COINRG
Internet-Draft
Intended status: Informational
Expires: 6 May 2021

I. Kunze
K. Wehrle
RWTH Aachen University
D. Trossen
Huawei Technologies Duesseldorf GmbH

2 November 2020

Transport Protocol Issues of In-Network Computing Systems
draft-kunze-coinrg-transport-issues-03

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 raises several questions regarding the use of transport protocols in connection with COIN.

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 6 May 2021.

Copyright Notice

Table of Contents

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-02].

This draft discusses 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. Terminology

COIN element: Device on which COIN functionality can be deployed

3. 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, this approach of manually adjusting traffic patterns and applications to correctly incorporate changes made by the network is not 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.

One possibility is to directly route on service or functionality identifiers instead of sending individual packets between locator-addressed network elements [I-D.draft-sarathchandra-coin-appcentres-03]. While this aligns the routing more clearly with the communication between computational elements, selecting the 'right' computational endpoint (out of possibly several ones) becomes critical to the proper functioning of the overall service.

Further 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 treatment of the transmitted data, i.e., which COIN functionality to execute, be represented in the addressing of the request?

3. How can end-hosts direct computational requests to different computational endpoints of the same service in different network locations, i.e., decide where the COIN functionality is executed?

4. How to decide which computational endpoint to choose (from possibly several ones existing in the network)?

5. How can devices which do not implement COIN functionality be integrated into the systems without breaking the COIN or legacy functionality?

4. 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. Summarizing, 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.

Along those options above, the question arises how shorter-term 'messages' (or transactions) of the computation should be handled compared to the often longer-term management of the network resources needed to transmit the packets across one or more such messages. For instance, error control may be best applied to the individual messages between computational endpoints, while congestion control may be applied across several messages at the level of the relation between the network elements hosting the computational endpoints. In this view, the notion of a 'flow' may separate message or transaction handling from the resource management aspect, where a flow may be divided into sub-flows (said messages or transactions) with error control being applied to those sub-flows but resource management being applied to the overall flow. Such choice of flow granularity would consequently have a significant impact on how and where computations can be performed as well as ensuring that end-hosts know who has altered the data and how.

5. Collective Communication

COIN scenarios may exhibit a collective communication semantic, i.e., a communication between one and more computational endpoints, as is for example illustrated by use cases in [I-D.draft-sarathchandra-coin-appcentres-03]. With this, unicast and multicast transmissions become almost equal forms of communication, as also observed in work on information-centric networking (ICN).

Yet, these many-point relations may be ephemeral down to the granularity of individual service requests between computational endpoints which questions the viability of stateful routing and transport approaches used for long-lived multicast scenarios such as liveTV transmissions.

This is particularly pertinent for the transport layer where reliability and flow control among a quickly changing set of receivers is a challenging problem. The ability to divide receiver groups with the support of in-network COIN elements may provide solutions that will cater to the possible dynamics of collective communication among computational endpoints.

6. 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 3) and subsequently authenticated. The Proof of Transit [I-D.draft-ietf-sfc-proof-of-transit-08] 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?

7. 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.

8. 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 to create awareness for the multifaceted interaction between the transport protocols and COIN elements.

8.1. Reliability

Applications require a reliable transport whenever it is important that all data is 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 4), COIN elements might have to hold contextual information for a prolonged time once they identify an impeding 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. This is especially interesting in the context of collective communication where reliability mechanisms could make use of the multi-source nature of the communication and leverage the presence of many COIN elements in the network, for

instance by using network coding techniques, which in turn may benefit from COIN elements participating in the reliability mechanism. In all 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.

Another aspect to consider is the 'unit' that needs to be reliably transferred. In stream-based transport protocols, such as TCP, packets represent the smallest unit of transfer. However, different choices in the flow granularity and a possible move to larger-than-a-packet messages or transactions, as suggested in Section 4, might make other approaches to reliability viable that operate on the basis of such messages.

Challenges/questions regarding reliability are:

1. What is the unit of reliable transfer?

2. How to utilize more than one computational endpoint in the reliability mechanism?

3. Should COIN elements be aware of retransmissions?

4. How can COIN elements identify missing packets or retransmissions?

5. Should COIN elements be explicitly notified about retransmissions?

6. To what extent should COIN elements be capable of incorporating retransmissions into their computation schemes?

7. How much storage capabilities are required for incorporating retransmissions?

8. How can COIN elements incorporate missing packets into their computations?

9. How to deal with state changes in COIN elements caused by data lost later in the communication chain and then retransmitted?

10. Should COIN elements be capable of requesting retransmissions/ answering retransmission requests?

11. Which devices should perform retransmissions?

12. Do COIN elements have to keep transport state?

13. How much transport state do COIN elements have to keep?

8.2. 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.

In the context of COIN, the granularity of flows may see a division into sub-flows or messages to better represent the used computational semantic as discussed in Section 4. This raises the question whether flow and congestion control should be applied to longer term flows (of many sub-flows or messages) or directly to sub-flows. Eventually, this could possibly lead to a separation of error control (for sub-flows) and flow control (for longer-term flows). A subsequent challenge is then how to reconcile the possible volatile nature of sub-flow relations (between computational endpoints) with the longer-term relationship between network endpoints that will see a flow of messages between them. This is particularly pertinent in collective communication scenarios, where many forward unicast sub-flows may lead to a single multicast sub-flow response albeit only for that one response message. Reconciling the various unicast resource regimes into a single (ephemeral) multicast one poses a significant challenge.

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?

4. How to reconcile message-level flexibility in transport relations between computational endpoints with longer-term resource stability between network elements participating in the computational scenario?

9. Security Considerations

COIN changes the traditional paradigm of a simple network and the corresponding end-to-end principle as it encourages computations in/by the network. Approaches designed to protect transmitted data, such as Transport Layer Security (TLS) which is even embedded into newer transport protocols like QUIC, rely on the end-to-end principle and are thus conceptually not compatible with COIN. Additionally, COIN elements often do not support required cryptographic functionality. Thus, there exist no out-of-the-box security solutions for COIN which means new security concepts have to be developed to allow for a secure use of COIN.

10. IANA Considerations

N/A

11. 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.

All in all, there is a wide range of transport features which offer improved performance in certain settings and for certain application combinations. However, as presented, it is unlikely that 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.

The general question summarizing this document is:

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

12. Informative References

[I-D.draft-ietf-sfc-proof-of-transit-08]
Brockners, F., Bhandari, S., Mizrahi, T., Dara, S., and S. Youell, "Proof of Transit", Work in Progress, Internet-Draft, draft-ietf-sfc-proof-of-transit-08, 1 November 2020, <http://www.ietf.org/internet-drafts/draft-ietf-sfc-proof-of-transit-08.txt>.

[I-D.draft-kutscher-coinrg-dir-02]
Kutscher, D., Karkkainen, T., and J. Ott, "Directions for Computing in the Network", Work in Progress, Internet-Draft, draft-kutscher-coinrg-dir-02, 31 July 2020, <http://www.ietf.org/internet-drafts/draft-kutscher-coinrg-dir-02.txt>.

[I-D.draft-sarathchandra-coin-appcentres-03]
Trossen, D., Sarathchandra, C., and M. Boniface, "In-Network Computing for App-Centric Micro-Services", Work in Progress, Internet-Draft, draft-sarathchandra-coin-

appcentres-03, 23 October 2020, <http://www.ietf.org/internet-drafts/draft-sarathchandra-coin-appcentres-03.txt>.

[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
D-52074 Aachen
Germany

Email: kunze@comsys.rwth-aachen.de

Klaus Wehrle
RWTH Aachen University
Ahornstr. 55
D-52074 Aachen
Germany

Email: wehrle@comsys.rwth-aachen.de

Dirk Trossen
Huawei Technologies Duesseldorf GmbH
Riesstr. 25C
D-80992 Munich
Germany

Email: Dirk.Trossen@Huawei.com