Enabling Heterogeneous 5G Simulations with SDN Adapters

Thien Pham(✉), Jeremy McMahon, and Hung Nguyen

Teletraffic Research Centre, The University of Adelaide,
Adelaide, SA 5005, Australia
{thien.pham,jeremy.mcmahon,hung.nguyen}@adelaide.edu.au

Abstract. 5G networks are expected to consist of multiple radio access technologies with a Software-defined networking (SDN) core, and so simulating these networks will require connecting multiple subnetworks with different technologies. Despite the availability of simulators for various technologies, there is currently no tool that can simulate a complete heterogeneous 5G network. In this work, we develop a novel SDN adapter to enable seamless inter-working between different simulation/emulation tools, such as NS-3, Mininet-WiFi, Omnet++, and OpenAirInterface5G. Using the adapter, we have built a large scale 5G simulator with multiple networking technologies by connecting existing simulators. We show that our adapter solution is easy-to-use, scalable, and can be used to connect arbitrary simulation tools. Using our solution, we show that Mininet-WiFi exhibits unreliable behaviour when connected to other networks. We compare our solution against other alternatives and show that our solution is superior both in terms of performance and cost. Finally, and for the first time, we simulate a large heterogeneous 5G network with all of the latest technologies using only a standard commodity personal computer.

Keywords: Simulation · Cross domain · Interoperability · Network slicing · SDN · NFV · LTE · 5G NR

1 Introduction

5G networks will consist of multiple radio access technologies along with a Software Defined Networking (SDN) core to allow for a large number of end devices and flexible network deployment. A typical complete 5G is illustrated in Fig. 1 [4,5,24,25]. In this network, we have an SDN core where network function virtualization and network slicing are used to provide different services to different end users. Multiple access technologies such as WiFi-6, LTE, and 5G New Radio are being, and will be, used to connect end devices to the core.

Emulation software such as OpenAirInterface5G (OAI5G) [19] and srsLTE [14] have been used to evaluate the performance of 5G networks. They are, however, resource demanding and require actual RF transmission over the air

H. Gao et al. (Eds.): TridentCom 2019, LNICST 309, pp. 163–183, 2020.
https://doi.org/10.1007/978-3-030-43215-7_12

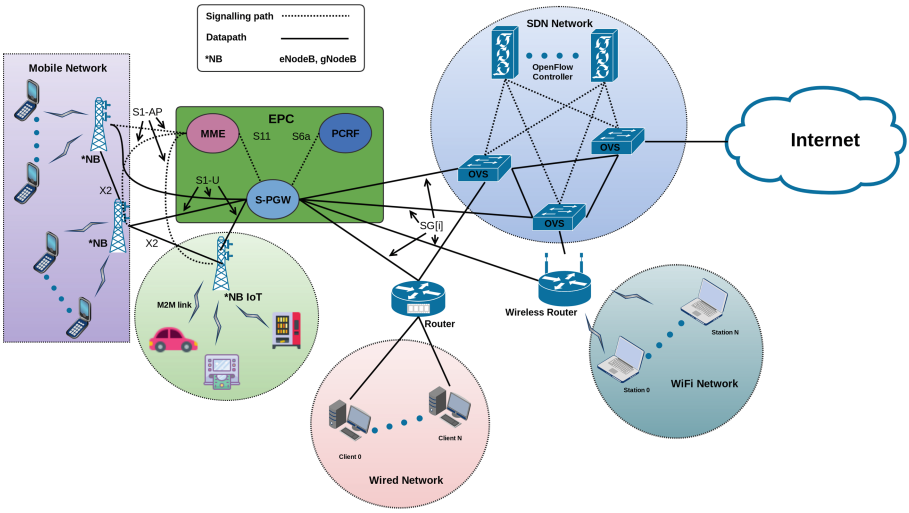


Fig. 1. The SDN-enabled 5G heterogeneous network

for real cellular connections. To run these emulations, end users must have a valid LTE spectrum license and furthermore, both tools support only up to 16 User Equipments (UEs).

Alternatively, network simulation tools provide a scalable and repeatable mechanism for researchers and engineers to test their ideas. Currently, there are no readily available simulation tools that can provide both heterogeneous access technologies and an SDN enabled core network. In the absence of a single tool fit for purpose, an alternative approach to simulate networks similar to those shown in Fig. 1 is to bridge multiple simulations together. As an example, we could bridge Mininet/Mininet-WiFi and Omnet++ so that we have the complementing access technologies of the separate tools together in a single network scenario. Nevertheless, bridging such tools is not necessarily simple. IP packets generated from one simulation often cannot cross to the other simulation over a simple interface for various reasons. Of the scenarios studied, these connectivity reasons include subnet differentiation, the lack of an address resolution mechanism (ARP), the network address translation process (NAT), and a missing external interface as the hook-in destination for simulator-side socket connections and the standard transport-level medium. To achieve this bridging, we investigate existing techniques but ultimately construct an SDN adapter for this bridging purpose. Our SDN adapter is versatile in both architecture and control logic, and is designed to address packet incompatibility issues at the transport level of the networks.

Our solution comes with a GUI to enable easy set up of the various simulation combinations. We show in this paper that the resulting 5G simulator with our SDN adapter is easy to use, scalable, and performs better than alternative solutions. Our 5G simulation solution, together with a number of examples where

multiple simulations are combined together to produce a complete end-to-end 5G network are publicly available [26,27]. Our tool enables the end user to set up 5G networks simulations and performance validation with ease.

2 Background

There are more than 30 network simulation software packages available and each of these simulations is generally designed for a specific use case. For example, some simulators are developed for targeted networks, eg. WiFi networks. Furthermore, some simulators are resource demanding and are not suitable to simulate large scale networks with hundreds of nodes. Reviews and comparison studies of these tools can be found in [18,20,23]. Most importantly, there is no current tool that allows us to simulate large heterogeneous 5G networks.

2.1 Network Simulation Tools

We concentrate here on network simulation tools that have capabilities to simulate the latest 5G technologies, especially the ability to simulate heterogeneous networks such as those described in Fig. 1.

NS3 - Network Simulator 3. NS3 has been widely used by academia and industry [28] for building networks with different technologies such as WLAN, WiFi, LTE, 6LowPAN and physical layer properties including mobility and RF propagation. Recently, **OFSwitch13** was introduced into NS3 by Chaves et al. [9]. The NS3 **LENA** project [15,16] was designed to simulate 4G LTE and is not capable of modeling 5G New Radio. Although multiple technology stacks can coexist in the same network simulation source code (a C++ description of the simulation), some simulations require taking control of the global configuration states via **Config::SetDefault** and **GlobalValue::Bind** such as TCP SegmentSize, ChecksumEnabled, and ChannelType. Each simulation stack is designed and tested in very specific and narrow scenarios without interoperability consideration and intervention. For example, to simulate a heterogeneous network we would need to connect the simulated wired, WiFi and LTE networks to a common SDN-powered OFSwitch13 backhaul. We have found in our simulations that when enabling every configuration of the required networking stacks, the simulations terminated early due to incomplete configuration. If we turn off the configuration of either the LENA LTE or OFSwitch13 stack, the NS3 script is able to execute again. There is currently no published report of successful integration of LENA LTE with OFSwitch in NS3.

Mininet-WiFi. Mininet-WiFi is the most popular emulation tool for SDN-enabled WiFi networks [12,13]. Mininet-WiFi can emulate WLAN and WiFi networks but has no support for LTE and has an upper limit of 100 nodes supporting WiFi and 6LowPAN. As Mininet-WiFi itself does not provide LTE

and other access technologies, to simulate a heterogeneous 5G network, we will need to connect Mininet-WiFi with other simulators. In this paper, we refer to Mininet-WiFi as Mininet.

Omnet++. Omnet++ with the INET plugin is the most versatile network simulation tool currently available to simulate modern network technologies [30,31]. To describe simulation scenarios, Omnet++ uses its own Domain Specific Language (DSL) called **Network Description** (NED). Plugins such as SimuLTE [32] are available for modelling LTE and LTE-Advanced networks. In terms of available SDN extensions, the OpenFlow 1.0 plugin was developed in [21] with performance analysis by Banjar et al. [8]. In 2015, the OpenFlow 1.3 update was introduced into Omnet++ [29] but no evaluation has been discussed for this update. Omnet++ lacks the richness of physical layer modelling provided by NS3 and furthermore, adding new models to Omnet++ is complex and time-consuming.

An overview of the aforementioned simulation tools' networking stacks are shown in Table 1.

Table 1. An overview of simulation tools' networking stacks

Tool	Technology					
	WAN	WiFi	LTE	5G NR	SDN	External interface
NS3	✓	✓	LENA✓	✗	OFSwitch13 ✓	TapDevice ✓
Mininet	✓	✓	✗	✗	OpenvSwitch ✓	(v)Ethernet ✓
Omnet++	✓	✓	SimuLTE ✓	✗	OpenFlow 1.0 outdated ✗	(v)Ethernet ✓
OAI5G	✗	✗	✓	✓	✗	✗

From Table 1, only NS3 is capable of simulating all required network technologies (except 5G NR), whereas Mininet cannot simulate LTE network technologies and Omnet++ does not support the de facto standard SDN protocol OpenFlow 1.3. As shown, there are no current tools that can simulate 5G heterogeneous networks with multiple access technologies and an SDN core. Our aim in this paper is to develop such a simulation tool.

2.2 Simulating Heterogeneous 5G Networks

There are two possible ways of building a heterogeneous 5G simulation tool. The first is extending one of the existing tools to incorporate the technologies that it lacks and the second is connecting multiple simulation tools with complementary technologies. The first option would be expensive with respect to both development time and maintenance time. For our proposed solution, we therefore choose the second option.

Connecting multiple simulated networks from different simulation packages, however, is not trivial. Despite numerous attempts, we did not succeed in producing a heterogeneous 5G simulations by simply linking NS3 with Omnet++ or Mininet-WiFi. We show in Fig. 2 the issues that were encountered when connecting networks from multiple tools. These include subnet differentiation, the lack of ARP proxy and responders, scalability and flexibility of the interplaying simulations. For example, if Simulator 1 has UEs connecting to Voice Server A which resides in a separate Omnet++ simulation, UEs cannot establish any connection to Server A as there are no ARP responders that help resolve Server A's MAC address. Similarly, if the Omnet++ simulation does not expose a common Ethernet interface sharing the same Ethernet data plane with Simulator 1, UEs from simulator 1 cannot reach Server A. Even if the two above mentioned problems are appropriately addressed, when Simulator 1 UEs trying to connect to NS3 Data Server B, due to different subnets in which Simulator 1's UEs and NS3 Data Server B belong to, they still cannot make connections without further network configuration changes.

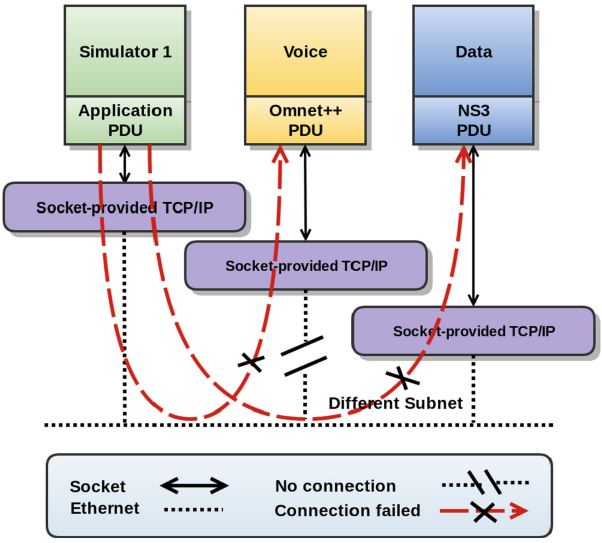


Fig. 2. Issues with connecting multiple simulated networks

2.3 Approaches to Connect Simulations and Networks

Network interoperability is a general issue that is not specific to simulation networks. To bridge multiple simulations in war games, the US Department of Defense (DoD) developed the **High Level Architecture** (HLA) and **Distributed Interactive Simulation** (DIS) framework [6]. In **DIS**, the application protocol is defined as binary formatted Protocol Data Units (PDUs) of

state descriptions such as entity states, environmental processes, signal transmissions. The interconnected networks of simulations use traditional packet-based TCP/IP protocol stacks and only solve the interoperability at the application protocol level.

Several other approaches to inter-connect networks [7,22] use the RESTful API to express discrete and stateful communication between devices. They cannot be used to connect multiple simulations for 5G networks because simulations are connected at transport level whereas Restful APIs are at the session level.

The previously mentioned approaches solve interoperability issues at the application level and are not suitable for network simulations that work at the transport layer or below. Recently, Halba et al. [17] showed that one of the most efficient ways to connect incompatible networks is to use a transport level solution, exploiting the standard TCP/IP stack that is already implemented in most networks. They presented a solution for connecting multiple technologies in vehicle communications by converting CANbus payloads into IP-based packets before connecting to a SDN terminal for further transportation to the destination. By utilizing the SDN paradigm and packet-based conversion of legacy protocols, the automotive applications do not have to deal with tightly coupled vendor-specific interfaces and protocols.

We adopt the approach in [17] of using the latest advances in SDN to connect simulated networks from multiple tools at the transport layer.

3 SDN Adapter for Connecting Simulations

As explained in Sect. 2.2, using SDN to connect simulations at the transport layer is the most efficient method with respect to both time and cost. An overview of

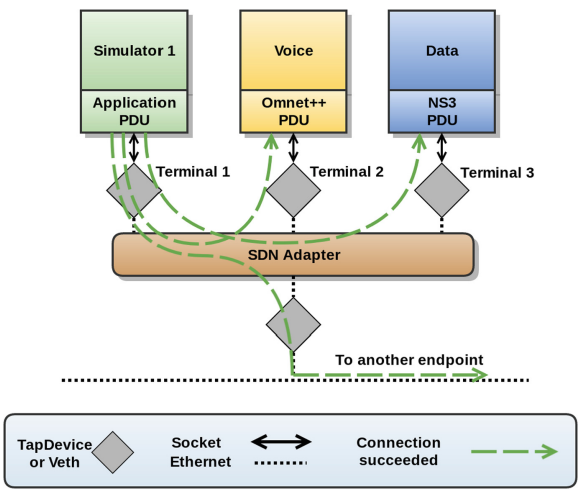


Fig. 3. SDN adapter integration

our SDN adapter solution is shown in Fig. 3. The issues identified in the previous section for connecting multiple simulation networks are resolved by using an SDN Adapter that intelligently forwards packets between simulated networks.

At the heart of our solution is a novel SDN adapter that acts as a bridge, allowing different simulation domains to communicate with each other.

3.1 Building Blocks

Our SDN adapter based solution consists of three key building blocks.

The Wiring Block. The various simulation modules bind to the interfaces at both ends of our SDN adapter to communicate with each other. The OpenFlow Controller will detect traffic and will perform appropriate stateful ARPs and SNAT/PNAT translation transparently and dynamically. For example, NS3 simulations for Wifi, SDN and LTE scenarios cannot be run as a single simulation, but can be separated into 3 isolated simulation processes and bridged together by our proposed SDN adapter. TapDevice works well with NS3 simulations but Mininet-WiFi and Omnet++ require *veth* devices for binding external interfaces as outlined in Fig. 4.

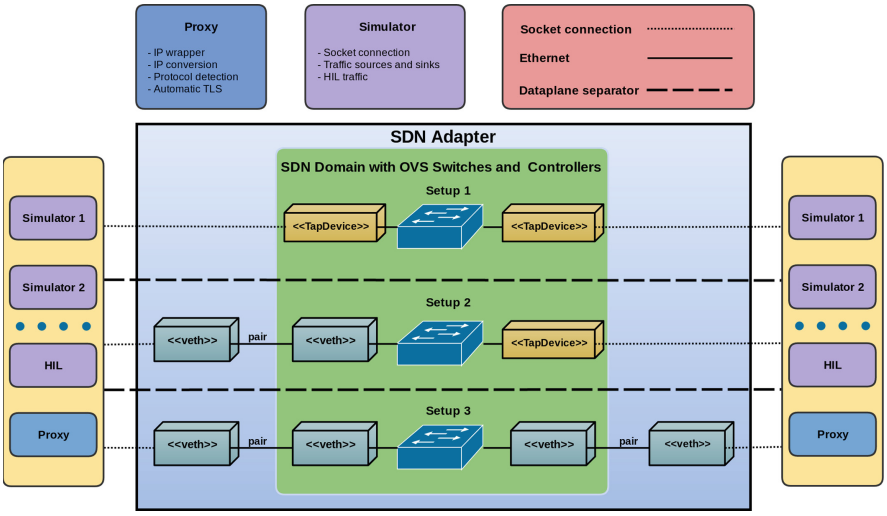


Fig. 4. Adapter architecture

The Controlling Block. OpenFlow controllers play a major role in interconnecting network domains. By providing ARP responders and NAT service, the controllers can actively manage the TCP/UDP states of ingress and egress traffic with OVS flows installation, provided all traffic is IP-based. Moreover, with

the flexibility of SDN technology, the traffic can be easily routed and manipulated according to desired design specifications. The current implementation of OpenFlow controllers is in Ryu [3] with an ARP responder and a simple NAT application. A Ryu-version of the QoS controller mentioned in [9] is also implemented to support the performance evaluation with Mininet-WiFi.

The Proxy Block. SDN by nature is designed to work with IP-based traffic. Therefore, for legacy protocols or application protocols with flexible transport mechanism, an IP wrapper/conversion is needed to encapsulate the payload (e.g. PDUs) into TCP/UDP Ethernet frames before entering the SDN terminal [17]. The **Linkerd** project develops such a proxy for network statistics monitoring and reporting [2]. The proxy is also able to detect network protocol based on the payload header, as well as provide automatic TLS encryption between endpoints suitable for secure orchestration of large simulation scenarios across multiple subnets and public infrastructures. The incorporation of a proxy is however not required for the scenarios studied in this paper.

3.2 SDN Adapter for 5G Simulations

To simulate the desired scenario in Fig. 1, there are three essential components that must be built to support SDN, LTE/5G NR, and our SDN Adapter respectively, as demonstrated in Figs. 5 and 6.

SDN Component. The SDN component is inspired from [9] whereby we reused the NS3 QoS controller scenario and ported the C++ version to the Ryu Python version which is compatible with OpenvSwitch. The performance of the two versions are similar and agree in bandwidth results.

Access Radio Component. The LTE components of Omnet++/SimuLTE and NS3 LENA are different from one another. In NS3, we directly used the **lena-simple-epc.cc** example and modified the remote host in which packets are forwarded to an external TapDevice interface towards the real environment. In SimuLTE, we used **lteCoreExample** as the starting scenario then gradually added up to 8 eNodeBs, with 1 UE per eNodeB as the complete SimuLTE simulation. The router entity in the **lteCoreNetwork.ned** needed extra modification of the configuration to be able to bind to external interfaces, for example, the parameter *numExtInterfaces* must be set to 1. Additionally, a routing table must be changed to match the SDN Adapter veth IP/Subnet as the binding interface for the simulation.

SDN Adapter Component. We have three distinct setups for the SDN adapter with the main variation being its virtual peripheral type (Tap or Veth), and four different simulation combinations to reconstruct the scenario in Fig. 1.

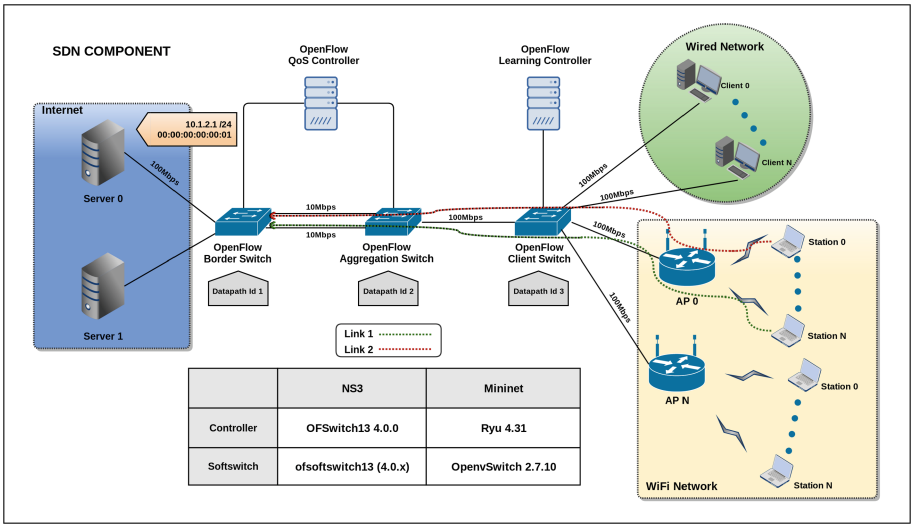


Fig. 5. SDN component

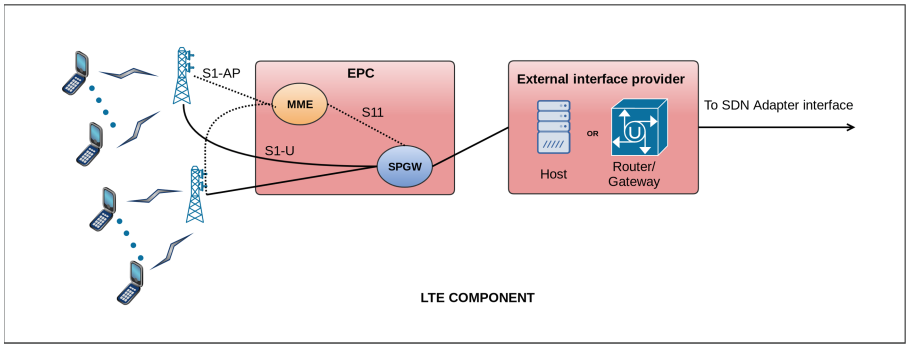


Fig. 6. LTE/5G NR component

3.3 Graphical User Interface for Setting up 5G Simulations

To facilitate easy set-up of 5G simulations with our SDN adapter, we have built a GUI that helps simplify the management of the simulation(s).

The SDN Adapter can be launched in most Linux distributions that support the creation of TUN/TAP devices, veth interfaces and the installation of OpenvSwitch. The process of creating the SDN Adapter is similar to Linux bridges. The SDN Adapter Designer GUI tool, shown in Fig. 7, allows end users to intuitively configure the adapter. This panel is used to design the layout and parameters for the SDN adapter such as IP and MAC addresses, interface type and name, and OpenvSwitch parameters. Once the design process is finished, the end user can start the creation process of the SDN adapter by running the

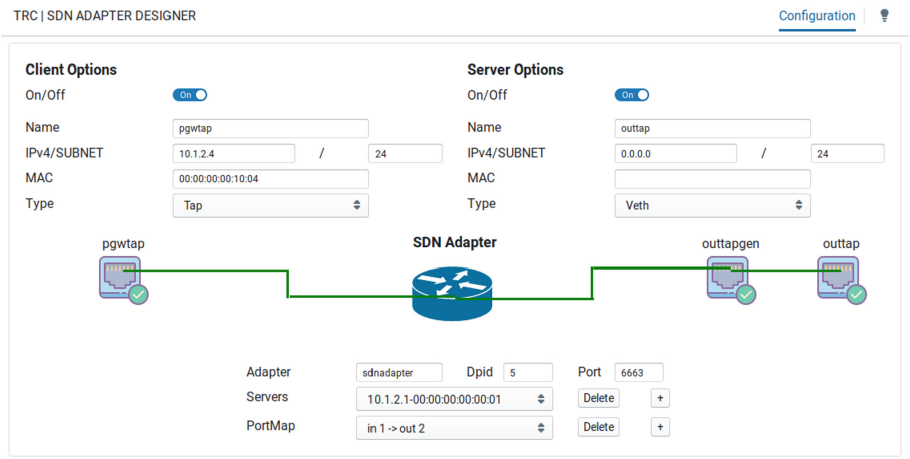


Fig. 7. SDN adapter designer configuration panel

auto-generated bash scripts displayed in the GUI as per Fig. 8. Then the user runs the Ryu controller with auto-generated code to control the SDN adapter.

At this stage, the user is able to start the simulations in arbitrary order. For example, the simulation with SDN servers should bind to the Server port of the SDN adapter by specifying the external interface name in the simulation code, then followed by the initiation of LTE simulation binding to the Client port of the SDN adapter in the same fashion. The whole process can be repeated if the user wants to add/remove ports to/from the SDN adapter for simulation binding modification.

4 Validation and Performance

We show in this section that our SDN adapter based solution can successfully connect multiple simulated networks to produce a 5G simulation with multiple network technologies. In order to simulate the scenario in Fig. 1, we can use the 4 combinations showed in Fig. 9 and those 4 scenarios are evaluated in this paper.

4.1 Measurement Settings

We replicate the methods used in [11] and treat the SDN adapter as a device-under-test (DUT) to measure the performance characteristics of the SDN adapter.

The VM is a VirtualBox image of Ubuntu 18.04.1 LTS with kernel 4.18.0-15-generic, 2 GiB of RAM and 1 CPU running at a constant rate of 3.5 GHz (Intel turbo boost disabled). The veths of the DUT are added into the VM as bridged adapters with driver **Paravirtualized Network** (virtio-net). The promiscuous

```
#!/bin/bash
# a tap device can be create by:
sudo ip tuntap add name pgwtap mode tap
# sudo ip tuntap add name outtap mode tap
# ...
# a veth pair can be create by:
# sudo ip link add pgwtap type veth peer name pgwtapgen
sudo ip link add outtap type veth peer name outtapgen

# OVS switch as SDN Adapter
sudo ovs-vsctl add-br sdnadapter
sudo ovs-vsctl set bridge sdnadapter protocols=OpenFlow13
sudo ovs-vsctl set bridge sdnadapter other-config:datapath-id=5
sudo ovs-vsctl set-controller sdnadapter "tcp:127.0.0.1:6663"
sudo ip link set sdnadapter promisc on
# Add port
sudo ovs-vsctl add-port sdnadapter pgwtap
# sudo ovs-vsctl add-port sdnadapter pgwtapgen
# sudo ovs-vsctl add-port sdnadapter outtap
sudo ovs-vsctl add-port sdnadapter outtapgen

# for example, if the we want the IP/HW MAC of pgwtap to be 10.1.2.4/24
# with mac address 00:00:00:00:10:04
sudo ip addr add 10.1.2.4/24 dev pgwtap
sudo ip link set pgwtap up address 00:00:00:00:10:04 promisc on
# sudo ip link set pgwtapgen up promisc on
# Similarly, for server port
sudo ip addr add 0.0.0.0/24 dev outtap
sudo ip link set outtap up promisc on
sudo ip link set outtapgen up promisc on
```

Fig. 8. Setup script autogen panel

mode must be turned on for both VM's vNIC and veth in order to pass packets freely between the two VMs.

Before the experiment, the Ryu controller needs to be configured with IP - MAC as:

- Client: **10.1.2.5/24 - 00:00:00:00:10:05**
- Server: **10.1.2.1/24 - 00:00:00:00:00:01**

Additionally, the Ethernet interfaces of the client VM and server VM must be configured with the same IP - MAC as shown above.

4.2 SDN Adapters Introduce Minimal Overheads

We show here that our SDN adapter provides an easy way to connect simulated networks without introducing overheads into the network. We cross compare our adapter based solution with a solution that uses a vanilla Linux bridge to connect trivial networks from multiple simulators. The setup is shown in Fig. 10a.

Ping is used to test latency as suggested by [10]. Using Iperf, we also measure the jitter performance (the difference in inter-arrival time of transmitting UDP packets).

The client successfully pings the server with performance shown in Fig. 10b. Using Iperf to measure the jitter level with a standard bandwidth of 10 Mbps and the results are shown in Fig. 10c.

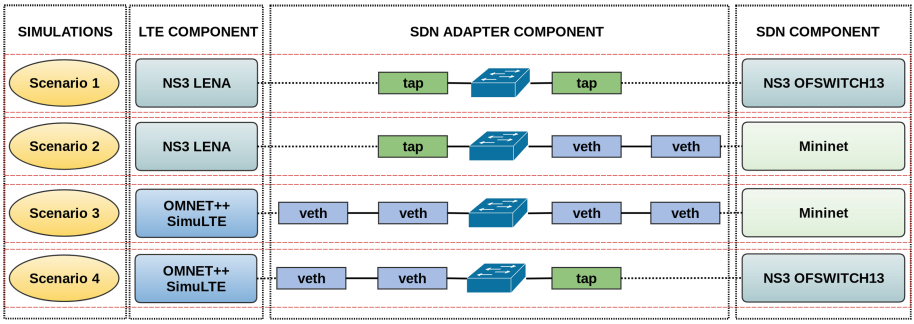


Fig. 9. Simulation combinations and their corresponding SDN adapter setups

The results show similar performance between our SDN adapter DUT and the vanilla Linux bridge DUT, indicating that our SDN adapters do not introduce any additional delay into the networks, which will in turn preserve simulation fidelity. Furthermore, our SDN adapter has a more stable ping RTT than the Linux bridge. On the other hand, the Linux bridge has less jitter than our SDN adapter by an approximate amount of 3 µs. This can be explained by the simplicity of the Linux bridge as the default data path behaviour is just forwarding from one port to the other.

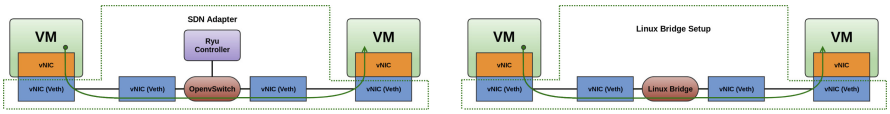
Overall, the ping RTT and jitter overheads are limited to the micro-second scale which is very insignificant with respect to impacting simulation traffic.

4.3 SDN Adapter Integrated Scenarios and Evaluation

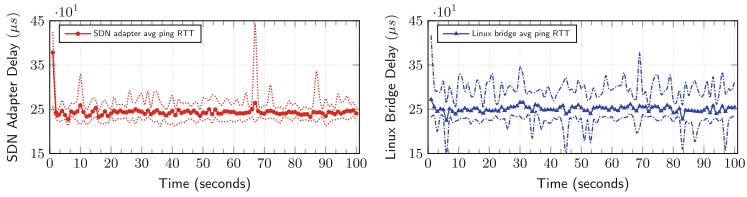
In this section, we show that the SDN Adapter can successfully interconnect different simulations with different technologies. To illustrate this capability, the scenarios will generally include an LTE component (8 & 100 UEs) and an SDN network core with two servers, 3 OVS switches and two OpenFlow Controllers as shown in Fig. 5. For SimuLTE, the LTE component will only incorporate 8 UEs and 8 eNodeBs. We will present a complete 5G network with 5G NR in a later section.

NS3 Lena LTE and NS3 OFSwitch13. Measurement results of the total throughput of both Server 1 and Server 2, with simulation time ranging from 1 to 100 s, are shown in Fig. 11a. The results illustrate that our SDN Adapter successfully enabled the two separate simulations to communicate.

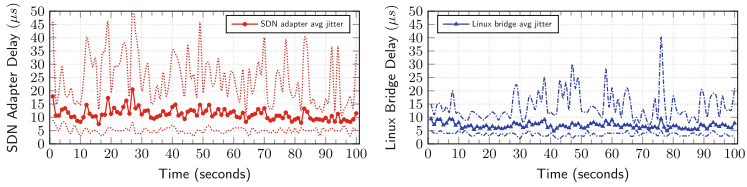
The WAN clients are on the NS3 OFSwitch13 side which are responsible for keeping the simulation running, otherwise it will terminate prematurely due to empty event queues. In Fig. 11a, the value **ns3::LteEnbRrc::SrsPeriodicity** has been increased from the default value of 40 to 160 in order to support 100 connected UEs. The time scheduled for each UE has been reduced significantly due to the mismatch between simulation time and real clock time. This has caused the decline in LTE UDP throughput from 5 Mbps to 1 Mbps.



(a) SDN adapter and Linux Bridge as DUT



(b) Ping



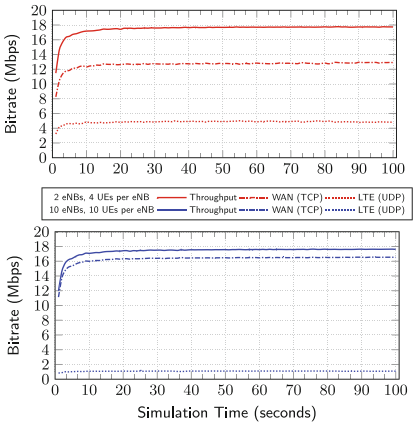
(c) Iperf

Fig. 10. An end-to-end comparison of SDN adapter and Linux bridge performance

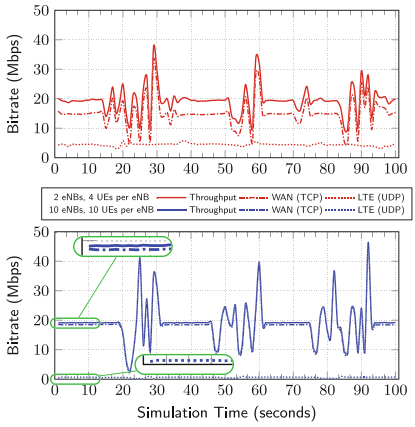
NS3 Lena LTE and Mininet. The performance results of this combination are shown in Fig. 11b. The results agree with those of Scenario 1 as the throughputs are very similar. Overall, the SDN Adapter has shown its ability to bridge NS3 LTE and Mininet SDN to form a complete SDN-enabled LTE network simulation. The simulation also highlights an issue with Mininet-WiFi that has not been reported before: for every 20 s, the measurement of the Mininet Iperf server throughput experiences unstable values. This issue appears more frequently with shorter report intervals, such as 1 s.

We perform further analyses of the Mininet-WiFi behaviors and summarize the findings below:

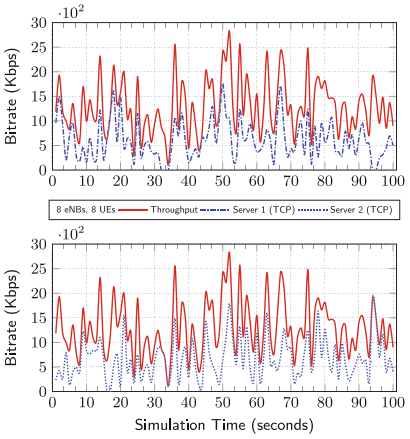
- The problem can be reproduced in up-to-date systems with several stable OpenvSwitch versions - we used OpenvSwitch 2.7 and 2.9 in Ubuntu 18.04.
- Network topologies are irrelevant to the cause of the problem as we have tested with different symmetric topologies ranging from 2 hosts - 1 switch, 4 hosts - 1 switch, and 4 hosts - 2 switches.
- OpenFlow Controllers have no contributing effects to this problem, as once the necessary flows get installed into the OpenvSwitch, it is not involved in packet processing logic.



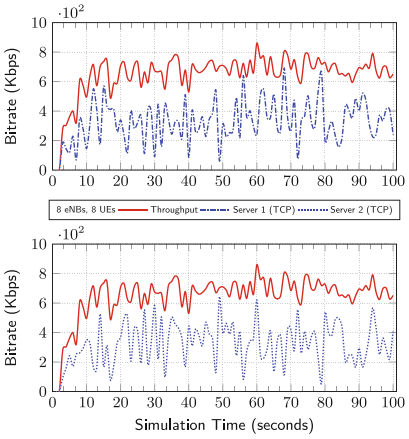
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3



(d) Scenario 4

Fig. 11. Server bandwidth

- Increasing the connection bandwidth does reduce the variation e.g. 10 Mbps bandwidth will have larger bandwidth reading variations compared to 100 Mbps.

We believe this is a local issue of Mininet - a potential bug. From previous experiments, Mininet performance is on par or even better than NS3 OFSwitch13 with respect to some measures, but this measurement lag problem has introduced noise into our throughput results.

Omnet++ SimuLTE and Mininet. The performance results of this combination are shown in Fig. 11c. SimuLTE is inherently built on top of an INET

extension of Omnet++ to provide LTE and LTE-Advance simulations. With the proven SDN stack in Mininet SDN, this scenario has the benefit of quickly introducing SDN into an Omnet++ simulation. The Omnet++ simulation needs to be run in **express mode**. Some other notable configurations are that the scheduler-class is real time (inet::cSocketRTSScheduler), the ppp queue frame capacity must be large enough (10000–100000 packets), and the external interface should be setup according to an **emulation** example.

Omnet++ SimuLTE and NS3 OFSwitch13. The results are shown in Fig. 11d. The two simulations can communicate with stable output results. Compared to Mininet SDN, however, the connection speed is slower and the throughput never increases more than 1 Mbps. Other measures such as packet drop and delay are insignificant as the reliable and bottleneck-free characteristics of the SDN adapter has been validated in Sect. 4.2.

4.4 Comparisons with Alternative Solutions

As there are no current solutions for connecting simulated networks, we are not so much comparing existing solutions but rather envisaging alternative approaches that could be taken by the end user. In particular we consider three alternative methods:

1. One or more routers (Cisco enterprises, small household routers)
2. Dedicated Linux hosts with manual configuration of local firewall (e.g. iptables)
3. Hand-crafted scripts/programs as middleware between simulators

As listed above, the physical routers provide NAT and ARP services for connecting simulators running on different hosts. Dedicated traditional computers with Linux installed can be used to replay routers' NAT and ARP services simply by using iptables with one NAT rule per subnet. Lastly, there is always a way for experienced users with highly-skilled computer literacy to manually create middleware-level software which replicates NAT and ARP services between simulators.

For comparison, our SDN Adapter can automatically resolve subnet incompatibility with only 2 flows per translation. With the embedded proactive ARP responders, it is able to activate "half connections" regardless of the starting order of simulations. A quick comparison can be also quantitatively estimated in Table 2. The $ sign is used to measure the scale of cost implementing the corresponding method.

In terms of cost, in both software and hardware required for connecting simulations, routers and dedicated Linux hosts are more expensive compared to hardware-less open source-based handcrafted scripts and our SDN adapter. Furthermore, enterprise-grade routers (such as Cisco) are more expensive than general computers running Linux. Most routers have embedded Web-based GUI to aid the setup and management which may result in shorter time cost. However,

Table 2. Comparison on alternative solutions to SDN Adapter

Solution	Measure				
	Cost	Time cost	Skill required	Scalability	Flexibility
Method #1	$$$	1 h to days	Intermediate	Depends*	Low - Physical wiring
Method #2	$$	1 h to weeks	Intermediate	Low	Low - Physical wiring
Method #3	0	1 week to months	Advanced	Depends*	High - Virtual wiring
SDN Adapter	0	Less than 1 h	Beginner	High	High - Virtual wiring

*: Expensive and enterprise-grade routers have great scalability by design, while small household routers usually support less than 20 connected devices. Hand-crafted scripts may adjust to the level of scalability needed by the programmer, but usually have low scalability.

our SDN adapters also have an accompanying GUI called SDN Adapter Designer tool which significantly reduces the setup time to less than 1 h. To implement simulation connectivity using methods from 1 to 3, users need to have experience and at least an intermediate level of computer-based skills and literacy (programming, configuration, troubleshooting, etc). On the other hand, to use our SDN adapter, users are only required to have beginner skills in computer literacy with the ability to convert simulation specification into SDN adapter configuration and run the provided auto-generated codes. As already mentioned, for scalability and flexibility measures, users can freely spawn as many as SDN adapters they require, since our SDN adapter is hardware-independent and virtually controlled by software.

5 Complete 5G Simulations

We began our work with a requirement of simulating a heterogeneous 5G network. As the standalone 5G standard is not coming until mid 2020, we build our 5G network based on the non-standalone definition in [5] where 5G networks are comprised of multiple access technologies, 5G New Radio, LTE (e-gNodeB), virtualized and cloud-based solutions and a network slicing paradigm. Figure 12 shows the scenario of our complete 5G simulation implementation. In this figure, the simulated 5G network slice consists of sub-networks from different tools. OAI5G provides the flexible fronthaul, with simulated UEs (up to 256 UEs with expansion build) and one nFAPI gNodeB (Release 14–15). Note that OAI5G has recently integrated the open-nFAPI [1] introduced by Cisco and the specification can be found at the Small Cell Forum [4]. The NextEPC allows a fully featured EPC which is 100% virtualized. Moreover, the SDN adapter will bridge the NextEPC and Mininet-WiFi to incorporate an SDN backend into the slice. nFAPI enables Remote Radio Unit (e.g. a PNF entity) sharing between multiple vendors, allowing cost saving and increasing resource utilization.

5.1 Setting up the Simulations

OpenAirInterface5G. OAI5G is compiled to give **lte-softmodem.Rel14** and **lte-uesoftmodem.Rel14** which are the gNodeB and simulated UEs respectively. The standard configuration targeting nFAPI of the gNodeB and UEs has been listed below.

- Configuration: nFAPI-targeted configuration files
- Mobile Country Code - Mobile Network Code: 208 - 93 (Eurecom France)
- RF Band: 38
- MME address: s1utap's address or eth0 of the NextEPC VM
- UE profile: reused UE 0 (IMSI, SIM KEY, Operator Key)
- 5G NR link: through the oip[0-255] interfaces
- Note: **ue_ip.ko** kernel module must be loaded if NAS_UE_USE_TUN=0

The UE performance is tested using Iperf through the oip[n] interface (n = 0 to 255 in expansion mode, otherwise 0 to 15 in standard mode) exposing as the 5G NR link.

NextEPC. A VM running Ubuntu 18.04 is used to run NextEPC. The installation is through the Ubuntu package manager for NextEPC (HSS, SGW, PGW, MME, PCRF) and its Web GUI application which is used for adding new UE profile into the HSS's database. Following the Web UI and OAI5G UE configuration, we add the UE 0 profile into the HSS database. Without this step, the

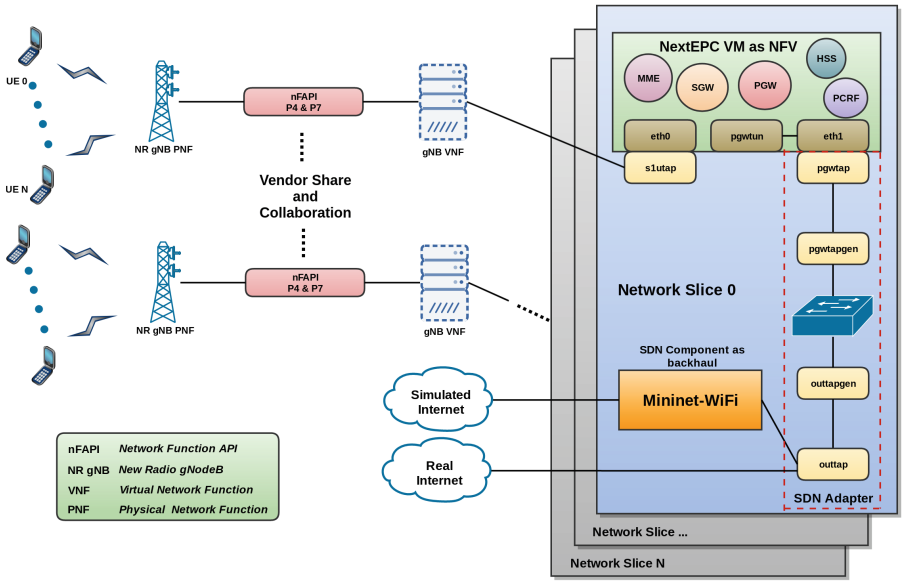


Fig. 12. A 5G network slice with OAI5G, NextEPC, SDN Adapter and Mininet-WiFi

simulated UE will be rejected during the attachment process. Lastly, the NextEPC VM must be in forward mode (ip_forward is enabled). The NAT setup using iptables is not required since the SDN adapter already provides such functionality.

Mininet-WiFi. A Mininet-WiFi component is reused from the scenario shown in Fig. 5 as there is no additional modification required.

SDN Adapter. The SDN adapter setup is straightforward using our provided SDN Designer tool. The only additional step required is to switch the Client Port interface type to veth and then the end user can start running the bash scripts and the Ryu controller.

5.2 Performance

All of these components have been run on a commodity PC with 4 cores, 3.6 GHz CPU speed and 24 GB RAM. For this network, the UEs require 10.41 GB of RAM and the gNodeB requires 587.5MB. CPU loads are low, averaging 21% and 12.9% during Iperf and UDP tests respectively.

In Fig. 13 we present the throughput achieved by the UEs. As shown, most UEs have the stable throughput of 1 Mbps per UE in the simulation. For real UEs (e.g. mobile phones) with a software-defined radio-powered gNodeB, the throughput will be achieved with a higher rate than 1 Mbps.

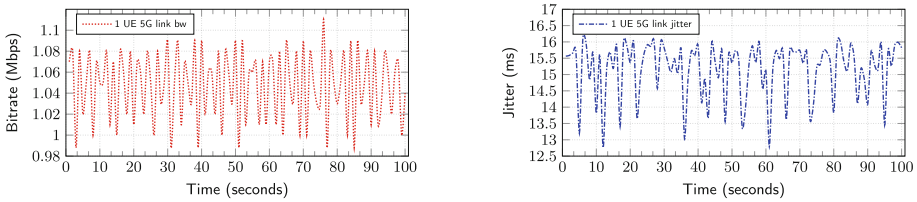


Fig. 13. Performance characteristics of a 5G network slice

6 Conclusion

5G non-standalone networks are currently being deployed worldwide, with standalone variants coming in 2020. Unfortunately, there is no current simulation tool that can simulate a complete 5G network with multiple radio access technologies and an SDN core. In this paper we have developed, for the first time, a 5G simulator that can be used to test all of these technologies simultaneously. Our solution connects multiple networks, provided by existing simulation tools, through a novel SDN adapter. We show that our adapter can connect different flavours of simulators and separate physical domains in a modern 5G network.

Our simulator can be run on commodity hardware and is easy to set up. We will make the simulator and all code used in this paper publicly available. Our SDN adapter also has potential beyond 5G simulations, for example, it can be used to connect Internet of Things (IoT) networks with different technologies. We intend to investigate these applications in future work.

References

1. GitHub - cisco/open-nFAPI: An open source implementation of the Small Cell Forum's Network Functional API (nFAPI). https://github.com/cisco/open-nFAPI
2. Linkerd. https://linkerd.io
3. Ryu SDN Framework. https://osrg.github.io/ryu/
4. Small Cell Forum Releases. http://scf.io/en/documents/082_-_nFAPI_and_FAPI_specifications.php
5. What is 5G? https://www.cisco.com/c/en/us/solutions/what-is-5g.html
6. IEEE Standard for Distributed Interactive Simulation-Application Protocols. IEEE STD 1278.1-2012 (Revision of IEEE STD 1278.1-1995), pp. 1–747, December 2012. https://doi.org/10.1109/IEEESTD.2012.6387564
7. Alaya, M.B., Banouar, Y., Monteil, T., Chassot, C., Drira, K.: OM2M: extensible ETSI-compliant M2M service platform with self-configuration capability. Procedia Comput. Sci. **32**, 1079–1086 (2014). https://doi.org/10.1016/j.procs.2014.05.536. http://www.sciencedirect.com/science/article/pii/S1877050914007364
8. Banjar, A., Pupatwibul, P., Braun, R., Moulton, B.: Analysing the performance of the OpenFlow standard for software-defined networking using the OMNeT++; network simulator. In: 2014 Asia-Pacific Conference on Computer Aided System Engineering (APCASE), pp. 31–37, February 2014
9. Chaves, L.J., Garcia, I.C., Madeira, E.R.M.: OFSwitch13: enhancing Ns-3 with OpenFlow 1.3 support. In: Proceedings of the Workshop on Ns-3, WNS3 2016, pp. 33–40. ACM, New York (2016)
10. Emmerich, P., Gallenmüller, S., Raumer, D., Wohlfart, F., Carle, G.: MoonGen: a scriptable high-speed packet generator. In: Proceedings of the 2015 ACM Conference on Internet Measurement Conference - IMC 2015, pp. 275–287 (2015). https://doi.org/10.1145/2815675.2815692. arXiv:1410.3322
11. Emmerich, P., Raumer, D., Gallenmüller, S., Wohlfart, F., Carle, G.: Throughput and latency of virtual switching with open vSwitch: a quantitative analysis. J. Netw. Syst. Manag. **26**(2), 314–338 (2018). https://doi.org/10.1007/s10922-017-9417-0
12. Fontes, R.R., Afzal, S., Brito, S.H.B., Santos, M.A.S., Rothenberg, C.E.: Mininet-WiFi: emulating software-defined wireless networks. In: 2015 11th International Conference on Network and Service Management (CNSM), pp. 384–389, November 2015
13. Fontes, R.D.R., Rothenberg, C.E.: Mininet-WiFi: a platform for hybrid physical-virtual software-defined wireless networking research. In: Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference, SIGCOMM 2016, pp. 607–608. ACM, New York (2016)
14. Gomez-Miguelez, I., Garcia-Saavedra, A., Sutton, P.D., Serrano, P., Cano, C., Leith, D.J.: srsLTE: an open-source platform for LTE evolution and experimentation. arXiv:1602.04629 [cs], February 2016

15. Gupta, R.: Real-Time LTE Testbed using ns-3 and LabVIEW for SDN in CROWD. Span (2015)
16. Gupta, R., et al.: LabVIEW based Platform for prototyping dense LTE Networks (2014)
17. Halba, K., Mahmoudi, C.: In-vehicle software defined networking: an enabler for data interoperability. In: Proceedings of the 2nd International Conference on Information System and Data Mining, Lakeland, FL, USA, ICISDM 2018, pp. 93–97. ACM, New York (2018). https://doi.org/10.1145/3206098.3206105
18. Imran, M., Said, A.M., Hasbullah, H.: A survey of simulators, emulators and testbeds for wireless sensor networks. In: 2010 International Symposium on Information Technology, vol. 2, pp. 897–902. IEEE, June 2010
19. Kaltenberger, F., Knopp, R., Nikaein, N., Nussbaum, D., Gauthier, L., Bonnet, C.: OpenAirInterface: open-source software radio solution for 5G. In: European Conference on Networks and Communications (EUCNC), Paris, France (2015)
20. Khana, A.U.R., Bilalb, S.M., Othmana, M.: A Performance Comparison of Network Simulators for Wireless Networks. arXiv:1307.4129 [cs], July 2013. arXiv:1307.4129
21. Klein, D., Jarschel, M.: An OpenFlow extension for the OMNeT++ INET framework. In: Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques, SimuTools 2013, Cannes, France, pp. 322–329. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium (2013). http://dl.acm.org/citation.cfm?id=2512734.2512780
22. Kotstein, S., Decker, C.: Reinforcement learning for IoT interoperability. In: 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), pp. 11–18, March 2019. https://doi.org/10.1109/ICSA-C.2019.00010
23. Kumar, A., Kaushik, S.K., Sharma, R., Raj, P.: Simulators for wireless networks: a comparative study. In: 2012 International Conference on Computing Sciences, pp. 338–342, September 2012
24. Lin, B.S.P., Lin, F.J., Tung, L.P.: The roles of 5G mobile broadband in the development of IoT, big data, cloud and SDN. Commun. Netw. **8**(1), 9–21 (2016). https://doi.org/10.4236/cn.2016.81002. http://www.scirp.org/Journal/Paperabs.aspx?paperid=63807
25. Maksymyuk, T., Dumych, S., Brych, M., Satria, D., Jo, M.: An IoT based monitoring framework for software defined 5G mobile networks. In: Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, IMCOM 2017, Beppu, Japan, pp. 105:1–105:4. ACM, New York (2017). https://doi.org/10.1145/3022227.3022331
26. Pham, T.: pthien92/sdn-adapter-designer-react-typescript, November 2019. https://github.com/pthien92/sdn-adapter-designer-react-typescript. Original-date: 2019-11-01T11:36:03Z
27. Pham, T.: pthien92/simulations-scripts, November 2019. https://github.com/pthien92/simulations-scripts. Original-date: 2019-11-01T12:15:51Z
28. Riley, G.F., Henderson, T.R.: The ns-3 network simulator. In: Wehrle, K., Güneş, M., Gross, J. (eds.) Modeling and Tools for Network Simulation, pp. 15–34. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12331-3_2
29. Salih, M.A., Cosmas, J., Zhang, Y.: OpenFlow 1.3 extension for OMNeT++. In: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, pp. 1632–1637, October 2015. https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.246

30. Varga, A.: INET Framework for the OMNeT++ Discrete Event Simulator (2012)
31. Varga, A., et al.: The OMNeT++ discrete event simulation system. In: Proceedings of the European Simulation Multiconference (ESM 2001), vol. 9, p. 65 (2001)
32. Virdis, A., Stea, G., Nardini, G.: Simulating LTE/LTE-advanced networks with SimuLTE. In: Obaidat, M.S., Ören, T., Kacprzyk, J., Filipe, J. (eds.) Simulation and Modeling Methodologies, Technologies and Applications. AISC, vol. 402, pp. 83–105. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26470-7_5