Computer Networks 125 (2017) 90–102

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Energy-Efficient dynamic virtual network traffic engineering for north-south traffic in multi-location data center networks

Mirza Mohd Shahriar Maswood[a], Chris Develder[b], Edmundo Madeira[c], Deep Medhi[a,*]

[a] University of Missouri–Kansas City, Kansas City, MO, USA
[b] Ghent University – Imec, Belgium
[c] University of Campinas, Brazil

ARTICLE INFO

Article history:
Received 2 November 2016
Revised 7 March 2017
Accepted 12 April 2017
Available online 19 April 2017

Keywords:
Data center networks
Resource optimization and allocation on-demand
Request blocking
Energy efficiency
Virtual network
Dynamic traffic engineering

ABSTRACT

We consider the problem of allocating data center (DC) resources for cloud enterprise customers who require guaranteed services on demand. In particular, a request from an enterprise customer is mapped to a virtual network (VN) class that is allocated both bandwidth and compute resources by connecting it from an entry point of a data center to one or more hosts while there are multiple geographically distributed data centers to choose from. We take a dynamic traffic engineering approach over multiple time periods in which an energy-aware resource reservation model is solved at each review point. For the energy-aware resource reservation problem, we present a mixed-integer linear programming (MILP) formulation (for small-scale problems) and a heuristic approach (for large-scale problems). Our heuristic is fast for solving large-scale problems where the MILP problem becomes difficult to solve. Through a comprehensive set of studies, we found that a VN class with a low resource requirement has a low blocking even in heavy traffic, while the VN class with a high resource requirement faces a high service denial. Furthermore, the VN class having randomly distributed resource requirement has a high provisioning cost and blocking compared to the VN class having the same resource requirement for each request although the average resource requirement is same for both these VN classes. We also observe that our approach reduces the maximum energy consumption by about one-sixth at the low arrival rate to by about one-third at the highest arrival rate—this also depends on how many different CPU frequency levels a server can run at.

1. Introduction

The increasing growth of cloud based applications such as video streaming, web search, distributed file systems, scientific computations, software libraries and document collection made the data centers (DC) a popular platform in the Internet world. Companies such as Amazon, Google, Facebook, and Yahoo! routinely employ data centers for storage, web services and large-scale computations [1–3]. With the increase in demand, the size and number of DCs are increasing day by day. Large-scale data centers are set up with a large number of servers that are interconnected through routers, switch, and high speed links [4]. Due to the growing usage of data centers, the expenses of maintenance are also increasing. Power consumption is a major concern in operating data centers as most of the equipment in data centers are temperature sensitive and cooling through air and water is necessary to keep the temperature within an acceptable limit. Moreover, operating the servers, routers and switches also requires a huge amount of power. Data centers in the USA consumed about 91 billion kilowatt hours annually in 2013 and are estimated to consume 140 billion kilowatt-hours of electricity annually by 2020 [5]. Hence, reducing the energy consumption of data centers has been a challenging research problem. The ultimate aim behind designing a data center is reducing the expenses while gaining the highest efficiency.

There has been a number of contributions so far to increase the efficiency of data centers by better utilizing the server resources, applying traffic engineering techniques to reduce the bandwidth and other operational costs. Some of them [6,7] focus on energy efficient resource provisioning using dynamic traffic engineering. However, to our knowledge, no work has considered how both compute resources at the end hosts and network resources inside the data center are allocated to satisfy the request of virtual network (VN) customers while minimizing both energy consumption and bandwidth cost. Secondly, most work related to traffic engi-

* Corresponding author.
E-mail addresses: mmnt7@mail.umkc.edu (M.M.S. Maswood), chris.develder@intec.ugent.be (C. Develder), edmundo@ic.unicamp.br (E. Madeira), dmedhi@umkc.edu (D. Medhi).

http://dx.doi.org/10.1016/j.comnet.2017.04.042
1389-1286/© 2017 Published by Elsevier B.V.

neering of intra-DC networks consider east-west traffic, i.e., the intra data center traffic between hosts. In our work, we focus instead on enterprise customers' requests that result in north-south traffic in data centers requiring both network bandwidth and server resources. In particular, we address serving different enterprise customer groups using VNs at data centers through dynamic traffic engineering by allocating both network bandwidth and processing resources efficiently, while factoring in energy consumption. That is, we consider the north-south traffic environment where each request consists of a two-tuple demand: one for data center network bandwidth and the other for the processing demand at the end hosts.

Our work has three notable contributions beyond the existing work.

- For the dynamic traffic engineering problem, we present a novel mixed-integer linear programming (MILP) formulation that is solved at each review point to minimize a composite objective that consists of bandwidth cost, energy consumption cost and DC-VN mapping cost from a traffic engineering point of view while satisfying the virtual network customers by using the minimum amount of resources from data centers. The MILP formulation allows the flexibility that requests arriving at a particular review point may be allocated to any of the available data centers; for the selected data center, it may use any of the entry points for the north-south traffic at the north end, and any of the hosts available at the south-end. Our formulation also considers a penalty cost for a blocked request due to the potential loss of revenue.
- We present a heuristic as an alternative to solving the MILP formulation to test the performance of our framework for more realistic large scale data center networks. Our heuristic compares favorably with the solutions obtained for the MILP formulation for small-scale problems, and is fast to solve large-scale problems.
- We present an insight on how different classes of VN customers are affected in terms of resource allocations with north-south traffic in data centers. For instance, we address the following questions: How does each VN class perform? Is there any difference in the level of satisfaction among different VN classes in terms of cost and blocking, if so then by how much? By what percentage can we reduce energy consumption? How does the performance vary for different VN classes in a comparatively large data center?

The rest of the paper is organized as follows. In Section 2, we present the optimization formulation of the traffic engineering problem to be solved at each review point. In Section 3, we propose our heuristic. In Section 4, we summarize the simulation setup and parameter details. Results are discussed in Section 5. The related work is discussed in Section 6. Finally, in Section 7, we summarize our concluding remarks and discuss potential future work.

2. Model formulation

Our dynamic traffic engineering approach considers new request arrivals at random from customers, for which the resource allocation (both data center network bandwidth and host resources) is done at review point $t \in T$, where T is a discrete temporal window for dynamic traffic engineering consisting of review points. The duration of a new VN request that uses the data center is assumed to be random. Note that since the data center is set up to serve VN customers, at any time instant, there are existing VN tunnels and host resources allocated for prior requests. Thus, any (micro-)workload that needs immediate access to resources, that is, workload that cannot wait until the next review point, is

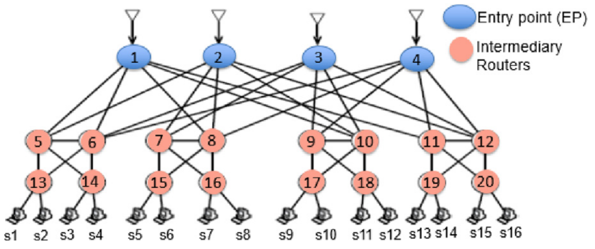


Fig. 1. Data center topology [8].

assumed to be served by existing VN channels and host resources assigned to the customers that were set up at earlier review points. Since such immediate workloads are served through existing resources, they are not modeled in our work. In other words, the scope of our work is to consider new requests at review points that are major requests requiring allocation of new bandwidths, virtual network tunnels and new resources. For this, we first present a mixed-integer linear programming (MILP) formulation in which we attempt to accommodate as many requests as possible while minimizing the resources requirement towards satisfying those requests in order to reduce the overall cost. To illustrate our approach, consider the single data center network topology shown in Fig. 1, which depicts just one site of the multi-location data center that our model considers. The entry point in a data center is then the north-end and the serving host is the south-end of the north-south traffic. Our approach assumes that there is a central controller that is responsible for solving the proposed optimization model and setting up the allocations. For instance, this can be accomplished by using a software-defined network (SDN) based approach.

In our model, each request consists of 2-tuple $\langle h, r\rangle$ where h is the bandwidth demand of the request and r is the processing resources required from a serving host. Thus, at a particular review point t, if a VN customer $v \in V$ has a request, the request tuple is further represented by $\langle h^{vk}(t), r^{vk}(t)\rangle$, which is to be served by data center $d \in D$. While the bandwidth demand needs to be satisfied by the capacity of the links within the data center $l \in L_d$ from the entry point $i \in I_d$ to a server $j \in J_d$, the processing resources must be satisfied by the servers' available resources. We assume that there is a given set of paths $P^{vd}_{ij}(t)$ from the entry point i to server j, which could be potentially different at each review point t.

For energy consumption, we consider that every server can run at a given set of CPU frequencies $f \in F$. At each particular frequency, a server works at a particular processing capacity a^d_{jf}. A specific amount of power b^d_{jf} is required to run the server at that frequency. If we run the server at the highest frequency, it offers the highest processing capacity, but consumes the highest amount of power. All notations used in our model are summarized in Tables 1 and 2.

We now present the constraints in our formulation. First, one DC out of the N DCs ($D = \{DC_1, \ldots, DC_N\}$) is at the most selected to meet request k from VN v at review point t:

$$\sum_{d\in D} u^{vkd}(t) \le 1, \quad k \in K_v(t), v \in V \tag{1}$$

Once a datacenter is responsible to fulfill the link bandwidth demand request k from VN v, then this data center must be the one from which the capacity is allocated for the bandwidth demand:

$$s^{vkd}(t) = h^{vk}(t)u^{vkd}(t), \quad k \in K_v(t), v \in V, d \in D \tag{2}$$

Next, either the total link bandwidth demand must then be served by the chosen data centers or if there is not enough bandwidth to serve a request from a particular VN, then this request will be

Table 1
Constants used in formulation.

Constants/Parameters:
D = Set of data centers, $N = \#(D)$
J_d = Set of servers in one data center
I_d = Set of entry points in one data center
V = Set of virtual networks
$K_v(t)$ = Set of requests from virtual network v at review point t
F = Set of frequencies in which server j can run
L_d = Set of links in one data ceneter
$P_{ij}^{vkd}(t)$ = Set of paths from entry point i to server j in datacenter d for request k of VN v at review point t
M = A large positive number
ε = A very small positive number
b_{jf}^{d} = Power consumption in server j of data center d at frequency f
$h^{vk}(t)$ = Bandwidth demand for request k of VN v at review point t
$r^{vk}(t)$ = CPU processing capacity demand for request k of VN v at review point t
a_{jf}^{d} = Capacity of server j of data center d at frequency f
$c_l^d(t)$ = Available capacity on link l of data center d at review point t
$\delta_{ijpl}^{vkd}(t)$ = Link-path indicator: 1 if path p which is set up from entry point i to server j uses link l of data center d in order to satisfy request k generated by VN v that comes to that entry point i of that data center d at review point t to be served, 0 otherwise
$\beta^d(t)$ = Normalized cost of data center d at review point t
α, μ, γ are weight parameters related to 3 optimization objectives

Table 2
Variables used in formulation.

Variables
$u^{vkd}(t)$ = Binary decision variable to choose data center d to satisfy request k from virtual network v at review point t
$s^{vkd}(t)$ = Bandwidth allocation going to data center d for request k of virtual network v at review point t
$\tilde{s}^{vk}(t)$ = Artificial bandwidth allocation for request k of virtual network v at review point t
$q^{vk}(t)$ = Binary decision variable to choose real allocation for request k of virtual network v at review point t
$\tilde{f}^{vk}(t)$ = Binary decision variable to choose artificial allocation assuming a very high penalty cost for request k of virtual network v at review point t
$y_{ij}^{vkd}(t)$ = Bandwidth allocation for request k of VN v from entry point i to server j of data center d at review point t
$\tilde{y}_{ij}^{vkd}(t)$ = Binary decision variable to select request k of VN v to be satisfied which comes to entry point i and served by server j of data center d at review point t (this parallels $y_{ij}^{vkd}(t)$)
$x_{ijp}^{vkd}(t)$ = Bandwidth allocation in path p, if request k comes to entry point i of data center d is transferred to server j uses path p at review point t
$z_l^{vkd}(t)$ = Bandwidth needed on link l of datacenter d for request k of VN v at review point t
$e_j^{vkd}(t)$ = The requirement of CPU processing capacity from server j of datacenter d to satisfy the request k coming from VN v at review point t
$g_{ij}^{vkd}(t)$ = Server resource (CPU processing capacity) allocation for request k of VN v through entry point i to server j of data center d at review point t
$\tilde{g}^{vk}(t)$ = Artificial server resource (CPU processing capacity) allocation for request k of VN v at review point t
$w_{jf}^{vkd}(t)$ = Binary decision variable to choose the optimum frequency f from the range of available frequencies of server j of data center d to meet the required demand of CPU processing capacity for request k of VN v at review point t

labeled as an artificial allocation, $\tilde{s}^{vk}(t)$, that allows us to keep a count on also blocked requests:

$$\sum_{d\in D} s^{vkd}(t) + \tilde{s}^{vk}(t) = h^{vk}(t), \quad k \in K_v(t), v \in V \tag{3}$$

We force the decision of choosing the binary variable of artificial allocation if a request cannot be served by limited resources:

$$\tilde{s}^{vk}(t) \le M\tilde{f}^{vk}(t), \quad k \in K_v(t), v \in V \tag{4}$$

A request from a VN can only be considered for either a real allocation or an artificial allocation but not for both at review point t:

$$\tilde{f}^{vk}(t) + q^{vk}(t) = 1, \quad k \in K_v(t), v \in V \tag{5}$$

If a request is considered for real allocation, the total link bandwidth demand then must be served by the chosen real data centers:

$$\sum_{d\in D} s^{vkd}(t) = h^{vk}(t)q^{vk}(t), \quad k \in K_v(t), v \in V \tag{6}$$

The total amount of the link bandwidth demand from particular VN v that will be served by data center d is the summation of the bandwidth that is allocated from all chosen entry points i to all chosen servers j of data center d at review point t:

$$\sum_{i\in I_d}\sum_{j\in J} y_{ij}^{vkd}(t) = s^{vkd}(t), \quad k \in K_v(t), v \in V, d \in D \tag{7}$$

Next, we introduce a binary shadow variable $\tilde{y}_{ij}^{vkd}(t)$ corresponding to $y_{ij}^{vkd}(t)$ to track one-to-one mapping from entry point i to server j at review point t by using a large positive number M and a small positive number ε:

$$y_{ij}^{vkd}(t) \le M\tilde{y}_{ij}^{vkd}(t), \quad j \in J_d,\ i \in I_d,\ k \in K_v(t),\ v \in V,\ d \in D \tag{8}$$

$$y_{ij}^{vkd}(t) \ge \varepsilon\tilde{y}_{ij}^{vkd}(t), \quad j \in J_d,\ i \in I_d,\ k \in K_v(t),\ v \in V,\ d \in D \tag{9}$$

Here, (8) and (9) together addresses the requirement that $\tilde{y}$ is 1 when the corresponding variable y has a positive flow; otherwise, $\tilde{y}$ as 0 when y is 0.

The bandwidth that is allocated to a particular path from entry point i to server j of data center d is given by using the path flow variables x_{ijp}^{vkd}:

$$\sum_{p\in P_{ij}^{vkd}(t)} x_{ijp}^{vkd}(t) = y_{ij}^{vkd}(t), \quad j \in J_d,\ i \in I_d,\ k \in K_v(t),\ v \in V,\ d \in D \tag{10}$$

If any bandwidth is allocated on particular path p to satisfy a portion of the request k of bandwidth demand h^{vk} from any VN v, then all the links associated with that path has to carry that portion of demand h^{vk}.

Therefore, we can determine the link flow on l for tuple $\langle v, d\rangle$:

$$\sum_{i\in I_d}\sum_{j\in J_d}\sum_{p\in P_{ij}^{vkd}(t)} \delta_{ijpl}^{vkd}(t)x_{ijp}^{vkd}(t) = z_l^{vkd}(t)$$
$$l \in L_d, k \in K_v(t), v \in V, d \in D \tag{11}$$

while the total amount of bandwidth required in one link l of data center d to satisfy the requests of all VNs must not exceed the capacity of that link of this data center:

$$\sum_{v\in V}\sum_{k\in K_v(t)} z_l^{vkd}(t) \le c_l^d(t), l \in L_d, d \in D \tag{12}$$

Furthermore, we must determine whether a request can be served with limited server resources or not. If there is a resource limitation to serve a particular request from a VN at review point t, then the binary variable to choose an artificial allocation for that request will be 1. This condition is satisfied by the following constraints:

$$\sum_{d\in D}\sum_{i\in I_d}\sum_{j\in J_d} g_{ij}^{vkd}(t) + \tilde{g}^{vk}(t) = r^{vk}(t), \quad k \in K_v(t),\ v \in V \tag{13}$$

$$\tilde{g}^{vk}(t) \le M\tilde{f}^{vk}(t), \quad k \in K_v(t), v \in V \tag{14}$$

$$\sum_{d\in D}\sum_{i\in I_d}\sum_{j\in J_d} g_{ij}^{vkd}(t) = r^{vk}(t)\, q^{vk}(t), \quad k \in K_v(t),\ v \in V \tag{15}$$

Next we address resource allocation of $r^{vk}(t)$ to the appropriate tuple $\langle d, i, j\rangle$, ensuring this in accordance with shadow variable $\tilde{y}$.

$$g_{ij}^{vkd}(t) \leq M\tilde{y}_{ij}^{vkd}(t), \quad j \in J_d,\ i \in I_d,\ k \in K_v(t),\ v \in V,\ d \in D \quad (16)$$

$$g_{ij}^{vkd}(t) \geq \varepsilon\tilde{y}_{ij}^{vkd}(t), \quad j \in J_d,\ i \in I_d,\ k \in K_v(t),\ v \in V,\ d \in D \quad (17)$$

$$\sum_{i\in I_d} g_{ij}^{vkd}(t) = e_j^{vkd}(t),\ j \in J_d,\ k \in K_v(t),\ v \in V,\ d \in D \quad (18)$$

In (18), $e_j^{vkd}(t)$ represents the total amount of resources required from server j to satisfy a request from VN v at time t that uses the server coming through all entry points of a particular data center. The total resources allocated to each request from a particular server must be less than or equal to the available resources of that server of a data center:

$$e_j^{vkd}(t) \leq \sum_{f\in F} a_{jf}^d w_{jf}^{vkd}(t),\ j \in J_d,\ k \in K_v(t),\ v \in V,\ d \in D \quad (19)$$

Finally, a particular server j running at a particular frequency f can produce a particular capacity a_{jf}^d. However, a server cannot run at more than one frequency at a time:

$$\sum_{f\in F} w_{jf}^{vkd}(t) \leq 1, j \in J_d, d \in D, k \in K_v(t), v \in V \quad (20)$$

To achieve the goal of the optimization problem, we consider four cost components in the objective function: the network bandwidth cost, the server resource cost, the data center location cost and the penalty cost for those requests which are not satisfied by the limited resources identified through the artificial allocation. Furthermore, since resources are of different types, we take a utility function-based approach by assigning weights to different components that form the objective function. The first three sources of costs are assigned different weight parameters, α, μ, γ, to understand the influence of each term on the overall decision, while the penalty term is assigned a high penalty through parameter M. Thus, our goal is to accommodate as many requests as possible and this can be accomplished by minimizing the amount of resources used. That is, the objective function can be written as:

$$\min \alpha \sum_{d\in D}\sum_{v\in V}\sum_{k\in K_v(t)}\sum_{l\in L_d} z_l^{vkd}(t)$$
$$+\mu \sum_{d\in D}\sum_{j\in J}\sum_{v\in V}\sum_{k\in K_v(t)}\sum_{f\in F} b_{jf}^d w_{jf}^{vkd}(t)$$
$$+\gamma \sum_{d\in D}\sum_{v\in V}\sum_{k\in K_v(t)} \beta^d(t) u^{vkd}(t) + M\sum_{v\in V}\sum_{k\in K_v(t)} \tilde{f}^{vk}(t) \quad (21)$$

To summarize, our unified formulation addresses decision choices at three different levels: data center, entry point, and then the destination server. Secondly, we take power consumption into account in determining the right frequency for operating a server. Finally, we consider four cost components in the composite objectives.

3. Cost effective heuristic

The MILP problem is an NP-hard problem. Thus, due to the limitation of the optimization model to generate optimal solutions quickly for large scale problems in a dynamic traffic engineering framework, we have developed a heuristic shown in Algorithm 1. For the heuristic, we use the notations from Tables 1 and 2.

At a particular review point t, for all incoming requests with bandwidth and resource requirements, this heuristic attempts to obtain the best possible solution at this review point. The input for this heuristic and the output returned by this heuristic are given below:

Algorithm 1 Cost effective heuristic.

```
for all $d \in D$ do
  update $a_{jf}^d, f \in F, j \in J_d, d \in D$
  update $c_l^d, l \in L_d, d \in D$
end for

while $V \neq \emptyset$ && $D \neq \emptyset$ do
  for all $d \in D$ do
    for all $i \in I_d$ do
      for all $NS(j) \in EP(i)$ do
        $c_j^d = \max(a_{jf}^d)$
        for all $v \in V$ do
          for all $k \in K_v$ do
            if $c_j^d \geq r^{vk}$ then
              for all $f \in F$ do
                $a_{jf}^d = \min(a_{jf}^d) \geq r^{vk}$
              end for
              if $c_j^d \geq a_{jf}^d$ then
                $c_j^d = c_j^d - a_{jf}^d$
                $w_{jf}^{vkd} = 1$
                $r^{vk} = 0$
              end if
            end if
          end for
        end for
        for all $v \in V$ served by NS do
          use leftmost shortest path $p \in P_{ij}^{vkd}, i \to j$
          for all $l$ used in $p$ do
            if $c_l^d \geq h^{vk}$ then
              $z_l^{vkd} = c_l^d - h^{vk}$
              $h^{vk} = 0$
              $map \to u^{vkd}$
              $V = V \setminus k$
            end if
          end for
        end for
      end for
    end for
    $D = D \setminus d$
  end for
end while

count_blocking = 0
for all $v \in V$ do
  for all $k \in K_v$ do
    if $r^{vk} == 0$ && $h^{vk} == 0$ then
      return $u^{vkd}, w_{jf}^{vkd}, z_l^{vkd}$
    else
      count_blocking ++
    end if
  end for
end for
```

DC related Input: Number of DCs (N), all paths available $p \in P_{ij}^{vkd}$ $i \to j$, capacity of each link (c_l^d), capacity of each server at different frequencies (a_{jf}^d).

VN related Input: Resource requirement (r^{vk}) and bandwidth requirement (h^{vk}) to satisfy the requests at review point t.

Output: Near optimal solution to satisfy a request or report that request as blocked.

The heuristic works on the first fit principle. At first, the heuristic updates the existing capacity of resources based on the given input discussed above. To find the best way of allocating resources,

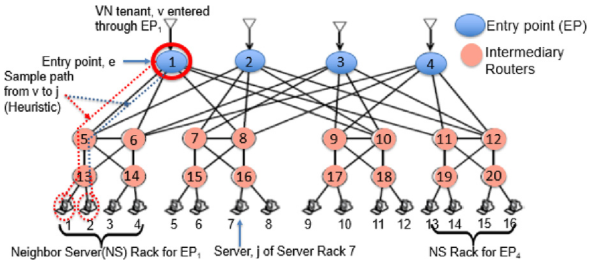


Fig. 2. Data center topology [8].

the heuristic picks one data center among all available data centers and continues to use it until either all servers or required links to establish a path from an entry point to a server are exhausted. Among all available entry points in that DC, the heuristic starts with one entry point (EP) and continues to allocate requests through this point until either all neighbor servers (NS) or all required links to establish a path from that entry point to an NS are occupied. By neighbors, we mean that two edge switches are considered as the neighbor edge for each entry point; then, for a particular entry point, the servers which are connected to this neighbor edge of the entry point are considered as a neighbor server (NS) for this entry point. From all the available neighbor servers, the heuristic picks a server from a neighbor server rack and continue to use the servers from that rack until all servers are occupied. When all the servers from that rack are occupied or do not find not enough capacity for any of the required links to establish a path, the heuristic starts with another neighbor server rack. This way the heuristic continues to allocate from the available resources to satisfy all the requests arriving at a review point.

In our approach, a server's goal is to fill as many requests as it can. To do so, at first, this server starts with the maximum available capacity and continue to fit requests until it reaches the limit of its capacity or all the requests are allocated with required compute resources. While doing so, from all the available capacity of that server, the heuristic tries to find the minimum capacity using which resource requirement from one request can be satisfied. After finding the minimum resource requirement, this quantity is reduced from the maximum available capacity. Through this, the heuristic is able to determine the best capacity in which a server should run. Furthermore, the heuristic gives us the information that by running the server at this frequency, how processing capacity that is generated is fractionally allocated among different requests. After being ensured about the resource fulfillment from a server, the heuristic uses the leftmost shortest path to route all the requests that can be satisfied by that server from the entry point to the targeted server. Now, for all the requests served by this server, once the shortest path is established, link capacity is modified by reducing the required link capacity from the currently available link capacity (from the given input in the review point) for each link.

We illustrate the heuristic using Fig. 2. When a number of requests arrives at a review point, each request is attempted in a sequential order. The first request picks the leftmost data center (where data centers are numbered left to right) and enters through EP1 (if available). Then, it tries the leftmost shortest path, 1-5-13, to reach server 1. if server 1 is not available, it tries server 2. In the case of resources not available either at server 1 or 2, the request tries the path 1-6-14 to reach server 3 or server 4. In case none of the paths or servers are accessible from EP1 to satisfy the request, then an entry through EP2 is initiated to reach server 5, 6, 7, or 8. Thus, the attempts are made in the following order: 1-5-13-s1, 1-5-13-s2, 1-6-14-s3, 1-6-14-s4, 2-7-15-s5, 2-7-15-s6, 2-8-16-s7, 2-8-16-s8, and so on. This hunting process is continued until the request is fulfilled by a data center, a server with a path with the required bandwidth; consequently, the available bandwidth and server resources are updated on the path and the server. If after trying all data centers and paths and servers, the request cannot not be satisfied, it is deemed blocked. It may be noted that at any review point, a request may not be satisfied, but one next in its sequence may be satisfied. This is because the next request may have less bandwidth and/or resource requirements than the previous one since we assume that arriving requests are heterogeneous.

4. Simulation study setup and parameter values

To conduct our study, we use the data center topology shown in Fig. 1. We set a maximum of two data centers ($N = 2$) in our study. Each data center is considered to be identical in this study; the number of servers in each data center are the same and all links inside the data center are set with the same capacity. For the MILP model, we set $P_{ij}^{vd}(t) = 4$ paths from an entry point to a server in which the bandwidth will be allocated to satisfy a specific request for the duration of this request.

We divided our studies into eight cases that are clustered into three groups as listed in Table 3 (H, VH$_a$, VH$_b$, and VR in this table are described later in this section). The first group, Group-A, consists of Case-1, Case-2, and Case-3, where the number of servers in each data center is set to 16 and the capacity on each link is set to 12, to reflect small-scale DCs. Comparing the results of the heuristic against the MILP formulation in a dynamic traffic engineering environment was done for Case-1 and Case-2. The MILP formulation used at each review point of the dynamic traffic engineering problem was solved using AMPL/CPLEX (v 12.6.0.0). Beyond this size, CPLEX was found to be highly time consuming to obtain even a near optimal solution by setting a CPLEX option of node limits to 1000 for the branch-and-cut method. In Case-3, we used the heuristic to compare two types of demands.

The second group, Group-B, in Table 3 consists of Case-4, Case-5, and Case-6 for large-scale DCs. In this group, we varied the number of servers from 800 to 1,600, and capacity of each link between 500 and 1000 to understand a number of situations, which were solved using the heuristic. Case-4 is to consider the situation where the processing capacity is the bottleneck. Case-5 considers the scenario when the link capacity is the bottleneck, while Case-6 is also is a case with capacity bottleneck while with a larger number of servers and entry points.

The third group, Group-C, in Table 3 consists of Case-7 and Case-8 is to exclusively understand energy consumption. For this study, it suffices to use a small-scale DC, but we change the frequency options to understand the gain in energy consumption.

We considered $V = 3$ classes of virtual networks to represent three different groups of enterprise customers that generate requests. Recall that a request is represented by the tuple $\langle h, r\rangle$. We varied $\langle h, r\rangle$ to create different types of demands to run the simulation for different cases as shown in Table 3; these are summarized in Table 4. Type-H in Table 4 assumes that all VN classes are homogeneous in terms of $\langle h, r\rangle$; this type was used in Case-1. Type-VH$_a$ reflects heterogeneous VN classes different bandwidth and processing demands, using $\langle h^1, r^1\rangle = \langle 3, 0.3\rangle$, $\langle h^2, r^2\rangle = \langle 6, 0.6\rangle$, $\langle h^3, r^3\rangle = \langle 9, 0.9\rangle$. VN-2 here requires twice as much resources as VN-1 while VN-3 requires three times as much resources as VN-1. This allows us to see how each VN class is treated by the DC due to heterogeneity.

Type-VR is similar to VH$_a$ except that we allow variation of the demand to be uniformly chosen at random within each VN from a range, i.e., $\langle h^1, r^1\rangle = \langle unif\{2, 3, 4\}, unif\{0.2, 0.3, 0.4\}\rangle$, $\langle h^2, r^2\rangle = \langle unif\{5, 6, 7\}, unif\{0.5, 0.6, 0.7\}\rangle$, $\langle h^3, r^3\rangle = \langle unif\{8, 9, 10\}, unif\{0.8, 0.9, 1.0\}\rangle$. The three types, type-H, type-VH$_a$, and type-VR,

Table 3
Summary of cases (Group-A: Case-1, Case-2, Case-3; Group-B: Case-4, Case-5, Case-6; Group-C: Case-7, Case-8).

Case	Description	# of servers in each data center	Link capacity of each link
Group-A			
Case-1	CPLEX and heuristic for demand type H: small-scale (Frequency-SetA)	16	12
Case-2	CPLEX and heuristic for demand type VH_a: small-scale (Frequency-SetA)	16	12
Case-3	Heuristic for demand type VH_a and VR: small-scale (Frequency-SetA)	16	12
Group-B			
Case-4	Heuristic for demand type VH_b with Processing Capacity as Bottleneck: large-scale, 4 entry points (Frequency-SetB)	800	1000
Case-5	Heuristic for demand type VH_b with Link Capacity as Bottleneck: large-scale, 4 entry points (Frequency-SetB)	800	500
Case-6	Heuristic for demand type VH_b: large-scale, 8 entry points (Frequency-SetB)	1600	600
Group-C			
Case-7	High Frequency Options: HFO (using demand type VH_b and Frequency-SetB)	16	12
Case-8	Low Frequency Options: LFO (using demand type VH_b and Frequency-SetC)	16	12

Table 4
Values of the general parameters used for this research for VN customers with different demand types.

Demand types	Parameters	Values
Type-H: Homogenous Bandwidth and CPU Processing Capacity for each request from all 3 VNs	Bandwidth Demand from VN-1, VN-2 and VN-3	6
	CPU Processing Capacity Demand from VN-1, VN-2 and VN-3	0.6
Type-VH_a: Different Bandwidth and CPU Processing Capacity demand for different VNs while the demand is fixed within each VN	Bandwidth Demand-VN-1	3
	Bandwidth Demand-VN-2	6
	Bandwidth Demand-VN-3	9
	CPU Processing Capacity Demand-VN-1	0.3
	CPU Processing Capacity Demand-VN-2	0.6
	CPU Processing Capacity Demand-VN-3	0.9
Type-VR: Different Bandwidth and CPU Processing Capacity demand for different VNs while with random within a fixed range for each request from a particular VN	Bandwidth Demand-VN-1	*unif*{2, 3, 4}
	Bandwidth Demand-VN-2	*unif*{5, 6, 7}
	Bandwidth Demand-VN-3	*unif*{8, 9, 10}
	CPU Processing Capacity Demand-VN-1	*unif*{0.2, 0.3, 0.4}
	CPU Processing Capacity Demand-VN-2	*unif*{0.5, 0.6, 0.7}
	CPU Processing Capacity Demand-VN-3	*unif*{0.8, 0.9, 1}
Type-VH_b: Similar to Type-VH_a except of having different values for CPU Processing Capacity demand	Bandwidth Demand-VN-1	3
	Bandwidth Demand-VN-2	6
	Bandwidth Demand-VN-3	9
	CPU Processing Capacity Demand-VN-1	0.1
	CPU Processing Capacity Demand-VN-2	0.5
	CPU Processing Capacity Demand-VN-3	1

Table 5
Frequency-SetA: CPU frequencies, capacities and operational cost [9].

Frequency option	1	2	3	4	5	6	7	8
Normalized capacity	0.5385	0.6038	0.6692	0.7346	0.8	0.8645	0.9308	1
Power consumption (watts)	60	63	66.8	71.3	76.8	83.2	90.7	100

Table 6
Frequency-SetB: CPU frequency options, capacities and operational cost.

Frequency option	1	2	3	4	5	6	7	8	9	10
Normalized capacity	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Power consumption (watts)	10	20	30	40	50	60	70	80	90	100

are used in the first group of studies (Case-1, Case-2, and Case-3) listed in Table 3.

Type-VH_b listed in Table 4 are also heterogeneous demand but with an wider gap for processing requirements between the three VN classes. This type was used in the rest of studies (Cases-4 to Case-8).

For server frequencies, we used three sets of frequencies, labeled Frequency-SetA, Frequency-SetB, and Frequency-SetC shown in Tables 5, 6, and 7, respectively. Frequency-SetA was used in the small-scale DC study, Group-A (Case-1, Case-2, and Case-3). Frequency-SetB was created for two purposes: to allow more frequency options and to uniformly spread out normalized capacity; this set was used for in the large-scale DC study, Group-B (Case-4, Case-5, and Case-6). Finally, Frequency-SetC with less frequency

Table 7
Frequency-SetC: CPU frequency options, capacities and operational cost.

Frequency option	1	2	3
Normalized capacity	0.3	0.6	1
Power consumption (watts)	30	60	100

option was created to understand the energy consumption gain with larger number of frequency options compared lesser number of frequency options; this is used in Group-C (Case-7 and Case-8) for the energy consumption study.

All arrivals for the dynamic traffic engineering simulation were generated randomly. Specifically, we assumed that the request arrivals was a Poisson process and the service duration for the request arrivals was assumed to follow the negative exponential distribution with an average value of 5 time units measured in terms of the number of discrete review points. Note that with an increase in the arrival load, the system may not have sufficient capacity to accommodate all requests. Thus, our simulation environment also recorded any requests that were not satisfied by the system by tracking the blocked requests to determine the blocking rate. Through our initial experimentation, we attempted to find the arrival rate for which the blocking was approximately 1%. We refer to that arrival rate as a normal loaded network condition, and assigned the normalized load of 1.0. We then continued to increase the arrival rate until we found the arrival rate for which the average blocking was approximately 10% to indicate highly overloaded condition. Also, through our initial experimentation, we chose the weight factors for each term in the objective (21) and set them as $\alpha = 0.3, \mu = 0.05, \gamma = 8.1$ to understand the influence of the three cost components on the overall provisioning cost. They were chosen to give higher importance on the DC-VN mapping cost, followed by the bandwidth cost and finally, by the energy consumption cost, without any one of them being delegated to being an insignificant cost.

For our dynamic traffic engineering simulation, we first determined the warm-up time and then collected the data for a steady-state region after the warm-up time. For each arrival rate, we used 10 independent seeds and reported the results on the average value. We also computed the confidence interval and found the 90% confidence interval to be approximately 5% in cost variation for low arrival rates to 2.5% for high arrival rates.

5. Results

The scope of the simulation study is to understand the following issues: (1) comparison of the optimization model and the heuristic for dynamic traffic engineering, (2) service performance impact due to service heterogeneity as identified through VH_a and VR types of demands and answer the questions we raised in Section 1, and (3) reduction in power consumption due to our approach compared to the benchmarking when all server runs at its maximum capacity (labeled as "no optimization").

The choice of the parameters in our study was motivated by the set of questions we posed in Section 1 leading to formulating the following two postulates:

Postulate-1: We postulate that when the bandwidth demand and the resources (per request) vary uniformly from an average value, the cost and the blocking would be higher compared to when the bandwidth demand and resources for each request are fixed.

Postulate-2: We postulate that by taking three values for the requested bandwidth h and CPU resource r, i.e., the tuple $\langle h, r\rangle$ for different VN classes in increasing order, the VN class with the lowest resource requirement would receive better treatment (lower blocking and cost) by the network than the other. In the following subsections, we discuss the three broad scopes of our study while bringing up the postulates as applicable.

5.1. Comparison between CPLEX and Heuristic

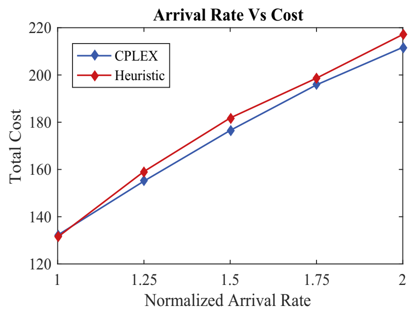
The purpose of our first set of experiments was to validate the performance of the heuristic compared to the MILP solution obtained using CPLEX. Indeed, we did not expect the MILP to scale to large problem instances, but hoped that our heuristic would provide solutions that were reasonably close to the MILP solution from CPLEX. More specifically, we compared them over the entire simulation duration for dynamic traffic engineering, not at a particular review point. For performance measures, we used the average cost and average blocking over the simulation duration.

Consider Case 1 first from Group-A, where the demands were homogeneous (H). From Fig. 3a, we observed that the maximum mean deviation between the result obtained from CPLEX and the heuristic was 2.99% for the average cost of provisioning for Case-1. This deviation was observed when the network was 50% more overloaded than the normalized request arrival rate to the network for the existing resources. However, this deviation did not increase as the load continued to go up, as we could see just a 1.75% deviation when the average arrival rate of the incoming traffic was 75% more than normalized arrival rate. From this figure, we note that the cost incurred from the solution by the heuristic is slightly higher than the CPLEX solution. Now, if we look at Fig. 3b, we can find that the maximum mean deviation between the heuristic and CPLEX is 3.69% at the 75% overloaded condition. Overall, we note that the blocking caused by using CPLEX was slightly higher than the heuristic at high overload. This can be understood by the greedy nature of CPLEX at each review point in solving the MILP problem exactly.

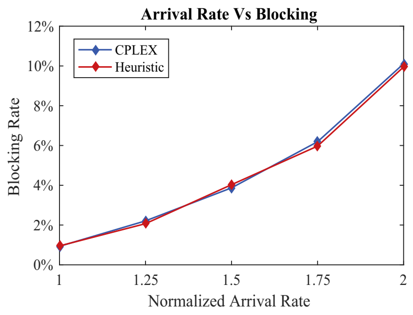
The pattern of this deviation can be further explained by considering the fact that the actual requests which were blocked by the heuristic and the MILP solution, might be different ones. In other words, the requests accepted by each approach would be different at a review point, meaning that their service durations would be different as well. Consequently, the residual bandwidth and resources available at future review points seen by the heuristic and the MILP approach could be different; this further led to requests blocked by the heuristic being different than the MILP solution. That is, Fig. 3a and b do not necessarily imply that the heuristic was better than CPLEX due to less blocking, but it rather showed that the performance between CPLEX and the heuristic for demand type H was almost similar in terms of average cost and average blocking. This observation is also true for the three individual cost constituents (bandwidth, energy consumption, and DC-VN mapping) as we can see from Fig. 3c.

Next we considered Case 2, where each VN had a different bandwidth and resource demand, labeled as type VH_a. From Fig. 4a and b, the maximum mean deviation between CPLEX and the heuristic is observed for VN class 3, which required additional resources per request compared to the other two VN classes. Higher resource requirements means high blocking for this VN class and this difference widens as the load increases. Again, the deviation in performance does not necessarily indicate that CPLEX would be better than the heuristic, or otherwise. Even though the maximum overall blocking rate is 11.53%, the maximum blocking rate for VN class 3 is 19.11%, which illustrates the performance deviation in a high blocking (overloaded) situation. We found that the maximum difference in cost for the VN class to be 3.14%. From Fig. 4c, we also note differences in the bandwidth cost for VN3 between the heuristic and CPLEX; in addition both the postulates are satisfied regardless of whether CPLEX or the heuristic is used for this case.

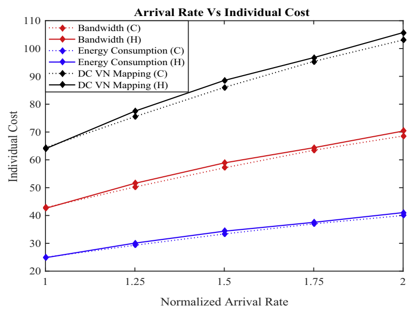
We now briefly comment on the computation time between CPLEX and the heuristic. For Case-1 and 2, we observe that our heuristic was approximately 240 × faster than CPLEX without much loss on the quality of the solution obtained in terms of cost and blocking.



(a) Total Cost

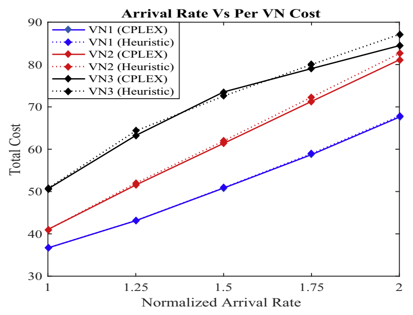


(b) Average Blocking Rate

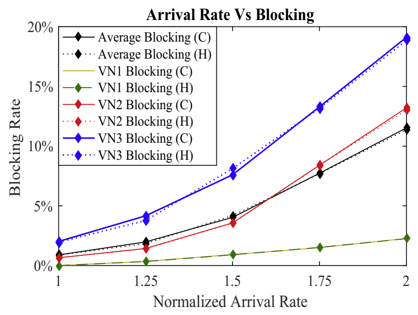


(c) Individual Cost

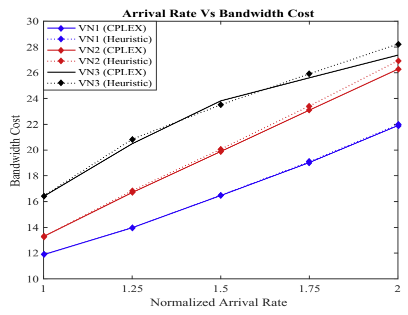
Fig. 3. CPLEX (C) vs. Heuristic (H) for Case-1.



(a) Each VN Cost

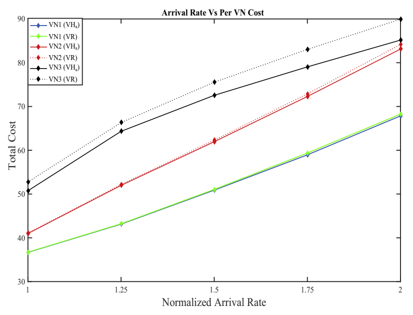


(b) Average and per VN Blocking Rate

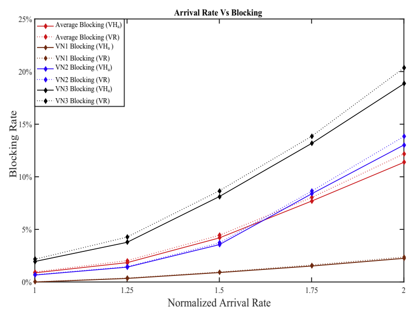


(c) Bandwidth Cost for each VN

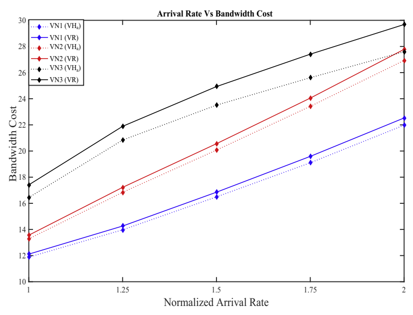
Fig. 4. CPLEX (C) vs. Heuristic (H) for Case-2.



(a) Each VN Cost



(b) Average and per VN Blocking Rate



(c) Bandwidth Cost for each VN

Fig. 5. VH$_a$ vs. VR (Case-3).

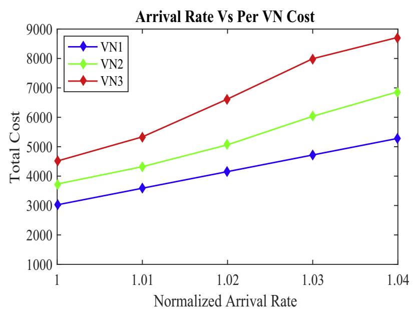
5.2. Service impact due to demand types VH$_a$ and VR

In this subsection, we study the impact of traffic patterns, in particular the cases of sets of heterogeneous requests (VH$_a$, with fixed, but different resource requirements for each VN class), and random variations in the resource requirements within each class (VR) as listed in Case-3. We report results for the heuristic solution, since we have established its solution quality in the previous subsection.

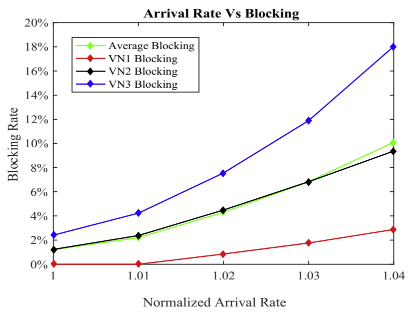
In Fig. 5a b, we present how the cost and blocking varies respectively for these two types of demands as the incoming load increases. We found that there is little difference in blocking for VN-1 between Case-VH$_a$ and Case-VR. On the other hand, this difference is noticeable for VN-2, and quite prominent for VN-3 as this class requires significantly more resources. In other words, the VN class for which the resource requirement for each request was randomly distributed within a range had a high blocking rate compared to the VN class having the same resource requirements for each request. The cost of providing connectivity for each VN customer is shown in (Fig. 5a). Naturally, the cost of provisioning VN-1 is always the lowest, regardless of the arrival rate, due to the lower resource requirements. Now, revisiting Postulates 1 and 2, we can see that our result satisfied both the postulates. We also plot the bandwidth cost for each VN as shown in Fig. 5c and observe almost the same behavior as in Fig. 5a.

5.3. Cost and service impact for large topology

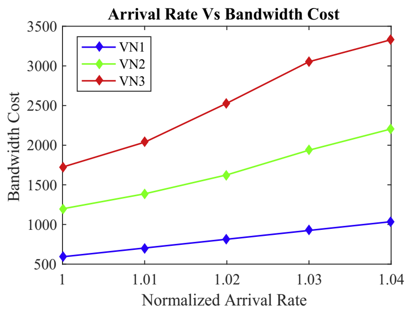
We now move to Group-B of the study. We divide the study reported in this section into two scenarios to address two sources of potential bottlenecks in the system. In the first scenario, we investigated how different VN classes were treated by the data centers when the servers' processing capacity was the bottleneck—this is labeled as Case-4 of Table 3. To consider this scenario, we provided abundant capacity to all links of the data centers to ensure that no request would face blocking because of not getting the sufficient amount of bandwidth that is required by that request; rather, the only blocking possible in this scenario was due to the lack of server resources. From Fig. 6a, we see that the cost of VN3 was always higher than the other two groups of VNs. However, the slope of in-



(a) Each VN Cost



(b) Average and per VN Blocking Rate



(c) Bandwidth Cost for each VN

Fig. 6. Performance analysis of demand type VH$_b$ (Case-4).

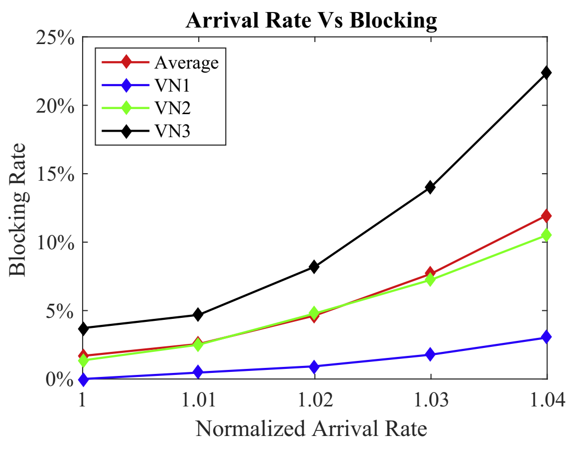


Fig. 7. Total and VN Blocking: VH$_b$ (Case-5).

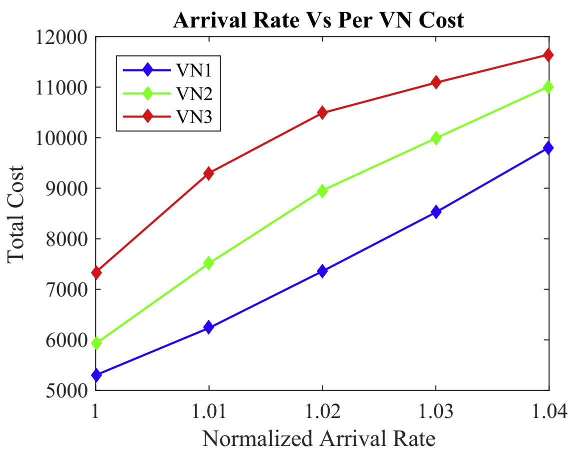


Fig. 8. Per VN Cost: VH$_b$ (Case-6).

crease in cost for VN3 started to reduce after the incoming traffic load reached 1.3% of the normal load as the blocking rate exceeded 10% (Fig. 6b) for this class. However, for the other two classes, we noticed a steady slope of increasing cost.

We further observe that VN2, having the resource requirement in between VN3 and VN1, and its cost and blocking are also at the middle of these classes, presented in Fig. 6a and b. From Fig. 6b, we can further observe that blocking for a customer class with less resource requirements (like VN1) is always lower. Even with a high traffic situation, the blocking rate for this class is less than 5%, where the blocking rate for VN2 and VN3 reached at around 10% and 18%, respectively. In Fig. 6c, the bandwidth cost for each VN class shows that there is a similar behavior to Fig. 6a.

Next, we investigated how the quality of service varied when the primary source of the bottleneck was network capacity compared to the server resources being the bottleneck, i.e., Case-5. From Fig. 7, we see that VN class 3 was more strongly affected than the other two classes. Thus, customers having a greater bandwidth requirement (i.e., VN 3), received worse treatment (more blocking) in a network having less link capacity compared to other types of customer classes having less bandwidth requirements, especially as the overload increases.

Again, both postulates held. However, the level of impact was different on the VN with the highest resource requirements depending on where the bottleneck in the system was.

Next, we tested the scalability of our framework using our heuristic for a larger data center system than Cases 4 and 5 by considering 8 entry points and 1600 servers in each data center; recall that this is listed as Case-6 in Table 3. We found that our developed heuristic could find the solution for this large topology as well. We present the average cost for each VN in Fig. 8. We noticed that the pattern of this figure is nearly similar to demand type VH of Case-3 as shown in Fig. 6a, and the postulates held.

5.4. Energy consumption

Another aim of our work is to reduce energy consumption. To understand how our approach helps to reduce energy consumption, we simulated two additional cases listed as Group-C in Table 3. First, we considered the data center topology with four entry points for the small-scale problem of 16 servers in each data center. We considered two options. In the high frequency option (Case-7), we considered that the CPU of each server could run at one frequency among ten different options, while in the low frequency option (Case-8), we reduced the number of frequency options to understand how the energy consumption varied. From Fig. 9 and Table 8, we clearly observe that our approach reduced the energy consumption by 84.83% at the low arrival rate (the best case with 10 frequency options available: Case-7) to 66.81% at the highest arrival rate (at the worst case with low frequency options available: Case-8), compared to if all servers were running at the highest frequency (labeled as “no optimization”). From Fig. 9, and Table 8, we further observed that our approach gained more reduction in energy consumption when the servers ran at more frequency options. We also point out that energy consumption for the low frequency options was more than the high frequency options, especially when the incoming traffic load was high. Hence, in brief, the percentage of reduction in energy consumption achieved through our approach depends on the granularity of available frequency in which the CPU of the servers could run.

Finally, to understand how much reduction in energy consumption could be achieved through our approach for a large-scale

Table 8
Percentage in energy reduction achieved by our heuristic compared to no optimization.

Normalized arrival rate (small-scale)	1	1.25	1.5	1.75	2
Case-7: HFO	84.83%	81.17%	78.1%	75.16%	72.11%
Case-8: LFO	82.58%	78.21%	74.56%	70.44%	66.81%
Noramalized arrival rate (large-scale)	1	1.01	1.02	1.03	1.04
Case-4	69.02%	63.58%	56.44%	48.45%	42.60%

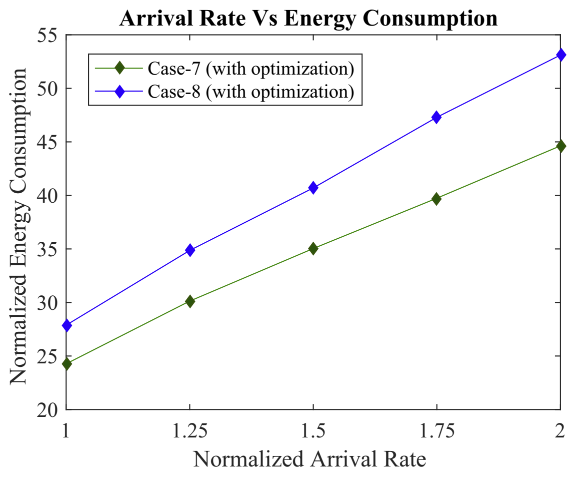


Fig. 9. Comparison of energy consumption cost between Case-7 and Case-8, with energy optimization.

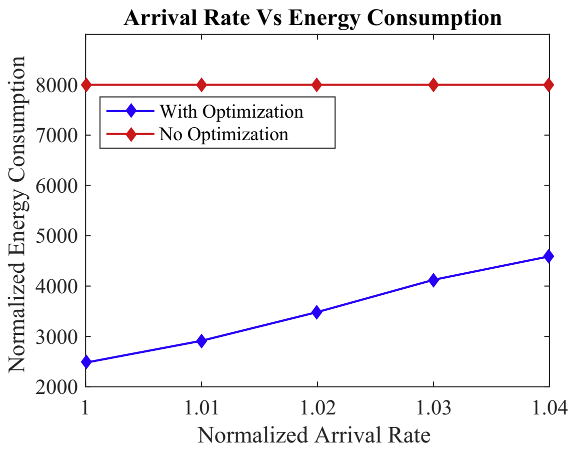


Fig. 10. Energy consumption cost with energy optimization and no optimization (Case-4).

problem, we further considered Case-4 that consists of 800 servers, mentioned earlier in Table 3, compared to no optimization. The findings are depicted in Fig. 10 and Table 8. From Fig. 10 and Table 8, we see that our approach reduced the energy consumption to 69.02% of the maximum energy cost at low arrival rate to 42.6% at the highest arrival rate compared to benchmarking with no optimization. The most significant factor to notice from this figure is that the reduction in energy consumption was less compared to Fig. 9. The reason behind this is that, in this case, we used the processing capacity of servers as the bottleneck. This means that all the servers of the available data centers was in use at the highest arrival rate. This ensures the maximum utilization of the servers' processing capacity. In consequence, the energy consumption cost became slightly higher than the result shown in Fig. 9. However, now the energy consumption by the servers was far less compared to no optimization. From these analyses, we can say that our approach can help design an energy efficient data center networking system.

5.5. Summary of observations

We now summarize the key observations:

1. In a dynamic traffic engineering environment, our heuristic is comparable to the MILP formulation using CPLEX in terms of cost and blocking. Our heuristic is approximately 240 times faster than CPLEX for small-scale problems and can be used for large-scale problems.
2. In general, the VN class with a higher resource requirement faces significantly higher blocking as the arrival rate increases while having a noticeably higher cost. A small random perturbation on the resource requirement of the VN class with the highest resource requirement can have a noticeably different performance impact at a high arrival rate. This is even though the average resource requirement is the same for both VN classes.
3. Blocking sharply increases at a much smaller overload for large-scale problems than compared to the small-scale problems. This behavior is consistent with a single-link loss system model (without routing and server selection) that can be computed with the Erlang-B blocking formula. The nonlinear concave behavior of Erlang-B blocking is well known as the load and capacity increase, impacting blocking, especially when the services have heterogeneous bandwidth requirements; see [10, Chapter 11] for a discussion.
4. Our approach reduces energy consumption by 42% to 84% depending on the granularity of the frequency options available and compared to when the servers are running at the highest frequency.

6. Related work

Early research on data center networks investigated architectural construction, operation and scalability of DCs [11–16]. Joint VM placement and routing for data center traffic engineering was addressed by Jiang et al. [17]. Similar to Jiang et al. [17], we also consider our problem from a traffic engineering point of view but we do not focus on VM placement; rather, we keep routing flexible in such a way that no dedicated server is required to satisfy demand from a particular VN. Any idle server is able to handle the request from any VN tenant. To satisfy a particular request, a server is chosen based on the resource demand and available resources of the server. Unlike their work, we take bandwidth guarantee into consideration. The issue of multiple service classes with heterogeneous requirements have been addressed for access control [18,19]; however, they do not consider two-tuple demands nor the implication of network routing.

Different approaches of optimization have been addressed in different research papers. In [20], a scheme has been proposed to optimize both virtual machine placement and traffic flow routing through dynamic VM migration and programmable flow-based routing. Xiang et al. [21] proposes an optimization technique to reduce both the latency and cost of data center.

Recently, much research has been done to increase the energy efficiency of a data center network [22–27]. A new data center architecture is presented in [23] and [24]. In [23], authors proposed a novel data center network architecture using optical multiple-input multiple-output (MIMO) orthogonal frequency division multiplexing (OFDM) technology. To achieve high energy efficiency, they used passive optical switch (PON) and parallel signal detection technology to detect multiple optical channels simultaneously while using a single photodetector [24]. proposed a SDN based Arrayed waveguide grating routers (AWGR) PON data center interconnection design to improve energy efficiency. Different techniques have been proposed in [22,25,26] and [27] to reduce the energy consuption of a data center in the network level.Yang et al. [22] talks about a solution to reduce energy consumption by using switch ports and link bandwidth optimally to avoid congestions and balance the load to increase the transmission capacity and save a significant amount of network energy in Data Center Network. However, they didn't consider optimizing energy in the server level. An ILP formulation followed by a heuristic is proposed in [25] to reduce the energy consumption in software defined data center networks by activating the switches selectively and scheduling multi-path routing carefully, according to the traffic demands in data center. A routing scheme has been proposed to reduce the energy consumption in the network level of data center in [26] which selects the flows iteratively to consume the residual capacities in the active nodes and allocate routes to flows based on the distributions of nodes, residual capacities and flow demands. A correlation-aware power optimization algorithm has been presented in [27] to dynamically combine traffic flows onto a small set of links and switches to shut down as many network devices as possible for reducing energy consumption.

Qian and Medhi [28] discussed the servers' operational cost optimization without taking data center architecture into consideration. and they did not consider the on-demand model either. In [8], authors presented a formulation to optimize the link cost in one data center, while we consider connecting multiple data centers. Unlike Owens and Medhi [8], we take two factors into account, which are energy consumption by the servers, and the DC VN mapping cost. In our earlier conference paper Maswood et al. [29], we combined three cost components (reducing link costs, power cost, and the DC VN mapping cost) together and impose weight parameters on each of these components to reflect their relative importance. In this paper, we extended the optimization model to consider requests that are not satisfied by explicitly introducing a set of artificial variables along with penalty costs for requests not satisfied. Furthermore, we now present a heuristic that can be used in large-scale problems. A novel contribution beyond the state-of-the-art is the dynamic nature of our model to provide on-demand service considering north-south traffic and finding the optimal resource requirement to contain service blocking within a tolerable range. Our model allows us to study service differences among different service classes identified through virtual networks. Moreover, we can also identify which servers are not used to serve the VN requests at a particular time, which allows us to determine servers in a lower power consumption mode.

7. Conclusion and future work

In this work, we presented a dynamic traffic engineering framework for resource allocation due to north-south traffic in a multi-location data center environment. We presented a novel MILP formulation and alternately a heuristic that is solved in this framework at each review point. Our approach is geared for enterprise customers that require resource guarantees from data centers.

We found that the MILP formulation is suitable for up to 32 servers. For higher traffic situation, our heuristic approach is much more suitable, and we tested and presented results for up to 3200 servers.

Our comprehensive study allowed us to answer a number of questions when resource requirements may vary for each request as well as may differ between different customers. In general, we observed that VN customers with the lowest resource requirements face the lowest blocking as the traffic is increased in the system. For VN customers with high resource requirement, blocking is significantly higher for heavy traffic.

A key observation is that our approach significantly reduces energy consumption compared to servers running at the highest frequency and it works better when we have more frequency options to choose from at which a server is allowed to operate. In other words, more frequency options for a server means higher reduction in energy consumption.

There are several future directions we wish to address. We do not allow partial fulfillment of a request if there is lack of sufficient resources to fully consider a request. Furthermore, we plan to add performance evaluation on the loads to a data center based on its geographical distance from different VNs. We also plan to explore different allocation policies so that service performance are comparable for different VN customer groups.

Acknowledgments

This research is partially supported by a School of Graduate Studies Research Grant at the University of Missouri-Kansas City, partially supported by National Science Foundation Grant # 1526299, and partially supported by CNPq (National Counsel of Technological and Scientific Development), Brazil.

References

[1] Amazon elastic computer cloud (Amazon EC2).
[2] D.F. Carr, How google works, Baseline Mag. 6 (6) (2006).
[3] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[4] A. Hammadi, L. Mhamdi, A survey on architectures and energy efficiency in data center networks, Comput. Commun. 40 (2014) 1–21.
[5] Natural Resources Defense Council, America's data centers consuming and wasting growing amounts of energy, 2014.
[6] J. Buysse, K. Georgakilas, A. Tzanakaki, M. De Leenheer, B. Dhoedt, C. Develder, Energy-efficient resource-provisioning algorithms for optical clouds, J. Opt. Commun. Netw. 5 (3) (2013) 226–239.
[7] M. Shojafar, N. Cordeschi, D. Amendola, E. Baccarelli, Energy-saving adaptive computing and traffic engineering for real-time-service data centers, in: Proc. of 2015 IEEE International Conference on Communication Workshop (ICCW), 2015, pp. 1800–1806.
[8] M.A. Owens, D. Medhi, Temporal bandwidth-intensive virtual network allocation optimization in a data center network, in: Proc. of 2013 IEEE International Conference on Communications (ICC), 2013, pp. 3493–3497.
[9] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, N. Gautam, Managing server energy and operational costs in hosting centers, in: ACM SIGMETRICS Performance Evaluation Review, vol. 33, 2005, pp. 303–314.
[10] D. Medhi, K. Ramasamy, Network routing: algorithms, protocols, and architectures, Morgan Kaufmann/Elsevier, 2007.
[11] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: ACM SIGCOMM Computer Communication Review, vol. 38, 2008, pp. 63–74.
[12] G.P. Alkmim, D.M. Batista, N.L. da Fonseca, Approximated algorithms for mapping virtual networks on network substrates, in: Proc. of 2012 IEEE International Conference on Communications (ICC), 2012, pp. 1460–1465.
[13] A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, P. Patel, S. Sengupta, VL2: a scalable and flexible data center network, in: ACM SIGCOMM Computer Communication Review, vol. 39, 2009, pp. 51–62.
[14] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, S. Lu, BCube: A high performance, server-centric network architecture for modular data centers, ACM SIGCOMM Comput. Commun. Rev. 39 (4) (2009) 63–74.
[15] F. Hao, T. Lakshman, S. Mukherjee, H. Song, Enhancing dynamic cloud-based services using network virtualization, in: Proc. of the 1st ACM workshop on Virtualized infrastructure systems and architectures, 2009, pp. 37–44.
[16] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, A. Vahdat, PortLand: a scalable fault-tolerant layer 2 data center network fabric, in: ACM SIGCOMM Computer Communication Review, 39, 2009, pp. 39–50.

[17] J.W. Jiang, T. Lan, S. Ha, M. Chen, M. Chiang, Joint VM placement and routing for data center traffic engineering, in: Proc. of 2012 IEEE INFOCOM, 2012, pp. 2876–2880.
[18] B. Kraimeche, M. Schwartz, Analysis of traffic access control strategies in integrated service networks, IEEE Trans. Commun. 33 (1985) 1085–1093.
[19] D. Medhi, A. van de Liefvoort, C. Reece, Performance analysis of a digital link with heterogeneous multislot traffic, IEEE Trans. Commun. 43 (1995) 968–976.
[20] W. Fang, X. Liang, S. Li, L. Chiaraviglio, N. Xiong, Vmplanner: optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers, Comput. Netw. 57 (1) (2013) 179–196.
[21] Y. Xiang, T. Lan, V. Aggarwal, Y.F.R. Chen, Joint latency and cost optimization for erasurecoded data center storage, ACM SIGMETRICS Perform. Eval. Rev. 42 (2) (2014) 3–14.
[22] T. Yang, Y.C. Lee, A.Y. Zomaya, Energy-efficient data center networks planning with virtual machine placement and traffic configuration, in: Proc. of 2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), 2014, pp. 284–291.
[23] P.N. Ji, C. Kachris, I. Tomkos, T. Wang, Energy efficient data center network based on a flexible bandwidth mimo ofdm optical interconnect, in: Proc. of 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), 2012, pp. 699–704.
[24] A. Hammadi, T.E.H. El-Gorashi, J.M.H. Elmirghani, Energy-efficient software-defined AWGR-based PON data center network, in: Proc. of the 2016 18th International Conference on Transparent Optical Networks (ICTON), 2016, pp. 1–5.
[25] D. Zeng, G. Yang, L. Gu, S. Guo, H. Yao, Joint optimization on switch activation and flow routing towards energy efficient software defined data center networks, in: Proc. of 2016 IEEE International Conference on Communications (ICC), 2016, pp. 1–6.
[26] L. Wang, A. Fern, F. Zhang, J. Wu, Z. Liu, et al., Multi-resource energy-efficient routing in cloud data centers with network-as-a-service, in: Proc. of 2015 IEEE Symposium on Computers and Communication (ISCC), 2015, pp. 694–699.
[27] X. Wang, Y. Yao, X. Wang, K. Lu, Q. Cao, Carpo: correlation-aware power optimization in data center networks, in: Proc. of 2012 IEEE INFOCOM, 2012, pp. 1125–1133.
[28] H. Qian, D. Medhi, Server operational cost optimization for cloud computing service providers over a time horizon, in: Proc. of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services (Hot-ICE), 2011.
[29] M.M.S. Maswood, C. Develder, E. Madeira, D. Medhi, Dynamic virtual network traffic engineering with energy efficiency in multi-location data center networks, in: Proc. of the 28th International Teletraffic Congress, 2016, pp. 10–17.

Mirza Mohd Shahriar Maswood received his B.Sc. degree in Electronics and Communication Engineering from Khulna University of Engineering and Technology, Bangladesh in 2010. He received his M.Sc. in Electrical Engineering from University of Missouri-Kansas City, USA in 2015. Currently, he is a Ph.D. student in Telecommunications and Computer Networking in University of Missouri-Kansas City. He is an assistant professor in the dept. of Electronics and Communication Engineering at Khulna University of Engineering and Technology. Also he is working as a graduate teaching assistant in University of Missouri-Kansas City. His research interests include Data center networking, Traffic Engineering and Neural Networking.

Chris Develder currently is associate professor with the research group IDLab in the Dept. of Information Technology (INTEC) at Ghent University - imec, Ghent, Belgium. He received the M.Sc. degree in computer science engineering and a Ph.D. in electrical engineering from Ghent University (Ghent, Belgium), in Jul. 1999 and Dec. 2003 respectively (as a fellow of the Research Foundation, FWO). From Jan. 2004 to Aug. 2005, he worked for OPNET Technologies, on (optical) network design and planning. In Sep. 2005, he re-joined INTEC as a postdoctoral researcher, and as a postdoctoral fellow of the FWO since Oct. 2006 (until 2012). In Oct. 2007 he obtained a part-time, and since Feb. 2010 a fulltime professorship at Ghent University. He has stayed as a research visitor at UC Davis (Jul.–Oct. 2007), CA, USA and at Columbia University, NY, USA (Jan. 2013–Jun. 2015). He was and is involved in various national and European research projects (e.g., IST David, IST Phosphorus, IST E-Photon One, BONE, FP7 Alpha, FP7 Geysers, FP7 Increase, FP7 C-DAX). Chris currently leads two research teams within IDLab, one on information retrieval and extraction, the other on smart grids. His research interests also still include optical networks (dimensioning, modeling, optimization, esp. for grid/cloud computing). He regularly serves as reviewer/TPC member for international journals and conferences. He is Senior Member of IEEE and Member of ACM.

Edmundo R. M. Madeira is a Full Professor at the University of Campinas (UNICAMP), Brazil. He received his Ph.D. in Electrical Engineering from UNICAMP in 1991. He has published over 150 papers in national and international conferences and journals. He was the General Chair of the 7th Latin American Network Operation and Management Symposium (LANOMS11), and he is a Technical Program Co-chair of the 15th IFIP/IEEE International Symposium on Integrated Network Management (IM17). He is a member of the editorial board of Journal of Network and Systems Management (JNSM), Springer. His research interests include cloud and edge computing, network management and future Internet.

Deep Medhi is Curators' Distinguished Professor in the Department of Computer Science Electrical Engineering at the University of Missouri-Kansas City, USA, and a honorary professor in the Department of Computer Science & Engineering at the Indian Institute of Technology–Guwahati, India. He received B.Sc. in Mathematics from Cotton College, Gauhati University, India, M.Sc. in Mathematics from the University of Delhi, India, and his Ph.D. in Computer Sciences from the University of Wisconsin-Madison, USA. Prior to joining UMKC in 1989, he was a member of the technical staff at AT&T Bell Laboratories. He served as an invited visiting professor at the Technical University of Denmark, a visiting research fellow at Lund Institute of Technology, Sweden, and State University of Campinas, Brazil. As a Fulbright Senior Specialist, he was a visitor at Bilkent University, Turkey, and Kurukshetra University, India. He is the Editor-in-Chief of Springer's *Journal of Network and Systems Management*, and serves (or served) on the editorial board of *IEEE/ACM Transactions on Networking, IEEE Transactions on Network and Service Management, IEEE Communications Surveys & Tutorials, Telecommunications Systems, Computer Networks*, and *IEEE Communications Magazine*. He has published over 150 papers, and is co-author of the books, *Routing, Flow, and Capacity Design in Communication and Computer Networks* (2004) and *Network Routing: Algorithms, Protocols, and Architectures* (1st edition in 2007, and 2nd edition in 2017), published by Morgan Kaufmann Publishers, an imprint of Elsevier Science. His research interests are: multi-layer networking; network virtualization; data center optimization; network routing, design, and survivability; and video quality-of-experience. His research has been funded by NSF and DARPA.