DOI: 10.3217/978-3-99161-057-1-134

Towards a plug and play population-based structural health monitoring aggregation pipeline design for resource constrained systems

D. R. Lim[1, https://orcid.org/0009-0007-9068-9428], A. J. Ferguson[2, https://orcid.org/0000-0002-3794-705X], C. O'Higgins[2, https://orcid.org/0000-0001-7034-005X], D. Hester[2, https://orcid.org/0009-0007-2280-4340], R. Woods[1, https://orcid.org/0009-0007-2280-4340]
[1]School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT9 5AH, U.K.
[2]School of Natural and Built Environment, Queen's University Belfast, Belfast BT9 5AG, U.K.
email: {dlim04, a.j.ferguson, c.ohiggins, d.hester, r.woods}@qub.ac.uk

ABSTRACT: The practical implementation of Population-based Structural Health Monitoring (PBSHM) often involves distributed which face challenges from limited compute resources, power budgets, and variable communication bandwidths, for example, IoT devices with battery-powered wireless sensor network gateways uploading data over metered connections. Building upon the data flow architecture presented in our first paper in this special session, this paper demonstrates how inherent state within data streams' can be leveraged for optimisation. This paper introduces a novel, plug-and-play aggregation pipeline specifically designed to address these limitations. We present an optimised data representation and transmission strategy that minimises computational and bandwidth requirements at the network edge. By leveraging efficient data serialisation techniques, our pipeline achieves a significant reduction in data payload size with negligible information loss, thereby enhancing the scalability and financial viability of PBSHM systems. This work validates an enabling technology for the real-world deployment of large-scale, low-power monitoring ecosystems. It does so by comparing two data formats, JavaScript Object Notation (JSON) and Concise Binary Object Representation (CBOR), using monitoring data from a long-term bridge campaign. The results show a reduction in the volume of transmitted data by up to 12.2 times.

KEY WORDS: Population-based Structural Health Monitoring (PBSHM); Resource-constrained systems; data optimisation; data serialisation.

1 INTRODUCTION

Infrastructure monitoring requires methods to track structural condition and optimise maintenance schedules. Structural health monitoring (SHM) has developed considerably in recent years [1]. Population-Based SHM (PBSHM) offers potential advantages over traditional SHM approaches by analysing data across multiple structures rather than analysing each independently. This approach enables knowledge transfer between similar structures, improving damage detection and diagnostic capabilities [2], [3], [4], [5].

While the definition of a computing framework for PBSHM is established [6], the practicalities of data logistics in heterogeneous, resource-scarce environments remain a significant research gap. Implementing PBSHM systems requires the integration of data from many different monitoring systems. Previous work has developed a methodology for the transmission of data from a data generator, to the PBSHM database [7].

Practical deployment faces challenges that haven't been fully addressed to date, for example, many monitoring scenarios such as wireless sensor networks, remote sites, and temporary deployments operate with limited resources. These systems must operate within constraints on bandwidth, computing power, energy supply, and storage capacity. These limitations become more significant as PBSHM networks expand to include more structures and sensors.

The focus of this work lies in the creation of a holistic and optimised data aggregation pipeline tailored for resource constrained systems on the network edge. We directly address the challenge of minimising data footprints prior to their arrival at a central repository, a critical step that has hitherto been underexplored. This paper thus provides a foundational component for the next generation of truly scalable and deployable PBSHM systems.

This paper presents a stateful, connection-oriented approach to PBSHM data transmission designed for resource-constrained environments. Our transmission process establishes session contexts for data generators, handling metadata exchange during session initialisation rather than with each data sample for the period of the session. This reduces redundant information while maintaining schema compliance and plug-and-play functionality, allowing new data generators to join the system with minimal configuration. Functionality that becomes particularly useful for temporary monitoring campaigns or rapid PBSHM monitoring system rollout.

The main contributions of this work are:

- Development of a stateful, session-oriented pipeline architecture for PBSHM data transmission that reduces redundancy through session payload optimisation while retaining plug-and-play data generator integration.
- Demonstration and validation of a resource-efficient aggregation pipeline, using empirical evaluation with field monitoring campaign data to demonstrate its feasibility for real-world PBSHM applications.

The paper is organised as follows: Section 2 introduces the concept of a session-oriented pipeline; Section 3 details the case study used to validate the data and a proposed pipeline design; Section 4 analyses the results; and Section 5 concludes with a summary and future directions.

2 METHODOLOGY

To optimise data flow from a structure to a PBSHM server, a key aspect that must be considered is the data representation

DOI: 10.3217/978-3-99161-057-1-134

used. The PBSHM schema defines the representation that must be provided to the PBSHM Framework.

The PBSHM data integration pipeline (Figure 1) segments the process from the data generator, which is an entity that produces data either already compliant with or transformable to the PBSHM schema, to the PBSHM Network, Framework, and Database, using defined scopes [7]. Using this definition, anything in and transmitted to the PBSHM Network, Framework and Database scope must be within the PBSHM schema representation. However, this approach allows data within scopes preceding the PBSHM Network, Framework, and Database to take any form, as long as it can be reconciled with the PBSHM schema, even if it does not initially conform to it. By exploiting this, we can attempt to optimise the data representation in the "structure" and "aggregation" scopes.

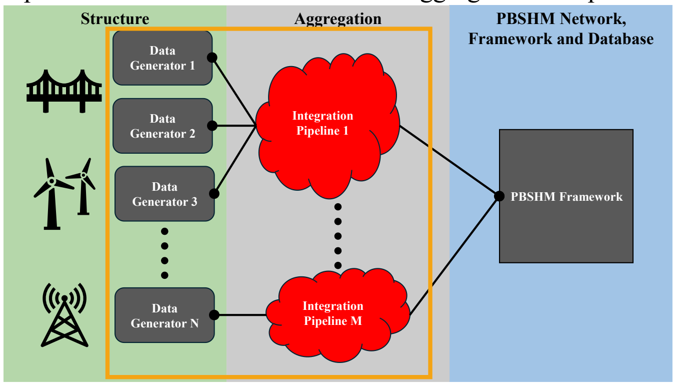


Figure 1. The PBSHM data integration system in which copes where data representations are not enforced is highlighted

Although the data representation is not explicitly defined within these scopes, there are significant advantages to developing a reusable implementation, particularly for time series channel data. Optimising data representations is crucial when considering the application of streaming techniques to handle vast amounts of data efficiently. Given the potential for integrating multiple data generators to monitor a single system, such as data from various sensors in an SHM system, SCADA data, and weather data from an application programming interface (API) - a unified aggregation pipeline can provide a sensible solution. This approach would support all these sensors cohesively, ensuring efficient and effective data management. Creating a generic data representation that reduces redundancy within that data and therefore communications bandwidth utilisation would allow systems with resource constraints to share data integration pipelines which increases scalability while maintaining acceptable bandwidth utilisation.

A primary motivation for performing data reduction and standardisation directly on the sensor node is to achieve a truly 'plug-and-play' and interoperable system. While it is possible to offload metadata enrichment to a gateway or middleware server in its entirety, such an approach creates a dependency, binding the sensor node to a specific gateway with the requisite processing capabilities. This method, by contrast, ensures that each sensor node transmits a self-contained, efficient, and standardised data packet. This makes the capable of being integrated into any PBSHM ecosystem with a compatible endpoint, without requiring specialised intermediate hardware. This approach enhances deployment flexibility, as a node can be replaced or relocated with the guarantee that it will function correctly with the central server, independent of the local network topology or available gateway resources. This is particularly advantageous in large-scale, heterogeneous deployments where multiple vendors and network types may coexist.

For a plug-and-play aggregation pipeline, enough information must be encoded into the data representation to enable compliance with the PBSHM schema when the data is transmitted to the PBSHM server whilst reducing the redundancy of information within said data representation. To achieve this, we must first identify the key aspects of the PBSHM schema that must be retained.

In this paper, we focus on time series channel data. Therefore, the key information that's encoded into a channel data object and must therefore be recoverable is:

- Structure name
- Population name
- Timestamp
- Channel name(s)
- Channel type(s)
- Channel unit(s)
- Channel value(s)

First, we must consider implementing this in such a way that all this information is explicitly stated for each timestamp. To illustrate this, consider two JavaScript object notation (JSON) objects compliant with the PBSHM schema that represent multiple sensor readings at two different time stamps one second apart as shown in Figure 2.

We can identify multiple items of redundant information across the two objects (i.e., structure name, population, channel names, channel types and channel units).

Many communication protocols incorporate mechanisms for tracking the origin of messages, enabling the inference of the communication state. For instance, protocols like TCP/IP and MQTT include metadata that identifies the sender and the context of the message. This capability allows us to avoid sending redundant information, as we can pair existing data with the sender's identity.

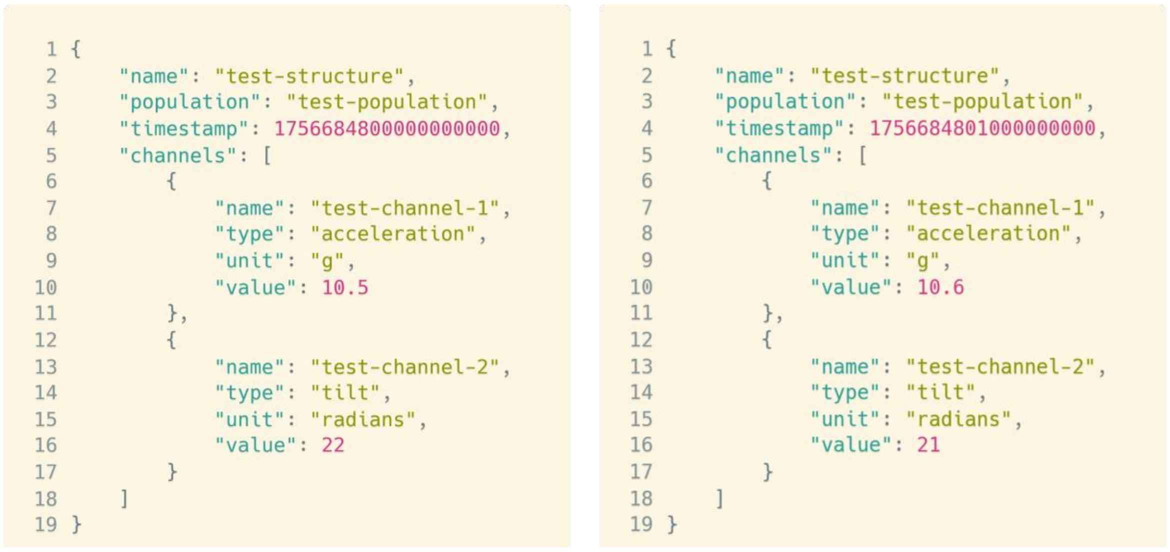
Therefore, we can split up the object into duplicated and non-duplicated data (when considering multiple objects from the same data generator) as shown in Table 1. As can be seen; by only transmitting updated data (or non-duplicated fields), we can significantly limit the number of fields that have to be transmitted for every object.

Table 1. Identifying duplicated and non-duplicated fields within the PBSHM Schema when considering multiple time series objects

Duplicated Fields	Non-duplicated Fields
Structure name	Timestamp
Population name	Channel value(s)
Channel name(s)	
Channel type(s)	
Channel unit(s)	

As such, we can define two new objects, the *initialisation* in which all the duplicated fields for a given data generator are

DOI: 10.3217/978-3-99161-057-1-134



```json
{
    "name": "test-structure",
    "population": "test-population",
    "timestamp": 1756684800000000000,
    "channels": [
        {
            "name": "test-channel-1",
            "type": "acceleration",
            "unit": "g",
            "value": 10.5
        },
        {
            "name": "test-channel-2",
            "type": "tilt",
            "unit": "radians",
            "value": 22
        }
    ]
}
```

```json
{
    "name": "test-structure",
    "population": "test-population",
    "timestamp": 1756684801000000000,
    "channels": [
        {
            "name": "test-channel-1",
            "type": "acceleration",
            "unit": "g",
            "value": 10.6
        },
        {
            "name": "test-channel-2",
            "type": "tilt",
            "unit": "radians",
            "value": 21
        }
    ]
}
```

Figure 2. Two example JSON documents containing multiple sensor readings one second apart

represented and the *sample* in which we represent the non-duplicated fields.

With these new objects, we can define the process by which communications take place. For the data generator, first, we send the initialisation object, and then we send every sample object available as shown in Figure 3. These sample objects can optionally be added to a buffer to increase the efficiency of communications (i.e. only turning on radios for brief periods or trying to limit the number of packets sent). For the case in which each sample object is sent individually the buffer size would be one.

With these communication processes if there was a need to change the format of the sample objects (i.e. adding a channel, removing a channel or changing properties about a channel) this could also be achieved by closing the communication session and re-opening it with the new channel information in the same way the session is initialised as described previously. Furthermore, if the data generator was disabled for any reason the communications would be closed, and the process would also have to be re-initialised. In such cases, any incomplete sample objects and session state affected by the disconnection would be discarded. Any samples still in the send buffer would then need to be retransmitted where de-duplication would take place between the transit and the PBSHM framework, network and database.

Within the aggregation scope (shown in Figure 4), we then receive the initialisation message from the data generator and then receive the buffer (which fulfils the role of the cache from the PBSHM data integration system). From the buffer, we add the “redundant fields” to the sample to make a JSON object compliant with the PBSHM Schema and then transmit said object to the PBSHM Framework. We then wait to receive a new buffer or close the communication.

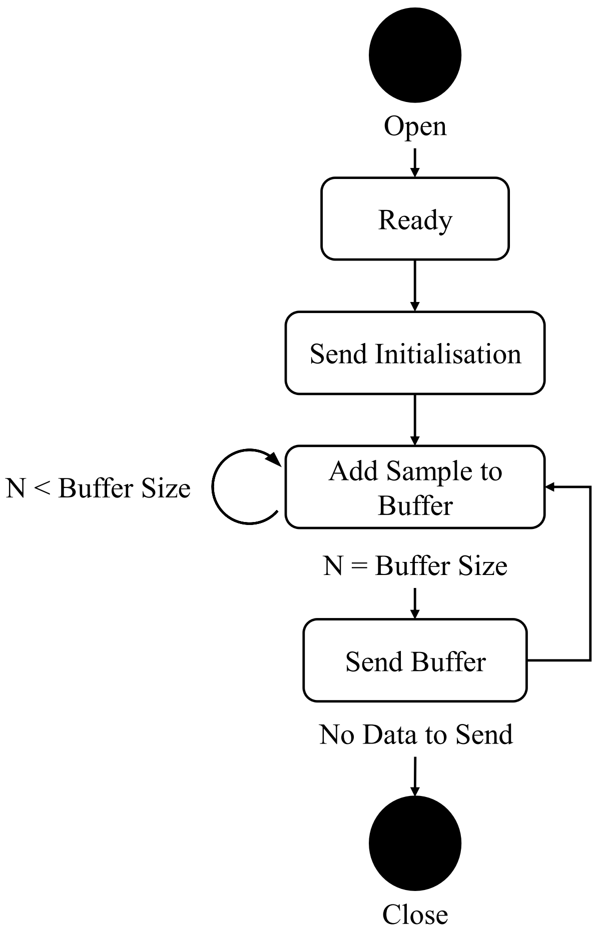


Figure 3. Data generator communications process

These two processes allow the use of a reduced communication schema to reduce the bandwidth utilised in the transmission of PBSHM data from a given data generator.

DOI: 10.3217/978-3-99161-057-1-134

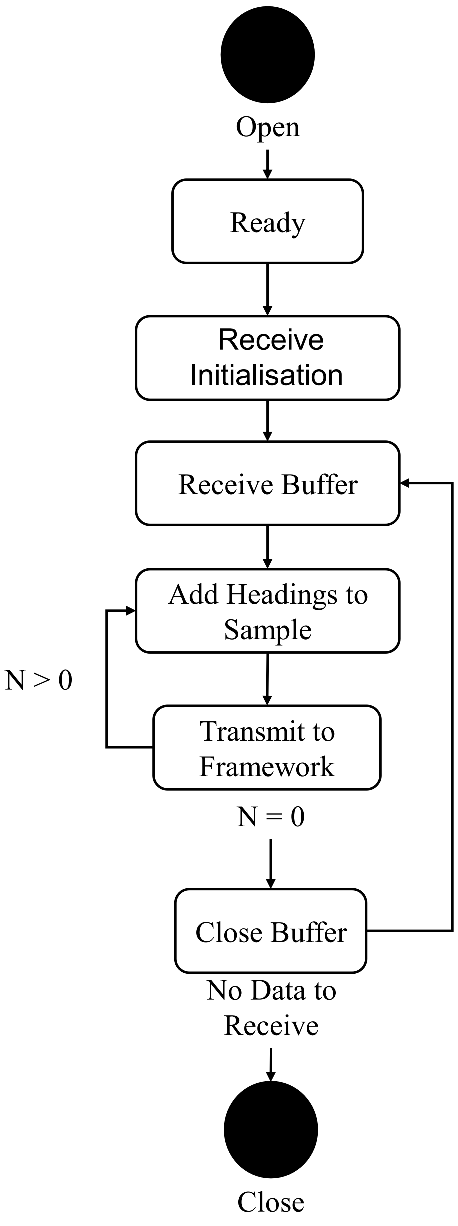


Figure 4. Aggregation pipeline communication process

3 CASE STUDY

The structure under test is a bridge that supports a dual carriageway, which consists of two lanes designated for one-way traffic, as well as a footpath for pedestrians. The structure of the bridge is composed of 13 concrete Y beams that provide support for a reinforced concrete deck. Adjacent is an older masonry arch bridge. These two bridges are constructed without any visible gap between them, making it unclear how much the concrete bridge deck is restrained on one side.

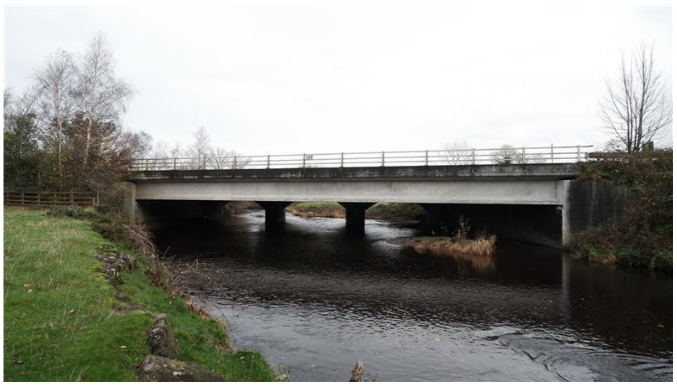


Figure 5. The structure under test in the case study

The SHM system implemented for this bridge was designed with versatility and ease of installation in mind. This system allows for the accelerometer to be installed directly on the surface of the bridge deck. The placement of the accelerometer is optimised by accounting for the variations in bridge types. It comprises a single MEMS accelerometer and an environmental sensor. The accelerometer employed is the Multifunction Extended Life (MEL) Data Logger from Gulf Coast Data Concepts. This device measures acceleration in three axes within a range of ±2 g and includes a real-time clock to timestamp each acceleration measurement. The acceleration data collected by the sensor is stored locally on an SD card at a sampling rate of 128 Hz, and the sensor is powered by two D-cell batteries, which provide a run-time of up to 60 days of continuous recording. The MEL accelerometer is housed in an enclosure that is securely attached to the deck of each bridge. An example of one of these enclosures is shown in Figure 6.

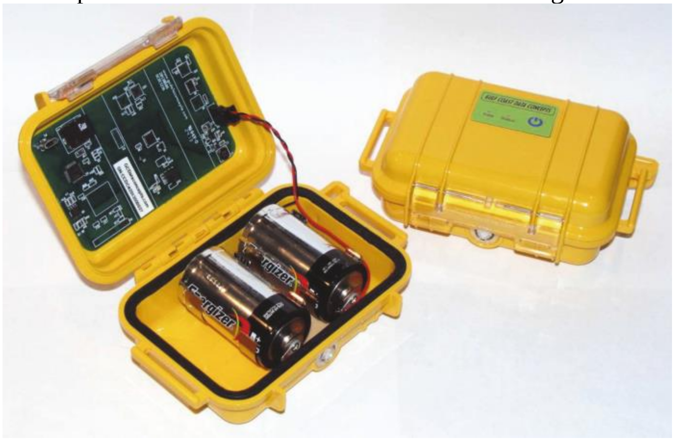


Figure 6. MEL accelerometer and enclosure

DOI: 10.3217/978-3-99161-057-1-134

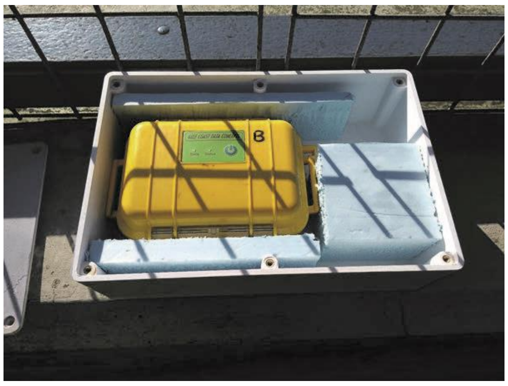


Figure 7. MEL sensor in place

In addition to the accelerometer, the SHM system includes an environmental sensor that measures both air temperature and humidity. To ensure accurate temperature readings, the environmental sensors are not placed within the enclosure to avoid the effects of solar gain. Instead, these sensors are positioned out of direct sunlight, to provide a representative measurement of the local air temperature.

3.1 Data description

The data provided by the system is in a comma-separated values (CSV) format, of which an example showing ten values is shown in Table 2. Within the CSV document, the first 9 rows contain contextual information about the samples. The following rows contain time series data.

Table 2. Extract from the CSV data provided by the data logger containing headers and 15 samples

;Title	http://www.gcdataconcepts.com	x2-2	Kionix KXRB5-2050		
;Version	1107	Build date	Oct 20 2015	SN:CCDC100226A4EB2	
;Start_time	2018-11-08	11:48:23.440			
;Temperature	13	deg C	Vbat	3076	mv
;Gain	high				
;SampleRate	128	Hz			
;Deadband	0	counts			
;DeadbandTimeout	0	sec			
;Headers	time	Ax	Ay	Az	
0.222	49	328	-13136		
0.229	28	314	-13129		
0.237	7	296	-13099		
0.245	21	296	-13101		
0.253	30	324	-13136		
0.261	33	341	-13161		

To evaluate the efficacy of the proposed pipeline architecture, a 30-minute window of the data has been selected. This can be seen represented in Figure 10.

3.2 Test pipeline models

As such, we can develop a pipeline that would allow transmission of data from the bridge under test to the PBSHM server which is shown at the top of the Universal Markup Language (UML) diagram of Figure 8. This model of the pipeline first extracts the data from the GCDC data store and converts it to the PBSHM schema format. Data in the PBSHM schema format is then loaded into the PBSHM Schema Cache on the Docker Server, where it is then loaded into the PBSHM server via a Representational State Transfer (REST) API.

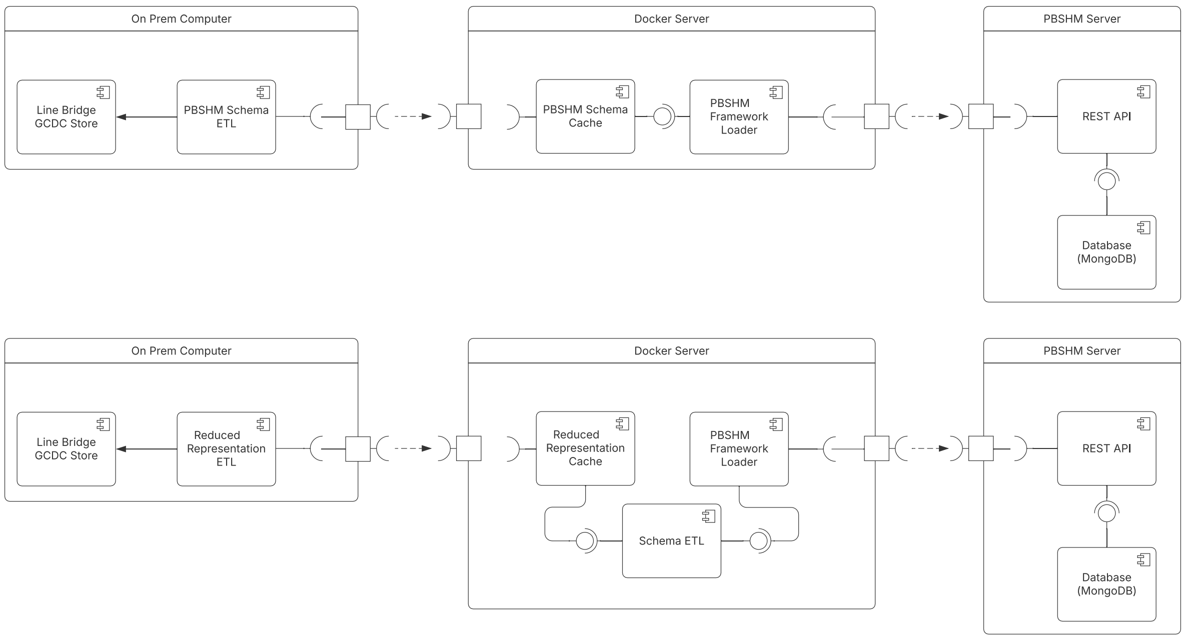


Figure 8. Universal markup language diagram showing two data integration pipelines (PBSHM Schema rep. above, reduced data rep. below)

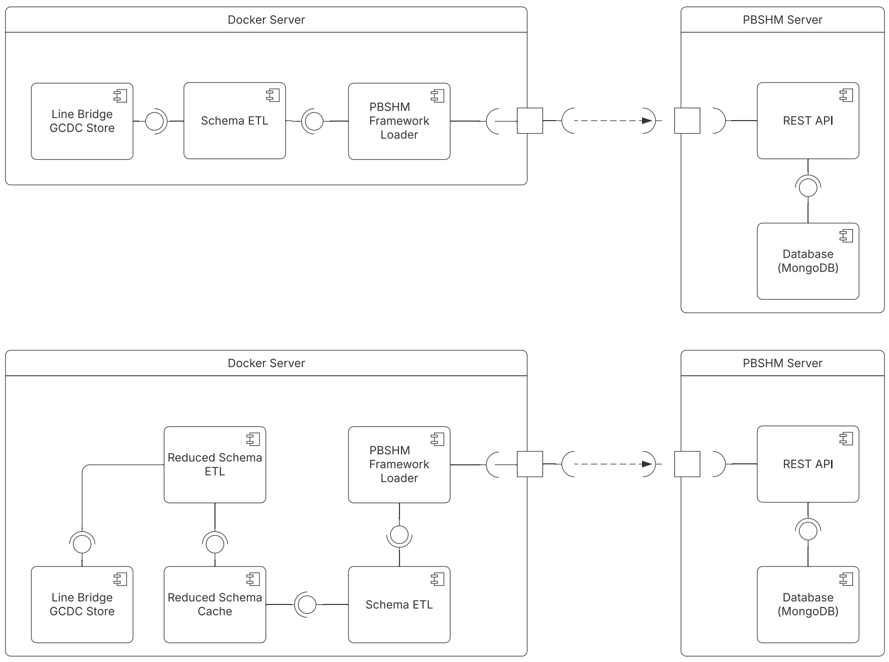


Figure 9. Universal markup language diagram showing the simulation test bench (PBSHM Schema rep. above, reduced data rep. below)

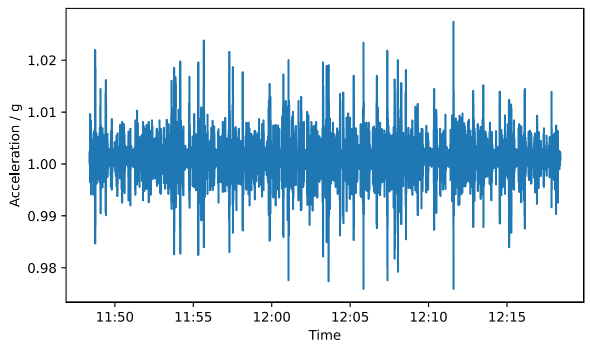
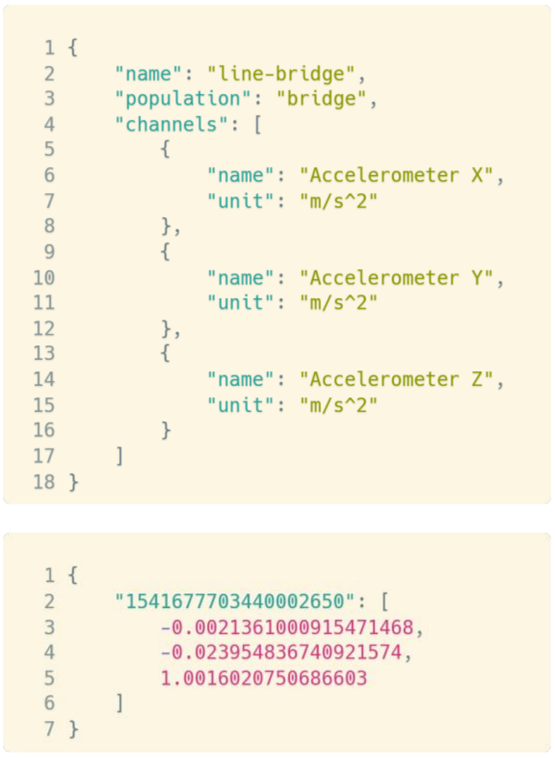


Figure 10. Single-axis raw acceleration over 30 minutes measured using the MEL accelerometer

The diagram at the bottom of Figure 8 shows a similar pipeline, however, this uses a reduced data representation as discussed in Section 2. Within the on-premises computer, data is extracted from the GCDC store and transformed into an initialisation object (which contains all the duplicated fields shown in Table 1) and a set of samples objects. The initialisation object, of which an example is given at the top of Figure 11, is first transmitted to the Reduced Representation Cache which resides on the docker server. Samples, an example of which is shown in the bottom of Figure 11, are then transmitted to the Reduced Representation Cache. These samples are then transformed into PBSHM Schema compliant files which are then uploaded to the PBSHM server via the PBSHM server.

From these models, we can simulate the performance of the multiple data representations (namely the PBSHM schema in a JSON format and concise binary object representation (CBOR) format as well as a reduced representation in both JSON and CBOR formats) using real-world data captured from the line bridge over a simulation test bench.


```
{
    "name": "line-bridge",
    "population": "bridge",
    "channels": [
        {
            "name": "Accelerometer X",
            "unit": "m/s^2"
        },
        {
            "name": "Accelerometer Y",
            "unit": "m/s^2"
        },
        {
            "name": "Accelerometer Z",
            "unit": "m/s^2"
        }
    ]
}
```

```
{
    "1541677703440002650": [
        -0.0021361000915471468,
        -0.023954836740921574,
        1.0016020750686603
    ]
}
```

Figure 11. Initialisation object (above) and sample object (below) in the reduced data representation

DOI: 10.3217/978-3-99161-057-1-134

4 RESULTS

To assess the performance of the methodology presented in Section 2 a simulation of the systems shown in Figure 8 was developed using individual docker containers running on a docker server (with the PBSHM schema version of the pipeline shown above in Figure 9 and the reduced representation version shown below).

This simulation test bench emulates all of the components in the systems described in Figure 8, however, integrates all the components that were shown to be implemented with on-premises equipment in the Docker Server to allow for greater control of the data pipeline and easy access to metrics.

In this paper, the size of the total payload for the selected 30-minute window is the metric used to evaluate the efficacy of the pipeline. This was chosen as within this setting, the size of the data transmitted over the period will likely impact bandwidth requirements and power requirements for resource-constrained systems.

This test bench was used to evaluate multiple data representations. The PBSHM Schema in JSON format was used as a benchmark data representation as it provides all the necessary context for the PBSHM server. This was then repeated by the reduced representation in JSON format. Finally, both tests were repeated with CBOR which allows a 1:1 mapping with JSON allowing it to be implemented with little to no modification to the existing PBSHM schema.

Table 3. Size of a 30-minute capture (230,400 samples of tri-axis acceleration data) in CSV, PSBHM Schema, and reduced representation data formats

Format	Size (MiB)	Ratio to CSV
CSV	5.14	1.00
PBSHM Schema (JSON)	131.42	25.58
Reduced (JSON)	34.30	6.68
PBSHM Schema (CBOR)	45.26	8.81
Reduced (CBOR)	10.77	2.10

From the results captured and presented in Table 3, it can be concluded that using the PBSHM schema in a JSON format dramatically increases the storage required for the 30-minute sample window compared to the original CSV data. This can be significantly reduced using a compressed format like CBOR, reducing the data size by 2.9x.

However, by using the reduced data representation, this can be further reduced. In the JSON format, the reduced data representation offered a 3.8x size reduction whilst the CBOR format of the reduced data representation offered a 12.2x size reduction when compared to the PBSHM Schema in JSON format. These results are plotted as shown in Figure 12.

By utilising the reduced data representation in the CBOR format, the ultimate size of data to be transmitted is 2.1x the original CSV data.

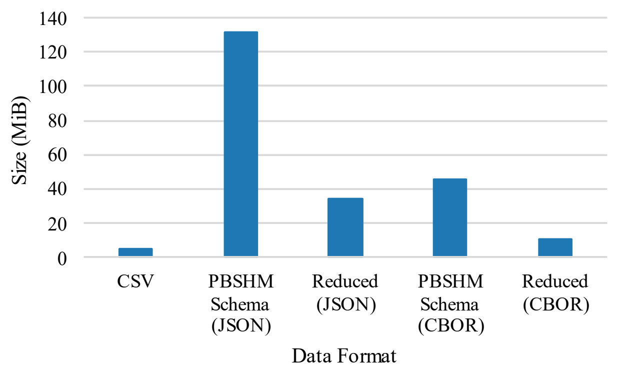


Figure 12. Bar chart plotting the different data representations and formats of a 30-minute capture (230,400 samples of tri-axis acceleration data)

5 DISCUSSION, LIMITATIONS AND FUTURE WORK

The findings presented in this paper provide a strong proof-of-concept for the proposed pipeline; however, several limitations should be acknowledged, and these point towards avenues for future research.

The results in Table 2 show a reduced file size from CSV format produced by the data acquisition system when compared to the schema under test. This shows that in situations in which an aggregation pipeline can be implemented on a case-by-case basis dependent tailored to the data produced by the data generator there is an inherent advantage to doing so. This will be true in almost all cases where this is possible as no extra information is required to allow the transformation of data produced by the data generator into the PBSHM schema format. However, as stated, this requires that each implementation of the aggregation scope be tailored to the specific data generator.

By utilising this plug and play aggregation pipeline design, it is possible, to reuse components across multiple data generators. This limits the effort required to develop new tailored aggregation components for each data generator as well as allows data from many data generators to be integrated into the PBSHM data ecosystem using few data aggregation components.

However, the validation was conducted using a homogenous set of accelerometer data. The “plug-and-play” architecture is designed to be extensible, yet its performance with a more diverse array of sensor modalities, such as strain gauges, acoustic emission sensors, or environmental sensors has not yet been empirically validated. Future work will focus on developing and testing data models for these sensor types within the pipeline.

Furthermore, while our work focuses on data representation, it does not currently incorporate specific protocols for secure data transmission. For deployment on critical infrastructure, ensuring data integrity and confidentiality through authenticated encryption schemes (e.g., using TLS/DTLS) could be employed.

A key focus of our future work will be on hardware-in-the-loop testing, further developing robust reference implementations of the technology demonstrated within the aggregation scope to provide practical guidance for both current and future SHM system operators seeking to integrate their monitoring within PBSHM data domains.

6 CONCLUDING REMARKS

This paper has presented the potential of session-defined communication for PBSHM data within PBSHM data aggregation pipelines. By defining the processes that would be undertaken to achieve reduced bandwidth utilisation, we have demonstrated how to achieve reduced bandwidth requirements whilst maintaining plug-and-play functionality for PBSHM monitoring systems.

We demonstrate that it is possible to reduce the amount of data required to provide the necessary context for PBSHM when transmitting sample data providing a method to reduce the required data by 12.2x. This results in a transmission size that is 2.1x the original data size, which whilst a marked increase, allows for reasonable requirements for resource-constrained PBSHM monitoring systems.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) through the ROSEHIPS project (Grant EP/W005816/1). For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising. The authors of this paper also gratefully acknowledge Daniel Brennan of The University of Sheffield for providing insights into the PBSHM ecosystem.

REFERENCES

[1] V. R. Gharehbaghi *et al.*, ‘A Critical Review on Structural Health Monitoring: Definitions, Methods, and Perspectives’, *Arch Computat Methods Eng*, vol. 29, no. 4, pp. 2209–2235, Jun. 2022, doi: 10.1007/s11831-021-09665-9.

[2] L. A. Bull *et al.*, ‘Foundations of population-based SHM, Part I: Homogeneous populations and forms’, *Mechanical Systems and Signal Processing*, vol. 148, p. 107141, Feb. 2021, doi: 10.1016/j.ymssp.2020.107141.

[3] J. Gosliga, P. A. Gardner, L. A. Bull, N. Dervilis, and K. Worden, ‘Foundations of Population-based SHM, Part II: Heterogeneous populations – Graphs, networks, and communities’, *Mechanical Systems and Signal Processing*, vol. 148, p. 107144, Feb. 2021, doi: 10.1016/j.ymssp.2020.107144.

[4] P. Gardner, L. A. Bull, J. Gosliga, N. Dervilis, and K. Worden, ‘Foundations of population-based SHM, Part III: Heterogeneous populations – Mapping and transfer’, *Mechanical Systems and Signal Processing*, vol. 149, p. 107142, Feb. 2021, doi: 10.1016/j.ymssp.2020.107142.

[5] G. Tsialiamanis, C. Mylonas, E. Chatzi, N. Dervilis, D. J. Wagg, and K. Worden, ‘Foundations of population-based SHM, Part IV: The geometry of spaces of structures and their feature spaces’, *Mechanical Systems and Signal Processing*, vol. 157, p. 107692, Aug. 2021, doi: 10.1016/j.ymssp.2021.107692.

[6] D. S. Brennan, J. Gosliga, E. J. Cross, and K. Worden, ‘Foundations of population-based SHM, Part V: Network, framework and database’, *Mechanical Systems and Signal Processing*, vol. 223, p. 111602, Jan. 2025, doi: 10.1016/j.ymssp.2024.111602.

[7] D. R. Lim, A. J. Ferguson, D. S. Brennan, D. Hester, and R. Woods, ‘A methodology for data collection and aggregation in population-based structural health monitoring ecosystems’, presented at the 13th International Conference on Structural Health Monitoring of Intelligent Infrastructure, Graz, Austria, Sep. 2025.