Embedding Algorithm for Virtualizing Content-Centric Networks in a Shared Substrate

Shengquan Liao*, Xiaoyan Hong†, Chunming Wu*, Ming Jiang ‡
*College of Computer Science and Technology, Zhejiang University, P.R. China
†Department of Computer Science, University of Alabama, Tuscaloosa, AL 35487
‡Institute of Software and Intelligent Technology, Hangzhou Dianzi University, P.R. China
shengquan-liao@163.com, hxy@cs.ua.edu, wuchunming@zju.edu.cn, jmzju@163.com

***Abstract*—Network virtualization enables diversified network architectures to coexist in a substrate network. Content-centric network (CCN), as one of the major proposals for the future network, can be deployed in the virtualized environment. However, the recent work that has attempted in this direction has not addressed issues concerning the unique resource usage of CCN in resource allocations during the embedding procedure when instantiating a virtual CCN (VCCN). The unique problem is the cache component, i.e., the content store, in CCN. This paper will develop a VCCN Mapping Algorithm (VCCNMA) that optimizes the performances of VCCN considering the content store request in addition to CPU and bandwidth. Since the resource allocation problem is NP-hard, this paper thus proposes a heuristic algorithm. Several sets of simulation experiments are performed to evaluate the acceptance ratio, the network cost, the storage utilization ratio and load balance. Results show that the algorithm can achieve better performances. Moreover, the paper also studies the tradeoffs between the network cost and the load balance.**

Keywords - virtual network mapping; content-centric networking; network virtualization; distributed storage allocation; storage load balance

I. INTRODUCTION

Content-Centric Networking (CCN) [1] is one of the main clean-slate architecture for the future Internet [2]. Under this communication architecture, consumers broadcast *interest* packets and those who have the content chunks serve the *data* packets. During the transmission, routers can comprehend the content name and cache the content chunk in their content stores. Therefore, it is possible for any routers to satisfy an *interest* packet, if the content name matches a cached chunk, without further forwarding the *interest* packet to the data source. The CCN architecture has many benefits, for example, improving performances [3] and lowering network costs [4]. On the other hand, network virtualization allows different network services and experiments to coexist in a shared substrate network [5, 6 and 7]. It is regarded as a technology that is able to become a long-term solution for reconciling the co-existence issue for Today's Internet and multiple future network innovations [8]. As such, we believe that CCN, a popular future network proposal, can be embedded as a sliced instance in the virtualized environment. In addition, we also believe that embedding CCN is beneficial both for its real production uses and experimental validations. It can be expected that the large-scale network testbeds (e.g. GENI [6] and PlanetLab [9]) can be fully utilized to validate performances of innovations in CCN.

Prof. Chunming Wu is the corresponding author.

This paper is to address the resource allocation problem in the VCCN embedding procedure that instantiates a VCCN onto the substrate network. The major problem is how to allocate resources of the substrate network to support the request of slicing a VCCN request. Usually, CPU and bandwidth are considered in general virtual network embedding modules [10, 11 and 12]. However, CCN has its unique cache component, i.e., the content store [1]. No existing work has considered the storage demands in the embedding issue. Yet, as shown in this paper, such a consideration can lead to efficient algorithms that improve the performance of a sliced CCN.

In this paper, we will develop an efficient VCCN Mapping Algorithm (VCCNMA) to optimize the performance of CCN when running in the network virtualization environment by considering its unique content store allocation needs. The key idea we propose in VCCNMA is to distribute the requested storages along the routing paths in the substrate network, whilst assigning CPU and bandwidth resources in a greedy way. However, the storage allocation optimization problem is proven to be a Integer Linear Program (ILP) which is a typical **NP-hard** problem [11]. As such, we relax the ILP to a convex programming, and introduce a heuristic solution (*SAA*) for the distributed storage allocation issue. Several simulation experiments are then performed to evaluate VCCNMA in term of the acceptance ratio, the network cost, the storage utilization ratio and load balance. The results show our algorithm can achieve better performances.

In addition, the embedding algorithm needs to consider the tradeoff between the access cost and the balance of the storage load. The access cost can be lowered if more storage resources are assigned closer to the destination node. However, this strategy can threaten the storage load balance needed by the Internet Service Providers (ISPs). To tackle the conflict, we introduce a weight factor in the proposed heuristic algorithm. The desired trade-off between the load balance and the access profit can be achieved by adjusting the weight. Simulations are performed to show how the weight influences the tradeoff.

The rest of the paper is organized as follows: In Section II, we present the embedding model and problem formulation. Section III describes VCCNMA in detail. Section IV gives simulation results validating our proposed algorithm. Finally, a conclusion is presented in Section V.

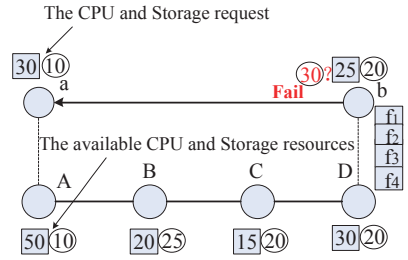
II. Consideration of Storage Allocations

A. Models for Virtual and Substrate Networks

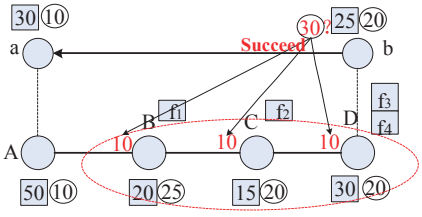
Current researches on embedding a virtual network to a substrate focus on improving the acceptance ratio and reducing the network cost, where constraints are usually the CPU capacity and bandwidth in virtual nodes and links. Hence, corresponding solutions are mostly heuristic so as to reduce the computational complexity [10, 11 and 12]. There are other works trying to implement CCNs in the virtualized environment [13, 14]. However, issues concerning the unique resource usage of CCNs during the embedding procedure, specifically, the storage resource for content store, remain open.

Here we present a virtual network to physical network (V2P) model that captures the VCCNs' mapping. The model includes adding the "storage" attribute for virtual/physical nodes and the direct routing constraint for virtual links in comparison to the general model in Ref. [10, 11 and 12]. The proposed core strategy is to distribute the requested storage to several physical nodes along the substrate path which carries a virtual link. Thus, we use a directed graph to denote the virtual network, and an undirected graph for the physical network. Under this model, the traditional undirected virtual link can be viewed as two overlaid directed ones. Further, we treat the forwarding behavior of *interest* packets to be bi-directional in the substrate network to ease the modeling of locating chunks in CCN. This treatment is reasonable because the *interest* packets are small and consume little bandwidth. Such a model has reduced complexities in the resource allocation, but it keeps CCN protocol behavior intact.

B. Benefit analysis of distributed storage allocation



(a) Traditional Virtual Network Embedding



(b) Virtual Network Embedding with Distributed Storage Allocation

Fig. 1: Mapping CCN using two virtual network models

The acceptance ratio is the main metric to gauge the successfulness of a mapping algorithm. It can be enhanced when the storage capacities of all nodes along the substrate path are cultivated for distributed storage allocations. In our model, the locations and the capacities of the storage nodes are both considered. Fig.1 depicts the CCN mapping under the traditional and proposed virtual network embedding model, respectively. The virtual network (VN) has only two nodes and one link ($b \rightarrow a$), and the physical network (PN) has four nodes on a line (D↔C↔B↔A). The number in the square box indicates the requested or available CPU resources on the virtual/physical nodes, and the number in round box indicates the requested or available storage resources. In this case, we assume the requested bandwidth of ($b \rightarrow a$) can be satisfied by the substrate network.

A traditional forwarding mapping creates a tunnel for the virtual link onto the substrate path. As shown in Fig.1.a, the virtual link ($b \rightarrow a$) is mapped onto the substrate path (D→C→B→A), where the bandwidth resources are reserved for data packets and the virtual node a and b are embedded onto the substrate A and D, respectively. If the requested storage of the virtual node b is 30, the embedding of b to D would fail because the substrate node D does not have enough storage resources to host b's need in the traditional way. In contrast, the requested storage of 30 can be satisfied in distributed storage allocations. As shown in Fig.1.b, the requested storage 30 is distributed to nodes {B, C, D}, and the virtual network embedding can succeeds. Hence, the distributed storage allocation has enhanced the VCCNs embedding acceptance ratio.

Another advantage of distributed storage allocations is that the routing efficiency would not be reduced in CCNs, rather, be often increased. Take b's requested storages being 20 and its cached chunks being $\{f_1, f_2, f_3, f_4\}$ as an example. In the traditional mapping, all the requested storage of b would be mapped onto D, and node A can get "f_1" from D in three hops (Fig.1.a). By the distributed storage allocation, "f_1" can be stored in B. Routing from D to B is in two hops, and A gets "f_1" from B in one hop. As a result, the total access cost is three hops which is the same as that in traditional mapping, when the access frequency F is 1.

More benefits can be achieved when F is bigger ($F \gg 1$). According to the current usage of CCN, it is reasonalbe to assume that F is much bigger than 1 because popular content chunks are cached in routers [13]. Thus, in Fig.1.a, if A requests content "f_1" from D F times, the total network cost is $3F$. However, in distributed storage allocation (Fig.1.b), the first visit has a cost of 3. For the rest *F-1* requests, only 1*(F-1) hops are needed because "f_1" has been cached in B after the first request. Thus, the total cost reduces to $F+2$. The cost reduction is 2(F-1). In general, with distributed storage allocations, the cost reduction is $\sum_{0 \le i \le length} ic_i(F-1)$ in average, where *length* is the total hops of the substrate path and c_i is the size of cache which is allocated to the ith node away from the source node on the substrate path.

C. Whose requested storage can be divided?

Here we analyze how to distribute the requested storages of virtual nodes. Although distributing requested storages of source nodes has potential to achieve higher access profits. Not all the virtual source nodes' requested storages could be partitioned. Further work of transformation of the input virtual network graph is needed. Using the term of graph theories, virtual source nodes could be classified according to their out-degrees. For instance, the set $\{a\}$ and $\{b,c,d\}$ in Fig.3.a contain

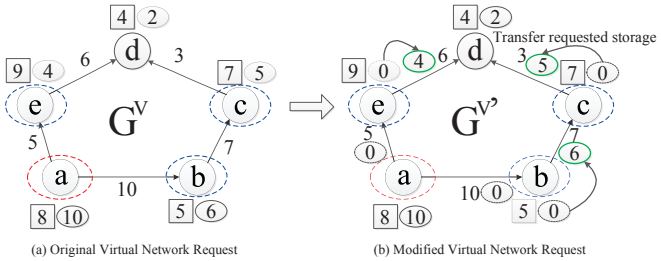


Fig. 2: Preprocessing virtual network inputs

nodes of out-degree of two and one, respectively. Fairness consideration speaks against find dividing and distributing the requested storage of virtual nodes with two or more out-degrees. As the example in Fig.2.a shows, if we assign the requested storage of *a* on the substrate path of the virtual link(*a*→*b*), the virtual node *c* will suffer unfair treatment, because more hops to access the cached contents in *b* would be needed. Hence, we suggest to distribute the requested storage of the source nodes whose out-degree is 1 in the virtual network.

With the above analysis, each input virtual network can be transformed to a new graph. The requested storage of the source nodes which meet the afore-mentioned requirements can be transferred to its adjacent virtual link and so the virtual link has the "storage" attribute. Hence, the original input (Fig.2.a) can be transformed to Fig.2.b.

D. Problem Formulation

It is noted that the storage distribution along the substrate path does not have to be uniform. There are cases that one wanting more cost reductions tends to assign more storage to substrate nodes that are closer to the destination. For example, if we transfer C's allocated storages (10 units) to B in Fig.1.b, 10**(F-1)* more benefits could be achieved under the average *F* accesses for each unit. Doing this would, however, jeopardize the storage load balance in the substrate network, because a balanced storage load distribution is needed to increase the acceptance ratio [10]. There is obviously a trade-off between the contradictory requirements and we found that VCCN mapping with Balancing the Access Profit and the storage Load Balance (VCCN-BAPLB) is equivalent to the traditional virtual network embedding problem, where the storage load and the access benefit from distributed storage allocations under any mapping can be viewed as 0 when the requested storage of all virtual nodes is 0. It has been proven that the virtual network embedding problem can be reduced to a **NP-hard** problem [15]. Therefore, VCCN-BAPLB is also **NP**.

III. Algorithm Design

In this section, we formulate our object which seeks for a trade-off between the access benefit and the storage load balance during the storage allocation. After the issue is proven to be NP, a heuristic algorithm (SAA) is proposed for the solution. VCCNMA starts with a greedy node embedding, and then uses the K-shortest algorithm to find substrate paths for virtual links, followed by distributing requested storage of virtual links to nodes along the substrate path with SAA.

A. Trade-off between the access benefit and the load balance

We denote the substrate network by an undirected graph G^S=(N^S, E^S, A_N^S, A_E^S), where N^S and E^S are the set of substrate nodes and links, respectively. Substrate nodes and links are associated with their attributes A_N^S and A_E^S, which accounts for CPU, storage and bandwidth. On the other hand, the virtual network is described by a directed graph G^V=(N^V, E^V, C_N^V, C_E^V), where N^V and E^V refer to the set of virtual nodes and links, respectively, and C_N^V and C_E^V are constraints for virtual nodes and links in terms of CPU, storage and bandwidth. The nodes in N^V which satisfy the requirements for distributed storage allocations are put into the set Φ. Thus, the VCCN embedding can be defined as the following mapping:

$$\boldsymbol{M}: G^V \longmapsto (N', P', R_N, R_L),$$

where $N' \subset N^S$, P' is a loop-free path in the substrate network. R_N and R_L are the resources of substrate nodes and links allocated to the virtual network G^V. The mapping is feasible when the following conditions are satisfied:

$$bw_{res}(p') \geq bw(e^v), \text{ for } e^v \longmapsto p' \quad (1)$$

$$cpu_{res}(n') \geq cpu(n^v), \text{ for } n^v \longmapsto n' \quad (2)$$

$$storage_{res}(n') \geq storage(n^v), \text{ for } n^v \longmapsto n', \text{ iff } n' \notin \Phi \quad (3.1)$$

$$\sum_{n' \in p'} storage_{res}(n') \geq storage(n^v), \text{ for } n^v \longmapsto n', \text{ iff } n' \in \Phi \quad (3.2)$$

n' and p' are elements in N' and P', respectively. n^v and e^v are virtual nodes and links in G^V. It means the requested bandwidth should be fulfilled in the substrate path. That also requires the requested CPU and storage resources of virtual nodes must be satisfied by the residual resources of its host nodes, if the virtual nodes' requested storages cannot be divisible. Otherwise, we distributed the requested storages of divisible nodes to the physical nodes along the substrate path.

If the requested storages of v_1 is divisible and the virtual link $\overrightarrow{v_1 v_2}$ is mapped onto the substrate path $\overrightarrow{p}$=(n_0^s, n_2^s, ..., n_t^s), we should distribute the requested storages of v_1 to its physical path $\{n_0^s, n_2^s, ..., n_t^s\}$ for maximizing access profits and minimizing load differences in the storage mapping.

Define 1: We use L= $\max\{\frac{\sum_{n^v \uparrow n^s} storage(n^v)}{storage(n^v)}\}$, $n^s \in \{n_0^s, n_1^s, ..., n_t^s\}$ to denote the storage's load balance on the substrate path. $\sum_{n^v \uparrow n^s} storage(n^v)$ is the sum of requested storages of virtual nodes mapped onto n^s. We can see that L is the largest storage utilization ratio among nodes on the substrate path.

Define 2: If the size of a content chunk is 1, the requested storage C_{total} of the virtual node v_1 is distributed to nodes $\{n_0^s, n_1^s, n_2^s, ..., n_t^s\}$ as the vector $\{c_0^s, c_1^s, c_2^s, ..., c_t^s\}$, where t is the length of the substrate path and c_i is the cache size allocated to the ith node away from the source node in the path. As is analyzed in Section II, we can define the access profits of distributed storage allocations as:

$$B = \sum_{0 \leq i \leq t} i \cdot c_i \cdot (F - 1) \quad (4)$$

The more storages are allocated for substrate nodes closer to the destination node, the more benefits could be achieved. However, it would break the load balance constraint. Hence, we introduce a integer linear program to describe this problem.

$$\textbf{objective}: \text{minimize } \alpha \cdot K \cdot L - B \quad (5)$$

s.t.

$$c_i \leq storage_{res}(n_i^s),\ 0 \leq \mathrm{i} \leq \mathrm{t} \quad (6)$$

$$\sum_{0 \leq i \leq t} c_i = C_{total} \quad (7)$$

$$c_i \in \mathbf{N},\ 0 \leq \mathrm{i} \leq \mathrm{t} \quad (8)$$

The objective (Eq.5) is to guarantee the storage's load balance (***Define 1***) of substrate nodes, whilst maximizing the access benefits (***Define 2***) in distributed storage allocations. Constraint (6) says that the substrate nodes must have enough capability to accept those allocated storages; while constraint (7) guarantees that the sum of distributed storage resources is equal to the total requested storage of the virtual source node. In Eq.8, the allocated storages for all nodes along the path should be a multiple of the content chunk size, assuming the size of a content chunk is 1.

The distributed storage allocation is a integer linear program which is a well-known NP-hard problem [11]. However, the constraint (8) could be relaxed as the following constraint:

$$c_i \geq 0 \quad (9)$$

After the relaxation, we name the new linear program as ***Storage_LP_RELAX***. It is based on a convex optimization as formulated in the following:

We suppose that the current storage load vector of all nodes along the substrate path is S=(s_0,s_1,s_2,...,s_t) and the solution vector is *C*=(c_0,c_1,c_2,...,c_t). If the storage capacity vector for these nodes is R=(r_0,r_1,r_2,...,r_t), our objective (Eq.5) could be rewritten as follows:

$F(C)=\alpha\cdot\mathrm{K}\cdot\max\{\frac{s_i+c_i}{r_i}\}$-$P\cdot C^T\cdot$(*F-1*), $i \in \{0, 1, 2, ..., \mathrm{t}\}$

where *P*=(0,1,2,...,t) is coefficient vector to calculate access benefits. Then we prove that our objective is convex.

To simplify our description, we replace $\max\{\frac{s_i+c_i}{r_i}\}$(i $\in$ $\{0,1,2, ..., \mathrm{t}\}$) with max $\{\frac{S+C}{R}\}$. Also, we assume the solution space of ***Storage_LP_RELAX*** is Θ. *X* and *Y* are the two solution vectors in Θ. If $\theta \in (0, 1)$ and $(1\text{-}\theta)\cdot X+\theta\cdot Y \in \Theta$:

$$\begin{aligned}
&\mathrm{F}((1\text{-}\theta)\cdot\mathrm{X}+\theta\cdot\mathrm{Y}) \\
&= \alpha\cdot\mathrm{K}\cdot\max\{\tfrac{S+(1-\theta)\cdot X+\theta\cdot Y}{R}\} + \mathrm{P}\cdot((1\text{-}\theta)\cdot X^T+\theta\cdot Y^T)\cdot(F\text{-}1) \\
&\leq \alpha\cdot\mathrm{K}\cdot\max\{\tfrac{2S+(1-\theta)\cdot X+\theta\cdot Y}{R}\} + \mathrm{P}\cdot((1\text{-}\theta)\cdot X^T)\cdot(F\text{-}1) \\
&\quad + \mathrm{P}\cdot(\theta\cdot Y^T)\cdot(F\text{-}1) \\
&\leq \alpha\cdot\mathrm{K}\cdot\max\{\tfrac{S+(1-\theta)\cdot X}{R}\} + \mathrm{P}\cdot((1\text{-}\theta)\cdot X^T)\cdot(F\text{-}1) \\
&\quad + \alpha\cdot\mathrm{K}\cdot\max\{\tfrac{S+\theta\cdot Y}{R}\} + \mathrm{P}\cdot(\theta\cdot Y^T)\cdot(F\text{-}1) \\
&\leq \mathrm{F}((1\text{-}\theta)\cdot\mathrm{X}) + \mathrm{F}(\theta\cdot\mathrm{Y})
\end{aligned}$$

Therefore, the objective function *F(C)* is convex, and this convex optimization (linear program) can be solved in polynomial time with the interior-point polynomial algorithm or ellipsoid algorithm [16].

B. VCCN Mapping Algorithm

When the requested storage of all virtual nodes is set as 0, the VCCN mapping issue is equal to the traditional virtual network embedding which is **NP**. Hence, we can conclude the VCCN mapping is a **NP** problem, and a heuristic algorithm (VCCNMA) is then proposed for the problem. Initially, we preprocess the input virtual network with discussions in Sec.II.C. After that, we give the following formula:

$$\mathrm{rank}(n^s) = \mathrm{storage}(n^s)\cdot\mathrm{cpu}(n^s)\cdot \sum_{l \in Neighbor(n^s)} bw(l) \quad (10)$$

to compute the weight for virtual/substrate nodes. Then, virtual nodes are mapped onto substrate nodes one to one according to their ranks in values calculated with formula (10), followed with the link mapping. In the link mapping phase, K-shortest-path algorithm is utilized to find substrate paths for those virtual links. Finally, we call ***SAA*** algorithm to distribute the requested storages of virtual links to all nodes along the path. If the link/node mapping and ***SAA*** succeed, we accept this virtual network.

In ***SAA*** (**Algorithm 1**), for each virtual link in the modified virtual network $G^{V'}$, we choose the shortest path which met the requested storages from its candidate path set to support the virtual link for the reason that it consumes least bandwidth resources. Then, we solve the relaxed linear program (***Storage_LP_RELAX***) so as to distribute requested storages to all nodes along the substrate path. Since the solution vector is real, we then apply the branch-and-cut algorithm to get the final integer assignment.

TABLE I: *P-code of SAA*

Algorithm 1: Storage Allocation Algorithm (SAA)

Input: the virtual network $G^{V'}$, and the path set P^S=$\bigcup_{1 \leq i \leq |E^{V'}|} P_i^s$. ($P_i^s$ is the K-path set including *k* candidate substrate paths for the virtual link e_i^v in $G^{V'}$)

Begin:

1. For each $e_i^{v'} \in G^{v'}$
2. Sort elements in P_i^s according to their lengths.
3. BOOL Flag = 0.
4. For each $p_i^s \in P_i^s$ // p_i^s=(n_0^s, n_1^s, n_2^s, ..., n_t^s)
5. if $\sum_{0 \leq i \leq t} storage_{res}(n_i^s) \geq storage(e_i^{v'})$
6. Flag = 1, p = p_i^s //Save the path
7. Break
8. end if
9. End for
10. if Flag = 0, then reject the request.
11. Solve ***Storage_LP_RELAX(storage($e_i^{v'}$), p)***
12. Apply the branch-and-bound search for the final integer assignment.
13. End for

End

C. Complexity Analysis

In VCCNMA, the time complexity of the node mapping is $O(N^2)$ for the greedy algorithm. The time complexity of the link mapping is $O(logN \cdot N^3)$ owe to the k-shortest-path algorithm's ($O(logN \cdot N + k \cdot N)$) ($k \ll logN$) and the maximal number of links $O(N^2)$. Also, the ***Storage_LP_RELAX*** could be solved in polynomial time by the ellipsoid algorithm or Karmarkars interior point algorithm [16]. Although the complexity of the branch-and-cut algorithm is exponential, we can limit the iteration number for a approximate solution in polynomial time.

IV. PERFORMANCE EVALUATION

The purpose of our evaluation is to validate that our heuristic algorithm can achieve better performances in the acceptance ratio, the access cost, the storage utilization ratio and load balance. The comparisons are made to a baseline algorithm and a random algorithm as no direct related work exists. Further, the simulations also show that the tradeoff between the load balance and the access profit can be controlled by a weight factor.

A. Simulation settings

In this paper, the simulation environment is the same as that in Ref.[12]. We use the brite tool [17] to generate a substrate

network with 100 nodes. Each pair of nodes in the network is connected with a link with a probability 0.5. The available CPU and storage resources for a physical node follow an uniform distribution between 50 and 100. Also, the bandwidth resource for a link is uniformly distributed in the range from 50 to 100.

In contrast, the number of VN nodes follows a unique distribution between 2 and 10. Also, each pair of virtual nodes is randomly connected with probability 0.5. The CPU, Storage and bandwidth for a virtual node/link are all uniquely distributed from 10 to 20. The virtual network requests (VNRs) arrive following a Poisson process with an average rate of 4 VNs per 10 time units. Each VNR's lifetime is 50 units. We run for about 5000 time units, which corresponds to 2000 VNRs on average in one instance of simulation.

We compare our algorithm with a baseline algorithm (BL-VNE) [12] and a random algorithm (RVNE) [18]. BL-VNE and RVNE both do not distribute the requested storages along the substrate path. The difference between them is that BL-VNE choses the host nodes according to their ranks in resources in the node mapping phase, while RVNE randomly select the mapping nodes.

B. Metrics

1) Acceptance ratio: We define our acceptance ratio as formula (11), which is the division of the number of accepted virtual networks and the total VNRs.

$$\textit{acceptance ratio} = \frac{\sum_{i=0}^{T} M(G_i^V)}{\sum_{i=0}^{T} G_i^V} \tag{11}$$

2) Average network cost: The total network cost is the sum of expenses to access those requested storages via the virtual link $e^{v'}$ with an average frequency $F^{e^{v'}}$. We can get the Average Network Cost (ANC) by dividing the total cost with the number of virtual links.

$$ANC_1 = \frac{\sum_{e^{v'} \in E^{v'}} length_{e^{v'}} \cdot \sum_{0 \le i \le length_{e^{v'}}} c_i^{e^{v'}}}{|E^{v'}|} \quad \text{iff: } F_{e^{v'}}=1 \tag{12}$$

$$ANC_{F_{e^{v'}}} = \frac{\sum_{e^{v'} \in E^{V'}} \sum_{0 \le i \le length} (F^{e^{v'}}-1)(length-i)c_i^{e^{v'}}}{|E^{v'}|} + AVC_1 \quad \text{iff: } F_{e^{v'}} > 1 \tag{13}$$

$c_i^{e^{v'}}$ is the requested storage of the virtual link $e^{v'}$ allocated for the ith node away from the source node on the path. *length* is the total hops of the substrate path of $e^{v'}$, and $F^{e^{v'}}$ is the average access frequency for $e^{v'}$'s requested storages.

3) Average storage utilization ratio and standard deviation of the storage resources in the substrate network: The average storage utilization ratio is a quotient of the sum of current substrate nodes's storage utilization ratio divided by the number of nodes. It is presented in formula (14):

$$\overline{U_{storage}} = \frac{\sum_{n^s \in N^S} \frac{\sum_{n^v \in N^V \wedge n^v \uparrow n^s} storage(n^v)}{storage(n^s)}}{|N^S|} \tag{14}$$

As for the standard deviation of the storage resources, we defined as formula (15):

$$U_\sigma = \sqrt{\frac{\sum_{n^s \in N^S} (\frac{\sum_{n^v \in N^V \wedge n^v \uparrow n^s} storage(n^v)}{storage(n^s)} - \overline{U_{storage}})^2}{|N^S|}} \tag{15}$$

C. Simulation results and analysis

Initially, α is assigned as 1, while K is set as 100 to balance the magnitude of the two values. After that, we solve the integer linear program with ILOG CPLEX [19] and get the statistical results in the acceptance ratio, the network cost, the average storage utilization and load balance (standard deviation). Then, we observe the relation between the access frequency and the access benefits. Finally, we discuss how α influences the storage load balance and the network cost.

Fig.3 shows VCCNMA outperforms BL-VNE (10%) and RVNE (20%) in term of the acceptance ratio. Without the distributed storage allocation, virtual nodes in BL-VNE and RVNE should find physical nodes that concurrently meet CPU and storage requests as their hosts. In contrast, virtual nodes only need to find those with enough CPU as their hosts in VCCNMA. Hence, the node mapping of VCCNMA is easier to succeed than that of BL-VNE and RVNE, which leads to a higher acceptance ratio. Correspondingly, this means more virtual network requests could be successfully mapped onto the substrate network with VCCNMA. Therefore, each substrate node would have a higher storage load in average, as has been confirmed in Fig.4.

Fig.5 describes comparisons of three algorithms on the standard deviation of the storage utilization. Fig.5 shows that VCCNMA can keep a higher performance in the storage load balance in substrate nodes. That is because VCCNMA breaks big storage requests into multiple caches. Thus, the differences in the storage utilization among substrate nodes are smaller than those of the other two schemes. Hence, the storage load is much more balanced in substrate nodes.

As is shown in Fig.6, the higher the average access frequency is, the more benefits the distributed storage allocation can gain. More specifically, the network cost is almost the same when the average access frequency (F) is 1. However, we can obtain more benefits when F changes from 2 to 6. If the requested storages are allocated closer to the clients, clients can access these content chunks via fewer hops. Obviously, the access benefits is proportional to the access frequency.

Fig.7 and Fig.8 show that α can adjust the tradeoff between the storage load balance and the network cost. When α becomes larger, optimizing the storage load balance becomes more important in our objective function. Hence, the standard deviation of the storage utilization would be smaller and smaller. As shown in Fig.7, when α changes from 0.1 to 5, the storage load is more and more balanced in the substrate nodes because its standard deviation is becoming smaller. In contrast, the benefit would decrease and the average network cost would increase, which has been confirmed in Fig.8 where the network cost increasingly becomes larger when α becomes bigger.

V. Conclusion

In this paper, we introduced the problem concerning the storage need when embedding CCNs onto the substrate network, and proved that the problem is **NP**. As such, we

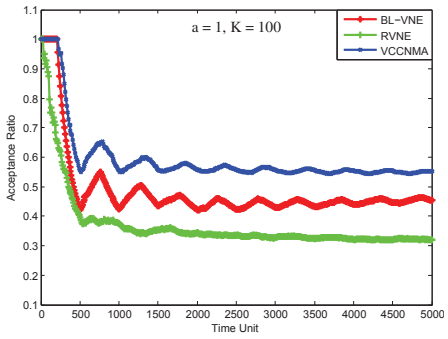


Fig. 3. Comparison on Acceptance Ratio

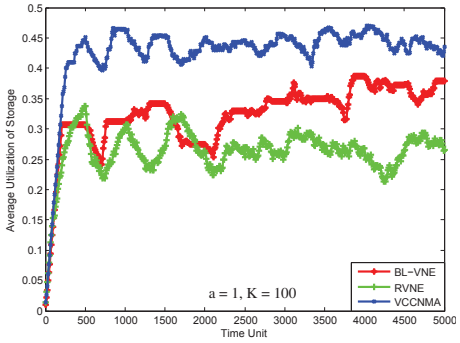


Fig. 4. Comparison on Average Storage Utilization

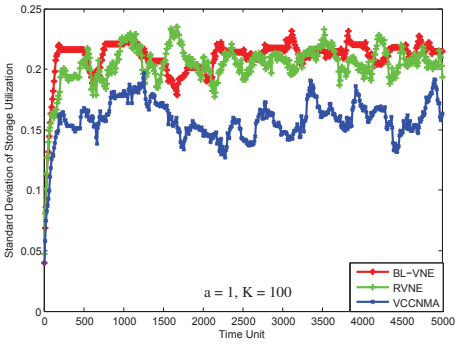


Fig. 5. Comparison on Standard Deviation of Storage Resources

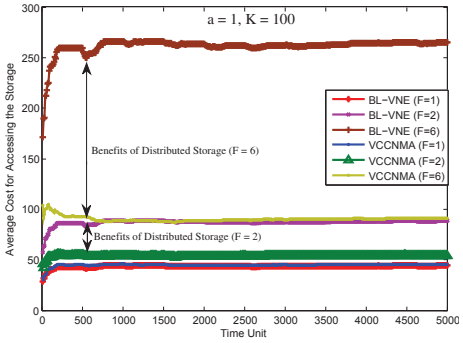


Fig. 6. Comparison on the average network cost with different F

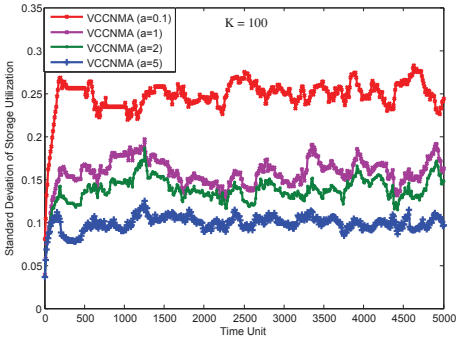


Fig. 7. Comparison on storage load balance with different α

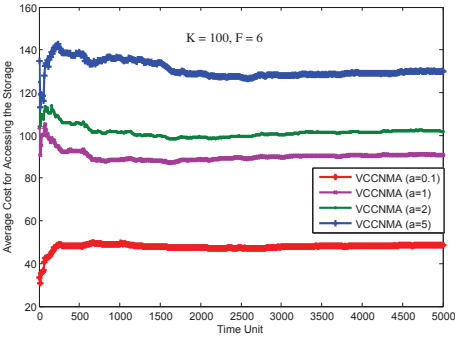


Fig. 8. Comparison on the average network cost with different α

developed a heuristic algorithm (VCCNMA) to solve for efficient embedding that yields better VCCN performances. Several simulation experiments have validated that VCCNMA can achieve higher performances in the acceptance ratio, the average storage utilization, the storage load balance and network cost. The results also show that we can control the tradeoff between the load balance and the access benefits with a weight factor.

ACKNOWLEDGMENT

This work is supported by the National Basic Research Program of China (2012CB315903), the Program for Zhejiang Leading Team of Science and Technology Innovation (2011R50010-05, 2013TD20), the National Science and Technology Support Program (2014BAH24F01), 863 Program of China (2015AA016103), and the National Natural Science Foundation of China (61379118).

REFERENCES

[1] Van Jacobson, Diana K. Smetters, et al. *Networking Named Content*, Proc. of ACM CoNext09, pp. 1-12, 2009.

[2] Pedro Henrique, V. Guimaraes, et al. *Experimenting Content-Centric Networks in the Future Internet Testbed Environment*, pp. 1383-1387.

[3] Psaras I, Clegg R G, et al. *Modeling and evalutaion of CCN-caching trees, in Networking 2011, Springer Berlin Heidelberg*, pp. 78-91, 2011.

[4] Tyson G, Kaune S, et al. *A trace-driven analysis of caching in content-centric networks*, Proc. of IEEE ICCCN 2012, pp. 107, 2012.

[5] Anderson T, Peterson L, et al. *Overcoming the Internet impasse through virtualization*, Computer, 2005(4): 34-41.

[6] GENI: www.geni.net

[7] Bavier A, Feamster N, Huang M, et al. *In VINI veritas: realistic and controlled network experimentation*, Proc. of ACM SIGCOMM Computer Communication Review, 36(4): 3-14, 2006.

[8] Nick Feamster, Lixin Gao, et al. *How to Lease the Internet in Your Spare Time*, in ACM SIGCOMM Computer Communication Review, 37(1): 61-64, 2007

[9] PlanetLab: www.planet-lab.org

[10] Zhu Y., Ammar M. H., *Algorithms for Assigning Substrate Network Resources to Virtual Network Components*, Proc. of IEEE INFOCOM 2006, pp. 1-12, 2006.

[11] Chowdhury N. M. M. K., Rahman M. R., et al. *Vritual Network embedding with coordinated node and link mapping*, Proc. of INFOCOM 2009, pp. 783-791, 2009.

[12] Minlan Yu, Yung Yi, et al., *Rethinking virtual network embedding: substrate support for path splitting and migration*, in ACM SIGCOMM Computer Communication Review, 38(2): 17-29, 2008.

[13] Masato Ohtani, Keiichiro Tsukamoto, et al. *VCCN: Virtual Content-Centric Networking for Realizing Group-Based Communication*. In Proc. of ICC 2013.

[14] S. Salsano, et al. *Information centric networking over SDN and OpenFlow: Architecture aspects and experiments on the OFELIA testbed*. Vol. 57, No. 16, pp. 3207-3221, Nov. 2013.

[15] D. G. Andersen. *Theoretical approaches to node assignment*, Unpublished Manuscript, 2002.

[16] A. Schrijver, *Theory of linear and integer programming*, John Wiley & Scons, 1986.

[17] BRITE : Topology Generator, [Online] Available: http:www.cs.bu.edubrite

[18] S. Zhang, et al. *Virtual Network Embedding with Substrate Support for Parallelization*, in Proc. of Globecom 2012, pp. 2615-2620, 2012.

[19] ILOG IBM., *Cplex optimization studio*, [online:] http:www-01.ibm.comsoftwarecommerceoptimizationcplex-optimizer