Mobile and wireless communications Enablers for the Twenty-twenty Information Society-II

Deliverable D7.3
Final 5G visualization

Version: v1.0

2017-06-30

http://www.5g-ppp.eu/

Deliverable D7.3
Final 5G visualization

Grant Agreement Number:	671680
Project Name:	Mobile and wireless communications Enablers for the Twenty-twenty Information Society-II
Project Acronym:	METIS-II
Document Number:	METIS-II/D7.3
Document Title:	Final 5G visualization
Version:	v1.0
Delivery Date:	2017-06-30
Editor(s):	Nandish P. Kuruvatti, Jose F. Monserrat, Andrzej Szczygiel
Authors:	Gary Huang, Industrial Technology Research Institute Jakob Belschner, Deutsche Telekom Andrzej Szczygiel, Janmedia Mauro Boldi, Telecom Italia David Garcia-Roger, Carlos Herranz, David Martín-Sacristán, Jose F. Monserrat, and Danaisy Prado, Universitat Politècnica de València Nandish P. Kuruvatti, Ji Lianghai, University of Kaiserslautern
Keywords:	Visualization, Unity 3D, Demonstration
Status:	Final
Dissemination level:	Public

Abstract

This deliverable summarizes the final status of the open source visualization platform, which will be used to showcase all key 5G RAN design concepts investigated in METIS-II, and document its usage at various external events.

Revision History

Revision	Date	Description
0.1	2017-02-02	ToC agreed
0.2	2017-05-21	First draft version ready for internal review
0.3	2017-05-28	Draft version after internal review. Ready for external review
1.0	2017-06-20	Final version after revision from the external reviewers

Executive summary

In a time when no one knew exactly what 5G is, METIS-II contribution towards consensus on how to understand this technology and its scope was fundamental. In this task, the visualization tool based on Unity and developed in METIS-II is a fundamental piece to build this consensus. It allows all parties to have a common understanding about 5G and to educate the end users on what they can expect from mobile networks in 2020.

The work on this tool closes with this deliverable in its last version (more than 135 different releases), which is symptomatic of all the effort dedicated by METIS-II in not only defining what to say, but above all how to say it to make the technology understandable by the general public. It is not only a game, this has also become a very powerful interface to represent, even in real time, the results of powerful simulators (up to 5 of them have been integrated into the tool) in order to make a better evaluation of the 5G system performance.

This deliverable summarises the final status of the open source visualization platform, which has been used to showcase all key 5G RAN design concepts investigated in METIS-II. Apart from describing the sections of its version 135 and how to operate with it, this deliverable also includes all details needed for others to use this tool, to interface with it and to integrate new features into this open source visualization tool.

Details about accessing the code, setting up the menus, interacting with other tools via system sockets or FTP, or logical information to be superimposed on the scene are included. It also explains the use of the tool and its impact on different fairs and events, in order to learn about those strategies that have yielded better results in terms of the transmission of information.

Contents

List of Abbreviations and Acronyms

3GPP	Third Generation Partnership Project
5G-PPP	5G Private Public Partnership
AIV	Air Interface Variant
BBU	BaseBand Unit
BS	Base Station
CoMP	Coordinated Multi-Point
CP	Control Plane
C-RAN	Cloud/Centralized Radio Access Network
CSI	Channel State Information
D2D	Device to Device
E2E	End to End
EuCNC	European Conference on Networks and Communications
GUI	Graphical User Interface
ICT	Information and Communication Technology
ITU	International Telecommunication Union
KPI	Key Performance Indicator
LTE	Long Term Evolution
LTE-A	LTE Advanced
METIS	Mobile and wireless communications Enablers for Twenty-twenty (2020) Information Society
mMTC	massive Machine Type Communication
MWC	Mobile World Congress
NAT	Network Address Translation
NF	Network Function
OTT	One Trip Time
RAN	Radio Access Network
RM	Resource Management
RRM	Radio Resource Management
RRU	Remote Radio Unit
RTT	Round Trip Time
SLA	Service Level Agreement
UE	User Equipment
uMTC	ultra-reliable Machine Type Communication
UP	User Plane
URLLC	Ultra Reliable and/or Low Latency Communication
V2V	Vehicle to Vehicle
VoIP	Voice over IP
VP	Visualization Platform
WP	Work Package
xMBB	eXtreme Mobile BroadBand
XML	Extensible Markup Language

1 Introduction

In METIS-II [MII-WEB], one of the key objectives is to enable the 5G concepts to reach and convince decision makers from non-ICT industries [MII16-D22], [MII16-WP] and [MDB+16]. Thus, it is necessary to have easy-to-understand illustrations of envisioned 5G use cases and proposed technical solutions, targeting also the non-experts. A professional 3D visualization tool (METIS-II Visualization Platform) has been introduced by METIS-II to allow viewers to interact with 5G enabled scenarios [MET13-D11], [MET15-D15] and [MII16-D11], while permitting simulation driven data to be intuitively evaluated. The main functions of this tool and its data processing features are the points discussed along this deliverable. It will also describe the potential future evolution and development directions of the METIS-II Visualization Platform.

This report can also be considered as the user manual of the METIS-II Visualization Platform. It is intended for speakers who want to use the last version of the Platform for demonstrations and concept discussion, but also for developers or researchers who want to integrate their research output into the tool. In any case, the complete Visualization Platform together with example simulators binaries and source code are available in the repositories listed in this deliverable.

This deliverable also shows the experience of using the METIS-II Visualization Platform throughout the different events in which it has been presented, including the Barcelona MWC 2016 and the last EuCNC in Oulu, held in June 2017. The virtual reality, Android and desktop versions have been complemented to perfection, offering each of them a unique experience better adapted to each type of visitor.

1.1 Objectives of the document

The objective of this document is to describe the final METIS-II Visualization Platform developed and used in METIS-II. It gives an overview about the key concepts described in the last version of the METIS-II Visualization Platform, including use cases, Air Interface Variants (AIVs) harmonization solutions, network slicing concepts and 5G architecture. Further, the document discusses the implementation of the evaluation/visualization concept, highlighting the users and interaction paradigms taken into account in the project. From static material to a full interactive experience with a system simulation tool running in a remote terminal, the platform allow for a dynamic configuration of the show. Annexes compile a detailed documentation of the user manual for the integration of new functions.

The deliverable also summarizes the information presented in the demonstrations made by the project until the last event in June 2017 right before the delivery of this document. Then, we provide guidelines required to access the platform for people outside the project. Finally, and with the lessons learnt, this deliverable provides the outlook of the future development in this research line beyond the project end.

1.2 Structure of the document

This document is structured as follows: Section 2 deals with the description of the key 5G design concepts included in the last version of the METIS-II Visualization Platform. Section 3 explains the configuration and interaction with the platform. Different alternatives are contemplated depending on the level of requested interactivity. How to set up the scene, configure the menus and integrate simulation tools for visualizing real time results are explained also in Section 3. Section 4 deals with the implementation aspects of evaluation/visualization in METIS-II, showing the details of the demonstrations made along the project and describing the Android and virtual reality versions of the platform. Section 5 deals with access to the METIS-II Visualization Platform, its benefits, and how to download all the available material. A summary is detailed in Section 6, with the main conclusions and future research lines. Finally, the official developer manual of the METIS-II Visualization Platform is provided in the annexes.

2 Key 5G RAN design concepts visualized

One of the main purposes of METIS-II Visualization Platform is to communicate to generic audiences the 5G RAN design concepts. In the Project's structure these concepts are grouped into Key Innovation Pillars and are tied to consecutive work packages (WPs) [MII16-D22]. Even in the earlier stages of the Visualization Platform development, the Key Innovation Pillars driven approach was already present, and the use cases that were investigated and evaluated in the platform were based on this logic. At the final stage, when the direct interaction with simulators was introduced, these pillars are not only considered in the use case scenario origins, but also sets of visual and data driven elements and examples on which the future investigation into the areas of interest can be initiated. In the figure below, the different parts included in the Visualization Platform are described. The next sections deal with the description of each of those parts.

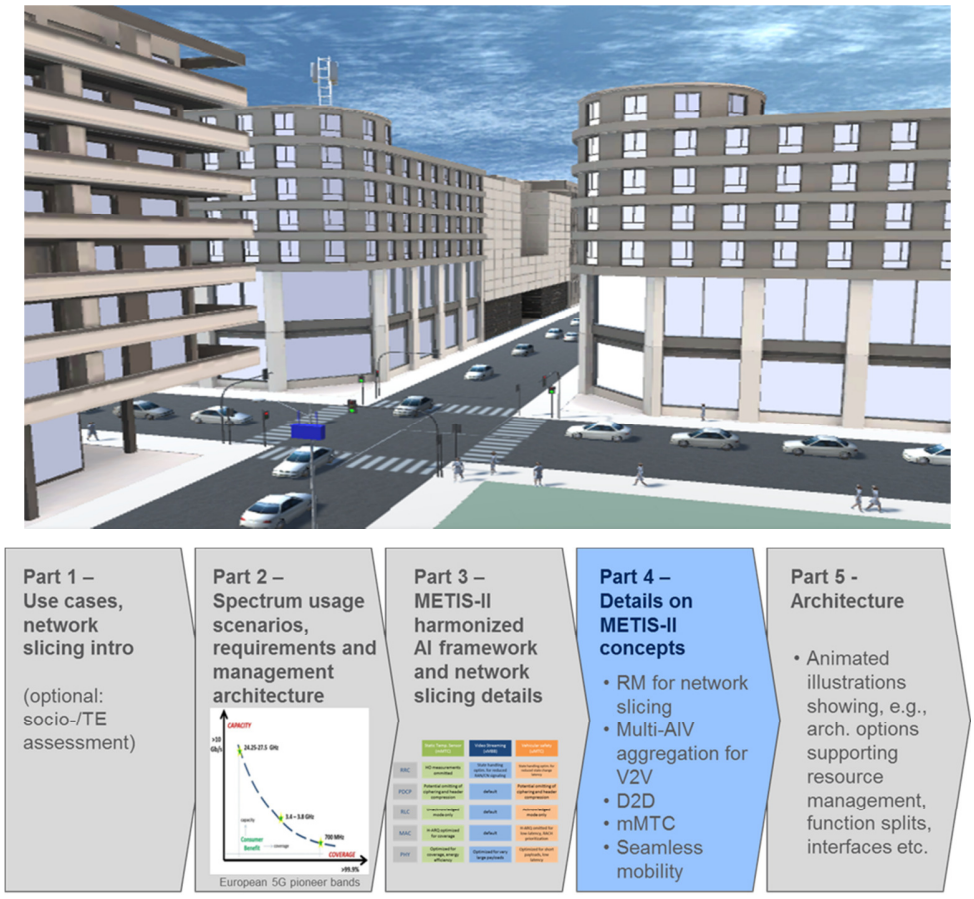


Figure 2-1: The Visualization Platform and the different parts visualized therein.

2.1 Use cases and network slicing introduction

This demo part introduces the audience to the main service types extreme Mobile BroadBand (xMBB, a.k.a. eMBB), ultra-reliable Machine Type Communication (uMTC, a.k.a. Ultra Reliable and/or Low Latency Communication (URLLC)) and massive Machine Type Communication (mMTC), and introduces also the concept of network slicing.

The message that is conveyed by this part of the demo is that METIS-II has developed an overall 5G RAN design able to accommodate a wide range of services with diverse requirements, as foreseen for the 5G era. In particular, it has developed a clear view on how different business interests would be supported through network slicing, and which exact implication on the RAN design this will have.

The services/Network Slices section of the visualization demo introduces viewer into different illustrations of service types (xMBB, URLCC, mMTC). Figure 2-2 depicts an example of user (pedestrian) wearing a virtual reality device and availing high data rate xMBB service.

Figure 2-2: Example of xMBB service Virtual Reality.

Figure 2-3 shows another example of xMBB service, where infotainment video is being availed by a user in car (high mobility).

Figure 2-3: Example of xMBB service Video.

Figure 2-4 depicts an example of URLLC service, where cars communicate using highly reliable and low latency links to exchange critical information to ensure traffic safety and accident avoidance.

In Figure 2-5 a scenario of mMTC is shown, where large number of pedestrians in the network have wearable devices with sensors (e.g., on body health sensors) communicating with network. The overall service illustration is given in Figure 2-6.

Further, the demo illustrates various options for network slicing. Figure 2-7 illustrates an example where each service type is supported by a dedicated slice. Safety critical URLCC information, measurements from health sensors (mMTC) and 4K video (xMBB) are supported by dedicated slices.

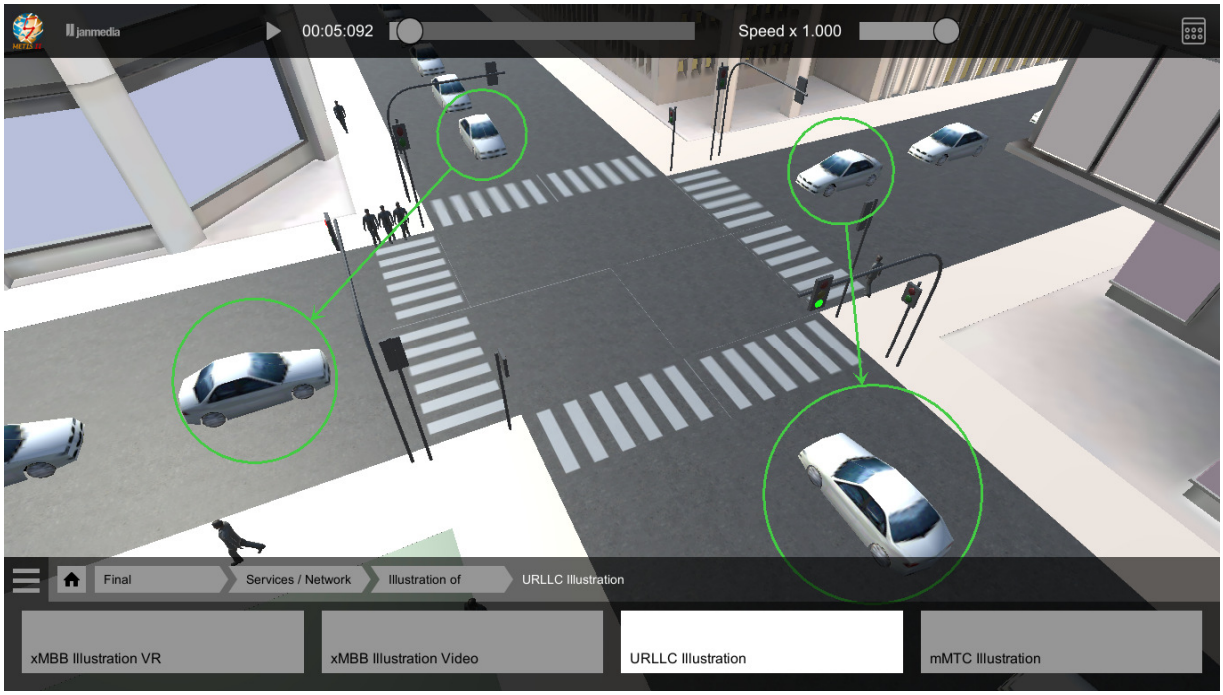


Figure 2-4: URLLC scenario for traffic safety.

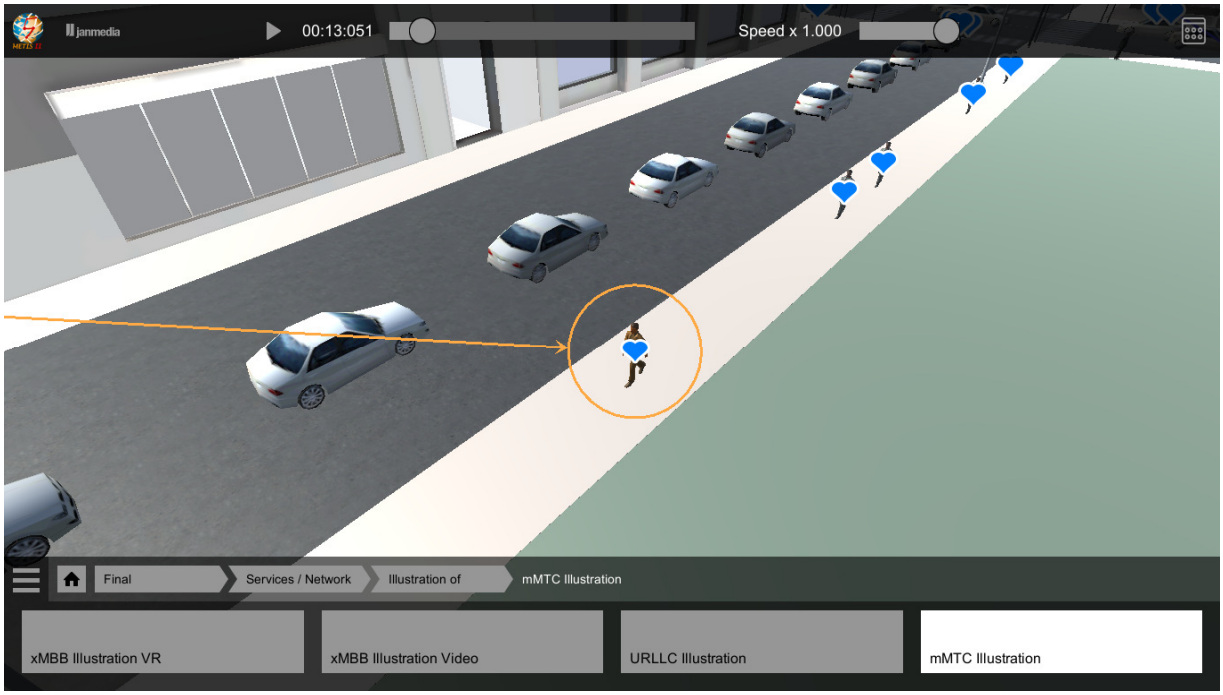


Figure 2-5: Example of mMTC scenario.

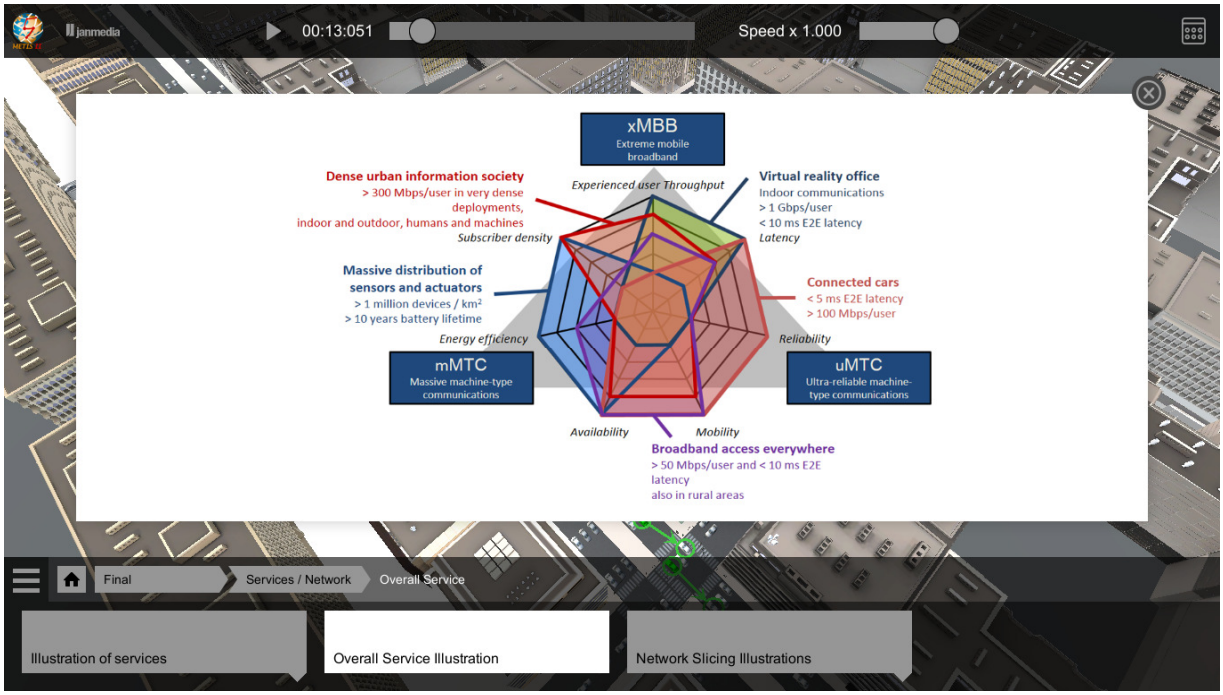


Figure 2-6: Overall service illustration.

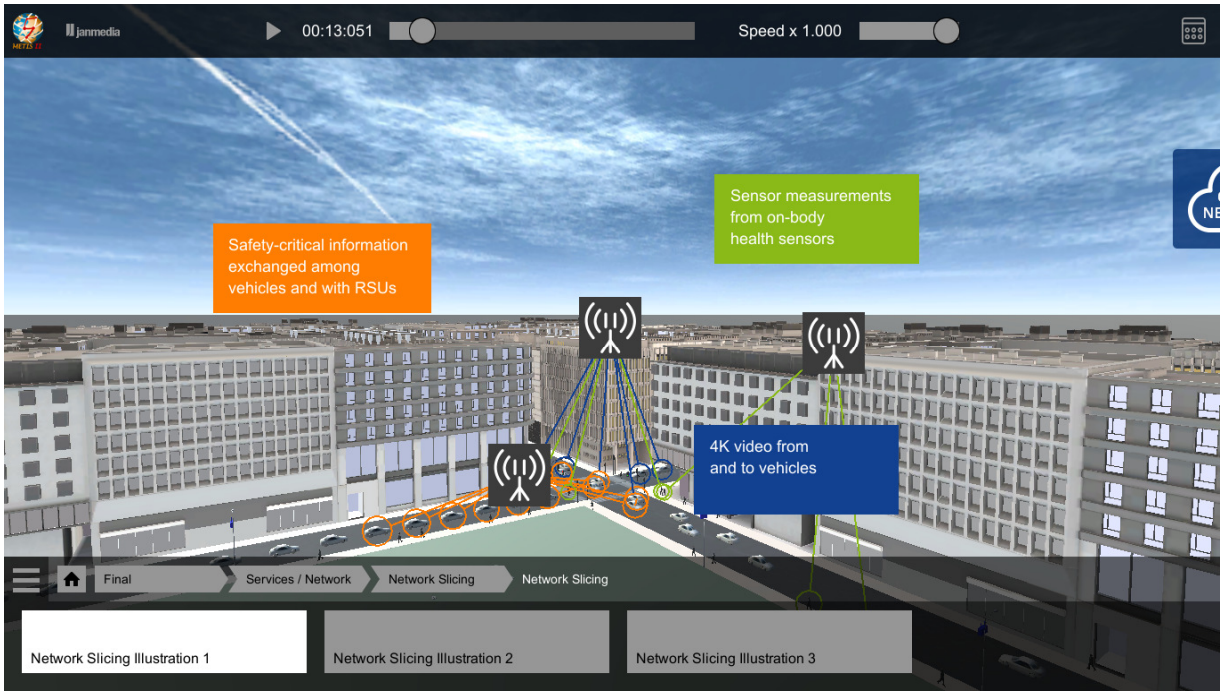


Figure 2-7: Network Slicing illustration 1.

Another network slicing example is illustrated in Figure 2-8, where core network functions are placed either at the edge or remotely, in order to support latency-critical (URLCC) and video traffic (xMBB) respectively.

Figure 2-8: Network Slicing illustration 2.

Figure 2-9 illustrates another network slicing illustration showing application residing in cloud and low frequency AIV being used for larger coverage (mMTC service).

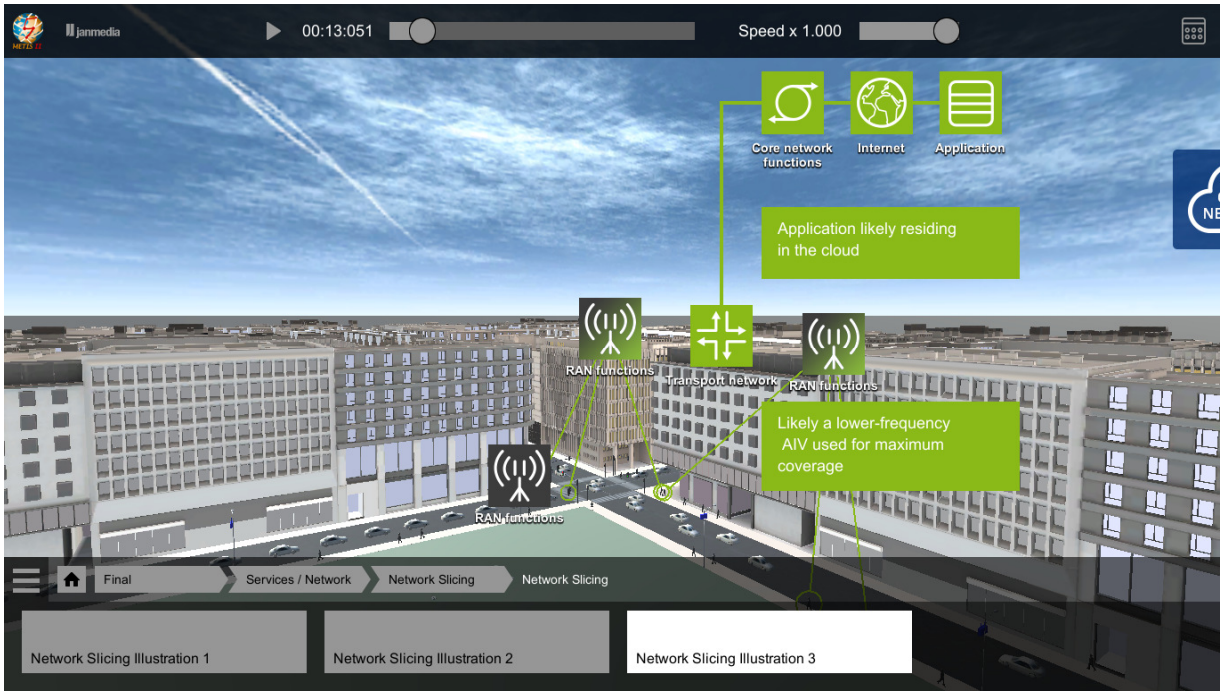
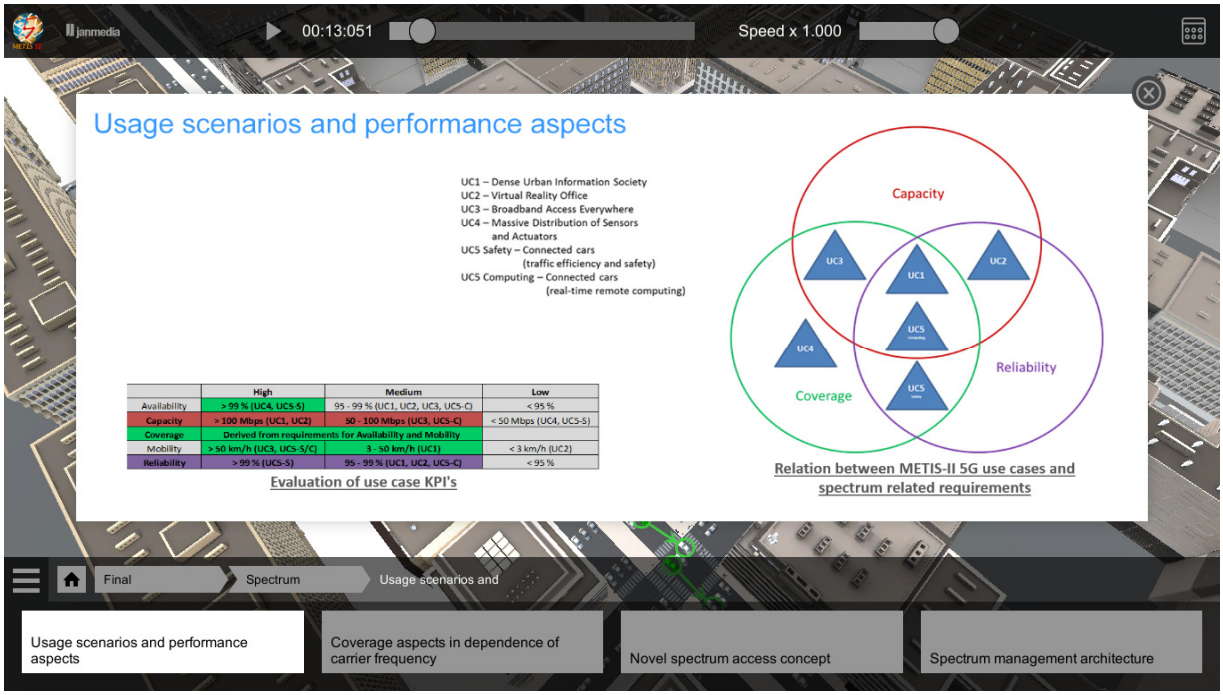


Figure 2-9: Network Slicing illustration 3.

2.2 Spectrum usage scenarios, requirements and management architecture

The key messages exposed in this part are that the 5G success depends on the access to contiguous, wide and globally harmonized new frequency bands [MII17-D32]. Moreover, this part highlights the spectrum rationale, that is, the usage scenarios and the coverage aspects in dependence of carrier frequency. It also considers enablers and scenarios for future spectrum usage: novel dynamic spectrum access concept and spectrum management architecture.

Figure 2-10 illustrates an example in the tool, including the usage scenarios, performance aspects, relation between use cases and spectrum related requirements.



	High	Medium	Low
Availability	> 99 % (UC4, UC5-S)	95 - 99 % (UC1, UC2, UC3, UC5-C)	< 95 %
Capacity	> 100 Mbps (UC1, UC2)	50 - 100 Mbps (UC3, UC5-C)	< 50 Mbps (UC4, UC5-S)
Coverage	Derived from requirements for Availability and Mobility		
Mobility	> 50 km/h (UC3, UC5-S/C)	3 - 50 km/h (UC1)	< 3 km/h (UC2)
Reliability	> 99 % (UC5-S)	95 - 99 % (UC1, UC2, UC5-C)	< 95 %

Figure 2-10: Spectrum requirements for 5G use cases.

2.3 Harmonized AI Framework and network slicing details

In this part the user can access the various slices as it has already been introduced in part 1. The user gets familiar with the notions of slicing, the RAN configuration modes for meeting the services requirements, and the various AIVs as they were studied in the METIS-II project [MII17-D42]. This also enables the user to get used to with how slicing and services can be linked.

Related to AIVs, METIS-II developed an overall AI framework for all bands and services. A key aspect in this work has been AIVs harmonization, which has benefits in terms of reduced standardization / implementation complexity and reduced latency, as illustrated in this demo part.

Figure 2-11 shows the part of demo tool for harmonized AI and network slicing details. This part gives an overview of METIS-II AI considerations, motivating the need for harmonization among different AIVs. Further, the benefits of AIV harmonization in terms of complexity and chip area savings are highlighted. Similarly, network slicing details are summarized in Figure 2-12, showing each network slice tailored for a specific service type (xMBB, uMTC, mMTC) and functions in each layer designed to meet the respective service requirements.

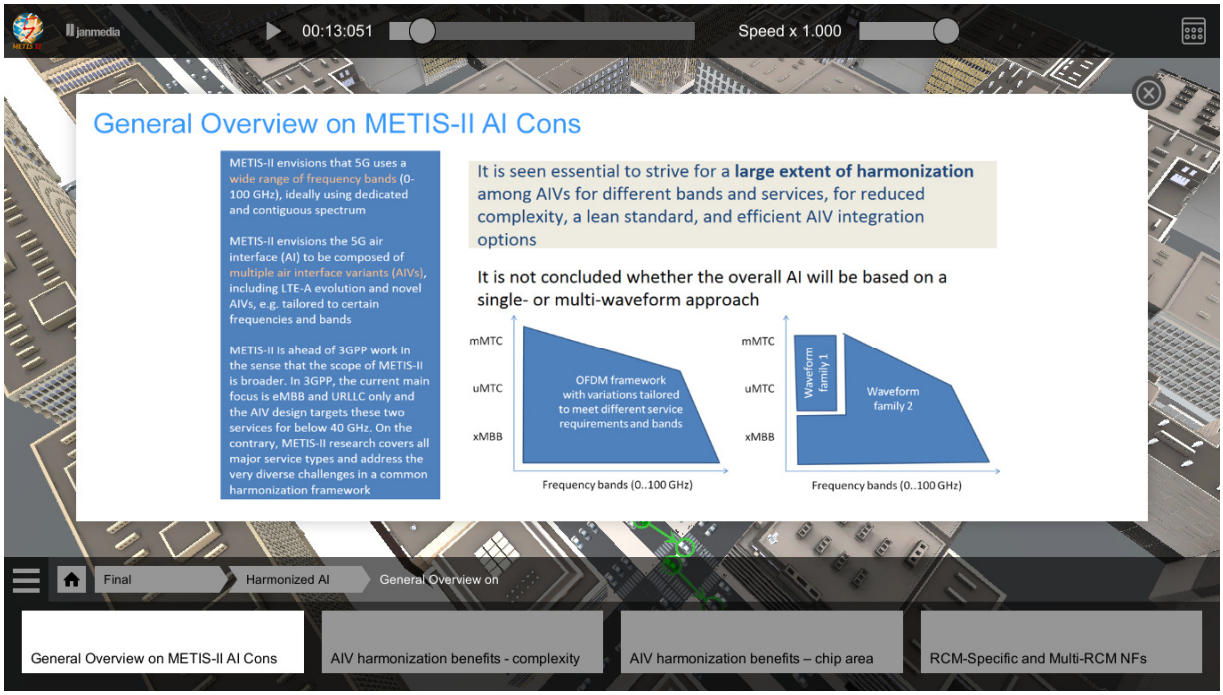


Figure 2-11: Harmonized AI details.

Figure 2-12: Network slicing details.

2.4 Detailed demonstrations of METIS-II concepts

This section is devoted to the explanation of the METIS-II concepts based on demonstrations illustrated in the Visualization Platform. Five demonstrations are included in the Visualization Platform, some of which interact, in real time, with simulation tools running in distant servers.

2.4.1 Multi-slice resource management

As part of a real-time simulation connected to the Visualization Platform, users can experience the developed approach for a radio resource management that is aware of network slices, the so called Multi-slice Resource Management (RM). Network slices can be configured based on customized parameters, especially the Service Level Agreement (SLA), provided by the user. In addition, the amount of data traffic in the system can be configured and changed by the user. Based on these inputs, the resulting performance, in comparison to the SLA, is displayed, as depicted in Figure 2-13. For more details on the RM approach itself, see [MII17-D52].

Figure 2-13: Real-time simulation of Multi-slice Resource Management.

2.4.2 Multi AIV aggregation for V2V communication

Autonomous driving is expected to be a reality in the following years and without any doubt it will change how transportation is understood nowadays. The criticality of the communication path between vehicles and/or the infrastructure however will impose severe restrictions in terms of reliability and latency. For this reason, 5G is envisioned to play a prominent role as an enabling technology of the autonomous driving.

METIS-II consortium partners envisioned a use case to demonstrate how 5G could enable the autonomous driving. More precisely, we rely on a promising feature of 5G technology: the AIV

harmonization, which allows seamless switching among air interfaces. In our case, we focused on switching the operation frequency only, and transmissions can be on 5.9 GHz or 28 GHz bands.

The baseline model is an urban scenario, i.e. Madrid grid, in which each vehicle transmits a single Cooperative Awareness Message (CAM) every 10 ms. In order to simplify the simulations, all CAMs are assumed to be of 1600 bytes. Transmissions are made in the 5.9 GHz band, with a limited system bandwidth of 10 MHz. In dense traffic conditions, the baseline model is proved to be unfeasible for CAM delivery; a given transmit car is unable to deliver CAMs to the neighbouring vehicles due to the limited bandwidth which is quickly saturated. Such scenario is depicted in Figure 2-14, in which the red arrows coming from the transmitting car depict erroneous transmission of CAM to the pointed vehicles.

To overcome the bandwidth limitation, a possibility is to deploy a carrier frequency in the millimetre wave (mmW) regime, which is plenty of free bandwidth to exploit. As Figure 2-15 shows, CAMs are delivered to a few vehicles, just to the very close neighbouring vehicles. However, CAMs are not delivered further given poor propagation conditions and blockage situations. So, METIS-II partners advocate for CAM transmissions in low and high frequencies; one out of ten transmissions will be in the low frequency band in order to expand the successful delivery range, as it can be seen in Figure 2-16.

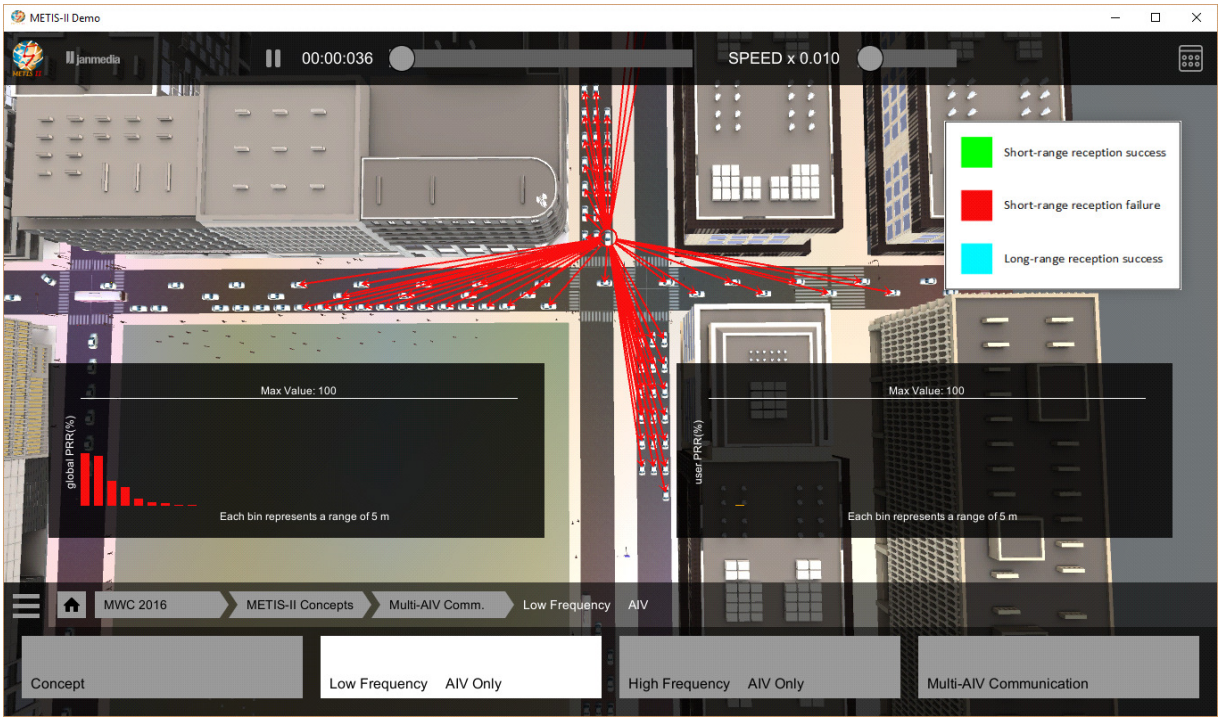


Figure 2-14: CAM transmission over 5.9 GHz.

Figure 2-15: CAM transmission over 28 GHz.

Figure 2-16: Alternate CAM transmission over 5.9 and 28 GHz.

2.4.3 Exploitation of D2D communication to enhance mMTC services

Billions of mMTC devices are foreseen to be deployed in next ten years and therefore potentially open a new market for next generation wireless network. However, mMTC applications have different characteristics and requirements compared with the services provided by legacy cellular networks. For instance, an mMTC device sporadically requires transmitting a small data packet containing information generated by sensors. At the same time, due to the massive deployment of mMTC devices, it is inefficient to charge their batteries manually and thus a long battery life is required for mMTC devices. In this sense, legacy networks designed to serve human-driven traffics in real time cannot support mMTC efficiently. In order to improve the availability and battery life of mMTC devices, context-aware device-to-device (D2D) communication can be exploited [LLS17]. By applying D2D communication, some mMTC users can serve as relays for other users who experience bad channel conditions.

In this scheme, UEs can be configured with one of the three different transmission modes, as following:

- cellular transmission mode, in which the devices upload their reports to base stations (BSs) with cellular links;
- relay transmission mode, in which the devices are configured by the network to relay the user plane date packets from remote user equipment (UEs) and meanwhile transmit their own reports to BS;
- D2D transmission mode, in which the remote UEs transmit their reports to relay UEs.

From efficiency point of view, D2D communication should be applied in cases where transmitter and receiver are nearby each other. Thus, a clustering approach is required at BS to make sure that a relay UE only serves remote UEs in its proximity. Afterwards, BS needs to dynamically select proper transmission mode for each UE, taking into account of the context information (e.g., channel state information (CSI) between BS and the UE, location and battery level information). Therefore, the proposed D2D communication can be divided into two steps:

- clustering devices into different groups;
- dynamic selection of transmission mode for each UE.

In order to evaluate our proposed scheme, a system level simulator aligned with the real world is used in this work. And the key performance indicators to be evaluated are the service availability and battery life of mMTC sensors. One thousand sensors are statistically deployed inside one building and the user information (e.g. positions and colours) of these sensors are generated by the simulator and written into a binary file introduced in the Annexes. The overall sensors are categorized into three different classes and shown by different colours. A sensor with white colour represents a sensor which can be connected to the BS and have a battery life longer than the 10 years requirement, by using cellular uplink. A black colour shows the sensors which can be connected to the BS but cannot meet the battery life requirement by using cellular

uplink. Last but not least, a blue colour is used to show the sensors which are out of the cellular uplink coverage. The plot of sensors with different colours can be done by using another binary file introduced in Annexes.

Figure 2-17: Unity to showcase the exploitation of D2D to enhance mMTC.

A snapshot of the scene from the visualization platform is provided in Figure 2-17. As can be seen in this example, three clusters are applied and thus maximally three sensors can act as the relays at the same time for the remote sensors who seek for assistance. For illustration of the different transmission modes, grey lines are used here to show the D2D link from remote UEs to their relays and green lines represent the links between the base station and the relay sensors. The binary file introduced in the Annexes is required to plot the corresponding lines here. For the two ends of one D2D link, the sensor acting as the relay point is tied with a blue sphere around it and the remote sensors are tied with black spheres. Moreover, for the sensors which stay in the cellular mode, they are tired with light green spheres and no line is used to represent the link to the BS, since there are potentially a large number of these sensors. To visualize the performance, the percentage of the served users is plotted at the right bottom, as a global key performance indicator (KPI). With this global KPI, we can see the percentage of the mMTC users that can be served by using only the cellular uplink and using both the cellular uplink and the proposed D2D communication, respectively, at the different time instant. The binary file used to illustrate this global KPI is detailed in the Annexes. Since it is assumed that a ten years battery life requirement is targeted for the mMTC service, it is necessary to limit the duration of the show. Therefore, we use a time duration of one second in the visualization platform to show the case of one hundred days in real world. In this case, totally $3650(\text{days})/$

$100(\text{days per second}) \approx 36\ \text{seconds}$ are displayed in the visualization platform and the system performance can be observed in this duration.

2.4.4 D2D mobility management framework

In vehicle-to-vehicle (V2V) communication scenarios, mobility impacts the reliability of D2D links. Consider the following scenario in a legacy network (i.e., LTE or LTE-A): two vehicles, connected via D2D and moving in the same direction while keeping a distance apart from each other, hand over their control plane (CP) connections to a common target BS one after the other at different moments. During the time between the first handover and the second handover communicating vehicles remain in different cells and D2D communication between the vehicles in the user plane (UP) is disrupted, e.g., due to different D2D resource allocations from different eNBs. Therefore, data packets are dropped during the disruption time. Adapting multi-connectivity and mobility management framework in 5G D2D as discussed in METIS-II [MII17-D62], possible D2D communication disruption time due to handover can be reduced. Consequently, the disruption impact of an active D2D communication can minimized. This implies an improvement in reliability of the D2D communication.

In the demo, we compare the multi-connectivity D2D mobility management method in METIS II 5G with the legacy mobility management method. The legacy method is the one currently specified in 3GPP LTE or LTE-A network [3GPP16-23401]. In the Madrid Grid [MET13-D61] demonstration environment, 8 BSs (pico cells) are deployed on the top of buildings surrounding grass area (green colour), whereas four of them are located at the four corners and the other four are at the middle of the four edges. The demo shows performance gain in terms of the reliability [MII16-D21], which accounts for the percentage of packets correctly received within the given maximum E2E latency (OTT or RTT depending on the service). Voice over Internet Protocol (VoIP) application based on G.729 [ITUT12-G729] is adopted to measure the reliability of D2D transmission. In addition, the maximum E2E latency (i.e., from D2D sender to D2D receiver) is set to 20 ms for VoIP packet freshness.

Figure 2-18 shows a snapshot of D2D communication with multi-connectivity-enabled D2D mobility management scene in the Unity visualization platform. Graph at the right-bottom corner shows the reliability of the two mobility management methods. The values are reliability for the past one second and thus periodically updated in one second unit. Reliability results are based on Matlab implementation of above mentioned 5G D2D mobility management framework. In Matlab implementation, two devices connect to their respective serving BSs (purple dash lines in Figure 2-18). Also, one of the devices establishes an additional connection (light blue dash line in Figure 2-18) to the serving BS of the other device. Unity takes inputs as data trace, line trace and graph trace files generated by Matlab.

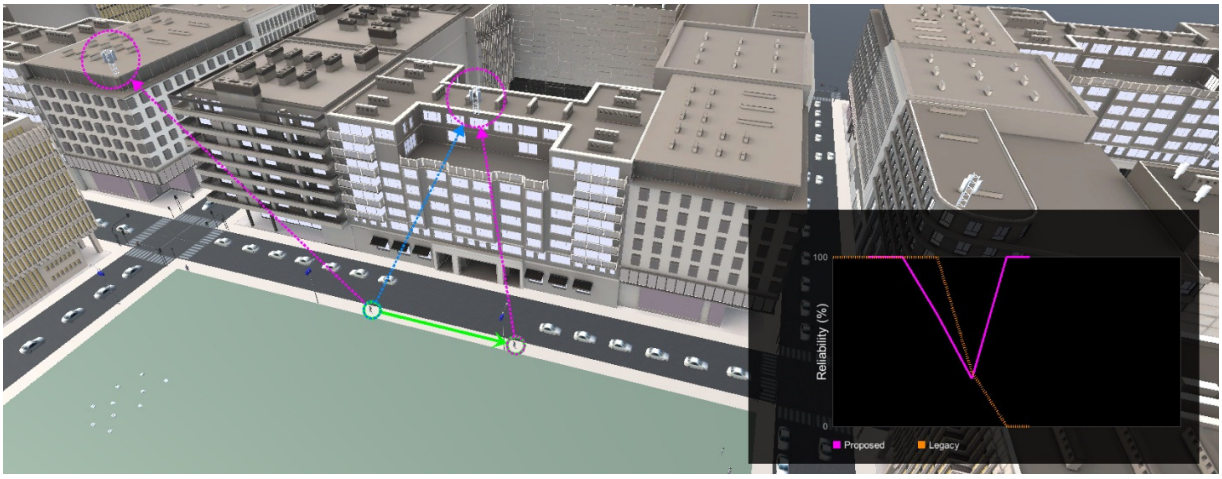


Figure 2-18: Unity Snapshot of integrated D2D mobility management framework implementation in Matlab.

2.4.5 Handover improvements in centralized RAN deployments

Two promising technologies to achieve high robustness and low interruption time for the handover in millimetre wave bands are conditional handover and make-before-break handover. In conditional handover, the target cell is prepared before the actual handover takes place, hence eliminating the possibility of a handover failure due to UE being not able to communicate with the source cell. In make before break handover, the UE maintains the communication link to both source and target cells during the handover, hence eliminating the gap in the data exchange that is caused by the UE accessing the target cell.

Another important enhancement of the 5G will be the centralized RAN, where higher layer protocols reside in a centralized unit (CU) and the lower layer protocols in distributed radio units (DU). The RAN centralization benefits also the UE mobility, stemming from the centralized RRM, context handling, and centralized unit acting as a common anchor point towards the core network.

In the two handover demos, the basic principles of the conditional and make before break handover in are illustrated for a centralized RAN architecture and compared with the baseline (LTE) handover. The demos are based on pre-recorded traces of UE placement, SNR, throughput, and link status.

In the first demo, the UE walks from one distributed unit (DU) to another while making a conditional handover. The fast degradation of the serving cell in the building corner is illustrated by showing the signal to noise ratio (SNR) relative to three DUs seen by the UE, with a marker showing the current SNR. The improved robustness is illustrated by comparing the throughput of conditional handover to legacy (LTE) handover, and noting a long throughput gap in LTE due to radio link failure.

In the second demo, the UE walks from on DU to another while making a make before break handover. The benefit of maintaining connectivity to both source and target cell during handover is illustrated by a throughput plot, make before break handover providing a boost in the cell edge throughput with no interruption in the user data. For the legacy (make before break) handover, there is a gap of couple of tens of milliseconds in data transmission.

The centralized architecture is visible in both demos and can be explained to reduce the signalling associated with candidate cell preparations and path switch.

2.5 System architecture

One important requirement for the 5G RAN is to provide sufficient flexibility for placement of Network Functions (NFs). METIS-II has defined different architecture deployment scenarios [MII16-D22], including also a wireless self-backhauling scenario. It is important to consider how NFs can be split over different physical architecture deployments, and which intra-RAN interfaces between the physical entities would correspondingly be needed.

One key aspect in this respect is the data rate required on the resulting fronthaul interfaces, for instance between a Remote Radio Unit (RRU) at an antenna site and a Baseband Unit (BBU) hosting the full radio protocol stack or upper parts of it in a decentralized or centralized way (cloud/centralized-RAN, C- RAN).

In addition to data rate requirements, also latency aspects are a critical issue for the selection of suitable splits, for instance limiting the implementation of certain functionalities (e.g. coordinated multi-point (CoMP) processing) in the case of some deployment scenarios. A key consideration here is to design 5G RAN functions to avoid strict timing relations between the protocol layers, and to have a clearer split between time-synchronous and time-asynchronous functions.

In this section of the demo we present results of the different split options, analysing its pros and cons and the scenarios in which each option is best suited. Figure 2-19 illustrates the recommended split option for non-ideal xhaul. The recommended split option for case of fiber to edge is shown in Figure 2-20. Figure 2-21 shows recommended split option for local RAN clouds and self-backhauling is illustrated in Figure 2-22.

Figure 2-19: Non-ideal xhaul.

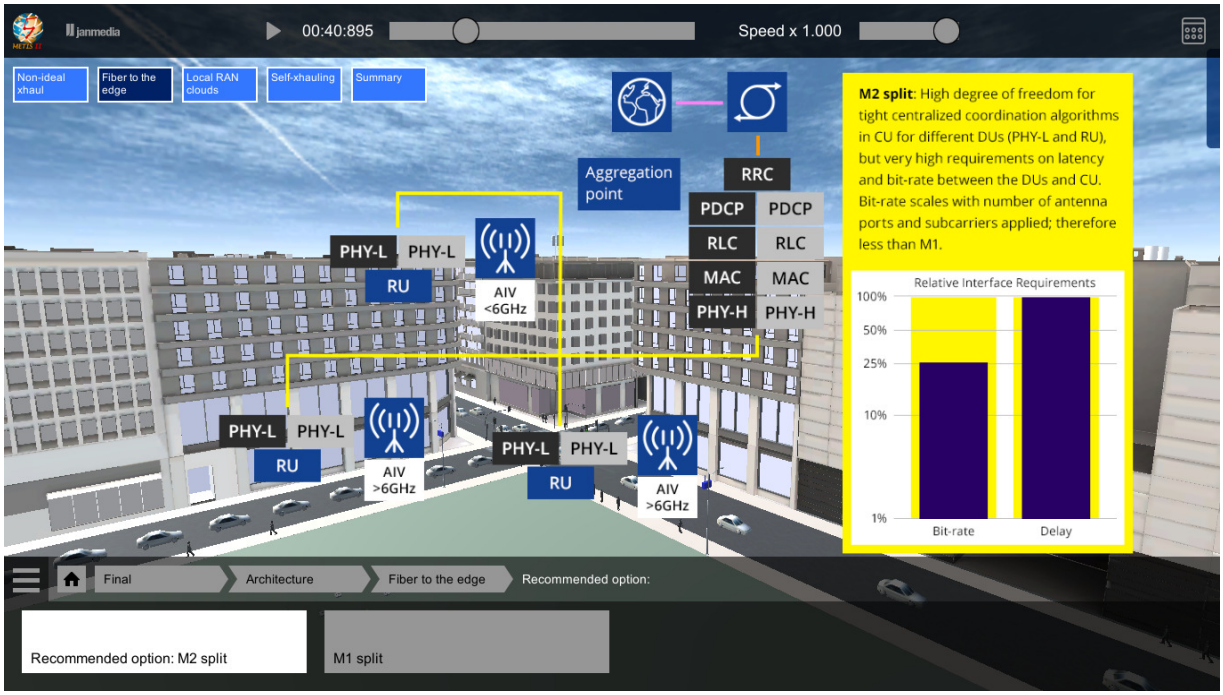


Figure 2-20: Fiber to the edge.

Figure 2-21: Local RAN clouds.

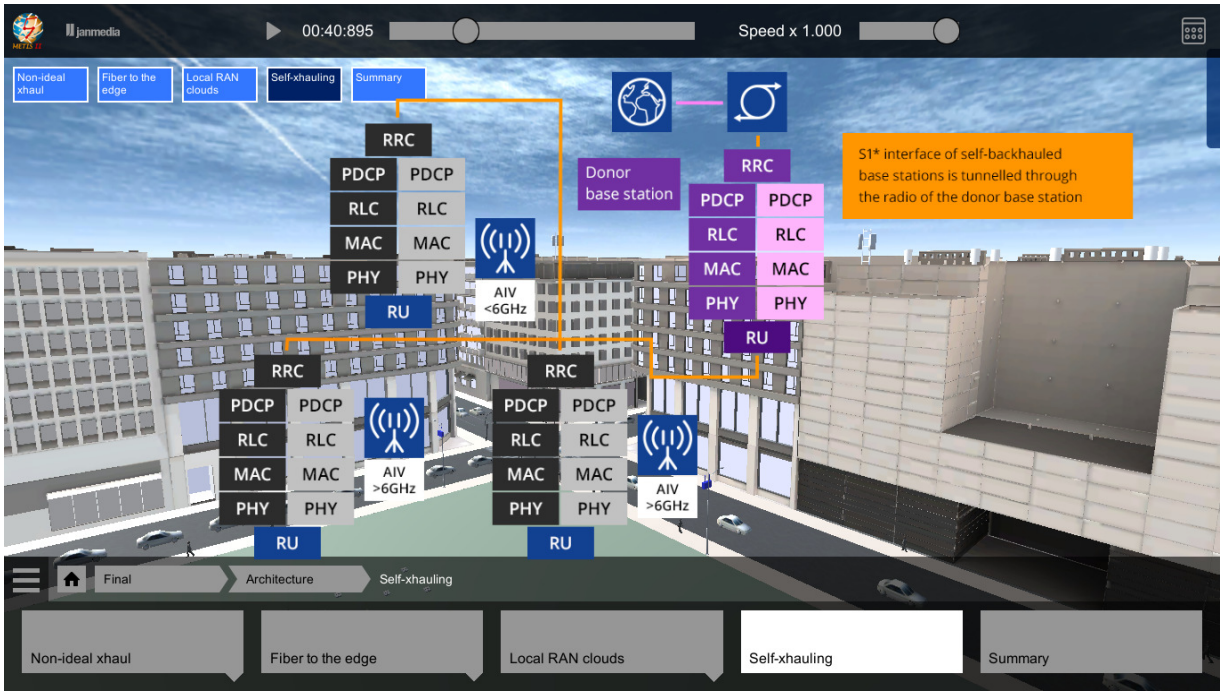


Figure 2-22: Self x-hauling.

3 Possible data exchange configurations in the Visualization Platform

The final build of the Visualization Platform can be configured to read and reflect data from virtually any source. From story driven scenarios built statically in 3D environments to simulator-generated traces visualized in real time, it is possible to build and evaluate visual representations of various concepts tied to the 5G era. It's also possible to establish two-way communication with the data sources, which in effect brings interaction on-board. It is most beneficial in the context of simulator interaction, where decisions made by users can have almost instant influence on the simulation results. This might lead to easily achieved comparison cases enriching evaluation outcome.

3.1 Scripted Unity based visualizations

The most straightforward use case definition mechanism is based on reflecting the required scenario directly in Unity.

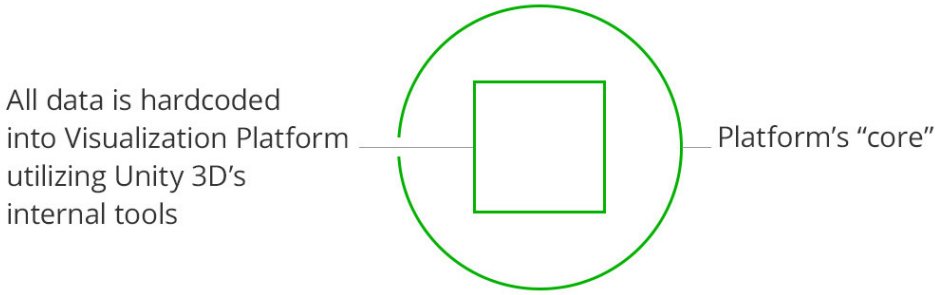


Figure 3-1: Visualization Platform configuration scheme with all data hardcoded in the application.

By creating appropriate 2D and 3D objects (visualization and information layer elements) and animating them on a timeline, it is possible to create demonstrations and presentations. Explanatory visual material like pictures, static graphs, text, etc., can be presented similarly to what is experienced during any standard presentation. In this case the data is hardcoded into Unity engine and compiled into the final application build. The obvious drawback of this approach lies in the need of opening and recompiling the project each time a change in the required scenario flow occurs.

3.2 Hardcoded data defined in XML

The more flexible approach stems from the possibility of separating the data from the Unity application's core. The elements of the presentation are turned into libraries of objects (buildings, UE representations like cars or pedestrians, access points, connection metaphors etc.). These elements are then operated by data saved in the file and can be edited easily. For clear structure communication, .xml format was chosen for these files.

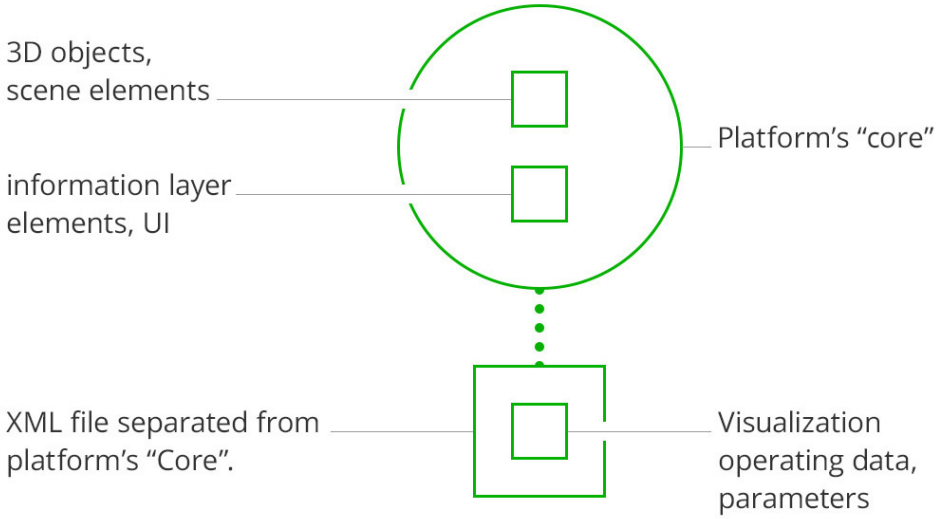


Figure 3-2: Visualization Platform configuration scheme with data hardcoded in xml.

This approach (after being backed up by relevant documentation) eliminates the need for advanced Unity knowledge and allows for more flexible parameter manipulation and use case definition.

3.3 Local files (plus binary format advantages)

It is possible to make the use case definition process even more separated from the Unity application, which in some cases may affect evaluation in a positive way. In this approach the data is located in files stored on the same machine (in theory any file type).

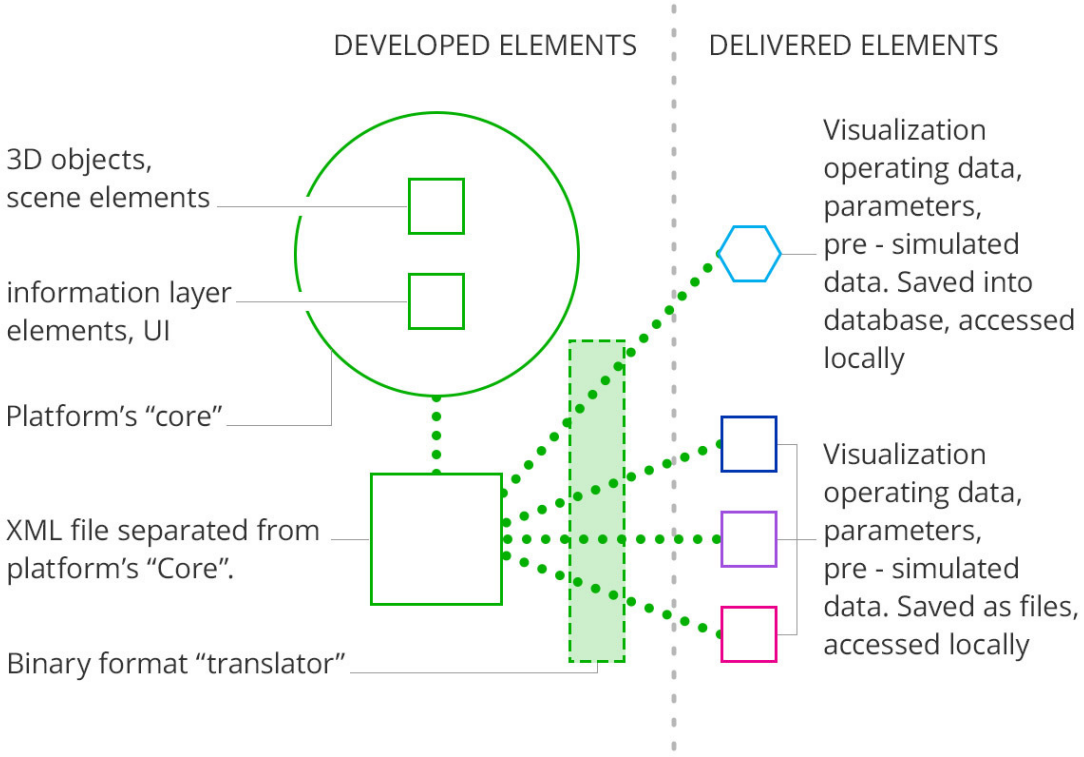


Figure 3-3: Visualization Platform configuration scheme with data in local files.

The access paths are defined within the .xml file presented in Section 3.2 This allows for pre-simulated data to be saved externally and visualized within the platform. As the data source files might have different formats which sometimes resulted in large file sizes, one more step was suggested and introduced: binary format translation. Turning data source files into binary format reduced their sizes by 60-80%, which in sequence made the final solution perform smoother. The possibility of storing data in a local database was also tested with positive results.

3.4 Remotely accessed data located in files / database

The next natural and necessary step entailed accessing data files remotely. Here the decision to compress data files into binary format turned out to be very effective, as it reduced the impact of the connection speed. The initial attempts were made to access the files used for local data storage via standard protocols. After they turned out to be successful, it became clear that all the use cases could be defined via data available online. Similarly, the online database access was tested and returned positive results.

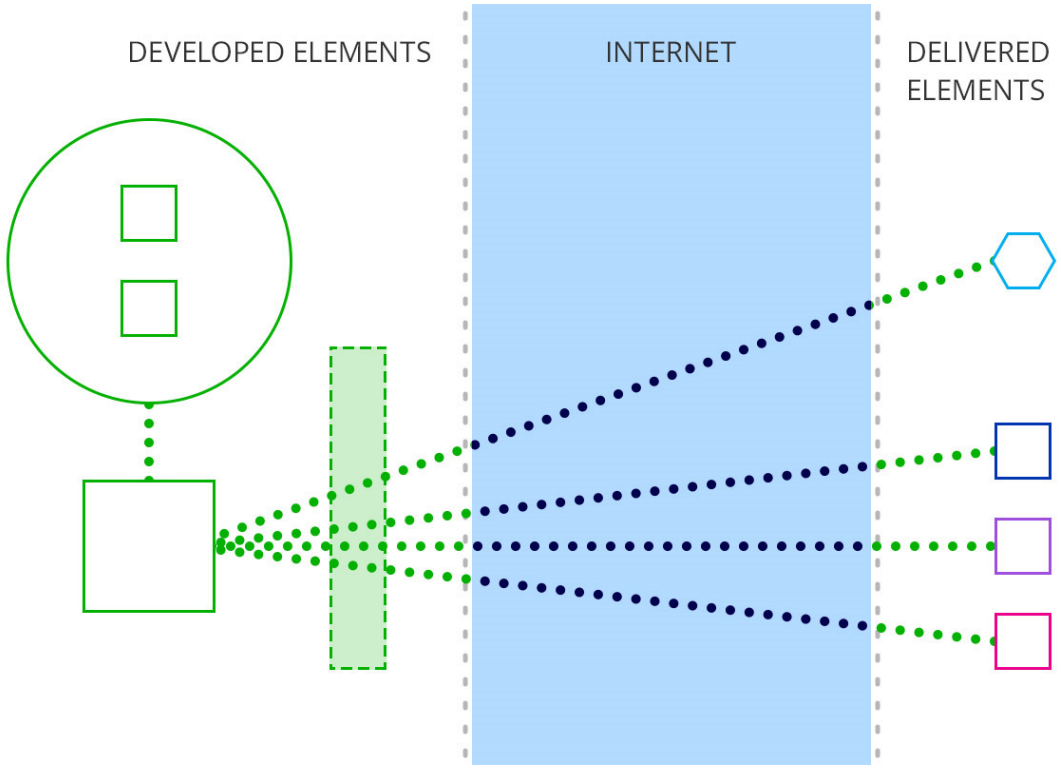


Figure 3-4: Visualization Platform configuration scheme with data in remote files.

This approach could be utilized for more structured data processing, though for METIS-II objectives it was not necessary to put more effort into developing database oriented information exchange solution.

3.5 Dynamic communication and interaction with simulators

Finally it became clear that the most beneficial mechanism from the perspective of the potential Visualization Platform users should allow for direct interactions with simulators. Utilizing socket-based communications, a solution was developed which initially allowed for defined simulator direct data access. Based on the experiences taken from this step, it became imaginable to communicate with any simulator, given that the parameter and syntax synchronization would be possible. At the same time similar solutions could be developed leading to interaction with the simulator in real time, with the only caveat being the simulator's response time and some possible lags stemming from the connection parameters.

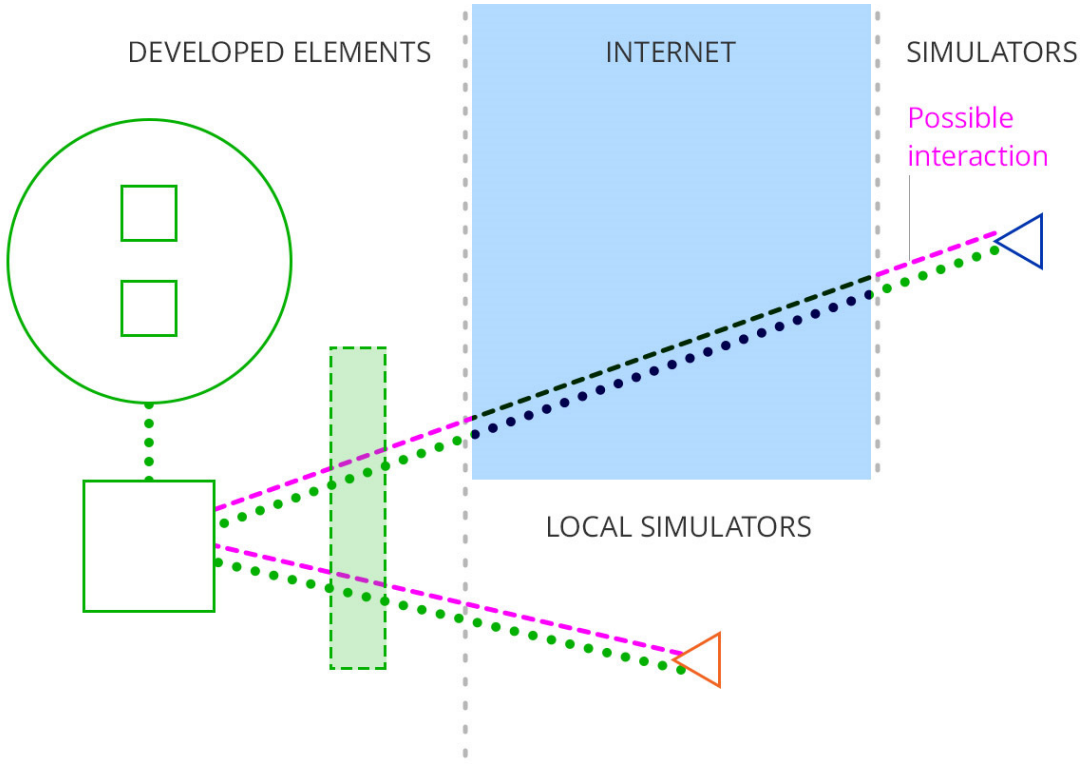


Figure 3-5: Visualization Platform configuration scheme with data exchange from concurrent simulations.

Ultimately, it is already clear that all the use cases could be created from dynamically delivered data. Moreover, with the available documentation it should be possible for virtually any party to develop their own scenarios and evaluate them within the Visualization Platform.

3.6 Mixed configurations

What is most important for METIS-II visualization and evaluation goals stems from the fact that all the approaches mentioned above can be combined to allow maximum flexibility in the use case creation process.

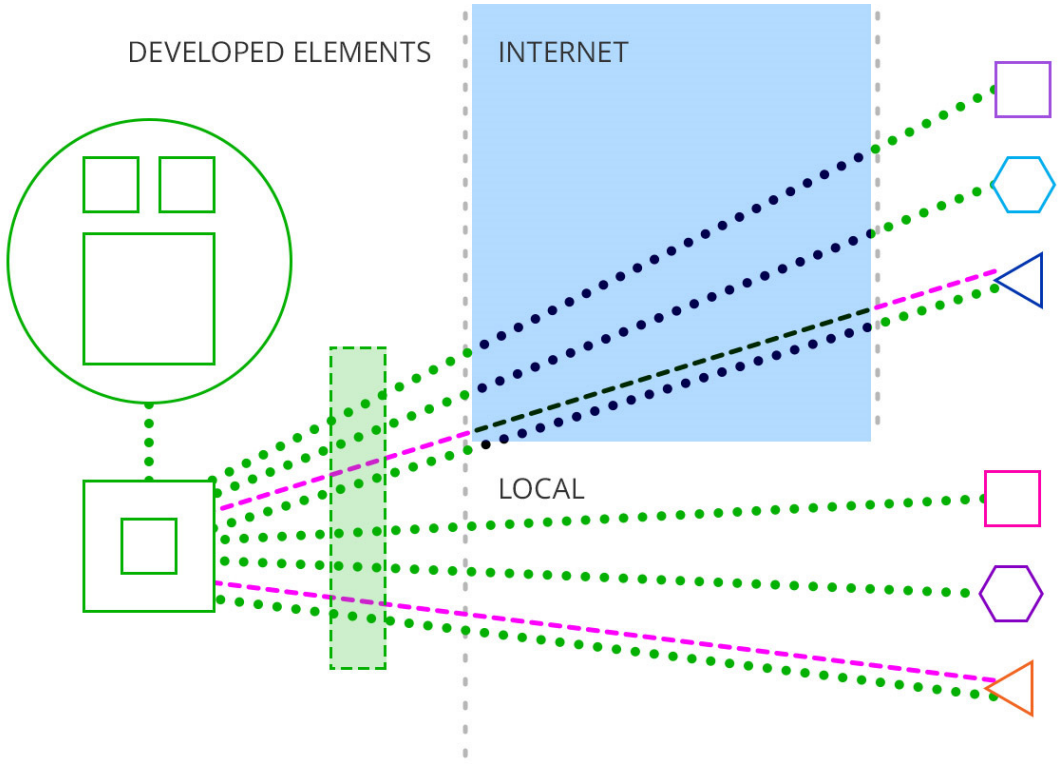


Figure 3-6: Visualization Platform configuration scheme with multiple data sources.

It is possible, for example, to generate simulated traces of interactions between network elements and to enrich them with scripted animations of how they influence the surrounding situation. It is possible to create an animated scenario and enrich it with pre-calculated data. Thanks to this approach, any future findings of 5G based research should, without problems, be visualized within the METIS-II Visualization Platform.

4 Visualization Platform features summary

The data exchange configuration models described in Section 3 were tested and utilized throughout the whole research and development time span of METIS-II. Almost any step taken to develop new features was justified by real needs. This approach allowed the Visualization Platform to have been used already during ICT 2015 conference – roughly 4 months after the project had started.

4.1 Scripted use cases

From the early stages of the project it was important to communicate METIS-II goals and ideas. Without consolidated data output or simulation results it made no sense to work on the data interpretation mechanisms, so it was clear the first visualization effort would be focused on scripted storytelling and concept presentation. The first version of the Visualization Platform was released just before ICT 2015.

4.1.1 ICT 2015

For ICT 2015 the platform already had “Madrid Grid” environment interpreted and implemented in 3D space, where the buildings, cars, pedestrians and other 3D models were created along with the basics of visual communication elements, accompanied by mobility traces used for simulations. This initial setup served as the base on which KPIs of METIS-II were illustrated in the form of scripted 3D representations that reflected state-of-the-art presumptions of relevant research fields explored by WPs.

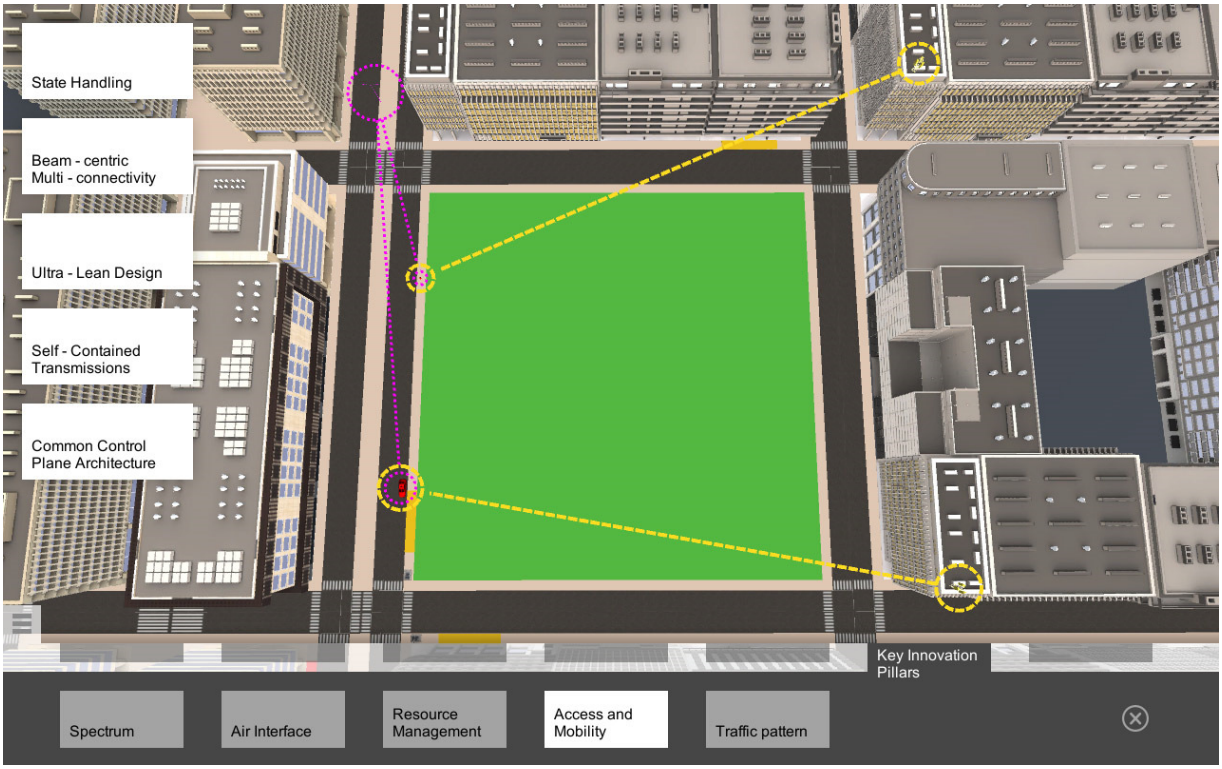


Figure 4-1: Screenshot of ICT 2015 demonstration.

These representations were accompanied by explanatory material, presented in the form of fine-tuned “slides”. This became standard for further iterations whenever the intended content had to be presented in a more traditional way.

4.2 Static data based use cases

As soon as preliminary sets of data started to emerge from simulations conducted by respective WPs, the platform was enriched with first data exchange mechanisms. Different paths were taken into account, but it soon turned out the simulation results intended to be visualized could have been delivered in the form of large files. In addition, the files could have come from different sources, which could have affected the possibility of developing a common data access interface. In that point in time the decision taken was to translate data files into binary format (based on earlier-agreed standards), which reduced the file sizes by 60-80% and became a step toward the universal data exchange mechanism. The second major version of the Visualization Platform was released just before MWC 2016.

4.2.1 MWC 2016

The version of the platform presented during MWC 2016 already utilized traces and data generated by other WPs to visualize concepts researched by METIS-II.

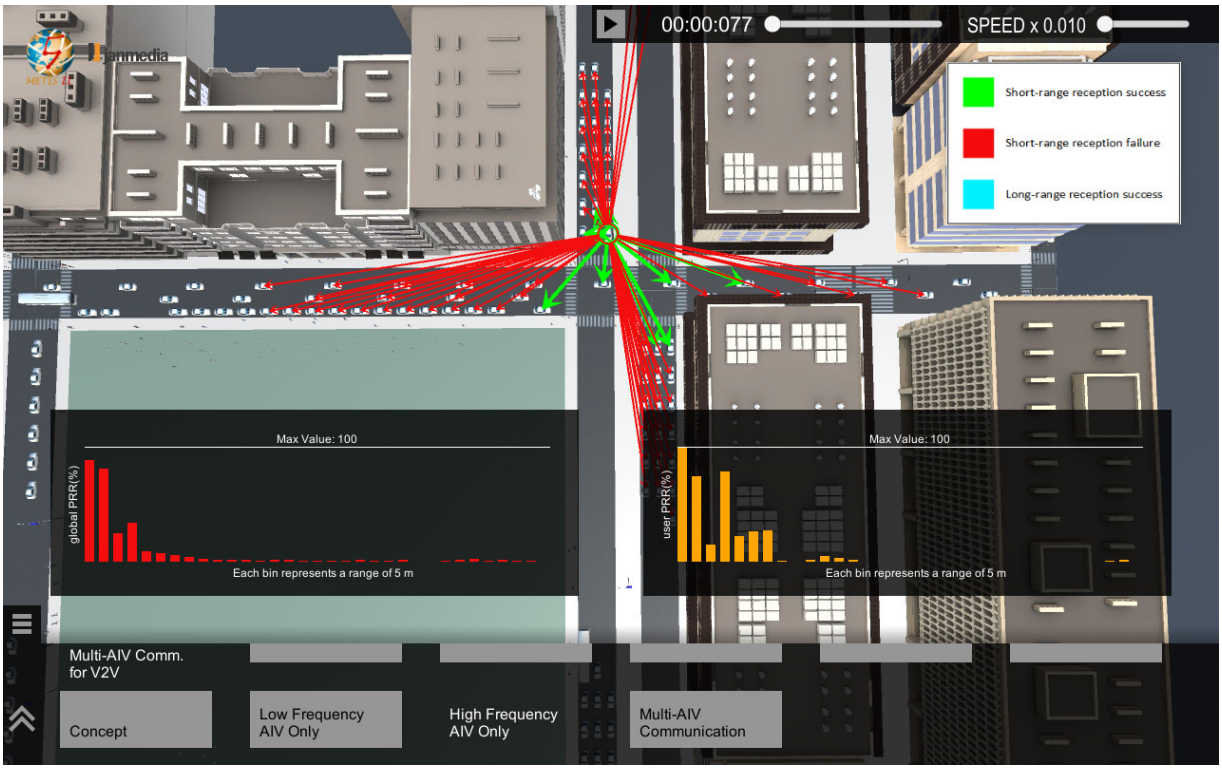


Figure 4-2: Screenshot of MWC 2016 demonstration.

Additional elements were also created, like data operated graphs, contextual illustrations and icons, and the first attempts were made to allow creating visual representations to consecutive project participants.

4.3 Online data based use cases

The final step in the tool development was to link it to external simulators. Being able to connect the Visualization Platform (more or less) directly to simulators would in practice mean the tool is ready to handle all the tasks defined as its proposed features at the beginning of the project. Multiple solutions were taken into account and finally the web socket based approach was chosen as the most efficient. The first interaction items were also implemented, initially allowing the users to start the simulation directly from the Unity application level. After the specific simulator was connected, a more universal solution started being developed, to allow for other simulators and setup options to be considered in the future.

4.3.1 EuCNC June 2017

EuCNC held in Oulu was the last big event in which METIS-II participated as a Project. The exhibition included the last version of the METIS-II Visualization Platform, with the desktop, android and virtual reality versions. The five representatives of METIS-II in the event covered more than fifty visits in the three days that the fair lasted, including several representatives from the European Commission, and a number of actors from verticals, mostly from car industry.

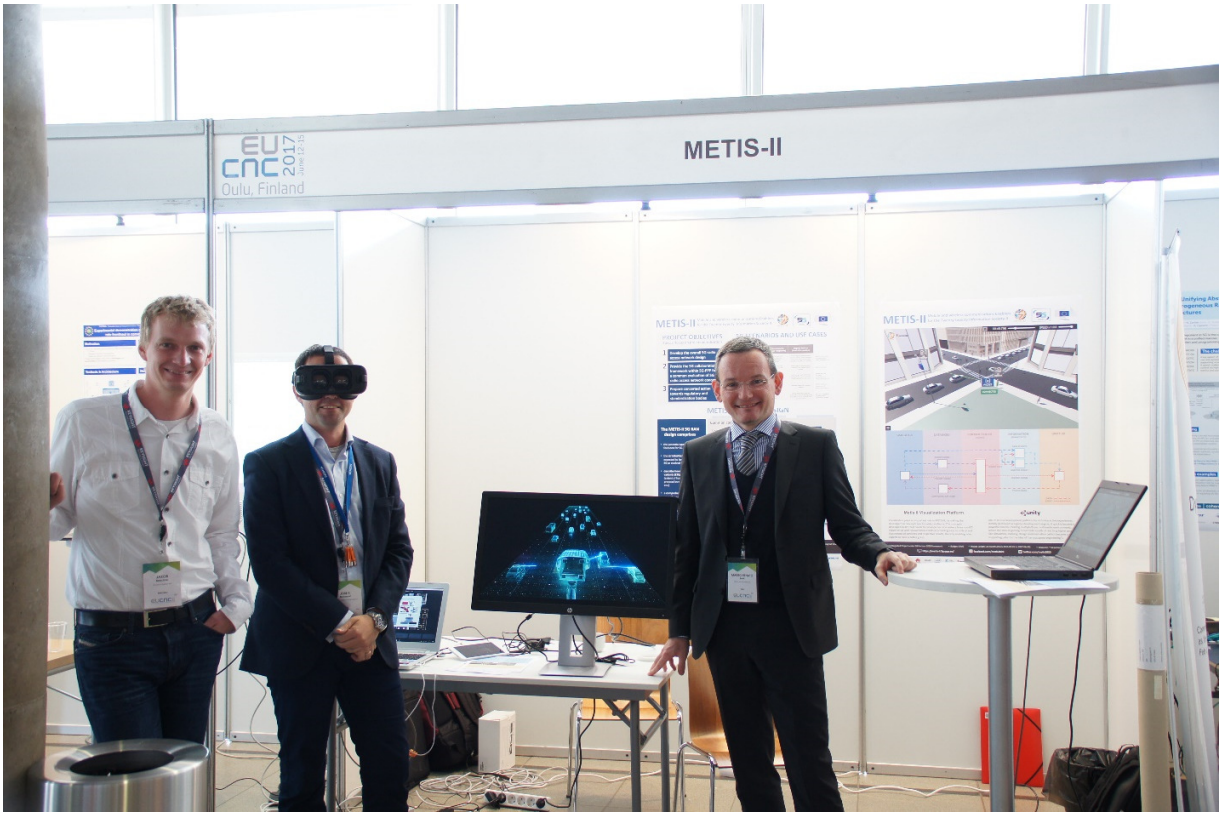


Figure 4-3: METIS-II booth at EuCNC 2017 held in Oulu.

4.4 Android and Windows tablet versions

With all the features mentioned above, it is important to note that the Visualization Platform can be easily compiled into an Android application and also easily installed on Windows tablets. Of course, some online features might be unavailable (especially for the Android version) and there could be some performance fine-tuning tweaks involved, but in general it was already proven during MWC 2016 and EuCNC 2017 events that major features and functions of the tool are not degraded in the process.

4.5 Virtual reality version

A simplified version of the Visualization Platform has also been implemented in virtual reality (VR). Unity has built-in support for the Oculus family of VR devices, notably the Oculus Rift Development Kit 2 (a VR headset with stereoscopic OLED display, 1080×1200 resolution per eye, a 90 Hz refresh rate, and 110° field of view) [OC-RIFT] and the consumer edition of the Gear VR (a mobile headset which requires either a Galaxy S8, S8+, S7, S7 edge, Note5, S6 edge+, S6, or S6 edge) [OC-GEAR]. Using Unity's VR capabilities and Oculus' functionalities, it has been possible not only to transport the Madrid Grid scenario to VR, but also to include the control information and the logical information layer in the scenario. A particular feature of the mobile version is that it relays on the accelerometer, gyroscopes and touchpad integrated in the Gear VR for looking and interacting. Additionally, a spatial version of the user interface (UI) has been designed for the mobile VR version, which positions the UI within the environment itself using Unity's world space canvas render mode. The spatial UI is invoked and dismissed via double tapping on the touchpad; and the UI is attached to the camera, so that as the player moves their head around the UI will stay in a fixed position. Finally, the mobile version displays a reticule that allows the user to select the UI buttons via touchpad taps. As the user experience is completely mobile, a device with VR rendering capability, such as the Samsung Galaxy S7, can be used to operate the demo without the need of any cable.

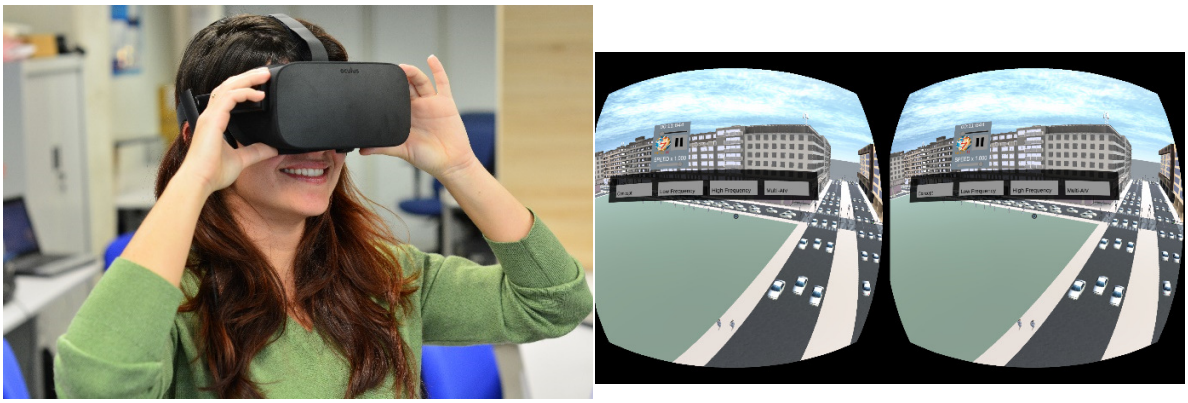


Figure 4-4: Virtual Reality experience with a screenshot of the demonstration.

5 Making the platform universally accessible

Apart from the research-based goals set by METIS-II ambitions, it was important to make the Visualization Platform accessible for virtually anyone. The .xml file responsible for data exchange and parameter control includes comments serving as the basic documentation. It has been evolving into a form of live document that is updated every time a new functionality is brought on-board. The stable Visualization Platform builds are shared on METIS-II website along with the documentation file. The source code is also shared between the project partners on the SVN server. In the final step it should be possible for virtually any party to build their own use cases and to visualize their simulation results within the platform (in both online and offline modes).

Furthermore, the choice of Unity as a basis for this platform facilitates universal accessibility. It is open to both users and developers, responsive to technological changes and strongly present in the mobile domain, compatible with open source programming and modelling software like Blender [BLENDER], among others, and it guarantees that the outcomes and findings of METIS-II will be easily accessible and usable for future purposes (further development, evolutionary and derivative works, etc.), for instance, in further phases of 5G-PPP. It will also help the stiff models understood by industry engineers evolve into meaningful visual experiences accessible to wide audiences.

5.1 Benefits

A visualization platform, made universally accessible presents the following benefits for the industry and academy:

- The platform allows the partners from other 5G-PPP projects and entities outside 5G-PPP to utilize the platform and demonstrate the technologies developed by them, allowing better understanding of concepts.
- The platform can be used as a tool to plug-in newly developed concepts as new features and be shown in scientific fairs and exhibitions for a wider audience.
- The platform can be used for academic purposes by universities (e.g., lectures, practical lab courses etc.,) to teach new students and young professionals, evolving 5G technologies.

5.2 Challenges

In order to make the platform universally accessible by virtually anyone, the following challenges are encountered:

- The platform needs to be maintained on a repository with appropriate version checks.

- Care should be taken that the platform is accessible universally only as a downloadable file, thereby avoiding the corruption of working version by a third party.
- Appropriate documentation and guidelines need to be provided as supplements with the platform, making it easier for new users to use the platform.
- Licensing the platform for third party usage must be properly regulated, to avoid liability in case of any damage caused to third party.
- Support to customers/users in the creation and/or integration of new scenarios.
- Portal to allow any feedback and FAQs.

5.3 Guidelines to access the platform - reference

The final Visualization Platform along with guidelines to use it are hosted in a repository with appropriate version checks. The repository also contains several example scenarios of visualization from various partners, showing evolution of visualization platform and its development methodology. Each example is accompanied with a short text description about the concepts being visualized and relevant references. Anybody can access these folders from browser when they have the public credentials.

The folders containing the aforementioned contents can be accessed from URL:

https://131.246.109.49/repos/METIS-II_Public/

The user credentials required to access the repository are as below:

Username: public

Password: METIS_II4public

6 Conclusion and Visualization Platform's possible future use and development

The Visualization Platform has been a key achievement in METIS-II project and was successfully presented in many international events, with a large appreciation from the audience. Being developed in an open mode, the platform can be further extended and improved in the next months, possibly within the framework of 5G-PPP; the partners that worked on it in the project will still further work on the next developments internally and in collaboration whenever this will be possible

As concepts related to development and evolution of 5G will keep on emerging in the near future, it is important to state that the METIS-II Visualization Platform is ready to face the foreseeable challenges. In the current version, it is possible to imagine any use case being presented and evaluated within the tool by adding new elements to the library of physical objects, like new devices, sensors, vehicles, and environment elements, among others, as well as information layer entities like visualizations of advanced propagation or path loss patterns, qualitative and quantitative representations of values, advanced data visualization elements, etc. As the scene elements are visualized as 3D objects it is also possible to add them more details from both visual and data-related perspective. In the next future, it could be relevant to produce more accurate representations of the real world and to bring the ideas closer to the user perspective.

Finally, as Unity engine can by complemented with Virtual Reality and Extended Reality libraries, the Visualization Platform may become more experience focused, enabling future users to enter the world of 5G even before it actually exists. Further, it is possible to introduce hardware into the loop of visualization and control. Various physical features and performance indicators of the hardware or hardware system can be visualized by the platform. Meanwhile, it can be made available to the user to control the hardware or system functioning from the tool graphical front end.

7 References

[3GPP13-R130092] 3GPP R1-130092, “Discussion on UE-UE Channel Model for D2D Studies”, TSG RAN WG1 Meeting 72, St Julian's, Malta, 28th January - 1st February 2013.

[3GPP16-23401] 3GPP TS 23.401, “General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access (Release 13)”, December 2016.

[3GPP16-38913] 3GPP TR 38.913, “Study on Scenarios and Requirements for Next Generation Access Technologies (Release 14)”, October 2016.

[BLENDER] https://www.blender.org/

[ITUT12-G729] ITU-T Recommendation G.729 “Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)”, June 2012.

[LLS17] Ji Lianghai, M. Liu, H. D. Schotten, “Context-aware Cluster Based Device-to-Device Communication to Serve Machine Type Communications”, in 3rd International Workshop on 5G RAN Design at IEEE International Conference on Communications (ICC), Paris, May, 2017.

[MDB+16] P. Marsch et al., "5G Radio Access Network Architecture: Design Guidelines and Key Considerations," in IEEE Communications Magazine, vol. 54, no. 11, pp. 24-32, November 2016.

[MET13-D11] ICT-317669 METIS, Deliverable 1.1 “Scenarios, requirements and KPIs for 5G mobile and wireless system”, April 2013.

[MET13-D61] ICT-317669 METIS, Deliverable 6.1 “Simulation guidelines, Oct. 2013.

[MET15-D15] ICT-317669 METIS, Deliverable 1.5 “Updated scenarios, requirements and KPIs for 5G mobile and wireless system with recommendations for future investigations”, April 2015.

[MII16-D11] ICT-671680 METIS-II, Deliverable 1.1 “Consolidated scenarios, requirements, use cases and qualitative techno-economic feasibility assessment,” January 2016.

[MII16-D21] ICT-671680 METIS-II, Deliverable 2.1 “Performance evaluation framework”, January 2016.

[MII16-D22] ICT-671680 METIS-II, Deliverable 2.2 “Draft Overall 5G RAN Design,” June 2016.

[MII16-WP] ICT-671680 METIS-II, White Paper on “Preliminary Views and Initial Considerations on 5G RAN Architecture and Functional Design”, March 2016.

[MII17-D32] ICT-671680 METIS-II, Deliverable 3.2 “5G spectrum scenarios, requirements and technical aspects for bands above 6 GHz”, June, 2017.

[MII17-D42] ICT-671680 METIS-II, Deliverable 4.2 “Final air interface harmonization and user plane design”, April, 2017.

[MII17-D52] ICT-671680 METIS-II, Deliverable 5.2 “Final Considerations on Synchronous Control Functions and Agile Resource Management for 5G”, March 2017.

[MII17-D62] ICT-671680 METIS-II, Deliverable 6.2 “Final asynchronous control functions and overall control plane design”, April, 2017.

[MII-WEB] ICT-671680 METIS-II https://metis-ii.5g-ppp.eu

[OC-GEAR] Oculus Gear VR General Information, https://www.oculus.com/gear-vr/

[OC-RIFT] Oculus Rift General Information, https://www.oculus.com/rift/

A First trip through the Visualization Platform

This annex explains how to download and execute the Visualization Platform, its appearance, and the application files structure supporting the whole platform.

A.1 Download and execution

The version of visualization platform described in this document is available at the METIS-II repository as described in Section 5.

After downloading and extracting the zip file, the visualization platform consists of two elements:

- an executable file, main.exe, and
- a data folder, main_Data.

Once the executable file is executed, a window shows the possibility to choose the quality of the representation and the screen resolution to be used, as illustrated in Figure A-1.

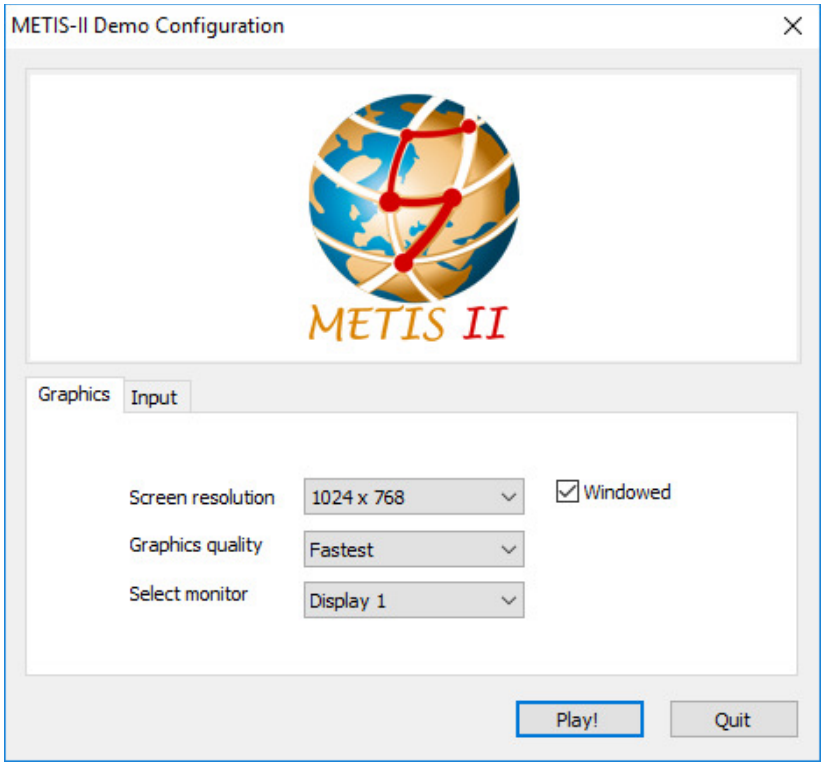


Figure A-1: Video settings window.

A.2 Appearance

Once the “Play!” button is clicked, a main view shows a 3D urban scenario known in METIS-II as the Madrid Grid scenario (Figure A-2). It can be noticed at the first sight that this is a living city with real traffic. There is only one GUI element, which is located at the bottom-left corner: the menu toggle button that shows or hides the menu and the visualization options. Figure A-3 shows the visualization options in the upper part and the menu bar with the menu options in the lower part. The menu bar contains several buttons, and some of them can show more menu options (multi-level).

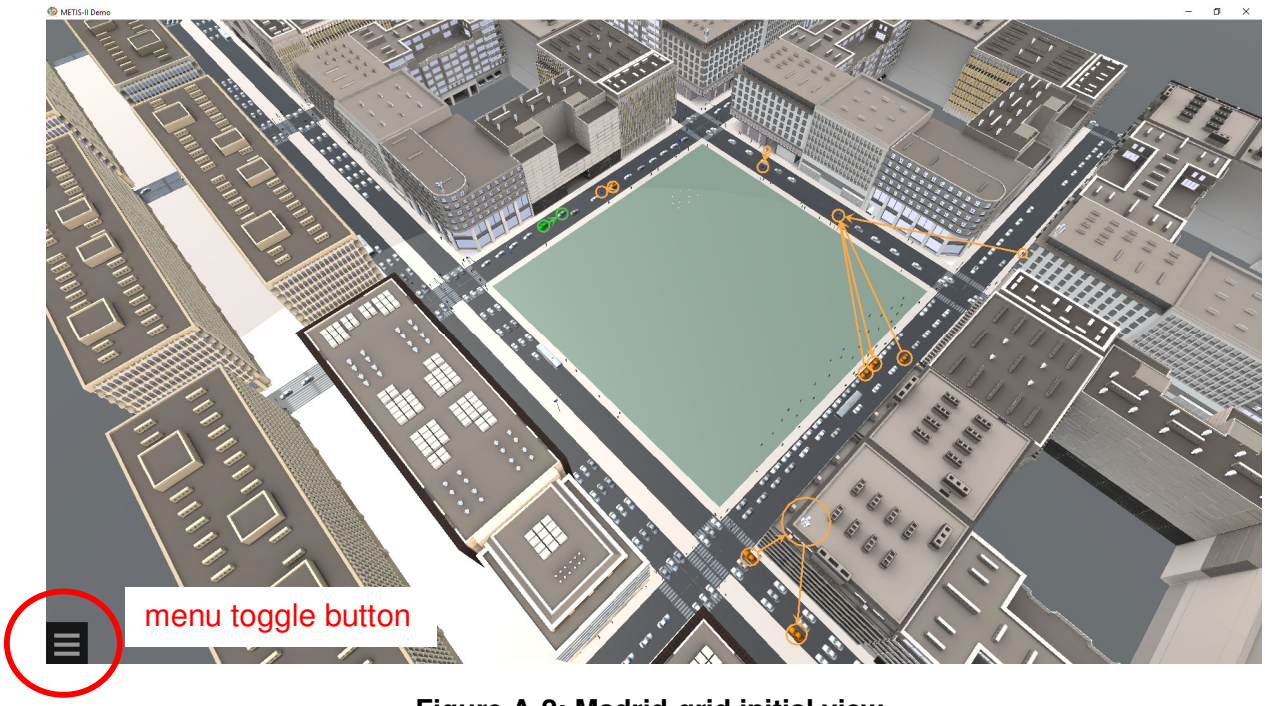


Figure A-2: Madrid-grid initial view.

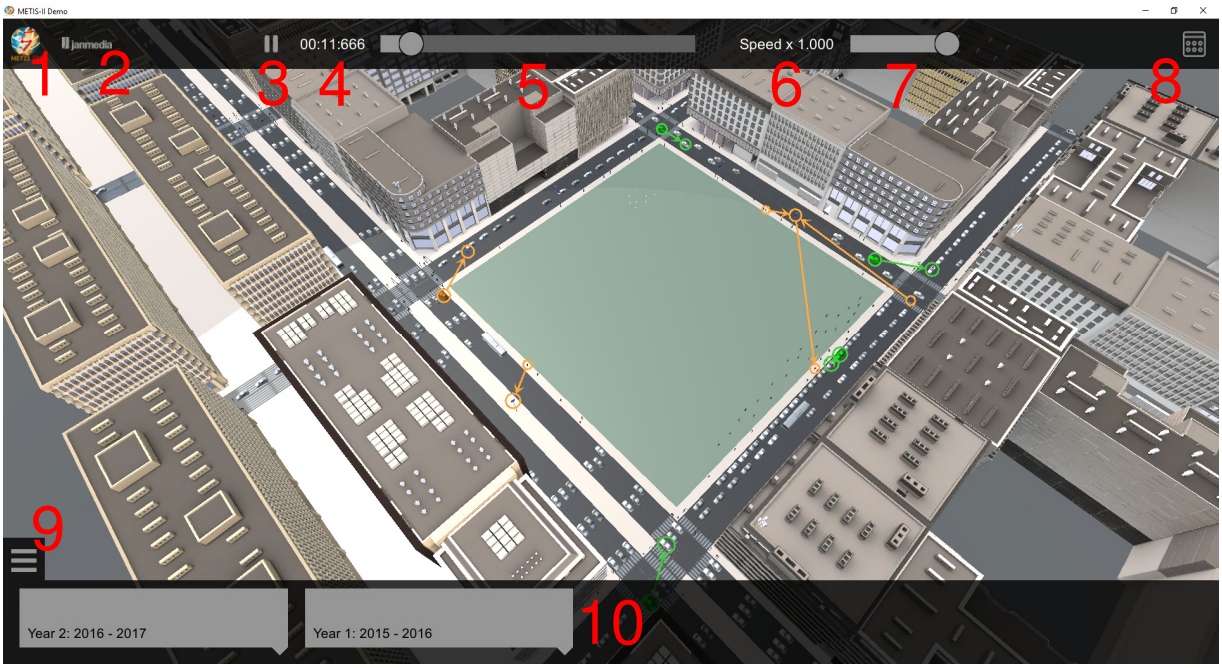


Figure A-3: Madrid-grid initial view and GUI elements identification.

The GUI elements in the initial view are identified from 1 to 10:

1. METIS-II logo. This is a button that closes the VP when clicked.
2. Janmedia logo. Information about METIS-II partner Janmedia, who developed the VP.
3. Pause/resume button. It controls the reproduction of the scene.
4. Current reproduction time.
5. Reproduction time slide bar. By dragging the bar, it is possible to move the scene to the desired reproduction time.
6. Current reproduction speed.
7. Reproduction speed slide bar. By dragging the bar, it is possible speed up or slow down the reproduction.
8. Side menu toggle. Show or hides the side panel with extra data if available.
9. Menu toggle button. Shows or hides the menu.
10. Menu bar that contains the menu options.

It is possible to click on a vehicle. After doing that, the camera starts following the selected vehicle as it moves around the scenario (Figure A-4).

Figure A-4: Follow view mode.

Figure A-5 and Figure A-6 are some use case view examples in the VP showing the graphical capabilities of the tool. In summary, the number of objects that can be used are:

- Mobile objects (pedestrians, cars or buses) that move according to a mobility trace
- Traffic lights
- Base stations (macro cell or small cell)
- Lines or arrows
- Circles
- Icons
- Dynamic (moving) icons
- Spheres
- Bars
- Charts
- Static overlay material

One of the capabilities of the VP is to change dynamically the color of some objects, as illustrated in Figure A-6.

Figure A-5: Use case view with icons, lines and coloring object.

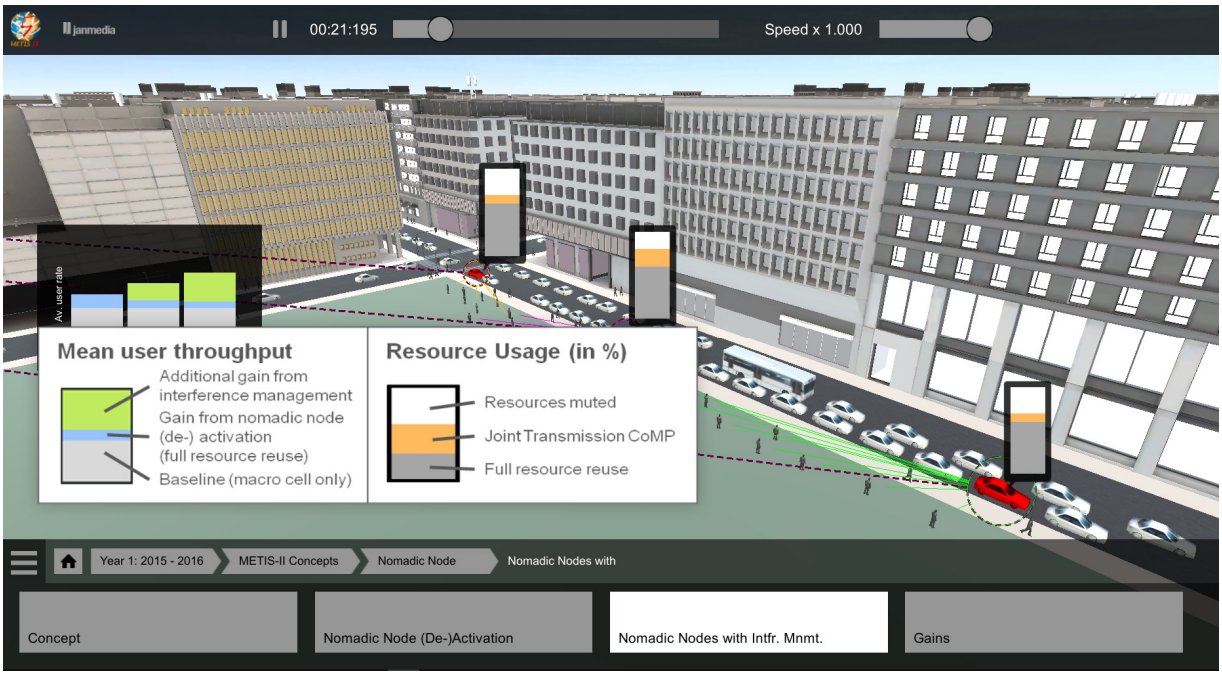


Figure A-6: Use case view with static overlay material, bars, charts, lines, circles and coloring.

A.3 Application files structure

As mentioned before, the visualization platform consist of an executable file and a data folder. The main content of the data folder is the StreamingAssets folder. This folder contains the icons root folder (Icons), the images root folder (Pics), and the data root folder (Data). The icons and images root folders contain, respectively, icons and static contents shown in the visualization platform. The data root folder contains the main.xml file and the graphical representation binary files. The main.xml file defines the VP appearance by deciding the configurations of the views to be presented to the user. This configuration includes the list of graphical representation files that should be read for each view.

In summary the structure of the data root folder is:

- StreamingAssets - assets root folder
- Icons - icons root folder
- Pics - images root folder
- Data - data root folder
- main.xml
- graphical representation binary files

There are multiple types of graphical representation binary files:

- LineTraceNode is used to draw arrows between objects represented in the VP.
- IconTraceNode is used to draw icons close to objects represented in the VP.
- TrafficNode is used to indicate the position of a set of objects in periodic samples of time, i.e., contains mobility traces.

In order to customize the VP to show a specific use case, both the main.xml and the graphical representation files should be modified.

B Configuration of.xml file

This annex explains the structure and contents of the main.xml file.

The following code contains an excerpt of the main.xml file in the last visualization platform release from METIS-II.

```
<?xml version="1.0" encoding="utf-8"?>

<Main xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" selectedChildIds="all">

   <MainNode xsi:type="MainNode" id="main" duration="180" selectedChildIds="all">

      <lineParams>

         <LineParams id="1" hexColor="#888888" thickness="2" fromCircle="false"
toCircle="false" toArrow="0" />

         <LineParams id="2" hexColor="#888888" thickness="2" fromCircle="false"
gap="5" dash="10" toCircle="false" toArrow="0" />

         <LineParams id="3" hexColor="#44cc44" thickness="3" fromCircle="false"
toCircle="false" toArrow="0" />

      </lineParams>

      <fontStyles>

         <FontStyle id="font-style-0" fontSize="30" fontStyle="2"
hexColor="#443300"/>

      </fontStyles>

      <exclusiveChilds>

         <Node label="Year 2: 2016 - 2017">

         (…)

         </Node>

         <Node label="Year 1: 2015 - 2016">

         (…)

         </Node>

      </exclusiveChilds>

      <childs>

         <Node xsi:type="GridContainerNode" id="grid-container"
viewType="GridContainerView" selectedChildIds="grid-0">

            <exclusiveChilds>

               <Node xsi:type="GridNode" id="grid-0" viewType="MadridGrid"
replaceBuildings="true"/>

               <Node xsi:type="GridNode" id="grid-1" viewType="MadridGrid"
replaceBuildings="true" dontReplaceIds="Building-rt_09-12_(2)"
selectedChildIds="all"/>

               <Node xsi:type="GridNode" id="grid-2" viewType="MadridGrid"
```

```
removeBuildings="true" />

            </exclusiveChilds>

         </Node>

         <Node xsi:type="TrafficNode" id="car-traces"
src="simplifiedTC2_Car_sim180s_ step0.1s_withAngles.bin" interval="0.1" idPrefix="c"
mobileType="CarSedan01" />

         <Node xsi:type="TrafficNode" id="ped-traces"
src="simplifiedTC2_Ped_sim180s_ step0.1s_90peds_withAngles.bin" interval="0.1"
idPrefix="p" mobileType="Person" />

         <Node xsi:type="TrafficNode" id="bus-traces"
src="simplifiedTC2_Bus_sim180s_ step0.1s_withAngles.bin" interval="0.1" idPrefix="b"
mobileType="Bus" />

         <Node id="micro-cells" idPrefix="l" childViewType="lamp-single-ap-002"
selectedChildIds="all">

            <childs>

               <Node id="0" x="160" z="282.5" rotationY="90" />

               <Node id="1" x="207" z="282.5" rotationY="90" />

               <Node id="2" x="254" z="282.5" rotationY="90" />

            </childs>

         </Node>

         <Node id="ran" idPrefix="r" selectedChildIds="all">

            <childs>

               <Node id="0" x="0" y="65" z="350" viewType="RAN" />

               <Node xsi:type="IconsTraceNode" hOffset="-71" vOffset="-55"
src="core-net-icon-trace.xml" />

            </childs>

            <icons>

               <Icon id="1" src="01_core_network.png" />

            </icons>

         </Node>

         <Node xsi:type="LightsNode" id="car-lights-horizontal" duration="90"
childViewType="TrafficLightSingle" selectedChildIds="all">

            <periods>

               <Period endTime="2" phase="yellow" />

               <Period endTime="50" phase="red" />

               <Period endTime="87" phase="green" />

            </periods>

            <childs>

               <Node id="ch-h1-v2" x="128.5" y="0.2" z="545.5" rotationY="270"
scaleY="1.5" />

               <Node id="ch-h1-v3" x="266.5" y="0.2" z="545.5" rotationY="270"
scaleY="1.5" />
```

```
            </childs>
        </Node>
        <Node    xsi:type="CamTargetNode"    x="314.43"    y="174.92"    z="424.64"
rotationX="58.51" rotationY="224.17" rotationZ="0.00" />
        <Node     xsi:type="LineTraceNode"     id="temporary-lines"     src="line-
trace_view0.bin" isDefault="true" />
        <Node    xsi:type="ColorTraceNode"    id="temporary-colors"    src="color-
trace_view0.bin" isDefault="true" />
      </childs>
   </MainNode>
</Main>
```

As it can be observed in the code, the main.xml contents are:

- A couple of lines about xml encoding
- A MainNode that contains:
    - Definition of some global parameters (lineParams, fontStyles)
    - Definition of child Nodes
        - with xsi:type equal to: GridContainerNode, TrafficNode, LightsNode, CamTargetNode, LineTraceNode, ColorTraceNode, or
        - without xsi:type but with childViewType.
    - Definition of exclusiveChild Nodes

Each one of the `exclusiveChilds` of the MainNode represents one of the buttons of the menu bar in the lower part of the initial view. By pressing those buttons, the initial view may change to illustrate a different concept.

The `childs` of the MainNode are nodes that contain information used by the initial view or information that can be reused by all the views (all the `exclusiveChilds`).

The `childs` are nodes without selection restrictions in the VP, but only one of the `exclusiveChilds` can be selected at one time. The nodes among the `childs` or `exclusiveChilds` may contain also other `childs` or `exclusiveChilds`, and this way the structure can be nested.

The following sections explain in detail how to configure the main.xml file to achieve different graphical representations. First, some parameters found in many nodes are described. After that, each graphical representation capability is presented and the nodes used to use this capability are explained.

B.1 Global node attributes

The set of global attributes, which are also found in multiple xml elements, are found in Table B-1.

Table B-1: Common parameters.

Parameters	Description
`@x, @y, @z`	single values, default: 0, position of the visual representation of the node in 3D space relative to the parent
`@rotationX, @rotationY, @rotationZ`	single values (degrees), default: 0, rotation of the visual representation of the node relative to the parent
`@scaleX, @scaleY, @scaleZ`	single positive values, default 1, local scale of the visual representation of the node
`@viewType`	string, default: “NodeView”, name of the visual representation of the node
`@childViewType`	string, name of the visual representation of the child nodes
`@label`	string, default: null, used in navigation buttons etc.
`@exclusiveChilds`	child nodes, only one can be selected
`@childs`	child nodes, no selection restrictions
`@selectedChildIds`	list of `@childs id` separated by commas, or all - to select all
`@setStartTime`	single value, default: -1, sets presentation time when node is selected
`@speed`	greater than 0, 1 is normal speed - sets playback speed when node is selected
`@disableIds`	list of childs to temporary deselect when node is selected (string separated by commas). State of selection of this child is restored, when node is deselected Node is looking among their children first, then in the children of parent, etc. It allows to temporarily disable or replace presentation nodes, for example replacement of car traffic with other bin file.
`@repeatAfterBroken`	boolean, try to reload file if not found

B.2 Line styles definitions

To represent lines in the VP, the line styles have to be previously defined in an xml element called lineParams. This element contains a list of line styles, each of them with the parameters shown in Table B-2.

Table B-2: Parameters for definition of icons.

Parameters	Description
`@id`	used to instantiate the line style in LineTraceNode binary files
`@hexColor`	color of line (#RRGGBB format)
`@thickness`	line thickness in pixels
`@fromCircle`, `@toCircle`	true/false - decide whether to draw a circle around the origin (`@fromCircle`) or the destination (`@toCircle`)
`@fromArrow`, `@toArrow`	value in {0,1, -1 (default)} Arrow style: 0 means filled arrow, 1 means non-filled arrow, -1 means no arrow
`@fromArrowDistance`, `@toArrowDistance`	override default distance between center of object and pointing arrow (value in meters)
`@gap`, `@dash`	single values in pixels - allows to draw dashed lines - if both are greater than zero

Line styles definitions are cascading. It means that the line style definition valid for certain id in a specific node is the definition found within this node for that id, if there is a definition for that id in this node, or the definition found in the nearest ancestor node which provides a definition for that id, in other case.

Example:

```
<lineParams>
	<LineParams id="3" hexColor="#dd2255" thickness="2"/>
	<LineParams id="2" hexColor="#882200" thickness="3"/>
	<LineParams id="1" hexColor="#ff00ff" thickness="3" toCircle="false" toArrow="1"/>
	<LineParams id="10" hexColor="#660044" thickness="3" gap="5" dash="10" toCircle="false" toArrow="0" toArrowDistance="0.25"/>
	<LineParams id="11" hexColor="#ff44aa" thickness="10" gap="2" dash="10"/>
</lineParams>
```

B.3 Icons definition

To represent icons in the VP, the icons have to be previously defined in an xml element called icons. This element contains a list of icons, each of them with the parameters shown in Table B-3

Table B-3: Parameters for definition of icons.

Parameters	Description
`@id`	used to instantiate the icon in the binary file IconTraceNode
`@src`	file path, home dir for relative path is icons root folder
`@scaleX,@scaleY`	single positive value, default: 1. Defines the size of icon. Note that all icons are scaled with the interface considering that the reference interface resolution is 1280 x 768. Interface is expanded (horizontally or vertically) if screen aspect ratio does not fit reference resolution.

Icons definitions are cascading. It means that the icon definition valid for certain id in a specific node is the definition found within this node for that id, if there is a definition for that id in this node, or the definition found in the nearest ancestor node which provides a definition for that id, in other case.

Example:

```
<icons>
   <Icon id="0" src="icon-0.png" scaleX="1" scaleY="1"/>
   <Icon id="1" src="icon-1.png"/>
   <Icon id="2" src="icon-2.png"/>
</icons>
```

B.4 Definition of traffic objects

To represent pedestrians, cars and buses, use a TrafficNode with the parameters shown in Table B-4.

Table B-4: Parameters for definition of traffic objects.

Parameters	Description
`@src`	file path to binary file. Paths are absolute, start with "file://" or "http://", or relative to data root folder
`@idPrefix`	`idPrefix` of generated objects, single char, determines type of traffic objects, values: “c” - car, “p” - pedestrian, “b” - bus
`@interval`	time interval in seconds between consecutive samples
`@mobileType`	allows to select 3D model of traffic objects, values: “CarSedan01”, “Person”, “Bus”

Examples:

```
<Node xsi:type="TrafficNode" id="car-traces" src="simplifiedTC2_Car_sim180s_step0.1s_withAngles.bin" interval="0.1" idPrefix="c" mobileType="CarSedan01" />

<Node xsi:type="TrafficNode" id="ped-traces" src="simplifiedTC2_Ped_sim180s_step0.1s_90peds_withAngles.bin" interval="0.1" idPrefix="p" mobileType="Person" />

<Node xsi:type="TrafficNode" id="bus-traces" src="simplifiedTC2_Bus_sim180s_step0.1s_withAngles.bin" interval="0.1" idPrefix="b" mobileType="TempBus" />
```

B.5 Camera settings node

Use a CamTargetNode indicating the position and orientation of the camera with the parameters: `@x`, `@y`, `@z`, `@rotationX`, `@rotationY`, and `@rotationZ`.

Example:

```
<Node xsi:type="CamTargetNode" x="254.63" y="91.40" z="339.69" rotationX="63.17" rotationY="156.38" rotationZ="0.00"/>
```

B.6 Definition of pop-up image

Use a StaticContentNode with the parameters shown in Table B-5.

Table B-5: Parameters for definition of camera objects.

Parameters	Description
`@src`	file path - absolute - starts with "file://" or "http://" or relative to home data dir images root folder
`@align`	string of 2 chars determines position of origin point and pivot point, first letter for horizontal align: "l" - left, "r" - right, "c" - center, second for vertical align: "t" - top, "b" - bottom, "m" - middle

Example:

```
<Node xsi:type="StaticContentNode" label="Static Content 1" src="small.png" align="br"/>
```

B.7 Definitions of static objects: macro cells, small cells, static pedestrians, radio access network

Use a Node without type but with parameters @idPrefix and @childViewType, with as many childs as objects of the same type have to be represented. The parameters for the parent node and childs are indicated in Table B-6.

Table B-6: Parameters for definition of static objects.

Parameters	Description
`@idPrefix`	(1 char) all objects must be wrapped by the parent with that id
`@childViewType`	Object to visualize, some examples: • `lamp-single-ap-001`, for lamppost • `Antenna5G_01`, for an antenna or base station • `Person`, for a pedestrian • `Sensor`, for a sensor • `RAN`, for a Core Network object
`@selectedChildIds`	`"all"` to visualize all child objects
`@id`	Id of objects, must be a number
`@x, @y, @z, @rotationX, @rotationY, @rotationZ, @scaleX, @scaleY, @scaleZ`	See Table B-1

Example adding a lamp post:

```
<Node idPrefix="l" childViewType="lamp-single-ap-001" y="0.2"
selectedChildIds="all">
      <childs>
            <Node id="0" x="160" z="282.5" rotationY="90"/>
            (...)
      </childs>
</Node>
```

Example adding an antenna (or base station):

```
<Node idPrefix="a" childViewType="Antenna5G_01" selectedChildIds="all">
      <childs>
            <Node id="0" x="264" y="32.5" z="426"/>
            (...)
      </childs>
</Node>
```

Example adding a pedestrian:

```
<Node id="static-pedestrians" childViewType="Person" selectedChildIds="all"
y="0.35" idPrefix="t">
      <childs>
            <Node id="0" x="152.87" z="400.61" rotationX="0"
rotationY="39.58337" rotationZ="0" scaleX="1" scaleY="1" scaleZ="1"/>
            (...)
      </childs>
</Node>
```

Example adding a sensor:

```
<Node id="sensors" idPrefix="s" childViewType="Sensor"
selectedChildIds="all" y="0.25">
      <childs>
            <Node id="0" x="177.00" z="300.00"/>
            (...)
      </childs>
</Node>
```

Example adding a radio access network representation object (note that an icons section is added to the xml code compared to the previous examples because the childViewType RAN has no visible representation associated):

```
<Node id="ran" idPrefix="r" childViewType="RAN" selectedChildIds="all">
      <childs>
            <Node id="0" x="210" y="80" z="335"/>
            <Node xsi:type="IconsTraceNode" hOffset="-71" vOffset="-55"
src="core-net-icon-trace.xml" />
      </childs>
      <icons>
            <Icon id="1" src="01_core_network.png" />
      </icons>
</Node>
```

B.8 Object color change

Use a ColorTraceNode indicating with an `@src` parameter the binary file with information to change the colors. To build the binary file see Section C.1.

Example:

```
<Node xsi:type="ColorTraceNode" src="color-trace.bin"/>
```

B.9 Lines between objects

Use a LineTraceNode indicating with an `@src` parameter the binary file with information to represent the lines. To build the binary file see Section C.2.

Example:

```
<Node xsi:type="LineTraceNode" src="line-trace.bin"/>
```

B.10 Lines between objects visible all the time

Use a LineLiveNode indicating with an `@src` parameter the binary file with information to represent the lines. To build the binary file see Section C.3.

Example:

```
<Node xsi:type="LineLiveNode" id="LINESDEMO" src="http://metisbin.test.janmedia.pl/
download.php?caseid=LINESDEMO"/>
```

B.11 KPI bars next to objects

Use a node of type BarsTraceNode with the parameters in Table B-**7**

Table B-7: Parameters for BarsTraceNode.

Parameters	Description
`@src`	file path
`@hexColors`	colors, determines number of values in bar
`@hOffset, @vOffset`	in pixels, the position of the left bottom corner to the center of the object
`@width`	in pixels - bar width
`@height`	in pixels - bar height
`@maxValue`	maxValue - height of bars determined by the proportion value/maxValue

See Section C.4 for instructions to build the binary file.

Example:

```
<Node xsi:type="BarsTraceNode" hOffset="10" vOffset="10" width="40" height="100"
maxValue="100" hexColors="#ff0000,#ffff00,#008800" src="bars-trace.bin"/>
```

B.12 Display of chart plot in one corner of the screen

Use a node of type ChartTraceNode with the parameters in Table B-8

Table B-8: Parameters for ChartTraceNode.

Parameters	Description
`@src`	file path
`@hexColors`	colors, determines number of values in bar
`@align`	string of 2 chars determines position of origin point and pivot point, first letter for horizontal align: "l" - left, "r" - right, "c" - center, second for vertical align: "t" - top, "b" - bottom, "m" - middle
`@hOffset, @vOffset`	in pixels, offset between origin point and pivot point
`@barWidth`	width of single bar (default: 40)
`@gap`	defines gaps between bars (default: 10)
`@padding`	padding of container (default: 30)
`@numBars`	defines number of bars
`@maxValue`	maxValue - height of bars determined by the proportion value/maxValue
`@xLabel, @yLabel`	axis labels

See Section C.5 for instructions to build the binary file.

Example:

```
<Node xsi:type="ChartTraceNode" id="chart-trace-0" align="rb" hOffset="30" vOffset="30" barWidth="40" barHeight="100" barGap="10" numBars="3" maxValue="100" hexColors="#ff0000,#ffff00,#008800" yLabelDistance="8" xLabelDistance="8" xLabels="20 m,40 m,60 m" yLabel="Some values" isMaxValueVisible="true" src="chart-trace.bin"/>
```

B.13 Illustration of icons next to infrastructure elements

To visualize icons next to other visualized objects, first the icons have to be defined as explained before. Then, use a node of type IconsTraceNode with the parameters in Table B-9.

Table B-9: Parameters for IconsTraceNode.

Parameters	Description
`@src`	file path to the IconsTraceNode binary file containing the information to visualize
`@hOffset, @vOffset`	in pixels, the position of the left bottom corner of first icon to the center of the object
`@gap`	defines gaps between icons

See Annex C.6 for instructions to build the binary file. The binary file includes for each icon an id corresponding to the specific field

Example:

```
<Node xsi:type="IconsTraceNode" hOffset="30" vOffset="30" gap="10" src="icons-trace.bin"/>
```

B.14 Illustration of animated icons next to infrastructure elements

Use an AnimIconsTraceNode indicating with an `@src` parameter the binary file with information to represent the icons. To build the binary file see Annex C.7.

Example:

```
<Node xsi:type="AnimIconsTraceNode" src="anim-icons-trace.bin"/>
```

B.15 Display spheres containing elements

Use a SpheresLiveNode indicating with an `@src` parameter the binary file with information to represent the spheres. To build the binary file see Annex C.8.

Example:

```
<Node xsi:type="SpheresLiveNode" id="SPHEREDEMO" src="http://metisbin.test.janmedia.pl/download.php?caseid=SPHEREDEMO"/>
```

B.16 Status data update

Data structure: `id1,timestamp;id2,timestamp;id3,timestamp`...

Tries to find nodes by id and checks whether the loaded data is up to date. If not, reloads nodes. It allows you to update the data remotely.

`@interval` - in seconds

```
<Node xsi:type="PingNode" url="http://metisbin.test.janmedia.pl/check.php" interval=".25"/>
```

C Graphical representation binary files

This annex describes the structure of the graphical representation binary files.

C.1 ColorTraceNode

4 bytes (int32) - number of changes

Repeat as many times as number of changes:

- 4 bytes (single) - time in seconds
- 1 byte (char) - prefix of object `id`
  - 'c' - car, 'b' - bus, 'p' - pedestrian, 'a' - macrocell, 'l' - smallcell,
  - 's' - sensors, 't' - static pedestrians,
  - 'C' - for all cars, 'B' for all buses etc.
- 4 bytes (int32) - object `id`
- 4 bytes (int32) - color value

C.2 LineTraceNode

4 bytes (int32) - number of changes

Repeat as many times as number of changes:

- 4 bytes (single) - time in seconds
- 1 byte (char) - prefix (defined in xml) of object id1
- 4 bytes (int32) - object id1
- 1 byte (char) - prefix (defined in xml) of object id2
- 4 bytes (int32) - object id2
- 4 bytes (int32) - id of line style, if 0 line disappear

C.3 LineLiveNode

4 bytes (int32) - number of changes

Repeat as many times as number of changes:

- 1 byte (char) - prefix (defined in xml) of object id1

- 4 bytes (int32) - object id1
- 1 byte (char) - prefix (defined in xml) of object id2
- 4 bytes (int32) - object id2
- 4 bytes (int32) - id of line style

C.4 BarsTraceNode

4 bytes (int32) - number of changes

Repeat as many times as number of changes:

- 4 bytes (single) - time in seconds
- 1 byte (char) - prefix (defined in xml) of object id
- 4 bytes (int32) - object id
- 4 bytes (single) - value1
- 4 bytes (single) - value2
- 4 bytes (single) - value3

C.5 ChartTraceNode

4 bytes (int32) - number of changes

Repeat as many times as number of changes:

- 4 bytes (single) - time in seconds
- 1 byte (byte) - bar index
- 1 byte (byte) - value index
- 4 bytes (single) – value

C.6 IconsTraceNode

4 bytes (int32) - number of changes

Repeat as many times as number of changes:

- 4 bytes (single) - time in seconds
- 1 byte (char) - prefix (defined in xml) of object id
- 4 bytes (int32) - object id
- 1 byte (char) - operation ('+' - add icon to object, '-' - remove, 'x' - clear icons)
- 4 bytes (int32) - icon id (for add or remove)

C.7 AnimsIconsTraceNode

4 bytes (int32) - number of changes

Repeat as many times as number of changes:

- 4 bytes (single) - time in seconds
- 4 bytes (single) - duration of anim in seconds
- 1 byte (char) - prefix (defined in xml) of object start id
- 4 bytes (int32) - object start id
- 1 byte (char) - prefix (defined in xml) of object end id
- 4 bytes (int32) - object end id
- 4 bytes (int32) - icon id

C.8 SpheresLiveNode

4 bytes (int32) - number of changes

Repeat as many times as number of changes:

- 1 byte (char) - prefix of object id
- 4 bytes (int32) - object id
- 4 bytes (int32) - sphere id

D Real time interaction with simulators

This annex describes a special use case of the Visualization Platform in which part of the information visualized comes from a simulator running in parallel. The annex is focused on a socket-based interaction.

D.1 General description of the solution

This section describes the general description of the solution implemented to integrate the METIS-II visualization platform with external simulators. This integration enables a real time interaction in local mode, with both entities running is the same machine, and in remote mode, with each entity in a different machine.

D.1.1 The control widget

The process is controlled by the user running the visualization platform. Once the user starts the visualization platform and selects a section with real time interaction, a control widget showing multiple parameters and buttons appears at the right part of the screen. The appearance of the control widget is shown in Figure D-1.

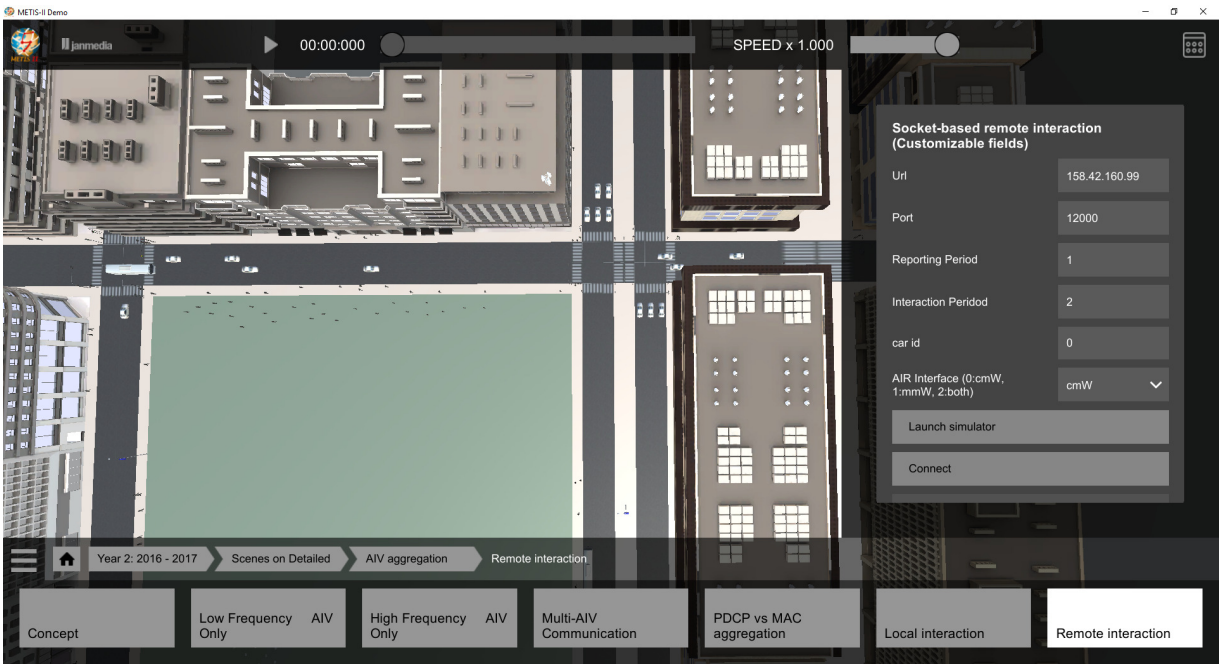


Figure D-1: Appearance of the control widget.

This control widget has a couple of text lines in the upper part that describe its purpose. Below those lines, there are multiple elements that allow the user to configure some parameters related to the simulation and the interaction between the visualization platform and the simulator.

In the example shown in Figure D-1, there are six elements. These elements are five text boxes and one selection list. Specifically, the parameters that can be changed in this example are:

- url of the simulation server (text box),
- port of the remote simulation server to get connected (text box),
- reporting period (text box),
- interaction period (text box),
- car id: ID of the car focus of the visualization (text box),
- AIR interface: centimeter waves, millimeter waves or combination of both (selection list).

The first four parameters are related to the interaction itself, while the last two are related to the simulation.

In addition to the configuration elements, the control widget presents several buttons to launch the simulator in the server (“Launch simulator” button), establish a socket connection (“Connect” button), start, pause, and resume the simulation (“start”, “pause” and “resume”, respectively).

D.1.2 Message passing with sockets

In order to communicate the visualization tool and the simulation tool binary messages are exchanged between the visualization platform and the simulator using sockets.

The visualization tool writes the simulation parameters to a message “params” that is read periodically by the simulation tool to check if any parameter has been changed (the period being “interaction period” times the “reporting period” seconds). The message “params” contents are configurable.

The params message payload is structured according to the following:

- Object id: int32
- Air interface option: byte
- Reporting period: byte
- Interaction period: byte
- Timestamp: single
- Stop flag: byte

The simulation tool, writes the simulation statistics to binary messages with certain periodicity (“reporting period” seconds). The visualization tool checks if the binary messages with results are received in order to load them and represent the information that they contain. These results messages can be of different types.

Socket messages are divided into two fields: header and payload. The header is a five-byte field in which the first byte indicates the type of message and the following four bytes (integer) is the message payload size (it does not include the header size). The payload is a length-variable field which contains the message. The payload information makes use of the legacy binary file structure. The header field values are shown in Table D-1.

Table D-1: Socket message header field value according to the file conveyed in the payload.

Type of message	Header value: Hex (ASCII)
params	0x30 (0)
Colors	0x40 (@)
Lines/arrows	0x41 (A)
Bars	0x42 (B)
Charts	0x43 (C)
Icons	0x44 (D)
Animated icons	0x45 (E)
Graphs	0x47 (G)
Scene objects	0x4F (O)

An example of the socket based interaction is described in Figure D-2. Communication relies on a bidirectional socket, and both the VP and the simulator take the role of (implement) the socket server and client at the same time. The VP sends messages from port 12000 and listens to port 12001 and the simulator listens to port 12000 and sends messages to the VP on port 12001. The example depicted in Figure 9 illustrates a VP and UPV simulator running in different (remote) computers. UPV simulator has a public IP (158.42.160.99) and the VP can be host on any IP address. However, this port setup allows having both the simulator and the VP running on the same machine (localhost configuration) and ensures there is not any kind of NAT filtering by any router/firewall in the communication path when using two remote computers.

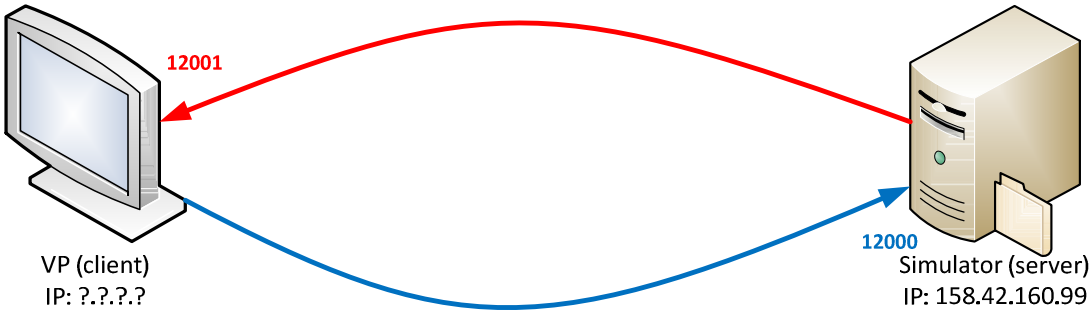


Figure D-2: Socket message exchange architecture using remote computers.

D.1.3 Interaction procedure

The interaction procedure comprises the following points:

- After pressing “Launch simulator”, the visualization platform makes a system call to run the simulator executable file, in local mode, or a call that launches the simulator in a remote server (a php call for example), in remote mode. As a results, a black window may appear in local mode denoting that the simulator is started.
- After pressing “Connect”, a sockets connection is established between the visualization platform and the simulator. The visualization platform writes the parameters set in the control widget to the binary message “params” and sends it to the simulator. In addition to the parameters, also a time stamp (set to 0 initially, and a stopping flag (set to 0 too) are written to the parameters file. If the connection is successful, “Start” button is enabled.
- After pressing “Start”, the simulator reads the parameters message file received and starts the simulation.
- With a periodicity set by the parameter “reporting period”, the simulator writes some statistics of the last “reporting period” seconds into binary messages that are sent to the visualization platform.
- The simulator continues the simulation for as many reporting periods as indicated by the parameter “interaction period” and stops.
- The simulator checks the “params” message. If at that time, neither the time stamp nor the stopping flag have changed “params” message will be checked again. If the stopping flag is set, simulation ends. If the stopping flag is not set, but the time stamp is changed and some parameter is changed, the simulation is accordingly configured. Otherwise, simulation continues without reconfiguration.
- Whenever the visualization platform detects that binary messages are received from the simulator, it reads them and represents the graphical information that they contain when

the time of visualization reaches the start of the period to which the binary message corresponds to.

- During the normal operation, the visualization platform user is allowed to change some parameters (such as the Car id parameter and AIR interface in the example). These changes are reflected in “params” message.
- Also during the normal operation, the visualization platform changes the time stamp in the “params” mesage periodically. Specifically, the first change is after loading the binary messages corresponding to the last reporting period of an interaction period.

Finally, let us clarify the role of “reporting period” and “interaction period”. In unity, value entered as "reporting period" is passed to PartNodes linked by @partsNodeIds as partDefinition.duration. Value entered as "interaction period" decides how often parameters file for the simulator is saved. For example: if reporting period is set to 5 seconds and interaction period is set to 4 it looks like:

- first saved in 0 second,
- next in 15 second - (interaction period - 1) * reporting period - indicates when the last part starts,
- next in 35 second - after 20 seconds (interaction period * reporting period) - indicates when the last part of next interaction period starts,
- next in 55 second - after 20 seconds etc.

D.1.4 UPV example

At the time of writing this documentation, the UPV hosts a simulator on IP: 158.42.160.99. To run the simulator in the remote machine, write the following address in your navigator of choice’s address bar:

>> http://158.42.160.99/simulator/index.php?interaction=3

The server automatically detects the client making the web request. It is important that the simulator is called in the very first place. Multiple requests will end with a single instance of the simulator, since each time the simulator is called the server kills any running instance of the simulator in the server.

D.1.5 Customizable socket based interaction

The content the VP (i.e. message payload) sends to the simulator in “params” message is fully customizable and can be adapted to the target simulator, but the header and payload message structure is maintained. The customizable payload accepts the types of data collected in Table D-2:

Table D-2: Customizable socket based payload supported type of data.

Supported type of data	Number of bytes
char	1
short	2
int	4
float	4
bool	1
string	1 x number of characters in the string

D.2 Socket based interaction xml configuration

This section details the socket based communication using the customizable socket based interaction plugin. An xml example code is given below. Three new elements appear in this code: PartsNode, SocketNode, SocketParam, and SocketInterfaceNode. These elements will be explained in the following sections.

```
<Node label="Socket Test" description="no description..."
xsi:type="SocketInterfaceNode" selectedChildIds="all" disableIds="temporary-
lines,temporary-colors">
   <childs>
      <Node id="socket" xsi:type="SocketNode" url="127.0.0.1" port="12000"
reportingPeriod="5" interactionPeriod="2" partsNodeIds="line-parts-socket-test">
      <transmittedParams>
            <SocketParam type="byte" valueByte="1" label="Byte Param" hint="Enter
0-255 integer"/>
            <SocketParam type="short" valueShort="2" label="Short Param"
hint="Enter short value"/>
            <SocketParam type="int" valueInt="3" label="Int32 Param" hint="Enter
Int32 value"/>
            <SocketParam type="float" valueInt="4" label="Single Param" hint="Enter
Single value"/>
            <SocketParam type="bool" valueBool="true" label="Boolean Param"/>
            <SocketParam type="string" valueString="xyz" label="String Param"
hint="Enter string"/>
         </transmittedParams>
      </Node>
      <Node xsi:type="PartsNode" id="line-parts-socket-test" duration="180">
         <partsDefinition
            partsType="LineTraceNode"
         />
      </Node>
   </childs>
</Node>
```

D.2.1 PartsNode configuration

The parameters of the PartsNode node and their description are summarized in the next table.

Table D-3: PartsNode parameters.

Parameter	Description
`label`	Text to be shown in the view button
`selectedChildIds`	Child nodes to be considered in the view. Set to “all” by default
`disableIds`	Global nodes to disable in the view
`duration`	Duration of the view in seconds

The PartsNode node may include the definition of line parameters using a lineParams. The configuration of this section is the same presented before.

In addition, the PartsNode node may have as childs different nodes. For example, a PartsNode of type LineTraceNode, that defines the characteristics of the graphical representation binary files used for arrow representation. See its configuration in Annex C.2. Or, a PartsNode of type ChartTraceNode that defines the characteristics of the graphical representation binary files used for bar charts representation, and the characteristic of the chart to be used. See its configuration in Annex C.5. Identically, nodes to describe the representation of bars could be included.

Finally, a TrafficNode that defines the mobility files to be read, could be included.

D.2.2 SocketNode configuration

SocketNode is a plugin that enables a socket-based interaction with simulators and allows to customize the list and types of parameters passed to the simulator. SocketNode has no visual representation. If the user of the visualization tool needs to edit parameters, SocketNode must be wrapped by SocketInterfaceNode. The parameters used to configure this node are summarized in Table D-4.

Table D-4: SocketNode parameters description.

Parameter	Description
`url`	Url of connection host.
`port`	Number of port.
`command`	Path to simulator (in local mode) or php call including as parameter the url of the ftp server in the visualization tool side (in remote mode). Command is invoked by press “Launch Simulator” button in view of SocketInterfaceNode.
`arguments`	Arguments passed to simulator executable
`partsNodeIds`	Id (comma separated) indicating PartNodes. Values must be the same as PartNodes id. PartNodes must be SocketNode siblings.
`reportingPeriod`	Parameter passed to the simulator - used to determine the time intervals of data portions sent by the simulator. Editable in view of SocketInterfaceNode before simulation starts.
`interactionPeriod`	Parameter passed to the simulator - used to determine number of data portions after which the VP sends the current parameter values to the simulator. Editable in view of SocketInterfaceNode before simulation starts.
`transmittedParams`	List of SocketParam nodes that define parameters send to the simulator.

D.2.3 SocketParam configuration

SocketParam allows to define the custom parameter sent from the VP to the simulator. Configuration parameters are described in Table D-5.

Table D-5: SocketParam parameters description.

Parameter	Description
`type`	Defines type of parameter value. Possible values: • bool – 1 byte – 0 as false, 1 as true • byte • float – 4 bytes - represents a single-precision 32-bit IEEE 754 value, • int – 4 bytes - signed 32-bit integer, • list – 1 bytes – index of selected list option, • short – 2 bytes – signed 16-bit integer, • string – first byte contains number of chars, then a set of characters into a sequence of bytes.
`valueBool`, `valueByte`, `valueFloat`, `valueInt`, `valueShort`, `valueString`	Initial value of parameter
`list`	Comma-separated strings - used in view of SocketInterfaceNode as dropdown options.
`label`	Label used in view of SocketInterfaceNode.
`hint`	Additional hint used in view of SocketInterfaceNode when input filed is empty.
`isHidden`	Used in view of SocketInterfaceNode to prevent from displaying input field for param.
`isInitialOnly`	Used in view of SocketInterfaceNode to disable input field after simulation was started.

D.2.4 SocketInterfaceNode

Allows to control the connection with the simulator and change the parameters declared in SocketNode. Displays buttons: “Launch Simulator” (only if command param in SocketNode is defined), “Connect”, “Start/Stop”, “Pause”, "Resume” to control the simulation flow. It also display the form with input fields related to SocketNode.url, SocketNode.port and SocketNode.transmittedParams list.

D.2.5 UPV example

The following code is the xml configuration of the remote interaction with UPV simulator at the time of writing this document. The appearance of the widget for this code has been shown in in Figure D-1.

```
<Node label="Remote interaction" selectedChildIds="all" disableIds="car-traces,temporary-lines,temporary-colors">
    <childs>
        <Node label="Socket-based remote interaction (Customizable fields)" xsi:type="BaseWidgetNode" viewType="gui/widget/SocketGenericView" selectedChildIds="all">
            <childs>
                <Node id="socket" xsi:type="SocketNode"
                    url="158.42.160.99"
                    port="12000"
                    reportingPeriod="1"
                    interactionPeriod="2"
                    partsNodeIds="line-parts-socket-test,chart-parts-socket-test"
                    command="http://158.42.160.99/simulator/index.php?interaction=3"
                    arguments=""
                    >
                    <transmittedParams>
                        <SocketParam type="int" valueInt="0" label="car id" hint="enter car id"/>
                        <SocketParam type="list" label="AIR Interface (0:cmW, 1:mmW, 2:both)" list="cmW, mmW, both"/>
                    </transmittedParams>
                </Node>
                <Node xsi:type="PartsNode" id="line-parts-socket-test" duration="180">
                    <partsDefinition
                        partsType="LineTraceNode"
                    />
                </Node>
                <Node xsi:type="PartsNode" id="chart-parts-socket-test" duration="180">
                    <partsDefinition
                    partsType="ChartTraceNode"
                    align="rb" hOffset="0" vOffset="0"
                    barWidth="10" barHeight="120" barGap="5"
```

```
                    numBars="33" maxValue="100" hexColors="#ffa500"
                    yLabelDistance="5" xLabelDistance="5"
                    xLabel="Each bin represents a range of 5 m"
                    isMaxValueVisible="true"
                    yLabel="user PRR(%)"
                    />
                </Node>
            </childs>
        </Node>
        <Node      xsi:type="TrafficNode"      id="car-traces-upv"      src="upv-phase-
1/mobility_trace.bin" interval="0.1" idPrefix="c" mobileType="CarSedan01" />
    </childs>
</Node>
```

The following code includes the lines of the previous code that should be different for the local interaction with UPV simulator at the time of writing this document. Specifically, note that the url, command, and arguments are different to those shown for the remote interaction.

```
<Node id="socket" xsi:type="SocketNode"
    url="127.0.0.1"
    port="12000"
    reportingPeriod="1"
    interactionPeriod="2"
    partsNodeIds="line-parts-socket-test,chart-parts-socket-test"
    command="C:\UPV_simulator\UPV_simulator.exe"
    arguments=" -t 3 -d 180 -i C:\UPV_simulator\Traces -o C:\UPV_simulator\OutputFiles
-v C:\ftpFolder -r localhost -m mobility_trace_100users.txt">
```