IPv6 Compression Techniques

and Performance

A Thesis

Submitted to the Faculty

of

Drexel University

by

Ryan Measel

in partial fulfillment of the

requirements for the degree

of

Master of Science in Electrical Engineering

February, 2011

Acknowledgements

I'd like to thank Dr. Moshe Kam, my advisor, for his guidance and wisdom, Dr. William Regli, for getting me started, Joseph Macker, for the oppurntunity to learn at the NRL, former and current labmates at the Data Fusion Lab and ACIN, my family, for their love and support, and Our Lord, for the many blessings He has bestowed.

Dedications

For my parents, Susan and Ernest.

Table of Contents

List of Figures

List of Tables

The heavens declare the glory of God; and the firmament showeth his handiwork.

Day unto day uttereth speech, and night unto night showeth knowledge.

There is no speech nor language, where their voice is not heard.

Their line is gone out through all the earth, and their words to the end of the world.

—Psalms 19:1–4

Abstract

IPv6 Compression Techniques
and Performance

Ryan Measel
Advisor: Moshe Kam, Ph. D.

The adoption of IPv6 over its predecessor IPv4 has been a slow process despite that IPv6 offers many advantages is terms of flexibility, security, and routing in addition to a vastly larger addressing space. The disadvantage is that IPv6 needs a larger header size as well to provide these features. A great amount of overhead accumulates for IPv6 transmissions with small packet sizes, such as network management and real-time traffic. Several header and payload compression techniques have been proposed to mitigate this effect. In this paper, the various compression techniques are identified and evaluated. A testbed was devised to further investigate a select set of protocols. Traffic shaping was used to impose non-ideal network constraints, such as delay, packet loss, and bit error. It was shown that header compression is highly effective at reducing overhead, especially for small packet sizes. RObust Header Compression (ROHC) was found to perform the best under network constraints with the exception of packet reorderings. Payload compressions techniques were less useful as they require overhead in the form of either a per-packet dictionary or a synchronization mechanism, and the data payload cannot be previously compressed or encoded.

1. Introduction

Internet Protocol version 4 (IPv4) was the first networking layer protocol to be adopted for wide use, and has been used by the US military since its inception, over three decades ago. In the 1990s, it became apparent that IPv4 was incapable of serving the growing needs of network operations and the Internet. The development of IPv6 began with the objective of replacing and improving upon IPv4 in areas that include scalability and flexibility. While many concepts were carried over from IPv4, IPv6 includes a number of new capabilities for improved routing, security, quality of service (QoS), and mobility, in addition to a vastly increased address space.

On 9 June 2003, the Department of Defense (DoD) Chief Information Officer (CIO) signed a memorandum outlining the DoD's goal to "transition to [Internet Protocol Version 6] IPv6 for all inter and intra networking across the DoD by FY2008." However, there was limited progress towards achieving this goal due to various concerns regarding security, performance, product availability, cost, and other deterrents. Nonetheless, IPv6 offers features that are of value, and it remains the only viable replacement for IPv4, the current networking protocol, which will soon run out of address space. As such, efforts to prepare for IPv6 and optimize its usage are still vital.

A primary concern of IPv6 versus IPv4 is the increased header size. IPv4 headers are 20 bytes while IPv6 headers are a minimum of 40 bytes. This becomes problematic especially in tactical radio environments where bandwidth is often very low. For large packets, the difference may be negligible, but for small packets, such as network diagnostics and situational awareness messages, the difference can be significant. To address this concern, various header and payload compression techniques have been proposed by commercial and standards bodies.

The purpose of this work was to assess the available IPv6 compression standards and techniques, analyze their effectiveness, and identify the conditions and applications for which they are best suited. The remainder of the document is organized as follows:

- In Chapter 2, a list is provided of standards and protocols corresponding to IPv6 compression. Commercial offerings and support for these protocols are also discussed.
- Chapter 3 describes the functional operation of the compression techniques.
- Chapter 4 includes an overview of testing for select protocols and the obtained results.

- Finally, conclusions and recommendations concerning the evaluated techniques, including relevant scenarios and applications, are presented in Chapter 5.

2. Identification of IPv6 Compression Techniques

2.1 Standards, Drafts and RFCs

A list of standards and Request For Comments (RFCs) that pertain to IPv6 compression may be found below along with a short description.

- RFC 2393 [55] - IP Payload Compression Protocol (IPComp).

  IPComp provides a framework for implementing payload compression in IP datagrams. It is intended for communication over slow or congested links. Any compression scheme can be used as long as it is lossless which implies no information is lost. The compression is performed in a stateless manner such that packet decompression is independent of other packets.

- RFC 2460 [29] - Internet Protocol, Version 6 (IPv6).

  RFC 2460 is the base specification for IPv6 which defines some of the terminology and the header format.

- RFC 2507 [30] - IP Header Compression.

  This document specifies a header compression technique developed for point-to-point communication. The method can be used on IPv6, UDP, TCP, and encapsulated IPv6. It was designed to be operable on links with high packet-loss rates.

- RFC 2508 [27] - Compressing IP/UDP/RTP Headers for Low-Speed Serial Links.

  This document defines a method for Compressing IP/UDP/RTP (CRTP) headers which was designed specifically for real-time traffic using the RTP protocol on low-speed links. It is limited to real-time traffic transported over UDP/RTP.

- RFC 3095 [25] - RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed.

  ROHC is a header compression technique based on the concept of omitting fields from the header. This is done via an agreement between the sender and receiver so that both know which fields are being omitted. ROHC can be applied to many different types of headers, including but not limited to IPv6. Each supported protocol header has an associated profile

which defines what can be omitted. This document in particular contains profiles for RTP, UDP, ESP, and uncompressed.

- RFC 3173 [54] - IP Payload Compression Protocol (IPComp).

  This is an update to the IP Payload Compression Protocol. The protocol is stateless and lossless. It obsoletes RFC 2393.

- RFC 3241 [24] - Robust Header Compression (ROHC) over PPP.

  Configuration parameters have to be agreed upon by each end of the ROHC compression over a PPP link. A series of Network Control Protocols (NCPs) are used by PPP to handle the negotiation of these parameters. This document defines the configuration both IPv4 and IPv6 for negotiating ROHC parameters on a PPP link.

- RFC 3544 [42] - IP Header Compression over PPP.

  Configuration parameters have to be agreed upon by each end of the IPHC compression over a PPP link. A series of Network Control Protocols (NCPs) are used by PPP to handle the negotiation of these parameters. This document defines both IPv4 and IPv6 configurations for negotiating IPHC parameters on a PPP link. It obsoletes RFC 2509.

- RFC 3545 [43] - Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering.

  RFC 3545 defines the Enhanced Compressed RTP (ECRTP) protocol which is an update to RFC 2508, CRTP. CRTP did not perform well over links with high packet loss and delay. Several extensions to CRTP were put in place in order to minimize the effects of a constrained network environment and thereby making it more resilient to packet loss, jitter, and delay. Similar to CRTP, ECRTP is also limited to real-time UDP/RTP traffic on point-to-point links.

- RFC 3759 [35] - RObust Header Compression (ROHC): Terminology and Channel Mapping.

  RFC 3759 is a clarification of RFC 3095. It provides definitions of logical elements in the operation of ROHC, such as instances, channels, feedback, and contexts. It also provides examples of ROHC operation. This document provides the best overview of ROHC.

- RFC 3843 [36] - RObust Header Compression (ROHC): A Compression Profile for IP.

  This document extends RFC 3095 and provides a specification of an IP only profile for ROHC.

- RFC 4362 [37] - RObust Header Compression (ROHC): A Link-Layer Assisted Profile for IP/UDP/RTP.

  This RFC defines another compression profile for IP/UDP/RTP by using functionality of the link layer to improve the efficiency of the compression. It obsoletes RFC 3242.

- RFC 4815 [38] - RObust Header Compression (ROHC): Corrections and Clarifications to RFC 3095.

  RFC 4815 provides clarifications and corrections for several RFCs relating to ROHC, including 3095, 3241, 3843, and 4109, in order to prevent misinterpretations and promote interoperability.

- RFC 4919 [44] - IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statements, and Goals.

  This document is an informational RFC that defines Low-Power Wireless Personal Area Networks (LoWPANs) and the advantages of running IPv6 over such networks.

- RFC 4944 [47] - Transmission of IPv6 Packets over IEEE 802.15.4 Networks.

  RFC 4944 defines and enables link-local IPv6 traffic over LoWPANs. It provides for mesh-routing, IPv6 and UDP header compression, and fragmentation over an IEEE 802.15.4 link-layer.

- RFC 4997 [31] - The RObust Header Compression (ROHC) Framework.

  RFC 4997 is an explanation and simplification of the Robust Header Compression Framework. It is not intended to obsolete RFC 3095. Rather, it is meant to complement it and define the ROHC framework and uncompressed profile separately. It does not modify or update the framework provided by RFC 3095.

- RFC 5095 [22] - Deprecation of Type 0 Routing Headers in IPv6.

  RFC 5095 addresses a security vulnerability of IPv6 by deprecating the use of Type 0 Routing Headers. This updates RFC 2460 and RFC 4294.

- RFC 5225 [49] - RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP, and UDP-Lite.

  RFC 5225 introduces some simplifications to the rules controlling the actions of the compression endpoints. It expands upon the robustness mechanisms to improve the success rate of

decompression. Furthermore, it specifies compression profiles under the ROHC framework for RTP/UDP/IP, RTP/UDP-Lite/IP, UDP/IP, UDP-Lite/IP, IP and ESP/IP.

- RFC 5795 [53] - The RObust Header Compression (ROHC) Framework.

  RFC 5795 obsoletes RFC 4997. It is an explanation and simplification of the Robust Header Compression Framework. It is not intended to obsolete RFC 3095. Rather, it is meant to compliment it and define the ROHC framework and uncompressed profile separately. It does not modify or update the framework provided by RFC 3095.

- draft-ietf-6lowpan-hc-07 [34] - Compression Format for IPv6 Datagrams in 6LoWPAN Networks.

  This document is a draft for an RFC which will update RFC 4944. It discusses updated compression options for transmitting IPv6 datagrams over IEEE 802.15.4 wireless networks. Stateful address compression using pre-shared network prefixes is permitted, but the draft does not specify methods for distributing such prefixes.

- draft-ietf-6lowpan-nd-09 [56] - Neighbor Discovery Optimization for Low-power and Lossy Networks.

  This document is a draft for an RFC which will update RFC 4944. It proposes changes to IPv6 Neighbor Discovery that better suit LoWPANs, and covers distribution and maintenance of network prefixes.

- IEEE 802.15.4 [7] - IEEE Standard for Information technology- Telecommunications and information exchange between systems- Local and metropolitan area networks- Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs),"

  The IEEE 802.15.4 standard specifies the physical and MAC layer for wireless personal area networks (WPAN). 6LoWPAN is based on this standard.

2.2 Commericial Offerings

2.2.1 Ericsson

Ericsson developed a new protocol based off of CRTP called RObust Checksum-based header COmpression (ROCCO) [57]. ROCCO is a robust header compression scheme targeted at real-time

audio and video streams over cellular links. Primarily, this was intended to facilitate the use of Voice-over IP.

While ROCCO [39] was developed as an alternative to CRTP, it is similar to ECRTP in that it employs checksums for header decompression validation. It also has some unique features such as additional feedback packet types and various compression profiles. Optional feedback types include acknowledgement of an updated context, an indicator for gaps in the packet stream, and an update on link loss characteristics. Compression profiles were developed for both voice and video.

Work ceased on this protocol in 2000 and was incorporated into the initial draft for ROHC [2].

2.2.2 Nokia

More recently, Nokia has published research on multi-hop compression [58] for sensor networks. The compression schemes that have been discussed (i.e. ROHC, ECRTP) are link based compression schemes. This is costly for the wireless nodes forwarding traffic since every node has to decompress and recompress each packet before it is forwarded along. A new mechanism was developed to allow for compression of a packet from end to end which improves the multi-hop performance.

2.2.3 Effnet AB

Effnet AB is a Swedish company that develops commercial header compression technologies [5]. They actively research and develop header compression techniques for commercial offerings and offer four products: Effnet IPHC, Effnet CRTP, Effnet ROHC, and Effnet HC-Sim. Effnet IPHC is compliant with RFC 2507 as well as 3GPP. Effnet CRTP is compliant with RFC 2508 and can optionally support ECRTP. Effnet ROHC is compliant with RFC 3095 and RFC 4815 and offers support for 3GPP2, MBMS, and VoIP flow compression. Effnet HC-Sim is a simulator designed specifically for testing IP header compression algorithms. Notably, in 2005, QUALCOMM licensed Effnet IPHC for use in its CDMA2000 and WCDMA chipsets [6].

2.2.4 Cisco

Cisco offers both RTP and TCP header compression for their routers [11]. The compression can be enabled over process-switched paths, fast-switched paths, and Cisco Express Forwarding-switched (CEF-switched) paths. The routers support RFC 1144, RFC 2507, and RFC 2508. RFC 2508 is the basis of CRTP which can be used for IPv6 traffic.

2.2.5 Microsoft

Windows Server supports IP Header Compression (IPHC) for Point-to-Point Protocol (PPP). This feature implements RFC 2507 and RFC 3544, the specifications for IPHC and IPHC over PPP respectively. This compression scheme "is not a configurable option for IPv6" [8].

2.2.6 6LoWPAN Implementations

Several commercial solutions exist for 6LoWPAN implementation. For example, Arch Rock [17] and Sensinode [20] offer wireless sensor hardware platforms along with 6LoWPAN protocol stacks that support the original RFC 4944. Atmel [18] offers similar platforms with software that supports early versions of the draft-ietf-6lowpan-hc update to RFC 4944. Open source operating systems for low-power sensor networks, such as Contiki [10] and TinyOS [15], offer 6LoWPAN support to a wide variety of hardware platforms and are also based on early versions of the drafts that will update RFC 4944.

3. Operation of IPv6 Compression Techniques

3.1 Header Compression

Header compression is the process by which extraneous parts of the protocol headers are compressed in order to reduce the overhead on packet transmissions. Often, this compression implies omission as not all fields in the header are required for successful transmission. It is possible to define a header compression scheme for any protocol in the transport, network, or MAC layers. IPv6 is a network layer protocol and is of particular interest with regards to header compression due to its large header sizes.

Many researchers have implemented header compression in simulation only [45][48]. Others have deployed various types of header compression schemes using commercial implementations of Robust Header Compression (ROHC) [52]. There are also examples of header compression techniques being employed on different standards and network configurations like Mobile WiMAX [59], NEMO [51] and mesh networks [41].

While many header compression schemes have been developed, only a few were chosen to be evaluated. Namely, CRTP, ECRTP, and ROHC were selected based on their maturity, level of commercial integration, and perceived future value.

3.1.1 Compression RTP (CRTP)

CRTP is an extrapolation of the compression framework specified in RFC 2507 [30] which is compatible with both IPv4 and IPv6. The framework differentiates between two classes of network traffic: TCP and non-TCP. CRTP specifies IP/UDP/RTP as a subclass of the non-TCP class. While it is possible to compress just one of the protocols instead of all three, the resulting size of the compressed header is between 2-4 bytes regardless. There is substantially more to gain by compressing the transport and network layer headers together.

The compression is setup across a single link with a compressor at the sender and a decompressor at the receiver. A session context is established for each stream. A session context is characterized by the IP source, IP destination, UDP source port, UDP destination port, and the RTP synchronization source (SSRC) field. Each session context is assigned an identifier number and a sequence number. The limit on the number of contexts is negotiated by the compressor and decompressor. A session

state is established that contains the portions of the headers which will remain constant and delta values for the fields that change at a constant rate. A packet header can then be reconstructed by the decompressor by adding the first order differential (the delta values) to the uncompressed header values.

CRTP defines five packet types:

- FULL_HEADER contains all the necessary information to construct the uncompressed header as well as the context identifier and sequence number. It is used for synchronizing the compressor and decompressor.
- COMPRESSED_UDP is a packet with a compressed IP/UDP header of 6 bytes and possibly additional uncompressed headers. This is used when a field in the RTP header that is normally constant changes or when there is no RTP header. If the RTP SSRC has changed, then this packet will also update the session context.
- COMPRESSED_RTP is a packet with a compressed IP/UDP/RTP header. The minimum size of this header is 2 bytes. It also includes a new delta value for any field that has changed by some value other than the previous specified delta value.
- CONTEXT_STATE is a packet used for feedback sent from the decompressor to the compressor. It is sent when the decompressor receives a compressed packet with a sequence number that has changed by more than 1 which indicates a packet loss has occurred. The context is labeled as corrupted and the decompressor will discard all compressed packets related to that context until a FULL_HEADER is received to resynchronize the sequence number.
- COMPRESSED_NON_TCP is a packet with a compressed IP/UDP header and without differential coding. This packet type can only be used for IPv6 since it has no ID field. It is more resilient to packet loss than the COMPRESSED_UDP since it doesn't use differential coding.

CRTP performs best on links with short delay and no packet reordering. A long delay on the link is problematic since all compressed packets that are received after a packet is loss are discarded until the context can be resynchronized. A resynchronization takes at least one Round Trip Time (RTT) during which a CONTEXT_STATE is sent by the decompressor and a FULL_HEADER is returned by the compressor. This can result in many discarded packets especially on a high bandwidth link. The same will occur for a packet reordering, which can happen if the link is tunneled over an IP

network. The decompressor would receive a packet with a sequence number incremented greater than 1 which will trigger a resynchronization even though no packets have been lost.

It should also be noted that the use of feedback in the scheme implies a duplex link. CRTP can be used over a half duplex link, but the performance would be severely degraded.

3.1.2 Enhanced Compressed RTP (ECRTP)

ECRTP [43] has been designed in order to address some of the issues of CRTP. It is intended to increase the robustness of CRTP against packet loss and packet reordering so that the compression scheme can be applied for links over tunnels and virtual circuits. Two new features were added to achieve this.

First, some packet loss can be tolerated by packaging context updates in COMPRESSED_UDP packets. Differential and absolute RTP values can be included in the packet in order to maintain synchronization. This exploits the fact that CRTP fails not when a packet is lost but when the next packet arrives. If that next packet contains the relevant context update, then the delta values can be added to the previous values multiple times. The context will not be corrupted and no resynchronization will be required.

Second, the compressor will add a checksum of the header when a UDP checksum has not been included. The decompressor can use this checksum to ensure the validity of a context update even after a packet loss. The UDP checksum is normally required to validate the header decompression after a packet loss when applying a context update. This would not be possible when UDP checksums are disabled. The addition of a header checksum applied by the compressor will allow for header validation even in the absence of the UDP checksum.

These additional features do add to the size of the compressed header. Therefore, ECRTP is only more efficient than CRTP on high loss links or those prone to packet reordering.

3.1.3 RObust Header Compression (ROHC)

ROHC was designed specifically for point-to-point, wireless links that suffer from various network constraints including packet loss, high delay, and bit error. In this sense, it can be viewed as a replacement for CRTP which performs poorly in such scenarios.

RFC 3759 "Terminology and Channel Mapping Examples" [35] explains the overall structure of the protocol and the environment it was designed to operate within. Basic environment concepts can be broken down into the terms: network element, network interface, and logical channel. A network

element (commonly called a node) is an entity that engages in communication over a network interface with other network elements. Each network element in a network may contain one or more network interfaces which it may communicate over, each network interface having its own unique IP identity. At this point, it is easiest to visualize a TCP/IP router with multiple ports (Figure 3.1). The router is a network element, and each of its ports is a network interface with a unique IP addresses. When the router communicates over each one of its ports, information is transmitted across one of potentially many logical channels. Each logical channel can be considered a point-to-point logical connection provided by the lower link lever protocol (e.g., Point-to-Point Protocol (PPP), Ethernet, Wi-Fi). These logical channels may actually be shared physical channels such as in Ethernet where a single port of a router may be physically connected to several neighboring routers, or wireless domains where a broadcast medium essentially connects many routers.

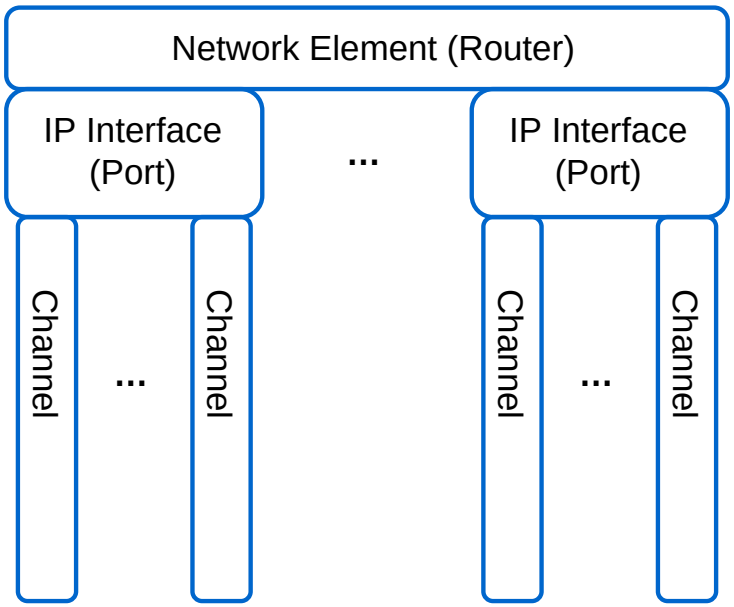

Figure 3.1: A network element with many logical channels per interface (adapted from [35]).

The terms instance, compressor, and decompressor are rather straightforward. An ROHC instance performs either header compression or decompression. The ROHC compressor accepts full IP packets as input and returns compressed packets, while the decompressor accepts compressed packets and outputs the decompressed version. An ROHC channel denotes the logical directed con-

nection from a compressor to a decompressor operating on separate ends of a logical channel. The information flow consists of compressed packets sent from the compressor to the decompressor. The separately defined ROHC feedback channel denotes the information flow in the opposite direction from decompressor to compressor for purposes of feedback to control compression levels and states, etc. This separation is done to promote flexibility for uni- and bi-directional links. If header compression is desired along both directions of a bi-directional logical channel, two compressor-decompressor pairs must be started, one for each direction (Figure 3.2).

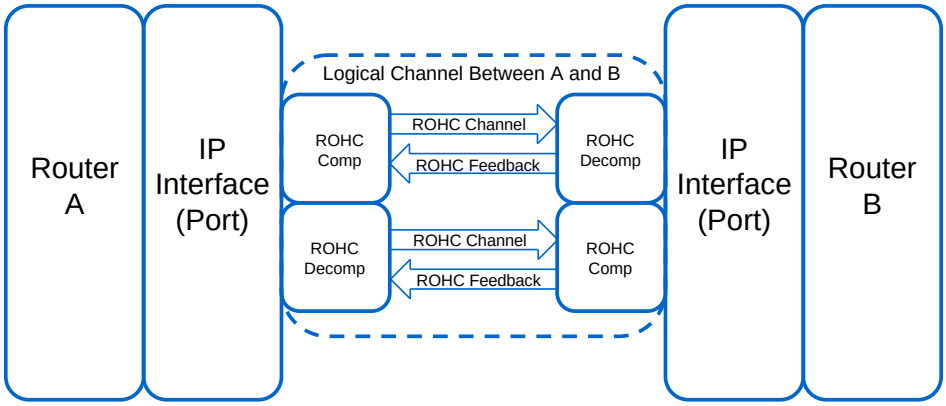

Figure 3.2: An example logical bi-directional channel with compression.

Compression and decompression are performed "underneath" the IP layer, on a per-link basis. In a sense, a compressed header itself is not routable because it must be decompressed before being passed up to the IP layer. However, if several routers with ROHC-enabled links are chained together, it is certainly feasible to route an IP packet along the multi-hop route by successively compressing and decompressing the header. It is also certainly possible and intended, especially in cellular and satellite networks, that only a subset of logical channels (particularly low bandwidth channels) are ROHC-enabled.

A router switches packets based on their IP address which is stored in the IP header of a packet. This is the same portion of the header that IPv6 header compression techniques operate on. For this reason, an unmodified router would be unable to route a compressed header under ROHC. At the initialization of a compressed link, the sender and receiver agree on portions of the header to omit. If a router was not part of this agreement and received a packet with a compressed header, it would

be unable to successfully parse the header and route it. However, switches can successfully forward packets with compressed ROHC headers since they operate at the link layer. A switch routes frames based on the MAC address in the frame header so it does not matter what has been compressed or omitted in the network or transport layer headers.

Furthermore, compression of a multi-hop route must be conducted over each single-hop link individually. The transmitter-receiver pair of routers corresponding to each single-hop link must agree upon a compression format/state before compression takes place over that respective link.

3.2 Packet Payload Compression

Packet compression schemes encode the entire packet payload using any of number of available data compression algorithms. For the purposes of IP packets, it would be necessary to employ a lossless compression scheme to ensure exact recovery of the original packet. The alternative is lossy compression where it is only required to recover an approximation of the original data. This allows lossy compression to achieve higher compression ratios but, to do so, it must assume none of original data is critical. Certainly, this is an assumption that cannot be made for packets since they may contain any type of information. Additionally, if the compressed payload contains headers for upper level protocols, then the exact header data must be recovered.

3.2.1 Dictionary Compression

Most lossless compression algorithms are built using a hash function. The input data is analyzed and mapped into a dictionary. This is done by looking for common sequences which can be assigned a shorter index in the dictionary. The data can then be decompressed by referencing the index of the dictionary. The two main encoding algorithms are Huffman coding and arithmetic coding. The former is faster but has poorer results for data with high entropy. Meanwhile, arithmetic coding produces almost optimal compression ratios for any type of data [13]. Many more encoding algorithms have been developed and optimized for specific types of data. Adaptive coders are also used with compression algorithms which can improve performance as more data is compressed.

High compression ratios can be gained through using compression on the packet payloads, but it is not without limitations [3]. The dictionaries require a substantial amount of memory. Also, the dictionaries must be synchronized for proper decompression at the receiver, and the task of encoding and decoding the data can be computationally intensive depending on the type of data

and the encoding scheme employed. In addition, it becomes useless if a higher layer has already compressed or encrypted the data.

3.2.2 Packet-by-Packet Dictionary Compression

One approach to overcome the problems associated with storing and synchronizing dictionaries is using a packet-by-packet algorithm [1]. A new dictionary is generated for each packet. The dictionary is then included with the compressed data in the packet. By including a new dictionary for each packet, it is not required to store a common dictionary on the sender or receiver and there is nothing to synchronize. Additional overhead would be incurred by including the dictionary with the compressed packet. It is unknown whether this outweighs the overhead incurred by feedback mechanisms used for dictionary synchronization when they are stored locally. Hewlett Packard introduced a form of packet-by-packet compression called HP PPC which can be run on HP routers [1]. It incorporates packet-by-packet compression and run length encoding. Run length encoding checks for sequences of a single character repeated multiple times. It replaces the sequence with a single character and the number of times in which it occurs.

3.2.3 Guess-Table-Based Compression

Another approach to packet compression is a guess-table-based compression algorithm [1]. The algorithm uses preceding bytes to predict what the next byte will be. At the sender, if the guess is correct, then the byte is omitted from the transmission. At the receiver, if the guess is correct, then the byte is reinserted. Both the sender and receiver use the same guess algorithm to ensure lossless compression. The Predictor Compression Algorithm is an example of guess table compression in conjunction with a running dictionary. This algorithm uses less processing and can therefore be run on links with a higher speed since it imposes less processing delay. Due to its reliance on a running dictionary, the algorithm still suffers from memory limitations and the necessity for synchronization. It is also available on some HP routers.

3.2.4 Delayed Dictionary Compression

Delayed dictionary compression [46] is an alternative implementation of the dictionary based compression. It is intended to address the issue of packet drops and packet reorderings. The construction of the dictionary is delayed by some arbitrary amount selected by the compressor. By doing this, compressed packets do not depend on any preceding packets sent within the delay. This

built-in buffer time allows the scheme to be more robust against drops and reordering but it is not completely tolerant.

3.2.5 Packet Compression in Sensor Networks

A packet compression scheme has been proposed for embedded wireless sensor networks as well [40]. The overall goal is saving power for the node. This is accomplished through compressing packets and transmitting less which, in turn, saves power. It is a dictionary approach with synchronization feedback mechanisms. It is debatable how useful such a scheme would be as the downside of a dictionary based compression algorithm is the high use of computation and memory, neither of which a traditional sensor node has.

4. Evaluation of Compression Techniques

To evaluate and analyze the protocols of interest, a wireless testbed was used for performing over-air tests as well as traffic shaping on network flows of varying types. A testbed consisting of two wireless nodes was constructed to conduct experiments with selected compression techniques. The emerging ROHC protocol was focused on since CRTP is already an established protocol and ROHC is designed to surpass it. As stated in RFC 3545 (ECRTP), "ROHC is expected to be the preferred compression mechanism over links where compression efficiency is important" [43].

The overall layout of the wireless testbed is shown in Figure 4.1. The two end nodes are HP Compaq tc4200 Tablet PCs which are running Ubuntu 9.04. The nodes are connected over two network interfaces, wireless IEEE 802.11b and wired IEEE 802.3 Ethernet. The wireless connection is on an ad-hoc network, while the wired connection runs through a dedicated network bridge.

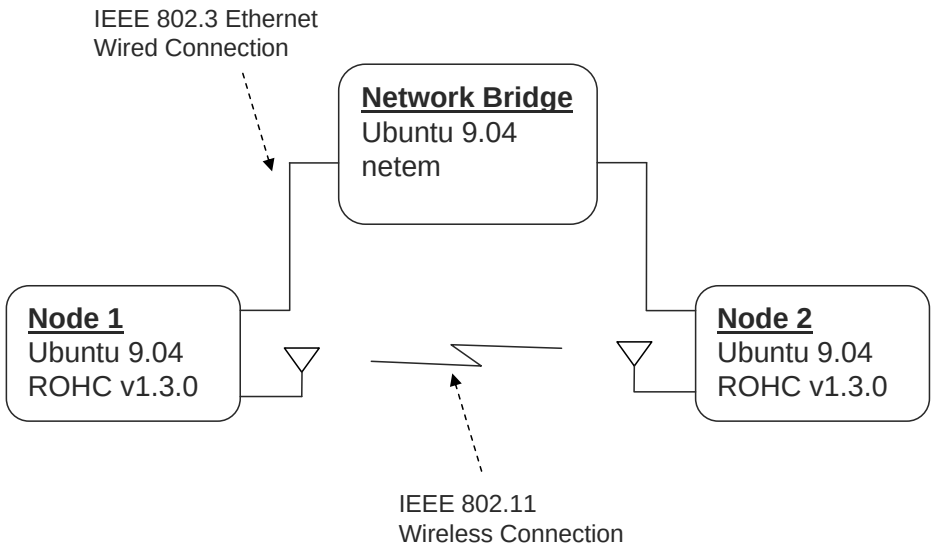

Figure 4.1: A graphical representation of the wireless testbed.

A network bridge is a transparent switch on the data link layer (layer 2 of the OSI model). In this instance, the bridge has two interfaces and forwards packets received on one interface to the other. Not only does this provide a convenient monitoring point for the traffic on the link, but it also

allows an opportunity to perform traffic shaping. Traffic shaping is the process of imparting some characteristic on the network traffic across a link. The bridge also runs Ubuntu 9.04 and netem [14], a tool for traffic shaping. netem is able to emulate network properties such as packet loss, delay, reordering, and duplication. For bandwidth rate control, a token bucket filter is used. The wired connection is used for all tests involving traffic shaping while the wireless link is used for experiment validation and practical testing.

4.1 ROHC Implementation

To test ROHC, a set of libraries developed by the French Space Agency (CNES), Thales Alenia Space (TAS), and Viveris Technologies was utilized. This effort [23] is an update to a deprecated sourceforge implementation [4]. The libraries contain the functions necessary for a full ROHC implementation. The code is open source under the GPL2+ license. All results presented below were achieved using version 1.3.0 of the libraries which was released on March 22, 2010.

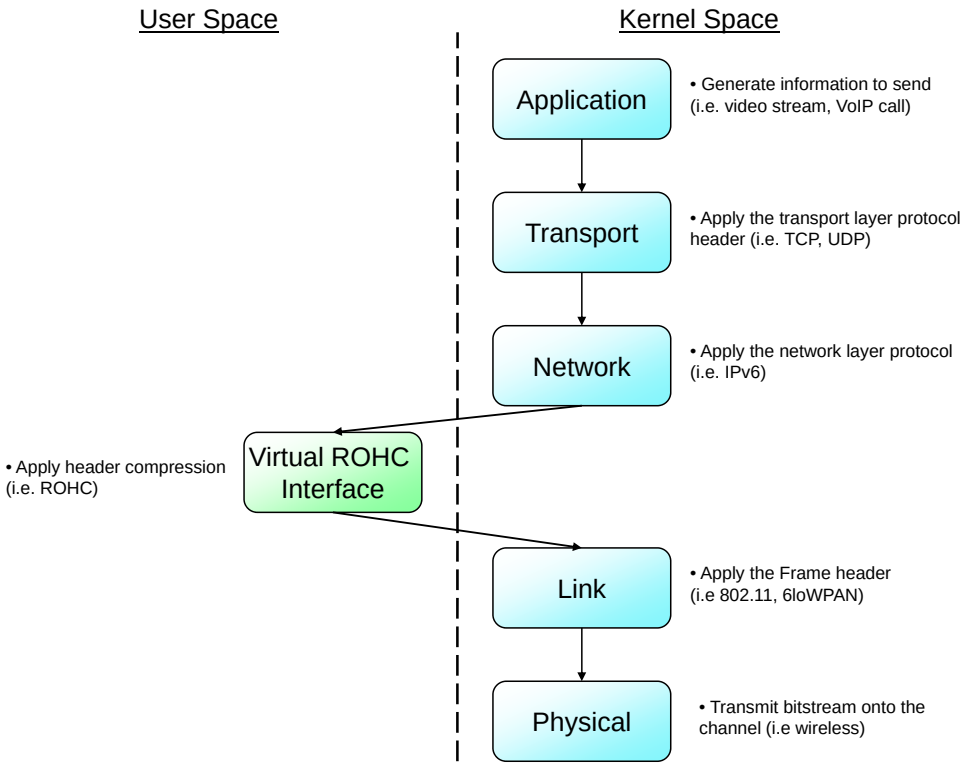

Figure 4.2: ROHC implementation.

The implementation is shown in Figure 4.2 in reference to the OSI model. It is a user space library which provides an API for accessing packets that have been queued by the kernel packet filter. Packets on the network stack can be accessed before they are sent out of the network interface. The packets can then be filtered out or altered and re-injected back onto the network stack in the kernel. Similarly, packets that are received can be filtered or altered and handed back up the stack for processing by the application. This functionality can be exploited to apply header compressions to outbound packets and decompression to inbound packets according to the ROHC specification.

The main advantage to doing these operations in user space, as opposed to in the kernel, is that it will allow for increased flexibility and ease of programming. If the filter was coded in the kernel, the kernel would need to be recompiled for each change of the code. Additionally, it would be more difficult to debug the code as segmentation faults do not lend themselves to straight forward solutions.

The main disadvantage of implementing ROHC in the user space is that the queue time for a packet will increase. A packet is generated by an application and then passed onto the network stack. The packet is given a transport header (i.e. TCP, UDP) and then an IP header (i.e., IPv6). At this point, it must be grabbed off of the stack and taken into user space. The header will be compressed and then passed back onto the stack in the kernel. This is assuredly slower than the packet simply remaining in the kernel. Even so, the issue is not troublesome for two reasons. First, every packet sent or received that must be compressed is offset by the same amount which results in a net effect of shifting the whole flow by a small delta. Second, the time for the packet to move to user space and back is still negligible compared to the high transmission times of low bandwidth channels. On high bandwidth channels where this delay may become a problem, header compression would not likely be implemented.

4.2 Implementation Validation

To test the implementation, a constant stream of 40 byte UDP packets was generated using MGEN [34] to mimic a low bandwidth real-time stream with no packet losses and a constant delay. Both an uncompressed stream and a compressed stream using the ROHC IPv6/UDP compression profile are shown in Figure 4.3.

The uncompressed packets are all comprised of a 40 byte IPv6 header, 8 byte UDP header, and 40 byte data payload for a total of 88 bytes.

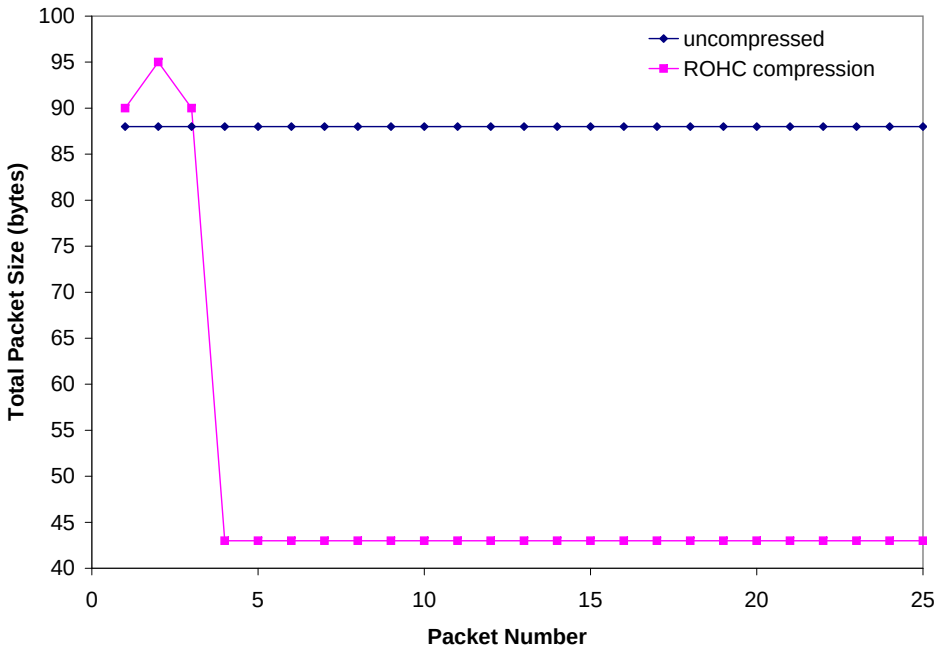

Figure 4.3: Compressed and uncompressed total packet size.

The packets compressed by ROHC begin slightly larger than the uncompressed packets. The first three packets are larger due to an increased header size of 50 bytes as confirmed by Figure 4.4. These are IR packets which are used while the protocol is in the IR state. This state is intended to develop the static parts of the compression context at the decompressor which is necessary to initialize a stream and to recover after a failure. In this case, the stream has just started so the static context is being initialized. After these first three packets, the compression begins and the header size is reduced to 3 bytes. This is a reduction of 45 bytes from the uncompressed header. Overall, the original stream was compressed to 55.5% of the original size for these 25 packets. If no losses are assumed, the compression ratio would approach 0.489 as the number of packets in the stream approaches infinity. The compression ratio is defined as the total size of the compressed packet divided by the total size of the corresponding uncompressed packet.

Clearly, there is a minor trade-off associated with the negotiation of the context state. There is overhead incurred by the IR packets. For ROHC, it typically takes three packets to establish the context which is additional 8-16 bytes of overhead. Accordingly, if a transmission was short enough (less than five packets), it would not warrant establishing a context to apply compression. The setup overhead becomes negligible for any amount of traffic past this level. The link context

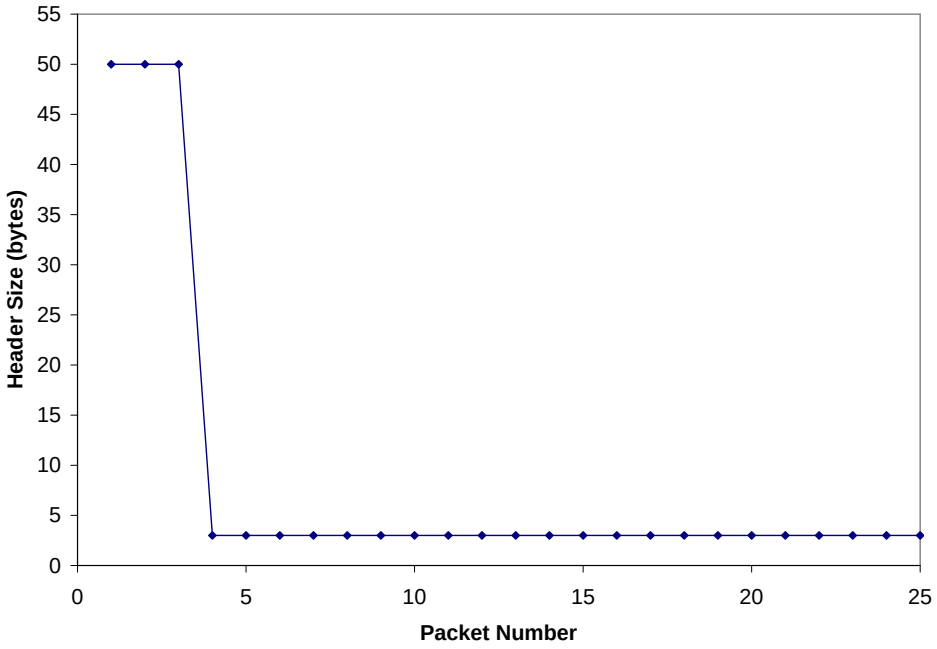

Figure 4.4: ROHC header size.

does not expire either, so even if the transmission is short, it is still worthwhile to establish the compression if it is anticipated that the link will be used in the future.

4.3 Protocol Testing

ROHC is built on a framework that is extendable to any protocol for which a profile is developed. We investigated some of the more common protocols and their interaction with header compression.

4.3.1 IPv6

Table 4.1: IPv6 header format.

bits	**0-3**	**4-11**	**12-31**	**32-47**	**48-55**	**56-63**
0	Ver.	Traffic Class	Flow Label	Payload Length	Next Header	Hop Limit
64	Source Address					
192	Destination Address					

Table 4.2: Description of IPv6 header fields.

Field	Description	Size (bytes)
Characterization	Various parameters associated with the flow	8
Source Address	IP address of the destination	16
Destination Address	IP address of the destination	16
	Total Size	40

4.3.2 User Datagram Protocol (UDP)

UDP is a transport layer (OSI model, layer 4) protocol. It is connectionless which means there is no handshaking process involved between communicating agents, unlike TCP. It does not guarantee reliability or packet ordering such that datagrams may be out of order, duplicated, or lost without notice. Furthermore, no congestion control is performed to scale back the transmit rate when the network is over loaded.

Table 4.3: UDP header format.

bits	0-15	16-31
0	Source Port Number	Destination Port Number
32	Length	Checksum

Table 4.4: Description of UDP header fields.

Field	Description	Size (bytes)
Source Port	UDP port of the source	2
Destination Port	UDP port of the destination	2
Length	Length of the packet header and data payload	2
Checksum	1's complement of 16-bit words across the packet	2
	Total Size	8

UDP is extremely useful for a myriad of network applications which do not depend on reliable transfer. All real-time traffic, such as video streaming and Voice-over-IP (VoIP), falls into this category, as traffic that is lost or delayed is not useful for immediate playback. Many network utilities and management protocols also utilize UDP including Domain Name System (DNS), Dynamic Host

Configuration Protocol (DHCP), Simple Network Management Protocol (SNMP), and Routing Information Protocol (RIP). The compression ratio achieved for IPv6/UDP packets over a normal wireless connection can be found in Figure 4.5. Transmitted over IPv6, UDP packets have a total header size of 48 bytes.

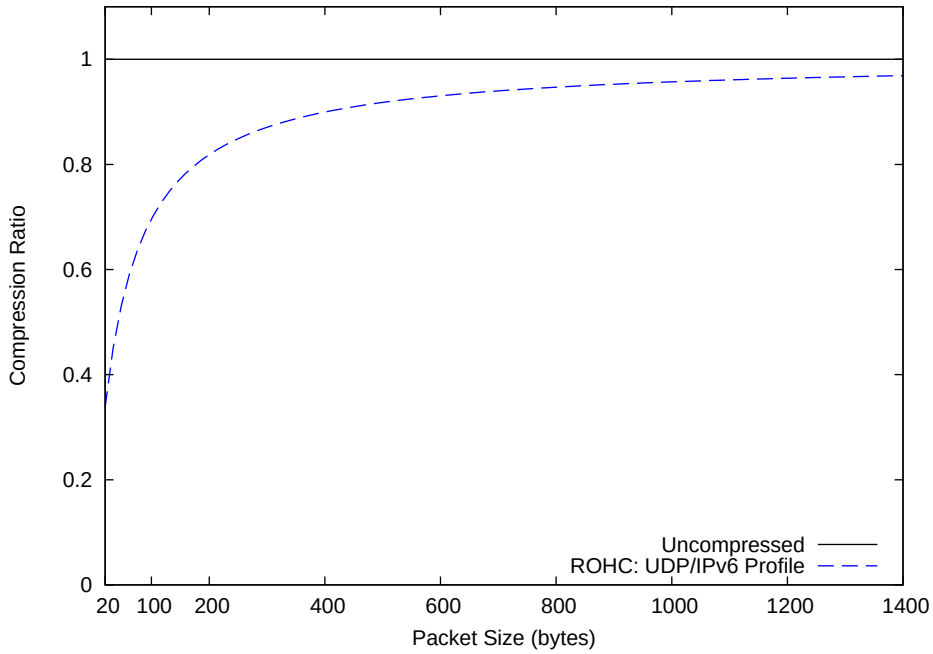

Figure 4.5: ROHC compression ratio for UDP/IPv6.

4.3.3 User Datagram Protocol Lite (UDP-Lite)

UDP Lite was developed for error-prone environments. Normal UDP packets are discarded if they have an invalid checksum which can be caused by a single bit error anywhere across the packet. UDP Lite allows for the scope of the checksum to be limited to the header. This is useful for applications where damaged payloads are preferable to discarded packets, such as VoIP. Otherwise, it operates the same as UDP and has the same header size where the length field has been replaced by the checksum coverage. The length field is able to be excluded because the information can be extracted from the other headers on the datagram.

Since UDP Lite has the same header size and the same operation as UDP with the exception of

Table 4.5: UDP-Lite header format.

bits	**0-15**	**16-31**
0	Source Port Number	Destination Port Number
32	Checksum Coverage	Checksum

Table 4.6: Description of UDP-Lite header fields.

Field	**Description**	**Size (bytes)**
Source Port	UDP port of the source	2
Destination Port	UDP port of the destination	2
Checksum coverage	Length covered by the checksum	2
Checksum	1's complement of 16-bit words across the length defined by the checksum coverage	2
	Total Size	8

how the checksum is calculated, ROHC compresses it the same as it would normal UDP. Therefore, Figure 13 may be referenced for the compression ratio of UDP Lite over a wireless link as well.

4.3.4 Real-time Transport Protocol (RTP)

RTP is an application layer protocol used for streaming real-time media, such as VoIP, videos, and video conferencing. RTP is used in conjunction with UDP. It enables stream synchronization through timestamps and sequence numbering and also indicates format of encapsulated media.

Table 4.7: RTP header format.

bits	**0-1**	**2**	**3**	**4-7**	**8**	**9-15**	**16-31**
0	Ver.	P	X	CC	M	PT	Sequence Number
32	Timestamp						
64	SSRC Identifier						
96	CSRC Identifiers (optional)						

Figure 4.6 shows the compression ratio for a RTP/UDP/IPv6 stream with no additional concatenated streams (12B RTP header). On top of IPv6 and UDP, RTP packets have a total header size of 60 bytes.

Table 4.8: Description of RTP header fields.

Field	**Description**	**Size (bytes)**
Characterization	Various parameters associated with the flow	2
Sequence Number	Sequence number of the packet	2
Timestamp	The time value used for stream synchronization	4
SSRC Identifier	An identifier for the stream	4
CSRC Identifier (optional)	The identifier numbers for streams that should be concatenated onto the current stream	(4)
	Total Size	12 (16, 20, . . .)

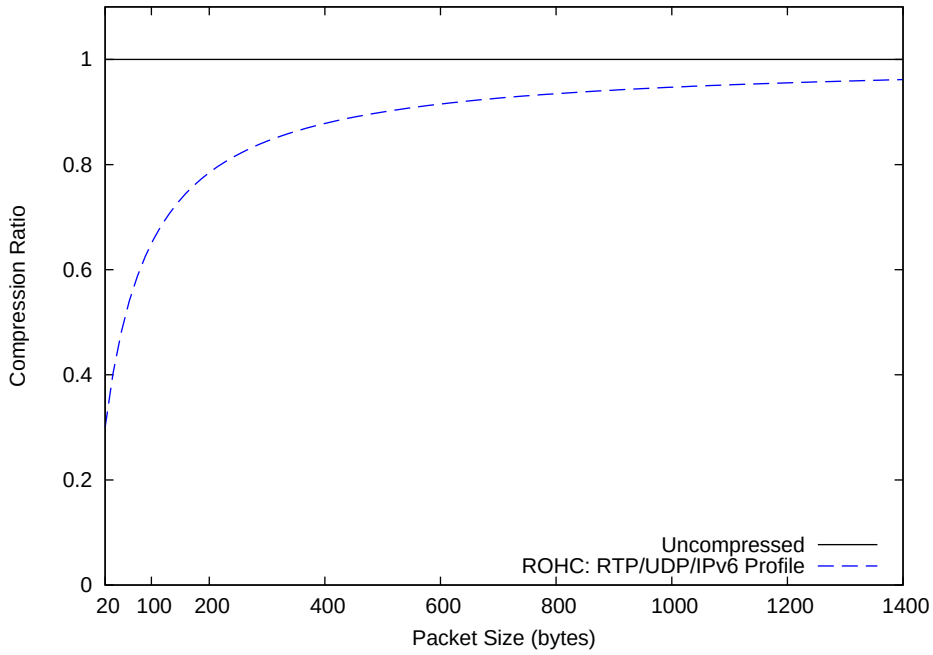

Figure 4.6: ROHC compression ratio for RTP/UDP/IPv6.

4.3.5 Internet Control Message Protocol Version 6 (ICMPv6)

ICMPv6 is a network layer protocol used for diagnostics and error reporting. Error reporting includes messages such as time exceeded (TTL expired), destination unreachable, and packet too big among others. Informational messages can also be sent using ICMPv6 for pings, neighbor advertisements/solicitations, and router advertisements/solicitations. ICMPv6 is sent over IPv6 as a header extension.

Table 4.9: ICMPv6 header formats.

bits	**0-7**	**8-15**
0	Type	Code
16	Checksum	

Table 4.10: Description of ICMPv6 header fields.

Field	**Description**	**Size (bytes)**
Type	Type of message: error or informational	1
Code	Type dependent	1
Checksum	1's complement of 16-bit words across the length of the header	2
	Total Size	4

The ICMPv6 protocol compression was tested using pings. Figure 4.7 shows the compression ratio for a ICMPv6/IPv6 packet. On top of IPv6, ICMPv6 packets have a total header size of 44 bytes.

4.3.6 Protocols Compared

Figure 4.8 shows all of the aforementioned protocols combined onto a single plot. The RTP/UDP/IPv6 stream achieves the greatest compression ratio because the RTP/UDP/IPv6 packets have the largest header (60 bytes). The UDP and ICMPv6 streams have nearly the same compression ratio curve due to their relatively close total header sizes which are 48 and 44 bytes, respectively.

The figure shows encouraging results for ROHC compression on normal (non-lossy) wireless links.

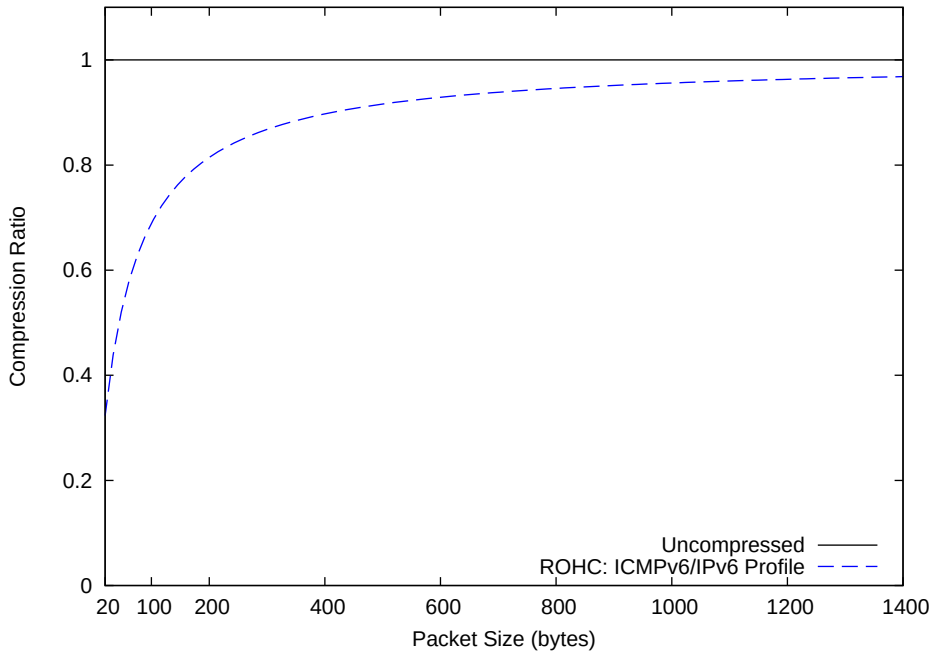

Figure 4.7: ROHC compression ratio for ICMPv6.

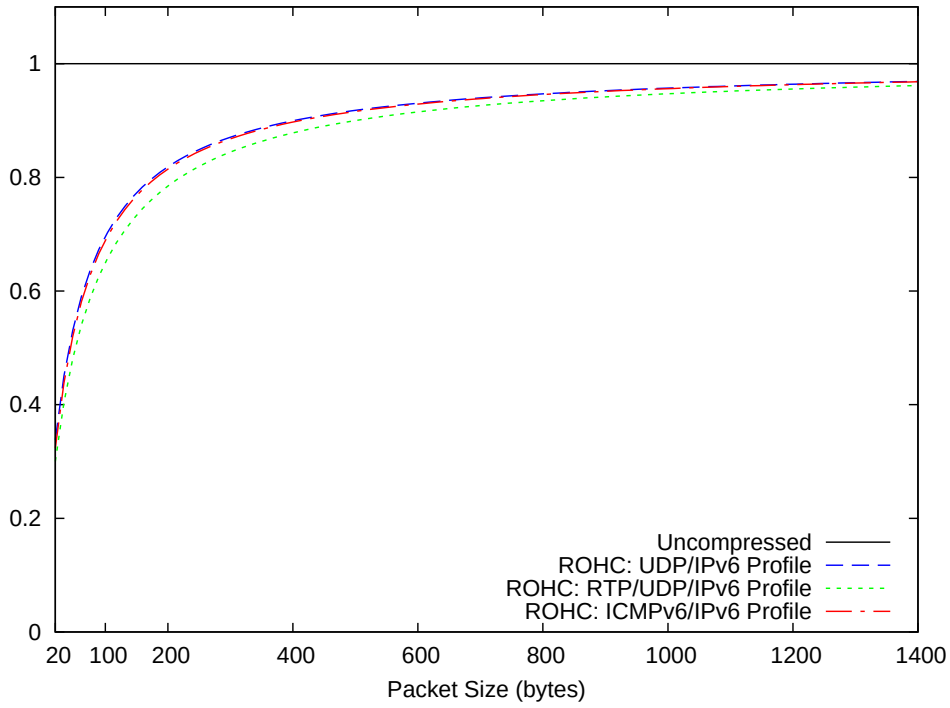

Figure 4.8: ROHC compression ratio comparison.

At low payload sizes, a substantial compression ratio can be achieved for all of the tested protocols. Even at large payload sizes near normal the Maximum Transmission Unit (MTU), there is still an appreciable gain from the compression.

A quick clarification must be made concerning the interpretation of the compression ratio plots. It is true that for all profiles, a small packet size leads to an increased compression ratio. This is intuitive since the header size remains constant and is independent of the size of the payload. As the payload size decreases, the header becomes an increasingly large percentage of the overall packet. This indicates that compressing the header will have a greater effect for small packets. Even so, it does not imply that it is preferred for an application to send small packets. Total overhead will always be minimized by sending the largest payload allowed. This always conserves bandwidth more so than fragmenting a packet and compressing the headers.

4.4 Performance Under Network Constraints

4.4.1 Packet Loss

Packet loss refers to the event that a packet transmission fails to reach its destination. In a wireless environment, packet loss can occur for numerous reasons including channel interface, physical obstruction, and multi-path fading among others. In this case, none of the packet is received, so the data must be resent or skipped. This is problematic for CRTP since a loss is not detected until the next packet arrives. If the loss occurs in a burst, it is possible many packets will have been lost before the decompressor knows about it. At that point, the context state will have to be reestablished, causing the failed decompression of several more packets in the process.

In the presence of packet loss, only a slight increase in overhead was observed for ROHC. This seemed counter-intuitive at first, but closer inspection of the operation of the protocol revealed its resilience [44]. Initially, the compression channel is opened in unidirectional mode. Once the first packet is received by the decompressor, it responds with a request to transition into optimistic mode. The purpose of the optimistic mode is to maximize compression efficiency while limiting the amount of traffic sent via the feedback channel. The feedback is only used to send error recovery messages which are necessary after a context invalidation. It is important to note that the context is not invalidated after every lost or damaged packet. It is possible for lower layers to catch and discard erroneous packets. Also, it is possible for a packet to be damaged without corrupting the checksum or the compressed header. In this latter scenario, the packet will still be decompressed and passed

to the upper layers. This results in sparse usage of the feedback channel and minimal overhead.

ROHC was designed to withstand packet loss over wireless links. Static fields of the header never change so they may be decompressed regardless of lose packets. Furthermore, ROHC allows for the change of the differential fields to be applied multiple times. This means ROHC is most prone to decompression failures when the packet is received but the compressed header is corrupted.

In experiments, ROHC performed as well in a lossy environment ($<$ 20% loss) as in an ideal environment. Essentially, it was still able to operate appropriately for levels of loss where the data itself is still useful. Normally, once a link is experiencing more than 5% to 10% loss, applications will begin to break. ROHC is not a reliable protocol. It is only meant to compress the link, not provide recovery mechanisms or data assurance. Therefore, if necessary, loss has to be handled by one of the other link protocols.

4.4.2 Bit Errors

A bit error occurs when a single bit is altered in the presence of noise. The noise is caused by a multitude of sources in a wireless channel. Bit errors pose more of a problem for ROHC than packet loss since a packet can be received in error. The packet header will fail to decompress if the bit error occurs in the header. Meanwhile, the stream context will only be invalidated when a bit error occurs in the checksum for the ROHC compressed header. When this occurs, a negative acknowledgement (NACK) must be sent to the compressor to reestablish the context. Fortunately, due to the small size of the compressed header, it is unlikely bit errors will frequently occur in the header. In testing, only a few NACKs were observed even at relatively high BER Rates of 1e4 to 1e5 (1 error per 1,000 to 10,000 bits). At higher bit error rates, ROHC will frequently fail to decompress. Though, at those levels of bit error rate, uncoded data payloads will already be unusable regardless of whether the header is correct or not.

4.4.3 Delay

Packet delay is the time necessary for a packet to get from the transmitter to the receiver. ROHC is mostly tolerant packet delay since feedback is used sparingly even while in bidirectional mode. Theoretically, packet delay can be problematic for compression schemes which rely more heavily feedback. This is because high delay will result in many packets being sent before the first one is received. If an error occurs or a resynchronization is required, all the packets transmitted in the interim will be unable to be decompressed. Fortunately, ROHC responds well to packet

loss as previously discussed. Feedback is not significantly employed. Positive acknowledgments are sent for significant context updates in O-mode and all context updates in R-mode. Negative acknowledgments are sent when the decompressor is out of sync which is indicated by successive packets failing to decompress.

4.4.4 Packet Reordering

Packet reordering occurs when a packet with a newer sequence number arrives at the receiver before a packet with an older sequence number. Reordering normally only occurs on larger wired networks and virtual circuits where it is possible for a packet to take different paths to reach the receiver. On a wireless point-to-point link, where protocols such as ROHC and CRTP are intended to be used, packet reordering does not occur. Neither ROHC nor CRTP are tolerant to packet reordering. Both protocols would assume the unreceived, older packet has been lost. Fortunately, ECRTP was developed specificially with this purpose in mind. For networks in need of header compression where packet reordering is a possibility, ECRTP is the preferred protocol.

4.4.5 Bandwidth

The main goal of all network compression protocols is to reduce overhead in order to improve goodput. Goodput is the rate of useable data being sent. Alternatively, it can be defined as the application level throughput since it does not include overhead from network protocols. Reducing overhead through compression improves the goodput for a constant network throughput. To exemplify this, Figure 4.9 shows a series of plots comparing the goodput of compressed and uncompressed UDP streams for packets of size 40B, 100B, 500B, and 1400B. The goodput is plotted against the bandwidth (theoretical maximum throughput) of the channel. Each point represents an average of several trials.

For each tested bandwidth and packet size, the compressed stream had a higher goodput relative to the uncompressed. Coinciding with our compression ratio curves, the gain in goodput is substantially greater for smaller packet sizes. This may be seen by comparing the 40B packet size to the 1400B packet size. Compressing the packet brings the goodput closer to the theoretical limit which is the channel bandwidth, even at low bandwidths such as 9.6kb and 64kb.

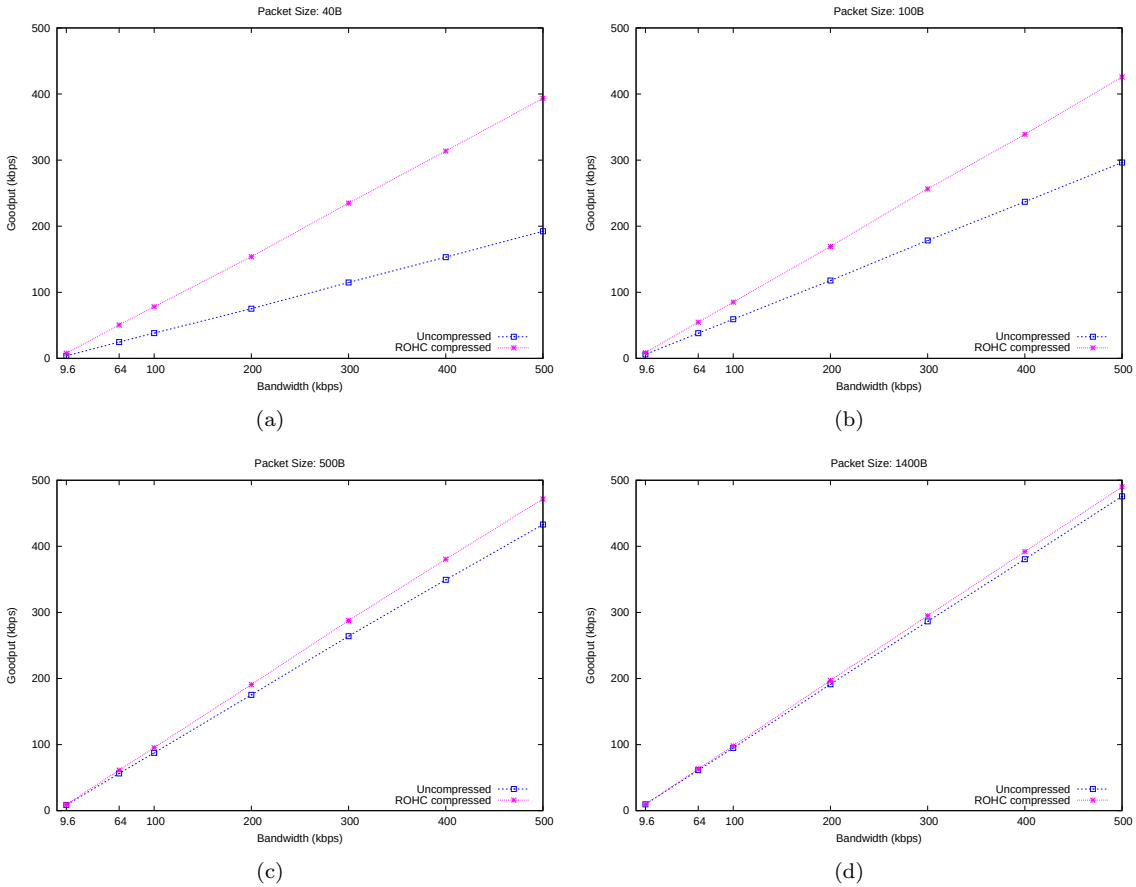

Figure 4.9: Goodput for compressed and uncompressed UDP streams.

5. Conclusions and Recommendations

5.1 Application and Scenario Characterization

Since the compression achieved for a packet is relative to the size of the packet, it is necessary to characterize the traffic which will be sent over the compressed link. The expected compression ratio for a given application can be determined by looking at the distribution of the packet size for an application.

Many applications have nearly constant packet sizes. Such applications commonly perform network diagnostics and management or situation awareness. The packet sizes rarely change since the packet format is well defined and the same fields are included for each packet. A list of some common network applications is shown in Table 5.1 along with their average packet size, incorporated headers, and the expected compression ratio.

Table 5.1: Compression ratio for common network protocols.

Application	**Headers Incorporated**	**Average Packet Size (bytes)**	**Expected Compression Ratio**
DHCP	IPv6, UDP	102.21	0.7071
DNS	IPv6, UDP	390.68	0.8997
ICMPv6			
- Error Messages	IPv6, ICMPv6	variable	variable
- Echo Request	IPv6, ICMPv6	4	0.1667
- Echo Reply	IPv6, ICMPv6	4	0.1667
- Router Solicitation	IPv6, ICMPv6	12	0.2857
- Router Advertisement	IPv6, ICMPv6	12	0.2857
- Neighbor Solicitation	IPv6, ICMPv6	20	0.3750
- Neighbor Advertisement	IPv6, ICMPv6	20	0.3750
- Multicast Listener	IPv6, ICMPv6	32	0.4737

Many user applications have traffic flows with higher variance packet sizes. An example of such a flow is an internet session where the user visits various web pages and downloads images and text. An internet session was captured, and its packet size distribution is shown in Figure 5.1. Interestingly, the packet sizes are either very small (< 50B) or very large (> 1400B). The small packets are mostly a mixture of HTTP calls and handshakes that occur when using the TCP protocol for HTTP

requests. The large packets are delivering the images and other media requested when accessing a page. To find the expected compression ratio in this scenario, a summation of the compression ratio multiplied by the probability density at each discrete packet size was calculated. For this example, the compression ratio was 0.6562.

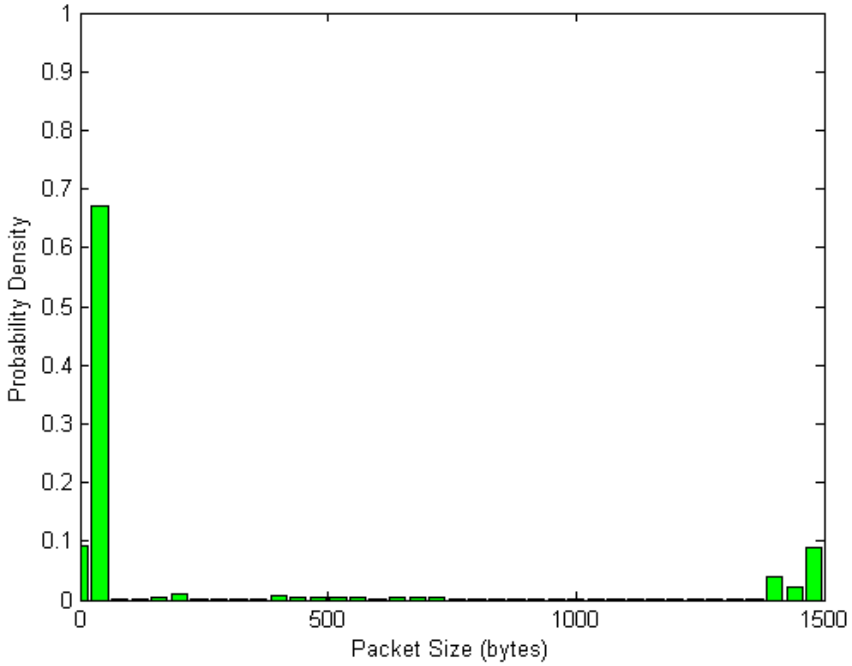

Figure 5.1: Probability density of the packet size for a http web session.

Another example of traffic with variable packet sizes is real-time traffic, such as VoIP and video streaming. This data is time sensitive such that packets are sent at time intervals required by the data payload contained. Accordingly, packets are not always filled to the MTU. The packet size distribution is shown in Figure 5.2 for a five minute VoIP call. A large percentage of the packets are less than 100B such that compression ratio achieved for this call was 0.5964.

For both applications with low variance packet size and high variance packet size, appreciable compression ratios can be achieved. This is due to the fact that payloads are not always filled up to the MTU. Smaller packets are used to convey short messages, updates, or time sensitive information. Only file transfers are going to employ primarily large packet sizes. Header compression is a useful

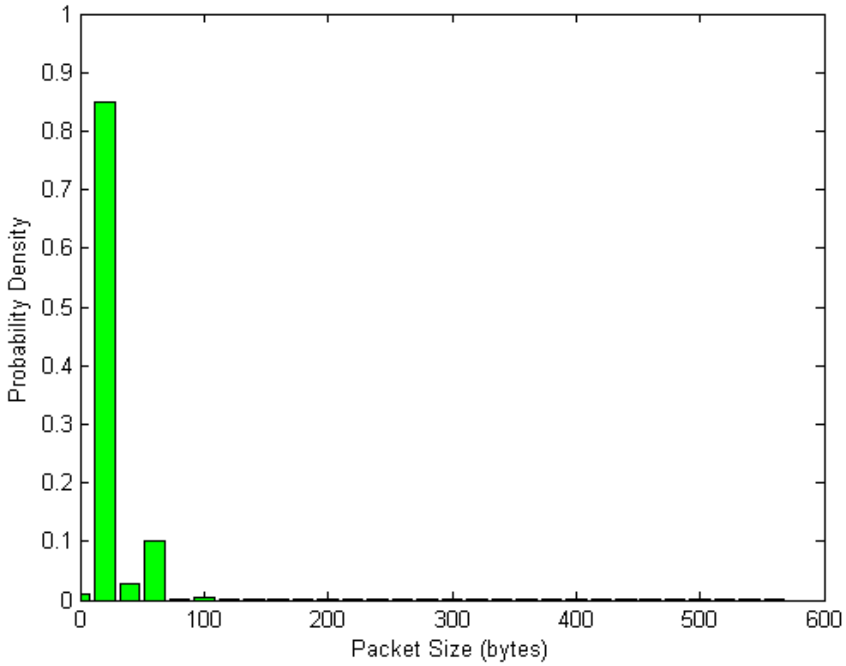

Figure 5.2: Probability density of the packet size for a VoIP call.

asset for reducing overhead and conserving bandwidth for IPv6 network traffic.

5.2 Conclusions

5.2.1 Header Compression Schemes

It has been shown that header compression is useful for reducing overhead, especially for traffic with low average packet sizes. By reducing overhead to improve goodput, header compression converses bandwidth which is most useful on low bandwidth links where high goodput is a top priority. The benefits of header compression on faster links is debateable.

CRTP, ECRTP, and ROHC were investigated and evaluated due to their maturity, current level of integration, and perceived utility in future applications. CRTP was intended to work over point-to-point links with short delays and no packet reordering. It performs well in such scenarios and has already been employed commercially in this manner. ECRTP was designed to allow for compression over IP tunnels and virtual circuits where there is both long delays and packet reordering. The enhancements employed in ECRTP do increase the overhead, yet the compression savings remains non-trivial. ROHC was designed for point-to-point wireless links so that it would be robust against

packet loss while achieving the maximum level of compression efficiency.

Furthermore, ROHC performs well in the face of various network constraints. It is generally unaffected by packet loss, packet delay, and jitter. It is most susceptible to bit errors which can corrupt the compressed header. Even so, in environments with high bit error rate, the data payload will already be corrupted regardless of the success of the header decompression. It is suggested that ROHC should be used over CRTP or ECRTP for point-to-point links. ROHC, similar to CRTP, cannot tolerate packet reordering, so ECRTP should still be used in the case where the link travels over an IP tunnel.

5.2.2 Packet Payload Compression Schemes

Packet compression schemes certainly can save significant amounts of bandwidth when they are properly utilized. Yet, similar to header compression algorithms, either full context must be included for each packet or a synchronization mechanism is required. In addition, packet compression requires both processing and memory capacity while being rendered useless if the data has already been compressed by a higher layer or encrypted. This essentially makes it unusable for video streams, audio streams, other compressed files, and anything encrypted which severely constrains its applications.

HP recommends that packet compression only be employed for streams or packets that are less than 768kbps [1]. This is due to the processing delay incurred by compressing the payload of a packet. The processing time increases as the size of the packet increases. For slower links, this is not a problem as the transmission times are high. When the processing time exceeds the transmission time on the link, then the compression algorithm will actually begin to decrease bandwidth. Additionally, it is shown in [3] that the compression ratio for packet compression does yield positive gain until the packet size is over 200 bytes.

From these observations, it is possible to see that there is only a window of packet sizes in which packet payload compression would be beneficial. The upper boundary is set by the processing time relative to the transmission time. The processing could become a limiting factor for field devices with low memory and computational powers. The lower threshold is set at the point at which the compression ratio is falls blow unity. It is harmful to operate with packet compression under this limit as the compression savings is less than the overhead incurred by sending or synchronizing the dictionary.

It appears that packet payload compression is not a viable option. The conditions imposed are too restricting for it to be useful in practice. The highest gains should be achieved through the use

of header compression. If the data itself needs to be compressed, it is most likely better suited to perform the action at a high layer.

Bibliography

[1] Hp case study: Wan link compression on hp routers. http://1541114www.hp.com/rnd/support/manuals/pdf/comp.pdf, 1995.

[2] The rocco homepage. http://1541114www.ludd.luth.se/~larsman/rocco/>, June 2000.

[3] A review of ip packet compression techniques. 2003.

[4] Rohc . robust header compression, 2003. http://rohc.sourceforge.net/>, 2003.

[5] The experts in ip header compression. http://www.effnet.com/sites/effnet/pages/uk/default.asp, 2004.

[6] Qualcomm licenses ip header compression software. http://www.cellular-news.com/story/13751.php, August 2005.

[7] Ieee standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans). IEEE Standard, 2006.

[8] Enable or disable ip header compression in ppp. http://technet.microsoft.com/en-us/library/cc754846(WS.10).aspx, February 2008.

[9] blip, the berkeley ip implementation for low-power. http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip, June 2009.

[10] Contiki, the operating system for embedded smart objects - the internet of things. http://www.linuxfoundation.org/collaborate/workgroups/networking/netem, 2009.

[11] Express rtp and tcp header compression. http://www.cisco.com/en/US/docs/ios/12_0t/12_0t7/feature/guide/rtpfast.html, 2009.

[12] Iris, wireless measurement system. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/IRIS_Datasheet.pdf, 2009.

[13] Lossless data compression. http://en.wikipedia.org/wiki/Lossless_data_compression>, September 2009.

[14] netem. http://www.linuxfoundation.org/collaborate/workgroups/networking/netem, 2009.

[15] Tinyos. http://www.tinyos.net/>, April 2009.

[16] Wsn starter kit. http://www.xbow.com/Products/productdetails.aspx?sid=230, 2009.

[17] Arch rock. www.archrock.com, May 2010.

[18] Atmel. www.atmel.com, May 2010.

[19] Ipv6 low power personal area network parameters. http://www.iana.org/assignments/6lowpan-parameters/6lowpan-parameters.xhtml, March 2010.

[20] Sensinode. www.sensinode.com, May 2010.

[21] Wireshark, a network protocol analyzer. www.wireshark.org, May 2010.

[22] J. Abley, P. Savola, and G. Neville-Neil. Deprecation of Type 0 Routing Headers in IPv6. RFC 5095 (Proposed Standard), December 2007.

[23] B. Barvaux. Robust header compression (rohc) library. `https://launchpad.net/rohc`, 2009.

[24] C. Bormann. Robust Header Compression (ROHC) over PPP. RFC 3241 (Proposed Standard), April 2002. Updated by RFC 4815.

[25] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed. RFC 3095 (Proposed Standard), July 2001. Updated by RFCs 3759, 4815.

[26] B. Bougard, F. Catthoor, D.C. Daly, A. Chandrakasan, and W. Dehaene. Energy efficiency of the ieee 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives. pages 196 – 201 Vol. 1, mar. 2005.

[27] S. Casner and V. Jacobson. Compressing IP/UDP/RTP Headers for Low-Speed Serial Links. RFC 2508 (Proposed Standard), February 1999.

[28] J. Davies. *Understanding IPv6*. Microsoft Press, 2nd edition, January 2008. ISBN-10: 0735624461.

[29] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFCs 5095, 5722, 5871.

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

[31] R. Finking and G. Pelletier. Formal Notation for RObust Header Compression (ROHC-FN). RFC 4997 (Proposed Standard), July 2007.

[32] S. Hagen. *IPv6 Essentials*. O'Reilly Media, 2nd edition, May 2006. ISBN-10: 0596100582.

[33] I-Hsuan Huang, Chun-Shou Lin, Ching-Sung Chen, and Cheng-Zen Yang. Design of a qos gateway with real-time packet compression. pages 1 –4, oct. 2007.

[34] J. Hui and P. Thubert. Compression format for ipv6 datagrams in 6lowpan networks. Internet-Draft, September 2010.

[35] L-E. Jonsson. RObust Header Compression (ROHC): Terminology and Channel Mapping Examples. RFC 3759 (Informational), April 2004.

[36] L-E. Jonsson and G. Pelletier. RObust Header Compression (ROHC): A Compression Profile for IP. RFC 3843 (Proposed Standard), June 2004. Updated by RFC 4815.

[37] L-E. Jonsson, G. Pelletier, and K. Sandlund. RObust Header Compression (ROHC): A Link-Layer Assisted Profile for IP/UDP/RTP. RFC 4362 (Proposed Standard), January 2006. Updated by RFC 4815.

[38] L-E. Jonsson, K. Sandlund, G. Pelletier, and P. Kremer. RObust Header Compression (ROHC): Corrections and Clarifications to RFC 3095. RFC 4815 (Proposed Standard), February 2007.

[39] L.E. Jonsson. Rocco - a header compression framework and realized schemes. `http://www.ludd.luth.se/~larsman/rocco/presentations/ietf47-rohc-rocco.pdf`, March 2000.

[40] Hailing Ju and Li Cui. Easipc: a packet compression mechanism for embedded wsn. pages 394 – 399, aug. 2005.

[41] Sangkil Jung, Sangjin Hong, Kyungtae Kim, Junghoon Jee, and Eunah Kim. Voice transmission enhancing model on wireless mesh networks. pages 4949 –4954, jun. 2007.

[42] T. Koren, S. Casner, and C. Bormann. IP Header Compression over PPP. RFC 3544 (Proposed Standard), July 2003.

[43] T. Koren, S. Casner, J. Geevarghese, B. Thompson, and P. Ruddy. Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering. RFC 3545 (Proposed Standard), July 2003.

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

[45] E. Martinez, A. Minaburo, and L. Toutain. Cross layer rohc compression for multicast video streaming. pages 2878 –2882, may. 2008.

[46] Y. Matias and R. Refua. Delayed-dictionary compression for packet networks. volume 2, pages 1443 – 1454 vol. 2, mar. 2005.

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

[48] Sangheon Pack, Joo-Chul Lee, and Jung-Soo Park. Hybrid robust header compression in proxy mobile ipv6 over wireless mesh networks. pages 415 –419, may. 2008.

[49] G. Pelletier and K. Sandlund. RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite. RFC 5225 (Proposed Standard), April 2008.

[50] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. Spins: security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, 2002.

[51] P. Rawat, J.M. Bonnin, and A. Minaburo. Optimising the use of robust header compression profiles in nemo networks. pages 150 –155, apr. 2008.

[52] P. Rawat, L. Toutain, and A. Minaburo. Using sigcomp and rohc in wireless networks. pages 1 –5, feb. 2008.

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

[54] A. Shacham, B. Monsour, R. Pereira, and M. Thomas. IP Payload Compression Protocol (IPComp). RFC 3173 (Proposed Standard), September 2001.

[55] A. Shacham, R. Monsour, R. Pereira, and M. Thomas. IP Payload Compression Protocol (IPComp). RFC 2393 (Proposed Standard), December 1998. Obsoleted by RFC 3173.

[56] Z. Shelby. Neighbor discovery optimization for low-power and lossy networks. Internet-Draft, April 2010.

[57] K. Svanbro, H. Hannu, L.-E. Jonsson, and M. Degermark. Wireless real-time ip services enabled by header compression. volume 2, pages 1150 –1154 vol.2, 2000.

[58] C. Westphal. Layered ip header compression for ip-enabled sensor networks. volume 8, pages 3542 –3547, jun. 2006.

[59] Hyunje Woo, Jooyoung Kim, Meejeong Lee, and JeongMin Kwon. Performance analysis of robust header compression over mobile wimax. volume 3, pages 1742 –1746, feb. 2008.

[60] G. Zhou, C. Huang, T. Yan, T. He, J. A. Stankovic, and T. F. Abdelzaher. Mmsn: Multi-frequency media access control for wireless sensor networks. pages 1 –13, apr. 2006.

[61] Yi Zou and Krishnendu Chakrabarty. Sensor deployment and target localization in distributed sensor networks. *Trans. on Embedded Computing Sys.*, 3(1):61–91, 2004.