Packet Reordering on Gateway as a Representative of Data Processing Techniques

Martin Johannes Waltl
Supervisor: Corinna Schmitt
Seminar Sensor Nodes: Operation Modes, Networks and Applications SS2012
Lehrstuhl Netzarchitekturen und Netzdienste, Lehrstuhl Betriebssysteme und Systemarchitekturen
Fakultät für Informatik, Technische Universität München
Email: martin.waltl@mytum.de

ABSTRACT
Wireless Sensor Networks (WSN) are becoming an important tool for data acquisition in environmental monitoring. Their advantages are cheap deployment, long system life time and little maintenance during operation. In many applications the WSN use a multi-hop communication protocol to transmit their data to the sink node. The arriving data is of poor quality suffering from duplicates, packet loss, device reboots and unsynchronized time stamps caused by local clock drifts. Some of these problems have not been solved by appropriate system designs yet. Therefore, it is necessary to correct these communication artefacts and enhance data quality in a post sequent step. Several algorithms to cope with these challenges and aim higher data accuracy already exist. In this work a model-based approach for the reconstruction of the temporal packet order in a multi-hop WSN is discussed. Furthermore, the challenges for WSN in harsh environments are presented by the PermaSense project, which observes permafrost changes in the alpine region. The multi-hop communication protocol Dozer is described in more detail to highlight the reasons for packet loss and duplicate generation.

Keywords
Environmental monitoring, WSN, Dozer, Data Processing

1. INTRODUCTION
In many technical applications and scientific fields the observation of different phenomena is a necessity for the understanding of processes and their interactions. The analysis of these data delivers new knowledge about the phenomena or is the basis to control technical systems. Wireless Sensor Networks (WSN) are one option to support the collection of these information. Advantages of WSN are their unattended operation over a long period of time, flexible adoption of the sensor nodes, cheap deployment and operation without human interaction.
WSNs consist of sensor nodes, which are equipped with a short-range radio. The sensor nodes acquire the measurements using their sensor equipment and transmit the data to a specified sink. The communication can ether be directly from a sensor to the sink node, or indirectly using multi-hop communication over several nodes. At the sink node all data is collected and stored for further data processing [2]. Some examples of sensor networks are the monitoring of heritage buildings [3], data center monitoring [10] and environmental monitoring of permafrost changes in the Swiss Alps [1].
Environmental monitoring is the continuous observation of natural phenomena, mostly studied over a long period of time ranging from days up to several years. It can support researchers and scientists with reliable information to verify existing models or gather data for future predictions. The collected data serves as basis for statistical analysis and optimization of existing models. Schmitt and Osenberg state that the major aim of environmental monitoring is to support decision making for economic, political and social authorities [13]. The high data quality of these scientific measurements is essential for more quantitative and qualitative conclusions. Furthermore, the analysis of environmental data allows to identify the impacts of natural disasters and human interventions to the environment. A more detailed introduction on environmental monitoring and state-of-the-art developments can be found in the journal 'Environmental Monitoring and Assessment' (Springer). [1]
WSN are one opportunity to support environmental monitoring. An appropriate WSN deployment accomplishes the acquisition of reliable data over its operational period, hence allowing the sensing of long-term evolutions. The field of applications for environmental monitoring is extremely broad and covers many areas of research. This work deals with the permafrost phenomena in the Alps, which is one example of environmental monitoring by a WSN. The PermaSense project is a project that investigates permafrost by the deployment of a WSN in the Swiss Alps [1]. Its architecture, deployment, data acquisition process and challenges in the harsh alpine environment are discussed in the following. Besides that, the problems within a multi-hop communication WSN like packet loss, duplicate generation and disordered arrival of data packets at the sink node are described. To provide valuable measurements for appropriate analysis it is necessary to account for these communication artefacts. One solution is a model-based approach presented by Keller et. al in [7] to reconstruct the temporal order of packets.

The outline of this seminar paper is as follows: The next section contains some background information on multi-hop communication and introduces the Dozer protocol. Section 3 gives an overview of the PermaSense project and Section 4 presents a model-based approach to enhance data quality by correcting for the artefacts of multi-hop communication. The last two sections present related work on data processing techniques and a conclusion.

[1] http://www.springerlink.com/content/0167-6369/open/

doi: 10.2313/NET-2012-08-2_05

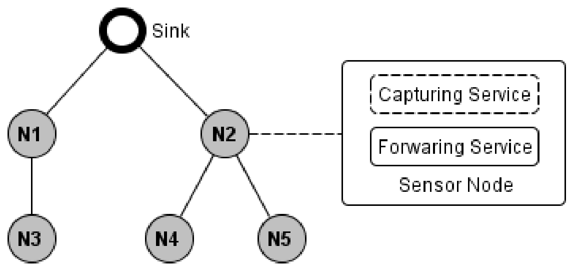

Figure 1: Example of a tree architecture for a WSN. The sink is the root of the communication tree and receives data from each node. Sensor nodes act as forwarding node for their children and can optionally perform data capturing [2].

2. BACKGROUND INFORMATION

Sensor networks for environmental monitoring are often distributed over a large area. One possible WSN architecture is a tree as shown in Figure 1. The network consists of sensor nodes that collect data and transmit it to one sink node, which is the root of the tree. Wireless sensor nodes are equipped with a short-ranged radio to send their data over the network. In many applications only a few nodes have a direct connection to the sink node, due to the large network expansion and physical obstacles that prohibit a direct communication. The other nodes are indirectly connected over other nodes, which forward the data to the sink. The latter is described as multi-hop communication. Consequently, the sensor nodes provide a sensing and forwarding functionality.

2.1 Dozer communication protocol

Dozer is a data gathering protocol developed for WSN with a multi-hop scenario [2]. It is designed to fulfil the demands of environmental monitoring applications which includes self-stabilizing properties, enable reliable data transfer and is optimized to maximize the system life time. Particularly it provides the process of periodic data collection, can account for topology changes and is optimized for ultra-low power consumption of the sensor nodes. The topic of minimizing power consumption is essential for WSN, because in many scenarios the nodes are battery powered and must endure several years without maintenance.

The Dozer system constructs a data gathering tree, with one sink node as its root. All other sensor nodes perform data acquisition and forwarding of network packets towards the sink. If a network node fails or is temporally unavailable, Dozer handles local recording of the data and rebuilds the gathering tree. When the node reconnects to the network, it is integrated again and the buffered data is transmitted to the sink node [2]. The Dozer protocol is described in more detail, because it is used by the PermaDAQ architecture in [1] and fits the assumptions in [7] to perform the introduced post processing technique.

2.2 Dozer system architecture

In the Dozer system architecture each sensor node is a child and parent node simultaneously. As parent node, it receives packets from its children and forwards them to its parent, until the packets arrive at the sink node. As child, the node

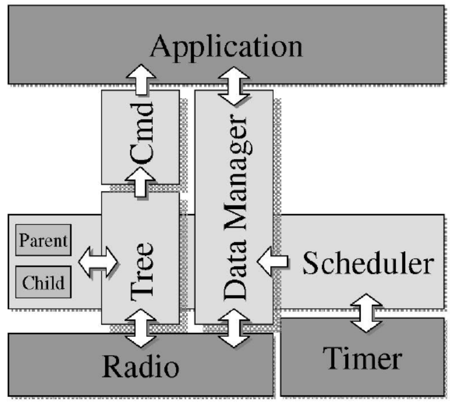

Figure 2: Architecture of the Dozer protocol. The four main components are represented by the light gray boxes. The arrows show the command flow between the different components [2].

sends data packets to its parent. Each node has a packet queue that is shared by its own data packets and the packets from its children. The sending process is designed after the Time Division Multiple Access (TDMA) mechanism, where each child is assigned to a time slot to transmit its data to the parent. In this design, the nodes only control the communication to their adjacent nodes and act locally in the network. The reason for this decision is that a global TDMA mechanism for the whole network would be too expensive concerning energy consumption and system life time. As a consequence, each node handles two schedules: *a*) a schedule for the transmission to the parent node *b*) a schedule to administer its children and assign their time slots.

Figure 2 shows the architecture of the Dozer system with the four main components represented by the light gray boxes. The *Radio* and *Timer* component are hardware elements of the sensor node. All data packets are transmitted using the *Radio* component. The *Timer* module is an essential part, because it delivers the interrupts for the precise execution of the parent and child schedules. The *Application* component represents higher layer applications of the sensor node. Dozer operates using a periodic TDMA frame transmission between all adjacent nodes. This frame starts with a beacon, that is used as synchronization for the children and allows them to compute their transmission time slots. The four main components are explained in more detail.

The **Tree Maintenance** component is responsible for the building of the gathering tree (see Figure 1). When a sensor node is powered up, the first action is to integrate itself into the network. In case of a connection loss (i.e. node is covered with snow) it iteratively tries to reconnect to the network. The reconnection is power consuming and according to strict power requirements this is only done spontaneously. Consequently, the reintegration of a sensor node can take a significant amount of time. In this state, the sensor nodes performs local storage of the acquired data to counteract data loss.

The management of the two time schedules *parent* and *child* is handled by the **Scheduler** component. As parent it accepts requests from child nodes and assigns them to a time slot. Each TDMA frame has a limited amount of time slots,

doi: 10.2313/NET-2012-08-2_05

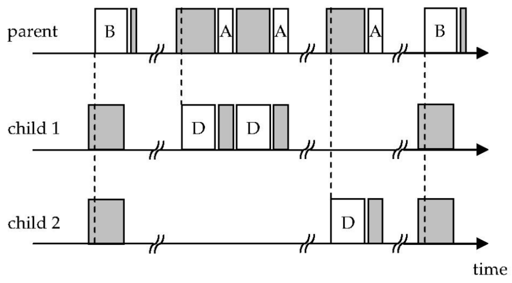

Figure 3: Example for message reception at a parent with two children. The upload slots for the current frame are determined with the beacon (B) as reference. All data messages (D) are acknowledged (A) to ensure data integrity [2].

which is equal to the maximum number of children a node can serve. On the other side as child, the node listens for the parents' beacon and computes the time for its assigned time slot when to send its data. The beacon is also used to synchronize to the parent. Due to the periodic event of the TDMA frame, the children can compute the transmission time slot and stay in idle mode at very long time to reduce power consumption and extend system life time.

Another goal of Dozer is to ensure data integrity. Therefore, each data packet needs to be acknowledged by the parent node. The component **Data Manager** controls and supervises the transmission process of the packets. Figure 3 shows an example communication process for one parent and two children. The parent node starts the TDMA frame with the transmission of a beacon, which signals the start of the frame. The beacon is time stamped at each child node and used as synchronization to compensate for local clock drift. Based on this time stamp each child computes its transmission time slot, which was negotiated at connection establishment. When the scheduler signals the beginning of the upload slot, the Data Manager tries to transmit queued messages. The packets are acknowledged by the parent to ensure data integrity and account for transmission failures. In case of a transmission failure, the node stops sending and makes a new sending attempt in the next time slot. Although, the node could retry in the same slot, this behaviour tries to overcome temporal interruptions of the radio channel. Nonetheless, data loss can occur, but these situations are explained in more detail in Section 4.

The main data flow is specified from the sensor nodes to the sink. In many applications it is desirable to convey control information from one to several nodes. Therefore, the **Command Manager (Cmd)** component provides a backward channel via the beacon messages. Control information are transmitted within the beacon signals downwards the gathering tree. Every node forwards the control information of a parent beacon in its own beacon, thus the information reaches every node in the network. This design allows application to specify custom messages in the Dozer protocol system.

3. PERMASENSE

The PermaSense project has the goal to gather real-time environmental data at high quality on high-mountain permafrost in the Swiss Alps. Scientists want to investigate the permafrost situation in the alpine region and its response to the climate change. A more detailed introduction to this topic can be found in [5]. Besides that, the PermaSense project acts as a prototype for the deployment of WSNs in harsh environments. The alpine region is a favoured terrain as a benchmark for system robustness due to its difficult environmental conditions. PermaDAQ is the WSN system architecture within the PermaSense project to monitor permafrost changes at the Matterhorn (Switzerland, 3450m) [1]. Prior to this project there has been one deployment at the Jungfraujoch (Switzerland 3500m) in 2006/07 [15], which served as basis for the improvement of the PermaDAQ architecture. The aim of the PermaDAQ architecture was to develop a robust sensor network, that delivers high quality data in the mountain terrain for a three year period with unattended operation.

3.1 System requirements

The alpine region and the requirement of several years continuous operation demanded a new system design for the necessary WSN. Major challenges are *a*) the aimed unattended operation time of several years without physical repair, *b*) wide temperature range from -40°C to +60°C, *c*) survival in the harsh high-alpine environment (rock falls, avalanches, snow, ice, rime, lightning, storm) and *d*) the desired data yield of 99%. The sensor nodes are exposed to the harsh environment and can loose their connection to the gathering network, due to coverage with snow or other circumstances. In this situation, the sensor nodes can't deliver data to the sink node. Therefore, another requirement is to provide an autonomous storage capability of at least 6 months to compensate for sensor unavailability. The experiences from the prior project at the Jungfraujoch revealed three main design challenges, which are coped in the PermaDAQ system architecture [1]:

Precision Sensing – For precise measurements it is important to reduce the influences during the sensing and support the creation of exact time stamps. Accurate timing is important, because it delivers higher quality for data analysis and enables data recovery if the sensor needed to store the data locally. One finding from the first deployment was that the simultaneous operation of sensor equipment and radio transmission resulted in corrupted data. The new design needed to be more robust and must consider these effect to allow precision sensing.

Reliability in Harsh Environment – The severe conditions of the mountain region require a robust mechanical design and well durability of the components to ensure high data quality and permit long life time.

Energy Constraints – The long life time requirement can only be achieved by an efficient energy management of the sensor unit. Beutel et. al use the Dozer protocol to minimize energy consumption and a Li-$SOCL_2$ battery. This type of battery is optimized for slow discharge at low temperatures.

3.2 WSN architecture

Figure 4 displays the tiered architecture of the PermaSense project with sensors, wireless sensor network, base station and backend. All components are designed to operate independently to prevent system failure in case of a malfunction of one tier.

doi: 10.2313/NET-2012-08-2_05

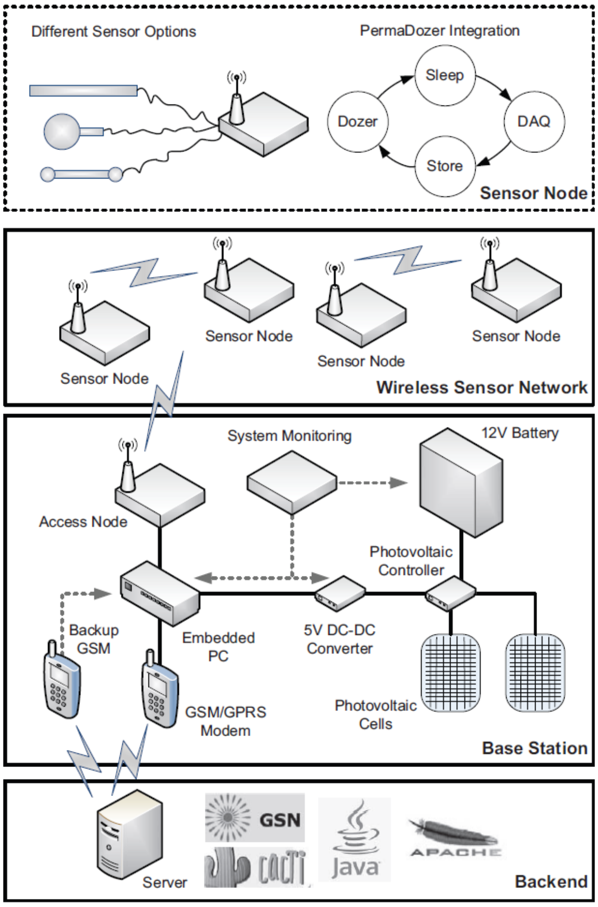

Figure 4: PermaSense architecture [1].

WSN – The PermaDAQ WSN is build of sensor nodes that have been developed for the measurements of permafrost changes. Each sensor node consists of a Shockfisch TinyNode[2], a self-developed sensor interface board (SIB) which runes the Dozer protocol and includes an SD card with 1GB external storage. The SIB is connected to the different sensor units, that perform the measurements for the permafrost observation. In the current implementation there are six different sensors [1]: *1*) a sensor rod for profiling of temperature and electrical conductivity in solid rock, *2*) crack meters consisting of a linear potentiometer for measuring movements, *3*) thermistor chains for profiling of temperatures in rocks, *4*) digital water pressure sensors to assess water flow in cracks, *5*) analog earth pressure cells for assessing ice stress inside lager crack and *6*) self potential sensors using analog differential conductivity measurements with electrodes mounted on the rock surface. The sensor node is powered by an Li-$SOCL_2$ battery. In the current deployment, 25 nodes are installed with a node spacing between 10m to 150m.

Base Station – The base station is an embedded PC platform, that is the sink of the WSN. It collects the system data and transmits it to the backend via an GPRS/EDGE module. The base station is powered by a solar cell system, because it requires more energy and can be constructed in a protective area compared to the sensor nodes.

[2]`http://www.tinynode.com`

Backend – The backend consists of a server which stores the retrieved data in a data base, running a global sensor network (GSN) application. The GSN is a flexible network management software for WSN data and provides a graphical overview.

3.3 Operation and data acquisition

The sensor nodes run the Dozer system, which drives the data acquisition and transmission processes. For the PermaSense project Keller et. al developed an own implementation named PermaDozer that is optimized for their application. Figure 4 illustrates the periodic operation of the PermaDozer integration within the sensor node at the top. Dozer is based on a periodic duty cycle separated into data acquisition, storing, transmission and returning the system into sleep mode to minimize power consumption. The process in the PermaDozer is as follows: The node is in *sleep* mode until the timer initializes the data acquisition process (*DAQ*). When the data is obtained it is *stored* and the system returns to the normal dozer operation, where the data transmission is handled (*Dozer*). Afterwards the system switches back into *sleep* mode.

In the current implementation of the PermaSense project the sensing is performed every 2 minutes and the Dozer beacon is sent every 30 seconds. This activity schedule is optimized for little de-synchronization and energy constraints. Consequently, the beacon rate limits the sensing time to less than 30 seconds within two consecutive beacons. The DAQ is triggered every two minutes and must complete within 30 seconds, before the next beacon of the Dozer protocol arrives. Within the DAQ each sensor measures sequentially, because the simultaneous operation of two sensors would interfere and result in a lower data accuracy [1].

When the data acquisition is finished, the data is appended to the message queue and uploaded along the gathering tree by the Dozer protocol. As mentioned in the beginning, it is possible that nodes become disconnected from the WSN. In this case, the acquired data is locally stored on a SD card and uploaded when the connection is re-established. A deeper insight into the PermaDozer operation exceeds the scope of this work, but can be found in [8].

4. RESTORING TEMPORAL ORDER

The data quality of a WSN suffers from local clock drift, packet duplicates, node reboots and packet loss. Another problem of multi-hop routing protocols (like Dozer) is that the packets don't arrive in correct temporal order, because the packets can be routed along different paths or are locally buffered and delayed in time. Therefore, it is necessary to correct the received data by detecting duplicates and reconstruct the temporal order. This post processing step can be performed on powerful systems, which reduces the complexity of sensor nodes and their energy consumption. Therefore, a formal model describing the data acquisition and transmission process is required. Keller et. al developed in [7] a model-based approach to reconstruct the temporal packet order for a WSN with a tree architecture.

Another reason for data inaccuracy is that the data acquisition equipment suffers from noise, outliers and inaccuracy due to faulty calibration. This effects can be minimized by an appropriate design of the sensor unit, sensing process and detailed analysis on the sensor behaviour in the targeted region. In this work, the reconstruction algorithm doesn't rely on the sensor data, why the topic of calibration is not discussed further. The next subsection introduces the underlying system model, necessary for the model-based approach in Subsection 4.2. Section 4.3 presents a case study that evaluates the performance of the model-based approach on data of the PermaSense project.

4.1 System model

The specification of a formal model is a crucial part, because the temporal order is reconstructed on these assumptions. In general a WSN consists of sensor nodes that periodically send packets via a tree structure to the sink node (see Section 2). A packet is a tupel of five elements (o, s, p, $\tilde{t_s}$, t_b), where o is the node address, s the sequence number (bounded by s_{max}), p the payload, $\tilde{t_s}$ the estimated sojourn time and t_b the absolute arrival time at the sink. The sojourn time is the packet residence time within the network until it reaches the sink node. As a system assumption, the sink node has an absolute clock that does not suffer from any clock drift and restarts. On the contrary, the sensor nodes are committed to restarts and have a free running clock that suffer from local clock drift.

Nodes are exposed to two different kinds of restarts: *a*) a warm restart is caused by a malfunction of the operation system, initiated by the watchdog timer and resets the system clock and *b*) a cold restart is equal to a hardware restart, where the system clock and sequence number are reset and the message queue is emptied (data loss). After a restart the sensor node initiates data capturing immediately, hence the time difference between two consecutive packets can be lower than T. Cold restarts are one reason for packet loss. There is no global synchronization within the network. Every node provides two different services [7]:

Packet Capturing Model – The sensor nodes perform data capturing at a constant interval T, that is relative to the local clock. In each interval a new packet is generated with the data as payload and a sequence number, which is incremented afterwards. One issue of a sensor node is that it suffers from a local clock drift due to physical properties of the oscillator caused by i.e. temperature variations or fluctuating supply voltage.

Forwarding Model – In a multi-hop network, packets are conveyed along the parent nodes to the sink. Each packet is acknowledged by the parent node to prevent data loss. The system model assumes that the packets are stored in a message queue and forwarded after a sojourn time, which equals the residence time of the packet within the sensor node. When a packet i is send from a node e, the sojourn time of the packet is updated according to Equation 1.

$$\tilde{t_s}(i) = \tilde{t_s}(i) + t_{residence}(i, e) \tag{1}$$

The sojourn time received at the sink, represents the sum of the sojourn times of the passed nodes. It is only an estimation of the real sojourn time, because the time measurements are taken correspondingly to the nodes' local clocks. The actual transmission time of a packet between nodes is neglected, because it is significantly smaller than the residence time. At the sink, each packet is time stamped at receipt with t_b to estimate its generation time. Duplicates can occur if the acknowledgement for a packet does not arrive in time and the child node retransmits the packet.

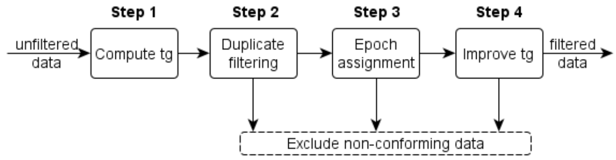

Figure 5: Steps of the model-based approach [7]. (t_g - generation time)

4.2 Model-based approach

The model-based approach to reconstruct the temporal order is based upon the previous system model and subdivided into four steps (see Figure 5). The result is a list of temporal ordered packets, excluding non-conforming packets [7].

Step 1: Compute packet generation time interval – The packet generation time is computed by subtracting the estimated sojourn time of a packet i from the packet arrival time at the sink node:

$$\tilde{t_g}(i) = t_b(i) - \tilde{t_s}(i) \tag{2}$$

This is only an approximation of the generation time, because the sensor nodes suffer from clock drifts and have a time measurement uncertainty of $(-\hat{t_u},0]$. The local clock drift is within the range $[-\hat{p}, \hat{p}]$ and $h * \hat{t_u}$ accounts for the worst-case error of timing uncertainties over h hops. Accounting for these effects we get an estimation interval for the generation time:

$$t_g(i) \in [t_g^l(i), t_g^u(i)] \tag{3}$$

where

$$t_g^u(i) = t_b(i) + \frac{\tilde{t_s}(i)}{1+\hat{p}} \tag{4}$$

$$t_g^l(i) = t_b(i) - \frac{\tilde{t_s}(i) + \hat{h} * \hat{t_u}}{1-\hat{p}} \tag{5}$$

Step 2: Duplicate filtering – The next step is to get rid of duplicates within the full data set D. Duplicate packets are determined by the same source address (o), same sequence number (s), equal payload (p) and overlapping generation times intervals. Duplicate packets can have different $\tilde{t_s}$, hence packets can take different paths to the sink. The problem is transformed to find the maximum independent set in a graph. A graph $G = (V, E)$ consists of vertices V that are marked as duplicates and their interconnections E. Two vertices v and w are connected by an edge $(v, w) \in E$, if they have overlapping generation time intervals:

$$(v, w) \in E \Leftrightarrow (t_g^u(v) \geq t_g^l(w)) \wedge (t_g^u(w) \geq t_g^l(v)) \tag{6}$$

A standard algorithm is used to find the maximum independent subset $I \subseteq V$. Therefore, the full data set D (unfiltered data) is separated into a subset of duplicate packets and the corresponding graph is constructed. In this graph the maximum independent set I is computed. All packets that correspond to a vertex $z \in V \setminus I$ are marked as duplicates and removed. The resulting duplicate free data set is used for further processing. This simple graph based selection algorithm can also reject non duplicate packets incidentally.

But the superior goal to obtain a duplicate free data set is weighted more important than incidentally removed packets.

Step 3: Epoch assignment – The sequence number s is bounded by the maximum value s_{max} with $0 \leq s < s_{max}$. An epoch contains maximum s_{max} subsequent packets. The reasons for epochs are an overrun of the sequence counter ($(s+1) \bmod s_{max}$) and cold restarts, which reset the sequence counter to zero. Epochs are labelled with incremental index $e \in N$ and each packet i is assigned to exactly one epoch $e(i)$. The following statement determines the epoch assignments of packets with different sequence numbers. If the packets have similar sequence numbers, they belong to different epochs, because the data set is duplicate free.

$$t_g(k) < t_g(l) \Leftrightarrow (e(l) < e(k)) \vee ((e(k) \equiv e(l)) \wedge (s(k) < s(l))) \tag{7}$$

Packets from the same epoch share the same epoch centre T_c. The epoch centre is computed by subtracting the sensing interval T multiplied by the sequence number of the packet i from its generation time:

$$T_c(i) = \tilde{t_g}(i) - s(i) * T \tag{8}$$

$T_c(i)$ is the estimated epoch centre, because $\tilde{t_g}$ is only an estimation. Epoch centres for packets k and l of the same epoch are therefore not equal, but close to the real epoch centre T_c. ΔT is the allowed window of the accepted epoch centre for packets within the same epoch.

$$e(k) = e(l) \Rightarrow |T_c(k) - T_c(l)| \leq \Delta T \tag{9}$$

If nodes are not exposed to restarts, two epoch centres are always $s_{max} * T$ apart. On account of the local clock drift and node restarts, the difference between two epoch centres is $< s_{max} * T$. Therefore, Keller et. al specify the epoch assignment as follows [7, 5.3]:

> *All packets whose "epoch centers" are close enough are assigned to the same epoch, whereas packets whose "epoch centers" have a large distance are assigned to different epochs.*

This definition is determined by the equations (9) and

$$|T_c(l) - T_c(k)| > \Delta T_c \tag{10}$$

where the virtual epoch centres of the two packets k and l must be at least ΔT apart. All packets that violate (9) or (10) are marked as non-conforming, which is illustrated by Figure 6. The choice of the parameter ΔT is crucial, because it determines the tolerance of the epoch assignment.

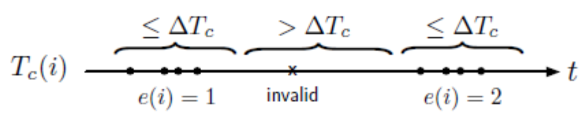

Figure 6: Two epoch centres must be at least ΔT apart [7].

Another problem is caused by two cold restarts happening shortly one after another. In this case, packets from two disjoint epochs can be within the same epoch centre interval. Therefore, the minimum distance between two epoch centres must be at least $2 * \Delta T$ apart, to guarantee a sufficient separation of epochs. Based on these definitions the algorithm for the epoch assignment comprises the following steps:

1. Packets exceeding a maximum $t_{s,max}$ are removed.
2. Compute the virtual epoch centres for all packets (8).
3. Remove packets that violate the definitions (9) or (10).
4. Assign a unique *id* to the remaining packets, reflecting the temporal order of the packets by equation (11). Afterwards remove all packets that have the same id, which is caused by two consecutive restarts that are separated less than $2 * \Delta T$.

$$id(i) = e(i) * s_{max} + s(i) \tag{11}$$

The result of this algorithm is that each packet has a unique id and the filtered data set satisfies:

$$t_g(k) < t_g(l) \Leftrightarrow id(k) < id(l) \tag{12}$$

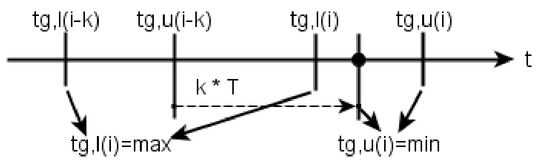

Figure 7: Illustration of forward reasoning [7].

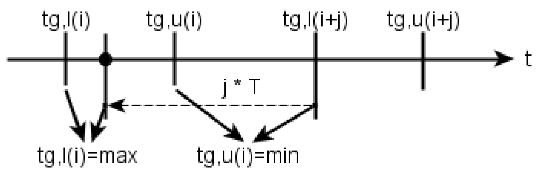

Figure 8: Illustration of backward reasoning [7].

Step 4: Forward/Backward reasoning – The generation time is limited by an upper and lower bound after step 1. The additional sequence information obtained in step 3 and the generation time interval T can be exploited to tighten this interval. Therefore, Keller et. al use forward and backward reasoning, where k is the number of generation time intervals between two consecutive packets in the ordered data set. The forward and backward reasoning are illustrated in the Figures 7 and 8. It is essential that the data set is ordered accordingly $id(i) < id(i+1)$. For each packet i the tightening mechanism comprises the following steps. First the backward and forwarding reasoning is executed, which return a lower and upper bound for $t_g(i)$. Next the upper and lower bounds are combined accordingly:

$$t_g^l(i) = max(t_g^l(i-k), t_g^l(i), t_g^l(i)_{backward}) \tag{13}$$
$$t_g^u(i) = min(t_g^u(i)_{forward}, t_g^u(i), t_g^u(i+j)) \tag{14}$$

If the packet violates $t_g^u(i) < t_g^l(i)$ it is non-conforming and removed from the list. This process is executed for each packet. The result of this step is that the packets feature tightened generation time intervals within the ordered set [7].

doi: 10.2313/NET-2012-08-2_05

4.3 Evaluation

Keller et. al compare their model-based approach to an simple heuristic to evaluate its performance [7]. As data basis they use data sets from the PermaSense project that was gathered by the WSN described in Section 3. The data sets comprise measurements from July 2008 until May 2010, which represent three deployment phases. The three phases have different system behaviour: (Phase A) highly non-conforming system behaviour, (Phase B) sensor nodes subject to a high frequency of unplanned warm restarts and (Phase C) high accumulation of transmission delays of several hours to days by one third of the sensor nodes. They used three metrics to evaluate their model-based approach in comparison to the simple heuristic: *1*) Packet acceptance rate, *2*) correctness of retrieved packet sequence and *3*) improvement of generation time intervals.

In their first analysis they applied the model-based approach and the simple heuristics to retrieve the temporal order by each one. Afterwards they compared the found sequence violations for both approaches and against ground truth data, which was obtained from the SD cards of the sensor nodes. Both the model-based approach and the simple heuristics deliver very good results with little sequence violations and a high packet acceptance rate, for the last two phases B and C that represent normal system behaviour. In the evaluation of phase A the model-based approach resulted in a much better reconstruction of the temporal order and a higher packet acceptance rate than the simple heuristics.

The second analysis focused on the improvement of the generation time intervals. Since the system behaviour in phase A is not according to the assumed system model, only phase B and C were used for this evaluation. The model-based approach was again applied on the two last phases and the improvement of the generation time was evaluated. The results are that forward and backward reasoning lead to a reduction of the interval in 90% of the packets. In 75% of the cases, the generation time interval could be reduced by at least half, compared to step 1. On the absolute scale, the initial generation time was decreased by up to 100s and the mean was reduced from 8.1s to 2.8s. Concluding, the authors state that forward and backward reasoning deliver a considerable improvement in general and compensate for introduced worst-case uncertainties based on the clock drift and the missing global synchronization.

5. RELATED WORK

WSN are distributed event-based systems that differ from conventional communication systems by the requirement of energy constraints, a favoured communication flow from many-to-one (sensor node to sink) and low bandwidth demand. The whole data communication and data aggregation process is designed to optimize energy consumption and ensure a long sensor node life time without maintenance [9].
Sensor networks suffer from several drawbacks like clock drift, temporary drop outs of sensor nodes, restarts and a dynamic network topology [1]. One resulting problem is that the temporal order of the arriving packets does not agree with the logical order of reception at the sink node. Therefore, counter measures are necessary to achieve the correct temporal packet order. There are two basic approaches to cope this problem [7].
The first approach is to establish a global time base over the whole network. If all network nodes are synchronized the sensor nodes can time stamp each measurement correctly. This time stamp allows a simple reordering at the data sink, although if packets arrive unsorted. Nonetheless, this approach requires a synchronization protocol for the WSN. Keller et. al state in [7] that introducing a global synchronization scheme is cost intensive, because it increases the node complexity and reduces the battery life time. Another problem is, that the sensor nodes can loose the network connectivity for a longer period and would not receive timing updates, hence suffer from local clock inaccuracies again.
The Flooding Time Synchronization Protocol (FTSP) is a synchronization protocol designed for WSN. It operates by periodically flooding the network with synchronization messages and should be robust against link and node failures [12]. Werner-Allen et. al implemented the FTSP in their sensor network to perform volcano monitoring. They reported an inaccurate synchronization by FTSP, caused mainly by the unstable network topology, which resulted in time offsets from several hours [16]. This additional contextual information is transmitted with the sensor data and enables to reconstruct a global time stamp for this measurement at the sink. However, Elson and Römer state that a global time synchronization performed by Network Time Protocols (NTP) is inappropriate for WSN. In their opinion the sensor nodes should maintain their own timescale. They propose an application depended network design to minimize these negative influences and state to use domain knowledge to improve data quality [4]. Another possibility to establish a global time scale is to exploit application domain properties and obtain timing information i.e. by microseismics [11] or measurements of the sunlight [6].
The second approach is to perform packet filtering at the sink node. Post processing can be performed on powerful machines without energy constraints and reducing sensor node complexity in return. On the other site, an accurate model of the data aggregation process is necessary to allow an appropriate reconstruction of the temporal order. In general, a system model includes many simplifications, because modelling the exact behaviour would result in too complex models. Traditional post processing is performed on application domain knowledge for packet filtering and outlier detection. In these cases, data cleaning and filtering relies only on the measured sensor data [14].

Keller et. al propose a two-staged post processing alternative to enhance data accuracy [7]. In the first stage, filtering is only performed on packet header information to reconstruct the temporal order using packet sojourn time, sequence counter or various time stamps. This ordered data set serves as input for the second stage, which performs filtering and outlier detection on the gathered sensor data in combination with application domain knowledge. The major difference from [7] to previous methods is, that they merely consider application header information of the packets instead of pure sensor data. Therefore, Keller et. al developed a formal model of the data acquisition and transmission process in a multi-hop WSN. The formal model is applied on the unfiltered data to reconstruct the temporal packet order. Their evaluations reveal promising results compared to a simple heuristics and real packet order. Moreover, their model-based approach delivers satisfactory results if the system behaviour is violated [7].

doi: 10.2313/NET-2012-08-2_05

6. CONCLUSION

Wireless sensor networks are an interesting technology to perform monitoring of areas and processes. Environmental monitoring is one possible field of application, where WSN can support data acquisition. In this work, the PermaSense project was introduced as an example of a WSN for environmental monitoring. The focus was on the challenges of WSN in harsh environments and extreme weather conditions to ensure reliable data acquisition. Its system design and the deployment showed that the PermaDAQ architecture can endure the challenges in the alpine region. Furthermore, the multi-hop protocol Dozer was briefly introduced to highlight the reasons for duplicates and packet loss. The second part of this work covered the reconstruction of the correct temporal packet order using a model-based approach from [7]. This approach relies on a system model that specifies the properties of the WSN, as well as the data acquisition and transmission process. The model was applied on the unfiltered data sets of the PermaSense project and compared to a simple heuristics. The evaluation showed a good packet acceptance rate and that the model-based approach is a suitable method to reconstruct the temporal order of packets. Additionally, the model-based approach delivered more accurate results in contrast to the simple heuristics, though the data was obtained under non-conforming system behaviour.

The focus of this work was to give a short overview on the communication artefacts of WSN and the necessity of post data processing techniques to improve data quality. The model-based approach developed in [7] was presented as one example for post processing to correct for these effects and reconstruct the temporal packet order. In addition to that, environmental monitoring in harsh environments and its challenges for WSN have been introduced by the PermaSense project, which relies on multi-hop communication.

7. REFERENCES

[1] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel. PermaDAQ: A scientific instrument for precision sensing and data recovery in environmental extremes. In *Processing of the International Conference on Sensor Networks (IPSN)*, pages 265 – 276, April 2009. IEEE Computer Society.

[2] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: Ultra-Low Power Data Gathering in Sensor Networks. In *Processing of the 6th International Symposium on Sensor Networks (IPSN)*, pages 450 – 459, April 2007. IEEE Computer Society.

[3] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon. Monitoring heritage buildings with wireless sensor networks: The Torre Aquila deployment. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, pages 277 – 288, Washington, DC, USA, 2009. IEEE Computer Society.

[4] J. Elson and K. Römer. Wireless sensor networks: a new regime for time synchronization. In *SIGCOMM Comput. Commun. Rev.*, 33(1), pages 149 – 154, January 2003.

[5] S. Gruber and W. Haeberli. Mountain Permafrost. In *Permafrost Soils*, Vol. 16 of *Soil Biology*, pages 33 – 44, 2009. Springer Berlin/Heidelberg.

[6] J. Gupchup, R. Musăloiu-E., A. Szalay, and A. Terzis. Sundial: Using sunlight to reconstruct global timestamps. In *Wireless Sensor Networks*, Vol. 5432 of *Lecture Notes in Computer Science*, pages 183 – 198, 2009. Springer Berlin/Heidelberg.

[7] M. Keller, L. Thiele, and J. Beutel. Reconstruction of the correct temporal order of sensor network data. In *10th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 282 – 293, April 2011. IEEE Computer Society.

[8] M. Keller, M. Woehrle, R. Lim, J. Beutel, and L. Thiele. Comparative performance analysis of the PermaDozer protocol in diverse deployments. In *36th Conference on Local Computer Networks (LCN)*, pages 957 – 965, October 2011. IEEE Computer Society.

[9] L. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops*, pages 575 – 578, 2002. IEEE Computer Society.

[10] C.-J. M. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao. RACNet: a high-fidelity data center sensing network. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, (SenSys)*, pages 15 – 28, New York, NY, USA, 2009. ACM.

[11] M. Lukac, P. Davis, R. Clayton, and D. Estrin. Recovering temporal integrity with data driven time synchronization. In *Proceedings of the International Conference on Information Processing in Sensor Networks, (IPSN)*, pages 61–72, Washington, DC, USA, 2009. IEEE Computer Society.

[12] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 39–49, New York, NY, USA, 2004. ACM.

[13] R. Schmitt and C. Osenberg. Detecting Ecological Impacts (1st Edition) - Concepts and Applications in Coastal Habitats. Academic Pr Inc, 1996.

[14] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. In *Wireless Sensor Networks*, Vol. 2920 of *Lecture Notes in Computer Science*, pages 307–322, 2004. Springer Berlin/Heidelberg.

[15] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin. Permasense: Investigating Permafrost with a WSN in the Swiss Alps. In *Proceedings of the 4th workshop on Embedded Networked Sensors (EmNets)*, pages 8 – 12, New York, NY, USA, 2007. ACM.

[16] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th symposium on Operating systems design and implementation (OSDI)*, pages 381–396, Berkeley, CA, USA, 2006. USENIX Association.