Realization of congestion-aware energy-aware virtual link embedding

Minh Pham
University of Technology Sydney
Sydney, Australia
minh.pham@student.uts.edu.au

Zenon Chaczko
University of Technology Sydney
Sydney, Australia
zenon.chaczko@uts.edu.au

Doan B. Hoang
University of Technology Sydney
Sydney, Australia
doan.hoang@uts.edu.au

***Abstract*— Network virtualization is an inherent component of future internets. Network resources are virtualized and provisioned to users on demand. The virtual network embedding entails two processes: node mapping and link mapping. However, efficient and practical solutions to the link mapping problem in software-defined networks (SDN) and data centers are still lacking. This paper proposes a solution to the link mapping process that can dynamically interact with the routing protocols of the substrate network to allocate virtual link requests to the underlying substrate links and satisfies optimizing cost, minimizing energy consumption, and avoiding congestion (CEVNE LiM) concurrently. CEVNE LiM is realized as a composite application on top of the SDN controller running the Segment Routing (SR) application. The performance of the CEVNE LiM algorithm is compared with the k-shortest path (link mapping) algorithm and shows its superior performance in terms of the overall runtime, the average path length, the average node stress, the average link stress and the overall energy consumption.**

Keywords—virtual link embedding, SDN application, cost saving, energy saving, congestion avoidance, segment routing

I. Introduction

Network virtualization is considered the future of the Internet. Originally it was conceived for evaluating new network protocols in the research-lab testbeds such as G-Labs [1] or 4WARD [2]. Recently, it has become an integral part of cloud data centers and telecommunication infrastructures. The main challenge in the network virtualization is the optimal allocation of the substrate resources to the on-demand virtual networks, and it is called the virtual network embedding (VNE) challenge. The VNE objectives are often to minimize the costs, maximize the revenues, optimize the network performance (delays, bandwidths, throughputs, QoS), and minimize the energy consumption.
The VNE resource allocation problem necessitates two processes: node mapping and link mapping. The node mapping process maps virtual nodes onto substrate nodes that satisfy the virtual nodes' resource demands. The link mapping process maps virtual links between virtual nodes onto substrate links connecting the embedded substrate nodes. In the traditional VNEs, the link mapping process is passive as it retrieves the shortest path between two end-points [3]. Existing VNE, passive link mapping process are not satisfactory in the real production environments that require objectives such as network load balancing, overall energy consumption, or network congestion avoidance. This paper proposes the CEVNE LiM solution on the leaf-spine fabric of a cloud data center with three concurrent objectives optimizing the cost, minimizing the energy consumption, and avoiding network congestion over an SDN substrate running SR protocol. The evaluation shows that the CEVNE LiM algorithm prevails the k-shortest path algorithm in achieving all three objectives: the cost saving, the energy saving and the congestion avoidance. CEVNE LiM can be leveraged in the virtual network function provisioning and placement in network function virtualization (NFV), service function chaining (SFC). CEVNE LiM can be extended to be the functional component of the reusable function block (RFB) specialized in LiM.
The paper is organized as follows. Section 1 is the introduction, section 2 presents the related work, section 3 presents the CEVNE LiM problem statement, section 4 presents the proposed architecture to realize CEVNE LiM process, section 5 presents the CEVNE LiM algorithm and realization, section 6 presents the evaluation, discussion and future works, and section 7 concludes the paper.

II. Related Work

Zahavi, et al. [4] proposed Links as a Service (LaaS) in data centers as a mechanism to isolate tenants' virtual networks. The LaaS architecture is built on top of the cloud architecture, which consists of a client front-end, a scheduler and a network controller. The evaluation showed that LaaS provides full tenant isolation at 10% reduced cost. However, the proposal did not consider the energy saving and congestion avoidance objectives.
Marotta, et al. [5] proposed an energy efficient and robust Virtual Network Function (VNF) placement. The Green Robust VNF Placements (GRVP) is a heuristic solution with three separate steps: the VNF Chain (VNFC) placements, the robust heuristics, and the latency constraint flow routing. It was shown that the solution took less than 1 second to place a flow in a large mobile network. However, the proposal did not consider cost saving and congestion avoidance objectives.
Chowdhury, et al. [6] proposed the Reallocation of Virtual Network Embedding (ReVINE) to eliminate the bottlenecks in the VNE. The proposal consists of the MIP formulation for the optimal ReVINE to balance between the bottleneck of substrate links and the total bandwidth consumption of the virtual networks, and a heuristic algorithm based on simulated annealing. The performance evaluation shows that ReVINE mitigates the substrate link bottlenecks. However, the proposal did not consider the cost saving and energy saving objectives.
Li, et al. [7] proposed an energy aware management technique for the leaf-spine fabric in cloud data centers. The proposal keeps track of dynamic workload, especially on spine switches to enable switches that are necessary for the current workload. However, the proposal did not consider cost saving and congestion avoidance objectives.
All the above link mapping algorithms are designed to work on a single objective, a multiple-objective virtual link mapping is required.

III. CEVNE Virutal Link Embedding Problem Statement

A. Problem statement

Given the SN with a set of N_S substrate nodes and E_S of substrate links, and the set of substrate paths P_S with N substrate nodes and M substrate links:

$$SN = \{N_S, E_S, P_S\}$$

Each substrate node $m \in N_S$ has a CPU resource. Each substrate link $(m, n) \in E_S$ has a bandwidth resource. A given VNR demands N_V virtual nodes and E_V virtual links:

$$VN = \{N_V, E_V\}$$

Each virtual node $u \in N_V$ has a CPU demand. Each virtual link $(u, v) \in E_V$ has a bandwidth demand.
Given the virtual network embedding process consisting of two separate processes: node embedding and link embedding processes. The node embedding process has been implemented successfully, and provided the results of substrate nodes that virtual nodes are embedded on.

B. Objectives

The VNE process needs to implement the mapping of E_V virtual links on top of substrate paths of the substrate network SN that satisfy the bandwidth demands, and three objectives concurrently:

- minimizing the cost of bandwidth resources used in virtual link embedding
- minimizing the network congestion in the substrate networks and virtual networks
- minimizing the total energy consumption

Additionally, these objectives are subject to

- the bandwidth capacity of the substrate links
- the topology of the substrate network, virtual network
- the results of the node embedding process

C. Problem formulation

The objective function and constraints are modelled into a mathematical program (MP) as follows, $h^i_{u,v}$ is the portion of the bandwidth of substrate link (m, n) allocated to virtual link i:

$$\min\left(\sum_{p_{m,n}}\sum_{i}\sum_{(u,v)\in p_{m,n}} h^i_{u,v}\right) \tag{1}$$

$$h^i_{u,v} \geq bw^i_{u,v}\ , \forall (u, v) \in E_S, \forall i \tag{2}$$

$$\sum_{i}\sum_{(u,v)\in p_{m,n}} h^i_{u,v} \leq r * bw(m, n),\ \ \forall \left(p_{m,n}\right) \in \mathcal{P}_S, \forall i \tag{3}$$

$$0 < r \leq R \tag{4}$$

The objective function (1) attempts to minimize the bandwidth resources allocated to all virtual links in E_V of the VNR, each virtual link is allocated the bandwidth resource $h^i_{u,v}$ on all the substrate links $(u, v) \in E_S$ of the substrate path $p_{m,n} \in \mathcal{P}_S$ having two ends (m, n) where the virtual nodes of virtual link i are embedded on. (1) is also used to minimize the total energy consumption, which is proportional to the length of the substrate path that the virtual link is embedded on, the shorter the path, the less number of active nodes. The constraint expressed in (2) ensures that the allocated bandwidth resource satisfying the virtual link bandwidth demand $bw_{u,v}$ of the virtual link i. The constraint expressed in (3) ensures that the total bandwidth resource allocated to virtual links on substrate link (u, v) of substrate path $p_{m,n}$ is limited to the value of substrate link bandwidth multiplied by the congestion ratio r. In the constraint expressed in (4), r is the congestion ratio to set a limit on the substrate link usage to minimize the maximum link utilization; r has an upper limit of R, which is calculated in advance based on the hose traffic demand model [8].

D. Motivation for a heuristic

The virtual link embedding is NP-Hard as it is reduced to the un-split-table flow problem [3], therefore, a heuristic algorithm is required to find a solution that satisfies all three objectives. The greedy approach could not provide the solution that can integrate with network monitoring in real-time for the optimal operation requirement, nevertheless, applying SDN paradigm may allow the integration with any monitoring applications based on SDN attributes of programmability, central control. Additionally, SDN-based approach will allow the link embedding algorithm to select the appropriate substrate paths based on SDN ability to steer the network flows at fine-grained and coarse-grained levels, SDN central control and central view. Therefore, the link embedding algorithm will be realized as an SDN application on top of SDN controller.

IV. PROPOSED ARCHITECTURE TO REALIZE CEVNE LIM

CEVNE LiM heuristic algorithm will embed VNR's virtual links on substrate paths satisfying three objectives concurrently for optimal network operation requirement. CEVNE LiM heuristic will be based on the micro-service architecture for extended SDN application [9], which facilitates new service creation, service composition in a three-tier architecture with UI tier, business tier and database tier.
The architecture to realize CEVNE LiM process is depicted in Fig. 1. The three-tier architecture of CEVNE LiM stays in the application layer of the SDN controller, in which the UI tier consists of REST API and CLI, the business tier consists of new services for cost saving, energy saving and congestion avoidance; the database tier consists of link residual capacities.

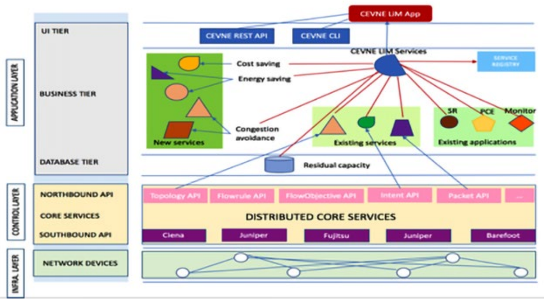

Fig. 1: The architecture to realize CEVNE LiM application

A. Services to handle link embedding multiple objectives

In this section, MSs are proposed for each CEVNE LiM objective cost saving, energy saving and congestion avoidance. The monitoring application provides real-time status to exclude substrate paths, which are alerted as congested. The path selection algorithm selects the substrate path based on objectives to embed the virtual link.

1) Services implementing cost saving, energy saving, congestion avoidance

a) Cost saving:

The selected path satisfying the cost saving objective is the shortest path between two end-points. The shortest paths can be retrieved using path management API core service, or the function equal cost multi path (ECMP) shortest path graph (SPG) in SR application. Fig. 2 presents an example of the path satisfying the cost saving objective, in which the shortest paths between two end-points A and B are A-S1-B or A-S2-B. The other paths between A and B are A-S1-C-S2-B, A-S2-D-S1-A, which are not shortest paths.

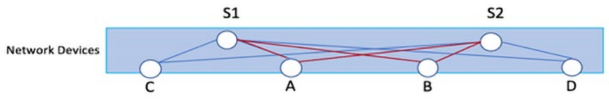

Fig. 2: the example of the cost saving between A and B.

b) Energy saving

The energy saving objective is implemented based on the resilience design of the leaf-spine fabric to steer traffic flows to the fall-back spine in case of network failure. There are links from reserved spines to leaf nodes. To save energy, the reserved spines and their links are put into sleep mode, they will be turned into active mode in case of emergency. Fig. 3 presents an example of a lead-spine fabric with a resilience design, spine S3 is the reserved one, S3 and all its links S3-A, S3-B, S3-C, S3-D, S3-E are put in sleep mode to save energy.

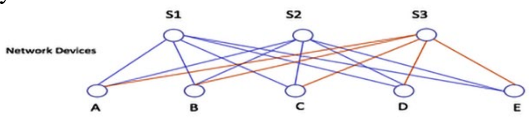

Fig. 3: an example of energy saving of leaf-spine fabric.

c) Congestion avoidance

The congestion avoidance objective is implemented by spreading the traffic evenly to all active network devices. The dynamic resource allocation algorithm to minimize stress and to prevent fragmentation [10] is used to select the least usage substrate path. Initially, the fabric is symmetric. After provisioning some VNRs, the resources are allocated to virtual nodes and links, their residual resources are different for each device. The substrate links' residual resources are stored in the database. The congestion-aware algorithm retrieves the fabric's residual resources to select switches having the largest residual resource in the active spine list, which may not be the shortest paths.

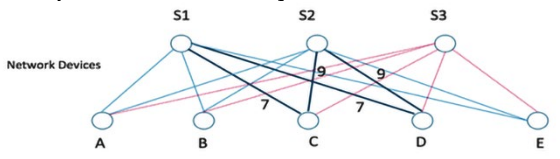

Fig. 4: an example of a congestion avoidance in leaf-spine fabric

In Fig. 4, the leaf-spine fabric has 3 spine nodes and 5 leaf nodes, spine S3 is in sleep mode. The paths between C and D are C-S2-D and C-S1-D, to avoid congestion, path C-S2-D is selected because of highest residual capacity, path C-S3-D is not counted as it is in sleep mode.

2) Monitoring application

Applying one network monitoring method as specified in [11], OpenNetMon provided an end-to-end measurement of throughput, packet loss, latency and link utilization in the SDN network, or applying the In-band network telemetry [12] using P4 language to monitor real-time network operation. The monitor application is used to watch for highly-utilized substrate paths (nearly congestion), which will be excluded from future virtual link embedding. The results of congestion avoidance and the monitoring application will complement each other.

3) Path selection strategy

The path selection algorithm (PSA) receives the two end-points as input, provokes the services to implement cost saving, energy saving and congestion avoidance to receive the results of each objective. The PSA selects the substrate path using the steps: (i) the energy saving result covers the cost saving result, (ii) the congestion control result and monitoring result should match, the lowest residual link will be alerted of potential congestion in the monitoring application, (iii) the PSA result should be the intersection between the cost, energy saving and the congestion control and monitoring results. (iv) If the PSA gives no result, the congestion control result will be selected. (v) If the PSA gives multiple results, one is selected randomly. The PSA is designed to be a simple decision algorithm as it needs to respond quickly in a high-speed fabric. With the PSA, the selected substrate path is ensured to satisfy all objectives.

To illustrate how the PSA works, Fig. 5 presents an example of the leaf-spine fabric of four spine nodes and seven leaf nodes. The two end points are C and E, the cost saving objective results in substrate paths C-S1-E, C-S2-E, C-S3-E, and C-S4-E; the energy saving objective results in substrate paths C-S1-E, C-S2-E, C-S3-E; the congestion avoidance objective results in the descending order based on the residual link capacity C-S2-E, C-S3-E, and C-S1-E; as the substrate path C-S1-E is alerted as a congestion path by the monitoring application, the substrate path for the virtual link embedding between node C and E is C-S2-E.

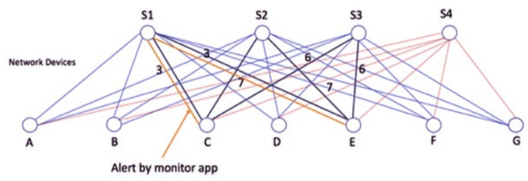

Fig. 5: example of PSA in the leaf-spine fabric

V. CEVNE LINK EMBEDDING ALGORITHM AND REALIZATION

A. CEVNE link embedding algorithm

The input for CEVNE LiM algorithm are the virtual network request (VNR), the substrate node end-points that virtual nodes are embedded on, the bandwidth demands of virtual links. LiM algorithm will call the PSA to select the substrate paths for each virtual link, which satisfy three objectives and not in congestion state according to the monitoring application. LiM repeats the process until all virtual links are embedded, or an error occurs and the whole VNR is terminated.

Algorithm: CEVNE LiM algorithm

```
1   Input:
2       virtual network request
3       substrate node end-points that virtual nodes are
    embedded on
4       Virtual link bandwidth demands
5       reserved spine nodes $spine_{res}$ // from network operators
6   Output:
7       substrate path for each virtual link
8   For each virtual link $e_v$ in $E_V$
9      (m,n) = embedded end-points of $e_v$;
10      $path_s$ = PSA(m,n);
11      If $path_s$ is empty
12          Return;
```

```
        End if
        IntentMgr.submit(new intent($path_s$));
    End For
    Function PSA(m,n)
        {p1} = cost_saving(m,n) ∩ energy_saving(m,n);
        {p2} = congestion_avoidance(m,n);
        {$p_r$} = ({p1} ∩ {p2})
        If {$p_r$} is empty
            Return $p_{res}$ ∈ {p2};
        else
            Return $p_{res}$ ∈ {$p_r$};
        End if.
    End function
    Function cost_saving(m,n)  // applying the shortest path between (m,n)
        {$p_{cost}$} = ∅;
        For each p ∈ PathAPI.shortest_path(m,n);
            If p.residual ≥ $e_v$.bwdemand
                {$p_{cost}$}.add(p);
            End if
        End for
        Return {$p_{cost}$}
    End function
    Function energy_saving(m,n)  // applying on the cost saving results
        {$p_{energy}$} = cost_saving(m,n).excludes($spine_{res}$);
        Return {$p_{energy}$}
    End function
    Function congestion_avoidance(m,n)  // applying on all paths between (m,n)
        {$p_{temp}$} = PathAPI.get_paths(m,n);
        $p_{t1}$ = "";
        For each $p_t$ ∈ {$p_{temp}$}
            If ($p_t$.residual ≥ $e_v$.bwdemand)
                If ($p_{t1}$ == "")
                    $p_{t1}$= $p_t$;
                Else if ($p_t$.residual > $p_{t1}$.residual)
                    $p_{t1}$= $p_t$;
                End if
            End if
        End for
        Return $p_{t1}$
    End function
```

B. CEVNE LiM application realization

CEVNE LiM algorithm is developed into an SDN composite application [9] called CEVNE LiM application that is on top of the ONOS controller in a leaf-spine fabric SDN substrate network running SR application. The leaf-spine fabric is configured in ONOS controller and is presented in the ONOS UI as in Fig. 6.

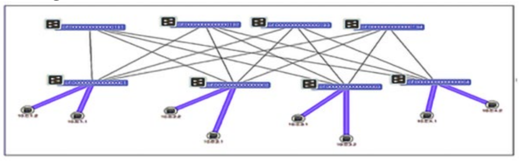
Fig. 6: Substrate network configured as leaf-spine fabric in ONOS UI

VI. PERFORMANCE EVALUATION

The CEVNE LiM performance evaluation includes test scenarios setup, running CEVNE application and analyzing results. The next subsection describes the test scenario setup.

A. Test scenario setup

The test scenario includes a substrate network and multiple virtual networks that are to be virtualized and provisioned over the substrate. The substrate network is a leaf-spine fabric with four leaf switches, four spine switches, eight hosts, and links connecting switches and hosts as in Fig. 7. The leaf, spine switches, which are white boxes running open-source software, are configured to run SR protocol on ONOS controller as specified in Atrium [13]. The substrate network and their capacities are specified in TABLE 1.

TABLE 1: Substrate network capacities

	Total number	CPU capacity	BW capacity
Leaf switches	4	1	
Spine switches	4	1	
Hosts	8	100	
Link leaf-spine	16		10
Link leaf-host	8		1

In TABLE 1, hosts have large CPU capacities, they are ready to have virtual nodes embedded on. Links between leaf and spine switches have high bandwidth capacity, they are ready for virtual links to be embedded on.

We design different test cases with various VNRs' requirements to test the performance of CEVNE LiM in terms of runtime, average path length, average node stress, average link stress, energy consumption, acceptance ratio and total costs / revenue. VNRs in test cases are different in topologies and resource demands as specified in TABLE 2.

TABLE 2: The design of test scenarios

Test cases	No of virtual nodes	CPU demands	No of virtual links	BW demands
1	3	10	2	3
2	5	10	4	3
3	4	10	3	3
4	5	10	4	3
5	4	10	3	3

In TABLE 2, there are 5 different VNRs, with the number of requested virtual nodes ranges from 3 to 5, and the number of requested virtual links ranges from 2 to 4.

B. CEVNE execution platform

The CEVNE LiM application is compiled, deployed and activated on the ONOS controller as CEVNE virtual link service. CEVNE LiM application implements PSA energy-saving algorithm, cost-saving algorithm and congestion-aware algorithm. The CEVNE LiM is invoked via the CLI or the REST API. Each virtual link detail is input as a tuple of the source and destination host locations, and the bandwidth demand. The result of the CEVNE LiM application runs on the ONOS controller is shown in the Fig. 7.

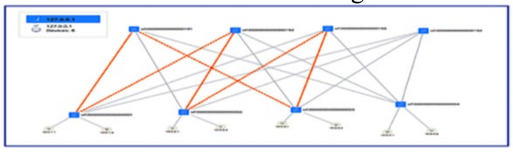
Fig. 7: CEVNE LiM application execution in ONOS UI

In Fig. 7, the red-dashed lines are the selected substrate paths, where the virtual links are embedded on. On the substrate paths, the flow rules are installed on leaf and spine switches to establish the route. They are highlighted as the result of the flow rule creation for each substrate path. The CEVNE LiM execution in the ONOS CLI for the VNR in Fig. 7 is summarized in TABLE 3.

TABLE 3: Results of CEVNE LiM when running a test case

	Virtual links	*Node mapping results*	*CEVNE LiM results*
1	**AB**	host 1, host 2	The hosts are connected to leaf 01.
2	**BC**	host 2, host 3	onos:cevne-get-mapped-paths of: 001 of: 002 5000000;DefaultPath{src=of: 192/2, dst=of: 192/1,

			type=INDIRECT, state=ACTIVE, expected=false, links=[DefaultLink{src=of: 192/2, dst=of: 002/3, type=DIRECT, state=ACTIVE, expected=false}, DefaultLink{src=of: 001/3, dst=of: 192/1, type=DIRECT, state=ACTIVE, expected=false}], cost=ScalarWeight{value=0.0}}
3	**AE**	host 1, host 5	onos:cevne-get-mapped-paths of: 001 of: 003 5000000;DefaultPath{src=of: 191/3, dst=of: 191/1, type=INDIRECT, state=ACTIVE, expected=false, links=[DefaultLink{src=of: 191/3, dst=of: 003/4, type=DIRECT, state=ACTIVE, expected=false}, DefaultLink{src=of: 001/4, dst=of: 191/1, type=DIRECT, state=ACTIVE, expected=false}], cost=ScalarWeight{value=0.0}}
4	**ED**	host 5, host 4	onos:cevne-get-mapped-paths of: 003 of: 002 5000000;DefaultPath{src=of: 193/2, dst=of: 193/3, type=INDIRECT, state=ACTIVE, expected=false, links=[DefaultLink{src=of: 193/2, dst=of: 002/2, type=DIRECT, state=ACTIVE, expected=false}, DefaultLink{src=of: 003/2, dst=of: 193/3, type=DIRECT, state=ACTIVE, expected=false}], cost=ScalarWeight{value=0.0}}

In TABLE 3, the CLI command cevne-get-mapped-paths invokes the virtual link mapping for virtual links AB, BC, AE, and ED.

Row 1: The virtual link AB is the link between host 1 and host 2, which is within the leaf 01, they are connected in the VXLAN connecting the ToR switch (leaf 1) and the hosts.

Row 2: The virtual link BC is between host 2 and host 3, which is the path between leaf 01 (host 2 location) and leaf 02 (host 3 location). The path between leaf 01, and spine 02 (port 1), spine 03 (port 2) and leaf 02 is selected. Data in row 3 and row 4 are explained similarly as in row 2.

C. *Evaluation results*

The CEVNE LiM performance is evaluated by analyzing its implementation results and comparing them with those of the k-shortest path link-mapping algorithm [14]. The metrics used to compare the behavior of the two algorithms are the total runtime, acceptance ratio, average path length, average node stress, average link stress, and energy consumption. These metrics are selected because they reflect the three CEVNE objectives: cost saving, energy saving and congestion awareness. The following subsections will examine each of the results.

1) *Runtime*

The runtime results show that CEVNE LiM converged faster than the k-shortest path algorithm. This explains the simple path selection algorithm of CEVNE LiM. In the Alevin framework, the runtime includes the configuration time for each test run. In ONOS controller, the configuration is run once, and its runtime is calculated separately. It takes the CEVNE LiM application about 3ms to embed each virtual link; and the total runtime is the multiplication of the number of virtual links in the VNR. Fig. 8 shows that k-shortest path runtime is in the range of 60–70ms for the five test cases, while CEVNE LiM runtime is in the range of 10–30ms; 30ms is when the CEVNE LiM application is first loaded. This shows that the CEVNE LiM achieves it cost saving objective with the quick response time.

2) *Average path length*

The average path length results show that the CEVNE LiM approach always have the shorter substrate path length compared to the k-shortest path algorithm. This is the result of the CEVNE LiM application searching for the shortest paths between two end-points. The k-shortest path approach configures the fabric as a normal bi-directed graph, and searches in the whole graph for each virtual link. This affects both the path length and the runtime.

In Fig. 8, the CEVNE LiM algorithm has 2.0 as the average path length for any substrate paths that the virtual link is embedded on. The k-shortest path algorithm has the average path length in the range of 3.0–4.0. This shows that the CEVNE LiM achieves its cost saving objective, and its energy saving objective because the longer the path, the more active substrate nodes and more energy consumption.

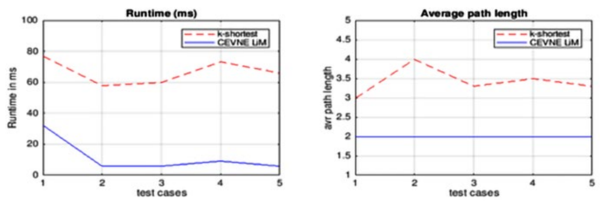

Fig. 8: Results of runtime and average path length

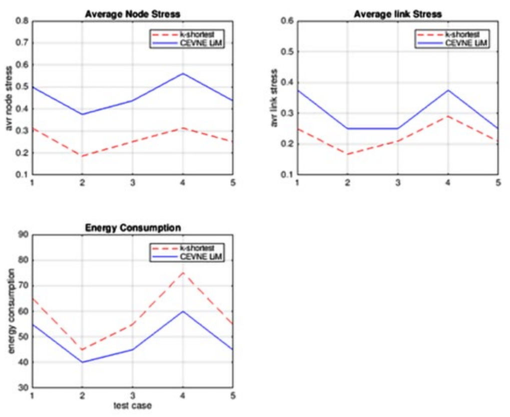

Fig. *9*: Results of average node stress, link stress, and energy consumption.

3) *Average node stress, average link stress*

In the Alevin framework, the average node stress is the ratio of total of all node stress over the total number of substrate nodes. Similarly, the average link stress is the ratio of total of all link stress over the total substrate links. These are similar to the average node utilization and average link utilization introduced in [15].

In CEVNE, the average node stress and average link stress are calculated for leaf and spine switches on the fabric. If each virtual node of a virtual link is connected to a different leaf switch, the average node stress involves three nodes: two leaf switches and one spine switch. The average link stress involves two substrate links: links between each leaf and the spine switches. For congestion control purposes [16], the link mapping is spread out to different leaf and spine switches as implemented in the path selection algorithm. The average node stress and the average link stress results show that the CEVNE LiM algorithm distributes the link mapping onto more substrate nodes and links than the k-shortest path algorithm.

In Fig. 9, the k-shortest path algorithm has the average node stress in the range of 0.2–0.3 while the CEVNE LiM's average node stress is in the range 0.38–0.55. The k-shortest path algorithm has the average link stress in the range of 0.18–0.3 while the CEVNE LiM's average link stress is in the range 0.25–0.38. The CEVNE LiM has on average 0.2 (20%) more node stress than the k-shortest path approach. The

CEVNE LiM has on average 0.1 (10%) more link stress than the k-shortest path approach.

4) Energy consumption

The energy consumption results show that CEVNE LiM algorithm achieves its energy saving objective as specified in 2) average path length. As CEVNE LiM has shorter average path length than the k-shortest path algorithm, CEVNE LiM has less number of active nodes along the paths, resulting in less power consumption. In Fig. 9, the energy consumption of CEVNE LiM is 10 units lower on average compared to the k-shortest path algorithm.

5) Acceptance ratio

The acceptance ratios of both algorithms are the same, and at the 100% in all test cases. The results of runtime, average path length, average node stress, average link stress, and energy consumption show that CEVNE LiM algorithm achieves three objectives, and prevails the k-shortest path algorithm in these objectives.

D. Discussion

CEVNE LiM can be applied to NFVI, SFC, can be considered as the functional component of RFB specialized in LiM. The virtual link embedding process is a necessary step in the virtual network function provisioning, placement in NFV, SFC; NFVI is also SDN-based, hence, the CEVNE LiM can be leveraged in these processes, in which the bandwidth requirement is specified as a latency constraint, to offer cost saving, energy saving, congestion avoidance objectives. CEVNE LiM can be extended to be the functional component of the RFB specialized in LiM [17].

CEVNE LiM focuses on three objectives concurrently on the SDN network. The CEVNE LiM application realizes each objective as a micro-service and implements the path selection algorithm to search for the selected path based on each objective result. The evaluation shows that CEVNE LiM achieves its three objectives of cost saving, energy saving and congestion aware.

The choice of SDN and SR technologies, and SDN-SR based leaf-spine fabric as the substrate network. The CEVNE LiM chooses the SN that is configured as a leaf-spine fabric using the SR application, which is the next generation fabric, where the automation is introduced via the programmability and service composition on top of the SDN controller. The CEVNE LiM approach is based on SDN paradigm that realizes it as SDN applications that are modular, composable, and extendable [18].

VII. CONCLUSION

CEVNE link mapping algorithm is proposed to minimize the cost, the energy and to avoid the network congestion in allocating substrate link resources to virtual links. Based on SDN and SR technologies, the CEVNE LiM is an active virtual link mapping process that can interact with the routing rules on switches to select substrate paths for the virtual link mapping. It is completely different from the passive link mapping algorithms in the traditional VNE. The CEVNE LiM receives the node mapping results from the CEVNE NoM algorithm and the VNR topology as inputs and searches the substrate paths that satisfy the virtual links' bandwidth demands and objective constraints based on its path selection algorithm. In the evaluation, the CEVNE LiM algorithm is compared with the k-shortest path algorithm in the Alevin framework. The results of both approaches are analyzed according to the runtime, average path length, average node stress, average link stress, and energy consumption. The evaluation shows that the CEVNE LiM algorithm prevails the k-shortest path algorithm in achieving all three objectives: the cost saving, the energy saving and the congestion avoidance. CEVNE LiM can be applied to NFVI, SFC, can be considered as the functional component of the RFB specialized in LiM.

REFERENCES

[1] D. Günther, R. Henjes, B. Reuther, and P. Müller, "German-lab experimental facility.," presented at the Future Internet-FIS 2010, 2010.

[2] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization.," *Computer Networks,* vol. 54, no. 5, p. 15, 2010.

[3] A. Fischer, J. F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *Communications Surveys & Tutorials, IEEE,* vol. 15, no. 4, p. 19, 2013.

[4] E. Zahavi, A. Shpiner, O. Rottenstreich, A. Kolodny, and I. Keslassy, "Links as a Service (LaaS): Guaranteed tenant isolation in the shared cloud," in *Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems*, 2016, p. 12: ACM.

[5] A. Marotta, E. Zola, F. D'Andreagiovanni, and A. Kassler, "A Fast Robust Optimization-based Heuristic for the Deployment of Green Virtual Network Functions. ," *Journal of Network and Computer Applications,* vol. 95, p. 12, 2017.

[6] S. R. Chowdhury *et al.*, "Revine: Reallocation of virtual network embedding to eliminate substrate bottlenecks," presented at the IEEE/IFIP Integrated Network Management Symposium (IM). 2017.

[7] X. Li, C. Lung, and S. Majumdar, "Green spine switch management for datacenter networks," *Jounal of Cloud Computing: Advances, Systems and Applications* vol. dec 2016, p. 19, 2016.

[8] E. Oki, *Linear programming and algorithms for communication networks: a practical guide to network design, control, and management.* CRC Press., 2012

[9] M. Pham and D. B. Hoang, "SDN applications-The intent-based Northbound Interface realisation for extended applications," presented at the NetSoft Conference and Workshops, Seoul, Korea, 2016.

[10] R. Mijumbi, J. Serrat, J. Rubio-Loyola, N. Bouten, F. De Turck, and S. Latre, "Dynamic resource management in SDN-based virtualized networks," in *Network and Service Management (CNSM), 2014 10th International Conference on*, 2014, pp. 412-417.

[11] P. Tsai, C. Tsai, C. Hsu, and C. Yang, "Network monitoring in software-defined networking: a review," *IEEE Systems,* vol. 12, no. 4, p. 12, 2018

[12] C. Kim *et al.* (2016, July 5th). *Inband network telemetry (INT) specification*. Available: http://p4.org/wp-content/uploads/fixed/INT/INT-current-spec.pdf

[13] S. Das. (2016). *Atrium*. Available: https://github.com/onfsdn/atrium-docs/wiki

[14] M. T. Beck, C. Linnhoff-Popien, A. Fischer, F. Kokot, and H. de Meer, "A simulation framework for virtual network embedding algorithms.," presented at the Telecommunications Network Strategy and Planning Symposium (Networks), 2014.

[15] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking (TON),* vol. 20, no. 1, p. 14, 2012.

[16] D. Medhi, Network routing: algorithms, protocols, and architectures. Morgan Kaufmann., 2010.

[17] A. Mimidis-Kentis *et al.*, "The Next Generation Platform as A Service: Composition and Deployment of Platforms and Services," *Future Internet,* 2019

[18] D. B. Hoang, "Software Defined Networking – Shaping up for the next disruptive step?," *Australian Journal of Telecommunications and the Digital Economy,* vol. 3, no. 4, p. 15, 20