Improved Algorithm For Creating An Optimized Network Diagram

Dr. Amer Tahseen Abu Jassar,
Computer Science Department, Information Technology Faculty, Ajloun National University, Jordan. amer_abu_jassar@hotmail.com
Prof. Safwan Al Salaimeh,
Software Engineering Department, Information Technology Faculty, Aqaba University of Technology, Jordan, Safwan670@yahoo.com
Dr. Mohammad Salim Al Hababsah,
Ministry of Education / Jordan, moh.uk88@gmail.com

Article Info: Volume 83
Page Number: 108- 115
Publication Issue:
January –April 2021

Article History
Article Received: *04 October 2020*
Revised: *14 November 2020*
Accepted: *22 December 2020*
Publication: *23 January 2021*

Abstract:
In this paper, a developed algorithm is presented to create an optimal network diagram. Some of the models that are used to graph the network were studied, in addition to studying the current SDN algorithm, with a general description of the most important problems encountered in the network diagram, the most important rules necessary to build it, and the most important algorithms necessary to create graphs with the choice of the optimal path.

Keywords: networks, graph, optimal path, algorithm, diagram, description.

Introduction:

Recently, we see that a person cannot live without using the Internet. Since using the Internet, he can make an appointment with the engineer, farmer, teacher, and others, in addition to payment sites, search for the information he needs, and so on.

At the present time, we see that the telecom companies that provide telecom services have become an integral part of the life of any person, organization, or even an entire country.

We see that there is a significant increase in the number of Internet users every year [1], which requires an increase in the resources of the companies providing this service periodically in order to provide communication service around the world.

It is known that every network operator owns a different set of equipment, the number of which increases every year. To ensure the wide spread of the network, as it needs to use new equipment located in one of the designated rooms equipped with an adequate power source. Which requires many costs: such as increased energy consumption, the need to hire qualified professionals, in addition to the capital operating costs. Also, this is in addition to the process of renewing and developing some devices that have become very old and do not meet the purpose and do not perform the necessary function, which leads to repeated "purchase - design - integration - publish" cycles. This does not lead to an increase in the revenues of the network operator. Rather, it leads to the fact that the construction cost of the network is exceeding the revenues, as the life span of the equipment becomes shorter [2].

Research importance:

By studying the current path used to develop operator networks, we see that it no longer meets the goals for which it was set, as it requires the need to present new perspectives on the work of service providers and network operators. The importance of this research lies in finding a solution to this problem which is the virtual representation of network functions - NFV functions, associated with the concept of Software Defined Networking - Software Defined Network (SDN) [3]. This concept will save money on purchasing new equipment.

The object of research is the virtual functions of the software-defined network SDN.

Research topic: A topic is to define a series of default functions for the optimal location of nodes in a network diagram.

Research objective: The aim of this research is to develop a model for the optimal construction of a network graph for a series of default functions for the stable operation of a software-defined SDN.

Research Question:

To achieve the goal, the following issues were identified:

1- Analyzing the educational, scientific and methodological literature on the subject area, and identifying the advantages, advantages and disadvantages of software-defined networks;

2 - Investigate the current implementation of building a network diagram for a communication network;

3 - To present a mathematical model of a communication network graph;

4 - Development of an algorithm for building a communication network diagram;

5 - To reduce the shortcomings of the current implementation of building a telecommunications network diagram.

The current mechanism used to build network diagrams:

It is known that the architecture of NFV MANO (Network Function Virtualization, called NFV, Management and Coordination, called MANO) is being developed by a special group of European Telecommunications Standards Institute - Industry Specification Group (ISG) ETSI NFV. It defines the standard for managing and coordinating all NFV resources within cloud data centers. These resources include computing, network resources, in addition to storage systems, virtual machines, etc. [4].

The goal of MANO is to create the infrastructure to deploy and manage network functions in a flexible and seamless manner, in addition to simplifying the tension that can arise due to the rapid pace of the emergence of new virtual network functions on the market [5].

When studying and examining the properties of NFV, the relationships and data needed for other connections used in hypothetical correlation abstractions are identified. To simulate the process of connecting virtual circuits to virtual network functions, NFVs are used for connection points (CP), which are virtual and / or physical interfaces for virtual network functions (virtual network function, with VNF) and their associated features. And there are network elements there that serve the network (NS). NS is a set of networking features that define behavioral features and specifications. Thus, NS can be viewed from an architectural point of view as a schedule for the transfer of network functions (hereinafter referred to as NF), interconnected by way of supporting the network infrastructure [6].

The network connection topology describes how different VNFs communicate and how data flows over these connections, regardless of the location of the underlying physical network elements. The network forwarding table defines the sequence of VNFs which must be traversed by a set of packets that

meet certain criteria.
The network routing habit diagram, which is generated by the NS operator, contains parameters that define which packets should be routed through the schedule. A simple example of this is source address routing or a number of other different applications. Different forwarding schedules can be created in the same network topology based on different criteria.
A network graph is a mathematical object that contains a set of graph vertices (VNFs) and a set of edges, that is, connections between pairs of vertices.
At the moment, all configuration of the network elements interaction with the NS operator. It must create the VNFs, configure it, and create the necessary CPs in addition to the virtual links (virtual link, called VL), which describe the basic connection architecture, along with other necessary parameters [7].
From the above, we see that among the most important flaws of this algorithm:
1- Building a network diagram manually.
2- There are no alternative charts;
3- Not using the optimal path;
4- Network graph is time consuming due to large number of default functions.
Based on the aforementioned defects, we see that it is clear that the process of creating network diagrams needs to be automated and to a simplification process, and here lies the **scientific novelty of work**. In the next paragraph, we will introduce a new solution for creating a network diagram for a series of virtual functions, which includes the use of a software module that aims for automatic building.

Clarify the problem of building an optimal network diagram:
The task of generating network diagrams cannot be completely ignored by the NS operator, because situations arise in which the sequence of VNF communication is often important.

Description of the proposed algorithm for NS operator activity:
1- Create virtual network functions;
2- Configure virtual network functions by creating CPs;
3- Placement of pathways between VNFs, when necessary. If the interaction order of the virtual network functions is not important, the entire diagram will be generated automatically.

problem formulation:
To solve the traffic routing problem, it requires a limited set of nodes (VNFs) for the network graph, and the latency between nodes at present is known. This requires finding the best way to redirect traffic, with the least amount of time.

When solving the routing problem, the following limitations must be considered:

1- The traffic should not be through one head more than once;
2- Traffic must be through all peaks.

The most important terms used in this research
The indirect graph loaded G = (V, E, U) and is usually in three groups:
1- V = (v1, v2,…, vn) is a set of vertices, the number of vertices is assumed to be | V | = n;
2- E⊆V × V is the set of edges of the graph {vi, vj} = e (i, j), we will assume that | E | = m is the number of edges in the graph.
3- U: E → [0, + ∞) is the weight function on the edges, the weight of the edge is u (vi, vj) between the vertices vi and vj. For vertices not connected by an edge, put u (vi, vj) = ∞.
The loaded graph will be specified using edge length matrices.
If {vi, vj} ∈E, then the vertices vi, vj are called adjacent.
The edge e (i, j) is incident to the vertices vi

and vj if vi, vj∈V and e (i, j) ∈E.
The number of edges of the graph incident to vi is called the degree d (vi) of the vertex vi.
If m≪ n2 is true, then the graph is called sparse.
If any pair of vertices is connected by one edge, then such a graph is called complete.
A path in a graph is an alternating sequence of vertices and edges ∏ (io, jk) = ∏ (v (i0), v (jk)) = v (i0), e (i0, i1), v (i1),…, v (i (k-1)), e (i (k-1), ik), v (ik), each edge is incident to two vertices - immediately preceding it and immediately following it.
The sum of the weights of the edges entering the path is the path length.
The shortest path ∏ (vi, vj) between the vertices vi and vj is the path that has the minimum length between the vertices vi and vj.
The length of the shortest path between vi and vj (m (vi, vj) = 0, i = 1,…, n) is called the distance m (vi, vj) between the vertices vi and vj.
If there is a path between any two vertices of the graph, that is, m (vi, vj) <∞ for all i, j, then the graph is called connected.
Graphs without loops and multiple edges are called simple graphs.
Several shortest paths of equal length can exist between the initial and final vertices of the graph. This circumstance in this work is not essential, therefore, for all references to the shortest path, we mean any shortest path.

Let ∏ (vi, vj) be the shortest path in the graph G. Then any of its subpaths is also the shortest path.

The classical problem:
The classical shortest path problem (SP) [8] in its most general form is formulated as “find paths between elements of two sets of vertices V1 and V2 of a graph so that the lengths of the found paths are minimal in a given graph between the corresponding vertices”.

The shortest path problem is one of the most important in algorithmic graph theory.
There are two variants of MDSP tasks:
1- non-stationary problem - the weight of the edge e (i, j) is equal to the travel time from vi to vj, which is given by a predefined function of time, i.e. the weight of an edge e (i, j) in a certain path depends on the start time of traversing the path and on the time spent on traversing the edges preceding e (i, j);
2- variant of the problem, in which the graph changes at certain intervals, being static between these intervals. The second variant of the problem can have two formulations: a partially dynamic problem (you can either only remove edges, or only add), a fully dynamic task (both deletion and insertion of edges are allowed). In both formulations, the operation of changing the weight of the edge set Weight (i, j, w) is allowed.
The non-stationary DSP problem using the techniques for the SP problem is solvable in polynomial time on graphs that have the FIFO property [2], and NP-hard on graphs that do not have this property [2]. DSP tasks with periodic graph changes can be solved by reducing them to the corresponding tasks and calculating the shortest paths of the graph "from scratch" whenever necessary, or using the shortest path reoptimization technique [8].

The Solution of the classical problem
Solution of the classical problem SP on the spanner H obtained for a given graph G. It is said that the graph H = (V, Es): Es⊆E (α, δ) is the spanner of the graph G = (V, E) if ∀vi , vj∈V mh (vi, vj) ≤ α · mg (vi, vj) + δ, where mh (vi, vj), mg (vi, vj) are the shortest distances between vertices vi, vj in graphs H and G, respectively .
In the simplest case, when the SSSP problem is considered, a breadth-first search (BFS) is considered the classical solution search algorithm [2]. BFS can solve the problem on both undirected and directed graphs. The

principle of operation of the algorithm consists in traversing the vertices of the graph in the order of detection and performing the relaxation process for arcs between the current vertex and each new detected one. The process of relaxation of arcs is the refinement (reduction), if possible, of the estimate of the path length from the original vertex to the newly discovered one by drawing the path through the current vertex. BFS is used as the basis of algorithms for many other graph problems and has a time complexity estimate of O (m + n), being the fastest for SSSP on unloaded graphs.
There are also a number of important algorithms for non-classical shortest path problems. Algorithm A * (A star) [6] is an informed search algorithm used to find the shortest paths from the original vertex s if additional information is known for the graph. The traversal of vertices in the algorithm is performed by the BFS method, and the order of traversal of vertices is determined by the heuristic function f (x) = g (x) + h (x), where g (x) is the known distance from s to x, and h (x) is the heuristic an estimate of the distance from s to x, which should be an admissible heuristic [9].
When solving the SPSP problem, bidirectional versions of the algorithms used for pathfinding (A *, BFS, Dijkstra) can be applied. In bidirectional search [10], a simultaneous search is performed from the source vertex s and the destination vertex t. The asymptotic complexity of bidirectional search is defined through the branching factor b [7] as O (b (d / 2)) + O (b (d / 2)) relative to the complexity O (bd) of the unidirectional version of the search.
If the graph is directed acyclic (DAG), the solution to the SSSP problem on it can be obtained in O (m + n) time [10]. First, the vertices of the graph are sorted in topological order, after which the edges outgoing from each vertex are weakened.
Algorithms of this kind are based on the concept of a hierarchy of vertices - the shortest paths between highly distant graph vertices more often pass through highway nodes than others. Accordingly, such algorithms combine a method for constructing a hierarchy and an algorithm used to find paths on the resulting hierarchy.
Using the BFS Breadth First Search algorithm, determine the arc length:

m = min si, i ϵ [1; N], (1)

where si - is a value calculated by the formula:

si = ∑ (j = 1) (N-1) qi · ti, (2)

where qij = {0,1} is a Boolean function, the value of which is set depending on the presence of the response of the vertex v (j + 1) to the signal from vj for the th graph,
tij - time of execution of the test command by the vertex vj.
Let us define an algorithm for finding the optimal network graph of a sequence of virtual functions:
1- Recursively, starting from the vertex "Primary Inputs" (from the inputs of the circuit to the outputs), for all adjacent vertices, we will make a natural connection of the corresponding relations and calculate the value of si for the final vertex.
2- Recursively, starting from the “Primary Outputs” vertex (from the circuit output to the inputs), for all adjacent vertices, we will make a natural connection of the corresponding relations, and project the result to the initial vertex.
To implement the construction of graphs, we will consider 2 ways of representing them: using a distance matrix and using an adjacency list.
The distance matrix (Table 1) is a convenient way to represent dense graphs in which the number of edges is approximately equal to the number of vertices in a square. In this view, a matrix of size M by M is filled, where M is the number of vertices, placing the value

of the distance between the i-th and j-th vertices in the cell A [i] [j]:

	VNF#1	VNF#2	VNF#3
VNF#1	0	20	27
VNF#1	20	0	13
VNF#1	27	15	0

Table 1 - Distance matrix

Distance matrix is a graph representation method suitable for directed and undirected graphs. Undirected graphs are represented as a symmetric matrix A, since if there is an edge between i and j, then it is both an edge from i to j and an edge from j to i. Thanks to this property, it is possible to reduce the memory usage by almost half by storing the elements only in the upper part of the matrix, above the main diagonal [12- 15].

With this representation, you can quickly check if there is an edge between the vertices v and u by simply looking at cell A [v] [u] [16 - 19].

Adjacency lists are another way to represent graphs. This method of representation is more suitable for sparse graphs, that is, graphs whose number of edges is much less than the number of vertices in a square. This view uses a two-dimensional array A containing M lists. Each list A [i] contains all the vertices u, so there is an edge between v and u. The memory required for the presentation is M + N, where M is the number of vertices and N is the number of edges in the graph.

Comparing the ways of representing graphs, we conclude that adjacency lists are more suitable for the described subject area, since in the graphs to be built, the number of edges is much less than the number of vertices in a square, therefore, there is no need to use adjacency tables, which take much more memory.

The results:

The result of solving the problem posed above on the graph G is the distance matrix Mr. The elements of the distance matrix M for the vertices vi, vj from the graph G are changed by the formula m (vi, vj) = mr (vi, vj) and the matrix of sequences P is changed by the formulas:

p (vi, vj) = {(pr (vi, vj) if m (vi, vj)> mr (vi, vj) and p (vi, pr (vi, vj)) = ∞

p (vi, pr (vi, vj)) if m (vi, vj)> mr (vi, vj) and p (vi, pr (vi, vj)) ≠ ∞) (3)

where vi is the first vertex with coordinates (Xk, Yl), k = 1 ... N, l = 1 ... N,

vj - the second vertex with coordinates (Xq, Yp), q = 1… N, p = 1… N,

pr (vi, vj) are elements of the matrix of followers P of the graph G. The paths found between the vertices of the graph G are guaranteed to be the shortest.

The values of the matrices M and P are specified as follows:

m (vi, vj) = u (vi, vj) (4)

p (vi, vj) = {(vi if u (vi, vj) ≠ ∞

∅ if u (vi, vj) = ∞) (5)

Conclusion:

From the above, the requirements for the program unit can be formulated as follows:

- The program module should build N graphs of minimum length;
- The program unit should create a perfect network diagram;
- The program unit must be equipped with the unit test.

In this paper, an analysis of the current model used to create network diagrams was made, and it was found that it suffers from several problems. It has also been suggested to solve these problems in the automation process. The algorithms are considered to create the optimal paths in the graph, and a description of a mathematical model that helps to represent the network graph is also presented, and the requirements for the program unit have also been defined.

References:

1. Farrell J. Java Programming / Course Technology 2015 – 1026 p.
2. Hof P. van't, Kamiński M., Paulusma D., Szeider S., Thilikos D.M. On Graph Contractions and Induced Minors // Discrete Applied Mathematics. 2012. Vol. 160. P.
3. Oaks S. Java Performance: The Definitive Guide – O'Reilly, 2014.
4. Pilgrim, P. Digital Java EE 7 Web Application Development / P. Pilgrim — Packt Publishing, 2015. – 486 c.
5. Sam Newman, Building Microservices - O'Reilly Media, 2015. – 280 pages.
6. Safwan Al Salaimeh, Amer Abu Zaher // Developing Enterprise system with CORBA and JAVA integrated Technologies, Journal annals computer Science Series, Vol 9. No1, 2011. Romania.
7. Jang, H. Park, and V. K. Prasanna, "A fast Algorithm for computing a histogram on reconfigurable mesh", IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol. 17, No.2. 1995.
8. Khaldoun Al besoul , Safwan Al Salaimeh, The Structure of logistics organizational technological system, Journal information society, Vol.4, Num. 7, June, 2007,
9. Faten Hamad & Abdelsalam Alawamrah, Measuring the Performance of Parallel Information Processing in Solving Linear Equation Using Multiprocessor Supercomputer, Modern Applied Science, Vol. 12, Issue, 3, 2018.
10. Mohammad, Q., & Khattab, H, New Routing Algorithm for Hex-Cell Network, International Journal of Future Generation Communication and Networking, 8(2), 295-306. 2015.
11. Rajalakshmi, K, Proposed a Parallel Algorithm for Solving Large System of Simultaneous Linear Equations, 2009.
12. Khaled Batiha, Safwan Al Salaimeh, (2016)// Development sustainable algorithm optimal resource allocation in information logistics systems. International journal of computer applications (IJCA), March 2016 edition. USA.
13. Pasetto, D., & Akhriev, A, A comparative study of parallel sort algorithms. In Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion (pp. 203-204). ACM, 2011.
14. Safwan Al Salaimeh, // A new model for information logistics system Architecture, Journal of Theoretical and Applied Information Technology, June, Vol.28. No.1, 2011, Pakistan.
15. Scholl, S., Stumm, C., & Wehn, Hardware implementations of Gaussian elimination over GF (2) for channel decoding algorithms. In AFRICON, , September, 2013.
16. Guernsey, GY, Process Dynamics: Modeling, Analysis, and Simulation, Prentice Hall, United Kingdom, 2003. [4] Otto Bretscher, Linear Algebra with Applications, Prentice Hall, UK, 2008.

17. Silvia T. Akuna, Software processing modeling, Springer, USA, 2005.
18. Hossein Bidgoli, Modern Information Systems for Managers, Academic Press, 2009.
19. Howard Bandy, Modeling Trading System Performance, Blue Owl Press, 2011.