H2020 5G Dive Project
Grant No. 859881

D3.2 Results of initial validation campaign of vertical use cases

Abstract

This deliverable of D3.2 is the second deliverable of WP3 after the first deliverable of D3.1. The goal of this deliverable is to provide the initial validation results of different 5G-DIVE use cases defined in D1.1, i.e., Digital Twin (DT), Zero Defect Manufacturing (ZDM) and Massive Machine-Type-of-Communication (mMTC) use cases for Industry 4.0 (I4.0) trial, and Drone Collision Avoidance System (DCAS) and Intelligent Image Processing for Drones (IIPFD) use cases for Autonomous Drone Scouting (ADS) trial. Experiments of each use case have been performed for implementation validation. And the experimental results are evaluated against the technical requirements defined in D1.1. Furthermore, the updated information of trial sites in Taiwan and the initial plan for use case integration are also provided in this deliverable.

Document properties

Document number	D3.2
Document title	Results of initial validation campaign of vertical use cases
Document responsible	EAB
Document editor	Chenguang Lu (EAB)
Editorial team	Samer Talat (ITRI), Chenguang Lu (EAB), Gyanesh Patra (EAB), Chao Zhang (ULUND), Per Ödling (ULUND), Saptarshi Hazra (RISE), Muhammad Febrian Ardiansyah (NCTU), Timothy William (NCTU), Tzu-Ya Wang(III), ChenHao Chiu (ITRI), June Liu (Askey), KJ Liu (Askey), Laura Caruso (UC3M), Milan Groshev (UC3M), Carlos Guimarães (UC3M), Andee Lin (ITRI), Ivan Paez (Adlink), Aitor Zabala (TELCA), Javier Sacido (TELCA), Matteo Pergolesi (TELCA), Ibrahim Hemadeh (IDCC), Filipe Conceicao (IDCC), Hergys Rexha (Abo Akademi University)
Target dissemination level	Public
Status of the document	Final
Version	1.0

Production properties

Reviewers	Antonio De la Oliva, Bengt Ahlgren, Ibrahim Hemadeh, Ivan Paez, Milan Groshev, Carlos Guimarães, Matteo Pergolesi, Samer Talat

Document history

Revision	Date	Issued by	Description
1.0	2021-02-28	Chenguang Lu	Final version

Disclaimer

This document has been produced in the context of the 5G Dive Project. The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement Nº H2020-859881.

Contents

List of Tables

List of Figures

List of Acronyms

3GPP	3rd Generation Partnership Project
4G	4th Generation
5G	5th Generation
5G NR	5th Generation New Radio
5G-DIVE	eDge Intelligence for Vertical Experimentation
AC	Alternating Current
ACK	Acknowledgement
ADS	Autonomous Drone Scout
AI	Artificial Intelligence
AP	Average Precision
API	Application Programming Interface
APPs	Applications
AWS	Amazon Web Services
BASS	Business Automation Support Stratum
CDF	Cumulative Distribution Function
CNN	convolutional neural networks
COVID-19	Coronavrus Disease 2019
CPE	Customer Premises Equipment
CPU	Central Processing Unit
CRAN	Cloud Radio Access Network
D1.1	Deliverable 1.1
DASS	Data Analytics Support Stratum
DC	Data Center
DCAS	Drone Collision Avoidance System
DEEP	5G-DIVE Elastic Edge Platform
DL	Downlink
DNS	Domain Name System
DT	Digital Twin
E2E	Exchange-to-Exchange
EAB	Ericsson AB
EagleEYE	Aerial Edge-enabled Disaster Relief Response System
EDC	Edge Data Center
EFS	Edge and Fog System
eMMB	Enhanced Mobile Broadband
eNB	Evolved Node Base Station
EPC	Evolved Packet Core
EuCNC	European Conference on Networks and Communications
E-UTRAN	Evolved UMTS Terrestrial Radio Access Network
FIM	Fog Infrastructure Manager
FOrcE	Fog Orchestration Manager
Gbps	Gigabytes per second
GGC	Greengrass Core
GHz	Gigahertz

gNB	5G NR base stations
GPS	Global Positioning System
GPU	Graphics processing unit
GUTI	Globally Unique Temporary Identifier
H2020	Horizon 2020
HD	High Definition
HTTP	Hypertext Transfer Protocol
HW	Hardware
I4.0	Industry 4.0
ID	Identity
IDCC	Interdigital Europe
IE	Intelligence Engine
IEEE	Institute of Electrical and Electronics Engineers
IESS	Intelligence Engine Support Stratum
III	Institute for Information Industry
IIPFD	Intelligent Image Processing for Drones
IMDEA	Madrid Institute for Advanced Studies
IMSI	International Mobile Subscriber Identity
IoT	Internet of Things
IP	Internet Protocol
ITRI	Industrial Technology Research Institute
JSON	JavaScript Object Notation
K8S	Kubernetes
KPI	Key Performance Indicator
LED	Light Emitting Diode
LoRa	Long Range
LTE	Long Term Evolution
LTS	Long Term Support
LWM2M	Lightweight M2M standard
MAC	Medium Access Control
mAP	mean Average Precision
Mbps	Megabits per second
MiniPC	Minicomputer
MIRC	Microelectronics and Information Systems Research Center
ML	Machine Learning
MME	Mobility Management Entity
mMTC	Massive Machine Type of Communication
MQTT	Message Queuing Telemetry Transport
MTC	Machine Type Communications
NBI	Northbound interface
NCTU	National Chiao Tung University
NDI	Network Device Interface
NR	New Radio
NSA	Non-standalone
OCS	Orchestration and Control System

OPTUNS	SDN-based optical-tunnel-network system
PER	Packet Error Rate
PHY	Physical (Layer)
PiH	Person in need of a help
PoC	Proof of Concept
PTZ	Pan, Tilt and Zoom
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
RAN	Radio Access Network
RAT	Radio Access Technology
RF	Radio Frequency
ROS	Robot Operating System
RSRP	Reference Signal Received Power
RSSI	Received Signal Sterngth Indicator
RTSP	Real Time Streaming Protocol
RTT	Round Trip Time
SA	Standalone
SCTP	Stream Control Transmission Protocol
SFP	Small Form-factor Pluggable
Sgw	Serving Gateway
SINR	Signal to interference plus noise radio
SQL	Structured Query Language
TBD	To Be Determined
TCP	Transmission Control Protocol
TED	Test Dataset
TELCA	Telcaria S.A
TLS	Transport Layer Security
TW	Trained Weight
UAV	Unmanned Aerial Vehicle
UC	Use Case
UC3M	University Carlos III of Madrid
UDP	User Datagram Protocol
UE	User Equipment
UI	User Interface
UL	Uplink
ULUND	Lund University
URLLC	Ultra-reliable low-latency communication
USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
VM	Virtual Machine
VPN	Virtual Private Network
WAN	Wide Area Newtwork
Wi-Fi	Wireless Fidelity
WP	Work Package

ZDM	Zero Defect Manufacturing
ZMQ	ZeroMQ

Executive Summary

This deliverable of D3.2 is the second deliverable of WP3 after D3.1 [1]. The goal of this deliverable is to provide the initial validation results of different 5G-DIVE use cases defined in D1.1 [2], i.e., Digital Twin (DT), Zero Defect Manufacturing (ZDM) and Massive Machine-Type-of-Communication (mMTC) use cases for Industry 4.0 (I4.0) trial, and Drone Collision Avoidance System (DCAS) and Intelligent Image Processing for Drones (IIPFD) use cases for Autonomous Drone Scouting (ADS) trial. The main achievements of D3.2 are: (1) Experiments of each use case have been performed for use-case implementation validation. The experimental results are evaluated against the technical requirements defined in D1.1, which indicates that the developments of the use cases are all on track. (2) The initial plan for use case integration are provided, presenting the basic ideas how all use cases can be integrated under one common DEEP platform (5G-DIVE Elastic Edge Platform). The following provides a short summary of the contents in Section 2-6, respectively.

In WP3, the I4.0 and ADS trials are planned in Taiwan. However, this plan may get affected by the current COVID-19 pandemic situation which may last the whole 2021. Therefore, in Section 2, we provide the information regarding such risks. Basically, an alternative plan is prepared for I4.0 trial, which can be performed in 5TONIC lab in Spain, in case that travels to Taiwan would not be possible from European partners or the time when they are allowed makes impossible the installation of the equipment..

In Section 3, the initial experimental validation results are presented for the three use cases planned for the I4.0 trial. The results indicate that the development of three use cases are all on track. The DT use case has done 3 experiments with 4G and 5G network regarding synchronization performance between twins, impact from traffic load and adaptive control-loop configuration. The results show that 5G is needed for DT to work properly, while 4G is insufficient due to too large latency and jitters. The ZDM use case presented two experimental results of 4G vs Wi-Fi connectivity benchmarks, as well as MQTT vs HTTP for the designed telemetry solution. It also shows that 4G doesn't fulfil the latency requirement of ZDM and indicates that 5G is required. The mMTC use case has done experiments regarding cloud native design of IEEE 802.15.4 and LoRa. The experiments show that resource can be more efficiently utilized in these two cases, respectively. The network performance regarding bit rate and reliability is not affected, while latency is increased due to resource pooling, but at an acceptable level for mMTC IoT applications.

Section 4 presents the initial validation results for the two use cases planned for the ADS trial. The validation was done first through five experiments in terms of latency and throughput, regarding different network components of wireless connectivity of 4G between drone and network and Wi-Fi between drones, NCTU's high-performance Edge data center named OPTUNS, iMEC server hosting network Core functions like serving gateway, and 5G EPC supporting non-standalone (NSA) mode. Further, the application-level performance of two use cases, i.e., DCAS and IIPFD use cases, were evaluated through experiments. The experimental results show all components developed so far are working well as expected. It also shows that 4G is insufficient to fulfil the ADS use-case requirements defined in WP1, with respect to latency and throughput. 5G-NSA is expected to meet these requirements, which will be evaluated in the final trial activities.

In Section 5, an update regarding the trial site information is first provided for both I4.0 and ADS trials. Then, we present the initial plan of use-case integration regarding how all 5G-DIVE use cases will be integrated under one common DEEP platform. The main features are (1) BASS will develop orchestrator drivers to support different orchestrator frameworks (e.g., K8s and FogO5) which are used by different use cases; (2) IESS provides a common IESS catalog for storing use-case specific ML/AI models; (3) all use cases will adopt a common DASS component using Eclipse Zenoh developed in 5G-DIVE. In this way, a common DEEP platform can serve multiple vertical use cases simultaneously.

Finally, Section 6 concludes the deliverable.

1. Introduction

The last WP3 deliverable (D3.1 [1]) described the trial sites for the Industry 4.0 (I4.0) and Autonomous Drone Scouting (ADS) trials and laid out the initial trial plan. In this deliverable (D3.2), the focus is on presenting the initial validation results through experiments for all use cases, as well as provide the initial plan for integrating all use cases together with one DEEP platform. The rest of the deliverable is organized as follows.

In Section 2, the risks and impacts due to current COVID-19 pandemic are discussed. I4.0 trial in Taiwan may be affected due to the travel restrictions for European partners. A backup plan is discussed to have the I4.0 trial in 5TONIC lab in Madrid, Spain, instead.

Section 3 presents the testbed setup and the initial validation results of three use cases, i.e., Digital Twin (DT), Zero Defect Manufacturing (ZDM) and Massive Machine-Type-of-Communication (mMTC) use cases, which are planned for the I4.0 trial. Each use case has done various experiments for performance validation of the testbed design. The obtained experimental results are evaluated against the technical requirements defined in D1.1 [2].

Section 4 presents the testbed setup and the initial validation results of two ADS use cases, i.e., Drone Collision Avoidance System (DCAS) and Intelligent Image Processing for Drones (IIPFD) use cases. The performance is evaluated by experiments, both on component and application levels. The technical requirements defined for ADS use cases in D1.1 are evaluated based on the experimental results.

In Section 5, the trial site description is updated with latest information for both I4.0 and ADS trials in Taiwan. Further, an initial plan of use-case integration is provided, describing the way to integrate all use cases with a common DEEP platform.

Finally, Section 6 concludes the deliverable with conclusions.

2. Risks due to COVID-19 pandemic

The current pandemic situation is not relieved. Most of European partners are still having working-from-home policy due to the new waves of COVID-19. Travel restrictions are still in place. Only essential travels can be approved by executive level of management. Another complication is that 14 days quarantine is required when entering Taiwan (and other countries) from abroad. Given the latest information, the pandemic is likely not to be over in 2021. Travel restrictions of European partners may not be lifted in the whole year. These pose high risks to the WP3 works regarding the planned trials in Taiwan in the second half of 2021.

The risk level of ADS trial in Taiwan is low because all partners involved are from Taiwan and there seems no travel restrictions within Taiwan. However, the risk level is high for the I4.0 trial in Taiwan since all involved partners are from Europe and they may not be able to travel to Taiwan in time due to the reasons mentioned above. To manage the risks without having to cancel the whole I4.0 trial plan, a backup plan has been discussed that the I4.0 trial can be performed in 5TONIC facility in Madrid, Spain, instead. It looks like a viable solution in case travels to Taiwan would be impossible or too late. Travels within Europe seem easier to manage for European partners. As a result, we will prepare both possibilities for the time being and will make a decision soon where to perform the final I4.0 trial activities.

3. Initial validation results of I4.0 use cases

In this section, we present the initial validation results of the three I4.0 use cases, regarding network and system performances, such as throughput, latency, resource utilization and robot movement accuracy etc. Sections 3.1, 3.2 and 3.3 present the current testbed setup and experimental results of Digital Twin, ZDM and mMTC use cases, respectively. In Section 3.4, the initial evaluation against the KPI described in D1.1 is provided for all three use cases based on the experiment results.

3.1. I4.0-UC1: Digital Twin

In this section, we present the results of the Digital Twin (DT) use case, a virtual application which represents the replica of a robotic arm. The application enables teleoperation by a remote user who can monitor the movements of the real robot and perform other operations. The validation of the use case was conducted against a real experimental setup. First, we describe the 5TONIC lab configuration and equipment. Then, we illustrate a set of three experiments which highlight the benefits brought by 5G connectivity and by edge computing to the performance of the robot control and to its digital replica, fulfilling the KPI requirements identified in D1.1 [2].

3.1.1. Testbed Setup and 5G connectivity

The location of our experimental testbed is the 5TONIC Laboratory of the IMDEA Networks Institute (Madrid, Spain) [3]. Figure 2 illustrates the testbed setup comprising of the following equipment:

- Fog Devices:
  - Niryo One Robotic manipulator: 6-axis robotic arm powered by ROS-1
  - MiniPC: for running control and Digital Twin interface
- 4G/5G connectivity:
  - 5G base station: Ericsson BB 6630 baseband and AIR 6488 radio
  - 5G CPE: Huawei Pro Balong 5000, with an Ethernet interface to the robot
  - 4G base station: Ericsson BB 5216 baseband and Radio 2203 with integrated antenna
  - 4G CPE: Huawei B315s-22, with an Ethernet interface to the robot
- Edge: DELL PowerEdge R430 Rack Server [4], hosting the Digital Twin App VMs (2 vCPUs and 4GB of RAM)
- Cloud: emulated by adding artificial delay (7 ms, which was the time needed to reach Google DNS from the network, thus a good indicator of the typical cloud latency) to the link between the 4G/5G cores and the edge DC using a traffic control utility (Linux tc)

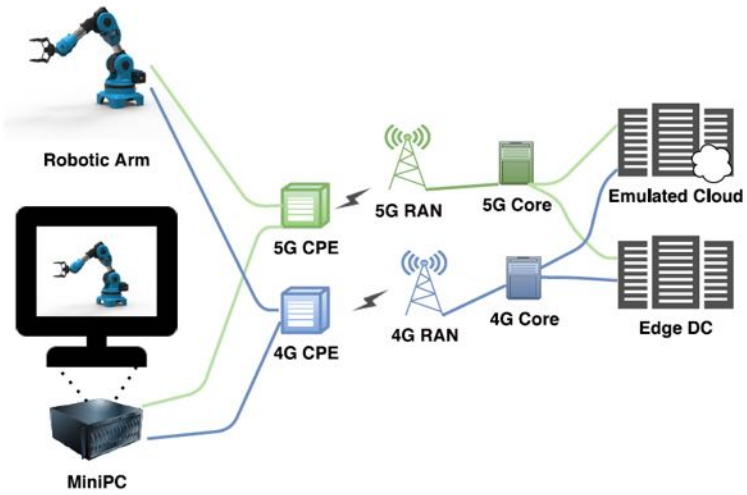

FIGURE 1 DIGITAL TWIN TESTBED SETUP

Measurements have been done to compare the network performance of 4G and 5G network: unlike 5G, 4G connectivity could not fulfill the initial requirements, proving that 5G technology is essential for the correct functioning of the use case. The results regarding latency (average value (± std deviation value)), packet loss (%) and throughput (average value (± std deviation value)) are listed in Table 1. Currently, only 5G NSA is supported in the testbed. It will be upgraded to 5G SA later.

TABLE 1 MEASUREMENT RESULTS OF 4G AND 5G NETWORK

Measurement	4G	5G (NSA)
RTT Latency	23.88 (± 5.84) ms	6.56 (± 1.04) ms
Jitter	2.32 (± 0.35) ms	0.46 (± 0.18) ms
Packet Loss	0 %	0 %
UL throughput	44 (± 0.20) Mbps	96 (± 1.81) Mbps
DL throughput	72 (± 1.04) Mbps	600 (± 13.50) Mbps

3.1.2. Experimental Results

The following experiments are aimed at evaluating the control performance of the teleoperated robotic arm and the accuracy of the fog computing Digital Twin (DT) application. Three experiments were conducted to validate the system with respect to different connectivity (4G and 5G) and architectural configurations (edge technology vs cloud):

A. Synchronization test: accuracy of synchronization between the robotic arm movements and those of the Digital Twin.
B. System load: impact of link cross traffic on the synchronization performance.

C. Control-loop: adapting the robot control-loop parameter to network conditions in order to preserve the synchronization performance.

3.1.2.1. Experiment A: synchronization test

The purpose of this experiment is to evaluate how the *twins* (i.e., the physical robot and its digital replica) synchronize their movements in different scenarios. A performance comparison was conducted considering three variables: *(i)* network connectivity (Net): 4G vs 5G; *(ii)* location of the Digital Twin application (DT): Edge vs Cloud; and *(iii)* location of the ROS computing stack (Compute): Robot vs Edge. Measurements were carried out for five (1. 5G-Edge-Edge, 2. 4G-Edge-Robot, 3. 4G-Cloud-Robot, 4. 5G-Edge-Robot, 5. 4G-Cloud-Robot) different configurations using the different combinations of these options.

To evaluate the synchronization performances, we sent a continuous stream of commands from the Digital Twin application to the physical robotic arm and in the meantime analyzed the joint states vector (the 6-axis vector representing the relative position of a robot) of both twins.

Then, we computed the distance of the two robots from the axis origin, took their ratio and evaluated how this parameter changed over time. The percentage ratio indicates the goodness of the synchronization procedure.

Figure 2 shows the synchronization results in different cases. In the configuration option where computing stack is located in the robot, the Digital Twin app is able to fully synchronize with its physical counterpart when using 5G (green line), which was impossible when relying on 4G (red line). Moreover, due to latency, 4G connectivity could not enable the computational offloading to the edge. If an instruction is not processed within the control-loop time (which defines the lifespan of an instruction), it is simply discarded. For this reason, the case 4G-Edge-Edge was neglected, and we only considered 5G-Edge-Edge. Performance level were kept in the case computation was moved to the edge, the robot worked smoothly despite its computational load being relocated in the edge server (orange line).

The cloud solutions, i.e., 5G-Cloud-Robot (blue line) and 4G-Cloud-Robot (purple line), were not feasible. The high latency caused the robot to behave unpredictably (when using 5G) or to not move at all (when using 4G).

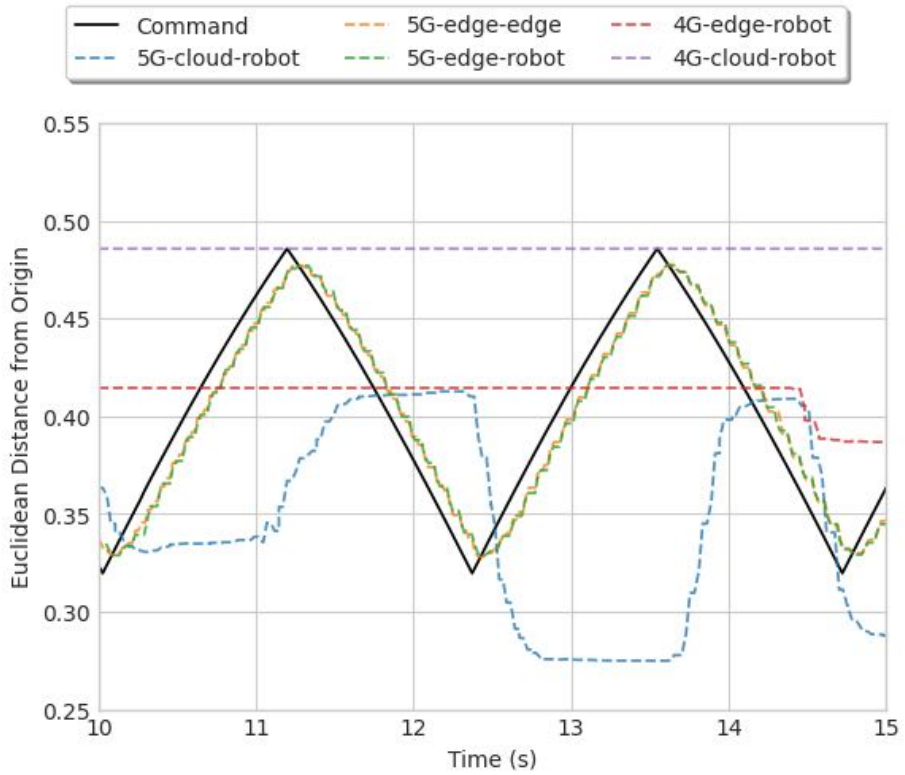

FIGURE 2 SYNCHRONIZATION RESULTS

3.1.2.2. Experiment B: system load

We repeated the synchronization experiment in case of putting a traffic load on the wireless link. The aim was to evaluate the loading limit. The impact of injecting four different traffic loads (i.e., 5Mbps, 10Mbps, 25Mbps, 50Mbps, and 75Mbps) at the uplink of the wireless link was evaluated.

Figure 3 depicts again the synchronization accuracy between the robotic arm and its digital replica for different system loads. As the load increases, the synchronization quality worsens at some point. We can refer to this breaking point as the saturation point of the link. A high impact on the synchronization between the physical and digital robot can be witnessed at approximately 75Mbps for 5G.

If we could leverage on 5G SA core network together with the slicing of the RAN, different QoS levels could be exploited and the digital twin delay-sensitive traffic (URLLC type) could be prioritized over the background traffic (eMMB type), ensuring the correct operation of Digital Twin even under link saturation.

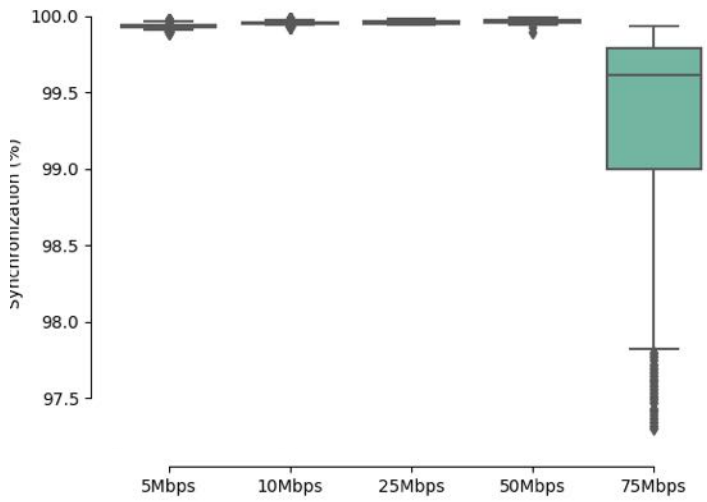

FIGURE 3 SYNCHRONIZATION ACCURACY BETWEEN ROBOTIC ARM AND DIGITAL TWIN REPLICA USING 5G NSA

3.1.2.3. Experiment C: adaptive control-loop configuration

With this experiment, we showed the capability of the control-loop parameter to self-adapt upon changes in the network, in order to guarantee a minimum Quality of Experience (QoE) level.

First, we introduced an artificial delay in the communication link between the DT App and the robot. Then, we exploited the Control-Loop Optimization module, which we implemented as part of the analytic stack, that is the intelligence layer (IEES) to be integrated in the next phase of the project, to adapt the control-loop according to the measured RTT and following a threshold-based mechanism. The algorithm could be further enhanced with AI/ML or statistics method involving other parameters (e.g., latency, jitter, packet loss, etc.).

We compared the synchronization performance and then the travelled distance of the physical robot w.r.t the control-loop duration. The travelled distance is the sum of the moved distance at every command execution. The larger the number of commands executed by the robot is, the longer the cumulative travelled distance gets. Moreover, the smaller the control-loop, the larger the number of executed commands that fits into a given amount of time. This explains the correlation between the control-loop duration and the total travelled distance: when the control-loop time is short, the robot changes position faster and covers more distance.

From the plot in Figure 5, we can see that if a certain amount of latency (from approximately 10 ms to 90 ms) is introduced due to a change in the network conditions w.r.t. a fixed control-loop of 20 ms and no adaptive mechanism is applied, the system suffers from poor synchronization performance (which means some commands are discarded). On the other hand, if an adaptive optimization mechanism is adopted by automatically enlarging the control-loop according to the increasing latency, the DT can keep the synchronization performance, with the price to pay being a slight decrease in the speed of the robotic movements.

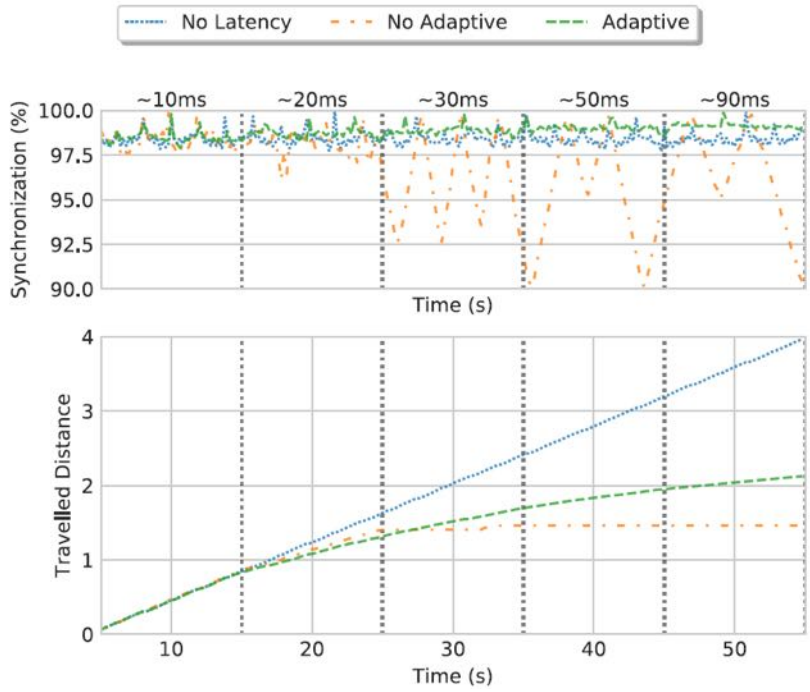

FIGURE 4 ADAPTIVE CONTROL-LOOP

3.1.3. BASS integration experiences

DT use case has completed the first integration efforts with the BASS, which provides support to Kubernetes at this stage. Basically, BASS provides a common platform for vertical operators to manage the DT use case. There has been a three step/phase integration of the Digital Twin use case components:

1. Containerization (Docker) phase: first implementation of the PoC, easy to rapidly create prototypes with a low overhead and a strong focus on the applications.
2. Resource Orchestrator phase: from Docker the PoC was migrated to Kubernetes.
3. BASS phase: Once the scenario was validated in Kubernetes, the BASS Vertical Service Descriptor was created to deploy the service through the BASS.

The main challenges overcome, and lessons learned have been:

- Consistent deployment of the whole service components in phase I was a challenge, as in the Digital Twin use-case the components are tightly coupled, and a consistent deployment was required. In order to correctly deploy the use case, we identified the need to specify the deployment conditions for each of the entities present in the stack. For example, some of the entities needed to launch specific ROS topics, which other entities needed to connect to before deploying, leading to a set of complex deployment conditions, which in phase I we could not handle. Consistent deployment was solved in phase II by using Kubernetes initContainers, allowing to configure the conditions of when a component should start instantiating.
- The deployment with specific parameters (static IP addresses, environment variables, DNS configuration, mounting devices) in phase I, have been a limiting factor when migrating the scenario to a different environment. When moving to phase II the deployment used native DNS in Kubernetes, which is capable of exposing the components to the network by their name and

removing static network parameters. Support for mounting devices was also added to allow access to the robot physical interface.

- Complexity of Kubernetes descriptors. Although the Kubernetes API is very flexible when defining a deployment, it can also overwhelm the developer with many choices. During Phase II the digital twin developers required assistance from Kubernetes experienced developers. This necessity of simplifying the deployments without losing flexibility was taken into account as a feature to include in the BASS. At phase III the digital twin development team, only required some minor assistance to generate the Vertical Service Descriptor. When the BASS NBI (Northbound interface) documentation will be completed, we expect that the Vertical developing the descriptor will not require of any assistance.

3.1.4. Next steps

We plan to integrate our stack with the fog05 platform, so that we will have part of the Digital Twin containers working on an E2E system involving a Fog Orchestration Manager (named FOrcE in fog05) and a Fog Infrastructure Manager (FIM) forming the OCS platform as per project requirements. This would require a careful analysis of how the Digital Twin modules can be distributed along the Cloud-to-things continuum (Fog, Edge and Cloud), as we can have different scenarios with different performances.

The Analytic stack, that is the applications making up the IESS, must be developed in the next stage of the project. The main challenge will be to process the data coming from the robot and apply AI/ML and statistical methods in order to implement one or more of the following features: movement prediction, task learning, predictive maintenance, control-loop optimization. In particular, the optimization methods of the control-loop will be improved evaluating its correlation with other network metrics not considered at this stage, such as delay, jitter and packet loss.

Eventually, the IESS will need to be integrated with the other components of the DEEP, namely the BASS and the DASS.

3.2. I4.0-UC2: ZDM

In this section, we present the results of the Zero Defect Manufacturing (ZDM) use case validation against a real experimental setup. The purpose of the ZDM use case is to demonstrate the feasibility of AI based defect detection over the network, for the products coming out of a production line in the factory side. In this experiment, the latency and bandwidth measurements suggest that 4G is unsuitable for executing the ZDM use case, while Wi-Fi can provide a baseline for next step testing with 5G. Finally, the experimental setup is detailed as follows.

3.2.1. Experiment Setup

The ZDM experiment took place at InterDigital office in London, UK. This experiment has a three-part setup, a Production Line side, an Edge side, and a Cloud side, as illustrated in Figure 5.

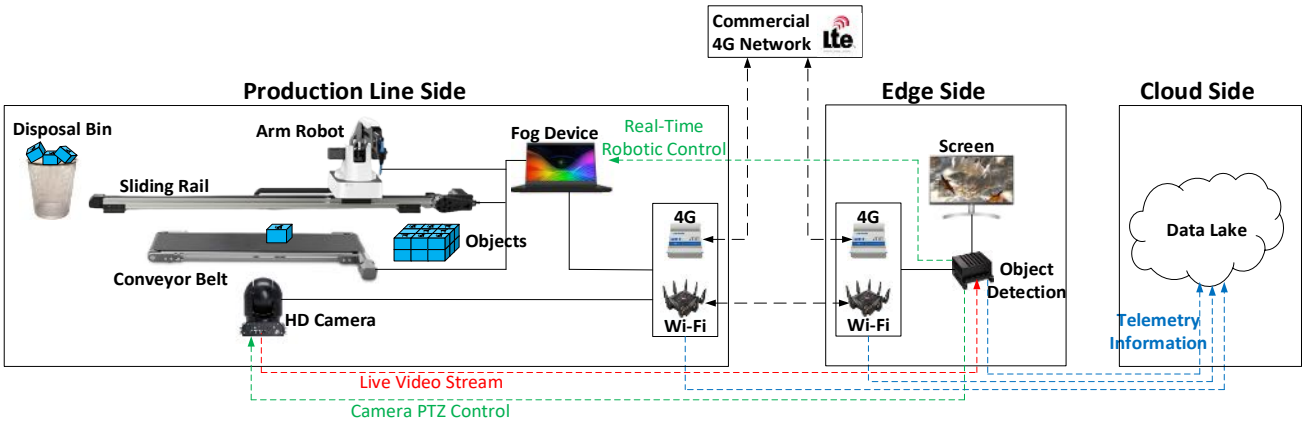

FIGURE 5 ZDM USE CASE SETUP.

In this experiment we used the following equipment:

- Production Line side: This part of the setup includes end-to-end production line components:
  - Dobot Magician robot arm: 4-axis robotic arm controlled using Python
  - Dobot Conveyor Belt: simulates a complete production line
  - Dobot Sliding Rail: high-precision linear rail
  - Cubes: with stickers of products
  - P200 Birddog Camera: runs the Network Device Interface (NDI) protocol with high resolution (1080P60 and 1080P30) with ±350-degree continuous pan, ±120-degree continuous tilt and 30X optical zoom. This camera is used for live streaming the production process to the Edge device. The camera is used for streaming a live video to the edge side over the network, as shown in the red dashed lines in Figure 5. Similarly, the green dashed lines between the edge device and the camera portrays the control commands sent from the edge side to the camera over the network.
  - Alienware Laptop: Fog device running the Python code that controls the arm, conveyor belt and sliding rail. It also exchanges commands with the Edge device.
- Edge side: The edge side is used here mainly for applying the object detection using the live video streamed by the camera over the 4G/Wi-Fi connection.
  - Jetson AGX XAVIER: GPU workstation that deploys AI-powered object detection software (YOLOv3) for detecting defect objects from the video streamed from the Production Line side. The edge device also sends commands to the fog device over the network, as shown by the green dashed lines between the edge and fog device in Figure 5.
- Connectivity: In this setup, we use 4G and Wi-Fi technologies to establish connection between the production line side and the edge side.
  - Teltonika RUTX11 4G modem (two modems): Cat 6 with DL up to 300 Mbps. The connection between the edge side and the production line side is established through the commercial 4G network.
  - ASUS ROG AX1100 router (two routers): Tri-band 802.11ax Wi-Fi 6 10 Gigabit gaming router. Here, the connection between the edge side and the production line side is established using a point-to-point connection between the two routers.

- Cloud:
  - AWS cloud: for storing telemetry data. The telemetry data is sent from different devices, such as the modems, routers and edge device, as shown by the blue dashed lines in Figure 5.

3.2.2. Experiment flow

The purpose of the ZDM use case is to demonstrate the feasibility of AI based defect detection over a 5G commercial network, for the products coming out of a production line in the factory side [1]. At the factory side, where the real setup is shown in Figure 6, products come out at the last stage of production, rolling on a conveyor belt. Some of these products can be defective and hence, the process is supervised at the Edge side and, if a defect is detected, the Edge node instructs the factory through the Fog node to perform actions.

FIGURE 6 FACTORY SIDE SETUP.

To this goal, the camera is deployed in a location where it can record the factory products rolling out of the factory in the conveyor belt, cubes are used for emulating products using stickers as seen in Figure 7. The video is streamed over to the Edge side, through one of the options for wireless connectivity (i.e., Wi-Fi and 4G).

FIGURE 7 CUBES ARE USED FOR EMULATING PRODUCTS USING STICKERS.

Once received at the Edge side, the edge node, which is the Xavier GPU workstation, captures the live stream, as shown in Figure 8 and runs through the YOLOv3 algorithm to perform object detection. YOLOv3 [5] was pre-trained to perform object detection of images of everyday objects, and it was pre-trained to consider an image of a bottle as a defective object. The Edge side is capable of sending commands to the Fog device (i.e., Alienware laptop), which is directly connected to the robot arm at the factory side. These commands are sent to the Fog device in real-time, in case a bottle object is detected (i.e., a bottle sticker is detected), so that the robot arm is triggered to remove it from the conveyor belt and dump it into the disposal bin.

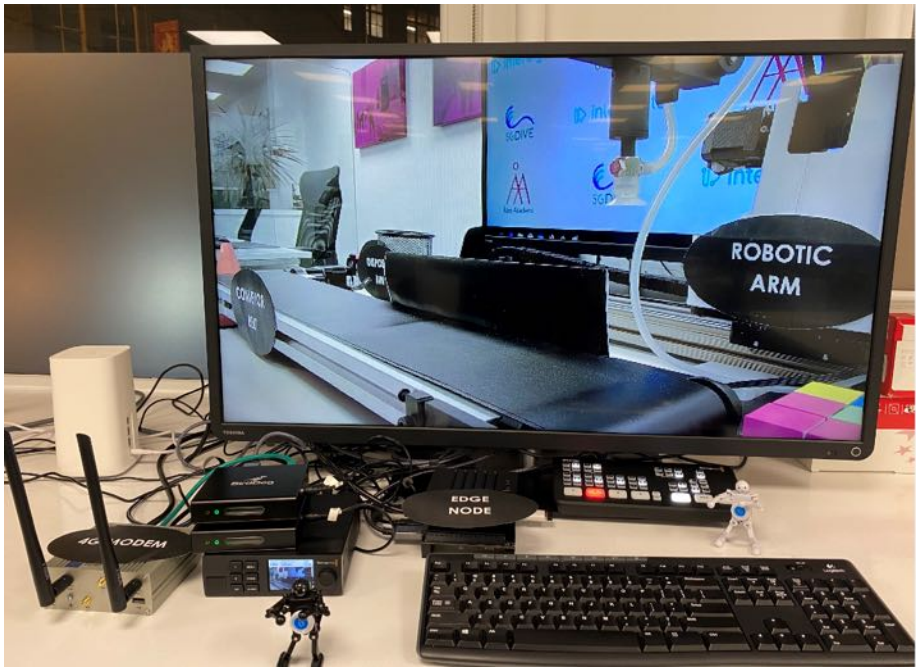

FIGURE 8 EDGE SIDE SETUP.

While the described supervision system and process runs, various telemetry data are collected from the connectivity devices and the Edge node HW. This data is stored in a cloud-based data lake solution for future processing and intelligence generation.

3.2.3. Telemetry solution

In this section, we present the telemetry solution that serves as a telemetry data framework that collects and stores information from different sources, such as the 4G modem, the object detection app and the edge node hardware. This telemetry framework is used for object detection applications and for future intelligent engines.

In order to train the new Intelligent Engines that will be deployed at the Edge side, as proposed in Section 3.2.5, telemetry data needs to be transported to the Cloud for training and retraining purposes. This communication would introduce an extra delay to the training and retraining processes. It is therefore important to quantify the introduced delay.

A large amount of data is required from different components including application, edge platform and network, where single-sourced and static data acquisition methods cannot meet the data requirements. It is desirable to have a framework that integrates multiple telemetry approaches from different components. Therefore, a telemetry framework is proposed which consists in two parts: the edge part which is described on the left of Figure 9 and the cloud part which is on the right side. At the edge side of the schematic, we have a GPU-based platform employed on the Jetson Xavier. The platform is hosting an intelligent application which uses a convolutional neural network (CNN) for performing real-time video inference, and an agent which is collecting several metrics listed on the table below, from the application, platform, and network, which is called the telemetry agent. Metrics of various components of the platform are collected and formatted as a JSON object and sent to the other part of the framework. On the cloud part the data is analysed, and actions are taken as a feedback controlling the behaviour of the intelligence performed on the edge side.

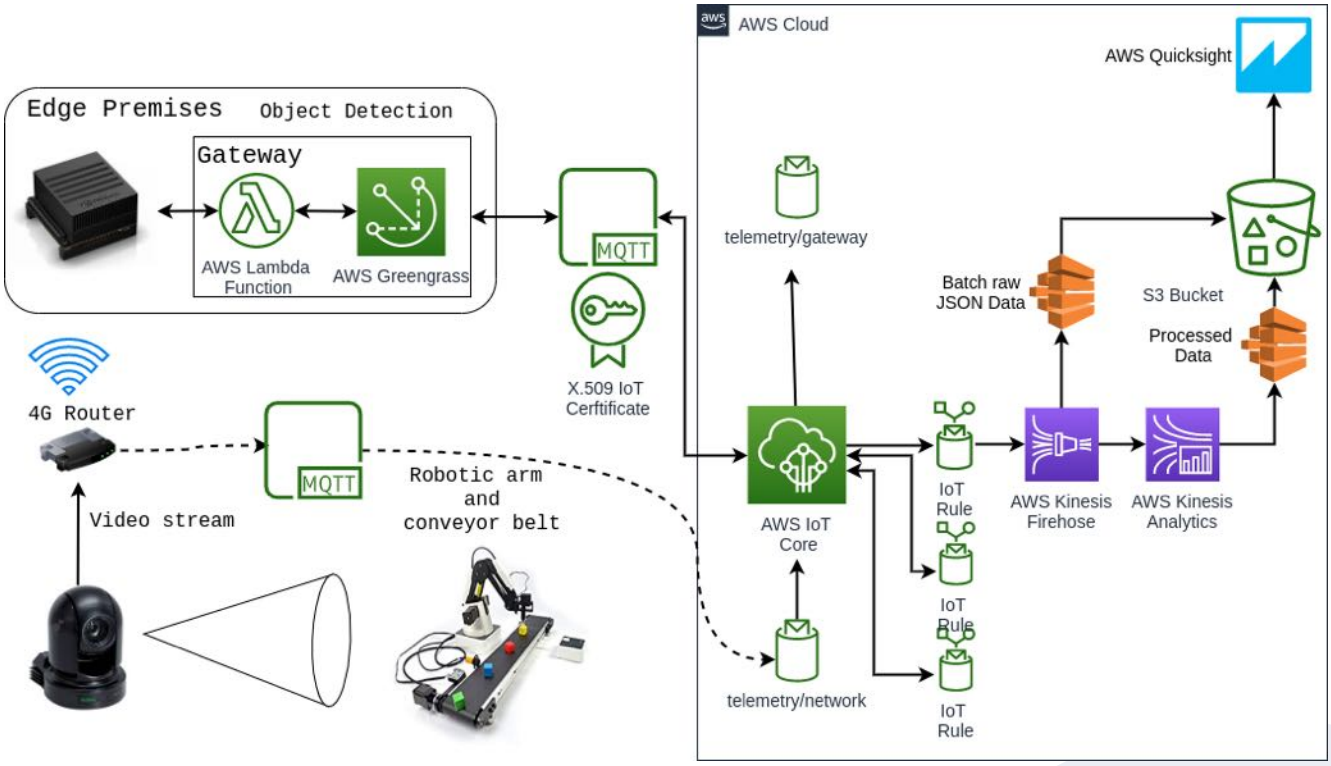

FIGURE 9 PRACTICAL IMPLEMENTATION OF THE TELEMETRY FRAMEWORK

The practical implementation used is based in the Amazon Web Services (AWS) cloud environment, as illustrated in Figure 9. Below there is the list of the components and the services.

- AWS IoT Core: cloud platform for IoT device data.
- AWS Kinesis Firehose: data transporting service from IoT Core to other AWS services.
- AWS Kinesis Analytics: data processing service for refinement of the IoT data.
- AWS Lambda: serverless computing function.
- AWS Simple Storage Service (S3): object storage service in the cloud.
- AWS IoT Greengrass: software that extends cloud services to edge devices.

In Table 2 below are listed telemetry parameters collected from the Modem, Edge node and Application. These parameters are stored at the cloud side using the JSON format, as shown in Figure 10.

TABLE 2 LIST OF TELEMETRY PARAMETERS COLLECTED FROM THE MODEM, EDGE NODE AND APPLICATION

Application	Edge Node – Jetson	4G Modem
Frame ID	CPU Utilization(%)	Modem ID
Timestamp	GPU Utilization(%)	RSSI(dbm)
FPS	Memory Usage(MB)	RSRP(dbm)

objects[{“class_id”: ,
“name”:
 	
“relative_coordinates”:
{
“x”:
“y”:
“width”:
“height”:
}
“confidence”:
}
]
	Temperature CPU(°C)	SINR(dbm)

	Temperature GPU(°C)	Modem Temperature(°C)
	Power GPU(mW)	Uplink data rate(Mbps)
	Power Platform(mW)	Downlink data rate(Mbps)
	Swap size(MB)	
	Timestamp	
	Nvidia Encoder, Decoder status(ON/OFF)	

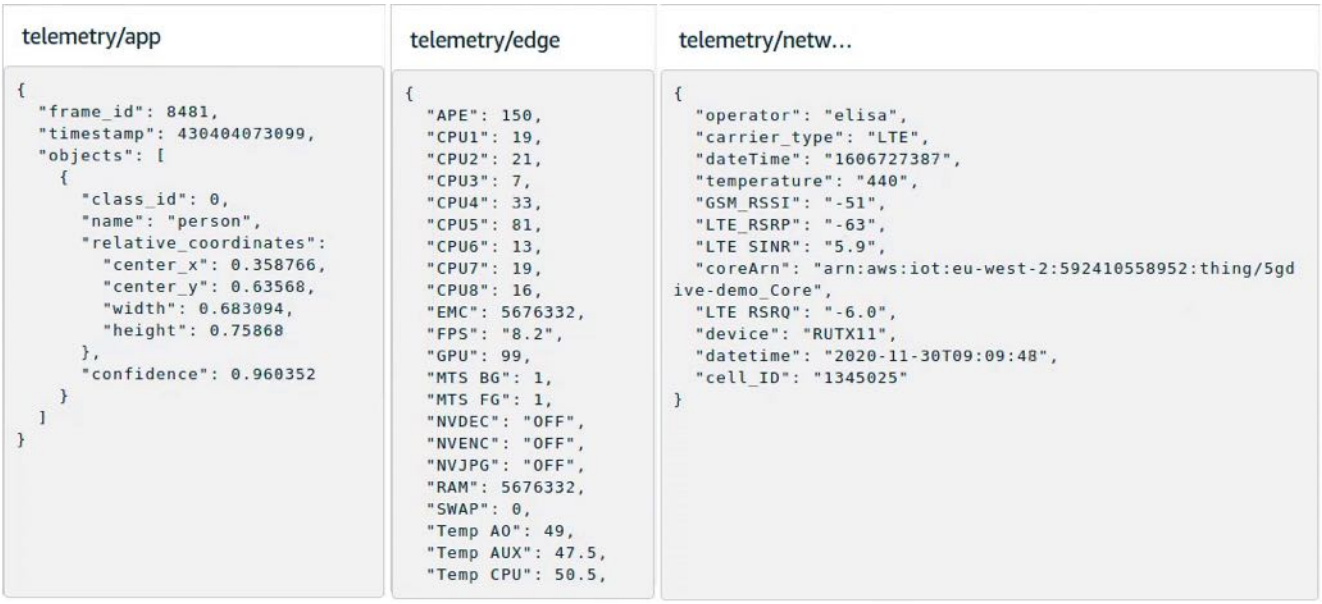

FIGURE 10 SCREENSHOT OF TELEMETRY DATA FROM APPLICATION, HARDWARE AND NETWORK (FROM LEFT TO RIGHT)

The workflow process of the cloud-based telemetry solution is as follows: at first, on the edge device shown in Figure 9 the telemetry agent (i.e., a python script) is running and collecting metrics from the application (i.e., the object detection application), AI neural network (i.e., YOLOv3) and hardware platform (i.e., 4G modems and the edge device). The collected information is packed into a JSON object and sent to the Greengrass core (GGC), located at the edge side, as seen in Figure 9. The GGC is registered with AWS IoT Core on the cloud side and uses the MQTT protocol to forward the JSON objects to the cloud. The IoT Core of Figure 9 provides a Device Gateway, which manages active device connections and a powerful Message Broker, which routes the messages with low latency. Furthermore, the AWS IoT Core provides mutual authentication and encryption, ensuring all data is exchanged between AWS and the devices are secure. All data is sent securely using Transport Layer Security (TLS) 1.2 with X.509 digital certificates on port 443. Once a telemetry data message is received by the IoT core we use AWS IoT Rules to send messages to AWS Kinesis Firehose delivery stream, one stream sends data to the S3 bucket which stores the raw version of the telemetry, and another stream goes to AWS Kinesis Analytics that through Structured Query Language (SQL) statements does a processing of the data like aggregating metrics in intervals of time, computing statistics, or histograms. Then after the data processing is done, transfer of the results happens to the bucket in CSV format.

3.2.4. Experimental results

In this section, we present the results obtained from the two experiments executed for the ZDM use case at InterDigital London Labs, namely Experiment A (wireless connectivity) and Experiment B (telemetry solution), in order to validate the feasibility of AI based defect detection over the network, for the products coming out of a production line in the factory side

3.2.4.1. Experiment A – wireless connectivity

This experiment was executed over two phases, namely Phase 1 and Phase 2. In Phase 1, both the Factory and Edge sides have a single Wi-Fi access point in premises. Both access points are directly connected, i.e., bridged together using a point-to-point topology, in order to establish communication between both sides. An Open VPN solution is used to interconnect the link layer of both local networks.

RTT and bandwidth measurements were conducted between the laptop at the production line side and the Xavier workstation at the edge side. RTT measurements were carried out using command ping from the command line console over 15 minutes (app. 900 pings) whereas the bandwidth measurements were taken using the iPERF tool [6] over 10 minutes of measurement (600 runs with a window size of 45 Kbytes). The Wi-Fi results obtained are presented in Table 5.

In Phase 2 of the experiment, the connection is established over the commercial 4G network (EE) using two RUTX11 4G modems. One 4G modem is connected to the production line side, while the other 4G modem is connected to the edge device, where they communicate over the commercial 4G network. These modems are bridged together using Open VPN to insure both sides are connected to the same local network.

Again, RTT and bandwidth measurements were conducted between the laptops present both at the factory and Edge sides. RTT measurements were carried using command ping from the command line console over 15 minutes (app. 900 pings) whereas the bandwidth measurements were taken using the iPERF tool [6] over 10 minutes of measurement (600 runs with a window size of 45 Kbytes). The 4G results obtained are reported in Table 5.

TABLE 5 EXPERIMENT A AND B MEASUREMENTS: RTT AND BANDWIDTH BETWEEN THE FOG DEVICE AND THE EDGE DEVICE.

Wireless Technology	RTT (ms) (Averaged over 15 minutes)		Video Bandwidth (Mbps) (Averaged over 10 mins)	
	Avg	Std	Avg	Std
Wi-Fi	5.9	3.7318	360.39	69.2629
4G	82.01	38.2224	10.1	1.9936

Note here that we ran this experiment indoors where the 4G modems experienced a weak signal reception (approx. -100 dBm RSRP), which affected the achievable throughput and latency of the 4G modems. Should the experiment have been trialled outdoors, then the RTT and the bandwidth of the 4G modems would have been significantly improved and on par with the Wi-Fi measurements.

Although Wi-Fi measurements in Phase 1 have shown an improved capability compared to 4G, it can only support short-range wireless communications with a moderate number of connected devices. Hence, Wi-Fi should be exclusively adopted for the ZDM use case where a point-to-point connection is available between the edge node and the production line with a small number of devices, especially those with high bandwidth requirements (i.e., high resolution cameras).

On another note, the weak signal reception experienced in Phase 2 has affected the Edge-to-Factory connectivity by introducing a high latency and a reduced bandwidth. Having a better received signal strength would inarguably improve the achievable latency and bandwidth level. However, 4G cannot support high number of devices, especially those with high bandwidth requirements (i.e., high

resolution cameras) and cannot meet the infinitesimally low latency requirements set for the ZDM use case.

In the next phase (Phase 3), the above experiment setup will be trialled again but with 5G SA connectivity. In this phase, the ZDM use case can benefit from the 5G capabilities in the terms of supporting a massive number of bandwidth-hungry devices, while simultaneously providing a very low latency. This is our expectation that with 5G, the throughput and latency will be significantly improved compared to 4G and Wi-Fi so as to benefit the ZDM use case by enabling a higher camera resolution (higher bandwidth) in the uplink and much faster response time (lower latency) from the defect detection application in the downlink.

3.2.4.2. Experiment B - telemetry solution

The measurements provided in this section represent the latency of transporting the telemetry data from the Edge side towards the Cloud side. These measurements are relevant to future object detection applications and intelligent engines, which rely on the telemetry data collected and can be time-sensitive.

Communication to the AWS services endpoint and IoT core can be done with two communication protocols: MQTT or HTTP. AWS IoT Greengrass Core uses MQTT protocol with QoS-1[1] for publishing data to the cloud. All measurements were done with the Edge platform connected to a commercial 4G network, and the IoT core deployed in eu-west-2 region (London). Messages are of fixed size and simulate sensor data on a device, subsequently they are sent to the Greengrass core with the frequency of 5s. As shown in Figure 11, the X-axis on the graph shows the published messages on the IoT core and the Y-axis the measured latency for each message sent. This measurement was done by sending successively 1600 MQTT messages, at the rate of one MQTT message per 5 secs. The measurements show an average message latency of 156ms with a standard deviation of 149ms, when using MQTT. Note here that these results present a one-way latency measurement, and the clock of the edge device is synchronized with the cloud clock from AWS.

[1] QoS-1 is one of three MQTT modes (QoS-0, Qos-1 and QoS-2), where data is sent at least once.

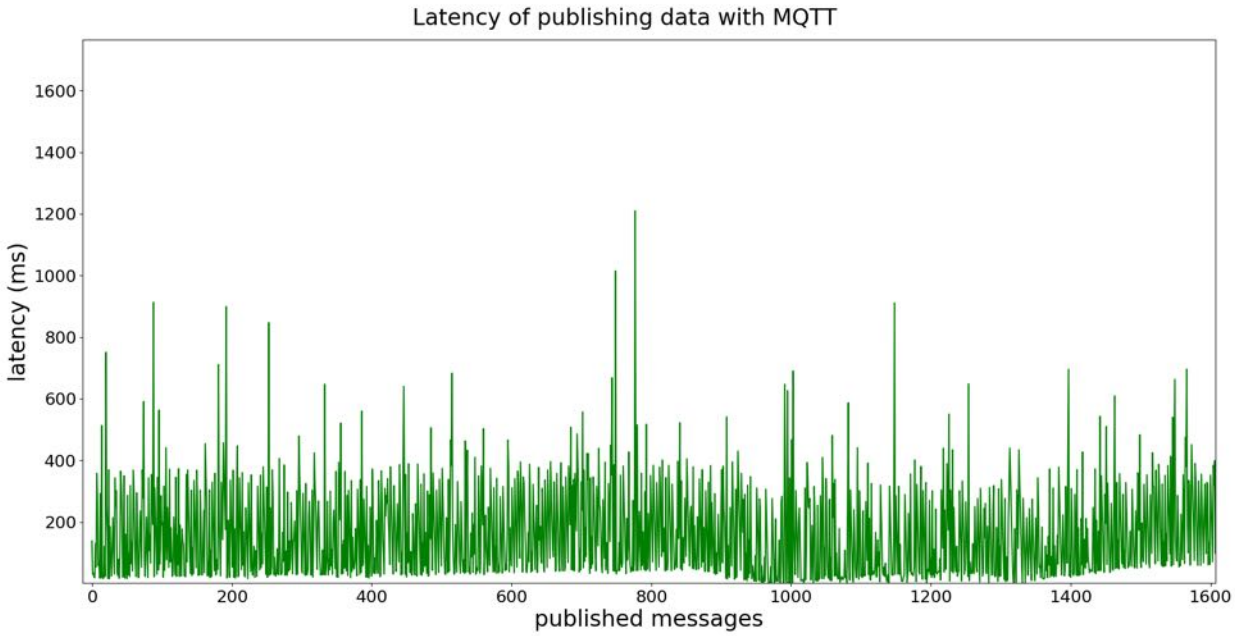

FIGURE 11 ACHIEVED ONE-WAY LATENCY OF 1600 MQTT MESSAGES.

Figure 12 shows the measurement results using HTTP instead. The latency is larger with peaks over 6s, a sign that possibly the protocol does not scale well compared to MQTT when sending multiple messages. The measurement was done by sending successively 1600 HTTP messages, at the rate of one HTTP message per 5 secs. The measurements show an average message latency of 565ms with a standard deviation of 415ms. It can be seen from Figure 11 and Figure 12 that the MQTT protocol achieves a lower latency than HTTP, and hence MQTT is used as the main data publishing protocol in this experiment. Again, these results present a one-way latency measurement, and the clock of the edge device is synchronized with the cloud clock from AWS.

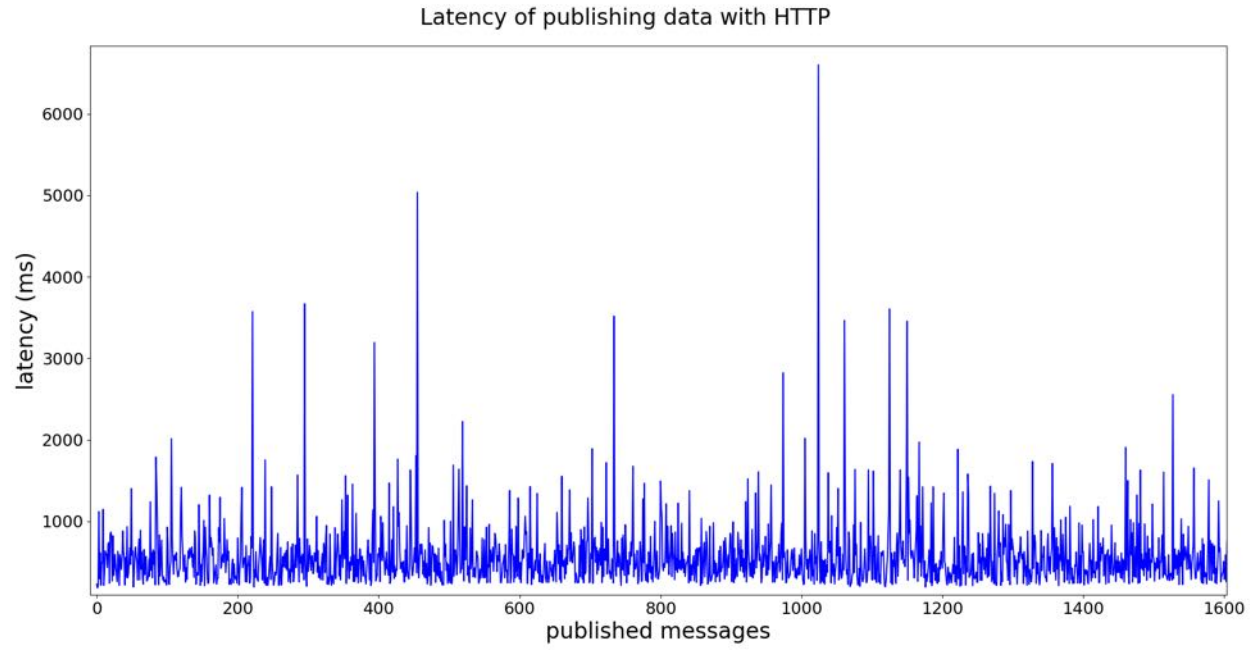

FIGURE 12 ACHIEVED ONE-WAY LATENCY OF 1600 HTTP MESSAGES.

3.2.5. Next steps

The next steps for the ZDM use case will include the replication of the experiment but now with a standalone (SA) 5G network replacing the current 4G connectivity option. The integration of the ZDM use case with the other elements of the DEEP platform, namely the DASS, the BASS and the IESS is also planned.

The introduction of a RAN control app is also in the plans, where telemetry data will be used to train an Intelligent Engine (IE) that will perform traffic splitting, steering and switching at the factory side. The objective of this RAN control app is to efficiently use available access technologies (e.g., 5G and Wi-Fi) and steer access traffic to provide slice/service requirements such as reliability or high throughput for multiple users.

3.3. I4.0-UC3: massive MTC

In this section, we present the test results of the developed mMTC testbeds for performance validation at this stage. The focus is on cloud native design of softwarized IoT stacks facilitating resource saving and pooling. Section 3.3.1 presents the test results of a higher-layer resource pooling design with one higher-layer (i.e., L2 and above) network function serving multiple RAN (i.e., PHY) functions of IEEE 802.15.4. The results show that the CPU usage is significantly reduced while a similar performance is achieved. Section 3.3.2 focuses on the lower-layer design with one PHY-layer network functions of LoRa serving multiple cells. The test results show that a significant resource pooling gain can be achieved, up to 55 full-traffic cells can be served by one instance of PHY-layer network function using a mini-PC based setup. Although the latency is increased due to resource pooling, it is still acceptable for mMTC IoT applications.

3.3.1. End-to-end testbed of IEEE 802.15.4

We show the feasibility of cloud-native design for IoT communication stacks by sharing a network function across multiple RAN functions (PHY layer). The RAN function is usually more resource intensive than the network function in virtualized IoT applications. This opens the opportunity of a network function being shared across multiple RAN functions, thereby reducing resource overprovisioning.

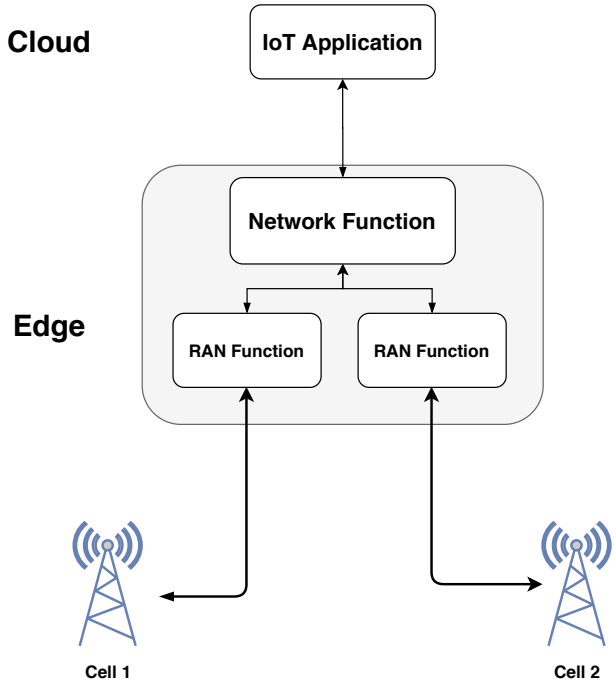

FIGURE 13 ILLUSTRATION OF CLOUD-NATIVE DESIGN OF 802.15.4

Figure 13 shows our cloud-native design for resource-pooling of network function. We have two RAN function serving two cells, which shares the same network function. However, IoT network functions are not designed to be multiplexed across multiple RAN functions. We make two key changes in the implementation of the network function to enable multiplexing of the network functions.

Asynchronous MAC:

Medium Access Control manages access to the radio channel for each packet transactions between two nodes. However, we want to manage multiple radio channels using a multiplexed approach. This creates problem, particularly in coupled communication, such as acknowledgement (ACK) of a transmitted packet. The MAC waits for the ACK resulting in blocking the flow for other RAN functions as shown in Figure 14. We design an asynchronous MAC to solve this. The MAC transmit and receive chains are decoupled with a shared buffer for maintaining the state of ongoing packet transmissions. The buffer uses call-backs to update the state of the ongoing transmissions. In case of failure of the ongoing transmissions, timer events manage the retransmission mechanisms. These asynchronous methods allow us to decouple the coupled transactions and use the waiting time to process (transmit and receive) packets on other RAN functions as shown in Figure 14.

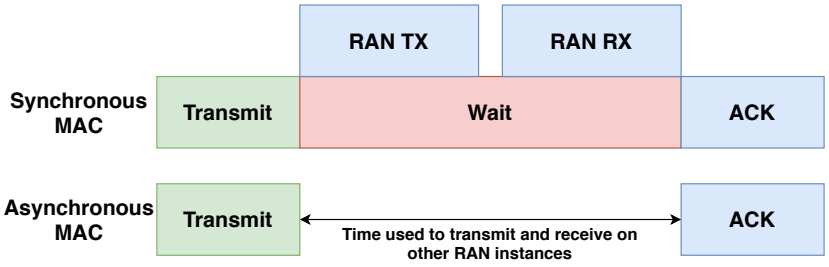

FIGURE 14 ILLUSTRATION OF ASYNCHORNOUS MAC DESIGN

RAN interface resolution:

To communicate to a specific IoT Node, the network function needs to resolve the correct RAN function to reach the node. IoT network functions are usually designed to work with a single RAN interface and lacks methods for RAN interface resolution. To address this issue, we introduce RAN interface resolution from link layer address. During network association events, the network function tracks the RAN interface used for a particular link layer address. The RAN interface is added as an additional routing parameter in the routing table. During subsequent unicast downlink traffic, the RAN interface is resolved from the link address in the routing table. The packets are then routed to the appropriate RAN instance as shown in Figure 15.

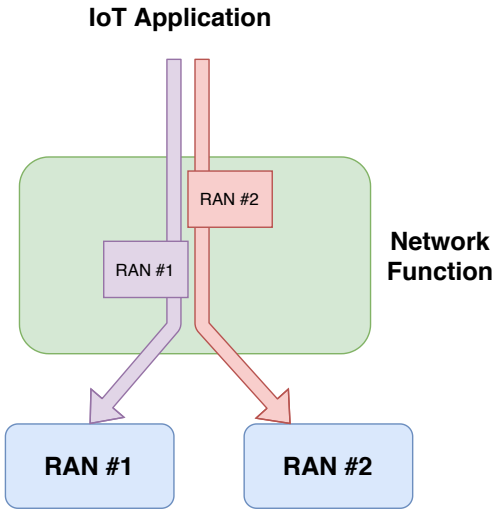

FIGURE 15 ILLUSTRATION OF RAN INTERFACE RESOLUTION

3.3.1.1. Experimental setup

We perform a feasibility study of the cloud-native approach using IEEE 802.15.4 [7] as our RAN function (PHY) and using Contiki-NG [8] as our network function. The experimental setup is shown in Figure 16.

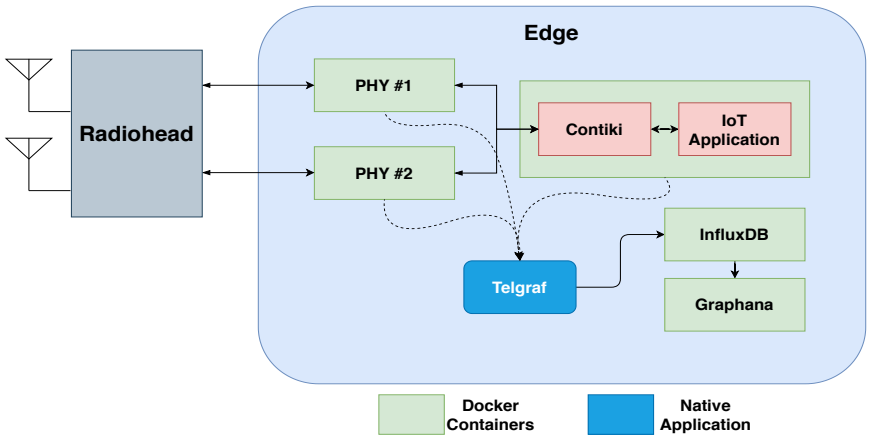

FIGURE 16 EXPERIMENTAL SETUP

We have two computing subsystems: Radiohead and Edge. The Radiohead is responsible for radio communication with the IoT nodes. It uses an USRP B210 [9] as the radio front-end. The receiver

processing chain channelizes the radio sample stream into IEEE 802.15.4 channels. The channelized sample stream is then cross correlated to IEEE 802.15.4 preamble, to reduce the data traffic to the edge. The processed radio sample stream is then forwarded to the edge using ZMQ-TCP [10] over an ethernet connection. The transmitter processing chain, receives radio samples from the edge using ZMQ-TCP over ethernet, combines the multiple radio sample stream and forwards it to the USRP B210.

Figure 16 shows the experimental setup, the Edge runs the RAN (PHY#1 and PHY #2) and network functions (Contiki). In our setup, we use two RAN function instances and a single network function instance. The RAN function implements the baseband processing of IEEE 802.15.4 for a single radio channel. The network function uses ZMQ-TCP connection to connect to the RAN function. We use ZMQ pub-sub message-pattern for the transmitter and ZMQ push-pull message-pattern for the receiver. The network function is responsible for management of communication to the IoT nodes for IoT applications. We use Leshan LWM2M demo server as our IoT application, with Zolertia firefly [11] nodes being used for IoT nodes.

3.3.1.2. Experimental results

We connect 8 Zolertia firefly nodes over two radio channels to our IoT application using our experimental setup. To evaluate the feasibility of our approach we compare the round-trip latency of communication to IoT nodes while measuring the computing resources required by the RAN and network function to using dedicated network stack functions for each radio channel.

Figure 17 shows the CDF of end-to-end ping RTT between the edge and the IoT nodes, using the cloud-native or resource pooling approach and using dedicated network stack for each radio channel. We use two cases to evaluate our result:

Case 1: We connect all the nodes to the IoT application over the two channels and ping a single IoT node every second. This case compares the latency of the resource pooling approach to using dedicated network stack for each channel. We observe comparable RTT as well as Packet Error Rate (PER) to using dedicated network stack.

Case 2: To evaluate the impact of resource-pooling while balancing moderate data traffic from multiple RAN functions simultaneously, we setup a background ping process to another IoT node through the second RAN function. We observe comparable RTT to the single channel case, with PER increasing by 1%.

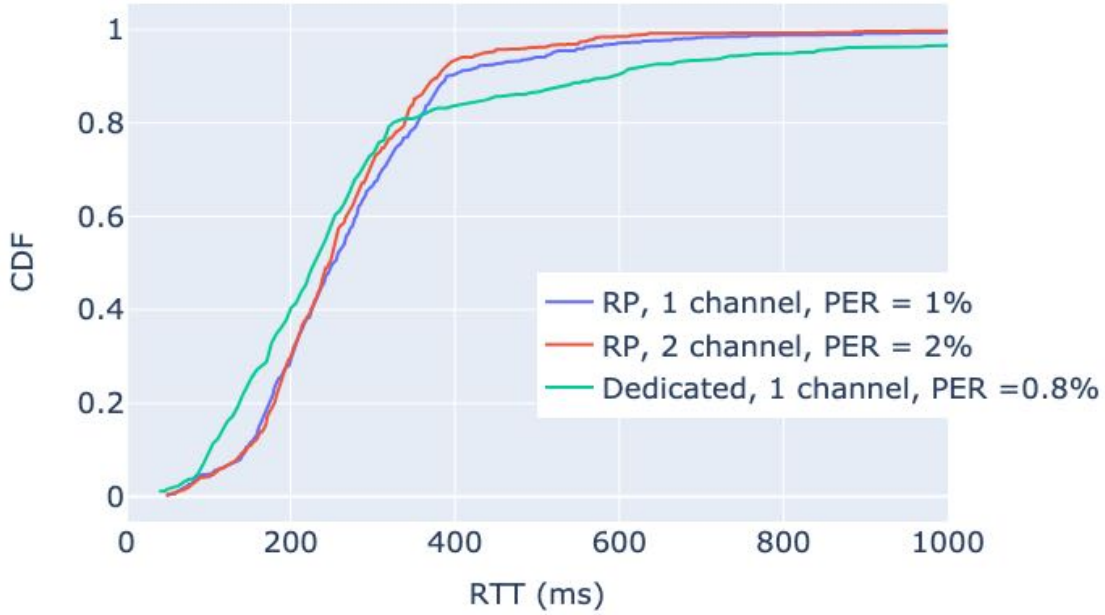

FIGURE 17 EXPERIMENTAL RESULTS OF ROUND-TRIP-TIME (500 PACKETS)

Figure 18 compares the computational resource usage for Case 2 in resource-pooling and dedicated stack case. We use docker stats monitoring to monitor the CPU usage every second. We observe that the resource-pooling case consumes significantly less compute resources in comparison to the dedicated stack case. The improvements can mostly be attributed to the removal of busy wait periods in the asynchronous MAC. The resource-pooling approach also removes duplication of network management overhead present when managing each channel separately using dedicated stack.

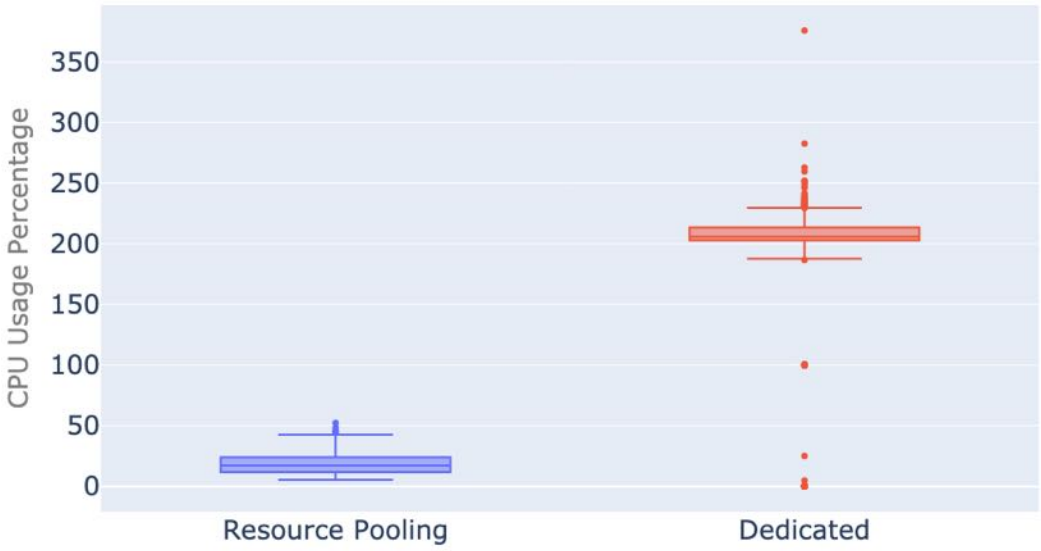

FIGURE 18 EXPERIMENTAL RESULTS OF CPU USAGE

3.3.2. Emulation testbed of LoRa for larger scale

In this Section, we present the initial validation results for our large-scale emulation testbed. We developed an emulation testbed [12] and implemented LoRa as an example to showcase the concept

and design. The testbed is validated by comparing the one-cell emulation results with real LoRa IoT devices. In this work, we further extend the testbed by virtualizing LoRa decoding function into an Edge container with a CRAN architecture. In Section 3.3.2.1, we present the overview of experimental setup and elaborate key components of the testbed. In Section 3.3.2.2, we show the measurement results as well as resource pooling gain and limits provided by the testbed with cloud native design.

3.3.2.1. Experimental setup

The experimental setup of the emulation testbed is shown in Figure 19. The emulation testbed contains three main parts, i.e., radio head emulation, edge function and monitoring function. The radio head and Edge functions are virtualized in two Docker containers [13] running on top of a Linux based Ubuntu system. Meanwhile, software for monitoring system metrics is running locally in Ubuntu. Detailed description of each part is elaborated as below:

- Radio head emulation: emulate LoRa packet I/Q samples

  Packet source functions are running in docker containers to emulate cell-level LoRa packet I/Q samples. Data traffic is controlled by modifying the cell number in Radio head container.

- Edge: running real LoRa receiver software

  An open-source LoRa receiver [14] is virtualized and running in the Edge container.

- Monitoring: visualize system metrics
  - Telegraf [15]: a plugin-driven server agent for collecting and reporting metrics for data from databases, systems, and IoT devices.
  - InfluxDB [16]: an open-source time series database, real-time visibility.
  - Grafana [17]: a multi-platform open-source analytics and visualization web application.

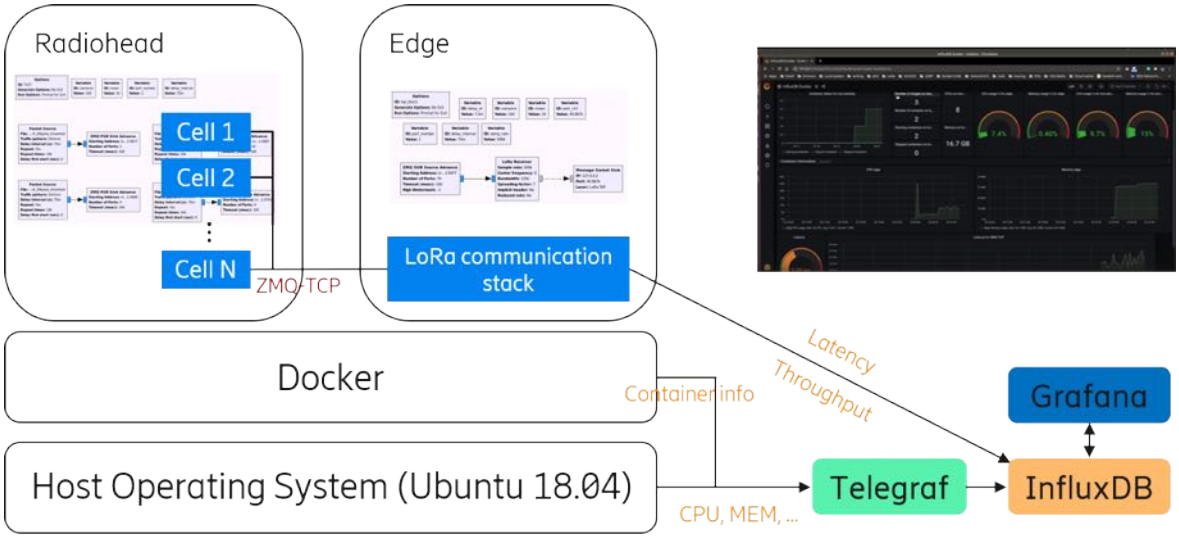

FIGURE 19 EMULATION TESTBED SETUP

During run-time, the radio head will publish LoRa I/Q samples to ZeroMQ (ZMQ) [10] pub sockets. The edge will then subscribe to the ZMQ sockets to retrieve the I/Q data and decode the message.

Meanwhile, Telegraf collects system resource data information such as CPU & memory resource utilization, and container running status every 0.1 second. Then Telegraf reports the data to the time series database – InfluxDB.

In the case of latency monitoring, we generate timestamp messages at the radio head and send the message synchronously with LoRa packets. On the edge side, we record the time of retrieving LoRa packets from ZMQ sockets and calculate the time difference as the buffering latency. Also, with the knowledge of retrieved data, we can easily calculate the throughput of the system. Similarly, we report the latency and throughput data to InfluxDB and then visualize the data together with other system information with a Grafana dashboard. An example of the Grafana dashboard is illustrated in Figure 20.

FIGURE 20 AN EXAMPLE OF GRAFANA DASHBOARD

In the first row of the dashboard, we have several panels showing the system information e.g., container information (number of container images, number of currently running containers, etc.), CPU core number, system total memory and CPU & memory utilization. The second row contains two panels illustrating edge container resource utilization i.e., edge container CPU and memory utilization. The last two rows depict the packet buffering latency and system throughput obtained from edge container.

3.3.2.2. Experimental results

The purpose of this measurement is to evaluate the resource pooling gain obtained with a cloud native architecture for IoT use case. In this measurement, we considered four different traffic loads and compared system metrics including CPU utilization, memory utilization, throughput etc. for each case. Detailed description of each case is listed as below:

- Case 1: one-cell baseline case, 127 LoRa packets are transmitted to Edge container per 10 seconds. In this case, there is only one cell in the Radio head container, indicating no resource pooling gain on Edge side.
- Case 2: 30 cells, 30×127 LoRa packets are transmitted to Edge container per 10 seconds.

- Case 3: 55 cells, 55×127 LoRa packets are transmitted to Edge container per 10 seconds, maximum supported cells achieved.
- Case 4: 58 cells, system overloaded, 58×127 LoRa packets are transmitted to Edge container per 10 seconds.

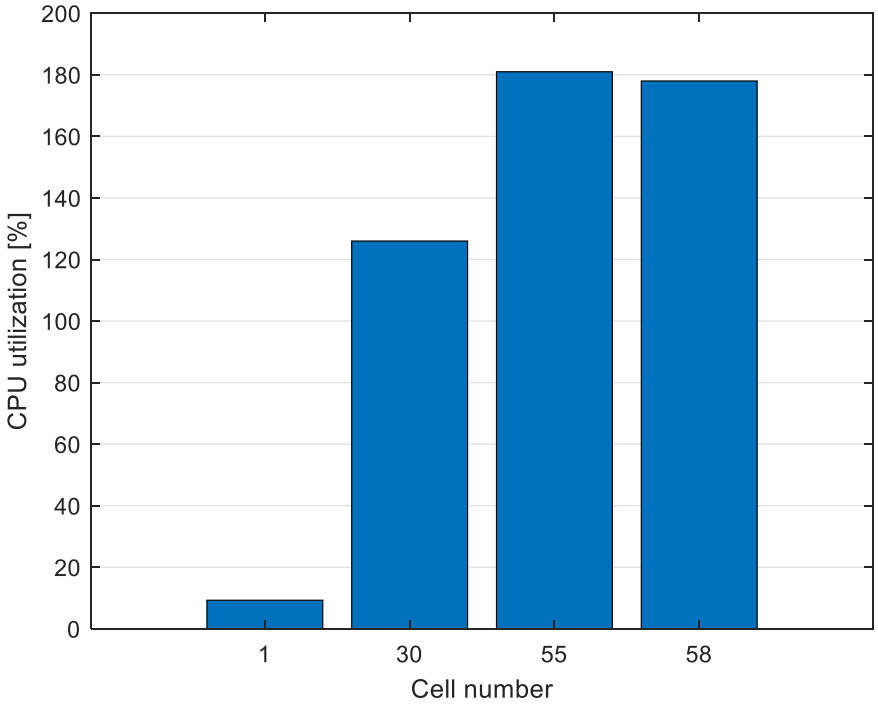

FIGURE 21 CPU USAGE FOR DIFFERENT CASES

Figure 21 shows the comparison of average CPU usage of 400 groups of data for four different cases. With the number of cells equal to 1, 30 and 55, the CPU usage continues to increase as traffic load increases. The reason is that the powerful CPUs used in Edge server can process very fast. Therefore, the receiving time between the adjacent packets, is much longer than the processing time, and the idle time between two packets processing can be allocated to process the packets from other cells. When the number of cells further increases, no more CPU resources can be allocated since maximum CPU utilization is reached. The results show the resource from one CPU core can be pooled to serve many cells with full traffic. In practice, the cells will not be fully loaded simultaneously. It means that the CPU resources can be shared with even more cells, which can make the resource utilization significantly more efficient comparing to prebooking the resources per cell for its peak load.

Figure 22 compares the additional latency of LoRa packets due to buffering behaviour for different cases. Each group of data consists of 323 measured data points obtained from the edge. We consider three cases with the number of cells equal to 1, 30 and 55. It is shown that one-cell case has the smallest mean latency value and variance value, while 55-cell case has the largest mean latency value and variance value. The results confirm that pooling more cells increases the packet delay. For 58-cell case, since the CPU fails to handle such heavy traffic, the incoming packets are continuously buffered in the queue and the latency keeps increasing in time until system memory runs out.

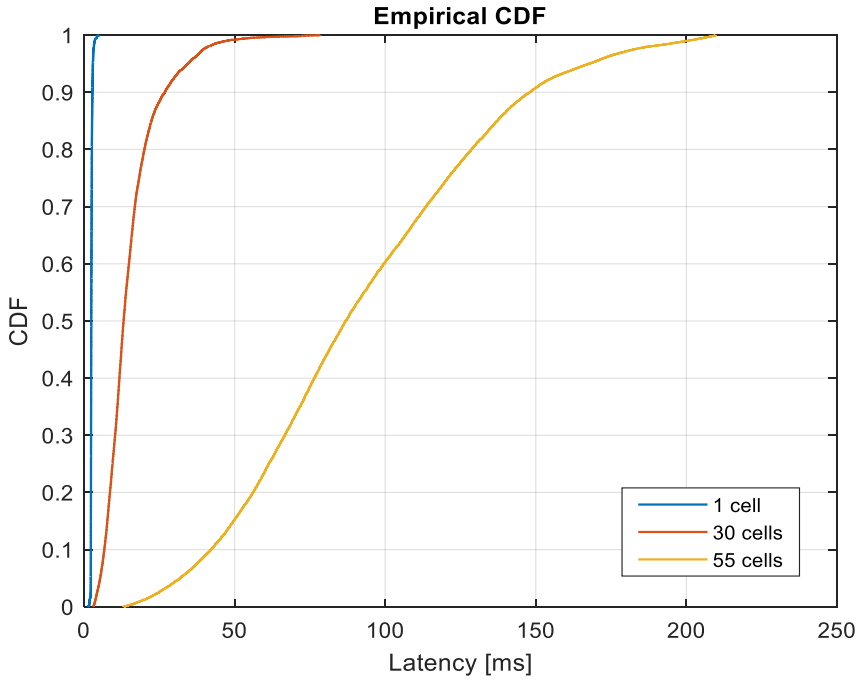

FIGURE 22 BUFFER LATENCY FOR DIFFERENT CASES

Table 3 summarizes the statistics of 300 groups of measured throughput data for four test cases. We can conclude that system throughput increases with the increase of the number of cells in the radio head, i.e., increasing traffic load, when system is not overloaded. The throughput will not further increase when the system is saturated as CPU cannot process more data.

TABLE 3 SYSTEM THROUGHPUT FOR DIFFERENT CASES

Number of cells	**Average throughput(mean, variance) [MB/s]**
1	2.09, 0.28
30	52.78, 2.87
55	89.36, 3.43
58	90.04, 4.04

3.3.3. Next steps

The current results show a good potential for Cloud native design of IoT stacks in increasing resource utilization efficiency. The testbeds will be further developed to add orchestration features using K8s for automation and auto scaling. An intelligent application of PHY security enhancement based on RF fingerprinting will be further developed and integrated into the tested. In addition to IoT devices, we plan to further investigate security enhancement using RF fingerprinting techniques in cellular systems, such as 5G. Moreover, the relevant DEEP components of BASS, IESS and DASS are going to be developed and integrated with other I4.0 use cases in one common DEEP platform.

3.4. Initial KPI validation of I4.0 use cases

The following Table 4, Table 5 and

Table 6 summarize the initial KPI validation against the technical requirements for I4.0 use cases defined in 5G-DIVE Deliverable 1.1 [2] based on the experimental results presented in Sections 3.1, 3.2 and 3.3 above.

As we can see from Table 5 for DT use case, end-to-end latency, the downlink bandwidth (capacity) available for the remote control of the robot and the uplink bandwidth (capacity) available for video for future needs of the intelligence engines were tested and found to be satisfying the requirements.

TABLE 4 DIGITAL TWIN USE CASE KPI EVALUATION BASED ON THE EXPERIMENT RESULTS

Technical Requirements (WP1)	Description	Value	**5G (NSA)**
TR-I4.0-UC1-01	Reference Availability	99.9999%	Not tested
TR-I4.0-UC1-02	Reference Reliability	99.999%	Not tested
TR-I4.0-UC1-03	Reference E2E Latency	20ms	✓
TR-I4.0-UC1-04	Remote control bandwidth per instance (DL)	100Kbps	✓
TR-I4.0-UC1-05	Video bandwidth per instance (UL)	5Mbps	✓
TR-I4.0-UC1-06	Reference Connection density	<1000 device per Km2	Not tested
TR-I4.0-UC1-07	Reference Coverage	Local using one cell, no handover	✓

TABLE 5 ZDM USE CASE KPI EVALUATION BASED ON THE EXPERIMENT RESULTS

Technical Requirements (WP1)	Description	Value	4G
TR-I4.0-UC2-01	Availability	99.9999%	Not tested
TR-I4.0-UC2-02	Expected Reliability	99.9999%	Not tested
TR-I4.0-UC2-03	Reference E2E Latency	10ms	✗
TR-I4.0-UC2-04	Remote control bandwidth per instance (DL)	100Kbps	✓
TR-I4.0-UC2-05	Video bandwidth per instance (UL)	5Mbps (per camera)	✓

As shown in

Table 6 for mMTC use case, most of requirements can be fulfilled according to the obtained test results with the resource pooling feature developed.

TABLE 6 MMTC USE CASE KPI EVALUATION BASED ON THE EXPERIMENT RESULTS

Technical Requirements (WP1)	Description	Value	Evaluation
TR-I4.0-UC3-01	Application layer end-to-end latency	<10 s for cellular IoT (5G requirement) [18], < a few 100s ms for non-cellular IoT.	Experiment results shows these can be managed with resource pooling.
TR-I4.0-UC3-02	Air interface bit rate	A few 10s bps – a few100s kbps for cellular IoT [18], a few 100s kbps to 2 Mbps for non-cellular IoT.	Experiment results shows the air interface performance is not affected by doing resource pooling.
TR-I4.0-UC3-03	Connection density	Up to 1 million devices/km2 for cellular IoT (5G requirement).	Experiments show the scalability can be improved with more efficient resource usage which help achieve this KPI efficiently.
TR-I4.0-UC3-04	Coverage	164 dB maximum coupling loss at a rate of 160 bps for cellular IoT (5G requirement), 10-100 meters for non-cellular IoT.	Experiment results shows the air interface performance is not affected by doing resource pooling. So it indicates this is achievable.
TR-I4.0-UC3-05	Availability	99.99%	Not tested at this time.
TR-I4.0-UC3-06	Link reliability	High (exact value TBD, massive MTC requires lower reliability than eMBB and URLLC)	Not tested at this time.

4. Initial validation results of ADS use cases

In this section, we presented the initial validation results for ADS use cases. First, end-to-end experimental setup including its different components is reported in Section 4.1. Then, the mission scenario and flow are presented in Section 4.2. Finally, the end-to-end results followed by the KPI validation are presented in Section 4.3.

4.1. End-to-End Experimental setup

The end-to-end experimental setup is shown in Figure 23, depicting the different components used in the validation of the ADS use cases.

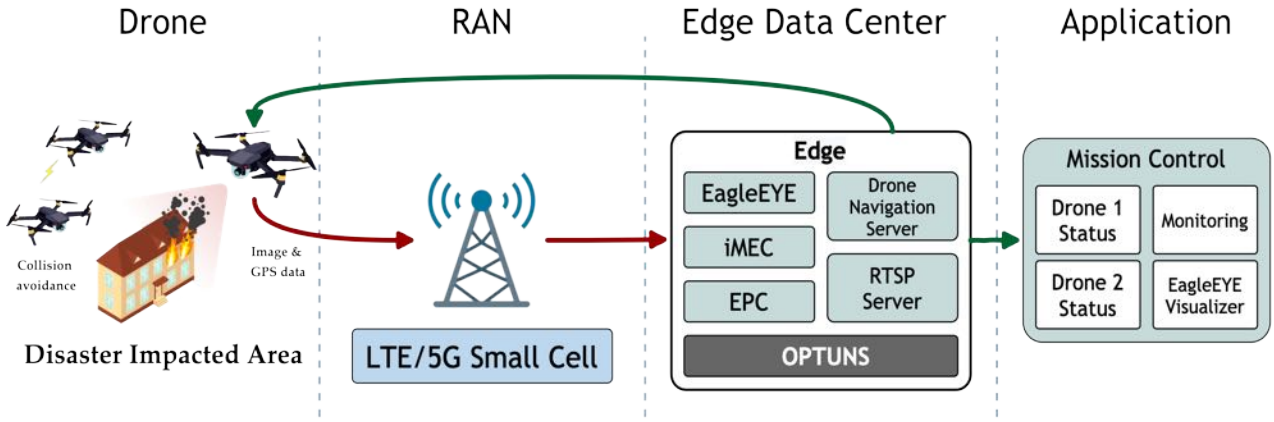

FIGURE 23 ADS EXPERIMENTAL SETUP

4.1.1. Drone

The Drone used in the field trials (see Figure 24) has the upgraded some of initial specifications provided in [1] to provide as better services for ADS use cases. The upgraded specifications include the total weight (excluding battery) is About 7kg, maximum effective take-off weight is 15kg, battery 16000ah 6s25c, maximum bearable wind: Level 6. Also, the drone is equipped with a 4G/5G CPE to be linked with 5G NSA.

FIGURE 24 DRONE IN ADS USE CASES

The Drone is also equipped with enhanced components (see Figure 25):

- Jetson Nano: Small computer with low power consumption and high performance. Initially Raspberry Pi created unacceptable latency during the stream of video. Hence, we added Jetson Nano to decrease the processing latency needed to encode the video from the camera by half of the original time period.
- Four core ARM® Cortex®-A57 MPCore processor
- NVIDIA Maxwell™ The architecture is equipped with 128 NVIDIA CUDA® Cores.
- 4 GB memory. It is responsible for connecting the UAV with the control system, and receiving the image output from the camera and converting it into the RTSP stream push.
- Video camera: Using Logitech c930e webcam, the highest video quality is full HD 1080p, Support H264 video compression, and can provide 4x zoom
- PTZ: Using Tarot 3D PTZ, the rotation direction (Pan): + - 330 degrees, pitch direction (tilt): + - 200 degrees, roll direction (roll): + - 48 degrees
- Battery: Using high discharge lithium polymer battery (16000mAh 25C Li-Po)

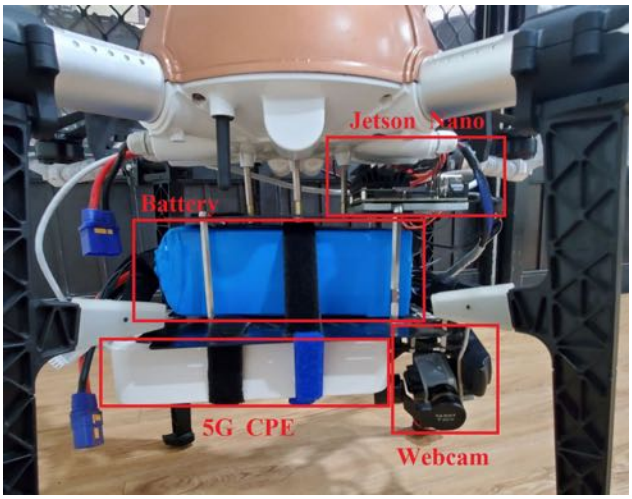

FIGURE 25 DRONE COMPONENTS

In ADS, the developed system completes all Drone flight plans and control on the webpage [19]. In particular, after the flight plan member/designer log in on the homepage, he/she will access the Drone control interface as shown in Figure 26. This control interface (using the webpage [19]) first selects the Drone equipment to be controlled, and the information panel contains the Drone connection status, real-time image stream picture, Drone battery voltage, security lock status, GPS signal bit number, current flight altitude, GPS coordinates and azimuth, local weather and wind speed information as shown in Figure 27.

FIGURE 26 5G-DIVE DRONE CONTROL SYSTEM

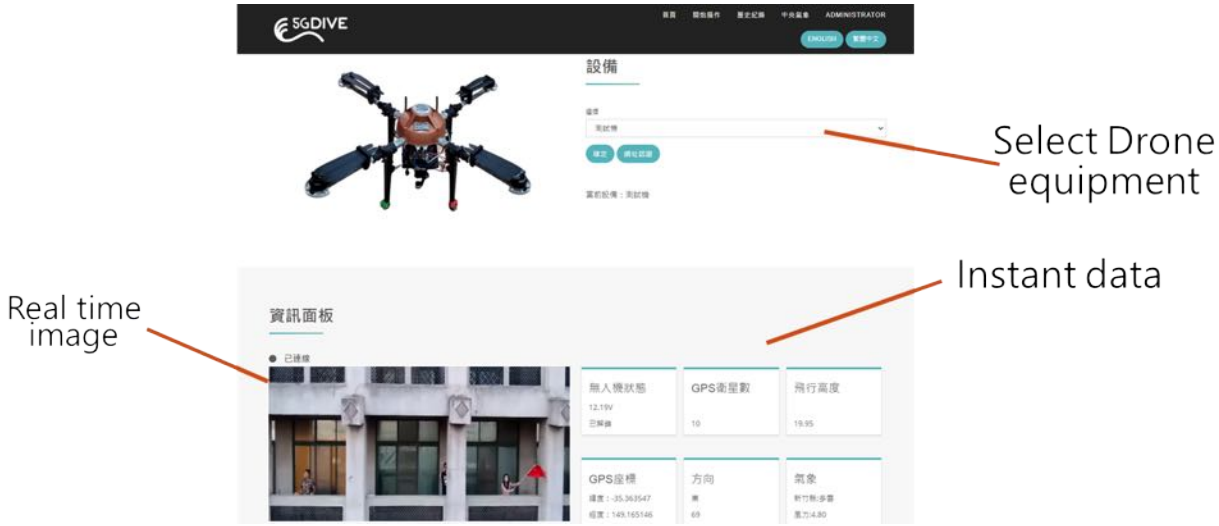

FIGURE 27 DRONE CONTROL INTERFACE

4.1.2. Connectivity

For the connectivity, we have the 4G/5G CPE (CPE-RTL0200) on-board of the drone. The CPE is connected to the small cell that is installed on the 6th-floor balcony of the MIRC building. The setup steps for the small cell and CPE can be found in Appendix 8.1 and Appendix 8.2. Figure of the small cell and the CPE can be seen in Figure 28 and Figure 29. The small cell is first connected to a PoE L2 switch and then directly connected to the Edge Data Centre in the MIRC building via a 1 GB SFP optical fibre link. A more detailed discussion on Edge Data Centre and the connectivity mapping will be discussed in Section 4.1.3.

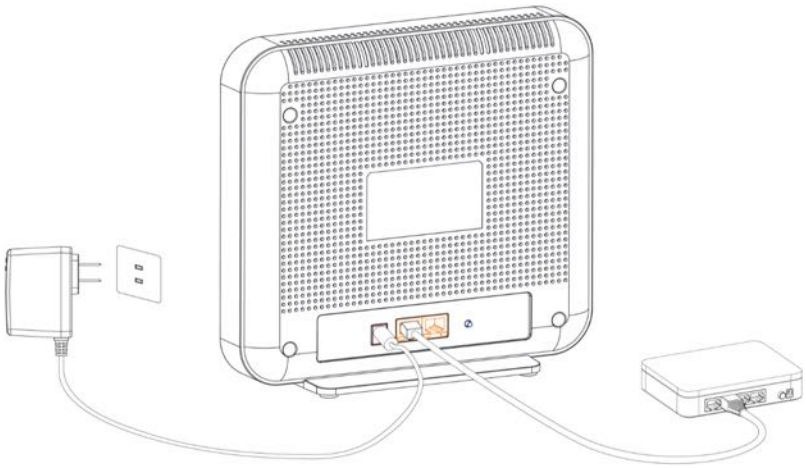

FIGURE 28 SMALL CELL

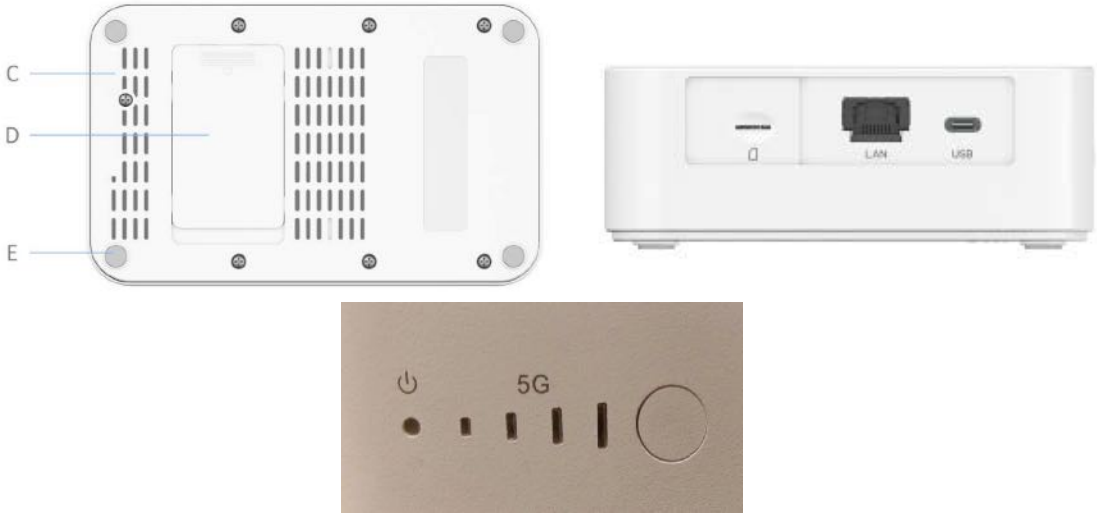

FIGURE 29 5G CPE

In Edge Data Centre, there are two main components: iMEC for 4G/5G users' application data, and NSA EPC for user authentication. Together, they provide a fully functional 3GPP 4G/5G mobile network core. iMEC is a powerful computing platform located on the border between the radio access network and core network. It consists of not only the serving gateway (SGW) that interacts with other 3GPP components by standard signalling, but also the routing module that intercepts and redirects packets to the localized applications. iMEC also provides the virtualized platform and management gadgets for Container-based and VM-based applications.

As for the core network, the NSA EPC is dealing with the UE attachment, authentication, mobility management, and bearer control. Inside NSA EPC is the MME module, handling the control plane. It handles the signalling related to mobility and security for UE access and it is also responsible for the tracking and the paging of UE in idle mode. In this trial, the MME sends the session control signal to iMEC for the establishment of default bearer for UE.

4.1.3. Edge data centre

The Edge Data Centre is located in the basement of the MIRC building in NCTU. In the Edge Data Centre, we dedicate 2 racks of server for the use of 5G-DIVE project. The traffic from the small cell enters the Edge Data Centre through OPTUNS [20]. Details on the complete connectivity mapping is shown in Figure 30 below. The traffic flow is as follows. (i) Traffic from the small cell enter OPTUNS, and then (ii) OPTUNS distribute the traffic to the respective applications.

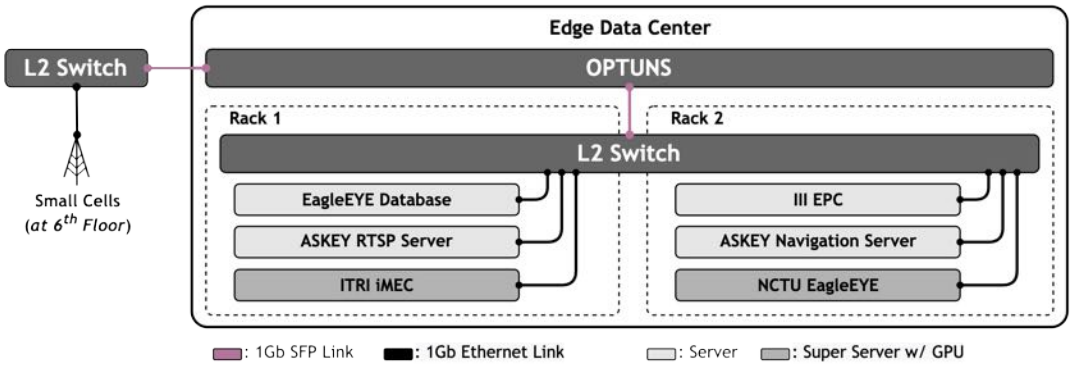

FIGURE 30 EDGE DATA CENTER CONNECTIVITY SETUP

4.1.4. EagleEYE

EagleEYE stands for Aerial Edge-enabled Disaster Relief Response System. It is an intelligent drone-based emergency response solution deployed at the edge to process aerial drone video data and provide real-time Person in need of a help (PiH) GPS information. EagleEYE components are packaged inside container micro-services and are orchestrated and deployed using Kubernetes via ITRI's iMEC. EagleEYE micro-services architecture can be seen in Figure 31 below.

The details for five of the EagleEYE micro-services are as follows:

1. Database services. This service facilitates the storage of telemetry data and other important data like drone status and worker application status. The remaining micro-services use publish/subscribe mechanisms to push or get information from the database service. For example, a scheduler service publishes image information, and each detection service consumes that information to perform an object detection algorithm. In EagleEYE, Mongo Database [21] is utilized to store telemetry data, drone status and worker application status. While Redis Database [22] is utilized to temporarily store image information and to perform publish-subscribe mechanism for inter-container communications.
2. Web service. This service is mainly used to facilitate the communication between containers and database service through RESTful APIs. This APIs are accessible for all of the deployed EagleEYE micro services. For example, this service is responsible to communicate with the Drone Navigation Server in collecting GPS information.
3. Scheduler service. This service manages the collected frames sent by the drone(s) and controls the task allocation for the Detection service. The tasks are frames that are delegated into the available Detection Service in a Round-Robin fashion. At the same time, it also sends the original frames into the Visualizer Service for visualization.

4. Detection service. This service consumes frames and detects the availability of PiH objects in a particular frame. Along the detection process, it validates and verifies whether the detected PiH objects are valid or not based on a batch of frames. Once verified, it sends bounding box information of the detected PiH objects into Visualizer Service.
5. Visualizer service. This service finalizes the information before being visualized to the end-user. It captures original frames, while at the same time, it loads GPS information in this particular time and collects bounding box information from Scheduler Service. Finally, all relevant information is presented displayed properly for visualization.

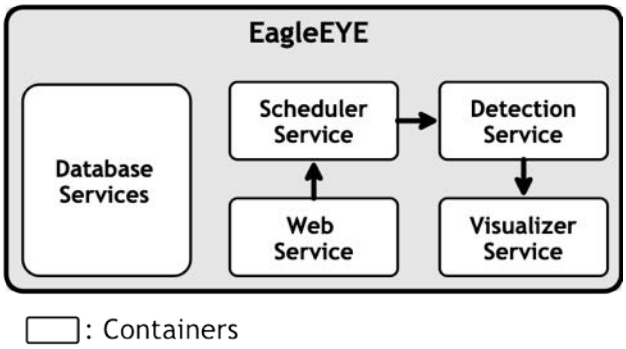

FIGURE 31 EAGLEEYE MICRO-SERVICES ARCHITECTURE

4.1.5. RTSP server

As for the RTSP server, we are using an open-source implementation from [23]. The RTSP server is used to receive video feed from the drone, as well as visualizing the outputs of EagleEYE PiH detection. The drone first sends the video feed to the RTSP server installed on the edge. EagleEYE then capture and extract video feed from the RTSP server into images, compress each image and send them into EagleEYE scheduler service. Once processed, the outputs the PiH detection results are sent back to the RTSP server to be visualized. The RTSP protocol is used to transport a H264 video codec from the drone camera to the Edge Data Centre via a TCP connection.

4.2. Mission scenario and flow

In the ADS mission scenarios, we used the drone control interface (i.e., drone navigation server) to execute the target missions of ADS. This interface is divided into a single destination flight and a multi-point path flight. The planner/designer can input the coordinate height in the map to start the flight plan and fly to the destination coordinate point as shown in Figure 32.

FIGURE 32 DRONE SINGLE PANEL INTERFACE

Moreover, the Control interface multi-point path (i.e., multiple GPS coordinates) flight is shown in Figure 33. The planner can enter the coordinate height in the map to plan the path. The coordinate points will be added to the list in order. After the path planning is completed, the command will be sent out to drone, and the drone will fly in order according to the coordinate points in the list. After the drone scans the emergency (PiH using EagleEYE), the GPS coordinate points are automatically recorded in the list of emergency PiH for rescue tasks.

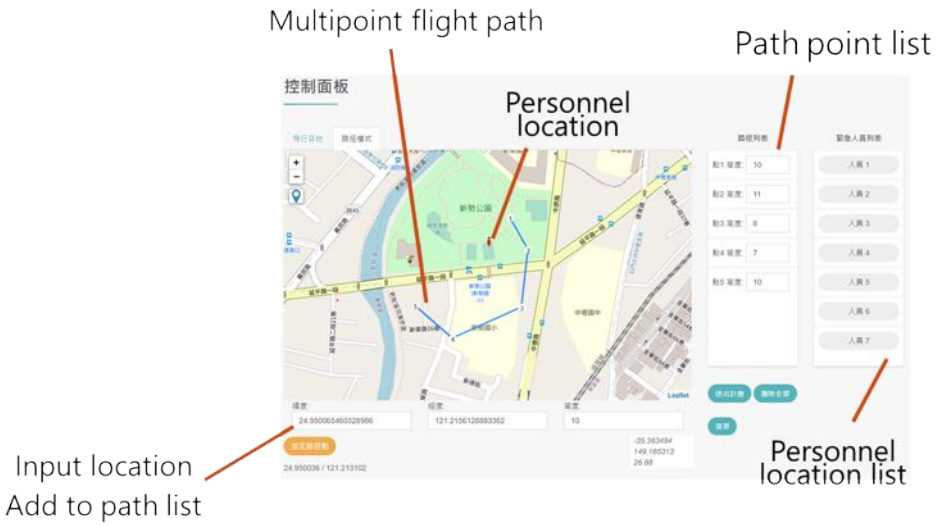

FIGURE 33 DRONE MULTIPLE PANEL INTERFACE

4.2.1. ADS-UC1: Drone Collision Avoidance System

Traditionally, drones are navigated through a set of GPS coordinates which are pre-loaded into the drone navigation software. This scheme, known as Way-point navigation, does not allow autonomous modification to the flight path. In this validation, we showcase the enhancement of the current

navigation system to enable local and remote data processing as well as the dynamic change for flight trajectory for the drone fleet.

To accomplish this, a coordination mechanism among drones in the drone fleet with edge and fog computing is required. At the edge, the Drone Navigation Server can remotely monitor and control each drone. At the fog, each drone uses fog-to-fog communication to share GPS data to support Drone Collision Avoidance System (DCAS). The DCAS interacts with Drone Navigation Server through the mobile network as shown in Figure 34.

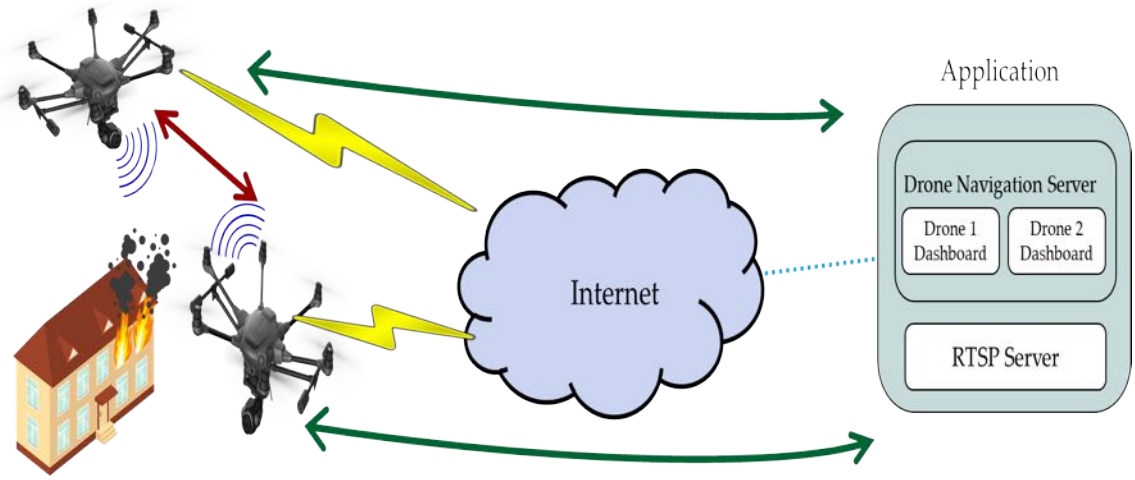

FIGURE 34 DRONE FLEET NAVIGATION ARCHITECTURE

The developed DCAS at the fog node on each drone will detect potential collisions and takes over the control of the drone autonomously to avoid the collisions. The key idea of DCAS is using virtual cylinders (see Figure 35) to detect potential collisions. Each drone has two concentric cylinders. These two cylinders are centred on the position of the drone itself as follows:

1. A Collision Cylinder is used to detect potential collisions. If Collision Cylinders were overlapped, the drone then tries to avoid potential collisions.
2. A SlowDown Cylinder is used to slow down the drone's speed. If SlowDown Cylinders were overlapped, the drone then tries to slow down its flying speed.

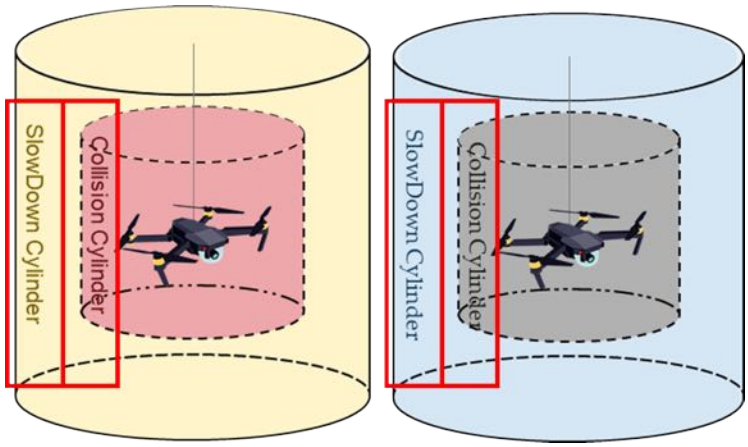

FIGURE 35 VIRTUAL CYLINDERS OF DCAS

In the drone fleet navigation mission shown in Figure 36, each drone keeps exchanging Drone ID and GPS data. Based on the updated trajectories, the drone navigation server commands the drones to head to the target destinations of the flight mission. When DCAS detects a potential collision, it performs a collision avoidance procedure until the risk is resolved.

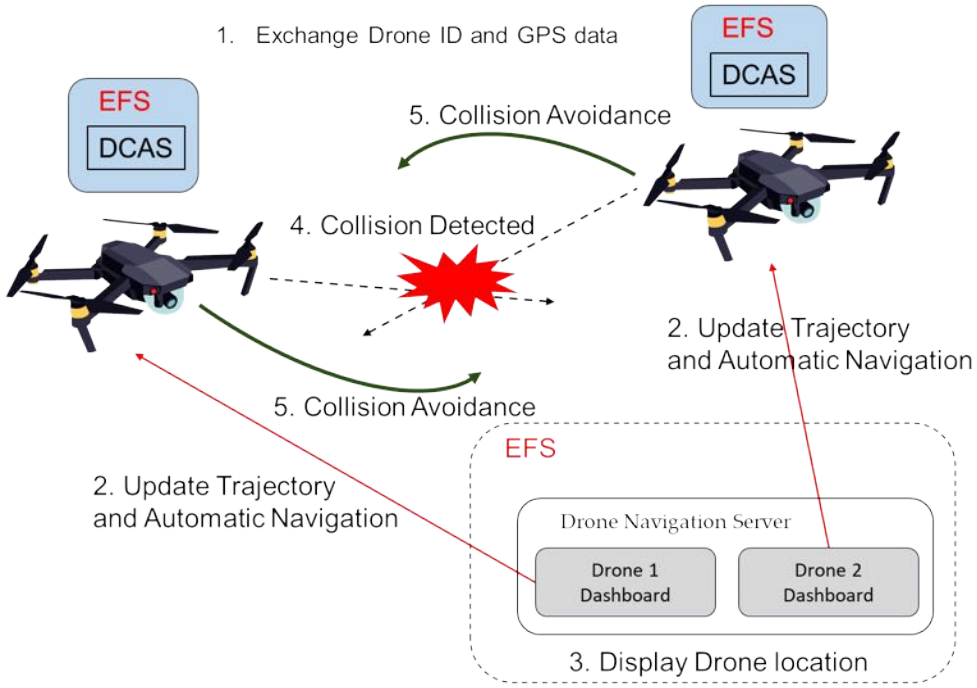

FIGURE 36 DRONE FLEET NAVIGATION MISSION WITH DCAS

ADS-CU1 mission operation procedures are as follows. We have two drones connected to the Drone Navigation Client/Server through the mobile network for monitoring and dynamic control of the drones as shown in Figure 37. Besides, we initialize DCAS on each drone and then DCAS will connect with drone navigation client. More details are explained in [24].

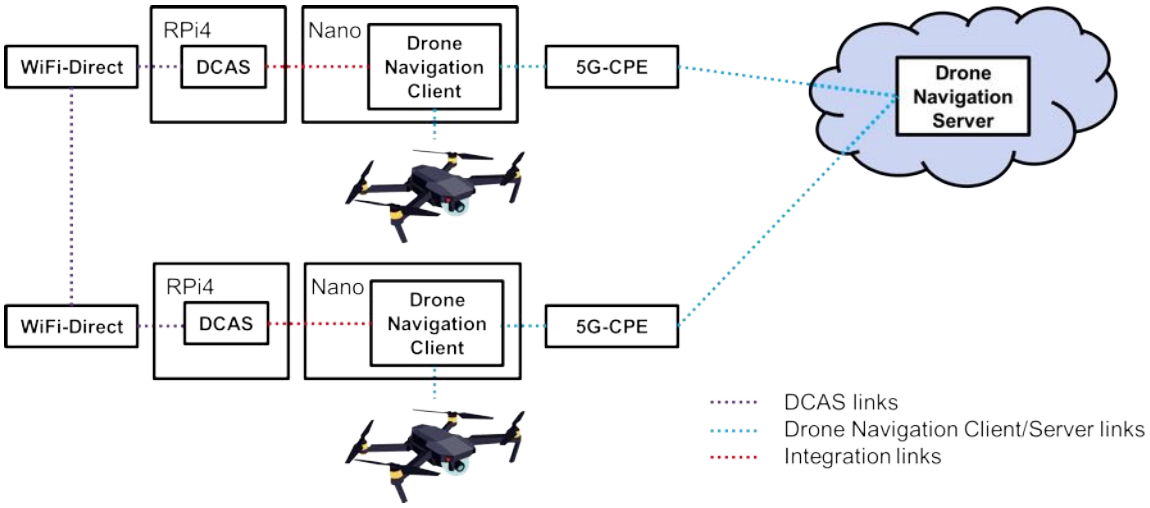

FIGURE 37 DCAS VALDIATION SETUP

In our validation, the drone flies on the ITRI campus. The Drone Navigation Server to controls the drones to take off and set the drones to fly to different destinations. During the flight, if the drones are too close with one another as depicted in Figure 38, the DCAS on each drone will detect potential collisions and takes over the control of the drone for a period of time. After the potential collision is resolved, the drone control returns to the Drone Navigation Server to continue the mission. During the field trial, the location of the drones is displayed on the dashboard (see Figure 39). The video stream from the camera on the drone during the field trial is also shown in Figure 40.

FIGURE 38 DCAS TESTING

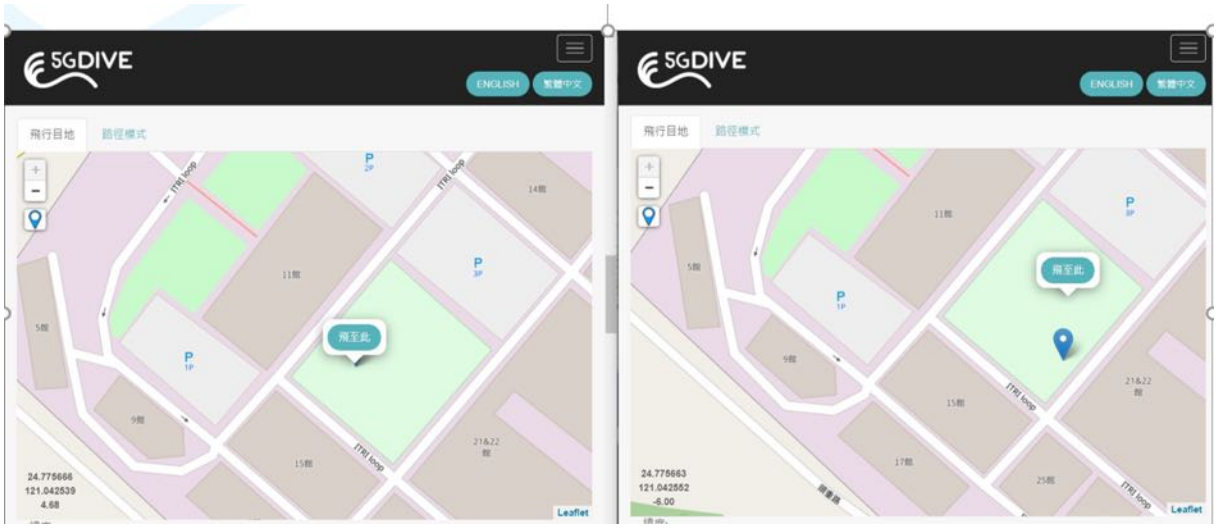

FIGURE 39 DASHBOARD VIEW FOR DCAS TESTING

FIGURE 40 CAMERA VIEW DURING DCAS TESTING

4.2.2. ADS-UC2: Intelligent Image Processing for Drones

The mission's operation procedure of ADS-UC2 are as follows. First, PiH is searched and located in a disaster impacted area in real-time. In our case, PiH is a person who is waving a flag on the 6th floor of

FIGURE 41 PERSON IN NEED OF HELP: FIRST PERSON VIEW (LEFT), DRONE VIEW (RIGHT)

MIRC building balcony (Figure 41). Second, when a PiH is found, the drone will be deployed from the command centre to the PiH GPS location autonomously. Thus, the automatic navigation of the drone drives it towards the location of PiH.

FIGURE 43 ADS-UC2 MISSION SCENARIO: (A) START OF THE MISSION, (B) WHEN A PIH IS FOUND

For the mission, we fly 2 drones, Drone-1, and Drone-2. At the beginning of the mission, Drone-1 will be set and initialized at the backside of the MIRC building. Once Drone-1 is ready, it will take-off from the ground and start doing an autonomous search routine at the backside of the MIRC building. The search itself will be performed approximately from the 3rd floor until the 7th floor of the MIRC building in a zig-zag pattern. Covering all of the backside of the building. During the search, Drone-1 video feed, as well as its GPS location, will be streamed to the EDC using the 4G wireless network. At the EDC,

FIGURE 42 EAGLEEYE OUTPUT

EagleEYE captures the video feed and perform PiH Detection continuously in real-time. EagleEYE PiH detection output can be viewed by the rescue team to monitor the progress. Once EagleEYE detects for PiH, the PiH GPS location will be saved and forwarded to the drone navigation system. With the detection of PiH, Drone-1 completed the search and will land. The drone navigation system received the PiH GPS location and then calculates new waypoints for Drone-2 for automatic trajectory update. Drone-2 will use these waypoints and then fly towards the PiH location autonomously. More details

on the mission scenario can be seen in Figure 43. During the mission, the observer can also view the live mission progression. In Figure 42, we can see the output of the EagleEYE system. To the right of the screen, is the raw video that is captured by the drone. To the left of the screen is the output of EagleEYE. In the bottom left of the screen is the current GPS location of the drone and the detection status of PiH. When a PiH is found, a box will be drawn around the PiH.

4.3. Experimental results

We performed a benchmark to show the baseline network performance during the validation. The benchmark setup can be seen in Figure 44. In this benchmark, we setup a smartphone as a client (representing the stationary drone). This smartphone is connected to the 4G communication network. We perform a ping test to measure the latency, and iPerf3 test to measure the throughput. For the ping test, we get an average of 64.5ms latency with a deviation of 23.4ms. As for the throughput, we get an uplink throughput of 19.5Mbps, and 27.1 Mbps of downlink throughput. A summary of the results can also be seen in

Table 7. Also, it is expected to provide 600Mbps for DL and 100Mbps for UL during using the ideal environment for 5G-NSA solution during the last period of this project.

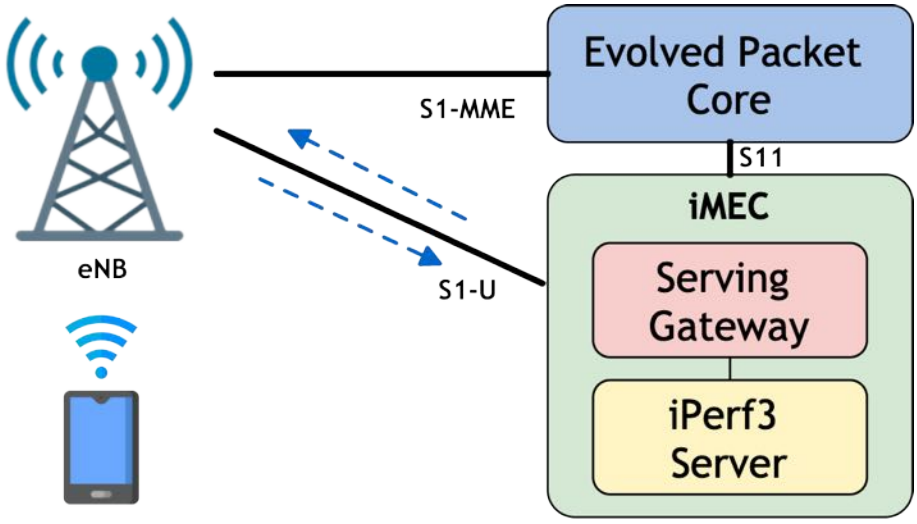

FIGURE 44 BASELINE NETWORK SETUP

ANOTHER BENCHMARK TO SHOW THE BASELINE NETWORK PERFORMANCE DURING THE FIELD TRIAL IS ALSO PERFORMED. IN THIS BENCHMARK, WE MEASURE THE RTT BETWEEN TWO DRONES AS DEPICTED IN

Table 7. The drone-to-drone communication system is evaluated with Raspberry Pi4 and ASUS-USB-AC68 Wi-Fi dongle with 5GHz frequency. The evaluation is conducted on ITRI campus football field, the diagonal distance is approximately 113 meters. As for the evaluation result, the system achieves an average latency of 3.185ms round-trip time. The result shows that the drone-to-drone communication system can support the packet transmission of DCAS in real-time. Also, several test for different components of ADS have been done as shown in the following subsections.

TABLE 7 BASELINE NETWORK TESTING RESULTS

Test Item	Parameter	Result
Latency (End-to-End)	*ping test; 100 samples*	Avg.:64.5ms; Dev.: 23.4ms
Throughput (Uplink)	*iPerf3; ran for 5 minutes*	Avg.: 19.5Mbps
Throughput (Downlink)	*iPerf3; ran for 5 minutes*	Avg.: 27.1Mbps
Latency (Drone-to-Drone)	*ping test; 100 samples* *distance between drones 113meters*	Average :3.185ms (RTT) Dev. 0.65ms

4.3.1. Experiment A: OPTUNS

Power consumption and end-to-end latency were measured to validate for OPTUNS performance. In the power consumption benchmark OPTUNS and an electrical spine-leaf network is configured to include 24 server racks and operate at 40Gb/s per port under fully loaded condition. In this power consumption benchmark, OPTUNS can achieve 82.6% power saving compared to electrical spine-leaf networks. As for the end-to-end latency, NetPipe [25] is used. Each packet for the observed traffic (TCP) is 1 byte long, and that for the background traffic (UDP) is 1470 bytes in length. The experimental results of mean and p99 end-to-end latency under various loads and traffic locality values are shown in Figure 45 below. A mean and p99 end-to-end latency result of less than 17 μs is achievable by OPTUNS. More details on OPTUNS can be found in [20].

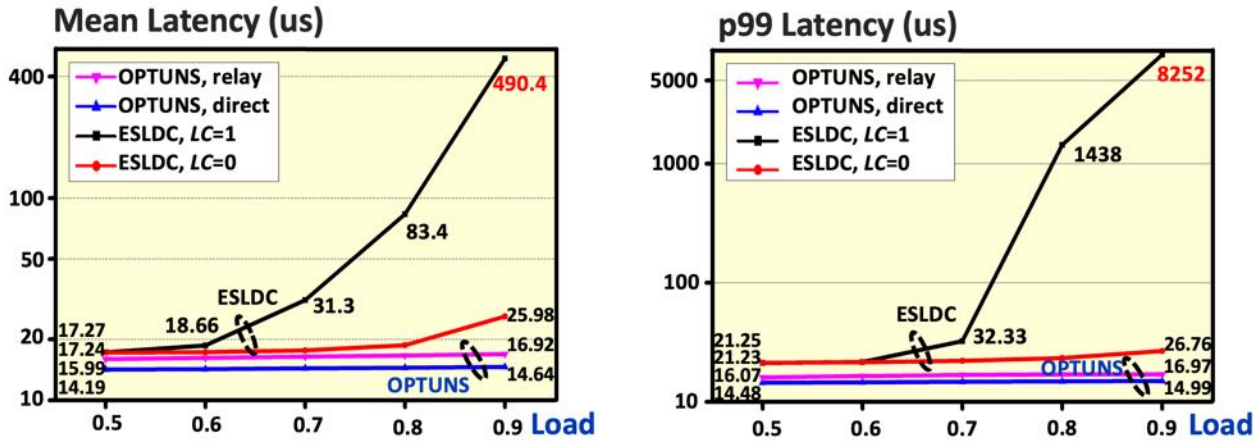

FIGURE 45 OPTUNS END-TO-END LATENCY BENCHMARK

4.3.2. Experiment B: iMEC

To measure the bandwidth that iMEC can handle, we leverage XENA 3GPP packet generator to inject traffic from 3GPP S1-U interface and read the bandwidth number from App as shown in Figure 46 iMEC Lab setup for bandwidth test. On 1Gbps physical line rate, iMEC handles 960Mbps data flow in uplink direction, from eNB to the localized application, as well as 950Mbps in reverse direction. The results show iMEC is able to achieve at around 95% bandwidth utilization.

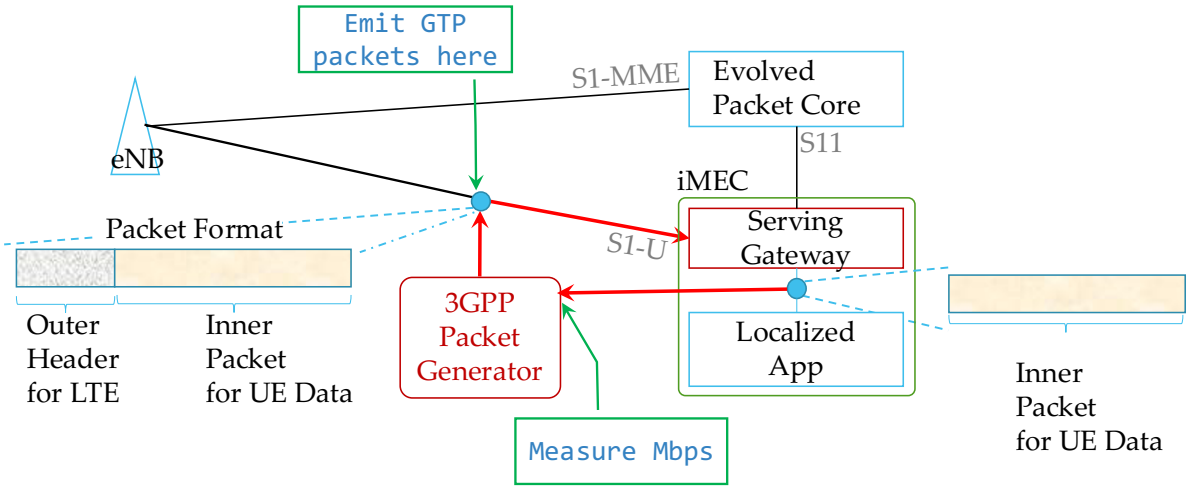

FIGURE 46 IMEC LAB SETUP FOR BANDWIDTH TEST

Latency is another important performance index for transmitting what a drone sees to the control center. To measure the latency incurred by iMEC, a drone keeps pinging the localized drone application server, as shown in Figure 47 Lab setup for latency test. Based on a set of 50 measurements, it has minimum 30ms, maximum 73ms, average 49.28ms, and standard deviation 10.537, which means round-trip-time has 95% confidence between 46.359ms to 52.201ms.

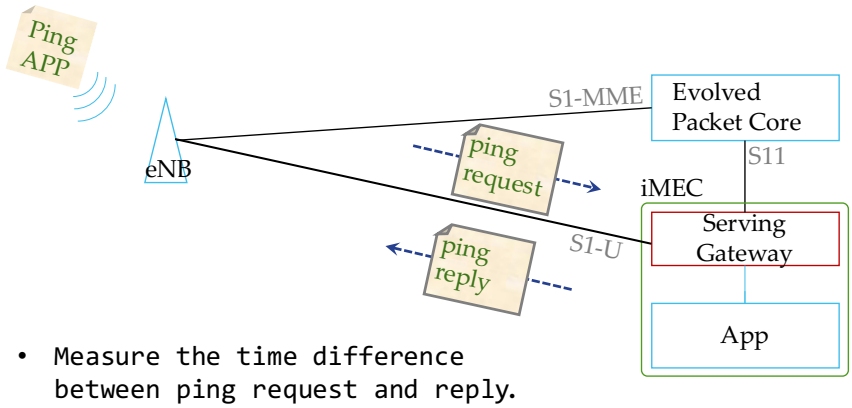

FIGURE 47 LAB SETUP FOR LATENCY TEST

4.3.3. Experiment C: EagleEYE

EagleEYE employs YOLOv3 [26], a CNN based algorithm to perform the real-time image processing performance for the detection of PiH. In the validation, we categorize PiH as a person who is holding and waving a flag. EagleEYE is trained using a combination of the public dataset for 'person', and 'flag' object [27], [28]. EagleEYE will use the detection of these 2 objects to make decisions if those detected objects can be categorized as a valid PiH or not. More details on the public dataset used for training can

be seen in Table 8 below. We call the weight trained with these datasets as Trained Weight 1 or TW-01 for short.

TABLE 8 EAGLEEYE TRAINING DATASET

Trained Weight	'Person' [27], [28]	'Flag' [28]	Total Training Images
TW-01	3000 imgs [27], 3000 imgs [28]	6000 imgs	6000 imgs

As for the experimental results, EagleEYE achieves an Average Precision (AP) of 98.21% for 'person' object, and 75.91% for 'flag' detection. A summary of these test results as well as its mean Average Precision (mAP) is provided in Figure 9 below. For the test, we are using our custom dataset which we call TED-01. TED-01 test dataset consists of 57 images of a person who is waving a flag at a roadside on NCTU campus. For the per-frame processing latency, we recorded a latency of 44.8ms. With the help of pipelining and parallel computation techniques, we can perform the frame processing in real-time.

TABLE 9 TESTING RESULTS OF EAGLEEYE

mAP Test	TED-01		
	AP of Person	**AP of Flag**	**mAP**
TW-01	98.21%	75.91%	87.06%

4.3.4. Experiment D: DCAS

DCAS is a distributed on-drone computing system to detect and avoid potential collisions between drones based on GPS data exchanged with the drone-to-drone communication system. More details on DCAS can be found on [24]. The DCAS is integrated with drone navigation server/client to enable drone collision avoidance function. During the drone mission, DCAS would detect and avoid potential collisions by sending new trajectories to the drone navigation client. The DCAS is evaluated with drone emulation software [29] within a PC (Intel Core i7-4510U CPU, 4GB Memory, Ubuntu 16.04 LTS).

At the beginning of the DCAS emulation, two drones are emulated with 42 meters between each other. Both two drones fly toward each other. When the drones are getting too close, i.e., the collision cylinders of the drones are overlapped, the DCAS will detect and avoid the potential collision by swapping both of them. Eventually, we emulation the result with three different radii of the collision cylinders is shown in Figure 48.

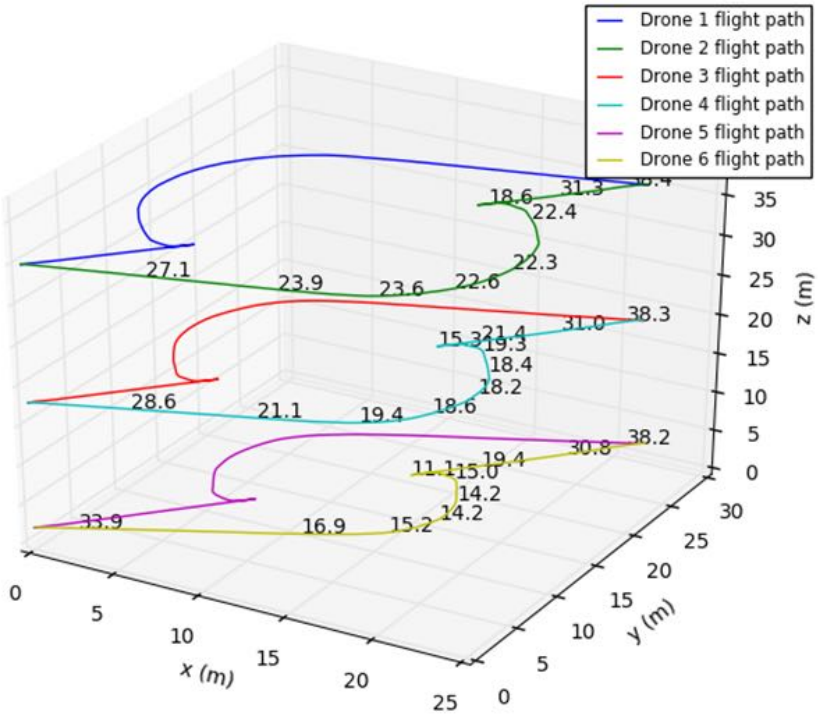

FIGURE 48 VARIOUS DCAS TRAJECTORY MEASUREMENTS

In this figure, the top 2 lines are the trajectories of the 2 drones with collision cylinders is 11 meters. The middle 2 lines are the trajectories of the 2 drones with collision cylinders is 9 meters. The bottom 2 lines are the trajectories of the 2 drones with collision cylinders is 7 meters. The numbers on the line are the distance between the 2 drones during the emulation. As for the evaluation result, the smaller radius of the collision cylinder, the smaller airspace to perform collision avoidance. This evaluation also shows that the functionality of DCAS is configurable by changing the radius of collision cylinders. In addition, we measure the effect of the Collision Cylinder Radius versus Elapsed time. Besides, a number of drone effect on Elapsed time is shown in Figure 41. Obviously, a higher Radius requires more time to reach target GPS location for any given mission. Also, as the number of drones increases, we notice a minor time increment to reach target GPS location for any given mission.

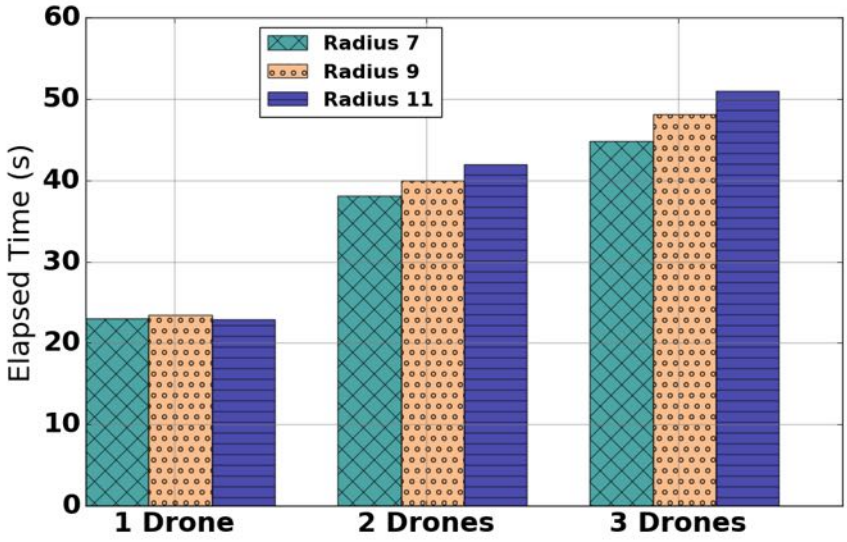

FIGURE 49 ELAPSED TIME VS NUMBER OF DRONES MEASUREMENTS

4.3.5. Experiment E: NSA EPC

For NSA EPC, we both do the functionality test and performance test via emulators like Spirent Landslide and VIAVI TeraVM to make sure the ability for supporting the ADS use cases.

In the functionality test, it takes the surround test to verify the necessary features of NSA EPC like E-UTRAN/5G NR connectivity, Authentication and Security for the device attachment, and Static/Dynamic IP address allocation.

To ensure the stability of both the device attachment and session establishment in the control plane, the stress test shown as Table 10 is monitoring the multiple devices reattach using IMSI and GUTI, after UEs succeed the attach procedure for 15 seconds in the idle mode.

TABLE 10 CONTROL PLANE ATTACH STRESS TEST OF EPC

Attach UEs	Reattach Activation Rate (UEs/Sec)	Attach Accept (UEs)
1,500	25	1,500
10,000	0.5	10,000
10,000	1	7,849
10,000	3	5,072

For the data plane performance test, it follows the ADS use cases network architecture which NSA EPC integrates with iMEC, and then using the VIAVI TeraVM to emulate UE, eNB and gNB to take the stress test. As Figure 50 shown below, the throughput can achieve UL and DL almost 1 Gbps performance.

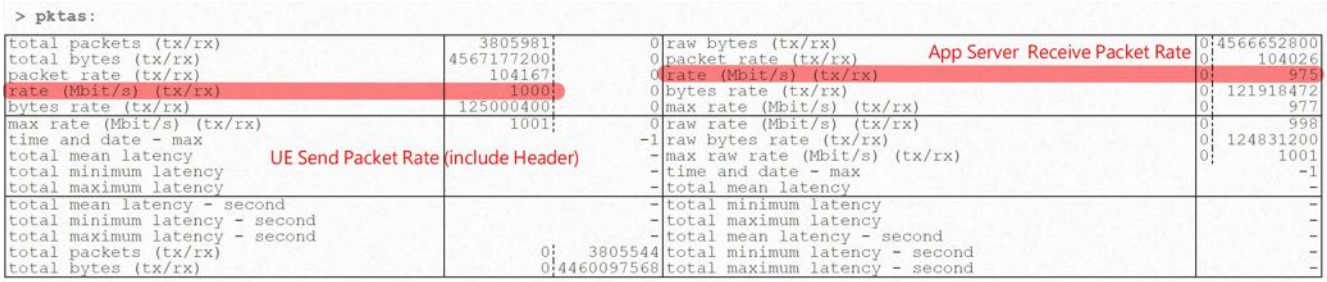

FIGURE 50 DATA PLANE THROUGHPUT STRESS TEST OF NSA EPC

4.4. Initial KPI validation of ADS use cases

The following Table 11 summarizes the initial KPI validation against the technical requirements for ADS use cases defined in WP1 [2] based on the experimental results presented in Section 4.3 above. As we can see from Table 11 for ADS-UC1/ADS-UC2 use case, end-to-end latency, the downlink bandwidth used for the remote control of the drone and the uplink bandwidth utilized by video for future needs of the intelligence engines were tested and found to be satisfying the requirements for only one drone. Hence, 5G-NSA network is needed to satisfy the demand especially when ADS use case adopts three drones or more. The latency can be also improved as adapting the 5G-NSA solution in ADS.

TABLE 11 ADS USE CASE KPI EVALUATION BASED ON THE EXPERIMENT RESULTS

Technical Requirements (WP1)	Description	Value	Evaluation with 4G Network*
TR-ADS-UC1/UC2-01	Uplink data rate (Drone to Network)	50 Mbps	Not satisfied
TR-ADS-UC1/UC2-02	Downlink data rate (Network Drone)	150 Mbps	Not satisfied
TR-ADS-UC1/UC2-03	Uplink Latency (Drone to Network)	100ms	✓
TR-ADS-UC1/UC2-04	Downlink latency (Network to Drone)	20ms	Not satisfied
TR-ADS-UC1/UC2-05	Positioning accuracy	10m	Tested and around 1.5m
TR-ADS-UC1/UC2-06	Altitude	15m to 100m	✓
TR-ADS-UC1/UC2-07	UE speed	0 to 10m/s	Tested up to 5m/s
TR-ADS-UC1/UC2-08	Number of UEs	3 to 5	2, 3 planned in 2021
TR-ADS-UC1/UC2-09	Image quality	1080p	✓
TR-ADS-UC1/UC2-10	Service reliability	99.99%	Not tested

*5G NSA from June 2021

4.5. Next Steps

Connectivity wise, the next step for ADS will be to replace the 4G network solution with 5G-NSA solution. As for implementation wise, the next steps for ADS are threefold. First, the addition of drone charging spot for the multiple drone trial. Second, Zenoh [30] integration as data transmission protocol for streaming data from the drone to the edge. And third, update of EagleEYE [31] processing pipeline to better support for multiple drone trial.

5. Initial integration plan

In this section, an update regarding more information of the trial sites of I4.0 and ADS trials in Taiwan are first provided. Then the initial use-case integration plan is presented regarding how all 5G-DIVE use cases in trials are planned to be integrated with one common DEEP platform.

5.1. Trial site update

In the previous D3.1 [1], we provided the initial information regarding the trial sites of I4.0 and ADS. In this deliverable, the updated information is provided in this section.

5.1.1. I4.0 trial site

The initial I4.0 trial site was located at ADLINK's headquarters in Taipei, Taiwan. The selected location provides facilities for performing experiments and pilot deployments in a realistic scenario, both regarding infrastructure and connectivity. The machinery testing laboratory is located in 235, Taiwan, New Taipei City, Zhonghe District, Jianyi Road, No. 166, B1 floor, building 9F. The following Figure 51 presents a preliminary time plan for the integration activities that may be carried out during the second year of the project, 2021. Note that this plan is likely to be affected by the travel restrictions due to the ongoing COVID-19 pandemics. As described in Section 2, there is an alternative for the European partners to trial I4.0 use cases in 5TONIC premise, in Madrid, Spain, instead.

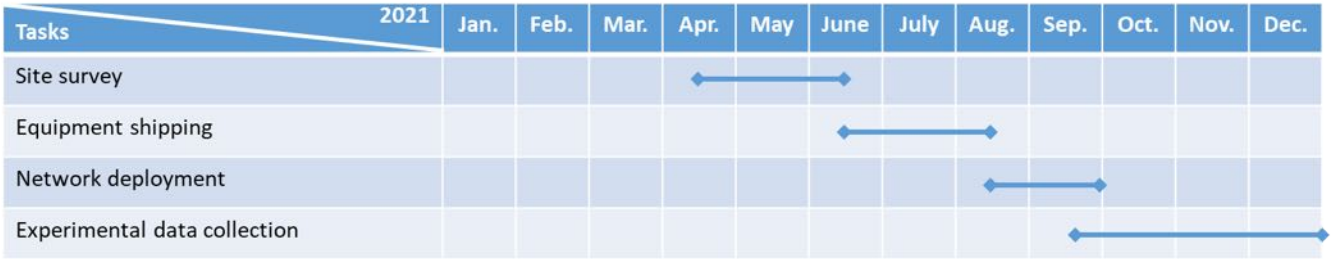

FIGURE 51 INTEGRATION I4.0 TRIAL SITE TIME PLAN

In addition, the expected 5G equipment to be deployed on the trial site is composed of the following items described in the following Table 12.

TABLE 12. 5G EQUIPMENT FOR I4.0 FIELD SITE

5G equipment	Quantity	Vendor/model	Transmit power
5G Core	1	Ericsson	N/A
Baseband Unit	1	Ericsson / BB 6630	N/A
Indoor radio unit	1	Ericsson / IRU 8846	N/A
Radio Dot	2	Ericsson / Dot 4489	Tx power: 4x250mW
5G UE	4	TBD	Power class 2/3, 26/23 dBm (400/200mW)

5.1.2. ADS trial site

In the following, more information of the trial sites for two use cases in ADS trial is provided, respectively.

5.1.2.1. ADS-UC1: ITRI Campus Football Field

For ADS-UC1, the football field is 70 meters long and wide. The size of the football field and its GPS location are shown in Figure 52 . The photos of the field can be seen in Figure 53 below, showing that the football field has enough space, which is very suitable for developing drone-related applications.

FIGURE 52 ITRI FOOTBALL FIELD DIMENSION & GPS LOCATION

FIGURE 53 VIEWS OF ADS-UC1 FIELD TRIAL

5.1.2.2. ADS-UC2: NCTU Campus MIRC Building

The ADS-UC2 trial is conducted at the Microelectronics and Information Systems Research Center (MIRC) [32] building premises. The MIRC building is an 8-story building located inside of NCTU campus. For the trial, we used the back area of the MIRC building. At the back area of the building is also where we set up the command centre booth, and also the viewing area. The dimensions of the building and its GPS location can be seen in Figure 54. Photos of the surrounding area can be seen in Figure 55 below. As seen in the photos, we have a very limited area to work with. We only have around 6m of width. Make it even more challenging are the presence of trees surrounding the areas.

FIGURE 54 MIRC BUILDING DIMENSION & GPS LOCATION

FIGURE 55 SURROUNDING AREA OF MIRC TRIAL SITE

5.2. Initial plan for use-case integration

In WP3, one of the main challenges is to develop a common DEEP platform that can serve multiple vertical use cases simultaneously. In this section, we present the current design and plan how 5G-DIVE use cases will be integrated under the same DEEP platform. The basic system design supporting integration of multiple use cases is illustrated Figure 56. In Figure 56, the upper part illustrates the common DEEP components (in BASS and IESS) shared by all use cases and the lower part illustrates the use-case specific components, including the user applications and services, orchestrator and IESS inference applications, as well as active monitoring module, which subjects to use-case specific design choices. The following describes the key features of the design and how multiple use cases can be supported.

- BASS provides one unified Dashboard as the common user interface for verticals to manage all user applications and services, as well as their corresponding intelligence capabilities from IESS.
- BASS is designed to support multiple orchestration frameworks (illustrated as Orchestrator X and Y in Figure 56) to orchestrate the user applications and services, thanks to the BASS module of orchestrator drivers in BASS, which translates the business blueprints to the instructions understood by different orchestration frameworks, e.g., K8S and FogO5 etc. This gives the developers freedom to select the suitable orchestrators to develop with. For example, K8s is widely supported with the richness of available open-source tools in its eco-system which makes the development work more efficient. FogO5 fits more the applications running on constrained devices. If one use case involves using both constrained devices and more capable computing infrastructure (e.g., Edge severs), it can use FogO5 to orchestrate the constrained resources while K8s to orchestrate the more powerful resources. This makes the DEEP design more flexible and easier to be adopted. Existing project can be easily ported to DEEP platform.
- IESS has the following four key components:
    - IESS catalog stores the AI/ML models to be used by different user applications and services.

- Intelligence engines provides the training services to update the AI/ML models stored in the IESS catalog. The training can be done either with the data sets collected from the user applications and services or the external data sets ported from outside.
- IESS inference applications collect the data from the user applications and services. Then the models obtained from the IESS catalog are applied to the data to draw inference results and then send the results to the user applications and services. In Figure 56, the interference applications are illustrated close to the user applications and services, to emphasize that the interference applications are use-case specific where other IESS components are common.
- IESS manager interacts with BASS to manage other IESS components.

- In this initial plan, DASS is mainly used to provide the data path between components and manage the data flows.

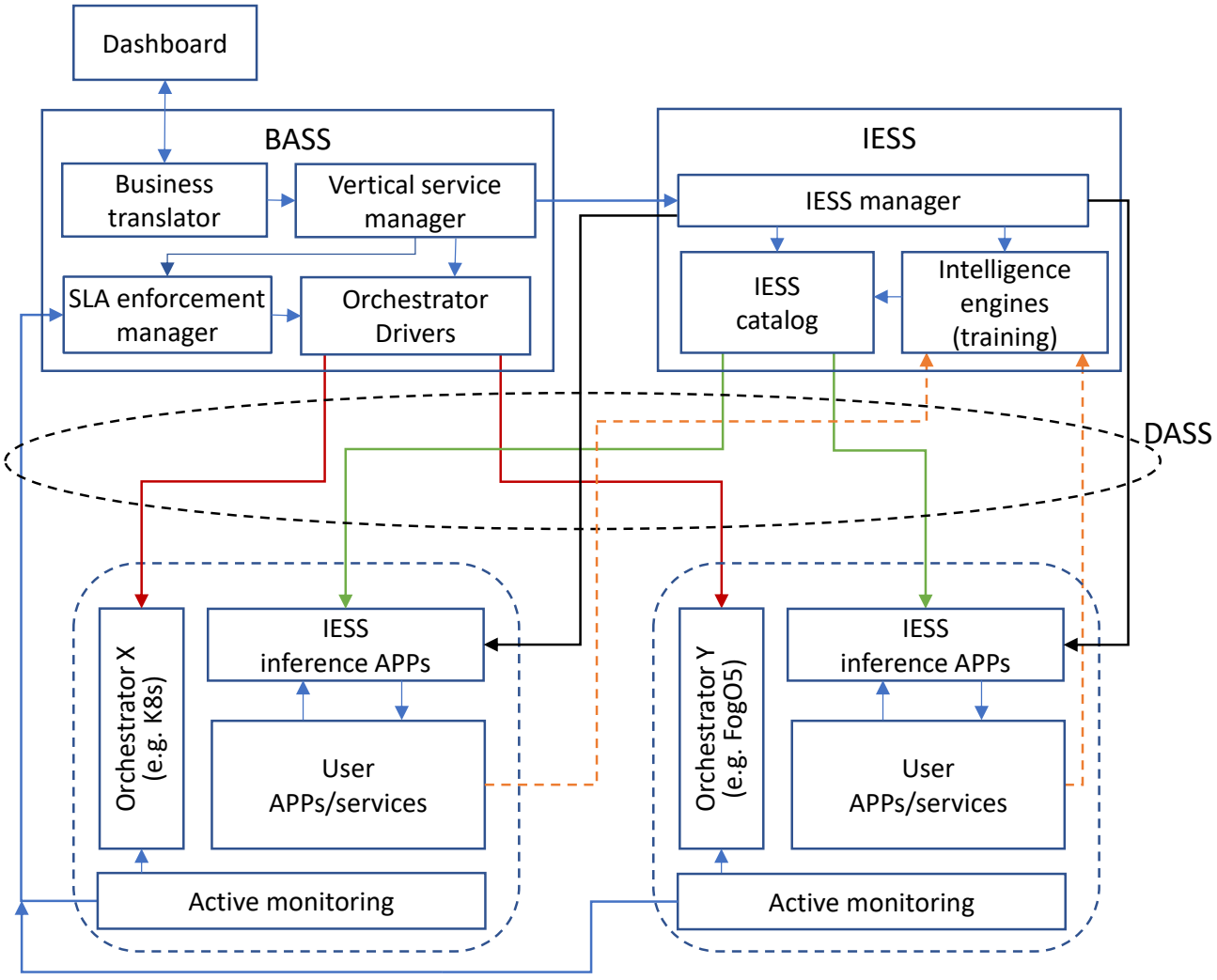

FIGURE 56 ILLUSTRATION OF USE CASE INTEGRATION WITH A DEEP PLATFORM

In this way, multiple use cases can be integrated using a common DEEP platform providing one user-interface management, multiple orchestrator support and ML/AI capabilities. The following describes some more details about the integration plan for all 5G-DIVE use cases in both I4.0 and ADS trials using the same DEEP platform.

- **BASS integration:**
  - Common user interface (Dashboard) managing all use cases.
  - Description of all the use-cases by means of the Vertical Service Blueprints, a structured data model to define vertical service templates for Digital Twin, ZDM, mMTC, DCAS and IIPFD use cases, respectively.
  - Validation of the deployment and life-cycle management of all the use-cases via the BASS dashboard
  - Enable use-case monitoring with the BASS
- **IESS integration:**
  - A common IESS catalogue storing use-case specific ML/AI models used by all use cases.
  - Regarding training, some use cases perform only training with offline data sets, e.g., the data set obtained from lab/trial experiments, external sources (e.g., public library), etc., while others perform training by using the operational data sets gathered when the system is on operations. Table 13 shows how different use cases plan to train their AI/ML models.

TABLE 13 TRAINING WITH INTERNAL AND/OR EXTERNAL DATA SETS FOR DIFFERENT USE CASES

Data set	DT	ZDM	mMTC	DCAS	IIPFD
Use operational data sets?	Yes	Yes	No	NA	No
Use offline data sets?	(Yes)*	Yes	Yes	NA	Yes

*According to the development status only the data generated by the DT service is being considered to train the AI/ML models, however the possibility of enhancing the dataset with relevant and additional data to improve the service or the network performances is currently under evaluation.

- **DASS integration:** All use cases plan to use Eclipse Zenoh for a common DASS implementation.
- **Orchestrator:** Two types of orchestrators, i.e., K8s and FogO5, are planned to be used by different use cases. Table 14 shows what orchestrators each use case plan to use.

TABLE 14 ORCHESTRATOR USED BY DIFFERENT USE CASES

Orchestrator	DT	ZDM	mMTC	DCAS	IIPFD
K8s	Yes	No	Yes	No	Yes
FogO5	Yes	Yes	No	Yes	Yes

6. Conclusions

This deliverable has presented a detailed view of the initial validation results of different 5G-DIVE use cases. We have elaborated on the setup of Digital Twin (DT), Zero Defect Manufacturing (ZDM) and Massive Machine-Type-of-Communication (mMTC) use cases for Industry 4.0 (I4.0) trial. Also, we have elaborated on the setup of Drone Collision Avoidance System (DCAS) and Intelligent Image Processing for Drones (IIPFD) use cases for Autonomous Drone Scouting (ADS) trial. Then, mission scenarios and flows are presented for all the aforementioned use case of I4.0 and ADS pilot. Indeed, the experiment results depicted how I4.0 and ADS pilots utilize the DEEP, fog and edge computing platform, and 5G connectivity in different levels toward a better and reliable services. In addition, the integration plan toward the end of this project has been presented including updated information of trial sites in Taiwan. In summary the achievements and future direction is summarised in Table 15.

The specification in this deliverable already served as a basis for initial implementations and reported several experiment results. In the next deliverable, we will report on performance evaluation of 5G-DIVE platform in vertical field trials while ensuring that KPI are achieved for 5G intelligent services.

TABLE 15 ACCHIVMENTS AND FUTURE DIRECTIONS PER USE CASE

Use Case	Main Achievements	Future Direction
DT	Experiments on 5G connectivity ,to perform robot control and to perform its digital replica	To apply AI/ML and statistical methods in order to implement one or more of the following features: movement prediction, task learning, predictive maintenance, control-loop optimization.
ZDM	Experiments on current 4G connectivity option to find the defected goods and experiments on telemetry solution usage	Integrate the testbed with 5G SA network, as well as other elements of the DEEP platform, namely the DASS, the BASS and the IESS. The introduction of a RAN control app is also in the plans, where telemetry data will be used to train an Intelligent Engine (IE) to efficiently use available access technologies (e.g., 5G and Wi-Fi) and steer access traffic to provide slice/service requirements such as reliability or high throughput for multiple users.
mMTC	Experiments on cloud native design of softwarized IoT stacks facilitating resource saving and pooling	To develop and add orchestration features using K8s for automation and auto scaling. An intelligent application of PHY security enhancement based on RF fingerprinting will be further developed and integrated into the tested. Further, it is also planned to investigate the possibility to use RF fingerprinting for cellular systems such as 5G.
DCAS	Experiments on 4G/Wi-Fi to detect a potential collision and	To integrate 5G-NSA solution and utilize DCAS in drone charging spot for the multiple drone trial

	performs a collision avoidance procedure until the risk is resolved.	
IIPFD	Experiments on end to end 4G network for object detection for PiH	To integrate 5G-NSA solution. To integrate Zenoh as data transmission protocol for streaming data from the drone to the edge. Update of EagleEYE processing pipeline to better support for multiple drone trial.

7. References

[1] "D3.1: Definition and setup of vertical trial sites," 2020. [Online]. Available: https://5g-dive.eu/wp-content/uploads/2020/05/D3.1.pdf.

[2] "D1.1: 5G-DIVE architecture and detailed analysis of vertical use cases," 2020. [Online]. Available: https://5g-dive.eu/wp-content/uploads/2021/01/D1.1_Final.pdf.

[3] "5TONIC Lab, Madrid, Spain," [Online]. Available: https://www.5tonic.org/.

[4] "DELL PowerEdge R430 Rack Server," [Online]. Available: https://www.dell.com/dm/business/p/poweredge-r430/pd.

[5] J. Redmon, "YOLO: Real-Time Object Detection," [Online]. Available: https://pjreddie.com/darknet/yolo/.

[6] V. GUEANT, "Iperf - The TCP, UDP And SCTP Network Bandwidth Measurement Tool.," 2021. [Online].

[7] B. Bloessl, "GNURadio IEEE 802.15.4 Source Code," 2021. [Online]. Available: https://github.com/bastibl/gr-ieee802-15-4.

[8] "Contiki-NG webpage," 02 2021. [Online]. Available: https://github.com/contiki-ng/contiki-ng.

[9] "Ettus Reasearch," 2021. [Online]. Available: https://www.ettus.com/.

[10] "ZMQ documentation," 01 02 2021. [Online]. Available: https://zeromq.org/get-started/.

[11] "Zoleria Firefly," 2021. [Online]. Available: https://zolertia.io/product/firefly/.

[12] "5G-DIVE Innovations Specification," 10 06 2020. [Online]. Available: https://5g-dive.eu/wp-content/uploads/2021/01/D2.1-5G-DIVE-innovations-specification_v1.0_compressed.pdf.

[13] "Docker documentation," 17 01 2021. [Online]. Available: https://docs.docker.com/get-started/.

[14] "GitHub/gr-lora," 17 01 2021. [Online]. Available: https://github.com/rpp0/gr-lora.

[15] "Telegraf documentation," 17 01 2021. [Online]. Available: https://docs.influxdata.com/telegraf/v1.17/.

[16] "InfluxDB documentation," 10 02 2020. [Online]. Available: https://docs.influxdata.com/influxdb/v2.0/get-started/.

[17] "Grafana documentation," 06 02 2020. [Online]. Available: https://grafana.com/docs/grafana/latest/getting-started/.

[18] O. Liberg etal., Cellular Internet of Things: Technologies, Standards, and Performance, Elsevier Science Publishing Co Inc, 2017.

[19] “Big Data IoT WebPage,” [Online]. Available: https://5gdive.bigdataiot.com.tw/.

[20] Yuang, M., Tien, P.L., Ruan, W.Z., Lin, T.C., Wen, S.C., Tseng, P.J., Lin, C.C., Chen, C.N., Chen, C.T., Luo, Y.A. and Tsai, M.R., “OPTUNS: Optical intra-data center network architecture and prototype testbed for a 5G edge cloud,” *Journal of Optical Communications and Networking*, vol. 12, no. 1, pp. A28--A37, 2020.

[21] “Mongo Database,” [Online]. Available: https://www.mongodb.com/. [Accessed 23 9 2020].

[22] “Redis Database,” [Online]. Available: https://redis.io/. [Accessed 23 9 2020].

[23] “EasyDarwin,” [Online]. Available: https://github.com/EasyDarwin/EasyDarwin. [Accessed 23 9 2020].

[24] “D2.1: 5G-DIVE Innovations Specification,” 2020. [Online]. Available: https://5g-dive.eu/wp-content/uploads/2021/01/D2.1-5G-DIVE-innovations-specification_v1.0_compressed.pdf.

[25] “NetPipe,” [Online]. Available: https://www.ameslab.gov/scl/netpipe.. [Accessed 1 12 2020].

[26] Redmon, Joseph and Farhadi, Ali, “Yolov3: An incremental improvement,” in *arXiv preprint arXiv:1804.02767*, 2018.

[27] Lin, Tsung-Yi and Maire, Michael and Belongie, Serge and Hays, James and Perona, Pietro and Ramanan, Deva and Doll{\'a}r, Piotr and Zitnick, C Lawrence, “Microsoft coco: Common objects in context,” in *European conference on computer vision*, 2014.

[28] Kuznetsova, Alina and Rom, Hassan and Alldrin, Neil and Uijlings, Jasper and Krasin, Ivan and Pont-Tuset, Jordi and Kamali, Shahab and Popov, Stefan and Malloci, Matteo and Duerig, Tom and others, “The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale,” in *arXiv preprint arXiv:1811.00982*, 2018.

[29] “DroneKit,” [Online]. Available: https://dronekit-python.readthedocs.io/en/latest/develop/sitl_setup.html . [Accessed 3 12 2020].

[30] “Zenoh,” [Online]. Available: http://zenoh.io/. [Accessed 2 12 2020].

[31] Ardiansyah, Muhammad Febrian and William, Timothy and Abdullaziz, Osamah Ibrahiem and Wang, Li-Chun and Tien, Po-Lung and Yuang, Maria C, “EagleEYE: Aerial Edge-enabled Disaster Relief Response System,” in *2020 European Conference on Networks and Communications (EuCNC)*, 2020.

[32] “Microelectronics and Information System Research (MIRC-NCTU),” [Online]. Available: https://eic2.nctu.edu.tw/. [Accessed 5 12 2020].

[33] 3GPP, “TS 22.104 v17.2.0: Service requirements for cyber-physical control applications in vertical domains, 2019-12”.

[34] “GNURadio webpage,” [Online]. Available: https://www.gnuradio.org/.

[35] “LoRa webpage,” [Online]. Available: https://www.semtech.com/lora.

[36] “Pycom Fipy webpage,” [Online]. Available: https://docs.pycom.io/index.html.

[37] P. Vijayendra Walvekar, “Virtualizing LoRa baseband functionalities to the Edge,” 2019.

[38] “5TONIC Lab,” [Online]. Available: https://www.5tonic.org/.

8. Appendix A: ADS connectivity setup

The setup of small cell and CPE operation are detailed in the following subsections.

8.1. Small cell setup

In field trials, the small cell quick setup is as follow:

1. Connect your Small cell to the Internet: Connect the yellow Ethernet cable from the yellow "WAN" port on your Small cell Dual-mode Enterprise to an available Ethernet port on your internet router.
2. Power on your Small Cell: Plugin your AC power adaptor into the power outlet and connect the power cable to the Small cell Dual-mode Enterprise power port.
3. Let your Small Cell setup: After you power on your SmallCell Dual-mode Enterprise, it will go through the self- installation. It can take up to 5~10 minutes to complete the device setup (see Figure 57). Your Small cell Dual-mode Enterprise may download updates and restart during this time.

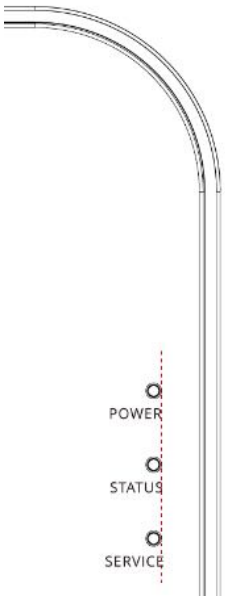

FIGURE 57 SMALL CELL MODES

- Power (Solid Blue): Power on. Device self-testing and update complete.
- Status (Solid Blue): Successful connection established with mobile operator network.
- Service (Solid Green): The device is ready to provide service.

4. Wall mount your Small Cell if desired: Your Small cell Dual-mode Enterprise setup is complete. You should see improved signal strength from your device as shown in Figure 58. Make your first call to enjoy more dependable voice calls and more reliable high-speed data connection.

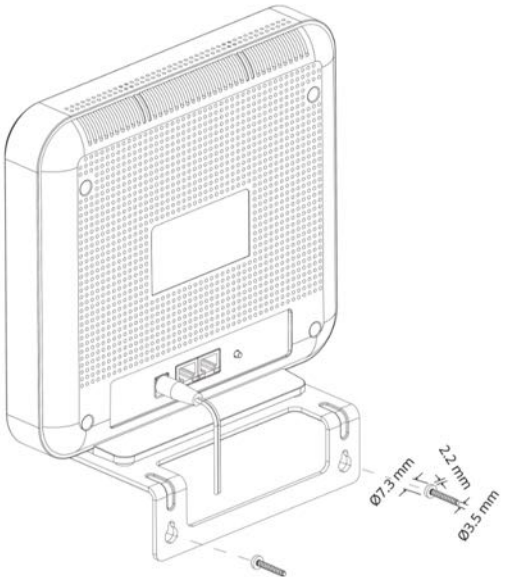

FIGURE 58 SMALL CELL INSTALLATION

TABLE 16 SMALL CELL LED INDICATORS

Case	State to Indicate	Power	Status	Service
1	Power On	RED	OFF	OFF
2	Network is initialising	GREEN	OFF	OFF
3	IPsec Tunnel is initialising	BLUE	RED FLASHING	OFF
4	LTE Service is initialising	BLUE	BLUE FLASHING	OFF
5	Lost Connecting with MME	BLUE	RED	OFF
6	Settting Error at HeMS	BLUE	OFF	RED
7	LTE Service Ready	BLUE	BLUE	GREEN
8	LTE Service In Progress	BLUE	BLUE	GREEN FLASHING
9	LTE Service In Maximum Capacity	BLUE	BLUE FLASHING	GREEN FLASHING
10	Over Temperature	RED FLASHING	NO CHANGE	NO CHANGE
11	RF Issue	RED FLASHING	RED FLASHING	RED FLASHING
12	Software Update	BLUE FLASHING	BLUE FLASHING	BLUE FLASHING
13	GPS Sync Progressing	NO CHANGE	NO CHANGE	NO CHANGE
14	GPS Position Fixed	NO CHANGE	NO CHANGE	NO CHANGE

8.2. CPE setup

After the CPE boot successful, you can access **WebUI Login** (see Figure 59**)**

- Open the browser and browse http://192.168.1.1
- Username/Password: admin/admin

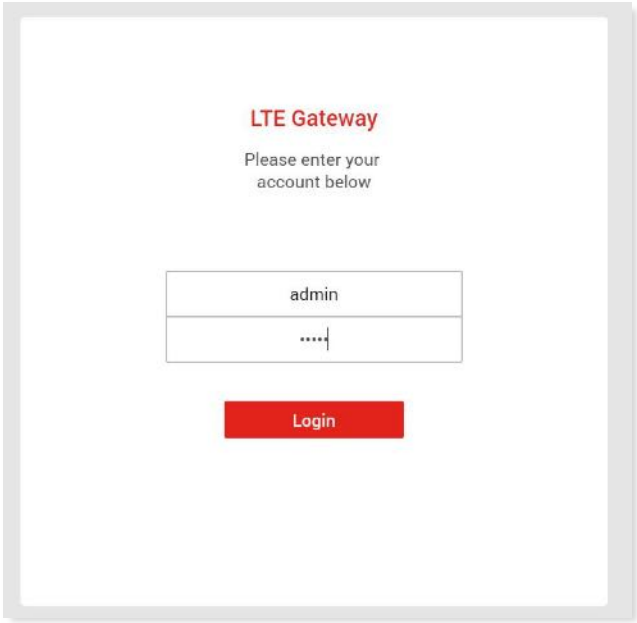

FIGURE 59 CPE WEB LOGIN

Next, you can check CPE Network Status via **Web UI** as shown in Figure 60.

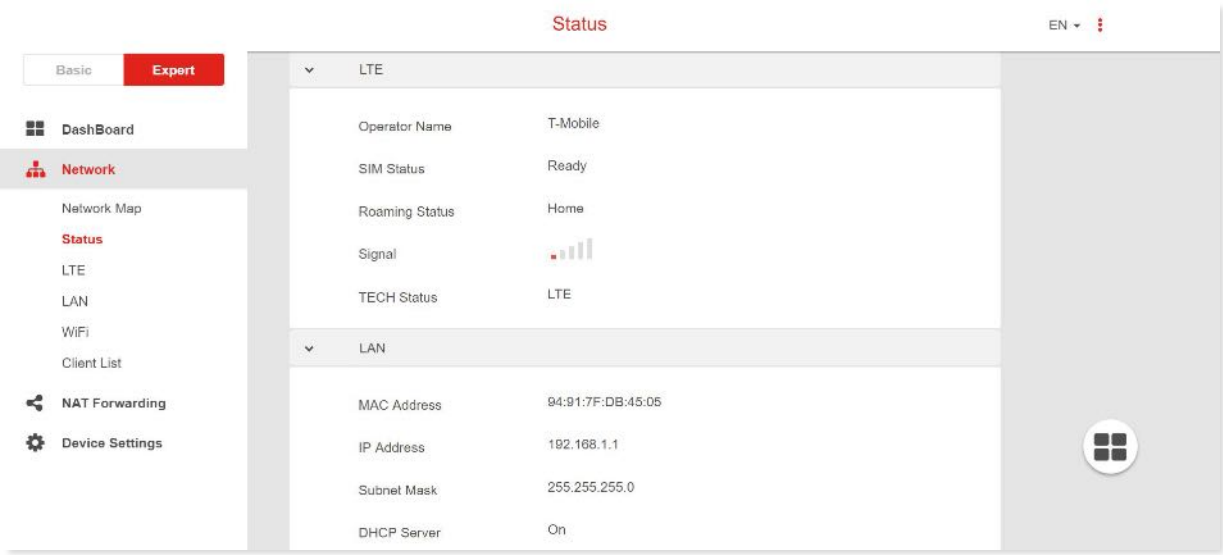

FIGURE 60 CPE CONNECTION STATUS

In addition, CPE connection status can be checked for LED Behavior in **Dual Modes** as follow (see Figure 61):

- Common Operation LED Behavior: Power LED shows in Blue color while the system is powered. Four LEDs are used to indicate the system, battery, Wi-Fi and RAT status
- Query Operation LED Behavior: To present cellular signal Quality and to present battery capacity level

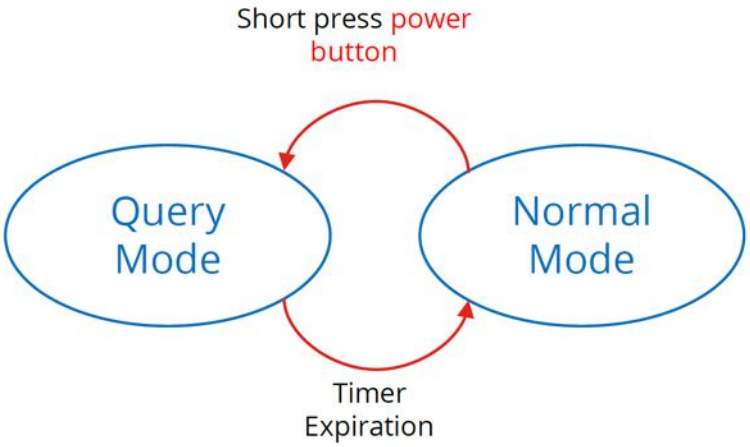

FIGURE 61 CPE MODES

CPE has also two colors to indicate the status of RAT: LTE or 5G, Wi-Fi, Battery and System (see Figure 62)

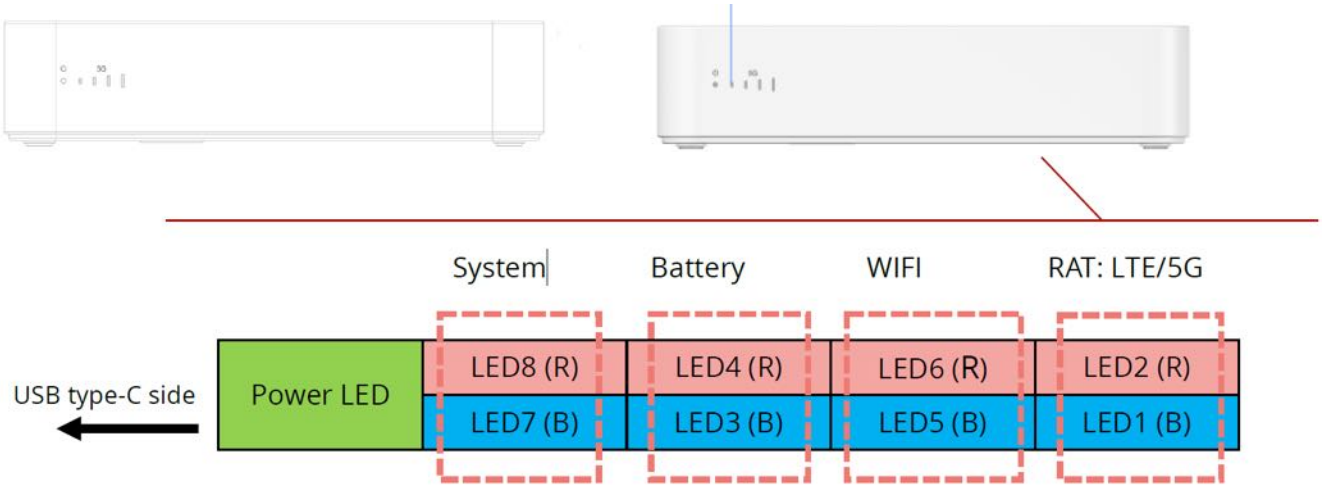

FIGURE 62 CPE LED INDICATORS

CPE Normal Mode

TABLE 17 CPE NORMAL MODE

Indicator	Description	System (Grp4)	Battery(Grp3)	WIFI(Grp2)	LTE/5G(Grp1)
System	System Shutdown	Off	off	off	Off
	System Booting	Blue, blink	off	off	Off
	System Error	Red, Solid	off	off	off
	System Ready	Blue, Solid			
RAT	LTE Ready				Blue, Solid
	5G Ready				Purple, Solid
	Connection Error				Red, Solid
WiFi	WiFi turned on			Blue, Solid	
	WiFi turned off			Off	
	WiFi Error			Red, Solid	
Battery	Battery charging		Red, Solid		
	Battery discharging		Blue, Solid		

CPE Query Mode

- Shortly press the Power Button to enter query mode: It will show signal level in 5s and change to battery level in 5s

TABLE 18 CPE QUREY MODE

Indicator	Level	System (Grp4)	Battery (Grp3)	Wifi (Grp2)	RAT (Grp1)
1. RAT Signal Strength	Good Signal	Blue, Solid	Blue, Solid	Blue, Solid	Blue, Solid
	Fair Signal	Blue, Solid	Blue, Solid	Blue, Solid	off
	Bad Signal	Blue, Solid	Blue, Solid	off	off
	No Service	Blue, blink	off	off	off
2. Battery Level Status	>75%	Red, Solid	Red, Solid	Red, Solid	Red, Solid
	50~75%	Red, Solid	Red, Solid	Red, Solid	off
	25~50%	Red, Solid	Red, Solid	off	off
	<25%	Red, blink	off	off	off

Finally, we can CPE check registration status as shown in Figure 63 and Figure 64 .

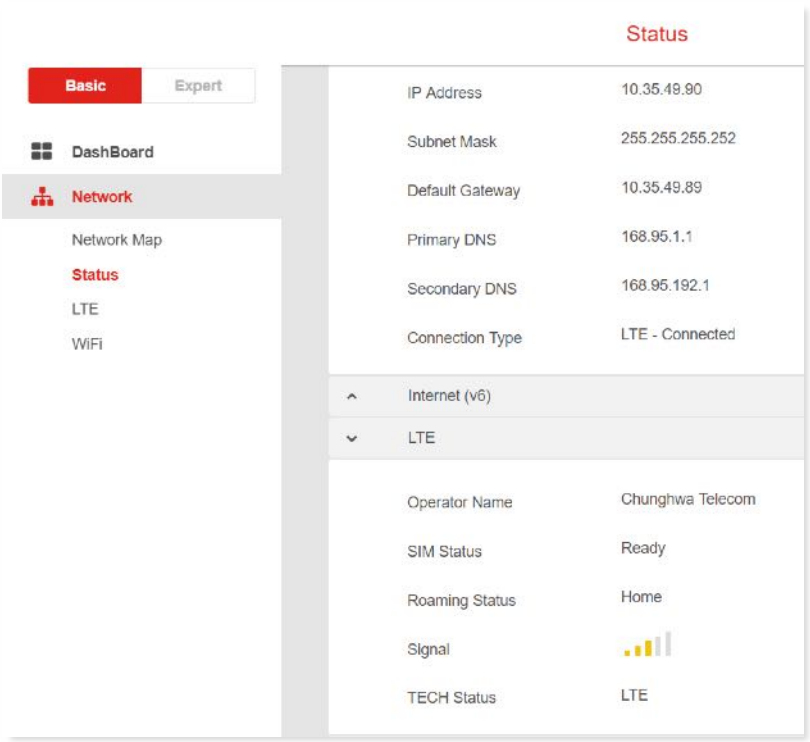

FIGURE 63 CPE REGISTRATION ON 4G

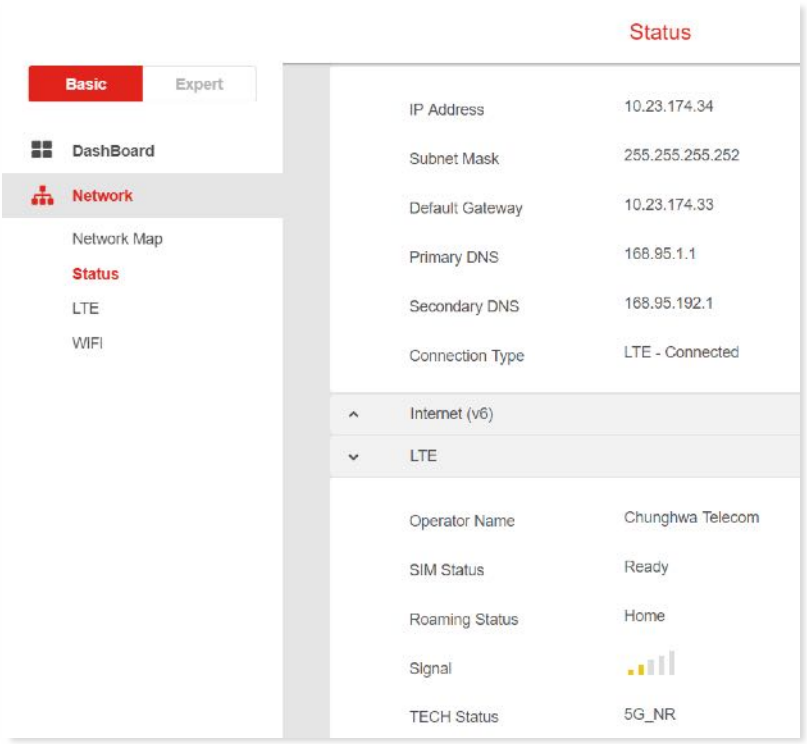

FIGURE 64 CPE REGISTRATION ON 5G