Overcoming the Sorrows of the Young UDP Options

Raffaele Zullo, Tom Jones, Gorry Fairhurst
University of Aberdeen, UK
{raffaele, tom, gorry}@erg.abdn.ac.uk

***Abstract*—A recently proposed extension adds support for options to a UDP datagram creating an options area after the UDP payload. This leverages redundancy between IP payload length and UDP length. The area carries per-message options, encoded in a way that resembles TCP Options. This design is impacted by the way the UDP checksum is processed. We evaluate how implementations calculate the UDP checksum and unveil an underlying level of ossification that was silently affecting UDP. We demonstrate a range of schemes that result in a range of pathologies, and reduce the expected success for deployment. We detail the extent of each of the issues discussing their genesis and their implications, and describe an approach to neutralize the most widespread pathology. Our findings show how this change can overcome the major obstacles to deployment and significantly enhances the chances of adoption of the new extension.**

I. Introduction and background

The User Datagram Protocol (UDP) [1] is a widely used layer-4 protocol characterized by a minimal set of transport services [2] and a fixed-length header. UDP supports a wide range of applications and newer transport protocols have been implemented on top of UDP by using UDP as substrate (e.g., [3]). Edeline et al. investigated the potential for new transport protocols built atop of UDP compared to TCP [4].

Although UDP has no room for options or extensions, transport options are defined for many IETF transport protocols, including TCP, SCTP and DCCP. These all use options to provide flexibility in protocol design and facilitate evolution by addition of features to a protocol after its initial development.

UDP-Lite [5] extended UDP to support for applications that can tolerate payload corruption. It uses a different IP protocol number and leverages the difference between the IP packet size and the UDP datagram size to assigns the bytes after the UDP payload to an area not protected by a checksum.

A. Deploying New Methods Across the Internet

Deployment of new protocols, new options, and new extensions to protocols have been shown to be constrained by network devices on path that have implemented all or a part of a transport protocol specification. When network devices rely on the presence of a header field or the semantics of specific header information, this can lead to ossification where an endpoint has to supply a specific header to receive the network service that it desires.

Ossification is also an *intra*-protocol process: while TCP has been successfully extended with options to increase the maximum receive window, timestamps, and selective acknowledgements, more recent extensions are facing deployment difficulties. TCP Fast Open has encountered obstruction by network devices built to enforce the original handshake requirements. This impairment is not related to the options mechanism itself, but to the nature of the new introduced functionality.

UDP itself is not immune from ossification. One example is the amendment of the original IPv6 specifications to allow endpoints to use a zero UDP checksum (to enable tunnel transports that carry an already checksum-protected packet) [6] [7]: these datagrams can still face interference by network devices.

There is a need to evaluate interference due to existing network devices when evaluating a new protocol or extension. Tools such as Pathspider [8], Tracebox [9], Mobile Tracebox [10], Copycat [4] can help assess deployment issues. Learmonth et al. used large-scale measurements to explore traversal of IP packets setting the ECN field [8]. Honda et al. examined the extent to which it is still possible to use options mechanism to extend TCP [11], while Hesmans et al. focused on the lessons learned from previous options to guide the design of future extensions [12]. Zullo et al. analysed limitations due to middleboxes in cellular and Wi-Fi networks, highlighting adoption of extensions and performance tuning proceed at a slower pace compared to user devices [10].

B. UDP Options

UDP-Options (UDP-O) [13] is a work in progress within the Internet Engineering Task Force. This proposal seeks to define an extension method to UDP that adds the possibility to include options in UDP datagrams [13]. The format for a UDP-O datagram, shown in Fig. 1, uses the area after the payload, sometimes known as the surplus area, to carry a set of per-message options. Each option is encoded as a type length value, similar to the encoding of TCP Options. The support for options can enable evolution in the face of the present ossification of the UDP transport.

The options added to a UDP datagram can be either local to a message or affect a stream of messages in a soft-state manner [13]. At the receiver, options are either consumed by the stack or are passed out-of-band to the upper layer protocol.

UDP-O can provide common parameters relating to individual datagrams and communicate remote parameters, such as the receiver maximum datagram size, or support for an optional transport function. Options can signal metadata about a stream to the network path, such as loss reports, RTT, and

978-3-903176-27-0

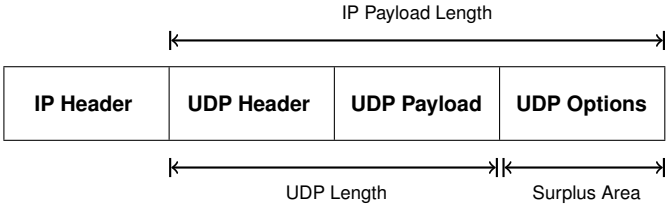

Fig. 1: UDP Options area.

TABLE I: UDP Options path pathologies observed.

Pathology	Notes
UDP Checksum validation	Correct UDP Checksum IP Payload Checksum 3rd Checksum 4th Checksum
Length consistency check	UDP Length = IP Payload Length

ECN feedback. This can be valuable even when the datagram payload is encrypted [3] opening the possibility of common tools for on-path devices to analyse the provided metadata. UDP-O can provide generic support for higher level transport features such as keep-alive and probe packets (e.g. to facilitate NAT traversal), or innovative methods such as Firewall and Service Tickets (FAST [14]) to assist in identifying flows (e.g., connection tracking by firewalls). This alleviates the burden of implementing each function in every application. This could, for example, provide the probing method [15] needed to discover the maximum size of UDP datagram that can be sent across a path, enabling UDP Datagram Packetization Layer PMTU Discovery (DPLPMTUD) [16].

A fragmentation option [13] can provide the necessary support for applications that need to send datagrams larger than the PMTU, by enabling transport-layer segmentation. This avoids the pitfalls of IP-layer fragmentation [17] and could benefit DNSSEC by enabling transport of large responses [18].

An open source UDP-O implementation on top of the FreeBSD Operating System is available along with implementations for common network testing and evaluation tools, including iperf, packetdrill and wireshark [19].

Despite the simplicity of the UDP-O mechanism, we recognised a need to examine and measure the Internet traversal properties of packets that carry UDP-O datagrams. In this exploration, we demonstrate the existence of various pathologies, including ossification resulting from network devices that check the value of the IP Payload length, or have interpreted the length field in different ways. We also reveal cases where the transport checksum has been miscalculated. In understanding traversal pathologies, we have been able to propose a method to avoid a significant proportion of cases where traversal fails and propose a checksum compensation option (CCO) that can significantly improve deployability.

The remainder of this paper is organized as follows: Sec. II describes the UDP-O format and current limitations to deployment; Sec. III describes the tools and methodology; Sec. IV analyses the measurements; Sec. V evaluates the impact on deployability; and finally, Sec. VI concludes this paper.

II. UDP Options pathologies

This section describes a preliminary analysis of networks and paths to discover the set of path pathologies that will be encountered when sending datagrams with UDP-O.

We tested a set of edge and core networks, exploring paths to servers that use well-known UDP protocols, such as DNS and STUN. We use this to analyse UDP-O datagram traversal and delved into the anomalous cases using middlebox location techniques to detect interfering devices [9]. This investigated the root cause of interference by manipulating packets to send probes to help understand whether datagrams setting UDP-O were discarded or only forwarded under specific conditions.

The set of discovered pathologies is summarized in Table I. Note that IPv4 packets also compute a network-layer checksum. This checksum did not limit traversal for UDP-O because it is computed on the IP header bytes and the IP Total Length.

A. Construction of the UDP-O Checksum

To understand the pathologies, one needs to first examine the method used to check the integrity of UDP datagrams. The transport checksum [1] computes the 16 bit one's complement of the one's complement sum of all 16 bit words in a pseudo header and the datagram payload [20]. The length of the payload corresponds to the transport header Length field.

An alternative scheme that calculates the same checksum could use the IP Payload length obtained by the IP Total Length minus IP Header Length (or, for IPv6, as Payload Length minus the length of the extension headers, if present), since for a regular UDP datagram (with no options) this results in the same size as the UDP Length. However the outcome of this second scheme is different when the IP length field reflects the presence of options. A scheme that uses the IP Payload length instead of the UDP Length, will fill the pseudo-header with the wrong value and the checksum could then be computed over all IP Payload bytes. Therefore, despite the simplicity of using the UDP checksum, an ambiguity exists in the role of the two length fields when this is used with UDP-O. This can have a serious impact on the probability of successful traversal across a path, and hence the deployability.

B. Pathologies from Evaluation of the UDP-O Checksum

A first sign of a potential issue was found in a seemingly innocuous bug detected in FreeBSD, where the checksum lengths were based on the IP length. This was fixed during our UDP-O implementation [21]. The range of issues is much broader that just this length mismatch.

To understand the range of traversal pathologies, it is important to recognise that the length of a UDP datagram appears as an input three times during computation of the transport checksum (it is used in the UDP header, in constructing the pseudo-header, and to count the number of bytes covered by the checksum). These uses are shown in Fig. 2. When

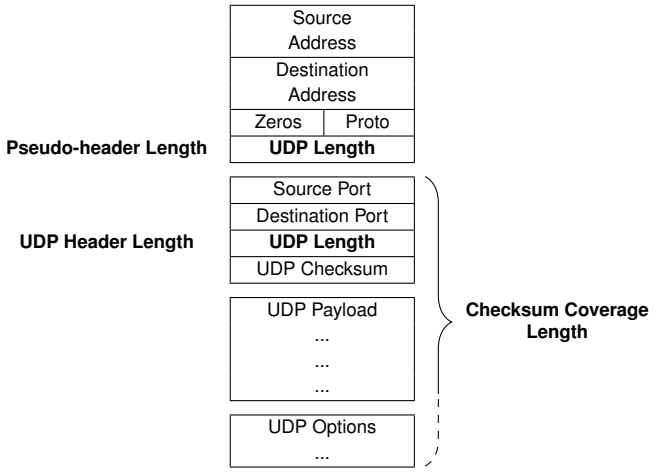

Fig. 2: Checksum computation for the three Length values.

options are included, each usage of the length can result in a different checksum result. The UDP header Length contains always, by definition, the correct value but the combinations of the other fields can result in four schemes with differing checksum values. The types of different checksum schemes that we identified are summarised in Table II.

Our analysis also shows that datagrams can experience multiple successive checks using different schemes. In some cases, more than one check was performed in the same system, perhaps one implemented in the network stack and one in the interface (e.g., when performing checksum offload).

For UDP, these variations in the implementation of the checksum scheme are benign. The scheme has no impact on the resulting checksum value, and hence no impact on the probability that a UDP datagram successfully traverses a network path. The use of multiple checks is benign for UDP-O only when all the checks validate the checksum using the same scheme. Otherwise, use of a different scheme would result in a different checksum value and discard of the datagram.

C. Checks in the Network

On certain paths all UDP-O datagrams were discarded, regardless of which checksum was used. We identified three other interferences within the network.

The end-to-end model does not require network devices to verify transport integrity of packets that they forward. However, our preliminary tests demonstrated that some do validate the UDP checksum. Our investigation revealed checksum computation in middleboxes both in client networks (such as home NATs or Carrier Grade NATs [22]) and in proximity of UDP servers on the Internet (such as firewalls or IDS/IPS [23]). Network devices could also check the consistency between the UDP Length field and the IP payload Length field. For example this could be checked by a security device to avoid a flow exposing a communications side channel.

UDP, and hence UDP-O permit inhibiting the checksum verification for a UDP datagram, by inserting a zero value in the UDP checksum field. Interference by on-path network devices was also found when this zero checksum was used, although this was only evident on a small subset of paths.

We investigated paths using zero checksum and further using the total set of 65,536 values for the checksum on a fixed packet, concluding that some paths check consistency between the UDP Length and IP payload length, discarding datagrams where this fails. We also used the Tracebox [9] methodology to detect other potential middlebox interference, such as deletion or alteration of the surplus area, but we did not find evidence of such pathologies.

D. Overcoming the Constraint of Multiple Checksum Schemes

Our results revealed a much lower probability of traversal than expected, due to conflicting use of different checksum schemes. The 2nd scheme was most widespread, where the IP payload length is used both in the UDP pseudo-header and the coverage length (see Sec. IV). At first sight, these results seem to indicate that UDP-O datagrams are unable to traverse many Internet paths.

We propose a way to neutralise the impact on path traversal resulting from different checksum schemes by introducing a new UDP option, the Checksum Compensation Option (CCO) [24]. The CCO provides an integrity check for the options area using a 2-byte checksum and a 2-byte pseudo-header containing the length of the options. This relies on the observation that unlike a CRC, the order of computation for a checksum does not impact the final result. The CCO therefore adds fields to the UDP options area that compensate for the difference between a correct checksum and the checksum calculated using an IP Payload length (the most common pathology). This result in the same checksum value when either checksum scheme is used (Fig. 3). The CCO must be aligned with the start of the UDP Datagram. Any padding must be taken into account to also compensate for misalignment between the UDP header and the first byte of the options area [24].

III. Path Traversal Measurement

We developed *Tracemore* [25] to perform the measurements in this work. This was derived from the *Mobile Tracebox* code base [26]. We extended its core, written in C and compiled through the Android NDK, to enable deployment on Linux. This tool can customise the IP and UDP fields, including lengths and checksums, forging UDP and UDP-O packets compliant with the presented schemes or with a checksum not compliant with any scheme. The UDP Payload can also be edited using crafted application packets, e.g. a DNS request.

A. Measurement Methodology

Based on the set of discovered pathologies, we built a test suite that utilises a sequence of ten datagrams, shown in Table III. This comprises: *(i)* three UDP datagrams; *(ii)* seven UDP-O datagrams, one with the CCO.

A set of three UDP datagrams are used to characterise the traversal of a path in the absence of using UDP-O (packets *#1-3*). A reply to the first datagram indicates that the path is

TABLE II: Four UDP Checksum computation schemes observed.

	Scheme	UDP Header	UDP Pseudo-header	Checksum Coverage
1	Correct UDP Checksum	UDP Length	UDP Length	UDP Length
2	IP Payload Checksum	UDP Length	IP Payload Length	IP Payload Length
3	3rd Checksum	UDP Length	UDP Length	IP Payload Length
4	4th Checksum	UDP Length	IP Payload Length	UDP Length

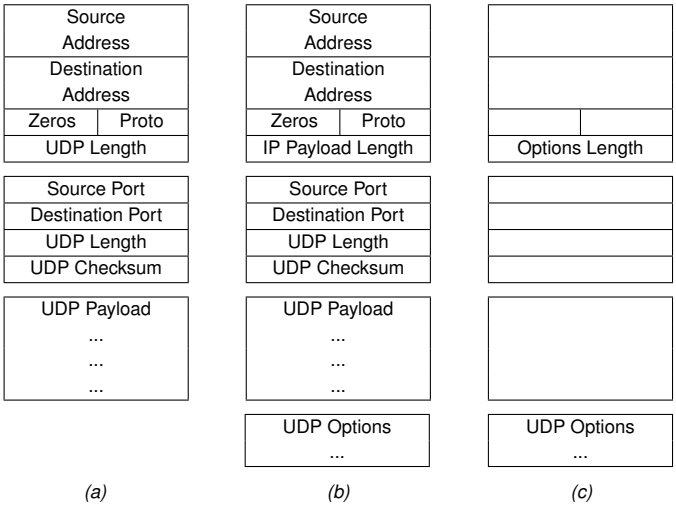

Fig. 3: UDP Checksum computation using UDP Length (a), IP Payload length (b) and the delta between the two (c).

currently forwarding packets (if there is no reply, the test is not continued). The next two datagrams respectively probe to determine whether there is any interference with the use of a zero checksum (packet *#2*); and whether the path performs validation by setting a bad UDP checksum value (in which case, packet *#3* would be discarded).

A set of UDP-O datagrams characterise traversal along the tested path (packets *#4-10*) with a checksum computed according to each of the four schemes in Table II. Each probe provides specific information about the potential interference with UDP-O (see Table I). All UDP-O datagrams carry only a UDP MSS Option [13], except for the last, which includes both a CCO and the MSS Option.

A UDP-O datagram sent with a correct checksum and no CCO option is used to evaluate traversal according to the original specifications (packet *#4*). This is repeated for the other potential pathologies (packet *#5-7*). A UDP-O datagram with a zero transport checksum (packet *#8*) has a dual purpose: it confirms the presence of any interference caused by other than checksum validation, e.g., resulting from an IP and UDP length mismatch, and evaluates UDP-O traversal with a zero checksum, for options that should not be covered by a checksum [13]. For completeness, we also test a UDP-O with a bad checksum (packet *#9*), expecting a response similar to that for packet *#3*. A final UDP-O datagram sent with a correct checksum and including the CCO option is used to evaluate traversal with checksum compensation (packet *#10*).

The set of probe datagrams are sent sequentially from different source ports with a 3 second timeout, repeating each datagram up to 3 times to reproduce the test.

TABLE III: UDP and UDP-O packets in the test suite.

#	Packet	Notes
1	UDP	Correct CS
2	UDP	Zero CS
3	UDP	Bad CS
4	UDP Options	Correct CS
5	UDP Options	IP Payload CS
6	UDP Options	3rd CS
7	UDP Options	4th CS
8	UDP Options	Zero CS
9	UDP Options	Bad CS
10	UDP Options	With CCO

TABLE IV: List of servers targeted by UDP-O measurements.

Protocol	IP	Origin	Addresses	ASes
STUN	IPv4	Full range scan	66K	8K
DNS	IPv4	Alexa Top-1m	190K	15K
HTTP	IPv4	Alexa Top-1m	125K	5K
DNS	IPv6	Alexa Top-1m	17K	1.1K
HTTP	IPv6	Alexa Top-1m	12K	0.3K

B. Measurement Datasets

We next describe measurements performed in November 2018 using a node in the EU SeeWeb data center [27], chosen based on the path transparency of the upstream network, essential for this type of analysis. Initial measurements were replicated on an Azure server (US) [28], also tested for path transparency. We observed only a minimal difference. To construct the Alexa Top-1m list [29] at the client location, we obtained DNS and HTTP lists running the Hellfire Targets server [30], a high-performance DNS resolver designed for generating Internet measurement target lists. For each domain in a list, it produces JSON data containing IP addresses (one IPv4 address and one IPv6 address, if applicable) of either the domain or its authoritative DNS server, AS, Country code, prefix, name and rank, which it caches from the RIPEstat database. Hellfire was executed at the University of Aberdeen.

Our measurements probed paths to a list of STUN, DNS and HTTP servers. The number of different targets and ASes is reported in Table IV. For STUN and DNS servers, each UDP and UDP-O test packet carries a STUN BIND request or a DNS query. This is expected to trigger a response from the destination server. If a response is received the probe packet is considered as succeeding to traverse the path to the destination.

The subset of probe packets that reach the destination provides information about the set of pathologies that affect a

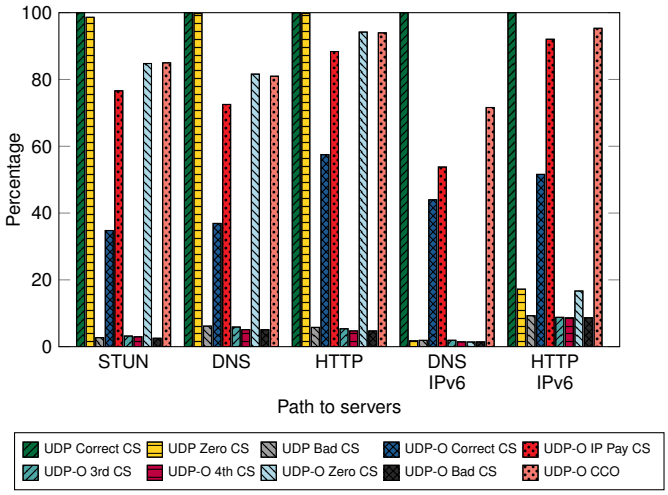

Fig. 4: Overall traversal rate of probe packets.

path. The STUN servers list was obtained from a preliminary full-scan of the IPv4 address range. The DNS list contains the IPv4 and IPv6 addresses of authoritative DNS servers extracted from the Alexa Top-1m list.

To increase the number of paths that we tested, we included paths to HTTP servers. Although these servers are not expected to reply to a UDP datagram on port 80, a portion do reply with an ICMP (or ICMPv6) Port Unreachable message [31] [32]. Others silently drop the UDP datagram. Any received ICMP messages indicate that the test datagrams had reached the target destination. Conversely a lack of ICMP feedback could be due to various reasons, e.g. ICMP filtering at the remote endpoint or on the return path (which may not have been coincident with the forward path). The list of HTTP targets was obtained by resolving Alexa Top-1m websites and removing duplicates. The target list was further reduced by testing if the methodology could be applied (about one quarter of HTTP servers were eligible).

We also measure network paths, even in cases where the packet does not reach the target HTTP server, but is intercepted by a middlebox close to the server, e.g., a firewall that responds with an ICMP error message. Any consistent reply received to a subset of UDP and UDP-O packets shows the presence of one of more UDP-O pathologies in the path to the target.

The results reported in the next section show how measurements from these paths to HTTP servers are qualitatively and quantitatively similar to those that would be obtained from a UDP server, especially for IPv4.

IV. Results

The results obtained from the tests are shown in Fig. 4: The column marked "UDP-O Correct CS" shows a limited successful traversal ratio for a UDP-O datagram using the original specification (packet *#4*), compared to datagrams with an IP Payload checksum, or zero checksum (packets *#5*, *#8*). The CCO Option (packet *#10*) increases traversal to 80% for STUN and HTTP targets. Each probe provides specific information about the path, but their combination can highlight the pathology or pathologies that affect the path.

TABLE V: Path characterization for each test.

Path characterization	**Tests**									
	1	2	3	4	5	6	7	8	9	10
Any Checksum	✓	*	✓	✓	✓	✓	✓	*	✓	✓
Correct UDP CS only	✓	*	x	✓	x	x	x	*	x	✓
IP Payload CS only	✓	*	x	x	✓	x	x	*	x	✓
3rd CS only	✓	*	x	x	x	✓	x	*	x	x
4th CS only	✓	*	x	x	x	x	✓	*	x	x
Correct CS or IP Pay CS	✓	*	x	✓	✓	x	x	*	x	✓
Compensated CS only	✓	*	x	x	x	x	x	*	x	✓
Zero CS only	✓	✓	x	x	x	x	x	✓	x	x
No UDP-O traversal	✓	*	*	x	x	x	x	x	x	x

We next characterise paths using a combination of probe results (as detailed in Table V):

a) Any checksum: all tested packets traverse this path, including those with a bad UDP checksum. This unexpected phenomenon has been observed on paths to both UDP servers and to HTTP servers. Packets with an incorrect checksum should have been discarded, suggesting the destination does not validate the checksum.
b) Correct UDP checksum only: a response is received only to UDP-O datagrams that carry a correct checksum.
c) IP Payload checksum only: only datagrams compliant with this scheme can traverse the path.
d) 3rd checksum only.
e) 4th checksum only. In (c)-(e) UDP-O datagrams with a correct checksum are discarded.
f) Correct UDP checksum or IP Payload checksum: datagrams compliant with one of the two schemes are accepted, while all other non-zero checksums are discarded.
g) Checksum compensated by CCO only: only datagrams compliant with both schemes (using the CCO) reach the destination. This can be explained by a series of checksum validations implementing the two different schemes.
h) Zero checksum only: datagrams with a zero checksum are ignored when detecting checksum pathologies. However, we identified paths where all non-zero checksum datagrams were dropped, which could be explained by multiple checksum validations.
i) No UDP-O traversal: paths where no UDP-O datagrams reach the destination. Discard suggests this is likely due to a consistency check between the IP and UDP length.

Table V takes into account the interdependence between tests. For instance if a path is traversed by a datagram #4 (correct checksum), it is expected to be traversed also by a datagram #10 (but not vice versa). A negligible fraction of measurements (about 1%) indicate inconsistent outcomes, most likely due to further interference by middleboxes (e.g., firewalls triggered by many consecutive STUN BIND requests or DNS queries received from the same host), temporary congestion, or potentially the presence of a load balancer that routes different test datagrams through different paths.

Fig. 5 shows the proportion of the different listed categories,

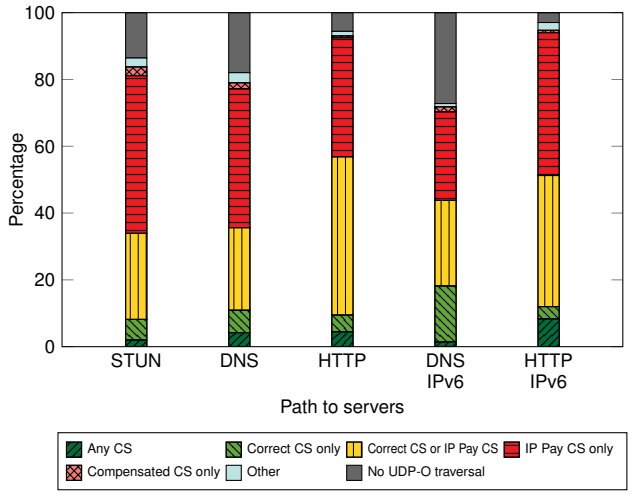

Fig. 5: Path chacterisation by application (results in *green* and *yellow* show traversal using the standardised method; *green*, *yellow*, *red* and *pink* show the cumulative traversal when the CCO is used; *blue* and *grey* represent other pathologies).

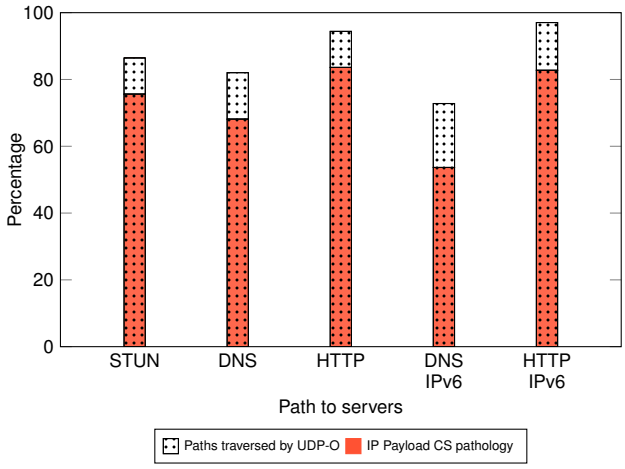

Fig. 6: Paths traversed by at least one UDP-O datagram and paths affected by IP Payload checksum pathology.

grouped by the type of server. To improve the readability, the categories with the smallest percentages, i.e., 3rd checksum only, 4th checksum only and zero checksum only, were included into the "Other" category. We presented these for the sake of completeness and to highlight the diversity of middlebox interference discovered.

The IP Payload checksum pathology (alone or in conjunction with the benign pathology) is the most widespread. To determine its prevalence, we grouped all the affected paths (reported in Fig. 6) and compared these to the set of paths traversed by at least one UDP-O datagram. This showed that at least one validation using the wrong scheme was observed for more than 80% of the paths traversed by UDP-O. On paths that completely block UDP-O, it is not possible to distinguish which checksum computation is performed. Not all paths affected by the pathology are problematic. For instance, UDP-O can be still deployed on paths where both checksums are accepted according to the original specifications. The problematic paths can be traversed using the CCO.

A. Path Traversal using the CCO

Fig. 7 shows the paths traversed by UDP-O datagrams with a correct checksum (categories *a*, *b* and *f* of the previous classification) and the paths traversed only by UDP-O datagrams carrying a CCO (categories *c* and *g*). The CCO significantly increases the number of paths that can be traversed and for IPv4 paths to STUN and DNS servers, the increment from using the CCO is even greater than the number of paths originally traversed by UDP-O.

Despite this improvement, about 16% of paths can not be traversed using the CCO, including paths where another scheme (the 3rd or 4th checksum) or multiple incompatible schemes are used or paths that block UDP-O datagrams (e..g. due to a length consistency check). The ossification on these paths cannot be mitigated and they would need to be updated to enable traversal of UDP-O datagrams.

We further analysed the contribution of using the CCO on a per AS basis to identify whether operational practice influenced the outcome. We found paths to destinations belonging to the same AS were widely different and a significant portion of ASes included paths with different pathologies and characteristics. Fig. 8 shows the percentage of paths where UDP-O datagrams traverse by default and the percentage that work using a CCO. To improve plot readability, ASes are ordered decreasing by the traversal rate using the CCO and then by the traversal rate without the CCO. Three main regions are visible in the figure:

1) ASes in which all paths can be traversed by UDP-O (63.29%): the majority benefit from using the CCO and a large number support UDP-O only when using the CCO;
2) ASes in which a subset of paths can be traversed by UDP-O (17.58%): the CCO increases the number of traversed paths for most ASes in this region.
3) ASes in which no measured path could be traversed by UDP-O, without or with the CCO (19.13%).

B. Path Traversal using a Zero Checksum

One possibility is to allow a UDP zero checksum with UDP-O to improve traversal in cases where the UDP checksum would otherwise be validated using an incorrect scheme. Table VI compares the results for UDP-O traversal using a combination of a CCO and a zero checksum. It highlights the number of paths that can be supported by both solutions, or only one of the two. These results are for IPv4.

A datagram with zero checksum is expected to traverse not only paths enabled by the CCO, but also paths that use one of the other checksum schemes or where multiple and conflicting schemes are used. The results of our measurements show that paths traversed with a zero checksum are not always better than paths traversed with just a CCO: there is a small portion of paths where the CCO works and zero checksum datagrams

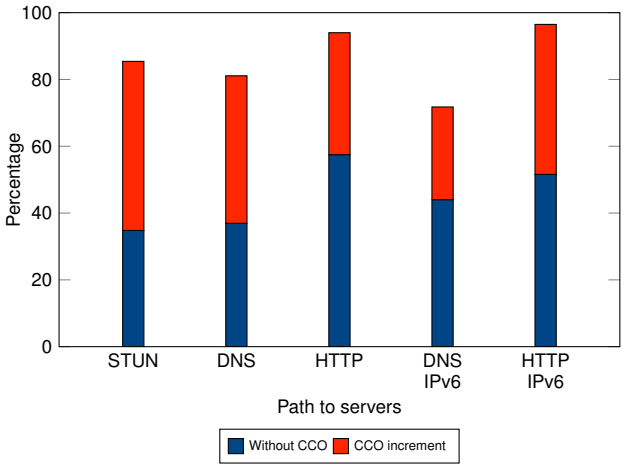

Fig. 7: Path traversal without and with CCO.

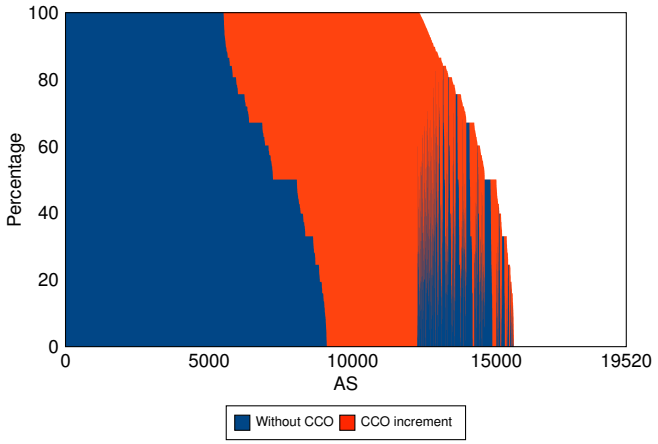

Fig. 8: Percentage of paths traversed by UDP Options packets without and with CCO within each AS.

are discarded. Table VI also reports UDP traversal with zero checksum to highlight that a modest interference with zero checksum was also observed with regular UDP datagrams.

The IPv6 node receiver behavior is required to silently discard all IPv6 packets that carry a datagram with a zero UDP checksum. IPv6 was updated in 2013 [7], to allow specific use cases to utilise a zero transport checksum. It would therefore be useful to understand whether paths also forward IPv6 UDP-O datagrams with a zero checksum. However, our present methodology relies on a response from a destination. We observed a response for only 3% of test cases, preventing further analysis using our tool. Future work will investigate UDP-O traversal for IPv6 with a zero checksum.

V. Discussion

Our results unveil an underlying level of ossification that was silently affecting UDP. In exploring the deployability of UDP-O, our work has revealed different patterns of interference from that were also exposed by previous work [9] [10] [11]. Interference with TCP Options, such as MPTCP [12], concern only one layer, the changes described here involve operation at both the network and transport layers.

A. Ambiguity in the Calculation of the Transport Checksum

Many UDP stacks compute a checksum using the IP Payload length and this scheme is more prevalent than the correct scheme using the UDP length. This design choice could follow the method used for TCP: because the TCP header has no length field, the length of a segment is deduced from the IP header and the checksum computed over all transport-layer bytes, as observed for UDP datagrams in anomalous devices.

Our results show a common case where a path accepts either the correct checksum or one based on the IP length (but all other non-zero checksum values are invalid). A possible explanation is that these two validations occur at different layers within a single device.

To verify this hypothesis we analysed several Linux devices that exhibited this phenomenon, including workstations, servers and Android smartphones, realising that the IP Payload checksum validation was only observed when checksum offloading enabled. A Linux [33] host accepts a datagram when the checksum has been verified by the network interface checksum offload engine, otherwise the checksum is verified in the kernel. Packets with a valid IP payload checksum are accepted by the offload device, while packets with a correct UDP checksum are validated by the kernel. UDP-O packets with a correct checksum need to be validated by the kernel. Offload must also be disabled for transmission (to avoid generating a wrong checksum), requiring kernel checksum computation. When using the CCO, the transport checksum in packets is validated directly by the offload device, only requiring the host stack to verify the CCO covering the options payload.

Rather than require all network devices to be updated to enable traversal of UDP-O, we introduce the CCO which would enable continued coexistence of the two schemes.

In future measurement work, we plan to perform scans over other UDP protocols (on IPv4 full range and IPv6 target lists [34]) to validate our findings on a larger dataset.

TABLE VI: Comparison of UDP-O traversal using CCO and Zero CS on IPv4 paths.

		STUN	**DNS**	**HTTP**
UDP	Zero CS	98.61%	99.73%	99.75%
UDP-O	CCO	84.98%	80.97%	93.95%
	Zero CS	84.78%	81.60%	94.19%
	Both	83.72%	80.66%	93.77%
	Only CCO	1.26%	0.31%	0.18%
	Only Zero CS	1.06%	0.94%	0.42%

B. Deployability of UDP Options with a Zero Checksum

As a potential remedy to path pathologies, we also analysed traversal when the transport checksum is disabled. The limited increment in paths traversed, combined with the loss of a similar percentage of paths due to network devices that drop

datagrams with a zero checksum, suggests that using a zero checksum is not a suitable alternative to using a CCO.

However, there could be other reasons why a UDP-O packet wishes to not set the checksum. For instance, when an encapsulation device does not have full access to the UDP datagram payload. This could be the case for a method seeking to reproduce UDP-Lite semantics by transmitting a payload only partially covered by checksum defined in the UDP-O area. Such a method would not be able to utilise the CCO. However, we anticipate that there is limited usefulness for UDP-Lite semantics, and note that this increases variability in path traversal and anyway complicates the implementation of UDP-O. The authors suggest this may not justify its final inclusion in the specification.

C. Deployability of UDP Options using a CCO

We proposed introduction of the CCO to UDP-O, and showed that this can overcome the major obstacles to traversal with the current network equipment and therefore significantly enhance the chances of adoption of the new extension.

The results show that the CCO can improve the traversal of UDP-O both in terms of single paths and traversal across an AS. The CCO mechanism has since been adopted as a part of the UDP Option specification [13], by incorporating a recommendation to use a (redesigned) Option Checksum (OCS). Importantly, the design of the OCS needs to include the length field in the computation to achieve the benefits of the CCO. Our findings suggest that this field is needed for acceptable traversal, when the UDP-O checksum is non-zero.

D. Genesis of UDP Options Pathologies

We observed middleboxes ossifying around a check for the consistency of the UDP length (e.g., to detect malformed packets or check no covert channel is present). This has a direct impact on the transport function. In other cases, a range of UDP checksum schemes has had an undesired impact that combines to ossify the transport function.

We contacted the manufacturer of one device that blocked UDP-O. They confirmed that their middleboxes performed a consistency check between the IP and UDP length, along with other integrity checks on datagrams and discarded them in the case of a length mismatch. Another network equipment manufacturer explained that their default behavior for a stateful firewall was to discard all packets with incorrect checksums. This is reasonable since, before applying rules that involve the transport layer to the packet, transport integrity should be verified. These rules would need to be revised to enable deployment of UDP-O along paths that include these devices.

VI. Conclusions and Future Work

This paper has analysed the use and computation of the UDP checksum. It reports a range of implemented checksum schemes that have been deployed. This analysis has been used to assess the deployability of UDP Options (UDP-O) which seeks to enable interoperable common transport options with the potential to add flexibility to the core design of UDP.

We report for the first time the current limitations that UDP-O will face due to existing network equipment, revealing an unforeseen level of ossification for UDP. We explain a set of discovered path pathologies and then detail each pathology to discuss its extent, as well as suggesting the origin. Related to UDP-O, we evaluate the implications of the different pathologies and propose an option to mitigate these effects (the CCO). We then show how this can significantly increase UDP-O traversal. We explore the use of UDP zero checksum as a potential alternative proposal and compare that with the use of the CCO, finally concluding the CCO offers most benefit.

References

[1] J. Postel, "User datagram protocol," Tech. Rep., RFC 768.

[2] G. Fairhurst and T. Jones, "Transport Features of the User Datagram Protocol (UDP) and Lightweight UDP (UDP-Lite)," 2018, RFC 8304.

[3] J. Iyengar and M. Thomson, "QUIC: A UDP-based multiplexed and secure transport," 2020, draft-ietf-quic-transport, IETF, Work in progress.

[4] K. Edeline, M. Kühlewind, B. Trammell, and B. Donnet, "copycat: Testing differential treatment of new transport protocols in the wild," in *Applied Networking Research Workshop*, 2017, pp. 13–19.

[5] L.-A. Larzon, M. Degermark, S. Pink, L.-E. Jonsson, and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)," 2004, RFC 3828.

[6] M. Eubanks, P. Chimento, and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets," 2013, RFC 6935.

[7] G. Fairhurst and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums," April 2013, RFC 6936.

[8] I. R. Learmonth, B. Trammell, M. Kuhlewind, and G. Fairhurst, "Pathspider: A tool for active measurement of path transparency," in *Applied Networking Research Workshop*, 2016, pp. 62–64.

[9] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *ACM Internet Measurement Conf. (IMC)*, October 2013.

[10] R. Zullo, A. Pescapé, K. Edeline, and B. Donnet, "Hic sunt proxies: Unveiling proxy phenomena in mobile networks," in *Network Traffic Measurement and Analysis Conf. (TMA)*. IEEE, 2019, pp. 227–232.

[11] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is it still possible to extend TCP," in *ACM Internet Measurement Conf. (IMC)*, November 2011.

[12] B. Hesmans, F. Duchene, C. Paasch, G. Detal, and O. Bonaventure, "Are tcp extensions middlebox-proof?" in *Hot topics in Middleboxes and Network Function Virtualization*, 2013, pp. 37–42.

[13] J. Touch, "Transport options for UDP," 2019, draft-touch-tsvwg-udp-options, Work in progress.

[14] T. Herbert, "Firewall and service tickets," 2019, draft-herbert-fast-04, Work in progress.

[15] G. Fairhurst and T. Jones, "Datagram PLPMTUD for UDP Options," 2020, draft-fairhurst-tsvwg-udp-options-dplpmtud, Work in progress.

[16] G. Fairhurst, T. Jones, I. Tuexen, M Ruengeler, and T. Voelker, "PLPMTUD for Datagram Transports," 2020, draft-ietf-tsvwg-datagram-plpmtud, Work in progress.

[17] R. Bonica, G. Huston, R. Hinden, O. Troan, and F. Gont, "IP Fragmentation Considered Fragile," 2019, draft-ietf-intarea-frag-fragile, Work-in-Progress.

[18] O. Kolkman and R. Gieben, "DNSSEC operational practices," 2006, RFC 4641.

[19] "FreeBSD UDP-O implementation," https://github.com/uoaerg/freebsd.

[20] B. Braden, D. Borman, and C. Partridge, "Computing the internet checksum," 1988, RFC 1071.

[21] "FreeBSD commit r334705," see https://svnweb.freebsd.org/base?view=revision&revision=334705.

[22] P. Richter, F. Wohlfart, N. Vallina-Rodriguez, M. Allman, R. Bush, A. Feldmann, C. Kreibich, N. Weaver, and V. Paxson, "A multi-perspective analysis of carrier-grade nat deployment," in *Internet Measurement Conf.*, 2016, pp. 215–229.

[23] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," in *ACM SIGCOMM*, August 2012.

[24] G. Fairhurst, T. Jones, and R. Zullo, "Checksum Compensation Options for UDP Options," 2018, draft-fairhurst-udp-options-cco, Work-in-Progress.
[25] R. Zullo, "Tracemore," 2018, http://www.middleboxes.org/tracemore.
[26] R. Zullo, A. Pescapé, K. Edeline, and B. Donnet, "Hic sunt NATs: Uncovering address translation with a smart traceroute," in *Network Traffic Measurement and Analysis Conference (TMA)*, 2017.
[27] "SeeWeb datacenter," https://www.seeweb.it/en/data-center.
[28] "Microsoft Azure," https://azure.microsoft.com.
[29] "Alexa Top sites," https://www.alexa.com/topsites.
[30] "Hellfire," 2018, see https://github.com/irl/hellfire.
[31] J. Postel, "Internet control message protocol," IETF, RFC 792, 1981.
[32] A. Conta, S. Deering, and M. Gupta, "ICMPv6," 2006, RFC 4443.
[33] "Linux kernel source tree," 2019, https://github.com/torvalds/linux/blob/master/net/ipv4/udp.c.
[34] O. Gasser, Q. Scheitle, S. Gebhard, and G. Carle, "Scanning the ipv6 internet: towards a comprehensive hitlist," *arXiv preprint arXiv:1607.05179*, 2016.