Radio Access Network Virtualization for the Social Internet of Things

Radio Access Network Virtualization for Social Internet of Things with Cloud Computing

He Li, *Member, IEEE,* Mianxiong Dong, *Member, IEEE,* Kaoru Ota, *Member, IEEE,*

***Abstract*—In Social Internet of Things (SIoT), devices are often divide to groups with different relationship in social networks. The radio access network (RAN) which is often used for connecting IoT devices to the IoT cloud service is short of supporting the SIoT groups. Thus, to distinguish SIoT groups in the RAN, network virtualization with SDN structure is a preferable with its scalability and usability. However, it is hard to support enough groups due to the rule space limitation of existing SDN enabled devices. In this paper, we first introduce the SDN based RAN virtualization framework then state the problem of maximizing the number of SIoT groups with limited SDN rule space. We also propose an efficient algorithm to solve this problem. The extensive simulation shows that our solution provides more SIoT groups than the original SDN-based RAN virtualization.**

***Index Terms*—Software Defined Networks, Radio Access Networks, Internet of Things, Social Networks, Network Virtualization**

I. Introduction

The Internet of Things (IoT) is a novel network concept that is rapidly gaining ground in the scenario of modern wireless telecommunications [1]. Through unique addressing schemes and standard communication protocols, are able to interact with each others and cooperate with acquaintances in their social relations. Thus, with convergence of IoT and social networks, the Social IoT (SIoT) paradigm will improve the quality of everyday life with interconnected intelligent objects. In SIoT, the network is usually divided to distinct groups to control the degree of interaction among things that are friends. Further, the scalability of network groups is guaranteed similarly with the human social networks [2]. Cloud computing is a potential solution to support scalable groups in SIoT, in which the computing capability can be divided and distinguished with the varying social networks [3].

He Li, Mianxiong Dong and Kaoru Ota are with Muroran Insitute of Technology, Japan.

However, the network resources still need more flexible and scalable methodologies to support group networks in SIoT. Usually, in a general IoT structure, the devices are first connected to the sink nodes or other aggregators then connected to the radio access network (RAN). As the network flows from massive devices are transferred in the same RAN, existing RAN structure is short of dividing network resources for distinguished SIoT groups. Network virtualization is a potential solution that provides independent vRAN for each SIoT group who is connecting the same network [4]. Some forward-looking work proposes some solutions that implement network virtualization in RANs. However, in existing RAN virtualization, base station virtualization and core network virtualization are separated to two parts. It is hard to manage two virtualized parts cooperatively since the different features between two devices.

SDN-enabled mobile or wireless networks become a possible solution to combine the management of base stations and access networks with the decouple control plane and data plane [5]. In SDN-enabled network, all devices are managed by a centralized controller and network operators can put strategies of network virtualization to the controller instead of different devices. Meanwhile, with the flexible control semantics, SDN can provide a fine grained network control for each user in the same network. In this paper, we propose a framework to provide virtual RAN (vRAN) over a single physical SDN-enabled RAN infrastructure.

However, since limitation of existing SDN technology, it is hard to build a RAN virtualization straightforwardly. Since SDN adopt forwarding rules as the basic control unit to manage network traffic, the number of rules is increased dramatically after network virtualization [6]. If the rule space is not enough, for processing new packet, network devices have to communicate to the controller, which dues to much longer latency than direct processing

[7]. Even though the issue brings little influence to the overall performance, the additional latency will harm the quality of service (QoS) especially for some latency sensitive applications (e.g., Voice over IP, cloud gaming, etc.).

As a result, in the RAN virtualization framework, we state the problem that how to assign limited rule space to maximize the number of SIoT group vRANs as well to satisfy the latency requirement. We also propose an efficient allocation algorithm to solve the problem. For verification of our solution, we evaluate our algorithm by simulation and the result shows our allocation performs better than the other SDN enabled virtualization.

The contribution of this paper can be summarized as follows.

- First, we We first introduce an SDN based RAN structure to provide network connection between the SIoT groups and cloud services. Based on that structure, we propose a RAN virtualization framework to provide vRAN to isolate each SIoT groups.
- We then state the problem to maximum SIoT groups with limited rule space in the SDN devices as well to guarantee the latency requirement from the SIoT groups. To solve this problem, we design a efficient algorithm to allocate the rule space to the SIoT groups.
- We take the performance evaluation of the allocation algorithm with extensive simulations, and discuss the allocated number of SIoT groups in different settings. We also compare our algorithm with some other allocation methods and the results show our strategy performs better than others.

The rest of this paper is summarized as follows. Section II reviews the related work. Our framework design are introduced in Section III. Section IV presents the problem formulation. Section V gives the simulation results. Finally, Section VI concludes this paper and give the future work.

II. Related Work

In this section, we introduce some previous and background work about the RAN virtualization for SIoT with cloud computing. First, we discuss the previous works about SIoT. Then, we brief the network virtualization in RANs, including the base station virtualization and core network virtualization.

A. Social Internet of Things

The idea that connecting IoT and social networks together has begun to appear to the literature. Guinard et al. [8] proposed the convergence of IoT and social networks. In their work, an individual can share the services offered by her/his smart objects with either her/his friends or their things. In that work, the reference social network is a social network of humans and it is utilized by things as an infrastructure for service advertisement, discovery, and access. That contribution violates the IoT vision in which the objects should interact spontaneously to offer value-added services to humans.

Kranz et al. [9] investigate the implications of integration between the IoT and the social networks and describe a few exemplary applications. Jian et al. [10] analyze the social attributes which reflect the social relations of nodes. There a sort of quantification of the social relationships among mobile nodes is also performed by means of parameters such as an interaction factor and a distance factor. Besides, the authors study the behavior of mobile nodes by applying the typical theory of the social networks.

Atzori et al. [3] identify appropriate policies for the establishment and the management of social relationships between objects in such a way the resulting social network is navigable. Further, the describe a possible architecture for the IoT that includes the functionalities required to integrate things into a social network.

B. RAN Virtualization

Since IoT usually use RAN to connect things to cloud services, RAN virtualization is potential method to dynamically provide isolated network for each SIoT groups.

Base station virtualization solutions exist commercially today for traditional mobile network operators (MNOs) to cut operating costs. For example, the virtual base transceiver station (vBTS) [11] is one such virtualized base station solution that enables sharing radio components at the hardware level and running multiple base station protocol stacks in software.

Meanwhile, some work [12] proposes to virtualize the Long Term Evolution (LTE) network by implementing a hypervisor in the eNodeB. Each entity runs its LTE stack in a virtual machine. The

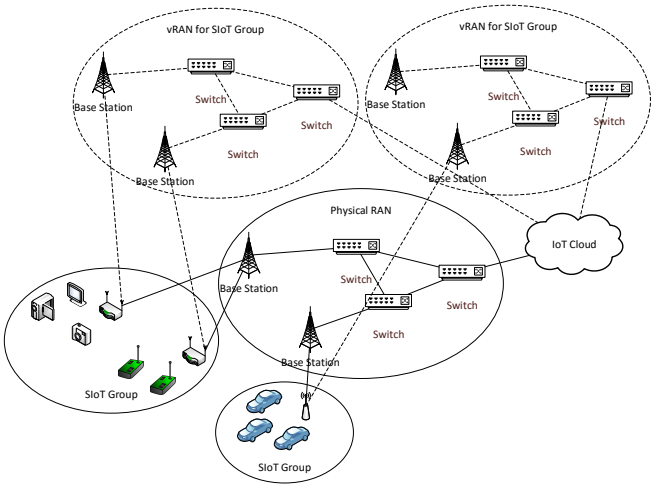

Fig. 1. Typical mode of vRANs for SIoT. Two vRANs are built for different SIoT groups

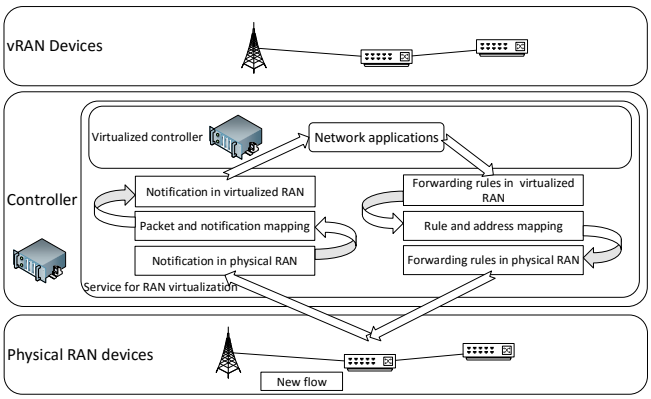

Fig. 2. Main procedures for processing network flows from vRAN to physical RAN

hypervisor allocates spectrum to the different entities in accordance with a specified guarantee. An entity can request either a fixed or dynamic guarantee based on its current load up to a maximum amount of resources. European FP7 project Flexible Architecture for Virtualizable Future Wireless Internet Access (FLAVIA) [13] is defining and prototyping a new base station architecture with the objective of enabling a higher level of programmability. This enhanced base station programmability functionality is being explored in the context of wireless access virtualization, and project participants contribute to the 3GPP RSE Study Item efforts.

In parallel, for the core network virtualization, the emerging software-defined networking (SDN) paradigm is a key enabler to simplify the network provisioning, management, reconfiguration, and control of such virtualized and shared SIoT group networks. Wireless SDN requires the identification of abstractions for wireless primitives and functions, which compromise between flexibility and ability of the abstractions to permit developing innovative wireless functions and mechanisms. An overview of ongoing SDN standardization efforts can be found in [14].

III. Framework Design

In this section, we first introduce a potential mode of radio network virtualization that provide virtual access networks for different SIoT groups. Then, we show main procedures in the RAN virtualization framework with the SDN enabled RAN.

With network virtualization technology, the network infrastructure provider can build multiple vRANs based on a single RAN. With that mode, SIoT groups only need to require enough resources in their vRAN from the infrastructure provider, as well as the IoT cloud service, then provide data communication services to the IoT devices. As shown in Fig. 1, two SIoT groups connect to the IoT cloud through the same physical RAN. To isolate the connection of two groups, two vRANs are built on the physical RAN. In each vRAN, the operators have their vRAN devices and topology to connect their IoT devices to the cloud.

As discussed in Section II, we use SDN to support RAN virtualization. Further, for each vRAN, we also provide the SDN enabled interface for operators executing the network applications. Thus, we add the virtualization framework in the controller of the SDN enabled RAN. The structure of the RAN virtualization framework is shown in Fig. 2. In that structure, we add a service application in the controller to provide virtualized controller service for each vRAN. Virtualized RAN operators can deploy applications for their specified forwarding strategies in their virtual controllers.

Even though we can add a service in the controller to support controller virtualization, physical RAN devices cannot support virtualization. Devices have the capacity to distinguish which vRAN the coming packets belong to. Therefore, in the service for RAN virtualization, we add two main processes to transfer the physical flow control to the virtual flow control.

The first processing transfers the notifications and

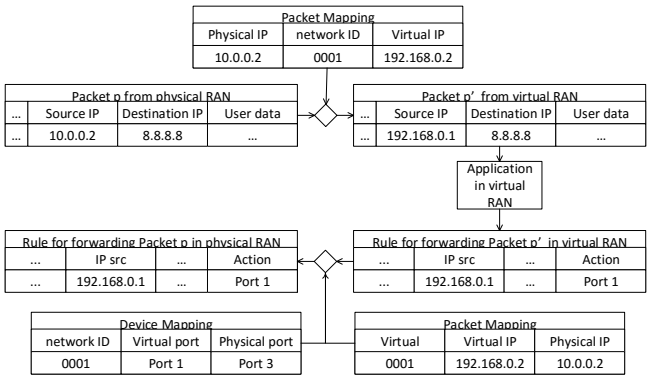

Fig. 3. Example of transferring packets and rules in RAN virtualization framework

packets from physical devices to virtual controllers. In the SDN methodologies, when the existing forwarding rules cannot support the coming packet during switch processes a new network flow, the switch will send a notification contained the new packet to the controller. The controller will generate a set of rules for processing the packet and place the rules to the switch memory. In the virtual RAN structure, before the virtual controller receives the notification, we use a notification and packet mapping to transfer the notification to virtualized one from the virtual device. With the correctly transferred notification, the network application of the vRAN generates the rules for processing the packet.

The second processing transfers the rules generated by the virtualized controller to the forwarding rules for processing the packets in the physical network. Since the virtualized controller has no information about the physical RAN, the generated rules cannot be used for processing packets in the physical RAN. Therefore, we use rule and address mapping to transfer the rules to the ones worked in the physical RAN. After transferring, the controller for the physical RAN places the rules to the switch waiting for processing.

For better understanding the network virtualization framework, we use a simple example to describe some details of the transferring. We choose an IP based RAN with support of existing Openflow protocol. After a new packet come to the physical switch, the switch sends the packet to the controller for the physical RAN. When the controller receives the packet, vRAN service first finds which IP address and vRAN ID of the source IP address and destination IP address mapped. As shown in Fig. 3, the service finds the source IP address 10.0.0.2 belongs to 0001 vRAN and the mapped virtual IP address is 192.168.0.2. Therefore, the service replaces the source IP address and sends the packet with the transferred notification to the virtual controller of 0001 vRAN. After that, the application in the controller generates the related forwarding rule for the packet. The service receives the rule then transfers the rule for the forwarding in the physical network. The action of the rule for vRAN is forwarding packet to port 1. Since the port number of the virtual device and the physical one is different, the service finds port 3 in the physical RAN is mapped to port 1 in the device mapping. Meanwhile, the service also finds the related IP address in the rule through the packet mapping. After replacing the match fields, the controller will transfer the packet and place the rule to the switch for further packets if the rule space allocated to the vRAN is enough.

Meanwhile, we also implement a prototype of this framework design in the mininet environment with OpenvSwitch (OVS) [15]. Since the Open Network Foundation only provides a prototype of RAN with openflow, we use the OVS device to simulate the base station device in the prototype. We use the OVS as main SDN devices in the network and several POX applications as the virtual controller. Between the POX applications and the OVS, we add a procedure to receive all messages from both POXs and OVS. In that procedure, an address table is used for mapping the virtual IP addresses to the physical IP addresses. With that address table, the IP address is changed in that procedure to support network virtualization. Further, for the base station devices, we add a wireless link object with mobility to simulate the wireless connection.

This prototype is far from realistic deployment. The RAN protocols in the real-world are too complex to be implemented in the SDN environment. Most of the network protocols need to be redeveloped into the SDN applications. However, both academies and companies continue developing the SDN/OpenFlow enable RAN products and try to implement the RAN protocols by SDN applications. Therefore, after the network applications for RAN protocols are developed, it is possible to implement our design in the future SDN enabled RAN.

IV. Problem Statement and Algorithm

In our network virtualization, multiple vRANs are built for different SIoT groups. Each groups wants its network has enough performance. However, as existing SDN devices are designed for the single network, the rule space is not enough for sharing between too many vRANs. When the rule space is short for placing rules, some packets need to be forwarded by the controller. Usually the latency between devices and the controller is much longer the latency in the device hardware. This latency will influent the user IoT seriously when a number of packets are processed by the controller. Meanwhile, the user IoT requirement of different groups is usually different. The groups which focus on some delay sensitive applications (e.g., IP communication, desktop interaction, game streaming, etc.) will require shorter latency than the normal ones. Therefore, we have to state the problem then find the optimized allocation method to allocate the limited rule space for each vRAN. In this section, we first state the problem that how to allocate limited rule space to vRANs with different requirement. Then, we describe the allocation algorithm to solve the problem.

We define a set V to denote the networks. In V, we use v_i to denote each vRAN. In vRAN v_i, we use F_i to denote the set of existing flows and f_{ij} to denote the each flow. We define a function $f_{ij}(t)$ to indicate in time t whether a packet exists in flow f_{ij}. Meanwhile, we assume each flow has almost one packet per one time slot. Therefore, for each flow, the total traffic to time t can be described as $\sum_{\tau=0}^{t} f_{ij}(\tau)$. In the physical RAN, we use S to denote the set of switches and s_k to denote each switch.

Further, the paths of flows are different with the forwarding strategy. In most network topologies, the forwarding path of each flow has only part of switches. Therefore, to describe the relationship between flows and switches, we use a $0-1$ function $X_{ijk}(t)$ to indicate whether a flow f_{ij} is forwarded in a switch S_{ijk} as follows.

$$X_{ijk}(t) = \begin{cases} 1, & f_{ij} \text{ is fowarded in } s_k \\ 0, & f_{ij} \text{ is not fowarded in } s_k \end{cases} \tag{1}$$

In SDN enable networks, for forwarding each flow, the switch needs to know the corresponding rules. The switches can get the rules from the local rule space and the controller. Obviously, the latency of accessing the local rule space is much shorter than communication with the controller. However, since vRAN framework uses TCAM to store forwarding rules, the rule space is not enough for store all rules in switches. We use a value r_i to denote the number of rule entries assigned to vRAN v_i. We simplify the rule size for each flow is one entry and use another $0-1$ function $Y_{ijk}(t)$ to indicate whether the rule for flow f_{ij} is placed in a switch s_k in time t as follows.

$$Y_{ijk}(t) = \begin{cases} 1, & \text{rule for } f_{ij} \text{ is placed in } s_k \\ 0, & \text{rule for } f_{ij} \text{ is not placed in } s_k \end{cases} \tag{2}$$

Since the number of rule entries in each switch is limited, we use a value C to denote the rule space for store rules. Therefore, the number of rules is no more the rule space as follows.

$$\sum_{i=1}^{V} \sum_{j=1}^{|F_i|} X_{ijk}(t) Y_{ijk}(t) \leq C \tag{3}$$

If a switch can not find a corresponding rule for the incoming packet, it needs a latency L to processing the packet in the controller. The processing latency in time t in vRAN v_i can be formulated as follows.

$$l_i(t) = \sum_{j=1}^{|F_i|} L \sum_{k=1}^{|S|} X_{ijk}(t) f_{ij}(t)(1 - Y_{ijk}(t))) \tag{4}$$

In the vRAN framework, the rule space in each switch is shared by vRANs of SIoT groups. For management of rule space sharing, in the vRAN framework, we assignment the rule space to each vRAN according the latency requirement. Thus, as we assign the rule space to each vRAN v_i, for switch s_k, the rule space used is also satisfied as follows.

$$\sum_{j=1}^{|F_i|} X_{ijk}(t) Y_{ijk}(t) \leq r_i \tag{5}$$

To satisfy the requirement of latency, we set a latency Q_i for each vRAN v_i and the average latency per each packet brought by the limited rule space should be small than the required latency as follows.

$$\frac{\sum_{\tau=1}^{t} l_i(\tau)}{\sum_{\tau=1}^{t} \sum_{j=1}^{|F_i|} f_{ij}(\tau)} \leq Q_i \tag{6}$$

And we use α_{ik} to denote the average ratio that rule of a flow in vRAN v_i is not placed in the switch s_k as follows.

$$\alpha_{ik} = \sum_{\tau=1}^{t} \frac{\sum_{j=1}^{|F_i|} f_{ij}(\tau) X_{ijk}(\tau)(1 - Y_{ijk}(\tau))}{\sum_{j=1}^{|F_i|} f_{ij}(\tau)} \quad (7)$$

Usually, since the physical RAN providers want more SIoT groups in its network for better scalability, the problem is how to maximize the number of vRAN with the guarantee of latency. Therefore, we formulate the rule space assignment problem as follows.

$$\begin{aligned} &\text{max: } |V| \\ &\text{wt: } L \sum_{k=1}^{|S|} \alpha_{ik} \leq Q_i \\ &\sum_{i=1}^{|V|} r_i \leq C \\ &\sum_{j=1}^{|F_i|} X_{ijk}(\tau) Y_{ijk}(\tau) \leq r_i, \forall 1 \leq \tau \leq t \end{aligned} \quad (8)$$

The problem of rule space assignment in RAN virtualization (RARV): given a physical RAN and a set of vRAN, the RARV problem attempts to assign rule space to the maximum number of vRAN such that the average latency for forwarding a packet in each vRAN is satisfied the IoT requirement.

To solve the problem, since most vRANs use the same devices, we define a requirement first greedy allocation algorithm shown in Algorithm IV. In line 1, we first assign rule space to the vRAN with a small number of required rule space entries. First, allocation sorts all networks with their required rule space then traverse all networks from the one who requires less rule space to the one who requires more. In the traversing from line 2 to 22, the allocation sorts all switches of the traffic forwarded in each switch in line 3. After sorting, the procedure in the traversal places the rules one by one to the rule space of the switch which has enough space from line 5 to 17. When the average latency is more than the latency requirement, the procedure is ended and the allocation will start the procedure again for the next switch in line 10. When the procedure cannot place any of rules for the current vRAN, the allocation will remove all rules of the network and begin the procedure for the next in line 14. If the procedure finished the placement success, the allocation increase the number of the satisfied vRAN by one in line 20. After all networks are processed, the final number of the satisfied vRAN is the optimized result of the rule space allocation problem.

Algorithm 1 Rule space allocation for vRAN

1: Sort all v_i in V that for $V = \{v_1, v_2, ..., v_{|V|}\}$,$r_1 < r_2 < ... < v_{|V|}$;
2: **for** v_i in V **do**
3: Sort all s_j in S that for $S = \{s_1, s_2, ..., s_{|S|}\}$, $f_{i1} > f_{i2} > ... > f_{i|S|}$;
4: **while** $\bar{t_i'} > t_i$ **do**
5: **for** s_j in S **do**
6: **if then**$f_{ij} > 0$ and the capacity of s_j is no more than R
7: Put a rule for v_i in switch s_j;
8: $r_i' \leftarrow r_i' + 1$;
9: **if** $\bar{t_i'} > t_i$ **then**
10: Break;
11: **end if**
12: **end if**
13: **if** $\bar{t_i'}$ has no changed **then**
14: remove all rules of v_i
15: Break;
16: **end if**
17: **end for**
18: **end while**
19: **if** $\bar{t_i'} \leq t_i$ **then**
20: $N \leftarrow N + 1$
21: **end if**
22: **end for**

V. Performance Evaluation

In the section, we use some simulation based programs to test the performance of our allocation algorithm. We first introduce the simulation setting then analyze the results of the performance evaluation.

We use mininet as the main method to simulate the network. Since we implement a prototype in mininet environment, we use the TreeTopo class to build the network topology. We choose the data from Muroran city where our university locates for the network settings in the simulations. From the data of Docomo which is biggest MNO in Japan, the number of base stations in Muroran city is 92 until around 2015. Thus, in our simulations, we use

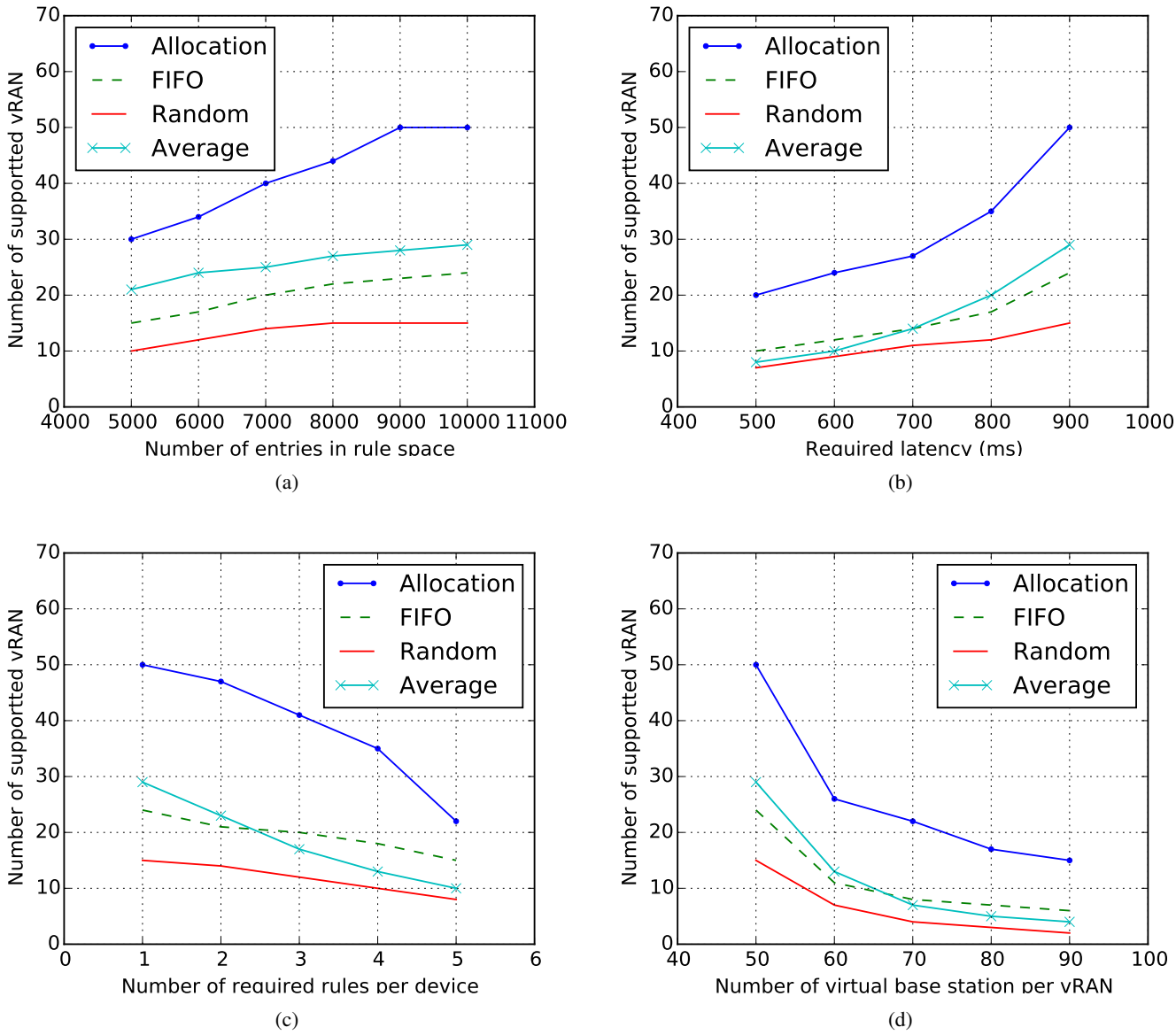

Fig. 4. Evaluation results outperform the comparison algorithm with different settings.

90 as the maximum number of base stations and connect the base station with a randomly generated tree topology.

For each vRAN, for testing the its effect on the result, we set the number of virtual base stations from 50 to 90 and increased 10 per each step. As the population of Muroran city is 89058 on March 31th, 2015 and we assume each citizen has one device connected to the IoT service, the maximum number of devices is set to 90000. Thus, with the number of base stations varying, the number of devices is also set from 50000 to 90000 and increased 10000 per each step. As Muroran city has 51 divisions, we divided the devices in to 50 groups. Therefore, the number of devices per each vRAN is from 1000 to 1800 increased with the number of base stations. From the parameters of existing SDN products, We set the number of rule space entries per each switch is from 5000 to 10000 to test the dependency between the supported number of vRANs and the hardware capacity. As a device usually has no more than 5 functions, we define the average number of flows per each device is from 1 to 5 to test the influence brought by the increasing of the number of network flows. For each flow, the traffic is randomly distributed in the time domain with different number of packages per second (PPS). Since we put the controller in the

Google compute engine, the communication latency between the devices and the controller is about 1000 ms. The latency required by IoT of each vRAN is set from 500 ms to 900 ms and increased by 100 ms per step. The duration time of each round of evaluations is set to one hour. We use the default placement strategy and we also move the flows from one base station to another in the vRAN. We use two existing algorithm as comparisons to our allocation, including random allocation , average allocation, and first-in-first-out (FIFO) which assign the rule space to the vRAN by default sequence used in [6]. We execute the simulation 20 times and record the average value in each result.

We first test the influence by the number of rule space entries in each switches. We fix the average number of required flows per each device is 1, the required latency is 900 ms, the number of devices per each SIoT groups is 1800, and the number of base stations is 90. As shown in Fig. 4(a), we find the number of supported vRANs by our allocation is more than other three methods. The number of supported networks with our allocation algorithm is 30 while the number of random and average allocation is near to 20. With the increasing of rule space, the physical network can support more vRANs. When the number of rule space entries in switches increases to 10000, the number of supported networks with our allocation become 50 while the number of other three methods is no more than 30. With more space entries in each switch, the increment of our allocation is more than other three methods.

For testing the influence by the IoT requirement, we change the required latency from 500 ms to 900 ms. Meanwhile, we fix the number of rule space entries is 10000 in each switch. From the result shown in Fig. 4(b), we find the IoT requirement can affect the result obviously. After we decrease the IoT requirement, the physical network can support much more vRANs than small required latency. Especially, when the latency is increased to 900 ms, the number of supported vRAN is more than two times of the number of 500 ms. Our solution still performs much better than other three methods.

We test the influence brought by the number of users required rules. In the test, the required latency from the IoT requirement is set to 500 ms. As shown in Fig. 4(c), we find that our allocation still performs better than other algorithms. With our allocation, the number of supported vRANs is more than two times than FIFO allocation when the average number of users required rules is 1. When user required more rules for forwarding their flows, the number of supported vRANs decreases quickly. When the average number of users require rules increases to 5, the number of supported vRAN with our allocation algorithm is nearly 20 while the number of other three methods is less than 15. Even worse, the number of supported networks with random allocation is less than 10.

Since more virtual base stations mean better coverage ratio and more users exist in the network, our last test the effect brought by the number of virtual base stations. We set the number of required rules from each user is one. From the result shown in Fig. 4(d), we find RAN virtualization is hard to provide large scale vRANs. With the increasing of base stations, the number of supported vRAN is significantly decreased. Even with our solution, the physical network can support no more than 20 vRANs when each vRAN has 90 base stations. Other three solutions perform similarly and the number of supported vRANs of the Random algorithm is less than 5.

VI. Conclusion

From our design of the rule processing in the controller, we consider SDN based structure is a potential solution for future RAN virtualization especially for the IP based RAN. To solve the problem that maximizing the number of support vRAN with limited SDN rule space, we propose an algorithm for rule space allocation. We also develop a prototype in the simulator and evaluate the performance of our allocation algorithm with simulations. The simulation results show that our solution performs better than the default FIFO based rule allocation used in the previous work. As a result, we will focus on this problem and give a better solution including both flow forwarding and radio bands optimized RAN virtualization.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[2] ——, "From "smart objects" to "social objects": The next evolutionary step of the internet of things," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 97–105, January 2014.

[3] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (siot) when social networks meet the internet of things: Concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594 – 3608, 2012.

[4] X. Costa-Perez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan, "Radio access network virtualization for future mobile carrier networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 27–35, July 2013.

[5] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "Softran: Software defined radio access network," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 25–30.

[6] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Computing*, vol. 17, no. 2, pp. 20–27, March 2013.

[7] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 7–12.

[8] D. Guinard, M. Fischer, and V. Trifa, "Sharing using social networks in a composable web of things," in *The 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops 2010)*, March 2010, pp. 702–707.

[9] M. Kranz, L. Roalter, and F. Michahelles, "Things that twitter: social networks and the internet of things," in *What can the Internet of Things do for the Citizen (CIoT) Workshop at The Eighth International Conference on Pervasive Computing (Pervasive 2010)*, 2010, pp. 1–10.

[10] A. Jian, G. Xiaolin, Z. Wendong, and J. JinHua, "Nodes social relations cognition for mobility-aware in the internet of things," in *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, Oct 2011, pp. 687–691.

[11] I. Vanu, "Vanu - home," http://www.vanu.com/, accessed April 1, 2015.

[12] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "Lte wireless virtualization and spectrum management," in *Proceedings of The Third Joint IFIPWireless and Mobile Networking Conference (WMNC 2010)*, Oct 2010, pp. 1–6.

[13] G. Bianchi, "Line spacing in latex documents," http://www.ict-flavia.eu/, accessed April 1, 2015.

[14] X. Costa-Pérez, A. Festag, H.-J. Kolbe, J. Quittek, S. Schmid, M. Stiemerling, J. Swetina, and H. van der Veen, "Latest trends in telecommunication standards," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 2, pp. 64–71, Apr. 2013.

[15] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010, pp. 19:1–19:6. [Online]. Available: http://doi.acm.org/10.1145/1868447.1868466