UDP Lite for Real Time Multimedia Applications

Lars-Åke Larzon[1], Mikael Degermark[1,2], Stephen Pink[1,2]

[1]Luleå University of Technology, Luleå
[2]Swedish Institute of Computer Science, Stockholm
{lln,micke,steve}@cdt.luth.se

Abstract

We introduce UDP Lite – a lightweight version of UDP with increased flexibility in the form of a partial checksum. It allows senders to specify packets as partially insensitive to errors. The coverage of the checksum is specified by the sending application on a per-packet basis. Because of its close relationship to UDP, UDP Lite is easily integrated into an existing UDP implementation.

UDP is a simple best-effort transport protocol that adds multiplexing and an optional checksum to IP. Unlike TCP, UDP does not provide reliability, in-order delivery or congestion control, which has made it especially popular among delay-sensitive real-time applications.

Audio/video applications often prefer damaged packets over lost packets. One way for an application to allow delivery of damaged packets is to disable the UDP checksum. This would mean, however, that important application-specific headers might pass unverified. Also, in the next version of IP, IPv6, the UDP checksum is mandatory since there is no header checksum in IPv6. These applications could benefit from using UDP Lite instead of UDP. By reflecting the UDP Lite policy with a partial checksum onto the link layer, the gain can be even higher.

1 Introduction

The number of applications that are sensitive to network delay is increasing. Examples include real-time communication between two or more hosts, media on demand, networked multiplayer games, streaming applications, etc. Most of these applications use the User Datagram Protocol (UDP)[5] as their transport protocol. UDP has properties that make it suitable for these kinds of applications:

- The data rate is defined by the sending application.
- Incoming packets are delivered immediately to the receiving application, even if they arrive out of order.
- Lost packets will not cause retransmissions by the transport layer.
- For validation purposes, the Internet checksum [6] can verify the UDP headers and the data payload.

The UDP protocol headers are shown in figure 1. Shaded fields are the fields of the pseudo header provided by the IP layer, and white fields belong to the UDP header. The UDP checksum covers the conceptual IP pseudo-header in order to protect against misrouted packets.

If the checksum is enabled and fails at the receiving side, e.g., due to a single-bit error, the entire packet is discarded. Many real-time applications encode audio/video in a format that handles single-bit errors in the data payload better than the loss of a full packet. A packet with a few bit errors will cause a glitch in the experienced audio/video, while a lost packet can cause an

IP pseudo-header	Source address		
	Destination address		
	Zero	Proto	UDP length
UDP header	Source port		Destination port
	Length		Checksum

Figure 1: The UDP headers, from RFC 768 [5].

annoying pause for audio or a noticeable disturbance for video. There are coding schemes for which a packet loss can make subsequently received packets unusable since they depend on data in the lost packet. In such cases it would be better to avoid interruptions by delivering packets with acceptable errors to the application.

This is relatively easy to do when using IPv4 since the UDP checksum is optional.[1] By turning the checksum off, it is up to the receiving application to detect if there are errors, and if they are acceptable. This will add extra processing overhead and complexity at the receiving side. A better solution would be to move parts of this functionality away from the user application into the transport layer. Moreover, disabling the checksum completely is dangerous because errors in network or transport layer headers can cause packets to be misdelivered; guarding against this kind of error is an important function of the transport layer.

UDP Lite is designed to provide a partial checksum that only covers as much of the user data that the sending application specifies as necessary. Errors in the rest of the packet are ignored because they are assumed to be acceptable for the destination application. This increased flexibility is achieved while maintaining the simplicity of UDP. UDP Lite can be easily integrated into or derived from most existing UDP implementations.

With a partial checksum such as the one provided by UDP Lite, it is undesirable that link layers drop packets due to errors that are acceptable according to the sender. We have investigated two ways to reflect the UDP Lite policy onto the link layer.

The paper is organized as follows. First, we present the UDP Lite protocol in section 2. In that section we also present ways to reflect the UDP Lite policy onto the link layer. A brief description of measurements of UDP traffic is presented in section 3, together with comments on the results. A brief discussion of how different applications could benefit from UDP Lite is presented in section 4. Finally, conclusions are presented in section 5.

2 *UDP Lite*

2.1 BASIC DESIGN

When designing a network protocol, there is a tradeoff between simplicity, flexibility and optimality. One reason for using UDP for delay-sensitive applications is the low protocol overhead. Adding new functionality for in-order delivery or error recovery would most likely increase delay. Therefore, the design of UDP Lite is focused on increasing the flexibility of classic UDP while preserving its simplicity.

The motivation behind UDP Lite is the hypothesis that with an increasing number of real-time applications that use UDP, the number of packets dropped due to a few single-bit errors that might be acceptable for the destination application will also increase. Applications with real-time audio/video data transmissions often use coding algorithms where some bit errors in the data are preferred over a lost packet. Therefore, it would be best if the data validation mechanism did not drop packets because of a few acceptable bit errors. This requires that packets are divided into *sensitive* and *insensitive* parts. Errors in the sensitive part of a packet should result in dropped packets, while errors in the insensitive part should not.

By using a *partial checksum* that only covers the sensitive part of a packet, this policy can be achieved. Since the receiver will only calculate the checksum over the sensitive data, errors in the insensitive part will be ignored. The amount of sensitive data in a packet is specified by the sending application in the UDP Lite header. To avoid complexity, the protocol requires that the sensitive data in a packet start at the beginning. With this requirement, the only new information needed in the header is how many bytes at the beginning of the packet are sensitive to errors. This new design has the effect of allowing the next-level protocol above UDP, e.g., RTP, the Internet Real Time Protocol [7], to have its header checksummed without having to checksum the RTP user data.

This means that the only difference between classic UDP and UDP Lite is that the UDP Lite header carries information about how many bytes from the beginning of the packet are included in the checksum calculation. As shown in figure 1, there is redundant information about the packet length in the UDP headers. The UDP Length field in the IP pseudo-header is calculated by subtracting the size of the IP header[2] from the packet length field in the IP header. The Length field in the UDP header is the length of the UDP header plus the UDP payload. The UDP Length and the Length fields mismatch only when there is padding after the UDP payload.[3] By replacing the Length field with a *Coverage* field, we obtain the headers shown in figure 2.

The Coverage field specifies how many bytes, starting from the first byte of the UDP Lite header, are sensitive

---

[1] In IPv6 [8] however, the UDP checksum is mandatory since there is no IP header checksum.

[2] For IPv6, the sizes of extension headers are also subtracted.

[3] Our investigations show that packets with padding after the UDP payload are rare. Such padding does not fill any purpose.

<table>
<tr><td rowspan="3">Pseudo header</td><td colspan="3">Source address</td></tr>
<tr><td colspan="3">Destination address</td></tr>
<tr><td>Zero</td><td>Proto</td><td>UDP Length</td></tr>
<tr><td rowspan="2">UDP Lite header</td><td colspan="2">Source port</td><td>Destination port</td></tr>
<tr><td colspan="2">Coverage</td><td>Checksum</td></tr>
</table>

Figure 2: The UDP Lite headers

to errors. In addition to this, the UDP Lite header and the IP pseudo-header are always verified by the checksum, which means that the least acceptable value of the coverage field is eight (the number of bytes in the UDP Lite header). With a checksum coverage value equal to the packet length, UDP Lite packets are treated just as classic UDP packets with the checksum enabled.

2.2 THE LINK LAYER

The key feature of UDP Lite is the partial checksum. It allows senders to specify packets as partially insensitive to errors. It can be argued, however, that the link layer will already have dropped such packets before they are received by the UDP Lite protocol. A response to this argument is that Internet traffic studies have shown that many errors are generated inside nodes rather than during transmission.

An exception to this reasoning is that some network links are naturally lossy; some wireless links cannot be made reliable due to delay and spectrum efficiency considerations. For these kinds of links, the UDP Lite policy with a partial checksum can make a difference to the application. We will examine two approaches to how this can be achieved.

2.2.1 A PARTIAL LINK LAYER CHECKSUM

The most straight-forward solution would be to have a partial checksum at the link layer for each hop along the path between two endpoints. This is clearly the architecturally cleanest alternative. However, as it requires replacement of, or changes to, several existing standards for link layer framing it is not feasible as a short-term solution. Changing the link-layer standards and deploying these new standards will take a long time if at all possible. Moreover, there is little gain from doing this over links with low error rates.

For links known to cause many errors, normally detected by link layer protocols, one should design a new link layer protocol where frames carry a partial checksum. Such would be the case for wireless LANs or cellular radio networks with a high ratio of real-time flows. Coverage by the link layer checksum could be determined by peeking at the checksum coverage field in the UDP Lite header when sending a frame carrying a UDP Lite packet.

2.2.2 IGNORING LINK LAYER CHECKSUM ERRORS

The most promising short-term approach, we think, is not to have a partial checksum at the link layer level, but to modify the device driver to ignore checksum errors for incoming frames carrying UDP Lite packets. This modified driver can be installed on arbitrary computers in the network without disturbing other nodes. If this modification is implemented in a gateway for a lossy network, errors will not cause packets to be dropped by the gateway. Instead, damaged packets will traverse the network to the receiving host, where they are dropped if the errors were in the part of the packets specified as sensitive. To use this approach, there must be a way for the hardware to ignore link layer checksum errors. Currently, there are several hardware devices for various link layer types that support this.

We believe that ignoring the link layer checksum is the best short-term approach for reflecting the UDP Lite policy onto the link layer without introducing new link-layer framing. It can be deployed at strategic places in the network without requiring changes to other computers.

3 *Measurements*

To investigate the need for UDP Lite, we have performed two types of tests: *passive* and *active* tests. During the passive tests, UDP traffic was tapped out of a shared network segment by the *tcpdump* application. In the active tests, well-known packet sequences were sent to a specific receiver. In both cases, the traffic was stored in trace files for later analysis.

3.1 PASSIVE TESTS

The analysis of the passive tests shows how often UDP packets will be dropped by the receiving host due to a failing checksum. There were 3 passive test sequences:

- 1,500,000 UDP packets that have travelled at least 10 hops each.
- 3,500,000 UDP packets from a university subnet with ~350 connected computers.
- 10,000,000 UDP packets from a city network with about 200 home users connected through ADSL and cable modems.

The results from analysing these test sequences show that roughly 0.8%, or 1 out of 125, of the UDP packets that reach the destination will be discarded due to a failing checksum. A closer study of the traces shows that the majority of the UDP traffic in the first case comes from streaming multimedia with its source in the US. For the last two traces, most of the errors – around 95% - are related to multiplayer games. In both cases, the errors detected belong to a low number of flows – typically less than 3.

This is an indication of the frequency of erroneous packets that are detected by the Internet checksum used in UDP. Packets lost along the way are not shown in this analysis, nor are the error patterns in the packets. These limitations are the motivation for the active tests.

3.2 Active tests

The traffic sent in these tests was generated by a random traffic generator which given a seed value generates a sequence of packets which vary in size and content. For each test sequence, 100,000 packets with a size varying from 120 to 640 bytes were sent from a sender to a receiver at a rate of 20 packets per second (50ms between the start of each packet). At the receiving side, the traces were compared to the packet sequences generated. The test sequences were as follows:

- Between two computers on the same Ethernet segment with no background traffic.
- Between two computers interconnected via ADSL modems over a 1 km copper wire.
- Between two computers on the same cable modem segment with no background traffic.
- Between two computers connected to the Internet via Ethernets. There are 6 routers between the two computers.
- Same as above, but with 13 routers between the two computers.

The results from these test sequences can be summarized as follows. The number of erroneous packets delivered to the receiver does not increase with distance. The first test sequence with two computers on the same Ethernet segment experienced 0.70% erroneous packets while the last test sequence with 13 routers between the endpoints experienced 0.73% erroneous packets. The number of lost packets on the other hand increased from 1 out of 12,500 to 1 out of 7,200. Reordering of packets never occurred in any test sequence. The Internet checksum never failed in detecting erroneous packets. When errors occurred, there were mostly single-bit errors. At most, 4 consecutive bits were inverted. It never happened that two consecutive packets contained errors.

3.3 Comments to the results

It is not uncommon that the kind of applications mentioned in section 3.1 have a sending rate of 25-30 packets per second. Given that 1 out of 125 packets are erroneous, packets will be dropped every 4-5 seconds. Since different applications suffer differently from packet losses, a general statement about the harm caused by this kind of loss can not be easily made.

4 *Usage of UDP Lite*

Much research has gone into providing application reliability above UDP. Examples include new coding schemes, application-specific protocols, forward error correction, statistical studies of UDP traffic and more. UDP Lite should be added to this list, but it takes a different approach. Instead of trying to compensate for packet loss due to errors, we assume that many are acceptable in real-time usage scenarios and instead add a mechanism to let the sending application specify which part of a packet is sensitive and which is not. In this section, we briefly discuss how different types of coding schemes could benefit from UDP Lite.

4.1 Progressive coding schemes

In progressive coding schemes, data is divided into multiple segments. First a segment with a coarse description is transmitted, followed by segments with increased level of detail. For video codings, you could first transmit a blurry image and then let the following data refine it until the entire picture has the correct resolution. Audio codings can first transmit the amplitude for chosen key frequencies and then refine the sound by transmitting side frequencies. One example of an application that uses this kind of coding scheme is the popular RealPlayer [4] used for receiving streamed video and/or audio.

With UDP Lite, the sending application can choose to checksum the first coarse description while letting the following segments pass unverified. Errors at the receiving side appear as errored pixel segments in a video picture or sound glitches in audio transmissions. Without UDP Lite, errors would most likely cause a greater amount of data to be lost since whole packets are discarded.

4.2 PCM AUDIO

PCM codings are popular for audio conferencing and IP telephony. A common configuration is a sample frequency of 8 kHz with 40 ms sound clips. This will generate packets with a payload of 640 bytes, headers excluded. Assuming that the RTP protocol [7] is used, an extra 12-bytes header is used. With classic UDP, the entire packet is verified which gives a total of 672 bytes. Single-bit errors that result in dropped packets will be handled in different ways depending on the application. Some replace the lost sound clip with 40 ms silence, while others insert noise or repeat the last sample. For the PCM case, the best replacement for a damaged sound clip is usually the damaged sound clip itself. UDP Lite can be used to verify only the protocol headers while accepting errors in the sound data. The headers are in total 32 bytes, which means that only 4.7% of the entire packet will be validated by the checksum. This reduces the risk of lost packets significantly, especially in environments with high error rates, such as some wireless environments.

4.3 OTHER EXAMPLES

MPEG video codings send data using three different frame types; I-, P- and B-frames. I-frames hold information about an entire video frame, while P- and B-frames only include the differences to other frames. Loss of I- or P-frames will affect other frames as well. Usually, it is better to deliver damaged P- and B-frames than discarding them. UDP Lite can be used to implement this.

There are of course many coding algorithms that do not cope well with single-bit errors. For example, when the data is heavily compressed, single-bit errors are normally harder to accept.

5 *Conclusions*

We have described UDP Lite, a new transport protocol that is a more flexible version of the commonly used UDP. Its main feature is its ability to divide the packet into two parts; one that is more sensitive to errors and one that is less sensitive. Using this mechanism, the sending application can specify that errors are acceptable in part of the data payload in order to reduce the number of unnecessarily discarded packets.

To avoid frames carrying UDP Lite packets from being discarded by the link layer due to errors in the insensitive part of the packet, we suggest as a short-term solution that the receiving network interface ignores link layer checksum errors for frames with UDP Lite packets. Damaged packets will be delivered to UDP Lite, which then decides whether the packet should be discarded or not.

For links that can deliver a significant ratio of damaged packets, it is important to reflect the UDP Lite policy of sensitive and insensitive data onto the link layer. Designs for new link layer protocols for such links should allow checksums to be partial; the insensitive part of the payload should *not* be covered by a checksum.

5.1 CURRENT AND FUTURE WORK

In addition to the analysis of the test data, we have two independently implemented prototypes of the UDP Lite protocol, in NetBSD and FreeBSD respectively. A modified version of VAT [3] with UDP Lite support has also been implemented. After implementing modified device drivers for various interfaces, we have a testbed which will be used to evaluate the end-user experience within various network environments.

Further, we plan to investigate whether the Internet checksum used by UDP is the best choice with a possibly disabled link layer checksum. We will see if there is a checksum with stronger error detection properties than the Internet checksum and small additional computational overhead.

References

[1] Fluckiger, F., *Understanding Networked Multimedia – Applications and Technology*, Prentice-Hall 1995, ISBN 0-13-190992-4

[2] Goralski, W., *ADSL and DSL Technologies*, McGraw-Hill 1998, ISBN 0-07-024679-3

[3] Jacobson, V., McCanne S., *Vat – X11-based audio conferencing tool*, Lawrence Berkeley Laboratory, University of California, Berkeley.

[4] *RealNetworks homepage*, URL: http://www.real.com/.

[5] Postel, J., *User Datagram Protocol*, Internet Request for Comments RFC 768, August 1980.

[6] Braden, R., Borman, D., and Partridge, C., *Computing the Internet checksum*, Internet Request for Comments RFC 1071, September 1988.

[7] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V., *RTP: A transport protocol for Real-Time Applications*, Internet Request for Comments RFC 1889, January 1996.

[8] Deering, S., Hinden, B., *Internet Protocol, version 6 (IPv6) Specification*, Internet Request For Comments RFC 1883, December 1995.