Workgroup: COINRG
Internet-Draft: draft-irtf-coinrg-use-cases-03
Published: 11 March 2023
Intended Status: Informational
Expires: 12 September 2023
Authors: I. Kunze, RWTH Aachen; K. Wehrle, RWTH Aachen; D. Trossen, Huawei
M. J. Montpetit, Concordia; X. de Foy, InterDigital Communications, LLC
D. Griffin, UCL; M. Rio, UCL

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, further identifying their essential requirements.

Status of This Memo

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 functionality is generally provided by the end-hosts while network nodes are 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, with the rise of new services, some of which are described in this document, there are more and more fields 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, may help to achieve the desired guarantees and behaviors as well as increase overall performance.

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 fields.

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 requirements for an evolving solution space of COIN capabilities is another objective of the use case descriptions. To achieve this, the following taxonomy is proposed to describe each of the use cases:

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

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

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

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

5. Research questions: Essential questions that are suitable for guiding research to achieve the identified opportunities.

6. Requirements: Description of requirements for any COIN capability solutions that may need development along the opportunities outlined in item 4; we limit requirements to those directly describing COIN capabilities, recognizing that any use case will realistically hold many additional requirements for its realization.

This document discusses these six aspects along a number of individual use cases. A companion document [USECASEANALYSIS] is tasked with performing a cross-use case analysis, i.e., summarizing the key research questions and identifying key requirements across all use cases.

2. Terminology

This document uses the terminology outlined in [TERMINOLOGY].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Providing New COIN Experiences

3.1. Mobile Application Offloading

3.1.1. Description

The scenario can be exemplified in an immersive gaming application, where a single user plays a game using a VR headset.
The headset hosts functions that "display" frames to the user, as well as the functions for VR content processing and frame rendering that also incorporate input data received from sensors in the VR headset.

Once this application is partitioned into constituent (COIN) programs and deployed throughout a COIN system, utilizing the 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 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) functions for a flexible composition and a distributed execution. In our example above, most functions of a mobile application can be categorized into any of three, "receiving", "processing", and "displaying" function groups.

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 see today as 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.

Figure 1 shows one realization of the above scenario, where a 'DPR app' is running on a mobile device (containing the partitioned Display(D), Process(P) and Receive(R) COIN programs) over an SDN network. 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 ('processing server').

```
                          +----------+ Processing Server
        Mobile            | +------+ |
     +---------+          | |  P   | |
     |   App   |          | +------+ |
     | +-----+ |          | +------+ |
     | |D|P|R| |          | |  SR  | |
     | +-----+ |          | +------+ |        Internet
     | +-----+ |          +----------+           /
     | |  SR | |              |                 /
     | +-----+ |            +----------+     +------+
     +---------+           /|SDN Switch|_____|Border|
               +-------+ /  +----------+     |  SR  |
               | 5GAN  |/        |           +------+
               +-------+         |
   +---------+                   |
   |+-------+|               +----------+
   ||Display||              /|SDN Switch|
   |+-------+|   +-------+ / +----------+
   |+-------+|  /|WIFI AP|/
   ||   D   || / +-------+     +--+
   |+-------+|/                |SR|
   |+-------+|                /+--+
   ||  SR   ||          +---------+
   |+-------+|          |Playstore|
   +---------+          | Server  |
       TV               +---------+
```

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/many-viewing scenarios, thereby realizing a service-level multicast capability towards more than one (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 micro-services as described in our use case and also do not allow for the dynamic selection and redirection of micro-service 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 few milliseconds for aligning with the framerate of the display micro-service.

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 micro-services 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.

The draft at [APPCENTRES] outlines a number of enabling technologies for the use case, some of which have been realized in an Android-based realization of the micro-services as a single application, which is capable to dynamically redirect traffic to other micro-service 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.

* The support for service-level routing of requests (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.

* The ability to identify service-level COIN elements will allow for routing service requests to those COIN elements, including PNDs, therefore possibly allowing for new COIN functionality to be included in the mobile application.

* The support for constraint-based selection of a specific (COIN) program instance over others (constraint-based routing in [APPCENTRES]) 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 with app-level packaging methods?

* RQ 3.1.2: How to reduce latencies involved in (COIN) program interactions where (COIN) program instance locations may change quickly?

* RQ 3.1.3: How to signal constraints used for routing requests towards (COIN) program instances in a scalable manner?

* RQ 3.1.4: How to identify (COIN) programs and program instances?

* RQ 3.1.5: How to identify a specific choice of (COIN) program instances over others?

* 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 at packet forwarding speed?

*RQ 3.1.8: What COIN capabilities may support the execution of (COIN) programs and their instances?

3.1.6. Requirements

*Req 3.1.1: Any COIN system MUST provide means for routing of service requests between resources in the distributed environment.

*Req 3.1.2: Any COIN system MUST provide means for identifying services exposed by (COIN) programs for directing service requests.

*Req 3.1.3: Any COIN system MUST provide means for identifying (COIN) program instances for directing (affinity) requests to a specific (COIN) program instance.

*Req 3.1.4: Any COIN system MUST provide means for dynamically choosing the best possible service sequence of one or more (COIN) programs for a given application experience, i.e., support for chaining (COIN) program executions.

*Req 3.1.5: Means for discovering suitable (COIN) programs SHOULD be provided.

*Req 3.1.6: Any COIN system MUST provide means for pinning the execution of a service of a specific (COIN) program to a specific resource, i.e., (COIN) program instance in the distributed environment.

*Req 3.1.7: Any COIN system SHOULD provide means for packaging micro-services for deployments in distributed networked computing environments.

*Req 3.1.8: The packaging MAY include any constraints regarding the deployment of (COIN) program instances in specific network locations or compute resources, including PNDs.

*Req 3.1.9: Such packaging SHOULD conform to existing application deployment models, such as mobile application packaging, TOSCA orchestration templates or tar balls or combinations thereof.

*Req 3.1.10: Any COIN system MUST provide means for real-time synchronization and consistency of distributed application states.

3.2. Extended Reality and Immersive Media

3.2.1. Description

Virtual Reality (VR), Augmented Reality (AR), and immersive media, now globally referred to as Extended Reality (XR) and the bases for the metaverse, are the drivers of a number of advances in interactive technologies. While initially associated with gaming and entertainment, metaverse 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 multi-party interactions under strict delay requirements.

Indeed, XR applications require real-time interactivity for immersive and increasingly mobile immersive applications with tactile and time-sensitive data. Because high bandwidth is needed for high resolution images and local rendering for 3D images and holograms, they are difficult to run over traditional networks which limits some of its potential benefits in the collaborative space. As a consequence, innovation is needed to unlock the full potential of the applications.

3.2.2. Characterization

XR experiences, especially those involving collaboration, are difficult to deliver with a client-server cloud-based solution as they require a combination of: stream synchronization, 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. Furthermore, when XR deals with personal information and protected content, XR application must also provide a secure environment and ensure user privacy. Additionally, the sheer amount of data needed for and generated by the 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 to find the optimal caching and processing solution and hopefully reduce the size of the data that needs transiting through the network. Other mechanims such as data filtering and reduction, functional distribution and partitioning are also needed to accommodate the low delay needs for the same applications.

Those characterisitics of XR makes it ideal to profit from some COIN capabilities to enable XR over networks. COIN can enable the distribution of the service components across different nodes on the path from content source to rendering destination. For example, data filtering, image rendering, and video processing leveraging different HW capabilities with combinations of CPU and GPU at the network edge and in the fog, where the content is consumed, 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, multi-variate 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

XR profits from extensive research in the past years in gaming, machine learning, network telemetry, high resolution imaging, smart cities, and IoT. Information Centric Networking (and related) approaches that combine publish subscribe and distributed storage are also very suited for the multisource-multidestination applications of XR. Hence XR solutions exist and are more and more deployed outside entertainment, and with the focus 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 remain mostly one-way: information is sent to the destination, rendered and displayed based on local processing. A lot of the video information goes upstream. There are still very little truly interactive immersive media applications over networks except within a single subnetwork that is characteristic of some smart cities, manufacturing applications or training.

3.2.4. Opportunities

While delay is inherently related to information transmission and 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:

* Reduced latency: 20 ms is usually cited as an upper limit for XR applications. Storage and preprocessing of scenes in local elements (includng 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, pre-fetching and pre-caching, 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 realted to caching/fetching but also context based flow direction and aggregation.

* Monitoring: Since bandwidth and data are fundamental for 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 artificial intelligence in network management to learn about flows and adapt resources based on both dataplane 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: How can the interoperability of CPU/GPU be optimized and used jointly with PNDs to combine low-level packet filtering and redirection with the higher layer processing needed for image processing, feature selection, and haptics?

* RQ 3.2.4: Can the use of joint learning algorithms across both data center and edge computers be used to create 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 3.2.5: Can COIN improve the dynamic distribution of control, forwarding, and storage resources and related usage models in XR?

* RQ 3.2.6: How COIN provide the necessary infrastructure for the use of interactive XR everywhere?

3.2.6. Requirements

* Req 3.2.1: COIN systems for XR MUST allow joint collaboration across networks not just as part of the same subnetwork.

* Req 3.2.2: COIN systems for XR SHOULD provide multi-stream combining in the network for multi-views and efficient transmission.

* Req 3.2.3: COIN systems for XR SHOULD be able to dynamically include extra streams for data-intensive services and processes.

* Req 3.2.4: COIN systems for XR MAY use edge networking and computing for improved performance and performance management independent of a cloud connection.

* Req 3.2.5: COIN systems for XR MAY integrate local and fog caching with cloud-based pre-rendering.

* Req 3.2.6: COIN systems for XR SHOULD jointly optimize COIN and higher layer protocols to reduce latency especially in data-intensive applications at the edge.

* Req 3.2.7: COIN systems for XR SHOULD support nomadicity and mobility.

* Req 3.2.8: COIN systems for XR SHOULD provide means for performance optimization that reduces transmitted data and optimizes loss protection.

* Req 3.2.9: COIN systems for XR MAY provide means for trend analysis and telemetry.

* Req 3.2.10: COIN systems for XR SHOULD integrate PNDs with holography, 3D displays, and image rendering processors for offering to improve service location selections.

* Req 3.2.11: COIN systems for XR MAY provide means for managing the quality of XR sessions through reduced in-network congestion and improve flow delivery by determining how to prioritize XR data.

3.3. Personalized and interactive performing arts

3.3.1. Description

This use case covers live productions of the performing arts where the performers and audience are in different physical locations. The performance is conveyed to the audience through multiple networked streams which may be tailored to the requirements of individual audience members; and the performers receive live feedback from the audience.

There are two main aspects: i) to emulate as closely as possible the experience of live performances where the performers and audience 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 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 - amongst, above or behind the performers (but within some limits which 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/product-placement, other enhancements/filters to make the performance accessible to disabled audience members (removal of flashing images for epileptics, alternative color schemes for color-blind audience members, etc.).

3.3.2. Characterization

There are several chained functional entities which 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

  - 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 the audience member's chosen perspective

  - to undertake remote rendering according to viewer position, e.g., 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 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 audience preferences and requirements makes it unfeasible 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 audience members' personalized streams.

* Rendering of VR headset content to follow viewer head movements has an upper bound on lag to maintain viewer 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 undertake the personalization 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 from performers to audience 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 the Mobile Application Offloading, Extended Reality, and Content Delivery Networks use cases.

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, multi-flow 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 audience members?

* RQ 3.3.2: How far from the direct network path from performer to audience 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 the COIN System?

* 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, iii) routed through disjoint paths through different PNDs/routers? This RQ raises issues associated with synchronisation across multiple media streams and sub-streams [RFC7272] as well as time synchronisation 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 endsystems?

3.3.6. Requirements

* Req 3.3.1: Users SHOULD be able to specify requirements on network and processing metrics (such as latency and throughput bounds).

* Req 3.3.2 The COIN System SHOULD be able to respect user-specified requirements and constraints when routing flows and selecting PNDs for executing (COIN) Programs.

* Req 3.3.3: A COIN System SHOULD be able to synchronize flow treatment and processing across multiple related flows which may be on disjoint paths.

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 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 pre-processed or aggregated with other information before it can be used. PLCs are not designed for this array of tasks and computations could theoretically be moved to more powerful devices. These devices are no longer close to the controlled objects and induce additional latency. 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.

```
 reference
   state      ------------        --------    Output
---------->  | Controller | ---> | System | ---------->
           ^  ------------        --------       |
           |                                     |
           |   observed state                    |
           |                    ---------        |
            -------------------| Sensors | <-----
                                ---------
```

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. 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, e.g., 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 [RUETH] 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?

-How to account for the limited computational precision of PNDs, typically only allowing for integer precision computation, while floating-point precision is needed by most control algorithms (cf. [KUNZE-APPLICABILITY])?

-How to find suitable tradeoffs regarding simplicity of the control function ("accuracy of the control") and implementation complexity ("implementability")?

*RQ 4.1.2: How to distribute the simplified versions in the network?

-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, farer away), "very accurate & very high latency" (cloud environments, far away)?

-Who decides which control instance is executed and how?

-How do the different control instances interact?

4.1.6. Requirements

*Req 4.1.1: The interaction between the COIN execution environments and the global controller SHOULD be explicit.

*Req 4.1.2: The interaction between the COIN execution environments and the global controller MUST NOT negatively impact the control quality.

*Req 4.1.3: Actions of the COIN execution environments MUST be overridable by the global controller.

*Req 4.1.4: Functions in COIN execution environments SHOULD be executed with predictable delay.

*Req 4.1.5: Functions in COIN execution environments MUST be executed with predictable accuracy.

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 and upgrading is hence not always feasible or possible. A major challenge is thus to devise a methodology that is able to handle such amounts of data over limited access links.

Data filtering and pre-processing, 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 pre-processing 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 thus 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, part 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 packet-based sub-sampling 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 variant is 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 manifold computations. Some of them are very complex or need the complete sensor data during the computation, but there are also simpler operations which 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. While they allow for handling large volume applications, they are tied to performant server machines and upscaling the information density also requires a corresponding upscaling of the compute infrastructure.

4.2.4. Opportunities

* (Semantic) packet filtering based on packet header and payload, as well as multi-packet information can (drastically) reduce the data volume, possibly even without losing any important information.

* (Semantic) data (pre-)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

* 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?

  - Which criteria for filtering make sense?

* RQ 4.2.3: Which kinds of COIN programs can be leveraged for (pre-)processing steps?

  - How complex can they become?

* RQ 4.2.4: How to distribute and coordinate COIN programs?

* RQ 4.2.5: How to dynamically change COIN programs?

* RQ 4.2.6: How to incorporate the (pre-)processing and filtering steps into the overall system?

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

4.2.6. Requirements

* Req 4.2.1: Filters and preprocessors MUST conform to application-level syntax and semantics.

* Req 4.2.2: Filters and preprocessors MAY leverage packet header and payload information.

* Req 4.2.3: Filters and preprocessors SHOULD be reconfigurable at run-time.

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 either directly assist humans or perform tasks autonomously. 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.

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. 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.

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.

4.3.5. Research Questions

* RQ 4.3.1: Which additional safety measures can be provided?

  - Do these measures actually improve safety?

* RQ 4.3.2: Which sensor information can be combined and how?

4.3.6. Requirements

* Req 4.3.1: COIN-based safety measures MUST NOT degrade existing safety measures.

* Req 4.3.2: COIN-based safety measures MAY enhance existing safety measures.

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 and they often utilize DNS-based indirection to serve the request on behalf of the origin server.

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, who will eventually serve the user's request. The replication architecture and mechanisms itself differs from one (CDN) provider to another, and often utilizes 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) 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.

5.1.4. Opportunities

* Supporting service-level routing of requests (service routing in [APPCENTRES]) to specific (COIN) program instances may improve on end user experience in faster retrieving (possibly also more, e.g., better quality) content.

* Supporting the constraint-based selection of a specific (COIN) program instance over others (constraint-based routing in [APPCENTRES]) may improve the overall end user experience by selecting a 'more suitable' (COIN) program instance over another, e.g., avoiding/reducing overload situations in specific (COIN) program instances.

* Supporting Layer 2 capabilities for multicast (compute interconnection and collective communication in [APPCENTRES]) may increase the network utilization and therefore increase the overall system utilization.

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?

*RQ 5.1.2: What forwarding methods may support the required multicast capabilities (see [FCDN])?

*RQ 5.1.3: What are the right routing constraints that reflect both compute and network capabilities?

*RQ 5.1.4: Could traffic steering be performed on the data path and per service request? If so, what would be performance improvements?

*RQ 5.1.5: How could storage be traded off against frequent, multicast-based replication (see [FCDN])?

*RQ 5.1.6: What scalability limits exist for L2 multicast capabilities? How to overcome them?

5.1.6. Requirements

Requirements 3.1.1 through 3.1.6 also apply for CDN service access. In addition:

*Req 5.1.1: Any solution SHOULD utilize Layer 2 multicast transmission capabilities for responses to concurrent service requests.

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

5.2.1. Description

Layer 2 connected compute resources, e.g., in regional or edge data centers, base stations, and even end user devices, provide the opportunity for infrastructure providers to offer CFaaS-like offerings to application providers. App and service providers may utilize the compute fabric exposed by this CFaaS offering for the purposes defined through their applications and services. In other words, 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, a tenant here defined as the provider of at least one application. For this, we foresee an interaction between 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, e.g., 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) Layer 2 networks, which in turn allows for connecting compute resources for a distributed computational task. 5G-LAN [SA2-5GLAN] specifies a cellular L2 bearer for interconnecting L2 resources within a single cellular operator. The work in [ICN5GLAN] outlines using a path-based forwarding solution over 5G-LAN as well as SDN-based LAN connectivity together with an ICN-based naming of IP and HTTP-level resources to achieve computational interconnections, including scenarios such as those outlined in Section 3.1. L2 network virtualization (see, e.g., [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 (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 (constraint-based routing in [APPCENTRES]) will allow for optimizing both the CFaaS provider constraints as well as tenant-specific constraints.

* Supporting Layer 2 capabilities for multicast (compute interconnection and collective communication in [APPCENTRES]) will allow for increasing 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.

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 L2 fabric?

*RQ 5.2.2: How to dynamically integrate resources, particularly when driven by tenant-level requirements and changing service-specific constraints?

*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?

5.2.6. Requirements

Requirements 3.1.1 through 3.1.6 also apply for the provisioning of services atop the CFaaS. In addition:

*Req 5.2.1: Any solution SHOULD expose means to specify the requirements for the tenant-specific compute fabric being utilized for the service execution.

*Req 5.2.2: Any solution SHOULD allow for dynamic integration of compute resources into the compute fabric being utilized for the app execution; those resources include, but are not limited to, end user provided resources. From a COIN system perspective, new resources must be possible to be exposed as possible (COIN) execution environments.

*Req 5.2.3: Any solution MUST provide means to optimize the interconnection of compute resources, including those dynamically added and removed during the provisioning of the tenant-specific compute fabric.

*Req 5.2.4: Any solution MUST provide means for ensuring that availability and usage of resources is accounted for.

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, e.g., be P4 programs, OpenFlow rules, or higher layer programs. This feature can enable other use cases described in this draft to be deployed using virtual networks services, over underlying networks such as datacenters, 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 counters 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. Section 5.1 of [I-D.ravi-icnrg-5gc-icn] 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, e.g., a 5G Non-Public Network. 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.

```
                                     ..... Tenant ........
                          P4 program :                   :
                          deployment :         Operation :
                                     V                   :
  +-----+  air interface +----------------+              :
  | UE1 +----------------+                |              :
  +-----+                |                |              :
                         |                |              :
  +-----+                |                |              V
  | UE2 +----------------+     5GLAN      |      +------------+
  +-----+                |    Logical     +------+ Controller |
                         |     Switch     |  P4  +-------+----+
  +-----+                |                |  runtime     |
  | UE3 +----------------+                |  API         |
  +-----+                |                |              |
                         |                |              |
  +-----+                |                |              |
+-+ UE4 +----------------+                |              |
| +-----+                +----------------+              |
|                                                        |
| Fixed or wireless connection                          |
|                                    P4 runtime API      |
|  +---------+           +-------------------------------+
+--+ Device1 |           |
|  +---------+           |
|                        |
|  +---------+    +------+-----+
`--+ Device2 +-----+ P4 Switch  +--->(fixed network)
   +---------+    +------------+
```

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 datacenter 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 datacenters 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., request specific QoS for some flows in 5G or datacenters, 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 be able to influence, and be influenced by, the underling network. Since some information and actions may be available on some nodes and not others, underlying network awareness may impose additional constraints on distributed network programs location.

* 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, while allowing 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 [I-D.hsingh-coinrg-reqs-p4comp] and [Sultana].

* RQ 5.3.3: Multi-Tenancy Support: 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.

* 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 the present section focused on packet/flow processing.

5.3.6. Requirements

* Req 5.3.1: A COIN system supporting virtualization SHOULD enable tenants to deploy COIN programs onto their virtual networks.

* Req 5.3.2: A virtual COIN program SHOULD process flows/packets once and only once (or at least once for idempotent operations), even if the program is distributed over multiple PNDs.

* Req 5.3.3: Multi-tenancy SHOULD be supported for virtual COIN programs, i.e., instances of virtual COIN programs from different tenants can share underlying PNDs. This includes requirements for secure isolation between tenants, and fair (or policy-based) sharing of computing resources.

* Req 5.3.4: Virtual COIN programs SHOULD support mobility of endpoints.

6. Enabling new COIN capabilities

6.1. Distributed AI

6.1.1. Description

There is a growing range of use cases demanding for the realization of AI capabilities among distributed endpoints. Such demand may be driven by the need to increase overall computational power for large-scale problems. From a COIN perspective, those capabilities may be realized as (COIN) programs and executed throughout the COIN system, including in PNDs.

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).

6.1.2. Characterization

Examples for large-scale AI problems include biotechnology and astronomy related reasoning over massive amounts of observational input data. Examples for localizing input data for privacy reasons include radar-like application for the development of topological mapping data based on (distributed) radio measurements at base stations (and possibly end devices), while the processing within radio access networks (RAN) already constitutes a distributed AI problem to a certain extent albeit with little flexibility in distributing the execution of the AI logic.

6.1.3. Existing Solutions

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

A number of activities on distributed AI exist in the area of developing the 5th and 6th generation mobile network with various activities in the 3GPP 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 requests (service routing in [APPCENTRES]), with AI services being exposed to the network and executed as part of (COIN) programs in selected (COIN) program instances, may provide a highly distributed execution of the overall AI logic, thereby addressing, e.g., localization but also computational concerns (scale-in/out).

* The support for constraint-based selection of a specific (COIN) program instance over others (constraint-based routing in [APPCENTRES]) may allow for utilizing the most suitable HW capabilities (e.g., support for specific AI HW assistance in the COIN element, including a PND), while also allowing to select resources, e.g., based on available compute ability such as number of cores to be used.

* 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?

* RQ 6.1.2: How to achieve scalable multicast delivery with rapidly changing receiver sets?

* RQ 6.1.3: What COIN capabilities may support the collective communication patterns found in distributed AI problems?

* RQ 6.1.4: How to provide a service routing capability that supports any invocation protocol (beyond HTTP)?

6.1.6. Requirements

Requirements 3.1.1 through 3.1.6 also apply for general distributed AI capabilities. In addition:

* Req 6.1.1: Any COIN system MUST provide means to specify 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.

* Req 6.1.2: Any COIN system MUST provide support for app/micro-service specific invocation protocols for requesting (COIN) program services exposed to the COIN system.

7. Security Considerations

Deploying COIN solutions to the use cases described in this document may pose a risk for security breaches if the solutions are not deployed with security and authentication mechanisms in place. In particular, many early PND-based approaches work on unencrypted plain text data and often customize packet payload to account for missing capabilities of early-generation PNDs. Such need for operating on unencrypted data either limits the applicability of COIN solutions to those parts of the packet transfer that happens to be unencrypted or poses a strong requirement to user data to not being encrypted for the sake of utilizing COIN capabilities; a requirement hardly aligned with the strong trend to encrypting all user-related data in traversing packets. Thus, even without having analyzed the use cases in more detail in this document, designing meaningful solutions for providing authentication as well as incorporating the rightfully ongoing trend to more and more end-to-end encrypted traffic into COIN will be key.

8. IANA Considerations

N/A

9. Conclusion

This document presented use cases gathered from several fields that can and could profit from capabilities that are provided by in-network and, more generally, distributed compute capabilities. We distinguished 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 identified opportunities arising from utilizing COIN capabilities as well as research questions that may need to be addressed before being able to reap those opportunities. We also outlined possible requirements for building a COIN system that may realize these use cases.

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 supply-chain, health care, intelligent cities and transportation and even financial services (among others).

With this in mind, updates to this document might become necessary or desirable in the future to capture this extended view on what may be possible. We are, however, confident that the current selection of use cases, each describing the dimensions of opportunities, research questions, and requirements, already represents a useful set of scenarios that yield themselves for a subsequent analysis that is currently intended to be performed in [USECASEANALYSIS]. Through this, the use cases presented here together with the intended analysis provide direct input into the milestones of the COIN RG in terms of required functionalities.

10. Acknowledgements

The authors would like to thank Chathura Sarathchandra for reviewing the document and David Oran, Phil Eardley, Stuart Card, Jeffrey He, Toerless Eckert, and Jon Crowcroft for providing comments on earlier versions of the document.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

11.2. 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>.

[ETSI] ETSI, "Multi-access Edge Computing (MEC)", 2022, <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, "A Flexible and Efficient CDN Infrastructure without DNS Redirection of Content Reflection", <https://arxiv.org/pdf/1803.00876.pdf>.

[GLEBKE] Glebke, R., Henze, M., Wehrle, K., Niemietz, P., Trauth, D., Mattfeld MBA, 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>.

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

[GSLB] Cloudflare, "What is global server load balancing (GSLB)?", 2022, <https://www.cloudflare.com/learning/cdn/glossary/global-server-load-balancing-gslb/>.

[I-D.hsingh-coinrg-reqs-p4comp] 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>.

[I-D.ravi-icnrg-5gc-icn] 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>.

[ICN5GLAN] 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), 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), 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", 2022, <https://blogs.oracle.com/cloud-infrastructure/post/first-principles-l2-network-virtualization-for-lift-and-shift>.

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

[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/rfc/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/rfc/rfc8039>.

[RUETH] Rueth, 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, 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>.

[Stoyanov]
Stoyanov, R. and N. Zilberman, "MTPSA: Multi-Tenant Programmable Switches", ACM P4 Workshop in Europe (EuroP4'20) , 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", 2020, <https://flightplan.cis.upenn.edu/flightplan.pdf>.

[TERMINOLOGY] Kunze, I., Wehrle, K., Trossen, D., Montpetit, M., de Foy, X., Griffin, D., and M. Rio, "Terminology for Computing in the Network", Work in Progress, Internet-Draft, draft-irtf-coinrg-coin-terminology-00, 10 March 2023, <https://datatracker.ietf.org/doc/html/draft-irtf-coinrg-coin-terminology-00>.

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

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

[USECASEANALYSIS] Kunze, I., 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-00, 10 March 2023, <https://datatracker.ietf.org/doc/html/draft-irtf-coinrg-use-case-analysis-00>.

[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), DOI 10.1109/etfa.2018.8502456, September 2018, <https://doi.org/10.1109/etfa.2018.8502456>.

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
Concordia University
1455 De Maisonneuve
Montreal H3G 1M8
Canada

Email: marie@mjmontpetit.com

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