Protocol-Agnostic Compression for Resource-Constrained Wireless Networks

Travis L. Massey, Ankur Mehta, Thomas Watteyne, Kristofer S.J. Pister  
BSAC, University of California, Berkeley, USA  
{tmlmassey,mehtank,watteyne,pister}@eecs.berkeley.edu

Abstract—Reducing the time the radio is on in wireless devices results in lower power consumption, so sending the same data in fewer bytes can greatly extend the lifetime of a network. In this paper, we explore the use of protocol-agnostic packet compression, a technique orthogonal to current explicit compaction techniques. Because it functions as a transparent layer inside a communication stack and makes no assumption about the specific protocols used, it is generic enough to be used on multiple technologies.

Compression is performed by identifying patterns in recent packets and replacing those patterns with bit flags in the transmitted packet. We present the results of compressing actual packet traces collected from several commercial networks using this algorithm and discuss the resource trade-offs of the algorithm. Results indicate compression ratios between 40% and 80%, yielding predicted energy savings of 30-70% in a typical time-synchronized network.

I. INTRODUCTION

The application space for resource constrained wireless networks covers fields as diverse as building automation, industrial monitoring, body sensor networks, home electronics, computer interfacing, and energy applications. Because of the broadening conditions under which they have been demonstrated to operate reliably, these networks have received significant attention in the last decade. Standardization bodies such as the IETF and the IEEE are standardizing the protocol stacks of tomorrow’s networks.

Removing the need for wires to network smart objects also means that most of these devices become battery powered. Luckily, standards such as IEEE802.15.4 [1] have been developed for low-power radios such as the ubiquitous CC2420 [2], which consumes about 65mW when on. If left on all the time, a wireless device equipped with such a radio would have a lifetime of only 5 days on a set of two AA batteries. To further improve lifetime, smart medium access control (MAC) protocols can tune the duty cycle of the radio to below 1%, yielding lifetimes of multiple years.

IEEE802.15.4 is the de-facto standard for low-power radios, with compliant radios currently equipping nearly all resource-constrained wireless networks. Fig. 1 shows the energy consumed over time when a packet is sent from an eZ430-RF2500 wireless sensor node. At time $t = 0$, the radio turns on its voltage regulator and waits for its oscillator to settle to the proper frequency before transmission can begin. This startup process consumes $5.2\mu C$. Fig. 1 was measured when the radio transmitted a full 128-byte-long packet. To transmit a full packet, the radio consumes $5.2\mu C + 27.0\mu C + 63.2\mu C = 95.4\mu C$.

\[ C = \frac{L_{\text{original}} - L_{\text{compressed}}}{L_{\text{original}}} \]  

This paper explores protocol agnostic packet compression, which relies on two principles:

- the technique used is agnostic to the structure of the packet being compressed. No assumption is made regarding the contents of the headers. This is a key difference from header compaction described in Section II-B;
- while two nodes maintain common state to compress/decompress properly, there is never any explicit state exchange. Instead, each node builds an identical state locally. No additional network traffic is needed.

The technique presented in this paper is founded upon pattern recognition, in which multiple-byte patterns are replaced by single-bit flags. All nodes, based on a buffer of previously transmitted/received packets, use a simple algorithm to identify sequences of consecutive bytes recurring in the packets. These sequences are called patterns and are stored in a pattern list. A transmitting node, when asked to transmit a packet, removes the previously identified patterns from that packet and replaces them with a set of one-bit flags indicating which patterns were removed. This set of flags, call the tag, is transmitted at the beginning of the packet. A receiving node decompresses the packet by inserting the patterns indicated in the tag.

The severe resource constraints of wireless sensor nodes

![Fig. 1. A 70% compression ratio translates into 66% less energy consumed.](chart.png)
present significant challenges in the implementation of a
pattern-based compression algorithm. Typical wireless sensor
nodes have very little memory; for example, the ubiquitous
TelosB mote features only 10kB of RAM and 48kB of ROM
memory – very little space for the state maintained by the
algorithm. Moreover, because we do not want to explicitly
transmit the shared state (i.e. the pattern list), this list must
be built identically on both ends of a wireless link. This
differs from the standard approach used in dictionary-based
algorithms used in desktop file compression tools.

The ways of overcoming those challenges are described in
Section III, which also presents a practical and simple agnostic
packet compression method. Prior to that, Section II details re-
lated work in the field. Section IV presents our implementation
of the compression scheme, as well as experimental results on
two different packet traces collected from commercial wireless
sensor networks. Section V discusses potential improvements
to the compression algorithm. Section VI concludes this paper.

II. EXISTING COMPRESSION TECHNIQUES

A. Agnostic Compression Techniques

Compression techniques that are agnostic to the content of
the data typically use dictionaries. When processing a stream
of bytes, the compression algorithm recognizes often-repeated
patterns, stores those patterns into a dictionary, and indicates
into the stream of data where those patterns are. Patterns are
typically identified by their key, i.e. the rank of the pattern in
the dictionary. The actual pattern is only written once in the
dictionary, and replaced multiple times by small keys. This
idea is used in popular file compression tools such as zip, gzip
and winzip. The output file of these algorithms is made up of
two parts: the dictionary (typically written at the beginning of
the file) and the compressed data.

In this work, because an individual packet is so small, it
makes little sense to prepend the packet with its dictionary.
Instead, for each link, each node has a local copy of the dic-
tionary (which we call pattern list), and only the compressed
data is sent over the air. Moreover, instead of indicating the
keys of the patterns inline with the data, a tag consisting of a
set of binary flags indicating whether or not the corresponding
pattern has been compressed out of the packet is prepended
to the data.

B. Explicit Compaction Techniques

Data compaction, which depends on a priori knowledge of
the format, has appeared in limited use previously. An early
proposal by Van Jacobson (VJ) sought to increase the useful
throughput on low data rate serial modem connections [3]
by removing the information that remains constant over the
duration of the connection, including addresses, ports, and
offsets from TCP/IPv4 headers. The header is further pared
down as information that can be determined from other fields
or other layers is removed.

Robust Header Compression (ROHC) [4] demonstrates im-
proved robustness over links with high error rates, specifically
those in wireless networks, and generalizes the compaction al-
gorithm to several common protocols. ROHC is more dynamic
than VJ header compression in that it gradually progresses the
state of the compactor based upon its "confidence" that it can
predict each successive header. Major limitations of ROHC
are the large quantity of memory required to support many
simultaneous contexts and the required memory bandwidth.

The Internet Engineering Task Force (IETF), through its
working group IPv6 over Low power Wireless Personal Area
Networks (6LoWPAN) has proposed a header compaction scheme for transmitting IPv6 packets in IEEE802.15.4 net-
works. While early versions of the technique are referenced in
[5], it has been developed into an IETF standard [6]. Through
deep understanding of the interaction between IEEE802.15.4,
IPv6, and UDP, 6LoWPAN is able to remove fields which
are redundant among those headers, thereby reducing the size
of the packets being transmitted over the air. Fig. 2 shows
a packet compacted using 6LoWPAN assuming the simplest
case of a point-to-point link.

6LoWPAN removes a number of fields in the IPv6 and
UDP headers because they take well-known values, or because
they can be inferred from fields in the IEEE802.15.4 header.
The drawback of a technique such as 6LoWPAN is that it
relied upon deep understanding of the header stack, in this
case UDP over IPv6 over IEEE802.15.4, and hence cannot act
as a transparent layer. Moreover, the compaction only applies
to routing and transport headers, leaving the MAC header,
application payload, and 6LoWPAN-specific header fields.
Fortunately, explicit compaction and agnostic compression are
not mutually exclusive and can be used simultaneously.

III. PROTOCOL-AGNOSTIC COMPRESSION FOR WIRELESS
NETWORKS

A. Overview

Upon receiving a packet from the upper layer, the compres-
sion algorithm compares that packet against the pattern lis-
t and identifies which patterns are present in the packet. It then
transmits a shorter version of the packet, where patterns have
been replaced by flags in a tag attached at the beginning of
the packet. Upon reception, the receiving node performs the
inverse operation.

In order for this compression technique to be applied, a node
must maintain the following state for each of its neighbors for
both transmission and reception:

- a pattern list. This contains the patterns that have been
  identified by the pattern discovery engine (detailed in
  Section III-C) used to compress a packet. The pattern
  list is empty when a node is first switched on.
- a recent packet buffer. For the receiving node, this con-
  tains the most recent packets that have been received; for
  the transmitting node, this contains the latest packets that
  have been sent. This buffer is also initially empty.

Three parameters are required to tune the algorithm:

1Note that RFC4944 [6] will probably be made obsolete by [7], a work-
in-progress by the IETF.
Fig. 2. Representative packet compacted by 6LoWPAN. IPv6 and UDP headers are compressed from 48 bytes to 7 bytes.

- $B$, the number of entries in the packet buffer.
- $P$, the number of entries in the pattern list.
- $S$, the minimum size of a pattern, in bytes.

Note that $B$, $P$ and $S$ are common for all nodes in the network and do not change with time. While this section describes the algorithm, Section IV discusses how to set those parameters based upon experimental results from real-world traces.

A pattern is defined as a contiguous sequence of bytes at a specific location inside the packet. Because patterns are identified only based upon recurring byte sequences, this technique is agnostic to the protocols and can be extended to technologies other than IEEE802.15.4.

The transmitter and receiver must maintain the same patterns in their pattern list without explicitly transmitting signaling information over the air. The algorithm for building the pattern list is described in Section III-C.

A packet to be transmitted is sent down from the upper layer, and is compressed transparently before being passed to the lower layer (see Fig. 3). Compression proceeds according to the following 4 steps:

1) Before the packet is compressed, a CRC is computed on its uncompressed form. This will be appended to the compressed packet and used to verify correct decompression.

2) The packet is compared against the patterns in the pattern list. Known patterns identified in the packet are removed from the packet, and the corresponding flag in the tag is raised.

3) The $P$-bit (typically 8, 16 or 24) tag is prepended to the compressed packet. The CRC is appended, and the resulting data is sent to the lower layer.

4) The state is updated with the uncompressed packet.

Decompression proceeds according to the following 4 steps:

1) The receiving node reads the tag and inserts the corresponding patterns into the compressed packet to form the candidate decompressed packet.

2) The receiver recomputes the CRC on the candidate packet. If this packet passes the check, it is used as the decompressed packet. If CRC fails, the packet may not have been not compressed in the first place and the initially received packet is used as-is.

3) The decompressed packet is passed on to the upper layer.

4) The state is updated with the decompressed packet.

The state update is identical for both the transmitter and receiver, and is performed using the same uncompressed packet. Given identical a priori states, the following 2 steps will yield the same a posteriori states:

1) The uncompressed packet is analyzed by the pattern discovery engine. Possible new patterns are discovered by comparing the current packet with the ones in the recent packet buffer (details in Section III-C). If a new pattern is found, it will be added to the pattern list.

2) The uncompressed packet is inserted into the recent packet buffer, replacing the oldest packet.

B. Data Representation

The recent packet buffer is composed of $B$ arrays of bytes, each as long as the maximum packet length (128 bytes in the case of IEEE802.15.4). The packets are stored in this buffer with a first in, first out (FIFO) replacement policy.

A pattern, a contiguous sequence of bytes at a given position in a packet, is represented by its starting position, length, and contents. Patterns stored in the pattern list follow a least recently used (LRU) replacement policy.

2Note that this CRC field is used solely to detect successful decompression; lower layers typically use their own CRC to detect transmission errors.
C. Pattern Discovery

The goal of the pattern discovery engine is to find patterns while maintaining low complexity. Pattern discovery consists of a byte-wise comparison between the current packet and each of the packets in the recent packet buffer. When the number of consecutive byte matches is greater than or equal to the minimum pattern size $S$, the corresponding sequence of bytes is added to the pattern list (duplicates are avoided). The minimum pattern size $S$ ensures that the pattern list is not populated by numerous short patterns at the expense of longer patterns which yield greater compression. Note that pattern discovery is only performed on non-compressed packets.

IV. IMPLEMENTATION AND EXPERIMENTAL RESULTS

A. Data collection

We test the compression algorithm by using IEEE802.15.4 packet traces collected from two commercial wireless sensor networks; 62,101 packets were collected from company A and 63,988 packets were collected from company B\(^3\). Packets were recorded using Integration’s IEEE802.15.4 USB Dongle, on a single channel in the 2.4 GHz band. Company A data was recorded from a data collection network which uses 16-bit source and destination IEEE802.15.4 addresses. Company B data was recorded from a network performing numerous large file transfers; 64-bit IEEE802.15.4 addresses were used.

B. Implementation Details

The traces collected at company A and company B were run off-line through an implementation of the compression algorithm in order to predict the energy gain had these networks been running protocol agnostic compression. Thus we are able to precisely quantify the gain of this scheme in real-world commercial applications, something an in-lab deployment would not allow. The algorithm was implemented for a single link per the description of Section III, sans CRC.

C. Compression Ratio $C$

Compression ratio $C$, as defined in (1), is plotted versus the parameters $B$, $S$, and $P$. As is seen in Figs. 4-6, the compression ratios are in the range of 50 ± 5% for data from company A and 75 ± 3% for data from company B. These results are consistent with the observation that the packets from company B have a relatively large static header and are generally quite repetitive given the nature of a file distribution, whereas the company A data is far less regular and, as a result, benefits less from compression.

D. Impact of the recent packet buffer size $B$

When $B$ is large, the pattern discovery engine can compare the current packet to a greater number of recent packets, and can hence find more patterns. The regions of positive slope in Fig. 4 intuitively correspond to more points of comparison from which to discover patterns useful for compression. The decrease in $C$ for large $B$ comes as a result of useful patterns being removed from the pattern list (according to the LRU replacement policy) in favor of new patterns identified from older packets, which may be less relevant than patterns identified from more recent packets. $C$ peaks for lesser values of $B$ and for shorter minimum packet lengths $S$ because patterns are more quickly cycled through the pattern list due to the increase in number of patterns being identified. As can be seen from Fig. 4 and Fig. 5, $B$ and $S$ roughly track each other. As $S$ increases to large values, eventually the compression ratio with respect to $B$ flattens out because few patterns of that length exist, but those that do exist are likely to be often used, representing a large static portion of the packet.

E. Impact of the minimum pattern size $S$

Additional insight can be gained into the relationship of $B$ to $S$ by carefully examining Fig. 5. The data from company A indicates that, for small values of $B$, the change in $C$ with $S$ is relatively smooth, while as $B$ becomes large the transition between the compression ratio for small $S$ and large $S$ becomes more abrupt. This abrupt transition occurs because above a minimum pattern size useful patterns are

\(^3\)Unfortunately, marketing considerations do not allow us to disclose the names of these companies.
no longer being ejected from the pattern list; all of the most meaningful patterns are retained. Data from company B shows little variation with B over a range of S because the patterns in that data set tend to not change quickly, and the small change in C over the entire range of S suggests that the patterns in the company B data are typically long.

Another interesting feature to note from Fig. 5 is the convergence of the curves for various B to a common C as S exceeds four bytes for company A. This indicates that the only patterns being compressed for large S are those that rarely change, so the rapidly changing pattern must be four bytes long.

F. Impact of the pattern list size P

Lastly, the pattern list size, P, was varied coarsely from 8 to 24 in increments of 8, corresponding at each step to one additional tag byte required at the beginning of the compressed packet. Fig. 6 suggests that company A data benefits from a larger pattern list, whereas company B data does not. Because company A data has a much greater degree of randomness than company B data, it can benefit from the additional pattern allowance. Data from company B changes infrequently enough that fewer than eight patterns are required for effective compression. Additional patterns beyond those required result in unnecessary tag bytes being added to the beginning of the packet; indeed, the 3% decrease in C for the company B data corresponds to two additional bytes added to a 60- to 70-byte packet. Finally, we notice also that the declining C for increasing B for company A data is less pronounced for larger P. As the number of patterns that may be stored is increased, fewer useful patterns are being removed due to the large B so C does not decrease as drastically.

G. Parameter Tuning Conclusions

The intuitive result that the compression ratio can be improved by providing the algorithm with more memory is shown to be true to the extent that the penalty of adding additional tag bytes does not outweigh the benefit of additional patterns. However, the parameters must be increased with care such that the aforementioned dangers of unbalanced parameters are avoided. That being said, it is also clear that eventually the compression ratio saturates and there is little compression benefit to further tuning the parameters. From a memory and computation standpoint, it is advantageous to remain at the edge of this saturated region, or even at a point with lower compression ratio. As is evident from Figs. 4-6, the compression ratio does not change drastically over the full range of B, S, and P; leaving each of the parameters at or near their minimum values is likely sufficient.

V. POTENTIAL IMPROVEMENTS

Robustness of the algorithm may be improved by reserving two bits from the tag as a counter to indicate the state of the pattern table. In the presence of lossy links, this will aid in the detection of the loss of synchronization between two nodes' state. If synchronization is lost, the pattern list may be flushed. While this causes a few packets not to be compressed (less than B), this does not cause any packet to be lost.

There are many possible means of improving the efficiency of the algorithm. One such approach would employ learning to reduce the width the recent packet buffer according to which bytes in a packet are commonly compressible. In the case of encrypted payloads this may present significant memory savings. A second approach would be to modify the pattern list replacement policy to consider pattern length, frequency, or overlap with other patterns.

Finally, given its simplicity, the algorithm could be translated into digital logic and fabricated on an application-specific integrated circuit. Implementing the algorithm on an ASIC reduces the cost of computation, both in time and energy, and compression becomes transparent for the programmer.

VI. Conclusion

We have demonstrated a compression algorithm for resource-constrained wireless networks that is agnostic to network protocols and which does not require explicit signaling messages to maintain shared state between communicating nodes. Furthermore, due to the use of a CRC, compressed and uncompressed packets may coexist in a network. Compression ratios above 50% and 75% were achieved on two traces gathered from commercial IEEE802.15.4 networks. This translates into energy savings of the same order in a time-synchronized network, potentially more than doubling the lifetime of the network. Agnostic compression can be used at any layer in a protocol stack, and may be used in conjunction with existing compaction techniques such as 6LoWPAN.

REFERENCES

[2] CC2420, 2.4 GHz; IEEE 802.15.4 / ZigBee-Ready RF Transceiver (Rev. B), Texas Instruments, Inc., 20 March 2007, data Sheet SWSRS041B [available online].