diagram complexity. Of particular interest will be a user interface that provides a graph of the time differential value over time to verify that the system is operating correctly.

Implementation Details

For the proof-of-concept implementation, the Avateq AVQ1022(1st Gen) RF Layer Monitoring Receiver is used to realize callouts 2) through 6). A laptop computer was provided to collect long-term logs and other information as well as form the bridge between the receiver and the broadcast gateway. The software developed for the prototype running on the laptop provides functions 7), 10), and 11) shown in Figure 1 above. The receiver also provides a user interface which shows details of the reception process. This division allowed more development flexibility between Triveni Digital and Avateq accelerating the availability of the functions for testing.

The resulting prototype implementation is diagramed in Figure 2 below. Descriptions of each callout follow the figure.

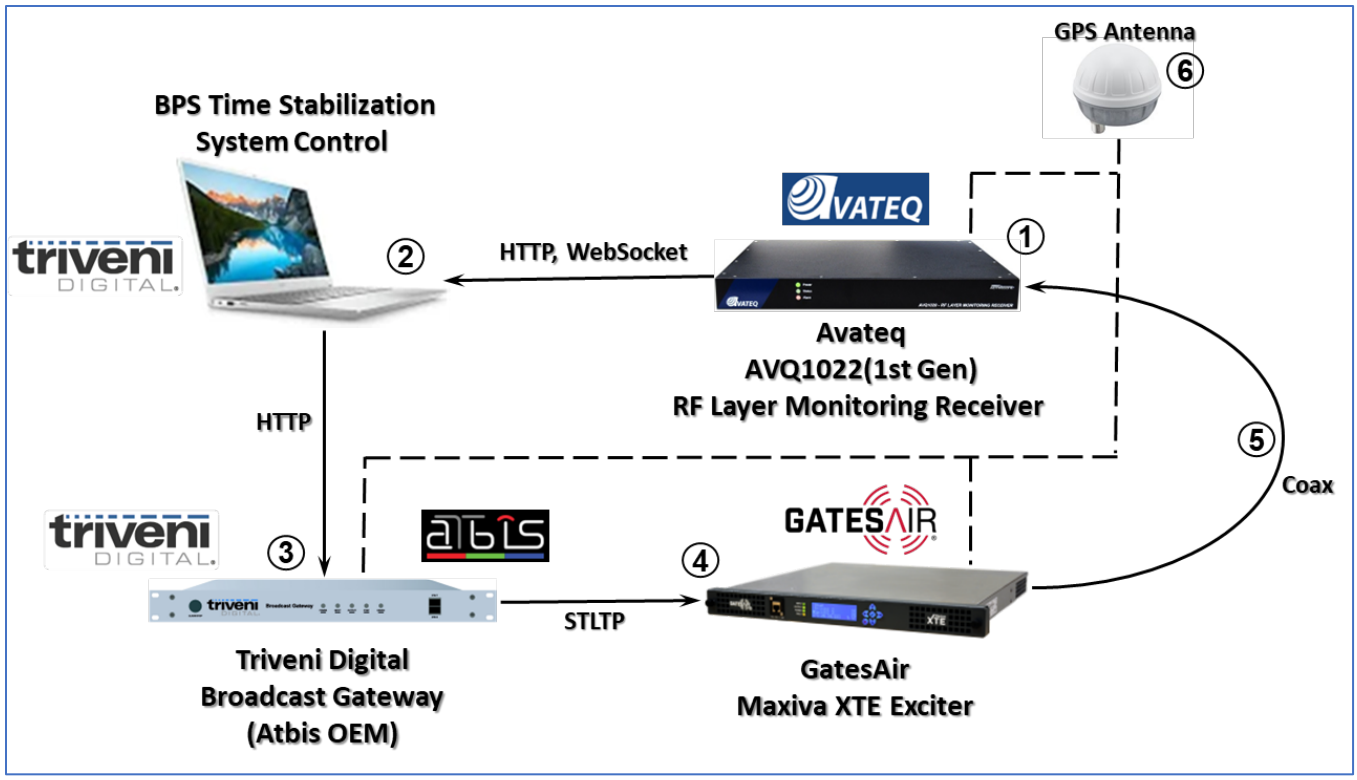

FIGURE 2: PROTOTYPE BPS TIMING LOOP IMPLEMENTATION.

1) Avateq AVQ1022(1st Gen) RF Layer Monitoring Receiver, modified to measure the arrival time of each frame and compare it with the L1D announced time.
2) BPS Time Stabilization system control laptop, providing a convenient user interface to control the various devices in the loop as well as view resulting data. All configuration and control are performed from the user interfaces displayed on the laptop. To ease configuration and avoid conflicts, communication between the components of the loop has been separated into physically disparate networks.
3) Broadcast Gateway, accepting control information from the system control laptop based on measurement data from the receiver and adjusting the emission time of the exciter.
4) Control and baseband packet data are sent to the exciter using scheduler/studio to transmitter link tunneling protocol (STLTP). The exciter in this prototype loop was not modified and is behaving as nominally designed.
5) The prototype system routes the RF output of the exciter directly to the receiver input.
6) GPS is used by the prototype systems to synchronize clocks and provide reference signals.

Receiver Implementation Details

The Avateq AVQ1022(1st Gen) RF Layer Monitoring Receiver is a central component of the proof-of-concept stabilization loop so details of its implementation and filtering capabilities are provided. Figure 3 provides a functional block diagram of the receiver internal subsystems.

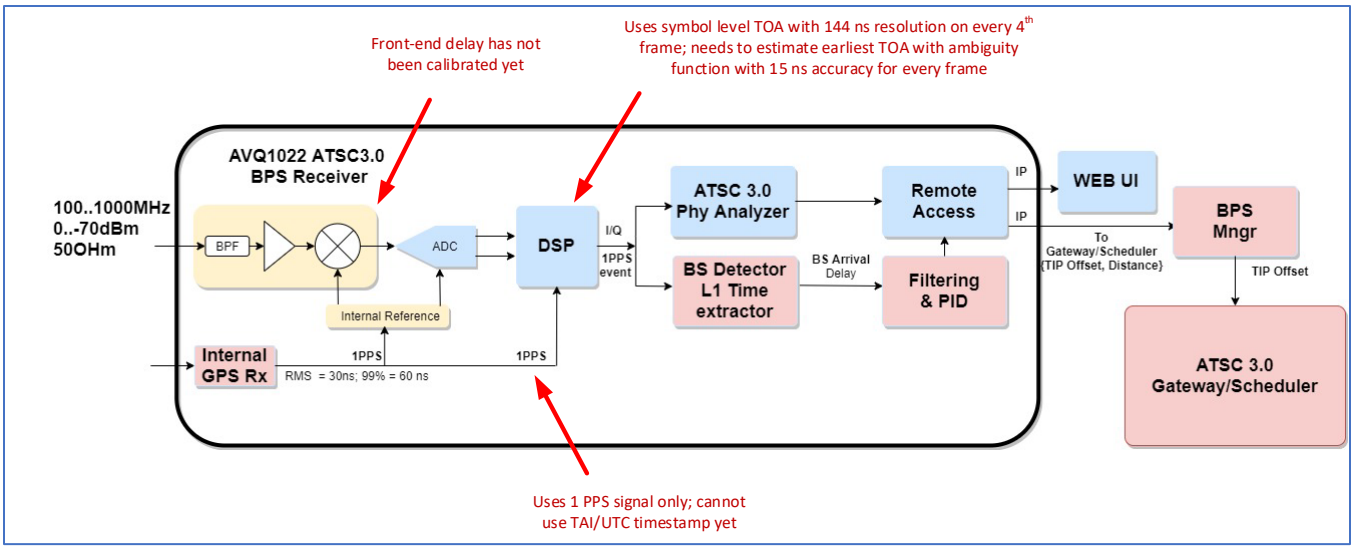

FIGURE 3: BPS RECEIVER LOGICAL DIAGRAM

The AVQ1022 receiver architecture utilizes the software defined radio (SDR) concept with a frequency agile analog front-end, signal digitizing circuitry and digital signal processor (DSP) core implemented in a field-programmable gate array (FPGA). The DSP core outputs I/Q samples for further processing by software based functional blocks. The I/Q samples are marked with a 1PPS event. All frequency converting, sampling and digital processing stages are synchronized with an external or receiver-generated internal GPS module's 1PPS signal.

The software based ATSC 3.0 Phy Analyzer block provides a comprehensive set of measurements and troubleshooting tools to monitor RF signal quality and transmitter system performance.

The Bootstrap Detector block provides a sequence of the Bootstrap arrival delays calculated as a time difference between L1D_time_xxx and the time of receiving the first Bootstrap sample relative to the GPS 1PPS pulse.

The Bootstrap arrival delays are additionally filtered with a moving average FIR, with a user-defined filter length. The filter length is carefully adjusted as it affects both the measurement accuracy and the control loop convergence speed (settling time).

The filter output directly feeds a proportional-integral (PI) controller. The PI Controller is designed with user adjustable proportional and integral path gain coefficients: Kp and Ki.

The resulting PI controller output in a form of TIP adjustment is available through the receiver remote interface and can be used by a BPS controller to adjust the TIP offset in ATSC 3.0 exciter.

An additional parameter, TIP Reference, is introduced to set the TIP Adjustment value as an initial "zero" position, *i.e.*, a position with zero distance from a transmitting antenna and zero Bootstrap arrival delay.

The newly implemented BPS-related functionality provides the following:

- Detecting, estimating, and tracking any difference between the reported L1D_time_xxx and the calculated Bootstrap arrival time.

- Calibrating transmitter-side equipment to account for possible DSP and analog processing delays at a transmitter site and precisely adjusting the L1D_time_xxx time to the instant when the first sample of the first symbol of Bootstrap is transmitted.
- Verification of transmitter equipment synchronization and stability by estimating possible sample rate shifts causing the Bootstrap arrival delay drift.

Proof-of-concept Prototype Performance

The prototype BPS solution developed by the partner companies demonstrates that a feedback control loop can increase the accuracy of the transmitted timestamps of an ATSC 3.0 transmission chain. We ran two sets of prototype transmission systems with different exciters, and in both cases the instantaneous time delay of the stabilized system was less than 300 nsec. We learned that the different exciters had different time response characteristics, meaning they have different transfer functions. We found that the optimal PI controller settings were different in each case. Our prototypes were stable for many days.

We have also demonstrated that the proposed time stabilization solution can be easily integrated into an existing transmission chain. Most of the elements of the feedback control loop are add-ons, meaning they do not require modifying the existing transmission chain. One impact to the transmission chain is that new gateway software is required that can receive the PI controller output and make necessary adjustments to the timestamps. Note that no changes of any kind were required for either exciter controlled by the prototype systems. The controls provided by the existing broadcast gateway were sufficient.

The solution we have demonstrated is 100% compliant with the ATSC 3.0 standard. No changes to the standard were necessary. We have also demonstrated the solution using transmission equipment from multiple ATSC 3.0 vendors, reinforcing standard compliance.

Finally, we have identified opportunities to make the system more accurate as discussed in a subsequent section below.

Receiver Prototype Performance

The yellow and blue curves exemplified in Figure 4 correspond to raw measurements of the Bootstrap arrival delay and the filtered and PID-processed TIP adjustment, respectively. The present implementation of the receiver hardware limits the resolution of the time of arrival delay to approximately 144 nsec. However, the sophisticated processing of the raw measurements implemented in the receiver helps constrain the TIP adjustment variations to within approximately ±70 nsec.

The measurement resolution for the prototype is limited by the sampling rate of the baseband signal. The receiver uses the first sample of the first symbol of the bootstrap to calculate the time of arrival. We observed that the time of arrival, denoted by the yellow line, can jump by ±2 samples, or ±288 nsec. Although the time of arrival offset measurements were rather jittery, the averaged values were much smoother, meaning that an external receiver synchronizing its clock to the filtered and/or smoothed time values should achieve much better accuracy.

It is worth mentioning that both the exciter and the time of arrival measurement receiver introduced errors which were found to be ±2 samples, but due the low resolution and accuracy we could not characterize the errors contributed by each individual piece of equipment.

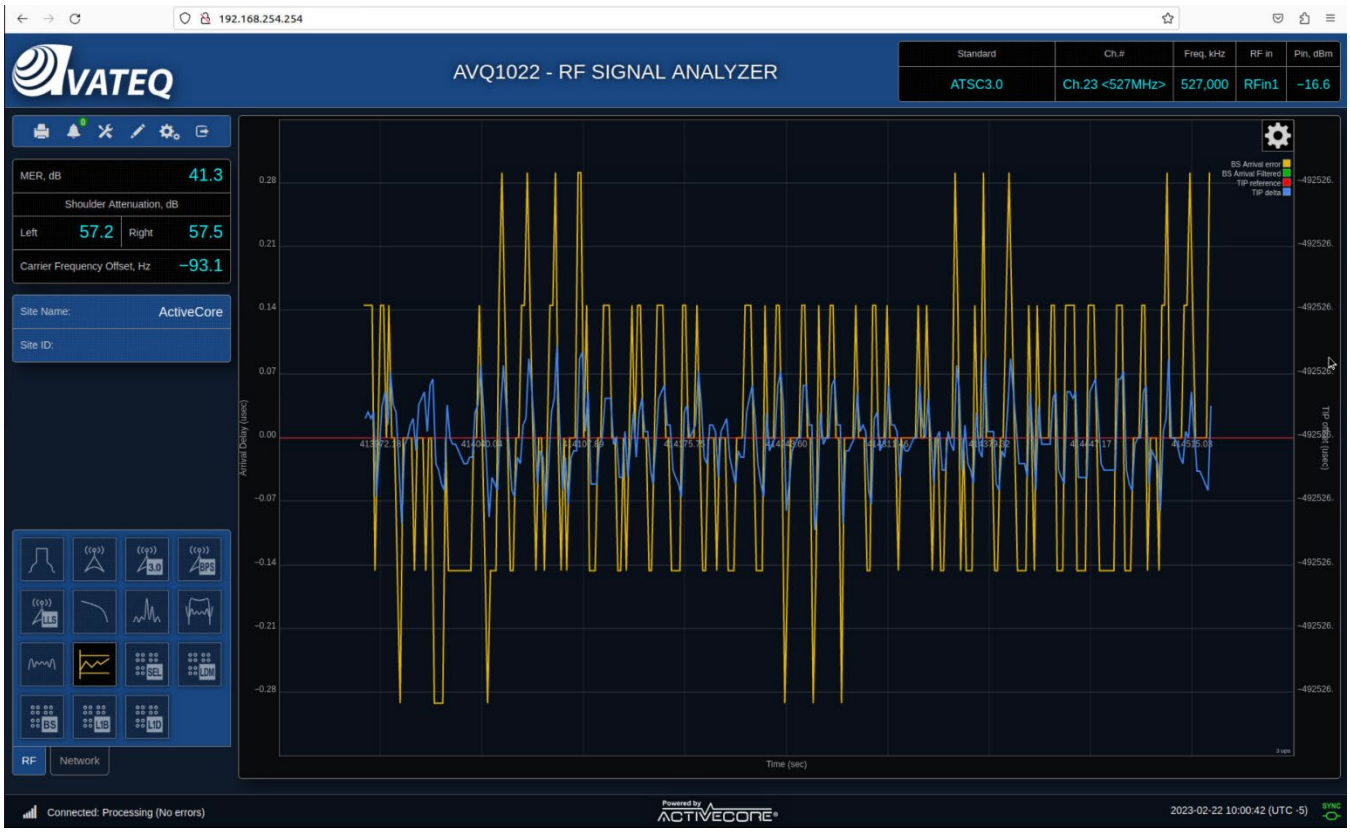

FIGURE 4: EXAMPLE RECEIVER BPS TIME ARRIVAL GRAPH. SEE TEXT.

System Control Prototype Performance

The purpose of the system control prototype was to obtain measurement data from the receiver, log the data to provide for historical measurement and offline analysis, create the bps_info data structure and interface with the broadcast gateway. The software developed was flexible, anticipating that the proof-of-concept prototype might require changes as the project evolved. This proved to be true, and the resultant user interface is shown in Figure 5.

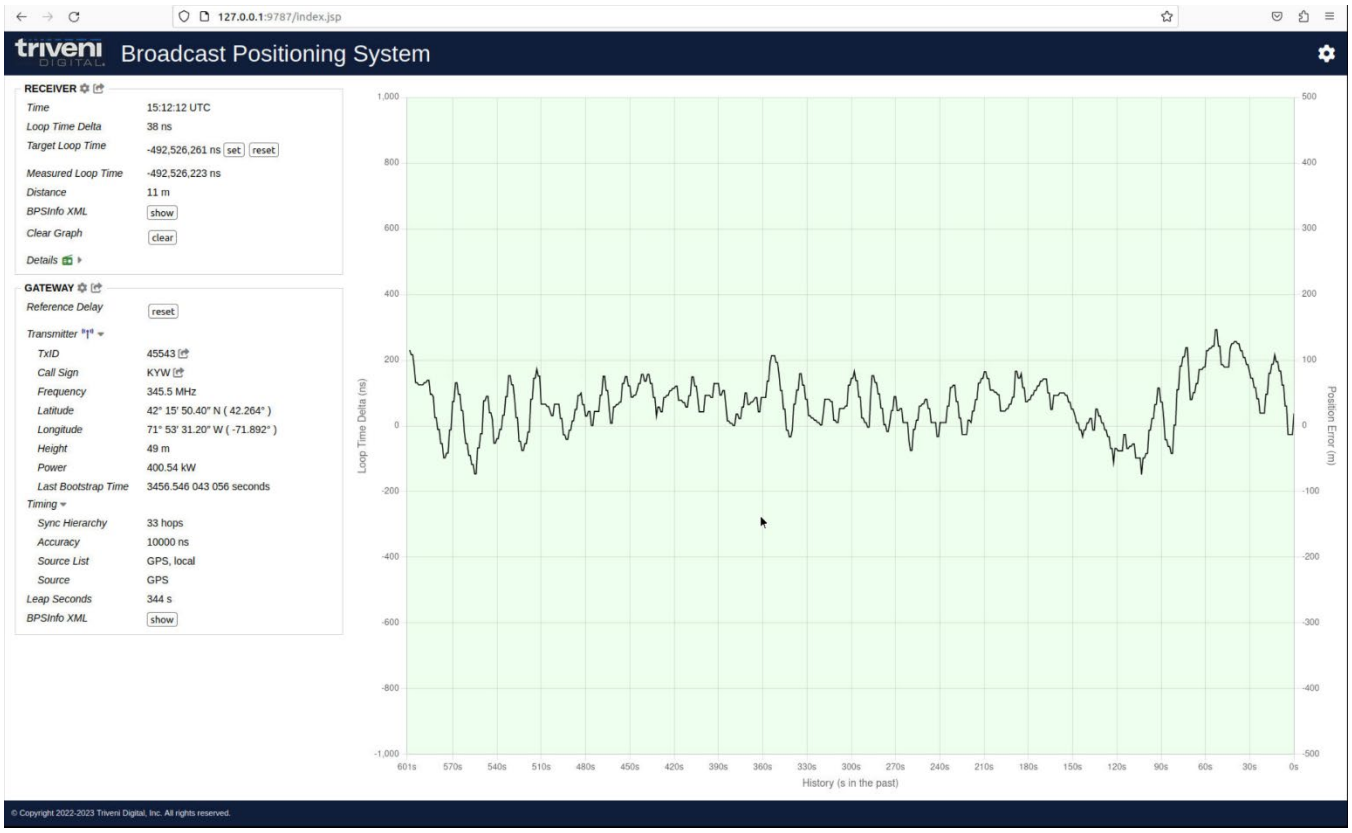

FIGURE 5: EXAMPLE BPS CONTROL SYSTEM USER INTERFACE.

Figure 5 shows a snapshot of the control system during operation. In the left panel, there is data which was either configured through this user interface or obtained from the receiver. The top portion of the panel shows receiver information with buttons to set the “Target Loop Time” shown as the “TIP reference” red line in Figure 4 above, reset that time to zero, and clear the graph. There are also links to launch the receiver user interface and configure the connection to the receiver. Finally, there is a button to view the bps_info data structure read from the receiver after it has passed through the system.

The Gateway portion of the left panel shows information regarding the transmitter, its location, call sign, frequency and other information that is included in the bps_info data structure to identify the transmitter. There is also a button in this panel to access the bps-info structure that will be sent to the broadcast gateway. An example of the dialog shown when this button is clicked is provided in Figure 6 below.

Finally, the graph that makes up much of the control system user interface is a longer time scale version of the blue graph shown in the receiver Figure 4. Note that the snapshots shown in Figures 4 and 5 were not synchronized in time so the illustrative figures cannot be directly compared.

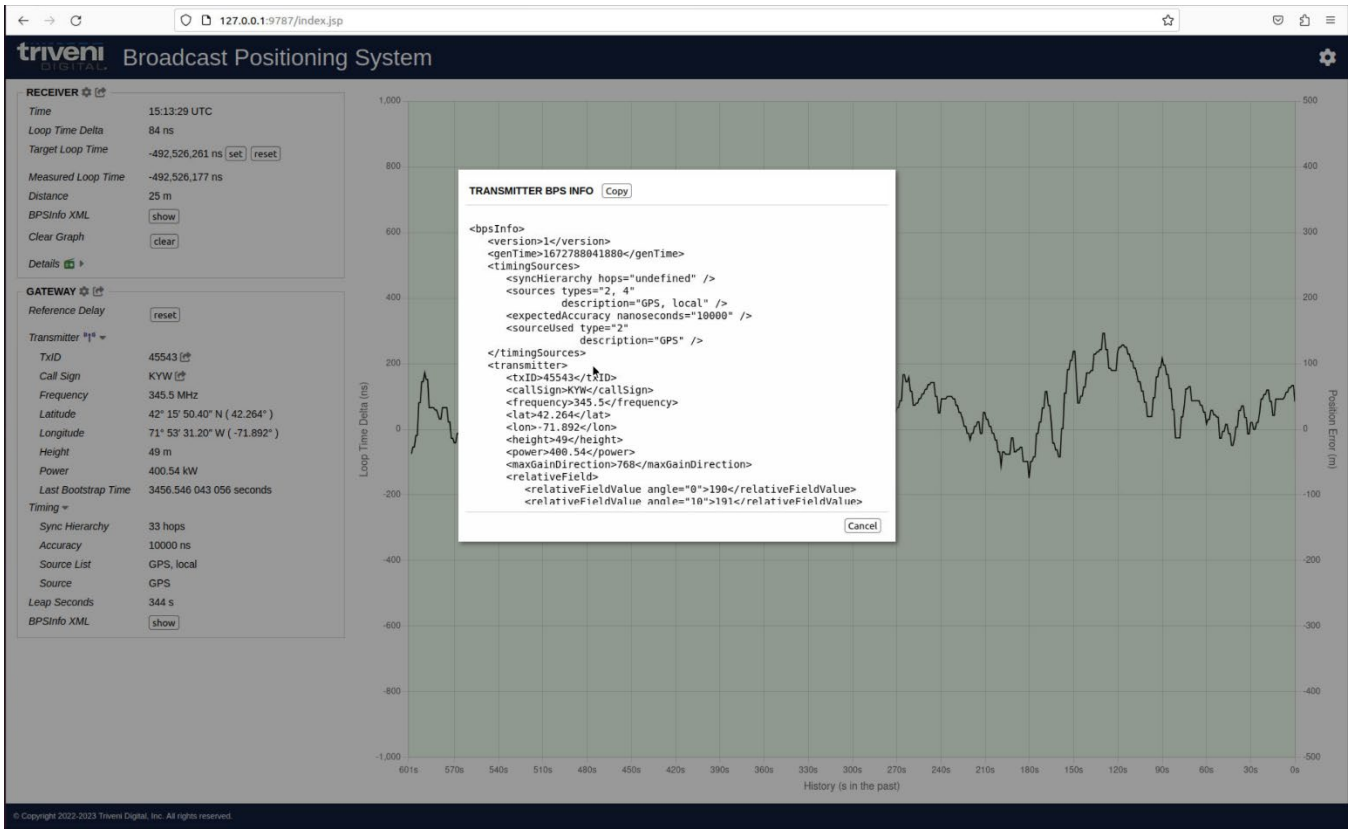

FIGURE 6: EXAMPLE CONTROL SYSTEM BPS_INFO DIALOG.

The bps_info structure, as the dialog shown in Figure 6 exemplifies, was provided to demonstrate the ability to send the data through the control loop. Though not used directly by this prototype, ultimately, the bps_info will be used by other receivers to determine the location of the transmission and be able to determine their distance from that transmitter. In the current implementation, the bps_info data structure was encoded as XML and sent as a user-defined LLS packet [3]. This allows unmodified ATSC 3.0 analyzers to view the data. See Annex A for a detailed description of the bps_info semantics.

Future Work and Improvement Suggestions

In developing and demonstrating the time stabilization algorithm with existing ATSC 3.0-compliant transmission chains, we identified a number of ways to improve accuracy and efficiency. Our suggestions follow:

1) Develop improved capabilities to timestamp the baseband samples in the receiver with TAI or UTC.
2) The receiver front-end delay needs to be accurately characterized so that we can determine, from the baseband sample timestamps, when the RF signal impinged on the receiver antenna.
3) The time of arrival measurements need to have better resolution and accuracy. One suggestion is to reconstruct the bootstrap waveform in the receiver, correlate with the received signal using an ambiguity function [4] and interpolate between the detection peaks. Using a higher sampling rate will help. Consider doubling the sampling rate for time of arrival measurements.
4) Measure the timestamping errors of every transmitted frame and apply the corrections to every transmitted frame. In other words, add more processing power.

5) The controller output displays some periodic oscillation which is most likely due to the delay in applying the corrections. Reduce the delay in transmission chain with the goal of applying the latest timing corrections before the next frame is transmitted.
6) Once the timing measurement becomes more accurate with higher resolution, experiment with a proportional–integral–derivative controller (PID) in the loop. The derivative part will predict the timing deviations and thus may help with the internal framing delay issues.
7) With more accurate time of arrival measurements, characterize the timing jitter caused by the exciters. If the exciters are too jittery, work with exciter manufacturers and figure out ways to mitigate the timing variability.
8) Populate the API fields for past measurements so that an external receiver can apply those corrections and can achieve a more accurate time reference for past transmissions. This will help with synchronizing the local clock of an external receiver.
9) Consider incorporating the bps_info() message in the ATSC 3.0 standard and making it part of the preamble.

Conclusion

A proof-of-concept prototype was built that demonstrates time stabilization of a BPS-enabled ATSC 3.0 transmission facility. We used commercial-grade exciters and other transmission chain equipment to prove compliance with the ATSC 3.0 standard. The initial design of the prototype achieved reasonable accuracy and stability but with some timing jitter. We have identified the causes that affected the timing performance and have suggested a number of ideas to improve performance.

References

[1] Mondal, T., *et a*l., “Broadcast Positioning System (BPS) Using ATSC 3.0,” Proceedings of the 2021 NAB Broadcast Engineering and Information Technology (BEIT) Conference.

[2] ATSC, “ATSC Standard: Physical Layer Protocol,” Doc. A/322:2022-11, Advanced Television Systems Committee, Washington, DC, 14 November 2022.

[3] ATSC, “ATSC Standard: Signaling, Delivery, Synchronization, and Error Protection,” Doc. A/331:2022-11, Advanced Television Systems Committee, Washington, DC, 30 November 2022.

[4] Stein, S., “Algorithms for Ambiguity Function Processing,” IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 29, Issue 3, Jun 1981, pp. 588-599.

Appendix A

A mandatory presence in the following table means that if the `bps_info` message is transmitted, those fields must be populated within the message.

Syntax	*No. of bits*	*Format*	*Presence*
bps_info(){			
message_length	16	unsigned integer	mandatory
version	8	unsigned integer	
timing_source_info(){			
sync_hierarchy	7	unsigned integer	
num_independent_sources	6	unsigned integer	
for (i=0;i< num_independent_sources;i++){			
source_type_list	4	unsigned integer	
}			
expected_accuracy	16	unsigned integer	
source_used	4	unsigned integer	
}			
self_measurement_info(){			
call_sign	42	array of 7 6-bit unsigned integers	
tx_id	13	unsigned integer	
tx_freq	32	32-bit floating point	
geodetic_lat	64	64-bit double precision	mandatory
geodetic_lon	64	64-bit double precision	mandatory
geodetic_height	64	64-bit double precision	mandatory
radiated_power	32	32-bit floating point	
for (i=0;i<36;i++){			
antenna_pattern_relative_field	252	array of 36 7-bit unsigned integers	
}			
max_gain_direction	10	unsigned integer	
prev_bootstrap_time_sec	32	unsigned integer	
prev_bootstrap_time_msec	10	unsigned integer	
prev_bootstrap_time_usec	10	unsigned integer	
prev_bootstrap_time_nsec	10	unsigned integer	
prev_bootstrap_time_error_nsec	16	signed integer	
}			
leap_seconds	8	unsigned integer	mandatory
num_neighbors	6	unsigned integer	
for (i=0;i<num_neighbors; i++){			

neighbor_measurement_info(){			
call_sign	42	array of 7 6-bit unsigned integers	
tx_id	13	unsigned integer	
tx_freq	32	32-bit floating point	
geodetic_lat	64	64-bit double precision	
geodetic_lon	64	64-bit double precision	
geodetic_height	64	64-bit double precision	
radiated_power	32	32-bit floating point	
for (i=0;i<36;i++){			
antenna_pattern_relative_field	252	array of 36 7-bit unsigned integers	
}			
max_gain_direction	10	unsigned integer	
reported_bootstrap_time_sec	32	unsigned integer	
reported_bootstrap_time_msec	10	unsigned integer	
reported_bootstrap_time_usec	10	unsigned integer	
reported_bootstrap_time_nsec	10	unsigned integer	
bootstrap_toa_offset	32	signed integer	
prev_bootstrap_time_sec	32	unsigned integer	
prev_bootstrap_time_msec	10	unsigned integer	
prev_bootstrap_time_usec	10	unsigned integer	
prev_bootstrap_time_nsec	10	unsigned integer	
prev_bootstrap_time_error_nsec	16	signed integer	
}			
}			
reserved_bits	as needed		mandatory
bps_crc	32	unsigned integer	mandatory
}			

`message_length` – This field indicates the length of the message, including the CRC bits, in units of bytes.

`version` – This field represents the version of the message.

`sync_hierarchy` – This field indicates the number of hops needed to transfer time from a reference ATSC 3.0 tower to the tower that is using neighboring towers' ATSC 3.0 signal as timing reference. If an ATSC 3.0 tower does not use the timing information from any neighboring tower to synchronize its own clock, its hierarchy is 0. For example, towers that receive traceable time from NIST or USNO using satellite, fiber, or other methods but do not use any neighboring tower's signal for synchronization will report 0 in this field. If a tower is using a `sync_hierarchy` 0 tower's signal for timing reference, that tower will report its `sync_hierarchy` as 1. Generally, if a tower is listening to the signals from neighboring towers for clock synchronization, and if n is the lowest `sync_hierarchy` among the received signals, that tower will report n+1 as its own `sync_hierarchy`.

num_independent_sources – This field indicates the number of independent timing sources used at the ensemble clock deployed at the tower location for clock synchronization. For example, if a TV station uses GPS, one cesium clock, eLORAN, and 5 neighboring towers as input to the ensemble clock, that tower will report the number 8 in this field. If a tower does not use an ensemble clock but uses another backup clock if its primary clock fails, the reported value will be 1. The corresponding `source_type_list` parameter will be updated accordingly in that case.

source_type_list – This field indicates the various timing sources used as reference at the ATSC 3.0 transmission facility. The following table is used to indicate the types:

Value	*Source Type*
0	UTC from NIST or USNO securely delivered to the TV station
1	GPS
2	ATSC 3.0 signal emitted from neighboring towers
3	Local clock, such as cesium or rubidium
4	eLORAN
5	Reserved
6	Reserved
7	Reserved
…	…
15	Reserved

If the transmission facility uses an ensemble of clocks, it will list the most accurate timing source of the ensemble as its source.

expected_accuracy – This field indicates the accuracy of the TV station clock 99% of the time compared to UTC. The unit is nanoseconds. A value of 200 means that the TV station clock is synchronized in such a way that the local clock is within 200 ns of UTC 99% of the time.

source_used – This field indicates the type of timing source used at the transmission facility. If the tower clock is derived from an ensemble of timing devices, a value of 0 will be reported. If the tower uses just one of the clocks available to it, that source will be reported using the table described under `source_type_list` parameter.

call_sign – This field states the 3 to 7-letter call sign of the TV station. If the call letters are shorter than 7 characters, the remaining fields are padded with “space” characters. The mapping of 6-bit unsigned integers and the letters in the call sign follows:

Binary	*Character*
000000	space
000001	hyphen
000010	A
000011	B
…	…
011011	Z
011100	0
011101	1
…	…
100101	9

100110	reserved
…	reserved
111111	reserved

tx_id – This field represents the `txid_address` field as it is defined in ATSC A/322 with specific codes assigned according to the table at txid.nabpilot.org.

tx_freq – This field indicates the middle of the assigned television channel frequency in units of MHz.

geodetic_lat – This field represents geodetic latitude of the midpoint of the transmit antenna in WGS 84 LLA convention.

geodetic_lon – This field represents geodetic longitude of the midpoint of the transmit antenna in WGS 84 LLA convention.

geodetic_height – This field represents geodetic height of the midpoint of the transmit antenna in WGS 84 LLA convention.

radiated_power – This field represents ERP of the radiating antenna in units of kilowatts.

antenna_pattern_relative_field – This field represents the antenna pattern by the relative field values reported every 10 degrees. The values are arranged clockwise starting at true north, meaning that the 1st value of the 36-element array represents the relative field strength in the direction north, the 2nd value represents the relative field strength in the direction 10 degrees clockwise from north, and so on. The relative field values are scaled such that the maximum value, which is 1.0, is scaled to the integer 127 in fixed point representation.

max_gain_direction – This field represents the direction where relative field strength is maximum, i.e., 1.0. The angle is measured clockwise from north. The angle is reported as a fixed-point value such that 360 degrees is scaled to 1023 in fixed point notation.

prev_bootstrap_time_sec – This field will be populated with the `L1D_time_sec` value, as defined in ATSC A/322, of the immediately previous transmitted frame. For `self_measurement_info`, the transmitting tower will provide this information. For `neighbor_measurement_info`, the transmitting tower will receive the value in `bps_info` message from a neighboring tower and report that value in this field.

prev_bootstrap_time_msec – This field will be populated with the `L1D_time_msec` value, as defined in ATSC A/322, of the immediately previous transmitted frame. For `self_measurement_info`, the transmitting tower will provide this information. For `neighbor_measurement_info`, the transmitting tower will receive the value in `bps_info` message from a neighboring tower and report that value in this field.

prev_bootstrap_time_usec – This field will be populated with the `L1D_time_usec` value, as defined in ATSC A/322, of the immediately previous transmitted frame. For `self_measurement_info`, the transmitting tower will provide this information. For `neighbor_measurement_info`, the transmitting tower will receive the value in `bps_info` message from a neighboring tower and report that value in this field.

prev_bootstrap_time_nsec – This field will be populated with the L1D_time_nsec value, as defined in ATSC A/322, of the immediately previous transmitted frame. For self_measurement_info, the transmitting tower will provide this information. For neighbor_measurement_info, the transmitting tower will receive the value in bps_info message from a neighboring tower and report that value in this field.

prev_bootstrap_time_error_nsec – This is the difference between the actual time when the first sample of the first symbol of the bootstrap was transmitted and the reported bootstrap transmission time in the L1D_time_sec, L1D_time_msec, L1D_time_usec, and L1D_time_nsec fields mentioned in ATSC A/322. The time difference is measured as actual transmission time minus reported transmission time in nanosecond unit. For self_measurement_info, the transmitting tower will measure and provide this information. For neighbor_measurement_info, the transmitting tower will receive the value in bps_info message from a neighboring tower and report that value in this field.

leap_seconds – This field represents the current number of leap seconds expressed as TAI – UTC.

num_neighbors – This field indicates the number of neighboring signal measurements reported in the message. If this field is set to 0, no neighbor_measurement_info will be populated in the message.

reported_bootstrap_time_sec – This field will be populated with the L1D_time_sec value, as defined in ATSC A/322, of the most recent frame transmitted by a neighbor tower and received at the transmitting tower.

reported_bootstrap_time_msec – This field will be populated with the L1D_time_msec value, as defined in ATSC A/322, of the most recent frame transmitted by a neighbor tower and received at the transmitting tower.

reported_bootstrap_time_usec – This field will be populated with the L1D_time_usec value, as defined in ATSC A/322, of the most recent frame transmitted by a neighbor tower and received at the transmitting tower.

reported_bootstrap_time_nsec – This field will be populated with the L1D_time_nsec value, as defined in ATSC A/322, of the most recent frame transmitted by a neighbor tower and received at the transmitting tower.

bootstrap_toa_offset – This field represents the difference between the time of arrival at the transmitting (self) antenna of the neighboring bootstrap signal and the reported timestamp in L1D_time_sec, L1D_time_msec, L1D_time_usec, and L1D_time_nsec fields as mentioned in ATSC A/322. The unit of this time offset is nanoseconds.

reserved_bits – This field indicates the number of reserved bits, 0 to 7, that is required to byte-align the message.

bps_crc – This field will contain the CRC value as computed according to Section 6.1.2.2 of A/322 over the contents of bps_info message excluding the bps_crc field.