International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 10, No. 1, March 2021, pp. 32~36
ISSN: 2089-4864, DOI: 10.11591/ijres.v10.i1.pp32-36

Systematising troubleshooting of disputes in network

Sowmya K. B., Thejaswini A.
Department Department of Electronics and Communication Engineering, R V College of Engineering Bengaluru, Karnataka, India

Article Info

Article history:

Received Jun 2, 2020
Revised Oct 29, 2020
Accepted Nov 27, 2020

Keywords:

Distributed management information tree
Managed objects
Software defined networking

ABSTRACT

With the growing network size, virtualization everywhere, it is getting more difficult to configure and manage the network devices. Software Defined Networking (SDN) is a way to address these problems. Application Centric Infrastructure (ACI) is the Cisco's solution to SDN, with centralized automation and policy-driven application profiles. If there is any bug in the network or problem with the expected functionality of the network, ACI cases are opened in the Technical Assistance Centre (TAC) for troubleshooting the issue. Engineers currently troubleshoot ACI cases manually by using Command Line Interface (CLI) and trace for different events triggered by the policy pushes by logs generated at different stages of the ACI and from different servers responsible for this, which indeed is a very tedious, time consuming task and is prone to manual errors. This paper describes a way to automate the entire ACI troubleshooting process with the user-friendly GUI which can show the entire information needed for troubleshooting by extracting relevant information at every layer. By making use of FSM models the proposed solution can be extended to other areas which involve log analysis using CLI to extract relevant information and is not just limited to ACI.

Corresponding Author:

Sowmya. K. B
Department of Electronics and Communication Engineering
R V College of Engineering
Mysore Rd, RV Vidyaniketan, Post, Bengaluru, Karnataka 560059, India
Email: kb.sowmya@gmail.com

1. INTRODUCTION

The Software Defined Networking (SDN) is a networking approach that separates data plane and control plane which enables the data plane devices to be directly programmable by the control plane and the infrastructure below to be abstracted for network and application services. Application Centric infrastructure (ACI) is SDN solution of Cisco. While troubleshooting or solving an ACI related case, Distributed Management Environment (DME) logs are considered for audit. ACI is adopted in networking because it is characterized by simplified automation by an application-driven policy model, real-time centralized visibility, open software flexibility, scalable performance and multi-tenancy in hardware etc, but the complexity involved in solving an ACI related case is reflected in the logs itself, which are usually the outputs of Managed Objects (MOs) arranged systematically in files and folders. Assuming a problem started at a time and is associated with the configuration made by the person who has configured the network, this configuration that is pushed is not localized to just one device, instead it is pushed to all other nodes that might be affected by the ACI which includes redundancy measures also. To exactly identify the problem usually 4 different kind of files are considered: Access logs, Nginx, Policydistr, Policymgr which are associated with ACI policy pushes. These 4 logs give us a description of how the config was distributed

[1-4]. But each log is an output of different managed objects running asynchronously including the redundant measures too. As the systems get more intelligent and complex, the outputs given by each process also gets increasingly complex. Having less readable complex and inherently difficult logs is not only a problem for customer alone, nstead increases traffic at TAC centers because even a minor problem may create incomprehensible log output. From this project, device configuration pushes on ACI products can be visualized with a GUI, without having to look for data in a Command Line Interfaces (CLI). The project is built on dynamic roots with scalable and reliable methodologies like FSM and JSON representation, so that it can be scaled further on stretching to other networking technologies other than ACI wherever log parsing and visualization is done. The aim is to incorporate the project into app section of APIC GUI itself so that the time involved in downloading and decrypting them can be narrowed down.

2. SOFTWARE DEFINED NETWORKING AND PERFORMANE OF ACI

2.1. Necessity and features of SDN

There is a necessity of connected devices with the growing technological demands. As a result, networks and internet are expanding at the increasing rate. With such a huge network size, it is hard to configure, control and manage network devices. Software Defined Networking (SDN) architecture provides a solution to this by separating the control and data forwarding capabilities of networking devices. In traditional network, these capabilities were vertically stacked on the devices and hence not giving enough flexibility [5-8]. SDN not only helps in configuring devices by a software method, but also helps in the maintenance and management of the network itself in case anything goes wrong with device or any faults arise etc. since the whole control plane capabilities is vested in the hands of the centralized controller.

To facilitate this, it makes use of two types of Application Policy Interfaces (APIs) viz Southbound APIs and Northbound APIs. The Southbound APIs are responsible for providing the instruction sets to the functionality of forwarding devices. It also ensures smooth communication control plane devices and forwarding devices by defining communication protocol between them, and hence are the important elements in separating data plane and control plane capabilities of the devices. Whereas, the Northbound APIs are used by the application developers for developing applications relevant to the network. In most of the cases, northbound APIs program the devices responsible for forwarding by abstracting the instruction-set used by southbound APIs, at a lower level [9-13]. Also, network hypervisors are responsible for layer III functionality of the SDN by supporting various virtual machines to share a common hardware resource.

2.2. Features of Application Centric Infrastructure

Combines the full mesh spine-and-leaf topology with finely tuned routing and several protocols that allow network abstraction from physical infrastructure. This flexible and redundant network enables grouping of hosts to apply communication policies that define the device the devices that can communicate with each other. Even though ACI uses a completely different communication model from the outside network, the entire ACI topology appears as a regular switch to the outside devices. At a very basic level, ACI can be thought of as a Clos network of Nexus 9000 Series Switches with an Application Policy Infrastructure Controller (APIC) management platform. In a Clos network, every lower tier switch is connected to every upper tier switch, creating a redundant, high performance mesh where traffic is distributed between links [14-18].

The APIC, or network management platform, provides customers with a single place from which to manage the network. Like the Unified Computing System (UCS), the APIC uses abstraction to ease the configuration burden, while pushing down profiles that are tailored to applications that will be instantiated on the individual switches. In doing so, ACI solution becomes much more than a Clos network of Nexus 9000 Series Switches with the management platform [19-21]. Instead, it becomes a fully integrated, open system. ACI supports the same access tools that are available on individual Nexus Series Switches with a management platform. Instead, it becomes a fully integrated, open system.

3. MANAGED OBJECTS AND DISTRIBUTED MANAGED INFORMATION TREE FOR TROUBLESHOOTING

Classes are used to abstract and modeling everything include policy, resource, topology, event and faults in ACI. Each object is an instance of the certain class that inherits the properties and method. Together all objects store the configuration and run-time data of the ACI fabric. The following gives the description of how managed objects (MOs) are organized. Everything is an object. Objects are hierarchically organized. Distributed Managed Information Tree (dMIT) contains comprehensive system information like discovered components, system configuration and operational status including statistics and faults.

A single logical dMIT presented to user through REST interface on any APIC. Internally dMIT is split into various services and shards in various APICs. In APIC MIT, everything is fully defined as an object. These definitions include the entity itself, and a full description of the entity, including its configuration, state, run-time data, description, reference objects, and lifecycle position. The tree is distributed across the entire APIC cluster, so it may be referred to as the DMIT. The objects are organized in a hierarchical way, creating logical object containers. Every object has a parent, except the top object, root, which is the top of the tree. Objects include a class, which describes the type of object such as a port, network path, or Packages identify the functional areas to which the objects belong.

The DMIT also contains comprehensive system information about discovered components, system configuration, and operational status including statistics and faults. The unified RESTful API automatically delegates a request to the corresponding components in the MIT. Every component and attribute are represented as an object in the Cisco APIC MIT and every object can be manipulated via REST.

4. RESULTS AND INFERENCE

Depending on the affected node, timestamp of the event, changeset etc. the steps involved in automating the troubleshooting process varies. Results for the changeset "conversion from proxy to flood" follows the steps of finding timestamp from aaaModLR server, pull out the POST URL from the JSON file, collect message ID, transaction ID etc. where each data extraction from the files is dependent on the previously collected data. Hence, a Finite State Machine (FSM) is built where these procedures can be defined depending on the previous triggers.

These FSMs play an important role in automating the entire procedure. Figure 1. shows how a typical FSM looks like for the data extraction from Nginx server, where the inputs are HTTP method, timestamp and the affected node, the output is the Message ID which will be later used for the Policy distributer server. These FSMs in the form of a JSON file can be directly imported to the python script responsible for log parsing. To create the above-mentioned FSM and the FSMs related to other configuration pushes, a web UI is developed, where one can define the FSM states, add events and add relevant call backs to define the set of operations which have to follow one another on trigger [22-25].

Figure 1. Data extraction from nginx server

Figure 2. shows the screenshot of the web UI to create the FSM easily. The output of this page will be a JSON file which the user can download directly and use them in the python scripts which has the DMIT and managed objects handled efficiently. Once the DMIT is built, it is integrated with data from different servers like Nginx, AAA, Message broker, ConfigConfMO, Intra Fabric Messager (IFM) etc. of the data center in the ACI fabric.

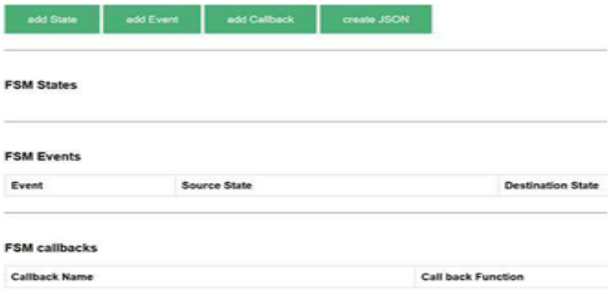

Figure 2. Screenshot of the web UI to create the FSM

The required information at different stages of the ACI policy push configuration be viewed easily as the data flow path is clearly defined and relevant information and keywords are displayed clearly. While traditionally, engineers used to grep for the required data across lots of showtech files and folders with the help of Command Line Interface (CLI). As it is hovered over every node in the GUI, relevant information at that stage is displayed. The suitable JSON files corresponding that stage can also be viewed by zooming into that node of the GUI.

From the above-mentioned results, it can be inferred that the entire ACI troubleshooting process can be automated with the user-friendly GUI which can show the entire information needed for troubleshooting by extracting relevant information at every layer of the ACI. This information needed is not a general one which is common to all the cases, hence this issue is addressed by making use of finite state machines which considers all the possible cases and the data flow path is defined for each of them. By making use of FSM models the proposed solution can be extended to other areas which involve log analysis using CLI to extract relevant information and is not just limited to ACI. Different FSMs can be designed to suite different applications and finally a JSON file will be generated which has all the relevant information in the form of a dictionary. Finally, all the required data can be viewed in a readable format by uploading this JSON file to the GUI other than doing it manually. Some notable features of this automation tool are to Reduce time complexity while increasing value addition to the ACI team Scalable to other technologies.

5. CONCLUSION

ACI architecture ensures agility, programmability, better performance and reduced complexity. This is based on the application centric model where application requirements decide the network not the other way around. As is a result it is the future to all the networking topologies as it doesn't include configuration of individual physical devices explicitly, rather are automatically done whenever new policy push is done to the application policy infrastructure controller, which saves a great amount of time and eradicates the errors which may occur through manual configuration. This work was an attempt to save the time of engineers even more during the troubleshooting process of the issues raised due to the configurational changes or policy pushes made to the ACI fabric. Generally solving an ACI case approximately takes two weeks to grep through all the logs for audit. By using the above described automation tool, this process is going get faster and more reliable.

REFERENCES

[1] A. Pras, J. Schonwalder, M. Burgess, O. Festor, G. M. Perez, R. Stadler, and B. Stiller, "Key Research Challenges in Network Management," *IEEE Communication Magazine*, vol. 45, no. 10, pp. 104-110, 2007.

[2] B. Jennings, S. V. D. Meer, S. Balasubramaniam, D. Botvich, M. O Foghlu, and W. Donnelly, "Towards Autonomic Management of Communications Networks," *IEEE Communication Magazine*, vol. 45, no. 10, pp. 112-121, 2007.

[3] S. Adams, *ITIL V3 Foundation Handbook*, The Stationery Office, 2009.

[4] D. Opeenheimer, A. Ganapathi, and A. Petterson, "Why do internet services fail and what can be done about it?" *in USITS'03: Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems, Seattle, WA, USA*, vol. 67, pp. 1-16, Mar. 2003.

[5] J. Strassner, "How policy empowers business-driven device management," *in Third International Workshop on Policies for Distributed Systems and Networks*, 2002, pp. 214-217.

[6] Z. Kerravala, *As the value of enterprise networks escalates, so does the need for configuration management*, The Yankee Group, Jan. 2004.

[7] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *in Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, 2014.

[8] N. El Moussaid, A. Toumanari and M. El Azhari, "Security analysis as software-defined security for SDN environment," *2017 Fourth International Conference on Software Defined Systems (SDS), Valencia*, 2017, pp. 87-92, doi: 10.1109/SDS.2017.7939146.

[9] O. Michel and E. Keller, "SDN in wide-area networks: A survey," *2017 Fourth International Conference on Software Defined Systems (SDS), Valencia*, 2017, pp. 37-42, doi: 10.1109/SDS.2017.7939138.

[10] R. L. Smeliansky, "SDN for network security," in *2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), Moscow*, 2014, pp. 1-5, doi: 10.1109/MoNeTeC.2014.6995602.

[11] D. B. Hoang and M. Pham, "On software-defined networking and the design of SDN controllers," in *2015 6th International Conference on the Network of the Future (NOF), Montreal, QC*, 2015, pp. 1-3, doi: 10.1109/NOF.2015.7333307.

[12] D. R. Garrison and M. F. Cleveland-Innes, "Foundations of distance education," In M. F. Cleveland-Innes and D.R. Garrison, Eds., New York, NY: Routledge, *An introduction to distance education: Understanding teaching and learning in a new era*, 2010, pp. 13-25.

[13] D. R. Garrison and H. Kanuka, "Changing distance education and changing organizational issues," New York, NY: Routledge, In W. J. Bramble and S. Panda, Eds., *Economics of distance and online learning: Theory, practice, and research*, 2008, pp. 13-25.
[14] P. Goodyear, S. Banks, V. Hodgson, and D. McConnell, "Research on networked learning: An overview Dordrecht," Netherland: Kluwer Academic Publishers, *Advances in Research on Networked Learning*, 2004, pp. 1-9.
[15] D. Harris, "Transforming distance education: In whose interests?" *In* T. Evans, M. Haughey and D. Murphy, Eds., *International handbook of distance education Bingley, UK: Emerald*, 2008, pp. 417-432.
[16] B. Holmberg, "A theory of teaching-learning conversations," In M. G. Moore, Ed., *Handbook of distance education* 2rd ed., Mahwah, NJ: Erlbaum, 2007, pp. 69-75.
[17] P. Ice, "The future of learning technologies: Transformational developments," In M. F. Cleveland-Innes and D.R. Garrison, Eds., *An introduction to distance education: Understanding teaching and learning in a new era*, 2010, pp. 137-164.
[18] C. Jones, *Networked learning: An educational paradigm for the age of digital networks*, *Verlag*, Switzerland: Springer, 2015.
[19] K. Swan, "Teaching and learning in post-industrial distance education," In M. F. Cleveland-Innes and D. R. Garrison, Eds., *An introduction to distance education*: Routledge, 2010, pp. 108-134.
[20] H. Kanuka C. Brooks, "Distance education in a post-Fordist time: Negotiating difference," In M. F. Cleveland-Innes and D. R. Garrison Eds., *An introduction to distance education: Understanding teaching and learning in a new era*, 2010, pp. 69-90.
[21] P. Trowler, "Wicked issues in situating theory in close-up research," *Higher Education Research & Development*, vol. 31, no. 3, pp. 273-284, 2012.
[22] A. Woodley, "But does it work? Evaluation theories and approaches in distance education," In T. Evans, M. Haughey and D. Murphy Eds., *International handbook of distance education*, 2008, pp. 585-608.
[23] T. Ryberg and C. Sinclair, "The relationships between policy, boundaries and research in networked learning," *In T. Research, boundaries, and policy in networked learning*, 2016, pp. 1-20.
[24] S. Guri-Rosenblit, "Distance education in the digital age: common misconceptions and challenging tasks," *Journal of Distance Education*, vol. 23, no. 2, pp. 105-122, 2009.
[25] S. Guri-Rosenblit, *Digital technologies in higher education: Sweeping expectations and actual effects*, Nova Science Publishers, Inc., 2011.

BIOGRAPHIES OF AUTHORS

Ms. Sowmya K B received the B. E degree in Electronics and Communication Engineering from the Vivekananda College of Engineering and Technology, Puttur, India, in 2006, M. Tech degree in Electronics from the Sir. M. Visveswaraya Institute of Technology, Bengaluru, India, in 2012, bagging rank from VTU, Belagavi. She was working as Assistant Professor in PA College of Engineering, Mangaluru, India for Nine years. Currently she is working as Assistant Professor in R V College of Engineering, Bengaluru, India. Her research interests include VLSI and Signal Processing, Image processing, Low power Architectures and VLSI Design. She has contributed many National and International journals to various reputed journals and conference.

Ms. Thejaswini A is doing her Bachelor of Engineering in Electronics and Communication Engineering Department, RV College of Engineering, Bengaluru, India.