QUIC Transport Protocol

- Development and basic features
- Connection establishment; data transfer
- Avoiding ossification; limitations

QUIC over UDP

- **Why run QUIC over UDP rather than over IP?**
- To ease end-system deployment in user-space applications
  - User-space applications, running over UDP, are easy to build
    - BSD sockets; portable → same API works everywhere
    - Widely understood programming model
    - No need for privileged access
  - No portable, unprivileged, interface to build applications that run directly over IP
    - Implementations have to run within the operating system kernel
    - Deploying kernel updates is difficult
    - Deploying application updates is straightforward
- To work around protocol ossification due to middleboxes
  - Firewalls block anything other than TCP and UDP

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2626780, I
Communications Surveys & Tutorials

1

De-ossifying the Internet Transport Layer: A Survey and Future Perspectives

Giorgos Papastergiou, Gorry Fairhurst, David Ros, Anna Brunstrom, Karl-Johan Grinnemo, Per Hurtig,
Naeem Khademi, Michael Tüxen, Michael Welzl, Dragana Damjanovic and Simone Mangiante

***Abstract*—It is widely recognized that the Internet transport layer has become ossified, where further evolution has become hard or even impossible. This is a direct consequence of the ubiquitous deployment of middleboxes that hamper the deployment of new transports, aggravated further by the limited flexibility of the Application Programming Interface (API) typically presented to applications. To tackle this problem, a wide range of solutions have been proposed in the literature, each aiming to address a particular aspect. Yet, no single proposal has emerged that is able to enable evolution of the transport layer. In this work, after an overview of the main issues and reasons for transport-layer ossification, we survey proposed solutions and discuss their potential and limitations. The survey is divided into five parts, each covering a set of point solutions for a different facet of the problem space: 1) designing middlebox-proof transports, 2) signaling for facilitating middlebox traversal, 3) enhancing the API between the applications and the transport layer, 4) discovering and exploiting end-to-end capabilities, and 5) enabling user-space protocol stacks. Based on this analysis, we then identify further development needs towards an overall solution. We argue that the development of a comprehensive transport layer framework, able to facilitate the integration and cooperation of specialized solutions in an application-independent and flexible way, is a necessary step toward making the Internet transport architecture truly evolvable. To this end, we identify the requirements for such a framework and provide insights for its development.**

***Index Terms*—Transport protocols, protocol-stack ossification, API, middleboxes, user-space networking stacks.**

I. Introduction and background

Networks can and do vary significantly in the set of functions they offer and their ability to move data between endpoints. The transport layer operates across the network and is responsible for efficient and robust end-to-end communication between network endpoints. The term end-to-end is often associated with a principle, called the end-to-end argument [1]. This suggests that "functions placed at low levels of a system may be redundant, or of little value, when compared with the cost of providing them at that low level." This argument followed Schroeder et al.'s earlier work on system design and security [2], and is now generally considered as a simple guide for which services should be realised at the transport layer.

The transport layer was designed to hide the details and variability of the network service from the applications that need to use it. The Internet's transport layer also contains other functions that are difficult or impossible to provide within a network, such as reliability, verification of delivery, flow control to prevent the application from overwhelming the remote endpoint, congestion control to prevent the application from overwhelming the network, etc. People using the Internet mostly run applications that are based on the Transmission Control Protocol, TCP [3], which provides these transport functions.

Some applications need a different set of services to those offered by TCP. For example, a web client may wish to be able to prioritize sub-flows carrying specific objects, a multimedia flow may prefer timeliness to reliable delivery, and IP telephony can be tolerant to packet loss or in some cases to bit errors. There are many cases where TCP simply does not meet the need of applications—yet it ends up being used because it "just works," but not necessarily very well [4]. Applications that do not want the transport semantics of TCP typically just use the User Datagram Protocol, UDP [5]. While UDP provides flexibility that allows any set of services to be defined, every function needed has to be implemented at the application layer.

Some initiatives have developed alternate protocols to TCP, suited for other application types, for instance: the Datagram Congestion Control Protocol (DCCP) [6] was proposed to support streaming multimedia; the Stream Control Transport Protcol (SCTP) [7], [8] originally targeted telephony signaling; UDP-Lite [9] supports error-tolerant audio and video services over wireless links. However, despite being standardized, with available implementations for common platforms, these transports are seldom seen in the general Internet, and TCP and UDP remain the only widely used transports.

A. Transport-layer ossification: overview of issues

Why do developers and users not adopt more modern protocols? It is not because new transports do not meet a real need. The following paragraphs examine the main reasons for this *ossification* of the transport layer.

G. Papastergiou and D. Ros are with Simula Research Laboratory, P.O. Box 134, 1325 Lysaker, Norway. E-mail: {gpapaste,dros}@simula.no.

G. Fairhurst is with University of Aberdeen, AB24 3UE Aberdeen, United Kingdom. E-mail: gorry@erg.abdn.ac.uk.

A. Brunstrom, K.-J. Grinnemo and P. Hurtig are with Karlstad University, Universitetsgatan 2, 651 88 Karlstad, Sweden. E-mail: {anna.brunstrom,karl-johan.grinnemo,per.hurtig}@kau.se.

N. Khademi and M. Welzl are with University of Oslo, P.O. Box 1072 Blindern, 0316 Oslo, Norway. E-mail: {naeemk,michawe}@ifi.uio.no.

M. Tüxen is with Münster University of Applied Sciences, Bismarckstraße 11, 48565 Steinfurt, Germany. E-mail: tuexen@fh-muenster.de.

D. Damjanovic is with Mozilla Corporation. E-mail: ddamjanovic@mozilla.com.

S. Mangiante is with EMC Corporation, Ovens, Co. Cork, Ireland. E-mail: simone.mangiante@emc.com.

1553-877X (c) 2016 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

G. Papastergiou *et al*, "De-ossifying the Internet transport layer: A survey and future perspectives", IEEE Communications Surveys and Tutorials (Nov. 2016). https://dx.doi.org/10.1109/COMST.2016.2626780/

Ossification (1/2)

- Deployment experience → if a field in a protocol is visible to the network, someone will implement a middlebox that relies on its presence
- Once a protocol has been widely deployed, very hard to change

Is it Still Possible to Extend TCP?

Michio Honda*, Yoshifumi Nishida*, Costin Raiciu†, Adam Greenhalgh‡,
Mark Handley‡, Hideyuki Tokuda*
Keio University*, Universitatea Politehnica Bucuresti†, University College London‡
{micchie,nishida}@sfc.wide.ad.jp, costin.raiciu@cs.pub.ro
{a.greenhalgh,m.handley}@cs.ucl.ac.uk, hxt@ht.sfc.keio.ac.jp

ABSTRACT

We've known for a while that the Internet has ossified as a result of the race to optimize existing applications or enhance security. NATs, performance-enhancing-proxies, firewalls and traffic normalizers are only a few of the middleboxes that are deployed in the network and look beyond the IP header to do their job. IP itself can't be extended because "IP options are not an option" [10]. Is the same true for TCP?

In this paper we develop a measurement methodology for evaluating middlebox behavior relating to TCP extensions and present the results of measurements conducted from multiple vantage points. The short answer is that we can still extend TCP, but extensions' design is very constrained as it needs to take into account prevalent middlebox behaviors. For instance, absolute sequence numbers cannot be embedded in options, as middleboxes can rewrite ISN and preserve undefined options. Sequence numbering also must be consistent for a TCP connection, because many middleboxes only allow through contiguous flows.

We used these findings to analyze three proposed extensions to TCP. We find that MPTCP is likely to work correctly in the Internet or fallback to regular TCP. TcpCrypt seems ready to be deployed, however it is fragile if resegmentation does happen—for instance with hardware offload. Finally, TCP extended options in its current form is not safe to deploy.

Categories and Subject Descriptors

C.2.2 [**Computer-communication Networks**]: Network Protocols; C.2.6 [**Computer-communication Networks**]: Internetworking

General Terms

Measurement, Design, Experimentation, Standardization

Keywords

Middleboxes, Measurements, TCP, Protocol design

IMC'11, November 2–4, 2011, Berlin, Germany.
Copyright 2011 ACM 978-1-4503-1013-0/11/11 ...$10.00.

1. INTRODUCTION

The Internet was designed to be extensible; routers only care about IP headers, not what the packets contain, and protocols such as IP and TCP were designed with options fields that could be used to add additional functionality. The great virtue of the Internet was always that it was *stupid*; it did no task especially well, but it was extremely flexible and general, allowing a proliferation of protocols and applications that the original designers could never have foreseen.

Unfortunately the Internet, as it is deployed, is no longer the Internet as it was designed. IP options have been unusable for twenty years[10] as they cause routers to process packets on their slow path. Above IP, the Internet has benefited (or suffered, depending on your viewpoint) from decades of optimizations and security enhancements. To improve performance [2, 7, 18, 3], reduce security exposure [15, 29], enhance control, and work around address space shortages [22], the Internet has experienced an invasion of middleboxes that *do* care about what the packets contain, and perform processing at layer 4 or higher *within* the network.

The problem now faced by designers of new protocols is that there is no longer a well defined or understood way to extend network functionality, short of implementing everything over HTTP[25]. Recently we have been working on adding both multipath support[11] and native encryption[5] to TCP. The obvious way to do this, in both cases, is to use TCP options. In the case of multipath, we would also like to stripe data across more than one path. At the end systems, the protocol design issues were mostly conventional. However, it became increasingly clear that no one, not the IETF, not the network operators, and not the OS vendors, knew what will and what will not pass through all the middleboxes as they are currently deployed and configured. Will TCP options pass unchanged? If the sequence space has holes, what happens? If a retransmission has different data than the original, which arrives? Are TCP segments coalesced or split? These and many more questions are crucial to answer if protocol designers are to extend TCP in a deployable way. Or have we already lost the ability to extend TCP, just like we did two decades ago for IP?

In this paper we present the results from a measurement study conducted from 142 networks in 24 countries, including cellular, WiFi and wired networks, public and private networks, residential, commercial and academic networks. We actively probe the network to elicit middlebox responses that violate the end-to-end transparency of the original Internet architecture. We focus on TCP, not only because it is by far the most widely used transport protocol, but also because while it is known that many middleboxes modify TCP behavior [6], it is not known how prevalent such middleboxes are, nor precisely what the *emergent behavior* is with TCP extensions that were unforeseen by the middlebox designers.

M. Honda *et al.,* "Is it still possible to extend TCP?", In Proc. ACM Internet Measurement Conference, Berlin, Nov. 2011. https://dx.doi.org/10.1145/2068816.2068834

Ossification (2/2)

- Protocol ossification affected the design of:
  - TLS 1.3
  - Multipath TCP
  - TCP Fast Open
  - TCP Selective Acknowledgements
  - …

- Increasingly viewed as a problem in the standards community
  - Difficult to evolve network protocols to address new requirements
  - A system that can no longer evolve and change will die

Avoiding Ossification in QUIC

- Three tools to prevent ossification of QUIC:
  - Published protocol invariants
  - Pervasive encryption of transport headers
  - GREASE

- QUIC is a new design – want to **avoid ossification**, starting with initial deployments
- Design the protocol to make it difficult, ideally impossible, for middleboxes to interfere with QUIC connections

QUIC Invariants

- Properties of QUIC that will remain unchanged as new versions of the protocol of developed
  - Explicit guidance to middlebox designers: can assume these properties of QUIC will not change
  - The IETF **will** change other fields or properties between QUIC versions
- What invariants are guaranteed?
  - Packets start with a long header or a short header
    - The first bit of long header packets is always 1; they include version number, destination- and source-connection identifiers
    - The first bit of short header packets is always 0; they include a destination connection identifier
  - Connections can arbitrarily switch between long header and short header packets

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|1|X X X X X X X|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Version (32)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| DCID Len (8)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Connection ID (0..2040)           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SCID Len (8)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Source Connection ID (0..2040)              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

QUIC Long Header Invariants

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|0|X X X X X X X|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Destination Connection ID (*)               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

QUIC Short Header Invariants

X = bit may take arbitrary value

https://datatracker.ietf.org/doc/draft-ietf-quic-invariants/

Avoiding Ossification via Pervasive Encryption

- QUIC encrypts as much data as possible
  - Entire packet **except** invariant fields and the last 7-bits of the first byte is encrypted; entire packet is authenticated
  - Encryption keys for initial handshake packets are derived from connection identifiers
    - Contain **`ClientHello`** and **`ServerHello`**, which are unencrypted when using TLS over TCP, and **don't need to be encrypted**
    - QUIC provides no more security than TLS over TCP, but makes it **expensive** for middleboxes to read handshake
  - Rest of QUIC connection protected by TLS 1.3 as normal
- QUIC authenticates **all** data
  - If a middlebox changes the headers, the change can be detected

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|1|X X X X X X X|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Version (32)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| DCID Len (8)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Connection ID (0..2040)           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SCID Len (8)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Source Connection ID (0..2040)              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                       QUIC Long Header Invariants
```

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|0|X X X X X X X|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Destination Connection ID (*)               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                      QUIC Short Header Invariants

                                  X = bit may take arbitrary value
```

https://datatracker.ietf.org/doc/draft-ietf-quic-invariants/

Avoiding Ossification via GREASE

- QUIC makes extensive use of GREASE
  - Every field encrypted or has a value that is unpredictable
    - New connections use randomly chosen connection identifies
    - Clients randomly try to negotiate new version numbers
      - Version numbers matching **`0x?a?a?a?a`** for testing version negotiation – will be rejected by servers
    - Unused header fields given randomly chosen values
  - Goal is that middleboxes can't make any assumptions about QUIC header values → nothing in the header is predictable
  - Hopefully avoids ossification → nothing to ossify around

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|1|X X X X X X X|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Version (32)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| DCID Len (8)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Connection ID (0..2040)           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SCID Len (8)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Source Connection ID (0..2040)              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                        QUIC Long Header Invariants
```

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|0|X X X X X X X|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Destination Connection ID (*)               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                       QUIC Short Header Invariants

                                     X = bit may take arbitrary value
```

https://datatracker.ietf.org/doc/draft-ietf-quic-invariants/

QUIC Benefits and Costs

- **Why is QUIC desirable?**
  - Reduces secure connection establishment latency
  - Reduces risk of ossification; easy to deploy
  - Supports multiple streams within a single connection
- **Why is QUIC problematic?**
  - Libraries and support new, poorly documented, and frequently buggy
  - CPU usage is high compared to TLS-over-TCP
    - TCP stack currently much better optimised
    - TCP and TLS often has hardware offload; QUIC doesn’t yet
  - These issues will be resolved – but it will take some years before QUIC is as stable and performant as TLS-over-TCP
- **TCP lasted 40 years – QUIC is a similarly long-term project, that’s only just reached version 1.0**

Improving Secure Connection Establishment

- Limitations of TLS 1.3
- QUIC transport protocol
- TCP has dominated for 40 years – is QUIC the future?