A Scalable Resource Allocation Scheme for NFV: Balancing Utilization and Path Stretch

Y.T. Woldeyohannes*, Ali Mohammadkhan†, K.K. Ramakrishnan†, Yuming Jiang*
*Norwegian University of Science and Technology, NTNU, Trondheim, Norway
†University of California Riverside, California, USA

***Abstract*—Network Function Virtualization (NFV) implements network middlebox functions in software, enabling them to be more flexible and dynamic. NFV resource allocation methods can exploit the capabilities of virtualization to dynamically instantiate network functions (NFs) to adapt to network conditions and demand. Deploying NFs requires decisions for both NF placement and routing of flows through these NFs in accordance with the required sequence of NFs that process each flow. The challenge in developing NFV resource allocation schemes is the need to manage the dependency between flow-level (routing) and network-level (placement) decisions.**

We model the NFV resource allocation problem as a multi-objective mixed integer linear programming problem, solving both flow-level and network-level decisions simultaneously. The optimal solution is capable of providing placement and routing decisions at a small scale. Based on the learnings from the optimal solution, we develop ClusPR, a heuristic solution that can scale to larger, more practical network environments supporting a larger number of flows. By elegantly capturing the dependency between flow routing and NF placement, ClusPR strikes a balance between minimizing path stretch and maximizing network utilization. Our experiments show ClusPR is capable of achieving near-optimal solution for a large sized network, in an acceptable time. Compared to state-of-the-art approaches, ClusPR is able to decrease the average normalized delay by a factor of $1.2 - 1.6\times$ and the worst-case delay by $9 - 10\times$, with the same or slightly better network utilization.

I. Introduction

Middleboxes such as firewalls, VPN gateways, proxies, intrusion detection and prevention systems, etc., play a central role in today's Internet by providing network resident functionality that examines and potentially modifies the end-to-end traffic flow [1]. Implementation of network resident functionality is gradually migrating to software platforms, providing additional flexibility and extensibility for the capabilities of the network compared to purpose-built hardware appliances. Evolving the network's capabilities can thus involve lower capital expenditures as the software can run on commercial off-the-shelf (COTS) hardware. Network Function Virtualization (NFV) decouples the software of network functions from the physical machine and runs it on virtual machines, or more recently on "containers" [2]. This also brings greater flexibility in resource management as instances of the network functions (NFs) can be created dynamically, and the capacity for a particular function can be scaled up or down depending on traffic demand. Sequences of NFs are common, and the overall service provided by the network by such sequences of NFs is termed "service function chaining" [3].

A. The Challenge

Resource management of NF service chains continues to be a challenge because of the complexity involved. An NFV resource allocation (NFV-RA) mechanism has to make decisions at multiple levels to ensure resources are properly utilized, while performance requirements of flows are met. Resource allocation decisions have to be made both at the network-level and at the flow-level. At the network-level, the allocation algorithm determines the number of NF instances to instantiate in the network to process the flows. In addition, the algorithm needs to determine the placement of the NFs or the physical machines that should host the NFs. However, this network-level decision making has to be coupled with the decision making at the flow-level, as the flow has to be sequenced through the NF instances to form the desired service chain. The flow-level decision making includes the determination of the route for the flow based on the service chain requirement and the choice of the NF instances initiated in the network.

Suboptimal decisions, resulting in NFs placed on network nodes not along the shortest path will result in "path stretch", contributing to increased latency for the flow. In addition, it is important to use resources efficiently and avoid over provisioning of resources so as to maximize the utilization of the network. Finally, it is critical to consider both link capacity and node processing capacity (in terms of CPU cores) when making the resource allocation decisions. The *interdependence between network-level (placement, instantiating the requisite number of NFs) and flow-level (routing) decisions makes the NFV-RA problem new and challenging*, warranting the recent attention it has received in the literature.

There have been a number of papers on NFV-RA published in the recent past. A detailed survey on existing NFV-RA approaches can be found in [4]. Some have focused on NF placement alone [5], [6]. Some

recent works have tried to solve NF placement and flow routing jointly. One group of works seeks to minimize the path stretch (delay) while limiting the utilization of the network [7], [8] and thereby limiting capacity. Another seeks to increase capacity by allowing the network utilization to increase, but at the expense of higher delay [9], [10]. Additionally, some of the existing approaches consider only the node capacity constraint (e.g. the MILP formulation in [11] and [12]), ignoring link capacity constraints.

In general, these literature approaches have only addressed part of the NFV-RA problem, far from solving it completely, which demands balancing between path stretch and utilization, factoring all the key resources, and concurrently solving the NF placement and flow routing problem.

B. Our Contributions

Our first contribution is that we model the joint NF placement *and* flow routing problem as a multi-objective mixed integer linear programming (MILP) problem. *The model is able to allocate NF instances and find end-to-end routes of flows while maintaining the precedence constraint among NFs of a service chain*. The MILP is solved using conventional (CPLEX) solvers for a reasonable scale problem with realistic parameters. The results provide us valuable insights to develop a heuristic solution capable of solving the NFV-RA problem for larger scale in a reasonable time.

Secondly, we develop a heuristic-based NFV-RA scheme, ClusPR, which strikes a balance between minimizing the path stretch experienced by flows and maximizing the utilization of the network. ClusPR captures the dependency between the routing and placement decisions. Both our MILP model and ClusPR consider link and node capacity constraints in making their decision. Unlike most papers in the literature that restrict the placement of NFs in the Cloud [10], [13], *ClusPR can be used in a more general setting where NFs may be hosted not only in the cloud but also on the edge computing nodes.*

To the best of our knowledge, there is no existing work that is scalable, creates a balance between minimization of the path stretch and maximization of the utilization of the network and considers the various resource limitations as our proposed approach ClusPR.

In summary, the contributions of this paper are:

- A novel multi-objective MILP formulation for the joint NF placement and flow routing problem.
- ClusPR, a heuristic-based scalable resource allocation scheme that is developed taking as input the insights observed from the MILP's results.
- Results from a number of experiments with realistic topologies demonstrate the effectiveness of ClusPR.

II. The System Model

We consider a network of nodes and links, modeled as a directed graph, $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes in the network and $\mathcal{L}$ is the set of links interconnecting the nodes. A node can be, a data-center, a router or a commercial off-the-shelf (COTS) hardware together with a router. The network carries a set of flows, $\mathcal{F}$, and supports a set of NFs, denoted as $\mathcal{V}$. For each NF type, multiple instances may be instantiated on one or multiple nodes.

A node n is characterized by the number of cores at the node, denoted as K_n. *An NF instance can be hosted on any node that has enough number of cores*. An NF instance $v \in \mathcal{V}$ hosted on node n is characterized by its service rate of requests, μ_n^v. Here, we assume that the instances of the same NF type at the same node have the same processing capacities. Note that an instance of type v may need multiple cores, k^v. In addition, an NF instance can process multiple flows whose NF service chains include the NF of the instance. We use D_n^v to denote the expected nodal delay for type v NF at node n, consisting of both processing delay and queueing delay for flows with NF type v at the node.

Each link $l \in \mathcal{L}$ is assumed to be bi-directional, and we use $l_{n'}^n$ to represent the link from node n to n' and C_l its expected transmission rate of bits. Each node n has a set of outgoing links represented by $\mathcal{L}_n^{out}$ and a set of incoming links, $\mathcal{L}_n^{in}$. The expected delay on link l, that comprises transmission and propagation delays, is written as D_l.

A flow $f \in \mathcal{F}$ is a sequence of data packets that are generated at expected rate λ_f and sent from a source to a destination node, traversing a sequence of intermediate nodes and links in the network. Each flow f has a specified service chain of NFs, denoted as $\overrightarrow{S}_f = (S_f^1, S_f^2 ..., S_f^{J_f})$, which is an *ordered* sequence of required NFs that the flow's packets must go through, where $S_f^j \in \mathcal{V}$ denotes the j^{th} NF on flow f's service chain and $J_f := |\overrightarrow{S}_f|$ is the length of the NF chain of flow f. We assume that the NFs in a flow's service chain are different.

In this paper, in addition to the sequence of NFs to be followed for the service chain, each flow f also has an end-to-end delay requirement, denoted as D_f, between the source node s_f and the destination node d_f of the flow. The end-to-end delay is composed of two types of delays: total delay on links, denoted as D_T and total delay on nodes, denoted as D_P.

III. The NFV Resource Allocation Problem

As discussed earlier, in a network supporting NFV, resource allocation decisions should be made both at the network-level and at the flow-level. For the former, an NFV resource allocation mechanism needs to decide the number of NF instances to instantiate in the network

to process the flows and at which nodes in the network such NF instances should be placed. For the latter, the mechanism needs to decide how to route the flows to go through the NF instances according to the order of NFs in their service chains and at the same time to meet the flows' throughput and delay requirements.

A flow is admitted to the network if and only if there are NF instances that can serve all the services in the NF service chain of the flow without violating the flow's delay requirement. Indicator variable $I_n^v(f;j)=1$ denotes that an NF instance v hosted at node n is used by the j^{th} service on the service chain of flow f, and indicator variable $I_l(f;j,n_j;j+1,n_{j+1})=1$ denotes that link l is used by flow f to route from the j^{th} to $(j+1)^{th}$ NF service, hosted at node n_j and n_{j+1} respectively. Since more than one instance of the same NF type may be hosted at the same node, we use an integer decision variable y_n^v to represent the number of NF type v instances that are hosted at node n. Note that, while y_n^v is a network-level decision variable that decides the number and placement of NF instances, $I_n^v(f,j)$ and $I_l(f;j,n_j,j+1,n_{j+1})$ are flow-level decision variables that specify which NF instances will be used by a given flow and which links will be used to route the flow respectively.

For this NFV resource allocation problem, three objectives are of interest: *(1) to maximize the number of flows admitted to the network, (2) to minimize the use of nodal processing capacities or cores, and (3) to minimize the utilization of link capacities*, where (2) and (3) are purposed to maximally leave resources for future use. The objective functions can then be represented as:

$$\text{max.} \quad \sum_{\forall f\in\mathcal{F}}\left\lfloor \frac{\sum_{j=1}^{J_f}\sum_{\forall n\forall v} I_n^v(f;j)}{J_f}\right\rfloor \tag{1}$$

$$\text{min.} \quad \sum_{\forall n\forall v}\frac{y_n^v k^v}{K_n} \tag{2}$$

$$\text{min.} \quad \sum_{\forall l\forall f\forall n_j\forall n_{j+1}}\sum_{\forall j}^{J_f}\frac{I_l(f;j,n_j;j+1,n_{j+1})\lambda_f}{C_l} \tag{3}$$

The above three objective functions are combined into a single-objective function, using the traditional weighted sum method [14]. Since, maximizing a given function is equivalent to minimizing the negative of the function, the single-objective function is to minimize the summation of the objective functions (2), (3) and negative of (1). The three objective functions are weighted equally, with unit weights. For positive weights, the optimal solution of the single-objective representation is also a Pareto optimal solution of the multi-objective problem [14].

A. Constraints

In solving the MILP problem, a number of constraints must be satisfied, which can be classified as: *capacity constraints, delay constraints,* and *NF chaining constraints*.

The capacity constraints are to ensure that the total traffic rate on any link does not exceed the link's transmission capacity (i.e., Constraint (III-A)), the total number of cores allocated to NF instances at any node does not exceed the cores at that node (i.e., Constraint (5)), and the total processing capacity required for admitted flows for any NF instance does not exceed that instance's processing capacity (i.e., Constraint (6)):

$$\sum_{\forall f\forall n_j\forall n_{j+1}}\sum_{\forall j}^{J_f} I_l(f;j,n_j;j+1,n_{j+1})\lambda_f \le C_l \quad \forall l\in\mathcal{L} \tag{4}$$

$$\sum_{\forall v} y_n^v k^v \le K_n \quad \forall n\in\mathcal{N} \tag{5}$$

$$\sum_{\forall f}\sum_{\forall j}^{J_f} I_n^v(f;j)\lambda_f \le \mu_n^v \quad \forall v\in\mathcal{V},\forall n\in\mathcal{N} \tag{6}$$

Related also are two constraints implied by the definition of $I_n^v(f;j)$. One is that a flow is assigned to use NF instance of type v on node n only if there is at least one instance of NF type v hosted on node n. (i.e., Constraint (7)). Another is the constraint that any NF in the service chain of any flow is served at most by one such NF instance (i.e., Constraint (8)).

$$y_n^v \ge I_n^v(f;j) \quad \forall n\in\mathcal{N},\forall v\in\mathcal{V},\forall f\in\mathcal{F},\forall j\in J_f \tag{7}$$

$$\sum_{\forall n\forall v} I_n^v(f;j)\le 1 \quad \forall f\in\mathcal{F},\forall j\in J_f \tag{8}$$

The second category are delay constraints to ensure that a flow is admitted only if its end-to-end delay requirement is met (i.e., Constraint (3)):

$$D_T + D_P < D_f \quad \forall f\in\mathcal{F} \tag{9}$$

where $D_T = \sum_{\forall l\forall n_j\forall n_{j+1}}\sum_{\forall j}^{J_f} I_l(f;j,n_j;j+1,n_{j+1})D_l$ and $D_P = \sum_{\forall j}^{J_f}\sum_{\forall v}\sum_{\forall n} I_n^v(f;j)D_n^v$ are respectively the total link delay and the total nodal delay that the admitted flow f experiences in the network.

The third category is NF chaining constraints, which are used to ensure that the order of NFs of any flow is followed when it is routed from its source node, through NF instances of its service chain and finally to its destination node. As shown below, several constraints, i.e., (10) – (16), are involved in this category. In these constraints, variable j is used to represent the service order, i.e., $j=1$ represents the first service, $j=2$ the second service, and so on in an NF service chain. For the

source and the destination we use $j = 0$ and $j = J + 1$, respectively.

We start with Constraints (10) and (11) that can be regarded as the flow conservation equations for the set of nodes that host NF instances assigned to serve a flow. Specifically, Constraint (10) ensures that one of the *outgoing links of node* n_j that is running the j^{th} service of flow f has to be assigned for routing flow f from its j^{th} to $(j+1)^{th}$ service order. The placement decision variables $I^v_{n_j}(f;j)$ and $I^v_{n_{j+1}}(f;j+1)$ are multiplied to make sure that the constraint applies to the cases where node n_j is used for the j^{th} service and node n_{j+1} is utilized for the $(j+1)^{th}$ service. Similarly, Constraint (11) makes sure that one of the *incoming links of node* n_{j+1} that runs $(j+1)^{th}$ service of flow f has to be assigned for routing flow f from its j^{th} to $(j+1)^{th}$ service. We remark that both (10) and (11) are not linear but since they are multiples of binary variables they can easily be substituted by a set of linear equations.

$$\forall f \in \mathcal{F}, \forall j \in 1, ..J-1, \forall n_j, n_{j+1} \in \mathcal{N}:$$

$$\sum_{l \in \mathcal{L}^{out}_{n_j}} I_l(f; j, n_j; j+1, n_{j+}) I^v_{n_j}(f;j) I^v_{n_{j+1}}(f; j+) = 1 \tag{10}$$

$$\sum_{l \in \mathcal{L}^{in}_{n_{j+}}} I_l(f; j, n_j; j+1, n_{j+}) I^v_{n_j}(f;j) I^v_{n_{j+1}}(f; j+) = 1 \tag{11}$$

Constraint (12) is a flow conservation constraint of the *intermediate nodes* that do not host NF instances for the flow but still should route the flow in a given service order path. It makes sure that, if one of the incoming links of a node is assigned for a given service order then one of the outgoing links of the same node has to be assigned to the same service order.

$$\forall f \in \mathcal{F}, \forall j \in 0, ..J, \forall n \in \mathcal{N}/n \neq n_j, n_{j+1}:$$

$$\sum_{l \in \mathcal{L}^{in}_{n}} I_l(f; j, n_j; j+1, n_{j+1}) - \sum_{l \in \mathcal{L}^{out}_{n}} I_l(f; j, n_j; j+1, n_{j+1} = 0, \tag{12}$$

Constraints (13) and (14) are source node flow conservation constraints. Constraint (13) ensures that one of the outgoing links of the *source node* of flow f is used to route from the 0^{th} to 1^{st} service of the service chain. As defined earlier, the source node represents the 0^{th} service. That is, node n_0 is indeed s_f, the source node of flow f. In addition, Constraint (14) is used to assign one of the incoming links of a node (n_1) serving the 1^{st} service of the service chain.

$$n_0 = s_f, \forall n_0, n_1 \in \mathcal{N}, \forall f \in \mathcal{F}:$$

$$\sum_{l \in \mathcal{L}^{out}_{n_0}} I_l(f; 0, n_0; 1, n_1) = I^v_{n_1}(f;1) \tag{13}$$

$$\sum_{l \in \mathcal{L}^{in}_{n_1}} I_l(f; 0, n_0; 1, n_1) = I^v_{n_1}(f;1) \tag{14}$$

Constraints (15) and (16) are flow conservation constraints for the destination node. Constraint (15) makes sure that one of the incoming links of the *destination node* is assigned to route from the node hosting the last NF of the service chain (n_J) to the destination node (d_f) that is also represented as the $J+1$ service, as defined earlier. Constraint (16) completes the route by assigning one of the incoming links of the destination node to the last J to $(J+1)^{th}$ service order path.

$$n_{J+1} = d_f, J = J_f, \forall n_J, n_{J+1} \in \mathcal{N}, \forall f \in \mathcal{F}:$$

$$\sum_{l \in \mathcal{L}^{in}_{n_{J+1}}} I_l(f; J, n_J; J+1, n_{J+1}) = I^v_{n_J}(f, ; J) \tag{15}$$

$$\sum_{l \in \mathcal{L}^{out}_{n_J}} I_l(f; J, n_J; J+1, n_{J+1}) = I^v_{n_J}(f; J) \tag{16}$$

B. Observations from Solving the MILP Problem

We implemented our MILP model and used CPLEX to arrive at the optimal placement on a small topology of 11 nodes and 13 links. Though the results from such small scale experiments are not generalizable, the results provide us guidance to develop a heuristic solution that can solve the placement and routing problem at much larger scale. Based on solving the MILP for a number of scenarios (service chains varying in length, nodes having varying number of cores, and NFs needing multiple, but different numbers of cores), whose details are omitted due to space limitation, we make the following observations.

1) **Observation 1 (O1)**: It is desirable that all the nodes chosen for NF placement are on the shortest path of at least one of the flows. If a node that is on the shortest path of many flows has enough capacity to host all the NFs required for those flows, the optimal solution places all the NFs on that node.
2) **Observation 2 (O2)**: Different instances of the same type of NF are mostly placed on different nodes in the optimal solutions.
3) **Observation 3 (O3)**: The total number of flows that can be accommodated is a function of many factors such as the the node and link capacities, NFs' service rate and flow types. Prioritizing flows that require popular services (at least one) increases the total number of flows that can be accommodated in the network.
4) **Observation 4 (O4)**: More than the minimum number of NF instances needed might be instantiated to satisfy the performance requirements (e.g., stringent delay) of some flows.

IV. ClusPR: A Heuristic Approach for NFV Resource Allocation

Based on the observations summarized in Sec. III-B, we develop ClusPR, a heuristic-based NFV resource allocation algorithm. ClusPR consists of three phases: Initialization/**Clus**tering, **P**lacement, and **R**outing phase. Fig. 1 shows the overall design of ClusPR.

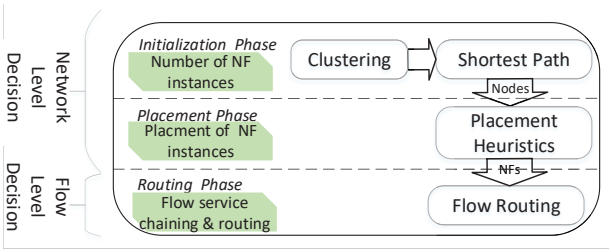


Fig. 1: ClusPR Resource Allocation Phases

Given that a number of flows potentially need to share the same NF instance, the NF may not be on the shortest path of all of the flows. Thus, a significant number of flows may have to deviate from their shortest path to be served by the NFs instantiated. However, this deviation needs to be constrained, so as not to violate the delay requirement of flows.

ClusPR uses clustering of access nodes (e.g. last hop routers to sources and destinations in a wide-area network) to minimize the deviation. Access nodes are clustered based on their proximity to each other i.e access nodes that are close to each other are clustered together. *Because of the proximity between clustered access nodes, flows originating from access nodes of a cluster and going to access nodes in another cluster have a high probability of sharing path, thus, can also potentially share NF instances with no or minimal path stretch.* Inspired by this insight, *ClusPR groups flows into intra-cluster(cluster) and inter-cluster(cluster-pair) flows and performs the placement of NF instances for each of these groups of flows independently.*

After clustering, nodes that are on the shortest path of flows are identified for each of the intra-cluster and inter-cluster set of flows. Note that the clustering is targeted at finding close by access nodes, and the shortest path nodes need not necessarily belong to a cluster or cluster-pair. The shortest path nodes that have a processing power for hosting NFs are regarded as the "best" candidates for hosting NFs. *Information about the path of each group of flows is captured through the shortest path nodes selected*. In the placement phase, *NF instances are placed on the "best" candidate shortest path nodes or their neighboring nodes.* Subsequently, flows are mapped to instances and routed while respecting their precedence constraints for service chains. A flow will be assigned to NF instances that have the minimum overall end-to-end communication cost.

A. *Initialization Phase*

The *first module* in the initialization phase is *clustering*. In this phase, access nodes are clustered based on proximity. Nodes that are either the source or destination of one or more flows are identified as access nodes. A network clustering algorithm, Kruskals algorithm [15], which is a minimum spanning tree (MST) based clustering algorithm [16], is used.

The *second module* is the *shortest path*. Dikjstra's shortest path algorithm is adapted for shortest path calculation between all pairs of access nodes (in a cluster or cluster-pair) and used to identify nodes that are on the shortest path of flows. This set of shortest path nodes are considered the "best" nodes for placing NFs (as noted in **O1**). The number and type of services required by each of the flows between access nodes can be different. To account for these differences, each node keeps a *weight* and a *list*. The *weight* is used to record the number of flows between the access nodes whose shortest path passes through the node. The *list* is used to record the different types of services these flows require.

For example, if a node is on the shortest path between three pairs of access nodes that have 3, 5 and 10 flows between them, the node will have a *weight* equal to 18. In addition, if these flows require DPI, proxy and firewall services, the node will have a *list* containing these three services.

Shortest path nodes are ordered based on their *weight:* the higher the *weight* of a node the higher its priority for hosting NF instances. In other words, nodes that are on the shortest path of many flows are given higher priority, as noted in (**O1**). If the *weight* of the nodes is equal, then nodes that have higher processing power are given priority over nodes that have lower processing power. Next the placement decisions are made for intra-cluster group of flows followed by inter-cluster flows starting from a cluster that has the largest number of intra-cluster flows.

B. *Placement Phase*

The placement phase receives the set of best candidate nodes from the initialization phase. The required type and number of instances to serve a set of flows in a cluster or cluster-pair are calculated. This set of *NF types are ordered according to their popularity, which is measured by the number of flows that require the NF*. The most popular NFs are prioritized to be placed first, considering (**O3**), with ties broken by prioritizing the NF requiring more processing power. NFs are prioritized based on their type. The number of each type of NFs to be instantiated is recorded.

Bin-Packing: The placement *heuristic does a bin-packing of the prioritized NFs on the set of "best" candidate shortest path nodes or their neighboring nodes*. An NF instance is placed on a node if and only if the

node has the NF type in its *list*. This constraint is used to make sure that an NF instance placed on a node is needed by the flows whose shortest path passes through the node. An NF type that is on top of the priority queue of NF types is taken and the ordered queue of "best" candidate nodes is iterated through until a node that has the NF type in its *list* is found. Once a node is found the NF is placed and the number of instances of the NF type is decreased. The node will then be regarded as an *active node* for placing the next NF type.

Diversity: The placement *heuristic diversifies the types of NFs placed on a node*. That is, if more than one instance of a given type of NF is needed, *the algorithm prioritizes placing different types of NF instances in one node* rather than placing multiple instances of the same type of NF on the node. If different types of NFs are placed on one node, the probability that a flow can get all of the services it requires from one node will be high, as noted in (**O2**). Serving a flow's chain in one node has advantages such as decreasing the communication cost and the delay experienced by the flow.

Next, the following NF type is picked from the queue of NFs and placed on the *active node* provided that the NF is found in its *list*. If not the algorithm returns to the queue of the nodes, and looks for another node that has the NF in its *list*. After placing one instance of all types of NFs, the algorithm returns back to the top of the queue of NFs and place the second instances.This process is repeated until all the instances of all NF types are placed.

C. *Routing Phase*

Finally, the routing phase assigns NF instances and determines the routes for all flows. A flow will use NF instances that are placed on the set of shortest path nodes and/or their neighboring nodes for the cluster or cluster-pair the flow belongs to. Out of this set of NF instances, *a flow is assigned to NF instances that have the smallest end-to-end cost*. In order to select these NFs, *a flow's routing is modeled as a multi-stage shortest path problem* in which the stages of the multi-stage graph represent the services in the service chain of the flow.

For constructing the graph, the costs on the links of the graph need to be calculated. The costs can be calculated using shortest path algorithms such as Dijkstra's algorithm. The shortest path costs of the links from the source node to the nodes hosting the first NF instance of the chain and the links from the destination node to nodes hosting the last NF of the service chain need to be calculated for each of the flows. The costs of the links between the stages (nodes hosting NFs in the chain) are calculated only once, which decreases the computation complexity of constructing the multi-stage graph.

Dynamic programming is used to find the optimal shortest path in a multi-stage graph. To formulate the dynamic program, two distance notations are adapted. $C(n, n^{'})$ is used to represent the cost of the shortest distance between node n and $n^{'}$ that belong to two consecutive stages. e.g., $C(s_f, n_{1^{st}nf})$ represents the cost of the shortest distance between the source node of flow f (s_f) and node ($n_{1^{st}nf}$) that hosts the 1^{st} service instance. $D(n, j)$ represents the shortest distance between node n that is hosting the j^{th} service of the flow to the destination node(d_f), e.g., $D(n_{2^{nd}nf}, 2)$ represents the shortest distance from node $n_{2^{nd}nf}$ that is hosting the 2^{nd} service to the destination node.

The dynamic program formulation is given as

$$D(s_f, d_f) = \min_{n_{1^{st}} \in \mathcal{N}_{1^{th}nf}} (C(s_f, n_{1^{st}nf}) + D(n_{1^{st}nf}, 1)) \tag{17}$$

$$D(n_{j^{th}nf}, j) = \min_{n_{j^{th}nf} \in \mathcal{N}_{j^{th}nf}} (C(n_{j^{th}nf}, n_{(j+1)^{th}nf}) + D(n_{(j+1)^{th}nf}, j+1)) \tag{18}$$

$\mathcal{N}_{j^{th}nf}$ is the set of nodes that are on the shortest path or the fewest hops away from the shortest path nodes and are hosting the flow's j^{th} service type for the cluster or cluster-pair the flow belongs to. The dynamic program is solved starting from the destination node until the source node is reached. Note that the *clustering helps in decreasing the computation complexity by decreasing the possible set of instances($\mathcal{N}_{j^{th}nf}$) that a flow can choose from*, which increases the scalability of ClusPR. The algorithm checks for all the possible pairs of combinations of available instances and *chooses the instances that give the minimum overall communication cost*.

V. Experimental Results

We extensively analyze the performance of ClusPR and compare its performance with two alternatives, E2 [10] and [9] (refered to as "Deploying" for ease of reference). We report here on experiments with the Rocketfuel topology AS 1221 [17] shown in Fig. 2 used as a test network. We classify nodes in the topology as "access" (in blue), "edge" (green) and "core" (orange) nodes, in a manner similar to [18]. NFs are considered to be hosted on (or adjacent to) edge and core nodes. Each node hosting NFs has 4 cores and each instance of an NF requires one core, with a service rate of 10Mbps.There are 5 types of NFs (e.g., Firewall, Deep Packet Inspection (DPI), Network Address Translator (NAT), Intrusion Detection System (IDS) and Proxy). All the links have a capacity of 1 Gbps, and the delays on the links are: access-edge: 3 ms; edge-core: 10 ms; core-core: 40 ms. The source and destination nodes of flows as well as the services required by the flows are generated randomly. The arrival rate of flows is assumed to follow a log-normal distribution [19] and the length

of the service chain for each flow is assumed to be equal to two. The service types in the chain of each flow are generated randomly. The MILP model in Section III which is solved using CPLEX is not able to scale to the network scenario considered in this experiment. The main performance measures we evaluate are total delay, path stretch and the number of flows admitted. In addition, we have done a trade-off analysis between the number of NF instances instantiated and the delay of flows.

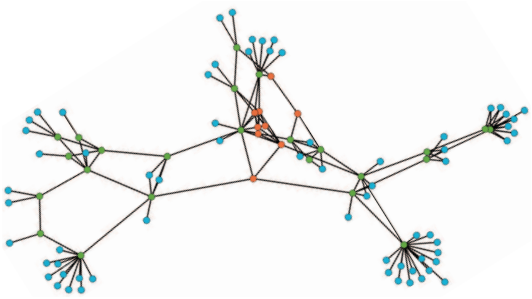
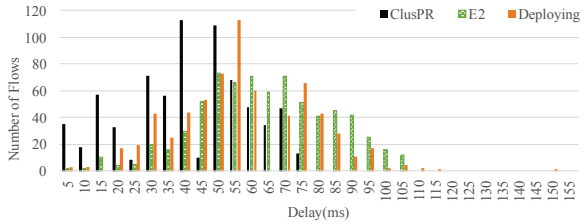


Fig. 2: Test Network:100 nodes and 294 links

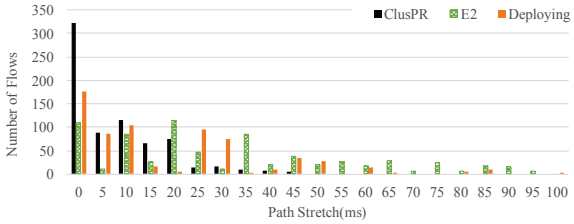
Average and Worst-case Delays: ClusPR restricts flows to use instances placed near/on the shortest path nodes of a cluster or cluster-pair that flows belong to. This constraint helps ClusPR minimize the worst-case delay of flows. Fig. 4a and 4b show the average and worst-case normalized delays of flows respectively. Fig. 5 shows Cumulative Distribution Function(CDF) of the delay normalized with respect to the shortest path delay. ClusPR is able to achieve a worst-case normalized delay that is $9-10\times$ less than the worst-case delay of E2 and Deploying. In addition, the averaged normalized delay of ClusPR is $1.2-1.6\times$ less compared to E2 and Deploying

Total Delay and Path Stretch: The total communication delay flows experience is a summation of links propagation delay and queuing delay on the NF instances of service chains. Assuming that the queuing delay of flows is small compared to the propagation delay [20], the total delay of a flow is then measured as the summation of propagation delay on its links. The path stretch is measured as the difference between the total delay and the shortest path delay. Fig. 3 shows the total delay and path stretch distributions with 720 flows that have average flow arrival rate of 0.5 Mbps. Both E2 and ClusPR instantiated 74 instances for serving the flows.

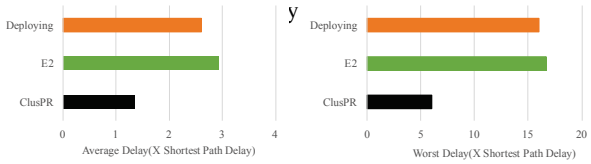
ClusPR has a shorter path stretch compared to both Deploying and E2 for the *same number of instances*. The performance gain is achieved because ClusPR takes as input the path of flows in making NF placement decisions. ClusPR seeks to minimize the path stretch by placing NF instances near the shortest path of flows. In addition to placing NFs near the path of the flows that need them, ClusPR diversifies the type of NFs placed on a node by placing different types of NFs. This increases



(a) Distribution of Delay



(b) Additional Delay from Path Stretch



(a) Average Delay (b) Worst Delay

Fig. 4: Average and Worst-case Normalized Delay

the probability that a flow can get all of the services of its chain in one node. Which in turn can decrease the additional path stretch incurred for fulfilling the precedence constraint. Both E2 and Deploying do not explicitly reduce the path stretch of flows.

Delay Requirement Satisfaction of Flows: For this experiment, flows are assumed to have a specified delay requirement in terms of the maximum normalized delay that they can tolerate. The normalized delay requirement of flows is assumed to be uniformly distributed between $1-2.5\times$ their shortest path delay.

The number of flows whose delay requirement is satisfied is shown in Fig. 6. ClusPR is able to satisfy the delay requirement of 87% of the flows while E2 and Deploying managed to fulfill the delay requirement of 60% and 70% of the flows respectively. ClusPR satisfies the delay requirement of 17% to 27% more flows compared to E2 and Deploying.

Trade-off Analysis: One of the observation from the optimal solutions of the MILP model in Section III-B(**O4**) is that more NF instances might need to be instantiated if stringent delay requirement of flows is to be satisfied. An experiment is conducted in order to analyze the trade-off between the number of NF instances instantiated and the delay performance of flows. Fig. 7 shows the path stretch of flows when ClusPR is instantiating 74 and 88 NF instances. The path stretch has decrease with an increase in the number of NF instances.

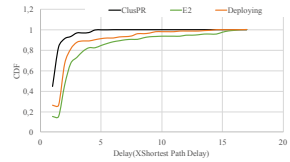


Fig. 5: CDF of Normalized Delay

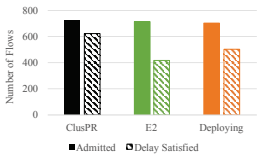


Fig. 6: Flow Statistics

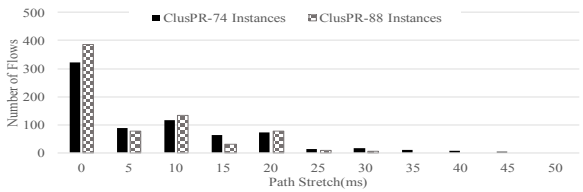
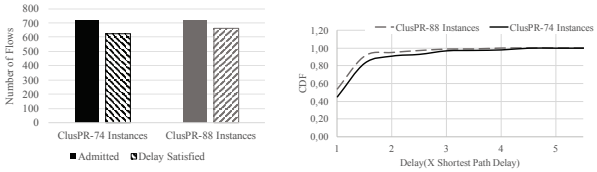


Fig. 7: Additional Delay from Path Stretch



(a) Flow Statistics (b) CDF of Normalized Delay

Fig. 8: Trade-off Analysis B/n Number of Instances and Delay

With 74 instances the delay requirement of 87% of the flows is satisfied. By increasing 14 more NF instances, the delay requirement of 92% of the flows is satisfied as shown in Fig. 8a. As can be inferred from the CDF in Fig. 8b, the average and worst-case normalized delays have also seen a slight decrease with an increase in the number of NF instances. The average normalized delay has reduced from $1.36\times$ to $1.32\times$ and the worst-case delay has decreased from $6\times$ to $5\times$ the short path delay.

VI. Conclusion

The flexibility brought about by NFV can potentially change the way networks are managed and services are provisioned. Nevertheless, an efficient resource allocation algorithm is needed to instantiate NF instances when and where needed, and route flows through them accordingly. In this paper, a comprehensive novel multi-objective MILP model is formulated for the NFV-RA problem. Based on the useful insights obtained from the optimal solutions of the MILP model, a clustering-based heuristic scheme, ClusPR, is developed. ClusPR is scalable and balances between minimizing the path stretch and maximizing the network utilization. By factoring in information about the path of flows in the NF placement decision making and diversifying NFs, ClusPR is able to considerably minimize the path stretch. Compared to the state-of-the-art approaches ClusPR has managed to decrease the average normalized delay by a factor of $1.2 - 1.6\times$ and the worst-case delay by $9 - 10\times$, while utilizing the same number of instances. This minimization of the path stretch has enabled ClusPR to satisfy the delay requirement of 17% to 27% more flows. In addition, the trade-off between the number of NF instances and delay is shown by demonstrating the gain in delay performance for an increase in the number of NF instances.

References

[1] J. Sherry et al., "Making middleboxes someone else's problem: network processing as a cloud service," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 13–24, 2012.

[2] W. Zhang et al., "Opennetvm: A platform for high performance network service chains," in *workshop on Hot topics in Middleboxes and Network Function Virtualization*. ACM, 2016.

[3] Y. Zhang et al., "Steering: A software-defined networking for inline service chaining," in *Network Protocols (ICNP), 2013 21st IEEE International Conference on*. IEEE, 2013, pp. 1–10.

[4] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[5] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Network and Service Management (CNSM), 2015 11th International Conference on*. IEEE, 2015, pp. 50–56.

[6] R. Cohen et al., "Near optimal placement of virtual network functions," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1346–1354.

[7] V. Sekar et al., "Design and implementation of a consolidated middlebox architecture," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012, pp. 323–336.

[8] Q. Li, Y. Jiang, P. Duan, M. Xu, and X. Xiao, "Quokka: Latency-aware middlebox scheduling with dynamic resource allocation," *Journal of Network and Computer Applications*, vol. 78, pp. 253–266, 2017.

[9] T.-W. Kuo et al., "Deploying chains of virtual network functions: On the relation between link and server usage," in *INFOCOM, 2016 Proceedings IEEE*. IEEE, 2016.

[10] S. Palkar et al., "E2: a framework for nfv applications," in *25th Symposium on Operating Systems Principles*. ACM, 2015.

[11] B. Addis et al., "Virtual network functions placement and routing optimization," in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*. IEEE, 2015, pp. 171–177.

[12] S. Mehraghdam et al., "Specifying and placing chains of virtual network functions," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 7–13.

[13] M. Xia et al., "Network function placement for nfv chaining in packet/optical data centers," in *Optical Communication (ECOC), 2014 European Conference on*. IEEE, 2014, pp. 1–3.

[14] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[15] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.

[16] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Transactions on computers*, vol. 100, no. 1, pp. 68–86, 1971.

[17] N. Spring et al., "Measuring isp topologies with rocketfuel," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.

[18] A. Afanasyev et al., "Interest flooding attack and countermeasures in named data networking," in *IFIP Networking Conference, 2013*. IEEE, 2013, pp. 1–9.

[19] Y. Zhang et al., "On the characteristics and origins of internet flow rates," in *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4. ACM, 2002, pp. 309–322.

[20] F. P. Kelly, "Models for a self–managed internet," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 358, no. 1773, pp. 2335–2348, 2000.