SFHC.KOM: Stateful Header Compression for Wireless Sensor Networks

Andreas Reinhardt, Parag S. Mogre, Tobias Koenig, Ralf Steinmetz
Multimedia Communications Lab, Technische Universität Darmstadt, Rundeturmstr. 10, 64283 Darmstadt, Germany
Email: {andreas.reinhardt, parag.mogre, tobias.koenig, ralf.steinmetz}@kom.tu-darmstadt.de

***Abstract*—Nodes in wireless sensor networks are generally confined in the energy budget available for their operation, hence energy-aware application design is mandatory to achieve long node lifetimes. Radio transmissions represent an inherent, but energetically costly characteristic of sensor networking. Significant reductions in the overall energy consumption can thus be achieved by reducing both the number of packet transmissions as well as the corresponding packet lengths. Data compression is a viable approach to conserve energy by increasing the information density within packets and thus transmitting shorter packets on the radio. We investigate the compression of packet headers in wireless sensor networks in this paper. Inspired by technologies used in the Internet and characteristics stemming from existing sensor network deployments, we propose a novel scheme for stateful header compression. Our scheme is specifically designed to consider both static and mobile leaf nodes, and can thus be applied in a majority of sensor network deployments. We analyze our scheme in different settings and show that its application leads to reductions of the required transmission energy and thus extended node lifetimes.**

I. Introduction

The tight energy budgets available to the nodes in Wireless Sensor Networks (*WSN*s) present a major challenge for application designers to utilize the available battery charge optimally. While microcontrollers (*MCU*s) exhibit current consumptions of less than one milliampere under full load (e.g. the MSP430), radio transceiver technology has not kept up with this pace of optimization, and current devices (e.g. the CC2420 [1]) still expose an energy consumption in the order of tens of milliamperes. To optimize the energy consumption of radio communications, the use of energy-aware MAC protocols and means to increase the information density within packets have been developed. We have previously analyzed approaches to reduce the radio energy consumption by increasing the information density within packet payloads through data compression [2], and observed measurable energy savings. To supplement the benefits of payload compression, we address header compression (*HC*) mechanism in this paper.

HC aims to eliminate redundancies in header fields of subsequent packets by omitting fields with static or deterministic values from transmission. It has its origins in IP-based networks [3], where often highly similar headers are sent throughout a communication session. When analyzing traffic in common WSN deployments, similar traffic characteristics can be observed; when e.g. tree-based routing [4] is used, packets carrying sensed data are generally forwarded to the next hop in the direction of the sink and hence have – amongst other identical fields – identical destination addresses. Even when mobility is present in the network, or routing schemes introducing diversity, such as PiRAT [5], are employed, many fields of the packet header remain unchanged.

Header compression algorithms can operate in either stateful or stateless manner. While *stateless* algorithms can only compress packet headers to a limited extent as the nodes do not keep any connection state, they do not require any synchronization between sender and receiver. In contrast, *stateful* approaches pose additional overhead to establish a common connection context, containing information that is omitted from subsequent headers. Greater size reductions can be achieved when applying stateful algorithms, at the cost of increased resource utilization. We address this drawback by presenting a lightweight stateful HC mechanism in this paper, specifically tailored for application in WSNs with support for any routing protocol and the mobility of leaf nodes.

The contributions of this paper are as follows: We compare existing approaches for header compression in both WSNs and the Internet in Sec. II. We analyze the applicability of existing concepts to the characteristics of WSNs in Sec. III, and present the design decisions taken. Resulting from our analysis, we present SFHC.KOM, a stateful header compression algorithm, catering to the demands of WSNs with both mobile and fixed nodes, in Sec. IV. In particular, our solution addresses ambient assisted living scenarios, where a fixed infrastructure is combined with mobile sensing nodes. Many HC approaches are inapplicable in such a setting, as they do not consider the specific requirements and limitations of WSNs. We prove the applicability and benefits of our approach and analyze the packet overhead and energy consumption of our header compression algorithm in different representative settings in Sec. V. We summarize the key insights obtained from our work in Sec. VI, and provide directions for future research.

II. Related Work

The compression of packet headers in IP-based networks has been initially presented in 1984 by Farber et al. in [3]. Since then, many algorithms have been published to reduce the header size of IP-based traffic. In general, header compression schemes can be classified into *hop-by-hop* and *end-to-end* mechanisms. While hop-by-hop header compression mechanisms often operate on the lower layers of the network stack, mechanisms applying the end-to-end concept generally try to exploit similarities in the packet headers of upper layers.

A stateful hop-by-hop compression scheme, reducing the 40 byte TCP/IP header to sizes as small as 3 bytes, is presented in RFC 1144 [6]. Operating in hop-by-hop manner, the mechanism makes several assumptions about the characteristics of TCP/IP headers, such as the presence of monotonically increasing sequence numbers. A noteworthy contribution is its concept of assigning connection numbers to individual streams, allowing to maintain the compression mode even when multiple connections are open. Degermark et al. present an IP header compression scheme for IPv4 and IPv6, compatible for the use with TCP and UDP, in RFC 2507 [7]. Its application can reduce an entire packet header to 4 to 7 bytes, including the 2 byte checksum used in TCP and UDP. In practice, the scheme establishes an individual context for each TCP stream by sending an uncompressed header periodically. All intermediate packets refer to the last full header and thus only transmit differential data. Robust Header Compression (ROHC) [8] uses multiple compression states and always operates in the best compression state it can confidently use for a connection. The state selection is based on various criteria relating to successful compression and decompression (e.g., ACKs and NACKs). ROHC is especially suited to eliminate both error and loss propagation as it uses additional cyclic redundancy checks. Additions to ROHC include the use of forward error correction [9] and the application of Lempel-Ziv-Welch coding on compressed headers to achieve even greater size reductions [10].

In contrast to the presented mechanisms that establish a context for all further transmissions, Stateless Header Compression [11] builds on the observation that the required stream management generates overhead in terms of both computation and memory. Instead, IP headers are compressed by creating shorthand aliases for addresses in the same subnetwork and replacing all address fields in the headers. By assigning different levels of aggregation capabilities, further header fields are also transferred in a compressed, but stateless manner.

Specifically crafted for WSNs, the 6LoWPAN standard [12] targets to make IPv6 applicable in WSNs. Inherently considering the payload size restriction present in IEEE 802.15.4 [13], it also defined a header compression scheme that only makes use of stateless header compression to keep the memory consumption low. As devices in the same WSN are assumed to already share a common state, some fields (such as the used IP version or the common subnetwork prefix) present in common IPv6 headers are consistent throughout a 6LoWPAN sensor network and are thus omitted. The use of Dynamic Link Labels [14] addresses the length of MAC addresses on the data link level of WSNs. As often only a limited number of communication partners are present within a node's radio range, the approach reduces the header length by assigning each node a short unique identifier.

When using the presented hop-by-hop mechanisms, long routes generate additional overhead for the necessary recompression operations at all intermediate nodes. The reduction of this overhead is addressed in end-to-end approaches, such as SEEHOC [15]. Each connection is assigned a unique connection identifier, which is composed of the MAC addresses of both sender and receiver. Management of the soft states along the path is maintained by periodic uncompressed update messages. Finally, hybrid approaches exist, combining end-to-end compression with hop-by-hop means. Both Westphal [16] and Arango et al. [17] hereby propose to compress all headers including network layer headers on a hop-by-hop basis, while all transport layer and above streams are additionally compressed in an end-to-end fashion.

III. Design Considerations

When taking a closer look at stateless header compression, it becomes clear that the extent of header size reductions is limited, as only a pre-defined set of header field values can be transferred in a compressed manner. In practice, the header compression mechanism defined in RFC4944 [12] is optimized for link-local IPv6 communication, with specific efficient encodings for UDP, TCP, and ICMPv6 packets. However, transport protocols specifically tailored to WSNs (such as ESRT [18] or ERTP [19]) cannot be encoded using 6LoWPAN HC, clearly limiting its generic applicability in WSNs.

In contrast, the application of stateful header compression necessitates overhead to mutually establish a common shared knowledge about the connection (the *compression context*). Although establishing a context naturally poses additional overhead on a connection, the stateful operation allows to replace a greater fraction of information through referring to the previously defined context. Besides, by maintaining shorthand terms for the header fields in the context, a limitation to previously defined header fields is not inherently given. Stateful header compression allows for even smaller packet header sizes than when applying stateless mechanisms, and can thus directly reduce the required transmission energy demand.

A. Design Goals

While most existing research on header compression originates from Internet, where TCP/IP traffic is prevalent, dedicated transport protocols are rarely employed in sensor networks, and if so, a wide range of highly WSN-specific protocols is known. Although means to make TCP viable for the use in WSNs exist [20], none of the protocols has yet advanced to the de-facto standard in sensor networks. In contrast to the Internet, means towards compressing packet headers in WSNs should thus not make any assumptions on the employed transport protocol, and need to operate independently.

The second major constraint for embedded sensing systems is their limitation in terms of computational capabilities and memory [21]. The header compression algorithm should thus expose a small code footprint as well as low resource demand to store required information. Its implementation needs to be sufficiently lightweight, as savings can only be achieved when the compression and decompression of headers consumes less energy than the transmission of uncompressed headers over the radio. The tradeoff therefore is to find a header compression algorithm which reduces the sizes of compressed headers to a small fraction of their initial size to allow for savings through shorter radio transmission times, while adhering to the resource limitations of the given node platform.

To adapt to changes in the network quickly, the algorithm should expose a small overhead for connection establishment and low management traffic overhead throughout a communication session. Both aspects are primarily motivated by the possible presence of mobile nodes, which the HC algorithm needs to be capable of integrating seamlessly. Finally, the corresponding changes to the topology as well as the unreliable nature of radio links in WSNs additionally necessitate means to recover from packet losses.

B. Design Decisions

Taking the aforementioned requirements for a header compression algorithm into account, we have taken a set of decisions during the design phase of our stateful header compression scheme, termed SFHC.KOM. To allow for maximum size reductions whilst keeping the resource demand low, we have chosen to implement stateful hop-by-hop compression of packet headers. Independent of the used transport protocol, many header fields (as discussed in detail in Sec. IV-C) only change in a deterministic manner, such as monotonically increasing sequence numbers. Some header fields even remain constant throughout an entire communication session. We have thus chosen to apply a stateful HC approach, where such values are maintained in a context state instead of being present in each outgoing packet. In further communications, this mutually agreed context state is then referred to by a unique identifier, termed the connection identifier (*CID*).

Our stateful header compression algorithm builds on the observation that in most WSN deployments, e.g. in environmental monitoring [22], the deployed network is static and only experiences changes to the topology when nodes fail or links vary in quality. However, settings are also known where the fixed infrastructure is extended by mobile nodes connecting as leaf nodes to the routing tree. To support both settings, we specifically design our algorithm with options to cater for mobile behavior, such as persons carrying sensor nodes while walking through the WSN deployment. Clearly, the integration of mobile nodes poses a set of challenges; whenever a mobile node changes its position, changes might take place in the set of its neighbor nodes and thus the available CIDs. To avoid CID collisions when a node enters a new environment, we propose a two-stage approach for the connection establishment, specifically adapted to scenarios where fixed gateways and mobile leaf nodes are present. The corresponding compression states are presented in detail in Sec. IV-A. We assume that nodes have local knowledge about their mobility, defined using a flag in the application software. However, further means could also be used to set this mobility flag, such as GPS information, acceleration, or the extent of changes in the set of neighboring nodes over time.

Finally, error recovery is a crucial point in stateful HC. When packet losses or transmission errors occur during establishing a context, the entire HC session is impacted, possibly degrading the packet reception rate at the receiver to zero. We dedicatedly address error recovery by applying framed referential coding [23], a mechanism where a reference frame is transmitted uncompressed, and followed by a series of incremental differential packets that allow reconstruction of the original value when combined with the reference frame.

IV. Stateful Header Compression for WSNs

Being stateful, our mechanism requires to establish a context, containing a reference header and the allocated CID, prior to the exchange of data. In brief, once the context has been established, all packet headers are replaced by the corresponding CID, and only differential information on the changed fields is inserted to successfully apply framed referential coding.

A. Compression States

Our implementation of SFHC.KOM is based on the state machine depicted in Fig. 1. After initialization, all nodes start in the C-NONE state, where no header compression is applied. However, once a packet transfer over a link without established context is requested by the application, SFHC.KOM directly tries to establish a C-DEST context with the destination node, and on success enters the C-DEST state, where all header fields except the destination address are contained in the context. If the context negotiation fails, e.g. because header compression is unsupported at the destination node, the C-NONE state is maintained.

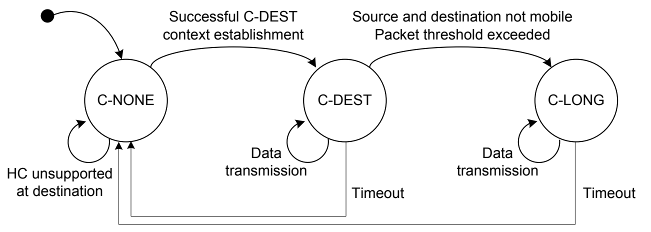


Fig. 1. State diagram of the header compression algorithm

The C-DEST operation is maintained through regular data transmissions. Only when no packets have been transferred for an extended amount of time[1], a context loss is assumed and the state machine reverts to the C-NONE state. The transition to the C-LONG mode, where all header fields are contained in the context, is only taken when both source and destination node have static positions and the number of packets transferred over the connection has exceeded a threshold th, which should be adapted to the average node degree. In our implementation, we have used a value of 20 for th. Analog to C-DEST, the efficient C-LONG encoding is used on all packets until a timeout occurs. If the C-LONG is not supported at the destination node, or the context establishment does not complete successfully, the nodes remain in the C-DEST state.

The motivation to distinguish between two separate modes of operation is based on the impact of node mobility and thus the expected duration of the connection. When mobility is

[1]The timeout values need to be aligned to the envisioned usage scenario.

Algorithm 1 Actions taken on packet transmission request

```
$d$: address of destination node
$c$: counter for number of packets sent to $d$

if $d$ is element of $L_{LONG}$ then
    send packet using established C-LONG context
else if $d$ is element of $L_{DEST}$ then
    send packet using established C-DEST context
    increment $c$
    if $c > th$ and source node is not mobile then
        try establishing C-LONG context
        reset $c$
    end if
else if $d$ is element of $L_{NONE}$ then
    send packet uncompressed
    increment $c$
    if $c > th$ then
        try establishing C-DEST context
        reset $c$
    end if
else
    send packet uncompressed
    try establishing C-DEST context
end if
```

present within the network, and corresponding nodes change their location and thus their neighborhood frequently, complex context establishment processes are obviously inapplicable due to their overhead. On the contrary, when the topology is fixed, a more complex negotiation protocol can be applied for connection establishment, as the additional expenditure is expected to be counterbalanced by the reduced sizes of packet headers throughout the remainder of the communication session. In our approach, the current state of a connection is determined by its duration. When a connection will only be short-lived (such as because of the mobility of the participating nodes), greater overhead might not be amortized throughout the message exchange and is thus undesirable. In contrast, when a fixed topology is given, and not expected to change over time, a greater overhead might be tolerable, when subsequent packet headers can be compressed with higher gain.

When considering the three possible HC states, it becomes clear that the header compression module on each node needs to maintain a table containing the state of all connections. Once a message transfer to any node has taken place, its address is entered into one of three lists. In list L_{NONE}, all nodes without support for header compression are stored. List L_{DEST} contains the addresses of all nodes with which a C-DEST context has been established, and list L_{LONG} all nodes for which a C-LONG context is present. While the necessity for the latter two tables is obvious, maintaining L_{NONE} is essential to avoid periodic attempts to establish a context with nodes without support for header compression. When a packet transmission is triggered by the application software, the destination address is extracted and Algorithm 1 executed.

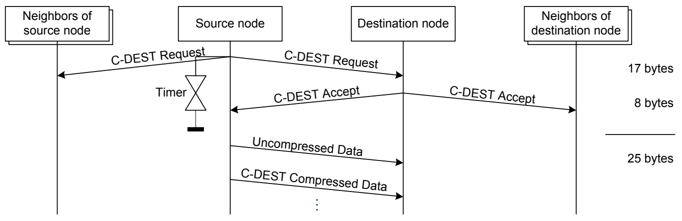


Fig. 2. Initialization of C-DEST mode

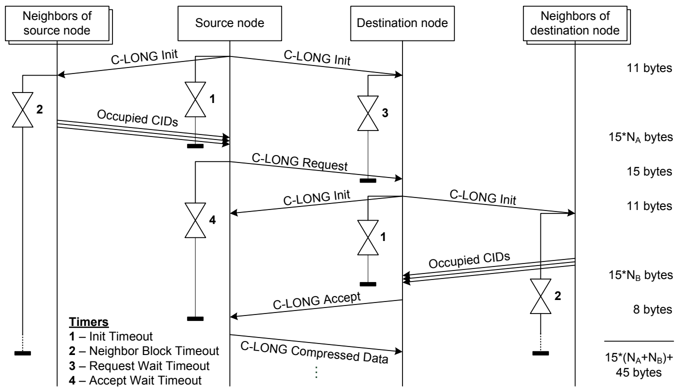


Fig. 3. Initialization of C-LONG mode

B. Assignment of Connection Identifiers

The C-DEST mode has been designed to have a short context establishment phase with low overhead. Its CID negotiation operation is shown schematically in Fig. 2. In C-DEST, the address of the destination node is retained in the packets to allow for unique addressing even when mobility is present. All CIDs are thus only relevant in combination with the destination address, which is contained in a dedicated two byte header field. Given the presence of the destination address in the compressed headers, all nodes are addressed in a unique manner, so the process of context establishment reduces to the efforts of finding a free CID at both nodes. As shown in the figure, the sender node therefore broadcasts a C-DEST request packet, containing the destination address and a bit vector containing its available CIDs. The destination node determined the smallest available CID from the received packet and its locally available CIDs, and confirms the C-DEST with an accept message, containing the selected CID. The context state is then confirmed by sending a single packet with uncompressed header to establish the context. When a C-DEST context cannot be established (e.g., when no CIDs are available or header compression is not supported), a timer started when sending the request packet terminates the attempt to set up the C-DEST context.

Due to the greater overhead induced by negotiating a C-LONG context, it is only triggered when a sufficient number of packets have been exchanged, indicating that a long-lived session is present. When the initiating node is not marked as mobile, it can start the negotiation process of the C-LONG mode, which is depicted in Fig. 3. The initiating node starts the process by sending an initialization packet. It contains the desired destination node as well as the request to receive all of its neighbours' occupied CIDs. At the same time, it starts a timer (Timer 1) for the initialization timeout, at whose expiry it assumes that all occupied CIDs have been returned. All neighbors, not including the destination node, return the bit vector containing their occupied CID numbers, and start a timer (Timer 2) themselves, during which they do not initiate an own C-LONG negotiation. At the reception of a C-LONG init message, the destination node starts a third timer (Timer 3), during which no other context establishment is undertaken. It is only canceled when a request message is received, containing all CIDs which are available at both the sender as well as all his neighbor nodes. Once the destination has successfully received the request message from the source, containing the list of available CIDs, timer 3 is stopped.

When sending a request message, the source node also starts an expiry timer (Timer 4), during which the C-LONG establishment must have completed successfully, indicated by the reception of a C-LONG accept packet. The destination node however only sends this accept packet once it has collected all of its neighbours' occupied lists to choose a unique CID available at source and destination as well as all of their neighbors. The smallest available CID from both the request message and the destination's neighboring nodes is then returned to the initiator in the accept message, which concludes the initialization process. As the C-LONG state is only reached when a C-DEST context has been established, all C-LONG messages are compressed using the C-DEST context. In case no CID is available, a reject message is sent to notify the sender to remain in C-DEST state. At a later time, the establishment of C-LONG can then be retried.

To increase the reliability of context negotiations, link-layer acknowledgements are used for all C-LONG packets that contain bit vectors of occupied CIDs, as well as for the request and accept messages in both modes.

C. Compressed Header Structure

To differentiate between whether C-DEST and C-LONG are used in the headers, the frame control field (FCF) is used. The IEEE 802.15.4 standard [13] uses the first three bits of the FCF to distinguish packets types defined in the standard, but declares one bit as reserved. We exploited this property to differentiate between compressed and regular packets. Since packets with the reserved bit set are being discarded by devices adhering to the standard, compressed packets do not interfere with uncompressed traffic. The structure of the overall headers for the different modes are shown in Figs. 4-6. The **Frame Control Field (FCF)** controls the structure of the following header fields. Both C-LONG and C-DEST reduce the FCF to only six bits, which determine the following header structure (i.e., C-DEST vs. C-LONG, the presence of an AM type, etc.) and the interpretation of the following fields (data, acknowledgement, or context establishment traffic). The **Data Sequence Number (DSN)** allows detection of duplicated frames. To achieve a robust algorithm and maintain the capability of duplicate detection, we apply framed referential coding on the DSN. The Delta-DSN-field is 4 bits long, and in consequence, reference frames are sent every 15 packets the latest. Following the DSN, the **Connection Identifier (CID)** is transmitted. We have selected a six bit wide CID range, allowing for up to 64 contexts to be established in parallel. The length of the CID field limits the number of connections a node can handle and determines the size of the context tables used for header compression. The **AM type** is not part of the IEEE 802.15.4 header, but is specific to the TinyOS operating system and comparable to the notion of ports in TCP or UDP. However, as the number of applications being used concurrently in WSNs is generally limited to a very small number, we base our algorithm design on the assumption that the AM type will change rarely. Finally, all fields remaining static in the uncompressed header with respect to a given link are being omitted and are thus no longer part of any of the compressed headers (Destination PAN ID, Source MAC address, and parts of the original FCF). These static fields are stored in the context table and can thus be reconstructed using the CID extracted from a received packet and the context table stored by the receiving node. The CID in C-LONG is thus responsible for context identification (decompression) and destination addressing, which makes unique CID allocation in this mode particularly important.

Both mechanisms have timeout values when setting up a compression context. In case the recipient of the initialization message cannot interpret the header fields accordingly, no response is given, and the timer will automatically cancel the context establishment procedure. The node is then entered into L_{NONE}. Additionally, a separate timer flushes entries from the table periodically to recover from situations where the initialization packet was lost and the node has thus been interpreted as not responding. For reference, we compare the header sizes of our algorithm over standard TinyOS header and headers with 6LoWPAN HC applied in Table I. While C-DEST reaches size of up to 60%, the C-LONG mode can even reduce packet headers by up to 80% of their original size.

When modeling the packet size reductions analytically, and also taking the required overhead for establishing the context into account, the results shown in Fig. 7 are obtained. Including the reference frame transmitted whenever the four bit wide DSN field reaches its wraparound point, it becomes clear that the additional packets required to establish a context amortize after a number of transferred packets with compressed headers. For C-DEST this is the case after only five transmissions, while in C-LONG between 20 and 67 packets are required (depending on the number of immediate neighbor nodes) to make up for the overhead of context establishment under the assumption that no losses occur on the radio channel.

TABLE I
COMPRESSION RATIOS FOR SELECTED HEADER TYPES

Header type	Header length	
	with AM type/port	without AM type/port
TinyOS header	80 bits	72 bits
6LoWPAN HC	56 bits	48 bits
C-DEST header	40 bits	32 bits
C-LONG header	24 bits	16 bits

```
n  Byte  4*n            4*n+1          4*n+2          4*n+3
   +-------------+--------------+--------------+--------------+
0  |   Frame Control Field      |     DSN      |  Dest. PAN   |
   +-------------+--------------+--------------+--------------+
1  |  Dest. PAN  | Destination MAC address     | Source MAC..
   +-------------+--------------+--------------+--------------+
2   ..Source MAC |   AM type    |
   +-------------+--------------+
```

Fig. 4. IEEE 802.15.4-compliant header with (optional) AM type

```
n  Byte  4*n            4*n+1          4*n+2          4*n+3
   +-------------+--------------+--------------+--------------+
0  | 6bit FCF | 6bit DSN | CID |  Destination MAC address    |
   +-------------+--------------+--------------+--------------+
1  |   AM type   |
   +-------------+
```

Fig. 5. C-DEST compressed packet header with (optional) AM type

```
n  Byte   0              1              2
   +-------------+--------------+--------------+
0  | 6bit FCF | 6bit DSN | CID |   AM type    |
   +-------------+--------------+--------------+
```

Fig. 6. C-LONG compressed packet header with (optional) AM type

D. Leaf Node Mobility and Packet Losses

While many of the existing header compression mechanisms targeted for the Internet assume a lossless channel, this assumption does not generally hold true in WSNs [24], [25]. The less reliable character of data transmissions in WSNs necessitates the design of HC algorithms with dedicated redundancy and/or error recovery mechanisms, which is generally only present in stateless HC mechanisms, or when specifically addressed. To recover from losses of data packets, we have applied framed referential coding and defined our headers to optionally contain the AM type, such that SFHC.KOM can tolerate losses of any data packet except for the periodic context updates. To avoid loss propagation, i.e. allow for reconstruction of all transferred differential packets, reference packets should thus be transferred using acknowledgements.

Arango et al. have shown in [17] that context losses can occur frequently when nodes are mobile. Especially when short-lived neighborhoods with low traffic data rates are given, the overhead for context negotiation can even introduce additional energy requirements to cater for the compression setup. Similarly, changes to the packet routing can be manyfold. SFHC.KOM however allows HC contexts, set up in a hop-by-hop manner, to be used by multiple data streams. Additionally, mobile nodes are confined to using the C-DEST mechanism, which relies on the combination of destination address and CID to define the unique context reference, reducing the overhead for context establishment.

V. Evaluation

To prove the algorithm's applicability, we have implemented SFHC.KOM in the TinyOS operating system and performed a set of evaluations, which we present in this section.

A. Setup

We base our evaluation on the assumptions that sensor nodes have strong limitations in terms of energy and computational capabilities, such as given for the TelosB platform [26], which we use in all conducted simulations and experiments. Following the structure of many environmental monitoring scenarios, such as PermaSense [22], we assume a tree-based network topology, where all sensor data is forwarded to the root node. As radio transmission ranges are limited, we assume that nodes are confined to communications with their immediate neighbors only. To forward data to the sink, the multi-hop tree-based Collection Tree Protocol (CTP) [4] is used.

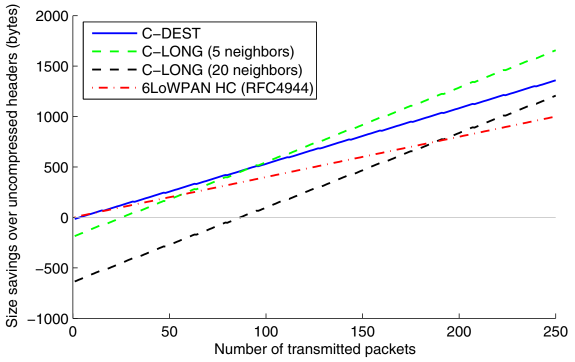


Fig. 7. Analytical model of savings when using C-LONG and C-DEST

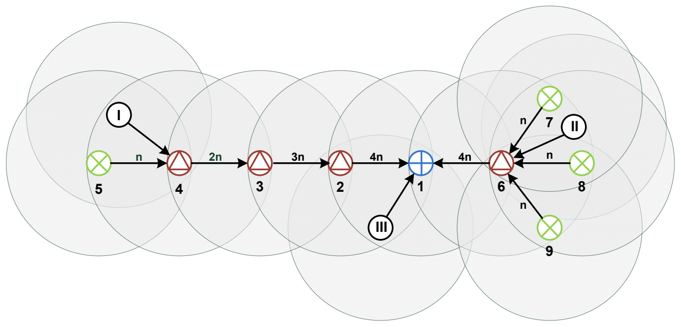


Fig. 8. Topology used for all conducted experiments

Our evaluation is based on the topology depicted in Fig. 8. A chain of nodes (labelled node 1 to 5) hereby simulate tree parts with a great depth, and are thus useful to analyze the capabilities for data transfer over long distances. In contrast, a cluster of nodes (labelled 6 to 9) represents a wide tree, where a node (node 6) must aggregate and forward the data of a number of nodes. Three mobile node positions (labelled I, II, and III) are depicted in the figure, but only used for the evaluation of our algorithm under the assumption that a hybrid network is present, comprised of a static network deployment and a single mobile node that takes one of the possible locations, which we discuss in Sec. V-D.

For test data, we have used data packets that allowed a relatively simple checking of sending success at the sink. As header compression operates independently of the packet payloads by design, we have selected the payload to carry a two byte sequence number as well as the source address to allow backtracking packets to their origin. The CTP header added another 8 bytes to the payload, so that the overall payload size was 12 bytes. Packets were transmitted periodically at an interval of 2,500ms to eliminate the likeliness of congestion at the sink. The application was run for 1,600 seconds in

TABLE II
SAVINGS WHEN APPLYING HC IN A SCENARIO WITH ONE MOBILE NODE

Pos.	Time in range	Emitted data	HC overhead	Savings
I	3.9 sec	19 bytes	17 bytes	-10 bytes
II	14.3 sec	95 bytes	17 bytes	18 bytes
III	306.8 sec	2318 bytes	17 bytes	837 bytes

simulation time, during which an overall number of more than 10,000 packets were emitted, with a number exceeding 600 packets at each source node, excluding the initializiation packets sent by the CTP protocol. Our implementation of SFHC.KOM required 13,786 bytes of program memory and 2,540 bytes of RAM, each representing about 25% of the TelosB's available resources. However, the tables containing the context state information thereby represent the major contribution to the resource demand, and a linear decrease in RAM demand of around 16.6 bytes per entry can be observed when reducing the number of CIDs used.

B. Static Network Topology using C-DEST

In this setup, we have assumed a topology as depicted in Fig. 8, but configured all nodes to assume that they are mobile. This disallows them to enter the C-LONG mode, and thus enables dedicated measurements when only the C-DEST mode is used. After the CTP tree had been established, we have initiated the periodic transmission of packets, with each first transmission initiating the establishment of a C-DEST context. We have measured the time required to establish the C-DEST context to be 47ms on average, confirming our assumption that context negotiation is performed quickly and sufficiently small timeout values can be selected. Even in dense networks, the nodes can thus create the required contexts within reasonable time. Detailed results on the overall volume of transmitted data and the corresponding types are indicated in Fig. 9(a), differentiating between unicast management traffic (such as CTP initialization and unicast messages employed for context establishment), broadcast messages for the same purpose, and data volume transferred in C-DEST and C-LONG modes. The figure clearly shows the prevalence of C-DEST traffic as well as the uncompressed broadcast messages accompanying the context establishment process.

C. Static Network Topology using C-LONG

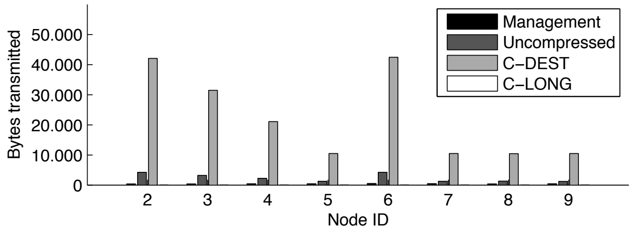
In a second simulation, we have configured all nodes to assume that they are statically located. This allows all of them to enter the C-LONG mode after successful transmissions have taken place using C-DEST. Establishing the C-LONG context consumes 5,249ms on average, however mostly influenced by the timeout values at both source and destination when having requested their neighbors to transmit their available CID vector, which are set to 2,500ms. When comparing our HC algorithm over the transmission of uncompressed packets, header size reductions of 74.7% are observed, effectively reducing the headers from 10 to 2.53 bytes. To allow direct comparison, the results are compared the C-DEST mode in Fig. 9(b). Notably, the greater management overhead is counterbalanced by a smaller data volume transferred using C-LONG, clearly exposing the benefits of reduced header sizes.

D. Hybrid Topology with a Mobile Node

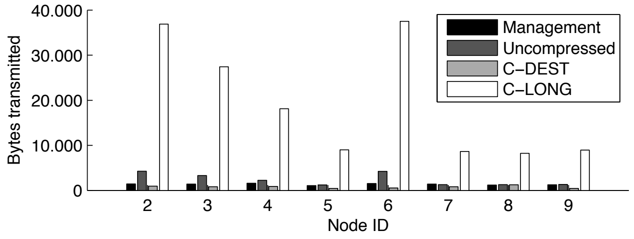
To assess the algorithm's behavior when a static network topology under presence of a mobile leaf node is given, we have assumed the same setting as shown in Fig. 8 and marked all nodes as static, allowing them to use the C-LONG mode. An additional mobile node (labelled node 10) was inserted into the network, and assigned three different positions within the topology in successive time steps. The mobile node was placed at positions I, II, and III to establish a context with the corresponding nodes 4, 6, or 1, respectively. The results for the size savings are shown in Table II, and indicate that already after a number of 5 packets successfully transmitted in C-DEST mode, equalling a duration of less than 15 seconds within range of the static node, savings in terms of the overall transmission sizes are gained. Due to the use of the C-DEST mode, this threshold value is also independent from the number of neighbors the node has.

E. Energy measurement

To verify that not only reductions in the actual packet sizes, but also in the required energy, are given, we have used COOJA/MSPsim [27], which is known to return accurate energy estimates, to evaluate the energy demand of our solution. We have set up a second scenario with a single sender and receiver node each. The receiver was configured to use the NullMAC implementation, i.e. the radio transceiver was always in an active state. To monitor the energy consumption of the sending node better, we have allowed it to be put into sleep immediately after each transmission to better observe the overall energy required to process and transmit a packet with 4 bytes payload size. To achieve values for our energy expenditure estimation, we have set the packet transmission interval to 2 packets per second and run the experiment for 100 seconds. Results for this experiment are shown in Table III.



(a) Total number of bytes transmitted in C-DEST mode



(b) Total number of bytes transmitted in C-LONG mode

Fig. 9. Evaluation of the C-DEST and C-LONG compression modes

TABLE III
RADIO AND MCU ENERGY EVALUATIONS USING COOJA/MSPSIM

	Radio energy	MCU energy	Total	Savings
C-NONE	13.3 mJ	26.2 mJ	39.5 mJ	0%
C-DEST	11.2 mJ	26.9 mJ	38.1 mJ	3.5%
C-LONG	10.1 mJ	26.8 mJ	36.9 mJ	6.6%

It becomes clear that the energy required by the radio transceiver in transmission mode reduces from 13.3mJ to 11.2mJ, representing radio energy savings of 16% in the C-DEST mode. When C-LONG is used, the consumption even drops to 10.1mJ, a decrease of 24% over uncompressed transmission. However, these savings are counterbalanced by the operations necessary to perform the compression step; these are however energetically less demanding, so overall savings of up to 6.6% can still be observed. The minor drop in the MCU energy consumption between the C-DEST and C-LONG modes originates from the fact that reconstruction of the full header is performed identically in both variants, but the shorter durations for the necessary communication between MCU and radio device leads to a smaller energy requirement.

VI. CONCLUSION

We have presented SFHC.KOM, a stateful header compression algorithm for WSNs. It features lightweight mechanisms for context establishment and maintenance, which are well suited for application on resource-constrained sensing systems. Specifically designed to operate on both mobile and fixed nodes, it achieves header size reductions of up to 60% between mobile nodes, outperforming the header compression means defined in the 6LoWPAN standard. In case the network topology is stable, headers can be reduced even further through application of the C-LONG compression mode. Thus, SFHC.KOM is suitable for application in a wide range of WSN deployments found today.

We have analyzed resource demands, performance and simulated the energy expenditure for our mechanism over a setting where no header compression is applied. Our results shows transmission energy savings of 24% and overall energy savings of up to 6.6% in a realistic setting. The small negotiation overhead in C-DEST mode, which amortizes after only five transmissions containing real data, also proves the algorithm applicable to mobile application scenarios. In a next step, we plan to evaluate the impact of the MAC protocol used, as well as analyze the performance of SFHC.KOM under the presence of large and/or highly dynamic networks.

REFERENCES

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

[2] A. Reinhardt, D. Christin, M. Hollick, J. Schmitt, P. S. Mogre, and R. Steinmetz, "Trimming the Tree: Tailoring Adaptive Huffman Coding to Wireless Sensor Networks," in *Proceedings of the 7th European Conference on Wireless Sensor Networks (EWSN)*, 2010.

[3] D. Farber, G. Delp, and T. Conte, "Thinwire protocol for connecting personal computers to the Internet," RFC 914 (Historic), 1984.

[4] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009.

[5] N. El Rachkidy, A. Guitton, B. Bakhache, and M. Misson, "PiRAT: Pivot Routing for Alarm Transmission in Wireless Sensor Networks," in *Proceedings of the 34th IEEE Conference on Local Computer Networks (LCN)*, 2009.

[6] V. Jacobson, "Compressing TCP/IP Headers for Low-Speed Serial Links," RFC 1144 (Proposed Standard), 1990.

[7] M. Degermark, B. Nordgren, and S. Pink, "IP Header Compression," RFC 2507 (Proposed Standard), 1999.

[8] K. Sandlund, G. Pelletier, and L.-E. Jonsson, "The RObust Header Compression (ROHC) Framework," RFC 5795 (Proposed Standard), 2010.

[9] V. Suryavanshi and A. Nosratinia, "Error-Resilient Packet Header Compression," *IEEE Transactions on Communications*, 2008.

[10] J. Lim and H. Stern, "IPv6 Header Compression Algorithm Supporting Mobility in Wireless Networks," in *Proceedings of the IEEE SoutheastCon*, 2000.

[11] C. Westphal and R. Koodli, "Stateless IP Header Compression," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2005.

[12] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), 2007.

[13] 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.

[14] G. Kulkarni, C. Schurgers, and M. Srivastava, "Dynamic Link Labels for Energy Efficient MAC Headers in Wireless Sensor Networks," in *Proceedings of the 1st IEEE International Conference on Sensors*, 2002.

[15] M. Kaddoura and S. Schneider, "SEEHOC: Scalable and Robust End-to-End Header Compression Techniques for Wireless Ad Hoc Networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, 2004.

[16] C. Westphal, "Layered IP Header Compression for IP-enabled Sensor Networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2006.

[17] J. Arango, S. Pink, S. Ali, D. Hampel, and S. Dipierro, "Header Compression for Ad-Hoc Networks," in *Proceedings of the IEEE Military Communications Conference (MILCOM)*, 2005.

[18] O. B. Akan and I. F. Akyildiz, "Event-to-Sink Reliable Transport in Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, 2005.

[19] T. Le, W. Hu, P. Corke, and S. Jha, "ERTP: Energy-efficient and Reliable Transport Protocol for data streaming in Wireless Sensor Networks," *Computer Communications*, vol. 32, no. 7-10, 2009.

[20] A. Dunkels, J. Alonso, and T. Voigt, "Making TCP/IP Viable for Wireless Sensor Networks," in *Work-in-Progress Session of the 1st European Workshop on Wireless Sensor Networks*, 2004.

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

[22] 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 *Proceedings of the 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2009.

[23] P. Seeling, M. Reisslein, T. Madsen, and F. H. P. Fitzek, "Performance Analysis of Header Compression Schemes in Heterogeneous Wireless Multi-Hop Networks," *Wireless Personal Communications*, vol. 38, no. 2, 2006.

[24] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin, "Statistical Model of Lossy Links in Wireless Sensor Networks," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[25] J. Zhao and R. Govindan, "Understanding Packet Delivery Performance In Dense Wireless Sensor Networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[26] *TelosB Datasheet*, Crossbow Technology. [Online]. Available: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf

[27] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, and P. J. Marrón, "COOJA/MSPSim: Interoperability Testing for Wireless Sensor Networks," in *Proceedings of the 2nd International Conference on Simulation Tools And Techniques For Communications, Networks And Systems (Simutools)*, 2009.