

University of Oslo, Department of Informatics [Impact case #5]

Institution: University of Oslo
Administrative unit: Department of Informatics (IFI)
Title of case study: Transport Services (TAPS)
Period when the underpinning research was undertaken: 2010 – now
Period when staff involved in the underpinning research were employed by the submitting institution: 2009 – now
Period when the impact occurred: 2018 – now

1. Summary of the impact

TAPS replaces Berkeley sockets with an IETF standard API that avoids the compile-time binding of applications to protocols and network interfaces. This makes the Internet altogether much more flexible and easier to upgrade. It facilitates access to new services and offers Operating System and middleware designers an opportunity to differentiate their TCP/IP stack implementations.

Apple is one of the main contributors to the TAPS standard and has followed TAPS principles in their “Network.Framework” API, which is offered to all Apple developers since 2018.

Rumours talk of at least one more large implementation that is under development. The TAPS standard is now very close to finalisation; it is expected that the biggest impact will happen after publication.

2. Underpinning research

The academic literature contains many proposals for more flexible network APIs, dating several decades back, with no real-life impact whatsoever. The relevant difference for TAPS to have an impact, and thus the novelty of the research leading up to it, was to:

1. Identify how to carry out this work in a way that can realistically be standardised by the Internet Engineering Task Force (IETF) (e.g., this is why, different from all other prior proposals, TAPS follows a *bottom-up* design),
2. Design and implement a TAPS prototype to demonstrate how the underlying ideas work.

This involved:

- Carefully designing a suitable API: on the one hand, as much as possible must be abstracted, as exposing unnecessary elements makes the application-protocol binding more rigid. On the other, some services *must* be exposed, or an application can never make use of them. To achieve this, we:
 - Identified the services offered by a suitable set of transport protocols at the right abstraction level ([RFC8303], *primarily our work, January 2016 – February 2018*).
 - Systematically identified which services require “application-specific knowledge” (knowledge that only applications have), making them indispensable in the API surface, and identified mismatches and redundancies ([RFC8923], *only our work, December 2015 – October 2020*),
 - Carried out the final API design ([1] and [draft-ietf-taps-interface], *where we are a major contributor, April 2018 – now*).
- Exploring (measuring) the feasibility of using various network protocols mechanisms (e.g., the DiffServ Code Point (DSCP)) across the “best effort” Internet ([3][4], *primarily our work*).

- Designing, implementing and evaluating machinery that shows how to dynamically choose and configure protocols and network interfaces, even when a protocol may not be supported over an Internet path. ([2], joint work with partners in the NEAT project)

Most of the underpinning research was carried out in the EC H2020 project “NEAT”, and some more follow-up work was done in the national “toppforsk” project “OCARINA” that was funded by the Norwegian Research Council.

Relevant key researchers at the unit:

Michael Welzl, full professor

Stein Gjessing, full professor

Naeem Khademi, postdoctoral researcher from 2015 to 2018

Runa Barik, Ph.D. student from 2014 to 2019

3. References to the research

- [1] Michael Welzl, Safiqul Islam, Michael Gundersen, Andreas Fischer: "Transport Services: A Modern API for an Adaptive Internet Transport Layer", IEEE Communications Magazine 59(4), April 2021. DOI 10.1109/MCOM.001.2000870
- [2] Naeem Khademi, David Ros, Michael Welzl, Zdravko Bozakov, Anna Brunstrom, Gorrry Fairhurst, Karl-Johan Grinnemo, David Hayes, Per Hurtig, Tom Jones, Simone Mangiante, Michael Tuexen and Felix Weinrank: "NEAT: A Platform- and Protocol-Independent Internet Transport API", IEEE Communications Magazine 55(6), 2017. DOI 10.1109/MCOM.2017.1601052
- [3] Runa Barik, Michael Welzl, Gorrry Fairhurst, Ahmed Elmokashfi, Thomas Dreibholz, Stein Gjessing: "On the Usability of Transport Protocols other than TCP: A Home Gateway and Internet Path Traversal Study", Elsevier Computer Networks, Volume 173, 22 May 2020. DOI 10.1016/j.comnet.2020.107211
- [4] Runa Barik, Michael Welzl, Ahmed Elmokashfi, Thomas Dreibholz, Safiqul Islam, Stein Gjessing: "On the Utility of Unregulated IP DiffServ Code Point (DSCP) Usage by End Systems", Elsevier Performance Evaluation, vol. 135, August 26, 2019. DOI 10.1016/j.peva.2019.102036, ISSN 0166-5316.
- [RFC8303] Michael Welzl, Michael Tuexen, Naeem Khademi (eds.): "On the Usage of Transport Features Provided by IETF Transport Protocols", RFC 8303 (Informational), February 2018. DOI 10.17487/RFC8303
- [RFC8923] Michael Welzl, Stein Gjessing: "A Minimal Set of Transport Services for TAPS Systems", RFC 8923, (Informational), October 2020. DOI 10.17487/RFC8923
- [draft-ietf-taps-interface] B. Trammell, Ed., M. Welzl, Ed., R. Enghardt, G. Fairhurst, M. Kuehlewind, C. Perkins, P. Tiesel, T. Pauly: "An Abstract Application Layer Interface to Transport Services", IETF Internet-draft (work in progress) draft-ietf-taps-interface-24, December 2023, available from <https://datatracker.ietf.org/doc/draft-ietf-taps-interface/24/>

4. Details of the impact

The theoretical foundations of networking that were laid out in the OSI model expect much more flexibility than the Internet has ever achieved: they include the ability to seamlessly change protocols at each layer, because the internals of a layer are not visible to the users of this layer. Until now, Internet standards have never followed this design. Changing protocols at the transport layer is practically impossible: without TAPS, an application that is written against TCP cannot seamlessly use other protocols *and benefit from the mechanisms that these protocols offer*.

Using other, more modern protocols, along with mechanisms that test whether these protocols “work” across an Internet path and otherwise fall back to TCP, is a difficult effort for application programmers. It is therefore only worthwhile for giant industry players such as Google (who have done exactly this with QUIC, in their Chrome browser).

This complexity makes it hard for SMEs to write competitive network code.

TAPS moves all the functionality that is related to selecting, testing, configuring and utilising a transport protocol inside the transport layer, where it belongs. Applications specify their needs and get them satisfied, following the Internet’s “best effort” model; this allows to upgrade the network below the API. Specifically, TAPS:

1. Makes it much easier for programmers to benefit from modern protocols and mechanisms (and their code will automatically benefit from new protocols and mechanisms even when these are deployed in later updates of the network stack).

Beneficiaries: SMEs which need to write efficient network code.

2. Creates new opportunities for diversification of the network stack below the API.

Beneficiaries: Developers of middleware, network libraries, or Operating Systems.

3. Creates new opportunities for diversification inside the network, as supporting a new, previously unused, protocol may suddenly become beneficial for customers. As we and others have shown, this even goes beyond “just” IETF transport protocols, and it can help the deployment of novel network architectures.

Beneficiaries: Network service providers.

To date, only impacts 1) and 2) have happened, as Apple has made use of TAPS principles in their “Network.Framework” API, which is offered to all developers since the beta versions of iOS 12 and macOS Mojave, in 2018.

Note that, after more than a decade in the making, the TAPS standard is not yet finished. If a company offers a TAPS API before the final specification is published, there is a risk of having to change the API for programmers due to a standard update – for this reason, steps towards TAPS have been rather cautious, but this is expected to change upon publication. As Apple has gradually taken more and more of the TAPS functionality on board, the flexibility of their network stack has already been increasing – for example, since 2022, a network connection can use TCP or QUIC without the application being aware of the change.

How has the research underpinned the impact, how was the impact achieved:

We led the entire effort. This involved:

- **Research prototype #1:** technical proof that the TAPS idea 1) *can* efficiently be implemented, and 2) *can* yield technical benefits. Each of these arguments were developed with results from a master thesis that we guided, and a published paper from the thesis.
- **Creating the TAPS Working Group.** This was an intense community building effort: creating a mailing list and webpage, discussing the ideas there, writing several documents and convincing colleagues to write documents (to outline how TAPS could work and why it is needed), as well as organizing three events at IETF meetings (a “bar BoF” and two “BoFs”). This work, which received a large amount of help from Gorrry Fairhurst at the University of Aberdeen in particular, is documented here:

<https://sites.google.com/site/transportprotocolservices/home>

It led to the **creation of the TAPS Working Group on 24. September 2014.**

- **Creating and technically leading the European H2020 research project “NEAT” to accompany the TAPS effort.** We formed the consortium, led the technical proposal writing effort, and were technical manager of this project, which ran from March 2015 until April 2018 (top scores at the final review). Details are here: <http://neat-project.org>
The NEAT code is **research prototype #2**, which allowed us to maintain a leading role in the early design stages in the TAPS Working Group.
- **Continued involvement in TAPS until the time of writing.** Including group creation, this has been more than a decade of intense IETF work. At the time of writing, the group’s github repository shows 813 closed issues and 638 closed PRs; the latest published interface draft is currently at version 25, and there are three drafts in total that the group is working on. We are editor of one draft, co-author of one draft, and a minor contributor to the third draft (which is the shortest, easiest document).

Many dissemination and communication efforts have taken place – including, but not limited to the activities of the NEAT project. If TAPS will be a major success (beyond “just” Apple), community uptake will happen automatically via the IETF after publication of the RFCs. This remains to be seen.

We do not claim to have actively involved Apple: this contact was established by TAPS collaborator Brian Trammell (ETH Zürich at the time, now with Google Switzerland). Also, we cannot claim to truly have had an influence on how Apple developers write their code – this may be the case, or it may not.

What we can claim, however, is that, thanks to TAPS, Apple follows an open standard.

This is officially declared here:

<https://github.com/apple/swift-nio-transport-services>

“Network.framework is Apple’s reference implementation of the proposed post-sockets API that is currently being worked on by the Transport Services Working Group (taps) of the IETF.”

5. Sources to corroborate the impact

(alphabetic order)

Anna Brunstrom, Karlstad University, anna.brunstrom@kau.se
Stuart Cheshire, Apple Inc., cheshire@apple.com
Jon Crowcroft, Cambridge University, Jon.Crowcroft@cl.cam.ac.uk
Reese Enghardt, Netflix, ietf@tenghardt.net
Gorry Fairhurst, University of Aberdeen, gorry@erg.abdn.ac.uk
Tommy Pauly, Apple Inc., tpauly@apple.com
Colin Perkins, University of Glasgow, csp@csperkins.org
Philipp S. Tiesel, SAP SE, philipp@tiesel.net
Brian Trammell, Google Switzerland GmbH, ietf@trammell.ch
Éric Vyncke, Cisco Systems, evyncke@cisco.com