Investigating a Reduced Size Real-Time Transport Protocol for Low-Bandwidth Networks

Josephine N. Kakande[1], Keith L. Ferguson[1], Mqhele E. Dlodlo[2] and Gerhard de Jager[2]
Council for Scientific and Industrial Research[1]
P. O. Box 395, Pretoria 0001, South Africa
Tel: +27 12 8413028
and Department of Electrical Engineering
University of Cape Town[2]
email: {nkakande, kferguson}@csir.co.za[1]; {Mqhele.Dlodlo, Gerhard.DeJager}@uct.ac.za [2]

Abstract-Optimization of bandwidth usage for video streaming is of paramount importance in networks where low bitrate links are typical. Among the solutions proposed to address this problem is header compression. Real-Time Transport Protocol (RTP) and RTP Control Protocol (RTCP) are the major protocols responsible for the delivery of real-time media. Previously, header compression at the hardware layer and multiplexing of media frames into single RTP packets was considered sufficient for reducing the bit overhead of the RTP packets. However, at the very low bit rates encountered in congested and low throughput networks, multiplexing and hardware compression do not suffice for end-to-end delivery and therefore the use of a lightweight version of RTP, defined in this work as RTP-Lite, requires investigation. A cyclical approach to compression of the RTP headers was used with different compression cycle patterns for Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) transport. Measurements over the public Internet showed that end-to-end compression of the RTP header at the application layer was achieved with the expected reduction in required throughput and minimal degradation of packet loss and jitter performance.

Index Terms—End-to-end, header compression, low-bandwidth, multimedia streaming, RTP

I. INTRODUCTION

Multimedia applications today comprise one of the key classes of Internet traffic. However, the Internet is a best-effort transmission platform with no quality of service (QoS) guarantees, which creates several challenges for real-time multimedia streaming. The major hurdles are fluctuating throughput and other QoS-related issues including packet loss, jitter and delay.

In the developing world, limited-capacity Internet infrastructure results in links that exhibit low end-to-end bit rates due to network congestion. Effective rates as low as 5 kbps have been measured [1]. Wireless technologies that are becoming a more and more prevalent means of Internet connectivity possess capacity constraints relative to fixed line infrastructure. This further enhances the requirement for protocols geared towards supporting low throughput capacities. One approach to tackle this problem has been compression of the packet headers for a media flow so as to achieve rate savings and alleviate the streaming problems encountered on low bit rate links.

The principal protocol used for real-time media streaming is the Real-Time Transport Protocol (RTP) [2]. RTP operates at the application layer, and provides end-to-end delivery services for supporting real-time applications including interactive audio and video. Real-Time Control Protocol (RTCP) [2] adds QoS functionality to RTP, furnishing packet loss and jitter or time arrival statistics. RTP/RTCP operate above the connectionless and best-effort User Datagram Protocol (UDP) or the connection-oriented, reliable Transmission Control Protocol (TCP) and the Internet Protocol (IP).

Header compression is the process of reducing protocol header overhead, while still maintaining end-to-end transparency [3]. Previously, header compression at the hardware layer and multiplexing of media frames into single RTP packets was considered sufficient for reducing the bit overhead of the RTP packets. By increasing the number of frames in an RTP packet the header overhead is reduced but this increases the susceptibility to errors and packetization delay [4]. At the very low bit rates encountered in congested and low bandwidth networks however, multiplexing and hardware compression do not suffice. Therefore the use of a smaller size RTP header protocol at the application layer, hereafter referred to as RTP-Lite, is investigated in this work. The compression efficient RObust Header Compression (ROHC) [3] scheme is used to provide a benchmark for evaluating the performance of the proposed RTP-Lite protocol. The network performance in terms of throughput, jitter and loss relative to standard RTP and ROHC are investigated and the improvements obtained by using RTP-Lite identified.

The outline of the rest of the paper is as described below. Section II contains an overview of header compression. In section III, the proposed RTP-Lite frame work is presented. In section IV the experimental setup and tools are described, followed by results analysis in section V. Section VI contains the conclusions and recommendations for future work.

II. Header Compression Overview

Header compression typically involves a compressor transmitting data to a receiving node equipped with a decompressor. Header compression is facilitated by the fact that header fields tend to contain redundant information [5, 6]. Two classes of header redundancy have been identified as intra-packet redundancy, whereby different headers within a packet contain information that can be inferred from other headers contained in the same packet and inter-packet redundancy, whereby there are no or minimal additive differences between corresponding header field values for successive packets [6]. As such some fields may be omitted completely from the compressed header once their information has been stored at the decompressor, or only the changes (delta parameters) in some predictably varying fields are conveyed. For successful decompression both sender and receiver need to be aware of and store some shared flow-specific information referred to as the context and to remain synchronized throughout the session [3, 7].

Compression efficiency, expressed in terms of compression gain as given in Equation 1, refers to how much throughput is saved by a compression scheme.

$$\text{Compressio n Gain (\%)} = \frac{\text{Uncompress ed header size - Compressed header size}}{\text{Uncompress ed header size}} * 100 \qquad (1)$$

Several header compression implementations have been proposed over the years. The Van Jacobson Protocol was designed for compression of TCP/IP datagrams [8]. It reduced the 40 byte TCP/IP header to as low as 3 bytes and was designed for low-speed serial links of 0.3 kbps to 19.2 kbps.

Later, an IP/UDP/RTP compression scheme called Compressed RTP (CRTP) was devised [9] with the goal of reducing the RTP header size for transmission of video and audio traffic over limited bit rate links (14.4 kbps and 28.8 kbps) with low round-trip times. CRTP was found to operate poorly on unreliable and long round-trip time (RTT) links leading to inefficient bandwidth usage as packets arriving within an RTT after loss of context could not be properly decompressed. Furthermore it was necessary to send large context-restoration headers to reinstate the shared state of header information between the compressor and decompressor.

To cater for long RTTs, packet loss and reordering, Enhanced CRTP (ECRTP) was designed to provide repeated updates by the compressor of the context state stored at the decompressor [7]. These updates included not only the delta parameters but also some full parameters of the context. While ECRTP was less compression efficient than CRTP, it provided better performance over unreliable links with large delays.

ROHC was developed as a link-by-link robust compression scheme to handle the high bit error rates and long delays associated with wireless links [3]. Robustness in this context refers to the ability of the scheme to handle packet losses and errors. ROHC, which has a relatively high compression efficiency and is suited for varying link technologies, compresses the RTP/UDP/IP packet header length to 2 to 4 bytes. It operates at ‘layer 2.5’ above the link layer but below the IP layer.

III. Proposed RTP-Lite Framework

To derive the RTP-Lite compression scheme the RTP header fields were categorised according to how often their values changed and how important it was that they be conveyed in the header of a packet. Three classes were identified as listed below:

- Always (A): For fields that always need to be sent.
- Rarely (R): For fields which need only be sent infrequently because after initial transmission they rarely change.
- Initially (I): For fields that need only be transmitted once either because of redundancy or because they are known beforehand.

The proposed protocol incorporates the use of standard RTP full-header packets that provide a reference for decompressing the ensuing compressed header packets.

UDP is the primary transport protocol used with RTP on account of its features such as low delay and support for multicasting (one-to-many transmission) that is suited to broadcasting. The nature of real-time data requires the low delay that is provided by UDP but at the cost of unreliable delivery. TCP is unsuited to time-critical applications due to its flow control, error correction and retransmission characteristics. However, TCP guarantees packet delivery and therefore packet loss or reordering do not need to be considered for RTP-based applications that can tolerate the delay associated with TCP. Furthermore, TCP traffic is able to traverse firewalls. Table 1 outlines the RTP header fields and their classification for the proposed RTP-Lite compression over both UDP (U) and TCP (T) transport. The fixed RTP header is of 12 bytes.

A. *RTP-Lite over UDP*

The fields that are excluded from the compressed RTP-Lite header for UDP transport are described below:

- Padding, Extension, SSRC: Usually remain the same for all session packets and hence are omitted from the RTP-Lite headers.
- CSRC Count, CSRC: Assumed to be unchanging [1, 3] and that there is no mixer along the end-to-end path.
- Payload Type: Some coders compress media into multiple bit rates, e.g. the G.722.1 coder encodes audio signals into 24 kbps, 32 kbps or 48 kbps [10]. Out-of-band methods such as Session Description Protocol

(SDP) are then used to convey bit rate information. It was assumed that the type of media encoding and bit rate remained constant for the entire streaming session.

TABLE I. RTP HEADER FIELD CLASSIFICATION

Field	Size (bits)	Details	U	T
Version (V)	2	Indicates the version number of RTP, 2 being the most recent.	I	I
Padding (P)	1	Indicates the presence of any end padding bits. The last octet of padding states how many padding bits there are.	R	R
Extension (X)	1	If set, it indicates that there is one header extension after the fixed header.	R	R
CSRC Count (CC)	4	Shows the number of contributing sources for the payload of the packet.	R	R
Marker (M)	1	Permits significant occurrences such as media frame boundaries to be marked.	A	A
Payload type (PT)	7	Specifies the payload type and the encoding or media compression schemes.	R	R
Sequence number	16	Increments by one for each successive packet sent out. Used by the receiver to detect packet loss and for packet reordering.	A	I
Timestamp	32	Identifies when the first octet in the RTP data packet was sampled. May be used for jitter and delay calculations.	A	A
Synchronisa-tion source (SSRC) Identifier	32	Randomly chosen number used to differentiate between different media streams in the same RTP session.	R	R
Contributing source (CSRC) Identifiers	Varies	Contains the list of SSRCs for the hosts whose information is contained in the RTP payload. Created by any mixers along the end-to-end path.	R	R

The compressed RTP-Lite header is shown in Figure 1.

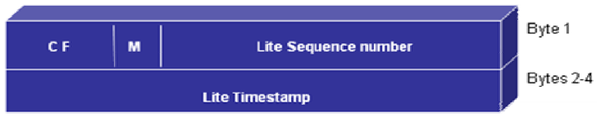

Figure 1: RTP-Lite over UDP header

The RTP header is compressed to 4 bytes, with the structure described below:

- Compression Field (2 bits): Has a value of 3. Differentiates the RTP-Lite header from the normal RTP header.
- Marker (1 bit): Same as in the 12-byte RTP header.
- Lite Sequence number (5 bits): Increments by one for successive 4-byte header packets sent, up to a maximum value of 31. At the client this field's value is added to the saved sequence number from the last uncompressed header, to regenerate the actual packet sequence number. The 5-bit field size ensures byte alignment.
- Lite Timestamp (24 bits): Difference between the timestamp of the last sent 12-byte header packet and the timestamp of the ensuing compressed RTP-Lite headers. The decompression process is similar to that of the Lite sequence number. The timestamp interval for regularly increasing timestamps of a video session is given in Equation 2 [11].

$$\text{RTP Timestamp Interval} = \frac{\text{Clock Rate (Hz)}}{\text{Frame Rate (frames per second)}} \qquad (2)$$

Video media encoding formats have a clock rate of 90,000 Hz [12]. The 24-bit Lite timestamp field thus encompasses a range of values that for a video stream can comfortably accommodate the sum of timestamp intervals for 31 successive 4-byte RTP-Lite header packets. Typically audio has smaller timestamp intervals than video [11] and therefore this range would again suffice.

A timeout mechanism is used as the proposed protocol operates in a unidirectional manner. The basic concept of the RTP-Lite timeout pattern is to match the compression cycle with the error characteristics of the channel. The ratio of uncompressed to compressed headers per cycle may be used to trade off error robustness for compression ratio. Therefore, on a highly reliable channel, a 1-31 cycle could be adopted where one uncompressed header packet is followed by 31 RTP-Lite compressed header packets. On a more loss-prone channel, the cycle may be reduced to 3-10 where 3 uncompressed header packets are sent per shorter cycle to reduce the probability of context storage errors. Shorter cycles trade off quick error recovery for lower compression gain. A 3-31 cycle was chosen to demonstrate the viability of this cyclical approach. The chosen number of uncompressed packets per cycle is three, rather than one, to increase the error robustness such that the loss of the single uncompressed packet does not result in an incorrect state being shared by server and client.

B. *RTP-Lite over TCP*

The 4-byte compressed RTP-Lite header structure for TCP transport is similar to that for UDP transport. The difference is that the last 5 bits of the first byte are unused because reliable, in order TCP packet delivery means that a sequence number field is unnecessary. However, to maintain byte alignment, the 5 bits are still required as padding. At the client the saved sequence number is incremented by one for each compressed header packet received until the next 12-byte RTP header packet is received. The Lite sequence number range is not a limitation as is the case for RTP-Lite over UDP. Therefore for a video stream, it can be surmised from Equation 2 that at least 186 RTP-Lite 4-byte header packets can be sent per cycle. A 3-

183 cycle was tested.

C. *Comparison with existing Header Compression Schemes*

The RTP-Lite compression framework builds upon existing RTP header compression protocols such as CRTP [9], ECRTP [7] and ROHC [3]. A key similarity is the concept of transmitting delta values to achieve a smaller header. Furthermore, like the three mentioned header compression schemes RTP-Lite is less error and loss resilient than the corresponding uncompressed RTP protocol.

RTP-Lite however differs from CRTP, ECRTP and ROHC in the following aspects:

- It achieves end-to-end compression of the RTP header and not merely link-by-link compression without needing to resort to tunneling with the associated disadvantages such as increased delay [16].
- RTP-Lite differential encoding uses as a reference not the previous packet but the most recent uncompressed packet header. As a result, RTP-Lite is less affected by the loss of a single packet than the other compression schemes. ROHC and CRTP are susceptible to loss propagation whereby loss of a single packet results in a loss of context for multiple received compressed header packets and possibly a significant delay before the context is restored [3, 9]. While the larger differential values of RTP-Lite mean a larger compressed header, the added robustness offsets the shortcoming of reduction in achievable compression gain.
- RTP-Lite is a simpler compression scheme to implement than CRTP, ECRTP and ROHC. For the proposed unidirectional operation, there is only one type of compressed header packet. The other compression protocols utilise multiple types of compressed header packets which while providing greater compression gain, make the compression schemes more complex. Furthermore, RTP-Lite unlike CRTP is not dependent on lower layers such as the link layer [9].
- RTP-Lite is a transport layer-independent compression protocol. Unlike the other header compression schemes that are UDP-specific, RTP-Lite can also be deployed over TCP making it a more flexible means to achieve throughput savings.

IV. Experimental Method

The open source Live555 streaming library [13] and ROHC-1.2.0 library [14] were used. Live555 was modified to create the compressed header packets and to decompress them at the client. The ROHC-1.2.0 library implementation of end-to-end header compression using tunneling was used to provide a baseline for evaluating the RTP-Lite protocol's performance. This work focused on low throughput networks and therefore the parameters for the video test clip were set accordingly. The video sequence of the bear advert with a sub-qcif (128x96) pixel resolution at 10 frames per second and a duration of 30 seconds was used. The clip was converted using ffmpeg, a media transcoding application [15] from the original avi format to the mpeg4 ES (elementary stream) format that Live555's test MPEG4ES streamer program could process. The average bit rate of the converted file was 54.5 kbps.

Two machines were set up to establish a media session over the public Internet as shown in Figure 2. The server was located at the University of Cape Town (UCT) in Cape Town, South Africa, with the client being stationed at the Council for Scientific and Industrial Research (CSIR) offices in Pretoria, South Africa. Traceroute results revealed that there were nine hops between the server and client.

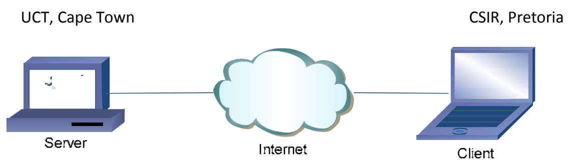

Figure 2: Illustration of streaming experimental setup

The server was a Proline UATX desktop machine with Intel Pentium(R) Dual-Core CPU E5200 @ 2.50 GHz and 2GB of RAM. The client machine was a Dell Inspiron E140S laptop with Intel(R) CPU, T1350 single core microprocessor @1.86 GHz and 504 MB RAM. All experiments were conducted with the server and client both using the Ubuntu 8.04 operating system with Linux kernel version 2.6.24.

Thirty sets of experiments were conducted each for the UDP and TCP transport investigations to provide an averaged performance result. For UDP, a ROHC tunnel was first established between the server and client. Each test set involved a comparison experiment for a standard RTP stream, an RTP-Lite stream and a ROHC stream run simultaneously between the end hosts using three separate Live555 streaming sessions. The experiments were carried out simultaneously to eliminate as much as possible the effect of the spurious changes in network conditions that characterise the public Internet and to ensure environments that were nearly identical for the flows being compared. ROHC-1.2.0 does not provide RTP over TCP functionality and therefore the TCP experiments involved only RTP and RTP-Lite flow comparison. All jitter, throughput, loss, packet size and header size measurements were recorded from Live555.

V. Results and Analysis

A. *RTP-Lite over UDP*

1) *Header Size and Compression Gain:* The measured average RTP-Lite packet header size for the 3-31 packet cycle was 4.7 bytes, less than the 12 byte average of the standard RTP headers. As ROHC operates below the

application layer, the ROHC streams' RTP header size measurements obtained from Live555 corresponded to those of standard RTP. The measured average packet size values for RTP and RTP-Lite were 634.9 bytes and 627.6 bytes respectively. The average RTP header compression gain computed using Equation 1 was 61%.

2) *Throughput:* The average RTP throughput rates for the 30 repetitions of the experiments were 56.84 kbps, 56.21 kbps and 56.85 kbps for the RTP, RTP-Lite and ROHC flows, respectively. The throughput results are plotted in Figure 3. The magnitude of throughput gain was small relative to the total throughput rate, but it is sufficient to verify the lower link capacity required by the RTP-Lite stream. The ROHC and standard RTP throughput values were almost the same, which is expected as the ROHC packets are decompressed below the network layer and so the RTP headers arriving at the application layer are in an uncompressed format.

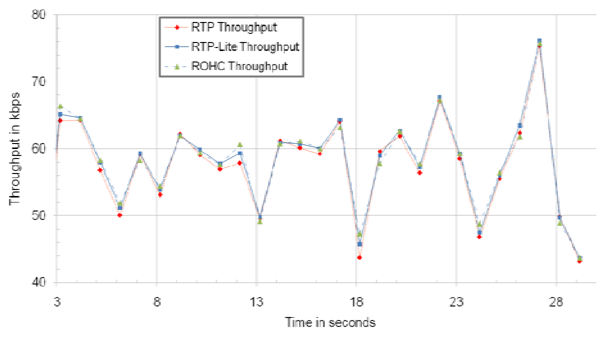

Figure 3: Throughput for the RTP, RTP-Lite and ROHC flows over UDP

3) *Loss*: In all 30 repetitions of the experiments, the percentage packet losses for the three flows remained below 1%. The average percentage loss for the RTP flows was 0.08%, for the ROHC flows it was 0.01% and that of the RTP-Lite flows was 0.17%. The RTP-Lite flows experienced a greater average percentage loss than the RTP flows. This correlates to the expectation of a higher packet loss rate for end-to-end compression [8] but the magnitude of increase is not significant. ROHC, because of its robust design and feedback mechanisms, showed better performance than both standard RTP and RTP-Lite.

4) *Jitter:* For the standard RTP flow the average measured jitter was 0.82 ms and for RTP-Lite the average measured jitter was lower at 0.77 ms. The ROHC flow had an average jitter of 0.91 ms. The jitter results are plotted in Figure 4. The increased jitter for ROHC can be attributed to the tunneling mechanism that degrades delay performance [16]. Thus the jitter experienced by RTP-Lite packets was on average 6% less than that experienced by the RTP stream packets while that of ROHC was 10% higher than for standard RTP. A smaller jitter value implies that a smaller buffer is needed at the client to compensate for the delay variations. The 6% reduction in jitter illustrates that overall RTP-Lite does not result in added jitter, as depicted in Figure 4.

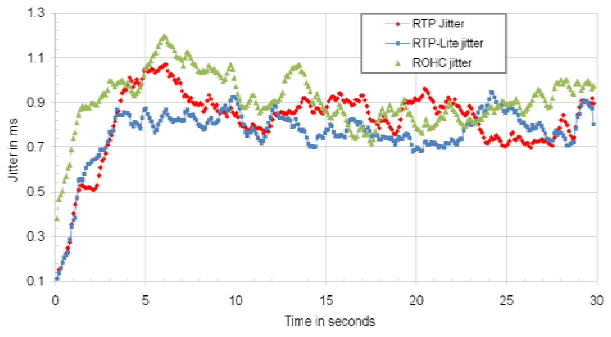

Figure 4: Jitter for the RTP, RTP-Lite and ROHC flows over UDP

B. *RTP-Lite over TCP*

1) *Header Size and Compression Gain:* The average RTP-Lite over TCP flow packet header size for the 3-183 packet cycle was found to be 4.1 bytes. The measured average packet sizes for RTP and RTP-Lite were 634.9 bytes and 628.0 bytes respectively. The difference in these average sizes was 7.9 bytes, confirming the expected reduction in average header size from 12 bytes to 4.1 bytes. The average header compression gain calculated using Equation 1 was found to be 66% indicating the obvious, that the larger the number of compressed packets per cycle, the greater the compression efficiency.

2) *Throughput:* The average throughput rates of all 30 repetitions of the experiments for RTP and RTP-Lite were 56.88 kbps and 56.19 kbps, respectively. The throughput results are plotted in Figure 5 and they show that the throughput rate for RTP-Lite is less than that of standard RTP, indicating more efficient use of link capacity.

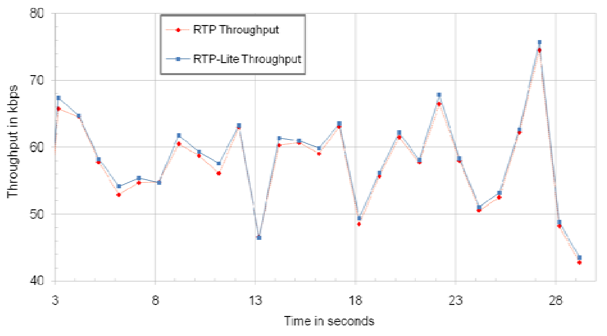

Figure 5: Throughput for the RTP and RTP-Lite flows over TCP

3) *Jitter:* For the standard RTP flow the average measured jitter was 4.97 ms while for RTP-Lite the average reported was 5.08 ms. Figure 6 is a plot depicting the measured jitter characteristics of the RTP and RTP-Lite flows. With TCP transport, RTP-Lite has a 2% greater average jitter than standard RTP that can be partly attributed to the retransmission properties of TCP that affect the timing of packet delivery. The increase in jitter is considered to be an insignificant effect on the stream quality relative to standard RTP. The measured average interpacket gap values for RTP and RTP-Lite were 90.86 ms and 90.87

ms respectively while the average maximum interpacket gap values were 231.70 ms and 217.86 ms, respectively. Thus standard RTP packets were more likely to be treated as late arrivals at the client and be discarded than RTP-Lite packets.

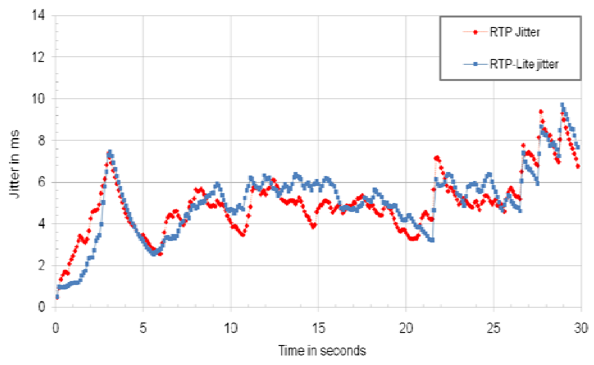

Figure 6: Jitter for the RTP and RTP-Lite flows over TCP

VI. Conclusion

An application layer compression scheme for RTP was developed. Improved throughput efficiency was demonstrated using the proposed RTP-Lite protocol. For the uncompressed-compressed packet header cycle of 3-31 that was used in the experiments over UDP, the average header compression gain was 61%. Larger throughput gains could be attained by reducing the number of uncompressed header packets sent in each cycle for a highly error-resilient channel. RTP-Lite over TCP for a 3-183 cycle generated an average header compression gain of 66%. For UDP transport, a reduction in measured jitter of 6% and 15% was achieved by RTP-Lite compared to standard RTP and ROHC, respectively. The 2% increase in RTP-Lite jitter relative to standard RTP for TCP transport was considered insignificant. The RTP-Lite over UDP streams experienced a slightly greater average loss than the corresponding RTP and ROHC streams that is offset by the throughput gains that are of paramount importance on low bit-rate links. The performance improvements were more pronounced for the UDP case, from which it can be concluded that the proposed RTP-Lite protocol is better suited to UDP than TCP.

As future work, for added robustness RTP-Lite could be extended to provide a feedback mode at the application layer with extensions to the RTCP protocol. Furthermore, RTP-Lite could be modified to adapt the packet cycle dynamically to the error and loss characteristics of the channel over which it is deployed. Use of a feedback channel such as RTCP may permit adaptation of the cycle pattern on the fly. WLSB (Windows Least Significant Bit) encoding of the sequence number and Timestamp fields may be incorporated to attain greater compression gain levels.

References

[1] S. R. Gandham and K. V. Muppalla, “Header compression mechanism for transmitting RTP packets over wireless links,” US Patent application, Pub. No. 20090268667, Publication date 29 October 2009.

[2] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” RFC 3550, July 2003.

[3] 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, July 2001.

[4] Keiko Tanigawa, “Simple RTP Multiplexing Transfer Methods for VoIP,” IETF Internet Draft, May 1999. http://tools.ietf.org/html/draft-tanigawa-rtp-multiplex-01.

[5] C. Lamy-Bergot and P. Vila, “Multiplex header compression for transparent cross-layer design,” Proc. IEEE International Conference on Networking (ICN`04), March 2004, pp. 1084-1089.

[6] F.H.P. Fitzek, S. Hendrata, P. Seeling and M. Reisslein, “Header Compression Schemes for Wireless Internet Access”, chapter in Wireless Internet, CRC Press, 2004.

[7] 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, July 2003.

[8] V. Jacobson, “Compressing TCP/IP Headers for Low-Speed Serial Links,” RFC 1144, February 1990.

[9] S. Casner and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links," RFC 2508, February 1999.

[10] P. Luthi and R. Even, "RTP Payload Format for ITU-T Recommendation G.722.1", RFC 5577, July 2009.

[11] C. Perkins, “RTP Audio and Video for the Internet,” Addison-Wesley, New York, USA, 2003.

[12] S. Casner, “Media Type Registration of Payload Formats in the RTP Profile for Audio and Video Conferences,” RFC 4856, March 2007.

[13] Live555 Streaming Library. http://www.live555.com/liveMedia/public/.

[14] ROHC-1.2.0 Library. French space agency (CNES), Thales Alenia Space (TAS) and Viveris Technologies, “Robust Header Compression library”. https://launchpad.net/rohc.

[15] FFmpeg. http://www.ffmpeg.org/index.html.

[16] X. Zhou, M. Jacobsson, H. Uijterwaal and P. Van Mieghem, “IPv6 delay and loss performance evolution,” International Journal of Communication Systems, Vol. 21, No. 6, June 2008, pp. 643-663.

Josephine Nakato Kakande received her Masters degree in Electrical Engineering specializing in Telecommunications from the University of Cape Town in 2010. She is currently a Researcher with the Real Time Video Coding Project at the CSIR Meraka Institute. Her research interests include multimedia streaming, communication protocols and wireless networks.