IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 15, 2019, accepted July 22, 2019, date of publication August 1, 2019, date of current version August 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2932477

Topology-Aware Resource-Efficient Placement for High Availability Clusters Over Geo-Distributed Cloud Infrastructure

TRUONG-XUAN DO AND YOUNGHAN KIM
Internet Infrastructure System Technology Research Center, Soongsil University, Seoul 156-743, South Korea

Corresponding author: Younghak Kim (younghak@ssu.ac.kr)

This work was supported in part by the MSIT(Ministry of Science and ICT), South Korea, under the ITRC (Information Technology Research Center) support program (IITP-2019-2017-0-01633) supervised by the IITP (Institute of Information and Communications Technology Planning and Evaluation), and in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government (MSIT) (No.2017-0-00613, Development of Content-oriented Delay Tolerant networking in Multi-access Edge Computing Environment).

ABSTRACT A management and orchestration framework (MANO) in network function virtualization (NFV) enables the agile deployment and operation of virtual network functions over a geographically distributed cloud infrastructure. This facilitates the deployment of redundancy models (i.e., high availability clusters) over different cloud centers, to guarantee the high availability of network services. In particular, in the telecommunications field, availability and resiliency are always required at a high level. Existing placement algorithms only consider one type of redundancy model at a given time. However, in reality, different redundancy configurations can be utilized to ensure the availability of virtual functions. In this article, we present an optimization model and topology-aware resource-efficient placement algorithm (TARE), which can be employed to optimally deploy high availability clusters with different redundancy configurations over geo-distributed cloud infrastructures. This model takes into account the different requirements of various high availability clusters in terms of bandwidth and computing resource demands. By simulation, the TARE has better performance than other baseline solutions in terms of the bandwidth usage, while maintaining an acceptable level of availability.

INDEX TERMS Network function virtualization, redundancy model, high availability clusters, distributed cloud.

I. INTRODUCTION

Virtual network functions (VNF) [1], which are created by a management and orchestration framework (MANO) [2] over a network function virtualization infrastructure (NFVI) [3], provide a high level of automation and cost reduction. The network service that consists of chained VNFs is described in a cloud orchestration template (e.g., TOSCA), and is deployed by a central MANO. VNFs can be deployed on different cloud centers across different geographical sites. Deployment decisions are determined based on network operators' preferences or placement strategies.

When deploying a network service, reliability is one of the mandatory requirements [4]. A popular way to increase reliability is to configure network functions using redundancy models (a.k.a., high availability clusters). High availability cluster is a group of virtual network functions that act like a single system and provide continuous uptime by employing redundancy models. High availability clusters are often used for load balancing, backup, and failover purposes. In general, there are two types of redundancy models: Active-Standby and Active-Active [5]. The types and configurations of redundancy models depend on the reliability requirements of various network services. In cloud environments, the deployment of high availability clusters can be easily performed in the case of a single cloud center. However, in a geo-distributed cloud infrastructure with multiple cloud centers [6], the VNFs in one cluster can be deployed across different geographical locations. Therefore, placement strategies need to consider more constraints to optimally place these high availability clusters over a geo-distributed cloud infrastructure. First, these placement strategies are required to incorporate constraints related to resource capacities. These resources could be computing resources of cloud centers used to host VNFs.

The associate editor coordinating the review of this manuscript and approving it for publication was Lo'ai A. Tawalbeh.

These resources could be bandwidth resources of physical links in a wide area network (WAN) used to transfer the ongoing state from active to standby functions. Second, when high availability clusters are deployed, minimum availability level constraints must also be satisfied.

Previous studies have focused on placement problems tackling either with availability aspects or resource aspects. Some studies have focused on determining the optimal number of standby functions and their locations to ensure that the availability value is not below a given threshold. These studies have attempted to design high availability clusters and place these clusters to satisfy a specific availability level, without considering the network bandwidth and computing resource demands for different redundancy models. Some other studies have considered placement problems that have taken into account the bandwidth consumption caused by state transfers between active and standby VNFs. However, these are limited to only one redundancy model (i.e., the Active-Standby model). In this paper, our work differs from related research by considering a placement model for a set of high availability cluster requests that have different redundancy configurations. Our work also considers two aspects concurrently: resource demands for different cluster requests and their availability level requirements. To solve this model, we first propose some baseline solutions, called greedy reliable placement, combined with shortest path routing. To solve the model more efficiently, we decompose and group these cluster requests into two categories, corresponding to two kinds of topologies. Then, we propose a topology-aware resource-efficient (TARE) heuristic algorithm to place these cluster requests across geo-distributed cloud centers. Our algorithm exploits the differences in the topologies of different requests to place cluster requests over a geo-distributed cloud infrastructure in a resource-efficient manner. From simulation results, our proposal is observed to outperform other greedy approaches in terms of the bandwidth usage while maintaining the availability level of each request at an acceptable level.

The remainder of this paper is organized as follows. In Section II, we provide a survey of related works, focusing on placement problems of VNFs with and without redundancy. In Section III, we classify and calculate the resource demands for different basic redundancy configurations. In Section IV, we describe how to calculate the availability of one availability request after deployment. In Section V, we formulate our placement problem. In Section VI, we present several common greedy solutions, and our topology-aware approach. Then, Section VII presents our simulation results. Finally, Section VIII concludes the paper.

II. RELATED WORKS

In this section, we present a summary of related works and background knowledge in the area of VNF placement over geo-distributed cloud infrastructure.

A. VNF PLACEMENT WITHOUT REDUNDANCY

Different optimization models have been proposed to deal with the placement problem of VNFs over geo-distributed cloud infrastructures. In [7], the authors placed VNFs in a hybrid environment that consists of dedicated hardware-based services and virtual function-based services with the objective of minimizing the utilized number of physical nodes. A quick algorithm was proposed to solve this model. In [8], the authors attempted to place VNFs to optimize the distance cost between the clients and serving VNFs and the setup costs of VNFs. A near-optimal approximation algorithm was proposed to solve this problem. In [9], the authors proposed a multi-objective optimization model to place the virtual mobile functions with two key design objectives: network load and data center resources. A Pareto optimal solution was proposed to solve this model. In [10], a multi-objective optimization model for placing state management functions of service-based 5G networks was proposed. The authors proposed an adaptive approach to obtain the balance between two key design objectives. In terms of a chain of VNFs, the authors in [11] considered an optimization model that attempted to optimize the total cost including license cost, energy consumption cost, and network cost. The authors proposed a heuristic-based algorithm to obtain a close-to-optimal solution. In [12], the authors proposed a custom greedy algorithm to place and steer traffic for a chain of VNFs. In [13], the authors studied a placement problem of creating optimal service function chains. They attempted to minimize inter-cloud traffic and response time in a multi-cloud scenario, considering important constraints such as total deployment costs and service level agreements (SLAs). However, the above studies did not consider the placement of standby VNFs, which are used to protect the active VNFs in cases of network failure.

B. AVAILABILITY-UNAWARE VNF PLACEMENT WITH REDUNDANCY

In [14], the authors attempted to place standby VMs over a tree topology-based cloud center, and to minimize the reserved network bandwidth in the case of active VM failure. The authors decomposed into two subproblems that first place the standby VMs and then find the most efficient links between standby and active VMs. In [15], the authors assumed that multiple active VMs can share the same standby VMs if these active VMs do not concurrently fail. The authors proposed a mixed integer linear programming (MILP) model to maximize the number of active VMs and minimize the number of standby VMs. However, these related works did not provide any methods to estimate the availability of system.

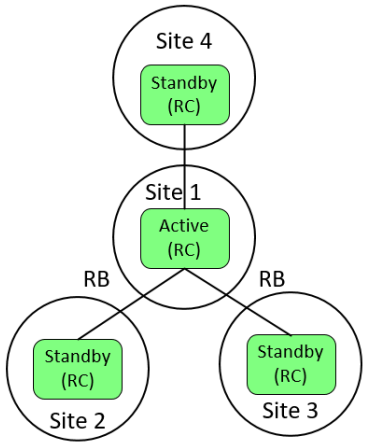
C. AVAILABILITY-AWARE VNF PLACEMENT WITH REDUNDANCY

In [16], the authors considered mobile edge computing (MEC) environments, in which the hosts are not reliable

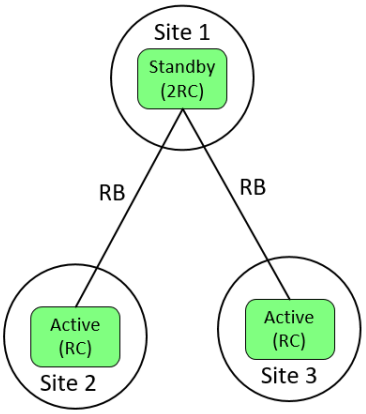
and there are the probability of failures. The authors proposed a model to optimize the CPU/memory cost while keeping the probability of at least the required number of VMs being available above a certain threshold. In [17], the authors characterized the risk of violating the availability requirements by capturing the impact of failure events. The authors also proposed a model to minimize the energy consumption and the risk of violating the availability requirements. However, these related works only provided the estimation of available VMs based on the failure probability, and did not estimate the availability value of protection plan. In [18], the authors considered a shared-risk node group failure model, and estimated the availability of system based on the availability of servers. The authors proposed an exact integer nonlinear programing (INLP) and a heuristic to solve the problem. In [19], the authors addressed a more generic problem in a mobile environment by considering the availability of access links in addition to the availability of servers and software components. The authors proposed greedy heuristics and variable neighborhood search algorithms to maximize the total availability. In [20], the authors considered a placement problem for standby functions in the service-based 5G networks, which takes the availability of different deployment locations into account. A usage-aware heuristic was proposed to efficiently solve the problem. In terms of a chain of VNFs, the authors in [21] proposed a greedy shortest path solution to place the primary functions and backup functions in a chain. In [22], the author proposed to use a depth-first search algorithm to place primary functions and the lowest availability greedy strategy to place backup functions. However, these approaches did not consider various redundancy models that can have different requirements in terms of network bandwidth and computing resource demands in conjunction with the availability value estimation.

III. REDUNDANCY MODELS AND RESOURCE DEMAND

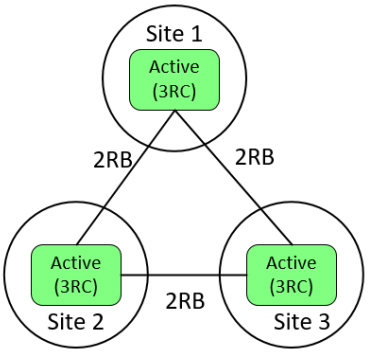
In general, there are two types of redundancy models: active-standby and active-active. For the active-standby model, standby VNFs must be deployed to protect active VNFs. We assume that a general active-standby model including multiple active VNFs and multiple standby VNFs can be decomposed into two basic types: one active multiple standby (oAmS) and multiple active one standby (mAoS), as shown in Fig. 1a and Fig. 1b, respectively. As depicted in Fig. 1a, one active VNF is protected by multiple standby VNFs, which can be employed for highly reliable network services. As depicted in Fig. 1b, multiple active VNFs are protected by one standby VNF, which reduces the required amount of resources for redundancy. The active-standby configuration does not require a load distribution function to balance the traffic load to active VNFs. The active-standby model can be further divided into two modes including hot-standby that consumes resources and cold-standby that does not consume resources. For the active-active model (mA), shown in Fig. 1c, there are no dedicated standby VNFs, and all VNFs are at active status and processing the traffic load.



(a) One active and multiple standby



(b) Multiple active and one standby



(c) Mutiple active

FIGURE 1. Redundancy models.

This configuration requires a load distribution function or a load balancer to steer incoming traffic to active VNFs. This model can be faster compared to the Active-Standby model, because a load balancer can be reconfigured to direct traffic out of the failed VNFs.

In terms of resource consumption, different redundancy models demand different amounts of computing and bandwidth resources. For example, assume that one active VNF consumes the amount *RC* of computing resources, and the virtual link used to replicate the ongoing state [23] of one

active VNF to a standby VNF requires the amount RB of network bandwidth. In the case of one active and three standby shown in Fig. 1b, we need to allocate three standby VNFs with RC of computing resources (hot-standby mode) and 0 of computing resources (cold-standby mode). Three virtual links are required to allocate with bandwidth RB. In the case of two active and one standby shown in Fig. 1a, we need to allocate one standby VNF with the amount $2RC$ of computing resources (hot-standby mode) which is proportional to the number of active VNFs and 0 of computing resources (cold-standby mode). Similarly, two virtual links are required the bandwidth RB. In the case of multiple active no standby shown in Fig. 1c, for the sake of simplicity we assume that the active VNF replicates its state to all other active VNFs [24]. Therefore, the amount of computing resources required for the active VNFs are $3RC$ and $4RC$, respectively. Each virtual link between two active VNFs must reserve the amount of bandwidth $2RB$ for state replication.

In general, we assume that we have a cluster request r with the configuration (ACT_r, STB_r, RC, RB), where ACT_r is a set of active VNFs in the request r, and STB_r is a set of standby VNFs in the request r. The unit computing resource demand of one active VNF is RC and unit bandwidth resource demand for state replication from one active VNF is RB. We denote by RC_a and RC_s the total computing resource demands for the active VNF a and standby VNF s, respectively. We also denote by RB_l the bandwidth resource demand for a virtual link l. The resource demands for each type of redundancy model are calculated as follows.

One Active multiple Standby:

$$\begin{aligned} RC_a &= RC \\ RC_s &= \begin{cases} RC & \text{if hot-standby} \\ 0 & \text{if cold-standby} \end{cases} \\ RB_l &= RB \end{aligned} \tag{1}$$

Multiple Active one Standby:

$$\begin{aligned} RC_a &= RC \\ RC_s &= \begin{cases} \sum^{a \in ACT_r} RC & \text{if hot-standby} \\ 0 & \text{if cold-standby} \end{cases} \\ RB_l &= RB \end{aligned} \tag{2}$$

Multiple Active only:

$$\begin{aligned} RC_a &= \sum^{a \in ACT_r} RC \\ RB_l &= 2RB \end{aligned} \tag{3}$$

IV. AVAILABILITY CALCULATION

When a high availability cluster request is deployed over a geo-distributed cloud infrastructure, the availability of network service provided by the active VNFs, not only depends on the number of active and standby VNFs in the cluster request, but also on the availability of cloud centers where the VNFs in the cluster request are deployed and the availability of virtual links among the VNFs. We denote the availability

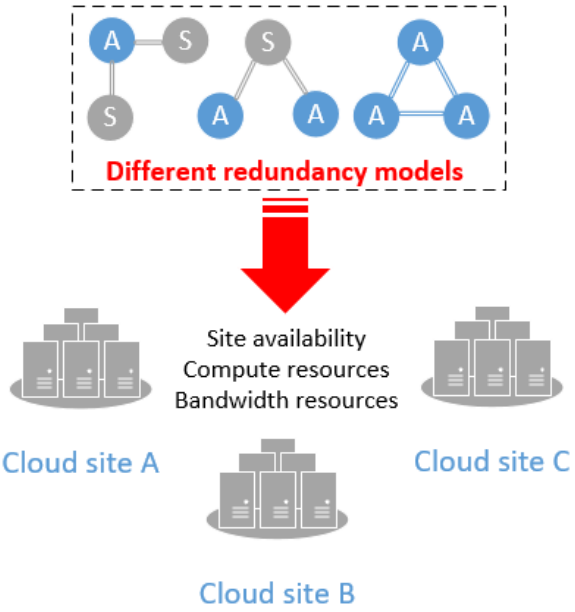


FIGURE 2. High availability cluster deployment over geo-distributed cloud.

values of a cloud center n and physical link e by A_n and A_e, respectively. The availability value of running software instance that provides the network service (i.e., VNF) is A_f. The availability value of hardware server hosting a VNF is A_d. We denote by AL_s the availability of a virtual link used to replicate the state to a standby VNF s. To calculate the availability of each cluster request A_r after placement, we define two decision variables X_n^a and Y_n^s. These are binary variables that are equal 1 if the active VNF a or standby VNF s is placed on the cloud center n and 0 otherwise, where $a \in ACT_r, s \in STB_r, r \in CR$, and $n \in N$. The availability of a cluster request after the deployment is calculated as follows.

$$\begin{aligned} A_r = 1 - \prod_{n \in N} (1 - A_n (1 - \prod_{a \in ACT_r} (1 - A_f A_d X_n^a) \\ \prod_{s \in STB_r} (1 - A_f A_d AL_s Y_n^s))) \end{aligned} \tag{4}$$

The availability of virtual link for state replication to standby node s is calculated as:

$$AL_s = 1 - \prod_{a \in ACT_r} (1 - \prod_{e \in P_{as}} A_e) \tag{5}$$

where P_{as} is the set of physical links on which the virtual link between the active VNF a and standby VNF s is mapped.

V. PROBLEM FORMULATION

We denote by N a set of cloud centers located at different geographical locations and by E a set of physical WAN links between cloud centers. Each cloud center n has an upper bound CA_n on computing resource and each physical link e has an upper bound BW_e on bandwidth resource. We denote a set of cluster requests by CR. We assume that each request will take the form of one of the three redundancy models described in the previous section. This assumption is

TABLE 1. Notations.

Notation	Description
N	Set of cloud centers
E	Set of WAN links between cloud centers
CR	Set of cluster requests
ACT_r	Set of active VNFs in cluster request r
STB_r	Set of standby VNFs in cluster request r
NU_r	The total number of VNFs in cluster request r
VL_r	Set of virtual links between VNFs in request r
RC_a	Computing resource demand for active VNF a
RC_s	Computing resource demand for standby VNF s
RB_l	Bandwidth resource demand for virtual link l
CA_n	Computing resource capacity of cloud center n
BW_e	Bandwidth resource capacity of physical link e
A_n	Availability value of cloud center n
A_e	Availability value of physical link e
A_r	Availability of cluster request deployment r
AL_s	Availability of virtual link to standby function s
BC_e	Bandwidth consumption of physical link e
CC_n	Computing consumption of cloud center n
X_n^a	Binary value, equal 1 if active VNF a placed on center n
Y_n^s	Binary value, equal 1 if standby VNF s placed on center n
U_e^l	Binary value, equal 1 if virtual link l mapped to physical link e

adopted owing to the fact that every redundancy model can be decomposed into these three models. A cluster request r in CR is denoted by $r(ACT_r, STB_r, RC, RB)$ (this notation is explained in the previous section). The placement and routing problem is defined as follows:

Definition: For a cluster request r in CR, the placement problem consists of finding the locations for active and standby VNFs over geo-distributed cloud centers. The routing problem is that of finding a set of physical links for mapping the virtual links in the cluster request. The goal is to minimize the network bandwidth consumption while still satisfying the availability requirements.

- No cloud center can exceed its computing resource capacity.
- Exist a set of physical links connecting two VNFs in the cluster request, with sufficient bandwidth for state transfer.
- The placement should satisfy the minimum availability requirements and deployment policy.

To solve our problem, we propose an Integer Nonlinear Programing (INLP) model. The notation of parameters and variables are shown in Table 1.

Decision variables:

X_n^a: a binary variable, it is equal 1 if the active VNF a is placed on the cloud center n, and 0 otherwise, where $a \in ACT_r, r \in CR, n \in N$

Y_n^s: a binary variable, it is equal 1 if the standby VNF s is placed on the cloud center n, and 0 otherwise, where $s \in STB_r, r \in CR, n \in N$

U_e^l: a binary variable, it is equal 1 if the virtual link l is mapped to physical link e, and 0 otherwise, where $l \in VL_r$, $r \in CR, e \in E$

Objective:

$$Minimize \sum_{e \in E} BC_e \tag{6}$$

Computing resource constraint:

$$\forall n \in N: \quad CC_n = \sum_{r \in CR} \Big(\sum_{a \in ACT_r} X_n^a RC_a + \sum_{s \in STB_r} Y_n^s RC_s \Big) \leq CA_n \tag{7}$$

Link bandwidth constraint:

$$\forall e \in E: \quad BC_e = \sum_{r \in CR} \sum_{l \in VL_r} U_e^l RB_l \leq BW_e \tag{8}$$

Availability constraint:

$$\forall r \in CR: \quad A_r \geq A_r^{min} \tag{9}$$

Deployment policy constraint:

$$\forall r \in CR, n \in N: \quad \sum_{a \in ACT_r} X_n^a + \sum_{s \in STB_r} Y_n^s < NU_r \tag{10}$$

The constraint 7 guarantees that the total computing consumption of VNFs on one cloud center does not exceed the its capacity. The constraint 8 guarantees that the bandwidth consumption of virtual links is not over its capacity. The constraint 9 guarantees that after deployment all cluster requests satisfy their minimum availability requirement. The constraint 10 guarantees that the VNFs in one cluster have to be deployed on at least two different centers. This constraint can make the cluster deployment more resilient to natural disasters.

VI. HEURISTIC ALGORITHMS

In this section, we propose two different heuristic approaches to our problem. The first is based on popular greedy strategies combined with a reliable placement algorithm. The second is topology-aware heuristic (TARE) approach which attempts to reliably place the cluster requests over geo-distributed cloud centers in a resource-efficient manner.

A. GREEDY RELIABLE HEURISTICS

The key concept of greedy approach is to place VNFs in a cluster request across the cloud centers following one of policies below.

- **Best bandwidth resources**: The cloud center has the highest bandwidth resources on all of the associated physical links.
- **Best availability**: The cloud center has the highest availability value.
- **Best computing resources**: the cloud center has the most computing resources.

In this manner, the VNFs tend to be deployed on the same cloud center. This can minimize the bandwidth consumption, because there is no need for WAN links between cloud centers. However, if all VNFs are deployed on the same cloud center, the availability of cluster request will be decreased,

and may not satisfy the minimum requirement. As pointed out in [16], [20], the availability of a cluster request achieves high values when the VNFs in the cluster request are deployed on different cloud centers. Therefore, to satisfy the availability constraint, in these greedy approaches, we attempt to increase the distribution level of VNFs (i.e., the minimum number of different centers used to host VNFs) until the availability requirement is met. At the same time, for the routing problem, we employ a shortest path algorithm to map virtual links. This also helps increase the availability of the cluster request (i.e., shorter path means higher availability) and save the network bandwidth.

The greedy reliable (GR) heuristic is presented in detail in Algorithm 1. Depending on the chosen greedy policy, we have a corresponding algorithm. For example, if the greedy policy is bandwidth, we have the bandwidth greedy reliable algorithm (i.e., BWGR). If the greedy policy is availability, we have the availability greedy reliable algorithm (i.e., AVGR). If the greedy policy is computing resource, we have the computing greedy reliable algorithm (i.e., COGR).

- In Steps 3-4 of Algorithm 1, we first calculate the resource demands of the cluster request using the function *calResourceDemand*(). Then, we sort the cloud centers N according to one of the above greedy policies. We sort the requests in decreasing order of the request size, which is calculated by the number of VNFs in the request. In other words, we first place and map the largest requests on the cloud centers with the highest bandwidth, computing resources, or availability.
- In Steps 5-31 of Algorithm 1, the two variables H and *visitedCenters* are used to control the distribution level of VNFs across geo-distributed cloud centers. We can increase the value of H to increase the availability of the cluster deployment. We start with $H = 2$ to force the cluster request to deploy over at least two different centers. In Steps 9-20 of Algorithm 1, for each VNF in the request r we select one cloud center n according to one of the greedy policies above. Then, if the center n has sufficient computing resources and does not have any VNFs from the cluster request already placed, the center n will selected for placing the VNF. We repeat Steps 9-20 until all VNFs have been placed.
- In Steps 21-22 of Algorithm 1, the short-path routing algorithm, shown in Algorithm 2, is invoked to map virtual links of the cluster request on the physical network
- In Steps 23-29 of Algorithm 1, the availability of a request is calculated using the function *calculateAvai*(). This function implements the above availability calculation formula given in Section IV. If the placement results satisfy the availability requirement, then the actual placement and routing are executed, and the bandwidth and computing resources are decreased. If the placement results do not satisfy the availability requirement, then H is increased to enhance the availability of the request.

Algorithm 1 Greedy Reliable Algorithms

1: **Input**: N, E, CR, A, CA, BW
2: **Begin**:
3: $RC_a, RC_s, RB_l \leftarrow calResourceDemand()$
4: Sort CR by *requestSize* in a decreasing order
5: **for** $r \in CR$ **do**
6: **while** $H \leq size(r)$ **do**
7: $H \leftarrow 2$
8: $visitedCenters \leftarrow 0$
9: **while** $\exists$ VNF $\in r$ **do**
10: Get VNF a in request r
11: Select $n \in N$ according to one greedy policy
12: **if** $CA_n \geq RC_a$ and $n \notin visitedCenters$ **then**
13: Place VNF a on n
14: Update X, Y
15: Remove VNF a from request r
16: **if** $visitedCenters \leq H$ **then**
17: Add n into $visitedCenters$
18: **end if**
19: **end if**
20: **end while**
21: $virLinks \leftarrow VL_r$
22: $U \leftarrow$ **SP-R**$(X, Y, virLinks, RB)$
23: $A_r \leftarrow calculateAvai(r, X, Y, U)$
24: **if** $A_r \geq A_r^{min}$ **then**
25: Do actual placement and decrease
26: network and computing resources
27: **else**
28: Increase H
29: **end if**
30: **end while**
31: **end for**
32: **Finish**
33: **Output**: X, Y, U

We employ the shortest path algorithm, shown in Algorithm 2 to map virtual links in the cluster request. This algorithm attempts to find the set of physical links that are shortest while having sufficient bandwidth resources to replicate state between active and standby VNFs. This algorithm is based on the well-known Dijkstra algorithm to find the shortest path between two nodes in a graph. However, the algorithm is modified to able to find shortest path with sufficient bandwidth resources.

- In Steps 3-7 of Algorithm 2, for each virtual link l, we obtain two cloud centers m and n which host two VNFs of l. We initiate a graph with vertices N and edges E. If the bandwidth of edge is less than bandwidth demand of virtual link l, then the cost of edge is set to 1. Otherwise, it is set to ∞.
- In Steps 8-13 of Algorithm 2, we run the *Dijkstra* algorithm to obtain the shortest path between m and n. If the shortest path consists of all physical links with sufficient bandwidth, then the virtual link l can be mapped.

Algorithm 2 Short-Path Routing Algorithm (SP-R)

Input: X, Y, $virLinks$, RB
Begin:
for $l \in virLinks$ **do**
  Obtain m, n if $X_m^{l[0]} = 1$ and $X_n^{l[1]} = 1$
  Create a *Graph* with vertices, edges N, E
  Set cost of edge to 1 if $BW_e \geq RB_l$
  Set cost of edge to ∞ if $BW_e \leq RB_l$
  $P(n, m) \leftarrow Graph.Dijkstra(n, m)$
  if All links in $P(n, m)$ have cost 1 **then**
    l can be mapped on $P(n, m)$
    Update U, and decrease BW_e
  else
    Can not find path
  end if
end for
Finish
Output: U

B. TOPOLOGY-AWARE RESOURCE-EFFICIENT HEURISTIC

These greedy reliable algorithms are normally inefficient in terms of network resources. We observe that the topologies of different cluster requests can be classified into two types: star topology (e.g., mAoS and oAmS) and mesh topology (e.g., mA). Therefore, we devise an alternative heuristic solution, namely the topology-aware resource-efficient algorithm (TARE) which can exploit this observation in the placement and routing algorithm. The TARE algorithm is presented in Algorithm 3. In Algorithm 3, we also calculate resource demands for each request using the function *calResourceDemand*(). Then, based on the request topology, we call a corresponding algorithm: *PlaceStarTopo*() is for placing and routing cluster requests with a star topology and *PlaceMeshTopo*() is for placing and routing cluster requests with a mesh topology.

Algorithm 3 Topology-Aware Resource-Efficient Algorithm

Input: N, E, CR, A, CA, BW
Begin:
$RC_a, RC_s, RB_l \leftarrow calResourceDemand()$
Sort CR by *requestSize* in a decreasing order
for $r \in CR$ **do**
  Determine request as star or mesh topology
  if Star topology **then**
    $PlaceStarTopo(N, E, CR, A, CA, BW, RC, RB)$
  end if
  if Mesh topology **then**
    $PlaceMeshTopo(N, E, CR, A, CA, BW, RC, RB)$
  end if
end for
Finish
Output: X, Y, U

We observe that to minimize the total network bandwidth resources while still satisfying the availability requirements, the VNFs should be deployed on different cloud centers, and the total hop distance between the VNFs in the request should be as short as possible. Therefore, for requests with a star topology, we attempt to place the VNFs in a request using a shortest tree, as shown in Fig.3. In Fig. 3, a star request is comprised of a center VNF (e.g., the active VNF) and edge VNFs (e.g., standby VNFs). We first select the cloud center with the highest number of usable links. A usable link is defined as a direct physical link between two centers with sufficient bandwidth. From this cloud center, we construct a shortest tree with the root node being this cloud center. Each leaf node contains information, such as the cloud center name, availability value, and computing resources. The path from the root node to a leaf node is always the shortest path. As shown in Fig. 3, when we perform the placement for the request, the center VNF will be placed on the root node provided that it has sufficient computing resource. Then, the edge VNFs are placed on the leaf nodes of the shortest tree in turn. We attempt to place the edge VNFs of the request over each level of the shortest tree corresponding to $K = 1, 2, 3, \ldots$. For each level of the shortest tree, we prioritize the node with the highest availability. For each placement of one edge VNF, the virtual link connecting an edge VNF to the center VNF is also mapped to the set of physical links from the leaf node to the root node. Using this shortest tree, we can always check the bandwidth remaining on the reverse path to the root node. Therefore, we can determine whether the reverse path has sufficient bandwidth to carry traffic from the center VNF to edge VNFs.

Algorithm 4 details how to build a shortest tree.

- In Steps 3-7 of Algorithm 4, we setup two variables *allCenters* and *visitedCenters*, which are used to store all the cloud centers and visited cloud centers. Then, we create a tree node *rootNode*, which stores information on the initial cloud center *startCenter* and *shortestTree* is pointed to the *rootNode*. We add the *rootNode* to *currentNodes*.
- In Steps 8-23 of Algorithm 4, we obtain the cloud centers in *currentNodes* and place them into *visitedCenters*. In Steps 11-20, for each *node* in *currentNodes* we obtain the neighboring cloud centers of the center in *node*. If the neighboring center has not been yet visited, then we create a new child node and attach it to *node*. Then, we add this new child node into *newCurrentNodes*. In Steps 21-22, we update *currentNodes* to *newCurrentNodes*, and remove all *visitedCenters* from *allCenters*. We repeat Steps 8-23 until all cloud centers in *allCenters* have been visited.

Algorithm 5 presents detailed steps for placing and routing star requests.

- In Steps 3-4 of Algorithm 5, we sort cloud centers in decreasing order of the number of usable links, and obtain the center VNF and edge VNFs of the request.

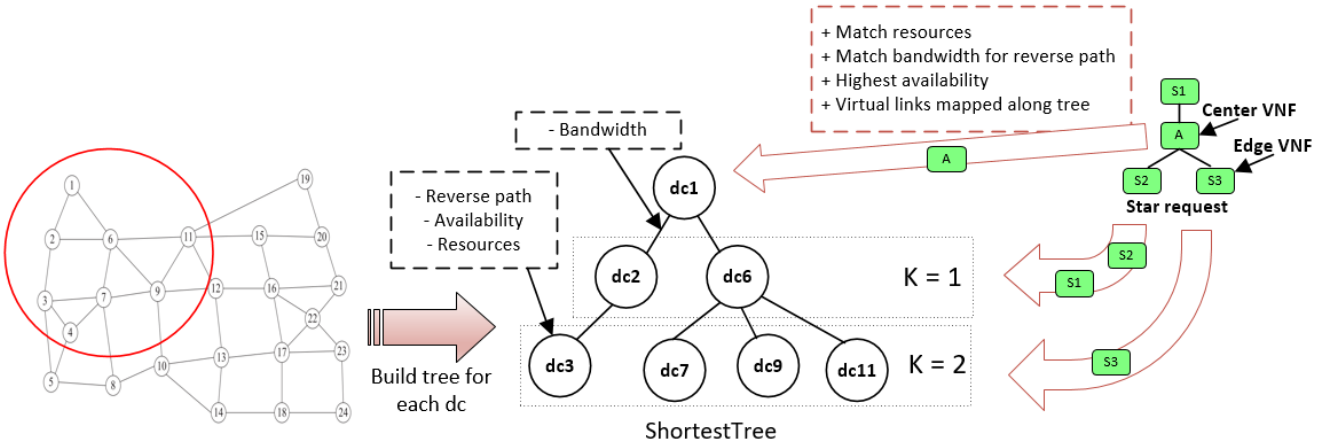


FIGURE 3. Placement and routing for star requests.

Algorithm 4 Build a Shortest Tree

```
Input: N, E, startCenter
Begin:
allCenters ← N
visitedCenters ← ∅
rootNode ← Node(startCenter)
shortestTree ← rootNode
Add rootNode into currentNodes
while allCenters do
    Add centers in currentNodes into visitedCenters
    newCurrentNodes ← ∅
    for node ∈ currentNodes do
        neighbors ← all neighbors of center in node
        for n ∈ neighbors do
            if n ∉ visitedCenters then
                childNode ← Node(n)
                node.addChild(childNode)
                newCurrentNodes ← childNode
            end if
        end for
    end for
    currentNodes ← newCurrentNodes
    Remove visitedCenters from allCenters
end while
Finish
Output: shortestTree
```

- In Steps 7-26 of Algorithm 5, we select the cloud center with the highest number of usable links to place the center VNF and build a shortest tree *shortestTree* using Algorithm 4. Then, we browse each node in the *shortestTree* and check computing constraints for each node on which we place an edge VNF and bandwidth constraints for the reverse path to map a virtual link. We browse nodes in *shortestTree* from the lowest to the highest level. At each level, we prioritize nodes with the high availability A_p first. A_p is calculated as the product of hardware server availability value and center availability value.
- In Steps 28-32 of Algorithm 5, after completing the placement and routing, the availability for the request will be calculated. If the availability requirement is satisfied, then we can complete the placement for the request. Otherwise, we attempt to perform another round of placement with a different cloud center for the center VNF.

For requests with a mesh topology, we attempt to select the best combination of cloud centers to place our requests. The best combination is defined as a group of cloud centers that has the same size as the request (i.e., the number of VNFs in the request) and has the most usable links (i.e., direct links with sufficient bandwidth). In Fig. 4, a request with four active VNFs is placed on four cloud centers that make up a best-matched combination with the most usable physical links.

The algorithm for placing and routing mesh requests is presented in Algorithm 6.

- In Steps 3-7 of Algorithm 6, we attempt to obtain the size of request, the peer VNFs, and a list of combinations of cloud centers. The size of each combination is equal the size of the request. We remove combinations without sufficient computing resources, and sort the remainder by the number of usable links in decreasing order.
- In Steps 9-12 of Algorithm 6, we place peer VNFs on each combination, and call the shortest-path algorithm to map the virtual links on the physical topology.
- In Steps 13-18 of Algorithm 6, similarly to the star request case, the availability of the request is calculated and checked whether the availability requirement is satisfied. If so, we can complete the placement. Otherwise, we perform the placement and routing with another combination.

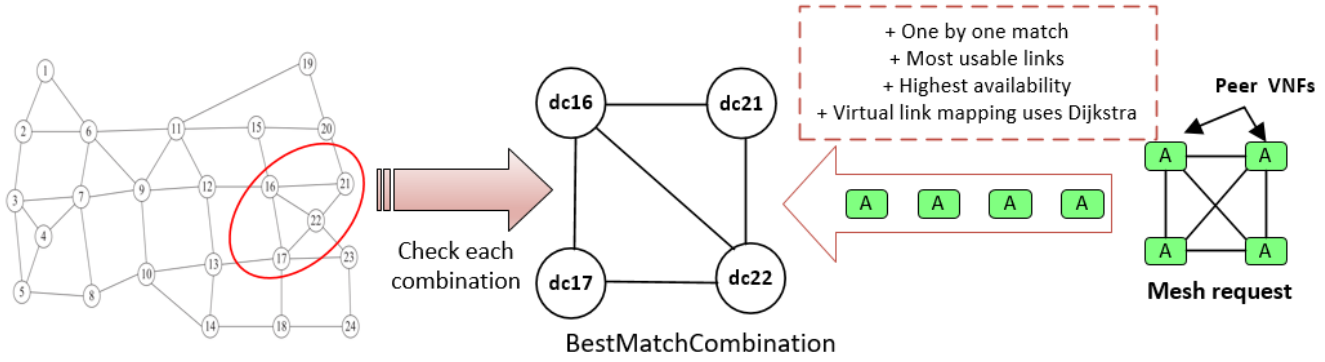


FIGURE 4. Placement and routing for mesh requests.

Algorithm 5 Placement and Routing for Star Requests

Input: $N, E, CR, A, CA, BW, RC, RB$
Begin:
Sort N by *usableLinkNum* in a decreasing order
Obtain *centerVNF* and *edgeVNFs*
for $n \in N$ **do**
  if $CA_n \geq RC_{centerVNF}$ **then**
    Place *centerVNF* on n
    $placedNodes \leftarrow n$
    $shortestTree \leftarrow$ Algorithm 4
    $K = 1$
    while $K \leq shortestTree.maxHeight$ **do**
      $treeNodes \leftarrow getNodesLevelK()$
      Sort *treeNodes* by A_p in decreasing order
      Select each *treeNode* from *treeNodes*
      $l \leftarrow virtualLink(centerVNF, edgeVNF)$
      $revPath \leftarrow getreversePath(treeNode)$
      if $CA_{treeNode} \geq RC_{edgeVNF}$ and $BW_{revPath} \geq RB_l$ **then**
        Update X, Y, U
        $placedNodes \leftarrow treeNode$
      end if
      if $\exists edgeVNF$ **then**
        Increase K
      else
        $reqPlaced \leftarrow True$ and break
      end if
    end while
    if *reqPlaced* **then**
      if $A_r \geq A_r^{min}$ **then**
        Do actual placement and decrease
        network and computing resources
        break
      end if
    else
      continue
    end if
  end if
end for
Finish
Output: X, Y, U

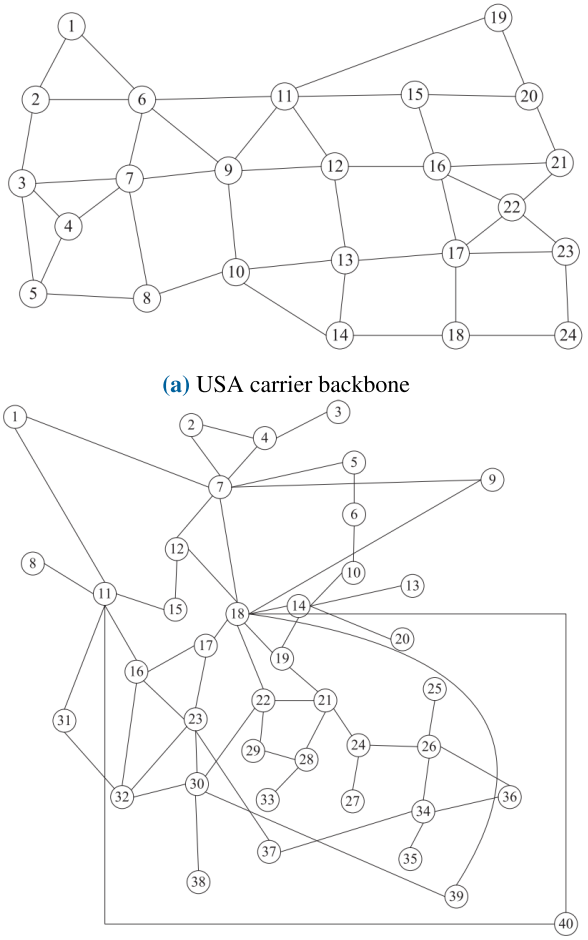
Algorithm 6 Placement and Routing for Mesh Requests

Input: $N, E, CR, A_n, A_e, CA_n, BW_e, RC, RB$
Begin:
$reqSize \leftarrow len(ACT_r + STB_r)$
Obtain *peerVNFs*
$combList \leftarrow getCombination(reqSize, N)$
Remove all *comb* in *combList* if not satisfy resource constraints
Sort *combList* by *usableLinkNum* in decreasing order
for $comb \in combList$ **do**
  Place *peerVNFs* on *comb*
  Update X, Y
  $virLinks \leftarrow VL_r$
  $U \leftarrow$ **SP-R**$(X, Y, virLinks, RB)$
  if $A_r \geq A_r^{min}$ **then**
    Do actual placement and decrease
    network and computing resources
    break
  else
    continue
  end if
end for
Finish
Output: X, Y, U

VII. EVALUATION AND NUMERICAL RESULTS

A. SIMULATION SCENARIOS

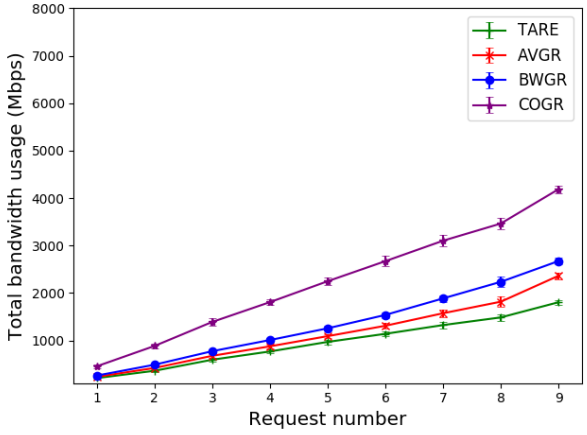
We verify our proposed algorithms, including the three greedy reliable algorithms (BWGR, AVGR, and COGR) and the topology-aware algorithm (TARE) using two networks: the USA carrier backbone and the GEANT European research network, as displayed in Fig. 5. The simulation parameters are partially taken from the reference [18]. The simulation parameters of cluster requests are adjusted to guarantee that the resource demand is less than the resource capacity. By this way, our placement model can be feasible. The simulation parameters are set as follows: the computing resource values CA_n are taken from the set $\{1000, 1500, 2500, 5000\}$. The physical links have the bandwidth capacities BW_e from the set $\{3000, 5000, 10000\}$.



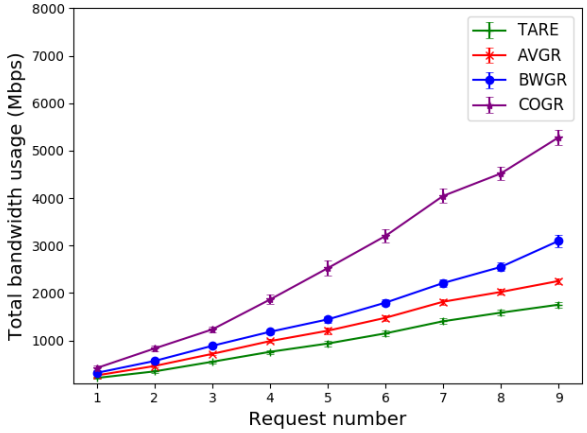
(a) USA carrier backbone

(b) GEANT European research network

FIGURE 5. USA carrier backbone and GEANT European networks.



(a) USA Network



(b) GEANT Network

FIGURE 6. Bandwidth Consumption with USA and GEANT Networks on request number.

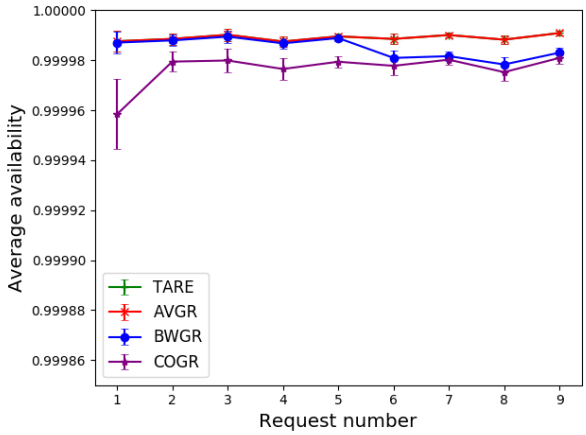
The availability values A_n and A_e of cloud centers and physical links are chosen randomly among $\{0.99, 0.999\}$. The availability of VNF software component A_f and hardware server A_d is set to 0.99. Because multiple cluster requests have different minimum availability levels, for the sake of simplicity, we set the minimum availability for each request to 0.95. This minimum availability value is adjusted to be satisfied by the cluster request with the least number of VNFs. For each cluster request r, the number of standby VNFs in STB_r and active VNFs in ACT_r are randomly selected in the set $\{2, 3, 4, 5\}$. The unit bandwidth resource demand RB is set to 10($Mbps$). The unit computing resource demand RC is set to 10 units. We evaluate our algorithms in terms of the following performance metrics.

- **Average availability**: measured as the average availability of all cluster requests after placement.
- **Total bandwidth usage**: measured as the sum of the bandwidth consumption for state replication on all WAN links.

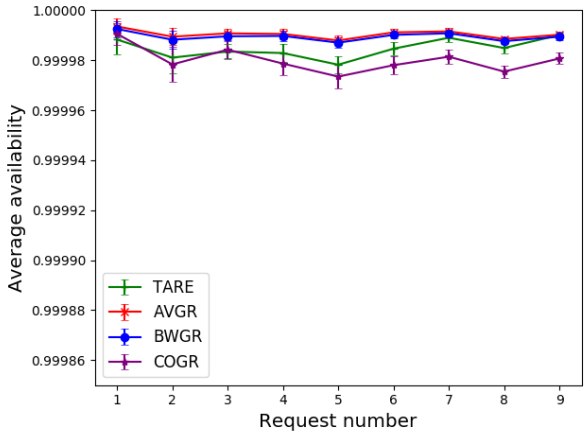
We develop two simulation scenarios. First, we vary the number of cluster requests, which includes all types of redundancy models, to compare the proposed algorithms when the number of cluster requests increases. Second, we create a set of cluster requests that only includes one type of redundancy model, to figure out the impact of each type of redundancy model on the overall performance. For active-standby model, both hot-standby mode and cold-standby mode are configured with the ratio of 1 to 2. For each case, we run 100 times and calculate the 95 percent confidence interval for all figures.

B. IMPACT OF NUMBER OF REQUESTS

Figs. 6a and 6b depict the bandwidth usage for the four algorithms as the number of requests increases. Regardless of the network topologies, the TARE approach always achieves the best performance. The main reason for this is that TARE is aware of the request topology, and can place the cluster requests on the cloud centers that are close to each other, and can form the shortest paths for communications between VNFs in the clusters. The COGR has the worst bandwidth usage, which results from two factors. The first is that the COGR focuses on the cloud centers with the highest computing resources, which results in the probability

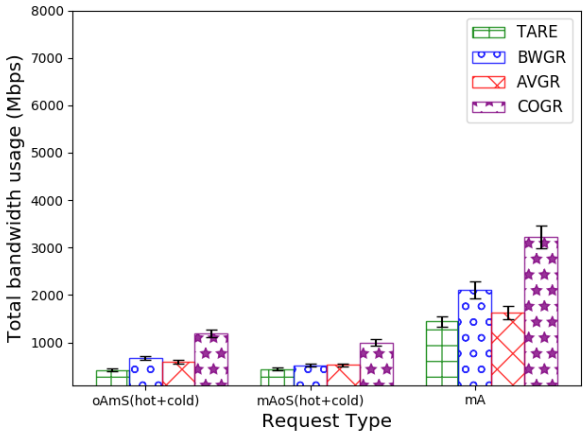


(a) USA Network

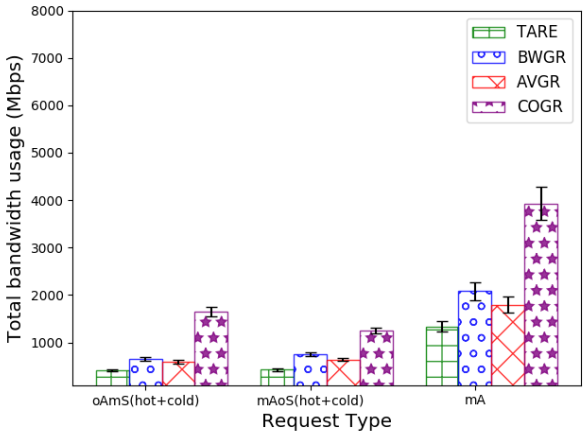


(b) GEANT Network

FIGURE 7. Average Availability with USA and GEANT Networks on request number.



(a) USA Network



(b) GEANT Network

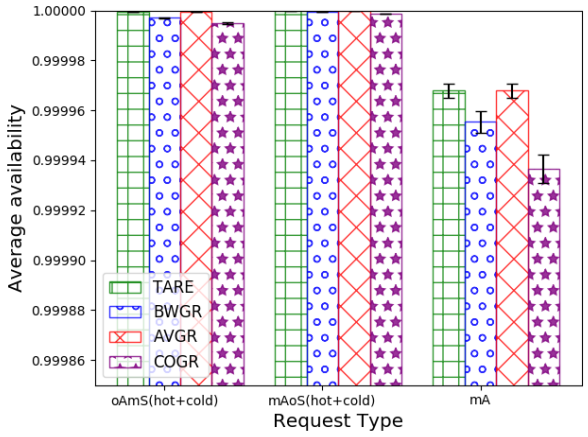
FIGURE 8. Bandwidth consumption with USA and GEANT Networks on the request type.

of not satisfying the minimum availability is higher than other approaches. Therefore, the VNFs in a request must be distributed over more cloud centers to satisfy availability requirement. At the same time, the cloud centers that are selected to host these VNFs are far from each other. This results in more physical links being required to carry the traffic, and the bandwidth usage also increases accordingly. The second factor is that the COGR does not consider the bandwidth resources of cloud centers when performing the placement. This reduces the probability of finding a shortest path between the placed VNFs. The performance of the BWGR and AVGR lies between the TARE and COGR, which results from the fact that each approach has its own feature for optimizing the bandwidth usage. The BWGR attempts to place the requests on the cloud centers with higher bandwidth resources, which leads to higher probability of finding shorter paths for state transfer between VNFs. The AVGR attempts to place the requests on the cloud centers with higher availability values, which makes the deployment of VNFs in one request become more centralized on one cloud center. Therefore, the AVGR can optimize the bandwidth usage of WAN links.

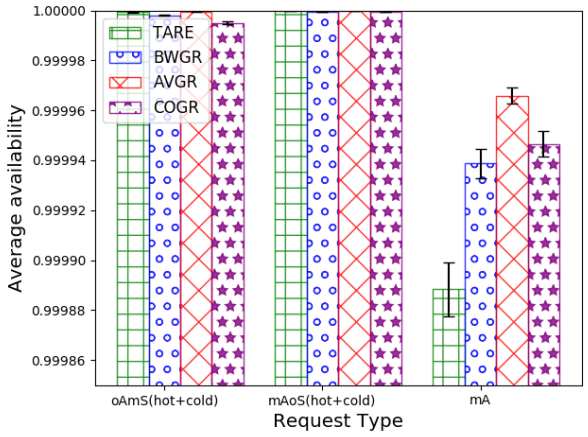
Figs. 7a and 7b depict the average availability of clusters as the number of requests increases. We can observe that all approaches achieve the minimum availability requirement. The TARE, AVGR, BWGR, and COGR have quite close performance with each other. In both topologies, the AVGR is a little higher and the COGR is a little lower than others. This is because the COGR does not employ any strategies that can improve the availability level of cluster requests during the placement process. In comparison with the COGR, the BWGR and TARE attempt to minimize the number of physical links used to map the virtual links of clusters. This strategy helps increase the availability level of cluster requests. In comparison with other approaches, the strategy of AVGR is to deploy VNFs on the cloud centers with the highest availability, which always makes the AVGR achieve the highest availability level.

C. IMPACT OF REQUEST TYPE

Figs. 8a and 8b depict the bandwidth usage in three cases, in which we perform simulations with one type of redundancy model at a time. Regardless of the cluster request type,



(a) USA Network



(b) GEANT Network

FIGURE 9. Average availability with USA and GEANT networks on the request type.

the TARE always achieves the best performance, and the COGR has the worst performance. The performance of the BWGR is slightly better than the AVGR. The performance gain is much clearer in the case of mesh cluster requests (i.e., mA) than for star requests (i.e., oAmS and mAoS). We can see that for the same physical topology, the bandwidth usage in the case with only mesh cluster requests is higher than for cases with only star cluster requests. This is because mesh cluster requests demand more bandwidth and computing resources than star requests. Therefore, more bandwidth resources are required to satisfy the resource demands in the case of mesh requests. In addition, the physical links run out of bandwidth resources more quickly when placing mesh cluster requests. Therefore, there is a lower probability of finding shorter paths between VNFs, which leads to higher bandwidth usage.

Figs. 9a and 9b depict the average availability as the request type varies. We observe that the TARE achieves the quite high level of average availability in most cases. In particular, for the cluster requests with a tree topology the TARE can achieve the near-optimal performance. For the cluster

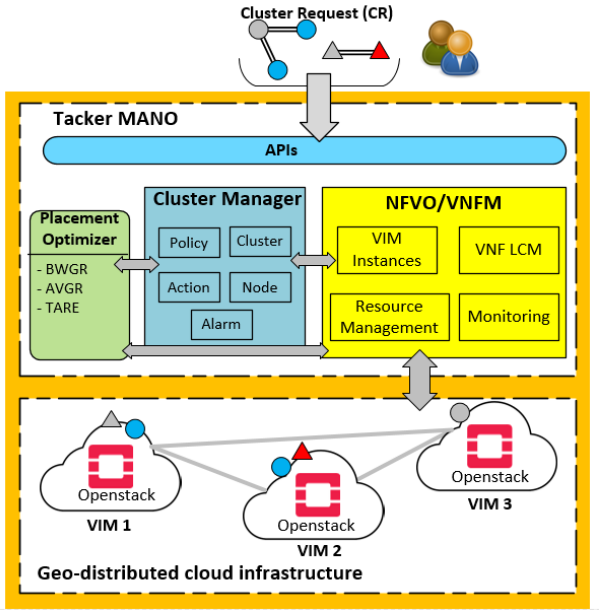


FIGURE 10. Prototype implementation.

requests with a mesh topology, the TARE does not achieve the near-optimal performance. However, the TARE still satisfies the availability requirement and is not the worst solution in terms of availability level.

VIII. PROTOTYPE IMPLEMENTATION

Figs. 10 describes the our envisaged prototype implementation. We employ opensource Openstack [25] software to deploy our cloud infrastructure. For the management and orchestration framework, we employ Openstack Tacker [26], which features basic VNF and network service management functions. These functions include VNF life cycle management (VNF LCM), virtual infrastructure management (VIM Instances), resource management, and monitoring functions. To enable Tacker to deploy and manage high availability clusters, we introduce two new components in the Tacker. The first component is Placement Optimizer, which runs our proposed algorithms (e.g., TARE). The placement optimizer receives the information (e.g., network topology and resource capacity from) from the monitoring and resource management functions, and output the placement results (e.g., VNFs and their locations). The second component is Cluster Manager, which receives the placement results and invokes the VNF lifecycle management-related APIs to deploy and configure these VNFs on the corresponding cloud centers. The Cluster Manager consists of five main objects: Cluster, Node, Action, Policy, and Alarm. The Node represents the VNF. The Cluster represents the cluster of VNFs and their roles (e.g., active or standby). The Action defines recovery actions, which will be executed when an trigger event occurs. The trigger event can be configured by the Alarm. The Policy that is attached to a cluster will define the corresponding Action, Alarm, and load balancing policy for the cluster.

IX. CONCLUSIONS

In this paper, we first studied the placement problem for high availability clusters with different redundancy models over geo-distributed cloud infrastructures. These redundancy models have different requirements in terms of bandwidth and computing resource demands and the minimum availability. We formulated our problem as an INLP model, and proposed several reliable greedy algorithms and a TARE heuristic algorithm to efficiently solve our problem. We performed simulations on two different physical network topologies with the variation of request number and request type. We observed that the TARE could achieve a better performance in terms of bandwidth resource consumption while maintaining the availability at an acceptable level.

REFERENCES

[1] *Network Functions Virtualisation (NFV); Virtual Network Functions Architecture*, Standard ETSI GS NFV-SWA 001 V1.1.1, 2014.
[2] *Network Function Virtualization (NFV), Management and Orchestration*, Standard ETSI GS NFV-MAN 001 v 1.1.1, ETSI, 2014.
[3] *Network Function Virtualization (NFV), Architecture Framework*, Standard ETSI GS NFV 002 v 1.1.1, ETSI, 2013.
[4] E. Vargas and S. BluePrints, "High availability fundamentals," *Sun Blueprints Ser.*, 2000.
[5] *Network Functions Virtualisation (NFV); Resiliency Requirements*, Standard ETSI GS NFV-REL 001 v1.1.1, ETSI, 2015.
[6] (Dec. 2017). *Opnfv Multisite*. [Online]. Available: https://wiki.opnfv.org/display/multisite/Multisite
[7] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. 10th Int. Conf. Netw. Service Manage. (CNSM) Workshop*, Nov. 2014, pp. 418–423.
[8] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 1346–1354.
[9] A. Basta, A. Blenk, K. Hoffmann, H. J. Morper, M. Hoffmann, and W. Kellerer, "Towards a cost optimal design for a 5G mobile core network based on SDN and NFV," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 4, pp. 1061–1075, Dec. 2017.
[10] T.-X. Do and Y. Kim, "State management function placement for service-based 5G mobile core architecture," *Mobile Netw. Appl.*, vol. 24, no. 2, pp. 504–516, Apr. 2019.
[11] L. Wang, Z. Lu, X. Wen, R. Knopp, and R. Gupta, "Joint optimization of service function chaining and resource allocation in network function virtualization," *IEEE Access*, vol. 4, pp. 8084–8094, 2016.
[12] M. Mechtri, C. Ghribi, and D. Zeghlache, "VNF placement and chaining in distributed cloud," in *Proc. IEEE 9th Int. Conf. Cloud Comput.*, Jun. 2016, pp. 376–383.
[13] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Comput. Commun.*, vol. 102, pp. 1–16, Apr. 2017.
[14] H. A. Alameddine, S. Ayoubi, and C. Assi, "Protection plan design for cloud tenants with bandwidth guarantees," in *Proc. 12th Int. Conf. Design Reliable Commun. Netw. (DRCN)*, Mar. 2016, pp. 115–122.
[15] R. S. Couto, S. Secci, M. E. M. Campista, and L. H. M. K. Costa, "Server placement with shared backups for disaster-resilient clouds," *Comput. Netw.*, vol. 93, pp. 423–434, Dec. 2015.
[16] H. Zhu and C. Huang, "Availability-aware mobile edge application placement in 5G networks," in *Proc. IEEE GLOBECOM*, Dec. 2017, pp. 1–6.
[17] Z. Yang, L. Liu, C. Qiao, S. Das, R. Ramesh, and A. Y. Du, "Availability-aware energy-efficient virtual machine placement," in *Proc. IEEE ICC*, Jun. 2015, pp. 5853–5858.
[18] S. Yang, P. Wieder, R. Yahyapour, S. Trajanovski, and X. Fu, "Reliable virtual machine placement and routing in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2965–2978, Oct. 2017.
[19] M. Casazza, P. Fouilhoux, M. Bouet, and S. Secci, "Securing virtual network function placement with high availability guarantees," in *Proc. IFIP Netw.*, Jun. 2017, pp. 1–9.
[20] T.-X. Do and Y. Kim, "Usage-aware protection plan for state management functions in service-based 5G core network," *IEEE Access*, vol. 6, pp. 36906–36915, 2018.
[21] L. Qu, C. Assi, K. Shaban, and M. Khabbaz, "Reliability-aware service provisioning in NFV-enabled enterprise datacenter networks," in *Proc. 12th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2016, pp. 153–159.
[22] W. He, X. Chen, X. Qiu, S. Guo, and P. Yu, "ASCO: An availability-aware service chain orchestration," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, Apr. 2019, pp. 590–593.
[23] *Network Functions Virtualisation (NFV); Report on Scalable Architectures for Reliability Management*, Standard ETSI GS NFV-REL 002 V1.1.1, ETSI, 2015.
[24] *Network Functions Virtualisation (NFV); Reliability; Report on Models and Features for End-to-End Reliability*, Standard ETSI GS NFV-REL 003 V1.1.1, ETSI, 2016.
[25] Openstack Foundation. *Openstack*. Accessed: Jun. 15. [Online]. Available: https://www.openstack.org/
[26] Openstack Foundation. *Tacker*. Accessed: Jun. 15. [Online]. Available: https://www.openstack.org/software/releases/stein/components/tacker

TRUONG-XUAN DO received the B.E. degree in electronics and telecommunication engineering from the Hanoi University of Science and Technology, Vietnam, in 2011, and the M.S. degree in information and telecommunication engineering from Soongsil University, South Korea, in 2013, where he is currently a Ph.D. Researcher with the Department of Information and Telecommunication Engineering. His current research interests include cloud networking, software-defined networks (SDN), network function virtualization (NFV), protocol design and network optimization for SDN, and NFV-based 5G mobile core networks. He also involves in development of open source tools for managing the Openstack cloud software.

YOUNGHAN KIM received the B.S. degree in electronics engineering from Seoul National University, Seoul, South Korea, in 1984, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Seoul, in 1986 and 1990, respectively. From 1987 to 1994, he was a Senior Member of the Research Staff with the Digicom Institute of Telemetics, Seoul, South Korea, where he did the research on data and computer communication systems. Since September 1994, he has been with Soongsil University, where he is currently a Professor with the School of Electronic Engineering. He is also the Director of the Internet Infrastructure System Technology Research Center (IISTC), supported by the MSIP (Ministry of Science, ICT & Future Planning), South Korea. His research interests include future 5G mobile networking, SDN (software-defined networking), network function virtualization (NFV), ICN (information centric networking), and sensor networking. He is a member of the IEICE.