

Transport Protocol Issues of In-Network Computing Systems
draft-kunze-coinrg-transport-issues-01

Abstract

Today's transport protocols offer a variety of functionality based on the notion that the network is to be treated as an unreliable communication medium. Some, like TCP, establish a reliable connection on top of the unreliable network while others, like UDP, simply transmit datagrams without a connection and without guarantees into the network. These fundamental differences in functionality have a significant impact on how COIN approaches can be designed and implemented. Furthermore, traditional transport protocols are not designed for the multi-party communication principles that underlie many COIN approaches. This document discusses selected characteristics of transport protocols which have to be adapted to support COIN functionality.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
2.	Addressing	3
3.	Flow granularity	3
4.	Authentication	4
5.	Security	5
6.	Advanced Transport Features	5
6.1.	Reliability	5
6.2.	Flow/Congestion Control	7
7.	Security Considerations	9
8.	IANA Considerations	9
9.	Conclusion	9
10.	Informative References	9
	Authors' Addresses	10

[1.](#) Introduction

A fundamental design choice of the Internet is that the network should be kept as simple as possible while complexity in the form of processing should be located on end-hosts at the edges of the network. This choice is reflected in the end-to-end principle which states that end-hosts directly address each other and perform all relevant computations while the network only delivers the packets without modifying them. Transport protocols are consequently designed to facilitate the direct communication between end-hosts.

In practice, the end-to-end principle is often violated by intransparent middleboxes which alter transmitted packets, e.g., by dropping or changing header fields. Contrary to that, computing in the network (COIN) encourages explicit computations in the network which introduces an intertwined complexity as the computations on the end-hosts depend on the functionality deployed in the network. It further challenges traditional end-to-end transport protocols as they are not designed to address in-network computation entities or to include more than two devices into a communication. Some of these problems are already presented in [I-D.[draft-kutscher-coinrg-dir-01](#)].

This draft intends to discuss potential problems for traditional transport protocols in more detail to raise questions that the COIN

community needs to solve before a widespread application of COIN functionality is sensible. Collaboration with other IRTF and IETF groups, such as PANRG, the IETF transport area in general, or the LOOPS BOF, can help in finding suitable solutions.

2. Addressing

The traditional addressing concept of the Internet is that end-hosts directly address each other with all computational intelligence residing at the network edges. With COIN, computations move into the network and need to be integrated into the established infrastructure. In systems where the whole network is under the control of the network operator this integration can be implemented by explicitly adjusting the communication schemes based on the COIN functionality. Considering larger scales, e.g., Internet-wide deployment, this approach of manually adjusting traffic patterns and applications to correctly incorporate changes made by the network is no longer feasible.

What is needed are ways to specify which kind of functionality should be applied to the transmitted data on the path inside the network and maybe even where or by whom the execution should take place. Such functionality could for example be implemented using an indirection mechanism which routes a packet along a pre-defined or dynamically chosen path which then realizes the desired functionality. Related concepts are Segment and Source Routing as well as (Service/Network) Function Chaining/Composition.

Main challenges/questions are:

1. How should end-hosts address the COIN functionality?
2. How can the end-hosts influence where or by whom the functionality is executed?
3. How can devices which do not implement COIN functionality be integrated into the systems without breaking the COIN or legacy functionality?

Another question is how the transmitted data is to be treated by the devices implementing the functionality.

3. Flow granularity

Core networking hardware pipelines such as backbone switches are built to process incoming packets on a per-packet basis, keeping little to no state between them. This is appropriate for the general task of forwarding packets, but might not be sufficient for COIN as

information that is needed for the computations can be spread across several packets. In a TCP stream, for example, data is dynamically distributed across different segments which means that the data needed for application-level computations might also be split up. In contrast to that the content of UDP datagrams is defined by the application itself which is why the datagrams could either be self-contained or information can be cleverly distributed onto different datagrams. The common scheme is that different transport protocols induce different meanings to the packets that they send out which needs to be accounted for in COIN elements as they have to know how the received data is to be interpreted. There are at least three options for this.

1. Every packet is treated individually. This respects the possibilities that are already offered by all networking equipment.
2. Every packet is treated as part of a message. In this setting, the packet alone does not have enough information for the computations. Instead, it is important to know the content of the surrounding packets which together form the overall message.
3. Every packet is treated as part of a byte stream. Here, all previous packets and potentially even all subsequent packets need to be taken into consideration for the computations as the current packet could, e.g., be the first of a group of packets, a packet in the middle or the final packet.

The flow granularity consequently has a significant impact on how computations can be performed and where. Apart from how the COIN elements should treat the transmitted data, another important aspect is how it can be ensured that the end-hosts know who has altered the data and how.

4. Authentication

The realisation of COIN legitimizes and actively promotes that data transmitted from one host to another can be altered on the way inside the network. This opens the door for foul play as all intermediate network elements - no matter if they are malicious or misbehaving by accident, COIN elements, or 'traditional' middleboxes - could simply start altering parts of the original data and potentially cause harm to the end-hosts. What is needed are mechanisms with which the receiving host can verify (a) how and (b) by whom the data has been altered on the way. In fact, these might very well be two distinct mechanisms as one (a) only focusses on the changes that are made to the data while (b) requires a scheme with which COIN elements can be uniquely identified (could very well relate to [Section 2](#)) and

subsequently authenticated. The Proof of Transit [I-D.[draft-ietf-sfc-proof-of-transit-04](#)] concept of the SFC WG could be applicable for proving that a packet has indeed passed the desired COIN elements, although it does not provide means to validate which changes were made by the known nodes.

Main challenges/questions are:

1. How are changes to the data within the network communicated to the end-hosts?
2. How are the COIN elements that are responsible for the changes communicated to the end-hosts?
3. How are changes made by the COIN elements authenticated?

5. Security

Many early COIN concepts require an unencrypted transmission of data. At the same time, there is a general tendency towards more and more security features in communication protocols. QUIC, e.g., encrypts all payload data and almost all header content already inside the transport layer. This makes current COIN concepts infeasible in settings where QUIC connections are used as the COIN elements do not have access to any packet content. Using COIN thus also depends on how well security mechanisms like encryption can be integrated into COIN frameworks.

Together, the four aspects presented in [Section 2](#) to [Section 5](#) form a set of fundamental properties for a basic transport-compatible realization of COIN. In the following, we briefly discuss selected additional transport features to create awareness for the multifaceted interaction between the transport protocols and COIN elements.

6. Advanced Transport Features

Depending on application needs, different transport protocols provide different features. These features shape the behavior of the protocol and have to be taken into account when developing COIN functionality. In this section, we focus on the impact of reliability as well as flow and congestion control.

6.1. Reliability

Applications require a reliable transport whenever it is important that all data has been transmitted successfully. TCP[RFC0793] provides such a reliable communication as it first sets up a

dedicated connection and then ensures the successful reception of all data. In contrast, UDP[RFC0768] is a connectionless protocol without guarantees and COIN elements working on UDP transmissions must be robust to lost information. This is not the case for applications on top of TCP, but the retransmissions and the TCP state, which TCP uses to achieve the reliability, make packet processing for COIN more complex due to at least three reasons.

The concept of retransmissions bases on the end-to-end principle as retransmissions are performed by the sender if it has determined that the receiver did not receive the corresponding original message. Both participants can then act knowing that parts of the overall data are still missing. For simple COIN elements, which are not aware of the involved TCP states and which do not track sequence numbers, it is difficult to identify (a) that a packet in the sequence is missing and (b) that a packet is a retransmission. One question is whether COIN elements should incorporate an understanding for retransmissions on the basis of existing transport mechanisms or if a COIN-capable transport should include dedicated signals for the COIN elements.

Apart from challenges in identifying retransmissions, there is also the fact that they are sent out of order with the original packet sequence. Depending on the chosen flow granularity (see [Section 3](#)), COIN elements might have to hold contextual information for a prolonged time once they identify an impending retransmission. Moreover, they might have to postpone or cancel computations if data is missing and instead schedule later computations. The main question arising from this is: to what extent should COIN elements be capable of incorporating retransmissions into their computation schemes and how much additional storage capabilities are required for this?

When incorporating COIN elements into the retransmission mechanisms, it is also an interesting question whether it should be possible to request or perform retransmissions from COIN elements. Considering a setting with COIN elements that are capable of detecting missing packets and retransmission requests, it might improve the overall performance if the COIN element directly requests or performs the retransmission instead of forwarding the packet/request through the complete sequence of elements. In both cases, the aforementioned storage capabilities are relevant so that the COIN elements can store enough information. The general question, i.e., which nodes in the sequence should do the retransmission, has already been worked on in the context of multicast transport protocols.

Depending on the extent of realization of the presented retransmission features, COIN elements might almost have to implement some of TCP's state to fulfil their tasks. Considering that

different COIN elements have different computational and storage capacities, it is very likely that not every form of transport integration into COIN can be supported by every available COIN platform. The choice of devices included into the communication will hence certainly affect the types of transport protocols that can be operated on the COIN networks.

Challenges/questions regarding reliability are:

1. Should COIN elements be aware of retransmissions?
2. How can COIN elements identify missing packets or retransmissions?
3. Should COIN elements be explicitly notified about retransmissions?
4. To what extent should COIN elements be capable of incorporating retransmissions into their computation schemes?
5. How much storage capabilities are required for incorporating retransmissions?
6. How can COIN elements incorporate missing packets into their computations?
7. How to deal with state changes in COIN elements caused by data lost later in the communication chain and then retransmitted?
8. Should COIN elements be capable of requesting retransmissions/ answering retransmission requests?
9. Which devices should perform retransmissions?
10. Do COIN elements have to keep transport state?
11. How much transport state do COIN elements have to keep?

Another aspect is flow and congestion control to avoid overloading the receiving end-host and the network.

6.2. Flow/Congestion Control

TCP incorporates mechanisms to avoid overloading the receiving host (flow control) and the network (congestion control) and determines its sending rate as the minimum value of what both mechanisms determine as feasible for the system. This approach is based on the notion that computing and forwarding hosts are separated and is

challenged by the inclusion of COIN elements, i.e., computing nodes in the network.

Flow control bases on explicit end-host information as the participating end-hosts notify each other about how much data they are capable of processing and consequently do not transmit more data as the other host can handle. This only changes if one of the end-hosts updates its flow control information.

Congestion control, on the other hand, interprets volatile feedback from the network to guess a sending rate that is possible given the current network conditions. Most congestion control algorithms hereby follow a cyclical procedure where the sending end-hosts constantly increase their sending rate until they detect network congestion. They then decrease their sending rate once and start to increase it again.

In this traditional two-fold approach, loss, delay, or any other congestion signal (depending on the congestion control algorithm) induced by COIN elements (only in case that they are the bottleneck) is interpreted as network congestion and thus accounted for in the congestion control mechanism. This means that the sending end-host may repeatedly overload the computational capabilities of the COIN elements when probing for the current network conditions instead of respecting general device capabilities as is done by flow control.

Consequently, the question arises whether COIN elements should be able to participate in end-to-end flow control.

Main challenges/questions are:

1. Should COIN elements be covered by congestion control?
2. Should COIN elements be able to participate in end-to-end flow control?
3. How could a resource constraint scheme similar to flow control be realized for COIN elements?

All in all, there is a wide range of non-essential transport features which offer improved performance in certain settings and for certain application combinations. However, as presented, it is likely that not all of the features/types of transport protocols can be supported on every COIN element. It might make sense to define different classes of COIN-ready transport protocols which can then be deployed depending on the concretely available networking/hardware elements. Alternatively, each of the features could be treated separately and

they could then be composed based on the demands of an application at-hand.

Overall, the general question summarizing all of the other challenges/question is:

Which transport features should be supported by COIN, how can they be identified and how can they be provided to application designers?

7. Security Considerations

TBD

8. IANA Considerations

N/A

9. Conclusion

The advent of COIN comes with many new use cases and promises improved solutions for various problems. It is, however, not directly compatible with the end-to-end nature of transport protocols. To enable a transport-based communication, it is thus important to answer key questions regarding COIN and transport protocols, some of which are raised in this document.

10. Informative References

- [I-D.[draft-ietf-sfc-proof-of-transit-04](#)]
Brockners, F., Bhandari, S., Mizrahi, T., Dara, S., and S. Youell, "Proof of Transit", [draft-ietf-sfc-proof-of-transit-04](#) (work in progress), November 2019.
- [I-D.[draft-kutscher-coinrg-dir-01](#)]
Kutscher, D., Karkkainen, T., and J. Ott, "Directions for Computing in the Network", [draft-kutscher-coinrg-dir-01](#) (work in progress), November 2019.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, [RFC 768](#), DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, [RFC 793](#), DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.

Authors' Addresses

Ike Kunze
RWTH Aachen University
Ahornstr. 55
Aachen D-50274
Germany

Phone: +49-241-80-21422
Email: kunze@comsys.rwth-aachen.de

Klaus Wehrle
RWTH Aachen University
Ahornstr. 55
Aachen D-50274
Germany

Phone: +49-241-80-21401
Email: wehrle@comsys.rwth-aachen.de

