Mind the SmartGap: A Buffer Management Algorithm For Delay Tolerant Wireless Sensor Networks

Pehr Söderman[1], Karl-Johan Grinnemo[2], Markus Hidell[1], and Peter Sjödin[1]

[1] NSLab, School of ICT
KTH Royal Institute of Technology, Stockholm, Sweden
pehrs@kth.se, mahidell@kth.se and psj@kth.se
[2] Department of Computer Science
Karlstad University, Karlstad, Sweden
karlgrin@kau.se

Abstract. Limited memory capacity is one of the major constraints in Delay Tolerant Wireless Sensor Networks. Efficient management of the memory is critical to the performance of the network. This paper proposes a novel buffer management algorithm, SmartGap, a Quality of Information (QoI) targeted buffer management algorithm. That is, in a wireless sensor network that continuously measures a parameter which changes over time, such as temperature, the value of a single packet is governed by an estimation of its contribution to the recreation of the original signal. Attractive features of SmartGap include a low computational complexity and a simplified reconstruction of the original signal. An analysis and simulations in which the performance of SmartGap is compared with the performance of several commonly used buffer management algorithms in wireless sensor networks are provided in the paper. The simulations suggest that SmartGap indeed provides significantly improved QoI compared the other evaluated algorithms.

1 Introduction

Delay Tolerant Wireless Sensor Networks (DT-WSNs) are networks that combine concepts from delay-tolerant networking (DTN) and wireless sensor networks (WSN). In this work, we consider networks of constrained devices which sense their environment, and communicate sensor data (such as temperature and humidity) through wireless links. Sensor data is forwarded, possibly via multiple hops, to a sink node which gathers and stores the data for further processing.

Nodes in a DT-WSN can be stationary or moving; they can be location-aware or not; and, they can be homogeneous or heterogeneous. Connectivity between the nodes may be scheduled, intermittent or opportunistic. As an example, consider a WSN deployed in a rural area where there is no communication infrastructure. With the help of DTN data mules, sensor data is transported from the WSN to a central location where the data is stored and made available for further analysis. Figure 1 shows an example of such a network.

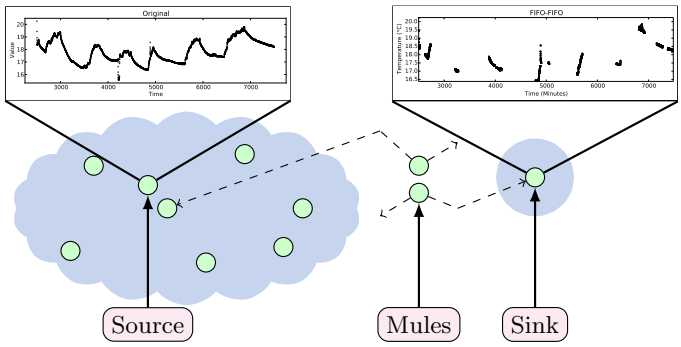

Fig. 1: An example of a DT-WSN comprising a number of spatially isolated wireless sensors. The source sensor nodes measure the environment by collecting measurement samples, which are transported by mobile data mules to a sink node. The sink attempts to reproduce the measurement from the samples. As the capacity of the network is limited, some samples are lost in transit, and the quality of the reproduced measurement is affected.

Although, there is a significant amount of work on both WSNs and DTNs, there is less work on the combination of the two network types. Previous work exists on DT-WSN systems in laboratory settings, such as the Wind Tunnel Monitoring system by Lou et al. [1] and the Data Elevator testbed by Pottner et al. [2]. Furthermore, Zennaro reports experiences from a field trial with a network to monitor water quality in the Blantyre district of Malawi [3].

In this work, we consider buffer management algorithms for DT-WSNs. The typically intermittent-delivery, long-latency and low-bandwidth characteristics of these networks, enforce a store-and-forward behaviour on the nodes, and make buffer management pivotal to uphold a high network performance. Yet, there is relatively little previous work on buffer management in DT-WSNs. To illustrate the importance of buffer management in DT-WSNs, consider a straightforward approach that uses a head-drop buffer policy. In this policy, when a buffer is full, the first packet in the buffer, i.e. the oldest one, is dropped. Assume that a node collects one sample per minute and stores the sample in the buffer. Further assume that, for some reason, the node is unable to communicate for a period of one hour. Then only the last 10 packets will remain in the buffer, and the remaining 50 packets will be dropped. In other words, newly arrived packets take priority over already queued packets. As a result, there could be long periods of measurements during which the sink node gets no measurement data at all.

Previous work on buffer management in DT-WSNs can be divided into two broad categories: work that considers packets to be *transparent* and work that considers packets *opaque*. If packets are considered transparent, buffer management has the capability to parse the data being transported and make decisions based on packet content. In contrast, if packets are opaque, buffer management

decisions are only based on the information available in the header of the packets, and not on packet content.

There are some clear advantages with algorithms that treat packets as transparent. These algorithms can for example use compression of data or prioritise data based on content. This approach is typically based on the notion of *Quality of Information* (QoI) [4], i.e. a measure of how well the service provided by the network meets the applications' needs. By definition, QoI is application-specific, so there is no universally agreed upon method to measure or quantify it. Examples of such algorithms, all of which target QoI, include the work of Liu et al. [5], Humber and Ngai [6], and Alippi et al. [7]. In the algorithm proposed by Liu et al., the buffered data is replaced with a linear approximation when the buffer is full. Humber and Ngai estimates the importance of a packet when it is created based on how much the sampled value stored in the packet differs from previous values, and then assigns a priority class to the packet on the basis of this estimate. Lastly, Alippi et al. focus on energy saving, and dynamically adjust the sample rate based on the frequency of the property being sampled.

Algorithms that treat packets as opaque do not depend on code to parse the data being transported. Algorithms such as FIFO are easy to understand and implement, but their performance may not be optimal. One attempt to provide improved performance compared to FIFO while still considering packets opaque is the work by Nasser et al. [8] which proposes a Dynamic Multilevel Priority (DMP) packet scheduling scheme in which sensor nodes are organised into a hierarchical structure. Sensor nodes that have the same hop distance from the sink node are considered to be located at the same hierarchical level, and a Time Division Multiplexing Access (TDMA) scheme is used to prioritise packets from different levels. Another example is Lyu et al. [9] which suggests a multi-queue Last In First Out (LIFO) queueing policy. Their main argument is that LIFO works better than FIFO for real-time applications because it achieves a shorter delay in congested situations, especially when packets are limited by a deadline.

We see a clear need for algorithms which considers packets to be opaque: For one thing, we believe that transparent buffer management techniques in which sensor nodes parse the contents of packets are potentially expensive in terms of computational and energy resources. Also, they are not general purpose solutions and therefore inflexible – each sensor node must be equipped with code for parsing the data that flows through the network. In our scenario, the data mules would need to be aware of the format of the data they carry. At the same time, we see a need for buffer management algorithms that give priority to the data samples that are most important in the reconstruction of the original measurement at the sink node, i.e. QoI targeted buffer management algorithms.

In this work, we present *SmartGap* – a QoI-targeted buffer management algorithm which considers packets opaque. SmartGap is a novel buffer management algorithm that tries to maximise the combined value of the packets in the buffer. It accomplishes this by letting the priority of a packet be determined by the *gap* the packet would inflict – if dropped – on a complete series of measurement.

The remainder of the paper is organised as follows. Section 2 presents and explains the design of the SmartGap algorithm. An analysis of SmartGap's main characteristics is given in section 3. Section 4 provides a comprehensive evaluation of SmartGap and compares its performance with three commonly used buffer management algorithms in DT-WSNs. Section 5 discusses the benefits and limitations of SmartGap. The paper concludes in section 6 with a summary of the paper and some remarks on future work.

2 The SmartGap Algorithm

Buffer management schemes can typically be split up into two parts, a *queueing policy* and a *forwarding strategy* [10]. The queueing policy decides which packets in a buffer to discard when the buffer space is exhausted, while the forwarding strategy decides the order in which packets in the buffer should be forwarded.

An insight, which follows from the Nyquist-Shannon sampling theorem is that the quality of a reconstructed signal depends on the sampling frequency. So, to be able to reconstruct the signal as faithfully as possible, we wish to maximise the minimum number of samples in any given time period. In other words, the collected samples should be evenly distributed in time. As each sample is transported inside a packet, it follows that we want to minimise the maximum time gap between any two consecutive packets. Hence, the problem is to design a buffer management algorithm that during periods of congestion distributes packet losses evenly over time. The following section elaborates on the problem by demonstrating how common buffer management algorithms such as FIFO and Random distributes the packets. Next, we provide a detailed description of the SmartGap algorithm.

2.1 FIFO Buffer Management

Let us consider a DT-WSN that contains a source node, a sink node, and a data mule which moves in a random pattern between the source and sink nodes. The data mule collects data from the source node and uploads it to the sink node. Furthermore, assume that the the buffer space and transport capacity of the mule is insufficient to handle the load. Given that the source node employs FIFO (First In, First Out) as both queuing policy and forwarding strategy, the outcome could resemble that shown in Figure 2.

As follows from the figure, the delivered data is very unevenly distributed over time – during some periods, all sampled data is delivered, while there are also long periods with no or few data samples are delivered – something which makes it hard to reconstruct the original series of measurement. This is to be expected, as the FIFO strategy always picks the oldest packet in the buffer for forwarding or discarding.

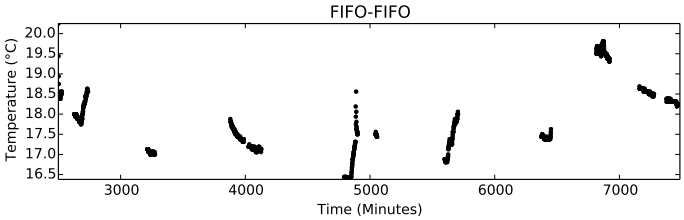

Fig. 2: Delivered data from a simulated DT-WSN (random-waypoint) which employs FIFO queueing policy and forwarding strategy. Note the long gaps in the delivered data.

2.2 Random Buffer Management

A naive attempt to spread out packets more evenly could be to use the Random algorithm, i.e. to randomly discard samples at times when the buffer is full, and to randomly pick the packet to forward next. Figure 3 shows the result of using such a buffer management policy for the same system as used in Figure 2. Compared to FIFO, the Random algorithm spreads out the data more. However, note that data is still clustered since the Random algorithm has a bias towards delivering recent packets. This can be explained by viewing the buffer management problem as an urn problem. Assume that data is constantly added to a buffer (urn), from which packets are randomly removed – either by forwarding or discarding. As this is an iterative process, a packet added early has higher probability to be chosen for removal than a packet added later. When we simulate the algorithm, we can clearly see this effect.

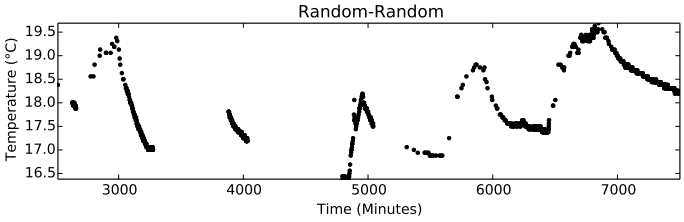

Fig. 3: Delivered data from a simulated DT-WSN (random-waypoint) which employs Random queueing policy and forwarding strategy. There are still gaps in the delivered data, but the gaps are reduced compared to the FIFO case.

2.3 SmartGap

The SmartGap algorithm attempts to further shorten the duration of periods with few delivered packets. SmartGap calculates the gap in time that would

result from removing a specific packet from the buffer, and then gives priority to packets that cover large gaps. SmartGap is based on the notion of *creation time*: the time when a packet, P, was created at the source sensor node, $time(P)$.

Definition 1: Interpacket gap The interpacket gap represents the difference in creation time between two packets. For two packets P_i and P_j the interpacket gap is $\left|time(P_i) - time(P_j)\right|$.

Definition 2: The gap metric The SmartGap algorithm is based on the *gap* metric. For a given packet buffer in a node, sort all packets in the buffer according to creation time. Then the gap metric for a packet P_n is the interpacket gap between the preceding packet, P_{n-1}, and the succeeding packet, P_{n+1}, packet in the buffer:

$$Gap(P_n) = \left|time(P_{n+1}) - time(P_{n-1})\right|.$$

For example, consider a buffer with three packets, P_0, P_1, and P_2, with creation times 1, 3, and 4, respectively. Then we obtain:

$$Gap(P_1) = \left|time(P_2) - time(P_0)\right| = |4 - 1| = 3.$$

The computations of the gap metric for the first and last packet in the buffer are slightly more complex, since these packets do not have both a preceding and a succeeding packet.

Depending on whether SmartGap is used as a forwarding strategy or a queuing policy, these two border cases are handled differently. When used as a queuing policy, SmartGap considers the first and last packets to have an infinite gap metric, and will therefore not discard them. When used as a forwarding strategy, SmartGap estimates the gap metric for the first and last packet as twice the interpacket gap between these packets and the closest packet in the buffer. This is based on an assumption that the packets are evenly distributed. Thus, if P_0 is the first packet and P_N the last packet, we have:

$$\begin{aligned} Gap(P_0) &= 2 \cdot \left|time(P_1) - time(P_0)\right| \\ Gap(P_N) &= 2 \cdot \left|time(P_N) - time(P_{N-1})\right| \end{aligned}$$

2.4 SmartGap Queuing and Forwarding

SmartGap uses the gap metric to prioritise packets. As a queuing policy, SmartGap will discard the packet with the lowest gap metric. In other words, packets in bursts have higher probability of being discarded than single packets. When used as a forwarding strategy, SmartGap will forward the packet with the largest gap metric. This means that sparsly distributed packets are more likely to be forwarded than clustered ones.

Figure 4 illustrates how SmartGap is able to distribute the packets more evenly as compared to both the FIFO and Random buffer management schemes. The nodes still run out of buffer space when the path between the source and the sink node has insufficient capacity, but since SmartGap distributes packets more evenly, trends in the data are clearly visible.

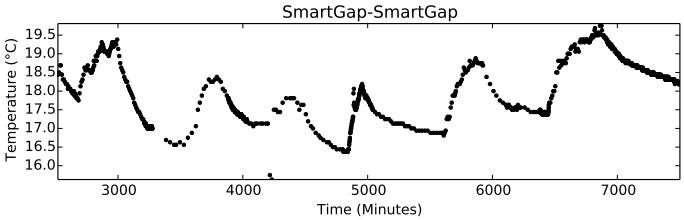

Fig. 4: Delivered data from a simulated DT-WSN (random-waypoint) which employs SmartGap. As follows, SmartGap gives even shorter periods with few or no delivered data as compared to a Random queuing policy.

3 Analysis

This section contains an analysis of the performance of SmartGap. First, an upper bound for the gap metric is established. Next, statistics for the variation of the gap in the examples in section 2 are presented. Finally, the computational complexity of SmartGap is discussed.

3.1 Upper Bound for the Gap

SmartGap determines which packets to discard and which to forward in the buffer. However, since the ultimate goal of SmartGap is to minimise the largest interpacket gap among all packets in the network, it is interesting to examine the effect on the largest gap of discarding a packet from the buffer. For this purpose, we use the term *maximum gap* of a set of packets to denote the largest interpacket gap between two consecutive packets in the sequence obtained by ordering the packets according to their creation time.

Theorem 1: Assume that $P_0 \dots P_N$ are packets distributed among a number of nodes communicating reliably (i.e. without packet loss or with retransmissions on each hop). Each packet has an associated gap metric, calculated according to Definition 2. Discarding packet P_n, where $0 < n < N$, from a buffer will create an interpacket gap not larger than $Gap(P_n)$.

Proof: Assume without loss of generality that $P_0 \dots P_N$ are ordered in a sequence according to creation time, so that P_0 is the youngest packet and P_N the oldest one, and that packet creation times are distinct – two different packets in the sequence do not have the same creation time. Let P_{n-1} denote the packet immediately before P_n in the sequence, and P_{n+1} the packet immediately after. Hence, discarding P_n will create an interpacket gap in the sequence of size $G = \left| time(P_{n+1}) - time(P_{n-1}) \right|$. We want to show that $G \leq Gap(P_n)$.

Assume that the packets in the buffer where P_n is queued are ordered according to creation time (again, without loss of generality). If P_{n-1} and P_{n+1} are both in the same buffer as P_n, then P_{n-1} must be immediately before P_n, and P_{n+1} immediately after P_n. Hence, when discarding P_n, the size of the newly created interpacket gap (G) is equal to $Gap(P_n)$, the gap metric for P_n in the

buffer (by Definition 2). Otherwise, if not both P_{n-1} and P_{n+1} are in the same buffer as P_n, it means that the packet immediately before P_n in the buffer is younger than P_{n-1}, and/or the packet immediately after P_n in the buffer is older than P_{n+1}. From this follows that $Gap(P_n)$, the gap metric for P_n in the buffer, is larger than G, the interpacket gap created by removing P_n. Hence, $G \leq Gap(P_n)$. □

Theorem 2: The maximum gap created by discarding a packet in a buffer according to the SmartGap strategy is $2\frac{T}{N-1}$ where T is the interpacket gap between the oldest and the newest packet in the network, and $N \geq 3$ is the number of packets in the buffer.

Proof: Assume that the network has a single node, and the packets stored at a sensor node are evenly distributed in time between 0 and T. Also assume that packets are created at time 0 and at time T. Then the interpacket gap between any two consecutive packets is $\frac{T}{N-1}$, and the maximum interpacket gap in the buffer after discarding a packet is $2\frac{T}{N-1}$. If the packets are not evenly distributed between 0 and T there will be a packet P_n such that $Gap(P_n) \leq 2\frac{T}{N-1}$. If not, all consecutive pairs of interpacket gaps have to be larger than average, which is clearly impossible. If there are multiple nodes in the network, Theorem 1 tells us that the maximum gap will not grow larger due to the packets stored at the other nodes. □

3.2 Variation of the Gap

The design goal of SmartGap is to minimise the maximum interpacket gap. In the simulation in section 2, three different algorithms are evaluated using a random waypoint mobility model and a buffer size of 110 packets. We repeat the simulation 30 times and calculate the confidence intervals for the mean, max and standard deviation of the interpacket gap. The result is presented in Table 1. As expected, there is no significant difference in the mean between the three algorithms. However, there is indeed a significant difference in both the maximum value and the variance: SmartGap provides a significant reduction of both maximum and standard deviation over FIFO as well as Random. The reason to this is the tendency of the FIFO and Random algorithms to discard consecutive packets.

Algorithm	Mean	Max	Standard Deviation
FIFO	3.99 (3.64, 4.33)	800.43 (713.75, 887.11)	38.66 (34.98, 42.34)
Random	4.15 (3.79, 4.51)	388.16 (313.85, 462.47)	17.27 (14.66, 19.88)
SmartGap	4.17 (3.81, 4.53)	18.50 (15.94, 21.05)	3.86 (3.36, 4.35)

Table 1: Interpacket gap in the three examples, 30 repetitions of the simulation, with 95% confidence intervalls presented. Note that SmartGap provides a lower maximum and standard deviation of the interpacket gap, as intended.

3.3 Computational Complexity

SmartGap has a low computational complexity. By memorising the gap metric for a packet, and keeping an ordered set of references to the packets in the buffer, the gap metric needs to be calculated at most three times for each packet received, and two times for each packet transmitted or discarded. So SmartGap has linear complexity, $\mathcal{O}(n)$, where n is the number of packets received by the node. Calculating the gap metric requires extracting the creation time from the header of the packets and performing basic arithmetic operations.

4 Evaluation

In the previous section, we established an upper bound for the size of the maximum interpacket gap. We also compared SmartGap with other buffer management algorithms and found that SmartGap provides a more even distribution of packets. This section provides a more comprehensive evaluation of SmartGap. Particularly, the QoI provided by SmartGap is compared with a select of other well-known buffer management algorithms, namely:

- First In First Out (FIFO),
- Random choice (Random),
- A priority queue based on Humber and Ngai [6] (Humber-Ngai).

All studied buffer management algorithms, including SmartGap, may be used both as queueing policy and forwarding strategy. FIFO and Random are straightforward algorithms that consider packets opaque. Humber-Ngai is a sliding-window algorithm which considers packets transparent. The algorithm calculates a sliding window over the packets as they are created, and if a new packet carries a value that differs more than a certain amount from the values in previous packets, the new packet is given a high priority. Humber-Ngai's algorithm also compresses data by removing samples when there are no significant changes.

Apart from the studied buffer management algorithms, we have also simulated Oldest First, Youngest First, and First In Last Out, however, since neither one of them differ much from FIFO in terms of performance, they are omitted from our evaluation. We have also considered the algorithms proposed by Liu et al. [5] and Alippi et al. [7], but found these algorithms to be less suited for DTN data mules. Instead, they are primarily intended for limiting the data rate on source nodes. We consider this approach complementary to the buffer management algorithms evaluated here.

In the following, to be able to differentiate between queuing policy and forwarding strategy, a particular buffer management scheme is denoted: *queuing policy-forwarding strategy*. For example, "FIFO-Random" denotes the buffering scheme that employs a FIFO queueing policy and a Random forwarding strategy.

4.1 Simulation Setup and Datasets

Therefore a custom-built simulation system has been developed, focusing on buffer management in DT-WSN. The simulation system has been built in Python, on top of the discrete event simulation package, SimPy [11]. In our simulation system, a DT-WSN is modelled as a set of nodes with pre-set buffer sizes connected with links. Packets emanate from source nodes that model wireless sensor devices. They are forwarded toward sink nodes, i.e. controllers, along network paths comprising links and intermediate nodes. The intermediate nodes model both fixed data aggregation nodes and mobile data mules. A separate mobility model is used to pre-calculate the meetings between data mules and their neighbouring nodes. Routing is done using the probabilistic routing protocol, PRoPHET [12], a routing protocol introduced by Lindgren et al. The rationale for using ProPHET is first and foremost that it is one of a few routing protocols for DT-WSNs that has been standardised and it is regularly used as a baseline when evaluating routing protocols, e.g. by Case et al. [13]. To allow the routing to stabilise, the simulation runs for 2500 simulated minutes, i.e. around 42 hours, before the actual experiment starts.

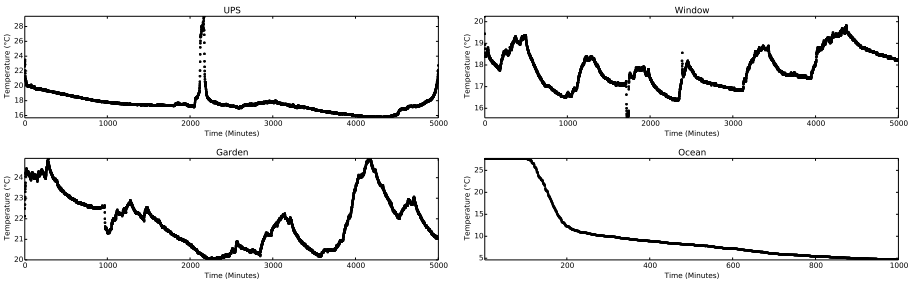

Fig. 5: The four temperature datasets used in the simulations. The data sets are available online [14–17]

Since the data being transported influences the outcome of the experiments, our simulation system is trace driven and runs from temperature data sets from real-world WSNs. Particularly, the evaluation is made against four different datasets, “Ocean”, “UPS”, “Windows”, and “Garden”, which are depicted in Figure 5. The datasets are selected to represent a spectrum of different types of WSN traffic. The “Ocean” dataset is based on a series of deep sea CDT (Conductivity, Temperature and Depth) measurements from the National Data Buoy Center (NDBC) [18], and is available online [17]. A reason for including this dataset in our evaluation is to enable comparisons between SmartGap and other buffer management algorithms beyond the three already included in the evaluation. Already, the “Ocean” dataset was used by Lou et al. [19] in a validation of their scheme for compressive sampling. The remaining three datasets, “UPS”,

"Windows" and "Garden", are all captured from a WSN deployed in and around a property in Uppsala, Sweden, and are available online [14–16].

4.2 Simulation Results

To evaluate the performance of SmartGap in terms of QoI, we simulate a DT-WSN of size 1000×1000 meters. The DT-WSN comprises ten nodes: one wireless temperature sensor node, one controller node and eight data mules. The mules move according to the widely used random-waypoint mobility model [20]. The speed of the mules is 5 metres/minute, and they have a range of 50 metres. We test multiple combinations of buffer sizes and buffer management algorithms. The QoI experienced in a simulation run is estimated using the mean absolute error or MAE between the original (f) dataset and the one being reconstructed at the controller node (g) using a cubic interpolation:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |f_i - g_i|$$

A low MAE reflects a high QoI. This method of estimating the QoI is based on the method used by Humber and Ngai [6]. We expect this measure of QoI to be correlated with the size and the distribution of the interpacket gaps. We repeat the simulation 30 times, re-seeding the random waypoint simulation each time. In this way, we vary the distribution of meetings, and this is what causes the differences in the outcome. We present mean results and confidence intervals.

Figure 6 shows the outcome of the simulations with a varying queueing policy and a fixed forwarding strategy, FIFO. In other words, the figure illustrates how the buffer algorithms perform as queueing policies with increasing buffer sizes. The smallest buffer we simulate is 10 packets. Buffers smaller than 10 packets leave little room for effective buffer management, and our experience is that for buffers of that size, the choice of algorithm has little impact on the outcome. The largest buffer we simulate is 1500 packets, which is a buffer large enough to acommodate all data without discarding any packets, and hence there is no difference between the buffer algorithms. We expect a smaller buffer to give a larger error, and thus a lower a QoI, and a larger buffer to give a smaller error, and thus a higher QoI.

Our first observation is that Humber and Ngai's algorithm [6] almost perform the same as FIFO. The MannWhitney U test (p=0.05) accepts the alternative hypothesis that the two algorithms performs differently for buffer sizes larger than 800 packets on the Window and Garden Data sets, but otherwise rejects it. It appears that the Humber and Ngai algorithm is sensitive to the parameterisation, which needs match the characteristics (primarily variance and autocovariance) of the data. We tried to find a reasonable configuration of the algorithm experimentally, but of the settings we tried significantly outperformed FIFO for all data sets and buffer sizes.

Next, we observe the scale of the MAE. The resolution of the temperature sensors is about 10^{-1}, and errors much smaller than this would for any practical

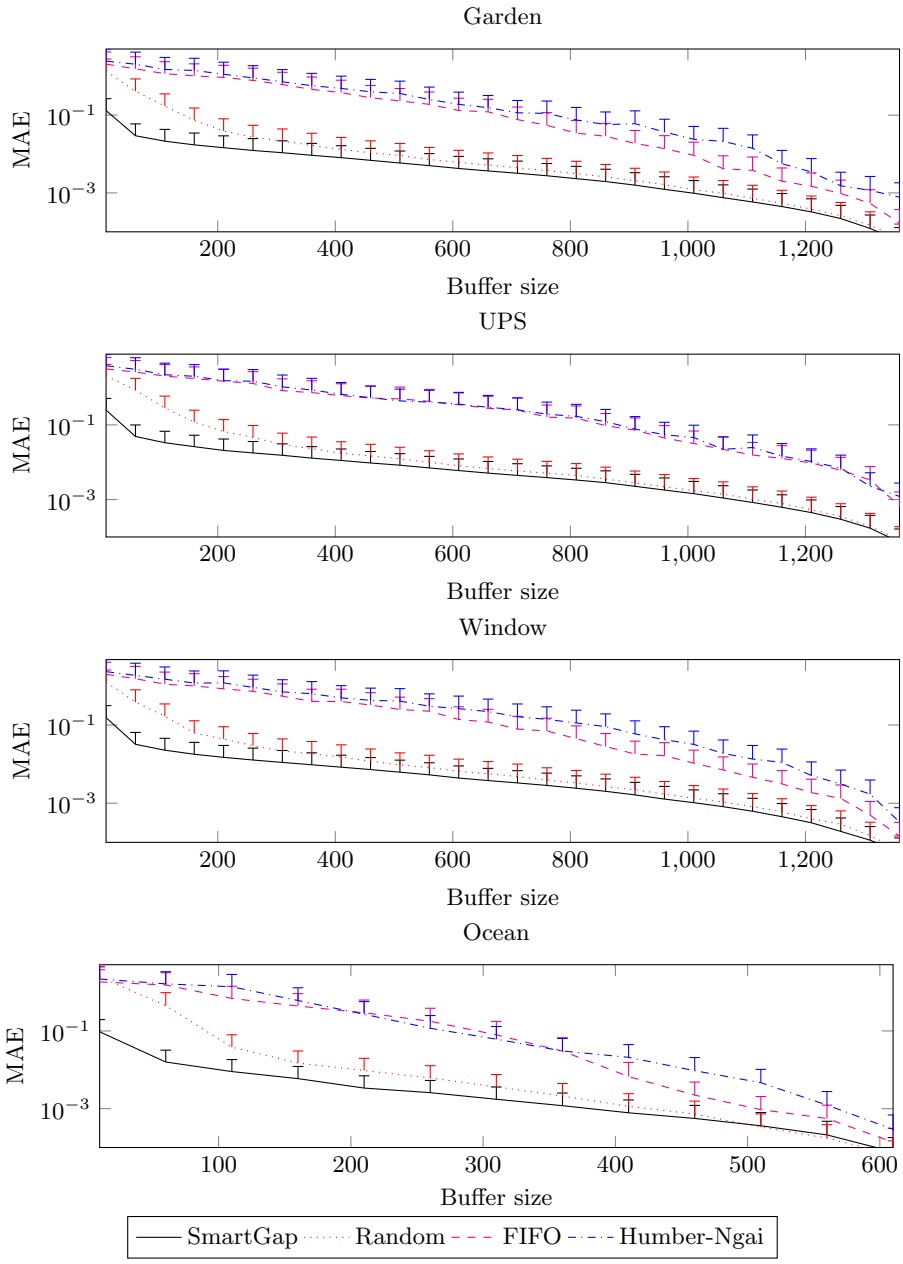

Fig. 6: Results from simulations using the FIFO forwarding strategy, with 95% upper confidence intervals presented. The lower confidence intervalls excluded for legibility. Humber-Ngai gives a small but significant reduction in MAE in the Ocean and Garden data sets, especially for larger buffer sizes. Random gives a significant reduction in MAE over both FIFO and Humber-Ngai, and SmartGap further reduces the MAE, especially for small buffer sizes.

application be dominated by the resolution of the sensors. In addition, we note that the confidence intervals in the outcome for the FIFO/Humber-Ngai strategies are large compared to SmartGap. The reason to this is that the selected mobility model has a larger impact on both FIFO and Humber-Ngai than on SmartGap. Since we re-seed the mobility model for each run, the distribution of the meeting times will vary, and, in comparison with SmartCap, this appears to have greater impact on the outcome of the FIFO and Humber-Ngai simulations.

Another observation is that for all buffer sizes, SmartGap outperforms Random which in turn outperforms FIFO. In section 3.2 this trend is shown for a single buffer size when studying the variance and maximum interpacket gap. In the extended simulations reported here, we note that the observation holds true across the range of buffer sizes simulated, regardless of which data set is used. Thus, SmartGap fullfills the design goal of providing a significant improvement in QoI.

It is interesting to note the scale of the buffer sizes. To obtain an MAE of 10^{-1}, SmartGap requires a buffer of about 10 - 15 packets. To obtain a similar result with Random the buffer must be increased to 150 - 200 packets, while the FIFO and Humber-Ngai discard policies require buffer sizes of 200 - 800 packets. Hence, SmartGap creates the potential for a design choice. An application could take advantage of this improved QoI, but could also maintain the QoI while freeing up resources. This could for example be used to reduce device cost or improve device lifetime through energy saving.

SmartGap can also be used as a forwarding strategy. As previously mentioned, this will give priority to packets with a high gap metric when forwarding packets. We expect the forwarding strategy to have a smaller impact on the experiment outcome. The forwarding strategy decides the order in which packets are delivered towards the destination, and our QoI does not depend on delivery order. Instead, it depends on whether a packet arrives to the destination or not. Repeating the experiment, using SmartGap as the forwarding strategy instead of FIFO did not give a significant change in the experiment outcome.

To summarise, these results indicate that SmartGap could provide a significant improvement in QoI when used as a queuing policy, and that it provides QoI at least on par with other algorithms when used as a forwarding strategy.

5 Discussion

After presenting the simulation results we follow with a short discussion of the motivation for developing SmartGap and the limitations of this buffer management algorithm.

5.1 The raison d'être of SmartGap

Although it might seem that SmartGap and similar buffer management solutions are superfluous, and could be avoided through proper network provisioning, we argue otherwise. For example, one seemingly straightforward way to provide a

high QoI would be to dimension buffers so that the likelihood of running out of buffer space is minimal. However, predicting the required buffer space in a DT-WSN is extremely difficult, and the available hardware may not be capable of providing the required buffer space. In a survey by Hempstead et al. [21], the available storage space of the examined wireless sensors ranged from under 1 KB up to 138 KB. Moreover, even if the sampling is dimensioned for the lowest available bandwidth and buffer space, the sensor device will be unable to opportunistically take advantage of occasional extra available bandwidth, and buffer over-provisioning may in practice be a too costly alternative.

A seemingly attractive replacement to buffer management, would be to use compression or aggregation to reduce the amount of data being transferred between nodes in the wireless sensor network. For example, Vuran et al. [22] propose the use of temporal and spatial correlation in the data, and Al-Karaki et al. [23] present a number of suitable algorithms for data aggregation in WSNs. However, these and other approaches require data mules to be able to parse the data being collected, and perform potentially expensive operations, something which make them less suitable as general solutions. Still, it should be noted that compression and aggregation are indeed attractive solutions in specialised DT-WSNs, not least since they enable an explicit tradeoff between communication and computation resources, and thus could open up for significant energy savings.

5.2 Limitations of SmartGap

SmartGap relies on a few underlying assumptions, and if these do not hold true, SmartGap is unlikely to perform well. In particular, SmartGap assumes that the packets being transported in the DT-WSN are self-contained and correlated. In other words, the application must be able to interpret one packet alone, and there must be a correlation between packets to exploit. Normally, for environment sensing applications, sampling is done at a higher rate than the frequency of the underlying physical property being measured, as can be expected from Nyquist-Shannon. However, if the frequency of the underlying signal is too high, the correlation becomes weak. In that case, SmartGap would not provide any clear advantage. Consequently, in such scenarios it may be better to obtain a group of samples, collected very closely in time, and extrapolate the rest of the signal.

Self-containment is primarily a problem if samples do not fit into a single packet. This can for example happen if the sample is an image whose size is larger than the Maximum Transmission Unit, i.e. each sample will occupy multiple packets. If one packet is discarded, the rest of the packets that belong to the same sample become more or less worthless, and in this situation the strategy used by SmartGap is likely to cause more harm than good.

6 Conclusion

In this paper, we have proposed a novel buffer management algorithm, *SmartGap*, for Delay-Tolerant WSNs, i.e. WSNs with occasional connectivity between mo-

bile and stationary wireless sensor nodes. SmartGap estimates the Quality of Information, or QoI, of packets and gives priority to packets that contribute more to QoI. SmartGap determines the priority of a sample packet by the error or *gap* that a loss of this packet would impose on an overall series of measurements. An analysis and a simulation-based evaluation of SmartGap have been conducted, in which the algorithm is compared with a select of buffer management algorithms. According to the evaluation, SmartGap provides significant improvements in QoI compared to the alternative algorithms when used as a queueing policy, and performs at least as good as these algorithms when used as a forwarding strategy. The largest improvements are obtained in situations where the buffer space is small, and large amounts of data are discarded.

SmartGap is primarily intended for networks such as the water monitoring system in Malawi [3], where connectivity is opportunistic, memory space is limited, and the fidelity with which the the original signal can be recreated is crucial for the quality of the results.

The evaluation has been made using simulations with a random waypoint mobility model. A number of alternative mobility models have been developed [24], often in association with the development of routing protocols. One direction in which we wish to continue this work, is to test the performance under alternative mobility models, including replicating routing, and explore the interaction between the mobility model, the routing algorithm and the buffer management algorithm. As part of this work, we wish to study the fairness characteristics of SmartGap. Finally, encouraged by our promising simulation results, we would like to deploy SmartGap in a real world setting.

References

1. C.-J. Luo, M.-T. Zhou, and Z.-Y. Cao, "Disruption-Tolerant Wireless Sensor Networks for Wind Tunnel Monitoring," *International Conference on Apperceiving Computing and Intelligence Analysis*, pp. 408–411, 2008.
2. W.-B. Pöttner, F. Büsching, G. von Zengen, and L. Wolf, "Data elevators: Applying the bundle protocol in delay tolerant wireless sensor networks," *Mobile Adhoc and Sensor Systems (MASS)*, pp. 218–226, 2012.
3. M. Zennaro, "Wireless Sensor Networks for Development: Potentials and Open Issues," Ph.D. dissertation, KTH Royal Institute of Technology, 2010.
4. V. Sachidananda, A. Khelil, and N. Suri, "Quality of Information in Wireless Sensor Networks : A Survey," *ICIQ*, vol. 1, pp. 1–15, 2010.
5. C. Liu, K. Wu, and J. Pei, "An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, pp. 1010–1023, 2007.
6. G. Humber and E. C.-H. Ngai, "Quality-Of-Information Aware Data Delivery for Wireless Sensor Networks: Description and Experiments," *IEEE Wireless Communication and Networking Conference*, pp. 1–6, Apr. 2010.
7. C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri, "An Adaptive Sampling Algorithm for Effective Energy Management in Wireless Sensor Networks With Energy-Hungry Sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 2, pp. 335–344, Feb. 2010.

8. N. Nasser, L. Karim, and T. Taleb, "Dynamic Multilevel Priority Packet Scheduling Scheme for Wireless Sensor Network," *IEEE Transactions on Wireless Communications*, vol. 12, no. 4, pp. 1448–1459, Apr. 2013.
9. M. R. Lyu, "Congestion performance improvement in wireless sensor networks," *2012 IEEE Aerospace Conference*, pp. 1–9, Mar. 2012.
10. A. Lindgren and K. K. Phanse, "Evaluation of Queueing Policies and Forwarding Strategies for Routing in Intermittently Connected Networks," in *2006 1st International Conference on Communication Systems Software & Middleware*. IEEE, 2006, pp. 1–10.
11. S. Scherfke and O. Lünsdorf, "SimPy - Discrete Event Simulation for Python," 2014. [Online]. Available: http://simpy.readthedocs.org/
12. A. Lindgren, A. Doria, E. Davies, and S. Grasic, "Probabilistic Routing Protocol for Intermittently Connected Networks," RFC 6693 (Experimental), Internet Engineering Task Force, Aug. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6693.txt
13. N. T. M.-c. Case, T. Spyropoulos, S. Member, and K. Psounis, "Efficient Routing in Intermittently Connected Mobile Networks: The Multiple-Copy Case," *EEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 77–90, 2008.
14. P. Söderman, "UPS data set," *figshare*, May 2014. [Online]. Available: http://figshare.com/articles/UPS_data_set/1018702
15. ——, "Window data set," *figshare*, May 2014. [Online]. Available: http://figshare.com/articles/Window_data_set/1018703
16. ——, "Garden data set," *figshare*, May 2014. [Online]. Available: http://figshare.com/articles/Garden_data_set/1018700
17. ——, "Ocean data set," *figshare*, May 2014. [Online]. Available: http://figshare.com/articles/Ocean_data_set/1018701
18. NOAA, "National Data Buoy Center," 2008. [Online]. Available: http://www.ndbc.noaa.gov/
19. C. Luo, F. Wu, J. Sun, and C. Chen, "Compressive data gathering for large-scale wireless sensor networks," *IEEE Transactions on Mobile computing and networking*, no. 800, pp. 145–156, 2009.
20. T. Imielinski and H. F. Korth, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile Computing*, 1996, pp. 153–181.
21. M. Hempstead, M. J. Lyons, D. Brooks, and G.-Y. Wei, "Survey of Hardware Systems for Wireless Sensor Networks," *Journal of Low Power Electronics*, vol. 4, no. 1, pp. 11–20, Apr. 2008.
22. M. C. Vuran, O. B. Akan, and I. F. Akyildiz, "Spatio-temporal correlation: theory and applications for wireless sensor networks," *Computer Networks*, vol. 45, no. 3, pp. 245–259, Jun. 2004.
23. I. Al-Karaki, R. UI-Mustafa, and A. Kamal, "Data aggregation in wireless sensor networks - exact and approximate algorithms," *2004 Workshop on High Performance Switching and Routing, 2004. HPSR.*, pp. 241–245, 2004.
24. T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, Aug. 2002.