A Connectionist Approach to Dynamic Resource Management for Virtualised Network Functions

Rashid Mijumbi*, Sidhant Hasija*, Steven Davy*, Alan Davy*, Brendan Jennings* and Raouf Boutaba‡
‡D.R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada
*Telecommunications Software and Systems Group, Waterford Institute of Technology, Ireland

***Abstract*—Network Functions Virtualisation (NFV) continues to gain attention as a paradigm shift in the way telecommunications services are deployed and managed. By separating Network Functions (NFs) from traditional middleboxes, NFV is expected to lead to reduced CAPEX and OPEX, and to more agile services. However, one of the main challenges to achieving these objectives is on how physical resources can be efficiently, autonomously, and dynamically allocated to Virtualised Network Functions (VNFs) whose resource requirements ebb and flow. In this paper, we propose a Graph Neural Network (GNN)-based algorithm which exploits Virtual Network Function Forwarding Graph (VNF-FG) topology information to predict future resource requirements for each Virtual Network Function Component (VNFC). The topology information of each VNFC is derived from combining its past resource utilisation as well as the modelled effect on the same from VNFCs in its neighbourhood. Our proposal has been evaluated using a deployment of a virtualised IP Multimedia Subsystem (IMS), and real VoIP traffic traces, with results showing an average prediction accuracy of 90%. Moreover, compared to a scenario where resources are allocated manually and/or statically, our proposal reduces the average number of dropped calls by at least 27% and improves call setup latency by over 29%.**

Keywords*—Network Functions Virtualisation, Resource Management, Artificial Intelligence, Topology Awareness, Neural Networks, Machine Learning.

I. Introduction

Service provision in the telecommunications industry has traditionally been based on the use of specialised Network Appliances (NAs) for each NF. This tight coupling usually means that even slight changes in the operation of a given NF could necessitate replacement of the NA on which it runs. This short lifetime of the NAs leads to increased Capital Expenses (CAPEXs). In addition, the fact that NAs are specialised calls for specialised maintenance and limits flexibility, leading to increased Operating Expenses (OPEXs). These issues, combined with the need for strict adherence to regulations, network stability and service quality, usually lead to extended product development cycles. Moreover, due to the fierce and ever increasing competition from services provided over-the-top, Telecommunications Service Providers (TSPs) have found themselves with consistently reducing average revenues per user, and therefore declining profitability. Therefore, TSPs are faced with an urgent need to find innovative and less expensive ways to increase and/or efficiently utilise network capacity and functionality, and achieve better service agility.

NFV [1], [2] has been proposed as a possible path towards this end. The main idea of NFV is to take advantage of recent advances in virtualisation technologies to decouple NFs (e.g. firewall, load balancing) from dedicated NAs so as to run them in generic servers which may be located in datacenters or at centralised TSP points of presence. Thanks to NFV, different NFs can evolve independent of each other, and of hardware. Furthermore, by running VNFs in virtualised resources (e.g. Virtual Machines (VMs)), network resources can be efficiently allocated through dynamic scaling. Finally, NFV promises to lead to more efficient operations through automated and centralised management of networks and services.

However, NFV is still in infancy and making its anticipated gains a reality still faces a number of challenges. One of the most important of these challenges relates to efficiently and autonomously managing resources that are allocated to VNFs [3]. Specifically, there is need for algorithms to determine how resources from the Network Functions Virtualisation Infrastructure (NFVI) are shared among the VNFs. These algorithms should have capabilities of scaling VNF resources vertically and/or horizontally while meeting two conflicting objectives. On one hand, VNFs should be allocated enough resources at all times to meet service quality requirements. On the other hand, only the needed amount of resources should be allocated to the VNFs to ensure efficiency. Given that network traffic and hence the load of such VNFs vary over time, and since spinning-up new resources (horizontal scaling) may take some time (in case the VNFs run in VMs), there is need for an automated way of determining such resource needs ahead of time so that resources are availed when needed without causing system outages or inefficiently using them.

In this paper, we propose a topology-aware, dynamic and autonomous system for managing resources in NFV based on the concept of GNNs [4]. Our proposal is motivated by the fact that in a Service Function Chain (SFC), network traffic traverses VNFs in a sequence. This implies that resource requirements of a given VNF may depend on those of other VNFs in the chain. Therefore, we start by modelling each VNFC[1] in a SFC as two parametric functions, each implemented by a Feedforward Neural Network (FNN). The task of each pair of FNNs representing a given VNFC is to learn (in a supervised way) the trend of resource requirements of the VNFC. This is achieved by combining historical local VNFC resource utilisation information with the information collected from its neighbours to forecast future resource requirements of the VNFC. In particular, the first FNN expresses the dependence of the resource requirements of each VNFC on the resource

[1]A VNFC is an internal component of a VNF which provides a VNF provider a defined sub-set of that VNF's functionality, with the main characteristic that a single instance of this component maps 1:1 against a single virtualisation container [5]. According to the ETSI, while VNF implementations must be standard and hence expose standard interfaces, VNFC implementations may be VNF provider specific.

978-3-901882-85-2

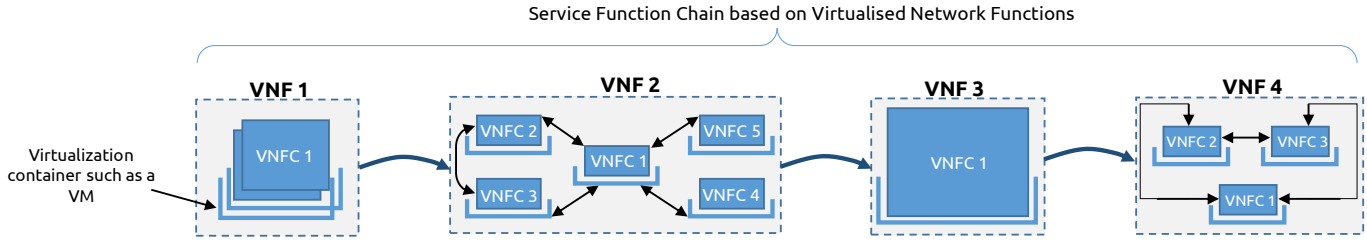

Fig. 1. NFV Service Function Chain. VNF 1 has a single VNFC while VNF 2 has multiple VNFCs. The VNFCs may be horizontally or vertically scaled. While VNFs are connected to each other by directed links in a chain, the VNFCs may contain both directed and un-directed links, in a vendor-specific topology.

requirements of VNFCs in its neighbourhood. This is input into the second FNN which forecasts the resource requirements of the VNFC. The resource requirement forecast is in turn used to automatically spin-up and configure new VNFCs or turn them off as required, just in time. To the best of our knowledge, dynamic and automated management of resources in NFV is still an open research problem, and learning techniques based on artificial intelligence are particularly interesting possible solutions.

The rest of this paper is organised as follows: We describe the problem in Section II and introduce GNNs in Section III. The proposed GNN-based resource allocation model and the corresponding learning algorithm are detailed in Sections IV and V respectively. Our proposal is evaluated in Section VI, related work discussed in Section VII, and the paper concluded in Section VIII.

II. Problem Description

The delivery of end-to-end services often requires packets, frames, and/or flows to traverse an ordered or partially ordered set of abstract NFs in what is known as an SFC. In NFV, such NFs are deployed in virtualised resources, and are hence known as VNFs. An example of such a SFC is shown in Fig. 1, in which the SFC is composed of 4 VNFs each connected to others by a directed link. Each VNF may be composed of one or more VNFCs, each hosted in a virtualisation container (virtual machines, linux containers, etc.). The VNFCs in a VNF are linked to each other by a combination of directed and undirected links, and work together to provide the required functionality of the VNF. Throughout this paper, Fig. 1, and in particular VNF 2 and its internal structure (the topology of constituent VNFCs) will be used as a running example to illustrate various aspects of our proposal. However, we use such specific and simple illustrations only to enhance clarity for the reader. Our proposal can be applied with ease to any SFC whose topology can be represented in the form shown in Fig. 1.

In order to have the SFC shown in Fig. 1, a number of problems should be solved. First, physical infrastructure must be deployed. Then, there must be algorithms to optimise the placement of virtual containers (or VNFs) onto the available physical servers. Finally, throughout the lifetime of the SFC, it is necessary to determine the actual amount of resources allocated to each virtualisation container and/or how many virtualisation containers are used for each VNFC. These three problems are refered to as server placement, function placement, and dynamic resource allocation respectively [3]. Server placement and function placement have already attracted a lot of attention, for example in [6], [7] and [8], [9], [10] respectively, and are out of scope for this paper.

In this paper, we focus on dynamic resource allocation. We consider that the VNFs (and hence VNFCs) have already been placed/mapped in the respective virtual resources on which they run. This work is motivated by the fact that the resource requirements of each VNF change over time with changes in traffic, which calls for ways of increasing and reducing resources allocated to the VNFCs as needed. Even more, since there is a non-negligible delay in spinning-up new resources (such as VMs), waiting until the system is over-loaded so as to scale resource up could negatively impact user QoS. In addition, having to wait until the load has fallen below a certain level to scale resources down could lead to inefficient resource utilisation. Moreover in complex and big networks, the scale involved cannot be managed manually, or this would end up defeating one of the main selling points of NFV - which is the flexibility and efficiency that comes with scaling resources up and down. Undoubtedly, automating resource management actions is a critical requirement for the success of NFV. Such a resource management approach should ensure that while the VNFCs have enough resources allocated to them at all times so as to meet quality of service requirements, that these resources are not left idle during periods of low resource utilisation. This paper makes a contribution to this end. The next two sections introduce the concept of GNNs and how it has been used to develop a system that forecasts the resource requirements of each VNFC, in order to obtain advance information of the VNFC's upcoming resource needs, allowing an orchestration entity to satisfy such needs just in time.

III. Graph Neural Networks

GNNs [4] are a supervised learning model aimed at solving problems in the graphical domain. The main idea of GNNs is to define each node n in the graph based on its *features*, f_n, and to complement this by the information (features) observed in the *neighbourhood*, $n^\star$, of the node. While there may be different definitions of neighbourhood, what is used in this paper is a set of nodes directly connected to node n. Using these two information sources, the GNN model determines a *state* s_n for each node n, which is then used to determine an *output* o_n for the same node. The determination of the state and output for each node is governed by equations (1) and (2) respectively.

$$s_n = \sum_{m \in n^\star} h_w\Big(f_n, f_m, s_m\Big), \forall n \tag{1}$$

$$o_n = g_w\Big(s_n, f_n\Big), \forall n \tag{2}$$

where f_m and s_m are the features and state of neighbour $m \in n^*$ respectively. It is possible to also include the features f_{mn} of the direct link between n and m in equation (1) only resulting in a problem with more dimensions. h_w and g_w are parametric functions which express the dependence of the state at each node on the state of its neighbourhood, and the dependence of the node output on its state, respectively. h_w is known as the *transition function* while g_w the *output function*. Equations (1) and (2) represent the activity of a network consisting of units which compute h_w and g_w for each node. This is the main idea of GNNs, an information diffusion mechanism, in which a graph is processed by a set of units (h_w and g_w), each one corresponding to a node of the graph, which are linked according to the graph connectivity. These units update their states and exchange information until they reach a stable equilibrium. Interested readers are referred to [4] for more details about the model. It should suffice to say here that by directing the diffusion process, the model is expected to converge exponentially fast, and be stable while determining the node states, and hence the GNN output.

IV. GNN-based Dynamic Resource Management

Since neighbouring VNFCs will usually be part of the same SFC, resource fluctuations at one VNFC are expected to influence resource requirements at its neighbours as traffic flows from one VNFC to the other. This dependency of VNFCs on their neighbourhood makes the connectionist approach derived from the GNN model an interesting fit as an approach for managing resources in NFV. Therefore, the GNN-based dynamic resource management system proposed in this paper is derived from equations (1) and (2), and is shown in Fig. 2 for a single VNFC. As can be seen, the system is comprised of four main components: (1) SFC features, (2) VNFC states, (3) state computation, and (4) output computation. In what follows, these components are described[2].

A. SFC Feautures

The SFC features are the observations or monitoring data from the VNFCs, and consitute the input to both h_w and g_w. In an NFV environment, these features represent the network parameters (such as CPU or RAM utilisation levels) that can be measured. As proposed by ETSI [11], the SFC in Fig. 1 may be represented as a VNF-FG. In this paper, we consider the resulting VNF-FG at the granularity of VNFCs, i.e., the nodes represented in the VNF-FG are VNFCs rather than VNFs.

Specifically, we model a SFC as a directed graph $G(N, L)$, where N represents the set of VNFCs and L the set of links between these VNFCs. An example of such a representation is given in Fig. 3 which is based on the SFC in Fig. 1. As can be seen in the figure, any given subset of VNFCs make up a VNF (e.g. n_1, n_2, n_3, n_4 and n_5 make up VNF 2 from Fig. 1). Each VNFC $n \in N$ has a set of features $f_n \in \mathbb{R}^{D_N}$ which represent a measurable resource for the VNFC, such as VNFC memory m_n, CPU c_n, processing delay d_n, etc. In

[2]It is worth noting that Fig. 2 only shows the model for a single VNFC. Such a system would have to be duplicated for each VNFC in a given SFC, with the resulting topology being based on that of the VNF-FG.

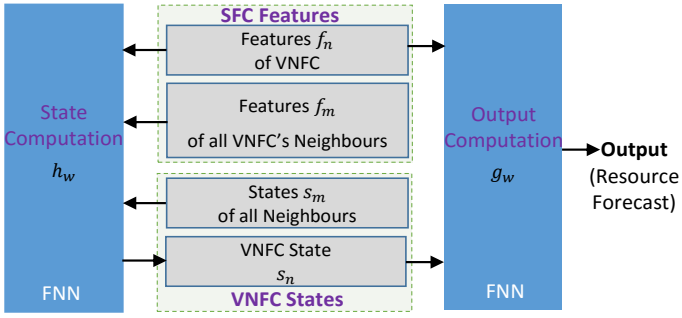

Fig. 2. GNN-based Resource Forecasting Model for a single VNFC

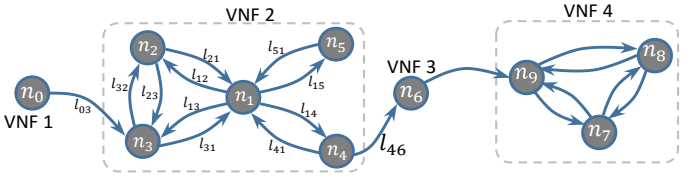

Fig. 3. SFC Modelling: VNFC Directed Graph

the same way, each link $l_{nm} \in L$ which connects VNFC n to m is characterised by a set $f_{nm} \in \mathbb{R}^{D_L}$ of features, which could represent link delay d_{nm}, bandwidth b_{nm}, etc. D_N and D_L refer to the dimensions of the feature sets for VNFCs and links respectively. Equations (3) and (4) show example feature sets for VNFC n and link l_{nm} respectively, for which $D_N = 3$ and $D_L = 2$.

$$f_n = \begin{bmatrix} c_n \\ m_n \\ d_n \end{bmatrix} \tag{3}$$

$$f_{nm} = \begin{bmatrix} b_{nm} \\ d_{nm} \end{bmatrix} \tag{4}$$

The objective is to monitor the features of each VNFC over time, and to use such historical observations, as well as the historical observations from the VNFC's neighbours to predict its subsequent features, which – in this case – represent future VNFC resource requirements. In order to define both historic and future resource utilisation, we refer to the VNFC and connected link features at (discrete) time step t by $f_n(t)$ and $f_{nm}(t)$ respectively. At any time t, we should be able to predict future resource utilisation using a finite horizon of past resource utilisation measurements. We denote the number of past measurements included in such a horizon as π. An example of current ($c_n(t)$, $m_n(t)$ and $d_n(t)$) and π previous measurements is shown by the vectors in equations (5) and (6). Using the observations represented by equations (5) and (6), the objective is to predict – say – the CPU requirement $c_n(t+\tau)$ of VNFC n at a time τ time steps after t. In the rest of this paper, wherever f_n or f_{nm} is used, it should be interpreted to mean the set containing $f_n(t)$ or $f_{nm}(t)$ plus the full history of features over the period π.

It is important to note that modelling of links and their features is only included here for completeness of the model, as the link features will not be used as neighbourhood information for VNFCs. The reason is that we consider that the resource utilisation profile of a directed link is directly dependent on

that of the VNFC at its source from which the traffic originates, and hence, the information obtained from a VNFC would be similar to that obtained from the link.

$$f_n(t) = \begin{bmatrix} c_n(t) \\ m_n(t) \\ d_n(t) \\ . \\ . \\ . \\ c_n(t-\pi) \\ m_n(t-\pi) \\ d_n(t-\pi) \end{bmatrix} \quad (5) \qquad f_{nm}(t) = \begin{bmatrix} b_{nm}(t) \\ d_{nm}(t) \\ b_{nm}(t-1) \\ d_{nm}(t-1) \\ . \\ . \\ . \\ b_{nm}(t-\pi) \\ d_{nm}(t-\pi) \end{bmatrix} \quad (6)$$

B. VNFC States

In line with the VNF design patterns proposed by ETSI [5], we consider that VNFCs can be *stateful*, with each VNFC $n \in N$ having a state $s_n \in \mathbb{R}^{S_D}$ of dimension S_D. The state s_n is derived from combining the features of a given VNFC with those from other VNFCs in its neighbourhood using the function h_w. This implies that the state of a given VNFC is dependent on the topology or connectivity of the VNF-FG. Such topology-awareness is represented in Fig. 4 which shows the dependencies of VNFCs in VNF 2 on each other. The Fig. depicts that, for example, the state s_1 of VNFC n_1 is dependent on the states s_2, s_3, s_4, and s_5 of all directly connected VNFCs, as well as the corresponding features f_2, f_3, f_4, and f_5. The state s_n is determined using equation (1). This means that, considering Fig. 4, the state s_3 of VNFC n_3 is given by (7).

$$s_3 = h_w(f_3, f_2, s_2) + h_w(f_3, f_1, s_1) \quad (7)$$

C. State Computation

State computation involves using equation (1) to determine the state for each VNFC. However, as can be observed from the equation, for any given pair of directly connected VNFCs, the state of each of them depends on that of the other. Therefore, the main task of state computation is to find a method to solve equation (1). The existence and uniqueness of a solution to (1) is guaranteed by Banach's fixed-point theorem [4], [12]. However, this requires that the global function h_w is a contraction map with respect to s, i.e., equation (8) must hold for some constant $0 \leq \rho < 1$ and any two state vectors $s_a, s_b \in \mathbb{R}^{S_D}$, where $||.||$ represents a vector norm.

$$||h_w(s_a) - h_w(s_b)|| \leq \rho||s_a - s_b|| \quad (8)$$

When equation (8) is satisfied, state computation is achieved using a classic iterative scheme given in equation (9) where $s(i)$ is the i^{th} iteration of the computation. This way, the function h_w stores the current state $s(i)$, and when called, calculates the next state $s(i+1)$. It can be observed that this makes the current state $s_n(i)$ of a VNFC n dependent on the previous state $s_m(i-1)$ of its neighbour m.

$$s_n(i+1) = \sum_{m \in n^\star} h_w\Big(f_n, f_m, s_m(i)\Big), \forall n \quad (9)$$

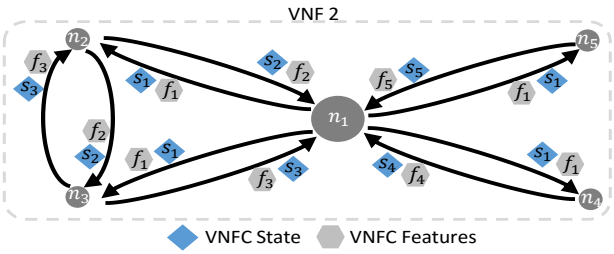

Fig. 4. States and Features from VNFC Neighbourhood. Each VNFC receives as input the state and features from all VNFCs that have a directed edge towards it. It is worth noting that the neighbourhood effects of VNFC's 1 and 3 on 2 have been omitted from this figure and all the proceeding analyses only for brevity, and keeping the representations simple. However, the proposed solution takes into account the dependencies of all the VNFCs in the SFC.

Moreover, the dynamic system containing equation (9) for all VNFCs in the SFC converges exponentially fast to the solution of equation (9), i.e., convergence to the fixed point [4]. The solution is equal to the convergence point of equation (9) for any initial value $s(0)$. In this paper, a FNN is used as h_w. This way, we can ensure that h_w is a contraction map by limiting its parameters, i.e., the range of values that the weights w of the FNN can take on [13]. As will be discussed in the next section, this is achieved by using an *error function* designed with this requirement in mind.

D. Output Computation

Output computation involves taking as input the states calculated by the h_w functions in the previous subsection, and combining it with the feature set of the VNFC to forecast a future resource requirement. The final output (forecast resource requirement) of a given VNFC is produced by another unit, which implements g_w for all VNFCs using equation (10).

$$o_n(i) = g_w\Big(s_n(i), f_n\Big), \forall n \quad (10)$$

The function g_w can be any general parametric function as long as it can be trained in a supervised manner, and the gradient of its output with respect to its input can be calculated. The original model proposed by [4], which is also adopted in this paper, uses a FNN for g_w.

Summary: To summarise, the proposed model is defined by equations (9) and (10) which takes as input the resource utilisation observations (features) of a SFC and outputs, for each VNFC, a forecast for the specified resource requirement. The interaction between equations (9) and (10) is illustrated in Fig. 5 for VNFC 2. It can be observed that we replace each VNFC with a pair of parametric functions h_w and g_w. In fact, we can say that each VNFC has one g_w function and as many h_w functions as it has neighbours, with each h_w computing the effect of the neighbour on the VNFC's state. These effects are then summed up in line with equation (9). As already discussed, each h_w or g_w is implemented by a FNN. Each h_w function stores the current state of the VNFC, and, when activated, calculates the next state using the observed VNFC features and the information (features and state) from a given neighbour. The g_w uses the state obtained from combining all h_w outputs to determine the final output (resource forecast) of the VNFC.

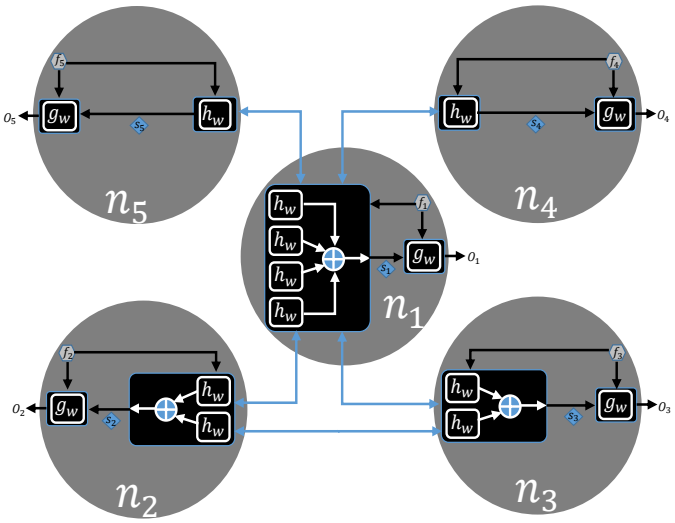

Fig. 5. GNN-based model for VNF 2: Each VNFC replaced by two functions

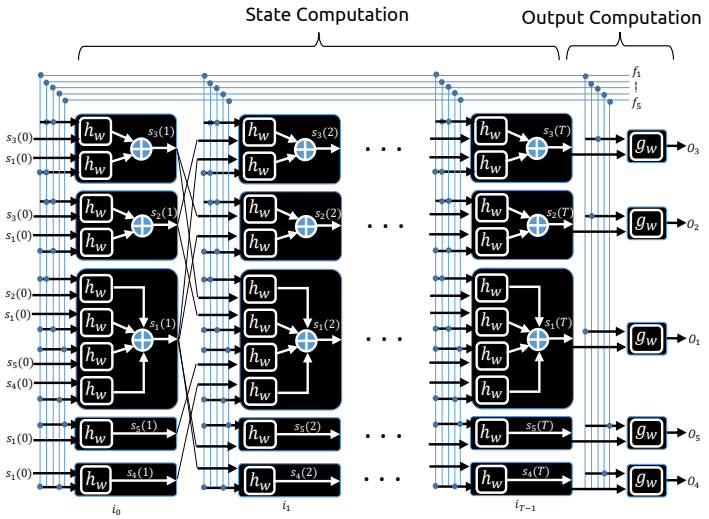

Fig. 6. SFC encoding network to iteratively determine the states of VNFCs

SFC Encoding Network: As explained in section IV-C, state computation entails an iterative approach using equation 9. This may be achieved by representing the model in Fig. 5 as an encoding network. In the encoding network, the model in Fig. 5 is unfolded into multiple layers, where all the g_w functions are placed in one layer, which is preceded by multiple layers each of them having all the h_w units. Each layer i corresponds to an iteration in which the state $s(i+1)$ is computed for each VNFC. The h_w units of any two consecutive layers are connected following VNF-FG connectivity. An example of such an unfolded network is shown in Fig. 6 for VNFC 2, for T iterations of state computation.

This process is summarized in algorithm 1. As can be seen, the process consists of three main steps: (1) observing the resource utilisation of the VNFC as well as that in its neighbourhood, (2) using the observed resource utilisation to determine the state of the VNFC, and (3) using results from the first two stages to determine the forecast resource utilisation.

V. Learning and Adaptation

In order to achieve forecasts that correctly approximate actual resource requirements, the two functions h_w and g_w must be trained. This involves using data that has both inputs f and target outputs ξ, to adapt the weights w of the FNNs to the task under consideration. In the case of the problem addressed in this paper, we need to have sample data, that shows for a given state (resource utilisation profile) $s_n(t)$, the resource utilisation $o_n(t+\tau)$ at a given time in the future. This learning task can be posed as the minimisation of a penalised quadratic cost function (11).

$$e_w = \sum_{n \in N} \left(\frac{1}{2} \Big(o_n - \xi_n \Big)^2 + \beta L\Big(o_n \Big) \right) \tag{11}$$

The first term in equation (11) is the standard error term usually used for training FNNs [14]. The second term is a penalty function which is added to the error function to ensure that the function h_w is a contraction map. The relative importance of the second term can be adjusted using the constant β. The second term, which has been adapted from the one used in [13], is meant to limit the values that can be assumed by the weights w to low values. This is achieved by using the function L (defined below) to penalise the FNN whenever its output is above a given threshold μ, known as the contraction constant. In this paper, since all inputs to the system are first scaled to the range $(0, 1)$, the constant μ and β are both set to 1.

$$L(y) = \begin{cases} (y-\mu)^2 & \text{if } y > \mu \\ 0 & 0 \end{cases}$$

The learning objective is to find the weights w for each h_w and g_w such that the cost function (11) is minimised. The learning algorithm used in this paper is based on gradient-descent, and involves four main steps:

1) At iteration $k = 0$, the weights w of h_w and g_w respectively are initialized randomly between -0.5 and $+0.5$.
2) At each iteration $k = k + 1$, state and output computation is done using equations (9), and (10) respectively,
3) Computing the gradient $\frac{\partial e_w}{\partial w}$ of cost function (11) with respect to the parameters w for all h_w and g_w,
4) Updating the weights w for all h_w and g_w using equation (12), where α is the learning rate.

$$w(k+1) = w(k) - \alpha \frac{\partial e_w}{\partial w} \tag{12}$$

Steps 1 and 4 are obvious, while step 2 has been discussed in sections IV-C and IV-D. Step 3 is realised by using backpropagation-through-time (BPTT) [15], [4]. BPTT involves carrying out the traditional back propagation [15] on the unfolded network (Fig. 6) to compute the gradient of the cost function for each h_w and g_w and summing all the gradients up. The learning and weight adaptation algorithm is summarised in algorithm 2.

Algorithm 1 GNN-based Model for NFV $G(N, E)$

Initialise: w, iteration $i = 0$, state $s(i) = 0\ \forall n \in N$

procedure OBSERVATION
    Observe f for all VNFC's and their neighbourhoods
end procedure

procedure STATE COMPUTATION
    while $(i < T)$ **do**
        Compute $s(i+1)$ using equation (9)
        $i = i + 1$
    end while
end procedure

procedure OUTPUT COMPUTATION
    Compute $o(i)$ using equation (10)
end procedure

Algorithm 2 Learning and Adaptation

procedure LEARNING AND ADAPTATION
    Initialise: w, $k = 0$
    while (stopping criterion not satisfied) **do**
        Compute state s and output o using algorithm 1
        $\frac{\partial e_w}{\partial w} \longleftarrow$ Back Propagation Through Time
        Update w using equation (12)
        $k = k + 1$
    end while
end procedure

VI. EVALUATIONS

A. Experimental Setup

The proposed system has been evaluated using the setup shown in Fig. 7. The deployment is comprised of 6 main components: Clearwater cloud IMS, Openstack, User Equipments (UEs), Monitoring, Domain Name System (DNS), and the algorithms being tested. Clearwater [16] is an open source IMS core, developed by Metaswitch Networks. It is composed of five core nodes named bono, sprout, homestead, homer, and ralf. Bono is a Session Initiation Protocol (SIP) edge proxy which provides a Web Real-Time Communications (WebRTC) interface to UEs. It is the anchor point for UEs to the Clearwater system. Sprout is a SIP registrar and authoritative routing proxy which handles UE authentication. It includes a memcached cluster storing client registration data. Homestead provides a web services interface to sprout for retrieving authentication credentials and user profile information. It runs as a cluster using cassandra as the store for mastered/cached data. Homer is a standard XML Document Management Server (XDMS) used to store multimedia telephony (MMTEL) service settings documents for each user of the system using cassandra as the data store. Ralf provides an HTTP API that both Bono and Sprout can use to report billable events that should be passed to the Charging Data Function (CDF). It uses a memcached to store and manage session state.

In our implementation, UEs are realised using SIPp [17]. SIPp is an open source test tool/traffic generator for the SIP protocol. Two SIPp instances were created each running in a VM. Each SIPp instance has 50,000 unique registered users.

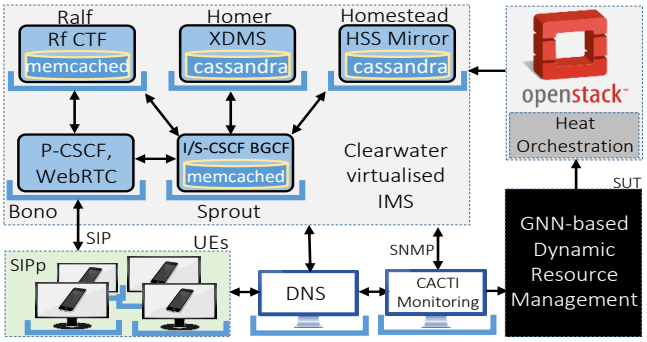

Fig. 7. NFV Implementation Used for Evaluations

Calls originate from users on one SIP instance to users on the other. In order to monitor the resource utilisation of the VNFCs in the system, we used Cacti [18] an open-source, web-based network monitoring and graphing tool which polls all system nodes using Simple Network Management Protocol (SNMP). Finally, we use BIND [19] an open source implementation of DNS to allow Clearwater nodes identify each other, and for load distribution when any of the nodes has more than one instance.

In the experiments, each of the Clearwater nodes represents a VNFC[3], and is hosted in a VM running in Openstack. Therefore, the basic evaluation system deployment included 10 VMs running in Openstack (5 for Clearwater nodes, 2 for UEs, 1 for DNS, 1 for Cacti monitoring, and 1 hosting the system under test (the proposed algorithms). These VMs, and additional ones (for horizontal scaling of Clearwater) were automatically deployed in Openstack using Heat Orchestration Templates (HOTs) [20].

B. Setup Parameters

Each VM used in the tests has 1vCPU, 2GB RAM and 8 GB Storage, each running Ubuntu 14.04. Calls were generated from one UE to the other following a Poisson distribution with an average arrival rate of 10 calls per second, and each call lasting an average of 180 seconds following a negative exponential distribution. To model a time of day effect on traffic arrivals, the above arrival pattern is repeated after every 50,000 calls, with the arrival rate and call duration being halved and doubled alternately. During the duration of each call, real voice and video media are transmitted between the UEs. The voice/video content is derived from VoIP (Skype) traces [21] which contain network traffic captured on the main link of Politecnico di Torino involving Skype traffic from students, researchers, professors and administration staff. The original 3.75 GB of end-to-end voice only and voice+video calls traces with about 40 million packets was split into 40 .pcap files, each with about 1 million packets. For each established call, one of these media files (chosen at random) was played to simulate real voice or video media. Three sets of experiments

[3]It is important to note that while our experimental setup involves multiple VNFCs that make up a single VNF, it does not limit our proposal to a single VNF. This is because irrespective of the number of VNFs or constituent VNFCs, for as long as a topology of the functions in a SFC can be created, then the GNN-based model can be used. This also includes situations where the VNF may be a blackbox, in which case it would considered as having a single VNFC.

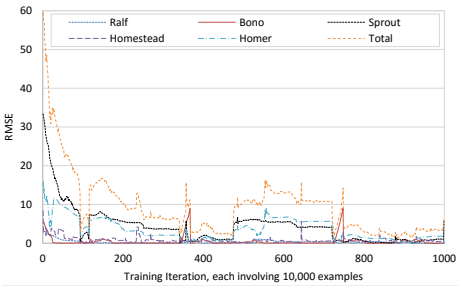

Fig. 8. Root Mean Square Error

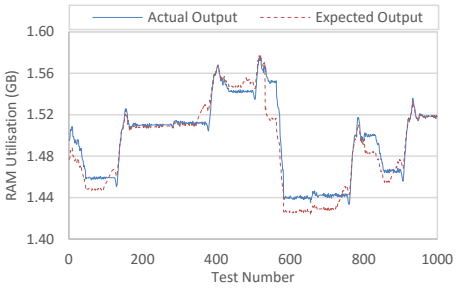

Fig. 9. Ralf RAM Utilisation

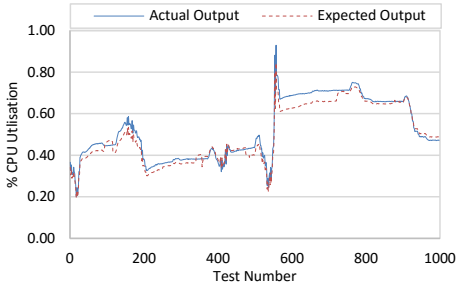

Fig. 10. Homer CPU Utilisation

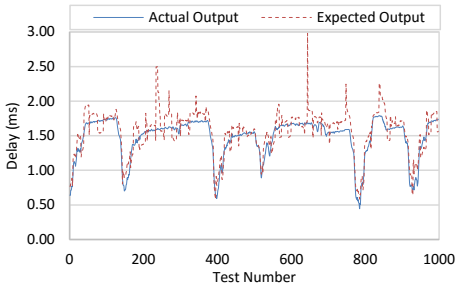

Fig. 11. Homestead Processing Delay

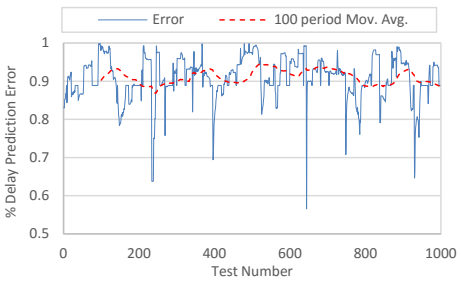

Fig. 12. Percentage Error on Delay Prediction

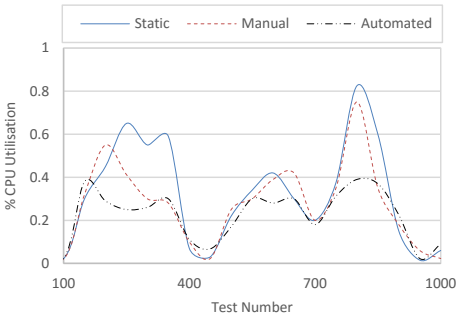

Fig. 13. Percentage CPU for Homer

were done. In each experiment, measurements of resource parameters (CPU, RAM, latency, Call drops) for all Clearwater nodes were taken every 15s. The first experiment was used to collect 10,000 data points which were used to train the FNNs. The history and forecasting periods used were $\pi = \tau = 20$, implying that for each VNFC, the last 20 observations were used to predict the resource requirement 20 time units in the future. In the second experiment, The trained system was tested to determine its prediction accuracy over 1,000 measurements, in which case, every 15s, the system was run to determine an output, and its output compared to the actual resource requirements 20 time units later. Finally, the system predictions were used to actually effect resource allocations in the Clearwater system. In this case, the system was programmed to effect a deployment of a new VNFC whenever it predicted that the % CPU utilisation of a given VNFC would exceed 40%, and where possible (if more than one are available), to reduce the number of deployed VNFC's when the predicted utilisation is 20%. The motivation behind using 40% and 20% respectively as the thresholds is VNF specific. In our monitoring of the normal operation of the Clearwater VNF, we observed that the VMs had a relatively low CPU utilisation most of the time, but that beyond 40% of CUP utilisation, performance (call drops) would degrade, while below 20% of CPU utilisation, the call drop rate remained almost unchanged. It is worth noting that these thresholds may be different for a different VNF. This was compared to a scenario where the resources were not changed at all (static), and another one where a manually programmed deployment was performed. The main difference between the manual programming and the proposal given in this paper is that in the manual approach, the process of scaling resources is only started after a given threshold is reached, while in our proposal, the reaching of this threshold is predicted ahead of time, and the scaling process started before the threshold is actually reached.

C. Results

The evaluation results are shown in Figs. 8 - 16. Fig. 8 shows results from the first set of experiments (training), while Figs. 9 - 12 evaluate the prediction accuracy of the trained system. Figs. 13 - 16 are based on 100 period moving averages. Results from evaluating the effect of the resulting system are shown in Figs. 13 - 16. From Fig. 8, it can be observed that the prediction root mean square error (computed using equation (11)) is initially high, and falls almost exponentially until it becomes stable after about 700 iterations (each with 10,000 training examples) of the learning and adaptation algorithm[4]. It is worth remarking that for our experimental setup, each iteration takes about of 45s to complete, giving a total training period (for 1,000 iterations) of about 13hours. However, since the learning/training phase is an offline process, it does not affect the online performance of the system. After the training period, the weights of all the FNNs in the model are saved in a file, from where they can be loaded every time a prediction is needed. Therefore, we observed that each online prediction required about 2s, including the time required to read the weights from a file. Moreover, in systems that are time critical, the prediction system could be kept running, in which case the time needed to load the weights from file can be saved. This way, our evaluations showed that a prediction can be obtained in about 5ms.

With a final RMSE of about 5 as shown in Fig. 8, and considering this is the total error for 10,000 examples and 5 VNFCs, it can be concluded that the system achieves an approximate accuracy of about 99% percent on the training data set. However, this level of accuracy is not realised when the system is tested on a new data set as shown in Figs. 9 - 11, which show the prediction accuracy of the RAM utilisation for

[4]It should be mentioned here that the stopping condition in algorithm II is 1000 iterations.

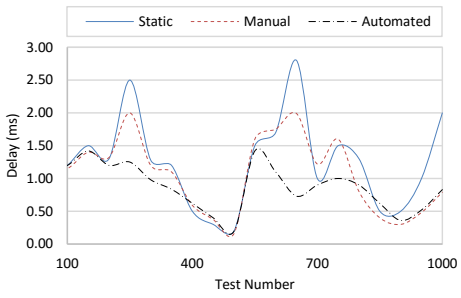

Fig. 14. Effect on Processing Latency

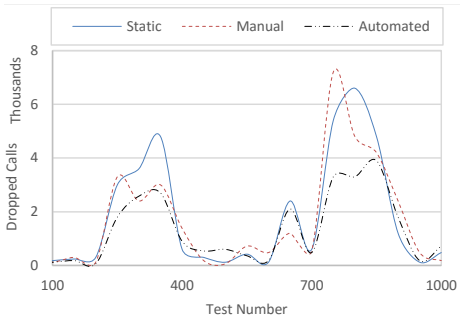

Fig. 15. Effect on Calls Dropped

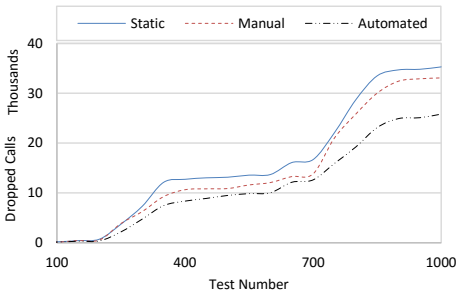

Fig. 16. Cumulative Call Drops

ralf, CPU utilisation for homer, and request processing latency for homestead respectively. First, it can be noted from Fig. 12 that the accuracy on latency prediction is about 90%. This loss in accuracy can be explained by the error term used during the weight learning phase which attempts to prevent the weights from assuming large values. By limiting the value of the weights, the FNNs may be prevented from generalisation with high accuracy i.e., the capability to use acquired knowledge on new tasks.

In Fig. 13, we show the evolution of % CPU utilisation for homer for the three scenarios described above. It can be observed that the proposed automated approach correctly forecasts the trend in resource utilisation, and deploys an additional homer VNFC, leading in a reduction in the load of the current VNFC as the load is now shared. When the utilisation reaches 40%, the manual scenario also triggers the deployment of an additional homer VNFC, which takes some time to start taking up load, but when it eventually does, the resource utilisation of the original VNFC reduces compared to the static scenario in which the number of VNFCs is not altered. Similar profiles can be seen in the other two cases when the utilisation crosses the 40% mark. It is also worth noting that there is not a very big difference in the performance of the manual and automated scenarios when the resources have to be scaled down (when utilisation is below 20%). This can be explained by the fact that since it does not require any preparation to shutdown resources (VMs in this case), predicting the need to shutdown in the use case under consideration does not give any advantage since in any case both scenarios have to wait until a certain point is reached before scaling down. However, both approaches would still perform better than the static approach in which resources would be left allocated, even when unutilised.

The performance results discussed above can still be observed in Figs. 14 - 16 in which the automated approach outperforms the other two approaches. In fact, it can be seen that the total number of calls dropped due to the system being overloaded over the entire testing period is 29% lower for the proposed approach compared to the manual one. Moreover, it is important to state that our prediction is mainly based on system load, and does not take into consideration the effect of traffic arrivals. It is possible that by attempting to predict traffic arrivals, and incorporating this into the model may yield better results. However, since we used synthetic traffic arrivals (because we could not get more practical data), trying to predict this could have been trivial. This could be an interesting future consideration.

VII. Related Work

Resource Management in NFV involves a number of sub-problems [3]. However, until now most current approaches have concentrated on the placement of servers [7] and VNFs [9], [22], [23]. These approaches do not consider the need to autonomously and dynamically scale the resources allocated to VNFs whose load may vary over time. As stated in the first NFV white paper [2] the automation and efficiency of such processes is of paramount importance to the success of NFV. This requires efficient and timely deployment and tear-down of resource containers on which VNFs run to match changing traffic.

With regard to dynamic resource management, three approaches are usually followed: control theory [24], [25], performance dynamics modeling [26] and workload prediction [27], [28]. However, such generic resource management approaches cannot be trivially applied to NFV environments due to the additional challenges that result from the need to simultaneously consider multiple resource types (such as CPU, memory, latency). Moreover, these resource types are not only segmented into many VNFCs and their connecting links, but the VNFCs may also require different quality of service guarantees.

Perhaps the closest approaches to the current work are in [29], [30] where the authors propose machine learning techniques for dynamic allocation of resources in network virtualisation environments. The authors model the nodes and links in a physical network as agents which use reinforcement learning to allocate resources to virtual nodes and links as requirements change. However, the nature of SFCs in NFV present additional challenges since the graphs that represent the VNFs are directed, which makes the VNFCs dependent on each other, and hence the GNN approach proposed in this paper more suitable in such a scenario.

In summary, our proposal enhances the state-of-the-art in that it complements VNF placement which is quite well studied, with a way to autonomously and dynamically scale up and down the initially allocated resources. This way, resources can be reserved for VNFs only when they are needed. Moreover, GNNs as used in our proposal are well suited for such a problem due to the ability to take advantage of topology dependencies which result when the load of VNFs is dependent on that of its neighbours. To the best of our knowledge, this is the first attempt to automate resource management in NFV through machine learning, and by taking advantage of the topology of the VNF-FG.

VIII. Conclusion

In this paper, we have proposed an automated, dynamic and topology-aware resource management approach for NFV environments. The proposal models each VNFC in a SFC as a pair of parametric functions which combine the observed resource utilisation profile at a given VNFC with that observed at its neighbours so as to predict future resource requirements. The predicted resource requirements can then be used to spin up new resources or plan global resource availability for the whole system. Through evaluations using a deployment of a virtualised IMS, and using real VoIP traces, we have evaluated our proposal, and showed that it can achieve a prediction accuracy of about 90%, and is able to enhance the processing delay and call drop rate by 27% and 29% respectively.

However, there might be some room to improve the current system so as to have even better generalisation accuracy by considering error functions with different penalty terms. Moreover, the backpropagation through time algorithm used for training the SFC encoding network requires to store the states of each parametric function. If the SFC is large, this might require a considerable amount of memory. Therefore, future work will attempt to find more efficient ways of training the SFC encoding network.

Acknowledgment

This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) and is co-funded under the European Regional Development Fund under Grant Number 13/RC/2077.

References

[1] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[2] R. Guerzoni, "Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for Action. Introductory white paper," in *SDN and OpenFlow World Congress*, June 2012.

[3] R. Mijumbi, J. Serrat, J. l. Gorricho, S. Latre, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, January 2016.

[4] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, Jan 2009.

[5] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV-SWA 001: Network Functions Virtualisation (NFV); Virtual Network Functions Architecture," http://www.etsi.org/deliver/etsi_gs/NFV-SWA/001_099/001/01.01.01_60/gs_nfv-swa001v010101p.pdf, December 2014.

[6] D. Ta, S. Zhou, W. Cai, X. Tang, and R. Ayani, "Network-aware server placement for highly interactive distributed virtual environments," in *Distributed Simulation and Real-Time Applications, 2008. DS-RT 2008. 12th IEEE/ACM International Symposium on*, Oct 2008, pp. 95–102.

[7] R. Mijumbi, J. Serrat, J. L. Gorricho, J. Rubio-Loyola, and S. Davy, "Server placement and assignment in virtualized radio access networks," in *Network and Service Management (CNSM), 2015 11th International Conference on*, Nov 2015, pp. 398–401.

[8] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On Orchestrating Virtual Network Functions in NFV," in *11th International Conference on Network and Service Management (CNSM) Mini-Conference*, 2015.

[9] H. Moens and F. D. Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, Nov 2014, pp. 418–423.

[10] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Transactions on Network and Service Management*, vol. PP, no. 99, pp. 1–1, 2016.

[11] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV 001 V1.1.1: Network Function Virtualization. Use Cases," www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf, October 2013.

[12] M. A. Khamsi and W. A. Kirk, *Banach Spaces: Introduction*. John Wiley & Sons, Inc., 2001, pp. 125–170. [Online]. Available: http://dx.doi.org/10.1002/9781118033074.ch6

[13] V. D. Massa, G. Monfardini, L. Sarti, F. Scarselli, M. Maggini, and M. Gori, "A comparison between recursive neural networks and graph neural networks," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 2006, pp. 778–785.

[14] R. Rojas, *Neural Networks: A Systematic Introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1996.

[15] J. C. Principe, J. M. Kuo, and B. de Vries, "Backpropagation through time with fixed memory size requirements," in *Neural Networks for Processing [1993] III. Proceedings of the 1993 IEEE-SP Workshop*, Sep 1993, pp. 207–215.

[16] Metaswitch Networks, "Project Clearwater," http://clearwater.readthedocs.io/en/latest/index.html, June 2016.

[17] Rob Day, "SIPp," http://sipp.sourceforge.net/, June 2016.

[18] The Cacti Group, Inc., "Cacti," http://www.cacti.net/, June 2016.

[19] Internet Systems Consortium, "BIND," https://www.isc.org/downloads/bind/, June 2016.

[20] OpenStack, "Heat Orchestration Templates," https://wiki.openstack.org/wiki/Heat, June 2016.

[21] TSTAT, "TCP STatistic and Analysis Tool: Skype Traces," http://tstat.polito.it/traces-skype.shtml, June 2016.

[22] R. Mijumbi, J. Serrat, J. L. Gorricho, and R. Boutaba, "A path generation approach to embedding of virtual networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, pp. 334–348, Sept 2015.

[23] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9.

[24] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, "Dynamic energy-aware capacity provisioning for cloud computing environments," in *Proceedings of the 9th International Conference on Autonomic Computing*, ser. ICAC '12. New York, NY, USA: ACM, 2012, pp. 145–154. [Online]. Available: http://doi.acm.org/10.1145/2371536.2371562

[25] W. Pan, D. Mu, H. Wu, and L. Yao, "Feedback control-based qos guarantees in web application servers," in *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, Sept 2008, pp. 328–334.

[26] W. S. Lai, M. E. Chiang, S. C. Lee, and T. S. Lee, "Game theoretic distributed dynamic resource allocation with interference avoidance in cognitive femtocell networks," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2013, pp. 3364–3369.

[27] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 14–28, Jan 2014.

[28] F. Jokhio, A. Ashraf, S. Lafond, I. Porres, and J. Lilius, "Prediction-based dynamic resource allocation for video transcoding in cloud computing," in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Feb 2013, pp. 254–261.

[29] R. Mijumbi, J. Serrat, and J. L. Gorricho, "Self-managed resources in network virtualisation environments," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 1099–1106.

[30] R. Mijumbi, J.-L. Gorricho, and J. Serrat, *Monitoring and Securing Virtualized Networks and Services: 8th IFIP WG 6.6 Proceedings of AIMS 2014, Brno, Czech Republic, June 30 – July 3, 2014.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, ch. Contributions to Efficient Resource Management in Virtual Networks, pp. 47–51.