AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your paper. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

AQ1: Author: Please confirm or add details for any funding or financial support for the research of this article.
AQ2: Note that if you require corrections/changes to tables or figures, you must supply the revised files, as these items are not edited for you.
AQ3: Please provide the accessed date for References [1]–[3], [25], [26], and [39].
AQ4: Please confirm that the edits made to Reference [7] is correct as set.
AQ5: Please provide the report number for Reference [14].
AQ6: Please provide the complete details and exact format for Reference [38].
AQ7: Please provide the page range for Reference [42].

A Requirement-Oriented Design of NFV Topology by Formal Synthesis

A. H. M. Jakaria, Mohammad Ashiqur Rahman, and Carol Fung

***Abstract*—Computer networks today heavily depend on expensive and proprietary hardware deployed at fixed locations. Network functions virtualization (NFV), one of the fastest emerging topics in networking, reduces the limitations of these vendor-specific hardware with respect to the flexibility of network architecture and elasticity in handling varying traffic patterns. Many defense mechanisms against cyberattacks, as well as quality enhancing techniques have been proposed by leveraging the capabilities of the NFV architecture. NFV allows a flexible and dynamic implementation of virtual network functions in virtual machines running on commercial-off-the-shelf (COTS) servers. These quality enhancing network functions often work as a filter to distinguish between a legitimate packet and an attack packet and can be deployed dynamically to balance the variable attack load. However, allocating resources to these virtual machines is an NP-hard problem. In this paper, we propose a solution to this problem and determine the number and placement of the virtual machines (VMs) hosted on COTS servers. We design and implement two separate automated frameworks for defense and quality maintenance that model the resource specifications, incoming packet processing requirements, and network bandwidth constraints. It uses satisfiability modulo theories (SMT) for modeling this synthesis problem and provides a satisfiable solution.**

***Index Terms*—NFV architecture, formal modeling, DDoS security, network QoS, synthesis.**

I. Introduction

INFORMATION security while maintaining quality of services (QoS) is one of the topmost priorities for businesses and organizations. ISPs and online services, especially small or medium-sized organizations that lack the resources to discern any difference between legitimate and attack traffic, can be damaged severely by cyberattacks. Some recent incidents prove that different types of attacks are becoming stronger and more frequent day by day. For example, the *KrebsOnSecurity* was the target of a DDoS attack with traffic of close to 620 Gbps in September 2017 [26]. The *Spamhaus* DDoS attack in January 2016 [25] generated 602 Gbps attack. In the third quarter of 2017, organizations faced an average of 237 DDoS attack attempts per month [39]. Since QoS is a necessary feature for today's complex network services, it is important to have high quality and low cost solutions to defend against cyberattacks for businesses and organizations.

Many solutions for cyber defense are composed of proprietary hardware. Upgrading or adding new network functions typically enforces the integration of more of these hardware appliances which requires time and imposes high costs. They cannot satisfy the automation, scalability, and robustness of today's network security operations. Traditional methods of threat detection and QoS management are limited by the restricted computation capacity and inflexibility of involved dedicated hardware, such as firewall and routers.

In NFV technology, network functions are implemented and deployed as virtual machines (VMs) in the form of software that runs on the commodity hardware. The VMs run on these general purpose hardware systems, which not only provides the benefit of elasticity, but also reduces the cost by running on low-cost commodity platforms like x86- or ARM-based servers instead of specialized hardware. The use of NFV opens a new opportunity for businesses and organizations, to find a low cost solution to combat cyberattacks. NFV offers much less complex network architecture, reduced power usage, lower OpEx, lower CapEx, and low time-to-market for launching new functionalities [1], [2]. It allows testing new apps more easily and offers an improved flexibility in assigning virtual network functions (VNFs) to hardware.

Utilizing VNFs to defend against cyberattacks while maintaining QoS has become a common trend these days [3]. However, using the available server resources efficiently is a challenge because they are limited. The physical properties of the servers, such as memory, storage, CPU, etc., determine the capabilities of the VNFs running on the VMs within these servers [14]. There are few works available in the literature providing the formal modeling of such a network architecture [15]. None of them solves this problem in a timely and responsive manner. In this work, we present two novel frameworks, VFenceSynth and VTVSynth, which solve this bin packing problem using formal verification from security and quality maintenance point of view, respectively. These are automated frameworks for synthesizing virtual network configurations and placements of VMs, using constraint satisfaction checking by SMT. This paper mainly contributes to the following.

Manuscript received June 7, 2018; revised December 14, 2018 and May 21, 2019; accepted May 30, 2019. The associate editor coordinating the review of this paper and approving it for publication was S. Latre. *(Corresponding author: A. H. M. Jakaria.)*

A. H. M. Jakaria is with the Department of Computer Science, Tennessee Tech University, Cookeville, TN 38505 USA (e-mail: ajakaria42@students.tntech.edu).

M. A. Rahman is with the Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174 USA (e-mail: marahman@fiu.edu).

C. Fung is with the Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: cfung@vcu.edu).

Digital Object Identifier 10.1109/TNSM.2019.2920824

1) It formally models the resources and network topology that implements NFV.
2) It provides an efficient solution for the resource allocation problem for different types of VNFs.
3) It provides a thorough implementation and evaluation of the automatic synthesis tool.

We briefly introduced VFenceSynth in a previous work [23].

The rest of this paper is organized as follows: Section II presents an overview of NFV use cases and related works. We discuss the framework of the solution in Section III. Section IV describes the formal model of a DDoS defense mechanism, its implementation, and the dynamic adaptability of the solution and a case study. The formal model and a case study for a quality management technique are discussed in Section V. The evaluation results of our model are presented in Section VI. Finally, we conclude the paper in Section VII.

II. Background and Research Objective

This section briefly overviews two different use cases of NFV, related works, and our objectives that use formal verification to solve the resource optimization problems in NFV.

A. An Overview of Use Cases of NFV

In a virtualized network platform, dynamic deployment and maintenance of software-based security services can be utilized to cope with the sophisticated network attacks that fool valid network services. Currently, many security vendors and Internet service providers are trying to establish common interfaces for NFV-based security services. A firewall, intrusion detection system (IDS), intrusion prevention system (IPS), and other security services could be instantiated as virtual network services. NFV platforms allow third parties or vendors to develop security applications with network resources using the underlying standard security interfaces.

Protection Against DDoS: Dynamic creation of VNFs deployed to defend against DDoS attacks has been discussed in the literature [15], [24]. Fayaz *et al.* [15] proposed a flexible and elastic DDoS defense system, Bohatei, that shows the benefits of software defined networking (SDN) [37] and NFV in the context of DDoS defense. It utilizes NFV architecture to dynamically configure scale (*e.g.,* 10 Gbps vs. 100 Gbps attacks) and type (*e.g.,* SYN proxy vs. DNS reflector defense) of DDoS defense realized by defense functions running on VMs. A NFV-based elastic data filtering architecture to defend against DDoS attack was discussed in [24] by Jakaria *et al.* The defense filtering network consists of dynamically created VNFs, as shown in Fig. 1. On the physical layer, there are one or more product servers that provide online services to customers from the Internet, and some commodity servers that are connected to each other. Each server hosts VMs to realize different types of VNFs, such as dispatchers, switches, and agents. The dispatcher runs a load balancing algorithm and distributes incoming traffic to the corresponding agents. It keeps tracking which agents are being used for which flows and dispatches the packets accordingly. The VNFs are organized in a way so that attack flows will be handled by filtering agents and they will filter out the spoofed traffic by performing a spoofed

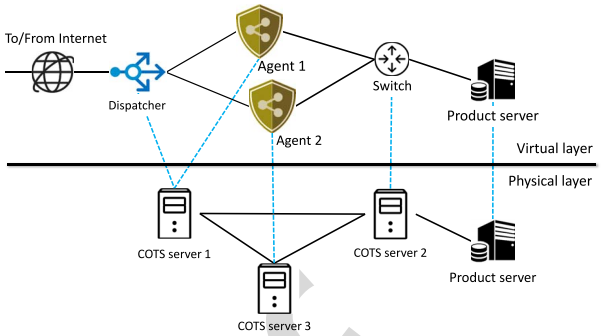


Fig. 1. An NFV network topology that defends DDoS attacks on the target using multiple agents [24].

handshake with the source, as well as the product server. Each agent maintains a whitelist of valid sources. They let legitimate packets flow through it while dropping suspicious ones. Source are added to a whitelist upon a successful spoofed TCP handshake with the source client. Consequently the agent performs a spoofed handshake with the product server, and after that, a connection is established. A source is removed from the whitelist upon the termination of a flow or in the case of a timeout for unresponsive clients. The number of VNFs and their capability of processing packets depend on the commodity servers' available resources. The placement of the dispatcher and the agents, and their deployment decision are challenges that need to be addressed.

QoS Maintenance: Another utilization of NFV can be found in maintaining quality of service (QoS) in an enterprise or home network. Using NFV, we can implement virtual customer premises equipment (vCPE), which can be scaled according to service requirements [21]. Virtual evolved packet core (vEPC) and IP multimedia subsystem (vIMS) are some other ideas that helps to enhance QoS in networks. Abstracted virtual networks on top of physical networks can be enhanced using virtual appliances, increasing fine-grained controls and isolation, as well as insertion of acceleration or security services.

One use case is the virtualization of CPE devices [18]. As discussed in the paper, virtualization of typical CPE devices such as residential gateways (RG) and set-top boxes (STB) in time-shifted IPTV will increase QoS. Fig. 2 presents the network architecture for this purpose. In a traditional set up, the RGs and STBs are dedicated devices installed in customer premises. Residential gateways are responsible for processing the traffic to and from the customers. Typically, they consist of a range of network components such as firewalls, DHCP servers, VPN gateways, NAT routers, etc. Set-top boxes are specific to customers which are basically used to convert the network traffic to a format recognizable by TVs. They can utilize their memory in times of buffering live broadcast TV or video on demand. The quality of experience of the customers greatly depends on the memory capacity of the STBs in times of channel changing for live TV streaming. Whenever a customer changes channels for live TV, a burst of data is first unicast to the STB of that customer which is the data for 30 to 60 seconds of video play. During this period, the STB joins the multicast group of that particular channel [5]. STBs are

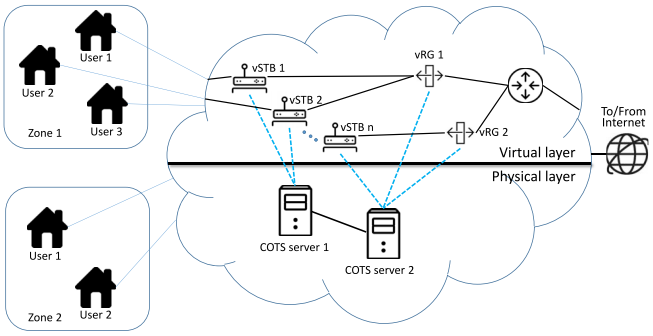


Fig. 2. An NFV network topology to enhance QoS in home area network using virtual set top boxes and residential gateways (adapted from [18]).

also utilized for recording videos and store in a local hard-drive for viewing later. There is a certain capacity of each STBs which can be utilized for video storage.

If these CPE devices are virtualized and moved to data centers of the service providers, customers are required to maintain low-cost devices only for physical connectivity. Virtualization of such network functions reduces the operating expense by avoiding maintenance and updating the properties of the physical devices. Dynamic modification of virtual STB and RG properties such as memory, storage and processing rate enhances the QoS. The service provider can offer virtually unlimited storage space to the clients, as well as enable access to various services and shared contents from different locations. It also allows the service providers to offer novel services more quickly and smoothly.

B. Related Work

Here we present a literature review of NFV applications in security and QoS maintenance. We also discuss resource management and VNF placement mechanisms on COTS servers.

Security against DDoS: There are several works available in the literature that introduce the usage of NFV to security systems. Rebahi *et al.* designed and developed a virtual security appliance (vSA) that is capable of detecting various network attacks while offering an acceptable level of performance [40]. A cloud-based architecture [44], and VGuard [16], a tool based on NFV, have been developed essentially to counter DDoS attacks. Liyanage *et al.* introduced NFV-based security apps to protect the LTE architecture [29]. Pastor and Lopez proposed use cases for an open operation, administration, and management (OAM) interface to virtualized security services for home/residential network access [38]. However, most of them do not address the issue of resource management in terms of VNF placement when deploying the associated VMs. Software-defined Networking (SDN), along with virtualized network functions was utilized to create a DDoS mitigation technique by Yan *et al.* [43].

Fayaz *et al.* [15] proposed an ISP-centric deployment model, where an ISP offers DDoS-defense-as-a-service to its customers by deploying multiple datacenters consisting commodity servers to run standard VNFs. The authors formulated the resource management problem as a constrained optimization via an integer linear program (ILP) which takes several hours to provide a solution, which is sufficient for an adversary to compromise the system. With two greedy algorithms, they defined a hierarchical decomposition of the resource optimization problem into two stages. However, many different greedy algorithms are possible for a problem and there is no structured way to find the most effective ones. Greedy algorithms have the possibility of being stuck in a local optimum, and falsely indicate a sub-optimal solution.

Marchetto *et al.* proposed a VNF placement model for industrial Internet of Things that focuses on minimizing the latency between two or more endpoint devices. They also verify proper policy enforcement for reliable connectivity and security by analyzing misconfigurations [32].

VNGuard, a framework proposed by Deng *et al.* [13], performs management of virtual firewalls that protect virtual networks (VNs). The framework also proposes an approach based on ILP to find an optimal virtual firewall placement, which fulfills resource and performance constraints. However, the solution only addresses virtual firewalls and cannot cope with more sophisticated attacks like DDoS.

QoS of IPTV: There are several works in the literature that proposes virtualiaztion of IPTV network functions. Han *et al.* discussed the opportunities and challenges in different use cases of NFV [18]. They discussed two use cases of NFV in mobile core network and home area network in terms of performance, manageability, reliability and security.

Aggarwal *et al.* took the advantage of virtualization of network functions to propose a common infrastructure based IPTV services [5]. They focused on two services - video on demand (VoD) and Live broadcast TV. They proposed an algorithm that provides the minimum number of servers required to fulfill all the requests of these services. The authors also provided a generalized framework in a virtualized environment to compute the resources needed to support multiple services without missing the deadline for any service [6].

Hysenbelliu and Teresa proposed a cloud-based architecture for a IPTV service in the data center of a software media communications ISP [22]. They used NFV and SDN techniques to achieve more secure, scalable, and cost effective services.

NFV Architecture Design: Addis *et al.* [4] proposed a mixed integer linear programming (ILP) formulation for a generic VNF routing optimization problem. They used an NFV network model for ISP operations. Mehraghdam *et al.* proposed a VNF graph that can be mapped to the network [34]. They found a placement solution for VNFs which are chained together, given the limited network resources. Bari *et al.* focused on determining the required number and placement of VNFs that optimizes network operational costs [10]. Luizelli *et al.* proposed a formalization of the VNF placement and chaining problem [30]. They solved an ILP model while minimizing end-to-end delays. Chowdhury *et al.* redesigned algorithms to place and migrate VMs based on the utilization of CPU and memory [11]. In [33], Masdari *et al.* provided a survey and analysis of the existing VM placement schemes proposed in the literature for the cloud computing and data centers. The authors in [12] studied the placement of VNFs and provided near optimal algorithms based on distance and set up costs, service level agreements, etc. In [27], the authors

mainly considered the incremental deployment of software based middleware, while [31] presents the study of algorithmic solutions that exploit the flexibility of SDN, and proposes solutions for the deployment of middleboxes.

Ayoubi *et al.* proposed a Cut-and-Solve based technique to find the optimal placement of VNF instances [8]. Their approach maximizes the number of policy-aware traffic flows. In [7], the authors proposed a flexible service function chain (SFC) orchestration that allocates compute and network resources while considering a relaxed traversal order for VNFs. They considered the semantics of the VNFs and used a mixture of ILP and heuristics to solve the problem scalably.

Hawilo *et al.* proposed an intelligent VNF orchestrator that can decide between migration and re-instantiation to resume a VNF after an outage [20]. They used an MILP-based model and make sure that the downtime of VNFs and latency of SFCs are minimized. Li and Qian presented a framework that abstracts the high-level objective of minimizing hardware resources by minimizing the number of VNF instances [28]. Their ILP-based solution provides a quick solution to minimize the number of VNFs while making sure that at least one VNF associated with the policy of a flow is instantiated on its path.

All of the above works did not consider any specific network security (e.g., DDoS mitigation) or quality (e.g., IPTV) issue while solving for placement of VNFs, and do not provide a fine-grained server-specific deployment solution. We synthesize the mapping between the physical servers in the substrate network and the VNFs while solving the resource allocation. We also consider the semantics of different types of VNFs. For example, a dispatcher and an agent need to communicate with each other according to the DDoS mitigation technique, but agents may not need to talk, hence can be placed irrespective of their communication requirements with each other.

C. Research Challenge and Our Objective

A service provider needs an efficient way of assigning its available computing and network resources to the virtual defense system that maintains quality of experience for the end users. They need to decide how many VMs of each type to run on each available server so that the incoming and outgoing traffic is handled properly. Doing it in a timely and responsive way is a challenge. A straight forward solution is to form a large combinatorial NP-hard problem [9]. However, it takes hours or longer to compute the solution for the combinatorial problem. Our purpose is to introduce a synthesis tool that can solve this problem efficiently. We propose a formal method-based framework that models the NFV topology design as a satisfiability problem and finds an NFV deployment plan that satisfies necessary security and quality requirements through solving the satisfiability problem. In particular, we propose two separate frameworks to explain the NFV topology synthesis for two scenarios: (i) DDoS defense, and (ii) QoS in IPTV services. We name the first framework as VFenceSynth, and the second one as VTVSynth, respectively. We build two separate tools for the proposed frameworks. The results are found within a sustainable period of time, that allows reconfiguration of the VM deployment strategy quickly. Although our model is based on specific types of VMs, such as agent, dispatcher, STB, etc., it can be altered to work for any generic resource allocation and VM placement problem that arise when implementing NFV-based techniques.

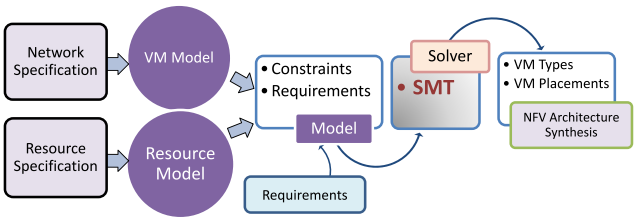


Fig. 3. The framework architecture of NFV network synthesis.

III. NFV Architecture Synthesis Framework

The framework follows a top-down approach for the automation of NFV network architecture design. Fig. 3 presents an overview of the general framework for automated synthesis of the NFV architecture. It formally models the COTS server resource configurations, as well as the specifications of VMs. It formalizes the NFV architecture design synthesis problem as a VNF deployment plan that includes the determination of VM placements and properties (*e.g.,* type, memory, storage, and CPU), satisfying the packet processing requirements and quality of service, within the physical resource constraints. Finally, it encodes the synthesis problem into SMT logics and provides a feasible solution using an efficient solver.

Use Case-Based Design (VFenceSynth): We build a specific framework called VFenceSynth, based on the DDoS mitigation dispatcher-agent [24] technique discussed in Section II-A. In this case, the network topology of servers, their resources, and network bandwidth, as well as the attack traffic intensity, are provided to the model as input from a text file. The packet processing requirements (*i.e.,* the attack traffic properties), the physical network topology including bandwidths of the links and latency of the servers from ingress points are also modeled by the framework. The output from the SMT solver provides the count and placement of VMs implementing dispatchers and agents. The tool can provide quick solutions to the problem based on the traffic changes over time. It is worth mentioning here that our model and implementation are flexible enough to modify the types of these VNFs easily. For the sake of simplicity and a specific example, we choose to do the model based on this use case.

Use Case-Based Design (VTVSynth): We present another framework named VTVSynth, based on the quality enhancing technique of IPTV services discussed in [18]. In this case, we take the number of customers in different home areas, their requirements for a quality experience of IPTV services (*i.e.,* STB storage, memory, network bandwidth, etc.), as well as the available COTS server specification. The tool provides an output that specifies the locations of different VMs implementing virtual STBs and RGs. A service provider can utilize this

TABLE I
NOTATION TABLE FOR VFENCESYNTH

Notation	Definition
$D^V_{i,j}$	Is VM j of commodity server i deployed?
$T^V_{i,j}$	Type of VM j of commodity server i.
$M^V_{i,j,t}$	Memory of VM j of type t on commodity server i.
$C^V_{i,j,t}$	CPU of VM j of type t on commodity server i.
$P_{i,j,t}$	Packet processing rate of VM j of type t on server i.
M^S_i	Memory of commodity server i.
C^S_i	CPU core of commodity server i.
$R_{i,j,k,l}$	Is VM j on server i is reachable from VM l on k?
$B^V_{i,j,k,l}$	Required virtual bandwidth between VM j on server i and VM l on server k.
$B^P_{i,k,z}$	Required physical bandwidth of z^{th} link on the path from server i to server k.

solution to set up its virtual network functions on its available COTS infrastructure.

IV. FORMAL MODEL OF VFENCESYNTH

This section discusses the formal model of the requirements and the constraints of VFenceSynth. Table I lists the variables used in the model. We also present the implementation details and a sample case study to explain the model.

A. Preliminary

VNFs deployed on commodity servers intercept the incoming traffic towards product servers. They drop malicious traffic and allow the legitimate portion to reach the product server. NFV orchestrator reroutes the traffic from the ingress points to these VNFs. The VNFs need to be deployed efficiently on the servers, so that resource of the servers is well-utilized, as well as the VNFs are able to deal with increasing attack traffic.

B. Packet Processing Requirement Model

Besides the fact that these servers have lower cost than dedicated vendor-specific hardware, they often have a limited amount of resources in terms of memory, CPU, bandwidth, etc. VMs are installed on these servers that share the limited resources. Our model assumes that no VM is deployed across more than one server. In our design, we propose two types of VNFs: dispatcher and agent. All the incoming packets are forwarded to the dispatchers. A dispatcher forwards the packets to several agents. Hence, a dispatcher requires high memory and CPU, which allow it to process high volume of attack traffic, as well as normal traffic. It should be installed on a server that can provide required resources to ensure that the dispatcher can efficiently dispatch the incoming packets to the agents and is not overwhelmed by the large number of packets. If $\mathbb{T}$ is the set of all the types, in our case, $\mathbb{T} = \{\mathrm{D}, \mathrm{A}\}$.

1) Resource Requirement Model: A packet processing system makes use of many system components. It is important to identify individual contributions of the components to the performance of the overall system [41]. Memory and CPU cycles of the COTS servers are one of the main resources that we consider in this research. A pipeline-based system that processes only a fixed amount of bytes (basically the header) of each packet and that forwards this data to the following stage in each clock cycle achieves a packet throughput corresponding to its CPU clock rate [19]. Of course, there might be other resources, such as I/O bus bandwidth, that are limited in supply when it comes to the processing of incoming traffic from clients or the Internet. It is very easy to change the framework to take other inputs that affect these requirements.

Let $M^V_{i,j,t}$ be the memory of VM j of type t running on server i; while $C^V_{i,j,t}$ be the CPU of that VM. We provision proper utilization of the memory of the servers. The sum of the memories of all the deployed VMs on a server should be between a minimum and a maximum threshold percentage of the physical memory of that server which is available for the VMs. If μ and $\bar{\mu}$ are the maximum and minimum allowed percentage of memory utilization, respectively, then the following should hold:

$$\forall_{i\in\mathbb{S}} \left(\sum_{j,t} M^V_{i,j,t} \leq \mu \times M^S_i \right) \wedge \left(\sum_{j,t} M^V_{i,j,t} \geq \bar{\mu} \times M^S_i \right) \quad (1)$$

The sum of the CPUs of all the deployed VMs on a server should not exceed the actual physical CPU of that server.

$$\forall_{i\in\mathbb{S}} \sum_{j,t} C^V_{i,j,t} \leq C^S_i \quad (2)$$

Regardless of the type of a VNF, the VM that contains it, must be allocated enough memory and CPU so that it has the best possible packet processing capabilities. The packet processing rate of a VM depends on the memory and CPU of the VM. If $P_{i,j,t}$ refers to the packet processing rate of a VM, we can express this as a function of memory and CPU, where α_t is a constant that determines the impact of memory and β_t is a constant determining the impact of CPU on the packet processing rate for a particular type, and $T^V_{i,j}$ determines the type of VM j on server i.

$$\forall_{t\in\mathbb{T}} \left(T^V_{i,j} = t \right) \rightarrow P_{i,j,t} = \left(\alpha_t \times M^V_{i,j,t} \right) \times \left(\beta_t \times C^V_{i,j,t} \right) \quad (3)$$

If a VNF is deployed on a VM, it needs to have memory and CPU greater than a minimum value. $\mathrm{M}^{\min}_{\mathrm{t}}$ and $\mathrm{C}^{\min}_{\mathrm{t}}$ refer to these minimum values for type t, respectively. This ensures that the packet processing rate depends both on memory and CPU. If $D^V_{i,j}$ denotes if VM j is deployed on sever i, the following holds:

$$\forall_{t\in\mathbb{T}} \left(T^V_{i,j} = t \right) \wedge D^V_{i,j} \rightarrow M^V_{i,j,t} \geq \mathrm{M}^{\min}_{\mathrm{t}} \quad (4)$$

$$\forall_{t\in\mathbb{T}} \left(T^V_{i,j} = t \right) \wedge D^V_{i,j} \rightarrow C^V_{i,j,t} \geq \mathrm{C}^{\min}_{\mathrm{t}}. \quad (5)$$

2) Network Bandwidth Requirement Model: We take the overall network topology of the system as an input to our solver. That is, we know how the servers are connected to the ingress points, as well as to each other. The bandwidth of each link in the topology is also provided. It is required that the packet processing rate of the VMs does not exceed

the bandwidth of the physical links, otherwise it would be impossible for the VMs to communicate with each other.

We denote the reachability between two VMs running on two VMs by $R_{i,j,k,l}$, where VM *j* is running on server *i* and VM *l* is running on server *k*. Each deployed agent should be reachable from the dispatcher, and vice versa. No communication is required between the agents only.

$$D^V_{i,j} \ \wedge \ D^V_{k,l} \wedge \Big(T^V_{i,j} = \mathrm{D}\Big) \wedge \Big(T^V_{k,l} = \mathrm{A}\Big) \ \rightarrow \ R_{i,j,k,l} \tag{6}$$

$B^V_{i,j,k,l}$ is the required virtual bandwidth between two deployed VMs. Although there is a two-way communication between the dispatcher and an agent, we do not simply add up the packet processing rates of the communicating VMs to get the bandwidth. Instead, we double the processing rate of the agent. This is because, by design, the dispatcher will not forward more packets to an agent than it can handle. The dispatcher may need a higher bandwidth from the ingress point to itself, but it distributes the traffic to several agents which lessens the requirement of higher bandwidth on the other side. Thus, the required bandwidth between these two VMs should be, at least, twice as much as an agent can process. Consequently, if two VMs do not need to communicate, the virtual bandwidth between them should be zero.

$$R_{i,j,k,l} \ \rightarrow \ B^V_{i,j,k,l} \geq \ \min\big\{\big(2 \times P_{k,l}\big), \big(P_{i,j} + P_{k,l}\big)\big\} \tag{7}$$

$$\neg R_{i,j,k,l} \ \rightarrow \Big(B^V_{i,j,k,l} = 0\Big) \tag{8}$$

The bandwidth of each link on the path from the server implementing the dispatcher and the server implementing the agents should be sufficiently high so that the VMs in them can talk to each other. $\mathbb{Z}$ is the set of all links on the path from a server to another server. $B^P_{i,k,z}$ is the physical bandwidth of z^{th} link on the path from server *i* to server *k*. The physical bandwidth of each link should be no less than the virtual bandwidth required by all the communicating VMs that are using that link. This will ensure the required throughput for the communication among the VMs. We model the constraint considering δ as the percentage of the physical bandwidth that a user would like to allocate for virtual communication:

$$R_{i,j,k,l} \ \rightarrow \ \forall_{z \in \mathbb{Z}} \forall_{i,k} \sum_{j,l} B^V_{i,j,k,l} \leq \delta \times B^P_{i,k,z}. \tag{9}$$

3) Constraints on VNF Location: Sometimes there are certain service level agreements (SLA) with some customers about the placement of the VNFs. As the servers in a data center can be located in various locations, regulations may restrict the provider from placing the certain types of VNFs on certain servers. For example, an enterprise client may wish to keep the data and traffic within a comfortable geographical boundary. If $\mathbb{S}_l$ denotes the set of servers located in geographical location *l* and *L* is the total number of locations, then, $\mathbb{S} = \bigcup_{l=1}^{L} \mathbb{S}_l$. If $\mathbb{L}_t$ is the set of all permissible locations, and $\mathbb{L}'_t$ is the set of forbidden ones for a VNF of type *t*, then the VMs should be deployed on the servers located in benign locations:

$$\forall_i \forall_j (i \in \mathbb{S}_l) \wedge (l \in \mathbb{L}_t) \wedge \Big(T^V_{i,j} = t\Big) \ \rightarrow \ D^V_{i,j} \tag{10}$$

In an NFV environment, single point of failure has been an issue already discussed in literature [42]. These issues can be software- or hardware-based. If a single server contains all required VNFs, the overall system becomes vulnerable to hardware failure. We provision a reasonable distribution of VMs among all available servers to avoid such a failure. The number of VMs on a server should usually be below a threshold of the maximum possible VMs on that particular server. If γ is the threshold percentage defined by the provider, and $\mathrm{V}^{\max}$ is the maximum possible VMs in a server *i*,

$$\forall_i \sum_j D^V_{i,j} \leq \gamma \times \mathrm{V}^{\max}. \tag{11}$$

C. Agent and Dispatcher Specific Requirement Model

The following constraint ensures that if a VM is deployed, it is either a dispatcher or an agent:

$$D^V_{i,j} \ \rightarrow \ \Big(T^V_{i,j} = \mathrm{D}\Big) \vee \Big(T^V_{i,j} = \mathrm{A}\Big) \tag{12}$$

$$\neg D^V_{i,j} \ \rightarrow \ \Big(T^V_{i,j} \neq \mathrm{D}\Big) \wedge \Big(T^V_{i,j} \neq \mathrm{A}\Big) \tag{13}$$

The combined packet processing rate of all the agents should be no less than the incoming packet rate. Let $P^{\mathrm{A}}_{i,j}$ be the processing rate of j^{th} agent located on server *i*. If *P* is the total number of ingress points and R_p is the incoming packet rate at ingress point *p*, then the following holds:

$$\Big(T^V_{i,j} = \mathrm{A}\Big) \ \rightarrow \ \sum_{i,j} P^{\mathrm{A}}_{i,j} \geq \sum_{p=1}^{P} R_p \tag{14}$$

The total packet processing rate of all the dispatchers assigned for an ingress point should be at least equal to the incoming packet rate at that ingress point.

$$\Big(T^V_{i,j} = \mathrm{D}\Big) \ \rightarrow \ \sum_{i,j} P^{\mathrm{D},p}_{i,j} \geq R_p \tag{15}$$

We want the incoming traffic to be forwarded to the dispatchers first, which in turn, forward it to the agents. We provision one dispatcher per ingress point, but we try to minimize the total number of dispatchers. We consider the latency of each server from the ingress points, is a function of number of hops (transmission, processing and queuing delay at routers) and the propagation delay, as a metric to choose servers for dispatchers. The server that is closest to an ingress point should be chosen to deploy the corresponding dispatcher. In case the ingress points are in closer proximity, we may consider deploying one dispatcher for multiple ingress points. In that case, we maintain a threshold latency (L_{th}) that is greater than the latency between an ingress point and a server.

$$\forall_{p \in \mathbb{P}} \ \exists_{i \in \mathbb{S}} \ \Big(T^V_{i,j} = \mathrm{D}\Big) \ \rightarrow \ \big(L_{p,i} \leq \ L_{th}\big). \tag{16}$$

Algorithm 1 Optimal Network Synthesis

```
S_t^min := 0
S_t^max := S_t
if Solver returns SAT then
    Get Model, M and Get updated S_u
    S_t^max := S_u
    C = 0
    repeat
        S_t := (S_t^min + S_t^max)/2
        Update the constraints associated to S_t
        if Solver returns SAT then
            Get Model, M and Get updated S_u
            S_t^max := S_u
        else
            S_t^min := S_t
        end if
        C := C + 1
    until (S_t^max − S_t^min ≈ 0) ‖ (C = C^max)
end if
```

D. SMT Encoding, Query Formulation, and Solving the Model

The NFV network synthesis problem is formalized as the satisfaction of the conjunction of all the constraints in the equations in Section IV. We implement our model by encoding the system configuration and the constraints into SMT logics [36]. In this encoding purpose, we use the Z3, an efficient SMT solver [35].

The solver checks the verification constraints and provides a satisfiable (SAT) result if all the constraints are satisfied. The SAT result provides a SAT instance, which represents the value assignments to the parameters of the model. According to our objective, we require the assignments to the following variables: (i) the decision variable referring to whether a VM is deployed, $D^V_{i,j}$, *i.e.,* the placement of the VMs and (ii) the type of the deployed VMs, $T^V_{i,j}$. A 'true' value to $D^V_{i,j}$ means that VM *j* on server *i* is deployed, while integer values to $T^V_{i,j}$ suggests that VM *j* on server *i* is of a type corresponding to that integer. In our case, the integer values corresponding to the types are D and A.

E. Optimal Synthesis Determination

The synthesis result represents a comprehensive network deployment plan for the NFV architecture. There are usually more than one satisfiable model, which have different number of utilized COTS servers. The number of utilized servers will be less than the provided total number of COTS servers. If S_t is the total number of servers and S_u is the number of utilized servers, we can choose the most cost efficient deployment plan with minimum number of servers among all alternative satisfiable models for the same set of constraints. We assume that all the servers have similar configuration, so reducing the number of servers will reduce the available computing and network resources.

We use Algorithm 1 that updates the minimum (S^{min}_t) and maximum (S^{max}_t) values of the number of available servers, and finds the optimal network plan. Their values are used to update S_t, which is used to try for a more optimal solution. However, finding the optimal solution using the algorithm usually required more time than a single satisfiable model. The algorithm requires several invocations of the model solver and often confronts with UNSAT results, which usually take longer time. The time complexity of this algorithm becomes $\mathcal{O}(T \times \log_2 D)$, where *T* is the time for one model solution and *D* is the difference between S^{min}_t and S^{max}_t. For example, if we start with 20 available servers to process 140 Gbps of traffic, then the first SAT solution utilizes 8 of them. Algorithm 1 provides an optimal solution of 3 servers for this problem and takes around 24 minutes. The user can control the number of iterations of the model (C^{max}) and generate a sub-optimal solution if required. However, in scenarios where a quick solution is required, a single satisfiable solution is sufficient.

F. VFenceSynth in Dynamic Scenarios

Cyberattack patterns and intensity are very flexible and can change very frequently. These changes require the virtual defense solutions to be more flexible, and properly reconfigured and replaced, so that the product servers receive the same protection during and after changes. A defense mechanism should be designed to dynamically adapt to the attack changes.

Dynamic Behavior: VFenceSynth is capable of providing a solution within a sustainable period of time when these changes occur. However, it is not always desirable to furnish a completely new solution whenever the attack pattern varies. This is because of the Maximum Tolerable Period of Disruption (MTPOD) may be limited for the provider and the cost involved in shutting down some VMs and setting them up on new locations might degrade the quality of service. The dispatcher and the agents are generally associated with some flows and changing their positions requires a lot of handling for the NFV orchestrator. VFenceSynth can use a previously found solution as an input and generate a new output based on it. In essence, in the case of a small change in attack pattern, the deployment and location of the existing VMs do not change and additional agents are installed on a server or some agents are put to sleep while satisfying all the constraints. When the attack intensity change is above a certain threshold, we need to deploy everything from scratch. This threshold value can be determined by the user of the tool. This is an indicator of how much the user wants to tolerate without reimplementing the whole network topology.

When the amount of incoming packets decreases, it is possible to free up some VMs (agents). We are aware that the agents might still being applied to scrutinize some flows. We run a load balancing algorithm that checks for the load on the agents, *i.e.,* the number of flows handled by each agent. If the load for an agent is below a certain threshold, flows are migrated to some nearby agents and the less loaded agent is completely freed. It can, consequently, be put to sleep and the resources (M^S_i and C^S_i) for the corresponding server are updated for the next run of VFenceSynth, with the resources

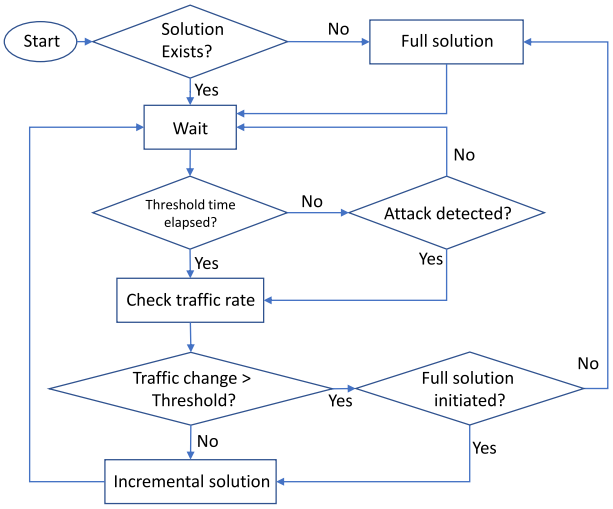


Fig. 4. VFenceSynth solution for dynamic update of NFV topology.

($M^V_{i,j,t}$ and $C^V_{i,j,t}$) that were allocated to and used by the agents that are retiring.

In case of an increase in incoming packet rate, we often need to add more agents to the system. If the increase is below a certain threshold (*e.g.,* 20%), we keep the existing solution. That means, the existing 'true' values of placement ($D^V_{i,j}$) and VM type ($T^V_{i,j}$) are kept unchanged. The decision variables that were 'false' in the previous solution, are made available for new assignments. VFenceSynth assigns new values to these new variables, so that the conjunction of equations 1 through 16 are still satisfiable. We refer to this as an incremental VFenceSynth solution. In some cases, the new VMs are installed on servers that already have some VMs running, while in other cases, they are deployed on new servers that were unused so far. In case of an unavailability of a new server when needed, VFenceSynth returns an UNSAT result.

In times of attack patterns or intensity change over the threshold proposed by the user, we run VFenceSynth from scratch and find a completely new solution. In the case of increase, some existing VMs need to be moved to other places with the network traffic being moved alongside. Several algorithms have been discussed in the literature when migrating VMs. However, these algorithms suffer from usage of high traffic buffering, which might have negative effects on the performance of the cybersecurity mechanisms [17]. We leave this for our future work, as it is beyond the scope of this paper.

Execution Scenario: The VFenceSynth solution can be of two types: one is a full solution, another one is an incremental solution utilizing the existing one, which take less time. Fig. 4 shows a flowchart of how the two types of VFenceSynth solutions are utilized. We build and deploy a new solution every user set interval (e.g., 30 min), or if the dispatchers report any cyberattack to the NFV orchestrator. In case of considerable traffic change, the incremental solution is deployed until a full solution is provided by VFenceSynth. This is because, the full solution may take some time to build; the network needs to be secured in the meantime.

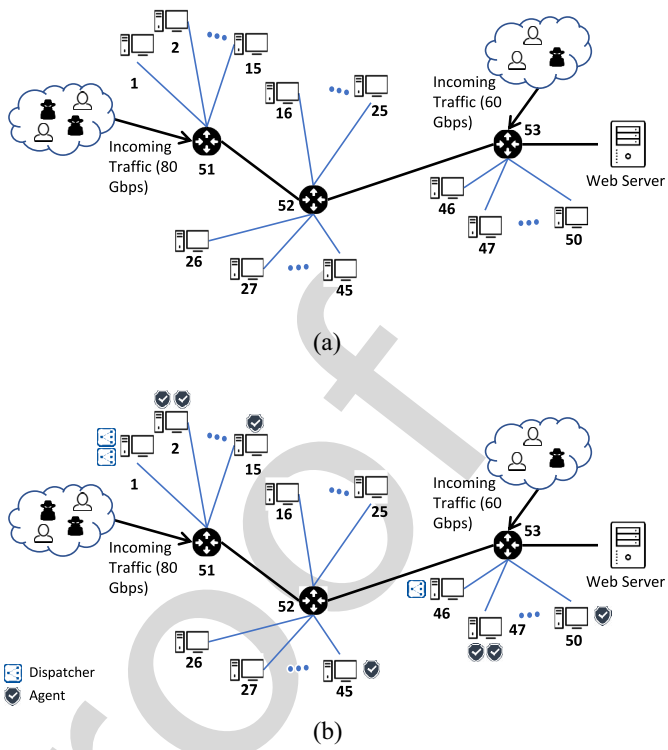


Fig. 5. (a) The physical network topology of the COTS servers and (b) The physical network topology of the servers and VMs implemented in them.

TABLE II
SAMPLE CODE

```
1. assert(=> (IsVMDeployed_0_0) (>= VMMemory_0_0 8))
2. assert(=> (not IsVMDeployed_0_0) (= VMCpu_0_0 0))
3. assert(>= (+ TotalAgentPackProcRate_0_0
    TotalAgentPackProcRate_0_1
    TotalAgentPackProcRate_0_2
    ... TotalAgentPackProcRate_0_14) 140)
4. assert(>= 16
    (+ VMMemory_0_0 VMMemory_0_1... VMMemory_0_4))
```

G. A Synthetic Case Study

DDoS Mitigation: In this section, we present an example case study of synthesizing the virtual network topology. A network of commodity servers is shown in Fig. 5(a). Table III presents the input file used in this case. In this example, there are 50 COTS servers in the NFV network that are connected to each other. From an input text file, we read the physical connectivity of these servers through routers in terms of the bandwidth of links and latency from the ingress points. Memory is provided in GB and CPU is in number of cores. In our case, communication is required between a dispatcher and an agent. Virtual bandwidth between any two dispatcher and agent must be in accordance with the bandwidth of the physical links. In this example, we consider two ingress points, each receiving 80 Gbps and 60 Gbps of traffic respectively. Table II shows some SMT-Lib code snippet generated by VFenceSynth. For example, line 1 in the code shows that if a VM is deployed, it should consume at least 8 GB of memory, while line 3 suggests that the combined packet processing rate of the agents should be at least 140, which is the traffic rate during an attack.

TABLE III
INPUT TO THE EXAMPLE

```
#Number of locations
5
#Preferred locations
1 2 3 5
# Number of physical/COTS server
50
# Memory (GB), CPU (#core), location and latency (ms) from of COTS servers
32 5 1 50
32 5 1 50
..
# Number of routers
3 # Router IDs: 51 52 53
# Number of links in the network
53
# Network topology: source, destination and link bandwidths (Gbps)
1 51 500
2 51 250
51 52 1000
..
# Number of ingress points and router IDs corresponding to them
2 51 53
# Incoming traffic rate (Gbps)
80 60
```

Full Solution: VFenceSynth gives a SAT result for this example. From the resultant SAT instance, we find the deployed VM types along with their placements in the servers. Fig. 5(b) shows the placements of the network functions. The result shows that three dispatchers need to be deployed on server 1 and 46, while several agents need to be installed on server 2, 15, 45 and so on. The dispatchers are deployed on servers that are closer to the ingress point in terms of latency. There are only three dispatchers required, because they only need to dispatch the whole incoming traffic, be it legitimate or an attack, to all the deployed agents. Hence, the dispatcher needs more resources than the agents. The packet processing rate of the dispatcher is, at least, equal to the incoming packet rate at its corresponding ingress point, while the combined packet processing rate of the agents is more than the total incoming rate. VFenceSynth also provides the required memory, CPU and the packet processing rate of each VM.

Incremental Solution: If we increase the incoming traffic to 70 Gbps at the second ingress point, which makes the total incoming traffic rate less than our example threshold value (20%), we observe that the existing placements of the agents remain the same except one new agent is deployed on a new server (server 5), although the existing servers had more resources available to accommodate the new VM. This is because of the bandwidth constraints associated with the servers already used. We may recall that the physical bandwidth of the links must be greater than or equal to the virtual bandwidth between an agent and the dispatcher.

V. FORMAL MODEL OF VTVSYNTH

This section discusses the formal model of the NFV architecture for QoS maintaining IPTV network. Table IV lists the variables used in the model.

A. Preliminary

An IPTV service provider utilizes commodity servers to set up VNFs working as STBs or RGs in a home network. These VNFs need to be efficiently set up on the servers so that the resources are utilized well as well as the service is provided with better quality.

TABLE IV
NOTATION TABLE FOR VTVSYNTH

Notation	Definition
$uType_{z,i}$	Type of customer i residing in zone z.
$uMem_{z,i}$	Memory used by customer$i$ in zone z.
$uSto_{z,i}$	Storage used by customer$i$ in zone z.
$uBW_{z,i}$	Bandwidth used by customer$i$ in zone z.
$uServer_{z,i}$	Server index assigned to customer$i$ in zone z.
$uRg_{z,i}$	RG index assigned to customer$i$ in zone z.
$uStb_{z,i}$	STB index assigned to customer$i$ in zone z.
$s_{z,j}$	j^{th} server in zone z.
$rg_{z,s,k}$	k^{th} RG in server s in zone z.
$stb_{z,s,l}$	l^{th} STB in server s in zone z.
$rgProc_{z,s,k}$	Data processing rate of k^{th} RG in server s in zone z.
$stbSto_{z,s,l}$	Storage of l^{th} STB in server s in zone z.
$stbMem_{z,s,l}$	Memory of l^{th} STB in server s in zone z.
$servSto_{z,j}$	Storage of j^{th} server in zone z.
$servMem_{z,j}$	Memory of j^{th} server in zone z.

B. IPTV Network Model

The type of each customer needs to be within a certain range. If *UT* is the maximum possible number of types for customers, then the following is true:

$$\forall_{z\in\mathbb{Z}}\forall_{i\in\mathbb{U}}\, u_{z,i} \rightarrow \big(uType_{z,i} \geq 1\big) \wedge \big(uType_{z,i} \leq UT\big) \tag{17}$$

For a particular type of customer, the memory assigned to that customer ($uMem_{z,i}$) should be less than or equal to the maximum possible memory UM_t for type *t*.

$$\forall_{z\in\mathbb{Z}}\forall_{i\in\mathbb{U}}\big(uType_{z,i} = t\big) \rightarrow \big(uMem_{z,i} \geq 1\big) \wedge \big(uMem_{z,i} \leq UM_t\big) \tag{18}$$

The same should be true for both utilized storage and bandwidth for a customer of a particular type. This is represented in the following two equations, where $uSto_{z,i}$ and $uBW_{z,i}$ are the utilized storage and bandwidth, respectively for the customer, while US_t and UB_t are the maximum possible storage and bandwidth for type *t*:

$$\forall_{z\in\mathbb{Z}}\forall_{i\in\mathbb{U}}\big(uType_{z,i} = t\big) \rightarrow \big(uSto_{z,i} \geq 1\big) \wedge \big(uSto_{z,i} \leq US_t\big) \tag{19}$$

$$\forall_{z\in\mathbb{Z}}\forall_{i\in\mathbb{U}}\big(uType_{z,i} = t\big) \rightarrow \big(uBW_{z,i} \geq 1\big) \wedge \big(uBW_{z,i} \leq UB_t\big) \tag{20}$$

If $uServer_{z,i}$ denotes the server index for a customers, where its STB and RG will be hosted, and *SC* is the maximum number of servers in a zone, then it can be represented by the following equation:

$$\forall_{z\in\mathbb{Z}}\forall_{i\in\mathbb{U}}\, u_{z,i} \rightarrow \big(uServer_{z,i} \geq 1\big) \wedge \big(uServer_{z,i} \leq SC\big) \tag{21}$$

If at least one customer is utilizing a server for its RG or STB, then that server should be on. If $s_{z,j}$ is a boolean variable

denoting if a server is on, then the following should be true:

$$\bigvee_i \big(uServer_{z,i} = j\big) \ \rightarrow s_{z,j} \tag{22}$$

Let $rg_{z,s,k}$ be a boolean denoting whether the k^{th} RG in zone z and server s is deployed. If there is at least one customer whose RG matches the index k and whose server matches the index s, then that particular RG should be deployed.

$$\bigvee_i \big(uRG_{z,i} = k\big) \wedge \big(uServer_{z,i} = s\big) \ \rightarrow rg_{z,s,k} \tag{23}$$

We maintain provision of shared STBs among customers who have similar subscription properties. For example, if two customers subscribe to the same package consisting of same number of channels, same storage and bandwidth capacities, they can share the same STB that can satisfy the combined requirements without raising complexity of resource allocation. If $stb_{z,s,l}$ denotes a boolean denoting whether the l^{th} STB in zone z and server s is deployed, then the following equation is true, which express that at least one customer has to use an STB for it to be deployed.

$$\bigvee_i \big(uSTB_{z,i} = l\big) \wedge \big(uServer_{z,i} = s\big) \ \rightarrow stb_{z,s,l} \tag{24}$$

Let the traffic processing rate of the k^{th} RG in zone z and server s be $rgProc_{z,s,k}$. The rate should exceed the total bandwidth requirements of all the customers using this RG.

$$rg_{z,s,k} \wedge \big(uRG_{z,i} = k\big) \ \rightarrow rgProc_{z,s,k} \geq \sum_i uBW_{z,i} \tag{25}$$

Similarly, the memory ($stbMem_{z,s,l}$) and storage ($stbSto_{z,s,l}$) of an STB which is deployed in zone z and server s should exceed the requirements of the customers using it.

$$stb_{z,s,l} \wedge \big(uSTB_{z,i} = l\big) \rightarrow \big(stbMem_{z,s,l} \geq uMem_{z,i}\big) \wedge \big(stbSto_{z,s,l} \geq uSto_{z,i}\big) \tag{26}$$

The following equation ensures that the total number of deployed RGs and STBs is less than the total number of customers in a particular zone:

$$\left(\sum_{s,k} rg_{z,s,k} \leq \sum_i u_{z,i}\right) \wedge \left(\sum_{s,l} stb_{z,s,l} \leq \sum_i u_{z,i}\right) \tag{27}$$

If $servMem_{z,j}$ denotes the memory of the j^{th} server in zone z, then it should exceed the total memory used by all the STB deployed on it.

$$servMem_{z,j} \geq \sum_{s,l} stbMem_{z,s,l} \tag{28}$$

Similarly, the storage of a server, $servSto_{z,j}$ should exceed the total storage of all the STBs deployed on this server.

$$servSto_{z,j} \geq \sum_{s,l} stbSto_{z,s,l}. \tag{29}$$

TABLE V
INPUT TO EXAMPLE 2

```
# Number of zones
5
# Number of clients in each zones
50 40 45 35 50
# Number of Client types
10
# Number of clients of each types for zones, where each row represents a zone
6 2 8 4 1 3 7 10 9
2 4 3 3 9 8 6 0 0 2
...
# Max memory (GB), max storage (TB), and max bandwidth (Mbps) for types
1 2 3
2 3 4
...
# Number of servers per zone
10
# Memory of servers (GB)
8 10 14 10 18 10 8 12 8 10
# Storage of servers (TB)
20 15 20 25 25 10 20 10 20 10
```

C. SMT Encoding, Query Formulation, and Solving the Model

For this model, the synthesis problem is formalized as the satisfaction of the conjunction of all equations in Section V-B. The solver checks the constraints and provides a SAT result if all the constraints are satisfied, which represents the value assignments to the parameters of the model. In this case, we require the assignments to the following variables: (i) the decision variable referring to whether an RG and STB are deployed, $rg_{z,s,k}$ and $stb_{z,s,l}$, *i.e.,* the placement of the VMs and (ii) the processing rates and storage of the VMs, as well as the servers: $rgProc_{z,s,k}$, $stbMem_{z,s,l}$, $servSto_{z,j}$, etc.

D. VTVSynth in Dynamic Scenarios

Typically, VTVSynth is run for a full new solution of VM placements whenever a service provider needs a deployment plan for all the customers located in several service zones. The user can set a threshold for changes in the customer requirements for which a full solution may not be required, rather, an incremental solution is adequate, where the current positions of all the VMs remain unchanged. For example, whenever a new customer subscribes for TV service, or changes its subscription plans, we do not need to run the full solution. VTVSynth can provide a solution keeping the existing positions of the VMs, and creating new locations for new VMs if required.

E. An Example Case Study

In this case study, we present a case to synthesize the virtual network functions in a home area network (HAN) for virtual IPTV solutions. Table V presents the input file used for this case. We consider 5 different zones with 50, 40, 45, 35 and 50 IPTV customers respectively. There are 10 different types of customers. The input files specifies the number of each types of customers in each zone. For example, in zone 1, there are 6 customers of type 1, 2 customers of type 2, and so on. Different types have different requirements, which are provided in the input file. For example, for type 1 customers, the maximum required memory is 1 GB, maximum storage is 2 TB, and the maximum bandwidth is 3 Mbps.

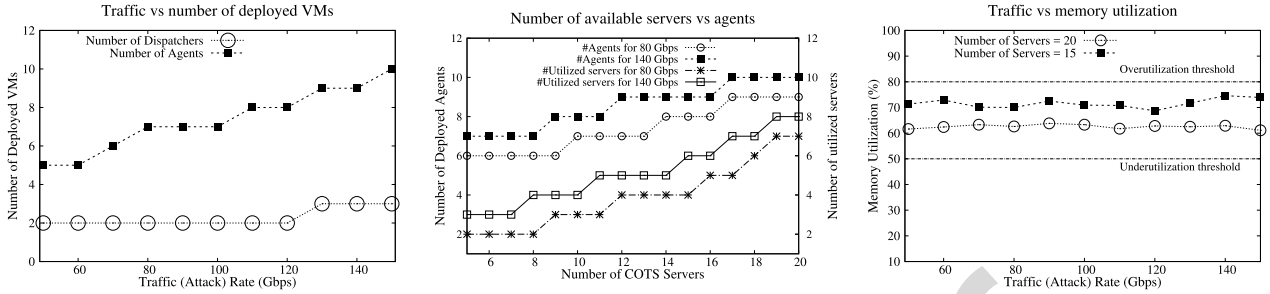


Fig. 6. For VFenceSynth (a) number of required VMs w.r.t. incoming traffic rate, (b) number of deployed agents and utilized servers w.r.t. number of available COTS servers, and (c) memory utilization w.r.t. incoming traffic rate.

The available number of servers per zone is provided along with their memory and storage. A service provider utilizes these servers for setting up the virtual RGs and STBs. In this problem scenario, vTVSynth provides a SAT result. The result provides values of the model variables that correspond to the placement of the virtual RGs and STBs satisfying all the requirements and constraints of the IPTV network. For example, it tells that the virtual RGs and STBs for this zone should be hosted on servers 1, 2 and 4. It also tells that the STB for customer 1 in zone 1 needs to be deployed on server 1 of this zone. Again, there should be 4 RGs for this zone, which should be set up on server 1 and server 2. The solution also provides the identification of servers used for each client, as well as the RGs and STBs assigned to them. According to the model, each client is assigned an STB, which meets the requirements of memory and storage of the client's type. For example, client 2 in zone 1 is of type 2, which has a memory requirement of 2 GB, and storage requirement of 3 TB. The STB assigned for this client meets these requirements. An RG is assigned to one or more clients, which processes all the incoming and outgoing traffic for these clients. The solution makes sure that the processing rate of an RG exceeds the total bandwidth requirements of all the clients using it.

VI. Evaluation

In this section, we present the evaluation of VFenceSynth and VTVSynth. We present the relationships among different parameters and the scalability of the developed tools.

A. Methodology

We ran experiments on different network topologies of different configurations and connectivity of 5−100 COTS servers to evaluate VFenceSynth. The servers are equipped with memory between 16−48 GB and CPU cores of 2−7. On the other hand, to evaluate VTVSynth, we performed experiments on different network topologies with different numbers (50−400) of customers of different types located in different numbers of zones. We considered at least one server per zone. The servers are equipped with memory ranging from 16−48 GB, and storage ranging from 10−30 TB. The tools were run on a machine running Windows 10 and equipped with an Intel Core i7 Processor and a 16 GB memory.

B. Relationships Between Deployment Parameters

To evaluate VFenceSynth, we increased the traffic rate, which includes the attack packets, gradually from 50 to 150 Gbps, and observed the number of deployed dispatchers and agents. This is demonstrated in Fig. 6(a). In this case, with the increment of traffic, the number of dispatchers increases very slowly, in comparison to the number of agents. It is possible for the same number of agents to process a certain range of traffic rate. In the figure, the number of agents remains the same between traffic rates of 80 and 100 Gbps. As the traffic rate passes beyond 100 Gbps, the number of agents increase to 8 from 7; it remains the same up to 120 Gbps.

Fig. 6(b) shows the number of deployed VMs, as well as the number of utilized servers, with respect to the number of total available servers in the network topology for a certain amount of incoming traffic (80 and 140 Gbps). As the number of available servers increases, the number of agents also increases slowly. The resource and bandwidth constraints are responsible for this, as the tool tries to find a solution utilizing all the prospective VMs. The number of candidate servers for deploying VMs increases in the system as there are more servers. Possible bottlenecks in bandwidth and single point of failures can be avoided by utilizing more servers. The graph demonstrates that the number of required agents and utilized servers for 140 Gbps of traffic is higher than that of 80 Gbps of traffic. We can also observe from the graph that the increasing rate of number of utilized servers is higher than the rate of increasing agents. This justifies our finding of Fig. 6(c). This graph shows the memory utilization of the utilized servers, which is the ratio of total memory of the VMs and memory of the utilized servers. The memory utilization is slightly higher for lower number of available COTS servers in the network. For a certain number of servers, the memory utilization remains almost constant as traffic increases. Also, we can observe that the memory of the servers is neither overutilized, nor underutilized. In this experiment overutilization threshold was set to 80%, while the underutilization was set to 50%.

We present the relationship between the number of deployed VMs (STBs and RGs) and the number of customers in Fig. 7(a). The figure shows that the number of deployed virtual STBs increases with the number of customers. Some STBs are shared between customers of exactly same requirements (e.g., same number of subscribed channels, same STB size, etc.).

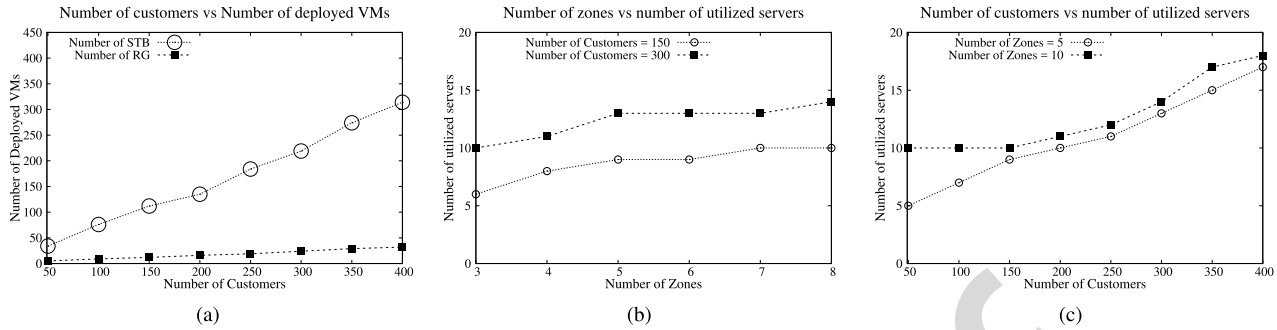


Fig. 7. For VTVSynth (a) Number of deployed VMs w.r.t. number of customers, (b) number of utilized servers w.r.t. number of zones, and (c) number of utilized servers w.r.t. number of customers.

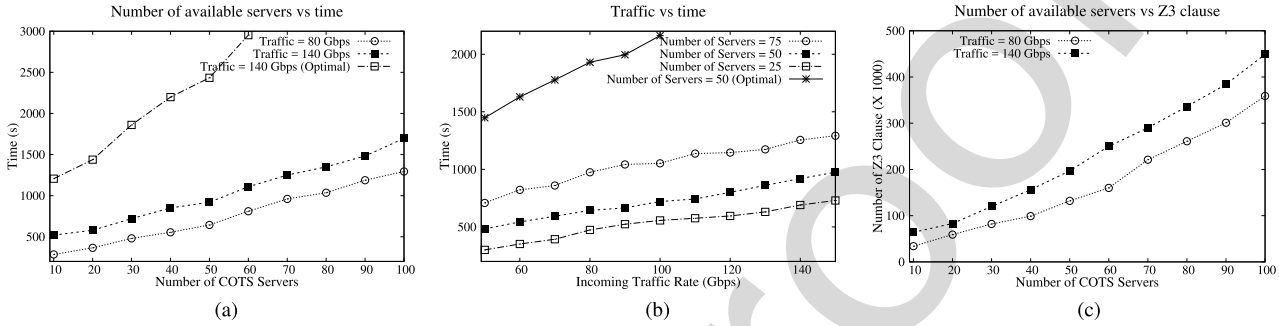


Fig. 8. (a) VFenceSynth synthesis time w.r.t. number of available COTS servers, (b) the model synthesis time w.r.t. incoming Traffic Rate (Gbps), and (c) number of z3 clauses created by VFenceSynth w.r.t. number of COTS servers.

The increment of number of RGs is slower compared to the increment of STBs. This entails that multiple customers can be served with a single RG, as long as the RG is capable to process that traffic for all its connected customers.

The number of utilized servers increase very slowly, and remains almost the same with the increment of number of zones, for a certain number of total customers. This is demonstrated in Fig. 7(b). The small increment of utilized servers for the same number of customers is due to the fact that at least one server in each zone needs to be used. With the increment of zones, there are more potential utilized servers, which needs to be used for customers distributed in these zones.

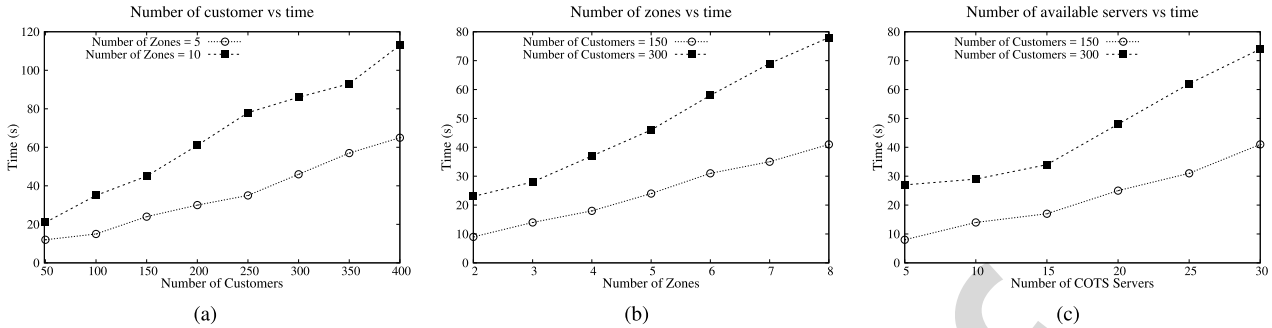
Fig. 7(c) presents the relationship between the number of customers and the total number of utilized servers. We can observe that the increment of number of utilized servers is almost linear with the increment of number of customers. We provision at least one server per zone in our modeling of VTVSynth. This is reflected in the figure, as well. For 5 zones, the minimum number of utilized servers is 5, while for 10 zones, it is 10.

C. Scalability Analysis

We evaluate the scalability of both VFenceSynth and VTVSynth by analyzing the time required to synthesize the virtual topology by varying the problem size. The synthesis time includes the model generation time and the constraint verification time for the first full solutions. We do not present any comparison with other works, as there is no other work that deals with the same problem of formal synthesis of NFV, to the best of our knowledge.

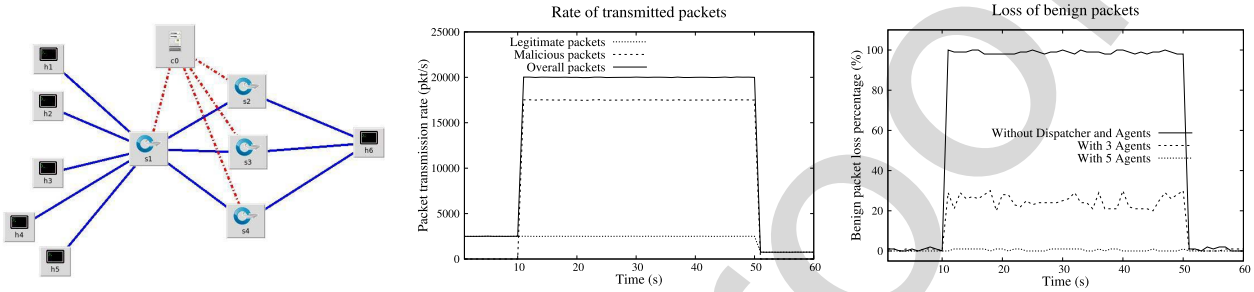
We also observe the model synthesis time by varying the number of available servers while keeping the incoming traffic rate constant in Fig. 8(a). We observe the time for scenarios with attack rate of 80 Gbps and 140 Gbps by increasing the number of servers from 10 to 100. It was observed that the time increases significantly with the increase of the number of servers. As the number of available servers increases, the problem size increases in terms of the number of possible flows between the dispatcher and an agent. Hence, the number of resource constraints increases with the increment of servers. Verification of more constraints is required as the model size increases, and more time is required to reach a solution. We also show the time for finding the optimal number of servers if the number of available servers is presented on the x-axis. We can observe that the time increases rapidly with the number of servers, as the time involves multiple solutions. If the network size is considerably large, the time for synthesizing the NFV topology will be infeasibly high. In such a case, we can divide the network into smaller subnets, divide the operation accordingly, and solve the individual problem for each subnet. The overall solution will be the combined NFV topology, although the result can be far from the optimal one. We leave this part for our future research, as this would take adding further constraints for collaboration among all the subnets.

The VFenceSynth model synthesis time with respect to the incoming traffic rate is shown in Fig. 8(b). Three scenarios with 25, 50 (including optimal) and 75 servers are presented in the graph. We observe that the required time increases



(a) (b) (c)

Fig. 9. (a) VTVSynth model synthesis time w.r.t. number of customers, (b) the model synthesis time w.r.t. number of zones, and (c) the model synthesis time w.r.t. number of available COTS servers.



(a) (b) (c)

Fig. 10. (a) The Mininet topology for simulation, (b) Packet transmission rate over simulation time, and (c) Benign packet loss over simulation time.

almost linearly with incoming traffic rate. As the attack traffic increases, added VMs need more resources to process the traffic. As a result, the constraints become more strict to solve. As a result, the solver takes more time to find a solution. It is worth mentioning that an incremental solution can provide a result in much quicker time. For example, a full solution for 50 servers processing 80 Gbps of traffic takes 643 seconds. However, if we consider a problem which solves only 10 Gbps of incremented traffic while keeping all other constraints the same, it takes less than a minute to solve the problem. The time to provide an incremental solution, however, depends on how many variables are assigned values from a previous solution and how many are made available for new assignments.

In Fig. 8(c), the number of clauses generated by the z3 solver was measured for a varying number of servers. These clauses correspond to the Z3 assertions that need to be satisfied. As the number of servers increases, the number of clauses also gradually increases. We observed that for a certain number of servers, there is not much difference in the number of generated clauses for 80 and 140 Gbps traffic.

In Fig. 9, we present the scalability of VTVSynth. As we increase the number of customers, the time required to synthesize the network increases almost linearly, as shown in Fig. 9(a). We can also observe that with the increase in the number of zones, the rate of increment of time increases. For example, synthesis time is more for 10 zones than 3 zones for the same number of customers. The rate of time increment increases more rapidly for 10 zones, as we increase the number of customers. With the increment in the number of zones, the number of available servers increases, and each server is associated with several clauses. As a result, the solver takes longer time to satisfy the constraints.

Fig. 9(b) shows the linear increment of time with respect to the increment of number of zones. As there are more different z3 clauses are created for more zones, the verification time increases. The same is true for increment of number of available servers. We notice that the time increases almost linearly with the increase of number of available servers. This is demonstrated in Fig. 9(c).

D. Simulation of VFenceSynth

We use the Mininet VM for the simulation of the effectiveness of VFenceSynth. We use a network topology consisting of several hosts, some Openflow-enabled switches that can be designed using software definition, as shown in Fig. 10(a). We use a Floodlight controller running on a separate VM for the mesh nature of the network topology. With the help of the controller, we dynamically add flow table rules to the switches to mimic the properties of the dispatcher and the agents. The dispatcher balances the load by distributing different flows to different agents, while the agents maintain a whitelist of legitimate sources. One of the hosts run a simple HTTP server (host 6) on port 80. Some of the hosts (host 1 and 2) act as malicious clients, while some of them (host 3, 4, and 5) are legitimate clients. The malicious clients perform a DDoS attack on the server by sending a large amount of SYN packets over a particular period of time. The legitimate clients try to establish a connection through the three-way TCP handshake by sending WGET messages, and are served with a *html* page.

We show the packet transmission rate from the clients over a period of 1 min in Fig. 10(b). As shown in the graph, the attacker hosts start the SYN flood attack to cause a DDoS attack at 10 sec and lasts until 50 sec. The total packet transmission rate jumps up to 20,000 packets/sec. We chose the metric of packet loss percentage for benign packets. Fig. 10(c) presents the packet loss for three cases: the first one is when there is no defense mechanism in place, the second and third ones are the loss while utilizing 3 and 5 agents respectively, when they have a limited queue/buffer size. In case of no defense, almost 100% legitimate packets are lost. In case of 3 agents, we have a loss ratio of about 30% throughout the attack period. For this attack intensity, VFenceSynth synthesizes a network of VMs consisting of 5 agents. Incorporating 5 agents yields almost no packet loss.

VII. Conclusion

NFV-based network systems are increasingly becoming popular for cyber-defense and QoS maintenance. We propose VFenceSynth and VTVSynth, two separate frameworks that deal with challenges in allocating resources to VMs that implement virtual network functions for security and quality maintenance. They consider the requirements and resource constraints, and formally models the NFV architecture synthesis problem. The solution to the model provides the placements and classifications of the VMs. We evaluate the frameworks in different test networks by a varying number of servers, traffic intensity, and customer specifications. The results show that the tools can generate feasible results within a sustainable period of time. In future, we would like to extend this research to solve the software-defined infrastructure synthesis problem where we can integrate the NFV architecture synthesis.

References

[1] *Network Functions Virtualization—HP Technical Whitepaper*. [Online]. Available: http://www.hp.com/hpinfo/newsroom/press_kits/2014/MWC/White_Paper_NFV.pdf

[2] *Network Functions Virtualization—Network Operators Perspective*. [Online]. Available: http://tinyurl.com/jhhctwv

[3] *Network Functions Virtualization—Use Cases*. [Online]. Available: https://tinyurl.com/yac6umpz

[4] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *Proc. 4th Int. Conf. Cloud Netw. (CloudNet)*, 2015, pp. 171–177.

[5] V. Aggarwal, X. Chen, V. Gopalakrishnan, R. Jana, K. K. Ramakrishnan, and V. A. Vaishampayan, "Exploiting virtualization for delivering cloud-based IPTV services," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2011, pp. 637–641.

[6] V. Aggarwal, V. Gopalakrishnan, R. Jana, K. K. Ramakrishnan, and V. A. Vaishampayan, "Optimizing cloud resources for delivering IPTV services through virtualization," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 789–801, Jun. 2013.

[7] S. Ayoubi, S. R. Chowdhury, and R. Boutaba, "Breaking service function chains with Khaleesi," in *Proc. IFIP Netw. Conf. Workshops*, 2018, pp. 64–72.

[8] S. Ayoubi, S. Sebbah, and C. Assi, "A cut-and-solve based approach for the VNF assignment problem," *IEEE Trans. Cloud Comput.*, to be published.

[9] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[10] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Proc. 11th Int. Conf. Netw. Service Manag. (CNSM)*, 2015, pp. 50–56.

[11] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman, "Implementation and performance analysis of various VM placement strategies in CloudSim," *J. Cloud Comput.*, vol. 4, no. 1, p. 20, 2015.

[12] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 1346–1354.

[13] J. Deng *et al.*, "VNGuard: An NFV/SDN combination framework for provisioning and managing virtual firewalls," in *Proc. IEEE Conf. NFV SDN*, 2015, pp. 107–114.

[14] N. Egi *et al.*, "Understanding the packet processing capability of multi-core servers," Intel, Santa Clara, CA, USA, Rep., 2009.

[15] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defense," in *Proc. 24th USENIX Security Symp.*, 2015, pp. 817–832.

[16] C. J. Fung and B. McCormick, "VGuard: A distributed denial of service attack mitigation method using network function virtualization," in *Proc. 11th Int. Conf. CNSM*, 2015, pp. 64–70.

[17] A. Gember-Jacobson *et al.*, "OpenNF: Enabling innovation in network function control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 163–174, 2014.

[18] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[19] S. Hauger *et al.*, "Packet processing at 100 Gbps and beyond—Challenges and perspectives," in *Proc. ITG Symp. Photon. Netw.*, 2009, pp. 1–10.

[20] H. Hawilo, M. Jammal, and A. Shami, "Orchestrating network function virtualization platform: Migration or re-instantiation?" in *Proc. IEEE 6th Int. Conf. Cloud Netw. (CloudNet)*, 2017, pp. 1–6.

[21] N.-F. Huang, S.-J. Wu, I.-J. Liao, and C.-W. Lin, "Bandwidth distribution for applications in slicing network toward SDN on vCPE framework," in *Proc. 18th Asia–Pac. Netw. Oper. Manag. Symp. (APNOMS)*, 2016, pp. 1–4.

[22] E. Hysenbelliu and M. Teresa, "A cloud based architecture for IPTV as a service," in *Proc. Balkan Conf. Informat. Adv. ICT*, 2015, pp. 59–64.

[23] A. H. M. Jakaria, M. A. Rahman, and C. J. Fung, "Automated synthesis of NFV topology: A security requirement-oriented design," in *Proc. 13th Int. Conf. Netw. Service Manag. (CNSM)*, 2017, pp. 1–5.

[24] A. H. M. Jakaria, W. Yang, B. Rashidi, C. Fung, and M. A. Rahman, "VFence: A defense against distributed denial of service attacks using network function virtualization," in *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, 2016, pp. 431–436.

[25] S. Khandelwal. *602 Gbps! This May Have Been the Largest DDoS Attack in History*. [Online]. Available: http://thehackernews.com/2016/01/biggest-ddos-attack.html

[26] B. Krebs. *KrebsOnSecurity Hit With Record DDoS*. [Online]. Available: http://www.ibtimes.co.uk/biggest-internet-attack-history-threatens-critical-infrastructure-450969

[27] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (IEEE INFOCOM)*, 2016, pp. 1–9.

[28] X. Li and C. Qian, "An NFV orchestration framework for interference-free policy enforcement," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2016, pp. 649–658.

[29] M. Liyanage, I. Ahmad, M. Ylianttila, A. Gurtov, A. B. Abro, and E. M. de Oca, "Leveraging LTE security with SDN and NFV," in *Proc. IEEE 10th Int. Conf. Ind. Inf. Syst. (ICIIS)*, 2015, pp. 220–225.

[30] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, 2015, pp. 98–106.

[31] T. Lukovszki, M. Rost, and S. Schmid, "It's a match! Near-optimal and incremental middlebox deployment," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 1, pp. 30–36, 2016.

[32] G. Marchetto, R. Sisto, J. Yusupov, and A. Ksentinit, "Formally verified latency-aware VNF placement in industrial Internet of Things," in *Proc. 14th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, 2018, pp. 1–9.

[33] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *J. Netw. Comput. Appl.*, vol. 66, pp. 106–127, May 2016.

[34] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, 2014, pp. 7–13.

[35] L. D. Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Proc. Int. Conf. Tools Algorithms Construct. Analysis Syst.*, 2008, pp. 337–340.

[36] L. D. Moura and N. Bjørner, "Satisfiability modulo theories: An appetizer," in *Proc. Brazil. Symp. Formal Methods*, 2009, pp. 23–36.
[37] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, 3rd Quart., 2014.
[38] A. Pastor and D. Lopez, "Access use cases for an open OAM interface to virtualized security services," 2014.
[39] A. D. Rayome. *DDoS Attacks Increased 91% in 2017 Thanks to IoT*. [Online]. Available: https://www.techrepublic.com/article/ddos-attacks-increased-91-in-2017-thanks-to-iot/
[40] Y. Rebahi *et al.*, "Virtual security appliances: The next generation security," in *Proc. Int. Conf. Commun. Manag. Telecommun. (ComManTel)*, 2015, pp. 103–110.
[41] L. Rizzo, G. Lettieri, and V. Maffione, "Speeding up packet I/O in virtual machines," in *Proc. 9th ACM/IEEE Symp. Archit. Netw. Commun. Syst.*, 2013, pp. 47–58.
[42] Y. Tamura, K. Sato, S. Kihara, and S. Moriai, "Kemari: Virtual machine synchronization for fault tolerance," in *Proc. USENIX Annu. Tech. Conf. (Poster Session)*, 2008.
[43] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.
[44] S. Yu, Y. Tian, S. Guo, and D. O. Wu, "Can we beat DDoS attacks in clouds?" *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2245–2254, Sep. 2014.

A. H. M. Jakaria received the B.S. degree in computer science and engineering from the Bangladesh University of Engineering and Technology, Dhaka, in 2009. He is currently pursuing the Ph.D. degree in computer science with Tennessee Tech University, USA. He is actively involved with CESR and CEROC, Tennessee Tech as a Graduate Research Assistant. His primary research area includes information and network security for NFV and SDN. He is also interested in resiliency issues in UAV and IoT networks. He focuses on the formal modeling of the problems and solving them efficiently for automated synthesis of network topology and management strategies.

Mohammad Ashiqur Rahman received the B.S. and M.S. degrees in computer science and engineering from the Bangladesh University of Engineering and Technology, Dhaka, in 2004 and 2007, respectively, and the Ph.D. degree in computing and information systems from the University of North Carolina at Charlotte in 2015. He is an Assistant Professor with the Department of Electrical and Computer Engineering, Florida International University, USA. His research area covers a wide area of computer networks that includes computer and information security in both cyber and cyber-physical systems, formal security analysis, risk assessment and security hardening, secure and dependable resource management, and distributed computing.

Carol Fung received the B.S. and M.S. degrees in computer science from the University of Manitoba, Canada, and the Ph.D. degree in computer science from the University of Waterloo, Canada. She is an Assistant Professor with Virginia Commonwealth University. Her research area is network management and cybersecurity, including trust management, resource allocation, game theory, Bayesian inference theory, and crowdsourcing. Her research has applications in SDN/NFV networks, 5G networks, cyber security, and smartphone networks. She was a recipient of the IEEE/IFIP IM Young Professional Award in 2015, the University of Waterloo Alumni Gold Medal in 2013, and the Best Paper Awards in IM/NOMS.