Structural Health Monitoring by Payload Compression in Wireless Sensors Network: An Algorithmic Analysis

Md. Motaharul Islam¹Ngamsie Njimbouom Soualihou² and Safi Faizullah³
¹Assistant Professor, Department of Computer Science, Islamic University of Madinah, SAUDI ARABIA
²Head of IT, ORYX Consulting, CAMEROON
³Professor, Department of Computer Science, Islamic University of Madinah, SAUDI ARABIA

¹Corresponding Author: motahariut@gmail.com

ABSTRACT

Structural health monitoring is the fact of estimating the state of structural health or detecting the changes in structure that affect its performance. The traditional approach to monitor the structural health is by using centralized data acquisition hub wired to tens or even hundreds of sensors, and the installation and maintenance of these cabled systems represent significant concerns, prompting the move toward wireless sensor network. As cost effectiveness and energy efficiency is a major concern, our main interest is to reduce the amount of overhead while keeping the structural health monitoring accurate. Since most of the compression algorithm is heavy weight for wireless sensor network with respect to payload compression, here we have analyzed an algorithmic comparison of arithmetic coding algorithm and Huffman coding algorithm. Evaluation shows that arithmetic coding is more efficient than Huffman coding for payload compression.

Keywords---Wireless Sensor Network, TinyOS, Payload Compression, Structural Health Monitoring

I. INTRODUCTION

Wireless sensor network consists of a group of tiny sensor nodes, distributed in a wide geographic area, forming an ad-hoc network, collecting and conveying information regarding the area under surveillance. The data collected by these sensor nodes is aggregated and analyzed at a more capable node called as sink or gateway or base station. Wireless sensor network combines simple wireless communication, minimal computation facilities and some sort of sensing of the physical environment and this leads to a new paradigm of networks that can be deeply embedded in our physical environment, fueled by the low cost and wireless communication facilities.
The rest of the paper is organized as follows. Section 2 reviews background and related works. The proposed architecture is presented in section 3. Section 4 describes algorithmic analysis. Section 5 shows the performance evaluation, and finally fiction 6 concludes the paper.

II. BACKGROUND AND RELATED WORK

Reliable, data compression for wireless sensor networks is an ongoing area of research [1]. Recently several reliability protocols have been proposed. WISDEN which is a reliable data transport using a hybrid of end-to-end and hop-by-hop recovery, and low-overhead data time-stamping that does not require global clock synchronization in which data compression is been made using run length encoding This paper also study the applicability of wavelet-based compression techniques to overcome the bandwidth limitations imposed by low-power wireless radios. The architecture of Wisden was simple, a base station centrally collecting data) its design was a bit more challenging than that of other sensor networks built till date. Structural response data is generated at higher data rates than most sensing applications. Furthermore, this application required loss intolerant data transmission, and time synchronization of readings from different sensors. The relatively low radio bandwidths, the high packet loss rates observed in many environments, and the resource constraints of existing sensor platforms added significant challenges to this system [2]. Wisden used a vibration card, especially designed for structural applications. In addition to describing this card, the description of Wisden focused on its three novel software components:

Reliable Data Transport Wisden used existing topology management techniques to construct a routing tree but implemented a hybrid error recovery scheme that recovers packet losses both hop-by-hop and end-to-end.

Compression Wisden used a simple run-length encoding scheme to suppress periods of inactivity in structural response, but it also evaluated the feasibility of wavelet compression techniques to reduce Wisden’s data rate requirements and to improve latency.

Data Synchronization Wisden also implemented a data synchronization scheme that requires little overhead and avoids the need to synchronize clocks network-wide [2]. An accurate data acquisition system, high-frequency sampling with low jitter and time synchronized sampling were not provided in Wisden [1]. In [1] A Wireless Sensor Network (WSN) for Structural Health Monitoring (SHM) was designed, implemented, deployed and tested on the 4200ft long main span and the south tower of the Golden Gate Bridge (GGB). Ambient structural vibrations were reliably measured at a low cost and without interfering with the operation of the bridge. In the GGB deployment, 64 nodes were distributed over the main span and the tower, collecting ambient vibrations synchronously at 1 kHz rate, with less than 10μs jitter, and with an accuracy of 30μG. The sampled data was collected reliably over a 46-hop network, with a bandwidth of 441B/s at the 46th hop. The collected data agrees with theoretical models and previous studies of the bridge. The deployment is the largest WSN for SHM [1]. In this work a small packet size was a bottleneck for network data transmission bandwidth but increasing packet size was not a good solution for the Mica motes due to the limited amount of available RAM; a limitation resulting from an unshared buffer pool. In [3] WSN for structural health monitoring in which Huffman coding technique was used. In this work, sensor node was collecting data, and also processing the data package. They implemented this algorithm into the sensor node to reduce the packet size to be transmitted. In this work, an on-site WSN-based structure health monitoring of Chung-Sha Bridge (Taiwan) was implemented. The implemented WSN monitoring system successfully achieved the frequency analysis of the bridge structure by monitoring with 128Hz sampling rate from end nodes. A local-data processing node with ARM Cortex M3 processor was developed. Based on this local-data-processing node, Huffman compression algorithm was implemented and examined. The experimental results showed that the wireless transmission payload was reduced by 60% and node number of the implemented network could be increased by 3 times [3].

Both these two works are concerned with the performance. Wireless sensor networks in Structural Health Monitoring based on ZigBee technology and TPSN (Timing-sync Protocol for Sensor Network), ZigBee was used because it is the most popular low-cost, low-power wireless mesh networking standard available. It is Suitable for complex networks with large spatial extension, multi-hop networks and proprietary metering and automation solutions. But one of it drawback is that the interoperation between Zigbee devices and IP based devices [4]. So, our approach is to use WSN in which 6LOWPAN is used as an adaptation layer along with Payload compression using arithmetic compression. “6LoWPAN (Montenegro et al., 2007)” refers to “IPv6 over Low-Power Wireless Personal Area Network”. It defines an adaptation layer which allows transportation of IPv6 packets over IEEE 802.15.4 links. 6LoWPAN reduces the IPv6 packets to fit within the MTU (127 bytes) of IEEE 802.15.4 frames. To do that 6LoWPAN uses header compression and fragmentation and reassembly schemes [18-20].

III. PROPOSED ARCHITECTURE

Sensor Network is made up of many sensor nodes. These nodes are tiny in size. They have limited power supply, memory, and processing capability. Each sensor consists of, Transceiver, Power supply, Processor,
Memory and Sensors. The overall architecture of our sensor network is shown in the figure 1.

![Wireless Sensor Network](image)

Figure 1: Architecture of a Wireless Sensor Network

The sensor nodes are connected to one another in mesh topology fashion. Each node compresses the data using the arithmetic compression algorithm, and forward it to the sink node. The compression of data in each node i.e. local processing of data compresses the data further and thus saves power as well as increases the capacity of the nodes in the network. The target node, passes the compressed data to the sink node, and a further compression on that received data is being performed again, and then, sent to the user’s terminal. So, the sink node has more computation power because of the reduced amount of transfer it has to make.

During the implementation the sensors node are given id in order for us to uniquely identify them so that we can know which node is down or up. This id allocation allows us to specify the sink node, which is the node to which all the other nodes has to send the compressed data to. This node is usually given the id 500 because, the number 500 represents the root node, and any number can be assigned to any other node as their id. Doing so will then help the other node to locate the sink node and pass the data to it for further processing.

IV. ALGORITHMIC ANALYSES

Arithmetic algorithm

In this algorithm, the symbols are not replaced by some code words. Instead, the symbol in the data are being assigned, values, based on some mathematical model; the Arithmetic algorithm can treat the whole symbols in a list, or, in a data message to be transmitted as one unit. It does not use a discrete number of bits or some frequency distribution for the data’s symbols [6], instead each symbol is assigned an interval starting with the interval [0….1); So, at the beginning the probabilities of occurrence of a set of symbols together with the cumulative probabilities are taken into consideration; these cumulative probabilities are used for encoding and decoding. Figure 2 gives some pictorial representation in this regard.

The Encoding Process: The first step is to calculate the cumulative probabilities and then make ranges based on the obtained results. When we read a character, its range is considered as the new cumulative range on which our encoding will be done, that range is so divided into the sub parts, according to the probabilities of occurrence of the character being encoded; and the next symbol is read, and this process of sub part formation is repeated again and this goes on as long as we have a character to encode in our source data. Once the end of our source data is found, we take a fraction of our sub part range formed. Therefore, using a fraction taken from our range we can represent our entire source data into binary form. Let’s consider the following example [7]. From the encoded interval [0.6607, 0.66303]. A sequence of bits is assigned to a number that is located in this range.

During this encoding process two values are frequently calculated: The Upper bound and the Lower bound, and from those values are generated the probability distribution of the symbol to be encoded, they are obtained from the below formula:

- **Lower Bound:**

\[ L = \sum_{i=1}^{n} n^{n-i} C_i \prod_{k=1}^{i-1} f_k \]

- **Upper Bound:**

\[ U = L + \prod_{k=1}^{n} f_k \]

We can summarize the encoding algorithm using the following pseudo-code:[22]

Get symbol.

Initialize \( l \) and \( u \).

while (MSB of \( u \) and \( l \) are both equal to 0 or 1)

if (MSB of \( u \) and \( l \) are both equal to 1)

\( l \leftarrow l + \left[ \frac{(u - l + 1) \times \text{Cum_count}(x - 1)}{\text{TotalCount}} \right] \)

\( u \leftarrow u + \left[ \frac{(u - l + 1) \times \text{Cum_count}(x)}{\text{TotalCount}} \right] - 1 \)

while (Scale3>0)

{ }

Send b

Shift l to the left by 1 bit and shift 0 into LSB

Shift u to the left by 1 bit and shift 1 into LSB

while (Scale3>0)

{ }

Copyright © 2018. IJEMR. All Rights Reserved.
if \( E_3 \) condition holds

\begin{align*}
\text{Shift } l \text{ to the left by 1 bit and shift 0 into LSB} \\
\text{Shift } u \text{ to the left by 1 bit and shift 1 into LSB} \\
\text{Complement (new) MSB of } l \text{ and } u \\
\text{Increment Scale3}
\end{align*}

Image representing the full encoding process:

![Image of the encoding process](image)

Figure 2: Encoding process

The Decoding Process: Once the compress data reach the other end, it needs to be decompressed and this is being done by following some mathematical model, and then, applying the inverse process of the encoding. But to do that, the receiver needs to know the number of symbols that were sent as well as the probability \( l \) frequency distribution.

We can summarize the decoding process with the following algorithm: [22]

1. Initialize \( l \) and \( u \).
2. Read the first \( m \) bits of the received bit stream into tag \( t \).
3. \( k = 0 \)

\[
\left( \frac{(t - l + 1) \times \text{Total Count} - 1}{u - l + 1} \right) \geq \text{Cum_Count}(k)
\]

\( k \leftarrow k + 1 \)

4. \( l \leftarrow l + \left( \frac{(u - l + 1) \times \text{Cum_count}(x - 1)}{\text{Total Count}} \right) \)
While (MSB of \( u \) and \( l \) are both equal to \( b \) or \( E_3 \) condition holds)

\[
\text{If(MSB of } u \text{ and } l \text{ are both equal to } b)\]

shift \( l \) to the left by 1bit and shift 0 into LSB
shift \( u \) to the left by 1bit and shift 1 into LSB
shift \( u \) to the left by 1bit and shift 1 into LSB
shift \( t \) to the left by 1 bit and read next bit from received bitstream into LSB
Complement (new) MSB of \( l \), \( u \), and \( t \).

\[
T\text{otal}\_C\text{ount} - 1
\]

\[
u \leftarrow l + \left\lfloor \frac{(u - l + 1) \cdot Cum\_count(x)}{Total\_Count} \right\rfloor - 1
\]

V. PERFORMANCE EVALUATION

In performance evaluation, we have used the Huffman Coding as a reference as this algorithm was used in SHM [3]. The performance is measured in terms of Compression ratio, data size, memory consumption, and compressed data size.

In the figure 3, we compared the compression ratio (CR) of the Huffman coding with that of the arithmetic coding. It is being observed that, with the increasing in the size of the data, the compression ratio of arithmetic coding gets better than that of Huffman coding, and the compression ratio of arithmetic is almost more than the double of that of the Huffman coding as the data get higher. On the other hand, Huffman Coding does not give any better compression with the increase of size. We would like to mention that the data size in SHM is large and varied. Which is suitable for our algorithm.

![Figure 3: Algorithm Comparison (CR vs. Size)](image)

The following Table 1 show the data’s result used to draw the above graph.

<table>
<thead>
<tr>
<th>File Size</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>4.65</td>
</tr>
<tr>
<td>256</td>
<td>5.4</td>
</tr>
<tr>
<td>512</td>
<td>6.55</td>
</tr>
<tr>
<td>1024</td>
<td>7.73</td>
</tr>
<tr>
<td>2048</td>
<td>12.02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File Size</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>4.38</td>
</tr>
<tr>
<td>256</td>
<td>4.78</td>
</tr>
<tr>
<td>512</td>
<td>5.27</td>
</tr>
<tr>
<td>1024</td>
<td>5.64</td>
</tr>
<tr>
<td>2048</td>
<td>6.37</td>
</tr>
</tbody>
</table>

The above graph (Figure 4), shows, the amount of data capable to be compressed given the same memory boundary. It is seen from the graph that, the Arithmetic...
coding, can compress the whole data (100%) provided to it at a specific time, while the Huffman coding, can only compress 60% of the data provided to it, as well the RLE can only compressed 75 % of the data provided, then the node running these algorithms will run out of memory at some time. The compression ability of arithmetic algorithm when provided with the same memory block is better than that of the Huffman and RLE.

![Data comparison capable to be compressed by both algorithm](image)

**Figure 4: Data comparison capable to be compressed by both algorithm**

**VI. CONCLUSION**

Wireless sensor network is getting popular day by day. Its mobility and capability of real time communication has led many applications use this technology. Many developers and researchers are providing their support to enhance this sector. Many new protocols, architectures are being proposed in this purpose. Among many application areas, Structural Health Monitoring is very important. SHM has many advantages and it can be used to save both money and lives of the people of the country. By monitoring the important structures and by renovating or maintaining these structures when necessary can save a lot of money as well as keep the structures fit. Besides it can save many lives which are lost in different natural disasters.

Using WSN technology in SHM can save lives and make the system less costly and affordable. The structure’s health can be easily monitored, as well as structures in the remote areas can be monitored and can be easily maintained at a very low cost which will save lives in case of emergencies.

One of the main disadvantages of the WSN node are, the power constraint (because small size implies small battery) and limited memory. It is useful to mention that WSN node can be deployed in extreme environmental conditions and no human can go over to change a damaged node. So, we need to make use of the node as efficiently and as long as possible. So, the data should be transferred efficiently between sensors.

The compression algorithm and technique we proposed will increase the life time of the sensors as less data will be sent. Besides, the local processing will increase the life time as well as the number of sensors deployed in the network. We have chosen the arithmetic algorithm over other type of algorithm because of the advantages that it was offering on the compression ratio plan as well as the memory required for the processing of data. By observing the results generated by all the algorithms in our work, we can easily show that arithmetic algorithm is far more better that the other algorithm.

**ACKNOWLEDGMENTS**

We would like to thank the anonymous reviewers for their very useful comments that helped us enrich the quality and presentation of the paper a lot. This work was supported by a grant from the Islamic University of Madinah in 2017-2018.

**REFERENCES**


