ISSN:0975-9646

Evaluate Network Security and Measure Performance of Self Healing in 5G

[1]Haripriya N [2]Sangeethalakshmi G

Department of Computer Science, DKM College for Women, Vellore, Tamil Nadu, INDIA

Abstract **– A Network administrator must operate and maintain secure communication. A new technique in networking is SDN, it separates data plane and control plane, making network switches in the data plane simple packet forwarding devices and control plane it able to monitor and control entire network behaviours. . Using Self Healing we focus on SDN network in that technologies we describe enable network operators to implement a wide range of network policies in a high-level policy language and easily determine sources of performance problems. In Existing work, focused on metrics that measure the availability of the services, network functions, and resources (physical and logical) involved in the delivery of those services over 5G SDN-based networks. Our work focusing on transmission losses, in existing all works not focus on file transmission losses or packet losses , so we use divide and conquer method, it's able to reduce the number transmission losses and balance the network traffic load throughout network and protect network against eavesdroppers.**

Index terms: SDN, self healing, network security, Dynamic routing

1.INTRODUCTION

Software Defined Networking (SDN) has emerged as a promising approach for designing future software ecosystems that benefit from and accommodate variability and uncertainty. SDN has, so far, mostly focussed on management and on devising interaction frameworks and APIs to facilitate management of physical components of the network. Less attention has been devoted to meeting high-level application goals such as fault-tolerance and performance guarantees. These however, are becoming increasingly important given growing prevalence of SDN deployments in large-scale and heterogeneous settings involving large numbers of failure-prone components (such as e.g., data centres and clouds). For example, in a fat tree topology employed in Google data centres, switches are organised into redundantly connected hierarchies with low-end commodity switches placed at the leaves, and high-end switches populating the higher levels of the hierarchy. In fact, [1] identified improving robustness of SDNs as the next most important task to be addressed by the SDN research. In this paper, we propose a research program aimed at enabling fully autonomous real-time management of SDN enabled networks in the face of failures and load fluctuations. The main idea underlying our proposal is to leverage self-healing (SH) paradigm as an approach for boosting the SDN robustness and predictability.

The significant compromise to the robustness of any self-healing wireless network versus a wired network, or a centralized wireless network, is the increased latency and loss of throughput to the overhead costs of network maintenance and the inherent costs of store-and-forward messaging. To recover some of that lost network performance, developers need to focus on designing extremely efficient applications--those that take advantage of the processing power available in wirelessly enabled endpoints and that tailor the transport layer of the network to the needs of the particular application. This results in the need to do more careful code building and can mean a steeper learning curve for those wanting to use self-healing technologies. Ultimately, however, the results will be a proliferation of low-power, low-latency, highly scalable applications that will transform the technology landscape yet again.

Network application design--particularly the design of the "simple" applications most likely to run on severely resource-constrained hardware--often exhibits an unfortunate inertia. In the earliest days of embedded digital networks, developers used bus-sharing strategies such as query-response and token-passing to control traffic. Perhaps by habit, some developers try to employ these strategies even when a viable MAC layer is in place. Unfortunately, this redundant traffic control adds overhead and reduces network capacity; when the selection of a self-healing network is already squeezing network capacity, developers designing these networks should question the need to directly control access to it. These kinds of trade-off choices are the hallmark of design for self-healing networks.

One especially useful strategy for avoiding unnecessary overhead is to decentralize tasks within the network. Digital communications assume some degree of processing power at each node, and using that power to handle tasks in a distributed way often requires no hardware changes. Some types of processing are cheaper than sending data in self-healing networks--even the simplest devices can compare data against thresholds, so it is possible to limit messaging to cases where there is something interesting to say. Rather than a periodic temperature report, for example, a temperature-sensing device can be programmed to report "exceptions," conditions under which the temperature falls outside a prescribed range. This kind of exception-based or "push" messaging can greatly reduce traffic in a network, leaving more capacity for communication. Depending on the amount of processing power available at the endpoint and the complexity of the application, a significant portion of the data processing and analysis needed for an application can be done before the data ever leaves the endpoint. Developers need to evaluate their applications carefully to discover how useful this strategy will be for them.

Developers encounter another challenge at the point where data leaves the endpoint. In extremely resource-constrained applications, using extraordinary methods may be necessary to satisfy application specifications. As a concrete example, consider the conscious relaxation of encapsulation in the classical Open System Interconnection (OSI) model in a network using dynamic route selection. Developers need to ask, "Is pure separation of layers appropriate here?"

According to good programming practice and the OSI model, developers should encapsulate routing tasks in the network layer and transmission and reception tasks in the physical layer. Any interaction between these layers should be indirect and mediated by the network layer. Unfortunately, efficient dynamic route selection often depends upon immediate access to physical data such as signal strength or correlate error rate.

In this case, performance can suffer if there are artificial barriers to this interaction. Ultimately, this suggests that the OSI networking model may need to change to suit the characteristics of these new networks, whose importance is growing all the time and should not be underestimated.

2. LITERATURE SURVEY

Existing cellular networks suffer from inflexible and expensive equipment, complex control-plane protocols, and vendor-specific configuration interfaces. In this position paper[30], we argue that software defined networking (SDN) can simplify the design and management of cellular data networks, while enabling new services. However, supporting many subscribers, frequent mobility, fine-grained measurement and control, and real-time adaptation introduces new scalability challenges that future SDN architectures should address [30]. As a first step, we propose extensions to controller platforms, switches, and base stations to enable controller applications to (i) express high-level policies based on subscriber attributes, rather than addresses and locations, (ii) apply real-time, fine-grained control through local agents on the switches, (iii)perform deep packet inspection and header compression on packets, and (iv)remotely manage shares of base-station resources.

Programmable networks brought by Software Defined Networks (SDN)[29] are perceived by operators as cornerstone to reduce the time to deploy new services, to augment the flexibility and to adapt network resources to customer needs at runtime. However, despite the vulnerabilities identified due to the centralization of the intelligence on SDN, its research is more cantered on forwarding traffic and reconfiguration issues, not considering to a great extent the fault management aspects of the control plane.

The aim of this paper is to provide SDN[29] with fault management capabilities by using autonomic principles like self-healing mechanisms. We propose a generic self-healing approach that relies on a Bayesian Networks for the diagnosis block and it is applied to a centralized SDN infrastructure to demonstrate its functioning in the presence of faults.

3. RELATED WORKS

Security & Dependability in SDN

To the best of our knowledge, none of the SDN controllers proposed thus far address security and dependability beyond using simple authenticated communication channels and control data replication among controller instances. For example, no mechanisms are used to assure trusted switch controller association (to avoid malicious devices in the network) or to detect, correct or mask faults of system components. Moreover, no techniques are used to assure data integrity and confidentiality in or between controllers.

In a security and dependability perspective, one of the key ingredients to guarantee a highly robust system is fault and intrusion tolerance. The two main fault models are crash and Byzantine (a.k.a., arbitrary faults). Crash fault tolerant services support only benign failures such as a crashed process, operating system or machine, being a narrow subset of the arbitrary model. Byzantine fault tolerant (BFT) systems are capable of tolerating any abnormal behavior, i.e., intentional or non-intentional faults, while the service keeps its correct operation. Faults (e.g., bugs, mis configurations, attacks) and errors can be masked automatically as they happen, by using state machine replication [17]. Furthermore, in order to ensure the perpetual and unattended operation of the system, errors can be removed with self healing techniques [18], so that there is never an excessive number of a compromised device. Both automatic recovery and perpetual and unattended operation seem to be relevant objectives in the context of SDNs.

The literature on Byzantine fault tolerance is broad, ranging from large-scale systems [19, 20] to resource-efficient solutions [19, 21, 22]. Nevertheless, BFT alone is not enough to guarantee a highly available dependable system, needing self-healing mechanisms as a complement. Techniques such as proactive-reactive recovery [18], for example, can be used to assure the system liveness. These techniques rely on the idea of rejuvenating compromised components (be it by accidental or malicious faults).

Intrusion-tolerant architectures [23] are a step in the direction of this automatic security paradigm. Intrusion-tolerant systems remain working correctly and are capable of assuring properties such as integrity, confidentiality and availability, despite the presence of faulty or compromised components due to successful attacks.

A secure and dependable control plane helps improve the overall network resilience [24], which is our final goal. A resilient system is one that self-adapts to the dynamics of environment conditions, e.g., one that performs self-healing in the presence of persistent threats and where protection parameters, such as number of replicas, length of keys, etc., can automatically increase in case of a severe attack.

Secure and Dependable Control Platform

In this section we present the general design of the secure and dependable SDN control platform we propose. Figure 2 illustrates a simplified view of the architecture. In the remainder of this section we briefly introduce and discuss the several mechanisms which we consider using to address the threat vectors identified in SDNs.

Replication. One of the most important techniques to improve the dependability of the system is replication. As can be seen in figure 2, our controller is replicated, with three instances in the example. Applications should be replicated as well. Besides replicated instances of the controller, in the figure we can observe application B also replicated in all controller instances. This mixed approach ensures tolerance of both hardware and software faults, accidental or malicious.

Replication makes it possible to mask failures and to isolate malicious or faulty applications and/or controllers. Moreover, in case of a network partition, application B, with the proper consistency algorithms, will still be able to program all network switches, contrary to application A.

Diversity.

Another relevant technique to improve the robustness of secure and dependable systems is diversity [25, 26]. Replication with diverse controllers is a good starting case. The basic principle behind this mechanism is to avoid common-mode faults (e.g., software bugs or vulnerabilities). For example, it is known that off-the-shelf operating systems, from different families, have few intersecting vulnerabilities [26], which means that OS diversity constrains the overall effect of attacks on common vulnerabilities. In SDNs the same management application could run on different controllers. This can be simplified by defining a common abstraction for applications (a northbound API).

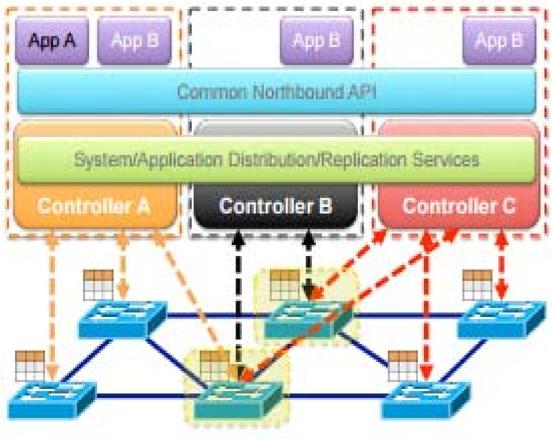


Figure . Secure & Dependable SDN

Self-healing mechanisms.

Under persistent adversary circumstances, proactive and reactive recovery can bring the system back to a healthy state, replacing compromised components, and keep it working virtually forever. When replacing components, it is important that the replacement be done with new and diverse versions of the components, whenever possible. In other words, we should explore diversity in the recovery process, strengthening the defense against attacks targeting specific vulnerabilities in a system.

Dynamic device association.

If a switch is associated with a single controller, its control plane does not tolerate faults. Once the controller fails, the control operation of the switch fails and the switch will need to associate with another controller. For this reason, a switch should be able to dynamically associate with several controllers in a secure way (e.g., by using threshold cryptography to detect malicious controllers and authentication, which would hinder man-in-the-middle attacks, for instance). A switch associated with different controllers would be able to automatically tolerate faults (crash or Byzantine, depending on the configuration). Other advantages include increasing control plane throughput (several controllers could be used for load balancing) and reducing control delay [27] by choosing the quickest-responding controller. Increasing the data plane programmability (near or in the network switches) would be helpful in this respect. Two approaches could be used for this purpose. One option would be to use general purpose CPUs inside the switch to replace some of the traditional functionality of custom ASIC, as in [28]. Another could be to have a proxy element acting on behalf of the switch. This element could be easily deployed in a small black box attached to the switch, with a general purpose micro-computer.

Trust between devices and controllers.

Establishing trust between devices and controllers is an important requirement for overall control plane trustworthiness. Network devices should be allowed to associate with controllers dynamically but without incurring in less reliable relationships. A simple approach would be to have authenticated white lists of known trusted devices, kept at controllers. However, this lacks the flexibility desired in a SDN. Another option is therefore to trust all switches until its trustworthiness is questioned. Malicious or abnormal behaviour could be reported by other switches or controllers, based on anomaly or failure detection algorithms. Once the trustworthiness of a switch or a controller would go below an accepted threshold, the switch would be automatically quarantined by all devices and controllers.

Trust between applications and controllers software.

As software components present changing behaviour due to aging, exhaustion, bugs, or attacks, a dynamic trust model as the one proposed in [12] is required. In this paper the authors propose and demonstrate the feasibility of a model to support autonomic trust management in component-based software systems. They use a holistic notion of trust to allow a trust or to assess the trustworthiness of the trustee by observing its behaviour and measuring it based on quality attributes, such as availability, reliability, integrity, safety, maintainability, and confidentiality. The proposed model can also be applied to define, monitor, and ensure the trustworthiness of relationships among system entities.

3. 1 Existing System

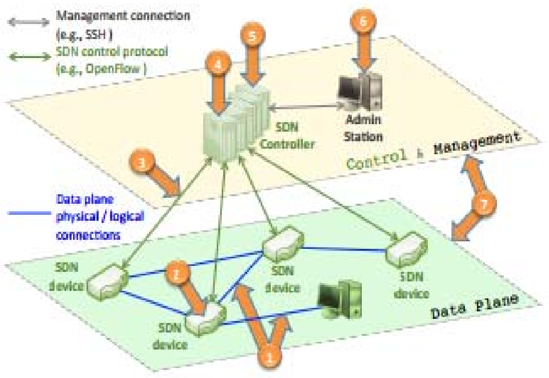
In existing work, they analyzed the vulnerabilities of SDN (Software-Defined Networks) and NFV (Network Function Virtualization) from a fault management perspective, while taking into account the autonomic principles. In particular, we focus on resiliency and we propose a Self-Healing based framework for 5G networks to ensure services and resources availability. In Existing work, focused on metrics that measure the availability of the services, network functions, and resources (physical and logical) involved in the delivery of those services over 5G SDN-based networks.

4. Problem Statement

Now a day's network administrator fails to provide a secure communication, and present more privacy breaches because presence of eavesdroppers and dos attacks, we aim to protect SDN network and also reduce bandwidth usages, transmission failures. Main advantages of SDN able to control entire network behaviors, because it separates to plane one is data plane and another one is control plane.

5. Proposed System

Our work focused transmission losses, in existing work not focused on file transmission losses, we using divide and conquer technique can reduce the number of data transmission and balance the traffic load throughput network and protect network against eavesdroppers. In general, the majority of network communications occur in an unsecured or "cleartext" format, which allows an attacker who has gained access to data paths in your network to "listen in" or interpret (read) the traffic. When an attacker is eavesdropping on your communications, it is referred to as sniffing or snooping. The ability of an eavesdropper to monitor the network is generally the biggest security problem that administrators face in an enterprise. Without strong encryption services that are based on cryptography, your data can be read by others as it traverses the network.


Management connection (e.g., SSH)
SDN control protocol (e.g., OpenFlow)
SDN Controller
Admin Station
Control & Management
Data plane physical / logical connections
SDN device
SDN device
SDN device
SDN device
Data Plane
1
2
3
4
5
6
7

6. Implementation

Module list

- Software defined network
- Network security
- Dynamic routing
- Self-healing performance management in SDN networks

6.1Module description:

6.1.1Software defined network

Software defined network is an approach to computer networking that allows network administrators to manage network services through abstraction of lower-level functionality. This is done by decoupling the system that makes decisions about where traffic is sent (the control plane) from the underlying systems that forward traffic to the selected destination (the data plane). The inventors and vendors of these systems claim that this simplifies networking.

6.1.2 Network security

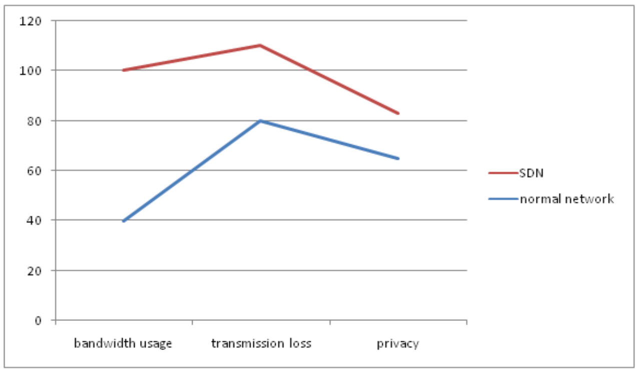
To evaluate the security metrics based on intermediate node performances, we evaluate node performances following factors, time delay, packet dropping or transmission loss this factor all considered during evaluate node performances.

6.1.3 Dynamic routing

Rather than ignoring the broadcast nature of the wireless medium, dynamic routing takes advantage of it. Messages are broadcast to all neighbors and forwarded according to a "cost-to-destination" scheme. Messages act as multiple objects rolling downhill toward their ultimate destination. Although this type of routing takes advantage of multiple redundant routes from originator to destination, it can also generate a lot of traffic on the network. Without modification, it can result in messages traveling in endless loops, jamming up the network.

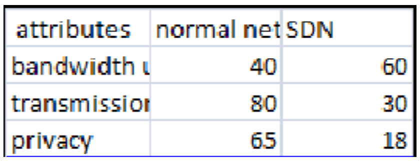
6.1.4 Self-healing performance management in SDN networks

We further propose to extend the basic self-healing paradigm with new techniques capable of reconfiguring routing networks so as to handle component failures as well as congested links and overloaded switches. These techniques will leverage the SDN control plane to gather live load information from network switches and to inject necessary reconfiguration actions.



6.1.4. Above figure is Normal Network vs SDN

Table



attributes	normal net	SDN
bandwidth u	40	60
transmissior	80	30
privacy	65	18

7. Divide-And-Conquer

In computer science, divide and conquer (D&C) is an algorithm design paradigm based on multi-branched recursion. A divide and conquer algorithm works by recursively breaking down a problem into two or more sub-problems of the same (or related) type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

This technique is the basis of efficient algorithms for all kinds of problems, such as sorting (e.g., quicksort, merge sort), multiplying large numbers (e.g. Karatsuba), syntactic analysis (e.g., top-down parsers), and computing the discrete Fourier transform (FFTs).

On the other hand, the ability to understand and design D&C algorithms is a skill that takes time to master. As when proving a theorem by induction, it is often necessary to replace the original problem by a more general or complicated problem in order to initialize the recursion, and there is no systematic method for finding the proper generalization. These D&C complications are seen when optimizing the calculation of a Fibonacci number with efficient double recursion.

The correctness of a divide and conquer algorithm is usually proved by mathematical induction, and its computational cost is often determined by solving recurrence relations.

Definition

Divide & conquer is a general algorithm design strategy with a general plan as follows

1. DIVIDE

A problem's instance is divided into several smaller instances of the same.Problem, ideally of about the same size.

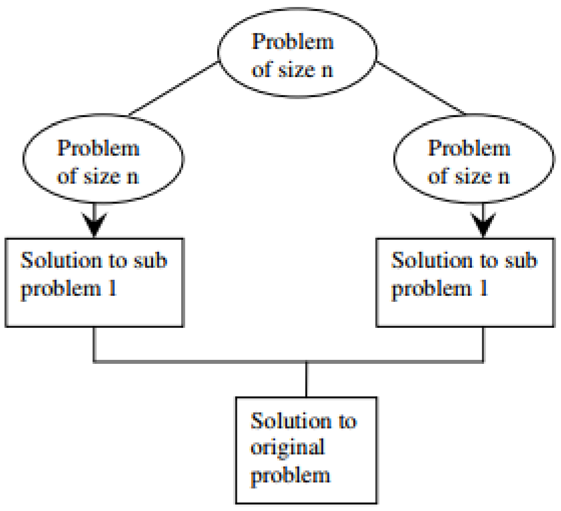
2. RECUR:

Solve the sub-problem recursively.

3. CONQUER:

If necessary, the solutions obtained for the smaller instances are combined to get a Solution to the original instance.

Diagram 1 shows the general divide & conquer plan


Problem of size n
Problem of size n
Problem of size n
Solution to sub problem 1
Solution to sub problem 1
Solution to original problem

Note

The base case for the recursion is sub-problem of constant size.

Advantages of Divide & Conquer technique

- For solving conceptually difficult problems like Tower of Hanoi, divide &conquer is a powerful tool.
- Results in efficient algorithms.
- Divide & Conquer algorithms are adapted foe execution in multi-processor machines.
- Results in algorithms that use memory cache efficiently.

Limitations of divide & conquer technique

- Recursion is slow
- Very simple problem may be more complicated than an iterative approach. Example: adding n numbers etc

8. CONCLUSION

Configuring computer networks is becoming increasingly vexing as network operators must perform increasingly sophisticated network management tasks. Our work focused against eavesdroppers and also transmission loss because now a day's major concern in any network management is to provide a secure communication, so we proposed a model using divide and conquer technique can reduce the number of data transmission and balance the traffic load throughout network and protect network against dos attack.

REFERENCES

[1] T. Koponen et al. "Onix: a distributed control platform for large-scale production networks". In: OSDI. 2010.
[2] N. Gude et al. "NOX: towards an operating system for networks". In: Comp. Comm. Rev. (2008).
[3] M. Caesar et al. "Design and implementation of a routing control platform". In: NSDI. 2005.
[4] M. Casado et al. "Rethinking Enterprise Network Control". In: IEEE/ACM Trans. on Networking 17.4 (2009).
[5] P. Porras et al. "A security enforcement kernel for OpenFlow networks". In: HotSDN. ACM, 2012.
[6] S. Shin et al. "FRESCO: Modular Composable Security Services for Software-Defined Networks". In: Internet Society NDSS. 2013.
[7] N. McKeown et al. "OpenFlow: enabling innovation in campus networks". In: Comput. Commun. Rev. (2008).
[8] S. Sorensen. Security implications of software-defined networks. 2012. url: http://goo.gl/BiXH2.
[9] S. M. Kerner. Is SDN Secure? 2013. url: http : / / goo.gl/lPn2V.
[10] D. Kushner. The Real Story of Stuxnet. 2013. url: http://goo.gl/HIEHQ.
[11] C. Tankard. "Advanced Persistent threats and how to monitor and deter them". In: Network Security (2011).
[12] Z. Yan and C. Prehofer. "Autonomic Trust Management for a Component-Based Software System". In: IEEE Trans. on Dep. and Sec. Computing 8.6 (2011).
[13] R. Holz et al. "X.509 Forensics: Detecting and Localising the SSL/TLS Men-in-the-Middle". In: Computer Security. LNCS. 2012.
[14] M. Georgiev et al. "The most dangerous code in the world: validating SSL certificates in non-browser software". In: ACM CCS. 2012.
[15] R. Sherwood et al. FlowVisor: A Network Virtualization Layer. Tech. rep. Deutsche Telekom Inc. R&D Lab, Stanford University, Nicira Networks, 2009.
[16] Y. G. Desmedt. "Threshold cryptography". In: European Transactions on Telecommunications 5.4 (1994).
[17] F. B. Schneider. "Implementing fault-tolerant services using the state machine approach: a tutorial". In: ACM Comput. Surv. 22.4 (Dec. 1990).
[18] P. Sousa et al. "Highly Available Intrusion-Tolerant Services with Proactive-Reactive Recovery". In: IEEE Trans. Parallel Distrib. Syst. 21.4 (2010).
[19] G. Veronese et al. "Efficient Byzantine Fault-Tolerance". In: IEEE Trans. on Computers 62.1 (2013).
[20] G. Veronese et al. "EBAWA: Efficient Byzantine Agreement for Wide-Area Networks". In: IEEE HASE. 2010.
[21] R. Kapitza et al. "CheapBFT: resource-efficient byzantine fault tolerance". In: 7th ACM EuroSys. 2012.
[22] J. Hendricks, G. R. Ganger, and M. K. Reiter. "Lowoverhead byzantine fault-tolerant storage". In: SIGOPS Oper. Syst. Rev. 41.6 (Oct. 2007).
[23] P. Verissimo et al. "Intrusion-tolerant middleware: the road to automatic security". In: IEEE Security & Privacy 4.4 (2006).

[24] J. Korniak. “The GMPLS Controlled Optical Networks as Industry Communication Platform”. In: IEEE Trans. on Industrial Informatics 7.4 (2011).

[25] S. Neti, A. Somayaji, and M. E. Locasto. “Software diversity: Security, Entropy and Game Theory”. In: 7th USENIX HotSec. 2012.

[26] M. Garcia et al. “Analysis of operating system diversity for intrusion tolerance”. In: Software: Practice and Experience (2013).

[27] B. Heller, R. Sherwood, and N. McKeown. “The controller placement problem”. In: HotSDN. ACM, 2012.

[28] J. C. Mogul and P. Congdon. “Hey, you darned counters!: get off my ASIC!” In: HotSDN. ACM, 2012.

[29] J. Sanchez, I. Grida Ben Yahia, N. Crespi, Self-healing Mechanisms for Software Defined Networks‖. AIMS 2014.

[30] Xin Jin et al., ―CellSDN: Software-Defined Cellular Core Networks

AUTHORS

Haripriya N is a Research Scholar in the Department of Computer Science at DKM College for Women, Vellore pursuing M.Phil in Thiruvalluvar University, Vellore. Her special research interests are in Computer Networks and Database Management Systems. She completed her Masters in Computer Applications from C.Abdul Hakeem College of Engineering and Technology, Visharam, Vellore.

Prof. Sangeethalakshmi G is an Assistant Professor in Department of Computer Science at DKM College for Women. She has a vast experience of around 12 years in Teaching. Her areas of interest are Microprocessor and Computer Architecture.

Prof.Sivasankari A is the Head of the Department at DKM College of Women. She has a deep knowledge in the field of computers. Her interests are in DBMS, Network Security, Multimedia, Java, VC++ and Microprocessor.