ICT-56-2020 "Next Generation Internet of Things"
Grant Agreement number: 957218

IntellioT

Deliverable D4.7
Dynamic network management (final version)

Deliverable release date	31/03/2023
Authors	1. SIEMENS: Andreas Zirkler, Arne Broering 2. EURECOM: Jérôme Härri 3. AAU: Beatriz Soret, Federico Chiariotti, Andreas Casparsen 4. UOULU: Sumudu Samarakoon, Dinesh Wadu 5. TTC: Martijn Rooker
Editor	Beatriz Soret (AAU)
Reviewer	Simon Mayer (HSG)
Approved by	PTC Members: (Vivek Kulkarni, Konstantinos Fysarakis, Sumudu Samarakoon, Beatriz Soret, Arne Bröring, Maren Lesche) PCC Members: (Vivek Kulkarni, Jérôme Härri, Beatriz Soret, Mehdi Bennis, Martijn Rooker, Sotiris Ioannidis, Anca Bucur, Georgios Spanoudakis, Simon Mayer, Filippo Leddi, Holger Burkhardt, Maren Lesche, Georgios Kochiadakis)
Status of the Document	Final
Version	1.0
Dissemination level	Public

Table of Contents

1 INTRODUCTION

1.1 Overview of IntellIoT framework

The IntellIoT project addresses the main technical challenges related to the support of intelligent edge Internet of Things (IoT) that deal with: (1) complex IoT applications which include functions for sensing, acting, reasoning, and control; (2) heterogeneous devices like edge computers and resource-constrained devices; (3) and big data from a huge number of data sources. In the IntellIoT communication architecture, the wired segment consists of the TSN (Time Sensitive Networking) controller and the TSN infrastructure used in a manufacturing plant to connect static industrial machinery. The rest of traffic in the network, static or mobile, is handled by the wireless segment, in a time-varying channel that introduces delays and errors. IntellIoT supports traffic with high data rates and Ultra Reliable Low-Latency Communication (URLLC), for which 5G (5th Generation) technologies are required[1].

This task has a three-fold **contribution** to the IntellIoT framework:

(C1) The **research activities** within the scope of wired and wireless resource allocation and configurations for intelligent IoT environments.

(C2) The implementation of the **communications resource manager** for the wireless segment, to be used in the UC demos at the end of the project. It consists of an xApp[2] integrated in the 5G infrastructure developed in Task 4.2.

(C3) The implementation of the **TSN controller** for the wired segment, to be used in the UC demos at the end of the project. The goal of the controller is to reserve resources on the TSN infrastructure developed in Task 4.1 and to cut-off malicious nodes.

This deliverable, being the final output of Task 4.3 ("Dynamic Network Management"), describes the research and implementation tasks related to the allocation and optimization of the communication resources, both wired and wireless. The first version of this deliverable, D4.3 "Dynamic Network Management (initial version)", provided the core communication models and research analysis to support the heterogeneity of traffic in IoT environments (C1). Moreover, a basic implementation of the communications resource manager (C2) and the TSN controller (C3) were described.

Since the infrastructure management is the basis of the IntellIoT framework, there is a strong interplay between Task 4.3 (and the content of D4.3) and several other Work Packages (WPs) and Tasks of the project: Task 4.3 receives valuable input from Task 2.1 ("Use cases specification & Open Call definition "), Task 2.2 ("Technology analysis & requirements specification "), and Task 2.3 ("Architecture specification & interoperability") concerning the project's use cases, the requirements, as well as the architectural design. In Task 2.3, the logical view of the IntellIoT infrastructure was defined (see Figure 1), with the two software components of Task 4.3 highlighted in the green box.

[1] 5G defines URLLC traffic with a target of up to 1 ms delay with a 99.999% of success probability
[2] xApps are applications built on the Radio Access Network (RAN) Intelligent Controller (RIC) that allow controlling and optimizing RAN functions and resources. For more details of the baseline 5G infrastructure the reader is referred to D4.2 ("5G Network Functionalities")

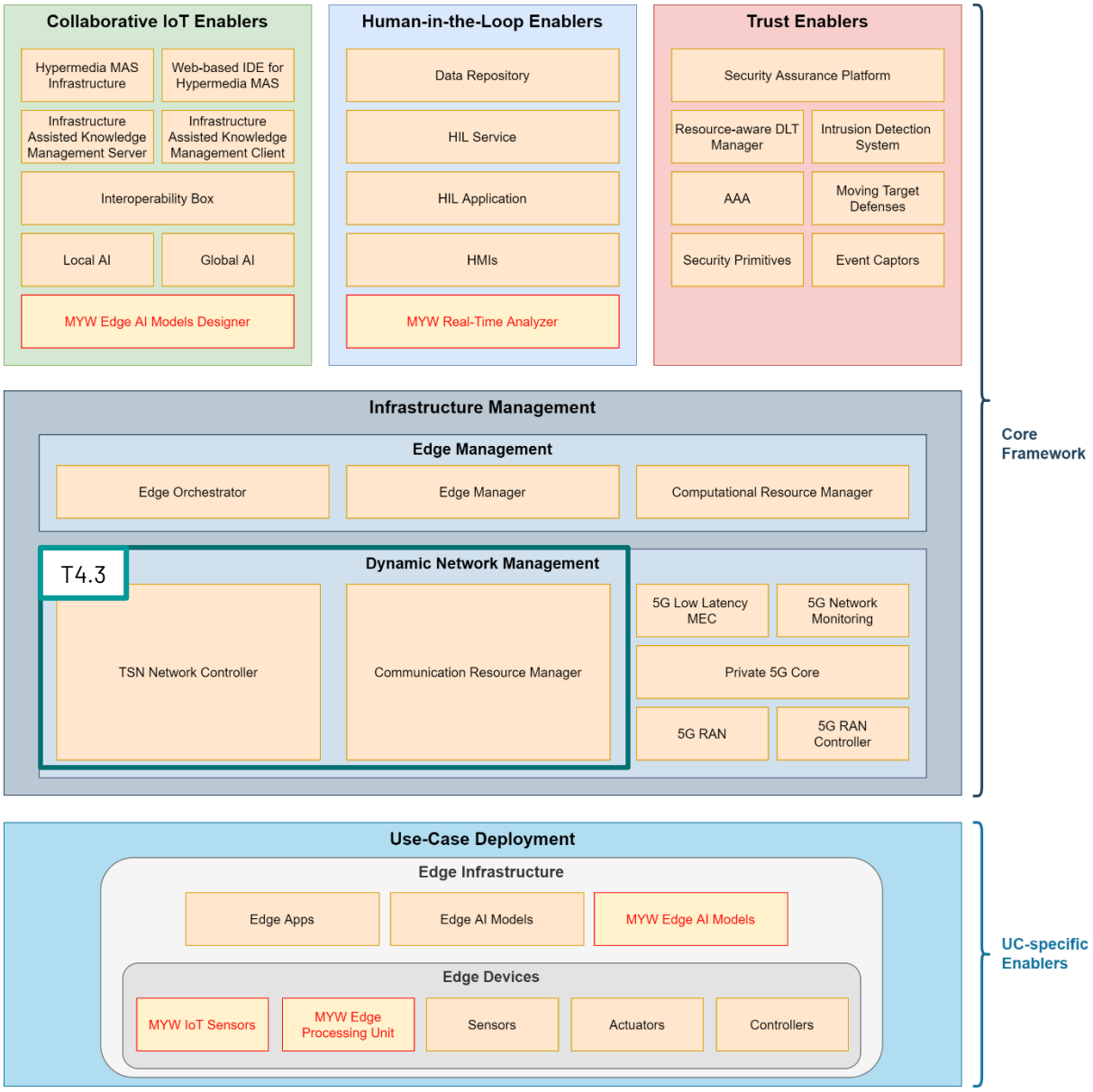


Figure 1 Logical view of IntellIoT infrastructure (D2.6)

The infrastructure management documented in this deliverable has been implemented in close consideration of the implementation of Task 4.2 "5G network functionalities", which provides the wireless infrastructure over which the wireless optimizations can be built upon. Moreover, there is a relation to Task 4.1 ("IoT/edge infrastructure management"), which allocates the computing resources, and the output of Task 4.4 "Trustworthy infrastructure by design", which provides the system trustworthiness. This task is also related to WP3, where the implementation of distributed and self-aware IoT applications with Human-In-the-Loop (HIL) relies on a flexible and reliable wired and wireless infrastructure. The components developed in this task (communications resource manager and TSN controller) are the input to the demonstration and evaluation tasks of WP5 for the remaining of the project.

1.2 Objectives and KPIs

This task contributes directly to the following IntellIoT objective:

Objective 2: Enable ultra-reliable low-latency communication over heterogeneous networks to enable tactile (real-time) and contextual (adaptive) interaction between IoT devices, humans, and services.

The fulfillment of this objective is verified through the following KPIs:

ID	KPI Description	Contributor	Description
2.5	Application-tailored definition and fulfilling of a reliability requirement for the three use cases, maintained under challenging network conditions and based on a data-driven prediction.	C1 + C2 (research + implementation)	C1: The research of timing and reliability, specifically for the most challenging VR/AR traffic, is described in Section 2.4., including a data-driven prediction to make an efficient use of the radio resources. C2: The communications resource manager includes functionality that supports the configuration of application-tailored requirements, e.g., by estimating the VR latency as explained in Section 2.5; further evaluations will be performed in WP5 until the end of the project and described in D5.6 Note: The 5G dynamic network management solutions are universal and therefore applicable to the three UCs and any other 5G traffic.

ID	Impact KPIs description	Contributor	Description
i11.1	Delivery of **open-source components for 5G** communication and dynamic network management supporting context-based and data-driven ultra-reliable low-latency communication for the NG IoT, as defined in **Obj. 2**.	C2 (implementation)	The Communications Resource Manager is an open source xApp (see Section 4) The functionality is completed, and the initial integration in WP5 is in progress. Further integration and testing are planned for the remaining of the project and will be part of D5.6.

The task contributes indirectly to the other objectives, which rely on reliable and adaptive communications.

Moreover, as part of the intermediate check of the IntellIoT project, the following challenge has been highlighted:

Collaborative IoT devices that execute **de-centralized Artificial Intelligence (AI)**-driven applications interacting with the HIL.

The communications infrastructure is an enabler for this challenge: it gives support to the data traffic generated by the system aiming at fulfilling the QoS (Quality of Service) requirements of said traffic. Specifically, IntellIoT proposes a collaborative paradigm where nodes use communication resources, whose use is optimized in this Task 4.3.

1.3 Outline

The research activities of the task are described in Sections 2, 3, 4, 5.1 and 5.2. The last subsection of Section 5, 5.3, describes the implementation of the communications resource manager for the wireless segment, to be used in the UC demos. Section 6 describes the implementation of the TSN controller. The deliverable is concluded in Section 7, where the next steps are identified.

1.4 Summary of modifications as compared to D4.3

As compared to D4.3, we emphasize the following updates:

- 1.2. Objectives and KPIs: updates of the status of KPIs and verification
- 2.4.3. AR/VR Data-driven analysis and optimization: new research models and results of data-driven optimization for AR/VR traffic
- 2.5. Implementation of the communications resource manager: description of the functionality, implementation, and performance of the communications resource manager
- 3.2. Selection of TSN features

2 WIRELESS RESOURCE MANAGEMENT FOR INTELLIOT ENVIRONMENTS

This section describes the research (C1) conducted on wireless resource management that gives the basis for the understanding of the models, tradeoffs and performance of a 5G network for the IoT, as well as the implementation of the communications resource manager (C2).

2.1 Timing and reliability in edge computing

The conventional approach to measure the timing performance in a wireless system is to consider the packet delay as the one and only metric to capture the latency requirements of a transmission. Thus, the **end-to-end latency** is obtained by adding the contribution of the different components to the latency budget and, if the total is above the target latency, then each component is optimized separately. In IoT scenarios such as the ones addressed by IntellIoT, it is meaningful to consider the joint computing-communication system design and use, beside the end-to-end latency, as both metrics reflect the freshness of the information at the receiver. This is motivated by the fact that numerous IoT applications like the ones considered in IntellIoT rely on the system transmitting real-time status updates of a process from a generating point to a remote destination [22]. Sensor networks, vehicular networks, and industrial control are examples of this kind of update process. One example in IntellIoT is the manufacturing plant in UC3 where sensors and actuators are connected with a 5G network, or the sensors in the tractor whose information is transmitted to the human when he/she has to remotely control it (UC1). Another example in the agriculture domain (as an extension of the current scope of UC1) is to consider a fleet of self-driving tractors sending periodic status updates like in a Vehicle to Vehicle (V2V) network.

For all these cases, the so-called **Age of Information (AoI)** represents the timeliness requirements by quantifying the freshness of the information at the receiver [23]. Specifically, the AoI computes the time elapsed since the latest received update was generated at any given moment in time, i.e., how old is the last packet received by the destination. Using AoI to measure the performance of the system can capture exactly the timespan between the process being measured and the user's view. Furthermore, AoI is often complementary to latency, as it requires a balance between update frequency and congestion, while latency always benefits from a reduction in traffic, i.e., fewer updates being sent.

An example of the AoI dynamics is plotted in Figure 2: the AoI grows linearly over time, then decreases instantly when a new update arrives.

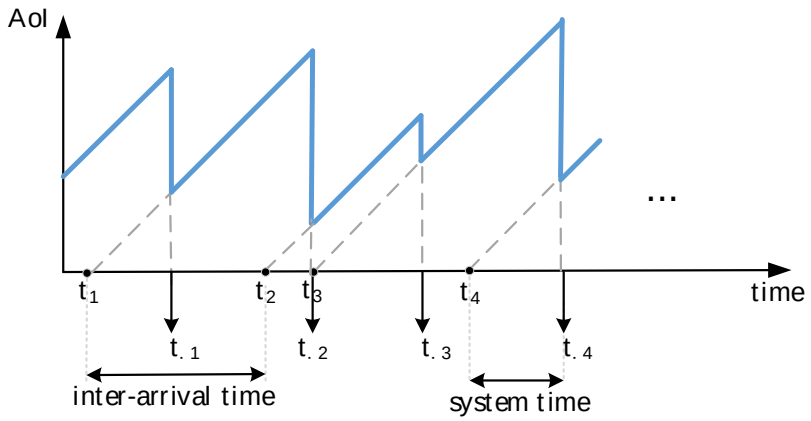


Figure 2. AoI dynamics. The packets are generated in t_1, t_2, ... and received at the destination at t'_1, t'_2, ... The inter-arrival time is the time between the generation of two packets at the source. System time is the sum of the queueing time and the service time.

Another age-related metric is the **peak AoI (PAoI)**, which is the maximum value of AoI for each update, i.e., how old the last packet was when the next one is received by the destination. As in other performance metrics of communication systems, the PAoI is more informative than the average age when the interest is in worst-case analysis. Edge computing in general is a good approach for information-age-sensitive IoT applications, as the transmission of sensor readings to a centralized cloud may require too much time and increases uncertainty, while processing data closer to the sensor that generated them can reduce the communication latency and thereby the overall AoI at monitoring or control process [24].

We chose therefore **AoI** and **PAoI** as two relevant timing metrics for the project. As the IntellIoT scenarios require high reliability, analyzing the *average* age (average AoI or average PAoI) is not enough, however: The tail of its distribution is also a very important parameter, as it directly affects the risk of control system failures like the tractor in UC1, the robot arm in UC3 and the doctor notifications in UC2. All of these must be bounded to low levels for a good system performance.

A more involved example of an age-sensitive application in IntellIoT is the HIL, which involves a loop communication with the transmission of Virtual Reality (VR) video from the tractor or the robot arm to the human, the human's reaction to this input data, and the control feedback to the tractor or the robot arm. Here, we need to go one step further and consider the **Age of Loop (AoL)**, to capture the loop timing relations. In [8], this metric is defined as the time elapsed since the piece of information causing the latest action or state (depending on the selected time origin) was generated. The AoL is then used by the authors to learn the control requirements of a Wireless Network Control System (WNCS) and to optimize the network resources. Depending on which direction is considered first to define the loop, Uplink (UL)[3] or Downlink (DL), we can define a UL AoL or a DL AoL, and we can observe the average and/or peak values. For example, these family of new metrics can be used to design a power allocation policy for the UL by observing the current UL AoL status. Likewise, we can define a modulation coding scheme for the DL transmissions by observing the DL AoL. [9] uses a classical inverted pendulum system model, which is a widely used benchmark problem in the control and Reinforcement Learning (RL) domain; however, the same principles can be applied to the IntellIoT HIL loop.

Figure 3 shows the behavior of the AoL in these scenarios, where the slave is the tractor in UC1 and the robotic arm in UC3.

[3] UL in the mentioned reference [8] is from the sensors to the controller and DL from the controller to the sensors

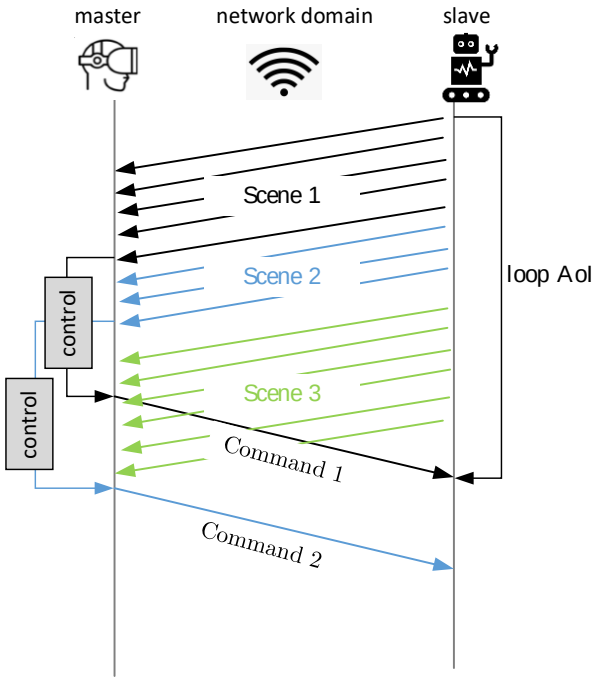


Figure 3. Example of HIL traffic and the AoL metric

The second set of metrics are suitable when the throughput and reliability of the communication must be observed. One of these metrics is the **Effective Capacity (EC),** which has been widely used to understand the tradeoff among reliability, throughput, and latency in buffer-aided communication systems. Specifically, the EC of a wireless system with a time-varying channel expresses the maximum achievable throughput under some QoS requirements. Mathematically, the EC of a channel process is:

$$EC(\theta) = \lim_{n\to\infty} \frac{1}{n\theta} \log E\left[e^{\theta R[n]}\right]$$

where θ is the QoS exponent and R[n] is the accumulated channel rate until time n, i.e., $R[n] = \sum_{m=0}^{n-1} r[m]$, and r[m] is the instantaneous channel rate. The physical time is divided into units referred to as symbol periods. A usual transmission strategy for broadband communications is to have the channel rate r[m] adapt to the channel variations and fulfill a given reliability requirement.

The QoS exponent θ captures the statistical QoS guarantees given in the form of a duplet (Dt; ε), where Dt is the target delay and ε is the probability of exceeding Dt. A small value of θ corresponds to a system that can only provide a looser QoS guarantee, while a larger θ leads to more stringent QoS requirements. Therefore, the EC of the channel decays with the QoS exponent, since the maximum supported arrival rate decreases as the QoS requirements become more stringent. Specifically, the relation of θ with the duplet (Dt; ε) is given by:

$$\varepsilon \leq \Pr(D_u > D^t) = \Pr(q > 0)\, exp(-\theta EC(-\theta) D^t)$$

Where $\Pr(q > 0)$ is the probability of not having an empty queue.

The EC is relevant for broadband and other high-rate traffic supported by buffering and with latency and/or reliability requirements. For example, the transmission of VR/Augmented Reality (AR) traffic involves the transmission of real-

time high-rate video and control commands with very high reliability. Other examples of critical traffic within IntellIoT are the exchange of ML/A models in decentralized learning and messages within the trustworthiness pillar. In this project we use the EC in the theoretical comparison of orthogonal and non-orthogonal strategies, as explained in Section 2.2.

Finally, since the definition of URLLC as one of the connectivity types of 5G NR, there has been a lot of focus on the reliability-latency performance. The well-known 10^{-5} reliability requirement of this kind of traffic is defined as the probability that a data of size D (usually small) is successfully transferred within a time period T (usually 1 ms). For example, a general URLLC reliability requirement for one transmission of a packet is $1\text{-}10^{-5}$ for 32 bytes with a user plane latency of 1ms. The generalization of this definition is the **reliability-latency curve** depicted in Figure 4, which gives the probability of communication interface i achieving a latency deadline l, i.e.,

$$F_i(l) = Pr[L \leq l|i]$$

The probability of error of such interface i (i.e., the probability of outage) is given by

$$P_e^i = 1 - F_i(l), \quad l \to \infty$$

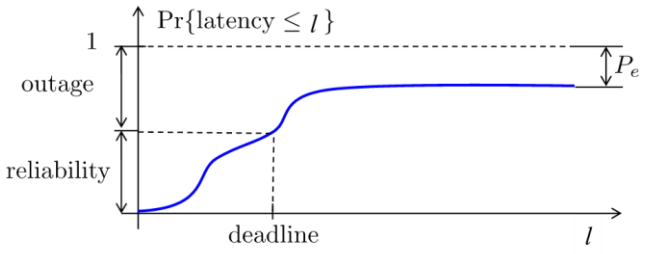


Figure 4 Reliability-latency curve

Timing in edge computing: a queueing analysis

Edge computing is gaining traction in several sectors such as telehealth, agriculture, manufacturing plants and robotics. Edge-driven services require strict control performance guarantees, which are only possible by carefully designing the communication system and the timing and reliability tradeoffs between control and communication. We aim at characterizing the timing and reliability tradeoffs in edge computing, with the interplay between the actual communication of the necessary data and the computation at the end. Limiting the AoI is a critical requirement which can influence the choice between local and edge-based computation [9] for IoT nodes. The problem becomes even more complex when considering multiple sources and different packet generation behaviors, along with limited communication capabilities [10].

We model the network as a tandem queue with 2-nodes, i.e., two queueing systems connected in series, where the first system represents the communication link and the second one represents the computing-enabled edge node and its task queue. Figure 5 shows an example where the communication buffer at the sensor and the task queue on the edge computing-enabled BS (Base Station) are the two queues in the tandem.

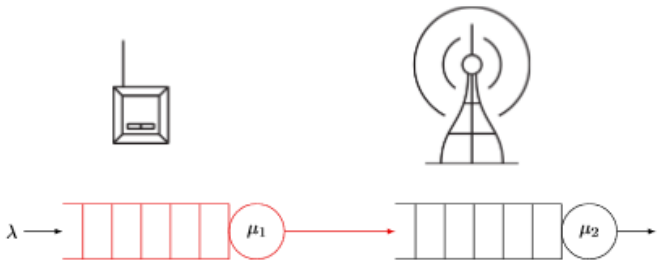


Figure 5. An example of the IoT edge computing use case: the sensor transmits data to a computing-enabled edge node, which needs to maintain information freshness

If the load on the computing-enabled edge node is time-constant, while communication is less predictable due to, e.g., dynamic channel variations and random access, then the M/M/1 - M/D/1 tandem queue is an appropriate model. An M/M/1 queue is a proper model for the channel if we assume an ALOHA system [11] with perfect Multi-Packet Reception, in which [13] the packets are not lost due to collisions and can only be lost due to channel errors. On the other hand, computation time is often modeled as a linear function of the data size in the literature [12], and updates of the same size would have a constant and deterministic service time; this situation is well-represented by an M/D/1 queue. Another option is to consider the computing load as also variable over time, leading to stochastic computing time, and represent the two systems by M/M/1 queues with different service rates [14].

An example of the AoI dynamics in edge computing is plotted in Figure 6: as in Figure 2, the AoI grows linearly over time, then decreases instantly when a new update arrives. Naturally, the age at the destination is never lower than the age at the intermediate node, as each packet that reaches the destination has already passed through the intermediate node, but the dynamics between the two are not trivial and serve as a motivation for this work. We analyze the distribution of the PAoI in a tandem queue with two systems with independent service times and a single source, where each infinite queue follows the First Come First Serve (FCFS) policy. We consider both the M/M/1 -- M/D/1 tandem and the M/M/1 -- M/M/1 case, covering common communication relaying and communication and computation scenarios.

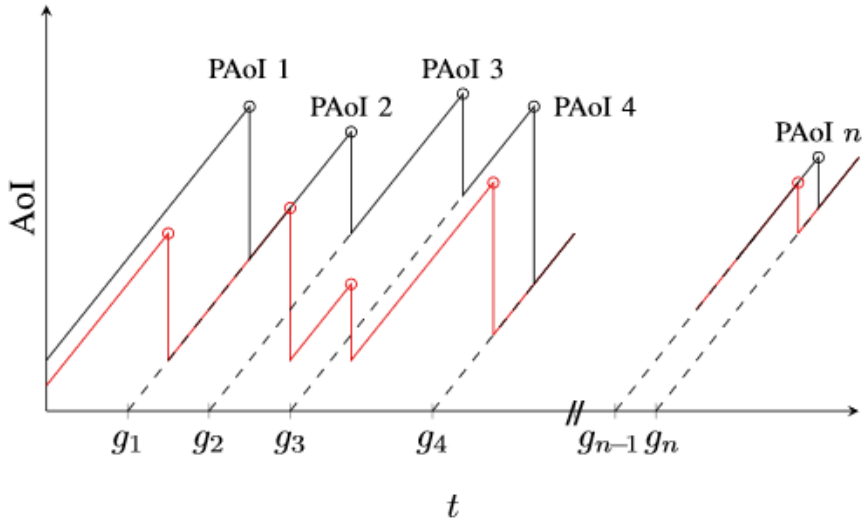


Figure 6. The time-evolution of the AoI in a 2-node tandem queue. The age at the destination is plotted in black. The age at the intermediate node is plotted in red. Circles indicate the transmission of a packet in each node

In our two-systems tandem queue, packets are generated at the first system by a Poisson process with rate λ and enter the first queue, whose service time is exponentially distributed with rate μ_1. When a packet exits the first system, it enters the second one, whose service time is a constant D or an exponential random variable with rate μ_2. Notice that we use conventional queueing terms: in a single queueing system, the queueing time is the time spent in the queue, and the service time refers to the time spent in the server, which is our model corresponds to communication time. The sum of both is usually referred to as the total system time. In the considered tandem queue, whose diagram is shown in Figure 6, the notation is extended as follows. Both queues are of infinite size and are oblivious to the content of the packets: There is no preemption of the updates, i.e. an older packet is not removed from the queue when a new update comes from the same source. We assume that the service times in the two systems are independent. The assumption is realistic for edge computing systems, as the communication and processing are usually independent, but not always correct in relay networks; it therefore needs a careful examination. Even if the service times are independent, however, the waiting times are not, as the queue at the second system depends on the output of the first one.

Using standard queueing terminology, the packet generation times in the tandem queue correspond to the arrival times at the first queue, whereas the receiving instants are the departure times in the second queue. We define the total system time for packet i as T_i: When a packet is received, the AoI is equal to $T_i = r_i - g_i$, i.e., the difference between the time r_i when it is received by the destination and the time g_i when it was generated. The PAoI (see Figure 6) is the maximum value of the AoI, i.e., the age at the instant immediately before the arrival of a new update. If we denote the interarrival (or inter-generation) time $Y_i = g_i - g_{i-1}$, then PAoI is given by

$$\Delta_i = r_i - g_{i-1} = r_i - g_i + g_i - g_{i-1} = T_i + Y_i$$

For each system j=1,2 and by abuse of notation, we further define the following values for each of the systems in the tandem, which are necessary for the analysis. The system time $T_{i,j}$ is defined as the sum of the waiting time $W_{i,j}$ and the service time $S_{i,j}$. $Y_{i,j}$ is the interarrival time at system j. For j=1, we have $Y_{i,1} = Y_i$, while for j=2 we have to consider the generation time at the first queue and the time spent in system 1, i.e., :

$$Y_{i,2} = g_i + T_{i,1} - (g_{i-1} + T_{i-1,1}) = Y_i + T_{i,1} - T_{i-1,1}.$$

Since the first queue is M/M/1, the system times for the two queues are independent, as proven by Reich [15] using Burke's theorem [16] provided each system is in steady state for packet i-1.

However, the values of Y_i and T_i are correlated, and the computation of the PAoI needs to account for this fact. The strategy for the analyses is to define an extended waiting time $\Omega_{i,j}$ as the difference between the previous packet's system time and the interarrival time at the system, i.e., $\Omega_{i,j} = T_{i-1,j} - Y_{i,j}$.

The reason we named $\Omega_{i,j}$ the *extended* waiting time is that $W_{i,j} = [\Omega_{i,j}]^+$, where $[x]^+$ is equal to x if it is positive and 0 if x is negative.

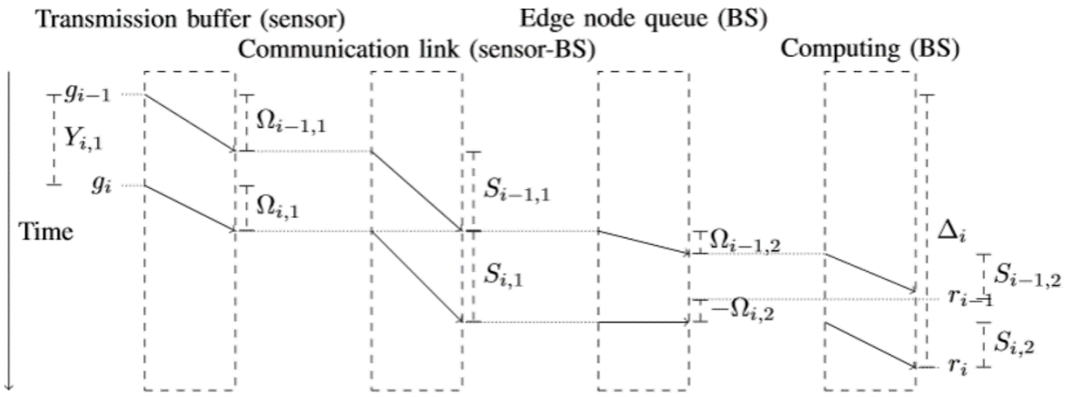


Figure 7. Schematic of the four steps a packet goes through in a tandem queue, highlighting the components of the PAoI

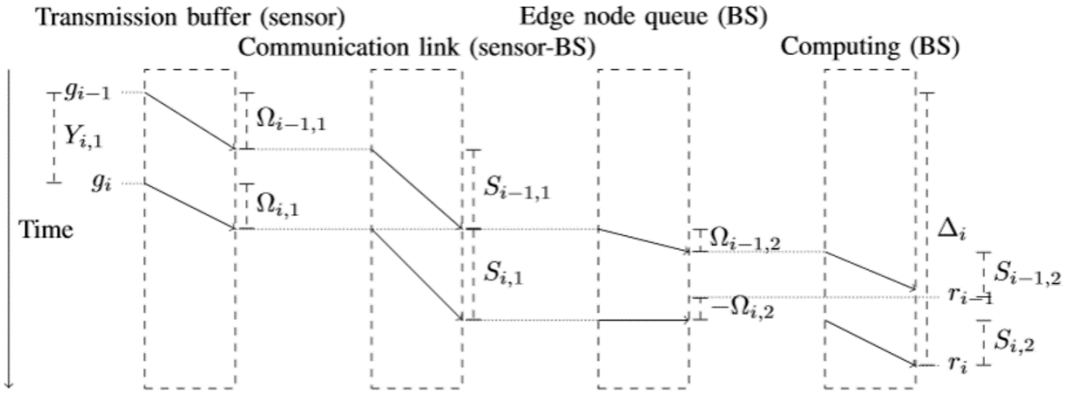


Figure 7 shows a possible realization of a packet's path through the tandem queue, highlighting the meaning of the extended waiting time: in the first system, in which packet i is queued, it corresponds to the waiting time, while in the second, in which the packet is not queued and enters service immediately, its negative value corresponds to the time between the departure of packet i-1 from the second system and the arrival of packet i at the same system. When W=0, we have a negative extended waiting time, as packet i arrives after packet i-1 leaves the system. In general, we have $\Omega_{i,2}=T_{i-2,2}-Y_{i,2}$ and the system time for packet i-1 is $T_{i-2,2}=W_{i-1,2}+D$, while we know the interarrival time $Y_{i,2}=S_{i,1}+[-\Omega_{i,1}]^+$.

Using the definition of the extended waiting time, the system time and PAoI distribution are calculated for a tandem M/M/1-M/D/1 and for a tandem M/M/1-M/M/1. The details of the model and the analysis can be found in [1]. In this deliverable, we skip the details of the analysis and show the most relevant numerical evaluations, optimizations, and conclusions.

First, Figure 8 shows the tail of the PAoI distribution, expressed using percentiles, as a function of the arrival rate λ packets per unit of time, for an M/M/1-M/D/1 tandem. The plot clearly shows that, to optimize the reliability of the communication and computation, the frequency of updates must be tuned carefully to balance between interarrival times and latency: Too frequent updates will clog the queues, while rare updates will not be sufficient to track the system. The effect of the service time D (i.e., serving at a rate $\mu = 1/D$ packets per unit of time) in the second system is shown in Figure 9: If we optimize the arrival rate, it is naturally better to have a faster computation, but this has

diminishing returns if D<1. In this case, computation can become faster and faster, but the real bottleneck of the system becomes the M/M/1 network connection. Improving only one of the two elements of the tandem does not maximize the benefit to the QoS, expressed as the high percentiles of the PAoI, as the other acts as a bottleneck and blocks any improvements. Notice that following classical queueing theory, both the arrival and service rates are expressed in packets per unit of time.

Interestingly, the considerations on the arrival rate are a bit different if we consider the average PAoI instead of the higher percentiles, as shown in Figure 8: In general, the optimal value of λ increases, as a higher update frequency is often better on average, although the possibility of longer queues makes the tail of the distribution longer.

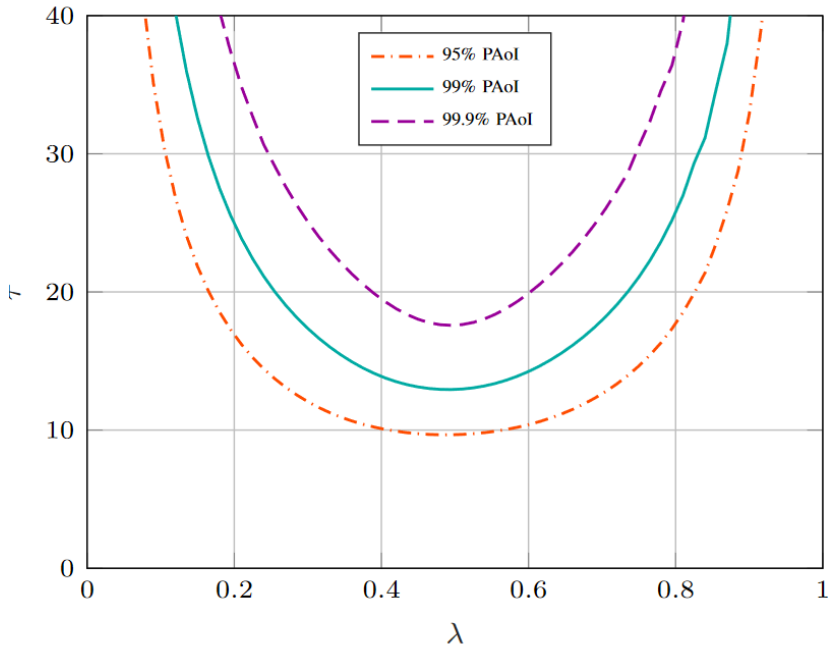


Figure 8. PAoI percentiles versus the arrival rate in the M/M/1-M/D/1 system

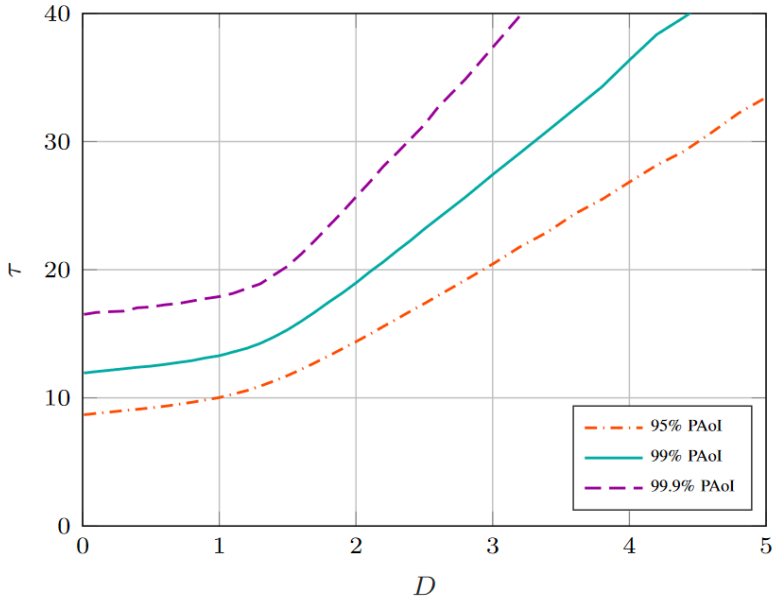


Figure 9. PAoI percentiles versus the system time in the M/M/1-M/D/1 system

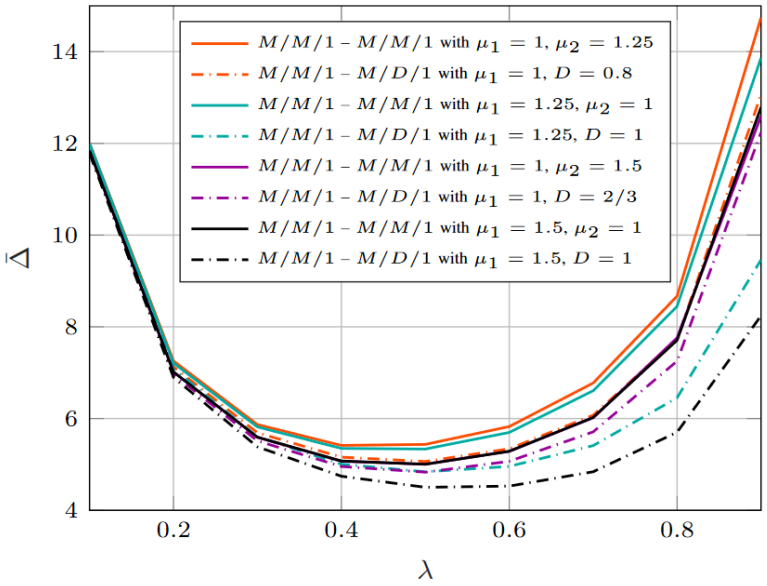


Figure 10. Cumulative Distribution Function (CDF) of the PAoI for the M/M/1-M/M/1 tandem for different values of the arrival rate

Table 1 summarizes the conclusions of this analysis.

Timing in edge computing
The study of PAoI is particularly important in edge computing systems, which are becoming ubiquitous in the IoT due to the timing requirements
Modeling the system as a tandem queue, it is possible to examine the tradeoffs in the allocation of computational and communication resources, as well as to determine the appropriate update frequency to optimize the average or worst-case PAoI
Control systems can rely on these freshness guarantees, enabling reliable remote applications in critical domains such as the UC1 Autonomous Guided Vehicle (AGV) trajectory control and the UC3 automated manufacturing.

Table 1. Key takeaways of timing in edge computing

2.2 Network slicing in 5G NR

In the wireless segment, IntellIoT is designed to support diverse traffic types that includes VR/AR video transmission, tactile feedback, IoT traffic from edge devices and edge infrastructure, Federated Learning (FL) traffic, Distributed Ledger Technology (DLT) transactions, etc., using the UL and the DL, like shown in Figure 11[4].

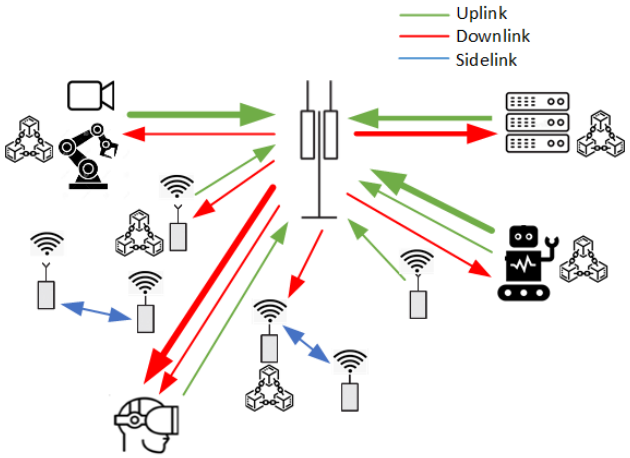


Figure 11. Example of heterogeneous UL, DL and SL traffic in an IoT environment

Network slicing is a key feature of 5G for the support of heterogeneous services and requirements, in the form of a virtualized technology framework [17]. Significant progress has been made in Third Generation Partnership Project (3GPP) regarding the core network and the functional aspects with, e.g., the definition of the network slice identifiers, and procedures and functions for slice selection. In the RAN, the conventional approach to slice is to allocate orthogonal radio resources to Enhanced Mobile BroadBand (eMBB), with very long payloads; Massive Machine Type Communication (mMTC), characterized by the large number of devices and the need of a random access mechanism to establish a connection; and URLLC, with small packets and low latency requirements [5]. This is directly applicable

[4] The Figure shows also the sidelinks (SL) for direct device-to-device communications, although this is not going to be showcased in IntellIoT

to IntellIoT UC1 and UC3 when HIL traffic is required: The communications resource manager will allocate two slices, one for the broadband VR/AR video and one for the tactile feedback and other control information. Besides, a default network slice can carry the rest of the traffic. In UC2, the traffic generated by FL is in principle more homogeneous, but the same principles can be applied to ensure the scalability of the solutions and future variations of the UC.

In the radio slicing, one of the key decisions is the multiple access scheme and specifically the choice between Non-Orthogonal Multiple Access (NOMA) and Orthogonal Multiple Access (OMA). In the conventional OMA approach, the communication resources for a given user/service are reserved at the beginning and cannot be shared with other users. This ensures the QoS but it leads to a waste of resources that is especially noticeable when the traffic is intermittent like in many IoT scenarios. In NOMA, by contrast, multiple users can be served simultaneously which allows enhanced spectral efficiency. The boundary of achievable rate pairs (in the case of two users) using NOMA is outside the capacity region achievable with OMA. This superiority is attained at the expense of increased complexity, specifically through the use of superposition coding at the transmitter and of Successive Interference Cancellation (SIC) at the receiver.

The principle of a SIC receiver is to subtract repeatedly the user-decoded signals from the composed multi-user signal to reduce the interference, until the signal of interest is decoded. The simplest case is with two users, such that the whole set of users is grouped in pairs according to the signal strength level, where one is the stronger (S) and the other one is the weaker (W). In uplink NOMA networks, S' signal is decoded first. In downlink NOMA, the inverse order is applied. This is illustrated in Figure 12.

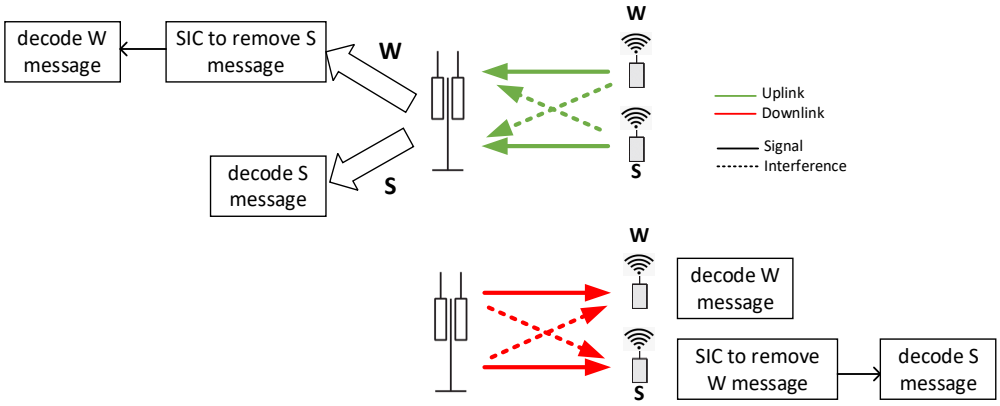


Figure 12. Downlink and uplink NOMA for two users: strong (S) and weak (W)

As introduced in Section 2.1, the heterogeneous and statistical QoS requirements of services such as VR/AR and some kinds of IoT traffic can be well-captured by the EC metric. In the following we see one example of application and evaluation of this metric in 5G networks. EC has been applied in many scenarios, including NOMA and OMA [18][19][20]. However, the typical approach is to assume a constant arrival process, neglecting the impact of the burstiness of the source. The first goal of this research is therefore to evaluate the performance of NOMA and OMA with time-varying traffic sources and channels that model the wireless channel and the IntellIoT data sources, respectively. Specifically, we evaluate constant traffic (that models the broadband VR/AR traffic) and Poisson- and Bernoulli-distributed traffic (to model different intermittent IoT traffic sources). We focus on the UL from, e.g., the edge devices and the cameras at the robot or the tractor to the BS.

System model: We consider a system where U single-antenna users transmit packets in the uplink to a single BS. The users are classified based on the channel conditions. The complex channel coefficient is denoted by $h_u(t)$, and $\alpha_u(t)$ is the power coefficient of user u. For the NOMA model, the users are sorted such that $|h_1(t)|^2 \leq |h_2(t)|^2 \leq |h_3(t)|^2 \leq \ldots \leq |h_U(t)|^2$. Moreover, the U users are grouped into N pairs. We assume that each pair has a non-overlapping subcarrier and there

is therefore no co-pair interference. Without loss of generality, we focus the study on one of the pairs, where s is the index of a center-cell user ("strong" user) and w the index of its paired cell-edge user ("weak" user).

Let $x_u(t)$ denote the symbol with zero mean and unit power sent by user u at time slot t. The superimposed signal received at the BS is:

$$y_u(t) = \Sigma_{u \in \{s,w\}} (P_u)^{½} h_u(t) x_u(t) + n(t)$$

where P_u is the uplink transmit power and n(t) Additive White Gaussian Noise (AWGN) of zero mean and noise power sigma. γ_u is the Signal to Noise Ratio (SNR) of user u.

Each user is modeled with a buffer system, where the buffer is ruled by an FCFS policy, the arrival process A represents the traffic source, and the service process R represents the time-varying channel transmission. Time is discrete. $a_u(t)$ and $r_u(t)$ denote the instantaneous traffic arrival and service rate, respectively, and A[n] and R[n] are the accumulated arrival and channel rate: $A[n] = \sum_{m=0}^{n-1} a[m];\ R[n] = \sum_{m=0}^{n-1} r[m]$.

The channel between the nodes and BS is block-fading, i.e., the fading is constant during each fading block, but it changes independently from one fading block to another. We assume Rayleigh fading with unit variance and channel conditions are mutually independent for all users.

We do the analysis under the assumption that the arrival and service processes are ergodic and the queue q is not congested and converges to a steady state. The delay violation probability is then written as a function of the EC, as introduced in Section 2.1, or the EB, specifically:

$$\varepsilon \leq \Pr(D_u > D^t) = \Pr(q > 0)\, exp(-\theta EB(\theta) D^t) = \Pr(q > 0)\, exp(-\theta EC(-\theta) D^t)$$

Where EB(θ) is the Effective Bandwidth (EB) function of the source and the rest of parameters where introduced in Section 2.1. The evaluation requires therefore evaluating the EB and EC functions for the traffic process and rate process, respectively, and then finding the crossing point of the two curves [21].

Traffic process: The following traffic models are considered:

1. Constant traffic. This is the usual case in the literature where the constant traffic source has a rate λ and the EB is:

   $$EB(\theta) = \lambda$$

   This simplifies the general equation and yields:

   $$\varepsilon \leq \Pr(D_u > D^t) = \Pr(q > 0)\, exp(-\theta^* \lambda D^t)$$

   Where θ^* denotes the crossing point of the EB and EC curves, which is the QoS exponent when both traffic and channel rate are a time-varying process.

   This deterministic model is a good initial model for the VR/AR traffic with a CBR encoder, although as we showed in *Figure 19* we need more advanced models to capture the variability even when a CBR encoder is used.

2. Poisson traffic. For Poisson traffic of rate λ, the EB function is

   $$EB(\theta) = \lambda \frac{e^{\theta} - 1}{\theta}$$

3. Bernouilli traffic. An intermittent user can modelled like a Bernoulli process, where a packet of size A_k/p arrives with probability p. The EB function of this process is given by

$$EB(\theta) = \lambda \frac{1}{\theta} \log\left(pexp\left(\frac{\theta Ak}{p}\right) + 1 - p\right)$$

4. Independent and Identically Distributed traffic. We can also consider a generic traffic source. If the process is i.i.d., then we can apply the Central Limit Theorem and write the EB of a Gaussian process in terms of the mean m and the variance σ, i.e.,

$$EB(\theta) = m + \frac{\sigma^2 \theta}{2}$$

5. Correlated traffic. The expression of the i.i.d. traffic can also be applied to correlated sources if the correlation decays sufficiently fast with time. Under these conditions, the cumulative arrival can be divided into blocks, such that inter-block correlation is negligible and the variance capturing the intra-block correlation [21].

Channel process: For the block-fading channel process, we also apply the central limit theorem and write the resulting EC of a Gaussian random variable in terms of the first two moments of the rate [21].

QoS exponent: The crossing point of the EB and EC function gives the QoS exponent. I.e., θ^* is the solution to: EB(θ) + EC(-θ) = 0

Numerical evaluations: Figure 13 shows the maximum supported queue delay violation probability versus the arrival rate of different traffic sources, for the strong and the weak users. The solid vertical line is the Shannon bound with no delay constraints. When statistical delay guarantees are imposed, the supported arrival rate must be reduced for the system to be able to meet the delay requirements. We observe the negative impact of a time-varying source for which the arrival rate is reduced for a fixed violation probability.

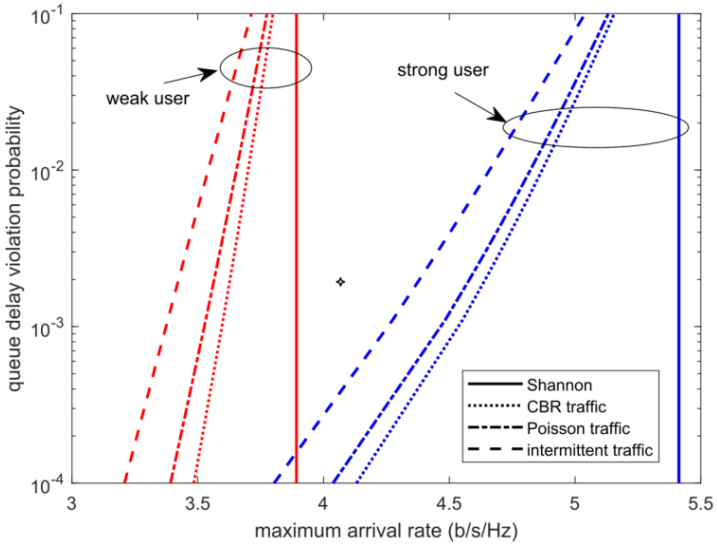


Figure 13. NOMA: queueing delay violation probability vs. maximum arrival rate. The frame duration is 1 ms and the target delay is set to 3 ms. For the intermittent user, the probability of generating a packet is p=0.5. The power coefficients of the strong and weak user are, respectively, 0.8 and 0.2

2.3 Radio Resource Management for Machine Learning applications

The next research component of this Section is related to the use of Machine Learning (ML) for radio resource management.

ML has become one of the key elements in automated system design due to its capability of discovering spurious correlations among the system interactions from large amounts of recorded data. In IoT systems, the data sources are many distributed edge devices connecting with one another mostly over wireless links. With the limited availability of radio resources that are possibly utilized for capacity-hungry and/or ultra-reliable low latent communication services that are independent from ML, there exists a challenge of accessing the data that are required for the learning. In this view, communication-efficient distributed learning methods are envisioned to offer learning at the wireless edge opposed to the classical centralized ML solutions. Among the most popular edge ML model training is Federated Learning (FL), in which the goal is to train a high-quality ML model in a decentralized manner, based on local model training and client-server communication [25] The design of FL relies on wireless resources for the communication between edge device (client) and the Parameter Server (PS) and on computational resources at clients for local training, which calls for a resource management solution for learning systems in the IntellIoT scenarios.

2.3.1 COMMUNICATION RESOURCE CONSTRAINTS

For the general strategy in which FL uses all clients in each FL cycle, the communication between the PS and clients is crucial. However, under limited communication resources, it is vital to utilize them over a subset of client devices that can upload their local models to the PS to enrich the learning process. Let $s_k(t) \in \{0,1\}$ be an indicator where $s_k(t) = 1$ indicates that the client k is scheduled by the server for uplink communication at time t and $s_k(t) = 0$ otherwise. To schedule several clients simultaneously, one Resource Block (RB) from a set $\mathcal{B}$ of B RBs is allocated to each scheduled client. Hence, we define the RB allocation vector $\lambda_k(t) = \left[\lambda_{k,b}(t)\right]_{b\in\mathcal{B}}$ for client k with $\lambda_{k,b}(t) = 1$ when RB b is allocated to client k at time t, and $\lambda_{k,b}(t) = 0$ otherwise. The client scheduling and RB allocation are constrained as follows:

$$s_k(t) \leq \mathbf{1}^{\dagger}\boldsymbol{\lambda}_k(t) \leq 1 \quad \forall k, t.$$

Here, $1^{\dagger}$ refers to the transpose of all one vector. In this view, the rate at which the k-th client communicates with the PS at time t is $r_k(t) = \sum_{b\in\mathcal{B}} \lambda_{k,b}(t) \log_2\left(1 + \gamma_{k,b}(t)\right)$ with the Signal to Interference and Noise Ratio (SINR) of $\gamma_{k,b}(t)$.

A successful communication between a scheduled client and the PS is defined by satisfying a target minimum rate. In this view, following the definition of the rate and the rightmost inequality of the above constraint, the communication requirement is imposed per RB allocation in terms of a target SINR γ_0 as follows:

$$\lambda_{k,b}(t) \leq \mathbb{I}\left(\gamma_{k,b}(t) \geq \gamma_0\right) \qquad \forall k, b, t.$$

Since the maximum number of clients that can be scheduled is bounded by the number of available RBs, the left side inequality of the initial communication constraint can be recast as

$$1^{\dagger}s(t) \leq B.$$

Satisfying the aforementioned constraints prevents stragglers introduced by the communication limitations, which reduces the FL training latency.

When frequent measurement of Channel State Information (CSI) is constrained, i.e., under imperfect CSI, it is vital to predict CSI accurately prior to RB allocation. In this case, the tools from Gaussian Process Regression (GPR), a Bayesian inference technique, can be utilized to jointly sample channels and predict CSI by exploiting the temporal dynamics of wireless links [26]. Here, GPR is used to predict the CSI between client k and PS over RB b using its past N samples as

well as to provide its uncertainty, which is accounted as the additional information $j_{k,b}(t)$ required for accurate future predictions. In this view, scheduling clients to maximize $\sum_k j_k^{\dagger}(t)\lambda_k(t)$ enables joint sampling and acquiring CSI information to improve channel prediction [26]. Here, $j_k^{\dagger}(t) = \left[j_{k,b}(t)\right]_{b\in\mathcal{B}}$.

2.3.2 COMPUTING RESOURCE CONSTRAINTS

The dynamic power consumption P_c of a processor at a client with clock frequency of ω is proportional to the product of $V^2\omega$, where V is the supply voltage of the computing that is approximately linearly proportional to ω [27]. Due to other tasks (sensing, monitoring, actuating) handled by the clients concurrently in addition to local model training, the available computing power P_c at a client fluctuates over time. In this view, given the local computation iterations (typically, measured in terms of Stochastic Gradient Descent (SGD) steps) M_k over the local dataset of size D_k, the minimum computational time required for client k is given by

$$\tau_{\mathrm{c},k}(t) = \frac{\mu_{\mathrm{c}} D_k M_k}{\sqrt[3]{P_{\mathrm{c}}^k(t)}/b}$$

Here, the constants μ_c and b are the number of clock cycles required to process a single sample and power utilized per computation cycle, respectively. To avoid unnecessary delays in the overall training that are caused by clients' local computations, we impose a constraint on the computational time, with threshold τ_0 over the scheduled clients as follows:

$$s_k(t) \leq \mathbb{I}\left(\tau_{\mathrm{c},k}(t) \leq \tau_0\right).$$

This constraint prevents clients acting as computation stragglers during the FL.

2.3.3 CLIENT SCHEDULING TOWARDS HIGH FL ACCURACY

Classical FL assumes ideal computation and communication at the clients (referred to as IDEAL hereinafter). For a given global dataset $\mathcal{D} = \cup_k \mathcal{D}_k$ aggregated over local datasets $\{\mathcal{D}_k\}$ and a predefined number of iterations T, by optimizing the model parameters $\boldsymbol{w}$ IDEAL method results in a loss of $F^*(\boldsymbol{w}(T), \mathcal{D})$. Compared to a centralized training design over large number of training iterations that yields the minimal loss of F_0, the IDEAL method results in a loss of accuracy $\varepsilon^*(T) = F^*(\boldsymbol{w}(T), \mathcal{D}) - F_0$. It is worth highlighting that as $T \to \infty$, the condition $\varepsilon^*(T) \to 0$ is held.

Under the limited computation and communication resources (in both perfect and imperfect CSI), any type of client scheduling method results in a loss $\varepsilon(T) \geq \varepsilon^*(T)$, i.e., bounded by the performance of IDEAL scenario. Hence, it is essential to devise a joint client scheduling and RB allocation solution for FL under the limited resources that minimizes $F(\boldsymbol{w}(T), \mathcal{D})$ (or $\varepsilon(T)$). Under imperfect CSI, the FL loss needs to be minimized while improving the acquisition of CSI information via maximizing $\sum_k \boldsymbol{j}_k^{\dagger}(t)\boldsymbol{\lambda}_k(t)$, where the FL (under limited resources) focuses on minimizing the following objective function [x]:

$$\text{minimize}_{\boldsymbol{w}(t),\boldsymbol{s}(t),[\boldsymbol{\lambda}_k(t)]_k \forall t}\ F(\boldsymbol{w}(T), \mathcal{D}) - \frac{\varphi}{T}\sum_{k,t} \boldsymbol{j}_k^{\dagger}(t)\boldsymbol{\lambda}_k(t),$$

where $\varphi(\geq 0)$ is a tradeoff parameter controlling the CSI exploration and FL loss minimization. Solving above objective under the communication and computation constraints is carried out with the aid of stochastic optimization tools including Luapunov optimization and linear programming [28]. The solution concept of joint client scheduling and RB allocation is illustrated in Figure 14.

The client scheduling solutions under three different conditions are presented as proposed methods in Table 2. Under a simulated environment, the performance of the proposed methods is compared with two state-of-the-art scheduling algorithms as well as the ideal setup (no constraints on communication and computation resources) that are listed under Table 2.

Table 2. Proposed algorithms, baselines and benchmark algorithm

	Model	Description
Proposed methods	QAW-GRP	PS uses dataset sizes to prioritize clients and GPR-based CSI predictions for scheduling.
	QAW	PS uses pilot-based CSI estimation and dataset sizes to prioritize clients.
	QUNAW	PS uses pilot-based CSI estimation without accounting dataset size of clients.
Baselines	RANDOM	Clients are scheduled randomly.
	PF	Clients are scheduled with fairness in terms of successful model uploading.
Ideal setup	IDEAL	All clients are scheduled assuming no communication or computation constraints.

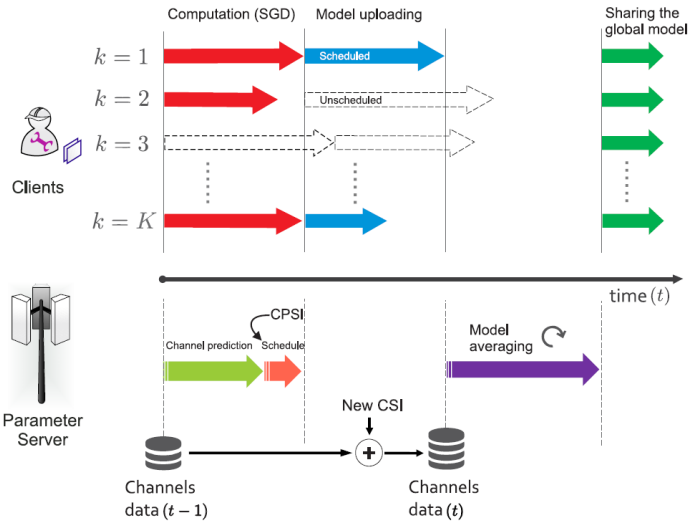


Figure 14. Joint client scheduling and RB allocation in FL training

For the performance comparison, we carry out FL training over $T = 100$ iterations. The loss in FL compared to centralized training model loss $\varepsilon(T)$ under different scheduling policies are shown in Figure 15 for fixed number of clients $K = 10$ (left) and fixed number of RBs $B = 5$ (right). Without computing and communication constraints, IDEAL shows the lowest loss while RANDOM and PF exhibit the highest losses due to the communication resource limitations-agnostic client scheduling. In contrast, the resource limitations-aware proposed methods QAW-GPR, QAW and QUNAW exhibit close to IDEAL performance. It is worth highlighting that the gap between proposed methods and baselines diminishes as the availability of radio resources increases in the system.

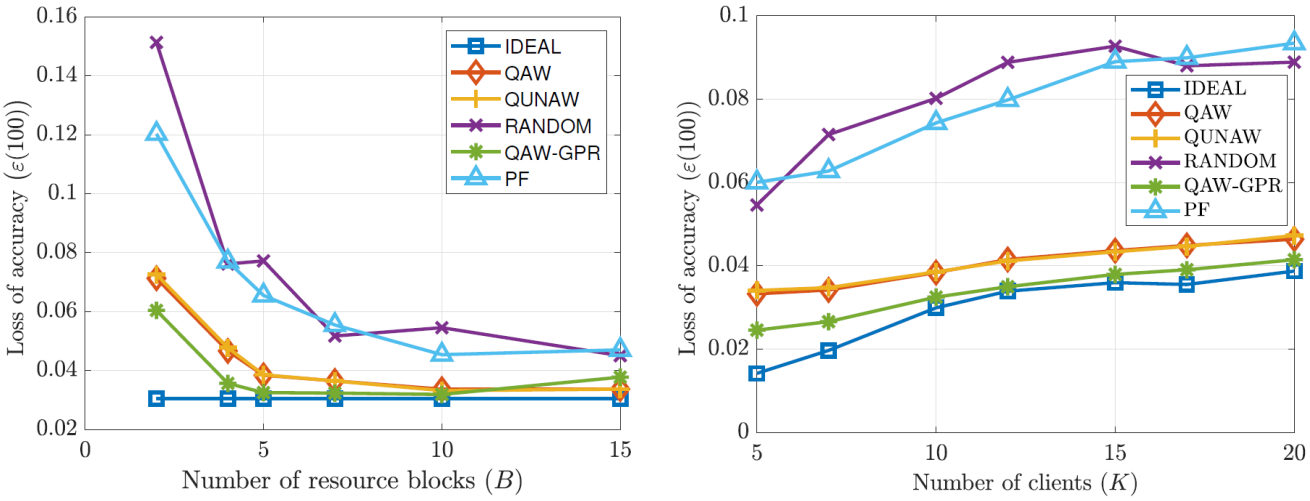


Figure 15. Comparison of Loss in FL under different scheduling policies with fixed number of clients (left) and fixed number of RBs (right)

The impact of local SGD iterations under limited computing power availability is analysed in Figure 16. Due to the assumption of unlimited computing power availability, IDEAL performs well with the increasing local SGD iterations M. Similarly, gradual reductions in the loss of accuracy for both QAW and PF can be seen as Ming from two to eight. However, further increasing M results in longer delays for some clients in local computing under limited processing power. Such computation stragglers do not contribute to the training in both PF (drops out due to the computation constraint) and QAW (not scheduled). However, the awareness of computation resources in QAW prevents the scheduling of stragglers, which yields better performance than PF.

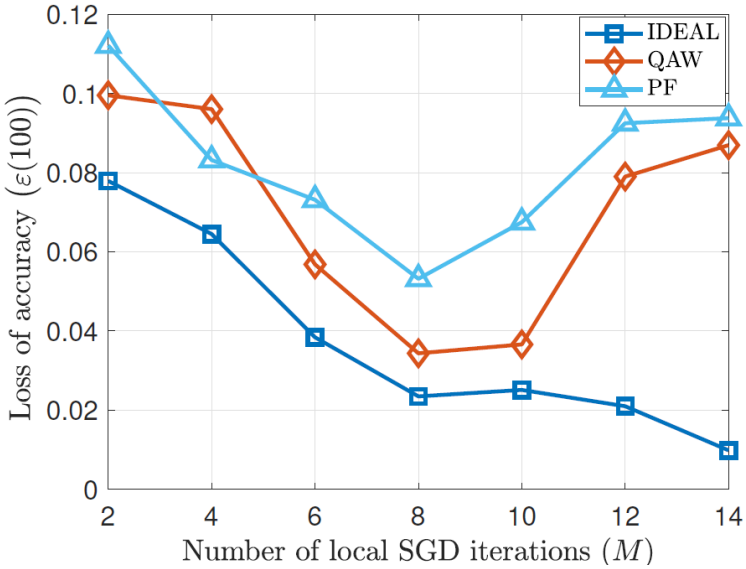


Figure 16. Impact of computation requirements on the loss of accuracy

To summarize, the conclusion of the study of Radio Resource Management (RRM) for learning are listed in Table 3.

RRM for learning
As the communication and computation resources become tighter in IoT networks, the resource and ML performance-agnostic scheduling policies yield poor learning.
RRM for learning becomes crucial for large-scale IoT networks with limited resources.
For a given target training time, the proposed client scheduling for FL ensures minimum loss over a centrally trained model yielding a performance close to the FL setup under unlimited resources.

Table 3. Key takeaways of RRM for learning

2.4 Radio Resource Management for human-in-the-loop communications

The final research component of this section is specific of the most challenging traffic in IntellIoT: the scenario with the human-in-the-loop.

2.4.1 HIL SYSTEM MODEL

We consider a HIL system, in which a human interacts with a partially automated system by controlling one or more of the devices. We can consider the use of automated tractors in agriculture as an example: if one of the tractors is in a situation it cannot handle automatically, its control will be transferred to a human operator, who will then receive the tractor's camera feed on a VR headset. As explained in Section 2, maintaining the information fresh is a key component to ensure safe and efficient operation: if the feed received by the human controller is old, either because of communication delay or of an insufficient update frequency, the operation of the tractor might be compromised. In this initial research we consider the VR traffic in the DL and only ACKs with a deterministic latency in the UL, and therefore we focus on the AoI set of metrics and not the AoL.

Naturally, the heaviest task, both in terms of communication and computation, is the transmission of the VR frames from the tractor to the user: frames are generated every few tens of milliseconds, with common values of the inter-frame time being between 16 ms (60 Frames Per Second (FPS)) and 33 ms (30 FPS).

Single link example:

Using a single link to deliver these frames can be insufficient for two main reasons:

- A temporary blockage, which can cause the loss of one or more frames, or a drop in the link's capacity, which can cause frames to be delayed and queued, can significantly increase the age;
- The capacity of the link even under ideal conditions might not be sufficient to deliver the frame on time, leading to longer and longer queues and, consequently, longer waiting times.

Taking the simplest case of VR traffic modeled as constant and transmitted in a buffer-aided system whose transmission time is exponentially distributed (D/M/1 queue), we can plot the reference results of latency and PAoI distribution. These evaluations are shown in Figure 17 and Figure 18, respectively, and they serve to illustrate the fundamental differences between the two metrics: while the packet latency decreases as the inter-frame interval τ increases (and therefore the load decreases), the PAoI shows a different behavior. Specifically, the PAoI is significantly increased with the largest value of τ=4, whereas the other two values of τ =1.5 and τ=2 provide the lowest PAoI mean and tail, respectively.

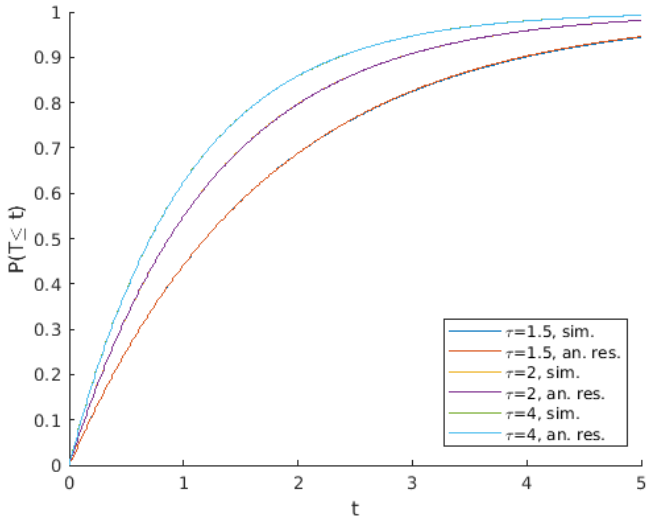


Figure 17. CDF of the latency of a D/M/1 system for 3 different values of the arrival rate: 1.5, 2 and 4

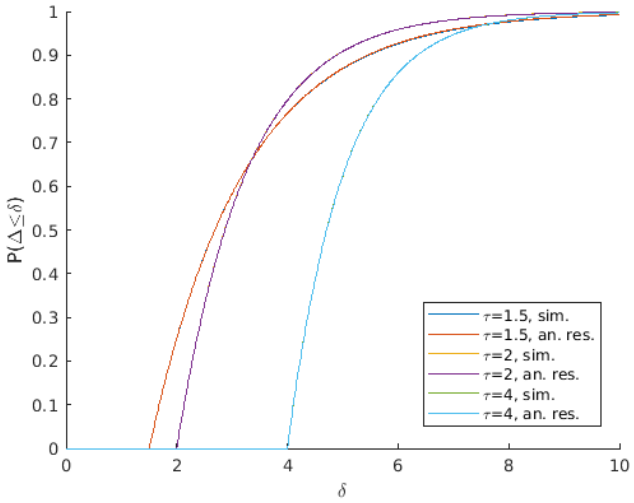


Figure 18. CDF of the PAoI of a D/M/1 system for 3 different values of the arrival rate: 1.5, 2 and 4

VR/AR traffic characterization:

We used traces from a publicly available dataset [6] to attempt to characterize VR traffic better. This analysis is necessary for a more intelligent resource allocation, as predicting the correct frame size can allow the BS to schedule packets more effectively and provide QoS to more sources at the same time.

The video traces were recorded using live VR applications on the VRidge platform, which uses the nVidia NVENC encoder with H264 encoding. This encoder is meant to provide a Constant Bit Rate (CBR) output, i.e., its frames should ideally have the same size. The first important observation (see *Figure 19*) is that this is not true: The figure shows that the rate is actually highly variable. The various lines show different window sizes for a moving-average filter on the

bitrate, averaging bitrate from a frame-by-frame level (the blue line with a window of size 1) to a 5-second basis, as the video is recorded at 60 FPS.

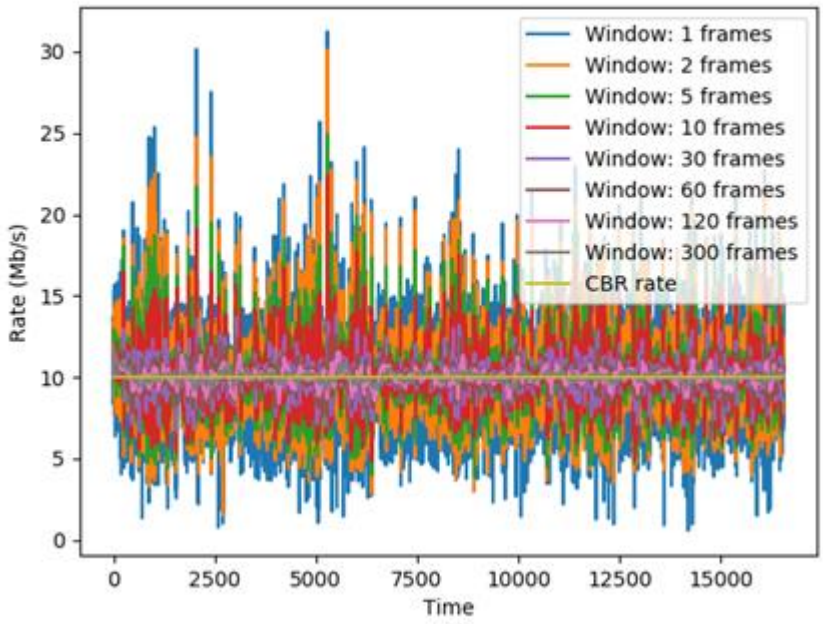


Figure 19. Moving average of the video bitrate over time

This variability has significant implications for resource allocation. Ideally, a CBR video stream would be allocated the same resources for each frame, with no waste and perfectly predictable requirements. However, the real level of uncertainty makes it necessary to leave a significant amount of slack in the resource allocation, as the content of the video and the movement of the user's field of view affect the bitrate in a highly unpredictable way [7].

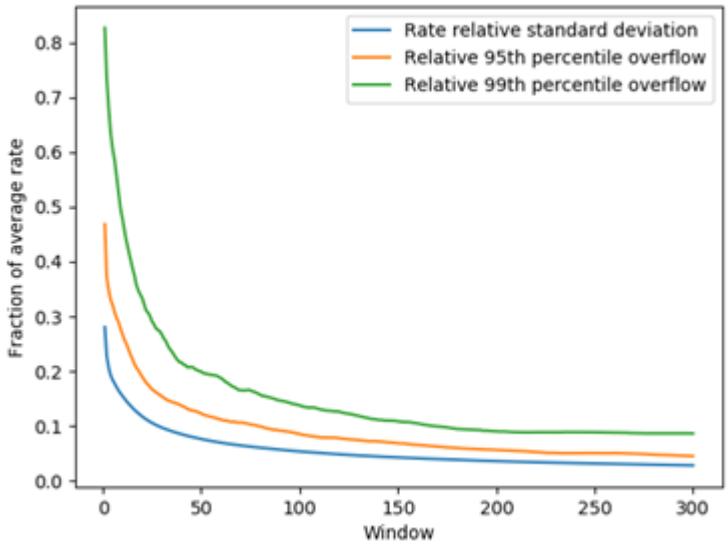


Figure 20 Relative increase in the rate as a function of the scheduling frequency

This effect is also highly dependent on the granularity of the resource allocation: If the scheduling is run at a coarser time granularity, subsequent frames can compensate for variations, reducing the variance of the bitrate. Figure 20 shows the relative increase in the bitrate as a function of the chosen window: The 99th percentile of the bitrate can be over 80% larger than the average if we consider each frame individually, requiring a large safety margin in the scheduling and thereby wasting significant network resources, but only about 15-20% if the scheduling is performed once every second.

2.4.2 VR/AR VIDEO TRANSMISSION WITH MULTIPLE PATHS

To overcome the limitations of a single link, we can look at the use of multiple paths, with strategies such as duplication or splitting the frames. Each scheme has advantages and disadvantages, as there is an important tradeoff between reducing the load on the system (i.e., keeping shorter queues), and increasing the reliability of each single frame.

VR/AR traffic requires a high throughput and strict latency guarantees, as the perception of real-time control needs to be maintained for the user to have a smooth experience. As we described in Section 2, AoI is a relevant metric: as UL transmission from the user to the server are only short commands and acknowledgments, we can assume that they have a constant latency, making AoI equivalent to AoL.

We considered the use of multiple parallel communication paths, which can be achieved in 5G networks (and beyond) by using multiple Radio Access Technology (RAT)s or multiple BS or integrating different technologies and spectrum bands over traditional infrastructure-based networks. Over the past few years, the use of multiple connections to provide stricter latency guarantees has been a subject of intense research, and AR/VR is one of the prime applications for this (including both the industrial use cases proposed in IntelIIoT, such as HIL control, and commercial scenarios such as VR gaming). As such, we considered a system with two paths, modelling each as a D/M/1 queue by assuming

that the video frames are generated with a constant frame rate. We can then assume that each frame of L bits can be split in different ways among the two paths:

An alternating system sends each frame on only one of the two paths, using path 1 for odd-numbered frames and path 2 for even-numbered ones;

- A **split** system simply divides the L bits into two equal subframes of L/2 bits. In order for the frame to be displayed, both subframes have to be delivered correctly;
- A **replicated** system transmits two copies of the whole frame over the two paths, so that each copy is sufficient to decode and display the frame. Naturally, this increases the load on the system;
- A **coded** system encodes the frame into two subframes of L/2 η, where η is the efficiency of the code. By itself, each subframe is not enough to decode the whole frame, but it can be used to show the user a lower-quality version of it. Coding schemes such as Multiple Description Coding (MDC) can implement this kind of system in practical video setups.

The four schemes are represented in Figure 21. We can then model the latency and PAoI of these systems as a function of the rate μ of each path, the frame interval τ, and the error rate ε. Naturally, the coded system's performance will also depend on the coding efficiency.

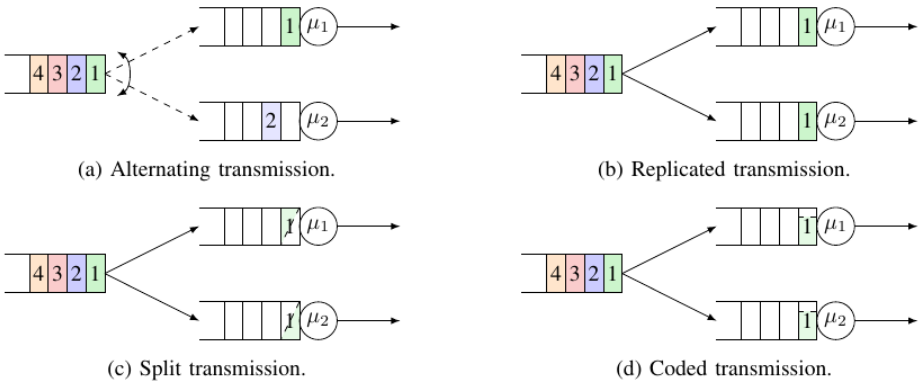


Figure 21. Schematic of the four considered schemes

We derived the closed-form expression for the complete distributions of the latency and PAoI for the four schemes, with a close approximation for the PAoI in the alternating scheme (for which our formula is a lower bound if ε >0). Figure 22 shows the 99th percentile of the latency as a function of τ, showing that less frequent transmissions lead to better latency performance: As the queues are less loaded, frames often find an empty system and are transmitted immediately. Naturally, the split system is the fastest, along with the low-quality version in the coded scheme, but this comes at a cost: The split system also has the highest error probability for frames (0.36 in this case, compared to only 0.04 for the replicated scheme), as losing one of the two subframes is enough to make the whole frame unrecoverable.

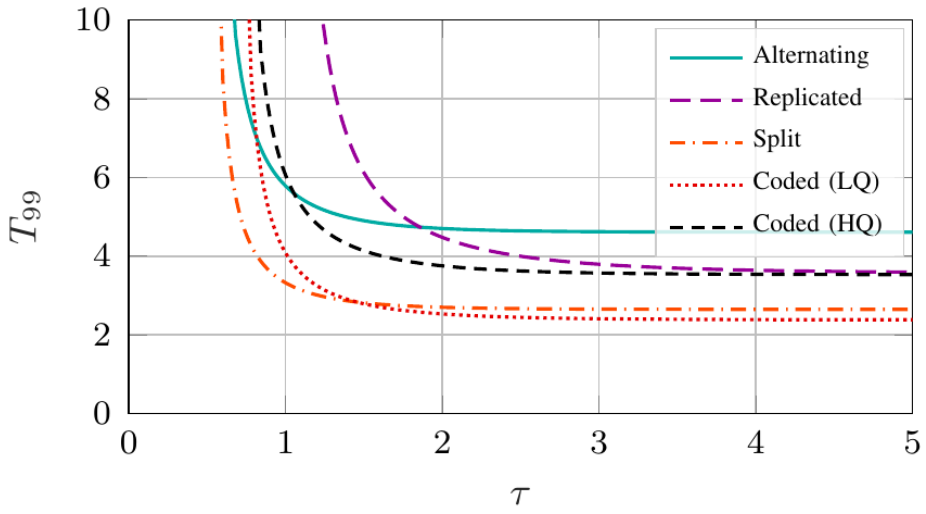


Figure 22. 99th percentile of the latency as a function of the inter-fame interval τ, with μ =1 and ε =0.2 for both paths.

The trade-offs and considerations are completely different for the PAoI, as losing a frame is not necessarily a problem, as long as the next frame is delivered soon enough. In this case, a low frame rate is not beneficial, as the inter-frame interval then becomes the major factor in the PAoI. The curve for all schemes has the typical U shape, as setting a high frame rate also increases the PAoI by increasing the latency for each individual frame. The minimum at which the two factors are balanced depends on the scheme and is at a higher τ for systems that put more traffic on the network, such as the replicated scheme. Surprisingly, the alternating scheme seems to perform better than most of the coded schemes, although they are all very similar, and the replicated scheme has the bonus of having relatively few frame losses: this can make inter-frame encoding more reliable in practical systems, potentially increasing the efficiency of the video encoding. The coded system with MDC has the best PAoI performance for the low-quality version, but the PAoI between subsequent high-quality frames is the highest of any schemes, so it is a good compromise if the low-quality version can be enough for acceptable control.

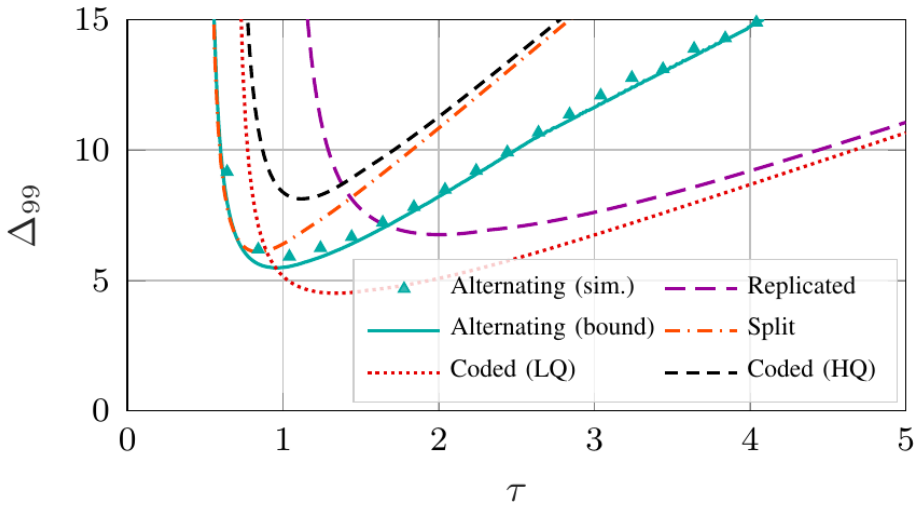


Figure 23. 99th percentile of the PAoI as a function of the inter-frame interval τ, with μ = 1 and ε = 0.2 for both paths

The results above were plotted for η=0.75, but the coding efficiency plays a huge role in the effectiveness of the coded system. We can see this in Figure 24, which shows the 99th percentile of the PAoI as a function of η: In this case, a higher coding efficiency can significantly improve the PAoI for the high-quality version, reducing the load on the systems and allowing for more efficient transmission.

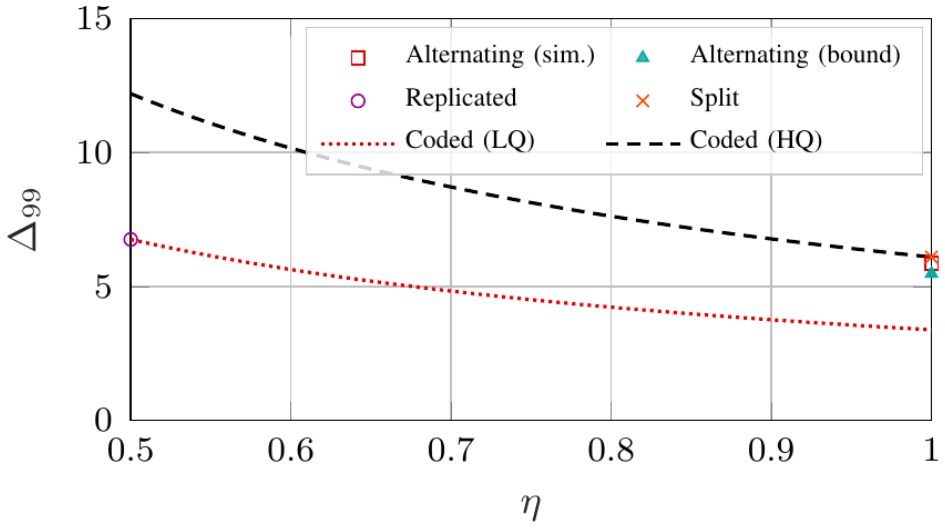


Figure 24. 99th percentile of the PAoI as a function of the coding efficiency η.

While the results above are still a theoretical model that is far from practical considerations, they give a strong indication on the trade-offs in AR/VR transmission over multiple connections in 5G and beyond. The work in the next section is an expansion of this theoretical model into a more practical analysis, using a data-driven approach instead of queuing theory as a main tool for analysis.

2.4.3 VR/AR DATA-DRIVEN ANALYSIS AND OPTIMIZATION

While the work above was based on simplified theoretical models, we have also considered a more realistic approach, based on actual traces from a VR application. The traces were recorded in collaboration with the University of Padova, and traffic analysis and deep packet inspection were used to extract the type and size of packets, both in the uplink and downlink. A short snippet from a trace is shown in Figure 25, with downlink (from the PC rendering the VR environment to the phone which acts as a visor) and uplink (from the visor to the PC) packets highlighted in different colors depending on their meaning. As the figure shows, most of the traffic volume is caused by the video frames, which are both more frequent and larger.

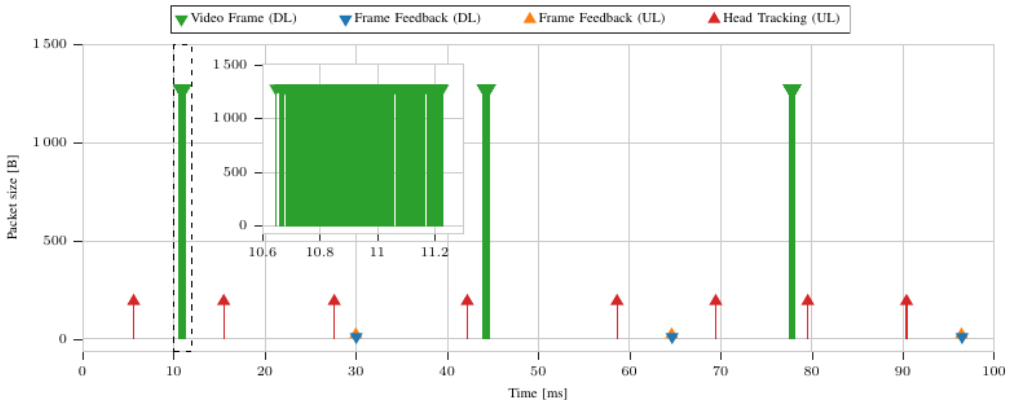


Figure 25. Traffic analysis for the VR application.

The VR framework we used (RiftCat Vridge, a commercial Google Cardboard VR engine) used the H.264 encoding, enabling the intra-block refresh option to make the traffic Constant Bit Rate (CBR), i.e., approximate the condition we had in the previous section by making all frames approximately the same size. However, the actual frame sizes were not exactly the same, with as much as a 30% variation: not as much as in standard Variable Bit Rate (VBR) encoding, which can have order of magnitude differences, but still significant, as can be seen in Figure 26.

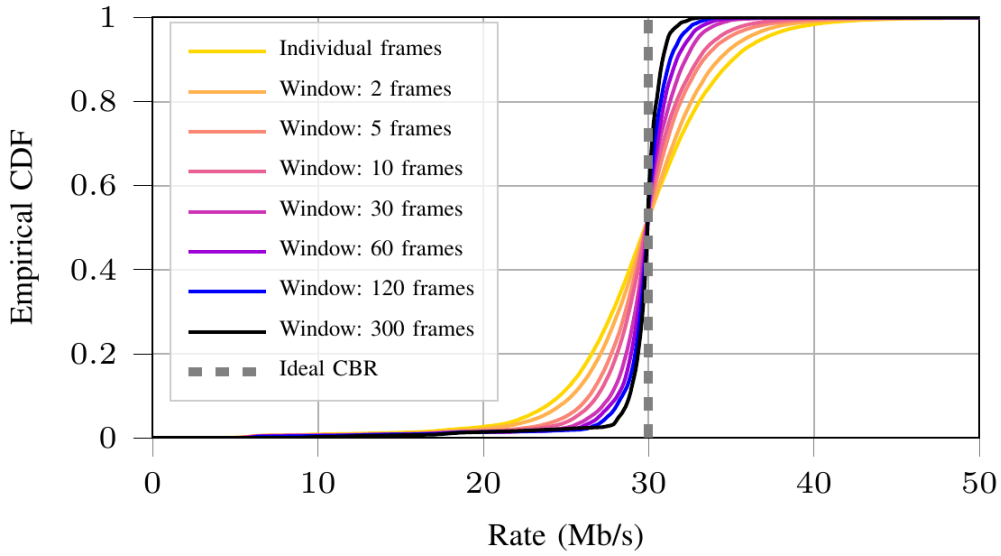
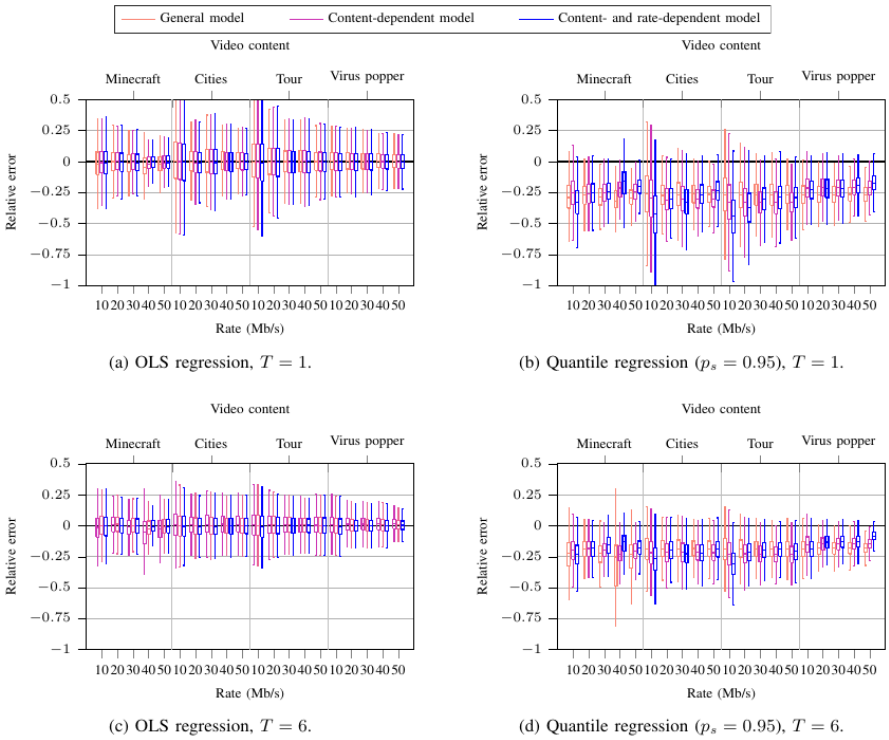


Figure 26. Frame size distribution with different smoothing windows.

We can consider a simple linear regression model to predict future frame sizes, as the patterns in the data seem simple enough. The error of the model is relatively high, indicating that there is a strong random component in the frame size,

but as Figure 27 shows, the model is easy to generalize, as regressors trained on the whole dataset perform about as well as the ones trained on the specific content and rate considered in the test.



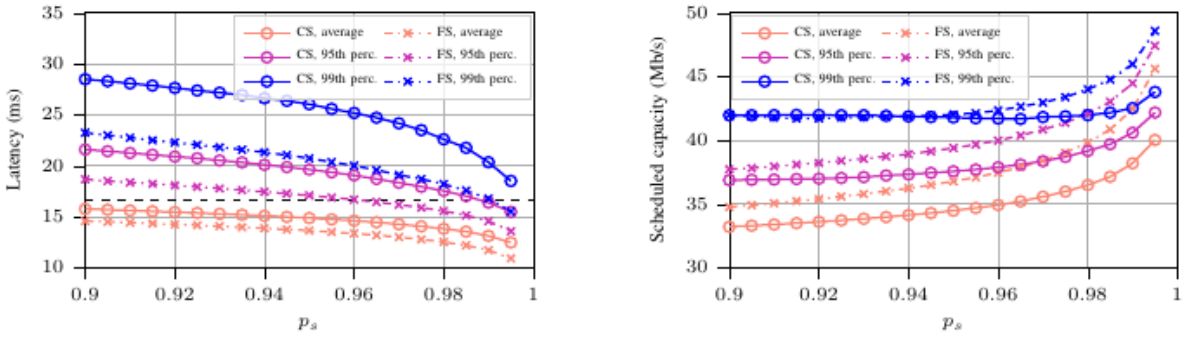
(a) OLS regression, $T = 1$.

(b) Quantile regression ($p_s = 0.95$), $T = 1$.

(c) OLS regression, $T = 6$.

(d) Quantile regression ($p_s = 0.95$), $T = 6$.

Figure 27. Prediction model error on the average (OLS) and 95th percentile of the frame size.



(a) Latency.

(b) Scheduled capacity.

Figure 28. Predictive slicing performance.

Finally, we can consider the use of this prediction: if we know the size of future frame sizes, we can adaptively slice the network resources so as to guarantee a low-latency VR transmission, which is crucial for the user experience, while using as few bandwidth resources as possible. As Figure 28 shows, there is a trade-off between latency and required capacity, but by carefully tuning the objective percentile p_s, the Constant Slicing scheme (CS), which gives the VR flow

a constant capacity for the next 100 ms, can achieve the same performance as the Frame-by-Frame Slicing scheme (FS), which assigns capacity dynamically for each frame based on the prediction.

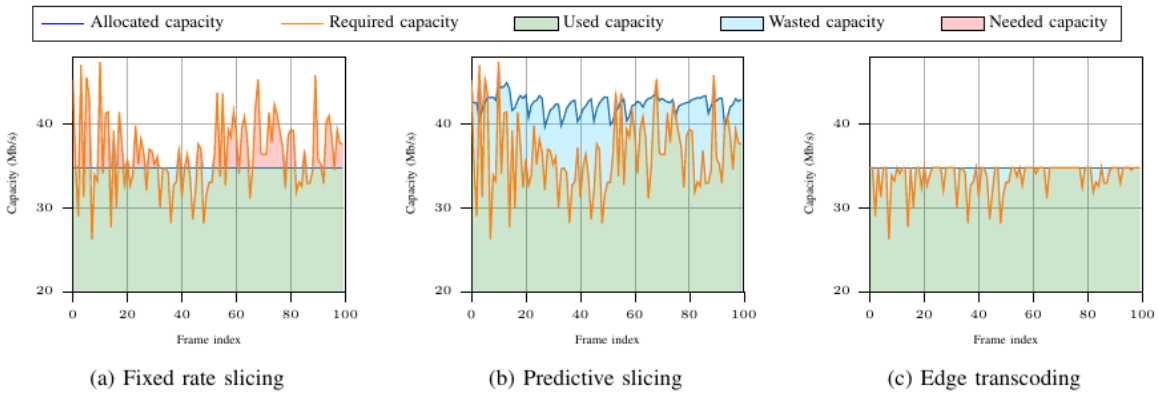
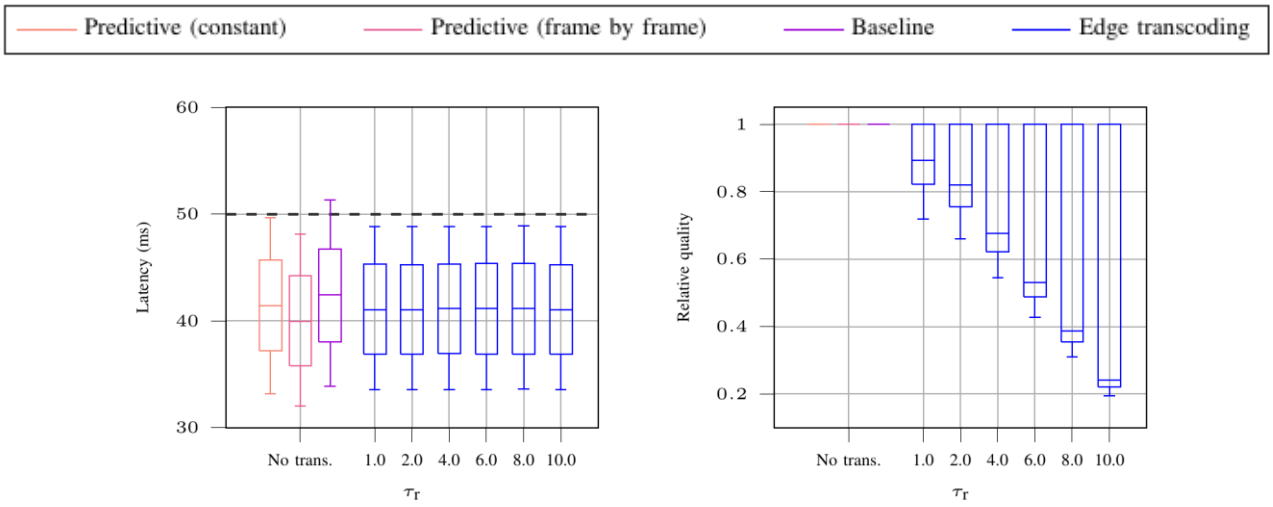


Figure 29. Edge transcoding latency schematic.

Another thing we can consider is edge transcoding: normally, movement data is sent from the HMD to the Cloud server rendering the VR content, which then renders the scene and transmits it back to the HMD. In the scheme on the right side of Figure 29, however, the Base Station (BS) can also decide to transcode the frame into a more compressed version, saving time and resources by slightly reducing the picture quality. The three possibilities are then fixed slicing (which has either significant overprovisioning or highly volatile latency), predictive slicing (which can use resources more efficiently, but still requires overprovisioning to guarantee a low latency), and transcoding, which has low latency and a low resource footprint, but reduces the picture quality. Figure 30 shows the performance of edge transcoding, as compared to slicing: we can see that edge transcoding can maintain low latency as consistently as predictive slicing (with a lower capacity utilization), but the quality of the video decreases sharply if the transcoding time τ_f grows: if the BS takes more than 2 ms to transcode the frame, the video quality decreases by 30% or more. The choice of the optimal scheme is then highly dependent on the amount of computational resources available to the BS, and whether the trade-off between computation and communication is worth it.



(a) Motion-to-photon latency. (b) Picture quality.

Figure 30. Edge transcoding performance.

The main conclusions of the study of RRM for HIL are listed in Table 4

RRM for HIL
VR/AR traffic is very demanding for the computation and communications infrastructure, requiring high throughput, high reliability, and low latency and AoI.
VR transmissions with state-of-the-art CBR encoders lead to variable traffic that calls for traffic prediction and data-driven solutions to optimize the use of resources.
Parallel transmissions can be exploited for VR/AR transmissions, and the optimal strategy depends on the complex tradeoffs among picture quality, frame rate, and timeliness requirements.

Table 4. Key takeaways of RRM for HIL

2.5 Implementation of the communication resource manager

Implementation: In the implementation side, the first step has been to create a local setup of 5G infrastructure in the form of a platform utilizing Open Air Interface (OAI). The OAI alliance is a non-profit consortium to develop ecosystem for open-source software/hardware development for the Evolved Packet Core (EPC) and RAN of 3GPP cellular networks, including 5G. IntellIoT has selected the specific implementation of EURECOM, developed in T4.2 for the specific purposes of IntellIoT. Within OAI, Mosaic5G provides an ecosystem of 5G R&D open-source platforms ranging from the centralized network control to the mobile edge network deployment. Specifically, Mosaic5G testbed architecture consists of seven software modules along with hardware components: OAI, FlexRAN, LL-MEC, Store, JOX FlexnCN and FlexRIC. The most relevant ones for this task are the first three:

1. **The OAI platform** is composed of 2 separate modules, the first is OAI-CN which exists for both 4G and 5G. The CN can be deployed disaggregated across different computers or on a single host by deploying a set of docker containers than are required for the CN in question.
   The second module, OAI-RAN, contains an executable that can be used to perform simulation and emulation of the Evolved NodeB (eNB), User Equipment (UE), New Radio (NR) UE and gnodeB (gNB). These modules can be both utilized for a simulation set-up, which skips lower layer functionalities (the radio), and allows easy scaling of UEs. This can be used to verify and test higher-level functionalities, e.g., scheduling policies and how this affects the experienced network performance viewed from the perspective of different UEs. They can, however, also be run in an emulation environment where the entirety of a communication system is realized, i.e. the radio functionalities are also performed by utilizing software defined radios, which produces more realistic results at the expense of scaling being more difficult. With OaI-CN and OaI-RAN the functionalities to realize a emulated 5G setup can be realized.
2. **FlexRAN** is a real-time RAN controller, which specifically controls underlying eNBs. The FlexRAN Application Programming Interface (API) offers the ability to extract information of the radio network through the BS (eNB). It furthermore facilitates radio slicing, to allocate a set amount of RBs, or bandwidth, to specific UEs whom may have a certain application that requires specific requirements (e.g., low latency, high reliability, …) that can be guaranteed through this allocation. This was initially used as a framework for the communication resource manager while the FlexRIC was being developed.
3. **FlexRIC** A newer implementation with similar functionalities to FlexRAN. It is, however, implemented with OpenRAN in mind as the near real-time RAN Intelligent Controller (RIC). New interfaces and different functionalities now available compared to FlexRAN with cross-generational support for both the eNB and gNB. From a practical perspective it can do the same in terms of slice allocation with different scheduling policies for the UEs, hence similarly bandwidth, low latency and such slices are feasible.
4. **LL-MEC** is a Low Latency Multi-access Edge Core that involves bringing the data closer to the user. This requires rerouting the data plane through a data centre on the edge of the network, close to the user, from which data can be stored and transmitted with lower latency compared to the alternative of data stored at far away data centres. This can be used to reduce the delay of some applications, as well as slicing on a network level.
5. **FlexCN** is a module meant for the control and monitoring of the system on a 5G CN. It currently only allows for monitoring of the CN in terms of the AMF and SMF. Specifically relevant is that the CN controller also allows the existence of application which will be a necessity for the slice creation and identification of UEs. That is, a translation must be done between the RNTI (on the gNB used for UE identification) and the IP, or IMSI which are known may be known to the UE. To create a slice for a specific user in a multi-user scenario an application for translating the UE identifier on the gNB may be required to map the RNTI to the correct UE intended for a slice.

The local setup was initially implemented with a simple setup for the future deployment and development of the Communication Resource Manager. The initial focus was on deployment of FlexRAN and performing the slicing only on the RAN level, which simplifies the setup equipment, and more will be needed for a more elaborate setup with more of the aforementioned software modules, i.e., LL-MEC and FlexCN. With the progressive updates the FlexRIC was released, which had a very different way to communicate and interactions with the base station it was connected with, which meant redesigning the way the Communication Resource Manager operated, when using the FlexRIC compared to the FlexRAN. This being a requirement with FlexRAN's limited support to only 4G, and FlexRIC supporting both 4G and 5G. Internal logic, and interaction with users requesting a slice could however be kept similar.

To test the implementation, it has been done over different configurations. The initial premise was with the FlexRAN using a deployment scenario with a single computer hosting simulated UEs connected to the eNB. This meant skipping the radio-level communication. The core was also hosted there in a containerized environment.

Initially the slicing mechanism implemented through the FlexRAN API was also tested and evaluated. When a slice is created it separates a set of frequency resources from the pool of available resources. While default users have access to the initial pool of resources and share these in a best effort manner, single or sets of UEs can be associated with a slice and have access to isolated resources. The slice of resources provides the users with some guarantees on the performance that they shall experience in terms of bandwidth and latency. These users experience would be independent of the best effort slice users, and thus creates a more deterministic performance on which guarantees could be made.

The setup was then extended as newer implementation became available, specifically FlexRAN has been switched out for FlexRIC, and the eNB, UE and CN switched out with their 5G counterparts. Similarly, a simulator was used when starting this, to ensure the initial development and integration of the Communication Resource Manager. Tests have in the following been performed with emulated setups where the UE, and nrUE were deployed on a different computer and connecting to the base station through a radio-link deployed through Software Defined radios. Here the slice and expected performance set for the slice could be investigated to see if it applies in a real scenario. Specifically, the ability to support a specified bitrate which the UE would have.
Due to limited functionalities parts of the testing meant for the 5G resource management is based on the 4G implementation as it has been more complete while the interfaces through the FlexRIC are similar between 4G and 5G. While 5G is the use case scenario and functionalities should be evaluated for it, the initial testing has been performed on 4G due to a more stable performance locally, and the fact the interface between the base station and FlexRIC are meant to be the same, independent of whether it is a gNB or eNB.

With the new interfaces and methods towards interacting with the FlexRIC compared to FlexRAN the features supported now are:

- **Creating a network slice** by requesting a set of resources for a user through choosing its Radio Network Temporary Identifier (RNTI)(i.e. a pseudo-random ID that is generated and granted to a UE). The slice created only applies in the downlink, as uplink slices have not yet been implemented on the gNB. Upon creating a network slice an identifier is returned identifying the slice. It works by creating a HTTP GET request.
- **Removing a network slice**. It works by using a HTTP DELETE request. In this form the slice ID must be sent along with the request.
- **Associating a UE, or new UEs to an existing slice** by using the slice ID and the RNTI of the UE to add. This will mean that the resources are to be shared between all occupants on a slice.
- When creating a slice one of the desired inputs has been a requested bitrate that should be available on the slice. This has been implemented by monitoring the radio link quality and allocating a set of physical resources to ensure that. When changes to the radio link occur, a reallocation may occur to ensure the slice guarantee. An example of this is visualised on Figure 31. This is done with a frequency to ensure the guaranteed bitrate is upheld.
- Taking an existing slice configuration, it can be updated to support a **different requested bitrate**.
- **Requesting information on a specific slice** by inputting its slice ID, or by requesting all the currently active slices that are being supported by the BS.

Some of these functionalities are directly exposed to be used through HTTP request, while others happen in the background of actively used for the creation and upholding of a slice definition. The interface with the Communication resource manager has been designed and implemented as a python-flask server and by using swagger for interface design. Interaction is with a set of predefined interfaces that take certain inputs to create, delete, or get information on a network slice.

Figure 31 shows one example of dynamic allocation of slices.

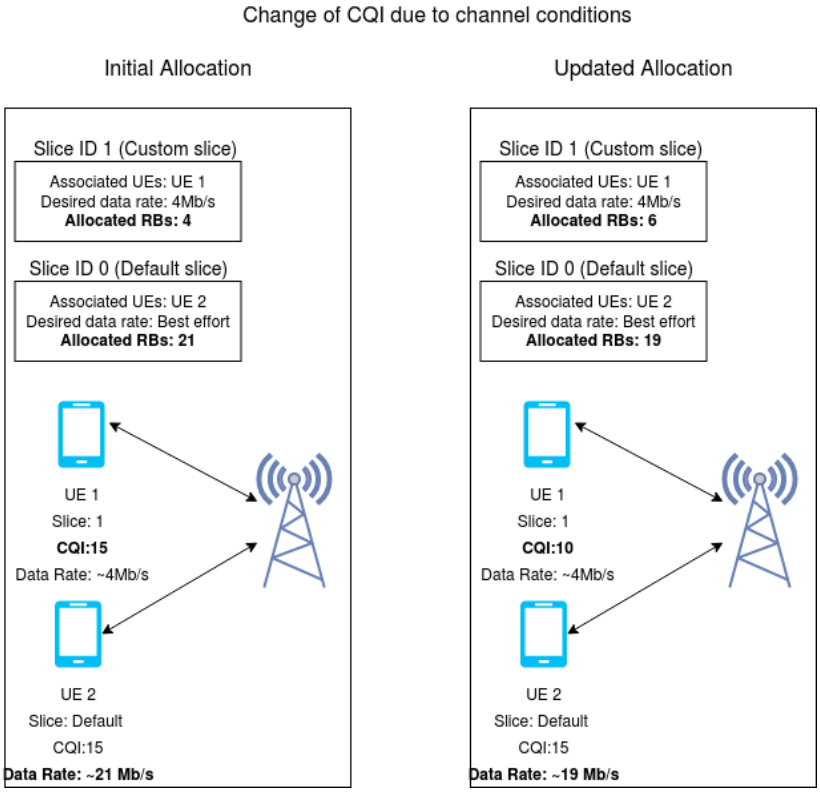


Figure 31. Example of dynamic resource allocation

Experimental measurements and estimation of VR latency over OAI: Towards creating a latency-based slice mechanism for VR application, we extend the previous theoretical and simulation studies and measure the experienced latency of VR data used in Section 2.4.3 but now being sent to a UE over the OAI network. The measurement is compared against a latency estimation based on RAN statistics. To measure this on the local setup at Aalborg University, we deploy a server on the core network acting as the MEC of UC1/UC3.

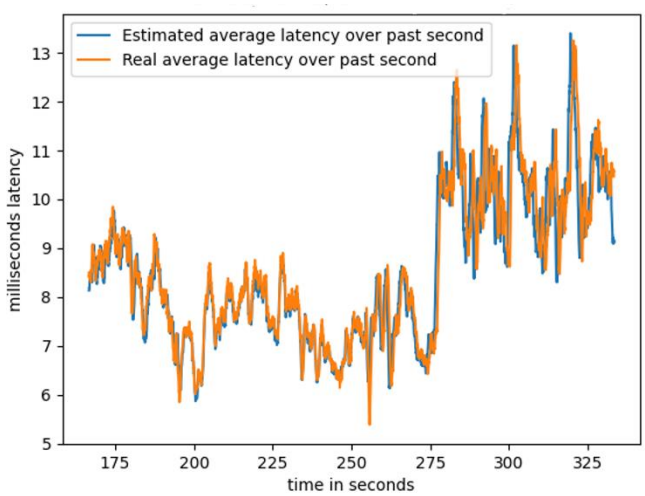


Figure 32. Estimated VR latency and comparison to experimental measures over a 5G OAI network. The parameters are 14 RBs, 60 FPS and 10Mbit/s encoding

The results are shown in Figure 32. We set a slice for the VR user with 14 Resource Blocks, a video encoding of 10Mbit/s and 60 FPS. The figure illustrates the measurements of the average latency over the past second window as measured by the Communications Resource Manager xApp on the flexRIC (blue), and the latency observed at the UE (orange). The average is calculated as a sliding window over the latency of the past 60 frames measured from RAN statistics and each frame received on the UE to make a one second average which we compare against each other. The UE latency is measured by encoding timestamps into UDP packets when the transmission of a video frame occurs, and subtracting it from the time upon arrival. The clocks of the computers are synchronized. We observe that the estimated latency matches the measurements very well. The small drift (e.g., around 275 to 325seconds) is related to imperfect measurements of video frames, specifically having less frames on the RAN than sent to the UE. Note that this difference is not due to lost packets, but related to the separation between video frames and the measurement. This estimation of the latency is used for a better (re-)configuration of the VR slice at the Communications Resource Manager.

3 WIRED NETWORK MANAGEMENT

The second part of the communication optimizations is the wired segment and TSN controller (C2). Ethernet networks can operate without management. However, to guarantee QoS parameters for certain data streams, explicit network management is required. The same is true for other possible management issues, e.g. network segmentation, access protection, etc., which is out of the scope of the project. The scope of the work on network management is rather to guarantee QoS parameters through the factory backbone, i.e. the wired communication network. In IntellIoT, a TSN network is used here, which allows to fulfill the QoS requirements described below, particularly concerning machine control and remote help from a human operator. Of course, unmanaged best-effort traffic for uncritical applications is still possible through the factory backbone network. To configure the network, we are using a network controller, i.e. we are following the fully centralized configuration model defined in IEEE 802.1Qcc, see Figure 33.

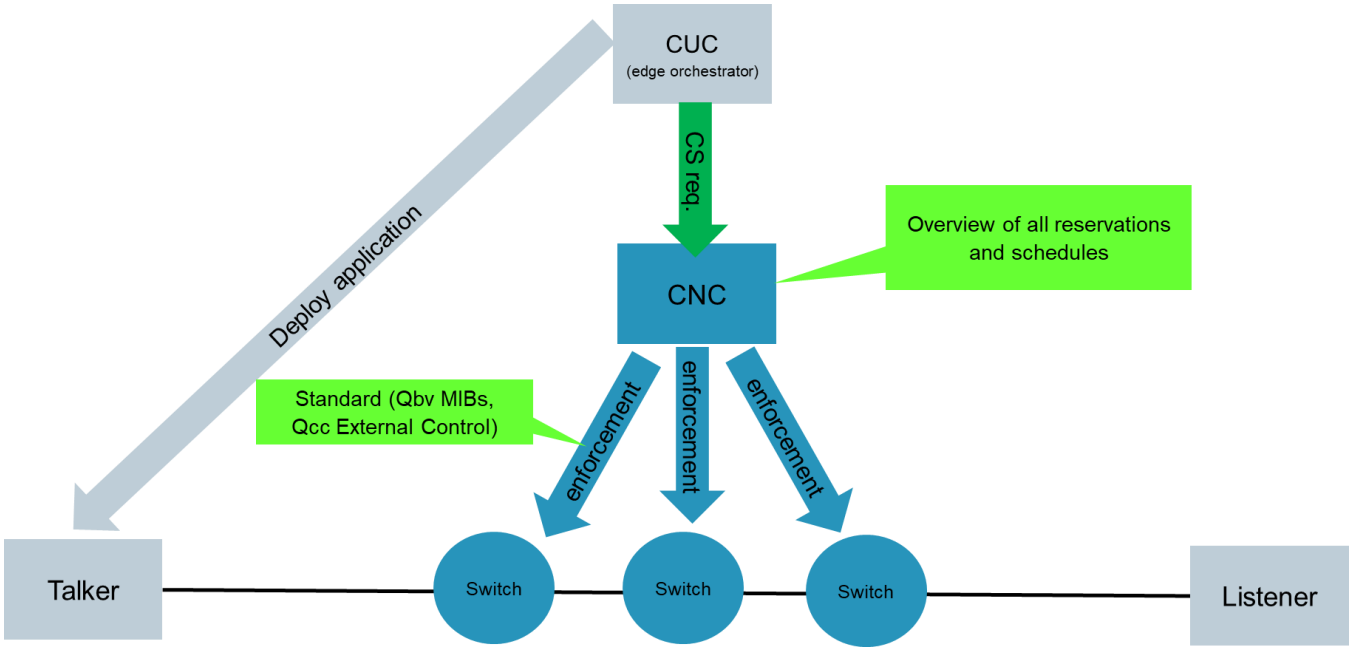


Figure 33. Fully centralized configuration model

The TSN controller described in this section conforms to the Centralized Network Configuration (CNC) described in IEEE 802.1Qcc, while our edge orchestrator takes the role of Centralized User Configuration (CUC). An Edge Orchestrator deploys the applications on HW resources in the edge, which are Talkers and Listeners from the network's point of view, and it requests the according communication services from TSN controller.

3.1 TSN controller & QoS requirements

The TSN controller gathers information about the active network topologies. To do so, it reads LLDP information from network devices. Additionally, it reads available YANG models from the devices to gather information about their capabilities and current configuration. End stations supporting LLDP can be detected in the same way. Information about the active topology and relevant properties, particularly available bandwidth and estimated delay values for new streams, are provided via the network controller's northbound HTTP interface, which is predominantly used by the edge orchestrator. Via the same interface, the TSN controller also receives so-called communication service requests, i.e., reservation enquiries for network resources. Communication service requests contain required

properties of the communication service, i.e., source node, destination node(s), packet size, and send cycle. Additional optional parameters, like delay or jitter constraints, send window constraints or preferred Virtual Local Area Network (VLAN) ID are possible. Future extensions, e.g. for reliability requirements, are planned. Communication service requests are predominantly, but not exclusively, filed by the edge orchestrator. Requests from standalone applications, from edge perspective so-called unmanaged applications, are possible.

For the received communication service requests, the TSN controller computes paths and schedules or reserves network resources via per-stream filtering and policing (PSFP), depending on the QoS requirements. The TSN controller enforces the computed communication services in involved network nodes, and end devices where applicable. It then informs the requester about the result of its computation and enforcement attempts. In success case, this contains a confirmation of the request, the achieved QoS properties (which are at least as good as requested) and eventually additional parameters like assigned VLAN ID or time-aware offset. In case of failure, it informs about the failure.

In IntellIoT's manufacturing use case, predominantly the following two applications bring interesting QoS requirements to the TSN network:

(1) The milling machine is set up with a controller in the form of an Edge Application, that means the control of the milling machine is not computed on a dedicated hardware, but on an edge device. This edge device does of course not have direct physical connection to the axes of the milling machine, like e.g., Pulse Width Modulated (PWM) signals or stepper motor pulses. It is rather connected to small TI Sitara controllers, which are controlling only one axis of the milling machine each, through the TSN backbone. The control signals are thus time critical and need to be executed synchronously to achieve high-quality milling results. Required QoS values are in the range of sub-milliseconds, while smaller values enable a faster and more precise operation of the machine. It should be pointed out that this differs substantially from so-called best-effort traffic, where the simple rule "faster is better" applies. Once the speed of operation of the milling machine is decided, the delay and jitter values taken for the calculation of the operation speed need to be guaranteed to ensure proper milling results.

(2) In the HIL scenario, a human operator directly controls the robot, based on the image she sees on her AR glasses. Same as in the tractor-example given in chapter 2, the AoL can be a good metric in this scenario. Experiences from gaming industry give rise to the assumption that a delay of up to 80ms from a hand movement to seeing the effect of the movement is tolerable. This delay comprises the detection of the movement in the stylus device, computation and serialization of a movement command, transport, reception, and execution of the movement command by the robot controller, inertia of the robot, capturing and streaming of the image through the camera, transport again, reception of the image and displaying it through the AR glasses. The overall performance of the system is a research objective in UC3, the time budget left for communication is to be identified. However, a guarantee for the communication delay is required, as a sudden change in the experienced delay will make a proper operation of the robot extremely difficult or even impossible.

3.2 Selection of TSN features

Based on the QoS requirements described above, proper TSN features have been selected. For the synchronous operation of the milling machine axes, synchronization based on IEEE1588v2 (Precision Time Protocol, PTP), is required. For maximum performance, i.e., optimized balance between speed of operation and quality of result, minimal delay and minimal jitter are the goal for the milling machine control. To achieve this, Time-Aware Scheduling (TAS),

(IEEE802.1Q-2018, formerly known as IEEE802,1Qbv) is the TSN feature of choice. Due to the static nature of the milling machine – even if occasional re-deployments of the control application are to be taken into account – the considerable effort for computation and deployment of TAS is certainly reasonable.

For the HIL use case, practical tests comprising the complete functional chain from the stylus input device through the robot and the camera to the AR glasses have shown, that TAS would not really be needed. Particularly, because the end devices in use, i.e. the robot controller, camera and HoloLens, are not ready to provide their send data at a pre-planned point in time. Thus, for this communication service, strict priority scheduling combined with PSFP (IEEE 802.1Q-2018, formerly known as IEEE802.1Qci) could be applied. Allowing for more jitter and additional delay of about 12,5µs per hop, this solution would avoid a re-configuration of all network nodes transporting data between a robot needing help and the operator in charge. Instead, only end points at the robot and operator side would need to be reconfigured to enable PSFP to protect the strict priority schedule from unplanned or unwanted packets in high priority queues. However, as we anyway need a TAS scheduled network due to the more stringent requirements from the milling machine, we decided to also use TAS scheduling for the HIL use case.

The conclusions of this study are listed in Table 5:

Wired network management
TSN offers several standards to improve QoS in Ethernet networks. Lower delay comes with higher cost and effort together with lower flexibility. We have selected the best match for IntellIoT's requirements: As TAS is needed for the milling machine requirements, we apply it to all streams for the sake of consistency.
TSN controller appears as an edge application and can be deployed somewhere in the edge environment.
TSN controller interacts with edge orchestrator and Security Assurance Platform to receive communication service requests and commands to exclude malicious devices. Application's QoS requirements are relayed through edge orchestrator.

Table 5. Key takeaways of wired network management

4 OPEN SOURCE

The Communications Resource Manager (C2) is available here:

https://gitlab.eurecom.fr/intelliot/communication-resource-manager

In the following, some relevant information to use the code.

Building and running the docker container

To build the image run "./build.sh" this will create the image. This will require internet to install needed dependencies

After making the image, the container should be runnable with docker-compose up. This will start a python flask server in the container, which supports the API calls defined previously. Note that the .env file contains IP of the FlexRIC which must be set, as well as IP for the manager itself. The available rate must also be defined as it is a requirement for the slice functionality to know the bandwidth available. The slicing allocates as a fraction of the entire rate, so if this is wrongly set the slices will not support the requested rate.

API defintion can be found in commrm-api.yaml

Functionalities

The slice mechanism works for both 4G and 5G utilizing the NVS slicing as developed for the flexRIC. Currently only downlink slices are supported, and the latency measures are intended for future work specifically towards guarantees of VR frames that are sent over the air.

Due to limitations with relating IP to unique identifier in the RAN (RNTI), currently only a single UE can have its own slice, and this is simplified to be the first UE joining the network. This can be addressed by encoding the RNTI and adding the specific RNTI when creating a slice as opposed to the first RNTI

Simple API tests

API can be called through application like postman, otherwise example ones from the terminal can be seen below

curl -X POST --user : http://$IP:8081/reservation -H'Content-Type:application/json' --data-raw '{"maxLatencyMillisUplink": 0.80, "addressType": "ipAddress", "maxJitterMillisUplink": 6.02, "maxLatencyMillisDownlink": 5.96, "maxJitterMillisDownlink": 5.63, "minThroughputMbpsDownlink": 5, "clientAddress": "192.168.55.55", "minThroughputMbpsUplink": 1.46}'

Get single reservation test curl -X GET --user : http://$IP:8081/reservation/1

Get All reservations test curl -X GET --user : http://$IP:8081/reservation

Delete reservation test curl -X DELETE --user : http://$IP:8081/reservation/1

Running

To run the application, make sure a flexRIC is running and connecting to a eNB or gNB. Only OpenAirInterface implementaitons should be suppported for this, and these must also have the flexRIC patch applied.

With a UE connected, or multiple, a network slice can be created that allocates part of the bandwidth to the first UE that joined the network

5 CONCLUSIONS AND NEXT STEPS

This final report of Task 4.3 has described the main challenges of the dynamic resource management in intelligent IoT environments. The task has been structured in research activities and implementation activities, the latter aiming at providing the two components for dynamic management to be integrated in WP5 and the UCs. In the research, the work has been divided for the different types of traffic: HIL, ML and mixed traffic and network slicing. For the implementation, the communications resource manager is an xApp integrated with the rest of 5G infrastructure provided by Task 4.2. For the TSN controller, the interfaces have also been defined and time-aware scheduling is to be considered. We note that this task provides components that are universal and to be integrated in 5G and TSN networks, therefore applicable to any 5G or TSN traffic or Use Case, beyond the ones defined in IntellIoT.

ACRONYMS

IoT	Internet of Things
URLLC	Ultra-Reliable Low Latency Communications
TSN	Time Sensitive Networking
5G	5th Generation
HIL	Human in the Loop
V2V	Vehicle to Vehicle
AoI	Age of Information
PAoI	Peak Age of Information
VR	Virtual Reality
AoL	Age of Loop
WNCS	Wireless Network Control System
UL	Uplink
DL	Downlink
RL	Reinforcement learning
EC	Effective Capacity
QoS	Quality of service
AR	Augmented Reality
ML	Machine learning
AI	Artificial Intelligence
BS	Base Station
FCFS	First Come First Serve
CDF	Cumulative Distribution Function
AGV	Autonomous Guided Vehicle
FL	Federated Learning
PS	Parameter Server
RB	Resource Block
SINR	Signal to Interference + Noise Ratio
CSI	Channel State Information
GPR	Gaussian Process Regression

SGD	Stochastic Gradient Descent
RRM	Radio Resource Management
FPS	Frames per second
CBR	Constant bit rate
RAT	Radio Access Technology
MDC	Multiple Description Coding
DLT	Distributed Ledger Technology
3GPP	Third Generation Partnership Project
RAN	Radio Access Network
eMBB	Enhanced Mobile BroadBand
mMTC	Massive Machine Type Communication
NOMA	non orthogonal multiple access
OMA	Orthogonal Multiple access
SIC	Successive Interference Cancellation
AWGN	Additive White Gaussian Noise
SNR	Signal to Noise Ratio
EB	Effective Bandwidth
OAI	OpenAirInterface
EPC	Evolved Packet Core
eNB	Evolved Node B
UE	User Equipment
NR	New Radio
gNB	gNodeB
API	Application Programming Interface
IMSI	Internal Mobile Subscriber Identity
RNTI	Radio Network Temporary Identifier
CQI	Channel Quality Indicator
MEC	Multi-access Edge Computing / Mobile Edge Computing
CNC	Centralized Network Configuration
CUC	Centralized User Configuration
VLAN	Virtual Local Area Network

PWM	Pulse Width Modulated
TAS	Time Aware Scheduling
PSFP	Pre-Stream Filtering and Policing

BIBLIOGRAPHY

[1] F. Chiariotti, O. Vikhrova, B. Soret, P. Popovski, "Peak Age of Information Distribution for Edge Computing over Wireless Links", IEEE Transactions on Communications, 2021

[2] F. Chiariotti, B. Soret, P. Popovski, "Latency and information freshness in multipath communications for virtual reality", 2021, preprint: https://arxiv.org/abs/2106.05652

[3] P. Popovski et al., "A perspective of time towards 6G", 2021, preprint: https://arxiv.org/abs/2106.04314

[4] E. Uysal et al., "Semantic communications in networked systems", 2021, preprint: https://arxiv.org/abs/2103.05391

[5] P. Popovski, K. Trillingsgaard, O. Simeone and G. Durisi, "5G Network Slicing for eMBB; URLLC, and mMTC: a communication-theoretic view," *IEEE Access*, pp. 55765-55779, 2018.

[6] M. Lecci, M. Drago, A. Zanella, and M. Zorzi. "An Open Framework for Analyzing and Modeling XR Network Traffic." IEEE Access 9 (2021): 129782-129795.oneM2M, TS-0001-V4.11.0, "Functional Architecture", Version 4.11.0, Jun. 2021. https://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=33796

[7] F. Chiariotti, "A Survey on 360-Degree Video: Coding, Quality of Experience and Streaming." Computer Communications 177 (2021): 133-155.

[8] P. M. de Sant Ana, N. Marchenko, P. Popovski and B. Soret, "Age of Loop for Wireless Networked Control Systems Optimization," *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2021, pp. 1-7, doi: 10.1109/PIMRC50174.2021.9569366.

[9] J. Gong, Q. Kuang, X. Chen and X. Ma, "Reducing Age-of-Information for Computation-Intensive Messages via Packet Replacement," *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2019, pp. 1-6, doi: 10.1109/WCSP.2019.8927943.

[10] C. Li, S. Li and Y. T. Hou, "A General Model for Minimizing Age of Information at Network Edge," *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 118-126, doi: 10.1109/INFOCOM.2019.8737437.

[11] N. Abramson, "The ALOHA system: Another alternative for computer communications," in Fall Joint Computer Conference, ser. AFIPS '70 (Fall). ACM, Nov. 1970, p. 281–285

[12] X. Song, X. Qin, Y. Tao, B. Liu, and P. Zhang, "Age based task scheduling and computation offloading in mobile-edge computing systems," in Wireless Communications and Networking Conference Workshop(WCNCW). IEEE, Apr. 2019

[13] J. Goseling, C. Stefanovic, and P. Popovski, "A pseudo-Bayesian approach to sign-compute-resolve slotted ALOHA," in International Conference on Communication Workshop (ICCW).IEEE, Jun. 2015,pp. 2092 – 2096.

[14] L. Green, "A queueing system in which customers require a random number of servers," Operations research, vol. 28, no. 6, pp. 1335–1346,Dec. 1980.

[15] E. Reich, "Note on queues in tandem," The Annals of Mathematical Statistics, vol. 34, no. 1, pp. 338–341, Mar. 1963

[16] P. J. Burke, "The output of a queuing system," Operations Research, vol. 4, no. 6, pp. 699–704, Dec. 1956

[17] A. Kaloxylos, "A Survey and an Analysis of Network Slicing in 5G Networks," in IEEE Communications Standards Magazine, vol. 2, no. 1, pp. 60-65, MARCH 2018, doi: 10.1109/MCOMSTD.2018.1700072.

[18] C. Xiao, J. Zeng, W. Ni, R. P. Liu, X. Su and J. Wang, "Delay Guarantee and Effective Capacity of Downlink NOMA Fading Channels," in IEEE Journal of Selected Topics in Signal Processing, vol. 13, no. 3, pp. 508-523, June 2019, doi: 10.1109/JSTSP.2019.2900938.

[19] W. Yu, L. Musavian and Q. Ni, "Link-Layer Capacity of NOMA Under Statistical Delay QoS Guarantees," in IEEE Transactions on Communications, vol. 66, no. 10, pp. 4907-4922, Oct. 2018, doi: 10.1109/TCOMM.2018.2832643.

[20] M. Amjad, L. Musavian and S. Aïssa, "NOMA Versus OMA in Finite Blocklength Regime: Link-Layer Rate Performance," in IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 16253-16257, Dec. 2020, doi: 10.1109/TVT.2020.3037488.

[21] B. Soret, M. C. Aguayo-torres and J. T. Entrambasaguas, "Capacity with Explicit Delay Guarantees for Generic Sources over Correlated Rayleigh Channel," in IEEE Transactions on Wireless Communications, vol. 9, no. 6, pp. 1901-1911, June 2010, doi: 10.1109/TWC.2010.06.081599.

[22] M. A. Abd-Elmagid, N. Pappas, and H. S. Dhillon, "On the role of age of information in the Internet of Things," IEEE Communications Magazine, vol. 57, no. 12, pp. 72–77, Dec. 2019

[23] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in International Conference on Computer Communications (INFOCOM). IEEE, Mar. 2012, pp. 2731–273

[24] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," IEEE access, vol. 6, pp. 6900–6919, Nov. 2017.

[25] Konečný, Jakub, et al. "Federated optimization: Distributed machine learning for on-device intelligence." arXiv preprint arXiv:1610.02527 (2016).

[26] M. Karaca, T. Alpcan, and O. Ercetin, "Smart scheduling and feedback allocation over non-stationary wireless channels," in 2012 IEEE International Conference on Communications (ICC), (Ottawa, Canada), pp. 6586–6590, IEEE, June 2012

[27] Castro, Inma T., Luis Landesa, and Alberto Serna. "Modeling the energy harvested by an RF energy harvesting system using gamma processes." *Mathematical Problems in Engineering* 2019 (2019).

[28] Wadu, Madhusanka Manimel, Sumudu Samarakoon, and Mehdi Bennis. "Joint Client Scheduling and Resource Allocation under Channel Uncertainty in Federated Learning." IEEE Transactions on Communications (2021).