Implementation and Evaluation of an Improved Header Compression for 6LoWPAN

Sreejesh V. K.
Department of Computer Science
Cochin University of Science and Technology
Kochi, Kerala 682 022
Email: vksreejesh.nair@gmail.com

G. Santhosh Kumar
Department of Computer Science
Cochin University of Science and Technology
Kochi, Kerala 682 022
Email: san@cusat.ac.in

***Abstract*—Extending IPv6 to IEEE 802.15.4-based Low power Wireless Personal Area Networks requires efficient header compression mechanisms to adapt to their limited bandwidth, memory and energy constraints. This paper presents an experimental evaluation of an improved header compression scheme which provides better compression of IPv6 multicast addresses and UDP port numbers compared to existing mechanisms. This scheme outperforms the existing compression mechanism in terms of data throughput of the network and energy consumption of nodes. It enhances throughput by up to 8% and reduces transmission energy of nodes by about 5%.**

Keywords-6LoWPAN; Header Compression; IPv6; Multicast Address; UDP

I. INTRODUCTION

IPv6 over Low power Wireless Personal Area Networks (6LoWPAN)[1] defines a protocol for transmission of IPv6 packets over 802.15.4 radio links. The 802.15.4 standard[2] specifies protocols for interconnection of low-power devices like wireless sensors to form low-power WPANs. Such networks are characterized by low throughput (250 kbps at 2.4 GHz), short link layer frames (127 bytes long), nodes with limited processing power and memory, short communication range (10 to 100 meters), and communication over multiple hops.

Supporting IPv6 to these heavily constrained devices and their networks pose several challenges. The minimum MTU (Maximum Transmission Unit) requirement in IPv6 is 1280 bytes whereas an 802.15.4 link layer frame is only 127 bytes long. Hence a link-layer fragmentation and reassembly mechanism is necessary for efficient transportation of IPv6 datagrams over LoWPAN links. In each frame as shown in Fig. 1 the header and FCS (Frame Check Sequence) together consume 25 bytes and the optional link-layer security header is 21 bytes long. This leaves only 81 bytes to carry link-layer data[1]. When a UDP packet is carried over such a frame, it can contain only 33 bytes of upper layer data, since the IPv6 (40 bytes) and UDP (8 bytes) headers together consume another 48 bytes. This means the transmission efficiency for upper layer data is as low as 26%. Header compression should be employed to improve this value by reducing the header overhead and providing more space to carry upper layer data.

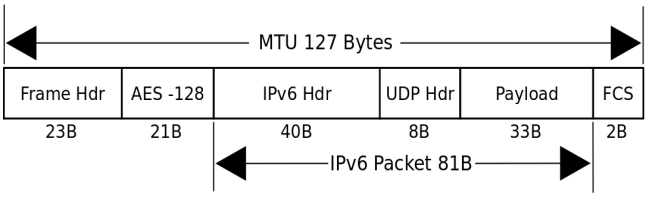

Fig. 1. IEEE 802.15.4 Frame

To address these challenges and to standardize the use of IPv6 over 802.15.4 radios, the Internet Engineering Task Force (IETF) chartered the 6LoWPAN Working Group[3] in 2005. The Working Group proposed the 6LoWPAN protocol stack architecture (Fig. 2) in which an adaptation layer is introduced between the network and link layers of the IP stack to reduce IP overhead. The adaptation layer performs fragmentation and reassembly, header compression and decompression, and support for layer-two forwarding to deliver an IPv6 datagram over multiple radio hops[4].

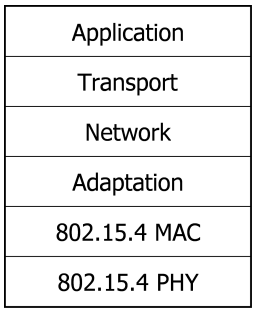

Fig. 2. 6LoWPAN Protocol Stack

The fragmentation mechanism encodes a datagram using multiple link frames. It does not include end-to-end recovery of lost fragments, assuming that link-layer acknowledgments will provide sufficient delivery success rates[5]. The fragmentation header is 4 or 5 bytes long and contains *datagram size* (indicates size of the original datagram), *datagram tag* (identifies all fragments of a particular datagram) and *datagram offset* (indicates the location of the fragment within the datagram).

Traditional flow-based header compression techniques do

not perform well with LoWPANs because in many LoWPAN applications, traffic is driven by infrequent readings or notifications rather than long lived flows. Therefore, the WG developed a new stateless compression format that compresses the headers by removing redundant information across the link, network and transport layers and by using compact forms for commonly used header field values. The header compression format also allows stateful compression of arbitrary IPv6 address prefixes. To accomplish this, each node maintains a context table containing IPv6 prefixes and their corresponding context values[5].

This paper presents the performance evaluation of an improved header compression mechanism proposed in [6] to compress IP and UDP headers in LoWPANs. The improved scheme is compared with the currently implemented header compression scheme in Contiki in terms of two metrics, namely, the throughput and transmission energy.

This paper is organized as follows. Section 2 presents an overview of related works. The improved header compression scheme is described in Section 3. Section 4 explains the experimental setup and evaluation procedure. The results are discussed in section 5, and we draw our conclusions in section 6.

II. Related Works

A. LOWPAN_HC1 and LOWPAN_HC2

The first header compression mechanisms, LOWPAN_HC1 and LOWPAN_HC2 were proposed in RFC 4944[4]. LOWPAN_HC1 defines a compression format for IP header. It assumes default values for IP version (v6), Traffic class and Flow label (both are zero). The payload length can be inferred from either the link layer header or the fragmentation header. The source and destination IPv6 addresses are link local; and their interface identifiers can be inferred from the link layer addresses. Next Header field is compressed to 2 bits and indicate UDP, ICMP or TCP. Hop Limit is not compressed and the full hop limit value is carried in line.

LOWPAN_HC2 defines a compression format for UDP header. It allows compression of source port, destination port and length fields. UDP length can be inferred from the IPv6 header. The commonly used port numbers in the range F0B0 - F0BF can be compressed down to 4 bits. UDP checksum is not compressed and is therefore carried in full.

The above scheme can compress the IPv6/UDP header down to 7 octets in the best case. It is very effective for link-local unicast communication. However, since it does not compress global and multicast addresses, it is insufficient for most practical uses of IPv6 in LoWPANs. With link-local multicast and global unicast communication, the compressed header sizes are 23 and 31 bytes respectively[7]. Furthermore, HC1/HC2 requires that the compressed headers must be contiguous. This constraint prevents the compression of a UDP header when an IPv6 extension header is present.

B. LOWPAN_HC1g

The LOWPAN_HC1g[8] is an extension of LOWPAN_HC1 with the capability to compress global unicast addresses. This is accomplished by assuming that a PAN is assigned a single compressible 64-bit global IP prefix. The prefix can be elided when either the source or destination address matches the compressible global IP prefix. With this scheme, the global address can be compressed down to 64 bits or 16 bits or can be elided completely.

Since HC1g can compress only a single shared global IP prefix, it is not useful when devices communicate with external LoWPAN or the Internet. In this case, the full 128-bit address must be carried in line. Again, the compression of multicast addresses is not supported in HC1g.

C. LOWPAN_IPHC and LOWPAN_NHC

The latest standards for compressing IPv6 and UDP headers, LOWPAN_IPHC and LOWPAN_NHC, are specified in RFC 6282[9]. LOWPAN_IPHC can efficiently compress multicast and global addresses in addition to link-local and unicast addresses. It also compresses the Hop Limit field by defining compact forms of commonly used IPv6 hop limit values. Moreover, compression of IPv6 extension headers is supported and a chain of arbitrary next headers can be encoded efficiently.

LOWPAN_IPHC employs stateless compression to compress link-local IPv6 addresses. Global IPv6 address compression uses shared contexts for arbitrary prefixes. The encoding contains an additional Context Identifier Extension (CID) byte when communicating with global address. The leftmost 4 bits of CID specify the context used for compressing the source address and the rightmost 4 bits specify the destination context. Context based compression allows us to compress up to 16 network prefixes and save 60 bits of payload when communicating with external networks[10]. The LOWPAN_IPHC base encoding is 2 bytes long and is shown in Fig. 3.

011	TF	NH	HLIM	CID	SAC	SAM	M	DAC	DAM
3	2	1	2	1	1	2	1	1	2

Fig. 3. LOWPAN_IPHC Base Encoding

The first three bits in the encoding format indicate the use of IPHC compression. The 2-bit HLIM field compresses commonly used hop limit values. TF field indicates whether the Traffic class and/or Flow label are compressed. NH=1 indicates that the next header is compressed using LOWPAN_NHC. If CID=1, a Context Identifier Extension byte immediately follows the DAM field. The mode of compression of source/destination address is given by SAC/DAC fields (1 indicates context based compression and 0 indicates stateless compression). The destination address is multicast if M=1. The SAM/DAM fields indicate the number of bits of the source/destination address carried in line. LOWPAN_IPHC can compress source address and unicast destination address

down to 64, 16 or 0 bits and multicast destination address down to 48, 32 or 8 bits in various scenarios.

LOWPAN_NHC uses variable length identifier to specify the next header and it can compress any arbitrary next header. The standard [9] describes UDP and IPv6 extension header encodings only. When compressing UDP headers, it is possible to elide the checksum, provided that additional upper-layer security mechanisms like Message Integrity Check (MIC) are used. Besides compressing port numbers based on F0B0 down to 4 bits as in [4], LOWPAN_NHC compresses port numbers in the range F000 – F0FF down to 8-bits. The format for UDP header encoding is shown in Fig. 4.

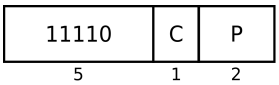

Fig. 4. LOWPAN_NHC Encoding for UDP Header

The first five bits identify this as a UDP header. If C=1, the checksum is compressed. The last two bits in the encoding format indicate the number of bits of the source and destination ports carried in line. The options are:
00 - All 16 bits of both ports carried in line
01 - All 16 bits of source port and 8 bits of destination port
10 - 8 bits of source port and all 16 bits of destination port
11 - 4 bits of both ports carried in line

LOWPAN_IPHC can compress the IPv6 header down to 2 octets with link-local communication. When routing over multiple hops, the header can be compressed down to 7 octets. With IPHC and NHC, the compressed header lengths are 6, 7 and 10 bytes respectively with link local unicast, link local multicast and global unicast communications[7].

III. Improved Compression Scheme

In [6], a compression scheme is proposed which extends the capability of LOWPAN_IPHC and LOWPAN_NHC by providing better compression of IPv6 multicast addresses and UDP port numbers. It utilizes the unused bit combinations in LOWPAN_IPHC so that more compression options are available for multicast addresses. It extends LOWPAN_NHC by making the source port and destination port compressions independent of each other. It also proposes a scheme to compress ICMP headers and routing headers in 6LoWPAN networks. The LOWPAN_IPHC-IMP encoding is shown in Fig. 5.

Fig. 5. LOWPAN_IPHC-IMP Base Encoding

The 2-bit CID field indicates whether the Source Context Identifier and/or Destination Context Identifier are present. The compression mode and the number of bits of source and destination addresses carried in line are given by the 3-bit Source Address/Destination Address fields. The remaining fields are interpreted in the same way as they are in IPHC encoding.

The UDP header compression in NHC-IMP is improved by separating the source port and destination port encodings so that they can be compressed independently of each other. As shown in Fig. 6 the length of UDP identifier is reduced to 3 bits and the source port and destination port numbers are encoded using 2 bits each. The port compression options in NHC-IMP are:
00 - All 16 bits of the port carried in line
01 - First 8 bits of the port is 0xF0 and elided. The remaining 8 bits are carried in line
10 - First 12 bits of the port is 0xF0B and elided. The remaining 4 bits are carried in line
01 - First 8 bits of the port is 0x00 and elided. The remaining 8 bits are carried in line

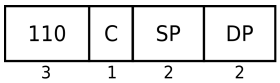

Fig. 6. LOWPAN_NHC-IMP Encoding for UDP Header

When both port numbers are in the range F000-F0FF, NHC-IMP will compress both of them to 8 bits each whereas NHC will compress only one of them. Well known port numbers in the range 0-255 are compressed to 8 bits by NHC-IMP but in NHC the full 16 bits of these port numbers must be carried in line.

IPHC-IMP can compress multicast addresses down to 48, 32, 8 and 0 bits in different scenarios. The all-nodes and all-routers multicast addresses can be completely elided in IPHC-IMP but in IPHC, the lowermost 8 bits of these addresses must be carried in line. Furthermore, the solicited node multicast address can be compressed to 24 bits in IPHC-IMP compared to 48 bits in IPHC.

IV. Experimental Evaluation

An experimental evaluation of the two protocols, namely LOWPAN_IPHC/NHC and LOWPAN_IPHC/NHC-IMP is presented in this section. The following subsections describe the hardware and software platforms used.

A. Hardware Platform

The experiments are performed using Crossbow's TelosB motes[11]. TelosB uses Texas Instruments MSP 430 micro-controller featuring a 16-bit, 8 MHz processor with 10 KB of RAM and 48 KB program flash memory. It is equipped with CC2420 radio[12] which operates in 2.4 GHz ISM (Industrial, Scientific and Medical) band and offers a data rate of 250 Kbps. Other features include USB programming capability and an optional sensor suite with integrated light, temperature and humidity sensors.

B. Software Setup

The operating system used is Contiki[13] version 2.5. It is an open source operating system for networked embedded devices and wireless sensor networks. It includes uIPv6, a lightweight IPv6 stack designed for constrained devices. Contiki supports 6LoWPAN header compression, IETF RPL IPv6 routing[14], and the IETF CoAP application layer protocol[15] among many other protocols and mechanisms. Moreover, standard Operating System features like threads, timers, random number generator, clocks, file system and command line shell are also provided in Contiki[16].

The 6LoWPAN implementation in Contiki 2.5 conforms to [4] and [18]. It implements 6LoWPAN header compression, fragmentation and layer three forwarding. The features of 6LoWPAN implementation[17] in Contiki 2.5 are:

- Supports HC1 and IPHC header compressions
- Currently supports 802.15.4 64-bit addresses only
- Next header is compressed only if it is UDP
- UDP checksum compression is not implemented
- Compression of IPv6 Extension Headers not supported

The existing 6LoWPAN implementation in Contiki 2.5 is extended by incorporating the new compression technique so that it now supports IPHC as well as IPHC-IMP. An application can choose the desired compression scheme by appropriately setting the compilation option in the project Makefile.

C. Experimental Setup

The experimental setup (Fig. 7) consisted of two TelosB motes, a source which sends UDP packets and a sink which receive them. The UDP payload length is varied from 4 bytes to 144 bytes. For each payload value, 10 measurements have been taken and the final result is the mean value of these measurements.

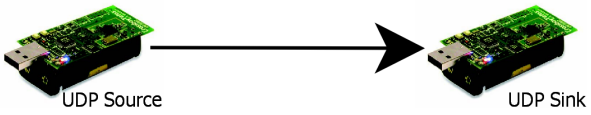

Fig. 7. Experimental Setup

D. Energy Consumption

The software based power profiler in Contiki shows the time spent by a mote in transmit and listen states. Using these and the standard values provided in the TelosB and CC2420 data sheets, the energy consumed by a mote can be determined. For example, the energy $E_{Transmit}$ consumed by a mote due to transmitting is given by

$$E_{Transmit} = T_{Transmit} \times Voltage \times I_{Transmit}$$

where $T_{Transmit}$ is the time spent by the mote in transmitting state and $I_{Transmit}$ is the transmit current. According to the data sheets, the mote operating voltage is 3V when attached to the USB port of a host computer and the transmit current is 17.4 mA.

V. Results and Discussions

The performance of IPHC and the proposed IPHC-IMP are compared in terms of data throughput and transmit energy per bit of the source.

Fig. 8 shows the average throughput obtained for the two compression schemes. IPHC-IMP outperforms IPHC because the number of bytes of MAC payload available to carry data is greater in IPHC-IMP compared to IPHC. When data payload length is 94 bytes, IPHC suffers from fragmentation and there is a decline in throughput. But IPHC-IMP is able to send the data unfragmented because of a lower header overhead. The improvement in throughput with IPHC-IMP at this point is 77.6%. In both schemes, throughput is found to decrease once fragmentation occurs. The average improvement in throughput with IPHC-IMP is 7.92%.

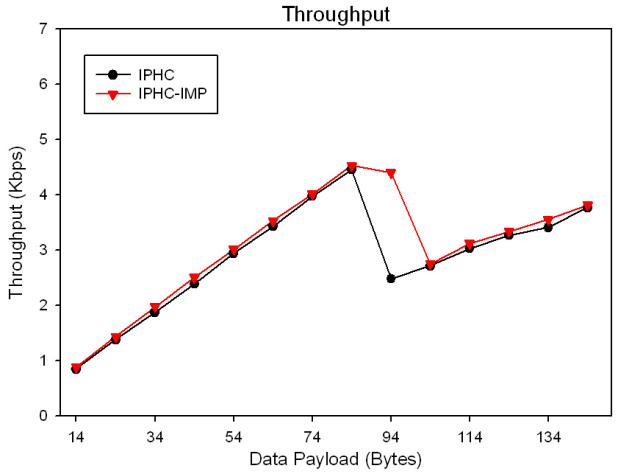

Fig. 8. Throughput

The average transmit energy per bit for the two schemes is shown in Fig. 9. We can notice that IPHC-IMP reduces the energy consumption of the source node. This is because IPHC-IMP results in the transmission of fewer number of bits compared to IPHC for any given data payload length. A sharp increase in energy consumption is noticed once the payload length exceeds the fragmentation boundary. Transmit energy is reduced by 4.84% on average with IPHC-IMP.

In light of the above discussions, it is confirmed that IPHC-IMP is a better scheme for 6LoWPAN header compression and offers higher throughput and consumes lesser energy compared to IPHC.

VI. Conclusion

This paper has presented an experimental evaluation of an improved header compression mechanism for 6LoWPAN networks using Contiki OS and TelosB motes. The performance of the proposed scheme is compared with the existing compression mechanism. Experimental results show that the proposed scheme increases throughput and reduces the energy consumption of nodes in the network.

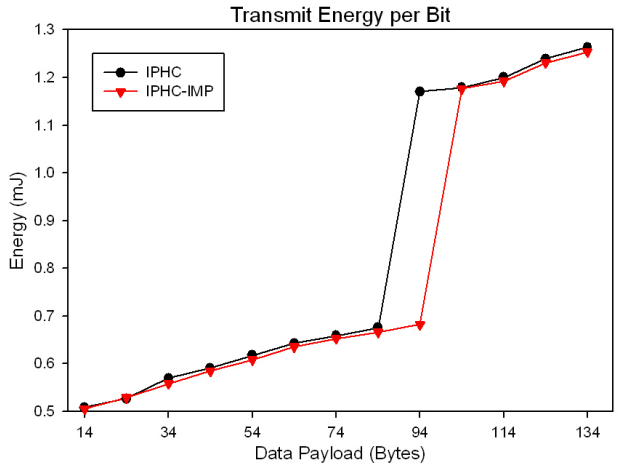

Fig. 9. Transmit Energy per Bit

A majority of the energy consumed by a sensor node is in communication while CPU consumes only a minor portion of the power consumption[19]. In this point of view, the improved compression scheme saves node energy and extends the average lifetime of the entire network.

REFERENCES

[1] N. Kushalnagar, G. Montenegro, and C. Schumacher, *"IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals"*, RFC 4919, IETF, Aug. 2007.

[2] IEEE Standards 802.15.4, *"Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)"*, IEEE Computer Society, Oct. 2003.

[3] IPv6 over Low power WPAN, http://datatracker.ietf.org/wg/6lowpan/charter/

[4] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, *"Transmission of IPv6 Packets over IEEE 802.15.4 Networks"*, RFC 4944, IETF, Sept. 2007.

[5] J. Ko, A. Terzis, S. Dawson-Haggerty, D. E. Culler, J. W. Hui, and P. Levis, *"Connecting Low-Power and Lossy Networks to the Internet"*, IEEE Communications Magazine, pp. 96-101, Apr. 2011.

[6] W. Huiqin, and D. Yongqiang, *"An Improved Header Compression Scheme for 6LoWPAN Networks"*, Ninth International Conference on Grid and Cloud Computing, Nov. 2010.

[7] J. Hui, D. Culler, S. Chakrabarti, *"6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture"*, IPSO Alliance, Jan. 2009.

[8] J. Hui, and D. Culler, *"Stateless IPv6 Header Compression for Globally Routable Packets in 6LoWPAN Subnetworks"*, Expired Internet-Draft, Jun. 2007. http://tools.ietf.org/html/draft-hui-6lowpan-hc1g-00

[9] J. Hui , and P. Thubert, *"Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks"*, RFC 6282, IETF, Sept. 2011.

[10] A. Ludovici, A. Calveras, M. Catalan, C. Gomez, and J. Paradells, *"Implementation and Evaluation of the Enhanced Header Compression(IPHC) for 6LoWPAN"*, The Internet of the Future, Lecture Notes in Computer Science, Springer Berlin Heidelberg, Volume 5733, pp. 168-177, 2009.

[11] TelosB Data Sheet, http://www.willow.co.uk/TelosB_Datasheet.pdf

[12] Chipcon CC2420 Data Sheet, http://www.ti.com/lit/ds/symlink/cc2420.pdf

[13] Contiki - Connecting the Next Billion Devices, www.sics.se/contiki/

[14] P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander, *"RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks"*, IETF, Mar. 2012, http://tools.ietf.org/html/rfc6550

[15] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, *"Constrained Application Protocol (CoAP)"*, Internet Draft (Work in Progress), Mar. 2012. http://tools.ietf.org/html/draft-ietf-core-coap-09

[16] The Contiki OS, http://www.contiki-os.org/p/about-contiki.html

[17] The Contiki Operating System Documentation, http://www.sics.se/ adam/contiki/docs/

[18] J. Hui, P. Thubert, *"Compression Format for IPv6 Datagrams in 6LoWPAN Networks"*, Expired Internet Draft, IETF, Oct. 2009. http://tools.ietf.org/html/draft-ietf-6lowpan-hc-06

[19] D. Estrin, *"Sensor Network Protocols"*, Eighth International Conference on Mobile Computing and Networking, Sep. 2002.