The UDP Lite Protocol
<draft-larzon-udplite-02.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of [RFC-2026].

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress.''

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This document describes the UDP Lite Protocol, which is similar to classic UDP [RFC-768], but aimed at applications which can handle a partially damaged payload in lossy network environments. If this feature is not used, it is semantically identical to classic UDP.

Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC-2119].
Introduction

The UDP Lite protocol is designed to provide flexible checksumming policies for applications that today use UDP (hereafter referred to as "classic UDP"). UDP Lite is particularly useful for real-time multimedia applications sending data over links with high bit-error rates, e.g., many kinds of wireless links.

This is a class of applications with delay bounds that leave little time to repair damage caused by transmission errors over the link. To meet low-delay requirements, classic UDP is usually used since it has no overhead for retransmission of erroneous packets, in-order delivery or error correction.

To make better use of bandwidth in high-error environments, many codecs for voice and video are designed to cope better with errors in the data payload than with loss of entire packets. However, no errors are allowed in the protocol headers. The checksumming policy of classic UDP, where the checksum either covers the entire datagram or nothing at all, is therefore not adequate. Moreover, in next version of IP, IPv6 [RFC-2460], the UDP checksum is mandatory and can not be disabled for error-tolerant applications. This will lead to a high packet loss rate in network environments with high bit error rates.

A transport protocol is needed where a checksum can protect vital information such as headers and can optionally ignore errors best dealt with by the application. The decision of what must be verified by the checksum is best specified by the sending application.

UDP Lite provides an optional partial checksum. Each packet can, but must not, be divided into a sensitive and an insensitive part. The checksum only covers the sensitive part. Errors in the insensitive part will therefore not cause packets to be discarded.

Compared to classic UDP, the partial checksum provides extra flexibility for applications with partially insensitive data. If the checksum is configured to cover the entire packet (which is recommended to be the default), UDP Lite is semantically identical to classic UDP, provided there is no padding of the IP datagram after the UDP payload. We recommend this to be the default.
Protocol description

The UDP Lite header is shown in figure 1. Its format differs from classic UDP in that the UDP Length field has been replaced with a Checksum Coverage field. This can be done since information about the UDP Lite packet length can be found in the length field of the IP pseudo-header.

```
+--------+--------+--------+--------+
|     Source      |   Destination   |
|      Port       |      Port       |
|--------+--------+--------+--------+
|    Checksum     |                 |
|    Coverage     |    Checksum     |
|--------+--------+--------+--------+
|                                   |
|           data bytes ...          |
|---------------- ...---------------+
```

Figure 1: UDP Lite Datagram Header Format

Fields

The fields ‘‘Source Port’’ and ‘‘Destination port’’ are defined as in [RFC-768].

Checksum Coverage is the number of bytes, counting from the first byte of the UDP Lite header, that are covered by the checksum. The UDP Lite header MUST always be included in the checksum. Despite this requirement, the Checksum Coverage is expressed in bytes from the beginning of the UDP Lite header in order to preserve compatibility with classic UDP. A Checksum Coverage of zero indicates that the entire UDP Lite packet is included in the checksum. This means that the value of the Checksum Coverage field MUST be either zero or at least eight.

Checksum is the 16-bit one’s complement sum of a pseudo-header of information from the IP header, the number of bytes specified by the Checksum Coverage (starting at the first byte in the UDP Lite header), virtually padded with zero bytes at the end (if necessary) to make a multiple of two bytes. If the computed checksum is zero, it is transmitted as all ones (the equivalent in one’s complement arithmetic). The transmitted checksum MUST NOT be zero.

UDP Lite uses the same conceptually prefixed pseudo header from the IP layer as classic UDP for checksumming purposes. The length of the UDP Lite packet is the value of the length field in the pseudo header. The format of the pseudo header differs for different versions of IP.
User Interface

A user interface should allow the same operations as for classic UDP. In addition to this, it SHOULD provide a way for the sending application to pass the checksum coverage value to the UDP Lite module.

We RECOMMEND that the default behaviour of UDP Lite is to mimic classic UDP by verifying the entire packet. Applications that want to define the payload as partially insensitive to bit errors SHOULD do that by a separate system call.

IP Interface

As for classic UDP, the IP module must pass the pseudo header to the UDP Lite module.

The IP layer MUST NOT pad the IP payload with extra bytes since the length of the UDP Lite payload delivered to the receiver depends on the length passed in the pseudo header.

UDP Lite and different versions of IP

For IP version 4 (IPv4), the classic UDP protocol is too well-established and widely spread to be replaced; UDP Lite could only be deployed as a separate protocol with its own protocol ID.

For IP version 6 (IPv6), it can be argued that classic UDP for IPv6 can be replaced by UDP Lite since a UDP Lite packet with a Checksum Coverage equal to the packet length is semantically identical to a classic UDP packet. UDP Lite MUST, however, have a protocol ID different from the one of classic UDP to support communication with IPv4 nodes.

Link layer support

Since UDP Lite can deliver packets with damaged payload to an application, frames carrying UDP Lite packets should not be discarded by a link layer checksum. A long term solution requires that the partial checksum policy of a UDP Lite packet should be reflected down, somehow, to the link layer.

Conclusions

We have presented the UDP Lite protocol. The main motivation for this new transport protocol is decreased packet error rates for real-time applications today using classic UDP in wireless network environments. UDP Lite provides a partial checksum which increases the
flexibility of classic UDP by making it possible to define a packet as partially insensitive to bit errors on a per-packet basis. If no part of a packet is defined as insensitive, UDP Lite is semantically identical to classic UDP. Due to this similarity between classic UDP and UDP Lite, we argue that classic UDP for IPv6 could be replaced by UDP Lite.

Contact info

Lars-Ake Larzon  
Department of CS & EE  
Lulea University of Technology  
S-971 87 Lulea, Sweden  
Email: lln@cdt.luth.se

Mikael Degermark  
Department of CS & EE  
Lulea University of Technology  
S-971 87 Lulea, Sweden  
Email: micke@cdt.luth.se

Stephen Pink  
Department of CS & EE  
Lulea University of Technology  
S-971 87 Lulea, Sweden  
Email: steve@cdt.luth.se

References


This draft expires September 10, 2000