Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2019.DOI

Real-Virtual World Device Synchronisation in a Cloud-enabled Social Virtual Reality IoT Network

ANDERSON AUGUSTO SIMISCUKA[1], (Student Member, IEEE), TEJAS MORESHWAR MARKANDE[1] and GABRIEL-MIRO MUNTEAN[1], (Senior Member, IEEE)
[1]School of Electronic Engineering, Dublin City University, Dublin, Ireland.

Corresponding author: Anderson Augusto Simiscuka (e-mail: anderson.simiscuka2@mail.dcu.ie).

This work was supported by the Irish Research Council and Dublin City University, grant number EPSPG/2015/178, and in part by the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 688503 (http://newtonproject.eu).

ABSTRACT Virtual Reality (VR) is currently being used in many different areas such as car prototyping, gaming, medical training, teaching, etc. Internet of Things (IoT) devices such as systems-on-a-chip (e.g. Raspberry Pi), smart appliances and sensors support a wide range of services, including machine automation, remote monitoring and control. This paper introduces a novel social VR-IoT environment, which allows users to share and control local or remote IoT devices in a virtual platform. Two approaches using the VR-IoT solution are presented: one local network-based and one cloud-based. The proposed VR-IoT environment contains VRITESS, the novel VR-IoT Environment Synchronisation Scheme, which facilitates a consistent and integrated experience for users by enabling control of real IoT objects with VR headsets. Control of some IoT objects in extreme environments or devices which are complex to operate, can be simplified in a virtual environment. The VRITESS synchronisation scheme maintains the real objects updated, following instructions given in the virtual world and vice-versa. Testing involved local network-based and cloud-based testbeds created with a VR headset and IoT devices at the Dublin City University's Performance Engineering Laboratory in Ireland. Test results demonstrated that lower latency is experienced in the local-network testbed in comparison with the cloud testbed. Further tests regarding the communications protocols implemented in the cloud testbed indicated that MQTT generates less delay and data traffic than REST.

INDEX TERMS Multimedia IoT, three-dimensional visualisation, virtual reality (VR), VR-IoT.

I. INTRODUCTION

ANALYSTS expect the Internet of Things (IoT) to network 500 billion devices and seamlessly deploy and use data aggregation, data analysis, data insight and data delivery by 2030 [1]. Devices such as smart appliances, wearables, health monitors, smart cars, etc. are changing the way users interact with devices. However, certain IoT devices are still complex to operate and do not provide a simple and mature user experience [2], [3]. Multimedia solutions such as [4], [5], [6], enable users to enjoy a rich media experience. These solutions combined with Virtual Reality (VR) technology can expand the perception of reality thanks to the introduction of realistic scenarios with auditory, tactile and visual capabilities, which translate the real world into an immersive virtual world [7], [8]. According to [9], it can be easier to interact and understand IoT devices by using gestures, language and other human senses. IoT objects such as landslide sensors or devices for water level monitoring, which can be located in hazardous places, and other IoT objects, which can be too complex to use, can be operated in an easier way in a virtual environment. The virtual environment also enables the social IoT to become reality: users will be able to consume content and share access to services and devices.

This paper proposes the VR-IoT Environment Synchronisation Scheme (VRITESS), an innovative mechanism, which enables users to seamlessly operate real IoT devices on a virtual environment. Through VRITESS, users can visualise data provided by beacons and operate sensors and other features available in single-board computers (e.g. Raspberry Pi). The visualisation is performed in a VR environment with a user-friendly interface facilitating the interaction with real objects. When a user manipulates a device in the virtual environment, besides the actions being executed in the virtual world, they will also be executed on the real objects. Operations in the real objects will also be reflected in the virtual ones, with VRITESS being responsible for the synchronisation between real objects and virtual objects. Fig. 1 illustrates the concept of interconnecting the IoT devices *j*, which are networked by smart gateways *i*, to the VR platform, where they can be visualised and manipulated. The details of the IoT and VR platforms integration will be further discussed later in this paper.

VRITESS was deployed and tested in two different situations using the VR-IoT solution: a local network-based approach and a cloud-based approach, allowing the analysis of performance metrics, such as network latency. Both testbeds contain beacons, an Oculus Rift and Raspberry Pis. Tests were conducted in the Performance Engineering Laboratory at the Dublin City University, Ireland. The VR environment recreated the laboratory in 3D, with virtual devices matching the real IoT ones located in the office. A synchronisation algorithm, which is part of the VRITESS solution, maintains the devices up-to-date, sending actions and events that happened in the virtual environment to the real objects, and vice-versa, updating changes of their states and maintaining consistency. Comparative testing results show that the cloud-based solution has slightly increased latency in comparison to the local-based deployment.

The remaining sections of this paper are organised as follows. Section II presents related works and Section III discusses the architectural design of the proposed solution. Section IV describes the proposed synchronisation mechanism and Section V details the architectures of the testbeds, testing scenarios and results. Conclusions and directions for future work end this paper in Section VI.

II. RELATED WORK

The related works of VR and IoT solutions relevant to this paper are classified in the following topics: multimedia IoT and protocols, VR background, VR and IoT integration, cloud-based IoT and VR solutions, and social impact. These categories contain solutions and approaches relevant to the scheme proposed in this paper, hence the importance of reviewing these works.

A. MULTIMEDIA IOT AND PROTOCOLS

A number of multimedia solutions supporting IoT systems have been recently introduced, such as a web application framework based on the Google Web Toolkit for improving the interaction between users and IoT devices [10]. Authors presented a visualiser for operating smart objects in a graphical and functional way. Tools for managing the devices and their interactions and for web services communications were also included. For testing purposes, healthcare applications

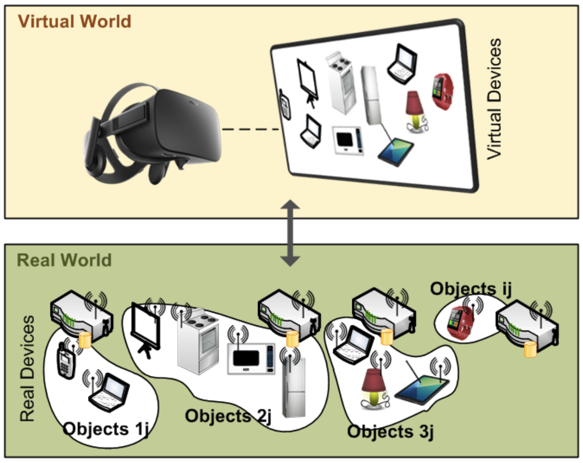

FIGURE 1. Real-world and virtual-world devices.

were integrated to the framework. Other IoT applications are yet to be integrated to the solution.

The concept of Multimedia IoT (MIoT) was formalised in [11]. The authors also introduced a QoE model for MIoT, with layers regarding physical devices, network, combination, application and context. For testing purposes, the authors developed a vehicle application with remote control, and presented the best QoE scenario among different conditions in relation to bitrate, synchronism speed, visualisation speed and map synchronisation quality.

In [12], authors proposed a home automation system based on a 3D virtual world. The goal is to help users interact with smart homes in a friendly and easy-to-use environment. A computer runs the graphical user interface, which is controlled with a mouse. MPEG-V standard is used along with a user-defined protocol. Authors did not consider VR for user interaction and did not implement different protocols in the solution to analyse the performance of the platform.

Authors in [13] introduced a web-based interactive framework for visualisation and authoring of indoor IoT environments containing sensors and devices in homes or small offices. An indoor 3D scene was configured to display attributes of the available physical sensors and actuators, also allowing the visual programming of virtual sensors and actuators. The testing of the proposed framework, however, did not consider a synchronisation scheme and did not include real physical sensors and actuators, which were emulated by the server.

Other mass market devices that have been employed for controlling and interacting with smart homes are the voice-driven virtual assistants and smart speakers [14], such as the Google Assistant and Amazon Echo, which, however, lack the graphical capabilities of other multimedia solutions.

Examples of communications protocols employed in IoT are the Hypertext Transfer Protocol (HTTP) and the Message Queuing Telemetry Transport (MQTT), which aim to meet the requirements of IoT networks in constrained environments.

REST (Representational State Transfer) is a communication protocol used in IoT, which runs over the HTTP protocol and uses a request/response model. REST is stateless, therefore, each request from a client to the server must contain all data necessary for the server to process the request, as the connection is always closed after the request. REST uses methods from the HTTP library such as GET, POST and DELETE requests, and can be implemented in most programming languages and embedded devices, as the library is part of most operating systems [15]. REST, however, requires high usage of network resources as it establishes a connection each time there is some transfer of data [16].

MQTT is a lightweight messaging protocol that uses TCP as the transport layer in the TCP/IP model [17]. MQTT is stateful, maintaining connections alive, and uses a publish/subscribe messaging model, consisting of subscribers, publishers and a broker. The subscriber receives the data that is sent by the publisher if it is subscribed to the topic(s) being published, and the broker remains centred within this message exchange to manage the events. MQTT aims to deliver messages reliably, minimise network bandwidth usage and be easy to implement on constrained devices [18]. MQTT uses short headers and data is transferred as byte arrays unlike REST, which needs to define content type. MQTT message types include CONNECT, PUBLISH, SUBSCRIBE, UNSUBSCRIBE and DISCONNECT. Minimum packet size is two bytes for a message, one byte for control field and one byte for packet length field, which is all that is needed in a DISCONNECT message.

The approaches reviewed in this section were used as inspiration for the development of the multimedia elements of the proposed solution, especially in relation to protocols, user experience and application scenarios. The need for user-friendly tools for IoT devices is evidenced by the research efforts in the creation of multimedia solutions to control these devices, especially in smart homes.

B. VR BACKGROUND

VR can be defined as a way of simulating the real world by the application of the immersion theory into a virtual 3D environment. In a certain virtual area, users interact with the space using their human senses in a similar way to the real world, and therefore, VR solutions mainly focus on three human senses: haptics, sight, and hearing. The main components and layers of a VR solution are the VR engine, software and database, in the system layer; input and output devices, in the middle layer; and users and tasks, in the application layer. VR can be used in many fields including military training, collaborative study rooms, healthcare, gaming, virtual tours and museums, prototyping, product modelling and viewing, etc. [19], [20].

Many works have discussed the relevance of user interfaces (UI), audio and visuals for VR applications, such as the ones presented in [21], [22], [23]. The UI of VR systems needs to consider how users will control the applications, how space should be perceived and the integration with the entire experience. Audio is also important as it works as a guide to users and its quality impacts user experience. For example, a VR application running the reproduction of an orchestral recital should prioritise audio without compression, while a gaming application would focus on the location of the element generating a sound for user guidance. Visuals can be made of 3D polygons or omnidirectional (360°) captured images of the real world. The graphics must not have distortions and need to accurately map the area available for user movement. The processing of this type of omnidirectional visual content requires very high resolution (e.g. 4K, 8K), massive storage, and consume a large amount of network bandwidth. Therefore, visual content delivery is a challenge and requires powerful hardware and/or compression without much loss in quality, such as the proposed compression approaches by the Joint Video Exploration Team (JVET) of ITU-T SG 16 WP3 and ISO/IEC JTC1/SC29/WG11.

Authors in [24] introduced a VR approach for exploring software-built cities using VR headsets and interactions with gestures. Authors also created a reusable gesture model for operating 3D environments. In order to test the solution, participants of the tests were asked to rate the usability of gestures and the overall VR experience. Results demonstrated that participants praised most of the introduced gestures.

Regarding VR synchronisation, authors in [25] implemented a computing offloading scheme via an mmWave 802.11ad 60GHz wireless link in order to share 360° VR content with a PC. Synchronisation algorithms were employed at packet level, handling real-time video transmission without packet loss.

Based on the idea of creating innovative ways of controlling IoT devices, VR seems like a suitable approach due to the ever-increasing consumer adoption of VR technology. The works reviewed in this section demonstrate the importance of a pleasant and user-friendly VR experience, and the feasibility of its use in diverse fields.

C. VR AND IOT INTEGRATION

VR and IoT integration is a novel and young research field and therefore there is little research work available in the literature. Some interesting recent works relevant to this paper have been identified and are presented in this section.

The creation, implementation, and evaluation of a Unity game on an engine-based application layer for a brain-computer interface was proposed in [26]. The game, which integrates 3D graphics, VR, and IoT devices, was only tested in a local-network environment.

An end-to-end system architecture that underlies IoT networks for seamless VR space was proposed in [27]. The architecture contains five stages: acquisition, classification, virtual image and video reconstruction, transmission, and consumer processing. The authors, however, did not present a testbed to demonstrate the architecture.

Authors in [28] proposed a layered model for the connection and interaction between people and smart devices. A virtual platform with the locations of smart devices in a city was implemented, creating a Virtual Environment of Things (VEoT). Users are able to interact with the smart objects using VR headsets and gestures, and the platform uses a multi-protocol sensing middleware and RESTful APIs for the connection of devices and real-time communication. Testing was limited to displaying the temperature of sensors, not allowing user interaction with actuators or other devices.

A VR engine for immersive smart city visualisation was proposed in [29]. The engine contains network features for big spacial data and online sharing with hash-based peer-to-peer (P2P) capabilities. Users can navigate in a map with real geographic space using avatars, and performance evaluation tests demonstrated high usability and user satisfaction. The engine, however, does not allow users to interact with smart devices of the city, being more useful for city planning purposes.

A platform for hyper-connected IoT-VR was presented in [30]. Users and devices can be interconnected in a virtual space, which contains customisable remote services. The authors built a testbed, which included a cloud server, but did not analyse network latency and performance using different protocols, and did not implement a synchronisation scheme to guarantee consistency between real and virtual devices.

The approaches for integrating VR and IoT presented in this section have limitations in their implementations, not showcasing the full potential for a VR-IoT solution. The first approach does not have cloud capabilities; the second one focuses on the architectural aspects of VR and IoT integration; the third solution did not fully implement user interaction controls for IoT devices, only displaying sensor data; the fourth approach is only applicable for city planning and overall viewing of smart data; and the fifth approach does not guarantee consistency between real and virtual devices and does not test a variety of protocols.

The solution presented in this paper needs to allow end-users to interact with IoT devices using intuitive controls, guarantee that real and virtual devices are synchronised, and test the performance of different communication protocols used for the VR and IoT integration. Another feature possible with a VR-IoT system is the ability of controlling objects remotely, something that can be addressed by hosting certain functionalities in cloud-based servers, therefore, the next section focuses on cloud solutions for IoT and VR.

D. CLOUD-BASED IOT AND VR SOLUTIONS

There are a number of works exploring IoT capabilities in the cloud. In [31], authors explained that the virtualisation of IoT devices allows users to easily share the devices. They also presented the multitenancy feature of cloud computing as an enabler for resource sharing to multiple users over spatial and time distributions.

Authors in [32] presented a cloud-based smart home platform for integrating smart home IoT services and managing operations. Users can manipulate and visualise the devices in the platform with a user interface. Authors also compare the data traffic and latency generated by HTTP and MQTT protocols.

An approach on sharing IoT devices among end-users was presented in [33]. The service-oriented approach allows IoT devices to provide data and resources, which are available in a cloud platform, at any given time from any place with connectivity. The approach gives the opportunity for application developers to reuse information provided by the users and create applications on top of the solution.

Authors in [34] proposed the concept of local IoT automation clouds. The solution design was demonstrated in a compartment climate control. Automation and real-time performance were also analysed.

VR has also benefited from cloud technologies. Computational off-loading, image processing and mobile capabilities are among the features found in cloud-based VR solutions. Authors in [35] investigated how extensive rendering being off-loaded to a cloud/edge/local server can make the VR/AR processing to be lighter on the headsets. There are, however, challenges from bitrate to latency requirements, and possible solutions to enable cloud/edge-based wireless VR/AR include asymmetric video encoding and rendering, and hybrid-casting.

In [36], authors presented a solution which uses off-loading as a service, in order to to resolve the mismatch between how mobile devices demand computing resources and how cloud providers offer them. Authors also designed a VR framework able to seamlessly operate a 3D game by computation off-loading.

The works reviewed in this section demonstrate that cloud technologies have been used in IoT and VR, mainly for data virtualisation and off-loading. The benefit of accessing data from anywhere brings challenges such as latency in communications, which can be an issue especially in VR applications. Therefore, the solution presented in this paper needs to compare the performance of local-based and cloud-based approaches, and demonstrate the feasibility of having features in the cloud.

E. SOCIAL IMPACT

In [37], the author affirmed that VR will change fundamental rules on how life has been lived throughout the entire length of human history, creating social consequences.

According to the work presented in [38], IoT data can be used to create novel ways of connecting people, with automated software networking people and objects. IoT objects such as smart watches, wrist bands, appliances, healthcare devices and cars can bring new ways of user interaction.

Similarly, VR provides a rich media experience that increases the sense of reality with auditory, graphical and tactile capabilities that translate the real world into a virtual one. The possibility of manipulating virtual devices using intuitive controls and human senses such as tactile gestures and language can make it easier for users to operate and interact with real IoT objects.

Additionally, the ability to share devices brings people together, and enables IoT services to be made accessible to users who do not possess certain devices.

Bringing IoT and VR technologies together can expand their societal impact and help each other become more user-friendly and mainstream to society.

A thorough review of related works was presented in this section and important IoT solutions were discussed in the context of this paper, highlighting their gaps, which are addressed by this work. Some of the related works do not perform analysis of network performance [12], [26], other research outputs do not discuss comparatively different protocols [12], [27], [30], and the remaining articles do not consider synchronisation in order to guarantee consistency between real and virtual devices [13], [30].

III. SOLUTION ARCHITECTURE

This section introduces the VRITESS solution architecture, its components and the communication process among them.

A. ARCHITECTURE DESCRIPTION

VRITESS is implemented on top of an IoT architecture, as illustrated in Fig. 2. The basic IoT architecture has been presented and used in [39], [40], [41], [42] and was extended in [43]. The architecture, which deploys VRITESS, has the following major components: *IoT objects* (e.g. smart appliances, wearables, health monitors, sensors) providing services to users and other devices – each of these IoT objects have two instances: real (the real IoT device) and virtual (the VR representation of the real object); *Smart Gateways* networking the IoT devices, allowing communication among connected objects; the *IoT Integration Platform (ITINP)*, a platform that contains a cloud-based server for the integration of services, including the interconnection of the smart gateways, and the *VR IoT Platform (VRITIP)*, composed of a VR server and VR headsets for the rendering of 3D virtual spaces containing the virtual representation of IoT objects, allowing users to interact with them using straightforward controls. VRITIP also maintains virtual objects up-to-date with the operations executed in the real ones, showing them in the virtual environment.

VRITIP and ITINP communicate over local networks or the cloud, using IoT protocols such as the MQTT protocol. In the local network approach, a local database stores the operations executed in the virtual and real devices, and based on timestamps, it keeps both types of devices updated. In the cloud approach, a cloud-based IoT protocol-enabled database keeps the operations and timestamps, while both ITINP and VRITIP constantly access the data. VRITESS is implemented in both ITINP and VRITIP, keeping devices updated with the information stored in the local or cloud-based databases. Network Time Protocol (NTP) is used to synchronise ITINP and VRITIP clocks over the networks, avoiding synchronisation errors [44]. Details on the implementation of the testbeds representing the architectural design are presented in Section V.

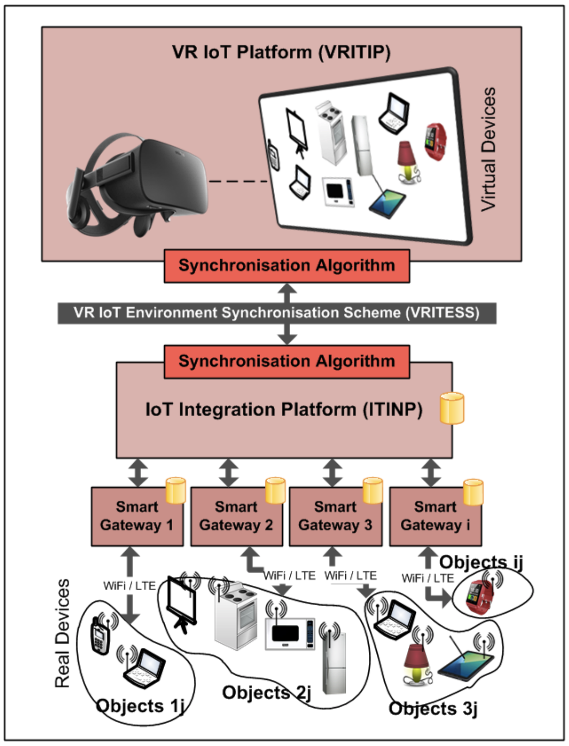

FIGURE 2. The VRITESS architecture.

B. COMMUNICATION PROCESS

A status structure is used by the gateways to store the actions regarding the devices. ITINP receives operations executed in real devices, VRITIP receives operations executed in virtual devices and VRITESS enables the two-way communication and synchronisation between these two. Both virtual and real devices need to be able to update their functionalities based on the latest information in these status structures.

Several devices can benefit from this communication structure, with operations such as turning on/off devices and appliances, reading sensor temperatures, movements and activities, turning on/off motors for opening and closing doors and curtains, etc. Users exist on both virtual and real world, so the status structure and its associated update mechanisms support tracking the way users interact with the devices in either world. The device status structure used in the local and cloud-based databases are presented in the following example (illustrating a virtual user turning off a virtual object):

```
{
"timestamp":  "2019-04-15T09:15:12.147"
"last_change":"2019-04-15T09:14:23.133"
"user_id":                "andersonsimis"
"type_of_user":                 "virtual"
"type_of_device":               "virtual"
"tags:              ["POWER_INSTRUCTION"]
"device_id":        "0036:5E25:0000:21DC"
"data"                              "OFF"
}
```

This operation will result in the real device also turning off and the appropriate status structure update:

```
{
"timestamp":  "2019-04-15T09:15:12.235"
"last_change":"2019-04-15T09:14:23.224"
"user_id":                "andersonsimis"
"type_of_user":                 "virtual"
"type_of_device":                  "real"
"tags:              ["POWER_INSTRUCTION"]
"device_id":      "0036:5E25:0000:21DD"
"data"                              "OFF"
}
```

This example illustrates a virtual user that turns off a virtual object, resulting in the same operation being executed on the real object. Operations performed in the virtual environment (e.g. turn on a device, change the temperature of a thermostat, etc.) are synchronised in ITINP and then updated in the database of the smart gateway responsible for interconnecting the corresponding real device, passing the instruction to the object using an IoT protocol such as MQTT.

The solution architecture contains databases at the level of the gateways and in the cloud, so it is possible to record the actions in real and virtual devices. When an action is performed, details are registered following the status structure presented in this section. Section V describes in detail the implementation of a local database (MySQL) in the local network-based solution and a cloud database (Adafruit IO) [45] in the cloud-based solution.

The database implementation also allows users to be identified when performing actions, therefore, when logging into the applications, users can only control the devices that are shared with or owned by them. The social aspect of the platform is the possibility of granting access to other users to control devices in the virtual platform.

The VRITESS synchronisation algorithm keeps the devices updated, according to the operations executed in the virtual and real versions of the devices, respectively.

IV. VRITESS SYNCHRONISATION

The VRITESS synchronisation algorithm is employed on both real world devices, at the level of ITINP (acting on the real objects), and virtual world devices, at the level of VRITIP (acting on the virtual objects), in order to maintain consistency.

Algorithm 1 presents the synchronisation of real-world and virtual-world IoT objects. It works by sending to ITINP and VRITIP the latest action performed by real and virtual users, based on the timestamp value of these actions. ITINP and VRITIP employ the Network Time Protocol (NTP), therefore, their clocks are synchronised over the network.

As seen in algorithm 1, the timestamps and data structure maintain the real and virtual devices up-to-date and synchronised with each other. The communication process follows the structure introduced in Section III.B. The *new_t*

Algorithm 1. Synchronisation of real and virtual-world IoT Objects

```
Require: old_t ← The last timestamp sent by ITINP
         and VRITIP; data[] ← The set of data,
         following the structure in section III.B
Output : new_t ← New timestamp sent back to ITINP
         and VRITIP; updated_data[] ← Updated
         data sent back to ITINP and VRITIP
1: new_t = old_t
2: foreach data in data[] do
3:     if (data.timestamp > old_t) then
4:         updated_data[].add(data.value)
5:         new_t = max(data.timestamp, new_t)
6:     end
7: end
8: return new_t, updated_data[]
```

variable stores the most recent timestamp when new actions are received (i.e. newer than the *old_t* variable, which stores the last executed action). Some of the instructions may be informative (e.g. a virtual gauge representing a sensor temperature) or actions (e.g. turn on/off a virtual appliance resulting in the same action on the real appliance).

All the data sent by ITINP and VRITIP is organised into an array of objects containing all *data* to be processed by the algorithm. Each object of the array contains the structure for communication, with fields including: data.timestamp (current timestamp), data.last_change (timestamp of last data modification), data.device_id and data.user_id (unique IDs for device and user, respectively – strings used as keys), and other strings such as data.type_of_user, data.type_of_device, data.tags (indicating the type of action, e.g. power related or the sensor type) and data.data (indicating the action to be executed, e.g. turn off/on).

The algorithm is triggered on an event-based fashion, therefore, every new event, e.g. turning on a light, will call a function that inserts it into a database. After this insertion process in the local or cloud database, the synchronisation mechanism is triggered to send the latest action to the corresponding virtual or real device, if a real or virtual device was manipulated, respectively.

V. REAL LIFE TESTBEDS DESCRIPTION

The VRITESS solution was deployed in two different approaches in the Performance Engineering Laboratory at the Dublin City University, Ireland.

Both a local network-based solution and a cloud-based solution were created, so that comparative performance differences between the two approaches can be assessed. In the network-based approach, a browser-based VR application was created, while in the cloud-based approach a 3D application was developed with Unity, a 3D engine. Both were tested with an Oculus Rift, real IoT devices and a Raspberry Pi, which interconnects the IoT devices using its General

Purpose Input Output (GPIO) pins to control LEDs and a servo-motor, and Bluetooth to receive data from beacons.

A. LOCAL NETWORK-BASED APPROACH

The local network-based implementation only allows local devices to be visualised in the VR headset. The VR headset and the computer that powers it communicate directly to the IoT devices using a local wireless network.

The testbed illustrated in Fig. 3 has the following major components: an Oculus Rift [46], two Raspberry Pis [47] and four Beeks IoT beacons [48], which are connected to the Raspberry Pis via Bluetooth Low Energy. A Dell Alienware computer [49], which is connected wirelessly with the Raspberry Pis in a local 802.11n network, supports the Oculus Rift using an HDMI port, and also renders the browser-based 3D virtual world. The user shown in Fig. 3 uses the Oculus Rift Touch controllers to select the available actions on the Oculus headset. This is replicated on the computer screen, which displays the same virtual environment seen on the headset. The actions (e.g. power off device, turn LED on, etc.) update the representation of the virtual Raspberry Pi displayed on the Oculus headset and are also executed on the real Raspberry Pis. The VR application also displays information (e.g. temperature) generated by the IoT beacons sent via Bluetooth.

The specifications of the Alienware computer, Beeks Beacons and the Oculus Rift used are available in Tables I, II and III, respectively.

The applications developed are as follows. The first application is a Java VR communication application, which was deployed on the Raspberry Pi, in order to read from and send commands to the in-board LED, read temperatures of the beacons connected to the Raspberry Pi via Bluetooth, and send the shutdown and restart instructions to the operating system of the Raspberry Pi. This Java application also communicates with the main computer (i.e. Alienware) using Java ServerSockets and Input/OutputStreams for receiving and sending instructions to the virtual IoT devices visualised in the VR headset.

The Alienware computer is responsible for running a Glassfish 4.0 web server [50] with the second Java application, which reads and writes to a MySQL database containing the status structures presented in Section III.B. These statuses are related to the real world IoT objects. This application updates the database with the received instructions from the Raspberry Pi Java application, and also sends instructions back to the Raspberry Pi.

A JavaServer Faces (JSF) web application [51], which also contains the VR application in HTML pages optimised for VR, is responsible for reading and writing the statuses in the database regarding the virtual objects. The JSF application receives and sends the current statuses of the virtual objects coming from the interactions of the user with the VR headset manipulating the options in the HTML pages optimised for VR.

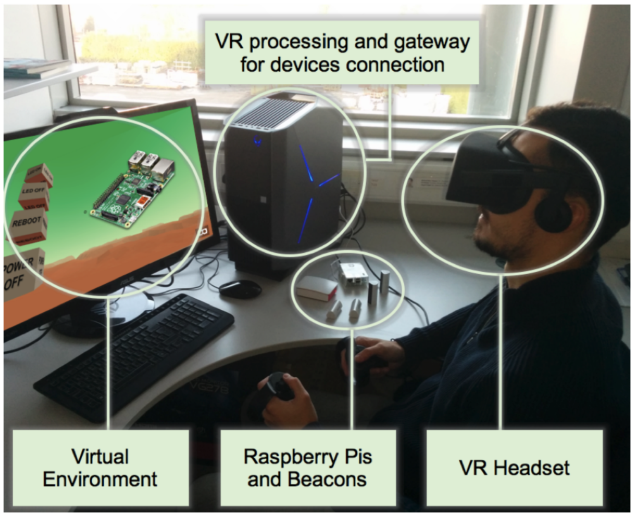
VR processing and gateway for devices connection
Virtual Environment
Raspberry Pis and Beacons
VR Headset

FIGURE 3. The implemented local network-based testbed.

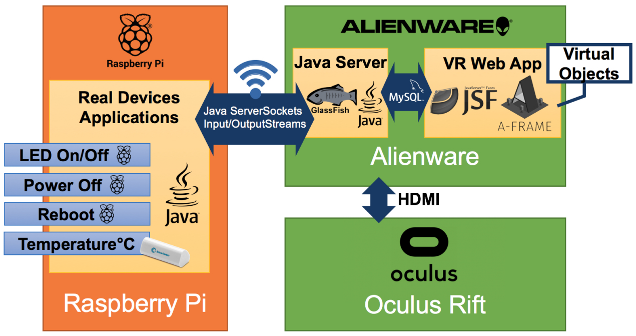

FIGURE 4. Local network-based solution architecture.

Testbed development started with the creation of the two Java applications in the Glassfish web server (i.e. Alienware) and the Raspberry Pi, as seen in Fig. 4. As mentioned earlier, threaded Java ServerSockets are used on both Glassfish server and Raspberry Pi, therefore several devices can connect to the Glassfish instance. The Java applications convert the actions performed by the users into objects, following the structure presented in the previous section, so they can be sent over to the other application (either on the Raspberry Pi or Alienware) and also stored in the database, by the use of the MySQL Java database driver. These objects are exchanged using the ServerSockets and Input/OutputStreams every 0.5 seconds. The objects carry all statuses for the five types of interactions that the web application enables, from virtual devices (in the web application) to real devices and vice-versa. The five types of interactions are: turn led on and off, reboot and restart, on the Raspberry Pi, and temperature measurement on the Raspberry Pi and beacons.

The 0.5s time interval for the local network testbed was selected based on the performance of the network and required communication time between real and virtual devices. Experimental tests did not show any significant improvement in the perception of the real-time execution of tasks for time

TABLE 1. Alienware Specifications

Parameter	Value
Model	Alienware Aurora R6
Processor	Intel Core i7-7700
RAM Memory	16GB
Hard Drive	1TB
SSD	256GB
Graphics Card	NVIDIA GeForce GTX 1080 8GB
Operating System	Windows 10
3D Development	Unity Personal 2017.3

TABLE 2. Beeks Beacons

Parameter	Value
Battery	3.6V / 2600mAh - Primary Lithium
Size	2.36" x 0.85" (60mm x 21mm)
Weight	1.0 oz (28 gr)
Temperature Range	-30°C to +77°C
Bluetooth Type	Bluetooth Low Energy 4.1
Bluetooth Sensitivity	-97dBm
Bt. Max Power Output	+5dBm
Bluetooth Antenna	0dBm Single Antenna, Omni Directional
Bluetooth Data Rate	1Mbit/s / 2Mbit/s
Bluetooth Security	128 bit AES
Power Consumption RX	7.5mA RX Active Mode
Power Consumption TX	6.5mA TX Active Mode
Power Consumption Sleep	1.6μA (SRAM retention and RTC running)
Power Output	-40dBm to +5dBm
CPU	Dual Code: ARM Cortex M3 and M0
Sensors	*High Accuracy Temperature sensor* *3 Axis Accelerometer* - Detect.: Freefall. Motion, Pulse, Transient - Custom detection: Door opening/closing with counter; human walking detection; driving detection, motor vibration learning *Magnetometer* - Custom detectable modes: Door opening and closing, Metal nearby trigger, car detection, electric motor, efficiency/torque *Light Sensor*
Internal Flash Memory	55KB Flash standard
LED	Red LED

intervals lower than this limit (i.e. after an action is taken in the virtual environment, it takes 0.5s for the action to be executed in the real environment, and vice-versa). However, any rate larger than 0.5s would impact severely the performance of the platform. Several update rates were tested in order to define a balance between user perception and performance (i.e. CPU and memory consumption), ranging from 0.1s to 2s. Update rates larger than 0.5s start to impact negatively the VR application. Such negative effects need to be avoided in order to prevent both poor quality and potential user motion sickness.

The developed JSF web application also runs on the Glassfish server and was built with responsive design for mobile device screens and VR devices. JSF is responsible for binding the user interface and the core of the Java application, which communicates with the IoT devices using sockets in the local network.

The HTML pages of the web application implement Mozilla's A-Frame [52], an HTML-based framework for VR

TABLE 3. Oculus Rift

Parameter	Value
Display	PenTile OLED
Graphics	2160x1200 (1080x1200 per eye) @ 90 Hz
Sound	Integrated 3D audio headphones (user removable/exchangeable)
Input	6DOF (3-axis rotational tracking + 3-axis positional tracking) through USB-connected IR LED sensor, which tracks via the 'constellation' method
Controller input	Oculus Touch motion tracked controllers
Connectivity	HDMI 1.3, USB 3.0, USB 2.0
Weight	470g (1.04lb)

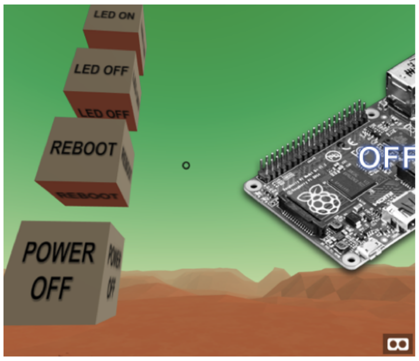

FIGURE 5. Virtual Raspberry Pi - device off.

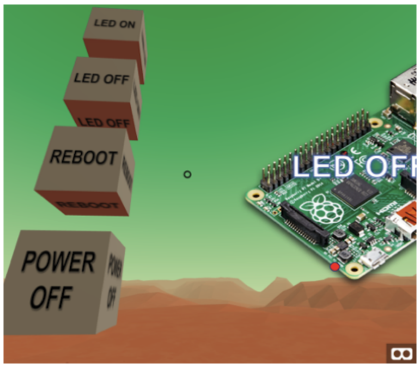

FIGURE 6. Virtual Raspberry Pi - led off.

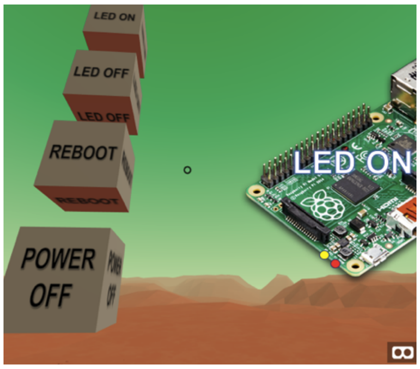

FIGURE 7. Virtual Raspberry Pi - led on.

development. A-Frame includes tags designed to enable head movements in smartphone-based VR headsets by natively reading smartphones' accelerometers and gyroscopes. It is also compatible with regular VR headsets. A-Frame supports and offers creation tools for 3D environments. It also duplicates graphical content for the use of smartphones inside of VR headsets such as the Google Cardboard (each half of the smartphone screen is seen by a different eye, creating a 3D effect with the help of the VR headset). A-Frame is compatible with web browsers such as Google Chrome and Mozilla Firefox, and also with the Oculus Rift.

Fig. 5 illustrates how the Web-VR application displays the virtual device turned off. In figs. 6 and 7, the virtual device has its LED off and on respectively. Users are allowed to control objects which they are granted permission. This allows users to keep devices private or share them with different users, developing a social network of virtual and real IoT devices. In order to enable device sharing, users are identified by user IDs, allowing them to share their devices with other users, therefore, the web application has login and share functions.

The MySQL database stores all data necessary for the monitoring and synchronisation of the solution, with timestamps, IDs for users, devices and tasks, and the permission lists with users and their granted devices.

B. CLOUD-BASED APPROACH

The cloud-based implementation allows for the real-world devices to be visualised and manipulated through the use of the Oculus Rift VR headset. Remote IoT objects can be accessed and receive the VR actions performed, as the actions are stored and synchronised on the Adafruit IO cloud server, as detailed in Fig. 8.

In this testbed, the major components are the VR headset (Oculus Rift) connected to an Alienware computer, a Raspberry Pi 3 Model B+, IoT objects (servo-motor – representing a motor to open and close doors – and light bulbs), and the cloud-based Adafruit IO server. For the cloud-based approach, a 3D virtual room was designed using Unity, containing wall switches, which are used to open/close doors and to turn on/off lights in the real-world, just as they are performed in the virtual environment, as seen in Figs. 9 and 10. Functionalities programmed using C# scripts, such as shutting down and rebooting the real-world system (i.e. Raspberry Pi), are also available in the VR application. Users interact with the 3D VR environment by using the Oculus Touch controllers.

The real-world part of the proposed solution contains a Raspberry Pi to which IoT objects, such as a servo-motor and a light bulb, are connected. The interfacing of the IoT objects to the Raspberry Pi is performed via the Raspberry Pi GPIOs and a program written using the Python programming language.

The cloud-based IoT server Adafruit IO is responsible for storing the commands received from the VR-world devices as well as the real-world devices. Adafruit IO manages "feeds"

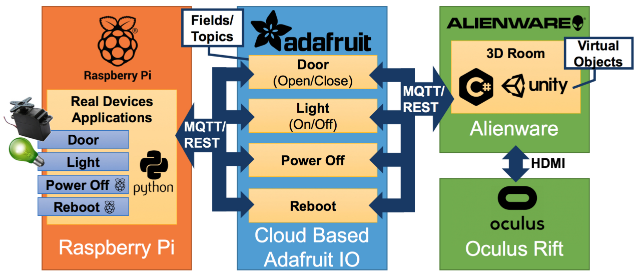

FIGURE 8. Cloud-based solution architecture.

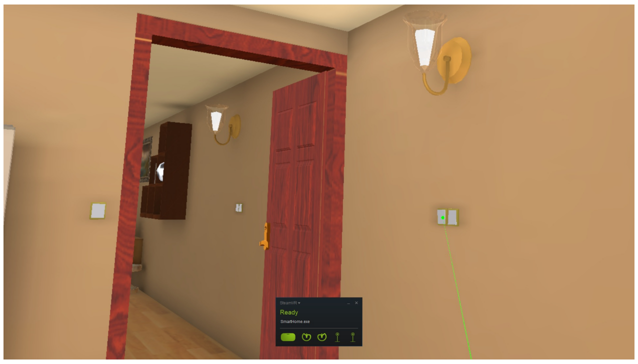

FIGURE 9. Virtual living room - light on.

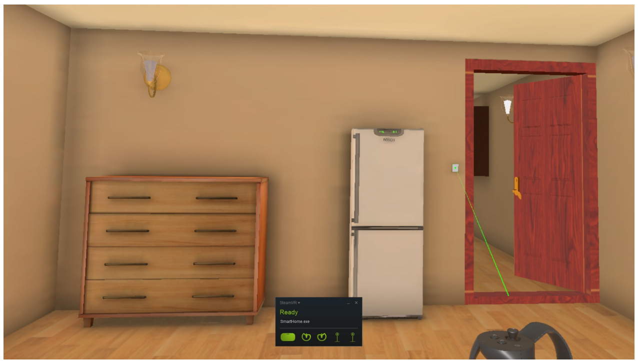

FIGURE 10. Virtual living room - open/close door.

that are available in its dashboard. These feeds store the commands for each device. These commands are accessed by the devices via communications protocols (i.e. MQTT and REST) at lower network layers with instructions such as open/close for the door and turn on/off for the lights, or shutdown and reboot instructions. The Raspberry Pi is also connected to Adafruit IO, and therefore it reads and writes to the feeds related to the devices attached to the Raspberry Pi (e.g. light bulb, servo-motor), updating the feeds with the latest status values synchronised by VRITESS.

Whenever an action is performed by users in the virtual or real worlds, such as pressing switches for turning on/off lights or opening/closing a door, the commands DOOR_OPEN, DOOR_CLOSE, LIGHT_ON or LIGHT_OFF are sent to the specific feeds available in the

TABLE 4. Delay (seconds) - MQTT and RESTful API

Delay (seconds)	RESTful API	MQTT Prococol
Maximum	1.353s	0.0469s
Minimum	0.439s	0.0419s
Average	0.826s	0.0437s
St. Dev.	0.301s	0.0017s

TABLE 5. Data Traffic (bytes) - MQTT and RESTful API

Direction		RESTful API	MQTT Prococol	Avg. MQTT Improvement
Outgoing Bytes	Maximum	385B	270B	34% (in relation to REST outgoing traffic)
	Minimum	362B	216B	
	Average	372B	245B	
	St. Dev.	8B	17.7B	
Incoming Bytes	Maximum	118B	94B	18% (in relation to REST incoming traffic)
	Minimum	95B	77B	
	Average	107B	88B	
	St. Dev.	7.4B	5.2B	

Adafruit IO cloud server, according to the actions performed. A menu is presented when pointing at devices such as doors or light bulbs, with the option of sharing these devices with other users. Users must be granted permissions for device manipulation and for sharing devices.

Fig. 11 presents the implemented cloud-based testbed in the Performance Engineering Laboratory. In order to keep the devices synchronised with their corresponding virtual or real devices, an implementation of communication protocols was necessary. Adafruit IO supports communications using MQTT or RESTful API. Adafruit IO has a current rate limit of 1 request per second (or 60 requests within 60 seconds), therefore lower latency is experienced in the local network approach, with a higher rate of 0.5s for sending/receiving messages, also illustrated in Fig. 12.

Extra testing was performed to test the latency in the cloud-based approach. Using the tests performed in the local approach as a control test case, it could be observed that simple database queries ran much faster in the local approach. Twenty select queries were performed in both local and cloud-based implementations, and the average retrieval time was 4ms for the local test running MySQL and 12ms in the cloud-based approach powered by Adafruit IO.

Tests were conducted in both MQTT and RESTful API in order to demonstrate which approach has the best performance in the cloud-based implementation of the VR-IoT environment, in terms of communication delay and data traffic.

Table IV shows the minimum, maximum, standard deviation and average delays of the ten times the application was executed, as seen in Fig. 13 (sorted from minimum to maximum delay). Delay here refers to the amount of time required to transfer the action performed either in a real-world device or the virtual reality device to the cloud-based IoT server. The delay calculations were performed using Wireshark. The cloud-based IoT server, Adafruit IO, runs on

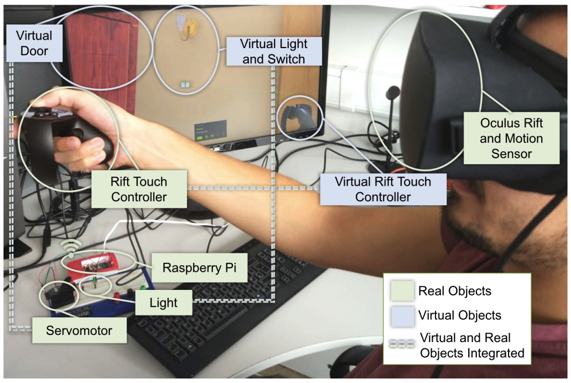

FIGURE 11. The implemented cloud-based testbed.

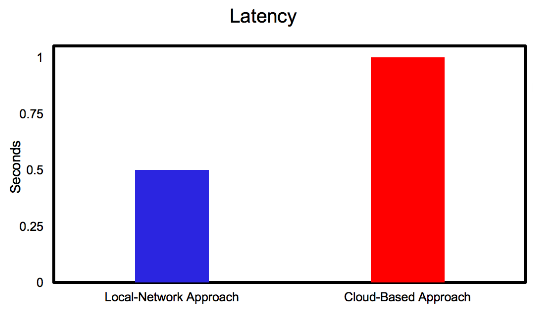

FIGURE 12. Latency in local-network approach and cloud-based approach.

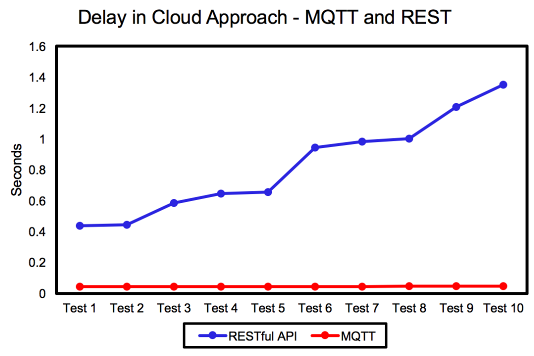

FIGURE 13. Delay (s) in cloud approach - MQTT and REST.

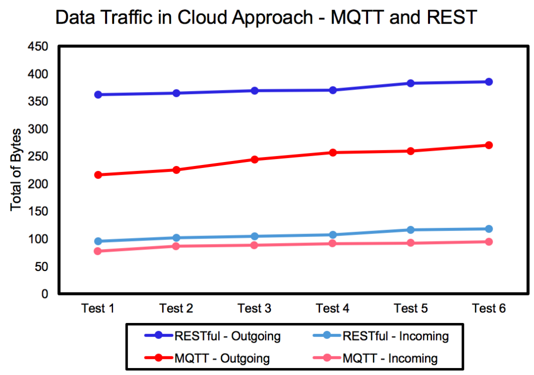

FIGURE 14. Data traffic (bytes) in cloud approach - MQTT and REST.

the ports 1883 and 8883 (for SSL encrypted connections), and the 8883 port was used for the tests. In order to monitor delay, Wireshark's TCP port was set to the cloud-based IoT server port, allowing the capture of all the outgoing packets, data, their time and length. One of the major reasons that impact RESTful API's delay in comparison to MQTT is the fact that a new connection is established each time the devices send data to the cloud and then the connection is terminated. MQTT optimises this process, keeping the connection alive once it is established. Tests demonstrated that, in average, there is 20 times less delay in MQTT communications in comparison to the RESTful API in the cloud-based VR-IoT environment.

Table V shows the number of outgoing and incoming bytes used for data communications when performing actions exchange in the VR-IoT environment (i.e. bytes measured for the six available actions: turn on/off LEDs, turn on/off servo-motor, reboot and shutdown), also demonstrated in Fig. 14 (sorted from minimum to maximum values). The MQTT and RESTful API implementations were compared in terms of outgoing and incoming data, respectively, in relation to the Raspberry Pi and Alienware communicating to the Adafruit server, and vice-versa. When considering average outgoing data, 245 bytes are used for MQTT in comparison with 372 bytes for REST and measuring average incoming data, 88 bytes are needed by MQTT versus 107 bytes by REST. Therefore, in terms of percentages about 34% less outgoing traffic and approximately 18% less incoming data is exchanged by the MQTT solution in comparison with the REST approach. This is also shown in Table V. The main explanation for these results is the lightweight design of MQTT, which maintains one TCP connection alive and uses small size headers in comparison to the RESTful API that runs over HTTP, which creates new TCP connections when needed.

VI. CONCLUSION AND FUTURE WORK

The innovative VR-IoT Environment Synchronisation Scheme (VRITESS) was introduced and described. The proposed VR-IoT platform allows users to operate IoT objects in a virtual environment and contains a synchronisation algorithm that maintains virtual and real IoT objects updated, according to actions and events that happened in both virtual and real environments, reflecting changes on each other.

Two approaches were employed for the VRITESS real-life testing: a local network-based solution and a cloud-based deployment with a 3D room developed with Unity. Testing results show that the cloud-based solution has higher latency in comparison to the local-based approach. In the cloud-based scenario, two communications protocols were employed: MQTT and RESTful API. Testing results demonstrated better performance in favour of MQTT, as it has achieved lower delay and requires less amount of data exchanged due to its lightweight design.

Future work includes network testing of multiple gateways with remote VR users manipulating objects located in different gateways, and also integration of additional smart devices, beacons and sensors into the virtual world.

REFERENCES

[1] Cisco, "Internet of Things: Connected Means Informed," 2016. [Online]. Available: https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf

[2] C. Rowland, "What's Different About User Experience Design for the Internet of Things?" 2015. [Online]. Available: https://uxmag.com/articles/whats-different-about-user-experience-design-for-the-internet-of-things

[3] E. Rubio-Drosdov, D. Díaz-Sánchez, F. Almenárez, P. Arias-Cabarcos, and A. Marín, "Seamless Human-Device Interaction in the Internet of Things," IEEE Transactions on Consumer Electronics, vol. 63, no. 4, pp. 490–498, Nov. 2017.

[4] G.-M. Muntean, P. Perry, and L. Murphy, "Objective and subjective evaluation of QOAS video streaming over broadband networks," IEEE Transactions on Network and Service Management, vol. 2, no. 1, pp. 19–28, Nov. 2005.

[5] A. N. Moldovan, A. Molnar, and C. H. Muntean, "EcoLearn: Battery Power Friendly E-Learning Environment for Mobile Device Users," in Learning-Oriented Technologies, Devices And Networks. Lambert Academic Publishing, 2011, pp. 273–296.

[6] G.-M. Muntean, P. Perry, and L. Murphy, "Subjective Assessment of the Quality-Oriented Adaptive Scheme," IEEE Transactions on Broadcasting, vol. 51, no. 3, pp. 276–286, Aug. 2005.

[7] F. Buttussi and L. Chittaro, "Effects of Different Types of Virtual Reality Display on Presence and Learning in a Safety Training Scenario," IEEE Transactions on Visualization and Computer Graphics, vol. 24, no. 2, pp. 1063–1076, 2018.

[8] A. Steed, S. Friston, M. M. Lopez, J. Drummond, Y. Pan, and D. Swapp, "An 'In the Wild' Experiment on Presence and Embodiment using Consumer Virtual Reality Equipment," IEEE Transactions on Visualization and Computer Graphics, vol. 22, no. 4, pp. 1406–1414, 2016.

[9] C. Peng, X. Tan, M. Gao, and Y. Yao, "Virtual Reality in Smart City," in Geo-Informatics in Resource Management and Sustainable Ecosystem. Communications in Computer and Information Science. Springer, Berlin, Heidelberg, 2013, pp. 107–118.

[10] A. Castellani and M. Dissegna, "WebIoT: A Web Application Framework for the Internet of Things," in Proc. of the IEEE Wireless Communications and Networking Conference Workshops (WCNCW), 2012, pp. 202–207.

[11] A. Floris and L. Atzori, "Quality of Experience in the Multimedia Internet of Things: Definition and Practical Use-Cases," in Proc. of the IEEE International Conference on Communications Workshops (ICC Workshops), 2015, pp. 1747–1752.

[12] J. Han, J. Yun, J. Jang, and K. R. Park, "User-Friendly Home Automation Based on 3D Virtual World," IEEE Transactions on Consumer Electronics, vol. 56, no. 3, pp. 1843–1847, Sep. 2010.

[13] Y. Jeong, H. Joo, G. Hong, D. Shin, and S. Lee, "AVIoT: Web-Based Interactive Authoring and Visualization of Indoor Internet of Things," IEEE Trans. Consum. Electron., vol. 61, no. 3, pp. 295–301, Sep. 2015.

[14] V. Kepuska and G. Bohouta, "Next-Generation of Virtual Personal Assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home)," in Proc. of the IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), 2018, pp. 99–103.

[15] V. Lampkin, W. T. Leong, L. Olivera, S. Rawat, N. Subrahmanyam, and R. Xiang, "Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry," 2012. [Online]. Available: https://www.redbooks.ibm.com/redbooks/pdfs/sg248054.pdf

[16] T. Yokotani and Y. Sasaki, "Transfer protocols of tiny data blocks in IoT and their performance evaluation," in IEEE 3rd World Forum on Internet of Things, 2017, pp. 54–57.

[17] S. Sreeraj, N. Suresh Kumar, and G. Santhosh Kumar, "A framework for predicting the performance of IoT protocols, a use case based approach," in Proceedings of the 2017 International Conference On Smart Technology for Smart Nation, SmartTechCon 2017, 2017, pp. 577–580.

[18] M. H. Asghar and N. Mohammadzadeh, "Design and simulation of energy efficiency in node based on MQTT protocol in Internet of Things," in International Conference on Green Computing and Internet of Things, ICGCIoT, 2016, pp. 1413–1417.

[19] N. Singh and S. Singh, "Virtual Reality: A Brief Survey," in Proc. of the International Conference on Information Communication and Embedded Systems (ICICES), Feb. 2017, pp. 1–6.

[20] Z. Zhang, M. Zhang, Y. Chang, E.-S. Aziz, S. K. Esche, and C. Chassapis, "Collaborative Virtual Laboratory Environments with Hardware in the Loop," in Cyber-Physical Laboratories in Engineering and Science Education. Cham: Springer International Publishing, 2018, pp. 363–402.
[21] P. Lelyveld, "Virtual Reality Primer with an Emphasis on Camera-Captured VR," SMPTE Motion Imaging Journal, vol. 124, no. 6, pp. 78–85, Sep. 2015.
[22] M. Narbutt, S. O'Leary, A. Allen, J. Skoglund, and A. Hines, "Streaming VR for immersion: Quality aspects of compressed spatial audio," in Proceedings of the 2017 23rd International Conference on Virtual Systems and Multimedia, VSMM, 2017, pp. 1–6.
[23] H. G. Kim, H. Lim, and Y. M. Ro, "Deep Virtual Reality Image Quality Assessment with Human Perception Guider for Omnidirectional Image," IEEE Trans. Circuits Syst. Video Technol., vol. PP, pp. 1–11, 2019.
[24] F. Fittkau, A. Krause, and W. Hasselbring, "Exploring Software Cities in Virtual Reality," in Proc. of the IEEE 3rd Working Conference on Software Visualization, 2015, pp. 130–134.
[25] T. T. Le, D. V. Nguyen, and E. S. Ryu, "Computing Offloading over mmWave for Mobile VR: Make 360 Video Streaming Alive," IEEE Access, vol. 6, pp. 66 576–66 589, 2018.
[26] C. G. Coogan and B. He, "Brain-Computer Interface Control in a Virtual Reality Environment and Applications for the Internet of Things," IEEE Access, vol. 6, pp. 10 840–10 849, 2018.
[27] D. You, B. S. Seo, E. Jeong, and D. H. Kim, "Internet of Things (IoT) for seamless virtual reality space: Challenges and perspectives," IEEE Access, vol. 6, pp. 40 439–40 449, 2018.
[28] M. Alessi, E. Giangreco, M. Pinnella, S. Pino, D. Storelli, L. Mainetti, V. Mighali, and L. Patrono, "A Web Based Virtual Environment as a Connection Platform between People and IoT," in Proc. of the 2016 International Multidisciplinary Conference on Computer and Energy Science (SpliTech), 2016, pp. 1–6.
[29] Z. Lv, T. Yin, H. Song, and G. Chen, "Virtual Reality Smart City Based on WebVRGIS," IEEE Internet of Things Journal, vol. 3, no. 6, pp. 1015–1024, Dec. 2016.
[30] M. I. Choi, L. W. Park, S. Lee, J. Y. Hwang, and S. Park, "Design and Implementation of Hyper-connected IoT-VR Platform for Customizable and Intuitive Remote Services," in Proc. of the 2017 IEEE International Conference on Consumer Electronics (ICCE), 2017, pp. 1–2.
[31] A. R. Biswas and R. Giaffreda, "IoT and Cloud Convergence: Opportunities and Challenges," in Proc. of the 2014 IEEE World Forum on Internet of Things (WF-IoT), Mar. 2014, pp. 375–376.
[32] Y.-T. Lee, W.-H. Hsiao, C.-M. Huang, and S.-C. Chou, "An Integrated Cloud-Based Smart Home Management System with Community Hierarchy," IEEE Trans. Consum. Electron., vol. 62, no. 1, pp. 1–9, Feb. 2016.
[33] Y. Benazzouz, C. Munilla, O. Gunalp, M. Gallissot, and L. Gurgen, "Sharing User IoT Devices in the Cloud," in Proc.of the 2014 IEEE World Forum on Internet of Things (WF-IoT), 2014, pp. 373–374.
[34] J. Delsing, J. Eliasson, J. Deventer, H. Derhamy, and P. Varga, "Enabling IoT Automation Using Local Clouds," in Proc. of the IEEE World Forum on Internet of Things (WF-IoT), 2016, pp. 502–507.
[35] X. Hou, Y. Lu, and S. Dey, "Wireless VR/AR with Edge/Cloud Computing," in Proc. of the 26th International Conference on Computer Communication and Networks (ICCCN), 2017, pp. 1–8.
[36] Y. Kang, H. Kim, and J. Kang, "Docker Based Computation Off-Loading for Video Game Based Mobile VR Framework," in Proc. of the 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017, pp. 123–125.
[37] M. E. Koltko-Rivera, "The Potential Societal Impact of Virtual Reality," Advances in Virtual Environments Technology: Musings on Design, Evaluation, and Applications, vol. 9, pp. 1–18, 2005.
[38] G. Kobayashi, M. C. Broens, M. E. Q. Gonzalez, and J. A. Quilici-Gonzalez, "The Internet of Things and its Impact on Social Relationships Involving Mutual Trust," in Proc. of the 2015 IEEE International Symposium on Technology and Society (ISTAS), 2015, pp. 1–6.
[39] A. A. Simiscuka and G. M. Muntean, "Age of Information as a QoS Metric in a Relay-Based IoT Mobility Solution," in Proc. of the 14th International Wireless Communications and Mobile Computing Conference (IWCMC), 2018, pp. 868–873.
[40] A. A. Simiscuka, C. H. Muntean, and G.-M. Muntean, "A Networking Scheme for an Internet of Things Integration Platform," in Proc. of the IEEE International Conference on Communications Workshops (ICC Workshops), 2017, pp. 271–276.
[41] A. A. Simiscuka and G.-M. Muntean, "A Relay and Mobility Scheme for QoS Improvement in IoT Communications," in Proc. of the IEEE International Conference on Communications Workshops (ICC Workshops), 2018, pp. 1–6.
[42] A. A. Simiscuka, M. Bezbradica, and G.-M. Muntean, "Performance Analysis of the Quality of Service- aware Networking Scheme for Smart Internet of Things Gateways," in Proc. of the 13th International Wireless Communications and Mobile Computing Conference (IWCMC), 2017, pp. 1370–1374.
[43] A. A. Simiscuka and G.-M. Muntean, "Synchronisation between Real and Virtual-World Devices in a VR-IoT Environment," in Proc. of the IEEE International Symposium on Broadband Multimedia Systems, 2018, pp. 1–6.
[44] P. Corcoran, "A Matter of Timing: Consumer Electronics and Network Time," IEEE Cons. Electronics Magazine, vol. 2, no. 4, pp. 20–25, 2013.
[45] "Adafruit IO," 2018. [Online]. Available: https://io.adafruit.com/
[46] "Oculus Rift," 2018. [Online]. Available: https://www.oculus.com/rift
[47] "Raspberry Pi," 2019. [Online]. Available: https://www.raspberrypi.org/
[48] "Beeks Beacons," 2017. [Online]. Available: http://bluvision.com/wp-content/uploads/2017/09/Specs-BEEKs-Industrial_1.pdf
[49] "Alienware Aurora," 2018. [Online]. Available: https://www.dell.com/en-ie/shop/desktops-and-all-in-ones/alienware-aurora/spd/alienware-aurora-r7-desktop
[50] "GlassFish," 2018. [Online]. Available: https://javaee.github.io/glassfish/
[51] "JavaServer Faces," 2018. [Online]. Available: https://javaee.github.io/javaserverfaces-spec/
[52] "A-Frame," 2018. [Online]. Available: https://aframe.io/

ANDERSON AUGUSTO SIMISCUKA (S'17) is a Ph.D student with the Performance Engineering Laboratory, School of Electronic Engineering, Dublin City University (DCU). He received the B.Sc. degree in Information Systems in 2014 from Mackenzie Presbyterian University, São Paulo, Brazil. He has worked in several telecom and software development projects in companies such as Wittel (2010-2013), DCU/Ericsson (E-Stream Project, 2014), Arkadin (2014) and IBM (2015). His research is mainly focused on Internet of Things communications performance and is funded by the Irish Research Council and DCU.

TEJAS MORESHWAR MARKANDE received his Masters in Engineering degree from the School of Electronic Engineering, Dublin City University. He received the B.E. degree in Electronics and Telecommunication in 2015 from University of Pune, India. He has worked in two prominent startups as embedded hardware and software developer (2015-2017). He was actively involved in research that focused on designing, implementing and testing a working model for IoT objects and virtual reality integration with the use of IoT protocols.

GABRIEL MIRO-MUNTEAN (M'04–SM'17) is an Associate Professor with the School of Electronic Engineering, Dublin City University (DCU), Ireland, and co-Director of the DCU Performance Engineering Laboratory. He has published over 300 papers in top-level international journals and conferences, authored three books and 18 book chapters, and edited seven additional books. His research interests include quality, performance, and energy saving issues related to multimedia and multiple sensorial media delivery, technology-enhanced learning, and other data communications over heterogeneous networks. He is an Associate Editor of the IEEE Transactions on Broadcasting, the Multimedia Communications Area Editor of the IEEE Communications Surveys and Tutorials, and a Reviewer for important international journals, conferences, and funding agencies. He is a Project Coordinator for the EU-funded project NEWTON http://www.newtonproject.eu.