VEHICULAR NETWORKING

SDN-Sim: Integrating a System-Level Simulator with a Software Defined Network

S. Ghosh, S. A. Busari, T. Dagiuklas, M. Iqbal, R. Mumtaz, J. Gonzalez, S. Stavrou, and L. Kanaris

ABSTRACT

With the introduction of diverse technology paradigms in next-generation cellular and vehicular networks, design and structural complexity are skyrocketing. The beyond-5G use cases such as mobile broadband, 5G-V2X, and UAV communications require support for ultra-low latency and high throughput and reliability with limited operational complexity and cost. These use cases are being explored in 3GPP Releases 16 and 17. To facilitate end-to-end performance evaluation for these applications, we propose SDN-Sim — an integration of a system-level simulator (SLS) with a software defined network (SDN) infrastructure. While the SLS models the communication channel and evaluates system performance on the physical and data link layers, the SDN performs network and application tasks including routing, load balancing, and so on. The proposed architecture replicates the SLS-defined topology into an SDN emulator for offloading control operations. It uses link and node information calculated by the SLS to compute routes in SDN and feeds the results back to the SLS. Along with the architecture, data modeling, and processing, replication and route calculation frameworks are proposed.

INTRODUCTION

Toward 5G/beyond 5G (B5G), the 3rd Generation Partnership Project (3GPP) is finalizing Release 16 and defining Release 17. In the area of vehicular networks, 3GPP, in partnership with the Fifth Generation Automotive Association (5GAA), is driving the efforts on 5G-based the LTE-V2X to 5G-V2X, the authors in [1] advocated the incorporation of software-defined networking (SDN) in the architecture to enhance the system performance through SDN’s capabilities in facilitating intelligent multihop routing, dynamic resource allocation, and advanced mobility support, among others.

To evaluate the performance of proposed algorithms, techniques, and frameworks for any new era of communication networks, numerical simulations, mathematical analyses, and field trials are the three main approaches being employed. Although analytically tractable, mathematical methods (e.g., stochastic geometry tools) are often constrained with simplifying assumptions that potentially limit their use in modeling large-scale, highly complex, and dynamic networks. Realistic performance can be measured in live operating environments. However, the economic and operational requirements are costly and practically infeasible for the early design and development stages. Hence, in the past few decades, simulations have become remarkably important tools for the assessment of networks’ performance due to the obvious cost and implementation advantages [4].

Depending on the performance metrics under investigation, simulators can be categorized into three: link-level simulators (LLSs), system-level simulators (SLSs), and network-level simulators (NLSs). An LLS examines detailed, bit-level physical (PHY) layer functionalities of a single link. An SLS evaluates the performance of links involving many base stations (BSs) and user equipments (UEs) at the medium access control (MAC) layer (with the PHY abstracted). It focuses on the radio access network/air interface and facilitates analyses of resource allocation, capacity, coverage, and spectral and energy efficiencies, among others. The NLS, however, assesses the performance of protocols across all layers of the network, including control signaling and backhaul/fronthaul issues. Performance is characterized using metrics including latency, packet loss, and so on [5].

Besides metric-based classification, simulators can also be grouped based on radio access technologies supported (cellular, vehicular, WiFi, etc.), coding language (MATLAB, Python, C++, etc.), licensing option (open source, proprietary, free of charge for academic use) or network scenario capabilities (LTE, 5G, B5G, etc.) [3]. While the SLS does not simulate beyond the MAC layer, the NLS simulates networks up to the application layer. However, the implementation and computational complexity of the NLS becomes very high when a large number of nodes are involved [4].

Another major paradigm shift in network design took place with the advent of SDN [6]. It decouples the control (signaling) plane from the data (forwarding) plane and runs applications in the application plane to manage the network. This brings transparency in network design and lets software developers write applications for man-
Another major paradigm shift in network design took place with the advent of SDN [6]. It decouples the control (signaling) plane from data (forwarding) plane and runs applications in the Application Plane to manage the network. This brings transparency in network design and lets software developers write applications for managing the networks, keeping the internal design in abstraction.

**Figure 1.** Schematic system architecture of SDN-Sim with full stack setup along with their core functions.

Therefore, SDN offers better modularity, programmability, agility, automation, and load balancing capability than do traditional networks. Also, the SDN-based approach is used in network design practices for cloud computing and 5G.

In this work, we present a novel SDN-based system-level simulator (SDN-Sim) platform where the SLS-Stage runs in MATLAB and the NLS-stage is based on python3. By inheriting all the benefits of SDN, the architecture is designed to reduce the overall computational complexity of the system considerably. The computationally demanding upper layer network functions (e.g., inter-cellular routing) are offloaded to the virtualized cloud infrastructure. Low-level network information (e.g., channel model, topology) are maped from SLS to SDN. Python is used for application development. Virtual infrastructure is set up using VMWare ESXi servers. OpenFlow and RESTConf are used as south and northbound protocols, respectively. Opendaylight is used as the SDN controller, while GNS3 and Mininet-wifi are used for data plane emulation.

This article proposes an architecture to integrate MATLAB-based SLS with SDN using OpenDaylight, Mininet-Wifi, and GNS3. The implementation is described with data modeling of the middleware and sequence modeling of messages exchanged among various entities.
Three applications (TopoBuild, TopoSense, and TopoRoute) are also presented for data fetching, interfacing, and processing. The remainder of this article is organized as follows. We describe the system architecture and implementation. The experimental setup and results are presented, followed by conclusions and future research directions.

**System Architecture and Implementation**

Figure 2 depicts the system architecture of SDN-Sim, with the SLS at the bottom and the SDN infrastructure running on the top. When the SLS does the channel modeling and scheduling, the SDN takes care of the upper layer functionalists such as IP routing and traffic control, described as follows.

**System-Level Simulator**

The tasks of the SLS (described below) are executed in loops of transmission time intervals (TTIs), and the results are averaged over several simulation runs or channel realizations [7].

**Scenario Setting:** The layout depicts a vehicular network of BSs/roadside units (RSUs) and vehicle mounted radio/UEs, configured with 3D locations of the nodes. UEs and vehicles are mobile and attached to their serving RSU, which further interfaces with the cellular networks through BSs with parameters such as line-of-sight probability, distance, and signal-to-noise ratio (SNR).

**Channel Modeling:** For all links, the path loss (PL), shadow fading (SF), transmit and receive antenna gains, and fast fading are calculated to estimate the channel of each user for both desired and interfering links.

**Scheduling:** Radio resources are allocated to users based on the scheduling algorithm. Resource blocks (bandwidth) and power are allocated either in quasi-random fashion (for open loop configuration) or based on feedback from the users in closed loop systems. The channel state information (CSI) feedback, together with other factors such as the traffic type of users, link adaptation strategy employed, and quality of service (QoS) demands, are used as decision determinants at the scheduling stage.

**Link Quality and Performance Estimation:** The signal-to-interference-plus-noise ratio (SINR) of the links are then estimated. Using the SINRs and the link abstraction model (for the block error rate, BLER), the throughput of the users and the capacities of the cells are calculated.

### Table 1: A comparative study of GNS3 and Mininet wifi as data plane engine.

|                  | GNS3          | Mininet-Wifi |
|------------------|---------------|--------------|
| **Similarities** | Both are open source, offer GUI based network design, support API-based interfacing with the external environment, have accessible physical interface, and host OVSs and dummy workstations for data plane |               |
| **Advantages**   | Not limited to only SDNs Use of actual OVSs Test result accuracy Realistic test cases Scalable via clustering Native Docker support Extensibility with apps from GNS3 marketplace | Rapid Implementation, no SDN/ OpenFlow configuration needed Link configuration Wireless support Native SDN support Built-in mobility and propagation models |
| **Demerits**     | No wireless support Manual configuration for SDN/OpenFlow and other supporting devices No link configuration Requires more physical resource to set up test environment | Limited to SDNs Test-result accuracy Scaliability via multi-threading No clustering supports No paravirtualization No app-based functional extensibility |

**Figure 2.** Sequence diagram for various message exchanges between components of SDN-Sim.
Recent developments in SLSs are mainly focused on solving problems on channel modeling [8], high-frequency communication [9, 10], coexistence and performance optimization [11], energy efficiency, latency, scheduling, and load balancing over a heterogeneous network [12], among others. The 5G Public-Private Partnership (5GPPP) project has described several aspects in softwarization, service management and orchestration in their architectural reference [13], which includes SDN, cloud computing, virtualization, and so on. Therefore, cloud computing is being employed to enhance the scalability and computational efficiency of 5G SLSs by offloading the computational load [14].

**SOFTWARE DEFINED NETWORK**

The tasks of the SDN are described as follows.

**Channel Information Update:** The SLS running in a MATLAB server with Open Database Connector (ODBC) driver updates the topology and channel modeling parameters to a centralized database.

**SDN Emulation:** The topology information is reactively fetched and emulates using TopoBuild in an SDN data plane through GNS3 and Mininet-wifi. The control plane starts communicating with Open-Flow.

**Route Calculation:** The application plane fetches topology from the control plane, translates it into a graph (using TopoSense), and finds the shortest paths between node pairs (using TopoRoute). These paths are fed back to the database and the controllers.

The workflow and communication sequence taking place between the various elements of the architecture are as given in Fig. 2.

**Update Communication Parameters:** The SLS performs channel modeling, radio resource allocation, and system performance evaluation. After optimizing the model for a given scenario, it generates several channel parameters (bandwidth, distance, path loss, latency, and delay) and BS parameters (position, range, received signal strength indication [RSSI], transmission power, etc.). Since parameters are calculated per BS, each BS updates their local dataset to the centralized database.

**Fetch Node and Link Parameters:** TopoBuild is a bespoke program to fetch SLS parameters from the central database and relay them to the SDN data plane for replicating the topology. BSs are placed as wireless access points, running OpenFlow protocol. Links, depending on types (i.e., fronthaul or backhaul), are assigned to BSs. A fronthaul link is wireless, therefore carrying several radio parameters including path loss, frequency band, RSSI, and so on.

**Topology Replication:** TopoBuild translates parameters obtained from the central database into a series of commands. In the proposed architecture, there are two possible infrastructures (GNS3 and Mininet-wifi) with three possible deployment options (full GNS3, full Mininet-wifi, and hybrid). Based on the deployment type and connection specifications, TopoBuild.py generates a script to replicate the topology and injects it to the specific engines (GNS3 and/or Mininet-wifi). A feature comparison between GNS3 and Mininet-Wifi is given in Table 1 and briefly described as follows.

| Table 2. Descriptions of attributes of the data model. | Node attributes |
|--------------------------------------------------------|-----------------|
| Node ID | Primary key, unique ID for each node |
| Type | Type of node (AP or host) |
| Range | Communication range of the node |
| Position | Location of node (3D Cartesian coordinate) |
| Channel | Operating channel number |
| Frequency | Operating frequency (in Hz) |
| Mode | Operating mode (G/GE/N/AC) |
| Tx_Power | Transmit power in megawatts |
| IP_Address | Node’s IP address |
| MAC_Address | Node’s MAC address |

| Access point attributes |
|--------------------------|
| Station association | List of host the AP is associated |
| AP_Association | ID of the AP to which the host is associated |
| RSSI | Relative received signal strength (at host end) |

| Flow table attribute |
|----------------------|
| DPID | Unique data path ID of the AP |
| Source_IP | Match field for source IP in ingress packet |
| Destination_IP | Match field for destination IP in ingress packet |
| Source_MAC | Match field for source MAC in ingress frame |
| Destination_MAC | Match field for destination MAC in ingress frame |
| Action | OpenFlow Action opcode |
| Timeout | Timeout (dead) timer |
| Packet_Count | Total packet count statistics |
| Byte_Count | Total bytes count statistics |
| Duration | Hold time for OpenFlow entry |

| Node_Channel_Map attributes |
|-----------------------------|
| Channel_ID | Foreign key to unique channel ID in table |
| Node_1 | Foreign key to first node ID |
| Node_2 | Foreign key to second node ID |

| Channel attributes |
|---------------------|
| Channel_ID | Primary key, channel identifier |
| Bandwidth | Channel bandwidth (megabits per second) |
| Distance | Distance between incident nodes in meters |
| Path loss | Path loss of the channel |
| Latency | Average latency (in milliseconds) |
| Delay | Average RTT (in ms) |
GNS3 Infrastructure: GNS3 is an open source network emulation software. It comprises a controller unit that interfaces through RESTful application programming interfaces (APIs), manages topology and so on, and a cluster of para-virtualized compute nodes that host virtual network functions (VNFs) such as routers, switches, firewalls as containers, or virtual machines (VMs). In SDN-Sim, Topobuild communicates with GNS3-Controller using a RESTful API, and the Open-V-switches (OVSs) run as Docker containers within the GNS3-Compute cluster. Therefore, GNS3 offers realistic test cases with a tested and produces more authentic results than those of a simulation. A downside is that it lacks support for wireless networking emulation; however, it allows access to a physical wireless card.

Mininet-WiFi Infrastructure: With a Wi-Fi extension, inheriting all the features of Mininet, it can now emulate wireless SDNs. With support of its python API, programming and configuring Mininet-Wifi is more user-friendly. Mininet-Wifi runs in a single sandbox with its own interactive command line. Thus, Topobuild uses raw sockets to inject Mininet-wifi commands for deployment and parameters updates. It also offers four mobility and five propagation models. Since Mininet-Wifi uses HWSIM drivers to simulate wireless networking, emulating a very large network is constrained. Also, it does not support clustering.

Accessing Control Plane with OpenFlow: Both GNS3 and Mininet-wifi host OVSs; thus, they communicate with the controller using OpenFlow v1.3 (OF1.3) as southbound protocol. OF1.3 uses bidirectional messaging to communicate with the switches. A switch requests a controller with a Packet-In message, and the controller replies back with a Packet-Out. SDN-Sim uses OpenDaylight — Beryllium SR4 (ODL) as controller, which runs as a VM. Both GNS3 (4.1) and Mininet-wifi (4.2) use TCP port 6633 and/or 6635 for communication. The control plane supports binding of several controller nodes with clustering to maintain scalability, high availability, and persistence in the data plane. ODL supports “Akka” clustering for this purpose.

Interfacing with the Application Plane with RESTConf: SDN-Controller interacts with the application plane using Northbound APIs. ODL uses RESTConf (RFC 8040). It is a RESTful version of NETCONF (RFC 6241) protocol, and uses JSON (RFC-7159) format to transfer data among REST enabled devices. In SDN-Sim, we use two RESTConf resources (Inventory and Topology). “Operational” resources are to read, and “Config” are to write. The inventory resource provides node-wise OpenFlow tables, and the topology of the network is provided by the Topology resource. The App TopoSense makes use of the resources to model a graph by fusing topology and flow-table information. TopoSense invokes TopoRoute to calculate the route for a given topology.

Route Calculation: TopoRoute uses the stochastic temporal edge normalization (STEN) [15] technique to find routes. It receives the topology and flow tables from TopoSense. The link parameters are already present in the database. By fusing them, a single source shortest path algorithm is run over every pair of vertices. and a set of all
possible routes are generated. In a traditional network, local routes are shared among neighbors to form routing tables. With the size of the network, due to propagation delay, the routing becomes significantly slow. However, in an SDN paradigm, discovery is done by the controller; hence, the topology graph is mapped proactively. Therefore, the shortest paths between every pair can be calculated in parallel. This speeds up the routing process and allows scalability in the network.

**OpenFlow Tables Update:** TopoRoute does event-driven update of the flow tables in the Central DB (7.1). The calculated routes are fed back to TopoSense, which eventually sends the routes back to the controller using the RESTConf Inventory-Config API. Once the central DB is updated, SLS picks the information, and an inter-cellular route is discovered to leverage the lower (physical and data link) layer operations. The complete operational sequence diagram is given in Fig. 2.

**Data Modeling and Design of the Central Database**

The central database plays the role of middleware between the SLS and the SDN. Table 2 shows the related entities and their corresponding attributes for the data model. There are three main entities: node, flow table, and channel. A node can be either an access point (AP) or a host. One host is associated with one AP, and one AP can associate with many hosts. Each AP is an OVS; thus, it contains flow table(s) and a unique data path identifier (DPID). A pair of nodes makes a channel, with a unique channel ID (mapping is recorded at Node_Channel_Map table). A channel between a host and an AP is called fronthaul (wireless), and between a pair of APs a backhaul (wired). No channel exists between two hosts. Since TopoSense reads information from the APs, the route calculation by TopoRoute takes place over the backhaul network.
Data analytics empowered by deep learning algorithms will learn the runtime behavior of the network and help to improve the network automation, self-organization, and state prediction ability. The proposed simulator facilitates the end-to-end performance evaluation of 5G and beyond 5G use cases such as 5G-V2X, where latency is a critical performance metric.

**DESIGN OF APPLICATION PLANE**

Three major apps in the application plane run three python apps: TopoBuild.py, TopoSense.py, and TopoRoute.py. In the previous sections, their usage has been mentioned. Figure 3 depicts their working principles using flowcharts.

**RESULTS**

Figure 4 depicts both the time complexity of various phases and the latency for three use cases (1, 7, and 19 sites; 3 sectors/each sector consists of 1, 25, 50, 75, 100 users) depicted in Figs. 4a–4c, respectively. The backhaul topology is configured wired and evaluated for linear (minimal) and mesh (maximal) connectivity. The time to build the network on an SDN platform is termed **Build Time**, and **Response Time** is the time the SDN takes to initiate traffic flow. Figure 4d compares the breakdown of time consumption, number of links, and routing time; a multi-threaded implementation of a shortest path algorithm limits the reactive route selection into sub-second intervals.

**EXPERIMENTAL SETUP**

During experiment, the following compute-node setup was followed. The SLS runs in a MATLAB server VM with the Database tool running, and an ODBC adapter connects to MySQL Database. A Mininet-wifi VM hosts BSs as APs, and OVSs and UEs as stations. ODL runs in an Ubuntu 64-bit 14.04 VM, and the application server VM hosts the MySQL Server. The database server along with the TopoSense.py, TopoBuild.py, and TopoRoute.py apps. A VMWare ESXi 6.5 server is used for the virtualization.

**LATENCY AND TIME COMPLEXITY**

Figure 4 depicts the latency of several stages of integration; SLS shares most of it. The setup phase consumes a significant amount of time depending on the network size and topology, which includes channel allocation and scheduling, SDN setup, flow table population, proactive route calculation, and so on. However, the runtime is reactive in nature, and response is in milliseconds (Fig. 4e). All the routes are pre-calculated and the network runs on a centralized virtual platform, which eliminates control packet exchange between devices to learn network topology. The time complexity of the proactive phase is of a high degree polynomial class, and the reactive phase is constant; thus, once the SDN is deployed, response time comes down to milliseconds scale.

**CONCLUSION AND FUTURE WORK**

In this article, we propose an architecture and methodology to integrate a system-level simulator with a software-defined networking infrastructure using a relational database as middleware. A comparative analysis on using a GNSS and Mininet-wifi-based data plane is given. A sequence model shows the data and control flow among several percolating entities. The data model of the middleware describes the information structuring, and a set of algorithms to fetch and replicate will extend this architecture by appending an analytics plane on top of it. Data analytics empowered by deep learning algorithms will learn the runtime behavior of the network and help to improve the network automation, self-organization, and state prediction ability. The proposed simulator facilitates the end-to-end performance evaluation of 5G and beyond 5G use cases such as 5G-V2X, where latency is a critical performance metric.

**ACKNOWLEDGMENT**

This project is funded by MSCA-RISE — Marie Sklodowska-Curie Research and Innovation Staff Exchange (RISE) scheme under the project SONNET (Self-Organisation towards minimized cost and eEnergy per bit for future Emerging radio Technologies) with grant agreement no. H2020-MSCA-RISE-2016-734545.

**REFERENCES**

[1] L. Zhao et al., “Vehicular Communications: Standardization and Open Issues,” IEEE Commun. Standards Mag., vol. 2, no. 4, Dec. 2018, pp. 74–80.
[2] S. A. Busari et al., “Millimetre-Wave Massive MIMO for Cellular Vehicle-to-Infrastructure Communication,” IET Intelligent Transport Systems, vol. 13, no. 6, June 2019, pp. 983–90.
[3] S. A. Busari et al., “Generalized Hybrid Beamforming for Vehicular Connectivity Using THz Massive MIMO,” IEEE Trans. Vehic. Tech., vol. 68, no. 9, Sept. 2019, pp. 8372–83.
[4] M. K. Müller et al., “Flexible Multi-Node Simulation of Cellular Mobile Communications: the Vienna 5G System Level Simulator,” EURASIP J. Wireless Commun. and Networking, vol. 2018, no. 1, Sept. 2018, p. 227.
[5] P. Alvarez et al., “Simulating Dense Small Cell Networks,” 2016 IEEE Wireless Commun. and Netw. Conf., Apr. 2016, pp. 1–6.
[6] D. Kreutz et al., “Software-Defined Networking: A Comprehensive Survey,” Proc. IEEE, vol. 103, no. 1, Jan. 2015, pp. 14–76.
[7] M. Rupp, S. Schwartz, and M. Tarantasis, The Vienna LTE-Advanced System Level Simulator, Springer, 2016.
[8] S. A. Busari et al., “Impact of 3D Channel Modeling for Ultra-High Speed Beyond-5G Networks,” 2018 IEEE GLOBECOM Work., Dec 2018, pp. 1–6.
[9] M. Xiao et al., “Millimeter Wave Communications for Future Mobile Networks,” IEEE JSAC, vol. 35, no. 9, Sept. 2017, pp. 1909–15.
[10] M. Mezzavilla et al., “End-to-End Simulation of 5G mmWave Networks,” IEEE Commun. Surveys & Tutorials, vol. 20, no. 3, 3rd qtr. 2018, pp. 2237–63.
[11] L. Zhang et al., “Performance Analysis and Optimization in Downlink NOMA Systems with Cooperative Full-Duplex Relaying,” IEEE JSAC, vol. 35, no. 10, Oct. 2017, pp. 2388–2412.
[12] G. Yang et al., “Low-Latency Heterogeneous Networks with Millimeter-Wave Communications,” IEEE Commun. Mag., vol. 56, no. 6, June 2018, pp. 124–29.
[13] 5GPPP, “View on 5G Architecture White Paper,” ONF, Tech. Rep., 2017; https://5gpp.eu/wp-content/uploads/2018/01/5GPPP-SA05-Architecture-White-Paper-Jan-2018-v2.0.pdf, accessed 16 July 2019.
[14] Y. Wang, L. Xu and L. Jiang, “Challenges of System-Level Simulations and Performance Evaluation for 5G Wireless Networks,” IEEE Access, vol. 2, 2014, pp. 1553–61.
[15] S. Ghosh, T. Dagdugias, and M. Iqbal, “Energy-Aware IP Routing Over SDN,” 2018 IEEE GLOBECOM, Dec. 2018, pp. 1–7.

**BIographies**

S. Ghosh has obtained his M.E. in software engineering, B.Sc. (Hons.) and M.Sc. in computer science, and M.Sc. in smart networks. He is also a recipient of CATE and Erasmus Mundus scholarships. He is currently working toward a Ph.D. in computer science and informatics, supported by the Marie-Curie Fund in the research area of machine learning’s application in self-organised SDN for 5G. His areas of interest include network programmability and automation, IoT, and deep reinforcement learning.

S. A. Busari received his B.Eng. and M.Eng. in electrical and electronics engineering from the Federal University of Technology Akure, Nigeria, in 2011 and 2015, respectively. He is currently an assistant lecturer at the aforementioned university, contributing toward the teaching of the undergraduate module.
in electronic and electrical engineering. His research interests include wireless channel modeling, millimeter-wave and terahertz communication protocols, massive MIMO, and system-level simulation methodologies for 5G and beyond networks.

T. DAGIAKLAS is the leader of the SuITE Research Group at London South Bank University. His research interests include smart internet technologies, media optimization across heterogeneous networks, QoE, virtual reality, augmented reality, and cloud infrastructures and services.

M. IQBAL is a senior lecturer in mobile computing in the Division of Computer Science and Informatics, School of Engineering. He is an established researcher and expert in the fields of mobile cloud computing and open-based networking for applications in disaster management and healthcare, community networks, and smart cities.

R. MUMTAZ has more than 12 years’ R&D experience in the wireless communications industry, where his present appointment is senior research scientist and technical manager at GS, Portugal. Prior to his current position, he worked as a research intern at Ericsson and Huawei Research Labs in 2005 at Karlskrona, Sweden. He received his M.Sc. from the Blekinge Institute of Technology (BTH), Karlskrona in 2011, and obtained his doctoral degree in 2016. His research interests include 5G radio communication protocols and architectures, and quantum communications.

J. GONZAELZ has more than 15 years’ R&D experience in mobile communications and practical experimentation. He obtained his M.Sc., and Ph.D. degrees in telecommunications from the University of Surrey, United Kingdom, in 1999 and 2004, respectively. He then became a senior researcher at the University of Surrey, where he was responsible for project development and research on radio protocols. In 2011, he founded GS, Portugal, and currently is the Project Coordinator for the H2020-SONNET project. His research interests include simulation methodologies and 5G radio resource management.

S. STAVROU holds a B.Eng. degree in computers and communications from the University of Essex, United Kingdom, and a Ph.D. degree in electronic engineering (telecommunications) from the University of Surrey, where he was a faculty member between 1997 and 2009. Currently, he is the dean of faculty of pure and applied sciences at the Open University of Cyprus, leading the Telecommunication Systems Research Lab. His research interests include localization, wireless sensor networks, smart cities, and IoT networks. He is a Fellow of the Higher Education Academy U.K. and the Chairman of Cyber Configuration of the European Security and Defence College.

L. KANARIS received a B.Sc. in aviation science from the Hellenic Airforce Academy in Athens. After obtaining an M.B.A., he worked on several research projects of the Cyprus MoD and represented the Republic of Cyprus in several technical telecom workgroups of international organizations. He is a member of the R&D Department of SIGINT. He has actively participated in various EU and national research projects investigating wireless positioning aspects. He is currently studying toward a Ph.D. (TU/e) concentrating on wireless positioning techniques and the evaluation of fingerprint databases.