Andreas Reinhardt, Matthias Hollick, Ralf Steinmetz: *Stream-oriented Lossless Packet Compression in Wireless Sensor Networks*. In: Proceedings of the Sixth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2009), p. 1 - 9, June 2009. ISBN 978-1-4244-2907-3

Stream-oriented Lossless Packet Compression in Wireless Sensor Networks

Andreas Reinhardt, Matthias Hollick, Ralf Steinmetz
Multimedia Communications Lab, Technische Universität Darmstadt
Merckstr. 25, 64283 Darmstadt, Germany
Email: {andreas.reinhardt, matthias.hollick, ralf.steinmetz}@kom.tu-darmstadt.de

***Abstract*—In wireless sensor networks, the energy consumption of participating nodes has crucial impact on the resulting network lifetime. Data compression is a viable approach towards preserving energy by reducing packet sizes and thus minimizing the activity periods of the radio transceiver. In this paper, we propose a compression framework utilizing a stream-oriented compression scheme for sensor networks. It is specifically tailored to the capabilities of employed nodes and network traffic characteristics, which we determine in a characterization of WSN traffic patterns. To mitigate the inapplicability of traditional compression approaches, we present the Squeeze.KOM compression layer. By shifting data compression into a dedicated layer, only minor modifications to applications are required, while efficient data transfer between nodes is provided. As a proof-of-concept, we implement a stream-based compression algorithm on sensor nodes and perform an experimental analysis to determine the potential gains under realistic traffic conditions. Results indicate that our presented lossless stream-oriented payload compression leads to considerable savings.**

I. Introduction

In most wireless sensor networks (*WSNs*), energy budgets of nodes are tightly limited [1], necessitating the design of applications with increased awareness to their energy consumption. As radio transmissions are an inherent and crucial characteristic of WSNs [2], but current radio transceivers, such as the widely used CC2420 device, still expose static power consumptions of tens of milliamperes [3], permanent operation of the radio transceiver leads to quick depletion of the battery in both transmission and reception mode. This issue can be approached in several ways, reaching from energy-aware MAC protocols to highly application-specific data compression algorithms, with the common purpose of minimizing the period in which the radio transceiver is active. These local energy optimizations can be supplemented by network level approaches, such as data aggregation [4], [5] or coding by ordering [6], both exploiting spatial correlation of sensor data.

Low power MAC protocols, like the synchronized S-MAC [7] or the asynchronous B-MAC [8], perform duty-cycling of radio transceivers to reduce the energy consumption and thus increase node lifetime. On application level, data compression specifically tailored to the purpose allows high compression efficiency, even when operating on sophisticated data structures, such as low-complexity video [9] or code updates [10]. Generic approaches towards packet level data compression are situated in-between these two layers, targeting to save energy by efficiently downsizing packet payloads or headers. A scheme for packet header compression is proposed for 6LoWPAN [11], while S-LZW [12] compacts blocks of logged data before transmission. We are however not aware of any lossless payload compression mechanisms that operate on generic data streams in WSNs.

In this work, we hence present our Squeeze.KOM compression layer, a transparent extension to sensor node platforms that allows to compress data efficiently in a lossless application-agnostic manner while requiring only minor modifications to existing application code. Energy efficiency is ensured by performing compression only if savings can be achieved thereby. Complex and highly demanding compression operations are inherently excluded, as they do not only decrease energy efficiency, but also add latency to the transmission. As Squeeze.KOM encapsulates all functions within a separate layer, it can be combined with application level data encoding, energy-aware MAC protocols, data aggregation mechanisms, or header compression. In the worst case, i.e. in applications with incompressible payload data, a mere one byte increase per packet is required, while significant savings can be realized when sensor data streams with temporal correlations are processed. Especially in delay-sensitive applications, payload compression becomes meaningful as other means of downsizing packets, such as data aggregation, cannot be applied due to the latency constraints.

We analyze application domains of wireless sensor networks and determine characteristics of the employed data packets in Section II. Data compression mechanisms are briefly recapitulated in Section III, with special regard to their applicability in WSNs. Subsequently, we present the Squeeze.KOM compression layer in detail (see Section IV) and evaluate its performance in Section V. Related work on data compression in sensor networks is presented in Section VI, and we draw conclusions in Section VII.

II. Characteristics of Sensor Network Data

Radio traffic in sensor networks exhibits distinct characteristics, primarily influenced by the scenarios in which WSNs are deployed. We analyze applications domains and sensor network deployments in this section, and determine properties of sensor network traffic.

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

A. Sensor Network Application Domains and Deployments

The design space for sensor networks is multi-dimensional [13], and existing applications can be located anywhere within this scope. Estrin et al. have identified application areas for WSNs in [14] and [15], including environmental and physiological monitoring, precision agriculture, smart spaces and inventory tracking. While in monitoring scenarios, sensor readings are commonly transferred to an external sink node, where they are centrally collected and analyzed [16], networks can also operate in ad hoc fashion, exchanging sensor readings and performing data processing on a local scale. In both cases, sensor readings taken from the physical environment surrounding the node are the predominant type of traffic on the radio.

In the Great Duck Island project, data packets were composed of a 25 byte payload containing temperature, humidity, barometric pressure, and light level readings [17]. In GlacsWeb, Martinez et al. deployed probes with temperature, pressure, orientation, external conductivity, and strain gauge sensors into a glacier [18], and transferred their readings in a 16 byte payload. The social behavior of zebras was tracked in ZebraNet [19], where GPS coordinates were transferred to a base station in packets of 64 bytes each. The SATIRE body area network targets to trace human activity patterns and keep track of the current location [20]. To cover rapid movements, it collects sensor readings at a rate of 55 packets per second with a payload size of 68 bytes each.

B. Data Characteristics in Wireless Sensor Networks

The prevalence of sensor readings in the payload of radio packets is a distinct characteristic of WSNs. As opposed to random values, sensor data originating from the physical environment often exhibits correlation in both spatial and temporal dimensions. Depending on the physical phenomenon of interest, readings may change slowly (such as humidity values), or even remain steady for long periods of time (e.g. location information in a network with low node mobility).

Spatial correlation is commonly addressed by means of clustering and data aggregation, effectively reducing the number of packets required to transfer data from a specific region within the sensor network to the destination. In contrast, temporal similarities of successive sensor readings make the resulting stream of data well suited for compression. The applicability of data aggregation is limited in some scenarios, like inventory tracking and user-centric sensing, where different users or objects within close proximity to each other will most likely return different sensor data. However, the temporal resemblance of sensor data is not affected hereby, so stream-oriented compression can still be applied to exploit this property.

Small packet sizes are a second inherent characteristic of sensor networks, and need to be considered specially. Sensor nodes fitted with IEEE 802.15.4 compliant transceivers are limited to a maximum packet length of 127 bytes, as defined by the standard [21]. Furthermore, these packets are typically addressed to a small set of recipients, predominantly nodes on the route to the sink, or local neighbors. A last characteristic aspect of sensor networks is the frequency of packet transmission; while data in environmental monitoring applications is often transmitted periodically or at the request of other nodes, other applications, such as directed diffusion [22], follow a push-based approach where the availability of requested sensor information initiates a transmission.

Fig. 1. Common sequence of compression steps

III. Applicability of Data Compression

Research on reducing the size of executables and data files has been conducted for many years to save valuable space on costly storage devices. After Ziv and Lempel presented their LZ77 algorithm in 1977 [23], many further approaches have been made towards compressing data. For a complete reference, we refer the interested reader to [24], while we only regard algorithms suited for sensor nodes, i.e. devices with tight resource constraints, at this stage.

Two kinds of compression algorithms exist: While *lossy* compression is used to achieve significant savings at the cost of distorted or lost information, *lossless* compression algorithms ensure that compressed data can be entirely restored. We focus on lossless data compression in this paper, as many sensor network applications are susceptible to errors introduced by lossy compression. As lossy compression algorithms additionally require information about both the targeted precision and the structure of the data to be compressed to minimize the loss of accuracy, they can not be applied in the desired application-agnostic manner and are thus not investigated in further detail.

A. Lossless Data Compression

Most lossless data compression algorithms comprise the sequence of actions shown in Fig. 1. In a first *preprocessing* step, data structures are re-arranged to increase their compressibility by reversibly rearranging the bytes. The most common algorithms include Move-to-Front (*MTF*) coding [25], and the Burrows-Wheeler transformation (*BWT*) [26], although a variety of modifications and alternatives are known (cf. [27]).

Subsequently, data *compression* takes place, reducing the size of the preprocessed data by eliminating redundancies. Run-length encoding (*RLE*) is a technique to reduce multiple successive appearances of the same symbol within an input string by replacing its repetitions with an appearance count field. The more complex LZ77 [23] and LZW [28] algorithms compare sequences in the input data to elements contained in a sliding window or a dictionary, respectively, and replace matching elements by according references.

The main compression operation is then followed by an *entropy coding* step that reduces the entropy of the output data. Range Coding (*RC*) reduces the information entropy by iteratively dividing a finite length number range proportionally to the occurrence count of all symbols within the input, and generates a single floating point number representing the entire

TABLE I
MAXIMUM ACHIEVABLE COMPRESSION RATIO FOR PERMASENSE DATA

Compressor	Output Size	Bytes per Packet	Output Ratio
Uncompressed	591900 bytes	30 bytes	1.0
GZIP	263746 bytes	13.37 bytes	0.45
BZIP2	297340 bytes	15.07 bytes	0.50
7-Zip LZMA	149078 bytes	7.55 bytes	0.25
7-Zip PPMD	284232 bytes	14.40 bytes	0.48

input sequence. A Huffman tree contains bit representations of used symbols, with their length inversely proportional to the frequency of their occurrence within the input stream. Entropy decoders either need the used symbol dictionary in advance, or operate on a static dictionary with strongly decreased coding gain. This limitation can be overcome by using adaptive coding approaches, e.g. dynamic Huffman codes [29].

As nodes in wireless sensor networks are generally fitted with low-power microcontrollers to allow for long lifetimes when operated on batteries, they are limited in terms of both CPU speed and available memory. Especially when applications with high memory demands are run on sensor nodes, additional data compression algorithms must expose small code and memory footprints. All algorithms presented in this section have been selected with regard to these constraints.

B. Compressibility of Sensor Data

The limited memory and computing resources on sensor nodes, and the precondition that data compression must be efficient in terms of energy consumption, prove many generic compression algorithms inapplicable. We have applied aforementioned compression algorithms on packets taken from the PermaSense project [30], the data being present in a packet of 30 bytes length with the structure shown in Fig. 2. Concisely, a 2 byte sequence number is followed by ten sensor readings of 2 bytes each, and the packet is terminated by an 8 byte timestamp.

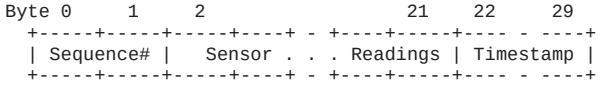


Fig. 2. Sample packet structure from the GSN PermaSense project

We have downloaded the data measured by PermaSense node 2036 from 15 November through 15 December 2008, yielding 19,730 packets, and analyzed the data set regarding its overall compressibility. The results are shown in Table I. It becomes clear that depending on the employed compression type, savings of up to 75% can be achieved when compressing the data in its entirety and making use of adaptive statistical compression techniques. Subsequently, the packets have been compressed one by one by a number of different combinations of the presented algorithms, and average resulting output sizes are shown in Table II. It is obvious from the results that only some sequences employing range coding provide results that are slightly smaller than the corresponding input data. However, in the worst case, the output was twice as large as the input, clearly disqualifying these compression mechanisms on a per-packet basis for sensor network traffic.

TABLE II
OUTPUT SIZE AND COMPRESSION RATIO OF PERMASENSE DATA

Operation Sequence	None	Entropy Coding Huffman	Range Coding
	30 bytes (1.0)	47.4 bytes (1.58)	28.7 bytes (0.96)
MTF	30 bytes (1.0)	51.4 bytes (1.71)	28.9 bytes (0.96)
BWT	34 bytes (1.13)	50.6 bytes (1.69)	31.9 bytes (1.06)
RLE	31 bytes (1.03)	47.9 bytes (1.6)	29.5 bytes (0.98)
LZ77	49.4 bytes (1.65)	60.6 bytes (2.02)	42.2 bytes (1.4)
LZW	32.6 bytes (1.09)	31.8 bytes (1.06)	31.8 bytes (1.06)
MTF RLE	30 bytes (1.0)	52.2 bytes (1.74)	29 bytes (0.97)
BWT RLE	37.5 bytes (1.25)	52.1 bytes (1.74)	34.5 bytes (1.15)
MTF LZ77	49.4 bytes (1.65)	60.6 bytes (2.02)	42.2 bytes (1.4)
BWT LZ77	49.4 bytes (1.65)	60.6 bytes (2.02)	42.2 bytes (1.4)
MTF LZW	32.6 bytes (1.09)	58.6 bytes (1.95)	31.8 bytes (1.06)
BWT LZW	32.6 bytes (1.09)	58.6 bytes (1.95)	31.8 bytes (1.06)

IV. THE SQUEEZE.KOM COMPRESSION LAYER

Packet compression can assist in maximizing the lifetime of WSNs by reducing the energy consumed by radio transmissions. The analysis of sensor data compressibility has however made clear that compression algorithms operating on an individual packet cannot achieve high compression gains due to the limited correlation between its payload contents.

We address the problem of limited compressibility by a different approach, exploiting the inherent characteristics of surveilled physical environments. As many environmental parameters exhibit high temporal correlation with slow changes over time, two packets sent successively can be expected to bear strong resemblance to one another. Based on transferring these differences between packets in a compressed way, the Squeeze.KOM sensor network compression layer can achieve compression gains superior to plain packet compression while consuming a modest amount of sensor node resources only. This is different from compression layers present in other radio stacks, such as the KSN RadioStack [31], which uses the DEFLATE [32] algorithm with a default window size of 32 kilobytes on all outgoing data.

A. System Overview

Squeeze.KOM is a transparent compression layer that can be seamlessly integrated with existing node platforms and applications, as it replicates the interfaces provided by the network layer. This necessitates only small modifications to the application code, allowing to adapt existing applications to the new layer easily. Operating on unidirectional streams of data additionally allows to exploit computational heterogeneity of platforms by selecting compression parameters with regard to a node's capabilities. A detailed overview of the elements in the compression layer is depicted in Fig. 3.

Packets originating from the application are separated in header and payload fields and forwarded to the core component of the layer, the compression framework. A comparison of the payload to a set of previously sent packets stored in a local transmission history allows the framework to determine whether transmitting a differential packet is feasible. If so, the payload is replaced by a reference to the most similar history

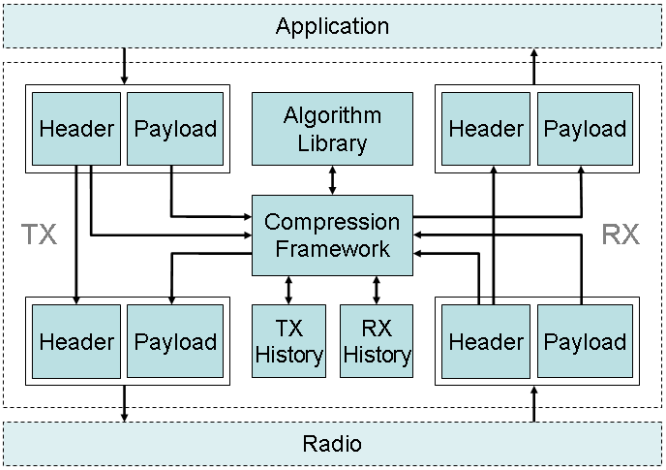


Fig. 3. Internal structure of the compression layer

element, and the difference between both sequences. Subsequently, the payload is analyzed regarding its compressibility and compressed if savings can be achieved thereby.

The receiver operates in reverse to the data encoding at the sender. Received packets are forwarded to the local compression framework, where compressed contents are first decompressed. When packet differences have been transferred only, the original sequence is restored by combining the received difference data with the local history, while full packets are directly forwarded to the application after inserting their payload into the reception history.

In case energy-efficient multi-hop data transfer to a sink node is required, i.e. no in-network processing needs to take place, data can be directly forwarded without the need for local decompression at intermediate nodes. The source address must however be retained within the packets to indicate to the receiver which index packet set to use. Extending the compression layer by routing functionalities is possible, though beyond the scope of this paper. Hence, currently all incoming packets are decompressed before taking any further actions.

B. Compression by Differential Transmission

Based on the determined characteristics that many subsequently sent packets in sensor networks bear high resemblance to each other, we propose an encoding scheme based on encoding payload differences to the last fully sent packet. To differentiate packet types in this paper, we term packets that were fully sent as *index* packets, denoting their payload as I_n, where n is the number of the entry within a locally stored array of index packets. A separate transmission history of index packets is maintained for each receiver node. It stores the index n, the corresponding data I_n and a hash value $Hash(I_n)$ of the data which is inserted in differential packets to ensure sender and receiver are in sync. Opposed to index packets which contain a full payload, we assign differential packets the term Δ_n, where n is the index of the referred index packet.

We have implemented three methods to generate the differential of I_n and the payload P to be sent:

1) By means of bytewise arithmetic subtraction ($P - I_n$), payloads with high similarity result in a sequence of values close to zero.
2) Performing an XOR operation on the payloads of similar packets ($P \oplus I_n$) results in an output stream with a sparse number of set bits.
3) Preceding the XOR operation in the previous method by a conversion of the payloads to Gray Code [33] results in an output stream ($GC(P) \oplus GC(I_n)$) with an even higher number of '0' bits, however at the cost of computationally expensive decoding.

To support compression of the bitstream generated by the XOR operation, we have implemented a distance coding scheme operating on bit level. As the maximum number of bits in an IEEE 802.15.4 packet is limited by the standard, we have used a variation of Golomb-Rice coding [34] that encodes the distance between '1' bits, i.e. the length of runs of '0' bits. This distance coding step simultaneously reduces the entropy in the output. The resulting average number of '0' bit runs in the output are compared in Table III, where differences of the current payload P_i to both the first packet P_0 and the previous P_{i-1} are compared. The threshold of RLE was set to only compress sequences of two or more repetitions of the same symbol, and the Golomb-Rice scheme was set to operate on a basis of 8. The observed small benefit of applying the Gray Code (*GC*) to the packets is however faced by an increased decoding effort and thus neglected in the further analysis.

The structure of our binary distance code is shown in Table IV. When no '0' bit is encountered between '1' bits in the input, the resulting code output is only one bit long, thus not increasing the size of the output. Runs of '0' bits with lengths of $1 \ldots 8$ are encoded in a 5 bit symbol, while run lengths of $9 \ldots 70$ are represented in an 8 bit symbol. Two symbols are reserved; an output with all bits set to '1' indicates the end of the input data (which is necessary as byte-alignment is not ensured), while the same code with the LSB unset signals that the number of '0' bits exceeds 71 and is therefore calculated as the sum of 71 and the value of the succeeding code.

TABLE IV
REALIZATION OF THE BINARY DISTANCE CODE

Bit Representation								Interpretation
0								No zero bit
1	0	n_2	n_1	n_0				n (1-8) zeroes
1	1	n_5	n_4	n_3	n_2	n_1	n_0	n (9-70) zeroes
1	1	1	1	1	1	1	0	71+n(next code) zeroes
1	1	1	1	1	1	1	1	End of data

If the application layer invokes a packet send call, the corresponding payload P is checked for similarities with the elements in the history table. If no entries with sufficiently high similarity are present, a new entry is created in the history table, and assigned the next available index. In case all indices are taken already, the existing index element which has not been used for the longest time is replaced with the new

TABLE III
PERFORMANCE STATISTICS OF THE PRESENTED DIFFERENTIATION ALGORITHMS, APPLIED TO PERMASENSE DATA

Input	Mean length of '0' runs	Mean length of '1' runs	Longest '0' run	Longest '1' run	Output Size after Compression/Coding RLE	Golomb-Rice	Binary Distance
P_i	2.15 bits	2.16 bits	35 bits	20 bits	31.05 bytes	75.72 bytes	44.98 bytes
$P_i - P_0$	2.35 bits	2.09 bits	95 bits	26 bits	29.49 bytes	72.89 bytes	43.05 bytes
$P_i \oplus P_0$	2.53 bits	1.89 bits	95 bits	15 bits	29.45 bytes	64.66 bytes	42.18 bytes
GC(P_i) $\oplus$ GC(P_0)	2.58 bits	1.75 bits	98 bits	7 bits	29.82 bytes	45.71 bytes	42.10 bytes
$P_i - P_{i-1}$	16.45 bits	3.43 bits	237 bits	25 bits	21.40 bytes	34.40 bytes	15.17 bytes
$P_i \oplus P_{i-1}$	18.40 bits	1.99 bits	237 bits	17 bits	21.39 bytes	20.71 bytes	13.39 bytes
GC(P_i) $\oplus$ GC(P_{i-1})	18.23 bits	1.45 bits	237 bits	7 bits	21.38 bytes	16.97 bytes	12.98 bytes

index packet and the corresponding hash value is calculated. On the other hand, if the similarity requirement is fulfilled, the differential between I_n and P is calculated. The resulting Δ_n is then transferred in conjunction with $Hash(I_n)$ to make sure an identical I_n is referred to at both sender and receiver. If both packet payloads are identical, an empty Δ_n is transmitted. Squeeze.KOM supports compression of both index and differential packets, denoted as $I_{n,enc}$ and $\Delta_{n,enc}$, respectively. However, due to the small compression gain for index packets observed in Table II, we focus on compression of delta packets in this paper. In case compression does not reduce the packet size or is not viable by means of energy efficiency, the data is sent uncompressed.

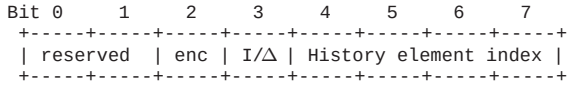


Fig. 4. Example bit structure of the status field

A status field is added to the data to transfer required information to the receiver, such as the index n of the element in the history table, flags whether the payload is compressed and if it represents an index or a differential packet. Currently reserved bits could be employed to provide QoS parameters to the receiver in future implementations. A sample implementation of the status field, as used in our current implementation, is shown in Fig. 4. Limited to a single byte, the overhead is reduced in cases of incompressible payload data, while allowing for a history size of 16 elements.

The memory consumption of the compression layer is dominated by the number of index packets stored for outgoing and incoming connections, and can thus be reduced both by limiting the number of entries for outgoing data, as well as by notifying neighbor nodes to reduce the number of entries allocated for their transmissions. Each sender node can decide whether its outgoing links are fitted with unique index lists, or if a common list is employed, however necessitating mechanisms to ensure all neighbors are in sync with the latest set of index packets.

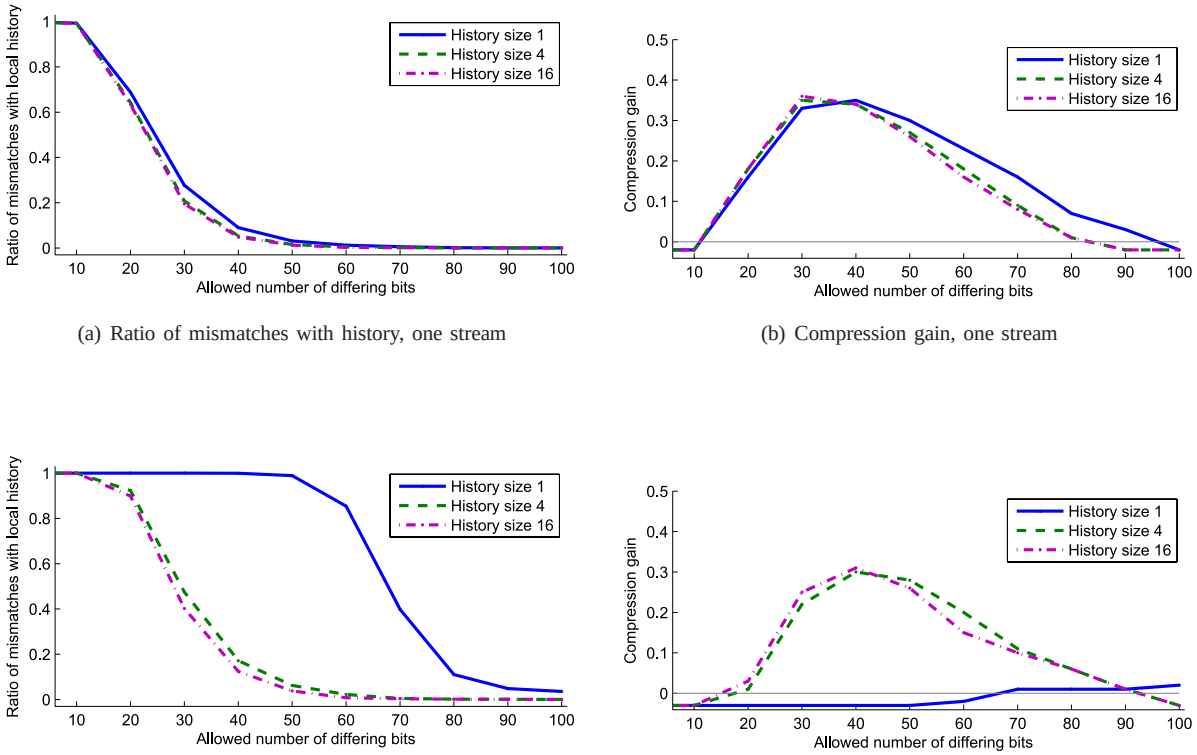
C. Decoding the Data

To restore the packet payload contents from $\Delta_{n,enc}$ (or Δ_n, respectively), a local copy of the referred index packet I_n is required. Receivers thus have to keep track of incoming index data, maintaining a history list with information about the set of formerly received indices n and the corresponding index packets I_n, identical to the table present at the sender side. The hash value does not need to be transferred with the index packet; instead it is calculated locally on reception of an index packet. Incoming packets $\Delta_{n,enc}$ contain a copy of the hash value of I_n and can thus determine whether the identical history element is referred to. In our current implementation, the length of the hash value has been defined as 8 bits. If a mismatch is detected, the packet can either be discarded, or the sender be prompted to retransmit the packet as an index packet, depending on the importance of the data. This check is essential to encounter lost index packets and ensure that nodes rejoining the network do not operate on outdated index data. By default, packets are transferred in a best-effort manner, where the compression layer is not required to provide any latency or delivery guarantees. If reliable transport is required, additional QoS mechanisms can be integrated by using the reserved fields, allowing to specify latency or reliability requirements.

V. PERFORMANCE EVALUATION

The presented compression layer has been implemented and analyzed in detail in a simulation environment. The identified tunable parameters and their impact on the overall performance of Squeeze.KOM are presented and discussed in the following subsections. Finally, the application has been ported and evaluated on SunSPOT nodes.

For the following experiments, we have set up a simple network topology with one sender and a single receiver node. Unless specified otherwise, we have assumed an ideal communication channel (no packet loss). For the evaluation, we have again used the PermaSense data set with the structure given in Table 2. The entire 19,730 packets were used for all conducted experiments, and the XOR operation on P and I_n, followed by binary distance coding were used to compress delta packet payloads, while index packets have been transferred uncompressed. To maintain the clarity of our analysis, we have empirically determined and retained some parameters throughout the experiments. This provides a representative assessment of the performance of Squeeze.KOM when applied on real sensor data for the given scenario. A generalization of the results can however not be performed directly, as optimal results depend on both the application and environmental parameters of the deployed WSN, which might differ significantly.



(a) Ratio of mismatches with history, one stream

(b) Compression gain, one stream

(c) Ratio of mismatches with history, four streams

(d) Compression gain, four streams

Fig. 5. Impact of the similarity threshold on history mismatches and compression gain for one and four streams

A. Impact of the Similarity Requirement Threshold

To generate a compressible differential packet, the payload to be compressed must exhibit sufficient similarity to an element stored in the local history. As the XOR operation is part of the differentiation step and needs to be performed anyway, Squeeze.KOM uses the Hamming distance as its metric for similarity to the history elements, i.e. it counts the number of '1' bits in the result of the XOR operation. It is necessary to define a maximum allowed number of differing bits as the threshold for similarity. Small threshold values hereby lead to highly compressible differential packets, as the resulting data features long runs of '0' bits and is thus well suited for the binary distance code. However, choosing too small threshold values also results in a greater number of mismatches with the local history, and thus to an increased number of index packet transmissions. In contrast, high threshold values result in less compressible delta packets. The impact of the similarity threshold has been analyzed for the single stream contained in the data set, and the results are shown in Fig. 5(a).

Clearly, a high value for the similarity threshold results in the smallest number of mismatches, and hence the smallest number of required (uncompressed) index packet transmissions. The compression gain, being an indicator for the achievable size reductions by applying Squeeze.KOM, is analyzed in Fig. 5(b) and confirms that high threshold values lead to degraded compression ratios.

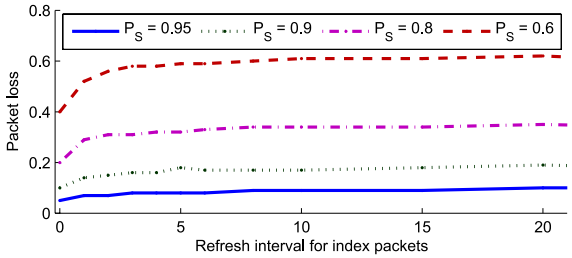
It is hence mandatory to find a compromise between threshold and compression gain. In the following simulations, we therefore assume a maximum allowed deviation of 40 bits (equalling the sixth part of the overall payload length of 30 bytes) for the given data set, where an average number of history mismatches of 9% has been determined.

B. Impact of the History Size

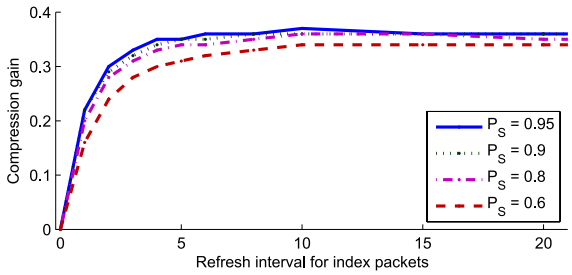
When a single stream of data with small changes over time is transferred with the determined settings, the presented algorithm leads to an average compression gain of around 35%. However, when multiple streams of data need to be transferred between two nodes, a correlation between these streams is not necessarily given.

The compression layer has hence been tested with four different PermaSense streams being transferred in parallel. The ratio of history mismatches is shown in Fig. 5(c), clearly confirming that a history size smaller than the number of distinct streams is insufficient to achieve gains by compression. Instead, the compression layer will exclusively transfer index frames in this case, thus increasing the packet size by one byte due to the status field. The compression gain is depicted in Fig. 5(d), proving that gains similar to the single stream case are possible when the history is dimensioned well.

It can also be noticed that, in both cases, larger history sizes lead to slightly increased maximum compression gains, but as well shift the corresponding threshold value to the left. This originates from the fact that less index replacements take place when large history sizes are used, and thus old index entries remain longer in the history list.



(a) Impact of the refresh interval on packet loss



(b) Impact of the refresh interval on the compression gain

Fig. 6. Impact of the refresh interval on packet loss and compression gain

C. Impact of Packet Loss

The presented single-hop scenario was modified to use a lossy channel, with packet success probabilities of $P_S = \{0.95, 0.9, 0.8, 0.6\}$. The application was configured to use a single PermaSense stream, a history size of one element and a similarity threshold of 40 bits. To recover from the loss of index packets, the sender node was extended by an index refresh interval. When the number of sent delta packets referring to the same index packet exceeds the refresh interval, the data to be sent is transmitted as an index packet. This may be the case even in cases when the similarity requirement is fulfilled, but is important to keep the index packet list at the receiver up to date.

It is clearly visible in Fig. 6 that in cases where index packets are sent more frequently, a higher success probability is given, however at the cost of not applying differential compression on these index packets. The compression gain is thus reduced in favor of an increased probability of receiving correct and complete data. For a large refresh interval, the overall packet loss rate roughly increases by 50% compared to uncompressed transmission. However, the refresh interval could easily adapt to the channel conditions and dynamically adjust itself during runtime.

D. Summary of the Simulation Results

The compression gain analysis of Squeeze.KOM has shown that under both idealized and realistic channel conditions, a data set taken from a real sensor network deployment can be reduced to less than two thirds of its initial size by using our compression layer. The tunable parameters have been presented, and the corresponding optimum settings for the given scenario been analyzed. A history size equal to the number of streams transported at a time has shown to already result in good compression gains. A similarity threshold below 20% of the payload bit length exposes highest compressibility values for the resulting differential packets due to the selected binary distance code. Although missing index packets add to the impact of packet loss when using the compression layer, good compression ratios can already be achieved with refresh intervals of only five packets, effectively alleviating the problem of lost index packets.

E. Real-World Evaluation

The compression layer has been ported to the SunSPOT platform[1], and data from a SunSPOT's integrated sensors was transmitted to a set of four sink nodes, following the payload structure indicated in Fig. 7. Identical to PermaSense packets, the payload is composed of a 2 byte sequence number followed by ten sensor readings (acceleration and tilt in three dimensions, temperature, light intensity, the state of the integrated momentary switches, and the current supply voltage level) of 2 bytes each, and an 8 byte timestamp. Although using the same structure as the PermaSense data, the chosen set of sensors differs, and the accelerometers introduce even faster changes of sensor readings, decreasing the compressibility of the data.

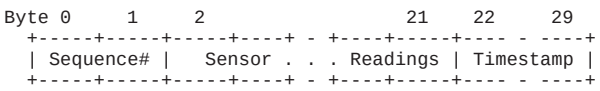


Fig. 7. Packet structure used for the real-world validation

The sink nodes were located in an office environment and distributed around the sender, at a distance of 5 meters. The sender was worn on the wrist of a typing person and configured to transmit 10,000 successive packets with sensor readings, sending each of them twice; once without compression to get an estimate on the real channel loss rate, and once using the Squeeze.KOM compression layer. The compression layer was again configured to use a history size of one element, a threshold value of 40 bits and a refresh interval of 5 packets, as determined in the preceding sections for the PermaSense data. Although this parameter set does not result from a previous analysis of the data characteristics, the results for packet loss and the savings achieved by packet compression compared in Table V confirm that the Squeeze.KOM compression layer is operating as intended on real hardware and exposes behavior similar to the simulation results. Besides the expected slight increase in unrecoverable packets, resulting from outdated or missing index elements, the use of the compression layer leads to compression gains of around 12% even in our scenario where highly dynamic accelerometer data comprised a major part of the payload.

[1]SunSPOTs have been selected for the sake of eased implementation. Squeeze.KOM is however not limited to this mote platform.

TABLE V
REAL-WORLD EVALUATION OF THE COMPRESSION LAYER

	Node 1	Node 2	Node 3	Node 4
Loss (uncompressed)	2.3%	6.4%	3.0%	4.3%
Loss (compressed)	2.7%	8.1%	4.7%	6.1%
Compression gain	12.1%	11.1%	12.0%	12.0%

VI. RELATED WORK

The area of data compression in wireless sensor networks has been addressed in different ways in recent research. Barr and Asanović have determined in [35] that the energy consumed by the radio device to transmit one byte of data on the radio can also be used to perform up to a thousand CPU operations. This allowed them to conclude that data compression was feasible by means of energy consumption if less energy was consumed by the compression operation than required to send the uncompressed packets. Their findings were however based on Compaq Personal Server handheld devices, offering many times the resources available on common WSN nodes. Sadler and Martonosi built on these results and analyzed energy consumptions of common sensor network platforms in [12], determining that between four thousand and two million instructions (depending on the employed radio transceiver device) can be executed by a MSP430 microcontroller at the same energy required to transfer one byte on the radio, confirming the general idea that data compression is viable on sensor nodes. They developed the S-LZW algorithm, optimized to run on sensor nodes and compress blocks of data stored on the node's external memory.

In [36], Kimura and Latifi also noticed that most existing compression applications designed for computers with megabytes of RAM and CPU speeds of hundreds of MHz, are not suited to be run on resource-constrained sensor node platforms. Instead, they summarize four approaches to compress different types of data in WSNs, applicable for resource-constrained sensor nodes. However, the compared approaches either specifically relate to spatially correlated data, or describe highly application-specific means of data compression. Pattem et al. also describe a scheme for routing with compression of spatially correlated data in [37].

Our approach bears resemblance to the RObust Header Compression framework (ROHC) [38], which recognizes similarities in header data of packets and transmits the differences only when it is sure that the receiver can decode them accordingly. However, ROHC requires steady network topologies, where sender and receiver can assume a long-term stability of their link, which is commonly not given in sensor networks. When packet structures are statically defined prior to node deployment, and some fields are known to remain constant or only change incrementally, the EasiPC packet compression scheme by Ju and Cui can be used to transmit changed fields only [39]. This method however requires developers to specify the type of each payload field in advance and fails when contents deviate from the definition. Another solution to transmit compressed differential data is the VCDIFF protocol [40]. It uses the original data to create a dictionary for compression of further packets and only transmits these encoded differential data to the receiver. As with ROHC, it must be ensured that the original data is present at the receiver as well, so both parties operate on the same dictionary.

Tsiftes et al. have focussed on compressing firmware updates that are transferred over the radio, applying the SBZIP algorithm [10], a derivative of the BZIP2 mechanism, modified for operation in sensor networks. However, it does not target to compress any application-generated data, but instead performs stateless compression of application code, thus being a good supplement to application-level encoding mechanisms. Finally, a couple of lossy compression algorithms exist, such as the transmission of packet predictions, as presented by Blass et al. in [41]. The approach of using Kalman filters to predict readings is however computationally expensive as well as lossy, and thus not directly related to our approach.

VII. CONCLUSIONS

We have presented Squeeze.KOM, a transparent compression layer that seamlessly integrates with any sensor node platform. The layer operates transparently, rendering major modifications to application software unnecessary. We have presented a stream-oriented compression scheme that deliberately exploits the temporal correlation of sensor data. Its performance on sensor data taken from real nodes has indicated that significant savings can be achieved, even when applying only basic differentiation and compression mechanisms.

Being entirely independent of application level compression and fully interoperable with energy-aware MAC layers, our solution can be integrated with any application, and on any platform with resources to spare. As the packet headers are left untouched, our approach can even be seamlessly combined with header compression to further compact the packet. Its lossless character allows to transfer both sensor readings as well as network management and topology control messages. Decreased packet transmission durations, resulting from packet size reductions, lead to energy savings and thus extended node lifetimes in WSNs.

REFERENCES

[1] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, pp. 102–114, 2002.

[3] Texas Instruments Inc., "CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver (Rev. B)," 2007. [Online]. Available: http://www.ti.com/lit/gpn/cc2420

[4] B. Krishnamachari, D. Estrin, and S. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," in *Proceedings of the International Workshop on Distributed Event-Based Systems (DEBS)*, 2002.

[5] T. Arici, B. Gedik, Y. Altunbasak, and L. Liu, "PINCO: A Pipelined In-Network COmpression Scheme for Data Collection in Wireless Sensor Networks," in *Proceedings of the 12th International Conference on Computer Communications and Networks (ICCCN)*, 2003.

[6] D. Petrović, R. C. Shah, K. Ramchandran, and J. Rabaey, "Data Funneling: Routing with Aggregation and Compression for Wireless Sensor Networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[7] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC protocol for Wireless Sensor Networks," Information Science Institute (ISI) / University of Southern California (USC), Tech. Rep. ISI-TR-543, 2001.

[8] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[9] E. Magli, M. Mancin, and L. Merello, "Low-Complexity Video Compression for Wireless Sensor Networks," in *Proceedings of the International Conference on Multimedia and Expo (ICME)*, 2003.

[10] N. Tsiftes, A. Dunkels, and T. Voigt, "Efficient Sensor Network Reprogramming through Compression of Executable Modules," in *Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2008.

[11] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4944.txt

[12] C. M. Sadler and M. Martonosi, "Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[13] K. Römer and F. Mattern, "The Design Space of Wireless Sensor Networks," *IEEE Wireless Communications*, vol. 11, no. 6, 2004.

[14] D. Estrin, R. Govindan, and J. Heidemann, "Embedding the Internet," *Communications of the ACM*, vol. 43, p. 5, 2000.

[15] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the World with Wireless Sensor Networks," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2001.

[16] J. K. Hart and K. Martinez, "Environmental Sensor Networks: A revolution in the earth system science?" *Earth-Science Reviews*, vol. 78, 2006.

[17] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[18] K. Martinez, R. Ong, and J. Hart, "Glacsweb: A Sensor Network for Hostile Environments," in *Proceedings of the First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2004.

[19] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with Zebranet," in *Proceedings of the 10th Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2002.

[20] R. K. Ganti, P. Jayachandran, T. Abdelzaher, and J. A. Stankovic, "SATIRE: A Software Architecture for Smart AtTIRE," in *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2006.

[21] IEEE Std, "802.15.4 Part 15.4: Wireless medium access control (MAC) and Physical layer (PHY) specifications for low-rate wireless personal are networks (LR-WPANs)," 2006.

[22] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[23] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, 1977.

[24] D. Salomon, *Data Compression: The Complete Reference*. Springer-Verlag, 2007.

[25] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei, "A Locally Adaptive Data Compression Scheme," *Communications of the ACM*, vol. 29, no. 4, 1986.

[26] M. Burrows and D. J. Wheeler, "A Block-sorting Lossless Data Compression Algorithm," digital Systems Research Center, Palo Alto, California, SRC Research Report 124, 1994.

[27] S. Deorowicz, "Second Step Algorithms in the Burrows-Wheeler Compression Algorithm," *Software Practice and Experience*, vol. 32, no. 2, 2002.

[28] T. A. Welch, "A Technique for High-Performance Data Compression," *Computer*, vol. 17, no. 6, 1984.

[29] J. S. Vitter, "Design and Analysis of Dynamic Huffman Codes," *Journal of the Association for Computing Machinery*, vol. 34, no. 4, 1987.

[30] 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)*, 2007.

[31] Institute of Program Structures and Data Organization, University of Karlsruhe, "KSN RadioStack Project," Online: http://www.ipd.uni-karlsruhe.de/~bestehor/KSN/RadioStack, 2008.

[32] P. Deutsch, "DEFLATE Compressed Data Format Specification version 1.3," RFC 1951 (Informational), 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1951.txt

[33] F. Gray, "Pulse Code Communication," *U.S. Patent 2,632,058*, filed 13 November 1947, issued 17 March 1953.

[34] R. F. Rice and J. R. Plaunt, "Adaptive Variable-Length Coding for Efficient Compression of Spacecraft Television Data," *IEEE Transactions on Communication Technology*, vol. COM-19, no. 6, 1971.

[35] K. Barr and K. Asanović, "Energy Aware Lossless Data Compression," in *Proceedings of the First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2003.

[36] N. Kimura and S. Latifi, "A Survey on Data Compression in Wireless Sensor Networks," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC)*, 2005.

[37] S. Pattem, B. Krishnamachari, and R. Govindan, "The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks," in *Proceedings of the Third International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004.

[38] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed," RFC 3095 (Proposed Standard), 2001, updated by RFCs 3759, 4815. [Online]. Available: http://www.ietf.org/rfc/rfc3095.txt

[39] H. Ju and L. Cui, "EasiPC: A Packet Compression Mechanism for Embedded WSN," in *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2005.

[40] D. Korn, J. MacDonald, J. Mogul, and K. Vo, "The VCDIFF Generic Differencing and Compression Data Format," RFC 3284 (Proposed Standard), 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3284.txt

[41] E.-O. Blass, L. Tiede, and M. Zitterbart, "An Energy-Efficient and Reliable Mechanism for Data Transport in Wireless Sensor Networks," in *Proceedings of the 3rd International Conference on Networked Sensing Systems (INSS)*, 2006.