P2ProxyLite: effective video streaming in wireless ad-hoc networks

R. Fracchia and C. Lamy-Bergot
Thales Communications, 160 boulevard de Valmy, 92740 Colombes, France

***Abstract*— Video streaming over wireless connections remains a challenge today: even when data link and transport layer protocols are tuned to avoid retransmissions and thus reduce the delay and the used bandwidth, the impact on the perceived video quality of imperfect and band-limited radio channels is not negligible. This phenomenon is even more pronounced in multi-hop ad-hoc networks, where the received video quality is further reduced by a cascade of wireless transmissions causing additional delay, losses and errors. We propose in this work to enhance video streaming performance in ad-hoc networks by introducing in mobile nodes an improved frame treatment at the Data Link layer, associated to caching capabilities at the transport layer. This approach allows, on the one side, the forwarding of multimedia packets even in case of residual bit errors, which can be tolerated by robust video decoders, and, on the other side, a path reduction when multiple users are interested in the same content.**

I. Introduction

Streaming of video content has often been indicated as a killer application for 3G wireless networks: even though service popularity took longer than expected to build up among end-users, the importance of video services is now testified by the explosion of video offer for streaming and sharing over the Internet and by the availability of applications to access these videos from widespread smart-phones. The diffusion of these applications on mass market portable devices has, as a consequence, introduced the thirst for similar services in different class of users, stuck even nowadays to devices offering basic communications. For instance, several organizations (i.e., rescue services, fire brigades, etc.) have shown an interest in next generation portable radio posts allowing video streaming.

Besides the well-know problems for video distribution over wireless channels, the support for multimedia transmissions on ad-hoc networks introduces further challenges. In these networks, where several hops and thus several wireless transmissions are necessary to reach the destination, the overall probability of transmission errors is higher and the need to save resources like bandwidth and battery is even more pronounced than in structured wireless networks.

At the same time, it is known that multimedia applications could be less sensitive than other applications to the transmission errors introduced along the path. Indeed, applications for audio and video data transmissions often use encoders that tolerate some bit errors in the data and thus the reception of an erroneous packet is preferred over a packet loss.

This work has been carried out thanks to the French National Research Agency who funded the RNRT DITEMOI Project

In this paper, we propose to exploit the distinctive robustness of multimedia applications to limit their bandwidth occupancy and to reduce the delay perceived by the client of a multi-hop wireless communication. The idea is, on the one hand, to avoid at the MAC layer unnecessary retransmissions of packets affected by errors and, on the other hand, to shorten the path necessary for content retrieval by introducing proxying and caching capabilities at the Real-time Transport Protocol (RTP) layer [1].

The rest of the paper is organized as follows. Work with a similar approach and thus related to our study is presented in Section II of this document. Section III overviews the proposed solution, while Section IV and Section V present the details of the P2ProxyLite approach and the modifications of the concerned protocols. Section VI reports the performance evaluation of the proposed scheme and Section VII concludes this work.

II. Related Work

The idea of transferring to the upper layer of the networking stack packets carrying corrupted payloads is due to the ability of audio, image and video decoders to absorb a residual error probability. Indeed, these decoders apply masking techniques based on the capability of human eyes and hears to “accept” or compensate residual defects. Several examples could be reported: the H.263 [2], H.264 [3], JPEG 2000 [4] standards are based on this assumption. Moreover, errors or losses could be corrected by introducing Forward Error Correction (FEC) techniques at the transport or the application layer. Work on this topic can be found in [5], [6], [7] and an IETF work-group (FecFrame [8]) has been recently created.

This approach at the application layer has however to be coupled with the support of lower layers: in the last years, different protocols allowing to transfer to the upper layers erroneous packets have appeared and became standard. In order to avoid packet loss for few acceptable bit errors, these protocols divide packets into *sensitive* and *insensitive* parts: errors in the sensitive part should result in dropped packets, while errors in the insensitive part should not. WiMAX [9], for example, introduces a partial CRC at the data link layer, limited to the frame header, thus allowing to transfer to the network layer frames with errors in the payload. Similarly, transport protocols like UDPLite [10] and DCCP [11] introduce a partial checksum on transport layer segments, thus allowing a verification of the integrity of the segment header

only or of a portion of the payload as well (e.g., the RTP header).

Packet caching in the network has also recently received a lot of attention: indeed, data caching can significantly improve the efficiency of information access in wireless ad-hoc networks by reducing the access latency and bandwidth usage. In order to cope with frequent disconnections causing network division in ad-hoc communications, [12] proposes to maintain in every node a small buffer for caching data packets that pass through it: when a downstream node encounters a forwarding error, an upstream node, with the pertinent data in its buffer and alternative route, can retransmit the data. Similarly, authors of [13] propose to replicate data items on mobile hosts to improve data accessibility. Work in [14] analyzes instead the cache placement problem in order to minimize the total data access cost in ad hoc networks with multiple data items and nodes with limited memory capacity. Finally, Cache-and-Forward, an architecture that addresses efficient mobile content delivery, has been recently presented in [15]: assuming rapidly decreasing storage costs, hop-by-hop opportunistic transport with large in-network caching and content-aware routing is proposed as a basic network functionality rather than overlay.

The novelty of the work presented here is to combine caching mechanisms with the delivery of multimedia packets with errors in the payload thus to exploit the robustness of many audio and video decoders.

III. The P2Proxylite approach

In this work we consider an ad-hoc network composed by m nodes where $n < m$ mobile terminals (called clients in the following) are interested in the same multimedia content. This content is streamed from a node (called source) that can be located more than one hop away. The different clients thus independently contact the source which starts n different transmissions of the same content. As a consequence, if one or more clients share at least a portion of the path to the source, the nodes along the common part of the path have to route several times the same content, with a corresponding waste of bandwidth. Moreover, even when paths are disjoint, retransmissions of multimedia packets affected by few errors are often useless, since errors can be usually concealed by the decoders, and furthermore introduce an annoying delay.

We thus propose in this work to combine two different axes of improvement: the calculation of the partial checksum and the introduction of intelligent proxies. Our aim is to improve the throughput of the video streaming, thus offering the possibility to either increase the video quality given a fixed bandwidth or to reduce the network capacity utilization given a constant video quality.

A. An overview

The proposed solution is composed by the combination of two approaches: partial checksum and data caching.

More in detail, we propose first to *forward packets with correct headers but corrupted payload* to the next hop toward the destination and to send them up to the application when the destination is reached. The bandwidth occupancy could thus be dramatically reduced by avoiding unnecessary retransmissions if a robust decoder is available at the receiver side.

It may happen, however, that the forwarded corrupted packets are not decodable by some users (i.e., users without a robust decoder) or that the error probability is high and the received information is not sufficient for a robust multimedia decoder to efficiently conceal the residual errors. P2ProxyLite nodes may in this case contact upstreams nodes until finding an error-free copy of the packet. We thus propose to *store the multimedia content* along the path to increase the probability of an error-free delivery. Moreover, correct packets may replace cached corrupted ones along the path. Finally, the cached content can be used to respond to new streaming requests, thus avoiding to contact the original video source and reducing the path length.

In practice, every P2Proxylite node works on the ongoing packets sent upstream and downstream to: i) associate each packet to an identifier and detect eventual bit errors on header or payload; ii) store the error status information and the packet in a cache R at the RTP level; iii) forward it to the next hop after a CRC recalculation; iv) intercept, analyze and forward the new content requests toward the source if the desired content is not locally cached or if the client has not been able to decode a corrupted copy; send the stored content otherwise. Each of these steps is accurately described in the following sections.

Finally, it has also to be noticed that is not necessary that all nodes implement the P2Proxylite functionalities: indeed, a few enhanced nodes, acting as proxy in the ad-hoc network, could be sufficient to reduce the number of retransmissions and the number of hops. This system can thus be incrementally deployable.

B. Forwarding of packets affected by errors

In this section we detail the additional steps carried on in the P2Proxylite nodes to enable the forwarding of packets whose payload is affected by errors:

- When a packet is received at the wireless interface, if the node is not the final destination of the communication, the headers integrity has to be verified. The forwarding to the following hop will be done only if all the packet headers (MAC, IP, transport and RTP) are error-free.
- When the received packet has a corrupted payload, if an error-free copy has been previously received and cached in R, the correct payload replaces the erroneous one and is transmitted to the next hop. If not, the corrupted packet will be forwarded to the next hop.
- At the Data Link layer, the CRC is recalculated before the transmission, even for corrupted packets. In this way, if the following node along the path does not implement P2ProxyLite functionalities, the packet will not be discarded unless further errors are introduced. The CRC recalculation allows also to determine if errors have been added in the last hop, as better explained in Section IV.

At the same time, since the CRC recalculation resets the error memory, we propose to introduce a new field in the

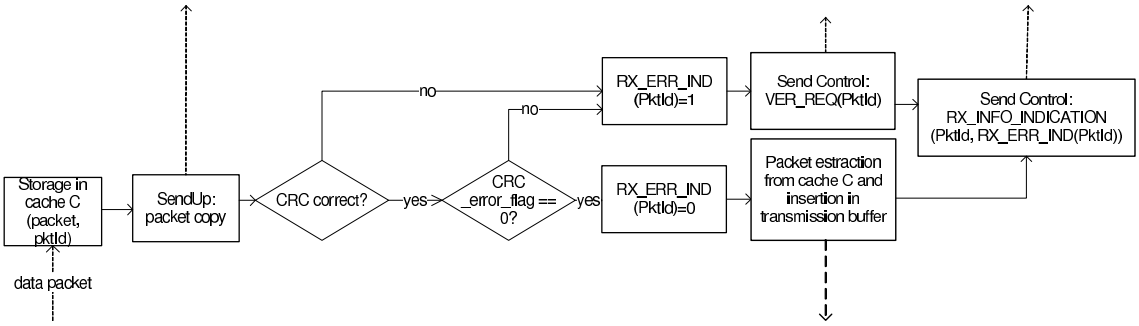

Fig. 1. Flow-chart of additional Data Link functions

TABLE I

$PktId$: IDENTIFIER OF THE MULTIMEDIA FLOW AND PACKET

File ID	Fragment ID	Additional info (optional)	CRC_error_flag
8 bytes	4 bytes	6 bytes	1 bit

packet, called CRC_error_flag, to keep the history of packet corruption, whose goal is threefold. First, by identifying error-free packets we can replace a cached corrupted packet with a correctly received one (or a corrupted received packet with a correct cached one); second, it introduces the possibility to forward also corrupted or only correct packets, depending on the client willingness; third, it indicates to the decoder a corrupted image reception and the need of a robust decoding process. We propose to insert this information, together with a content identifier, in the RTP extension header, introduced by setting the RTP Extension Bit to 1. The packet identification has to be unique and has to represent first the multimedia flow and then the packet in the flow. A possible format for the identifier, called $PktId$, is presented in Table I: the File Id could be obtained by hashing on the source channel in case of live streaming or on the file name in case of Video On Demand (VoD) (e.g., with algorithm MD5 on 64 or 128 bits [16]); the Fragment ID identifies the fragment in the file; the additional fragment information field is optional and could be used to indicate eventual operations done on the fragment (FEC, CRC type,...); finally, the CRC_error_flag is used to indicate if the payload is correct or corrupted.

The use of header compression mechanisms (e.g. RoHC [17]) could be envisaged to cope with the additional overhead introduced by the extension of the RTP header.

C. Path shortening via caching in intermediate nodes

Every time a packet is received on the wireless interface, a copy of the packet is transferred to the upper layer for storage in a cache at the RTP layer. The identifier of the packet is included in the RTP header, whose integrity can be verified by opportunely tuning the checksum coverage of transport layer protocols like DCCP et UDPLite: for this reason, the packets arriving at the RTP layer (and thus not discarded at the lower layers) carry a correct content identifier. RTP thus stores in the cache R for a time T_R the packet and its identifier, together with an identifier of the client (e.g., destination IP address and port number). Caching time may be short in case of live-streaming/interactive applications or long in case of VoD services. The effect of this parameter on data topicality will be subject of further study.

We additionally propose to intercept all the Real Time Streaming Control (RTSP) [18] commands, which are used to require, start and stop a streaming session. When a streaming request is received at the Data Link layer, the packet is sent to the session layer, in charge of handling the streaming sessions.

The session layer then compares the received request with the identifiers of the stored packets. If the desired content is not available, the request is routed to the source. If the content is available but the cached copy is affected by errors, the P2ProxyLite node compares the identifier of the source of the new request with the list of the identifiers of the previous clients: if the client identifier is in the list, a copy of that data has already been sent to that user. However, since the client is requesting the content again, it may not have been able to benefit from that transmission of data with errors in the payload: in this case, the request is routed toward the source, aiming at increasing the probability to deliver a correct copy of the multimedia content. Finally, if an error-free copy is cached (or if the P2ProxyLite node has not tried to forward that corrupted packet before), it directly responds with the stored content, thus avoiding to route the request toward the source.

IV. DATA LINK MODIFICATIONS

The flow chart of the additional operations required at the data link layer is reported in Figure 1.

When a data frame is received at the Data Link layer, the content identifier $PktId$, included in the RTP layer, is extracted and used to identify the received frame. This frame is stored in a cache C for a time T_C (corresponding to the time needed for integrity checks and cache R access operations): it will be removed from C and forwarded to the next hop after a verification of the headers integrity or it will be discarded at the expiration of T_C.

The CRC computed on the whole packet is then compared to the CRC included in the packet header: if they do not match, one or more bit errors affected the packet in the last hop and a verification of network, transport and RTP headers integrity is necessary. A request for this verification (e.g., VER_REQ message) is sent via a control interface to the RTP layer.

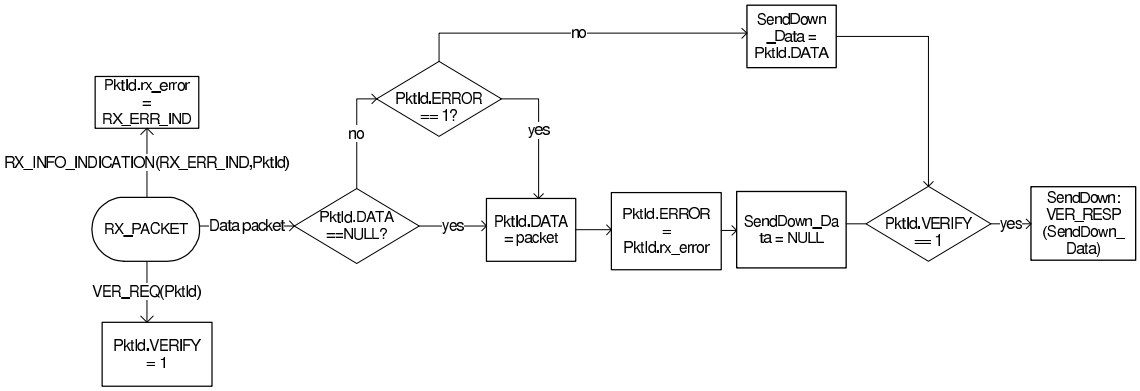

Fig. 2. Flow-chart of additional RTP functions

Conversely, if the computed CRC corresponds to the value included in the header, no errors have been introduced in the last hop and the status of the packet is unchanged from the previous hop (i.e., from the value reported in the CRC_error_flag). If the CRC_error_flag is equal to zero, no errors have been introduced along the path and the multimedia content is correct: the packet can thus be extracted from the cache C and queued for transmission to the next hop. Otherwise, the header is error free but the payload is affected by bit errors. In this case, a VER_REQ message is sent to the RTP layer even if the status of the packet is known: this allows to check if an error-free copy of that payload is cached at the RTP layer. The information on the packet status (i.e., RX_ERR_IND) is also sent to the RTP layer in the RX_INFO_INDICATION message, since we assume that the transport layer checksum verifies only the packet header.

The response to the verification request (i.e., the VER_RESP control message) - whose reception is not sketched in Figure 1 for sake of simplicity - may or not include a packet payload. If the control message does carry a packet payload, then an error-free copy has been previously cached in R and can replace the packet with errors: the payload included in the VER_RESP message is queued for transmission. Otherwise, no error-free copy is available: the packet temporarily stored in C is extracted, its CRC_error_flag is set to one and queued for transmission.

V. RTP MODIFICATIONS

At the RTP layer, the information carried by VER_REQ or RX_ERR_IND control messages is associated to the packet and cached as shown in Figure 2: the verification request is associated to the $PktId$ in the first case while the packet integrity value is stored in the second case.

When a data packet is received, the RTP layer verifies if a copy of that packet is stored in R.

If the packet is received for the first time (i.e., PktId.DATA==NULL) or if a corrupted copy is cached (i.e., PktId.DATA!=NULL and PktId.ERROR==1), the received packet is stored in R and the ERROR indication is set to the value received in the RX_ERR_IND message. If a verification has been required by the Data Link layer and the packet has been received by the RTP module (i.e., it has not been discarded by the transport layer), a VER_RESP message without payload is transmitted to the Data Link layer: this message indicates that there are no errors in the packet header but that no error-free copy of the payload is available.

Conversely, if a correct copy of the packet is cached, the received packet is discarded and, if a verification is needed, a VER_RESP message, carrying the error-free data packet, is transmitted to the Data Link layer.

The identifier of the client is finally associated to the content identifier, to maintain for a short time a list of the clients receiving a corrupted content: this information will be used to decide if a new request for the same content has to be forwarded to the source, as explained in Section III-C.

VI. SIMULATION RESULTS

In the following, we present the performance evaluation related to the two main P2ProxyLite functionalities.

To evaluate the gain derived from the path reduction associated to the caching function, we assume that two different clients C_1 and C_2 request at different times (C_1 before C_2) the same multimedia content to the source S, which is reached via a multi-hop transmission. Let be c_1 and c_2 the number of hops between the node S and the nodes C_1 and C_2 respectively and p the number of hops that the two multi-hop paths have in common (i.e., the hops from the source to the last P2ProxyLite node P in common to the two paths), like depicted in Figure 3. It follows that, to verify the condition of p hops in common and of $C_1 \neq C_2$, we have $c_1 \geq p$ while $c_2 > p$.

Without proxy functions, the transmission of each multimedia packet toward C_1 and C_2 requires a total of $N = c_1 + c_2$ hops. With the proxy function, the number of hops for the transmission toward C_1 is again c_1, while the number of hops for the transmission toward C_2 is $c_2 - p$, since node P cached the stream for node C_1 and directly responds to C_2 with the desired content. The resulting gain in terms of path reduction is thus $G = \frac{p}{c_1+c_2}$.

Figure 4 shows the gain G for a maximum distance of 30 hops to the source S. The gain is plotted as a function of

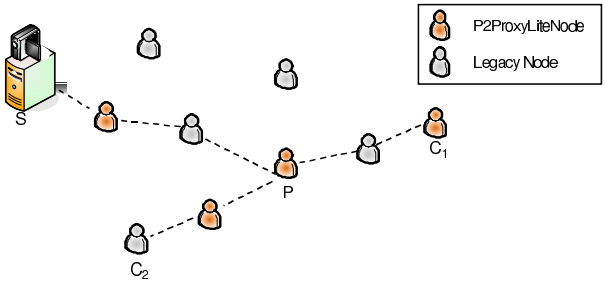

Fig. 3. Considered scenario for P2ProxyLite evaluation

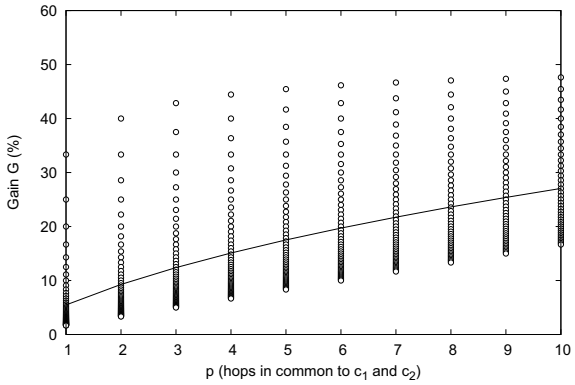

Fig. 4. Path reduction in case of two video requests

the hops p in common to c_1 and c_2 and for all the different combinations of c_1 and c_2 (represented by circles in the figure) satisfying the conditions $p \leq c_1 \leq 30$ and $p < c_2 \leq 30$. The average gain, indicated by the solid line in the figure, varies between 5.5% and 27% for p ranging from 1 to 10. For greater values of p such as 30 common hops, the average gain attains even 50%.

The performance of the video streaming is evaluated with a simulation framework based on OMNeT++ [19]. This tool simulates the transmission (with video bits actually generated and sent) of a precoded H264/AVC video and accurately models all the OSI layers from the application to the Data Link layer, while a simplified model of the physical layer is introduced.

The video streaming is started and controlled by RTSP which allows the end-user to request the desired content (the reference CIF Foreman sequence @30 Hz in the simulation). When the streaming is activated, the application layer at the source side transmits the images extracted from the precoded video. At the client side, a robust decoder transforms the received H264 frame into an uncompressed yuv video frame which is then displayed. Below the application layer, RTP fragments each image into packets and handles the image reconstruction at the receiver side. At the transport and network layers, the UDPLite transport protocol and IPv6 are used respectively. At the Data Link layer, a basic IEEE 802.11g MAC protocol is simulated, while data are transmitted at the physical layer with a data rate of 6 Mbps. An AWGN channel

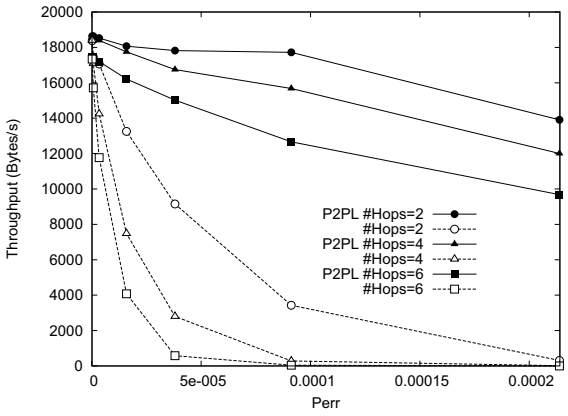

Fig. 5. Average throughput for different error probabilities and hops

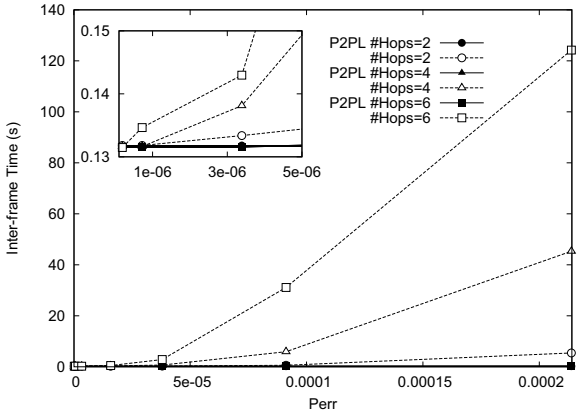

Fig. 6. Time between the reception of two consecutive video frames

model, introducing random bit errors, is used to represent the effects of the radio channel on the wireless transmission.

Figure 5 reports the average throughput at the application layer as a function of the per hop bit error probability, for three different distances between the source and the client (i.e., 2, 4 and 6 hops respectively). We can observe that, as expected, the throughput decreases when the error probability per hop or the number of hops increase. The throughput reduction is objectively smoother in the P2ProxyLite case, since packets are discarded only when affected by errors on the header. Conversely, in the reference case the corrupted packets are not forwarded, thus dramatically reducing the number of received packets. We can observe that for 4 or 6 hops, even with a per hop bit error probability lower than 10^{-4}, the throughput is zero in the reference case since packets are discarded with very high probability in one of the nodes along the path.

In Figure 6 we report the time elapsed between the reception of two consecutive video frames at the client side, as a function of the bit error probability and of the number of hops. In the reference scenario, the inter-frame time increases rapidly with the error probability since many packets, necessary to recreate the video frame, are discarded along the path. The inter-frame time remains instead constant with the P2ProxyLite approach.

Beside those statistics, we can evaluate the video quality as

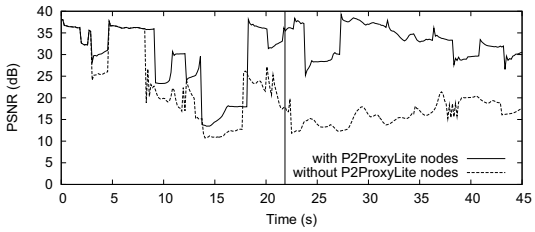

Fig. 7. PSNR *vs.* streaming time comparison in reference and optimized processing (above) and comparison of two video sequences quality at t=21.5s (below, left: reference, right: optimized).

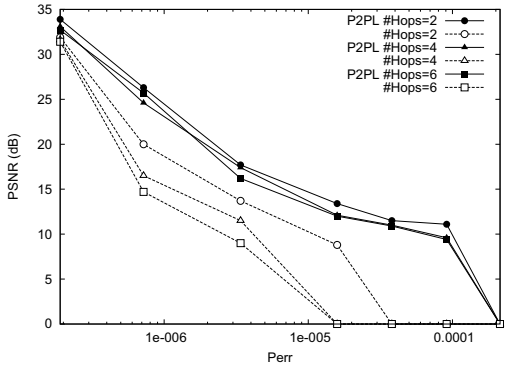

Fig. 8. Average PSNR for different error probabilities and hops

all the bits of the compressed video have been generated and transmitted. By reconstructing the video at the receiver side image after image, we derive the Peak Signal to Noise Ratio (PSNR), a classical objective measure of the video quality computed by comparing the original video with the received one. The upper part of Figure 7 reports the variation of the PSNR as a function of the streaming time, for a bit error probability of 10^{-6} and 4 hops, in two cases: the reference (dashed line) and the P2ProxyLite (solid line) realisations respectively. Moreover, to better illustrate the difference, we present as an example snapshots of the images decoded in the reference and optimized cases for the frame number 144, transmitted at $t = 21.5s$ and identified by a vertical line in the upper figure. Figure 7 allows thus at the same time an objective and a subjective evaluation of the video quality. In the two representations, we notice an actual improvement achieved with the P2ProxyLite solution.

Finally, we plot in Figure 8 the average PSNR as a function of the error probability and of the number of hops. We can notice that the trend observed in Figure 5 and Figure 6 is reflected also by the average PSNR: the P2ProxyLite approach offers a gain between 5 and 15 dB, depending on the number of hops and the bit error probability, thus significantly increasing the perceived video quality.

VII. Conclusion

This work presented a combined approach to improve the efficiency and the quality of video streaming in ad-hoc networks based on the enhancement of Data Link and RTP layer functionalities. By decreasing the number of retransmissions and by introducing caching mechanisms in enhanced nodes, the proposed scheme allows, from the one side, to reduce the delay of video streaming and to increase the final PSNR and, from the other side, to speed up the access to a multimedia content of interest for two or more clients. Results, achieved with a complete system simulator modeling the transmission of real video content over a full networking stack, confirm the gain introduced by the proposed solution in terms of delay, throughput and video quality perceived by the clients.

References

[1] S. Casner, H. Schulzrinne, et al., "RTP: A Transport Protocol for Real-Time Applications," Audio-Video Transport Working Group, IETF RFC 1889, January 1996.
[2] ITU-T and ISO/IEC JTC 1, "H.263 : Video coding for low bit rate communication," ITU-T Recommendation H.263.
[3] ITU-T and ISO/IEC JTC 1, "Advanced video coding for generic audio-visual services," ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), Version 1: May 2003.
[4] SO/IEC. 15444-1. Second edition. 2004-09-15. Information technology JPEG 2000 image coding system: Core coding system.
[5] C. Bergeron and C. Lamy-Bergot, "Modelling H.264/AVC sensitivity for error protection in wireless transmissions," in Proc. of the International Workshop on Multimedia Processing (MMSP'06), October 2006.
[6] C. Soldani, G. Leduc, F. Verdicchio, A Munteanu, "Multiple Description Coding versus Transport Layer FEC for Resilient Video Transmission," in Proc. of IEEE International Conference on Digital Telecommunications (ICDT'06), 2006.
[7] H. Seferoglu, U.C. Kozat, M.R. Civanlar, J. Kempf, "Congestion state-based dynamic FEC algorithm for media friendly transport layer," in Proc. of Packet Video Workshop (PV'09), 2009.
[8] http://www.ietf.org/dyn/wg/charter/fecframe-charter.html
[9] IEEE, "IEEE Standard for Local and Metropolitan Area Networks Part 16: air interface for fixed broadband wireless access systems, Standard 802.16-2004", IEEE, Washington, DC, 2004.
[10] IETF, "The Lightweight User Datagram Protocol (UDP-Lite)," RFC 3828, July 2004.
[11] IETF, "Datagram Congestion Control Protocol (DCCP)," RFC 4340, March 2006.
[12] A. Valera, W.K.G. Seah and S.V. Rao, "Cooperative Packet Caching and Shortest Multipath Routing in Mobile Ad hoc Networks," in Proc. of IEEE Infocom'03.
[13] T. Hara, "Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility," in Proc. of IEEE Infocom'01.
[14] B. Tang, H. Gupta, S. Das, "Benefit-based Data Caching in Ad Hoc Networks," IEEE Trans. on Mobile Computing, vol. 7, no. 3, pp. 208-217, Mar. 2008.
[15] S. Paul, R. Yates, D. Raychaudhuri, and J. Kurose, "The cache-and-forward network architecture for efficient mobile content delivery services in the future internet," in Proc. of the First ITU-T Kaleidoscope Academic Conference on Innovations in NGN: Future Network and Services, 2008.
[16] R. Rivest, "The MD5 message-digest algorithm (MD5)", IETF RFC 1321, April 1992.
[17] Bormann et al., "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed," IETF RFC 3095, July 2001.
[18] H. Schulzrinne et al., "RFC2326 - Real Time Streaming Protocol (RTSP)," IETF RFC 2326, April 1998.
[19] http://www.omnetpp.org/