https://doi.org/10.7840/kics.2025.50.5.773

K-means를 활용한 Simulated Annealing 기반 컨트롤러 배치 알고리즘

김 하 은◆, 류 동 균*, 최 홍 록*, 이 상 훈*, 김 준 형**,
박 진 호**, 박 현**, 김 기 훈**, 안 성 준***, 백 상 헌°

K-means Assisted Simulated Annealing Algorithm for Controller Placement

Haeun Kim◆, Dongkyun Ryoo*, Hongrok Choi*, Sanghoon Lee*, Junhyeong Kim**,
Jinho Park**, Hyun Park**, Kihun Kim**, Sungjoon Ahn***, Sangheon Pack°

요 약

소프트웨어 정의 네트워킹 환경에서 다중 컨트롤러 배치 문제는 지연 시간 최적화와 컨트롤러 장애 등 다양한 요인을 고려해야 하는 복잡한 최적화 문제이다. 이에 본 논문에서는 k-means와 시뮬레이티드 어닐링 알고리즘을 결합한 컨트롤러 배치 알고리즘 KASA-CP을 제안한다. 제안된 알고리즘은 k-means 알고리즘으로 초기 배치를 수행하여 시뮬레이티드 어닐링의 연산 효율성을 높임으로써 대규모 네트워크 환경에서도 효율적으로 동작할 수 있다. 실험 결과, 본 알고리즘은 기존 시뮬레이티드 어닐링 알고리즘에 비해 실행 시간 및 평균 지연 시간 측면에서 우수한 성능을 보여주었다.

Key Words : SDN, Multi-Controller Placement, Optimization, Failure Resilience, Backup Controller

ABSTRACT

The multi-controller placement problem (MCPP) in software-defined networking is a complex optimization problem. This paper proposes the K-means Assisted Simulated Annealing Controller Placement (KASA-CP) algorithm, which combines k-means and simulated annealing to address MCPP efficiently. KASA-CP improves the computational efficiency of simulated annealing by using k-means for initial placement, ensuring effective operation in large-scale networks. Experimental results show that KASA-CP outperforms the basic simulated annealing algorithm in terms of execution time and average latency.

※ 이 논문은 2024년 정부(방위사업청)의 재원으로 국방신속획득기술연구원의 지원을 받아 수행된 연구임(UU230002D)
◆ First Author : Korea University, School of Electrical Engineering, 170999@korea.ac.kr, 학생회원
° Corresponding Author : Korea University, School of Electrical Engineering, shpack@korea.ac.kr, 종신회원
* Korea University, School of Electrical Engineering, {rdk0610, ghdfhrooo, shunlee134}@korea.ac.kr, 학생회원
** Hanwha System, {jhkim94, jinho83.park, h82.park, kihun19.kim}@hanwha.com
*** Defense Rapid Acquisition Technology Research Institute, sj_ahn@add.re.kr
논문번호 : 202412-316-C-RN, Received December 12, 2024; Revised December 14, 2024; Accepted December 14, 2024

Ⅰ. 서 론

소프트웨어 정의 네트워킹 (Software-Defined Networking, SDN)[1]은 네트워크의 데이터 평면과 제어 평면을 분리하여 네트워크를 보다 유연하고 효율적으로 관리할 수 있도록 설계된 네트워크 구조이다. 네트워크 트래픽을 전달하는 데이터 평면과 트래픽을 제어하는 제어 평면을 분리하는 구조를 사용함으로써 SDN은 중앙 집중식 네트워크 제어를 가능하게 하고 높은 유연성과 동적 구성을 지원한다. 그러나 중앙 집중식 제어 방식이 모든 상황에서 효과적인 것은 아니다. 네트워크 규모가 크거나 복잡도가 높을 경우, 단일 컨트롤러에 과부하가 발생하거나 장애가 생길 위험이 있으며, 이는 전체 네트워크 성능 저하로 이어질 수 있다[2].

이러한 문제를 해결하기 위해 다중 컨트롤러 구조가 제안되었다. 다중 컨트롤러 구조에서는 여러 컨트롤러가 네트워크의 제어 작업을 분담함으로써 중앙 집중식 제어에서 발생할 수 있는 컨트롤러 과부하를 완화하고, 네트워크 규모의 확장을 가능하게 한다. 이 구조에서 각 컨트롤러를 네트워크 상에 어떻게 배치할 것인지가 중요한 연구 과제로 대두되었으며, 이를 다중 컨트롤러 배치 문제 (Multi-Controller Placement Problem, MCPP)[3]라 한다. 컨트롤러 배치는 네트워크의 지연 시간, 부하 공평성, 장애 복구 성능에 중대한 영향을 미치며, 부적절한 배치는 네트워크의 병목 현상이나 과도한 지연 시간을 초래할 수 있다. 하지만 컨트롤러 배치는 NP-hard 문제이며, 네트워크의 규모와 복잡성이 증가함에 따라 모든 가능한 조합을 탐색하여 최적해를 찾는 brute-force 알고리즘은 현실적으로 적용하기 어렵다. 따라서 대규모 네트워크 환경에서도 효율적으로 동작할 수 있는 컨트롤러 배치 알고리즘이 요구된다.

최근 이러한 요구를 충족하기 위해, SDN 환경에서 MCPP를 해결하기 위한 다양한 휴리스틱 알고리즘이 제안되고 있다. 그 중에서도 주목받고 있는 시뮬레이티드 어닐링 (Simulated Annealing, SA)[4]은 금속을 서서히 냉각시켜 결정을 형성하는 물리적 과정을 모방한 알고리즘이다. 이 알고리즘은 초기에는 탐색 범위를 넓게 설정하여 해를 탐색하고, 반복적인 탐색 과정을 통해 점진적으로 최적해에 수렴하도록 설계되었다. 특히, 시뮬레이티드 어닐링은 지역 최적해에 빠지는 것을 방지하는 메커니즘을 포함하고 있어, MCPP와 같이 복잡한 최적화 문제에서 전역 최적해를 탐색하는 데 효과적으로 적용될 수 있다.

시뮬레이티드 어닐링 알고리즘을 활용한 Simulated Annealing Partition-based K-Means (SAPKM)[5] 알고리즘은 소프트웨어 정의 광역 네트워크 (Software-Defined Wide Area Network, SD-WAN) 환경에서 MCPP를 풀기 위해 제안되었다. SAPKM은 네트워크를 k-means[6] 알고리즘을 적용하여 여러 서브 도메인으로 분할한 후, 시뮬레이티드 어닐링으로 각 서브 도메인에 최적의 컨트롤러를 배치하는 방식으로 동작한다. 이 과정에서 목표 함수를 통해 지연 시간을 최소화하고 각 도메인의 부하를 균등하게 분산시키는 데 중점을 둔다. 그러나 SAPKM은 컨트롤러 장애 시 복구 메커니즘을 고려하지 않아 네트워크 신뢰성 측면에서 한계를 가진다.

이를 해결하기 위해, 컨트롤러 장애 상황에서도 네트워크 성능 저하를 방지할 수 있는 다양한 기법들이 연구되고 있다. 대표적인 기법으로 기존 컨트롤러의 장애 발생 시 이를 대체할 수 있는 백업 컨트롤러 개념을 도입하는 방법이 제안되었다. Simulated Annealing-based Optimal Controller Placement with Capacity Constraint and Failure Awareness (SA-FFCCPP)[7]은 SDN 환경에서 컨트롤러 용량 제약과 장애 시나리오를 고려하여 메인 컨트롤러에 대한 백업 컨트롤러를 배치하는 알고리즘을 제안한다. SA-FFCCPP는 최악의 지연 시간 (worst-case latency)을 최소화하는 목표 함수를 사용하여 백업 컨트롤러 배치로 인한 지연 시간을 최적화하는 데 초점을 맞춘다. 이러한 메커니즘을 사용하여 백업 컨트롤러의 최악의 지연 시간을 줄이고, 장애 발생 시에도 네트워크의 안정적인 연결성을 제공함을 보여준다.

기존의 SDN 환경에서 MCPP를 해결하기 위한 연구들은 주로 부하 균형과 지연 시간 최적화에 중점을 두고 있다. 그러나 이러한 고려 요소들만으로는 네트워크의 동적 특성이나 장애 발생 등에 대한 대응력 측면에서 충분하지 않다. 예를 들어, 네트워크 트래픽이 급격히 증가하거나 특정 컨트롤러에 과도한 트래픽이 집중될 경우, 컨트롤러의 용량 제약을 초과하여 네트워크 성능이 저하될 위험이 있다. 또한, 컨트롤러 장애 시 신속한 백업 컨트롤러로의 전환이 이루어지지 않으면 네트워크의 지연 시간이 급격히 증가할 수 있다. 따라서 지연 시간과 부하 균형을 최적화하는 것 외에도, 백업 컨트롤러 할당 및 컨트롤러 용량 제약을 종합적으로 고려한 최적화가 필요하다. 이를 통해 장애 발생 시에도 신속하게 대응하여 최악의 지연 시간을 최소화하는 것이 매우 중요하다.

본 연구에서는 백업 컨트롤러 배치를 포함하는 컨트롤러 배치 알고리즘 K-means Assisted Simulated Annealing algorithm for Controller Placement (KASA-CP)을 제안

한다. 본 알고리즘은 스위치와 메인 컨트롤러, 그리고 스위치와 백업 컨트롤러 간의 평균 지연 시간을 최적화하는 것을 목표로 한다. 실험 결과, KASA-CP는 k-means와 시뮬레이티드 어닐링으로 구성된 두 단계의 알고리즘을 통해 연산 복잡도를 효과적으로 줄일 수 있음을 입증하였다.

Ⅱ. 문제 정의

본 연구에서는 SDN 환경에서 MCPP를 해결하기 위한 알고리즘을 제안하며, 시스템 모델은 그림 1과 같다. 네트워크는 그래프는 $G = (S, E)$로 표현되며, $S = \{s_1, s_2, \cdots, s_M\}$는 M개의 스위치 집합, E는 스위치 간 물리적 링크를 나타낸다. 각 링크의 길이는 해당 링크의 출발지와 목적지 노드 간 거리 및 네트워크 특성에 따라 결정되며, 이는 전파 지연 시간으로 변환된다. 컨트롤러 집합은 $C^k = \{c_1^k, c_2^k, \cdots, c_N^k\}$ (k = 1, 2)로 정의되며, k=1은 메인 컨트롤러를, k=2는 백업 컨트롤러를 나타낸다. 네트워크 내 메인 컨트롤러와 백업 컨트롤러는 각각 N개로 설정된다. 이때 모든 스위치는 OpenFlow 기능을 갖추고 있다고 가정하므로, 컨트롤러는 네트워크 내의 모든 스위치 위치에 배치될 수 있다. 스위치 s_i와 컨트롤러 c_j^k 사이의 최단 경로는 d_{ij} ($1 \le i \le M$, $1 \le j \le N$)로 나타내며, 이는 전파 지연 시간을 의미한다. 또한 스위치 s_i의 flow request rate는 L_i으로 표현되며 각 컨트롤러 c_j^k의 용량은 U_j^k로 나타낸다. 컨트롤러 노드 장애의 경우 컨트롤러의 개수를 초과하지 않는 범위 내에서 발생할 수 있으며, 장애 노드의 개수는 관리자가 설정할 수 있다. 이때 컨트롤러 장애는 스위치에서 감지할 수 있음을 가정한다.

본 연구에서 제안된 알고리즘의 목표는 각 스위치와 연결된 메인, 백업 컨트롤러와의 평균 지연 시간을 최소화하는 컨트롤러 배치를 수행하는 것이다. 따라서 제안된 네트워크 모델에서 스위치와 컨트롤러 간의 평균 전송 지연 시간을 최소화하기 위한 컨트롤러 배치 문제는 아래와 같은 최적화 문제로 정의할 수 있으며 식 (1)은 목표 함수를 나타낸다. x_{ij}^k는 이진 변수로, $x_{ij}^1 = 1$일때 컨트롤러 c_i^1가 스위치 s_i의 메인 컨트롤러임을, $x_{ij}^2 = 1$ 일때 컨트롤러 c_i^2가 스위치 s_i의 백업 컨트롤러임을 의미한다.

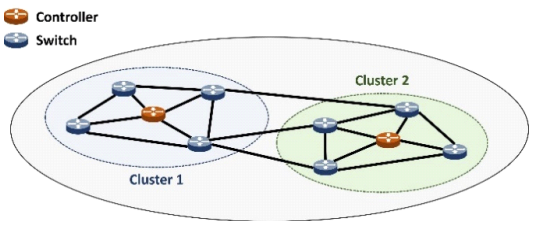

그림 1. 시스템 모델
Fig. 1. System model

$$\min\left\{\frac{1}{2|M-2N|}\sum_{i\in S}\sum_{j\in C}\sum_{k=1}^{2} d_{ij}x_{ij}^k\right\} \quad (1)$$

$$\sum_{j\in C} w_j^k = N, (k=1,2) \quad (2)$$

$$\sum_{j\in C} x_{ij}^k = 1, (k=1,2) \quad (3)$$

$$\sum_{i\in S} L_i x_{ij}^k \le U_j w_j^k, \forall j \in C \quad (4)$$

$$d(c_j^1, c_j^2) \le \delta_{max} \quad (5)$$

위의 수식에서 w_j^k는 이진 변수로, 컨트롤러가 j에 위치하면 1을, 그렇지 않으면 0을 나타낸다. 식 (2)는 w_j^k의 합이 총 N이 되도록 하여 N개의 컨트롤러가 모두 배치되도록 보장하는 제약 조건을 의미하며, 메인 컨트롤러와 백업 컨트롤러 모두에 대해 (k=1, 2) 만족한다. 식 (3)은 각 스위치에 메인 컨트롤러와 백업 컨트롤러가 하나씩 할당하도록 하며, 마찬가지로 메인 컨트롤러와 백업 컨트롤러 모두에 대해 (k=1, 2) 만족한다. 식 (4)는 각 컨트롤러에 할당된 스위치들의 flow request rate가 컨트롤러의 용량을 초과하지 않도록 함으로써 컨트롤러가 정상적으로 동작하도록 한다. 식 (5)는 메인 컨트롤러와 백업 컨트롤러 간의 응답 시간이 특정 임곗값을 초과하지 않도록 하는 제약 조건이다. 이 조건은 컨트롤러의 장애 발생 시 복구 성능을 보장하도록 한다. δ_{max}은 네트워크 관리자가 설정할 수 있으며 이 값에 따라 선택 가능한 백업 컨트롤러의 범위가 달라져 최종 성능에 영향을 미친다. 따라서 장애 빈도나 복구 시간 등을 고려하여 δ_{max} 값을 설정해야 한다.

Ⅲ. 지연 시간 최소화를 위한 컨트롤러 배치 알고리즘

본 논문에서 제안하는 알고리즘은 스위치와 컨트롤러의 평균 지연 시간을 최소화하고, 컨트롤러 장애 시 백업 컨트롤러의 원활한 동작을 보장하는 것을 목표로

Algorithm 1 KASA-CP Algorithm

K-means Initialization
Randomly select N initial centroids from S as C_{init}^1
while centroids have not converged **do**
  for each switch $s_i \in S$ **do**
    Assign s_i to the nearest centroid based on d_{ij}
    if assigned cluster exceeds capacity U_j **then**
      Reassign the switch with the largest delay
  Update centroids based on assigned switches
  for each cluster **do**
    Assign C_{init}^2 with the minimum average delay
Simulated Annealing Optimization
Initialize parameters $\{T_i, T_f, I_{max}, \alpha, O_{optimal}, iterations\}$
while $T > T_f$ **do**
  Generate a new solution
  Randomly move a main controller to a new position in S
  Reassign controllers based on the new position
  Compute O_{new} for the new solution
  Calculate $\Delta = O_{new} - O_{optimal}$
  if $\Delta < 0$ **or** $\Pr(\Delta, T) > Q$ **then**
    Accept the new solution
    Update $O_{optimal}$
  Decrease T using $T = \alpha \cdot T$
Output Final controller placements C^1, C^2

그림 2. KASA-CP 알고리즘
Fig. 2. KASA-CP algorithm

한다. KASA-CP 알고리즘은 메인 컨트롤러와 백업 컨트롤러의 평균 지연 시간을 동시에 최적화하는 복잡한 MCPP를 해결하기 위해 두 단계로 구성된다. 일반적으로 MCPP에서 전역 최적해를 탐색하는 데 사용되는 시뮬레이티드 어닐링 알고리즘은 복잡한 문제에서 효율적인 배치를 도출하는 데에 유용하다. 하지만, 무작위 탐색 방식으로 인해 네트워크 규모가 커질수록 연산 비용이 증가하고 수렴 속도가 느려지는 한계가 있다[8]. 이러한 문제를 완화하기 위해 본 연구는 시뮬레이티드 어닐링을 적용하기 전 단계에서 k-means 알고리즘으로 초기 컨트롤러 배치를 도출함으로써 탐색 효율성과 수렴 속도를 향상시키는 알고리즘을 제안한다. 두 단계의 최적해 탐색을 통해 최종 해가 전역 최적해에 근접하도록 하며, 수렴 속도를 향상시킬 수 있다.

KASA-CP 알고리즘의 동작과정은 다음과 같다. 컨트롤러 수 N이 주어졌을 때, 먼저 k-means 알고리즘을 통해 초기 컨트롤러의 위치 및 클러스터를 형성한다. 이후, 시뮬레이티드 어닐링 알고리즘을 사용하여 최적의 컨트롤러 위치를 탐색함으로써 스위치와 컨트롤러 간 평균 지연 시간을 최소화한다.

3.1 K-means 기반 초기 컨트롤러 배치

KASA-CP의 첫 번째 단계에서는 k-means 알고리즘을 사용한 초기 컨트롤러 배치가 수행된다. 먼저 네트워크 내 노드에서 무작위로 N개의 중심점을 선택하여 메인 컨트롤러의 초기 위치로 설정한다. 이후, M개의 스위치에 대해 메인 컨트롤러와의 최단 경로 지연 시간을 전파 지연 시간을 기준으로 계산하며, 이때 Dijkstra[9] 알고리즘을 활용한다. 각 스위치는 가장 가까운 메인 컨트롤러에 우선적으로 할당되며, 이후 컨트롤러가 할당된 스위치들의 총 flow request rate를 수용 가능 여부를 평가한다. 만약 총 flow request rate가 컨트롤러의 용량을 초과하는 경우, 지연 시간이 큰 스위치부터 우선적으로 두 번째 지연 시간을 가지는 대체 컨트롤러로 재할당하는 과정을 반복하여, 최종적으로 모든 요청이 용량 제약 내에서 처리되도록 한다.

초기 클러스터가 형성된 후, 메인 컨트롤러의 위치를 최적화하기 위해 반복적으로 클러스터 내 스위치와의 거리를 고려하여 목표 함수를 만족하는 중심점을 찾는 과정이 진행된다. 이 과정에서 컨트롤러 배치는 단순히 거리 기반, 즉 전파 지연 시간을 기준으로 이루어지며 클러스터 중심점이 더 이상 변하지 않을 때 최적화 종료된다. 최종 중심점은 초기 메인 컨트롤러 C_{init}^1로 설정되며, 이 메인 컨트롤러를 제외한 클러스터 내 스위치 노드 중 하나를 백업 컨트롤러로 지정한다. 이때 클러스터 내 스위치 노드 중 메인 컨트롤러와의 지연 시간이 δ_{max}를 넘지 않으면서 각 스위치 노드와의 평균 전파 지연 시간이 가장 작은 노드를 초기 백업 컨트롤러 배치 C_{init}^2로 설정한다.

K-means 알고리즘을 통해 형성된 초기 클러스터 및 컨트롤러 배치는 이후 시뮬레이티드 어닐링 알고리즘의 초기 조건으로 활용된다. 이와 같은 초기 배치는 최적해 탐색 효율성을 높이고, 알고리즘의 수렴 속도를 향상시키는 데 기여한다.

3.2 시뮬레이티드 어닐링 기반 컨트롤러 배치 최적화

첫 번째 배치 단계에서 생성된 초기 컨트롤러 배치 C_{init}^1, C_{init}^2를 기반으로 시뮬레이티드 어닐링을 수행하여 메인 및 백업 컨트롤러의 위치를 최적화한다. 이 과정은 초기 온도 T_i, 최종 온도 T_f, 반복 횟수 I_{max}, 온도 감소 비율 α와 같은 알고리즘 파라미터와 목표 함수 값 $O_{optimal}$, 반복 횟수를 나타내는 변수 $iterations$의 초기화로 시작한다.

초기화 단계 이후, 시뮬레이티드 어닐링은 3.1에서 생성된 스위치와 컨트롤러의 매핑 정보를 입력으로 받아 현재 각 스위치에 할당된 컨트롤러의 평균 지연 시간을 계산하고 이를 $O_{optimal}$로 저장한다.

최적화 과정에서는 이웃 솔루션을 생성하여 목표 함수 값을 평가하고, 현재 솔루션과 비교하는 과정을 반복

한다. 이웃 솔루션의 생선 과정은 다음과 같다. 먼저 메인 컨트롤러 중 하나의 위치를 무작위로 변경하고, 변경된 위치에 따라 각 스위치의 백업 컨트롤러 매핑을 다시 수행한다. 이때 메인 컨트롤러의 새로운 위치는 기존 클러스터 외부에 있는 스위치 노드로 할당할 수 있다. 변경된 메인 컨트롤러와 백업 컨트롤러 위치에 대해 다시 목표 함수 값을 계산하고, 이를 O_{new}에 저장 후 $O_{optimal}$와 비교하는 과정이 진행된다.

O_{new}와 $O_{optimal}$ 비교를 위해 먼저 이 차이를 계산 후, *Delta* 변수에 저장한다. 이 값은 두 솔루션의 목표 함수의 최적화 정도의 차이를 의미하며, 음수일 경우 새로운 솔루션이 기존 솔루션보다 더 나음을 의미한다. 또한, 0에서 1 사이의 무작위 값을 생성하여 변수 Q에 저장한다. 이 값은 새로운 솔루션의 수용 여부를 결정할 때 확률적 비교를 위한 기준이 된다. O_{new}와 $O_{optimal}$의 차를 의미하는 *Delta*가 음수, 즉 새로운 솔루션이 더 나은 경우에는 식 (6)의 확률이 1보다 크며 이는 Q보다 큰 값을 가지므로 새로운 솔루션이 채택된다. 반면, *Delta*가 양수인 경우 확률이 1보다 작아지며, Q와의 비교를 통해 새로운 솔루션의 채택 여부를 결정하게 된다. $Pr(\Delta, T) > Q$일 경우 새로운 솔루션을 선택하며 $Pr(\Delta, T) < Q$라면 현재의 배치를 유지하게 된다. 이러한 확률적 수용 방식을 적용함으로써 시뮬레이티드 어닐링 알고리즘이 편향되지 않고 다양한 솔루션을 탐색할 수 있도록 한다.

$$Pr(\Delta, T) = e^{-\Delta/T} \tag{6}$$

이웃 솔루션을 생성 및 평가 과정은 온도 T가 최종 온도 T_f에 도달할 때까지 반복된다. 초기 온도가 높은 경우에는 식 (6)의 T 값이 크므로 현재 솔루션에 비해 성능이 낮은 솔루션도 수용할 확률이 상대적으로 높아져 해 공간을 넓게 탐색할 수 있다. 하지만 온도가 점차 낮아짐에 따라 수용 확률이 감소하며, 이를 통해 전역 최적해에 수렴할 수 있도록 한다. 최종적으로, 시뮬레이티드 어닐링을 통해 얻어진 최종 배치 결과는 C^k로 저장된다.

Ⅳ. 시뮬레이션 결과

본 연구에서 제안된 KASA-CP 알고리즘의 성능을 평가하기 위해 Python으로 실험 환경을 구현하고, 세 가지의 시뮬레이션을 수행하였다.

실험 네트워크는 총 51개의 노드로 구성된 spider-web 토폴로지로 설계되었으며, 메인 컨트롤러와 백업 컨트롤러의 개수를 4에서부터 9까지 증가시키며 실험을 진행하였다. 각 스위치는 동일한 flow request rate인 400K requests/sec를 가지며 컨트롤러로 할당된 노드의 용량은 7800K requests/sec로 설정되었다. 이외의 실험 파라미터는 표 1에 정리되어 있다. 시뮬레이션은 총 세 가지로 구성되며 각 시뮬레이션의 결과를 Python으로 구현한 일반적인 시뮬레이티드 어닐링 기반의 컨트롤러 배치 알고리즘 SA와 비교 분석하였다.

표 1. 시뮬레이션 파라미터
Table 1. Simulation parameters

Number of nodes S	51
Initial temperature T_i	10
Final temperature T_f	0.0001
Cooling factor α	0.9
Number of iterations *Maxlter*	30
Number of controllers N	4~9

4.1 실행 시간 비교

제안된 KASA-CP 알고리즘이 SA와 비교하여 실행 시간 측면에서 얼마나 효율적인지를 평가하기 위해, 컨트롤러 개수를 4개에서 9개로 증가시키며 그 실행 시간을 비교하였다. 그림 3에 나타난 결과와 같이 실행시간이 SA에 비하여 크게 단축되는 것을 확인할 수 있다. 이는 KASA-CP가 초기 단계에서 k-means를 사용하여 서브 도메인을 분할하고, 각 서브 도메인 내 초기 컨트롤러 배치를 최적화함으로써 이후 시뮬레이티드 어닐링 과정에서 발생하는 과도한 연산을 줄였기 때문이다. 특히, 컨트롤러 수가 적을수록 두 알고리즘의 실행시간 차가 더 커지는 경향을 보였다. 이를 통해 네트워크 규모가 커질수록 k-means 기반의 초기 배치가 더욱 효과적이며, 제안하는 알고리즘이 계산 복잡도 측면에서 높은 효율성을 제공함을 보여준다.

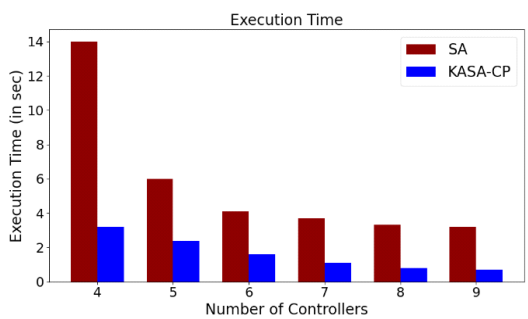

그림 3. 실행 시간 비교
Fig. 3. Comparison of execution time

4.2 스위치와 컨트롤러의 평균 지연 시간

KASA-CP 알고리즘의 평균 지연 시간 최적화 성능을 평가하기 위해, 스위치와 컨트롤러 간의 평균 지연 시간을 비교하였다. 그림 4에서 KASA-CP가 모든 컨트롤러 수에 대해서 SA보다 낮은 평균 지연 시간을 기록하였다. 특히, 컨트롤러의 개수가 8개와 9개일 때 KASA-CP는 SA에 비해 각각 약 83%, 90% 낮은 지연 시간을 보였다. 이는 KASA-CP가 메인 컨트롤러와 백업 컨트롤러 간의 거리 제약 조건을 고려하여 배치 최적화를 수행하기 때문에, 백업 컨트롤러와의 지연 시간도 효과적으로 관리할 수 있음을 보여준다.

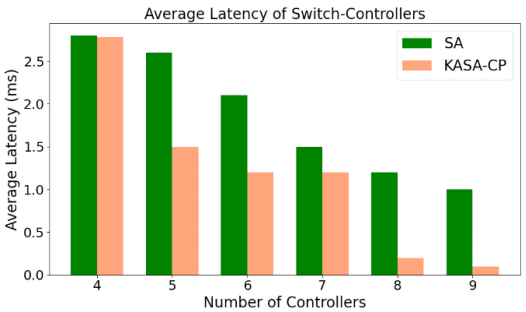

그림 4. 스위치와 컨트롤러의 평균 지연 시간 비교
Fig. 4. Comparison of latency between switch and controllers

4.3 컨트롤러 간 지연 시간 임곗값 만족 비율

그림 5는 KASA-CP 알고리즘에서 메인 컨트롤러와 백업 컨트롤러 간의 지연 시간이 임곗값을 만족하는 비율을 나타낸다. 본 연구에서는 임곗값을 1ms로 설정하였으며, 이를 기준으로 장애 발생 시 신속한 네트워크 복구를 위한 백업 컨트롤러 배치의 효과를 평가하였다. KASA-CP 알고리즘은 모든 실험 시나리오에서 SA보다 항상 높은 만족 비율을 보여주었으며, 이는 제안하는 알고리즘이 컨트롤러 장애 상황에서 네트워크 복원력을 향상시키는데 기여할 수 있음을 보여준다.

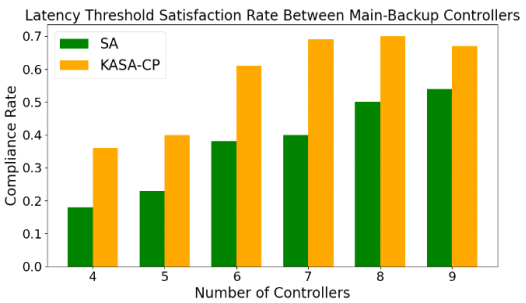

그림 5. 메인 컨트롤러와 백업 컨트롤러 간의 지연 시간 임곗값 만족 비율
Fig. 5. Latency threshold satisfaction rate between main and backup controllers

V. 결 론

본 연구에서는 SDN 환경에서 MCPP를 해결하기 위한 KASA-CP 알고리즘을 제안하였다. 제안된 알고리즘은 지연 시간 최적화를 기반으로 한 컨트롤러 배치와 장애 복구를 위한 백업 컨트롤러 할당을 하나의 목표 함수로 통합하여 최적화를 수행한다. 복잡한 최적화 문제를 효과적으로 해결하기 위해 k-means와 시뮬레이티드 어닐링을 단계적으로 수행하여 컨트롤러의 최적 배치를 도출하는 동시에 연산의 복잡도를 줄였다. 시뮬레이션 결과, KASA-CP는 기존의 시뮬레이티드 어닐링 기반 컨트롤러 배치 알고리즘과 비교하여 더 낮은 연산 복잡도와 지연 시간을 보여주었다.

그러나 k-means는 초기에 랜덤으로 설정한 클러스터 중심에 민감하여, 다른 조건을 고려하더라도 로컬 최적해에 빠질 가능성이 있다. 이에 따라 향후 연구에서는 본 알고리즘 환경에 적합하면서 초기 클러스터링 과정에서의 민감성을 줄일 수 있는 개선된 알고리즘을 고안할 계획이다. 또한, 더욱 다양한 네트워크 토폴로지와 트래픽 변동성을 반영한 실험 환경을 통해 KASA-CP의 적용 범위를 확장하고, 보다 일반화된 성능 평가를 수행할 예정이다.

References

[1] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *ACM SIGCOMM Computer Computer Commun. Review*, vol. 44, no. 2, pp. 87-98, Apr. 2014. (https://doi.org/10.1145/2602204.2602219)

[2] Y. Jeon, S. Jang, S. Pack, K. Kim, and H. Park, "Mobility-aware leader controller selection in distributed SDN," *J. KICS*, vol. 47, no. 1, pp. 76-78, Jan. 2022. (https://doi.org/10.7840/kics.2022.47.1.76)

[3] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. ACM SIGCOMM HoTSDN*, pp. 7-12, Helsinki, Finland, Sep. 2012. (https://doi.org/10.1145/2377677.2377767)

[4] D. Bertsimas and J. Tsitsiklis, "Simulated

annealing," *Statistical Sci.,* vol. 8, no. 1, pp. 10-15, Feb. 1993. (https://doi.org/10.1214/ss/1177011077)

[5] K. Yang, D. Guo, B. Zhang, and B. Zhao, "Multi-controller placement for load balancing in SDWAN," *IEEE Access*, vol. 7, pp. 167278-167289, Nov. 2019. (https://doi.org/10.1109/access.2019.2953723)

[6] A. Likas, N. Vlassis, and J. J. Verbeek, "The global K-means clustering algorithm," *Pattern Recognition*, vol 36, no. 2, pp. 451-461, Feb. 2003. (https://doi.org/10.1016/S0031-3203(02)00060-2)

[7] P. Aravind, G. S. Varma, and P. P. Reddy, "Simulated annealing based optimal controller placement in software defined networks with capacity constraint and failure awareness," *Journal of King Saud University-Computer and Inf. Sci.*, vol. 34, no. 8, pp. 5721-5733, Sep. 2022. (https://doi.org/10.1016/j.jksuci.2021.04.012)

[8] S. Mohanty, K. Kanodia, B. Sahoo, and K. Kurroliya, "A simulated annealing strategy for reliable controller placement in software defined networks," in *Proc. Int. Conf. Signal Process. and Integrated Netw. (SPIN)*, pp. 844-849, Noida, India, Feb. 2020. (https://doi.org/10.1109/spin48934.2020.9071317)

[9] J. R. Jiang, H. W. Huang, J. H. Liao, and S. Y. Chen, "Extending Dijkstra's shortest path algorithm for software defined networking," in *Proc. Asia-Pacific Netw. Operations and Manag. Symposium (APNOMS)*, pp. 1-4, Hsinchu, Taiwan, Sep. 2014. (https://doi.org/10.1109/apnoms.2014.6996609)

김 하 은 (Haeun Kim)

2024년 : 고려대학교 전기전자공학 학사

2024년~현재 : 고려대학교 전기전자공학과 석사과정

<관심분야> 소프트웨어 정의 네트워크, P4

[ORCID:0009-0007-2836-3160]

류 동 균 (Dongkyun Ryoo)

2021년 : 고려대학교 전기전자공학 학사

2021년~현재 : 고려대학교 전기전자공학과 박사과정

<관심분야> 5G/6G 모바일 네트워크, 차량 네트워크, 위성 네트워크

[ORCID:0009-0004-2192-3012]

최 홍 록 (Hongrok Choi)

2018년 : 고려대학교 전기전자공학 학사

2018년~현재 : 고려대학교 전기전자공학과 박사과정

<관심분야> 소프트웨어 정의 네트워크, 위성 네트워크, 머신러닝

[ORCID:0000-0001-8380-8707]

이 상 훈 (Sanghoon Lee)

2024년 : 아주대학교 전자공학 학사

2024년~현재 : 고려대학교 전기전자공학과 석사과정

<관심분야> 소프트웨어 정의 네트워크, P4

[ORCID:0009-0000-6996-7242]

김 준 형 (Junhyeong Kim)

2018년 : 아주대학교 소프트웨어학 학사
2020년 : 아주대학교 컴퓨터공학 석사
2023년~현재 : 한화시스템 선임연구원

<관심분야> 전술통신, 다계층네트워크, Segment Routing, AI
[ORCID:0009-0008-1268-723X]

김 기 훈 (Kihun Kim)

2002년 : 명지대학교 정보통신공학 학사
2015년 : 아주대학교 IT융합공학 석사
2008년~현재 : 한화시스템 수석연구원

<관심분야> 무선통신시스템, 국방전술 네트워크, IoT, 소프트웨어 정의 네트워크
[ORCID:0000-0002-3519-0268]

박 진 호 (Jinho Park)

2010년 : 인제대학교 전자지능로봇공학 학사
2020년 : 세종사이버대학교 정보보호학 석사
2017년~현재 : 한화시스템 전문연구원

<관심분야> 군 전술 네트워크, 보안 통신, 소프트웨어 정의 네트워크
[ORCID:0009-0007-3686-300X]

안 성 준 (Sungjoon Ahn)

1984년 : 연세대학교 전자공학 학사
1986년 : 연세대학교 전자공학 석사
2024년~현재 : 국방신속획득기술연구원 책임기술원

<관심분야> 위성통신, 데이터링크, 위성데이터링크, 저궤도위성, 정지궤도위성
[ORCID:0009-0009-8169-342X]

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]

백 상 헌 (Sangheon Pack)

2000년 : 서울대학교 컴퓨터공학과 학사
2005년 : 서울대학교 컴퓨터공학과 박사
2005~2006년 : 캐나다 워털루 대학교 Postdoctoral Fellow
2007년~현재 : 고려대학교 전기전자공학과 교수

<관심분야> 소프트웨어 정의 네트워크, P4, B5G/6G, 엣지 컴퓨팅, 차량 네트워크
[ORCID:0000-0002-1085-1568]