Stream: Internet Research Task Force (IRTF)
RFC: 9817
Category: Informational
Published: July 2025
ISSN: 2070-1721
Authors:
I. Kunze, *RWTH Aachen*
K. Wehrle, *RWTH Aachen*
D. Trossen, *Huawei*
M-J. Montpetit, *McGill*
X. de Foy, *InterDigital Communications, LLC*
D. Griffin, *UCL*
M. Rio, *UCL*

RFC 9817
Use Cases for In-Network Computing

Abstract

Computing in the Network (COIN) comes with the prospect of deploying processing functionality on networking devices such as switches and network interface cards. While such functionality can be beneficial, it has to be carefully placed into the context of the general Internet communication, and it needs to be clearly identified where and how those benefits apply.

This document presents some use cases to demonstrate how a number of salient COIN-related applications can benefit from COIN. Furthermore, to guide research on COIN, it identifies essential research questions and outlines desirable capabilities that COIN systems addressing these use cases may need to support. Finally, the document provides a preliminary categorization of the described research questions to source future work in this domain. This document is a product of the Computing in the Network Research Group (COINRG). It is not an IETF product and it is not a standard.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Research Task Force (IRTF). The IRTF publishes the results of Internet-related research and development activities. These results might not be suitable for deployment. This RFC represents the consensus of the Computing in the Network (COIN) Research Group of the Internet Research Task Force (IRTF). Documents approved for publication by the IRSG are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9817.

Copyright Notice

Table of Contents

1. Introduction

The Internet was designed as a best-effort packet network, forwarding packets from source to destination with limited guarantees regarding their timely and successful reception. Data manipulation, computation, and more complex protocol functionalities are generally provided by the end hosts, while network nodes are traditionally kept simple and only offer a "store and forward" packet facility. This simplicity of purpose of the network has shown to be suitable for a wide variety of applications and has facilitated the rapid growth of the Internet. However, introducing middleboxes with specialized functionality for enhancing performance has often led to problems due to their inflexibility.

However, with the rise of new services, some of which are described in this document, there is a growing number of application domains that require more than best-effort forwarding, including strict performance guarantees or closed-loop integration to manage data flows. In this context, allowing for a tighter integration of computing and networking resources for enabling a more flexible distribution of computation tasks across the network (e.g., beyond "just" endpoints and without requiring specialized middleboxes) may help to achieve the desired guarantees and behaviors, increase overall performance, and improve resilience to failures.

The vision of "in-network computing" and the provisioning of such capabilities that capitalize on joint computation and communication resource usage throughout the network is part of the move from a telephone network analogy of the Internet into a more distributed computer board architecture. We refer to those capabilities as "COIN capabilities" in the remainder of the document.

We believe that this vision of in-network computing can be best outlined along four dimensions of use cases, namely those that:

i. provide new user experiences through the utilization of COIN capabilities (referred to as "COIN experiences"),
ii. enable new COIN systems (e.g., through new interactions between communication and compute providers),
iii. improve on already existing COIN capabilities, and
iv. enable new COIN capabilities.

Sections 3 through 6 capture those categories of use cases and provide the main structure of this document. The goal is to present how computing resources inside the network impact existing services and applications or allow for innovation in emerging application domains.

By delving into some individual examples within each of the above categories, we outline opportunities and propose possible research questions for consideration by the wider community when pushing forward in-network computing architectures. Furthermore, identifying desirable capabilities for an evolving solution space of COIN systems is another objective of the use case descriptions. To achieve this, the following taxonomy is proposed to describe each of the use cases:

Description: A high-level presentation of the purpose of the use case and a short explanation of the use case behavior.

Characterization: An explanation of the services that are being utilized and realized as well as the semantics of interactions in the use case.

Existing Solutions: A description of current methods that may realize the use case (if they exist), though not claiming to exhaustively review the landscape of solutions.

Opportunities: An outline of how COIN capabilities may support or improve on the use case in terms of performance and other metrics.

Research questions: Essential questions that are suitable for guiding research to achieve the identified opportunities. The research questions also capture immediate capabilities for any COIN solution addressing the particular use case whose development may immediately follow when working toward answers to the research questions.

Additional desirable capabilities: A description of additional capabilities that might not require research but may be desirable for any COIN solution addressing the particular use case; we limit these capabilities to those directly affecting COIN, recognizing that any use case will realistically require many additional capabilities for its realization. We omit this dedicated section if relevant capabilities are already sufficiently covered by the corresponding research questions.

This document discusses these six aspects along a number of individual use cases to demonstrate the diversity of COIN applications. It is intended as a basis for further analyses and discussions within the wider research community. This document represents the consensus of COINRG.

2. Terminology

This document uses the terminology defined below.

Programmable Network Devices (PNDs): network devices, such as network interface cards and switches, which are programmable (e.g., using P4 [P4] or other languages).

(COIN) execution environment: a class of target environments for function execution, for example, an execution environment based on the Java Virtual Machine (JVM) that can run functions represented in JVM byte code.

COIN system: the PNDs (and end systems) and their execution environments, together with the communication resources interconnecting them, operated by a single provider or through interactions between multiple providers that jointly offer COIN capabilities.

COIN capability: a feature enabled through the joint processing of computation and communication resources in the network.

(COIN) program: a monolithic functionality that is provided according to the specification for said program and which may be requested by a user. A composite service can be built by orchestrating a combination of monolithic COIN programs.

(COIN) program instance: one running instance of a program.

COIN experience: a new user experience brought about through the utilization of COIN capabilities.

3. Providing New COIN Experiences

3.1. Mobile Application Offloading

3.1.1. Description

This scenario can be exemplified in an immersive gaming application, where a single user plays a game using a Virtual Reality (VR) headset. The headset hosts several (COIN) programs. For instance, the display (COIN) program renders frames to the user, while other programs are realized for VR content processing and to incorporate input data received from sensors (e.g., in bodily worn devices including the VR headset).

Once this application is partitioned into its constituent (COIN) programs and deployed throughout a COIN system, utilizing a COIN execution environment, only the display (COIN) program may be left in the headset, while the compute intensive real-time VR content processing (COIN) program can be offloaded to a nearby resource rich home PC or a Programmable Network Device (PND) in the operator's access network, for a better execution (faster and possibly higher resolution generation).

3.1.2. Characterization

Partitioning a mobile application into several constituent (COIN) programs allows for denoting the application as a collection of (COIN) programs for a flexible composition and a distributed execution. In our example above, most capabilities of a mobile application can be categorized into any of three groups: receiving, processing, and displaying.

Any device may realize one or more of the (COIN) programs of a mobile application and expose them to the (COIN) system and its constituent (COIN) execution environments. When the (COIN) program sequence is executed on a single device, the outcome is what you traditionally see with applications running on mobile devices.

However, the execution of a (COIN) program may be moved to other (e.g., more suitable) devices, including PNDs, which have exposed the corresponding (COIN) program as individual (COIN) program instances to the (COIN) system by means of a service identifier. The result is the equivalent to mobile function offloading, for possible reduction of power consumption (e.g., offloading CPU-intensive process functions to a remote server) or for improved end-user experience (e.g., moving display functions to a nearby smart TV) by selecting more suitably placed (COIN) program instances in the overall (COIN) system.

We can already see a trend toward supporting such functionality with relyiccng on dedicated cloud hardware (e.g., gaming platforms rendering content externally). We envision, however, that such functionality is becoming more pervasive through specific facilities, such as entertainment parks or even hotels, in order to deploy needed edge computing capabilities to enable localized gaming as well as non-gaming scenarios.

Figure 1 shows one realization of the above scenario, where a "DPR app" is running on a mobile device (containing the partitioned COIN programs Display (D), Process (P) and Receive (R)) over a programmable switching network, e.g., a Software-Defined Network (SDN) here. The packaged applications are made available through a localized "playstore server". The mobile application installation is realized as a service deployment process, combining the local app installation with a distributed deployment (and orchestration) of one or more (COIN) programs on most suitable end systems or PNDs (here, a "processing server").

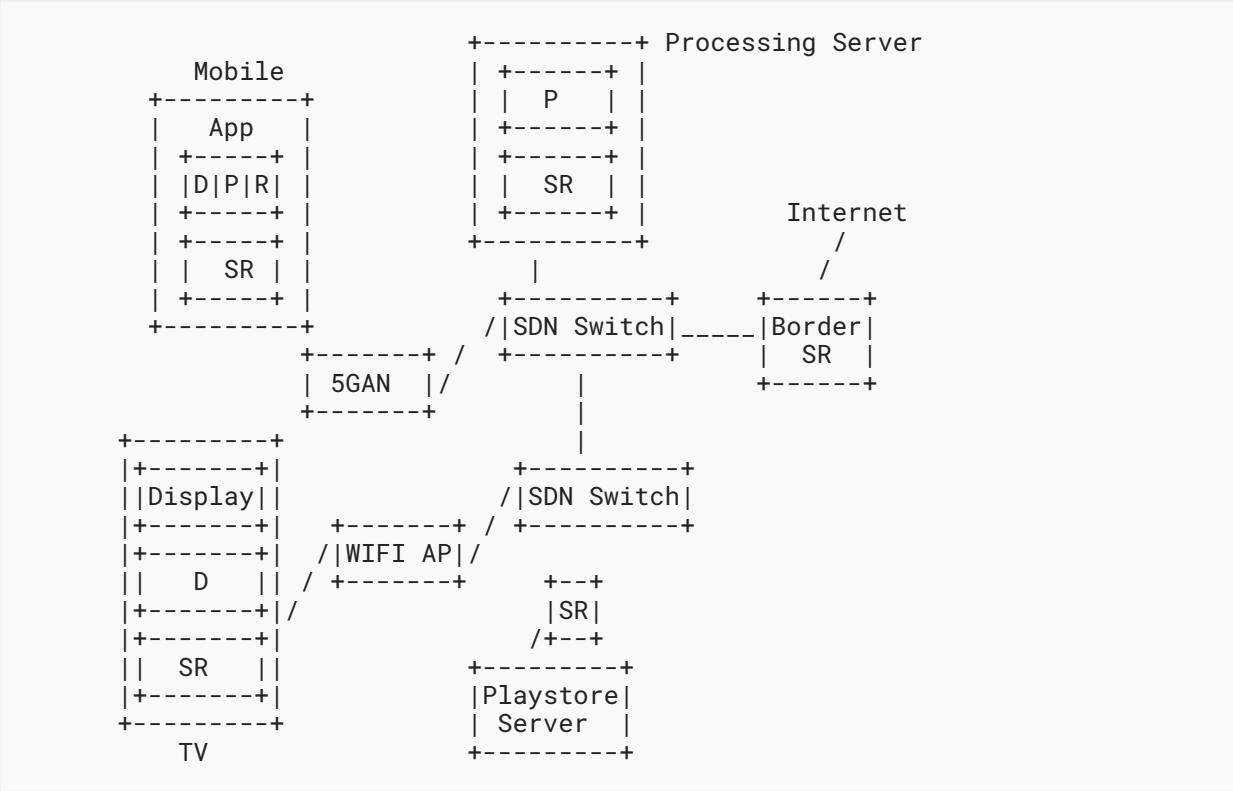

Figure 1: Application Function Offloading Example

Such localized deployment could, for instance, be provided by a visiting site, such as a hotel or a theme park. Once the processing (COIN) program is terminated on the mobile device, the "service routing (SR)" elements in the network route (service) requests instead to the (previously deployed) processing (COIN) program running on the processing server over an existing SDN network. Here, capabilities and other constraints for selecting the appropriate (COIN) program, in case of having deployed more than one, may be provided both in the advertisement of the (COIN) program and the service request itself.

As an extension to the above scenarios, we can also envision that content from one processing (COIN) program may be distributed to more than one display (COIN) program (e.g., for multi- and many-viewing scenarios). Here, an offloaded processing program may collate input from several users in the (virtual) environment to generate a possibly three-dimensional render that is then distributed via a service-level multicast capability towards more than one display (COIN) program.

3.1.3. Existing Solutions

The ETSI Mobile Edge Computing (MEC) [ETSI] suite of technologies provides solutions for mobile function offloading by allowing mobile applications to select resources in edge devices to execute functions instead of the mobile device directly. For this, ETSI MEC utilizes a set of interfaces for the selection of suitable edge resources, connecting to so-called MEC application servers, while also allowing for sending data for function execution to the application server.

However, the technologies do not utilize microservices [Microservices]; they mainly rely on virtualization approaches such as containers or virtual machines, thus requiring a heavier processing and memory footprint in a COIN execution environment and the executing intermediaries. Also, the ETSI work does not allow for the dynamic selection and redirection of (COIN) program calls to varying edge resources rather than a single MEC application server.

Also, the selection of the edge resource (the app server) is relatively static, relying on DNS-based endpoint selection, which does not cater to the requirements of the example provided above, where the latency for redirecting to another device lies within a few milliseconds for aligning with the frame rate of the display microservice.

Lastly, MEC application servers are usually considered resources provided by the network operator through its MEC infrastructure, while our use case here also foresees the placement and execution of microservices in end-user devices.

There also exists a plethora of mobile offloading platforms provided through proprietary platforms, all of which follow a similar approach as ETSI MEC in that a selected edge application server is being utilized to send functional descriptions and data for execution.

[APPCENTRES] outlines a number of enabling technologies for the use case, some of which have been realized in an Android-based realization of the microservices as a single application, which is capable of dynamically redirecting traffic to other microservice instances in the network. This capability, together with the underlying path-based forwarding capability (using SDN), was demonstrated publicly (e.g., at the Mobile World Congress 2018 and 2019).

3.1.4. Opportunities

- The packaging of (COIN) programs into existing mobile application packaging may enable the migration from current (mobile) device-centric execution of those mobile applications toward a possible distributed execution of the constituent (COIN) programs that are part of the overall mobile application.
- The orchestration for deploying (COIN) program instances in specific end systems and PNDs alike may open up the possibility for localized infrastructure owners, such as hotels or venue owners, to offer their compute capabilities to their visitors for improved or even site-specific experiences.
- The execution of (current mobile) app-level (COIN) programs may speed up the execution of said (COIN) program by relocating the execution to more suitable devices, including PNDs that may reside better located in relation to other (COIN) programs and thus improve performance, such as latency.
- The support for service-level routing of requests (such as service routing in [APPCENTRES]) may support higher flexibility when switching from one (COIN) program instance to another (e.g., due to changing constraints for selecting the new (COIN) program instance). Here, PNDs may support service routing solutions by acting as routing overlay nodes to implement the necessary additional lookup functionality and also possibly support the handling of affinity relations (i.e., the forwarding of one packet to the destination of a previous one due to a higher level service relation as discussed and described in [SarNet2021]).

- The ability to identify service-level COIN elements will allow for routing service requests to those COIN elements, including PNDs, therefore possibly allowing for a new COIN functionality to be included in the mobile application.
- The support for constraint-based selection of a specific (COIN) program instance over others (e.g., constraint-based routing in [APPCENTRES], showcased for PNDs in [SarNet2021]) may allow for a more flexible and app-specific selection of (COIN) program instances, thereby allowing for better meeting the app-specific and end-user requirements.

3.1.5. Research Questions

- RQ 3.1.1: How to combine service-level orchestration frameworks, such as TOSCA orchestration templates [TOSCA], with app-level (e.g., mobile application) packaging methods, ultimately providing the means for packaging microservices for deployments in distributed networked computing environments?
- RQ 3.1.2: How to reduce latencies involved in (COIN) program interactions where (COIN) program instance locations may change quickly? Can service-level requests be routed directly through in-band signaling methods instead of relying on out-of-band discovery, such as through the DNS?
- RQ 3.1.3: How to signal constraints used for routing requests towards (COIN) program instances in a scalable manner (i.e., for dynamically choosing the best possible service sequence of one or more (COIN) programs for a given application experience through chaining (COIN) program executions)?
- RQ 3.1.4: How to identify (COIN) programs and program instances so as to allow routing (service) requests to specific instances of a given service?
- RQ 3.1.5: How to identify a specific choice of a (COIN) program instance over others, thus allowing pinning the execution of a service of a specific (COIN) program to a specific resource (i.e., a (COIN) program instance in the distributed environment)?
- RQ 3.1.6: How to provide affinity of service requests towards (COIN) program instances (i.e., longer-term transactions with ephemeral state established at a specific (COIN) program instance)?
- RQ 3.1.7: How to provide constraint-based routing decisions that can be realized at packet forwarding speed (e.g., using techniques explored in [SarNet2021] at the forwarding plane or using approaches like [Multi2020] for extended routing protocols)?
- RQ 3.1.8: What COIN capabilities may support the execution of (COIN) programs and their instances?
- RQ 3.1.9: How to ensure real-time synchronization and consistency of distributed application states among (COIN) program instances, in particular, when frequently changing the choice for a particular (COIN) program in terms of executing a service instance?

3.2. Extended Reality and Immersive Media

3.2.1. Description

Extended Reality (XR) encompasses VR, Augmented Reality (AR) and Mixed Reality (MR). It provides the basis for the metaverse and is the driver of a number of advances in interactive technologies. While initially associated with gaming and immersive entertainment, applications

now include remote diagnosis, maintenance, telemedicine, manufacturing and assembly, intelligent agriculture, smart cities, and immersive classrooms. XR is one example of the multisource-multidestination problem that combines video and haptics in interactive multiparty interactions under strict delay requirements. As such, XR can benefit from a functional distribution that includes fog computing for local information processing, the edge for aggregation, and the cloud for image processing.

XR stands to benefit significantly from computing capabilities in the network. For example, XR applications can offload intensive processing tasks to edge servers, considerably reducing latency when compared to cloud-based applications and enhancing the overall user experience. More importantly, COIN can enable collaborative XR experiences, where multiple users interact in the same virtual space in real time, regardless of their physical locations, by allowing resource discovery and re-rerouting of XR streams. While not a feature of most XR implementations, this capability opens up new possibilities for remote collaboration, training, and entertainment. Furthermore, COIN can support dynamic content delivery, allowing XR applications to seamlessly adapt to changing environments and user interactions. Hence, the integration of computing capabilities into the network architecture enhances the scalability, flexibility, and performance of XR applications by supplying telemetry and advanced stream management, paving the way for more immersive and interactive experiences.

Indeed, XR applications require real-time interactivity for immersive and increasingly mobile applications with tactile and time-sensitive data. Because high bandwidth is needed for high resolution images and local rendering for 3D images and holograms, strictly relying on cloud-based architectures, even with headset processing, limits some of its potential benefits in the collaborative space. As a consequence, innovation is needed to unlock the full potential of XR.

3.2.2. Characterization

As mentioned above, XR experiences, especially those involving collaboration, are difficult to deliver with a client-server cloud-based solution. This is because they require a combination of multistream aggregation, low delays and delay variations, means to recover from losses, and optimized caching and rendering as close as possible to the user at the network edge. Hence, implementing such XR solutions necessitates substantial computational power and minimal latency, which, for now, has spurred the development of better headsets not networked or distributed solutions as factors like distance from cloud servers and limited bandwidth can still significantly lower application responsiveness. Furthermore, when XR deals with sensitive information, XR applications must also provide a secure environment and ensure user privacy, which represent additional burdens for delay-sensitive applications. Additionally, the sheer amount of data needed for and generated by XR applications, such as video holography, put them squarely in the realm of data-driven applications that can use recent trend analysis and mechanisms, as well as machine learning, in order to find the optimal caching and processing solution and ideally reduce the size of the data that needs transiting through the network. Other mechanisms, such as data filtering and reduction, and functional distribution and partitioning, are also needed to accommodate the low delay needs for the same applications.

With functional decomposition as the goal of a better XR experience, the elements involved in a COIN XR implementation include:

- the XR application residing in the headset,
- edge federation services that allow local devices to communicate with one another directly,
- edge application servers that enable local processing but also intelligent stream aggregation to reduce bandwidth requirements,
- edge data networks that allow precaching of information based on locality and usage,
- cloud-based services for image processing and application training, and
- intelligent 5G/6G core networks for managing advanced access services and providing performance data for XR stream management.

These characteristics of XR paired with the capabilities of COIN make it likely that COIN can help to realize XR over networks for collaborative applications. In particular, COIN functions can enable the distribution of the service components across different nodes in the network. For example, data filtering, image rendering, and video processing leverage different hardware capabilities with combinations of CPUs and Graphics Processing Units (GPUs) at the network edge and in the fog, where the content is consumed. These represent possible remedies for the high bandwidth demands of XR. Machine learning across the network nodes can better manage the data flows by distributing them over more adequate paths. In order to provide adequate quality of experience, multivariate and heterogeneous resource allocation and goal optimization problems need to be solved, likely requiring advanced analysis and artificial intelligence. For the purpose of this document, it is important to note that the use of COIN for XR does not imply a specific protocol but targets an architecture enabling the deployment of the services. In this context, similar considerations as for Section 3.1 apply.

3.2.3. Existing Solutions

The XR field has profited from extensive research in the past years in gaming, machine learning, network telemetry, high resolution imaging, smart cities, and the Internet of Things (IoT). Information-centric networking (and related) approaches that combine, publish, subscribe, and distribute storage are also very suited for the multisource-multidestination applications of XR. New AR and VR headsets and glasses have continued to evolve towards autonomy with local computation capabilities, increasingly performing much of the processing that is needed to render and augment the local images. Mechanisms aimed at enhancing the computational and storage capacities of mobile devices could also improve XR capabilities as they include the discovery of available servers within the environment and using them opportunistically to enhance the performance of interactive applications and distributed file systems.

While there is still no specific COIN research in AR and VR, the need for network support is important to offload some of the computations related to movement, multiuser interactions, and networked applications, notably in gaming but also in health [NetworkedVR]. This new approach to networked AR and VR is exemplified in [eCAR] by using synchronized messaging at the edge to share the information that all users need to interact. In [CompNet2021] and [WirelessNet2024], the offloading uses Artificial Intelligence (AI) to assign the 5G resources necessary for the real-time interactions, and one could think that implementing this scheme on a

PND is essentially a natural next step. Hence, as AR, VR, and XR are increasingly becoming more interactive, the efficiency needed to implement novel applications will require some form or another of edge-core implementation and COIN support.

In summary, some XR solutions exist, and headsets continue to evolve to what is now claimed to be spatial computing. Additionally, with recent work on the metaverse, the number of publications related to XR has skyrocketed. However, in terms of networking, which is the focus of this document, current deployments do not take advantage of network capabilities. The information is rendered and displayed based on the local processing but does not readily discover the other elements in the vicinity or in the network that could improve its performance either locally, at the edge, or in the cloud. Yet, there are still very few interactive and immersive media applications over networks that allow for federating systems capabilities.

3.2.4. Opportunities

While delay is inherently related to information transmission, if we continue the analogy of the computer board to highlight some of the COIN capabilities in terms of computation and storage but also allocation of resources, there are some opportunities that XR could take advantage of:

- Round trip time: 20 ms is usually cited as an upper limit for XR applications. Storage and preprocessing of scenes in local elements (including in the mobile network) could extend the reach of XR applications at least over the extended edge.
- Video transmission: The use of better transcoding, advanced context-based compression algorithms, prefetching and precaching, as well as movement prediction all help to reduce bandwidth consumption. While this is now limited to local processing, it is not outside the realm of COIN to push some of these functionalities to the network, especially as related to caching and fetching but also context-based flow direction and aggregation.
- Monitoring: Since bandwidth and data are fundamental to XR deployment, COIN functionality could help to better monitor and distribute the XR services over collaborating network elements to optimize end-to-end performance.
- Functional decomposition: Advanced functional decomposition, localization, and discovery of computing and storage resources in the network can help to optimize user experience in general.
- Intelligent network management and configuration: The move to AI in network management to learn about flows and adapt resources based on both data plane and control plane programmability can help the overall deployment of XR services.

3.2.5. Research Questions

- RQ 3.2.1: Can current PNDs provide the speed required for executing complex filtering operations, including metadata analysis for complex and dynamic scene rendering?
- RQ 3.2.2: Where should PNDs equipped with these operations be located for optimal performance gains?
- RQ 3.2.3: Can the use of distributed AI algorithms across both data center and edge computers be leveraged for creating optimal function allocation and the creation of semi-permanent datasets and analytics for usage trending and flow management resulting in better localization of XR functions?

- RQ 3.2.4: Can COIN improve the dynamic distribution of control, forwarding, and storage resources and related usage models in XR, such as to integrate local and fog caching with cloud-based pre-rendering, thus jointly optimizing COIN and higher layer protocols to reduce latency and, more generally, manage the quality of XR sessions (e.g., through reduced in-network congestion and improved flow delivery by determining how to prioritize XR data)?
- RQ 3.2.5: Can COIN provide the necessary infrastructure for the use of interactive XR everywhere? Particularly, how can a COIN system enable the joint collaboration across all segments of the network (fog, edge, core, and cloud) to support functional decompositions, including using edge resources without the need for a (remote) cloud connection?
- RQ 3.2.6: How can COIN systems provide multistream efficient transmission and stream combining at the edge, including the ability to dynamically include extra streams, such as audio and extra video tracks?

3.2.6. Additional Desirable Capabilities

In addition to the capabilities driven by the research questions above, there are a number of other features that solutions in this space might benefit from. In particular, the provided XR experience should be optimized both in the amount of transmitted data, while equally optimizing loss protection. Furthermore, the means for trend analysis and telemetry to measure performance may foster uptake of the XR services, while the interaction of the XR system with indoor and outdoor positioning systems may improve on service experience and user perception.

3.3. Personalized and Interactive Performing Arts

3.3.1. Description

This use case is a deeper dive into a specific scenario of the immersive and extended reality class of use cases discussed in Section 3.2. It focuses on live productions of the performing arts where the performers and audience members are geographically distributed. The performance is conveyed through multiple networked streams, which are tailored to the requirements of the remote performers, the director, the sound and lighting technicians, and the individual audience members. Performers need to observe, interact, and synchronize with other performers in remote locations, and the performers receive live feedback from the audience, which may also be conveyed to other audience members.

There are two main aspects:

i. to emulate as closely as possible the experience of live performances where the performers, audience, director, and technicians are co-located in the same physical space, such as a theater; and
ii. to enhance traditional physical performances with features such as personalization of the experience according to the preferences or needs of the performers, directors, and audience members.

Examples of personalization include:

- Viewpoint selection, such as choosing a specific seat in the theater or for more advanced positioning of the audience member's viewpoint outside of the traditional seating (i.e., amongst, above, or behind the performers, but within some limits that may be imposed by the performers or the director for artistic reasons);
- Augmentation of the performance with subtitles, audio description, actor tagging, language translation, advertisements and product placement, and other enhancements and filters to make the performance accessible to audience members who are disabled (e.g., the removal of flashing images for audience members who have epilepsy or alternative color schemes for those who have color blindness).

3.3.2. Characterization

There are several chained functional entities that are candidates for being deployed as (COIN) programs:

- Performer aggregation and editing functions
- Distribution and encoding functions
- Personalization functions
  - to select which of the existing streams should be forwarded to the audience member, remote performer, or member of the production team
  - to augment streams with additional metadata such as subtitles
  - to create new streams after processing existing ones (e.g., to interpolate between camera angles to create a new viewpoint or to render point clouds from an audience member's chosen perspective)
  - to undertake remote rendering according to viewer position (e.g., the creation of VR headset display streams according to audience head position) when this processing has been offloaded from the viewer's end system to the COIN function due to limited processing power in the end system or due to limited network bandwidth to receive all of the individual streams to be processed.
- Audience feedback sensor processing functions
- Audience feedback aggregation functions

These are candidates for deployment as (COIN) programs in PNDs rather than being located in end systems (at the performers' site, the audience members' premises, or in a central cloud location) for several reasons:

- Personalization of the performance according to viewer preferences and requirements makes it infeasible to be done in a centralized manner at the performer premises: the computational resources and network bandwidth would need to scale with the number of personalized streams.
- Rendering of VR headset content to follow viewer head movements has an upper bound on lag to maintain viewer Quality of Experience (QoE), which requires the processing to be undertaken sufficiently close to the viewer to avoid large network latencies.

- Viewer devices may not have the processing power to perform the personalization tasks, or the viewers' network may not have the capacity to receive all of the constituent streams to undertake the personalization functions.
- There are strict latency requirements for live and interactive aspects that require the deviation from the direct network path between performers and audience members to be minimized, which reduces the opportunity to route streams via large-scale processing capabilities at centralized data centers.

3.3.3. Existing Solutions

Note: Existing solutions for some aspects of this use case are covered in Section 3.1, Section 3.2, and Section 5.1.

3.3.4. Opportunities

- Executing media processing and personalization functions on-path as (COIN) programs in PNDs can avoid detour/stretch to central servers, thus reducing latency and bandwidth consumption. For example, the overall delay for performance capture, aggregation, distribution, personalization, consumption, capture of audience response, feedback processing, aggregation, and rendering should be achieved within an upper bound of latency (the tolerable amount is to be defined, but in the order of 100s of ms to mimic performers perceiving audience feedback, such as laughter or other emotional responses in a theater setting).
- Processing of media streams allows (COIN) programs, PNDs, and the wider (COIN) system/environment to be contextually aware of flows and their requirements, which can be used for determining network treatment of the flows (e.g., path selection, prioritization, multiflow coordination, synchronization, and resilience).

3.3.5. Research Questions

- RQ 3.3.1: In which PNDs should (COIN) programs for aggregation, encoding, and personalization functions be located? Close to the performers or close to the viewers?
- RQ 3.3.2: How far from the direct network path from performer to viewer should (COIN) programs be located, considering the latency implications of path-stretch and the availability of processing capacity at PNDs? How should tolerances be defined by users?
- RQ 3.3.3: Should users decide which PNDs should be used for executing (COIN) programs for their flows, or should they express requirements and constraints that will direct decisions by the orchestrator/manager of a COIN system? In case of the latter, how can users specify requirements on network and processing metrics (such as latency and throughput bounds)?
- RQ 3.3.4: How to achieve synchronization across multiple streams to allow for merging, audio-video interpolation, and other cross-stream processing functions that require time synchronization for the integrity of the output? How can this be achieved considering that synchronization may be required between flows that are:

  i. on the same data pathway through a PND/router,

  ii. arriving/leaving through different ingress/egress interfaces of the same PND/router, or

  iii. routed through disjoint paths through different PNDs/routers?

This RQ raises issues associated with synchronization across multiple media streams and substreams [RFC7272] as well as time synchronization between PNDs/routers on multiple paths [RFC8039].

- RQ 3.3.5: Where will COIN programs be executed? In the data plane of PNDs, in other on-router computational capabilities within PNDs, or in adjacent computational nodes?
- RQ 3.3.6: Are computationally intensive tasks, such as video stitching or media recognition and annotation (cf. Section 3.2), considered as suitable candidate (COIN) programs or should they be implemented in end systems?
- RQ 3.3.7: If the execution of COIN programs is offloaded to computational nodes outside of PNDs (e.g., for processing by GPUs), should this still be considered as COIN? Where is the boundary between COIN capabilities and explicit routing of flows to end systems?

3.3.6. Additional Desirable Capabilities

In addition to the capabilities driven by the research questions above, there are a number of other features that solutions in this space might benefit from. In particular, if users are indeed empowered to specify requirements on network and processing metrics, one important capability of COIN systems will be to respect these user-specified requirements and constraints when routing flows and selecting PNDs for executing (COIN) programs. Similarly, solutions should be able to synchronize flow treatment and processing across multiple related flows, which may be on disjoint paths, to provide similar performance to different entities.

4. Supporting New COIN Systems

4.1. In-Network Control / Time-Sensitive Applications

4.1.1. Description

The control of physical processes and components of industrial production lines is essential for the growing automation of production and ideally allows for a consistent quality level. Traditionally, the control has been exercised by control software running on Programmable Logic Controllers (PLCs) located directly next to the controlled process or component. This approach is best suited for settings with a simple model that is focused on a single or a few controlled components.

Modern production lines and shop floors are characterized by an increasing number of involved devices and sensors, a growing level of dependency between the different components, and more complex control models. A centralized control is desirable to manage the large amount of available information, which often has to be preprocessed or aggregated with other information before it can be used. As a result, computations are increasingly spatially decoupled and moved away from the controlled objects, thus inducing additional latency. Instead, moving compute functionality onto COIN execution environments inside the network offers a new solution space to these challenges, providing new compute locations with much smaller latencies.

4.1.2. Characterization

A control process consists of two main components as illustrated in Figure 2: a system under control and a controller. In feedback control, the current state of the system is monitored (e.g., using sensors), and the controller influences the system based on the difference between the current and the reference state to keep it close to this reference state.

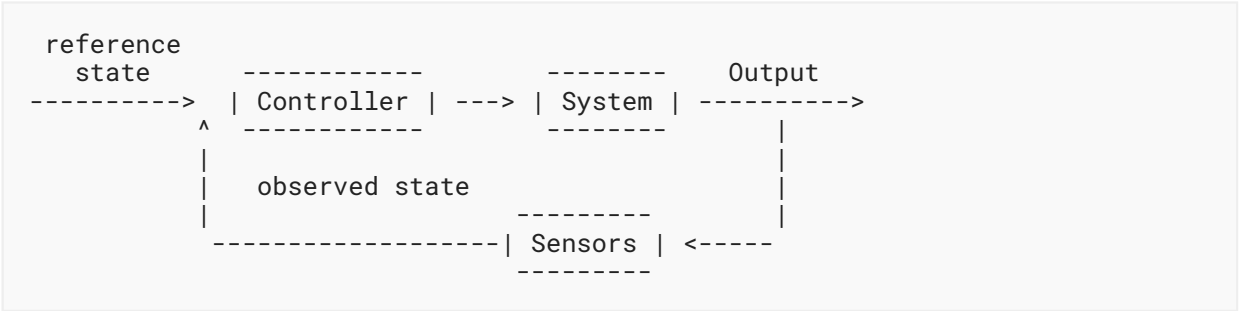

Figure 2: Simple Feedback Control Model

Apart from the control model, the quality of the control primarily depends on the timely reception of the sensor feedback, which can be subject to tight latency constraints, often in the single-digit millisecond range. Even shorter feedback requirements may exist in other use cases, such as interferometry or high-energy physics, but these use cases are out of scope for this document. While low latencies are essential, there is an even greater need for stable and deterministic levels of latency, because controllers can generally cope with different levels of latency if they are designed for them, but they are significantly challenged by dynamically changing or unstable latencies. The unpredictable latency of the Internet exemplifies this problem if, for example, off-premise cloud platforms are included.

4.1.3. Existing Solutions

Control functionality is traditionally executed on PLCs close to the machinery. These PLCs typically require vendor-specific implementations and are often hard to upgrade and update, which makes such control processes inflexible and difficult to manage. Moving computations to more freely programmable devices thus has the potential of significantly improving the flexibility. In this context, directly moving control functionality to (central) cloud environments is generally possible, yet only feasible if latency constraints are lenient.

Early approaches such as [RÜTH] and [VESTIN] have already shown the general applicability of leveraging COIN for in-network control.

4.1.4. Opportunities

- Performing simple control logic on PNDs and/or in COIN execution environments can bring the controlled system and the controller closer together, possibly satisfying the tight latency requirements.

- Creating a coupled control that is exercised via

  i. simplified approximations of more complex control algorithms deployed in COIN execution environments, and

  ii. more complex overall control schemes deployed in the cloud

  can allow for quicker, yet more inaccurate responses from within the network while still providing for sufficient control accuracy at higher latencies from afar.

4.1.5. Research Questions

- RQ 4.1.1: How to derive simplified versions of the global (control) function?
- RQ 4.1.2: How to account for the limited computational precision of PNDs that typically only allow for integer precision computation for enabling high processing rates, while floating-point precision is needed by most control algorithms (cf. [KUNZE-APPLICABILITY])?
- RQ 4.1.3: How to find suitable tradeoffs regarding simplicity of the control function ("accuracy of the control") and implementation complexity ("implementability")?
- RQ 4.1.4: How to (dynamically) distribute simplified versions of the global (control) function among COIN execution environments?
- RQ 4.1.5: How to (dynamically) compose or recompose the distributed control functions?
- RQ 4.1.6: Can there be different control levels, e.g., "quite inaccurate & very low latency" (PNDs, deep in the network), "more accurate & higher latency" (more powerful COIN execution environments, farther away), "very accurate & very high latency" (cloud environments, far away)?
- RQ 4.1.7: Who decides which control instance is executed and which information can be used for this decision?
- RQ 4.1.8: How do the different control instances interact and how can we define their hierarchy?

4.1.6. Additional Desirable Capabilities

In addition to the capabilities driven by the research questions above, there are a number of other features that approaches deploying control functionality in COIN execution environments could benefit from. For example, having an explicit interaction between the COIN execution environments and the global controller would ensure that it is always clear which entity is emitting which signals. In this context, it is also important that actions of COIN execution environments are overridable by the global controller such that the global controller has the final say in the process behavior. Finally, by accommodating the general characteristics of control approaches, functions in COIN execution environments should ideally expose reliable information on the predicted delay and must expose reliable information on the predicted accuracy to the global control such that these aspects can be accommodated in the overall control.

4.2. Large-Volume Applications

4.2.1. Description

In modern industrial networks, processes and machines are extensively monitored by distributed sensors with a large spectrum of capabilities, ranging from simple binary (e.g., light barriers) to sophisticated sensors with varying degrees of resolution. Sensors further serve different purposes, as some are used for time-critical process control, while others represent redundant fallback platforms. Overall, there is a high level of heterogeneity, which makes managing the sensor output a challenging task.

Depending on the deployed sensors and the complexity of the observed system, the resulting overall data volume can easily be in the range of several Gbit/s [GLEBKE]. These volumes are often already difficult to handle in local environments, and it becomes even more challenging when off-premise clouds are used for managing the data. While large networking companies can simply upgrade their infrastructure to accommodate the accruing data volumes, most industrial companies operate on tight infrastructure budgets such that frequently upgrading is not always feasible or possible. Hence, a major challenge is to devise a methodology that is able to handle such amounts of data efficiently and flexibly without relying on recurring infrastructure upgrades.

Data filtering and preprocessing, similar to the considerations in Section 3.2, can be building blocks for new solutions in this space. Such solutions, however, might also have to address the added challenge of business data leaving the premises and control of the company. As this data could include sensitive information or valuable business secrets, additional security measures have to be taken. Yet, typical security measures such as encrypting the data make filtering or preprocessing approaches hardly applicable as they typically work on unencrypted data. Consequently, incorporating security into these approaches, either by adding functionality for handling encrypted data or devising general security measures, is an additional auspicious field for research.

4.2.2. Characterization

In essence, the described monitoring systems consist of sensors that produce large volumes of monitoring data. This data is then transmitted to additional components that provide data processing and analysis capabilities or simply store the data in large data silos.

As sensors are often set up redundantly, parts of the collected data might also be redundant. Moreover, sensors are often hard to configure or not configurable at all, which is why their resolution or sampling frequency is often larger than required. Consequently, it is likely that more data is transmitted than is needed or desired, prompting the deployment of filtering techniques. For example, COIN programs deployed in the on-premise network could filter out redundant or undesired data before it leaves the premise using simple traffic filters, thus reducing the required (upload) bandwidths. The available sensor data could be scaled down using standard statistical sampling, packet-based sub-sampling (i.e., only forwarding every n-th packet), or using filtering as long as the sensor value is in an uninteresting range while forwarding with a higher resolution once the sensor value range becomes interesting (cf.

[KUNZE-SIGNAL]). While the former variants are oblivious to the semantics of the sensor data, the latter variant requires an understanding of the current sensor levels. In any case, it is important that end hosts are informed about the filtering so that they can distinguish between data loss and data filtered out on purpose.

In practice, the collected data is further processed using various forms of computation. Some of them are very complex or need the complete sensor data during the computation, but there are also simpler operations that can already be done on subsets of the overall dataset or earlier on the communication path as soon as all data is available. One example is finding the maximum of all sensor values, which can either be done iteratively at each intermediate hop or at the first hop where all data is available. Using expert knowledge about the exact computation steps and the concrete transmission path of the sensor data, simple computation steps can thus be deployed in the on-premise network, again reducing the overall data volume.

4.2.3. Existing Solutions

Current approaches for handling such large amounts of information typically build upon stream processing frameworks such as Apache Flink. These solutions allow for handling large-volume applications and map the compute functionality to performant server machines or distributed compute platforms. Augmenting the existing capabilities, COIN offers a new dimension of platforms for such processing frameworks.

4.2.4. Opportunities

- (Stream) processing frameworks can become more flexible by introducing COIN execution environments as additional deployment targets.
- (Semantic) packet filtering based on packet header and payload, as well as multipacket information can (drastically) reduce the data volume, possibly even without losing any important information.
- (Semantic) data preprocessing and processing (e.g., in the form of computations across multiple packets and potentially leveraging packet payload) can also reduce the data volume without losing any important information.

4.2.5. Research Questions

Some of the following research questions are also relevant in the context of general stream processing systems.

- RQ 4.2.1: How can the overall data processing pipeline be divided into individual processing steps that could then be deployed as COIN functionality?
- RQ 4.2.2: How to design COIN programs for (semantic) packet filtering and which filtering criteria make sense?
- RQ 4.2.3: Which kinds of COIN programs can be leveraged for (pre)processing steps and what complexity can they have?
- RQ 4.2.4: How to distribute and coordinate COIN programs?
- RQ 4.2.5: How to dynamically reconfigure and recompose COIN programs?
- RQ 4.2.6: How to incorporate the (pre)processing and filtering steps into the overall system?

- RQ 4.2.7: How can changes to the data by COIN programs be signaled to the end hosts?

4.2.6. Additional Desirable Capabilities

In addition to the capabilities driven by the research questions above, there are a number of other features that such large-volume applications could benefit from. In particular, conforming to standard application-level syntax and semantics likely simplifies embedding filters and preprocessors into the overall system. If these filters and preprocessors also leverage packet header and payload information for their operation, this could further improve the performance of any approach developed based on the above research questions.

4.3. Industrial Safety

4.3.1. Description

Despite an increasing automation in production processes, human workers are still often necessary. Consequently, safety measures have a high priority to ensure that no human life is endangered. In traditional factories, the regions of contact between humans and machines are well defined and interactions are simple. Simple safety measures like emergency switches at the working positions are enough to provide a good level of safety.

Modern factories are characterized by increasingly dynamic and complex environments with new interaction scenarios between humans and robots. Robots can directly assist humans, perform tasks autonomously, or even freely move around on the shop floor. Hence, the intersect between the human working area and the robots grows, and it is harder for human workers to fully observe the complete environment. Additional safety measures are essential to prevent accidents and support humans in observing the environment.

4.3.2. Characterization

Industrial safety measures are typically hardware solutions because they have to pass rigorous testing before they are certified and deployment ready. Standard measures include safety switches and light barriers. Additionally, the working area can be explicitly divided into "contact" and "safe" areas, indicating when workers have to watch out for interactions with machinery. For example, markings on the factory floor can show the areas where robots move or indicate their maximum physical reach.

These measures are static solutions, potentially relying on specialized hardware, and are challenged by the increased dynamics of modern factories where the factory configuration can be changed on demand or where all entities are freely moving around. Software solutions offer higher flexibility as they can dynamically respect new information gathered by the sensor systems, but in most cases they cannot give guaranteed safety. COIN systems could leverage the increased availability of sensor data and the detailed monitoring of the factories to enable additional safety measures with shorter response times and higher guarantees. Different safety indicators within the production hall could be combined within the network so that PNDs can give early responses if a potential safety breach is detected. For example, the positions of human workers and robots could be tracked, and robots could be stopped when they get too close to a human in a non-working area or if a human enters a defined safety zone. More advanced concepts could also include image data or combine arbitrary sensor data. Finally, the increasing

softwarization of industrial processes can also lead to new problems, e.g., if software bugs cause unintended movements of robots. Here, COIN systems could independently double check issued commands to void unsafe commands.

4.3.3. Existing Solutions

Due to the importance of safety, there is a wide range of software-based approaches aiming at enhancing security. One example are tag-based systems (e.g., using RFID), where drivers of forklifts can be warned if pedestrian workers carrying tags are nearby. Such solutions, however, require setting up an additional system and do not leverage existing sensor data.

4.3.4. Opportunities

- Executing safety-critical COIN functions on PNDs could allow for early emergency reactions based on diverse sensor feedback with low latencies.
- COIN software could provide independent on-path surveillance of control software-initiated actions to block unsafe commands.

4.3.5. Research Questions

- RQ 4.3.1: Which additional safety measures can be provided and do they actually improve safety?
- RQ 4.3.2: Which sensor information can be combined and how?
- RQ 4.3.3: How can COIN-based safety measures be integrated with existing safety measures without degrading safety?
- RQ 4.3.4: How can COIN software validate control software-initiated commands to prevent unsafe operations?

5. Improving Existing COIN Capabilities

5.1. Content Delivery Networks

5.1.1. Description

Delivery of content to end users often relies on Content Delivery Networks (CDNs). CDNs store said content closer to end users for latency-reduced delivery as well as to reduce load on origin servers. For this, they often utilize DNS-based indirection to serve the request on behalf of the origin server. Both of these objectives are within scope to be addressed by COIN methods and solutions.

5.1.2. Characterization

From the perspective of this draft, a CDN can be interpreted as a (network service level) set of (COIN) programs. These programs implement a distributed logic for first distributing content from the origin server to the CDN ingress and then further to the CDN replication points, which ultimately serve the user-facing content requests.

5.1.3. Existing Solutions

CDN technologies have been well described and deployed in the existing Internet. Core technologies like Global Server Load Balancing (GSLB) [GSLB] and Anycast server solutions are used to deal with the required indirection of a content request (usually in the form of an HTTP request) to the most suitable local CDN server. Content is replicated from seeding servers, which serve as injection points for content from content owners/producers, to the actual CDN servers, which will eventually serve the user's request. The replication architecture and mechanisms themselves differ from one (CDN) provider to another, and often utilize private peering or network arrangements in order to distribute the content internationally and regionally.

Studies such as those in [FCDN] have shown that content distribution at the level of named content, utilizing efficient (e.g., Layer 2 (L2)) multicast for replication towards edge CDN nodes, can significantly increase the overall network and server efficiency. It also reduces indirection latency for content retrieval as well as required edge storage capacity by benefiting from the increased network efficiency to renew edge content more quickly against changing demand. Works such as those in [SILKROAD] utilize Application-Specific Integrated Circuits (ASICs) to replace server-based load balancing with significant cost reductions, thus showcasing the potential for in-network CN operations.

5.1.4. Opportunities

- Supporting service-level routing of requests (such as service routing in [APPCENTRES]) to specific (COIN) program instances may improve on end-user experience in retrieving faster (and possibly better quality) content.
- COIN instances may also be utilized to integrate service-related telemetry information to support the selection of the final service instance destination from a pool of possible choices.
- Supporting the selection of a service destination from a set of possible (e.g., virtualized, distributed) choices, e.g., through constraint-based routing decisions (see [APPCENTRES]) in (COIN) program instances to improve the overall end-user experience by selecting a "more suitable" service destination over another, e.g., avoiding/reducing overload situations in specific service destinations.
- Supporting L2 capabilities for multicast (compute interconnection and collective communication in [APPCENTRES]), e.g., through in-network/switch-based replication decisions (and their optimizations) based on dynamic group membership information, may reduce the network utilization and therefore increase the overall system efficiency.

5.1.5. Research Questions

In addition to the research questions in Section 3.1.5:

- RQ 5.1.1: How to utilize L2 multicast to improve on CDN designs? How to utilize COIN capabilities in those designs, such as through on-path optimizations for fanouts?
- RQ 5.1.2: What forwarding methods may support the required multicast capabilities (see [FCDN]) and how could programmable COIN forwarding elements support those methods (e.g., extending current SDN capabilities)?

- RQ 5.1.3: What are the constraints, reflecting both compute and network capabilities, that may support joint optimization of routing and computing? How could intermediary (COIN) program instances support, for example, the aggregation of those constraints to reduce overall telemetry network traffic?
- RQ 5.1.4: Could traffic steering be performed on the data path and per service request (e.g., through (COIN) program instances that perform novel routing request lookup methods)? If so, what would be performance improvements?
- RQ 5.1.5: How could storage be traded off against frequent, multicast-based replication (see [FCDN])? Could intermediary/in-network (COIN) elements support the storage beyond current endpoint-based methods?
- RQ 5.1.6: What scalability limits exist for L2 multicast capabilities? How to overcome them, e.g., through (COIN) program instances serving as stateful subtree aggregators to reduce the needed identifier space (e.g., for bit-based forwarding)?

5.2. Compute-Fabric-as-a-Service (CFaaS)

5.2.1. Description

We interpret connected compute resources as operating at a suitable layer, such as Ethernet, InfiniBand, but also at Layer 3 (L3), to allow for the exchange of suitable invocation methods, such as those exposed through verb-based or socket-based APIs. The specific invocations here are subject to the applications running over a selected pool of such connected compute resources.

Providing such a pool of connected compute resources (e.g., in regional or edge data centers, base stations, and even end-user devices) opens up the opportunity for infrastructure providers to offer CFaaS-like offerings to application providers, leaving the choice of the appropriate invocation method to the app and service provider. Through this, the compute resources can be utilized to execute the desired (COIN) programs of which the application is composed, while utilizing the interconnection between those compute resources to do so in a distributed manner.

5.2.2. Characterization

We foresee those CFaaS offerings to be tenant-specific, with a tenant here defined as the provider of at least one application. For this, we foresee an interaction between the CFaaS provider and tenant to dynamically select the appropriate resources to define the demand side of the fabric. Conversely, we also foresee the supply side of the fabric to be highly dynamic, with resources being offered to the fabric through, for example, user-provided resources (whose supply might depend on highly context-specific supply policies) or infrastructure resources of intermittent availability such as those provided through road-side infrastructure in vehicular scenarios.

The resulting dynamic demand-supply matching establishes a dynamic nature of the compute fabric that in turn requires trust relationships to be built dynamically between the resource provider(s) and the CFaaS provider. This also requires the communication resources to be dynamically adjusted to suitably interconnect all resources into the (tenant-specific) fabric exposed as CFaaS.

5.2.3. Existing Solutions

There exist a number of technologies to build non-local (wide area) L2 as well as L3 networks, which in turn allows for connecting compute resources for a distributed computational task. For instance, 5G-LAN [SA2-5GLAN] specifies a cellular L2 bearer for interconnecting L2 resources within a single cellular operator. The work in [ICN-5GLAN] outlines using a path-based forwarding solution over 5G-LAN as well as SDN-based LAN connectivity together with an Information-Centric Network (ICN) based naming of IP and HTTP-level resources. This is done in order to achieve computational interconnections, including scenarios such as those outlined in Section 3.1. L2 network virtualization (see [L2Virt]) is one of the methods used for realizing so-called "cloud-native" applications for applications developed with "physical" networks in mind, thus forming an interconnected compute and storage fabric.

5.2.4. Opportunities

- Supporting service-level routing of compute resource requests (such as service routing in [APPCENTRES]) may allow for utilizing the wealth of compute resources in the overall CFaaS fabric for execution of distributed applications, where the distributed constituents of those applications are realized as (COIN) programs and executed within a COIN system as (COIN) program instances.
- Supporting the constraint-based selection of a specific (COIN) program instance over others (such as constraint-based routing in [APPCENTRES]) will allow for optimizing both the CFaaS provider constraints as well as tenant-specific constraints.
- Supporting L2 and L3 capabilities for multicast (such as compute interconnection and collective communication in [APPCENTRES]) will allow for decreasing both network utilization but also possible compute utilization (due to avoiding unicast replication at those compute endpoints), thereby decreasing total cost of ownership for the CFaaS offering.
- Supporting the enforcement of trust relationships and isolation policies through intermediary (COIN) program instances, e.g., enforcing specific traffic shares or strict isolation of traffic through differentiated queueing.

5.2.5. Research Questions

In addition to the research questions in Section 3.1.5:

- RQ 5.2.1: How to convey tenant-specific requirements for the creation of the CFaaS fabric?
- RQ 5.2.2: How to dynamically integrate resources into the compute fabric being utilized for the app execution (those resources include, but are not limited to, end-user provided resources), particularly when driven by tenant-level requirements and changing service-specific constraints? How can those resources be exposed through possible (COIN) execution environments?
- RQ 5.2.3: How to utilize COIN capabilities to aid the availability and accountability of resources, i.e., what may be (COIN) programs for a CFaaS environment that in turn would utilize the distributed execution capability of a COIN system?
- RQ 5.2.4: How to utilize COIN capabilities to enforce traffic and isolation policies for establishing trust between tenant and CFaaS provider in an assured operation?

- RQ 5.2.5: How to optimize the interconnection of compute resources, including those dynamically added and removed during the provisioning of the tenant-specific compute fabric?

5.3. Virtual Networks Programming

5.3.1. Description

The term "virtual network programming" is proposed to describe mechanisms by which tenants deploy and operate COIN programs in their virtual network. Such COIN programs can be, for example, P4 programs, OpenFlow rules, or higher layer programs. This feature can enable other use cases described in this draft to be deployed using virtual network services, over underlying networks such as data centers, mobile networks, or other fixed or wireless networks.

For example, COIN programs could perform the following on a tenant's virtual network:

- Allow or block flows, and request rules from an SDN controller for each new flow, or for flows to or from specific hosts that need enhanced security.
- Forward a copy of some flows towards a node for storage and analysis.
- Update metrics based on specific sources/destinations or protocols, for detailed analytics.
- Associate traffic between specific endpoints, using specific protocols, or originated from a given application, to a given slice, while other traffic uses a default slice.
- Experiment with a new routing protocol (e.g., ICN), using a P4 implementation of a router for this protocol.

5.3.2. Characterization

To provide a concrete example of virtual COIN programming, we consider a use case using a 5G underlying network, the 5GLAN virtualization technology, and the P4 programming language and environment. As an assumption in this use case, some mobile network equipment (e.g., User Plane Functions (UPFs)) and devices (e.g., mobile phones or residential gateways) include a network switch functionality that is used as a PND.

Section 5.1 of [ICN-5GC] provides a description of the 5G network functions and interfaces relevant to 5GLAN, which are otherwise specified in [TS23.501] and [TS23.502]. From the 5GLAN service customer/tenant standpoint, the 5G network operates as a switch.

In the use case depicted in Figure 3, the tenant operates a network including a 5GLAN network segment (seen as a single logical switch), as well as fixed segments. The mobile devices (or User Equipment nodes) UE1, UE2, UE3, and UE4 are in the same 5GLAN, as well as Device1 and Device2 (through UE4). This scenario can take place in a plant or enterprise network, using a 5G non-public network, for example. The tenant uses P4 programs to determine the operation of both the fixed and 5GLAN switches. The tenant provisions a 5GLAN P4 program into the mobile network and can also operate a controller.

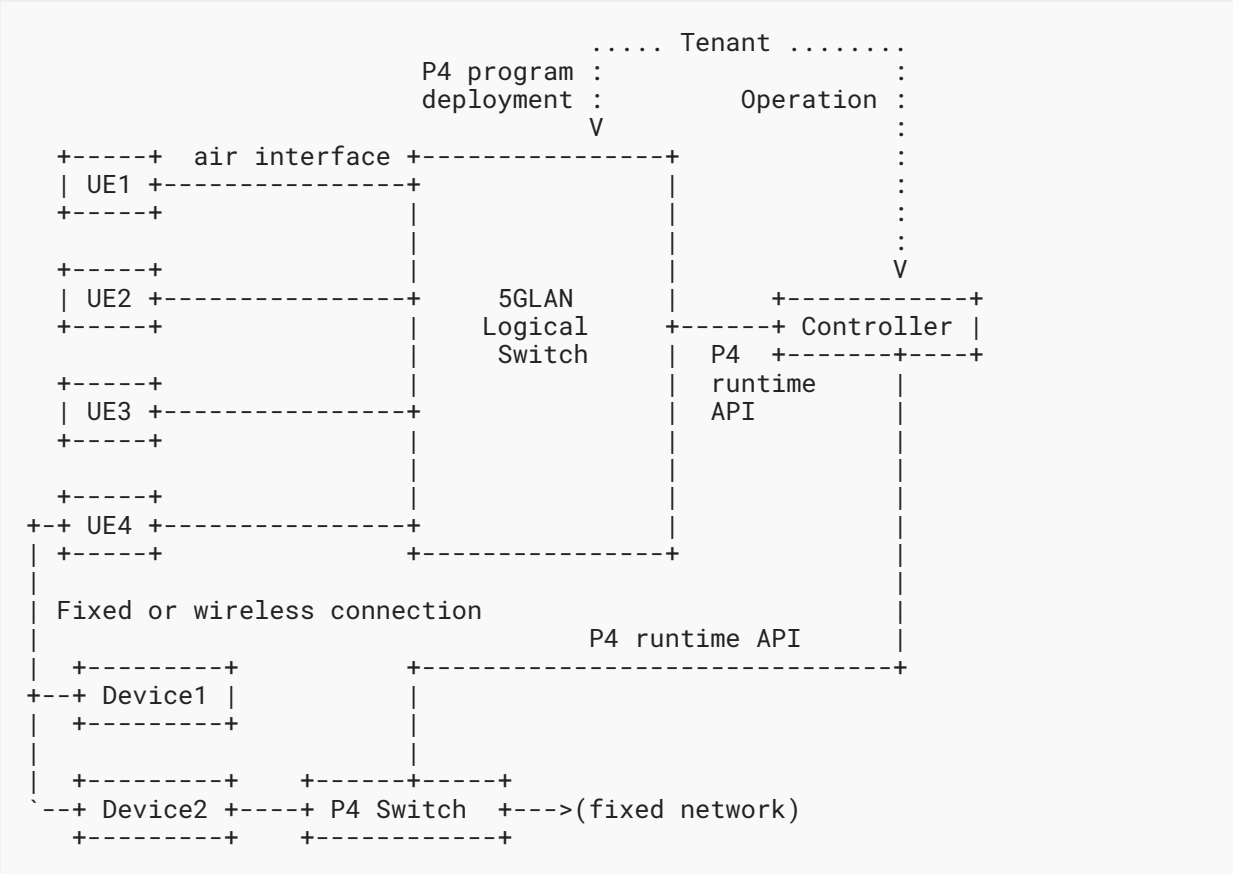

Figure 3: 5G Virtual Network Programming Overview

5.3.3. Existing Solutions

Research has been conducted, for example by [Stoyanov], to enable P4 network programming of individual virtual switches. To our knowledge, no complete solution has been developed for deploying virtual COIN programs over mobile or data center networks.

5.3.4. Opportunities

Virtual network programming by tenants could bring benefits such as:

- A unified programming model, which can facilitate porting COIN programs between data centers, 5G networks, and other fixed and wireless networks, as well as sharing controller, code and expertise.
- Increasing the level of customization available to customers/tenants of mobile networks or data centers compared to typical configuration capabilities. For example, 5G network evolution points to an ever-increasing specialization and customization of private mobile networks, which could be handled by tenants using a programming model similar to P4.

- Using network programs to influence underlying network services (e.g., requesting specific QoS for some flows in 5G or data centers) to increase the level of in-depth customization available to tenants.

5.3.5. Research Questions

- RQ 5.3.1: Underlying network awareness

  A virtual COIN program can both influence, and be influenced by, the underling network. Research challenges include defining methods to distribute COIN programs, including in a mobile network context, based on network awareness, since some information and actions may be available on some nodes but not on others.

- RQ 5.3.2: Splitting/distribution

  A virtual COIN program may need to be deployed across multiple computing nodes, leading to research questions around instance placement and distribution. For example, program logic should be applied exactly once or at least once per packet (or at least once for idempotent operations), while allowing an optimal forwarding path by the underlying network. Research challenges include defining manual (by the programmer) or automatic methods to distribute COIN programs that use a low or minimal amount of resources. Distributed P4 programs are studied in [P4-SPLIT] and [Sultana] (based on capability 5.3.2).

- RQ 5.3.3: Multi-tenancy support

  A COIN system supporting virtualization should enable tenants to deploy COIN programs onto their virtual networks, in such a way that multiple virtual COIN program instances can run on the same compute node. While mechanisms were proposed for P4 multi-tenancy in a switch [Stoyanov], research questions remain about isolation between tenants and fair repartition of resources (based on capability 5.3.3).

- RQ 5.3.4: Security

  How can tenants and underlying networks be protected against security risks, including overuse or misuse of network resources, injection of traffic, or access to unauthorized traffic?

- RQ 5.3.5: Higher layer processing

  Can a virtual network model facilitate the deployment of COIN programs acting on application-layer data? This is an open question, since this section focuses on packet/flow processing.

6. Enabling New COIN Capabilities

6.1. Distributed AI Training

6.1.1. Description

There is a growing range of use cases demanding the realization of AI training capabilities among distributed endpoints. One such use case is to distribute large-scale model training across more than one data center (e.g., when facing energy issues at a single site or when simply

reaching the scale of training capabilities at one site, thus wanting to complement training with the capabilities of another or possibly many sites). From a COIN perspective, those capabilities may be realized as (COIN) programs and executed throughout a COIN system, including in PNDs.

6.1.2. Characterization

Some solutions may desire the localization of reasoning logic (e.g., for deriving attributes that better preserve privacy of the utilized raw input data). Quickly establishing (COIN) program instances in nearby compute resources, including PNDs, may even satisfy such localization demands on the fly (e.g., when a particular use is being realized, then terminated after a given time).

Individual training "sites" may not be a data center, but may instead consist of powerful, yet stand-along devices that federate computing power towards training a model, captured as "federated training" and provided through platforms such as [FLOWER]. Use cases here may be that of distributed training on (user) image data, the training over federated social media sites [MASTODON], or others.

Apart from the distribution of compute power, the distribution of data may be a driver for distributed AI training use cases, such as in the Mastodon federated social media sites [MASTODON] or training over locally governed patient data or others.

6.1.3. Existing Solutions

Reasoning frameworks, such as TensorFlow, may be utilized for the realization of the (distributed) AI training logic, building on remote service invocation through protocols such as gRPC [GRPC] or the Message Passing Interface (MPI) [MPI] with the intention of providing an on-chip Neural Processor Unit (NPU) like abstraction to the AI framework.

A number of activities on distributed AI training exist in the area of developing the 5th and 6th generation mobile network, with various activities in the 3GPP Standards Development Organization (SDO) as well as use cases developed for the ETSI MEC initiative mentioned in previous use cases.

6.1.4. Opportunities

- Supporting service-level routing of training requests (such as service routing in [APPCENTRES]), with AI services being exposed to the network, where (COIN) program instances may support the selection of the most suitable service instance based on control plane information, e.g., on AI worker compute capabilities, being distributed across (COIN) program instances.
- Supporting the collective communication primitives, such as all-to-all, scatter-gather, utilized by the (distributed) AI workers to increase the overall network efficiency, e.g., through avoiding endpoint-based replication or even directly performing, e.g., reduce, collective primitive operations in (COIN) program instances placed in topologically advantageous places.

- Supporting collective communication between multiple instances of AI services (i.e., (COIN) program instances) may positively impact network but also compute utilization by moving from unicast replication to network-assisted multicast operation.

6.1.5. Research Questions

In addition to the research questions in Section 3.1.5:

- RQ 6.1.1: What are the communication patterns that may be supported by collective communication solutions, where those solutions directly utilize (COIN) program instance capabilities within the network (e.g., reduce in a central (COIN) program instance)?
- RQ 6.1.2: How to achieve scalable collective communication primitives with rapidly changing receiver sets (e.g., where training workers may be dynamically selected based on energy efficiency constraints [GREENAI])?
- RQ 6.1.3: What COIN capabilities may support the collective communication patterns found in distributed AI problems?
- RQ 6.1.4: How to support AI-specific invocation protocols, such as MPI or Remote Direct Memory Access (RDMA)?
- RQ 6.1.5: What are the constraints for placing (AI) execution logic in the form of (COIN) programs in certain logical execution points (and their associated physical locations), including PNDs, and how to signal and act upon them?

7. Preliminary Categorization of the Research Questions

This section describes a preliminary categorization of the research questions illustrated in Figure 4. A more comprehensive analysis has been initiated by members of the COINRG community in [USE-CASE-AN] but has not been completed at the time of writing this memo.

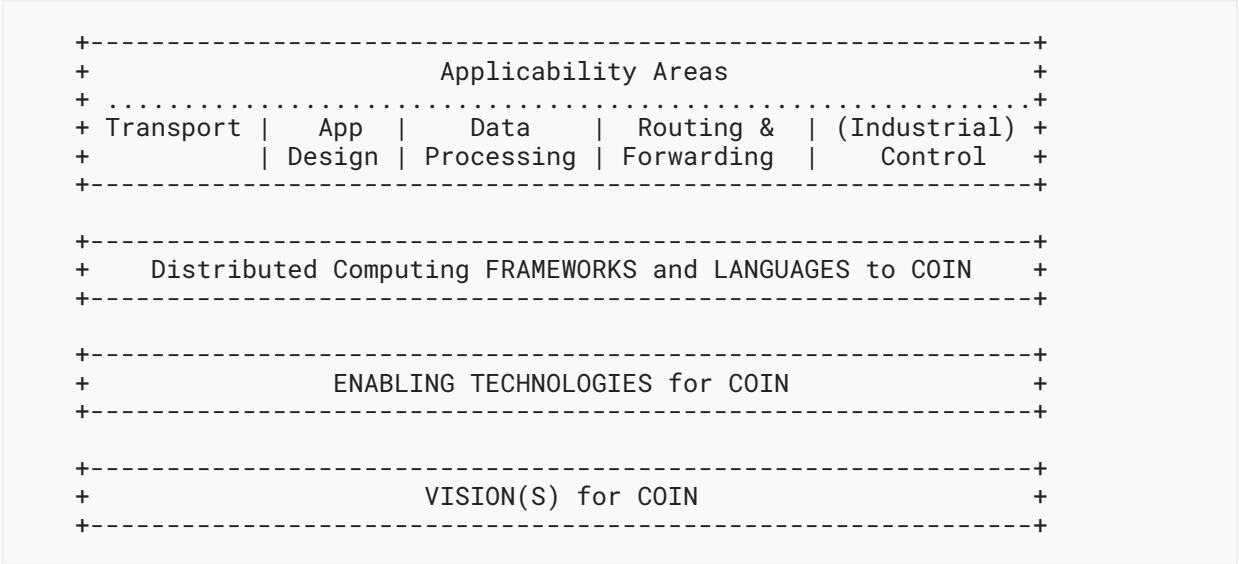

Figure 4: Research Questions Categories

The "VISION(S) for COIN" category is about defining and shaping the exact scope of COIN. In contrast to the "ENABLING TECHNOLOGIES" category, these research questions look at the problem from a more philosophical perspective. In particular, the questions center around where to perform computations, which tasks are suitable for COIN, for which tasks COIN is suitable, and which forms of deploying COIN might be desirable. This category includes the research questions 3.1.8, 3.2.1, 3.3.5, 3.3.6, 3.3.7, 5.3.3, 6.1.1, and 6.1.3.

The "ENABLING TECHNOLOGIES for COIN" category digs into what technologies are needed to enable COIN, which of the existing technologies can be reused for COIN, and what might be needed to make the "VISION(S) for COIN" a reality. In contrast to the "VISION(S) for COIN", these research questions look at the problem from a practical perspective (e.g., by considering how COIN can be incorporated in existing systems or how the interoperability of COIN execution environments can be enhanced). This category includes the research questions 3.1.7, 3.1.8, 3.2.3, 4.2.7, 5.1.1, 5.1.2, 5.1.6, 5.3.1, 6.1.2, and 6.1.3.

The "Distributed Computing FRAMEWORKS and LANGUAGES to COIN" category focuses on how COIN programs can be deployed and orchestrated. Central questions arise regarding the composition of COIN programs, the placement of COIN functions, the (dynamic) operation and integration of COIN systems as well as additional COIN system properties. Notably, COIN diversifies general distributed computing platforms such that many COIN-related research questions could also apply to general distributed computing frameworks. This category includes the research questions 3.1.1, 3.2.4, 3.3.1, 3.3.2, 3.3.3, 3.3.5, 4.1.1, 4.1.4, 4.1.5, 4.1.8, 4.2.1, 4.2.4, 4.2.5, 4.2.6, 4.3.3, 5.2.1, 5.2.2, 5.2.3, 5.2.5, 5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.3.5, and 6.1.5.

In addition to these core categories, there are use case specific research questions that are heavily influenced by the specific constraints and objectives of the respective use cases. This "Applicability Areas" category can be further refined into the following subgroups:

- The "Transport" subgroup addresses the need to adapt transport protocols to handle dynamic deployment locations effectively. This subgroup includes the research question 3.1.2.
- The "App Design" subgroup relates to the design principles and considerations when developing COIN applications. This subgroup includes the research questions 4.1.2, 4.1.3, 4.1.7, 4.2.6, 5.1.1, 5.1.3, and 5.1.5.
- The "Data Processing" subgroup relates to the handling, storage, analysis, and processing of data in COIN environments. This subgroup includes the research questions 3.2.4, 3.2.6, 4.2.2, 4.2.3, and 4.3.2.
- The "Routing & Forwarding" subgroup explores efficient routing and forwarding mechanisms in COIN, considering factors such as network topology, congestion control, and quality of service. This subgroup includes the research questions 3.1.2, 3.1.3, 3.1.4, 3.1.5, 3.1.6, 3.2.6, 5.1.2, 5.1.3, 5.1.4, and 6.1.4.
- The "(Industrial) Control" subgroup relates to industrial control systems, addressing issues like real-time control, automation, and fault tolerance. This subgroup includes the research questions 3.1.9, 3.2.5, 3.3.1, 3.3.4, 4.1.1, 4.1.6, 4.1.8, 4.2.3, 4.3.1, and 4.3.4.

8. Security Considerations

COIN systems, like any other system using "middleboxes", can have different security and privacy implications that strongly depend on the used platforms, the provided functionality, and the deployment domain, with most if not all considerations for general middleboxes also applying to COIN systems.

One critical aspect for early COIN systems is the use of early generation PNDs, many of which do not have cryptography support and only have limited computational capabilities. Hence, PND-based COIN systems typically work on unencrypted data and often customize packet payload, while concepts such as homomorphic encryption could serve as workarounds, allowing PNDs to perform simple operations on the encrypted data without having access to it. All these approaches introduce the same or very similar security implications as any middlebox operating on unencrypted traffic or having access to encryption: a middlebox can itself have malicious intentions (e.g., because it got compromised or the deployment of functionality offers new attack vectors to outsiders).

However, similar to middlebox deployments, risks for privacy and data exposure have to be carefully considered in the context of the concrete deployment. For example, exposing data to an external operator for mobile application offloading leads to a significant privacy loss of the user in any case. In contrast, such privacy considerations are not as relevant for COIN systems where all involved entities are under the same control, such as in an industrial context. Here, exposed data and functionality can instead lead to stolen business secrets or the enabling of DoS attacks,

for example. Hence, even in fully controlled scenarios, COIN intermediaries, and middleboxes in general, are ideally operated in a least-privilege mode, where they have exactly those permissions to read and alter payload that are necessary to fulfill their purpose.

Research on granting middleboxes access to secured traffic is only in its infancy, and a variety of different approaches are proposed and analyzed [TLSSURVEY]. In a SplitTLS [SPLITTLS] deployment, for example, middleboxes have different incoming and outgoing TLS channels, such that they have full read and write access to all intercepted traffic. More restrictive approaches for deploying middleboxes rely on searchable encryption or zero-knowledge proofs to expose less data to intermediaries, but those only offer limited functionality. MADTLS [MADTLS] is tailored to the industrial domain and offers bit-level read and write access to intermediaries with low latency and bandwidth overhead, at the cost of more complex key management. Overall, different proposals offer different advantages and disadvantages that must be carefully considered in the context of concrete deployments. Further research could pave the way for a more unified and configurable solution that is easier to maintain and deploy.

Finally, COIN systems and other middlebox deployments can also lead to security risks even if the attack stems from an outsider without direct access to any devices. As such, metadata about the entailed processing (processing times or changes in incoming and outgoing data) can allow an attacker to extract valuable information about the process. Moreover, such deployments can become central entities that, if paralyzed (e.g., through extensive requests), can be responsible for large-scale outages. In particular, some deployments could be used to amplify DoS attacks. Similar to other middlebox deployments, these potential risks must be considered when deploying COIN functionality and may influence the selection of suitable security protocols.

Additional system-level security considerations may arise from regulatory requirements imposed on COIN systems overall, stemming from regulation regarding, lawful interception, data localization, or AI use, for example. These requirements may impact, for example, the manner in which (COIN) programs may be placed or executed in the overall system, who can invoke certain (COIN) programs in what PND or COIN device, and what type of (COIN) program can be run. These considerations will impact the design of the possible implementing protocols but also the policies that govern the execution of (COIN) programs.

9. IANA Considerations

This document has no IANA actions.

10. Conclusion

This document presents use cases gathered from several application domains that can and could profit from capabilities that are provided by in-network and, more generally, distributed compute platforms. We distinguish between use cases in which COIN may enable new experiences (Section 3), expose new features (Section 6), or improve on existing system capabilities (Section 5), and other use cases where COIN capabilities enable totally new applications, for example, in industrial networking (Section 4).

Beyond the mere description and characterization of those use cases, we identify opportunities arising from utilizing COIN capabilities and formulate corresponding research questions that may need to be addressed before being able to reap those opportunities.

We acknowledge that this work offers no comprehensive overview of possible use cases and is thus only a snapshot of what may be possible if COIN capabilities existed. In fact, the decomposition of many current client-server applications into node-by-node transit could identify other opportunities for adding computing to forwarding, notably in the supply chain, health care, intelligent cities and transportation, and even financial services (among others). The presented use cases are selected based on the expertise of the contributing community members at the time of writing and are intended to cover a diverse range, from immersive and interactive media, industrial networks, to AI with varying characteristics, thus, providing the basis for a thorough subsequent analysis.

11. Informative References

[APPCENTRES] Trossen, D., Sarathchandra, C., and M. Boniface, "In-Network Computing for App-Centric Micro-Services", Work in Progress, Internet-Draft, draft-sarathchandra-coin-appcentres-04, 26 January 2021, <https://datatracker.ietf.org/doc/html/draft-sarathchandra-coin-appcentres-04>.

[CompNet2021] Chen, M., Liu, W., Wang, T., Liu, A., and Z. Zeng, "Edge intelligence computing for mobile augmented reality with deep reinforcement learning approach", Computer Networks, vol. 195, p. 108186, DOI 10.1016/j.comnet.2021.108186, August 2021, <https://doi.org/10.1016/j.comnet.2021.108186>.

[eCAR] Jeon, J. and W. Woo, "eCAR: edge-assisted Collaborative Augmented Reality Framework", arXiv:2405.06872, DOI 10.48550/ARXIV.2405.06872, 2024, <https://doi.org/10.48550/ARXIV.2405.06872>.

[ETSI] ETSI, "Multi-access Edge Computing (MEC)", <https://www.etsi.org/technologies/multi-access-edge-computing>.

[FCDN] Al-Naday, M., Reed, M. J., Riihijarvi, J., Trossen, D., Thomos, N., and M. Al-Khalidi, "fCDN: A Flexible and Efficient CDN Infrastructure without DNS Redirection of Content Reflection", arXiv:1803.00876v2, <https://arxiv.org/pdf/1803.00876.pdf>.

[FLOWER] Flower Labs GmbH, "A Friendly Federated AI Framework", <https://flower.ai/>.

[GLEBKE] Glebke, R., Henze, M., Wehrle, K., Niemietz, P., Trauth, D., Mattfeld, P., and T. Bergs, "A Case for Integrated Data Processing in Large-Scale Cyber-Physical Systems", Proceedings of the Annual Hawaii International Conference on System Sciences, DOI 10.24251/hicss.2019.871, 2019, <https://doi.org/10.24251/hicss.2019.871>.

[GREENAI] Magoula, L., Koursioumpas, N., Thanopoulos, A., Panagea, T., Petropouleas, N., Gutierrez-Estevez, M., and R. Khalili, "A Safe Genetic Algorithm Approach for Energy Efficient Federated Learning in Wireless Communication Networks",

2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 1-6, DOI 10.1109/pimrc56721.2023.10293863, September 2023, <https://doi.org/10.1109/pimrc56721.2023.10293863>.

[GRPC] "High performance, open source universal RPC framework", <https://grpc.io/>.

[GSLB] Cloudflare, "What is global server load balancing (GSLB)?", <https://www.cloudflare.com/learning/cdn/glossary/global-server-load-balancing-gslb/>.

[ICN-5GC] Ravindran, R., Suthar, P., Trossen, D., Wang, C., and G. White, "Enabling ICN in 3GPP's 5G NextGen Core Architecture", Work in Progress, Internet-Draft, draft-ravi-icnrg-5gc-icn-04, 31 May 2019, <https://datatracker.ietf.org/doc/html/draft-ravi-icnrg-5gc-icn-04>.

[ICN-5GLAN] Trossen, D., Wang, C., Robitzsch, S., Reed, M., AL-Naday, M., and J. Riihijarvi, "IP-based Services over ICN in 5G LAN Environments", Work in Progress, Internet-Draft, draft-trossen-icnrg-ip-icn-5glan-00, 6 June 2019, <https://datatracker.ietf.org/doc/html/draft-trossen-icnrg-ip-icn-5glan-00>.

[KUNZE-APPLICABILITY] Kunze, I., Glebke, R., Scheiper, J., Bodenbenner, M., Schmitt, R., and K. Wehrle, "Investigating the Applicability of In-Network Computing to Industrial Scenarios", 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), pp. 334-340, DOI 10.1109/icps49255.2021.9468247, May 2021, <https://doi.org/10.1109/icps49255.2021.9468247>.

[KUNZE-SIGNAL] Kunze, I., Niemietz, P., Tirpitz, L., Glebke, R., Trauth, D., Bergs, T., and K. Wehrle, "Detecting Out-Of-Control Sensor Signals in Sheet Metal Forming using In-Network Computing", 2021 IEEE 30th International Symposium on Industrial Electronics (ISIE), pp. 1-6, DOI 10.1109/isie45552.2021.9576221, June 2021, <https://doi.org/10.1109/isie45552.2021.9576221>.

[L2Virt] Kreger-Stickles, L., "First principles: L2 network virtualization for lift and shift", Oracle Cloud Infrastructure Blog, 9 February 2021, <https://blogs.oracle.com/cloud-infrastructure/post/first-principles-l2-network-virtualization-for-lift-and-shift>.

[MADTLS] Wagner, E., Heye, D., Serror, M., Kunze, I., Wehrle, K., and M. Henze, "Madtls: Fine-grained Middlebox-aware End-to-end Security for Industrial Communication", arXiv:2312.09650, DOI 10.48550/ARXIV.2312.09650, 2023, <https://doi.org/10.48550/ARXIV.2312.09650>.

[MASTODON] Raman, A., Joglekar, S., Cristofaro, E., Sastry, N., and G. Tyson, "Challenges in the Decentralised Web: The Mastodon Case", Proceedings of the Internet Measurement Conference, pp. 217-229, DOI 10.1145/3355369.3355572, October 2019, <https://doi.org/10.1145/3355369.3355572>.

[Microservices] Richardson, C., "What are microservices?", <https://microservices.io/>.

[MPI] Vishnu, A., Siegel, C., and J. Daily, "Scaling Distributed Machine Learning with In-Network Aggregation", arXiv:1603.02339v2, August 2017, <https://arxiv.org/pdf/1603.02339.pdf>.

[Multi2020] Sobrinho, J. and M. Ferreira, "Routing on Multiple Optimality Criteria", Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication, pp. 211-225, DOI 10.1145/3387514.3405864, July 2020, <https://doi.org/10.1145/3387514.3405864>.

[NetworkedVR] Ruan, J. and D. Xie, "Networked VR: State of the Art, Solutions, and Challenges", Electronics, vol. 10, no. 2, p. 166, DOI 10.3390/electronics10020166, January 2021, <https://doi.org/10.3390/electronics10020166>.

[P4] Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., and D. Walker, "P4: programming protocol-independent packet processors", ACM SIGCOMM Computer Communication Review, vol. 44, no. 3, pp. 87-95, DOI 10.1145/2656877.2656890, July 2014, <https://doi.org/10.1145/2656877.2656890>.

[P4-SPLIT] Singh, H. and M. Montpetit, "Requirements for P4 Program Splitting for Heterogeneous Network Nodes", Work in Progress, Internet-Draft, draft-hsingh-coinrg-reqs-p4comp-03, 18 February 2021, <https://datatracker.ietf.org/doc/html/draft-hsingh-coinrg-reqs-p4comp-03>.

[RFC7272] van Brandenburg, R., Stokking, H., van Deventer, O., Boronat, F., Montagud, M., and K. Gross, "Inter-Destination Media Synchronization (IDMS) Using the RTP Control Protocol (RTCP)", RFC 7272, DOI 10.17487/RFC7272, June 2014, <https://www.rfc-editor.org/info/rfc7272>.

[RFC8039] Shpiner, A., Tse, R., Schelp, C., and T. Mizrahi, "Multipath Time Synchronization", RFC 8039, DOI 10.17487/RFC8039, December 2016, <https://www.rfc-editor.org/info/rfc8039>.

[RÜTH] Rüth, J., Glebke, R., Wehrle, K., Causevic, V., and S. Hirche, "Towards In-Network Industrial Feedback Control", Proceedings of the 2018 Morning Workshop on In-Network Computing, pp. 14-19, DOI 10.1145/3229591.3229592, August 2018, <https://doi.org/10.1145/3229591.3229592>.

[SA2-5GLAN] 3GPP-5glan, "SP-181129, Work Item Description, Vertical_LAN(SA2), 5GS Enhanced Support of Vertical and LAN Services", 3GPP , 2021, <http://www.3gpp.org/ftp/tsg_sa/TSG_SA/Docs/SP-181120.zip>.

[SarNet2021] Glebke, R., Trossen, D., Kunze, I., Lou, D., Ruth, J., Stoffers, M., and K. Wehrle, "Service-based Forwarding via Programmable Dataplanes", 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), pp. 1-8, DOI 10.1109/hpsr52026.2021.9481814, June 2021, <https://doi.org/10.1109/hpsr52026.2021.9481814>.

[SILKROAD] Miao, R., Zeng, H., Kim, C., Lee, J., and M. Yu, "SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs", Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pp. 15-28, DOI 10.1145/3098822.3098824, August 2017, <https://doi.org/10.1145/3098822.3098824>.

[SPLITTLS] Naylor, D., Schomp, K., Varvello, M., Leontiadis, I., Blackburn, J., Lopez, D., Papagiannaki, K., Rodriguez Rodriguez, P., and P. Steenkiste, "Multi-Context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS", ACM SIGCOMM Computer Communication Review, vol. 45, no. 4, pp. 199-212, DOI 10.1145/2829988.2787482, August 2015, <https://doi.org/10.1145/2829988.2787482>.

[Stoyanov] Stoyanov, R. and N. Zilberman, "MTPSA: Multi-Tenant Programmable Switches", Proceedings of the 3rd P4 Workshop in Europe, pp. 43-48, DOI 10.1145/3426744.3431329, December 2020, <https://eng.ox.ac.uk/media/6354/stoyanov2020mtpsa.pdf>.

[Sultana] Sultana, N., Sonchack, J., Giesen, H., Pedisich, I., Han, Z., Shyamkumar, N., Burad, S., DeHon, A., and B. T. Loo, "Flightplan: Dataplane Disaggregation and Placement for P4 Programs", <https://flightplan.cis.upenn.edu/flightplan.pdf>.

[TLSSURVEY] de Carné de Carnavalet, X. and P. van Oorschot, "A Survey and Analysis of TLS Interception Mechanisms and Motivations: Exploring how end-to-end TLS is made 'end-to-me' for web traffic", ACM Computing Surveys, vol. 55, no. 13s, pp. 1-40, DOI 10.1145/3580522, July 2023, <https://doi.org/10.1145/3580522>.

[TOSCA] Rutkowski, M., Ed., Lauwers, C., Ed., Noshpitz, C., Ed., and C. Curescu, Ed., "TOSCA Simple Profile in YAML Version 1.3", OASIS Standard, February 2020, <https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/os/TOSCA-Simple-Profile-YAML-v1.3-os.pdf>.

[TS23.501] 3GPP, "System Architecture for the 5G System; Stage 2 (Rel.17)", 3GPP TS 23.501, 2021, <https://www.3gpp.org/DynaReport/23501.htm>.

[TS23.502] 3GPP, "Procedures for the 5G System; Stage 2 (Rel.17)", 3GPP TS 23.502, 2021, <https://www.3gpp.org/DynaReport/23502.htm>.

[USE-CASE-AN] Kunze, I., Hong, J., Wehrle, K., Trossen, D., Montpetit, M., de Foy, X., Griffin, D., and M. Rio, "Use Case Analysis for Computing in the Network", Work in Progress, Internet-Draft, draft-irtf-coinrg-use-case-analysis-02, 4 December 2024, <https://datatracker.ietf.org/doc/html/draft-irtf-coinrg-use-case-analysis-02>.

[VESTIN] Vestin, J., Kassler, A., and J. Akerberg, "FastReact: In-Network Control and Caching for Industrial Control Networks using Programmable Data Planes", 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 219-226, DOI 10.1109/etfa.2018.8502456, September 2018, <https://doi.org/10.1109/etfa.2018.8502456>.

[WirelessNet2024] Jia, J., Yang, L., Chen, J., Ma, L., and X. Wang, "Online delay optimization for MEC and RIS-assisted wireless VR networks", Wireless Networks, vol. 30, no. 4, pp. 2939-2959, DOI 10.1007/s11276-024-03706-4, March 2024, <https://doi.org/10.1007/s11276-024-03706-4>.

Acknowledgements

The authors would like to thank Eric Wagner for providing text on the security considerations and Jungha Hong for her efforts in continuing the work on the use case analysis document that has largely sourced the preliminary categorization section of this document.

The authors would further like to thank Chathura Sarathchandra, David Oran, Phil Eardley, Stuart Card, Jeffrey He, Toerless Eckert, and Jon Crowcroft for reviewing earlier versions of the document, Colin Perkins for his IRTF chair review, and Jerome Francois for his thorough IRSG review.

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

Marie-Jose Montpetit
McGill University
680 Sherbrooke Street W.
Montreal H3A 3R1
Canada
Email: marie-jose.montpetit@mcgill.ca

Xavier de Foy
InterDigital Communications, LLC
1000 Sherbrooke West
Montreal H3A 3G4
Canada
Email: xavier.defoy@interdigital.com

David Griffin
University College London
Gower St
London
WC1E 6BT
United Kingdom
Email: d.griffin@ucl.ac.uk

Miguel Rio
University College London
Gower St
London
WC1E 6BT
United Kingdom
Email: miguel.rio@ucl.ac.uk