Journal of Theoretical and Applied Information Technology
20th May 2016. Vol.87. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PERFORMANCE CHARACTERIZATION OF ROHC FOR SATELLITE-BASED UNIDIRECTIONAL LINKS USING ERROR-FREE CHANNELS

[1]WAY-CHUANG ANG, [2]MUHAMMAD-IMRAN SARWAR, [3]TAT-CHEE WAN
[1,2,3]National Advance IPv6 Centre (NAv6), Universiti Sains Malaysia, 11800 USM, Pulau Pinang, Malaysia
[3]School of Computer Sciences, Universiti Sains Malaysia, 11800 USM, Pulau Pinang, Malaysia
E-mail: [1]wcang79@gmail.com, [2]imran@nav6.usm.my, [3]tcwan@usm.my

ABSTRACT

Satellite communication systems play a vital role in providing Wide Area Network (WAN) due broader coverage but at the same time impose challenge for IP services in unidirectional Satellite link. This research evaluates RObust Header Compression (ROHC) for Unidirectional Lightweight Encapsulation (ULE) in terms of network performance and practical implementation design of a ROHC via Satellite test-bed. Moreover, mathematical model was presented to estimate the theoretical performance characteristics of ROHC compressed traffic which is then compared with empirical results. The experiments showed that ROHC delivered significant improvement in bandwidth utilization for packets with small payload sizes with up to 86% gain in throughput performance when compressing traffic. Packets with larger payload sizes exhibited an exponential decrease of throughput gain achievable through ROHC as the size of the payload increased. This paper also discusses the effectiveness of ROHC for IPv4 versus IPv6 traffic that was evaluated indicates IPv6 traffic streams benefited to a greater degree from ROHC than did IPv4 traffic streams, even on non-ideal links.

Keywords: *Satellite Communication, Unidirectional Lightweight Encapsulation (ULE), RObust Header Compression (ROHC), Digital Video Broadcasting – Satellite (DVB-S), Mesh Networking,*

1. INTRODUCTION

Satellite communication systems play a vital role in providing Wide Area Network due to their broadcast nature and wide geographical coverage as well as remote areas. Naturally, with the exponential growth of the Internet, satellite communication takes on a growing role in providing Internet Protocol (IP) services. Although the majority of IP services assume that the underlying transport medium is bidirectional in nature, satellite links are unidirectional. This condition presents a challenge to the provision of IP services over the satellite communication system. An approach such as the Link-Layer Tunneling Mechanism was proposed to overcome this shortcoming [1]. The Digital Video Broadcasting via Satellite (DVB-S) system is a standard developed by the DVB project to deliver digital content over satellite links. The DVB-S system is more commonly used to deliver audio/video content. To deliver IP packets over DVB-S, Multi-Protocol Encapsulation (MPE) [2] was developed to carry IP packets over the baseband of a DVB-S system, utilizing MPEG2 Transport Stream (MPEG2-TS) frames. However, due to the complexity and overhead of MPE, Unidirectional Lightweight Encapsulation (ULE) was later developed by the IETF as a better alternative to deliver IP packets over the same MPEG2-TS frames.

For end-to-end delivery of data over the Internet, IP and higher-layer headers are needed to ensure that data are sent to the designated recipients. However, for the delivery of packets from hop-to-hop, link-layer addresses alone are sufficient. Thus, for the provision of IP services over satellite communication systems, the combined overhead of MPEG2-TS frames, ULE, data link-layer headers, IP, as well as transport, headers, leads to inefficient bandwidth usage. The wastage of bandwidth is more significant when the payload sizes are small. For typical GSM-encoded VoIP traffic over an IPv6 network, the size of the audio data is less than the total size of the headers in an RTP packet. By applying header compression to the IP traffic, the incurred overhead can be reduced. Satellite communication systems are susceptible to noise introduced by the propagating medium, and this attribute is common to all wireless communication technologies.

Journal of Theoretical and Applied Information Technology
20th May 2016. Vol.87. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

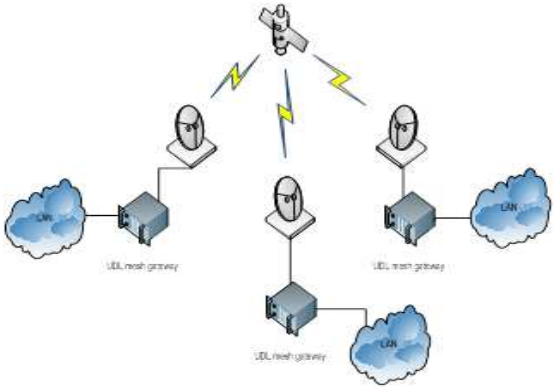

Figure 1: A UDL Mesh Network Consisting Of Three Sites

Several researchers have been working to leverage the ability to utilize header compression technique in the past. Most of these header compression were target to wired networks. In 1990, one of the first header compression technique CTCP or VJHC was introduced by Van Jacobson [3]. Perkin and Mutka introduces a new header compression scheme [4] that is more robust than VJHC but at the cost of less compression. This is done by introducing reference header that will be updated regularly. VJHC was enhance in IP Header Compression (IPHC) [5] technique that supports multiple headers such as UDP, IPv6 with additional TCP features. The effectiveness of header compression was enabled when Compressed Real Time Protocol (CRTP) [6] can able to compress 40 bytes of IPv4 packet header to 4 bytes compressed header if the checksum of UDP header is enabled. Although these number of mechanisms for compressing the headers of IP traffic are mentioned, the present research deals with RObust Header Compression (ROHC) [7] exclusively because it is IETF standard and apart from header compression it has the ability to tolerate losses and errors.

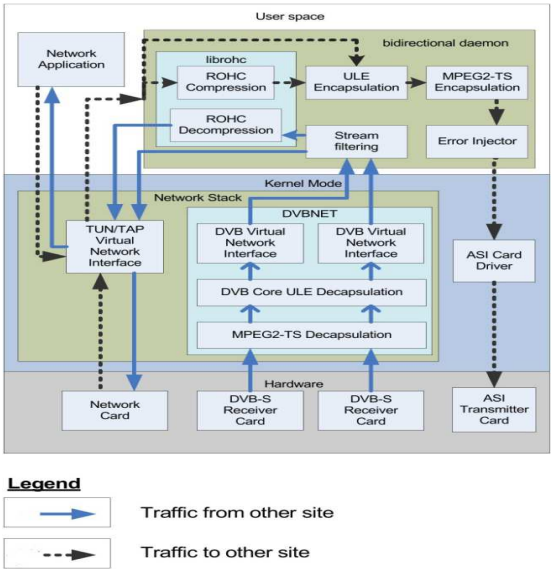

Figure 2: Detailed Software Components And Its Interaction With Traffic.

2. MOTIVATION

Using satellite communication systems has several drawbacks such as long propagation delay, cost to launch a satellite into space, expensive satellite communication equipment, and recurring costs of leasing bandwidth from the satellite-communication provider. For a single hop between earth stations (via geostationary earth orbit, GEO), the long propagation delay of GEO satellites is approximately 250 ms. The 500 ms round-trip time (RTT) delay makes it unsuitable for most interactive applications. Transport protocols like TCP rely on acknowledgment for flow control hence, the performance of TCP suffers when deployed in satellite networks. Although various techniques like TCP Hybla [8] have been proposed to solve this issue.

Multi-Protocol Encapsulation (MPE) is a standard proposed by [2] to carry network data over MPEG-2 TS frames, and was optimized to transport IPv4 packets. No payload type field is present in the MPE header that also carries the destination MAC address making the MPE format complicated by introducing a significant amount of overhead for small payloads. Unidirectional Lightweight Encapsulation (ULE) [9] is a standard put forth by the IP-over-DVB working group of the IETF to encapsulate network data over MPEG2-TS frames. Several studies have evaluated the performance characteristics of ULE [10] and compared them with those of MPE [11]–[13]. The results of these studies showed that ULE is the more efficient encapsulation format because the overhead incurred by ULE is less than that incurred by MPE.

DVB-S2 [14] is the second-generation DVB

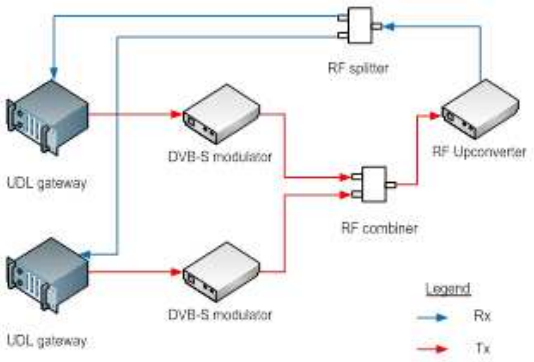

Figure 3: Configuration Of DVB-S Test-Bed

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

standard for satellite communication. DVB-S only supports QPSK modulation that translates to only 2 bits per symbol, whereas DVB-S2 allows for four types of modulation, namely, QPSK, 8PSK, 16APSK, and 32APSK. The most efficient modulation, 32APSK, is capable of carrying 5 bits per symbol. In addition, the DVB-S2 system also employs an Adaptive Coding and Modulation technique to improve bandwidth utilization through which a receiver will send feedback on the condition of the link. Based on the feedback, the feed will adjust the coding and modulation type to maximize the bandwidth utilization. The improvements introduced into DVB-S2 give it a 30% performance gain over DVB-S [15]. Instead of using MPEG2-TS frames to deliver data, DVB-S2 uses BaseBand Frame (BBFrame). To ensure backward compatibility with the old system, a MPEG2-TS frame can be encapsulated within a BBFrame, thus allowing MPE and ULE to be used for DVB-S2. However, this approach is not optimal because an additional layer of encapsulation is required. Thus, Generic Stream Encapsulation (GSE) [16] was introduced to reduce the overhead. A study was conducted to compare the efficiency of MPE, ULE, and GSE encapsulation over DVB-S2, and the results showed that GSE is the most efficient encapsulation [17].

Before data can be transferred through a network, several layers of encapsulation may have to be applied. At the end of this process, the data, which are part of the payload, are combined with the headers forming an IP packet. ROHC, which is standardized by the ROHC Working Group (ROHC WG) of the IETF [7], [18], is a header compression framework designed to work with error-prone links with long delay. The design assumes that the underlying link carrying the compressed packets does not reorder packets, whereas the reordering in the pre-HC link is acceptable. ROHCv2 [19] was proposed to improve and address the deficiency in the previous version. Unidirectional mode is typically used for unidirectional links where the decompressor is unable to send feedback to the compressor, or in multicast session where multiple negative acknowledgments from multiple decompressors will lead to an inefficient compression state [20]. The bidirectional reliable mode is the most robust mode that enforces tightest coupling between compressor and decompressor. A compressor cannot transit to a higher state unless an acknowledgment is received from the decompressor. This policy of bidirectional reliable mode renders it unfit for a link with long delay.

3. PROPOSED SYSTEM ARCHITECTURE

The present research is a continuation of the UDL (Uni-Directional Link) mesh network project. Figure 1 depicts a UDL mesh network with three sites. Each site consists of a UDL gateway that sits between the satellite link and the local area network (LAN), UDL gateway can be configured as a bridge or a router. Figure 2 depicts a more detailed picture of the software components on a UDL gateway. The daemon process, called bidirectional, was implemented to include the capability to compress packets using ROHC, as well as to decompress ROHC compressed packets using an external library called librohc. Decapsulation of MPEG2-TS frames and ULE SNDU is performed by the dvb_net subsystem. Each logical channel of MPEG2-TS stream, as indicated by PID in the MPEG2-TS frame header, is represented as a DVB virtual network interface. A more detailed

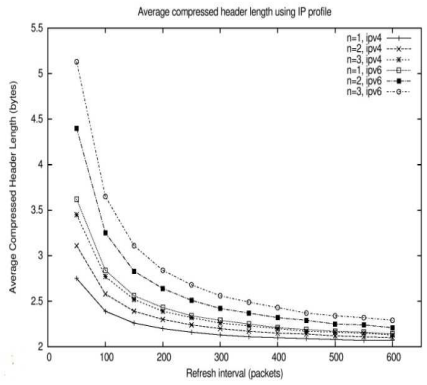

Figure 4: Average Compressed Header Length Using IP Profile For Ipv4 And Ipv6 Traffic

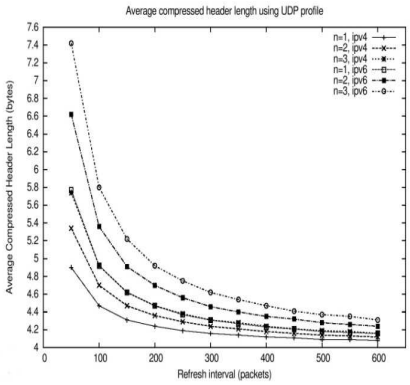

Figure 5: Average Compressed Header Length Using UDP Profile For UDP/Ipv4 And UDP/Ipv6 Traffic

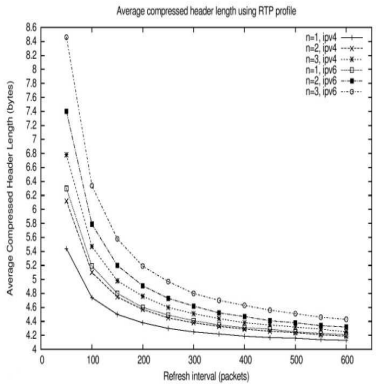

Figure 6: Average Compressed Header Length RTP/UDP/Ipv4 And RTP/UDP/Ipv6 Traffic

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

explanation of the implementation of the dvb_net subsystem is provided in Section II of [21]. Stream filtering will be applied to the incoming traffic. Compressed traffic will be passed to the decompressor before being sent to the tun/tap virtual network interface, whereas uncompressed streams will be channeled directly to the tun/tap virtual network interface. Figure 2 shows the overall process and its interaction with traffic.

For the transmission of MPEG2-TS streams to DVB-S modulator, an ASI card from Linear System was used for the experiment. This ASI card model has a peculiar behavior wherein transmission will only occur when half of its total buffers is filled. To minimize delay in the system, at least two buffer queues were used. For each queue, the minimal buffer size is 6 MPEG2-TS frames. Therefore, to effectively transmit a MPEG2-TS frame, a minimum of 6 MPEG2-TS frames has to be filled. If there is no incoming packet to fill the rest of 5 MPEG-TS frames, 5 NULL MPEG2-TS frames have to been sent by the bidirectional daemon from time to time to avoid stalling the first MPEG2-TS frame forever.

4. EXPERIMENTAL DESIGN

The research emphasizes two types of streaming tests: UDP and RTP. UDP streaming tests were used to measure the impact of header compression over UDP packets of different payload sizes using IP and UDP compression profiles. RTP streaming tests were used to measure the number of sustainable compressed RTP streams using IP, UDP, and RTP compression profiles. These streaming tests were conducted over unidirectional links with various degrees of bit error to measure the impact of bit errors on header compression. The configuration in Figure 3 was the DVB-S testbed used for this work. In the present research, the rtpfaker was developed because existing benchmark tools like D-ITG only generate version 1 RTP headers, whereas the RTP profile of ROHC only works on RTP version 2. For each RTP test, each RTP stream is an emulation of a GSM-encoded stream with payload of 33 bytes sent every 20 milliseconds at Constant Bit-Rate (CBR). The estimated jitter for an RTP stream can be calculated as [22].

$$Transitcurrent = TSarrival - TSRTP \quad (1)$$

$$Difftransit = |Transitcurrent - Transitprev| \quad (2)$$

$$Jittercurrent = Jitterprev + 1/16\,(Transitcurrent - Jitterprev) \quad (3)$$

The approach presented in Equations 1, 2, and 3 is suitable for estimating the interarrival time for the live streaming of RTP packets where TSarrival is received RTP packet timestamp, TSRTP RTP header timestamp, Transitcurrent denotes time taken for the source to deliver the current RTP packet to its destination and Current jitter is denoted by Jittercurrent approximated by using Equation 3. The previous jitter value is denoted by Jitterprev. The difference between Transitcurrent and Jitterprev is divided by 16 to limit the impact of the drastic change in the transit time on the computation of the jitter. To calculate the average interarrival time of RTP packets within the RTP stream with n number of RTP packets, the following equations were used:

$$Average_{RTP} = \frac{1}{n-1}\sum_{i=2}^{n} T_i - T_{i-1}$$

Let Ti be the arrival time of the RTP packet i. The jitter of interarrival time within the RTP stream is calculated by applying standard deviation onto the interarrival time using the following equation:

$$Jitter_{RTP} = \sqrt{\frac{1}{n-1}\sum_{i=2}^{n}((T_i - T_{i-1}) - Average_{RTP})^2}$$

The rtpsend is capable of sending more than 1 RTP stream simultaneously. Showing the results for each RTP stream is impractical; therefore, the averaged results from every RTP stream will be presented. Thus, for a test that carries m number of RTP streams, the overall average interarrival time is given below:

TABLE I
AVERAGE INTER-ARRIVAL TIME & JITTER OF RTP STREAM OVER ETHERLINK

Type of Test	Average Inter-arrival Time (ms)	Jitter
1 uncompressed IPv4/UDP/RTP stream	20.00	0.03
1 uncompressed IPv6/UDP/RTP stream	20.00	0.03
311 streams of IPv4/UDP/RTP	20.00	0.06
314 streams of IPv6/UDP/RTP stream	20.00	0.06

$$Average_{overall} = \frac{1}{m}\sum_{i=1}^{m} Average_{RTPi}$$

AverageRTPi represents average interarrival time of RTP stream i out of m number of RTP streams. Similarly, average jitter will be represented by the following formula:

$$Jitter_{overall} = \frac{1}{m}\sum_{i=1}^{m} Jitter_{RTPi}$$

Journal of Theoretical and Applied Information Technology
20th May 2016. Vol.87. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

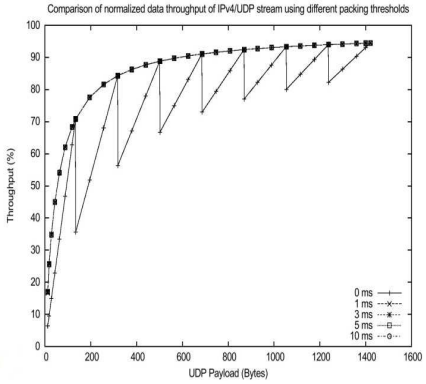

Figure 7: Normalized Data Throughput Of UDP/Ipv4 Stream With Different Payload Sizes Over Various Thresholds

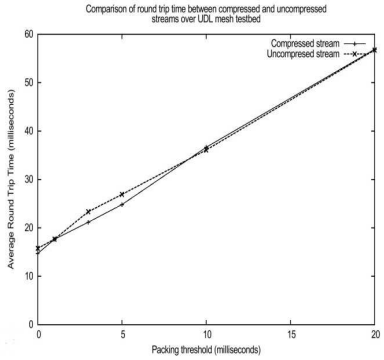

Fig. 8. Comparison Of Average Round Trip Time Between Compressed And Uncompressed Streams Over UDL Mesh Testbed

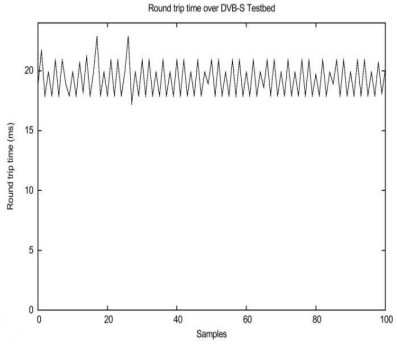

Fig. 9. Measured Transmission And Processing Round Trip Time Over UDL Mesh Testbed (Excluding Propagation Delay Over Geostationary Satellite Link)

The JitterRTPi denotes the jitter of interarrival time for RTP stream i out of m number of RTP streams. The rtpfaker was built specifically to

generate RTP streams with relatively high precision of interarrival time and low interarrival jitter. Table 1 shows the performance characteristics of the RTP

benchmark tool over a 100 Mbps Ethernet link connected using 2 meters of UTP cable. Even with 314 RTP streams, the jitter is still relatively low.

5. RESULT AND DISCUSSION

This section discusses performance characteristics of ROHC over a DVB-S testbed. The selection of parameters for the experiment was first determined. Using the predetermined parameters, subsequent tests were conducted, and the results were gathered and analyzed.

5.1. Average length of compressed packet header

The test results presented in this section were conducted using the ROHC unidirectional mode, where the maximum number of non-IR packets before transiting back to the IR state is refresh interval. Figures 4, 5, and 6 depict the average length of the compressed header using different compression profiles operating under various operational parameters. Using a shorter refresh interval led to a larger average length for compressed headers across all profiles, especially in the case of IPv6 traffic, because the static chain of IPv6 IR packet contains 32 bytes of IPv6 addresses. As refresh interval became longer, the average length of compressed headers decreased as well. Moreover, when refresh interval exceeded 350 packets, the rate of reduction in the average length of compressed headers diminished, along with the savings achieved using smaller values for n.

5.2. Packing Threshold

Regarding the packing thresholds selection on throughput and latency, two modes are available to encapsulate a ULE SNDU into MPEG2-TS frames padding mode and packing mode. Although operating in padding mode yielded better results in terms of latency, the process is not optimal in terms of bandwidth usage because some amount of bandwidth will be used by padding bytes. Padding mode is most inefficient when only 1 byte of a

Journal of Theoretical and Applied Information Technology
20th May 2016. Vol.87. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

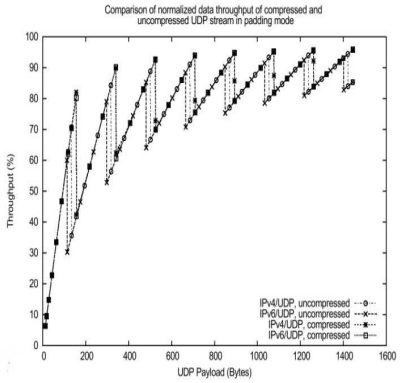

Fig. 10. Comparison Of Normalized Data Throughput Of Compressed And Uncompressed UDP Streams With Different Payload Sizes In Padding Mode

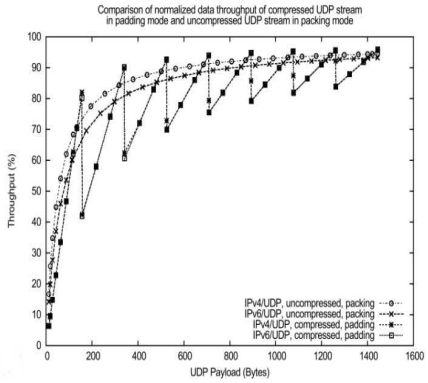

Fig. 11. Comparison Of Normalized Data Throughput Of Compressed UDP Stream In Padding Mode And Uncompressed UDP Stream In Packing Mode

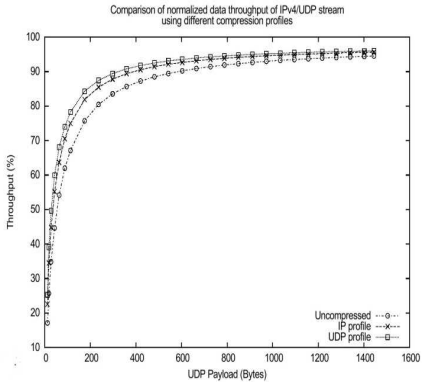

Fig. 12. Comparison Of Normalized Data Throughput Of Compressed Ipv4/UDP Streams

MPEG2-TS frame contains useful data, thus leaving 183 bytes of the payload unused. Knowing that 1 byte is equivalent to 8 bits, the maximum number of bits that has to be padded, Maxpadding, can be given as:

$$\text{Maxpadding} = 183 \times 8 \quad (8)$$

For a link operating at full capacity, the time taken to receive next 183 bytes is determined by the capacity of the link itself. This metric was used in the selection of packing threshold for this experiment. Bit rate is defined by the number of bits that can be delivered per second [23], and can be summarized using the following equation:

$$Bitrate = \frac{bit}{second} \quad (9)$$

Thus, using Equation 7, for a link with bit rate of BW, the maximum packing threshold in second, Packing, can be summarized as:

$$BW = \frac{Max_{padding}}{Packing} \quad (10)$$

By replacing Equation 7, the packing threshold, Packing, can be redefined as:

$$Packing = \frac{183 \times 8}{BW} \quad (11)$$

Using the formula above, the packing threshold for the 8 Mbps link is 0.183 milliseconds. However, the ULE encapsulator used in this experiment does not handle packing threshold in sub-millisecond. Thus, smallest allowable packing threshold is 1 millisecond. To ensure that, in practice, using 1 millisecond is not susceptible to the saw-tooth effect of padding mode, a series of throughput tests was conducted using different packing thresholds, as shown in Figure 7. The results shown in Figure 9 were measured using uncompressed IPv4/UDP streams. The following formula can be used to deduce the UDP payload size, where the nth valley is formed after the first spike.

PayloadSizen = 183 – OverheadULE – HeaderEthernet - HeaderIP – HeaderUDP + 1 + 184 x (n -1) (12)

where OverheadULE is the overhead introduced by ULE, whereas HeaderEthernet, HeaderIP, and HeaderUDP represent the header size of Ethernet, IP, and UDP, respectively.

Journal of Theoretical and Applied Information Technology
20th May 2016. Vol.87. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

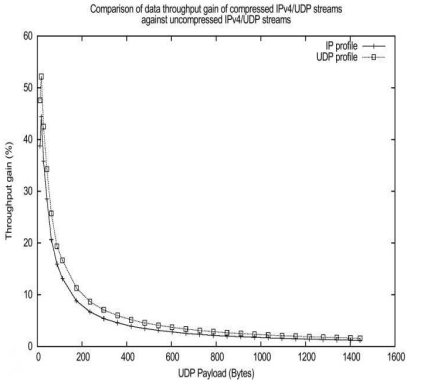

Fig. 13. Comparison of data throughput gain of compressed and uncompressed IPv4/UDP at different UDP payload sizes

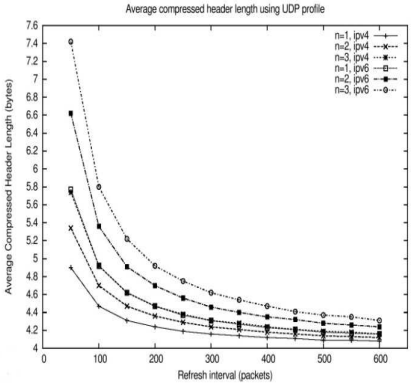

Fig. 14. Comparison of normalized data throughput of compressed and uncompressed IPv6/UDP streams at different UDP payload

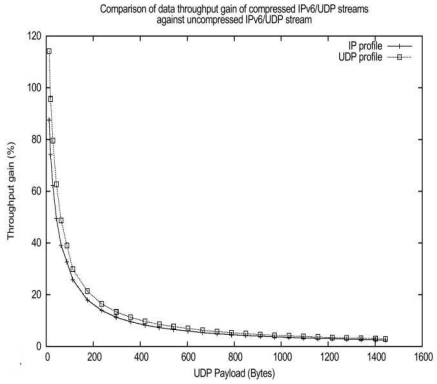

Fig. 15. Comparison of data throughput gain of compressed and uncompressed IPv6/UDP streams at different UDP payload

Figure 8 shows the effect of various packing thresholds over latency. Increasing packing threshold led to a linear increase of average round trip time. The processing overhead of the compressed stream did not introduce any significant latency. The observed jitter was caused by buffering in the device driver of the DVB-S receiver card. Figure 9 shows jitter in round trip time measured using ICMP message over DVB-S testbed using 1 millisecond packing threshold. Increasing the buffer size in the device driver of the DVB-S receiver card led to higher jitter than what is shown in Figure 9. Figure 10 shows the normalized data throughput of UDP streams with different UDP payload sizes in padding mode. The saw-tooth shape was formed for all type of streams. For uncompressed IPv6/UDP stream, the valleys were formed earlier than uncompressed IPv4/UDP stream because the IPv6 header is 20 bytes larger than the IPv4 header. In case of UDP streams compressed using UDP profile, the differences of data throughput of IPv4 and IPv6 UDP streams were not significant compared with the average compressed header lengths that were almost the same. For uncompressed streams, the payload sizes where the valleys were formed could be predicted accurately because the header size was consistent. The UDP payload sizes where the n-th valley is formed after the first spike can be predicted using the following formula:

$$PayloadSizen = 183 - OverheadULE - HeaderEthernet - AverageCompressed + 1 + 184 \times (n-1) \quad (13)$$

AverageCompressed is average size of compressed headers similar to the average lengths of compressed header are the floating point value, as shown in Figure 10. Figure 11 offers a comparison data throughput of compressed UDP streams in padding against uncompressed UDP streams in packing mode.

5.3. Ideal (error-free) links

5.3.1. UDP Tests

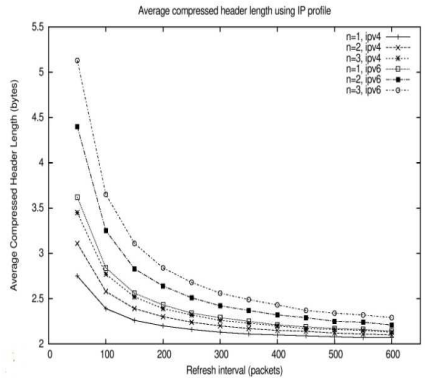

Fig. 16. The Number Of Parallel RTP Streams Sustainable Over 8Mbps DVB-S Link Using Different Compression Profile

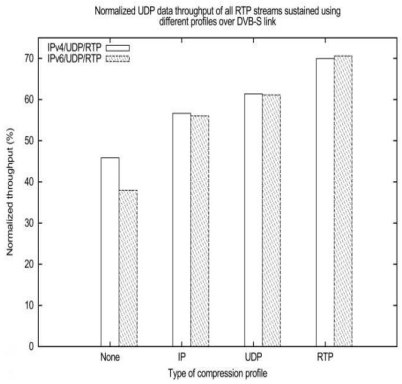

Fig. 17. Comparison Of Normalized Aggregated UDP Data Throughput For Maximum Number Of RTP Streams Supported By Different Compression Profiles

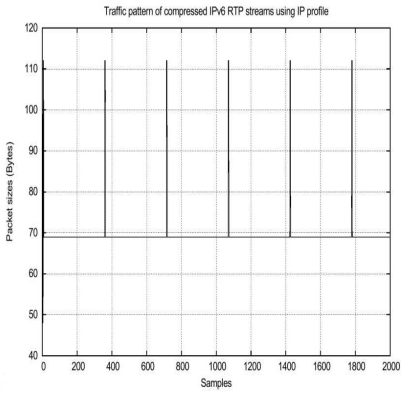

Fig. 18. Traffic Pattern Of Compressed RTP Streams Using IP Profile

Journal of Theoretical and Applied Information Technology
20th May 2016. Vol.87. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

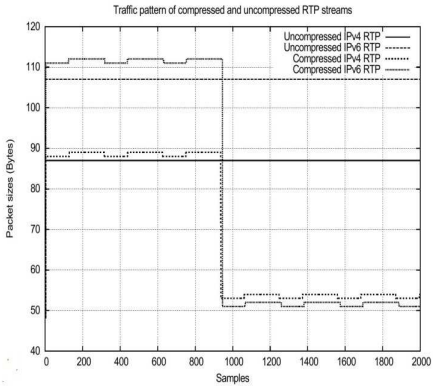

Fig. 19. Traffic Pattern Of Compressed And Uncompressed RTP Streams

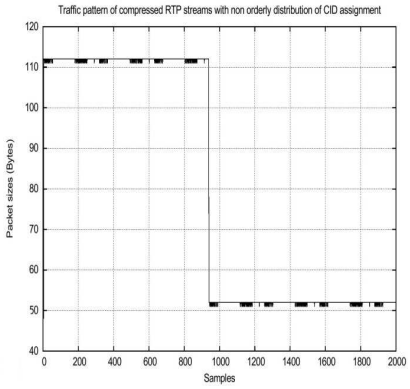

Fig. 20. Traffic Pattern Of Compressed RTP Streams With Non-Sequential Distribution Of CID Assignment

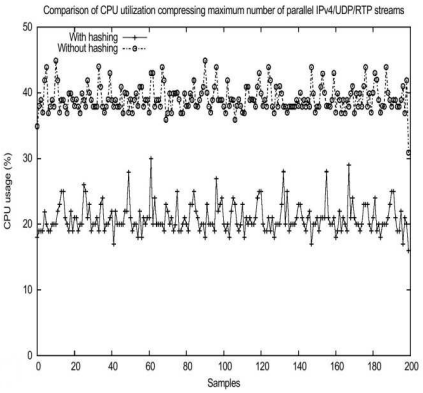

Fig. 21. Comparison Of CPU Utilization Compressing Maximum Number Of Ipv4/UDP/RTP Streams

The UDP data throughput tests presented in this section were conducted using iperf [24], whereas throughput gain was computed by deducting the throughput of compressed stream with the throughput of uncompressed stream, and then dividing the resultant difference with the throughput of the uncompressed stream. Figure 12 shows the comparison of the data throughput of compressed IPv4/UDP streams using IP and UDP profile against the data throughput of uncompressed IPv4/UDP stream at different UDP payload sizes.

In terms of the UDP data throughput, the gap between the compressed UDP stream and the uncompressed UDP stream was most significant when the UDP payload sizes were in the range of

200 bytes. However, in terms of percentage of throughput gain against the uncompressed UDP stream, the advantage was most obvious for small UDP payload, as shown in Figure 13.

Figure 14 shows the comparison of the data throughput of compressed IPv6/UDP streams using IP and UDP profile against the data throughput of uncompressed IPv6/UDP streams at different UDP payload sizes. Comparing Figure 12, the significance of header compression for IPv6 streams was more obvious due to the larger overhead of IPv6 streams. Moreover, Figure 15 shows that the throughput gain in terms of percentage for IPv6 UDP stream with 12 bytes of UDP payload was more than 100 percent when compressed using UDP profile. The effect of packet drops was negligible on compressed IPv6 streams because none of the fields within the IPv6 header depends on the correct interpretation of Sequence Number in the compressed header.

5.3.2. RTP Tests

Figure 16 shows the maximum number of parallel RTP streams sustainable over 8Mbps DVBS link using different compression profiles. The performance gap between IPv4 and IPv6 streams became marginal when the header compression was applied, because the difference in the average length of the compressed headers for IPv4 and IPv6 streams was only marginal. Using IP and UDP compression profile, the IPv4 RTP streams managed to slightly outperform the IPv6 RTP streams.

Given that each RTP packet is transmitted every 20 milliseconds, in the span of 1 second, there are 50 RTP packets within the RTP stream. Therefore, the number of bits used by a compressed RTP stream, bitsRTP, within a second, can be estimated using the formula given below:

$$BitsRTP = (8 + 14 + AverageCompressed + Payload) \times 50 \times 8 \quad (14)$$

AverageCompressed is the average length of compressed headers, and Payload is the size of payload in octet after the compressed header. The Ethernet header introduces 14 bytes of overhead, whereas ULE adds another 8 bytes. Assuming that every MPEG2-TS frame is shared by more than one compressed RTP packet, out of 188 bytes of MPEG2-TS frame, only 183 bytes can be used to carry compressed RTP packets, because 4 octets are used by MPEG2-TS frame header, and an additional octet is used by Payload Pointer. Given the number of bits used by the RTP stream, the maximum number of parallel RTP streams, numRTP, supported by a DVB-S link with the capacity of BW bps, can be summarized using the following formula:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$num_{RTP} = \frac{BW}{bits_{RTP}} \times \frac{183}{188}$$

Figures 16 and 17 show the normalized aggregated UDP data throughput for RTP streams compressed using different compression profiles. The IPv6 RTP streams compressed using the RTP profile managed to sustain slightly better data throughput than the IPv4 RTP streams compressed using the RTP profile. Using Equation 14, a comparison of the estimated maximum number of parallel compressed RTP streams that can be sustained by an 8 Mbps DVB-S link against observed results is given in Table 2 in percentage form. Table 3 shows average interarrival time and jitter of RTP streams as observed on the DVB-S testbed. With 1 RTP stream, with or without header compression, the jitter was significantly high due to the inherent jitter within the DVB-S testbed, as shown in Figure 9. However, when the bandwidth of the link was fully utilized, the jitter was lower than the jitter when the link was idle, except for the IPv6 RTP stream compressed with the RTP profile.

Figure 18 shows the traffic pattern of the first 2,000 samples of packets captured during the streaming of 249 parallel streams of IPv6 RTP streams compressed with IP profile. Compared with the traffic pattern shown in Figure 19, where more than 300 parallel streams of RTP streams were compressed using the RTP profiles; the short spikes showed up occasionally when the context on the compressor transits back to the IR state. All RTP streams share identical IP headers; thus, only one context was used to compress all RTP streams when the IP profile was used. Likewise, when the UDP profile was used to compress RTP streams, only one context was required for all RTP streams because all RTP streams shared similar UDP headers. The reason was the design limitation of the rtpfaker. The rtpfaker generates multiple RTP streams by assigning a different Synchronization Source (SSRC) to different RTP streams while maintaining the same IP addresses and UDP port number for all streams. This limitation is not unrealistic in the real network environment, where many nodes may share a single global IP address through Network Address Translation (NAT). For this type of network environment, the IPv4 traffic compressed using the IP profile will exhibit a similar pattern, as shown in Figure 18.

The RTP profile assigned a context to each individual RTP stream that forms a continuous burst of spikes in Figure 19 as a result of 300 contexts that need to send IR packets at the same time. The high jitter might be caused by the huge burst of IR packets causing temporary spikes in the traffic pattern exceeding the bandwidth of the link. The jitter might be reduced if each context did not transit to IR states in unison. The reason is that each context sent 3 IR packets, and therefore, for 314 parallel IPv6 RTP streams, 942 packets were sent in a burst that exceeds the bandwidth capacity.

Figure 19 shows a square-wave shape for the traffic pattern of RTP streams because the Context Identifier (CIDs)on the compressor were mostly unused, and thus can be assigned sequentially for each new context. Figure 20 projects a different picture when all available CIDs have been used. When all the available CIDs have been used, the compressor will need to reuse old CIDs from the existing idle contexts. Under this circumstance, the assignment of CIDs will not be ordered sequentially, and the square-wave shape is not obvious unless the figure is enlarged. Through observation, the DVB-S testbed manages to deliver to a few more parallel RTP streams in the case of Figure 19. The reason is that all CIDs with a smaller octet 87 size (0 to 127) could be assigned to RTP streams leading to less bandwidth requirement. In the case of Figure 20, only some of the 1 octet CIDs were reused for the active RTP streams. However, the traffic pattern of Figure 21 should be more relevant for any long-running system. In summary, the performance of the compressed IPv4 and IPv6 RTP streams should be very similar regardless of the profiles used, provided that the right refresh interval is chosen. In practice, the compressed IPv4 stream may show a slight disadvantage due to its dependence on the IP-ID field.

Another issue faced in compression using the RTP profile was the CPU processing overhead. For RTP profile, the compression of 311 IPv4 RTP streams led to 40% of CPU usage on a Pentium 4 2.8 GHz machine, as shown in Figure 21. Using an approach where contexts were hashed using information generated from the static context data, the CPU usage for compression was reduced to 20% for both IPv4 and IPv6 RTP streams.

6. CONCLUSION

Given the empirical results observed on the DVB-S testbed, applying RObust Header Compression over ULE stream delivered significantly better performance than the normal ULE stream. On a satellite link with BER as high as 10-4, ROHC compressed ULE stream managed to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

deliver better results than uncompressed stream under most circumstances. The main contributing factor to context damage was caused by a burst of packet losses. Through observation, the current CRC correction scheme suggested by the ROHC is insufficient to attempt the right correction due to unforeseen practical limitations when the experiment was first carried out. However, the results also showed that a reasonable CPU resource was required to process header compression and decompression. A potential solution to lower the CPU resource requirement for an underpowered machine is to make the ROHC compressor selectively perform the header compression only on smaller packets. The reason is that, for a bigger packet, the ratio of header overhead is smaller compared with a smaller packet.

REFRENCES:

[1] E. Duros, W. Dabbous, H. Izumiyama, N. Fujii, and Y. Zhang, "A link-layer tunneling mechanism for unidirectional links," *RFC3077*, vol. 3, 2001.

[2] E. ETSI, "301 192:" Digital Video Broadcasting (DVB); DVB specification for data broadcasting," *European Standard*, vol. 1, no. 1, 2000.

[3] V. Jacobson, "Compressing TCP/IP headers for low-speed serial links," 1990.

[4] S. J. Perkins and M. W. Mutka, "Dependency removal for transport protocol header compression over noisy channels," in *Communications, 1997. ICC'97 Montreal, Towards the Knowledge Millennium. 1997 IEEE International Conference on*, 1997, pp. 1025–1029.

[5] M. Degermark, B. Nordgren, and S. Pink, "IP Header Compression, Request for Comments 2507," 1999.

[6] S. L. Casner and V. Jacobson, "Compressing IP/UDP/RTP headers for low-speed serial links," *IETF*, 1999.

[7] C. Bormann and M. Degermark, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed," 2001.

[8] C. Caini and R. Firrincieli, "TCP Hybla: a TCP enhancement for heterogeneous networks," *International journal of satellite communications and networking*, vol. 22, no. 5, pp. 547–566, 2004.

[9] G. Fairhurst and B. Collini-Nocker, "Unidirectional lightweight encapsulation (ULE) for transmission of IP datagrams over an MPEG-2 transport stream (TS)," 2005.

[10] M. Sooriyabandara, G. Fairhurst, A. Ang, B. Collini-Nocker, H. Linder, and W. Stering, "A lightweight encapsulation protocol for IP over MPEG-2 networks: design, implementation and analysis," *Computer Networks*, vol. 48, no. 1, pp. 5–19, 2005.

[11] T. C. Hong, W. T. Chee, and R. Budiarto, "A comparison of IP datagrams transmission using MPE and ULE over Mpeg-2/DVB networks," in *Information, Communications and Signal Processing, 2005 Fifth International Conference on*, 2005, pp. 1173–1177.

[12] T. C. Hong, W. T. Chee, and R. Budiarto, "Simulation and Design of IP over DVB using Multi-Protocol Encapsulation and Ultra Lightweight Encapsulation," in *Proc. National Computer Science Postgraduate Colloquium*, 2005.

[13] G. Xilouris, G. Gardikis, H. Koumaras, and A. Kourtis, "Unidirectional lightweight encapsulation: performance evaluation and application perspectives," *Broadcasting, IEEE Transactions on*, vol. 52, no. 3, pp. 374–380, 2006.

[14] E. ETSI, "Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications," 2005.

[15] A. Morello and V. Mignone, "DVB-S2-Ready for lift off," *EBU Technical Review*, vol. 10, pp. 1–10, 2004.

[16] T. ETSI, "102 606 V1. 1.1 Digital Video Broadcasting (DVB)," *Generic Stream Encapsulation (GSE) Protocol*, 2007.

[17] A. Mayer, B. Collini-Nocker, F. Vieira, J. Lei, and M. Castro, "Analytical and experimental IP encapsulation efficiency comparison of GSE, MPE, and ULE over DVB-S2," in *Satellite and Space Communications, 2007. IWSSC'07. International Workshop on*, 2007.

[18] L. Jonsson, G. Pelletier, and K. Sandlund, "RFC 4995: The Robust Header Compression (ROHC) Framework," *IETE*, 2007.

[19] G. Pelletier and K. Sandlund, "Robust header compression version 2 (ROHCv2): profiles for RTP, UDP, IP, ESP and UDP lite," *IETE*, 2008.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[20] E. Martinez, A. Minaburo, and L. Toutain, "ROHC for multicast distribution services," in *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*, 2005, pp. 1540–1544.

[21] C. Prahauser and B. Collini-Nocker, "A second-generation architecture for Linux DVB networking," *International Journal of Satellite Communications and Networking*, vol. 28, no. 5–6, pp. 369–381, 2010.

[22] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," 2003.

[23] B. Forouzan, C. Coombs, and S. C. Fegan, *Introduction to data communications and networking*. McGraw-Hill, Inc., 1997.

[24] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf: The TCP/UDP bandwidth measurement tool." [Online]. Available: http://sourceforge.net/projects/iperf/. [Accessed: 13-Dec-2015].