Accepted Manuscript

Software defined cloud: Survey, system and evaluation

Yaser Jararweh, Mahmoud Al-Ayyoub, Ala' Darabseh, Elhadj Benkhelifa, Mladen Vouk, Andy Rindos

PII: S0167-739X(15)00328-3
DOI: http://dx.doi.org/10.1016/j.future.2015.10.015
Reference: FUTURE 2878

To appear in: *Future Generation Computer Systems*

Received date: 15 November 2014
Revised date: 10 October 2015
Accepted date: 19 October 2015

Please cite this article as: Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk, A. Rindos, Software defined cloud: Survey, system and evaluation, *Future Generation Computer Systems* (2015), http://dx.doi.org/10.1016/j.future.2015.10.015

Software Defined Cloud: Survey, System and Evaluation

Yaser Jararweh[a], Mahmoud Al-Ayyoub[a], Ala' Darabseh[a], Elhadj Benkhelifa[b], Mladen Vouk[c], Andy Rindos[d]

[a] *Department of Computer Science, Jordan University of Science and Technology, 22110 Jordan, e-mail: yijararweh@just.edu.jo, afdarabseh12@cit.just.edu.jo, maalshbool@just.edu.jo*
[b] *Staffordshire University, Stafford. UK, e-mail: e.benkhelifa@staffs.ac.uk*
[c] *Department of Computer Science, North Carolina State University, Raleigh, North Carolina, USA, e-mail: vouk@csc.ncsu.edu*
[d] *IBM Corporation, Center for Advanced Studies, Research Triangle Park, North Carolina, USA. e-mail: rindos@us.ibm.com*

Abstract

Next generation cloud systems will require a paradigm shift in how they are constructed and managed. Conventional control and management platforms are facing considerable challenges regarding flexibility, dependability and security that next generation systems must handle. The cloud computing technology has already contributed in alleviating a number of the problems associated with resource allocation, utilization and management. However, many of the elements of a well-designed cloud environment remain "stiff" and hard to modify and adapt in an integrated fashion. This includes the underlying networking topologies, many aspects of the user control over IaaS, PaaS or SaaS layers, construction of XaaS services, provenance and meta-data collection, to mention but few. In many situations the problem may be due to inadequacy of service abstraction. Software Defined Systems (SDSys) is a concept that help abstract the actual hardware at different layers with software components; one classical example of this abstractions are hypervisors. Such abstraction provides an opportunity for system administrators to construct and manage their systems, more easily, through flexible software layers. SDSys is an umbrella for different software defined subsystems including Software Defined Networking (SDN), Software Defined Storage (SDStorage), Software Defined Servers (Virtualization), Software Defined Data Centers (SDDC), Software Defined Security (SDSec) etc. and ultimately Software Defined Clouds (SDCloud). Individual solutions and seamless integration of these different abstractions remains in many respects a challenge. In this paper, the authors introduce Software Defined Cloud (SDCloud), a novel software defined cloud management framework that integrates different software defined cloud components to handle complexities associated with cloud computing systems. The first part of paper presents, for the first time, an extensive state of the art critical review of different components of software defined systems, constructing the proposed SDCloud. The second part of the paper proposes the novel concept of SDCloud, which is implemented and evaluated for its feasibility, flexibility and potential superiority.

Keywords: Cloud Computing, Software Defined Systems, Software Defined Security, Software Defined Storage, Software Defined Networking, Management Complexities, Software Defined Cloud (SDCloud)

1. Introduction

With the rabid shift to the cloud computing paradigm, one of the most critical issues faced by system administrators is the construction and management of systems in a manner that eliminates or hides their complexity from the end users, and, at the same time, maintain their control, flexibility, dependability, and security. To achieve these goals, Software Defined Systems (SDSys) has emerged recently to address these control and management challenges that exist in traditional platforms. The cost of management and administration operations are very high compared to other system operations in today's computing systems [1]. Thus, adopting the SDSys paradigm to cut the management and administration cost has become very appealing.

The cloud computing technology has already contributed in alleviating a number of the problems associated with resource allocation, utilization and management. However, many of the elements of a well-designed cloud environment remain "stiff" and hard to modify and adapt in an integrated fashion. This includes the underlying networking topologies, many aspects of the user control over IaaS, PaaS or SaaS layers, construction of XaaS services, provenance and meta-data collection, to mention but few. SDSys comes to supplement

the cloud computing technologies by addressing the way to efficiently relax this stiffness and integrate the elements that are still hard to modify or adapt in cloud computing. SDSys is a concept that help abstract the actual hardware at different layers with software components. Such abstraction provides system administrators with the ability to construct and handle all control and management decisions by a central decision maker, through flexible software layers. For example, one of the most important features of cloud computing is providing "on-demand" services; however, this is challenging to achieve in a decentralized control unit, where every component manages itself and has no information about other components. The request for adding a new decision or make some enhancements on the system requires knowing information about the other components of the system. Such information will have to be collected and analyzed in a single unit. Consequently, there will be some delay in response, which contrasts with the need for timely on-demand response. The problem of combining the achievement of multiple objectives with fast responses required by on-demand services in a single system is what SDSys is set to achieve[2, 3].

SDSys is an umbrella for different software defined subsystems including Software Defined Network (SDN), Software Defined Security (SDSec), Software Defined Storage (SDStorage), Software Defined Data Center (SDDC)[4], Software Defined Infrastructure (SDI), Software Defined Management (SDM), Software Defined Compute (SDCompute), Software Defined Server (SDSer), Software Defined Internet of Things (SDIoT) [5], Software Defined Radio (SDR), and Software Defined Enterprise (SDEn). The software layer can control different types of hardware devices in various contexts giving rise to the term Software Defined everything or anything (SDx).[1] Many advantages can be driven from SDSys such as increasing the performance, scalability, and security of the system to facilitate resource management.

SDSys provide the ability to control a wide range of computing resources in a work-flow driven and dynamic fashion by separating the control layer from the data work flow layer, i.e., isolating the control from hardware devices and setting it in a software layer. The main idea behind the SDSys is to build an orchestrated system, controller, to handle the control for all independent devices by using standard and general protocols.[2] This is achieved by means of virtualization. As virtualization is considered to be a key concept of the cloud computing technology, it also plays an essential role in SDSys. Virtualization creates a virtual platform for different devices or system components (like network, OS and storage devices) that emulates the characteristics of the real devices. The transformation from hardware control to software centric control is, therefore, achieved through virtualization, in which the functionalities of a single or multiple systems can be abstracted allowing the integration of the benefits of these systems into a multiple purpose function. as an illustration, consider the Software Defined Data Center (SDDC) example, which integrates a set of servers, storage devices, and networks into a single comprehensive system or resource pools.[3]

There are many reasons why the world is focusing so much on virtualization. The most important ones are as follows.

1. Resource sharing: If we have idle resources that exceed our need then we can increase the resource utilization by splitting them among several virtual machines. The resources can be anything like storage, disks or even links.
2. User isolation: In certain situations, there is a need to keep a level of confidentiality between the users. Using virtualization, each user may have its own VM, which is separate from other users' VMs.
3. Resources aggregation: If there is a need to perform a task that requires resources beyond what we have, then we can use the virtualization to build suitable and useful virtual resources large enough to complete the task.
4. Dynamic management and control: The users requirements are changing frequently which makes the response for these changes in virtualization easier than when dealing directly with physical resources.
5. Simplify the management: Manage devices through a software layer is easier than manage them in a physical layer.

It is the software defined concept and other related concepts such as "software deployed." In the former, the APIs and software are used to control and mange resources and devices. On the other hand, the software deployed concept means that the functionality of the service is deployed in a computer hardware object. Using the software to manage and control the resources is not a new concept. The essential difference, which was brought by the software defined concept is the ability of the control layer to control all the underlying

[1] http://www.memorableurl.com/2013/12/software-defined-anything.html
[2] http://www.meetup.com/MESS-LA/events/147433842/
[3] http://www.tvtechnology.com/from-the-editors/0145/software-defined-systems--virtualization/216662

resources regardless of their vendor variations by physically isolating them from the hardware resources in the data layer [6]. The concept of abstraction in SDSys is similar to the idea of Object Oriented (OO), where the implementation is separated from the interface representing the data layer and the control layer respectively in SDSys. The reason behind this separation is to simplify the modification process, since any change in the implementation will not affect the interface and vice versa [6].

To recall, implementing SDSys solutions for cloud computing is currently very fragmented and is still a growing research and development project, despite the its proven advantages and wider acceptance amongst specialists from academia and industry. The aim to achieve a fully integrated SDSys cloud computing, combining all aspects of software defined systems, is still far from being realised. This ultimate system, is referred to as Software Defined Cloud (SDCloud); a concept which was first introduced in the Harmony system [3]. Harmony proposed to integrate and manage the main components of any computing system such as the computing resources, storage resources and networks into one platform that represents the SDCloud concept. To the best of the authors' knowledge, the main attempt in literature to address a practical implementation of SDCloud is reported in [2] who presented an architecture for Software Defined Clouds for data centers. They consider different cloud applications and services, focusing on mobile cloud applications. However, the proposed architecture focuses mainly on SDN, Software-Defined Middleboxes Networking and Network Virtualization.

The proposed system in this paper is an attempt to further expand the SDSys capabilities in order to implement the ultimate aim of SDCloud, by integrating several aspects of SDSys, working cohesively together. In this paper, we focus mainly on the integration of Software Defined Networking (SDN), Software Defined Compute (SDCompute), Software Defined Storage (SDStorage), and software defined security (SDSec, which makes it, the most complex implementation of SDCloud, yet to be reported. SDCompute is considered as a core components of any SDCloud [7, 8, 9, 10, 11, 12]. SDCompute is the first step towards realising a complete SDSys which represent the virtualization of the physical server (nodes) into a set of virtual resources using one of virtualization technologies like Xen. The proposed system is implemented and evaluated by introducing major extensions to the Mininet simulation tool which is an open source tool introduced for SDN [13]. Since research in SDSys is still a growing and very fragmented area, it is worthwhile presenting a comprehensive review of the selected software defined subsystems, more specifically, SDN, SDStorage, and SDSec. This review is also a current gap in literature; it combines efforts from both academia and industry. Therefore, presenting this critical review in the paper is another worthwhile contribution. Due to the lack of academic published research in SDSys, and since there are more developments in SDSys by industry, the review included references to industrial white papers/pages and experts blogs, to ensure coverage.

The rest of this paper is structured as follows. Sections 2 to 4 provide state of the art work on the selected software defined subsystems, SDN, SDStorage, and SDSec respectively. Section 5 presents and discusses the proposed SDCloud architecture design and prototype integrating the aforementioned software defined subsystems. The proposed system prototype and experimental evaluation is the presented in Section 6. Finally, we conclude and present our future plans in Section 7.

2. Software Defined Network (SDN)

Software Defined Network (SDN), is considered to be the most popular form of SDSys, and it is much more researched than other forms of SDSys. As with any SDSys, SDN is also based on virtualization where the network resources are virtualized [14]. Virtualization of network resources is not a new concept. For example, in different types of multiple access schemes, channels are "virtualized" to allow multiple users to share the same physical channel. The same thing applies to the virtual local area network (VLAN), where the physical LAN is shared between different parts of the company. What is more interesting and relevant is the virtualization of the network functions where different network functions can be combined by using standard protocols. A European group called Network Function Virtualization (NFV) was established to enable and provide standards for network functions virtualization [14].

SDN is the latest innovation in computer networks. As any software defined system, SDN simplifies the network management by separating the control plane from the data plane. In network communication, the messages created by the user present the data plane. These messages need to be transferred to an appropriate destination. The network management component is responsible for finding the best path to send the message by using the control messages, and maintain any information related to the network like traffic and congestion. In SDN, data plane and control plane are separated. The data plane uses the forwarding tables prepared by the control plane in the controller to forward the messages, flow-packets [14].

The control plane in SDN is centralized, where the data plane is still distributed. Maintaining a centralized control plane speeds decision-making process, where the policy controls are dynamically changed

without the need to go through a long path to make the decision. In [14], the authors discussed how SDN work in case the centralized controller breaks down by some events, where another standby controller is used to control the network operations. In addition, control in SDN becomes programmable, where the control program can be easily changed when needed. With programmable control plane, the network can be divided into several virtual networks which are located on the same shared physical hardware, and every virtual network has a policy different from the other virtual networks [14]. Figure 1 shows an abstract view of the SDN architecture.

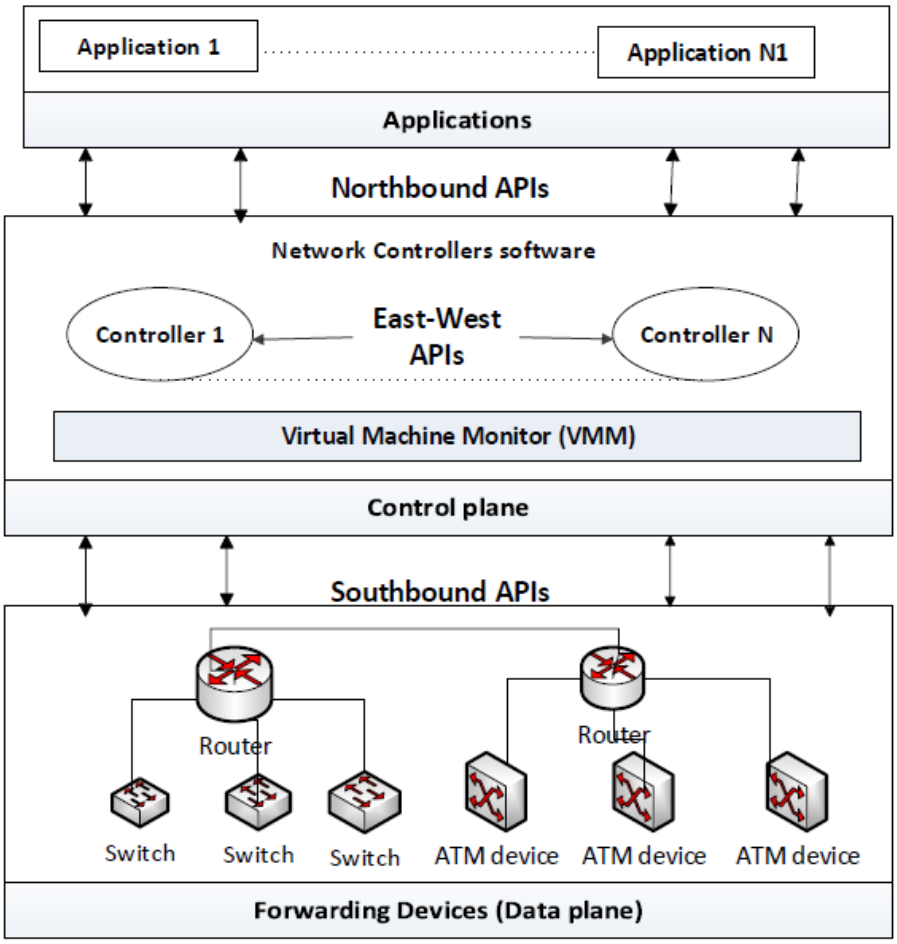


Figure 1: Software Defined Network (SDN) Architecture Design

The SDN architecture shown in Figure 1 illustrates how the control plane is surrounded by a group of APIs: southbound, northbound, and east-west APIs. The control plane uses southbound APIs to communicate with hardware residing in the data plane. The most widely known one is the OpenFlow API [15, 16]. The northbound APIs are used to communicate with the network applications. The east-west APIs are used to facilitate the communications between the controllers within the same domain or across neighboring domains. The SDN control plane is divided into two main layers; control and hypervisor layers. The former consists of a set of controllers. Floodlight and OpenDaylight are two examples of these controllers [15, 16]. Furthermore, there are many existing controllers installed to control the network communication. The hypervisor layer, which is called FlowVisor, resides between a set of controllers, the control layer, and the hardware assets, which is used as a hidden proxy between them. It is worth mentioning that the network application development is limited because the northbound APIs are not standardized yet [14]. The authors in [15] discussed two different architectures for SDN, OpenFlow and ForCES. The differences between them revolve around the system architecture, design, forwarding models, and interface protocols.

ForCES (Forwarding and Control Elements Separation) redefines the architecture of the network device. In the ForCES approach, the control plane and data plane are separated, but are located close together within the same physical network device. The approach has two logical elements: Control Elements (CE) and Forwarding Elements (FE), which are needed to allow the ForCES protocol to communicate. FE handles

each packet in the network by using under layer hardware, where the control functions are executed by the CE to define the way in which the FE should handle the packet. The Logical Function Block (LFB), which is located on FE is the most important building block in ForCEF system architecture. The LFB facilitate the CEs work to control the FEs design.

In contrast to the ForCEF, where the LFB is used to communicate and mange all the control between the FEs and CEs, the OpenFlow architecture depends on flow of packets between the controller in the control layer and the forward switch in the data plane, where the information and data exchanged between them are standardized. OpenFlow architecture consists of a forwarding switch and a controller. The switch combines a set of flow tables and has a secure abstracted layer to communicate with the controller via the OpenFlow protocol. The flow tables keep the information needed to forward and handle the packets such as rules to match received packets as set by the controller. According to the information extracted from the header of the received packet and the match rules in the flow table, a set of actions are executed. If there is no match found between the extracted information and any rules in the flow table entries, then the "table-miss entry" is applied [15].

The forwarding process in SDN requires special features in the forwarding devices that do not exist in traditional switches. E.g., the forwarding switches may store up to tens of thousands of forwarding rules, which requires a high computational power and large memory space. This is related to how scalable the network can be. A lot of proposed solutions like Devoflow, DIFANE, and Palette switches were discussed to cover the memory limitation bottleneck of the forwarding switches in the OpenFlow and make it somewhat scalable if there is a need to expand the forwarding rules in the switches[15, 16]. Other challenging features include network throughput and the speed of the packet handling process, which can especially important for certain types of networks such as critical networks [15].

Residing between the network applications and the forwarding elements, the controller serves as a middleware to facilitate the communication between these entities. Moreover, the controller is responsible for the control and management operations, which creates many challenges such as the controller's ability to efficiently handle a large number of new flow requests, which directly affects the network performance. Ethane, NOX-MT, Maestro, Beacon, and McNettle are examples of existing controllers designed with this issue in mind [15, 16] . Nevertheless, there are many performance issues not addressed like the intensive use of the dynamic memory allocation, and redundancy problem in memory location. Several methods were discussed by the authors of [16] to enhance the controllers performance. One of these methods is adapting multi-level, hierarchical controllers to address the scalability issue in SDN controllers, and distribute the controllers in an efficient way. The controllers need to communicate with the underlying forwarding devices frequently, which consumes their processing power and pose a new challenge to SDN. Another challenge is related to the controllers placement problem, which is concerned with the optimal number and locations of the controllers. Related to it is the type of link between the controller and the forwarding devices also has an effect on network latency.

Several issues have to be taken into account while designing SDN controllers such as to whether they are centralized, distributed or a hybrid of these two approaches. Current OpenFlow solutions assume all the management operations occur in a centralized control entity, which creates a single point of failure. While this might be suitable for small scale networks, it may not suitable for large scale or scattered networks. To handle such scenarios, a distributed control plane with several controllers is used. However, such solutions create other challenges such as synchronization between the different controllers. [16] discussed how Hyperflow [17] solved this issue by creating a logical control plane from physically distributed controllers. Furthermore, in multi-domain environments, the control process becomes more difficult due to heterogeneity of the underlying infrastructure. A distributed SDN control plane (DISCO) [18] was proposed to deal with these issues in an expeditious manner without the need for expensive human intervention. Another related point is control granularity which is concerned with whether the controller needs to be consulted for each packet, for the first packet of each "flow" of packets or even the first packet of an aggregated flows of packets. The last point is wether the policy enforcement strategy is reactive (e.g., Ethane) or proactive (e.g., DIFANE) [15].

Virtualization is considered as an efficient way to improve resources utilization by sharing the physical resources between several users. Furthermore, the frequent changes in the network require fast and dynamic responses. Virtualization can address this issue in an effective manner. FlowVisor plays a key role in SDN virtualization as a software layer between the controllers and network switches. Designing the virtualized system will reflect on SDN performance too. Several systems work to address the virtualization problems and limitations. They try to either enhance the SDN controller or FlowVisor, or add some extra layers to enhance the performance. Flexibility of the control layer which measures the scalability of the network, right network management, and keep the level of isolation between all different users are considered a

basic requirement for SDN virtualization. Different systems and solutions addressed in [16] which achieve these requirements like FlowN [19], Integrated system [20], MAC addressing system [21], EHU-OEF [22], Adaptive isolation system [23], LibNetVirt [24], etc. The isolation may come in different levels: isolation at the interface level, processing level, or even memory level. One important point that must be mentioned is how the isolation level influences the system latency. There is a trade-off: better isolation level leads to higher system latency.

A number research work and surveys have been published on SDN. In [25], the authors discussed most of SDN aspects and illustrate how this paradigm can support the Software Defined Environments (SDE). In addition, they show the IBM vision to consolidate the SDN idea by integrating their IBM SDN virtual environments (SDN-VE) product with the Neutron, OpenStack network platform [26] to extend SDN-VE feathers.

In other related published research [15, 16], the authors covered most SDN/OpenFlow aspects ranging from abstract concepts to solution deployment. The authors highlighted the motivations behind SDN by showing how SDN are built to support several organizations by facilitating the control management operations and enhancing its performance with lower cost. Moreover, they showed how distributing workload across virtualized hardware components can improve performance. Furthermore, they discussed how cloud computing providers can exploits SDN benefits to overcome issues related to the heterogeneity of switches/routers infrastructure. Different vendors have different switch characteristics. So, instead of managing and customizing each switch separately, SDN provide the ability to manage all switches by a single enforcement point, SDN control layer. Also it gives the cloud user the ability to use the cloud resources in an efficient way by creating slices/slivers and let the data to flow in transparency way.

The authors of [27] proposed a layered taxonomy survey about SDN. They showed an overview about the SDN and discussed its root and architecture as brought by the authors in [15]. In addition, they provided a brief summary of some OpenFlow forwarding device switches, and controllers. They defined several related aspects about it similar to: the switch/controller sources code language and if it is open source or not, which version of OpenFlow supports, also if the controllers used multi-threading mode or not, and which one has a GUI interface. Moreover, they list some SDN programming language platforms and compared between them in many respects. As an example the level of abstraction: high or low. Policy logic: active vs passive. Which ones support query language. The Type of the programming: Functional Reactive programming (FRP) or logical programming (LP), and other.

The first issue related to the acceptance of SDN in practical solutions is the hardware support. While OpenFlow requires essential changes in hardware architecture, the same cannot be said about other non-OpenFlow standards like ForCES [28]. The second issue is the existence of an easy-to-use network programming languages and tools. Several SDN programming tools were proposed to address these issues such as NetCore [29], Frenetic [30] and FlowVisor [31]. However, without an efficient way of performing the computationally intensive management and control tasks, SDN can never be adopted by the industry. In [16], the authors introduced a runtime system based on Glasgow Haskell Compiler (GHC) [32] as a solution to improve the performance by better controlling the number of system-calls and interrupts through the system, the allocation and de-allocation of memory locations, etc. Another advanced method to speed up the controller and enhance its performance is Control-Message Quenching (CMQ) [33]. The Palette scheme [34] follows a graph theoretic approach to split and distribute the large sized flow table over the entire SDN system. The authors of [35] proposed BalanceFlow, in which each controller sends its load information to the super controller allowing it to distribute the load in a fair way.

There are other factors, which can also effect SDN performance, such as Quality of Service (QoS). Many solutions were proposed to handle the QoS issues in SDN/OpenFlow systems. Open-QoS[36, 37] is considered one typical solution for these issues. Different aspects can each have an important role in SDN systems performance. Some of these aspects are mentioned in [16] in details including: the architecture of the controller, network OS, etc. Moreover, extra advance strategies were addressed to support the OpenFlow-based SDN systems like, Iterative Parallel Grouping Algorithm (IPGA) [38]. IPGA was proposed to handle the scheduling of flow prioritization management issue like a linear binary optimization problem.

The last issue related to practical deployment of SDN systems is the security factor. The new architecture of the SDN system derives and increases additional security targets by the attackers. The hypervisor virtualization layer, the SDN controllers, the OpenFlow protocol all of them are considered security holes which require an efficient security model to keep it safe. Cloud Security Alliance (CSA) survey and Network Intrusion Detection and Countermeasure Selection Strategy (NICE) [39] are two works talking about security concerns in virtualized network and cloud computing environments. Different detection algorithms and proposed security solutions are discussed in details as well. Classical security techniques will not work

probably to keep the SDN secure, since with the new system design there is a need to deploy new security solution which is able to guarantee the system secure and at the same time does not effect on network efficiency. A database DB model was proposed as security solution for SDN systems which called FRESCO-DB [40]. We will talk in details about security solution schemes for software defined systems later on in this paper.

SDN/OpenFlow can work well in multiple environments due to its flexibility, programmability, portability, and reliability, since it reduces the management overhead and the system complexity. The heterogeneity of the wireless network infrastructure needs a solution like SDN to decrease the system re-configuration and management operations of different underlying components. Two types of wireless network: Wireless sensor network and wireless mesh network introduced in [16] as examples of how the SDN/OpenFlow can be adapted in wireless networks. Flow sensor is a new concept in virtualized WSN. It stores the flow tables of the network and easy the wireless communication between different sensors. Such network vitalization increases the reachability of the sensors in the network. Another adaptation to SDN by the optical network represent another clear evidence of the importance of SDN paradigm in [41, 42].

The benefits and enhancement that brought by SDN are not insignificant. It reduces the gap between the adaptation of new technologies and keeping a highly performing system. A SDN platform was proposed in [43] to reduce the complexity of network monitoring in Telecom Italia Lab (TILAB) and handle the associated problems and issues that rose by the high throughput and complexity of the LTE technology adaptation. In such adaptation, the SDN platform provides a logical centralized control layer to monitor the underlining devices dynamically and in a fixable manner in a high bandwidth and complex network without any barriers. Indeed, several systems adapted the SDN platform in their architecture to accelerate the process of responding to different changes.

The abstraction in SDN simplifies the Autonomous System (AS) work. AgNOS [44], an agent-based framework was designed by integrating the features of SDN with AS intelligent agents in a single comprehensive system to permit the network to cover several failures and changes by itself. An additional knowledge layer added over the control layer in SDN to provide autonomous controlled SDN. The authors advocated that the AgNOS is the first autonomous management framework to control SDN environments. The knowledge layer communicates with controller to monitor and take the proper decision without the need to interact directly with the underlying infrastructure or administrator intervention. Such communication provides an abstracted view of the all hosts in the data plane, which facilitate and help the AS agent to monitor and execute several actions in the network. In fact, in large scale environments, it is difficult to deploy an intelligent agent for every host in the data plane. Instead, AgNOS framework provides an easy way to deploy these agents over the SDN controller.

The authors in [45] tried to mimic the idea of AgNOS to improve the inter-domain routing process in the Internet. The Internet consists of a large number of end-hosts and these hosts are organized into groups managed by AS domains. In traditional network architecture, the Border Gateway Protocol (BGP) [46] takes the responsibility of managing the inter-domain communication. The complexity of this architecture caused by the tightly-coupled relation between system components limits the evolution of the network. Moreover, other challenges that are not addressed yet by traditional networking architecture are faced when implementing or deploying any change in the routing rules including: scalability, security, consistencies, convergence latency, error detection response time, etc. In their proposed structure, the authors exploited the SDN features and extended it by adding an inert-SDN domain component, which works as the BGP protocol by holding its routing functionality. Consequently, the new component enabled fast deployment of any new application and accordingly solved the traditional structure issues. The results of the testbed experiments reflected how the new routing structure enhanced the system performance. Furthermore, it showd how the response time increases gradually when the network size grew or when a network path fail and an alternative path was chosen while two domains communication.

Software Defined Access Network (SDAN), which abstracts the benefits of SDN, is considered to be another extended framework to support broadband access. In SDAN several management and control functions are virtualized into control layer to optimize the network. More information about this framework was described by Kerpez, and Ginis in [47] in details.

3. Software Defined Storage (SDStorage)

The main goal for any software defined system is to hide all the complexities of the management and control functionality of the system resources from the end users. These resources may refer to any system component like the components and devices that store and process the data. SDSys provides a new concept

called Software Defined Storage (SDStorage) to facilitate and simplify such complexity, and at the same time, maintain an acceptable level of QoS [48]. SDStorage comes to handle all the challenges facing traditional storage systems. In data storage systems, especially large data storage like data centers, that store a huge amount of data and exploit virtualization to deploy the system. The data forwarding processing and the management processes occur at the same place, infrastructure assets, which increases the burden on the underlying devices and subsequently reduces the system performance. Typically, virtualization is used to enhance the utilization of the system resources; on the other hand, it increases the management overhead across the multiple layers produced when the system is virtualized. This may negatively affect the system's quality of service (QoS).

Many corporations realize the benefits of SDStorage and apply it in their storage centers. Examples include EMC Corporation, which lunched ViPR software as an implementation for SDStorage [49], IBM with Storwize software [50], and many others. All of them define SDStorage based on their own perspectives using different terms, and present their SDStorage solutions designed to work within SDE.

SDStorage is one of the most important subsystems in SDSys. It takes the responsibility of managing a huge data in storage systems by isolating the data control layer from the data storage layer. The control layer refers to the software component that manages and controls the storage resources, whereas the data layer refers to the underlying infrastructure of the storage assets. Such isolation will reduce the management complexity with this new system architecture design. Moreover, it will reduce the cost of the infrastructure by creating a single central control until to manage the different elements in the system regardless of their vendors rather than installing the control software on each element. The controller (software layer) applied various policies on different types of storage elements according to data flows, i.e., requests by different application [48]. Figure 2 illustrates the architecture of the SDStorage system and shows its key components.

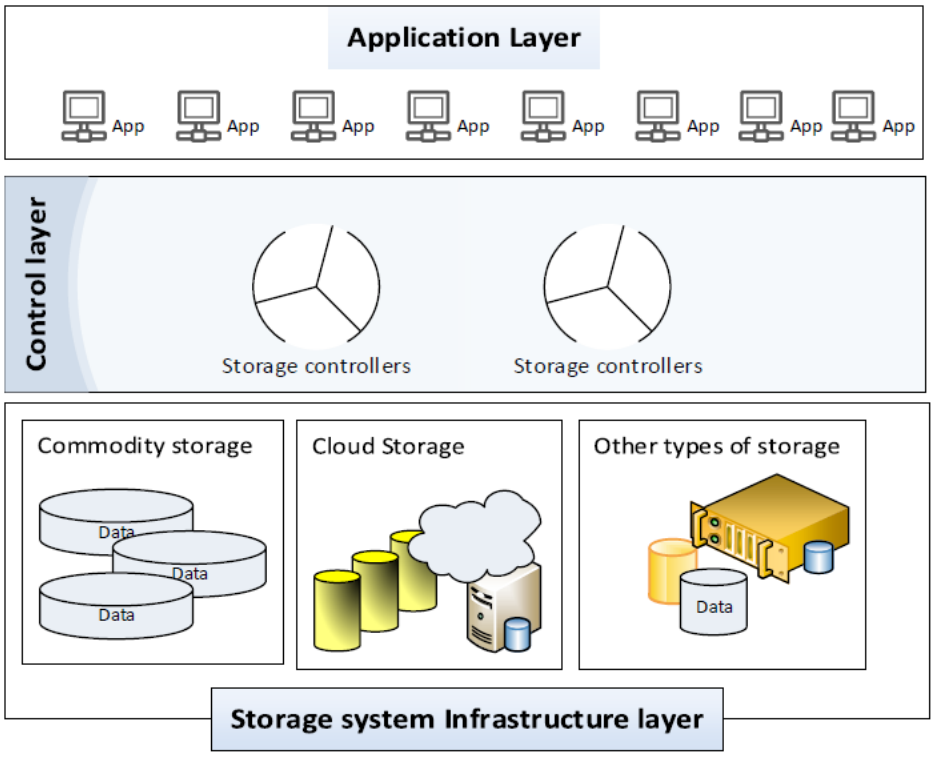


Figure 2: The Software Defined Storage (SDStorage) system architecture

The architecture consists of three layers: infrastructure layer, control layer, and application layer. The Infrastructure layer combines various storage devices storing the raw data. The controller in the control layer is considered the most critical element in SDStorage systems. It is where the storage resources in the infrastructure layer interact with the application layer. The control layer converts different policies to different instructions inside the system. The last layer in this architecture is the application layer which holds different applications and allows the end user to interact with storage devices.

One of the most important issues in designing data storage is to ensure that the applications' QoS requirements are applied successfully. Traditional storage technologies like HDD, PCM, SWD, SSD, etc.

do not support these requirements efficiently. Moreover, the automation and adaptation to changes in the dynamic environment is limited in current technologies. All these problems can be solved by SDStorage [51]. A comparison between the traditional storage model and the SDStorage model is presented in Figure 3.

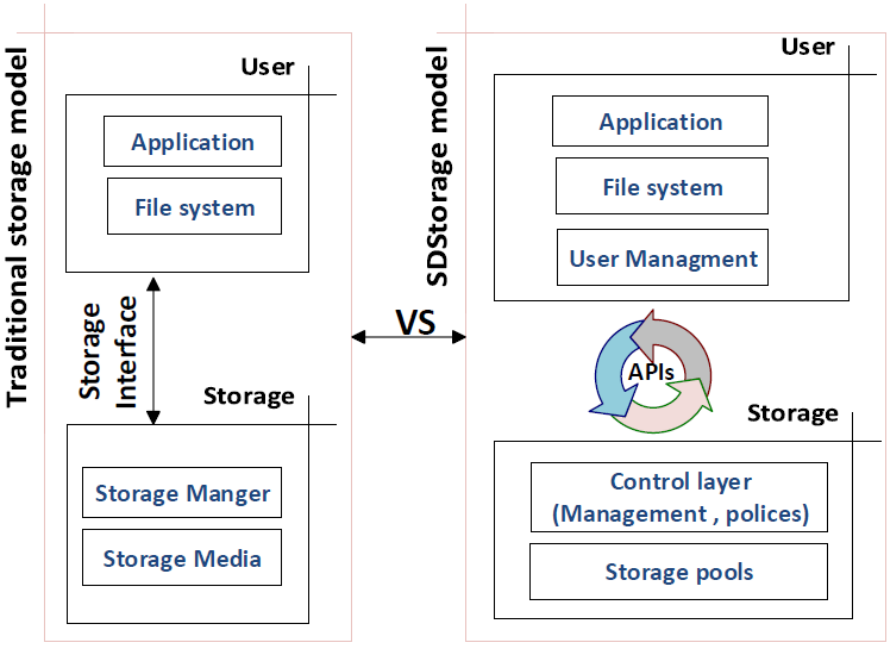


Figure 3: A design comparison between the traditional storage model and the SDStorage model

It is worth noting that in the traditional storage mode, an interface is used to interpret different user requests to storage media, and this interface must be changed when the data type in the storage media is changed (e.g., object, file. etc.). In contract, the SDStorage design has a control layer to define different policies to respond to a variety of requests from the end user through the APIs.

There are some misconceptions linked to the definition of SDStorage [51], which needs to be clarified.

1. Consider the storage software as software defined: The storage software is totally different from defined software, since the former is a tool model, where the later is an architecture which present how the system controlled and managed.
2. Consider any storage virtualization as SDStorage: There is a different between them, since the SDStorage consists of several components; storage virtualization is one of these components, not a complete SDStorage system.
3. Consider the VM environments as a necessary part in SDStorage: While there are some SDStorage system support VM environments, but there are some SDStorage systems do not support or restricted to VM environments.

SDStorage system like other systems has some characteristics that distinguish it from other systems and specifically from the traditional storage systems [51, 52].

1. Scale- out architecture: The resources in SDStorage system can be inserted and removed dynamically to increase the capacity of the SDStorage in a scale up and out fashion.
2. Commodity HW: The system uses commodity, and available resources to build the infrastructure whether it is considered a storage hardware or network communication resources. So in case there is a need to scale the system if any change occurs then the hardware is ready to be added easily without any effects in system performance and work.
3. Resource pooling: As we said the work of the SDStorage is based on virtualization, since all the system resources collected or abstracted on a single logical place and clustered into multiple groups called pools, which controlled by centralized control unit. This reduces the administrators overhead to allocate the resources, since the allocation or de-allocation the resources are done dynamically on-demand.

4. Abstraction: In contrast to traditional storage where there is a decentralized control, the SDStorage gathers all the underlying hardware on a single centralized unit. This unit can capture all the resources in the system, so any event occurs in any part of the system can be recognized and easily handled.
5. Automation: The operations on SDStorage system are done automatically to response to users request in case they need to configure storage space in the system. It also control and monitor the system to check if any reconfiguration is required.
6. Programmability: A lot of APIs are available to provide visible control for the resources, which integrate several system components to enable the system automation. The programmability handles the resources management and change services to meet the applications needs.
7. Policy Driven: One of the most important characteristics in SDStorage is the ability to control and handle all the policies in the system. The control is separated into two control layers, user and storage layers. Where the availability, reliability, latency and other issues specified by user layer. On the other hand, all issues that are required for maintain a high QoS level are handled by storage control layer like recovery from failures, resources migration, and others.

Building a SDStorage solution requires taken in the consideration some points to overcome the challenges that facing classical storage architecture solutions. In [51]some of these challenges illustrated in details, which include:

1. Allocation of the data: There are different data types; Block, Object, File and other.
2. Migration of the data.
3. Reliability of the data.
4. Communication tools(APIs and protocols)
5. Data type support
6. Data compression/ De-duplication
7. Maintain a high level of QoS
8. Handle the Metadata
9. Handle the failure in storage.
10. Communication with others Clients SDStorage
11. Monitor the system

Torsten Volk, Vice President of Product Management at ASG Systems Group, discussed in his blogs [4]the importance of SDStorage for the customers and explained the key and extra capabilities for any SDStorage solution to be taken into account by SDStorage vendors when developing or creating a SDStorage solution. The key capabilities means that there are a mandatory features must be exists in SDStorage solution. I.e: The SDStorage solution must be able to work with different hardware devices, heterogeneity infrastructure, in addition, it should be work with different storage type; NAS, DAS, RAM, Flash, and many other. Further, the solution should provide a centralized and an intelligent solution to manage the storage devices in the data plane. On the other hand, the extra or bonus capabilities means that there are some features may exists or not in SDStorage solution which add an extra value for that solution. I.e: If the solution provide policy driven provisioning, support cloud storage, and support the object storage data type, or it able to translate between different storage data types then these capabilities add an extra features for the SDStorage solution. Moreover he review and list out some SDStorage solutions and their vendors.

The rest of this section reviews and compares the main development for SDStorage solutions. The idea of SDStorage is like any other SDSys, which is mainly based on designing a new system architecture by separating the data plane from the control plane. In SDStorage system, all the data functionality and services are abstracted to the control plane to provide a central control point, where the storage devices hardware remain in the data plane. The differences between these solutions revolve around its performance, capacity, provided services, and other aspects.

IBM lunched its own virtualized novel storage, IBM Storwize, as SDStorage solution to support and complement the SDE [53]. Storwize has many functions to support virtualized environments and help enterprises to manage their huge data growth in an efficient manner. Storwize family provides several storage solutions that can be deployed easily by different size business storage systems. There are several benefits for adapting one solution from Storwize family series. A Graphical user Interface (GUI) to easily manage the storage, smart analysis dashboard to monitor the storage statues, efficient compression, single unified system

[4] http://blogs.enterprisemanagement.com/torstenvolk/2014/02/05/software-defined-storage-customers-care-part-1-2/

to easily manage the file and block types together, and virtualized the data storage, are all features of the Storwize [50]. In addition, Storwize provides a scalable, flexible, virtualized storage management solution for the cloud environments [54]. There are different storage systems lunched by Storwize to serve different purposes; including IBM Storwize V3700, IBM V5000, IBM V7000, and IBM SAN Volume Controller.

EMC introduced another SDStorage solution and provides storage-as-service to reduce the complexity of traditional storage systems. The new solution, which is called the EMC ViPR [49], transforms the underlying storage arrays into virtual resources pools, by installing multiple VMs on top of the physical hardware [55].

ViPR provides the same capabilities of the virtualization by abstracting the storage services from the data path and sit it inside the storage control path. The ViPR SDStorage solution simplifies the storage management and provides an extensible solution to add several storage arrays by creating the adapters to these arrays. The Block, Object, and File storage data types are supported by the ViPR solution, which adds a competitive advantage across other SDStorage solutions. The simplicity of ViPR means that the administrator can manage, add new services, and other operation from a single centralized unit. This centralized unit provides a storage visibility to capture the status of the underlying physical storage, and presents all the storage arrays by different vendors as a single virtualized storage layer.

The ViPR solution is open, in a sense that it supports various storage APIs. The APIs facilitate the services creation process and allow the developers to focus on the services and adapter creation rather than the management and configuration of the underlying storage devices. The extensibility of the ViPR allows enterprises to add different storage arrays by creating adapters and storage applications. There are a number of other implementations of ViPR solution to support other services; ViPR-Object, ViPR-Block, and ViPR-Hadoop Distributed File System for Big Data (HDFS) are examples of these services. NexentaStor [56] is another SDStorage platform created by Nexenta, one of the global leaders in SDStorage. NexentaStor provides a SDStorage solution with a SMARTS [57];

1. Security: by using multiple clones and snapshots to guarantee end-to-end data integrity.
2. Manageability: NexentaStor allows the user to transform their data storage infrastructure, gives them a simple solution to manage it in a flexible and agile manner with lower cost. In addition it provides a GUI for the users to help them in setup and configuration processes.
3. Availability: It runs several services like thin provisioning, periodically asynchronous replication, uses many clones and snapshots, and active-active controller to keep the system available every time.
4. Reliability: NexentaStor uses ZFS, file system to protect the data from corruption, end-to-end checksum and self-healing techniques and other services to maintain a reliable solution for the end users, lower Total cost of Ownership (TCO).
5. Scalability: NexentaStor is able to handle up to Petabytes data. The recent version NexentaStor 4 gives a faster, simpler, and stronger SDStorage solution, since it reduce 50% of the failure time, and integrated with OpenStack, VMware and Windows to keep a stronger, faster and a high performance system. Moreover, this solution offers many ecosystems that integrated to support cloud environments. Block and file storage services also provided by NexentaStor platform.

Beside Storwize, ViPR, and NexentaStor SDStorage solutions, Atlantis USX [58] is another SDStorage solution, which was proposed and implemented to accelerate and enhance the performance and capacity of the system with a minimum cost. Atlantis USX claimed that it can offer the same all-flash array, solid state disk storage system (SSDS), performance only with 50% of the cost of traditional storage arrays like Storage Area Network (SAN) or Network attach storage (NAS). Moreover, Atlantis USX runs real-time services like dataset compression and deduplicating to increase the storage capacity up to 10X, which give the enterprises the opportunity not to think about buying new storage arrays for 5 years.

Atlantis is fully-based software platform integrated with and installed in a virtualized host hypervisor between the physical storage and VMs. It provides an abstract view by pooling the all storage arrays from the infrastructure layer into virtualized storage pools. The platform of the Atlantis USX embedded a HyperDup Content-aware data services, which supports compression and deduplication for data reduction; Caching and Coalescing services to accelerate the IO data processes; Snapshots provisioning for data management; File replication service to support data mobility; Encryption to protect data and maintain a secure data storage, and finally it offers a unified storage service [58].

One of the most important features of any storage system or even cloud environments is the automation process. Atlantis enables automation storage functions by a set of REST APIs, integrated with a set of ecosystems to provide a QoS, resilience and self-provisioning services. In addition, Atlantis USX give enterprises the chance to deploy a virtual desktop Infrastructure (VDI)[58].

Several server based computing (SBC) solutions like Microsoft Remote Desktop Services (RDS), Citrix XenApp, and VMware Horizon can also be deployed by Atlantis to enhance and solve related system's

performance problems. Other features that differentiate Atlantis USX from other SDStorage solutions include its ability to use the RAM of the server beside storage devices to build more powerful systems. In addition, the Atlantis solution uses NAS as capacity backup tier to enhance the system space capacity[58].

It is essential in SDStorage to facilitate the end-to-end (e2e) policy enforcement. In [59]the authors implement a SDStorage architecture and called it IOFlow, where the IO flow between several components in two forms; point-point flows, or multi-point flows. The former implies the enforcement of policy from a single source to another single destination, where the latter implies enforcement of policy form a single or multiple sources to a single or multiple destinations.

In addition the above mentioned main developments for SDStorage, there are numerous other proposals for alternative solutions, but not widely used; to mention but known ones, Maxta [60], HITACHI [61], Datacore [62], CloudBytes [63], IBM SmartCloud [64] are all other attempts. The several SDStorage solutions vary in their capabilities and compete with each other to provide a solution with the highest efficiency at the lowest possible cost and how these solutions share the core principles of SDSys. In [51] the authors talk about some solutions and compared between them across a set of aspects. For instance, not all of the solutions support all the storage data type, block, file, object. The authors also reported that most of the solutions apply a centralized control plane and a few of them follow the distributed control plane. In fact, most of the centralized solutions pretend that this approach enhances the system visibility by making only one single control point for the all underlying system. An alternative approach to this can build a logically centralized control plane but physically distributed; this approach keeps the main SDStorage goal and at the same time increases the system performance and fault tolerance by dividing the workload across several controllers.

Therefore, the motivation for all SDStorage venders is how they can build a comprehensive, scalable, flexible, virtualized, secure, manageable, reliable, available, distributed, simple, cheap, resilience, and portable SDStorage solution.

4. Software Defined Security (SDSec)

Virtualization is increasingly being adopted considering the reduced running costs and increase system flexibility. However, security remains one of the major issues. with the emergence of SDSys, the challenge of security has been highlighted even more [65]. It is not practical to follow traditional security mechanisms in SDN and SDStorage systems. The new architecture in software defined systems requires alternative solutions to be able to deal with such differences. SDN central [66] discussed the main concerns that must be addressed when designing a security solution for SDN; to mention but a couple of these concerns, how the controller must be kept protected and secure to guarantee the availability of the controller anytime, also how to build the trust to protect the communication in the network and make sure that the controller does its assigned tasks. Regardless of the solution, which to be followed, it must be kept simple, cost effective and secure [66].

A new technology has therefore emerged under the software defined system, coined Software Defined Security (SDSec), which is an example of a Network Function Virtualization (NFV) [65]. The new technology works and provides a new way to design, deploy and manage security by separating the forwarding and processing plane from security control plane, similar to the way that SDN abstracts the forwarding plane from control and management plane. Such separation provides a distributed security solution, which scale as VMs by virtualizing the security functions and provides the ability to manage it as a logical, single system [66].

SDSec has been introduced as a solution to support virtualized infrastructure, including virtual network, storage or even virtual servers. In SDSec the functions of network devices like intrusion detection, firewalling and others are extracted from the hardware appliances to a software layer.

The idea behind the SDSec was first introduced by Cloud Security Alliance (CSA) [5]. To transfer their vision into reality CSA launched a Software Defined Perimeter (SDP) project as new security architecture in order to keep secure systems against network attacks [67]. SDP was designed as a complement for SDN to reduce the attacks on the network applications by keeping them unconnected until the users and devices are authenticated.

Traditional security mechanisms are not able to deal with virtualized environments. Indeed, the design of classical security devices is unable to protect the components of virtualized environments, since the traditional security depends on physical network devices and these devices cannot see the significant security

[5] https://cloudsecurityalliance.org/

activities inside virtualized environments. The changes brought by the virtualization including new virtual network topology, threats that come from hypervisors, and the changes of eliminating the roles in virtualized management, demonstrate the need of virtualized security. Such virtualization reduces the complexity and the cost of security operations; it also facilitates the deployment of security policies compared to the classical one by making it more accurate, seamless and context-aware. In addition by virtualizing the security the data center can automate all the security activates like firewalls configuration [65].

Catbird is another full software-based security solution for hybrid and private cloud, introduced to work in VMware environments [68]. Catbird protects the virtual resources automatically, monitor the system security to verify real time security control and enforces the necessary alerts when needed. The the latest release Catbird 6.0, formally called vSecurity, was launched as the first multi-firewalls and hypervisors integration solution. The Catbirds solutions was established to support the hypervisors of Microsoft and VMware as well as the VMware vCloud Networking and security firewall applications and Cisco Virtual Security Gateway (VSG).

vSecurity combined with VMware NSX is argued to simplify private data center deployment and provide accurate, scalable and agile data center. This integration provides vSecurity the visibility of the network VMs and assets. By one-click one can manage, deploy and configure a wide range of technical controls in a data center, saving the customers time and effort for instantiating, validating and monitoring their IT systems. The security in Catbird SDSec approach is based on enforcing policies not related or connected to any particular security device. This way policies can be enforced without the need to care about hardware location [69].

The Catbird was presented to guarantee secure virtual cloud by applying some functionality controls on the system. One of these functions controls events stream and checks if the system does the assigned tasks according to the compliance policies or not. In addition, managing different aspects in the system, like managing the changes and the configuration, triggers and decide a suitable solution in case of any violation occurs.

Catbird consists of two main elements; Catbird control center and a set of virtual machine appliances (VMAs) implemented as VMs. The system is configured like a mesh sensor topology, where the Catbird control center is localized in the center of the network as policy enforcement point to manage and distribute the security controls across the connected VMAs. For every virtual switch there is a Linux-based VMA implemented inside it, executing different security tasks through a hypervisor interface [68].

Catbird architecture is based on the logical trust zoning called Catbird TrustZones, which are logical policy groups. Every new VM has been discovered or created must be assigned to one of these trust zones to make sure that the network is working safely.

The Catbird implements a number of features and attributes that distinguish the SDSec approach from traditional security approaches, and eliminates traditional security bottlenecks that prevent the system from expansion and exploitation of virtual resources. The main ones are listed below:

1. Abstraction: As the SDN abstracts the control and management from hardware appliances, the SDSec does the same idea by abstract the security techniques from hardware layer and sit it in independent software layer. Catbird works in a form of policy envelops, which cover the hardware assets. In such away, the deployment of a new policy control is not affected by the assets or VMs location and it simplify the network work.
2. Automation: the Creation of a new VM or device in the system and allocating it in a specific trust zone is done automatically by the Catbird without the need for manual intervention. Moreover, the process of detecting any violation or vulnerability is also discovered automatically when an event occurs and then a appropriate alert is fired to address the problem. Such a flexibility is not found in traditional security approaches, where the process is mostly manual. The transition from a manual to an automatic process increases the system's speed and enhances its efficiency.
3. Elasticity: Since the Catbird is considered entirely software-based and unrestricted to Hardware, it is easy to scale up the system and adapt to new changes.
4. Concurrency control: In SDSec systems, security techniques and controls like intrusion detection, firewalling, violation monitoring etc. work together as a single cohesive system. Such cohesion improves the system security and accuracy and at the same time reduces the costs. In fact, the abstraction feather of SDSec provides a great approach to apply different aspects of security regardless of the underlining language of the assets' appliances, and this way it provides a higher level of control.
5. Visibility: When Catbird virtualized the security, it increases the ability to discover the problems and abnormal actions and activities.

6. Portability: The independent property of SDSec facilitates the deployment process of Catbird system even if the devices move from one location to another.

The main characteristics of SDSec from the perspective of Mr.Allwyn, the CTO/VP R&D, Networking & Security BU at VMware are not far away from the Catbird features. He tried to combine the SDN and SDSec concepts in one picture as key elements to support virtual data center (VDC). Figure 4 shows these characteristics from his perspective [70].

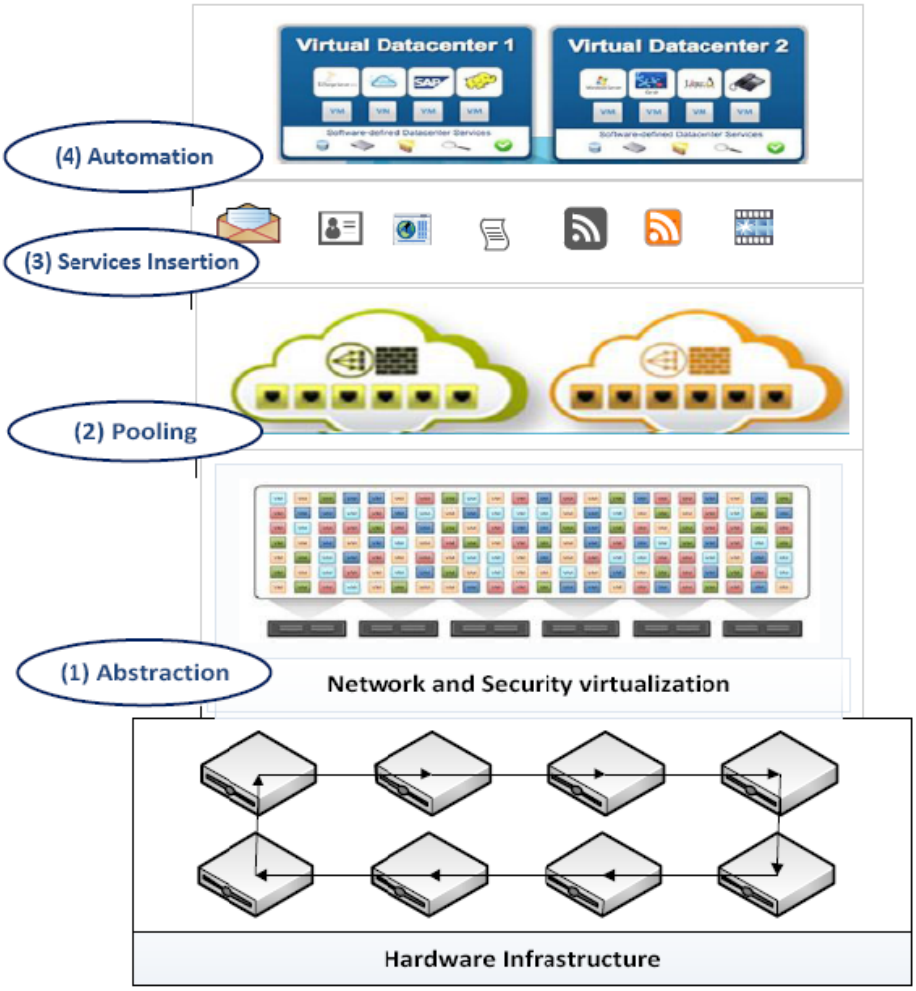


Figure 4: The illusion of SDSec solution characteristics

The features do not differ much from what bought by Catbird system. To build the stack the network and security controls abstracted from underline hardware layer and placed in Network and Security virtualization layers. The pooling feather reflects the idea of policy envelopes in Catbird system, where the security controls grouped into different specialized pools. The service insertion feather means the degree to which the system can scale up and expand by adding extra controls when needed.

Another solution for VMware comes to highlight how organizations can exploit cloud computing resources and at the same time protect the system from network threats. The new suite solution, which is called vShield, vCloud Networking and Security, part of vCloud suite covers the bottlenecks on physical security approach, and provides a single comprehensive virtual security framework [71, 72]. vShield improves security management and reduces the complexity associated with virtualized security solutions, especially for users who need to move to virtualized and cloud environments. It also provides the customer the ability to build various policy-based groups and establish logical boundary between them. Different solution products from vShield suite are integrated to achieve its goal. vShield app to protect the virtual data center applications. vShield can be looked at as an entryway solution to protect the virtual data center boundary and protect the communication between segmentations, as well as for load balancing inside the virtual data center. The vShield manger provides a centralized control point to manage all vShield components [71, 72].

NetCitadel group [73] tried to eliminate the need of manual reconfiguration and respond actions when an event or any change occurs in the network by its OneControl solution [73]. OneControl provides the users with an abstracted user interface, this interface facilitates the network management operations to configure the hardware devices such as network router, switch, or network firewall. A Specific policy language (SPL) is used by the OneControl to describe different policies from different independent vendors [73].

OneControl works like a central point controller [73], which receives any change or event in the network and verifies if this event requires any policy change. Typically, the OneControl uses a specific model, which is called the device configuration translator to generate the required data needed to be pushed to the devices. A Common event format is used to define the events by different sources, the overall system workflow and policy enforcement in OneControl SDSec solution is shown in figure 5.

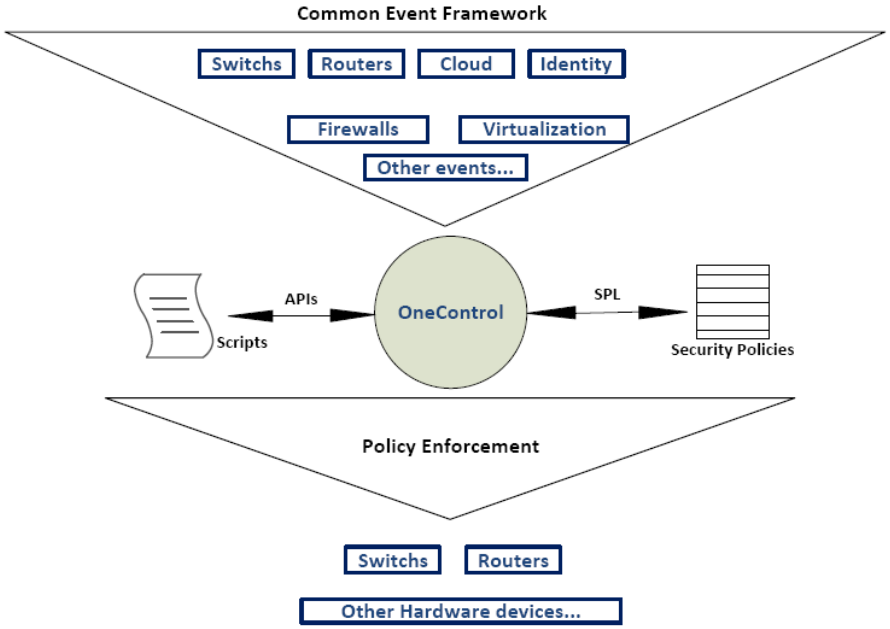


Figure 5: The workflow and policy enforcement in OneControl SDSec solution

As shown in the figure5, the scripts or daemons used by the OneControl to determine if the new event requires a policy change, the new policy rules pushed to specific devices by the controller to take the effects and make response. This allows the system mangers to detect and respond to various events by a set of REST APIs. Different network platforms are supported by OneControl like Cisco IOS, Cisco ASA, Juniper SRX, Juniper SSG, and Linux IPTables [73].

Another security company launched its own SDSec solution for SDN-based and cloud data center systems to fully exploit the benefits of virtualized environments, called vArmour [74]. vArmour offer solutions to scalability, flexibility, and costs bottlenecks that face traditional security techniques in virtualized environments. It provides a dynamic and secure protection for various organizations assets that work cloud computing, mobile applications and virtualization systems. vArmour protects distributed data, which is located across several servers in an efficient and fast manner to allow enterprises to adapt with the new business changes in real-time [74].

vArmour was integarted with Big switch network to provide a scalable dynamic, programmable security solution. The architecture of vArmour is like any software defined system architecture, where the control and forwarding plans are decoupled from each other. A centralized control plane was used by vArmour separated from the security and forwarding planes, and a set of vArmour enforcement point appliances constitute a security (SDSec) service layer to enforce a security rule to a whole data center and process the flow of packets to ensure the safety of VMs. The Big Network and Floodlight controllers were used by vArmour to test the integration with the OpenFlow protocol in the firewall [74].

The vArmour Application Security Gateway (ASG) resides next to OpenFlow-based switch to become a necessary part of OpenFlow fabric to detect if there is an unauthorized message from any compromised server. After that the ASG sends a notification to the big switch controller about the infected server. The controller then changes the traffic direction to another server until the compromised server is cleaned. In

this way the network operation will not be effected and the users do not notice any change in the network performance. In addition it prevents spreading the virus over a network and across several servers [74].

In 2012, HP Company released its own SDSec product, Sentinel [75]. Sentinel works like the previous SDSec products in Virtualized environments. HP launched Sentinel as a SDN security application to work with Open-flow-based controller to detect the threats before they cause damage in the system, and it can be used across every environment needing security to work well [75]. The solution was tested in some customized environments, one of them was a Grammar school. The School's manger decided to integrate with the HP sentinel security solution to protect his school IT infrastructure from malware, viruses, or unauthorized access. This decision was taken after realising the increased security threats due to the fact that students can then access the network by their laptops, phones, and many other internet access tools, and all of these devices can go outside the school environment and bring any form of spyware, malware etc to the school network. After the integration with HP Sentinel SDSec solution, students could access the network in their classrooms by their devices without much concern [76]. Implementing this solution is more suitable for environments, where OpenFlow-based hardware is adopted, but it could also work in other environments.

OrchSec [77] is an Orchestrator-based security solution comes to improve the security of the networks by exploiting the SDN characteristics. Network-visibility is one of the most important features of SDN, when the control plane decoupled from the data plane, which allow the controller to see the underlying infrastructure as a single component. The authors in [77] try to extend this feature and address the drawbacks of the previous network security solutions to provide a fixable, reliable, and high resolution attack detection, security solution. OrchSec not only abstracts the control plane from the data plane, but also abstracts and decouples the monitoring security functionality from the control functionality and set it at an additional architecture component as a Network monitor, to reduce the overhead on the controller for increasing the system performance. The architecture of the OrchSec uses multiple and diverse controllers to guarantee a reliable system. In addition, they focuses to build a loosely-coupled system components, where the security applications in the application layer do not rely on the type of the controller [77].

Figure6, show how the OrchSec architecture is designed. All the security applications are implemented inside the Orchestrator. Elsewhere, the sFlow[78] takes the responsibility to monitor the traffic to see abnormal actions. In such a way, the overhead on the controller reduces. A DNS amplifications security application was developed on the top of the orchestrator to handle the DNS attacks, which is a form of DDoS attacks [79]. An illustrative scenario was shown to explain how the OrchSec handle such type of attacks in an efficient way [79]. Other security applications can be developed on the top of the orchestrator to handle different types of attacks.

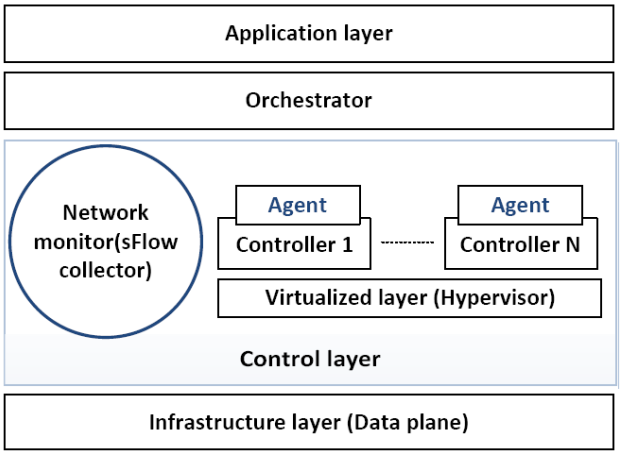


Figure 6: The Architecture of OrchSec design

Despite the fragmented efforts to develop SDSec solutions, such as those mentioned above, there are still some related issues that must be addressed. One of these issues is the centralized vs decentralized security solution. Most of the discussed solutions if not all of them are centralized-based SDSec solutions. This approach increases the probability of single point of failure, and leads to send all the traffic to one point, which causes an overload on that security control point, which in turn reduces the system performance. Moreover, in Centralized-based solutions, SDSec considers a critical component in the system it becomes a central target for attackers. Another challenge with a centralized solution is scalability to handle more

flow-requests in the system within real time, let alone thousands or millions of flow-requests.

Another point or issue that must be taken into consideration is how the hardware is able to realize the new shift with fast, flat, and programmable solution. Otherwise, a new class of hardware devices may be reuired. In addition, the above mentioned SDSec solutions like vArmour and sentinel are more suitable for OpenFlow-based hardware and do not support all network hardware and protocols platforms. These are all concened which will drive for new sophisticated solutions.

5. SDCloud: System Architecture and Design

The design principles of SDSys served as a motivation to propose a comprehensive system architecture framework as an indication of how the system architecture will be designed in the future. This framework in composed of the software defined subsystems previously discussed in this paper: SDStorage, SDN and SDSec. Many ideas are abstracted, integrated and extended inside our model. Such integration guarantees provisioning an entirely software-based architecture for any system. The motivation behind this work is the urgent need to find a scalable, portable, fast, secure, distributed, reliable, available and cloud based architecture solution with a minimum cost.

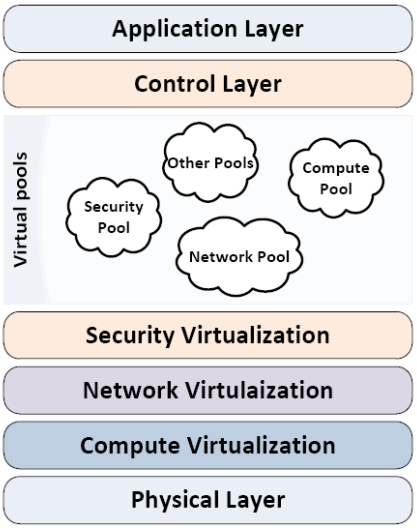


Figure 7: The main layers of SDSys Architecture in general

As shown in Figure 7, the general view of our framework does not differ much from any software defined system in its layered hierarchical design and the abstraction of the data plane from the control plane. However, we distinguish our framework through the internal organization of these layers and the extended functionality that have been added. Figure 8 presents the proposed framework with a detailed depiction of the key elements of these layers. The framework is divided into three general layers: the physical (infrastructure) layer, the control (middleware) layer and the applications (infrastructure as a service (IaaS)) layer. The details of each layer are as follows.

The Physical Layer. It is the first layer and a cornerstone in the framework since the other layers build over it and the different physical resources (storage, security, network, compute, etc.) are combined inside it. The role of these devices is restricted to performing the tasks according to the formulated rules generated in the control layer such as forwarding messages, storing data, packaging security appliances, etc. There is no control, management, or any decision making process inside these devices. Network assets and security assets are just two samples for the underlying hardware devices.

A virtual machine manager (VMM) is installed at the top of these devices to achieve high levels of CPU and memory utilization. Several VMs are instantiated over the VMM to accommodate as many requests as possible, in addition to providing load balancing. These VMs can be organized into different pools as shown in Figure 7 where each one is dedicated for one type of physical devices like network, security, etc. This way, the overhead and time of switching will be reduced where there are different VMs need to communicate and collaborate with each other to accomplish a specific task.

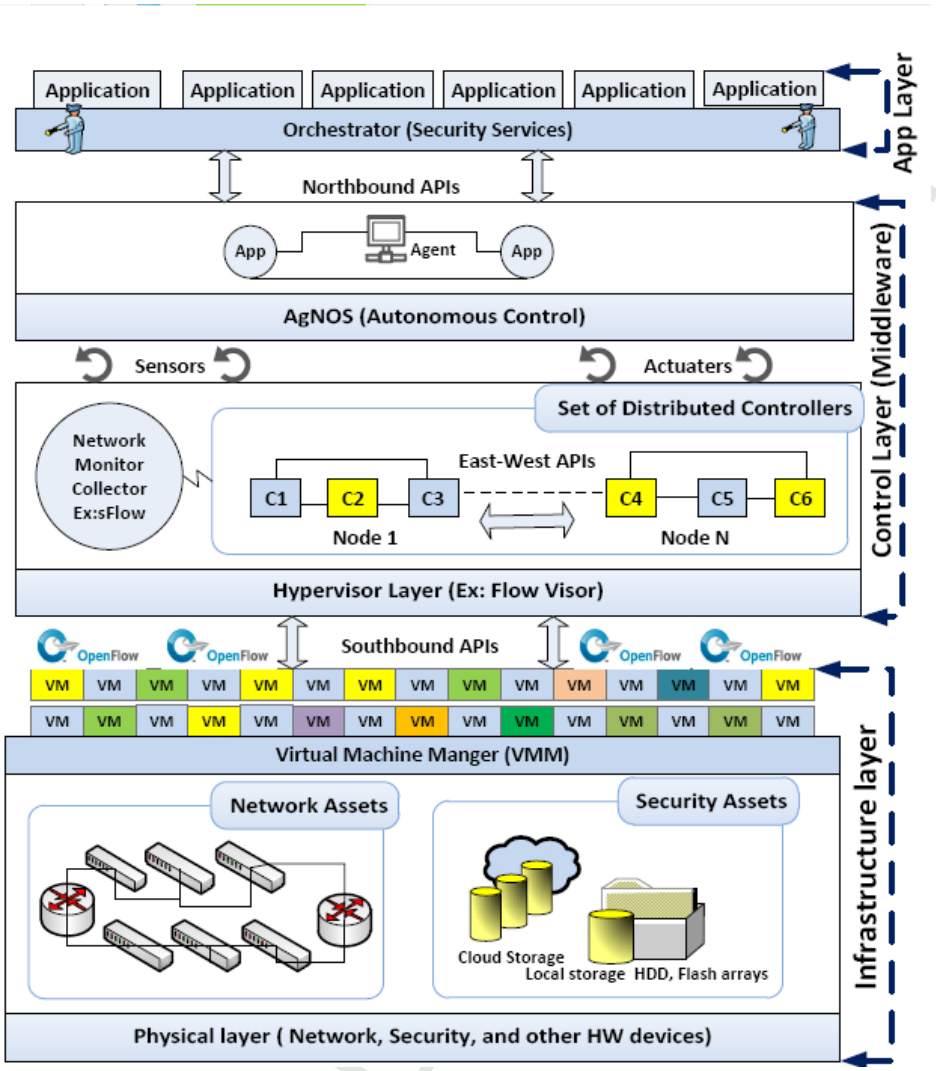


Figure 8: A prototype reflects the future of any system Architecture, SDSys design

The Control Layer. The second layer which is the middleware layer is the mainstay in our framework. It is divided into two sub-layers: the Hypervisor layer and the Agent Network Operating system (AgNOS), Autonomous control, layer.

- The Hypervisor layer: The responsibilities of this layer is to control and manage the underlying devices. All the rules and decisions are defined inside this middleware layer. Since having a centralized control unit with network visibility is one of the main features of any SDSys, this layer can see all the actions happening in the underlying devices.To overcome the problem of single point of failure we re-organize this layer to become logically centralized but physically distributed. In such a way, not only the fault-tolerance will be enhanced, but system performance will be increased too.

  To elaborate more on the last point, the set of controllers are assigned to different distributed nodes, where each node can communicate with the other nodes through the West and East APIs. Different types of controllers can exist inside each node. In other words, types refer to different controller implementation types and functionality types. NOX, Beacon, McNettle and other commercial types can be added. The main reason behind this diversity is to allow the system to handle the heterogeneous devices in the physical layer which differ in their characteristics. The functionality types means different dedicated controllers; SDStorage-C, SDN-C and SDSec-C. All of them should coordinate to optimize the system performance. These different types of controllers communicate and worked with each other as we illustrated in the previous section when we talked about the workflow inside the middleware.

  The idea of abstracting the monitoring functionality from the controller, which we previously discussed, is implemented in our framework. This component reduces the overhead on the controller by taking

some of its functionality. Further, it can communicate with the other controllers through the agents which are implemented at the top of each controller or even at the top of each node.

- The AgNOS layer: This is the second sub-layer inside the control layer. The high cost of management and administration operations compared to other system operations is the key motivation behind proposing this layer. An agent Network OS is implemented at the top of the controllers to provide a self-control, self-management, self-healing, and self-X. The sensors between the two sub-layers detect and transfer all the events occurring in the controllers and transform them to the agent in the AgNOS layer. The agent uses the knowledge base (KB) and runs different applications to decide the best actions to the associated event. It sends the actions to the controllers through the actuators which reside between the two sub-layers also. The control layer can communicate with the physical layer by the Southbound APIs (the most known one is the OpenFlow.

The Applications Layer. It is the last layer in our framework. The lower part of this layer is inspired form the idea of the OrchSec [77] architecture discussed before. In this mode all the security applications are implemented inside the orchestrator to handle the various security attacks. In case any change in the network status or an event occurs, the network monitor collector in the middleware layer detects it and informs the orchestrator about it. Now, instead of asking the SDN-C as what the OrchSec did, our Orchestrator asks the SDSec-C through the agent on the top of the SDSec-C to forward the problematic traffic back to the orchestrator to apply different security mechanisms. As for the upper part of this layer. It is the only visible layer for the end users and consists of many useful applications to serve several users requests and needs. This layer uses the Northbound APIs to communication with the control layer.

The SDCloud system has several advantages over traditional management systems architecture. These advantages include, but not limited to, its low infrastructure cost as it is using commodity hardware. Also, it provides the ability for heterogeneous resource pooling with the ability for scaling up and down these resources. This is accompanied with simple controlling and management for tasks mapping for the available resources and changes deployment. Moreover, its provide one point of view to monitor all the system components. It is worth mentioning that the proposed SDCloud system has some drawbacks that is represented by the incurred overhead on the controller which may negatively impact the performance under certain circumstances such as workload variations. The controller also presents a single point of failure in some cases (e.g. compromising the controller).

6. Experimental Results and Evaluation

In this section, we discuss the implementation details of the proposed framework. We also discuss the experiments conducted on it and the results obtained from these experiments. We start with how the framework is implemented by expanding an existing SDN simulator.

6.1. Simulation environment

There are several existing SDN simulators which give researchers an efficient way to evaluate their SDN frameworks by studying their behaviors and measuring their performances. The Mininet [13] simulator is considered the most commonly used framework by the SDN researchers due to its simplicity and usability, in addition to being an OpenFlow-based SDN simulator. Mininet is an open source simulator written in the python programming language and is built over the Ubuntu Linux distribution.

The elements of Mininet are organized into three main components: the host, which sends and receives the packets, the switch, which stores all the required rules to forward the packets to its destinations, and a central controller, which handles the functionality of control and management operations in the network. Mininet supports different types of virtualized hosts, switches and controllers. Furthermore, it provides two essential tests, ping and iperf, to check the reachability and the network bandwidth, respectively. The minimum topology supported by Mininet consists of two hosts (h1 and h2), one switch (s1) and a central controller (c0). However, the users are given the flexibility to extend this topology and build their own customized topologies to test the performance of their algorithms/solutions. One of the drawbacks of Mininet is its inability to handle large scale networks [80]. MaxiNet is an extension of Mininet to emulate a large scale environment by building a distributed emulation environment [80]. We customize the main components of the Mininet and build our customize topology to provide proof of the concept which support the proposed model.

6.2. Topology's Elements

Starting with the environment and characteristics of Mininet, we build our topology elements: host, switch and controller. The three main elements are illustrated in Figure 9. This figure reflects the proposed SDCloud model in general. As you note the compute devices are connected to aggregated switches and these switches are connected to a core switch. The core switch link directly with a centralized controller. The centralized controller combines the main controller for each software defined discipline; SDN controller (SDN-C), SDStorage controller (SDStorage-C), SDSec controller (SDSec-C) and SDCompute controller (SDCompute-C).

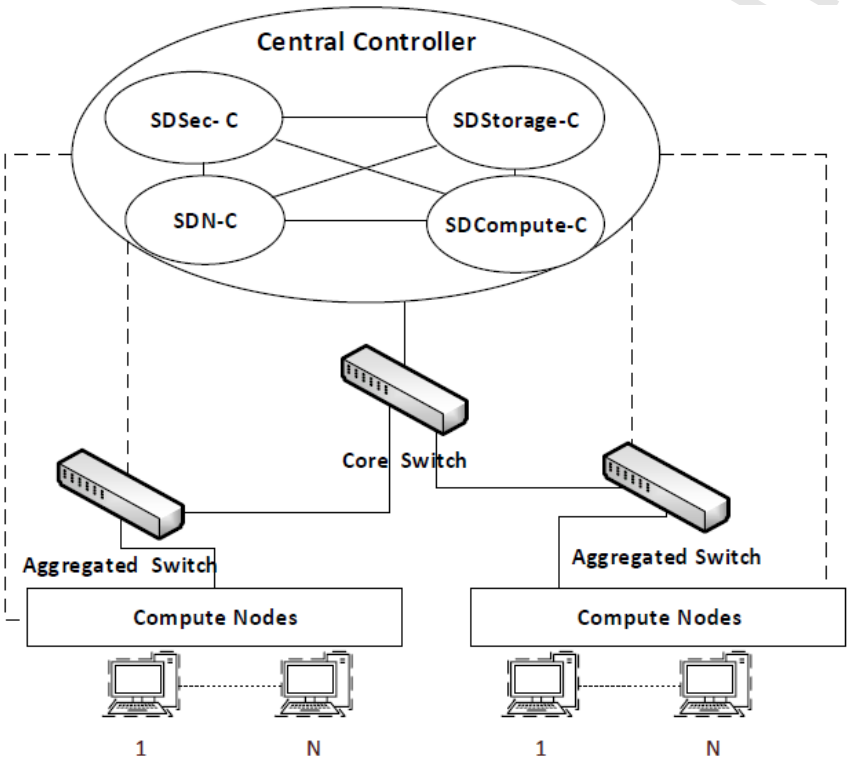


Figure 9: A general overview of the main components for any SDCloud topology

Host. Mininet gives the users the flexibility to customize the host based on their requirements. So, we extend the basic "Host" element in Mininet to meet our functionality. It is meant to be a simple host which may send or receive data traffic through the network. This host is a virtualized host like other Mininet hosts with some extra parameters, which are required for test purposes. We classify the hosts according to their functionality as follows:

- SDStorage_Host [81]: This storage host presents any type of the storage arrays inside which the user can store data. It is a host like any another SDN Host instance with some extra parameters related the storage emulation such as the number of directories, the number of files inside each directory, the size of the file and many other preferred parameters that can be created inside this host to build customized tests.
- SDCompute_Host: This host is working as any SDN and SDStorage Hosts with new features related to the compute part like the number of CPUs, RAMs and so on. For our experiments, we only focus on the RAMs component. We set 32 GB of RAM for each hosts. Users can request GB of RAM and reserve it to do their tasks. Each host can allow for a certain number of GB of RAMs for each request. The range we used is [1,8].

Switch. Mininet supports different types of switches: UserSwitch, KernelSwitch and OVSSwitch. For our test purposes, we choose to customize the first type due to its simplicity. In addition, this switch can serve the functionality of network, storage and compute switches. It is a combination of all of these switches (this switch inherits all of the following switches types).

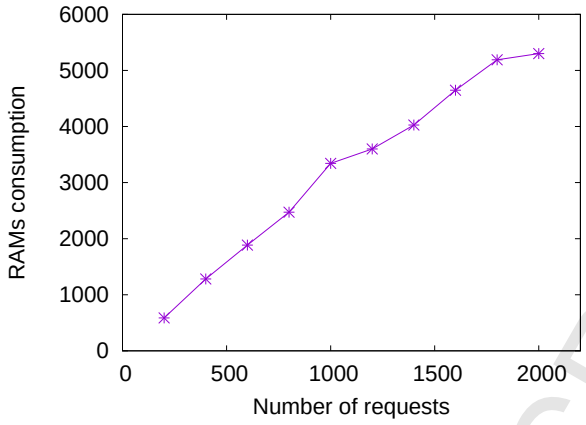


Figure 10: Total RAMs consumption of the compute hosts over several number of requests

- SDStorage_Switch[81]: The SDStorage_Switch inherits all the functions and parameters of the User-Switch and implements extra functionality related to our topology model. Inside this switch, a Function_Table is created to keep up-to-date information about all the _Hosts related to its status and keeps this information available to other hosts in the system.

- SDCompute_Switch: This switch contains two Tables; Compute table and Map table.

  – Compute table: Keeps information about the Compute hosts like; available RAM, Maximum allowed RAM, Power key statues (on/off), and so on.

  – Map table: Inside this table, information about the user requests are kept such as Sender IP, Destination IP, location of the allocated the RAM.

Controller. The controller uses the features of the "Controller" element in the Mininet which is the super class Controller for the all OpenFlow controllers. Mininet provides users the ability to choose the best controller implementation to control their topologies whether it is implemented locally inside the Mininet VM or externally by linking the topology to a remote controller. For our tests, we use the basic controllers for Mininet like the super class Controller itself and POX Controller [82] to implement our techniques by customizing the controllers for our purposes. This controller is entirely software-based and it is considered to be the engine for any SDSys systems, since all the network, storage, security and compute controls can be implemented here. Like the host and switch, this controller does the functionality of SDStorage and SDCompute beside the SDN functionality.

- SDStorage_Controller: This controller is considered the engine for any SDStorage system, since all the control operations are generated inside it. Get_Number_of_StoredFiles, isFull, Used_space, addFile and Available_space are examples of the functions which can be add to the SDStorage_Controller to control several aspects in SDStorage system. This controller monitors the underlying SDStorage_Host and handles different requests by the hosts. If any change occurs, this controller updates the Function_Table and after that it notifies and pushes these changes to the SDStorage_Switch.

- SDCompute_Controller: Inside this controller all the compute controls and functions are generated. This controller handles all the aspects related to the compute resources and then push these information to the compute switches to update the Compute table and Map Table.

6.3. Results and Analysis

For test purposes, we build a customized software defined topology. Inside this topology we use the elements described previously to configure and run it. We start with a simple experiment for the compute and storage resources. Figure 10 shows the effect of increasing the number of requests on the RAM consumption for the 256 hosts that are requested by another 256 hosts. As we notice, the RAM consumption for the hosts will dramatically increase.

As mentioned before, every SDStorage host has a number of directories and storage files, and each one has a capacity equal to 1000. For that, we try to capture the total used spaces of 80 hosts over several

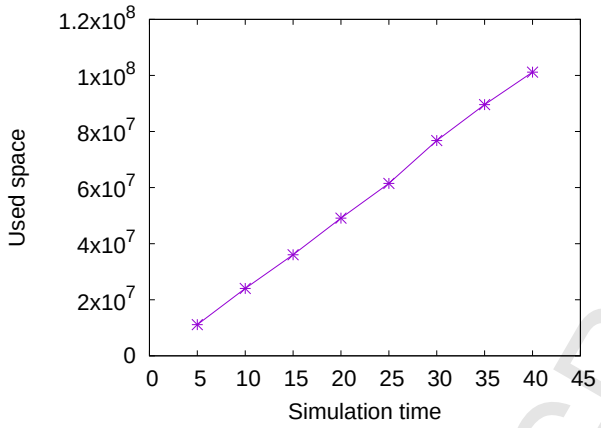
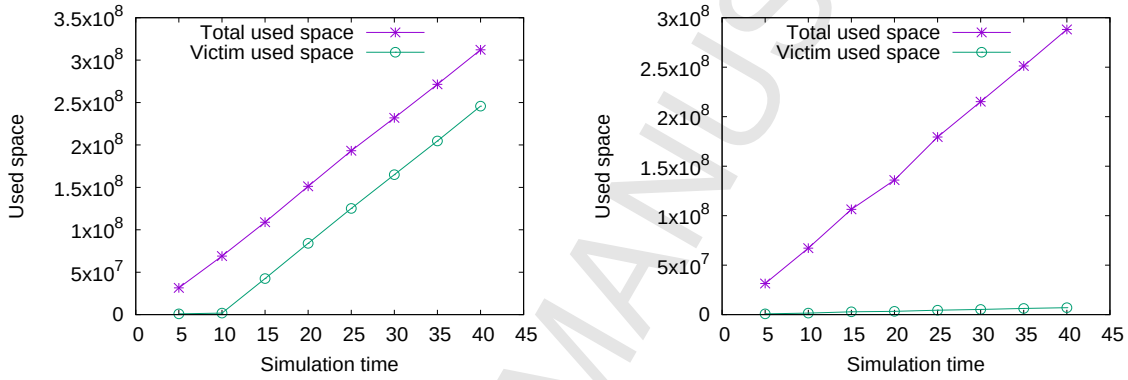


Figure 11: The total used space of receptors storage hosts over several experiment periods



(a) The variation without applying Software Defined solutions

(b) The variation with applying Software Defined solutions

Figure 12: A comparison between the total used space of the storage hosts and the victim host

experiment period times. Figure 11 shows how the used space increased dramatically when we expand the experiment time.

To demonstrate the SDSec aspects of the proposed framework, we build a simple test which generate a DoS attack and show its effect on the storage hosts. After that we present how the software defined solution detects this type of attack and handles it compared with non-software defined solution. We run this test for several experiment periods and we generate a DoS attack around the tenth second. Figure 12 shows the variation between the total used space and the victim host used space across these periods. Figure 12(a) shows this variation without Software Defined (SD) solutions whereas Figure 12(b) shows it with applying the SD solutions. As we notice, when the SD solutions are not applied, the variation is slightly low; which means that most of the requests are sent to the victim host. However, when applying the SD solution, the system works fine and the used space of the victim host is considered reasonable compared with the total used space.

Figure 13 shows the variation between the total number of requests that satisfied the security polices and the number of requests that satisfied the SLA. As shown in the figure, the controller accepts the requests which satisfy the access policies and rejects (drops) the remaining ones. Moreover, as can be seen, even if the request satisfies the polices, it might be ignored due to the SLA conditions.

We try to obtain the percentage of the useful used space in both cases; with and without SD solutions. For that, we fix the simulation time to 40 second and we generate a DoS attack around the tenth second. Figure 14 illustrates these values. For sure, applying the SD solutions increases the percentage of this useful used space.

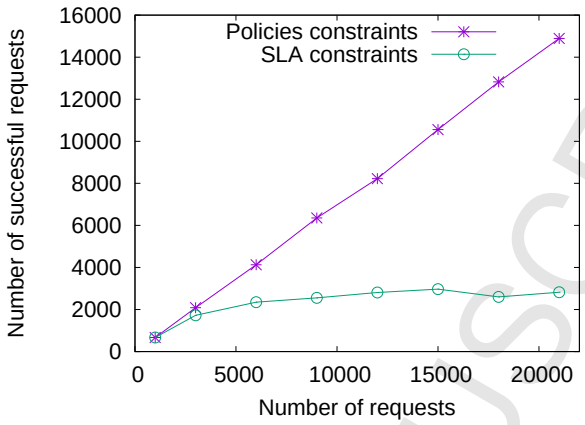


Figure 13: The variation between the total number of requests that satisfies the security polices and the number that satisfies the SLA conditions

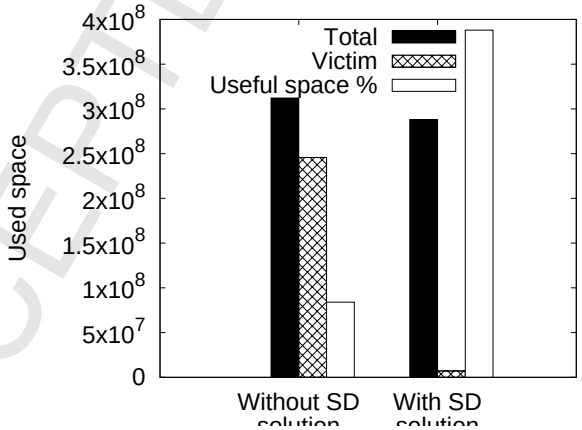


Figure 14: The overall used space consumption for the storage hosts compared with victim host used space

6.4. Further Discussion

The experiments of the previous subsection gives practical examples of some of the benefits of using the concept of software defined control and management for different components of the cloud computing paradigm such as storage and security. However, many other benefits are not shown in these experiments. For example, one of the strongest points used by proponents of software defined systems is their ability to accommodate the potential (and almost inevitable with time) heterogeneity in the hardware infrastructure of large-scale computing systems. For example, the introduction of a new device made by a different vender to a cloud computing system would require the security officer to manually set the policies for the device taking into account all the compatibility issues with existing hardware. This process can be easily automated using the controller of software defined security. Another benefit of using software defined principles which is not evident in the experiments is the ability to adapt to dynamic conditions in a better way. For example, if a new type of attacks is detected and a policy for it is devised, then the security officer would have to manually set the new policies for each type of device in the network. However, having a controller responsible for pushing down policies (new or old) into the switches greatly reduces the effort involved in such scenarios. Finally, having a centralized entity responsible for managing the different hardware resources in all parts of the cloud computing system gives it the ability to detect new threats and attacks even if they are well-orchestrated and involve many agents inside and outside the computing system.

The previous paragraph discussed the merits of using software defined concepts. However, there is a downside to using this concept. The first and most obvious one is the dependence on a single point of failure, which is the controller. This issue cannot be easily circumvented without violating the basic concept of software defined systems. However, there are many techniques proposed in the literature to mitigate its effect. Since the proposed system would be used by providers of large scale computing services, it is not hard for such providers to employ such mitigation techniques. Another issue is performance. Hardcoding rules and policies into the hardware devices responsible for executing them would definitely provide better performance than having to consult another entity (i.e., the controller) before execution. These problems represent the small price paid for the flexibility and adaptability of software defined systems and many people in academia as well as the industry are beginning to realize it.

7. Conclusion

This paper argues that the emergence of Software-Defined Systems was an inevitable result of the paradigm shift from traditional computing models to utility-based Cloud computing. Cloud computing providers typically rely on virtualization (an abstraction of computing resources through software) to effectively and efficiently manage their underlying hardware. Virtualization provides the ability to logically divide physical resources, which allows secure, efficient, multi-tenancy upon single machines. It also enables the ability to aggregate virtualized resources across multiple hosts, provide redundancy through resource migration and elasticity through rapid resource provision and cloning.The ability to abstract a large amount of computing resources, enabled environments that were highly dynamic and could rapidly respond to change. With this came complex and virtual network paths of resources, for individual workloads and therefore also came the need to automate this resource management to enable precise resource provisioning for individual applications.

The management complexities associated with the cloud computing paradigm required out of the box solutions. Integrating different forms of SDSys solutions in one holistic framework that tackle these complexities become an inevitable approach. This paper presents an advanced attempt towards achieving the ultimate aim of achieving a a comprehensive SDCloud. The proposed SDCloud system integrates several aspects of SDSys, including mainly Software Defined Networking (SDN), Software Defined Compute (SDCompute), Software Defined Storage (SDStorage), and software defined security (SDSec), which makes it, the most complext implementation of SDCloud, yet to be reported. The proposed system architecture is evaluated accordingly, which illustrates its feasibility and potential advantages as an integrated solution. The proposed system can be considered as a relatively complex proof of concept, which illustrates a further step towards a wholly automated software-based control framework for cloud computing (SDCloud)

ACKNOWLEDGMENT

The authors would like to thank the Deanship of Research at the Jordan University of Science and Technology for funding this work, grant number 20150050. Also, they would like to thank IBM and IBM Cloud Academy for their support.

References

[1] S. Hariri, B. Khargharia, H. Chen, J. Yang, Y. Zhang, M. Parashar, H. Liu, The autonomic computing paradigm, Cluster Computing 9 (1) (2006) 5–17. doi:10.1007/s10586-006-4893-0.
URL http://dx.doi.org/10.1007/s10586-006-4893-0

[2] R. Buyya, R. N. Calheiros, J. Son, A. V. Dastjerdi, Y. Yoon, Software-defined cloud computing: Architectural elements and open challenges, in: 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014, Delhi, India, September 24-27, 2014, 2014, pp. 1–12. doi:10.1109/ICACCI.2014.6968661.
URL http://dx.doi.org/10.1109/ICACCI.2014.6968661

[3] R. Grandl, Y. Chen, J. Khalid, S. Yang, A. Anand, T. Benson, A. Akella, Harmony: Coordinating network, compute, and storage in software-defined clouds, in: Proceedings of the 4th Annual Symposium on Cloud Computing, SOCC '13, ACM, New York, NY, USA, 2013, pp. 53:1–53:2. doi:10.1145/2523616.2525961.
URL http://doi.acm.org/10.1145/2523616.2525961

[4] M. A.-A. AlaDarabseh, Y. Jararweh, E. Benkhelifa, M. Vouk, A. Rindos, Sddc: A software defined datacenter experimental framework, in: FiCloud 2015, 2015.

[5] Y. Jararweh, M. Al-Ayyoub, E. Benkhelifa, M. Vouk, A. Rindos, et al., Sdiot: a software defined based internet of things framework, Journal of Ambient Intelligence and Humanized Computing 6 (4) 453–461.

[6] L. MacVittie, Stop conflating software-defined with software-deployed, http://virtualization.sys-con.com/node/2929360 [Online; accessed Oct-2014] (2014).

[7] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, ACM SIGOPS Operating Systems Review 37 (5) (2003) 164–177.

[8] E. Bugnion, S. Devine, M. Rosenblum, J. Sugerman, E. Y. Wang, Bringing virtualization to the x86 architecture with the original vmware workstation, ACM Transactions on Computer Systems (TOCS) 30 (4) (2012) 12.

[9] G. Kandiraju, H. Franke, M. Williams, M. Steinder, S. Black, Software defined infrastructures, IBM Journal of Research and Development 58 (2/3) (2014) 2–1.

[10] Y. Jararweh, M. Jarrah, Z. Alshara, M. N. Alsaleh, M. Al-Ayyoub, et al., Cloudexp: A comprehensive cloud computing experimental framework, Simulation Modelling Practice and Theory 49 (2014) 180–192.

[11] C. Li, B. Brech, S. Crowder, D. M. Dias, H. Franke, M. Hogstrom, D. Lindquist, G. Pacifici, S. Pappe, B. Rajaraman, et al., Software defined environments: An introduction, IBM Journal of Research and Development 58 (2/3) (2014) 1–1.

[12] M. Al-Ayyoub, Y. Jararweh, M. Daraghmeh, Q. Althebyan, Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure, Cluster Computing 18 (2) (2015) 919–932.

[13] R. de Oliveira, A. Shinoda, C. Schweitzer, L. Rodrigues Prete, Using mininet for emulation and prototyping software-defined networks, in: Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on, 2014, pp. 1–6. doi:10.1109/ColComCon.2014.6860404.

[14] R. Jain, S. Paul, Network virtualization and software defined networking for cloud computing: a survey, Communications Magazine, IEEE 51 (11) (2013) 24–31. doi:10.1109/MCOM.2013.6658648.

[15] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: Past, present, and future of programmable networks, Communications Surveys Tutorials, IEEE 16 (3) (2014) 1617–1634. doi:10.1109/SURV.2014.012214.00180.

[16] F. Hu, Q. Hao, K. Bao, A survey on software defined networking (sdn) and openflow: From concept to implementation, Communications Surveys Tutorials, IEEE PP (99) (2014) 1–1. doi:10.1109/COMST.2014.2326417.

[17] A. Tootoonchian, Y. Ganjali, Hyperflow: A distributed control plane for openflow, in: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, INM/WREN'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 3–3.
URL http://dl.acm.org/citation.cfm?id=1863133.1863136

[18] K. Phemius, M. Bouet, J. Leguay, Disco: Distributed sdn controllers in a multi-domain environment, in: Network Operations and Management Symposium (NOMS), 2014 IEEE, 2014, pp. 1–2. doi:10.1109/NOMS.2014.6838273.

[19] D. A. Drutskoy, Software-defined network virtualization with flown, Masters thesiss, Princeton University (2012).

[20] B. Sonkoly, A. Gulyas, F. Nemeth, J. Czentye, K. Kurucz, B. Novak, G. Vaszkun, Openflow virtualization framework with advanced capabilities, in: Software Defined Networking (EWSDN), 2012 European Workshop on, 2012, pp. 18–23. doi:10.1109/EWSDN.2012.25.

[21] J. Matias, E. Jacob, D. Sanchez, Y. Demchenko, An openflow based network virtualization framework for the cloud, in: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, 2011, pp. 672–678. doi:10.1109/CloudCom.2011.104.

[22] J. Matias, B. Tornero, A. Mendiola, E. Jacob, N. Toledo, Implementing layer 2 network virtualization using openflow: Challenges and solutions, in: Software Defined Networking (EWSDN), 2012 European Workshop on, 2012, pp. 30–35. doi:10.1109/EWSDN.2012.18.

[23] M. El-azzab, I. Bedhiaf, Y. Lemieux, O. Cherkaoui, Slices isolator for a virtualized openflow node, in: Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on, 2011, pp. 121–126. doi:10.1109/NCCA.2011.26.

[24] D. Turull, M. Hidell, P. Sjodin, libnetvirt: The network virtualization library, in: Communications (ICC), 2012 IEEE International Conference on, 2012, pp. 5543–5547. doi:10.1109/ICC.2012.6364673.

[25] C. Dixon, D. Olshefski, V. Jain, C. DeCusatis, W. Felter, J. Carter, M. Banikazemi, V. Mann, J. Tracey, R. Recio, Software defined networking to support the software defined environment, IBM Journal of Research and Development 58 (2/3) (2014) 3:1–3:14. doi:10.1147/JRD.2014.2300365.

[26] Openstack, http://www.openstack.org/ [Online; accessed Oct-2014].

[27] Y. Jarraya, T. Madi, M. Debbabi, A survey and a layered taxonomy of software-defined networking, Communications Surveys Tutorials, IEEE PP (99) (2014) 1–1. doi:10.1109/COMST.2014.2320094.

[28] A. Doria, J. H. Salim, R. Haas, H. M. Khosrav, W. Wang, L. Dong, R. Gopal, J. Halpern, Forwarding and control element separation (forces) protocol specification, Internet standards track document, Internet Engineering Task Force (IETF) (2010).

[29] N. Foster, A. Guha, M. Reitblatt, A. Story, M. Freedman, N. Katta, C. Monsanto, J. Reich, J. Rexford, C. Schlesinger,

D. Walker, R. Harrison, Languages for software-defined networks, Communications Magazine, IEEE 51 (2) (2013) 128–134. doi:10.1109/MCOM.2013.6461197.

[30] D. Kotani, K. Suzuki, H. Shimonishi, A design and implementation of openflow controller handling ip multicast with fast tree switching, in: Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on, 2012, pp. 60–67. doi:10.1109/SAINT.2012.17.

[31] Z. Bozakov, P. Papadimitriou, Autoslice: Automated and scalable slicing for software-defined networks, in: Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop, CoNEXT Student '12, ACM, New York, NY, USA, 2012, pp. 3–4. doi:10.1145/2413247.2413251.
URL http://doi.acm.org/10.1145/2413247.2413251

[32] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, R. Sherwood, On controller performance in software-defined networks, in: Proceedings of the 2Nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, Hot-ICE'12, USENIX Association, Berkeley, CA, USA, 2012, pp. 10–10.
URL http://dl.acm.org/citation.cfm?id=2228283.2228297

[33] T. Luo, H.-P. Tan, P. Quan, Y. W. Law, J. Jin, Enhancing responsiveness and scalability for openflow networks via control-message quenching, in: ICT Convergence (ICTC), 2012 International Conference on, 2012, pp. 348–353. doi:10.1109/ICTC.2012.6386857.

[34] Y. Kanizo, D. Hay, I. Keslassy, Palette: Distributing tables in software-defined networks, in: INFOCOM, 2013 Proceedings IEEE, 2013, pp. 545–549. doi:10.1109/INFCOM.2013.6566832.

[35] Y. Hu, W. Wang, X. Gong, X. Que, S. Cheng, Balanceflow: Controller load balancing for openflow networks, in: Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on, Vol. 02, 2012, pp. 780–785. doi:10.1109/CCIS.2012.6664282.

[36] H. Egilmez, B. Gorkemli, A. Tekalp, S. Civanlar, Scalable video streaming over openflow networks: An optimization framework for qos routing, in: Image Processing (ICIP), 2011 18th IEEE International Conference on, 2011, pp. 2241–2244. doi:10.1109/ICIP.2011.6116083.

[37] H. Egilmez, S. Dane, K. Bagci, A. Tekalp, Openqos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks, in: Signal Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific, 2012, pp. 1–8.

[38] B.-Y. Ke, P.-L. Tien, Y.-L. Hsiao, Parallel prioritized flow scheduling for software defined data center network, in: High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on, 2013, pp. 217–218. doi:10.1109/HPSR.2013.6602317.

[39] M. Canini, D. Venzano, P. Perešíni, D. Kostić, J. Rexford, A nice way to test openflow applications, in: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, NSDI'12, USENIX Association, Berkeley, CA, USA, 2012, pp. 10–10.
URL http://dl.acm.org/citation.cfm?id=2228298.2228312

[40] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, M. Tyson, Fresco: Modular composable security services for software-defined networks., in: NDSS, 2013.

[41] L. Liu, D. Zhang, T. Tsuritani, R. Vilalta, R. Casellas, L. Hong, I. Morita, H. Guo, J. Wu, R. Martinez, R. Munoz, Field trial of an openflow-based unified control plane for multilayer multigranularity optical switching networks, Lightwave Technology, Journal of 31 (4) (2013) 506–514. doi:10.1109/JLT.2012.2212179.

[42] S. Azodolmolky, R. Nejabati, E. Escalona, R. Jayakumar, N. Efstathiou, D. Simeonidou, Integrated openflow x2014; gmpls control plane: An overlay model for software defined packet over optical networks, in: Optical Communication (ECOC), 2011 37th European Conference and Exhibition on, 2011, pp. 1–3.

[43] L. Donatini, R. Garroppo, S. Giordano, G. Procissi, S. Roma, G. Foddis, S. Topazzi, Advances in lte network monitoring: A step towards an sdn solution, in: Mediterranean Electrotechnical Conference (MELECON), 2014 17th IEEE, 2014, pp. 339–343. doi:10.1109/MELCON.2014.6820557.

[44] A. Passito, E. Mota, R. Bennesby, P. Fonseca, Agnos: A framework for autonomous control of software-defined networks, in: Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on, 2014, pp. 405–412. doi:10.1109/AINA.2014.114.

[45] R. Bennesby, E. Mota, P. Fonseca, A. Passito, Innovating on interdomain routing with an inter-sdn component, in: Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on, 2014, pp. 131–138. doi:10.1109/AINA.2014.21.

[46] e. a. Rekhter, A border gateway protocol 4 (bgp-4), Technical report, IETF (2006).

[47] K. Kerpez, G. Ginis, Software-defined access network (sdan), in: Information Sciences and Systems (CISS), 2014 48th Annual Conference on, 2014, pp. 1–6. doi:10.1109/CISS.2014.6814134.

[48] FenggangWu, G. Sun, Software-defined storage, Report, University of Minnesota (Dec 2013).

[49] Transform your storage for the software defined data center with emc vipr controller, white paper H11749.4, EMC Corporation (2015).

[50] Choose a storage platform that can handle big data and analytics, Solution Brief TSS03158-USEN-01, IBM Corporation (2014).

[51] K. Palanivel, B. Li, Anatomy of software defined storage challenges and new solutions to handle metadata, Report, University of Minnesota (Aug 2013).

[52] The fundamentals of software-defined storage " simplicity at scale for cloud architectures", Technical report, Coraid Inc (2013).

[53] G. Crump, Storage switzerland: Software defined storage needs a platform, Technical Report TSL03137USEN, IBM, Inc (2013).

[54] Systems, T. Group, Transform your business with cloud-ready storage " realize wide-ranging cloud benefits using ibm storwize family systems", Technical Report-Solution Brief TSS03147-USEN-01, IBM Corporation (2014).

[55] Transform data center with vipr software-defined storage, white paper h11749.4, EMC Corporation (2014).

[56] Nexentastor solutions guide for mirantis openstack, white paper, Nexenta (2014).

[57] Nexentastor datasheet, DataSheet 20141009, Nexenta Systems, Inc (2014).

[58] Atlantis usx, http://www.atlantiscomputing.com/products/atlantis-usx [Online; accessed Oct-2014].

[59] E. Thereska, H. Ballani, G. O'Shea, T. Karagiannis, A. Rowstron, T. Talpey, R. Black, T. Zhu, Ioflow: A software-defined

storage architecture, in: Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, SOSP '13, ACM, New York, NY, USA, 2013, pp. 182–196. doi:10.1145/2517349.2522723.
URL http://doi.acm.org/10.1145/2517349.2522723

[60] Maxta storage platform (enterprise storage redefined, white paper, Maxta Corporation.

[61] Storage virtualization: How to capitalize on its economic benefits, white paper WP-435-E DG, Hitachi Data Systems (2015).

[62] Top 3 challenges impacting your data and how to solve them, white paper, DataCore Software Corporation (2014).

[63] Storage architected for the new-age datacenters, white paper, CloudByte Corporation.

[64] Ibm data and storage management software for midsized environments, Data Sheet TID14103-USEN-00, IBM Corporation (2014).

[65] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, A. Rindos, Sdsecurity: A software defined security experimental framework, in: Third Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA-2015), 2015.

[66] Security challenges in sdn (software-defined networks), https://www.sdxcentral.com/resources/security/security-challenges-sdn-software-defined-networks/ [Online; accessed Oct-2014].

[67] Software defined perimeter, white paper, Cloud Security Alliance (2013).

[68] Private cloud security, a catbird white paper, white paper, Catbird Networks, Inc (2014).

[69] Redefining security for the software-defined data center, Solution brief, CATBIRD AND VMWARE NSX SB (2014).

[70] A. Sequeiral, Unveiling software-defined networking and security at vmworld 2012, http://blogs.vmware.com/cto/unveiling-sdn-and-sdsec-architectures-at-vmworld-2012/ [Online; accessed Oct-2014] (2012).

[71] Vmware vshield virtualization-aware security for the cloud, white paper, VMware, Inc (2010).

[72] Vmware vcloud networking and security overview, white paper, VMware, Inc (2013).

[73] Netcitadels onecontrol platform the key to intelligent, adaptive network security, white paper, NetCitadel, Inc (2012).

[74] Big virtual switch and varmour, white paper, Big Switch Networks, Inc (2012).

[75] Solutions for hp virtual application networks "innovations for advanced software-defined networking leadership", Fact sheet, HP, Inc (2013).

[76] Ballarat grammar secures byod with hp network protector sdn application, Case study, HP, Inc (2014).

[77] A. Zaalouk, R. Khondoker, R. Marx, K. Bayarou, Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring and sdn control functions, in: Network Operations and Management Symposium (NOMS), 2014 IEEE, 2014, pp. 1–9. doi:10.1109/NOMS.2014.6838409.

[78] M. Wang, B. Li, Z. Li, sflow: towards resource-efficient and agile service federation in service overlay networks, in: Distributed Computing Systems, 2004. Proceedings. 24th International Conference on, 2004, pp. 628–635. doi:10.1109/ICDCS.2004.1281630.

[79] S. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks, Communications Surveys Tutorials, IEEE 15 (4) (2013) 2046–2069. doi:10.1109/SURV.2013.031413.00127.

[80] P. Wette, M. Draxler, A. Schwabe, F. Wallaschek, M. Zahraee, H. Karl, Maxinet: Distributed emulation of software-defined networks, in: Networking Conference, 2014 IFIP, 2014, pp. 1–9. doi:10.1109/IFIPNetworking.2014.6857078.

[81] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, A. Rindos, Sdstorage: A software defined storage experimental framework, in: IEEE International Conference on Cloud Engineering (IC2E 2015), 2015.

[82] Pox controller, http://sdnhub.org/tutorials/pox/ [Online; accessed Dec-2014].

Bio:

Yaser Jararweh

Yaser Jararweh received his Ph.D. in Computer Engineering from University of Arizona in 2010. He is currently an assistant professor of computer sciences at Jordan University of Science and Technology, Jordan. He has co-authored about seventy technical papers in established journals and conferences in fields related to cloud computing, HPC, SDN and Big Data. He was one of the TPC Co-Chair, IEEE Globecom 2013 International Workshop on Cloud Computing Systems, and Networks, and Applications (CCSNA). He is a steering committee member for CCSNA 2014 and CCSNA 2015 with ICC. He is the General Co-Chair in IEEE International Workshop on Software Defined Systems SDS -2014 and SDS 2015. He is also chairing many IEEE events such as ICICS, SNAMS, BDSN, IoTSMS and many others. Dr. Jararweh served as a guest editor for many special issues in different established journals. Also, he is the steering committee chair of the IBM Cloud Academy Conference.

.

Ala'a Drabseh

She is a Master students of computer science at Jordan University of Science and Technology (JUST). Her research interest include cloud computing , IoT and SDS.

Mahmoud Al-Ayyoub

Received his Ph.D. in computer science from Stony Brook University in 2010. He is currently an assistance professor of computer science at Jordan University of Science and Technology (JUST). His research interests include as cloud computing, high performance computing, machine learning and AI. He is the co-director of the High Performance and Cloud Computing research lab at JUST.

Mladen Vouk

Received Ph.D. from the King's College , University of London , U.K. He is Department Head and Professor of Computer Science, and Associate Vice Provost for Information Technology at N.C. State University, Raleigh, N.C., U.S.A. Dr. Vouk has extensive experience in both

commercial software production and academic computing. He is the author/co-author of over 300 publications. His research and development interests include software engineering, scientific computing and analytics, information technology (IT) assisted education, and high-performance computing and clouds. Dr. Vouk has extensive professional visibility through organization of professional meetings, membership on professional journal editorial boards, and professional consulting. Dr. Vouk is a member of the IFIP Working Group 2.5 on Numerical Software, and a recipient of the IFIP Silver Core award. He is an IEEE Fellow, and a recipient of the IEEE Distinguished Service and Gold Core Awards. He is a member of several IEEE societies, and of ASEE, ASQ (Senior Member), ACM , and Sigma Xi.

Andy Rindos

Andy Rindos is the head of the IBM Research Triangle Park Center for Advanced Studies (RTP CAS). He supports and coordinates university relations for the large IBM development and services. He coordinates the efforts to launch a campus research and education cloud computing using the open source Apache Virtual Computing Lab (VCL) solution. He was the Program Co-Chairs and Steering Committee member for the 1st and the 2nd International IBM Cloud Academy Conference 2012 and 2014.

***Biographies (Photograph)**

Yaser Jararweh

Mladen Vouk

Andy Rindos