Evolving the End-to-End Transport Layer in Times of Emerging Computing In The Network (COIN)

Ike Kunze
RWTH Aachen University
Aachen, Germany
kunze@comsys.rwth-aachen.de

Dirk Trossen
Huawei Technologies Germany
Düsseldorf, Germany
dirk.trossen@huawei.com

Klaus Wehrle
RWTH Aachen University
Aachen, Germany
wehrle@comsys.rwth-aachen.de

***Abstract*—The possibility of richer computing capabilities within Internet network elements, often captured as Computing in the Network (COIN), promises performance and flexibility gains to the wider Internet, akin to those seen in recent data center advances. At the same time, moving computation into the network is seemingly at odds with the fundamental end-to-end principle underlying the development of key technologies in the Internet. In this paper, we do not only argue that the latter is not the case, but we also shed light on what 'in the network' may or may not entail, aiming to sharpen a possible research agenda for COIN. Taking the transport layer as an example due to its typical end-to-end realization in and importance for today's Internet, we outline key design considerations for evolving towards a COIN-enabled transport capability. By further creating linkages to existing efforts and concepts, we provide possible future directions for the design of protocols for the future Internet.**

I. Introduction

A fundamental consideration for the Internet's design is that functions can be implemented correctly and completely only with the knowledge of the applications, as formulated by Saltzer et al. [1]. The Internet community has adopted this consideration as the fundamental end-to-end (E2E) principle [2], [3], postulating that end-hosts perform most, if not all, relevant computations. In contrast, the network is merely seen as performing suitable operations for the delivery of packets from a source to a destination, the latter typically expressed within the semantics of the IP addressing scheme. This view on the Internet is captured in the notion of the network being a 'dumb pipe', with all intelligence beyond simple packet delivery pushed to its endpoints.

Emerging technologies, such as Software-Defined Networking (SDN) or P4, seem to run counter this fundamental Internet design principle, postulating programmable forwarding actions that utilize packet header information as input; this is often termed as *Computing in the Network (COIN)*. However, Saltzer et al. do consider that "sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement" [1], which again aligns programmable networking technologies as well as COIN with possibly the most fundamental Internet design principle.

Nonetheless, there is ample ambiguity in Saltzer et al.'s formulations [1], most notably regarding the extent of the 'incompleteness' as well as the 'function' itself that may be considered for (partial) realization in the network. It is here where the question on what defines COIN needs answering: in relation to both, the original E2E argument and the functions traditionally implemented on endpoints only.

For this, we discuss in Sec. II the notion of E2E function-*internal* versus -*external* computations in an attempt to better contrast the 'in the network' terminology against the endpoints standing at the ends of a communication relation. Thereby, we concretize the more general COIN notion, specifically through proposing two design principles for COIN in Sec. III. We then apply this thinking to transport protocol mechanisms as an example of functions that are (i) traditionally implemented at endpoints but (ii) may well be impacted by as well as explicitly utilize merging in-network capabilities. For this, we focus on the aspects of addressing, flow granularity, and collective communication and first delve into considerations for realizing transport layer functions in the presence of COIN throughout Sec. IV before linking those to ongoing efforts and concepts in Sec. V. With this, we not only aim to provide guidance for future COIN-related research, but also considerations for the general future Internet. In particular, we link to the development of future transport protocols in light of expected increased capabilities of its underlying (New) IP and transport network layers, as discussed as part of possible future directions of needed work in Sec. VI.

II. What is COIN, Anyway?

The Internet [4] has long grown into a world-spanning system from its original research network roots. In the early stages, roles in the Internet were largely divided between *hosts* and the *network*, the latter designed to deliver packets between the former. This is illustrated in Fig. 1 (top): the network serves as a 'dumb pipe' for the endpoints, which in turn realize all functionality for transport layer and above.

However, with time, many new communication paradigms have emerged, extending the established peer-to-peer (P2P) communication of the Internet. For example, large Internet companies now provide centralized computing resources in their data centers, often denoted as the '*cloud*'. Recent developments in *edge computing* aim at decentralizing this (cloud-based) functionality towards the '*edge*', optimizing latency and localizing traffic overall. Furthermore, enabled by advances in networking hardware as well as corresponding new programming paradigms (SDN) and languages (P4), it is now possible

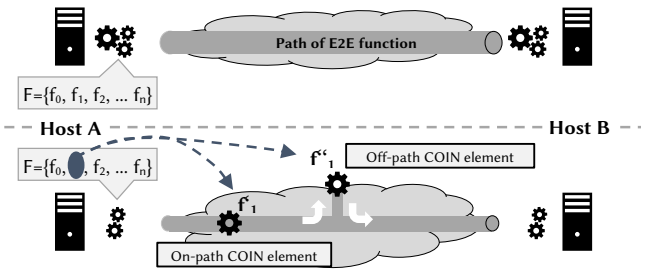

Fig. 1. Evolving from a dumb pipe (top) to a COIN vision with on- and off-path deployments (bottom).

to push previously host-centric computations into the network using programmable network devices (PNDs).

What may 'in-network' mean? All these paradigms are often linked to *Computing in the Network (COIN)* and corresponding discussions typically revolve around the 'place' of execution, captured by the 'in-network' aspect of COIN. Here, views range from restricting COIN to only considering computations on *networking hardware* to broader definitions interpreting COIN as a possible subset of edge or cloud computing. We generalize these considerations by broadly capturing any computation realized in *COIN elements*, i.e., devices providing COIN functionality, as shown in Fig. 1 (bottom). However, we do distinguish between COIN elements that are *on-path* in relation to the original E2E packet flow, e.g., PNDs, and *off-path*, e.g., in the context of typical edge, cloud, or P2P approaches. This location in terms of the original E2E packet flow can have important implications on the E2E function as we will discuss later. We next turn our focus from the 'place' to the 'computing' part in COIN.

Which computations matter to COIN? For an answer to this question, we hark back to the E2E argument [1], contrasting endpoints against the network facilitating communication between them. We argue that the notion of the *function* realized between the original endpoints (cf. Fig. 1, Hosts A and B) is essential for grasping what 'computing' in COIN may mean.

In order to conform to the E2E argument, any inserted computation must contribute to the original E2E function, thereby forming the incomplete version of the E2E function [1]. We term such computation as *E2E function-internal* (cf. Fig. 1 bottom), where the key to its insertion is that it enriches the original E2E function, while not violating its key requirements. In contrast, we consider computation that is inserted as an endpoint itself as being *E2E function-external*. We argue that it does not constitute COIN but merely traditional service chaining, e.g., using edge or cloud computing.

Takeaway. *Reflecting on COIN using the E2E argument identifies any* E2E function-internal *computation between endpoints as a possible form of COIN. These computations can then be divided into on- and off-path COIN functionality in terms of their place of execution 'in the network'.*

Based on our broad categorization for COIN, we next derive two design principles for E2E-compliant COIN functionality.

III. Design Principles for COIN

In the landscape of different computing paradigms with different compute locations, the next question for COIN is *how* and with that also *where* in the network function-internal computation should be inserted. Guidance can be found in the *simplicity principle* [5], which states that functionality should be kept as simple as possible, thereby limiting any needed additional complexity (in the network).

Let us illustrate this principle and its consequences for COIN by revisiting Fig. 1 (bottom). Consider a scenario where an E2E function-internal computation f_1 is to be deployed in a COIN environment. In our example, there are two possible deployment locations: on-path as $f_1^{'}$ or off-path as $f_1^{''}$. Each deployment location comes with a unique set of characteristics, e.g., regarding latency, compute complexity, or orchestration overhead. For illustration purposes, we choose 'latency' as the key requirement for our application.

The two deployment locations most likely have different impact on the additionally introduced latency. Specifically, the on-path COIN element does not induce any additional path latency, while forwarding packets to the off-path entity might unduly increase latency compared to just following through the normal path. On the other hand, the complexity of computation may be limited in PNDs compared to off-path COIN elements. However, as long as the computational capabilities suffice for the function-internal computations, on-path computation is clearly preferred from a latency perspective.

Similar considerations can be done for other optimization criteria. Consequently, deploying COIN functionality constitutes a multi-dimensional optimization problem. We argue that key to solving this optimization problem is that any additional functionality must not violate the original E2E function requirements and, additionally, strive for optimizing the key requirements, i.e., latency in the above example.

Based on these considerations and taking both, the E2E and the simplicity principle, into account, we can now formulate the following **first design principle for COIN**:

Any incomplete version of an E2E function must adhere to the original requirements of the E2E function, while enriching its functionality and optimizing functional complexity against its key communication requirement.

But simplicity is not the only guidance here. *Transparency* of any intended insertion of function-internal computation towards the functions at the endpoints is equally important. Computation that is inserted without such transparency may lead to problems in the overall operation of the E2E function, letting us formulate our **second design principle for COIN**:

Any incomplete version of an E2E function should be inserted in full transparency to the functions at the endpoints.

Let us take Performance Enhancing Proxies (PEPs) as an example to illustrate our discussion so far: The insertion of PEPs aims at enhancing the capabilities of TCP albeit in a manner that is not transparent to the E2E hosts, leading to well-recognized problems with PEPs in real deployments [6].

As another example, realizations of Network Address Translation (NAT) (in the network) do not impede the best effort

delivery of IP packets (expressed through latency as the primary requirement), while providing an enriched functionality in the form of address translation. While initial NAT solutions were implemented on-path, emerging carrier-grade NATs have been realized off-path, e.g., in cloud or edge sites, enabled by increasing capabilities for processing large numbers of flows without adding significant latency. However, the non-transparent insertion often leads to problems, such as non-reachability of hosts behind NATs. We can see from both examples how a positively intended function insertion may still lead to problems if the above principles are not followed.

Takeaway. *COIN functionality can be deployed in compliance with the E2E principle as long as the inserted functionality (i) is E2E function-internal, (ii) does not violate the original requirements, enriches the functionality and optimizes against the key communication requirements, and (iii) is inserted in full transparency to the endpoints.*

IV. Considerations for Transport

Let us now use the E2E principle compliant view on COIN to discuss considerations for novel transport layer solutions. We chose the transport layer as the most direct layer atop the packet delivery functions provided by the network itself, with transport functions being traditionally implemented in endpoints only. As such, we see the transport layer as not only being possibly impacted by emerging in-networking computing capabilities but also as a possible target for explicitly designing novel solutions with COIN capabilities in mind, positioning the transport layer as an ideal candidate for highlighting the challenges but also opportunities introduced through COIN. We do so by focussing on three key aspects presented in the following subsections.

A. Addressing

The core challenge for inserting COIN elements into a communication relation is that today's systems are designed for E2E communication. Consequently, end-hosts address each other while addresses of networking devices are only used for forwarding packets to the correct destination. Integration of the COIN functions into the E2E communication is required in order to create function-internal computation and adhere to the E2E argument.

***Implicit* integration** of *on-path* COIN functionality may be manageable in smaller or private deployments. For example, functionality can be placed at specific network locations that are expected to be passed by the E2E communication and end-hosts can be adapted according to the expected computation on the transmitted data. In large-scale or public deployments, however, where, e.g., only parts of the traffic might be intended to be subject to COIN or where the concrete paths of traffic are typically unknown in advance, this approach is not feasible. Additionally, an implicit approach would generally not enable the *off-path* notion of COIN as traffic is still steered along the standard path. Note that implicit integration still requires adaptations on the endpoints to satisfy our second design principle of full transparency.

***Explicit* steering** of traffic to the COIN elements, including specifying what *functionality* should be applied to the data, significantly broadens the possible scope of COIN. In particular, it extends the problem of addressing towards capturing the *communication semantics* in relation to the original E2E function that is being enriched, particularly in light of single COIN elements providing a richer set of possible functionality. Approaches could be similar to transport addressing in the form of IP address and port information. However, there are no means for specifying multiple IP/port pairs in current transport protocols and the use of ports may even be too limiting for app-specific functionality, possibly requiring richer semantics for 'services' being addressed.

Instance selection. It might further be the case that there are multiple instances of the same COIN service at different network locations. As a consequence, an endpoint may want to explicitly select one of those service instances, e.g., using constraints for the selection rather than directly specifying the network locator of the desired COIN elements. How to encode such constraints is crucial, especially with respect to the supported semantics for selection and when considering our first design principle.

Affinity. In addition, any relationship with COIN elements might need to ensure *affinity* to a particular member of the set of COIN elements that provide a desired functionality, e.g., due to ephemeral state being created through an ongoing transaction. For instance, orchestration functionality may be implemented using an indirection mechanism which routes a packet along a pre-defined or dynamically chosen path along which to realize the desired functionality. Traffic may be routed here based on service or functionality identifiers instead of sending individual packets between locator-addressed network elements [7], while selecting the 'right' computational endpoint (out of possibly several ones) becomes critical to the proper functioning of the overall service [8].

Efficient forwarding. Besides the decision of directing requests to COIN elements, the efficient execution of this decision is crucial in the form of packet forwarding operations. Programmable forwarding technologies, such as P4, may allow for rich decisions being implemented at the forwarding plane [9], while multi-optimality routing approaches [10] may choose offline routing operations to realize semantic-rich decision criteria for traffic steering. The tradeoff here may lie between forwarding complexity and possible dynamicity of frequently changing routing state.

Legacy integration. Lastly, we recognize that not all devices necessarily partake in COIN. To allow COIN utilize legacy systems, it is important to provide backwards compatibility without compromising COIN nor legacy functionality.

B. Flow Granularity

Today's networking hardware is built to process incoming packets on a per-packet basis, keeping little to no state between them. While appropriate for packet forwarding, this may pose challenges for transport layer semantics spanning

many packets, captured as a *flow*, often with interrelated state between packets.

For instance, TCP dynamically distributes data across different *segments* within a data stream, so that data needed for application-level computations might be split up across multiple segments by the transport protocol. In contrast, semantics for UDP datagram payload are defined by the application itself, i.e., the datagrams can be self-contained or information can be distributed across different datagrams. Yet other notions of flow may even consider relations between flows across different sender-receiver relations. Transport protocols are often designed with specific 'flow' semantics in mind, driven by application needs and requirements, while it is crucial to understand how COIN capabilities could support all or at least some of them.

Different levels of flow granularity. We recognize (at least) three levels of flow granularity: (i) Every packet is treated *individually*, mapping to the capabilities of existing networking equipment. (ii) Every packet is treated as *part of a message*. The packet alone does not have enough information for computation and it is important to know the content of surrounding packets forming the overall message. (iii) Every packet is treated as *part of a byte stream*. All previous packets and, potentially, even all subsequent packets need to be taken into consideration for the computations.

Granularity selection. The diversity in available COIN elements raises the problem of finding an appropriate flow granularity, possibly requiring a runtime selection of the 'right' flow granularity in one situation over another. The best analogy here is that of maximum transfer unit (MTU) size discovery with its limited computational operation of packet fragmentation to understand the possible complexity of determining the right granularity for COIN elements. Given that COIN elements are possibly less in numbers than intermediary routers, other approaches than on-path discovery should be investigated, such as those combining the granularity selection with orchestrating the computational function at the COIN elements. Furthermore, different granularities might even impact the previously discussed instance selection procedures and affinity concepts.

Impact on resource management. Another key aspect to consider is how different flow granularities might affect the short- and long-term management of (network) resources. For instance, error control may be best applied to the smallest available flow units, e.g., on the scale of individual packets, while congestion control may be applied to the relation between the network elements hosting the computational endpoints, i.e., across the overall flow. Considering that COIN elements might offer additional functionality, solely accounting for error and congestion control might be insufficient as, e.g., the forwarding capacities might still be sufficently available while only advanced compute capacities become a bottleneck. Extending the notion of resource management, similar to the aforementioned extension of MTU path discovery, might thus become necessary for richer COIN functionality. In this view, the notion of a "flow" may provide guidance for where to separate "message" handling from overall resource management.

Affinity. As discussed in Sec. IV-A, affinity must be ensured for an application-level transaction, executed at one of possibly many possible endpoint instances. Here, the notion of a flow may be utilized to make the necessary routing decision for packets belonging to the same flow, while a new flow may be routed to another network location providing the same (service) functionality as the previously chosen one. Flow information could potentially be used to delineate and, thus, signal such affinity of one packet to a previous one but the question on how to encode such flow information across different types of transport protocols is critical to ensure that such affinity treatment is not limited to specific applications. While such affinity questions are also relevant in other contexts, the unique challenge introduced by COIN is that affinity might now be required for the entire path instead of ensuring affinity to endpoints only.

C. Collective Communication

In extending basic unicast and multicast semantics, *collective communication* refers to messages being exchanged between one and more computational endpoints, e.g., illustrated in [7], where unicast and multicast transmissions become almost equal forms of communication, as is also observed in work on Information-Centric Networking (ICN) [11].

As a particular characteristic, these many-point relations may be *ephemeral* down to the granularity of individual requests between endpoints, which questions the viability of stateful routing and transport approaches used for multicast scenarios such as live TV transmissions, where receiver groups are often long-lived and rather stable.

Dynamic receiver sets. Instead, collective communication may see receiver set changes at every request, posing challenges to congestion but also error control. COIN elements may allow for separating those parts with more stable relations, e.g., in the core of the network, from those with more fluctuating receiver sets, e.g., at the network edge, applying different mechanisms for both error and congestion control in the separate parts. This is not a new approach per se, as PEPs and their usage, e.g., in satellite networking, show. However, the realization in COIN elements may make their deployment more suitable, where performance variances stem from the nature of the communication, not from the network. Furthermore, the ability to divide receiver groups with the support of COIN elements may also support solutions that use *random linear network coding* [12] as an ACK-free mechanism for error control.

V. Related Concepts and Efforts

The fundamental challenges underlying the areas described in Sec. IV have not just emerged with COIN, but are a constant focus of work in different domains. Consequently, there already exist several concepts and efforts that address the described challenges and might be applicable solutions, even for COIN. In this section, we discuss selected concepts for each area in more detail.

A. Addressing

As defined through the IP-based addressing, hosts specify the intended destination of outgoing packets, while the network takes care of delivering them accordingly.

Existing Solutions. *Source Routing* aims at providing more control to end-hosts over the path that packets should take by allowing senders to (partially) define the route through the network. This mechanism can, e.g., be leveraged to steer traffic along a chosen COIN-enabled path and, thus, trigger any desired COIN functionality. Segment Routing [13] is a modern, IETF-standardized variant of Source Routing for IPv6 and MPLS. While Source Routing provides full transparency to endpoints (cf. Sec. III), it merely defines through *which devices* a packet should go, not *which functionality* to execute.

Service Function Chaining (SFC) describes a process to provide the latter aspect, i.e., steer traffic through a predefined list of service functions, e.g., firewalls [14], as defined through its SFC architecture [15] and the next service header (NSH) [16] mechanism. Interpreting COIN functionality as service functions could make SFC applicable to COIN at Layer 2 and Layer 3, but also at name level [17].

Shaping future solutions. While the above concepts build upon existing communication principles, other efforts re-evaluate or even rethink addressing schemes. Jia et al. [18] provide a gap analysis of existing solutions (including the ones mentioned above), identifying a number of issues that arise from the specific point solutions to extending addressing. The authors argue for both *flexibility* and *extensibility* of addressing which are key aspects that any solution to the research questions outlined in Sec. IV would benefit from.

King and Farrel [19] provide an overview of efforts on addressing and routing that incorporate semantics beyond the one defined by IPv6, covering both existing IETF solutions and ongoing research; efforts they collectively term 'semantic routing'. In a companion document [20], King et al. outline several challenges that exist for such extensions of the addressing semantic, some of which align with the issues identified in this document. More importantly, they discuss the possible deployment of semantic routing solutions, e.g., as an overlay or limited to a single *Limited Domain* [21]. They further analyze corresponding challenges, some of which also apply to a COIN environment, while not being limited to it. Examples include the intended scope of any enhanced addressing (e.g., identifying *on-path* COIN elements) or the description of path characteristics that COIN traffic would need to adhere to, which is especially interesting considering our first design principle.

Work on *ICN*, e.g., in the IRTF ICN Research Group (ICN RG) [11], studies the addressing of information rather than endpoints, opening up the possibility of providing information from different sources, including COIN elements.

Khandaker et al. [8] address services directly as a named entity to support concepts like virtualization of service endpoints and provisioning within edge and in-network locations. Packets can be forwarded either through a shim layer (atop IPv6) routing capability or via ingress-based traffic steering.

Takeaway. *Overall, there seems to be wide-spread consensus that the existing addressing concepts are non-sufficient for the diverse use cases existing today. The introduction of COIN adds another view on this topic as COIN elements as well as different functionality residing on the same COIN element might have to be addressed as well. One of the main challenges will be how these different approaches might be brought together architecturally, as also argued by Jia et al. [18].*

B. Flow Granularity

Flow granularities are defined in transport protocols through their semantic for the unit of transfer. Upper layer protocols in turn map their application data with their own semantic into the transport semantic. Careful consideration of flow granularities is thus essential for sensible E2E functionality.

Shifting granularities. The initial HTTP, e.g., only allowed one request/response per TCP connection. With *persistent connections*, HTTP/1.1 enabled several *consecutive* interactions while HTTP/2 even allows for multiple *parallel* interactions using an additional stream abstraction, although still multiplexed over a single TCP connection. In HTTP/3, the different streams are even mapped to dedicated QUIC streams within the overall QUIC connection. While this growing complexity allows for higher efficiency at the end-hosts, COIN elements might be challenged by multiplexed information, especially in the fully-encrypted settings introduced by QUIC. In these instances, it is not possible to clearly distinguish the contained information into their individual streams, e.g., complicating congestion and error control. Since those mechanisms explicitly operate at the endpoint level, work is required that would allow for inserting functionality on-path but explicitly located at selected COIN elements.

Effect on affinity. The notion of flow granularity is used by Liu et al. [22] to link the relation of application level interactions to a specific service instance in scenarios where more than one service instance may serve requests for a given service. Here, the problem of *instance affinity* arises when needing to send one or more interactions to the same instance before being able to choose another instance (e.g., based on computing or network metrics). Khandaker et al. [8] propose the realization of instance affinity through an interplay of on-path name discovery and IPv6-based affinity relations, requiring changes to existing transport implementations, particularly in supporting multi-path relations where the initial discovery may traverse a different path from the IPv6-based packets.

Chain granularity. SFC allows to form a service chain, expressed through the NSH as entries into a next hop table maintained at each Service Function Forwarder (SFF) [15]. Packet classification takes place at the entry point of the chain, therefore providing a notion of flow granularity where the chain is treated as the 'unit of transfer'. Chaining can take place at Layer 2 or Layer 3, but also at a name-based layer (such as HTTP), as proposed by Trossen et al. [17].

New approaches. A recent proposal for a new transport protocol proposes a message-oriented flow granularity to achieve a better fit to COIN [23]. In particular, the authors argue that

TCP's stream abstraction is incompatible with many of the envisioned benefits of COIN, such as on-path data mutation and in-network caching. This consideration is supported by the fact that many COIN approaches leveraging PNDs choose UDP, i.e., a packet-based granularity, for easier handling [24].

Takeaway. *TCP's stream abstraction has long been used in many applications. Yet, recent works suggest that it might not be the best fit considering COIN, instead proposing alternatives which need better understanding as to their benefits.*

C. Collective Communication

The existing TCP/IP networking stack is generally designed for unicast delivery, while IP multicast provides network support for group-based delivery of packets to many recipients. With COIN, however, collective communication, i.e., communication to several albeit often changing communication endpoints, is gaining importance, e.g., for distributed AI.

Work in the ICN RG considers multicast and unicast delivery as communication models realized by the same communication method, e.g., utilizing an interest-data model. Trossen [25] formalizes this approach by defining an ad-hoc multicast semantic labelled *forward request return multicast*, where the return path multicast is achieved through utilizing information from incoming service requests. The utilized transport network technology may be that of SDN or BIER [26], where the former uses an OpenFlow-compatible approach to path-based forwarding with constant state requirements for the in-network forwarders.

As an impact to the transport layer, Trossen proposes to separate longer-lived resource management from shorter-lived transaction handling to increase efficiency of the ephemeral return path communication at the transport level.

Takeaway. *While IP multicast has long established multicast operations at the network level, new forms of supporting collective communication are still in their infancy. Nonetheless, they are of great importance to provide network level support for some of the foreseen COIN use cases [7].*

VI. Road Ahead

There is a continued innovation in packet processing technologies, e.g., evidenced by the introduction of SDN and P4. This continuous evolution pushes the boundaries of what future PNDs may evolve into. More specifically, we foresee capabilities of today's edge gradually transitioning into future PNDs, therefore pushing the limits of what can be achieved on-path closer to those of existing edge computing approaches.

These evolving capabilities of COIN, however, also pose the challenge of programming distributed capabilities, adhering to our two COIN design principles (cf. Sec. III). Future protocols will need to integrate with suitable *programming environments* to place, migrate, and terminate those 'incomplete versions of the function' foreseen by the E2E argument.

With this and in contrast to Saltzer et al. [1], we see these environments as evolving from the design time to runtime, where the rather static capability placement of today's orchestration frameworks shifts to one that allows for doing so under changing conditions, both of network and communication endpoints. As a consequence, we envision the road ahead for COIN as one of distributing protocols and computations as sets of functions across endpoints and networking devices alike, doing so automatically, intelligently, and transparently.

References

[1] J. Saltzer, D. Reed, and D. Clark, "End-To-End Arguments in System Design," *ACM TOCS*, vol. 2, no. 4, 1984.

[2] B. Carpenter, "Architectural Principles of the Internet," IETF, RFC 1958, 1996.

[3] ——, "Internet Transparency," IETF, RFC 2775, 2000.

[4] J. Quarterman and S. Carl-Mitchell, "What is the Internet, Anyway?" IETF, RFC 1935, 1996.

[5] R. Bush and D. Meyer, "Some Internet Architectural Guidelines and Philosophy," IETF, RFC 3439, 2002.

[6] G. Fairhurst, A. Sathiaseelan, H. Cruickshank, and C. Baudoin, "Transport challenges facing a next-generation hybrid satellite Internet," *Int. J. Satell. Commun. Network.*, vol. 29, no. 3, 2011.

[7] I. Kunze, K. Wehrle, D. Trossen, M. Montpetit, X. de Foy, D. Griffin, and M. Rio, "Use Cases for In-Network Computing," IRTF, Internet-Draft, 2022, work in Progress.

[8] K. Khandaker, D. Trossen, R. Khalili, Z. Despotovic, A. Hecker, and G. Carle, "CArDS: Dealing a New Hand in Reducing Service Request Completion Times," in *IFIP Networking '22*, 2022.

[9] R. Glebke, D. Trossen, I. Kunze, D. Lou, J. Rüth, M. Stoffers, and K. Wehrle, "Service-based Forwarding via Programmable Dataplanes," in *HSPR '21*, 2021.

[10] J. Sobrinho and M. Ferreira, "Routing on Multiple Optimality Criteria," in *ACM SIGCOMM '20*, 2020.

[11] "Information-Centric Networking, IRTF Research Group," 2022. [Online]. Available: https://datatracker.ietf.org/rg/icnrg/about/

[12] S. Li, R. Yeung, and N. Cai, "Linear Network Coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, 2003.

[13] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture," IETF, RFC 8402, 2018.

[14] P. Quinn and T. Nadeau, "Problem Statement for Service Function Chaining," IETF, RFC 7498, 2015.

[15] J. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," IETF, RFC 7665, 2015.

[16] P. Quinn, U. Elzur, and C. Pignataro, "Network Service Header (NSH)," IETF, RFC 8300, 2018.

[17] D. Trossen, D. Purkayastha, and A. Rahman, "Name-Based Service Function Forwarder (nSFF) Component within a Service Function Chaining (SFC) Framework," IETF, RFC 8677, 2019.

[18] Y. Jia, D. Trossen, L. Iannone, P. Mendes, N. Shenoy, L. Toutain, A. Chen, and D. Farinacci, "Gap Analysis in Internet Addressing," IETF, Internet-Draft, 2022, work in Progress.

[19] D. King and A. Farrel, "A Survey of Semantic Internet Routing Techniques," IETF, Internet-Draft, 2021, work in Progress.

[20] D. King, A. Farrel, and C. Jacquenet, "Challenges for the Internet Routing Infrastructure Introduced by Semantic Routing," IETF, Internet-Draft, 2022, work in Progress.

[21] B. Carpenter and B. Liu, "Limited Domains and Internet Protocols," IETF, RFC 8799, 2020.

[22] P. Liu, P. Eardley, D. Trossen, M. Boucadair, L. Contreras, and C. Li, "Dynamic-Anycast (Dyncast) Use Cases and Problem Statement," IETF, Internet-Draft, 2022, work in Progress.

[23] B. Stephens, D. Grassi, H. Almasi, T. Ji, B. Vamanan, and A. Akella, "TCP is Harmful to In-Network Computing: Designing a Message Transport Protocol (MTP)," in *HotNets '21*, 2021.

[24] F. Hauser, M. Häberle, D. Merling, S. Lindner, V. Gurevich, F. Zeiger, R. Frank, and M. Menth, "A Survey on Data Plane Programming with P4: Fundamentals, Advances, and Applied Research," 2021. [Online]. Available: http://arxiv.org/abs/2101.10632

[25] D. Trossen, "Realizing Forward Requests Return Multicast Semantic with BIER," IETF, Internet-Draft, 2022, work in Progress.

[26] "Bit Indexed Binary Replication, IETF Working Group," 2022. [Online]. Available: https://datatracker.ietf.org/wg/bier/about/