COINRG
Internet-Draft
Intended status: Informational
Expires: 21 August 2021

I. Kunze
K. Wehrle
RWTH Aachen
D. Trossen
Huawei
M.J. Montpetit
Concordia

17 February 2021

Use Cases for In-Network Computing
draft-irtf-coinrg-use-cases-00

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 is a best-effort network that offers limited guarantees regarding the timely and successful transmission of packets. Data manipulation and protocol functionality is generally provided by the end-hosts while the network is kept simple and only intended as a "store and forward" packet facility. This design-choice is suitable for a wide variety of applications and has helped in the rapid growth of the Internet. However, there are several domains which, e.g., demand a number of strict performance guarantees that cannot be provided over regular best-effort networks or require more closed loop integration to manage data flows. In this context, flexibly distributing the computation tasks across the network can help to achieve the guarantees and increase the overall performance.

However, different domains and different applications have different requirements and it remains unclear and the topic of academic research whether there can be a common solution to all COIN scenarios or if solutions have to be tailored to each scenario.

This document presents a series of applications and their requirements to illustrate the importance of COIN for realizing advanced applications. Based on these, the draft aims to create a taxonomy of elementary COIN scenarios with the goal of guiding future research work.

2. Terminology

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

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. Taxonomy

The use cases in this draft aim at outlining the specific capabilities that in-networking capabilities may bring to their realization. To attain this goal, we will use the following taxonomy to describe each of the use cases:

1. Description: explanation of the use case behavior

2. Characterization: explanation of the services that are being realized and the semantics of interactions in the use case

3. Existing solutions: if existing, outline current methods of realizing the use case

4. Opportunities and research questions for COIN: outline how PNDs may support or improve on the use case case in terms of performance and other metrics and state essential questions that are suitable for guiding research.

5. Requirements: describe the requirements for any solutions that may need development along the opportunities outlined in item 4

4. Industrial Use Cases

The industrial domain is characterized by diverse sets of requirements which often cannot be provided over regular best-effort networks. Consequently, there is a large number of specialized applications and protocols designed to give 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. In the Industrial Internet of Things (IIoT), however, more and more parts of the industrial production domain are interconnected. This increases the complexity of the industrial networks, makes them more dynamic, and creates more diverse sets of requirements. In these scenarios, solutions on the link layer alone are not sufficient.

The challenge is to develop concepts that can satisfy the dynamic performance requirements of modern industrial networks. COIN presents a promising starting point because it allows to flexibly distribute computation tasks across the network which can help to manage dynamic changes. As specifying general requirements for the industrial production domain is difficult due to the mentioned diversity, this document next characterizes and analyzes different scenarios to showcase potential requirements for the industrial production domain.

4.1. IIoT Network Scenario

Common components of the IIoT can be divided into three categories as illustrated in Figure 1. 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.

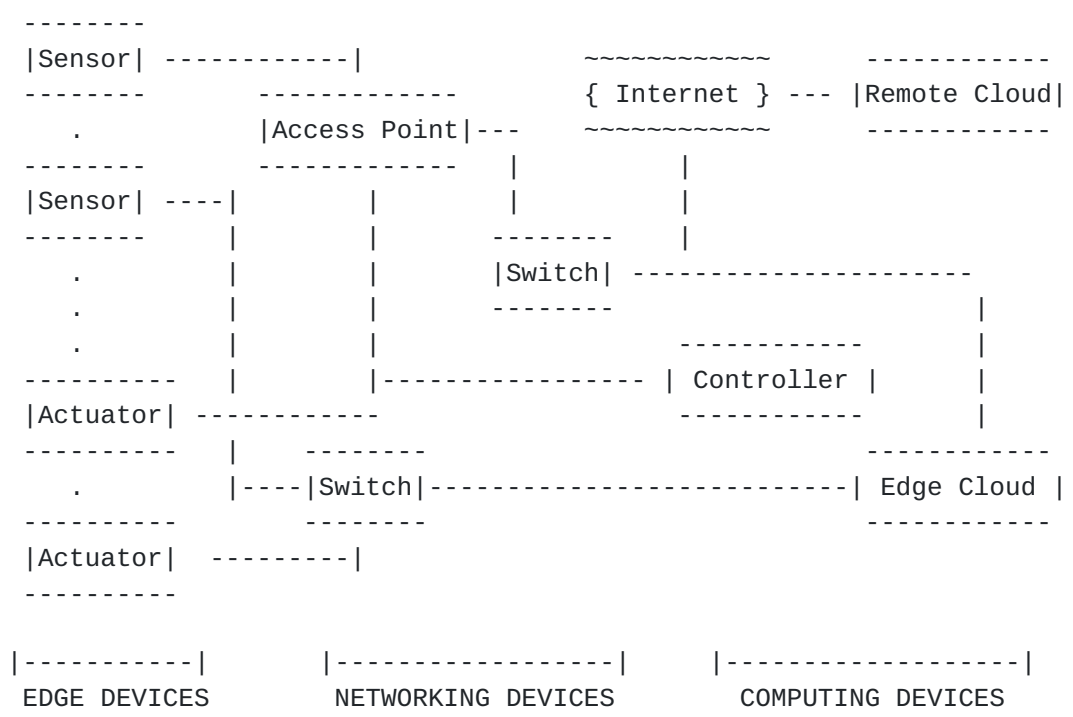

Figure 1: Industrial networks show a high level of heterogeneity.

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.

4.2.2. Characterization

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

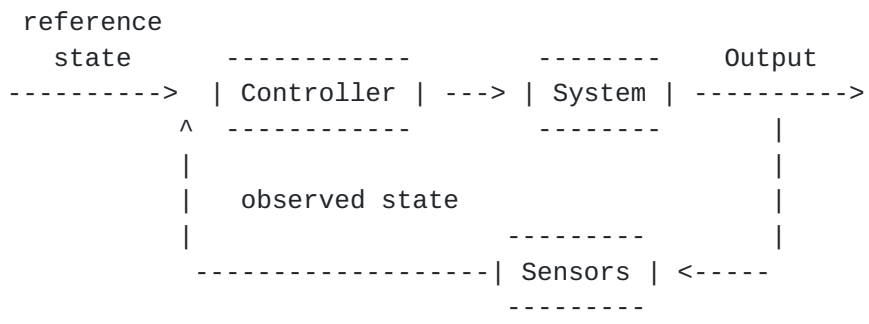

Figure 2: Simple feedback control model.

Apart from the control model, the quality of the control primarily depends on the timely reception of the sensor feedback, because the controller can only react if it is notified of changes in the system state. Depending on the dynamics of the controlled system, the control 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 off-premise cloud platforms are included.

4.2.3. Existing Solutions

Control functionality is traditionally executed on PLCs close to the machinery and these are only rarely upgraded. Further, the PLCs require vendor-specific implementations and are often hard to 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.

4.2.4. Opportunities and Research Questions for COIN

Control models, in general, can become involved but 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:

* Speed up control update rates by leveraging privileged position in the network

The problem, however, is that networking 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. 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:

* How can one derive the simplified versions of the overall controller?
  - How complex can they become?
  - How can one take the limited computational precision of networking devices into account when making them?
* How does one distribute the simplified versions in the network?
* How does the overall controller interact with the simplified versions?

4.2.5. Requirements

Req 4.2.1: Control approaches MUST provide stable/deterministic latencies.

Req 4.2.2: Control approaches SHOULD provide low latencies.

Req 4.2.3: The interaction between the in-network control function and the overall controller SHOULD be explicit.

Req 4.2.4: Actions of the control approaches SHOULD be explicit to the overall controller.

Req 4.2.5: Actions of the control approaches MUST be overridable by the overall controller.

4.3. Large Volume Applications

4.3.1. Description

In the IIoT, processes and machines can be monitored more effectively resulting in more available information. This data can be used to deploy machine learning (ML) techniques and consequently help to find previously unknown correlations between different components of the production 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. Similar to the considerations in Section 4.2, off-premise cloud platforms offer cost-effective solutions with a high degree of flexibility and scalability. While the unpredictable latency of the Internet is only a subordinate problem for this use case, moving all data to off-premise locations primarily poses infrastructural challenges.

4.3.2. Characterization

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 in-network computing techniques hardly applicable as they typically work on unencrypted data. Adding security to in-network computing 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 7.

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 and Research Questions for COIN

There are at least two concepts that might be suitable for reducing the amount of transmitted data in a meaningful way using in-network computing:

1. filtering out redundant or unnecessary data

2. aggregating data by applying pre-processing steps within the network

Both concepts require detailed knowledge about the monitoring infrastructure at the factories and the purpose of the transmitted data.

4.3.4.1. Traffic Filtering

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.

A trivial idea for reducing the amount of data is 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 would 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. 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 and stream filtering at line-rate

* Filtering based on packet header and payload, as well as multi-packet information

Challenges/Research Questions:

* How can traffic filters be designed?

* How can traffic filters be coordinated and deployed?

* How can traffic filters be changed dynamically?

* How can traffic filtering be signaled to the end-hosts?

4.3.4.2. In-Network (Pre-)Processing

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.

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 programmable networking devices is also interesting, because sensor data is usually streamed. In this context, the following research questions can be of interest:

Opportunities:

* Semantic data aggregation

* Computation across multiple packets and leveraging packet payload

Challenges/Research Questions:

* Which (pre-)processing steps can be deployed in the network?

  - How complex can they become?

* How can applications incorporate the (pre-)processing steps?

* How can the programming of the techniques be streamlined?

4.3.5. Requirements

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

Req 4.3.2: Filters or preprocessors MAY leverage packet header and payload information

Req 4.3.3: Filters or preprocessors SHOULD be reconfigurable at run-time

4.4. Industrial Safety

4.4.1. Description

Despite 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. The increased availability of sensor data and the detailed monitoring of the factories can help to build additional safety measures if the corresponding data is collected early at the correct position.

4.4.2. 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. Yet, it is worthwhile to investigate whether such solutions can introduce additional safety measures.

4.4.3. Existing Solutions

Note: Will be added later.

4.4.4. Opportunities and Research Questions for COIN

Software-based solutions can take advantage of the large amount of available sensor data. Different safety indicators within the production hall can be combined within the network so that programmable networking devices can give early responses if a potential safety breach is detected. A rather simple 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:

* Early emergency reactions based on diverse sensor feedback

Research Questions:

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

4.4.5. Requirements

Req 4.4.1: COIN-based safety measures MUST NOT degrade existing safety measures.

Req 4.4.2: COIN-based safety measures MAY enhance existing safety measures.

5. Immersive Experiences

5.1. Mobile Application Offloading

5.1.1. 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 micro-services and deployed in an app-centric execution environment, only the "display" micro-service is left in the headset, while the compute intensive real-time VR content processing micro-services can be offloaded to a nearby resource rich home PC, for a better execution (faster and possibly higher resolution generation).

5.1.2. Characterization

Partitioning an application into micro-services allows for denoting the application as a collection of functions for a flexible composition and a distributed execution, e.g., 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 micro-services of an application and expose them to the execution environment. When the micro-service sequence is executed on a single device, the outcome is what you see today as applications running on mobile devices. However, the execution of functions may be moved to other (e.g., more suitable) devices which have exposed the corresponding micro-services to the environment. The result of the latter is flexible 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).

Figure 3 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) micro services) over an SDN network. The packaged applications are made available through a localized 'playstore server'. The application installation is realized as a 'service deployment' process, combining the local app installation with a distributed micro-service deployment (and orchestration) on most suitable AppCentreS ('processing server').

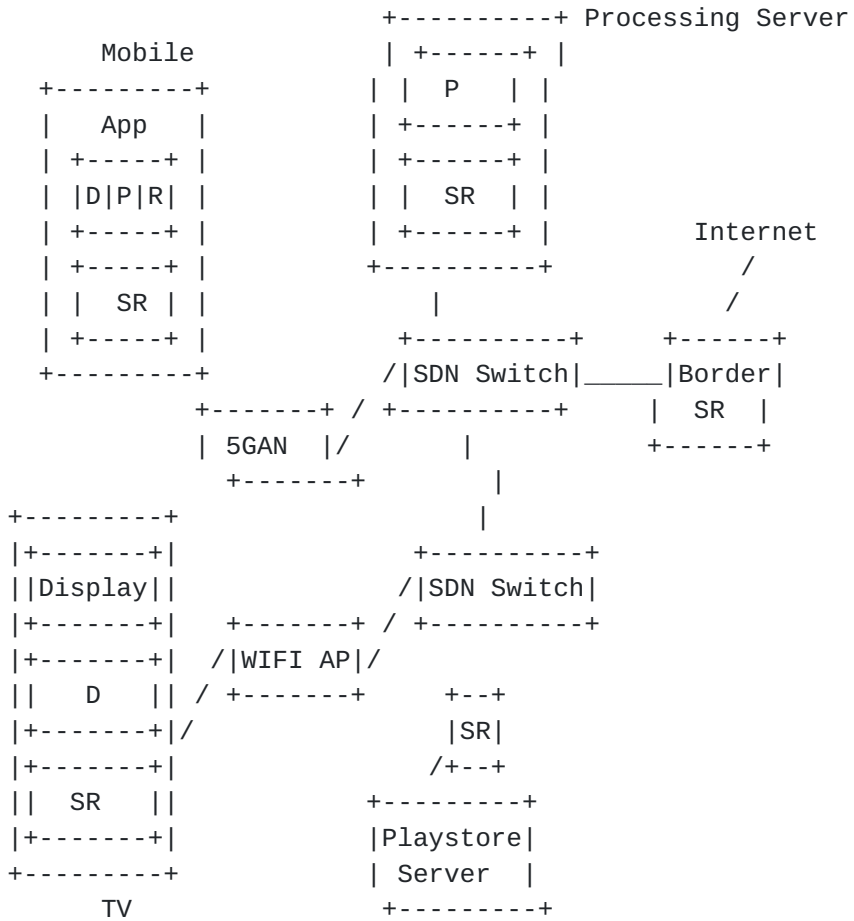

Figure 3: Application Function Offloading Example.

Such localized deployment could, for instance, be provided by a visiting site, such as a hotel or a theme park. Once the 'processing' micro-service is terminated on the mobile device, the 'service routing' (SR) elements in the network routes requests to the previously deployed 'processing' micro-service running on the processing server' AppCentre over an existing SDN network. As an extension to the above scenarios, we can also envision that content from one processing micro-service may be distributed to more than one display micro-service, e.g., for multi/many-viewing scenarios.

5.1.3. Existing Solutions

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

5.1.4. Opportunities and Research Questions for COIN

Opportunities:

* execution of app-level micro-services (service deployment in [APPCENTRES])

* supporting service-level routing of requests (service routing in [APPCENTRES])

* support the constraint-based selection of a specific service instance over others (constraint-based routing in [APPCENTRES])

Research Questions:

* How to combine service-level orchestration frameworks with app-level packaging methods?

* How to reduce latencies involved in micro-service interactions where service instance locations may change quickly?

* How to signal constraints used for routing in a scalable manner?

* How to provide constraint-based routing decisions at packet forwarding speed?

* What in-network capabilities may support the execution of micro-services?

5.1.5. Requirements

Req 5.1.1: Any app-centric execution environment MUST provide means for routing of service requests between resources in the distributed environment.

Req 5.1.2: Any app-centric execution environment MUST provide means for dynamically choosing the best possible micro-service sequence (i.e., chaining of micro-services) for a given application experience. Means for discovering suitable micro-service SHOULD be provided.

Req 5.1.3: Any app-centric execution environment MUST provide means for pinning the execution of a specific micro-service to a specific resource instance in the distributed environment.

Req 5.1.4: Any app-centric execution environment SHOULD provide means for packaging micro-services for deployments in distributed networked computing environments. The packaging MAY include any constraints regarding the deployment of service instances in specific network locations or compute resources. Such packaging SHOULD conform to existing application deployment models, such as mobile application packaging, TOSCA orchestration templates or tar balls or combinations thereof.

Req 5.1.5: Any app-centric execution environment MUST provide means for real-time synchronization and consistency of distributed application states.

5.2. Extended Reality (XR)

5.2.1. Description

Virtual Reality (VR) and Augmented Reality (AR) taken together as Extended Reality (XR) are at the center of a number of advances in interactive technologies. 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.

5.2.2. Characterization

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. Thus, XR is difficult to deliver with a client-server cloud-based solution as it requires 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. Many XR services that involve 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. Hence, XR uses recent advances in in-network programming, distributed networks, orchestration and resource discovery to support the XR advanced immersive requirements. 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. This includes computing in the nodes from content source to destination.

5.2.3. Existing Solutions

Related XR or XR-enabling solutions using in-network computation or related technologies include:

* 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 "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.

5.2.4. Opportunities and Research Questions for COIN

Opportunities:

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. The opportunities include:

* 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.

Research Questions:

There is a need for more research resource allocation problems at the edge to enable interactive operation and quality of experience in VR. These include multi-variate and heterogeneous goal optimization problems requiring advanced analysis. Image rendering and video processing in XR leverages different HW capabilities combinations of CPU and GPU. Research questions include:

* 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?

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

* 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?

* Can COIN improve the dynamic distribution of control, forwarding and storage resources and related usage models in XR?

5.2.5. Requirements

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. Since XR deals with personal information and potentially protected content XR 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. The requirements can be summarized as:

Req 5.2.1: Allow joint collaboration.

Req 5.2.2: Provide multi-views.

Req 5.2.3: Include extra streams dynamically for data intensive services, manufacturing and industrial processes.

Req 5.2.4: Enable multistream, multidevice, multidestination applications.

Req 5.2.5: Use new Internet Architectures at the edge for improved performance and performance management.

Req 5.2.6: Integrate with holography, 3D displays and image rendering processors.

Req 5.2.7: All the use of multicast distribution and processing as well as peer to peer distribution in bandwidth and capacity constrained environments.

Req 5.2.8: Evaluate the integration local and fog caching with cloud-based pre-rendering.

Req 5.2.9: Evaluate ML-based congestion control to manage XR sessions quality of service and to determine how to priortize data.

Req 5.2.10: Consider higher layer protocols optimization to reduce latency especially in data intensive applications at the edge.

Req 5.2.11: Provide trust, including blockchains and smart-contracts to enable secure community building across domains.

Req 5.2.12: Support nomadicity and mobility (link to mobile edge).

Req 5.2.13: Use 5G slicing to create independent session-driven processing/rendering.

Req 5.2.14: Provide performance optimization by data reduction, tunneling, session virtualization and loss protection.

Req 5.2.15: Use AI/ML for trend analysis and data reduction when appropriate.

5.3. Personalised and interactive performing arts

5.3.1. 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.).

5.3.2. Characterization

There are several chained functional entities which are candidates for in-network processing.

* 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 in-network processing 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 to audience preferences and requirements makes it unfeasible for this to be done at the performer premises: it will require large amounts of processing

power to process individual personalised streams as well as large amounts of network bandwidth to transmit personalised streams to each viewer.

* rendering of VR headset content to follow viewer head movements has an upper bound on lag to maintain viewer QoE.

* 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 the live and interactive aspects that require the deviation from the direct path between performers and audience is minimised, reducing the opportunity to leverage large-scale processing capabilities at centralised data-centres.

5.3.3. Existing solutions

To be added.

5.3.4. Opportunities and Research Questions for COIN

Opportunities:

* See Characterization.

Research Questions:

* Where should the aggregation, encoding and personalisation functions be located? Close to the performers or close to the audience members?

* How far away from the direct network path from performer to audience can they be located, considering the latency implications of path-stretch and the availability of processing capacity?

* 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?

5.3.5. Requirements

The chain of functions 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).

6. Infrastructure Services

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. Other solutions may desire the localization of reasoning logic, e.g., for deriving attributes that better preserve privacy of the utilized raw input data.

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 and Research Questions for COIN

Opportunities:

* supporting service-level routing of requests (service routing in [APPCENTRES])

* support the constraint-based selection of a specific service instance over others (constraint-based routing in [APPCENTRES])

* collective communication between multiple instances of AI services

Research Questions:

* similar to use case in Section 5.1

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

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

* What in-network capabilities may support the collective communication patterns found?

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

6.1.5. Requirements

Req 6.1.1: Any app-centric execution environment MUST provide means to specify the constraints for placing (AI) execution logic in certain logical execution points (and their associated physical locations).

Req 6.1.2: Any app-centric execution environment MUST provide support for app/micro-service specific invocation protocols.

6.2. Content Delivery Networks

6.2.1. 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.

6.2.2. Characterization

From the perspective of this draft, a CDN can be interpreted as a (network service level) application with 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.

6.2.3. 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.

6.2.4. Opportunities and Research Questions for COIN

Opportunities:

* supporting service-level routing of requests (service routing in [APPCENTRES])

* support the constraint-based selection of a specific service instance over others (constraint-based routing in [APPCENTRES])

* supporting Layer 2 capabilities for multicast (compute interconnection and collective communication in [APPCENTRES])

Research Questions: in addition to those for Section 5.1,

* how to utilize L2 multicast to improve on CDN designs? How to utilize in-network capabilities in those designs?

* what forwarding methods may support the required multicast capabilities (see [FCDN])

* how could storage be traded off against frequent, multicast-based, replication (see [FCDN])

* what scalability limits exist for L2 multicast capabilities? How to overcome them?

6.2.5. Requirements

Req 6.2.1: Any app-centric execution environment SHOULD utilize Layer 2 multicast transmission capabilities for responses to concurrent service requests.

6.3. CFaaS

6.3.1. Description

App-centric execution environments, consisting of Layer 2 connected data centres, provide the opportunity for infrastructure providers to offer CFaaS type of offerings to application providers. Those app providers utilize the compute fabric exposed by this CFaaS offering for the purposes defined through their applications. In other words, the compute resources can be utilized to execute the desired micro-services of which the application is composed, while utilizing the inter-connection between those compute resources to do so in a distributed manner.

6.3.2. 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.

6.3.3. Existing Solutions

NOTE: material on solutions will be added here later

6.3.4. Opportunities and Research Questions for COIN

Opportunities:

* supporting service-level routing of requests (service routing in [APPCENTRES])

* support the constraint-based selection of a specific service instance over others (constraint-based routing in [APPCENTRES])

* supporting Layer 2 capabilities for multicast (compute interconnection and collective communication in [APPCENTRES])

Research Questions: similar to those for Section 5.1, in addition

* how to convey app-specific requirements for the creation of the L2 fabric?

* how to dynamically integrate resources, particularly when driving by app-level requirements and changing service-specific constraints?

* how to utilize in-network capabilities to aid the availability and accountability of resources?

6.3.5. Requirements

Req 6.3.1: Any app-specific execution environment SHOULD expose means to specify the requirements for the tenant-specific compute fabric being utilized for the app execution.

Req 6.3.2: Any app-specific execution environment 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.

Req 6.3.3: Any app-specific execution environment 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 6.3.4: Any app-specific execution environment MUST provide means for ensuring availability and usage of resources is accounted for.

7. 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].

8. IANA Considerations

N/A

9. Conclusion

There are several domains that can profit from COIN.

Industrial scenarios have unique sets of requirements mostly focusing around tight latency constraints with high required bandwidths.

NOTE: Further aspects will be added once more use cases are added to the draft.

10. List of Use Case Contributors

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

* Dirk Trossen has contributed the following use cases: Section 5.1, Section 6.1, Section 6.2, Section 6.3.

* Marie-Jose Montpetit has contributed the XR use case (Section 5.2).

* David Griffin and Miguel Rio have contributed the use case on performing arts (Section 5.3).

11. References

11.1. Normative References

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

11.2. Informative References

[APPCENTRES]
Trossen, D., Sarathchandra, C., and M. Boniface, "In-Network Computing for App-Centric Micro-Services", Work in Progress, Internet-Draft, draft-sarathchandra-coin-appcentres-04, 26 January 2021, <http://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 52nd 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.fink-coin-sec-priv]
Fink, I. and K. Wehrle, "Enhancing Security and Privacy with In-Network Computing", Work in Progress, Internet-Draft, draft-fink-coin-sec-priv-01, 8 September 2020, <http://www.ietf.org/internet-drafts/draft-fink-coin-sec-priv-01.txt>.

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

[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>.

[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