Research Annals of Industrial and Systems Engineering

www.raise.reapress.com

Res. Ann. Ind. Syst. Eng. Vol. 1, No. 2 (2024) 80–87.

Paper Type: Original Article

Availability and Reliability in Software Defined Networks

Amir Abbas Shojaie[1,*], Taha Seyedsadr[1]

Institute of Communications, Transportation and Logistics, ST.C, Tehran Branch, Islamic Azad University, Tehran, Iran; Amir@ashojaie.com.

Citation:

Received: 19 July 2024 *Revised: 24 September 2024* *Accepted: 23 November 2024*	Shojaie, A. A., & Seyedsadr, T., (2024). Availability and reliability in software defined networks. *Research annals of industrial and systems engineering, 1*(2), 80-87.

Abstract

Today's computer networks experience a momentum toward a modern era. There must be some modifications in their servicing form, the network bandwidth and network delay should be increased and minimized respectively to achieve more effective performance. SDNs compared with current networks are less costly and more flexible. In SDN, the control unit is separated from data flow. Increasing reliability and accessibility are among most important SDN's challenges. In this survey various reliability forms in software have been reviewed, subsequently a SDN architecture model has been studied containing 6 hosts and a storage. The results have shown that with increase in links' number among hosts, higher the network reliability is.

Keywords: Software defined networking, Reliability, Markov chains, Availability.

1 | Introduction

Today's computer networks experience the momentum of technology and architecture to new technology and architecture and their service provision must be modified. The mobile and video world requires wider bandwidth and faster response while exposes minimal delay. Nowadays people tend to communicate via video services and applications, also cloud-based applications usage has been increased in businesses. Today, technologies are required which can direct WAN network traffic faster and more intelligently. Occurrence of problems, Shortcomings and inability in meeting the new needs has led to create new approaches to network architecture, including the Software Defined Network (SDN) [1], [2].

1.1 | Software Defined Networks

In fact, SDNs are a new architecture in computer networks that are cheaper and more flexible compared to current networks. Routers and switches play the main role in routing and transferring information in current networks, in contrast the network control section is separated form information flow section in SDN and a

Corresponding Author: Amir@ashojaie.com

https://doi.org/10.22105/raise.v1i2.42

central controller directs the network. Mostly, the central controller is an Open Source software and the routers which merely direct data, have less intelligence. In SDNs, routers and switches may be deployed virtually on the network [1], [2].

SDN components are divide into three layers:

1.1.1 | Infrastructure layer

This layer contains physical infrastructures such as switches and routers which play the role of simple and intelligent-less components. It is worth mentioning that the

routing in the network is the second layer or Control Plane duty in SDN architecture.

In general, this layer is responsible for directing packets into a path recognized suitable by controller. This layer is called Data Plane too.

1.1.2 | Control plane

It includes a central software controller which can be implemented on a server. The controller communication with the infrastructure is done through APIs. Handling data transfer through best path is up to this layer. Controller monitors data flow and network traffic dynamically in SDN architecture, so making decision and smartening is done by it and devices such as switches and routers to this issue.

1.1.3 | Application layer or management plane

This layer contains various programs and applications used on network level and often interact with end-user such as audio and video software. Mostly, business related programs belong to this layer. Usually, these programs communicate to controller via APIs.

South band API: this band's main duty is to establish communication between control layer and physical layer.

North band API: this band's main duty is to establish communication between control layer and application layer or management layer.

1.2 | Advantages of Using SDN Architecture

Advantages of using SDNs includes saving on purchasing and supporting hardware equipment such as routers and switches, optimized bandwidth utilization. A SDN could be a proper option for providing the following services [3]:

I. A good platform for cloud services provision.

II. Enterprise ICT services secure utilization through devices such as smartphones, tablets and laptops, and protecting network security.

III. Reducing and smartening network traffic.

IV. Distributing and processing Big data.

V. Providing a proper platform for Internet of Things (IoT).

But there is a long way to go before exploiting SDN-based network technology. For example some challenges such as scalability and security, reliability and availability are some of the issues which need to be addressed.

2 | Literature Review

One of the most significant challenges of traditional networks is the link failure recovery; in some cases, the network administrator interference is essential for rebuilding the data path in the network [4].

In a general view of SDN networks, it seems that these networks will provide users with customized recovery algorithms, when the failed signal is received in the recovery mechanism, the network controller directs data

into a substitution path. Same as switches and routers failure in traditional networks which causes network disruption, in SDN, misusing or attacking the controller will result in network to fail.

For this reason, network reliability can be increased through backup controllers, in these cases, coordinating and updating information among main controller and backup ones is a necessary factor.

Main controller information Backup is performed when a failure is detected [5]. Paper [6] discusses various algorithms for determining the location of controllers in order to maximize the reliability of the SDN. Among the studied algorithms, the annealing algorithm has yielded the best results.

The issue of controller substitution has been studied in various papers [7]. Authors in [8] discussed the controllers' deployment and its impact on network traffic, in [9] authors used the Pareto algorithm to deploy controllers in the network. [6] surveyed controllers deployment based on maximizing network reliability. Unlike previous works, [7] has examined the distance between the controller and switches using different algorithms based on two criteria: distances optimization and reliability maximization.

Paper [10] considers a hypothetical model with three switches, six hosts and one storage for SDN. Hosts and switches accessibility to storage is considered in two modes: first, regardless of the switches' role and second with taking them into account.

2.1 | Evaluation of System Reliability Using Markov Model

In this model, every component of the system is considered as a block with input and output terminals. The block operates like a switch, if the component is working then the switch is closed; otherwise when the component has failed to perform, the switch is open [11]. Considering the fact that every component's failure may lead to some blocks being open or closed, the whole system is assumed to be working when there is at least one path between the input and output of the model, and if there is no such a path the system is in failure situation [12].

2.1.1 | Markov model

In order to understand the Markov process, let us divide the time into three states: previous, current, and future. The future state of this process does not depend on the path it has passed before, and it solely depends on its current state [12]. Markov model is a stochastic model used to model randomly changing systems. Every state describes a disjoint combination of hale and faulty modules of the system. When the system has n modules, with each module having the possibility to be in working or failure condition, one can consider $n + 1$ states for the system. In such a system, a failure in a previously working component or maintenance of a previously failed module can cause a change in the system state [12], [13].

In the Markov model, the possibility of failure for a module with failure rate λ, during the time instance Δt is assumed to be $\lambda \Delta t$ [14]. Among the different Markov models, let us mention the discrete Markov chain and the hidden Markov model. If a system has a different state $S_1, \dots, S_n$ in each instant, that is, a discrete condition with regular distances, then the system state will change based on a set of probabilities. For a suitable description of this system one needs to know the current state along all the previous states of the system, see *Fig.1* [15], [16]

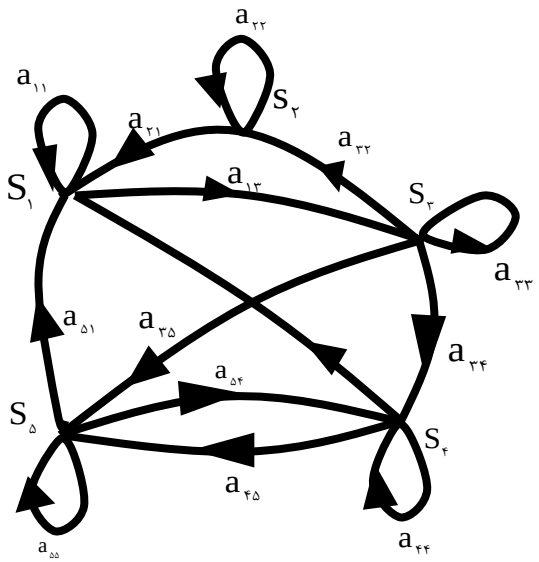

Fig. 1. A Markov chain with five states.

The Markov chain is a model whose output is a set of states where each state corresponds to an observation. One can produce a sequence of expected observations and calculate the probability of its occurrence in the Markov chain [12]. In this process, every state corresponds to an observable event, while in the hidden Markov model observations are random variables of states. Thus, the model is a stochastic hidden model and it is observable through a set of random processes that produce the sequence of observations [16], [17].

2.1.2 | Hidden Markov model

Hidden Markov model was introduced in the late 1960s and its usage in diverse applications is growing rapidly. There are two main reasons for this development: First of all, this model has a strong mathematical structure that provides the theoretical basis of many applied disciplines. Secondly, the hidden Markov model can be used for a variety of different applications, assuming it has been formed in a suitable way.

In the non-hidden Markov model, every state corresponds to an observable event, whereas in the hidden Markov model, observations are probabilistic functions of the states. Therefore, the resultant model is a stochastic model with a hidden random process and it is observable just through a set of random processes that produce the sequence of observations. In the general schematic of the hidden Markov model, shown in *Fig. 2*, different states of X (previous, current, and future) represent the hidden states and different states of Y (previous, current, and future) represent the observable states [18]

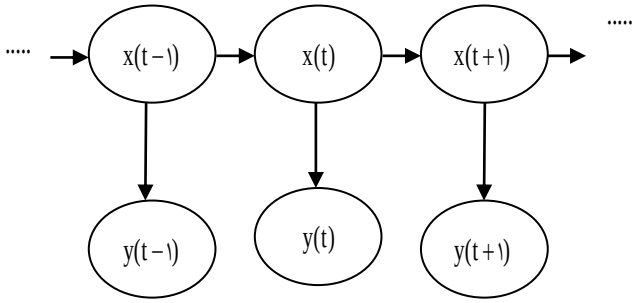

Fig. 2. The general schematic of the hidden markov model.

2.2 | Evaluation of System Reliability Using Poisson Model

It is assumed that the software's test step includes three processes: failure observation, fault discovery, and fault elimination. Software faults are divided into three categories: simple faults, independent complex faults,

and dependent complex faults. It is also assumed that no new faults could occur during the fault elimination process. Hence, the total number of discovered faults during the time interval $(0, t)$ could be obtained from the following relation [17], [19]:

$$M(t) = m_1(t) + m_2\big(t - \varphi_1(t)\big) + m_3(t - \varphi_2(t)). \quad (1)$$

2.2.1 | Modeling the elimination of simple faults $m_1(t)$

Since it is possible to eliminate the simple faults from the system immediately after being observed by a tester, one can ignore the time delay between observing the failure, discovering the fault and its elimination [20].

2.2.2 | Modeling the elimination of independent complex faults $m_2(t)$

When dealing with complex faults the software test team needs more time in order to diagnose the failure causes and eliminate them. For the independent complex faults the time delay between discovery of the fault and its elimination cannot be ignored [19].

2.2.3 | Modeling the elimination of dependent complex faults $m_3(t)$

The possibility to eliminate these types of faults is usually less than the independent complex faults, and one cannot eliminate them before resolving the cause of failure [19].

2.3 | Evaluation of System Reliability Using Petri Net

The theory of Petri nets were invented in 1962 by Carl Petri. Simplicity and high capability of using these nets to evaluate software architecture and create operational models has drawn lots of attention to them [17], [20].

3 | An Example of SDN Architecture

A common SDN architecture is represented in *Fig. 3*. Six hosts and a storage are devised to interconnect according to Table 1, in which six hosts are partitioned into three pairs. Basically, host5 and host6 are designated to manage the shared storage by system administrator. Network controller determine switches' routing topology in order to implement the SDN architecture. Assuming the network user tends to store some information on host1 or host2, the controller assigns static IPs to each host and it is capable of specifying routing paths among the host to accelerate the operation within network. Terminologically, it should be pointed out that a network is called reachable or available, in which transactions and interactions between hosts and storage system are always conducted regardless of any unexpected failure. Also, existing of more paths reaching the storage system for user, makes the network more available.

Table 1. How hosts and storage communicate.

	Host 1	Host 2	Host 3	Host 4	Host 5	Host 6	Storage
Host 1	0	0	1	0	0	0	0
Host 2	0	0	0	1	0	1	0
Host 3	0	0	0	0	1	1	0
Host 4	0	0	0	0	0	1	0
Host 5	0	0	0	0	0	0	1
Host 6	0	0	0	0	0	0	1
Storage	0	0	0	0	0	0	1

As it is shown in the table above, for instance, host1 only communicates to host3. It is hos1 and host2 responsibility to receive users' data and transfer it to storage for persistent saving with by means of other hosts, this process is shown in *Fig. 3*.

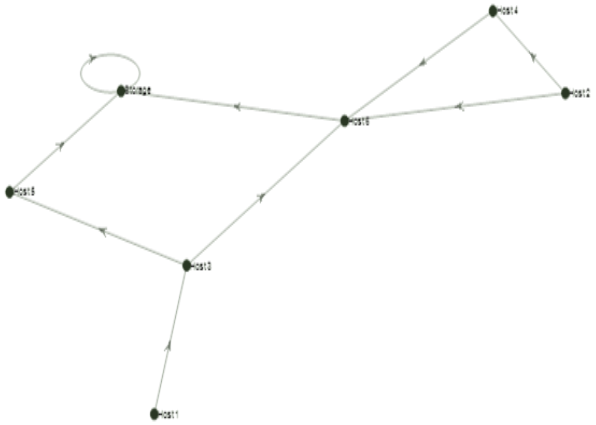

Fig. 3. How hosts and storage communicate.

Fig. 4 is representation of the table above in which every state is designated its corresponding value.

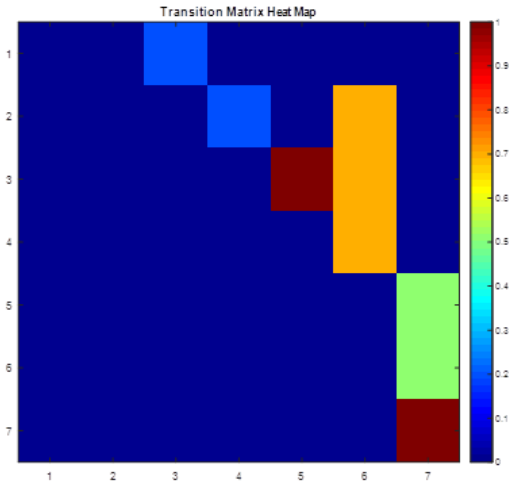

Fig. 4. Links reliability in the model.

As it is clear from *Fig. 4*, the dark blue color indicates lack of link or connection with zero probability and dark red color implies utmost Reliability.

The architecture makes link between (host1, host2), (host3, host4) and (host5, host6) pairs in order to boost reliability and its impact on overall system reliability is examined. The changes are demonstrated in *Fig. 5*.

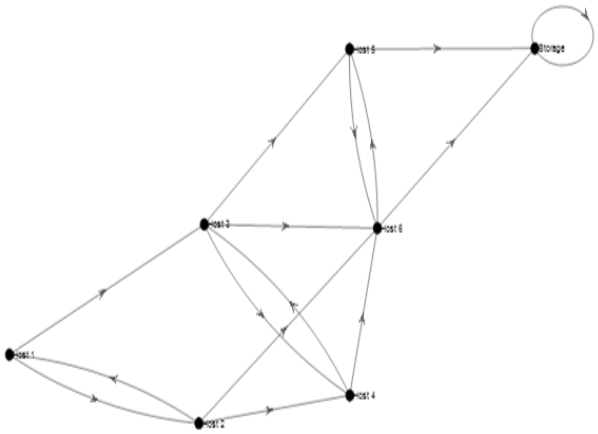

Fig. 5. How hosts and storage communicate after more links establishment.

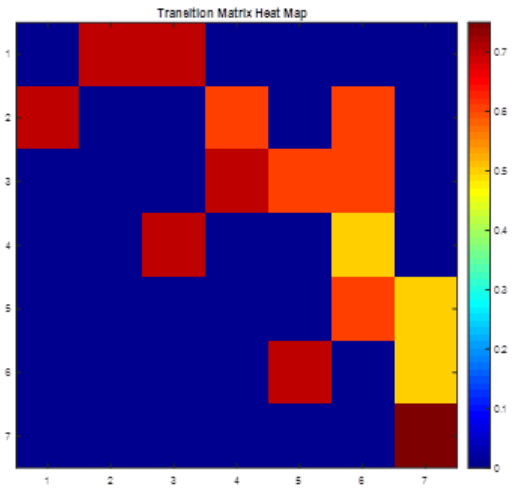

Fig. 6. The model reliability after more links establishment.

4 | Conclusion

Initially, SDN was discussed and reliability examination methods in SDN was reviewed. Eventually, a particular architecture was studied containing six hosts and a storage. Network reliability was calculated and determined by means of Markov chain. If more paths exit for user to reach to the storage, the network will be more available. The results above imply the management and control of reliability, and availability in SDN. At end, it seems to improve reliability and recovery in SDN there is need for more survey on increasing mean failure time and decreasing mean repair time, seeking higher availability to be achieved.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability

All data are included in the text.

Funding

This research received no specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] Ghasempour, M. (2016). Software-based networks (with practical workshop) level: beginner - intermediate. *Kerman province university jihad*. https://b2n.ir/j78826

[2] Alirezaei, Saeideh, Khosravi, H. (2016). Software-defined networking (sdn) review. *The second national conference on new approaches in computer and electrical engineering*. Roudsar, Iran. Civilica. (**In Persian**). https://civilica.com/doc/522613

[3] Horvath, R., Nedbal, D., & Stieninger, M. (2015). A literature review on challenges and effects of software defined networking. *Pcomputer science*, 64, 552–561. https://doi.org/10.1016/j.procs.2015.08.563

[4] Valdivieso Caraguay, Á. L., Benito Peral, A., Barona Lopez, L. I., & Garcia Villalba, L. J. (2014). SDN: evolution and opportunities in the development IoT applications. *International journal of distributed sensor networks*, *10*(5), 735142. https://doi.org/10.1155/2014/735142

[5] Valdivieso Caraguay, A. L., & Garcia Villalba, L. J. (2017). Monitoring and discovery for self-organized network management in virtualized and software defined networks. *Sensors*, *17*(4), 731. https://doi.org/10.3390/s17040731

[6] Hu, Y., Wang, W., Gong, X., Que, X., & Cheng, S. (2014). On reliability-optimized controller placement for software-defined networks. *China communications*, 11(2), 38–54. https://doi.org/10.1109/CC.2014.6821736

[7] Liu, J., Liu, J., & Xie, R. (2016). Reliability-based controller placement algorithm in software defined networking. *Computer science and information systems, 13*(2), 547–560. https://doi.org/10.2298/CSIS160225014L

[8] Heller, B., Sherwood, R., & McKeown, N. (2012). The controller placement problem. *ACM sigcomm computer communication review*, 42(4), 473–478. https://doi.org/10.1145/2377677.2377767

[9] Hock, D., Hartmann, M., Gebert, S., Jarschel, M., Zinner, T., & Tran-Gia, P. (2013). Pareto-optimal resilient controller placement in sdn-based core networks. *Proceedings of the 2013 25th international teletraffic congress (ITC)*. IEEE. (pp. 1–9). https://doi.org/10.1109/ITC.2013.6662939

[10] Nguyen, T. A., Eom, T., An, S., Park, J. S., Hong, J. B., & Kim, D. S. (2015). Availability modeling and analysis for software defined networks. 2015 IEEE 21st pacific rim international symposium on dependable computing (PRDC). IEEE. (pp. 159–168). https://doi.org/10.1109/PRDC.2015.27

[11] Khosravi, S., Kargari, M., Teimourpour, B., Talebi, M., Eshghi, A., & Aliabdi, A. (2024). GHM: an ensemble approach to fraud detection with a graph-based hmm method. 2024 10th international conference on web research (ICWR). IEEE. (pp. 99–104). https://doi.org/10.1109/ICWR61162.2024.10533348

[12] Zargar, R. H. M., & Yaghmaee Moghaddam, M. H. (2020). Development of a markov-chain-based solar generation model for smart microgrid energy management system. *IEEE transactions on sustainable energy*, 11(2), 736–745. https://doi.org/10.1109/TSTE.2019.2904436

[13] Zhang, Q., & Liu, Y. (2022). Reliability evaluation of markov cyber--physical system oriented to cognition of equipment operating status. *Computer communications*, 181, 80–89. https://doi.org/10.1016/j.comcom.2021.10.004

[14] Raj, J. S., & Rachel, I. S. (2013). A survey on reliability scheduling on grid computing. *2013 7th international conference on intelligent systems and control (ISCO). IEEE.* (pp. 331–334). https://doi.org/10.1109/ISCO.2013.6481173

[15] Kang, K., Nam, M. Y., & Sha, L. (2013). Model-based analysis of wireless system architectures for real-time applications. *IEEE transactions on mobile computing*. IEEE. 12(2), 219–232. https://doi.org/10.1109/TMC.2011.260

[16] Zaidi, P., & Javadani, T. (2017). Review of three models for assessing software reliability. *The first national conference on information technology, communications and soft computing*. Esfahan, Iran. Civilica. (**In Persian**). https://civilica.com/doc/517851

[17] Zaidi, P., & Javadani, T. (2017). Review of three software reliability assessment models. *National conference on information technology, communications and software computing*. Esfahan, Iran. Civilica. (**In Persian**) https://civilica.com/doc/517851

[18] Darwish, M. M. G. S. (2017). Intrusion detection using the phi markov model. *The 9th national command and control conference of Iran.* Tehran, Iran. Civilica. (**In Persian**) https://civilica.com/doc/661340/

[19] Mieghem, P. (2006). *Performance analysis of communications networks and systems.* Cambridge University Press. http://dx.doi.org/10.1017/CBO9780511616488

[20] Khatrian, S., Yaghoubi, T. (2014). A new model for evaluating software reliability based on the inhomogenous poisson process. *Conference on computer engineering and sustainable development with a focus on computer networking, modeling and systems security.* Mashhad, Iran, Civilica. (**In Persian**). https://civilica.com/doc/239073

[21] Akhtarian, S., & Yaghoobi, T. (2016). A generalized model for software reliability evaluation based on non-homogeneous poisson process. *Journal of statistical sciences, 10*(1). (**In Persian**). http://jss.irstat.ir/article-1-250-en.html

[22] Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G., & others. (1995). *Modelling with generalized stochastic Petri nets*. John Wiley & Sons Inc. https://iris.unito.it/handle/2318/25289