copycat: Testing Differential Treatment of New Transport Protocols in the Wild

Korian Edeline*, Mirja Kühlewind‡, Brian Trammell‡, Benoit Donnet*
* Université de Liège, Montefiore Institute – Belgium
‡ ETH Zurich, Networked Systems Group – Switzerland

ABSTRACT

Recent years have seen the development of multiple transport solutions to address the ossification of TCP in the Internet, and to ease transport-layer extensibility and deployability. Recent approaches, such as PLUS and Google's QUIC, introduce an upper transport layer atop UDP; their deployment therefore relies on UDP not being disadvantaged with respect to TCP by the Internet.

This paper introduces `copycat`, a generic transport protocol testing tool that highlights differential treatment by the path in terms of connectivity and QoS between TCP and a non-TCP transport protocol. `copycat` generates TCP-shaped traffic with custom headers, and compares its performance in terms of loss and delay with TCP. We present a proof-of-concept case study (UDP vs. TCP) in order to answer questions about the deployability of current transport evolution approaches, and demonstrate the extent of `copycat`'s capabilities and possible applications.

While the vast majority of UDP impairments are found to be access-network linked, and subtle impairment is rare, middleboxes might adapt to new protocols that would then perform differently in the wild compared to early deployments or controlled environment testing.

CCS CONCEPTS

• **Networks → Transport protocols**; **Network measurement**;

KEYWORDS

`copycat`, differential treatment, UDP, ossification

ACM Reference format:
Korian Edeline*, Mirja Kühlewind‡, Brian Trammell‡, Benoit Donnet* * Université de Liège, Montefiore Institute – Belgium ‡ ETH Zurich, Networked Systems Group – Switzerland . 2017. `copycat`: Testing Differential Treatment of New Transport Protocols in the Wild. In *Proceedings of ANRW '17, Prague, Czech Republic, July 15, 2017,* 7 pages.
DOI: 10.1145/3106328.3106330

1 INTRODUCTION

Most Internet applications today are built on top of the Transmission Control Protocol (TCP), or some session-layer protocol that uses TCP, such as the Hypertext Transfer Protocol (HTTP) or WebSockets. Indeed, the ubiquity and stability of TCP as a common facility that handles the hard problems of reliability and congestion control is a key factor that has led to the massive growth of the Internet.

However, not every application benefits from the single-stream, fully-reliable service provided by TCP. In addition, the ubiquitous deployment of network address translators (NATs) and firewalls that only understand a limited set of protocols make new protocols difficult to deploy. Previous attempts to deploy new protocols such as SCTP [21] were hindered by this ossification [8], as well as by the difficulty of rapid deployment of new kernel code across multiple platforms. The deployment of middleboxes that "understand" TCP also limit the ability to deploy new TCP options and features [9]. Much of the design work in Multipath TCP [4, 5], for example, addressed middlebox detection and avoidance.

This has led to a current trend in transport protocol design to use UDP encapsulation to solve this problem. Google's QUIC [6] and the WebRTC data channel [10] both use UDP as an outer transport protocol. In both cases, the transport protocol dynamics (connection establishment, reliability, congestion control, and transmission scheduling) are handled by the inner protocol. In the case of the WebRTC data channel, this is *SCTP over Datagram Transport Layer Security* (DTLS) [26]; in the case of QUIC, it is the QUIC transport protocol itself. This is a new kind of encapsulation. In contrast to traditional tunneling, this approach borrows the "wire image" of UDP for two key benefits: userspace deployability of new transports, due to the ubiquitous availability of UDP sockets for unprivileged, userspace programs; and NAT/firewall traversal, as most such devices recognize UDP ports. The *Path Layer UDP Substrate* (PLUS) effort within the IETF [25] generalizes this approach for new transport protocols.

This situation pleads for protocol deployability and performance assessment. To this end, a widespread methodology consists in implementing new protocols in network simulators to investigate their characteristics. While this solution can be advantageous, it only provides a partial view as the way the actual network behaves is far from the theoretical view provided by simulators.

Therefore, in this paper, we advocate for "in-the-wild" protocol testing in addition to simulations and controlled environments and introduce a novel measurement tool called `copycat`.[1] It aims at evaluating experimental protocols' deployability, or the extensibility of existing ones by comparing their processing by the network, in terms of connectivity and performance, with regular TCP connections.

ANRW '17, Prague, Czech Republic

DOI: 10.1145/3106328.3106330

[1] Sources are freely available at https://github.com/mami-project/copycat

copycat creates TCP traffic with another protocol's (UDP or non-UDP) wire image, and performs full-mesh measurements on test networks in order to determine if differential treatment of experimental protocols' and TCP packets might disadvantage non-TCP congestion-controlled traffic. copycat is generic as it allows for testing a wide variety of transport protocols without the need to actually implement them. It is thus suitable for immediate deployment on any distributed measurement testbed. It also eases the network traces analysis by working only with TCP-controlled flows.

To demonstrate copycat capabilities, we investigate the UDP vs. TCP scenario by deploying copycat on PlanetLab and cloud provider Digital Ocean [3]. In summary, we see evidence of complete blocking of UDP for between 2% and 4% of terrestrial access networks, and find that blocking is primarily linked to access network; these results are in line with reported QUIC performance [22]. We note that these networks are not uniformly distributed throughout the Internet: UDP impairment is especially concentrated in enterprise networks and networks in geographic regions with otherwise-challenged connectivity. Where UDP does work on these terrestrial access networks, we see no evidence of systematic impairment of traffic with UDP headers. The strategy taken by current efforts to encapsulate new transports over UDP is therefore fundamentally sound.

The remainder of this paper is organized as follows: Sec. 2 positions this work regarding the state of the art; Sec. 3 describes copycat implementation and how to use it; Sec. 4 demonstrates copycat capabilities through a UDP vs. TCP use case; finally, Sec. 5 concludes this work by summarizing its main achievements

2 RELATED WORK

Many network measurement tools have been proposed to evaluate reachability or transport protocol performance and analysis. Netalyzr [12] determines whether a particular service, identified by its port number and transport protocol, is reachable. iPerf [23] computes maximum achievable bandwidth alongside other QoS metrics for TCP, UDP, or SCTP. tbit [14] infers the impact of current network environment on TCP behavior. Happy Eyeballs algorithm [27] helps dual-stack applications to choose the network protocol to use between IPv4 and IPv6 while avoiding connectivity problems, or to select appropriate and supported transport protocols [17]. More recently, PATHspider [13] performs A/B testing to identify path transparency impairments.

Sarma evaluates TCP and UDP performances through simulation in a particular context (QoS) considering two queuing mechanisms: RED and Drop Tail [20]. Bruno et al. develop models for analyzing and measuring UDP/TCP throughput in WiFI networks [2]. While some of these results provide insights and background knowledge on aspects of UDP as well as TCP performance, they cannot be used to answer the question of differential treatment between two protocols in the Internet (covering different access network technologies), unlike copycat.

Hätönen et al. [7] investigate differential treatment by NATs. They look at NAT timeouts on a variety of 34 home gateway devices available in 2010, and found a median idle timeout for bidirectional UDP traffic of about 3 minutes, with a minimum timeout of 30 seconds. In contrast, median idle timeout for TCP was 60 minutes, with a minimum of 5 minutes.

Network and transport-layer performance in general is a well-studied field: Qian et al. [19] look at the characteristics of measured TCP traffic. Paxson et al. [18] focuses on packet dynamics of TCP bulk transfers between a limited set of Internet sides. Pahdye et al. [16] investigate TCP behavior of web servers, assuming no interference in the network. Xu at al. [28] use UDP-based traffic to evaluate characteristics of cellular networks. They also test TCP throughput to ensure that no UDP throttling was performed in the tested network that would tamper their results. Melia et al. [15] evaluate TCP and UDP performance in an IPv6-based mobile environment.

Packet encapsulation for network measurements as employed by copycat is a common technique, as well, particularly for middlebox identification. For instance, the TCPExposure [9] client sends TCP packets over raw IP sockets toward a user controlled server. The server sends back received and to-be-sent headers as payload so that the client can compare what was sent to what was received.

copycat is the first attempt to directly evaluate performance differences of congestion-controlled, reliable traffic based solely on the wire image (i.e., whether a TCP, UDP, or another protocol header is seen on the traffic by the network).

3 COPYCAT

In this section, we introduce copycat, our generic, multiplatform, and lightweight connectivity and Quality of Service (QoS) differential treatment measurement tool. It is able to compare many different protocols statelessly without having to implement them or having to write code. It supports multiple operating systems, it is PlanetLab-compliant, and it minimizes the overhead introduced by tunneling.

copycat simultaneously runs pairs of flows between two endpoints, one is a reference TCP flow and the other one is a TCP flow using a custom header as an "outer" transport, to evaluate differences in connectivity and QoS due to differential treatment based on transport protocol. This tunneled TCP flow emulates the wire image of the new protocol to the network but retains the TCP traffic characteristics to enable a valid comparison between the two tested flows. The two flows run in parallel with the exact same 4-tuples, to obtain flows with the most similar possible treatment from the network apart from middleboxes, but with different transport headers. By comparing performances of these two flows, we are able to isolate differences that can be attributed to differential treatment by the path.

copycat encapsulation details are shown in Fig. 1. **Reference** refers to the regular TCP flow used as a groundtruth for flow performance comparison, while **Experimental** is the custom flow, TCP-controlled by the tunneled IP and TCP headers, whose outer encapsulation, entirely specified in command lines. It can be a UDP header, plus an optional additional header placed inside the UDP datagram, before the tunneled headers. This encapsulation can be used for testing varieties of protocols relying on UDP (e.g., PLUS [25], QUIC [11]). Or it can also be a custom, non-UDP transport header with any IP protocol number. This encapsulation can be

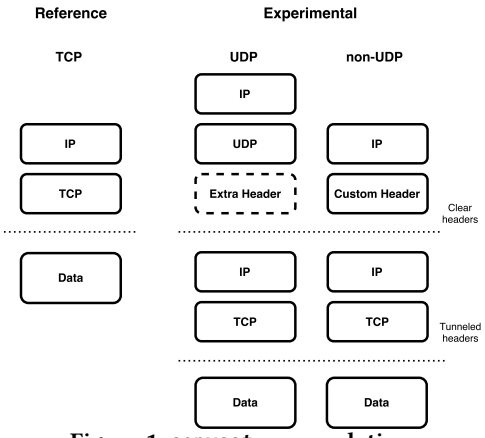

Figure 1: copycat encapsulations.

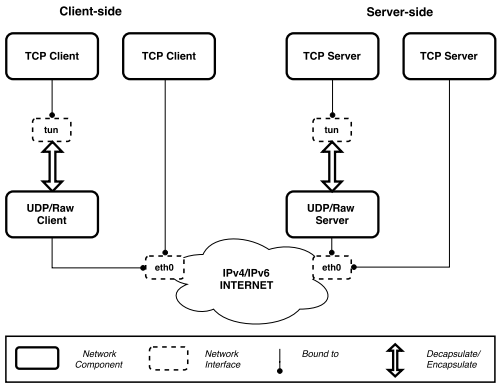

Figure 2: copycat measurement methodology.

used for stateless testing of many IP transport protocols (DCCP [11], native SCTP [21], UDP options [24]).

As shown in Fig. 2, the custom flow is obtained by tunneling a TCP flow. To achieve this, copycat first creates a tun virtual network interface that simulates a network layer device and operates at Layer 3. In our measurement setup, each node runs both the copycat client and the server. On the client side, the TCP client connects to its peer via the Internet-facing interface and receives data from it, writing it to the disk. The custom client consists of the TCP client bound to the tun interface, which is in turn bound by copycat to either a UDP or a raw socket on the Internet-facing interface. copycat thus works as a tunnel endpoint, encapsulating TCP packets from tun in UDP or custom headers, and decapsulating received UDP or custom packets back to TCP packets to tun. The server-side consists of a similar arrangement, listening for connections from clients and sending data to them. The client waits for both transfers, via TCP and TCP-controlled UDP or custom protocol, to be completed before connecting to the next destination.

copycat offers modularity by allowing the user to configure multiple parameters such as the role of each node (client, server, or both), the flow scheduling, and the IP version. The flow scheduling can either be to run both reference and experimental flows *in parallel* or to run one after the other has completely finished, *sequentially*. The IP version can either be IPv4, IPv6, or both. For the latter, each clients opens two pairs of flow instead of one, the first consists in an IPv6 reference flow and an IPv4 experimental flow and the second in a IPv4 reference flow and an IPv6 experimental flow.

A copycat measurement campaign involves two types of actors: a set of *servers* hosting data in the form of simple files and a set of *clients* sequentially connecting to each server to download the files. The client and server sets can be identical (See Sec. 4.1 for a complete case study).

To avoid unwanted fragmentation and ICMP message-too-long errors, and to ensure that packets from both tunneled and non-tunneled flows are equally sized, the Maximum Segment Size (MSS) of the tunneled TCP flow is decreased by the size of the tunnel headers (IP header + transport headers).

copycat is coded in C[2] to minimize the tunneling overhead. I/O multiplexing is handled using select(). All network traces are captured at the internet-facing interface using libpcap on both clients and servers.

4 UDP FOR INTERNET TRANSPORT EVOLUTION

In this section, we show how copycat can find a suitable usage in comparing the performance of UDP encapsulation (i.e., TCP inside UDP) with regular TCP. In particular, with copycat, we are able to determine whether UDP encapsulation will work in the present Internet, and that connectivity and performance of UDP traffic are not disadvantaged with respect to TCP based only on the presence of a UDP header. copycat can be used to create TCP traffic with UDP's wire image, and perform full-mesh measurements on a wide variety of test networks, in order to determine if differential treatment of UDP and TCP packets might disadvantage congestion-controlled traffic with UDP headers.

In particular, Sec. 4.1 explains how measurements were performed. Results (UDP blocking, throughput, and initial latency) are presented in Sec. 4.2 to Sec. 4.4. Finally, in Sec. 4.5, we discuss lessons learned from this case study.

4.1 Measurement Setup

We tested the viability of UDP for transport evolution with copycat by straightforwardly using the UDP encapsulation mode with no extra header.

We deployed copycat on the PlanetLab distributed testbed by selecting 93 nodes (one per subnetwork) from the entire pool (153) of available nodes between March 6th and April 23rd, 2016. The selected nodes are located in 26 countries across North America (44), Europe (29), Asia (13), Oceania (4), and South America (3). Considering PlanetLab port binding restrictions (e.g., 80, 8000, and 53, 443 on certain nodes), we chose seven ports–53, 443, 8008, 12345, 33435, 34567, and 54321– respectively DNS, HTTPS, HTTP alternate, a common backdoor, the Atlas UDP traceroute default, and an unused and an unassigned port, to maximize routers policy diversity.

For each port and pair of nodes, we generated flows of different sizes. The smallest flow is calibrated not to exceed the TCP initial window size, to ensure that all data segments will be sent at once.

[2]See https://github.com/mami-project/copycat

Dataset	Throughput (kB/s)				Latency (ms)				Connectivity		
	< 200		> 200		< 50		> 50		# Probes		No UDP Connectivity
	# flows	median	# flows	median	# flows	median	# flows	median	total	failed	% of probes
PlanetLab	740,721	0.05	34,896	0.16	745,947	0.00	29,370	-1.65	30,778	825	2.66%
DO v4	12,563	0.03	3,637	-0.37	9,381	-0.02	6,819	-0.44	135	0	0.00%
DO v6	15,459	0.07	224	-0.16	15,656	0.00	27	3.63	135	0	0.00%

Table 1: Raw number of bias measurements (throughput and initial latency) per sub dataset ("DO" stands for Digital Ocean). The 50ms cut-off roughly corresponds to inter-continental versus intra-continental latency. Global overview of UDP blocking is also provided.

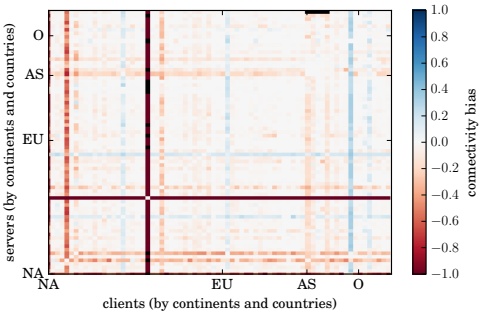

Figure 3: Connectivity bias among PlanetLab nodes, excluding ports 53 and 443. Positive (blue) values mean UDP is better-connected than TCP. Black dots mean "no connectivity" (for both UDP and TCP).

Then, we increase the size of the flows by arbitrary factors to observe the impact of differential treatment for congestion-controlled traffic with larger flows. Overall, we generated 20 pairs of flows of 1, 3, and 30 TCP initial windows, and 10 pairs of flows of 300 and 1, 500 TCP initial windows, for a total of 1,634,518 flows.

We also deployed `copycat` on 6 Digital Ocean (DO) nodes, located in six countries across North America (2), Europe (3), and Asia (1). Given the less restrictive port binding policies and the more restrictive bandwidth occupation policies, we tested ports 80 and 8000 in addition of the PlanetLab ports. For each port, we generated 20 pairs of flows of 1, 3, and 30 TCP initial windows size between May 2nd and 12th, 2016. We repeated the same methodology for both IPv4 and IPv6. This dataset consists in 32,400 IPv4 and 31,366 IPv6 flows.[3]

Table 1 provides an overview on our main results. In summary, we show that, aside from blocking of UDP on certain ports, as well as relatively rare blocking of all UDP traffic on about one in thirty access networks, UDP is relatively unimpaired in the Internet. We explore the details of the table and additional measurement results in the subsections below.

4.2 UDP Blocking

Fig. 3 shows an heatmap describing connection bias per path in the `copycat` results. A bias of +1.0 (blue) means all UDP connections between a given receiver (X-Axis) and sender (Y-Axis) succeeded while all TCP connections failed, and a bias of -1.0 (red) means all TCP connections succeeded while all UDP connections failed. The axes are arranged into geographic regions: North America (NA), Europe (EU), Asia (AS), Oceania and South America (O). The connectivity matrix for PlanetLab nodes provided by Fig. 3 suggests that impairment is access-network linked. One node blocks all inbound and outbound UDP traffic, and has TCP connectivity problems to some servers as well. Otherwise, transient connectivity impairment shows a clear dependency on node, as opposed to path.

[3]The dataset is available at https://observatory.mami-project.eu/

port	UDP blocked	# probes
53[4]	0.55%	1,829
443[5]	4.12%	3,034
8008	2.60%	5,307
12345	2.45%	5,233
33435	2.77%	5,309
34567	2.44%	5,115
54321	3.07%	4,951

Table 2: Percentage of probes (identified as 3-tuples (IPsrc, IPdst, Portdst)) on PlanetLab, that have never seen a UDP connection but at least one TCP connection.

Table 2 provides the percentage of probes that have never seen a UDP connection but, at least, one TCP connection. Statistics are provided by port, as measured by `copycat` on PlanetLab. As shown, UDP is more blocked than TCP but to a small extend. UDP is blocked in, roughly, between 1% and and 5% of the probes. We observed two China-based nodes blocking all UDP traffic from one node also based in China. This advocates for a fall-back mechanism when running the Internet over UDP (i.e., switching back from UDP-based encapsulation to native TCP packets). Note that quite large absence of UDP connection for port 443 (QUIC) is mainly due to PlanetLab port binding restrictions on nodes without any connectivity problem. Anecdotally, we found one New Zealand node blocking both UDP and TCP traffic from all China-based nodes.

4.3 Throughput

To evaluate the impact of transport-based differential treatment on throughput, we introduce the relative *throughput_bias* metric for each pairs of concurrent flows. This is computed as follows:

$$throughput\_bias = \frac{(throughput_{udp} - throughput_{tcp})}{min(throughput_{tcp}, throughput_{udp})} \times 100. \quad (1)$$

[4]Node pool reduced to 41 because of PlanetLab port 53 usage policies.
[5]Node pool reduced to 55 because of PlanetLab port 443 usage policies.

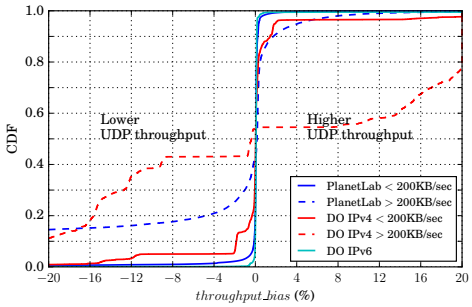

Figure 4: Relative throughput bias, as measured by copycat ("DO" stands for Digital Ocean). Positive values mean UDP has higher throughput. DO IPv6 has not been split in two due to lack of enough values (see Table 1).

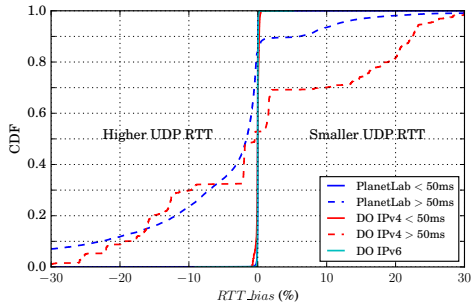

Figure 6: UDP/TCP initial *RTT_bias* as measured by copycat ("DO" stands for Digital Ocean). Positive values mean UDP is faster. DO IPv6 has not been split in two due to lack of enough values (see Table 1).

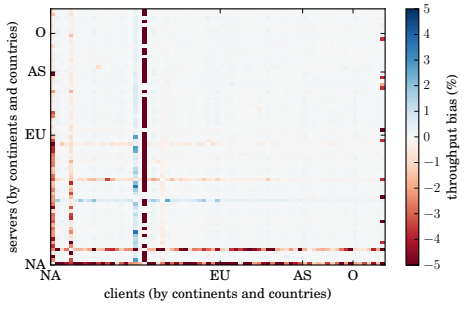

Figure 5: Relative throughput bias among PlanetLab nodes as measured by copycat. Positive (blue) values mean UDP has higher throughput.

A positive value for *throughput_bias* means that UDP has a higher throughput. A null value means that both UDP and TCP flows share the same throughput.

Fig. 4 provides a global view of the *throughput_bias*. Dataset has been split between flows < 200 KB/sec and flows > 200KB/sec, except for Digital Ocean IPv6, as the number of measurements is too small to be representative. Table 1 gives the size of each sub dataset and the relative median bias for throughput and latency.

As observed, in general, there is no bias between UDP and TCP. For both Digital Ocean dataset, the non-null biases are mostly evenly distributed in favor and disfavor of UDP. In PlanetLab, we observe an extreme case where TCP performs better than UDP, the 4% and 2% highest *throughput_bias* in absolute value are respectively higher than 1% and 10%. As shown in Fig. 5, those extreme cases, represented as dark red lines, are endpoint-dependent. We also notice a single probe where the UDP throughput is better than TCP (see Fig. 5). Consistently with UDP connectivity bias (see Fig. 3), we do not see evidence on path dependency for throughput.

The loss rate of congestion controlled traffic in steady state, where the link is fully utilized, is mostly determined by the congestion control algorithm itself. Therefore, there is a direct relation between throughput and loss. However, as TCP congestion control reacts only once per RTT to loss as an input signal, the actual loss rate could still be different even if similar throughput is achieved.

Here, we understand *loss* as the percentage of flow payload lost, computed from sequence numbers. A value, for instance, of 10% of losses means thus that 10% of the flow payload has been lost.

Generally speaking, the loss encountered is quite low, given that small flows often are not large enough to fully utilize the measured bottleneck link. As expected based on the throughput observed, we see no significant loss difference in both PlanetLab and Digital Ocean when comparing TCP and UDP, except of 3.5% in favor of UDP for the largest flow size (6MB). However, this is inline with a slightly lower throughput caused by a slightly larger initial RTT, as discussed in the next section.

4.4 Initial Latency

Since all copycat traffic is congestion controlled, throughput is influenced by the end-to-end latency. We use initial RTT measured during the TCP handshake as baseline for this metric.

In the fashion of *throughput_bias* (see Eqn. 1), we introduce the relative *RTT_bias* metric for each pair of concurrent flows. This is computed as follows:

$$RTT\_bias = \frac{(RTT_{tcp} - RTT_{udp})}{min(RTT_{tcp}, RTT_{udp})} \times 100. \quad (2)$$

A positive value for *RTT_bias* means that UDP has a smaller initial latency (i.e., performs better than TCP). A null value means that both UDP and TCP flows share the same initial latency.

The median latency bias is also listed in in Table 1. For PlanetLab, there is no latency bias for flows with an initial RTT of 50ms or less and a slight bias towards higher latency for UDP for flows with larger initial RTTs. For Digital Ocean we also observed a slight bias towards higher latency for UDP for IPv4 and no bias for IPv6 (considering 27 flows with a larger RTT than 50ms as not representative). This is confirmed by the CDF shown in Fig. 6.

The 2% and 1% most biased flow pairs have an *RTT_bias* respectively lower than -1% and -10%. For the Digital Ocean IPv4 campaign, 40% of the generated flows have an *RTT_bias* between 1% and 30% in absolute value. The difference between IPv4 and IPv6

on Digital Ocean appears to be due to the presence of a middlebox interfering with all IPv4 traffic, both TCP and UDP.

This difference in latency also explains the slight throughput disadvantage as seen in the previous section given latency results follow nearly the same shape as the initial RTT (see Fig. 6).

4.5 Lessons Learned

In this case study, we asked the question "is UDP a viable basis and/or encapsulation for deploying new transports in the Internet?". We focused on two aspects of the answer: connectivity and differential treatment of TCP and TCP-congestion-controlled UDP packets to see if simply placing such traffic in UDP headers disadvantages it. In this section, based on the `copycat` data presented above, we discuss lessons learned.

First, **UDP provides a viable common basis** for new transport protocols, but only **in cases where alternatives exist** on access networks where UDP connectivity is unavailable or severely compromised. QUIC provides a good illustration here. It was developed together with SPDY, which has been defined over TCP and TLS as HTTP/2 [1], and its first target application is HTTP/2. Since HTTP/2 has a natural fallback to TLS over TCP, this alternative can be used on the $1-5\%$ of networks where QUIC packets over UDP are blocked or limited. However, this fallback approach limits QUIC's applicability to application layer protocols that can be made run acceptably over TCP.

Second, the dataset collected with `copycat` provides evidence that the **vast majority of UDP impairments are access-network linked**, and that **subtle impairment is rare**. This means that accurate fallback decisions are easy to arrive at – a connection racing design similar to Happy Eyeballs [27] as used by QUIC is sufficient – and can often be cached based on client access network as opposed on access-network/server pair.

We made no attempt to confirm claims of defensive rate-limiting of UDP traffic with this work, as doing so would in essence require UDP-based denial of service attacks on the networks under measurement. However, we note that Google reports a reduction in the amount of UDP rate limiting they have observed since the beginning of the QUIC experiment [22]. This makes sense: rate limitation must necessarily adapt to baseline UDP traffic volumes, and as such poses no limitation to the gradually increasing deployment of UDP-based transport protocols. However, it also indicates the need for work on mechanisms in these UDP protocols to augment the denial-of-service protection afforded by rate-limiting approaches.

5 CONCLUSION

This paper introduced `copycat`, a novel measurement tool to detect differential treatment by middleboxes of new transport protocols compared to TCP. `copycat` generates TCP-shaped traffic with custom headers and compares it with standard TCP. One of the key feature of `copycat` is that it easily allows research to evaluate not only transport connectivity but also QoS characteristics (e.g., throughput, loss, delay) while writing a minimum amount of code. `copycat` is open-source and freely available (see https://github.com/mami-project/copycat).

To demonstrate `copycat` capabilities, we focused on a simple and straightforward question: "is UDP a viable basis and/or encapsulation for deploying new transports in the Internet?". From our measurements, we conclude that UDP is relatively unimpaired in the Internet, despite blocking of UDP on certain ports as well as relatively rare blocking of all UDP traffic on certain access networks. This means that, indeed, running the Internet over UDP is globally possible. Further, we find that impairments to UDP-based traffic are access-network linked. Therefore, simple dynamic fallback mechanisms for UDP-encapsulated transports are a viable approach to work around the vast majority of impairments encountered. A node needs not measure or remember anything about its peers, but only about its connectivity to the Internet, to determine when to fall back.

`copycat` complements the set of existing measurement tools as it can separate effects of differential network treatment from differences in the traffic characteristics of new transport protocols by using a tunneling approach. We have shown that `copycat` is applicable to answer underlying measurement questions on connectivity as well as QoS for UDP encapsulation as an approach for deployment of new transports. Next we will perform further testing to access the actual impairments and the prevalence of these impairments for native deployment of new protocols, enabling new possibilities in transport evolution and the development of more advances fallback mechanisms.

ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 688421. The opinions expressed and arguments employed reflect only the authors' views. The European Commission is not responsible for any use that may be made of that information.

REFERENCES

[1] M. Belshe, R. Peon, and M. Thomson. 2015. *Hypertext Transfer Protocol Version 2 (HTTP/2).* RFC 7540. Internet Engineering Task Force.
[2] R. Bruno, M. Conti, and E. Gregori. 2008. Throughput Analysis and Measurements in IEEE 802.11 WLANs with TCP and UDP Traffic Flows. *IEEE Transactions on Mobile Computing* 7, 2 (February 2008), 171–186.
[3] Digital Ocean. 2017. Simple Cloud Computing, Built for Developpers. (2017). See https://www.digitalocean.com.
[4] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar. 2011. *Architectural Guidelines for Multipath TCP Development.* RFC 6182. Internet Engineering Task Force.
[5] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. 2013. *TCP Extensions for Multipath Operation with Multiple Addresses.* RFC 6824. Internet Engineering Task Force.
[6] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk. 2016. *QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2.* Internet Draft (Work in Progress) draft-hamilton-early-deployment-quic-00. Internet Engineering Task Force.
[7] S. Hätönen, A. Nyrhinen, L. Eggert, S. Strowes, P. Sarolahti, and M. Kojo. 2010. An Experimental Study of Home Gateway Characteristics. In *Proc. ACM Internet Measurement Conference (IMC).* 260–266.
[8] D. A. Hayes, J. But, and G. Armitage. 2009. Issues with Network Address Translation for SCTP. *ACM SIGCOMM Computer Communication Review* 39, 1 (January 2009), 23–33.
[9] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda. 2011. Is It Still Possible to Extend TCP. In *Proc. ACM Internet Measurement Conference (IMC).* 181–194.
[10] R. Jesup, S. Loreto, and M. Tuexen. 2015. *WebRTC Data Channels.* Internet Draft (Work in Progress) draft-ietf-rtcweb-data-channel-13. Internet Engineering Task Force.
[11] E. Kohler, M. Handley, and S. Floyd. 2006. *Datagram Congestion Control Protocol (DCCP).* RFC 4340. Internet Engineering Task Force.

[12] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. 2010. Netalyzr: illuminating the edge network. In *Proc. ACM Internet Measurement Conference (IMC)*. 246–259.
[13] I. R. Learmonth, B. Trammell, M. Külewind, and G. Fairhust. 2016. PATHSpider: A Tool for Active Measurement of Path Transparency. In *Proc. Applied Networking Research Workshop (ANRW)*. 62–64.
[14] A. Medina, M. Allman, and S. Floyd. 2004. Measuring Interactions Between Transport Protocols and Middleboxes. In *Proc. ACM Internet Measurement Conference (IMC)*. 336–341.
[15] T. Melia, R. Schmitz, and T. Bohnert. 2004. TCP and UDP Performance Measurements in Presence of Fast Handovers in an IPv6-Based Mobility Environment. In *Proc. World Telecommunications Congress (WTC)*.
[16] J. Pahdye and S. Floyd. 2001. On Inferring TCP Behavior. In *Proc. ACM SIGCOMM*.
[17] G. Papastergiou, K.-J. Grinnmo, A. Brunstrom, D. Ros, M. Tüxen, N. Khademi, and P. Hurtig. 2016. On the Cost of Using Happy Eyeballs for Transport Protocol Selection. In *Proc. Applied Networking Research Workshop (ANRW)*.
[18] V. Paxson. 1997. End-to-End Internet Packet Dynamics. In *Proc. ACM SIGCOMM*.
[19] F. Qian, A. Gerber, Z. Mao, S. Sen, O. Spatscheck, and W. Willinger. 2009. TCP Revisited: a Fresh Look at TCP in the Wild. In *Proc. ACM Internet Measurement Conference (IMC)*. 76–89.
[20] M. P. Sarma. 2013. Performance Measurement of TCP and UDP Using Different Queuing Algorithm in High Speed Local Area Network. *International Journal of Future Computer and Communication* 2, 6 (2013), 682.
[21] R. Stewart. 2007. *Stream Control Transmission Protocol.* RFC 4960. Internet Engineering Task Force.
[22] I. Swett. 2016. QUIC Deployment Experience @Google. (July 2016). See https://www.ietf.org/proceedings/96/slides/slides-96-quic-3.pdf.
[23] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs. 2005. Iperf: the TCP/UDP Bandwidth Measurement Tool. (2005). See http://dast.nlanr.net/Projects.
[24] J. Touch. 2016. *Transport Options for UDP.* Internet Draft (Work in Progress) draft-touch-tsvwg-udp-options-02. Internet Engineering Task Force.
[25] B. Trammell and M. Kühlewind. 2016. *Path Layer UDP Substrate Specification.* Internet Draft (Work in Progress) draft-trammell-plus-spec-00. Internet Engineering Task Force.
[26] M. Tuexen, R. Seggelmann, and E. Rescorla. 2011. *Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP).* RFC 6083. Internet Engineering Task Force.
[27] D. Wing and A. Yourtchenko. 2012. *Happy Eyeballs: Success with Dual-Stack Hosts.* RFC 6555. Internet Engineering Task Force.
[28] Y. Xu, Z. Wang, W. K. Leong, and B. Leong. 2014. An End-to-End Measurement Study of Modern Cellular Data Networks. In *Proc. Passive and Active Measurement Conference (PAM)*. Springer, 34–45.