Capgemini engineering

intel

Intelligent 5G L2 MAC Scheduler

Powered by Capgemini NetAnticipate 5G on Intel Architecture

Table of Contents

Executive Summary

Capgemini Engineering along with it's 5G partner ecosystem, has developed a cognitive 5G Medium Access Controler (MAC), which is first in a series of activities planned as part of Capgemini's Project Marconi. Project Marconi includes a series of innovations targeted towards realizing cognitive capabilities as part of Intelligent to RAN Controller nodes. These implementations utilize capabilities offered as part of Capgemini RATIO ORAN RIC powered with NetAnticipate5g and also integrated with Intel FlexRAN. The key idea here is to enhance network performance in terms of spectral efficiency, QoS, and network resource utilization. The Marconi team demonstrated ~15% improvement in network performance on 5G MAC scheduler and near-RT RIC platform. NetAnticipate5G, leveraging Intel AI software, Analytics Zoo, and optimized framework running on Intel® Xeon® scalable processors, improved AI accuracy to 55% and reduced AI inference time to 0.64msec, a 41% improvement[1], which is a critical requirement for such ultra low latency and reliable ml based solutions.

As industries move towards the adoption of Industry 4.0, they are demanding new, more advanced ways to provide services enabled by 5G networks. Many of the advanced 5G-based services we foresee are focusing on private 5G networks, industrial IoT, digital twins, cobots, XR-based immersive gaming and many other applications. Most of these services are changing the way users will experience them. It is expected that many of these 5G-powered services will utilize technologies such as augmented reality (AR), holographic calls, immersive mixed-reality gaming, and even remote-controlled robotic arms. These new services will not only add immense complexity and load to network resources, but they will also require dynamic provisioning of advanced QoS traffic classes like Ultra Reliable Low Latency Communications (URLLC), massive Machine Type Communications (mMTC), and enhanced Mobile Broadband (eMBB), to enable the services. Also, it is almost impossible for CSPs to provision resources dynamically for such demanding services as part of the 5G network. This has led companies to start thinking about adding intelligence to their networks, which gives rise to alliances working on intelligent radio networks.

Capgemini Engineering is part of these alliances and is developing RIC platform enablers. We offer our award-winning NetAnticipate5G framework that introduces AI as part of the RAN intelligent controller (RIC) to support autonomous intelligent operations. Also, RATIO is a Capgemini Engineering RIC platform based on O-RAN Alliance specifications for developing a disaggregated intelligent RAN controller.

Introduction

RAN functions, such as admission control, radio resource scheduler, mobility management, and radio resource management, are currently more static rule-based and cannot adapt to the dynamics of the network or the services accessed by the user. This requires adding intelligence to the RAN functions to assist in the decision-making process, which leads to improved user experience and more efficient network resource utilization.

With the objective of improving RAN functions, Capgemini Engineering launched Project Marconi, in collaboration with Intel, to develop an industrialized approach for introducing ML-assisted intelligence to RAN functions, as shown in Figure 1 below.

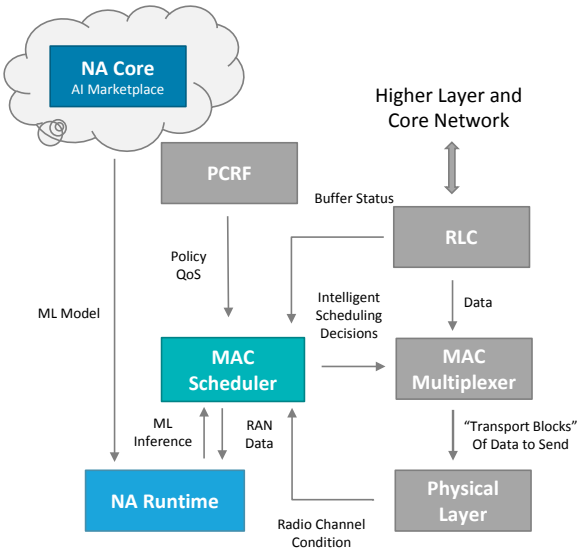


Figure 1. ML-Infused Intelligent MAC Scheduler

Source: Capgemini Engineering

Project Marconi started with research and development to introduce intelligence into the most critical and time-sensitive layer 2 MAC scheduler functions that play a key role in radio resource allocations and ensure QoS requirements are met. A few of the scenarios considered as candidates for an ML-infused intelligent MAC scheduler included Link Adaptation, Massive MIMO, and multi-user MIMO.

Phase I of project Marconi focused on improving efficiency in spectral efficiency by introducing ML-infused intelligence to link adaptation. Implementing an intelligent MAC scheduler requires changes to the resource allocation currently built on conventional Infinite Impulse Response (IIR) filter-based algorithms. This IIR-based approach cannot accurately forecast changes to the UE signal quality by virtue of variations in mobility patterns. The algorithms act on the current reported CQI, RLC buffer occupancy (BO), and UE RF measurements to allocate resources to a particular UE and the modulation coding scheme to be used.

With the introduction of advanced AI- and ML-based techniques, Project Marconi focused on utilizing advanced AI/ML techniques to improve the estimation of UE-perceived signal quality and its changes based on its forecasted mobility patterns.

The process to enable implementation of ML models that serve this purpose included:

1. Selecting the most relevant KPIs that help UE mobility learning and perceived channel quality.

2. Selecting applicable AI/ML/DL techniques keeping in mind the stringent inference time budget, which is a few milliseconds.

3. Continuing hyperparameter tuning of deep machine learning algorithms to ensure highly accurate UE channel quality estimates.

4. Profiling compute resource utilization for machine learning (ML)/ deep learning(DL) based inferences. The idea here is to ensure they can run as part of an existing RAN controller, with Intel Xeon processor, without need for additional hardware, such as FPGAs, hardware accelerators, or GPUs for deep-learning model inferences.

Project Marconi: Enhanced 5G Performance Using AI and ML

Project Marconi started by analyzing different functions performed by a MAC scheduler and its supporting algorithms used for provisioning QoS. Using this process, we studied about 200 MAC scheduler parameters and variables to identify a possible approach for introducing AI and ML to assist the MAC scheduler in enhancing spectral efficiency and improving QoS. We started with link adaptation, one of the most critical MAC scheduler functions. We studied its algorithms and identified key problem statements.

The primary job of gNodeB's MAC layer is scheduling the UEs and providing them optimum available radio resources, ensuring requested bandwidth is achieved. Link adaptation logic, which runs as part of the MAC scheduler, monitors the UE's radio conditions and the data transmission quality to estimate and appropriately adapt the scheduling parameters.

Link adaptation schemes adaptively adjust over-the-air transmission parameters in response to a change of radio channel for both downlink and uplink. The 3GPP 5G NR standard supports an adaptive modulation and coding (AMC) scheme for downlink and uplink transmissions. The serving gNodeB can adapt the MCS level based on the DL channel quality reports and HARQ feedback from the UE in the downlink. The transmit power of downlink control channels is adapted based on channel quality reports from the UE. The MIMO mode is adapted according to CQI reports from the UE while considering system parameters such as number of users, ACK/NACK, CQI variation, preferred MIMO feedback mode, etc.

In the uplink direction, the serving BS may adapt the MCS level based on the uplink channel quality estimation, allocated resource size, and the maximum transmission power of the UE. The transmit power of the uplink control channels (excluding the initial PRACH channel) is adapted according to power control commands.

A few of the critical challenges observed with the current approach for allocating optimum MCS include the following:

- Currently, the MAC scheduler utilizes an infinite impulse response (IIR) based approach to estimate the effective CQI and then apply adjustments to the current CQI based on the estimated and past CQI
- After which, CQI is also adjusted based on the current BLER level and the transmission rate. Both steps are based on the IIR, which are based on estimates that are not reasonable estimates of future CQI-based UE mobility patterns are not a good indicator of data transmission patterns
- These poor estimates sometimes lead to UE being allocated a higher MCS. This, in turn, creates a situation where the UE tries to send more data than can be transmitted, which leads to an increase in HARQ NACKs
- On the other hand, poor estimates sometimes lead to UE being allocated a lower MCS, which leads to UE sending less data than can be sent on the allocated radio resources. This leads to poor utilization of radio resources and degraded spectral efficiency

A Capgemini Engineering research study followed, scheduling parameters and KPIs to ascertain how ML can be applied to forecast UE mobility patterns. (See Figure 2.)

Crnti	cqi_widebandCQICodeWordOne
userLocationType	cqi_widebandCQICodeWordTwo
ueCategory	dlWidebandCQICodeWordTwo
ueQueueLoad	dlWidebandCQICodeWordOne_2
numLogicalChannels	dlWidebandCQICodeWordTwo_2
mcsPriority	cqi_currentCQI_WB_cw1
initialMcs	cqi_previousCQI_WB_cw1
assignedRBQueue Load	cqi_effectiveCQI_WB_cw1
userLocationType	cqi_currentCQI_WB_cw2
ueCategory	cqi_previousCQI_WB_cw2
mcsPriority	cqi_effectiveCQI_WB_cw2
cqi_cqiMode	cqi_currentCQI_SB_cw1
cqi_widebandCQI CodeWordOne	cqi_previousCQI_SB_cw1
cqiwidebandCQICodeWordTwo	cqi_effectiveCQI_SB_cw1
cqi_currentDLBLERForCW0	cqi_currentCQI_SB_cw2
cqi_currentCQI_SB_cw2	cqi_previousCQI_SB_cw2
cqi_dlBlerForCW0	cqi_effectiveCQI_SB_cw2
cqi_dlBlerForCW1	dlWidebandCQICodeWordTwo
sch_modulationSchemeFactor	dlWidebandCQICodeWordOne_2
dlWidebandCQICodeWordOne	dlWidebandCQICodeWordTwo_2
mcsPriority	cqi_currentCQI_WB_cw1
cqi_cqiMode	

Figure 2. Set of MAC scheduler parameters analyzed as part of the Capgemini Engineering study

Compiled by Capgemini Engineering

In this study, we analyzed various parameters in use as part of the MAC scheduling algorithm. We performed detailed statistical exploratory analysis by applying different statistical techniques, plotting respective line charts and even tried to generate a correlation matrix. We performed all this to identify a set of useful parameters for creating machine-learning models. The key objective for these models was to learn UE mobility patterns to better forecast UE's mobility to assist the MAC scheduler in efficient link adaptation.

Figures 3 through 10 capture the characteristics of a few of the KPIs and show their correlation. A few of the key parameters captured during the test run show that it isn't humanly possible to understand the variation of one parameter like CQI and its effect on HARQ ACK/NACK or even user experienced throughput. Hence, it is difficult to derive an empirical formula based on these MAC scheduler parameters that can help improve the estimation of UE mobility and its impact on perceived channel quality/CQI. For improving link adaptation, the key aspect considered here was UE learning mobility patterns based on trying to estimate and forecast the UE perceived channel quality at the time when UE traffic is scheduled.

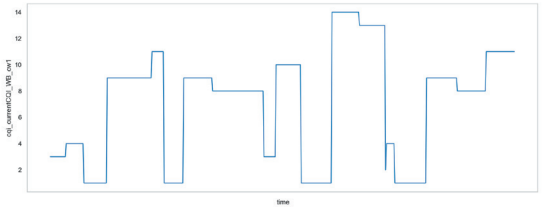


Figure 3. Current CQI WB CW1

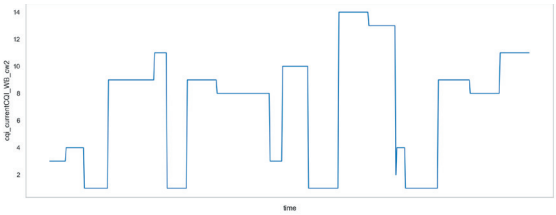


Figure 4. Current CQI WB CW2

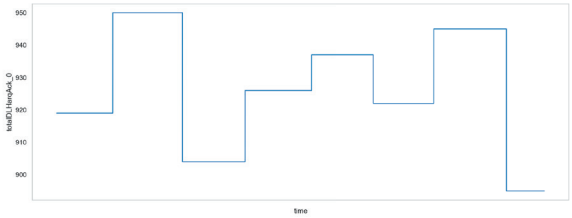


Figure 5. Total DL HARQ ARK

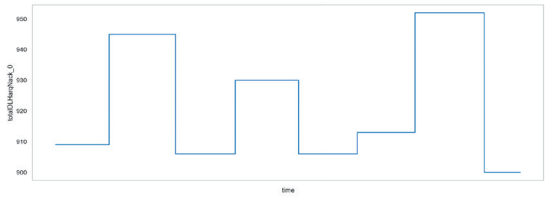


Figure 6. Total DL HARK NACK

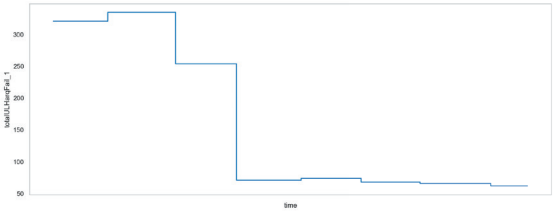


Figure 7. Total UL HARQ fail

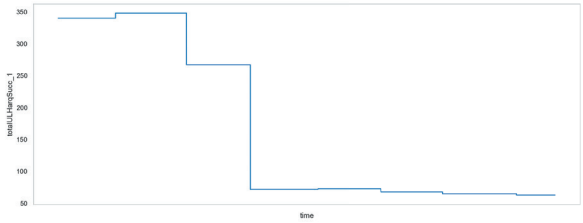


Figure 8. Total UL HARQ success

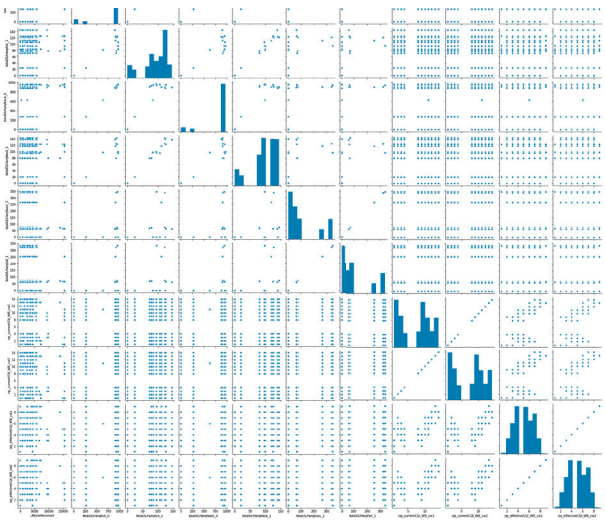


Figure 9. Feature correlation matrix

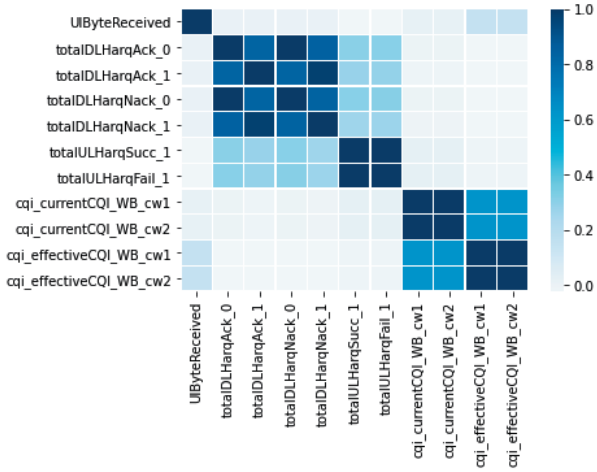


Figure 10. Feature correlation heatmap

Another key challenge for improving the current approach is the time available for scheduling decisions, which happens about every millisecond (msec). This means ML-based algorithms need to ensure they do not introduce any delays. The minutest delay here may severely negatively impact the overall QoS and the spectral efficiencies.

Last but not least, another critical factor to be considered is to ensure the ML-based updates should be capable of running on the existing gNodeB platform without introducing hardware accelerators or GPUs.

Subsequent sections cover key aspects of project Marconi to introduce optimized ML-based MAC scheduling that complies with all the objectives and constraints detailed above.

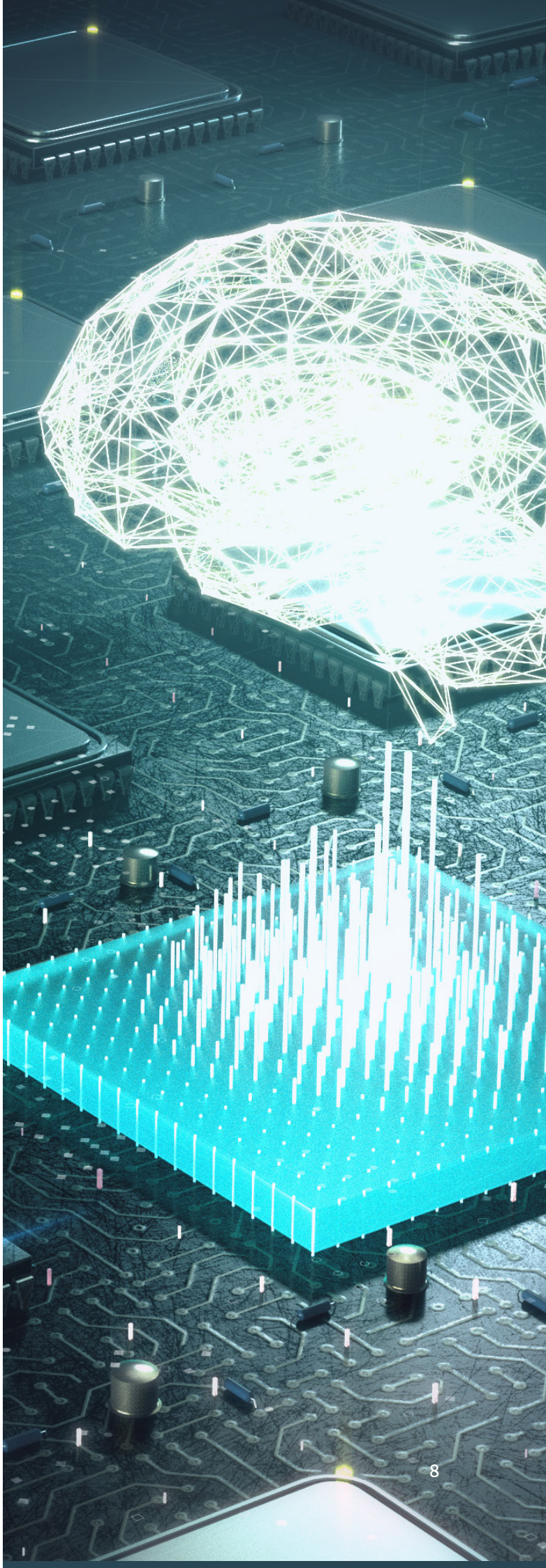
The ML-based approach

Capgemini Engineering and Intel collaborated on Project Marconi to develop an approach that introduces ML intelligence for the MAC scheduler. This section explores the challenges for selecting the appropriate ML techniques to be applied.

Key challenges

There is a variety of challenges for the ML-based approach. Here we focus on six.

- Currently, the g/eNodeB MAC telemetry interface does not implement a telemetry interface to share the MAC scheduler UE's specific measurements and scheduler data. A thorough study was done to define a telemetry interface to collect data from Layer 2 as no such interface or service models were available that covered such stats
- Time constraints for MAC scheduler operations were among the critical challenges for telemetry data collection and sending back the control actions
- Time constraints became a key challenge because of the critical time budget of defining deep learning (DL) models in under 10 msec. The goal was to ensure the DL model inference could run in one msec
- Introducing DL model inference code inside the g/eNodeB code delays MAC scheduler operations due to the synchronous DL prediction cycle
- Based on the above factors, we devised an asynchronous architecture where ML would work as a separate execution thread, and the g/eNodeB would consume ML output once it was available instead of waiting for it
- Resource consumption is a challenge because the stringent time constraints of both the g/eNodeB and the DL inference engine are required to run on the same machine to avoid any delay due to inter-process communication (IPC). Hence, both g/eNodeB and the ML inference engine will be competing for hardware resources like CPU threads and memory. Therefore, we need to ensure that the ML inference engine must not be greedy to cause the g/eNodeB to starve for resources. Also, the inference engine was restricted to work on a single CPU logic core along with the g/eNodeB processes



Model architecture

Many experiments were performed with different ML and DL algorithms. After multiple trials, the finalized model architecture included:

- 3D Tensor (N, Tx, F) from the g/eNodeB are inputs, where N is the number of UEs, Tx is past timesteps over which data is collected, and F is the features of the KPIs
- The model then produced 3D Tensor (N, Ty, F) data as output, where Ty is the future forecast for each UE. For example, a model accepting inputs as (8, 80, 2) and producing output (8, 20, 2) means that the model collects data from 8 UEs that last 80 timesteps and forecasts 20 future timesteps. These outputs are then weighed and merged into a single timestep and sent back to g/eNodeB as part of the control actions
- Training is performed by collecting data in the above format over a long duration to capture various mobility patterns of different UEs. The training aims to reduce the root mean square errors (RMSE) across all features

The hardware platform

We implemented training and inferencing of the deep learning models on 3rd gen Intel® Xeon CPU based server. With both platforms, we tested model inferencing performance on only one logical core (i.e., one thread) Intel® Xeon® Scalable processors.

All second-generation and later Intel Xeon Scalable processors feature Intel® Advanced Vector Extensions (Intel® AVX-512), and Intel® Deep Learning Boost (Intel® DL Boost), which can help accelerate machine-learning training and inference. Third-generation Intel® Xeon® Scalable processors feature enhanced Intel DL Boost with the industry's first x86 support of Brain Floating Point 16-bit (bfloat16) and Vector Neural Network Instructions (VNNI) bring enhanced AI inference and training performance. Depending on the AI workload, these technologies can deliver up to 1.93% more AI training performance and 1.87% more AI inference performance than the previous generation.

Intel FlexRAN

With the FlexRAN software reference architecture, Intel offers a blueprint to accelerate the development of vRAN and Open RAN solutions, helping equipment manufacturers and operators reduce time, effort, and cost. FlexRAN enables fast development of software-based LTE and 5GNR Base Stations that can be instantiated on any wireless network node from edge to core. The block diagram in Figure 11 shows the FlexRAN layer 1 (L1) PHY application, which takes radio signals from the RF front-end and provides real-time signal and physical layer processing on servers built with Intel® Xeon® Scalable processors. The Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction set is fully utilized to efficiently implement L1 signal processing tasks. The FlexRAN reference architecture is optimized for NFVi cloud-native deployments with the usage of DPDK-based networking using hardware capabilities of Intel® Ethernet Network Adapter 700/800 series. The FlexRAN reference architecture performs the entire 4G and 5G layer 3, 2, and 1 processing with help from dedicated Intel hardware accelerators for FEC (Intel® FPGA Programmable Acceleration Card N3000 or Intel® vRAN Accelerator ACC100 Adapter) as well as cryptographic accelerators (Intel® QuickAssist Technology), thereby making more processing power available to increase cell capacity and edge-based services and applications.

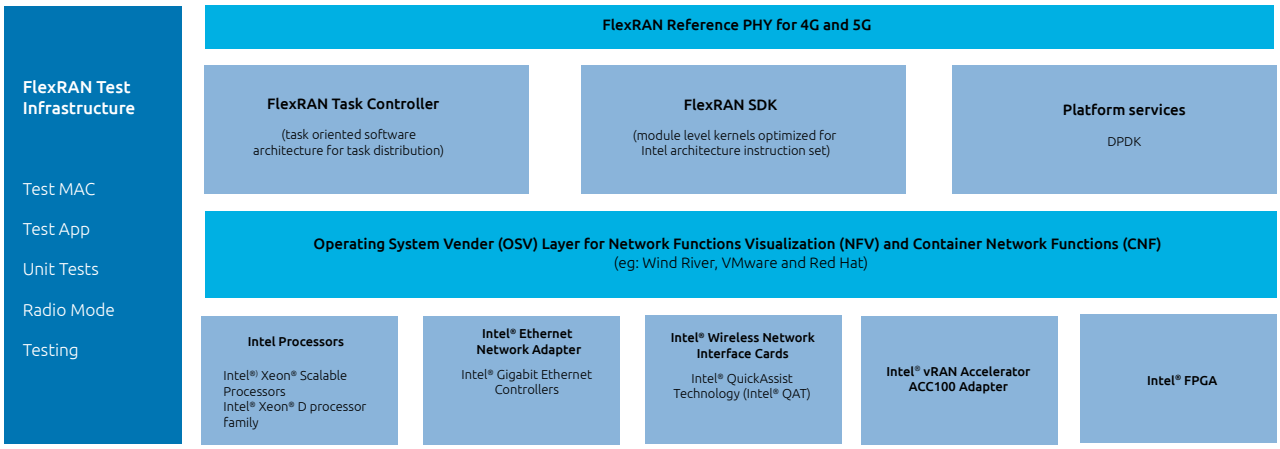


Figure 11. Intel FlexRAN Reference Architecture

Source: Intel

Intel hardware is helping improve performance for the ML models. Both 2nd Gen Intel Xeon Scalable processor and 3rd Gen Intel Xeon Scalable processor CPU platforms support Intel® Advanced Vector Extensions 512 (Intel® AVX-512), a new instruction set launched since the 2nd Gen Intel Xeon Scalable Processors. With ultra-wide 512-bit vector operations capabilities, Intel® AVX 512 can accelerate performance for demanding computational tasks such as AI and DL workloads. Furthermore, the 3rd Gen Intel Xeon Scalable processor server supports Vector Neural Network Instruction (VNNI), an extension of Intel® AVX 512 to accelerate AI/DL inference. Also, with faster cores and larger memory, 3rd Gen Intel Xeon Scalable processor provides even better performance than prior generations.

The ML software platform

NetAnticipate5G and the enhanced DL libraries offered by Intel were used as part of different machine learning experiments performed as part of this activity. The software stack for implementing the AI/DL models includes a number of open-source tools.

The NetAnticipate 5G

It provides all the essential ingredients starting with the telemetry data collection, data ingestion, exploratory analysis, model training and deployment pipelines. It utilizes AI/ML solutions from Intel mentioned in the following sections on model training and inference. This enables NetAnticipate to train complex DL models that are highly optimized for inference speed and accuracy on Intel® Xeon® CPU-based compute without the need for hardware accelerators such as GPU, FPGA, HDDL, etc.

Analytics Zoo

It is an open-source unified data analytics and AI platform developed by Intel. It simplifies scaling a range of AI models for distributed training or inference on a big data cluster, such as Apache Spark. It also provides optimized implementation libraries for the most popular deep learning frameworks, such as TensorFlow and PyTorch, for optimized runtime performance. Analytics Zoo provides a time series solution called Chronos, which has a built-in DL model for time series analysis and AutoML for automatic feature generation, model selection, and hyperparameter tuning for seamless scaling with integrated analytics and AI pipelines.

Optimized libraries

It includes the Intel® Math Kernel Library, Intel® Integrated Performance Primitives, and Intel® Data Analytics Acceleration Library. Other optimized libraries are available, such as the Intel® Machine Learning Scaling Library and the oneAPI Deep Neural Network Library (oneDNN).

Intel® oneAPI AI Analytics Toolkit (AI Kit)

For accelerating end-to-end machine-learning and data science pipelines includes several DL frameworks such as TensorFlow and PyTorch optimized for the Intel® architecture, and the Model Zoo for Intel® Architecture, and Intel® AI Quantization Tools for TensorFlow.

NetAnticipate 5G for a self-driving network

The Capgemini Engineering NetAnticipate5G is an award-winning autonomous network for realizing zero-human touch network operation. It analyzes many hidden and hierarchical influencers to enhance network performance, predict potential network anomalies, build autonomous decisions, and take preventive actions. An autonomous feedback loop ensures the network self-learns to improve actions it takes over time.

The NetAnticipate5G framework helps CSPs build a state-of-the-art self-learning network. (See Figure 12.) NetAnticipate5G uses the Lambda architecture for data processing that enables the network to self-learn through continuous feedback using closed-loop automation.

The framework uses a multi-level intent orchestrator to apply the identified intents back to the network in a simplistic way.

It combines the best of both worlds: batch processing and stream processing. This pattern consists of three layers:

1. **Batch layer:** Ingests and processes data on persistent storage such as HDFS and S3
2. **Speed layer:** Ingests and processes streaming data that has not been processed by the batch layer yet
3. **Serving layer:** Combines outputs from the batch and speed layers to present merged results

The batch layer is used for initial model training and model building from scratch. The speed layer is used for incremental updates of the model. A batch layer enables accurate predictions while the speed layer allows for real-time updating, which is critical to responsiveness.

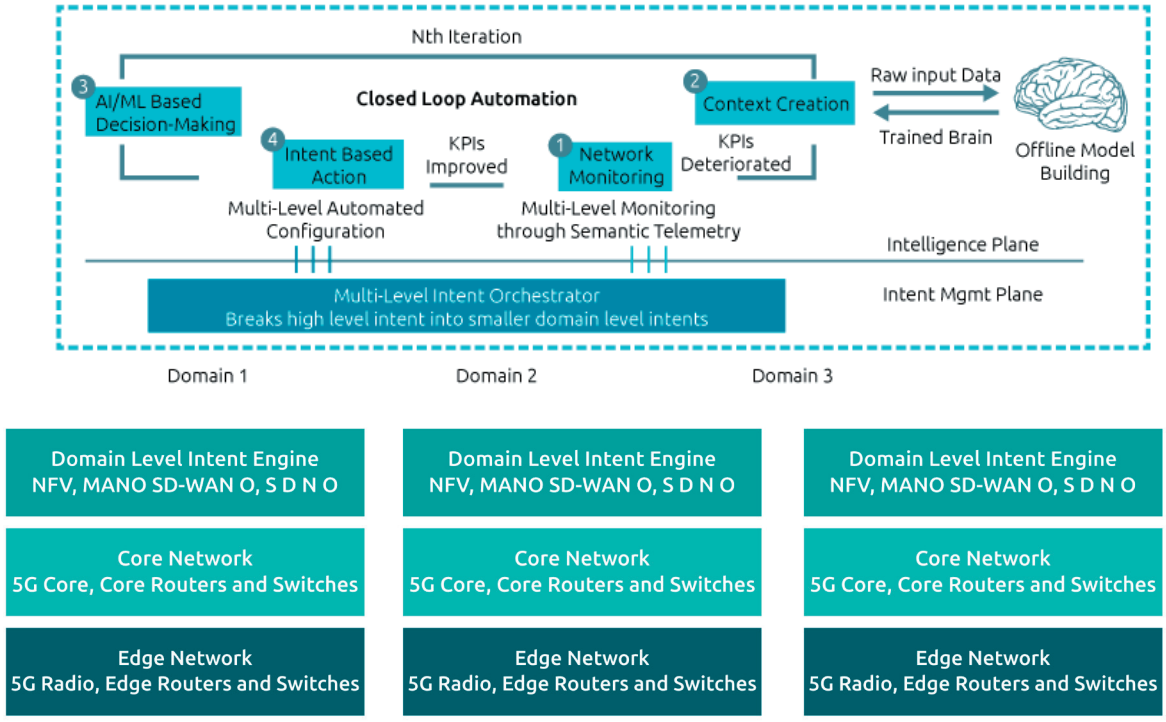


Figure 12. The NetAnticipate 5G SLN architecture

Source: Capgemini Engineering

Capgemini Engineering RATIO: An ORAN RIC platform

The Capgemini Engineering Open-RIC platform called RATIO supports a fully disaggregated RIC architecture, based on hardened ORAN SC components – currently the Cherry release. (See Figure 13.)

The NetAnticipate5G framework is fully aligned to the ORAN Alliance specification. It supports multi-vendor CU/DU through standard ORAN interfaces, which helps avoid vendor locking and allows an operator to choose different vendors for CU/DU and x/rAPPs.

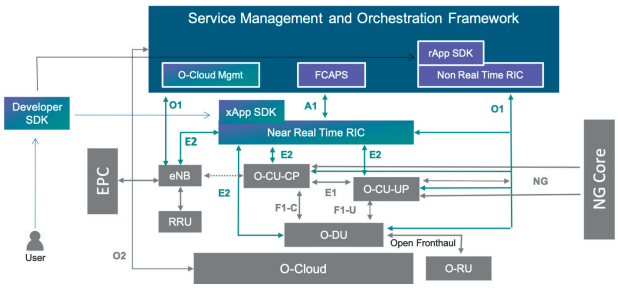


Figure 13. The Capgemini Engineering RATIO ORAN-based architecture

Source: Capgemini Engineering

CU/DU solution based on O-RAN architecture

- Capgemini 5G SA Solution running as disaggregated O-CU, O-DU running as part of virtualise cloud based infra
- Integration of Capgemini CU/DU with partner's Cloud Infra & RIC
- It is already integrated with VMware Infra
- Support for E2 interfaces in CU/DU and integrated with VMware RIC and partner's xApp

Non-RT RIC solution

Features of the Non-RT RIC solution includes:

- Based on hardened ORAN SC components and Capgemini Engineering's NetAnticipate5G framework, it is fully aligned to O-RAN Alliance specification
- NetAnticipate5G framework manages the entire lifecycle of rApps
- Provides conflict mitigation functionality
- Provides SDK for designing and deploying new rApps
- Supports IPSec for A1 interface confidentiality, a Capgemini Engineering AI/ML Solution

Capgemini's AI/ML solution is based on NetAnticipate5G framework and provides complete lifecycle management (LCM) of AI/ML that includes data preparation, training and inference.

- NetAnticipate5G provide complete MLOps AI/ML LCM for rApp and xApps.
- NetAnticipate5G supports various AI/ML deployment scenarios, including:
  - Training on SMO/Non-RT RIC and inference on Non-RT RIC
  - Training on SMO/Non-RT RIC and inference on Near-RT RIC
  - Training on SMO/Non-RT RIC and Inference on O-CUO-DU

Capgemini Engineering Near Realtime RIC

Salient features of the Capgemini Engineering near-RT RIC solution include:

- Based on hardened ORAN SC components and fully aligned to O-RAN Alliance E2AP, E2GAP, and E2SM specification
- NetAnticipate5G framework manages the entire lifecycle of xApps
- Provides SDK for designing and deploying new xApps
- xApp Runtime on NetAnticipate5G Framework

xApp Runtime on NetAnticipate5G Framework

xApp runtime environment uses the NetAnticipate5G Framework to provide an execution environment of xApps using ML models ML models can either be tightly coupled inside xApp or loosely coupled using REST APIs. In the latter case, Ml models are hosted in model serving.

Capgemini Engineering SMO

The Service Management and Orchestration (SMO) Framework is a lightweight ONAP-based platform that terminates the O1 interface from Near RT RIC. The O1/VES interface supports the performance and fault monitoring in SMO. The O1/NETCONF interface supports configuration management of the network elements in the O-RAN. VES subscription/un-subscription is also performed via NETCONF because VES itself does not provide such a function. Standard VES collector is used for rare events like FM and CM. A high-velocity VES collector (HV VES) is used for real-time event streaming needed for PM.

ML-infused link adaptation experiment details and results

Lab test environment

L1 bypass mode

In L1 bypass mode, both Layer 1 and UE are simulated using the L1&UE simulator. This setup helps create different call and mobility patterns for up to 256 simultaneous UEs. (See Figure 14.)

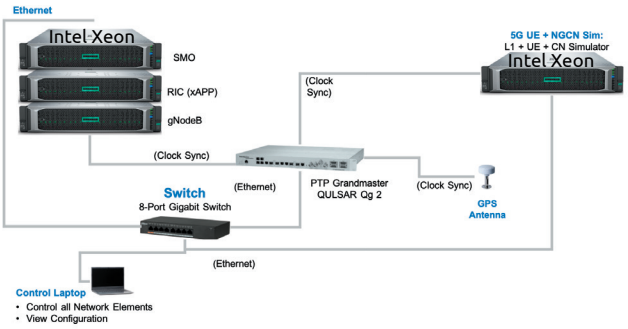


Figure 14. L1 Bypass mode lab setup

Source: Capgemini Engineering

End-to-end test setup

After running the tests in L2 bypass mode, the experiments were performed in the end-to-end environment with TM500 for setting up calls with the g/eNodeB-connected real RF interface. (See Figure 15.)

Model experiments

During the course of the research, the following different models were evaluated as part of the ML experiments:

- SimpleExpSmoothing (SES) and HoltWinter(HW)
- S/ARIMA
- Prophet – Univariate, does not recognizes time in msec
- GRU [Encoder - Decoder]
- LSTM [Encoder - Decoder]
- Temporal convolutional network

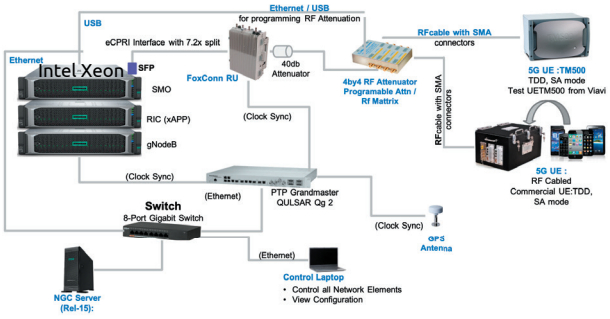


Figure 15. E2E lab setup

Source: Capgemini Engineering

During all the experiments, performance stats of both model accuracy and model inference time were recorded. Also, profiling stats were collected on the Intel® hardware mentioned above. Experiments were repeated with a different number of logical CPU threads as constraints. The execution time of ML model inference and resource consumption along with model RMSE were considered as primary stats.

Below are test results of AI performance improvement with Intel AI , resources consumption, and Spectral Efficiency improvement.

AI inference[1]

	Execution time (ms)
Baseline	1.08ms
Optimization with Intel AI*	0.64 ms (41% improvement)

*Intel optimized framework & OneAPI on 3rd Gen Intel® Xeon® Scalable Processors on one logical CPU core with hyperthreading switched - on

AI accuracy[1]

	RMSE(root mean square errors)
Baseline	2.14
Optimization with Intel AI*	0.97 (55% improvement)

Source: Capgemini Engineering

*Analytics Zoo AutoML on 3rd Gen Intel® Xeon® Scalable Processors

Resource consumption

CPU	Memory (MB)
1 LT (3rd Gen Intel® Xeon® Scalable Processors)	396

Spectral efficiency improvements

Link Adaptation stats	Spectral efficiency
With ML	15%

Source: Capgemini Engineering

Conclusion

By introducing ML-infused Link adaptation, we observed gains in the spectral efficiency on the order of 15% that resulted in cell throughput gain of 11.76%. Also, Project Marconi demonstrated that ML could be deployed for intelligent decision-making as part of RAN Layer 2 time-critical functions, on standard Intel Xeon processors, without introducing any

additional hardware requirement on CSPs.These are promising results for CSPs that could help them enhance the user service experience and reduce churn while improving radio resource utilization.

This instills confidence in the Project Marconi team to continue on the research and introduce ML to infuse intelligence and enhance the performance of multi-user MIMO and massive MIMO scheduling functions.

Our plan now is to showcase our initiatives through the O-RAN Alliance and work with the organization to share our work as xApp use cases.

Author

Subhash Chopra
Director - Technology
Capgemini Engineering

Contributors

Walid Negm, Subhankar Pal, Pankaj Rawat from **Capgemini Engineering**

Vitaliy Zakharchenko, Tong Zhang, Junwei Deng, Jane Wan, Shan Yu, Jason Dai, Hong Huisuk (Kevin) from **Intel**

References

1. Whitepaper, "Learn, Adapt and Protect with NetAnticipate, Self-Learning Network for the Zettabyte Era of Network Traffic," Capgemini NetAnticipate5G
   https://capgemini-engineering.com/as-content/uploads/sites/13/2020/03/learn-adapt-and-protect-with-netanticipate.pdf

2. Solution Brief, "Altran's NetAnticipate Accelerates Self-learning Networks, Harnessing the Power of Intel® AI," Intel
   https://builders.intel.com/docs/aibuilders/altrans-netanticipate-accelerates-self-learning-networks-harnessing-the-power-of-intel-ai.pdf

3. Mehta, NB Goldsmith, AJ, "Performance analysis of link adaptation in wireless data networks," Nov. 2020, IEEE Explore
   https://ieeexplore.ieee.org/document/891875

4. Karmakar, Raja, Chattopadhyay, Samiran, Chakraborty, Sandip, "Dynamic link adaptation for High Throughput wireless access networks," Feb. 25, 2015 IEEE Xplore
   https://ieeexplore.ieee.org/document/7413630

5. Ku, Gwanmo, MacLean Walsh, John "Resource Allocation and Link Adaptation in LTE and LTE Advanced: A Tutorial," Dec. 2014, Research Gare
   https://www.researchgate.net/publication/272490406_Resource_Allocation_and_Link_Adaptation_in_LTE_and_LTE_Advanced_A_Tutorial

6. "Intel-tensorflow/Tensorflow," Github
   https://github.com/Intel-tensorflow/tensorflow

7. "IntelAI/Models," Github
   https://github.com/IntelAI/models/

8. Brief, "3rd Gen Intel Xeon Scalable Processors,"
   https://www.intel.com/content/www/us/en/products/docs/processors/xeon/3rd-gen-xeon-scalable-processors-brief.html

9. "IntelAI/models," GitHub https://github.com/IntelAI/models/tree/master/datasets,
   https://github.com/IntelAI/models/tree/master/datasets
   https://github.com/Intel-tensorflow/tensorflow -b bf16/base , commit

10. "3rd Gen Intel® Xeon® Scalable Processors Brief," Intel
    https://www.intel.com/content/www/us/en/products/docs/processors/xeon/3rd-gen-xeon-scalable-processors-brief.html

Capgemini engineering

About Capgemini Engineering

Capgemini Engineering combines, under one brand, a unique set of strengths from across the Capgemini Group: the world leading engineering and R&D services of Altran – acquired by Capgemini in 2020 - and Capgemini's digital manufacturing expertise. With broad industry knowledge and cutting-edge technologies in digital and software,Capgemini Engineering supports the convergence of the physical and digital worlds. It helps its clients unleash the potential of R&D, a key component of accelerating their journey towards Intelligent Industry. Capgemini Engineering has more than 52,000 engineer and scientist team members in over 30 countries across sectors including aeronautics, space and defense, automotive, railway, communications, energy, life sciences, semiconductors,software & internet and consumer products

Learn more at:

www.capgemini-engineering.com

Write to us at:

engineering@capgemini.com

Notices & Disclaimers by Intel

[1] Baseline: 1-node, 2x Intel® Xeon® CPU E5-2620 v4 @ 2.10GHz on HPE DL380 Gen9 with 128 GB (16 slots/ 8GB/ 2400) total DDR4 memory, microcode 0xb000012, HT on, Turbo on, CentOS Linux release 7.8.2003, 3.10.0-862.el7.x86_64, 1x 40GB SSD, Temporal Convolution Networks , Python 3, Pytorch, ONNX runtime, AI performance tested using one Xeon core conducted by Capgemini Engineering on 12/24/2020. New: 1-node, 2x Intel® Xeon®) Platinum 8360Y @ 2.4GHz on Intel® Server System M50CYP2SB2U with 256 GB (16 slots/ 16GB/ 3200) total DDR4 memory, microcode 0x8d055260, HT on, Turbo on, Ubuntu 9.3.0; 17ubuntu1~20.04, 5.4.0-72-generic, Temporal Convolution Networks , Python 3, Pytorch, One API, ONNX runtime, AI performance tested using one Xeon core conducted by Capgemini Engineering on 04/28/2021.

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex. Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure. Your costs and results may vary. Intel technologies may require enabled hardware, software or service activation. Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy. © Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.