PAC World 2022

Topic #8: Synchrophasors, applications and benefits

Performance and Applications of Synchronised Waveform Data Compression

Steven Blair, Jason Costello
Synaptec, UK

steven.blair@synapt.ec

Abstract

Fundamental changes in power grids due to decarbonization require advanced monitoring and automated analysis. Capturing synchronized waveform data from voltage and current sensors, sometimes referred to Continuous Point on Wave (CPOW) monitoring, offers several capabilities beyond synchrophasors from Phasor Measurement Units (PMUs).

However, the obvious drawbacks in manipulating, transferring, and storing waveform are the high data bandwidth and storage requirements. Therefore, access to streaming synchronized waveform data is typically restricted to substation local area networks (LANs). This paper reports on a platform to address these issues and therefore to deliver wide-area waveform monitoring in a way which is convenient and practical. It is shown how a lossless data compression method designed for streaming waveform data can significantly reduce data bandwidth requirements and improve end-to-end efficiency and latency. Data bandwidth requirements can be reduced to 5-15% of the original size.

The same approach can be applied to both real-time streaming and offline data storage, with reduced file size compared to other industry formats such as COMTRADE and PQDIF. It supports any sampling rate, any number of samples per message, and arbitrary configurations of measurement quantities to be sent. An implementation of the scheme, called Slipstream, has been open sourced to enable industry adoption.

1 Introduction

Monitoring wide-area transient phenomena is increasingly of importance for ensuring efficient grid operation and stability. There is growing evidence that synchronized waveform-based monitoring, which is sometimes referred to as Continuous Point on Wave (CPOW) monitoring, will be increasingly important for power system monitoring, protection, and control [1]–[4]. For example, in the Great Britain grid, oscillations contributed to a major outage in August 2019 which resulted in disconnecting supplies of 1 in 10 customers to secure the wider transmission system [5], yet synchrophasor monitoring systems, by design, cannot capture the full frequency range of all possible oscillation modes. Solar photovoltaic (PV) integration has caused widespread interharmonics to be observed, which could not be properly investigated, even with wide-area synchrophasor measurements, due to frequency aliasing [6]. Local power quality metrics such as harmonics and other transient events can also be computed from waveform data, which is not possible from synchrophasors. New techniques in fault classification and fault location require waveform data to perform pattern matching, with improved classification results at higher waveform sampling frequencies [7], [8]. In summary, synchrophasor data do not provide the detailed waveform, harmonic, and frequency dynamic range required to fully detect and analyse phenomena and events in power systems.

Figure 1 illustrates the value of waveform data compared to PMU data sources. The plots represent positive sequence voltage magnitude from a PMU (upper plot) and the underlying waveform data (lower plot) for the same simulated event. Clearly waveform data provides richer information about the system, including harmonics and fast-acting transients.

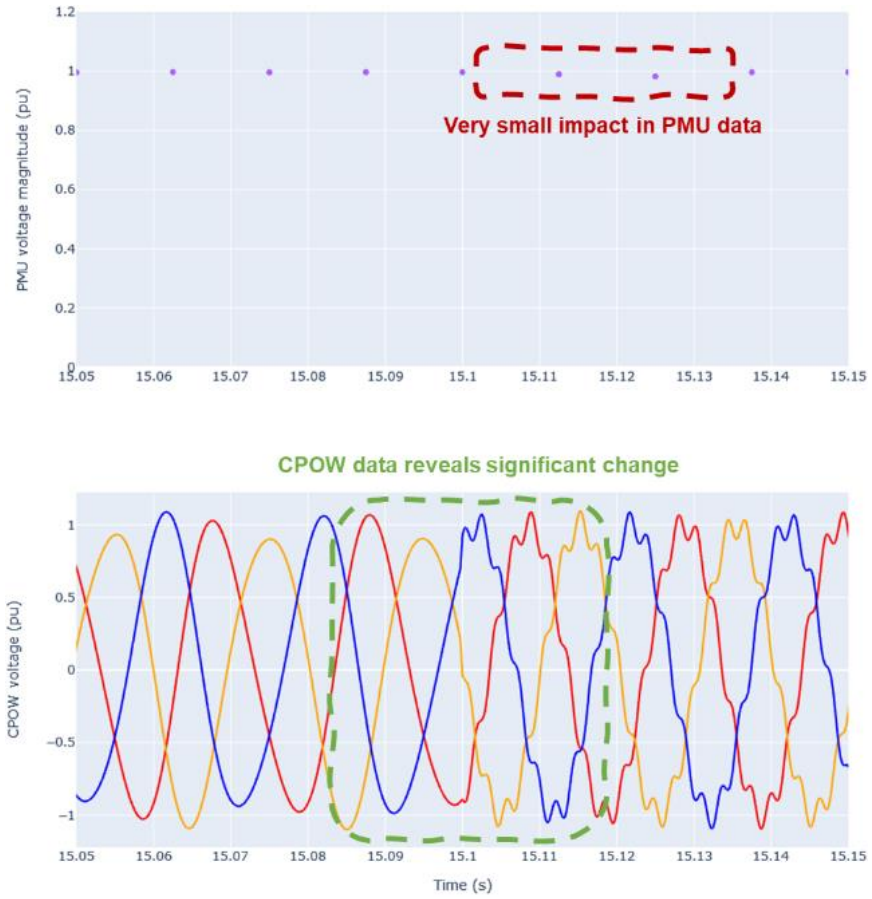

Figure 1: Comparison of PMU and waveform (CPOW) data

There is therefore a need to provide greater wide-area visibility which includes the full waveform data. However, the quantity of raw data which is generated and potentially transferred over a wide-area network (WAN) is much greater than typical synchrophasor or SCADA data streams. In particular, very high sampling rate applications such as travelling wave protection demand improved encoding methods to reduce data transfer [9].

This paper presents a new waveform data compression approach, called Slipstream, which significantly builds on the contributions in [10]. The method provides very high compression performance yet is lossless and therefore allows all data to be fully reconstructed by the receiver. Time synchronization of the samples and data quality information are strictly preserved. The performance of encoding and decoding is also very efficient, such that it is much faster to compress the data than to manipulate the raw uncompressed data. The end-to-end latency for transmitting waveform streams over a local-area network (LAN) or WAN is also reduced. Slipstream is open source and available at [11].

2 Background

2.1 Existing Standards for Waveform Data Transfer

2.1.1 Real-time Data Streaming

The main existing standard for streaming synchronized waveform data is the IEC 61850-9-2 Sampled Value (SV) Ethernet-based protocol, with some additional conventions for merging units defined in IEC 61869-9. IEC 61850-90-5 extends SV for transfer over an IP-based WAN.

The work in [10] achieved moderate compression performance while broadly minimizing the changes to the IEC 61850-9-2 protocol format. However, the new approach in this paper is a radial redesign which is not encumbered by the conventions of IEC 61850-9-2, and is optimized for efficiency and throughput. The Streaming Telemetry Transport Protocol (STTP) is presently under development as IEEE standard P2664 [12]. Open source reference implements are available at [13]. STTP is intended to supersede IEEE C37.118.2 as the future protocol for transferring synchrophasor data, as well as being suitable for

wider applications. STTP supports sending waveform timeseries data, encryption using TLS, and includes an optional lossless compression feature.

2.1.2 File-based Waveform Storage

The PQDIF format (IEEE Std. 1159.3) is designed as a self-contained description of a power quality, fault, or transient event. A PQDIF record typically contains a relatively short burst of waveform data and derived timeseries quantities. It includes a zlib-based compression feature. The COMTRADE format offers a similar capability to PQDIF, but does not group multiple “observation” records together in a single file or provide integrated data compression. Power quality meters typically also provide data in an arbitrary comma-separated values (CSV) format for simple decoding.

2.2 Existing Methods for Waveform Data Compression

2.2.1 Lossy Compression

Reference [14] presents impressive compression rates for waveform and PMU data using adaptive sub-band compression, but involves relatively complex encoding and results in some loss in data fidelity. This may be suitable for some applications, but it is undesirable to lose information for critical measurements in power systems. The processing requirements for a lossy encoder is also likely to be high, such as for performing Fourier or wavelet analysis to extract the prominent harmonic activity.

2.2.2 Lossless Compression

General purpose lossless compression algorithms, such as gzip, have been shown to be relatively ineffective and computationally expensive for typical power system waveform data [10]. The method in [10] shows that a bespoke compression method, with knowledge of the underlying data, can provide both high compression rates and good real-time performance when encoding and decoding. This concept is demonstrated for other domains in [15].

3 Compression Method

3.1 Design Principles

Slipstream is designed primarily for streaming raw measurement data, similar to the IEC 61850-9-2 Sampled Value protocol. It is designed to support high sample rate waveform voltage and current data, but also supports other measurement types and derived data such as frequency measurements. The data compression must be lossless to ensure that there is no impact on measurement fidelity.

The encoder is designed to be flexible. It supports any number of samples per message so that it can be used for different applications where efficient representation of measurement data is useful. For example, 2 to 8 samples per message may be appropriate for real-time applications while still benefitting from compression, and much larger messages could be used for archiving fault or event records, with excellent compression opportunities. A bit error in one message should only invalidate that message, and not future messages.

The encoding method is optimized for the lowest data size for 1) minimal network data transfer for real-time streaming, and 2) small file sizes for saving event captures to permanent storage.

It is assumed that out-of-band communications will agree the sampling rate, the number of variables to be transferred, the identifiers of all measurement sources, and other metadata. This is similar to IEEE C37.118.2 configuration frames, the STTP command channel, or IEC 61850’s SCL configuration files. This helps reduce the amount of data to be send in the main data stream in real-time to allow successful decoding. In particular, the sampling rate and the maximum number of signals encoded in each message must be fixed before real-time communications starts. All values in one data stream must use the same sampling rate.

It is important to ensure that data quality and time synchronization information are strictly preserved and provided for every data sample. Timestamps must be able to accurately represent waveform samples.

Implementations should prefer efficient encode and decode processing, with up-front memory allocations, where possible. However, the use of compression will naturally improve the end-to-end latency, due to the reduced data to be processed and transferred.

The proposed method produces a byte stream which is suitable for a variety of transport methods, such as Ethernet, UDP, TCP, HTTP, WebSocket, STTP, and saving to a file. The transport protocol is responsible for ensuring reliable delivery of messages, if that is required by the application.

3.2 Overview of Method

A key part of the design is encoding and compressing each data stream (e.g., signals from individual sensors) separately, and aggregating each compressed stream into the final message. This is the same method used in [10], and is similar to how high-performance timeseries databases such as TimescaleDB perform compression on columns of data over time, rather than generically compressing each row as they are added to the database [16].

Figure 2 illustrates the process for encoding each data stream.

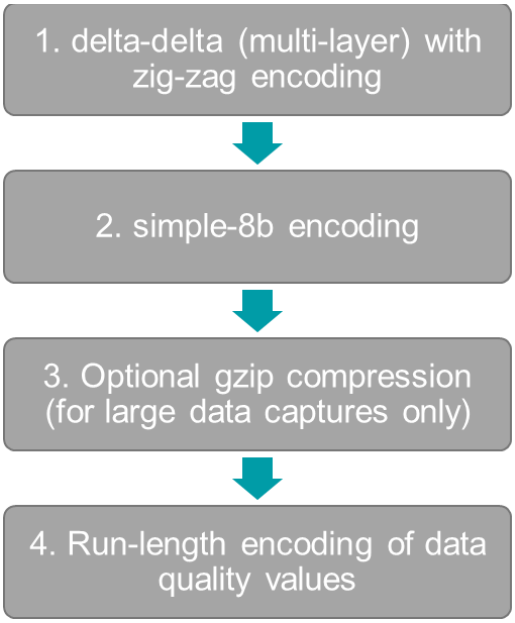

Figure 2: Compression encoder sequence for each data stream

First, delta-delta encoding is used to encode only the differences between consecutive samples, to significantly reduce the data to be encoded. The first sample must be encoded in full. The second sample is encoded as the difference from the first sample (delta encoding). All remaining samples are encoded using delta-delta encoding, and the number of "layers" of the delta-delta encoding can be configured. The encoder supports an arbitrary number of “layers”, which has the effect of applying a linear approximation model to the input signal at higher dimensions.

Wherever possible, variable length integer encoding is used to reduce the number of bytes required for encoding the results from the delta-delta encoding stage. Specifically, zig-zag encoding is used to cater for signed values, as is used in the Protocol Buffers serialization format [17].

If a relatively large number of values is included per message (such as for an event record), simple-8b encoding [18] can be used to improve the packing of multiple variable-length integer values. It is slightly better to use simple-8b for all values, even including the first and second delta values (which will be relatively large compared to subsequent delta values).

Large messages containing thousands of samples can also contain some redundancy after the delta-delta and simple-8b encoding, and can still benefit from generic gzip compression to increase the entropy of the message.

The data quality is assumed to not change very often. Therefore, it is encoded using run-length encoding (RLE). A special run-length of 0 is used to represent that all future values within the same message are the same. So, for the common case where the quality value is 0 for all samples, that can be encoded very efficiently in one byte for the value plus one byte for the number of samples.

3.3 Message Format

Figure 3 illustrates the message format used by the proposed compression method. The maximum number of samples to be encoded per message must be defined ahead of encoding, and must be shared with the decoder.

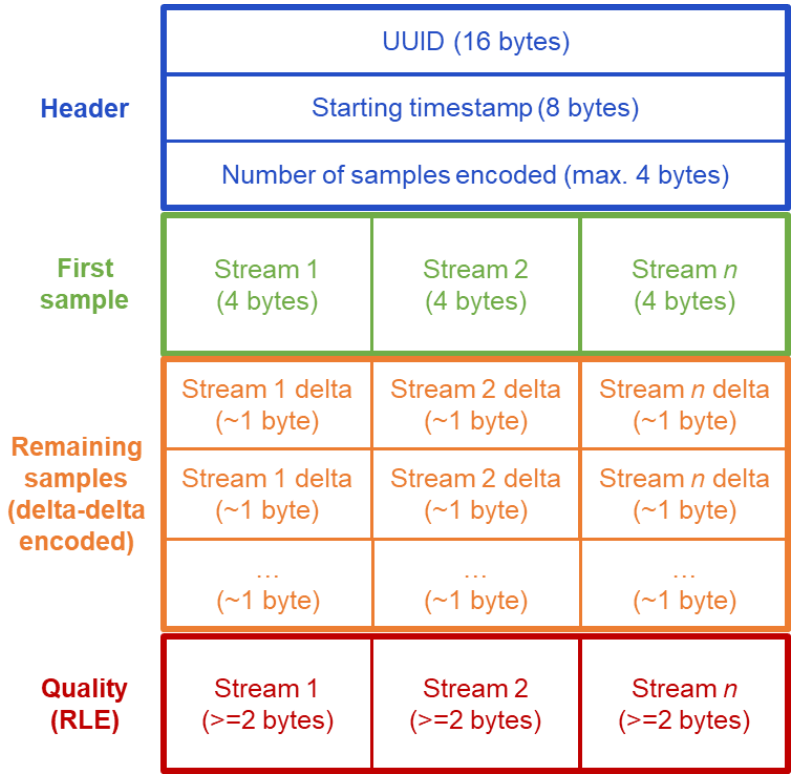

Figure 3: Overview of message format

There are four sections of each message:

1. Header (see below for details).
2. First sample data encoding, per input stream.
3. Second and later sample encoding, per input stream, using delta-delta encoding.
4. Quality values for each sample, per input stream, using RLE.

The message header contains the following fields:

1. Universally unique identifier (UUID), 16 bytes.
2. Timestamp of the first sample, 8 bytes.
3. Number of encoded samples, variable length, maximum of 4 bytes. This is required so that if a stream stops before the maximum number of samples per message, or if the required size is unknown when encoding starts, it is still possible to unambiguously decode the message.

Note that the message format is slightly different when simple-8b encoding is used, as the first and subsequent samples for each stream are processed together. Simple-8b encoding is only worthwhile for messages with at least approximately 16 samples per message.

3.4 Data Types

Data values follow the approach which has already been adopted for IEC 61850-9-2 encoding, and is elaborated in the following subsections.

3.4.1 32-bit signed integer for all data values

This requires a scaled integer representation for floating-point data, as used for SV data. However, the method could be extended for directly representing floating-point values in the future, such as using the method in [19] to compress the exponent and fraction (sometimes called the significand or mantissa) as

separate data streams. For floating-point numbers, XOR deltas produce better compression performance than numerical deltas.

3.4.2 32-bit unsigned integer for quality

This is intended to be based on the IEC 61850 quality specification, for which only 14 bits are presently used (including the "derived" indicator), and only 16 bits should ever be used. It is proposed here that the most significant byte is used for time quality, with the two least significant bytes used for data quality according to the IEC 61850 approach. The exact encoding is not prescribed at present, but 32 bits per data sample have been provisioned, and there is space for future expansion.

3.4.3 64-bit signed integer for timestamp

The timestamp format is based on the Go programming language [20] representation, which provides resolution of 1 ns. The 64-bit format is relative to 1st January 1970 UTC, and is limited to a date between the years 1678 and 2262. Timestamps in STTP are restricted to 100 ns resolution; while suitable for output values such as frequency, it is very inaccurate for waveform data, which could be sampled at inconvenient rates such as 14.4 kHz (so the 69444.44 ns sampling period would be truncated to 69400.00 ns, leading to an intrinsic 44.44 ns error). If the start of the data capture was always aligned to the second roll-over point then the fraction of second value would always be zero, but the protocol should not be restricted in this way. Similarly, IEC 61850 and IEEE C37.118.2 timestamps only dedicate 24 bits to the fraction of second and have a poor resolution limit of 59.6 ns. Note that the timestamp is only encoded once per message to indicate the timestamp of the first encoded sample.

3.5 Other Encoding Details

It is assumed that every sample is included for the defined message size. If a sample was missed (e.g. due to the sensor or underlying data source being unavailable), a zero sample should be added and the data quality should be adjusted appropriately. This simplifies the encoding and significantly reduces the amount of data to be sent because only the starting timestamp needs to be included per message, and all other timestamps can be inferred. Therefore, a single 64-bit field can encode the timestamp, rather than 64 bits per sample.

Decoders must have knowledge of the encoding parameters. This means that tools such as Wireshark may be unable to decode messages and provide diagnostic information, unless it is also able to access and decode the out-of-band data which describes the message format (i.e., the sampling rate and number of variables). It is not possible to decode the quality values until all the data values in a message are decoded first.

It is assumed that three-phase quantities may also include a neutral voltage or current component, similar to the IEC 61850 "LE" profile [21].

The IEC 61850-9-2 SV protocol allows some unnecessary flexibility. For example, in principle, ASDUs in the same message could be mixed from different datasets, although this would be rare in practice. This new protocol does not allow this, in favor of efficiency and simplicity.

It is assumed that bit error detection and correction are handled at the transport layer for communications.

3.6 Spatial Compression

The encoder has the option to specify one data stream as the “reference” for another. This means that only the differences relative to the reference waveform need to be encoded, leading to further compression efficiency. For signals that are typically very similar, such as temperature measurements and voltage measurements on the same phase on feeders connected to a common busbar, this offers excellent additional compression capabilities. This is termed “spatial compression” due to leveraging sensor data streams across multiple locations.

In the best case, this can reduce the number of bytes to be encoded by approximately half, compared to the normal compression approach outlined in this paper. However, relatively small phase differences between the signals, such as introduced by changes in the power system state, can make the spatial compression less effective.

4 Compression Performance

4.1 Overview

Compression performance can typically reduce data to approximately 10% of the theoretical uncompressed sample size (assuming 4 bytes for data, 4 bytes for quality, and 8 bytes for timestamp). Higher sampling rates can compress down to <5%, often requiring less than 1 byte per new sample on average. Shorter messages with fewer samples will achieve compression of 15-25%.

Compared to IEC 61850-9-2 SV, the performance is highly effective due to the additional overhead and repeated data inherent in the SV ASDU structure. For example, sampling at 14.4 kHz with 6 samples (ASDUs) per message (using the "LE" dataset with 8 variables) requires about 589 bytes for SV (including Ethernet header, but not including the "RefrTm" timestamp). This new protocol only requires approximately 134 bytes to convey the same information.

The protocol compression tends to perform better for higher sampling rates, because the difference between samples is less and, on average, fewer bytes are required. Similarly, the protocol compression tends to perform worse for larger RMS values of voltage and current because the differences between samples is greater.

4.2 Detailed Examples

Using tailored lossless data compression, Slipstream can efficiently stream waveform data, or store triggered event records. This ensures that disturbance data can be stored efficiently, and all data is available, including important context before and after the incident.

Data can typically be reduced to less than 10% of the original data size, as shown in Table 1[1]. Interestingly, higher sampling rates can produce smaller message sizes due to efficiencies resulting from smaller deltas between samples, combined with simple-8b encoding.

A data capture containing 10 three-phase voltage or current measurements captured at 4 kHz requires less than 1 Mbps of bandwidth, including the TCP/IP layers and the WebSocket Secure protocol layer to provide secured wide-area network transfer. The equivalent using IEC 61850-9-2 Sampled Values would require approximately 20 Mbps [10], without IP routing layers or encryption. This improvement means that it is possible to transfer waveform data over even legacy utility communications links.

Sampling rate (Hz)	Samples per message	Message size (bytes)	Size relative to original data
4000	10	210	16.4%
4000	4000	40778	8%
14400	6	123	16.9%
14400	14400	2812	0.2%
150000	150000	7238	<0.1%

Table 1: Typical compression performance for common values of samples per message

Table 2 compares the proposed compressed encoding performance to other formats: non-compressed data, CSV, and CSV with generic gzip compression. A dataset containing four voltage measurements (at 400 kV RMS) and four current measurements (500 A RMS) is assumed. A non-nominal system frequency of 50.03 Hz is used, to avoid unrealistic repetitive data which is easily compressible and would be misleading. Note that the size is calculated relative to the number of byes required to store a timestamp, value, and quality for each sample (16 bytes). The first three rows in Table 2 also benefit from an optimization due to only requiring the timestamp to be encoded once per record, rather than for each of the eight data values.

[1] Table 1 can be recreated from the source code at [11].

Data storage format	Sampling rate (Hz)	Samples per message	Message size	Size	Time to encode
Non-compressed raw binary data	14400	144000	10.4 MB	56.3%	45 ms
CSV	14400	144000	12.6 MB	68.1%	379 ms
CSV with gzip	14400	144000	4.2 MB	22.5%	527 ms
Slipstream	**14400**	**144000**	**0.6 MB**	**3.5%**	**100 ms**

Table 2: Comparison with other encoding approaches

The important result is that, in addition to the reduced data size, the encoding process is very fast with Slipstream compression, due to the reduced volume of data to be processed and written, combined with the fact that the compression method is intrinsically simple. This will result in further benefits and reduced latency when transmitting data over local and wide-area networks [10]. This approach therefore removes the key practical barrier of data bandwidth efficiency for utilities wishing to use and exploit waveform data sources.

Sampling rate (Hz)	Samples per message	Message size (bytes)	Size relative to original data
4000	10	236	18.4%
4000	4000	123738	12.1%
14400	6	141	18.3%
14400	14400	123213	6.7%
150000	150000	779918	4.1%

Table 3: Compression performance with high levels of harmonics and noise

Events can be stored at full resolution and sampling rate (at least at 4 kHz). Data quality information is also strictly maintained to clearly identify unreliable data points (e.g., due to loss of the time synchronization source). Derived quantities, such as synchrophasors and harmonics, can be calculated and visualized from the raw data.

4.3 Impact of Noise and Harmonics

A disadvantage of the proposed method is that changes in data values or quality values will tend to increase the message size. This means that more data must be send or recorded when important or interesting events occur, compared with the steady-state. Random noise in the encoded quantities will tend to reduce compression performance. Harmonics will also have this effect, but to a lesser extent than noise. Table 3 illustrates the worst-case, with high levels of harmonics (a THD in the current of 29%) and noise added relative to the results in Table 1, but the end-to-end benefit of compression is still compelling.

It can also be noted that using higher “layers” of delta-delta encoding only leads to improvements in ideal circumstances, with little or no noise present in the signals.

5 Compatibility with STTP

The Slipstream compression approach could be integrated with STTP in the following ways:

1. Using the “Buffer Block” capability which allows arbitrary blocks of binary data to be transferred.
2. Deeper integration with STTP as an alternative compression approach.

6 Demonstration using WebSocket Secure

Figure 4 illustrates Slipstream in use to encode multiple voltage and current measurements from a distributed optical sensing system, using the Synthesis [22] visualization and analysis software. Ten three-phase data streams, sampled at 14.4 kHz, are compressed and transmitted using the WebSocket Secure (WSS) protocol. The WSS stream is directed to the local server so that the received data can be visualized and compared to the original. This is achieved using less than 2 Mbps of data bandwidth to transmit all the waveform data.

Synthesis also delivers derived quantities, such as synchrophasors and harmonics, from the raw data, and can perform functions such as anomaly detection to assist with condition-based maintenance.

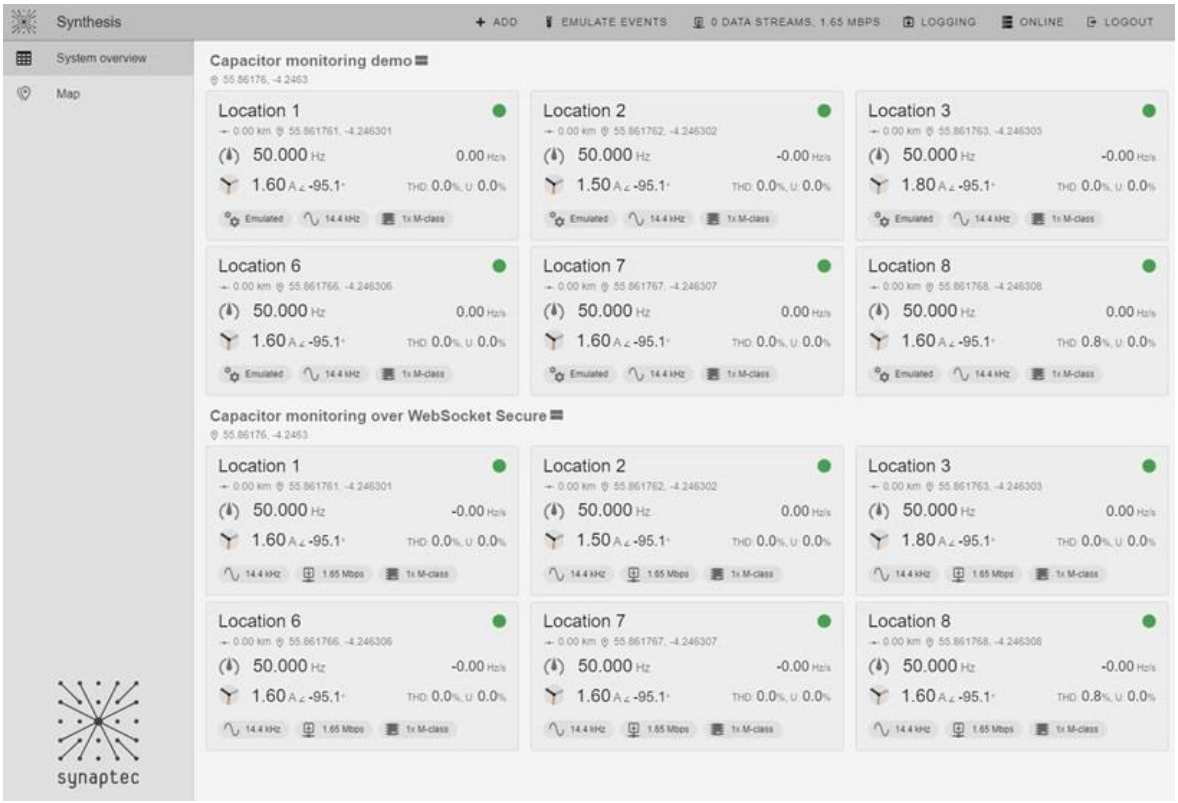

Figure 4: Demonstration of streaming compressed waveform data

7 Future Work

A formal specification of the Slipstream encoder and message format will be prepared. This will include recommended encoding options, such as the optimal number of delta-delta layers. Extending the software to support native floating-point numbers will be investigated. The use of SIMD instructions may also help to further improve encoding and decoding performance on supported processors.

8 Conclusions

The data compression method presented in this paper removes the barriers for utilities to stream, record, and analyse synchronized power system waveform data. It is especially useful for supporting applications such as: deeper classification of events (e.g., for root cause identification for electrical fault), detailed wide-area power quality investigations, building a history of transients experienced by assets for health monitoring, and post-event analysis of major system-wide disturbances.

Synchronized waveform data offers many new applications to support the growth of low-carbon technologies and ensure wider grid stability during this transition. The high-performance lossless data compression proposed in this paper, with an open-source implementation, is designed to be an accelerator for wide adoption of waveform monitoring.

9 References

[1] A. Silverstein and J. Follum, 'High-Resolution, Time-Synchronized Grid Monitoring Devices', NASPI. [Online]. Available: https://www.naspi.org/sites/default/files/reference_documents/pnnl_29770_naspi_hires_synch_grid_devices_20200320.pdf

[2] J. Follum *et al.*, 'Phasors or Waveforms: Considerations for Choosing Measurements to Match Your Application', Pacific Northwest National Laboratory, PNNL-31215, 2021.

[3] S. Blair and P. Orr, 'Synchronised Wide-Area Continuous Point on Wave Disturbance Recording', presented at the PAC World, 2021.

[4] W. Xu, Z. Huang, X. Xie, and C. Li, 'Synchronized Waveforms – A Frontier of Data-Based Power System and Apparatus Monitoring, Protection, and Control', *IEEE Trans. Power Deliv.*, vol. 37, no. 1, pp. 3–17, Feb. 2022, doi: 10.1109/TPWRD.2021.3072889.

[5] S. Blair, 'Building Resilience and Secure Automation into Transmission and Distribution Systems', *T&D World*, Dec. 02, 2020. https://www.tdworld.com/overhead-transmission/article/21149071/building-resilience-and-secure-automation-into-transmission-and-distribution-systems

[6] C. Wang, C. Mishra, K. D. Jones, and L. Vanfretti, 'Identifying Oscillations Injected by Inverter-Based Solar Energy Sources in Dominion Energy's Service Territory using Synchrophasor Data and Point-on-Wave Data', presented at the NASPI Work Group Virtual Meeting, Apr. 13, 2021. [Online]. Available: https://www.naspi.org/sites/default/files/2021-04/D1S1_02_wang_dominion_naspi_20210413.pdf

[7] X. Jiang, B. Stephen, and S. McArthur, 'Automated Distribution Network Fault Cause Identification With Advanced Similarity Metrics', *IEEE Trans. Power Deliv.*, vol. 36, no. 2, pp. 785–793, Apr. 2021, doi: 10.1109/TPWRD.2020.2993144.

[8] M. Izadi and H. Mohsenian-Rad, 'Synchronous Waveform Measurements to Locate Transient Events and Incipient Faults in Power Distribution Networks', *IEEE Trans. Smart Grid*, vol. 12, no. 5, pp. 4295–4307, Sep. 2021, doi: 10.1109/TSG.2021.3081017.

[9] V. Skendzic and D. Dolezilek, 'New and Emerging Solutions for Sampled Value Process Bus IEC 61850-9-2 Standard – An Editor's Perspective', presented at the Southern African Power System Protection & Automation Conference, Johannesburg, South Africa, 2017. [Online]. Available: https://cms-cdn.selinc.com/assets/Literature/Publications/Technical%20Papers/6797_NewEmerging_DD_20170309_Web2.pdf?v=20181019-141138

[10] S. M. Blair, A. J. Roscoe, and J. Irvine, 'Real-time compression of IEC 61869-9 sampled value data', in *2016 IEEE International Workshop on Applied Measurements for Power Systems (AMPS)*, 2016, pp. 1–6. doi: 10.1109/AMPS.2016.7602854.

[11] S. Blair, *Slipstream*. Synaptec Ltd, 2021. [Online]. Available: https://github.com/synapteccltd/slipstream

[12] J. R. Carroll and F. R. Robertson, 'A Comparison of Phasor Communication Protocols', Pacific Northwest National Laboratory, PNNL-28499, 2019. [Online]. Available: https://www.osti.gov/servlets/purl/1504742

[13] 'Streaming Telemetry Transport Protocol', *GitHub*. https://github.com/sttp

[14] X. Wang, Y. Liu, and L. Tong, 'Adaptive Subband Compression for Streaming of Continuous Point-on-Wave and PMU Data', *IEEE Trans. Power Syst.*, vol. 36, no. 6, pp. 5612–5621, Nov. 2021, doi: 10.1109/TPWRS.2021.3072882.

[15] D. Lemire and L. Boytsov, 'Decoding billions of integers per second through vectorization', *Softw. Pract. Exp.*, vol. 45, no. 1, pp. 1–29, Jan. 2015, doi: 10.1002/spe.2203.

[16] 'TimescaleDB 2.3: Improving columnar compression for time-series on PostgreSQL', *Timescale Blog*, May 26, 2021. https://blog.timescale.com/blog/timescaledb-2-3-improving-columnar-compression-for-time-series-on-postgresql/

[17] 'Encoding | Protocol Buffers | Google Developers'. https://developers.google.com/protocol-buffers/docs/encoding#signed_integers

[18] V. N. Anh and A. Moffat, 'Index compression using 64-bit words', *Softw. Pract. Exp.*, vol. 40, no. 2, pp. 131–147, 2010, doi: 10.1002/spe.948.

[19] M. P. Andersen and D. E. Culler, 'BTrDB: Optimizing Storage System Design for Timeseries Processing', in *14th USENIX Conference on File and Storage Technologies (FAST 16)*, 2016, pp. 39–52. Accessed: Mar. 25, 2016. [Online]. Available: https://www.usenix.org/conference/fast16/technical-sessions/presentation/andersen

[20] The Go Authors, 'The Go Programming Language Specification'. https://golang.org/ref/spec

[21] UCA International Users Group, ‘Implementation Guideline for Digital Interface to Instrument Transformers Using IEC 61850-9-2’, 2004.
[22] ‘Synthesis™: our visualisation and analytics platform | Synaptec’. https://synapt.ec/products/synthesis