Scalable and Cost Efficient Algorithms for Virtual CDN Migration

Hatem Ibn-Khedher*, Makhlouf Hadji†, Emad Abd-Elrahman*§, Hossam Afifi* and Ahmed E. Kamal‡
*Institut Mines-Telecom (IMT), Telecom SudParis, Saclay, France.
§Computer and Systems Department, National Telecommunication Institute (NTI), Cairo, Egypt.
Email: {hatem.ibn_khedher, emad.abd_elrahman, hossam.afifi}@telecom-sudparis.eu
†Technological Research Institute - IRT SystemX, 8 avenue de la vauve, 91120, Palaiseau, France.
Email: see http://makhlouf.hadji.free.fr/
‡Department of Electrical Computer Engineering, Iowa State University, Ames, IA 50011-3060, USA.
Email: see http://www.ece.iastate.edu/ kamal/

***Abstract*—Virtual Content Delivery Network (vCDN) migration is necessary to optimize the use of resources and improve the performance of the overall SDN/NFV-based CDN function in terms of network operator cost reduction and high streaming quality. It requires intelligent and enticed joint SDN/NFV migration algorithms due to the evident huge amount of traffic to be delivered to end customers of the network. In this paper, two approaches for finding the optimal and near optimal path placement(s) and vCDN migration(s) are proposed (OPAC and HPAC). Moreover, several scenarios are considered to quantify the OPAC and HPAC behaviors and to compare their efficiency in terms of migration cost, migration time, vCDN replication number, and other cost factors. Then, they are implemented and evaluated under different network scales. Finally, the proposed algorithms are integrated in an SDN/NFV framework.**

***Index Terms*—vCDN; SDN/NFV Optimization; Migration Algorithms; Scalability Algorithms.**

I. Introduction

Software Defined Networks (SDN) and Network Function Virtualization (NFV) [1] are two paradigms which aim to virtualize certain network functions while adding more flexibility and increasing the overall network performance. ETSI has highlighted virtualized CDN (vCDN) among the major NFV use cases [2]. The main purpose of vCDN is to allow the operator to dynamically deploy on demand virtual cache nodes to deal with the massive growing amount of video traffic. Scaling in/out, caching as a service etc.. are also among the key benefits of vCDN.

The overall virtualization challenges for CDN transition to vCDN are addressed in the research project DVD2C [3]. In a previous work, we have investigated the main network issues in virtual machine migration and especially for vCDN use case [4] and in an SDN/NFV optimization context [5]. Fig. 1 depicts distributed vCDN nodes deployed by a centralized vCDN manager assisted with SDN/NFV controllers. Currently, despite the importance of optimization tasks, such functions are missing in the global architecture. We therefore, try in this paper to contribute with two algorithms and explain how they are integrated in the operators virtual LAN. The proposed optimization algorithms in this paper consider vCDN migration problem inside the network operator. Moreover, the major objective from both algorithms is to minimize the total cost of content migration while minimizing the additional extra-costs needed for caching, streaming, and replication number. Through the first optimization (i.e. OPAC: Optimal Placement Algorithm for virtual CDN), we are going to formulate an exact algorithm based on a mathematical model for deciding the optimal location to migrate a vCDN or to instantiate (place) a new vCDN on demand to satisfy users quality requirements. Further, to cope with scalability problems of exact algorithms, we adapt a heuristic algorithm (i.e. HPAC: Heuristic Placement Algorithm for virtual CDN) to deal with our constraints when large scale networks need to be optimized. In this algorithm, we exploit the well known Gomory-Hu method to find a near to optimum point of operation. Finally, as the optimization algorithms deal with many heteroclite parameters, a clear view on how/where/when they are extracted and how they are reflected on a typical SDN/NFV architecture is diagrammed.

The rest of this paper is organized as follows: Section II highlights the optimization algorithms in the virtualization context through the related work. Then, Section III details OPAC parameters, constraints and its objective function. Section IV details our heuristic optimization algorithm (HPAC). Section V evaluates the two algorithms and gives a comparison between

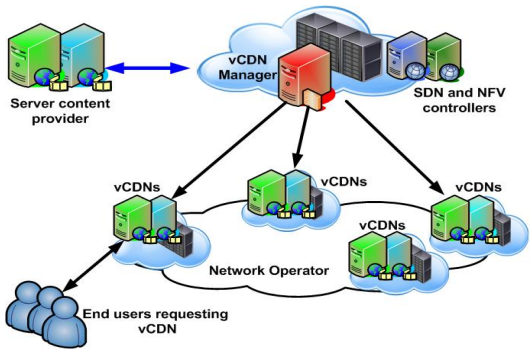


Figure 1: SDN/NFV-assisted CDN virtualization.

© 2016, Hatem Ibn-Khedher. Under license to IEEE.
DOI 10.1109/LCN.2016.23

them under predetermined metrics besides an integration diagram. Section VI concludes the work.

II. Related Work

This section highlights the relevant NFV placement approaches either exact or heuristic as follows: Niels et al. [6] provided a framework for multimedia delivery CDN-based on NFV. They used the three-layer structure of the network to seek the optimal locations of cache nodes. The optimization model lacks many constraints such as server's storage capacity.

Michele et al. [7] used a mixed architecture where real and virtual CDN nodes coexist. Although they present an interesting hybrid solution, their architecture considers only the placement.

Hendrick et al. [8] proposed a model for resource allocation of Virtual Network Functions (VNFs) within an NFV Infrastructure. Authors gave a hybrid solution inside the network operator that mixes the hardware and software-based network functions. Their model is a good abstraction of the deployment of NFV in an operator's network. But, it is still too general and does not address CDN specificity.

Bernardetta et al. [9] raised an important network flow problem related to compression/decompression processes applied on through-traffic passing by VNF instances. They propose a multi-level objective function for VNF placement optimization. Among the limitations of their approach is the utilization of a prioritization method to solve the problem which leads to sub-optimal results especially when the cost functions are orthogonal as in their case.

Mathieu et al. [10] proposed a placement problem optimisation for the Deep Packet Inspection (DPI) networks through designing a virtualized DPI (vDPI) solution. Authors aim to propose a model capable of minimizing network load and total number of deployed vDPI engines. This contribution is useful for our CDN placement problem. There are however major differences between the deployment of DPIs and CDNs. Moreover, authors do not introduce a migration scenario.

Mathieu et al. [11] tried in a second contribution, to solve the same aforementioned problem using heuristic algorithm which focuses on where to place the vDPI assisted with SDN paradigm. The proposed mechanism relies on a fitness function calculation to solve the problem.

Miloud et al. [12] introduced the placement problem of virtual PDN/S-GW in the mobile core network. They propose three heuristic algorithms to solve the NFV placement problem based on applications and services types in the virtual instance selection process.

Further, in most of the above exact/heuristic-based algorithms, the optimization of the virtual CDN migration is not considered and, to our knowledge, no contribution exploits the Gomory-Hu algorithm used in our solution as a scalable and a robust approach.

III. OPAC: Optimal Placement Algorithm for Virtual CDN

In this section, we specify the parameters and the constraints that are defined and proposed in formulating the optimization model OPAC. This formulation determines the migration of vCDN nodes to the optimal locations. We quote in Table I the main system and network parameters, and decision variables.

- *The decision variables*:
  1) The binary variable x_f^s indicates the placement of the streaming headend, and its migration from one server to another (best/optimal) location s. It is defined as:
  $$x_f^s = \begin{cases} 1 & \text{if } f \text{ migrates to } s \\ 0 & \text{Otherwise} \end{cases} \quad (1)$$
  2) The binary variable $y_{v,f}^s$ indicates a client v needs a vCDN service, and the server s caches it. It is defined as :
  $$y_{v,f}^s = \begin{cases} 1 & \text{if } v \text{ needs } f \text{ and } s \text{ caches } f \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$
  3) The binary variable $z_{i,j}^{v,f}$ indicates whether a link (i,j) is used (from i to j) to stream from a server s replicating f (the one for which $y_{v,f}^s = 1$) to client v.
- *The constraints*:
  1) The binary variable y should be less than or equal to x. In fact, y equals to 1, if and only if v needs f, and f is located on server s. This means that x should be equal to 1. Otherwise, if y equals to 0, then x may be equal to 0 or 1.
  $$\forall s \in S : y_{v,f}^s \leq x_f^s \quad (3)$$
  Note that a vCDN can be replicated more than once.

TABLE I: Mathematical Notation

Parameters	***Definition***
V	The set of client group nodes
S	The set of server nodes
D^s	Maximum throughput of the streaming server $s \in S$
F	The set of vCDN nodes
f_{size}	vCDN' s size $(vRAM, vCPU, vDISK)$ $(f \in F)$
C^s	Maximum storage capacity of the server s
$L_{i,j}$	Link capacity between two nodes i and j (from i to j)
d_v^f	Throughput used for streaming the functionality f to the client group $v \in V$. It represents the user's demand requirements
m_f^s	The migration cost of the functionality f to s
Decision variables	***Definition***
x_f^s	Placement and migration binary variable which indicates that f should move from origin server to optimal server s
$y_{v,f}^s$	Binary variable which indicates the video hit from node v of f in server s
$z_{i,j}^{v,f}$	Binary variable indicating whether the link (i,j) is used to stream f to v

2) Only one optimal server should serve the client node that requests the functionality f:

$$\forall v \in V \mid d_v^f \neq 0 : \sum_{s \in S} y_{v,f}^s = 1 \tag{4}$$

3) The cost of streaming f by a server s should be less than or equal to the maximum server capacity:

$$\forall s \in S : \sum_{v \in V} \sum_{f \in F} y_{v,f}^s \times d_v^f \leq D^s \tag{5}$$

4) The storage of the optimal server should not exceed its maximum capacity:

$$\forall s \in S : \sum_{f \in F} x_f^s \times f_{size} \leq C^s \tag{6}$$

5) Flow balance or conservation constraint between the server s and the client node v should be as the following:

$$\sum_j z_{i,j}^{v,f} - \sum_j z_{j,i}^{v,f} = \begin{cases} 0 & \text{if } i \neq v, i \neq s \\ y_{v,f}^s & \text{if } i = s \\ -1 & \text{if } i = v \end{cases} \tag{7}$$

This is the network flow constraint. The sum of incoming flows must be equal to the outgoing ones.

6) Link capacity between source i and sink j should be larger than the flow on the link:

$$\forall i, j \in V \cup S : \sum_{v \in V} \sum_{f \in F} z_{i,j}^{v,f} \times d_v^f \leq L_{i,j} \tag{8}$$

The objective function is formulated in equation (9) (cost function):

$$\min \sum_{s \in S} \sum_{f \in F} x_f^s \times m_f^s \tag{9}$$

where m_f^s is a parameter depending on the position of s, the position of s_f (the server initially containing f before migration), and the position of s_v (the server initially connecting to the client group v). m_f^s depends also on the size of f and the operator policy.

For the sake of clarity, we propose an example of OPAC vCDN migration as shown in Fig. 2 . The algorithm is as follows: 1) As an input, the algorithm gathers SDN/NFV architectural information such as the initial placement of CDNs/vCDNs and all necessary dynamic parameters, 2) It needs also a prediction of end-users demands to execute OPAC, and 3) as a result, end-users requesting vCDN YouTube®, to watch a popular video in a live event for example, will be redirected to the videos hosted in the new optimal calculated location.

Recall that the network operator hosting the vCDN of YouTube content provider migrates the vCDN cache/stream node to an optimal PoP where resources are available and content quality streamed are satisfied.

The above problem is NP-hard due to our combinatorial complex system and therefore the proposed exact algorithm is difficult to scale up to decide where to deploy vCDN nodes in a large scale scenario. As a consequence, an efficient heuristic algorithm is proposed in the next section.

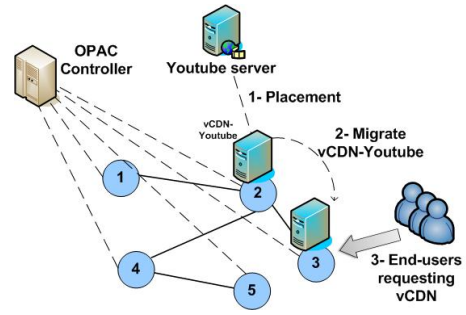


Figure 2: OPAC: vCDN-YouTube migration

IV. HPAC: Heuristic Placement Algorithm for Virtual CDN

Our proposal is strongly based on Gomory-Hu [1] tree transformation [13] of the initial network (represented by access, aggregate and core nodes) as shown in Fig. 5). In other words, HPAC transforms the input network into a G-H tree. Then, replicating vCDNs is performed thanks to the G-H tree allowing to efficiently reduce the number of edges to be considered when migrating contents.

Algorithm 1 summarizes the pseudo code of HPAC. Hereafter, we describe these main stages.

Algorithm 1 Heuristic algorithm for virtual CDN placement and migration

Input: V, S, D^s, F, f_{size}, C^s, $L_{i,j}$, d_v^f, m_f^s, $G = (V(G), E(G))$, s_v, s_f
Output: x_f^s, total migration cost
$GHT \leftarrow$ Gomory-Hu transformation $(G, L_{i,j})$
Tree-Exploration (GHT, s_v, s_f)
Migration-process$(D^s, C^s, L_{i,j}, f_{size}, d_v^f, m_f^s)$:
if $L_{i,j}, \leq d_v^f$ **then**
    Migrate-vCDN( )
end if

A. Gomory-Hu Transformation

At this stage, our heuristic HPAC computes the G-H tree of the CDN network or graph. The main advantage of G-H tree transformation is to compact this network graph structure using cuts to retain only feasible candidate topology and consequently lead to a smaller scale vCDN placement problem. We will insert a figure of the example of a G-H transformation.

For sake of clarity, we introduce the G-H tree transformation of our initial graph $G = (V(G); E(G))$. The G-H tree $GHT = (V(GHT); E(GHT))$ of the former graph can be built using the definition in [14]. There exists many algorithms that can build the G-H tree in polynomial times [15] by finding $N - 1$ maximum flow or minimum cuts between each pair of nodes in the graph, where $N = |V(G)|$. The G-H tree can be found

[1] noted by G-H in the rest of the paper

in polynomial time according to the algorithm used to find a maximum flow/minimum cut in the initial graph. But, we describe only one selected strategy that relies on the minimum Steiner cut algorithm presented in [16].

The transformation starts by initializing $V(GHT)$ to the set of the graph nodes (i.e., $\{V(G)\}$ and not $V(G)$). Concerning $E(GHT)$, it is initialized to an empty set. Besides, we define a queue list Q to enable the G-H tree construction. Q is initialized to $V(GHT)$ value. Then, while Q is not empty, we pull the first element S from this queue and we apply the minimum Steiner cut algorithm with the current set S (i.e., first Q element).

As the Steiner set in the new graph is obtained by contracting the entire sub-tree rooted at each neighbor in GHT of S into a single node. Consequently, two new sets of nodes S_1 and S_2 are generated from S and λ_{S_1,S_2} is the cut size. Accordingly, $V(GHT)$ should be updated by removing S and adding S_1 and S_2. Besides, $E(GHT)$ should be enhanced by adding a new edge between S_1 and S_2 with capacity λ_{S_1,S_2} (i.e., the cut size).

Furthermore, the queue Q will be enriched by adding S_1 (respectively S_2) if it includes at least two nodes, $|S_1| > 1$ (respectively $|S_2| > 1$). The former steps will be repeated until the G-H tree construction matches with an empty queue Q state.

The pseudo-code of the following G-H tree building strategy is summarized in Algorithm 2.

Algorithm 2 Gomory-Hu tree transformation algorithm

Input: A connected graph $G = (V(G), E(G))$
Output: A tree $GHT = (V(GHT), E(GHT))$
$V(GHT) = V(G)$, $E(GHT) = \emptyset$, $Q = \{V(G)\}$
while $Q \neq \emptyset$ **do**
  $S \leftarrow$ Pull(Q)://pull the first element from Q
  $\{S_1, S_2\} \leftarrow$ minimum-Steiner-Cut(S, GHT)
  $V(GHT) = \{V(GHT) \setminus S\} \cup \{S_1, S_2\}$
  $E(GHT) = E(GHT) \cup (S_1, S_2)$
  if $|S_1| > 1$ **then**
    5: $Q \leftarrow Q \cup S_1$
  end if
  if $|S_2| > 1$ **then**
    $Q \leftarrow Q \cup S_2$
  end if
end while

We give a simple example of a G-H tree transformation of a graph G as illustrated in Fig. 3. It is straightforward to see that the number of edges in the tree compared with the initial graph is reduced by 50%. Note that this G-H tree was built using only 6 iterations of Algorithm 2. The Gomory-Hu approach consists to reduce considerably the complexity of the problem by reducing the number of edges to be considered in our optimization. In addition, it allows quantifying the amount of flows that can be transferred between each couple of nodes in the graph (see Fig. 3). As a consequence, this method facilitates and eliminates a large number of non-feasible solutions before the optimization.

B. HPAC: Placement and Migration

Based on the G-H tree transformation, the number of nodes is equal to the number of nodes in the initial graph/network.

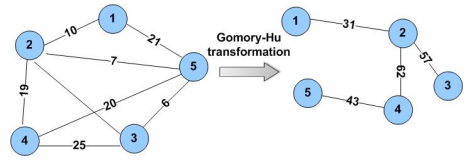


Figure 3: Example of a Gomory-Hu tree transformation.

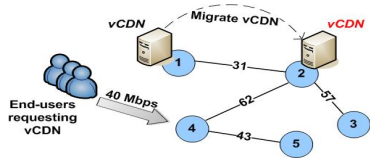


Figure 4: Example of HPAC vCDN content replication/migration.

However, the number of links in the derived G-H tree is at most equal to the number of nodes in the graph. Accordingly, the set of all paths can be easily computed. In fact, in the tree structure there is only one path between any couple of servers.

With this tree transformation leading to significantly reduced input sizes, the problem of vCDN placement and migration will be easily solved even for large problem instances (large number of nodes and arcs). This is due to the efficiency of the G-H tree transformation leading to reduce considerably the domain of feasible solutions. So, it allows us to find the near optimal solutions in few seconds. In the following, we give more details on the second stage of the HPAC algorithm to place and migrate vCDNs in a cost efficient manner.

The second stage of HPAC algorithm consists to explore the unique path from an access point to the server containing the vCDN. Thus, if the ingress flow cannot reach the destination (i.e. the vCDN), caused by a rupture node, then we will simply migrate the required vCDN to the rupture node of the G-H tree.

We propose the following example (see Fig. 4) to illustrate HPAC algorithm for migrating and placing judiciously vCDNs. It depicts a scenario of content replication/migration. In this scenario, a client cluster (group) representing grouped end-user requests for a vCDN with a specific throughput (i.e., content quality) equal to $40\ Mbps$. The proposed migration method searches to migrate the desired vCDN from its initial deployed position (server 1) to the suitable server (i.e., server 2 which has more than the required streaming capacity) in an efficient way (i.e., with minimum migration cost).

V. OPAC versus HPAC (Exact versus Heuristic)

For the interest of assessing the efficiency of OPAC and HPAC algorithms, we used [17] and [18] as optimization tools. In addition, different metrics/cost-factors can be defined as follows:

$$Migration\ cost = \sum_{s \in S} \sum_{f \in F} x_f^s \times m_f^s \tag{10}$$

$$Migration\ time = \sum_{s \in S} \sum_{f \in F} x_f^s \times f_{size} \times \max_{(i,j) \in P_{s_f,s}} \frac{1}{L_{i,j}} \tag{11}$$

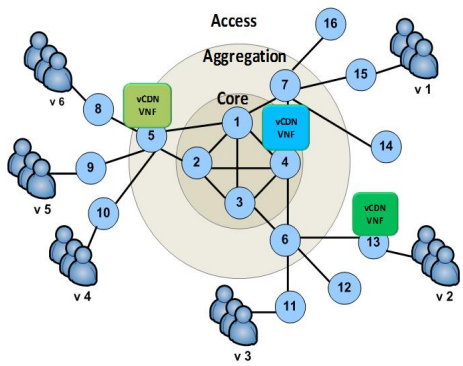


Figure 5: Network topology used for small scale.

Where $P_{s_f,s}$ is a given path from s_f to s. In fact, we are here assuming that the migration is done in a sequential way. If the migration is performed in parallel, then the migration time would be given by: $\max_{s\in S, f\in F}\left(x_f^s \times f_{size} \times \max_{(i,j)\in P_{s_f,s}} \frac{1}{L_{i,j}}\right)$. However, we will only consider (11).

$$Replica\ number = \sum_{s\in S\setminus\{s_f\}} \sum_{f\in F} x_f^s \tag{12}$$

$$vCache\ cost = \frac{\sum_{s\in S}\sum_{f\in F} x_f^s \times f_{size}}{\sum_{s\in S} C^s} \tag{13}$$

$$vStream\ cost = \frac{\sum_{v\in V}\sum_{f\in F} y_{v,f}^s \times d_v^f}{\sum_{s\in S} D^s} \tag{14}$$

Where the cost represents the amount of system resources (vCache) or network resources (vStream) consumed when migrating a vCDN from a server to another one.

Moreover, in order to decide which algorithm should we use and when, different network scales (i.e. small and large) and topologies are considered as follows:

A. Small Scale Scenario: a network operator snapshot

In this subsection, the optimization targets a small number of vCDNs. Therefore, a snapshot of a three-tier network operator architecture [2] is used for the evaluation as show in Fig. 5. Moreover, the main dissimilarities between the two approaches according to the defined metrics are showed hereafter:

Algorithm complexity: The algorithmic complexity of G-H-based HPAC is polynomial while it is exponential in OPAC. Table III illustrates and defines the computing complexity of the proposed algorithms OPAC/HPAC. It is clear that the exact formulation of the problem has an exponential number of feasible solutions defined by the convex hull of the discussed problem. To cope with scalability issues, we investigated a heuristic solution based on Gomory-Hu approach to reduce considerably the complexity problem as shown in Table III.

[2]NFVI PoP Network Topology (2016): https://tools.ietf.org/pdf/draft-bagnulo-nfvrg-topology-01.pdf

The run-time: The time convergence of the two algorithms is shown in the Fig. 6a . HPAC outperforms OPAC since the latter is a combinatorial-based algorithm and has an exponential complexity.

The total migration time: It is the duration needed to migrate a vCDN to the optimal/near optimal point of the deployment. Fig. 6b shows the average migration time needed for migration vCDN delivery functions to the optimal instantiation point taking into account the NFV constraints including system, network, and content quality parameters which are related to the vCDN functionalities. Although HPAC gives the shortest time as depicted in this figure, OPAC is still acceptable. Indeed, This metric is strongly related to the distance between servers as written in (11) and the time consumed for the vCDN migration process is less than $1minute$ for vCDN numbers equals 11.

The total migration cost: It is evaluated in both approaches (exact and heuristic). In Fig. 6c , the vCDN total migration cost is measured in terms of $Gigabits$ (Gb) against vCDN number. The Fig. 6c shows that vCDN-migration cost is increasing in both approaches for vCDN number ranging from 3 to 6. This is due to the non uniform client group distribution. It is noticeable that OPAC outperforms HPAC for vCDN ranging from 6 to 11. Nevertheless, HPAC proofs a low variation cost which leads to a better save of operator resources.

The total replication number: Under a random end-users matrix demand, the total replication number in both approaches is not significant (small) as shown in Fig. 6d. However, HPAC is the strictest approach when replicating vCDN in small scale scenario.

Other cost-factors: OPAC and HPAC are compared in terms of virtual cache (vCache) and virtual stream (vStream) cost-units [3]. In Fig. 6e cost unit is plotted against vCDN number. The virtual in-networking caching cost increases in both algorithms but they are still insignificant. Similarly, in Fig. 6f the virtual in-network streaming cost increases with vCDN number in both algorithms but it is still insignificant. Although the HPAC is slightly cost-efficient, both algorithms proofed an efficient network resource saving.

Experiments show that OPAC is still acceptable and saves about 96 % of the system resources and 94% of the network resources at $|F| = 11$. Nevertheless, it is still the most costly comparing to HPAC in terms of system and network resource usage.

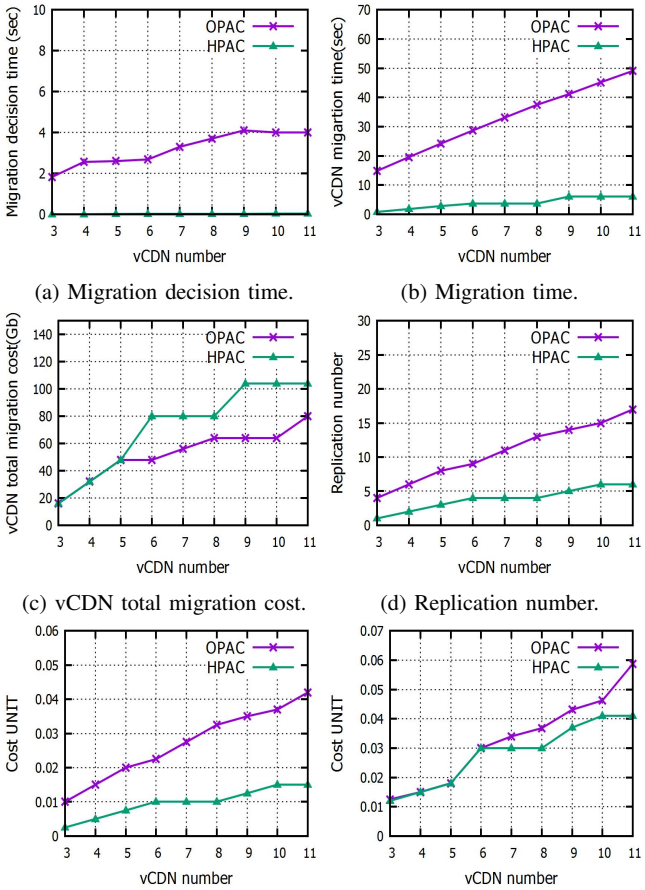
For the sake of clarifying the effectiveness of the HPAC, we define the Gap metric as follows:

$$Gap(C) = \frac{C_{HPAC} - C_{OPAC}}{C_{OPAC}} \tag{15}$$

Where C is a cost factor (e.g., migration cost).

Table II depicts the Gap metric (%) calculation formulated in (15) to estimate the difference in cost between OPAC and HPAC. It demonstrates the efficiency of the HPAC algorithm

[3]Cost-unit may be defined by the utilization percentage of the available system or network resource. It is the amount of resources (in GBytes) consumed when migrating content from a server to another one.



(a) Migration decision time. (b) Migration time.

(c) vCDN total migration cost. (d) Replication number.

(e) Virtual cache. (f) Virtual stream.

Figure 6: OPAC-HPAC comparison in the small network scale scenario.

TABLE II: Gap (migration cost): HPAC efficiency

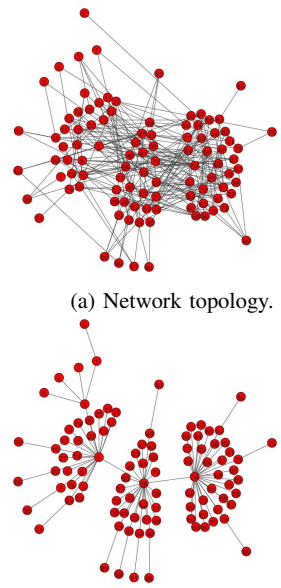
vCDN number	6	7	8	9	10	11
Gap (%)	0.66	0.42	0.25	0.62	0.62	0.3

in terms of total migration cost since the Gap values are close to zero. In other words, the objective function of our heuristic solution is always very close to the optimum. This justifies the efficiency of HPAC.

B. Large Scale Scenario: an Erdos-Renyi graph-based network operator

In this subsection, the optimization targets a large number of vCDNs (μ vCDNs/containers). The scenario relies on the well known Erdos-Renyi [4] undirected and weighted graph. The graph has 100 vertices (nodes) and 200 edges as shown in Fig.

[4]Erdos-Renyi graphs are random topologies that can be used to assess performance of our algorithms using random connectivity: one of real life scenarios. This topology is very similar to those generated by GT-ITM tool. Our objective in using this topology is to address well known random topologies to better test our solutions.



(a) Network topology.

(b) G-H-based transformation.

Figure 7: Network topology used for large scale.

7a. Its G-H-based transformation is shown in Fig. 7b which has only 99 edges (49.5%). These figures depict the topology used for evaluating HPAC algorithm in a large scale scenario. The OPAC algorithm could not be used here with reasonable resources.

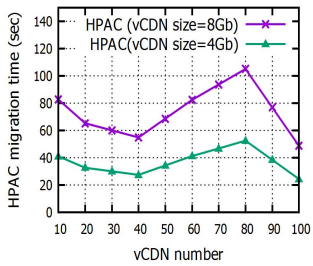
Fig. 8a and 8c depict the total migration delay time and the total migration cost respectively. Further, Fig. 8b shows the total replication number. These metrics are increasing until a decreasing slope at $|F| = 80$ representing the average vCDN number stabilizing the system. Furthermore, Fig. 9 shows the run-time of the OPAC and HPAC approaches in a large scale scenario. It demonstrates the feasibility/efficiency of HPAC since its run-time is in terms of a few seconds $(6\ sec)$ while OPAC run-time explodes from vCDN number equals 20.

C. Interpretations

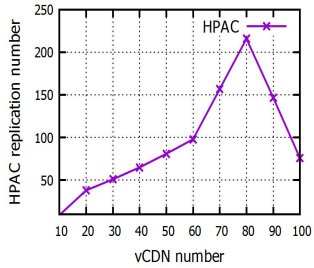
In this subsection, we quote the interpretations that can be revealed from the results above:

In small scale scenario: (the order of ten servers, ten client groups, and ten vCDNs), the exact approach is suggested to be used because its run-time is in terms of a few seconds (see Fig. 6a. OPAC is what we should choose as an optimization mechanism because it gives the lowest migration cost as shown in Fig. 6c .

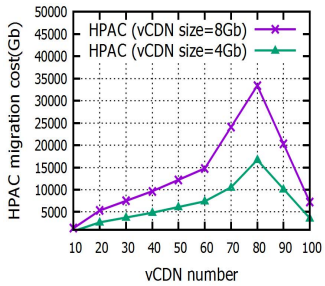
In large scale scenario: (the order of hundred servers, hundred client groups, and hundred vCDNs), the exact approach is suggested to be useful and the G-H-based HPAC algorithm is the alternative to solve the vCDN migration problem since its algorithm run-time is in terms of a few seconds as shown in Fig. 9 .



(a) Migration time/delay.



(b) Replication number.



(c) Migration cost.

Figure 8: HPAC in large network scale.

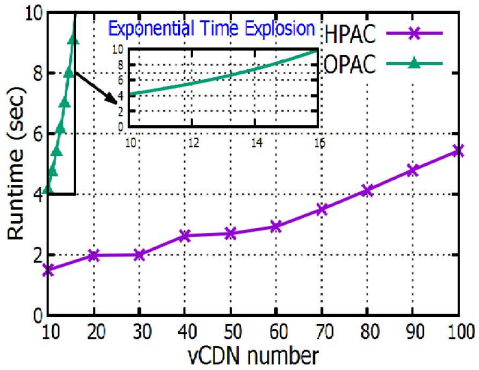


Figure 9: OPAC and HPAC run-time in large network scale.

A brief comparison between the two approaches which answers implicitly to the question when to use OPAC and when to use HPAC is given in the Table III .

For the sake of clarifying better the efficiency of our solutions, a brief comparison between our algorithms (OPAC/HPAC) and the state-of-the-art is shown in Table IV. We proposed an exact method based on linear integer model that will always find the optimal solution (for small and medium instances of the problem). This exact model is used as a benchmark (the best point of comparison) of a novel heuristic algorithm, elaborated to cope with scalability issues of the problem. Nevertheless, we also compared in Table IV our approaches to similar solutions in the literature.

D. Integration of the Algorithms

To our knowledge, the proposed optimization algorithms have to be integrated in a vCDN controller (as seen Fig. 2). It interacts with the vCDN manager software in a legacy NFV-MANO framework [19]. Further, as the migration problem deals with different heteroclite parameters and variables, a clear view about how they are extracted and reflected on an SDN/NFV framework is necessary.

In Fig. 10 and for the sake of simplicity, the two main use cases needed for integrating OPAC/HPAC algorithms by the

TABLE III: Efficiency comparison between OPAC and HPAC; SS: Small Scale, LS: Large Scale $N = |V(G)|$, $M = |E(G)|$

Metrics	***OPAC***	***HPAC***
Algorithm complexity	Exponential	$\mathcal{O}(N^3 * M^{1/2})$
Run-time	A few seconds in SS; a few minutes in *LS*	A few second in *SS/LS*
Total migration cost	Excellent in SS;unfeasible in *LS*	Good in *SS/LS*
Total migration time (i.e.,delay)	Good in *SS*;unfeasible in *LS*	Excellent in both *SS/LS*
Replication number	Free/loose in *SS*; unfeasible in *LS*	Free/loose in *SS/LS*
vCache/vStream cost	Good minimization in *SS*	Excellent minimization in both *SS/LS*

TABLE IV: Comparison between OPAC/HPAC and state-of-the-art

Work	***Approach***	***Metrics***	***Limitations***
Niels et al. [6]	Exact	Bandwidth, deployment cost	No virtual node cooperation, no QoE parameters, missed capacity and RAM constraints
Michele et al. [7]	Exact	Usage cost	Known traffic distribution (static), unreliable
Hendrick et al. [8]	Exact	Deployment cost, service request	Virtual CPU, virtual storage, virtual capacity, and QoE constraints are missed
Bernardetta et al. [9]	Exact	Allocated computing resources, link utilization	Prioritization method with orthogonal cost functions, virtual RAM and QoE constraints are missed
Mathieu et al. [10]	Heuristic	CAPEX, server's load	Missed NFV constraints
Mathieu et al. [11]	Exact and Heuristic	CAPEX, server's load	Offline measurement, and no migration
Miloud et al [12]	Exact and Heuristic	Traffic's load	Lacks system constraints and there is no VNF (PDN-GW) migration
OPAC&HPAC algorithms	Exact and Heuristic	System, network, and quality metrics	No limitations

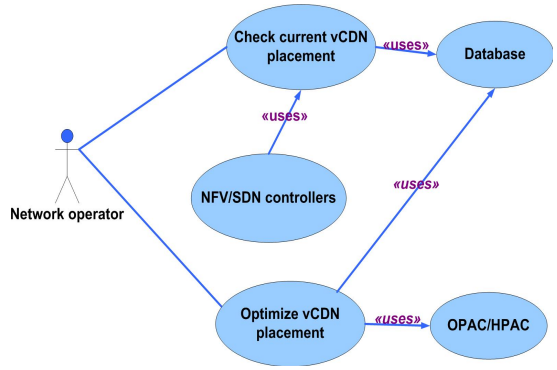


Figure 10: Main use-cases for OPAC/HPAC in an SDN/NFV framework.

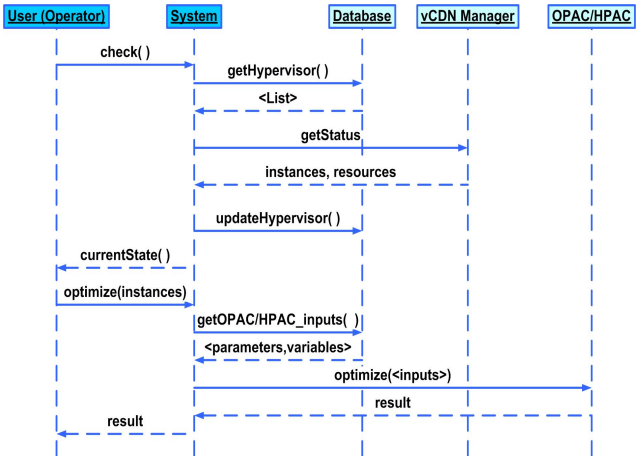


Figure 11: Sequence diagram for OPAC/HPAC integration.

network operator are depicted. Indeed:

- Use case 1: The network operator checks the current placement of vCDNs. To do this, he queries a Database using SQL structured language and an NFV/SDN controllers (e.g., Openstack [20]/Opendaylight [21]) using API interfaces. Openstack horizon and Opendaylight DLUX provide respectively the system and network resource information needed to complete the operator database.
- Use case 2: The network operator requests for the vCDN optimization placement/migration result using either OPAC or HPAC according to the actual network scale (small or large).

From an operator perspective, integrating the proposed algorithms may follow the sequence diagram represented by the Fig. 11. Indeed, the main stakeholders are:

- User (Operator): Operator of the infrastructure, deciding the migration or not of a vCDN instance according to the current state of the environment and the result given by the optimization algorithms OPAC/HPAC.
- System: This system, offering the Users an interaction with the OPAC/HPAC algorithms inputs and results. It still maintains the operator database updated by the necessary information about the operator's servers (*getHypervisor()*) and the provider's vCDNs (*getStatus()*).
- Database: It contains the information about the SDN/NFV infrastructure status and values. In this case, it is an SQLite database.
- vCDN Manager: It takes care of the interaction with the software managers running in the architecture, to poll from them the required information for the OPAC/HPAC algorithm. It will use internally API calls to the different software managers (Openstack Horizon, Opendaylight, etc.).
- OPAC and HPAC Algorithms: Implementation of the optimization algorithms that optimize the vCDN placement and migration based on exact/heuristic optimization approaches. A file is given as an input and a file is returned as an output of the selected algorithm. The algorithm selection depends mainly on the network scale.

For simplicity, when the user (operator) executes the optimize command, the system fetches the operator databases to get the required information $(getOPAC/HPAC_input())$in order to launch in turn the optimize command. Then, OPAC/HPAC algorithm decides where to place and migrate the vCDNs and provide the system/operator the result. Finally, the system executes the migration process according to our results and releases the dedicated resources in case of Service Level Agreement (SLA)-expiration/vCDN-delete-request.

Recall that the proposed algorithms involved also the content provider who wanted to rent a cloud of vCDN for its customers. Therefor, from a content provider perspective, the main exchanges between the content provider, the vCDN manager and the user (operator) are depicted:

- The content provider (e.g., YouTube) requests a vCDN creation during a specific time (e.g., 2 hours) and with a specific QoE (e.g., excellent) and to cover a specific region (e.g. Paris).
- The network operator, representing the owner of the NFV infrastructure, checks the current state of its NFV resources and call the proposed optimization algorithms through the provided *system*.
- This system interacts with a operator database server to retrieve the required information about network topology, NFV resources and update another database dedicated to the placement/migration decision.
- The NFV Infrastructure (NFVI) administrator interacts with the decision's database to migrate the resources while keeping the signed SLA between the content provider and the network operator valid and standing.
- Optionally, the NFVI administrator may let the content provider as the manager of the video content.
- The decision database is updated by the operator for security issues.
- In case of SLA expiration (e.g, covering time expiration, vCDN delete-request, etc..), the operator releases the dedicated NFV resources.

VI. Conclusion

This paper presents two optimization solutions either for the placement or the migration problem of virtual CDNs. Multiple constraints that are strongly related to the CDN virtualization are taken into account such as vCDN size, content resolution and system/network requirements. In addition, the two optimization algorithms OPAC and HPAC target respectively different network scales. Then, they are modeled and implemented. In small scale networks, results show that $|F|$ have more significant impact on the average migration cost in the case of using HPAC rather than OPAC. Nevertheless, it is noticeable that HPAC converges to the optimal solution and outperforms OPAC in specific metrics. In large scale, HPAC gives a short execution time (e.g. the run-time was in terms of a few seconds) which proofs its efficiency and scalability. In this network scale, The HPAC's results show that $|F|$ have insignificant impact on the migration time, the migration cost and the replication number since the system reaches quickly its stability. Moreover, the two approaches are integrated in an SDN/NFV framework and the main use case, sequence diagram from either an operator or content provider perspective. Future work can be conducted on integrating the aforementioned placement algorithms using Information-Centric Networking (ICN) context regarding its ubiquitous and in-networking caching features.

References

[1] B. Han, V. Gopalakrishnan, L. Ji, S. Lee, "Network Functions Virtualization: challenges and opportunities for innovations," IEEE Communications Magazine, vol. 53, pp. 90-97, February 2015.

[2] ETSI GS NFV V1.1.1: ETSI-NFV Network Functions Virtualization (NFV); use cases, October 2013.

[3] French FUI-18 DVD2C project: https://dvd2c.cms.orange-labs.fr/public-dvd2c/bienvenue-sur-le-site-du-projet-dvd2c.

[4] H. Ibn-Khedher, E. Abd-Elrahman, and H. Afifi, "Network issues in virtual machine migration," IEEE International Symposium on Networks, Computers and Communications (ISNCC). Hammamet, vol. 1, pp. 1-6, May 2015.

[5] H. Ibn-Khedher, E. Abd-Elrahman and H. Afifi, "OMAC: Optimal migration algorithm for virtual CDN," 23rd International Conference on Telecommunications (ICT), Thessaloniki, pp. 1-6, May 2016.

[6] N. Bouten, J. Famaey, R. Mijumby, B. Naudts, J. Serrat, S. Latr, and F. Truck, "Toward NFV-based multimedia delivery," IFIP/IEEE International Symposium on Integrated Network Management (IM). Ottawa, pp. 738-741, May 2015.

[7] M. Mangili, F. Martignon, and A. Capone, "Stochastic planning for content delivery : unveiling the benefits of Network Functions Virtualization," IEEE 22nd International Conference Network Protocols (ICNP). Raleigh, pp. 344-349, October 2014.

[8] H. Moens and F. de Turck, "VNF-P: A model for efficient placement of virtualized network functions," 10th International Conference on Network and Service Management (CNSM) and Workshop. Rio de Janeiro, pp. 418-423, November 2014.

[9] B. Addis, D. Belabed, M. Bouet, S. Secci, "Virtual Network Functions placement and routing optimization," IEEE 4th International Conference on Cloud Networking (CloudNet). Niagara Falls, pp. 171-177, October 2015.

[10] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of vDPI functions in NFV infrastructures," 1st IEEE Conference on Network Softwarization (NetSoft). London, pp. 1-9, April 2015.

[11] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized Deep Packet Inspection in SDN," IEEE Military Communications Conference (MILCOM). San Diego, pp. 992-997, November 2013.

[12] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," IEEE Wireless Communications & Networking Conference (WCNC). Istanbul, pp. 2402-2407, April 2014.

[13] R. E. Gomory, T. C. Hu, "Multi-terminal network flows," Journal of the Society for Industrial and Applied Mathematics, pp. 551-570, 1961.

[14] B. Korte and J. Vygen, *Combinatorial optimization: theory and algorithms,* 4th ed. Springer Publishing Company, Incorporated, 2007.

[15] R. Ramesh, T. Telikepalli and P. D. Kavitha, *"An (mn) Gomory-Hu tree construction algorithm for unweighted graphs,"*, Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, New York, pp. 605-614, 2007.

[16] R. Cole and R. Hariharan, *"A fast algorithm for computing steiner edge connectivity,"* Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing. New York, pp. 167-176, 2003.

[17] IBM ILOG CPLEX Optimization Studio: http://www-01.ibm.com/software/websphere/products/optimization/cplex-studio-community-edition/.

[18] The Jupyter Notebook: http://ipython.org/notebook.html

[19] ETSI GS NFV-MAN 001 V1.1.1 (2014): Network Functions Virtualization (NFV); management and orchestration, December 2014.

[20] Openstack For Telecom & NFV: https://www.openstack.org/telecoms-and-nfv/

[21] OpenDaylight Platform:https://www.opendaylight.org/start