Transport Area
Internet-Draft
Intended status: Informational
Expires: June 7, 2014

T. Moncaster, Ed.
J. Crowcroft
University of Cambridge
M. Welzl
University of Oslo
D. Ros
Telecom Bretagne
M. Tuexen
Muenster Univ. of Appl. Sciences

December 4, 2013

Problem Statement: Why the IETF Needs Defined Transport Services
draft-moncaster-tsvwg-transport-services-01

Abstract

The IETF has defined a wide range of transport protocols over the past three decades. However, the majority of these have failed to find traction within the Internet. This has left developers with little choice but to use TCP and UDP for most applications. In many cases the developer isn’t interested in which transport protocol they should use. Rather they are interested in the set of services that the protocol provides to their application. TCP provides a very rich set of transport services, but offers no flexibility over which services can be used. By contrast, UDP provides a minimal set of services.

As a consequence many developers have begun to write application-level transport protocols that operate on top of UDP and offer them some of the flexibility they are looking for. We believe that this highlights a real problem: applications would like to be able to specify the services they receive from the transport protocol, but currently transport protocols are not defined in this fashion. There is an additional problem relating to how to ensure new protocols are able to be adopted within the Internet, but that is beyond the scope of this problem statement.

Status of This Memo

Table of Contents

1. Introduction

The IETF has defined a wide array of transport protocols including UDP [RFC0768], TCP [RFC0793], SCTP [RFC4960], UDP-Lite [RFC3828], DCCP [RFC4340] and MPTCP [RFC6824]. In most cases new protocols have been defined because the IETF has established that there is a need for a set of behaviours than cannot be offered by any existing transport protocol.

However, for an application programmer, using protocols other than TCP or UDP can be hard: not all protocols are available everywhere, hence a fall-back solution to TCP or UDP must be implemented. Some protocols provide the same services in different ways. Layering decisions must be made (e.g. should a protocol be used natively or over UDP?). Because of these complications, programmers often resort to either using TCP (even if there is a mismatch between the services provided by TCP and the services needed by the application) or implementing their own customised solution over UDP, and the opportunity of benefiting from other transport protocols is lost. Since all these protocols were developed to provide services that solve particular problems, the inability of applications to make use of them is in itself a problem. Implementing a new solution e.g. over UDP also means re-inventing the wheel (or, rather, re-implementing the code) for a number of general network functions such as methods to interoperate through NATs and PMTUD.

We believe this mismatch between the application layer and transport layer can be addressed in a simple fashion. If an API allowed applications to request transport services without specifying the protocol, the transport system underneath could automatically try to make the best of its available resources. It could use available transport protocols in a way that is most beneficial for applications and without the application needing to worry about problems with middlebox traversal. Adopting this approach could give more freedom for diversification to designers of Operating Systems.

1.1. Changes in This Version (to be removed by RFC Editor)

From draft-moncaster-tsvwg-transport-services-00 to -01: Editorial corrections and clarifications including:

* Updated Section 2.1 to highlight that we will take a hybrid approach to identifying Transport Services, both top down (by examining existing APIs) and bottom up (by looking at existing transport protocols).

* Updated Section 2.2 to commit to delivering at least one example API for this work.

* Replaced Section 4. The new version makes it clear that we will preserve the status quo where the transport may or may not choose to implement security.

2. Transport Services

The transport layer provides many services both to the end application (e.g. multiplexing, flow control, ordering, reliability) and to the network (e.g. congestion control). For the purposes of this document we define Transport Services as follows:

- A Transport Service is any service provided by the transport layer that can only be correctly implemented with information from the application.

The key word here is "information" -- many existing transport protocols function perfectly adequately because the choice of protocol implicitly includes information about the desired transport capabilities. For instance the choice of TCP implies a desire for reliable, in-order data delivery. However we think that such implicit information is not always sufficient. The rest of this section explains how we propose to identify Transport Services and how those services might then be exposed to the application.

2.1. Identifying Transport Services

One of the key aspects of this work is how to identify which Transport Services should actually be supported. We are taking a two-pronged approach. Rather than trying to identify every possible service that popular applications might need, we will survey a given set of common APIs that applications use to communicate across the network. We will complement this with a bottom-up approach where we establish the set of services that have already been published in RFCs coming from the Transport Area. This way, much of the discussion about the need to specify these services has already taken place, and it is unnecessary to re-visit those discussions. It is our hope that this approach will lead to identifying a set of service primitives that can be combined to offer a rich set of services to the application.

2.2. Exposing Transport Services

These Transport Services would be exposed to the application via an API. The definition of such an API and the functionality underneath the API are beyond the scope of this problem statement. We briefly describe three possible approaches below.

One approach could be to develop a transport system that fully operates inside the Operating System. This transport system would provide all the defined services for which it can use TCP as a fall-back at the expense of efficiency (e.g., TCP's reliable in-order delivery is a special case of reliable unordered delivery, but it may be less efficient). To test whether a particular transport is available it could take the Happy Eyeballs [I-D.wing-tsvwg-happy-eyeballs-sctp] approach proposed for SCTP -- if the SCTP response arrives too late then the connection just uses TCP and the SCTP association information could be cached so that a future connection request to the same destination IP address can automatically use it.

Polyversal TCP [PVTCP] offers another possible approach. This starts by opening a TCP connection and then attempts to establish other paths using different transports. The TCP connection ensures there's always a stable fallback. Having established the initial connection, PVTCP can then use service requests coming through setsockopt() to select the most appropriate transport from the available set.

Another approach could be to always rely on UDP only, and develop a whole new transport protocol above UDP which provides all the services, using a single UDP port. Instead of falling back to TCP, this transport system could return an error in case there is no other instance of the transport system available on the other side; the first packets could be used to signal which service is being requested to the other side (e.g., unordered delivery requires the receiving end to be aware of it).

3. Why Now?

So why do we need to deal with this issue now? There are several answers. Firstly, after several decades of dominance by various flavours of TCP and UDP (plus limited deployment of SCTP [RFC4960]), transport protocols are undergoing significant changes. Recent standards allow for parallel usage of multiple paths (MPTCP [RFC6824] and CMT-SCTP [I-D.tuexen-tsvwg-sctp-multipath]) while other standards allow for scavenger-type traffic (LEDBAT [RFC6817]). What sets these apart from e.g. DCCP [RFC4340] is that they have already seen deployment in the wild -- one of the Internet's most popular applications, BitTorrent, uses LEDBAT and MPTCP is already seeing deployment in major operating systems [Bonaventure-Blog]. Meanwhile there is a trend towards tunnelling transports inside UDP -- SCTP over DTLS over UDP is now being shipped with a popular browser in order to support WebRTC [RFC6951][I-D.ietf-tsvwg-sctp-dtls-encaps] while RTMFP [I-D.thornburgh-adobe-rtmfp] and QUIC [QUIC] are recent examples of transport protocols that are implemented over UDP in user space. In a similar vane, Minion [I-D.iyengar-minion-protocol] is a proposal to realise some SCTP-like services with a downwards-compatible extension to TCP.

All of a sudden, application developers are faced with a heterogeneous, complex set of protocols to choose from. Every protocol has its pro's and con's, but often the reasons for making a particular choice depend not on the application's preferences but on the environment (e.g., the choice of Minion vs. SCTP would depend on whether SCTP could successfully be used on a given network path).

Choosing a protocol that isn't guaranteed to work requires implementing a fall-back method to e.g. TCP, and making the best possible choice at all times may require sophisticated network measurement techniques. The process could be improved by using a cache to learn which protocols previously worked on a path, but this wouldn't always work in a cloud environment where virtual machines can and do migrate between physical nodes.

We therefore argue that it is necessary to provide mechanisms that automate the choice and usage of the transport protocol underneath the API that is exposed to applications. As a first step towards such automation, we need to define the services that the transport layer should expose to an application (as opposed to today's typical choice of TCP and UDP).

4. Security Considerations

Whether or not to enable TLS[RFC5246] is currently left up to individual protocol implementations to decide. While there is some debate about whether this is correct we have chosen to keep the status quo.

5. IANA Considerations

This document makes no request to IANA although in future an IANA register of Transport Services may be required.

6. Conclusions

After decades of relative stagnation the last few years have seen many new transport protocols being developed and adopted in the wild. This evolution has been driven by the changing needs of application developers and has been enabled by moving transport services into the application or by tunnelling over an underlying UDP connection.

Application developers are now faced with a genuine choice of different protocols with no clear mechanism for choosing between them. At the same time, the still-limited deployment of some protocols means that the developer must always provide a fall-back to an alternative transport if they want to guarantee the connection will work. This is not a sustainable state of affairs and we believe that in future a new transport API will be needed that provides the mechanisms to facilitate the choice of transport protocol. The first step towards this is to identify the set of Transport Services that a transport protocol is able to expose to the application. We propose doing this in a bottom-up fashion, starting from the list of services available in transport protocols that are specified in RFCs.

7. Contributors and Acknowledgements

Many thanks to the many people that have contributed to this effort so far including Arjuna Sathiaseelan, Jon Crowcroft, Marwan Fayed and Bernd Reuther among many others.

D. Ros and M. Welzl were part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). T. Moncaster and J. Crowcroft are part-funded by the European Union's Seventh Framework Programme FP7/2007-2013 under the Trilogy 2 project, grant agreement no. 317756.

8. Comments Solicited

To be removed by RFC Editor: This draft is the first step towards an IETF BoF on Transport Services. Comments and questions are encouraged and very welcome. They can be addressed to the current mailing list <transport-services@ifi.uio.no> and/or to the authors. We also have a website at <https://sites.google.com/site/transportprotocolservices/>

9. References

9.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, July 2004.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, December 2012.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, January 2013.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013.

9.2. Informative References

[Bonaventure-Blog]
Bonaventure, O., "Blog Entry: MPTCP used in iOS 7", September 2013.

[I-D.dreibholz-tsvwg-sctpsocket-multipath]
Dreibholz, T., Becke, M., and H. Adhari, "SCTP Socket API Extensions for Concurrent Multipath Transfer", draft-dreibholz-tsvwg-sctpsocket-multipath-06 (work in progress), July 2013.

[I-D.ietf-tsvwg-sctp-dtls-encaps]
Tuexen, M., Stewart, R., Jesup, R., and S. Loreto, "DTLS Encapsulation of SCTP Packets", draft-ietf-tsvwg-sctp-dtls-encaps-02 (work in progress), October 2013.

[I-D.iyengar-minion-protocol]
Jana, J., Cheshire, S., and J. Graessley, "Minion - Wire Protocol", draft-iyengar-minion-protocol-02 (work in progress), October 2013.

[I-D.thornburgh-adobe-rtmfp]
Thornburgh, M., "Adobe’s Secure Real-Time Media Flow Protocol", draft-thornburgh-adobe-rtmfp-10 (work in progress), July 2013.

[I-D.tuexen-tsvwg-sctp-multipath]
Amer, P., Becke, M., Dreibholz, T., Ekiz, N., Jana, J., Natarajan, P., Stewart, R., and M. Tuexen, "Load Sharing for the Stream Control Transmission Protocol (SCTP)", draft-tuexen-tsvwg-sctp-multipath-07 (work in progress), October 2013.

[I-D.wing-tsvwg-happy-eyeballs-sctp]

Wing, D. and P. Natarajan, "Happy Eyeballs: Trending Towards Success with SCTP", draft-wing-tsvwg-happy-eyeballs-sctp-02 (work in progress), October 2010.

[PVTCP] Nabi, Z., Moncaster, T., Madhavapeddy, A., Hand, S., and J. Crowcroft, "Evolving TCP: how hard can it be?", Proceedings of ACM CoNEXT 2012, December 2012.

[QUIC] Roskind, J., "Quick UDP Internet Connections", June 2013.

Authors' Addresses

Toby Moncaster (editor)
University of Cambridge
Computer Laboratory
J.J. Thomson Avenue
Cambridge CB3 0FD
UK

Phone: +44 1223 763654
EMail: toby.moncaster@cl.cam.ac.uk

Jon Crowcroft
University of Cambridge
Computer Laboratory
J.J. Thomson Avenue
Cambridge CB3 0FD
UK

Phone: +44 1223 763633
EMail: jon.crowcroft@cl.cam.ac.uk

Michael Welzl
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Phone: +47 22 85 24 20
EMail: michawe@ifi.uio.no

David Ros
Telecom Bretagne
Rue de la Chataigneraie, CS 17607
35576 Cesson Sevigne cedex
France

Phone: +33 2 99 12 70 46
EMail: david.ros@telecom-bretagne.eu

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
Steinfurt 48565
DE

EMail: tuexen@fh-muenster.de