BlueArch–An Implementation of 5G Testbed

Saptarshi Ghosh, Emeka Ugwuanyi, Tasos Dagiuklas, and Muddesar Iqbal
Division of Computer Science & Informatics, School of Engineering London South Bank University, London, SE1 0AA, UK
Email: ghoshs4@lsbu.ac.uk; ugwuanyie@lsbu.ac.uk; tdagiuklas@lsbu.ac.uk; m.iqbal@lsbu.ac.uk

Abstract —In this paper, we've proposed BlueArch, a testbed for 5G research and experimentations. It is a customized setup, that comprises of several opensource components. It provides a platform to create and customize virtual network infrastructures and benchmarks prototypes. BlueArch provides high flexibility in terms of customization, configuration, and programmability. It supports 5G features such as softwarisation, virtualization, and orchestration etc. Furthermore, it offers use cases like IoT, MEC, and SDN through various modes such as simulation, emulation, access to physical network and interfacing with other testbeds. The goal of this paper is to present the structural and functional building blocks of BlueArch, along with their organization and implementation. Finally, three uses cases are also given with results, to demonstrate the functionalities.

Index Terms—5G Testbed, Interoperability, SDN

I. Introduction

The fifth-generation mobile network (5G) era is about to begin. With massive anticipation, it is getting ready to hit the commercial market in 2020[1]. The underlying philosophy is to make a mobile network around people and things [2], that will satisfy the following use cases.

- Extreme Mobile Broadband (eMBB) for an on-demand gigabit connection
- Massive Machine Type Communication (mMTC) to connect a scalable sensor network
- Ultra Reliable Low Latency Communication (URLLC) for having real-time tactile internet.

In order to achieve these verticals, the European Union funded 5G public-private Partnership (5GPPP) working group has proposed certain architectural enablers such as Millimeter wave, Massive MIMO, Visible Light Communication (VLC), Heterogeneous Networks (Het-Net), Internet od Things (IoT) etc. Also, Softwarization [3]–[5] i.e. programming a virtualized network infrastructure using software has brought the communication engineering and software engineering into the same page. The network has eventually become smarter, robust and manageable. Programming a network in abstraction has given flexibility, unprecedented to the previous generation. Technologies such as software-defined networking (SDN), Network function virtualization (NFV), Management Orchestration (MANO), Machine Learning (ML), Virtualization and Cloud-Native plays play a key role to enable those verticals. Finally, Self-organized networks (SON) and Network Slicing has given the ability to manage and work with complex heterogeneous networks.

For the research community, 5G has been in the limelight for a while. Due to its diverse field application, it has brought researchers together from several domains and disciplines. One fundamental practice in such interdisciplinary research is to craft an artificial 5G test environment to implement, verify and validate concept before leading into prototyping. Therefore, a testbed is an essential tool for any 5G oriented research that comprises of all the enabling technologies such as SDN, ML, and Virtualization etc. and let the researcher test the concept in form of algorithms without bothering too much about the system implementation and operational details. Undoubtedly, building such a Testbed is a key step, however, there are three possible alternative forms in which they generally come.

Fully Simulated: This type of environment are the lightest alternatives. Mathematical models (such as delay, mobility and queuing models etc.) to calculate network parameters while simulating. A detailed comparison of such simulators like NS2, QualNet, OPNET, TOSSIM is presented by authors in [6]– [8]. These environments are limited to interfacing with external. Emulated: Network Emulators are typically UNIX based, using native drivers such as HWSim, these platforms can interface with physical systems and can produce more realistic results. Common Open Research Emulator (CORE) [9] is one of the classic examples of such an environment. Mininet is presently one of the most popular network emulators for SDN [10], [11]. Also, Mininet-Wi-Fi [12] an extension of standard Mininet is capable of emulating wireless networks with several mobilities and propagation models. Graphical Network Simulator (GNS3) is an option for advanced implementation, it provides virtualization, Docker containerization, and appliance support. One can use GNS3 to mimic an almost real scenario [13], [14]. These environments are typically heavy and some needs network configuration skills to prepare the test environment.

Hybrid: The Hybrid environment is the most advanced, they practically simulate and emulate the network and provides outstanding interfacing capabilities to the external world and other simulation environments. Netsim [15] is one of the top hybrid platforms used by many universities and corporate houses for both research

Manuscript received February 7, 2019; revised November 7, 2019.
Corresponding author email: ghoshs4@lsbu.ac.uk.
doi:10.12720/jcm.14.12.1110-1118

and production. These platforms are typically commercial hence includes likening.

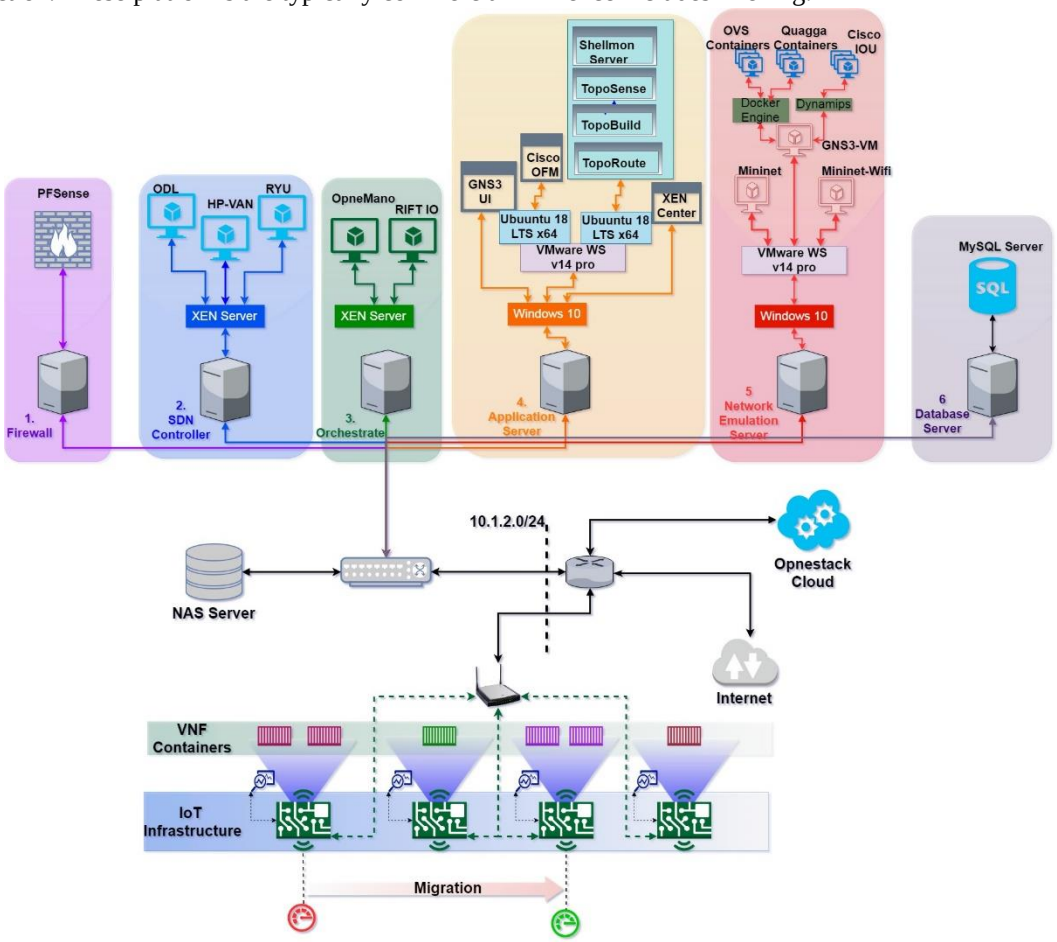

Fig. 1. Schematic diagram of BlueArch

A. Contribution

In this paper, we've presented our 5G Testbed BlueArch, as a part of the SONNET project [16]. It is an organization of several open source tools and supports all the necessary technologies including hybrid architecture, interfacing to external environment etc. This paper is useful as a guide to building such a testbed from scratch.

The rest is organized as, Section II describes the architecture and details of components and their implementation, Section III Demonstrates various use cases and experiments conducted and finally, we conclude and acknowledge at section IV & IV respectively.

II. System Architecture & Implementation

Fig. 1 depicts the schematic diagram of BlueArch. It's a hybrid platform that supports simulation, emulation, interfacing to external platforms and physical environment. It is a collection of six virtual machines (VMs) each runs a specific service, in case of scalability, it is provisioned to add more VMs to accommodate the need. The virtualized service-oriented architecture (v-SOA) provides easy recovery by snapshots and portability by migration. The implementation can be made physical by cloning the VMs into physical servers, this improves overall performance. A NAS server is used as shared storage and this make a private network running at 10.1.2.0/24 address space. A gateway router connects a wireless access point running on same private address space, an external OpenStack private cloud setup, and internet. A Mobile edge computing (MEC) implementation [17] using raspberry pi, is interfacing with the platform. This is a use case of IoT infrastructure with virtual network function (VNF) migration over the test bed. In the following section, BlueArch's features are described in detail.

- *Firewall:* For this implementation PfSense[18] opensource firewall is used. It is a Free BSD based implementation that along with a basic firewall provides services like network monitor, traffic shaper, load balancer, deep packet inspection, and routing. In the virtualized setup the WAN port of PfSense is connected to the bridged external network and the LAN port is connected to the all other VMs. For a physical setup, it must be placed between the gateway router and the private network.
- *SDN Controller*: In an SDN environment the control plane decides and governs the communication. It controls the underlying forwarding devices using the OpenFlow protocol, in this setup the forwarding devices are Open-V-Switch (OVS) [19]. For the

Control plane, there are the controllers are hosted, OpenDaylight (ODL), Ryu and HP- VAN. This supports the implementation of cross-platform, multi-controller infrastructure. The controller is hosted as VMs under a paravirtualized environment hosted by Citrix XEN server (which is an open source type 1 hypervisor like, VMWare ESXi). The paravirtualization is perhaps optional, but it gives better flexibility with cloning and Snapshots.

- *Orchestration:* BlueArch also supports ETSI MANO orchestration, Open Mano and RIFT.io orchestrators are hosted as VMs within a XEN environment. This supports orchestrating Virtual Network Functions (VNF) and leverage network slicing ability [20], [21] for optimal service and resource management. RIFT allows prebuilt VNFs to plug and play over the system, called onboarding.
- Application Server: This acts as the SDN application layer. Although the given implementation runs Windows 10 and VMWare workstation pro, one can use any opensource client operating systems such as Ubuntu 64 bit and XEN server or Virtual box as opensource alternatives. The client OS runs the GNS3 UI, a type 2 hypervisor and XEN center.
- **GNS3 UI** is the graphical user interface of the GNS3 software. This provides a platform to draw and design a network, accessing individual device with CLI though terminal or GUI using VNC is also possible from GNS3 UI. Fig. 2 shows the GNS3 layout of a sample network topology. GNS3 also allows emulated devices to access external network using NAT or Bridged connection. For large scale simulation, GNS3 offers a separate compute platform, typically virtualized, called GNS3 VM. While simulating a network GNS3 UI offload the devices to GNS3 VM via either QEMU virtualization or Docker containers. GNS3 UI also provides a RESTful web API to monitor activities and support Wireshark integration for traffic analysis and Deep packet inspection. GNS3 website provides a wide range of Docker/QEMU based open source appliances, some of the most popular ones are, OVS, Ostinato traffic generator, Ubuntu, Cumulus VX etc.

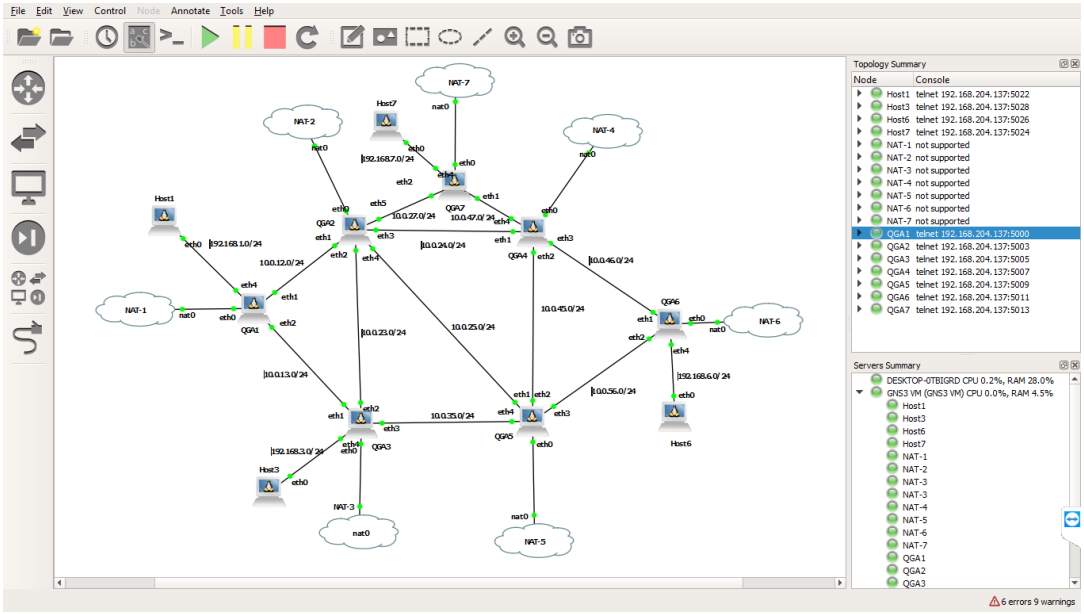

Fig. 2. Network design, A Sample topology in GNS3 UI

- **Type 2 Hypervisor** is used to host custom applications and some optional applications. The main reason for its placement is to isolate the homegrown apps from the rest of the system. The hypervisor is hosting two Ubuntu 64-bit VMs let them be VM1 and VM2. VM1 runs Cisco OpenDaylight Open Flow Manager (OFM) app [22], this is an extension of the ODL controller, using RESTConf protocol it communicates with ODL. The Flow Maker tool of OFM allows the user to easily create OpenFlow rules and manipulate switch wise OF tables. OFM runs its own web Node JS based server and hence the UI can be accessed from anywhere within the private network. VM2 hosts the bespoke applications, most of the prototyping is done here.
- **Custom Tools,** BlueArch presently contains four such prototyped applications, following four applications, written in python3.6 are developed for various projects running on top of BlueArch.
  i) *ShellMon:* This is a client-server-based resource monitor. The client is installed into a Linux system as a push agent, the server fetches system resource utilization. Fig. 3 shows a sample plot of ShellMon Server.
  ii) *TopoBuild:* These app uses are used for interacting with external simulators such as MATLAB or NS3. Data from the remote simulator is first written into the MySQL database server discussed in a later section. Using a MySQL client API *TopoBuild* reads the dataset and RESTful API it conveys to GNS3 or *Mininet* environment. The primary task is

to replicate the topology and network state such as node and channel properties to an SDN platform. *TopoBuild* is event-driven, therefore any change in the remote environment triggers changes in the testbed.

iii) *TopoSense:* This app uses the *OpenDaylight* northbound interface (NBI) to read and right controller information. Using *RESTConf* protocol it fetches the network topology and flow table information respectively from network/topology and node/inventory resources. The topology and flow table information are fused into a Graph Structure to compute various graph-theoretic algorithms such as shortest paths, Spanning Tree etc. Results are written back to the ODL, which the ODL translates into OpenFlow rules and inject into OVSs.

iv) *TopoRoute:* It is one of the subroutines of *TopoSense*, used for calculating routes. It calculates all pair shortest paths from topology generated by *TopoSense* and the set of paths are returned. Fig. 4 shows a sample plot of *TopoRoute* generated all pair shortest path from a 6 nodes topology.

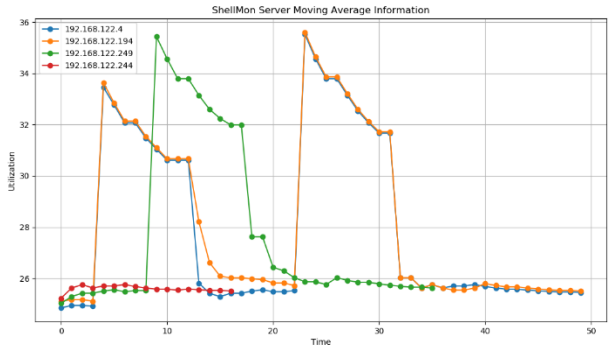

Fig. 3. ShellMon Server plotting live utilization of 4 Raspberry Pi nodes with over a IP network

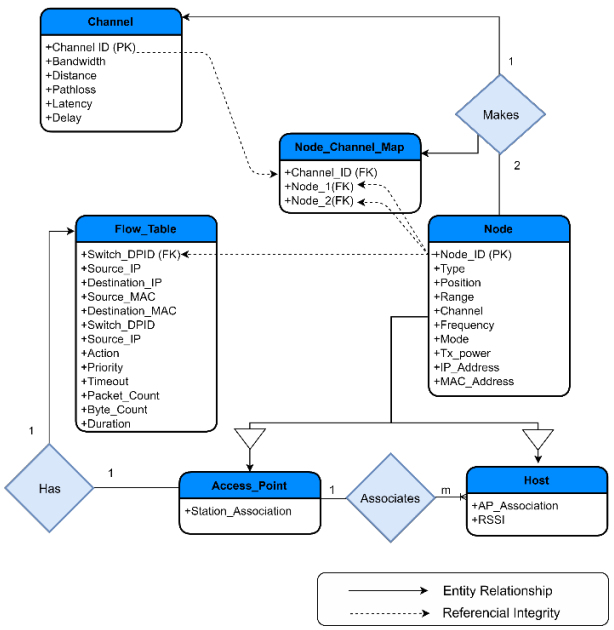

Fig. 4. ERD of a sample database schema to model data exchange between the testbed and a remote simulator

- *Network Emulation Server:* This is the most compute-intensive VM among the others in the testbed. This hosts three emulators Mininet for wired SDN simulation, Mininet Wi-Fi for wireless SDN simulation and GNS3 Compute hosting offloaded computation from GNS3 UI. Emulators are hosted as VMs. GNS3 VM mainly hosts OVS instances and Quagga software routers as Docker containers and Cisco IOU instances as QEMO VMs. For some legacy Cisco images, GNS3 offers an inbuild hypervisor called dynamips which can optimally schedule their resource allocation using *Idle-PC* tool.
- *Database Server:* The database server is running MySQL Server, primarily used as a middleware between the testbed and any external platform. Since SQL is a standard data modeling language, it provides superior compatibility. Fig. 5 shows an example schema to handshake live data from a remote MATLAB based simulator, further discussed in Use cases. Hence, of SQL enhances interoperability. One can use alternatives like Elastic Search for better throughput.

  The modeling of the schema has two primary entities Node and Channel. The node caries information such as node ID, operating frequency etc. whereas, the channel properties are Channel ID, Bandwidth etc. shown in Table I and II respectively.

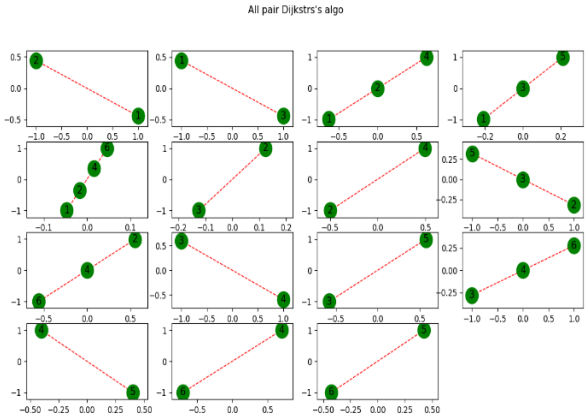

Fig. 5. All pair Shortest paths of a graph with 6 vertices, generated by Dijkstra's single source SPF algorithm

TABLE I: NODE TABLE ATTRIBUTES

Node Attributes	Description
Node ID	Unique ID for each node
Type	Access point or User equipment
Position	(x, y) axis of the node
Range	Range in meter
Channel	Operating channel
Frequency	Operating Frequency in Hz
Mode	The mode of transfer (b/g/n etc.)
Tx_Power	Transmit power in (db)
IP Address	Generated by the emulator
MAC	Generated by the emulator

TABLE II: CHANNEL PROPERTIES

Channel Attributes	Description
Channel ID	Unique ID for each channel
Bandwidth	E2E throughput in Mbps
Distance	Physical distance in meter

Pathloss	Measured from SLS
Latency	Packet processing time in ms
Delay	Round trip time in ms

A node can be a Host/UE or a Switch/AP. If a node is a switch/AP then it also contains its flow table, fetched from the controller. Each AP knows about the list of hosts it is associated to, each host knows the node-ID of its associated AP. Two nodes (one must be a switch) makes a channel that refer to the channel ID, two host nodes can't make a channel. Channel between two switches is a backhaul link and channel between a switch and a host is a fronthaul link.

III. Experiments & Use Cases

In this section, three experiments are described as each having different use cases.

Case 1: **RESCUE** a cloud-based IoT system for Disaster Recovery [23]. The referred paper demonstrates the Monitoring and Load balancing feature. The *ShellMon* client is installed on the IoT gateways and the server remotely monitors the Realtime resource utilization (Fig. 3).

Case 2: Self Migration of Docker Containers, interfacing with the physical network. In this use case, Raspberry Pi is used as a MEC node hosting a VNF as a Docker container. Whenever the MEC gets overloaded, the migration function gets self-triggered and initiates a post-copy migration to another MEC which is infeasible distance and having an adequate free resource to accommodate the immigrant container. Fig. 6 describes the migration.

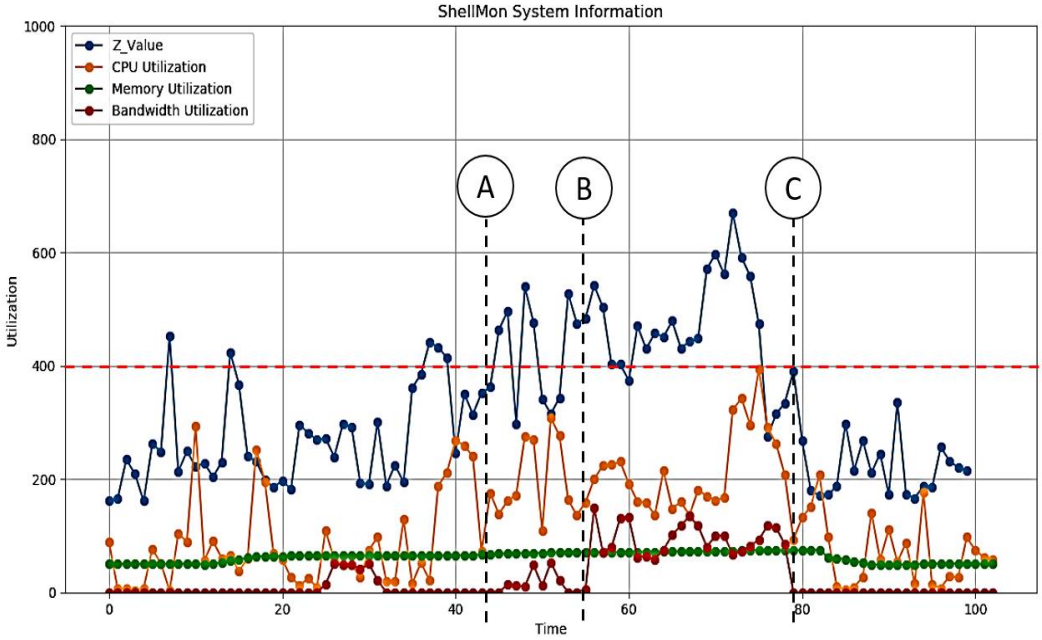

Fig. 6. Self Migration of VNF over MEC. Till mark A the average utilization is under the cut-off 40%, within mark A,B the average exceeds the cutoff, at B migration starts and continues till mark C hence and the utilization comes down.

Unlike Case 1, here *ShellMon* server also runs a learning agent that interprets the varying resource utilization as a time series. The learning agent learns a pattern from the time series and with given cutoff, if it forecasts a failure, it triggers a migration. The triggered migration works as following, we term the overloaded node as Victim and Target be the most feasible node where the victim can offload. The selection of Target is by a proprietary algorithm which is beyond the cope of this article. The algorithm chooses the container (or container chain) from the victim, to be offloaded and the generates a snapshot. Thereafter a secure channel is established between the source and the target and the transfer is initiated. After transfer if the container is still running then a new snapshot is taken sent otherwise the previous snapshot is used. Likewise, the system can dynamically choose between Pre-Copy and Post-Copy migration depending upon the situation.

A major difference between migration offered by docker swarm, the proposed migration also supports *Fall-back* feature, i.e. not only when a system goes off the migration is initiated, but also, when the system spins off again the migrated containers perhaps come back. In addition, the system is immune to *preventive failover*, i.e. the offloading takes place based on the probability of failure, which is learned from the trend of usage.

Fig. 6 depicts the experimental outcome, here the client is deliberately stressed to cause a migration. The X axis is the time window of 100 seconds. The Y axis is utilization, measured in a scale of [0,1000] where 1000 denotes 100% utilization. For experimental purpose, the cutoff value is set to 40% i.e. at 400 mark. The client monitors the CPU, memory and Network interface utilization and boiled them down into a single scaler called Z_Value. The detailed calculation of Z_Value is discussed in our previous paper [24]. The server only

learns from the time series of Z_Value for each client. During timestamp [0,40] there are some spikes that exceeds the cutoff, which the learning algorithm successfully detects as outlier. At the timestamp "A" the system is stressed which causes a hike in CPU utilization, that also hiked the Z_Value. At timestamp "B" the server detects the pattern as a overload and triggers migration, between "B" & "C" network activity can be evident, that is due to the transfer of the container to the Target over the network. At "C" the migration finishes, and the container is shuts off at the victim that causes a sudden fall of the CPU and Z_Value.

Case 3: Interfacing with remote Simulator. In this use case, a remote MATLAB based simulator computes Channel Models of a RAN, and BlueArch provides SDN support on top that by interfacing, Fig. 7 shows the sequence diagram of the handshaking process and Fig. 8 Shows the cumulative Result.

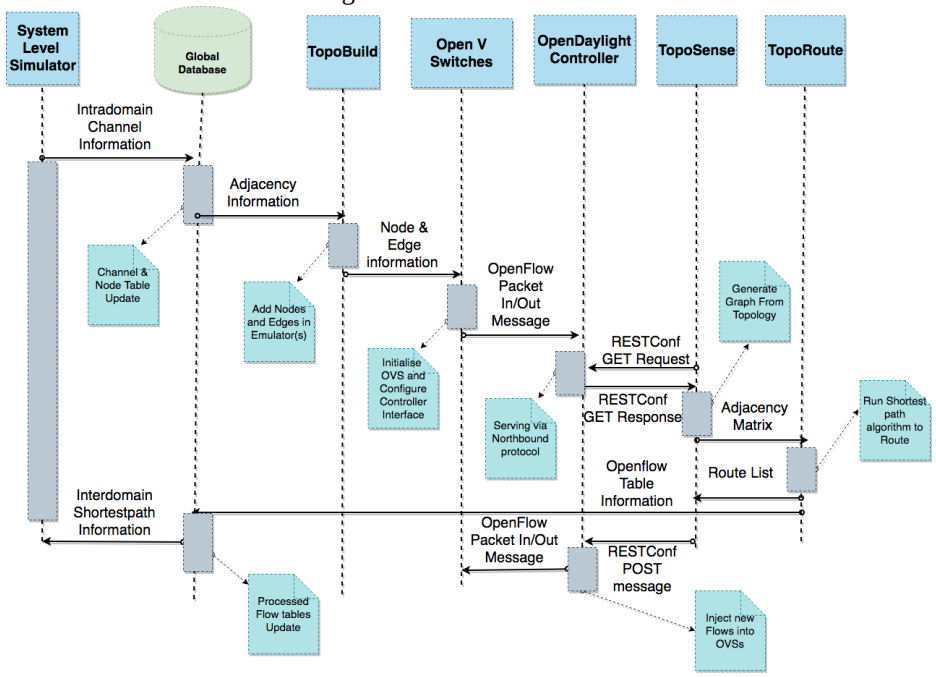

Fig. 7. Sequence diagram of Interfacing and control message exchanges between BlueArch and MATLAB using TopoBuild, OVS, ODL, TopoSense & TopoRoute

The MATLAB based system level simulator (SLS) prepares the channel model and scheduling of radio resources. The network layout is also given. *BlueArch*'s topology module comprising (*TopoSense, TopoBuild* & *TopoRoute*), introduced in section II.D. makes the SDN integration happen. First, based on the network layout, a schema is created on a shared Data Base, containing number of nodes, number of channel and their relevant information. Fig. 4 perhaps used as a reference. The tables are periodically filled by varying channel and node information from the simulator. *TopoBuild* fetches the adjacency information from the database and builds an Open-Flow equivalent using OVS in Mininet and GNS3. The RAN is emulated in Mininet and the Core in GNS3. Since the emulators are bridged to make them communicate. Once *TopoBuild* creates the emulations, it periodically reads the channel updates from the Database and applies on the emulated network. For the time being GNS3 API doesn't support link properties therefore the Core network links don't get altered. *TopoBuild* also adds the SDN controller, which is ODL in this case. ODL establishes OpenFlow communication with the OVSs. *TopoSense* uses RESTConf API to connect to the ODL northbound interface. It fetches the topology and flow table information from the Topology and inventory resources of ODL. Along with that the node & link costs from the database is fed into the Z_Value calculating model. All this information is fused into *TopoSense* to build a much more informed graph we call it *Meta-graph*, the node cost is then relaxed into edges using our Stochastic Temporal Edge Normalization (STEN) algorithm [24].*TopoRoute* then uses this *Meta-Graph* to find shortest path between all pair (Figure 5), and the response is first fed back to ODL using RESTConf which eventually conveys it to the OVSs using OpenFlow, and also the updates are sent to the shared database from which the MATLAB simulator reads it and learns about best routes across the RAN.

Fig. 8 depicts the summery outcome of the subjected use-case. 8.A. shows a schematic diagram of the network layout, running in MATLAB. There are three Base stations (BS1,2 & 3), each connects two user equipment (UE1,2…6) over a radio link which acts as a fronthaul. *TopoBuild* places a OVS for each BS, the OVS performs all layer 2 functions where the Layer 1 functions are performed in the SLS. Therefore, the backhaul is established between the OVSs which acts as a data plane (DP) for the SDN. The Control plane (CP) is provided by ODL and OpenFlow channel is established between the OVSs and ODL. The application server runs all the

custom application an connects to ODL via RESTConf protocol. The emulation is shown at 8.B. where the blue triangle represents the backhaul (Core) and green shades represents the fronthaul (RAN). Each OVS in this emulation is treated as an access points and UE as hosts. 8.C. depicts the topology view from ODL controller. Here all the OVS are discovered and UE's as hosts. ODL doesn't offer a separate view for wireless, thus each connection is shown as a link. 8.D. shows the Meta-graph generated by *TopoSense*, the view is generated using NetworkX and Matplotlib library. Each node in the graph is an object caries node information and flow information, where the edge object caries the link information, discussed in section II.F. the nodes are labeled with their MAC addresses for UE and Datapath ID (DPID) for OVS. The Meta-Graph is fed to the *TopoRoute* to generate pair-wise shortest paths. The link cost changes dynamically as the channel costs changes in MATLAB simulation, which results the change in edge length in the Meta-Graph. The SPF uses the length as an edge weight in order to calculate the shortest path.

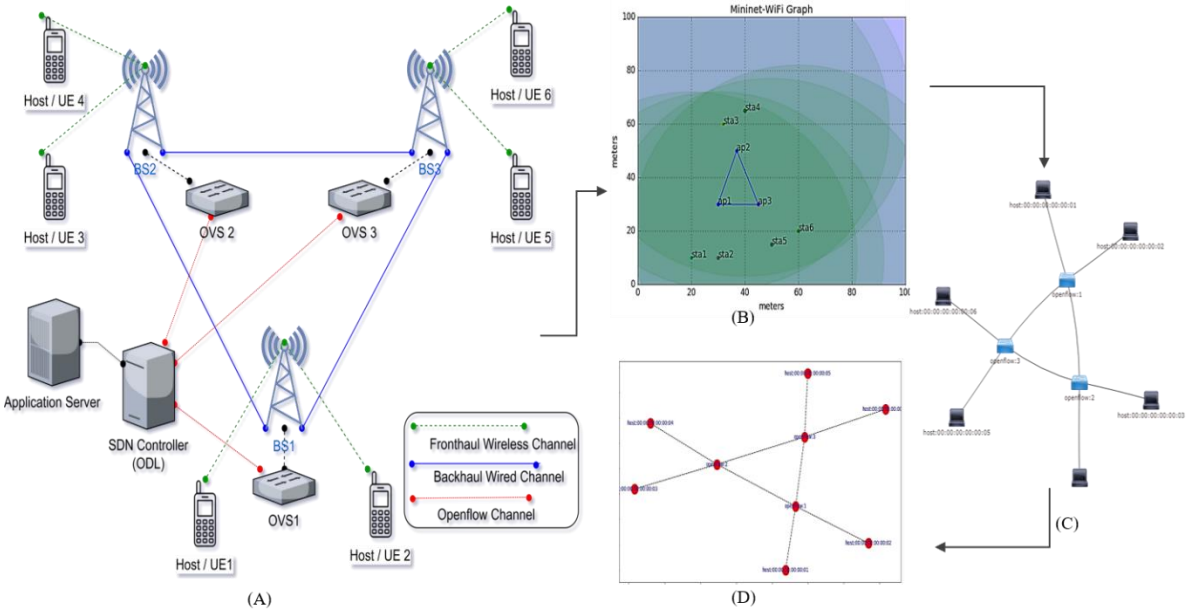

Fig. 8. Complete workflow of BlueArch. On the left the conceptual architecture is presented. (A) TopoBuild build the network in Mininet Wifi, (B) Using OpenFlow ODL reads the network and generate topology, (C) TopoSense, Using RESTConf protocol creates reads the flow table and topology and generate a graph

IV. Conclusion & Future Scopes

BlueArch a 5G testbed implementation that provides a hybrid platform for conducting various experiments including MEC, IoT, SDN, NFV etc. Various mode of tests includes simulation, emulation, and interacting with the physical network and remote testbed platforms. The building blocks of the testbed are Open Source and virtualized which makes the architecture interoperable and flexible. Several use cases are implemented on top of the platform validates its robustness, that includes Realtime resource monitoring, Intelligent service migration and Cross-Platform interfacing.

As a future extension, appending a Recurrent Neural Network (RNN) node for intelligent network automation that includes traffic analysis and route optimization and is planned under progress.

Acknowledgment

This work was supported in part by ''Self-Organization toward reduced cost and energy per bit for future Emerging radio Technologies'' with contract number 734545. The project has received research funding from the H2020- MSCA-RISE-2016 European Framework Program.

References

[1] 5G Infrastructure Association, 5G Pan-European Trials Roadmap Version 3.0, p. 6, 2018.
[2] 5G PPP Architecture Working Group, View on 5G Architecture (Version 1.0), 2016.
[3] 5GPPP, View on 5G Architecture (Version 2. 0), 2017.
[4] 5G PPP SN Working Group, Vision on Software Networks and 5G, 5G-PPP Initiat., vol. 2017, no. 1, pp. 1–38, 2017.
[5] G. S. Network and W. Group, "5G-PPP software network working group from webscale to telco, the cloud native journey," no. 7, pp. 1–25, 2018.
[6] A. Rachedi, S. Lohier, S. Cherrier, and I. Salhi, "Wireless network simulators relevance compared to a real testbed in outdoor and indoor environments," *Int. J. Auton. Adapt. Commun. Syst.*, vol. 5, no. 1, p. 88, 2012.
[7] H. Sundani, H. Li, V. Devabhaktuni, M. Alam, and P. Bhattacharya, "Wireless sensor network simulators a survey and comparisons," *Int. J. Comput. Networks*, vol. 2, no. 2, pp. 249–265, 2010.

[8] A. Stetsko, M. Stehlík, and V. Matyas, "Calibrating and comparing simulators for wireless sensor networks," in *Proc. - 8th IEEE Int. Conf. Mob. Ad-hoc Sens. Syst.*, 2011, pp. 733–738.
[9] J. Ahrenholz, "Comparison of CORE network emulation platforms," in *Proc. - IEEE Mil. Commun. Conf. MILCOM*, 2010, pp. 166–171.
[10] S. Salsano, P. L. Ventre, L. Prete, G. Siracusano, M. Gerola, and E. Salvadori, "OSHI - Open source hybrid IP/SDN networking (and its emulation on mininet and on distributed SDN testbeds)," in *Proc. - 2014 3rd Eur. Work. Software-Defined Networks*, 2014, no. 1, pp. 13–18.
[11] B. Lantz and B. O'Connor, "A mininet-based virtual testbed for distributed SDN development," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 365–366, 2015.
[12] Ramon dos Reis Fontes and C. E. Rothenberg, "Mininet wifi the User manual," 2015.
[13] J. Sendorek, T. Szydlo, and R. Brzoza-Woch, "Software-defined virtual testbed for iot systems," *Wirel. Commun. Mob. Comput.*, vol. 2018, 2018.
[14] K. Yao, W. Sun, M. Alam, and M. Xu, "A real-time testbed for routing network," pp. 256–270, 1999.
[15] Netsim. 'Netsim.' (2011). [Online]. Available: https://www.tetcos.com/index.html
[16] S. Mumtaz, *et al*., "Self-Organization towards reduced cost and energy per bit for future emerging radio technologies - SONNET," in *Proc. IEEE Globecom Work. GC Wkshps*, vol. 2018, pp. 1–6.
[17] E. E. Ugwuanyi, S. Ghosh, M. Iqbal, and T. Dagiuklas, "Reliable resource provisioning using bankers' deadlock avoidance algorithm in MEC for industrial IoT," *Ceskoslov. Gynekol.*, vol. 21, no. 6, pp. 422–423, 1956.
[18] The pfSense Team, *The pfSense Book*, 2017.
[19] Piolink, "Open vSwitch," no. 11, pp. 54–55, 2013.
[20] W. Guan, X. Wen, L. Wang, Z. Lu, and Y. Shen, "A service-oriented deployment policy of end-to-end network slicing based on complex network theory," *IEEE Access*, vol. 6, pp. 19691–19701, 2018.
[21] Y. Minami, A. Taniguchi, T. Kawabata, N. Sakaida, and K. Shimano, "An architecture and implementation of automatic network slicing for microservices," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. Cogn. Manag. a Cyber World*, 2018, pp. 1–4.
[22] J. Medved, *et al*., OpenDaylight OpenFlow Manager (OFM) App.
[23] T. Khan, S. Ghosh, M. Iqbal, G. Ubakanma, and T. Dagiuklas, "RESCUE : A resilient cloud-based iot system for emergency and disaster recovery," in *Proc. IEEE 20th Int. Conf. High Perform. Comput. Commun.*, 2018, pp. 1043–1047.
[24] S. Ghosh, T. Dagiuklas, and M. Iqbal, Energy-Aware IP Routing Over SDN. 2018 IEEE Global Communications Conference (GLOBECOM), 2018.

Saptarshi Ghosh received the B.Sc. degree (Hons.) in computer science from the University of Calcutta, India, in 2010, the M.E. degree in software engineering from Jadavpur University, India, as a GATE Scholar, in 2016, and the M.Sc. degree in smart networks from the University of the west of Scotland, U.K., as an Erasmus Mundus Scholar in 2017. He is currently pursuing the Ph.D. degree in computer science and informatics from London South Bank University, U.K., under SONNET (an MSCA-RISE project). He was a Software Developer with Webel Informatics Ltd., India, with 2 years of experience. His primary research interests lie in the field of applying machine learning in software defined networks for route optimization and self-organization in 5G.

Emeka E. Ugwuanyi received the B.Sc.degree in Computing from London School of Commerce in partnership with Cardiff Metropolitan University, U.K., in 2014, and the M.Sc. degree in internet and database systems from London South Bank University, U.K., in 2016, where he is currently pursuing the Ph.D. degree in computer science. His current research interest lies in resource management in multi-access mobile edge computing (MEC). This includes deadlock avoidance and cache storage management in MEC

Tasos Dagiuklas received the Engineering Degree from the University of Patras-Greece in 1989, the M.Sc. from the University of Manchester, U.K., in 1991, and the Ph.D. degree from the University of Essex-U.K. in 1995, all in Electrical Engineering. He is a leading researcher and expert in the fields of Internet and multimedia technologies for smart cities, ambient assisted living, healthcare, and smart agriculture. He has been a principle investigator, a co-investigator, a project and technical manager, a coordinator, and a focal person of over 20 internationally R&D and capacity training projects with total funding of approximately £5.0m from different international organizations. He is currently the Leader of the SuITE Research Group, London South Bank University, where he also acts as the Head of the Division in Computer Science. His research interests include smart internet technologies, media optimization across heterogeneous networks, QoE, virtual reality, augmented reality, and cloud infrastructures and services

Muddesar Iqbal received the Ph.D. degree from Kingston University in 2010 with a dissertation titled ''Design, development, and implementation of a high-performance wireless mesh network for application in emergency and disaster recovery''. He has been a principal investigator, a co-investigator, a project manager, a coordinator, and a focal person of over 10 internationally teamed research and

development, capacity building and training projects. He is an established researcher and expert in the fields of: mobile cloud computing and openbased networking for applications in education, disaster management, and healthcare; community networks; and smart cities. He is currently a Senior Lecturer in mobile computing with the Division of Computer Science and Informatics, School of Engineering, London South Bank University. His research interests include 5G networking technologies, multimedia cloud computing, mobile edge computing, fog computing, Internet of Things, software-defined networking, network function virtualization, quality of experience, and cloud infrastructures and services. He was a recipient of the EPSRC Doctoral Training Award in 2007