# Control Plane Latency Reduction for Service Chaining in Mobile Edge Computing System

Chi-Hsiang Hung, Yao-Chou Hsieh, and Li-Chun Wang
Institute of Electrical and Computer Engineering , National Chiao Tung University, Hsinchu, Taiwan
Email: hungch@g2.nctu.edu.tw, d408052003@hotmail.com.tw, lichun@g2.nctu.edu.tw

***Abstract*—Software-Defined Networking and Network Function Virtualization technologies are utilized to construct service chains cheaply and elastically. In Mobile Edge Computing platform, cloud servers can be deployed near to base stations. Migrating service functions of service chains to edge can reduce network latency or save unnecessary bandwidth consumption between edge and cloud datacenter. However, the control plane latency of SDN/NFV Mobile Edge Computing Platform is subject to the scalability of flow table in SDN switches due to flow table overflow problem. In this paper, we proposed Hash-based Group Management scheme to reduce the number of maintained flow entries by assigning users into groups and design a hash-based structure to efficiently maintain the group information. In our simulation results, the proposed scheme can reduce the number of maintain flow entries and table overflow ratio to improve the control plane latency.**

***Keywords*—SDN, NFV, Mobile Edge Computing, Service Chaining**

## I. Introduction

Service Chaining [1] is an enabling technology for the flexible management of specific service traffic, enforcing policies along the flow route according to the service requirements. A Service Chain (SC) is defined as steering of traffic through a set of service functions (SFs) in a specific order. Software Defined Network (SDN) [2] and Network Function Virtualization (NFV) [3] are technologies commonly used to realize service chaining. Usually, the SDN will be implemented by a standard called OpenFlow [4] which is managed by Open Networking Foundation (ONF). With the SDN and NFV, operators can easily build a customized SC according to different requests.

Mobile Edge Computing (MEC) [5], [6] is an emerging technology that provides cloud and IT services within the close proximity of mobile subscribers. It allows the availability of the cloud servers inside or adjacent to the base station. As services run close to end devices, it considerably can provide lower latency, faster reaction, higher users Quality of Service (QoS), and less congestion in other parts of the network. Traditionally, mobile network operators deploy various SFs in a main data center, which is called as core site, near to the core network. However, in MEC environment, operators can deploy light weight datacenters, which is called as edge site, near to base stations. In a MEC system, each site has an independent SDN controller and a cloud controller. By deploying SFs in edge sites, operators can reduce the network latency or offload the workload or bandwidth from core site. For migrating some SFs from core site to edge site, the MEC system needs to orchestrate multiple sites resources to provide an end-to-end SC. It will increase network latency due to interactions of control plane between multiple sites. In the point view of SDN network in MEC, the network latency for control plane are caused by the following three parts:

1) **Path decision for SC and flow entry installation**. For a SC request, routing path according to the policies needs to be transferred into flow entries and installed in OpenFlow Switches (OFSs) distributed in edge site and core site.
2) **Information synchronization between multiple sites**. In order to serve an end-to-end SC request, information of SFs between edge site and core site needs to be synchronized.
3) **Table overflow handling for OpenFlow switch**. Modern commodity switches only accommodates a few thousands of flow entries [7] using TCAM. When the number of entries exceeds the size of flow table, the least recently used entry will be removed. As a result, OFSs will re-send Packet-In messages to do routing lookup again by controller for the removed flow entry. It will increase a lot of the network latency.

For each SC request in MEC system, it is necessary to have a fast and efficient flow entry management mechanism to mitigate unnecessary network latency in the control plane. There are some works which are addressing about the issue of reducing the number of flow entries for service chaining. In the EnforSDN [8], if a SC which includes a firewall can be bypassed after the first packet has been checked by this firewall. Therefore, the routing path is decreased one hop to reduce the number of flow entries in OFS. However, it is only suitable for SCs that include firewall function. In the LightChain [9], they optimize the placement of SFs in a SC to reduce the number of flow entries by eliminating ping-pong traffic in a data center. However, this approach is suitable when the original SC path induces ping-pong traffic. Besides, the optimization process will increase network latency in the control plane. In the OpenSCaaS [10], they use a source MAC address to encode a SC request. Users who require the same type of SC will be given the same source MAC. Therefore, users with same source MAC will use the same flow entries and the total number of flow entries can be reduced. However, when too many users are given the same source MAC and served by the same server, the latency and packet drop rate will increase due to the overloading of servers.

In order to reduce the control plane latency for service chaining in MEC system, we focus on reducing number of flow entries used in the system. In this paper, we proposed Hash-based Group Management scheme (HGM) to manage the flow entries for service chaining. We assign users who subscribe the same SC and Service-Level-Agreement (SLA) into the same group, and each group has a group id tagged on packets sent from the group members. The OFSs can then match the group id to decide routing path of the packets to reduce the total number of flow entries. To efficiently maintain information of all groups, we also proposed a hash-based group table to reduce the computation time for assigning user into groups to reduce the control plane latency.

The remainder of this paper is organized as follows: Section II presents the overview of service chaining in MEC system. Section III details the proposed Hash-based Group Management (HGM). Section IV describes simulations and evaluation for the HGM. Finally, we conclude the paper and suggest some future research in Section V.

## II. System Overview

Fig 1 shows the architecture of our MEC system. Each UE will connect to an edge site through LTE or WiFi. In each site, we have a SDN controller, a data center, and our Hash-based Group Management (HGM). In each data center, there are various SFs that can be deployed for different SCs. HGM is a SC manager which can select the corresponding SCs for users according to the users' demand and use SDN controller to set the routing path of SC through REST API. In front of the OFSs in edge site, there is also a classifier that can classify the incoming flows into different group and tag group id on packets. In this architecture, we assume that C is the set of all kinds of SFs that can be deployed in core site. E is the subset of C and is consisting of specific SFs that are able to be deployed in each edge site with limited resources. Such that, SFs of a SC for one user may be deployed in edge site and/or core site. In our MEC system, when an edge site is not able to provide all SFs of a SC, a query for the remaining SFs will be passed to core site to fullfil the complete SC.

When UEs connect to the edge sites, they will subscribe SCs with SLA like acceptable latency and required bandwidth. Once the HGM receives the request of each UE, it will arrange resources for deploying SCs. In order to reduce the number of flow entries, HGM will assign users who subscribe the same SC and latency into the same group. Each group will have a group id tagged on the packets from group members. The id is tagged on either VLAN or IP TOS field of a packet by the classifier. The OFSs match the group id instead of MAC/IP address to decide the path to reduce the number of maintained flow entries. Besides, we also set a resource upper bound like bandwidth or user number for each group to avoid overloading of each SF. Once a group reaches its resource upperbound, new group will be created for the new users.

To efficiently maintain the information of all groups, we use a hash-based group table to reduce the computation time for assigning user into a group, such as Fig 2 shows. In general, information of groups will be maintained in a linear table. However, when we need to search an available group, we need to check each slot, which may increase the computation time when the table size is large. Therefore, with a hash-based structure, we can use a constant time for searching a group regardless of table size.

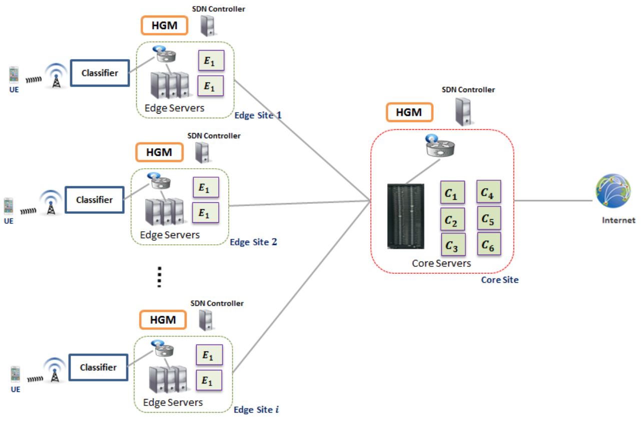
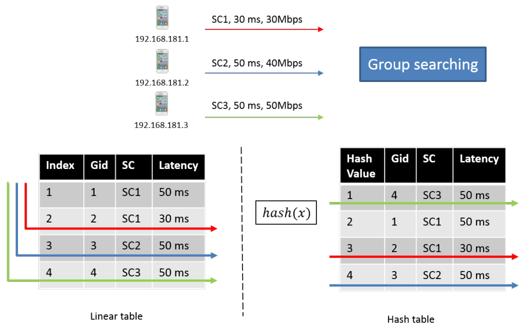


Fig. 1: The architecture of MEC system.



| Index | Gid | SC | Latency |
|---|---|---|---|
| 1 | 1 | SC1 | 50 ms |
| 2 | 2 | SC1 | 30 ms |
| 3 | 3 | SC2 | 50 ms |
| 4 | 4 | SC3 | 50 ms |

| Hash Value | Gid | SC | Latency |
|---|---|---|---|
| 1 | 4 | SC3 | 50 ms |
| 2 | 1 | SC1 | 50 ms |
| 3 | 2 | SC1 | 30 ms |
| 4 | 3 | SC2 | 50 ms |

Fig. 2: Linear search v.s. Hash table

## III. Proposed Hash-based Group Management

In the proposed MEC system, each HGM maintains a Group Table $GT$ which is implemented as a hash table with a linked list in each slot. Each node of the list records the information of a group like group's id $GID$, service chain type $SC$, currently used bandwidth $UBw$, number of users $UC$, and the information of group members $UserSet$. When mobile users are online, they will send a SC request including service chain type $SC_u$, latency $L_u$, and bandwidth $Bw_u$ to HGM. Then HGM will start Group Assignment function to assign user into suitable group depending on users request. Since SFs of a SC may distribute in edge and/or core site. The distribution of SFs for request SCu can be classified in the following two cases: (1) Edge site provides one or more SF of a $SC_u$. In this case, the users group will be assigned by the $HGM_{edge}$, and the group information will also be maintained at edge site. (2) Edge cannot provide any SF of a SC. In this case, the users group will be assigned by the $HGM_{core}$, and the group information will also be maintained at core site.

### *A. Group Assignment in Edge*

Algorithm 1 shows the group assignment procedure in edge site. In this algorithm, if edge site cannot provide any SF of the $SC_u$, the request will be forwarding to the $HGM_{core}$. After receiving the $GID$ from core site, it will setup a bypass path to core without going through any SF in edge site. However, if edge can provide one or more SF of the SC, we can serve the user in edge site. When user $u$ is online, the $HGM_{edge}$ uses $SC_u$ and $L_u$ to generate hash index by using hash function. With the index, the $HGM_{edge}$ can check whether a list of groups in $GT$ with the same SLA for $SC_u$ exists or not. If the list exists, ten check the reminder resources of each group in the list one by one. If the $HGM_{edge}$ can find a suitable group to serve user, it will assign the user to this group and update the group information. However, if the list is empty or the reminder resources of all groups in the list are insufficient, the $HGM_{edge}$ will create a new group for the user and insert it to the head of the list. Once a new group is created, SFs will be deployed on the servers. We will also setup a new SC path in both edge site and core site and ask classifier to tag $GID$ into users packets.

**Algorithm 1** Group Assignment in Edge

```
Require: $SC_u$, $L_u$, $Bw_u$, $IP_u$
Ensure: $GID$
 1: $SC_{edge}$ = $SC_u \cap E$;
 2: $SC_{core}$ = $SC_u - SC_{edge}$;
 3: if $SC_{edge} == \phi$ then
 4:     $GID_{edge}$ = NULL;
 5:     $SendRequestToCore(GID_{edge}, SC_{core}, L_u, Bw_u, IP_u)$;
 6:     $GID_{core}$ = $ReceiveFromCore()$;
 7:     $path$ = NULL;
 8:     $InstallFlowEntry(path, GID_{core})$;
 9:     $InstallTag(IP_u, GID_{core})$;
10:     return $GID_{core}$;
11: end if
12: $i$ = $hash(SC_u, L_u)$;
13: $g$ = List head of $GT[i]$;
14: if $g \neq$ NULL then
15:     if $UBw_g + Bw_u \leq MaxBw_g$ and $UC_g + 1 \leq MaxC_g$ then
16:         $UBw_g$ = $UBw_g + Bw_u$;
17:         $UC_g$ = $UC_g + 1$;
18:         $UserSet_g$ = $UserSet_g \cup IP_u$;
19:         $InstallTag(IP_u, GID_g)$;
20:         return $GID_g$;
21:     end if
22: end if
23: $j$ = $CreateNewGroup(SC_u, L_u, Bw_u, IP_u)$;
24: Insert $j$ to the head of $GT[i]$;
25: $path$ = $DeploySF(SC_{edge}, GID_j)$;
26: $InstallFlowEntry(path, GID_j)$;
27: $SendRequestToCore(GID_j, SC_{core}, L_u, Bw_u, IP_u)$
28: $InstallTag(IP_u, GID_j)$;
29: return $GID_j$;
```

### *B. Group Assignment in Core*

In the core site, the $HGM_{core}$ will execute Group Assignment function when it receives requests from edge site. Algorithm 2 shows the group assignment procedure in core site. In this algorithm, if the request contains a $GID$, the $HGM_{core}$ just needs to deploy the SFs that edge site cannot provide. Then it will setup the SC path in core site with the $GID$. Otherwise, the $HGM_{core}$ needs to assign the user into groups maintained in core site or create a new group for user and send the assigned $GID$ back to edge site.

**Algorithm 2** Group Assignment in Core

```
Require: $GID_{edge}$, $SC_{core}$, $L_u$, $Bw_u$, $IP_u$
Ensure: $GID_{core}$
 1: if $GID_{edge} \neq$ NULL then
 2:     $path$ = $DeploySF(SC_{core}, GID_{edge})$;
 3:     $InstallFlowEntry(path, GID_{edge})$;
 4:     return NULL;
 5: end if
 6: $i$ = $hash(SC_{core}, L_u)$;
 7: $g$ = List head of $GT[i]$;
 8: if $g \neq$ NULL then
 9:     if $UBw_g + Bw_u \leq MaxBw_g$ and $UC_g + 1 \leq MaxC_g$ then
10:         $UBw_g$ = $UBw_g + Bw_u$;
11:         $UC_g$ = $UC_g + 1$;
12:         $UserSet_g$ = $UserSet_g \cup IP_u$;
13:         return $GID_g$;
14:     end if
15: end if
16: $j$ = $CreateNewGroup(SC_{core}, L_u, Bw_u, IP_u)$;
17: Insert $j$ to the head of $GT[i]$;
18: $path$ = $DeploySF(SC_{edge}, GID_j)$;
19: $InstallFlowEntry(path, GID_j)$;
20: return $GID_j$
```

## IV. Simulation and Evaluation

We use Mininet [11] to simulate our MEC topology as Fig 1 shows. In this topology, there are three edge sites and a core site. In each site, there is a Ryu controller with HGM, two OFSs, and two servers. There is also a classifier in each edge site. To simulate user traffic in the MEC system, we generate 1∼5 flows with different SC requests randomly for each user in the three edge sites. Table I shows the parameters of the system setting in our simulation. In order to evaluate our HGM, firstly we evaluate the computation time of HGM. Secondly, we compare the table overflow ratio user-based and group-based scheme. Finally, we evaluate the control plane latency in our environment.

TABLE I: parameters in the simulation

| | |
|---|---|
| number of flow entries/switch | 8,000 |
| number of SFs in edge site | 2 |
| number of SFs in core site | 5 |
| resource of a group | 10 users and 1Gbps bandwidth |
| number of SFs/SC request | 1∼5 |
| SLA of a SC request | 4 latency level<br>10∼50 Mbps bandwidth |

In order to make the simulation environment closer to the real network conditions, we deploy PC which running Ryu controller and Pica8-3297 OFS to measure the latency for flow entry installation, table miss handling. For cross site communication of HGMs, we deploy 3 PCs which include 1 classifier and 2 Ryu controller with HGM. Table II shows the results of the latency in the two testbeds. Then we will apply the results to the subsequence simulation to evaluate the performance of proposed MEC system.

TABLE II: latency of operations

| operation | delay (ms) |
|---|---|
| flow entry installation (table has capacity) | 2.42 |
| flow entry installation (table is full) | 4.06 |
| table miss handling | 7.48 |
| tag entry installation | 1.44 |
| communication between HGMs | 0.221 |

## A. Computation time of HGM

Here we compare two different data structures used to maintain the group information. The first one is general linear-based structure and the second one is the proposed hash-based structure. In the linear-based structure, the system needs to search an available group for a user by checking the records one by one. The proposed hash-based structure needs only one hash operation for checking whether the group information exists or not. Fig 3 shows the comparison of computation time of these two approaches. The linear-based approach is susceptible to the increasing of the number of users. However, the hash-based approach is relatively fast.

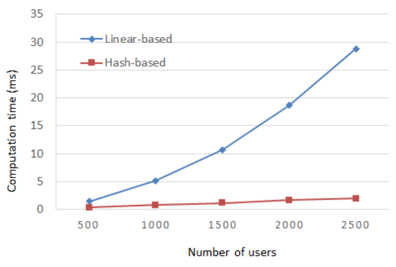


Fig. 3: Computation Time of HGM

## B. Table overflow ratio

Fig 4 shows the table overflow ratio in core site. In the User-based scheme, almost 1,500 users will make the capacity of OFSs reach the upper bound. When new users keep entering the system, the table overflow may occur, which means some active entries will be replaced. However, in HGM scheme, table overflow will not occur until there are more than 15,000 users. When the number of users are more than 25,000, the table overflow ratio is less than 10% by using HGM. It's more scalable than User-based scheme.

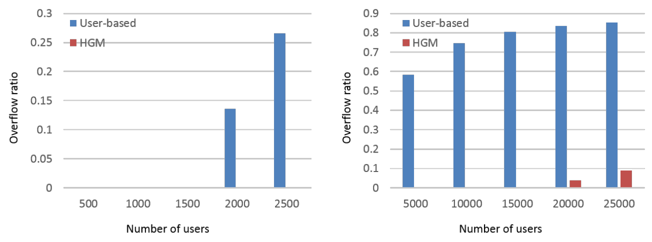


Fig. 4: Table overflow ratio

## C. Control plane latency

Fig 5 shows the average latency per flow for configuring the routing path for each flow. In general User-based scheme, for the first packet of each flow, we need to install the rules related to the request. However, in HGM scheme, we only need to install rules when a new group is created. Therefore, the average latency for configuring a flow by using HGM is close to 50% than User-based scheme when the number of users is lower than 2,500. When the number of users is more than 5,000, the table overflow ratio is growing up rapidly by using User-based scheme. Table III shows the comparison of latency per flow between the two schemes for the number of users is more than 5,000. The results show that, the average latency per flow by using HGM is 20% to 24% of User-based scheme.

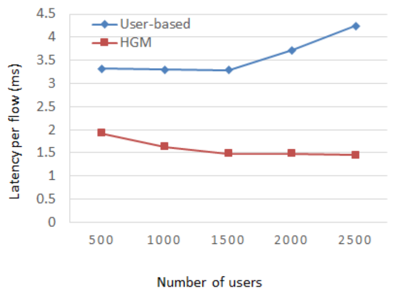


Fig. 5: Control plane latency of each flow

TABLE III: Comparison of latency per flow

| #users | User-based | HGM | Ratio |
|---|---|---|---|
| 5,000 | 5.81 ms | 1.42 ms | 0.24x |
| 10,000 | 6.71 ms | 1.41 ms | 0.21x |
| 15,000 | 7.05 ms | 1.42 ms | 0.20x |
| 20,000 | 7.23 ms | 1.54 ms | 0.21x |
| 25,000 | 7.32 ms | 1.78 ms | 0.24x |

## V. Conclusion

In this paper, we designed HGM and distributed them into each site to manage the service chaining for MEC system. The HGM assigns users who subscribe the same SC and SLA into the several groups with a group id to reduce the maintained flow entries. The time complexity of group assignment is reduced to a constant time by using hash-based group tables. In our simulation results, the HGM can mitigate the table overflow ratio significantly. Besides, the average latency for each flow can be reduced 20% to 24% than using User-based scheme when the number of user is more than 5,000. The proposed HGM is efficient to reduce the network latency of control plane in SDN/NFV-based MEC system.

In this paper, we focus on the evaluation of the proposed HGM, therefore we consider that the resource is infinite in edge sites. In the future, we will set a resource upper bound in each edge site and adjust the number of edge sites of MEC system to analysis the system scalability and the control lane performance in core site.

## References

[1] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 216–223, 2017.

[2] "The Software Define Network architecture overview Version 1.1." [Online]. Available: https://www.opennetworking.org

[3] "An Open Platform to Accelerate NFV." [Online]. Available: https://www.opnfv.org

[4] "The OpenFlow specification." [Online]. Available: https://www.archive.openflow.org/wp/documents.

[5] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Mobile edge computing towards 5G: Vision, recent progress, and open challenges," *China Communications*, vol. 13, pp. 89–99, 2016.

[6] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *10th IEEE International Conference on Intelligent Systems and Control (ISCO)*, 2016, pp. 1–8.

[7] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014.

[8] Y. Ben-Itzhak, K. Barabash, R. Cohen, A. Levin, and E. Raichstein, "EnforSDN: Network policies enforcement with SDN," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 80–88.

[9] A. Hirwe and K. Kataoka, "LightChain: A lightweight optimization of VNF placement for service chaining in NFV," in *IEEE NetSoft Conference and Workshops (NetSoft)*, 2016, pp. 33–37.

[10] W. Ding, W. Qi, J. Wang, and B. Chen, "OpenSCaaS: An Open Service Chain as a Service Platform Toward the Integration of SDN and NFV," *IEEE Network Magazine*, vol. 29, pp. 30–35, 2015.

[11] "Mininet overview." [Online]. Available: https://www.mininet.org/overview.