Internet Engineering Task Force	G. Fairhurst
Internet-Draft	University of Aberdeen
Intended status: Informational	M. Westerlund
Expires: September 23, 2010	Ericsson Research
	March 22, 2010

IPv6 UDP Checksum Considerations
draft-fairhurst-tsvwg-6man-udpzero-02.xml

Abstract

This document examines the role of the transport checksum when used with IPv6, as defined in RFC2460. It presents a summary of the trade-offs for evaluating the safety of updating RFC 2460 to permit an IPv6 UDP endpoint to use a zero value in the checksum field to indicate that no checksum is present. The document describes issues and design principles that need to be considered and provides recommendations.

Table of Contents

---

1. Introduction

The User Datagram Protocol (UDP) transport was defined by RFC768 (Postel, J., "User Datagram Protocol," August 1980.) [RFC0768] for IPv4 RFC791 (Postel, J., "Internet Protocol," September 1981.) [RFC0791] and is defined in RFC2460 (Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," December 1998.) [RFC2460] for IPv6 hosts and routers. A UDP transport endpoint may be either a host or a router. The UDP Usage Guidelines (Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers," November 2008.) [RFC5405] provides overall guidance for application designers, including the use of UDP to support tunneling. These guidelines are applicable to this discussion.

This section provides a background to key issues, and introduces the use of UDP as a tunnel transport protocol.

Section 2 describes a set of standards-track datagram transport protocols that may be used to support tunnels.

Section 3 evaluates proposals to update the UDP transport behaviour to allow for better support of tunnel protocols. It focuses on a proposal to eliminate the checksum for this use-case with IPv6 and assess the trade-offs that would arise.

Section 4 reviews the trade offs and provides recommendations.

1.1. Background

TOC

An Internet transport endpoint should concern itself with the following issues:

- *Protection of the endpoint transport state from unnecessary extra state (i.e. Invalid state from rogue packets).
- *Protection of the endpoint transport state from corruption of internal state.
- *Pre-filtering by the endpoint of erroneous data, to protect the transport from unnecessary processing and from corruption that it can not itself reject.
- *Pre-filter of incorrectly addressed destination packets, before responding to a source address.

UDP, as defined in [RFC0768] (Postel, J., "User Datagram Protocol," August 1980.), 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 UDP header includes a 16-bit one's complement checksum that provides a statistical guarantee that the payload was not corrupted in transit. This also allows a receiver to verify that the endpoint was the intended destination of the datagram, because the pseudo header 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. Although the IPv4 UDP (Postel, J., “User Datagram Protocol,” August 1980.) [RFC0768] checksum may be disabled, applications are recommended to enable UDP checksums [RFC5405] (Eggert, L. and G. Fairhurst, “Unicast UDP Usage Guidelines for Application Designers,” November 2008.).

IPv4 UDP checksum control is often a kernel-wide configuration control (e.g. In Linux and BSD), rather than a per socket call. There are Networking Interface Cards (NICs) that automatically calculate TCP/UDP checksums on transmission if a checksum of zero is sent to the NIC, using a method known as checksum offloading.

The network-layer fields that are validated by a transport checksum are:

- Endpoint IP source address (always included in pseudo header of checksum)
- Endpoint IP destination address (always included in pseudo header of checksum)
- Upper Layer Payload type (always included in pseudo header of checksum)
- IP length of payload (always included in pseudo header of checksum)
- Length of the network layer extension headers (i.e. By correct position of checksum bytes)

The transport-layer fields that are validated by a transport checksum are:

- Transport demultiplexing, i.e. ports (always included in checksum)
- Transport payload size (always included in checksum)

Transport endpoints also need to verify correctness of reassembly of any fragmented packets (unless the application use of the payload is corruption tolerant as indicated by UDP-Lite's checksum coverage field). For UDP, this is normally provided as a part of the integrity check. Disabling the IPv4 checksum prevents this check. A lack of checksum can lead to issues in a translator or middlebox (e.g. Many IPv4 Network Address Translators, NATs, rely on port numbers to find the mappings, packet fragments do not carry port numbers, so fragments get dropped). RFC2765 (Nordmark, E., “Stateless IP/ICMP Translation Algorithm (SIIT),” February 2000.) [RFC2765] provides some guidance on the processing of fragmented IPv4 UDP datagrams that do not carry a UDP checksum.

IPv6 does not provide a network-layer integrity check. The removal of the IPv6 header checksum released routers from a need to update a network-layer checksum on a hop-by-hop basis when they changed the IPv4 Time-To-Live (TTL) or IPv6 Hop Count. The IP header checksum calculation was seen as redundant for most traffic (TCP and UDP with checksums enabled), and people wanted to avoid this extra processing. However, there was concern that the removal of the IP header checksum in IPv6 would lessen the protection of the source/destination IP addresses and result in a significant (a multiplier of ~32,000) increase in the number of times that a UDP packet was accidentally delivered to the wrong destination address and/or apparently sourced from the wrong source address when UDP checksums were set to zero. This would have had implications on the detectability of mis-delivery of a packet to an incorrect endpoint/socket, and the robustness of the Internet infrastructure. The use of the UDP checksum is required[RFC2460] (Deering, S. and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” December 1998.) when applications transmit UDP over IPv6.

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.

1.2.1. Motivation for new approaches

TOC

A number of tunnel protocols are currently being defined (e.g.. Automated Multicast Tunnels, AMT (Internet draft, draft-ietf-mboned-auto-multicast-09, “Automatic IP Multicast Without Explicit Tunnels

(AMT)," June 2008.) [AMT], and the Locator/Identifier Separation Protocol, LISP (Internet draft, draft-farinacci-lisp-12.txt, "Locator/ID Separation Protocol (LISP)," March 2009.) [LISP]). These protocols have proposed an update to IPv6 UDP checksum processing. These tunnel protocols could benefit from simpler checksum processing for various reasons:

- Reducing forwarding costs, motivated by redundancy present in the encapsulated packet header, since 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).
- Eliminating a need to access the entire packet when forwarding the packet.
- Enhancing ability to traverse middleboxes, especially NATs.
- A desire to use the port number space to enable load-sharing.

1.2.2. Reducing forwarding cost

TOC

It is a common requirement to terminate a large number of tunnels on a single router/host. Processing costs per tunnel concern both state (memory requirements) and processing costs.
Automatic IP Multicast Without Explicit Tunnels, known as AMT (Internet draft, draft-ietf-mboned-auto-multicast-09, "Automatic IP Multicast Without Explicit Tunnels (AMT)," June 2008.) [AMT] currently specifies UDP as the transport protocol for tunneled packets carrying tunneled IP multicast packets. The current specification for AMT requires that the UDP checksum in the outer packet header SHOULD be 0 (see Section 6.6). It argues that the computation of an additional checksum, when an inner packet is already adequately protected, is an unwarranted burden on nodes implementing lightweight tunneling protocols. The AMT protocol needs to replicate a multicast packet to each gateway tunnel. In this case the outer IP addresses are different for each tunnel and therefore require a different pseudo header to be built for each UDP replicated encapsulation.
The argument concerning redundant processing costs is valid regarding the integrity of a tunneled packet. In some architectures (e.g. PC-based routers), other mechanisms may also significantly reduce checksum processing costs: There are implementations that have optimised checksum processing algorithms, including the use of checksum-offloading. This processing is readily available for IPv4 packets at high line rates. Such processing may be anticipated for IPv6 endpoints, allowing them to reject corrupted packets without further processing.

Relaxing RFC 2460 to minimise the processing impact for existing hardware is a transition policy decision, which seems undesirable if at the same time it yields a solution that may reduce stability and functionality in future network scenarios.

1.2.3. Need to inspect the entire packet

TOC

The currently-deployed hardware in many routers uses a fast-path processing that only provides the first n bytes of a packet to the forwarding engine, where typically n < 128. This prevents fast processing of a transport checksum over an entire (large) packet. Hence the currently defined IPv6 UDP checksum is poorly suited to use within routers that are unable to access the entire packet and do not provide checksum-offloading.

1.2.4. Interactions with middleboxes

TOC

In IPv4, UDP-encapsulation may be desirable for NAT traversal, since UDP support is commonly provided.
IPv6 NAT traversal does not necessarily present the same protocol issues as for IPv4. It is not clear that NATs will work the same way for IPv6. Any change to RFC 2460 is going to require rewriting IPv6 (or defining it) NAT behaviour to achieve consistent widescale deployment.
The requirements for IPv6 firewall traversal are likely be to be similar to those for IPv4. In addition, it can be reasonably expected that a firewall conforming to RFC 2460 will not regard UDP datagrams with a zero checksum as valid packets, and if such a mode were to be defined for IPv6 these may also need to be updated.
Key questions in this space include:

- What types of middleboxes does the protocol need to cross (routers, NAT boxes, firewalls, etc.), and how will those middleboxes deal with these packets?
- What do IPv6 routers do today with zero-checksum UDP packets?
- What other IPv6 middleboxes exist today, and what would they do?

TOC

1.2.5. Support for load balancing

The UDP port number fields have been used as a basis to design load-balancing solutions for IPv4. This approach could also be leveraged for IPv6. However, support for extension headers would increase the complexity of providing standards-compliant solutions for IPv6.
An alternate method could utilise the IPv6 Flow Label to perform load balancing. This would release IPv6 load-balancing devices from the need to assume semantics for the use of the transport port field. This use of the flow-label is consistent with the intended use, although further clarity may be needed to ensure the field can be consistently used for this purpose, (e.g. ECMP [ECMP] (, “Using the IPv6 flow label for equal cost multipath routing in tunnels (draft-carpenter-flow-ecmp),” .)).
Router vendors could be encouraged to start using the IPv6 Flow Label as a part of the flow hash.

---

2. Standards-Track Transports

TOC

2.1. UDP with Standard Checksum

TOC

UDP with standard checksum behaviour is defined in RFC 2460, and should be the default choice. Guidelines are provided in [RFC5405] (Eggert, L. and G. Fairhurst, “Unicast UDP Usage Guidelines for Application Designers,” November 2008.).

2.2. UDP-Lite

TOC

UDP-Lite [RFC3828] (Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, “The Lightweight User Datagram Protocol (UDP-Lite),” July 2004.) offers an alternate transport to UDP, specified as a proposed standard, RFC 3828. A MIB is defined in RFC 5097 and unicast usage guidelines in [RFC5405] (Eggert, L. and G. Fairhurst, “Unicast UDP Usage Guidelines for Application Designers,” November 2008.). UDP-Lite has been implemented, e.g. as a part of the Linux kernel since version 2.6.20.
UDP-Lite 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/corruption in the insensitive part will not cause the datagram to be discarded by the transport layer at the receiving host. A minor

side-effect of using UDP-Lite is that this was specified for damage-tolerant payloads, and some link-layers may employ different link encapsulations when forwarding UDP-Lite segments (e.g. Over radio access bearers). When the checksum covers the entire packet, which should be the default, UDP-Lite is semantically identical to UDP and is specified for use with IPv4 and IPv6. It uses an IP protocol type (or IPv6 next header) with a value of 136 decimal. This value is different to that used by UDP.

2.2.1. Using UDP-Lite as a Tunnel Encapsulation

TOC

Tunnel encapsulations can use UDP-Lite (e.g. Control And Provisioning of Wireless Access Points, CAPWAP), since UDP-Lite provides a transport-layer checksum, including an IP pseudo header checksum, in IPv6, without the need to traverse the entire packet.
In the LISP case, the bytes that would need to be "checksummed" for UDP-Lite would be the set of bytes that are added to the packet by the LISP encapsulating router. When an IPv4/UDP header is per-pended by a LISP router, the LISP ETR needs to calculate the IP header checksum over 20 bytes (the IP header). If an IPv6/UDP-Lite header were per-pended by a LISP router, the ETR would need to calculate an IP header checksum over 48 bytes (the IP pseudo header and the UDP header). This results in an increase in the number of bytes to be the checksummed for IPv6 (48 bytes rather than 20), but this is not thought to be a major processing overhead for a well-optimized implementation where the pre-pended header bytes are already in memory.

2.3. IP in IPv6 Tunnel Encapsulations

TOC

The IETF has defined a set of tunneling protocols. These do not include a checksum, since tunnel encapsulations are typically layered directly over the Internet layer (identified by the upper layer type field) and are also not used as endpoint transport protocols. That is, there is little chance of confusing a tunnel-encapsulated packet with other application data that would result in corruption of application state or data.
From the end-to-end perspective, the principal difference is that the Next Header field identifies a separate transport, which reduces the probability that corruption could result in the packet being delivered to the wrong endpoint or application. 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. In contrast, the UDP port space is shared by

many diverse application and therefore UDP de multiplexing relies solely on the port numbers.

---

3. Evaluation of proposal to update to RFC 2460 to support zero checksum

TOC

This section evaluates a proposal to update IPv6 [RFC2460], to provide the option that some nodes may suppress generation and checking of the UDP transport checksum. 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 type field also provides some protection.
- 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 performed using the IPv4 header checksum.
In IPv6, these checks occur within the endpoint stack using the UDP checksum information. 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.

3.1. Alternatives to the Standard Checksum

TOC

There are several alternatives to the normal method for calculating the UDP Checksum that do not require a tunnel endpoint to inspect the entire packet when computing a checksum. These include (in decreasing complexity):

- *Delta computation of the checksum from an encapsulated checksum field. Since the checksum is a cumulative sum (RFC 1624), an encapsulating header checksum can be derived from the new pseudo header, the inner checksum and the sum of the other network-layer fields not included in the pseudo header of the encapsulated packet. This would not require access to the whole packet, but does require header fields to be collected across the header, and arithmetic operations on each packet. The method would only work for packets that contain a 2's complement transport checksum (i.e. it would not be appropriate for SCTP or when IP fragmentation is used). The process may be easier for IPv4 over IPv6 encapsulation, where the encapsulated IPv4 header checksum could be used as a basis.

- *UDP-Lite. Where the checksum coverage may be set to only the header portion of a packet. This requires a pseudo header checksum calculation only on the encapsulating packet header, which includes extracting the UDP payload length for the pseudo header, however this is expected to be also known when performing packet forwarding. The value may be cached per flow/destination, and subsequently combined only with the Length field to minimise per-packet processing.

- *The UDP Tunnel Transport, UDPTT [UDPTT] (, “The UDP Tunnel Transport mode,” Feb 2010.)(if progressed), where UDP is modified to derive the checksum only from the encapsulating packet protocol header. This value does not change between packets in a flow. The value may be cached per flow/destination to minimise per-packet processing.

- *UDP modified to disable checksum processing [UDPZ] (, “UDP Checksums for Tunneled Packets,” (Oct 2009.)(if progressed). This requires no checksum calculation.

These options are discussed further in later sections.

3.2. Applicability of method

TOC

The expectation of the present proposal to permit omission of UDP checksums [UDPZ] (, “UDP Checksums for Tunneled Packets,” (Oct 2009.) is that this would apply only to IPv6 router nodes that implement specific protocols. However, the distinction between a router and a host is not always clear, especially at the transport level. Systems (such as unix-based operating systems) routinely provide both functions. There is no way to identify the role of a receiver from a received packet.

Any new method would therefore need a specific applicability statement indicating when this mechanism can (and can not). There are additional requirements, e.g. that fragmentation is not performed, since correct reassembly can not be verified at the receiver without a checksum. This would also open the receiver to a wide range of mis-behaviours. This implies disabling host-based fragmentation. Policing this and ensuring correct interactions with the stack implies much more than simply disabling the checksum algorithm for specific packets at the transport interface. There are also proposals to simply ignore a specific received UDP checksum value, however this also can result in problems (e.g. when used with a NAT that always adjusts the checksum value).
The IETF should carefully consider constraints on sanctioning the use of this mode. If this is specified and widely available, it may be expected to be used by applications that are perceived to gain benefit.
Any solution that uses an end-to-end transport protocol (rather than an IP in IP encapsulation) also needs to minimise the possibility that end-hosts could confuse a corrupted or wrongly delivered packet with that of data addressed to an application running on their endpoint.

3.3. Effect of packet modification in the network

TOC

When a checksum is used with UDP/IPv6, this 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.
P packets may be corrupted as they travers an Internet path. Evidence has been presented [Sigcomm2000] (http://conferences.sigcomm.org/sigcomm/2000/conf/abstract/9-1.htm, “When the CRC and TCP Checksum Disagree,” 2000.) to show that this was once an issue with IPv4 routers, and occasional corruption could result from bad internal router processing in routers or hosts. These errors are not detected by the strong frame checksums employed at the link-layer (RFC 3819). There is no current evidence that such cases are rare in the modern Internet, nor that they may not be applicable to IPv6. It therefore seems prudent not to relax this constraint. The emergence of low-end IPv6 routers and the proposed use of NAT with IPv6 further motivate the need to protect from this type of error.
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 needs to 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.

3.3.1. Corruption of the destination IP address

TOC

An IP endpoint destination address could be modified in the network (corrupted by errors). This modification can not be detected in the network when using IPv6. This is not a concern in IPv4, because the IP header checksum will result in this packet being discarded by the receiving IP stack.
There are two possible outcomes:

* Delivery to an address that is not in use (the packet will not be delivered, but could result in an error report).

* Delivery to a different address. This modification will normally be detected by the transport checksum, resulting in silent discard. Without this checksum, the packet would be passed to the port demultiplexing function. If an application is bound to the associated ports, the packet payload will be passed to the application (see the subsequent section on port processing).

3.3.2. Corruption of the source IP address

TOC

This section examines what happens when the source IPv6 address is corrupted in transit. (This is not a concern in IPv4, because the IP header checksum will result in this packet being discarded by the receiving IP stack).
Corruption of an IPv6 packet's source address does not result in the IP packet being delivered to a different endpoint protocol or destination address. If only the source address is corrupted, the packet will likely be processed in the intended context, although with erroneous origin information. The result will depend on the application or protocol that processes the packet. Some examples are:

* An application that requires pre-established context may disregard the packet as invalid, or could map this to another context (if a context for the modified source address was already activated).

* A stateless application will process the packet outside of any context, a simple example is the ECHO server, which will respond with a packet to the modified source address. This would create unwanted additional processing load, and generate traffic to the modified endpoint address.

* Some applications build state using the information from packet headers. A previously unused source address would result in receiver processing and the creation of unnecessary transport-layer state at the receiver. For example, RTP flows commonly employ a source independent receiver port. State is created for each received flow. Reception of a packet with a corrupted source address would result in accumulation of unnecessary state in the RTP state machine, including collision detection and response (since the same Synchronization source, SSRC, value will appear to arrive from multiple source IP addresses).

In general, the effect of corrupting the source address will depend upon the protocol that processes the packet and its robustness to this error. For the case where the packet is received by a tunnel endpoint, the application is expected to correctly handle a corrupted source address.
This effect is more difficult to quantify when several fields have been modified in transit, and the receiving application is not that originally intended.

3.3.3. Delivery to an unexpected port

TOC

This section considers what happens if one or both of the UDP port values are corrupted in transit. (This can also happen with IPv4 in the zero checksum case, but not when UDP checksums are enabled or with UDP-Lite). If the ports were corrupted in transit, packets may be delivered to the wrong process (on the intended machine) and/or responses or errors sent to the wrong application process (on the intended machine). There are several possible outcomes for a packet that passes and does not use the UDP checksum validation:

* Delivery to a port that is not in use. This is discarded, but could generate an ICMPv6 message (e.g. port unreachable).

* It could be delivered to a different node that implements the same application, where the packet may be accepted, generating side-effects or accumulated state.

* It could be delivered to an application that does not implement the tunnel protocol, where the packet may be incorrectly parsed, and misinterpreted, generating side-effects or accumulated state.

The probability of this happening depends on the statistical probability that the source address and the destination port of the datagram (the source port is not always used in UDP) match those of an existing connection.
Unfortunately, this may be more likely for UDP than for connection-oriented transports: (a) There is no handshake prior to communication and no sequence numbers (as in TCP, DCCP, SCTP). Together this makes it hard to verify that an application is given only the data associated with a session. (b) Applications writers often bind to wild-card values in endpoint identifiers and do not always validate correctness of datagrams they receive. While we could revise these rules and declare naive applications as Historic, this is not realistic - the transport owes it to the stack to do its best to reject bogus datagrams.
If checksum coverage is suppressed, the application needs to provide a method to detect and discard the unwanted data. The encapsulated tunnel protocol would need to perform its own integrity checks on any control information and ensure an integrity check is applied to the tunneled packet. It is not reasonable to assume that it is safe for one application to use a zero checksum value and that other applications will not. It is important to consider the possibility that a packet will be received by a different node to that for which it was intended, or that it will arrive at the correct tunnel destination with the wrong source address in the external header.

3.3.4. Validating the network path

TOC

IP transports designed for use in the general Internet should not assume specific characteristics. Network protocols may reroute packets and change the set of routers and middleboxes along a path. Therefore transports such as TCP, SCTP and DCCP are designed to negotiate protocol parameters, adapt to different characteristics, and receive feedback that the current path is suited to the intended application.
Applications using UDP and UDP-Lite need to provide their own mechanisms to confirm the validity of the current network path.
Any application/tunnel that seeks to make use of zero checksum must include functionality to both negotiate and verify that the zero checksum support is provided by the path and validate that this continues to work (e.g., in the case of re-routing events) between the intended parties. This increases the complexity of using such a solution.

TOC

3.4. Comparision

This section compares different methods. This includes two proposals for updating the behaviour of UDP. These are provided as examples, and do not seek to endorse any specific method or suggest that these proposals are ready to be standardised.
Comparison of functions for selected methods

	UDP	UDPv4 zero	UDPL	IP in IPv4	IP in IPv6	UDPv6	UDPv6 zero	UDPTT
Incremental cksum update?	X	-	X	N/A	N/A	X	-	X
Verification of IP length?	X	X	X	X	X	X	X	X
Detect dest addr corruption?	X	X	X	X	-	X	-	X
Detect NH addr corruption?	-	-	-	X	-	-	-	-
Flow demux fields present?	X	X	X	-	X	X	X	X
Detect port corruption?	X	-	X	N/A	N/A	X	-	X
Detect illegal pay length?	X	X	-	N/A	N/A	X	X	-
Detect pay corruption?	X	-	?	N/A	N/A	X	-	-
Static cksum per flow?	-	X	-	N/A	N/A	-	X	X
Partial/full midbox support?	X	*	?	?	?	X	?	?
Restricted tunnel behaviour	X	*	X	X	?	X	-	X

X   = Provided/supported
-   = Not provided/supported
N/A = Not applicable
?   = Partial support
*   = Supports a subset of functions (i.e. not all combinations)

Table 1

---

4. Requirements on the specification of transported protocols

TOC

If the IETF were to revise the standard for UDP using IPv6 for specific use-cases there are a set of questions that would need to be answered. These include:
Is there a reason why IP in IP is not a reasonable choice for encapsulation?

*Examples of arguments for requiring an encapsulation beyond IP-in-IP include the need for NAT traversal and/or firewall traversal. However, the use of any non-standard transport protocol or variant would also require specific support in middleboxes.

* Another example is a need to perform port-demultiplexing (e.g. for load balancing). This need could be met using UDP, UDP-Lite, or other transports, or by utilising the IPv6 flow label.

Is there a reason why UDP-Lite is not a reasonable choice for encapsulation?

* One argument against using UDP-Lite is that this transport is that this transport is not implemented on all endpoints. However, there is at least one open source implementation.

* Another argument against using UDP-Lite is that it uses a different IPv6 Next Header, which is currently not widely supported in middleboxes (see previous).

* It has also been argued that UDP-Lite requires a checksum computation. The UDP-Lite checksum, for instance includes the length field, but need not include the IP payload, and therefore would not require access to the full datagram payload by the tunnel endpoints.

If the IETF needs to revise the rationale for UDP checksums in RFC 2460, should we remove the checksum or replace it with one closer to UDP-Lite (e.g. UDPTT)?
Topics to be considered in making this decision:

* The role of a router and host are not fixed, and a consistent method must be specified that can be used on all nodes. 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. permitting the UDP checksum to be disabled only on a router). In IPv6, a node selects the role of a router or host on a per interface basis. It is important to note that protocol changes intended for one specific use are often re-used for different applications.

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

* Load balancing may not be enabled for all transport protocols.

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

* Appropriate mechanisms to negotiate and validate the properties of the network path, including consideration of the impact of rerouting.

* Whether this requires restrictions on recursive tunnels (e.g. Necessary when the endpoint is not verified).

If a zero checksum approach were to be adopted by the IETF, the specification should consider adding the following constraints on usage:

1. The method must be specified to verify the integrity of the inner (tunneled) packet.

2. The tunneling protocol must not allow fragmentation of the inner packets being carried. We would suggest the following elaborations of the above restrictions, if a change in the IPv6 specification moves forward: That is, an inner IPv4 packet with a UDP checksum equal to 0 must not be tunneled

3. If a method proposes selective ignoring of the checksum on reception, it needs to provide guidance that is appropriate for all use-cases, including defining how currently standardised nodes handle any new use.

4. Other tunneling protcocols that use the UDP checksum equal to 0 must not be tunneled themselves, even if more deeply encapsulated packets have checksums or other integrity checking mechanisms.

5. Non-IP inner (tunneled) packets must have a CRC or other mechanism for checking packet integrity.

6. The specification needs to consider whether to prevent recursive tunnels (e.g. necessary when the endpoint is not verified).

7. It is recommended that general protocol stack implementations do not by default allow the new method. The new method should remain restricted to devices serving as endpoints of the lightweight tunneling protocol adopting the change.

---

5. Summary

TOC

This document examines the role of the transport checksum when used with IPv6, as defined in RFC2460.
It presents a summary of the trade-offs for evaluating the safety of updating RFC 2460 to permit an IPv6 UDP endpoint to use a zero value in the checksum field to indicate that no checksum is present. A decision not to include a UDP checksum in received IPv6 datagrams could impact a tunnel application that receives these packets. However, a well-designed tunnel application should include consistency checks to validate any header information encapsulated with a packet and ensure

that a an integrity check is included for each tunneled packet. When correctly implemented, such a tunnel endpoint will not be negatively impacted by omission of the transport-layer checksum. However, other applications at the intended destination node or another IPv6 node can be impacted if they are allowed to receive datagrams without a transport-layer checksum.

In particular, it is important that already deployed applications are not impacted by any change at the transport layer. If these applications execute on nodes that implement RFC 2460, they will reject all datagrams without a UDP checksum.

The implications on firewalls, NATs and other middleboxes need to be considered. It should not be expected that NATs handle IPv6 UDP datagrams in the same way as they handle IPv4 UDP datagrams. Firewalls are intended to be configured, and therefore may need to be explicitly updated to allow new services or protocols.

If the use of UDP transport without a checksum were to become prevalent for IPv6 (e.g. tunnel protocols using this are widely deployed), there would also be a significant danger of the Internet carrying an increased volume of packets without a transport checksum for other applications, potentially including applications that have traditionally used IPv4 UDP transport without a checksum. This result is highly undesirable. In general, UDP-based applications need to employ a mechanism that allows a large percentage of the corrupted packets to be removed before they reach an application, both to protect the applications data stream and the control plane of higher layer protocols. These checks are currently performed by the UDP checksum for IPv6, or the reduced checksum for UDP-Lite when used with IPv6.

Although the use of UDP over IPv6 with no checksum may have merits for use as a tunnel encapsulation and is widely used in IPv4, it is considered dangerous for all IPv6 nodes (hosts and routers). Other solutions need to be found. This requires rthat the IPv4 and IPv6 solutions to differ, since there are different deployed infrastructures.

---

6. Acknowledgements

TOC

Brian Haberman, Brian Carpenter, Magaret Wasserman, Lars Eggert, Magnus Westerlund, others in the TSV directorate.

Thanks also to: Rémi Denis-Courmont, Pekka Savola and many others who contributed comments and ideas via the 6man, behave, lisp and mboned lists.

TOC

7. IANA Considerations

This document does not require IANA considerations.

8. 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 be used to detect unusual behaviours.

9. References

TOC

9.1. 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).

9.2. Informative References

TOC

[AMT]	Internet draft, draft-ietf-mboned-auto-multicast-09, "Automatic IP Multicast Without Explicit Tunnels (AMT)," June 2008.
[ECMP]	"Using the IPv6 flow label for equal cost multipath routing in tunnels (draft-carpenter-flow-ecmp)."
[LISP]	Internet draft, draft-farinacci-lisp-12.txt, "Locator/ID Separation Protocol (LISP)," March 2009.
[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).
[RFC2765]	Nordmark, E., “Stateless IP/ICMP Translation Algorithm (SIIT),” RFC 2765, February 2000 (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).
[Sigcomm2000]	http://conferences.sigcomm.org/sigcomm/2000/conf/abstract/9-1.htm, “When the CRC and TCP Checksum Disagree,” 2000.
[UDPTT]	“The UDP Tunnel Transport mode,” Feb 2010.
[UDPZ]	“UDP Checksums for Tunneled Packets,” (Oct 2009.

Appendix A. Document Change History

TOC

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

Individual Draft 00 This is the first DRAFT of this document - It contains a compilation of various discussions and contributions from a variety of IETF WGs, including: mboned, tsv, 6man, lisp, and behave. This includes contributions from Magnus with text on RTP, and various updates.

Individual Draft 01 This version corrects some typos and editorial NiTs and adds discussion of the need to negotiate and verify operation of a new mechanism (3.3.4).

Individual Draft 02 Version -02 corrects some typos and editorial NiTs.

*Added reference to ECMP for tunnels.

*Clarifies the recommendations at the end of the document.

Authors' Addresses

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
	Magnus Westerlund
	Ericsson Research
	Torshamgatan 23
	Stockholm, SE-164 80
	Sweden
Phone:	
Fax:	
Email:	magnus.westerlund@ericsson.com
URI:	