Workgroup: COINRG
Internet-Draft: draft-irtf-coinrg-use-cases-02
Published: 7 March 2022
Intended Status: Informational
Expires: 8 September 2022
Authors: I. Kunze K. Wehrle D. Trossen
RWTH Aachen RWTH Aachen Huawei
M.J. Montpetit X. de Foy
Concordia InterDigital Communications, LLC
D. Griffin M. Rio
UCL 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 in several contexts, it has to be carefully placed into the context of the general Internet communication.

This document discusses some use cases to demonstrate how real applications can benefit from COIN and to showcase essential requirements that have to be fulfilled by COIN applications.

Table of Contents

1. Introduction

The Internet was designed as a best-effort packet network that offers limited guarantees regarding the timely and successful transmission of packets. 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 design choice has shown suitable for a wide variety of applications and has helped in the rapid growth of the Internet.

However, with the expansion of the Internet, 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, 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 core to the efforts in the COIN RG; we refer to those capabilities as 'COIN capabilities' in the remainder of the document.

We believe that such 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 the presence of computing resources inside the network impacts existing services and applications or allows for innovation in emerging fields.

Through delving into some individual examples within each of the above categories, we aim to outline opportunities and propose possible research questions for consideration by the wider community when pushing forward the 'in-network computing' vision. Furthermore, insights into possible requirements for an evolving solution space of collected COIN capabilities is another objective of the individual use case descriptions. This results in the following taxonomy used to describe each of the use cases:

1. Description: Purpose of the use case and 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: Describe, if existing, current methods that may realize the use case.

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

5. Research questions: State essential questions that are suitable for guiding research to achieve the outlined opportunities

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

In Section 7, we will summarize the key research questions across all use cases and identify key requirements across all use cases. This will provide a useful input into future roadmapping on what COIN capabilities may emerge and how solutions of such capabilities may look like. It will also identify what open questions remain for these use cases to materialize as well as define requirements to steer future (COIN) research work.

2. Terminology

The following terminology has been partly aligned with [I-D.draft-kutscher-coinrg-dir]:

(COIN) Program: a set of computations requested by a user

(COIN) Program Instance: one currently executing instance of a program

(COIN) Function: a specific computation that can be invoked as part of a program

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

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

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

(COIN) Execution Environment: a class of target environments for function execution, for example, a JVM-based execution environment 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

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 combining with 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) programs may be left in the headset, while the compute intensive real-time VR content processing (COIN) programs 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 (COIN) functions may be moved to other (e.g., more suitable) devices, including PNDs, which have exposed the corresponding (COIN) programs as individual (COIN) program instances to the (COIN) system by means of a 'service identifier'. The result of the latter 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 suitable 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 (COIN) program deployment (and orchestration) on most suitable end systems or PNDs ('processing server').

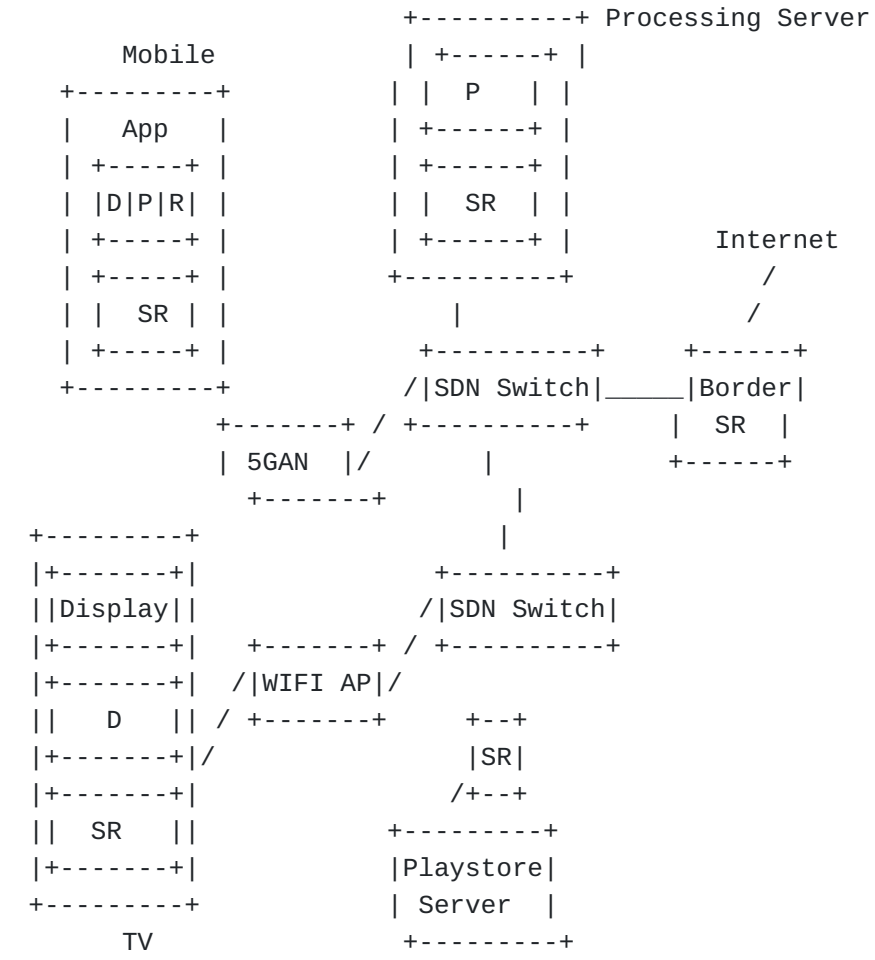

Figure 1: Application Function Offloading Example.

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/many-viewing scenarios, thereby realizing a service-level multicast capability towards more than one (COIN) program.

3.1.3. Existing Solutions

NOTE: material on solutions like ETSI MEC will be added here later

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 application towards 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 identifying service-level in-network computing elements will allow for routing service requests to those COIN elements, including PNDs, therefore possibly allowing for new in-network 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 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 in-network 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 (the metaverse) taken together as Extended Reality (XR) are the drivers of a number of advances in interactive technologies. XR is one example of the Multisource-Multidestination Problem that combines video, haptics, and tactile experiences in interactive or networked multi-party and social interactions. While initially associated with gaming and entertainment, XR applications now include remote diagnosis, maintenance, telemedicine, manufacturing and assembly, autonomous systems, smart cities, and immersive classrooms.

Because XR requirements include the need to provide real-time interactivity for immersive and increasingly mobile immersive applications with tactile and time-sensitive data and high bandwidth for high resolution images and local rendering for 3D images and holograms, they are difficult to run over traditional networks; in consequence innovation is needed to deply the full potential of the applications.

3.2.2. Characterization

Collaborative XR experiences 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. XR deals with personal information and potentially protected content this an 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 can use recent trend analysis and mechanisms, including machine learning to find these trends and reduce the size of the data sets. Video holography and haptics require very low delay or generate large amounts of data, both requiring a careful look at data filtering and reduction, functional distribution and partitioning.

The operation of XR over networks requires some computing in the nodes from content source to destination. But a lot of these remain

in the realm of research to resolve the resource allocation problem and provide adequate quality of experience. These include multi-variate and heterogeneous goal optimization problems at merging nodes requiring advanced analysis. Image rendering and video processing in XR leverages different HW capabilities combinations of CPU and GPU at the edge (even at the mobile edge) and in the fog network where the content is consumed. It is important to note that the use of in-network computing for XR does not imply a specific protocol but targets an architecture enabling the deployment of the services.

3.2.3. Existing Solutions

In-network computing for XR profits from the heritage of extensive research in the past years on Information Centric Networking, Machine Learning, network telemetry, imaging and IoT as well as distributed security and in-network coding.

*Enabling Scalable Edge Video Analytics with Computing-In-Network (Jun Chen Jiang of the University of Chicago): this work brings a periodical re-profiling to adapt the video pipeline to the dynamic video content that is a characteristic of XR. The implication is that we "need tight network-app coupling" for real time video analytics.

*VR journalism, interactive VR movies and meetings in cyberspace (many projects PBS, MIT interactive documentary lab, Huawei research - references to be provided): typical VR is not made for multiparty and these applications require a tight coupling of the local and remote rendering and data capture and combinations of cloud (for more static information) and edge (for dynamic content).

*Local rendering of holographic content using near field computation (heritage from advances cockpit interactions - looking for non military papers): a lot has been said recently of the large amounts of data necessary to transmit and use holographic imagery in communications. Transmitting the near field information and rendering the image locally allows to reduce the data rates by 1 or 2.

*ICE-AR [ICE] project at UCLA (Jeff Burke): while this project is a showcase of the NDN network artchitecture it also uses a lof of edge-cloud capabilities for example for inter-server games and advanced video applications.

3.2.4. Opportunities

*Reduced latency: the physical distance between the content cloud and the users must be short enough to limit the propagation delay

to the 20 ms usually cited for XR applications; the use of local CPU and IoT devices for range of interest (RoI) detection and fynamic rendering may enable this.

*Video transmission: better transcoding and use of advanced context-based compression algorithms, pre-fetching and pre-caching and movement prediction not only in the cloud.

*Monitoring: telemetry is a major research topic for COIN and it enables to monitor and distribute the XR services.

*Network access: push some networking functions in the kernel space into the user space to enable the deployment of stream specific algorithms for congestion control and application-based load balancing based on machine learning and user data patterns.

*Functional decomposition: functional decomposition, localization and discovery of computing and storage resources in the network. But it is not only finding the best resources but qualifying those resources in terms of reliability especially for mission critical services in XR (medicine for example). This could include intelligence services.

3.2.5. Research Questions

*RQ 3.2.1: Can current programmable network entities be sufficient to provide the speed required to provide and execute complex filtering operations that includes metadata analysis for complex and dynamic scene rendering?

*RQ 3.2.2: How can the interoperability of CPU/GPU be optimized to combine low level packet filtering with the higher layer processors needed for image processing and haptics?

*RQ 3.2.3: Can the use of joint learning algorithms across both data center and edge computers be used to create optimal functionality allocation and the creation of semi-permanent datasets and analytics for usage trending 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?

3.2.6. Requirements

*Req 3.2.1: Allow joint collaboration.

*Req 3.2.2: Provide multi-views.

*Req 3.2.3: Include extra streams dynamically for data intensive services, manufacturing and industrial processes.

*Req 3.2.4: Enable multistream, multidevice, multidestination applications.

*Req 3.2.5: Use new Internet Architectures at the edge for improved performance and performance management.

*Req 3.2.6: Integrate with holography, 3D displays and image rendering processors.

*Req 3.2.7: All the use of multicast distribution and processing as well as peer to peer distribution in bandwidth and capacity constrained environments.

*Req 3.2.8: Evaluate the integration local and fog caching with cloud-based pre-rendering.

*Req 3.2.9: Evaluate ML-based congestion control to manage XR sessions quality of service and to determine how to priortize data.

*Req 3.2.10: Consider higher layer protocols optimization to reduce latency especially in data intensive applications at the edge.

*Req 3.2.11: Provide trust, including blockchains and smart-contracts to enable secure community building across domains.

*Req 3.2.12: Support nomadicity and mobility (link to mobile edge).

*Req 3.2.13: Use 5G slicing to create independent session-driven processing/rendering.

*Req 3.2.14: Provide performance optimization by data reduction, tunneling, session virtualization and loss protection.

*Req 3.2.15: Use AI/ML for trend analysis and data reduction when appropriate.

3.3. Personalised 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 theatre; and ii) to enhance traditional physical performances with features such as personalisation of the experience according to the preferences or needs of the audience members.

Examples of personalisation include:

* Viewpoint selection such as choosing a specific seat in the theatre 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 colour schemes for colour-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

* Personalisation 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: 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 in-network

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:

- *Personalisation of the performance according to audience preferences and requirements makes it unfeasible to be done in a centralised manner at the performer premises: the computational resources and network bandwidth would need to scale with the number of audience members' personalised 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 personalisation or the viewers' network may not have the capacity to receive all of the constituent streams to undertake the personalisation 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 minimised, which reduces the opportunity to route streams via large-scale processing capabilities at centralised data-centres.

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 personalisation functions on-path as (COIN) Programs in PNDs will avoid detour/stretch to central servers which increases latency as well as the consumption of bandwidth on more network resources (links and routers). For example, in this use case the chain of (COIN) Programs and propagation over the interconnecting network segments for performance capture, aggregation, distribution, personalisation,

consumption, capture of audience response, feedback processing, aggregation, 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 laugher or other emotional responses in a theatre setting).

*Processing of media streams allows (COIN) Programs, PNDs and the wider (COIN) System/Environment to be contextual aware of flows and their requirements which can be used for determining network treatment of the flows, e.g. path selection, prioritisation, multi-flow coordination, synchronisation & resilience.

3.3.5. Research Questions:

*RQ 3.3.1: In which PNDs should (COIN) Programs for aggregation, encoding and personalisation 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 network synchronisation across multiple streams to allow for merging, audio-video interpolation and other cross-stream processing functions that require time synchronisation for the integrity of the output? How can this be achieved considering that synchronisation 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?

*RQ 3.3.5: Where will COIN Programs will 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 - 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 in-network processing? Where is the boundary between in-network processing 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) and the COIN System should be able to respect those requirements and constraints when routing flows and selecting PNDs for executing (COIN) Programs.

*Req 3.3.2: A COIN System should be able to synchronise flow treatment and processing across multiple related flows which may be on disjoint paths.

4. Supporting new COIN Systems

While the best-effort nature of the Internet enables a wide variety of applications, there are several domains whose requirements are hard to satisfy over regular best-effort networks.

Consequently, there is a large number of specialized appliances and protocols designed to provide the required strict performance guarantees, e.g., regarding real-time capabilities.

Time-Sensitive-Networking [TSN] as an enhancement to the standard Ethernet, e.g., tries to achieve these requirements on the link layer by statically reserving shares of the bandwidth. However, solutions on the link layer alone are not always sufficient.

The industrial domain, e.g., currently evolves towards increasingly interconnected systems in turn increasing the complexity of the underlying networks, making them more dynamic, and creating more diverse sets of requirements. Concepts satisfying the dynamic performance requirements of modern industrial applications thus become harder to develop. In this context, COIN offers new possibilities as it allows to flexibly distribute computation tasks across the network and enables novel forms of interaction between communication and computation providers.

This document illustrates the potential for new COIN systems using the example of the industrial domain by characterizing and analyzing specific scenarios to showcase potential requirements, as specifying general requirements is difficult due to the domain's mentioned diversity.

4.1. Industrial Network Scenario

Common components of industrial networks can be divided into three categories as illustrated in Figure 2. Following [I-D.mcbride-edge-data-discovery-overview], EDGE DEVICES, such as sensors and actuators, constitute the boundary between the physical and digital world. They communicate the current state of the physical world to the digital world by transmitting sensor data or let the digital world interact with the physical world by executing actions after receiving (simple) control information. The processing of the sensor data and the creation of the control information is done on COMPUTING DEVICES. They range from small-powered controllers close to the EDGE DEVICES, to more powerful edge or remote clouds in larger distances. The connection between the EDGE and COMPUTING DEVICES is established by NETWORKING DEVICES. In the industrial domain, they range from standard devices, e.g., typical Ethernet switches, which can interconnect all Ethernet-capable hosts, to proprietary equipment with proprietary protocols only supporting hosts of specific vendors.

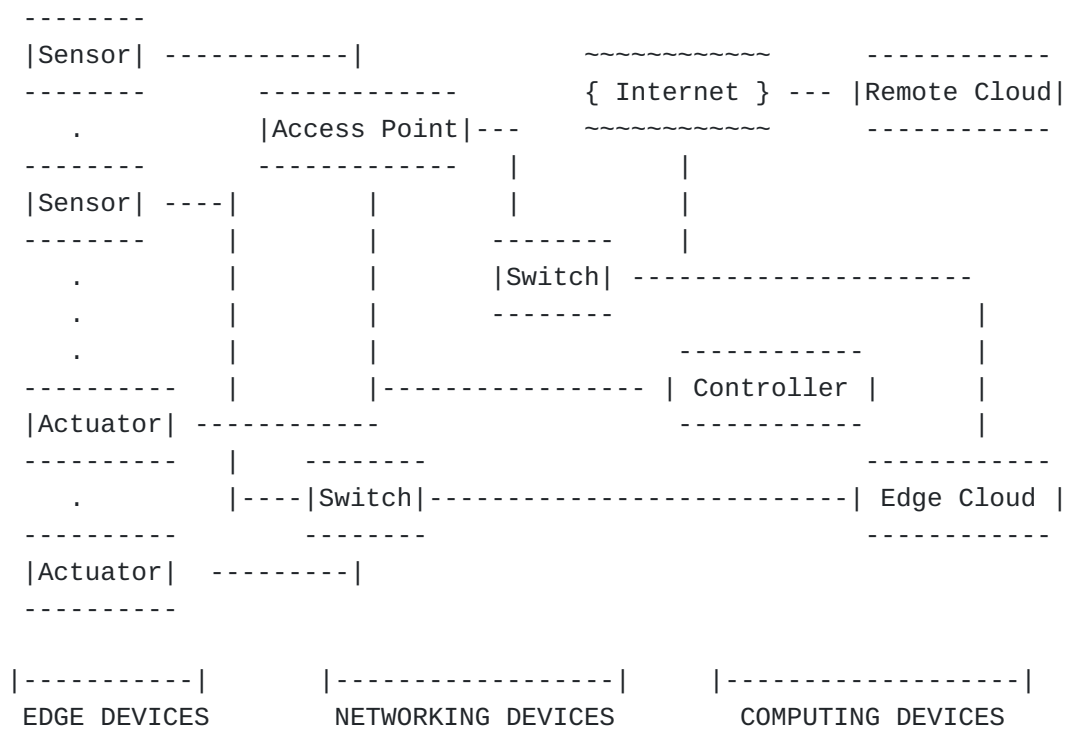

Figure 2: Industrial networks show a high level of heterogeneity.

4.2. In-Network Control / Time-sensitive applications

4.2.1. Description

The control of physical processes and components of a production line 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 amount 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.

4.2.2. Characterization

A control process consists of two main components as illustrated in Figure 3: 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.

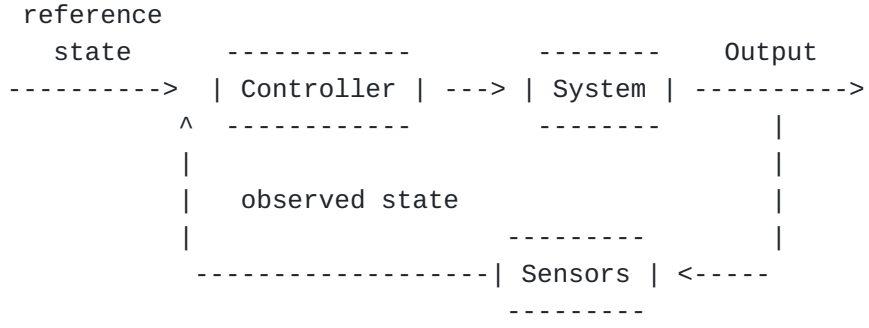

Figure 3: Simple feedback control model.

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.

4.2.3. 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.

4.2.4. Opportunities

COIN offers the possibility of bringing the system and the controller closer together, thus possibly satisfying the latency requirements, by performing simple control logic on PNDs and/or in COIN execution environments.

While control models, in general, can become involved, there is a variety of control algorithms that are composed of simple computations such as matrix multiplication. These are supported by some PNDs and it is thus possible to compose simplified approximations of the more complex algorithms and deploy them in the network. While the simplified versions induce a more inaccurate control, they allow for a quicker response and might be sufficient to operate a basic tight control loop while the overall control can still be exercised from the cloud.

Opportunities:

* Execute simple (end-host) COIN functions on PNDs to satisfy tight latency constraints of control processes

4.2.5. Research Questions

Bringing the required computations to PNDs is challenging as these devices typically only allow for integer precision computation while floating-point precision is needed by most control algorithms. Additionally, computational capabilities vary for different available PNDs [KUNZE]. Yet, early approaches like [RUETH] and [VESTIN] have already shown the general applicability of such ideas, but there are still a lot of open research questions not limited to the following:

Research Questions:

* RQ 4.2.1: How to derive simplified versions of the global (control) function?

  - How to account for the limited computational precision of PNDs?

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

* RQ 4.2.2: 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?

4.2.6. Requirements

* Req 4.2.1: The interaction between the COIN execution environments and the global controller SHOULD be explicit.

* Req 4.2.2: The interaction between the COIN execution environments and the global controller MUST NOT negatively impact the control quality.

* Req 4.2.3: Actions of the COIN execution environments MUST be overridable by the global controller.

* Req 4.2.4: Functions in COIN execution environments SHOULD be executed with predictable delay.

* Req 4.2.5: Functions in COIN execution environments MUST be executed with predictable accuracy.

4.3. Large Volume Applications - Filtering

4.3.1. Description

In modern industrial networks, processes and machines can be monitored closely resulting in large volumes of available information. This data can be used to find previously unknown

correlations between different parts of the value chain, e.g., by deploying machine learning (ML) techniques, which in turn helps to improve the overall production system. Newly gained knowledge can be shared between different sites of the same company or even between different companies [PENNEKAMP].

Traditional company infrastructure is neither equipped for the management and storage of such large amounts of data nor for the computationally expensive training of ML approaches. Off-premise cloud platforms offer cost-effective solutions with a high degree of flexibility and scalability, however, moving all data to off-premise locations poses infrastructural challenges. Pre-processing or filtering the data already in COIN execution environments can be a new solution to this challenge.

4.3.2. Characterization

4.3.2.1. General Characterization of Large Volume Applications

Processes in the industrial domain are monitored by distributed sensors which range from simple binary (e.g., light barriers) to sophisticated sensors measuring the system with varying degrees of resolution. Sensors can further serve different purposes, as some might be used for time-critical process control while others are only used as 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]. Using off-premise clouds for managing the data requires uploading or streaming the growing volume of sensor data using the companies' Internet access which is typically limited to a few hundred of Mbit/s. While large networking companies can simply upgrade their infrastructure, most industrial companies rely on traditional ISPs for their Internet access. Higher access speeds are hence tied to higher costs and, above all, subject to the supply of the ISPs and consequently not always available. A major challenge is thus to devise a methodology that is able to handle such amounts of data over limited access links.

Another aspect is that business data leaving the premise and control of the company further comes with security concerns, as sensitive information or valuable business secrets might be contained in it. Typical security measures such as encrypting the data make COIN techniques hardly applicable as they typically work on unencrypted data. Adding security to COIN approaches, either by adding functionality for handling encrypted data or devising general

security measures, is thus an auspicious field for research which we describe in more detail in Section 8.

4.3.2.2. Specific Characterization for Filtering Solutions

Sensors are often set up redundantly, i.e., part of the collected data might also be redundant. Moreover, they 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.

4.3.3. 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.

4.3.4. Opportunities

PNDs and COIN execution environments are in a unique position to reduce the data rates due to their line-rate packet processing capabilities. Using these capabilities, it is possible to filter out redundant or undesired data before it leaves the premise using simple traffic filters that are deployed in the on-premise network. There are different approaches to how this topic can be tackled.

A first step could be to scale down the available sensor data to the data rate that is needed. For example, if a sensor transmits with a frequency of 5 kHz, but the control entity only needs 1 kHz, only every fifth packet containing sensor data is let through. Alternatively, sensor data could be filtered down to a lower frequency while the sensor value is in an uninteresting range, but let through with higher resolution once the sensor value range becomes interesting.

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.

Opportunities:

* (Semantic) packet filtering based on packet header and payload, as well as multi-packet information

4.3.5. Research Questions

*RQ 4.3.1: How to design COIN programs for (semantic) packet filtering?

  -Which criteria for filtering make sense?

*RQ 4.3.2: How to distribute and coordinate COIN programs?

*RQ 4.3.3: How to dynamically change COIN programs?

*RQ 4.3.4: How to signal traffic filtering by COIN programs to end-hosts?

4.3.6. Requirements

*Req 4.3.1: Filters MUST conform to application-level syntax and semantics.

*Req 4.3.2: Filters MAY leverage packet header and payload information.

*Req 4.3.3: Filters SHOULD be reconfigurable at run-time.

4.4. Large Volume Applications - (Pre-)Preprocessing

4.4.1. Description

See Section 4.3.1.

4.4.2. Characterization

4.4.2.1. General Characterization of Large Volume Applications

See Section 4.3.2.1.

4.4.2.2. Specific Characterization for Preprocessing Solutions

There are manifold computations that can be performed on the sensor data in the cloud. Some of them are very complex or need the complete sensor data during the computation, but there are also simpler operations which can 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.

4.4.3. Existing Solutions

See Section 4.3.3.

4.4.4. Opportunities

Using expert knowledge about the exact computation steps and the concrete transmission path of the sensor data, simple computation steps can be deployed in the on-premise network to reduce the overall data volume and potentially speed up the processing time in the cloud.

Related work has already shown that in-network aggregation can help to improve the performance of distributed ML applications [SAPIO]. Investigating the applicability of stream data processing techniques to PNDs is also interesting, because sensor data is usually streamed.

Opportunities:

* (Semantic) data (pre-)processing, e.g., in the form of computations across multiple packets and potentially leveraging packet payload

4.4.5. Research Questions

* RQ 4.4.1: Which kinds of COIN programs can be leveraged for (pre-)processing steps?
  - How complex can they become?
* RQ 4.4.2: How to distribute and coordinate COIN programs?
* RQ 4.4.3: How to dynamically change COIN programs?
* RQ 4.4.4: How to incorporate the (pre-)processing steps into the overall system?

4.4.6. Requirements

* Req 4.4.1: Preprocessors MUST conform to application-level syntax and semantics.
* Req 4.4.2: Preprocessors MAY leverage packet header and payload information.
* Req 4.4.3: Preprocessors SHOULD be reconfigurable at run-time.

4.5. Industrial Safety

4.5.1. 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 decent 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.

4.5.2. 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.

4.5.3. 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.

4.5.4. Opportunities

COIN systems could leverage the increased availability of sensor data and the detailed monitoring of the factories to enable additional safety measures. Different safety indicators within the production hall can be combined within the network so that PNDs can give early responses if a potential safety breach is detected.

One possibility could be to track the positions of human workers and robots. Whenever a robot gets too close to a human in a non-working

area or if a human enters a defined safety zone, robots are stopped to prevent injuries. More advanced concepts could also include image data or combine arbitrary sensor data.

Opportunities:

* Execute simple (end-host) COIN functions on PNDs to create early emergency reactions based on diverse sensor feedback

4.5.5. Research Questions

* RQ 4.5.1: Which additional safety measures can be provided?
  - Do these measures actually improve safety?
* RQ 4.5.2: Which sensor information can be combined and how?

4.5.6. Requirements

* Req 4.5.1: COIN-based safety measures MUST NOT degrade existing safety measures.
* Req 4.5.2: 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) storing said content closer to end users for latency reduced delivery with DNS-based indirection being utilized 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, implementing a distributed logic for distributing content from the origin server to the CDN ingress and further to the CDN replication points which ultimately serve the user-facing content requests.

5.1.3. Existing Solutions

NOTE: material on solutions will be added here later

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 reduces 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 situation 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 those request question for Section 3.1:

- RQ 5.1.1: How to utilize L2 multicast to improve on CDN designs? How to utilize in-network 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 at 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 centres, base stations and even end-user devices, provide the opportunity for infrastructure providers to offer CFaaS type of 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 inter-connection 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 interconnect all resources suitably into the (tenant-specific) fabric exposed as CFaaS.

5.2.3. Existing Solutions

NOTE: material on solutions will be added here later

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

Similar to those for Section 3.1. In addition:

*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 in-network 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

For the provisioning of services atop the CFaaS, requirements 3.1.1 through 3.1.6 should be addressed, too. 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 inter-connection 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 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 for example 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 needs 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 use 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 4, the tenant operates a network including a 5GLAN network segment (seen as a single logical switch), as well as fixed segments. This can be in a plant or enterprise network, using for an example a 5G Non-Public Network (NPN). The tenant uses P4 programs to determine the operation of the fixed and 5GLAN switches. The tenant provisions a 5GLAN P4 program into the mobile network, and can also operate a controller. 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).

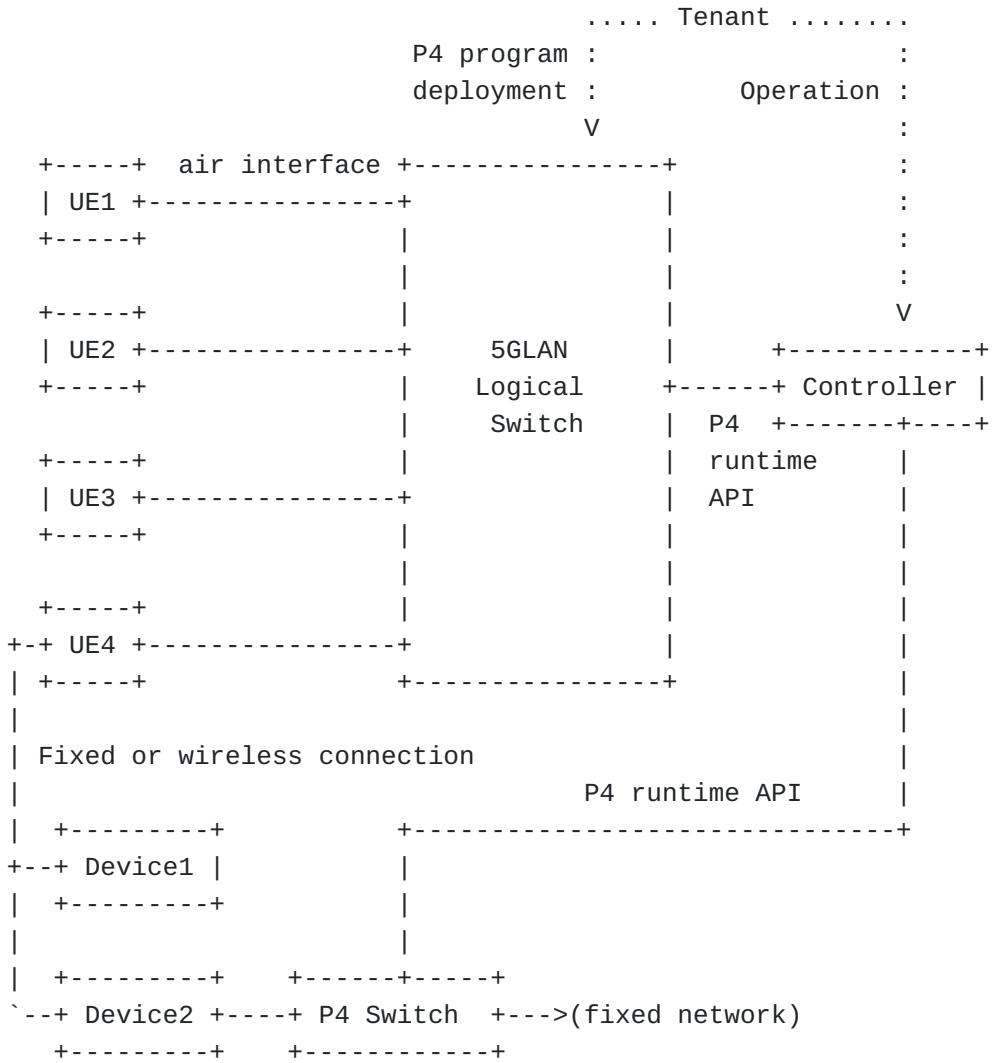

Figure 4: 5G Virtual Network Programming Overview

Looking in more details in Figure 5, the 5GLAN P4 program can be split between multiple data plane nodes (PDU Session Anchor (PSA) User Plane Functions (UPF), other UPFs, or even mobile devices), although in some cases the P4 program may be hosted on a single node. In the most general case, a distributed deployment is useful to keep traffic on optimal paths, because, except in simple cases, within a 5GLAN all traffic will not pass through a single node. In this example, P4 programs could be deployed in UPF1, UPF2, UPF3, UE3 and UE4. UE1-UE2 traffic is using a local switch on PSA UPF1, UE1-UE3 traffic is tunneled between PSA UPF1 and PSA UPF2 through the N19 interface, and UE1-UE4 traffic is forwarded throughan external Data Network (DN). Traffic between Device1 and Device2 is forwarded through UE4.

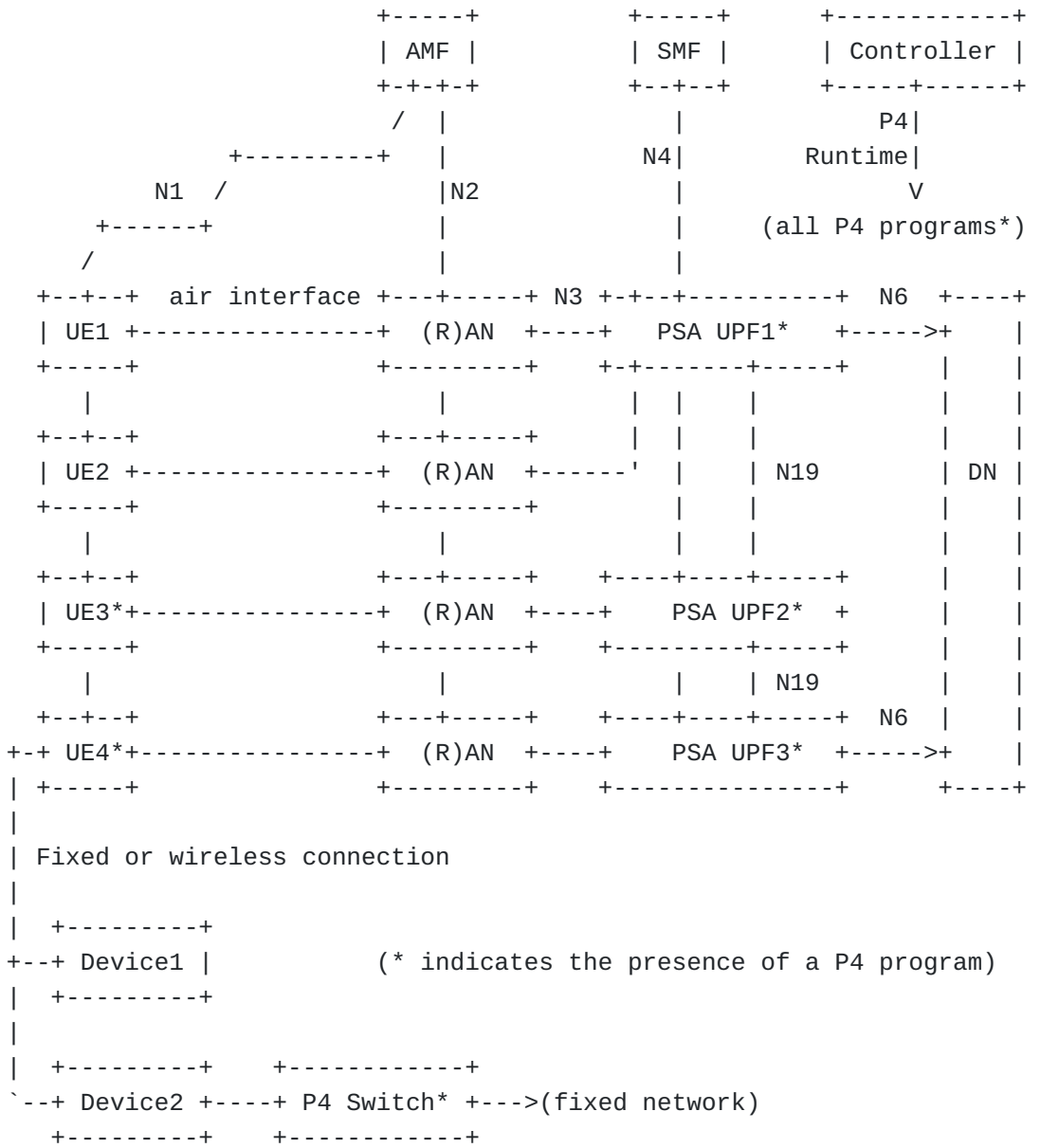

Figure 5: 5G Virtual Network Programming Details

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 developped 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 in-network computing 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, when compared with 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 service (e.g., request specific QoS for some flows in 5G or datacenters), to increases 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 (e.g., the 5G network or data center). For example, a virtual COIN program may be aware of the slice used by a flow, and possibly influence slice selection. 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. As a primary reason for this, program logic should be applied exactly once or at least once per packet, while allowing optimal forwarding path by the underlying network. For example, a 5GLAN P4 program may need to run on multiple UPFs. 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 mechanism were proposed for P4 multi-tenancy in a switch [Stoyanov], research questions remains, about isolation between tenants, 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, 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 constitute 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.

NOTE: material on solutions like ETSI MEC and 3GPP work will be added here later

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

*RQ 6.1.1: similar to use case in Section 3.1

*RQ 6.1.2: What are the communication patterns that may be supported by collective communication solutions?

*RQ 6.1.3: How to achieve scalable multicast delivery with rapidly changing receiver sets?

*RQ 6.1.4: What in-network capabilities may support the collective communication patterns found in distributed AI problems?

*RQ 6.1.5: 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. Analysis

The goal of this analysis is to identify aspects that are relevant across all use cases to help in shaping the research agenda of COINRG. For this purpose, this section will condense the opportunities, research questions, as well as requirements of the different presented use cases and analyze these for similarities across the use cases.

Through this, we intend to identify cross-cutting opportunities, research questions as well as requirements (for COIN system solutions) that may aid the future work of COINRG as well as the larger research community.

7.1. Opportunities

To be added later.

7.2. Research Questions

After carefully considering the different use cases along with their research questions, we propose the following layered categorization to structure the content of the research questions which we illustrate in Figure 6.

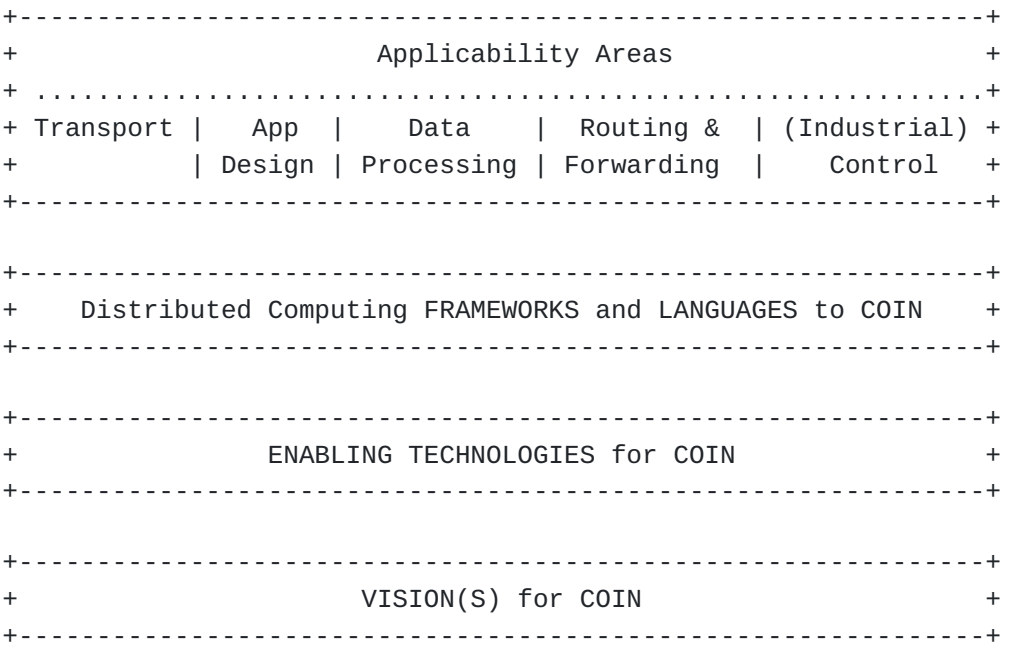

Figure 6: Research Questions Categorization

7.2.1. Categorization

Three categories deal with concretizing fundamental building blocks of COIN and COIN itself.

- *VISION(S) for COIN: Questions that aim at defining and shaping the exact scope of COIN.

- *ENABLING TECHNOLOGIES for COIN: Questions that target the capabilities of the technologies and devices intended to be used in COIN.

- *Distributed Computing FRAMEWORKS and LANGUAGES to COIN: Questions that aim at concretizing how a framework or languages for deploying and operating COIN systems might look like.

Additionally, there are use-case near research questions that are heavily influenced by the specific constraints and goals of the use

cases. We call this category "applicability areas" and refine it into the following subgroups:

- *Transport:
- *App Design:
- *Data Processing:
- *Routing & Forwarding:
- *(Industrial) Control

7.2.2. Analysis

7.2.2.1. VISION(S) for COIN

The following research questions presented in the use cases belong to this category:

3.1.8, 3.2.1, 3.3.5, 3.3.6, 3.3.7, 5.3.3, 6.1.2, 6.1.4

The research questions centering around the COIN VISION dig into what is considered COIN and what scope COIN functionality should have. In contrast to the ENABLING TECHNOLOGIES, this section looks at the problem from a more philosophical perspective.

7.2.2.1.1. Where to perform computations

The first aspect of this is where/on which devices COIN programs will/should be executed (3.3.5). In particular, it is debatable whether COIN programs will/should only be executed in PNDs or whether other "adjacent" computational nodes are also in scope. In case of the latter, an arising question is whether such computations are still to be considered as "in-network processing" and where the exact line is between "in-network processing" and "routing to end systems" (3.3.7). In this context, it is also interesting to reason about the desired feature sets of PNDs (and other COIN execution environments) as these will shift the line between "in-network processing" and "routing to end systems" (3.1.8).

7.2.2.1.2. Are tasks suitable for COIN

Digging deeper into the desired feature sets, some research questions address the question of which domains are to be considered of interest/relevant to COIN. For example, whether computationally-intensive tasks are suitable candidates for (COIN) Programs (3.3.6).

7.2.2.1.3. (Is COIN)/(What parts of COIN are) suitable for the tasks

Turning the previous aspect around, some questions try to reason whether COIN can be sensibly used for specific tasks. For example, it is a question of whether current PNDs are fast and expressive enough for complex filtering operations (3.2.1).

There are also more general notions of this question, e.g., what "in-network capabilities" might be used to address certain problem patterns (6.1.4) and what new patterns might be supported (6.1.2). What is interesting about these different questions is that the former raises the question of whether COIN can be used for specific tasks while the latter asks which tasks in a larger domain COIN might be suitable for.

7.2.2.1.4. What are desired forms for deploying COIN functionality

The final topic addressed in this part deals with the deployment vision for COIN programs (5.3.3).

In general, multiple programs can be deployed on a single PND/COIN element. However, to date, multi-tenancy concepts are, above all, available for "end-host-based" platforms, and, as such, there are manifold questions centering around (1) whether multi-tenancy is desirable for PNDs/COIN elements and (2) how exactly such functionality should be shaped out, e.g., which (new forms of) hardware support needs to be provided by PNDs/COIN elements.

7.2.2.2. ENABLING TECHNOLOGIES for COIN

The following research questions presented in the use cases belong to this category:

3.1.7, 3.1.8, 3.2.2, 4.3.4, 4.4.4, 5.1.1, 5.1.2, 5.1.6, 5.3.1, 6.1.3, 6.1.4,

The research questions centering around the ENABLING TECHNOLOGIES for COIN dig 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), this section looks at the problem from a practical perspective.

7.2.2.2.1. COIN compute technologies

Picking up on the topics discussed in Section 7.2.2.1.1 and Section 7.2.2.1.2, this category deals with how such technologies might be realized in PNDs and with which functionality should even be realized (3.1.8).

7.2.2.2.2. Forwarding technology

Another group of research questions focuses on "traditional" networking tasks, i.e., L2/L3 switching and routing decisions.

For example, how COIN-powered routing decisions can be provided at line-rate (3.1.7). Similarly, how (L2) multicast can be used for COIN (vice versa) (5.1.1), which (new) forwarding capabilities might be required within PNDs to support the concepts (5.1.2), and how scalability limits of existing multicast capabilities might be overcome using COIN (5.1.6).

In this context, it is also interesting how these technologies can be used to address quickly changing receiver sets (6.1.3), especially in the context of collective communication (6.1.4).

7.2.2.2.3. Incorporating COIN in existing systems

Some research questions deal with questions around how COIN (functionality) can be included in existing systems.

For example, if COIN is used to perform traffic filtering, how end-hosts can be made aware that data/information/traffic is deliberately withheld (4.3.4). Similarly, if data is pre-processed by COIN, how can end-hosts be signaled the new semantics of the received data (4.4.4).

In particular, these are not only questions concerning the functionality scope of PNDs or protocols but might also depend on how programming frameworks for COIN are designed. Overall, this category deals with how to handle knowledge and action imbalances between different nodes within COIN networks (5.3.1).

7.2.2.2.4. Enhancing device interoperability

Finally, the increasing diversity of devices within COIN raises interesting questions of how the capabilities of the different devices can be combined and optimized (3.2.2).

7.2.2.3. Distributed Computing FRAMEWORKS and LANGUAGES to COIN

The following research questions presented in the use cases belong to this category:

3.1.1, 3.2.3, 3.3.1, 3.3.2, 3.3.3, 3.3.5, 4.2.1, 4.2.2, 4.3.2/4.4.2, 4.3.3/4.4.3, 4.3.4, 4.4.4, 5.2.1, 5.2.2, 5.2.3, 5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.3.5,

This category mostly deals with how COIN programs can be deployed and orchestrated.

7.2.2.3.1. COIN program composition

One aspect of this topic is how the exact functional scope of COIN programs can/should be defined. For example, it might be an idea to define an "overall" program that then needs to be deployed to several devices (5.3.2). In that case, how should this composition be done: manually or automatically? Further aspects to consider here are how the different computational capabilities of the available devices can be taken into account and how these can be leveraged to obtain suitable distributed versions of the overall program (4.2.1).

In particular, it is an open question of how "service-level" frameworks can be combined with "app-level" packaging methods (3.1.1) or whether virtual network models can help facilitate the composition of COIN programs (5.3.5). This topic also again includes the considerations regarding multi-tenancy support (5.3.3, cf. Section 7.2.2.1.4) as such function distribution might necessitate deploying functions of several entities on a single device.

7.2.2.3.2. COIN function placement

In this context, another interesting aspect is where exactly functions should be placed and who should influence these decisions. Such function placement could, e.g., be guided by the available devices (3.3.5, c.f. Section 7.2.2.1.1) and their position with regards to the communicating entities (3.3.1), and it could also be specified in terms of the "distance" from the "direct" network path (3.3.2).

However, it might also be an option to leave the decision to users or at least provide means to express requirements/constraints (3.3.3). Here, the main question is how tenant-specific requirements can actually be conveyed (5.2.1).

7.2.2.3.3. COIN function deployment

Once the position for deployment is fixed, a next problem that arises is how the functions can actually be deployed (4.3.2,4.4.2). Here, first relevant questions are how COIN programs/program instances can be identified (3.1.4) and how preferences for specific COIN program instances can be noted (3.1.5). It is then interesting to define how different COIN program can be coordinated (4.3.2,4.4.2), especially if there are program dependencies (4.2.2, cf. Section 7.2.2.3.1).

7.2.2.3.4. COIN dynamic system operation

In addition to static solutions to the described problems, the increasing dynamics of today's networks will also require dynamic solutions. For example, it might be necessary to dynamically change

COIN programs at run-time (4.3.3, 4.4.3) or to include new resources, especially if service-specific constraints or tenant requirements change (5.2.2). It will be interesting to see if COIN frameworks can actually support the sometimes required dynamic changes (3.2.4). In this context, providing availability and accountability of resources can also be an important aspect.

7.2.2.3.5. COIN system integration

COIN systems will potentially not only exist in isolation, but will have to interact with existing systems. Thus, there are also several questions addressing the integration of COIN systems into existing ones. As already described in Section 7.2.2.2.3, the semantics of changes made by COIN programs, e.g., filtering packets or changing payload, will have to be communicated to end-hosts (4.3.4,4.4.4). Overall, there has to be a common middleground so that COIN systems can provide new functionality while not breaking "legacy" systems. How to bridge different levels of "network awareness" (5.3.1) in an explicit and general manner might be a crucial aspect to investigate.

7.2.2.3.6. COIN system properties - optimality, security and more

A final category deals with meta objectives that should be tackled while thinking about how to realize the new concepts. In particular, devising strategies for achieving an optimal function allocation/ placement are important to effectively the high heterogeneity of the involved devices (3.2.3).

On another note, security in all its facets needs to be considered as well, e.g., how to protect against misuse of the systems, unauthorized traffic and more (5.3.4). We acknowledge that these issues are not yet discussed in detail in this document.

7.2.2.4. Applicability Area - Transport

The following research questions presented in the use cases belong to this category:

3.1.2

Further research questions concerning transport solutions are discussed in more detail in [TRANSPORT].

Today's transport protocols are generally intended for end-to-end communications. Thus, one important question is how COIN program interactions should be handled, especially if the deployment locations of the program instances change (quickly) (3.1.2).

7.2.2.5. Applicability Area - App Design

The following research questions presented in the use cases belong to this category:

4.3.1, 5.1.1, 5.1.3, 5.1.5

The possibility of incorporating COIN resources into application programs increases the scope for how applications can be designed and implemented. In this context, the general question of how the applications can be designed and which (low-level) triggers could be included in the program logic comes up (4.3.1). Similarly, providing sensible constraints to route between compute and network capabilities (when both kinds of capabilities are included) is also important (5.1.3). Many of these considerations boil down to a question of trade-off, e.g, between storage and frequent updates (5.1.5), and how (new) COIN capabilities can be sensibly used for novel application design (5.1.1).

7.2.2.6. Applicability Area - Data Processing

The following research questions presented in the use cases belong to this category:

3.2.3, 4.4.1, 4.5.2

Many of the use cases deal with novel ways of processing data using COIN. Interesting questions in this context are which types of COIN programs can be used to (pre-)process data (4.4.1) and which parts of packet information can be used for these processing steps, e.g., payload vs. header information (4.5.2). Additionally, data processing within COIN might even be used to support a better localization of the COIN functionality (3.2.3).

7.2.2.7. Applicability Area - Routing & Forwarding

The following research questions presented in the use cases belong to this category:

3.1.2, 3.1.3, 3.1.4, 3.1.5, 3.1.6, 5.1.2, 5.1.3, 5.1.4, 6.1.5,

Being a central functionality of traditional networking devices, routing and forwarding are also prime candidates to profit from enhanced COIN capabilities. In this context, a central question, also raised as part of the framework in Section 7.2.2.3.3, is how different COIN entities can be identified (3.1.4) and how the choice for a specific instance can be signalled (3.1.5). Building upon this, next questions are which constraints could be used to make the forwarding/routing decisions (5.1.3), how these constraints can be signalled in a scalable manner (3.1.3), and how quickly changing

COIN program locations can be included in these concepts, too (3.1.2).

Once specific instances are chosen, higher-level questions revolve around "affinity". In particular, how affinity on service-level can be provided (3.1.6), whether traffic steering should actually be performed on this level of granularity or rather on a lower level (5.1.4) and how invocation for arbitrary application-level protocols, e.g., beyond HTTP, can be supported (6.1.5). Overall, a question is what specific forwarding methods should or can be supported using COIN (5.1.2).

7.2.2.8. Applicability Area - (Industrial) Control

The following research questions presented in the use cases belong to this category:

3.2.4, 3.3.1, 3.3.4, 4.2.1, 4.4.1, 4.5.1

The final applicability area deals with use cases exercising some kind of control functionality. These processes, above all, require low latencies and might thus especially profit from COIN functionality. Consequently, the aforementioned question of function placement (cf. Section 7.2.2.3.2, e.g., close to one of the end-points or deep in the network, is also a very relevant question for this category of applications (3.3.1).

Focusing more explicitly on control processes, one idea is to deploy different controllers with different control granularities within a COIN system. On the one hand, it is an interesting question how these controllers with different granularities can be derived based on one original controller (4.2.1). On the other hand, how to achieve synchronisation between these controllers or, more generally, between different entities or flows/streams within the COIN system is also a relevant problem (3.3.4). Finally, it is still to be found out whether using COIN for such control processes indeed improves the existing systems, e.g., in terms of safety (4.5.1) or in terms of performance (3.2.4).

7.3. Requirements

To be added later.

8. Security Considerations

Note: This section will need consolidation once new use cases are added to the draft. Current in-network computing approaches typically work on unencrypted plain text data because today's networking devices usually do not have crypto capabilities.

As is already mentioned in Section 4.3.2, this above all poses problems when business data, potentially containing business secrets, is streamed into remote computing facilities and consequently leaves the control of the company. Insecure on-premise communication within the company and on the shop-floor is also a problem as machines could be intruded from the outside.

It is thus crucial to deploy security and authentication functionality on on-premise and outgoing communication although this might interfere with in-network computing approaches. Ways to implement and combine security measures with in-network computing are described in more detail in [I-D.fink-coin-sec-priv].

9. IANA Considerations

N/A

10. Conclusion

This draft presented use cases gathererd 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 (i) enable new experiences, (ii) expose new features or (iii) improve on existing system capabilities, and (iv) other use cases where COIN capabilities enable totally new applications, for example, in industrial networking.

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 to reap those opportunities. We also outlined possible requirements for realizing a COIN system addressing these use cases.

But of course this is only a snapshot of the potential COIN use cases. 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 (amonsts others). As these become better defined they will be added to the list presented here. We are, however, confident that our analysis across all use cases in those dimensions of opportunities, research questions, and requirements has identified commonalities that will support future work in this space. Hence, the use cases presented are directly positioned as input into the milestones of the COIN RG in terms of required functionalities.

11. List of Use Case Contributors

*Dirk Trossen has contributed the following use cases: Section 3.1, Section 5.1, Section 5.2, Section 6.1.

*Marie-Jose Montpetit has contributed the XR use case (Section 3.2).

*David Griffin and Miguel Rio have contributed the use case on performing arts (Section 3.3).

*Ike Kunze and Klaus Wehrle have contributed the industrial use cases (Section 4).

*Xavier De Foy has contributed the use case on virtual networks programming (Section 5.3)

12. References

12.1. 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/info/rfc2119>.

12.2. 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://www.ietf.org/internet-drafts/draft-sarathchandra-coin-appcentres-04.txt>.

[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/>.

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

[I-D.fink-coin-sec-priv] Fink, I. B. and K. Wehrle, "Enhancing Security and Privacy with In-Network Computing", Work in Progress, Internet-Draft, draft-fink-coin-sec-priv-03, 22 October 2021, <https://www.ietf.org/archive/id/draft-fink-coin-sec-priv-03.txt>.

[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://www.ietf.org/archive/id/draft-hsingh-coinrg-reqs-p4comp-03.txt>.

[I-D.mcbride-edge-data-discovery-overview]
McBride, M., Kutscher, D., Schooler, E., Bernardos, C. J., Lopez, D. R., and X. D. Foy, "Edge Data Discovery for COIN", Work in Progress, Internet-Draft, draft-mcbride-edge-data-discovery-overview-05, 1 November 2020, <https://www.ietf.org/archive/id/draft-mcbride-edge-data-discovery-overview-05.txt>.

[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://www.ietf.org/archive/id/draft-ravi-icnrg-5gc-icn-04.txt>.

[ICE] Burke, J., "ICN-Enabled Secure Edge Networking with Augmented Reality: ICE-AR.", ICE-AR Presentation at NDNCOM. , 2018, <https://www.nist.gov/news-events/events/2018/09/named-data-networking-community-meeting-2018>.

[KUNZE] 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>.

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

[PENNEKAMP] Pennekamp, J., Henze, M., Schmidt, S., Niemietz, P., Fey, M., Trauth, D., Bergs, T., Brecher, C., and K. Wehrle, "Dataflow Challenges in an Internet of Production: A Security & Privacy Perspective", Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy - CPS-SPC'19, DOI 10.1145/3338499.3357357, 2019, <https://doi.org/10.1145/3338499.3357357>.

[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>.

[SAPIO] Sapio, A., "Scaling Distributed Machine Learning with In-Network Aggregation", 2019, <https://arxiv.org/abs/1903.06701>.

[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>.

[TRANSPORT] Kunze, I., Wehrle, K., and D. Trossen, "Transport Protocol Issues of In-Network Computing Systems", Work in Progress, Internet-Draft, draft-kunze-coinrg-transport-issues-05, 25 October 2021, <https://www.ietf.org/archive/id/draft-kunze-coinrg-transport-issues-05.txt>.

[TS23.501] 501, 3gpp-23., "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] 502, 3gpp-23., "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>.

[TSN] "IEEE Time-Sensitive Networking (TSN) Task Group", <https://1.ieee802.org/tsn/>.

[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
Montreal
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