논문 22-47-01-03 The Journal of Korean Institute of Communications and Information Sciences '22-01 Vol.47 No.01
https://doi.org/10.7840/kics.2022.47.1.13

분산 SDN 환경에서 학습 기반의 마이그레이션 타겟 컨트롤러 선택 기법

설 해[◆], 백 상 헌[°], 김 기 훈[*], 박 현[*]

A Target Selection Scheme for Learning-Based Switch Migration in Distributed Software-Defined Networks

Xue Hai[◆], Sangheon Pack[°], Kihun Kim[*], Hyun Park[*]

요 약

다중 컨트롤러가 있는 분산 소프트웨어 정의 네트워크에서 가변적인 트래픽의 변화는 각 컨트롤러 사이의 부하 불균형을 발생시키기 쉽다. 이를 해결하기 위해 스위치 마이그레이션 (Switch Migration: SM) 기술을 사용할 수 있는데 동적인 네트워크 상황을 고려해서 최적의 마이그레이션 타겟 컨트롤러를 선택하는 것은 여전히 중요한 이슈로 남아있다. 본 논문에서는 기댓값 최대화 (Expectation Maximization) 학습 알고리즘을 기반으로 타겟 컨트롤러를 결정하는 LSM (Learning-based SM) 기법을 제안한다. LSM은 OpenFlow Packet-In 메시지 전달 이력을 학습하여 잠재적인 타겟 컨트롤러의 가능성 값을 최대화하기 할 수 있는 기댓값 최대화 알고리즘을 사용한다. OpenDayLight 기반의 실험 결과는 LSM을 사용하였을 경우 기존 기법 보다 더 높은 처리량과 더 낮은 패킷 손실률을 얻을 수 있음을 보여준다.

Key Words : Controller selection, Expectation-maximization algorithm, Software-defined networking, Switch migration

ABSTRACT

In distributed software-defined networking with multiple controllers, traffic variations can easily cause load imbalance among individual controllers. Thus, switch migration (SM) techniques have been introduced to address this problem. However, appropriate selection of the target controller for SM considering the dynamic nature of networks remains a challenge. In this paper, a learning-based SM (LSM) scheme is proposed to select the most appropriate target controller for SM operation. LSM employs the expectation-maximization algorithm to maximize the likelihood value of the potential target controller by learning the OpenFlow Packet-In message forwarding history. The experimental results demonstrate that LSM substantially outperforms existing schemes in terms of throughput and packet loss rate.

※ This work was supported by a grant-in-aid of HANWHA SYSTEMS

◆ First Author : University of Shanghai for Science and Technology, xuehai@korea.ac.kr, 정회원
° Corresponding Author : Korea University, School of Electrical Engineering, shpack@korea.ac.kr, 종신회원
* Hanwha System, {kihun19.kim@hanwhasystems.com, 정회원; h82.park@hanwhasystems.com

논문번호 : 202110-289-B-RU, Received October 17, 2021; Revised November 5, 2021; Accepted November 7, 2021

Ⅰ. Introduction

Due to the ever-expanding scale of software-defined networks (SDN), the scalability of the centralized controllers becomes a key issue. For high scalability, a logically centralized but physically distributed controller structure is widely used. However, such statically determined mapping results in load imbalance among controllers and resource underutilization[1]. That is, only certain controllers become overloaded, whereas the others remain underutilized and thus become cold spots. This further makes the control plane unable to adapt to traffic variations[2,3]. To address this problem, workloads of overloaded controllers should be reduced and shifted to the cold spots that are expected to achieve higher network performance. To this end, the switch migration (SM) techniques have been proposed and their effectiveness clearly proved in[1] with a concise case study.

Even though, with SM, the reliability and scalability of the distributed controllers can be sufficiently enhanced, SM must be implemented with a well-designed mechanism to decide which controller should be selected for SM. For example, random or unsuitable target controller selection may lead to load imbalance because the capacity of the controller is not accurately considered. A couple of SM schemes have been introduced[4,5,6]. Dixit et al.[4] proposed an elastic distributed controller architecture in which a controller pool is dynamically grown or shrunk according to traffic conditions, and the load is dynamically shifted across controllers. The nearest controller is selected as the target[4] to save the migration time. Cui et al.[5] proposed a load-balancing strategy among multiple SDN controllers in which the response time was used to measure the controller load, and they derived an appropriate response time threshold to detect overloaded controllers. Sahoo et al.[6] applied KarushKuhn-Tucker conditions for target controller selection in the software-defined wide area network architecture and proposed a novel strategy to optimize the response time of control messages during SM. However, some of these works may cause imprecise target controller selection (e.g., [4]) and others may generate complicated selection procedures (e.g., [5, 6]). To overcome these limitations, a new efficient SM scheme must be developed for appropriate target controller selection while considering dynamic network conditions.

In this paper, a learning-based switch migration (LSM) scheme is proposed to decide which controller is preferred for SM. The controller selection problem is analytically formulated, and the decision is made based on the learning with the incoming history of OpenFlow Packet-In messages by means of the expectation maximization (EM) algorithm. The evaluation results over an OpenDayLight-based testbed demonstrate that LSM substantially outperforms existing schemes in terms of throughput and packet loss rate.

Ⅱ. System Model

In this section, we first present the LSM network model. Subsequently, we explain the formulation of the controller load in Openflow-based SDN.

2.1 Network Model

In Openflow-based SDN, a packet is forwarded to a relevant controller from an OpenFlow switch that contains the non-corresponding flow entry to match it. The unmatched packet and its header are encapsulated in a Packet-In message, and the master controller decides the routing path and installs flow entries for it. Although the controller load is affected by several factors such as the process of Packet-In messages, the global network topology, communication patterns among controllers, and the number of flow entries, the processing of Packet-In messages is generally regarded as the most vital part in controller resource utilization[7,8]. That is, Packet-In messages should be evenly distributed to multiple controllers to avoid any load imbalance situations.

The LSM network model, which is depicted in Fig. 1, consists of multiple OpenFlow switches and controllers. The controllers exhibit the same performance and partition the entire network into N

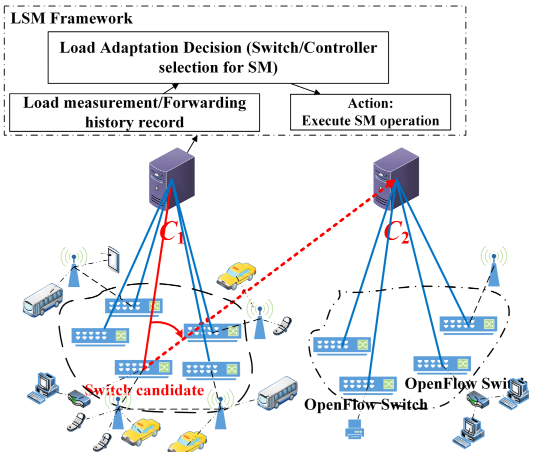
LSM Framework
Load Adaptation Decision (Switch/Controller selection for SM)
Load measurement/Forwarding history record
Action: Execute SM operation
C_1
C_2
Switch candidate
OpenFlow Switch
OpenFlow Switch

Fig. 1. The network model of LSM.

domains. In addition, the traffic volume and controller workload are changed dynamically. Each controller in LSM has three modules: 1) a load measurement/forwarding history record module, 2) a load adaptation decision module, and 3) an action module[6]. The load measurement/forwarding history record module aims to measure the load condition of the controller and store the Packet-In message incoming records of individual controllers. Meanwhile, the load adaptation decision module selects a candidate switch that forwards its Packet-In messages to the controller. To this end, the EM algorithm is employed to determine the target controller with the smallest likelihood value, which is elaborated in Section 3. Finally, the action module carries out the SM operation, which migrates the candidate switch to the newly selected target controller.

2.2 Controller Load Model

In SDN, SM occurs in three cases. First, the switches should be migrated if the relevant controller is shut down or in sleep mode to reduce energy consumption. Second, if the congested traffic load exceeds the capacity of all the controllers (i.e., all of the controllers are fully utilized), an alternative controller should be added, and the overloaded switches should be migrated to it. Third, if the aggregated traffic load is beyond the capacity of the controller, the corresponding switches should be migrated to other underutilized ones. In this case, a new controller does not need to be added. Note that this work only considers the last case, as the same as[2]. Two parameters (i.e., CPU and bandwidth usages) of the controller need to be considered for load measurement. Let p_k and b_k be the unit cost of CPU and bandwidth for processing a Packet-In message. Then, the resource utilization η_{kn} of controller c_n by switch s_k, where $k \in \{1, 2, ..., K\}$, and $n \in \{1, 2, ..., N\}$, is given by

$$\eta_{kn} = (u_n^C \frac{j_k \times T \times p_k}{\mu_n} + u_n^B \frac{j_k \times T \times b_k}{\rho_n}) \quad (1)$$

where j_k is the number of Packet-In messages generated by s_k to c_n per unit time. T is the elapsed time. u_n^C and u_n^B are the weights for CPU and bandwidth costs, respectively, and their sum is 1. Note that μ_n and ρ_n represent the CPU and bandwidth capabilities of c_n. Then, the total load of c_n can be defined as[6]

$$L(c_n) = \sum_{K=1}^{K} [r_{k,n} \times \eta_{kn}] \quad (2)$$

where $r_{k,n}$ is a Boolean function to represent the mapping between switch s_k and controller c_n, which is given by

$$r_{k,n} = \begin{cases} 1 & \text{if } s_k \text{ is managed by } c_n \\ 0 & \text{otherwise} \end{cases}. \quad (3)$$

Subsequently, the average network load of N controllers can be derived as

$$\overline{L} = \frac{1}{N} \sum_{n=1}^{N} L(c_n). \quad (4)$$

To quantize the load measurement, the ratio of the current load to the average load, ξ, is defined as

$$\xi = \frac{L(c_n)}{\overline{L}} \tag{5}$$

where ξ < 1 illustrates the load is underutilized. On the other hand, ξ = 1 and ξ > 1 represent that the load is stable and overloaded, respectively. For the load measurement module, it can be concluded that the first case is regarded as underutilized, and the other two cases are regarded as overloaded, which triggers SM operations[6].

2.3 Procedure of SM

The whole procedure of SM is depicted in Fig. 2. At first, the initial overloaded master controller A transmits a Start Migration request message to the selected new controller B (see (1) in Fig. 2). Once B receives the message, it requests to transit its role to Equal by forwarding a Role Request message to the switch S (see (2)-(3) in Fig. 2). After B receives the Role Response message, it notifies A that it is ready for migration (see (4) in Fig. 2). B may receive asynchronous messages (e.g., Packet-In) while completing this role change, but it does not process them since A is still the master one. In order to identify a precise timing for the migration, A transmits a dummy Add-Flow-Mod message to S by adding a new flow entry without matching any incoming packet (see (5) in Fig. 2). Subsequently, it forwards another Delete-Flow-Mod message to remove this entry (see (6) in Fig. 2). Accordingly, the switch S forwards a Flow-Removed message to both controllers because A and B are in the Equal roles (see (7) in Fig. 2). This Flow-Removed message triggers an ownership transfer for S from A to B, and henceforward, only B takes a charge of processing all the messages forwarded from S. At last, A sends a Terminate Migration message to B (see (8) in Fig. 2), which then sends a Role-Request message to S to change A to the Slave role upon receiving the Role-Request message[1] (see (9)-(10) in Fig. 2).

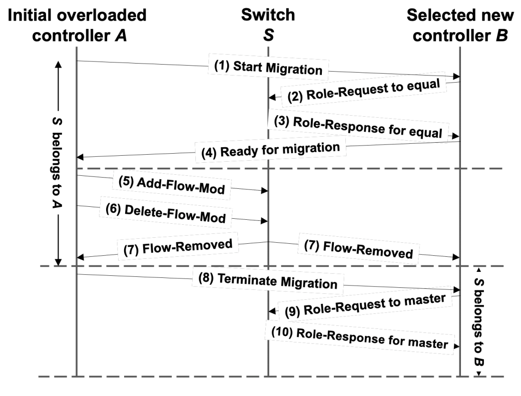

Fig. 2. The procedure of SM.

Ⅲ. LSM Scheme

In LSM, we assume that a global network view is shared among distributed controllers. Whenever an SM event is accomplished, the controllers must conduct synchronization to preserve the global network view. The observation history of forwarding Packet-In messages is denoted by $\{o^1, o^2, \ldots, o^m\}$, which consists of m independent observations for one controller. Further, $o^i = 1$ if a Packet-In message is processed by the controller; otherwise, $o^i = 0$.

Based on these observations, we aim to derive the likelihood function to fit the parameters of a model $p(o,h)$, which is given by

$$\hat{\varphi} = \sum_{i=1}^{m} \log p(o^i;\varphi) = \sum_{i=1}^{m} \log \sum_{h^i} p(o^i,h^i;\varphi). \tag{6}$$

Note that (6) is derived from the marginal distribution of o^i, and there is no direct way to calculate the value of φ. Here, Jensen's inequality is adopted. Assuming that Q_i is the distribution over h for each i, the right-hand equation of (6) is derived as

$$\begin{aligned}\sum_{i=1}^{m} \log \sum_{h^i} p(o^i,h^i;\varphi) &= \sum_{i=1}^{m} \log \sum_{h^i} Q_i(h^i) \frac{p(o^i,h^i;\varphi)}{Q_i(h^i)} \\ &\geq \sum_{i=1}^{m} \sum_{h^i} Q_i(h^i) \log \frac{p(o^i,h^i;\varphi)}{Q_i(h^i)}.\end{aligned} \tag{7}$$

Specifically, $f(x) = \log x$ is a concave function because $f''(x) = -1/x^2 < 0$, while $x \in R^+$. Hence, $\sum_{h^i} Q_i(h^i)[\frac{p(o^i,h^i;\varphi)}{Q_i(h^i)}]$ is the expectation of $[p(o^i,h^i;\varphi)/Q_i(h^i)]$. Consequently, by reusing Jensen's inequality, we have

$$f(E_{h^i \sim Q_i}[\frac{p(o^i,h^i;\varphi)}{Q_i(h^i)}]) \geq E_{h^i \sim Q_i}[f([\frac{p(o^i,h^i;\varphi)}{Q_i(h^i)}])] \tag{8}$$

where $h^i \sim Q_i$ indicates that the expectations are with respect to h^i obtained from Q_i, which allows us to proceed (7). Furthermore, for any set of distributions Q_i, the lower bound of $\hat{\varphi}$ can be obtained from the inequality of (7). That is, the algorithm can seek the optimal decision by raising the lower bound continuously.

To tighten the lower bound, the step involving Jensen's inequality in (7) is expected to hold with the equality. Therefore, constant c is taken over the expectation as

$$c = \frac{p(o^i,h^i;\varphi)}{Q_i(h^i)} \tag{9}$$

where $\sum_h Q_i(h^i) = 1$ and $Q_i(h^i)$ is the posterior distribution of h_i given o^i and φ. Therefore, according to the conditional probability function, $Q_i(h^i)$ is derived as

$$Q_i(h^i) = \frac{p(o^i,h^i;\varphi)}{p(o^i;\varphi)} = p(h^i \mid o^i;\varphi). \tag{10}$$

That is, Q_i is set to be the posterior distribution of h^i given o^i and setting parameter φ. Based on (7) and (10), we provide a pseudocode of the EM-based controller selection algorithm in Fig. 3. First, the initial forwarding histories of each controller are randomly selected (see line 1 in Fig. 3). After that, the expectation (E) step is conducted based on (10), providing a lower bound on the loglikelihood $\hat{\varphi}$, which needs to be maximized (see line 3 in Fig. 3). In addition, for the maximization (M) step, we must maximize (7) to obtain a new setting of φ (see line 4 in Fig. 3). Finally, these two steps are executed repeatedly until a preset iteration count B is reached (see lines 2 to 5 in Fig. 3). The obtained results are sorted in a descending order (see line 6 in Fig. 3), and the smallest one is selected as the target controller, which implies that the controller processes the smallest number of Packet-In messages (i.e., it is underutilized) and it is the most appropriate controller for SM (see line 7 in Fig. 3).

Algorithm 1 EM-based controller selection algorithm.

```
1: Choose initial forwarding histories of each controller
   $\{o^1, o^2, \dots, o^m\}$
2:    repeat
3:        for each $i$
              $Q_i(h^i) = p(h^i|o^i;\varphi)$
4:        for each $Q_i(h^i)$
              $\varphi = \arg\max_{\varphi} \sum_i \sum_{h^i} Q_i(h^i) \log \frac{p(o^i,h^i;\varphi)}{Q_i(h^i)}$
5:    until satisfying $B$
6: Sort $\varphi$ in descending order
7: Adopt the controller with the smallest $\varphi$ for SM operation
```

Fig. 3. EM-based controller selection algorithm.

Ⅳ. Evaluation Results

To evaluate the performance of LSM, we built a testbed using three OpenDayLight controllers supporting the OpenFlow protocol 1.3v. The Mininet emulator supporting 2.3v is installed on Intel i7 processor with 32G RAM PC. Each physical machine runs Ubuntu 18.04.5 LTS with JDK1.8. A well-known fat-tree topology, which consists of 25 core switches, 50 aggregation switches, and 50 edge switches, was considered for performance evaluation. It is hard to overload a SDN controller due to the loop-back problem in Mininet. To handle this issue, the controller capacity is maintained as low as possible (i.e., 2000 Packet-In message per second), and the CPU and bandwidth capabilities follow the same ones in[5]. The iPerf and Cbench tool are used for traffic generation and measurement. The packet arrival rate is set to 1000 pps and each packet size is set to 1.5 KB.

The performance of LSM (in terms of throughput and packet loss rate) is compared with 1) SMDM, which selects the controller based on the migration efficiency[2], 2) DNMA, which selects the nearest controller as the migration controller[4], and 3) SMCLBRT, which selects the controller considering the response time[5]. Note that all of them follow the same procedure in Fig. 2; however, they choose different target controllers depending on their algorithms.

4.1 Throughput

Fig. 4 shows the number of migrated switches in different schemes. It can be seen that LSM has the smallest number of migrated switches whereas DNMA has the largest one. This results affect the TCP throughput as shown in Fig. 5.

Fig. 5 shows the TCP throughput when TCP flows have run for 30 minutes on the given fat-tree network topology. Note that LSM always selects the latest ten observations as the initial training data and the iteration count B is set to five. Fig. 5 shows that LSM outperforms all the other schemes. This is because LSM always selects the recent Packet-In message records as its initial parameters, which affects the target controller decision significantly.

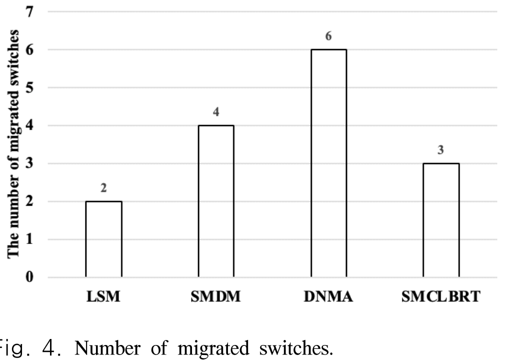

Fig. 4. Number of migrated switches.

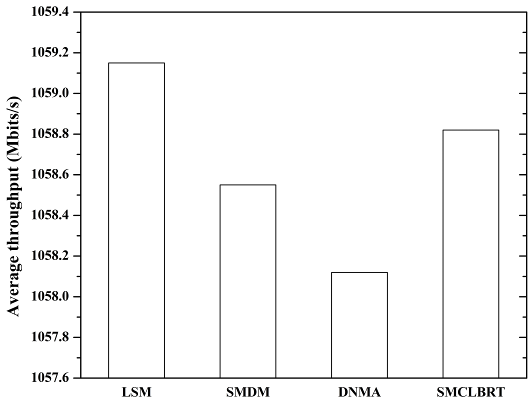

Fig. 5. Average throughput.

Thus, it can select the most appropriate controller for SM by considering current network situations and avoid any potential load imbalance situations.

That is, LSM has strong vitality regardless of the network changes. Furthermore, LSM provides a faster controller selection procedure that mitigates the Packet-In processing delay at the controller, compared with the efficiency-aware scheme of SMDM and the complex decision process of SMCLBRT. Specifically, the time complexities of LSM/DNMA, SMDM and SMCLBRT are $O(n^2)$, $O(n^4)$, and $O(n^3)$, respectively.

As shown in Fig. 4, DNMA always chooses the nearest neighbor controller as the target and therefore the selected controller may be overloaded which leads new load imbalance, and DNMA shows the lowest throughput.

4.2 Packet Loss Rate

Fig. 6 shows the packet loss rate in UDP flows of the four SM schemes. Note that packet loss event can occur due to the buffer overflow of the switch during SM. Intuitively, as more packets are transmitted, more packets can be lost as a result of the inappropriate controller selection. From Fig. 6, it can be found that the packet loss rate of LSM is found to be the lowest. Approximately, LSM shows the reduced packet loss rates by 14.7%, 23%, and 8.3%, respectively, compared with SMDM, DNMA, and SMCLBRT. This is because the target controller selection in LSM contributes to avoid new possible

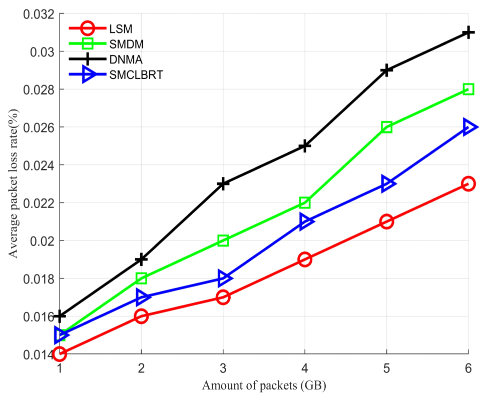

Fig. 6. Average packet loss rate.

unevenness among controllers and the packet loss events.

V. Conclusion

In this paper, we proposed the LSM scheme based on the EM algorithm, which utilizes the record of Packet-In message forwarding history as the initial learning parameters for appropriate controller selection. The procedures to select the switch candidate and target controller in LSM are more straightforward and adaptive to dynamic network conditions than those in existing schemes. The evaluation results demonstrated that LSM outperforms the classical and state-of-the-art SM schemes in terms of throughput and packet loss rate. Therefore, we believe LSM can be widely employed in practical SDNs. In our future work, we will leverage advanced learning techniques such as deep reinforcement learning for enhanced load balancing in SM, and more realistic network topologies will be considered.

References

[1] Y. Xu, M. Cello, I. Wang, A. Walid, G. Wilfong, C. Wen, M. Marchese, and H. Chao, Dynamic, “Switch migration in distributed software-defined networks to achieve controller load balance,” *IEEE J. Sel. Areas in Commun.*, vol. 37, no. 3, pp. 515-529, Mar. 2019.

[2] C. Wang, B. Hu, S. Chen, D. Li, and B. Liu, “A switch migration-based decision-making scheme for balancing load in SDN,” *IEEE Access*, vol. 5, pp. 4537-4544, Mar. 2017.

[3] Y. Kyung, K. Hong, S. Park, and J. Park, “Load distribution method over multiple controllers in SDN,” *J. KICS*, vol. 40, no. 6, pp. 1114-1116, Jun. 2015.

[4] A. Dixit, F. Hao, S. Mukherjee, T. Lakshrnan, and R. Kompella, “Towards an elastic distributed SDN controller,” in *Proc. ACM SIGCOMM Wkshps. Hot Topics in Softw. Defined Netw.*, pp. 7-12, Aug. 2013.

[5] J. Cui, Q. Lu, H. Zhong, M. Tian, and L. Liu, “A load-balancing mechanism for distributed sdn control plane using response time,” *IEEE Trans. Netw. and Serv. Manag.*, vol. 15, no. 4, pp. 1197-1206, Dec. 2018.

[6] K. Sahoo, et al., “ESMLB: Efficient switch migration-based load balancing for multicontroller SDN in IoT,” *IEEE Internet of Things J.*, vol. 7, no. 7, pp. 5852-5860, Jul. 2020.

[7] D. Ono, S. Izumi, T. Abe, and T. Suganuma, “A design of port scan detection method based on the characteristics of packet-in messages in openflow networks,” in *Proc. Asia-Pacific Netw. Oper. and Manag. Symp.*, pp. 120-125, Sep. 2020.

[8] O. Adekoya, A. Aneiba, and M. Patwary, “An improved switch migration decision algorithm for SDN load balancing,” *IEEE Open J. Commun. Soc.*, vol. 1, pp. 1602-1613, Oct. 2020.

설　해 (Xue Hai)

2014년 : 건국대학교 Information and Communication Engineering 학사
2016년 : 한양대학교 Computer and Software 석사
2020년 : 성균관대학교 Computer Engineering 박사
2021년~현재 : University of Shanghai for Science and Technology (USST) 조교수
<관심분야> 소프트웨어 정의 네트워크, 머신 러닝, 엣지 컴퓨팅
[ORCID:0000-0002-4567-6771]

백 상 헌 (Sangheon Pack)

2000년 : 서울대학교 Computer Engineering 학사
2005년 : 서울대학교 Computer Engineering 박사
2005년~2006년 : 캐나다 워털루 대학교 Postdoctoral Fellow
2007년~현재 : 고려대학교 Electrical Engineering 교수
<관심분야> 소프트웨어 정의 네트워크, P4, B5G/6G, 엣지 컴퓨팅, 차량 네트워크
[ORCID:0000-0002-1085-1568]

김 기 훈 (Kihun Kim)

2002년 : 명지대학교 정보통신공학 학사
2015년 : 아주대학교 IT융합공학 석사
2008년~현재 : 한화시스템 수석연구원
<관심분야> 무선통신시스템, 국방전술 네트워크, IoT, SDN
[ORCID:0000-0002-3519-0268]

박　현 (Hyun Park)

2005년 : 광운대학교 컴퓨터공학부 학사
2007년 : 광운대학교 컴퓨터공학과 석사
2010년~현재 : 한화시스템 전문연구원
<관심분야> 군 전술통신망, 모바일 네트워크
[ORCID:0000-0003-2643-3130]