Journal of Innovative Engineering and Research (JIER)
Vol.- 4,Issue -2, October 2021, pp. 1-5 (5 pages)

An Analytical for Traffic Engineering Techniques in Software Defined Networks

PRAGATI GANDHI[1], NEETU GYANCHANDANI[1], SANTOSH PAWAR[2], RITESH YADAV[2]

[1] *J. D. College of Engineering & Management, Nagpur 441501, India*
[2]*School of Engineering, Dr. A. P. J. Abdul Kalam University, Indore 452016, India*
Corresponding Author Email: gandhipragati92@gmail.com

Abstract*—Several maintenance activities in today's network infrastructures often necessitate substantial human interaction and can be extremely complex, requiring the consideration of several inputs to achieve effective configurations. In this regard, this thesis provides an optimization architecture capable of providing network administrators with effective and stable routing configurations automatically. Traffic engineering is crucial in assessing a network's efficiency and stability. How to deal with traffic rerouting in the event of a network breakdown is a big problem in traffic engineering. TEFR (Traffic Engineering for Fault Recovery) displays rerouted traffic after a network failure. The full flow form is used in this topology's Universal Single-link Traffic Rerouting (USTR). At the same degree of optimality, USTR performs better and with less complexity than other programs, allowing all efficient network connections to be secured. Since the maximum flow approach was used to solve the traffic rerouting problem, it rigorously proves the correctness and sophistication of USTR. We test USTR on real-world network topologies with a large number of ties and nodes, and the running time is very satisfactory. In particular, we use SDN to construct a prototype of USTR.

Index Terms*— Traffic Engineering, Software Defined Networking, Linear Programming, Failure Recovery, Optimal Rerouting

I. INTRODUCTION

IP-based networks are becoming the primary networking infrastructures used for an increasing range of diverse applications and services. This situation increased the need for reliable and integrated resources capable of assisting network maintenance activities and ensuring the proper preparation of resilient network infrastructures [1]. In this case, multiple elements must be properly calibrated and organized to achieve appropriate network service levels. Regardless of the numerous specific solutions available to ensure adequate network efficiency, the efficient configuration of routing protocols remains critical in the networking sector. Indeed, correct routing configurations are needed to increase network resource utilization while also providing upper-layer protocols, software, and overlay systems with a reliable, resilient, and optimized communication infrastructure. Many autonomous systems (ASes) now depend on traffic engineering (TE) to pick routes that make the most use of their network resources. This is especially significant considering the high cost of network infrastructure and the fierce competition in the Internet ISP industry. Many experiments have been inspired by the

relevance of traffic engineering in recent years, and several traffic engineering algorithms have recently been proposed. The architecture of traffic engineering algorithms is heavily influenced by traffic characteristics. Unfortunately, although most ASes' traffic demand is relatively steady most of the time, there are times when traffic is highly volatile, with unexpected traffic surges that ramp up very rapidly, leaving little time for a traffic engineering algorithm to re-compute or adapt. Nowadays, there is an increase in the number of distributed network apps and utilities, and efficiency is an important factor in this. This clustered application place greater demands on network efficiency, such as lower packet delay. Various search engines, e-commerce platforms, and video gaming, for example, suffer from content loss as network latency increases. Regardless of network faults, many traffic engineering topologies are used to ensure network efficiency. However, to ensure continuous QoS, network success must be immune to network errors. As a result, we suggest the Universal Single-link Traffic Rerouting (USTR) solution, which is based on the maximal flow method to compute an optimum BPT. USTR produces an optimum solution in the operating time of where is a divergence from the optimal and cgap is the ratio of the network's maximal potential to minimal capacity. When all have the same degree of optimality, the numerical complexity of USTR is one or two orders of magnitude smaller than that of LPTR. We rigorously prove the feasibility and accuracy of USTR, which transforms the problem of computing an optimum BPT for a connection on a given topology and a given traffic vector into the problem of computing a maximum flow in a certain flow network derived from certain technical methods, since it is the first time it has been used to solve the single-link failure problem. Extensive trials and tests are carried out to validate the efficacy of our method. Since unified TE is commonly used in practice, we use software defined networking (SDN) technologies to execute our USTR solution, in which the controller has a global network view and configures all subordinate switches. Open-Daylight (ODL) is a shared open source SDN controller that supports Open Flow. This allows the ODL controller to manage not only Open Flow switches but also standard routers (switches) that follow the Netconf protocol. As a result, we implement our USTR algorithm in an ODL controller and pre-install BPTs using label forwarding

technology. The following are the paper's contributions.

•We suggest USTR for calculating optimum BPTs effectively. When all have the same degree of optimality, the numerical complexity of USTR is one or two orders of magnitude smaller than that of LPTR.

•We rigorously prove the viability and correctness of USTR. Extensive trials and tests are carried out to validate the efficacy of our processes.

• Specifically, we use SDN to incorporate and validate the USTR prototype. It is compatible with OpenFlow, MPLS, and IP networks.

II. LITERATURE REVIEW

In this paper [5] the author suggests Universal Single-interface Traffic Rerouting, a highly effective traffic rerouting method for dealing with an ideal BPT for a link on a given topology and a given traffic vector. USTR will work out a (1+)-optimal BPT that has an indistinguishable dimension of optimality from LPTR. In terms of TE implementation, USTR is comparable to LPTR and much superior to CSPF. In terms of processing time, USTR is a couple of requests faster than LPTR and almost equal to CSPF. USTR's running time is entirely justified in order to secure all links in a TE interim. In this case, the developer employs SDN to run a USTR model and test its execution. The controller is ODL since it can operate in OpenFlow, MPLS, and IP networks. ODL uses VLAN ID to perform name sending in the reinforcement guiding knowledge plane. IP fast reroute techniques are used in this paper [6] to recover bundles in the information plane after connectivity failures. The previous analysis included mechanisms forensuring disappointment recovery from at most two-connect disappointments. We create an IP fast reroute strategy that utilizes proven curve disjoint spreading over trees to ensure recovery from up to (k-1) device failures in a k-edge-connected network. Since bend disjoint traversing trees can be built in sub-quadratic time to the extent of the network, our methodology is extremely adaptable. Using exploratory outcomes, we show that using circular section disjoint crossing trees to recover from different disappointments decreases way more than previously proven approaches. If a switch or a link fails in a circuit exchanging or package exchanging network, the parcels that are passing through the failed connection, path, or switch are lost and dropped before the network self-re-meets, regardless of whether there is a substitute path keeping a strategic distance from or bypassing the network failure. During this time, the specific targets would be inaccessible from the source, causing network traffic and degrading network efficiency parameters. The network re-merges phase necessitates a significant amount of energy ranging between a few milliseconds to several seconds. To address this issue, the Internet Engineering Task Force (IETF) created the IP Fast ReRoute (IPFRR) framework [7]. Many steering conventions are created to recover a network from such a network failure using the IPFRR system. Strategies for "the network structure problem with availability prerequisites" are considered in this paper [8]. These ILP- and multi-commodity stream-based strategies design a low-cost network (from the start) that is ready to transmit requests without clogging under a specified set of disappointment circumstances, each of which includes a large number of communication disappointments. Following two edge disappointments, a few proposals for providing networks have been suggested. In general, these do not fix anticipated clog (other than by multiplying limits where streams cover). Furthermore, they do not necessarily add up to disappointments at a larger caliber. A few articles have been proposed to extend the development of disjoint trees. This investigation [9] demonstrates that a solution occurs consistently for all twofold edge disillusionments as long as the network graph is 3-related, and a heuristic is given. Another option for dealing with disillusionment in pre-configured support routes is to reconfigure the paths. This paper discusses and outlines such a theory as the preface to our arrangements. Reconfiguration offers an amazing structure since the fortification course for the second failed edge is chosen subject to accurately learning what has already tumbled (instead of dealing with every possible dissatisfaction). Reconfiguration strategies, in like fashion, add up to more viable to higher-assortment annoyances. Regardless, as our most cynical condition assures, the focus is on assessment and structure of reconfiguration designs provided a settled veiled network. The study in progress also considers re-provisioning of support paths, but only up to the association stage. There is a large body of literature on "p-cycles." A p-cycle is a preconfigured cycle of unit limit constructed from as much of the network as possible. It provides fast remaking of dissatisfactions on the loop in the same way as "straddling" ranges of both end-centers around the cycle do. The primary framework was suggested for single disillusionments [10] however, it has since been expanded to twofold dissatisfactions and SRG (shared peril gathering) frustrations. Although one of our arrangements additionally makes use of pulses, it differs in a significant way from p-cycles. As far as everyone is concerned, there is no p-cycle progression ensuring blockage-free modifying for an arbitrary number of annoyances. Furthermore, while other papers on p-cycles suggest a set of direct tasks (ILPs) to locate "perfect" cycles in a given network, we propose a simple (and nearly perfect) plan of edges that can be integrated with assured execution and without an estimate. In this paper [11], an ILP is presented that gives a basic stream errand (with stream part) and a "stream redistribution" plot that guarantees no blockage with k frustrations under the following condition: provided any set F of edges with maximum scale usage up to k, all streams can be guided on F without stopping. This work is similar to, but not identical to, ours; while we suggest a network strategy, they provide confirmation on a network satisfying a pre-condition without precisely holding an eye on the best way to assemble such a system. In this paper [12], a fast-reroute scheme is used to build up a reinforcement path when interface disappointments occur; however, it is not

effective for various disappointments that occur as often as possible in spine networks. Consider a convention for reconfiguring impaired reinforcement ways after a communication failure, thus increasing survivability from a subsequent failure. Switch-to-switch connections in the spine network carry traffic from various start-to-finish relationships. If a device failure occurs, any of the associations that are navigating it the failure links also fizzle. The primary focus is on recouping start to finish partnerships using way protection technologies. Even though insurance is successful in asset use, it has the drawbacks of higher multifaceted type, low adaptability, and long recuperation periods necessitate. In link stability, MPLS fast reroute is used to pre-process exchange ways to deal with double connection disappointments, which are becoming increasingly complex. Since a first connection disappointment may affect the reinforcement method of the second connection, the pre-processed reinforcement methods for each connection will need to consider any possible combination of disappointments from different connections. The authors of this paper [13] describe the best method for developing ideal sets of trees for weighted organized maps. Partitions incidence edges to each hub into "insurance charts," one of which is available after a disappointment: the first is handled by guarantee graphs, and the second by disjoint trees. Considers the improvement problem of locating the greatest number of preconfigured FRR reinforcement methods in the following case. Assume that a second disappointment, e2, happens along the reinforcement path of a first fizzled tip, e1, which is corrected using e2. At that point, this reinforcing method should rule out e1. A heuristic is given.

III. Proposed Methodology

This section discusses the system overview in detail, proposed algorithm, and mathematical model of the proposed system.

A. System Overview

Figure 1. shows the architecture of the proposed system. The explanation of the system is as follows:

- Network Generation - Initially network is generated vertices/nodes are connected with the edges.
- Path Generation -After generating the source and sink node. Generate all possible paths from source to sink node.
- Get Shortest Path- Find the shortest path from all generated paths from source to sink node.
- Key generation and distribution - Key Generation Center generates the keys and distributes the keys to each node. Perform the route generations from the source to the sink node.
- Data Encryption - Data is generated at each node. After generating the data, data is encrypted at each node by using the ECC algorithm.
- Energy Consumption - Calculate how much energy is consumed for each sensor node along the shortest path.
- Data Authentication - After evaluating the hash value at the source node, the sink node verifies the authentication of data.
- Data Decryption –The Sink node receives the data from the source node and decrypts the data by the appropriate key.

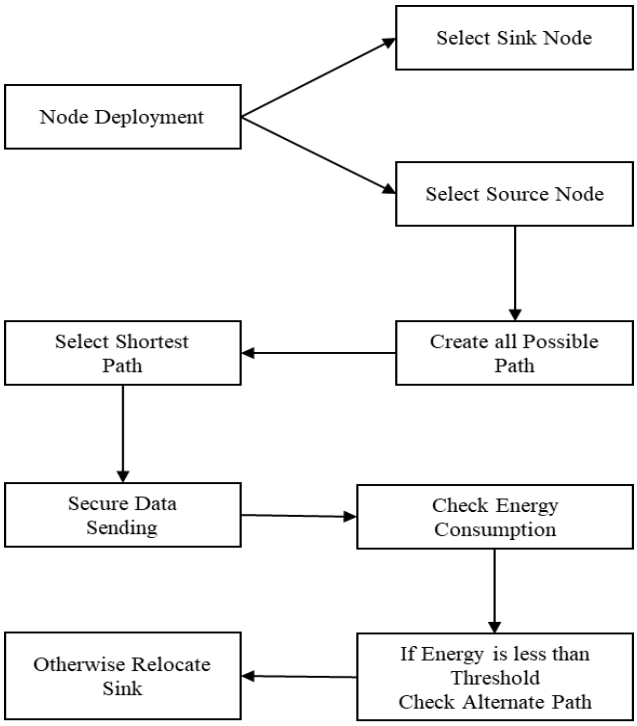


Figure 1: System Architecture

B. System Implementation

1) Mathematical Formulation

System S is represented as

S= {N, S, D, P, Sp, K, d}

- Deploy nodes

N = {N1, N2, ... ,Nn}

N is set of all deployed nodes.

- Create Source

S = {S1, S2…Sn}

Where S is a set of all Sources.

- Create Sink Node

D = {D1, D2... Dn}

Where D is a set of all sink nodes.

- Find all Paths

P = {P1, P2,....,Pn}

Where P is a set of all Paths.

- Select Shortest Path

Sp = {Sp1, Sp2, Sp3... Spn}

Where Sp is the set of all Shortest Path.

- Generate the keys for authentication

K = {K1, K2... Kn}
Where K is a set of all Keys.

- Send the data from source to sink node.

d= {d1, d2, d3 ...dn}
Where F is a set of all data transmitted.

2) Proposed Algorithm

- Create a network graph as Graph g(v,e) where; V are vertices/nodes and E are edges.
- Select Source and sink node from all sensor nodes.
- Generate all paths from source to sink node.
- Select the Shortest path from all generated paths.
- Generate public/private keys and distributes them to the source and the sink node.
- Generate the data at the source and send it to the sink node via the shortest path.
- Encrypt the data with the private key.
- Calculate the energy consumption of each node which are present in the shortest path.
- Decrypt the private key and Authenticate received data at the sink node.
- If energy is going to die of sensor nodes from the path, then select the alternate path.
- Send the data from source to sink node through the alternate path and calculate energy consumption.
- Again energy expires of sensor nodes of alternate path.

Description:
The above Algorithm describes the steps of the proposed system. Initially, the network is created with sensor nodes, source, and sink nodes. Then we generate all paths from source to sink node and select the shortest path for data sending purposes. Sensor nodes are not efficient if energy consumption is more.

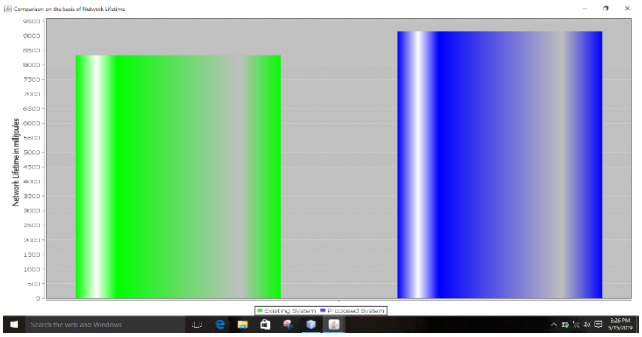
Therefore, the system selects the alternate path for communication between sources and sink nodes together calculating energy consumption and residual energy levels at each sensor node. The encryption algorithm encrypts the data by using the ECC algorithm with the private key. Data is verified by its hash value. Only verified data is accepted by the sink node. The decryption of the data is performed at receiving sensor node with the appropriate keys. Again sensor nodes of the alternate shortest paths are expected to get expired and finally we trigger the procedure of sink relocation.

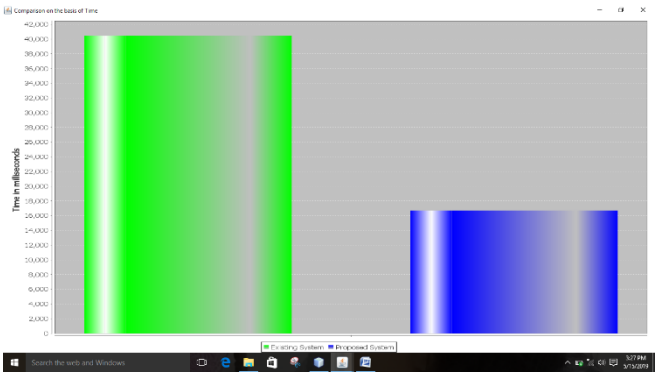
3) ECC Algorithm

- Sender and Receiver
  Calculate edB = S = (S1, S2)
- Sender sends a message M E to Receiver as follows:
  Calculate (S1 * S2) mod N = K
  Calculate K * M = C, and send C to Sender.
- Receiver receives C and decrypts it as follows:
  Calculate (S1 * S2)modN = K
  Calculate (K-1)modN
  (Where N = E)
- K-1*C = K-1*K*M = M

IV. Experimental Results

The Experiment is conducted on i5 machine, with 8 GB RAM and 256 GB SSD. The experiment is being done using the JUNG Simulator and the entire implementation has been done using java language. The results of the experiment are obtained based on 3 parameters which are time utilization, energy utilization, and residual energy or network lifetime. The results of the experiment are shown in the following graphs:





V. CONCLUSION

The study presented here is a review of existing literature and researches conducted in the area of traffic engineering. Some of the methods are applicable for traditional network structure whereas some are related to the techniques developed for SDN. The challenges related to the implementation and future enhancement required for the up-gradation of the systems are also discussed. Software Defined Networking (SDN) is a promising approach in the networking paradigm. It distinguishes the control plane of the network from the plane which is used for data forwarding. It enables and provides the solution for many problems in the traditional network architecture. It reduces the complexity in network management by managing the network centrally. It also

presents the network programmability and providing a global view of a network and its state.

REFERENCES

[1] T. Elhourani, A. Gopalan, and S. Ramasubramanian, "Ip fast rerouting for multi-link failures," in IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, April 2014, pp. 2148–2156.
[2] Q. She, X. Huang, and J. P. Jue, "Survivable routing for segment protection under multiple failures," in OFC/NFOEC 2007, March 2007, pp. 1–3.
[3] H. Kim, M. Schlansker, J. R. Santos, J. Tourrilhes, Y. Turner, and N. Feamster, "Coronet: Fault tolerance for software defined networks," in ICNP '12, Oct 2012, pp. 1–2.
[4] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks", IEEE Commun. Surv. & Tutor., vol. 16, no. 3, pp. 1617–1634, 2014.
[5] Anmin Xu, Jun Bi, Baobao Zhang, Tianran Xu, Jianping Wu, "USTR: A High-performance Traffic Engineering Approach for the Failed Link", 38th International Conference on Distributed Computing Systems, IEEE 2018.
[6] Theodore Elhourani, Abishek Gopalan, Srinivasan Ramasubramanian, "IP Fast Rerouting for MultiLink Failures", IEEE/ACM Transactions on Networking Volume: 24, Issue: 5, October 2016.
[7] Mahesh Bhor, Deepak Chatrabhuj Karia, "Network recovery using IP fast rerouting for multi link failures", International Conference on Intelligent Computing and Control (I2C2), IEEE Xplore: March 2018.
[8] Rakesh K. Sinha, Funda Ergun, Kostas N. Oikonomou, K. K. Ramakrishnan, "Network Design for Tolerating Multiple Link Failures Using Fast Re-Route (FRR)", IEEE , 2014.
[9] Ajay Todimala, K. K. Ramakrishnan and Rakesh K. Sinha, "Cross-layer Reconfiguration for Surviving Multiple-link Failures in Backbone Networks", IEEE, 2009.
[10] Rozita Yunos, Siti Arpah Ahmad, Noorhayati Mohamed Noor, Raihana Md Saidi, Zarina Zaino, "Analysis of Routing Protocols of VoIP VPN over MPLS Network", 2013 IEEE Conference on Systems, Process & Control (ICSPC2013), 13 - 15 December 2013
[11] Cezary ˙ Zukowski, Artur Tomaszewski, Michał Pióro, David Hock, Matthias Hartmann and Michael Menth, "Compact node-link formulations for the optimal single path MPLS Fast Reroute layout", Advances In Electronics And Telecommunications, VOL. 2, NO. 3, SEPTEMBER 2011
[12] Maria Hadjiona, Chryssis Georgiou and Vasos Vassiliou, "A Hybrid Fault-Tolerant Algorithm for MPLS Networks", Department of Computer Science University of Cyprus.
[13] Suksant Sae Lor, Redouane Ali, Raul Landa, and Miguel Rio, "Recursive Loop- Free Alternates for Full Protection Against Transient Link Failures, IEEE, 2010.