This paper has been accepted for publication on IEEE International Workshop on Communication and Networking for Swarms Robotics (RoboCom) 2022

CloudRAFT: A Cloud-based Framework for Remote Experimentation for Mobile Networks

Sabarish Krishna Moorthy[1], Chencheng Lu[1], Zhangyu Guan[1], Nicholas Mastronarde[1],
George Sklivanitis[2], Dimitris Pados[2], Elizabeth Serena Bentley[3], and Michael Medley[3]
[1]Dept. of Electrical Engineering, University at Buffalo, Buffalo, NY 14260, USA
[2]Dept. of Electrical Engineering and Computer Science, Florida Atlantic University, USA
[3]Air Force Research Laboratory (AFRL), Rome, NY 13440, USA
Email: {sk382, clu27, guan, nmastron}@buffalo.edu, {gsklivanitis, dpados}@fau.edu
{elizabeth.bentley.3, michael.medley}@us.af.mil

***Abstract*—In this article we explore new techniques that can enable open remote experimentation for mobile networks. We first propose a cloud-based framework called CloudRAFT, based on which experimenters are allowed to remotely access and control experimental resources via public cloud AWS and share the resulting data and code via the cloud. Then, we discuss the enabling techniques for CloudRAFT, including Amazon serverless service, VNC-based remote command line, and Websocket-based real time communications, among others. Finally, we showcase the application of these techniques in enabling remote access to UB NeXT, a software-defined testbed that has been developed at University at Buffalo for wireless mobile network modeling, optimization and deployment. This work verifies the feasibility of accessing, controlling and sharing wireless testbeds through a remote public cloud.**

***Index Terms*—Wireless Experimentation, Software-defined Radios, Testbed Sharing, Amazon Web Services (AWS).**

I. Introduction

Over the past decade, significant efforts have been made by the wireless community to build open, shared experimental facilities. A notable effort is the National Science Foundation (NSF) Platforms for Advanced Wireless Research (PAWR) Program [1]. The goal of PAWR is to develop four city-scale shared experimentation platforms for advanced wireless research. As of today, three PAWR platforms have been developed: the Platform for Open Wireless Data-driven Experimental Research (POWDER) [2], the Cloud enhanced Open Software defined Mobile wireless testbed for city-scale deployment (COSMOS) [3], and the Aerial Experimentation Research Platform for Advanced Wireless (AERPAW) [4]. A fourth platform, the Wireless Living Lab for Smart and Connected Rural Communities (aka ARA), was launched recently [5].

ACKNOWLEDGMENT OF SUPPORT AND DISCLAIMER: (a) Contractor acknowledges Government's support in the publication of this paper. This material is based upon work funded by AFRL, under AFRL Contract No. #FA8750-20-1-0501. (b) Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL.

DISTRIBUTION STATEMENT A: Approved for Public Release; distribution unlimited AFRL-2021-3478 on 12 October 2021. Other requests shall be referred to AFRL/RIT 525 Brooks Rd Rome, NY 13441.

While these existing facilities have significantly advanced the experimental research for next-generation wireless systems, it is still challenging to fully meet the needs of experimental wireless research in terms of *diversity*, *generalizability* and *accessibility*. First, existing community experimentation platforms focus on rather specific areas in wireless research (e.g., AERPAW focuses on wireless UAV systems), while there are many research areas in wireless systems that still lack well-designed community shared experimental facilities such as sensing and networking for robotic autonomy, mobility intelligence, underwater and underground communications and networking, among others. Second, existing facilities are deployed in pre-selected environments with nodes installed at fixed locations. For example, PAWR platforms focus on experiments "in-the-wild" with wireless systems deployed in outdoor environments, however it is challenging to test the generalizability of the obtained results to different environments (e.g., indoors) or climates (which may affect significantly the signal propagation behavior, particularly in mmWave and terahertz frequency bands [6], [7]). The same concern also applies to experimental research in data-driven wireless systems, where it is crucial to assure the generalizability (or transferability) of the trained models and the resulting artificial intelligence (AI) and machine learning (ML) algorithms [8]. Moreover, as experimental research becomes more and more important for the wireless community, the needs of experimental resources (e.g., software-defined radios, computing capability, and bandwidth) may exceed what can be offered by existing community facilities in the near future. *These challenges cannot be addressed without a systematic shift in the paradigm of sharing experimentation facilities for mobile networks.*

In this paper we propose a new approach called CloudRAFT for remote experimentation and testbed sharing for wireless mobile networks. At the core of CloudRAFT is to develop an open control plane based on a public cloud for management of the data, code and hardware resources associated with the testbeds. We claim the following two contributions in this paper.

This paper has been accepted for publication on IEEE International Workshop on Communication and Networking for Swarms Robotics (RoboCom) 2022

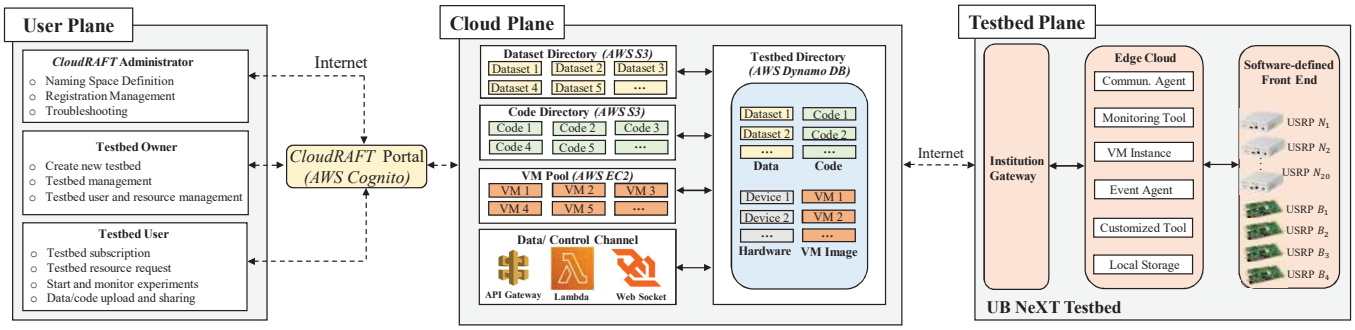


Fig. 1: CloudRAFT: Cloud-based Remote Access for Wireless Testbeds.

- We first propose the CloudRAFT framework, based on which experimenters are allowed to remotely access and control experimental resources via public cloud Amazon Web Service (AWS) [9] and share the resulting data and code via the cloud. The enabling techniques are also discussed, including Amazon serverless service, Virtual Network Computing (VNC)-based remote command line, and Websocket based real time communications.
- We test these enabling techniques by connecting UB NeXT to AWS. UB NeXT is a software-defined testbed that has been developed at the University at Buffalo for wireless network modeling, optimization and deployment. Demonstrations of remote experiments over UB NeXT via AWS are also presented.

II. CloudRAFT Framework

The overall architecture of CloudRAFT is illustrated in Fig. 1, which consists of three planes: *User Plane*, *Cloud Plane*, and *Testbed Plane*.

A. User Plane

CloudRAFT uses this plane to manage three types of users, i.e., the *CloudRAFT Administrator*, the *Testbed Owner*, and *Testbed User*. The CloudRAFT administrator will oversee the operation of CloudRAFT, including user and testbed registration management, name space management for testbed resources, and troubleshooting. Testbed owners can create new profiles for their testbeds, manage the testbed profiles by adding testbed devices (e.g., USRP software-defined radio (SDR) [10]) and configure their parameters (e.g., the USRP type and IP address), approve and manage testbed users, and schedule and oversee the experiments on their testbeds. Lastly, testbed users will be allowed to subscribe to those testbeds connected to CloudRAFT, reserve testbed resources (e.g., computing and SDR resources), conduct experiments and monitor their status and output, and finally share the collected datasets via CloudRAFT. All the users will access CloudRAFT via the Internet using a CloudRAFT web portal deployed on AWS. User authorization and access control for CloudRAFT resources, including the dataset repository, code repository and virtual machine (VM) pool will be powered by AWS Cognito, a user identity, data synchronization and secure management service provided by AWS [11]. In this work we design CloudRAFT based on AWS taking the advantages of AWS's well-developed integrated web hosting, data storage and cloud computing services.

B. Cloud Plane

The primary challenges in testbed sharing are at least two-fold. First, it is complicated and costly for testbed owners to develop and deploy their own dedicated software, hardware and user interfaces for remote access, user authorization, experiment scheduling, and data storage. Second, for researchers, the complexity, inflexibility, and non-uniformity in accessing different testbeds can discourage their use. For example, most testbeds are deployed in subnets and can only be accessed through the corresponding local gateways. To access a testbed, a user needs to contact the testbed owner first, who will then contact the gateway manager to add the user to the allowed-user list. This whole process is time-consuming and repels interested users.

To address these challenges, as illustrated in Fig. 1, the Cloud Plane provides software and tools that can be used for testbed owners to conveniently share their testbeds with the community; hide the complexity in accessing different testbeds by enabling researchers to register a single CloudRAFT account and then subscribe to different testbeds; provides a bridge between researchers and testbed developers; and finally accelerates the formation of a mature ecosystem for experimental research for the wireless community focusing on mobile networks. In CloudRAFT, this can be accomplished by five major modules of the Cloud Plane, i.e., *Testbed Directory*, *Dataset Repository*, *Code Repository*, *Virtual Machine (VM) Pool*, and *Data and Control Channel*.

Testbed Directory. This directory maintains the basic information of the testbeds connected to CloudRAFT and hence increases their visibility to users. For each testbed, a table will be created and managed based on DynamoDB, a fully managed proprietary NoSQL database service provided by AWS [9]. Each table includes general information, such as the testbed name, testbed owner, supported research topics, allowed-user list, among others; and resource information, such as the hardware devices available to the testbed and their parameters, e.g., the type, image version and IP address of the USRP SDRs, the list of datasets and software that are applicable to the testbed, the list of VM images prebuilt for the

testbed, and others. Each table also tracks the corresponding testbed's usage status, the availability of its devices (e.g., online, offline, and occupied) as well as upcoming scheduled experiments. The design of the Testbed Directory is extendable i.e., once a new testbed is created in CloudRAFT by its owner, a blank testbed profile and the associated DynamoDB tables will be created and initialized automatically, which can be further configured by the testbed owner.

Dataset and Code Repositories. In order to store and manage the experimental datasets and code, a dataset and code repository will be designed and hosted on AWS S3 (Simple Storage Service), which provides object storage through a web-service interface. Each dataset (or code) will be associated, through the above discussed testbed directory tables, to one or more testbeds so it will be accessible to the corresponding testbed users. Once an experiment begins, the user will be able to load the selected dataset (or code) either to the VM launched in the *VM Pool* within the *Cloud Plane* through a high-speed connection (25 Gbps), or to the edge cloud in the *Testbed Plane* using a data channel established based on "AWS SDK for Java" (which provides a Java API for AWS infrastructure services including AWS S3, EC2 and DynamoDB). New datasets collected during experiments will be first uploaded to the dataset repository, and later loaded to the VM for annotation and sharing. The *Dataset and Code Repositories* will effectively alleviate the burden of data/code storage, management and sharing for both testbed owners and users.

VM Pool. For each experiment, one or more VMs will be launched using the AWS EC2 (i.e., Elastic Compute Cloud) service. For each testbed, CloudRAFT will provide a set of prebuilt VM images with pre-installed software and libraries as needed for the experiments. Through the launched VM, the experimenter can i) execute *non-time-sensitive* programs in the VM, e.g., routing, transport-layer rate control, motion control, among others; ii) access the reserved edge cloud using `ssh` (or `frp` if the edge cloud is deployed in a subnet) to execute *time-sensitive* programs, e.g., baseband signal processing, scheduling, and power control; and iii) monitor the status and output of the experiments. The VM pool will be interfaced with the dataset/code repository via high-speed in-datacenter connections.

Data and Control Channel. Two types of channels will be established for each experiment, a data channel and a control channel, to bridge the Cloud Plane and the Testbed Plane. The data channel transfers datasets and code from the Dataset/Code Repository to the testbed edge cloud, and vice versa. The control channel sends control commands from Cloud Plane to the testbed. The testbed will then execute the commands on the edge cloud. For example, by sending the command `uhd_find_devices` to the edge cloud (which is connected to front-end SDRs via Ethernet cables), the edge cloud will return information about the currently active and available USRP SDRs. The two channels are designed based on a combination of *AWS Lambda*, *Gateway API* and *WebSocket API* services. AWS Lambda acts as the serverless computing service, runs code in response to events and automatically manages the computing resources required by that code, and *Gateway API* and *WebSocket API* act as the "front door" for applications to access data, business logic, or functionality from backend services, e.g., the predefined Lambda functions.

C. Testbed Plane

The Testbed Plane provides a set of three tools in addition to the custom tools made available by the testbed owners. These include the *Communication Agent*, which serves as a relay between the Cloud Plane and the edge cloud for data transfer; the *Event Agent*, which receives and executes control commands from the cloud plane; and the *Monitoring Tool*, which feeds the real-time status of various computing and SDR devices to the Cloud Plane.

III. Enabling Techniques

In this section we identify several enabling techniques for CloudRAFT, including Amazon's serverless service (ASS), websocket based real time communications, VNC-based remote command line, and fast reverse proxy.

A. Amazon Serverless Service

With ASS [12], [13], the cloud service providers take care of infrastructure management tasks allowing testbed and CloudRAFT developers to focus on the program development. This is very beneficial in the design of the cloud plane of CloudRAFT. The ASS offers a wide array of Application Programming interfaces (APIs) including REpresentational State Transfer (REST) API Gateway, Websocket API Gateway, Lambda Function, DynamoDB, Simple Storage Service (S3), and Cognito, among others.

The REST and Websocket API Gateways act as a firewall that will block, redirect or forward the requests from the users. They also take care of detection and execution of specific parameters in the user requests. The Lambda Function takes care of the execution of custom code defined by the testbed developers based on the parameters in the user requests. The DynamoDB is a key-value and document database that delivers low-latency responses. S3 is a static file storage server and provides easy-to-use management features and also configures the access controls based on the experiment requirements. Finally, Cognito provides a simple user-friendly GUI that allows users to sign up, log in and control their access to CloudRAFT via a webportal.

B. Websocket Based Real Time Communication

The websocket based real time communication is the backbone of the CloudRAFT framework that takes care of the communications between the CloudRAFT control plane and the testbed edge server. These functions can be designed based on the WebSocket protocol. When the user types a command in the webpage, the command will be sent to an AWS Websocket API Gateway, which will then forward the command to the related Lambda function according to the *"action"* parameter in the WebSocket requests. The Lambda function will then

(a) (b) (c)

Fig. 2: (a) Snapshot of USRP SDR deployment, (b) edge servers, and (c) ground robot vehicle.

further forward the command to the edge cloud servers. A Python script will run in the edge server waiting for the messages from the Websocket Gateway. Upon reception of the commands, the script will execute them and return the results to the WebSocket API Gateway, where the results will be returned to the experimenters. The same logic is applicable for the real-time update of testbed resources except that the periodic signalling messages will be transmitted between the edge cloud server and the AWS cloud.

The WebSocket API gateway manages a routing table with different actions and the corresponding call functions. For example, the routing table consists of two special route keys, *"connect"* and *"disconnect"*. The former is used when a user wants to establish a connection to the edge server and the latter will be used for tearing down the connection. The server maintains a data table that contains all active hosts connected to the server. When the client host sends the connect message with `action:$connect`, this request will be forwarded to the `handle_connect` function, which will add the user name and the connection ID of the host to the data table and indicate it as an active user. Similarly, after the experiments, the host can send the disconnect message with `action:$disconnect`, and then the `handle_disconnect` function will be invoked to remove the corresponding entry from the data table. Other messages include `action:$send_message` and `action:$p2p_message` with their corresponding functions `handle_message` and `handle_p2pmessage`. The former handles all the broadcast messages while the latter deals with point-to-point messages.

C. Fast Reverse Proxy

It is important for testbed owners to provide secure connections to their local networks that can be accessed by experimenters from outside of their local networks. This can be achieved by implementing a reverse proxy such as Fast Reverse Proxy (FRP) [14]. FRP acts like a tunnel that can redirect the Internet requests from one IP address to another using TCP or UDP connections. This allows us to hide the testbed address and at the same time preserve the integrity of the testbed information.

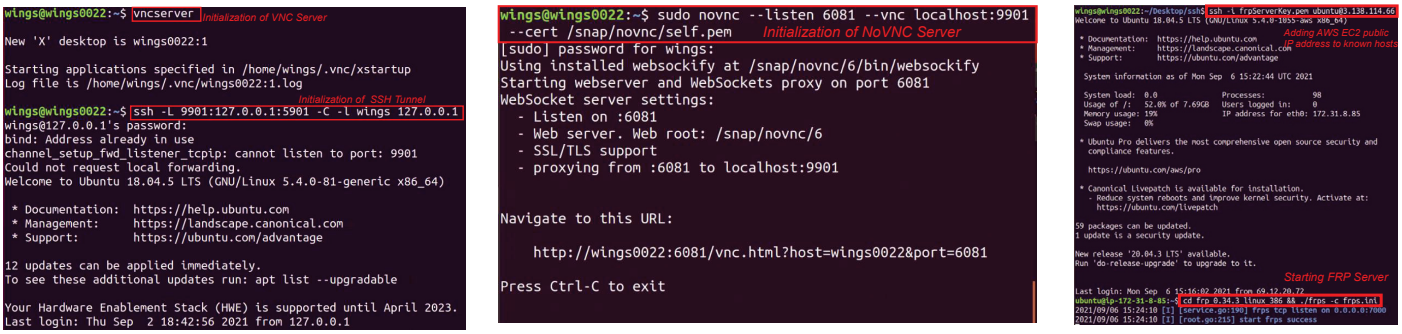
D. VNC-Based Remote Command Line

The remote execution of commands from a unified web portal can be enabled by Virtual Network Computing (VNC), a technique that can achieve desktop sharing with low overhead. VNC is platform-independent and follows a client-server-based architecture allowing multiple clients to connect to one VNC server simultaneously without any compromise in performance. VNC is based on an extensible protocol called Remote Frame Buffer protocol (RFB) [15], which allows connections between different versions of VNC clients and servers. An FRP server can be used to tunnel the connection between the public cloud server and the edge server located at the testbed. Before the VNC server connects to the FRP server, an SSH tunnel is established to enhance the security of the connection.

IV. Prototyping and Demonstrations

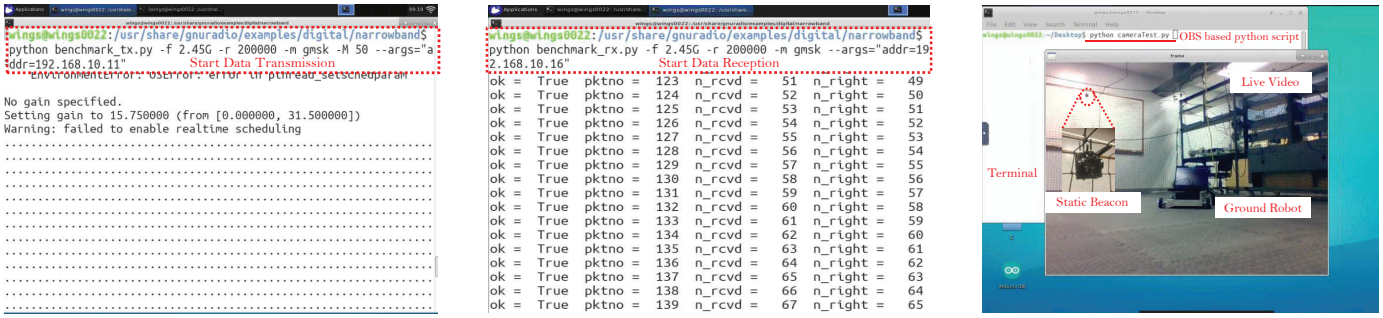
In this section we prototype CloudRAFT by integrating UB NeXT, a software-defined testbed for wireless network modeling, optimization and deployment. Next we first give a brief description of the NeXT testbed and then showcase the remote access to the testbed through CloudRAFT.

A snapshot of the NeXT testbed is shown in Fig. 2, where there are two major components: edge server and front-end SDR. The edge server consists of five Dell EMC R340 PowerEdge workstations, each with Intel Xeon E-2246G 3.6 GHz CPU and Ubuntu v18.04. The workstations serve as the controlling hosts of the front-end SDRs for baseband signal processing. The front-end SDR is divided into a *static front-end SDR* and a *mobile front-end SDR*. The former supports experiments with stationary nodes and the latter supports experiments with mobile nodes. For mobile experiments, the SuperDroid programmable robot [16] is used to carry the software radios. In order to control the movement of the ground vehicular robot we use a Marvelmind super beacon [17] to provide a precise indoor positioning and navigation system. The front-end SDRs consist of 20 N210 and 4 B210 USRPs. The N210 USRPs are powered by three CyberPower



(a) (b) (c)

Fig. 3: (a) Initializing VNC server and SSH tunnel; (b) initializing NoVNC server; and (c) NoVNC webpage.



(a) (b) (c)

Fig. 4: Demonstration of (a) point-to-point data transmission; (b) point-to-point data reception; and (c) online video streaming for remote experiment monitoring.

PDU41001 Switched Power Distribution Units (PDU). The PDUs allow experimenters to remotely turn on and off the software radios remotely and reboot them in case of technical issues. The edge servers and the front-end SDRs are connected using two switches with Gigabit Ethernet cables.

Establishing Secure Connection. For remote experiments, it is important to establish a secure connection between the experimenter's personal computer and the edge server that controls the front-end SDRs. This can be accomplished following three steps, as shown in Figs. 3(a)-(c).

(i) Initialization of VNC Server and SSH Tunnel. First, the testbed owner starts the secure VNC server using the `vncserver` command. After successful server initialization, an ssh tunnel is created on the edge server that will securely forward the traffic from the VNC server to the edge server. This is done using command `ssh -L 9901:127.0.0.1:5901 -C -l wings 127.0.0.1`, where the `-L` switch specifies the port bindings. In this case we are binding port 5901 of the remote connection to port 9901 on the local edge server. The `-C` switch enables compression and the `-l` switch specifies the remote login name. A snapshot of the terminal is shown in Fig. 3(a).

(ii) Initialization of NoVNC Server. Second, a NoVNC server is started, which acts as a web-based VNC server. This is achieved using command `sudo novnc --listen 6081 --vnc localhost:9901 --cert/snap/novnc/self.pem`, where `cert` is the certificate for the proxy to encrypt its traffic using WebSocket. Figure 3(b) shows a snapshot of the terminal of this step.

(iii) Initialization of FRP Sever and Client. As discussed earlier, the FRP server plays an important role in exposing the local private IP address of the edge server to a public IP address that can be accessed by the testbed users remotely. This is achieved by instantiating an Amazon Elastic Compute Cloud (EC2) instance to create a public IP address and a RSA encrypted private key. Then, the public IP address provided by FRP-EC2 instance is added to the list of known hosts using the following command `ssh -i frpServerKey.pem ubuntu@3.143.218.142`, where `frpServerKey.pem` is the private key. Then, the FRP proxy server can be started using the command `./frps -c frps.ini`. The terminal snapshot is shown in Fig. 3(c). Finally, the FRP client can be started using command `./frpc -c frpc.ini`.

After successful initialization of the FRP server and the client, the remote edge server can be accessed using the following URL: ⟨`public_ip_address`⟩`/vnc.html`. The access link along with login credentials will be transmitted to testbed users after an initial request form to ensure the safety and integrity of the testbed operations.

Remote Experiment: Point-to-Point Communications. In

this demonstration we will showcase remote experimentation through CloudRAFT. We consider narrow-band point-to-point communications between two USRP N210 software-defined radios, with IP addresses `192.168.10.11` (tx) and `192.168.10.16` (rx), respectively. Two terminals are opened remotely for tx and rx. The first terminal executes the transmission command `python benchmark_tx.py -f 2.45G -r 200000 -m gmsk -M 50 --args="addr=192.168.10.11"`. Similarly, the second terminal executes the receiving command `python benchmark_rx.py -f 2.45G -r 200000 -m gmsk --args="addr=192.168.10.16"`, where `-f`, `-r`, `-m`, `-M`, `--args` specify the center frequency, transmission rate, modulation scheme, total data size and the address of the USRP devices, respectively. The corresponding snapshot with the packet transmission and reception are shown in Figs. 4 (a) and (b). *Notice that both the transmission and receiving programs are started remotely through the secure link established in the above demonstration.*

Live Monitoring of Remote Experiment. In this demonstration we aim to monitor remotely the movement of the mobile nodes during the experiment. To this end, we stream the live video of the experiment based on a combination of Open Broadcaster Software (OBS) [18] and Real-Time Messaging Protocol (RTMP) [19]. The built-in webcam of Dell Inspiron Laptop is used to capture the live video of the experiment, as shown in Fig. 4(c). To enable indoor navigation of the ground robot vehicle without GPS reception, six static beacons deployed on the net enclosure (the enlarged version of the static beacon is shown in Fig. 4(c)) and one mobile beacon is carried by the robot vehicle.

V. Conclusions and Future Work

In this paper we have presented CloudRAFT, a cloud-based remote experimentation framework for mobile networks. We discussed the enabling techniques for CloudRAFT. We have also developed a new software-defined testbed called NeXT and showcased the remote access to NeXT based on CloudRAFT. Through this work we verified the feasibility of accessing, controlling and sharing wireless testbeds through a remote public cloud. In future work, we will standardize the communication interfaces between the cloud plane and testbed plane within the CloudRAFT framework and then fully integrate the NeXT testbed by connecting other software radios available at University at Buffalo, including mmWave radios and UAV systems, among others.

References

[1] "Platforms for Advanced Wireless Research (PAWR) Program." [Online]. Available: https://advancedwireless.org/

[2] J. Breen, A. Buffmire, J. Duerig, K. Dutt, E. Eide, M. Hibler, D. Johnson, S. K. Kasera, E. Lewis, D. Maas, A. Orange, N. Patwari, D. Reading, R. Ricci, D. Schurig, L. B. Stoller, J. Van der Merwe, K. Webb, and G. Wong, "POWDER: Platform for Open Wireless Data-driven Experimental Research," in *Proc. of International Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*, London, United Kingdom, Sep. 2020.

[3] D. Raychaudhuri, I. Seskar, G. Zussman, T. Korakis, D. Kilper, T. Chen, J. Kolodziejski, M. Sherman, Z. Kostic, X. Gu, H. Krishnaswamy, S. Maheshwari, P. Skrimponis, and C. Gutterman, "Challenge: COSMOS: A City-Scale Programmable Testbed for Experimentation with Advanced Wireless," in *Proc. of International Conference on Mobile Computing and Networking*, London, United Kingdom, April 2020.

[4] V. Marojevic, I. Guvenc, R. Dutta, M. L. Sichitiu, and B. A. Floyd, "Advanced Wireless for Unmanned Aerial Systems: 5G Standardization, Research Challenges, and AERPAW Architecture," *IEEE Vehicular Technology Magazine*, vol. 15, no. 2, pp. 22–30, June 2020.

[5] "Agriculture and Rural Communities (ARA)." [Online]. Available: https://arawireless.org/

[6] S. K. Moorthy and Z. Guan, "Beam Learning in MmWave/THz-band Drone Networks Under In-Flight Mobility Uncertainties," *IEEE Transactions on Mobile Computing*, accepted for publication, Oct. 2020.

[7] ——, "FlyTera: Echo State Learning for Joint Access and Flight Control in THz-enabled Drone Networks," in *Proc. of IEEE International Conference on Sensing, Communication and Networking (SECON)*, Como, Italy, June 2020.

[8] Z. Ke, Z. Li, Z. Cao, and P. Liu, "Enhancing Transferability of Deep Reinforcement Learning-Based Variable Speed Limit Control Using Transfer Learning," *IEEE Transactions on Intelligent Transportation Systems*, accepted, May 2020.

[9] "Amazon Web Services." [Online]. Available: https://aws.amazon.com/

[10] "Ettus research." [Online]. Available: https://www.ettus.com/

[11] "AWS Cognito." [Online]. Available: https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

[12] T. E. Anderson, M. D. Dahlin, J. M. Neefe, D. A. Patterson, D. S. Roselli, and R. Y. Wang, "Serverless Network File Systems," in *Proc. of ACM symposium on Operating systems principles*, Copper Mountain, Colorado, Dec. 1995.

[13] G. Adzic and R. Chatley, "Serverless Computing: Economic and Architectural Impact," in *Proc. of Joint Meeting on Foundations of Software Engineering*, Paderborn, Germany, Aug. 2017.

[14] "Fast Reverse Proxy (FRP)." [Online]. Available: https://github.com/fatedier/frp

[15] T. Richardson and K. R. Wood, "The RFB Protocol," *ORL, Cambridge*, Jan. 1998.

[16] "Configurable - Programmable Mecanum Wheel Vectoring Robot - IG52 DB." [Online]. Available: https://www.superdroidrobots.com/shop/item.aspx/configurable-programmable-mecanum-wheel-vectoring-robot-ig52-db/1788.

[17] "Super-Beacon." [Online]. Available: https://marvelmind.com/product/super-beacon/.

[18] T. Heycke and L. Spitzer, "Screen Recordings as a Tool to Document Computer Assisted Data Collection Procedures: A Tutorial using the Open Broadcaster Software (OBS)," *Psychologica Belgica*, 2019.

[19] A. Nurrohman and M. Abdurohman, "High Performance Streaming Based on H264 and Real Time Messaging Protocol (RTMP)," in *Proc. of IEEE International Conference on Information and Communication Technology (ICoICT)*, Bandung, Indonesia, May 2018.