INTERNET-DRAFT
TSV WG
January 24, 2002
Expires: July 24, 2002

Lars-Ake Larzon
Lulea University of Technology, Sweden
Mikael Degermark
Stephen Pink
The University of Arizona, USA

The UDP Lite Protocol
<draft-ietf-tsvwg-udp-lite-00.txt>

Status of this Memo

Please direct comments to the TSV WG mailing list: tsvwg@ietf.org

Abstract

This document describes the UDP Lite protocol, which is similar to classic UDP [RFC-768], but can also serve applications which in lossy network environments prefer to have partially damaged payloads delivered rather than discarded. If this feature is not used, UDP Lite 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

Why another transport protocol?

First, there is a class of applications which prefer to have damaged data delivered rather than discarded by the network. A number of codecs for voice and video fall into this class. These codecs are designed to cope better with errors in the payload than with loss of entire packets.

Second, there are a number of link technologies where data can be partially damaged. Several radio technologies exhibit this behavior when operating at a point where cost and delay is sufficiently low.

Third, intermediate layers should not prevent such applications to run well over such links. The intermediate layers are IP and the transport layer. IP is not a problem in this regard since the IP header has no checksum which covers the IP payload. The generally available transport protocol best suited for these applications is UDP, since it has no overhead for retransmission of erroneous packets, in-order delivery or error correction. However, the UDP checksum either covers the entire datagram or nothing at all. Moreover, in the next version of IP, IPv6 [RFC-2460], the UDP checksum is mandatory and must not be disabled. The IPv6 header does not have a header checksum and it was deemed necessary to always protect the IP addressing information by making the UDP checksum mandatory.

A transport protocol is needed that conforms with the properties of link layers and applications described above. The error-detection mechanism of the transport layer must be able to protect vital information such as headers, but also to optionally ignore errors best dealt with by the application. What should be verified by the checksum is best specified by the sending application.

UDP Lite provides a checksum with optionally partial coverage. When using this option, a datagram is divided into a sensitive part (covered by checksum) and an insensitive part (not covered by checksum). Errors in the insensitive part will not cause the datagram to be discarded. When the checksum covers the entire datagram, which SHOULD be the default, UDP Lite is semantically identical to UDP.

Compared to UDP (hereafter referred to as "classic UDP"), the partial checksum provides extra flexibility for applications with partially insensitive data.

Protocol description

The UDP Lite header is shown in figure 1. Its format differs from classic UDP in that the Length field has been replaced with a Checksum Coverage field. This can be done since information about UDP packet length can be provided by the IP module in the same manner as for TCP [rfc-793].

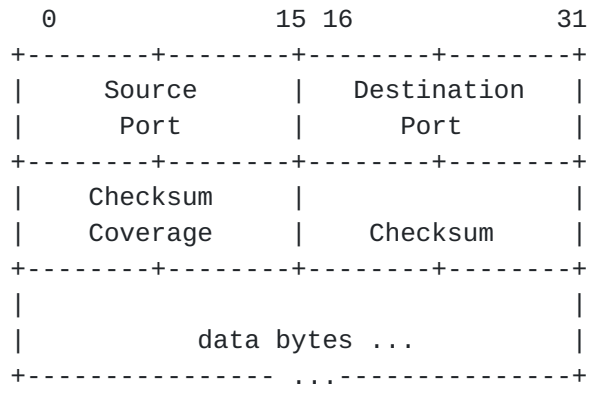

Figure 1: New UDP Header Format

Fields

The fields ``Source Port'' and ``Destination port'' are defined as in the UDP specification [RFC-768].

Checksum Coverage is the number of bytes, counting from the first byte of the new UDP 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 new UDP 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 of the 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 new UDP 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 all zeroes.

Pseudo header

Similar to classic UDP, UDP Lite uses a conceptually prefixed pseudo header from the IP layer for checksumming purposes. The format of the pseudo header is the same as for classic UDP, and differs for different versions of IP. The pseudo header of UDP Lite is different from the pseudo header of classic UDP in one way: The value of the length field of the pseudo header is not taken from the UDP Lite header, but rather from information provided by the IP module. This computation is done in the same manner as for TCP [RFC-793], and implies that the length field of the pseudo header includes the UDP Lite header and all subsequent bytes in the IP payload.

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. There SHOULD also be a way to pass the checksum coverage value to the receiving application, or at least let the receiving application block delivery of packets with coverage values less than a value provided by the application.

We RECOMMEND that the default behaviour of UDP Lite is to mimic classic UDP by having the coverage field match the length of the UDP Lite datagram and 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 on the sender side. Applications that wish to receive payloads which were only partially covered by a checksum SHOULD inform the system by a separate system call.

IP Interface

As for classic UDP, the UDP Lite module must pass the pseudo header to the UDP Lite module. The UDP Lite pseudo header contains the IP addresses and protocol fields of the IP header, and also the length of the IP payload which is derived from the length field of the IP header.

The IP module 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 of the IP payload.

Lower layer considerations

Since UDP Lite can deliver packets with damaged payloads to an application that wants them, frames carrying UDP Lite packets need not be discarded by lower layers when there are errors only in the

insensitive part. For a link layer that supports partial error detection, the Coverage field in the UDP Lite header MAY be used as a hint of where errors should be detected. Link layers that do not support partial checksums SHOULD detect errors in the entire frame. In general, lower layers SHOULD detect errors at least in the sensitive part of the frame using strong error detection mechanisms, but need not do so for the insensitive part.

Note that it is usually only over links where errors are frequent that the partial checksum feature of UDP Lite can make a difference to the application. On links where errors are infrequent it is RECOMMENDED that lower layeers detect errors in the entire packet.

Jumbograms

The Checksum Coverage field is 16 bits and can represent checksum coverage up to 65535 octets. This allows arbitrary checksum coverage for IP datagrams, unless they are Jumbograms. For Jumbograms, the Checksum can cover either the entire payload (when the Checksum Coverage field has the value zero), or else at most the initial 65535 octets of the UDP Lite datagram.

Backwards compatibility

The syntactic and semantic similarity between UDP Lite and classic UDP suggests that they might share the same protocol identifier. This section explores some consequences of doing so.

There are no known interoperability problems between classic UDP and UDP Lite if they were to share the protocol identifier of classic UDP. To be more precise: there is no case where a potentially problematic packet is delivered to an unsuspecting application; a UDP Lite payload with partial checksum coverage cannot be delivered to UDP applications, and UDP datagrams which only partially fills the IP payload cannot be delivered to UDP Lite applications.

If the protocol identifier was shared between UDP and UDP Lite and a UDP Lite implementation sends UDP Lite packets with partial checksums to a classic UDP implementation, the classic UDP implementation would silently discard them because a mismatching pseudo header would cause the UDP checksum to mismatch. Neither the sending nor the receiving application would be notified. The obvious solutions to this include

1) explicit application in-band signaling (not using the partial checksum coverage option) to enable the sender to learn whether the receiver is UDP Lite enabled or not, or

2) use of out-of-band signaling such as H.323, SIP, or RTCP to convey

whether the receiver is UDP Lite enabled.

If UDP Lite has its own separate protocol identifier, on the other hand, a system unaware of UDP Lite would return ICMP Protocol Unreachable error messages to the sender. This simple method of detecting UDP Lite unaware systems is the primary benefit of having separate protocol identifiers.

Therefore, this draft proposes to allocate a new protocol identifier for UDP Lite.

Security considerations

The security impact of UDP Lite is twofold. First, applications who do not expect damaged payloads are bound to malfunction if damaged payloads are delivered to them. To avoid this, we RECOMMEND that the sending and the receiving side application both explicitly enable the partial checksum option. Packets with partial checksums SHOULD NOT be delivered to applications that have not enabled the partial checksum option.

Second, there is the question of how UDP Lite interacts with authentication and encryption mechanisms. When the partial checksum option of UDP Lite is enabled, it is fine with the application if the insensitive part of a packet changes in transit. This is contrary to the idea behind most authentication mechanisms; authentication succeeds when the packet has not changed in transit. Unless authentication mechanisms that operate only on the sensitive part of packets are developed, authentication will always fail on UDP Lite packets where the insensitive part has been damaged.

Encryption is also an issue when using UDP Lite. If a few bits of an encrypted packet are damaged, the decryption transform will typically spread this error so that the packet becomes too damaged to be of use. Most strong encryption transforms today exhibit this behaviour, for good reason. It might be possible to develop encryption transforms which would not spread damage in this way when the damage occurs in the insensitive part of the packet. A class of such transforms would be transforms where the sensitive part is encrypted using a strong transform as usual, and the insensitive part is encrypted by XORing it with a cryptographic hash computed over the cleartext of the sensitive part. However, it is clear that with most transforms in use today, encryption eliminates the benefits that the partial checksum coverage option of the UDP Lite might bring.

IANA considerations

We request that a new ip protocol identifier is allocated for UDP

Lite.

Conclusions

We have presented the UDP Lite protocol. The main motivation for this new variant of the classic UDP transport protocol is decreased packet error rates for damage-tolerant applications today using classic UDP in harsh network environments. UDP Lite provides optionally partial checksum coverage 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.

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 Computer Science
The University of Arizona
P.O. Box 210077
Tucson, AZ 85721-0077
Email: micke@cs.arizona.edu

Stephen Pink
The University of Arizona
P.O. Box 210077
Tucson, AZ 85721-0077
Email: steve@cs.arizona.edu

Normative References

[RFC-768] Postel, J., "User Datagram Protocol," RFC 768, Information Sciences Institute, August 1980.

[RFC-793] Postel, J., "Transmission Control Protocol," RFC 793, Information Sciences Institute, September 1981.

[RFC-2460] Deering, S., Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, IETF, December 1998.

Informative References

[RFC-2026] Bradner, S., "The Internet Standards Process," RFC 2026, Harvard University, October 1996.

[RFC-2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," Harvard University, March 1997.