A Resilience-Oriented and NFV-Supported Scheme for Failure Detection in Software-Defined Networking

by

He Li

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

© He Li, Ottawa, Canada, 2018

Abstract

As a recently emerging network paradigm, Software-Defined Networking (SDN) has attracted considerable attention from both industry and academia. The most significant advantage of SDN is that the paradigm disassociates the control logic (i.e., control plane) from the forwarding process (i.e., data plane), which are usually integrated into traditional network devices. Thanks to the property of *centralized control*, SDN enables the flexibility of dispatching flow policies to simplify network management. However, this property also makes the SDN environment vulnerable, which will cause network paralysis when the sole SDN controller runs malfunction. Although several works have been done on deploying multiple controllers to address the failure of a centralized controller, their drawbacks are leading to inefficiency and balance loss of controller utilization, provoking resource idling as well as being incapable to suffice flow outburst.

Additionally, the network operators often put a great deal of effort into discovering failure nodes to recover their networks, which can be mitigated by applying failure detection before the network deterioration occurs. Network traffic prediction can serve as a practical approach to evaluate the state of the OpenFlow-based switch and consequently detect SDN node failures in advance. As far as prediction solution is concerned, most researchers investigate either statistical modeling approaches, such as Seasonal Autoregressive Integrated Moving Average (SARIMA), or Artificial Neural Network (ANN) methods, like Long Short-Term Memory (LSTM) Neural Network. Nonetheless, few of them study the model merging these two mechanisms regarding multi-step prediction.

This thesis proposes a novel system associated with Network Function Virtualization (NFV) technique to enhance the resilience of SDN network. A hybrid prediction model based on the combination of SARIMA and LSTM is introduced as part of the detection module of this system, where the potential node breakdown can be readily determined so that it can implement smart prevention and fast recovery without human interaction. The results show the proposed scheme improves the performance concerning time complexity compared with that of previous work, reaching up to 95% accuracy while shortening the detection and recovery time by the new combined prediction model.

Acknowledgements

It is a great pleasure to conduct my research and complete my thesis in PARADISE Laboratory, where I have received valuable guidance and suggestions. Primarily, I would like to offer my sincere appreciation to my supervisor Prof. Azzedine Boukerche, who gave me the chance to explore the potentials within his sea of knowledge, and provided valuable and insightful feedback to my work. His passion and rigorousness influenced my attitude towards study and future work, which will help me conquer obstacle along the way.

I would also like to express my deepest gratitude to Dr. Robson E. De Grande, who has been a great mentor during school and my research. His patience and advice often cleared my confusion and lightened my inspiration. Without his motivation, kindness, knowledge and rich experience, I cannot imagine how many difficulties I might have encountered.

Moreover, I feel especially grateful for what Dr. Peng Sun has done for me. He helped proofread my thesis and provided many critical comments, enhancing the logicality and the readability of my thesis manifold.

Finally, I would like to thank my parents, friends, and colleagues for giving tremendous support and encouragement, which keeps me moving forward. This memory will be my priceless treasure lasting forever.

Publication Related to Thesis

He Li, Robson Eduardo De Grande, and Azzedine Boukerche. "An efficient CPP solution for resilience-oriented SDN controller deployment." *In Proceedings of 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2017.

Table of Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

ACF	Autocorrelation Function
AIC	Akaike Information Criterion
ANN	Artificial Neural Network
API	Application Programming Interface
ARIMA	Autoregressive Integrated Moving Average
BF	Brute Force
BFD	Bidirectional Forwarding Detection
BIC	Bayesian Information Criterion
BSS	Business Support System
CAPEX	Capital Expense
CCMM	Control Channel Maintenance Module
COP	Control Orchestration Protocol
CPC	Consistent Policy Composition
CPE	Customer Premises Equipment
CPP	Controller Placement Problem
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DCP	Dynamic Controller Provisioning
DDoS	Distributed Denial of Service
DGC	Dependency Graph Construction
DHCP	Dynamic Host Configuration Protocol
DMAM	Decision Making Application Module
DTC	Dynamic Tensor Completion
ELM	Extreme Learning Machine
EMD	Empirical Mode Decomposition
ENN	Elman Neural Network
ESTI	European Telecommunications Standards Institute
FLI	Failure Location Identification
FMC	Follow Me Cloud
FMR	Flow Migration Reduction
GA	Genetic Algorithm

GRU	Gated Recurrent Units
HA	High Availability
ICN	Industrial Control Network
ICS	Industrial Control System
ICT	In-band Control Tree
IDMD	Inter-Domain Mobility Database
ILP	Integer Linear Programming
IMF	Intrinsic Mode Function
IOS	Internetwork Operating System
ISP	Internet Service Provider
ITS	Intelligent Transportation System
KNN	K-Nearest Neighbor
KVM	Kernel-based Virtual Machine
LSTM	Long Short-Term Memory
MANO	Management and Orchestration
MAPE	Mean Absolute Percentage Error
MIMO	Multiple-Inputs Multiple-Outputs
MISO	Multiple-Inputs Single-Outputs
ML	Machine Learning
MMP	Multi-Model Prediction
MNH	Multi-domain Network Hypervisor
MP	Management Pattern
NAT	Network Address Translation
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NIC	Network Interface Card
NMS	Network Management System
NMSE	Normalized Mean Square Error
NN	Neural Network
NRMSE	Normalized Root Mean Square Error
NV	Network Virtualization
OF	OpenFlow
ONOS	Open Network Operating System
OPEX	Operating Expense
OS	Operating System
OS3E	Open Science, Scholarship and Services Exchange
OSI	Open System Interconnection
OSS	Operation Support System
PACF	Partial Autocorrelation Function
PAM	Path Alive Monitoring
PCI	Peripheral Component Interconnect
PSA	Pareto Simulated Annealing
PSO	Particle Swarm Optimization

QoS	Quality of Service
ReON	Resilience-Oriented and NFV-supported
RNN	Recurrent Neural Network
ROM	Resilient Orchestration Module
SA	Simulated Annealing
SARIMA	Seasonal Autoregressive Integrated Moving Average
SDN	Software-Defined Networking
SOM	Self-Organizing Map
SP	Serial Propagated
SQL	Structured Query Language
STN	Software Transactional Networking
SVM	Support Vector Machine
SVR	Support Vector Regression
VEPA	Virtual Ethernet Port Aggregator
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
VON	Virtual Optical Network
VPN	Virtual Private Network
VTN	Virtual Tenant Network
WIDE	Widely Integrated Distributed Environment
WRED	Weighted Random Early Detection

Chapter 1

Introduction

This chapter introduces the background of NFV and SDN, and presents the motivation and objectives of the research, which is related to the problems that SDN paradigm probably faces. Additionally, it summarizes the contributions of this thesis, and outlines the content construction at the end of this chapter.

1.1 Motivation

In order to meet the requirements of different services for specific purposes, Internet Service Providers (ISPs) in traditional network have to incrementally deploy diverse network devices, which are called middleboxes. The increasing quantity of physical equipment leads to a series of Capital Expenses (CAPEX) and Operating Expenses (OPEX), where CAPEX stands for the amount of cost in purchasing new appliances and OPEX denotes a set of continuous overhead for middlebox maintenance and fees of annexes [5]. Moreover, the life cycle of middlebox hardware cannot immediately catch up with the users' needs due to the expensive prices and the long process of purchase (e.g., average four years) [18]. To address the dilemma above, Network Function Virtualization (NFV) [19] has been introduced as a solution that delivers network services through utilizing virtualization technology.

Furthermore, It is also a challenge for network operators to deal with a large number of communication devices in the traditional network, where the operators need to maintain the entire network by configuring each router since the routing logic and forwarding decisions are vertically integrated. The management issue occurs both in wireless networks and wired metworks [20, 21, 22]. Therefore, Software-Defined Networking (SDN) [23] was presented to tackle the management difficulties of complex network equipment, where control logic is abstracted as another independent network service from the forwarding function. Conventional SDN model consists of a controller and several switches, where the controller is responsible for managing and dispatching routing rules when switches only deal with forwarding functions according to the flow table. This mechanism simplifies the design of switches and enables the centralized control, which shows a decent case with 100% link usage for Google [24].

In general, the scalability of SDN is limited due to the capability of the single-controller architecture and the physical positioning of the switches [25]. However, NFV technique facilitates the decoupling of controlling functions from the hardware to implement them as applications running in the Virtual Machines (VMs). Additionally, the forwarding functions can be separated from the switches and be set up in the data center by NFV, which allows flexible network deployment. The use of NFV in SDN leads to several benefits [26], such as lower CAPEX and OPEX, shorter deployment period with lower risk, easier to scale, and so on. Therefore, many works of literature have been presented seeking the cooperation between SDN and NFV.

Although there are many advantages that the cooperation between these two paradigms can bring into production, several technical challenges exist in associating NFV with SDN [27], such as virtualization performance, controller security, controller placement, and resilience, which reveals a fundamental research point in the future.

1.2 Objectives

SDN Resilience is a common issue. An abundance of resilient solutions have been proposed addressing the problem by employing distributed controllers [28, 29, 30, 31]. Some works attempt to provide controlling function migration, which deploys the controller into a VM and recovers the network functions by restoring the controller VM [32, 33]. Meanwhile, some others focus on adopting an alternative path from the source switch to the controller to achieve SDN controller recovery by changing the flow table [7].

In this thesis, the major objectives are to design a resilience-oriented and NFV-supported scheme in SDN environment, which can monitor the network, predict failures and recover from breakdowns automatically without human interaction, and to improve the recovering efficiency of SDN. The scheme is supposed to include the essential functions as follows:

1. **Monitor function**. This function supervises the traffic flow of entire SDN network by collecting the statistical data from the data plane. This behavior serves two purposes: 1) to obtain the training feeds, which works as the inputs of the prediction function to build a solid prediction model of network traffic; 2) to provide the real-time data to the detection function for evaluating, which allows for determining the network failure.

2. **Prediction function**. This function leverages historical data to build a precise forecast model by using an improved hybrid method that consists of SARIMA-based and ANN-based approaches. The proposed method aims at enhancing the accuracy of multi-step prediction, which can serve as the baseline of failure detection.

3. **Detection function**. This function determines whether or not the disruption is about to occur according to the baseline of normal network traffic created by the proposed forecast method. The recovery action is only triggered when the network traffic flow is beyond the threshold of the baseline.

4. **Recovery function**. To recover network from both controller and *datapath* failures, this function selects the backup plan including controller candidates and secondary datapath beforehand, applying the recovery when the potential failures are detected.

In the proposed SDN scheme, the selection of the controller candidates is based on if the positions of the backup controllers can reach other switches with minimum overheads, which represents latency here. The solutions to the Controller Placement Problem (CPP) were introduced to place the controller in a location with the least latency within the SDN domain in order to enhance the QoS of the network [34]. Thus, we can consider the proposed controller resilience problem as a variant of CPP, which can be solved by using the CPP-related methods.

The failure detection can be utilized by the proposed scheme to improve the efficiency of implementing resilience. An efficient localization protocol enables to identify the failure node quickly [35, 36, 37, 38, 39, 40, 41]. The network pattern provided by the network prediction can associate with the detection to provide better performance, enabling the system to detect the network traffic abnormity beforehand. However, the accuracy of prediction result using the proposed method still needs to be proven, in order to ensure the effectiveness of detection baseline.

1.3 Contributions

In this thesis, the contributions are summarized as follows:

1. To provide a systematic review of previous studies, this thesis classifies existing methods concerning SDN network resilience. The features and limitations of each method are outlined.

2. To abstract the controller and switch functions as VNFs that can both be deployed into the same machine, this thesis presents a novel SDN resilient scheme. These two functions are independently working that one will be halted when the other one is activated,

3. To simplify the fast recovery for the SDN controller, this thesis formulates the SDN resilience issue as CPP and applies proposed ranking algorithms to implement optimization for finding the optimum position. The proposed method improves the efficiency of selecting controller candidates while keeping up to 95% accuracy regarding the optimal baseline.

4. To enhance multi-step predicting performance under the out-of-sample condition, this thesis introduces a hybrid method consisting of SARIMA and LSTM, which can be used as the baseline of failure detection in the proposed scheme. The result illustrates that the hybrid method outperforms either single prediction approaches (i.e., SARIMA or LSTM), in multi-step prediction.

1.4 Outline

The remainder of this thesis is organized as follows: Chapter 2 introduces some related concepts on SDN and NFV, comparing their differences, which further demonstrates their collaborative opportunities and benefits. Chapter 3 gives a review of related works, including the methods of network prediction and the categorized resilient solutions in SDN. Chapter 4 proposes a resilience-oriented scheme for SDN and introduces an improved hybrid model for network traffic prediction, where it is divided into two primary sections (i.e., ReON and LSARIMA). In Chapter 5, the experimental specification is described, and results of proposed methods compared with previous approaches are discussed. Finally, Chapter 6 concludes the results and presents the direction of future research.

Chapter 2

Background

This chapter presents some technical concepts on SDN and NFV. It also breaks down their architectures, giving readers an in-depth understanding of these two paradigms, while providing an overview of the values that the cooperation between these two techniques can deliver.

2.1 Software Defined Networking

As an emerging network paradigm, SDN has obtained the attention from both academia and industry, which initially presented the concept of *programmable networks* [42, 43]. In traditional networks, control plane and forwarding plane (i.e., data plane) are bound together in a single routing appliance. Therefore, network operators need to manually configure each network device independently, which is very inefficient and time-consuming to the large-scale and complex network topology. Nonetheless, SDN decouples the network control logic from the underlying physical devices (see Figure 2.1), which facilitates the forwarding behavior to be managed via southbound protocols, such as OpenFlow [44], ForCES [45] and etc.

2.1.1 OpenFlow Switch

One of the most important elements in SDN is the OpenFlow (OF) switches that transfer network traffic regarding the forwarding policies distributed by a controller, which diminishes the overhead of a routing process. Figure 2.2 depicts the basic infrastructure of an OF switch, where the architecture incorporates three components:

- **A Flow Table** stores the flow entries that are installed by the controller when receiving the *packet-in* messages. Each flow entry consists of a packet header, an action, and statistics (see Figure 2.3). Specifically, the packet header defines the flow type, and the action section tells the switch how to handle the matched flow. In addition, the statistics collect information about each flow, including the number of packets, the total bytes and the duration time.

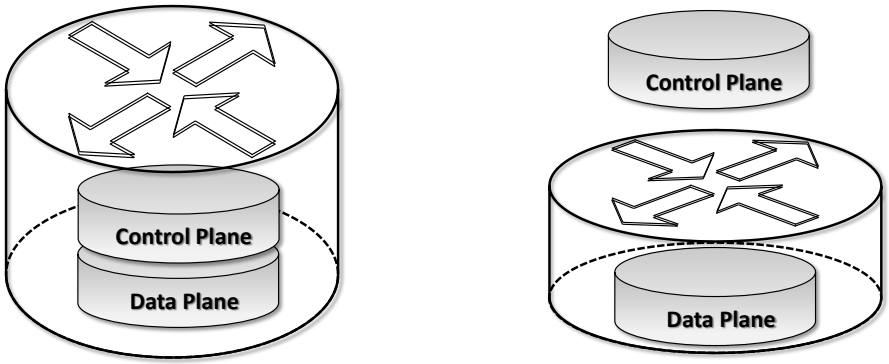


Figure 2.1: Comparison between traditional and SDN paradigms [1].

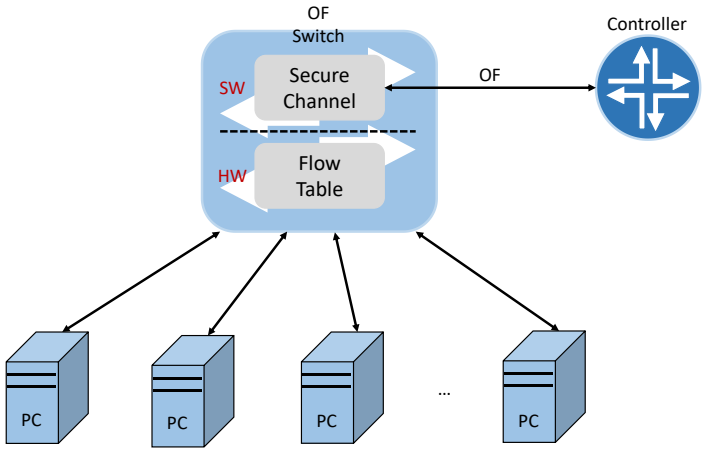


Figure 2.2: OF switch infrastructure.

- **A Secure Channel** connects the switches to one or multiple controllers, ensuring that the data are encrypted, which transmits the commands or packets between the controller and switches by using the OF protocol.
- **The OF Protocol** was presented by [2] to design *programmable networks*, whose goal is to lessen budgets of facility deployment and to accelerate implementation of new ideas. It also enables users to program the flow-table in various OF-supported switches and routers, where the network operators can separate network traffic into particular flows according to different purposes.

2.1.2 SDN Controller

Being different from the typical IP forwarding networks using routers, SDN transmits data based on OF flows instead of packets. Figure 2.4 illustrates that the controller can ad-

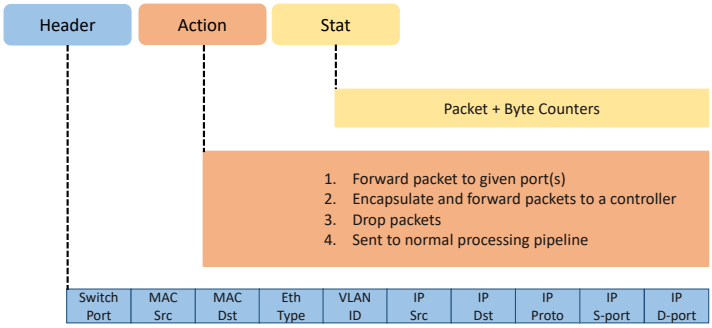


Figure 2.3: Flow entries in an OF switch [2].

ministrate the switches through southbound protocols while other applications control the communication between SDN controllers via northbound APIs. Thanks to the separation of the control and data planes, centralized control and agility are two main characteristics of the SDN network. Since the control functions of SDN can be programmed, network administrators can adjust the network configuration dynamically to meet the traffic requirements, which enables the network updates to become more limber and flexible, thereby leading to simplification of network management. Additionally, the status of switches propagates to the controller by *packet-in* packets, which allows the controller to gain a global view of the network, knowing all the information of the entire network topology, including link state, bandwidth, cost, and error. As a result, forwarding devices called SDN agents [46] can simply convey the packets by accepting the policies dispatched by controller without consideration.

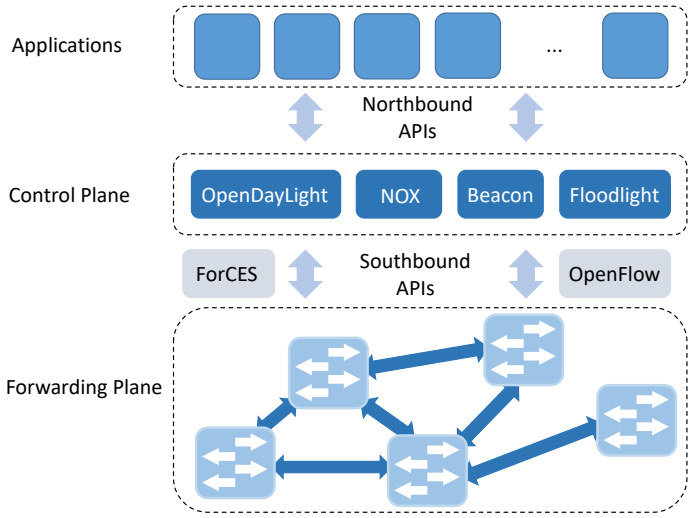


Figure 2.4: SDN structure.

There are many OF-based controllers, such as NOX [47], Beacon [48], Floodlight [49]

and OpenDaylight [50] (see Figure 2.4). Among these platforms, OpenDayLight and Open Network Operating System (ONOS) [51] are two most prevalent SDN controllers in the industry, both of which support a variety of OF protocols and provide a set of northbound APIs for building diverse network applications, in order to facilitate flexible network management. Furthermore, those open source SDN controllers are excellent examples of the cooperation between SDN and NFV, which can be deployed into VM instances. As a result, the collaboration minimizes operational expenses and balances the use of existing middleboxes and virtual network functions [52]. A detailed comparative study among current SDN controllers is conducted in [53], demonstrating the performance of the controlling function is possible not to be dedicated to specific hardware.

2.2 Virtualization

In general, traditional network services are constrained in the dedicated devices, which includes the prototype SDN controllers as well. The virtualization techniques help setting the controlling function free from the restriction of a physically fixed machine [52]. Virtualization schemes can be categorized as *hypervisor-based* or *container-based* according to the layer of deployment. Each scheme has its own advantages and disadvantages, which will be the key points for ISPs to make a decision for their implementation in NFV networks.

Hypervisor-Based virtualization simulates virtual hardware (e.g., CPU processor, memory, and storage.) based on the underlying physical machine. One or multiple Operating Systems (OSes) are installed on the top of the simulated hardware (see the left diagram in Figure 2.5). This kind of virtualization is implemented at a virtualized hardware level, meaning that there is a hypervisor running on top of the base OS. The hypervisor monitors and controls those guest OSes working on the virtual hardware (i.e., VMs including the guest OSes and simulated hardware), such as VMWare Workstation, Microsoft Virtual PC, KVM, and Xen. Because of the isolation between the guest OS and the host OS, the former can be deployed as another type of OS that is different from the latter such as a Linux guest OS, which can run on the Windows-based OS, and vice versa.

A container-based solution has attracted many attentions from the industry because of the high performance it displays [54]. In contrast to the hypervisor-based scheme, container-based virtualization works at the OS level, meaning that each container shares the same kernel of the host OS (see the right diagram in Figure 2.5). The primary merit of this type of virtualization is lightweight because the volume of each container is minuscule, which facilitates the quantity of the container to be enormous in an OS. Compared to the limited number of guest OSes running on top of the host, the large number of containers demonstrates superiority in service provisioning.

The NV or NFV design can utilize both virtualization schemes. However, network operators are supposed to consider the domain of their work when they choose one of the virtualization solutions [55, 56]. For example, if they prefer executing various duplicates of the same single function such as database SQL request, a container could be a better choice. Conversely, if the operators expect the flexibility of multiple selections for specific

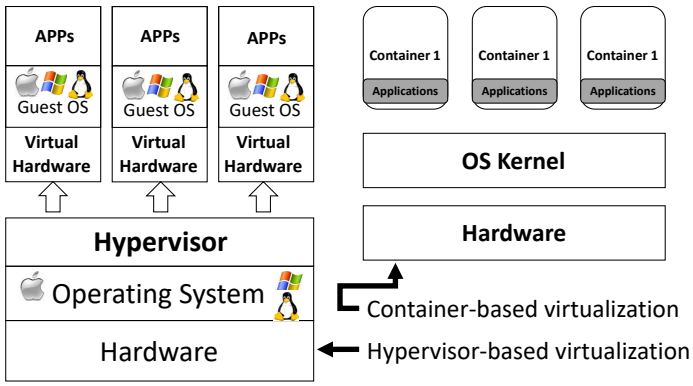


Figure 2.5: Different virtualization structures [3].

applications that run simultaneously, a hypervisor-based VM will be the preference that meets the expectations.

2.2.1 Network Virtualization

Network Virtualization (NV) or network slicing aims at virtualizing the network topology, logically isolating the virtualized nodes or tenants from each other for network requirements, which is shown in Figure 2.6. The technology of logicality isolation is not novel. Numerous organizations have adopted it for various reasons, such as scalability [57, 58, 59, 60], fault isolation, security [61] and network abstraction [62].

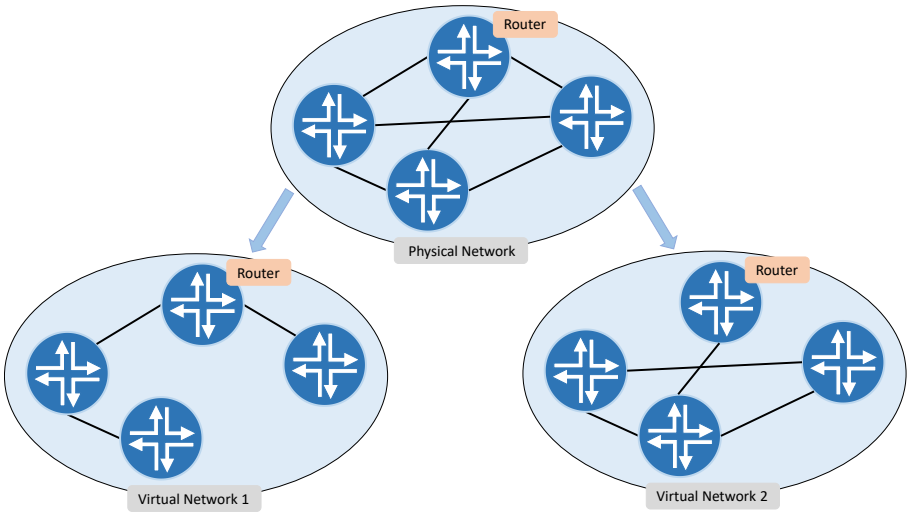


Figure 2.6: An example of network virtualization.

The NV contains plenty of aspects, including network device virtualization (e.g., virtual

NIC, virtual switch), link virtualization (e.g., VLAN, tunnels), virtual networks (e.g., VPN) and experimental testbeds (e.g., PlanetLab, NS2, Mininet) [63]. Additionally, one of the most famous applications in the industry is the Graphical Network Simulator 3 (GNS3) from Cisco [64], which provides the emulation of complex networks that are similar to the practical ones by using Cisco IOS, leading to a significant reduction of resource and time consumption in experimentation. Organizations use this software for designing network topology before they start to implement the deployment. Sometimes, network simulation can provide a platform for users to design and test their noval algorithms [65, 66, 67, 68, 69, 70, 71, 72], which is difficult when doing experiments uwith limited resources.

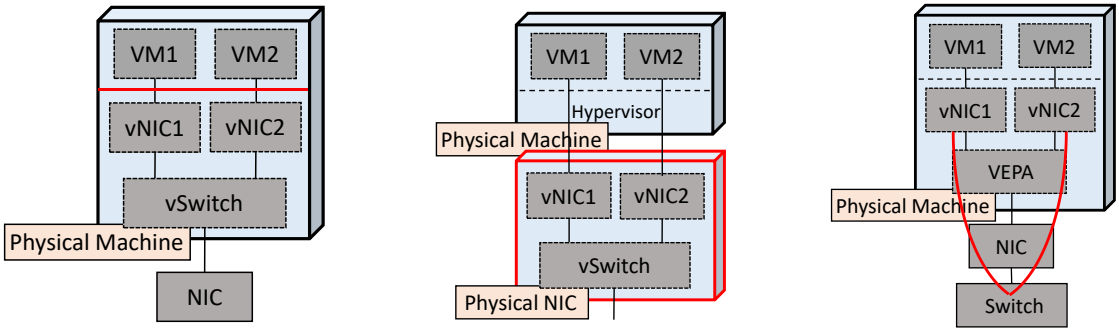
There are several benefits when employing NV in practice, such as flexible network composition, and efficient allocation of virtual network resources. With network virtualization, companies and researchers can make use of these advantages to simplify the tasks that schedule, scale and adjust workloads and resources to meet advancing computing demands [73]. This is also applied when researchers do experiments by using simulation testbed [74, 75, 76, 77, 78].

The most favorable use case in the industry is the virtualization of Network Interface Card (vNIC), which is usually implemented when there are multiple VMs deployed in only one physical carrier with one physical NIC [4]. Since each VM needs at least one vNIC, three kinds of solutions are provided to address this problem: 1) One way is to allow the hypervisor to create more vNICs than VMs, which communicate mutually through a vSwitch that connects to a physical NIC (see 2.7a). The advantages of this approach are transparent and straightforward, but there will be significant software overhead, and external devices do not easily manage it. 2) Another way is to virtualize multiple vNICs by leveraging physical NIC, which is applied by many NIC vendors (see 2.7b). In this method, the chip of physical NIC emulates various vNIC ports by using SR-IOV on the PCI bus. 3) The last one is to utilize a Virtual Ethernet Port Aggregator (VEPA) as virtual channels for inter-VM communication (see 2.7c), which lets an external physical switch handle the communication policies and transmit the network state to other VMs controlled by the same hypervisor [4].

2.2.2 Network Function Virtualization

Compared with NV, NFV refers to separate the network services from dedicated physical appliances and abstracts these functions as Virtual Network Functions (VNFs) running on industry-standardized hardware, e.g., Intel x86 kernel servers. Those VNFs can be implemented in VMs, serving as network applications that function identically to middleboxes. Thus, the profits produced by NFV are tremendous, which promises flexibility and scalability of network deployment, programmability of network functions, as well as dynamic resource allocation. As a result, the goal of "faster, cheaper and easier" will be reachable. One of the use cases of NFV is the application scenario for Customer Premises Equipment (CPE) [5, 79].

Figure 2.8 demonstrates that each CPE consists of network components such as modems, switches, routers, firewalls, NAT and DHCP servers in the traditional network. Since the



(a) (b) (c)

Figure 2.7: Different methods for NIC virtualization [4].

components are deployed straightly at the customer sites, the ISPs have to visit every single customer to transform the whole topology map if they intend to modify any of the components. Consequently, the duplicate operations cause the burden to both ISPs and customers.

In contrast, on the right side of Figure 2.8, it displays an explicit NFV model for CPE, which exhibits the simplicity of managing the network functions. The VNFs are located in a common area that could be a data center shared among distinct customer sites. If any change of network functions occurs, the updates will be applied to each of the customers immediately. For instance, if the ISP plans to transmit packets with a different public IP address, the NAT rules are only required to be adopted in the shared block at the ISP instead of being altered at each site, which applies for all customers. It potentially diminishes the operational cost in the more extensive scale network.

Regarding flexibility and scalability, the VNFs can be regarded as software applications, which are decoupled from the hardware equipment and can be easily installed in the VMs. Therefore, the physical location will no longer be the limitation for flexible installation of network services because both the hardware and software entities are mutually independent. In other words, NFV technology allows VNFs to be deployed in any position that network administrators want according to the accessible migration properties of VMs, but engineers usually prefer to aggregate these VNFs in the data center in order to achieve smooth management. It also provides the ability that allows distinct advancements between VNFs and commodity machines, which means network operators can upgrade either one while keeping the other one unchanged.

Regarding programmability, VNFs can be easily re-modified by developing the functionality based on customer needs since they are programmable. It helps to lessen the mismatch between the implementation of user requirements and future ideas for innovation. This reduces the extra cost of buying new devices, as well as to prolong the life cycle of network robustness. Without necessarily purchasing additional devices, NFV technology paves the way to solving the bloat of middleboxes in the current networks.

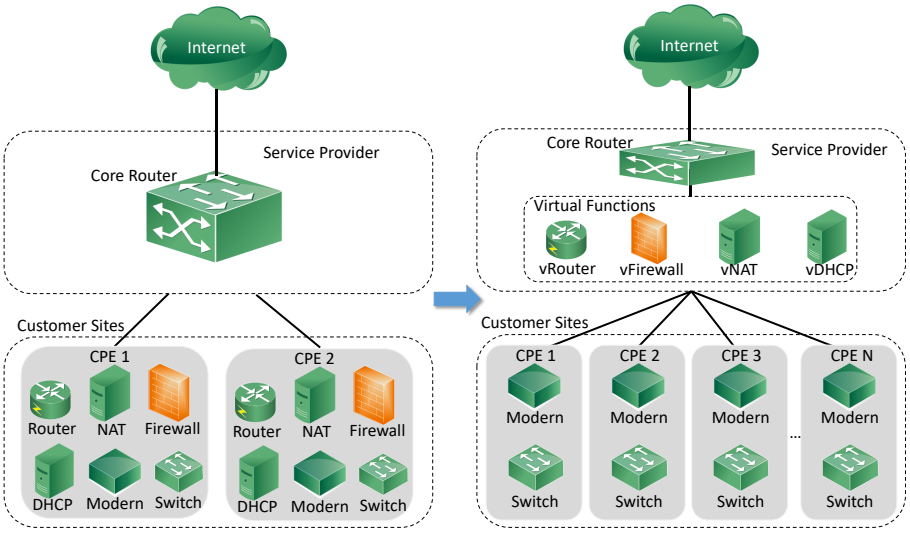


Figure 2.8: CPE scenario using NFV [5].

Regarding dynamic resource allocation, network services can be instantiated as software components associated with a collection of virtual resources. Thanks to the separation between the VNFs and the physical platforms, it provides the possibility that scales the working range of network services by dynamically assigning resources. For example, according to the actual demands for checking the state of application layer header, network functions such as the application layer firewalls require more computing resources. Subsequently, NFV can allocate more CPU capabilities for the firewall VMs through diminishing other computing resources reserved by the idle VMs, which provisions the fine resource balancing competence [5].

It is worthwhile to notice that introducing the idea that insulates network functions from hardware does not imply to deprecate the existing dedicated apparatuses instantly. On the contrary, NFV enables VNFs running on VMs to coexist with middleboxes, which will become a better solution to shift towards the NFV-based network, incrementally. Additionally, the hybrid scheme not only cuts down the budgets for bulk purchase but also offers a method that saves energy (e.g., no extra hardware that needs more power than VMs) to a certain degree.

According to the white paper [80], the NFV construction consists of NFV Infrastructure (NFVI), VNFs, and NFV Management and Orchestration (NFV MANO), which is shown in Figure 2.9.

- **NFVI.** Figure 2.9 exhibits that the NFVI, which provides the foundation for VNF deployments, is comprised of virtual and physical resources. In general, the physical resources, contain a batch of commercial machines and the networking links between these devices. In another word, the resources provide the computing abilities, storage, and network competencies to the users. Correspondingly, the virtual counterparts

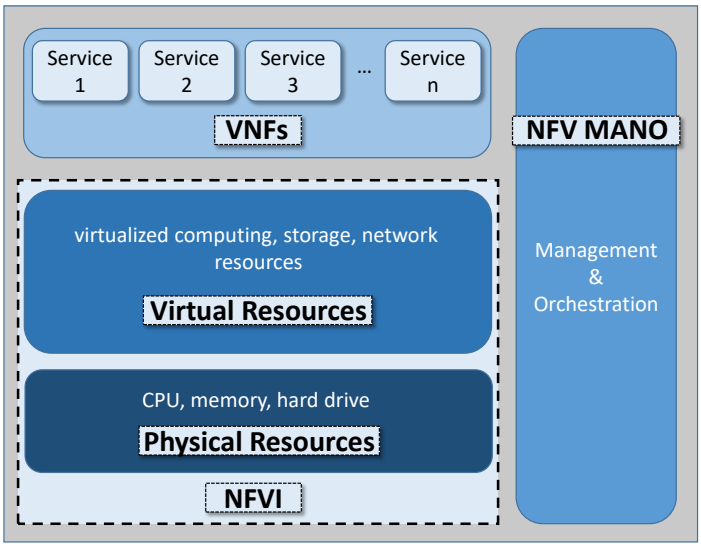


Figure 2.9: NFV architecture.

comprise the resources that are abstracted (i.e., logically decoupled) from the underlying apparatuses by using a virtualization layer that can be based on a hypervisor or a container. In addition, the virtual resources contain the virtual network topology (e.g., virtual links and nodes) [23].

For instance, the VMs in the DC serve as the computing and storage resources, while the connection between VMs can represent the virtual links that are logically isolated by the hypervisor in a host. Furthermore, a virtual node represents the software-based component deployed in a VM, with being associated with routing functionality. Thus, VNFs can be benefited from this structure, being implemented by leveraging the APIs provided by the infrastructure, which is applicable for VNFs to allocate both virtual and physical resources dynamically.

- **VNFs.** Network functions, which "can be implemented in a single operator network or interwork between different operator networks" [80], are a set of well-defined functional behaviors to support the higher-level services inside a network infrastructure. For example, the network functions in the enterprise network environment can involve various elements, such as DHCP servers that assign the IP addresses to different hosts, and firewalls that filter the unwanted or malevolent traffic flow. Hence, the aggregation of network functions becomes a network service offered by an ISP.

  Furthermore, a VNF is the abstraction of a network function in the case of NFV, which can be comprised of a single component, or multiple components. As a result, the VNF can be run in a single VM or be distributed in multiple VMs. In the latter case, every single component of the VNF is deployed in a distinct VM. Nevertheless, the whole process is transparent to the end-users, which means the performances between the services whether based on the network functions running on specific hardware or VMs are identical. This phenomenon indicates that the functionality of

the services does not depend on the counterpart of the ingredient network functions (could be VNFs).

- **NFV MANO.** Generally, an infrastructure without a management system could lead to low efficiency of resource utilization. To address this problem, the ETSI proposed the NFV MANO framework [80, 81], whose task is to provision governing abilities to the whole NFVI eco-system. The framework includes the configuration and lifecycle management (e.g., instantiation, update, scaling and termination) of VNFs, lifecycle management of virtual and physical resources during virtualization. Notably, this framework undertakes all the indispensable elements of virtualization-oriented management in NFVI, including fault management, configuration management, accounting management, performance management, and security management. Thus, it assists optimizing the administration procedure.

  Besides, the NFV MANO framework defines and uses the northbound APIs [81]. Those APIs not only hold for constructing communications between components in the NFVI but also support the coordination with other historical management entities such as Business Support System (BSS), and Operation Support System (OSS). Hence, the MANO framework allows the coexistence of managing VNFs and network functions running on inherited devices. Unfortunately, the compatibility with BSS/OSS is narrow in the NFV MANO, which does not support well or have the complete control of those external management systems. Thus, it could be arduous for the management system to stipulate its reaction to the heterogeneous network services containing both VNFs and conventional network functions. Since the standards of NFV architecture is still in its developing stage, many complementary details are strongly required.

2.3 Comparison between SDN and NFV

The concentrated points of SDN and NFV are different, where SDN aims at dividing the overall network control from the packet processing while NFV focuses on detaching network services from the particular hardware elements. However, they have many similarities in their purposes. For instance, they both seek for virtualization, where NFV intends to virtualize the network functions while SDN expects to virtualize the control functions [5]. Thus, it could be an interesting and promising direction to investigate combing SDN with NFV since they are highly complementary [82]. As shown in Figure 2.10, the SDN planes can be mapped to the parts in the NFV regarding to the same color (i.e., management plane against virtual network functions, control plane against orchestration, forwarding plane against physical & virtual resources).

Specifically, NFV technology can assist SDN to abstract the controlling and administration applications as VNFs to run on a VM. This behavior refers that the applications or services initially running on the SDN controller can be programmed as distinct VNFs running on the hypervisors. In reverse, SDN can play an essential role in accelerating deployment of NFV MANO system, which also has the competence to support multi-tenant

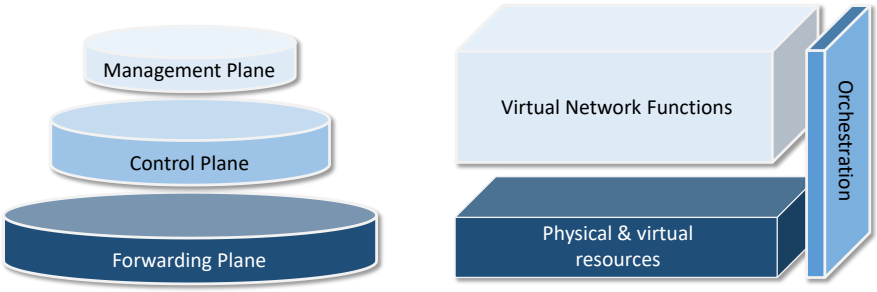


Figure 2.10: Mapping SDN to NFV construction.

NFVIs as it can enable the network virtualization. Besides, SDN can manage resources and functionality such as provisioning, the configuration of network connectivity, bandwidth, automation of operations, monitoring, security and policy control [83].

Moreover, SDN necessitates separating the data plane from the control plane, but NFV can be consistent with the existing network. The difference makes the requirements of network structure between SDN and NFV disparate. Therefore, it remains to put more efforts on their associate enhancement, where two fields need to be developed before enabling SDN in the NFV network: 1) One is to improve the southbound APIs that control a large number of communication protocols. 2) The other one is to strengthen the controller schemes.

In the first aspect, the original OF only works between control and forwarding layers though it is one of the most common protocols for connecting SDN controllers with SDN tenants. Also, the functionality of OF does not support application layer since it was initially designed for approaching protocols under transportation layer. Hence, it cannot offer operations to the network functions such as QoS, and data classification. Since most current network deployments of SDN depend on one single logically-centralized controller, this design produces poor scaling performance and impacts the reliability and the resiliency. Subsequently, it is urged to extend the OF protocol, so that it can support the application-layer functions in NFV networks and deploy SDN controllers in a distributed environment [84].

In the second aspect, it is desired to enhance the distribution management among the controllers when deploying multiple controllers in the NFV-enabled SDN environments. As a representative of SDN-enabled NFV architecture, OpenNF [85, 86] is designed to address the problem that additional control mechanisms are required when handling of the packet is redistributed across a series of network function instances. The proposed control plane of OpenNF allows massive flows to be reallocated to other network function instances, knowing and controlling the states of both internal network function and network forwarding. Also, the OpenNF Controller consists of NF state manager and flow manager, using an association of events and updates to make network configuration and policy decision.

Chapter 3

Related Work

This chapter reviews the related work of the proposed scheme in the thesis and splits them into two sections: network resilience and network prediction. Section 3.1 introduces and categories three kinds of resilient solutions in SDN, including flow detour, function immigration, and controlling function distribution. Section 3.2 presents two types of forecast methods, consisting of the statistical and machine learning approaches.

3.1 Network Resilience

In traditional networks, each router (or OSI layer-3 switch) stores the routing tables, which pre-program the failover paths. Since the routers hold their control logic, the node failure only impacts the particular router within the region. The corresponding backup paths can be activated when one router becomes faulty. Nonetheless, the situation is different in the SDN environments because the switches only contain forwarding functions and the controller takes charge of the routing management of the whole network.

In general, SDN remains at the initial and immature stage so that it could lead to several research directions including resiliency. At the beginning of SDN implementation, a control-centralized model is designed such that one single controller can be deployed to manage the entire network since the network scale is initially narrow in a campus. Consequently, it raises the rate of causing the problem of single point of failure, which means the whole network management will be inoperable if the controller goes down. In addition to controller failure, control channel failure is another issue of the reliability of control plane. Note that the reliability here represents the robustness of connection between controllers and switches [7].

To tackle a resilience issue, many kinds of literature proposed a variety of solutions. According to their functionalities, the following summarizes those methods as three types:

1. **Flow detour**. This method chooses an alternative path from the source switch to the controller by installing new flow entities when the original link fails.

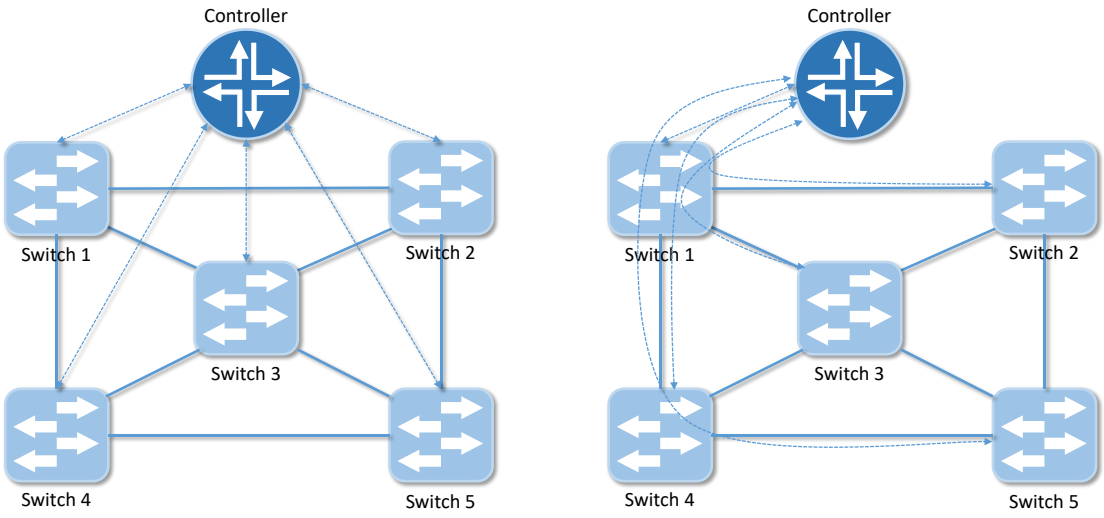
2. **Function immigration**. This method deploys controlling functions in the VMs, backing up the initial network state, which could recover the controller from a fine state when the failure occurs.

3. **Controlling function distribution**. This method employs multiple controllers, synchronizing the flow tables between controllers. When the master controller runs malfunction, the backup one will take charge of the network as soon as possible.

3.1.1 Flow Detour

To maintain the reliability of SDN, switches, and controllers are supposed to keep the connection even under unexpected link failure. Thus, SDN requires abundant links or nodes to support the backup. The controllers manage their switches via the connections called control channels, which consists of out-of-band connection and in-band connection.

Figure 3.1 illustrates the archetype of control channel. Because of the simplicity of implementation, most of the current researchers prefer to utilize the out-of-band connection to control the network, where this kind of connection needs a dedicated physical port to dispatch control packets. To save the resources, network operators would adopt in-band connection in the current network. Many proposals aimed at enhancing the performance of failure recovery by modifying or adding the items of the flow table.

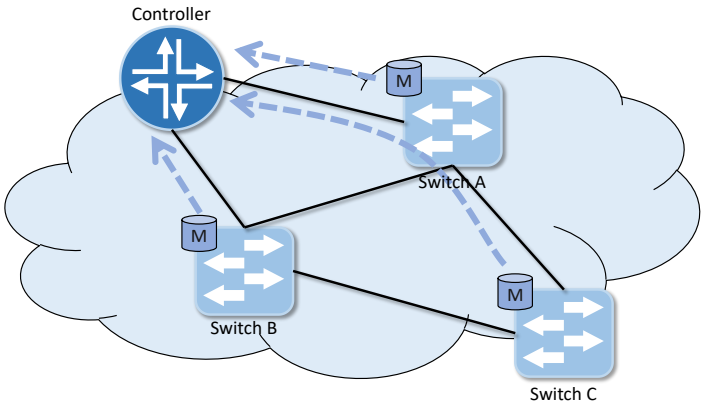


(a) Out-of-band connection (b) In-band connection

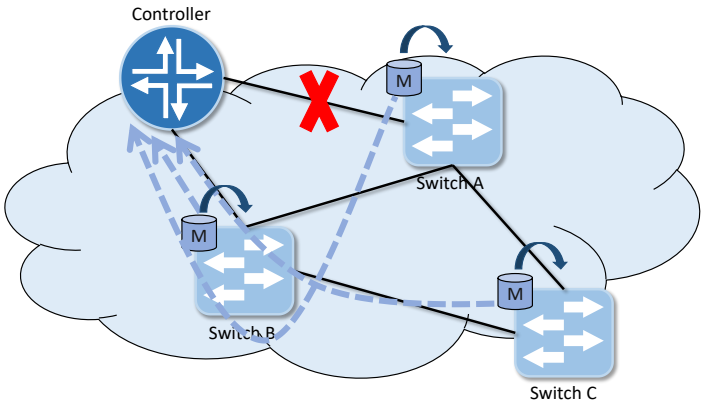
Figure 3.1: Redrawn types of control channel [6].

ResilientFlow

ResilientFlow [7] introduced a Control Channel Maintenance Module (CCMM) to allow all switches to detect channel failure and restore the control channel through a detour path by themselves. As shown in Figure 3.2a, a switch can be either setup to connect to a controller directly through the link for control channel only, or through another switch. For instance, the former way communicates through the path between the controller and Switch A or Switch B, and the latter way communicates through the path between the controller and Switch C via A or B. Note that, the modules are deployed into the switches mainly.



(a) Before the link failed



(b) After the link failed

Figure 3.2: Redrawn operation mode of ResilientFlow [7].

A CCMM monitors the link status of switches by leveraging heartbeat packets with the CCMM nearby. Meanwhile, the CCMM will swap its network topology information with other neighbors at a specified interval, including the information of controllers. Once the network status alters, the CCMM will advertise the changed topology maps to other peers.

Furthermore, the CCMM of a switch will recompute an alternative path to the controller when it detects the control channel failure. Then, the module installs new flow entries into the switches base on the chosen route.

ICT fast recovery

In order to reduce the interval of failure detection, which is usually too long for SDN to accomplish fast failure recovery, a monitoring cycle is suggested to explore the status of all links within the region, obtaining the information in an acceptable time [6]. The authors also proposed a protected In-band Control Tree (ICT) to protect the in-band control channel, where the ICT delivers control messages between the controller and switches.

In the typical case, the controller can gain the working state of each switch by receiving the Path Alive Monitoring (PAM) packets. When the link failure occurs, the detection process of the system will go into the procedure of Failure Location Identification (FLI), where the switch that receives an FLI packet will transmit the packet to the next-hop node through the tracking path and send a copy back to the controller. Consequently, the node that does not send back the FLI packet is next to the breakdown links.

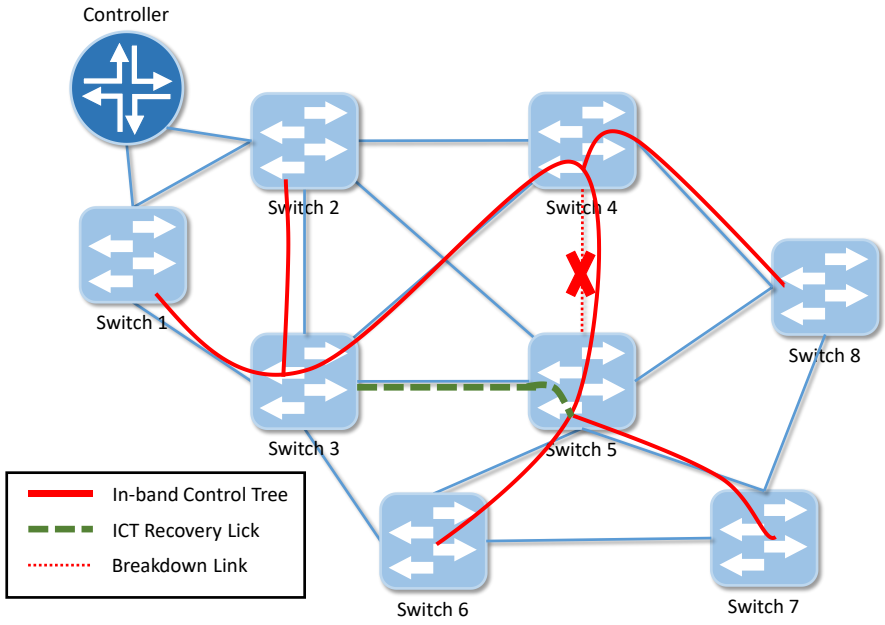


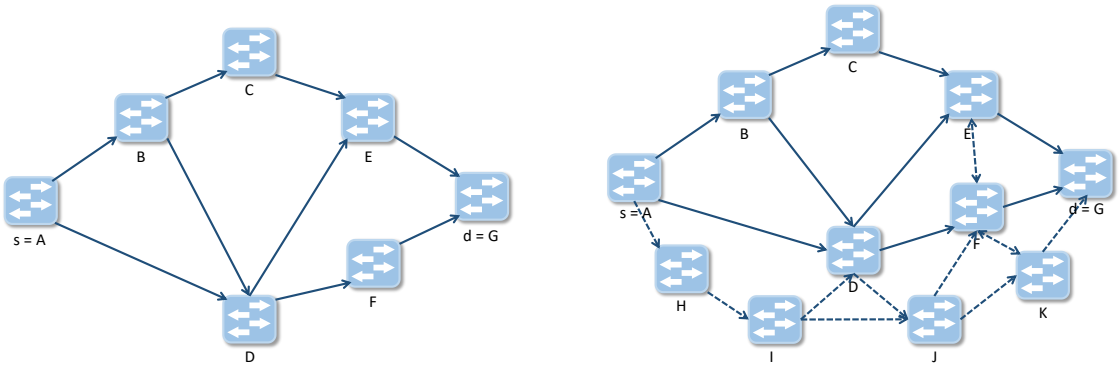
Figure 3.3: Redrawn example of ICT [6].

As shown in Figure 3.3, it shows a scenario of ICT process, where the failure link between Switch 4 and Switch 5 leads to a split of ICT into two sub-trees. In the proposed design, the backup route for each link in the ICT is pre-setup, where the controller resumes the two sub-trees by issuing modified flow entities to Switch 3 after the failure is detected

(see the green dashed line). Then, the ICT is recovered, and the back path is activated as well.

ARRP

To tackle the limitations of traditionally path-based routing schemes, presented an Adaptive Resilient Routing via Preorder (ARRP) is proposed to deploy additional functionality in switches to generate activation/deactivation tag announce link/node failure and recovery events [8]. In that case, the SDN controller can hold the entire view of the data plane since the network configuration is pre-installed. Being associated with Preordered Graph (PrOG), the proposed method can obtain latency-constrained optimal resiliency under arbitrary accidents, without invoking the controller for processing path recalculation.



(a) Primary PrOG (b) Backup PrOG

Figure 3.4: Redrawn preorder-enabled routing scheme with PrOGs [8].

Figure 3.4 depicts that ARRP scheme builds up a preorder map for packet-dispatching route according to the chosen routing cover, where the scheme is mapping a flow not only to a single path but also multiple paths. In contrast to conventionally path-based routing, ARRP leverages all relevant links with enumerating and specifying all possible paths. Also, the ARRP scheme dynamically transmits the packets through the paths that are included in the PrOG. Therefore, it conquers the limitations that traditional routing protocols may have, which could achieve the adaptively flexible routing and better QoS by using topological diversity.

BFD-ICN

Similar to the concept of ARRP, Vestin et al. presented a resilience-based framework for Industrial Control Networks (ICN) by utilizing Bidirectional Forwarding Detection (BFD) [87]. In addition, the framework configures the primary and backup routes in advance. In that case, the control latency can be minimized in a significant way, which could lead to a better performance guarantee after losses or failures occur.

In the proposal, a controller employs two techniques, including flexible packet duplication and fast failover. Links detect the failures and recover locally by using local BFD sessions. As a result, the working mechanism allows the recovery to react promptly.

3.1.2 Function Immigration

Another type of SDN resilience is to deploy the controller functions or resilience management system into clusters or data centers, which enables fast and flexible recovery. Moreover, this kind of resilient methods can scale up or down the utilization of control plane according to the network conditions.

MP-SDN

To protect the network from the malicious attacks, such as Distributed Denial of Service (DDoS), and worm propagations, Smith et al. proposed a policy-controlled Management Pattern (MP) [88]. The MP abstracted the resilience services as OF applications and described the deployment mechanisms to tackle the specific network threats.

The overall pattern pre-configures the rules that define the particular OF applications to solve the corresponding network events, which enables the reuse of diversified scenarios or different parts of the network. Furthermore, the MP-SDN method extended the PReSET [89] simulation platform to evaluate the performance of policy-based resilience strategies.

SPN OS

The SPN OS framework is based on a new controller platform called Software-Programmed Networking (SPN), which is evolved from SDN and uses the OF protocol, targeting at Object-Oriented Network Virtualization (OONV). The concept of Virtual Network Object (VNO) is introduced as an abstraction of both network topology and network. VNO contains all the network properties and services behaviors [9, 90]. The VNOs also provide the fine-grained control over features of the underlying physical network, such as bandwidth and backup paths. Additionally, the sharing policy and schedule could be written in the service pattern of VNO. Thus each VNO would utilize resources based on the defined patterns when being activated. The topology of each VNO can be generated from the pre-defined templates, such as “big switch”, fat tree, and ring topology.

SPN OS consists of a VNO Pool and an Arbiter as shown in Figure 3.5. All the VNOs are stored in the pool that uses efficient storage primitives while the logically-centralized Arbiter takes charge of those VNOs. Furthermore, the Arbiter gets involved in the tasks related to physical resource allocation for VNOs, with calculating and continuously updating mapping patterns regarding the requirement of the current network.

The main advantage of this framework is that it provides easy restoration and live migration since it can back up the entire network components including VNFs as a VNO

image. The concept of the VNO image is similar to the image of a VM. Therefore, network operators can easily manipulate the network deployment. However, this framework is still a prototype, where the authors do not give any information about the performance.

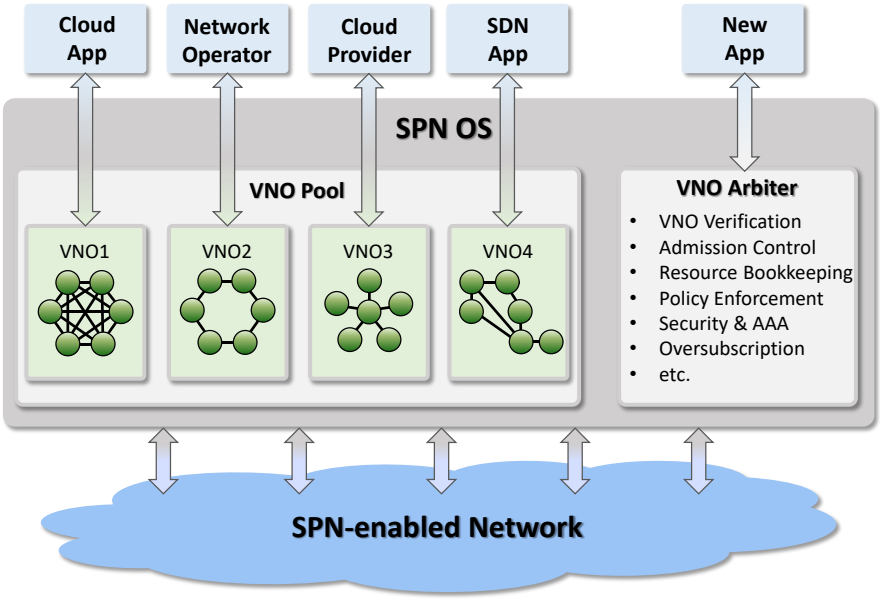


Figure 3.5: SPN OS framework [9].

VTN Orchestrator

In the conventional SDN network, when a new tenant network needs to be created, the tenant SDN controller implementation usually is manually installed and deployed on a dedicated machine, which leads to an extra cost for purchasing devices and a time-consuming process. Thus, authors in [10] proposed a hierarchical NFV-MANO architecture (see Figure 3.6), which virtualized the control functions and moved them into the cloud to provide dynamical controlling services when a new Virtual Tenant Network (VTN) is generated.

As shown in Figure 3.6, the authors deploy an NFV orchestrator to manage the whole virtual SDN networks and several VNF managers on top of the NFVI virtualization layer. The entire system consists of VNF managers and NFV orchestrator (vSDN manager and NO), global cloud and network orchestrator (GCNO), intra-domain cloud and network orchestrator (ICNO), multidomain network hypervisor (MNH) and multidomain SDN orchestrator (MSO).

From the top down, vSDN manager is proposed to manage the lifecycle of a VNF, requesting the creation, deletion, and configuration of VMs. NO is dominating the lifecycle of physical and virtual resources. GCNO is responsible for the global administration of this hierarchical system, which enables the incorporation of heterogeneous transport networks and implements centralized control (i.e., end-to-end services between inter-domain clouds).

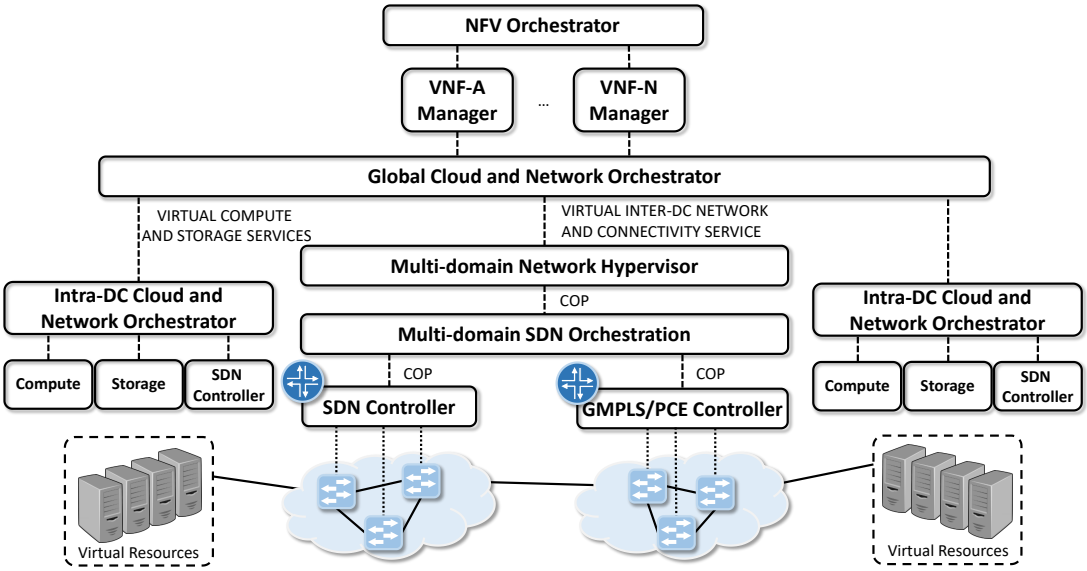


Figure 3.6: Multiple layer NFV-MANO architecture [10].

ICNO is in charge of the elements within the same cloud domain, which concerns computing services (e.g., VM instance management), image services (e.g., storage of images) and networking services (e.g., network connectivity within a region). MNH can remotely control the network connections by abstracting all the physical network resources into virtual ones, defining the logical topology through network slicing technique. MSO proposed in that paper provides a unified common northbound (NB) API, which communicates by the Control Orchestration Protocol (COP), allowing simplification and optimization related to scalability and compatibility between various modules in SDN.

VON-CSC

[33] presented a flexible approach to enhance the performance of NV in the complex SDN orchestration system above. This approach implements on-demand Virtual Optical Networks (VONs) by combining SDN, NV, and NFV, where VNO entities are controlled by the independent Customer SDN Controller (CSC). The authors deploy the controlling functions into the cloud to achieve SDN resiliency by VM migration. Regularly, the system saves the initially normal state of the controlling VM and the logical topology and resumes them when the error occurs.

Moreover, the experimental results illustrate the possibility of simplifying network operations, in which the deployment time for a vSDN controller is 69 seconds, and for necessary network connection is 4 seconds. This method shows the advantages of cooperation among SDN, NFV, and cloud, including enhancing the network reliability. Thus, this study starts up a promising research direction of engaging virtualization technique to improve the SDN resiliency.

3.1.3 Controlling Function Distribution

Besides the two kinds mentioned above of resilient methods, there is another prevalent type of solution being employed to advance SDN network dependability. The solution utilizes multiple controllers deployed within the same domain or in different domains, which is the universal solution to address the resilient issues in SDN. Also, challenges such as state synchronization [91, 92, 93], data distribution management [94, 95] and data propagation [96, 97, 98], generally exist when dealing with distribution schemes.

CPRecovery

In the early development of OF protocol, the network paralysis may occur in SDN due to its control-centralized property. Consequently, a proposal that the OF controllers can configure *master-slave* mode has been presented to improve the resiliency of the control plane since the OF protocol version 1.2 [25].

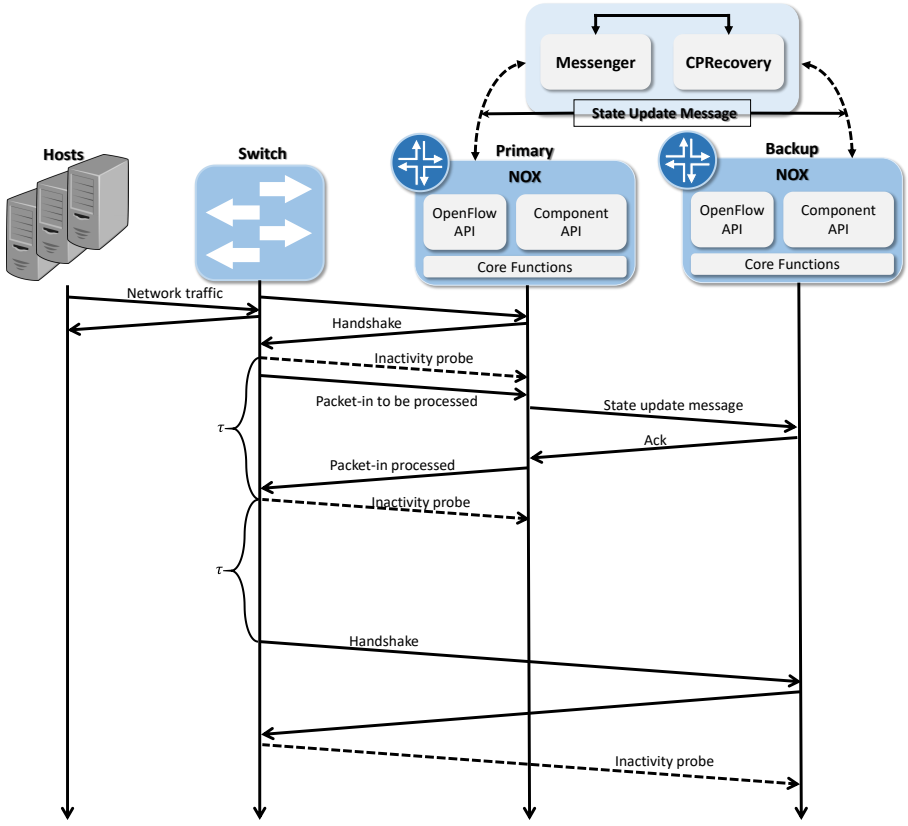


Figure 3.7: Working mechanism of CPRecovery component [11].

Although the OF protocol allows setting up more than one backup controller in case of network failure, it does not facilitate the collaboration between the primary and candidate control planes. As a result, the limitation leads to a dilemma that network administrators

have to re-configure to the candidates when the functions of prime controller collapse because of the loss of network configuration. The CPRecoery component [11] based on "primary-backup" protocol was presented to supplement the deficiency of OF protocol. It offers a method of seamless communication between the dominant controller and the backup ones, which keeps the consistency with the failure-free state of the last network.

As illustrated in Figure 3.7, the working mechanism of CPRecovery component is divided into two phases: replication phase and recovery phase. In the replication phase, the primary controller, whose *isPrimary* flag sets to be true, works as what the normal OF controller does, with receiving and processing the *packet-in* messages in the source table. Meanwhile, the secondary controllers will ignore the request. Besides, the CPRecovery component keeps synchronizing the flow entries of both types of control planes by the *messenger* module. After an interval τ, the switch assumes that the controller crashes if it does not receive the response from the controller to its *inactivity probe* packet. In the recovery phase, the CPRecovery component enables the switches to search the secondary controller previously defined in its candidate list during the first stage when the failure is detected. In the next step, the secondary controller starts to take over the entire network as the primary through modifying the state flag *isPrimary* to true. At the same time, the current dominant controller keeps sending *state update* messages to the previous primary controller to let it stay as a backup in case of its functionality resumes.

DCP

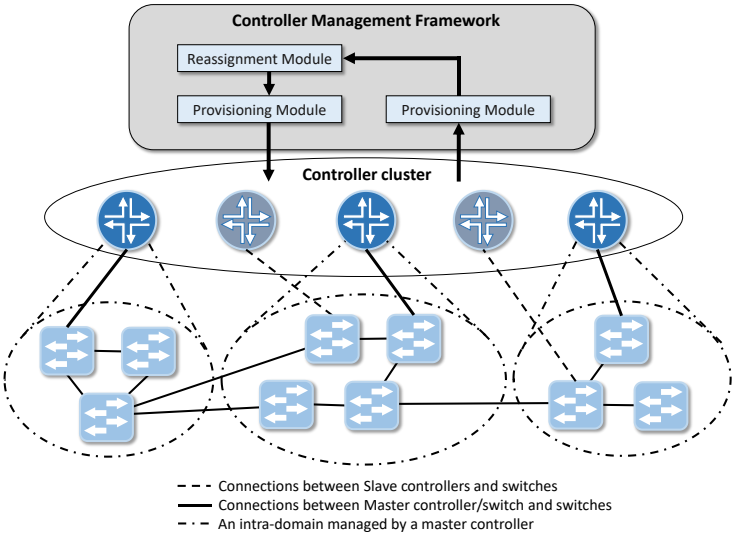


Figure 3.8: DCP system architecture [12].

In a large-scale WAN deployment, the placement of an SDN controller cannot always be optimal that guarantees the latencies between the controller and the switches are at the minimum. Additionally, it is difficult to manage such wide range of network by using a

single controller, which may limit the resource capacity and service performance. However, the proposal of deploying multiple controllers that aims at addressing this concern is likely to bring a new issue called Dynamic Controller Provisioning Problem (DCPP), whose solution is to find out the trade-off between the efficiency of flow setup and the management of controllers. In another word, the DCP methods [12] are proposed to deploy multiple controllers dynamically according to the network requirement, whose goal is to minimize the management overhead of multiple controllers while providing sufficient QoS in SDN.

The DCP management framework consists of monitoring, reassignment and provisioning modules, as illustrated in Figure 3.8. Specifically, the monitoring module manages the heartbeat messages between the controllers, which ensures the high availability of control functions. The reassignment module decides to recycle the idle controllers and designates a new controller to share the massive loads or to replace the one torn down, which usually depends on the criteria determined by the monitoring module. The provisioning module provides the controllers according to the orders from the reassignment module.

AED

Chen et al. formulate the controller selection problem as an optimization problem [13]. To maximize the utilization of controlling function, the authors presented an Adaptive Elastic Distributed (AED) SDN architecture by utilizing the greedy algorithms. The architecture migrates the switches to the new control relying on a threshold band. For load balancing, AED will migrate the switches to the existing active controllers and deactivate the original controller to save the energy, when the entire network load reaches the bottom of given threshold band. In the meanwhile, the control pool also reassigns parts of switches to a new inactive controller when the total loads of the controllers exceed the ceiling threshold.

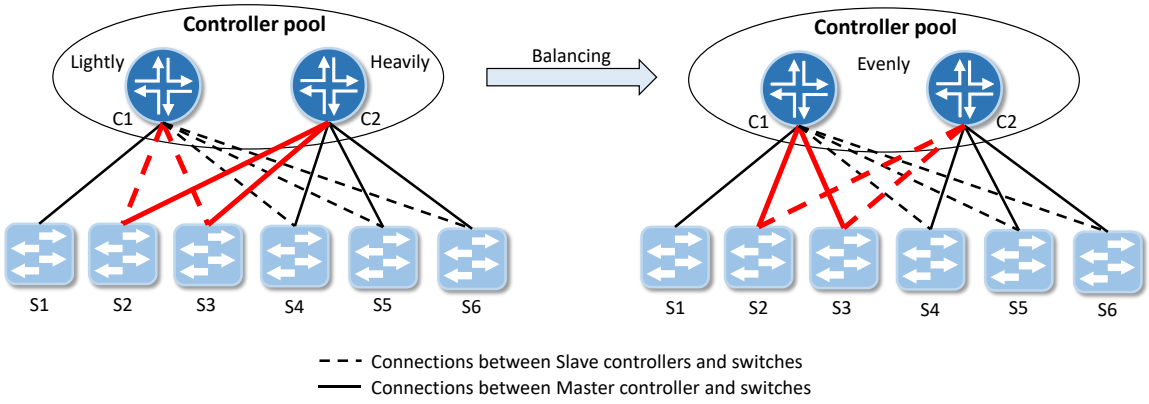


Figure 3.9: AED balances loads among controllers [13].

Figure 3.9 demonstrates the process of balancing control loads among the controllers when the loads reach the higher threshold calculated the AED method, where there are two controllers (i.e., C1 and C2) in the controller pool. The controller C1 is the master

controller of switch S1 and is also the slave controller of switch S2 to S6. By contrast, the controller C2 is the master controller that manages the switches from S2 to S6. Initially, the controllers obtain uneven controlling loads due to the dynamic change of network, where C2 handles heavier loads of 5 switches than C1 that deals with only one switch. After checking the feasible less-load controller, AED balancing algorithm starts to migrate the switches to the controller, where the administrator of switches S2 and S3 is shifted from C2 to C1, thereby the controlling loads are even again.

When the average network load is under the lower threshold, the AED shrinking algorithm begins to save the energy by reducing the size of controller pool, which transfers all the switches to the slave controller and deactivates the original master one. Figure 3.10 depicts the AED approach inspects the least-load controllers, analyzing if the slave counterpart can take over all the switches, where C1 is the backup controller of switches S4, S5 and S6, while C2 is the master. Furthermore, AED reassigns all the switches to C1 and inactivates C2, which shrinks the controller pool.

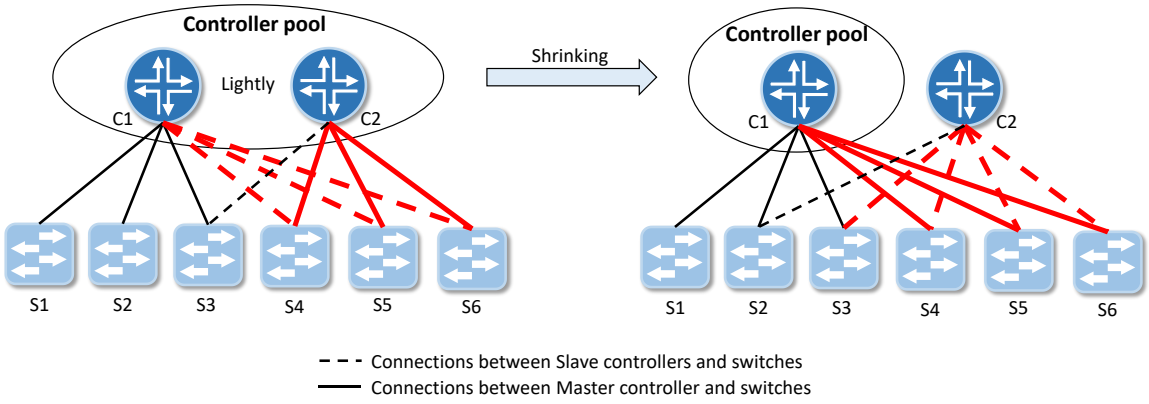


Figure 3.10: AED shrinks the controller pool [13].

DISCO

Many state-of-the-art models that mainly focus on deploying distributed controllers into data centers may face a problem that spawns and deals with a bulk of synchronizing message. To address this issue, authors in [31] proposed DIstributed SDN COntrol (DISCO) plane on top of the Floodlight controller, which consists of the intra-domain and the inter-domain portions. To summarize, DISCO allocates a controller taking charge of a small SDN domain, while supporting the lightweight management for inter-domain controllers.

Regarding intra-domain functionalities, it assists the controller to compute the prioritization of flows, which allows redirecting the traffic when the network issues occur. Besides, the system stores all the information on intra-domain and inter-domain in an extended database. Regarding the inter-domain functionalities, DISCO contains two major components - a messenger and agents, where the messenger discovers the neighbor counterpart

and maintains the communication channels, and the agents swap knowledge with other controllers beyond domain.

DFMCC

Follow Me Cloud (FMC) [99] was presented to provide smoothly continuous migration of services according to the mobility of users. However, there remain some limitations in the aspect of scalability when applying SDN paradigm to the traditional FMC implementations, since the number and location of FMC controllers are static. An elastic method for FMC-based systems is proposed for distributing the SDN control place on a two-layer architecture, which consists of a global controller (G-FMCC) and multiple local controllers (L-FMCCs) [28].

Explicitly, the first layer of Distributed Follow Me Cloud Controller (DFMCC) is the G-FMCC that is a constant active controller managing the modification of OF Rules for seamless service migration. The second layer of the proposed system is the L-FMCCs, which are dynamically provided and allocated according to the network requirements. To improve the flexibility of the system, authors extend the on-demand ability of L-FMCCs by leveraging NFV technique, which runs them on VMs.

To maintain the mapping information between domains and controllers, DFMCC uses an external Inter-Domain Mobility Database (IDMD), whose contents are updated by the G-FMCC according to the deployment status of L-FMCCs. Also, the Decision Making Application Module (DMAM) deployed in the system is used for deciding the migration occasion of services.

IRIS-CoMan

Since very few studies on management plane of SDN are investigated, there are a series of problems being introduced when combining diversified and heterogeneous management channels, which includes the problems concerning scalability, availability, and inaccurate and unreliable management.

To alleviate the dilemmas above, the IRIS-controller [14] as shown in Figure 3.11 is designed to administrate large-scale SDN networks by composing a cluster of OF-based controllers, which enhances the controller capacity and High Availability (HA). Additionally, the virtualized controllers are supported by an IRIS controller middleware, namely High Scalability and Availability (HiSA), which serves as a globally higher-level controller. This middleware provides functionalities such as load balancing, IRIS controller failure detection, failure failover, and switch migration.

Besides, a hierarchically recursive network abstraction mechanism is used to assist the cooperation between various SDN domains. That work proposed a Software-Defined Virtual Monitoring Function (SuVMF) architecture to collect, to analyze and to filter the incoming packets. When monitoring the network traffic, the detecting function of SuVMF can easily distinguish abnormal network events, which lessens the overhead of controller handling many status change messages.

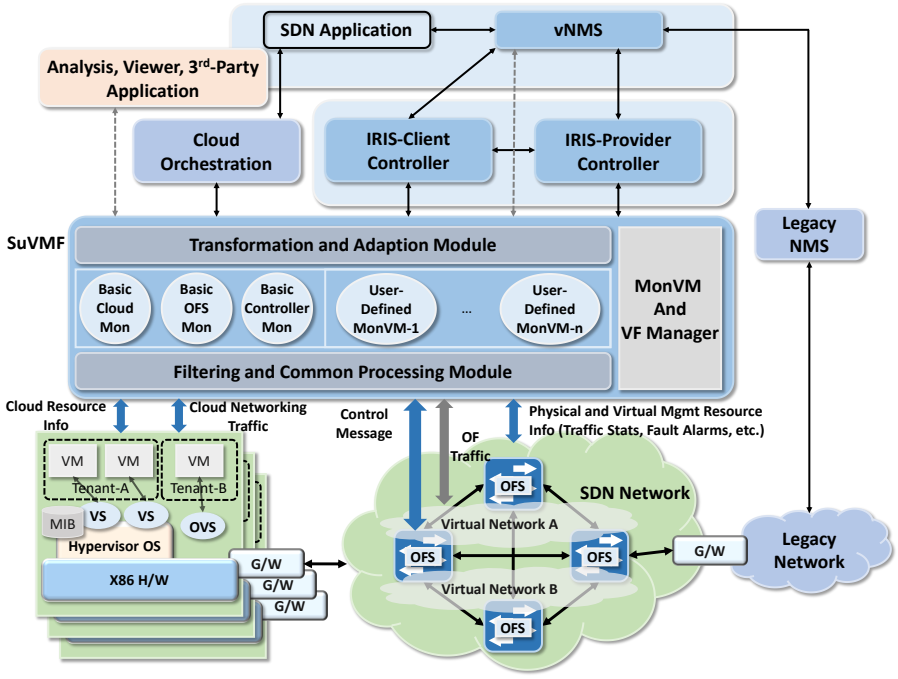


Figure 3.11: IRIS-CoMan architecture [14].

The IRIS controller, the IRIS-SuVMF and the virtual Network Management System (vNMS) compose the proposed efficient SDN orchestration system called IRIS-CoMan, where SuVMF works as a management middlebox for the controllers, vNMS, and the rest physical networks, providing diverse SB APIs for communication. Thereby, the system supplies an efficient way to detect the errors by the monitoring function and to recover the network failures by the vNMS.

STN

The authors of [100] formulate the Consistent Policy Composition (CPC) problem, introducing the interaction model between the data plane and multiple control planes. An approach called Software Transactional Networking (STN) associated with policy updates was introduced to capture the correct sequential composition, in order to solve the CPC problem.

In the framework, two protocols were presented, consisting of a FixTag protocol and a ReuseTag one. Specifically, the FixTag algorithm enables the controllers to employ the updates on the forwarding plane straightly, which also addresses the conflicts when setting up the new events. The ReuseTag algorithm identifies a general sequence of policy events that are most possibly installed. According to this initial study on a distributed and fault-tolerant, the model contributed to facilitate the design and operation of a robust SDN control plane.

OptimalFlow

[101] developed a hierarchical SDN control plane named OptimalFlow for Industrial Control Systems (ICS) by leveraging the solutions to the Integer Linear Programming (ILP) optimization problem. The object of this method is to lessen the issues such as failure and disruptive cyber attacks, whose OptimalFlow controller connects to the second-layer of OF-based controllers and renovates the network topology according to the ILP resolutions. Those controllers in the lower layer inspect and control a set of emulated or actual switches within the single SDN domain.

Additionally, this architecture further provides two algorithms to diminish the influence of renewing events on ICS flows, which includes Flow Migration Reduction (FMR) and Dependency Graph Construction (DGC). The FMR ensures the flows only being affected by the significant disturbance. Meanwhile, the DGC prevents link congestion through assembling a dependency graph for network updates.

The OptimalFlow controller can combine or split the SDN domains by reassigning the data planes to the sub-controllers, with harmonizing routing decisions. Those decisions consider the properties regarding QoS, security, and reliability, which primarily targets the large-scale network regions.

3.1.4 Summary

Among the categories of resilience-oriented methods mentioned in previous subsections, the one employing multiple controllers is the most predominant in the industry. Other related distributed algorithms or protocols can be found in [102, 103, 104, 105, 106, 107, 108, 109]. However, the issues that this kind of methods may face will be discussed in Subsection 4.1.1. This subsection summarizes some resilience approaches above, which are compared in Table 3.1, depicting the aspects of control channel alteration, hierarchical architecture, virtualized, distributed ability, and some key features.

The control channel alteration stands for that the type of control channel changed to recover the failures, excluding the case that only the destination IP of the controller has changed without altering the forwarding scheme. The hierarchical architecture means there exists a higher level of the management framework for scheduling the controllers. The virtualized property denotes whether the SDN paradigm of the framework has been virtualized and deployed into the cluster or data center. The distributed feature represents the proposed model has orchestrated multiple controllers for network resilience purpose.

Table 3.1: Comparison of Various Resilient Methods in NFV-Supported SDN

	Features	**Control Channel Alteration**	**Virtualized**	**Hierarchical**	**Distributed**	**Year**
ResilientFlow [7]	Detects channel failure and restores the control channel	Yes	No	No	No	2015
ICT [6]	Implements fast recovery by using rebuilding in-band control tree	Yes	No	No	No	2015
ARRP [8]	Redirects the flow via preorder map	Yes	No	No	No	2016
BFD-ICN [87]	Pre-configures the primary and backup routes	Yes	No	No	No	2015
MP-SDN [88]	Pre-configures management patterns to implement network reliability	No	Yes	No	No	2014
SPN OS [9]	Provides the easy restoration and live migration by using VNO image	No	Yes	Yes	No	2015
VTN Orc. [10]	Unifies a NB API for simplifying communication	No	Yes	Yes	Yes	2015
VON-CSC [33]	Flexibly deploys on-demand VONs, offering recovery by VM migration	No	Yes	Yes	Yes	2016
CPRecovery [11]	Allows seamless replication of state between the primary and backup controllers	No	No	No	Yes	2012
DCP [12]	Dynamically assigns controlling resources based on the network requirements	No	No	Yes	Yes	2013
DISCO [31]	Allows a controller administrates the intra-domain SDN while providing lightweight management for inter-domain controllers	No	No	Yes	Yes	2014
AED [13]	Dynamically reassigning the switches to the controllers in order to balance the controlling loads	No	No	No	Yes	2015
DFMCC [28]	Installs external interdomain database	No	Yes	Yes	Yes	2015
IRIS-CoMan [14]	Implements accessible detection for abnormal events, providing high scalability and availability	No	Yes	Yes	Yes	2015
STN [100]	Formulates CPC problem	No	No	No	Yes	2015
OptimalFlow [101]	Harmonizes routing decisions in large-scale networks associated with ILP solutions	No	No	Yes	Yes	2016

3.2 Network Prediction

Prediction is not an novel idea in recent academia, which has actually been investigated in many areas for years, such as marketing in industry [110, 111], price in finance [112], energy consumption in power systems [113], traffic flow in Intelligent Transportation System (ITS) [114], etc. According to previous literature that studied the characteristics of Ethernet LAN Traffic, the researchers found that the nature of network traffic owns *self-similarity* and *long-range dependence* [115, 116]. Especially, the *self-similarity* property enables network traffic prediction, which facilitates the forecast model to be built more easily. Also, network prediciton enables to build up a baseline of normal network traffic, which can be used to detect abnormal traffic flow beforehand [117, 118, 119, 120], most of which are agent-based [121, 122]. This operation can improve the reliability of network, enhancing the fault-tolerant capacity [123, 124, 125, 126]. Security issues can also be alleviated when using a trust-based strategies [127, 128, 129, 130], which create an accurate prediction model.

The methods used for prediction can be mainly categorized as statistical modeling (i.e., model-driven), and Machine Learning (ML) (i.e., data-driven). This section reviews most of the prediction methods in both aspects, summarizing their critical attributes respectively.

3.2.1 Model-Driven Methods

This subsection discusses the existing literature by diving them into ARIMA-based methods and other modeling ones. Currently, most studies prefer to use ARIMA to forecast volatile but highly dependent data while leveraging SARIMA to predict seasonal observations. In general, the model-driven methods construct a forecast model according to the off-line historical data, which are sensitive to the selected time series and are especially applicable to the cases with distinct characters [131]. However, these generalization models are complex linear processes, which require sufficient amount of datasets to estimate the parameters of forecasting equations. Even though the most proper linear coefficients are found, it does not guarantee that results lead to accurate prediction results.

ARIMA-based Methods

ARIMA [132] model was presented to explain the relationship behind time series, which has been ordinarily applied in the prediction of time series and proven that can achieve a decent precision [133]. Based on this classical modeling approach, numerous variants are proposed. Seasonal ARIMA (SARIMA) [134] was developed from ARIMA, which can reveal the seasonality of time series and provide more accurate prediction results. The details of ARIMA and SARIMA will be discussed in Section 5.3.

Fractal ARIMA (FARIMA) [135] was proposed to solve the problem that the original ARIMA method cannot measure the short-range dependence in traffic trace, which also simplified and accelerated the fitting process by estimating the Hurst parameter H [115] to get differencing level $d = H - 0.5$.

Subset ARIMA [136] has the similar model architecture of ARIMA but with fewer parameters, which only considers specific nonzero coefficients of the sequences P and Q. Compared to the traditionally full-sized ARIMA model, the subset ARIMA lessens the number of parameter sequence to reduce the possibility of overfitting.

Based on FARIMA, Sliding FARIMA (SFARIMA) [137] was introduced to alleviate the time lag of FARIMA by increasingly iterating the sliding interval M when the mean square error grows, where the maximum value of M is the one-third length of the observations. Through multiple iterations, SFARIMA improves the performance in terms of describing the self-similar property of network traffic, which show its superiority of owning both short and long-range dependence characteristics.

Other Modeling Methods

It is not realistic to apply Fourier spectral analysis for nonlinear and non-stationary time series directly. Additionally, the Intrinsic Mode Functions (IMFs) associated with the Hilbert transform can generate multiple instantaneous frequencies that identify distinct groups of data from the embedded structures. Hence, Empirical Mode Decomposition (EMD) [138] was introduced to preprocess any convoluted datasets through separating them into a finite and small number of IMFs, which can thus adaptively analyze and model time series.

However, conventional EMD may encounter two issues. One occurs when time series is composed of white noise that has been distributed uniformly through the entire sequence, which forces the Fourier spectra of different IMFs to a unified shape. The other one happens when the data contains impurity besides noise, which causes the mode mixing problem that IMF consists of oscillations instead of having any physical meaning. Ensemble EMD (EEMD) [139] was developed to allow the mean IMF to stay within the original dyadic filter windows, diminishing the two preceding situations by repeatedly adding different white noise series into the observations and decomposing the resulting data into various IMFs.

Katsaros et al. in [140] suggested predicting the wireless network by using Markov Chains (MC), whose families show advantages on dealing with location and request prediction due to their generality. In addition, MC relies on the short memory principle, which can accurately forecast the human behavior that searches information via the host spots when traveling.

Wavelet Transform (WT) was presented in [141], which decomposes time series into different scales of a period at first, predicts the network traffic with the corresponding model in parallel at the next, and re-constructs the predicted results with wavelet method at the end. Consequently, it allows determining the correlation of network traffic better, which is hard to observe in the original historical data because of the mutual impacts of complex network protocols.

3.2.2 Data-Driven Methods

This subsection categorizes similar works into ANN-based and other ML methods. Because of the efficiency of the ANN families in processing the non-linear property of time series, it is broadly deployed in various fields, including economic, environment, traffic flow, and network traffic [133]. Most experiments show that ANN-based methods are superior to the ARIMA-based counterparts, gaining better match with the historical observations. Nonetheless, overfitting and overtraining are the major concerns to the ANN families [142, 143], which may degrade the forecasting performance.

ANN-based Methods

Artificial Neural Network (ANN) is inspired by the working mechanism of neurons in the human brain, which possesses abilities such as memorizing, pattern recognition, and forgetting. A typical 4-layer structure of the neural network is illustrated in Figure 3.12, which consists of an input/output layer respectively, and two hidden layers. The figure demonstrates that the architecture of NN may vary, depending on the number of input nodes (m), hidden nodes (i, j), output nodes (n) and the number of layers. It is common that the ANN-based variants are derived from this basic architecture of NN with more or less hidden layers, whose modifications mostly aim at the NN cell (i.e., neuron).

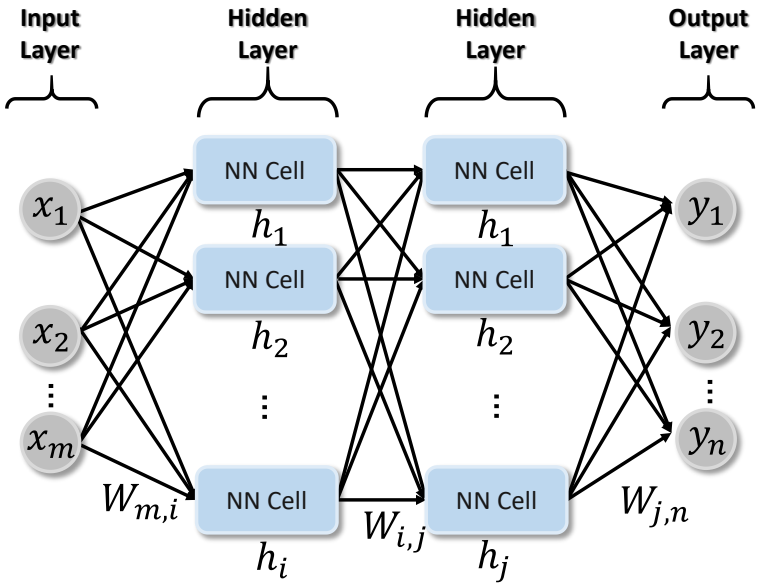


Figure 3.12: A typical 4-layer structure of neural network.

An advanced Elman Neural Network (ENN) [144] was proposed to address the limitation that the standard Elman network can only recognize a linear dynamic system, which leads to an unstable learning process of weights between units. This ENN fully leverages

the undertake layer node output in order to enhance the performance of Elman approximation, which adds and combines an extra undertake layer with the input units, so that two undertake layers formed by the feedback loop can lessen the learning error of weights.

Dynamic Tensor Completion (DTC) [145] was presented to tackle the shortcomings of conventional prediction methods that treat the time series as a one-dimensional array, which does not fully use the spatial correlation of data. By utilizing the multi-dimensional array, which serves as a tensor pattern, DTC can extract the underlying features from the traffic data. In the meanwhile, DTC constructs a two-step prediction framework: 1) The first step is to implement a rough prediction by using the intra-day trend of traffic flow, and then constructs a tensor pattern with the time series. 2) The second step is to derive and maintain the construction of the pattern. Thus, the accuracy of prediction can be enhanced dramatically.

Nonlinear AutoRegressive with eXogenous inputs (NARX) [133, 146] was applied to forecast nonlinear time series because it restricted the feedback loop of delayed outputs that impact performance, which ensures the computational efficiency to be comparable with the Turing machine in theory and the effectiveness in solving the gradient-descent learning in practice.

Since Extreme Learning Machine (ELM) solved the parameters training problem of NN with a linear equation, it accelerated the processing time of training. Genetic Algorithm (GA) is introduced to optimize the parameters of ELM, where the ELM designated an error threshold ε in case the GA optimization interferes too often [147]. When the sum of absolute value of the observed sequence minus the predicted sequence reaches the threshold, the GA will repeat the process optimization until the condition under the threshold is fulfilled.

Long Short-Term Memory (LSTM) [114, 148] and Gated Recurrent Units (GRU) [149] are two most popular branches of recurrent NN, which are famous for addressing the gradient vanishing problem with additional forget units. The details about the cell structures of LSTM and GRU will be discussed in Section 5.3.

Other ML Methods

Being different from other analysis of traffic flow, Sun et al. assumed the given traffic flows at a road link are independent of those at its adjacent links [150]. In general, the conditional independence relationship in a Bayesian network, which can be stated as that "a node is independent of its ancestors given its parents, where the ancestor/parent relationship is with respect to some fixed topological ordering of the nodes" [151], facilitates estimating the parameters and the forecast variables. In that case, the Bayesian network can be utilized to provide a precise prediction with less training data because the scale of the prediction model has been dwindling in this assumption.

K-Nearest Neighbor (KNN) has been widely employed in prediction and classification because of its nonparametric property, which stores historical observations into the historical database instead of directly putting into the training process, and measures the

similarity between samples and real-time input data [152, 153]. At the first step, the advanced KNN checks the duplication, reasonable range, consistency, and loss of both historical and real-time data. At the next step, it obtains the match state vector by comparing the instant data and the historical data in the database, then calculating their similarity by the Euclidean distance [154]. During each procedure of receiving a state vector, KNN will search for k nearest vectors in the database. Finally, the predicted value is the average of k nearest neighbors.

Being derived from Support Vector Machine (SVM), which constructs a hyperplane as the decision boundary that divides the datasets into two clusters, Support Vector Regression (SVR) [155] can be used in the regression analysis in addition to the functionalities of SVM. Furthermore, the SVR can gain the global optimum solution so that it can transform the nonlinear problem into the linear one by a radial basis kernel function [156]. In the end, the SVR implements prediction by solving the linear problem.

3.2.3 Hybrid Methods

EMD-MMP

Commonly, it is not sufficient to predict the network traffic through only leveraging a single model due to the nonlinear and irregular characteristics of the network flows. Each solo method has its expertise, for example, ARIMA is good at forecasting linear time series, and SVR specializes in classifying the dataset with strong autocorrelation. Thus, Dai et al. proposed applying multiple prediction models after decomposing the time series of network traffic through EMD [157]. Moreover, the EMD-MMP separates the new components into the high and low-frequency segments, which will be modeled by ARMA and SVR respectively. Eventually, the EMD-MMP sums up the resulting values as the final prediction.

KARIMA

Kohonen ARIMA (KARIMA) introduced in [158] has allowed predicting traffic flow by associating ARIMA model with a neural network called Kohonen Self-Organizing Map (SOM). The proposed technique splits the forecasting task to two steps: (i) using SOM in hexagonal layout as an initial classifier to differ individual classes and (ii) training a correspondingly independent ARIMA model for each defined class. This model initially iterates the p values from 1 to 5 and q values from 1 to 3, then finding the most fitting parameters for each class of observations (i.e., different road segments). Finally, KARIMA captures the feature of given time series and decides to forecast by using the ARIMA model in the corresponding class.

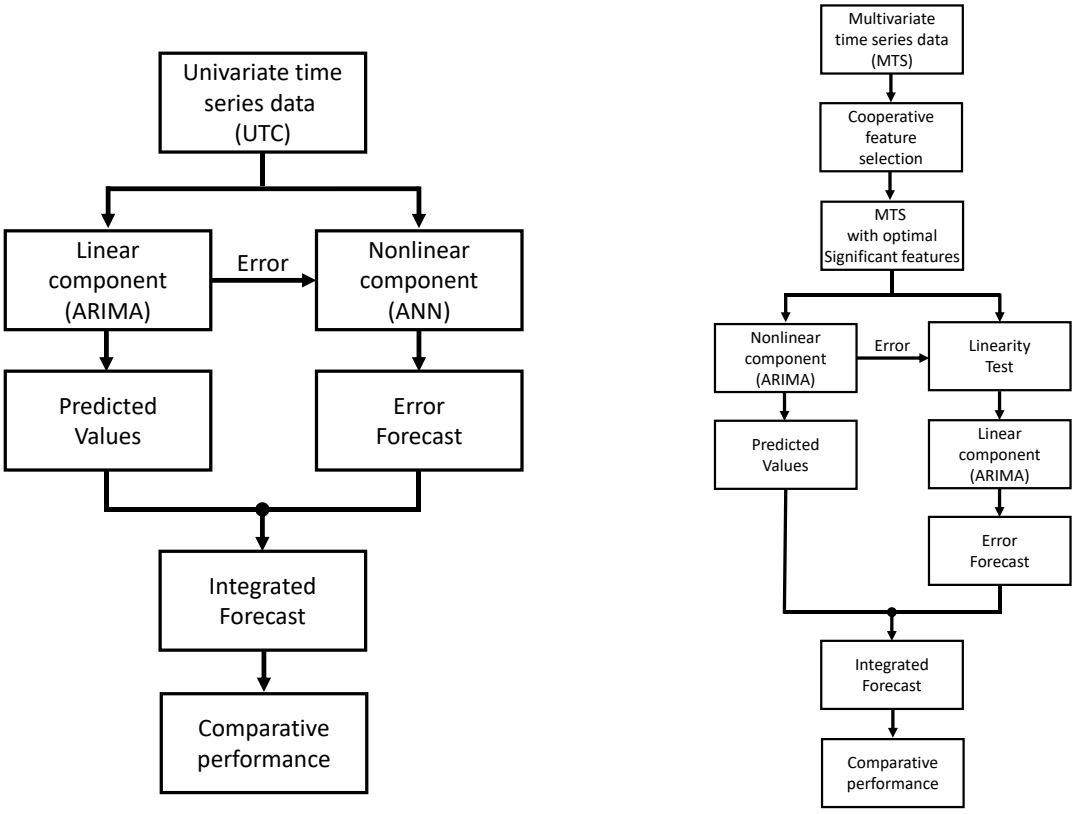
GRANN-ARIMA

Unlike traditional ARIMA-ANN models, Grey Relational Analysis ANN-ARIMA (GRANN-ARIMA) proposed in [15] adds a phase called cooperative feature selection, which filters

the unimportant and irrelevant features to accelerate the learning process and to improve the precision of prediction. In addition to differing from the conventional hybrid models, the rough predicted values are provided by ANN while ARIMA contributes the error forecast in GRANN-ARIMA. The procedure of GRANN-ARIMA is shown in 3.13a.

ARIMA-ANN

According to ARIMA-ANN methods, time series consists of the nonlinear and linear components. The sequence of the linear component will be forecasted by ARIMA-based approaches while ANN-based methods will be used to predict the counterpart (i.e., error) of nonlinear one. The hybrid model integrates the rough forecast values with the estimated errors, outputting the prediction [111, 159, 160, 161, 162]. The procedure of ARIMA-ANN methods is displayed in 3.13b.



(a) Traditional ARIMA-ANN procedure

(b) GRANN-ARIMA procedure

Figure 3.13: Procedures of different models [15].

3.2.4 Summary

To the best of our knowledge, most researchers prefer to adopt a unique solution of either being an ARIMA variant or an ANN-based alternative to predict time series. A few works consider the implementation of combining both methods, whose outcomes have usually been proved to surpass the performance of either solo counterpart. Most hybrid forecast models are applied in business areas (e.g., price [111], logistics amount [163], and stock [164]), none of which is as sophisticated as that in network traffic prediction.

However, the remarkable performance of the hybrid method concerning forecasting traffic flow [165] inspires our interests to seek similar application in predicting network traffic. Moreover, authors in [166] compare four ARIMA-ANN prediction models for network traffic data, whose experimental results show that the performance of multi-step prediction is worse than that of one-step prediction. Therefore, it is intriguing to improve the precision of multi-step network traffic prediction by leveraging ARIMA-ANN hybrid model. Variously heterogenous prediction approaches regarding vehicle traffic flow or network traffic are compared in Table 3.2.

Table 3.2: Comparison of Diversified Prediction Methods

	Features	**Model-Driven**	**Data-Driven**	**Year**
ARIMA [132]	Describes the relationship of autocorrelation and partial autocorrelation behind the time series	Yes	No	1971
FARIMA [135]	Simplifies the fitting process of ARMIA	Yes	No	1999
Subset ARIMA [136]	Building an ARIMA with less parameters, providing more stable error	Yes	No	1999
SARIMA [134]	Reveals the seasonality of the time series	Yes	No	2003
SFARIMA [137]	Continuously slides the forecast prediction to find the best fitting model	Yes	No	2009
EMD [138]	Decomposes the nonlinear and non-stationary time series into a small number of IMFs	Yes	No	1998
EEMD [139]	Repeatedly adds diversified white noise series into the time series and decomposes the resulting data	Yes	No	2009
MC [140]	Relies on the short memory principle to predict wireless network	Yes	No	2009
WT [141]	Decomposes time series into different time scales and reconstructs the predicted values with wavelet method	Yes	No	2012
ENN [144]	Adds an extra undertake layer to improve the learning stability of unit weights	No	Yes	2012
DTC [145]	Considers spatial correlation of the time series by using multi-dimensional array	No	Yes	2012
NARX [133]	Improves computational efficiency by restricting feedback architecture of recurrent neural network	No	Yes	2016
LSTM [114, 148]	Provides efficient forecast in non-linear datasets	No	Yes	2016
GRU [149]	Being similar to LSTM but with a simpler cell structure	No	Yes	2014
GA-ELM [147]	Uses the GA to optimize the parameters of ELM	No	Yes	2016
Bayesian Network [150]	Can predict with limited training data	No	Yes	2006
KNN [152, 153]	A nonparametric prediction method which checking the similarity between real-time and historic data	No	Yes	2013
SVR [155]	Maps a nonlinear problem into a linear one	No	Yes	2016
EMD-MMP [157]	Decomposes and partitions the time series into two portions, then summing up the expected results produced by ARMA and SVR	Yes	Yes	2014
KARIMA [158]	Classifies the ARIMA models with different parameters (p, d, q) based on the features of different time series	Yes	Yes	1996
GRANN-ARIMA [15]	Filters the irrelevant features during the stage of prepocessing, and implements error estimation by ARIMA	Yes	Yes	2008
ARIMA-ANN [159, 160, 111, 161, 162]	Perform better accuracy than those solo model ones	Yes	Yes	2016

Chapter 4

Proposed Scheme

To alleviate the resilient issues and to improve the network robustness in SDN, an advanced scheme is proposed, which is composed of two major proposals - ReON and L-SARIMA. The ReON method contains the main module with two characters consisting of Global Resilient Orchestration Module (GROM) and Local Resilient Orchestration Module (LROM). The monitor, prediction, detection and backup functions are implemented in GROM. Meanwhile, the surveillance and recovery functions are implemented in LROM.

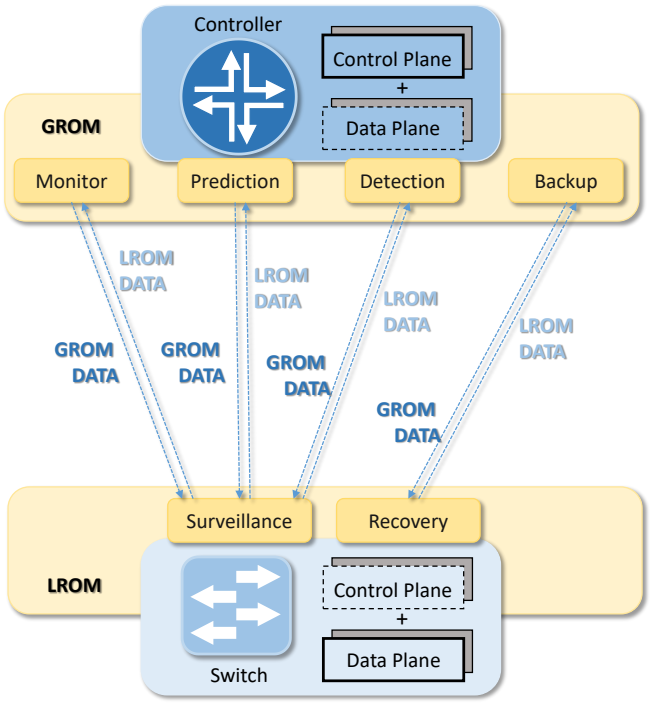


Figure 4.1: The proposed ReON architecture.

In reference [167], Thottan et al. categorized failure detection as rule-based approaches,

pattern matching, and statistical analysis. Specifically, rule-based approaches pre-define the symptom of a fault to identify the failure (e.g., excessive utilization of bandwidth, exceeded throughput, etc.). Pattern matching, which can be analogous to ML methods, builds up a traffic profile for the given network by online learning its specific feature (e.g., link utilization, packet loss, etc.). Statistical analysis such as AR, MA, ARMA, ARIMA and so on, provides offline learning on the characteristics of network traffic. The presented LSARIMA method combines the features of statistical analysis and pattern matching, which supports the prediction function with an undoubtedly accurate model.

Specifically, the goal of this proposed scheme is to present a new idea that SDN network can automatically detect and prevent the network collapses in advance, which assists network operators to reduce the substantial effort of analysis. By using a hybrid method consisting of SARIMA and LSTM, the prediction function can provide an accurate forecasting baseline to the detection function. Additionally, the GROM receives the status of the entire network and ranks the candidates that can turn into the controller, which then eliminates the potential failure according to the abnormal network traffic compared with the baseline. The brief architecture of the proposed scheme is shown in Figure 4.1.

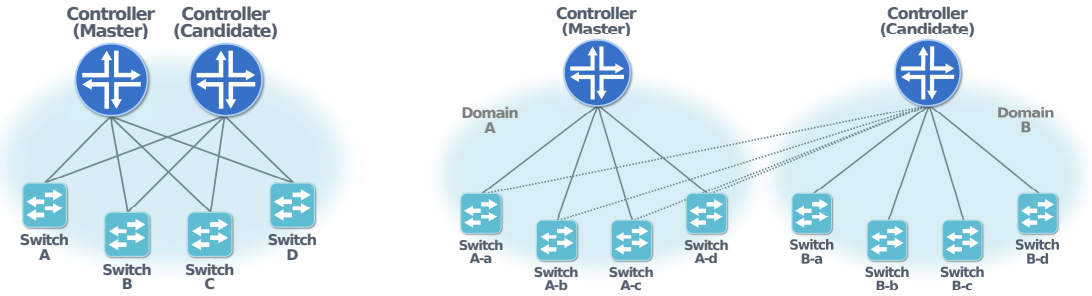
4.1 Resilience-Oriented and NFV-supported (ReON)

4.1.1 Multi-Controllers Problem Statement

In order to improve the reliability of SDN, many papers suggest applying distributed controlling functions in case that the single point of failure issue occurs in the networks [13, 30, 168]. The common idea among those studies is to deploy multiple SDN controllers within a single domain or across several domains. In both cases, the master controller dominates the setup of routing policies in the entire network. However, the candidate controller idles in the scenario of a single domain, or it manages the switches of other domains in the case of across several domains.

As shown in Figure 4.2a, if the controllers are in the same domain, the candidate controller remains inactive on the regular unless the primary controller failure happens, then it takes over the network activities. In addition, Figure 4.2b illustrates the case that the controllers are in different domains. For instance, the controller in *Domain_B* is the backup controller of *Domain_A*, which is also the dominant controller within its own domain simultaneously. The flows of switches in *Domain_A* (i.e., Switch_A-a to Switch_A-d) will be redirected to the controller in *Domain_B* when the controller of *Domain_A* fails.

However, the candidate controller in the former solution cannot be leveraged as thoroughly as the one in the latter situation because it does not participate in any network events when the master controller runs normally. Furthermore, a sudden outburst of traffic load increases the candidate controller in the second scenario when the master controller in *Domain_A* experiences a malfunction. Therefore, the proposed scheme alleviates the disadvantages of either case by utilizing NFV, which can convert a switch to a controller if necessary.



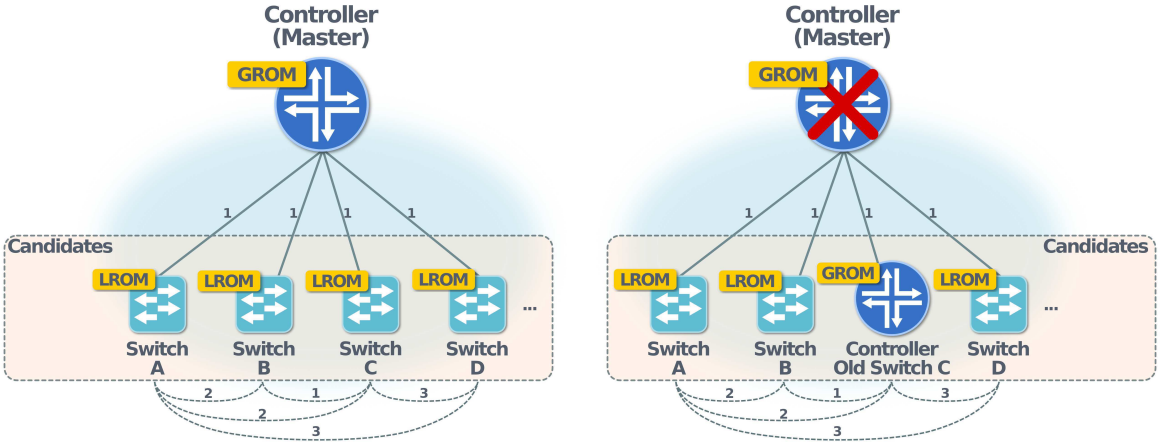
(a) Backup controller in the same domain

(b) Backup controller in the adjacent domain

Figure 4.2: Cases of backup controller placement.

For instance, Figure 4.3 depicts a partially-meshed network topology with weighted links, where the weights represent latencies and all switches are potential candidates to become an active controller. In the proposal, this thesis considers that the latency impacts on the network performance. Thus, a position can be determined to be optimal if it owns the least latency in total, which shares the same objective with the CPP [34].

In the proposed scheme, the orchestration module of the controller frequently calculates the latency given by the switches. Then it chooses the switch in the optimum position (i.e., the least overall latency) as the backup controller based on two metrics, consisting of CPP-average latency and CPP-maximum latency. Table 4.1 displays the matrix of latency in the scenario above, which demonstrates *Switch_A* and *Switch_C* possess the lowest values of maximum latency while *Switch_C* owns the least average latency as well. Therefore, *Switch_C* is the best option to replace the original controller that stops working.



(a) Before failure

(b) After failure

Figure 4.3: Controller failure scenario.

The average latency can illustrate the brief status of the network; however, the averaging calculation may neutralize the worst case. Thereby, maximum latency can be leveraged to

Table 4.1: The Matrix of Latency

	Controller	**A**	**B**	**C**	**D**	**Total**
Control	0	1	1	1	1	**4**
A	1	0	2	2	$3^{\#}$	8
B	1	2	0	1	4	8
C	1	2	1	0	$3^{\#}$	**7***
D	1	3	4	3	0	11

$\#$ denotes the minimum latency among all the maximum latencies of the switches
* denotes the minimum latency among the average latencies of the switches

compensate. Assume an undirected network graph $G(N, L)$ where N is the set of switches, L is the set of latencies between two directly connected switches, and n is the quantity of N. The latency matrix $m_{x,y}$ as shown in Table 4.1 contains the propagation latencies between each couple of switches, where $x, y \in N$ and $x \neq y$. Besides, the average and the maximum latencies are denoted as ϕ_{avgL} and ϕ_{maxL} correspondingly. Thus, we can obtain the least average and maximum of latencies from Equations 4.1 and 4.2 respectively.

$$\min \phi_{avgL} = \min_{(x \in N)} (\frac{1}{n} \sum_{(y \in N)} m_{x,y}) \tag{4.1}$$

$$\min \phi_{maxL} = \min_{(x \in N)} (\max_{(y \in N)} m_{x,y}) \tag{4.2}$$

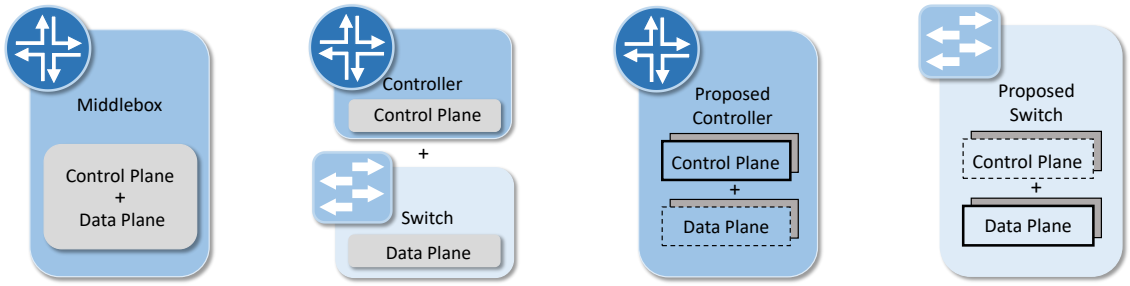
Moreover, the loss rate of a transmission path results in the declined of network reliability. Consequently, both loss rate and latency are supposed to be considered when choosing a path. In this thesis, it sets replacement metrics as γ_{avg} and γ_{max}, both of whose formulas include the loss rate of a path $P^{loss}_{x,y}$. Thus, the objective that finds the least average and maximum metric values are achieved by Equations 4.3 and 4.4.

$$\gamma_{avg} = \min_{x \in N} (\phi_{avgL} \in \min_{y \in N} P^{loss}_{x,y}) \tag{4.3}$$

$$\gamma_{max} = \min_{x \in N} (\phi_{maxL} \in \min_{y \in N} P^{loss}_{x,y}) \tag{4.4}$$

4.1.2 ReON Components

In general, the traditional OSI layer-3 network devices illustrated in Figure 4.4a, such as routers and firewalls, integrate the control and forward planes. In contrast, the Figure 4.4b demonstrates that the conventional concept of SDN prefers to decouple the integration of two planes, deploying them into different physical devices. However, the proposed scheme ReON in the thesis suggests abstracting the controller and the switch as two particular VNFs that are installed into the same machine, which are displayed in Figure 4.4c



(a) Traditional router (b) Conventional SDN (c) *ReON* controllers (d) *ReON* switches

Figure 4.4: Traditional network elements vs. proposed controllers and switches.

and 4.4d. Note that the bold black boxes in the last two figures denote the primarily working instances, while the ones in dash-lines denote the inactivated instances.

Although the original idea of SDN is to reduce the processing overhead of data plane, which only allows an OF switch to own the forwarding property, the design of ReON does not change this network paradigm and fully leverages the computing ability of the switch. Currently, many open-source SDN controllers show the decent processing performance, which proves the controlling function does not require a dedicated device. Especially ONOS, OpenDayLight, Iris, and MUL can usually be applied in DC [53]. Additionally, a lightweight carrier-grade controller consists of Link discovery, topology, storage, strategy making, flow table and control data modules, which can easily be deployed into a Linux KVM. Therefore, it is feasible to implement this proposed model.

4.1.3 Resilient Orchestration Module

In this work, an extra module with two roles is introduced to support the controlling and forwarding VNF components, which also collects the real-time network status and calculates the latency and loss between nodes. Consequently, the essential metrics computed by the resilient module can be utilized for determining the SDN controller recovery. The Resilient Orchestration Module (ROM) can assign the resilience ability within the entire SDN scope while preserving the software-defined and control-centralized characteristics. As discussed in 4.1.2, each device proposed in ReON involves both the controller VNF and the switch counterpart. For example, the switch VNF stays inactivated when the machine is running as a controller.

Figure 4.5 shows that the ROMs are deployed into the ReON controller and switches, which respectively denotes GROM and LROM. Each type of ROM has its representative functions: LROM consists of surveillance and recovery functions, while GROM includes monitor, prediction, detection and backup functions. Two roles of the ROMs are relative, where the candidate ReON machine with exclusively forwarding function (i.e., switch) can be promoted to become the controller when the breakdown occurs in the primary ReON

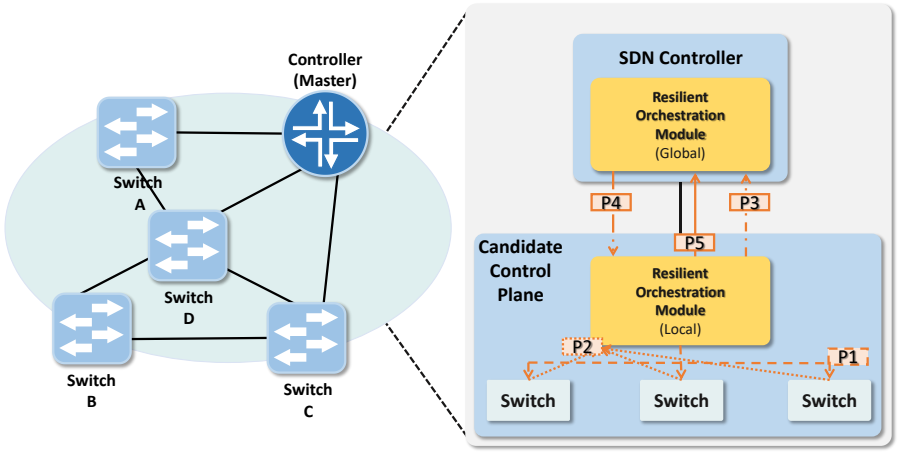


Figure 4.5: Resilient SDN Controller Architecture Design.

controller.

Local Resilient Orchestration Module (LROM)

1. **Surveillance Function**. Through requesting ICMP packets, the LROM of each switch surveils and accumulates the propagation latencies and losses between itself and other peers regularly, as described by *P1* in Figure 4.5. At the stage of *P2*, the average or the maximum latency among the switches can be computed after analyzing the loss probability. The average latency denotes the mean value of latencies from every switch to other rivals, and the maximum latency represents the largest value among the latencies obtained by each switch from itself to other nodes. The objective of this function is to acquire the minimum value from either strategy.

   Algorithm 1 is presented to work for the surveillance function in LROM, where the inputs are the array of partner switches, the number of partner switches, and the type of latency-selecting strategy. Meanwhile, the outputs are the latency chosen according to the corresponding strategy and the loss rate that is calculated by the division of the unreachable ping times by the number of peers. Moreover, the computed results will be transmitted to the GROM in the controller, as illustrated in *P3* of Figure 4.5. Each message of a switch contains a time stamp, so the controller starts to calculate a new processing era when it has received the packet with the very first time stamp. If the controller cannot receive the result message from some switches during a processing era, it will send the information requests to them and waits for a certain interval. After several attempts (i.e., three times in default), the controller will consider the LROM of the switch may encounter a problem and then continues the investigation by using the *detection function* 3.

Algorithm 1 LROM Calculation

Input: $switchArray$, $switchNum$, $latencyType$
Output: $latency$ and $loss$

```
1: function CalculateLatency(array, num, type)
2:     switchLatencyArray ← []
3:     switch ← self.switch
4:     for i = 0 → num − 1 do
5:         if switch != array[i] then
6:             switchLatencyArray[i] ← ping results
7:             if switchLatencyArray[i] == null then
8:                 switchUnreachableNum++
9:             end if
10:        end if
11:    end for
12:    lossRate ← switchUnreachableNum/num
13:    if type == averageLatency then
14:        latency ← avg(switchLatencyArray)
15:    else
16:        latency ← max(switchLatencyArray)
17:    end if
18:    return latency and lossRate
19: end function
```

2. **Recovery Function**. This function is triggered by the signal message from the detection function 3 of GROM in the controller when the potential failure is determined. When the recovery procedure takes effect, a replacing message is delivered by the candidate switch to other rivals and the original controller. The rest switches then send the confirmation messages to the candidate switch.

   Controller failure can be categorized into two cases - one situation is that the process of controller VNF encounters a malfunction but the ROM of the device remains normal so it can activate the forwarding VNF. The other one is that the controller machine is offline. In the first circumstance, the previous controller activates the switch VNF to join the switch pool when its GROM receives the replacing message and then disables the controller VNF. Since the candidate switch stores the cache flow tables from the primary controller previously by the *backup function* 4 in GROM, the peer switches direct the new *packet-in* packets to the new controller if they have not received the unchanged responses from the original controller for a specified period. In the second circumstance, the whole recovery process is similar to that in the first condition except the original controller is excluded from the pool of switches. Note that the original controller that becomes a switch is banned in the next candidate selection unless the problem is fixed.

   Both cases above operate in the post-failure scenario. However, the advantage of the proposed scheme is detecting the failure in advance according to the network

traffic pattern achieved by the *prediction function* 2 in GROM, which can replace the potentially broken controller before the failure happens.

Global Resilient Orchestration Module (GROM)

1. **Monitor Function**. After receiving the information on latency and transmission loss from the LROM, the monitor function of GROM ranks the priority of successor among switches and arranges the candidate list. Algorithm 2 serves as the core ranking function, whose inputs are the array of latency, the array of loss rate, the number of switches, as well as the acceptance threshold of loss rate. Also, the output is the sorted list of candidates. This algorithm searches the array of loss rate, with putting the switches that hold the value below the threshold into the queue.

Algorithm 2 GROM Ranking

Input: $latencyArray$, $lossArray$, $switchNum$, $threshold$
Output: $candidate$

```
1: function SelectCandidate(array1, array2, num, α)
2:     indexArray ← []
3:     for i = 0 → num − 1 do
4:         if array2[i] <= α then
5:             indexArray[count] ← array1[i]
6:             count + +
7:         end if
8:     end for
9:     sortedSwitchIndex ← QuickSort(indexArray)
10:    return sortedSwitchIndex
11: end function
```

In the next step, it sorts the switches by the decreasing sequence of latency, which leverages the quick sort method [169]. OF-based messages are dispatched to the switches to notify the candidate controllers. Moreover, those messages contain the information that needs to back up in the *backup function* 4. When the selected switches have received the messages, they send the keep-alive message to the controller at a given interval.

2. **Prediction Function**. Besides the latency and transmission loss, the GROM also collects the network traffic from switches and itself within the entire SDN domain periodically. Based on the historical data, the hybrid prediction model presented in Section 4.2 can construct a real-time forecast, which can be used as the baseline of regular traffic in *P5* of Figure 4.5. According to the pattern, the *detection function* 3 of GROM can conclude that whether the potential node failure will exist. Thus, this prediction function can be applied in two aspects, which consists of triggering the controller replacement and the route redirection of the network traffic flow.

3. **Detection Function**. This function supervises the network traffic of all nodes collected by the *monitor function* 1, which detects the abnormal network against the

baseline built by the *prediction function* 2. Furthermore, the detection result can announce to the *surveillance function* 1. When the controller identifies the potential failure, where it is about to be overloaded based on the workload that is being handled by itself at a given moment, the GROM sends messages of replacement identification to the candidates. Besides, the system considers the controlling VNF is suspended due to overload or malfunction when the controller does not respond the keep-alive messages beyond the timeout threshold. In both cases, the replacement process will be established.

4. **Backup Function**. After selecting the switches with the minimum latency as the candidate controllers, the GROM transmits the flow table to the LROMs of candidates in *P4* of Figure 4.5. Whenever the flow table changes, the information is synchronized between the controller and the candidates. As a result, the network can be rapidly recovered once the controller replacement is proceeding or the actual controller failure happens.

4.2 LSTM-assisted SARIMA (LSARIMA)

In reality, the actual time series of network traffic consists of neither pure linear nor absolute nonlinear [15], which are composed of both portions, so that it is challenging to approximate the precise result by using a single model. However, the ARIMA-based methods are usually experts in modeling the linear section, while the ANN-based methods can specialize on the nonlinear section [111]. Because of the self-similar nature of the Ethernet traffic, the observations of network flows can be considered as linear in the general trend with nonlinear white noise that represents outburst in some points. Additionally, the residuals between the expected and the historical values can be regarded as nonlinear as well.

In [170], the authors concluded that the ARIMA-based methods provide better accuracy than the ANN-based counterparts do in the aspects of building a prediction model for a general linear time series. In contrast, the ANN-based methods outperform the ARIMA-based ones regarding the non-linear time series. However, the criteria for the experiment they provided are under one-step prediction. Furthermore, research implemented in [166] explains that neither type of forecast methods can perform well under multi-step condition. Thus, a hybrid prediction model is required to provide a more precise forecast in multi-step.

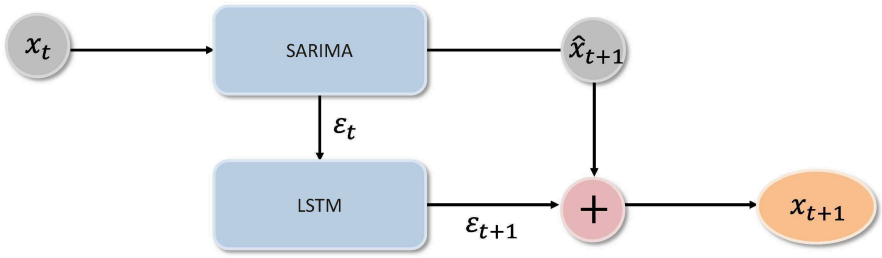


Figure 4.6: The Model of LSTM-assisted SARIMA.

Figure 4.6 demonstrates the process of L-SARIMA model, where the time series x_t is the consolidation of linear and nonlinear parts. The outputs of the SARIMA comprise sequence $\hat{x}_{t+1}$ that is the candidate prediction and ε_t that denotes the residuals of previous candidate prediction $\hat{x}_t$ with respect to Equation 4.5. Additionally, the LSTM yields the predicted residual ε_{t+1} through inputting ε_t, then the hybrid model produces the adjusted prediction x_{t+1} by combining $\hat{x}_{t+1}$ and ε_{t+1} according to Equation 4.6.

$$\varepsilon_t = x_t - \hat{x}_t \tag{4.5}$$

$$x_{t+1} = \hat{x}_{t+1} + \varepsilon_{t+1} \tag{4.6}$$

Specifically, the hybrid method detects if the historical data is seasonal at the beginning, in order to choose the base prediction model (i.e., SARIMA or ARIMA). Furthermore, the time series is divided into training and testing segments. After regressively determining the

most acceptable parameters for the (S)ARIMA part, which concerns the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) [171], the LSARIMA builds up the fundamental forecast model and predicts the candidate results at the "training" stage. Meanwhile, the LSTM part of the proposed method forecasts the predicted residuals by being fed with the previous residual sequence that calculated by the historical observation and candidate prediction.

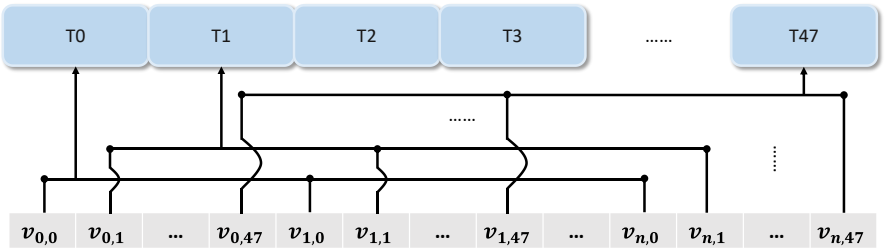


Figure 4.7: Dividing the Time Series into Small Time Blocks.

During the period of rectification, the hybrid model segregates the time series based on the values within the same time block. For example, the sequence $\{v_{0,0}, v_{1,0}, ..., v_{n,0}\}$ belongs to block T0, the sequence $\{v_{x,1}|x : 0 \rightarrow n\}$ belongs to block T1 as depicted in Figure 4.7. The rest sequences follow the same rule. Since there are 48 values in one-day data, there will be 48 groups of sequences in total. The modified outputs can be obtained by Equation 4.6.

Algorithm 3 LSARIMA Prediction

Input: *time_series*, *ratio*
Output: *expected*

function LSARIMA_Prediction(*time_series*, *ratio*)
    is_seasonal ← Detect_Seasonal(*time_series*)
    training ← *ratio* × *time_series*
    training_residuals ← *ratio*^{2} × *time_series*
    testing ← (1 − *ratio*) × *time_series*
    if *is_seasonal* == *true* **then**
        base_model ← Build_SARIMA_Model(*training*)
    else
        base_model ← Build_ARIMA_Model(*training*)
    end if
    residual_model ← Build_Residual_Model(*training_residuals*,
                        training)
    predict_residual ← LSTM(*residual_model*, *testing*)
    predict_network ← (S)ARIMA(*base_model*, *testing*)
    expected ← *predict_residual* + *predict_network*
    return *expected*
end function

As illustrated in Algorithm 3, LSARIMA separates the time series into training, residual training and testing portions with a ratio. During the procedure of building the base

model, LSARIMA keeps searching the $[p, d, q, P, D, Q]$ parameters according to the minimum values of AIC and BIC. When both base and residual models are constructed, the (S)ARIMA part predicts the raw network traffic and the LSTM part forecasts the residuals, where the predicted network traffic complements to the forecasting residuals. With adjusting the expected values, the prediction of the hybrid model can outperform either single model methods in multi-step prediction.

Chapter 5

Experimental Evaluation

This chapter presents the settings and methodology of experimental environments, including simulation testbed, time series observations, evaluation metrics, and the results of the comparison.

5.1 Experimental Setup and Observations

5.1.1 Time Series (Prediction Model Evaluation)

To evaluate the performance of the hybrid prediction model, we determined to use two sets of time series based on actual network traffic datasets. They are the Defense Advanced Research Projects Agency (DARPA) dataset from MIT [172] and the network traffic of backbone in Japan from Widely Integrated Distributed Environment (WIDE) Project [173].

The former one has shown its wide acceptance in related works on intrusion detection, which has been judged to be the bias of redundant records in the DARPA training sets recently [174] though. Nonetheless, the normal network traffic hidden in the dataset is approximate to the network traffic in reality, which is meant for assessment. Therefore, the regular network traffic is extracted from the continuous 17-day training data (i.e., 1999/03/01 - 1999/03/17), as exhibited in Figure 5.1. The entire dataset has been divided into 48-point values per day, which compose an 816-point (48×17) sample in total. Apparently, the observed data are seasonal and stationary.

The latter one is shown in Figure 5.2. The dataset is collected at 15-minute intervals for 15 days (i.e., 2017/01/01 - 2017/01/15), which displays a less noticeable pattern. In order to construct the prediction models smoothly, the format of the WIDE observations is preprocessed to be as similar as that of DARPA dataset, which is also 48 values for one day. Thus, the experimental sample contains 720 (48×15) elements.

Therefore, these two observations can represent two different kinds of datasets, which are used to justify the generalized usage of the proposed algorithm. Moreover, normalizing data is applied to diminish the biases between datasets with different sizes, which can

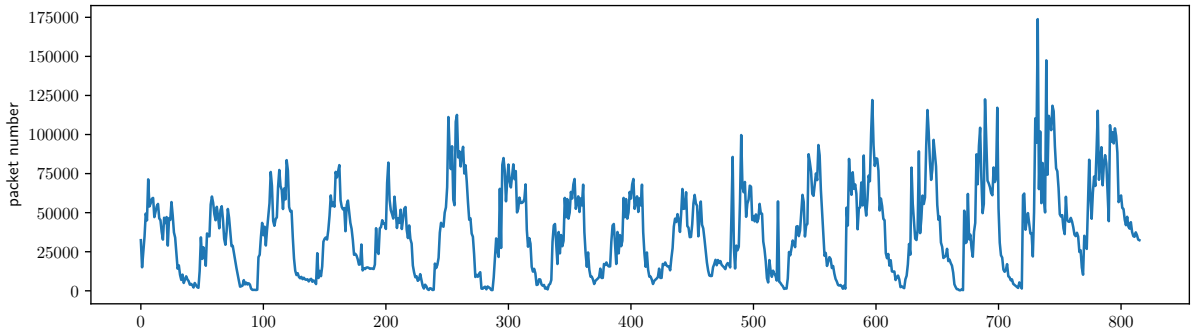


Figure 5.1: Normal Network Traffic in DARPA datasets.

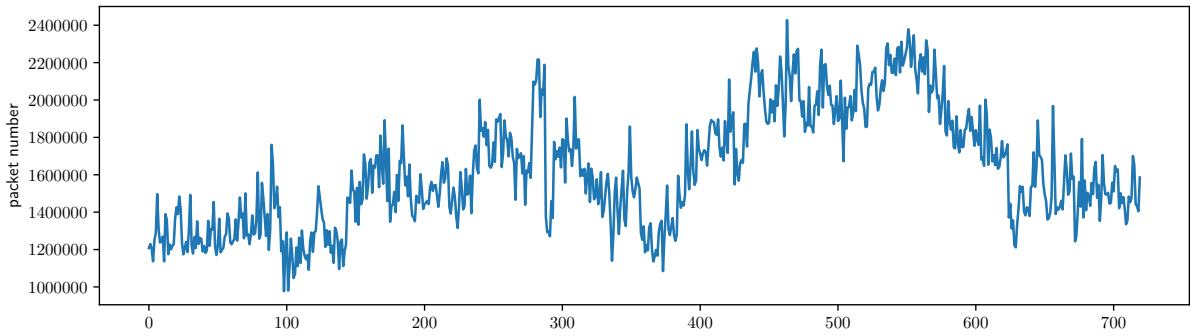


Figure 5.2: Network Traffic of Japanese Backbone in Project WIDE.

interpret the underlying relationship between observations [175]. The formula can be expressed as Equation 5.1.

$$\bar{X} = \frac{x_i - \min(X)}{\max(X) - \min(X)} \tag{5.1}$$

where X is the collection of observed data; x_i is one of the values in X; min() and max() mean the minimum and maximum values in the collection, respectively.

In this work, we choose 16 days out of 17 days as training data in DARPA datasets; meanwhile, 14 days out of 15 days are chosen as training data in WIDE datasets. Both periods of predictions are one day, which serves as an out-of-sample prediction of 48 points (i.e., multi-step prediction).

5.1.2 Testbed

Regarding evaluating the prediction performance of the hybrid model, we choose the Linux-based OS, Ubuntu 16.04 LTS and Python 2.7.13, where SARIMA and LSTM models are implemented based on the third-party python API named Statsmodels [176] and TensorFlow [177] to avoid the biases.

Regarding testing the efficiency of the proposed ReON model, we selected the experimental testbed consisting of the Mininet emulator [178] and the Ryu controller [179] that provides the fundamental SDN features. These two applications are deployed in different VMs. Note that the emulated OF switches in the Mininet are logically isolated by network slicing, in which the IP addresses cannot be set up. Therefore, an extra host with the identified IP is assigned to each switch for assessment purpose.

5.1.3 Topologies

During the experiments, two topologies are considered. One is the K-Fat-tree [16], which is usually a three-layer topology consisting of the core, aggregation, and edge layers. This topology is prevailing in the deployment of a scalable data center. Because of the abstraction of the SDN functionalities in the ReON model, the scenario of deploying the ReON VMs into the DC is extremely feasible.

The architecture of K-Fat-Tree is illustrated in Figure 5.3. Supposed there are k pods, each pod is composed of $(\frac{k}{2})^2$ servers, $\frac{k}{2}$ aggregation switches and $\frac{k}{2}$ edge switches. In addition, each core switch connects to all the pods while each aggregation switch connects to $\frac{k}{2}$ core switches and $\frac{k}{2}$ edge switches, and each edge switch connects to $\frac{k}{2}$ aggregation switches and $\frac{k}{2}$ servers. According to the definition of this topology, the increment of switches is dramatic when the k value increases, which can be used to analyze the time complexity.

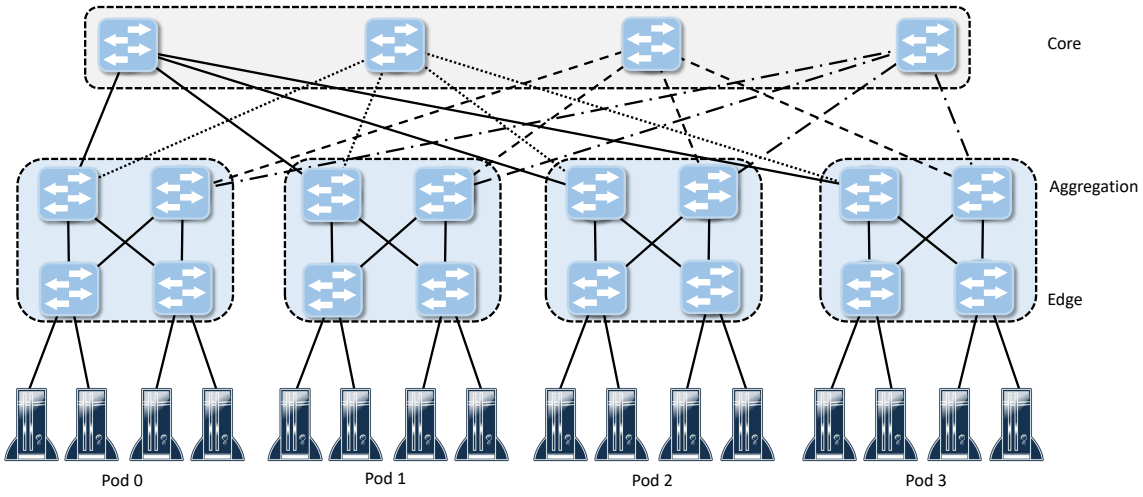


Figure 5.3: K-Fat-Tree Topology [16].

Another one is the Internet2 OS3E topology [17], which provides advanced layer-2 services of backbone network to research community. Moreover, it embodies the SDN structure, with being widely used in the area of studying CPP [180]. The reasons that pick the OS3E topology are twofold: 1) The ReON scheme proposed here can formulate the problem as the CPP, in which the topology is applied to assess the efficiency of the

presented model. 2) It is a sparse network consisting of 34 nodes and 43 edges, which can be a typical SDN scenario in reality. Thus, the accuracy concerning candidate controller selection and node failure detection can be evaluated through the topology.

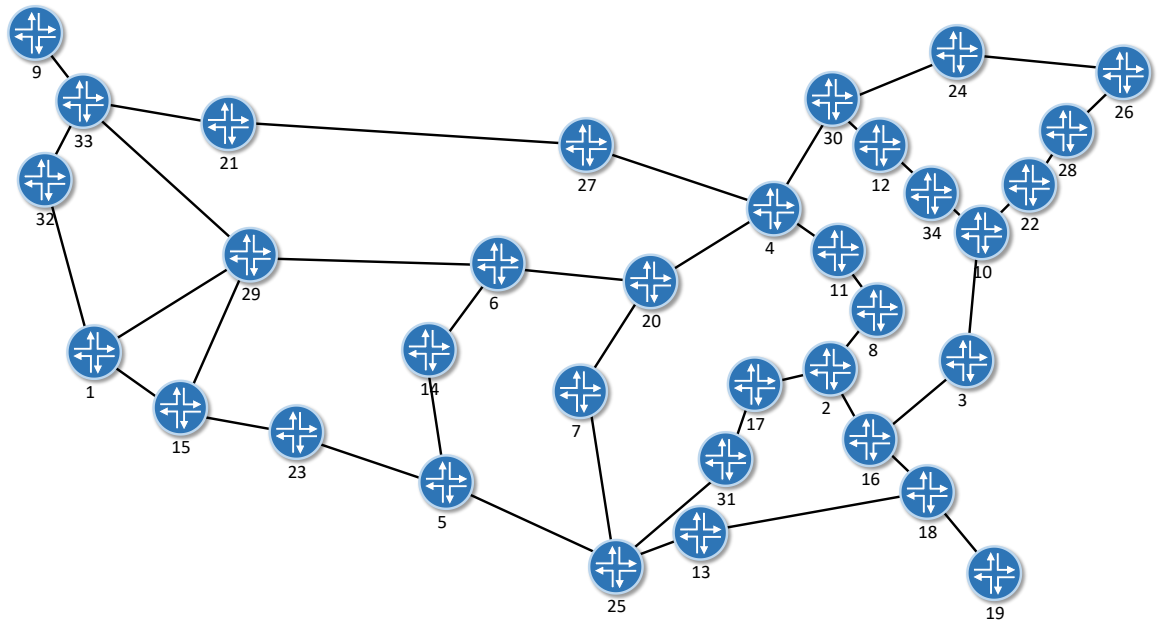


Figure 5.4: OS3E Topology [17].

5.2 Metrics

In order to test the performance of LSARIMA and ReON, this section splits the evaluation metrics into three categories, consisting of prediction accuracy, time complexity and position matching precision (i.e., mapping of node failure detection).

5.2.1 Prediction Accuracy

To guarantee the sanity of detection baseline, the pattern of normal network traffic built by the prediction function has to be as precise as possible. Four metrics are generally considered to estimate the forecasting performance [181, 156]:

1. **Normalized Mean Square Error (NMSE)**. NMSE is the average of squares of the differences between the expected and actual data after being normalized, as demonstrated in Equation 5.2.

$$NMSE = \frac{1}{n}\sum_{i=1}^{n}((norm(X_{real,i}) - norm(X_{predicted,i}))^2) \tag{5.2}$$

2. **Normalized Root Mean Square Error (NRMSE)**. As illustrated in Equation 5.3, NRMSE examines the similarity of residuals between observations and predictions, where lower values stand for fewer variances.

$$NRMSE = \frac{1}{\bar{X}_{real}} \sqrt{\frac{\sum_{i=1}^{n} (X_{real,i} - X_{predicted,i})^2}{n}} \tag{5.3}$$

3. **Mean Absolute Percentage Error (MAPE)**. MAPE formulated as Equation 5.4 is known as the inverse accuracy estimation for forecasting methods, which usually describes the errors as a percentage.

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} |\frac{X_{real,i} - X_{predicted,i}}{X_{real,i}}| \tag{5.4}$$

4. **R-squared ($\mathbf{R^2}$)**. The coefficient of determination or model match denoted as R^2 indicates typically the fitting percentage between the prediction model and the time series, which is defined as Equation 5.5.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \varepsilon_i^2}{\sum_{i=1}^{n} (X_{real,i} - \bar{X}_{real})^2} \tag{5.5}$$

Note that, the lower values $NMSE$, $NRMSE$, and $MAPE$ obtain, the better performance the method has. In contrast, the prediction model fits the observation better when the R^2 is higher.

5.2.2 Time Consumption

Time consumption presented here expresses the amount of time that a desired candidate position of placement is elected, which gives an overview over the complexity of the proposed algorithm, thereby allowing to clarify the efficiency of a method.

5.2.3 Position Matching

The most important factors in the proposed model are the accuracy of candidate controller selection and that of node failure detection, which determines the effectiveness of the whole system. Consequently, this thesis interprets them by employing the *match position rate*. This metric compares the outcome of candidate selection methods with the result of the Brute Force (BF) [182]. BF selection is an enumerative method for finding the globally optimized value, which provides the most optimal position of a replacement controller among the available switches. The BF presented here is associated with Dijkstra's shortest path algorithm [183], which is regarded as the evaluation baseline that is used to match against the proposed ReON scheme.

5.3 Existing Competitors

This section introduces the existing rivals that compete with the proposed methods in this thesis, where they are partitioned into two subsections. The first one describes the single models that are leveraged in network traffic prediction, while the second one expresses the advanced methods that are used as solutions to the CPP.

5.3.1 Network Prediction Models

During the experiments, two major collections of prediction methods are investigated, which are statistical approaches and ML methods as stated in Section 3.2. Furthermore, this thesis only considers a subdivision of ML family called ANN that is also known as deep learning. The representative models are depicted as follows.

Seasonal Autoregressive Integrated Moving Average

In order to pinpoint the relationship between values of a time series, Box and Jenkins generalized a fitting model named Autoregressive Integrated Moving Average (ARIMA), which is also called the Box-Jenkins model assuming the observations stationary [184]. This linear model formed as $ARIMA(p, d, q)$ is composed of AR, I and MA components, where the AR segment with parameter p denotes the hysteresis effects between an observation and previous p lagged counterparts, the I segment with parameter d denotes a d-degree differencing process to the raw time series data for making values stationary, and the MR segment with parameter q denotes the dependency between an residual error and previous q lagged ones [133]. Because of the generalized property, the ARIMA model has been ordinarily applied in time series prediction and proven that it can achieve a decent precision.

The SARIMA formed as $ARIMA(p, d, q) \times (P, D, Q)_S$ is an evolutive model from the ARIMA, where the former part is the same as ARIMA model and the latter part involves the seasonal terms with respect to the parameters in ARIMA formula. If the observations expose seasonality, such as the historical data of network traffic, the SARIMA model can provide more precise forecast results. We choose $\{x_t | t = 1, 2, ..., k\}$ to designate a time series here, thus the general formulation of $SARIMA((p, d, q) \times (P, D, Q)_S)$ can be expressed as the following formulas.

$$\phi_p(L)\Phi_P(L)(1 - L)^d(1 - L^S)^D x_t = \theta_q(L)\Theta_Q(L^S)e_t \tag{5.6}$$

$$\phi_p(L) = 1 - \phi_1 L - \phi_2 L^2 - ... - \phi_p L^p \tag{5.7}$$

$$\Phi_P(L) = 1 - \Phi_1 L - \Phi_2 L^2 - ... - \Phi_P L^P \tag{5.8}$$

$$\theta_q(L) = 1 + \theta_1 L + \theta_2 L^2 + ... + \theta_q L^q \tag{5.9}$$

$$\Theta_Q(L) = 1 + \Theta_1 L + \Theta_2 L^2 + ... + \Theta_Q L^Q \tag{5.10}$$

where L is the lag operator; e_t serves as a Gaussian white noise; $(1-L)^d$ and $(1-L^S)^D$ denote non-seasonal and seasonal differencing respectively; Equation 5.7 and 5.8 represent non-seasonal and seasonal AR terms respectively; Equation 5.9 and 5.10 stand for non-seasonal and seasonal MA terms respectively.

For formulating the SARIMA model with the time series, four model identification stages are required to execute [184]:

1. **Model Identification**. Test the stationarity and seasonality of observation by using the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF), in order to learn the size of lag and the orders of non-seasonal differencing d and seasonal differencing D. After that, exhaustively identify possible p, q, P and Q by utilizing AIC or BIC [171], whose objective is to find $\min(AIC_{p,q,P,Q})$ or $\min(BIC_{p,q,P,Q})$.

2. **Parameter Estimation**. Estimate the coefficients ϕ_p, θ_q, Φ_P and Θ_Q by leveraging the maximum likelihood technique. In case of overfitting the observations, it is suggested to pick the model with fewer parameters.

3. **Model Diagnostics**. Diagnose whether the residuals of the model are restricted to a Gaussian distribution with a constant mean and variance. If the residuals conform, it illustrates the selected model is a reasonably fine one to the time series. Otherwise, the process should return to Stage 1 in order to reproduce a better model with referring to the residuals.

4. **Forecasting Acceptance**. Verify the prediction results from the acceptable model with the actual observations if the model fits well.

Long Short-Term Memory Neural Network

LSTM integrated with the *forget units* that was proposed to solve the gradient vanishing problem, which enables the memory cells to determine which values they should leave behind. In order to accelerate the training process, a variant LSTM associated with "peephole connections" was presented [185]. The structure of an LSTM NN cell is depicted in Figure 5.5. This figure illustrates the LSTM cell that consists of four components: forget gate, input gate, input modulation gate and output gate [186]. Sometimes, the input gate and input modulation gate can be regarded as one gate.

Forget gate determines if the previous output information is going to be stored for the next output values. This determination is made by a sigmoid function based on Equation 5.11, where W_f represents the weight matrices, b_f represents the bias vectors, h_{t-1} represents the hidden state of memory cells, a_t represents the input time series and the

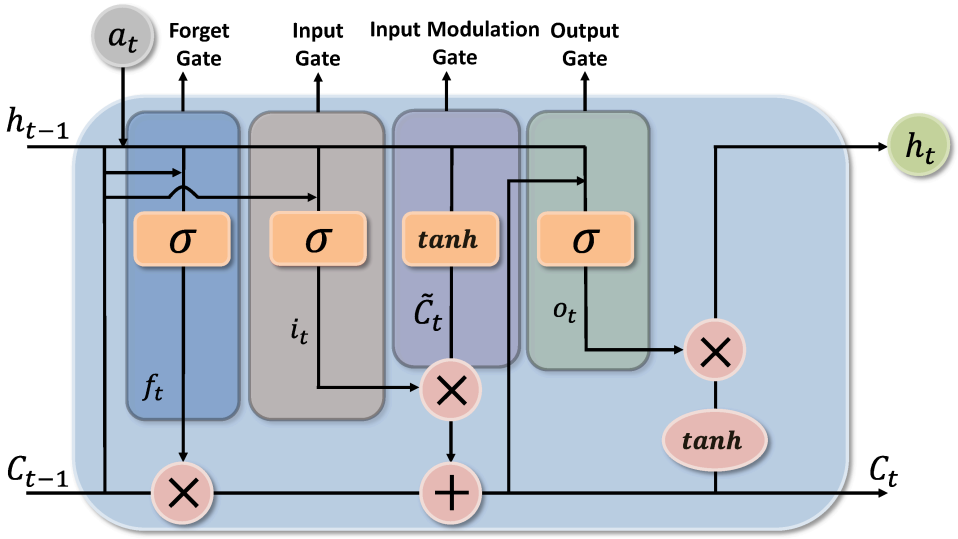


Figure 5.5: The Structure of Variant LSTM Cell.

outputs are denoted as f_t. Each value of f_t is between 0 and 1, which stands for the ratio of keeping the previous cell state C_{t-1}.

$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, a_t] + b_f) \tag{5.11}$$

In the next step, LSTM deposits the new input values into the cell state through input gate and input modulation gate. The input gate decides what values are updated by a sigmoid function (see Equation 5.12), while the input modulation gate generates a vector of new candidate values $\tilde{C}_t$ (see Equation 5.13). Then the new candidate output is calculated by $i_t * \tilde{C}_t$, which is used to replace the old state that is neglected by f_t. Thus, the cell state C_t can be obtained by Equation 5.14.

$$i_t = \sigma(W_i \cdot [C_{t-1}, h_{t-1}, a_t] + b_i) \tag{5.12}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, a_t] + b_C) \tag{5.13}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5.14}$$

Subsequently, the output gate of current LSTM cell supplies two kinds of values to the upcoming cell. One is the cell state C_t, the other one is the hidden state consisting of two portions, shown in Equation 5.15 and 5.16. To output the chosen values, the output gate restricts the cell state C_t within the range from -1 to 1 by using a *tanh* function and multiples it by a sigmoid result o_t.

$$o_t = \sigma(W_o \cdot [C_t, h_{t-1}, a_t] + b_o) \tag{5.15}$$

$$h_t = o_t * \tanh(C_t) \tag{5.16}$$

Note that "$W_x \cdot [a_t, b_{t-1}]$" stands for "$W_{xa}a_t + W_{xb}b_{t-1}$", " $*$ " in the equations above serves as the scalar product of two vectors or matrices, function $\sigma(x)$ and $\tanh(x)$ are defined as Equation 5.17 and 5.18, both of which are variant sigmoid functions.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5.17}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{5.18}$$

Gated Recurrent Units Neural Network

Although LSTM is a useful and precise machine learning prediction method, its complex processing logic makes the training procedure extremely expensive. Therefore, Cho et al. [187] presented GRU, a variant of LSTM, to simplify the learning process. Unlike LSTM, GRU replaces the forget and input gates with an update gate, meanwhile it merges the cell state into the hidden state. Generally, a common construction of GRU cell is composed of an update gate and a reset gate (see Figure 5.6). The formula structures of reset and update gates are similar to the counterparts of forget and input gates in LSTM with respect to Equation 5.19 and 5.20.

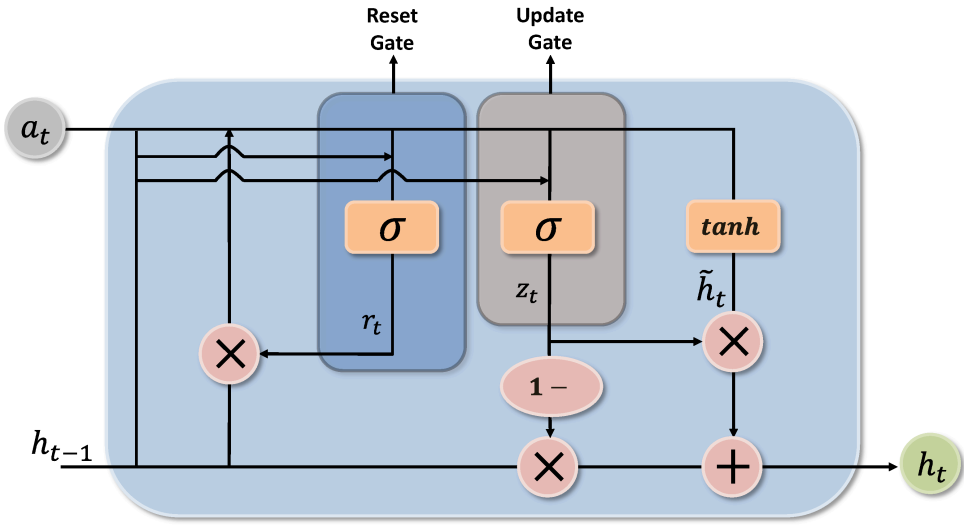


Figure 5.6: The Structure of GRU Cell.

$$r_t = \sigma(W_r \cdot [h_{t-1}, a_t]) \tag{5.19}$$

$$z_t = \sigma(W_z \cdot [h_{t-1}, a_t]) \tag{5.20}$$

To obtain the candidate hidden state, which is similar to $\tilde{C}_t$ in LSTM and can be regarded as currently new data, GRU filters the result with Equation 5.21. In this formula, GRU controls the previous cell state or memory with reset gate, which means candidate hidden state $\tilde{h}_t$ completely overlooks all preceding information if the value of reset gate equals to 0.

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{5.21}$$

In the end, GRU gets rid of the output gate, using currently hidden state h_t as the final output provided by the update gate instead. z_t can decide how much information from the previously hidden state should be forgotten and how much information from current candidate hidden state should be accumulated according to Equation 5.22.

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{5.22}$$

From the Equations 5.21 and 5.22 as mentioned earlier, it depicts that the reset gate can drop previously hidden state information that is irrelevant to the future one; the update gate manages the amount of prior hidden state to be kept. In this thesis, GRU is used to compare with LSTM, from which the model with better precision will be chosen as one of the comparative baselines.

5.3.2 Controller Placement Problem Solutions

Since the ReON scheme is formulated as a CPP, its algorithm can be compared with existing CPP solutions. Mostly, the goal of CPP is to find the global optimum; thus the heuristic approaches that solve the optimization problem are universal. This subsection primarily focuses on simulated annealing and particle swarm optimization.

Weighted Random Early Detection

In order to employ the applications of optimization solutions, this thesis uses Weighted Random Early Detection (WRED) [188] as functional criteria for selecting the optimum position, which is expressed as Equation 5.23. Also, the aggregation latency and rate of unreachability are modeled as the functional parameters in this equation, where *rate* is the ratio of unreachable peers in one switch, *latency* can represent average or maximum values based on different policies, and n is the weighted scalar.

$$\text{Input}_{WRED} = rate \times (1 - 2^{-n}) + latency \times 2^{-n} \tag{5.23}$$

Modified Simulated Annealing

Being a probabilistic technique that approaches to the globally optimal value of a given function, Simulated Annealing (SA) [189] is commonly used as a solution to the optimization problem, which is primarily favorable when the search space is discrete.

This Monte Carlo method has two unique properties:

1. It will move to a worse value than the current one with a probability, which prevents being stuck in the local optimum when analyzing in the search space. The temperature parameter usually determines the acceptance rate.

2. The acceptance rate of the worse value will continuously decrease with the increment of iteration, which allows a broader coverage of the search space at the beginning while converging promptly in the end.

Algorithm 4 is modified based on the Pareto Simulated Annealing (PSA) mentioned in [190] in order to fit in the scenario that achieves the optimum candidate location during the selection. It enables the presented SA algorithm to adopt the optimal position by leveraging the matrix of metric provided by WRED.

Algorithm 4 Simulated Annealing associated with WRED

Input: *latency*, *loss*, *iteration*, *acceptance*, *coolingSpeed*
Output: *optimalPosition*

function SA_WRED($array_{latency}, array_{loss}, n, ar, ratio$)
  $metrics \leftarrow$ WRED($array_{latency}$, $array_{loss}$)
  $T \leftarrow T_0$
  $position_{current} \leftarrow$ generatePosition(T_0, $metrics$)
  while $T > 1$ **do**
    $position_{new} \leftarrow$ generatePosition(T, $metrics$)
    if $latency_{new} < latency_{current}$ **then**
      $position_{current} \leftarrow position_{new}$
    else
      $P \leftarrow random[0, 1]$
      if $P > ar$ **then**
        $position_{current} \leftarrow position_{new}$
      end if
    end if
    if n iterations were finished at T **then**
      $T \leftarrow T \times ratio$
    end if
  end while
  return $position_{current}$
end function

Firstly, the algorithm starts with an initial temperature value T_0 when the experimental procedure begins, where the WRED calculates the metric of weight consisting of the

weighted values of each switch. Moreover, this work selects number 64 as the scalar n of Equation 5.23 since the rate of unreachability possesses higher priority than that the latency has in this particular scenario. Meanwhile, the current optimal position is produced by the T_0 and metrics.

Secondly, Algorithm 4 generates a new position according to the T and the metric matrix given by the WRED and compares the latency of new position with the counterpart of the current position in each iteration. During the searching loop, if the latency of the new position is less than that of the current one, the current position will be replaced by the new one. Otherwise, the replacement can still occur with the probability of acceptance P if it is larger than the threshold ar, where the P is randomly chosen within the fraction ranging from 0 to 1. As stated in the features of SA, value P allows the process of the optimum search to accept a case that is worse than the present one, which enables the algorithm to break the local optimization loop.

Finally, temperature factor T declines with a step factor *ratio* after n iterations of position generation are finished at the end of the period.

Modified Particle Swarm Optimization

Inspired by the study of the social and behavioral sciences, the Particle Swarm Optimization (PSO) was introduced to solve the complex optimization with constraints comprising multiple nonlinear functions [191, 192]. In this thesis, we can consider each switch as the initial place where a particle starts off, and changes its velocity and direction to a new position according to the metrics given by WRED in each iteration.

Based on [193], Algorithm 5 has been modified to fit the applications in this implementation. At the beginning of the procedure, PSO obtains the matrix of metrics through the WRED function, which is similar to what SA does. Additionally, the velocity and position matrixes are initialized randomly. During each iteration, each particle will generate a new velocity and position according to the new velocity, which then checks if it is better than the original position. If the new position is better than the original one, the particle will move to the new place. After all the iterations have been done, the optimal position will be the place where most particles prefer to stay.

Brute Force

The Brute Force (BF) algorithm is associated with Dijkstra's algorithm [183], which deploys SDN controller in each potential location of the topology, generating all the possible results. Among those results, the algorithm obtains the optimal one. Moreover, this BF-Dij is implemented to work as a baseline in order to measure the performance of each placement algorithm, which is used for evaluation purposes, seeking for the trade-off.

Algorithm 5 Particle Swarm Optimization associated with WRED

Input: *latency*, *loss*, *maximum_iteration*
Output: *optimalPosition*

function PSO_WRED($array_{latency}, array_{loss}, n$)
  $metrics \leftarrow$ WRED($array_{latency}$, $array_{loss}$)
  $velocity\_matrix_{particle} \leftarrow$ initialize randomly
  $position\_matrix_{particle} \leftarrow$ initialize randomly
  $it \leftarrow 1$
  while $it < n$ **do**
    for `<each particle>` **do**
      $velocity_{particle,new} \leftarrow$ computeNewVelocity($velocity\_matrix_{particle}$, $metrics$)
      $position_{particle,new} \leftarrow$ generatePosition($velocity_{particle,new}$, $metrics$)
      $position_{particle,current} \leftarrow$ corresponding position in $position\_matrix_{particle}$
      if $position_{particle,new}$ better than $position_{particle,current}$ **then**
        $position_{particle,current} \leftarrow position_{particle,new}$
      end if
    end for
  end while
  $position_{optimal} \leftarrow$ *position* holds the most amount
  return $position_{optimal}$
end function

5.4 Results

This section analyzes the results of the experiments, inspecting the performance of the proposed methods against the existing rivals. Particularly, the experimental outcome is discussed in three aspects with regarding the metrics in Section 5.2, including prediction accuracy, time consumption, and position matching.

5.4.1 Prediction Accuracy

Some references [156, 186] have demonstrated the one-step forecast of both SARIMA and LSTM provides a decent accuracy. However, the performance in multi-step prediction is not satisfactory, as in one-step forecast according to previous researches [166, 194].

The multi-step prediction usually leverages three common strategies: Serial Propagated (SP), Multiple Inputs Multiple Outputs (MIMO) and Multiple Inputs Single Output (MISO) [195]. In specific, these strategies have their advantages and disadvantages. SP model uses the predictive outputs at the previous moment as the additional inputs at the current one, which enhances the reliability of multi-step prediction but also undertakes the accumulation of small prediction error. MIMO structure provides a relatively low performance due to the complex weight interaction of the outputs. MISO architecture outperforms SP in the training phase; nonetheless, the accuracy lessens more aggressively than that of SP does when the time step of forecast increases.

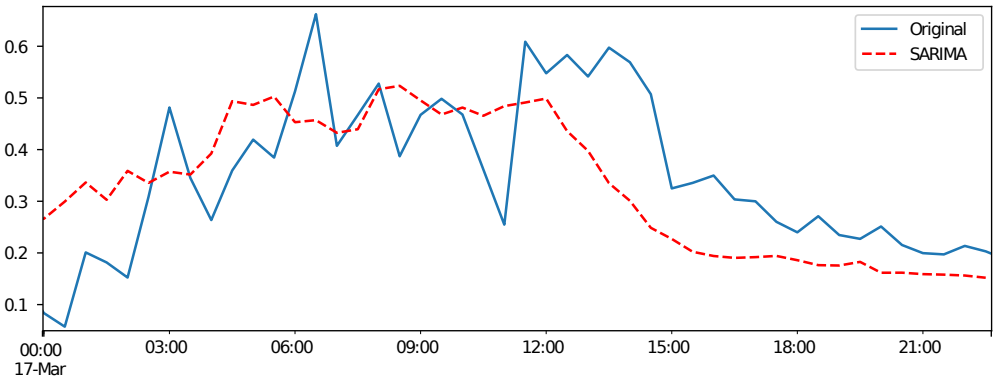


Figure 5.7: The MIMO performance of SARIMA in multi-step prediction.

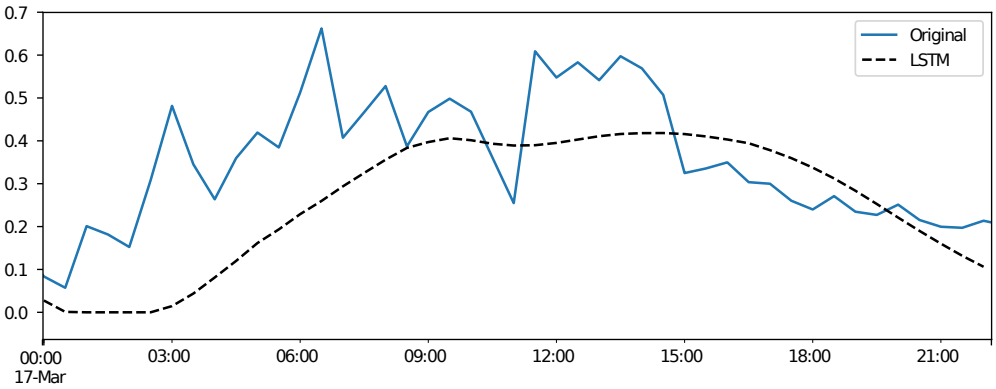


Figure 5.8: The SP performance of LSTM in multi-step prediction.

SARIMA predicts multiple points by applying the MIMO strategy, while LSTM forecasts the counterparts through the SP one. The experiments on multi-step prediction for both solely-used models SARIMA and LSTM are displayed in Figure 5.7 and 5.8 respectively, which also justifies the viewpoints in the existing literature.

The models compared in terms of prediction accuracy are analyzed as follows:

1. **SARIMA Model**. During the in-sample prediction experiment, this work constructs two separate ARIMA-based models based on different features of two datasets. In order to get the most fitting models, each element in the linear coefficient parameters $[p, d, q, P, D, Q]$ gets the values from 0 to 5, whose goal is to obtain the minimum AIC and BIC. Finally, the models $SARIMA(0, 1, 0)(1, 1, 2)_{48}$ in DARPA datasets and $ARIMA(2, 1, 2)$ in WIDE datasets attain the least values of the absolute sum of AIC and BIC, as exhibited in Table 5.1. Usually, the prediction model gets better accuracy when the absolute value of AIC trends to 0. Therefore, the models with the most appropriate parameters can become the baselines in their corresponding cases.

2. **LSTM Model**. The LSTM model chosen in this thesis is constructed with two

Table 5.1: AIC/BIC Values of SARIMA Model with Different Parameters

Non-seasonal(p, q)	**Seasonal(P, Q)**	**AIC**	**BIC**	**Absolute Sum**
(0, 0)	(1, 1)	-1471.4878	-1485.6010	2957.0888
(0, 0)	(1, 2)	**-1465.2695***	-1484.0872	**2949.3567***
(0, 0)	(2, 1)	-1486.6849	-1467.8672	2954.5521
(0, 0)	(2, 2)	-1484.9088	-1461.3867	**2946.2955***
(0, 1)	(1, 1)	-1649.9213	-1631.1037	3281.0250
(0, 1)	(1, 2)	-1648.7398	-1625.2178	3273.9576
(0, 1)	(2, 1)	-1652.8213	-1629.2993	3282.1206
(0, 1)	(2, 2)	-1651.0780	-1622.8515	3273.9295
(1, 0)	(1, 1)	-1588.3952	-1569.5775	3157.9727
(1, 0)	(1, 2)	-1586.9505	-1563.4284	3150.3789
(1, 0)	(2, 1)	-1589.1093	-1565.5872	3154.6965
(1, 0)	(2, 2)	-1587.2571	-1559.0306	3146.2877
(1, 1)	(1, 1)	-1679.6114	-1656.0893	3335.7007
(1, 1)	(1, 2)	-1678.6595	-1650.4330	3329.0925
(1, 1)	(2, 1)	-1680.2762	-1652.0498	3332.3260
(1, 1)	(2, 2)	-1678.1794	-1645.2485	3323.4279

hidden layers, which becomes a 4-layer LSTM. In order to minimize the loss of internal weights during the training process, the "adam" optimizer and the "tanh" activation function are selected. Different with other NN methods, there is a forget gate in LSTM, which helps to get rid of some "old" and "useless" values. The LSTM here chooses 0.2 as the dropout ratios.

As stated at the beginning of this section, the multi-step prediction of LSTM associated with SP strategy easily leads to an incredible error accumulation shown in Figure 5.8, which is a smooth curve that only depicts the trend of predicted values rather than matching the observation in details. After numbers of examinations, the model $LSTM(1, 2, 4, 1)$ is found out to outperform other models with different parameters of layers as illustrated in Table 5.2, which is thus used as another representative competitor.

3. **GRU Model**. The architecture of GRU is similar to that of LSTM but using GRU cells in the hidden layers instead. Since the structure of GRU cell is more straightforward than the counterpart of LSTM cell, we can assume that the processing time of GRU is less than that of LSTM. Nonetheless, what we concern is which model can obtain more accurate predicting performance.

4. **LSARIMA Model**. As described before, this hybrid model combines SARIMA model with LSTM model, where SARIMA predicted the raw forecast traffic data and LSTM predicted the expected residuals. After revising the raw prediction with the residuals, LSARIMA obtained the final output. The training parameters of LSARIMA were the same as those of LSTM while the linear coefficient parameters

Table 5.2: Comparison of LSTM Models (Out-of-Sample)

Layers	**Batch Size**	**Epoch**	**MAPE**
(1, 2, 2, 1)	8	100	37.1401
(1, 2, 2, 1)	8	1000	32.6759
(1, 2, 4, 1)	8	1000	**32.3808***
(1, 2, 10, 1)	16	1000	32.6279
(1, 4, 2, 1)	16	1000	33.2525
(1, 4, 4, 1)	16	1000	33.3332
(1, 4, 10, 1)	16	1000	32.9609
(1, 5, 10, 1)	8	1000	32.4687
(1, 5, 10, 1)	16	1000	35.9384
(1, 6, 10, 1)	8	1000	41.0154

were similar to those of SARIMA or ARIMA model aforementioned.

At first, the performance of LSTM and GRU are compared for the residual prediction, which explores the model with better performance. Figure 5.9 demonstrates LSTM displays a bit better accuracy than GRU because the cell construction of the former one is more complicated than that of the latter one, where the black curve represents the predicted data of the LSTM, and the red one stands for that of the GRU. In this case, the prediction provided by the LSTM model fits better with the original, being compared to that of GRU model. Therefore, the LSTM is selected as a component of LSARIMA, providing a relatively precise residual prediction so that it enhances the forecast results of the hybrid model.

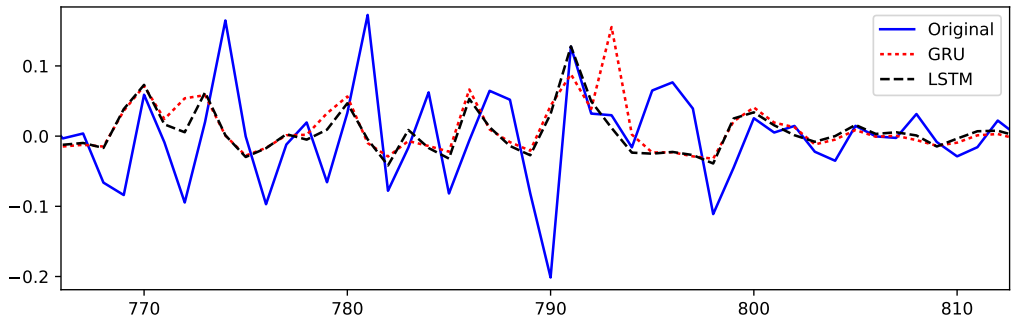


Figure 5.9: Residual MIMO prediction by LSTM & GRU.

The experimental results of prediction metrics are depicted in Table 5.3, which illustrates the proposed LSARIMA achieves an acceptable enhancement against other two models in both datasets. As mentioned in Subsection 5.4.1, single model neither LSTM nor SARIMA can get decent accuracy, where the error percentage is above 29% in MAPE, and the model match is under 0.5 in R2. To sum up, LSARIMA improves at least 31.74% in NMSE, 19.4% in NRMSE, 25.4% in MAPE, and 17.3% in R2. Although LSARIMA still

owns higher than 23% in MAPE and lower than 0.6 in R2, it exhibits the probability of accuracy improvement. In addition, Figure 5.10 and 5.11 show the forecasting results of network traffic predicted by the three models. The hybrid model LSARIMA fits the observation better than other two single models, especially the prediction between intervals 772 and 797 of DARPA datasets and intervals 675 and 705 of WIDE datasets.

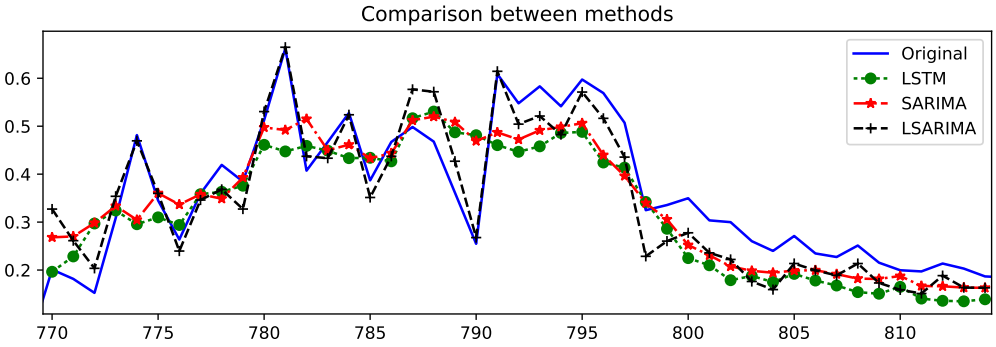


Figure 5.10: Prediction in DARPA datasets.

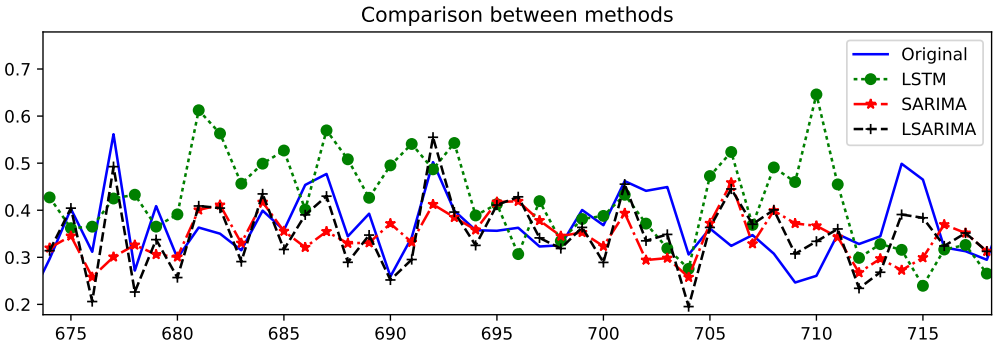


Figure 5.11: Prediction in WIDE datasets.

5.4.2 Time Consumption

In order to evaluate the processing efficiency of all the methods, we selected the K-Fat-Tree topology as the experimental scenario, where the number of core switches is set as the parameter within the region of [2, 19]. As mentioned in Subsection 5.1.3, the amount of switches develops effectively when the k value grows, thereby the total number of switches can range from 10 to 95. During this experiment, each set of parameter ran ten times and outputted the collections of latency and number of unreachability that contains the statistic of other switches that every single switch cannot reach.

Figures 5.12a and 5.12b illustrate the slope ratio of BF curve keeps a more ascending trend than those of other curves in both cases, which means the BF method consumes much more time than others do when the number of switches increases when being kept

Table 5.3: Prediction Metrics of Different Models

DARPA	**NMSE**	**NRMSE**	**MAPE(%)**	**R^2**
LSTM(1, 2, 4, 1)	0.0211	0.3807	32.3808	0.4446
SARIMA$(0,1,0)(1,1,2)_{48}$	0.0126	0.3190	31.5585	0.4690
LSARIMA	**0.0086**	**0.2571**	**23.5226**	**0.5672**
WIDE	**NMSE**	**NRMSE**	**MAPE(%)**	**R^2**
LSTM(1, 2, 4, 1)	0.0223	0.4449	33.8270	0.3656
ARIMA(2,1,2)	0.0118	0.3178	29.4275	0.4338
LSARIMA	**0.0089**	**0.2565**	**24.4192**	**0.5159**

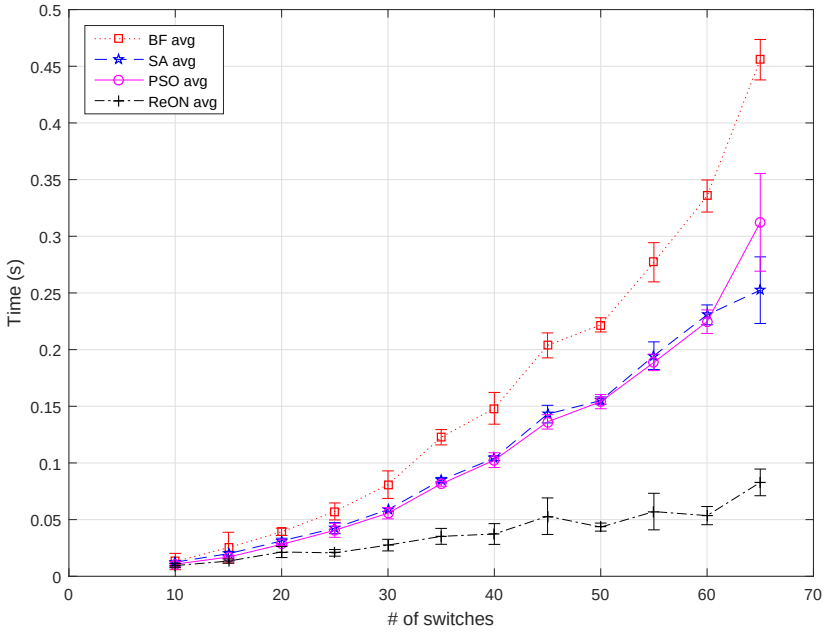
under 70. Nevertheless, the modified optimization method SA begins to dominate when there are more than 70 switches in the network graph, which seems not to get converged and approaches to the temperature threshold if the number of switches reaches the limit. As a result, the scalability of SA is inadequate in a network topology with large quantities of nodes, which raises the risk that the algorithm cannot get the optimal position.

However, in the average case, the time consumption of ReON scheme is always the most efficient among the CPP solutions thanks to the distributed resilience abilities.

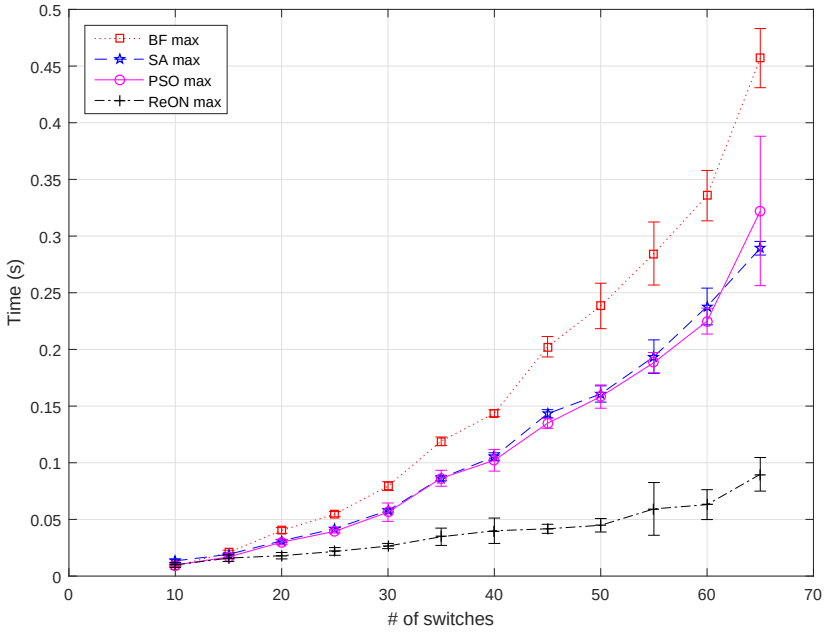
Additionally, we also consider the possible best and worst cases of the proposed method during the evaluation. Since the resilient abilities are distributed to the switches, the controller has to face two scenarios: 1) The best case that all the switches transmit the information of latencies and losses simultaneously to the controller. 2) The worst case that the asynchronous arrival of information occurs in the controller because the switches independently forward the message one by one.

In the former scenario, the total overhead of ReON is the sum of the processing time of the controller and the counterpart of the switch that costs most, which indicates the time complexity of ReON mainly depends on the complexity of the sorting algorithm. In this particular implementation for the analysis, the quicksort algorithm has been employed as the sorting strategy, which gives $O(n \log n)$ on average [196]. In the latter scenario, the total time consumption of ReON consists of the processing time of the controller and the accumulation of time of all switches; thus it becomes $O(n^2 \log n)$ on average.

From Figure 5.14a and 5.14b, which are composed of the error bar values of relevant algorithms, the outcomes of worst-case ReON and BF are comparable since the proposed algorithms are derived from the CPP approaches. The red polylines in both figures represent the performance of BF algorithm with average- and maximum-latency policies respectively while the blue ones denote the ReON performance in the worst case and the black ones denote the best case. Nonetheless, the overall ReON schemes perform better than the original BF solutions even in the worst case scenarios because ReON narrows the search scope in each iteration. Especially with the number of 95 switches, the results depict that the more partner switches exist, the better-related performance the scheme gains under the best condition. Moreover, the slope starts to become steeper when the number of switches

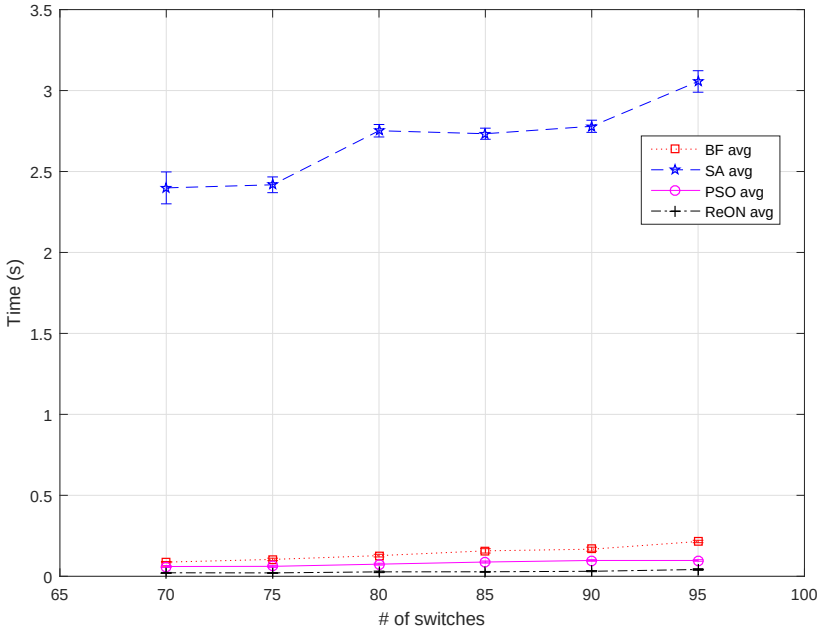


(a) Average latency

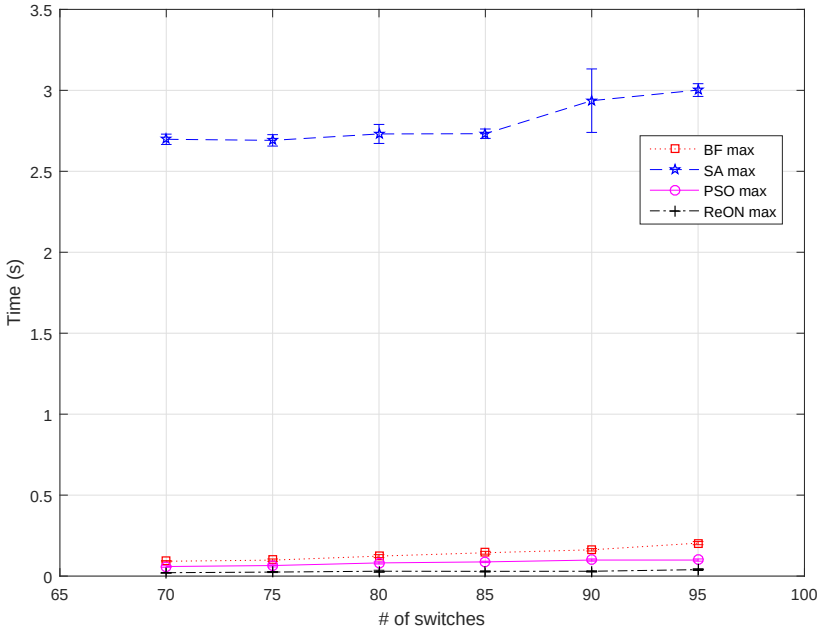


(b) Maximum latency

Figure 5.12: Comparison of time complexity between methods with less than 70 switches.



(a) Average latency

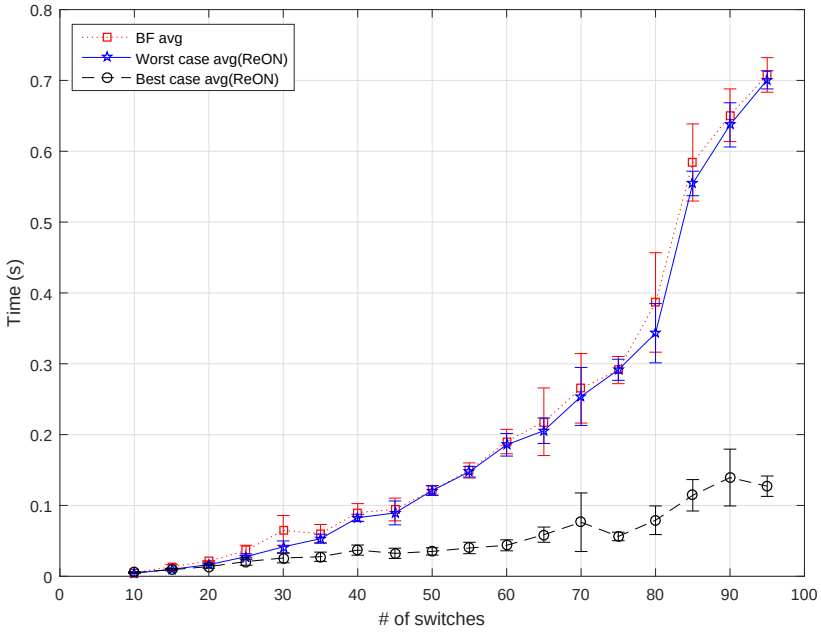


(b) Maximum latency

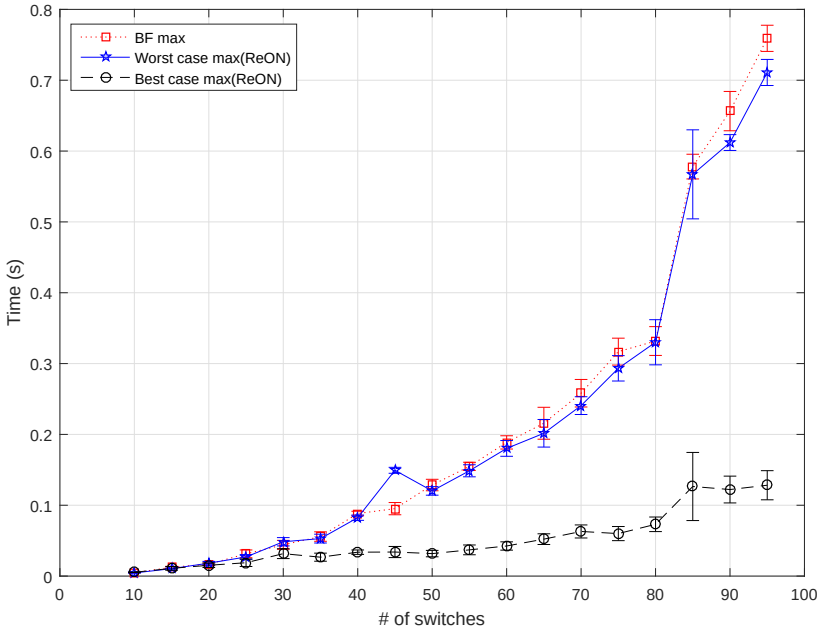
Figure 5.13: Comparison of time complexity between methods with more than 70 switches.

is located between 80 and 85. Therefore, the results depict the behavior where more partner switches exist in the area, which allows the scheme to reach better-related performance gains under the best condition. For instance, all switches collect latency data and analyze them in parallel before sending the preprocessed results to the controller, where the time consumption is calculated by the summation of the most significant overhead from a switch and the counterpart from the controller.

To see the vision clearly, Figure 5.15a and 5.15b are drawn based on the results from multiple tests, which represent the trends in average latency and the trends in maximum latency respectively. The boxes of figures denote the mean values with standard deviations. Additionally, the dispersed points, such as "+", "×", and "o", are the abnormal values in the data of each set. Moreover, the concave curves are the continuous results of corresponding methods mentioned above, which has justified the performance expectation, illustrating the advantage of distributed resilience ability. According to the outcomes, the dashed lines in the figures demonstrate that the ReON variants can obtain linear performance, costing less time than the original ones do. Furthermore, the trends of corresponding functions in Figure 5.15a and 5.15b have justified the performance expectation, illustrating the advantage of distributed resilience ability, where ReON costs less time.

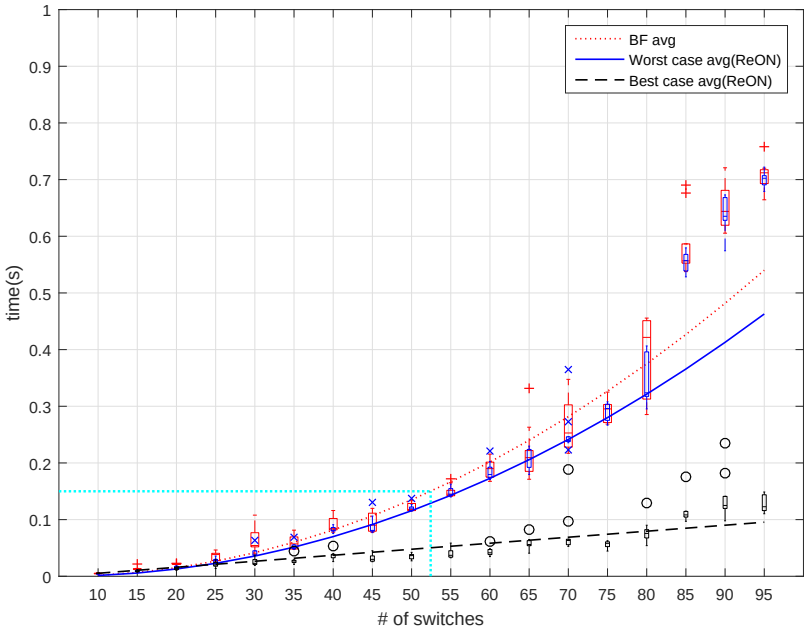


(a) Average latency

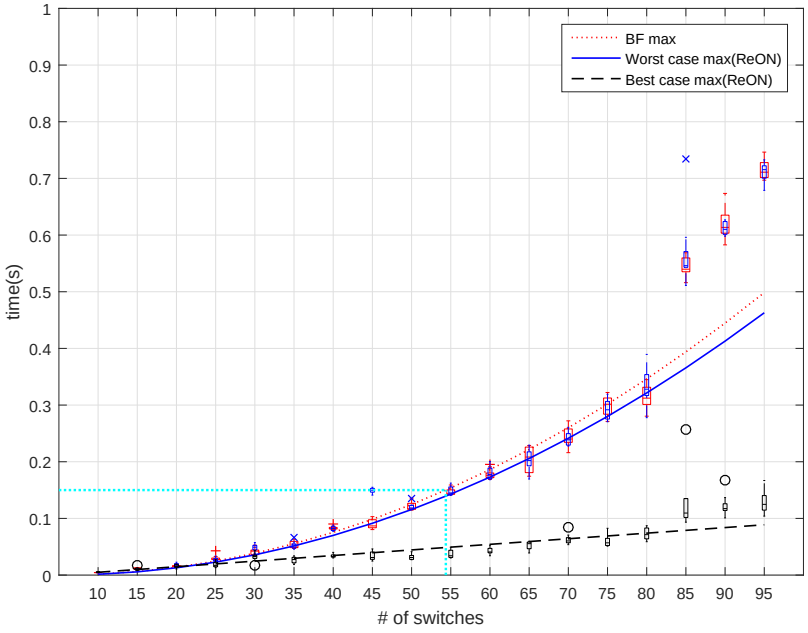


(b) Maximum latency

Figure 5.14: Comparison of time complexity between BF and ReON scheme.



(a) Trends in Average latency



(b) Trends in Maximum latency

Figure 5.15: Comparison of consumption trends between BF and ReON scheme.

5.4.3 Position Matching

During the experiment, the scenarios of OS3E topology have been run in 100 times, where the network graph is composed of 34 nodes and 43 edges. The baseline is the BF method associated with Dijkstra's algorithm (BF-Dij), which can provide the globally optimal positions among the evaluation datasets. In addition, the chosen candidates of ReON scheme and other methods are inspected against the baseline with two strategies that are average and maximum latencies. For illustrating the performance of the proposed method, three other competitors are introduced, including SA, PSO, and BF associated with Floyd-Warshall's algorithm (BF-FW) [169].

Figure 5.16 contains two y-axes, where the left one performs the accuracy axis, and the right one performs the axis of time consumption. The red bars in denote the accuracy results of corresponding methods by using the strategy of average latency, while the blue ones represent the accuracy results with the strategy of maximum latency. Moreover, the black curve in the figure serves as the mean values of the processing time of each algorithm. From this figure, it indicates the presented ReON scheme applying the maximum-latency strategy achieves better performance than that employing the average-latency policy, where the former one obtains around 95% accuracy while the latter one only reaches 80%. In contrast, the processing results of SA are in reverse, in which the performance of using average latency is slightly better than that of using maximum latency - 79% and 76%, respectively. Also, the PSO obtains 87% accuracy regarding the average-latency strategy and 85% accuracy concerning the maximum-latency one.

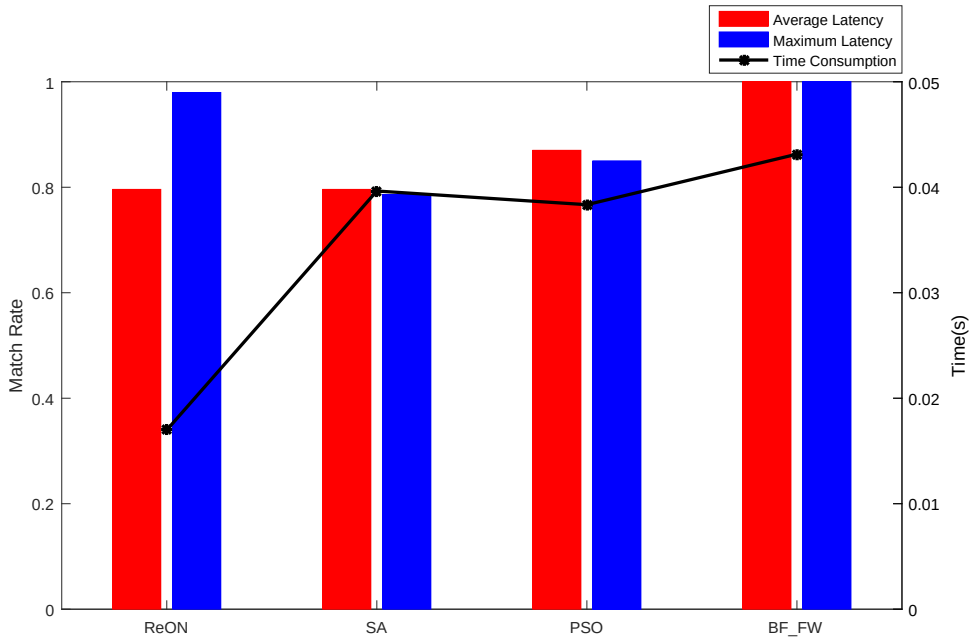


Figure 5.16: Comparison of accuracy between different methods.

According to the observations, it is noted that the proposed scheme considering average-

latency presents lower efficiency than that regarding maximum-latency. This behavior is a consequence of averaging the values, which consequently smooths and alleviates the impact of the worst case values. Thereby, it causes deviations on the outcomes. According to the results, BF-FW algorithm presented the best accuracy, reaching 100%. The decent prediction performance is originated from the fact that the method is based on brute force technique, which renders an optimal placement solution. Furthermore, the time complexity of BF-FW is $O(n^3)$, rendering high time complexity to reach a placement solution. Although BF-FW is more precise than other methods, its overhead is much higher than that of ReON, indicating that ReON scales up well, achieving similar placement efficiency.

To simplify and complement the obtained results, Figure 5.17 shows the distribution of backup controller positions with ReON and BF-Dij. As shown in the figure, the selections of BF and ReON are mostly located in Switch 7 and Switch 25 in terms of average latency even though accuracy is not high. Meanwhile, both outcomes of maximum latency are mainly situated in the Switch 29.

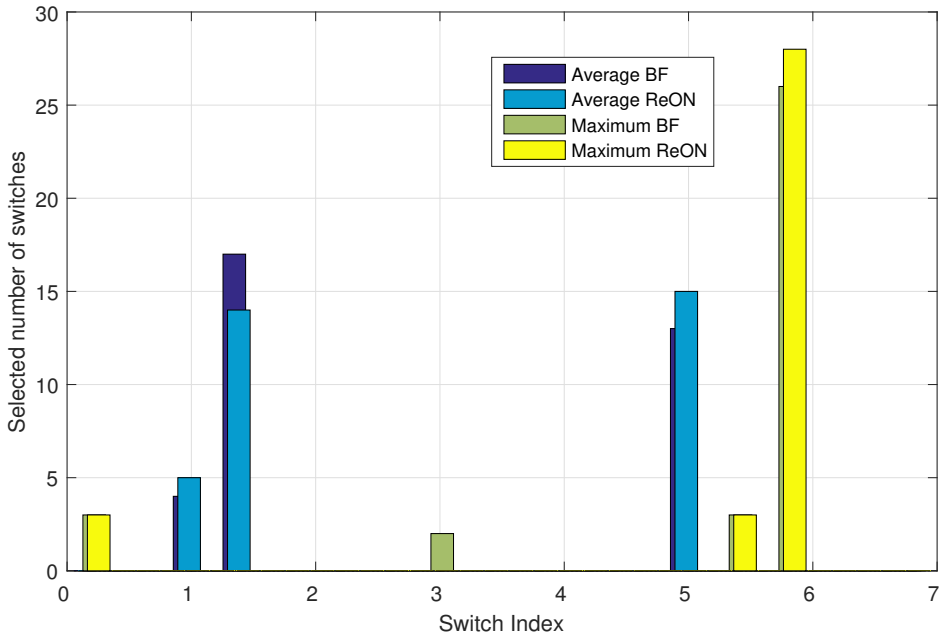


Figure 5.17: Distribution of backup positions.

In summary, ReON scheme improves the performance compared with the original CPP solution based on BF algorithm. The introduced scheme reduces the time consumption dramatically. Subsequently, it performs remarkably well by using maximum latency strategy. Even though the result of average latency method is not extremely precise, it remains in the generally selected backup positions. Moreover, neither accuracy nor processing time of SA method could provide better performance results when compared to the proposed approach.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, a generally resiliency-oriented SDN scheme called ReON is proposed, which abstracts both controller and switch services as individual VNFs and recombines them into the same industry-standardized machine. To provide a better quality of services, the controller placement is made through observing communication latency, which is a highly essential factor for the decision-making. The placement occurs by identifying an optimal position that owns the least latencies, which is more likely to lessen the risk of increasing propagation overhead. In addition, large overhead heavily impacts on the performance of a controller, leading to state inconsistency. Thus, this novel resilient method for the node detection in SDN can be analogous to the CPP solutions, which consists of two-layer resilience orchestration modules, whose characteristics can change when a controller fault occurs. The approach distributes the calculation to the switches rather than exclusively computing the placement in the controller. Therefore, it is feasible and useful when the presented applications are deployed in current cloud computing environments. Furthermore, a supplementary metric - the rate of unreachability - is presented to enhance the network reliability, and two resilience algorithms are designed to decrease the cost of heuristic-based approaches. The experimental results showed that the proposed method reduces time complexity in determining replacement controllers and achieves a decent performance in the case of maximum latency.

A hybrid model named LSARIMA is introduced, which predicts the network traffic with SARIMA and amends the raw prediction with forecast residuals by LSTM. A reasonable prediction model can construct a trustful baseline of detection. With this model, we attempt to improve the accuracy of multi-step traffic load predictions. By comparing the hybrid model with two individual models using two kinds of datasets, the results show LSARIMA improves the prediction performance at least 31.74% in NMSE, 19.4% in NRMSE, 25.4% in MAPE, and 17.3% in R^2.

6.2 Future Work

As future work, a deeper study based on this thesis will be conducted through analyzing the impact of placing multiple controllers in a diversified set of network topology. Through experiments, we have found it is possible to reduce the epoch of learning phase to improve the training speed of LSARIMA model. Also, different layer parameters and forget ratios can be adapted to increase the accuracy of multi-step prediction. These two aspects can help to improve the efficiency of building the failure detection baseline based on the hybrid prediction model.

Moreover, node failure detection can be utilized in multiple directions, such as controller recovery, and load balancing. Thus, it is interesting to investigate the redirection of network flow before the switch failure occurs. In this case, alleviating the jitter of false alarm is essential, which leads to study the cross-correlation between the network traffic of switches in order to enhance the accuracy of node failure detection in SDN environments.

References

[1] J. Lin, "Exploring software defined network.." [Online]. Available: http://tprc.tanet.edu.tw/tpnet2013/training/10210.pdf, 2013. Accessed on: January, 2018.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[3] S. Pillai, "Difference between hypervisor virtualization and container virtualization.." [Online]. Available: http://www.slashroot.in/difference-between-hypervisor-virtualization-and-container-virtualization, 2014. Accessed on: January, 2018.

[4] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, 2013.

[5] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De 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, 2016.

[6] S. S. Lee, K.-Y. Li, K. Y. Chan, G.-H. Lai, and Y. C. Chung, "Software-based fast failure recovery for resilient openflow networks," in *7th IEEE International Workshop on Reliable Networks Design and Modeling (RNDM)*, pp. 194–200, 2015.

[7] T. Watanabe, T. Omizo, T. Akiyama, and K. Iida, "Resilientflow: Deployments of distributed control channel maintenance modules to recover sdn from unexpected failures," in *Design of Reliable Communication Networks (DRCN), 2015 11th International Conference on the*, pp. 211–218, IEEE, 2015.

[8] E. Ramadan, H. Mekky, B. Dumba, and Z.-L. Zhang, "Adaptive resilient routing via preorders in sdn," in *Proceedings of the 4th ACM Workshop on Distributed Cloud Computing*, p. 5, 2016.

[9] X. Wang, C. Chen, P. Palacharla, M. Sekiya, S. Smolka, and N. Foster, "Spn os: Managing network services with virtual network objects," in *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pp. 149–155, 2015.

[10] R. Muñoz, R. Vilalta, R. Casellas, R. Martinez, T. Szyrkowiec, A. Autenrieth, V. López, and D. López, "Integrated sdn/nfv management and orchestration architecture for dynamic deployment of virtual sdn control instances for virtual tenant networks [invited]," *Journal of Optical Communications and Networking*, vol. 7, no. 11, pp. B62–B70, 2015.

[11] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient openflow-based networking," in *IEEE Network Operations and Management Symposium (NOMS)*, pp. 933–939, 2012.

[12] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks," in *9th IEEE International Conference on Network and Service Management (CNSM)*, pp. 18–25, 2013.

[13] Y. Chen, Q. Li, Y. Yang, Q. Li, Y. Jiang, and X. Xiao, "Towards adaptive elastic distributed software defined networking," in *34th IEEE International Performance on Computing and Communications Conference (IPCCC)*, pp. 1–8, 2015.

[14] T. Choi, B. Lee, S. Kang, S. Song, H. Park, S. Yoon, and S. Yang, "Iris-coman: Scalable and reliable control and management architecture for sdn-enabled large-scale networks," *Journal of Network and Systems Management*, vol. 23, no. 2, pp. 252–279, 2015.

[15] R. Sallehuddin, S. M. Shamsuddin, and S. Z. M. Hashim, "Hybridization model of linear and nonlinear time series data for forecasting," in *2nd IEEE Asia International Conference on Modeling & Simulation (AICMS)*, pp. 597–602, 2008.

[16] B. Lebiednik, A. Mangal, and N. Tiwari, "A survey and evaluation of data center network topologies," *arXiv preprint arXiv:1605.01701*, 2016.

[17] Internet2 Community, "Internet2 open science, scholarship and services exchange." [Online]. Available: http://www.internet2.edu/products-services/advanced-networking/layer-2-services/. Accessed on: January, 2018.

[18] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "Clickos and the art of network function virtualization," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, pp. 459–473, USENIX Association, 2014.

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

[20] A. Boukerche, *Handbook of algorithms for wireless networking and mobile computing*. Chapman and Hall/CRC, 2005.

[21] A. Boukerche, *Algorithms and protocols for wireless and mobile ad hoc networks*, vol. 77. John Wiley & Sons, 2008.

[22] A. Boukerche, "Algorithms and protocols for wireless sensor networks," *Hoboken: John Wiley & Sons*, 2009.

[23] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker, "Extending networking into the virtualization layer.," in *Hotnets*, 2009.

[24] C. Staff, "A purpose-built global network: Google's move to sdn," *Communications of the ACM*, vol. 59, no. 3, pp. 46–54, 2016.

[25] B. J. van Asten, N. L. van Adrichem, and F. A. Kuipers, "Scalability and resilience of software-defined networking: An overview," *arXiv preprint arXiv:1408.6760*, 2014.

[26] O. S. Brief, "Openflow-enabled sdn and network functions virtualization," *Open Netword Found*, 2014.

[27] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 655–685, 2016.

[28] A. Aissioui, A. Ksentini, and A. Gueroui, "An efficient elastic distributed sdn controller for follow-me cloud," in *11th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 876–881, 2015.

[29] E. Borcoci, R. Badea, S. G. Obreja, and M. Vochin, "On multi-controller placement optimization in software defined networking-based wans," *ICN*, p. 273, 2015.

[30] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed sdn controller," in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 7–12, 2013.

[31] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain sdn controllers," in *IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–4, 2014.

[32] S. S. Savas, M. Tornatore, M. F. Habib, P. Chowdhury, and B. Mukherjee, "Disaster-resilient control plane design and mapping in software-defined networks," in *16th IEEE International Conference on High Performance Switching and Routing (HPSR)*, pp. 1–6, 2015.

[33] R. Vilalta, A. Mayoral, R. Munoz, R. Casellas, and R. Martínez, "Multitenant transport networks with sdn/nfv," *Journal of Lightwave Technology*, vol. 34, no. 6, pp. 1509–1515, 2016.

[34] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the 1st ACM workshop on Hot topics in software defined networks*, pp. 7–12, 2012.

[35] H. A. Oliveira, E. F. Nakamura, A. A. Loureiro, and A. Boukerche, "Error analysis of localization systems for sensor networks," in *Proceedings of GIS'05*, pp. 71–78, ACM, 2005.

[36] A. Boukerche, H. A. Oliveira, E. F. Nakamura, and A. A. Loureiro, "Secure localization algorithms for wireless sensor networks," *IEEE Communications Magazine*, vol. 46, no. 4, 2008.

[37] H. A. B. F. De Oliveira, A. Boukerche, E. F. Nakamura, and A. A. F. Loureiro, "An efficient directed localization recursion protocol for wireless sensor networks," *IEEE Transactions on Computers*, vol. 58, no. 5, pp. 677–691, 2009.

[38] O. Abumansoor and A. Boukerche, "A secure cooperative approach for nonline-of-sight location verification in vanet," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 275–285, 2012.

[39] A. Boukerche, X. Fei, and R. B. Araujo, "An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange," *Computer Communications*, vol. 30, no. 14-15, pp. 2708–2720, 2007.

[40] A. Boukerche, C. Rezende, and R. W. Pazzi, "Improving neighbor localization in vehicular ad hoc networks to avoid overhead from periodic messages," in *Proceedings of GLOBECOM'09*, pp. 1–6, IEEE, 2009.

[41] K. Abrougui, A. Boukerche, and R. W. N. Pazzi, "Design and evaluation of context-aware and location-based service discovery protocols for vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 717–735, 2011.

[42] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[43] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[44] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 493–512, 2014.

[45] E. Haleplidis, D. Joachimpillai, J. H. Salim, D. Lopez, J. Martin, K. Pentikousis, S. Denazis, and O. Koufopavlou, "Forces applicability to sdn-enhanced nfv," in *3rd IEEE European Workshop on Software Defined Networks (EWSDN)*, pp. 43–48, 2014.

[46] M. Betts, S. Fratini, N. Davis, D. Hood, R. Dolin, M. Joshi, P. Doolan, and K. Lam, *SDN architecture*. Open Networking Foundation, 1 ed., 6 2014. ONF technical reference document.

[47] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[48] D. Erickson, "The beacon openflow controller," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 13–18, ACM, 2013.

[49] Project Floodlight, "Floodlight." [Online]. Available: http://www.projectfloodlight.org/floodlight/, 2012. Accessed on: January, 2018.

[50] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *15th IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–6, IEEE, 2014.

[51] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third ACM workshop on Hot topics in software defined networking*, pp. 1–6, 2014.

[52] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, "A comparison between several software defined networking controllers," in *12th IEEE International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS)*, pp. 223–226, 2015.

[53] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "Sdn controllers: A comparative study," in *18th IEEE Mediterranean Electrotechnical Conference (MELECON)*, pp. 1–6, 2016.

[54] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors," in *ACM SIGOPS Operating Systems Review*, vol. 41, pp. 275–287, ACM, 2007.

[55] A. Boukerche, A. Zarrad, and R. Araujo, "A cross-layer approach-based gnutella for collaborative virtual environments over mobile ad hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 7, pp. 911–924, 2010.

[56] R. W. Pazzi and A. Boukerche, "Propane: A progressive panorama streaming protocol to support interactive 3d virtual environment exploration on graphics-constrained devices," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 11, no. 1, p. 5, 2014.

[57] A. Boukerche and X. Fei, "A voronoi approach for coverage protocols in wireless sensor networks," in *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, pp. 5190–5194, IEEE, 2007.

[58] A. Boukerche and X. Fei, "A coverage-preserving scheme for wireless sensor network with irregular sensing range," *Ad hoc networks*, vol. 5, no. 8, pp. 1303–1316, 2007.

[59] C. G. Rezende, A. Boukerche, H. S. Ramos, and A. A. Loureiro, "A reactive and scalable unicast solution for video streaming over vanets.," *IEEE Trans. Computers*, vol. 64, no. 3, pp. 614–626, 2015.

[60] Z. Zhang, A. Boukerche, and R. Pazzi, "A novel multi-hop clustering scheme for vehicular ad-hoc networks," in *Proceedings of the 9th ACM international symposium on Mobility management and wireless access*, pp. 19–26, ACM, 2011.

[61] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba, "A novel solution for achieving anonymity in wireless ad hoc networks," in *Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pp. 30–38, ACM, 2004.

[62] B. McCouch, "Sdn network virtualization and nfv in a nutshell," *Network Computing*, 2014.

[63] A. Wang, M. Iyer, R. Dutta, G. N. Rouskas, and I. Baldine, "Network virtualization: Technologies, perspectives, and frontiers," *Journal of Lightwave Technology*, vol. 31, no. 4, pp. 523–537, 2013.

[64] M. Fuszner, "Gns3 graphical network simulator." [Online]. Available: https://www.csd.uoc.gr/~hy435/material/GNS3-0.5-tutorial.pdf, 2009. Accessed on: January, 2018.

[65] A. Boukerche and S. Rogers, "Gps query optimization in mobile and wireless networks," in *Proceedings of ISCC'01*, pp. 198–203, IEEE, 2001.

[66] R. B. Batista, A. Boukerche, and A. C. M. A. de Melo, "A parallel strategy for biological sequence alignment in restricted memory space," *Journal of Parallel and Distributed Computing*, vol. 68, no. 4, pp. 548–561, 2008.

[67] A. Bamis, A. Boukerche, I. Chatzigiannakis, and S. Nikoletseas, "A mobility aware protocol synthesis for efficient routing in ad hoc mobile networks," *Computer Networks*, vol. 52, no. 1, pp. 130–154, 2008.

[68] Z. Zhang, R. W. Pazzi, and A. Boukerche, "A mobility management scheme for wireless mesh networks based on a hybrid routing protocol," *Computer Networks*, vol. 54, no. 4, pp. 558–572, 2010.

[69] Y. Ren, R. Werner, N. Pazzi, and A. Boukerche, "Monitoring patients via a secure and mobile healthcare system," *IEEE Wireless Communications*, vol. 17, no. 1, 2010.

[70] R. W. Coutinho, A. Boukerche, L. F. Vieira, and A. A. Loureiro, "Geographic and opportunistic routing for underwater sensor networks," *IEEE Transactions on Computers*, vol. 65, no. 2, pp. 548–561, 2016.

[71] M. B. Younes and A. Boukerche, "Intelligent traffic light controlling algorithms using vehicular networks," *IEEE transactions on Vehicular Technology*, vol. 65, no. 8, pp. 5887–5899, 2016.

[72] R. W. Coutinho, A. Boukerche, L. F. Vieira, and A. A. Loureiro, "Design guidelines for opportunistic routing in underwater networks," *IEEE Communications Magazine*, vol. 54, no. 2, pp. 40–48, 2016.

[73] A. Boukerche and S. Samarah, "A novel algorithm for mining association rules in wireless ad hoc sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 7, pp. 865–877, 2008.

[74] A. Boukerche and C. Tropper, "A static partitioning and mapping algorithm for conservative parallel simulations," in *ACM SIGSIM Simulation Digest*, vol. 24, pp. 164–172, ACM, 1994.

[75] A. Boukerche and S. K. Das, "Dynamic load balancing strategies for conservative parallel simulations," in *Parallel and Distributed Simulation, 1997., Proceedings., 11th Workshop on*, pp. 20–28, IEEE, 1997.

[76] A. Boukerche, S. K. Das, A. Fabbri, and O. Yildiz, "Exploiting model independence for parallel pcs network simulation," in *Proceedings of PADS'99*, pp. 166–173, IEEE, 1999.

[77] A. Boukerche, S. K. Das, and A. Fabbri, "Swimnet: a scalable parallel simulation testbed for wireless and mobile networks," *Wireless Networks*, vol. 7, no. 5, pp. 467–486, 2001.

[78] A. Boukerche and L. Bononi, "Simulation and modeling of wireless, mobile, and ad hoc networks," *Mobile ad hoc networking*, pp. 373–409, 2004.

[79] ETSI Industry Specification Group (ISG) NFV, 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex - FRANCE, *ETSI GS NFV 001 V1.1.1: Network Function Virtualization. Use Cases*, 10 2013. A NFV document.

[80] ETSI Industry Specification Group (ISG) NFV, 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex - FRANCE, *ETSI GS NFV 002 V1.2.1: Network Functions Virtualization (NFV); Terminology for Main Concepts in NFV*, 12 2014. A NFV document.

[81] ETSI Industry Specification Group (ISG) NFV, 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex - FRANCE, *ETSI GS NFV-MAN 001 V1.1.1: Network Functions Virtualization (NFV); Management and Orchestration*, 10 2013. A NFV document.

[82] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[83] J. Metzler, "Sdn and nfv: Dynamic duo of next-gen networks?." [Online]. Available: http://www.networkcomputing.com/networking/sdn-nfv-dynamic-duo-next-gen-networks/1079608926, 2015. Accessed on: January, 2018.

[84] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, pp. 3–3, 2010.

[85] A. Gember-Jacobson, "Opennf." [Online]. Available: http://opennf.cs.wisc.edu/, 2014. Accessed on: January, 2018.

[86] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "Opennf: Enabling innovation in network function control," in *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 163–174, 2014.

[87] J. Vestin, A. Kassler, and J. Akerberg, "Resilient software defined networking for industrial control networks," in *10th International Conference on Information, Communications and Signal Processing (ICICS)*, pp. 1–5, 2015.

[88] P. Smith, A. Schaeffer-Filho, D. Hutchison, and A. Mauthe, "Management patterns: Sdn-enabled network resilience management," in *IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–9, 2014.

[89] A. Schaeffer-Filho, A. Mauthe, D. Hutchison, P. Smith, Y. Yu, and M. Fry, "Preset: A toolset for the evaluation of network resilience strategies," in *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pp. 202–209, 2013.

[90] X. Wang, Q. Zhang, I. Kim, P. Palacharla, and M. Sekiya, "Object-oriented network virtualization-an evolution from sdn to spn," in *Photonic Networks and Devices*, pp. NT1C–2, Optical Society of America, 2014.

[91] A. Boukerche, S. Hong, and T. Jacob, "An efficient synchronization scheme of multimedia streams in wireless and mobile systems," *IEEE transactions on Parallel and Distributed Systems*, vol. 13, no. 9, pp. 911–923, 2002.

[92] A. Boukerche and C. Dzermajko, "Performance evaluation of data distribution management strategies," *Concurrency and Computation: Practice and Experience*, vol. 16, no. 15, pp. 1545–1573, 2004.

[93] A. Boukerche and D. Turgut, "Secure time synchronization protocols for wireless sensor networks," *IEEE Wireless Communications*, vol. 14, no. 5, 2007.

[94] A. Boukerche, A. Roy, and N. Thomas, "Dynamic grid-based multicast group assignment in data distribution management," in *Proceedings of DS-RT'00*, p. 47, IEEE, 2000.

[95] A. Boukerche and Y. Ren, "A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks," in *Proceedings of PE-WASUN'08*, pp. 88–95, ACM, 2008.

[96] A. Boukerche and S. Nikoletseas, "Protocols for data propagation in wireless sensor networks," in *Wireless communications systems and networks*, pp. 23–51, Springer, 2004.

[97] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo, "Hpeq a hierarchical periodic, event-driven and query-based wireless sensor network protocol," in *null*, pp. 560–567, IEEE, 2005.

[98] R. W. Pazzi and A. Boukerche, "Mobile data collector strategy for delay-sensitive applications over wireless sensor networks," *Computer Communications*, vol. 31, no. 5, pp. 1028–1039, 2008.

[99] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013.

[100] M. Canini, P. Kuznetsov, D. Levin, and S. Schmid, "A distributed and robust sdn control plane for transactional network updates," in *IEEE conference on computer communications (INFOCOM)*, pp. 190–198, 2015.

[101] B. Genge and P. Haller, "A hierarchical control plane for software-defined networks-based industrial control systems," in *IEEE IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 73–81, 2016.

[102] A. Boukerche and C. Tropper, "A distributed graph algorithm for the detection of local cycles and knots," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 8, pp. 748–757, 1998.

[103] A. Boukerche, S. Hong, and T. Jacob, "A distributed algorithm for dynamic channel allocation," *Mobile Networks and Applications*, vol. 7, no. 2, pp. 115–126, 2002.

[104] A. Boukerche and A. Roy, "Dynamic grid-based approach to data distribution management," *Journal of Parallel and Distributed Computing*, vol. 62, no. 3, pp. 366–392, 2002.

[105] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba, "Sdar: a secure distributed anonymous routing protocol for wireless and mobile ad hoc networks," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pp. 618–624, IEEE, 2004.

[106] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba, "An efficient secure distributed anonymous routing protocol for mobile and wireless ad hoc networks," *computer communications*, vol. 28, no. 10, pp. 1193–1203, 2005.

[107] M. Elhadef, A. Boukerche, and H. Elkadiki, "Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols," in *Proceedings of the 4th ACM international workshop on Mobility management and wireless access*, pp. 18–27, ACM, 2006.

[108] M. Elhadef, A. Boukerche, and H. Elkadiki, "Performance analysis of a distributed comparison-based self-diagnosis protocol for wireless ad-hoc networks," in *Proceedings of MSWiM'06*, pp. 165–172, ACM, 2006.

[109] E. E. Ajaltouni, A. Boukerche, and M. Zhang, "An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure," in *Proceedings of DS-RT'08*, pp. 61–68, IEEE Computer Society, 2008.

[110] E. Fradinata, S. Suthummanon, N. Sirivongpaisal, and W. Suntiamorntuthq, "Ann, arima and ma timeseries model for forecasting in cement manufacturing industry: Case study at lafarge cement indonesiaaceh," in *IEEE International Conference on Advanced Informatics: Concept, Theory and Application (ICAICTA)*, pp. 39–44, 2014.

[111] A. Ozozen, G. Kayakutlu, M. Ketterer, and O. Kayalica, "A combined seasonal arima and ann model for improved results in electricity spot price forecasting: Case study in turkey," in *IEEE Portland International Conference on Management of Engineering and Technology (PICMET)*, pp. 2681–2690, 2016.

[112] T. Fischer and C. Krauß, "Deep learning with long short-term memory networks for financial market predictions," tech. rep., FAU Discussion Papers in Economics, 2017.

[113] S. I. Vagropoulos, G. Chouliaras, E. G. Kardakos, C. K. Simoglou, and A. G. Bakirtzis, "Comparison of sarimax, sarima, modified sarima and ann-based models for short-term pv generation forecasting," in *IEEE International Energy Conference (ENERGYCON)*, pp. 1–6, 2016.

[114] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, 2015.

[115] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic (extended version)," *IEEE/ACM Transactions on networking*, vol. 2, no. 1, pp. 1–15, 1994.

[116] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level," *IEEE/ACM Transactions on Networking (ToN)*, vol. 5, no. 1, pp. 71–86, 1997.

[117] A. Boukerche and M. S. M. A. Notare, "Behavior-based intrusion detection in mobile phone systems," *Journal of Parallel and Distributed Computing*, vol. 62, no. 9, pp. 1476–1490, 2002.

[118] A. Boukerche, K. R. L. Jucá, J. B. Sobral, and M. S. M. A. Notare, "An artificial immune based intrusion detection model for computer and telecommunication systems," *Parallel Computing*, vol. 30, no. 5-6, pp. 629–646, 2004.

[119] M. Elhadef, A. Boukerche, and H. Elkadiki, "A distributed fault identification protocol for wireless and mobile ad hoc networks," *Journal of Parallel and Distributed Computing*, vol. 68, no. 3, pp. 321–335, 2008.

[120] S. Samarah, M. Al-Hajri, and A. Boukerche, "A predictive energy-efficient technique to support object-tracking sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 2, pp. 656–663, 2011.

[121] A. Boukerche and X. Li, "An agent-based trust and reputation management scheme for wireless sensor networks," in *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, vol. 3, pp. 5–pp, IEEE, 2005.

[122] A. Boukerche, R. B. Machado, K. R. Jucá, J. B. M. Sobral, and M. S. Notare, "An agent based and biological inspired real-time intrusion detection and security model for computer network operations," *Computer Communications*, vol. 30, no. 13, pp. 2649–2660, 2007.

[123] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo, "A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications," in *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pp. 157–164, ACM, 2004.

[124] T. Antoniou, I. Chatzigiannakis, G. Mylonas, S. Nikoletseas, and A. Boukerche, "A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range," in *Proceedings of ANSS'04*, p. 43, IEEE Computer Society, 2004.

[125] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo, "Fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments," *Journal of Parallel and Distributed Computing*, vol. 66, no. 4, pp. 586–599, 2006.

[126] A. Boukerche, A. Martirosyan, and R. Pazzi, "An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks," *Mobile Networks and Applications*, vol. 13, no. 6, pp. 614–626, 2008.

[127] A. Boukerch, L. Xu, and K. El-Khatib, "Trust-based security for wireless ad hoc and sensor networks," *Computer Communications*, vol. 30, no. 11-12, pp. 2413–2427, 2007.

[128] Y. Ren and A. Boukerche, "Modeling and managing the trust for wireless and mobile ad hoc networks," in *Communications, 2008. ICC'08. IEEE International Conference on*, pp. 2129–2133, IEEE, 2008.

[129] A. Boukerche and Y. Ren, "A trust-based security system for ubiquitous and pervasive computing environments," *Computer Communications*, vol. 31, no. 18, pp. 4343–4351, 2008.

[130] A. Boukerche and Y. Ren, "A secure mobile healthcare system using trust-based multicast scheme," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 4, 2009.

[131] E. Bolshinsky and R. Friedman, "Traffic flow forecast survey," tech. rep., Computer Science Department, Technion, 2012.

[132] D. Bartholomew, "Time series analysis forecasting and control," *Journal of the Operational Research Society*, vol. 22, no. 2, pp. 199–201, 1971.

[133] N. Dengen *et al.*, "Comparison of sarima, narx and bpnn models in forecasting time series data of network traffic," in *2nd International Conference on Science in Information Technology (ICSITech)*, pp. 264–269, 2016.

[134] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results," *Journal of transportation engineering*, vol. 129, no. 6, pp. 664–672, 2003.

[135] Y. Shu, Z. Jin, L. Zhang, L. Wang, and O. W. Yang, "Traffic prediction using farima models," in *IEEE International Conference on Communications (ICC'99)*, vol. 2, pp. 891–895, 1999.

[136] S. Lee and D. Fambro, "Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1678, pp. 179–188, 1999.

[137] P. Wang, S.-Y. Zhang, and X.-J. Chen, "Sfarima: A new network traffic prediction algorithm," in *1st IEEE International Conference on Information Science and Engineering (ICISE)*, pp. 1859–1863, 2009.

[138] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N.-C. Yen, C. C. Tung, and H. H. Liu, "The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis," in *Proceedings of the Royal Society of London A: mathematical, physical and engineering sciences*, vol. 454, pp. 903–995, The Royal Society, 1998.

[139] Z. Wu and N. E. Huang, "Ensemble empirical mode decomposition: a noise-assisted data analysis method," *Advances in adaptive data analysis*, vol. 1, no. 01, pp. 1–41, 2009.

[140] D. Katsaros and Y. Manolopoulos, "Prediction in wireless networks by markov chains," *IEEE Wireless Communications*, vol. 16, no. 2, 2009.

[141] Y. Wei, J. Wang, C. Wang, and J. Wang, "Network traffic prediction by traffic decomposition," in *5th IEEE International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pp. 158–161, 2012.

[142] J. Long, K. Xueyuan, H. Huang, Q. Zhinian, and Y. Wang, "Study on the overfitting of the artificial neural network forecasting model," *Acta Meteorologica Sinica*, vol. 19, no. 2, p. 216, 2005.

[143] I. V. Tetko, D. J. Livingstone, and A. I. Luik, "Neural network studies. 1. comparison of overfitting and overtraining," *Journal of chemical information and computer sciences*, vol. 35, no. 5, pp. 826–833, 1995.

[144] X. Wang, C. Zhang, and S. Zhang, "Modified elman neural network and its application to network traffic prediction," in *2nd IEEE International Conference on Cloud Computing and Intelligent Systems (CCIS)*, vol. 2, pp. 629–633, 2012.

[145] H. Tan, Y. Wu, G. Feng, W. Wang, and B. Ran, "A new traffic prediction method based on dynamic tensor completion," *Procedia-Social and Behavioral Sciences*, vol. 96, pp. 2431–2442, 2013.

[146] H. Xie, H. Tang, and Y.-H. Liao, "Time series prediction based on narx neural networks: An advanced approach," in *IEEE International Conference on Machine Learning and Cybernetics*, vol. 3, pp. 1275–1279, 2009.

[147] S. Yi, T. Zhongda, Z. Lili, C. Xiaoqing, and S. Jian, "Networked control system time-delay prediction method based on genetic algorithm optimized extreme learning machine," in *IEEE Chinese Control and Decision Conference (CCDC)*, pp. 2331–2335, 2016.

[148] H. Shao and B. H. Soong, "Traffic flow prediction with long short-term memory networks (lstms)," in *IEEE Region 10 Conference (TENCON)*, pp. 2986–2989, 2016.

[149] K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *EMNLP*, 2014.

[150] S. Sun, C. Zhang, and G. Yu, "A bayesian network approach to traffic flow forecasting," *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 1, pp. 124–132, 2006.

[151] T. D. Nielsen and F. V. Jensen, *Bayesian networks and decision graphs*. Springer Science & Business Media, 2009.

[152] L. Zhang, Q. Liu, W. Yang, N. Wei, and D. Dong, "An improved k-nearest neighbor model for short-term traffic flow prediction," *Procedia-Social and Behavioral Sciences*, vol. 96, pp. 653–662, 2013.

[153] Z. Zheng and D. Su, "Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 143–157, 2014.

[154] M. M. Deza and E. Deza, "Encyclopedia of distances," in *Encyclopedia of Distances*, p. 94, Springer, 2009.

[155] W. Hu, L. Yan, K. Liu, and H. Wang, "A short-term traffic flow forecasting method based on the hybrid pso-svr," *Neural Processing Letters*, vol. 43, no. 1, pp. 155–172, 2016.

[156] F. Zhang, R. E. De Grande, and A. Boukerche, "Accuracy analysis of short-term traffic flow prediction models for vehicular clouds," in *Proceedings of the 13th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, pp. 19–26, 2016.

[157] L. Dai, W. Yang, S. Gao, Y. Xia, M. Zhu, and Z. Ji, "Emd-based multi-model prediction for network traffic in software-defined networks," in *11th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 539–544, 2014.

[158] M. Van Der Voort, M. Dougherty, and S. Watson, "Combining kohonen maps with arima time series models to forecast traffic flow," *Transportation Research Part C: Emerging Technologies*, vol. 4, no. 5, pp. 307–318, 1996.

[159] C. N. Babu and B. E. Reddy, "A moving-average filter based hybrid arima–ann model for forecasting time series data," *Applied Soft Computing*, vol. 23, pp. 27–38, 2014.

[160] M. Khashei and M. Bijari, "A novel hybridization of artificial neural networks and arima models for time series forecasting," *Applied Soft Computing*, vol. 11, no. 2, pp. 2664–2675, 2011.

[161] L. Wang, H. Zou, J. Su, L. Li, and S. Chaudhry, "An arima-ann hybrid model for time series forecasting," *Systems Research and Behavioral Science*, vol. 30, no. 3, pp. 244–259, 2013.

[162] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[163] Z. Jing and Z. Jin-fu, "Logistics amount forecasting based on combined arima and ann model," in *IEEE International Conference on Grey Systems and Intelligent Services (GSIS)*, pp. 594–597, 2009.

[164] R. K. T. Rathnayaka, D. Seneviratna, W. Jianguo, and H. I. Arumawadu, "A hybrid statistical approach for stock market forecasting based on artificial neural network and arima time series models," in *IEEE International Conference on Behavioral, Economic and Socio-cultural Computing (BESC)*, pp. 54–60, 2015.

[165] D. Zeng, J. Xu, J. Gu, L. Liu, and G. Xu, "Short term traffic flow prediction using hybrid arima and ann models," in *IEEE Workshop on Power Electronics and Intelligent Transportation System (PEITS'08)*, pp. 621–625, 2008.

[166] C. N. Babu and B. E. Reddy, "Performance comparison of four new arima-ann prediction models on internet traffic data," *Journal of Telecommunications and Information Technology*, no. 1, p. 67, 2015.

[167] M. Thottan and C. Ji, "Anomaly detection in ip networks," *IEEE Transactions on signal processing*, vol. 51, no. 8, pp. 2191–2204, 2003.

[168] M. Obadia, M. Bouet, J. Leguay, K. Phemius, and L. Iannone, "Failover mechanisms for distributed sdn controllers," in *IEEE International Conference and Workshop on the Network of the Future (NOF)*, pp. 1–6, 2014.

[169] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 2009.

[170] F. C. d. A. Pinto and M. C. Medeiros, "Misspecified neural network models and linear time series forecasting," *Learning and Nonlinear Models–Revista Brasileira de Redes Neurais*, vol. 1, no. 5, pp. 317–326, 2005.

[171] K. Aho, D. Derryberry, and T. Peterson, "Model selection for ecologists: the worldviews of aic and bic," *Ecology*, vol. 95, no. 3, pp. 631–636, 2014.

[172] X. Rao, C.-x. Dong, and S.-q. Yang, "Statistic learning and intrusion detection," *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pp. 576–576, 2003.

[173] J. Murai, A. Kato, H. Kusumoto, S. Yamaguchi, and T. Sato, "Construction of the widely integrated distributed environment," in *IEEE Region 10 Int. Conf. (TENCON)*, pp. 413–416, 1989.

[174] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pp. 1–6, 2009.

[175] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, 2015.

[176] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python," in *Proceedings of the 9th Python in Science Conference*, vol. 57, p. 61, SciPy society Austin, 2010.

[177] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, pp. 265–283, USENIX Association, 2016.

[178] Mininet Team, "Mininet." [Online]. Available: http://www.mininet.org/, 2011. Accessed on: January, 2018.

[179] Ryu Project Team, "Ryu sdn framework.." [Online]. Available: https://osrg.github.io/ryu-book/en/Ryubook.pdf, 2013. Accessed on: January, 2018.

[180] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in sdn-based core networks," in *25th IEEE International Teletraffic Congress (ITC)*, pp. 1–9, 2013.

[181] J. Barros, M. Araujo, and R. J. Rossetti, "Short-term real-time traffic prediction methods: a survey," in *IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pp. 132–139, 2015.

[182] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware controller placement for software-defined networks," in *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pp. 672–675, 2013.

[183] S. Kadry, A. Abdallah, and C. Joumaa, "On the optimization of dijkstras algorithm," in *Informatics in Control, Automation and Robotics*, pp. 393–397, Springer, 2011.

[184] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[185] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, vol. 3, pp. 189–194, 2000.

[186] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic flow prediction," in *IEEE Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 324–328, 2016.

[187] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[188] Cisco Corporation, "Class-based weighted fair queueing and weighted random early detection." [Online]. Available: http://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/fswfq26.html. Accessed on: January, 2018.

[189] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[190] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale sdn networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.

[191] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th IEEE International Symposium on Micro Machine and Human Science, 1995*, pp. 39–43, 1995.

[192] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Encyclopedia of machine learning*, pp. 760–766, Springer, 2011.

[193] Y.-J. Gong, J.-J. Li, Y. Zhou, Y. Li, H. S.-H. Chung, Y.-H. Shi, and J. Zhang, "Genetic learning particle swarm optimization," *IEEE transactions on cybernetics*, vol. 46, no. 10, pp. 2277–2290, 2016.

[194] Y. Seo, S. Kim, and V. P. Singh, "Multistep-ahead flood forecasting using wavelet and data-driven methods," *KSCE Journal of Civil Engineering*, vol. 19, no. 2, pp. 401–417, 2015.

[195] F.-J. Chang, Y.-M. Chiang, and L.-C. Chang, "Multi-step-ahead neural networks for flood forecasting," *Hydrological sciences journal*, vol. 52, no. 1, pp. 114–130, 2007.

[196] Y. B. Jmaa, K. M. Ali, D. Duvivier, M. B. Jemaa, and R. B. Atitallah, "An efficient hardware implementation of timsort and mergesort algorithms using high level synthesis," in *IEEE International Conference on High Performance Computing & Simulation (HPCS)*, pp. 580–587, 2017.