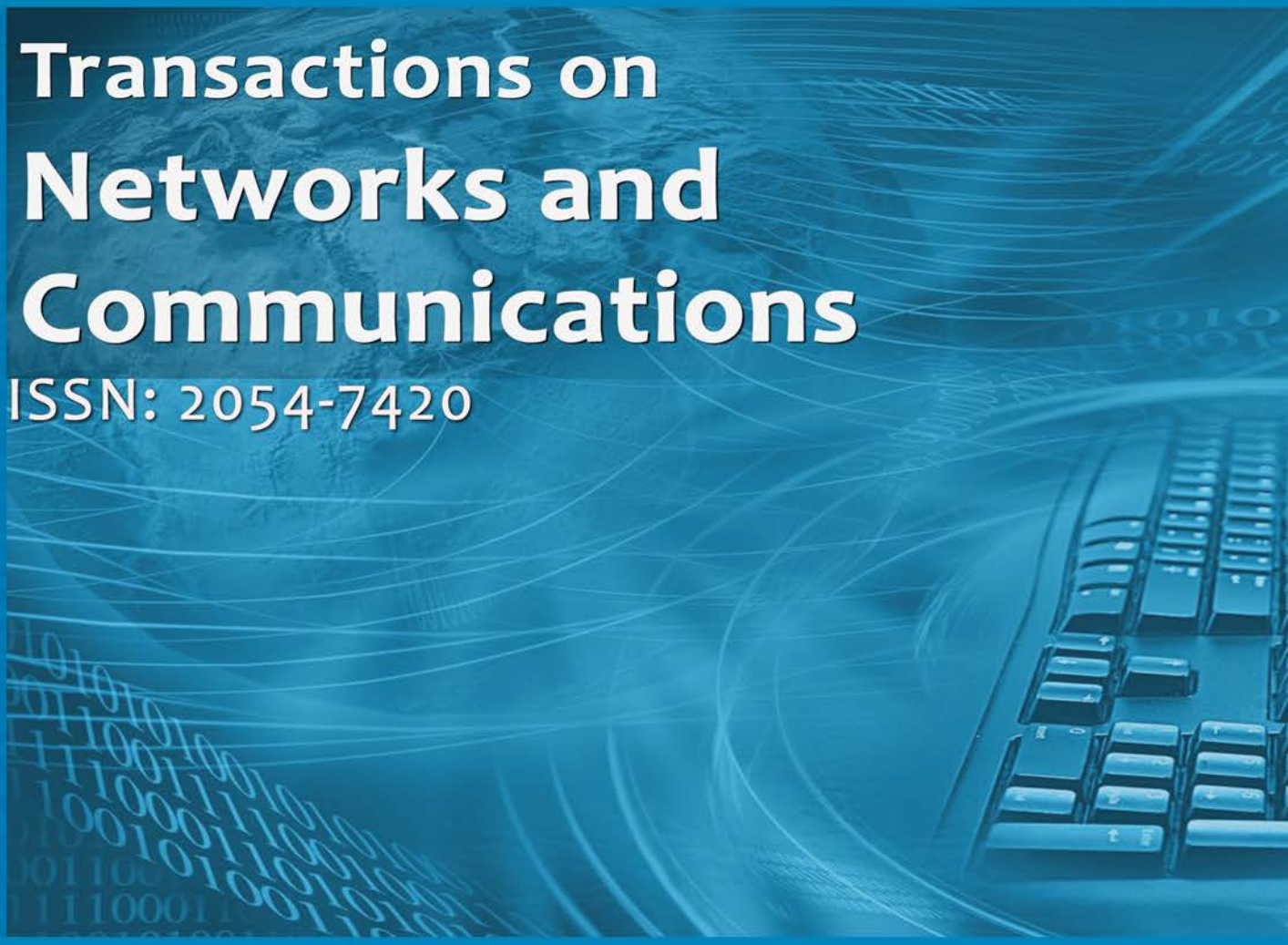
Volume 6 No. 2, April 2018

Transactions on Networks and Communications

ISSN: 2054-7420



Society for Sciences and Education - United Kingdom

TABLE OF CONTENTS

DISCLAIMER

All the contributions are published in good faith and intentions to promote and encourage research activities around the globe. The contributions are property of their respective authors/owners and the journal is not responsible for any content that hurts someone's views or feelings etc.

SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 6, NO. 2
ISSN: 2054 -7420

Vertical Handover Study on 4G Category vs. 5G Category for 3GPP Generation Mobile Systems and Non-3GPP Wireless Networks

Dr Omar Khattab
Department of Computer Networks & Communications
Computer Sciences & Information Technology, King Faisal University
Kingdom of Saudi Arabia
okhattab@kfu.edu.sa

ABSTRACT

Nowadays, wireless communication technologies have become an integral part of people's daily life and businesses all over the world. Due to the rapid increase in the number of the Mobile Users (MUs) who demand the service of communicating via wireless access networks, the wireless communication technologies have evolved from the first generation to the fifth generation. The Next Generation Wireless Systems (NGWSs) consists of heterogeneous wireless access networks of 3GPP generation mobile Systems (e.g., UMTS, LTE) and non-3GPP wireless networks (e.g., WiFi, WiMAX), where MUs can access these technologies and services using a single device. This paper presents a study for Vertical Handover (VHO) approaches of 3GPP and non-3GPP proposed in the literature and classifies them into two categories for which their characteristics are discussed.

Keywords: Vertical Handover; Mobile Systems; Wireless Networks; 4G; 5G.

1 Introduction

The rapid evolutions in broadband wireless communication technologies and the growing Mobile Users' demand (MUs) for communication services anywhere, anytime are driving an evolution toward the seamless integration between different Radio Access Technology (RATs) in heterogeneous wireless communication technologies to provide the best connected services to the MU constantly [1]. The benefits of heterogeneous wireless communication technologies are many and varied. These include: flexibility, reducing cost, simplifying the operation and maintenance, rapid deployment of services and applications, new services, high data transmission, customisation, support multimedia services at lower cost of transmission, the mobility of the sessions and the possibility to transfer the context [1].

The Next Generation Wireless Systems (NGWSs) consists of heterogeneous wireless communication technologies of 3GPP and non-3GPP, where MUs can access these technologies and services using a single device. This device is equipped with multiple radio interfaces include devices capable of supporting multiple RATs by incorporating several interface cards and appropriate software for switching between multiple access systems (Vertical Handover (VHO)).

DOI: 10.14738/tnc.62.4353
Publication Date: 12th April 2018
URL: http://dx.doi.org/10.14738/tnc.62.4353

This paper presents background information on heterogeneous wireless communication technologies. Then, it overviews VHO approaches of 3GPP and non-3GPP proposed in the literature and classifies them into two categories for which their characteristics are discussed.

The rest of the paper is organized as follows: In section 2, background information on heterogeneous wireless communication technologies is presented. In section 3, classifications for VHO approaches of 3GPP and non-3GPP are presented. In section 4, a comparison for VHO approaches of 3GPP and non-3GPP is presented and finally, section 5 concludes the paper.

2 Background on Heterogeneous Wireless Communication Technologies

In this section, background information on heterogeneous wireless communication technologies is presented to answer the following questions: how have wireless communication technologies evolved? What are heterogeneous wireless communication technologies? Who needs heterogeneous wireless communication technologies? Why are heterogeneous wireless communication technologies necessary? and finally, what is the handover management within heterogeneous wireless communication technologies.

1. How Have wireless communication technologies Evolved?

During the last few years, telecommunication authorities were busy while working out how to emerge to the next generation of wireless technology environment which was motivated by the growing demand for advanced telecommunication services which require wider spectrum and higher QoS [2]. Besides, the telecommunication industry experts are required to develop an interoperability strategy for new mobile wireless systems which can satisfy MUs' demands of telecommunication systems [2]. This section presents a background of the main wireless communication technologies, as shown in Table 1.

2.1 GSM

GSM is a 2G mobile system which is the first one to specify digital modulation and network level architectures and services, the first important set of Radio Frequency (RF) for GSM standard started at 1900 MHz [3]. GSM was first introduced in Europe in 1991 and today is one of the most popular digital mobile telecommunications systems widely used over the world [3]. Due to the increase of the number and the requirement of GSM subscriber the GSM is still an attractive area for research in the field of mobile telecommunication [3-5].

2.2 UMTS

2G like GSM were originally designed for efficient delivery of voice services. 3G systems like UMTS were designed from the beginning for mobile voice and data users [6]. Therefore, UMTS is the evolution of GSM system and General Radio Packet Service (GPRS) developed by Third Generation Partnership Project (3GPP) to increase the support for some features such as data rate in radio interface and the compatibility for the two services domains: Packet Switched (PS) and Circuit Switched (CS) data transmission [7]. Some of the most common keys drive of this type of UMTS access technology [8]:

- Growth in the market for fixed networked multimedia services.
- Increasing demand for rapid and remote access to information.
- E-Commerce and transaction based applications.

2.3 Wi-Fi

The Wi-Fi (IEEE 802.11) is wireless networks designed to provide broadband for Wireless Local Area Network (WLAN) where the MUs use the mobile devices (e.g., mobiles and laptops) to access the internet in small geographic area such as university's buildings, airports and railway stations. Over 97% of laptops today come with Wi-Fi as a standard feature and an increasing number of handhelds and Consumer Electronics (CEs) devices are adding Wi-Fi capabilities [9] as Wi-Fi technology in conformance with IEEE 802.11 are growing every year [28, thesis]. The initial standard IEEE 802.11, which came in 1997, had a data rate of 1 Mbps [10]. By year 1999 this was changed; 802.11a (54 Mbps at wider frequency band), 802.11b (11 Mbps, same frequency band but a different modulation technique) and 802.11g (using modulation technique of 802.11a but frequency band of 802.1lb) [10]. During the period between 1990-2000, the IEEE committee, which had already created wired LAN standards (802.3 Ethernet), started processing wireless LAN standard [10]. As Ethernet was dominant at that time, the committee decided to make wireless standard 802.11 compatible with Ethernet above data link layer; however, it was different from Ethernet in link layer and physical layer due to various issues faced the wireless communication [10].

2.4 4G

Growing demand for new applications required to be supported by new mobile systems such as Voice over Internet Protocol (VoIP), video conference, Push to-talk-over Cellular (PoC), multimedia messaging, multiplayer games, Virtual Private Networks (VPNs), web browsing, email access, audio and video Streaming, content download of ring tones, video clips and File Transfer Protocol (FTP) [11]. These applications require higher throughput, wider bandwidth, smaller delay and innovative transmission methods which will give higher spectral efficiency and good quality [2]. Therefore, WiMAX and LTE wireless communication technologies are considered as candidates to achieve the 4G requirements announced by International Telecommunication Radio Communication Sector (ITU-R) which is known as International Mobile Telecommunication-Advanced (IMT-Advanced) [2]. Figure 1 shows the geographical locations of the deployment of 2G, 3G and 4G.

2.4.1 WiMAX

WiMAX (IEEE 802.16) is a telecommunication mobile system designed to provide high speed broadband wireless access which is a probable replacement candidate for mobile system (e.g., GSM) or can be used as an overlay to enhance capacity [12]. There are many versions of WiMAX (IEEE 802.16) standards. The IEEE 802.16d (802.16-2004) provides fixed WiMAX network while IEEE 802.16e (802.16-2005) is an amendment to 802.16-2004 and it is directed to support for mobility; therefore, also known as "Mobile WiMAX" [12]. The WiMAX revision IEEE 802.16m expected to offer peak rates of at least 1 Gbps fixed speed and 100 Mbps to MUs [13].

2.4.2 LTE

3GPP's LTE standard evolved from the high speed packet access cellular standards. LTE is a telecommunication mobile system designed to provide higher data rate, higher throughput and lower air-interface latency compared with 2G and 3G systems [14]. This higher performance makes it possible to enhance the broadband data on demanding applications beyond web browsing and voice which require higher data rate and stricter QoS constraints such as video service [14].

2.5 5G

5G is defined as upcoming mobile system beyond 4G (B4G) which provides substantial features compared to the current mobile systems [15, 16]:

- Better coverage area.
- Higher data rate (around 1Gbps).
- Lower battery consumption.
- Higher security.
- Better spectral efficiency and Energy efficiency.
- Availability of Artificial Intelligence inspired applications.
- Not harmful for human health.
- Economic services due to low deployment cost.
- It has been concluded in [16] that "the final success of 5G will depend upon when it is fully implemented and the new services and contents made available to MUs". Figure 2 shows the geographical locations of the deployment of 2G, 3G, 4G LTE and 5G.

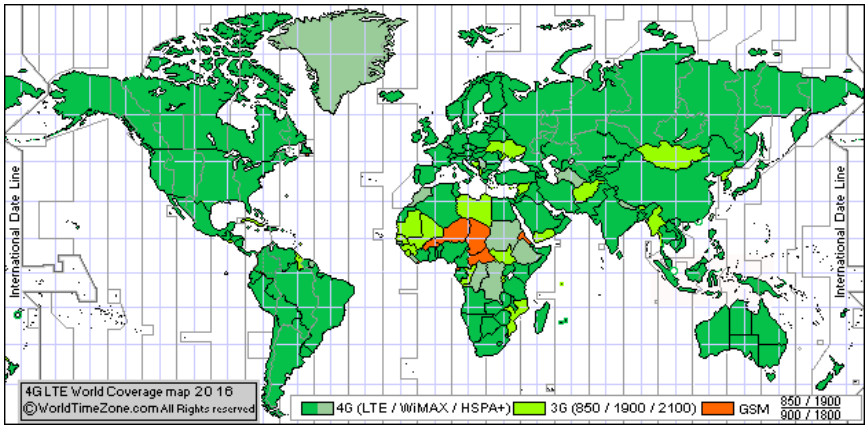


Figure 1. 2G, 3G and 4G world coverage map [17]

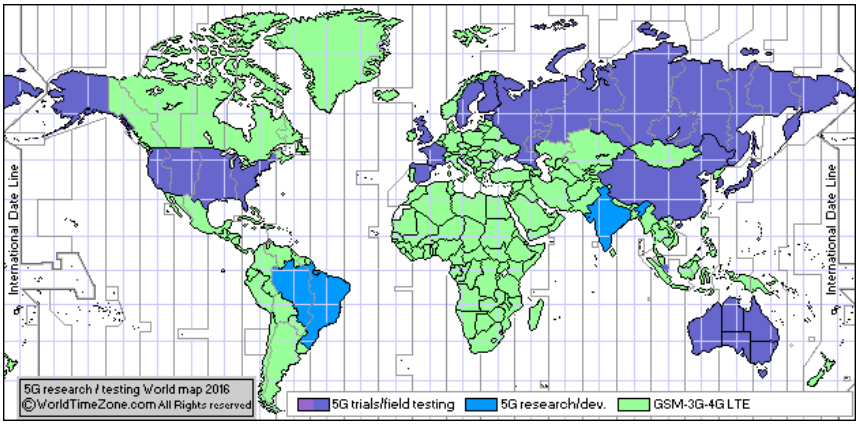


Figure 2. 2G, 3G, 4G LTE and 5G world coverage map [18]

Table 1. Advantages and disadvantages for UMTS, Wi-Fi, WiMAX, LTE and 5G [1, 12, 14-16, 19-27]

Access Technology	Type	Advantages	Disadvantages
UMTS (3G, Wide Area Network (WAN))	3GPP	▪ Wide coverage area. ▪ High security.	▪ Not suitable small, indoor and densely populated area. ▪ High service cost. ▪ High deployment cost. ▪ Low medium data rate from 144 Kbps to 2 Mbps.
Wi-Fi (Wireless Local Area Network (WLAN), IEEE 802.11, Local Area Network (LAN))	Non-3GPP	▪ Cheap service cost. ▪ Low deployment cost. ▪ Support rates from 1 Mbps to 54 Mbps.	▪ Limited in large space mobility. ▪ Weak security.
WiMAX (Metropolitan Area Network (MAN), IEEE 802.16, 4G)	Non-3GPP	▪ Medium coverage area. ▪ Medium service cost. ▪ Medium deployment cost. ▪ Medium security. ▪ Scalability. ▪ The current WiMAX revision IEEE 802.16m expected to offer peak rates of at least 1 Gbps fixed speed and 100 Mbps to MUs.	▪ Limited in large space mobility.
LTE (E-UTRAN, 4G)	3GPP	▪ Wide coverage area. ▪ High security. ▪ High throughput. ▪ Low air interference latency compared with 2G/3G systems. ▪ As set by ITU for IMT-Advanced: increased peak data rate, DL 3 Gbps and UL 1.5 Gbps (LTE-Advanced).	▪ High service cost. ▪ High deployment cost.
5G (B4G)	3GPP	▪ Better coverage area. ▪ Higher data rate (up to 1Gbps). ▪ Lower battery consumption. ▪ Higher security. ▪ Better spectral efficiency and Energy efficiency. ▪ Not harmful for human health. ▪ Economic services due to low deployment cost. ▪ Availability of Artificial Intelligence inspired applications.	The final success of 5G will depend upon when it is fully implemented and the new services and contents made available to MUs.

2. What are heterogeneous wireless communication technologies?

The growing demand for services (e.g., web browsing, file downloading and e-mail) from MUs anywhere, anytime is on the increase regardless of the technological constraints which are associated with different types of RATs such as UMTS, WiMAX and LTE, besides, there is no single RAT is able to satisfy the requirements for all different wireless communications scenarios. Therefore, the telecommunication operators are required to develop an interoperability strategy for these different types of existing networks to get the best connection anywhere, anytime between heterogeneous wireless communication technologies [28].

3. Who needs heterogeneous wireless communication technologies?

There are two main parties that need heterogeneous wireless communication technologies; the first one is the operator and the second is the MUs. The operators always seek to improve the final user experience and optimum use of the network by making a transition from the source network to target network as transparent as possible. The thing which will be reflected positively on operators to get more subscribers (users' loyalty) and more profit eventually; this is shown in Fig. 3. On the

other side, the MUs need to maintain network capability anywhere, anytime without interruption on their ongoing sessions.

Figure 3. Operators' vsion of using heterogeneous wireless communication technologies communication technologies

4. Why are heterogeneous wireless communication technologies necessary?

3GPP and non-3GPP include multiple integrated mobile systems and wireless networks and all of them coexist in a heterogeneous wireless access environment. At the same time each RAT has its advantages and disadvantages. Therefore, the complementarity between RATs is still required due to their characteristics. For example, the integration between WiMAX and LTE would satisfy MUs' demands to ongoing their sessions without noticeable degradation. Consequently, it would allow the service provider to get more profit.

5. What is the handover management within heterogeneous wireless communication technologies?

Handover management is a process which allows the MUs to continue their ongoing sessions when moving within the same RAT coverage areas or traversing different RATs. In heterogeneous wireless communication technologies, the handover management is crucial because RATs typically differ in terms of multiple parameters such as RSS, data rate, reliability, service cost, security, power consumption requirements, coverage area and latency. Therefore, complementarity to these RATs through VHO interworking architectures is essential to provide ubiquitous wireless access ability with the best available access network which suits the MU's requirements (e.g., high coverage area, high data rate and low cost).

3 Classifications for VHO Approaches of 3GPP and non-3GPP

This section presents VHO approaches of 3GPP and non-3GPP proposed in the literature and classifies them into two categories based on 4G and 5G for which their characteristics have been discussed. We identify the two categories as: (A) 4G based category which includes 4G and/or the rest of 3GPP previous generation mobile systems (2G-3G) and/or non-3GPP wireless networks. (B) 5G based category which includes 5G and/or the rest of 3GPP previous generation mobile systems (2G-4G) and/or non-3GPP wireless networks.

3.1 4G Category

In this category, plenty of VHO approaches have been proposed in the literature. In [29], [30] and [31], seventeen, fifteen and ninety nine VHO approaches have been surveyed, respectively. It has been noticed in [29-31] that the VHO approaches are mostly in the practical where the evolution

methods in these surveys are various between real environment, testbed, simulation experiment and analytical modeling.

3.2 5G Category

Although one of the most important goals of 5G is to provide ubiquitous wireless access abilities [15], it would not be perfectly able to achieve the goal without cooperating with the rest of wireless communication technologies of 3GPP and non-3GPP. To the best of my knowledge the current research works of this category are mainly confined in 5G where the recent overview of VHO in 4G and 5G has showed that in [32].

4 Comparison for VHO Approaches of 3GPP and non-3GPP

Section III has presented two categories of VHO approaches based on 4G and 5G for which their characteristics have been discussed. 4G Category has substantially presented and evaluated its VHO approaches using various evaluations methods. As the final success of 5G will depend upon when it is fully implemented and the new services and contents made available to MUs, it would be preferable to design and develop scenarios of 5G Category for evaluating real-world deployments, testbed, simulation experiment and analytical modeling compared with the huge number of previous works in 4G Category.

5 Conclusion

This paper has presented a study for VHO approaches of 3GPP and non-3GPP proposed in the literature and classified them into two categories based on 4G and 5G for which their characteristics have been discussed. It has been concluded that the 5G Category should be an active area of research compared with 4G Category which has obviously succeeded in presenting and evaluating plenty of VHO approaches.

REFERENCES

[1] Haji, A.; Ben Letaifa, A.; Tabbane, S.;, “Integration of WLAN, UMTS and WiMAX in 4G,” 16th International Conference Electronics, Circuits, and Systems 2009 (ICECS 2009), 13-16 Dec 2009, pp. 307-310.

[2] Torad, M.; El Qassas, A.; Al Henawi, H.;, “Comparison between LTE and WiMAX Based on System Level Simulation Using OPNET Modeler (Release 16),” 28th National Radio Science Conference 2011 (NRSC 2011), 26-28 Apr 2011, pp. 1-9.

[3] Guifen, G.; Guili, P.;, “The Survey of GSM Wireless Communication System,” International Conference on Computer and Information Application 2010 (ICCIA 2010), 3-5 Dec 2010, pp. 121-124.

[4] Kazemi, R.; Mosayebi, R.; Etemadi, S.M.; Boloursaz, M.; Behnia, F.;, “A Lower Capacity Bound of Secure End to End Data Transmission via GSM Network,” 6th International Symposium on Telecommunications 2012 (IST 2012), 6-8 Nov 2012, pp. 1015-1020.

[5] Tae, S.K., Soo, W.K.;, “A Method for Reducing a Bumblebee Noise Generated by a GSM Technology in a Smartphone,” 12th International Conference on Intelligent Systems Design and Applications 2012 (ISDA 2012), 27-29 Nov 2012, pp. 927-930.

[6] Nkansah-Gyekye, Y.; Agbinya, J.I.;, “A Vertical Handoff Decision Algorithm for Next Generation Wireless Networks,” 3rd International Conference on Broadband Communications, Information Technology & Biomedical Applications, 23-26 Nov 2008, pp. 358-364.

[7] Yi, B.L.; Yieh, R.H.; Yuan, KC.; Imrich, C.;, "Mobility Management: from GPRS to UMTS," Wireless Communications and Mobile Computing, vol. 1, no. 4, 2001, pp. 339-359.

[8] Darnbrough, J.;, "Market Aspects of UMTS," IEE Colloquium on Personal Communications in the 21st Century (I), 5 Feb 1998, pp. 1-26.

[9] Rabbani, M.G.; Kamruzzaman, J.; Gondal, I.; Ahmad, I.;, "A new Resource Distribution Model for Improved QoS in an Integrated WiMAX/WiFi Architecture," 7th International Wireless Communications and Mobile Computing Conference 2011 (IWCMC 2011), 4-8 Jul 2011, pp. 266-271.

[10] Datar, R.V.;, "WiFi and WiMAX-Break through in Wireless Access Technologies," IET International Conference on Wireless, Mobile and Multimedia Networks, 11-12 Jan 2008, pp. 141-145.

[11] David, M.S.; Jose, F.M.; Jorge, C.P.; Daniel, C.; Salvador, G.; Narc´ıs, C.;, "On the Way Towards Fourth-Generation Mobile:3GPP LTE and LTE-Advanced," Journal on Wireless Communications and Networking, vol. 2009, no. 4, 2009, pp. 1-10.

[12] Sengar, S.S.; Tyagi, N.; Singh, A.P.;, "A Survey on WiMAX-3G interworking," 3rd International Conferenceon Communication Software and Networks 2011 (ICCSN 2011), 27-29 May 2011, pp. 54-58.

[13] Ahmadi, S.;, "An Overview of Next-Generation Mobile WiMAX Technology," IEEE Communication Magazine, vol. 47, no. 6, Jun 2009, pp. 84-98.

[14] Talukdar, A.; Mondal, B.; Cudak, M.; Ghosh, A.; Fan, Wang.;, "Streaming Video Capacity Comparisons of Multi-Antenna LTE Systems," 71st Vehicular Technology Conference 2010 (VTC 2010-Spring), 16-19 May 2010, pp. 1-5.

[15] Abdullah M.; Sultan, A.; Hassan, A.;, "4G and 5G Mobile Communication Networks: Features Analysis, Comparison and Proposed Architecture," International Journal of Computer Science and Technology, vol. 7, no. 2, Jun 2016, pp. 154-160.

[16] Arun, A.; Gourav, M.; Kabita A.;, "The 5th Generation Mobile Wireless Networks- Key Concepts, Network Architecture and Challenges," American Journal of Electrical and Electronic Engineering, vol. 3, no. 2, Mar 2015, pp. 22-28.

[17] 4G LTE World Coverage Map. (2017). WorldTimeZone. Retrieved 16 Apr, 2017, from https://www.worldtimezone.com/4g.html.

[18] 5G Field Testing, Trials, Research, Development World Coverage Map. (2017). WorldTimeZone. Retrieved 16 Apr, 2017, from https://www.worldtimezone.com/5g.html.

[19] Akkari, N.; Tohme, S.;, "Introducing Intelligent Vertical Handover in Next Generation Networks," International Conference Signal Processing and Communications 2007 (ICSPC 2007), 24-27 Nov 2007, pp. 728-731.

[20] Chang, J.M.; Abichar, Z.; Chau-Yun, H.;, "WiMAX or LTE: Who Will Lead the Broadband Mobile Internet?," IT Professional, vol. 12, no. 3, May-Jun 2010, pp. 26-32.

[21] Seddigh, N.; Nandy, B.; Makkar, R.; Beaumont, J.F.;, "Security Advances and Challenges in 4G Wireless Networks," 8th Annual International Conference on Privacy Security and Trust 2010 (PST 2010), 17-19 Aug 2010, pp. 62-71.

[22] 3GPP TR 25.814 v7.1.0, "Physical Layer Aspects for Evolved Universal Terrestrial Radio Access (UTRA) Release 7".

[23] Khan, M.M.A.; Ismail, M.F.; Dimyati, K.;, "Seamless Handover between WiMAX and UMTS," 9th Malaysia International Conference on Communications 2009 (MICC 2009), 15-17 Dec 2009 pp. 826-830.

[24] Alkhayat, I.; Kumar, A.; Elmaghraby, A.;, "Seamless Connectivity Scheme for Heterogeneous Wireless Networks," International Symposium on Signal Processing and Information Technology 2008 (ISSPIT 2008), 16-19 Dec 2008, pp. 363-368.

[25] Ben-Jye, C.; Jun-Fu, C.;, "Cross-Layer-Based Adaptive Vertical Handoff With Predictive RSS in Heterogeneous Wireless Networks," IEEE Transactions on Vehicular Technology, vol. 57, no. 6, Nov 2008, pp. 3679-3692.

[26] [26] The Mobile Broadband Standard. (Jun/2013). 3rd Generation Partnership Project (3GPP). Retrieved 27 May, 2014, from http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced.

[27] Fangmin, X.; Luyong, Z.; Zheng, Z.;, "Interworking of Wimax and 3GPP Networks Based on IMS [IP Multimedia Systems (IMS) Infrastructure and Services]," IEEE Communications Magazine, vol. 45, no. 3, Mar 2007, pp. 144-150.

[28] Angoma, B.; Erradi, M.; Benkaouz, Y.; Berqia, A.; Akalay, M.C.;, "HaVe-2W3G: A Vertical Handoff Solution between WLAN, WiMAX and 3G Networks," 7th International Wireless Communications and Mobile Computing Conference 2011 (IWCMC 2011), 4-8 Jul 2011, pp. 101-106.

[29] 9] Khattab, O.; Alani, O.;, "A Survey on Media Independent Handover (MIH) and IP Multimedia Subsystem (IMS) in Heterogeneous Wireless Networks," International Journal of Wireless Information Networks (IJWIN), Springer, vol. 20, no. 2, Jun 2013, pp. 215-228.

[30] Khattab, O.; Alani, O.;, "A Survey on MIH vs. ANDSF: Who Will Lead the Seamless Vertical Handover Through Heterogeneous Networks?," International Journal of Future Generation Communication and Networking (IJFGCN), vol. 6, no. 4, Aug 2013, pp. 1-11.

[31] Ahmed, A.; Boulahia, L.; Gaiti, D.;, "Enabling Vertical Handover Decisions in Heterogeneous Wireless Networks: A State-of-the-Art and A Classification," IEEE Communications Surveys & Tutorials, vol. PP, no. 99, 2013, pp. 1-36.

[32] Feras, Z.; Suhaidi, H.; Adib, H.;, "Vertical Handover in Wireless Heterogeneous Networks," Journal of Telecommunication, Electronic and Computer Engineering, vol. 9, no. 1-2 , 2017, pp. 81-85.

SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 6, NO. 2
ISSN: 2054 -7420

End-to-End Service Delivery with QoS Guarantee in Software Defined Networks

Qiang Duan
Information Sciences & Technology Department, Pennsylvania State University, Abington USA
qduan@psu.edu

ABSTRACT

Software-Defined Network (SDN) is expected to have a significant impact on future networking. Although exciting progress has been made toward realizing SDN, application of this new networking paradigm in the future Internet to support end-to-end QoS provisioning faces some new challenges. The autonomous network domains coexisting in the Internet and the diverse user applications deployed upon the Internet call for a uniform Service Delivery Platform (SDP) that enables high-level network abstraction and inter-domain collaboration for end-to-end service provisioning. However, the currently available SDN technologies lack effective mechanisms for supporting such a platform. In this paper, we first present an SDP framework that applies the Network-as-a-Service (NaaS) principle to provide network abstraction and orchestration for end-to-end service provisioning in the SDN-based future Internet. Then we focus our study on two enabling technologies for such an SDP to achieve QoS guarantee; namely a network abstraction model and an end-to-end resource allocation scheme. Specifically, we propose a general model for abstracting the service capabilities offered by network domains and develop a technique for determining the required amounts of bandwidth in network domains for end-to-end service delivery with QoS guarantee. Both the analytical and numerical results obtained in this paper indicate that the NaaS-based SDP not only simplifies SDN service and resource management but also enhances bandwidth utilization for end-to-end QoS provisioning.

Keywords: Software-Defined Network (SDN), Service Delivery Platform (SDP), Network-as-a-Service (NaaS), QoS Provisioning.

1 Introduction

Software-Defined Network (SDN) is emerging network architecture that may have a significant impact on the development of future networking technologies. SDN architecture decouples network control and data forwarding functions; thus enabling network control to become directly programmable and underlying network infrastructure to be abstracted for applications [1]. Key features of SDN include separation between control plane and data plane, logically centralized network control, and programmability of the control plane. These features combined together gives SDN some great advantages in networking, including simplified and enhanced network configuration and operation, flexible and efficient network control and management, and improved network performance for meeting various application requirements. Therefore, SDN is expected to play a crucial role in the future Internet.

DOI: 10.14738/tnc.62.4373
Publication Date: 13th April 2018
URL: http://dx.doi.org/10.14738/tnc.62.4373

SDN architecture and its enabling technologies recently formed an important research area that has attracted extensive attention from both academia and industry. Active research topics in this area include SDN-enabled switching devices, SDN controllers, network operating systems, various network control/management applications, protocols between the data and control planes (southbound interface), and Application Programming Interfaces (APIs) for programming the control plane (northbound interface). Exciting progress has been made on SDN development and numerous research results have been reported in literature [2], [3], [4].

Although the SDN architecture has been successfully applied in some networking systems such as enterprise networks, data center networks, and inter-data center communications, adoption of this new networking paradigm in a large-scale internetworking scenario such as the future Internet faces new challenges that must be further investigated. One of the key issues lies in end-to-end service delivery across heterogeneous network domains with QoS guarantee for meeting diverse user requirements. In an enterprise or data center network, the user applications, network controller, and data forwarding devices all belong to the same administration domain; therefore, information of underlying network infrastructure can be made available to upper layer applications easily. However, in the Internet end users (computing applications) and network service providers often belong to different domains; therefore detailed information of network states may not be directly visible to applications. In addition, end-to-end communication paths in the Internet often traverse multiple autonomous systems operated by different organizations. End-to-end service provisioning in such a heterogeneous networking scenario requires a higher-level network abstraction for flexible interaction between users and service providers and loose-coupling collaboration among the involved autonomous systems. This calls for a service delivery platform that supports flexible and effective user-network interaction and inter-domain collaboration.

However, currently available SDN technologies lack an effective mechanism for building such a service delivery platform. Although a variety of SDN controllers have been developed, there is no standard yet for achieving interoperability between these controllers. What resulted is that no single vendor could deliver a standard-based northbound API for application development, or a standardized interface between controllers. In a large-scale inter-domain networking scenario, it is not feasible to require all autonomous network domains to adopt the same type of SDN controller. Therefore, lack of interoperability between SDN controllers prevents applications from functioning seamlessly across different controllers for inter-domain network service provisioning. Recent works on inter-domain networking in SDN mainly focused on distributed collaboration between SDN controllers for routing. End-to-end service delivery across heterogeneous SDN domains has not been sufficiently studied.

Recently, application of the service-orientation principle in SDN to address the challenging problem of end-to-end service delivery started attracting researchers' attention. The Service-Oriented Architecture (SOA) [5] offers an effective mechanism to enable flexible interactions among autonomous systems to meet diverse service requirements. SOA has been widely adopted in various areas, including Cloud computing and Web services, as the main model for service delivery. Application of the SOA principle in networking leads to a Network-as-a-Service (NaaS) paradigm, which enables networking resources and functionalities to be utilized by users as services through a standard abstract interface, much like computational resources are utilized as services in Cloud computing. NaaS enables abstraction of networking systems into network services that can be discovered, selected, and accessed by users; thus offering a flexible mechanism for user-network interaction. Network services can also be orchestrated for

end-to-end service provisioning. Network abstraction enabled by NaaS also allows flexible collaboration among autonomous network domains via loose-coupling service interactions. Therefore, NaaS may greatly facilitate end-to-end service delivery in the future Internet.

The research work reported in this paper tackles the challenging problem of end-to-end service delivery in SDN by exploiting the NaaS notion. We first present a framework of a Service Delivery Platform (SDP) that applies the NaaS paradigm in SDN to enable high-level network abstraction and inter-domain network service orchestration. Then we focus our study on two enabling technologies for the SDP to provide end-to-end QoS guarantee; namely an abstraction model for network service capabilities and an end-to-end resource allocation scheme for performance guarantee. Specifically, we propose a general model for abstracting service capabilities of network domains, which is then applied to composite network services for modeling capability of end-to-end service delivery. Based on the service capability model for network abstraction, we develop a technology that can be employed at the SDP to determine the required amount of bandwidth for achieving end-to-end QoS guarantee. Bandwidth utilization of the SDP is then analyzed and the obtained results show that such an SDP with a global network view may improve bandwidth utilization for end-to-end QoS provisioning.

SDN brings in potential benefits for enhancing future networking from at least two aspects: i) simplifying network control and management and ii) enhancing service provisioning for meeting diverse application requirements. Although the first aspect has been explored by many efforts, the second aspect has received less attention. The proposed SDP framework and the relevant technologies developed in this paper aim to address this issue in order to fully realize the potential of the emerging SDN paradigm in the future Internet.

The rest of the paper is organized as follows. In Section II we discuss challenges to end-to-end service delivery in SDN and review related works. A framework of a NaaS-based SDP for end-to-end service delivery in SDN is presented in Section III. We propose a high-level abstraction model for network service capabilities and apply the model to end-to-end network services in Section IV. Then in Section V we develop a technique for determining required bandwidth to achieve end-to-end QoS guarantee and analyze bandwidth utilization achieved by the SDP with this technique. Numerical results are provided in Section VI. We draw conclusions in Section VII.

2 End-to-end Service Delivery in SDN – Challenges and Solutions

2.1 Challenges to End-to-End Service Delivery in SDN

Recent rapid advancement in SDN research has yielded diverse technologies for realizing this new network architecture. Various SDN-enable switches have been developed. Although OpenFlow [6] has been widely adopted for controlling switches in the data plane, it is not the only southbound interface for

SDN. Possible protocols that may potentially play the same role include Forwarding and Control Element

Separation (ForCES) [7], Path Computation Element Communication Protocol (PCEP) [8], Protocol Oblivious Forwarding (POF) [9], and OpFlex [10]. Wide varieties of SDN controllers and network operating systems have also been developed. These include both centralized controllers such as NOX [11], Beacon [12], and Floodlight [13], and distributed network operating systems such as ONIX [14], ONOS [15], and HyperFlow [16].

Diversity in available SDN technologies brings in challenges to end-to-end service provisioning across multiple domains in SDN-based future Internet. Autonomous systems in the Internet should have the freedom to employ various SDN technologies, including switches, southbound protocols, and network controllers, that fit their particular networking needs. On the other hand, the objective of service provisioning is to deliver network services across the heterogeneous domains for meeting the diverse requirement of end users. Therefore, end-to-end service provisioning in the future Internet requires not only effective inter-domain collaboration but also flexible interaction between upper layer user applications and the underlying network domains.

However, the currently available SDN technologies lack sufficient capability of meeting this requirement for end-to-end service delivery. Development of network controllers often lacks consideration of interoperability with controllers from other vendors. Distributed network controllers mainly focus on cooperation among multiple homogeneous controllers in the same domain; thus are insufficient to handle heterogeneity of the controllers in multi-domain cases. Moreover, despite rapid development on standard southbound interface, currently there is no common standard for the northbound API between SDN controllers and network control/management applications. These applications are often developed based on the API provided by a particular type of controller; thus are tightly coupled with the controller design. Such tight coupling between applications and controllers significantly limits the capability of service provisioning over heterogeneous controllers in a multi-domain SDN environment.

Recently some study on inter-domain issues in SDN has been reported in the literature. SDNi [17] is a protocol recently proposed by IETF for coordinating operations and exchanging information between SDN controllers in different domains. The implementation of SDNi suggested in [17] is to extend BGP for information exchange. However, the hop-by-hop nature of BGP makes routing among domains in a decentralized manner without knowledge of end-to-end routes, which may not be able to achieve a global optimal path for end-to-end QoS provisioning. Research reported in [18] and [19] employs the SDN principle to address the inter-domain routing problem. Both works are based on BGP; thus are limited by its decentralized feature to fully realize the SDN benefit of centralized control with a global network view. The inter-AS routing proposed in [18] assumes that homogeneous controllers, specifically the NOX-OpenFlow controller, are used in all domains; thus may not be applicable to large-scale multi-domain scenarios. The multi-AS routing control platform proposed in [19] assumes the existence of a mechanism to communicate with SDN domain controllers without detailed discussion on the realization of such a mechanism.

In [20] the authors argue that BGP is a poor candidate for inter-domain routing in SDN and propose decoupling between routing and policy control to facilitate interoperability among SDN domains. The distributed control plan proposed in [21] employs a message-oriented communication bus for information exchange among SDN domain controllers. The aforementioned research focuses on controller collaboration for inter-domain routing in SDN. End-to-end service provisioning needs more than just routing across multiple domains. Flexible interaction between user applications and the SDN controllers in different domains of the underlying network infrastructure is another important aspect that so far has received little attention. It requires a high-level network abstraction, loose-coupling interaction between applications and controllers, and flexible collaboration among heterogeneous controllers.

2.2 Network-as-a-Service in SDN – a Promising Solution

The Service-Oriented Architecture (SOA) [5] offers a promising approach to addressing the challenges for end-to-end service provisioning in multi-domain SDN. The SOA can be described as architecture within which all functions are defined as independent services with invokable interfaces that can be called in defined sequences to form business processes. A *service* in SOA is a module that is self-contained (i.e., the service maintains its own states) and platform-independent (i.e., the interface to the service is independent with its implementation platform). Services can be described, published, located, orchestrated, and programmed through standard interfaces and messaging protocols. A key feature of SOA is "loose-coupling" interaction among heterogeneous systems, which allows entities to collaborate with each other while keep themselves independent. This feature makes SOA very effective architecture for coordinating heterogeneous systems to provide services that meet various application requirements.

Application of the service-orientation principle in networking provides a promising approach to addressing some challenges in the future Internet. Such a service-oriented networking paradigm is referred to as *Network-as-a-Service* (NaaS), in which networking resources are abstracted and utilized in form of SOA-compliant network services. In principle, a network service may represent any type of networking component at different levels, including an entire network domain, a single physical or virtual network, or an individual network node. Multiple network services can be combined into one composite inter-network service through a service orchestration mechanism.

Recently the NaaS paradigm has started attracting attention from the networking research community and interesting progress has been reported in the literature. Costa *et al.* proposed a NaaS model for data center networks in [22] for enabling Cloud tenants to have direct access to network infrastructure for improving service performance. Cloud-based network architecture that combines the Cloud service model with the network openness enabled by SDN was proposed in [23] in order to offer various network protocol services. An SDN control platform called Meridian was presented in [24], which provides a service-level network model with connectivity and policy abstractions for Cloud networking. Bueno and his colleagues developed a NaaS-based Network Control Layer (NCL) that provides an abstraction layer to obtain homogeneous control over heterogeneous network infrastructure [25].

The above works made interesting progress of applying NaaS in SDN for network service provisioning; however, they mainly focus on single-domain cases or assume homogeneous SDN controllers. The framework proposed in [22] assumes that applications can directly acquire detailed knowledge of underlying network infrastructure, which is reasonable in a single data center environment but not realistic for the large scale Internet with multiple autonomous domains. The prototype given in [23] for realizing the proposed network architecture used NOX controller and OpenFlow protocol for controlling all switches. Both Meridian platform and NCL were implemented based only on Floodlight controller. Cooperation between SDN domains with heterogeneous controllers for end-to-end service delivery is still an opening issue that has not been sufficiently addressed yet.

In order to address this important challenging issue, preliminary study of applying NaaS in SDN to support end-to-end QoS provisioning was presented in our previous work [26]. In this paper, we further develop the idea of NaaS-SDN integration to propose a framework of a NaaS-based Service Delivery Platform (SDP) for a multi-domain SDN environment. This platform provides a high-level abstraction of each SDN domain as a network service and enables network service orchestration for end-to-end service delivery. Then we

particularly investigate two key technologies for achieving end-to-end QoS guarantee through this SDP – an abstract model for network service capabilities and a technique for end-to-end bandwidth allocation. Our analysis results also indicate that end-to-end service delivery enabled by the SDP with a global network view improves resource utilization for QoS provisioning.

The research reported in [27] and [28] shares some similar ideas with the work presented in this paper. Zhu *et al.* proposed Software Service Defined Network (SSDN) architecture in [27], which employs SOA-based Enterprise Service Bus (EBS) to build a network software service layer that allows networking resources in multiple domains with different SDN controllers to be federated for end-to-end service delivery. However, some key technologies for achieving QoS guarantee with such a service layer, for example abstraction of service capabilities and cross-domain resource allocation, were not addressed in [27]. The authors of [28] developed a distributed QoS architecture for SDN, which employs a hierarchical control plane where a super controller coordinates the local SDN controllers in multiple domains to support end-to-end multimedia streaming. However, [28] did not give any specific mechanism for the super controller to coordinate heterogeneous SDN controllers in different network domains for service delivery. On the other hand, the research of [27], [28] and our work reported in this paper may complement each other. The ESB-based mechanism described in [27] may be applied to implement communications among the SDP, domain controllers, and end user applications in the framework proposed in this paper. Our SDP framework offers an approach to realizing the hierarchical control plane presented in [28].

3 A NaaS-Based Service Delivery Platform in SDN

The framework of a NaaS-based Service Delivery Platform (SDP) in a multi-domain SDN environment is shown in Figure 1. In this framework, each network domain may have its own choice of SDN technologies, including data plane switches, SDN controllers, and the southbound interface. A domain may also implement various control programs upon its own SDN controller to perform functions such as QoS routing and traffic engineering within the domain scope. Each network domain is abstracted as a network service through a NaaS interface, which provides a high-level abstraction of networking capabilities of the entire domain, including both forwarding and control functionalities, to the SDP. The NaaS interface also allows the SDP to specify its networking requests and policies to each domain. The NaaS-based network abstraction makes network infrastructure of each domain transparent to upper layer applications; thus enabling SDP to coordinate the resources provided by network domains for delivering network services to support diverse user applications.

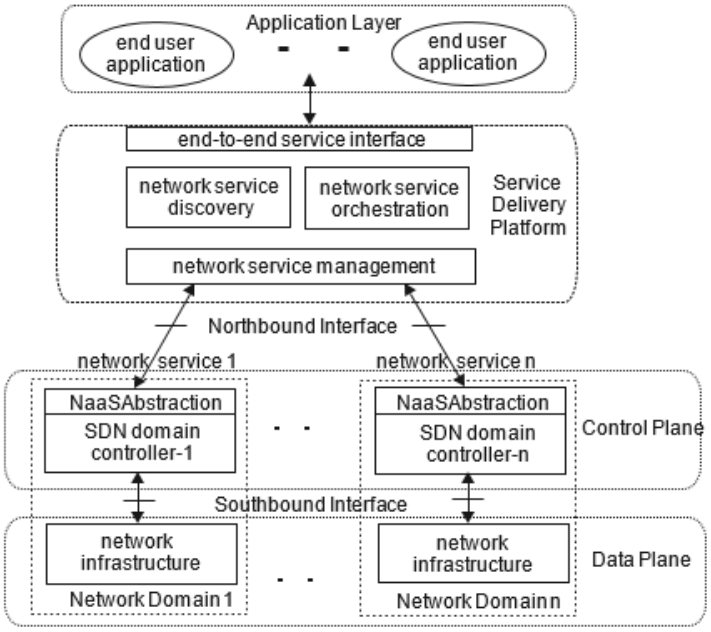


Figure 1. NaaS-based Service Delivery Platform in Software-Defined Network

The SDP serves as a middleware between upper layer user applications and the underlying network infrastructure consisting of heterogeneous domains. Key components of the SDP include a service interface and the modules for service management, service discovery, and service orchestration. The SDN controller of each network domain is responsible to publish and update an abstract model of the domain service capability at the service management module. The service interface allows upper layer user applications to specify their requests for end-to-end network services. Upon receiving a service request from an end user, the service discovery module searches the service registry maintained by the service management module to discover a network service for meeting the request. If no single network service provided by any individual domain can meet the requirement, the orchestration module will search for a service chain of multiple network services and orchestrate them for end-to-end service delivery. Then the service management module will send requests to the SDN controllers of all domains involved in service delivery for this user to allocate sufficient bandwidth for meeting user QoS requirement. In addition to these key components, the SDP may also perform some global network management functions, for example user authentication, service request authorization, end-to-end path computation, and traffic engineering.

The proposed SDP framework combines advantages of NaaS and SDN for improving end-to-end service provisioning in the future Internet. The separated data/control planes and logically centralized controlling enabled by SDN allows a global control mechanism over heterogeneous network infrastructure. NaaS provides a high-level abstraction of autonomous networking systems and enables loose-coupling collaboration among them. The proposed NaaS-based SDP offers a uniform platform upon which third party service providers can develop and deploy new end-to-end network services to meet various application requirements without knowing detailed implementations of underlying network infrastructure. Such a service delivery platform enables a new business model in which a service provider can lease networking resources from various domains and orchestrate the resources for end-to-end network service provisioning. Such a business model is similar to the model for Cloud service provisioning,

which allows service providers to lease computing resources from infrastructure providers for offering Cloud services to end users.

The proposed SDP also offers a promising approach toward federated management of networking and computing resources (such as CPU capacity and storage space in Clouds) to enable converged network and Cloud service provisioning in a Software Defined Environment. In such an environment, both networking and computing resources can be abstracted as services by following a uniform SOA-based mechanism, and then can be orchestrated to form composite network-Cloud services to end users. The NaaS-based

SDP also supports incremental SDN deployment in the Internet. Network domains implemented with non-SDN technologies can also be exposed as network services to the SDP, as long as they realize a NaaS interface for network abstraction, and then can be involved in end-to-end service delivery with SDN domains through service orchestration provided by the SDP.

Another important advantage of the proposed SDP is to realize the benefit of logically centralized control promised by the SDN paradigm in large-scale multi-domain networking environments. Such a centralized control with a global network view is particularly important for achieving end-to-end service delivery with QoS guarantee in the Internet consisting of various autonomous systems. Due to the heterogeneity of network protocols and technologies in these systems, exposure of networking capabilities to a central control unit without appropriate abstraction would lead to unmanageable complexity. The high-level abstraction enabled by the SDP addresses the diversity challenge; thus making centralized control for end-to-end QoS provisioning possible.

The presented framework gives functional architecture for a service delivery platform for inter-domain SDN, which may be realized with various implementations. Enabling technologies are required for implementing key functions in two categories: i) internal modules of the SDP, mainly including the service management, discovery, and orchestration modules; and ii) interfaces for the SDP to interact with user applications and network domains, including the service interface and the network abstraction interface. Recent research on NaaS has yielded various technologies for network service description, discovery, and composition. A summary of these technologies can be found in the survey paper [29]. These technologies form the foundation for implementing the key modules in the SDP. Standard interfaces for network and service abstractions form the other key aspect for realizing the SDP. From an end user's perspective, the SDP plays the role of a service broker in the SOA architecture; therefore, standard Web Service interfaces between service consumers and a service broker can be applied to realize the service interface between the SDP and the upper layer user applications. The network abstraction interface between the SDP and various network domains is essentially an SDN northbound interface. RESTful Web Service has been widely adopted for implementing a northbound interface in SDN. Application Layer Traffic Optimization (ALTO) [30] and Interface to Routing System (I2RS) [31], are two RESTful compatible protocols based on which a network abstraction interface may be realized.

A key for the NaaS-based SDP to achieve end-to-end service delivery with QoS guarantee in a multi-domain SDN environment is to discover, select, and orchestrate the appropriate network services with sufficient service capabilities that meet the performance requirements specified by end users. In order to achieve this objective, each network domain should provide the SDP with a high-level abstraction of its service capability information. In addition, the SDP should be able to determine the minimum service

capacity required for achieving end-to-end QoS guarantee. Therefore, an abstract model of network service capabilities and a method for determining required service capacity are two key enabling technologies for the proposed SDP, which are the focus of study for the rest of this article.

4 High-Level Abstraction Model for Network Service Capability

The SDP needs information about service capabilities of network domains in order to achieve end-to-end QoS provisioning. On the other hand, NaaS-based network abstraction requires hiding detailed information of network infrastructure. To balance the conflicting requirements from these two aspects, in this section we propose a high-level abstraction model of network service capabilities, which allows SDN domain controllers to provide the SDP with necessary information without exposing details of network infrastructure. Such a model should meet the following requirements in order to support end-to-end QoS provisioning in a large-scale multi-domain SDN environment: i) providing a high-level abstraction of topology and states of underlying network infrastructure, ii) presenting information about network capabilities required by the SDP for end-to-end QoS provisioning, iii) being agnostic to network implementations thus applicable to heterogeneous network domains, and iv) being extendable to model capabilities of composite network services for inter-domain service delivery.

4.1 Abstraction Model for Single Network Service Capability

We first consider modeling capabilities of single network services that virtualize the networking functionalities of individual network domains. In general, the service capability information about an individual network domain needed by the SDP for end-to-end QoS provisioning can be described at a high level from the following two aspects: *virtual connections* provided by the network service among the border nodes of the domain, and *capacity* of data transportation on each virtual connection. From a service provisioning perspective, topology of a network domain may be abstracted as a full mesh of virtual connections between any pair of border nodes of the domain. The SDP just needs to know if a network service provides a virtual connection from an ingress node to an egress node, and if so how much data transport capacity is available on the virtual connection. The actual path between the nodes is determined by the SDN controller in that domain, which has knowledge of the physical topology and network states of the entire network domain.

Therefore, for a network service that virtualizes a domain with n border nodes, a high-level abstraction of its service capability can be modeled by a matrix

$$\boldsymbol{C} = \begin{pmatrix} c_{1,1}\ c_{1,2} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{n,1}\ c_{n,2} & \cdots & c_{n,n} \end{pmatrix} \tag{1}$$

and each matrix element $c_{i,j}$ is defined as

$$c_{i,j} = \begin{cases} P_{i,j}, & \text{the network service provides virtual connection from node i to node j} \\ 0, & \text{otherwise} \end{cases}$$

where $P_{i,j}$ is called the *capacity profile* for the virtual connection from node i to node j, whose definition will be given later in this subsection. Each non-zero element $c_{i,j}$ in the matrix $\boldsymbol{C}$ indicates existence of a virtual connection from node i to node j and also describes the transport capacity available on the connection.

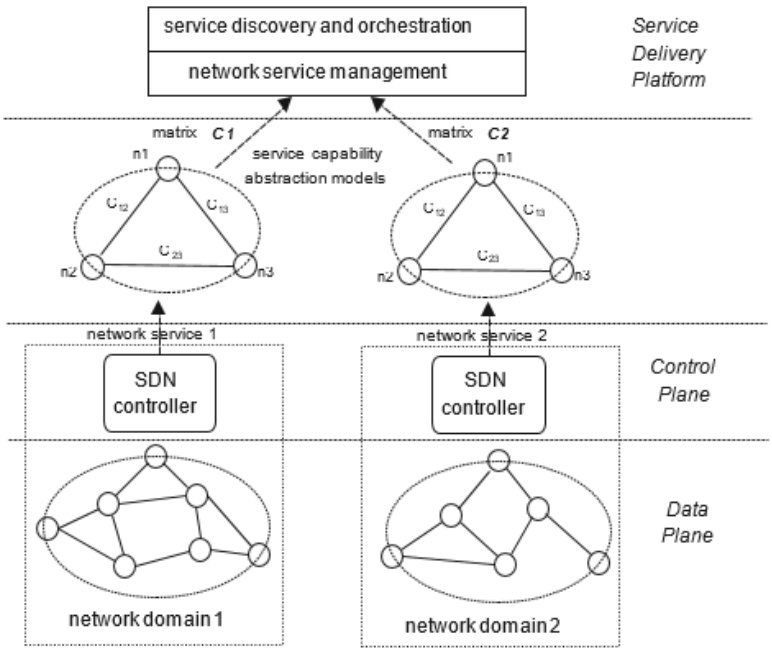


Figure 2. Abstraction of topology connectivity and transport capacity of network services

As illustrated by the example shown in Figure 2, physical topology of reach network domain is abstracted as a full mesh of virtual connections among border nodes of the domain. Service capability information of the domain is described by a matrix ***C*** presenting a set of virtual connections and the associated capacity profiles. Each SDN controller publishes the matrix ***C*** of its domain to the SDP service management module via the NaaS interface. Such a matrix exposes capability information of a network service to its potential users while keeping its implementation transparent to the users. Such a high-level abstraction of network domain internal topology and states is necessary for achieving scalability in disseminating, updating, and inquiring such information in a large scale inter-domain SDN environment; therefore meeting the requirement i) for an abstraction model.

In order to meet the requirement ii) for providing information needed for QoS provisioning, a profile $P_{i,j}$ is used as the value of each non-zero element $c_{i,j}$ of matrix ***C***. This profile describes the capacity that can be guaranteed by the network service for data transportation from node *i* to node *j*. Due to the wide variety of networking technologies employed in heterogeneous network domains, such a capacity profile must be independent to network implementations in order to meet the requirement iii). In addition, the profile should also be in a form that can be easily extended to describe the capacity of end-to-end service delivery across multiple domains; thus meeting the requirement iv). In order to develop a service capacity profile that meets all the above requirements, we employ the *service curve* concept from *network calculus* theory [32]. The service curve in network calculus is defined as follows.

Let $R(t)$ and $R^*(t)$ respectively be the accumulated amount of traffic that arrives at and departs from a system by time t. Given a non-negative, non-decreasing function, $S(\cdot)$, where $S(0) = 0$, we say that the system guarantees a *service curve* $S(\cdot)$ for the flow, if for any $t \geq 0$ in the busy period of the system,

$$R^*(t) \geq R(t) \otimes S(t) \tag{2}$$

where $\otimes$ denotes the convolution operator defined as $h(t) \otimes x(t) = \inf_s:0_{\leq s \leq t} \{h(t - s) + x(s)\}$.

Essentially a service curve of a networking system describes the minimum amount of service capacity guaranteed by the system. In our network abstraction model, we employ the service curve guaranteed by the network service for the virtual connection from node *i* to node *j* as the capacity profile $P_{i,j}$. Since a service curve is a general function for describing network service capacity, it is independent with network implementations thus applicable to model service capabilities of heterogeneous network domains.

In order to limit the overheads between domain controllers and the SDP for publishing and updating matrix $\boldsymbol{C}$, it is desirable to present a capacity profile with a simple data structure. Toward this end, we define a Latency-Rate capacity profile as follows. If a network service guarantees a virtual connection a service curve

$$\beta(r,\vartheta) = \max\{0, r(t-\vartheta)\} \tag{3}$$

then we say that the virtual connection has a Latency-Rate (LR) profile, where ϑ and r are respectively called the *latency* and *rate* parameters of the profile. A LR profile can serve as the capacity model for virtual connections provided by typical network domains. In order to achieve end-to-end QoS guarantee, the SDP requires each network domain involved in service delivery to provide a minimum bandwidth. Such a minimum bandwidth guarantee is described by the rate parameter r in the LR profile. Data transportation in a network domain experiences a fixed delay that is independent with traffic queuing behavior, for example signal propagation delay, link transmission delay, switch process delay, etc. The latency parameter ϑ of the LR profile is to characterize this part of fixed delay.

Please be advised that although the capacity profile is implementation agnostic, the profile for each virtual connection provided by a network domain is constructed by the SDN controller in the domain; therefore profile parameters are related to the implementation and control/management policies of the domain. For example, the physical path for a virtual connection from node *i* to node *j* is established by the SDN controller following the path computing policy of the domain. Then the available bandwidth on this path will be the service rate parameter $r_{i,j}$ in the capacity profile $P_{i,j}$. If there exists multiple physical paths from *i* to *j* and the domain policy allows parallel data delivery; then the controller may aggregate the available bandwidth on all the paths to get the service rate parameter $r_{i,j}$.

For a typical network domain where transport capacity of the virtual connection from any node *i* to any node *j* can be modeled by a LR profile $\beta(r_{i,j},\vartheta_{i,j})$, the matrix element $c_{i,j}$ can be presented by a simple data structure with two parameters $[r_{i,j},\vartheta_{i,j}]$. The abstraction model provides the key information of service capability needed by the SDP for QoS provisioning using a small set of parameters. Therefore, LR capacity profile reduces the communication overheads between domain controllers and the SDP for publishing and updating service capability information; thus improving system scalability.

4.2 Abstraction Model for Composite Network Service Capability

To achieve service delivery across network domains, the SDP orchestrates multiple network services to form a composite network service that provides an end-to-end virtual connection. Therefore the SDP also requires a model for abstracting end-to-end capabilities of composite network services. The proposed capability model for single network services can be extended for supporting network service composition.

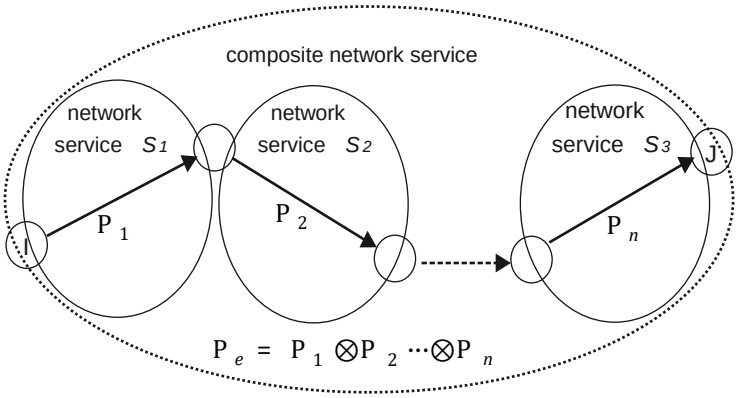


Figure 3. Capacity profile for a virtual connection provided by a composite network service

Known from network calculus, the service curve guaranteed by a series of tandem servers can be obtained through the convolution of all the service curves guaranteed by individual servers. Since the capacity profile of a virtual connection is essentially a service curve of the connection, the capacity profile of an end-to-end virtual connection traversing multiple domains can be determined by following the convolution theorem in network calculus. Suppose the source node *i* and the destination node *j* are in different domains, and the orchestration module selects *n* domains, which are abstracted by network services S_k, $k = 1,2,\cdots,n$ respectively, to form a composite service for providing a virtual connection from *i* to *j*, as shown in Figure 3. The connection from *i* to *j* consists of *n* virtual links, each is provided by a single network service. Suppose the capacity profile for the virtual link provided by service S_k is P_k, and the capability profile for the end-to-end virtual connection is denoted as P_e, then P_e can be determined as

$$P_e = P_1 \otimes P_2 \cdots \otimes P_n. \tag{4}$$

If each network service S_k guarantees a LR profile $\beta(r_k,\vartheta_k)$ for its virtual link, then it can be proved by following convolution theorem in network calculus that capacity profile for the end-to-end virtual connection is

$$P_e = \beta(r_e,\vartheta_e) = \beta(r_1,\vartheta_1) \otimes \cdots,\otimes\beta(r_n,\vartheta_n) \tag{5}$$

where $r_e = \min\{r_1,r_2,\cdots,r_n\}$ and $\vartheta_e = \vartheta_1 + \vartheta_2 + \cdots + \vartheta_n$.

Equation (5) implies that if the service capacity of each link of an end-to-end virtual connection can be described by a LR profile, then capacity of the virtual connection provided by a composite network service can also be modeled by a LR profile. The latency parameter of the end-to-end LR profile is equal to the summation of latency parameters of all links and the end-to-end service rate is limited by the bottleneck link with the least service rate value.

The proposed Matrix ***C*** and capacity profile provide a general abstraction model that can be used by SDN controllers in all network domains to publish service capability information of their network infrastructure at the SDP service management module. Publication and updating of the model could be implemented based on some available protocols, for example Application Layer Traffic Optimization (ALTO) [30]. Each SDN domain controller can implement an ALTO server that regularly disseminates a network-map and a cost-map to the SDP service management module, which can act as an ALTO client.

Matrix ***C*** and the associated capacity profiles can be presented as a network-map together with a cost map. The service management module then combines the network-maps and cost-maps of all domains

into a global network view that can be used by the service orchestration module for end-to-end service provisioning. ALTO supports RESTful Web service interface between servers and clients; thus supporting NaaS-based network abstraction interface between the SDP and SDN domain controllers.

5 Resource Allocation for End-to-end QoS Provisioning in SDN

End-to-end QoS provisioning in a multi-domain SDN environment requires allocation of sufficient networking resources in all the domains that are involved in service delivery for meeting user requirements. In the proposed SDP framework each domain is abstracted as a network service through a NaaS interface. Therefore, selecting and orchestrating the appropriate network services with sufficient data transport capacity for end-to-end service delivery is a key to QoS provisioning. The abstraction model developed in last section allows each domain to provide information about its service capability to the SDP, which forms the basis for network service selection and orchestration. For supporting QoS for an end user, the SDP also needs to determine the amount of service capacity required for meeting user performance requirement and assures that such capacity be allocated in each involved network domain. On the other hand, the SDP wants to minimize bandwidth consumption for each user in order to improve resource utilization in network infrastructure. Therefore, a method for determining the minimum amount of service capacity for meeting the end-to-end performance requirement specified by a user is an important technology needed by the SDP for QoS provisioning, which will be developed in this section.

5.1 Service Demand Profile

End users need to provide SDP with information about their networking demand in order for the SDP to select appropriate network services and determine required service capacity for meeting user requirements. In order to allow the wide variety of user applications to specify their diverse networking demands, we define a general demand profile $D\{C, Q, \mathrm{L}\}$. In this profile, element C gives the connectivity requirement, which can be specified by the addresses of source and destination for data transportation required by the application. The element Q in the profile is to specify QoS expectation for the service, which comprises a set of performance parameters such as the maximum delay and/or minimum throughput for data transportation. Since traffic flows with different load characteristics will require different amounts of service capacity for achieving a certain level of performance, we include a load descriptor L in the demand profile to characterize the traffic that a user application will load the network service. Considering the various user applications with diverse load characteristics, such a load descriptor must be general to support different types of traffic flows; while on the other hand be concise enough to be processed easily. The *arrival curve* concept in network calculus is employed here to develop a general load descriptor that meets such requirements.

Let $R(t)$ denote the accumulated amount of traffic arrives from a flow by time t. Given a non-decreasing, non-negative function, $\mathrm{L}(\cdot)$, the flow is said to have an arrival curve $\mathrm{L}(\cdot)$ if

$$R(t) - R(s) \le \mathrm{L}(t - s) \qquad \forall\, 0 < s \le t. \tag{6}$$

The arrival curve gives an upper bound for the amount of traffic that a user application can load a service delivery system; therefore is employed here as the load descriptor in a service demand profile. Since such a descriptor is defined as a general function of time, it can be used to describe the traffic load generated by any user application.

Currently most QoS-capable networks apply traffic regulation mechanisms at network boundaries to shape arrival traffic from end users. The traffic regulators most commonly used in practice are leaky buckets. A traffic flow constrained by a leaky bucket has a load profile

$$L(p,\rho,\sigma) = \min\{pt, \sigma + \rho t\} \quad (7)$$

where p, ρ, and σ are called respectively the peak rate, the sustained rate, and the burst size of the flow. Flow-based data forwarding in SDN data plane allows per-flow traffic regulation to be implemented easily at entry switches. Each user can specify its traffic load using a descriptor with p, ρ, and σ parameters, which may be enforced by a leaky bucket shaper at the SDN switch where user traffic enters the network. Therefore, it is reasonable to assume the availability of a leaky bucket load descriptor for the traffic flow of each user.

5.2 Minimum Capacity in Network Service for QoS Guarantee

Upon receiving the demand profile that a user submits with its service request, the SDP needs to determine the minimum service capacity required on a virtual connection for meeting the QoS expectation, which is the basis for network service selection and orchestration. In this subsection, we develop a technique for SDP to determine the minimum service capacity for meeting a given QoS requirement. We focus our analysis on the maximum delay and minimum throughput as performance parameters since they are required by most user applications with QoS expectation.

We first consider the case that a user application only requires the minimum throughput T_{req} as its QoS expectation; i.e., $Q = \{T_{req}\}$. Since throughput is the only QoS requirement, the minimum capacity C_{min} required on a virtual connection for supporting this user just needs to be T_{req}; that is, $C_{min} = T_{req}$. The capacity profile P of a virtual connection essentially gives a lower bound of the capacity that a network service guarantees to the connection. Following network calculus the minimum capacity available on the virtual connection can be determined as

$$b_{min} = \lim_{t\to\infty}[P/t] \quad (8)$$

Therefore, this virtual connection meets the user's requirement if $b_{min} \geq T_{req}$.

Suppose data transport capacity of a virtual connection can be modeled by a LR profile, i.e., $P = \beta(r,\vartheta)$, then

$$b_{min} = \lim_{t\to\infty}[r(t-\theta]/t = r. \quad (9)$$

If the virtual connection traverses n domains, then the end-to-end connection consists of n virtual links each modeled by a profile $\beta(r_i,\vartheta_i)$, $i = 1,2,\cdots,n$. According to (5) and (9), the end-to-end transport capacity $b_{min} = r_e = \min\{r1,r2,\cdots,r_n\}$. Therefore, the virtual connection meets the user's throughput requirement if $r_e \geq T_{req}$. Since throughput is the only QoS requirement, the minimum capacity C_{min} required on a virtual connection just needs to be T_{req}; that is, $C_{min} = T_{req}$.

Then we analyze the case that a user application only requires the maximum service delay D_{req} as its QoS expectation; i.e., $Q = \{D_{req}\}$. Consider a virtual connection selected for serving a user application, suppose the capacity profile of the connection is P and the traffic flow of this user has a load descriptor L, then network calculus shows that the maximum service delay guaranteed by the connection to this traffic flow is

$$d_{max} = sup_{t:t>0}\{inf\{\delta: \delta \geq 0\, L(t) \leq P(t+\delta)\}\}. \quad (10)$$

In order to determine the minimum service capacity C_{min} for meeting the requirement $d_{max} \leq D_{req}$, we apply the effective bandwidth concept in network calculus here. Considering a traffic flow with a cumulative arrival process $R(t)$ constrained by an arrival curve $L(t)$; for a fixed, but arbitrary delay requirement D_{req}, the effective bandwidth $R_e(D_{req})$ of the flow is defined as the minimum service rate required to serve the flow with $d_{max} \leq D_{req}$. Therefore, the effective bandwidth for the flow can be obtained as

$$R_e(D_{req}) = sup_{0\leq t_0 \leq t}\left\{\frac{R(t)-R(t_0)}{t-t_0+D_{req}}\right\} = sup_{s\geq 0}\left\{\frac{L(s)}{s+D_{req}}\right\} \quad (11)$$

If the maximum delay is the only QoS expectation of a user, then the minimum service capacity required by the user is the effective bandwidth of its traffic flow; that is, $C_{min} = R_e(D_{req})$.

Suppose a traffic flow having a leaky-bucket load descriptor $L(p,\rho,\sigma)$ is served by a virtual connection with a LR capacity profile $\beta = (r,\vartheta)$, then following (10) and (11) the effective bandwidth for meeting a delay requirement D_{req} can be determined as

$$R_e(D_{req}) = \begin{cases} \rho & D_{req} \geq D_{max} \\ \frac{p\sigma}{[(D_{req}-\theta)(p-\rho)+\sigma]} & D_{min} \leq D_{req} \leq D_{max} \\ not\ available & D_{req} < D_{min} \end{cases} \quad (12)$$

where $D_{min} = \vartheta$ and $D_{max} = \vartheta + \sigma/\rho$.

Equation (12) shows that if the expected delay upper-bound is greater than a threshold D_{max}, then the required service capacity is equal to the sustained rate ρ of the arrival traffic. Actually, service capacity on a virtual connection should be no less than the sustained rate of a flow in order for the connection to guarantee any upper bounded service delay for the flow. On the other hand, no delay expectation that is tighter than D_{min} can be met by a virtual connection with finite service capacity. This is because the underlying network infrastructure always introduces a certain amount of latency for data delivery due to its physical properties. For any expected delay upper bound between D_{min} and D_{max}, the required minimum service capacity is a function of traffic parameters (p,ρ,σ), the latency parameter ϑ, and the delay requirement D_{req}.

For an end user that has both minimum throughput T_{req} and maximum delay D_{req} as QoS expectation, the minimum service capacity that must be available on a virtual connection for providing QoS guarantee to the user will be $C_{min} = \max\{T_{req}, R_e(D_{req})\}$.

5.3 Network Service Selection and Orchestration for End-to-End QoS Provisioning

The technique for determining the minimum required service capacity can be employed by the SDP to perform network service selection and orchestration for achieving end-to-end QoS guarantee. A general procedure of network service selection/orchestration is illustrated in Figure 4.

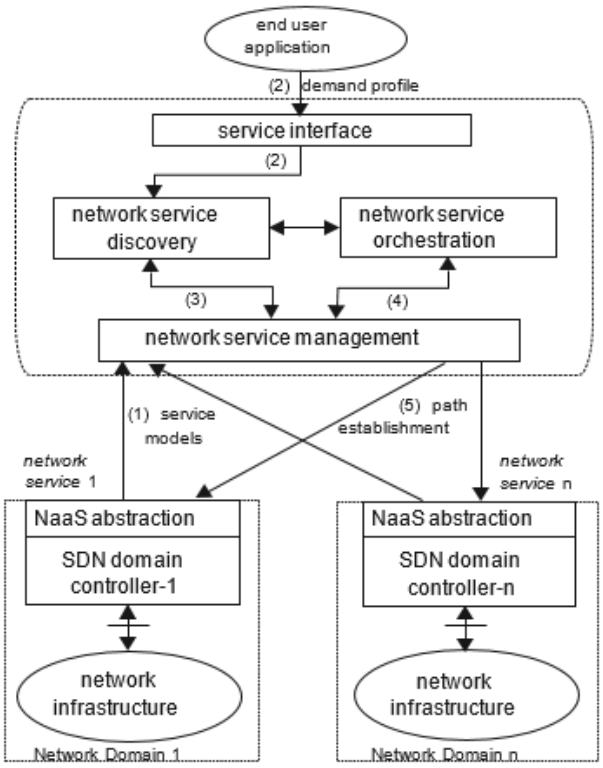


Figure 4. Network service selection and orchestration for end-to-end QoS provisioning

The SDN controller in each network domain is responsible for publishing the service capability model (the matrix ^{C}) of its domain at the SDP service management module. (step 1 in Figure 4). When an end user requests a network service from the SDP it submits a demand profile $D\{C, Q,$ L$\}$ through the service interface (step 2). The demand profile incudes a connectivity element C specifying the source and destination addresses (s,d), the Q element giving QoS requirements T_{req} and/or D_{req}, and a load descriptor L for the user's traffic flow. On receiving the request with the demand profile, the SDP service discovery module inquires the service management module for the capability models of all available network services. Connectivity and capacity information of single network services is first examined by the service discovery module to find a network service that can provide a virtual connection from s to d with sufficient capacity C_{min} for meeting QoS expectation (step 3). If a network service is found, the service management module will inform the SDN controller in the corresponding network domain for establishing a physical path and allocating bandwidth in network infrastructure (step 5). If no single network service can meet the requirements specified by the demand profile, the service orchestration module will search a chain of network services that can provide a virtual connection from s to d across multiple domains (step 4). The orchestration module determines the minimum capacity C_{min} required on the end-to-end virtual connection and only orchestrates network services with sufficient capacity for end-to-end service delivery. If such a service chain is found, the service management module will contact the SDN controller of each network domain involved in this service chain for establishing the physical path and allocating bandwidth in that domain (step 5).

The proposed SDP plays the role of a service broker for accepting end users' service requests, selecting appropriate network services for meeting users' requirements, and orchestrating multiple network services for inter-domain service delivery. Therefore, the SDP offers a platform that allows third party

providers to offer end-to-end network services by utilizing the services provided by infrastructure providers (who operate individual network domains). The SDP and domain controllers (representing infrastructure providers) establish a certain form of service contracts, which specify the service capabilities that the SDP expects from underlying network domains for supporting each virtual connection provided by the domains. The minimum service capacity that a domain must provide for a virtual connection, which can be determined by the technique developed in last subsection, is an important item included in the service contract for achieving QoS guarantee.

A centralized platform for service provisioning in a large scale networking environment may raise concerns about scalability issues. Interactions between the SDP and domain controllers cause overheads and delay that may limit scalability of the SDP for service management. NaaS-based network abstraction significantly simplifies information exchange between the SDN and domain controllers. The high-level abstraction model developed in this paper allows domain controllers to publish their service capability information with a relatively simple data structure; thus reducing overheads for network information exposure. Network service orchestration performed at the SDP searches available end-to-end paths based on a highly aggregated global virtual network topology; therefore does not need information dissemination among domain controllers as required by traditional distributed mechanisms for inter-domain routing. In order to establish paths for service delivery the SDP just needs to inform each domain controller about the required connectivity and capacity information. Such information can be described in a small set of parameters (the pair of source-destination border nodes for a virtual link and the minimum available bandwidth required on the virtual link); therefore limiting the control overheads between the SDP and domain controllers.

Please be noted that the proposed SDP framework is a logically centralized platform that may be realized by various implementations, which may have a distributed physical structure. The scalability issues associated with SDP-based service management shares a lot of similarity with the scalability of a centralized SDN controller controlling multiple switches in a large-scale network; therefore could be addressed by applying similar technical ideas. Although a thorough analysis on scalability of a NaaS-based SDP is an interesting and important problem, it is out of the scope of this paper and will be studied in our future work.

5.4 Bandwidth Utilization for End-to-End QoS Provisioning

The proposed SDP enables logically centralized service and resource management with a global network view for end-to-end QoS provisioning in a multi-domain SDN environment. Without such an SDP, the SDN controller in each domain provides a central control point but only within the scope of a single domain. No controller can obtain a purview of the entire path for service delivery across multiple domains; therefore, end-to-end QoS provisioning needs to be offered based on mutual collaboration between controllers in neighbor domains. With such a per-domain QoS mechanism, the end-to-end delay requirement is partitioned to a set of delay budgets, one for each domain involved in service delivery. Each domain has to determine and allocate sufficient amount of bandwidth in its own network infrastructure to guarantee its delay budget. With the proposed SDP, an end-to-end virtual path with QoS guarantee can be established through network service orchestration based on a global network view. The SDP determines the required service capacity on the path by viewing the entire path as if it belongs to one end-to-end virtual domain abstracted by a composite network service. The SDN controllers in all the

domains passed by the virtual path are required to allocate the same amount of effective bandwidth on the path, which is determined by the SDP. In this subsection, we study bandwidth utilization of the end-to-end QoS scheme enabled by the SDP and compare it with that of the per-domain QoS scheme without an SDP.

We consider a service delivery scenario in which a virtual path traverses n domains abstracted respectively by network services S_i, $i = 1,2,\cdots,n$, as shown in Figure 3. Denotes the virtual link provided by service S_i as l_i, and assume that the capacity profile of l_i is a LR profile $P_i = \beta(r_i,\vartheta_i)$, then the capacity profile of the end-to-end virtual path is $P_e = \beta(r_e,\vartheta_e)$, where $r_e = \min\{r1,\cdots,r_n\}$ and $\vartheta_e = \vartheta_1 + \vartheta_2 + \cdots + \vartheta_n$. Suppose the load descriptor for the traffic flow is $L(p,\rho,\sigma)$ and the expected end-to-end delay upper bound is D^e_{req}, then with the end-to-end QoS mechanism enabled by the SDP, the effective bandwidth that must be allocated to the virtual path for guaranteeing D^e_{req} can be determined by equation (12) as

$$R_e\left(D^e_{req}\right) = \frac{p\sigma}{\left(D^e_{req}-\theta\right)(p-\rho)+\sigma} \quad \left(D^e_{min} \le D^e_{req} \le D^e_{max}\right) \tag{13}$$

Where $D^e_{min} = \vartheta_e$ and $D^e_{max} = \vartheta_e + \sigma/\rho$. This is the amount of bandwidth that the SDP requests each domain controller to allocate for the virtual link provided by the domain.

Now we consider the case in which the per-domain QoS mechanism without a central SDP allocates effective bandwidth on a path passing the same set of n domains for meeting the same end-to-end delay requirement. Suppose the total delay requirement is partitioned to n delay budget D^i_{req}, $i = 1, 2, \ldots n$, one for each domain, then in the i-th domain the effective bandwidth that must be allocated on its virtual link l_i for meeting its delay budget will be

$$R_i\left(D^i_{req}\right) = \frac{p\sigma}{\left(D^i_{req}-\theta\right)(p-\rho)+\sigma} \quad \left(D^i_{min} \le D^i_{req} \le D^i_{max}\right) \tag{14}$$

where $D^i_{min} = \vartheta_i$ and $D^i_{max} = \vartheta_i + \sigma/\rho$.

Both R_e and R_i are the required amounts of bandwidth that need to be allocated in the domain S_i for meeting the same delay requirement, but the former is obtained by the SDP with a global network view while the latter is obtained by the local SDN controller in a single domain. To compare bandwidth utilization achieved by these two service management schemes, we defined U as the ratio between these two effective bandwidth values; that is,

$$U = \frac{R_i\left(D^i_{req}\right)}{R_e\left(D^e_{req}\right)} = \frac{\left(D^e_{req}-\theta_e\right))(p-\rho)+\sigma}{\left(D^i_{req}-\theta_i\right)(p-\rho)+\sigma}. \tag{15}$$

An observation we can have from (15) is that U = 1 when $p = \rho$. This implies that for a constant rate traffic flow ($p = \rho$) the end-to-end allocation scheme enabled by the SDP has the same level of bandwidth utilization as per-domain allocation does. This is because the effective bandwidth for a constant rate flow just needs to be the peak/sustained rate of the flow, and extra bandwidth allocation does not help improving delay performance; that is $R_e(D^e_{req}) = R_i(D^i_{req}) = p = \rho$.

We focus our analysis on the case of variable rate traffic flows; i.e. $p > \rho$. We define $\Delta d_e = D^e_{req} - \vartheta_e$ as an indicator to reflect how tight the end-to-end delay expectation is compared to the latency parameter of the service delivery path, which is the minimum delay that can be achieved on the path. A larger Δd_e value implies a relatively looser delay expectation. Similarly $\Delta d_i = D^i_{req} - \vartheta_i$ reflects the relative tightness of the delay budget for a single network domain S_i. Equation (15) shows that if $\Delta d_i \ge \Delta d_e$ then $U \le 1$; otherwise $U > 1$. This implies that if the delay budget for a single network domain, compared to the latency of the

virtual link in this domain, is relatively looser than the delay expectation for end-to-end service delivery, then the per-domain allocation scheme may actually require less amount of bandwidth than what is required by the SDP. However, given an end-to-end delay bound requirement, a loose delay budget for one network domain means tighter delay budgets thus more bandwidth consumption in other domains. Autonomous domains in the Internet are unlikely to sacrifice their own bandwidth resources for other domains' benefits.

Therefore we analyze a case that the end-to-end delay requirement is equally partitioned among all domains and assume the virtual links in all domains have an identical latency property; that is, $D^i_{req} = d$ and $\vartheta_i = \vartheta$ for $i = 1,2,\cdots,n$. Then $D^e_{req} = nd$ and from (5) we have $\vartheta_e = n\vartheta$.

Therefore,

$$R_e\left(D^e_{req}\right) = \frac{p\sigma}{n(d-\theta)(p-\rho)+\sigma}, \quad R_i\left(D^i_{req}\right) = \frac{p\sigma}{(d-\theta)(p-\rho)+\sigma}. \tag{16}$$

Then bandwidth ratio is

$$U = \frac{R_i(D^i_{req})}{R_e\left(D^e_{req}\right)} = \frac{n(d-\theta))(p-\rho)+\sigma}{(d-\theta)(p-\rho)+\sigma} = 1 + \frac{(n-1)(d-\theta)(p-\rho)}{(d-\theta)(p-\rho)+\sigma}. \tag{17}$$

Since we are considering variable rate flows ($p > \rho$) and the delay budget assigned to a network domain must be larger than the latency property of its network infrastructure ($d > \vartheta$), (17) shows that the bandwidth ratio U > 1 for end-to-end service delivery across domains ($n \geq 2$).

Equation (12) also gives a special case for loose delay expectation; that is, $R_e(D^e_{req}) = \rho$ when $D^e_{req} > D^e_{max} = \vartheta_e + \sigma/\rho$. Similarly, for per-domain allocation $R_i(D^i_{req}) = \rho$ when $D^i_{req} > D^i_{max} = \vartheta_i + \sigma/\rho$. Considering the above case in which $D^i_{req} = d = D^e_{req}/n$ and $\vartheta_i = \vartheta$ for $i = 1,2,\cdots,n$, then

$$\frac{D^e_{max}}{n} = \theta + \frac{\sigma}{n\rho} < D^i_{max} = \theta + \frac{\sigma}{\rho} \quad (n \geq 2) \tag{18}$$

Inequality (18) implies that for an end-to-end delay expectation that is looser than the maximum threshold; i.e., $D^e_{req} > D^e_{max}$, the effective bandwidth determined by the SDP will be $R_e(D^e_{req}) = \rho$. However, when dividing this delay expectation equally to obtain n delay budgets, one for each single domain, the obtained D^i_{req} might be tighter than the maximum delay threshold of its domain D^i_{max}; therefore the local controller may allocate more bandwidth; i.e. $R_i(D^i_{req}) > \rho$ in each domain.

The above analysis shows that in order to achieve the same level of delay performance guarantee in the considered scenarios, the per-domain QoS mechanism consumes more bandwidth in each individual network domain than the effective bandwidth determined by the SDP. This result indicates that the proposed SDP may not only simplify service management but also enhance bandwidth utilization for end-to-end QoS provisioning in a multi-domain SDN environment.

6 Numerical Results

Numerical results are given in this section to illustrate application of the developed techniques and obtained insights. We considered a networking scenario in which the SDP orchestrates the network services of three domains to provide an end-to-end virtual path for QoS provisioning. The path has been used to transport data for two applications, whose traffic flows, denoted as f_1 and f_2 respectively, are characterized by the following parameters: peak rate $p1$ = 60 Mbps, sustained rate $\rho 1$ = 1.5 Mbps, and

burst size $\sigma 1$ = 1.04 Mbits for f_1; and peak rate $\rho 2$ = 15 Mbps, sustained rate $\rho 2$ = 1.5 Mbps, and burst size $\sigma 2$ = 9.54 Mbits for f_2. These parameters are derived from traffic analysis reported in [33] and [34]. We assume that virtual links provided by all domains for the end-to-end path have a LR capacity profile.

Bandwidth allocation for achieving end-to-end delay performance guarantee is first analyzed. The amounts of effective bandwidth that the SDP must request from each domain to guarantee a set of delay requirements are determined and plotted in Figure 5, where effective bandwidth for f_1 and f_2 are denoted as R_e^1 and R_e^2 respectively. From this figure, we can see that the required amounts of bandwidth for both flows increase when the delay requirement value decreases. This means that more service capacity must be acquired by the SDP from the underlying network domains in order to provide a tighter end-to-end service delay guarantee.

Comparing the bandwidth curves in Figure 5 shows that R^2_e is greater than R^1_e for all delay requirements; that is, different amounts of bandwidth are required by these two flows for achieving the same delay performance on the same virtual path. This means that effective bandwidth is impacted by traffic load parameters as well as the delay requirement; thus verifying the necessity of including a load descriptor in a service demand profile in order for the SDP to achieve QoS guarantee. From Figure 5 we can also see that the R^2_e curve drops with increasing delay requirement value faster than the R^1_e curve does. This implies that for flows with different traffic load parameters, the same extent of improvement in delay performance requires different amounts of increment in effective bandwidth. Both the flows examined in our experiments have the same sustained rate ($\rho 1 = \rho 2$) but flow f_2 has greater burst size ($\sigma 2 > \sigma 1$). Such an observation we obtained from Figure 5 indicates that the parameter σ, which gives the maximum amount of traffic that an application can load on a virtual path continuously with its peak rate, has a strong impact on the required amount of bandwidth for achieving delay guarantee.

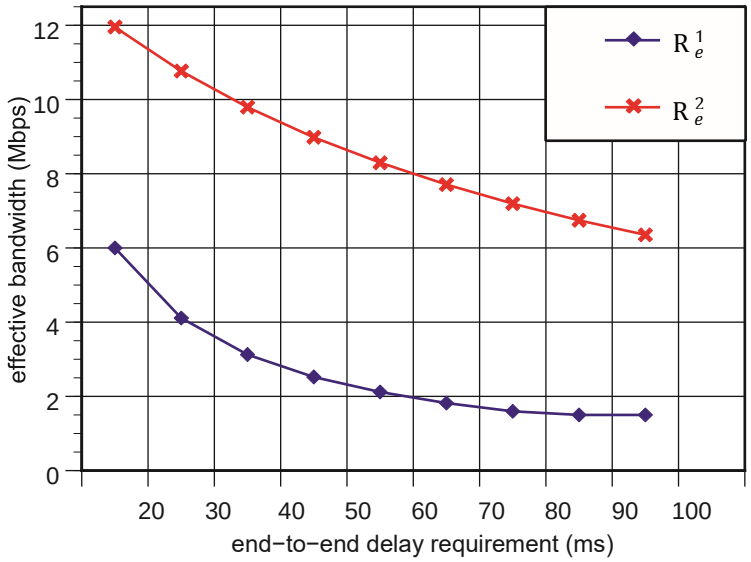


Figure 5. Effective bandwidth for end-to-end delay guarantee for flows f_1 and f_2

One can also notice from Figure 5 that the R^1_e curve becomes flat when the required delay bound is greater than a threshold (90 ms in this particular example) while the R^2_e curve keeps dropping with increasing delay bound value. This is because effective bandwidth is equal to the sustained rate of a flow when the required delay bound is looser than a threshold, as shown in equation (12) $R_e(D_{req}) = \rho$ when $D_{req} \geq D_{max}$. Also, the D_{max} threshold for a flow varies with the load parameters of the flow. In our experiment flow f_1

reaches such a threshold at around 90 ms where the R^1_e curve becomes flat; while f_2 does not reaches its threshold for all the delay bound values tested in the experiment; therefore the R^2_e curve keeps dropping with increasing delay requirement.

In order to evaluate bandwidth utilization achieved by the SDP with a global network view for QoS provisioning in a multi-domain SDN environment, we compare the end-to-end bandwidth allocation scheme performed by the SDP against the per-domain-based bandwidth allocation scheme discussed in Subsection V-D. We assume that the virtual links in the three network domains have identical LR capacity profiles and the end-to-end delay requirement is divided equally as the delay budgets for the three domains. We analyzed the amounts of effective bandwidth that will be determined by each individual SDN controller for meeting the delay budget in its domain. The obtained data for flows f_1 and f_2 are plotted in Figures 6 and 7, in which the per-domain allocation results for f_1 and f_2 are respectively denoted as R^1_d and R^2_d.

Figures 6 and 7 show that for a given flow, the amounts of effective bandwidth determined by the SDP with an end-to-end allocation scheme and by individual SDN controllers with per-domain allocation are both decreasing functions of the required delay bound. That is, more bandwidth is required by both schemes to achieve a tighter delay guarantee. Another important observation one can obtain from Figures 6 and 7 is that for both flows the SDP end-to-end allocation scheme always requires less amount of bandwidth than what per-domain allocation does in order to provide the same level of delay performance guarantee. The data shown in Figures 6 and 7 verify that the end-to-end bandwidth allocation enabled by the SDP with a global network view can achieve higher bandwidth utilization compared to the per-domain bandwidth allocation scheme of the conventional inter-domain QoS mechanism. This indicates that the SDP may realize the potential advantage of SDN logically centralized control vision to improve resource utilization for QoS provisioning in a multi-domain networking environment.

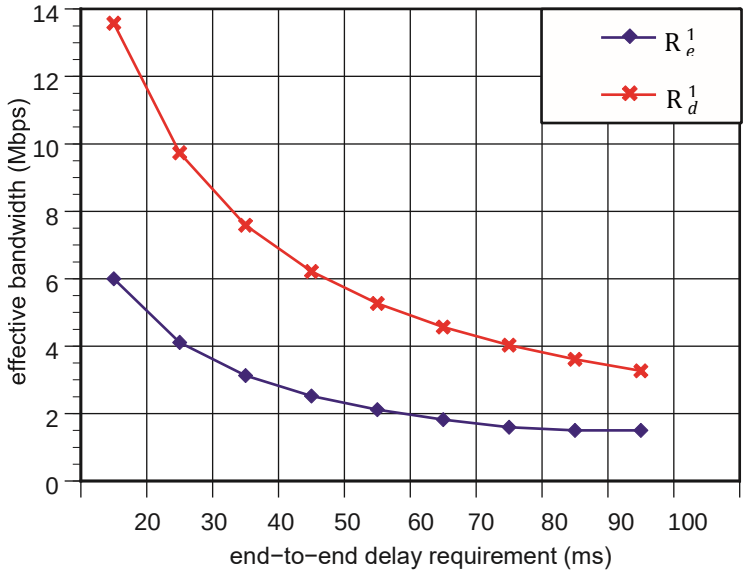


Figure 6. Comparison between effective bandwidth obtained by end-to-end and per-domain allocation schemes for flow f_1.

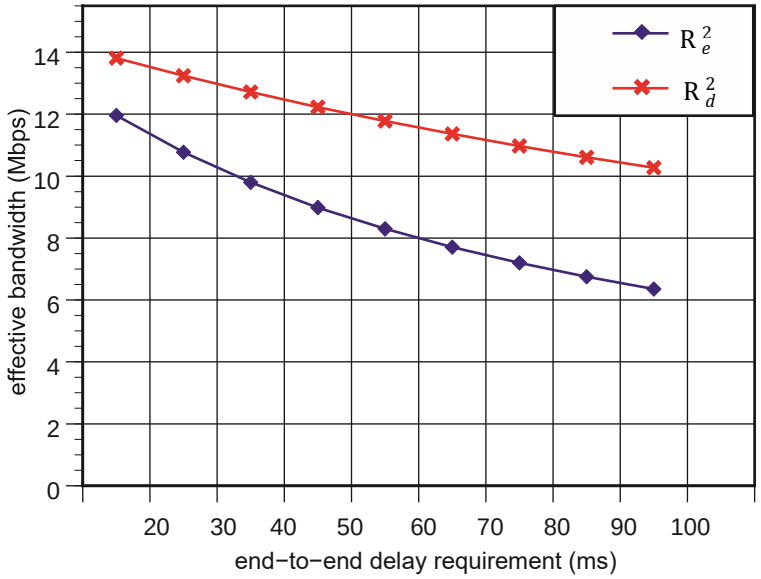


Figure 7. Comparison between effective bandwidth obtained by end-to-end and per-domain allocation schemes for flow f_2.

In order to examine the extent of improvement in bandwidth utilization introduced by the SDP, we also analyzed bandwidth ratios for flows f_1 and f_2, which are defined as $U_1 = R^1_d/R^1_e$ and $U_2 = R^2_d/R^2_e$ respectively. The obtained data are plotted in Figure 8. This figure shows that the bandwidth ratios of both flows are greater than 1; that is, end-to-end bandwidth allocation enabled by SDP achieves higher bandwidth utilization for providing delay performance guarantee. Comparison between the curves of U_1 and U_2 in Figure 8 shows that the two flows have different bandwidth ratio values for achieving the same delay requirement and $U_1 > U_2$ for all the delay bounds tested in our experiments. This implies that load parameters of a traffic flow also have an impact on the extent of improvement in bandwidth utilization introduced by the SDP to the flow.

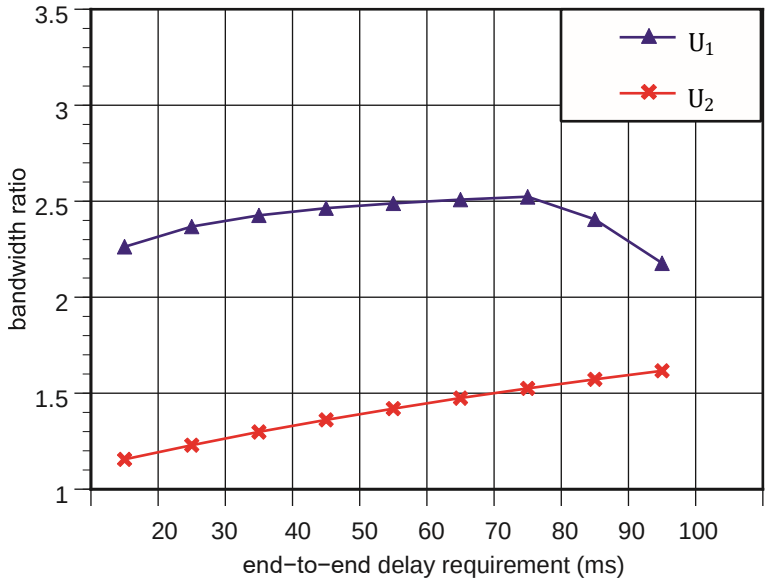


Figure 8. Bandwidth ratios for flows f_1 and f_2 for achieving different delay objectives.

We also noticed from Figure 8 that both U_1 and U_2 increase with the required delay bound in most cases except that U_1 drops when the delay requirement is greater than 80 ms. This implies that end-to-end bandwidth allocation enabled by the SDP typically achieves more improvement in bandwidth utilization for looser delay bounds than for tighter delay bounds. The exception happens flow f_1 when the delay bound is greater than the threshold (80 ms for in this experiment) beyond which end-to-end effective bandwidth is equal to the sustained rate of the flow; that is, the D_{max} beyond which $R_e(D_{req}) = \rho$ as shown in (12). Since the sustained rate is the minimum effective bandwidth that the SDP must request in each domain for achieving any delay bound guarantee, R^1_e stops decreasing for any looser delay bound requirement (as shown by the R^1_e curve in Figure 5); therefore will not further enhance bandwidth utilization. We noticed that even in this case U_1 is still well above 1; that is, end-to-end allocation saves bandwidth than per-domain-based allocation.

In order to evaluate the influence of the number of passed domains on improvement in bandwidth utilization, the bandwidth ratios of the two flows for guaranteeing a delay bound of 60 ms are tested with different numbers of domains passed by the end-to-end virtual path. The obtained results are plotted in Figure 9. This figure shows that both ratios increase with the number of domains, which implies that the more domains the virtual path traverses, the bigger is the difference between the amounts of effective bandwidth determined by the SDP and by individual domain controllers. We can also see from this figure that flows f_1 and f_2 have different bandwidth ratio values with the same number of domains, which reflects the influence of traffic load parameters on the bandwidth utilization improvement that can be achieved by the SDP. Comparing the two ratio curves in this figure shows that their increasing speeds with number of domains are quite different and U_1 increases much larger than U_2. This implies that the number of domains involved in service delivery has a stronger impact on bandwidth utilization to flow f_1 then to flow f_2, which again mainly due to the difference in the traffic load profiles of these two flows.

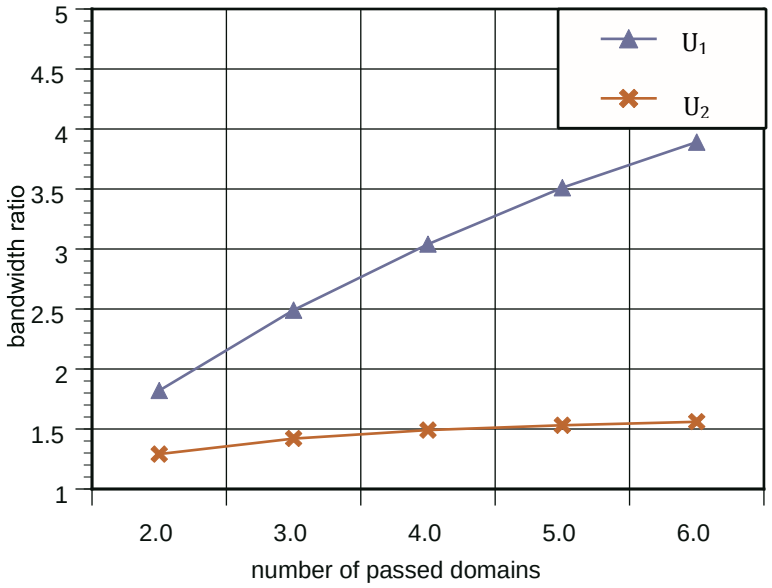


Figure 9. Bandwidth ratios for flows f_1 and f_2 passing different number of network domains.

7 Conclusions

In this paper, we studied the problem of end-to-end service delivery with QoS guarantee in the SDN-based future Internet. The autonomous network domains coexisting in the Internet and the diverse user applications deployed upon the Internet calls for a uniform Service Delivery Platform (SDP) that offers a high-level network abstraction and enables inter-domain collaboration for end-to-end service provisioning. However, the currently available SDN technologies lack effective mechanisms for supporting such a platform. In order to address this important and challenging issue, in this paper we first presented an SDP framework that employs the Network-as-a-Service (NaaS) principle to provide a high-level network abstraction and enables inter-domain collaboration through service orchestration for end-to-end service delivery. Then we focused our study on two key technologies, a network abstraction model and an end-to-end resource allocation scheme, for the SDP to achieve QoS guarantee. We proposed a general abstract model for characterizing the service capabilities offered by heterogeneous network domains and apply the model for abstracting end-to-end inter-domain network services. Then we developed a technique that can be used by the SDP to determine the minimum amount of bandwidth that must be allocated in each network domain involved in service delivery for achieving end-to-end QoS guarantee. We also examined bandwidth utilization of the SDP-based end-to-end resource allocation and compared it with per-domain based resource allocation. Both the analytical and numerical results obtained in this paper indicate that an SDP with the proposed network abstraction and resource allocation technologies not only simplifies service and resource management in SDN but also enhances bandwidth utilization for end-to-end QoS provisioning. Therefore, such an SDP framework offers a promising approach to fully realizing a key benefit promised by the SDN paradigm – logically centralized control for service provisioning to support diverse user applications – in a large-scale multi-domain networking environment.

REFERENCES

[1] ONF,“Open Networking Foundation Software-Defined Networking (SDN) Definition,” *https://www.opennetworking.org/sdn-resources/sdn-definition*, 2013.

[2] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, “A survey on Software-Defined Networking,” *IEEE Communications Surveys and Tutorials*, 2014.

[3] Y. Jarraya, T. Madi, and M. Debbabi, “A survey and a layered taxonomy of Software-Defined Networking,” *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, 2014.

[4] D. Kreutz, F. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[5] T. Erl, *Service-Oriented Architecture – Concepts, Technology, and Design*. Prentice Hall, 2005.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[7] A. Doria, J. H. Salim, R. Hass, H. Khosravi, W. Wang, L. D. and R. Gopal, and J. Halpern, “Forwarding and control element separation (ForCSE) protocol,” *Internet Engineering Task Force Sepcification*, Mar 2010.

[8] J. Vasseur and J. L. Roux, "IETF RFC5440: Path Computation Element Communication Protocol (PCEP)," Mar. 2009.

[9] H. Song, "Protocol-Oblivious Forwarding: Unleash the power of SDN through a futur-proof forwarding plane," in *Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13)*, pp. 127–132, Jan. 2013.

[10] M. Smith, M. Dvorkin, Y. Laribi, V. Pandey, P. Garg, and N. Weidenbacher, "OpFlex control protocol," *Internet Research Task Force Internet-Draft*, April 2014.

[11] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. Mckeown, and S. Shenker, "NOX: Toward an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[12] D. Erickson, "The Beacon OpenFlow controller," in *Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13)*, Jan. 2013.

[13] "Floodlight OpenFlow Controller." http://www.projectfloodlight.org/floodlight/.

[14] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "ONIX: a distirbuted cotnrol platform for large-scale production networks," in *Proc. of the 9th USENIX Conference on Operating Systems Design and Implementation*, Oct. 2010.

[15] U. Krishnaswamy, P. Berde, J. Hart, M. Kobayashi, P. Radoslavov, T. Lindberg, R. Sverdlov, and S. Zhang, "ONOS: An open source distributed SDN OS." available online: http://www.slideshare.net/ON-LAB/onos-open-network-operating-system-an-opensource-distributed-sdn-os.

[16] A. Tootoonchian and Y. Ganjali, "HyperFlow: a distributed control plane for OpenFlow," in *Proc. of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, Apr. 2010.

[17] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, and R. Sidi, "SDNi: A message exchange protocol for Software Defined Networks (SDNS) across multiple domains," *Internet Research Task Force Internet-Draft*, Jun 2012.

[18] R. Bennesby, P. Fonseca, E. Mota, and A. Passito, "An Inter-AS routing component for Software-Defined Networks," in *Proc. of the 2012 IEEE/IFIP Network Operations and Management Symposium (NOMS12)*, Aug. 2012.

[19] V. Kotronis, X. Dimitropoulos, and B. Ager, "Outsourcing the routing control logic: Better internet routing based on SDN principle," in *Proc. of the 11th ACM Workshop on Hot Topics in Networks (Hotnets'12)*, Oct. 2012.

[20] P. Thai and J. C. de Oliveira, "Decoupling policy from routing with software defined interdomain management: Interdomain routing for SDN-based networks," in *Proc. of the 2012 IEEE International Conference on Computer Communications and Networks (ICCCN'12)*, July 2012.

[21] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controller," in *arXiv preprint arXiv:1308.6138*, Aug. 2013.

[22] P. Costa, M. Migliavacca, P. Pietzuch, and A. L. Wolf, "NaaS: Network-as-a-Service in the Cloud," in *Proc. of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Apr. 2012.

[23] T. Feng, J. Bi, H. Hu, and H. Cao, "Networking-as-a-Service: a Cloud-based network architecture," *Journal of Networks*, vol. 6, pp. 1084–1090, July 2011.

[24] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, and G. Wang, "Meridian: An SDN platform for Cloud network services,"*IEEE Communications Magazine*, vol. 51, pp. 120–127, Feb. 2013.

[25] I. Bueno, J. Aznar, E. E. J. Ferrer, and J. A. Garcia-Espin, "An OpenNaaS based SDN framework for dynamic QoS control," in *Proc. of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov. 2013.

[26] Q. Duan, "Network-as-a-Service in Software-Defined networks for end-to-end QoS provisioning," in *Proc. of the 2014 IEEE Wireless and Optical Communications Conference*, May 2014.

[27] J. Zhu, W. Xie, L. Li, M. Luo, and W. Chou, "Software service defined network: Centralized network information service,"in *Proc. of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov. 2013.

[28] H. E. Egilmez and a. M. Tekalp, "Distributed QoS architectures for multimedia streaming over software defined networks,"*IEEE Transactions on Multimedia*, vol. 5, no. 16, 2014.

[29] Q. Duan, Y. Yan, and A. V. Valisakos, "A survey on service-oriented network virtualization toward convergence of networking and Cloud computing," *IEEE Transactions on Network and Service Management*, vol. 9, pp. 373–392, Dec.2012.

[30] R. Alimi, R. Penno, and Y. Yang, "Internet-Draft: Application Layer Traffic Optimiation (ALTO) protocol," Mar. 2014. [31] A. Atlas, J. Halpern, S. Hares, and D. Ward, "Internet-Draft: An Archiecture of Interface to the Routing System," June 2013.

[32] J. L. Boudec and P. Thiran, *Network calculus: a theory of deterministic queueing systems for the Internet*. Springer Verlag, June 2001.

[33] J. W. Roberts, "Internet traffic, QoS, and Pricing," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1389–1399, 2004.

[34] R. R. Boorstyn, A. Burchard, J. Liebeherr, and C. Oottamakorn, "Statistical service assurances for traffic scheduling algorithms," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 2651–2664, Dec. 2000.

SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 6, NO. 2
ISSN: 2054 -7420

Applying Big Data, Machine Learning, and SDN/NFV for 5G Early-Stage Traffic Classification and Network QoS Control

Luong-Vy Le[1], Bao-Shuh Paul Lin[2,3], Do Sinh[2]
[1]*College of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan*
[2]*Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan*
[3]*Microelectronics & Information Research Center, National Chiao Tung University, Hsinchu, Taiwan*
leluongvy.eed03g@nctu.edu.tw, bplin@mail.nctu.edu.tw, dosinhuda.cs04g@nctu.edu.tw

ABSTRACT

Due to the rapid growth of mobile broadband and IoT applications, the early-stage mobile traffic classification becomes more important for traffic engineering to guarantee Quality of Service (QoS), implement resource management, and network security. Therefore, identifying traffic flows based on a few packets during the early state has attracted attention in both academic and industrial fields. However, a powerful and flexible platform to handle millions of traffic flows is still challenging. This study aims to demonstrate how to integrate various state-of-the-art machine learning (ML) algorithms, big data analytics platforms, software-defined networking (SDN), and network functions virtualization (NFV) to build a comprehensive framework for developing future 5G SON applications. This platform successfully collected, stored, analyzed, and identified a huge number of real-time traffic flows at broadband Mobile Lab (BML), National Chiao Tung University (NCTU). Moreover, we also implemented network QoS control to configure priorities per-flow traffic to enable bandwidth guarantees for each application by using SDN. Finally, the performance of the proposed models was evaluated by applying them to a real testbed environment. The powerful computing capacity of the platform was also analyzed.

Keywords: Traffic classification; Machine Learning; Big Data; SON; 5G; InfoSphere; Streaming.

1 Introduction

In 5G networks and IoT contexts, small cells, heterogeneous networks (HetNets), wireless sensor networks (WSNs) are deployed everywhere to bring enhanced mobile broadband services to users, such as Internet of Everything (IoE) paradigms, cloud services, and real-time video streaming services. However, the diversity of broadband services carries many challenges for traffic engineering to provide a technical solution to improve the network QoS and QoE (quality of experience). For example, video streaming is a time-sensitive service, so any unexpected delay may result in bad QoE. Moreover, understanding traffic flow behaviors of running applications in the network play a critical role for network operators to implement QoS and QoE policies, network efficiency, resource management, load balancing, energy saving [1].

Recently, the early-state traffic classification has become an important topic in communication, and it was explored in many studies [2][3][4][5][6][7][8]. Generally, a traffic classification model can be divided into

DOI: 10.14738/tnc.62.4446
Publication Date: 1st May 2018
URL: http://dx.doi.org/10.14738/tnc.62.4446

two steps: Firstly, traffic is divided into flows, and their headers are extracted to define useful features for the classification model such as packet number, packet size. Secondly, the classification model trains and classifies traffic flows into different types of applications. Studies [7][8] introduced several popular approaches for the early-stage traffic classification: Port-based classification, payload-based classification, protocol behavior or heuristics based classification, statistical analysis based classification, deep packet inspection (DPI) approaches, and packet size approaches. The port-based approach is simple and easy to implement, however, because nowadays the number of applications using dynamic port is increasing, this model becomes inaccurate. On the other hand, the payload-based approach investigates packet payload to determine the signatures of known applications; therefore, it can only classify traffic flows for which signatures are available, and it usually requires substantial computing power and storage capacity, besides, this method also violates the privacy laws. Another example, the DPI approach, which aims at identifying traffic protocol patterns in the packets of different applications, is too complicated with high computational overheads. Fortunately, the most current studies, such as [6][5][9][10], concluded that traffic classification models based on the packet sizes of some first packets were powerful enough for achieving high classification performance. Furthermore, research [6] found that the most efficient number of packets used for traffic flows identification is from 5 to 7 packets. That means too many packets and too few packets may reduce the model efficiency. In addition, ML algorithms play a significant role in deciding the success of the model. Those studies focused on applying powerful and well-known ML algorithms as classifiers, such as Naïve Bayes, support vector machine (SVM), Random Forest, logistic, Hidden Markov Model (HMM) etc. For example, research [6] investigated the performance of 11 well-known ML models. Especially, In the study [2], we proposed a traffic classification model using HMM to classify the internet traffic of mobile broadband applications based on the packet sizes and packet transmission directions. As a result, it achieved 99.17% accuracy for 6 types of mobile applications.

One of the most essential application of the early-state traffic flow classification is network QoS control. In research [1], the authors investigated and proposed a systematic design approach to support QoS-guaranteed chaining services with considering the effects of both data plane and control plan messages. The delay of the services and SDN were evaluated

This study focuses on enhancing the current architecture to propose a comprehensive framework of integrating big data, ML SDN/NFV, and cloud for collecting, transforming, extracting, and analyzing mobile traffic flows and then building powerful mobile traffic classification at the early stage. Furthermore, a new and effective network QoS control based on Open Flow Switch is presented. The experiments are deployed on the experimental 4G/LTE & beyond 4G network testbed, located at MIRC/BML (Microelectronics and Information Research Center/Broadband Mobile Lab) in the campus of National Chiao Tung University).

The remainder of the paper is organized as follows: Section 2 introduces the experimental architecture based on SDN/NFV, big data, and ML; Section 3 demonstrates Open-SON platform based on big data, ML, and SDN/NFV; Section 4 implements, evaluates, and analyzes the early-state traffic flow classification application; Section 5 proposes and implement per-flow traffic QoS control based on SDN, Section 6 concludes the present study.

2 Experimental Architecture Based on Big Data, ML and SDN/NFV at BML

Recently, ML, big data, cloud, and SDN/NFV have been applying in developing 5G networks to support computing, communication, programming abstractions, data analysis, and management services. Those technologies have been applied to the experimental 4G/LTE&5G network testbed, located at MIRC/BML. This platform integrates several big data and ML environments (e.g. Apache Spark, IBM InfoSphere) that works in collaboration with the advanced SDN/NFV technologies (e.g. ONOS, OpenDaylight, P4, Docker) to design various 5G applications. Based on the platform, many applications were introduced. For example, research [11] investigated the roles of 5G in the development of cloud, big data, SDN, and IoT, and then it proposed an integrated architecture of these technologies for 5G; research [12] addressed the technology outlook of computing power and system characteristics of 5G Mobile broadband system; research [13] described and implemented cloud computing for various Mobile Augmented Reality (MAR) applications with smart mobile devices; research [14] showed the challenges of applying SDN/NFV to 5G and IoT. [15][16] investigated various ML algorithms to cluster, forecast, and manage handover behaviors and traffic behaviors of a huge number of cells based on analyzing a huge amount of collected data. This study proposes a comprehensive architecture for traffic flow classification, the architecture and real devices is described as Fig.1.

2.1 Physical Devices

- ✓ UEs: some modern UEs such as iPhone 5s, iPhone 6s, Zenphone 3 are used to open broadband services
- ✓ eNodeB: both indoor and outdoor RRUs are used to connect with Nokia BBUs located inside BML lab. Furthermore, we can implement experiment with multi-vendors (NSN (Nokia), Huawei) and multi-modes (TDD and FDD).
- ✓ EPC: (Evolve packet core): The BBUs are connected to 2 cores through a switch, the main core is located at ITRI and the open core (open5G core) is located in BML for developing 5G applications.
- ✓ SDN/NFV environment: SDN and NFV are recognized as key technologies studied for enhancing mobile network applications by providing the capacity of programmability for control and data plane of both RAN and EPC's elements.

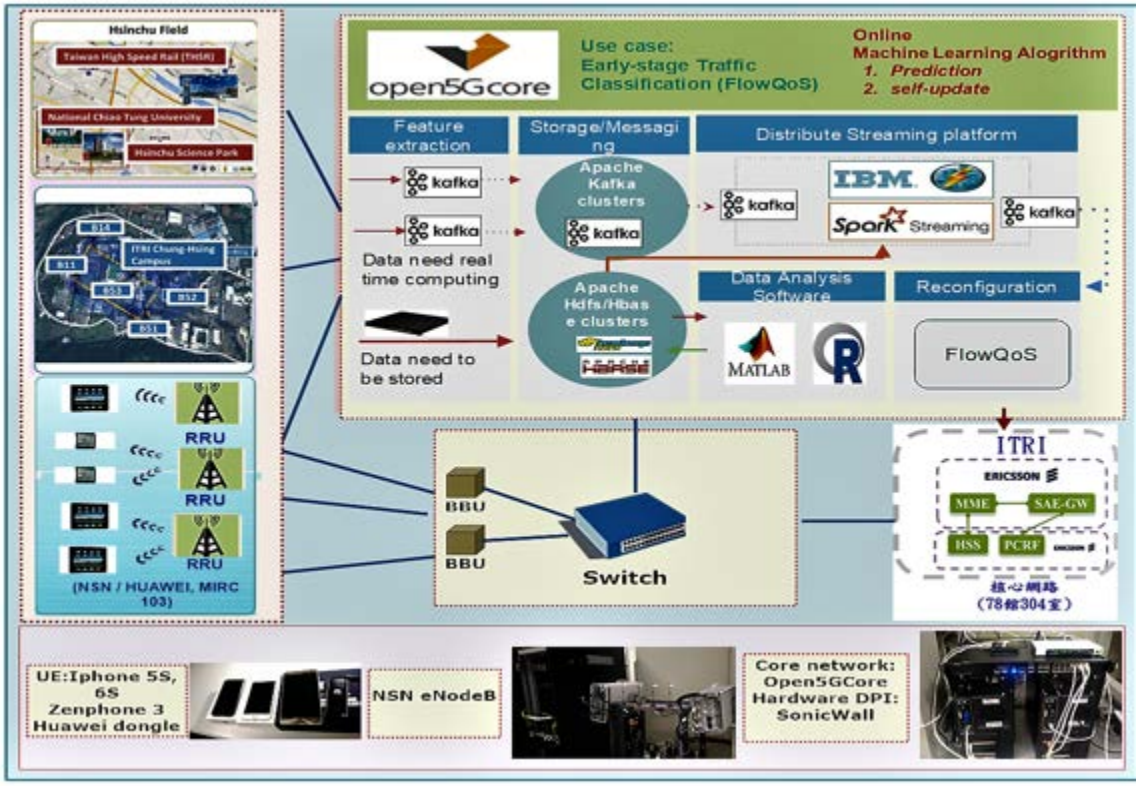


Figure 1. Experimental architecture and devices of BML

2.2 Experimental Process

Figure 1. demonstrates the process of data collection, data transformation, data storage, data analysis, visualization, and ML applications for both statistical models and online/streaming models in this study. Generally, the common process for applying ML and data mining to mobile networks usually involves 4 steps:

Step 1: Collect and storage mobile traffic and other data that necessary for the application from all sources of the network.

Step 2: Using data mining and machine learning to develop optimization models by extracting and analyzing the collected data at the open5Gcore

Step 3: Apply the models and optimization parameters to network components such as the SON in the main EPC, SDN controller, and eNodeB

Step 4: Analyze and evaluate the network performance through network KPIs (Key performance indicators) to determine whether the optimization model meets the expected results. If the network behavior achieves the expected performance, the new network parameters (NPs) will be applied. Otherwise, we need to identify problems such as change machine learning models or learning parameters.

3 Big Data and Machine Learning Platform for Empowering 5G SON

The SON of 5G consists of three main functions: self-configuration, self-optimization, and self-healing. Self-configuration provides plug-and-play functionality for both eNodeBs and EPC, for instance, when a new network element is added to the RAN such as a BBU or RRU, it will automatically download the necessary software as well as configure basic network parameters, such as the neighbor list, the radio

parameters, etc. Self-Optimization continuously optimizes and controls the network parameters to respond the real-time network states to ensure that the network is working efficiently at the peak performance. For example, load-balancing algorithms are used to optimize RAN traffic, energy-saving algorithms turn RAN elements on or off due to traffic load [15]. The self-healing responds to a failure or a malfunction in the network with two steps: The cell outage detection and the cell outage compensation. It involves remote diagnosis, abnormal detection, failure prediction, and compensation to keep the network operating smoothly by detecting problems, and then automatically changes the network parameters of the active elements. Therefore, it requires a powerful, cost-effective, and autonomous SON in 5G with full intelligence to meet the requirement of both users and operators.

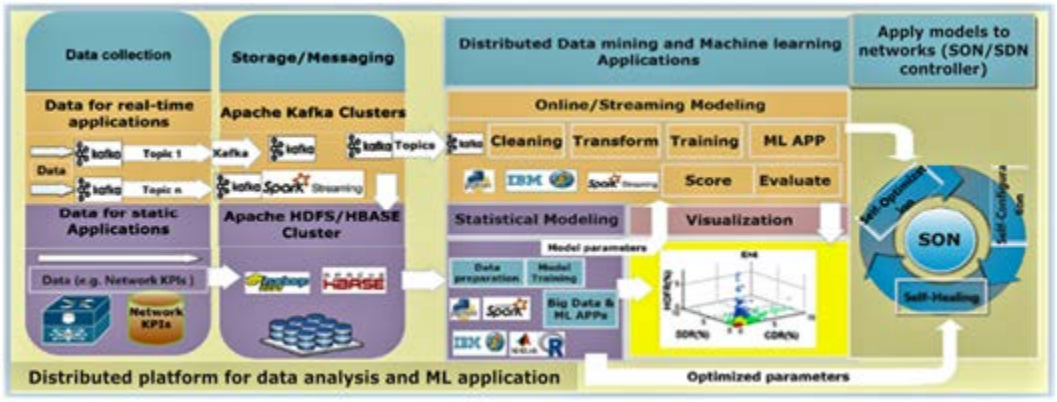


Figure 2 Open-SON for 5G platform

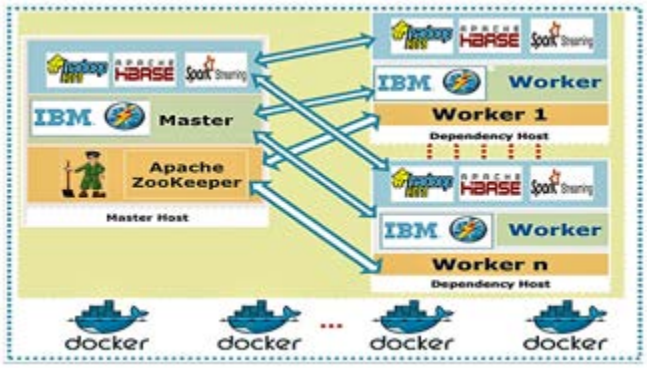


Figure 3. Distributed computing platform

3.1 Open Platform for 5G SON

Virtual SON (VSON), which integrates lately developed technologies such as software-defined wireless networking (SDWN), NFV, big data, ML, has been recently introduced as a prime proposal for future SON [17]. In this subsection, we propose and analyze an Open-SON for 5G in which we can apply data analytics and ML algorithms to develop variety SON applications. The big data and ML framework for the Open-SON shown in Fig. 2 composes four components: data collection; data storage; data analytics and ML applications; network configuration and optimization.

The platform is open to different types of technologies that can be easily integrated and deployed to build both online and offline applications. For example, Kafka, Flute, and Python are used to collect data from

different data resources; programming languages such as R, Matlab, Spark, and InfoSphere can be used for analyzing data and building various ML algorithms. Especially, these software platforms are compatible with one another, for example, Spark can support various languages: R, Python, Java, Scala. Moreover, to satisfy the flexibility and scalability requirements for 5G applications, the Open-SON must work in distributed computing system in which a host works as the master and multiple hosts work as workers or executors. Fig 3 shows that all components in the master and workers are deployed in Docker containers, this helps the deployment of computing application becomes easier, quicker, and more efficient. In this framework, ZooKeeper manages and controls all workers so that a distributed application can run on multiple executors in a cluster simultaneously. The executors coordinate among themselves to handle a specific task in a fast and efficient way. In other words, the software components such as Kafka, HBase, Spark, InfoSphere run concurrently and independently on multiple physical machines. This makes the system becomes more powerful fully with intelligence and automation. Finally, the distributed computing system are deployed and controlled based on SDN/NFV environments.

3.2 Machine Learning Algorithms for Empowering SON

Fig. 4 summarizes ML algorithms that are used to reinforce the Open-SON talent in building various applications such as clustering, classification, prediction, and forecasting.

Clustering applications usually utilize unsupervised algorithms such as K-means and Mixtures of Gaussians to group and identify a set of similar network parameters together, such as clustering network parameters (NPs), coverage area of a cell, traffic density, data compression, and abnormal detections. Especially, in research [15], we used K-means to cluster the handover behaviors of 2000 cells and then extracted the handover characteristics of each cluster and all cells in a cluster.

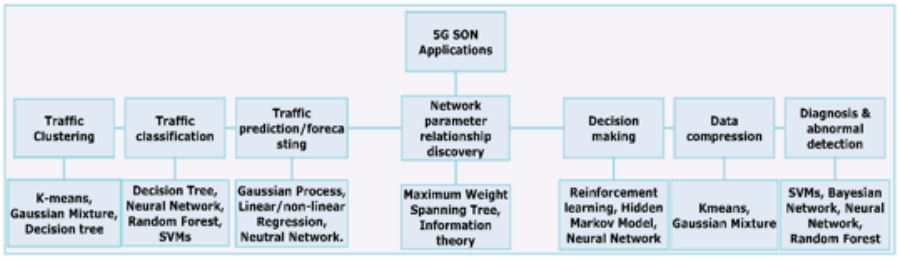


Figure 4. SON applications and ML algorithms

Forecasting and prediction applications aim to identify and predict precise trends of the network parameters, network events while operating. This helps the SON keeps track of network parameters and deploys further optimization applications. For example, in studies [15] and [16], we proposed several models to accurately and efficiently forecast future handover numbers and the traffic of a huge number of cells by using several ML algorithms: Neural network (NN), Gaussian process (GP), and linear regression. Other examples of typical prediction and forecasting applications that can be applied to 5G networks are subscriber tracking, HO trend prediction, power control, antenna adjustment (e.g., tilt, azimuth, transmitted power), and load balancing. Typical dynamic ML algorithms can be used for those applications are Kalman Filter, Random Forest, HMM, NN, Linear Dynamical Systems, and GP.

Classification applications can use both unsupervised and supervised algorithms to classify network parameters into the relevant groups based on some significant features. For example, the following

section will classify mobile traffic applications using several popular ML algorithms such as Random Forest, Decision Trees, support vector machine (SVM), etc.

4 Early-State Traffic Flow Classification

The process of the early-state traffic flow classification, which implemented on the platform of BML, can be divided into 2 steps: feature extraction and classification as shown in Fig. 5. The first step extracts and defines useful features for the classification model from the collected traffic flows of each application. It involves several pre-processing data processes, such as feature collection, data cleaning, and data transforming, to create a training dataset for the classifier. The second step builds classification models using different ML algorithms in InfoSphere, Spark, R, Matlab, etc. environments to implement the classification task. Finally, the classifiers will be applied to the SON and then new coming traffic flows will be classified for further processes such as network QoS control for each application.

4.1 Feature Extraction

Feature extraction and selection are essential steps deciding the accuracy and efficiency of classification models; therefore, the observed features must accurately represent the traffic behaviors of each application. As discussed in the introduction section, many studies such as [7] [9] proved that packet size and the number of packets carry enough information for the early-stage traffic classification. Generally, mobile applications are based on SSL/TLS built on the top of the TCP/IP protocol for security purpose [17].

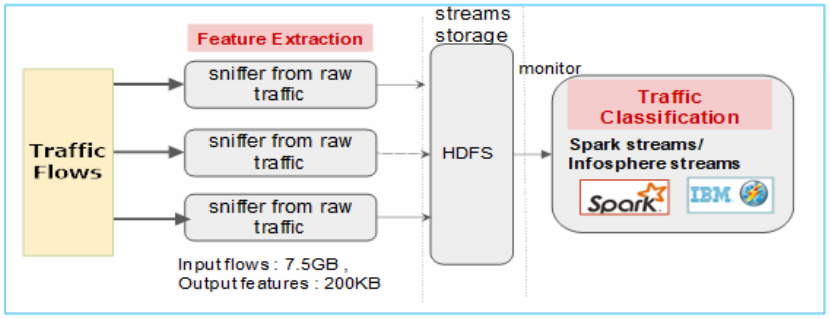


Figure 5. Traffic flow classification process

Table 1. Feature extraction of several traffic flows

connection (A-B)	packet.1.size	packet.2.size	packet.3.size	packet.4.size	packet.5.size	packet.count.A	packet.count.B	packet.count.A&B	byte.count.A	byte.count.B	byte.count.A&B	dport	sport	Application
10.100.11.91:53787 > 64.233.189.17:443	213	149	149	1398	639	12	3	15	5964	1865	7829	443	53787	SSL.GMail
10.100.11.91:53791 > 64.233.189.18:443	309	117	1398	527	309	11	3	14	5399	1721	7120	443	53791	SSL.GMail
10.100.11.91:53792 > 64.233.189.18:443	181	165	165	1398	655	12	3	15	5900	2057	7957	443	53792	SSL.GMail
10.100.11.91:53793 > 64.233.189.18:443	485	1397	357	469	149	14	2	16	7910	1060	8970	443	53793	SSL.GMail
10.100.11.91:53795 > 64.233.189.18:443	469	1397	981	1397	1397	60	3	63	63387	1849	65236	443	53795	SSL.GMail
10.100.11.90:49384 > 74.125.101.103:443	1229	1398	1398	1398	1398	77	1	78	99642	1398	101040	443	49384	SSL.YouTube
10.100.11.90:49379 > 140.113.14.38:443	1253	1398	340	1253	1398	13	1	14	8998	1398	10396	443	49379	SSL.YouTube
10.100.11.90:49380 > 140.113.14.38:443	1269	1398	564	1253	1398	13	1	14	9238	1398	10636	443	49380	SSL.YouTube
10.100.11.90:49382 > 140.113.14.15:443	1173	469	1333	389	1141	12	1	13	7718	389	8107	443	49382	SSL.YouTube
10.100.11.90:49416 > 74.125.101.212:443	1232	1398	1398	1398	1398	77	1	78	99446	1398	100844	443	49416	SSL.YouTube
10.100.11.90:49513 > 96.17.72.64:443	57	1398	1398	1398	1398	36	1	37	41341	45	41386	443	49513	SSL.Facebook
10.100.11.90:49517 > 23.2.16.11:443	57	1398	1398	1398	1398	55	1	56	68668	1398	70066	443	49517	SSL.Facebook
10.100.11.90:49495 > 31.13.95.8:443	57	45	45	1001	45	12	2	14	5820	1046	6866	443	49495	SSL.Facebook
10.100.11.90:49498 > 31.13.95.5:443	741	53	165	117	85	11	3	14	4555	255	4810	443	49498	SSL.Facebook
10.100.11.90:49377 > 74.125.203.154:443	1189	725	1173	725	53	11	1	12	7245	725	7970	443	49377	SSL.Google
10.100.11.90:49411 > 74.125.203.95:443	1397	1397	1397	277	1397	14	3	17	9133	3098	12231	443	49411	SSL.Google
10.100.11.90:49441 > 74.125.203.95:443	1397	1397	1397	101	1397	14	2	16	8797	842	9639	443	49441	SSL.Google
10.100.11.90:49915 > 192.229.145.200:443	325	853	1410	1410	929	15	1	16	11248	853	12101	443	49915	SSL.Skype
10.100.11.90:49918 > 192.229.145.200:443	325	853	1410	1315	341	14	1	15	10240	853	11093	443	49918	SSL.Skype
10.100.11.90:49917 > 192.229.145.200:443	325	853	1410	1395	325	14	1	15	10304	853	11157	443	49917	SSL.Skype

When an application session starts, there are several negotiation stages between the client side and the server side. For example, each TCP flow begins with the TCP three-way handshake. This process consists of several continuous interaction rounds, which contain one or multiple messages (layer 7 messages). Those messages are segmented, in other words, the TCP layer receives encrypted data from the above layer and adds a TCP header to create TCP segments. After that, each TCP segment is encapsulated into IP packets and exchanged with a peer. In TCP connection, TCP packets do not include a session identifier so that both endpoints identify the TCP session through the client's IP address and the port number. Here, the system captures incoming IP packets and parses traffic flows to extract the headers of IP packets. A traffic flow is defined as a bi-directional ordered sequence of packets, which consists of the same 5-tuple: source IP, destination IP, source port, destination port, and transport layer protocol. In this study, the features are extracted from the application layer and transport layer perspectives [5]. They consist of the number of the packets and the packet sizes of the first interaction round, the size of the first 5 packets of each TCP/UDP flow, the source port and the destination port of the connection. Table 1 is an example of some samples of several flows, each input data sample contains 13 features.

4.2 Machine Learning Algorithms for traffic flow classification

This section briefly introduces characteristics of several state-of-the-art ML algorithms that are used in this study

Naïve Bayes is a straightforward and powerful probabilistic classifier, which computes the probability of a data sample that belongs to each class by using Bayes' theorem. It assumes that all features are conditional independence with one another. That means the presence of one feature does not affect the presence of others. As a result, this model is easy to train and can provide impressive performance, even if it is working on a data set with millions of data samples.

Gradient Boosted Tree (GBT): GBT, an ensemble of decision trees, combines simple parameterized functions to achieve high accuracy for prediction and classification models. In other words, it iteratively trains multiple decision trees to minimize the cost function and provide more accurate prediction model by changing complex interactions in a simple fashion.

Random Forest (RF): RF is also known as an ensemble of decision trees computed in parallel fashion on a dataset by using random subsets to improve the multiclass classification rate and to overcome the over-fitting problem. To classify a data sample, different trees in the forest will learn on different subsets of data, and then they make classification on their own. Finally, the final class of the testing data sample is assigned to a class that has majority votes.

SVM is a popular supervised ML algorithm for pattern recognition, such as classification, regression, and abnormal detection. It analyzes training data to find the largest margin for linear and non-linear classifiers. While many ML algorithms are memory-based methods, that means the kernel function, which implicitly maps their inputs into high-dimensional feature space, must be calculated for all pairs of training points. As a result, it needs a huge amount of calculation during the training stage so that they easily lead to excessive computation and computing time, SVM, in another way, is a decision algorithm that classifies a new sample only depend on calculating a subset of the training data.

Neural network (NN): NN is a non-linear algorithm in which the computational scheme is based on the structure and functions of biological neural networks. It represents for popular technologies to provide

the best solution for many problems in which relationships between multiple input and output variables are complex. Recently, NN has been applied in many fields, including pattern recognition, speech recognition, image recognition, and natural language processing.

4.3 Experimental implementation

This subsection evaluates and compares the classification performance of several state-of-the-art ML algorithms to find out relevant algorithms for traffic classifications. Generally, classification models are form of supervised learnings, which include two phases, training phase and testing phase. For each experiment, a training dataset of 21000 traffic flow samples of 7 applications, 3000 flows for each application, was chosen randomly from the collected data from different connections. The input of each data sample consists of 13 features and the output is an application label as described in Table 1. Moreover, since the common values of the features are in different ranges, they need to be normalized into relevant ranges by applying a z-score normalization on all data columns. Finally, to evaluate the performance of each ML algorithm, a testing dataset of 3500 testing flow samples (500 flows for each application) was chosen randomly, and it must differ from the training dataset.

4.4 Experimental result

Firstly, we analyze the model performance through the confusion matrix of classification results. Table 2 & 3 show the classification result of SVM algorithm representing by number and percentages, respectively. For example, the number of 500 Facebook flows that are classified as Facebook, Gmail, Skype, Google, Instagram, YouTube, and Apple are 480, 0, 0, 14, 6, 0, 0, respectively. In other words, 96% Facebook flows were identified accurately, and the remainder were identified inaccurately, Google (2.8%) and Instagram (1.2%). That means some of Facebook flows have quite similar characteristics to those of Google and Instagram.

Table 2. Confusion matrix (number) of SVM Classification result

Classification result	SSL.Facebook	SSL.GMail	SSL.Skype	SSL.Google	SSL.Instagram	SSL.YouTube	HTTP.Apple
SSL.Facebook	480	0	0	14	6	0	0
SSL.GMail	0	493	0	3	4	0	0
SSL.Skype	0	0	496	0	0	2	2
SSL.Google	9	5	0	466	12	8	0
SSL.Instagram	13	7	3	7	465	1	4
SSL.YouTube	3	0	0	4	11	482	0
HTTP.Apple	15	2	1	8	6	0	468

Table 3. Confusion matrix (percentage %) of SVM Classification result

Classification result	SSL.Facebook	SSL.GMail	SSL.Skype	SSL.Google	SSL.Instagram	SSL.YouTube	HTTP.Apple
SSL.Facebook	96	0	0	2.8	1.2	0	0
SSL.GMail	0	98.6	0	0.6	0.8	0	0
SSL.Skype	0	0	99.2	0	0	0.4	0.4
SSL.Google	1.8	1	0	93.2	2.4	1.6	0
SSL.Instagram	2.6	1.4	0.6	1.4	93	0.2	0.8
SSL.YouTube	0.6	0	0	0.8	2.2	96.4	0
HTTP.Apple	3	0.4	0.2	1.6	1.2	0	93.6

Table 4. Classification result of different applications and ML algorithms

Application	SSL.Facebook	SSL.Gmail	SSL.Skype	SSL.Google	SSL.Instagram	SSL.YouTube	SSL.Apple
Naïve Bayes	91.4	92.2	95.8	89.8	89.2	96.2	93.6
SVM	96	98.6	99.2	93.2	93	96.4	93.6
NN	96.4	92.6	97.8	94	96.8	99	98
GBT	98.4	99.2	100	99	99.4	100	98.6
Random Forest	99.4	99.6	100	99.2	99	100	99.8

Table 4 summarizes and compares the classification performance of different ML algorithms for different applications. It is noticeable that all the algorithms can achieve high accuracy for identifying those applications, however, different models give different performances for different applications. In general, Table 5 summarizes the performance of each ML algorithm, as can be seen, Naïve Bayes gets worse

performance, SVM and Neural Network give a quite similar performance, and they are better than Naïve Bayes, while GBT and RF give the best performance for all traffic flows.

Table 5. Average classification performance of ML algorithms

Machine Learning Algorithm	Accuracy (%)
Naïve Bayes	92.60
SVM (Support Vector Machine)	95.71
Neural network	96.37
GBT	99.23
Random Forest	99.57

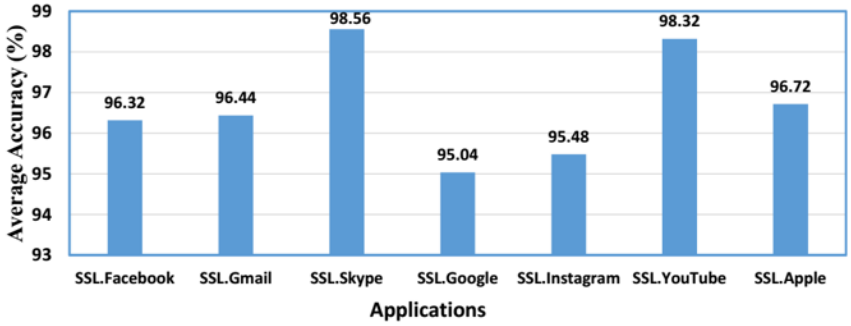


Figure 6. Average classification accuracy of applications

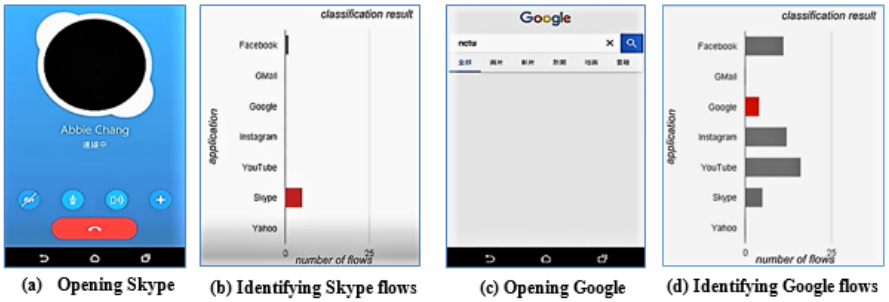


Figure 7. Online classification results

Moreover, during analyze the results, we noticed that some applications such as Google and Instagram are more difficult to identify than for others due to the fact that some of their flow features (e.g. the first packet size) vary significantly among flows of different connections and different user's actions on UEs (e.g. send an email, open an email), besides, they are also easy to confuse with those of other traffic flows. For more details, Fig.6 shows the average classification accuracy of each type of traffic flows, it is clear that YouTube and Skype are easier to identify and get the highest classification accuracies.

Online classification: this experiment uses SVM algorithm to identify online traffic flows of mobile applications. A Streaming application is usually described as a directed graph composing individual computing entities that interconnect and operate on a platform; therefore, it often integrates a monitoring application for scheduling and managing purposes. In this test, a UE was used to access mobile applications, and their traffic flows were classified into relevant applications. Moreover, with each application, we played different actions that may be sensitive the traffic flow behavior. For example, we analyzed typical action on Gmail such as send an email, send a reply, open an email, open chats, etc. The number of traffic flows of each application was also accumulated. Fig. 7 (a) illustrates that Skype application is opening on a UE, 7(b) shows the identification result and the flow accumulation of each application. Similarly, Fig. 7(c)&(d) show the result when the user is opening Google application. In summary, the classification model is able to classify online or streaming traffic flows with high accuracies.

4.5 Computing Performance of InfoSphere Cluster

This subsection evaluates the computing performance of classification system. An online classification model was deployed in the InfoSphere cluster [18], which consists of one computing master and 4 slaves, their names and IP address are shown in Fig.8. In this experiment, we randomly generated a stream of 10.000.000 data samples as classification testing dataset, then Kafka received the data, created topics, and produced data to InfoSphere, after that, InfoSphere classified the incoming flows by its SVM classifier. Fig. 9 shows the computing state of the InfoSphere cluster in which computing job is equally distributed for all the slaves under the control of the master. Fig. 10 summarizes the computing performance, it shows the input flow rate, output flow rate, and the latency. As can be seen, with different the input speed, the cluster classified all the data samples smoothly with a small time for scheduling and processing, the maximum speed is around 90.000 flows/second with low total latency (average about 10ms).

1	Name	2	IP
●	invpm27		10.0.20.67
●	invpm28		10.0.20.68
●	invpm29		10.0.20.69
●	invpm31		10.0.20.71
●	master		10.0.20.70

Figure 8. InfoSphere computing cluster

Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
invpm27:50010 (10.0.20.67:50010)	2	In Service	9.99 GB	4 KB	6.07 GB	3.92 GB	0	4 KB (0%)	0	2.7.1
invpm29:50010 (10.0.20.69:50010)	2	In Service	9.99 GB	4 KB	6.07 GB	3.92 GB	0	4 KB (0%)	0	2.7.1
invpm28:50010 (10.0.20.68:50010)	2	In Service	9.99 GB	4 KB	6.07 GB	3.92 GB	0	4 KB (0%)	0	2.7.1
master:50010 (10.0.20.70:50010)	2	In Service	29.98 GB	4 KB	10.02 GB	19.97 GB	0	4 KB (0%)	0	2.7.1
invpm31:50010 (10.0.20.71:50010)	2	In Service	9.99 GB	4 KB	6.07 GB	3.92 GB	0	4 KB (0%)	0	2.7.1

Figure 9. Computing state of InfoSphere cluster

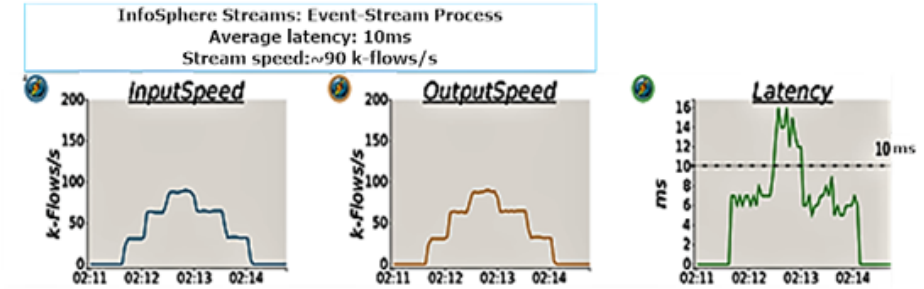


Figure 10. Computing speed and latency

In summary, the platform is powerful in supporting high computing capacity with low-latency. Moreover, it also provides an environment for collecting, transforming, and processing data of multiple streams from in inside and outside of the system. Therefore, it is relevant and powerful enough to be applied for the industrial case.

5 Network QoS Control for traffic applications Based on SDN

QoS control is a significant concept in mobile networks to prevent one mobile broadband application from degrading overall performance when it shares bandwidth with others. Therefore, how to manage QoS for multiple

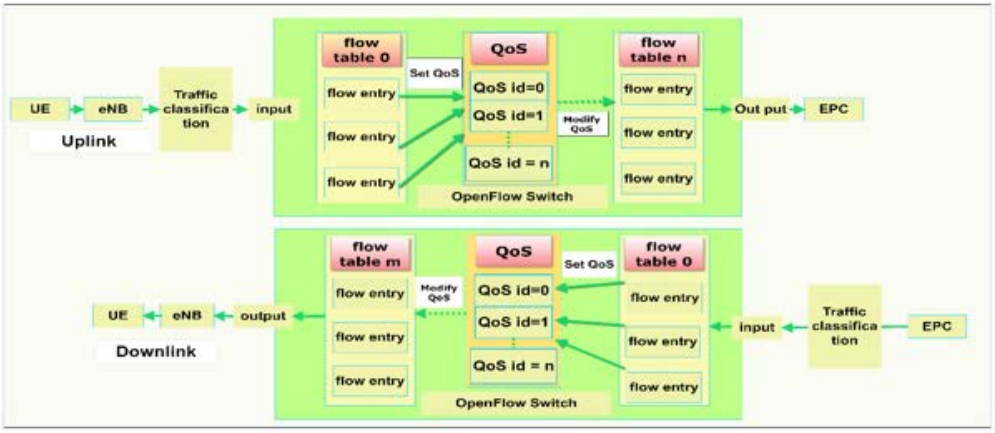


Figure 11. QoS control for traffic flows based on traffic classification

traffic flows is important to network operators in improving both user experience and network efficiency. Each application requires a QoS guarantee, for example, Skype and video streaming require small delay and jitter, while other data communications like Gmail, Google are more sensitive to packet loss. Hence, the network operator must preserve a relevant bandwidth for each application. The traditional technologies for network QoS control are Integrated Service (IntServ) and Differentiated Service (DiffServ). However, the former is too complex and not scalable, On the other hand, the latter is less complex, but does not provide strong QoS guarantees. Fortunately, SDN/NFV are emerging technologies that play important roles in enabling new approaches for QoS control such as the utilization of meter tables, ingress policing, Queue management, Virtual Network Embedding, etc. [1][19][20]. This section integrates the traffic flow identification system with a QoS management system based on SDN as described in Fig. 11. After a traffic flow is identified, it will be assigned a relevant bandwidth value by using one of above technologies. Among those methods, using meter tables and QoS queue in the flow table entries of OpenFlow Switch are the most relevant and effective approaches for controlling the QoS of traffic flows. The SDN controller utilizes the result of the classification application to setups flow table entries for each connection to instruct how the flows (packets) are executed. For example, in the meter table case, it uses meters, which are attached directly to flow entries, to measure and control the data rate of packets. Each meter table consists of multiple meter entries, which can support for different applications with different QoS policies. On other hand, the QoS queue method assigns a QoS ID for each application in the flow table entries. Fig. 11 describes the abstract process of implementing network QoS control in the mobile network for uplink and downlink directions. In this framework, the classification model is deployed close to Base Stations (edge network), this is important to support a variety of innovative applications and services quickly with very low latency like mobile edge computing (MEC) and Fog computing. Moreover, once the controller receives a traffic flow of an identified application, it will store the basic information of the connection, such as source IP address, destination IP address, application type, etc. Finally, based on these information, the SDN controller installs flow table entries and QoS policies to control the flows for both uplink and downlink directions. For example, in our case study, we set a bandwidth 3Mbps for YouTube application, then a UE was used to open YouTube video.

The result in Fig. 12 shows that the bandwidth provided for YouTube is 2.96 Mbps (around 3 Mbps).

Figure 12. QoS control result for YouTube

6 Conclusion

This study proposed a comprehensive platform based on big data, ML, and SDN/NFV to empower the SON of 5G. Moreover, the process of building SON applications, such as data collection, storage, analytics, and virtualization were also introduced . Specifically, in the case study, we applied various state-of-the-art ML algorithms to classify accurately mobile applications at an early stage, then traffic flows of each application were preserved a relevant bandwidth controlled by the network QoS control by using SDN controller. This is crucial to the SON in ensuring that an application can work at right function, the network resources are utilized effectively, and user experience is guaranteed. Especially, both offline and online learning models were considered and implemented successfully. In the future, the authors focus on utilizing the results of the study to develop a comprehensive architecture for 5G SON and 5G MEC based on P4, ONOS, and CORD (Central Office Re-architected as a Datacenter) platforms, which are considered as the key elements and solutions of SDN/NFV technologies for 5G.

ACKNOWLEDGMENT

This paper is particularly supported by "Aiming for the SPROUT Project - Center for Open Intelligent Connectivity" of National Chiao Tung University and Ministry of Education, Taiwan, R.O.C. and the Ministry of Science and Technology of Taiwan under Grants: MOST 106-2221-E-009-008

REFERENCES

[1] Y. J. Chen, L. C. Wang, F. Y. Lin, and B. S. Lin, “Deterministic Quality of Service Guarantee for Dynamic Service Chaining in Software Defined Networking,” *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 4, pp. 991–1002, 2017.

[2] I. C. Hsieh, L. P. Tung, and B. S. P. Lin, “On the classification of mobile broadband applications,” *IEEE Int. Work. Comput. Aided Model. Des. Commun. Links Networks, CAMAD*, pp. 128–134, 2016.

[3] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, “Internet traffic classification by aggregating correlated naive bayes predictions,” *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 1, pp. 5–15, 2013.

[4] W. Ke, Y. Wang, X. Lei, and B. Wei, “Spark-Based Feature Selection Algorithm of Network Traffic Classification,” *2017 13th Int. Conf. Comput. Intell. Secur.*, pp. 140–144, 2017.

[5] N. F. Huang, G. Y. Jai, H. C. Chao, Y. J. Tzang, and H. Y. Chang, “Application traffic classification at the early stage by characterizing application rounds,” *Inf. Sci. (Ny).*, vol. 232, no. 22, pp. 130–142, 2013.

[6] L. Peng, B. Yang, Y. Chen, and T. Wu, “How many packets are most effective for early stage traffic identification: An experimental study,” *China Commun.*, vol. 11, no. 9, pp. 183–193, 2014.

[7] G. Aceto, D. Ciuonzo, G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, “Traffic Classification of Mobile Apps through Traffic Classification of Mobile Apps through,” no. September, pp. 2–7, 2017.

[8] R. Alshammari and A. N. Zincir-Heywood, “Identification of VoIP encrypted traffic using a machine learning approach,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 27, no. 1, pp. 77–92, 2015.

[9] M. Shafiq, X. Yu, and D. Wang, "Robust Feature Selection for IM Applications at Early Stage Traffic Classification Using Machine Learning Algorithms," *2017 IEEE 19th Int. Conf. High Perform. Comput. Commun. IEEE 15th Int. Conf. Smart City; IEEE 3rd Int. Conf. Data Sci. Syst.*, pp. 239–245, 2017.

[10] B. Hullár, S. Laki, and A. György, "Efficient methods for early protocol identification," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 10, pp. 1907–1918, 2014.

[11] B. P. Lin, F. J. Lin, and L. Tung, "The Roles of 5G Mobile Broadband in the Development of IoT, Big Data, Cloud and SDN," *Commun. Netw.*, vol. 8, no, no. February, pp. 9–21, 2016.

[12] B. P. Lin, L. Tung, F. Tseng, I. Hsieh, Y. Wang, and S. Chou, "Performance Estimation of MAR for Outdoor Navigation Applications based on 5G Mobile Broadband by using Smart Mobile Devices."

[13] B. S. P. Lin, W. H. Tsai, C. C. Wu, P. H. Hsu, J. Y. Huang, and T. H. Liu, "The design of cloud-based 4G/LTE for mobile augmented reality with smart mobile devices," *Proc. - 2013 IEEE 7th Int. Symp. Serv. Syst. Eng. SOSE 2013*, pp. 561–566, 2013.

[14] D. Sinh, L. Le, L. Tung, and B. P. Lin, "The Challenges of Applying SDN / NFV for 5G & IoT," in *The 14th IEEE - VTS: Asia Pacific Wireless Communications Symposium (APWCS), Incheon, Korea, August 2017*.

[15] Le Luong Vy; Li-Ping Tung; Bao-Shuh Paul Lin, "Big data and machine learning driven handover management and forecasting," in *IEEE Standards for Communications and Networking (CSCN), 2017 IEEE Conference on*, 2017, pp. 214–219.

[16] L. Le, D. Sinh, L. Tung, and B. P. Lin, "A Practical Model for Traffic Forecasting based on Big Data , Machine-learning , and Network KPIs," pp. 3–6, 2018.

[17] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing Android Encrypted Network Traffic to Identify User Actions," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 1, pp. 114–125, 2016.

[18] B. S. Lin *et al.*, "The design of big data analytics for testing & measurement and traffic flow on an experimental 4G/LTE network," *2015 24th Wirel. Opt. Commun. Conf. WOCC 2015*, pp. 40–44, 2015.

[19] D. L. C. Dutra, M. Bagaa, T. Taleb, and K. Samdanis, "Ensuring End-to-End QoS Based on Multi-Paths Routing Using SDN Technology," *GLOBECOM 2017 - 2017 IEEE Glob. Commun. Conf.*, pp. 1–6, 2017.

[20] H. Krishna, N. L. M. Van Adrichem, and F. A. Kuipers, "Providing bandwidth guarantees with OpenFlow," *2016 IEEE Symp. Commun. Veh. Technol. Benelux, SCVT 2016*, pp. 0–5, 2016.

SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 6, NO. 2
ISSN: 2054 -7420

A Simple Greedy Algorithm for Energy-Efficient Communication in Small Multi-Interface Wireless Networks

[1,2]Christos Kaklamanis, [1]Stavros Maras, [1,2]Evi Papaioannou
[1]University of Patras, Patras, Greece;
[2]CTI "Diophantus", Patras, Greece;
kakl@ceid.upatras.gr; maras@ceid.upatras.gr; papaioan@ceid.upatras.gr

ABSTRACT

Wireless networks have become extremely popular recently due to the wide range of applications they support and also because sophisticated and affordable wireless devices like smartphones, tablets, etc have actually become part of our everyday life. Wireless devices have heterogeneous characteristics, like computational power, energy-consumption levels, supported communication protocols. Modern wireless devices are usually equipped with multiple radio interfaces like, WiFi, GPRS, Bluetooth, and can switch between different communication networks for meeting connectivity requirements and, thus improving quality of service and data collection perspectives. Establishing a connection between any two such devices requires that they are close and share at least one common available interface. If communication is established between two wireless devices, then the involved communication cost reflects the energy these devices consume and equals the cost for activating a particular common interface. In this setting, the objective is to suggest a cost-efficient interface activation plan which can guarantee low-cost communication for any two such wireless devices.

We model this practical problem as an instance of the Spanning Tree problem in an appropriately defined multigraph corresponding to the actual multi-interface wireless network. When connectivity is feasible, we propose and experimentally evaluate a simple greedy algorithm indicating which interfaces must be activated so that cost-efficient connectivity is established between any two wireless devices in the network.

Keywords: Small multi-interface wireless networks; Energy-efficient communication; Connectivity; Greedy algorithm.

1 Introduction

Wireless networks have become extremely popular during the recent years mainly due to the wide range of applications they support and also due to the fact that sophisticated and affordable wireless devices like smartphones, tablets, etc have actually become part of our everyday life. Wireless networks can be stand-alone network components (like for example a wireless network for a class or lab) or parts of larger networks and the Internet. Wireless devices have heterogeneous characteristics: they have different computational power, their energy-consumption levels vary, they support different communication protocols. Modern wireless devices are usually equipped with multiple radio interfaces like, for example,

DOI: 10.14738/tnc.62.4500
Publication Date: 1st May 2018
URL: http://dx.doi.org/10.14738/tnc.62.4500

WiFi, GPRS, Bluetooth. Therefore, they can switch between different communication networks for meeting connectivity requirements and improving quality of service. However, determining which interface must be activated on a wireless device depends on technical specifications of the device, communication requirements, connectivity constraints, necessary energy consumption. Even in the case that all other factors are neutral, energy consumption plays a crucial role for the selection of the interface to be activated, since, as long as a device runs out of battery, it can no longer be part of a wireless network. Besides benefits related to network infrastructure also data collection perspectives can be efficiently supported via the use of appropriate interfeces in multi-interface wireless networks.

We study a communication problem arising in wireless networks supporting multiple interfaces. These networks are composed of nodes which are wireless devices supporting some wireless interfaces. Communication between two such nodes requires the existence of at least one common interface and spatial proximity so that this specific shared interface can support their communication. If these conditions hold, then communication can be established. The involved cost essentially reflects the energy consumed and equals the cost of activating a particular interface which both nodes share. The objective is to activate interfaces at network nodes so that some connectivity property is maintained and the total activation cost is minimized. Various communication problems arise in multi-interface wireless networks based on the required connectivity property. We consider the problem termed as ConMI in [1] or Connectivity in [9]. In particular, we require that communication is established among all network nodes. The energy consumed by each device for the activation of a specific interface may vary substantially. Therefore, two cases are distinguished according to the interface activation costs: the more general one is when the activation cost for some interface is not the same at all network nodes; this is the heterogeneous case. In the homogeneous [1] or uniform cost [9] case, the cost of activating a particular interface is the same at all network nodes. Another important variant to some of the problems faced in multi-interface wireless networks concerns the total number of available interfaces supported in the overall network. The corresponding problems are in bounded or unbounded form depending on whether the total number of available interfaces is provided as a fixed constant or part of the input, respectively [9]. The unbounded version of such problems can be particularly useful for analytical results while the bounded version is more representative of practical cases.

Previous relevant work

Recent technological advances and a wide range of supported applications have made multi-interface wireless networks a very popular and wide-spread communication infrastructure. The study of communication problems arising in multi-interface wireless networks has attracted the interest of the research community. The key idea is to exploit the heterogeneity of the interfaces available in modern devices for reducing energy consumption and, consequently, extending network lifetime. Several well-known combinatorial optimization problems are then reconsidered with respect to this new feature.

Several basic problems studied for “traditional” wired and wireless networks have been reconsidered in this new setting [2], with an emphasis on problems related to network connectivity [3, 5] and routing [4]. Requirements for efficiency in energy consumption increase the complexity of these problems and raise new challenges. In [6], cost minimization in multi-interface wireless networks was studied. More precisely, given a graph representing desired connections between network nodes, the objective is to establish all graph edges by activating interfaces at network nodes of a minimum total cost. Several

variations of the problem are considered depending on the topology of the input graph (e.g., complete graphs, trees, planar graphs, bounded-degree graphs, general graphs) and on whether the number of interfaces is part of the input or a fixed constant. [6] considers both unit-cost interfaces and more general homogeneous instances. ConMI has been introduced in [7] which studies homogeneous instances of the problem. ConMI is proved to be APX-hard even when the graph modeling the network has a very special structure and the number of available interfaces is small (e.g., 2). In [7], a 2-approximation algorithm is presented by exploiting the relation of ConMI on homogeneous instances with the minimum spanning tree on an appropriately defined edge-weighted graph. Furthermore, [1] suggests an improved (3/2+ε)-approximation algorithm for ConMI. The algorithm is based on a challenging technique [10] that makes use of an "almost" minimum spanning tree in an appropriately defined hypergraph and transforms it to an efficient solution for connectivity. Better approximation bounds are obtained for special cases of ConMI such as the case of unit-cost interfaces. [9] provides a comprehensive survey on results from the recent relevant literature.

In this work, we focus on the heterogeneous, bounded form of ConMI. We model this practical problem as an instance of finding a spanning tree in an appropriately defined multigraph corresponding to the actual multi-interface wireless network. When connectivity is feasible, we propose, analyze and experimentally evaluate a simple greedy algorithm which indicates which interfaces must be activated so that cost-efficient connectivity is established between any two wireless devices in the network. We embed this technique into a proof-of-concept application for establishing energy-efficient communication within small groups of users (in classrooms, labs, meeting rooms, game rooms, etc) equipped with wireless devices (e.g., smartphones or tablets) supporting multiple interfaces. From a practical point of view, network infrastructure and data collection perspectives can highly benefit from the efficient management of available interfaces in multi-interface wireless networks.

The rest of the paper is structured as follows. In Section 2, we provide technical details regarding definitions and notation. In Section 3, we discuss our greedy approach to the connectivity problem in multi-interface wireless networks. We present experimental finding in Section 4 and conclude our report in Section 5.

2 Preliminaries: Definitions and Notation

In general, a multi-interface wireless network is modelled by a graph G = (V, E), where V represents the set of devices composing the network and E is the set of possible connections defined according to the distance between devices and the available interfaces that they share. Each $v \in V$ is associated with a set of available interfaces W(v). The set of all the possible available interfaces in the network is then determined by $\cup_{v \in V} W(v)$; we denote by k the cardinality of this set.

We say that a connection is established when the endpoint of the corresponding edge share at least one active interface. So, in our model, an edge $e_s = (u,v)_s$ exists for every interface s both u and v share, yielding a multigraph G which is assumed to be undirected and connected. If an interface s is activated at some node u, then u consumes some energy $c_u(s)$ for keeping s active and obtains a maximum communication bandwidth $b_u(s)$ with all its neighbors that share interface s. Furthermore, each possible edge $(u,v)_s$ has a cost equal to $c_u(s)+c_v(s)$.

For globally characterizing the interfaces each device supports, we use an interface assignment function W which covers graph G = (V, E), i.e., for each $(u,v) \in E$ it holds $W(u) \cap W(v) \neq \varnothing$. Our objective is to activate

interfaces at the nodes of V so that the resulting graph G' is a spanning tree for G, i.e. G' is connected, acyclic and spans all nodes of V. In the case when the resulting spanning tree G' has a minimum total edge-weight, we have a Minimum Spanning Tree (MST) for G.

Two classical deterministic greedy algorithms for constructing Minimum Spanning Trees are due to Kruskal [8] and Prim [11]. Both algorithms proceed by successively adding edges of smallest weight from those edges with a specified property that have not already been used. The main difference is the criterion used to select the next edge or edges to be added in each step. They are particularly simple and in fact solve the same problem by applying the greedy approach in two different ways and both always yield an optimal solution.

Kruskal's algorithm starts with an edge in the graph with minimum weight and builds the spanning tree by successively adding edges one by one into a growing spanning tree. It processes the edges in order of their weight values, from smallest to largest, including into the growing MST each edge which does not form a cycle with edges previously added. It stops after $|V|-1$ edges have been added. Kruskal's algorithm computes the MST of any connected edge-weighted graph with E edges and V vertices in time proportional to $|E|\log|E|$ (in the worst case) since sorting is the most time consuming operation.

Prim's algorithm constructs a minimum spanning tree incrementally, in a step-by-step fashion via a sequence of expanding subtrees. The initial subtree of the sequence consists of a single vertex selected arbitrarily from the set V of the vertices of the given graph. In each successive step, the algorithm expands the current tree greedily by simply adding to it the nearest vertex not in the tree. The distance of such a vertex is determined by the weight of the edge connecting it to the tree. In the case of at least two candidate nearest vertices, ties can be broken arbitrarily. The algorithm terminates when all vertices of the graph have been included in the spanning tree. Since the algorithm expands a tree by exactly one vertex during each step, the total number of required steps is n-1, where n is the number of vertices of V. The tree generated by the algorithm is obtained as the set of edges used for the tree expansions.

More precisely, the algorithm maintains two disjoint sets of vertices: one containing vertices that are in the growing spanning tree and another containing vertices not in the growing spanning tree. Then, it selects the lowest-cost vertex which is connected to the growing spanning tree but is not in the growing spanning tree and inserts it into the growing spanning tree. In order to avoid the creation of cycles, the algorithm marks the vertices which have been already selected and considers only those vertices that are not marked. Since each vertex is considered only once, the time complexity of the Prim's algorithm is $O((|V|+|E|)\log|V|)$.

3 Algorithm G_{MU}: A Simple Greedy Heuristic for Connectivity

3.1 Description and analysis

G_{MU} is a deterministic, greedy algorithm for connectivity in multi-interface wireless networks. It receives as input a graph G = (V,E) corresponding to a wireless network whose nodes support multiple interfaces. For each vertex $v \in V$ (representing a network node), information regarding supported interfaces and corresponding activation cost is provided. Each edge $e_s \in E$ connects pairs (u,v) of distinct vertices of V sharing interface s and has an associated weight equal to the sum of the activation cost of interface s at nodes u and v. Multiple edges are allowed between pairs of nodes sharing multiple (i.e., more than one)

interfaces. If $W(u) \cap W(v) \neq \varnothing$ for all $(u,v) \in E$, the algorithm returns a spanning tree $T = (V_T, E_T)$ for G, that is $V_T=V_G$, $E_T \subseteq E_G$.

More precisely, our algorithm works as follows. For a given input graph G = (V, E), the algorithm first checks whether there exists $(u,v) \in E$ for which the condition $W(u) \cap W(v) \neq \varnothing$ is false. If so, the algorithm terminates and fails to compute a spanning tree for G. If $W(u) \cap W(v) \neq \varnothing$ is true for all $(u,v) \in E$, a vertex set V_T is created and an arbitrary vertex $v \in V_G$ is added to it. Furthermore, a list L of edges is created where all edges $(u,v) \in E$ with $u \in V_T$ and $v \in V\backslash V_T$ are added. In this way, the formation of cycles in V_T is avoided. L is sorted in ascending order in terms of edge-weights. Then, the main loop of the algorithm is repeated until $V_T=V_G$. At each round, the first element of L (i.e., the edge of L of minimum weight) is added to T and L is updated so as to only include edges whose one endpoint belongs to V_T and the other is strictly in $V\backslash V_T$. Eventually, the algorithm terminates and returns a spanning tree T for G.

Below, the pseudocode for our greedy approach is presented.

Algorithm G_{MU}

Input: connected multigraph $G = (V_G, E_G)$

Output: spanning tree $T = (V_T, E_T)$, $V_T=V_G$, $E_T \subseteq E_G$

1. if there exists $(u,v) \in E$ for which $W(u) \cap W(v) = \varnothing$ then FAIL & TERMINATE; otherwise

2. V_T := a vertex of V_G chosen uniformly at random

3. L := all edges $(u,v) \in E_G$ such that $u \in V_T$ and $v \in V\backslash V_T$

4. While $V_T \neq V_G$

5. T := T $\cup$ min[L]

6. Update L (add/remove elements, sort)

7. RETURN spanning tree T & TERMINATE

Lemma 1 (Correctness)

Algorithm $\mathbf{G_{MU}}$ produces a spanning tree T for G=(V,E) when $W(u) \cap W(v) \neq \varnothing$, $\forall\ u,v \in V$.

Proof

If there exists $(u,v) \in E$ for which $W(u) \cap W(v) = \varnothing$, the algorithm terminates and fails to compute a spanning tree T for G (Step 1). Otherwise, a vertex $v \in V_G$ is chosen uniformly at random and added to V_T (Step 2). A list L is created containing all edges $(u,v) \in E_G$ such that $u \in V_T$ and $v \in V\backslash V_T$ (Step 3). If $|V_G|=1$, a (minimum) spanning tree T is returned with $|V_T|=|V_G|=1$ and the algorithm terminates (Step 7). Otherwise, the while loop is executed (Step 4).

In order to show that the returned graph is indeed a tree, we have to show that the resulted graph is connected and acyclic. Consider an instance after a while loop is executed and let $\{v_1, v_2, \ldots, v_n\} \in V_T$, $n < |V_G|$ and $\{v_{n+1}, \ldots, v_{|VG|}\} \in V_G\backslash V_T$. Assume that $e=(v_i, v_j)$ is an edge currently considered for addition to T. This implies that one of the endpoints of e, say v_i, must be in V_T. Then, for a cycle to be created, v_j must be

a vertex already visited, i.e., a vertex also in V_T. This is a contradiction since every edge $e=(v_i, v_j)$ considered for addition to T must have $v_i \in V_T$ and $v_j \in V \backslash V_T$ (or vice-versa). Furthermore, T will eventually contain all nodes of G, since T is gradually augmented until $V_T=V_G$. This completes the proof of Lemma 1.

Lemma 2 (Time complexity)

Algorithm $\mathbf{G_{MU}}$ requires $O(|V^3|)$ steps.

Proof

Assuming that the number of available interfaces is $O(|V|)$, Step 1 requires $O(|V^3|)$ steps, since $O(|V^2|)$ pairs have to be checked. Furthermore, $O(|V|)$ repetitions of the while loop (Step 4) are required. Adding an edge to the list L (Step 5) requires $O(|V|^2)$ steps. Updating L requires $O(|V||E|)$ steps and sorting L requires $O(|V||E|\log|E|)$ steps (Step 6). This gives an overall time complexity of $O(|V|^3)$.

Lemma 3 (Activation cost)

Algorithm $\mathbf{G_{MU}}$ yields a total activation cost of O(V).

Proof

The spanning tree computed by algorithm G_{MU} for an input graph G=(V,E) representing a wireless network whose nodes support multiple interfaces has |V-1| edges. Assuming that c is the maximum activation cost taken over all available interfaces yields a maximum total cost of O(V).

3.2 Implementation

Software and hardware

For our experimental study, we used Python 3. We preferred Python to other popular programming languages, like C++ or Java, because it is friendly to use and easy to learn; yet, it is a powerful programming language which allows simple and flexible representations of networks as well as clear and concise expressions of network algorithms. Python can be used on many operating systems, providing a standard library and plenty of community-contributed modules. Python is developed under an open source license, making it freely usable and distributable [13]. Python 3, in particular, offers new important programming features and facilities as well as improved memory management.

Furthermore, we used NetworkX for our implementation. NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. NetworkX provides improved features regarding numerical linear algebra and drawing and can facilitate tasks including loading and storing networks in various data formats, generation of random and classic networks, analysis of network structure, building network models, drawing networks, and so on. NetworkX is freely usable and distributable under the terms of the 3-clause BSD License [12].

We implemented and executed our experiments on a mac OSX machine with an Intel Core i7 2,2 GHz processor and 16 GB 1333 MHz DDR3 RAM.

Input

The input multigraph corresponding to a wireless network supporting multiple interfaces can be provided to our code either manually or automatically.

Providing the input manually requires the use of NetworkX packages together with a basic program in Python. First, network nodes are defined. For each node a unique id and the interfaces it supports together with their activation cost must be given. Then, connections between nodes are defined in terms of graph edges. Multiple edges are allowed between each pair of network nodes; each edge corresponds to an interface shared by the nodes of the pair. For each edge, its endpoints and its weight must be given. The weight of an edge (u,v) corresponding to a shared interface s equals the sum of the activation cost of interface s at nodes u and v. Figure 1 shows an input instance provided manually.

```
G.add_node(1, Interface1 = "yes", Interface2 = "no", ActCost1 = 10)
G.add_node(2, Interface1 = "yes", Interface2 = "yes", ActCost1 = 5, ActCost2 = 6)
G.add_node(3, Interface1 = "yes", Interface2 = "yes", ActCost1 = 22, ActCost2 = 20)
G.add_node(4, Interface1 = "yes", Interface2 = "yes", ActCost1 = 19, ActCost2 = 10)
G.add_node(5, Interface1 = "yes", Interface2 = "no", ActCost1 = 13)
G.add_node(6, Interface1 = "yes", Interface2 = "yes", ActCost1 = 12, ActCost2 = 10)
G.add_edge(1,2,key=1,weight=15)
G.add_edge(1,3,key=1,weight=32)
G.add_edge(1,4,key=1,weight=29)
G.add_edge(1,5,key=1,weight=23)
G.add_edge(1,6,key=1,weight=22)
G.add_edge(2,3,key=1,weight=27)
G.add_edge(2,3,key=2,weight=26)
G.add_edge(2,4,key=1,weight=24)
G.add_edge(2,4,key=2,weight=16)
G.add_edge(2,5,key=1,weight=18)
G.add_edge(2,6,key=1,weight=17)
G.add_edge(2,6,key=2,weight=16)
G.add_edge(3,4,key=1,weight=41)
G.add_edge(3,4,key=2,weight=30)
G.add_edge(3,5,key=1,weight=35)
G.add_edge(3,6,key=1,weight=34)
G.add_edge(4,5,key=1,weight=32)
G.add_edge(4,6,key=2,weight=20)
G.add_edge(5,6,key=1,weight=25)
```

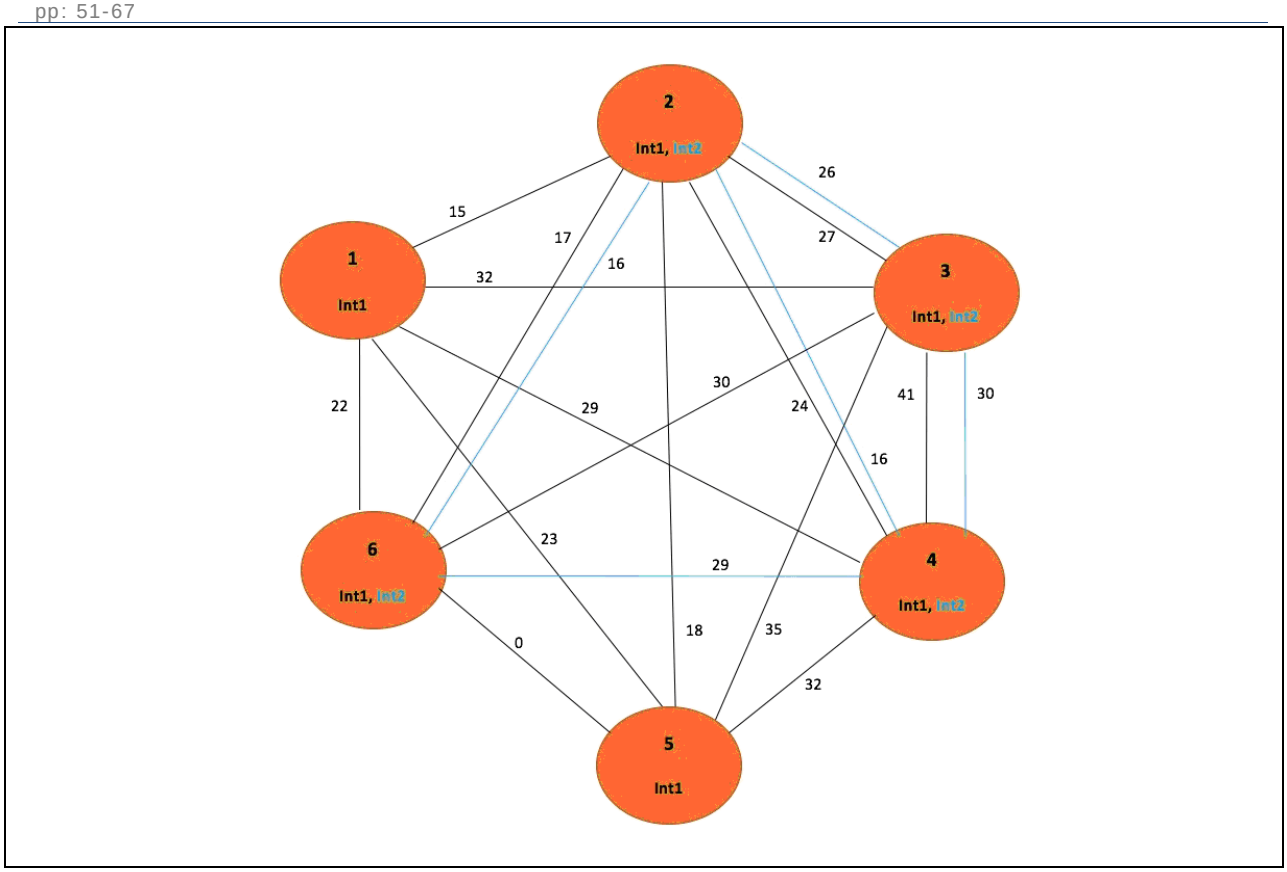


Figure 1: Manual generation of an input instance.

Providing the input automatically requires the execution of the "graph generator" function depicted in Figure 2.

```
# Graph generator function
#
# create the Vertices
#
for n in range(numOfVertices):
        n_corr = n+1
        G.add_node(n_corr)
        # randomly decide if Interface1 and Interface2 exists. If exists, add also an activation cost
        if random.randint(0,30)>0:
                G.node[n_corr]['Interface1'] = 'yes'
                G.node[n_corr]['ActCost1'] = random.randint(10,100)
        else:
                G.node[n_corr]['Interface1'] = 'no'
        if random.randint(0,30)>0:
                G.node[n_corr]['Interface2'] = 'yes'
                G.node[n_corr]['ActCost2'] = random.randint(10,100)
        else:
                G.node[n_corr]['Interface2'] = 'no'
for inj in G.nodes(data=True):
        if (inj[1]['Interface1'] == 'no') & (inj[1]['Interface2'] == 'no'):
                print('Not every pair of nodes have at least one common interface. Algorithm CANNOT execute')
                sys.exit()
#
# create the Edges
#
```

```
# add edges with Interface 1 in common
for ne in range(numOfEdges):
        n1 = random.randint(1,numOfVertices)
        n2 = random.randint(1,numOfVertices)
        # make sure the 2 randomly selected vertices are not the same
        if n1 == n2:
                n2 =+ 1
        if (G.node[n1]['Interface1'] == 'yes') & (G.node[n2]['Interface1'] == 'yes'):
                edgeWeight = G.node[n1]['ActCost1'] + G.node[n2]['ActCost1']
                G.add_edge(n1,n2,key=1,weight=edgeWeight)
# add edges with Interface 2 in common
for ne in range(numOfEdges):
        n1 = random.randint(1,numOfVertices)
        n2 = random.randint(1,numOfVertices)
        # make sure the 2 randomly selected vertices are not the same
        if n1 == n2:
                n2 =+ 1
        if (G.node[n1]['Interface2'] == 'yes') & (G.node[n2]['Interface2'] == 'yes'):
                edgeWeight = G.node[n1]['ActCost2'] + G.node[n2]['ActCost2']
                G.add_edge(n1,n2,key=2,weight=edgeWeight)
# print nodes and edges of graph
for g in G.nodes(data=True):
        print(g)
for i in G.edges(data=True, keys=True):
        print(i)
start=time.time() # start the timing of the algorithm
# check if every pair of nodes, connected with an edge, have at least one common interface
for itedge in G.edges(data=True):
        if not ((G.node[itedge[0]]['Interface1'] == 'yes') & (G.node[itedge[0]]['Interface1'] == 'yes')) \
        | ((G.node[itedge[0]]['Interface2'] == 'yes') & (G.node[itedge[0]]['Interface2'] == 'yes')):
                print('Not every pair of nodes have at least one common interface. Algorithm CANNOT execute')
                sys.exit()
```

Figure 2: "graph generator" function.

Our "graph generator" works as follows. Initially, the total number of graph vertices is randomly chosen via the use of function "random". Then, the input multigraph is generated by 4 consecutive "for" loops. The first "for" loop generates the vertices of the graph (attributing to each of them an id, supported interfaces and interface activation costs). The second "for" loop verifies that there exists at least one shared interface between each pair of vertices; if this is not the case, the "graph generator" terminates and restarts. The last two "for" loops are then used to generate the edges of the multigraph. The total number of edges is generated via the use of function "random". Endpoints are also randomly assigned to edges. Then, a verification process checks for edges having both endpoints assigned the same vertex. If no such edge exists, edge endpoints are checked for shared interfaces and the final input graph is produced.

Main part of the code

The core component of our code is presented in Figure 3, implements algorithm G_{MU} and works as follows. Initially, an arbitrary graph vertex is selected, assigned to variable ranV and printed on the screen. Then, a list L is created containing already explored vertices; ranV is inserted to L. Furthermore, a second list, namely sortEdges, is created containing edges to be considered for addition to the generated spanning tree; these are edges whose one endpoint is a vertex already explored (and, therefore, included in list L) and their other endpoint is a vertex not yet explored. The list sortEdges is sorted in ascending order of weight of included edges. The required spanning tree is then built through a main "while" loop. In particular, elements of the list sortEdges are printed on the screen, the edge e_{min} of the minimum weight

is assigned to variable ST (which corresponds to the spanning tree under generation) and the just explored endpoint of e_{min} is also added to L. Then, the list sortEdges is updated.

```
#
# st on conMI
#
# randomly select a vertex
ranV = random.randint(1,G.number_of_nodes())
print('Begin algorithm with randomly selected vertex:', ranV)
# create list of visited nodes and append the randomly selected vertex
l = []
l.append(ranV)
# find edge with minimum weight
sortEdges = sorted(G.edges([ranV],data='weight',keys=True),key=itemgetter(3))
while list(G.nodes()) != sorted(l):
        # print avalailable edges, select and prin the one with minimum weight
        print('edges to choose from:' ,sortEdges)
        print('edge in process',sortEdges[0])
        # add the selected edge to the spanning tree and its node to the list of already visited nodes
        ST.add_edge(sortEdges[0][0],sortEdges[0][1],weight=sortEdges[0][3],activatedInterface=sortEdges[0][2])
        print('Node to enter list:',sortEdges[0][1])
        l.append(sortEdges[0][1])
        # add the extra edges, based on the updated visited nodes' list
        sortEdges = sorted(G.edges([*l],data='weight',keys=True),key=itemgetter(3))
        # remove edges that both of vertices already exist on spanning tree
        for it in sortEdges:
                if not((it[0] in l) & (it[1] in l)):
                        filtSortEdges.append(it)
        sortEdges = filtSortEdges
        filtSortEdges = []
        l = list(set(l))
        print('visited nodes so far', l)
        print('------------------------------\n')
end = time.time() # finish the timing of the algorithm
# print the spanning tree
print('\nSpanning tree:')
for a,b,c in ST.edges(data=True):
        print('Edge',a,b,'has',c)
print("\nTIME:",end-start)
# initializing already activated attribute for ST nodes
for n in ST.nodes():
        ST.node[n]['alrAct1'] = 0
        ST.node[n]['alrAct2'] = 0
# spanning tree cost
sp = 0
for ed in ST.edges(data=True):
        if (ed[2]['activatedInterface'] == 1) :
                if (ST.node[ed[0]]['alrAct1']!=1) & (ST.node[ed[1]]['alrAct1']!=1) :
                        sp += ed[2]['weight']
                        ST.node[ed[0]]['alrAct1']=1
                        ST.node[ed[1]]['alrAct1']=1
                elif (ST.node[ed[0]]['alrAct1']!=1):
                        sp += G.node[ed[0]]['ActCost1']
                        ST.node[ed[0]]['alrAct1']=1
                elif (ST.node[ed[1]]['alrAct1']!=1):
                        sp += G.node[ed[1]]['ActCost1']
                        ST.node[ed[1]]['alrAct1']=1
        else :
                if (ST.node[ed[0]]['alrAct2']!=1) & (ST.node[ed[1]]['alrAct2']!=1) :
```

```
                                sp += ed[2]['weight']
                                ST.node[ed[0]]['alrAct2']=1
                                ST.node[ed[1]]['alrAct2']=1
                        elif (ST.node[ed[0]]['alrAct2']!=1):
                                sp += G.node[ed[0]]['ActCost2']
                                ST.node[ed[0]]['alrAct2']=1
                        elif (ST.node[ed[1]]['alrAct2']!=1):
                                sp += G.node[ed[1]]['ActCost2']
                                ST.node[ed[1]]['alrAct2']=1
#  cost for activating all available interfaces
costAll = 0
for ed in G.nodes(data='ActCost1'):
        if (ed[1]):
                costAll += ed[1]
for ed in G.nodes(data='ActCost2'):
        if (ed[1]):
                costAll += ed[1]
print('\nSpanning Tree cost',sp,'and Cost for activating all available interfaces',costAll,'\n')
```

Figure 3: The program implementing G_{MU}.

Figure 4(b) shows how our code works when executed on the input multigraph of Figure 4(a).

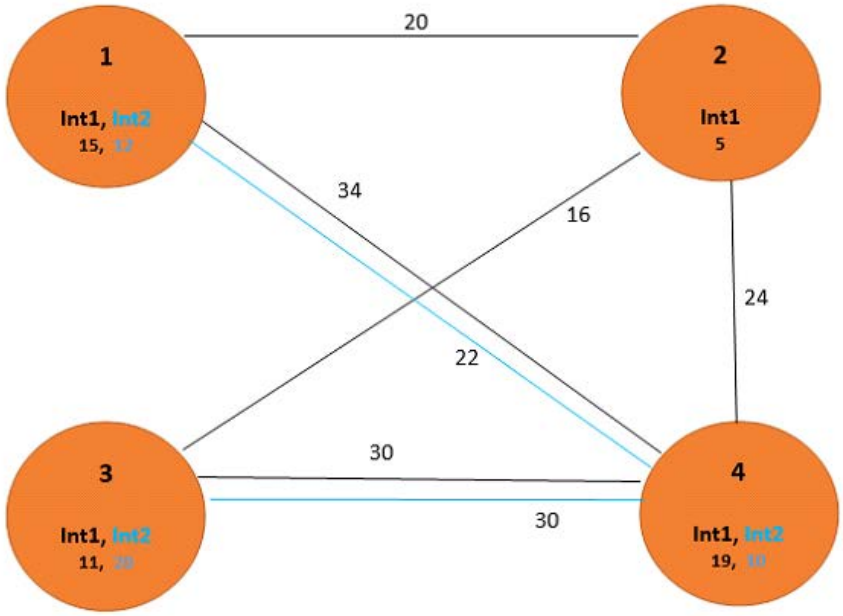


Figure 4(a): input multigraph.

```
Begin algorithm with randomly selected vertex: 3
edges to choose from: [(3, 2, 1, 16), (3, 4, 1, 30), (3, 4, 2, 30)]
edge in process (3, 2, 1, 16)
Node to enter list: 2
visited nodes so far [2, 3]
------------------------------

edges to choose from: [(2, 1, 1, 20), (2, 4, 1, 24), (3, 4, 1, 30), (3, 4, 2, 30)]
edge in process (2, 1, 1, 20)
Node to enter list: 1
visited nodes so far [1, 2, 3]
------------------------------

edges to choose from: [(1, 4, 2, 22), (2, 4, 1, 24), (3, 4, 1, 30), (3, 4, 2, 30), (1, 4, 1, 34)]
edge in process (1, 4, 2, 22)
Node to enter list: 4
visited nodes so far [1, 2, 3, 4]
------------------------------


Spanning tree:
Edge 3 2 has {'weight': 16, 'activatedInterface': 1}
Edge 2 1 has {'weight': 20, 'activatedInterface': 1}
Edge 1 4 has {'weight': 22, 'activatedInterface': 2}

TIME: 0.0004992485046386719

Spanning Tree cost 53 and Cost for activating all available interfaces 92
```

Figure 4(b): Generation of a spanning tree for the multigraph of part (a) by G_{MU}.

Each run is followed by the list of edges of the generated spanning tree, the total time (in seconds) required for the spanning tree generation, to total activation cost and the total cost for activating all available interfaces at all network nodes.

4 Experimental results

For our experimental analysis, a total of 30 automatically generated input instances have been used. Assuming the existence of 2 available interfaces, algorithm G_{MU} has been used to compute spanning trees for these instances. Below, we first give an example of an automatically generated input instance; then, we provide charts for the performance of our approach in terms of activation cost and execution time.

4.1 An example of an automatically generated input instance

Below, we provide screen captures of an indicative experiment. A wireless network of 38 nodes is automatically generated; there are 2 available interfaces. Figure 5 shows network nodes. Edges are depicted in Figure 6.

```
(1, {'Interface1': 'yes', 'ActCost1': 41, 'Interface2': 'yes', 'ActCost2': 23})
(2, {'Interface1': 'yes', 'ActCost1': 59, 'Interface2': 'yes', 'ActCost2': 18})
(3, {'Interface1': 'yes', 'ActCost1': 54, 'Interface2': 'yes', 'ActCost2': 12})
(4, {'Interface1': 'yes', 'ActCost1': 90, 'Interface2': 'yes', 'ActCost2': 99})
(5, {'Interface1': 'yes', 'ActCost1': 21, 'Interface2': 'yes', 'ActCost2': 36})
(6, {'Interface1': 'yes', 'ActCost1': 88, 'Interface2': 'yes', 'ActCost2': 82})
(7, {'Interface1': 'yes', 'ActCost1': 36, 'Interface2': 'yes', 'ActCost2': 82})
(8, {'Interface1': 'yes', 'ActCost1': 21, 'Interface2': 'yes', 'ActCost2': 100})
(9, {'Interface1': 'yes', 'ActCost1': 68, 'Interface2': 'yes', 'ActCost2': 68})
(10, {'Interface1': 'yes', 'ActCost1': 86, 'Interface2': 'yes', 'ActCost2': 58})
(11, {'Interface1': 'yes', 'ActCost1': 36, 'Interface2': 'yes', 'ActCost2': 69})
(12, {'Interface1': 'yes', 'ActCost1': 96, 'Interface2': 'yes', 'ActCost2': 70})
(13, {'Interface1': 'yes', 'ActCost1': 70, 'Interface2': 'yes', 'ActCost2': 78})
(14, {'Interface1': 'yes', 'ActCost1': 82, 'Interface2': 'yes', 'ActCost2': 27})
(15, {'Interface1': 'yes', 'ActCost1': 38, 'Interface2': 'yes', 'ActCost2': 14})
(16, {'Interface1': 'yes', 'ActCost1': 55, 'Interface2': 'yes', 'ActCost2': 40})
(17, {'Interface1': 'yes', 'ActCost1': 28, 'Interface2': 'yes', 'ActCost2': 80})
(18, {'Interface1': 'yes', 'ActCost1': 57, 'Interface2': 'yes', 'ActCost2': 76})
(19, {'Interface1': 'yes', 'ActCost1': 21, 'Interface2': 'yes', 'ActCost2': 21})
(20, {'Interface1': 'yes', 'ActCost1': 24, 'Interface2': 'yes', 'ActCost2': 98})
(21, {'Interface1': 'yes', 'ActCost1': 56, 'Interface2': 'yes', 'ActCost2': 96})
(22, {'Interface1': 'yes', 'ActCost1': 24, 'Interface2': 'yes', 'ActCost2': 31})
(23, {'Interface1': 'yes', 'ActCost1': 73, 'Interface2': 'yes', 'ActCost2': 36})
(24, {'Interface1': 'yes', 'ActCost1': 27, 'Interface2': 'yes', 'ActCost2': 32})
(25, {'Interface1': 'yes', 'ActCost1': 27, 'Interface2': 'yes', 'ActCost2': 27})
(26, {'Interface1': 'yes', 'ActCost1': 13, 'Interface2': 'no'})
(27, {'Interface1': 'yes', 'ActCost1': 25, 'Interface2': 'no'})
(28, {'Interface1': 'yes', 'ActCost1': 66, 'Interface2': 'yes', 'ActCost2': 92})
(29, {'Interface1': 'yes', 'ActCost1': 15, 'Interface2': 'yes', 'ActCost2': 28})
(30, {'Interface1': 'yes', 'ActCost1': 36, 'Interface2': 'yes', 'ActCost2': 58})
(31, {'Interface1': 'yes', 'ActCost1': 67, 'Interface2': 'yes', 'ActCost2': 81})
(32, {'Interface1': 'no', 'Interface2': 'yes', 'ActCost2': 92})
(33, {'Interface1': 'yes', 'ActCost1': 34, 'Interface2': 'yes', 'ActCost2': 72})
```

Figure 5: Nodes of an automatically generated input instance with 2 available interfaces

```
(17, 29, 2, {'weight': 108})
(17, 24, 2, {'weight': 112})
(17, 28, 2, {'weight': 172})
(17, 33, 2, {'weight': 152})
(18, 24, 1, {'weight': 84})
(18, 32, 2, {'weight': 168})
(19, 29, 1, {'weight': 36})
(19, 20, 1, {'weight': 45})
(19, 23, 1, {'weight': 94})
(19, 33, 1, {'weight': 55})
(19, 24, 2, {'weight': 53})
(20, 21, 1, {'weight': 80})
(20, 23, 1, {'weight': 97})
(21, 23, 1, {'weight': 129})
(21, 22, 1, {'weight': 80})
(21, 30, 2, {'weight': 154})
(22, 31, 1, {'weight': 91})
(22, 29, 2, {'weight': 59})
(23, 30, 1, {'weight': 109})
(23, 33, 2, {'weight': 108})
(24, 31, 2, {'weight': 113})
(24, 25, 2, {'weight': 59})
(25, 30, 1, {'weight': 63})
(25, 31, 1, {'weight': 94})
(25, 31, 2, {'weight': 108})
(25, 27, 1, {'weight': 52})
(25, 26, 1, {'weight': 40})
(25, 28, 2, {'weight': 119})
(26, 28, 1, {'weight': 79})
(28, 30, 1, {'weight': 102})
(28, 30, 2, {'weight': 150})
(28, 32, 2, {'weight': 184})
(29, 31, 1, {'weight': 82})
(30, 31, 1, {'weight': 103})
(30, 31, 2, {'weight': 139})
(30, 33, 2, {'weight': 130})
(32, 33, 2, {'weight': 164})
```

```
(1, 31, 1, {'weight': 108})
(1, 31, 2, {'weight': 104})
(1, 11, 1, {'weight': 77})
(1, 23, 1, {'weight': 114})
(1, 22, 1, {'weight': 65})
(1, 22, 2, {'weight': 54})
(1, 16, 1, {'weight': 96})
(1, 12, 1, {'weight': 137})
(1, 12, 2, {'weight': 93})
(1, 13, 1, {'weight': 111})
(1, 5, 1, {'weight': 62})
(1, 24, 1, {'weight': 68})
(1, 21, 2, {'weight': 119})
(1, 10, 2, {'weight': 81})
(1, 8, 2, {'weight': 123})
(1, 25, 2, {'weight': 50})
(1, 18, 2, {'weight': 99})
(1, 33, 2, {'weight': 95})
(1, 2, 2, {'weight': 41})
(1, 9, 2, {'weight': 91})
(2, 27, 1, {'weight': 84})
(2, 11, 1, {'weight': 95})
(2, 11, 2, {'weight': 87})
(2, 13, 1, {'weight': 129})
(2, 4, 1, {'weight': 149})
(2, 12, 1, {'weight': 155})
(2, 8, 1, {'weight': 80})
(2, 30, 1, {'weight': 95})
(2, 3, 1, {'weight': 113})
(2, 6, 2, {'weight': 100})
(2, 32, 2, {'weight': 110})
(2, 28, 2, {'weight': 110})
(2, 19, 2, {'weight': 39})
(2, 9, 2, {'weight': 86})
(3, 29, 1, {'weight': 69})
(3, 13, 1, {'weight': 124})
(3, 7, 1, {'weight': 90})
```

```
(3, 7, 2, {'weight': 94})
(3, 11, 1, {'weight': 90})
(3, 17, 1, {'weight': 82})
(3, 20, 1, {'weight': 78})
(3, 14, 2, {'weight': 39})
(3, 30, 2, {'weight': 70})
(3, 5, 2, {'weight': 48})
(3, 22, 2, {'weight': 43})
(4, 17, 1, {'weight': 118})
(4, 12, 1, {'weight': 186})
(4, 30, 1, {'weight': 126})
(4, 30, 2, {'weight': 157})
(4, 21, 1, {'weight': 146})
(4, 8, 1, {'weight': 111})
(4, 16, 2, {'weight': 139})
(4, 24, 2, {'weight': 131})
(4, 28, 2, {'weight': 191})
(4, 32, 2, {'weight': 191})
(5, 24, 1, {'weight': 48})
(5, 19, 1, {'weight': 42})
(5, 29, 1, {'weight': 36})
(5, 28, 1, {'weight': 87})
(5, 21, 1, {'weight': 77})
(5, 20, 1, {'weight': 45})
(5, 6, 1, {'weight': 109})
(5, 9, 2, {'weight': 104})
(5, 8, 2, {'weight': 136})
(5, 10, 2, {'weight': 94})
(6, 10, 1, {'weight': 174})
(6, 10, 2, {'weight': 140})
(6, 9, 1, {'weight': 156})
(6, 15, 1, {'weight': 126})
(6, 29, 1, {'weight': 103})
(6, 14, 1, {'weight': 170})
(6, 28, 2, {'weight': 174})
(6, 25, 2, {'weight': 109})
(6, 30, 2, {'weight': 140})
```

```
(6, 33, 2, {'weight': 154})
(7, 11, 1, {'weight': 72})
(7, 16, 1, {'weight': 91})
(7, 21, 1, {'weight': 92})
(7, 14, 1, {'weight': 118})
(7, 14, 2, {'weight': 109})
(7, 26, 1, {'weight': 49})
(7, 31, 1, {'weight': 103})
(7, 20, 1, {'weight': 60})
(7, 8, 1, {'weight': 57})
(7, 29, 2, {'weight': 110})
(7, 15, 2, {'weight': 96})
(7, 13, 2, {'weight': 160})
(7, 17, 2, {'weight': 162})
(7, 32, 2, {'weight': 174})
(8, 20, 1, {'weight': 45})
(8, 13, 1, {'weight': 91})
(8, 15, 1, {'weight': 59})
(8, 15, 2, {'weight': 114})
(8, 11, 1, {'weight': 57})
(8, 24, 1, {'weight': 48})
(8, 29, 1, {'weight': 36})
(8, 28, 2, {'weight': 192})
(9, 31, 1, {'weight': 135})
(9, 17, 1, {'weight': 96})
(9, 18, 1, {'weight': 125})
(9, 22, 1, {'weight': 92})
(9, 22, 2, {'weight': 99})
(9, 13, 2, {'weight': 146})
(9, 15, 2, {'weight': 82})
(10, 19, 1, {'weight': 107})
(10, 12, 1, {'weight': 182})
(10, 22, 1, {'weight': 110})
(10, 26, 1, {'weight': 99})
(10, 21, 1, {'weight': 142})
(10, 32, 2, {'weight': 150})
(10, 13, 2, {'weight': 136})
(11, 25, 1, {'weight': 63})
(12, 16, 1, {'weight': 151})
(12, 33, 1, {'weight': 130})
(12, 23, 1, {'weight': 169})
(12, 28, 2, {'weight': 162})
(12, 19, 2, {'weight': 91})
(12, 15, 2, {'weight': 84})
(12, 25, 2, {'weight': 97})
(13, 17, 1, {'weight': 98})
(13, 21, 1, {'weight': 126})
(13, 26, 1, {'weight': 83})
(13, 18, 2, {'weight': 154})
(13, 30, 2, {'weight': 136})
(13, 20, 2, {'weight': 176})
(13, 33, 2, {'weight': 150})
(13, 24, 2, {'weight': 110})
(13, 15, 2, {'weight': 92})
(13, 29, 2, {'weight': 106})
(14, 22, 1, {'weight': 106})
(14, 18, 1, {'weight': 139})
(14, 18, 2, {'weight': 103})
(14, 15, 2, {'weight': 41})
(14, 17, 2, {'weight': 107})
(14, 24, 2, {'weight': 59})
(15, 23, 1, {'weight': 111})
(15, 20, 1, {'weight': 62})
(15, 16, 1, {'weight': 93})
(15, 33, 1, {'weight': 72})
(15, 22, 1, {'weight': 62})
(15, 22, 2, {'weight': 45})
(15, 17, 2, {'weight': 94})
(15, 31, 2, {'weight': 95})
(16, 30, 1, {'weight': 91})
(17, 22, 1, {'weight': 52})
(17, 20, 2, {'weight': 178})
(17, 31, 2, {'weight': 161})
(17, 32, 2, {'weight': 172})
```

Figure 6: Edges of an automatically generated input instance with 2 available interfaces

Algorithm GMU executed for the input instance presented above computed the spanning tree depicted in Figure 7.

```
Spanning tree:
Edge 20 8 has {'weight': 45, 'activatedInterface': 1}
Edge 8 29 has {'weight': 36, 'activatedInterface': 1}
Edge 8 11 has {'weight': 57, 'activatedInterface': 1}
Edge 8 4 has {'weight': 111, 'activatedInterface': 1}
Edge 29 19 has {'weight': 36, 'activatedInterface': 1}
Edge 29 5 has {'weight': 36, 'activatedInterface': 1}
Edge 29 31 has {'weight': 82, 'activatedInterface': 1}
Edge 19 2 has {'weight': 39, 'activatedInterface': 2}
Edge 19 33 has {'weight': 55, 'activatedInterface': 1}
Edge 19 23 has {'weight': 94, 'activatedInterface': 1}
Edge 5 24 has {'weight': 48, 'activatedInterface': 1}
Edge 5 3 has {'weight': 48, 'activatedInterface': 2}
Edge 5 21 has {'weight': 77, 'activatedInterface': 1}
Edge 2 1 has {'weight': 41, 'activatedInterface': 2}
Edge 2 6 has {'weight': 100, 'activatedInterface': 2}
Edge 2 32 has {'weight': 110, 'activatedInterface': 2}
Edge 1 25 has {'weight': 50, 'activatedInterface': 2}
Edge 1 10 has {'weight': 81, 'activatedInterface': 2}
Edge 24 18 has {'weight': 84, 'activatedInterface': 1}
Edge 3 14 has {'weight': 39, 'activatedInterface': 2}
Edge 3 22 has {'weight': 43, 'activatedInterface': 2}
Edge 14 15 has {'weight': 41, 'activatedInterface': 2}
Edge 15 9 has {'weight': 82, 'activatedInterface': 2}
Edge 15 12 has {'weight': 84, 'activatedInterface': 2}
Edge 22 17 has {'weight': 52, 'activatedInterface': 1}
Edge 25 26 has {'weight': 40, 'activatedInterface': 1}
Edge 25 27 has {'weight': 52, 'activatedInterface': 1}
Edge 25 30 has {'weight': 63, 'activatedInterface': 1}
Edge 26 7 has {'weight': 49, 'activatedInterface': 1}
Edge 26 28 has {'weight': 79, 'activatedInterface': 1}
Edge 26 13 has {'weight': 83, 'activatedInterface': 1}
Edge 7 16 has {'weight': 91, 'activatedInterface': 1}

TIME: 0.01951456069946289

Spanning Tree cost 1501 and Cost for activating all available interfaces 3330
```

Figure 7: The spanning tree computed by algorithm G_{MU} for the input instance of Figure 5.

4.2 Activation cost

Figure 8 shows how the network size affects the performance of our greedy approach in terms of total activation cost. Assuming that the input multigraph is connected, we compare the total activation cost induced by G_{MU} to the cost of activating all available interfaces at all network nodes and, also, to O(|V|).

In terms of activation cost, G_{MU} obtains a performance linear in the size of the network; this implies that an extremely simple greedy solution can offer a satisfactory performance as long as the network size remains limited. Indeed, small-scale wireless networks composed of devices supporting a small number of interfaces do appear often in school classes, labs, meeting rooms, medical councils, etc. In such cases, a simple greedy approach like G_{MU}, although theoretically deemed to perform much worse than significantly more complex approaches from the recent literature, can suggest a useful practical solution.

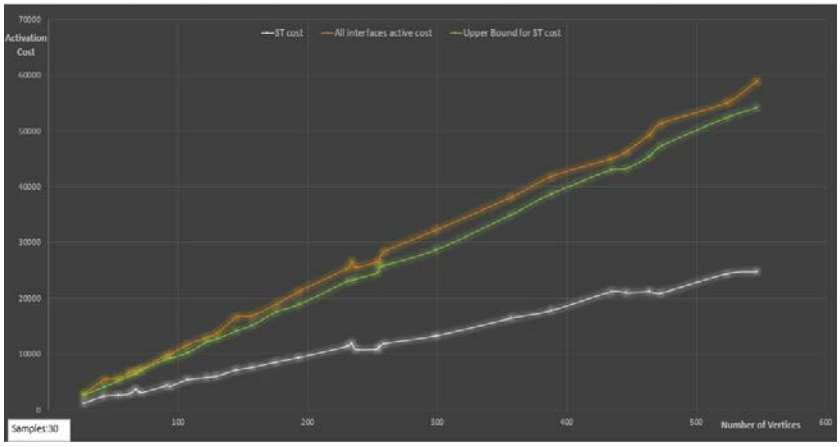


Figure 8: Activation cost of G_{MU} (white line) vs an upper bound for the ST activation cost (green line) and the cost for activating all available interfaces (orange line).

4.3 Execution time

Figure 9 shows how the network size affects the performance of our greedy approach in terms of execution time. Measurements were taken using the function "time" offered by Python. We measured the time interval needed to compute a spanning tree for an input graph G by algorithm G_{MU}. As it can be observed, for networks composed of at most 100 nodes, the running time of G_{MU} is less that 1 sec; for larger networks of approximately 500 nodes, the running time of G_{MU} remains low (at most 1 min) in practice.

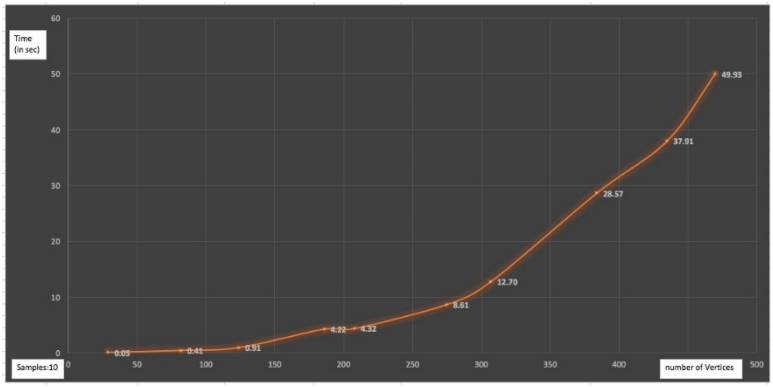


Figure 9: Running time of G_{MU}.

5 Conclusion

We addressed the problem of connectivity in small wireless networks composed of devices supporting multiple interfaces. From a practical point of view, network infrastructure and data collection perspectives can highly benefit from the efficient management of available interfaces in multi-interface wireless networks. We modelled this practical problem as an instance of the Spanning Tree problem in an appropriately defined multigraph corresponding to the actual multi-interface wireless network. We suggested a simple greedy algorithm that indicates which interfaces must be activated so that cost-efficient connectivity is established between any two wireless devices in the network. Our approach shows that simple solutions of theoretically poor performance can still be interesting in practice.

Our future plans include the implementation of the randomized polynomial-time approximation scheme of Prömel and Steger for solving almost exactly the MST problem in hypergraphs [10]; such an implementation would be extremely interesting as a stand-alone component but also as a building block of the (3/2+ε)-approximation algorithm presented in [1] for connectivity in multi-interface wireless networks.

REFERENCES

[1] Athanassopoulos, S., Caragiannis, I., Kaklamanis, C., Papaioannou, E., Energy-efficient communication in multi-interface wireless networks. Theory of Computing Systems, 2013. 52 (2), p. 285-296.

[2] Bahl, P., Adya, A., Padhye, J., Walman, A., Reconsidering wireless systems with multiple radios. ACM SIGCOMM Computer Communication Review, 2004. 34(5), p. 39-46.

[3] Cavalcanti, D., Gossain, H., Agrawal, D., Connectivity in multi-radio, multi-channel heterogeneous ad hoc networks. In Proceedings of the IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 05), 2005. IEEE Press, New York, p. 1322-1326.

[4] Draves, R., Padhye, J., Zill, B., Routing in multi-radio, multi-hop wireless mesh networks. In Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (Mobi- Com 04), 2004. ACM, New York, p. 114-128.

[5] Faragó, A., Basagni, S., The effect of multi-radio nodes on network connectivity: a graph theoretic analysis. In Proceedings of the IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 08), 2008. IEEE Press, New York.

[6] Klasing, R., Kosowski, A., Navarra, A., Cost minimisation in wireless networks with bounded and unbounded number of interfaces. Networks, 2009. 53(3), p. 266-275.

[7] Kosowski, A., Navarra, A., Pinotti, M.C., Exploiting multi-interface networks: connectivity and cheapest paths. Wireless Networks, 2010. 16(4), p. 1063-1073.

[8] Kruskal, J., On the shortest spanning subtree of a graph and the traveling salesman problem. In Proceedings of the American Mathematical Society, 1956. 7, p. 48-50.

[9] Navarra, A., D'Angelo, G., Di Stefano, G., Multi-interface wireless networks: complexity and algorithms. Chapter in Wireless Sensor Networks: From Theory to Applications, CRC Press, Taylor & Francis Group, 2014. p. 119-156.

[10] Prömel, H.J., Steger, A., A new approximation algorithm for the Steiner tree problem with performance ratio 5/3. Journal of Algorithms, 2000. 36, p. 89-101.

[11] Prim, R. C., Shortest connection networks and some generalizations. Bell System Technical Journal, 1957. 36(6), p. 1389-1401.

[12] NetworkX, https://networkx.github.io/documentation/stable/index.html

[13] Python, https://www.python.org/

