Empirical Results for High-Definition Video and Augmented Reality Content Delivery in Hyper-Connected Cars

Ovidiu-Andrei Schipor and Radu-Daniel Vatavu

MintViz Lab / MANSiD Research Center, University Ştefan cel Mare of Suceava
Email: schipor@eed.usv.ro, radu.vatavu@usm.ro

Software architecture and applications for the connected car can process and share a variety of digital content, among which high-definition video and Augmented Reality content, toward enhanced driving assistance, navigation, and infotainment systems and services. However, several technical challenges need to be overcome to make such systems and services viable and efficient, including dealing effectively with a variety of types of systems, devices, and platforms, either installed inside the vehicle or represented by the personal mobile and wearable devices of the drivers and passengers. In this paper, we outline these technical challenges and propose a software solution in the form of an event-based middleware layer by modeling the smart, connected car as a specific type of a smart environment. We employ an adapted version of Euphoria, a recently introduced software architecture for general-purpose smart environments, to implement asynchronous communications among heterogeneous input/output devices inside the vehicle. We also adapt Euphoria to fit into the four-layer infrastructure model of the connected car. We conduct a technical evaluation of the request-response time performance achieved with the Euphoria middleware for streaming digital content from 1 Mbps (360p@30fps) to 32 Mbps (4K@30fps) on various devices, either integrated in the vehicle, not integrated but used inside the vehicle, and devices outside the vehicle (the control condition). Our results show effective live streaming achieved for 2K content at 30fps with the 600 Mbps network (i.e., the connected car) and for 4K content at 30fps with the 1.7 Gbps network envisioned for hyper-connected vehicles. These results open up opportunities for high-definition video and AR applications in the automotive industry.

Categories and subject descriptors: Human-centered computing; Interactive systems and tools; Software and its engineering.

Keywords: Hyper-connected cars; Smart vehicles; Smart devices; Software architecture; Wearables; Augmented Reality; Euphoria; High-resolution video; Experiment.

1. INTRODUCTION

The automotive industry is moving toward the concept of a software-driven car to meet the increasing consumers' needs and demands for smart services inside connected vehicles (Burkacky et al., 2018; GreenCarCongres, 2018; Pelliccione et al., 2017). Autonomous driving and connectivity with road infrastructure, other vehicles, and pedestrians as well as data sharing between the connected vehicle and the drivers' and passengers' smart devices (Bilius and Vatavu, 2020) represent just a few examples of applications that necessitate complex in-vehicle software architecture (Petit and Shladover, 2015; Burkacky et al., 2018; Charette, 2009; Zheng et al., 2016). The growing need for smart services, high-definition content delivery, and fast communications for the connected car has led to the vision of a "hyper-connected vehicle" powered by 1.7 Gbps networks (Gai and Violante, 2016), which subsumes more data being

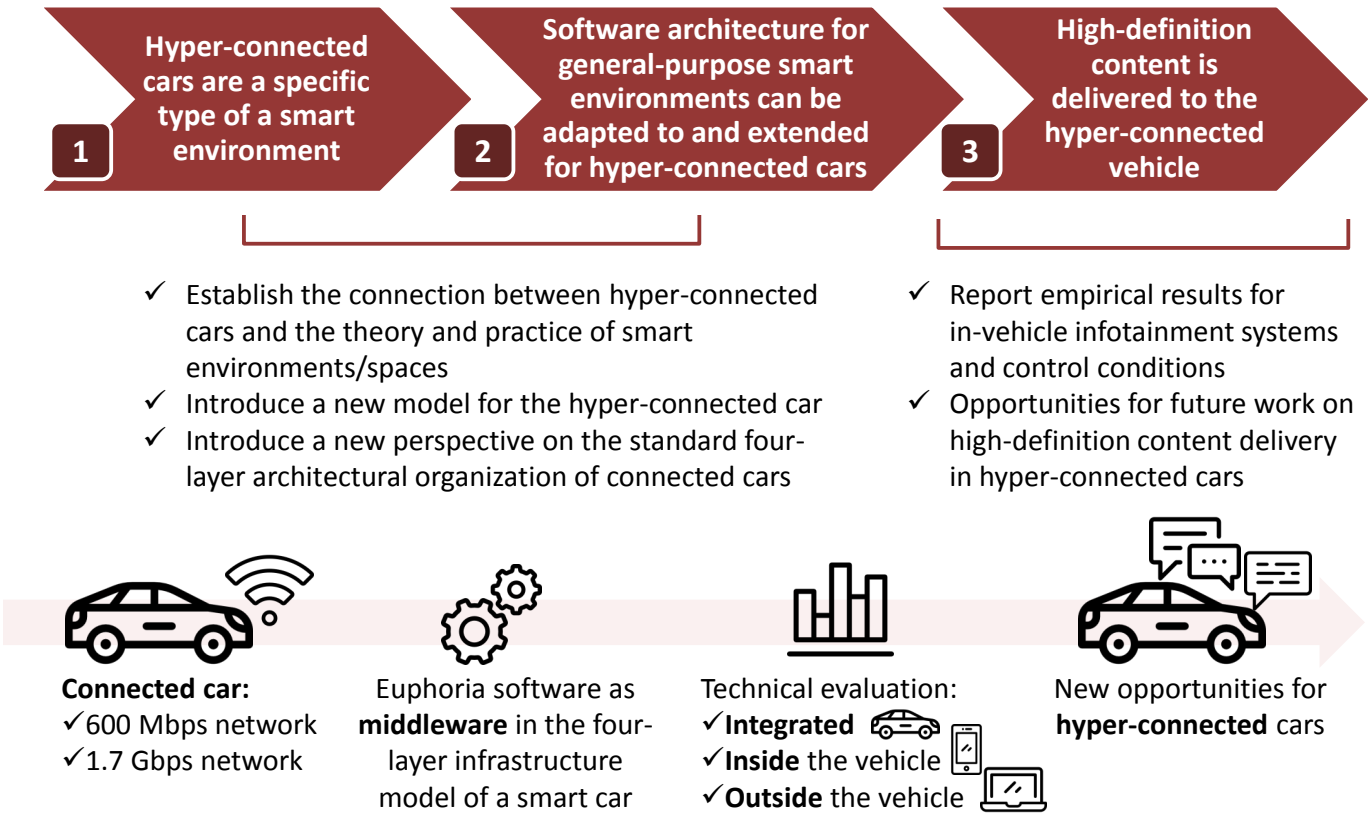

Figure 1. The three-step method adopted in this paper (new model for the smart car as a specific type of a smart environment, adaptation of software architecture for general-purpose smart environments to hyper-connected cars, and technical evaluation) and the corresponding theoretical and practical contributions that this paper makes at each step.

Figure 2. We model hyper-connected cars as a specific type of a smart environment co-inhabited by in-vehicle systems, drivers and passengers, and mobile and wearable devices.

processed, more content rendered and, consequently, more complex in-vehicle software architecture that needs to run efficiently. This vision includes high-definition video and Augmented Reality (AR) content delivered inside the vehicle.

Several application scenarios have been proposed in the scientific community for in-vehicle AR to enhance the driving and travel experience (Rameau et al., 2016; Moniri et al., 2012; Alhaija et al., 2018; Qiu et al., 2017; Yuan et al., 2018; Kim and Dey, 2016). One of the most frequent scenario has been to extend the driver's field of view with insights about what is happening around the vehicle (Alhaija et al., 2018; Olaverri-Monreal et al., 2010; Qiu et al., 2017; Rameau et al., 2016; Yuan et al., 2018), which enables the driver to become more aware of other entities, *e.g.*, vehicles, pedestrians, road infrastructure, etc. and, thus, to make better and more informed decisions that can increase driving safety. Other scenarios have addressed passengers by delivering more information about specific points of interest along the road (Schinke et al., 2010; Lim et al., 2015) and by enabling advanced interactions inside the vehicle and with the vehicle (Moniri et al., 2012; Kim and Dey, 2016; Bilius and Vatavu, 2020; Gheran and Vatavu, 2020). To enable such features, smart vehicles must be able to efficiently manage complex hardware and software modules and

components, starting with the modules in charge of data collection and ending with the specific processing pipelines and systems that present content to users (Alhaija et al., 2018; Rameau et al., 2016; Moniri et al., 2012).

In this context, there are several pressing technical challenges (GreenCarCongres, 2018; Pelliccione et al., 2017; Schipor and Vatavu, 2019). For example, when content is generated on-the-fly, processing pipelines need to be flexible enough to adapt to a wide range of situations, *e.g.*, the movement of pedestrians and of other vehicles or to changes in the quality and reliability of input data when, for instance, the car video cameras become obstructed by weather conditions such as rain or snow. Moreover, the in-vehicle infrastructure integrates heterogeneous hardware and software components that employ different technology, platforms, and operating systems; see Figure 2 for an illustrative example showing the in-vehicle infotainment system and the driver's smartwatch. In this context, an imperative design requirement is represented by loose coupling among the various modules, devices, and systems from the in-vehicle environment, a paradigm that has been considered only recently in the automotive industry (GreenCarCongres, 2018; Burkacky et al., 2018; Zheng et al., 2016; Gai and Violante, 2016). How high-definition video and AR content fits into this paradigm and how practical it is to deliver such content effectively inside the vehicle, given the currently available in-vehicle processing and communications technology, remains to be understood. In this paper, we present both theoretical and practical advances in this direction (see Figure 1 for a visual illustration of our method and contributions), as follows:

(i) We present an inventory of current design and engineering challenges regarding the delivery of high-definition video and AR content inside the vehicle, and we contribute a new formalization of the in-vehicle environment where the hyper-connected car is modeled as a specific kind of a smart environment.

(ii) Within the new framework of the connected car as a smart environment, we employ Euphoria (Schipor et al., 2019), a recently introduced software architecture for general-purpose smart environments, specifically designed to deal effectively with applications in which heterogeneous input/output devices share data. We propose an extension and adaptation of Euphoria for the standard four-layer infrastructure of connected cars – the Application, Service, Devices, and Electronics layers (Burkacky et al., 2018; Marisetty et al., 2009; Milosevic et al., 2018; Traub et al., 2017; Zheng et al., 2016) – for which Euphoria plays the role of middleware and handles subscriptions, notifications, and message processing between software clients.

(iii) We evaluate the technical performance of our software architecture for high-definition video and AR content delivered inside the vehicle. We report empirical results from a controlled experiment involving content size from 1 Mbps (*e.g.*, 360p video at 30 fps) to 32 Mbps (2160p Ultra-High Definition content at 30 fps) delivered to the infotainment system of a connected vehicle (Nissan Qashqai), which we compare against two other conditions represented by a conventional smartphone used inside the vehicle and a high-end laptop used outside the vehicle.

2. RELATED WORK

We discuss prior work on the design and engineering of systems and applications for connected cars, including Ethernet networks of sensors and devices, and we overview in-vehicle AR systems and applications.

2.1. The Connected and Hyper-Connected Car

The concept of a connected car describes a vehicle engaged in asynchronous communications with other entities to enhance the driving and travel experience of the passengers (Karnouskos and Kerschbaum, 2018). One possible model to look at and analyze the connected car is to include vehicles into the Internet-of-Things (IoT) (Gerla et al., 2014), which requires cars to have dedicated hardware and software architectures (GreenCarCongres, 2018; Gai and Violante, 2016; Traub et al., 2017; Burkacky et al., 2018) that implement IoT standards and communications. Another model presents the connected car as the central entity of its informational ecosystem, able to receive data from the environment as well as to produce data for other entities (Karnouskos and Kerschbaum, 2018; Lu et al., 2014). This perspective is specific to the Vehicle-to-Everything (V2X) paradigm, which includes Vehicle-to-Infrastructure (V2I), Vehicle-to-Vehicle (V2V), Vehicle-to-Network (V2N), Vehicle-to-Pedestrian (V2P), and Vehicle-to-Device (V2D) communications. By implementing 1.7 Gbps network communications, connected cars transition to hyper-connectivity.

The concept of a connected car is key for autonomous driving (Lugano, 2017). The US Transportation Security Administration has defined six levels of autonomous cars, from nonautonomous to fully autonomous vehicles (Liu et al., 2017). While most cars implement Level 2 (*i.e.*, the driver can disengage from certain functions, such as cruise control and parking), the automotive industry is now exploring Level 3 (full automation, but the driver still needs to be present) and Level 4 autonomy (no driver needed behind the wheel on controlled routes, such as highways); see Colquitt et al. (2017). The

transition from one level of autonomy to the next does not represent solely a quantitative shift (*i.e.*, a growth in sensing and computing resources), but also a qualitative one (Markwalter, 2017; Liu et al., 2017) that requires novel architectural approaches based on loosely coupled modules and flexible processing chains (Liu et al., 2017).

2.2. Ethernet Networks of Heterogeneous Devices for Hyper-Connected Cars

CES 2018 featured a Cisco and Hyundai demonstration of a flexible, scalable, and secure platform to support configurable interactions between the software and hardware modules inside the vehicle (GreenCarCongres, 2018; Corbett et al., 2016; Traub et al., 2017; Zheng et al., 2016). Special attention was dedicated to the integration of heterogeneous components, such as legacy hardware and software libraries from different vendors, into that platform. Also, instead of having multiple one-to-one communications between the various components of the platform, the new approach proposed a decoupled architecture where modules acted as hosts within the 1.7 Gbps Ethernet network. The resulting Software-Defined Vehicle architecture (GreenCarCongres, 2018) has played a key role for the next generation of hyper-connected cars.

In this context, advances in security, quality of service, and scalability can be achieved by incorporating knowledge and results from the field of computer networks (Corbett et al., 2016); *e.g.*, prior work has showed that Ethernet networks can successfully address most of the requirements of the automotive industry in terms of communications (Corbett et al., 2016). Moreover, a high-end network infrastructure enables connections with external systems, such as cloud data centers, smart city infrastructure, other vehicles, and pedestrians (Traub et al., 2017; Zheng et al., 2016). A report on future mobility of Burkacky et al. (2018) envisioned Ethernet technology as "the backbone of the car" to replace current approaches based on cluster data buses (Sharma et al., 2009) and to facilitate reuse of modules across different manufacturers (Staron, 2017). The report also highlighted Ethernet networks as a viable solution to handle the increasing number of on-board smart sensors. The transition to Ethernet networks is equally facilitated by the standardization of testing processes, *e.g.*, the automotive Ethernet TCP/IP stack. Also, the AUTomotive Open System ARchitecture (AUTOSAR) provides high level specifications for testing protocols (AUTOSAR, 2016), while the OPEN Alliance (OPEN Alliance, 2016) has released the Ethernet testing specifications for Electronic Control Units (ECUs).

2.3. Augmented Reality Systems for Hyper-Connected Cars

AR is one type of content that will become more and more relevant for the automotive industry. Recently, there has been a growing interest in designing AR applications for in-vehicle contexts of use (Jose et al., 2016; Cao et al., 2018; Colley et al., 2018, 2017), but the photorealism and responsiveness expected from modern AR applications demand high-performing devices, large network bandwidth, and high computational resources. For instance, the Augmented Reality for Enterprise Alliance (AREA, 2018) specifies a minimum on-board memory storage of 128 Gb for industry use case devices; the 2018 Ovum/Intel Report on 5G for media and entertainment forecast that 5G will unlock AR and VR applications (Ovum, 2018); while the global automotive AR/VR market is expected to reach 673 billion US dollars by 2025 (Liu, 2019). In this context, one key aspect to driver future development in this direction is to rethink the way in which the in-vehicle components and modules interact with each other to support drivers' and passengers' user experience with in-vehicle AR (Davies, 2018; Rao et al., 2014). These applications require the ability to handle dynamically generated content, manage a wide variety of technologies, and assure proper decoupling of hardware and software modules. Moreover, understanding the performance of high-definition video and AR content delivery in hyper-connected cars would constitute into key knowledge for practitioners, but experimental results in this direction are limited.

One way to deliver AR inside the vehicle is via virtual assistants (Lugano, 2017). In this regard, the automotive industry has adapted existing virtual assistants (*e.g.*, Daimler with Google Assistant, BMW and Nissan with Cortana, Ford with Alexa, General Motors with OnStar Go) and also created new ones (*e.g.*, Toyota Yui, Honda HANA, Volkswagen Sedric) (Barniuk, 2017). The goal is to combine safety with infotainment and IoT within a flexible software architecture (Lugano, 2017) that goes beyond current in-vehicle applications that can be characterized as borderline AR, *e.g.*, conventional parking assistance systems (Wang et al., 2014; Liu et al., 2017). Regarding the latter, hyper-connected cars can integrate data captured by external sensors and entities, such as the video cameras from the parking lot or nearby vehicles in order to achieve better collision detection performance (Khanna and Anand, 2016; Hammoudi et al., 2018; Mahendra et al., 2017). Such advanced features, however, put demands on the network bandwidth and processing power of the in-vehicle system, and the software architecture of such vehicles needs to be able to process content received from heterogeneous systems and generate AR content on the fly.

2.4. An Overview of Middleware for Smart, Connected Vehicles

The increase in the complexity of electronics and software from the connected vehicle has led to many attempts to design appropriate middleware. For example, InCloud (Saini et al., 2017) is a middleware framework for infotainment application development in the Vehicular Adhoc Network. InCloud presents several advantages: content is easy to integrate, software clients are lightweight since most of the processing is performed remotely, the coupling between the interface and the processing modules is loose, and several sources of content can be combined. The framework revolves around four design requirements: fusion of data from multiple sources, context-awareness, reusability, and loose coupling. Cisco and Hyundai proposed the Software Defined Vehicle architecture (GreenCarCongres, 2018) as a secure, multi-layer platform for integrating both software and hardware components; see subsection 2.2. Sadio et al. (2019) reported evaluation results for this platform, such as packet loss, throughput, and delay in network communications.

The in-vehicle environment hosts heterogeneous technologies (*i.e.*, hardware and software components from different manufacturers) that must be integrated reliably. Aastha (2019) designed and implemented a service-oriented middleware to integrate various types of ECUs with the AUTOSTAR specification. The implementation employed an Ethernet network for in-vehicle communications since other approaches (*e.g.*, CAN, LIN, or FlexRay) were not sufficient to manage the complexity of the driving assistance systems (Aastha, 2019; Gopu et al., 2016; Häckel, 2018). The middleware of Aastha (2019) relied on three types of components: providers, consumers, and registry, and enabled streams, messages, and remote procedure calls. Häckel (2018) extended the architecture with protocol abstraction through standardized interfaces, Internet-based technology, and lightweight capability to achieve compatibility with low-end and legacy ECUs. The middleware was evaluated using the OMNeT++ simulation software (Häckel, 2018).

Several approaches have been proposed to implement loose coupling between the components of the connected vehicle. According to the digital twin platform paradigm, the hardware components of a cyber-physical system have associated software counterparts (Yun et al., 2017); see Josifovska et al. (2019) that integrated the components of a digital twin system and highlighted their structural properties and interrelations. Interoperability and low coupling between heterogeneous components can also be achieved with component-based modeling (Hellwig et al., 2019), platform-agnostic logical models, and middleware. The approach of Hellwig et al. (2019) was to propose a tag-based multi-platform code generation for such components starting from their declaration.

3. CHALLENGES FOR DELIVERING HIGH-DEFINITION CONTENT TO THE HYPER-CONNECTED VEHICLE

In the following, we identify the main technical challenges regarding middleware for delivering content, including high-definition video and AR, in connected and hyper-connected vehicles (Schipor and Vatavu, 2019). To illustrate these challenges, we resort to a practical scenario mentioned repeatedly in prior work (Alhaija et al., 2018; Olaverri-Monreal et al., 2010; Qiu et al., 2017; Rameau et al., 2016; Yuan et al., 2018). In this scenario, the in-vehicle hardware and software modules interoperate to extend the driver's visual field to cover other entities around the vehicle, such as pedestrians, traffic signs, and other vehicles. The virtual 3D model of the space surrounding the car is generated based on a variety of input delivered by video cameras, proximity sensors, other vehicles, and road infrastructure. Drivers are presented with a virtual model by means of video projections, smartglasses, head-mounted displays, or holograms. Such systems aim to reduce the risks of road accidents by assisting drivers during complex maneuvers, such as overtaking other cars or parking. The in-vehicle system can also deliver passengers with enriched entertainment experiences (Lim et al., 2015; Bilius and Vatavu, 2020). We will use this scenario throughout this section to illustrate challenges for AR systems in connected cars.

3.1. Challenge #1: Production of Content On the Fly

AR content is dynamically generated and synchronized with the physical world (Alhaija et al., 2018; Qiu et al., 2017). This feature is achieved by real-time data acquisition and processing and via a specialized chain of hardware and software modules. To respond effectively to changes from the environment, this processing chain needs to add or remove components and reconfigure the processing pipeline on the fly. The corresponding technical challenge is to ensure flexible association between the components of the AR models and the modules of the in-vehicle system. While in a classical architecture the relationships between components are defined during design, a flexible processing pipeline requires a higher level of abstraction (Sharma et al., 2009; Staron, 2017). In this case, components need to deliver results in a standardized way without any knowledge of the potential consumers (Burkacky et al., 2018; Milosevic et al., 2018).

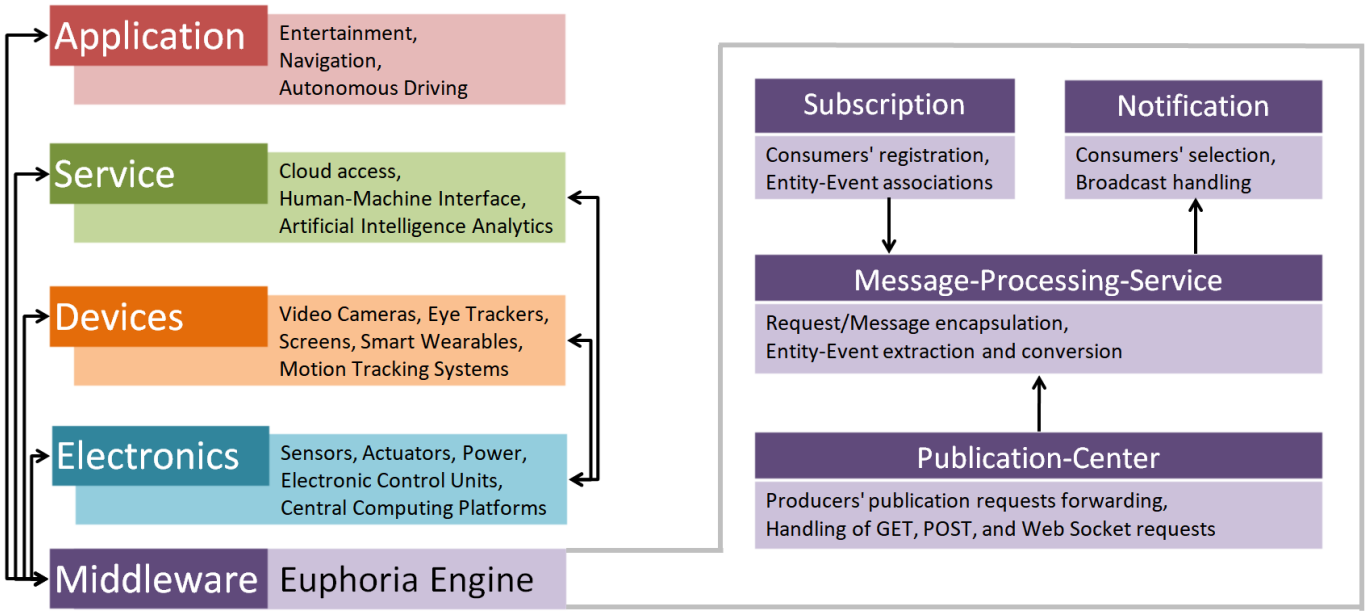

Figure 3. Integration of the Euphoria software architecture (Schipor et al., 2019), designed for general-purpose smart environments, into the standard four-layer infrastructure of the connected car at the middleware level.

3.2. Challenge #2: Heterogeneous Devices

Hyper-connected cars rely on a wide variety of hardware and software technologies (Staron, 2017; Pelliccione et al., 2017; Blom et al., 2016; Diaconescu et al., 2018), an aspect that is important to consider when designing and engineering in-vehicle AR (Burkacky et al., 2018; Staron, 2017). The data collected for the generation of the virtual models require inter-vehicle communications, high-speed networks, and effective and efficient transmission protocols. Furthermore, the generation of the virtual models requires robust computer vision algorithms for object detection and recognition under complex conditions, *e.g.*, bad weather, low light, and so on. Additionally, hyper-connected cars must assist drivers in specific situations, such as overtaking or parking, by presenting estimations of distances and emergency breaking, which are features that require real-time processing capabilities. An effective software architecture for the connected car should enable unobtrusive operation of such modules, interweaving engine operation, in-vehicle infotainment services, connections to smart devices and services from the cloud, and driving safety (Staron, 2017; Gai and Violante, 2016). In this context, a recent trend has gained traction in the automotive industry regarding new architectures that are highly inter-operable and that can host virtually any hardware and software technology (Sharma et al., 2009; Staron, 2017; Burkacky et al., 2018; Milosevic et al., 2018).

3.3. Challenge #3: Effective Decoupling of Hardware and Software Modules

Connected cars rest on the principle of high abstraction of their hardware and software components so that heterogeneous systems can be easily integrated. In fact, the automotive industry has gradually increased the level of abstraction from microcontrollers to systems-on-chip and to scalable computing platforms (Gai and Violante, 2016; Traub et al., 2017). By moving from a hardware-centered to a function-centered perspective and approach, software applications can run on a central computer (Gai and Violante, 2016; Traub et al., 2017; Burkacky et al., 2018). For example, AUTOSTAR (Gai and Violante, 2016) offers a standardized runtime environment for the Services and Applications layers; see Figure 3, left. Moreover, AUTOSTAR can operate in conjunction with embedded virtualization technologies and leverage their capabilities to hide the complexity of the hardware. To control the data flow, such an approach requires a central information server and a broker for the data transferred through the network (Traub et al., 2017). The project "Hercules" (Marko, 2018) is another example of a high-performing architecture for low-power embedded systems. One of its goals was the implementation of a homogeneous application programming interface (API) on top of heterogeneous commercial off-the-shelf platforms.

4. THE EUPHORIA SOFTWARE ARCHITECTURE FOR HYPER-CONNECTED CARS

A 2018 Report by McKinsey & Co. (Burkacky et al., 2018) presented a design for in-vehicle layered architecture connecting low-level components (*e.g.*, sensors, actuators, and power electronics) with high-level modules (infotainment, connectivity, and cloud access) via a middleware layer; see the left part of Figure 3. From this perspective, a significant part of the business logic of in-vehicle applications can be transferred to distinct processing layers, such as layers that implement Artificial Intelligence tasks or advanced analytics. The Applications layer can also be expanded to support multiple guest operating systems within the infotainment component and micro-kernel architectures (David and Matt, 2012). This blue-print was designed to deliver effective data redundancy mechanisms for safety-critical applications.

In this section, we connect to the four-layer model of the connected car (Burkacky et al., 2018) in which we integrate Euphoria (Schipor et al., 2019), a software architecture designed for general-purpose smart environments, to act as middleware. But, before, we introduce the concept of a hyper-connected car as a specific kind of a smart environment.

4.1. The Hyper-Connected Car as a Specific Kind of a Smart Environment

We model the hyper-connected car as a specific kind of a smart environment, where the in-vehicle modules, end users (driver and passengers), and other systems and devices (*e.g.*, smartphones, tablets, smartwatches, etc.) co-inhabit the in-vehicle environment and create, store, process, and share content. We adopt this perspective and model in order to introduce a software architecture for hyper-connected cars that (i) connects to the four layers designed by previous work for connected cars, *i.e.*, Applications, Services, Devices, and Electronics (Burkacky et al., 2018; Marisetty et al., 2009; Milosevic et al., 2018; Traub et al., 2017; Zheng et al., 2016) and (ii) is inspired from and builds on top of recent, high-performing software architecture designs for general-purpose smart environments (Schipor et al., 2019). Next, we present the characteristics of Euphoria (Schipor et al., 2019) that recommend its use for our purpose.

4.2. The Euphoria Software Architecture for General-Purpose Smart Environments and Heterogeneous Input and Output Devices

In the following, we conduct our discussion around the core engine of Euphoria (Schipor et al., 2019), a general event-driven architecture for engineering interactions in smart environments between heterogeneous entities; see Schipor et al. (2019) for technical details, evaluation of technical performance, and examples of applications in (Schipor et al., 2017; Schipor and Vatavu, 2018; Schipor et al., 2019b; Popovici et al., 2019; Schipor et al., 2019c).

Euphoria rests on ten design criteria: two handling techniques to describe the processing of events (*i.e.*, event-driven and asynchronous processing), four quality features that characterize how entities interact with each other (*i.e.*, adaptability, modularity, flexibility, and interoperability), and four contextual properties regarding specific technology (*i.e.*, web-based, open-source, smart space, and JavaScript); see Schipor et al. (2019). Also, Euphoria consists of five layers: Producers and Consumers (*i.e.*, actual devices and software components), Emitters and Receivers (standardized abstractions of actual producers and consumers), and the core Engine responsible for dispatching events and messages from producers to consumers. The asynchronous and event-driven design of Euphoria assures high decoupling between its components (Schipor et al., 2019). Moreover, the abstraction of standardized interfaces enables heterogeneous hardware and software modules to interoperate, which falls in line with recent trends in designing automotive architectures (Traub et al., 2017; Zheng et al., 2016). The informational flow is implemented by JSON messages delivered through WebSockets, and the engine is written entirely in JavaScript (Schipor et al., 2019). Euphoria has been used for engineering applications for smart environments (Schipor et al., 2017; Schipor and Vatavu, 2018; Popovici et al., 2019), and extended to meet other requirements; see Schipor et al. (2019b,c) for discussions about integrating peripheral interaction with smart environments and AR powered by the Euphoria architecture.

We chose Euphoria over other software architecture designs and platforms for smart environments (Fortino et al., 2012; Zahariadis et al., 2014; Marquardt et al., 2011; Fortino et al., 2014; Nebeling et al., 2014; Ledo et al., 2015; Roda et al., 2016; Mocanu and Schipor, 2017; Lou et al., 2017) because it implements more quality properties. For example, compared to the UbiComp middleware (Goumopoulos et al., 2009) and SPINE (Bellifemine et al., 2011), Euphoria was specifically designed for the web; compared to GS-GPE (Lou et al., 2017) and GPWS (Vatavu et al., 2012), Euphoria allows various input modalities for end users; and compared to MAPS (Aiello et al., 2010) that needs adaptations to support interactions, Euphoria implements them natively. We refer the interested reader to Table 1 from Schipor et al. (2019) and the associated discussion for more comparisons between Euphoria and other software

architectures and platforms proposed in the scientific literature in the context of smart environments.

4.3. Euphoria for the Hyper-Connected Car

Euphoria rests on common Internet standards and protocols implemented by virtually all programming languages and operating systems. This feature is useful in our application scenario, because handling high-definition content as well as specific content (video, AR) requires the integration of various types of devices, platforms, and applications. Due to the standardized header of the messages transmitted through Euphoria, the processing flow can be rerouted at software level to respond dynamically to various types of events.

We employ Euphoria to mediate non-critical interactions between in-vehicle systems since it is Ethernet-based, works with the abstraction of hardware and software modules, and employs the JSON format for messages. Figure 3 illustrates the integration of Euphoria with the main four layers of an in-vehicle architecture. The Applications level consists of all the software modules that provide interfaces for passengers (*e.g.*, part of the in-vehicle infotainment system and the advanced driving assistance system) (Okuda et al., 2014; Engen et al., 2009); entertainment, navigation, and autonomous driving applications rely on the Services layer; and the in-vehicle hardware is addressed by the Devices and Electronics layers. While Devices relate to high-level components with which users can interact directly, Electronics comprise the processing and power circuits of the vehicle. Each layer has a dedicated adapter to interface with the Euphoria middleware. However, critical, real-time, and safety-related functions benefit of direct informational shortcuts, as illustrated in Figure 3.

In order for external components to become Consumers,[1] they need to register with Euphoria by using the API provided by the Subscription module; see the right part of Figure 3. The registration process involves creation of a web socket and specifying which types of messages will be forwarded to the consumer. The consumer sends to the engine a list of entity-event associations and the Notification module will feed the consumer with messages that match those associations. Each message is encoded in the JSON format with a header section containing the entity and type of event; see Figure 4 for examples. The Publication-Center implements the connection between the engine and Producers.[2] Producers deliver messages both by HTTP requests (GET and POST) and

```
message{2} ▼
  header{3} ▶ entityName      :IVIS
              deviceIP        :192.168.0.10
              eventName       :SpeedLimit
              timestamp       :1568279952231
  body  {1} ▶ value           :50

message{2} ▼
  header{3} ▶ deviceName      :MyoArmband_1
              deviceIP        :192.168.0.11
              eventName       :Gesture
              timestamp       :1568279952231
  body  {3} ▶ gesture         :FingersSpred
              acceleration{3}▶ x       :-0.369
                               y       : 0.824
                               z       :-0.153
              orientation {3}▶ azimuth : 0.209
                               pitch   :-0.492
                               roll    :-0.626
```

Figure 4. Two examples of JSON messages for data exchange. The header sections contain metadata about the entity and the event, *i.e.*, the detection of a speed limit traffic sign (top) and the detection of a gesture (bottom). The body section consists of fields required to specify the context of the event, *i.e.*, the maximum speed and the gesture command, respectively.

via web sockets; see Figure 5. To avoid the decapsulation of large messages, Euphoria allows entities and event types to be specified directly in the request string; see Figure 6 for an example. The Message-Processing-Service is in charge with identifying the entity and event type of a message either from the request string or from the header section. This information is converted in an internal format and made available to the Notification module.

Modules can act both as producers and consumers, and the HTTP web client illustrated in Figure 5 exemplifies this aspect. The same functionality can be obtained using various programming languages and platforms. Once the request for creating a new client has been sent to the server, the subscription process is triggered; see Figure 6. Each consumer consists of a web socket and a list of entity-event associations. Messages received by Euphoria are dispatched according to their types.

4.4. Addressing the Challenges for Delivering High-Definition Content in Connected Cars using the Euphoria Middleware

Euphoria can successfully address the challenges of handling content in hyper-connected cars (see our discussion from Section 3), as follows:

[1] Terminology used by the Euphoria software architecture to denote applications that are the final recipients of content distributed through the architecture (created by producers); see Schipor *et al.* (Schipor et al., 2019) for details and examples.

[2] See the previous footnote.

```
function createConsumer(consumerQuery) {
return new WebSocket(CONSUMER_BASE_URL +
    consumerQuery);
}

function sendGetMessage(query) {
let xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET", PRODUCER_BASE_URL +
    query, true);
xmlhttp.send();
}

function sendPostMessage(query, body) {
//... any processing goes here ...
xmlhttp.send(body);
}

function sendWsMessage(query, body) {
let producer = new WebSocket(
    PRODUCER_BASE_URL_WS + query);
producer.onopen = () => {
producer.send(body);
}
}
```

Figure 5. JavaScript code exemplifying a web client application that creates Euphoria consumers and sends messages using HTTP requests.

```
// import Euphoria API
subscription=require('./Subscription');
notification=require('./Notification');
processing=require('./Processing');

function registerConsumer (ws, req) {
// get entity-event pairs
let categories = JSON.parse(req.query.
    categories);
// register the consumer
let consumer = subscription.createNewCons(
    ws, categories);
});

function dispatchMessage(message,head) {
let messageCategory = processing.
    getCategories(JSON.parse(head));
notification.notifyConsumers (message,
    messageCategory);
}
```

Figure 6. JavaScript code exemplifying a server application that registers Euphoria consumers and dispatches messages to registered consumers.

(i) *Euphoria can handle dynamically generated content.* In Euphoria, all the entities, *i.e.*, devices and software modules, are logically connected by the information from the headers of standardized messages. When a producer creates a message, it does not need to know which consumers will receive that message. Moreover, a consumer can dynamically select which messages to handle by sending the list of producers and event types to Euphoria. Therefore, associations between the various components of the AR model and the modules of the system can be dynamically established using the header of exchanged messages. This aspect is important when content is not static, but dynamically generated and modified by the various components of the application.

(ii) *Euphoria can manage a wide variety of in-vehicle technologies.* Euphoria was designed to use common Internet standards and protocols for data processing and exchange, *e.g.*, HTTP, WebSockets, and JSON, implemented in virtually all operating systems and programming languages. The core components of Euphoria are written in JavaScript and, thus, can run on any device equipped with a standard web browser. Embedded systems, devices, and software modules can be integrated within Euphoria by exposing their public interfaces via Internet protocols. Components can become consumers by instantiating web sockets with the help of the API from the Subscription module. Messages are delivered to Euphoria via HTTP requests and web sockets.

(iii) *Euphoria assures high decoupling for the hardware and software modules.* This outcome follows from a design requirement of Euphoria to integrate heterogeneous hardware and software components (Schipor et al., 2019). The registration and notification mechanisms implemented in Euphoria treat both the hardware and software modules unitarily so that the actual implementation of a component is transparent for developers and end users. Due to the standardized interfaces (see Figures 5 and 6), changes operated within a specific module remain isolated and do not propagate to the other levels.

5. EXPERIMENT

We conducted a controlled experiment to evaluate the technical performance of the Euphoria middleware for applications that stream content up to 4K ultra-high definition video. For this purpose, we implemented two JavaScript applications, which we ran on various devices and measured the request-response times for content delivered through Euphoria. One application acted as a PRODUCER and generated data of various sizes (from 1 Mbps to 32 Mbps) and streamed that data via Euphoria to the second application that acted as the CLIENT.

5.1. Design

Our experiment was a repeated-measure, within-subjects design with the following three independent variables:

(i) Content-Size: the size of the video or AR content, after compression, transmitted through Euphoria and ranging from 1 Mb to 32 Mb with the content size following a geometric progression with ratio 2, *i.e.*, 1, 2, 4, 8, 16, and 32 Mb. We chose these conditions to correspond to one second of video streamed at resolutions and frame rates of 360p@30fps (≈1 Mbps), 480p@30fps (2 Mbps), 720p@30fps (HD, 4 Mbps), 1080p@30fps (Full HD, 8 Mbps), 1440p@30fps (2K, 16 Mbps), and 2160p@30fps (4K, 32 Mbps), according to the recommended upload encoding settings, frame rates, and bit rates for YouTube videos.[3]

(ii) Device, ordinal variable with three conditions: the infotainment system of a Nissan Qashqai vehicle (Android) and two other devices (high-end laptop and conventional smartphone running Windows and Android, respectively), described in detail in the Apparatus section. These conditions correspond to: (1) device integrated in the vehicle, (2) device not integrated, but used inside the vehicle, and (3) high-end device outside the vehicle (the control condition).

(iii) Network, ordinal variable with two conditions: 600 Mbps@2.4GHz and 1.7 Gbps@5GHz, representing the signal rate characteristics of two wireless networks connecting each device with Euphoria.

Our experiment design had a total number of $6 \times 3 \times 2 = 36$ conditions to measure our dependent variable, Request-Response-Time (in milliseconds), when streaming digital content via the Euphoria middleware.

5.2. Task

We performed 100 repeated measurements (referred to as trials) for each combination of the experimental conditions. We generating random data of size ranging from 1 Mb to 32 Mb by selecting pixels at random from a high-resolution photograph. The order of the trials and the actual content that was transferred were randomized to prevent biases in time measurements due to caching effects in the wireless network out of our control.

5.3. Apparatus

Euphoria ran on a Windows 10 platform powered by a Dell Inspiron 15 laptop featuring an Intel Core i7-7500U 2.7 GHz CPU with 16 GB RAM. The in-vehicle system was a custom EDOTEC 499 infotainment device, integrated in a Nissan Qashqai (production version 5D 1.2L M/T 2WD/2015), featuring an Octa-Core 1.8 GHz CPU with x86-64 architecture, Spreadtrum SC9853 chipset, 2 GB RAM, and running Android 8; see Figure 7, left. Despite being a high-end device for in-vehicle infotainment, the EDOTEC 499 system supported only the 600 Mbps 2.4 GHz wireless network connection. Therefore, we also tested Euphoria on a Huawei P10 Lite smartphone with an Octa-Core 2.1 GHz CPU, 3 GB RAM, and Android 8 (a conventional smartphone) that supported the 1.7 Gbps 5 GHz wireless connection. We included the smartphone as a distinct experimental condition to understand the opportunity of using the 1.7 Gbps network inside the vehicle and, therefore, to draw implications for future designs of and requirements for in-vehicle infotainment systems. The smartphone condition also represents the situation where the driver or one of the passengers prefers to stream content directly on their personal mobile device, while inside the vehicle; see Figure 7, middle. We also employed a high-end laptop with an Intel Core i7-4710HQ 2.5 GHz CPU, 12 GB RAM memory, and Windows 10 as a control condition (Figure 7, right) to understand how the request-response times obtained for the infotainment system and the smartphone used inside the vehicle compare to those obtained on a high-end computing device used outside the vehicle. The 600 Mbps and the 1.7 Gbps Wi-Fi networks were implemented using the same equipment (ASUS RT-AC87U wireless router) that supported both modes of operation.

5.4. Results

Figure 8 illustrates the effects of Content-Size, Device, and Network independent variables on the Request-Response-Time as the average performance, for each combination of the independent variables, of the 100 trials.

The smallest response time (23 ms) was obtained when streaming one second of content in the condition 320p@30 fps between Euphoria and the high-end laptop (our control condition) over the 1.7 Gbps Wi-Fi network. The request-response times increased for the control condition up to 369 ms for content of size 2160@30fps and the 1.7 Gbps network and to 744 ms when the 600 Mbps network was used. This result shows that, in the control condition involving a high-end device, Euphoria can effectively deliver high-definition content.

Having established the baseline, we look at the performance of the infotainment system integrated in the vehicle, for which only the 600 Mbps wireless network connection was supported. Request-response times started at 114 ms for the 320p@30 fps condition for content size

[3] YouTube. Recommended upload encoding settings. https://support.google.com/youtube/answer/1722171?hl=en

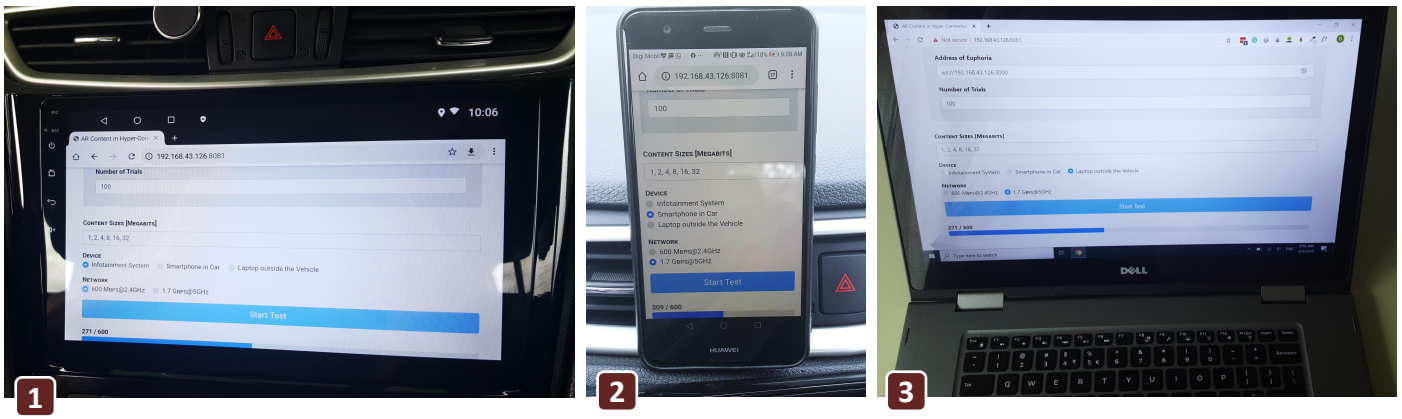

Figure 7. Apparatus employed in our experiment to measure request-response times: (1) infotainment system integrated in a a Nissan Qashqai, (2) smartphone used inside the vehicle, and (3) laptop outside the vehicle (control condition). *Note:* the same client application (HTML and JavaScript) was used on each device and is displayed in all the photos (1) to (3).

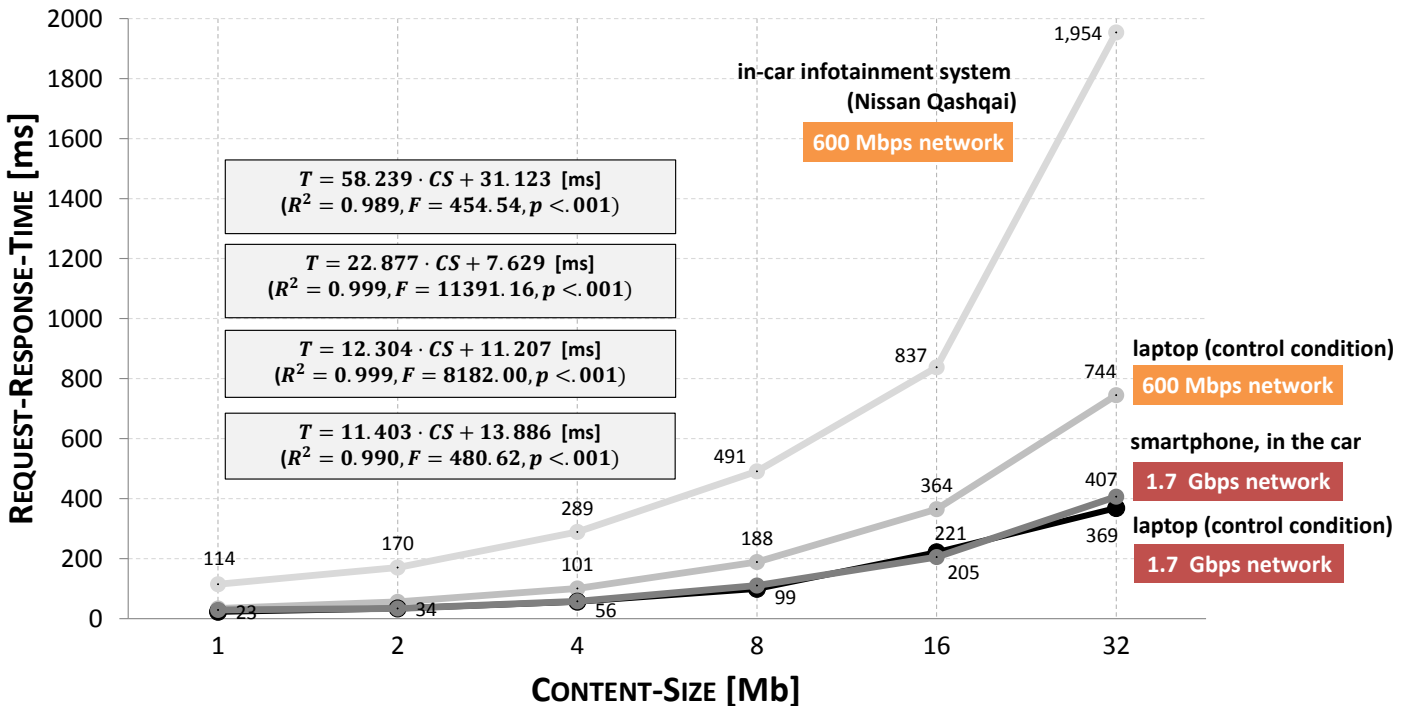

Figure 8. The effect of Content-Size, Device, and Network on the Request-Response-Time needed to stream one second of content of size ranging from 1 Mb to 32 Mb using the Euphoria middleware. *Note:* the horizontal scale is logarithmic.

and reached 837 ms for 1440p@30fps. These results show that the in-vehicle system can effectively deliver 2K content using the Euphoria middleware. However, in the most demanding condition represented by 4K content, the in-vehicle system delivered 1,954 ms, well beyond the 1 s limit. This result shows the limitations of current systems commonly available in today's vehicles (the infotainment system was limited to 600 Mbps connections only, and we could not make it to work with the 1.7 Gbps network) and indicates that, in order to obtain better request-response times, it is necessary to shift to faster network connections, *i.e.*, from the connected car to the hyper-connected car.

In the light of the baseline and integrated system results, we want to understand the potential of the Euphoria middleware to stream high-definition content to a hyper-connected car. To this end, we look at the smartphone condition where the smartphone, used inside the vehicle, was connected to the 1.7 Gbps

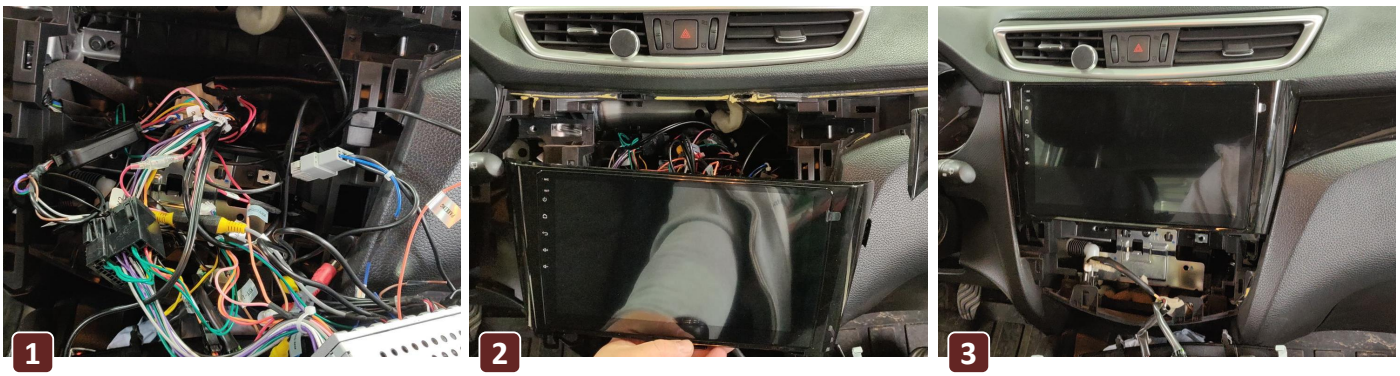

Figure 9. Our empirical results demonstrate the difference in technical performance between the status-quo (600 Mbps networks for the connected car) and what is possible with 1.7 Gbps networks (for the hyper-connected car). Future work will address integration of 1.7 Gbps compatible devices in the vehicle environment (steps 1-3 in the figure and see the text for details), which will open new opportunities for new empirical studies and applications for the hyper-connected car.

wireless network. The request-response time for the most demanding experimental condition (2160p@30fps) was just 407 ms; see Figure 8. This result shows the benefits of moving toward hyper-connected cars, for which Euphoria was able to deliver the same content in just 20.8% of the time needed by the integrated infotainment system using the 600 Mbps network connection.

A regression analysis showed statistically significant linear relationships (R^2>.989, p<.001) between Request-Response-Time (denoted with T in the equations shown in Figure 8) and Content-Size (denoted with CS) for the four Device × Network combinations examined in our experiment. Since the average request-response time measurements for the 1.7 Gbps network were upper bounded by 1,000 ms, we can conclude that the Euphoria middleware supports live streaming of Ultra High-Definition content up to 4K@30fps. The request-response time for the in-vehicle infotainment system indicates that 2K video content can be streamed even over a 600 Mbps network, and the smartphone condition demonstrates that 4K content at 30fps is equally achievable inside the vehicle when shifting to 1.7 Gbps connections.

6. CONCLUSION AND FUTURE WORK

We presented in this paper empirical results regarding software architecture for live streaming of high-definition content inside connected and hyper-connected vehicles. Our contribution was possible by adopting a model for the hyper-connected car as a distinct type of a smart environment, which enabled us to leverage recent advances in software architecture design for general-purpose smart environments, which we adapted and applied for the in-vehicle environment. Our empirical results showed that 2K video can be streamed for 600 Mbps Wi-Fi networks using the Euphoria middleware, but also that 1.7 Gbps support is needed (*i.e.*, moving toward the hyper-connected car in the automotive industry) in order to attain the 4K@30fps content limit inside the vehicle, which we were able to effectively demonstrate with the passenger's smartphone experimental condition. Moreover, the smartphone and the high-end laptop (the control) showed very similar performance for the 1.7 Gbps network. While our empirical results are sufficient to highlight the difference in technical performance between the status-quo (600 Mbps networks) and what is possible with hyper-connectivity (1.7 Gbps networks), future work will look at integrating 1.7 Gbps-compatible devices in the vehicle via the dedicated interfaces (*e.g.*, audio/video input, CanBus, radio antenna, illumination control, etc.); see Figure 9 for an illustration of work in progress. This approach will enable more experiments and reporting of more results in this direction.

Our results also open opportunities for deploying software applications for the hyper-connected car that render high-definition content to drivers and passengers, including AR applications. Future work will explore in-vehicle AR applications, supported by our architecture, such as photorealistic computer-generated graphics to assist drivers with enhanced navigation features, while providing enriched entertainment experiences to passengers. Exploring various input modalities to interact with high-definition content inside the vehicle (Bilius and Vatavu, 2020) represents an equally interesting direction for future work. Moreover, future work is needed to examine the impact of producing and delivering AR content not only in terms of the capabilities of the network, but also in terms of processing power of the CPUs/ECUs. We also believe that our method to model a specific environment as a smart environment in order to borrow and use readily available knowledge and tools could also be applied to other areas of applications for emerging technologies

in the IoT domain that relate to the concept of a smart, connected car.

7. ACKNOWLEDGMENTS

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI - UEFISCDI, project no. PN-III-P1-1.2-PCCDI-2017-0917 (21PCCDI/2018), within PNCDI III. The infrastructure was provided by Ştefan cel Mare University of Suceava and was partially supported by the project *"Integrated center for research, development and innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for fabrication and control,"* No. 671/09.04.2015, Sectoral Operational Program for Increase of the Economic Competitiveness, co-funded by the European Regional Development Fund. Original versions of the icons used to produce Figure 1 were made by Freepik (https://www.flaticon.com/authors/freepik, "Miscellaneous Elements" pack) from Flaticon (https://www.flaticon.com/), free for commercial use with attribution license.

REFERENCES

Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. 2018. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision* 126, 9 (2018), 961–972. DOI:http://doi.org/10.1007/s11263-018-1070-x

Augmented Reality for Enterprise Alliance. 2018. Augmented Reality Functional Requirements. https://thearea.org/area-resources/augmented-reality-functional-requirements/. Accessed: 2019-09-09.

Laura-Bianca Bilius, Radu-Daniel Vatavu. 2020. A Synopsis of Input Modalities for In-Vehicle Infotainment and Consumption of Interactive Media. *Proceedings of the ACM International Conference on Interactive Media Experiences* (2020). DOI:https://doi.org/10.1145/3391614.3399400

Hans Blom, De-Jiu Chen, Henrik Kaijser, Henrik Lönn, Yiannis Papadopoulos, Mark-Oliver Reiser, Ramin Tavakoli Kolagari, and Sara Tucci. 2016. East-ADL: An architecture description language for automotive software-intensive systems in the light of recent use and research. *International Journal of System Dynamics Applications (IJSDA)* 5, 3 (2016), 1–20. DOI:http://doi.org/10.4018/IJSDA.2016070101

O Burkacky, J Deichmann, G Doll, and C Knochenhauer. 2018. Rethinking car software and electronics architecture. *McKinsey & Co., February* (2018). https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/rethinking-car-software-and-electronics-architecture

Chu Cao, Zhenjiang Li, Pengfei Zhou, and Mo Li. 2018. Amateur: Augmented Reality Based Vehicle Navigation System. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)* 2, 4, Article 155 (2018), 24 pages. DOI:http://doi.org/10.1145/3287033

Robert N Charette. 2009. This car runs on code. *IEEE Spectrum* 46, 3 (2009), 3. https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code

Ashley Colley, Jonna Häkkilä, Meri-Tuulia Forsman, Bastian Pfleging, and Florian Alt. 2018. Car Exterior Surface Displays: Exploration in a Real-World Context. In *Proceedings of the 7th ACM International Symposium on Pervasive Displays (PerDis '18)*. ACM, NY, USA, Article 7, 8 pages. DOI:http://doi.org/10.1145/3205873.3205880

Ashley Colley, Jonna Häkkilä, Bastian Pfleging, and Florian Alt. 2017. A Design Space for External Displays on Cars. In *Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications Adjunct (AutomotiveUI '17)*. ACM, 146–151. DOI:http://doi.org/10.1145/3131726.3131760

Christopher Corbett, Elmar Schoch, Frank Kargl, and Felix Preussner. 2016. Automotive Ethernet: Security opportunity or challenge? *Sicherheit 2016-Sicherheit, Schutz und Zuverlässigkeit* (2016). https://dl.gi.de/20.500.12116/880.

Kleidermacher David and Jones Matt. 2012. In-vehicle infotainment software architecture: Genivi and beyond. https://www.eetimes.com/document.asp?doc_id=1279345.

Chris Davies. 2018. Your car windshield could be an AR hologram in 2020. https://www.slashgear.com/porsche-hyundai-wayray-augmented-reality-/holographic-windshield-investment-18546322/. Accessed: 2019-09-09.

Catalin Diaconescu, Dragos Sburlan, and Dorin-Mircea Popovici. 2018. Augmenting Selection by Intention for In-Vehicle Control and Command. In *Proceedings of RoCHI 2018, the 15th International Conference on Human Computer Interaction, 2018*. 107–110. https://dblp.org/rec/bib/conf/rochi/DiaconescuSP18.

Thomas Engen, Lone-Eirin Lervag, and Terje Moen. 2009. Evaluation of IVIS/ADAS Using Driving Simulators. Comparing Performance Measures in Different Environments. In *European Transport Conference, 2009 Association for European Transport (AET)*. https://trid.trb.org/view/907831.

Paolo Gai and Massimo Violante. 2016. Automotive embedded software architecture in the multi-core age. In *2016 21th IEEE European Test Symposium (ETS)*. IEEE, 1–8. DOI:http://doi.org/10.1109/ETS.2016.7519309

Mario Gerla, Eun-Kyu Lee, Giovanni Pau, and Uichin Lee. 2014. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In *2014 IEEE world forum on internet of things (WF-IoT)*. IEEE, 241–246. DOI:http://doi.org/10.1109/WF-IoT.2014.6803166

Bogdan-Florin Gheran, Radu-Daniel Vatavu. 2020. From Controls on the Steering Wheel to Controls on the Finger: Using Smart Rings for In-Vehicle Interactions. *Proc. of the ACM International Conference on Designing Interactive Systems (Companion)* (2020). DOI:https://doi.org/10.1145/3393914.3395851

GreenCarCongres. 2018. Cisco and Hyundai to bring "hyper-connected" car to production in 2019; software defined vehicle architecture. https://www.greencarcongress.com/2018/01/20180113-cisco.html. Accessed: 2019-09-09.

Richie Jose, Gun A. Lee, and Mark Billinghurst. 2016. A Comparative Study of Simulated Augmented Reality Displays for Vehicle Navigation. In *Proc. of the 28th Australian Conf. on Computer-Human Interaction (OzCHI '16)*. ACM, NY, USA, 40–48. DOI:http://doi.org/10.1145/3010915.3010918

Stamatis Karnouskos and Florian Kerschbaum. 2018. Privacy and integrity considerations in hyperconnected autonomous vehicles. *Proc. IEEE* 106, 1 (2018), 160–170. DOI:http://doi.org/10.1109/JPROC.2017.2725339

SeungJun Kim and Anind K Dey. 2016. Augmenting human senses to improve the user experience in cars: applying augmented reality and haptics approaches to reduce cognitive distances. *Multimedia Tools and Applications* 75, 16 (2016), 9587–9607. DOI:http://doi.org/10.1007/s11042-015-2712-4

Changmin Lim, Junyeong Choi, Jong-Il Park, and Hanhoon Park. 2015. Interactive augmented reality system using projector-camera system and smart phone. In *2015 International Symposium on Consumer Electronics (ISCE)*. IEEE, 1–2. DOI:http://doi.org/10.1109/ISCE.2015.7177800

Shanghong Liu. 2019. Global AR/VR Automotive Spending 2017 and 2025. https://www.statista.com/statistics/828499/world-ar-vr-automotive-spending/.

Ning Lu, Nan Cheng, Ning Zhang, Xuemin Shen, and Jon W Mark. 2014. Connected vehicles: Solutions and challenges. *IEEE internet of things journal* 1, 4 (2014), 289–299. https://ieeexplore.ieee.org/document/6823640.

Suresh Marisetty, Durgesh Srivastava, Joel Hoffmann, and Brad Starks. 2009. Low power Intel Architecture platform for in-vehicle infotainment. *Intel Technology Journal* 13, 1 (2009). https://www.eetimes.com/document.asp?doc_id=1276766.

Bertogna Marko. Hercules - High-Performance Real-time Architectures for Low-Power Embedded Systems. https://hercules2020.eu/. Accessed: 2019-09-09.

Milena Milosevic, Milan Z Bjelica, Tomislav Maruna, and Nikola Teslic. 2018. Software Platform for Heterogeneous In-Vehicle Environments. *IEEE Transactions on Consumer Electronics* 64, 2 (2018), 213–221. DOI:http://doi.org/10.1109/TCE.2018.2844737 .

Mohammad Mehdi Moniri, Michael Feld, and Christian Müller. 2012. Personalized in-vehicle information systems: Building an application infrastructure for smart cars in smart spaces. In *2012 Eighth International Conference on Intelligent Environments*. IEEE, 379–382. DOI:http://doi.org/10.1109/IE.2012.40 .

Ryosuke Okuda, Yuki Kajiwara, and Kazuaki Terashima. 2014. A survey of technical trend of ADAS and autonomous driving. In *Technical Papers of 2014 International Symposium on VLSI Design, Automation and Test*. IEEE, 1–4. DOI:http://doi.org/10.1109/VLSI-TSA.2014.6839646 .

Cristina Olaverri-Monreal, Pedro Gomes, Ricardo Fernandes, Fausto Vieira, and Michel Ferreira. 2010. The See-Through System: A VANET-enabled assistant for overtaking maneuvers. In *2010 IEEE Intelligent Vehicles Symposium*. IEEE, 123–128. DOI:http://doi.org/10.1109/IVS.2010.5548020 .

Ovum and Intel. 2018. How 5G Will Transform the Business of Media & Entertainment. https://ovum.informa.com/resources/product-content/intel-5g-ebook.

Patrizio Pelliccione, Eric Knauss, Rogardt Heldal, S Magnus Ågren, Piergiuseppe Mallozzi, Anders Alminger, and Daniel Borgentun. 2017. Automotive architecture framework: The experience of volvo cars. *Journal of systems architecture* 77 (2017), 83–100. DOI:http://doi.org/10.1016/j.sysarc.2017.02.005 .

Jonathan Petit and Steven E Shladover. 2015. Potential cyberattacks on automated vehicles. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2015), 546–556. DOI:http://doi.org/10.1109/TITS.2014.2342271 .

Irina Popovici, Ovidiu-Andrei Schipor, and Radu-Daniel Vatavu. 2019. Hover: Exploring cognitive maps and mid-air pointing for television control. *International Journal of Human-Computer Studies* 129 (2019), 95–107. DOI:http://doi.org/10.1016/j.ijhcs.2019.03.012 .

Hang Qiu, Fawad Ahmad, Ramesh Govindan, Marco Gruteser, Fan Bai, and Gorkem Kar. 2017. Augmented vehicular reality: Enabling extended vision for future vehicles. In *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications*. ACM, 67–72. DOI:http://doi.org/10.1145/3032970.3032976 .

François Rameau, Hyowon Ha, Kyungdon Joo, Jinsoo Choi, Kibaek Park, and In So Kweon. 2016. A real-time augmented reality system to see-through cars. *IEEE transactions on visualization and computer graphics* 22, 11 (2016), 2395–2404. DOI:http://doi.org/10.1109/TVCG.2016.2593768 .

Qing Rao, Christian Grünler, Markus Hammori, and Samarjit Chakraborty. 2014. Design methods for augmented reality in-vehicle infotainment systems. In *Proceedings of the 51st Annual Design Automation Conference*. ACM, 1–6. DOI:http://doi.org/10.1145/2593069.2602973 .

Torben Schinke, Niels Henze, and Susanne Boll. 2010. Visualization of off-screen objects in mobile augmented reality. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*. ACM, 313–316. DOI:http://doi.org/10.1145/1851600.1851655 .

Ovidiu-Andrei Schipor, Radu-Daniel Vatavu. 2019b. Towards Interactions with Augmented Reality Systems in Hyper-Connected Cars. In *Proceedings of the 2nd Workshop on Charting the Way towards Methods and Tools for Advanced Interactive Systems* DOI:http://ceur-ws.org/Vol-2503/paper1_12.pdf .

Ovidiu-Andrei Schipor and Radu-Daniel Vatavu. 2018. Invisible, Inaudible, and Impalpable: Users' Preferences and Memory Performance for Digital Content in Thin Air. *IEEE Pervasive Computing* 17, 4 (2018), 76–85. DOI:http://doi.org/10.1109/MPRV.2018.2873856 .

Ovidiu-Andrei Schipor, Radu-Daniel Vatavu, and Jean Vanderdonckt. 2019. Euphoria: A Scalable, Event-driven Architecture for Designing Interactions across Heterogeneous Devices in Smart Environments. *Information and Software Technology* (2019). DOI:http://doi.org/10.1016/j.infsof.2019.01.006 .

Ovidiu-Andrei Schipor, Wenjun Wu, Wei-Tek Tsai, and Radu-Daniel Vatavu. 2017. Software architecture design for spatially-indexed media in smart environments. *Advances in Electrical and Computer Engineering* 17, 2 (2017), 17–23. DOI:http://doi.org/10.4316/AECE.2017.02003 .

Ovidiu-Andrei Schipor, Radu-Daniel Vatavu, and Wenjun Wu. 2019b. Integrating Peripheral Interaction Into Augmented Reality Applications. *Proc. of the 2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)* (2019), 358-359. DOI:https://doi.org/10.1109/ISMAR-Adjunct.2019.00-12 .

Ovidiu-Andrei Schipor, Radu-Daniel Vatavu, and Wenjun Wu. 2019b. SAPIENS: Towards Software Architecture to Support Peripheral Interaction in Smart Environments. *Proceedings of the ACM on Human-Computer Interaction* 3, EICS (2019), 11. DOI:http://doi.org/10.1145/3331153 .

Kamal Kumar Sharma, Hemant Sharma, and AK Ramani. 2009. mCAR: software framework architecture for in-vehicle pervasive multimedia services. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Vol. 1.

Miroslaw Staron. 2017. *Automotive software architectures: An introduction*. Springer. https://www.springer.com/gp/book/9783319586090.

S. Tilkov, S. Vinoski. 2010. Node.js: Using javascript to build high-performance network programs.. *IEEE Internet Computing* 14, 6 (2010), 80–83. DOI:http://doi.org/10.1109/MIC.2010.145 .

Matthias Traub, Alexander Maier, and Kai L Barbehön. 2017. Future automotive architecture and the impact of IT trends. *IEEE Software* 34, 3 (2017), 27–32. DOI:http://doi.org/10.1109/MS.2017.69 .

Bingjie Yuan, Yan-Ann Chen, and Shaozhen Ye. 2018. A Lightweight Augmented Reality System to See-Through Cars. In *2018 7th International Congress on Advanced Applied Informatics (IIAI-AAI)*. IEEE, 855–860. DOI:http://doi.org/10.1109/IIAI-AAI.2018.00174 .

Bowen Zheng, Hengyi Liang, Qi Zhu, Huafeng Yu, and Chung-Wei Lin. 2016. Next generation automotive architecture modeling and exploration for autonomous driving. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 53–58. DOI:http://doi.org/10.1109/ISVLSI.2016.126 .

B., Markwalter: The path to driverless cars [cta insights]. IEEE Consumer Electronics Magazine **6**(2), 125–126 (2017)

Joshi, Aastha H.: Ethernet communication design with service oriented architecture (SOA) (2019) PhD Thesis.

Baraniuk, C.: Ces 2017: Car-makers choose virtual assistants (2017), http://www.bbc.com/news/technology-38526807

Colquitt, J., Dowsett, D., Gami, A., Equities, I.F., Jaysane-Darr, E., Partner, I.P.C., Manley, C., Equity, F., Shad, R., Income, I.F.: Driverless cars: How innovation paves the road to investment opportunity (2017)

Gopu, G., Kavitha, K., Joy, J.: Service oriented architecture based connectivity of automotive ecus. In: 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT). pp. 1–4. IEEE (2016)

Häckel, T.: A middleware solution for open and dynamic ICT architectures in future cars - hauptprojekt (2018)

Hammoudi, K., Cabani, A., Melkemi, M., Benhabiles, H., Windal, F.: Towards a model of car parking assistance system using camera networks: Slot analysis and communication management. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). pp. 1248–1255. IEEE (2018)

Khanna, A., Anand, R.: Iot based smart parking system. In: 2016 International Conference on Internet of Things and Applications (IOTA). pp. 266–270. IEEE (2016)

Liu, S., Tang, J., Zhang, Z., Gaudiot, J.L.: Computer architectures for autonomous driving. Computer **50**(8), 18–25 (2017)

Lugano, G.: Virtual assistants and self-driving cars. In: 2017 15th International Conference on ITS Telecommunications (ITST). pp. 1–5. IEEE (2017)

Mahendra, B., Sonoli, S., Bhat, N., Raghu, T., et al.: Iot based sensor enabled smart car parking for advanced driver assistance system. In: 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). pp. 2188–2193. IEEE (2017)

Sadio, O., Ngom, I., Lishou, C.: Design and prototyping of a software defined vehicular networking. IEEE Transactions on Vehicular Technology (2019)

Saini, M., Alam, K.M., Guo, H., Alelaiwi, A., El Saddik, A.: InCloud: a Cloud-based middleware for vehicular infotainment systems. Multimedia Tools and Applications **76**(9), 11621–11649 (2017)

Wang, W., Song, Y., Zhang, J., Deng, H.: Automatic parking of vehicles: A review of literatures. International Journal of Automotive Technology **15**(6), 967–978 (2014)

Yun, S., Park, J. H., Kim, W. T. (2017, July). Data-centric middleware based digital twin platform for dependable cyber-physical systems. In 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN) (pp. 922-926). IEEE.

Josifovska, K., Yigitbas, E., Engels, G. (2019). Reference framework for digital twins within cyber-physical systems. In 2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS) (pp. 25-31). IEEE.

Hellwig, A. D., Kriebel, S., Kusmenko, E., Rumpe, B. (2019). Component-based Integration of Interconnected Vehicle Architectures. In 2019 IEEE Intelligent Vehicles Symposium (IV) (pp. 153-158). IEEE.

OPEN Alliance's Technical Committee 8 (2016). ECU and Network Test, Test Specification ECU. https://www.opensig.org/tech-committees/tc. Accessed: 2020-05-29.

AUTOSAR (2016). Testability Protocol and Service Primitives. https://www.autosar.org/fileadmin/user_upload/standards/tests/1-1/AUTOSAR_PRS_TestabilityProtocolAndServicePrimitives.pdf. Accessed: 2020-05-29.

C. Goumopoulos, A. Kameas, P. Hellas, Smart objects as components of ubicomp applications, International Journal of Multimedia and Ubiquitous Engineering 4 (3). http://www.sersc.org/journals/IJMUE/vol4_no3_2009/1.pdf

F. Aiello, G. Fortino, R. Gravina, A. Guerrieri, A java-based agent platform for programming wireless sensor networks, Comput. J. 54 (3) (2011) 439–454. http://dx.doi.org/10.1093/comjnl/bxq019

F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, M. Sgroi, Spine: A domain-specific framework for rapid prototyping of wbsn applications, Softw. Pract. Exper. 41 (3) (2011) 237–265. https://doi.org/10.1002/spe.998

G. Fortino, A. Guerrieri, G. M. P. O'Hare, A. Ruzzelli, A flexible building management framework based on wireless sensor and actuator networks, J. Netw. Comput. Appl. 35 (6) (2012) 1934–1952. http://dx.doi.org/10.1016/j.jnca.2012.07.016

T. Zahariadis, A. Papadakis, F. Alvarez, J. Gonzalez, F. Lopez, F. Facca, Y. Al-Hazmi, Fiware lab: managing resources and services in a cloud federation supporting future internet applications, in: Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on, IEEE, 2014, pp. 792–799. https://doi.org/10.1109/UCC.2014.129

N. Marquardt, R. Diaz-Marino, S. Boring, S. Greenberg, The proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies, in: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST '11, ACM, New York, NY, USA, 2011, pp. 315–326. http://doi.acm.org/10.1145/2047196.2047238

G. Fortino, D. Parisi, V. Pirrone, G. Di Fatta, Bodycloud: A saas approach for community body sensor networks, Future Gener. Comput. Syst. 35 (2014) 62–79. http://dx.doi.org/10.1016/j.future.2013.12.015

M. Nebeling, E. Teunissen, M. Husmann, M. C. Norrie, Xdkinect: Development framework for cross-device interaction using kinect, in: Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS '14, ACM, New York, NY, USA, 2014, pp. 65–74. http://doi.acm.org/10.1145/2607023.2607024

D. Ledo, S. Greenberg, N. Marquardt, S. Boring, Proxemic-aware controls: Designing remote controls for ubiquitous computing ecologies, in: Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI '15, ACM, New York, NY, USA, 2015, pp. 187–198. http://doi.acm.org/10.1145/2785830.2785871

C. Roda, A. Rodríguez, E. Navarro, V. López-Jaquero, P. González, Towards an architecture for a scalable and collaborative ami environment, in: Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection, Springer, 2016, pp. 311–323. https://doi.org/10.1007/978-3-319-40159-1_26

I. Mocanu, O. A. Schipor, A serious game for improving elderly mobility based on user emotional state, in: The International Scientific Conference eLearning and Software for Education, Vol. 2, " Carol I" National Defence University, 2017, p. 487. https://doi.org/10.12753/2066-026x-17-154

Y. Lou, W. Wu, R.-D. Vatavu, W.-T. Tsai, Personalized gesture interactions for cyber-physical smart-home environments, Science China Information Sciences 60 (7) (2017) 072104. https://doi.org/10.1007/s11432-015-1014-7

R.-D. Vatavu, C.-M. Chera, W.-T. Tsai, Gesture profile for web services: An event-driven architecture to support gestural interfaces for smart environments, in: International Joint Conference on Ambient Intelligence, Springer, 2012, pp. 161–176. https://doi.org/10.1007/978-3-642-34898-3_11