

Layered IP Header Compression for IP-enabled Sensor Networks

Cédric Westphal

Abstract—Sensor networks introduce new constraints on wireless and ad-hoc networks, especially in terms of bandwidth limitations and power efficiency. If such network is still using IP, for instance if it is connected eventually to the internet, using IP header compression becomes a strong necessity. IP header compression is usually a link compression between the mobile device and the base station. In a sensor network however, there is a succession of wireless links. Link based compression might be costly, as each hop has to decompress the IP headers and re-compress them for further forwarding, consuming power for each operation. We present here a technique to alleviate this issue and provide multi-hop compression. The idea is to split the compression according to the packet's different layers: the network part of the header, and the transport part of the header are compressed differently, using two orthogonal mechanisms. We introduce a signaling that allows the nodes participating in the compression to establish the compression end-to-end. The multi-hop compression might be especially beneficial in sensor networks, where typically payload is small and the cost of transmitting any bit is quite high.

Keywords: Header compression, transport layer, network layer, end-to-end compression.

I. INTRODUCTION

Sensor networks introduce new constraints on wireless and ad-hoc networks, especially in terms of bandwidth limitations and power efficiency. To reduce the bandwidth and power consumption, some sensor networks use highly optimized MAC, network and transport layers. In particular, addressing and routing specifically designed for sensor networks, while reducing dramatically the drain of the resource, prohibit connecting the sensor network seamlessly to a global network such as the internet.

Using IP within the sensor network requires some optimization, as the payload in a sensor network is typically a few bytes, and the IP header is already 20 bytes, before the transport header is added. One potential technique would be to use IP header compression.

IP header compression has traditionally focused on the compression over a resource constrained link. For in-

stance, a mobile node (MN) communicating over the air interface to its access router (AR) would use header compression to reduce the size of the IP headers. Typical schemes would focus on the IP/UDP/RTP headers [1] or IP/TCP headers [2], [5].

However, it might be necessary to extend the compression over several links. In a sensor network or an ad hoc network for instance, the cost of transmitting each bit is much higher due to tight resource constraints. The use of traditional IP header compression on each link would increase the processing power at each node and create compression states to be maintained. This is frowned upon when the power of each node is limited, and the end of the node's duty cycle might wipe out any state anyway. A compression scheme is needed to carry the compression over several hops. Unlike a flexible IP header [6], it would allow to use the existing IP stack while saving resources.

The IP header compression algorithms we mentioned above function by studying similarities between consecutive packets within a single flow, and eliminating these similarities once the behavior of the packets in the flow becomes predictable. The compressor sits at one end of the link, and the decompressor at the other end restores the full packet header for further forwarding. These compressors function in the *time domain*, as they need several packets over time to acquire the compression state and go through a transient phase before reaping the compression benefits.

Other algorithms taking a different approach have been introduced: a user-based frequency-dependent algorithm [3] makes use of the correlation between the flows for a given user. A stateless algorithm [4] makes use of the routing information maintained by the AR in its forwarding table to compress the fields that can be aggregated. These algorithms function in a *space domain*, as they consider some address space (destination addresses for a given user, or destination prefixes for the nodes attached to a given router) for compression.

The first, traditional time-domain approach, can be construed as a vertical approach: the IP stack is compressed vertically across all the fields into the compressed header. The newer space-domain approach is a horizontal ap-

C. Westphal is with the Communication System Laboratory, Nokia Research Center, Mountain View, California. E-mail: cedric.westphal@nokia.com

proach: the correlation is not within the fields of the same packets, but across the same field for different flows within a common group.

To achieve multi-hop compression, we will describe a *hybrid* of the two classes of IP header compression which makes use of the different scalability characteristics of each and allows compression over several links.

In this document, we present an architecture which combines these two orthogonal architectures. The aim is to define an IP header compression architecture that allows header compression over several links. Multi-link or multi-hop header compression is required for mesh networks, for instance. Other networks have nodes with more functionality than other (for instance, smart edge nodes, and nodes strictly focusing on efficient forwarding inside the network), and being able to compress all the way to these specific nodes is an added feature to the networks.

II. ARCHITECTURE DESCRIPTION

A. IP header compression and the 7 layers

The OSI architecture defines 7 layers: physical, data link, network, transport, session, presentation and application layer. The IP header replicates part of this hierarchy. It has no physical or link layer, but an IP header for network, a TCP or UDP (or RCP) header for transport, and the data packets carrying the session, presentation and application layers information.

As an example, a voice session will have a IP/UDP/RTP/data format for the respective network/transport/session/application. Header compression usually happens between the layer 2 and the layer 3: network and transport information are mapped onto some link layer code at one end of the link. The code is decompressed into the original network and transport application. Usually, header compression is below layer 3, as some layer 2 information is used (for instance, to identify the source of the packets). However, in this document, we focus only at the layer 3 and above.

We wish to encode information pertaining to the routing and forwarding of packets orthogonally from the information relating to the transport and the application of the packet.

It could seem that there is a greater compression benefit to compress all the fields into one byte, as opposed to some fields onto one byte, and some other fields onto another byte. However, the purpose of separating the compressed fields according to different layers is as follows:

- 1) the absolute difference between 1 or 2 or 3 bytes is minimal compared to the compression gain.

Namely, from the network point of view, the performance gain is the same if a IP/TCP header is compressed from 44 bytes to 1 or 2 bytes.

- 2) there are some benefits to preserving the layer separation, the main one being that it allows for layered compression, detailed in the next section.

B. Layered Compression

The idea for layered compression is to use two different compression schemes, each one best fitting the layer it compresses.

1) *Network layer*: Network fields have a strong correlation from one flow to the next. Also, network fields are available at the network level: these are the only fields observed by most routers. This entails that these are the only fields that a router -and a network domain a fortiori- on the path can conveniently keep track of, or monitor.

As an example, consider one example of network compression architecture in Figure 1.

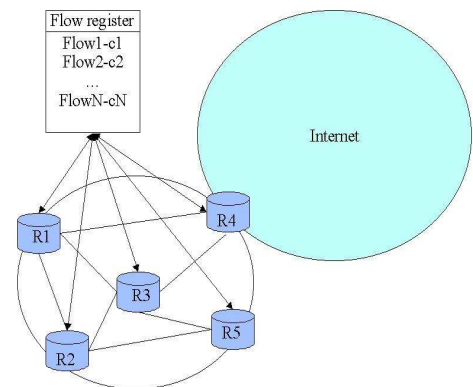


Fig. 1. A network layer compression architecture

In Figure 1, the network layer information is gathered from the flows at the router level, and send to a flow register, that compresses the flow information into a code, or a label. The flow register then makes the labels available to the router. The labels can be used inside the network to forward the packets. When the packets are about to leave the network, the last forwarding node can replace the label by the original uncompressed network header.

For illustration purpose, a specific instantiation in this particular case would be to replace the 32 bits IPv4 (or the 16 bytes IPv6) address referring to routers: R1, R2,...,R5 by the code: 1,2,...,5, which requires only 3 bits. The gateway R4 can then replace the code with the original address for packets leaving to the internet, or the IP address with the code for incoming packets from the internet.

This is a general case of asymmetric compression, where the compressor receives the code from an outside agent, and there is an asymmetry in the information required to compute the code. In this case, only the flow register has the big picture. Other possible implementation of the network layer compression would be by distributing the flow register (for instance by using forwarding table at each routers, or by computing flow tables at the link level to keep track of the most frequent/recent flows).

The network fields are the fields pertaining to the forwarding of the packets at the network level: these are destination, flow label, Ver, DS byte in IPv6, or Ver, DS byte, destination in IPv4. The state to be maintained is of the same granularity as the flow, namely it has to be maintained once per flow.

The network fields belong to the *space domain* evoked earlier.

2) *Transport layer*: Network layer is correlated between different flows. On the other hand, transport layer fields have a strong correlation from one packet to the next within the same flow. This yields two main consequences:

- the correlation can be deduced from a few consecutive packets, necessitating a transient period before compression gets effective,
- a state has to be maintained at the packet granularity: the state has to be updated after each packet, and so does the compression code.

The second consequence makes the compression of the transport layer impractical on a link-by-link basis, where intermediate nodes have to store and maintain a state, at a great cost to a sensor node with little memory for storage and power for processing. When a large number of flows request header compression at the transport layer, maintaining these states can become costly. The solution to avoid large cost has been to restrict header compression to the wireless link: the number of nodes is limited by the available bandwidth and the access point's limitations. However, in some instances, such as cellular networks, the access point at the IP level is already deep into the network, connecting many base stations and tens of thousands of users.

Dissociating the network and transport layer compression gives more versatility: the decompression of the transport layer does not have to happen at the same time as the network layer. This allows to pick the place to maintain the transport layer compression states where it is convenient: for instance, at the access router, at the edge of the access network, at some header compression proxy anywhere on the data path, or at the correspondent node. Of course, nodes that must see the transport layer, such as

firewalls, have to make sure that either they decompress the packets, or that the packets they see have a full transport header. It should be noted here that modern firewalls do maintain per flow state to steer the packets on the fast path, so it is possible to maintain a compression state as well so as to understand the compressed header within the firewall.

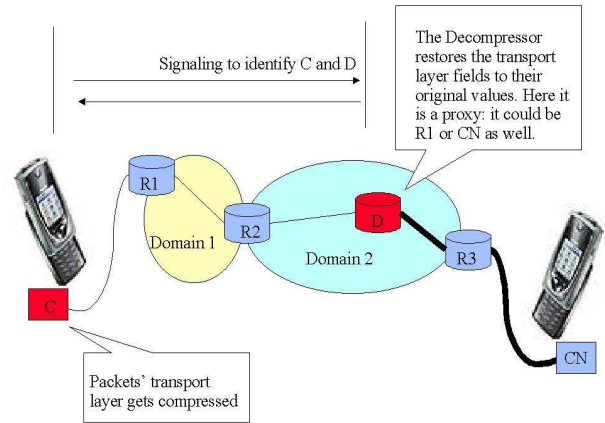


Fig. 2. A transport layer compression architecture

Figure 2 illustrates the architecture for the transport layer compression. The compressor and the decompressor have identified each other by way of some signaling, described later in this document (section III). The decompressor can be several hops away from the compressor. In the example of figure 2, the correspondent node (CN) does not support header compression. On the other hand, some header compression proxy supports header compression.

This architecture allows for the deployment of a header compression proxy, which provides the header compression service to a large number of servers. Indeed, it is reasonable to expect that while some service might gain from header compression, the cost to manage state might be dissuasive, and better left of to a dedicated platform.

The transport layer fields are all the fields in the header not part of the network fields.

3) *Multi-hop Architecture using Orthogonal Layers*: We summarize in table I the differences between the different layers and their roles in header compression.

The layered compression combines both approaches:

- the transport layer is compressed using *timecompression* between two end points, potentially the end points of the connection: in this case, it scales very nicely. No points in the network has two maintain states for a connection it does not participate in. The two end points only compress

Transport	Network
in flow correlation	out flow correlation
time	space
e2e	local
vertical across stack	horizontal across stack

TABLE I
ORTHOGONAL LAYERS OF COMPRESSION

and decompress one or a few streams, no matter how many concurrent flows happen in the network.

- the network layer is compressed using *space* compression at the edge of the network, potentially all the way through a multi-hop access network, or from an IP-addressable sensor node to a gateway.

By taking advantage of the characteristics of each, layered compression offer the benefits of header compression end-to-end. However, compression over several links require to signal to the network that the layered compression approach is being used.

III. HEADER COMPRESSION SIGNALING

In many cases, a compressed header is not recognized by the network, or full headers are required. There are security issues involved in compressing the IP headers. Mostly, intermediate packets have not access to the packet headers immediately anymore. Signaling is thus required to identify the points between which it is possible to use layered compression.

We discuss two cases to show how to take advantage of existing signaling for setting up end-to-end IP header compression. The first case is that of an ad-hoc network connected to the internet through a gateway. In this case, the sensor network uses ad hoc routing protocol, and we consider AODV.

The second case is that of the current internet, where a node is attached to a wired network via a wireless link. The other end point might also be a mobile node attached through a wireless link. We will see how to use the signaling to set up the compression over multiple hops.

A. Ad hoc Signaling for Multi-Hop Header Compression

When IP header compression is at layer 2.5, it is assisted with link layer signaling. However, for a header compression that spans several hops, the link layer information is not available anymore, and some IP signaling has to be added. In this section, we suggest the use of AODV to carry header compression information. Other signaling might be used, of course. The main point is to

introduce the use of layer 3 signaling to extend the reach of header compression over a multi-hop span.

It might seem paradoxical to advocate signaling for header compression: adding new packets and the corresponding extra bandwidth defeats the purpose of header compression. However, this solution is advocated whenever transport header compression is not supported or is too costly locally, and some decompressor has to be found several hops away. In this case, the use of signaling enables the transport compression, and allows the overall bandwidth to be reduced. The AODV signaling is used anyway for route discovery: by leveraging existing signaling, there is no added bandwidth use on the network.

The idea is to add an option to the AODV [9] route request RREQ message. The option would be processed similarly as in the RSVP case above. The end node would send the RREP with an option carrying the information about the existence of a decompressor on the path.

The signaling is as follow:

- the compressor sends a RREQ towards the destination with a transport header compression option.
- Each candidate decompressor receives the packet, and once one is willing to ensure the decompression of the packets, it writes itself into the RREQ as a compression option.
- If for some reason, another potential decompressor feels it is more able to perform the decompression, it replaces the previous decompressor in the compression option of the RREQ.
- if a node must see full header packets- for instance a firewall that is unable to maintain a transport compression state for this flow-, it inserts a flag so that no further downlink decompressor replaces a decompressor uplink from this node.
- another node can re-instate transport compression if it has been interrupted by setting itself as a new compressor, and changing the corresponding fields
- Once the packet is received by the destination, a RREP message is sent back to confirm to the ultimate decompressor that it has been elected, and to confirm to the compressor that a decompressor has been found.
- no state is maintained once the compressor and decompressor state engines have been identified

If the adhoc network is connected to the internet through a gateway, the gateway can flood the network with a broadcast message (it might do so periodically anyway) carrying an option informing the nodes participating in the network whether or not it supports the transport layer compression. In this case, no extra signaling is necessary at the start of the connection for each connection leaving

the ad hoc network.

B. Only the end link(s) is (are) wireless

For a sensor network which does not require ad hoc routing protocol, or for others network, other signalling can be used as well. We present here the use of RSVP [8], but SIP or other signaling can be used as well.

One application of the RSVP signaling would be to perform end-to-end or end-to-proxy header compression. For instance, the access network of a streaming audio server would gain from the header compression on his side by significantly reducing the transport bandwidth. Another example is VoIP, where both entities communicating are mobile. Instead of having the headers compressed/decompressed on one side, then compressed/decompressed on the other side, this architecture allows for the transport header to be compressed across the whole end-to-end route. It is more elegant and efficient, as it reduces the overall load on the network in between.

Note that since the uplink and the downlink routes may differ over several links, it is not possible to use in-band signaling to provide the state compression feedback.

The steps are the same as for the AODV signaling, with the PATH message replacing the RREQ, and the RESV message replacing the RREP. The reverse path should be set up in the same manner.

IV. PERFORMANCE EVALUATION

The benefits of header compression have been assessed in many papers for single link compression. The benefit of our header compression architecture is to extend compression gains over multiple links. The gain is thus the sum of the header compression gain for each link. In order to illustrate this, we consider a basic case and conduct some simple analysis to assess the power saved by multi-hop compression.

Take a sensor network, for which the nodes are connected to the sink via a binary tree. The sink has N leaves. Each leaf has two leaves, which have two more leaves, etc. The depth of the tree is 4. Without loss of generality, we only consider one of the N branches connected to the sink. We consider that all nodes have the same packet rate, and look only at the uplink, as the downlink is symmetric in terms of power usage.

We consider a Bluetooth air interface with a data rate of 720kbp/s as the link layer. 802.15.4 air interface would give similar results. However, we assume the sensor nodes run IPv4 on top of this. We consider the case of no compression, of an iterated link by link compression, and of

our multi-link compression. We consider a payload of size x bytes. We assume there are no error on the wireless link, as this assumption favors the uncompressed case: the more data is transmitted over the air, the more likely a re-transmission is required.

We consider that traditional *single link* compression maps an *uncompressed* $h_{uc} = 44$ bytes IP/UDP into a $h_{sl} = 3$ bytes packet [7], and that the *multi-hop* compression we introduced reduces the 44 bytes IP/UDP packet into $h_{mh} = 4$ bytes. This last figure comes from the fact that we assume that compressing the network and transport layers separately adds a byte eventually, as we described in Section II-B.

From [7] we take the value of $\tau = 25\mu s$ of CPU time for compression. We assume the same time for decompression. This underestimate the actual time for a sensor, as the processor used in [7] is much faster than that of any sensor mote. The transmission time is $(h_i + x)/R$ where $i = uc, sl, mh$ and R is the bit rate. Defining p_c and p_{tr} to be the power consumption rate for computation and for transmission/reception¹ respectively, we can compare the power consumption for the different scenarios.

For the uncompressed scenario, there are $(8 \times 4 + 4 \times 3 + 2 \times 2 + 1) = 49$ transmissions of $h_{uc} + x$, and 48 receptions, for a power usage proportional to $97(h_{uc} + x)/Rp_{tr}$.

For the single link compression, we have the same number of transmission and receptions, but the packet size is $h_{sl} + x$. Also, each transmission is associated with a compression, and each reception with a decompression. The power usage is thus proportional to $97[(h_{sl} + x)/Rp_{tr} + 2 * \tau p_c]$.

For the multi-hop compression, we still have the same number of transmission and receptions, but the compression is performed only once per packet. Thus the power usage is proportional to $97(h_{mh} + x)/Rp_{tr} + 15\tau p_c$.

We take the values for $p_c = 15mA$ and $p_{tr} = 24mA$ from the description of an Intel Imote [10] using Bluetooth. Figure 3 shows the normalized gain for different values of x ranging from 1 to 30 bytes. The power consumption is normalized so that it is 1 for all packet sizes for the uncompressed scenario. One can see for instance that header compression reduces the power consumption by a factor 4 for the iterated link compression, and 5 for the multi-hop compression for $x = 6$ bytes. This means that the lifetime of a battery operated sensor network would be multiplied by 4 in the iterated link compression case, and 5 in the multi-hop compression case.

The benefit from the multi-hop header compression de-

¹Reception power is almost equal to transmission power for typical sensor motes

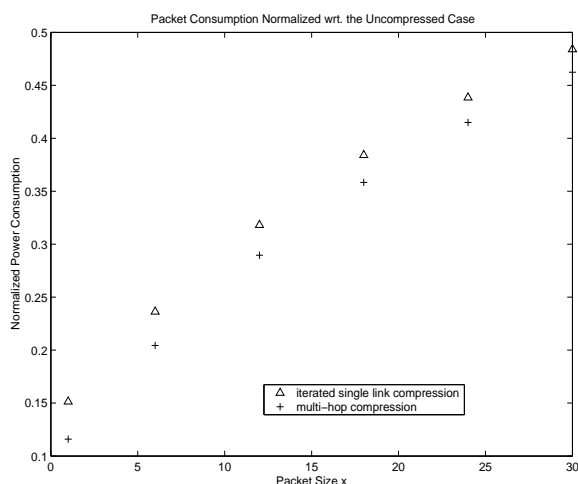


Fig. 3. Gain from the Header Compression

creases as x increases: the transmission power becomes preponderant over the reduced amount of computation, and the benefit of header compression itself becomes smaller. Also, we assumed a constant cost of computing power for compression. However, since this cost is dominated by a packet memory copy, it should be greater for the iterated single-hop compression, and our results under-estimate the gain of multi-hop compression. We are working on assessing this cost more accurately.

This basic analysis does not take into account the other benefits of multi-hop compression: for instance, a node is not dependent on a single root maintaining a compression state with multi-hop compression. As such, routing is flexible, and it is easy to recover from a node becoming instable by routing to another node.

We are in the process of implementing the compression scheme in an NS-2 environment [11]. The cost of establishing the compression for the connection will be assessed as well as the benefit gained from the compression using this tool for different classes of network topologies.

V. CONCLUSION

In this document, we introduced a header compression architecture which dissociates the network from the transport compression, allowing compression to happen on different links, domains, networks. The main purpose is to offer the performance of a statefull compression while delegating the maintenance of such a compression state to either a dedicated header compression proxy, or to a convenient decompressor, be it the access router or the correspondent node.

To ensure end-to-end IP header compression, we suggested the use of an extended RSVP signaling to identify the end points of the compression or the use of an option onto AODV.

This architecture allows network fields to be compressed without any delay, and transport field to be compressed and decompressed where it is convenient, improving on the performance of traditional header compression. It applies in ad hoc network environments using IP, where the cost of transmission and the cost of processing compression/decompression is higher than in a traditional network.

REFERENCES

- [1] C. Bormann, editor, *Robust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed*, IETF, ROHC working group, RFC 3095, July 2001.
- [2] V. Jacobson, R. Braden, D. Borman, *Compressing TCP/IP headers for low-speed serial links* IETF, Network working group, RFC 1144, Feb. 1990
- [3] C. Westphal, *Improvements on IP header compression*, in Proc. of IEEE Globecom Conference, San Francisco, December 2003.
- [4] C. Westphal, R. Koodli, *Stateless Header Compression* in Proc. of the IEEE ICC 2005, Seoul, Korea.
- [5] L-E. Jonsson, *Requirements on ROHC TCP/IP Header Compression* draft-ietf-rohc-tcp-requirements-08.txt, IETF, work in progress, Sept. 2004.
- [6] I. Solis, K. Obraczka, *The Case for a Flexible Header Protocol in Power Constrained Network* in Proc. of WCNC 2003, New Orleans, LA.
- [7] J. Lilley, J. Yang, H. Balakrishnan, S. Seshan, *A Unified Header Compression Framework for Low-Bandwidth Links* in Proc. of 6th International Conference on Mobile Computing and Networking, Mobicom 2000, Boston, MA.
- [8] R. Braden et al., *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*. RFC 2205, IETF, September 1997.
- [9] C. Perkins, E. Belding-Royer, S. Das, *Ad hoc On-Demand Distance Vector (AODV) Routing*. RFC 3561, July 2003.
- [10] L. Nachman, *Imote 2* TinyOS technology exchange, UC Berkeley, Feb. 2005, <http://www.tinyos.net/ttx-02-2005/platforms/ttx05-imote2.ppt>.
- [11] *The Network Simulator - ns-2* <http://www.isi.edu/nsnam/ns/>