Internet Engineering Task Force	G. Fairhurst
Internet-Draft	University of Aberdeen
Intended status: Informational	February 10, 2010
Expires: August 14, 2010	

TOC

The UDP Tunnel Transport mode
draft-fairhurst-6man-tsvwg-udptt-03

Abstract

This document proposes a standards track protocol called the UDP Tunnel Transport. This protocol updates the UDP processing of RFC 2460 for IPv6 hosts and routers. The update enables a sender to generate a UDP datagram where the UDP checksum is replaced by a header check determined only by the protocol header information. For this use, the document also updates the way the IPv6 UDP length field is interpreted. This mode is intended to minimise the processing cost for the transport of tunnel packets using UDP.

Status of this Memo

Copyright Notice

Table of Contents

---

1. Introduction

TOC

The UDP Tunnel Transport (UDPTT) is a protocol that updates the User Datagram Protocol (UDP) processing of RFC2460 (Deering, S. and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” December 1998.) [RFC2460] for IPv6 hosts and routers. UDPTT is intended to transport datagrams that carry tunnel-encapsulated packets. It is

not intended as a general purpose transport, since it is applicable only for cases where the tunnel application can provide a set of checks on the correctness of the received payload.
A UDPTT end point may be either a host or a router. The tunneling protocol introduces a header check that validates the delivery of the packet to the correct transport endpoint. This check is not intended as an authentication check (in the manner of a security protocol), but is introduced to reduce the probability that the endpoint stacks receive erroneous packets that may corrupt internal state, introduce unnecessary packet processing, or lead to ambiguous packet counts.
The way in which the header check is computed in UDPTT will usually result in a constant value for each UDPTT flow. This value may be cached as a part of the tunnel endpoint flow state. Once the tunnel has been created, this requires a receiver to perform a 16-bit comparison operation, rather than a 1's complement checksum. This approach was driven by a desire to eliminate expensive computation in routers that may need to handle many flows operating at high rate.
The next section provides background information on UDP variants and the use of UDP for tunneling. Section 2 defines the UDPTT protocol and section 3 provides information about the use of UDPTT.

1.1. Background

TOC

UDP is defined in [RFC0768] (Postel, J., “User Datagram Protocol,” August 1980.). This supports two checksum behaviours when used with IPv4. The normal behaviour is for the sender to calculate a checksum over a block of data that includes a pseudo header and the UDP datagram payload. The receiver validates this checksum to verify delivery to the intended transport endpoint.
The UDP header includes a 16-bit one's complement checksum that provides a statistical guarantee that the payload was not corrupted in transit. It also allows the receiver to verify that the endpoint was the intended destination of the datagram, because it includes a pseudo header that covers the IP addresses, port numbers, transport payload length, and Next Header/Protocol value corresponding to the UDP transport protocol. The length field verifies that the datagram is not truncated or padded. The checksum therefore protects an application against receiving corrupted payload data in place of, or in addition to, the data that was sent. Applications are recommended to enable UDP checksums [RFC5405] (Eggert, L. and G. Fairhurst, “Unicast UDP Usage Guidelines for Application Designers,” November 2008.), although UDP (Postel, J., “User Datagram Protocol,” August 1980.) [RFC0768] permits the option to be disabled when used with IPv4.
Unlike IPv4, when UDP datagrams are originated by an IPv6 node, the UDP checksum is not optional. The use of the UDP checksum is required when applications transmit UDP over IPv6 [RFC2460] (Deering, S. and R.

Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” December 1998.), since there is no network-layer integrity check. UDPTT provides an alternative intended to achieve at least equivalent protection to using IPv4 (with the associated header checksum) and UDP (with the checksum disabled).

UDP-Lite (Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, “The Lightweight User Datagram Protocol (UDP-Lite),” July 2004.) [RFC3828] provides a checksum with an optional partial coverage. When using this option, a datagram is divided into a sensitive part (covered by the checksum) and an insensitive part (not covered by the checksum). Errors in the insensitive part will not cause the packet to be discarded by the transport layer at the receiving host. When the checksum covers the entire packet, which should be the default, UDP-Lite is semantically identical to UDP. UDP-Lite is specified for use with IPv4 and IPv6, and uses an IP protocol type (or IPv6 next header) with a value of 136 decimal.

While UDP-Lite benefits from differential link error treatment, where the packet header is afforded higher protection on a radio link compared to the payload, this is explicitly not the goal of UDPTT. For UDPTT, the payload is expected to be protected by other integrity checks, and generally all parts of the packet will seek equal protection, as for UDP and TCP. Since UDP-Lite also includes the total packet length (extracted from the IP module), the calculated checksum depends on the size of the encapsulated packet, whereas in UDPTT, the checksum protection does not validate the actual size of the transport layer payload.

1.2. Use of UDP Tunnels

TOC

One increasingly popular use of UDP is as a tunneling protocol, where a tunnel endpoint encapsulates the packets of another protocol inside UDP datagrams and transmits them to another tunnel endpoint. Using UDP as a tunneling protocol is attractive when the payload protocol is not supported by middleboxes that may exist along the path, because many middleboxes support transmission using UDP. In this use, the receiving endpoint decapsulates the UDP datagrams and forwards the original packets contained in the payload [RFC5405] (Eggert, L. and G. Fairhurst, “Unicast UDP Usage Guidelines for Application Designers,” November 2008.). Tunnels establish virtual links that appear to directly connect locations that are distant in the physical Internet topology and can be used to create virtual (private) networks.

This is expected to be the normal use of UDPTT, where UDPTT may replace UDP as the tunnel transport when there is a desire to reduce processing costs at the tunnel endpoints. The end point for the UDPTT may be either a host or a router.

{Note: The current specification targets use with IPv6, however the method may also be applicable to IPv4}

---

2. Update to RFC 2460 to support UDTT

TOC

This section defines the update to IPv6 [RFC2460], if this document is approved for publication by the IETF.

2.1. Terminology

TOC

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 [RFC2119] (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.).

2.2. UDPTT Next Header Value

TOC

UDPTT datagrams are carried in the payload of IPv6 packets. UDP and UDPTT share the next header protocol number (decimal 17) and are differentiated only by the Length of the IP payload.

2.3. UDPTT Header Format

TOC

The UDPTT header is shown in figure Figure 1 (UDPTT Header Format). This format resembles that of UDP.

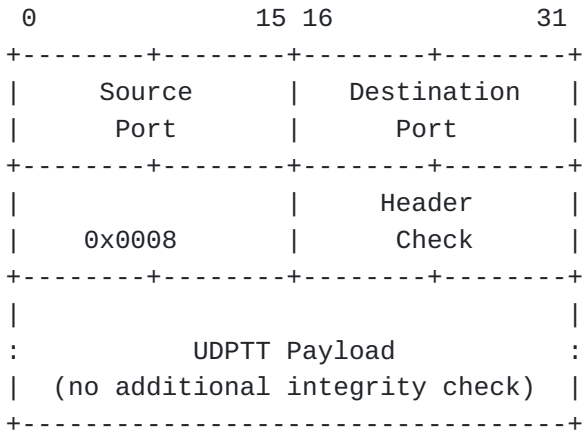


Figure 1: UDPTT Header Format

The source and destination ports are used in the same way as for UDP and UDP-Lite. UDPTT places the constant value 0x0008 in the position occupied by the Length field in UDP and the Checksum Coverage Field in UDP-Lite. The value of 0x0008 is legal for both UDP and UDP-Lite.
The length of the payload part is determined from the size information provided by the IP module in the same manner as for TCP (Postel, J., "Transmission Control Protocol," September 1981.) [RFC0793].
The Header Check field is a 16-bit value calculated as specified in the next section. This value is set by the sender and validated by the receiver.

2.4. UDP and UDPTT Datagrams with no payload

TOC

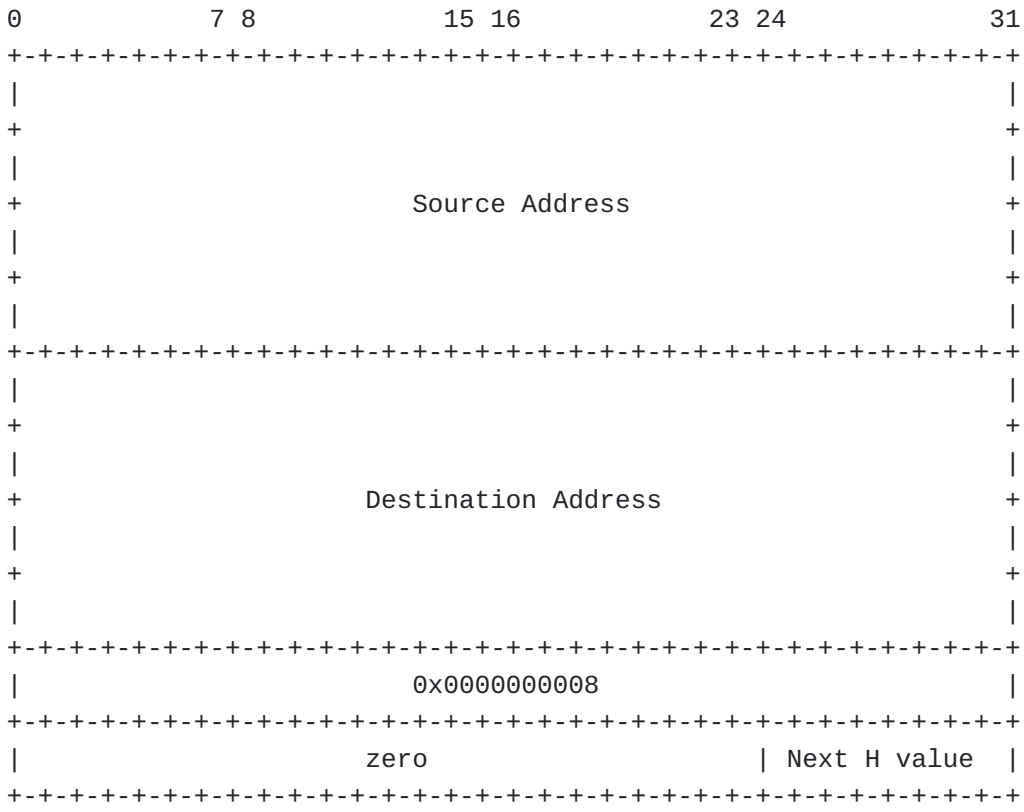
It is expected that UDPTT datagrams will carry a tunnel-encapsulated packet as payload. A UDPTT datagram with no payload is indistinguishable from a UDP datagram with no payload. Both have the same representation on the wire, and the same semantics at the sender and receiver. There is no need for a receiver to differentiate these packets.

2.5. Calculation of IPv6 UDPTT Header Check

TOC

The Header Check is computed as the 16-bit one's complement of the one's complement sum [RFC1071] (Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum," September 1988.) of

a pseudo-header of information collected from the IPv6 and UDPTT header fields.
The following illustration shows the UDPTT pseudo-header for IPv6:



UDPTT Pseudo header fields

*As in UDP, if the IPv6 packet carrying the UDPTT datagrams also carries an additional extension header that contains a Routing header, the Destination Address used in the pseudo-header is that of the final destination. At the originating node, that address will be in the last element of the Routing header; at the recipient(s), that address will be in the Destination Address field of the IPv6 header [RFC2460] (Deering, S. and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” December 1998.).

*The pseudo header of UDPTT is different from the pseudo header of UDP in one way: This pseudo header replaces the Upper-Layer Packet Length field, with a constant of 8. This value is identical to the Upper-Layer Packet Length field that would be

returned by a compliant IPv6 UDP stack with no transport-layer payload [RFC2460] (Deering, S. and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” December 1998.). (It differs from the value that would have been used by UDP-Lite, which utilises the length reported by the IP Module in the pseudo header). Encapsulated packets need to include their own methods to verify integrity and correct payload length.

*The Next H value in the pseudo-header is the value specified for UDP (17 decimal). This value will differ from the Next Header value in the IPv6 header if there are extension headers between the IPv6 header and the upper-layer header.

Prior to computation, the Header Check field MUST be set to zero. If the computed checksum is zero, it is transmitted as all ones (the equivalent in one's complement arithmetic) [RFC2460] (Deering, S. and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” December 1998.) specifies that IPv6 receivers must discard UDP datagrams containing a zero checksum, and should log the error. This processing is preserved in this update.
The way in which the Header check is computed in UDPTT will usually result in a constant value for each UDP flow. This value may be cached as part of the tunnel endpoint flow state. Once the tunnel has been created, a sender MAY insert the cached value instead of computing the checksum, and a receiver may then use a 16-bit comparison of the received value against the cached value, rather than a 1's complement checksum. This approach may be desirable to minimise expensive computation in routers that need to handle many UDPTT flows operating at high rate.

2.6. Multicast support for UDPTT

TOC

Like UDP and UDP-Lite, UDPTT MAY be used as a transport for multicast datagrams.

---

3. Using UDPTT

TOC

This section provides information for implementors and users of UDPTT.

TOC

3.1. UDPTT Usage Guidelines

Implementors may use UDPTT in the same way as UDP providing that the application does not need UDPTT to validate the tunnel-encapsulated packet. The protocol is not constrained to the semantics of one particular tunnel usage, and is believed compatible with a range of tunnel mechanisms. If the tunnel requires greater assurance that tunnel-encapsulated packet is correct or has been delivered to the correct end point (e.g. where control data is carried over UDPTT), then the tunnel encapsulation MUST introduce its own integrity checks. This is consistent with the expected behaviour of IETF-defined tunnel encapsulations.

IPv6 Jumbograms are not supported in the UDPTT protocol. If required, such packets may be sent using UDP.

The UDP Usage Guidelines (Eggert, L. and G. Fairhurst, “Unicast UDP Usage Guidelines for Application Designers,” November 2008.) [RFC5405] provides guidance for application designers the use of UDP to support tunneling. These guidelines also apply to this protocol.

Unlike UDP, UDPTT does not validate the Length field of the IP header when calculating the transport checksum. This design decision was driven by the goal that the checksum value should not normally change from packet to packet within a single transport flow. The omission of this value is a relaxation of the integrity check. Therefore:

- A tunnel receiver MUST discard UDPTT packets where the UDPTT payload size is less than the minimum required by the tunnelled protocol being transported.

- Application stacks SHOULD provide a way for a tunnel endpoint to identify whether UDPTT is to be used. This could be identified by a socket option, or similar operating system mechanism. A sender uses the socket option to include data in a UDPTT datagram (beyond the base UDP header), the receiver uses this to ensure the protocol stack passes data based on the Upper Layer payload, rather than the UDP header).

3.2. Requirements for Tunnelled Protocols

TOC

This section identifies the requirements for protocols transported within the payload of a UDPTT datagram.

Specifically, these requirements dictate that:

- An inner IPv4 (or IPv6) packet with a UDP checksum equal to zero MUST NOT be tunneled.

*The tunneling protocol and implementation MUST NOT be used to transport IPv4 or IPv6 packets that use network-layer fragmentation.

*A receiver MUST check the size of the tunnel-encapsulated packet based on information contained in the tunnel-encapsulated packet. A tunnel receiver MUST discard any tunnel-encapsulated packets where the reported length of the tunnelled packet is different to that reported by the IP module (reduced by the size of any header extensions present).

*A packet encapsulated over UDPTT MAY also use the UDPTT tunnel encapsulation mode, that is, tunnels may be recursively encpauslated. However, the inner encapsulated packet MUST provide an integrity check of the transported payload information (e.g. the inner encapsulated IPv6 packet MUST NOT itself use UDPTT or be an IPv4 UDP datagram with the checksum disabled).

*A tunnel protocol that introduces control information MUST provide its own integrity check methods (e.g. validating the integrity of all control information and the length of the control packet).

*Non-IP inner packets MUST use a CRC or other mechanism for checking packet length and integrity.

3.3. Backwards compatibility with RFC 2460

TOC

There are three possible behaviours when a UDPTT datagram is received by an IPv6 host that only supports UDP as defined in (Deering, S. and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” December 1998.) [RFC2460].

1. A receiver checksum algorithm that uses the transport header Length field to calculate the UDP checksum (as defined for UDP in (Deering, S. and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” December 1998.) [RFC2460]) will result in a valid checksum. However, the number of bytes forwarded to the upper layer, is dependent on how the payload length is interpreted when forwarding to the upper layer. An implementation could forward a number of bytes corresponding to the UDP Length field (i.e. 8 bytes), removing the payload part. Since the UDP Length could be interpreted as indicating there is no payload part. This behaviour would result in an application receiving null UDP datagrams. Application designers are encouraged to design their applications to be robust to

reception of such datagrams [RFC5405] (Eggert, L. and G. Fairhurst, “Unicast UDP Usage Guidelines for Application Designers,” November 2008.). Since no data is passed to the application, there is no danger of inserting unwanted bytes into the data stream at the receiver. This behaviour is safe, but no tunnel can be established until the stack is updated to support UDPTT.

2. A receiver with a checksum that uses the Upper-Layer Packet Length from the UDP Length field, and forwards a number of bytes corresponding to the IP Length field (less any extension headers present). This receiver will calculate a correct checksum. The transport layer will forward the UDP datagram towards the application with the payload part. This is also the expected behaviour for UDPTT. The application using the transport service will receive a set of bytes that are bit protected and therefore may have been modified in transit. Since the UDP payload length is not verified, the number of bytes could also be modified in transit. This behaviour may not be what was intended by a UDP application developer. A tunnel application designed for UDPTT can use this behaviour.

3. A receiver with a checksum that uses the IP Length field is not compliant with UDP defined in (Deering, S. and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” December 1998.) [RFC2460]). This receiver will silently discard the packet, because a mismatching pseudo header would cause the UDP checksum to fail. This behaviour is safe, but no tunnel can be established until the stack is updated to support UDPTT.

3.4. Middlebox Traversal and Incremental Checksum Update

TOC

Middlebox traversal needs to be considered when planning the deployment of any new transport protocol. Middleboxes are known to exist that verify the correctness of the UDP header. Following publication of this specification it is expected that middleboxes will support UDPTT:

*Middleboxes MUST NOT truncate IPv6 datagrams the UDP Header Length is 8 and the IP length exceeds this length.

*If required to update the transport checksum (UDPTT Header Check), a middlebox MAY use the incremental checksum update procedure (Mallory, T. and A. Kullberg, “Incremental updating of the Internet checksum,” January 1990.) [RFC1141].

*If required to validate the transport checksum (UDPTT Header Check), a middlebox MUST use the method defined in this document for IPv6 packets with a UDP length of 8.

This document does not modify the requirement that IPv6 receivers must discard UDP datagrams containing a zero checksum [RFC2460] (Deering, S. and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” December 1998.).

---

4. Acknowledgements

TOC

The author greatfuly acknowledges inputs provided by Magnus Westerlund and Marshall Eubanks on the first version of the draft. Discussion and inputs were provided by Philip Chimento to draft -01.

---

5. IANA Considerations

TOC

The IANA IPv6 Next Header registry entry for the decimal value 17 needs to reference this document in addition to the RFC 2460.

---

6. Security Considerations

TOC

Transport checksums provide the first stage of protection for the stack, although they can not be considered authentication mechanisms. These checks are also desirable to ensure packet counters correctly log actual activity, and can spot unusual behaviours.
Section 3.3 describes middlebox traversal. Firewalls and other security devices may need to be updated to correctly process UDPTT datagrams.
UDPTT presents a possibility of an attack where an attacker sends a flood of 'empty' UDPTT datagrams towards a tunnel endpoint. Datagram applications should be designed to safely receive null datagrams, there is therefore no danger that the tunnel receiver will insert unwanted bytes in the application stream, such packets can contribute to the load of a receiving tunnel server and any middleboxes that process the UDPTT packet stream.
Unlike UDP, the UDPTT Header Check does not validate the length field of the IP header when calculating the transport checksum. This design decision was driven by the goal that the checksum value does not normally change from packet to packet within a single transport flow. The omission of this value is a relaxation of the integrity check. However, a UDPTT application is required to provide its own integrity

check methods. If the IP length field of a UDPTT datagram was modified in transit, a reduced value would result in "truncation" of the packet payload, whereas an increase in value would result in additional data after the intended payload. Endpoints are required to discard any datagrams with an inconsistent length (after accounting for any extension headers that may be present).

Endpoints that enable the use of UDPTT in the same port number range as used for UDP SHOULD provide a method to allow a sending and receiving application to indicate which port use a specific mode. This could be performed using a socket option call to allow an application to request use of the UDPTT semantics.

UDPTT is compatible with the IPsec Encapsulation Security Protocol, ESP (Kent, S., "IP Encapsulating Security Payload (ESP)," December 2005.) [RFC4303], and the Authentication Header, AH (Kent, S., "IP Authentication Header," December 2005.) [RFC4302]. This may be used to encapsulated a UDPTT packet, although this introduces processing that may not be desirable in some deployment scenarios. IPsec may be used within a tunnel-encapsulated packet.

A section describes issues relating to backwards compatibility in hosts. This section may also be applicable to middleboxes that manipulate the transport-layer information.

---

7. References

TOC

7.1. Normative References

TOC

[RFC0791]	Postel, J., "Internet Protocol," STD 5, RFC 791, September 1981 (TXT).
[RFC0793]	Postel, J., "Transmission Control Protocol," STD 7, RFC 793, September 1981 (TXT).
[RFC1071]	Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum," RFC 1071, September 1988 (TXT).
[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC2460]	Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, December 1998 (TXT, HTML, XML).

7.2. Informative References

TOC

[ID-UDP-Z]	“IPv6 UDP Checksum Considerations,” 2010.
[RFC0768]	Postel, J., “User Datagram Protocol,” STD 6, RFC 768, August 1980 (TXT).
[RFC1141]	Mallory, T. and A. Kullberg, “Incremental updating of the Internet checksum,” RFC 1141, January 1990 (TXT).
[RFC3828]	Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, “The Lightweight User Datagram Protocol (UDP-Lite),” RFC 3828, July 2004 (TXT).
[RFC4302]	Kent, S., “IP Authentication Header,” RFC 4302, December 2005 (TXT).
[RFC4303]	Kent, S., “IP Encapsulating Security Payload (ESP),” RFC 4303, December 2005 (TXT).
[RFC5405]	Eggert, L. and G. Fairhurst, “Unicast UDP Usage Guidelines for Application Designers,” BCP 145, RFC 5405, November 2008 (TXT).

Appendix A. Why do we need a checksum?

TOC

Guidance on the use of UDP checksums with IPv6 is provided in UDP [ID-UDP-Z] (, “IPv6 UDP Checksum Considerations,” 2010.).
Previous research showed malformed packets can be received across the Internet, a side effect of broken internal processing (internal transfer errors) in routers or hosts. When the checksum is used with UDP/IPv6, it significantly reduces the impact of such errors, reducing the probability of undetected corruption of state (and data) on both the host stack and the applications using the transport service. Corruption in the network may result in:

*a datagram being mis-delivered to the wrong host/router or the wring transport entity within a host/router. Such a datagram should be discarded.

*a datagram payload being corrupted and delivered to the intended host/router transport entity. Such a datagram needs to be either discarded or correctly processed by an application that has its own integrity checks.

*a datagram payload being truncated by corruption of the length field. Such a datagram needs to be discarded.

The decision to omit an integrity check at the IPv6 level means that the transport check is overloaded with many functions including validating:

- the endpoint address was not corrupted within a router - this packet was meant for this destination and a wrong header has not been spliced to a different payload.
- the extension header processing is correctly delimited - the start of data has not been corrupted. The protocol types does this also to some extent.
- reassembly processing, when used.
- the length of the payload.
- the port values - i.e. the correct application gets the payload (applications should also check source ports/address).
- the payload integrity.

In IPv4, the first 4 checks are made by the IPv4 header checksum.
In IPv6, this checking occurs within the stack using the UDP checksum information. UDPTT also performs these checks (with the exception of the length field, which UDPTT performs using the tunnel encapsulated packet).
An IPv6 node also relies on the header information to determine whether to send an ICMPv6 error message and to determine the node to which this is sent. Corrupted information may lead to misdelivery to an unintended application socket on an unexpected host.
In tunnel encapsulations, payload integrity and length verification may be provided by higher layer tunnel encapsulations (often using the IPv4, UDP, UDP-Lite, or TCP checksums).
There are implications on the detectability of mis-delivery of a packet to an incorrect endpoint/socket, and the robustness of the Internet infrastructure.
The IETF has defined other tunneling protocols that do not include a check value. However, these are typically layered directly over the Internet layer (identified by the upper layer type field) and are not also used as endpoint transport protocols. Specifically packets are only delivered to protocol modules that process a specific next header value. The next header field therefore provides a first-level check of correct demultiplexing. Since the UDP port space is shared many diverse application, this check is not available when UDP is used as transport and therefore the demultiplexing relies solely on the destination port number.
Deterministic reporting of statistics is desirable: router/endpoint MIBs and other statistics gathering methods have the ability to detect

this type of error, rather than recording this as valid traffic between spurious endpoints.
Some IPv6 aware middleware and firewalls may drop or truncate UDPTT datagrams.
{Note: The author would be glad to know of specific cases of truncation and other behaviours.}

A.1. IPv4 Compatibility

TOC

The current version of this document does not specify encapsulation using IPv4 [RFC0791] (Postel, J., "Internet Protocol," September 1981.). For this network protocol. UDP is permitted to disable the UDP checksum and rely on the IPv4 header checksum.
{Note: Future versions of this document could also consider support for IPv4 if the WG considers this useful|}

A.2. Why not set the IPv6 UDP checksum to zero?

TOC

{This section to be expanded in future revisions}
Topics to be discussed:

- The role of a router and host are not fixed. It can not be assumed that a particular protocol (or transport mode) will only be used on a specific type of network node (e.g. the UDP checksum can be disabled only on a router). In IPv6, a node may select a role of a router or host on a per interface basis. Protocol changes intended for one specific use are often re-used for different applications.

- Why ignore checksum on reception is niave

- Behaviour of NAT/Middleboxes needs to be updated for UDPTT and for UDP cksum==0

- Implications on host acting as routers and transport end points.

- Requires restrictions on recursive tunnels that are not necessary with UDPTT

TOC

Appendix B. Applicability for AMT

This specification is intended to be suited to use with "Automatic IP Multicast Without Explicit Tunnels", also known as "AMT". AMT currently specifies UDP as the transport protocol for tunneled packets; that is, the outer packet carrying a tunneled (inner) packet. The specification is for packets carrying tunneled multicast data only. In AMT, the UDP checksum in the UDP header of the outer packet SHOULD be 0 (See draft-ietf-mboned-auto-multicast-09, Section 6.6). However RFC 2460 (IPv6) explicitly states that IPv6 receivers MUST discard UDP packets with a 0 checksum. So, while sending a UDP packet with a 0 checksum is permitted in IPv4 packets, it is explicitly forbidden in IPv6 packets. The computation of an additional checksum, when the inner packet(s) are already adequately protected, is seen to be an unwarranted burden on nodes implementing lightweight tunneling protocols.
The intention is that UDPTT offers a safe alternate approach to the IPv6 method currently defined in AMT.

Appendix C. Document Change History

TOC

{RFC EDITOR NOTE: This section must be deleted prior to publication}

Individual Draft 00 This is the first presentation of this document.

Draft -01 Phil Chimento helped define changes to improve the protocol.

- Added text on excluding the header length value in the pseudo header for the UDPTT Header Check.
- Rewrote security considerations
- Added caveats for protocols using UDPTT

Draft -02 Fixed typos from XML formatting

- Added some text on ICMP
- Middleboxes MUST use UDPTT semantics for UDPTT
- Added more text on why UDP cksum==0 may be bad
- Added text on UDPTT API needs
- Allowed recursion, of UDPTT, but not final inner protocol

Draft -03

Fixed typos and added reference to "IPv6 UDP Checksum Considerations"

Issues *More detailed analysis of the UDP cksum==0 case could be added

---

Author's Address

TOC

	Godred Fairhurst
	University of Aberdeen
	School of Engineering
	Aberdeen, AB24 3UE,
	Scotland, UK
Phone:	
Email:	gorry@erg.abdn.ac.uk
URI:	http://www.erg.abdn.ac.uk/users/gorry