IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 5 July 2023, accepted 30 July 2023, date of publication 4 August 2023, date of current version 14 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3302266

RESEARCH ARTICLE

DLVisor: Dynamic Learning Hypervisor for Software Defined Network

MOHAMED KHALAFALLA HASSAN[1,2], SHARIFAH HAFIZAH SYED ARIFFIN[2], (Senior Member, IEEE), SHARIFAH KAMILAH SYED-YUSOF[2], NURZAL EFFIYANA BINTI GHAZALI[2], MOHAMMED E. A. KANONA[1], KHALID S. MOHAMED[1], (Member, IEEE), MUTAZ H. H. KHAIRI[1], (Senior Member, IEEE), AND MOSAB HAMDAN[3], (Senior Member, IEEE)

[1]Faculty of Telecommunication and Space Technology, Future University, Khartoum 10553, Sudan
[2]Faculty of Electrical Engineering, University Technology Malaysia, Johor Bahru 81310, Malaysia
[3]Interdisciplinary Research Center for Intelligent Secure Systems, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

Corresponding author: Mohamed Khalafalla Hassan (memo1023@hotmail.com)

ABSTRACT Software Defined Network (SDN) is one of the modern networking technologies that provide network flexibility and simplifies network management. Virtual SDN (vSDN) enhances the flexibility of sharing physical networking resources by multiple slices representing multiple tenants or services where each tenant has control over their services or applications over the Virtual Network (VN). Network virtualization gives service providers more flexibility to offer new and innovative services with extra efficiency and reliability. Running multiple virtual networks over a given infrastructure creates challenges for efficient resource allocation mechanisms to avoid congestion and resource starvation and to maintain Service Level Agreement (SLA), where resource management in vSDN is carried out by hypervisors. Few studies have addressed dynamic resource allocation in the vSDN domain. Therefore, to efficiently utilize the resources of the virtualized networking infrastructure, network hypervisors must be proactive with self-reconfiguration capabilities to assign the physical resources and be highly adaptable and react to changing vSDN future demands. Thus, dynamic learning-based hypervisors are to improve hypervisor operations. Based on that, this study aims to enhance the vSDN technology to provide an enhanced proactive dynamic slice resource allocation mechanism, to improve traffic delivery and resource utilization. This can be fulfilled by proposing an enhanced intelligent forecasting model for vSDN slice resource utilization based on improved statistical and Machine Learning (ML) techniques. The proposed model will react dynamically to the concept drifts and then be utilized to develop a resource allocation mechanism for vSDN slice resource allocation. The improved dynamic forecasting resource allocation mechanism is verified through available real network traces datasets from various sources. The DLVisor with its Dynamic Learning Framework (DLF) can reduce overutilization and, consequently, resource starvation by 100% compared to the related benchmark.

INDEX TERMS Machine learning, resource allocation, resource forecast, software defined network, virtualization.

I. INTRODUCTION

Software-defined networking (SDN) has emerged as a promising networking technology that enables flexible data management in computer and communication networks whereby it separates the data forwarding plane and the control plane. On the other hand, network virtualization enables sharing of physical networking resources where tenants or slice owners have authority over their virtual network resources. Networks can exploit the benefits of SDN and Network Function Virtualization (NFV) through the virtualization of SDN networks. The SDN hypervisor separates the underlying physical SDN network into numerous logically separated vSDNs, each with its controller. For instance, each Virtual

The associate editor coordinating the review of this manuscript and approving it for publication was Mahdi Zareei.

TABLE 1. List of abbreviations.

Abbreviation	Full Name
5G	Fifth Generation
AD	Anderson darling
API	Application programming interface
CPU	Central processing unit
DLF	Dynamic learning framework
DM	Decision making
DB	Data base
FR	Flow rules
GRE	Generic routing encapsulation
GB	Giga bit
IIOT	Industrial internet of things
IoT	Internet of things
LDP	Links discovery protocol
LSTM	Long Short-Term Memory
MAC	Media access control
ML	Machine learning
MPLS	Multiprotocol label switching
NFV	Network function virtualization
NP-hard	Non-deterministic Polynomial-time hardness
NVP	Network virtualization platform
OF	Open Flow
OVS	Open virtual switch
QoS	Quality of Service
RAM	Random access memory
SDN	Software defined network
SLA	Service level agreement
SNMP	Simple network management protocol
SP	Service provider
VA	Virtual agent
VAO	Virtualization Agent Orchestrator
VLAN	Virtual local area network
VM	Virtual machine
VN	Virtual network
VNE	Virtual network embedding
vSDN	Virtual Software defined network
WDM	Wavelength division multiplexing

Machine (VM) and its guest operating system runs on a specific physical computing platform [1], [2], [3]. In addition to monitoring the virtual machines, the hypervisor assigns actual computing platform resources to each virtual machine. The hypervisor creates several vSDNs using the Open Flow (OF) protocol based on a particular physical network. Each vSDN corresponds to a slice of the entire network. Future networking technologies in Fifth Generation (5G) and beyond are to be enabled by vSDNs [4], [5], [6]. Running vSDN over a given infrastructure with efficient resource allocation mechanisms may improve the utilization of the networking hardware that meets tenant and service specifications and Service Level Agreement (SLA). Table 3 shows the list of symbols which will be used in the following section

In addition, vSDN enables network service providers to deliver new and innovative services with greater flexibility, efficiency, and dependability. Operating multiple virtual networks requires a significant amount of physical networking resources. Therefore, intelligent, and efficient resource allocation methods are essential. To persistently achieve and sustain the best performance, SDN network hypervisors must be designed to be adaptable, where it should adopt and enforce approaches for self-reconfiguration. To ensure better resource allocation and optimization, vSDN requires current and future network states to be forecasted and continuously run in online or semi-online mode to adapt to network demand variations. Such mechanisms and approaches must work in variable time scales to achieve high resource efficiency for the virtualized resources. Therefore, dynamic learning-based hypervisors are needed to improve the hypervisor operations. The design of hypervisor resource management algorithms in solving these challenges is an open research field and needs detailed investigation [1], [4].

To efficiently utilize the resources of the virtualized networking infrastructure, sophisticated forecasting-based resource allocation frameworks are needed for the physical resources. These frameworks must have some intelligence to cope with the dynamic Quality of Service (QoS) demand and be able to react autonomously to dynamic and self-organizing situations without affecting the SLA. Therefore, proactive approaches for managing bandwidth and network resources are highly needed [7], [8]. The proactive dynamic network resource allocation relies on the forecasting network demands and acts accordingly to enable a timely and dynamic response. Thus, the accuracy of predictive approaches was regarded as a vital factor in various applications of the predictive frameworks.

Accurate ML techniques are crucial and widely used in different applications, such as network traffic forecasts [9], [10], [11], [12], [13], [14], the Internet of Things (IoT) [15], and wireless communications [16]. The resource management in vSDN is performed by the SDN hypervisor. No proactive dynamic resource allocation mechanisms were provided in all related literature, the provided methodologies and frameworks were either static (they cannot adapt to traffic/network changes) or reactive due to working in current states with no resource forecasting which may lead to resource starvation. Therefore, it is essential to develop a dynamic learning framework to allocate and modify bandwidth slices for better resource utilization and to avoid resource starvation and SLA violations due to congestion and QoS degradation.

This study aims to enhance the vSDN technology to provide an improved dynamic slice resource allocation mechanism to improve resource utilization and minimize resource starvation. The specific objective of this work is to enhance the slice allocation mechanism in vSDN based on the proactive slice management based on resource (bandwidth forecast) in which will be reflected as a result in eliminating the count of overutilization and minimize congestion and resource starvation. This research contributes to vSDN technology by providing the hypervisors with the capabilities to manage and improve their performance autonomously since bandwidth slice management is one of the critical resources that need to work on short time scales to achieve high resource efficiency for the virtualized resources. Therefore,

this research will proactively provide ML learning-based hypervisors to avoid slice resource starvation and SLA violation in vSDN. This includes:

1) integrating an enhanced static forecasting model into vSDN slice management to improve vSDN slice utilization based on ML techniques.
2) adopting the static forecasting model using a dynamic learning framework (DLF) to reduce the static model forecasting error due to concept changes to update and improve model validity.
3) proposing proactive enhanced resource (bandwidth) slice allocation and supply and demand management in vSDN using the dynamic learning framework to minimize congestion and resource starvation. The proposed vSDN slice management framework will be named DLVisor.

The organization of the paper is as follows, section II provides a brief introduction of virtual network and virtualization, SDN and vSDN, section III discuss the state-of-the-art resource management in vSDN technology, section IV provides the overall methodology, algorithms, architecture and implementation of the dynamic bandwidth slice allocation, section VI discusses the results and findings while section VII provides the conclusion

II. BACKGROUND

Network Virtualization (NV) has been derived from the success of virtualization in the computing domain [1], [17]. It creates separate virtual networks (slices) through specific abstraction and isolation functional blocks [1]. In the networking domain, slicing concepts are already there. For example, optical fiber-based networking, Wavelength Division Multiplexing (WDM) [18] can create slices at the physical layer, while at the link layer, Virtual Local Area Network (VLAN) and Multiple Protocol Label Switching (MPLS) [7], [19] can be created.

On the contrary, network virtualization tends to establish slices of the entire network, i.e., to form virtual networks (slices) across all network protocol layers. At any moment, a given slice should have its resources (specific abstraction of the network topology, link bandwidths, switch computational resources and switch forwarding tables). Virtual network (slice) enables testing of new networking paradigms, regardless of the propriety and restrictions imposed by current internet structures and protocols [1]. Running multiple virtual networks involves consuming specific amounts of physical networking resources. Therefore, efficient, and sophisticated resource allocation mechanisms are highly needed [8], [20]. For instance, the interconnection between the virtual nodes, the virtual paths and VM Placement on the physical infrastructure, this is also known as Virtual Network Embedding (VNE) problem [1].

The VNE problem is Non-deterministic Polynomial-time hardness (NP-hard) and is still being extensively investigated. Generally, accepting and rejecting virtual network resource

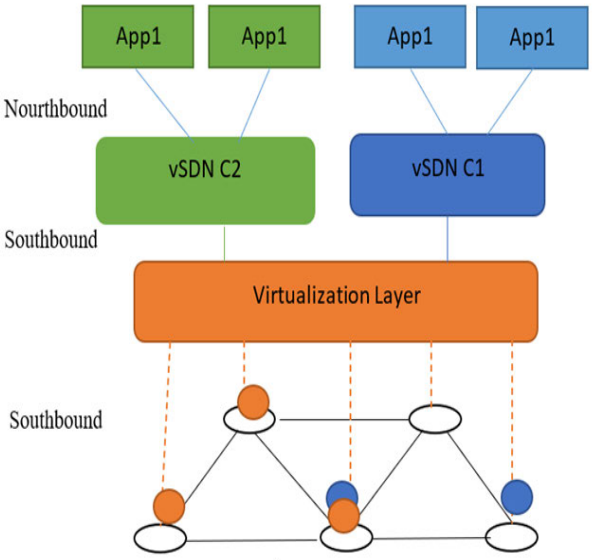


FIGURE 1. vSDN architecture.

allocation requests can be performed by admission control mechanisms. The current VNE optimization algorithms range from exact formulations, such as mixed-integer linear programs, to heuristic approaches based, it is possible to relate the resource assignment of vSDNs to the generic VNE problem and outline the use of the general VNE performance metrics in the SDN context. Several published overview surveys are already discussing network virtualization's principles, benefits, and approaches. For example, a detailed survey of network virtualized hypervisors for SDN can be found in [1].

The capability to program virtual networks using SDN is another important key aspect of total network virtualization [21]. Looking at legacy network virtualization, such as VLAN-based virtualization with no programming features, tenants will not be able to instruct switches to take actions such as traffic management, i.e., traffic steering. Nevertheless, to fully realize NFV, tenants must obtain virtual network resources, such as total views of network topologies and allocated networking resources, involving link data rates and network node resources. Moreover, providing isolated and programmable virtual networks has a significant advantage. In such a case, network operators can develop and test novel networking technologies with less imposed constraints [22]. In addition, NV is considered a key player in providing predictable (guaranteed) network performance [23]. Consequently, Service Providers (SPs) will be able to provide or provision new services over existing infrastructures in a much faster and more reliable way furthermore, SPs will allow their networks to dynamically alter and modify their virtual network according to the changing user and service demands [24], [25], [26]. The network virtualization layer allows hosting multiple controllers [26], [27]. The network hypervisor communicates with the underlying network hardware through the southbound interface via an SDN protocol, OF as an example. In the case of NV, the hypervisor operates

on top of the same southbound interface for virtual network operators and towards SDN network tenants. The hypervisor is interfaced with multiple southbound with several SDN controllers [27]. The SDN hypervisor operates as a proxy. It intercepts and translates the control messages between tenants and the physical SDN network. Fig 1 shows the vSDN architecture.

By combining SDN and NFV, tenants will have the advantage of flexibility in resource sharing through virtual network sharing in addition to having the programmability feature of SDN, while NFV provides the ability to program the virtual resources [25] whereby the combination is called vSDN [24], [26]. This can be seen in Fig 1 which shows the vSDN architecture.

III. RELATED WORK

This section discusses the related work in the context of the dynamic bandwidth (slice) management in vSDN. As discussed in previous sections, the hypervisors perform resource management tasks, and all vSDN hypervisors are considered extensions to the FlowVisor hypervisor.

FlowVisor (FV) is the first hypervisor for SDN networks, providing the feature of sharing SDN networking resources between multiple controllers. FV can ultimately run as stand-alone software on commodity hardware (server) [28]. FV is a general-purpose hypervisor and represents the foundations for other vSDN hypervisors. It offers node attributes isolation such as Central Processing Unit (CPU) and Flow table in addition to bandwidth as a link attribute isolation. FV mainly stressed approaches to isolate network traffic in experimental networks from traffic in production network whereby it provides flexible definitions of network slices. However, it added latency in OF messages and exhibits no admission control and no mechanisms for slice management and optimization.

MobileVisor in [29], applied the FlowVisor approach in mobile packet core networks; where FlowVisor functionality is integrated into the architectural structure of a virtual mobile packet network that is comprised of several underlying physical mobile networks such as 3G and 4G networks. It allowed Internet Service Providers (ISPs) to define policy and QoS-based service in addition to that mobile operator. Moreover, ISPs will be able to manage their charging policies more efficiently. No latency calculation is mentioned; however, since the hypervisor adopts FV, it lacks dynamic resource management.

In [30] authors introduced compositional hypervisor to provide a flexible platform that enables SDN network operators to choose various network applications developed for different SDN controllers. This will allow different applications written for specific controllers to run on other controllers written in another language; the compositional hypervisor establishes a composed policy that represents a prioritized list of rules per SDN switch provided by the corresponding SDN controller, and then the composite hypervisor forms a suitable composition configuration to process SDN rules, the study stressed on composition policy formation time which is added to the hypervisor OF latency, in addition, the priority list is static and do not allocate or manage resources dynamically.

Authors [21] proposed the Network Virtualization Platform (NVP), focusing on data center network resource abstraction managed by the cloud tenants for multi-tenant environments, which works as a controller to provide SDN tenants to operate their SDN controllers via Application Programming Interface (API) and control their slices in the data center. This is accomplished by forming a distributed controller cluster to scale tenants' load as required with virtualized switches to steer tenants' traffic to the corresponding virtual machines by focusing on software that resides inside the virtual switches on the host servers. NVPs generally establish logical data paths (tunnels) between the destination and the source Open Virtual Switch (OVS) where the logical path relates to the corresponding tenant slice using Generic Routing Encapsulation (GRE) tunnelling techniques.

OpenVirteX hypervisor in [31] was introduced with two primary contributions: topology and address virtualization. OpenVirteX extends FV by tackling the flow space problem. This is achieved by using of headers to distinguish vSDNs instead of providing the entire of header fields space to the vSDNs. In OpenVirteX, switches re-write virtually assigned Internet Protocol (IP) and Media Access Control (MAC) addresses utilized by the hosts of each vSDN (tenant). OpenVirtex did not add a valuable contribution to the dynamic resource management.

In [32], a Datapath centric hypervisor was introduced to address the redundancy issue in FV (Single point of failure) and enhance the virtualization layer performance through the implementation of virtualization function as switch extensions. It works by implementing Virtualization Agents (VA) inside switches in addition to the Virtualization Agent Orchestrator (VAO). The VAO's ultimate responsibility is the slice monitoring and configuration, example of this is adding or removing slices. On the other hand, VA is responsible for communicating with vSDN controllers in addition to resource abstraction for the VAO. If VAO fails, VA can continue to operate, avoiding a single point of failure in Flow visor architecture. Datacentric does not support bandwidth isolation but still can support QoS based on extensive evaluations. The VA case adds an overhead of 18 % compared to the reference case. Failover overhead latency is about 3ms. Although it can support QoS, a lack of bandwidth isolation will not allow dynamic bandwidth slice allocation.

In [33], CoVisor was introduced as an extension to a compositional hypervisor that facilitates the cooperation of heterogeneous controllers to work on the same type of traffic with more focus on improving SDN physical network performance, i.e. flow table space and topology abstraction in such a way to provide the necessary resources such as topology information or abstraction when needed, i.e. a load balancer does not necessarily require a detailed topology view

to decide to drop or forward a packet. In addition to that, it provided a form of security against fraud SDN controllers. CoVisor enhanced Compositional hypervisor latency overheads by two to three folds. However, the hypervisor did not provide dynamic resource allocation.

In [34], DFVisor (Distributed FlowVisor) was introduced to tackle the scalability issue of FV as a centralized SDN virtualization hypervisor. It addressed the possibility of extending SDN switches with hypervisor capabilities, producing enhanced OpenFlow switches which can be accomplished by extending SDN switches with a local tunnelling module and vSDN slicer. DFVisor utilizes GRE tunnelling for data plan slicing and encapsulating data flows which are beneficial in adopting QoS. DFvisor adopts two-level synchronized distributed databases, the first database (local) resides on switches, while the global database maintains slices of statistical information, network operation and scalability are improved. However, the proposed solution was intended for a cluster environment.

EnterpriseVisor in [35], is one of the most notable hypervisors regarding resource slice allocation. It introduces an extended software module to monitor and analyze slice utilization. In addition, linear programming was used to adjust the bandwidth slices dynamically, the proposed engine determines slice requesters and resource providers to meet service requirements. Furthermore, the EnterpriseVisor interacts with the Flow Visor, applying the slicing policy to configure the network. Accordingly, slice configuration can be adjusted to meet service requirements.

In [36], intent-based virtual network management platform based on SDN is proposed to automate the configuration and management of VN resources from the tenant side. The framework was based on OpenVirtex. The proposed framework simplifies resource definitions and management from the administrative side through high-level business requirement representations since VN resource management is a complicated and time-consuming process in addition to a lack of an available automated provisioning process. The intent layer helps the tenants to specify high-level requirements.

In [37] and [38], AutoVFlow was introduced as a distributed hypervisor to be used in wide-area networks where the underlying infrastructure span across a non-overlapping domain. The hypervisor is responsible for each domain and acts as a proxy that performs slice and abstraction mapping AutoVflow delegates administration from heavy load controllers to other less loaded controllers. Solid updates policy was maintained between the centralized and the distributed controllers since one slice can span multiple domains. Several identities were used, such as virtual MAC addresses, which can be different from one domain to another was found to be considerably high (around 5.85 ms). The control process is manual and does not consider dynamic load distribution.

In [39], ONVisor Hypervisor was presented as an SDN-NV platform to provide flexibility by adopting distributed hypervisor instances that allowed sharing of VN state. Moreover, it supports different SDN protocols such as Netconf and LISP. Finally, it was tested and evaluated in minimal scenarios. No dynamic resource allocation was discussed.

In [40], a management framework was introduced to provide bandwidth management through the definition of static thresholds for each slice based on slice prior priority; the framework was presented as an admission control mechanism. the proposed framework comprises a Decision Making (DM) Module, which decides how much bandwidth to allocate based on the request handler's request and the database module (DB) information. However, no detailed assessment was provided on latency and overheads. Moreover, the framework was static and was based only on current states.

In [41] DART framework can dynamically distribute the network bandwidth to different to utilize network resources efficiently. The framework adopts an admission control mechanism to distribute network bandwidth on demand. The scope was limited to the Industrial Internet of Things (IIOT) and only worked on the current state. In this framework, two modules were proposed, communication and publishing modules. The communication module is used to send and receive information, while the publishing module is used to coordinate between the controllers. The centralized component is responsible for advising the best possible bandwidth for each SDN controller. The admission control can be triggered by load, priority, and packet loss ratio to redistribute the network bandwidth. This paper used priorities for the traffic as a trigger based on the requested QoS level.

In [42], the PrioSDN resource manager (PrioSDN_RM) was presented as a resource management framework to provide admission control for virtualized SDN-based networks. The proposed mechanism applies limits on the resource utilization for the virtual slices. It adopts an approach to taking advantage of a bandwidth distribution mechanism to react dynamically to load changes. It relies on flow priority, not device priority. Moreover, the bandwidth threshold moves according to a predefined critical flow. the storage keeps tracking current bandwidth utilization, and the compute module computes the available bandwidth resources. Then, it allocates the necessary amount of bandwidth resources with the aid of the flow rules manager and threshold manager, which assigns the amount of bandwidth based on the priority of the flows, provided that the priority threshold can be moved based on flow predefined priority. The proposed methodology works are similar to our approach, but it works in current bandwidth consumption, which could lead to resource starvation.

In [43], the Libera hypervisor was introduced to address scalability and ease of tenants' capability to provide their services. The scalability issue was mainly solved by supporting VM migrations and architectural modifications in which Flow rules are reduced; therefore, bandwidth between controllers and virtual switches will be minimized. Libera is considered an extension to OpenVirtx. Resource scalability is carried out at the current time without considering future

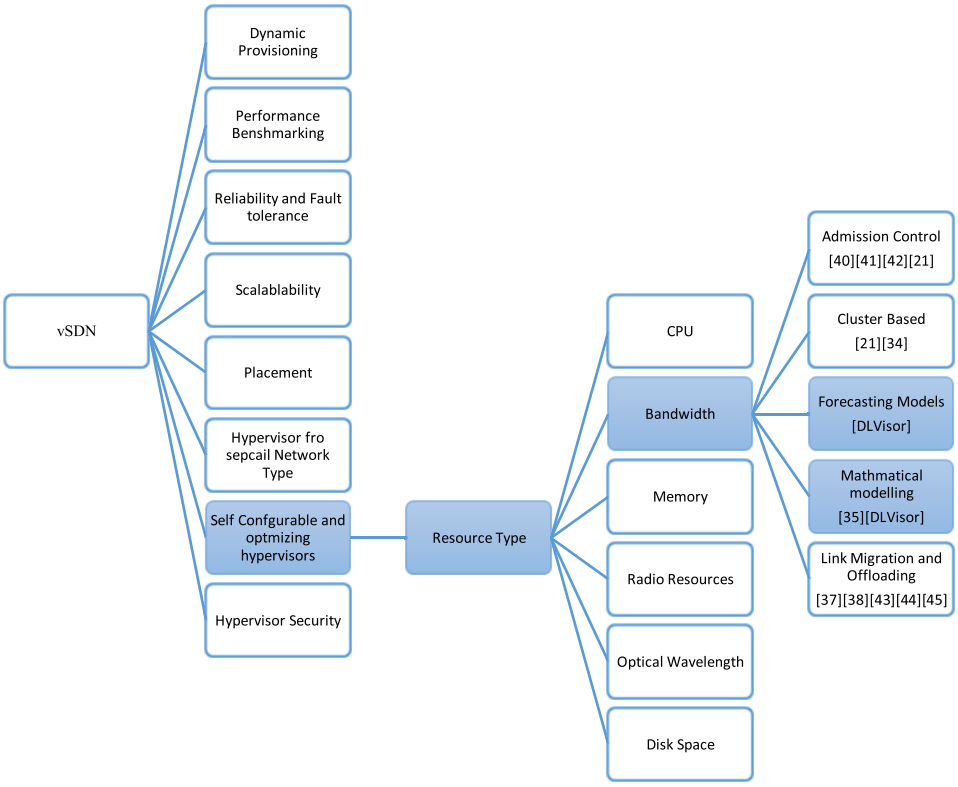


FIGURE 2. Scope of resource management in vSDN.

demands. Moreover, no detailed investigation was performed on the scalability based on VM migration.

In [44], TeaVisor was presented to guarantee bandwidth isolation in vSDN. The proposed hypervisor addressed the issue of overload links by using a greedy heuristic algorithm to split the traffic of overloaded links to multiple less-loaded paths. The results were promising and similar to what this paper is addressing. However, the algorithm is based on current traffic measurement, which may lead to a resource (bandwidth starvation).

In [45], the authors proposed a resource management architecture for multidomain SDN controller load scalability using a non-SDN-hypervisor. The framework is based on creating a new VM for the controller or migrating the SDN controller based on load elevation; Software agents were used in monitoring the controller loads. However, unlike SDN-based hypervisors, using a stand-alone hypervisor add issues of lack of resource isolation for SDN controllers. Moreover, creation and VM live migrations involve service downtimes. In all related literature, the provided methodologies and frameworks were either static (it cannot adapt to traffic/network changes) or reactive due to working in current states with no resource forecasting, which can lead to resource starvations.

As depicted in Fig 2, this paper will concentrate on bandwidth resource management in self-configurable and optimized vSDN hypervisors where the presented solution (DLVisor) will be the first to combine learned-based hypervisors through ML and mathematical-based resource management in vSDN.

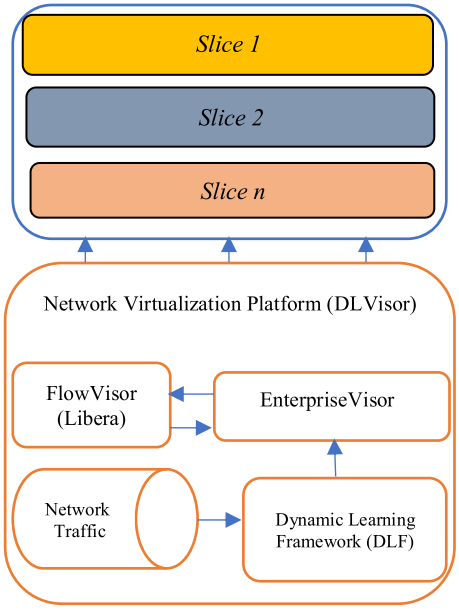


FIGURE 3. vSDN resource management components in DLVisor.

IV. DYNAMIC BANDWIDTH SLICE ALLOCATION FOR vSDN

This work is based on EnterpriseVisor [35] where it focuses on enhancing the existing hypervisors that should be able to perform regardless of the underlying topology and network demands. Therefore, hypervisors should exhibit capabilities to improve their performance autonomously and

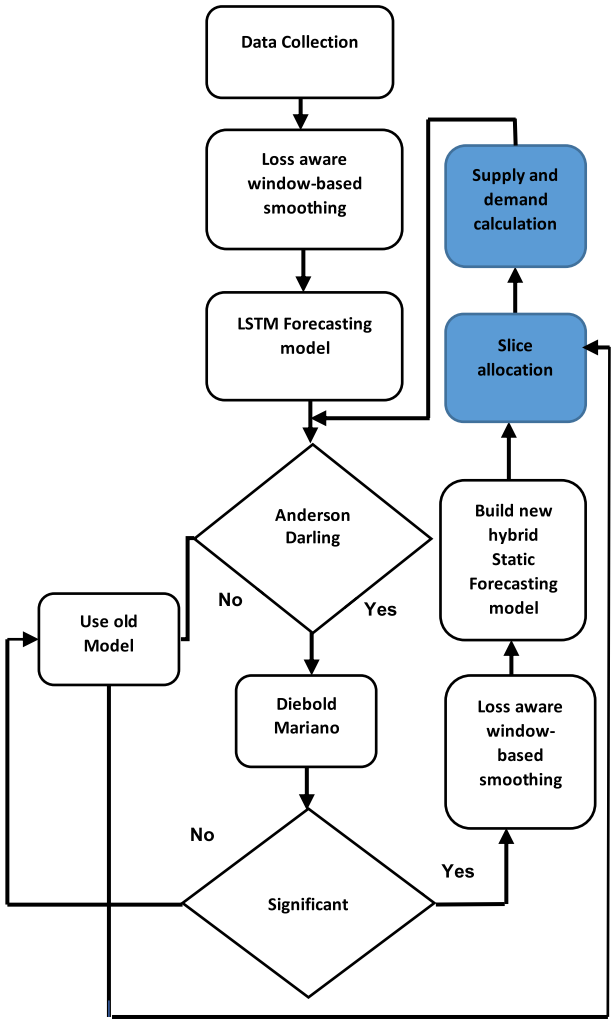


FIGURE 4. Flow chart for the DLVisor.

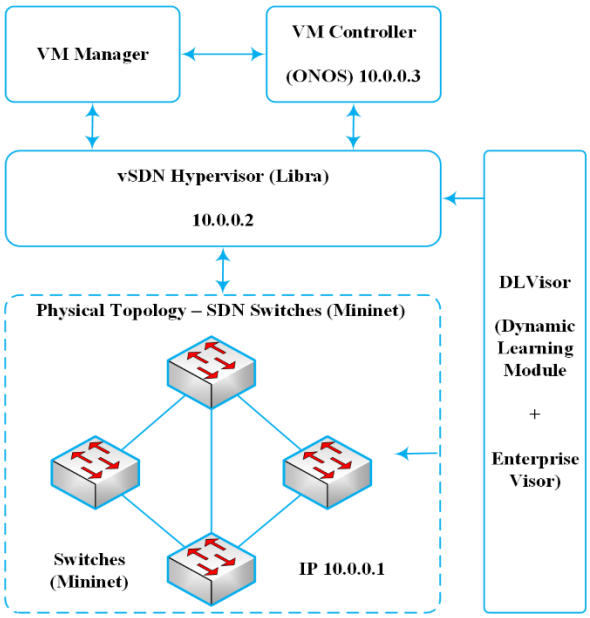


FIGURE 5. Testbed components.

TABLE 2. LSTM hyperparameters.

Parameter	*Name*
Library	*Tensorflow , Keras, NumPy, Sklearn*
Batch size	*1*
Epochs	*20*
Optimizer/Learning rate	*ADAM*
Loss function	*RMSE*
Neurons	*2*
Hidden layer	*1*
Activation function	*Sigmoid and Tanh*

TABLE 3. Dataset description.

No	Bandwidth Slice	Description
1	*LTE*	*Represents the aggregated backbone bandwidth traffic for LTE*
2	*ATN-OBY*	*LTE-enodeB traffic for a city*
3	*ATN-PSD*	*LTE-enodeB traffic for a city*
4	*MPLS*	*Represents the aggregated backbone traffic for corporate data centers*

transparently with minimum overheads. Bandwidth slice management is one of the critical resources that need to work in short time scales to achieve high resource efficiency for the virtualized resources. This can be achieved via cognitive and learning-based hypervisors and under varying mathematical modelling objectives and constraints. The proposed improved hypervisor shown in this work is named DLVisor (Dynamic learning hypervisor), whereby Fig 3 shows the proposed vSDN resource management components in DLVisor.

For this purpose and as shown in Fig 3, the Dynamic Learning Framework (DLF) introduced and discussed in our previous work [46] will be incorporated and integrated with the EnterpriseVisor resource management module in the EnterpriseVisor hypervisor. The DLF will capture the network bandwidth slice from the Cacti server using the Simple Network Management Protocol (SNMP). The live traces can be accessed directly from network storage in online, semi-online, or in batch mode. The dynamic learning approach will apply various hybrid Long Short-Term Memory (LSTM) windows-based smoothing algorithms with minimum data loss introduced earlier in [46]. Then, the best algorithm will be used to forecast the network traffic for each service slice or tenant. This work mainly focuses on vSDN virtualization where RAN end to end slicing is out of this paper 's scope and can be found in other related work such as in [47]. Three utilization levels will be defined as low utilization in the range of 0% to β, middle utilization is β to γ, and high utilization between γ and 100%. Fig 4 shows the overall flow chart for the DLVisor.

The white part indicates the DLF while the dark blue part indicates slice allocation and bandwidth management introduced by the EnterpriseVisor in line with the DLVisor components shown in Fig 3.

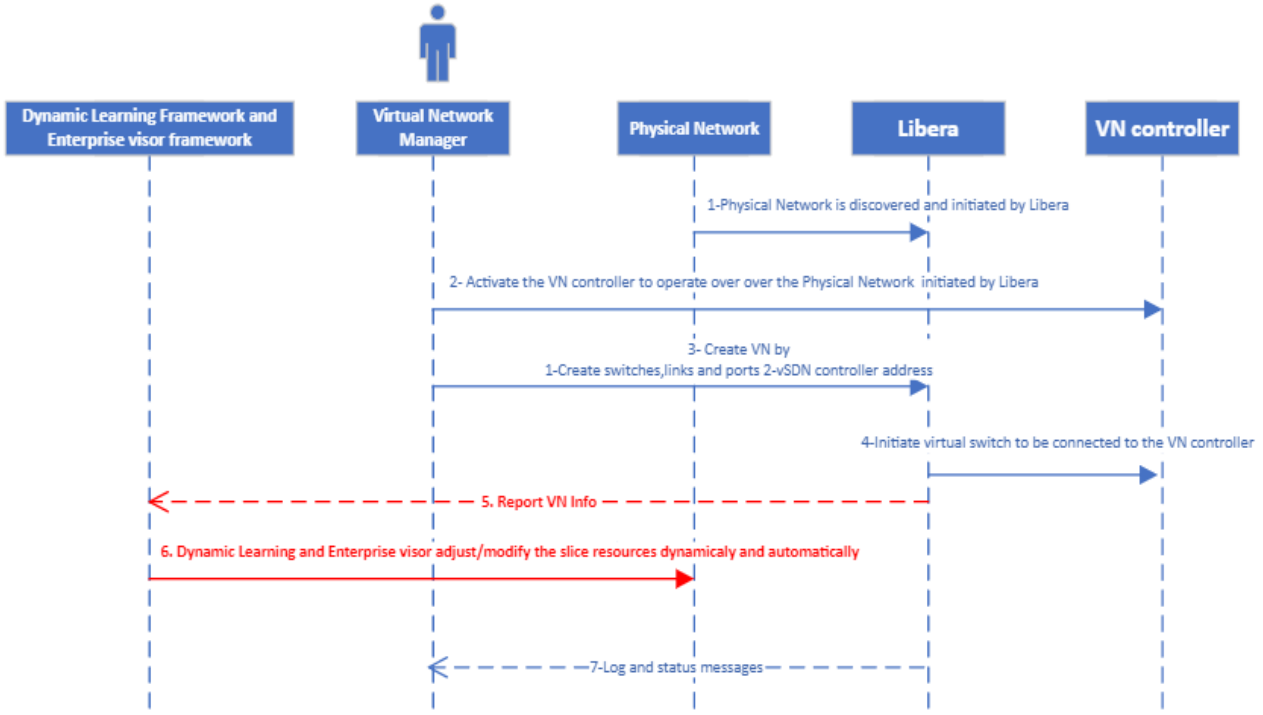


FIGURE 6. Sequence diagram for the testbed scenario.

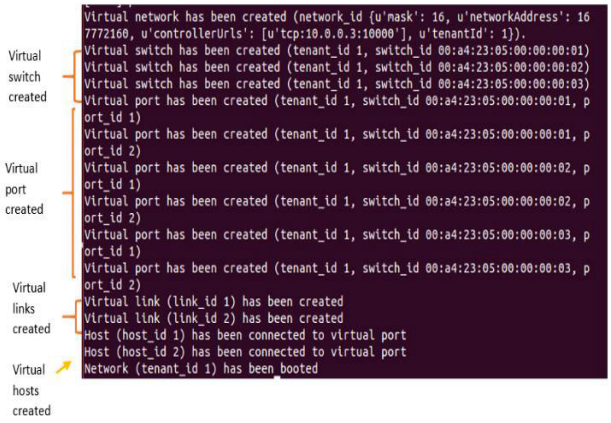


FIGURE 7. ATN-OBY slice creation.

As depicted in Fig 4, the flow starts with building an improved ML model by incorporating loss-aware window-based smoothing as a preprocessing technique to eliminate the unnecessary short/long-term noise components and avoid the erosion of periodic trends and patterns within the series noise and rapid traffic fluctuations, the output of this process is an improved hybrid LSTM-based ML that will be used for traffic forecasts. Then, to address ML model reliability and validity due to the rapid data characteristics and distribution changes resulting from the dynamic nature of the network properties, the forecasting frameworks must detect and adapt to all changes in the statistical properties of the network traffic.

FIGURE 8. Log messages during ATN-OBY slice creation.

FIGURE 9. The script used to modify the slice bandwidth limit by DLVisor.

Changes in traffic profiles, such as the sudden surge in traffic, occur due to changes or variations in the user's

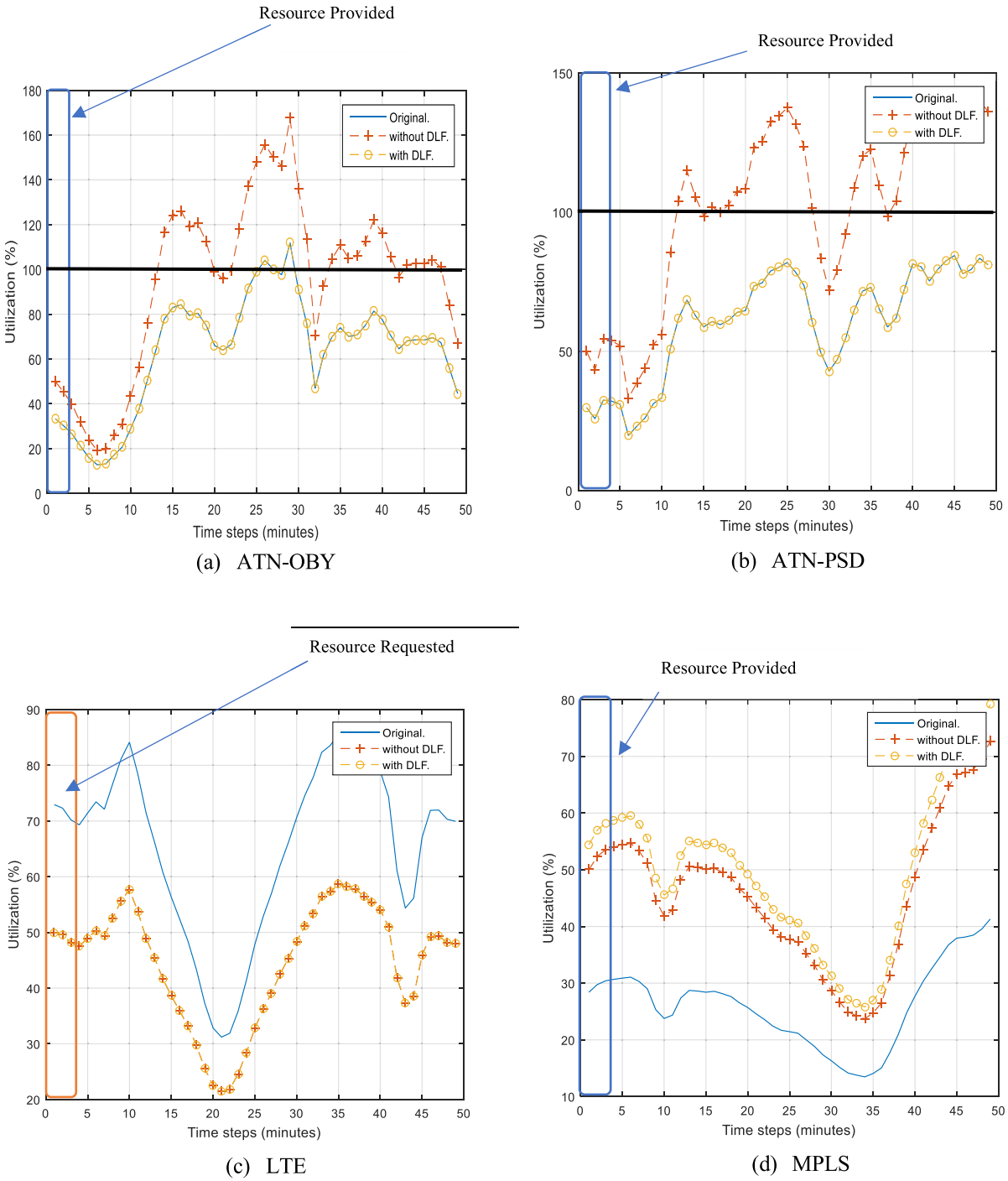
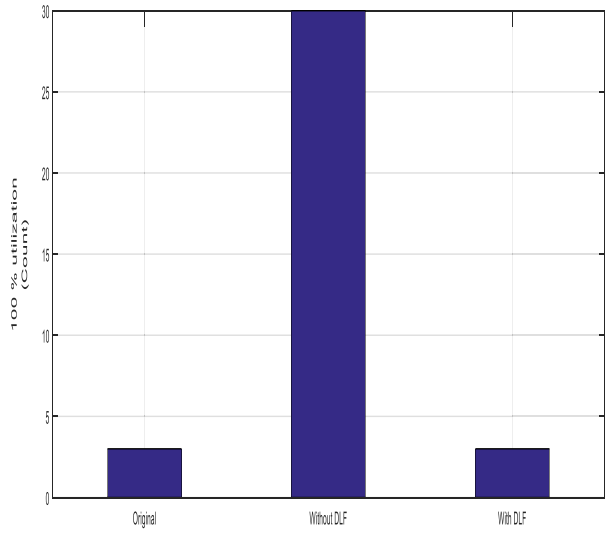
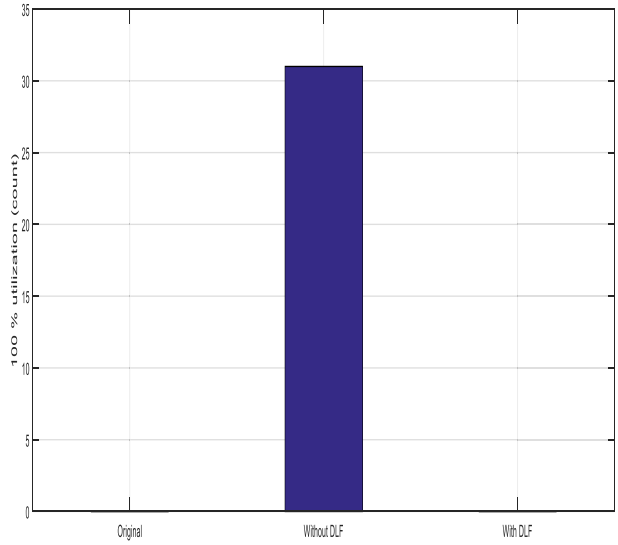


FIGURE 10. Slices utilizations in $W_d = 1$.

application behavioral demand. Therefore, a change detector using Anderson Darling (AD) is incorporated due to its sensitivity to detect changes in data characteristics, which may negatively affect the ML model accuracy. Then if a change is detected, a statistical significance test is used to validate the output of the forecasted data. Accordingly, if the output of the currently used ML algorithm is insignificant, the current (old) model is retained, otherwise, a new hybrid ML model is built provided the results are significant. The details of the process are already discussed in our previous work in [46]. Due to its reliability and simplicity, hyperparameter selection was conducted through grid search, as depicted in Table 2 [46]. Finally, the forecasted bandwidth is used as input to the slice allocation to identify the resource providers and resource requesters using slices classifications into low, medium, and high utilization slices, and then supply and



(a) ATN-OBY



(b) ATN-PSD

FIGURE 11. Counts of overutilization in $W_a = 1$.

demand calculations are performed to allocate the vSDN slices proactively to avoid overutilization to minimize and eliminate congestion and resource starvation.

A. DATASET

The dataset was collected from a premier Internet Service Provider (ISP) in which different bandwidth utilization time series were examined. The collected data represents the aggregated backbone traffic for Long Term Evolution (LTE), MPLS and enodeBs. Data was sampled by 50 and 350-time steps. Each time step represents 28.8 mins, where each 50-time step represents one day, and 350-time steps represent one week. This was attributed to the limitations of the data collection tool, while the values were interpolated and used for developing a time series model. Table 3 shows the data set names.

TABLE 4. List of symbols.

Symbol	Full Name
β	*Lower bound*
γ	*Upper bound*
G	*Graph*
v_{SDN}	*vSDN network nodes*
ε_{SDN}	*vSDN edges*
H_{SDN}	*SDN hypervisors*
r_{SDN}	*vSDN request*
R_{SDN}	*Total SDN request*
V^r	*Set of a virtual node of vSDN request*
c^r	*Virtual Controller node of vSDN request r*
t_j	*Time step j*
$\hat{y}$	*Forecasted bandwidth*
SL_i	*Slice i*
W_a	*Window a*
δ	*Statistically significant smoothed LSTM*
W_{tot}	*total number of slices*
t	*Time*
a	*Index*
h	*Number of resource providers*
g	*number of resource requester*
i	*index*
z	*Count number of 100% overutilization for* y_t
$\hat{R}\{..\}$	*Candidate Requester list*
$\hat{P}\{..\}$	*Candidte provider list*
$\widehat{p_m(i)}$	*Amount of forecasted provided resources*
X	*Count number of 100% overutilization for* $\hat{y}_t$
y_t	*Bandwidth utilization*
$\hat{y}_t$	*Forecasted bandwidth utilization*
$\widehat{r(i)}$	*Forecasted requested resources*
r_t	*Requested resources*
$\widehat{r_t}$	*Total requirement*
m	*index of slice provider*
C	*Cost function*
w_m	*Weight*
x_m	*Amount of resources*
S_i	*Minimum Bandwidth Gurantee for* i^{th} *Slice*
$\widehat{p_t}$	*Total resources from all resource provider*
$A_i(t)$	*Maximum Bandwidth allocation for* i^{th} *slice*
n	*Time steps*

B. SLICE MANAGEMENT IN vSDN

Table 4 shows the list of symbols which will be used in the following section

Algorithm 1 shows the Bandwidth Slice requesters and providers' allocation. The Algorithm 1's complexity is $O(nmh + m)$.

The vSDN network is modelled as a set of entities (nodes and edges) interconnected by a set of links. In this work, a network is modelled as graph $G(v_{SDN}, \varepsilon_{SDN})$ consisting of v_{SDN} network nodes (i.e., SDN switches) connected with ε_{SDN} edges. The SDN hypervisors are given by the set H_{SDN}, where H_{SDN} is a subset of v_{SDN}. The vSDN request, r_{SDN}, where $r_{SDN} \in R_{SDN}(R_{SDN}$ is set of total requests), is established between the SDN switches in V^r (Set of a virtual node of vSDN request) and controller c^r(Virtual Controller node of vSDN request r) at location donated by $c^r \in v_{SDN}$. The final objective is to map the controller c^r to the corresponding physical host switch, which is represented by

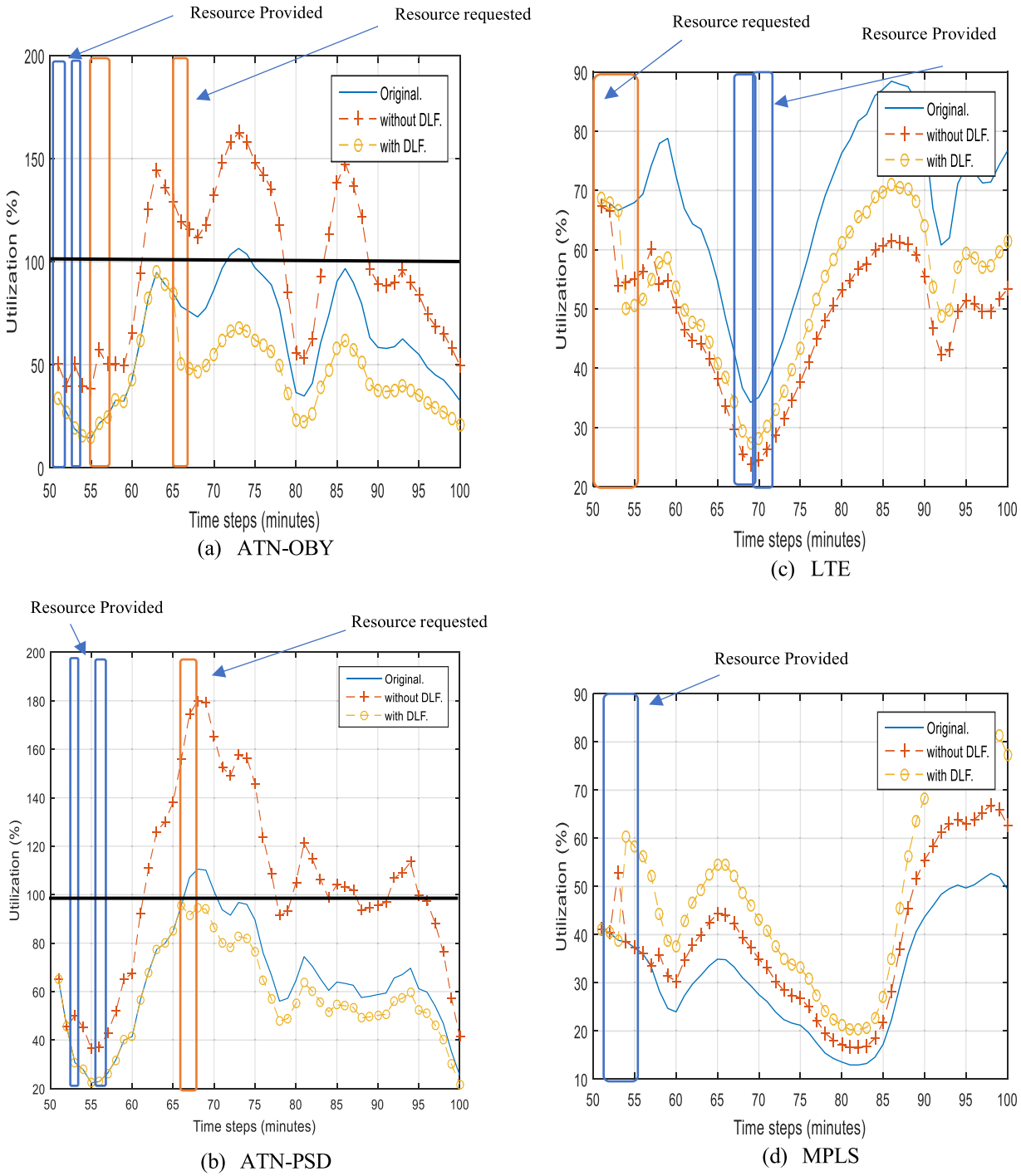
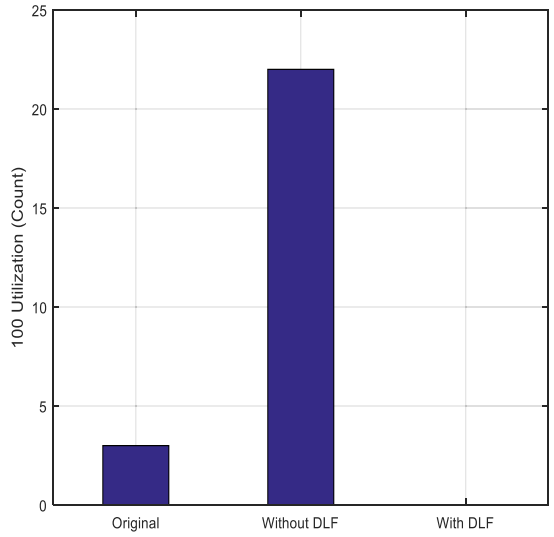


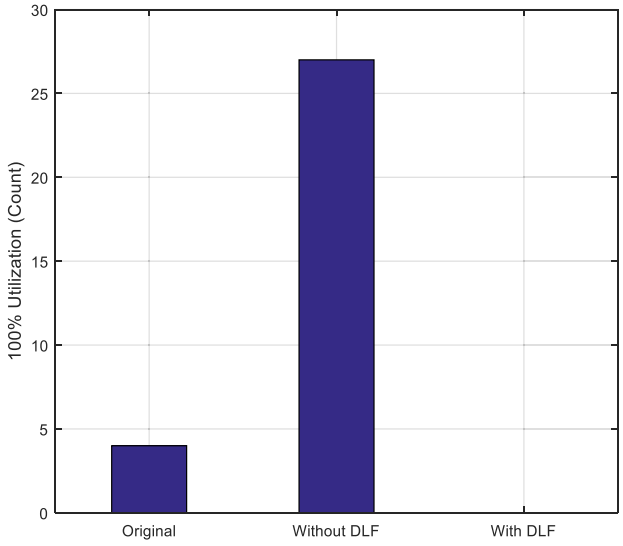
FIGURE 12. Slices utilizations in $W_a = 2$.

$\forall r_{SDN} \in R_{SDN}, \forall v^r \in V^r \cup c^r \rightarrow v_{SDN}$. Provisioning slices through the virtual and physical network is out of the scope of this paper. Algorithm 1 shows the resource requesters and provider classifications. The Algorithm searches in each time step $t_j \in \widehat{y}\ in\ SL_i \in W_a$, where $\widehat{y}$ is the forecasted bandwidth for the resources providers and resource requesters in slices SL_i running on a set of already provisioned v_{SDN} in window W_a.

The Algorithm starts with forecasting bandwidth $\widehat{y}_t$ as a time series of time steps t_i in slice SL_i belonging to window W_a. The slices are then allocated based on their calculated utilizations in u_i (lines 5 to 13 of Algorithm 1) to candidate requester. If the utilization is higher than the upper bound γ and to the candidate provider else, if the utilization is lower than the lower bound β, the list of the candidate providers and requesters are stored in the slices requesters list



(a) ATN-OBY



(b) ATN-PSD

FIGURE 13. Counts of overutilization in $W_a = 2$.

$\widehat{R}$ {..}and slices providers list $\widehat{P}${..}, and both lists $\widehat{R}${.. and $\widehat{P}${..} will be passed to Algorithm 2 to obtain the amount of resources that will provide $\widehat{p_m(i)}$ and the amount of requested resources $\widehat{r(i)}$. Then for all providers slices in the window W_a in the forecasted providers' list $\widehat{P}${..}, overutilization X is count number of 100% overutilization for the forecasted bandwidth $(\widehat{y_t})$, if $X \geq$ z, where z is the count number of 100% overutilization for all slices SL_i in window W_a using actual bandwidth y_t, $\widehat{p_m(i)}$ will be dropped from the providers' list $\widehat{P}${..} to avoid bandwidth starvation.

Algorithm 2 shows supply and demand calculations where Algorithm 2's complexity is $O(nmh)$. In line 4, the demand $\widehat{r(i)}$ is calculated from a set of slice requesters $\widehat{R}${..} which is limited to the values between the lower bound β and the upper bound γ, and most importantly, bounded by $Max(r_t)$.

Algorithm 1 Bandwidth Slice Requesters and Providers Allocation

Input: *SL_i: slice, $\widehat{y_t}$: forecasted bandwidth in slice SL_i, δ: statistically significant smoothed LSTM, Network infrastructure of $G(v_{SDN}, \varepsilon_{SDN})$, with r_{SDN} where $r_{SDN} \in R_{SDN}$ and connected to the controller c^r, given $\forall r_{SDN} \in R_{SDN}, \forall V^r \in v^r \cup c^r \rightarrow v_{SDN}$, i: index,j: index W_{tot}: total number of slices,t: time, a: index steps,β : lower Bound, γ : Higher bound , h: number of resource provider, g: number of resource requester*
i: index, z: Count number of 100% overutilization for y_t in $SL_i \epsilon W_{tot} \epsilon W_a$

Output:
$\widehat{R}${..} : Candidate Requester list ,$\widehat{P}${..} : Candidte provider list , X: Count number of 100% overutilization for $\widehat{y_t}$ in $SL_i \epsilon \widehat{p_m(i)} \epsilon W_a \epsilon \widehat{P}${..}

```
01: begin
02:R̂{..} :← ∅;
03:P̂{..} ← ∅;
04: for all time steps in forecasted bandwidth
slices t_j ε ŷ_t in SL_i ε W_tot ε W_a do  // using δ
05:  u_i ← Calculate slice utilization
06   while g =! 0
07     if u_i ≥ γ
08        R̂{..} ←: SL_i  //candidate of the requester
09     Else
10   while h =! 0
11     If u_i ≤ β
12        {P̂} ←: SL_i  //candidate of provider
13     Else
14 Calculate supply and demand using algorithm 2
15 for SL_i in p̂_m(i) in W_a ε P̂{..}
16     X ← Calculate slice over utilization
17     If X ≥ z
18-       Drop SL_i ε p̂_m(i) in W_a from P̂{..}
19 Loop:
```

In line 8, the number of supplied resources $\widehat{p_m(i)}$ from a set of supply slices $\widehat{P}..\{\}$ is calculated and bounded by constraints in Equation 2-5. In addition, the slice utilization after resource donation should be between the lower bound β and the upper bound γ. The amount of supplied resources requested from a requester and provided by m slice provider is calculated by the cost function, C, provided all constraints in Equations 2 to 5 are satisfied.

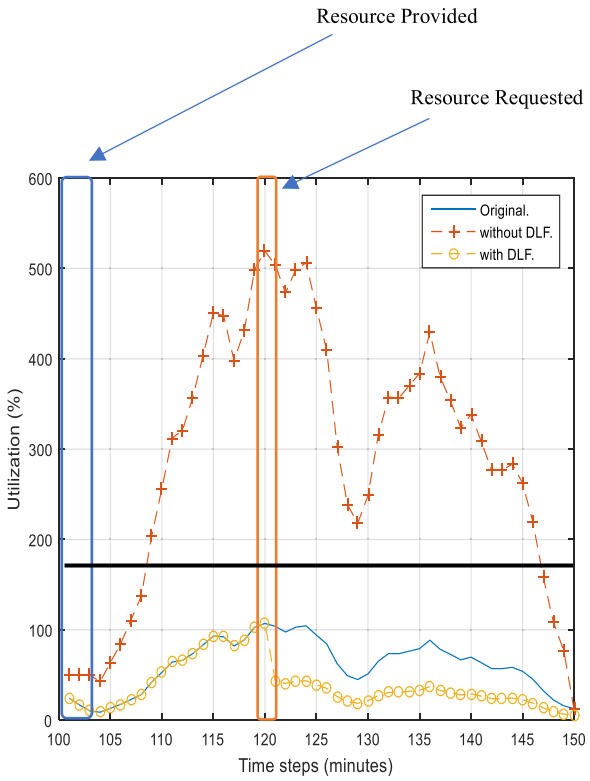
$$min(C) = \sum_{m=1}^{n} w_m x_m \tag{1}$$

Constraints:

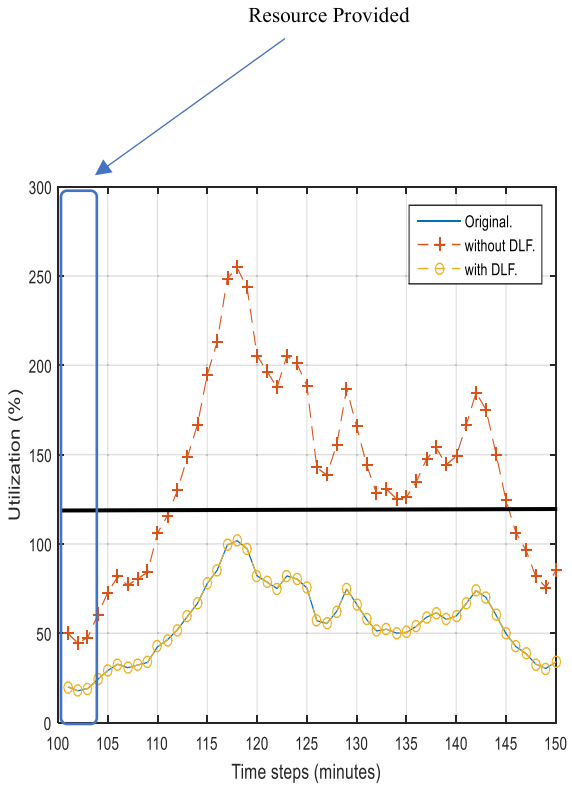
$$\sum_{m=1}^{n} x_m \leq p_t \tag{2}$$

$$\widehat{p_m(i)} \leq A_i(t) - S_i \tag{3}$$

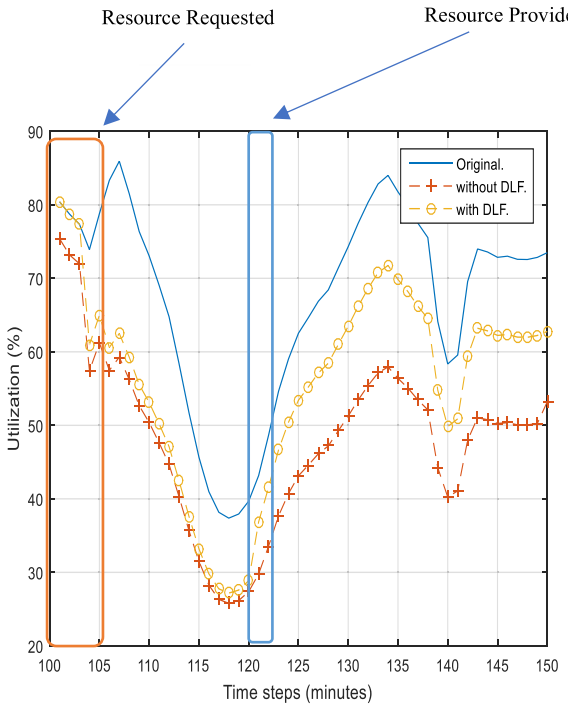
$$\sum_{i=1}^{n} min\left(\widehat{p_m(i)} \leq\right) \leq x_m \leq \sum_{i=1}^{n} Max\left(\widehat{p_m(i)}\right)$$
$$x_1, x_1, \ldots., x_n \geq 0 \tag{4}$$



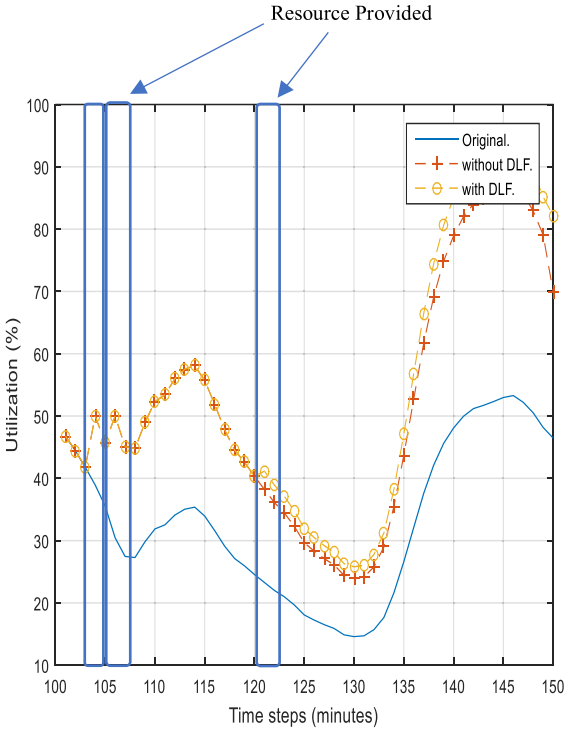
(a) ATN-OBY



(b) ATN-PSD

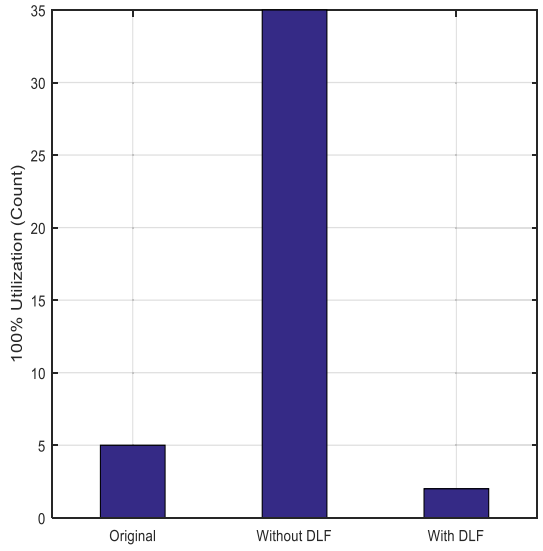


(c) LTE

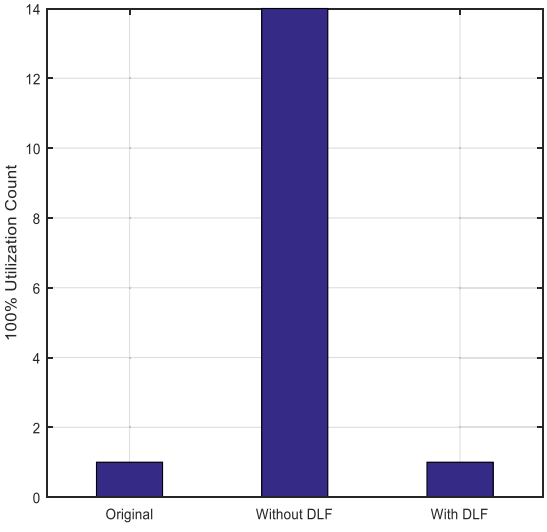


(d) MPLS

FIGURE 14. Slices utilizations in $W_d = 3$.



(a) ATN-OBY



(b) ATN-PSD

FIGURE 15. Counts of overutilization in $W_a = 3$.

$$min\,(r_t) \leq \sum_{m=1}^{n} x_m \leq Max(r_t) \qquad (5)$$

The cost function and constraints are adopted from the EnterpriseVisor framework [35]. The output from Algorithm 2 is the amount of provided resources $\widehat{p_m(i)}$ and the amount of requested resources $\widehat{r(i)}$ which is used by algorithm 2 for slice allocation.

This work is based on extending the resource management module embedded in the EnterpriseVisor framework. Furthermore, this paper adopts the same topology used in [35] Moreover, a test bed with Libera hypervisor is used emulation platform. Fig 5 shows the test bed components.

Fig 6 shows the sequence diagram for the testbed scenario, including the interaction between the DLVisor, vSDN network and Libera.

Algorithm 2 Supply and Demand Resource Calculation

Input: *$\widehat{R}\{..\}$: Candidate Requesters list, $P\{..\}$: Candidte provider list, SL_i: slice ,$\widehat{y}$: forecasted bandwidth in slice SL_i, i: index, z: Count number of 100% overutilization in using y_t in $SL_i \epsilon p_m(i)$, β : lower Bound, γ : Higher bound, A_i : Maximum Bandwidth allocation for i^{th} slice,$\widehat{r_t}$: total requirement, S_i : Minimum Bandwidth Gurantee for i^{th}Slice, $\widehat{p_t}$: Total resources from all resource provider*

Output: *C: Cost function, $\widehat{p_m(i)}$: the amount of Provided resources, $\widehat{r(i)}$: amount of resources Requested*

01: begin
02: for all time steps in forecasted bandwidth slices $t_j \epsilon \widehat{y_t}$ in $SL_i \epsilon W_a$ do
03 for SL_i in $\widehat{R}\{..\}$ do
04: solve $\widehat{r(i)}$ in $\beta \leq \frac{\widehat{y_t}}{A_i + \widehat{r(i)}} \leq \gamma$, Provided $\widehat{r_t} = \sum_{i=1}^{g} \widehat{r_i}$
05: If $R\{..\} \neq \emptyset$
06: Loop
07: Else
08: for SL_i in $\widehat{P}\{..\}$do
09: Solve $\widehat{p_m(i)}$ in $(\beta \leq \frac{\widehat{y_t}}{A_i - \widehat{p_m(i)}} \leq \gamma$, Where $\widehat{p_m(i)} \leq A_i - S_i$ and provided $\widehat{p_t} = \sum_{i=1}^{h} \widehat{p_m(i)}$
10: If $\widehat{p}\{..\} \neq \emptyset$
11: Loop
12: Else
13: Solve min C //Equation 1
14: Loop

The testbed is comprised of five functional entities represented by three VMs residing in a single host, the details of the virtual machines are shown in Table 5. The testbed platform was a computer with Core i7 2.1 GHZ CPU,32 GB RAM, 1GB network interface, 64-bit windows 10 operating system for host machines and with ubuntu Linux for the guest virtualized VMs. Table 5 shows the testbed VMs IPs

As depicted in Fig 6, the Virtual Network Manager (VNM) initially creates the physical network using the MININET-SDN simulator. Then, the physical network is discovered and initiated by the Libera hypervisor. The VN controller which is ONOS in our case is to be activated in ONOS VM using. As soon as the physical network is activated, Libera establishes multiple vSDNs networks while maintaining network isolation. It has been selected as an emulation platform due to its flexibility in creating VN and the ease of use of its programming features. Each tenant's VNM acts as the operator of the VN and submits various requests to Libera. It is worth to mention that, the proposed framework interacts horizontally with the traffic therefore it does not interfere vertically to the traffic flow/traffic path between the controllers and the vSwitches (Fig 5), accordingly it has the

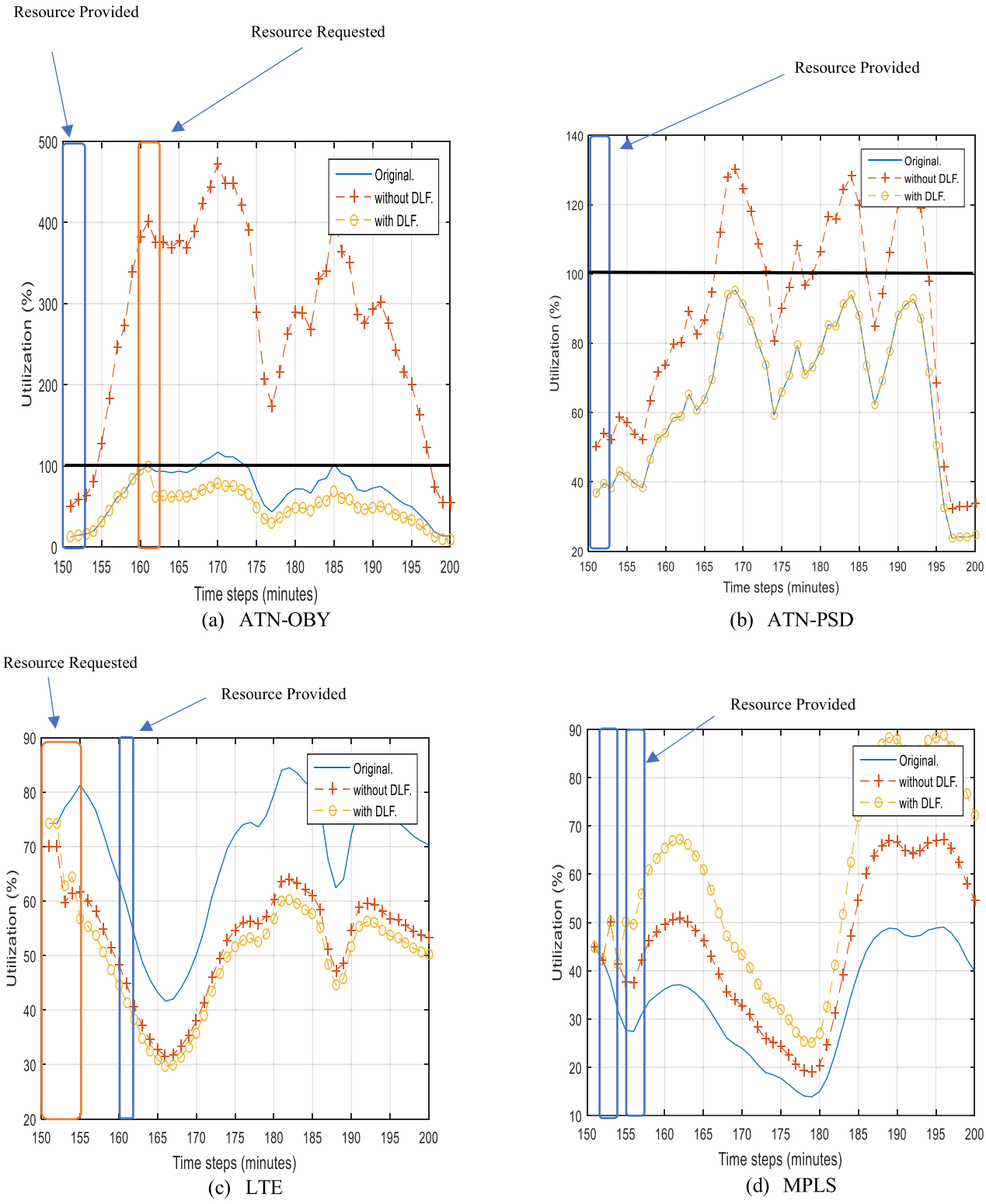
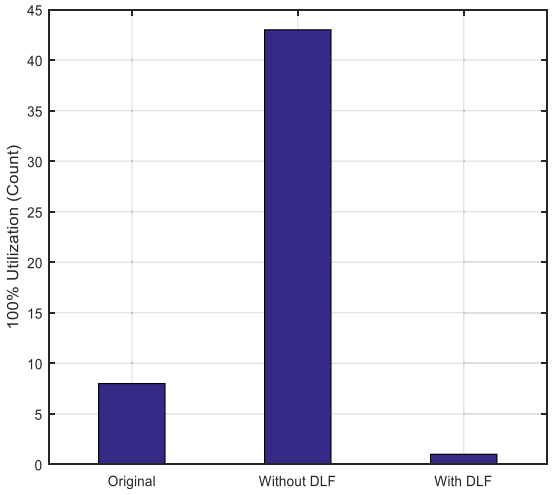


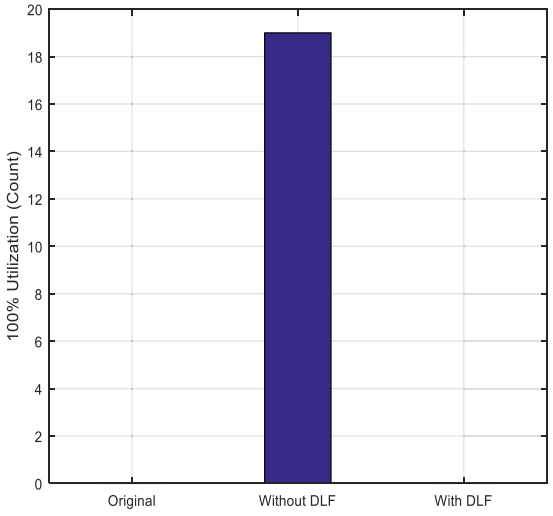
FIGURE 16. Slices utilizations in $W_a = 4$.

same FlowVisor Latency of 17ms when the OpenFlow's new flows processed and it increase port status response latency by roughly 0.71ms when the OpenFlow's port status is requested similar to the EnterpriseVisor latency [35].

Requests are classified into two categories: provisioning of topologies and topology modifications. First, the VNM starts creating a VN with a specified topology and VN entities. The term "virtual entities" refers to all entities that comprise the



(a) ATN-OBY



(b) ATN-PSD

FIGURE 17. Counts of overutilization in $W_a = 4$.

VN, such as vSwitches, ports, and links. For example, the VNM can specify whether a vSwitch is OpenFlow, white-box, or P4. Additionally, the VNM can construct several ports on each virtual switch. Similarly, a virtual link can be constructed by connecting two virtual ports.

In Fig 6 after the VNM establishes a VN and associated virtual Network Interfaces (NIs), the tenant operates the VN via the VN controller (VNC). The VNC can configure the vSwitch or virtual port by transmitting command messages to Libera through I2 (control channel). Libera offers a control channel for every virtual switch. Given that the vSwitch only belongs to one VN, each VN's control messages are routed separately to Libera. For Flow rules (FRs), the VNC can install desired FRs to any vSwitch at any time, thereby allowing packets to be dynamically forwarded or dropped. Additionally, the Libera collects statistical data from vSwitches and virtual ports. Accordingly, DLVisor can reconfigure or

TABLE 5. IP address of testbed.

No	VM name	IP address	Function
1	*Controller (ONOS)*	*10.0.0.3*	*VN Controller*
2	*Switches (MININET)*	*10.0.0.1*	*Physical Network*
3	*vSDN hypervisor (Libera)*	*10.0.0.2*	*Hypervisor*

modify slices as depicted in Fig 6, step 6. Figure 7 shows the output of slice one creation.

Four VNs were created in Libera. Fig 7 and 8 show the output of slice one creation. Fig 7 shows the creation of slice one, representing the ATN-OBY slice referred to as tenant _id 1; the slice is mapped to physical switches with corresponding switch IDs, ports, and links. In addition, the slice is also associated with its corresponding ONOS Controller at 10.0.0.3 through port 1000: TCP. Figure 8 shows more detailed log messages, including switch and link establishment. Figure 9 shows the script that is used to modify the slice bandwidth limit in step number 6 in Fig 6 and as a result of algorithms 1 and 2. Fig 8 shows the detailed log messages during ATN-OBY slice creation shows the script used to control the traffic queuing discipline (by default, First in First Out FIFO). It is based on a traffic control command in Linux that allows configuring packet schedular in support of ***qdisc***. On the other hand, the ***tbf*** argument indicates that the token bucket filter mechanism controls the traffic flow. It is stated here that the rate is limited to 90Mbps. The IPERF tool was used as a load generator to stress the VN slice bandwidth

V. RESULT AND DISCUSSIONS

In this work, the proposed resource allocation algorithm can only serve one resource requester at a time $\max(g) = 1$, and the maximum number of resource providers is four $\max(h) = 4$. Therefore, priorities were assigned to slice requesters. The priorities were assigned to LTE, ATN-OBY, ATN-PSD and MPLS respectively. In order to create a resource constraint in network bandwidth, the maximum amount of total requested resources and provided were selected to be equal to $(r_t) = (p_t)$. The maximum bandwidth limitation for the slices is 90 Mbps for ATN-OBY and ATN-PSD, 1.43 Gbps for MPLS slice and 1.5 Gbps for LTE slice. The maximum network capacity M is 3.2 Gbps and the target utilization rate, lower utilization bound, and upper utilization bound were selected to be ideal $(u_x) = 50\%$, $\beta = 40\%$ and $\gamma = 60\%$ respectively.

These values were determined empirically based on collected data set slices to create a situation where resource providers can experience resource starvation while providing their excess resources at the current time (t_j) and up to (t_{j+n}) where n is the number of forecasted time steps.

Fig 10 shows the utilization of ATN-OBY, ATN-PSD, LTE, and MPLS slices respectively, with and without a resource allocation using the dynamic learning framework (DLF) for all time steps in window 1 provided $t_j \epsilon \widehat{y}_t$ *in* $SL_i \epsilon W_a (a = 1$ and $j = 1$ to 50, i.e., the first 50 time steps); where the blue line in the graph shows the actual (original) slice utilization

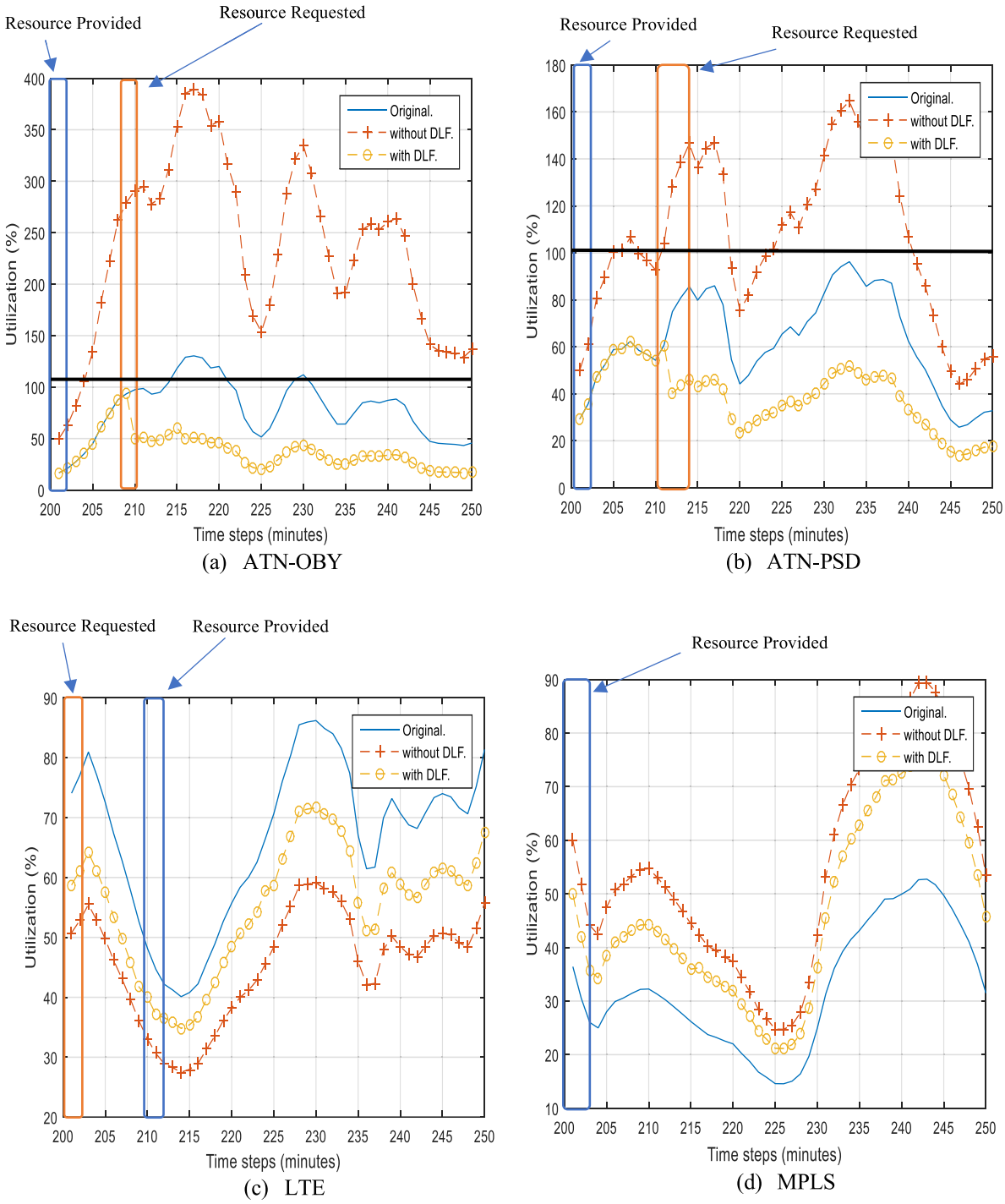
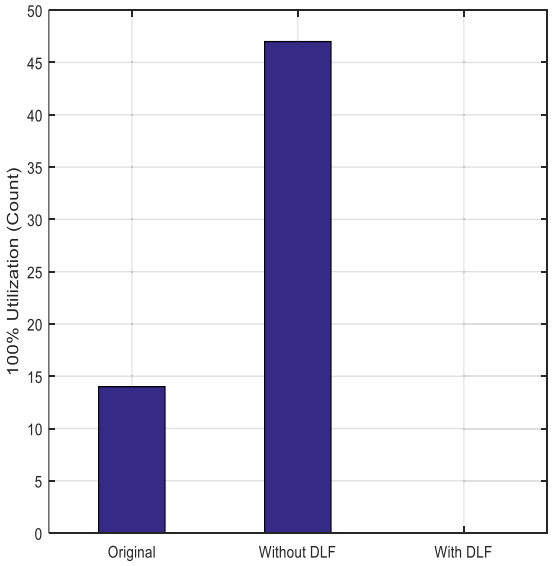


FIGURE 18. Slices utilizations in $W_d = 5$.

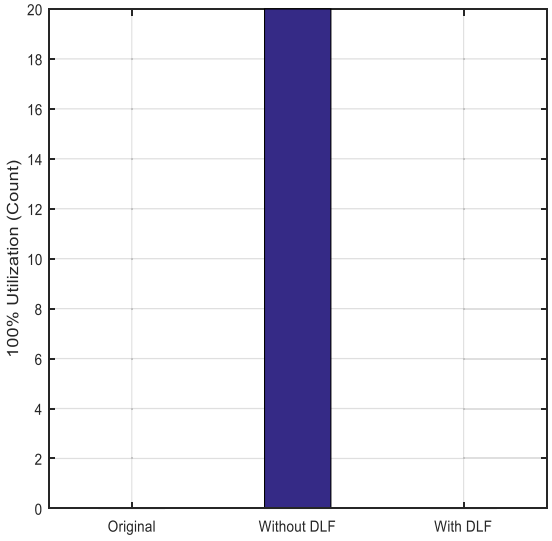
without using the resource allocation algorithm that was used as benchmark in this research. In window one and based on slice utilization, ATN-OBY slice, ATN-PSD slice, and MPLS slice are considered as resource providers due to its low utilization $u_i \leq \beta$, whereas LTE slice is considered a resource requester $u_i \geq \gamma$.

In time step 1 and According to algorithms 1 and 2, ATN-OBY, ATN-PSD, and MPLS slices will provide resources to other requested slices like LTE, since LTE slice utilization is more than the upper bound $u_i \geq \gamma$, where $\gamma = 60\%$ and other slice utilizations are less than the lower bound $u_i \leq \beta$ where $\beta = 40\%$ at target utilization $(u_x) = 50\%$; provided that all bounds and constraints are satisfied in Equations 1 to 5. The orange line depicts the newly utilization without DLF.

Fig 10a shows that slice utilization without DLF exceeds the full utilization of 100%, which causes the slice to starve



(a) ATN-OBY



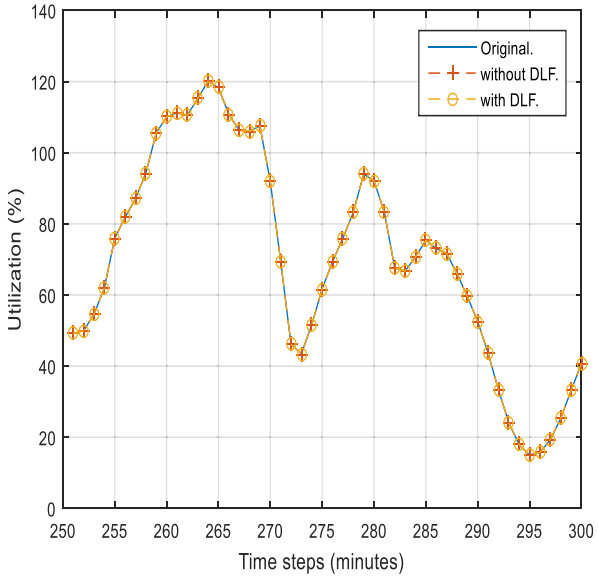
(b) ATN-PSD

FIGURE 19. Counts of overutilization in $W_a = 5$.

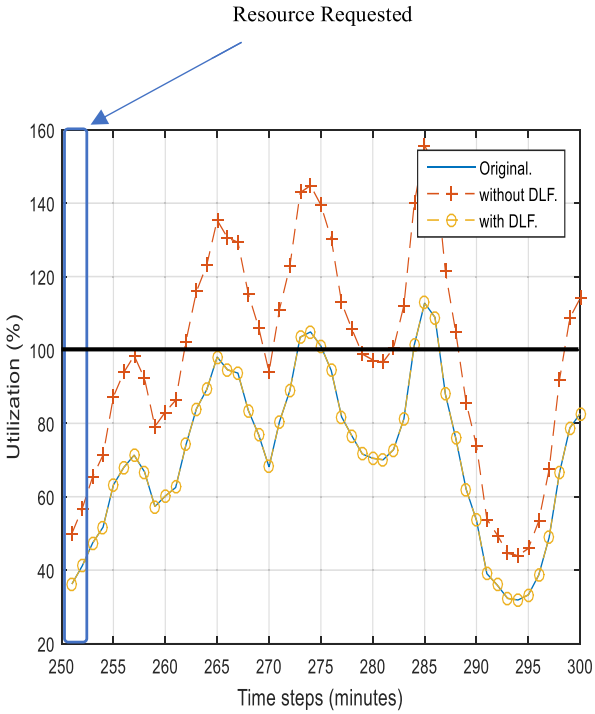
for resources due to the inability to recover the provided (donated) resources because of other slices utilizations. This is mainly due to applying the resource allocation algorithm in the benchmark (EnterpriseVisor) in real-time without considering the future demands. From the other side, in our proposed algorithm's supply and demand resource allocation, the calculations are based on future forecasted resource consumption (highlighted in yellow line). Therefore, there is prior knowledge of whether there will be resource starvation or not before deciding not to consider (drop) the resource provider (doner) from being considered as a candidate for a resource provider $\widehat{P}\{..\}$. The yellow line shows that the slice utilization is kept the same as the original slice utilization since the slice is not considered a slice provider after using our proposed DLF. The same analysis is applied to the ATN-PSD slice in Fig 10b. Meanwhile, in the MPLS slice, the utilization in resource allocation with DLF is increased due to extra resources provided to the requester to compensate the lost resources from dropped ATN-OBY and ATN-PSD slices. Fig 11 shows the count of times that the slice utilization exceeds the 100% utilization for ATN-OBY (a) and ANT-PSD in (b).

In Fig 11a, the ATN-OBY slice is overutilized 27 times more than when using the original (without resource allocation) and when using our DLF, confirming the proposed algorithm's effectiveness. However, in the ATN-PSD slice, the resource allocation without DLF introduces overutilization 30 times more than DLF. Fig 12 shows the slice utilization for window 2 for all $t_j \epsilon \widehat{y}_t$ *in* $SL_i \epsilon W_a$ ($a = 2$ and $j =$ 51 to 100). Likewise, the graph shows the actual (original) slice utilization without using the resource allocation algorithm and without DLF.

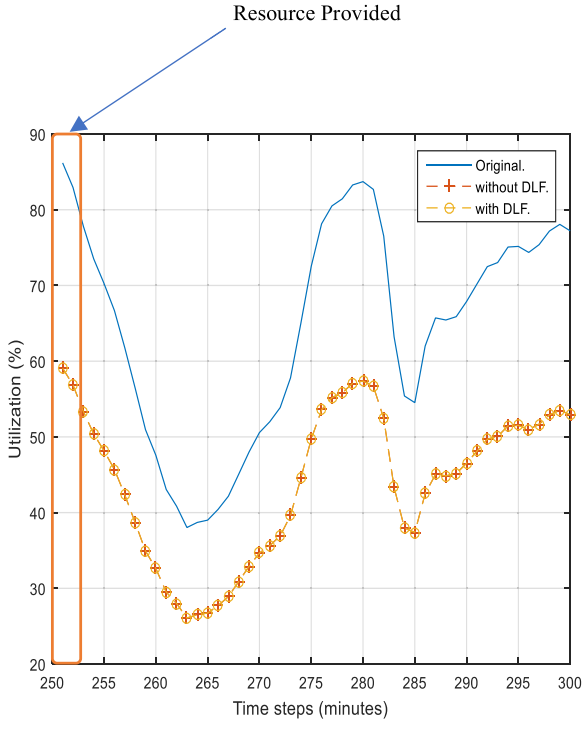
In window 2, ATN-OBY at time steps 51 and 53 was a resource provider when allocation without DLF was used. This is due to its low utilization, $\beta < 40\%$ (blue box). Nevertheless, at step 56, the slice requests a resource as the utilization is higher than the upper bound, $\gamma > 60\%$. In addition, the ATN-OBY slice is a resource requester at time step 66 using resource allocation with DLF. On the other hand, the ATN-PSD slice at time steps 53 and 56 is respectively a resource provider and resource requester when using allocation without DLF (blue box) and with DLF (orange box) respectively. In both ATN-OBY and ATN-PD, the orange line shows that slice utilization exceeds the full utilization of 100%, which causes the slice to starve for resources due to the inability to recover the provided (donated) resources that are in use. This is mainly due to applying the resource allocation algorithm in the benchmark in real-time without considering the future demands. The yellow line indicates slice utilization using our proposed DLF. Unlike the resource allocation without using DLF in yellow, the resource allocation using DLF does not cause overutilization (Bandwidth utilization). This is mainly due to the slices being dropped from resource providers' candidate list $\widehat{P}\{..\}$ since demand and supply calculation for the next 50-time steps reveals resource starvation when using resource allocation without DLF (the orange graph). On the other hand, the LTE slice is a resource requester and the requested resources in allocation without DLF are provided by the ATN-OBY slice at time step 51 and from all other slices at time step 53. This is obvious in the sharp drop in the consumed resources at time step 53. Meanwhile, a higher amount of resources were provided by the MPLS slice, which is reflected in a higher rise in consumed resources when using the allocation with DLF. The sharp increase in result was to compensate for the number of resources that were initially provided by the dropped slice (ATN-OBY slice and ATN-PSD slice). Fig 13a and Fig 13b show the count of times the slice utilization exceeds the 100% utilization for the ATN-OBY slice and the ATN-PSD slice respectively.



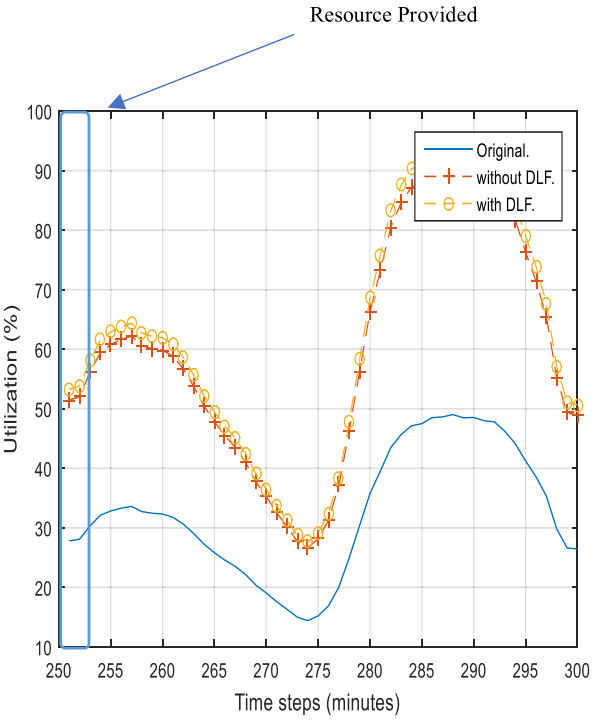
(a) ATN-OBY



(b) ATN-PSD



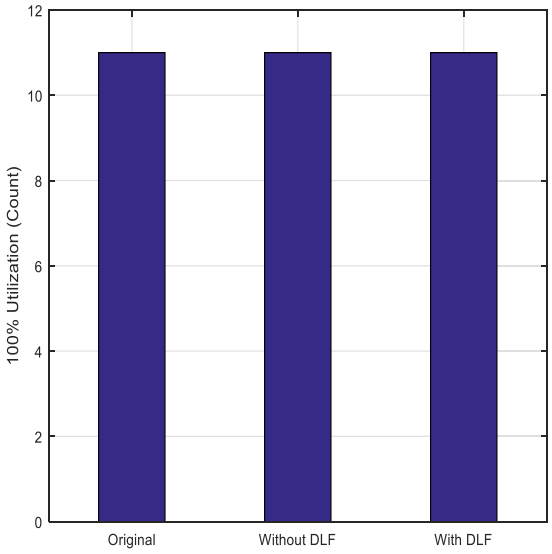
(c) LTE



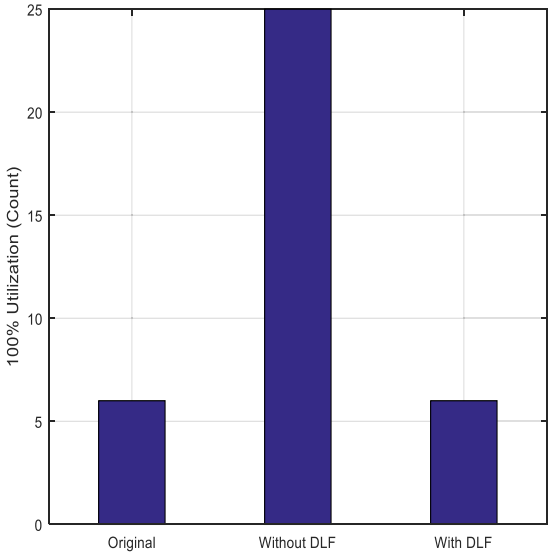
(d) MPLS

FIGURE 20. Slices utilizations in $W_a = 6$.

No slice overutilization was observed using the proposed DLF compared to the original resource allocation without DLF, confirming the effectiveness of the proposed algorithms. Fig 14 shows slice utilization for window 3 for all $t_j \epsilon \widehat{y}_t$ *in* $SL_i \epsilon W_a$ ($a = 3$ and $j = 101$ to 150) with and without DLF.



(a) ATN-OBY



(b) ATN-PSD

FIGURE 21. Counts of overutilization in $W_a = 6$.

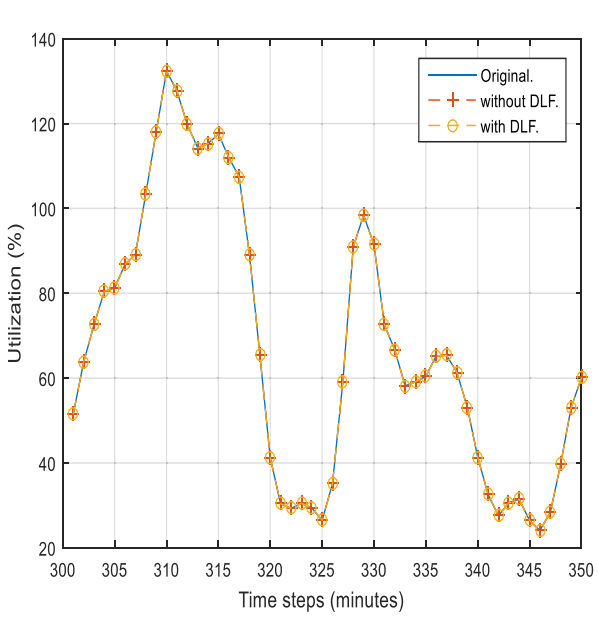
In this window, ATN-OBY and ATN-PSD slices provide resources from 100–105-time steps to LTE slice without using DLF due to their low utilization $\beta < 40\%$(blue box). Form other side, at time step 120, ATN-OBY requests a resource in which LTE and MPLS slices provide since ATN-OBY's utilization are approaching 100% utilization (in blue-original) and (in yellow with DLF). Conversely, due to its low utilization $\beta < 40\%$, the MPLS slice provides resources at time steps 104 and 106 to the LTE slice. In this window, the ATN-PSD slice does not provide any resources because the supply and demand allocation calculations were based on forecasted resource consumption. Therefore, there will is prior knowledge of whether there will be resource starvation or not before deciding not to consider (drop) the resource provider (doner) from being considered as a candidate for a resource provider $\widehat{\boldsymbol{P}}\{..\}$. The yellow line shows that the slice utilization is kept the same as the original slice utilization since the slice is not considered a slice provider after using our proposed DLF. The same analysis is applied to the ATN-OBY slice for time steps 100 to 120. The above resource allocation results show that resource allocation using DLF (yellow line) reduced overutilization compared to others. Figs 15a and Figs 15b show the count of times the slice utilization exceeds the 100% utilization mark for the ATN-OBY slice and ATN-PSD slice respectively.

From Fig 15a and Fig 15b, resource allocation with DLF reduced the count of 100% overutilization for the ATN-OBY slice and maintained the exact count compared to the original ratio for the ATN-PSD slice. In contrast to the ATN-OBY slice, the proposed resource allocation with DLF improved the resource allocation by reducing the number of overutilization counts as depicted in Fig 15b. Fig 16 shows slice utilization for window 4 for all $t_j \epsilon \widehat{y}_t$ *in* $SL_i \epsilon W_a$ ($a = 4$ and $j = 151$ to 200) with and without DLF.

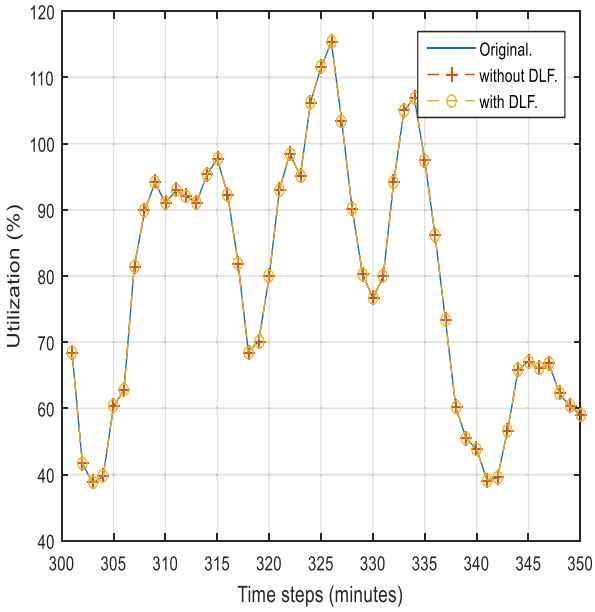
In Fig 16, the LTE slice is a resource requester since $\gamma > 60\%$, whereas the ATN-OBY and ATN-PSD slices provide resources to the LTE slice at step 151 (due to their low utilization of $\beta < 40\%$ in the blue box). This eventually leads to resource starvation for ATN-OBY, and ATN-PSD slices as depicted in the orange line in Figs 16a and 16b respectively. In contrast, when using DLF, since the supply and demand resource allocation were based on forecasted resource consumption, both slices are exempted from the resource providers' list $\widehat{P}\{..\}$ as illustrated by the yellow line in Figs 16a and 16b. Accordingly, there is prior knowledge of whether there will be a resource starvations or not, before deciding not to consider (drop) the resource provider (doner) as a candidate for a resource provider. The yellow line shows that the slice utilization is kept the same as the original slice utilization since the slice is not considered a slice provider after using the proposed DLF. Conversely, the MPLS slice provides resource to the LTE slice at steps 153 and 155 respectively. On the other hand, regarding resource allocation with DLF, ATN-OBY slice receives resources from LTE slice at step 162. It is evident that the resource allocation with DLF reduces the overall utilization and the count of overutilization. Fig 17a and Fig 17b show the number of times that the slice utilization exceeds the 100% utilization mark for ATN-OBY and ATN-PSD slices respectively.

From Fig 17a and Fig 17b, resource allocation using DLF in ATN-OBY slice reduces the overutilization count compared to the original and allocation without DLF. Likewise, in the ATN-PSD slice, the allocation using DLF reduces the overutilization to zero, similar to the actual utilization. Fig 18 shows slice utilization for window 5 for all $t_j \epsilon \widehat{y}_t$ *in* $SL_i \epsilon W_a$ ($a = 5$ and $j = 201$ to 250) with and without DLF.

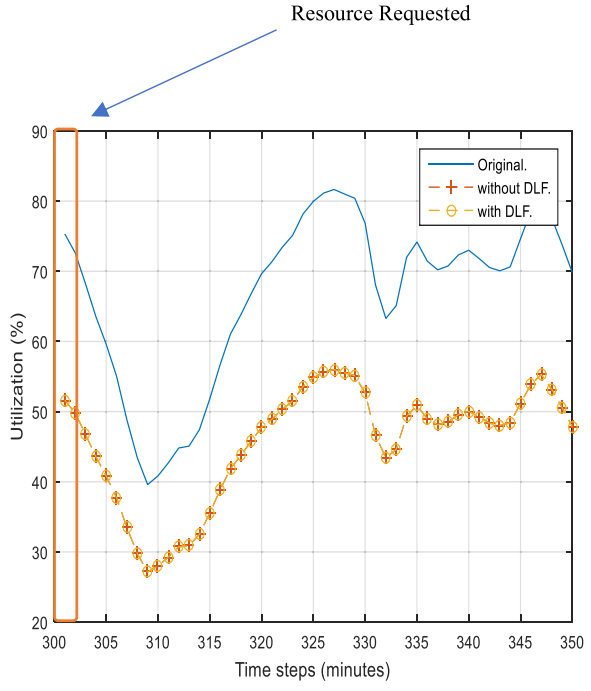
In Fig 18, the ATN-OBY, ATN-PSD, and MPLS slices (due to their low utilization $\beta < 40\%$ in the blue box) provide resources to the LTE slice at time step 201 since



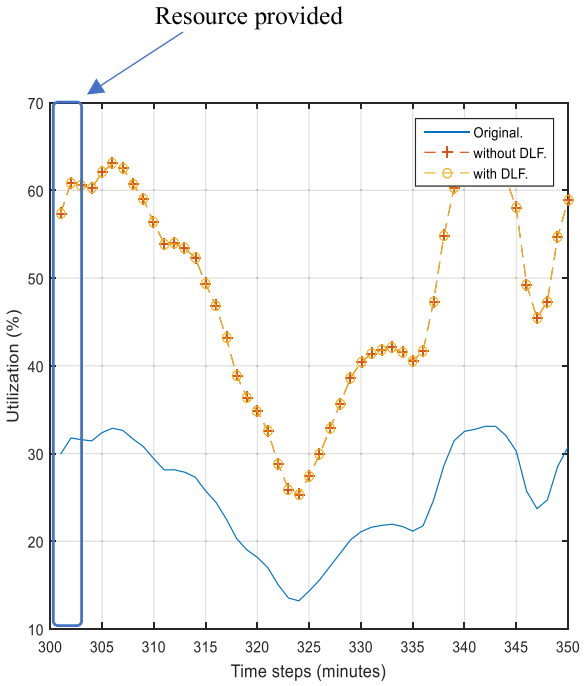
(a) ATN-OBY



(b) ATN-PSD



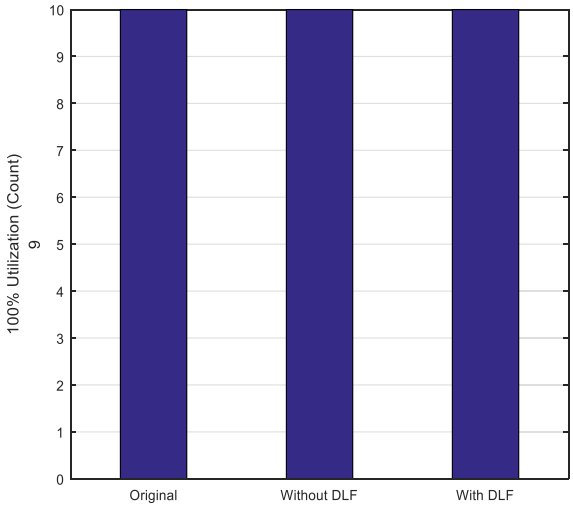
(c) LTE



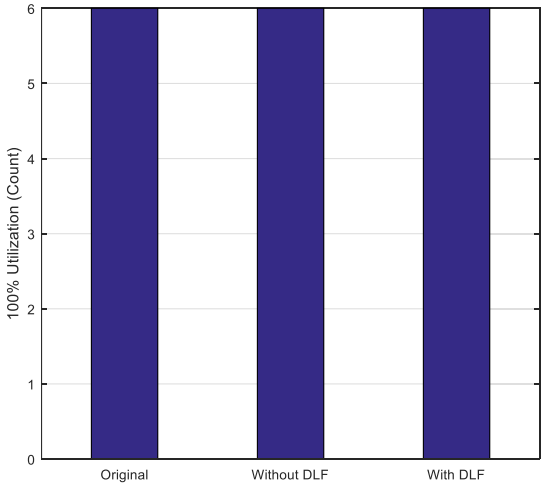
(d) MPLS

FIGURE 22. Slices utilizations in $W_a = 6$.

LTE is a resource requester with $\gamma > 60\%$. This eventually leads to resource starvation for ATN-OBY, and ATN-PSD as depicted in the red line in Figs 18a and 18b. Meanwhile, for resource allocation using DLF, the MPLS slice only provides the requested resources and compensates the dropped resources from ATN-OBY and ATN-PSD since the supply and demand resource allocation calculations were based on forecasted resource consumption; both slices are exempted from the resource providers list $\widehat{P}\{..\}$. This explains the low utilization in ATN-OBY and ATN-PSD (in yellow aligned with the actual utilization). On the other hand, ATN-OBY also requested resources at time step 210 provided by LTE slice. ATN-PSD also requested resources at time step 212, which LTE also provided. Figs 19a and 19b show the count



(a) ATN-OBY



(b) ATN-PSD

FIGURE 23. Counts of overutilization in $W_a = 7$.

of times that the slice utilization exceeds the 100% utilization for ATN-OBY and ATN-PSD, respectively

From Fig 19, the resource management using DLF outperform resource allocation without DLF and improves slice utilization for both resource providers ATN-OBY and ATN-PSD slices. Fig 20 shows slice utilization for window 6 for all $t_j \epsilon \widehat{y}_t$ *in* $SL_i \epsilon W_a$ ($a = 6$ and $j = 251$ to 300) with and without DLF.

In Fig 20, in Window 6 at time step 251 and without DLF, the ATN-PSD and the MPLS slices (due to their low utilization $\beta < 40\%$ in the blue box) provide resources to LTE slice since $\gamma > 60\%$. Meanwhile, using DLF and given that the supply and demand resource allocation calculations are based on forecasted resource consumption, ATN-PSD slice was exempted from the resource providers list $\widehat{P}\{..\}$ (Yellow line aligned with the actual) as in Fig 20b. Thus, the MPLS compensates for the lost amount with extra resources. This justifies the elevated utilization in MPLS for allocation using DLF (Yellow line) in Fig 20d. Fig 21a and Fig 21b show the count of times that the slice utilization exceeds the 100% utilization for the ATN-OBY slice and ATN-PSD slice, respectively.

In Fig 21, ATN-PSD resource allocation significantly improved overutilization compared to the allocation without DLF. Fig 22 shows slice utilization for window 7 for all $t_j \epsilon \widehat{y}_t$ *in* $SL_i \epsilon W_a (a = 7$ and $j = 300$ to $350)$ with and without the DLF.

In windows 7, only the MPLS slice with β <40% provides resources to LTE with γ >60%. Meanwhile the ATN-OBY and ATN-PSD slices do not provide or receive any resources. Fig 23a and 23b show the count of times that the slice utilization exceeds the 100% utilization for the ATN-OBY and ATN-PSD slices respectively.

In overall, DLVisor can improve resource allocation compared to our benchmark (EnterpriseVisor) by

1- Reducing and eliminating slice overutilization in EnterpriseVisor
2- Reduce resource starvation resulted from resource donation performed by the resource management module in the EnterpriseVisor
3- Improving the overall vSDN slice utilization

VI. CONCLUSION

In conclusion, the paper shows the development and implementation of methods for the slice (bandwidth) resource management in vSDN. It was found that resource management and bandwidth resource allocation, particularly in the current actual time, can lead to resource starvation due to resource overutilization; especially given that the calculation of the resource supply and demand in the related resource management solutions did not take future demands and rapid changes in traffic profiles into account. Therefore, proactive, intelligent resource management frameworks are needed. Thus, accurate and robust resource (traffic) forecasting algorithms are crucial. Accordingly, DLVisor was developed with a dynamic learning framework which combines smooth-aided ML algorithms to learn (forecast) future demands. It reacts and adapts to any significant changes in traffic profile using concept changes detectors and significant tests. The window-based methods were employed to reduce or eliminate the fluctuations in the data traffic, which can deteriorate the ML performance as per the previous studies. Finally, the improved dynamic learning framework are applied to the resource management framework to provide improved resources utilization and eliminate the overutilization resulting from form the supply and demand calculations.

REFERENCES

[1] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 655–685, 1st Quart., 2016.
[2] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: The Linux virtual machine monitor," in *Proc. Linux Symp.*, Canada, America, Jun. 2007, pp. 225–230.
[3] C. A. Waldspurger, "Memory resource management in VMware ESX server," *ACM SIGOPS Operating Syst. Rev.*, vol. 36, pp. 181–194, Dec. 2002.

[4] A. A. Blenk, "Towards virtualization of software-defined networks: Analysis, modeling, and optimization," Dept. Electron. Inf. Technol., Technische Universität München, Munich, Germany, Tech. Rep. 21.11.2017, 2018.
[5] F. Rodríguez-Haro, F. Freitag, L. Navarro, E. Hernánchez-sánchez, N. Farías-Mendoza, J. A. Guerrero-Ibáñez, and A. González-Potes, "A summary of virtualization techniques," *Proc. Technol.*, vol. 3, pp. 267–272, Jan. 2012.
[6] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005.
[7] M. Yu, J. Rexford, X. Sun, S. Rao, and N. Feamster, "A survey of virtual LAN usage in campus networks," *IEEE Commun. Mag.*, vol. 49, no. 7, pp. 98–103, Jul. 2011.
[8] A. Belbekkouche, Md. M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1114–1128, 4th Quart., 2012.
[9] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Services Appl.*, vol. 9, no. 1, pp. 1–99, Dec. 2018.
[10] M. H. H. Khairi, S. H. S. Ariffin, N. M. A. Latiff, K. M. Yusof, M. K. Hassan, F. T. Al-Dhief, M. Hamdan, S. Khan, and M. Hamzah, "Detection and classification of conflict flows in SDN using machine learning algorithms," *IEEE Access*, vol. 9, pp. 76024–76037, 2021.
[11] K. Z. Ghafoor, L. Kong, D. B. Rawat, E. Hosseini, and A. S. Sadiq, "Quality of service aware routing protocol in software-defined Internet of Vehicles," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2817–2828, Apr. 2019.
[12] M. K. Hassan, S. H. Ariffin, S. K. Syed-Yusof, N. E. Ghazali, and M. E. Kanona, "Analysis of hybrid non-linear autoregressive neural network and local smoothing technique for bandwidth slice forecast," *TELKOMNIKA, Telecommun. Comput. Electron. Control*, vol. 19, no. 4, pp. 1078–1089, 2021.
[13] M. Alauthman, N. Aslam, M. Al-kasassbeh, S. Khan, A. Al-Qerem, and K.-K. R. Choo, "An efficient reinforcement learning-based Botnet detection approach," *J. Netw. Comput. Appl.*, vol. 150, Jan. 2020, Art. no. 102479.
[14] X. Li, S. Li, P. Zhou, and G. Chen, "Forecasting network interface flow using a broad learning system based on the sparrow search algorithm," *Entropy*, vol. 24, no. 4, p. 478, Mar. 2022.
[15] A. R. Abdellah and A. Koucheryavy, "VANET traffic prediction using LSTM with deep neural network learning," in *Proc. Int. Conf. Next Gener. Wired/Wireless Netw.* Cham, Switzerland: Springer, 2020, pp. 281–294.
[16] S. K. Singh, M. M. Salim, J. Cha, Y. Pan, and J. H. Park, "Machine learning-based network sub-slicing framework in a sustainable 5G environment," *Sustainability*, vol. 12, no. 15, p. 6250, Aug. 2020.
[17] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Comput. Netw.*, vol. 61, pp. 5–23, Mar. 2014.
[18] H. Ishio, J. Minowa, and K. Nosu, "Review and status of wavelength-division-multiplexing technology and its application," *J. Lightw. Technol.*, vol. 2, no. 4, pp. 448–463, Aug. 1984.
[19] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni, "Traffic engineering with MPLS in the Internet," *IEEE Netw.*, vol. 14, no. 2, pp. 28–33, Apr. 2000.
[20] A. Leon-Garcia and L. G. Mason, "Virtual network resource management for next-generation networks," *IEEE Commun. Mag.*, vol. 41, no. 7, pp. 102–109, Jul. 2003.
[21] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. Jackson, and A. Lambeth, "Network virtualization in multi-tenant datacenters," in *Proc. 11th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, 2014, pp. 203–216.
[22] S. Shenker, L. Peterson, and J. Turner, "Overcoming the Internet impasse through virtualization," in *Proc. ACM HotNets-III*, 2004, pp. 1–8.
[23] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proc. ACM SIGCOMM Conf.*, Aug. 2011, pp. 242–253.
[24] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Comput.*, vol. 17, no. 2, pp. 20–27, Mar. 2013.
[25] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: A survey," *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 24–31, Nov. 2013.
[26] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014.
[27] R. Sherwood, M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T.-Y. Huang, P. Kazemian, M. Kobayashi, J. Naous, S. Seetharaman, D. Underhill, T. Yabe, K.-K. Yap, Y. Yiakoumis, H. Zeng, G. Appenzeller, R. Johari, N. McKeown, and G. Parulkar, "Carving research slices out of your production networks with OpenFlow," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 129–130, Jan. 2010.
[28] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium*, vol. 1, p. 132, Oct. 2009.
[29] N. Van Giang and Y. H. Kim, "Slicing the next mobile packet core network," in *Proc. 11th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2014, pp. 901–904.
[30] X. Jin, J. Rexford, and D. Walker, "Incremental update for a compositional SDN hypervisor," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, Aug. 2014, pp. 187–192.
[31] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, W. Snow, and G. Parulkar, "OpenVirteX: A network hypervisor," in *Proc. Open Netw. Summit (ONS)*, 2014, pp. 1–9.
[32] R. Doriguzzi-Corin, E. Salvadori, M. Gerola, M. Suñé, and H. Woesner, "A datapath-centric virtualization mechanism for OpenFlow networks," in *Proc. 3rd Eur. Workshop Softw. Defined Netw.*, Sep. 2014, pp. 19–24.
[33] X. Jin, J. Gossels, J. Rexford, and D. Walker, "CoVisor: A compositional hypervisor for software-defined networks," in *Proc. 12th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, 2015, pp. 87–101.
[34] L. Liao, A. Shami, and V. C. M. Leung, "Distributed FlowVisor: A distributed FlowVisor platform for quality of service aware cloud network virtualisation," *IET Netw.*, vol. 4, no. 5, pp. 270–277, Sep. 2015.
[35] J.-L. Chen, Y.-W. Ma, H.-Y. Kuo, C.-S. Yang, and W.-C. Hung, "Software-defined network virtualization platform for enterprise network resource management," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 179–186, Apr. 2016.
[36] Y. Han, J. Li, D. Hoang, J.-H. Yoo, and J. W. Hong, "An intent-based network virtualization platform for SDN," in *Proc. 12th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2016, pp. 353–358.
[37] H. Yamanaka, E. Kawai, and S. Shimojo, "AutoVFlow: Virtualization of large-scale wide-area OpenFlow networks," *Comput. Commun.*, vol. 102, pp. 28–46, Apr. 2017.
[38] H. Yamanaka, E. Kawai, S. Ishii, and S. Shimojo, "AutoVFlow: Autonomous virtualization for wide-area OpenFlow networks," in *Proc. 3rd Eur. Workshop Softw. Defined Netw.*, Sep. 2014, pp. 67–72.
[39] Y. Han, T. Vachuska, A. Al-Shabibi, J. Li, H. Huang, W. Snow, and J. W.-K. Hong, "ONVisor: Towards a scalable and flexible SDN-based network virtualization platform on ONOS," *Int. J. Netw. Manage.*, vol. 28, no. 2, p. e2012, Mar. 2018.
[40] S. Agliano, M. Ashjaei, M. Behnam, and L. L. Bello, "Resource management and control in virtualized SDN networks," in *Proc. Real-Time Embedded Syst. Technol. (RTEST)*, 2018, pp. 47–53.
[41] V. Struhár, M. Ashjaei, M. Behnam, S. S. Craciunas, and A. V. Papadopoulos, "DART: Dynamic bandwidth distribution framework for virtualized software defined networks," in *Proc. 45th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, vol. 1, Oct. 2019, pp. 2934–2939.
[42] L. Leonardi, L. Lo Bello, and S. Aglianó, "Priority-based bandwidth management in virtualized software-defined networks," *Electronics*, vol. 9, no. 6, p. 1009, Jun. 2020.
[43] G. Yang, B.-Y. Yu, H. Jin, and C. Yoo, "Libera for programmable network virtualization," *IEEE Commun. Mag.*, vol. 58, no. 4, pp. 38–44, Apr. 2020.
[44] G. Yang, Y. Yoo, M. Kang, H. Jin, and C. Yoo, "Bandwidth isolation guarantee for SDN virtual networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2021, pp. 1–10.
[45] A. Ahmadian and M. Ahmadi, "DC-CAMP: Dynamic controller creation, allocation and management protocol in SDN," *Wireless Pers. Commun.*, vol. 125, pp. 531–558, Feb. 2022.
[46] M. K. Hassan, S. H. Syed Ariffin, N. E. Ghazali, M. Hamad, M. Hamdan, M. Hamdi, H. Hamam, and S. Khan, "Dynamic learning framework for smooth-aided machine-learning-based backbone traffic forecasts," *Sensors*, vol. 22, no. 9, p. 3592, May 2022.
[47] G. Sun, K. Xiong, G. O. Boateng, G. Liu, and W. Jiang, "Resource slicing and customization in RAN with dueling deep Q-network," *J. Netw. Comput. Appl.*, vol. 157, May 2020, Art. no. 102573, doi: 10.1016/j.jnca.2020.102573.

MOHAMED KHALAFALLA HASSAN received the B.Sc. degree in computer engineering from Future University, Sudan, in 2004, and the M.Sc. degree in communication network engineering from University Putra Malaysia (UPM), in 2009. He is currently pursuing the Ph.D. degree in communication engineering with University Technology Malaysia (UTM). He is also an Associate Professor with Future University. He is also a researcher and an ICT specialist with 17 years of wide range of research and ICT experience. He has published 20 papers in international peer-reviewed conferences and journals. His research interests include forwards scattering radar, machine learning, NFV, vSDN, and resources management in communication networks.

SHARIFAH HAFIZAH SYED ARIFFIN (Senior Member, IEEE) received the B.Eng. degree (Hons.) in London, in 1997, the M.E.E. degree from Universiti Teknologi Malaysia, in 2001, and the Ph.D. degree from the Queen Mary, University of London, London, in 2006. She is currently an Associate Professor with the Faculty of Electrical Engineering, Universiti Teknologi Malaysia. She had published 116 articles, 17 copyrights, one integrated circuit, and one trademark. Her current research interests include the Internet of Things, ubiquitous computing and smart devices, wireless sensor networks, ipv6, handoff management, networks, and mobile computing systems.

SHARIFAH KAMILAH SYED-YUSOF received B.Sc. degree in electrical engineering from George Washington University, USA, in 1988, and the M.E.E. and Ph.D. degrees from UTM, in 1994 and 2006, respectively. She is currently a Full Professor with the Faculty of Electrical Engineering, UTM. Her research interest includes wireless communication.

NURZAL EFFIYANA BINTI GHAZALI received the M.S. degree in electrical engineering from the Shibaura Institute of Technology and the Ph.D. degree from UTM, in 2016. She is currently doing research in mobile computing, mobility management, network communication protocol, and sport monitoring systems.

MOHAMMED E. A. KANONA received the B.Sc., M.Sc., and Ph.D. degrees in telecommunication engineering from Future University, Sudan. He is currently the Deputy Dean of the Faculty of Telecommunication and Space Technology and the Head of the IoT Research Center. He actively involved in research about forward scattering radar. His research interests include information theory, SDN, the IoT, cloud computing, machine learning and neural networks, and mobile communication. He received the Best Paper Award from ICCCEEE20 Conference.

KHALID S. MOHAMED (Member, IEEE) received the bachelor's degree in telecommunication engineering from Future University, Sudan, in 2011, and the Master of Engineering degree in telecommunication and the Ph.D. (Engineering) degree in telecommunication from Multimedia University, Malaysia, in 2014 and 2020, respectively. He has been registered with the Sudanese Engineering Council (SEC) as a Professional Engineer, since January 2022, and the Board of Engineers Malaysia (BEM) as a Graduate Engineer, since April 2019. He is currently an Assistant Professor with the Faculty of Telecommunication and Space Technology and the Acting Director of the Innovation, Research and Development Center (IRDC), Future University. His research interests include cellular communication, 5G, intelligent reflective surfaces (IRSs), beamforming, and interference management in wireless networks.

MUTAZ H. H. KHAIRI (Senior Member, IEEE) received the B.S. degree in computer engineering from Future University, in 2002, and the M.S. degree in electrical engineering from Linkoping University, in 2007. He is currently pursuing the Ph.D. degree with Universiti Technologi Malaysia (UTM). From 2002 to 2007, he was a Lecturer with the Faculty of Engineering, Future University, and the Director of the Information Technology Department. He is also a researcher and an ICT specialist with 16 years of wide range of research and ICT experience. His research interests include software define networks (SDNs), machine learning, telecommunication networks, and antenna design and implementation.

MOSAB HAMDAN (Senior Member, IEEE) received the B.Sc. degree in computer and electronic system engineering from the University of Science and Technology (UST), Sudan, in 2010, the M.Sc. degree in computer architecture and networking from the University of Khartoum (UofK), Sudan, in 2014, and the Ph.D. degree in electrical engineering (computer networking) from the Faculty of Engineering, School of Electrical Engineering, Universiti Teknologi Malaysia (UTM), Malaysia, in 2021. From 2010 to 2015, he was a Teaching Assistant and a Lecturer with the Department of Computer and Electronics System Engineering, Faculty of Engineering, University of Science and Technology (UST). He is currently a Researcher with the Interdisciplinary Research Center for Intelligent Secure Systems, King Fahd University of Petroleum and Minerals, Saudi Arabia. His current research interests include software-defined networking (SDN), load balancing, network traffic classification, the Internet of Things (IoT), cloud computing, network security, and future networks.