Budapest University of Technology and Economics

Faculty of Electrical Engineering and Informatics

Department of Telecommunications and Media Informatics

Utilizing AI in 5G Edge Networks

Scientific Students' Association Report

Author:

Marton Aron Horvath

Supervisor at the department: Dr. Gábor Fehér, PhD, associate professor

External supervisor: Dr. Zsigmond Pap, PhD, Ericsson Hungary ltd.

Budapest, 2022

Abstract

As society is driving toward the information age, the telecommunications network, a fundamental component of this change, is undergoing rapid and profound changes. Due achieve these transformations, novel approaches such as Artificial Intelligence (AI) are required. Without the efficient support of the artificial intelligence systems networks would not be manageable since the complexity of information technology systems had already surpassed the level at which human operators could build and maintain them.

Artificial intelligence has made significant advances in various fields, including speech processing, image classification, and drug discovery. The explosive growth of data advances in machine learning (especially Deep Learning). Easy access to powerful computing resources compelling this trend. The widespread deployment of edge devices (such as IoT devices) generates an unprecedented amount of data, which allows for the development of accurate models and various intelligent applications at the network edge. However, that amount of data cannot be sent to the cloud for processing, due to the varying channel quality, traffic congestion, and privacy concerns, as well as enormous energy consumption.

Aside from cloud-based solutions, AI on edge devices (Edge AI) offers the benefits of rapid response with low latency, high privacy, increased robustness, and better network bandwidth utilization. New use cases for the edge are made possible by 5G. A crucial part of the 3GPP 5G core network, the User Plane Function (UPF) represents the data plane evolution of the Control and User Plane Separation (CUPS) strategy. The edge user plane is essential for maintaining company data on-premises and providing reduced latency.

In real-time applications, AI should be placed in the Edge and tightly integrated with the UPF application. Edge computing theory and techniques can significantly bridge the capacity of the cloud and the requirement of devices by the network edges. That can accelerate content delivery and improve the quality of mobile services.

In my study, I built experimental Edge Computing systems with different architectures, consisting of both tightly and loosely integrated AI solutions. I've done measurements on different architectures, investigated the improvement opportunities, and demonstrated the operation. The System consists of a traffic generator, an Ericsson UPF application, an AI component, and software playing the role of the Edge Application.

The most crucial part of the architectural structure is the location of the AI module, which could be placed in a Service Function as part of the 5G UPF or integrated into the source or the end application. The main goal of my investigation was to find the answer to the question: How could I achieve the ideal architecture?

Kivonat

Ahogy a társadalom az információs kor felé tart, a telekommunikációs hálózatok, amik alapvető elemei ennek a változásnak, gyors és mély változásokon mennek át. Ahhoz, hogy elérjék ezt a változást, újszerű megközelítésekre van szükség, mint például a Mesterséges Intelligencia (MI). A Mesterséges Intelligencia rendszerek hatékony hozzájárulása nélkül az információs technológiai rendszerek nem lehetnének menedzselhetőek, mert komplexitásuk már túlmutatott azon a szinten, amit az operátorok fel tudnának építeni és karban tudnának tartani.

A Mesterséges Intelligencia több területen is jelentős változáson ment keresztül, beleértve a beszéd feldolgozást, a képfelismerést, a gyógyszerek felfedezését. A megnövekedő adat a gépi tanulás (főként a mélytanulás) felé vitt bennünket. Az egyszerű hozzáférés az erős számítási erőforrásokhoz szintén erősítette ezt a trendet. A széleskörű edge eszközök jelenléte (mint az IoT eszközök) eddigieknél nagyobb adatmennyiséget tettek elérhetővé, ami által különböző pontos modelleket tudunk létrehozni az edge hálózatokban. Habár ekkora mennyiségű adatot már nem tudunk a felhőbe feldolgozásra küldeni, a különböző csatorna minőségek, torlódások és biztonsági megfontolások miatt, mint ahogy a magas energiafogyasztás miatt sem.

A felhő alapú megoldások mellet, a Mesterséges Intelligencia edge eszközökön való alkalmazása (Edge AI) előnyként nyújtja a gyors válaszidőt, a kis késleltetést, megnövekedett robusztusságot és a jobb hálózati kihasználtságot. Új felhasználási területek váltak elérhetővé az 5G megjelenésével. A 3GPP által meghatározott 5G core networknek meghatározó része a User Plance Function (UPF) valósítja meg a data plane-t, a kialakult Control and User Plane Separation (CUPS) stratégia során. Valós idejű alkalmazások esetén az MI-nek az edgben kell elhelyezkednie szorosan integrálva az UPF alkalmazással. Az Edge Computing elmélete és technikái lehetnek azok, amik biztosítják mind a felhő alapú számítástechnika kapacitását és az edge hálózaton lévő eszközök igényét is egyaránt. Ez felgyorsíthatja a tartalomszolgáltatást és a mobil szolgáltatások minőségét is javíthatja.

A dolgozatomban kísérleti Edge Computing rendszereket építettem, melyek közé tartoznak egyaránt szorosan és lazábban integrált MI megoldások. Mérésekkel végeztem a különböző architektúrákon, megvizsgáltam a fejlesztési lehetőségeket, és demonstráltam a működést. A rendszer tartalmaz egy forgalom generátort, az Ericsson UPF funkcióját, egy MI komponenst és egy szoftver helyettesíti az Edge alkalmazás funkcióját.

A legjelentősebb része az architektúrának az az MI modul elhelyezkedése a rendszerben, ez a modul elhelyezkedhet az 5G UPF-ben mint egy Service Function, vagy csatolható a forráshoz vagy a végponthoz is. A fő kérdés, amit meg szerettem volna válaszolni az az, hogy: Hogyan tudom elérni az ideális architektúrát.

Contents

1. Introduction

A new revolution is taking place in communication technologies. 5G opens a world of possibilities for social digitalization and industrial connectivity. On top of the physical infrastructure, the service-oriented end-to-end network slicing architecture in the 5G era may meet a diversity of service requirements.

Cloudification, which entails the transition from traditional hard box network functions to an all-on-cloud management plane, is a key feature of the network slicing architecture. The cloud refers not only to regional data centers but also to edge-cloud servers located near mobile service subscribers. With the rise of the Internet of Things (IoT), more data is generated by widely distributed and geographically dispersed mobile and IoT devices, likely more than by mega-scale cloud data centers.

According to Ericsson's prediction, IoT devices will generate 45 percent of the 40-ZB global Internet data in 2024. [1] Although these predictions are focused on Massive IoT, the new generation of mobile technologies also provides many opportunities for the industrial sector. One word may summarize the Industry 4.0 revolution: connectivity. With the proliferation of the Industrial Internet of Things (IIoT), cloud, and big data, connectivity will permit intelligent production. Smart devices can collect information such as indoor and outdoor location, status information, client usage patterns, and so forth. They are capable of not just gathering knowledge but also sharing it with their intended peers. This will aid in the development of efficient production processes in production environments, as well as scheduled preventative maintenance on machines.

Another advantage is detecting faults in the production chain as soon as feasible, which is critical for lowering production and maintenance costs. Sector 4.0 also focuses on solving optimization problems in the industry by utilizing data-driven services via smart devices, like Automated Guided Vehicles (AGVs), also known as an Autonomous Mobile Robots (AMRs) which are evolved into the key component of organized modern intralogistics. [2] They are portable robots, they use radio waves, vision cameras, magnets, or lasers to navigate along defined long lines or cables on the floor. To avoid obstacles, accidents, and injuries, they employ sensors like Light Detection and Ranging (Lidar) and cameras. Complex job sharing, data-driven decision making, and remote access to machinery are all possible with Industry 4.0 and IIoT. Because of the massive interconnectedness of things and their ability to collect and share data, security is a crucial necessity for IIoT and Industry 4.0 concepts.

Artificial Intelligence (AI) and Machine Learning (ML) products were used more widely in Industry 4.0 as big data and data streaming technologies improved. AI and machine learning applications are gaining traction in fields such as health, education, defense, security, and industry. Many tech giants are putting billions of dollars towards AI goods like self-driving cars. Pedestrian identification, collision detection, and other computer vision-based self-driving algorithms are being developed by Google, NVIDIA, and others.

Another question is where the AI should be installed and where the calculations should be performed. Edge networking is a complex and dynamic computing architecture aimed at bringing cloud resources closer to end-users, enhancing responsiveness, and lowering backhaul traffic. Decentralizing data centers and utilizing smart devices and network gateways to perform services on behalf of cloud computing are two strategies to create an edge computing network.

In my paper, I combine most of the technologies mentioned above to develop a solution for AI implementation in 5G edge networks.

2. Technological background

The technological background section is separated into three areas; the first part is an introduction to the 5G architecture's principles, highlighting the Mobile Network Gateway Function and the goal I aimed to accomplish. Then describe what Edge- and Distributed AI is, the main idea of my work based on these concepts. Lastly, the new approach to the artificial intelligence layer, and actual trends, where the state of the art nowadays.

2.1. Mobile Network Gateway Function

Without a foundational understanding of the 5G architecture, it is impossible to comprehend the mobile network's gateway function and its purpose. Such a gateway application is a User Plane Function (UPF), Figure 2.1-1 shows the main components of the 5G core network, that is the architecture since *3rd Generation Partnership Project* (*3GPP*) REL-15. [3] User Equipment (UE), for example a mobile phone, maintains the connection with Radio Access Network (RAN), data flows from RAN to UPF, which is functioning as a router, and from UPF the data flows to Data Network (DN). The flow of data from RAN to DN is the User Plane connectivity. Each instance of this connection is called in 5G as Protocol Data Unit (PDU) Session, they are unique to the device. Within the PDU Session, Quality of Service (QoS) is achieved by creating separate QoS Flows which are uniquely identified with QoS Flow ID. [4]

That is controlled by the Control Plane (CP), in that, Access and Mobility Management Function is responsible for maintaining the radio connection, Session Management Function (SMF) for interacting with the decoupled data plane, creating updating, and removing Protocol Data Unit (PDU) sessions and managing session context with the User Plane Function (UPF). Unified Data Management (UDM) has a huge database, which stores UICC card data and service policies.

In our case, the most important is UPF, it has several interfaces (N3, N4, N6), and N9 is to cascade UPFs together if needed. In N3 flows the data, radio packets encapsulated with GTP-U protocol, N6 forward it towards to internet, N4 maintain the connection with the control plane. When more UPF are required, they can be chained with the N9 interface. For example, in 5G, there is no separation between Packet Network Data Gateway (PGW), which handles all IP packet-based functions (Deep Packet Inspection, UE IP address allocation, etc.), and Serving Gateway (SGW), which is in charge of packet forwarding and routing, downlink packet buffering, etc. as there is in 4G. Hence chaining more UPF could be the solution.

Service Function Chaining (SFC), also known as Network Service Chaining, is a method that makes use of the capabilities offered by Software Defined Networks (SDN) to create composed network services, which are a connected and ordered sequence of Service Functions (SF) that must be applied to a particular packet and/or flow that has been previously classified. L2-7 firewalls, Deep Packet Inspection (DPI), video optimizers, load balancing servers, and Network Address Translation (NAT), among others, are some examples of SFs. From the perspective of the network operator, SFC provides tenants and network users with collections or suites of network services with various criteria and features that can be dynamically and automatically deployed on demand into the multi-tenant infrastructure. Albeit Service Chaining originally has not developed for our use-case, it has the ability for chaining the solution, hence chained services like Artificial Intelligence Module apparently should be implemented in the N6 interface.

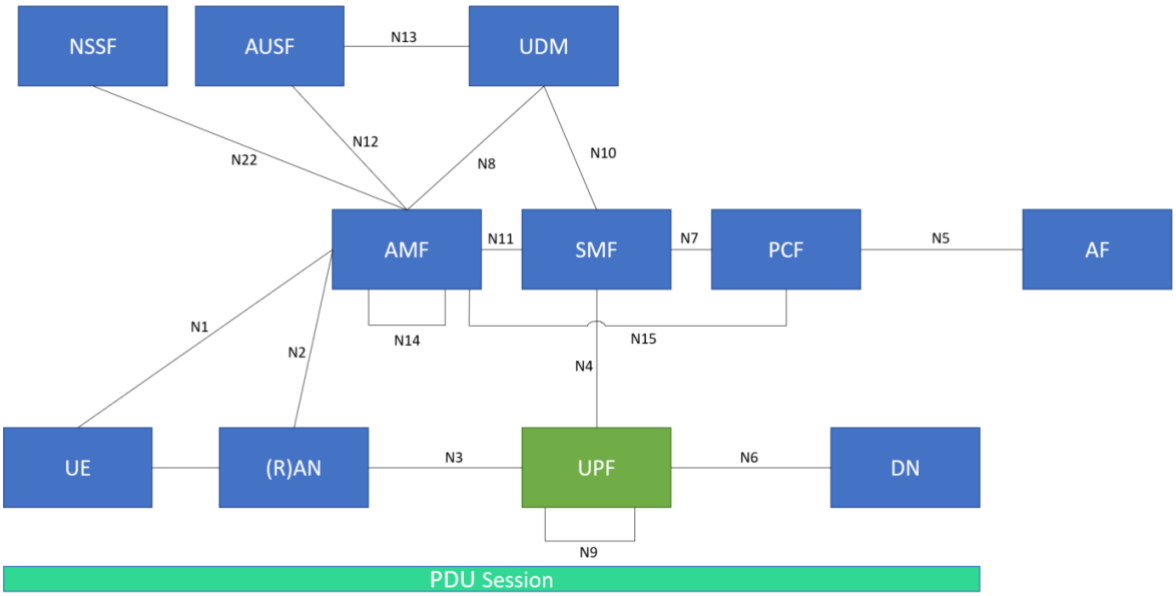

Figure 2.1-1 UPF in 5G network

In real-time systems, Artificial Intelligence implementation on the Edge could be beneficial, in our case the exact environment is the Edge UPF. In that case we can not only assist the other services running there in swiftly evaluating our data, but we can also have a direct impact on the mobile network.

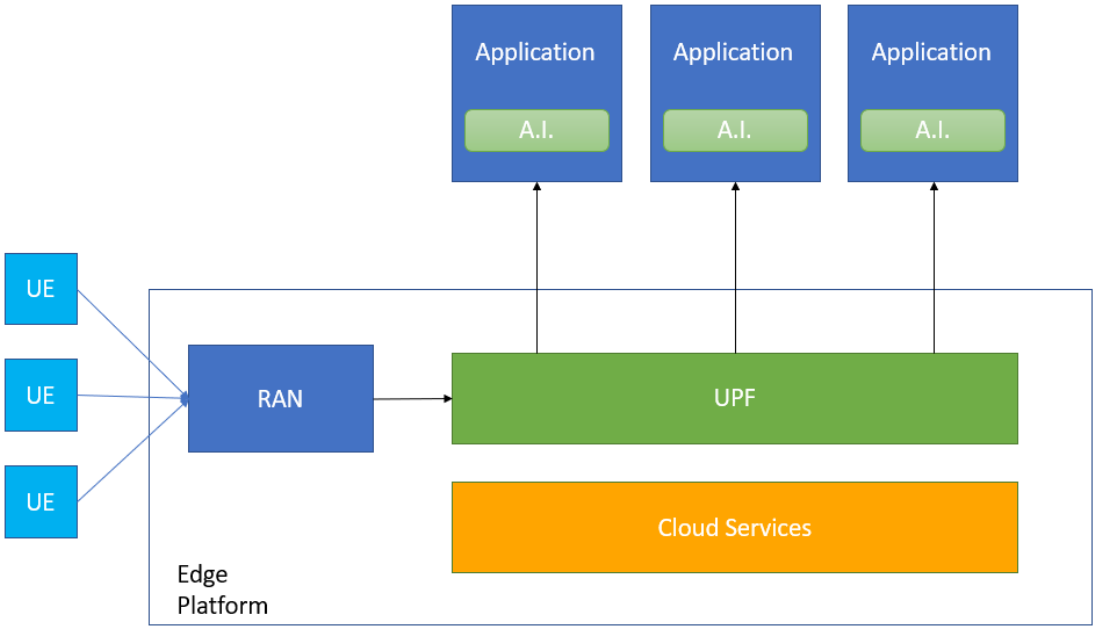

Figure 2.1-2 Traditional implementation of AI

Consider the following scenario. An Autonomous Guided Vehicle (AGV) with a 5G private network connection operates in a factory. The robot control program controls it from an edge-cloud located on the production site. In the factory, a radio network works, with signals received by a UPF and communicated to the robot controller. The UPF and the robot controller are both hosted on a local cloud.

A camera is installed on the AGV. This video stream is also sent to the edge, where it is processed by artificial intelligence, most likely on a mobile network. The goal is to properly process the camera so that the AGV can operate safely and without causing any accidents.

This AI must be part of the robot controller application in the traditional solution (Figure 2.1-2). This, however, has significant disadvantages. For example, the ominous camera is now only suited for this one function, and its replacement with one from another manufacturer could be difficult or to modify it to the right - for example, with HW acceleration - according to the AI module. Furthermore, if the AGV and the camera are handled by different firms, complications will occur.

I propose a new architecture setup to enable AI at the network edge in an efficient way as shown in Figure 2.1-3. The UPF, acting as a router, could segregate the control and camera signals from the AGV and send the former to the robot controller. A stand-alone service, which is the AI, receives the signal from the camera and delivers functionality. It provides an API that the robot controller can find, connect to, and use to acquire the desired outcome from the AGV's operation.

It is possible that the same signal stream is transmitted by several such AI services, each of which wants to process it, or that the stream is terminated by an application. For instance, by merging the data from several cameras to create a free-viewpoint video first, it would enable

the AI to pinpoint not just the incidence but also the exact location of the fault in the case of an issue. Before it can be used on a SCADA system, the result is released in English and then translated into Spanish by a third A.I. This necessitates the chaining ability of the processing entities.

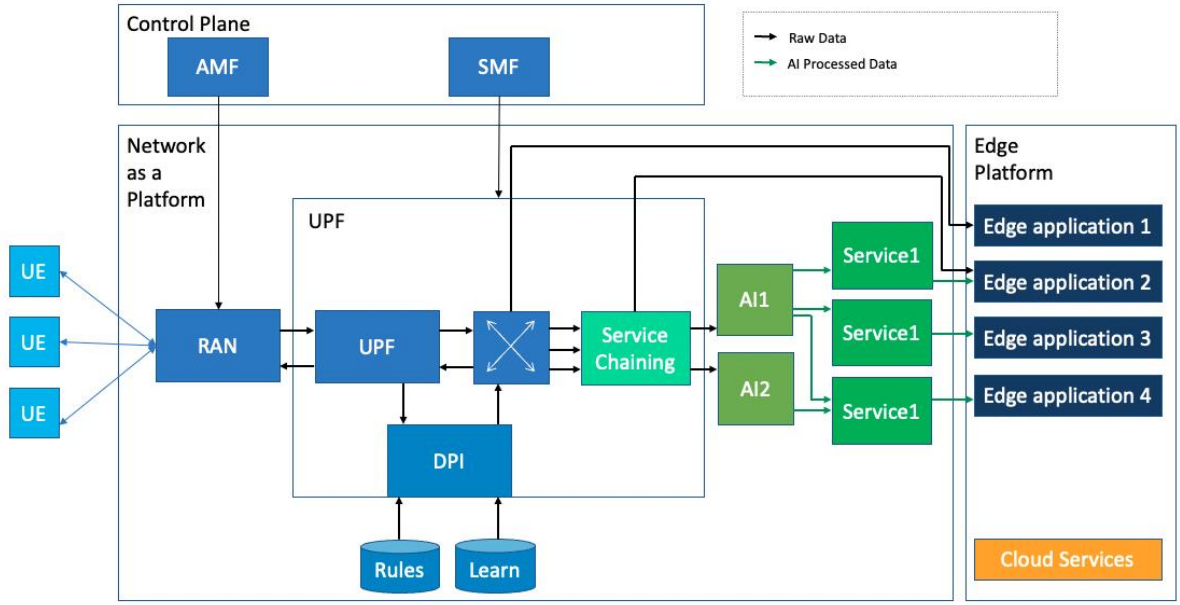

Figure 2.1-3 Architecture of planned Edge AI solution

The UPF application, can detect each data stream independently and route it to the proper service if necessary. If the data stream returns to the UPF after processing, the UPF can determine which service should be chained next until the data reaches its destination. As next topic I introduce edge- and distributed AI, since that is the main idea of my application.

2.2. Edge- and distributed AI

From the beginning of computer science, one of the main goals has been Artificial Intelligence (AI). Indeed, it is arguable that computer science's ultimate goal is to replicate, if not outperform, human intelligence.

The algorithms and models used are frequently executed in remote or local data centers (inference and/or training phases). While some intelligence is common in locally executed applications, many are limited to very specific functions and rely on proprietary models built into the application. [5]

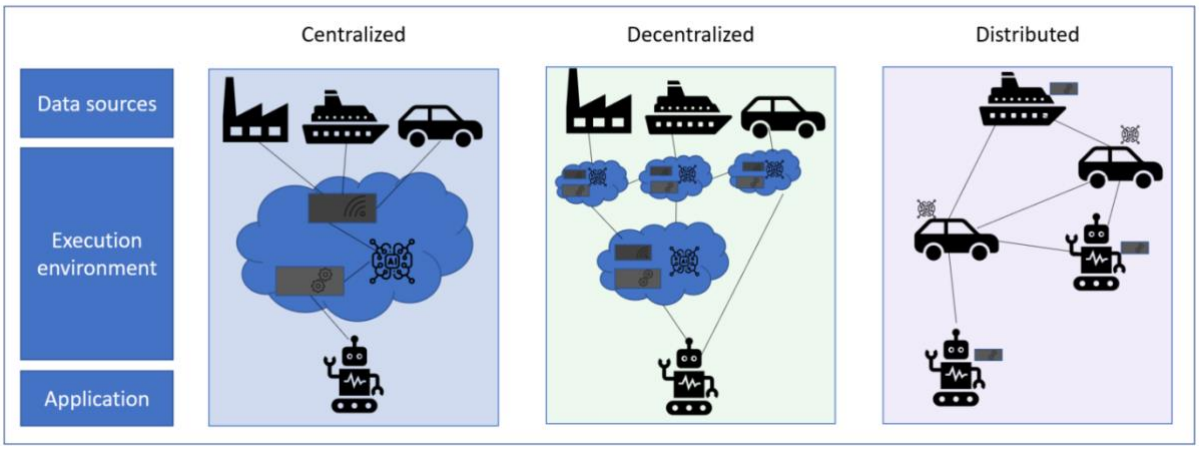

Figure 2.2-1 From centralized to decentralized AI deployments

As Figure 2.2-1 shows, AI could be implemented in different architectures. The most common solution is the centralized version, where everything is calculated in the cloud. In the decentralized architecture, the central control component gets kept, but some of the calculations happened on the edge. The distributed solution is peer-to-peer, the central unit is omitted here, and there is no dedicated node, they work cooperatively, and are organized by ad hoc methods like Intelligent Transport Systems.

The most obvious use case when we need to distribute the intelligence of the application is Internet of Thing (IoT). The IoT is a vision of a society in which every digital device may provide information that can be consumed locally and globally. When IoT started to emerge, experts predicted that in 2020, the Internet of Things will have connected more than 50 billion devices, so trends started to prepare for that huge number of nodes. [6] Although the numbers did not accomplish the predictions, the amount of IoT devices is tremendous and it constantly growing. They will generate vast data with characteristics such as increased size, velocity, modes, data quality, and heterogeneity. Meanwhile, 5G network evolution is becoming a major driving force for IoT growth. 5G is predicted to have more coverage, better throughput, lower latency, and huge capacity connection density [7], paving the path for billions of sensors to be connected over the Internet. Additionally, various prospective approaches and technologies, like as millimeter-wave (mmWave), massive multiple-input multiple-output (MIMO), and machine-to-machine (M2M) communications, have been proposed to help 5G networks adapt to IoT.

As a result, homogeneous and heterogeneous sensor networks [8] can connect enormous sensing equipment and contribute significantly to the provision of enhanced and intelligent services for humans. Data is intended to be saved or recorded for analysis, there will be a massive amount of data that needs to be processed automatically. As a result, machines, rather than humans, will interact with other machines to deliver services, develop added value, and operate based on data-analytics outcomes. Optimization, prediction, anomaly detection, and other functions are included in the processing of machine-generated data. [8] [9]

Why distribute intelligence processing a legitimate question?

It's reasonable to presume that distribution is advantageous in specific situations, such as local-data-centric solutions, scenarios requiring the usage of online learning, and settings with limited connectivity. However, it might be claimed that centralized data collection and processing systems may be applied to practically any situation.

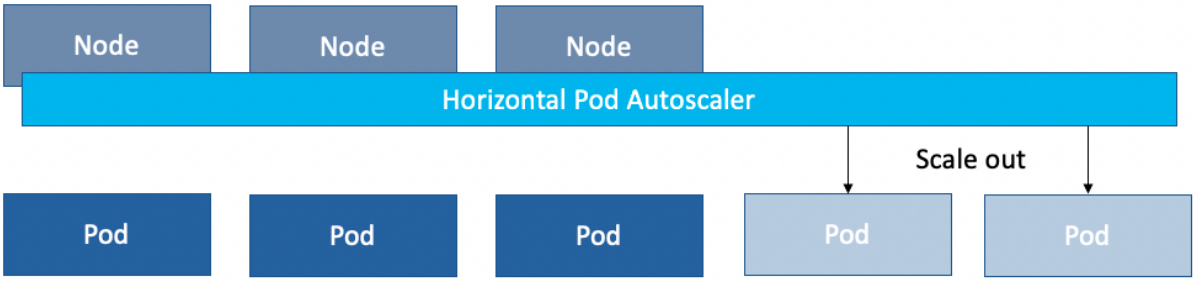

Figure 2.2-2 Scale the number of replicas in Kubernetes

Data centers and cloud infrastructures appear to have an almost limitless ability to expand their capabilities, for Example, the Horizontal Pod Autoscaler (HPA) is a tool that Kubernetes employs to monitor resource consumption and scale the number of replicas automatically (Figure 2.2-2). Although that adaptable scalability could result in an exponentially increasing power consumption, which is unfeasible.

Even though, in a world where everything that benefits from a connection will be connected, one would wonder if centralized processing of such a massive volume of data is feasible. Would processing this data be desirable or permissible within regulatory systems, even if it were possible? Is centralization still practical when scalability and enormous availability needs grow?

Another issue is the lack of AI handling interoperability. Most intelligence that is available for one application is not necessarily available for another; this implies that the second application's creators will have to create their intelligence from the beginning. There are two possible outcomes from this event. Only a few organizations specialize in offering services for a single type of application in the first situation. This paradigm is problematic because a small number of companies emerge as market leaders, dominating and dictating existing offers without considering interoperability. In the second situation, a huge number of disparate, different, and fragmented applications essentially perform the same function. Edge- and distributed AI have got a bit clarification, hence it provides the basic knowledge for next topic, which is the state of the art methodology for the AI layer.

2.3. New approach of Artificial Intelligence layer

Most applications that now use AI integrate the intelligence as part of the program itself, from a software engineering perspective. Alternatively, they may interact with other applications to obtain the necessary intelligence services. As a result, AI is a part of the application layer. It's appropriate for tightly connected systems and applications with well-defined boundaries

that aren't expected to change or be susceptible to a lot of variation. In such systems, the actual need for intelligence remains constant throughout time, as does the task and goal. A chess-playing program is an example of this type of application. The game's rules and context do not change, and the goal remains constant.

The limits of an integrated intelligence model are obvious in contexts where intelligence requirements are constantly changing over fewer functions or poor connection to the application architecture, to update the intelligence, the entire program must be updated. The simplest solution to this challenge is to detach intelligence from the application layer and make it as self-contained as feasible. The intelligence thus becomes a separate stratum that provides apps with services in the same manner that other layers and platforms do.

An application that can detect a nearby dangerous object for a child can be used as an example. Object recognition algorithms match the pictures with the location to detect nearby dangerous objects. Traditionally, the application uses the device's camera and GPS to take pictures, then applies object recognition algorithms to match the pictures with the location to detect nearby dangerous objects. If the application is built on top of an intelligence layer, the program will ask the intelligence layer to provide notifications when dangerous things are discovered in the area, rather than detecting them on its own. The application still must decide how to show the information and the level of detail required for the application's means, but it is exempt from the object recognition processing element. Figure 2.3-1 depicts the two approaches for integrating intelligence with the application layer.

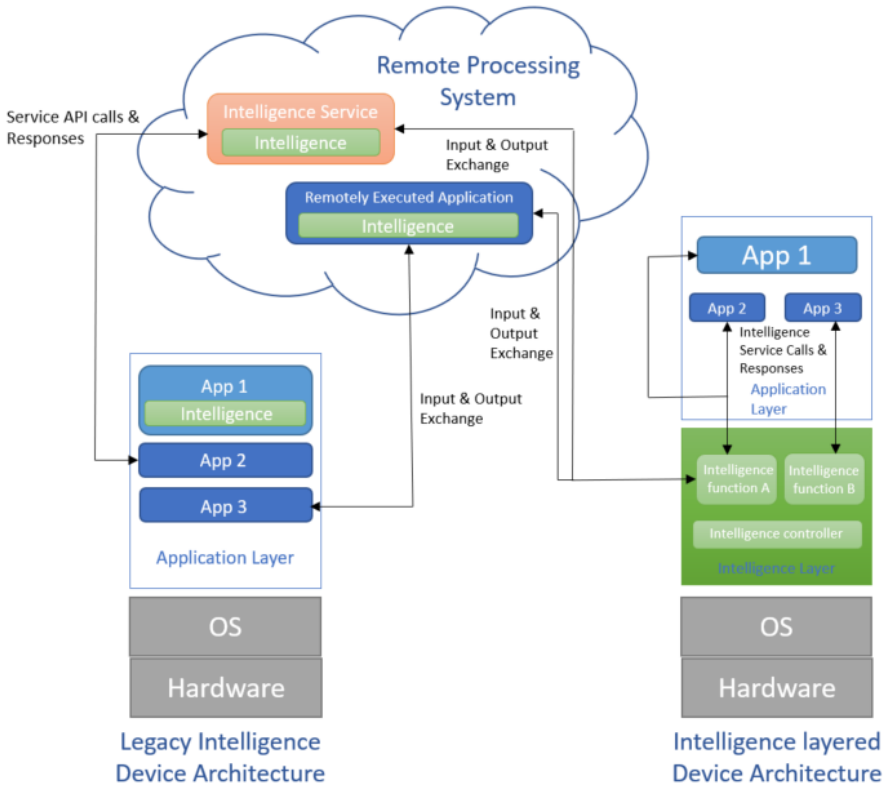

Figure 2.3-1 Comparison of an intelligence-layered device architeture with a legacy intelligence device architecture

Another consideration is where the model should be trained. Rather than believing that Edge AI (EI) should be the paradigm that fully uses the available data and resources throughout the hierarchy of end devices, edge nodes, and cloud data centers to improve the overall performance of training and inferencing a Deep Neural Network (DNN) model. This suggests that EI does not necessarily imply that the DNN model is fully trained or inferred at the edge, but rather that data offloading can be used to coordinate cloud–edge–device coordination. EI can be characterized into six stages, as illustrated in Figure 2.3-2, based on the volume and length of data dumping. [10] The following is a list of the many degrees of EI and their definitions.

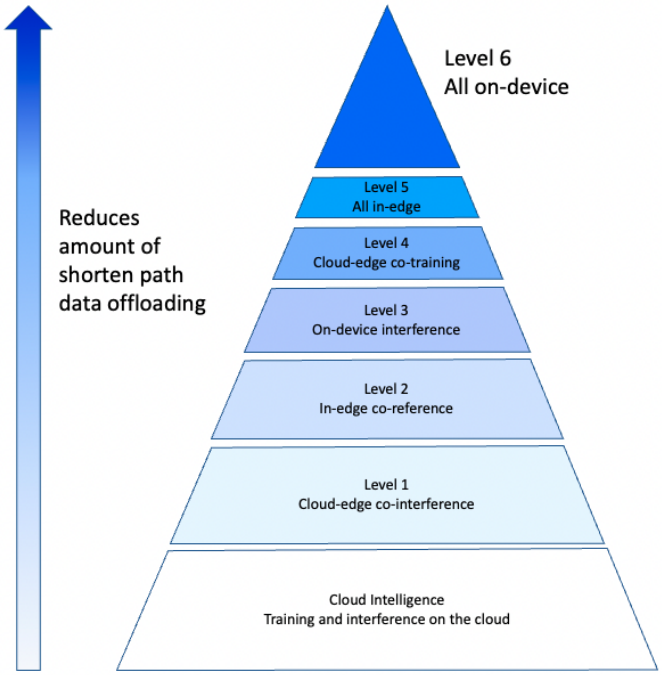

Figure 2.3-2 Six-level rating for EI

1) Cloud Intelligence: The DNN model is fully trained and inferred in the cloud.

2) Level 1—Cloud–Edge Conference and Cloud Training: Train the DNN model in the cloud but infer the DNN model using edge–cloud collaboration. Data is partially offloaded to the cloud in this case due to edge-cloud collaboration.

3) Level 2—In-Edge Conference and Cloud Training: Train the DNN model in the cloud but infer the DNN model locally. The term "in-edge" refers to a model inference that takes place at the network's edge, which can be accomplished by offloading data to edge nodes or neighboring devices entirely or partially (via Device-to-Device communication).

4) Level 3: On-Device Inference and Cloud Training: Train the DNN model on the cloud but infer the DNN model locally on the device. The term "on-device" refers to the fact that no data will be offloaded.

5) Level 4—Cloud–Edge Co-training and Inference: Training and inferring the DNN model using edge–cloud collaboration.

6) Level 5—All In-Edge: Training and inferencing the DNN model both in-edge.

7) Level 6— All On-Device: Training and inferencing the DNN model on-device.

The amount and path length of data offload decreases as the EI level rises. As a result, data offloading transmission latency decreases, data privacy is greatly enhanced, and WAN bandwidth cost drops. This, however, comes at the expense of greater computational latency and energy consumption. This disagreement suggests that there is no universal "best-level" EI; rather, the "best-level" EI is application-specific and should be determined by combining many variables such as latency, energy efficiency, privacy, and bandwidth cost. [10]

For example, if the scenario is an AGVs video stream, and we would like to detect when people are near the AGV to prevent accidents, the model could be trained everywhere since the detection of the people works always the same. Although the model should run on the edge, because response time is much better, and in safety-application latency should reduce to the minimum.

But if the goal is to implement a Simultaneous Localization and Mapping (SLAM) solution, the training should also run on the edge. The SLAM problem asks if a mobile robot can be placed in an unknown area and incrementally build a consistent map of that environment while simultaneously establishing its location within that map. A solution to the SLAM problem has been named the "holy grail" of mobile robotics, besides the increasing abstraction level of commands, since it would allow a robot to be autonomous.

It requires a lot of computing power; hence the location of the execution is important. The robot search for marker points in the stream and on the map also, after it found it then tries to triangle itself. Although references could be moved also, new items could appear and old ones disappear on the map, etc. It could be supported by AI, and Mobile Network also has information about the location, so the two could be combined and reach the level of accuracy which is impossible for the old implement AI in the application method. These and a lot of other use-cases are imaginable with that architecture, in the next chapter I describe the possible use-cases a bit deeper.

3. Use Case scenarios

Edge AI could be a plethora of new use cases, since it opens the opportunity for applications that demand minimal latency. These solutions can be place in a wide spectrum, thus my objective was to build an architecture that is able to provide each of them. Some of the possible solutions are mentioned above:

1. Enhance functional safety by utilizing external sensors. Instead of deploying light barriers in this case, cameras are used to detect people or other obstacles that are in the wrong location (Figure 2.3-1). As a result, this AI is a machine vision with a single bit of output. We must add this one bit to the Profinet dataflow. Therefore, a proxy that can combine external data with OT dataflow serves as the solution in this situation. In this situation, there is no need to change the OT system. That use-case could be helpful also for self-driving cars which are trying to avoid pedestrians at all costs, but also for AGVs in a factory environment. The issue is solvable with an Object Detection algorithm, which is able to detect human workforce and avoidable object on a simple video stream.

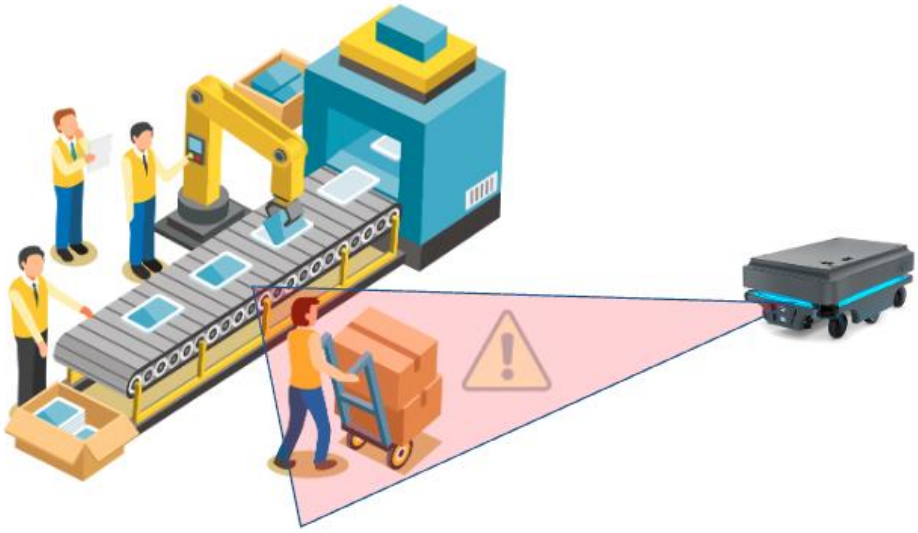

Figure 2.3-1 Enchance Functional Safety

2. Virtual sensors. We could employ less expensive sensors and allow artificial intelligence (AI) make them smarter instead of adding smart sensors to the AGV. For instance, a few moderately priced (stereo) cameras can be used in instead of fairly expensive LIDAR (e.G. SICK S30A-4111CL costs ~3400€). The objective in this more complex scenario is to lower the price of AGVs. We can perform this camera-to-lidar conversion for the current controllers if we don't want to replace them.

   Although deep learning has shown amazing results in camera localization, single-image algorithms currently used frequently lack robustness and produce huge outliers. Sequential (multi-images) or geometry constraint techniques, which can learn to reject dynamic objects and lighting circumstances to improve performance, have helped to address this to some extent.

As solution there's more methodologies, I would like to introduce a deep neural network architecture called Attention Guided Camera Localization (AtLoc) uses self-attention to identify camera postures from a single image. The proposed framework's modular structure, shown in Figure 2.3-2, includes a visual encoder, an attention module, and a posture regressor. The visual encoder condenses the scene of a single image into an implicit representation. The attention module computes the self-attention mappings based on the retrieved features to re-weight the representation into a new feature space. The 3-dimensional location and 4-dimensional quaternion that were added after the attention operators are further mapped into the camera posture by the pose regressor (orientation). [11]

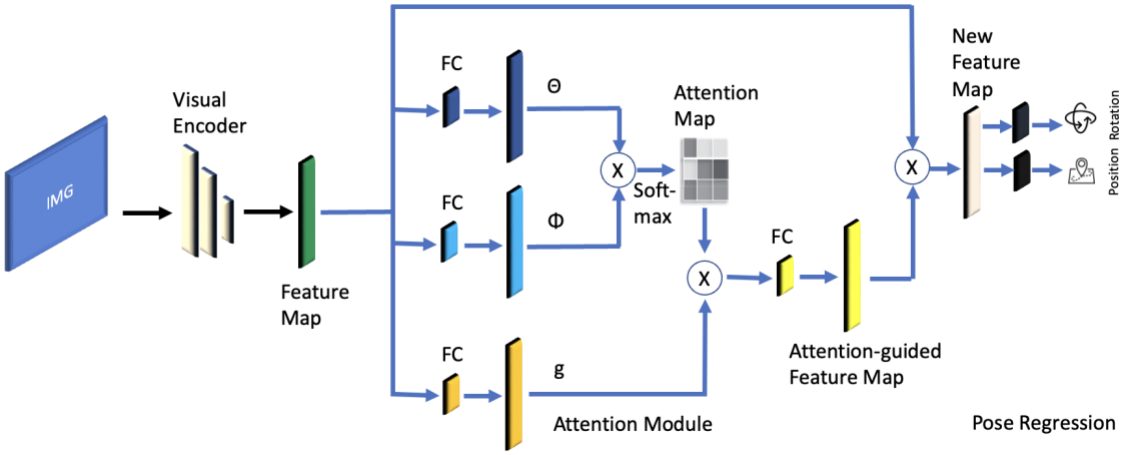

Figure 2.3-2 An overview of our proposed AtLoc framework

3. Increasing the precision of indoor positioning. Indoor localization typically has an accuracy of +-5 cm, albeit the requirement is 2cm or smaller. However, if we use the current CCTV camera to measure the AGV position optically, we can locally abide by the 2-centimeter criterion. In this a non-existing use-case today, because mobile robots are usually based on other methods (magnetic stripe, QR-code, different SLAM methods, etc.). Mobile positioning, however, would be effective if we could somehow improve accuracy. And this is an option. To localize cars outdoors with this +-2 cm accuracy, companies use differential GPS. However, there is currently no appropriate option for indoor.

   The high-tech Global Positioning System (GPS). was created as a result of decades of real-time location research and development that focused on military and civilian target tracking and navigation demands.

   The US Department of Defense controls the space and ground elements of the GPS architecture [12], while the user segment, or GPS receiver, offers services to the user or system. 24 satellites form a constellation that orbits the earth and provide coverage all across the world. Four satellites in line-of-sight are utilized for trilateration, which uses them to estimate the true location. GPS has a wide range of uses, including missile guidance, monitoring of people and vehicles, cellular system clock synchronization, geographic information systems, surveying, and mapping.

Ultrasonic (US), infrared (IR), radio-frequency (RF) based systems, such as radio-frequency identification (RFID), received signal strength (RSS) of RF signals, Bluetooth, wireless local area network (WLAN), ultra-wideband (UWB), camera-based vision analysis, etc., are some of the technology options for the design of an Indoor Localization Systems (ILS).

There are many different uses for ILSs in real life, and each one has specific needs. As a result, there is no one positioning system that is suitable for all applications, needs, and physical situations. A taxonomy put forth by Hightower and Borriello is helpful in providing direction for application developers [13]. Various factors, including system scalability, cost, coverage area, capacity, accuracy, and precision, are used to categorize location systems.

In general, infrastructure and mobile devices can be used to categorize the architecture of positioning systems based on equipment. The primary elements of the system that support location estimate are the infrastructure, as in GPS and satellites. These are referred to as base stations, beacons, transmitters, etc. in ILSs. The individual whose location needs to be determined is connected to the mobile devices, which act as receivers, such as a mobile device, listener, or receiver, or a GPS receiver.

The physical quantity to be measured, the measuring method, and finally the extraction of relevant location information based on the measurements are the three steps of a location system's process for obtaining location information, as shown in Figure 2.3-3. To measure physical quantities and determine location, sensing devices can employ any signal, including US, RF, IR, or vision. These signals provide coordinate information for reference nodes as they move between transmitters and receivers. The physical quantity is then calculated using a variety of techniques, including measuring the time of arrival (TOA), the time difference of arrival (TDOA), the angle of arrival (AOA), the received signal strength (RSS), etc. Several methods and algorithms are used to convert the raw data of a physical quantity measured into useable position information.

Triangulation/trilateration, Scene Analysis, Proximity, and fingerprinting [13] [14] are some of the techniques that have been categorized. The position that algorithms estimate can be either absolute or relative, and it varies from system to system. For example, GPS estimates absolute positioning for every device that is detected.

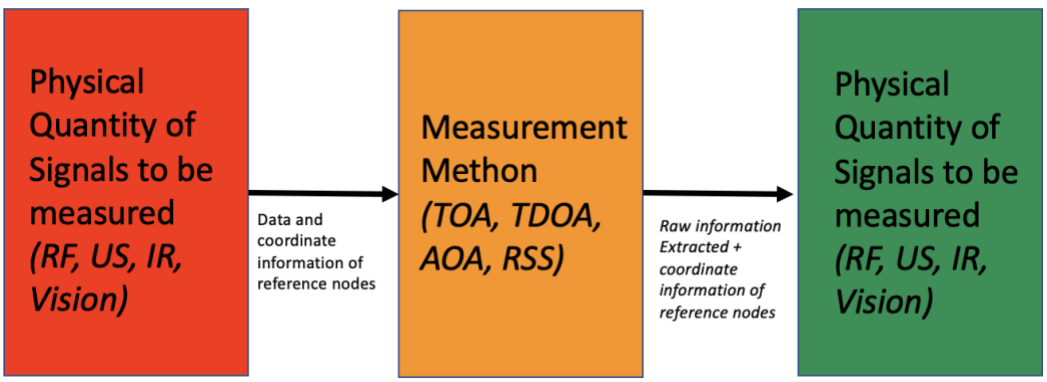

Figure 2.3-3 Phases of Location estimation

In comparison to other ILSs, US location systems provide a variety of benefits, including low system costs, dependability, scalability, great energy efficiency, and—most importantly—no room leakage. US ILSs offer centimeter-level precision (Table 2.3-1 Comparison of US systems Reviewed) and fine-grained localization. US systems have the capacity to serve more consumers at once thanks to their increased ability to track numerous mobile nodes simultaneously. [15]

Table 2.3-1 Comparison of US systems Reviewed

System	Spreading and Channel access	Update Rate	Measurement Method	Accuracy (cm)	Orientation (degrees)	Structure	Cost
Cricket	-	Low (1Hz)	TDOA	10 cm	3-5	Decentralized	Low
Buzz	-	High (33 Hz)	TOA	4-10 cm	Not Supported	Centralized, Decentralized	Low
Dolphin	Gold Codes / CDMA	High(20Hz)	TOA	3 cm	Not Supported	Centralized	Medium
D	Kasami codes FHSS	High	TOA, AOA	1,5 cm	4.5	Centralized	High
E	GOLD Codes/CDMA	High	TOA	2 cm	Not Supoerted	Centralized	

The three-dimensional position of an object can be ascertained using machine vision algorithms and video cameras. Standard mathematical algorithms can carry out this transition.

Using a common Brown-Roberts-Wells (BRW) phantom simulator in conjunction with the BRW angiographic localizer, preliminary accuracy tests of stereotactic localization with video cameras were carried out. The localization's precision was 1.5 mm. Freehand stereotactic localization of the position and orientation of surgical instruments is one potential use for machine vision techniques. These methods can be used to continuously check the location of instruments inside the cranial vault if the computer speed is fast enough. [16]

As a summary with few video cameras more precise results could be achieved than with traditional approaches. Hence if we use some camera streams with an extension of Deep Learning, the application can reach the desired accuracy.

4. Support cellular hints for mobile robots (based on NWDAF - network data analytics function). Compared to legacy cellular networks, 5G cellular networks have many new features, such as the Network Data Analytics Function (NWDAF), which gives network operators the option of implementing their own Machine Learning (ML)-based data analytics methodologies or integrating third-party solutions into their networks. [17]

   For instance, we can alert an Industry4.0 robot of potential cell-handover or poor network quality when it moves and welds simultaneously. Our chained service in this instance is

collaboration with the NWDAF. [18] Between the REST-only NWDAF and the industrial protocol-only OT system, our AI module could turn OT into IT and have an interface that adheres to the NWDAF standards. This NWDAF use case, which is still simply a plan, involves co-located external services.

5. Legacy robot control that is 5G compatible. The well-known P-I-D control is typically used for real-time robot control. Technology that is over than 60 years old but is still in use. However, they are incompatible with cellular networks, which frequently forces us to meet extremely strict Ultra Reliable Low Latency Communications (URLLC) standards. [19] We could accomplish the aims on networks of low quality while employing a superior robot control method. Hence, one option is to build a proxy that, on the outside, resembles a PID-controlled "robot" but, on the inside, controls the real robot using a fault-tolerant semi-open-loop horizon-control technique. (Investment protection, do not need to replace outdated hardware, just upgrade it.)
6. Massive gathering of IoT data. Although a Lightweight M2M (LwM2M) protocol does not even permit cooperation between low-power devices, we could collect data, pre-process it, and then send it back to the industrial system using an AI in the core. [20] Because LwM2M is encrypted in this use-case, the chained method is impractical. It is preferable to utilize a separate service.
7. AI on the UE, somehow integrate AI into the UPF. DPI and service chaining are solely two potential and practical deployment strategies. The network conceals the implementation of A.I., so we can still add it to the system. A.I. can operate on the UE and in micro-containers, which has two significant benefits: it can operate offline and has extremely low latency. I do not intend to address the significant orchestration issue that downloads themselves provide just yet. I want to concentrate on imparting these logics. Since training an AI requires a lot of power, it should be done on a UPF host using hardware acceleration (a GPU). Deep Learning Solution, which will be chained in the network**.** Therefore, in this model, we simply use the chained AI to learn and create a model, after which we download it to the UE.

Besides these, several other scenarios could be implemented, the main concept is the architecture which can host these use-cases, this architecture is introduced in the following chapter.

4. System Architecture

There is a lot of legacy AGVs in factories, which could be control more efficiently from the Cloud or from the Edge, since in that case AGVs have all the data during the functioning. When the controlling part moved from the machine into the Edge robots need wireless communication technology to communicate.

4.1. Communication Protocol

The cellular industry wants to enable ultra-reliable low latency communications (URLLC) for the industrial Internet of Things as it transitions to 5G mobile technology (IoT). [21] [22] Factory automation (FA) is a common application where closed-loop control applications execute periodic cycles with cycle lengths between 0.25ms and 50ms and require extremely low downtime with packet error rates of 10 or less. [23] Historically, wireline bus systems have handled these applications. [24] A paradigm shift toward moving FA communications from landline to wireless has taken hold in recent years. The numerous drawbacks of wireline solutions, such as their high installation and replacement costs for wiring as well as their maintenance-prone support for moving sensors and actuators, which rely on trailing cable systems, slip rings, or sliding contacts, are the driving forces behind this development.

The current wireless communication systems used in industrial automation typically operate in the industrial, scientific, and medical (ISM) radio band and use unlicensed technologies. These bands' broad bandwidths enable factories to handle the high volume of traffic typical of FA applications. However, the current commercial solutions are unable to meet the demanding dependability goals with latencies < 10ms.

In mobile broadband, where smart devices opportunistically prefer Wi-Fi over cellular technology, such a technique has long been prioritized. In order to increase capacity at a low cost, the cellular industry has continued its efforts to extend into the unlicensed band. But unlike FA, mobile broadband has a different economic reality. Resource contention in the unlicensed band does not noticeably degrade the service experience because latency is much less strict than in FA.

Additionally, cellular connectivity offers wide-area coverage, which sets it apart from unlicensed-band hotspots. Last but not least, mobile operators can spread the cost of the licensed band across a large user base.

4.2. Service Chaining

If AGVs are controlled from the edge, the controlling system becomes smarter and is capable of performing out more difficult tasks. Additionally, 5G is used as a communication protocol because of its lesser latency and more dependable coverage. AGVs under certain scenario

have UE on them. My ambition is to provide the cellular network an additional layer of intelligence in order to make the system smarter. I obtain it using the Service Chaining because it is the effective way for the task. My objective is to determine whether the employment of that architecture is effective, efficient, and helpful or not. What are the solution's benefits and drawbacks in comparison to locally running and edge-running solutions? Service Chaining can be used as a component of the solution even though it was not initially intended for that proposition.

Service Chaining is a widely used solution in cellular networks since 3G. Network Function Virtualization (NFV) is a technology, which purpose is to virtualize network functions and applications that were previously only available through hardware appliances. The goal of Software-Defined Networks (SDNs) implementation method is to replace numerous network equipment with more flexible software that runs on real servers, virtual machines, or clouds, allowing for much more flexible service chaining. Indeed, software service chaining allows operators to dynamically construct network services, addressing the need for both network optimization and revenue through the deployment of customer-dependent services.

Creating a service chain to support a new application used to take a long time and effort. It entailed purchasing network devices and connecting them in the proper order. Each service required its specific hardware device, which had to be configured individually with its command syntax. Because network operations are now implemented in software, procuring hardware is no longer required to construct a service chain. Furthermore, because application demands frequently increase over time, designing a chain that would not require frequent reconfiguration means over-provisioning to accommodate growth.

In Network as a Service (NaaS) model, the network provides Artificial Intelligence for higher-level applications. Loads of services require the transient data, hence an alternative use of the Service Chain, though not its intended purpose, maybe a viable choice.

Industrial systems are mostly divided into domains, each domain organized by different teams. Typically, domains could be Information Technology (IT) and Operational Technology (OT). OT is concerned with the management and control of physical equipment that exists and function in the real world. Controlling real-world equipment dates back to the dawn of industry and production. Electronics and digital technology have found a wide range of applications in operational control systems, including computerized numerical control machining systems.

OT focused on behaviors and outcomes, while IT focused on data and communication. Most industrial and factory control systems were not networked, resulting in silos of specialized equipment, each of which was electronic to some degree but unable to connect or share information. This indicates that each piece of equipment's physical functions was programmed or managed by human operators. Closed or proprietary protocols were utilized even by devices that allowed centralized control. IT encompasses the creation, processing, storage, security, and sharing of all types of electronic data using computers, storage, networking devices, and other physical equipment, infrastructure, and procedures.

In a typical industrial scenario where the purpose is to manage AGVs in a plant, the robots, as well as the application that supports them, are in the OT domain. It is considerable for robot developers to rent AI rather than develop it themselves. As an example, the network operator, implement it in the service chaining system of the network.

Cascading Solution: All middleboxes (MBoxes) in service provider networks are cascaded with this approach. The MBoxes are serially chained, as shown in Figure 4.2-1a. Despite its simplification, it has several flaws, the majority of which stem from the fact that all traffic flows must transit through all MBoxes, rather than just the ones that are desired. To begin, inserting MBoxes or upgrading existing ones necessitates a significant amount of human labor and operator experience. Second, any MBox failure might bring the entire service-chaining network to a halt. Third, the cost of processing and forwarding all traffic flows for each MBox rises.

Branching Solution: The service chains are designed ahead of time in this system, and traffic is then branched to the appropriate service chain. Deep Packet Inspection (DPI) is used to classify traffic flows, as seen in Figure 4.2-1b. The branching technique has been enhanced over the cascading deployment described above to address the aforementioned issues. Despite its utility, it is not without flaws. First, when too many traffic flows pass via service-provider networks, the DPI device may become a bottleneck. Second, making service-chaining changes, such as adding or removing MBoxes from an existing service chain, is difficult. Third, because the MBoxes cannot be multiplexed, it could result in exorbitant expenses.

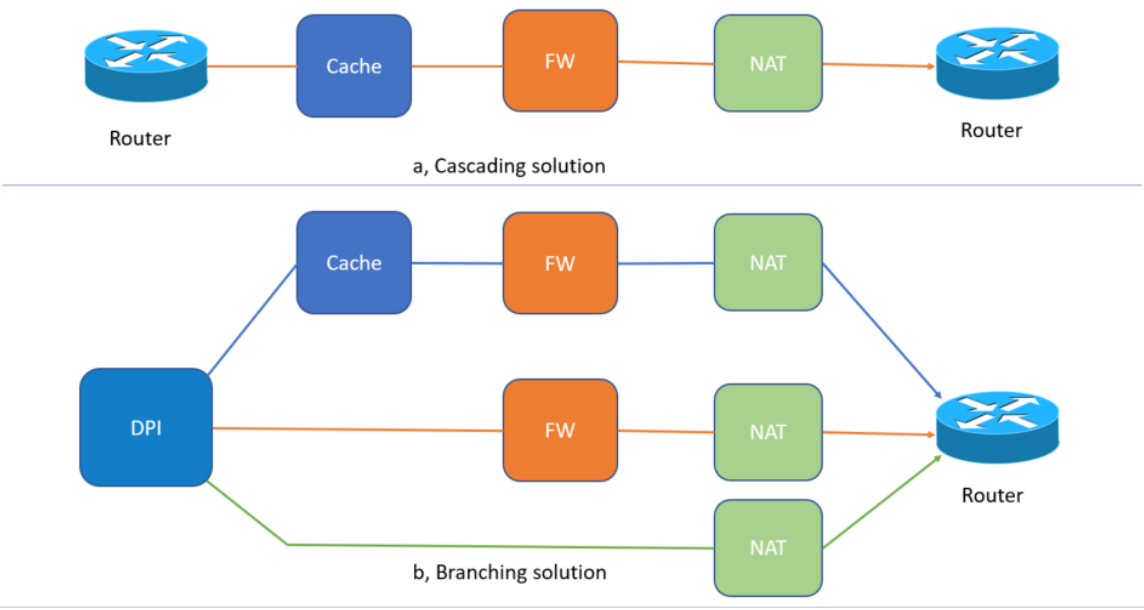

Figure 4.2-1 Traditional solution for service chain

Current service-chaining systems have many shortcomings [25], including limited scalability and expensive operational costs. As a result, academics and businesses are on the lookout for a new service-chaining solution that is both flexible and scalable. In our case the DPI module

selects which flow is to send to service chaining, the decision is based on pre-defined rules, which are applied for session.

Before the installation of the system and the implementation of the Artificial Intelligence layer, the architecture needs to be defined. In my case that means I need to define two different architectures, first is the real-life scenario, when the data is generated by real users, with physically existed equipment. The second is the demo deployment when my purpose is to represent the first one with simulations, where the traffic generator simulates the cases and the scenarios that could happen in a real-life environment. The two architecture needs only differ in the generation of the traffic and in the application, both of them are replaced by simulation. The traffic generator is replaced by a traffic generator, the simulation is with AppSimu.

4.3. Architecture of the real-life scenario

In the real-life scenario, the traffic is generated by real users, which means in daily life that the people generate it with their mobile equipment. Although my application focuses on the industrial sector where the equipment is typically not the traditional mobile phone, but rather robots, sensors, and other equipment which supports processes in the factory. Let us focus on the robots, it can be a single AGV, a fleet of AGVs, or maybe a huge group of different types of robots including robot arms, AGVs, and cobots.

The whole architecture is inFigure 4.3-1, in that case, the source of the data flow is a couple of Automated Guided Vehicles, they are communicating directly with Radio Access Network (RAN), then from RAN data flow to the User Plane Function, to the UPF, which is run on a server.

The virtual machine or container runs the Controller, which is the control plane, that controls the UPF. One of 5G's main concepts is Control and User Plane Separation (CUPS). In 5G systems the user plane and the control plane is separated, this allows running only a UPF on-prem, and the CP in the operator datacenter.

Controller's components are AMF responsible to maintain connection and SMF which is liable for the session, which means it controls the UPF. UDM tracks the sim cards, policies, etc. in the systems, and in that scenario, the robots use these SIMs, and UDR is a huge database. Controller is omitted in the demo deployment; however, it should be noted that it is a required component in that scenario.

Data forwards to UPF by RAN on the N3 interface, information flow has a session, and that has a lot of other parameters. The packet then arrives in a ramification, the two possible ways are slow path and fast path. The first packets are the most interesting, they need to be investigated to be able to describe the traffic. Packets flow through a Deep Packet Inspection and it applies the rules to the session if needed. For example, sometimes it needs to slow down the bandwidth, because of the tolerated but unsupported activity, sometimes it needs

to stop the price charging, for example in the case of services, which are provided by the mobile network provider without debit the monthly limit. After the investigation happened, the rest of the packet can take the fast path, and the further investigations can be omitted. After that it arrives at Service Chaining if it is defined in the ruleset, if not it leaps that step.

The AI has been implemented in the Service Chaining, not in the application, hence when we forward it finally to the application, the application gets the output of the Deep Neural Network also. When the Deep Learning Solution (DLS) is an object detection algorithm, which takes an AGV's camera's output as an input, we could send the list of recognized objects, the bounding boxes which appears on the video stream. Maybe the output is chained further, and different Artificial Intelligence services use it as input, the chained service could be for example responsible for detecting anomalies and prevent accidents, so the traffic flow which arrives in the application carries warning messages and the application can start to react to them.

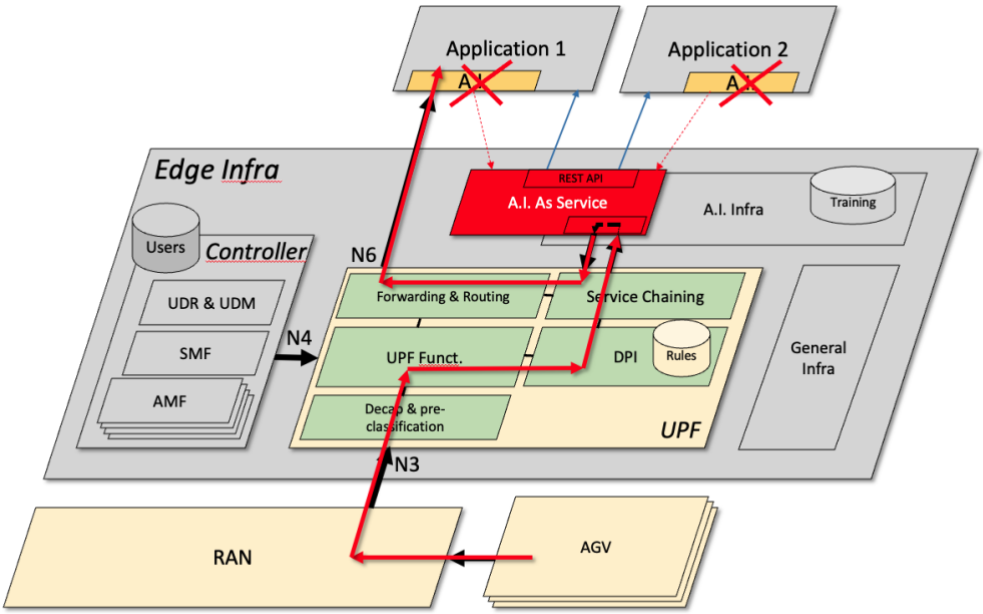

Figure 4.3-1 Architecure of the real-life scenario

4.4. Architecture of the demo deployment scenario

Before building a complete 5G core and implement a full-scale PoC, I have implemented simplified version, focusing on the UPF-AI connection. The complex and expensive system elements were simulated from software. In that case the traffic is generated by a traffic generator instead of real AGVs. In Figure 4.4-1 the most obvious difference, if we compare it with the previous one, is that in the case of that architecture there is no Controller included, the cause of the absence is the Traffic Generator, which simulates it. In that scenario everything happens between servers which have placed in a Flight Rack (FR), Apart from the source of data, the radio network, and the endpoint, which in this case is the application, the data travels inside UPF in the same way.

The traffic after Service Chaining flows back to the Traffic Generator. The reason is that when it stimulates the internet it acts as a drain and when it simulates the mobile traffic it acts as a generator.

A simulation is a representation of the operation of a real-world process or system. The parts which are not simulated should run in the same way also whether the traffic is simulated or not, that architecture is suitable for simulating a real-life event, and allows testing the implemented AI layer.

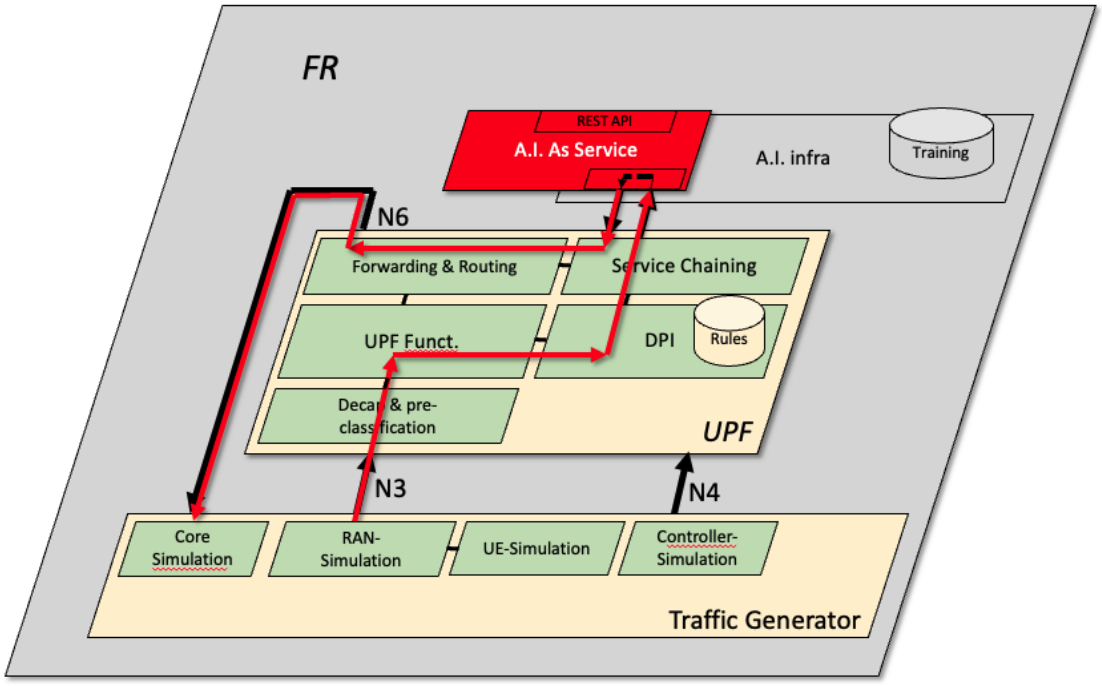

Figure 4.4-1 Architecure of the demo scenario

Excluding the A.I. As Service part, most components of the architecture are fixed. I described couples of scenarios, but I had to decide which one to implement. The next chapter describes the selected scenario.

5. Deep Learning Solution, which will be chained in the network

Several scenarios could be implemented in the network, and various types of data flow could be using different deep learning solutions, or deep learning solution chains, but as I created a Proof-of-Concept (PoC), I was considered to create a solution that legitimate every other solution also.

Object Detection is adequate for PoC purpose, since that comes with a lot of solvable issues, like receive video stream, process video stream, and encode the processed frames back to a video stream and forward it further. To solve the beforementioned issues high amount of processing requires on the data. Catching video stream needs to be fast as possible, Object Detection solutions requires a complex model, but despite that fact it should be run as fast as possible to minimize the latency which caused. The video encoding has a highest resource requirement, hence its critical to find the optimal solution.

The use-cases for that application are extensive. AGVs are equipped with a variety of sensors, including Line Sensors, Light Detection and Ranging (LIDAR), Microphones, and Cameras. A camera is commonly a prerequisite on AGVs, and it is not affordable. The robot should drive safely if it passes through an area where people are present, therefore object detection could be useful in a variety of circumstances like to prevent an accident. In addition, the method might be applied to any other situation in which a camera is present and capable of sending a video stream to the network.

The choice of an appropriate model for the solution is the first and most crucial step. After defining Object Detection, I went on to explain the approach I took in developing my solution.

5.1. Choosing model for Object Detection

The most important component in my Deep Learning solution is the model, several algorithms available and the choice of the right one is crucial. Making the right decision requires a basic knowledge about Object Detection and its algorithms.

Detecting instances of visual objects of a specific class (such as persons, animals, or cars) in digital photographs is an important computer vision task. Object detection's goal is to create computational models and algorithms that give one of the most fundamental bits of information required by computer vision applications: What objects are where?

Convolutional Neural Networks (CNNs) reemerged in 2012 [26]. Because a deep convolutional network can learn robust and high-level feature representations of an image, it's logical to wonder if it may be used to recognize objects. Object detection has progressed at an extraordinary rate since then.

Object detection is an important aspect of computer vision. Pose estimation, vehicle detection, and surveillance all benefit from object detection. Object detection algorithms differ from classification algorithms in that detection methods attempt to locate an object of

interest within an image by drawing a bounding box around it. Also, in an object detection instance, sometimes only just one bounding box creation is needed, there could be multiple bounding boxes representing distinct things of interest within the image, and it is not known how many until after the fact.

The main reason this problem cannot be solved by using a typical convolutional network followed by a fully connected layer is that the output layer's length is changeable — not constant — due to the fact that the number of occurrences of the items of interest isn't fixed. To solve this problem, a simple technique would be to extract different regions of interest from the image and use a CNN to classify the presence of the object within each zone. The problem with this method is that the objects of interest may be in different spatial locations and have different aspect ratios within the image. As a result, a large number of regions have to be chosen, which could take a long time to compute. The most popular algorithms are R-CNN, Fast R-CNN, Faster R-CNN, YOLO, and SSD.

In Table 5.1-1 I collected the parameters of the the fastest and most efficient models. SSD is intended for real-time object detection. Faster R-CNN creates boundary boxes using a region proposal network and then uses those boxes to classify objects. The entire process runs at 7 frames per second, which is considered state-of-the-art in accuracy. Far below the requirements of real-time processing. By removing the need for the region proposal network, SSD speeds up the process. SSD implements several enhancements, including multi-scale features and default boxes, to compensate for the drop in accuracy. These enhancements allow SSD to match the accuracy of the Faster R-CNN using lower resolution images, further increasing the speed. It achieves real-time processing speed and even outperforms the accuracy of the Faster R-CNN, according to the following comparison (Table 5.1-1). (The precision of the predictions is measured as the mean average precision mAP.) Considering these results, I decided beside SSD300 algorithm. Since the model has been chosen, the next step was to implement it.

Table 5.1-1 Object Detection algorithm comparison

System	VOC2007 test mAP	FPS	Number of Boxed	Input resolution
Faster R-CNN	73.2	7	~6000	~1000x600
YOLO	63.4	45	98	448x448
SSD300	77.2	46	8732	300x300
SSD512	79.8	19	24564	512

6. My object detection implementation

Training a new model from scratch would be a huge task since it needs a lot of data, and the training process requires a lot of computing power and time. The best models nowadays are trained in the cloud, which has seamlessly limitless GPU capacity, so I have chosen to use a trained model. In the previous chapter I mentioned the widely known algorithms, I have chosen SSD from them, since it is the fastest and most accurate.

I downloaded a well-trained model and I use it with python scripts. The model works on images, hence, to use it on a video stream I needed to process the video frame by frame. The model can identify 20 different items, including: background, aero plane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, sofa, train, TV monitor. The 21st is the background, but that category is kept for 'no match' cases. From these the most important is person, since I would like to recognize human presence. The model's accuracy was satisfactory, and it is suitable for usage in my application.

6.1. Architecture

The entire application operates on a flight rack, which is a portable 5G-SA core that contains several servers. I used three of them during my project: the Jump Host, which grants access to other servers; the Edge UPF, which provides us the router functionality in 5G networks, and the Cognitive Edge server, which is designated to run the AI module.

The running processes and the path of the packet flow is represented on Figure 6.1-1, on the Jump Host I stored the video file(s), which has HD resolution (1280x720p), I sent it through the system as testing purpose. The video file is sent with VLC to Traffic Generator inside the server. The next hop is the Edge UPF, which had Agent Marci pod in the beginning, but it was moved to the Cognitive Edge server because it needed more computing capacity, hence it forward towards the Cognitive Edge server, which runs Agent Marci pod, and it is also in the same Kubernetes cluster. Forwarding in Edge UPF is described in detail in Chapter 4.2 and 4.3, hence I do not go into details this time. After Agent Marci caught the packets, it processes it and sends back to Traffic Generator. The caught packets store in the Jump Host server, that means the stream can be replayable and has all the functionality what a video stream nowadays has.

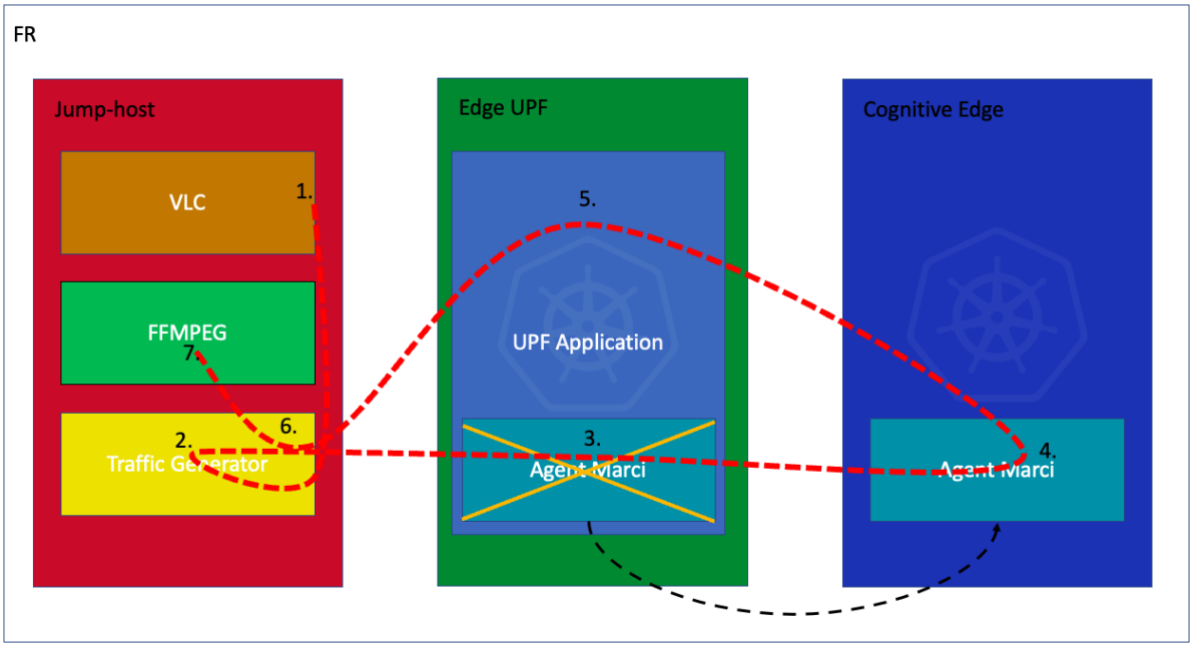

Figure 6.1-1 Running Processes on nodes

In that demo scenario I used a Traffic Generator to simulate traffic, that Traffic Generator works as a Gateway. I send a video file with VLC, divide it into UDP packets, and send it to Traffic Generator, it sends the received packets immediately to Edge UPF, in Edge UPF Deep Packet Inspection recognize that these packets need to send to service chaining, hence it forwards packets for Agent Marci. Agent Marci receives packets with OpenCV, and after processing the frames it recognizes the object in the frames and make a bounding-boxes around the object and labels them, for Example when recognize a person in the frame it draws a box around them and write 'person' label and accuracy rate to the box. After that Agent Marci generate a video stream from the frames and send back through the system, and then ffmpeg receives the packets. After making a tunnel to the ffmpeg container, the video stream is able to catch by a video player, for Example with VLC, or also implemented a simple webpage which has a video player, which is able to play the video. The Chart of the process shows in Figure 6.1-2.

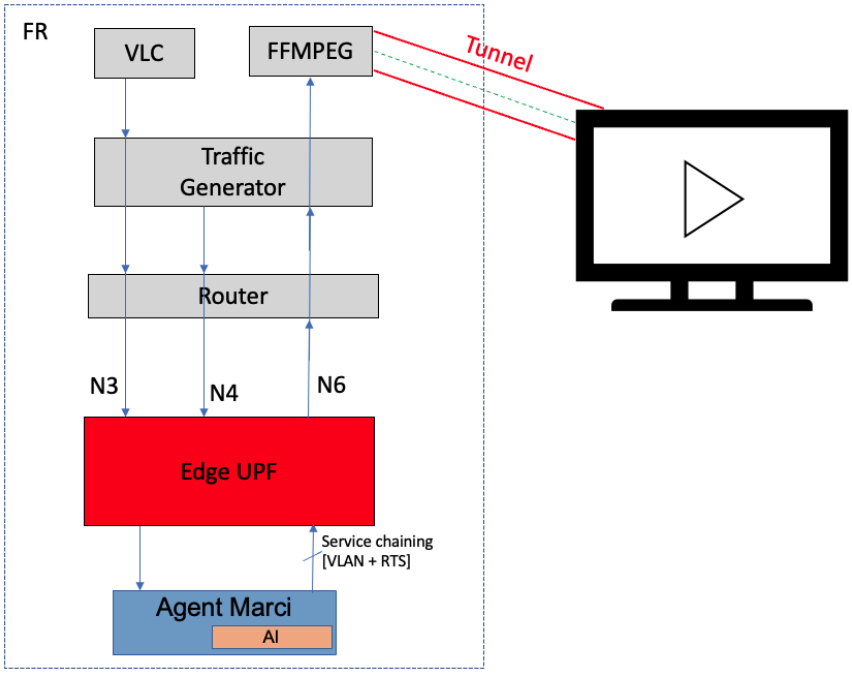

Figure 6.1-2 Video-streaming process chart

6.2. Decisions made for process' architecture

Every part of the process was built in purpose. First to imitate the video stream of the camera I needed to make a video stream, which in a real-life scenario is replaced by a stream of a real camera. I needed an UDP based video stream, since in mobile networks travels UDP packets.

I collected alternatives to find the best for my application, these are:

- GStreamer: A library used to build media handling component graphs. Its supported uses include everything from straightforward Ogg/Vorbis playing to advanced audio (mixing) and video (non-linear editing) processing. It's free and open-source, has demuxer, decoder, processer, encoder, muxer choice and setting, dynamic plugin loading, variety of formats, detailed pipeline and process debugger, and it cross-platform. I also has an experience with this framework, I sent AGV's video stream and to a laptop, where I use the stream to recognize traffic signs and sent back controlling commands to the robot.
- FFMPEG: The universal multimedia toolkit. It's free and open-source, cross-platform. Demuxer, decoder, processer, encoder, muxer choice and settings, dynamic plugin loading, detailed pipeline and process debugger, screen capture, streaming support, multiple graphical front ends, but it has a steep learning curve.
- VLC: Multipmedia player and framework that plays most multimedia files as well as DVDs, Audio CDs, VCDs, and various streaming protocols. It's free and open-source, cross-platform. No outside codecs are needed, support variety of formats. Supports streaming across private networks with sftp/ssh to any device which has VLC.
- MPV: A free media player for the command line. It's free and open-source, cross-platform, no outside codecs are needed, supports variety of formats. Able to cache

livestreams, has many front ends. But it has minimal interface, front-end required, harder to install then VLC.

I've chose to send the video file with VLC, because it made the stream suitable for Traffic Generator. The Traffic Generator forwarded the packets with DPDK through the system, the Linux Foundation oversees the open-source software project known as the Data Plane Development Kit (DPDK). The operating system kernel's processing of TCP packets is offloaded to user-space processes using a collection of data plane libraries and network interface controller polling-mode drivers. This offloading outperforms the interrupt-driven processing offered by the kernel in terms of compute efficiency and packet throughput (Figure 6.2-1).

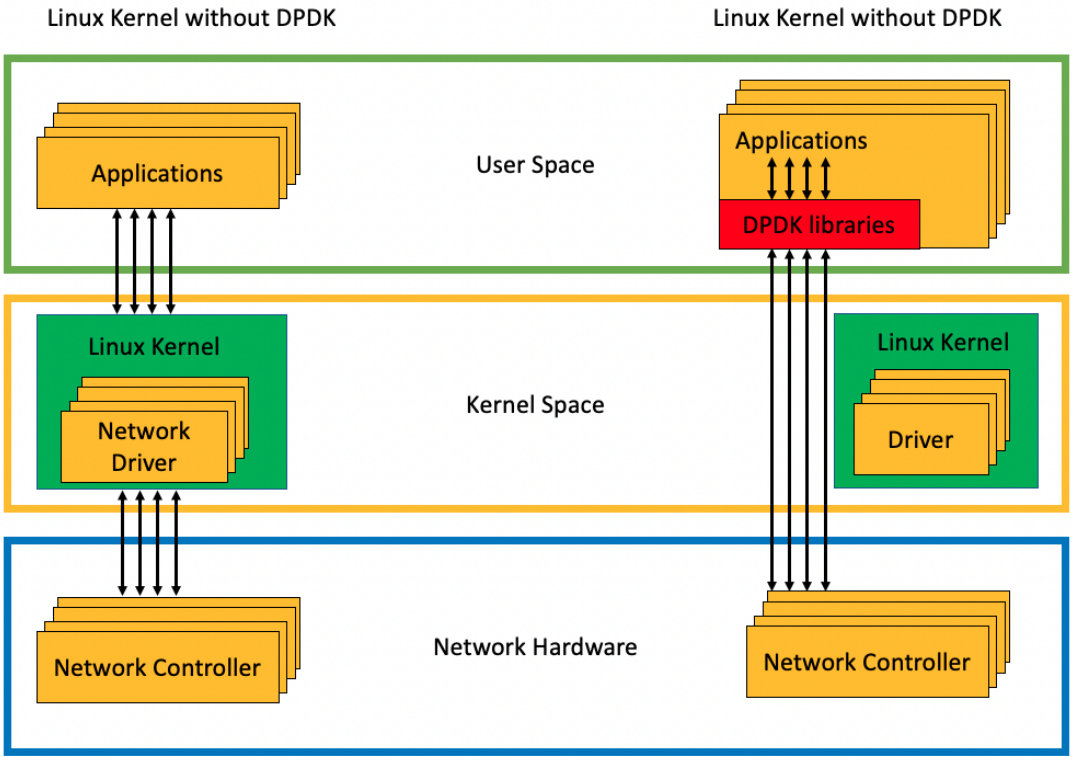

Figure 6.2-1 How DPDK works

The first problem occurred that point, since I had to process the packets, but I needed to catch DPDK's packets. There is a code which responsible to catch DKPK packets, it's written in C, and in production the best solution would be to do the whole processing with the extension of that code, but it would be a lot of development for a PoC solution, hence I decided beside python, which is currently the most popular language for Deep Learning purposes.

The build method involved sending packets into a session, reading them out, sending them back to another socket, and eventually sending them on to their intended destination. At that point I sent traffic through the system, and everything was ready to work except the processing part of the data.

6.3. Processing the stream

The most crucial part of the application was the processing part, which should be work nearly real-time. That means the processing needs to be fast as possible, which requires computational power.

The first step was to send a video stream through the system without implementation of AI and without the use of entire Service Chaining. I measured the latency and it diverged between **2-10 ms**, which is low enough for the most scenario.

After sending stream through the system worked, the next step was to implement Service Chaining in them and smart the system up. The container which is responsible for the AI called Agent-Marci, hence it is an AI agent and I named it after myself. Hence, I added a new pod to the Kubernetes environment, first on the Edge UPF's machine, but resources were allocated to other tasks which are required by the Edge UPF and the resource was insufficient for the deep learning and video receiver/encoder libraries, henceforth I was forced to move the module to the Cognitive Edge server, which was originally designated for that purpose. Since I calculated with that scenario, where the container requires huge number of resources, hence I has an experience in Deep Learning and Video Streaming from the past.

Cognitive Edge Server has 40 CPU, type of each is Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz, albeit I had huge number of CPUs, I tried to use the less resource which I can, since there could be more chains, or more element of the chain. Although I have known that I do not aim to develop a production grade application. I separated the reading and the processing into two different threads. First the encoding was the bottleneck. In most Object Detection application, the processing happens frames by frame, for processing mostly NumPy, pandas, and OpenCV are used, the Deep Learning Model has a fixed size input, and the output is an OpenCV frame. The output stream is mostly not a real stream, just figure the OpenCV frames one by one and it seems like a video. In that case the output is not a real video, just a sequence of pictures, and a lot of frames lost during the processing, but it not recognizable. Since I needed to send back a video stream, I had to encode frames into video.

First try was with OpenCV video writer, forasmuch the frames read out with the help of OpenCV, the output is also an OpenCV frame, the most obvious solution was to use its library to solve the issue. Unfortunately, that solution was too slow.

Other solution was to write bytes of the frames on standard output, then pipeline it to ffmpeg, which save the video file. That solution was faster, but not fast enough, hence I needed an optimized, multi-threaded solution.

There is a better solution, a full, adaptable, and reliable wrapper over FFmpeg, a top multimedia framework, is offered by WriteGear API. Real-time frames can be converted by

WriteGear into a lossless compressed video file with all necessary parameters (such asbitrate, codec, framerate, resolution, subtitles, etc.).

Additionally, WriteGear supports classic streaming protocols like RTSP/RTP and RTMP. With just a few lines of code, it is capable of sophisticated tasks like multiplexing video-audio with real-time frames and live streaming (for platforms like Twitch, YouTube, etc.). The best part is that WriteGear's unique Custom Commands function gives users total freedom to experiment with any FFmpeg option without relying on any external APIs.

Additionally, WriteGear offers flexible access to the VideoWriter API features from OpenCV for encoding video frames without compression.

The following two modes are the main ones that WriteGear uses (Figure 6.3-1):

- Compression Mode: To compress lossless multimedia files, WriteGear uses the robust internal encoders of FFmpeg.This mode enables us to quickly and easily take advantage of practically every parameter offered by FFmpeg, and it does so while quietly and robustly handling any problems or warnings.
- Non-Compression Mode: In this mode, WriteGear makes use of the fundamental features of the OpenCV VideoWriter API. The ability to alter encoding parameters and other crucial characteristics like video compression, audio encoding, etc. is not available in this mode, which supports all parameter transformations offered by the OpenCV VideoWriter API.

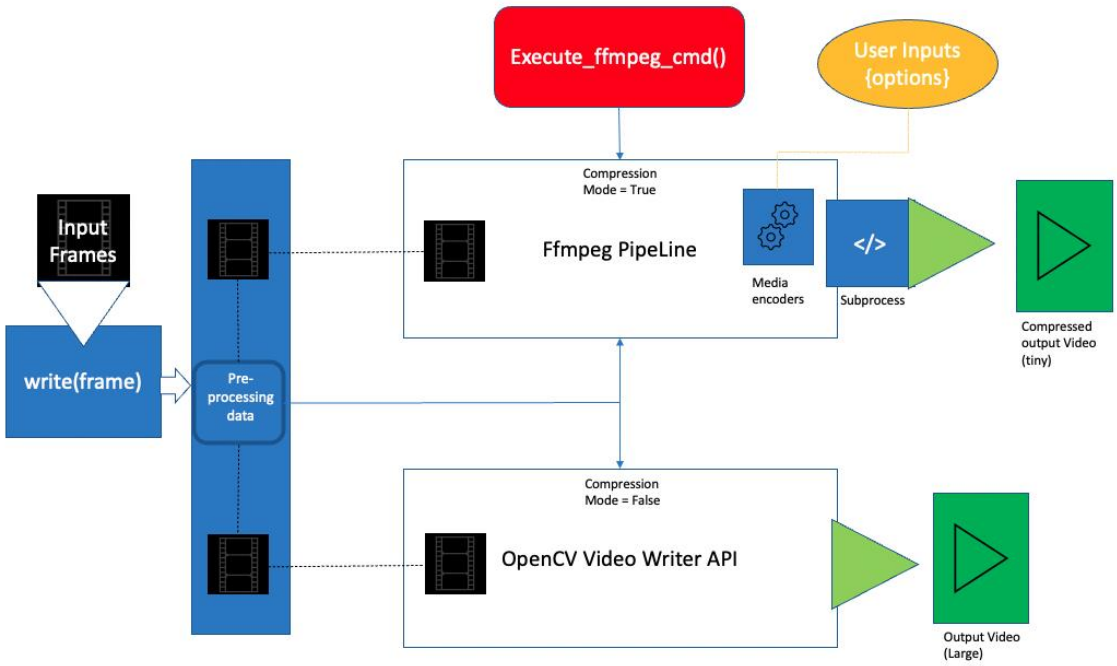

Figure 6.3-1 Functioning of VidGear's WriteGear framework

Video was saved to an m3u8 file, an UTF-8 Encoded Audio Playlist file is one that ends in the extension M3U8. Both audio and video players can use these plain text files to describe the location of media files.

Might see references to online files for an internet radio station in one M3U8 file, for instance. One more might be made on the computer to compile a playlist of own music or a collection of videos.

It doesn't matter which method to use, the result is the same: open the file to quickly and easily begin playing whatever the playlist points to. Might create an M3U8 file as a sort of shortcut to play those tracks in your media player if discover that urge to listen to the same music repeatedly.

Since the file segments are made continuously, I am able to transmit them back to their original location, and as I already indicated, the stream can start playing once a tunnel has established. (Figure 6.3-2).

Figure 6.3-2 Playing the stream

When I am optimized the encoding process, I started to measure the efficiency of the system, writing speed means how fast the writing is compared to the video's speed. That means 1x writing speed means real-time, hence that was the minimum which I wanted to accomplish. As I started to increase the number of the CPUs, I realized that above 5 CPU the performance do not improves anymore, hence I needed to debug what other issue could be cause that slow writing speed (Figure 6.3-3).

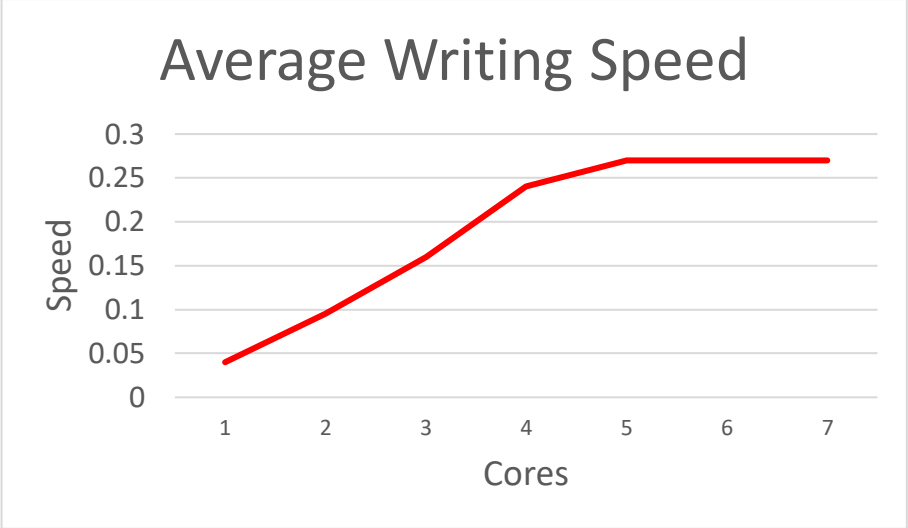

Figure 6.3-3 Speed improvment with Core number increase

And the other bottleneck was the reading speed of the frames, hence I had to implement a more efficient solution for the reading process. Then I implemented a multi-threaded version of reading frame class, which was fast enough. Then continued to increase the number of the CPUs. As I gave more and more resource to the container, I realized that my code could read out more frames (Figure 6.3-4), that means the slower reading rate caused frame loss. The desired 1x writing speed and 25 frames/reached (which was original FPS of the video) with 14 CPUs.(Figure 6.3-5)

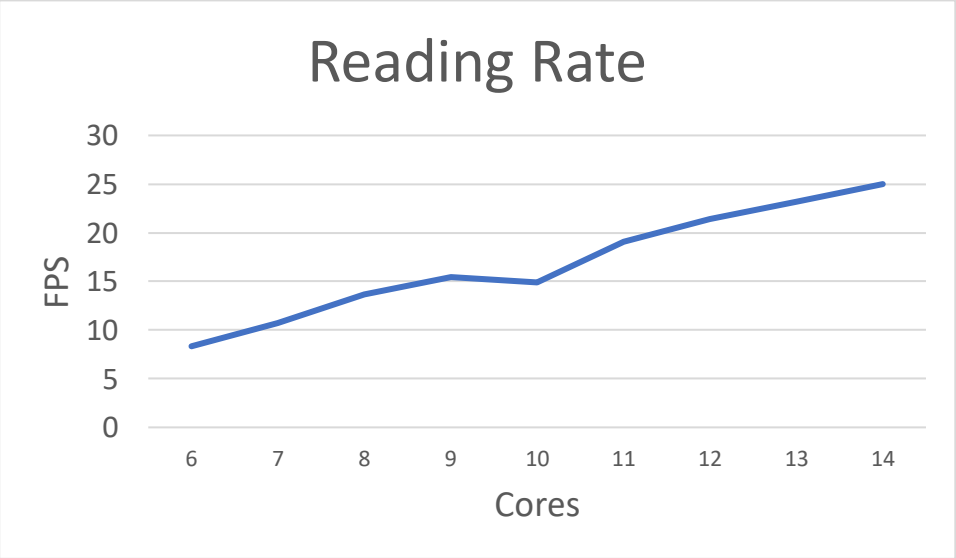

Figure 6.3-4 Number of frames has read

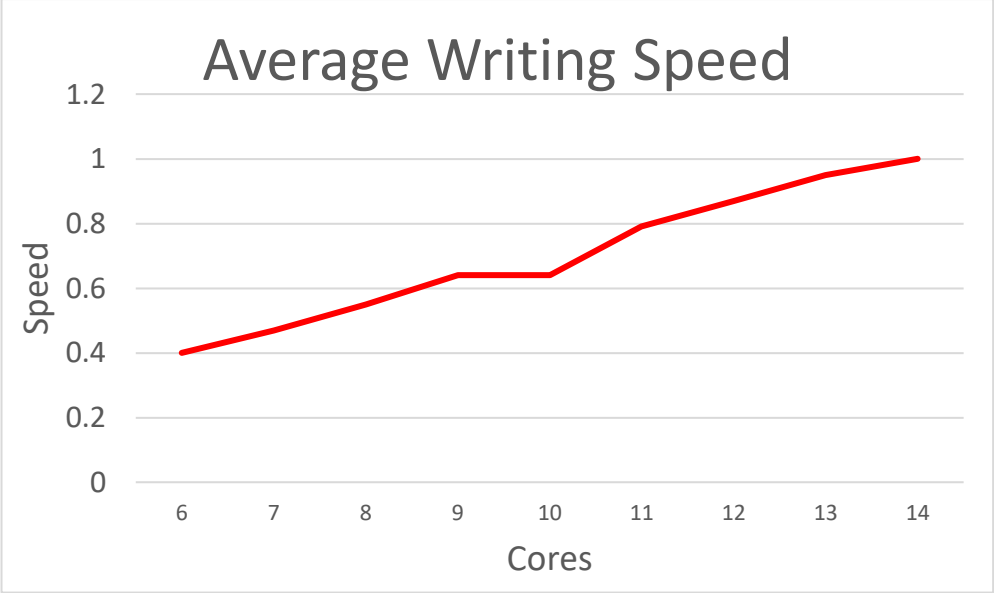

Figure 6.3-5 Speed improvment with Core number increase after multi threaded reading

6.4. Expected latency in HTTP live content streaming

My solution has an implementational glitch, since the solution requires to save 1s of video before it could send it forward, hence the delay is more than 1s. Albeit the large delay caused by bad implementation, the total minimum delay of the live content streaming via HTTP can estimate with the following equalation: $T_{tot} = T_{ssp} + T_{afs} + T_{hd} + T_b = 5d_{ms} + d_{link}$, where

d_{ms} – Segment Duration

T_{ssp} – Segmentation Delay, The server-side packetization is specific for segment-based streaming, and depends directly on the media segment duration, minimal packetization delay equal to the media segment duration: $T_{ssp} = d_{ms}$.

T_{afs} – Asynchronous fetch of media segments, the asynchronous fetch of the segments is due to that there is no synchronous signaling from the server that a new segment is ready, but the client must poll for data. To avoid unsuccessful fetches, the client needs to schedule the fetches some time period after the calculated ideal availability time, leading to a typical delay contribution: $T_{afs} = d_{ms}$

T_{hd} – HTTP download time, the HTTP download time does of course also matter. If the available bandwidth is used to a reasonable share, the download time of a media segment should be close to the segment duration time: $T_{hd} = d_{ms} + d_{link}$

T_b – Buffering in the client The buffering in the client is needed to provide a smooth media play-out, and hide transport jitter such a varying download time. To be more robust, it may be good to increase the buffer to 2 segments, leading to $T_b = 2\ d_{ms}$

d_{link} – Link duration

It is still necessary to select a value for d_{ms} given this formula. The best value depends on the sort of event, particularly the level of competition. Receivers of an HTTP streaming session shouldn't get the intended content significantly later than users of broadcast TV or a comparable technology. For RTP streaming, buffering and link delay are also present, hence the main delay difference in a straight comparison is $3d_{ms} - d_{RTP\_ps}$. The RTP packet serialization time, d_{RTP_ps}, is only a few tens of milliseconds here; for example, 20ms for a link at 160 kbps and a packet size of 1000 bytes. Therefore, a live HTTP streaming service has a roughly 3-second longer delay than the similar RTSP/RTP-based service when segment sizes are 1 second. [27]

Consider all of this, the approximately 1s delay is acceptable when the goal is to stream an event real-time, but not in our environment, where the avoidance of the accidence depends on couples of milliseconds. Hence my purpose is to find other solution which reduces the delay as much as possible. Albeit my hypothesis was that the Edge 5G Networks are able to host AI solutions and process the data flow seamlessly. With a bit performance optimalization a wide variety of use cases could be provided by the Edge UPF.

7. Conclusions and future plans

I built a system which is adequate for running AI modules in the edge 5G system, albeit the have not achieved the desired latency yet, I made a Proof-of-Concept architecture which is able select specific packets and run Deep Learning on them. I have chosen a solution which needs more processing than casual, and the system worked properly. In Chapter 3 I described a bunch of use-cases, which should be able to be hosted by my architecture. After I get a bit expert in edge 5G I would like to predict which are the beforementioned use-cases are implementable, and which are not (Table 6.4-1).

Table 6.4-1 Feasibility of the Use-case Scenarios

Use-Case	**Feasible?**	**Justification**
Enhance functional safety	YES	Albeit my video streaming solution was not getting the desired latency, in that solution we do not need to stream the video stream forward, just look at frames and send the important data in plain text.
Virtual sensors	YES	In that case we need a more complex Deep Learning Solution, but I describe it in that Chapter. That case that solution should be implemented in the Agent-Marci container.
Increasing the precision of indoor positioning	MODERATELY	With video processing more precise result can be achieve than with Ultra Sonic systems, the architecture is proper to process video streams, but it need to be effective in that scenario. But in case of intensive movement, it could be inaccurate, E.g. if the AGV moves with 1 m/s, 200 ms latency means 20cm inaccuracy.
Support cellular hints for mobile robots	YES	A Couple of research described its feasibility. [19]
Legacy robot control that is 5G compatible	MODERATELY	5G provides the desired low latency for the controlling, in the scenario where the Upstream is the robots feedback and the answer is the controlling command it would functioning properly. Albeit I would recommend to implement the controlling application (because its complexity) in the Edge-Cloud and use the Edge UPF solution only for supporting purposes.
Massive gathering of IoT data	YES	System is adequate for massive dataflow; DPI can forward the designated streams to further processing and the rest just pass through the system.
AI on the UE, somehow integrate AI into the UPF	NO	That scenario seems like a bit sci-fi from today's perspective, but when it get possible in the future I think it would be solvable by the architecture.

In the future I would like to decrease the latency of a stream as much as possible. I would not achieve as low latency as a native C code could, but that could be decreased further. My goals also contains that the demo scenario will be replaced with the real-life scenario. Hence, I move the whole architecture in a real 5G edge core, do not simulate anything, use real User Equipment, and send it back to real application. I also would like to write a REST API, which can control the Deep Learning Solution, and can get parameters, prediction results, etc. I also write the basics of that API, but I need to be done it to fit in 3GPP's standard.

To summarize the results the architecture could be host a wide variety of application and it opens a new direction in Smart Telecommunication System.

8. Table of Figures & Tables

9. List of abbreviations

The following table describes the significance of various abbreviations and acronyms used throughout the thesis. The page on which each one is defined or first used is also given. Nonstandard acronyms that are used in some places to abbreviate the names of certain white matter structures are not in this list.

Abbrevation	Meaning	Page
3GPP	3rd Generation Partnership Project	6
AOA	Angle of Arrival	17
AtLoc	Attention Guided Camera Localization	16
AGV	Automated Guided Vehicle	23
CUPS	Control and User Plane Separation	23
CP	Control Plane	6
CNN	Convolutional Neural Network	27
DN	Data Network	6
DPDK	Data Plane Development Kit	32
DLS	Deep Learning Solution	24
DNN	Deep Neural Network	13
DPI	Deep Packet Inspection	7
EI	Edge AI	13
FA	Factory automation	20
FR	Flight Rack	25
GPS	Global Positioning System	16
HPA	Horizontal Pod Autoscaler	11
IIOT	Industrial Internet of Things	5
ILS	Indoor Localization Systems	17
ISM	Industrial, Scientific, and Medical	20
IT	Information Technology	21
lR	Infrared	17
IoT	Internet of Things	5
Lidar	Light Detection and Ranging	5
LwM2M	Lightweight M2M	19
ML	Machine Learning	5
M2M	Machine-to-Machine	10
MBoxes	Middleboxes	22
mmWave	millimeter-wave	10
MIMO	multiple-input multiple-output	10
NAT	Network Address Translation	7
NaaS	Network as a Service	21
NWDAF	Network Data Analytics Function	19
NFV	Network Function Virtualization	21
OT	Operational Technology	21
PGW	Packet Network Data Gateway	7
PoC	Proof-of-Concept	27
PDU	Protocol Data Unit	6
QoS	Quality of Service	6

RAN	Radio Access Network	6
RF	Radio-Frequency	17
RFID	Radio-Frequency Identification	17
RSS	Received Signal Strength	17
SFC	Service Function Chaining	7
SF	Service Functions	7
SGW	Serving Gateway	7
SMF	Session Management Function	6
SDN	Software Defined Networks	7
SLAM	Simultaneous Localization and Mapping	7
TDOA	Time Difference of Arrival	17
TOA	Time of Arrival	17
URLLC	Ultra Reliable Low Latency Communications	19
US	Ultrasonic	17
UWB	Ultra-Wideband	17
UDM	Unified Data Management	6
UPF	User Plane Function	6
WLAN	wireless local area network	17

10. References

[1] e. a. Md. Asif-Ur-Rahman, Toward a Heterogeneous Mist, Fog, and, IEEE, 2019.

[2] A. V. G. F. M. M. M. A. H. N. R. S. R. Marcell Balogh, Cloud-Controlled Autonomous Mobile Robot, IEEE, 2021.

[3] TR 21.915, 3GPP, 2018.

[4] e. a. Patrick Marsch, 5G System Design: Architectural and Functional Considerations and Long Term Research, Wiley, 2018.

[5] R. M. a. J.-P. K. Edgar Ramos, Distributing Intelligence to the Edge and Beyond, IEEE, 2019.

[6] S. H. S. a. I. Yaqoob, A Survey: Internet of Things (IOT), IEEE, 2016.

[7] A. R. a. N. S. M. Agiwal, Next Generation 5G, IEEE, 2017.

[8] J. G. a. D. Reinsel, The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east, IDC, 2012.

[9] N. H. e. al., Ericsson mobility report, Ericsson AB, 2017.

[10] e. a. Zhi Zhou, Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing, IEEE, 2019.

[11] e. a. Bing Wang, AtLoc: Attention Guided Camera Localization, The Thirty-Fourth AAAI Conference on Artificial Intelligence, 2020.

[12] a. 1. C. B. Hofmann-Wellenho±: H. Lichtenegger, Global Positioning System. Theory and Practice, Springer-Verlag, 2001.

[13] J. H. a. G. Borriello, Location Systems for Ubiquitous Computing, 2001: IEEE.

[14] J. H. a. G. Borriello, Location sensing techniques, IEEE, 2001.

[15] e. a. Faheem Ijaz, Indoor Positioning: A Review of Indoor Ultrasonic Positioning systems, IEEE, 2013.

[16] e. a. M. Peter Heilbrun, Stereotactic Localization and Guidance Using a Machine Vision Technique, Karger AG, 1992.

[17] e. a. Salih Sevgican, Intelligent Network Data Analytics Function in 5G Cellular Networks using Machine Learning, IEEE, 2020.

[18] e. a. Mykola Beshley, A Self-Optimizing Technique Based on Vertical Handover for Load Balancing in Heterogeneous Wireless Networks Using Big Data Analytics, MDPI, 2021.

[19] e. a. Petar Popovski, Wireless Access for Ultra-Reliable Low-Latency Communication: Principles and Building Blocks, 2018, IEEE.

[20] C. A. L. P. a. F. J. Lin, Incorporating OMA Lightweight M2M protocol in IoT/M2M standard architecture, Milan, Italy: IEEE, 2016.

[21] e. a. Philipp Schulz, Latency Critical IoT Applications in 5G: Perspective on the Design of Radio Interface and Network Architecture, IEEE, 2017.

[22] FS_SMARTER — Critical Communications, 3GPP TR, 2016.

[23] B. H. e. al., Wireless Communication for Factory Auto- mation: An Opportunity for LTE and 5G Systems, IEEE, 2016.

[24] R. Zurawski, Industrial Communication Technology Hand- book, Second Edition, CRC Press, 2015.

[25] e. a. P. Quinn, Problem Statement for Service Function Chaining, IETF, 2015.

[26] I. S. a. G. E. H. A. Krizhevsky, „Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems,* 2012.

[27] T. E. P. F. G. M. K. Thorsten Lohmar, Dynamic Adaptive HTTP Streaming of Live Content, IEEE, 2011.

[28] X. Z. S. R. a. J. S. Kaiming He, „Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE,* 2015.

[29] R. G. J. D. T. D. J. Malik, „Rich feature hierarchies for accurate object detection and semantic segmentation," *UC Berkeley,* 2014.

[30] K. K. a. P. Krishnamurth, Properties of indoor received signal strength for WLAN location fingerprinting, 2004: IEEE.