Prioritized Redundancy of Data Storage in Wireless Sensor Networks

Cosmin Dini

University of Haute Alsace
34 rue du Grillenbreit 68008 Colmar – France
France
cosmin.dini@uha.fr

Pascal Lorenz

University of Haute Alsace
34 rue du Grillenbreit 68008 Colmar – France
France
lorenz@ieee.org

Abstract— **Wireless Sensor Networks have evolved into complex deployments, where their nodes can have a full network protocol stack, database systems, etc. The main rationed resource in such a deployment is energy. Having power usage tightly managed ensures a long operational life for the node in cases where replenishments (either via recharge or battery change) are difficult or impossible. The basic deployment of Wireless Sensor Networks consists of sensing nodes as well as a relay node (i.e., sink), which collects sensory data to be relayed via a reliable network. The sink node can become unreachable due to malfunction, scheduled uptime or, in the case of mobile sink nodes, due to being out of the sensor nodes' reach. In addition, the sensor nodes may decide against relaying data for some period. In these cases, optimal use of sensor node memory space also becomes critical. In this article, we classify data types and establish a set of node level approaches that can be taken to make the most of limited data storage via a prioritized data reduction. Wireless Sensor Networks often operate in locations with limited access while relying on a restricted set of resources. This calls for careful management of local assets such as energy and storage. Storage is used to host data of various interest levels for subsequent relay to a base station or for queries through the network. The size of the data needs to be managed in view of data's relevance. While there is a debate for complete or partial data extraction from sensors, having special data process functions and operation primitives proven useful for sensor OSs. To deal with robustness and reliability, data processing at the network/sensor level satisfies some of the reliability requirements, especially when communications are not operational. There are situations were data reduction is an alternative when storage is not longer available and data is aging, especially when some sensor links are not properly operational. Using predictions and optimized parameters to prioritize data reduction is a solution. While approaching the data reduction from the perspective of a single node is important, there are benefits from looking at Wireless Sensor Networks deployment as a whole. There is an opportunity to relocate some data to less active nodes and spare it from a reduction process. There are energy considerations to take into account in evaluating the potential benefit from spending some energy to relocate data. Three factors play a part in this: the source node, the receiver node, and the data itself. The source node needs to save enough energy to relay its data once the sink node becomes available. The receiver node needs to have space available, or at least have a lot of space occupied by low importance data. The data itself needs to be self-enclosed as far as parameters needed for importance computation. Another aspect that becomes feasible in a multi node environment is having redundant copies of data to protect against potential node failures. These copies have their own particularities, a notable one being that their importance function cannot depend on other data. This paper presents a methodology that enables the node to be useful by collecting data beyond the point where its data storage size would otherwise allow. We build upon primitive data reduction operations to construct a framework that can be used for a sensible data age-out scheme. The methodology enables various factors, both internal and external to the sensor, to influence the data aging process and the data reduction operations. Additionally, we define special heuristics for data reduction using a set of data processing primitives and special data parameters. We apply these heuristics via a methodology that enables various factors, both internal and external to the sensor, to influence the data aging process and the data reduction operations.**

Keywords - sensors; networks; storage; operations; data management; data sensor storage; data priority; optimized parameters; prediction models; prioritized redundancy.

I. INTRODUCTION

Wireless Sensor Networks (WSN) have appeared as a special case of wireless networks specialized in data gathering. They find a niche in industrial monitoring, military sensing, locating assets, etc. [1]. An advantage comes from the fact that such a deployment makes no assumption about pre-existing infrastructure. Therefore, the WSN nodes have to bring with them all the resources that they expect to utilize during their lifetime: power, storage, processing unit, antenna, etc. Figure 1 presents the classical WSN deployment of sensor nodes n_i, and a sink S. Each sensor node n can wirelessly communicate with other sensor nodes that are within reach, shown in dashed lines.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

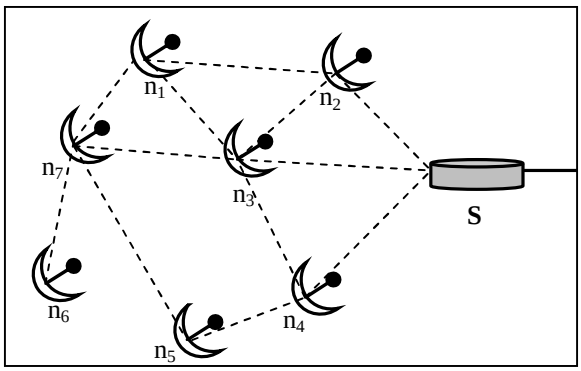


Figure 1. Classical WSN deployment.

Much research has been devoted to optimizing the usage of limited resources. In particular for WSNs, energy has been a main focus. Very resourceful power sources have been suggested, nuclear energy included. Both routing and dissemination protocols have been approached from the energy conservation standpoint. In addition, some proposals target replenishing energy from sources such as sun, wind, water flow, etc. [10].

Other resources related to WSN deployment have received less attention compared to energy source and usage. In this article, the focus is data storage, which just like energy, is limited. Unlike energy, which once used is gone, data storage space can be reclaimed by discarding existing stored data. Another difference between energy and data storage constraints is that one can propose ways of harnessing energy to prolong the life of a wireless node, but so far, there is no way of harnessing data storage space.

There are many examples outside of the WSN world where storage space is a factor. Many cases are outside of the technology world and include storage spaces, garages, warehouses, disk drives, kitchen drawers, video surveillance recording, etc. Unlike the WSN scenario, these cases allow for direct human intervention: additional storage space, although for a cost, can be achieved.

We consider a WSN node in cases where unloading the data is not possible at all times. The sink node, or the next hop routing node, may not be available at times. Referring to Figure 1, if the sink S is not available, nodes n_2, n_3, and n_4 cannot relay their data. Similarly, if n_7 is unavailable, n_6 cannot relay data as n_7 is the only possible next hop for routing. It is the responsibility of the node to use the storage space in order to hold the most relevant data until this data can be relayed at the expense of less relevant data. A set of business logic instructions that are deployed with the node provide the decision making. The same principles can apply to sensor nodes that are deployed and later physically retrieved for data extraction without ever having to wirelessly transmit any data.

Specialized operating systems, such as TinyOS, provide the primitive operation set for code execution. The data storage approach that we propose also needs to be supported by a set of primitive operations.

Unattended data collections in remote areas without ease of access pose a challenge for traditional network deployments. Usual assumptions no longer apply, specifically: available space is an issue, power is no longer reliably available, and tech support can not immediately intervene to fix unexpected issues [13].

Wireless Sensor Networks (WSNs) have evolved to answer to these constraints. In addition to having sensing capabilities, WSNs also perform the essential functions of traditional networks, such as routing. However, under the constraints imposed by isolation of the deployment, approaches to traditional algorithms need to be reconsidered [14].

There are two approaches to data handling in WSNs. On the first hand, a sensor network can be left to its bare functions of data collection. All data is then forwarded to a base station for data analysis. In such a case, there is little computation requirement from the processing unit on the sensor nodes. This can lead to large amounts of data circulating through the network, leading to excessive use of power, a critical resource in WSNs. A second approach is to leave the collected data on the sensor nodes, while retrieving specific needed data by running queries through the network. This limits the data transfer to the exact results of the data extraction. In this case, we deal with limitations caused by the storage space on the nodes. There are some approaches to alleviate somewhat the effects of a limited storage space, and they usually involve some sort of load sharing [21].

Whatever the mechanism to make better use of storage space, there is nothing that can produce more available storage space. At some point, data needs to be aged out. What data and how it is aged out has not been fully addressed in existing literature. In [12], a set of primitive operations to support a prioritized data reduction is presented. In its description, the approach was limited to presenting the simplest building blocks on a single node. The data reduction basically presented in [12] is extended to show how the primitives are used together in order to manage the storage size aspect of a WSN node operational state. Requirements posed by unattended data collections in remote areas become very challenging for traditional network deployments. The main problem is raised by the fact that users might look for full collected data, while effective business models take into consideration a small fraction of it.

Most of the WSNs (Wireless Sensor Networks) also perform the essential functions for data processing; one of the most important, in special cases of uncontrolled link availability, is data reduction under several the constraints driven by the nature of the data, the relevance of the data, the data dependency, and the business model using such data. A sensor on a node captures a time series representing the evolution of a sensed physical variable over space and time.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Reducing the amount of data sent throughout the network is a key target for long-term, unattended network sensors. A second target, equally relevant, is defined by unattended networks with unreliable links. In this case, gathered data might rapidly aging and exceed the storage capability at a given node. Data reduction mechanisms are used to partially handle these cases [22] [23] [36].

Unnecessary communication as well as appropriate data reduction techniques can be modeled in the case of physical phenomena with a pre-defined, application-dependent accuracy [36]. If an accepted error is bounded as [-*e*, +*e*], with *e* in *R*+, only values exceeding the predicted one by +/- *e* will be communicated. Similarly, if the errors of the gathered values are within the bonded interval, data reduction can be more simplified.

The difference between these two situations is the following; for communication, in the absence of notification from a given node, the receiving node assumes that the value obtained form the prediction model is in the required error bound. For unreliable links, despite if this assumption, the gathering node must reduce the amount of collected data that accumulates. In this paper, we present a series of heuristics on predictions used to summarize collected data.

The rest of the article first deals with primitive operations that are executed at node level independently of the rest of the WSN. We then present the state of the art in collaborative data storage as well as data reduction.. The next section reviews the primitives introduced in [12]. We then extend the single node model to present a complete framework. More heuristics for prediction on data processing are presented for multi-node complemented by prioritized data redundancy.

II. Related Work

Current work in WSN associated storage management revolves around energy efficiency in manipulating and querying the data [3][4], improving the characteristics of stored data [5] [6][7], and making use of adjacent nodes in order to gain access to additional storage [8][9].

Siegmund et al. [5] propose FAME-DBMS to provide a robust data storage solution. This system ensures reliability and integrity of the data, and provides a customizable query engine. It answers to the requirements related to data retrieval more so than to storing data. In order to deal specifically with encryption, Joao Girão et al. [6] present edTinyPEDS, an encryption data storage engine.

The energy usage is still relevant when focusing on storage and querying. Ahn and Krishnamachari [3] evaluated the scalability of a WSN performance with respect to the distributed nature of data.

Park and Elmasri [4] evaluated several storage schemes in terms of where the data is stored, what types of routing protocols are most appropriate, and finally the impact that each storage approach has on energy usage.

Khan et al. [8] presented the problem of data persistence in a congested WSN scenario. The main point is that congested networks can drop packets, which in turn translates to a waste of energy equivalent to the cost of sending the dropped packets. The proposed approach involves clustering where cluster nodes can act as temporary buffers during congestion periods.

Current work on WSN related storage has been limited to data characteristics and management across several nodes. The case of a standalone node has not yet been considered.

In the following, we present current approaches to deal with a continuously accumulating amount of data. We cover both aspects geared towards individual nodes and aspects geared towards network wide approaches.

2.1 Data Reduction

In their work on managing data in storage-centric WSNs, Diao et al. [20] touch on some of the issues surrounding the management of growing data amounts. There exists a common perception that, for historical data, every single bit of collected data must be stored and maintained. A distinction is made between “dumb data collection” sensor networks, and those that support queries throughout an entire WSN. There is clearly a serious potential for excessive data accumulation in the case of the latter. The potential for data aging to alleviate the problem is brought up, but no framework is established on how aging can be done in a sensible manner; vacuuming older data to secondary storage is proposed as a potential solution, but it assumes a rather reliable connection to a base station.

Khan et al. [15] address the point of data persistence within a network in the case of congestion along transmission paths. Similarities can be seen between this case and the case of excessively accumulating data. The proposed solution in the case of a data storage issue due to congestion is additional buffering along the path to a sink node. This cannot be a full answer to the problem of a growing amount of collected data. Whatever the about of storage on hand, there will always be applications and processes demanding even more data space.

Tilak et al. [16] only briefly touch upon the data aging aspect of storage, but concentrate more on the benefits of collaborative or global approaches.

The approaches that we have seen so far lack consideration for the exact data that is collected. The tendency to treat data uniformly, or as a function of time only, can leave out data of significant importance.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

A more functional solution to a single WSN node storage size management is given in [12]. The underlying assumptions are that we are dealing with a "dumb data collection" sensor network. The deployment is therefore only used to collect data and do the best it can, at each individual node, to maintain the most relevant data until an opportunity presents itself to relay all this data to a sink node. Data reduction is performed according to pre-established business case rules and several methods to achieve data reduction are presented. On a case by case basis, some data reduction approaches are preferable to others. We consider this to address the problem, but added benefit can be found in allowing nodes that experience a spike in data collection to share some of the data with neighboring node(s).

2.2 Collaborative Storage

Several approaches have been proposed with regards to collaborative storage. By having a wider view of the deployment, such protocols can attempt to address aspects such as alleviating situations where certain nodes produce more data than others, deduplication of data that was collected unknowingly by more than one sensor, and ultimately, as a side effect of deduplication, less energy is needed to relay the data to a base station.

In [17], Tilak et al. proposed a Cluster Based Collaborative Storage (CBCS) as a specific solution to collaborative storage. Several algorithm improvements in WSNs have this common approach of clustering nodes, electing cluster heads, and establishing an overall tree structure through the network. CBCS does this in the context of storage management. Nodes are grouped in clusters based on geographical data. Cluster heads (CH) are elected, one per cluster; they have the task of aggregating all data from the cluster nodes. This improves power use efficiency as only the CH needs to further relay the data towards the sink node. Such an approach seems to place more emphasis on the energy saving rather than dealing with large amounts of data. Aggregating all the data from the cluster nodes on the CH storage space cannot be beneficial when we are trying to solve storage issues.

In Figure 2, a visual representation of clusters is shown. Most clusters are created with geographical proximity in mind in order to minimize energy expenditures on communication. The designated or elected cluster heads, shown circled, handle the communication with a sink node, S, which is linked to a stable reliable network.

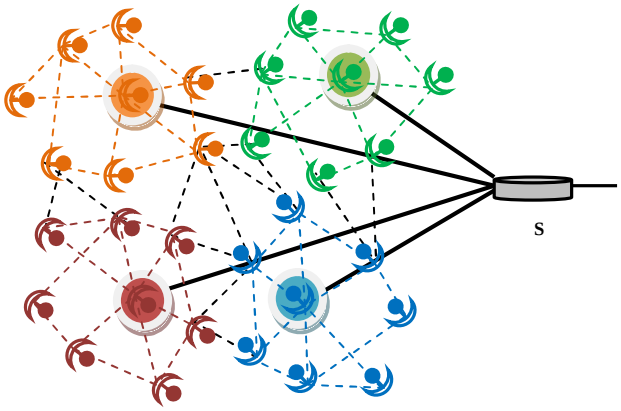


Figure 2. Clusters in a WSN.

In [18], Shenker et al. proposed a method, Data Centric Storage (DCS), to store data as identified by its name. Related data would in the end be stored either on the same or on neighboring sensor nodes. This would facilitate deduplication and also improve queries since data pertaining to a query would reside in the same proximity, therefore avoiding the need for queries to be run through large sections of the network. This approach is useful for WSNs that are designed to support searches, but does not apply to networks where data analysis is done offsite.

Siegmund et al. [19 addressed the issue of data integrity. Data redundancy is achieved in the network via the implementation of a new abstraction layer in the WSN. This layer can support the need for data redundancy. While robustness is of importance, there is very limited work in view of alleviating potential data overload in the network.

Collaborative storage presents challenges in terms of locating data during queries through the network. It also adds to the power requirements of the nodes. In the proposals published so far, there is no consideration for collaborative storage to mitigate memory limitations on some sensor nodes. There is also no effort to give a semantic to the stored data in order to assist with data age-out in the context of a node who is only storing the data without having produced it.

In the following, we summarize a data processing model introduced in [22] [23] and prediction approaches for data reduction [24].

In the past, the database community pushed different data-reduction operators, *e.g.*, aggregation and reduction, with no enough flexibility to handle extracting complete raw sensor readings (i.e., using "SELECT *" queries).

There are two specific needs to perform in-network data processing, *i.e.*, (i) to significantly reduce communications costs (energy), and (ii) to deal with link-down situations. In-network aggregation was proposed in [26][27], while data reduction via wavelets or distributed regression in [28][29].

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

All these techniques do not provide the desired data granularity, as requested by network users.

Managing data in a storage-centric approach was studied in different approaches, based on a reliable connection [32], additional buffering [30], or collaborative framework [31]. Details on collaborative storage are provided in [23].

OS primitives acting on recurring and non-recurring data collection have been proposed in [22]. Mainly, compression, thinning, sparsing, grain coursing, and range representation were used to deal with data aging in a pessimistic and optimistic approach. As a note, data deduplication was not considered in the above model. To optimize data reduction, concepts of data units, data importance, and compensation factors were introduced in [23]. Mainly, measurements are partitioned into contiguous intervals; data importance relates to the business semantics, while the compensation factor underlines the importance in business computation of a given data. The model considers that there is data that cannot be reduced in any circumstance. A mechanism for data dependency between different data units was presented in [23]. Further division of the data units (leading to more flexibility) was not considered at this point. Associated with the new concepts, the following functions were introduced: interval production function (only for recurring data), default compensation function, data importance function, and data reduction function. Solutions based on data redundancy (leading to more robust deployments and measurements) were not considered.

A data reduction specification use case considering data dependency across data units was presented in [23]. Consequently, appropriate values for data importance can be derived considering the constraints on the data importance computation as pertaining to two categories: (i) internal constraints and (ii) external constraints. External constraints are caused by factors over which input data has no effect. Such factors are data age and inherent interest in the data depending on the exact purpose of the data collection. Internal constraints represent inter- and intra-data dependencies.

A prediction model for approximate data collection is presented in [24]. The techniques are based on probabilistic models (BBQ system, [33]). We apply the prediction models to the framework proposed in [22][23] considering also the approximation scheme providing data compression and prediction [34] and predictive models from [35].

In this paper, we consider the PDR components introduced in [23] (Figure 1) to derive appropriate data reduction considering known correlations (spatial, temporal, etc.) and prediction models.

Several approaches and protocols have been proposed with regards to collaborative storage. By having a wider view of the deployment, such protocols attempt to address aspects such as alleviating situations, where certain nodes produce more data than others, deduplication of data that was collected unknowingly by more than one sensor, and ultimately, as a side effect of deduplication, less energy is needed to relay the data to a base station.

In [37], Tilak *et al.* proposed a Cluster Based Collaborative Storage (CBCS) as a specific solution to collaborative storage. Several algorithm improvements in WSNs have this common approach of clustering nodes, electing cluster heads, and establishing an overall tree structure through the network. CBCS does this in the context of storage management. Nodes are grouped in clusters, based on geographical data. Cluster heads (CH) are elected, one per cluster; they have the task of aggregating all data from the cluster nodes. This improves power use efficiency as only the CH needs to further relay the data towards the sink node. Such an approach seems to place more emphasis on the energy saving rather than dealing with large amounts of data. Aggregating all the data from the cluster nodes on the CH storage space cannot be beneficial when we are trying to solve storage issues.

In Figure 2, a visual representation of clusters is shown. Most clusters are created with geographical proximity in mind in order to minimize energy expenditures on communication. The designated or elected cluster heads, shown circled, handle the communication with a sink node, S, which is linked to a stable reliable network.

In [38], Shenker *et al*. proposed a method, Data Centric Storage (DCS), to store data as identified by its name. Related data would in the end be stored either on the same node, or on neighboring sensor nodes. This would facilitate deduplication and also improve queries since data pertaining to a query would reside in the same proximity, therefore avoiding the need for queries to be run through large sections of the network. This approach is useful for WSNs that are designed to support searches, but does not apply to networks where data analysis is done offsite.

Siegmund *et al.* [39] address the issue of data integrity. Data redundancy is achieved in the network via the implementation of a new abstraction layer in the WSN. This layer can support the need for data redundancy. While robustness is of importance, there is very limited work in view of alleviating potential data overload in the network.

Collaborative storage presents challenges in terms of locating data during queries through the network. It also adds to the power requirements of the nodes. In the proposals published so far, there is no consideration for collaborative storage to mitigate memory limitations on some sensor nodes. There is also no effort to give a semantic to the stored data in order to assist with data age-out in the context of a node who is only storing the data without having produced it.

Having multiple nodes interact in an effort to better manage storage space gives the opportunity for two considerations: storage space sharing and redundancy. In this chapter, we address these concepts from the perspective of prioritized data reduction.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

2.3 Storage space sharing

2.3.1 Load sharing

Most research related to sharing across several nodes the impact created by data refer to balancing out the data access hot-spots. Occasionally, data queried in a network will be accessed in a skewed manner resulting in a small set of nodes receiving a disproportionate amount of queries and hence taxing the energy storage of those specific nodes. In order to deal with these cases, methods to replicate the data are presented as well as methods to divide the data across several nodes. In [40], the problem is approached from the perspective of k-d tree mapped data. In such a mapping, data storage is done independently of the node that produced the data. When one node becomes overwhelmed with respect to the amount of data that it needs to store, a tree rearrangement is done in order to balance the data load.

A k-d tree covers a two dimensional surface by recursively dividing the surface in two parts. At the point where we have a single node inside a parcel, that parcel no longer undergoes division and the node becomes the leaf of the path that created the parcel.

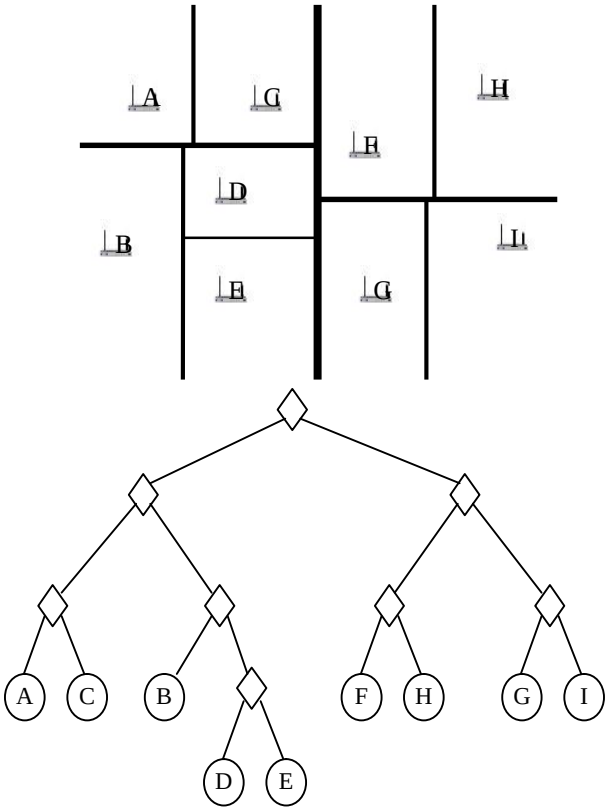


Figure 3. Division of a WSN in a k-d tree.

Another facet of load sharing is found in [41], where the authors address the problem of congestion at nodes on highly used paths. Not only are these nodes' energy supplies overtaxed, but the limited storage space that they have may not be enough to transiently hold all the packets. The proposal is to group the sensor nodes into small clusters, ideally a highly redundant deployment. When high traffic volume is sensed, the cluster head node starts to divert traffic to neighboring nodes. Because the nodes are in the vicinity, the congestion is localized to the specific area without spreading to other nodes. For critical data, the head node can make the decision to replicate the data several times so as to assure its continuity even in the event of a node failure. One drawback is a potential added delay caused by the added store and forward operations within a cluster.

2.3.2 Storage sharing

In this section, we cover the aspects of sharing the storage space that exists within an entire wireless sensor network. This is already done as part of geolocated data having as aim to facilitate queries. In our case, we assume a network that does not support in-network queries, but relays all data to the sink for offsite processing. Data produced by a node stays on that node until delivered to the sink. However, in the interest of balancing out space needs, we consider the possibility to forward certain data to other nodes as opposed to applying prioritized data reductions.

Until storage sharing, it was rather straight forward to perform a round of data reduction, i.e., select the data unit with the lowest importance and apply the reduction function. This could be done iteratively to bring the memory usage below a certain expected threshold. With an option to package and send data as a method of storage size recovery, several decision factors appear. How much data to send? How to select the most appropriate data to send? How to select the best node to send the data to? There are no definite answers in situations where external inputs dictate how much or how little data needs to be stored. There are however preferred ways to handle storage space sharing.

III. FRAMEWORK AND MECHANISMS FOR ONE NODE

3.1 Components of a WSN node

In this section, we describe the conceptualized components performing the tasks - known as well as newly proposed - associated with a WSN node.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

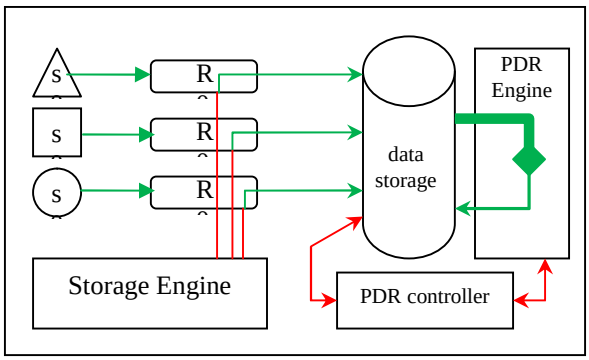


Figure 4. WSN node building blocks.

There are several physical sensors s on a WSN node, each specialized in sensing a specific parameter: temperature, pressure, etc. Each of these physical sensors has registers (R), which are updated to reflect a currently observed parameter vale. The Storage Engine (SE) has the appropriate business logic knowledge to determine at which point to trigger the transfer of data from the registers into the data storage. Prioritized Data Reduction (PDR) is achieved via a controller and an engine. The PDR controller performs periodic checks on the data storage and it can also accept traps from the data storage. When business knowledge dictates, the PDR controller instructs the PDR engine to perform specific data reduction operations.

We first focus on the operations executed inside the PDR engine. For that matter, we first characterize how the SE operates.

3.2 A model for data classification

Triggers or conditions are used to control data collection. These conditions trigger the SE to perform transfer operations from R into the data storage.

3.2.1 Conditions

We break down the conditions into two categories: value-based and sequence-based. Value-based conditions are evaluated as a whole and can immediately evaluate to TRUE or FALSE:

```
if ((temperature > α) && (humidity > β))
```

Sequence-based conditions are evaluated in sequence. We must establish a chain of TRUE evaluations in order for the entire condition to be considered TRUE. If one element in the sequence evaluates to FALSE, we pause there until the next round of evaluations:

```
sequence {
   s1: (temperature < 0C)
   s2: (temperature > 0C)
   s3: (temperature > 10C)
   s4: (temperature > 20C)
   s5: (temperature > 25C)
}
```

Sequences are useful in establishing trends. Even though one sequence item may temporarily be FALSE, once we arrive to s5 and it evaluates to TRUE, the sequence is said to evaluate to TRUE.

3.2.2 Data Collection

Data collections can be classified in two categories, based on the periodicity of the collection: there are samplings at defined intervals (recurring), or single shot samples (non-recurring).

3.2.2.1 Recurring Data Collection

Recurring data collections involve taking specific measurements at defined time intervals. One example of this type of data collection is temperature. We can specify such intervals at microseconds to hours and even less frequent. The sampling rate would depend on the exact use for the data collected and according to the business model. Not all recurring data sampling is enabled by default and continuously done throughout the live of the sensor. There are conditions that can trigger starting or stopping a series of such data collections.

As an example, we assume we are monitoring temperature on the side of a volcano in order to detect abnormally high values. We assume that baseline values are available. Under normal conditions, a few degrees difference warmer than prevailing temperatures may be acceptable, but once the temperature crosses a certain value (hinting of some sort of activity), it becomes interesting to start taking measurements of several factors: sound, land vibrations, gas composition, etc. When the ambient temperature returns to a specific value, it may not be of interest to sample a wide variety of parameters.

Recurring data sampling can be defined by a start condition, a stop condition, a sampled parameter, and a recurrence window.

```
start: (temperature > (baseline + Δ))
stop: (temperature <= baseline)
sample parameter: sound
recurrence: 1ms
```

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Once a collection has started, a second instance of the same collection cannot start even though the start condition evaluates as TRUE.

It serves no purpose to sample data any faster than the sensory devices can update registers holding the sensed data.

As a special case, the primitive values TRUE and FALSE can be used as a start condition and a stop condition respectively, and hence a continuous sampling is achieved.

We label as a *recurring data instance (RDI)* a recording of the entire set of data points from the collection start to collection stop, or to current time if the collection has not stopped.

Defining an RDI

RDI: (T_{start}, T_{end}, param, recurrence, resolution, compression), where:
T_{start}: start time
T_{end}: stop time
param: the parameter being collected
recurrence: how often the collection is done
resolution: how precise the stored value is
compression: boolean stating if the RDI has been compressed
e.g., RDI(2009/12/06 16:43:23, ongoing, temperature, 60 seconds, 0.01C, FALSE)

3.2.2.2 Non-Recurring Data Collection

Non-recurring data collection happens when a specific condition is met. It leads to a single value being stored every time the condition evaluated to true. Such conditions must be written as a sequence so as to avoid constant firing of the rule and hence leading to a constant parameter sampling.

---

sample condition:
  sequence {
    light < 50lx
    light > 10000lx
  }
sample parameter: temperature

What the example above means is that we are sampling the temperature of a location after the sun has come up and is providing a specific light intensity. The reason we require a value increase from under 50lx to over 10000lx is to establish a trend.

If we simply state the above as :

---

sample condition: (light > 10000lx}
sample parameter: temperature

---

then we would have continuous temperature sampling once the light goes over 10000lx.

We label as a *non recurring data instance (NRDI)* a non-recurring stored data recording.

Defining an NRDI

NRDI: (T, param, resolution, compression), where:
T: recording time
param: the parameter being collected
resolution: how precise the stored value is
compression: boolean stating if the NRDI has been compressed
e.g., NRDI(2009/12/06 16:43:23, temperature, 0.01C, FALSE)

3.3 Primitives for space optimization

In this section, we introduce primitives which are invoked inside the PDR Engine once the PDR Controller has identified data to be subjected to reduction.

3.3.1 Compression

Lossless data compression algorithms are widely available and used [11]. On a normal basis, compression and decompression cause little impact. On a sensor node, compressing certain portions of the data will yield available data space with no information loss. The loss is from a flexibility perspective. Once compressed, the data becomes a blob which should be treated as an atomic entity. The WSN node looses the capacity to discard partial data.

Such an approach is recommended for very critical data, and hence very important, that can never be discarded. Otherwise, it should be used for non-recurring data instances, or for portions of recurring data collections that can be dropped one whole section at a time. A parameterized compression is used to specify which span of a data instance should be compressed:

Compress[(a, b)](RDI) signals that only the data from time interval a to b is compressed, while the rest remains as initial.

The non-parameterized compression affects the entire RDI.

Usage on non-recurring data:

Compress(NRDI(2007/11/24 13:21:37, humidity, 0.01%, FALSE)) = NRDI(2007/11/24 13:21:37, humidity, 0.01%, TRUE)

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Usage on portions of recurring data by first dividing the data instance several data instances:

Compress[(2008/11/11 12:00:00, 2008/11/12 12:00:00)](RDI: (2008/11/10 12:00:00, 2008/11/13 12:00:00, temperature, 3600 seconds, 0.01C, FALSE)) = RDI: (2008/11/10 12:00:00, 2008/11/11 12:00:00, temperature, 3600 seconds, 0.01C, FALSE) + Compress(RDI: (2008/11/11 12:00:00, 2008/11/12 12:00:00, temperature, 3600 seconds, 0.01C, FALSE)) + RDI: (2008/11/12 12:00:00, 2008/11/13 12:00:00, temperature, 3600 seconds, 0.01C, FALSE)

At this point, the second data instance can undergo compression and becomes:

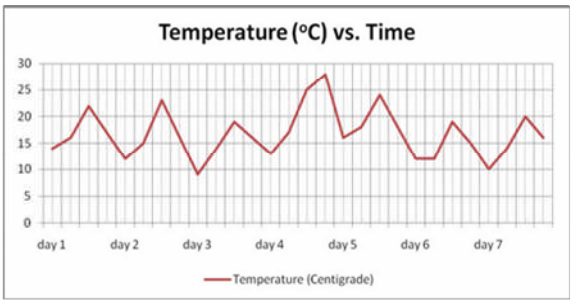
RDI: (2008/11/11 12:00:00, 2008/11/12 12:00:00, temperature, 3600 seconds, 0.01C, TRUE)

while the first and third data instance retain all original data.

3.3.2 Thinning

For a non-recurring data instance, thinning involves simply discarding the collected data. For recurring data, thinning involves discarding a contiguous amount of data that corresponds to a time span of low importance in the case of recurring data instances. This can be used when the collection has a cyclic pattern and a long sampling period gives little additional insight when compared to a somewhat shorter period, or a period with gaps.

To better clarify, we can resort again to the temperature sampling example. Let's assume that we are sampling temperature every minute. This has been going on for five months. Depending on the business case, it may be acceptable, without any significant impact to data significance, to either discard data pertaining to the third operational month, or to discard data pertaining to the third quarter of each operational month. Figure 5 shows the impact of thinning on a data sample.



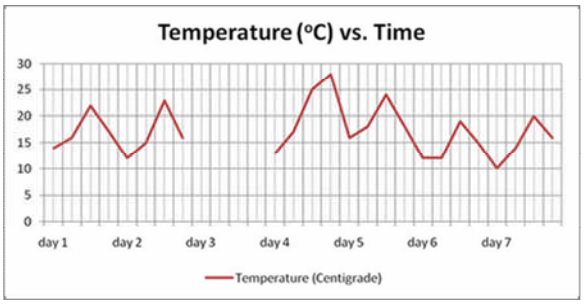


Figure 5. Impact of thinning on data top) initial data bottom) resulting data after thinning.

Thin[(a, b)](RDI) signals that the data collected between time a and time b are dropped from the data instance.

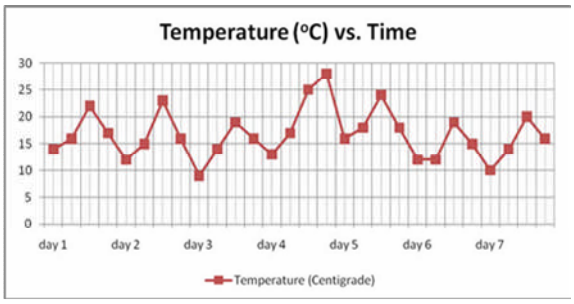
Usage of thinning on recurring data:

Thin[(2008/11/12 12:00:00, 2008/11/13 12:00:00)] (RDI(2008/11/10 12:00:00, 2008/11/17 12:00:00, temperature, 6 hours, 1C, FALSE)) = RDI(2008/11/10 12:00:00, 2008/11/12 6:00:00, temperature, 6 hours, 1C, FALSE) + RDI(2008/11/13 12:00:00, 2008/11/17 12:00:00, temperature, 6 hours, 1C, FALSE)

3.3.3 Sparsing

Sparsing can only be used recurring data collections. If the global pattern of fluctuation in the measurement is an important factor, then it is important not to lose entire spans of information. In such cases, the recurrence window can be widened by means of dropping values at regular intervals. The resolution suffers, but the overall pattern is conserved.

Looking at the same example as in 3.2, instead of dropping a full data set corresponding to day 3, we are going to double the sampling interval for days 2 and 3.



2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

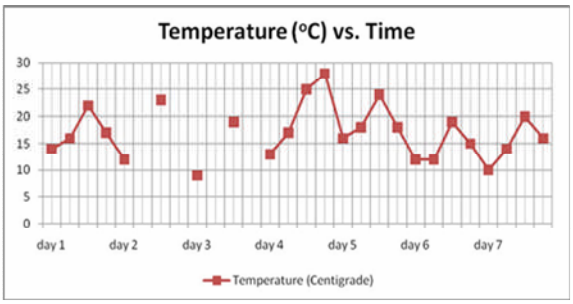


Figure 6. Impact of data sparsing top) on the left with initial data bottom) on the right after sparsing is applied.

The syntax for sparsing is Sparse[(a,b,r,s)](RDI), which means that for the time interval between a and b, r entries are removed out of every s entries.

Usage of sparsing on recurring data:

Thin[(2008/11/12 12:00:00, 2008/11/14 12:00:00, 1, 2)](RDI(2008/11/10 12:00:00, 2008/11/17 12:00:00, temperature, 6 hours, 1C, FALSE)) = RDI(2008/11/10 12:00:00, 2008/11/12 6:00:00, temperature, 6 hours, 1C, FALSE) + RDI(2008/11/12 12:00:00, 2008/11/14 6:00:00, temperature, 12 hours, 1C, FALSE) + RDI(2008/11/14 12:00:00, 2008/11/17 12:00:00, temperature, 6 hours, 1C, FALSE)

3.3.4 Grain coarsing

Data collections can be with very high precision, or can be with lower precision. For example, captured images can be anywhere from black and white to 24 bit images. To a lesser extent, this can be applied to numeric data which contains more precision in an 8 byte floating point representation versus a 2 byte integer representation.

Grain coarsing can be applied to both non-recurring and recurring data. Just like thinning and sparsing, we can opt to apply grain coarsing only to a specific interval of an RDI.

Because precision or resolution depends on the nature of the data, we give the example of an image whose resolution is measured in dpi. To specify a 50% decrease in DPI, the exact call would be GrainCoarse[(50%)](NRDI).

Usage of grain coarsing on an image:

GrainCoarse[(50%)](NRDI(2007/03/23 17:32:45, image, 300dpi, FALSE)) = NRDI(2007/03/23 17:32:45, image, 150dpi, FALSE)

3.3.5 Range representation

In some cases we would like to eliminate most of the collected data for a specific parameter over a certain time range, yet still keep some hint of where values were for that range. In such an instance, we can keep for example the minimum value, the maximum value, the average, as well as the number of values used to compute the average and how far apart in time those values were. Note that the minimum and maximum values are not necessarily correct, but they are the largest and smallest value as far as available data samples.

A case where this can be misleading is for example a recurring data sampling that has undergone extensive sampling. In that case, it is plausible that many of the brief spikes and dips in values are lost.

The range primitive applied to an RDI produce a tuple: Range(RDI) = Tuple(T_{start}, T_{end}, Min, Max, Average). A data span that is represented as a tuple can be considered for further space optimization in the same manner as an NRDI.

3.6 Usage of primitives

The proposed space optimization methods are primitives. They are not the last word on a specific data collection. It should be noted that the same data can be subjected multiple times to data reduction, and each time a different mechanism can be deemed appropriate. While data morphs, its importance changes as well.

We take the example of a single sensor node that is deployed in a dangerous climate area. The node is to collect several parameters over a long period of time. At the end, the node is either removed or a sink node is placed in proximity for a brief period in order to retrieve collected data.

The parameters that are to be surveyed are: temperature, barometric pressure, light, humidity, vibrations, and CO2 concentration. There are several recurring data collections as well as triggered single time collections.

We grade data importance from 0 to 1 as a continuous value. The value of 1 is reserved for critical data that under no circumstance should be reduced in size.

Critical data involves a full day's unaltered temperature and barometric pressure sampling every second on the first operational day, and every 10 minutes thereafter with importance 0.5. Except for the first day, recurring recordings of these two parameters are subject to 50% sparsing while at the same time their importance increases by half the interval between current importance and 1.

Whenever a vibration amounting to 2nd degree on the Richter scale, all parameters are collected for one hour at 5 second intervals and bear importance 1. This collection is followed by 24 hours of collections every minute with importance 0.8. This collection can be subjected to range representation based on 10 minute intervals, which data now becomes of importance 1.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

With this data set and requirements, the simulation is allowed to proceed both with data reduction and no data reduction. Here is what is found in each case:

3. 6.1 No data reduction

Under an approach of no data reduction, the storage space was used up in about 12 operational days. The following were found in the data storage:

- First day of temperature reading every second
- First day of barometric pressure reading every second
- Eleven days of temperature readings every 10 minutes
- Eleven days of barometric pressure readings every 10 minutes
- Data collections related to three vibration shocks: all parameters collected for one hour at 5 second intervals, and 24 hours every minute

3.6.2 With data reduction supported by proposed primitives

While allowing for the data reduction, here is what was found on the machine after 15 operational days:

- First day of temperature reading every second compressed
- First day of barometric pressure reading every second compressed
- Fourteen days of temperature readings every 10 minutes out of which the first six had been sparsed (i.e. are now every 20 minutes)
- Fourteen days of temperature readings every 10 minutes out of which the first six had been sparsed (i.e. are now every 20 minutes)
- Data collections related to five vibration shocks, two of which have been subjection to range representation

With all of the above in storage, there are still items that can be removed to make space for incoming data.

3.6.3 Comparison for freeing the space

Clearly the data reduction presents opportunity for storing more relevant data in the long run. There are some drawbacks regarding the resolution and recurrence of collected data, but the emphasis is placed on high importance data.

Now that we have proposed procedures for data reduction, the question is when such data reduction should be performed. There are competing constraints to consider:

- We do not want to run the data reduction process too often
- We do not want to get overenthusiastic with data reduction as the sink may soon be available for integral data transmission

The problems faced here are similar to garbage collections in processes such as JVM. The added complexity is that, as opposed to JVM garbage collector which collects true disposable garbage, the process we intend to run recovers space in exchange for some loss of lower importance data or loss of flexibility.

An assumption is made that while the storage recovery process is under way, the sensing devices have enough buffer space to store sensed data until the main processor is ready to take the data to the main storage unit.

3.6.3.1 The Pessimistic Approach

The main idea of a pessimistic approach in running a data reduction process is that the space available must always be able to accommodate the biggest possible data influx spike which cannot be dropped based on its importance. This rule can only be broken if the existing stored data has an importance level which makes it final and not subject to reduction.

Let's denote the total storage space with T, the occupied space by O, and the biggest possible influx spike by S. It follows that (T – O) >= S. Technically we can get away with T – O = S. However, we risk fluctuating values of O which leads to a repeated invocation of data reduction algorithm. The free space target depends highly on business logic and should be a value higher than S.

3.6.3.2 The Optimistic Approach

The optimistic approach keeps all data until it is time to store higher importance data. At that point, storage space will be made available by running a storage space recovery process while the incoming data is still in the sensing device buffers. For this approach, the assumption is made that data reduction is a quick process both in identifying the data subject to reduction, as well as the reduction process. In certain cases this may be true, and hence there is no need to have available space sitting unused just in case there is incoming data that needs to be stored.

IV. EXTENDING THE SINGLE NODE FUNCTIONALITY

In this section, we take as a starting point the Prioritized Data Reduction (PDR) presented in Section III and expand it to include a complete characterization of the data reduction mechanisms. The overall model of the sensor network under consideration is "dumb data collection": collection is done by the nodes, processing is done offsite. At this point, data deduplication is not addressed as part of PDR; it is left as future work.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

In Section III data importance was introduced as a continuous value assigned to collected data instances used in the data reduction process. The lowest value is 0 and the highest value is 1. In order to assign this value to collected data, the data needs to be divided in manageable pieces. Each such piece, called a *data unit*, has two associated values: *data importance* and *compensation factor*.

4.1 Data Units

Data collections can be both recurring and non recurring. The non recurring collections generate data that is stand alone and considered atomic. It makes sense to consider a non recurring data record as a single data unit.

Recurring data collections have several values over a potentially long period of time. Such a data needs to be divided in intervals that can be treated as a whole during the process of data reduction. We propose that such intervals be devised as contiguous time intervals. There is no need for the intervals to be of equal length. What the data units do is provide a contained set of data on which we can apply the primitives enumerated in Section III.

For the sake of simplicity, we also assume that once a data unit has been delimited within a recurring data collection, it is no longer subject to further division or merger with other data units. There may be flexibility gains from allowing such operations, but this is left as future work.

The lifecycle of a data unit starts with a data collection, which can potentially go through several rounds of data reduction, and ultimately possibly dropped, assuming a long period of sink node unavailability. If at some point, the sink node is available, data is simply unloaded and space freed.

To assist in selecting what data unit to target for data reduction, each data unit is reflected by a data importance, denoted I. Once a data reduction is performed, this needs to be reflected in future importance computations. For this purpose, each data unit is assigned a compensation factor, K.

4.2 Data Importance

Data importance, denoted as I, is a value that numerically reflects the relevance of a data unit for the business case. The raw value of I is primarily used to rank data units in view of data reduction, but the exact magnitude of the difference does not necessarily carry a meaning.

START
WHILE ‘storage availability level’ is below a ‘threshold’
    FIND ‘data unit ‘ with ‘lowest importance’
    APPLY ‘data reduction’ to that data unit
    ADJUST the compensation factor
    RECOMPUTE ‘importance’ for any dependent data unit

Reduction algorithm 1. High level PDRE process for data reduction.

Figure 3 shows an abstraction of the process that happens within the PDRE when the controller triggers a data reduction operation. The reduction can occur in one or more iterations. In the first step, the data unit with the lowest importance is found and the corresponding reduction method is applied. As a consequence, the compensation factor is adjusted. All other data units whose importance factor depends on the reduced data need to have their importance level recomputed.

4.3 Compensation Factor

The compensation factor, K, is an importance modifier that reflects the data reductions that have already been performed on the specific data unit. The compensation factor, is defined as a percentage value between -100% and +100% and it is used in the following manner:

$$I = \begin{cases} \text{if } K < 0, (1+K)*I' \\ \text{if } K > 0, I' + K*(I_{max}-I) \end{cases}$$

where I is the data importance for a data unit, I’ is the resulting data importance post data reduction, and I_{max} is the maximum possible value for data importance. In our example, we assume that data importance is continuous between 0 and 1; 1 is reserved for data that cannot be reduced under any circumstance.

To illustrate the application of the compensation factor, let’s take an example where a data unit’s importance value is evaluated to 0.75 before the compensation factor is applied.

- for a compensation factor K = 40%

  $I = 0.75 + 0.40(1 – 0.75) = 0.85$

- for a compensation factor K = -40%

  $I = (1 + (-0.40)) * 0.75 = 0.45$

4.4 Available Input

Determining data importance is in the hands of the business model. In this section, we identify input factors that can be used to establish the importance of a data unit.

- *age/collection time*

The data collection time is a relevant factor. Whether in a linear manner, exponential decay, or in some irregular manner, the collection time is relevant. For example, one may be interested to correlate rush hour to atmospheric CO_2 levels over a month’s period. In such a case, the time

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

dependency is rather irregular: the data collected just before, during, and just after rush hour are more important than data collected on a week-end.

- *self values*

The very collected values can affect the importance of a data unit. For example, for any parameter that is collected, any value outside of an expected interval may need further investigation. Hence, additional importance can be assigned to data units containing such values.

- *other data units of same instance*

In cases where a parameter's cyclic aberrant values are suspected, the values of a data unit affect the importance of data in other data units of the same data instance.

- *other data instances*

Similar to the case above, if correlation is to be made between multiple parameters, then special values of interest in a data unit of one parameter add importance to data units taken around the same time for other parameters measured.

4.5 Summarizing the components of RDI and NRDI

In order to make the jump from a single node model to a multiple node deployment, we have identified additional functions to better characterize data handling. In this section, we will review the components added to the model described in [12].

- interval production function (for RDI only)

For a recurring data instance, this function is used to produce data units. This function can be rather simple, such as grouping every fixed number of measurements in data units, or it can be more complex resulting in the creation of data units covering variable time spans.

- default compensation factor

The compensation factor is used to reflect already performed data reduction. A default value needs to be specified.

- data importance function

A data importance function which computes the importance of a data unit, taking into account some or all of the factors listed in section A.

- data reduction operation function

A data reduction function which, given a data unit outputs a smaller sized data unit based on its internal use of reduction primitives. It also changes the compensation factor so as to reflect upon future data importance computations.

4.6 A single node use case

Given the above extensions to the single node model, we now consider an example and get into details as far as the specifications for data collection. The approach to data collection reflects human perception with respect to quantifying and formulating data importance. In order to validate such approaches, a preliminary step is to simulate the setup intended for deployment.

4.6.1 Car traffic and CO_2 concentration use case

The main objective of the deployment in the example is to observe data that can be used to correlate car traffic volume with atmospheric CO_2 concentration. There is a broad pattern that is expected, with more CO_2 production during rush hours where there is more traffic. There are also temporary spikes in the detected levels when a large vehicle passes, such as a large truck. These spikes need to be accounted for.

The deployment consists of many sensors installed in a very large geographical area, but all in proximity of a street. Some of the sensors may be able to reach other for communication, but communication is very costly given the distance. The collection happens using a mobile base on a vehicle with. The frequency of data collection is not predetermined.

The sensors that are deployed have four sensory devices in addition to an internal clock: a gauge for CO_2 concentration, a sensor across the street to detect traffic levels, as well as to detect the presence of an oversized vehicle, a gauge for wind speed, and one for wind direction. The sensor is also able to take a panoramic photo and store the picture.

4.6.2 Data Collection Requirements

- Every 10 seconds, the CO_2 concentration is recorded continuously (data instance **A**)
- Matching the above, wind speed and direction is recorded continuously (data instance **B** and **C**)
- When a large car passes, the CO_2 concentration is recorded for 5 minutes at 1 second intervals (data instance **D**)
- Matching the above, wind speed and direction is recorded continuously (data instance **E** and **F**)
- If an unexpected spike in CO_2 is detected, a panoramic image is recorded (data instance **G**)
- The number of cars passing is recoded for every 10 second intervals (data instance **H**)

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

4.6.3 Data Reduction Specification

Each of the above data collections need to have specific interval production functions to generate data units, default compensation factor, data importance functions, and data reduction functions.

- data instances **A**, **B**, and **C**

The division in data units is done according to the time of day. The regions of higher interest (i.e. rush) are divided in smaller data units. That way, any data reduction algorithm applied to a data unit affects a smaller data interval.

Table1. Data unit size for data instances A,B, and C

time	label	data unit size
9pm – 4am	night	10 minutes
4am – 6am	pre rush	2 minutes
6am – 9am	rush	30 seconds
9am – 11am	post rush	2 minutes
11am – 2pm	day	5 minutes
2pm – 4pm	pre rush	2 minutes
4pm – 7pm	rush	30 seconds
7pm – 9pm	post rush	2 minutes

For the purpose of defining compensation factor, the reduction function, and the data reduction function, we divide the collected data into four sections. The first three sections span 7 days each, and the fourth section covers the remaining time interval. The days being labeled from 0 (most recent), here are additional parameters for the data units:

Table 1. Data handling parameters for instances **A**, **B**, and **C**

span	K	data reduction	data importance
day 0 – day 6	50%	25% trimming	set to 0.95
day 7 – day 13	25%	50% trimming	set to 0.85
day 14 – day 20	0%	75% trimming	set to 0.65
day 21 - end	-25%	75% trimming	set to 0.5

The use of K and the data reduction trimming primitive have been covered above. For the data importance, whenever a data unit moves from one span to another, its importance is set to a fixed value (as noted in the table). Only subsequent data reduction operations will affect this (via the compensation factor).

- data instances **D**, **E**, and **F**

In this case, we select a constant data unit interval for the entire span of the data collection. The data collection is rather shot, so the entire 5 minutes will be treated as a data unit.

For the remaining parameters of a data instance, we divide such instances into the 10 most recent, following 100 most recent, following 1000 most recent, and as a fourth, the remaining instances.

Table 2. Data handling parameters for instances **D**, **E**, and **F**

span	K	data reduction	data importance
10 most recent	20%	50% sparsing	set to 0.9
next 100	20%	50% sparsing	set to 0.75
next 1000	20%	50% sparsing	set to 0.50
after1000	-20%	50% sparsing	set to 0.50

The same approach as above is used in the case of data importance.

- data instance **G**

This data instance is non-recurring. A recording of this data is a data unit. Similar to the case above, we divide the instances in the 5 most recent, the following 10 captures, and the rest of the captures:

Table 3. Data handling parameters for instance **G**

span	K	data reduction	data importance
5 most recent	50%	20% resolution reduction	1
next 10	50%	20% resolution reduction	0.8
after1000	50%	20% resolution reduction	0.7

- data instance **H**

In this instance, we are dealing again with continuous data collection. In this case, counting the vehicles, we divide the collection in even data units. Just like in the first example, the data units are placed in several time spans: three spans are 30 days long, and the fourth span contains the remaining days. The data units are 5 minutes long.

Table 4. Data handling parameters for instance **H**

span	K	data reduction	data importance
day 0 – day 29	25%	50% grain coarsing	set to 0.80
day 30 – day 59	15%	50% grain coarsing	set to 0.70
day 60 – day 89	-15%	50% grain coarsing	set to 0.50
day 90 - end	-50%	50% grain coarsing	set to 0.40

4.6.4 Use Case Conclusions

The use case described is quite simple, yet it does the job of properly collecting the data. The model used needs the validation of intense simulation in order to validate the nature of the data it produces in various circumstances.

4.7 Data dependency

In the example so far, there was no dependency between the different data collections with regards to data values from a concurrent data collection. In this section, we expand on data instance A from the previous section in relation to data instance B. As a reminder, data collection A is the CO_2 concentration sensed, while B is wind speed.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

As a business case decision, we decide that certain atmospheric disturbances in the area of the sensor affect the relevance of the CO_2 concentration measurements. We are going to look at two aspects to determine dependency: (i) increase in wind speed and (ii) average wind speed within a data unit. We decide that wind acceleration beyond a fixed value *a* affects the importance of CO_2 measurements, while an average wind speed beyond a certain level *s* affects the importance of CO_2 measurements for four data units.

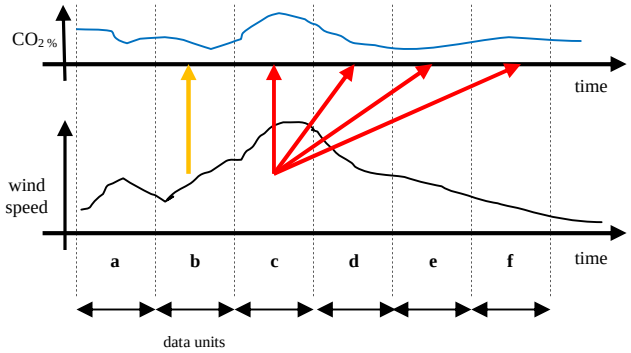


Figure 7. Data dependency across data units.

In Figure 7, several data units, labeled from a to f, are shown with corresponding reading of CO_2 concentration and wind speed. Data units b and c on the wind speed graph are of interest as they affect the importance of CO_2 readings. In data unit b, the wind acceleration causes an effect on the corresponding concentration reading. In data unit c, the average wind speed affects the data importance of four data units for concentration reading.

On a more specific note, here is a proposal for to compute the data importance mapped to a data unit of CO_2 concentration reading. We denote by *CO du* t_i - t_j the CO_2 concentration data unit between time ti and time tj, while *WS du* t_i - t_j refers to the wind speed data collection.

```
var tempI

if age(du ti - ti+1) < 7 days
        tempI ::= 0.95
else if age(du ti - ti+1) < 13 days
        tempI ::= 0.85
else if age (du ti - ti+1) < 20 days
        tempI ::= 0.65
else
        tempI ::= 0.50

if acceleration(WS du ti - ti+1) > a
        tempI ::= tempI * 50%

if average(WS du ti - ti+1) > s
        tempI ::= tempI * 10%
if average(WS du ti-1 - ti) > s
        tempI ::= tempI * 20%
if average(WS du ti-2 - ti-1) > s
        tempI ::= tempI * 50%
if average(WS du ti-3 - ti-2) > s
        tempI ::= tempI * 80%

I(COdu ti - ti+1) :: = tempI
```

Figure 8. Computing data importance with dependency.

At this point, we can generalize the constraints on the data importance computation as pertaining to two categories: (i) internal constraints and (ii) external constraints.

External constraints are caused by factors over which input data has no effect. Such factors are data age and inherent interest in the data depending on the exact purpose of the data collection.

Internal constraints represent inter- and intra-data dependencies. In the case presented above, increased wind renders CO_2 concentration less relevant and less usable; there can be events which affect the importance of data at any point along the timeline.

4.8 General implementation architecture

In this section, we present the general building blocks of the data collection process, as well as the prioritized data reduction process. While these processes act on the same data, they run in parallel as uncoupled processes.

Figure 9 presents the control steps for the data collection. Figure 10 presents the steps involved in the data reduction process. The black arrows denote sequence of steps. The red arrows denote control.

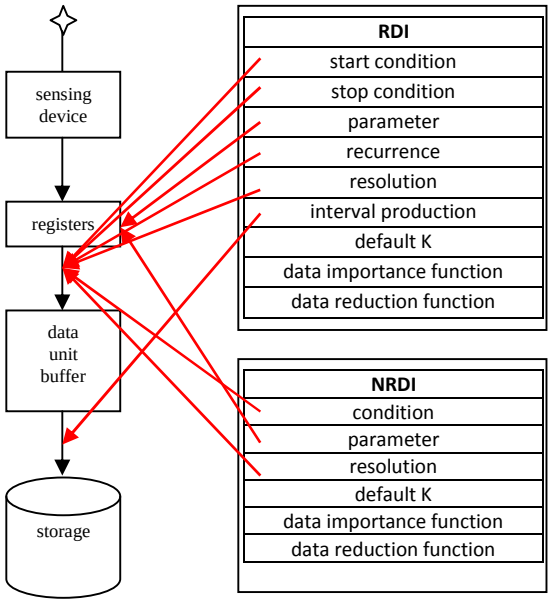


Figure 9. Storage management as per instance specifications.

In Figure 9, we have the physical sensors which make actual measurements. They have associated registers where the data readings are placed. This happens whether data is being collected or not. At this point, depending on the running condition of the recurring data instances, the

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

recurrence interval, and the resolution, the data is captured into a buffer. Enough data is accumulated to construct a full data unit. When complete, the data unit has its importance calculated and is placed into the storage. In case we are following an optimistic approach to data reduction, as described in [12], then we compute the data importance of the data unit in the buffer in order to compare with stored data. In case the data unit in the buffer is the one with lowest importance, then it will be the one to be reduced.

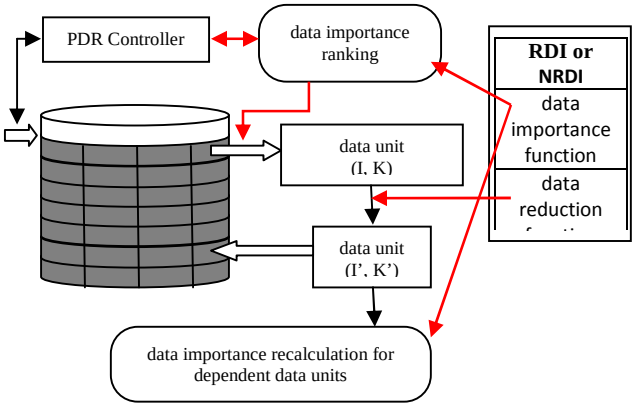


Figure 10. Data reduction based on data instance specifications.

Figure 10 only shows a single sensing unit and a single data instance. In deployments, there can be several sensing units for different parameters, and several data instances that are collected. There can be different instances collecting from the same sensor, but with different specifications, such as resolution for example.

In Figure 10, the flow of data reduction processing is presented. The PDR Controller continuously monitors the amount of data in the storage. If the levels cross a given threshold, then data importance ranking is consulted to identify the least important data. This data unit maps to a specific data instance which carries a data reduction function as well as an importance computation function. After a reduction, the data unit is written back to memory.

V. DATA HANDLING

5.1 Multi-prong consideraitons

5.1.1 Energy considerations

When encountering limited data storage space, a node has two options, (i) to reduce the size of the data by applying some primitives, or (ii) to relocate some data. In this section, we consider the factors that affect the decision.

Negotiations to find an appropriate host for data require energy. As the packets involved in negotiations don't contain data, they are fairly short so they don't consume much energy. The energy cost needed for the transfer of the data depends directly on the size of the data transferred. The node needs to save enough energy to eventually transfer the entire contents of the data to the sink node once the sink node becomes available.

Energy is wasted if a portion of data is relocated only to be transferred to the sink shortly after. In the case of unreliable sinks or sinkless deployments, expected access to sink is hard to compute. It can be done with statistical data, but reliability can vary. If the sink operates on a predefined schedule, it is feasible for this to be taken into consideration. A similar case to a predefined uptime schedule is a mobile sink. With some uncertainty, the sink will travel along a route with high probability of being in specific areas at certain times.

5.1.2 Data considerations

There are characteristics that make data appropriate for relocation. The size is one factor, as we want to have a sizable impact as a result of investing energy in the negotiation part. Other factors relate to the dependency between data units.

The importance calculation function should depend on self values or on time only. In that way, we can preserve the importance calculation function after the data has been moved. In some cases, the importance of a data unit depends on other data units of the same collection. It is conceivable that all the data units with importance interdependence relations be subject to a data transfer as a whole.

The reverse is also true, i.e., the data units to be transferred are used in computing the importance level of other data units. Bundling everything in the data to be relocated is an acceptable solution.

A special case is the situation where yet uncollected data can affect the importance of data units already collected and tagged for relocation. Such data is best not moved as this would imply changed to the importance calculation function.

The parameter K was introduced in Chapter 4 as a compensation factor to affect the importance value for data units that have undergone data reductions. A positive K will increase the importance of post reduction data, while a negative K will decrease it. The data that is considered for a move needs to have a positive K so as to make it through potential data reductions it will undergo on the receiver node.

5.1.3 Receiver node considerations

The conditions of the receiver node affect how the received data will be handled on that node. The potential recipient of the data needs to have a good situation of its storage space as well as a good outlook for future situations. The recipient also needs to have a good amount of energy left so as not to compromise the long term survival of the data being sent.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

The number of active collections and the amount of data they produce is an important factor. If there are many collections going on, the space on the receiving node is quickly filled up. Even if these collections produce low importance data units, a large number of ongoing collections generate lots of data reductions and data relocations. We want to avoid causing such events.

Inactive collections are collections whose start condition is not met. Some of these are predictable as to when collection actually starts, such as collections that only depend on time to be activated. Others depend on data values detected by sensors. Having a large number of inactive collections makes a node less desirable to act as a data recipient. These collections can start unexpectedly and produce data that competes for space with the relocated data.

Finally, the general importance profile of the stored data units are a factor in deciding if a node is a good recipient of relocated data. If most data on the receiver node is low importance, then the node is a good recipient of data. However, the nature of the importance functions on that node is to be taken into consideration. An event could trigger a recomputation that changes the data importance to higher values.

5.2 Sending vs. Reducing: the decision

The decision to favor data reduction vs. data relocation is subject to network's owner decisions. In this section, we provide a parameterized approach to handle this task. We specify three functions as follows:

- **SF(i)** specifies the fitness of a node *i* to act as a data source
- **RF(i)** denotes the fitness of a node *i* to act as a data receiver
- **DF({du})** denotes the fitness of a set of data units for a data transfer operation

We also specify a threshold value t such that:

*if (a*SF(n) + b*RF(m))*DF({du}) > t, then node n should initiate a data transfer of the {du} set of data units to node m, where a and b are parameters to give more or less weigh to each of the functions*

SF(i) is defined as a function of the source node's ability and opportunistic interest to spend energy on data movement related operations: negotiations and actual data movement. We define as LD(i) the percent of data stored on node *i* having an importance that is low and decreasing, that is, an importance smaller than a fixed value, decreasing in time, and a negative K. The K does not have to be negative for a fixed number of initial reductions, but it needs to be negative for remainders of the reductions or the reductions are quickly reaching a point where data is entirely dropped. Let *e* be the minimum residual energy required for a node n to send its data contents and *E* the current amount of energy in storage. The value of SF(i) directly related to available energy and to the lack of low and decreasing importance data units. We define SF(i) as follows:

$$SF(i) = k*(1-LD(i)) + j*(E/e) \quad (1)$$

where k and j are parameters to adjust the weights of the two factors.

RF(i) is defined as a function to quantify the potential receiver's node fitness to act as a receiver. With respect to energy, RF(i) behaves just like SF(i). With respect to the stored data profile, it behaves in an opposite manner. In addition to these two aspects, RF(i) needs to account for the effect of active collections as well as inactive data collections.

$$RF(i) = k*S*LD(i) + j*(E/e) - k*(Da + p*Di) \quad (2)$$

where Da is the storage debit of active collections and the Di is the potential storage debit of inactive collections. The constant p is used to decrease the impact of inactive collections.

DF(i) reflects on the fitness of data to be subjected to transfer. As a limiting factor, we have established that the bundle of data units must be self enclosed as far as importance calculation dependency. We propose the following formula for evaluating a data unit's fitness for transfer:

$$DF(i) = I*(trend(K) - k*|M - s|) \quad (3)$$

where M is the ideal data transfer size, s is the size of the current data slated for transfer, k is an adjustment parameter, and trend(K) is a function that evaluates the value trend of K in time and as potentially affected by reductions within the transferred data. The entire result relates to the overall importance I of the data bundle considered for transfer.

With these formulas, we have captured the essence of the decision making process of data transfer vs. data reduction. The approach is heavily parameterized as weighing different factors varies from one business case to another (Figure 11).

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

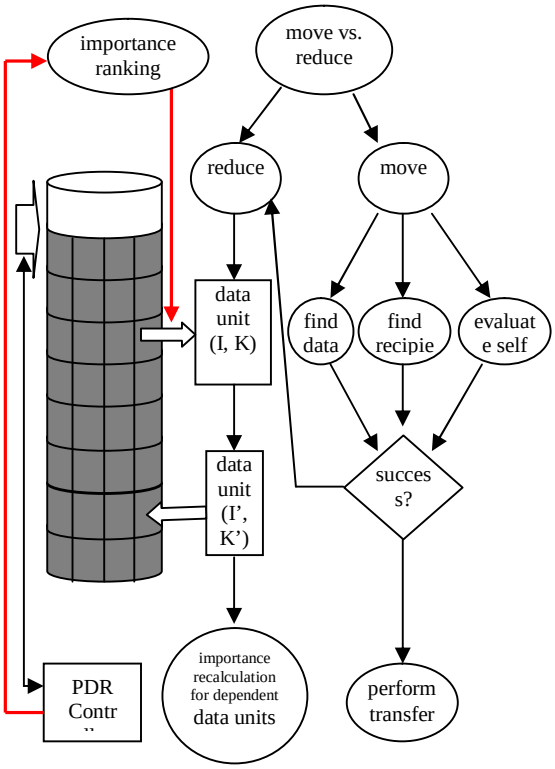


Figure 11. Data reduction via move or prioritized reduction.

We can assume that process to evaluate the source not for fitness to participate in a data send is straight forward: the value of the SF(i) is computed. Finding a recipient and finding appropriate data to send are more complex as they involve selecting from a pool of possibilities.

Finding data to transfer is a complex process. There can be a large number of data units, and therefore, considering all possibilities can be a lengthy process. We propose a heuristic on how to approach data targeting for relocation with the understanding that an exhaustive process may offer a better solution but at a higher cost. Data units that are targeted for transfer are incrementally added to the set of data for transfer as we seek to fulfill the requirements for self-enclosure for importance function evaluation. We stop as we reach a data size in the vicinity of M. In certain cases, we reach a dead end and can no longer have hope to find appropriate data to move.

The set of data units to be moved is incrementally increased in steps as follows:

- high importance and positive K data units on which no other data units' importance depends
    - data units' importance dependence limited to self values
    - data units' importance dependence limited to self values and time
- high importance data units with positive K whose importance depends only on other data units already collected (i.e., future data units collected have no impact on the importance of this data)
- recursively add the data units on which the above units depend for importance calculation
- as we reach a value close to M, we stop adding data units to the set of data targeted for a data transfer

We realize that recursively adding data units may be getting out of hand. If the minimum amount of self dependent data units amount to a very large amount of data, we can safely give up the idea of a transfer and opt for a data reduction.

The simplest approach to finding a recipient for data is to query the status of all reachable nodes, evaluate their RF(i) function and select the best option. Depending on the deployment, querying the immediate neighbors only may be artificially limiting the amount of potential recipient candidates for data relocation. Attempting to go the extra distance via additional hops requires the cooperation of other nodes along the path. These nodes need to commit energy reserves towards the data relocation process. Given that we have already selected the data that will be moved, there is a clear expectation as to what amount of energy would be required from the transient nodes during the data move.

Finding an appropriate receiver for the data follows the following steps:

- decide how many hops away we want the data to potentially go
- broadcast a message to the effect of finding a host, including the number of hops; if more than immediate neighbors are considered, include the data size to be able to obtain commitment along the transfer route
- reachable nodes reply with their current RF(i) computation
- select the best RF(i), compute (a*SF(n) + b*RF(m))*DF({du}) (see section 5.2.6) and if result is satisfactory, proceed with the transfer
- if the computation yields an unsatisfactory result, then data reduction is undertaken instead of transfer

5.2.1 Validation and calibration of heuristics

In the above section, we have proposed a series of heuristics to quantize the fitness of nodes to act as receiver or sender, as well as the fitness of data to be transferred. The proposed formulas are parameterized so they can be tweaked to several circumstances. In this section, we are going to evaluate the trends that the formulas generate under different circumstances depending on the statistical distribution of importance and compensation factor of stored data units

We recall that the formula to decide the fitness of a node to act as a sender is:

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

SF(i) = k*(1-LD(i)) + j*(E/e)

By its definition, the fitness function depends on the raw amount (not ratio) of high importance data and level of energy required to relay the entire data load towards the base station. We consider the scenario of different statistical distributions of data importance on a node, and in each case, we vary the amount of energy available. We consider the cutoff for LD(i) to be at the importance level of 0.5. Hence, in the graphs below, the blue bars represent (1-LD(i)).

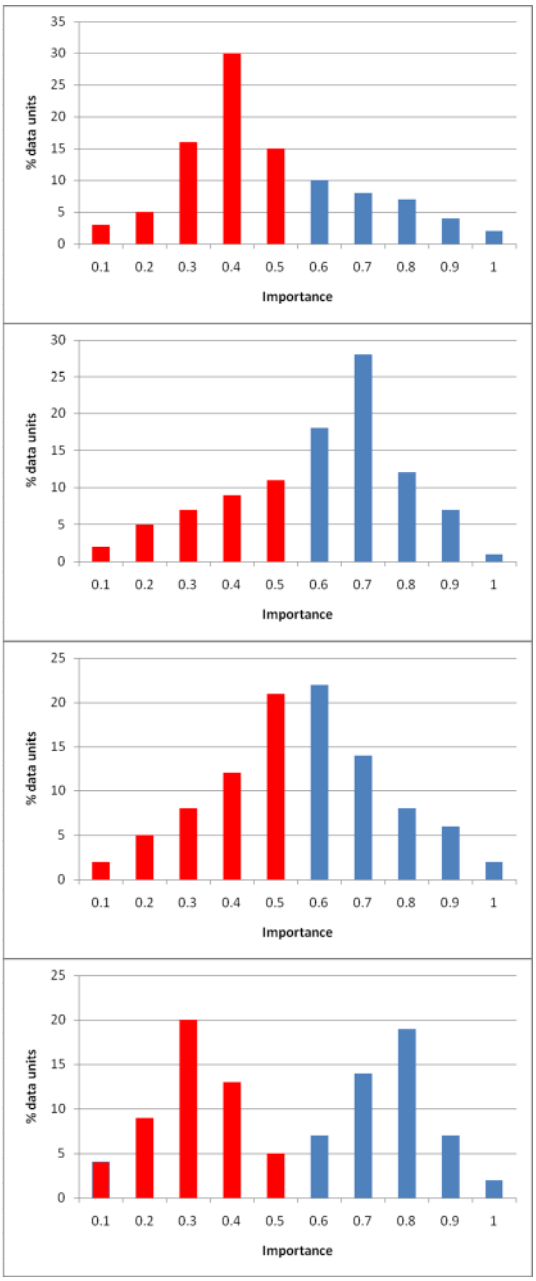


Figure 12. Data distribution scenarios for validation of SF(i).

The image above shows the scenarios considered with respect to the distribution of importance values. We have considered skewed, bell curve, and bimodal.

The initial proposal was that the amount of energy available would have a linear impact on the value of SF(i). This works well at points where the available energy levels E are within a few orders of magnitude of *e*, the energy needed for a full data transfer. At higher orders of magnitude (i.e orders of thousands, and above), the linear growth of j*(E/e) poses a problem as it identified a node as a "good node to send data" mostly based on its energy levels. If the nodes deployed fall in such a class, then the problem can be alleviated by using a logarithm to attenuate de higher orders of magnitude:

SF(i) = k*(1-LD(i)) + j'*log(E/e) (1')

Experiments with different LD(i) values only changes the balance between defining good source nodes via SF(i) and balanced by the ability to find receiver nodes via RF(i) defined in equation 2.

Along the same lines, simulations were run to further analyze the proposed formula for RF(i). We notice an important difference between SF(i) and RF(i): the formula for SF(i) depends on a percentage of storage while the RF(i) formula depends on the value of raw storage. The idea is that we don't want to put at a disadvantage nodes that, as per a deployment decision, have less storage space than others.

Analysis of RF(i) brings up a matter similar to SF(i) in that excessive energy levels can drive up the value of the function and artificially flag it as a good receiver node. In reality, this excess of energy may hide data space availability issues on the receiver node. it was found that in reality, aside from using a logarithm to attenuate the energy effect, a product rather than a sum gives a better assessment for a node's ability to receive data. Therefore, RF(i) as presented in equation 2 can be refined as:

RF(i) = j*log(E/e) (k*S*LD(i) - l*(Da + p*Di)) (2')

The evaluation of data fitness for transfer was defined in equation 3 as

DF(i) = I*(trend(K) - k*|M - s|)

Statistical evaluation on this formula with a linear relation between the trend of the compensation factor (trend(K)) and the deviation of the data size from an ideal size (|M-m|) is a good approximation. Since there are already strict conditions on assembling a set of data units for relocation, it would be counterproductive to discount the selected data set in case there are variations from the ideal size. Hence, we opt for a linear dependence on size deviations.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Additional simulations were conducted on the formula assembling the three heuristic items:

(a*SF(n) + b*RF(m))*DF({du})

We need both a pair of nodes to participate in the data transfer and a data set. If either of these items scores low with our evaluation, we want to prevent the data relocation. Hence, a product is appropriate.

For the purposes of the simulations in this section, we have compared trends in values. Specific parameter values can be assigned for very specific deployment scenarios, which must be validated on a case by case basis via simulation.

5.2.2 Relaxing data dependency for transfer

In the approach described above, we have restricted the data movement to data units whose importance function can be evaluated even after the move. This restricts the set of data units that can be targeted for relocation. While not going into the details of a full solution, we present in this section two approaches that can be used to mitigate the relocation of data units

A first approach is to simply replace the importance function for the relocated units with a version that does not depend on other data units. This basically changes the business case approach and may not be feasible while retaining a reasonable amount of intended resolution.

A more refined approach is to use expected values to replace the parameters inside of the importance function. It is less than desirable and requires some statistical approach in deciding what the expected values are. In the end, this approach may not yield a good compromise either because we may be interested in situations where unexpected values are encountered.

5.3 Redundancy

Up to this point, we have not addressed the issue of data redundancy. Having multiple copies of data in a wireless sensor network is important to balance the high incidence of node failure. In the framework presented in this thesis, having multiple copies of the same data runs counter to idea that we seek to make maximum use of a limited storage space.

5.3.1 Implementing redundancy

The implementation of redundancy is done at the specification of the data instanc. The additional parameters that we include relate to the number of that we want to store for that specific data instance. Hence, the data instance definition becomes:

Definition of an RDI with redundancy

RDI: (T_{start}, T_{end}, param, recurrence, resolution, compression, copies), where:
T_{start}: start time
T_{end}: stop time
param: the parameter being collected
recurrence: how often the collection is done
resolution: how precise the stored value is
compression: boolean stating if the RDI has been compressed
copies: integer stating on how many nodes the data needs to be stored
e.g., RDI(2009/12/06 16:43:23, ongoing, temperature, 60 seconds, 0.01C, FALSE, 4)

Figure 13. RDI definition with redundancy.

Defining an NRDI with redundancy

NRDI: (T, param, resolution, compression, copies), where:
T: recording time
param: the parameter being collected
resolution: how precise the stored value is
compression: boolean stating if the NRDI has been compressed
copies: integer stating on how many nodes the data needs to be stored
e.g., NRDI(2009/12/06 16:43:23, temperature, 0.01C, FALSE, 3)

Figure 14. NRDI with redundancy.

As data is collected one data unit at a time, these data units are stored locally (master copy) and copies are also stored on as many nodes as specified. These redundant nodes are not tied to one specific recurring data instance for example. The node collecting the data can select any other node to store the copies on. The process can be batched as node to node negotiations can be very costly for a small data segment at a time.

5.3.2 Importance calculation

We have seen how each data instance has a function that can be used to compute the relative importance of each data unit produced. When we deal with several copies of the same data unit, we need to specify a different importance value for each of the copies. Redundant copies have less importance than master copies. For this reason, in addition to the factors listed in section 4.8.4, we propose an importance decay function which is applied to the importance calculation of

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

redundant copies. For example, if we have a linear decay, the adjusted value of a data unit importance would be:

$$D(I(du)) = (1/r) * I(du), \quad (4)$$

where *r* is the redundancy level

As nodes collect data and go through data reductions, some of the redundant copies of specific data units may be dropped. This is normal as nodes prioritize data units as a function of importance. However, depending on the environments in which the redundant copies end up, higher level redundancy copies may disappear before lower level ones. In that case, we need to attach a message trigger to update the redundancy level of the copied of the dropped data unit. A limited flood mechanism can be used as it is not expected for data units to migrate very long distance from the point of origin.

Since redundant copies are targeted for transfer, the function used in calculating the redundant copies importance level must be computable on the receiving node. That restricts the computation function in a way similar to the restrictions imposed by load sharing. In consequence, if the data importance calculation function depends on other stored data, then redundant copies need to carry a completely new function, as opposed to a function simply affected by a decay function.

5.3.3 Load sharing under redundancy

The objective of redundancy is to protect the data from eventual sensor node failures. We therefore have several copies stored in the network. As a result of using load sharing to mitigate storage issues, we may at some point attempt to store more than one copy of a data unit on the same node. This should be prevented as it defeats the purpose of redundancy.

VI. Conclusion and Future Work

In general, there are no specific solutions other than stating that one needs a systematic approach to data aging. Current strategies do not address the problem from the perspective of sink node unavailability, but rather from the perspective of an intelligent WSN, which gives the possibility of executing queries through the entire network.

In this article, primitives were proposed in order to deal with increasing amounts of data stored by a sensor node. The scope is to have the basic mechanisms to gracefully discard lower importance data, or lose some flexibility and data resolution, for the benefit of higher importance data. The results presented in the example are encouraging. With a more complex business case, the algorithms can get more complex, but the overall objective remains the same.

The internal operation of a sensor node consumes little energy compared to the requirements for transmission. Hence, the energy issue has been disregarded with respect to operations taking place without transmission.

Data importance becomes a factor of the entire network. While the entire sensor network works together towards the data gathering task, there is competition amongst individual nodes for access to the storage available in the sensor network.

When going from the single node focus to the multiple node scenario, the positioning of the nodes is a factor. The distance matters for transmission power. If the nodes are placed at predetermined positions, the preferred communication links and energy requirements can be pre computed. If the placement of nodes is non-deterministic, self-organization algorithms are needed to guide communication paths for access to other nodes' storage.

We presented a methodology for prioritizing data processing in WSNs. In sensor networks with intermittent connections to a sink node, managing the limited storage size becomes a relevant task.

In our case, the constant availability of the sink node is not a given, which reflects the reality in which sensor networks often operate. We have proposed that the data be divided into manageable data units. These data units are evaluated and ranked in terms of importance. When it is time to reduce the data load, the lowest importance data units are subjected to a specified data reduction function.

An example was shown with the use case of a very large area CO_2 monitoring WSN serviced by a travelling data sink node. Several data were collected: CO_2 concentration, car traffic, wind speed and direction. A complete solution was proposed using only external constraints on the importance of the data. One of the data collections was further considered with respect to internal data constraints.

The expansion of the model to several node systems was studied. While some nodes may experience spikes in data production, other nodes may simply just collect low relevance monitoring data. We considered this case for collaborative storage and proposed solutions for data handling. The context in which importance evaluations and reduction functions operate so far is specific to the node that has produced the data. When we consider sharing other nodes' storage space, data that is being housed on a different node needs to be sent with proper instructions for handling. Internal dependencies can no longer be enforced; they need to be rephrased or simply dropped.

As a final step to having a robust solution, redundancy was addressed addressed. We have assumed smooth operation of the nodes with no equipment loss or malfunctions. This can very well be the case in a remote area. The degree to which redundancy is required and how

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

importance of redundant copies is computed need further exploration.

The decisions taken by the node while handling data make no assumption regarding scheduled, probabilistic, or statistical availability of the sink node. This additional information can affect the manner in which data reduction is applied, as well as the storage occupancy level at which data reduction processes are triggered.

Overall, the proposed solution and framework for a single node case are now robust enough to provide flexibility for future extensions.=Data reduction in unattended sensor networks with intermittent or non reliable connections is an important computation when considering data storage and data aging.

We proposed an optimization function using prediction to map the use of data reduction primitives, optimization parameters (K, I) and dependency constraints (contextual or bounding). The model considers a probability that a variation of a variable is correlated with a variation of another variable. A variant of average values can also be considered. A simple use case had shown the nature of dependencies and computation challenges for two correlated readings.

We have presented a proposal for prioritized data reduction in the light of a multi-node deployment. In such a case, it becomes feasible that a node facing storage space shortages look first at offloading some data to neighboring nodes. We have proposed a mechanism by which a node can quickly assess the situation and decide between the option of local data reduction or a date transfer.

The opportunity for redundancy was also evaluated. As wireless sensor networks are often deployed in unreliable and dangerous areas, node failure is high. Data redundancy reduces the possibility of critical data loss caused by node failures. We presented a mechanism to evaluate the importance level of data units that are redundant copies rather than master copies of the collected data.

Accurate evaluation of the model requires extensive simulations, where combinations of the primitives and data parameters are combined with various type of constraints. We estimate that finding some correlation patterns will favor the use of average values, leading to a reasonable computation effort.

REFERENCES

[1] Edgar H. Callaway, Jr. “Wireless Sensor Networks: Architectures and Protocols” , CRC Press, LLC, 2004

[2] Feng Zhao, Leonidas Guibas “Wireless Sensor Networks: an Information Processing Approach” 1st edition, Morgan Kaufmann, 2004

[3] Joon Ahn, Bhaskar Krishnamachari “Fundamental scaling laws for energy-efficient storage and querying in wireless sensor networks” The Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing, May 22-25, 2006, Florence, Italy, MobiHoc 2006:334-343

[4] Kyungseo Park, Ramez Elmasri “Query Classification and Storage Evaluation in Wireless Sensor Networks” 22nd International Conference on Data Engineering, April 3, 2006, Atlanta, GA, USA, ICDE Workshops 2006:35

[5] Norbert Siegmund, Marko Rosenmüller, Guido Moritz, Gunter Saake, and Dirk Timmermann. “Towards Robust Data Storage in Wireless Sensor Networks” Proceedings of Workshop on Database Architectures for the Internet of Things (DAIT). Birmingham, England, July 2009

[6] Joao Girão, Dirk Westhoff, Einar Mykletun, Toshinori Araki: “TinyPEDS: Tiny persistent encrypted data storage in asynchronous wireless sensor networks” Ad Hoc Networks (ADHOC) 5(7):1073-1089 (2007)

[7] E. Mykletun, J. Girao and D. Westhoff “Public Key Based Cryptoschemes for Data Concealment in Wireless Sensor Networks,” IEEE International Conference on Communications, 2006, Istanbul, Turkey

[8] Majid I. Khan, Wilfried N. Gansterer, Günter Haring “In-Network Storage Model for Data Persistence under Congestion in Wireless Sensor Network”. First International Conference on Complex, Intelligent and Software Intensive Systems, April 2007, Vienna, Austria, CISIS 2007:221-228

[9] Yongxuan Lai, Hong Chen, Yufeng Wang “Dynamic balanced storage in wireless sensor networks” 4th International Workshop on Data Management for Sensor Networks, September 2007, Vienna, Austria, DMSN 2007:7-12

[10] Davide Brunelli, Luca Benini, Clemens Moser, Lothar Thiele “An Efficient Solar Energy Harvester for Wireless Sensor Nodes” Design, Automation, and Test in Europe, March 2008, Munich, Germany, DATE 2008:104-109

[11] P. Ratanaworabhan, Jian Ke, M. Burtscher “Fast lossless compression of scientific floating-point data” Data Compression Conference, Salt Lake City, USA, DCC 2006. Proceedings 28-30 March 2006 Page(s):133 - 142 Digital Object Identifier 10.1109/DCC.2006.

[12] C. Dini and P. Lorenz, “Primitive Operations for Prioritized Data Reduction in Wireless Sensor Network Nodes”, Proceedings of the 2009 Fourth International Conference on Systems and Networks Communications, September 2009, Porto, Portugal, ICSNC 2009, pp. 274-280

[13] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, “Wireless sensor networks for habitat monitoring”, International Workshop on Wireless Sensor Networks and Applications, September 28, 2002, Atlanta, Georgia, WSNA’02, pp. 88-97

[14] E. H. Callaway, Jr. “Wireless Sensor Networks: Architectures and Protocols” , CRC Press, LLC, 2004

[15] M. I. Khan, W. N. Gansterer, and G. Haring, “In-Network Storage Model for Data Persistence under Congestion in Wireless Sensor Network:, First International Conferenece on Complex, Intelligent and Software Intensive Systems, April, 2007, Viena, Austria, CISIS’07, pp. 221-228

[16] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, “Collaborative storage management in sensor networks”, International Journal of Ad Hoc and Ubiquitous Computing, Volume 1, Issue 1/2 (November 2005), pp.47-58

[17] S. Tilak, W. Heinzelman, and N. Abu-Ghazaleh, “Storage Management Issues for Sensor Networks”, Poser at International Conference on Network Protocols (ICNP '03), Atlant, Georgia, November 2003

[18] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, “Data-centric storage in sensornets”, ACM SIGCOMM Computer Communication Review, Volume 33 , Issue 1 (January 2003), p. 137 - 142

[19] N. Siegmund, M. Rosenmuller, G. Moritz, G. Saake, and D. Timmermann, “Towards Robust Data Storage in Wireless Sensor Networks”, IETE Tech Rev [serial online] 2009 [cited 2010 Jan];26:335-40.
Available: http://tr.ietejournals.org/text.asp?2009/26/5/335/55280

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

[20] Y. Diao, D. Ganesan, G. Mathur, and P. Shenoy, "Rethinking Data Management for Storage-centric Sensor Networks", Proceedings of the Third Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar CA, January 7 - 10, 2007.

[21] S. Tilak, N. Abu-Ghazaleh, and W. B. Heinzelman, "Storage management in sensor networks". Mobile, Wireless and Sensor Networks: Technology, Applications and Future Directions, IEEE/Wiley. pp. 257-281.

[22] C. Dini and P. Lorenz, "Primitive Operations for Prioritized Data Reduction in Wireless Sensor Network Nodes", Proceedings of the 2009 Fourth International Conference on Systems and Networks Communications, September 2009, Porto, Portugal, ICSNC 2009, pp. 274-280

[23] C. Dini and P. Lorenz, "Prioritizing Data Processing in Wireless Sensor Networks", Proceedings of the 2010 Sixth International Confenrce on Networks and Services, March 2010, Cancun, Mexico, ICNS 2010, pp. 23-31

[24] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate Data Collection in Sensor Networks usingProbabilistic Models", Proceedigns of the 22nd International Conference on Data Engineering. ICDE 2006, Atlanta, USA http://www.cs.umd.edu/~amol/papers/icde06.pdf [Retrieved: August 20, 2010]

[25] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring", International Workshop on Wireless Sensor Networks and Applications, September 28, 2002, Atlanta, Georgia, WSNA'02, pp. 88-97

[26] C. Guestrin, P. Bodik, T.R., P. Mark, ans S. Madden, "Distributed regression: an efficient framework for modleing sensor network data", IPSN, 2004

[27] S.Madden, M.J. Franklin, J.M. Hellerstein, and W. Hoing, "Tag: a tiny aggregation service for ad hoc sensor networks" , SIGOP Oper. Syst. Rev. 36(SI):131-146, 2009

[28] S. Nath, P.B. Gibbons, S. seshan, and Z.R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks", Proceedings of the 2nd ACM Conference on embedded Networked Sensor Systems, SenSys 2004, Baltimore, MD, USA, 2004

[29] J. Hellerstein and W. Wang, "Optimization of in-network data reduction", DMSN, 2002

[30] M. I. Khan, W. N. Gansterer, and G. Haring, "In-Network Storage Model for Data Persistence under Congestion in Wireless Sensor Network:, First International Conferenece on Complex, Intelligent and Software Intensive Systems, April, 2007, Viena, Austria, CISIS'07, pp. 221-228

[31] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "Collaborative storage management in sensor networks", International Journal of Ad Hoc and Ubiquitous Computing, Volume 1, Issue 1/2 (November 2005), pp. 47-58

[32] Y. Diao, D. Ganesan, G. Mathur, and P. Shenoy, "Rethinking Data Management for Storage-centric Sensor Networks", Proceedings of the Third Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar CA, January 7 - 10, 2007.

[33] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model-driven data acqusition in sensor networks", VLDB, 2004

[34] I. lazaridis and S. Mehtotra, "Capturign sensor-generated time series with quality guarantees", ICDE, 2003

[35] S.M. Graham Cormode, M. Garofalakis, and R. Rastogi, "Holistic aggregates in a network world: Distributed tracking of approximate quantiles, SIGMOD, 2005

[36] Y.-Ae¨ . Le Borgne, et al., "Adaptive model selection for time series prediction in wireless sensor networks", Signal Process. (2007), doi:10.1016/j.sigpro.2007.05.015

[37] S. Tilak, W. Heinzelman, and N. Abu-Ghazaleh, "Storage Management Issues for Sensor Networks", Poster at International Conference on Network Protocols, ICNP 2003, Atlanta, Georgia, 2003

[38] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-centric storage in sensornets", ACM SIGCOMM Computer Communication Review, vol. 33, issue 1, January 2003, pp.137-142

[39] N. Siegmund, M. Rosenmuller, G. Moritz, G. Saake, and D. Timmermann, "Towards Robust Data Storage in Wireless Sensor Networks", IETE Tech Rev, 2009, vol. 26, issue 5, pp.335-40. Available: http://tr.ietejournals.org/text.asp?2009/26/5/335/55280 [accessed August 2010]

[40] M. Aly, N. Morsillo, and K. Pruhs "Zone sharing: a hot-spots decomposition scheme for data-centric storage in sensor networks" Second International Workshop on Data Management for Sensor Networks, DMSN 2005, Trondheim, Norway, pp.21-26

[41] Y. Lai, H. Chen and Y. Wang "Dynamic balanced storage in wireless sensor networks" 4th International Workshop on Data Management for Sensor Networks, DMSN 2007, Vienna, Austria, DMSN, pp.7-12

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org