Software Defined Access for HetNets

Vidya Sagar, R. Chandramouli, and K. P. Subbalakshmi

The authors present software defined access (SDA), an architecture for HetNets that incorporates various elements of SDN and software defined wireless networking (SDWN). SDA introduces a novel logical control path across radio interfaces and up to mobile devices. Unlike SDN and SDWN, SDA can be deployed without changing network elements of radio access technologies.

ABSTRACT

We present software defined access (SDA), an architecture for HetNets that incorporates various elements of SDN and software defined wireless networking (SDWN). SDA introduces a novel logical control path across radio interfaces and up to mobile devices. This control path allows SDA to regulate and configure data path via different RATs. Unlike SDN and SDWN, SDA can be deployed without changing network elements of RATs. The userspace implementation of client side utility is also presented to empower mobile devices with SDN-like flow management. This client with cloud hosted SDA servers presents an alternative for operator-independent and RAT-agnostic standalone SDA deployment. To demonstrate some of SDA's capabilities, we also present various testbed experimental measurement results.

INTRODUCTION

The proliferation of smart devices and mobile applications has brought web-based services closer to the masses. It has changed the way we order pizza, use taxis, and bank. Most importantly, it has transformed our means of personal and professional communication. This has given birth to ever increasing demands of mobile data traffic. Further enhancements in device, communication, and web technologies are going to fuel this even more. This growing demand has already put network infrastructure under a lot of stress. Due to fierce competition, average revenue per user (ARPU) is lagging behind this burgeoning demand. Increasing demand and decreasing revenue present an imminent threat to networks and operators.

To handle this situation, multiple solutions are being adopted. These solutions aim to improve network utilization, boost spectral efficiency, and enhance interworking among different radio access technologies (RATs), such as Long Term Evolution (LTE) and WiFi. We enumerate some of the widely discussed solutions below:

Traffic Engineering: Traditionally, networks offer minimal intelligence at the core with simple functionalities (e.g., routing). Traffic is usually routed through a fixed or the shortest path. This inefficient scheme causes congestion on some network paths, whereas others remain underutilized. Routing with awareness of load on different paths can improve overall system utilization. Moreover, network flows carrying traffic usually have diverse service requirements. For example, some applications and their flows are delay-tolerant; therefore, routing them through a non-shortest path while meeting their quality of service (QoS) requirements can reduce load on the shortest path. Such intelligent traffic engineering with awareness of load and flow requirements can improve network utilization and guarantee flow-specific QoS requirements.

Network Virtualization: To reduce operational cost, operators have increasingly embraced network virtualization. It facilitates shared network resources among multiple different operators with better service guarantees. Software defined networking (SDN) is an approach that virtualizes network elements (e.g., switches and routers) and shares them among different flows, services, and operators. SDN decouples the control plane from the data path, and uses a centralized controller to configure the data path through the control plane [1]. Network functions virtualization (NFV) is another approach, which virtualizes different network servers (name server, etc.) with the help of IT virtualization techniques. Although independent of each other, SDN with NFV can provide efficient traffic engineering and better service orchestration.

Heterogeneous Deployment: Increasing mobile traffic demands spectral efficiency. Current RATs already operate near theoretical limits of spectral efficiency at the physical layer. Further increments in spectral efficiency are expected from diversity gains (e.g., transmission or reception diversity). Heterogeneous networking (HetNet), with low-power intermediate nodes (e.g., femtocells) and small cells, can leverage these diversities and boost spectral efficiency even more. Additionally, HetNets are useful in reducing coverage holes and extending cell coverage.

MultiRAT Architecture: Multiple different RATs in a HetNet deployment operating at different frequencies is considered part of evolving fifth generation (5G) networks [2]. Although HetNet deployment can provide better connectivity to the core network through individual RATs, spectrum scarcity and growing demand advocate enhanced interworking among these multi-RAT HetNets. Such convergence requires an architecture to incorporate multiple RATs and facilitate interworking. Software defined wireless networking (SDWN) is expected to address this architectural need. SDWN interworks with multiple RATs and extends the control path to the infrastructure plane consisting of base stations, access points, and so on. Using an

This research was supported by the National Science Foundation under award no.1353340

The authors are with Stevens Institute of Technology.

SDN-like interface with applications, SDWN can also configure RATs to support application-specific demands. Different technologies mentioned above suggest that a futuristic network will have converged multi-RAT HetNet deployment and an SDWN/SDN-like architecture with enhanced interworking and intelligent traffic engineering [3]. In this work, we address such a converged multi-RAT HetNet architecture.

SDMN advocates changes to various RAT elements. Adopting these changes across different RATs will be an expensive and slow process. Therefore, we propose a readily deployable alternative called software defined access (SDA).

This work is organized as follows. We present an overview of SDN and SDWN architectures. Thereafter, we review a few relevant MultiRAT interworking mechanisms. Then we introduce the proposed SDA architecture and discuss different SDA procedures enabling MultiRAT interworking. The prototype implementation is explained, and experimental results are presented to demonstrate operator-independent QoS-aware offloading. Finally, we conclude with pointers to future work.

Overview: SDN, SDWN, and SDA

Figure 1 shows SDA along with SDN and SDWN. For simplicity, we have presented SDN, SDWN, and SDA independent of each other. However, SDN is generally considered to be part of SDWN, and we foresee SDWN adopting some of SDA's functionalities too. To understand the differences between these architectures, we first identify the following five planes.

Application Plane: The application plane consists of different application servers. NFV techniques can be employed to virtualize these servers for better service orchestration.

Control Plane: SDN has introduced this new plane consisting of an SDN controller. We place the SDWN controller and SDA controller in this plane. These controllers are responsible for intelligent traffic policing and employ the control path to enforce these policies. The SDN control path is limited to the core network, whereas SDWN extends the SDN control path to the RAT infrastructure (e.g., base station, access point), and the proposed SDA introduces a novel control path across the air interface up to a mobile device. All these programmable controllers can also interact with the application plane and adapt policies to accommodate different application-specific requirements. OpenFlow SDN architecture has standardized northbound application programming interfaces (APIs), which govern the interface between the control and application planes [1].

Data Plane: This plane consists of a data path with switches and routers constituting the legacy network core. The SDN control path terminates in this plane. OpenFlow SDN architecture suggests southbound APIs to administer traffic policies between the control and data planes [1].

Infrastructure Plane: This plane consists of different RAT elements (base station, access points, etc.). SDWN attempts to extend the SDN control path up to this plane. Although the SDWN architecture is still evolving, there are a few relevant architectures for SDWN. OpenRoads extends OpenFlow by using Simple Network Management Protocol (SNMP) [4]. On the

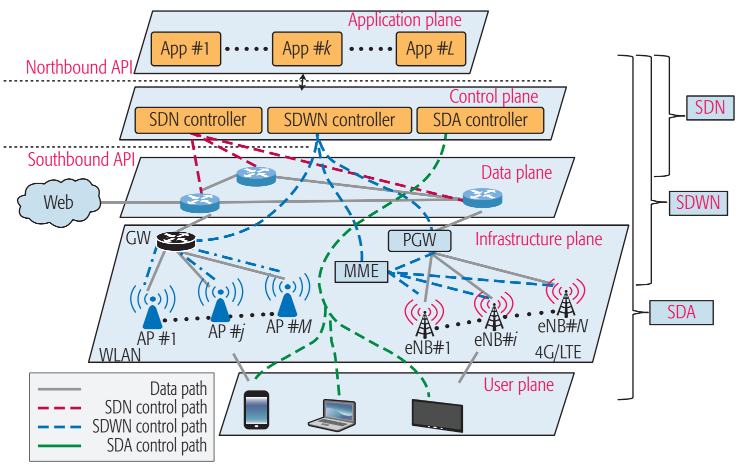

Figure 1. SDN, SDWN, and SDA.

other hand, RAT-specific southbound APIs are employed for HetNet SDN [5]. OpenRadio outlines refactoring radios to devise a programmable data plane for SDWN [6].

User Plane: The user plane is the last plane consisting of end-user equipments (UE) (e.g., tablets and smartphones). Due to the prevalence of multiple radios in these devices, they can connect to the network core via multiple RATs. By selecting different radios for different applications, these devices can act as wireless switches for traffic originating or terminating at them. Moreover, a data path over a given RAT can be exploited to configure a data path over other RATs. In multi-RAT deployments, a temporal logically separate control path can also be devised from multiple data paths. The proposed SDA architecture employs such a logical control path between UE and SDA controller.

Existing Interworking Methods

Interworking among different RATs will be a key component in emerging networks. We now present an overview of existing multi-RAT interworking mechanisms and their limitations in order to motivate the need for an innovative solution.

The Third Generation Partnership Project (3GPP) has adopted several measures to facilitate LTE-WiFi interworking for LTE-Advanced (LTE-A) systems. MAPCON allows UE to maintain simultaneous multiple data paths over different RATs [7]. The access network discovery and selection function (ANDSF) and IFOM mechanisms are proposed to enable RAT selection and flow mobility across different RATs [8]. These technologies rely on DualStack Mobile IPv6 (DSMIPv6), but it requires kernel-level modification to UEs. Proxy Mobile IPv6 (PMIPv6)-based flow mobility requires changes in RAT elements [9]. Apart from UE or RAT modifications, these technologies also mandate coordination among operators. Cellular and non-cellular services are usually marketed separately, and very often users subscribe to these services separately too. These 3GPP mechanisms may not be helpful for users with subscriptions to non-coordinating operators.

We attempt to address these shortcomings by introducing SDA. We propose SDA as a readily deployable architecture to facilitate operator-in-

We propose SDA as a readily deployable architecture to facilitate operator-independent multi-RAT interworking with minimal changes in existing RAT elements. We believe that the operator-independent architecture will also help operators by reducing RAT complexity and promoting better utilization of current RAT infrastructure.

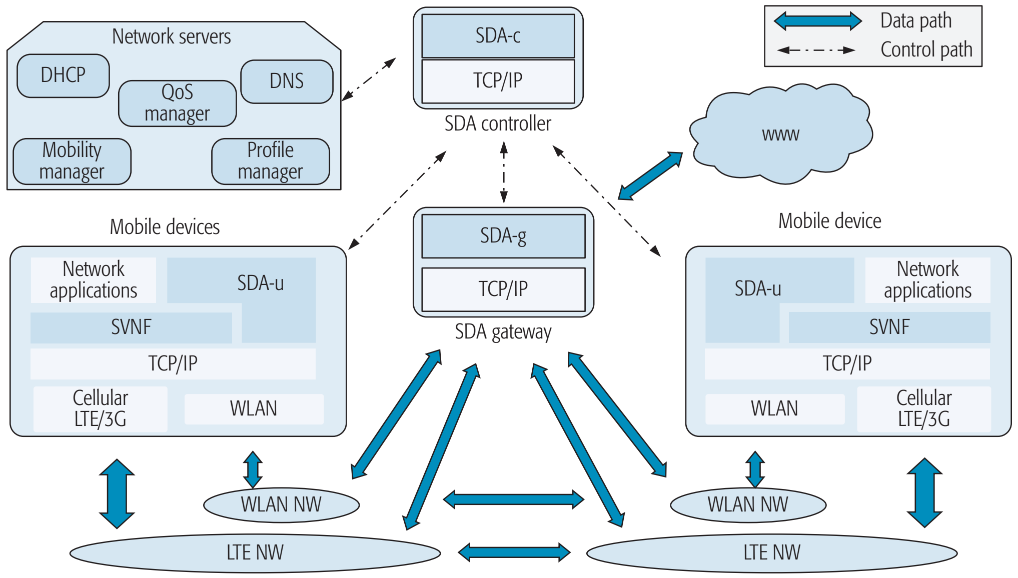

Figure 2. Proposed SDA architecture.

dependent multi-RAT interworking with minimal changes to existing RAT elements. We believe that the operator-independent architecture will also help operators by reducing RAT complexity and promoting better utilization of current RAT infrastructure.

SDA ARCHITECTURE

Figure 2 presents the proposed SDA architecture. It consists of two new network entities, SDA-Gateway (SDA-g) and SDA-Controller (SDA-c). On UE, it makes use of SDA userspace client (SDA-u) and SDA virtual network function (SVNF). SDA-g and SDA-c are assumed to be accessible via all the different RATs available to UEs. Details of different elements are presented below.

SDA Virtual Network Function: The SVNF is a system utility that provides a virtual network interface. It acts as a bidirectional conduit between applications and SDA-u. It maintains an IP address assigned by SDA-c. This RAT-independent IP address allows applications to maintain their sockets and flows during handover and coverage loss.

SDA-u: SDA-u is a client side utility running as a service on UEs. It is responsible for authenticating mobile devices with SDA-c. It also updates SDA-c with user preferences. It complies with traffic policies provided by SDA-c. SDA-u is responsible for configuration of the data path between SVNF and SDA-g. SDA-u splits various outgoing flows into subflows specific to different RATs. It also retrieves incoming flows from RAT-specific subflows. Other responsibilities of SDA-u include loss recovery, delay adjustment, and encryption of the data path.

SDA-g: SDA-g works as a proxy server and a gateway for the SDA network. On the data path, it performs functions similar to the SDA-u. It also reports different subflow characteristics (e.g., delay, loss, and throughput) to SDA-c.

SDA-c: It is the centralized SDA controller. SDA-c authenticates UEs and authorizes them in accordance with their profile and preferences. It establishes different data subflows between SDA-u and SDA-g. SDA-c enforces a multitude of traffic policies (e.g., offloading, aggregation) over these subflows. Additionally, it monitors the data path using measurement reports from SDA-u and SDA-g. It can also detect several impairments (interference, congestion, etc.) through these measurements and adapt traffic policies to ensure better QoS.

DATA AND CONTROL PATH

For the data path, SDA relies on multiple subflows. These subflows are network flows bound to different RATs. SDA-u and SDA-g disseminate outgoing user data over these subflows. They also retrieve incoming flows from these subflows. These dissemination policies of user flows over subflows are governed by SDA-c. SDA-c classifies traffic on a per flow basis and enforces different policies for different flows. These policies are essentially routing policies with weighted matrices, and provide mapping between user flows and RAT-specific subflows. SDA realizes flow offloading from a RAT by removing mapping of flows from that RAT or associated subflow. Similarly, aggregation can be achieved by selecting multiple RATs or subflows with appropriate weights.

For the logical control path, SDA depends on multiple UDP flows. It uses three-way handshake and *UDP hole-punching* to traverse through Network Address Translation (NAT). Communication over this control path dynamically selects the appropriate RAT to achieve better reliability and have the least impact on data flows. This logical flow is initiated by SDA-c during the registration process after successful authentication. Although this work does not consider symmetric and restricted cone NATs, we assume three-way handshake can be replaced with an appropriate procedure to enable UDP NAT traversal [10]. The SDA three-way handshake is as follows. SDA-c assigns a local port number and a connection identifier to SDA-u. SDA-c sends this information over an initial TCP connection used for authentication. Thereafter, SDA-u opens a UDP socket bound to *localhost* and communicates to SDA-c on the assigned UDP port with the connection identifier. SDA-c learns the public IP information (public IP address and port number) of SDA-u and sends an acknowledgment to SDA-u over a new control socket. It concludes three-way handshake. Whenever SDA-u registers to a new RAT, it makes use of the previously established control path to initiate this three-way handshake over the new RAT. It helps SDA-c to maintain a list of public IP information linked to the same SDA-u over different RATs. Finally, dynamic RAT selection in this logical control path is achieved by selecting appropriate destination IP information in downlink and source IP information in uplink communication.

Interworking with SDN/SDWN

SDA introduces one new node in the control plane (i.e., SDA-c) and one in the data plane (i.e., SDA-g). SDA can achieve operator independence by hosting these servers outside operators' networks. Additionally, SDA-c can interact with the SDN/SDWN controller of the underlying RATs for enhanced interworking and better resource management. We foresee SDWN/SDN to accommodate SDA controllers into their controllers. However, SDA can also be deployed independently without SDN/SDWN as a cloud service (i.e., *SDA-as-a-service*). We illustrate our prototype implementation of SDA-as-a-service later in this article.

Call Flow in SDA Systems

Figure 3 illustrates a sample call flow with registration, measurement, and flow offload procedures in SDA. We consider UE with LTE and WiFi radios. We assume LTE as the primary RAT and WiFi as the preferred RAT for flow offloading. Therefore, SDA-u will register and start communication over LTE; then the availability of WiFi will cause flow offload from LTE to WiFi. As shown in Fig. 3, SDA-u sends a *RegistrationRequest* to SDA-c using a TCP connection over LTE. This request consists of identity and authentication data as well as user preferences. SDA-c makes use of this information to authenticate and identify an appropriate QoS level for SDA-u. To configure the data path, SDA-c sends an *AddSession* message to SDA-g. This message includes *ConfigureSession*, which provides configuration for data and control path as well as periodic measurement reports. *ConfigureSession* also embeds a *PolicyUpdate* message to configure the transmission policy. The transmission policy enables SDA-g/SDA-u to schedule packets among different RATs. As discussed earlier, SDA transmission policies are essentially routing policies. In a standalone implementation of SDA, these policies do not interact with RAT-specific radio resource scheduling. Furthermore, SDA-g acknowledges successful configuration to SDA-c with an *AddSessionACK* message. Thereafter SDA-c creates a UDP socket for the control path and sends a *RegistrationACK* message to SDA-u indicating successful registration. This *RegistrationACK* also includes *ConfigureSession* consisting of configurations for control, data path, and measurement sessions. It also conveys an embedded *PolicyUpdate*

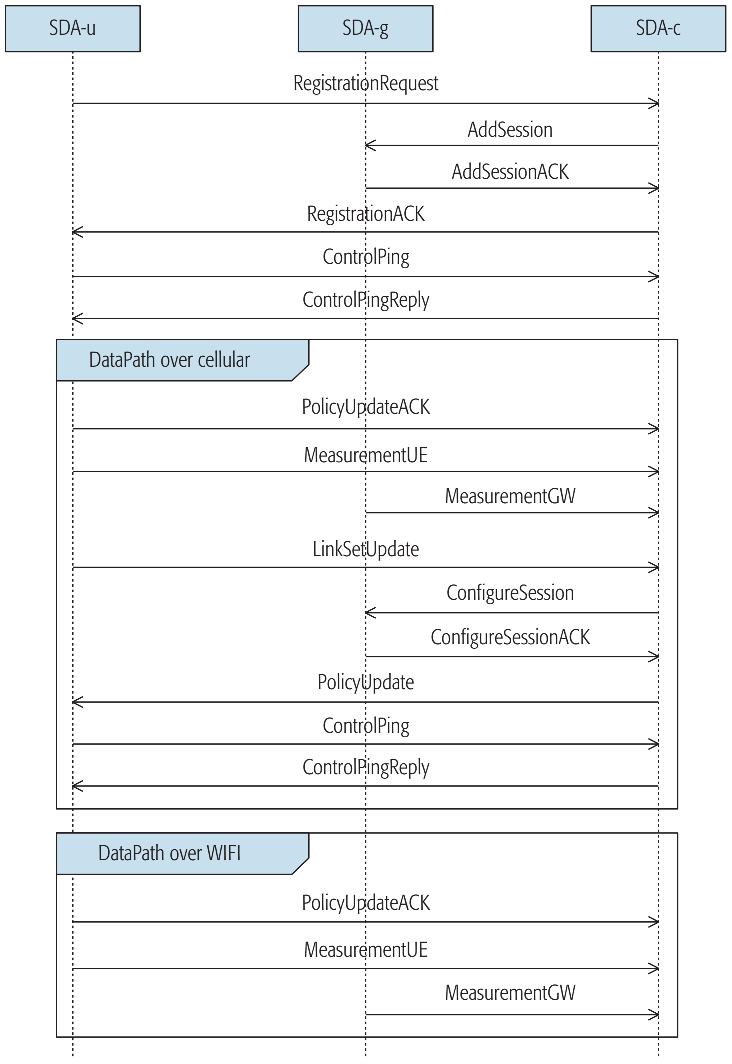

Figure 3. Sequence diagram for offloading.

message along with scheduling policy and initiates a three-way handshake for initial control path configuration. After successful registration, SDA-u sends *ControlPing* over UDP using the default RAT to SDA-c. SDA-c learns the public IP address (and port number) of SDA-u and sends an acknowledgment back by sending *ControlPingReply* over the control path. Finally, SDA-u sends *PolicyUpdateACK* to SDA-c and terminates the TCP session with SDA-c. Thereafter, user data is carried over LTE, periodic measurements are reported by SDA-u and SDA-g.

When WiFi becomes available, SDA-u sends *LinkSetUpdate* message to SDA-c with WiFi network information over the control path. Since WiFi is the preferred network, SDA-c triggers offload of data flows. It updates SDA-g about the new data path by sending *ConfigureSession*. SDA-g replies back with *ConfigureSessionAck* to SDA-c, indicating the completion of configuration. SDA-c sends a *PolicyUpdate* message to SDA-u with the new scheduling policy. This is followed by a three-way handshake to update the logical control path over WiFi. Once the control path over WiFi is configured, SDA-u sends a *PolicyUpdateACK* over WiFi, and all data flows are offloaded to WiFi.

Also, SDA-u and SDA-g send periodic measurement reports assisting SDA-c in monitoring.

Apart from priority-based offloading and registration, the above example also illustrates measurement procedures in SDA. SDA-c can configure these measurement sessions to identify a variety of conditions (e.g., coverage loss, congestion). This measurement information can be used to facilitate QoS aware traffic engineering.

Prototype Implementation

In this section, we discuss the standalone deployment of SDA-as-a-service. We make use of SDA-c and SDA-g servers hosted on a cloud infrastructure. SDA-u registers to SDA-c with its credentials and preferences. SDA-c facilitates UDP-based control and data paths to SDA-u. SDA-c also assigns an IP address that does not change with changes in the IP addresses of the underlying RATs. SDA-c makes use of different measurement reports from SDA-u and SDA-g to identify the optimal scheduling policy of user data over RAT-specific subflows.

Cloud hosted SDA servers provide the necessary framework for operator-independent SDA deployment with no changes in underlying RATs. To make SDA-as-a-service a readily deployable solution, we present the design of SDA-u implemented as a userspace application.

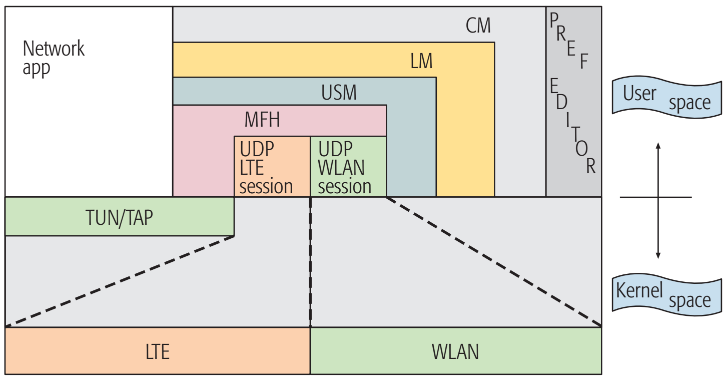

Figure 4. SDA-u design.

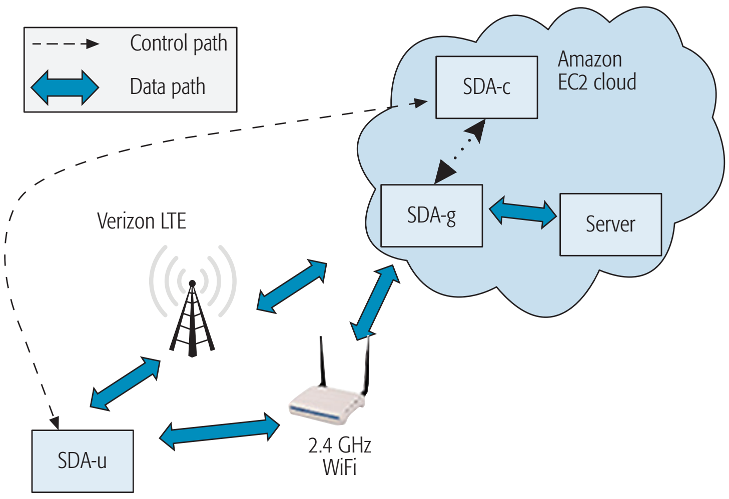

Figure 5. SDA-as-a-service experimental setup.

SDA-u Design

Our SDA-u design depends on basic functionalities (e.g., socket) provided by many operating systems (e.g., Unix/Linux, Windows). Figure 4 illustrates the design of SDA-u. Each component of SDA-u is explained below.

Tun/Tap: Tun/Tap is used as an SVNF here. It carries IP configuration provided by SDA-c. The Tun/Tap interface forwards all the outgoing packets sent by other network applications to SDA-u. It also delivers all the packets sent by SDA-u to the respective applications.

Multiple Flow Handler: The MFH is a core module for the data path. The MFH is common between SDA-g and SDA-u. It reads outgoing flows from Tun/Tap and employs different functions (e.g., encapsulation, encryption) to process them. Processed flows are forwarded to different UDP sessions/subflows bound to different RATs. It communicates with SDA-g and ensures multilink reliable UDP connectivity.

UDP Session Manager: The USM maintains and manages different UDP sessions with SDA-g. The USM reestablishes UDP sessions upon link failure or handover. It also takes care of the maintenance of the control path.

Link Manager (LM): The LM maintains IP connectivity with various RATs. It triggers RAT-specific procedures for registration and association. It requests the USM to reestablish subflows upon changes in IP configuration. It also identifies link losses/failures and requests the USM to close associated subflows.

Client Manager: The CM is the main controller of the SDA-u software system. It takes care of authentication and association with SDA-c and also reports measurement information requested by SDA-c. It enforces different policy updates from SDA-c.

Preference Editor: PrefEdit is a user interface. PrefEdit allows a user to modify preferences, RAT priorities, and RAT-specific procedures.

This design does not require kernel updates, and can be implemented and installed as an application. User space implementation of SDA-u along with cloud hosted SDA-c and SDA-g servers presents a readily deployable solution to facilitate operator-independent multi-RAT interworking without altering the underlying RATs.

Experimental Results

Figure 5 presents our SDA-as-a-service experimental setup. We implemented SDA-u on a mini-ITX unit with a 1.86 GHz Intel Atom processor, 4 GB RAM, and an Intel Advanced-N 6235 wireless card. SDA-c and SDA-g servers are deployed on Amazon ec2 cloud. All SDA entities were implemented on Ubuntu 14.04. A Netgear WNDR3400v2 access point with a university LAN is used to provide the WiFi data path. The LTE data path is established using a Pantech UML 290 card over Verizon's LTE network. The IPERF utility is used to evaluate throughput over this setup. To demonstrate the capability of SDA-as-a-service, we emulated congestion in the LTE data path. SDA-c identifies congestion and offloads flow from LTE to avoid throughput degradation. We call it QoS-aware offloading.

Figure 6 presents QoS-aware offloading under SDA-as-a-service. This figure presents a timeline plot of TCP throughput performance over the SDA data path. In this experiment, we assume LTE as the preferred network. For the first 20 s, the SDA data path is catered for by the LTE data path. During this period the WiFi data path is not used, although SDA-c monitors it through measurement reports. Meanwhile, the average throughput performance observed is 7.7 Mb/s. After 20 s, NETEM and IPTABLE utilities are used to reduce the LTE throughput to 500 kb/s. The SDA-c detects it as core network congestion and offloads traffic to the WiFi network. Up to the 40th second this simulated congestion is maintained, and traffic is served through the WiFi data path with 3.6 Mb/s of average throughput. When throttling of LTE is removed, SDAc detects this event. Since the preferred RAT (i.e., LTE) can now serve a larger throughput, SDA-c offloads flow to LTE. In this last stage, SDA data path is routed over LTE with an average throughput of 7.3 Mb/s. In this experiment, offloading delay is measured as 0.8 s for LTE to WiFi offloading and 1.3 s for WiFi to LTE offloading. LTE and WiFi networks exhibit different round-trip delays, 180 ms and 35 ms, respectively. When flow is offloaded from WiFi to LTE, due to increased delays, TCP throughput degrades initially, and thereafter recovers and saturates the LTE data path. This is why throughput drops after 40 s before ramping up to the appropriate LTE throughput.

These experimental results demonstrate the capability of standalone implementation of SDA (i.e., SDA-as-a-service) to offload flow to and from different RATs as well as the capability to detect congestion in the data path and adapt accordingly.

Advantages of SDA

As discussed above, *SDA-as-a-service* implementation makes SDA agnostic to network operators and the underlying RATs. We foresee SDA as a complementary to futuristic SDWN architectures. Some of the advantages of SDA are the following.

Logical Control Path: Extension of the logical control path up to mobile devices will provide better means of control and reconfigurability to SDWNs.

User-Centric Architecture: SDA introduces UEs into an SDWN-like architecture. This can help SDWNs to converge into a user-centric architecture. Such an architecture can bring fine-grained control over various network elements with an awareness of user preference and can provide superior QoS.

QoS-Aware Flow Mobility: SDWN-based flow offloading often relies on fixed preferences for RATs (e.g., offload to WiFi). Such offloading can result in poor user experience if offloaded flow encounters poor network conditions (e.g., interference). By adopting SDA like operator-independent offloading, SDWN can cater for seamless and QoS-aware offloading.

Application/Service Awareness: SDA provides integration of user preferences and application service requirements in SDWN/SDN architecture. For example, it can enable opportunistic multi-RAT aggregation for data-hungry applications.

Conclusion

We have presented SDA, a novel architecture to facilitate interworking in multi-RAT HetNets. SDA also provides better reconfigurability and

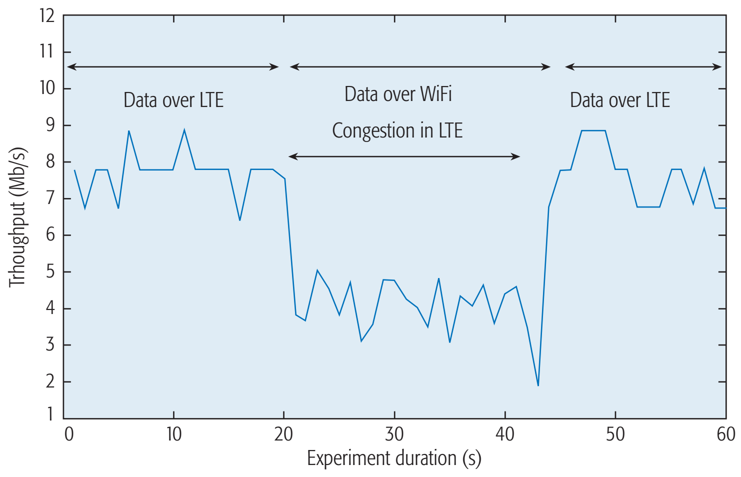

Figure 6. SDA data offloading performance.

control to SDWNs without changing any network elements. A standalone deployment of SDA, SDAas-a-service, is also introduced. Experimental results show the efficacy of the prototype deployment.

In the future, we will implement SDA-as-a-service on smart devices and evaluate the impact of these mechanisms on battery life.

References

[1] N. McKeown *et al.*, "Openflow: Enabling Innovation in Campus Networks," *SIGCOMM Comp. Commun. Rev.*, vol. 38, no. 2, Mar. 2008, pp. 69–74.

[2] B. Bangerter *et al.*, "Networks and Devices for the 5G Era," *IEEE Commun. Mag.*, vol. 52, no. 2, Feb. 2014, pp. 90–96.

[3] R. Wang, H. Hu, and X. Yang, "Potentials and Challenges of C-RAN Supporting Multi-Rats Toward 5G Mobile Networks," *IEEE Access*, vol. 2, 2014, pp. 1187–95.

[4] K.-K. Yap *et al.*, "Openroads: Empowering Research in Mobile Networks," *SIGCOMM Comp. Commun. Rev.*, vol. 40, no. 1, Jan. 2010, pp. 125–26.

[5] S. Tomovic *et al.*, "SDN-Based Concept of QoS Aware Heterogeneous Wireless Network Operation," *2014 22nd Telecommun. Forum Telfor*, Nov. 2014, pp. 27–30.

[6] M. Bansal *et al.*, "Openradio: A Programmable Wireless Dataplane," *Proc. 1st ACM Wksp. Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, 2012, pp. 109–14.

[7] 3GPP, "Multi Access PDN Connectivity and IP Flow Mobility," TS 23.861, Feb. 2010.

[8] 3GPP, "IP Flow Mobility and Seamless Wireless Local Area Network (WLAN) Offload; Stage 2," TS 23.261, Mar. 2012.

[9] A. de la Oliva *et al.*, "IP Flow Mobility: Smart Traffic Offload for Future Wireless Networks," *IEEE Commun. Mag.*, vol. 49, no. 10, Oct. 2011, pp. 124–32.

[10] R. Mahy, P. Matthews, and J. Rosenberg, "Traversal Using Relays Around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," IETF RFC 5766, Apr. 2010.

Biographies

Vidya Sagar (vsagar@stevens.edu) is a Ph.D candidate in the Department of Electrical and Computer Engineering, Stevens Institute of Technology. His research interests include cognitive, cloud and social networks.

R. Chandramouli (mouli@stevens.edu) is the Thomas Hattrick Chair Professor of Information Systems in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, and co-founder of Dynamic Spectrum, LLC and jaasuz.com. His research spans the areas of cognitive wireless networking, text mining, social media security, and analytics. His projects have been supported by the U.S. National Science Foundation, National Institute of Justice, Department of Defense (DoD) agencies, and industry

K. P. Subbalakshmi (ksubbala@stevens.edu) is a professor in the Department of Electrical and Computer Engineering, Stevens Institute of Technology. Her research interests include cognitive radio networks, network security, media forensics, and social networks. She is the Founding Chair of the Special Interest Group on Security of the IEEE TCCN. Her work has been supported by the National Science Foundation, National Institute of Justice, DoD, and industry. She is also co-founder of Dynamic Spectrum LLC and Jaasuz.com.