Software Defined Storage for Cooperative Mobile Edge Computing Systems

Jafar Al-Badarneh,* Yaser Jararweh,* Mahmoud Al-Ayyoub,*
Mohammad Al-Smadi* and Ramon Fontes†
* Jordan University of Science and Technology, Irbid, Jordan
jafar92.jaa@gmail.com, {yijararweh, maalshbool, maalsmadi9}@just.edu.jo
† University of Campinas, Campinas, Brazil
ramonrf@dca.fee.unicamp.br

***Abstract*—In this paper, we explore the use of both Software Defined Systems (SDS) and Network Function Virtualization (NFV) to deliver a wireless-based Software Defined Mobile Edge Computing (SDMEC) support for storage applications. The proposed approach aims to provide a MEC level service controlled by the software defined paradigm to enhance the provisioning and management of storage services over wireless connected spectrum. The proposed SDMEC has been implemented as an extension of the well-known Wireless Software Defined Networking (WSDN) emulator, Mininet-WiFi, to support wireless topologies that mimic real world environment setups. The framework includes a software defined cooperative MEC strategy for managing storage operation. It also provides the support for auto-scaling network storage resource based on the network demand. The findings of the experiments show the potential of our proposed approach and provide a great starting point for researchers to start considering such approach.**

I. Introduction

Network Function Virtualization (NFV) is under the spotlight as it delivers very promising outcome for telecommunication carriers [1]. NFV basically enables the decoupling of the hardware from the running software modules which provides support for running software modules on commodity hardware. This saves a lot of money, in addition to the advancements in the utilizing orphaned legacy hardware. NFV also reduces the risk in services deployment, as there is no longer a need to have dozens of configurations that should be made on site by the service vendor. With NFV, the process of deploying a service takes no longer that a couple of minutes for the service to be configured and be ready to operate properly.

NFV can be utilized to provide support for Mobile Edge Computing (MEC). One example is the application proposed by Nguyen et al. [2]. The authors proposed to utilize NFV in order to provide MEC level services, where MEC nodes are virtualized to deliver more services than those assigned to them. With this, MEC nodes are capable of serving DHCP services for the closest consumers as a way to maintain the best quality of Service (QoS). Of course, there exist some trade-offs as the service needs a fine-grained control as proposed by the authors.

Along with what NFV has enabled for networking advancements, Software Defined Networking (SDN) comes as another revolution in today's networking paradigms. SDN provides network architects with the simplest and most efficient way to manage and provision network resources. As stated by Costanzo et al. [3], SDN introduced nothing but networking simplification and evolvability. SDN basically delivers these properties by separating the data plane from the control plane, where the controller tends to propagate forwarding tables from the control plane to the data plane devices (routers/switches) [4]. Usually, network traffic is generated by the forwarding devices in the data plane. Network centralized/decentralized management represented by the controller(s) is/are responsible for finding the best routes to deliver network traffic for their intended destinations, in addition to maintaining a global information sightedness of the whole network resources as they're going to be provisioned by the controller (cf. SDN architecture and models [5]).

Recently, the Internet has been moving toward cloud-based services, as hosting and managing your own servers require a lot of time to be constructed in addition to the overhead being imposed by the administration and management of such systems. Cloud computing solves such issues by eliminating the aforementioned complexities and focusing more on the flexibility of delivering the service. The concept of Software Defined Systems (SDS) has emerged to deliver such management ease. SDS inherits the properties of network softwarization (SDN) and generalizes them to other systems. In the past few years, researchers have been experimenting with software defined systems such as Software Defined Storage [6] and Software Defined Security [7]. With its promised capabilities, SDS can be viewed as the future of Cloud computing [8] and Internet of Things [9].

Mobile Edge Computing (MEC) has evolved to deliver a major shift in Mobile Cloud Computing (MCC) [10]. The idea of MEC lies in mitigating network's end-to-end latency by delivering network services - Storage Service for instance- via Edge computing servers, thus, enhancing the quality of the user experience.

In this paper, we introduce wireless-based software defined storage simulation framework for MEC, where storage services are served collaboratively and provided by the closest MEC node to the end users. The framework is developed and validated as an extension to the Mininet-WiFi [11] software defined wireless networking emulator.

The rest of the paper is organized as follows. In Section II,

978-1-5386-2855-3/17/$31.00 ©2017 IEEE

we explore the current software defined storage solutions. In Section III, the technical details about the presented framework are presented. Section IV shows the experiments we conduct to evaluate our framework and their results. Finally, in Section V, we conclude our work and discuss the future improvements and prospects for the SDMEC Storage framework.

II. Software Defined Storage for Mobile Cloud Computing

According to IBM,[1] 2.5 quintillion bytes of data are being generated every day by users (social media posts, images, videos, etc.) as well as sensors. To keep pace with the exponential growth of data generated by our smart devices and IoT sensors, a new storage architecture has been introduced delivering storage services with lower management and maintenance costs [12].

The world has realized that traditional legacy storage services and architectures are no longer fit with the current demand. A long list of challenges faced by current traditional storage systems have been addressed in [13]. The need for a new storage architecture has been emerged as today's current legacy storage systems have failed to run with the current demand (such as cloud systems and VM-centric storage services). Another interesting challenge that contributed to such transition is that corporates are now demanding a scale-out storage support. Although there exist legacy storage architectures that were build to support scaling up the hardware for increasing storage capacities, they are hard to be provisioned and maintained, as the demand for extra storage requires new storage mediums - racks or storage shelves for instance - to be added to the customer's data centers, which is inefficient [14]. Due to these challenges and the current industry needs, the world is now moving towards more flexible storage solutions, with the ability to consider commodity hardware support, scale-out architectures and self-provisioning systems. This is where software defined storage (SDStorage) emerges as one of the appealing solutions.

SDStorage inherits its core concept from SDN with the separation between the data plane (data storage) and the storage control plane delivering more sophisticated ways for storage provisioning and management, and most importantly, coping up with today's storage demands. Moreover, it also supports: Abstraction, Resource pooling and Automation.

Currently, SDStorage is being tapped more by people from the industry rather than the academic world. Big corporates/vendors, like IBM, DELL and many others, have realized the importance of having SDStorage systems deployed by their data centers, in order to sustain in the market and, at the same time, enhance their QoS. According to a market survey published in 2015 by DataCore [15], 53% of the business who are willing to implement SDStorage in the industry were driven by the need for extending the life of their existing storage assets, thus, leading to a reduced Capital expenditures (CAPEX). The provisioning and management ease provided by SDStorage systems makes it also appealing to the corporates in the industry as it consequently increases the savings in the operating expenses (OPEX).

One example of SDStorage solutions from the industry is EMC ViPR provided by Dell EMC [16], which delivers SDStorage support with the ability to merge the controls of multi-vendor storage platforms into one platform, serving automatic provisioning, resource pooling and policy-driven storage services. Another example is Hwelett Packet (HP), which provides SDStorage solution as an integral part of their software defined data center (SDDC) solution [17]. It uses a Virtual Storage Alliance (VSA) for storage services and data deduplication. VSA basically provides the ability to exploit any unused storage capacity and turns it into a Storage Area Network (SAN). Similar systems has been introduced by other vendors such as IBM Spectrum storage [18], VMware software defined Storage [19], and DataCore [20].

As of academia, several researchers have investigated SDStorage. Huang et al. [21] proposed a SDStorage architecture based on the following properties:

1) It should facilitate the decoupling of the data plane (where the data is generated) from the storage control plane.
2) It should facilitate auto-scaling and self-provisioning via Adaptive Quick-Response (AQR) storage controller.
3) It has to be equipped with RESTful APIs as integration interfaces for the application layer's extensions.

The goals behind AQR-storage proposed by the authors is to serve a hybrid self-configured storage systems that can adapt and chose the storage mechanism (NAS (Network Attached Storage), DAS (Directed-Attached Storage) or SAN) according to a Service Level Agreement (SLA). The matching is performed utilizing an analysis module that is operating neural network to learn the best storage configuration and act upon that if any performance inconsistencies take place.

Another work is by Yang et al. [22], where the authors proposed an architecture that uses `OpenStack` to deliver a storage system. The proposed architecture supports heterogeneous storage systems such as Hadoop file system HDFS, SWIFT, and CETH.

Finally, due to the infeasibility of experimenting with software defined storage within a real working environment, a recent work done by Darabseh et al. [6] proposed an emulated SDStorage experimental framework that was developed as an extension to the well known SDN emulator Mininet [23]. The emulated architecture consists of mainly the same components agreed upon in the literature, where the storage plane is isolated from the storage control plane enabling a policy-driven storage service.

III. SDMEC Storage: a Software Defined Storage for MEC Systems

MEC forms a paradigm shift in MCC. MEC enables services to be delivered at the edge of the network, mitigating the number of subscribers requests offloaded to cloud servers, hence, reducing the requests' end-to-end latency [24]. The

[1] https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

motives behind incorporating MEC support as an integral part of cloud and telecommunication systems are expressed as follows:

- Reduce resource consumption of smartphones - as they have limited capabilities (storage capacity, battery capacity, and the processing power)- by offloading computations and storage operations to MEC nodes.
- Deliver a better Quality of Experience (QoE) by reducing latency for user-centric and context-aware applications, and enable more instantaneous cloud services.
- Mitigate traffic offload imposed over cloud servers by applications that require network intensive tasks.

As an evolution of MEC and decentralized cloud services, the concept of cloudlets has been introduced. Cloudlets tends to serve cloud services to mobile users within a limited range bounded by the WiFi coverage [25]. Although cloudlets can serve the objective of having cloud services provided at the edge of the network, but with the current demand for high computing applications and more storage capacities in addition to users mobility that cannot be served by the limited coverage of cloudlets, cloudlets become paralyzed.

The process of constructing and provisioning MEC is not an easy task. A lot of complexities and configurations should be made in order to serve the purpose of their existence. Also, one of the motives behind MEC is to deliver services corresponding to user-centric applications. Today's applications are more connected to each. So, in order for those applications to deliver a complete application experience, they have to communicate a lot of entities via RESTfull APIs for examples and such communication may consult other services that are not localized to user's MEC nodes. So, in order to have a cooperative MEC consultation strategy instead of offloading requests -other than those that exist in Local MEC- to cloud server, neighboring MEC nodes should be consulted. Such needs add more complexities to the system. Therefor, it is better to have a decent way of management of such services, which aims to hide all of these complexities in addition to offering a more reliable way of managing and provisioning network resources. This is where the role of software defined Systems comes into play.

A recent work proposed by Jaraweh et al. [26] introduced a software defined ubiquitous MEC platform for Cloud systems. The authors showed how MEC can be incorporated into software defined Cloud to overcome the challenges faced by MCC. The proposed work focuses on incorporating ME servers along side with mobile network base stations. An illustration of the proposed vision is depicted below in Figure 1.

A. Software Defined Storage Support for Mobile Edge Computing

SDMEC is considered an integral component of Software Defined Cloud (SDCloud) as stated in [27]. Our proposed framework represents an attempt to integrate SDStorage into MEC, as a way to deliver storage services for wirelessly connected nodes at the edge of the network. In [26], SDstorage

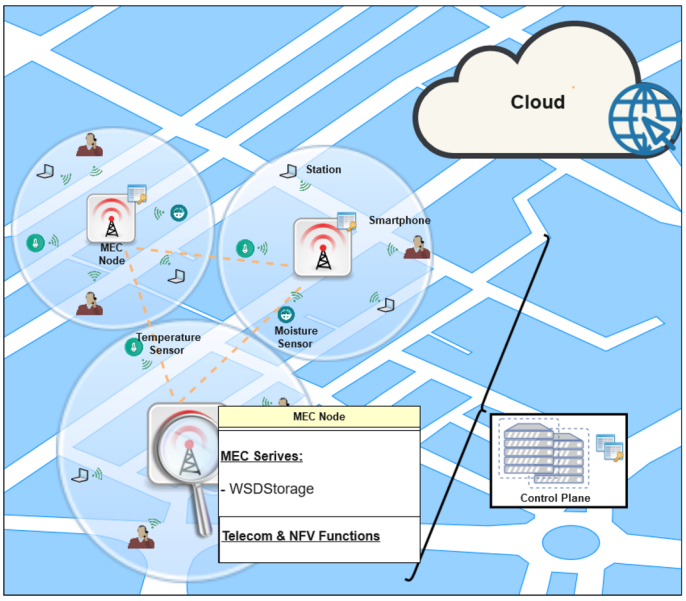

Fig. 1. Software Defined Mobile Edge Computing Storage Architecture

is being served via storage servers that co-exist along with network base stations. What differentiate our approach is the way we hock-up storage service with MEC. From the standpoint of the capabilities of the network base stations in delivering other services than merely managing the traffic flow of the connected nodes, we see that SDMEC storage can be incorporated with network base stations themselves without having any network-connected storage server, which adds some latency (regardless of how negligible it is). The proposed framework utilizes network base stations to deliver storage services. Such incorporation enhances the performance of MEC level applications, such as video streaming, big data analytics, edge health care systems, and sensor network applications.

The proposed SDMEC storage framework focuses on delivering support for context-aware applications -as those mentioned above- that require reliable access to highly-available storage mediums. The framework also aims to serve cooperative MEC data aggregation and big data analytics on the edge of the network. An example of this is analyzing and processing data generated by sensors and street traffic to be either used to take actions or forwarded to the cloud server to perform other centralized decisions.

SDMEC Storage architecture is inherited from Darabseh et al. [6]. It consists of three layers:

1) Data Layer (storage infrastructure): It consists of various storage assets that are managed by a Virtual Machine Manager (VMM).
2) Control Layer: It is responsible for managing and controlling storage resources. This layer contains a hypervisior to be able to manage storage resources in addition to two types of controllers; storage controller to perform storage related tasks), and a network controller to perform SDN related functions regarding MEC

connected nodes, validate storage access policies and manage storage requests by handing them over to the storage controller.

3) The application layer which holds different user-centric applications to interact and use MEC services via northbound APIs.

B. Cooperative MEC SDStorage support

The proposed SDMEC Storage mechanism works according to Algorithm 1. The variable set used in the algorithm is expressed in Table I.

As a use case, assume that a station, that is wirelessly connected to a MEC node, requests to store some data of a specific size. The Storage controller should serve the request. At first, the associated MEC node is set as the Local MEC to the requesting station. The controller checks if the amount of data to be stored is feasible to be stored within the Local MEC. The check process is invoked via a southbound API to the storage data plane. if so, the controller issues a store request with the data size given along with the access policy provided by the station as part of the store request. The store function performs a set of tasks: check the validity of the access policy provided by the station, issue the storage request with the appropriate storage setup, update the storage table for the new data entries, and, finally, disseminate the storage table to the Local MEC.

The previously mentioned flow forms the "sunny day" scenario, whereas the "rainy day" scenario happens when the Local MEC node is incapable of handling the storage request due to a shortage in the size of the available resources. In this case, the data gets chunked and distributed across neighboring MEC nodes. Before engaging other MEC nodes, the Local MEC node is being checked to see how much data it can handle within its available space (if it is not already full of course). If it does, the controller decides how many chunks of the data to be stored according to the amount of available space left, and, then, performs the storage request.

At this stage, the cooperation among MEC nodes takes place. The controller sorts MEC nodes according to their distance from the Local MEC (the closest first). The controller then loops over MEC nodes, to see who is going to handle the request either partially or completely. The remaining chunks are handled that same way they were handled by the Local MEC. If the number of tapped MEC nodes exceeds a predefined threshold, the cooperative search strategy is suspended as it is going to be infeasible in terms of time required to store the data, thus, preserving the QoS.

In this case, the controller instead issues an auto-scale operation in the storage resources among the approached MEC nodes, by activating extra storage resources. This process actually tends to cluster MEC nodes into virtual zones, as each set of MECs in a range within the specified threshold appears to form groups that will be consulted each time a storage request is initiated from a station within their ranges.

Algorithm 1 Software Defined MEC Storage

```
1:  LMec ← station.AssociatedMec()
2:  Size ← station.getDataSize()
3:  Stored = False
4:  if DataSize ≤ LMec.AvSpace() then
5:      StoreData(Data,Size,Pol)
6:      Stored = True
7:  else
8:      Chunks[] = Chunk(station.Data)
9:      F = LMec.FreeChunck()
10:     StoreData(Chunks,F,Pol)
11:     Size = Chunk.size − F.size
12: end if
13: NearbyMecs[] = sorted(MECs)
14: while not Stored do
15:     C = 0
16:     for Each Mec m in NearbyMecs[] do
17:         if C ≤ T and not Stored then
18:             if Size ≤ m.AvSpace() then
19:                 StoreData(Chunks,Size,m,Pol)
20:                 Stored = True
21:             else
22:                 StoreData(Chunks,Size,m,Pol)
23:             end if
24:             C ++
25:         else
26:             ScaleUpStorage(C)
27:         end if
28:     end for
29: end while
```

TABLE I
ALGORITHM VARIABLE SET

Variable	Description
LMec	Station's Local MEC
AvSpace()	a method to get the amount of available space
Pol	an Access Policy that grantees a secured access to storage resources
F	Available free chunks
C	Mec counter
T	Threshold for number of Mec nodes

IV. EXPERIMENTS AND RESULTS

SDMEC storage framework is implemented as an extension to Mininet-WiFi. We start by briefly discussing Mininet-WiFi.

Mininet-WiFi [11] is a tool that allows researchers to experience with a software defined wireless networking emulation environment. Mininet-WiFi is built as an extension to the well-known SDN emulator Mininet. It utilizes the Linux wireless networking driver (mac80211) to provide support for Wireless Stations and Access Points.

For the development of this work we have added the support for more types of nodes in order to run SD Storage functions. In our approach, `SDMEC` represents an extended type of Access

Point that inherits all of its functions and properties, in addition to SD storage related methods and APIs. The same applies for `SDStation` where it extends Station. The framework also supports customized types of `SD Stations`, like `SD Sensor` for example.

The framework requires a customized set of controllers, like `SDNController` and `SDStorage Controller`, each of which performs different tasks, with the ability for them to communicate via East/West APIs. `SDN Controller` performs basic SDN networking function in addition to those related to communicating `SDStorage controller` for storage requests by connected `SD Stations` in addition to handling Cooperative MEC functions, for example, distributing data among multiple MEC nodes, as they have to go through the network medium. The controller also validates storage requests access policies in order to drop any unauthenticated storage request made to the MEC, Whereas `SD Storage Controller` is responsible for handling Storage requests at the MEC as well as serving resource-scaling operations in case there is a shortage in MEC storage. The strategy for handling storage requests was presented earlier in the algorithm pseudo-code 1.

The cooperation behavior for handling storage request can be performed by following either one of two methods. The first method suggests to benefit from the cooperations among MEC nodes by distributing data sent among multiple MEC nodes, as the node's Local MEC cannot handle the sent amount of data. Therefor, the Local MEC will handle as much as it can, and the storage controller then decides to distribute the left amount of data to another neighboring MEC node(s). Such method enables more data distribution among several MEC nodes which might be vital for mitigating Single Point of Failure (SPOF) problems. Other than that, the distribution of data can make the process of recovering missing data easier. This method keeps the controller searching for MEC nodes with available space to handle the storage requests, which add latency to the overall request completion time.

The other method tends to limit the number of hubs a controller can go through in order to consult MEC nodes for serving the storage request, if the Local one fails. This is achieved by thresholding the number of MEC nodes to be tapped, and in case the controller could not find any MEC node with free space within the specified threshold, it triggers an order for the storage controller to scale-up storage resources for the Local MEC and neighboring ones -below the threshold- in order to be able to handle what ever amount of data left from the request. By adopting this method, the frequency of scale-up operations will increase as the controller is limited to the max amount it can get from all accessible MEC nodes around. On the other hand, it keeps the data close to the requested node (user). This will actually might enhance content-delivery services provided by the MEC node, and hence delivering a better QoE. Both methods were experimented with multiple data sizes and different number of MEC nodes.

The first experiment focuses on the latency of storage requests with multiple environment setups. The experiment

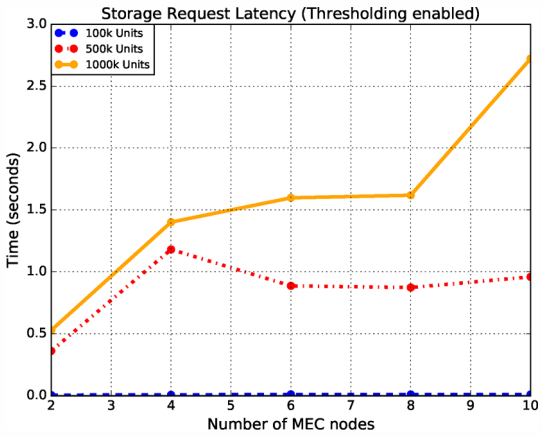

Fig. 2. Storage Latency in seconds with Search Thresholding enabled

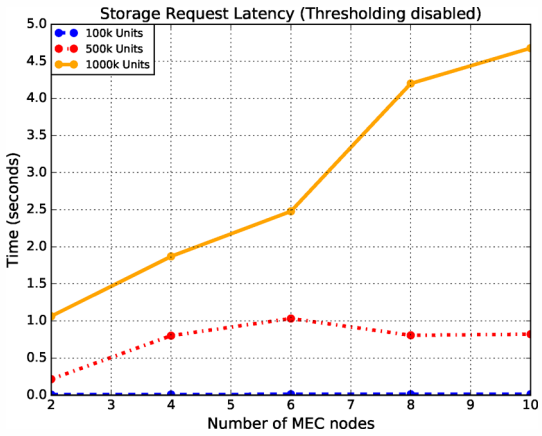

Fig. 3. Storage Latency in seconds with search Thresholding disabled

is repeated for various numbers of MEC nodes (2-10) and the resulting latencies are compared for different data sizes of storage requests (100k, 500k, 1000k) units of storage. The experiment is conducted twice, where the first represents the latency incurred with "Search Thresholding" enabled among MEC nodes, while the other assumes that the controller can consult any MEC node that belongs to the same network asking to store data.

The results of both experiments are depicted in Figure 2 and 3. As can be seen from the figures, the latencies incurred by having Thresholding enabled is relatively lower than those incurred by having the controller freely distribute data among MEC nodes. We can also notice that the big difference in latencies appear to be more clear with high scale data sizes.

The second experiment considers four MEC nodes comparing the latency resulting from different file sizes in the range [100k-1000k] storage units with both Thresholding enabled/disabled. The results are shown in Figure 4. The results show that having Thresholding enabled can be beneficial with large data sizes. So, if the environment under study is very

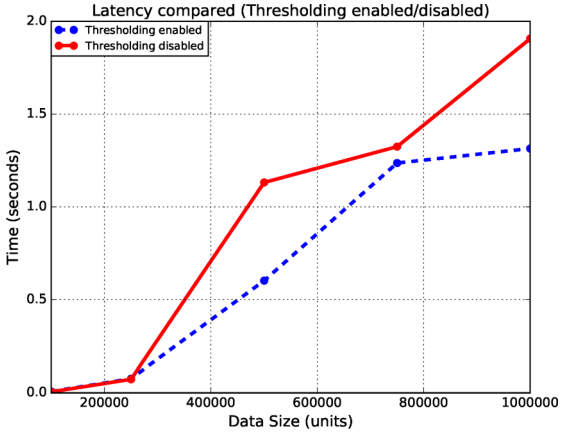

Fig. 4. Storage Latency in seconds compared when Thresholding enabled/disabled

dense, or encounters intense storage requests, thresholding might be the solution for maintaining the service with the least amount of latencies, whereas, other regular environment setups with relatively lower data scales can go with the option of allowing the data to be distributed regardless of how much hub is it away from the Local MEC node.

ACKNOWLEDGMENT

The authors would like to thank the Deanship of Research at the Jordan University of Science and Technology for supporting this work (Grant #20170007).

V. Conclusion

In this paper, we discussed our efforts to build a wireless-based software defined mobile edge computing (SDMEC) framework for storage applications. The aim of the proposed framework is to provide a MEC level service controlled by the software defined paradigm to enhance the provisioning and management of storage services over wireless connected spectrum. We implemented our SDMEC framework as an extension of Mininet-WiFi and evaluated it using a set of illustrative experiments. The findings of the experiments show the potential of our proposed approach and provide a great starting point inducing researchers to start considering such approach, enhance it, and build other systems upon it, such as software defined content-delivery systems.

References

[1] K. Joshi and T. Benson, "Network function virtualization," *IEEE Internet Computing*, vol. 20, no. 6, pp. 7–9, Nov 2016.

[2] K.-K. Nguyen and M. Cheriet, "Virtual edge-based smart community network management," *IEEE IC*, vol. 20, no. 6, pp. 32–41, 2016.

[3] S. Costanzo *et al.*, "Software defined wireless networks: Unbridling sdns," in *EWSDN*. IEEE, 2012, pp. 1–6.

[4] Open Networking Foundation, "Software-defined networking: The new norm for networks," ONF White Paper, Tech. Rep., April 2012, goo.gl/A01Pv8.

[5] W. Xia *et al.*, "A survey on software-defined networking," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[6] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos, "Sdstorage: A software defined storage experimental framework," in *IEEE IC2E*, March 2015, pp. 341–346.

[7] ——, "Sdsecurity: A software defined security experimental framework," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, June 2015, pp. 1871–1876.

[8] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi, and P. Narasimhan, "The case for mobile edge-clouds," in *2013 IEEE UIC/ATC*, Dec 2013, pp. 209–215.

[9] X. Sun and N. Ansari, "Edgeiot: Mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, December 2016.

[10] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, "The future of mobile cloud computing: Integrating cloudlets and mobile edge computing," in *2016 23rd International Conference on Telecommunications (ICT)*, May 2016, pp. 1–5.

[11] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg, "Mininet-wifi: Emulating software-defined wireless networks," in *2015 11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 384–389.

[12] M. Hilbert and P. López, "The world's technological capacity to store, communicate, and compute information," *Science*, vol. 332, no. 6025, pp. 60–65, 2011. [Online]. Available: http://science.sciencemag.org/content/332/6025/60

[13] F. Wu and G. Sun, "Software-defined storage," *Report. University of Minnesota, Minneapolis*, 2013.

[14] I. P. C. Edition, "Software defined storage for dummies," *New Jersey*, pp. 4–5, 2014.

[15] "The state of software-defined storage (sds) ,2015 market survey," 2015. [Online]. Available: https://www.datacore.com/sf-docs/default-source/whitepapers/english/the-state-of-sds-2015-survey.pdf

[16] "Dell emc vipr controller," 2014. [Online]. Available: https://www.emc.com/collateral/data-sheet/h11750-emc-vipr-software-defined-storage-ds.pdf

[17] "Storage for the software-defined data center," 2014. [Online]. Available: http://www8.hp.com/us/en/products/data-storage/data-storage-products.html?compURI=1410844

[18] IBM, "Ibm storage," 2016. [Online]. Available: https://www-03.ibm.com/systems/storage/spectrum/

[19] VMWare, "The vmware perspective on software-defined storage," 2014. [Online]. Available: http://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/solutions/vmware-perspective-on-software-defined-storage-white-paper.pdf

[20] DataCore, "Software-defined storage features and benefits summary," 2016. [Online]. Available: https://www.datacore.com/products/features

[21] M. J. Huang, C. F. Huang, and W. S. E. Chen, "Architecting a software-defined storage platform for cloud storage service," in *2015 IEEE International Conference on Services Computing*, June 2015, pp. 379–386.

[22] C. T. Yang, W. H. Lien, Y. C. Shen, and F. Y. Leu, "Implementation of a software-defined storage service with heterogeneous storage technologies," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, March 2015, pp. 102–107.

[23] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010, pp. 19:1–19:6. [Online]. Available: http://doi.acm.org/10.1145/1868447.1868466

[24] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, Jan 2016, pp. 1–8.

[25] M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: at the leading edge of mobile-cloud convergence," in *6th International Conference on Mobile Computing, Applications and Services*, Nov 2014, pp. 1–9.

[26] Y. Jararweh, M. Alsmirat, M. Al-Ayyoub, E. Benkhelifa, A. Darabseh, B. Gupta, and A. Doulat, "Software-defined system support for enabling ubiquitous mobile edge computing," *The Computer Journal*, pp. 1–15, feb 2017.

[27] Y. Jararweh, A. Doulat, A. Darabseh, M. Alsmirat, M. Al-Ayyoub, and E. Benkhelifa, "Sdmec: Software defined system for mobile edge computing," in *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, April 2016, pp. 88–93.