International Journal of Security and Its Applications
Vol. 10, No. 1 (2016), pp.155-166
http://dx.doi.org/10.14257/ijsia.2016.10.1.15

New Secure Load Sharing Algorithm in Network Layer

Mouhcine Chliah[1], Ghizlane Orhanou[2] and Said El Hajji[3]

Laboratory of Mathematics, Computing and Applications, Faculty of Sciences, University of Mohammed-V, Rabat, Morocco.
[1]mouhcine.chliah@gmail.com, [2]orhanou@fsr.ac.ma, [3]elhajji@fsr.ac.ma

Abstract

The objective of the present paper is to bring a protection of information against new threats and attacks, by introducing security issues in the elements of the network, operating at low level, especially at layer 3. We propose to arm them with a new method of processing, while dispatching messages based on a Secure Load Sharing algorithm (SLS algorithm), that can bring a big help in stopping attacks based on sniffing, like MitM. We provide implementation tests that show the efficiency of this new concept to bring more traffic security, without any negative impact on routers operations.

Keywords: *Routing, Network security, Graph theory, Load sharing, SLS*

1. Introduction

The emergence of cloud computing, mobile communications, and big data have exponentially expanded the dynamic nature of the global economy. At the same time, the number and the nature of threats has changed and evolved as well. New points of attack and exploitation are created and need to be identified and faced continually.

Indeed, due to recent technological advancements, the digital communications industry is going through dramatic changes. A new era of Internet and mobile facilities based on networking technology is emerging. Individuals, corporations, and government organizations are freely interconnected, creating strong demand for seamless network services with global availability, security, and mobility.

In this paper, we designed a new method of routing on top of existing routing protocols, to arm existing network elements, such as routers, with a kind of security; hence even with some new forms of attacks based on sniffing, or introduction of new un-trusted device inside the network, the traffic inside the network and especially the existing devices, will be protected from the threat coming from the new device.

More precisely, in this paper, we enhanced the existing load-balancing process [1] done by routers, by adding some rules during the selection of paths being used. Doing so, we'll ensure that routers will help in providing information confidentiality [2] while doing their day to day job, which is routing.

Indeed, routing is the process of selecting best paths in a network, and there are some routers that offer load-balancing, but without taking into consideration security aspects. With the proposed secured routing, we introduce a secure load sharing when routing and dispatching packets, by choosing the most secure paths to forward segments of the same packet via different paths. To do this, we have followed two major steps that will be exposed in the different sections of the present paper:

First, Using Graph Theory [3] in section II, we developed an algorithm which calculates all possible paths from Source to Destination based on Adjacency Matrix.

ISSN: 1738-9976 IJSIA

Then, choosing among all possible paths given by pathfinder algorithm, all combinations that meet a number of criteria, needed to introduce security; these criteria will be exposed in Section 3.

Below, in the next session, we'll give a brief description of Graph theory, and its use in computer network field. Then, we'll present the first step of our proposition to introduce the security issue in routers operation.

2. Graph Theory in Computer Network

A Graph is a pair G = (V, E) of sets such that $E \subseteq [V]^2$; Thus, the elements of E are 2-element subsets of V. The elements of V are the vertices (or nodes, or points) of the graph G, the elements of E are its edges (or lines). The usual way to picture a graph is by drawing a dot for each vertex and joining two of these dots by a line if the corresponding two vertices form an edge.

According to the problem to be dealing with, the notion of a graph can vary: A graph may have multiple edges, *i.e*. between two points there may be more than one edge; loops *i.e*. edges connecting a vertex to it, or directed *i.e*. each edge is not a 2-element set but an ordered pair. A path in a graph is a sequence of vertices $v_1 \ldots v_n$, such that each vertex connected to the following one by an edge.

In this paper, we'll deal with symmetric graph only, in the sense that if vertex a is related to vertex b, then vertex b is related to vertex a via same edge, and also no loops.

2.1. Presentation of the Pathfinder Algorithm

Each Graph could be represented as a simple Matrix. The adjacency matrix of a graph with n vertices is an n x n matrix $A = (a_{i,j})$ in which, the entry $a_{i,j} \neq 0$ if there is an edge from vertex i to vertex j, and is 0 if there is no edge from vertex i to vertex j.

$$A = (a_{ij})_{i,j=1}^{n} \text{ where } a_{ij} = \begin{cases} \neq 0 & \{i,j\} \in E \\ = 0 & \{i,j\} \notin E \end{cases}$$

$$\begin{pmatrix} 0 & a_{1,2} & \cdots & a_{1,j} \\ a_{2,1} & 0 & \cdots & a_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i,1} & a_{i,2} & \cdots & 0 \end{pmatrix}$$

We're modeling computer network to Graph, then to matrix, in order to make it easier for a router with a dedicated algorithm, to go through it and find all possible paths from one node to another in a fastest way.

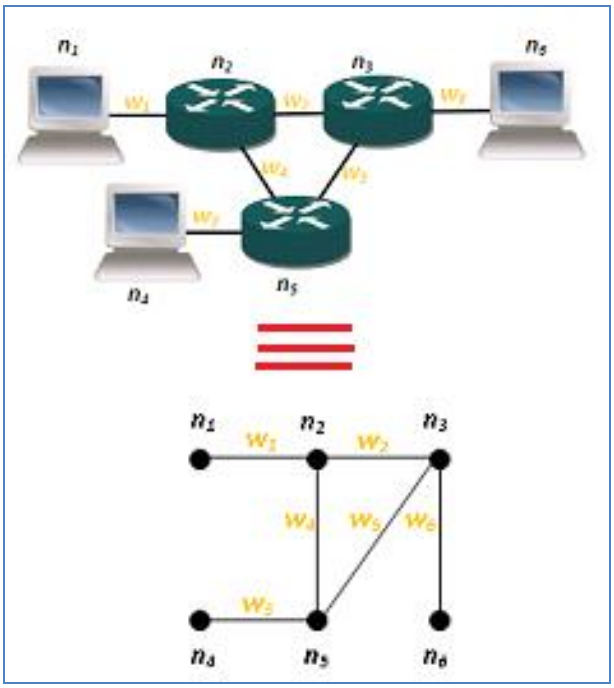

Figure 1. Modeling a Computer Network as a Graph

Using a graph, it's much easier to find all possible paths, from one point to another. From the last example, to go from n1 to n6 we have:

n1 ➔ n2 ➔ n3 ➔ n6

And

n1 ➔ n2 ➔ n5 ➔ n3 ➔ n6

The algorithm below is built maintly on one Recursive function, to browse end to end the adjacency matrix in order to find all possible paths from source to destination, then display them:

```
Input:
G=(S,E)
N := Number of vertices.
Integer path[n] := Will be used to store paths during exploration.
Boolean usedNodes[n] := Will be used to lock vertices already used.
Integer source := Source
Integer target := Destination
Integer tmp := tmp variable to be used to store total distance for given path.
pathFinder(source,0)
Funct pathFinder (Integer position, Integer depth)
path[depth]:=position
IF (position==target) THEN  // We reached the target
tmp := 0 // initiate total distance.
FOR(Integer i := 0 ; i <= depth ; i++)                    // display the solution
DISPLAY path[i]
IF (i==0) continue;
tmp := adjaMatrix[path[i-1]][path[i]] + tmp;
END FOR
DISPLAY <<" = "<<tmp<< endl;
return;
```

```
    END IF
    usedNodes[position] := true // We mark the vertices being used.
    FOR (Integer i: = 0; i<n; i++) {                    // We continue checking
remaining vertices.
    IF (adjaMatrix[position][i]==0 || usedNodes[i]) THEN continue END IF
    pathFinder (i,depth+1)
    END FOR
    usedNodes[position] := False; // unlock the vertices.
    END Function
```

Figure 2. Pathfinder Algorithm

2.2. Example of Pathfinder Algorithm Implementation

In below figure, we give an example of a network for which we applied the proposed algorithm.

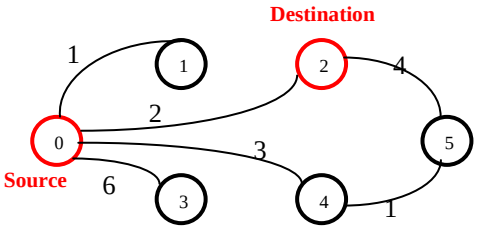

Figure 3. First Example of Network –Network 1-

The network is composed of 6 nodes related by edges with different weights.

Since each graph could be represented as a simple Matrix, we represent below the Adjacent Matrix for the studied network:

$$AdjaMatrix[6][6] = \begin{pmatrix} 0&1&2&6&3&0 \\ 1&0&0&0&0&0 \\ 2&0&0&0&0&4 \\ 6&0&0&0&0&0 \\ 3&0&0&0&0&1 \\ 0&0&4&0&1&0 \end{pmatrix}$$

Then, we apply the pathFinder algorithm to find all possible paths from a given source to a given destination.

As an example we can take:

- Source = n0.
- Destination = n2.

For this small network, two possible solutions were found, and also the best ones “Figure 4,” in order to go from Node 0 (n_0) to Node 2 (n_2) with two different total weights, 2, and 8.

```
From 0 to 2 possible routes are :
0 2   = 2
0 4 5 2   = 8
```

Figure 4. Algorithm Output for Network 1

We can read it as follow:
0 2 = 2 ➔Go directly from node0 to node2 with total weight 2.
0 4 5 2 = 8 ➔Go from node0 to node2 via node4 and node5 with a total weight of 8.

This algorithm has no negative impact on the routers performance.
Indeed, using the above algorithm, we have performed simulation for a network containing from 2 to 100 nodes. According to the achieved results, we can consider, as illustrated below, that no negative impact is foreseen on Routers if the number of hops does not exceed 12.

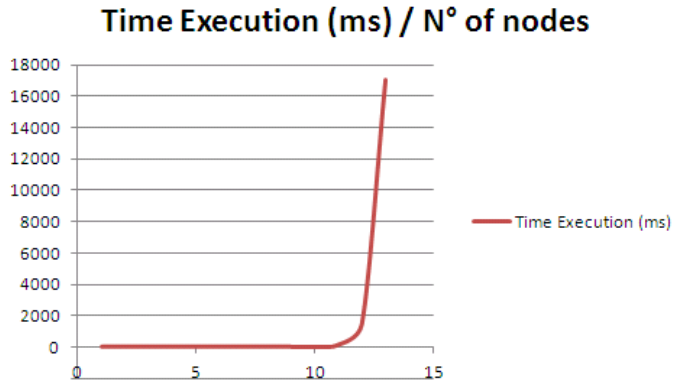

Figure 5. Impact of Number of Nodes (Hops) on Performance

This limit can be overcome if we divide logically a network to small cells containing at most 12 hops (like the Area notion used by OSPF protocol [4][6]).

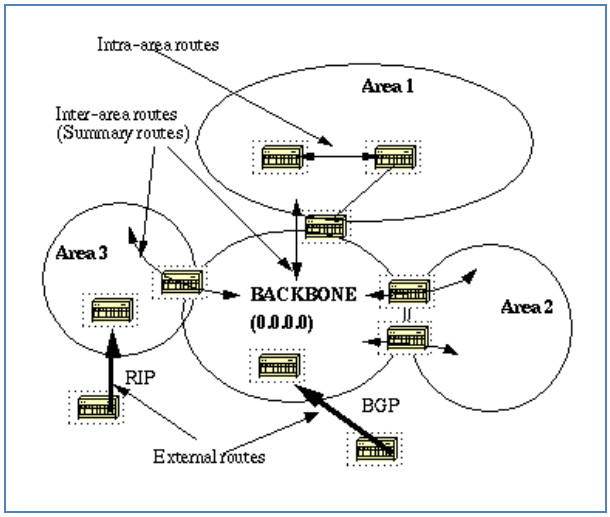

Figure 6. OSPF Area [5]

In the next section, we'll try to combine paths given by pathfinder algorithm that meet some criteria, in order to enhance the security while doing load sharing between different paths.

3. Secure Load Sharing

In this section, we'll introduce a new algorithm able to combine paths in order to satisfy some conditions that will be presented below. These conditions are mainly used to add security while dispatching messages. So first, we'll present the whole algorithm

called Secure Load Sharing (SLS Algorithm). Then, the second part will be dedicated to its implementation.

3.1. Paths Selection Algorithm

In the previous sections, we got all possible paths from point A to point B.
Below, we define some parameters needed in the path selection algorithm:

- P_n: Path n.
- EP_n: Set of intermediate nodes used for Pn .
- w_n: Total weight for P_n.
- m_n: Load to be considered on P_n .

Now, using initial algorithm's output, we'll define criteria to be used during path selection in a way to include security.

Condition 1:

Select paths (P_i & P_j) with same weight ($w_i = w_j$) if possible. With this condition, the router will not bother about the load to be used on each leg, and also with different intermediate nodes $EPi \cap EPj = \emptyset$.

The last condition of different intermediate nodes is the most important, because if one node is infected, by sniffing, or via MiTM, the attacker will be able to get only a part of packets that flows via infected node, hence not possible for him to get the end to end message.

Condition 2:

If not possible to apply choice recommended by condition 1, then we'll check for paths with different intermediate nodes $EPi \cap EPj = \emptyset$ even if $wi \neq wj$, in this case the load will be different on each path.

Below, we give an example to illustrate this issue.

By considering a second example representing a more complicated network than the first example presented in section 2:

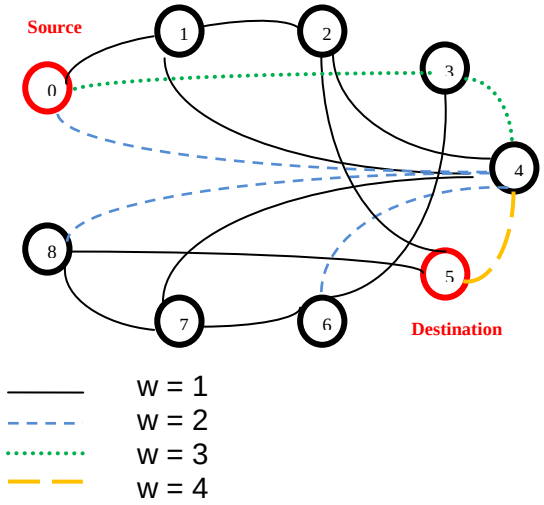

Figure 7. Example of Complicated Network, -Network 2-

The adjacency matrix will be presented as follow:

$$AdjaMatrix[9][9] = \begin{pmatrix} 0&1&0&3&2&0&0&0&0 \\ 1&0&1&0&1&0&0&0&0 \\ 0&1&0&0&1&1&0&0&0 \\ 3&0&0&0&3&0&1&0&0 \\ 2&1&1&3&0&4&2&1&2 \\ 0&0&1&0&4&0&0&0&1 \\ 0&0&0&1&2&0&0&1&0 \\ 0&0&0&0&1&0&1&0&1 \\ 0&0&0&0&2&1&0&1&0 \end{pmatrix}$$

Then we will have more possibilities (Example: Source n_0 and Destination n_5)

```
From 0 to 5 possible routes are :
0 1 2 4 3 6 7 8 5   = 10
0 1 2 4 5   = 7
0 1 2 4 6 7 8 5   = 8
0 1 2 4 7 8 5   = 6
0 1 2 4 8 5   = 6
0 1 2 5   = 3
0 1 4 2 5   = 4
0 1 4 3 6 7 8 5   = 9
0 1 4 5   = 6
0 1 4 6 7 8 5   = 7
0 1 4 7 8 5   = 5
0 1 4 8 5   = 5
0 3 4 1 2 5   = 9
0 3 4 2 5   = 8
0 3 4 5   = 10
0 3 4 6 7 8 5   = 11
0 3 4 7 8 5   = 9
0 3 4 8 5   = 9
0 3 6 4 1 2 5   = 9
0 3 6 4 2 5   = 8
0 3 6 4 5   = 10
0 3 6 4 7 8 5   = 9
0 3 6 4 8 5   = 9
0 3 6 7 4 1 2 5   = 9
0 3 6 7 4 2 5   = 8
0 3 6 7 4 5   = 10
0 3 6 7 4 8 5   = 9
0 3 6 7 8 4 1 2 5   = 11
0 3 6 7 8 4 2 5   = 10
0 3 6 7 8 4 5   = 12
0 3 6 7 8 5   = 7
0 4 1 2 5   = 5
0 4 2 5   = 4
0 4 3 6 7 8 5   = 9
0 4 5   = 6
0 4 6 7 8 5   = 7
0 4 7 8 5   = 5
0 4 8 5   = 5
```

Figure 8. Pathfinder Output for the Network 2

We got 38 possibilities (P_n with $n \in [1-33]$) that can be used to send messages from 0 to 5, but not all of them could be used, as we want to introduce security while dispatching messages.

If we apply the conditions exposed above to the example, one of possible solutions could be:

- n0 ➔ n3 ➔ n6 ➔ n7 ➔ n8 ➔ n5 = 7

&

- n0 ➔ n1 ➔ n2 ➔ n4 ➔ n5= 7

Here, the load will be 50% on each path, and since the intermediate nodes are different between the two paths, so we'll apply the first condition.

Depending on network architecture and available links (100 Mb, 1Gb, 10Gb …), It's not all the time possible to apply the first condition. Hence, with the second condition, *i.e.* choosing paths even if total weigh is different, the load will not be the same. For example, if on the first path the total weight is 2, and on the second path the total weight is 4, we'll

have one path with 100% more traffic on it, *i.e.* we'll have to send two packets on first path for one packet on second path.

- n0 ➔ n1 ➔ n2 ➔ n5 = 3

&

- n0 ➔ n4 ➔ n7 ➔ n8 ➔ n5 = 5

With:

m_1= 5 (62.5% load)
w_1 = 3
m_2= 3 (37.5% load)
w_2 = 5

We have presented above manually how these conditions (condition 1 and condition 2), could be applied. But, in order to automate this process, we have developed an SLS algorithm (Secure Load Sharing algorithm) to implement this new method.

The algorithm takes as input the output of the pathfinder algorithm (presented in section 2 of the present paper), to display all possible paths starting with those that meet first condition then second condition.

Then, the router will choose one or another solution depending on fastest paths. If the first condition is not applied due to unavailability of one path, the second solution will be chosen.

The SLS algorithm used is presented below:

```
INPUT:All possible paths from pathFinder function (Matrix called Tab)
Integer m,n := size of table given by pathfinder.
Integer  imin,vmin,nbrSol :=respectively  min index , min value and number of
solutions.
#Solutions that meet first condition
Display "Solutions with paths having same weight with no intersections"
  FOR (Integer i:=0;i<n-1;i++) THEN
    FOR (Integer j:=i+1;j<n;j++) THEN
      IF((tab[i][m-1]==tab[j][m-1]) AND noIntersect(tab[i],tab[j])) THEN
        DisplaySolutions (tab[i],tab[j]);
        ELSE
         Continue; END IF
END FOR; END FOR
#Solution that meet second condition
Display <<"Solutions with paths having diff weight with no intersections"
  FOR(Integer I := 0; i<n-1 ;i++) THEN
    FOR(Integer j := i+1 ; j<n ; j++) THEN
      IF ((noIntersect(tab[i],tab[j])) AND ((tab[i][m-1]!=tab[j][m-1]))) THEN
        DisplaySolutions (tab[i],tab[j]);
        ELSE
         Continue; END IF
END FOR;END FOR
```

Figure 9. SLS Algorithm

The next part of this section will be dedicated to present an implementation test of the SLS algorithm using the adjacency matrix defined in the beginning of this section.

3.1. SLS Algorithm Implementation

In this implementation, we are using a network presented by the adjacency matrix below (the same used to apply manually the SLS algorithm):

$$AdjaMatrix[9][9] = \begin{pmatrix} 0&1&0&3&2&0&0&0&0 \\ 1&0&1&0&1&0&0&0&0 \\ 0&1&0&0&1&1&0&0&0 \\ 3&0&0&0&3&0&1&0&0 \\ 2&1&1&3&0&4&2&1&2 \\ 0&0&1&0&4&0&0&0&1 \\ 0&0&0&1&2&0&0&1&0 \\ 0&0&0&0&1&0&1&0&1 \\ 0&0&0&0&2&1&0&1&0 \end{pmatrix}$$

Before applying the SLS algorithm, we apply the Pathfinder algorithm to the adjacency matrix "adjaMatrix". The result will be the input of the SLS algorithm.

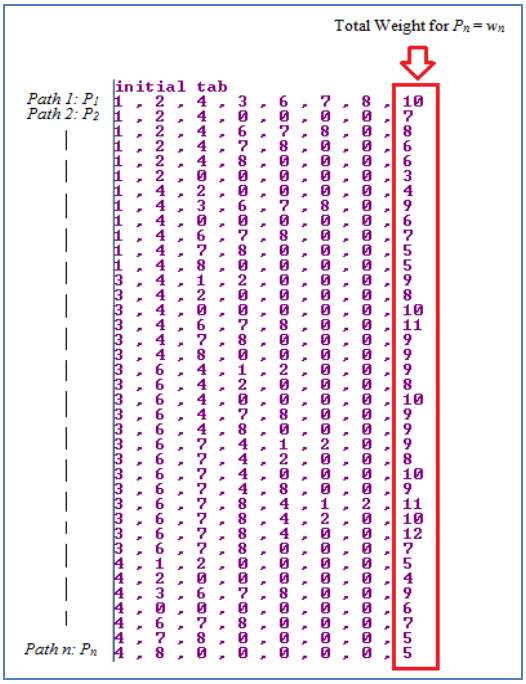

Figure 10. Input to SLS algorithm

Each line represents a path and each column will contain nodes to be used one by one to go from source (n_0) to destination (n_5), except the last column that represents the total weight of each path.

After that, the output of the PathFinder algorithm will sorted based on the total weight of each path.

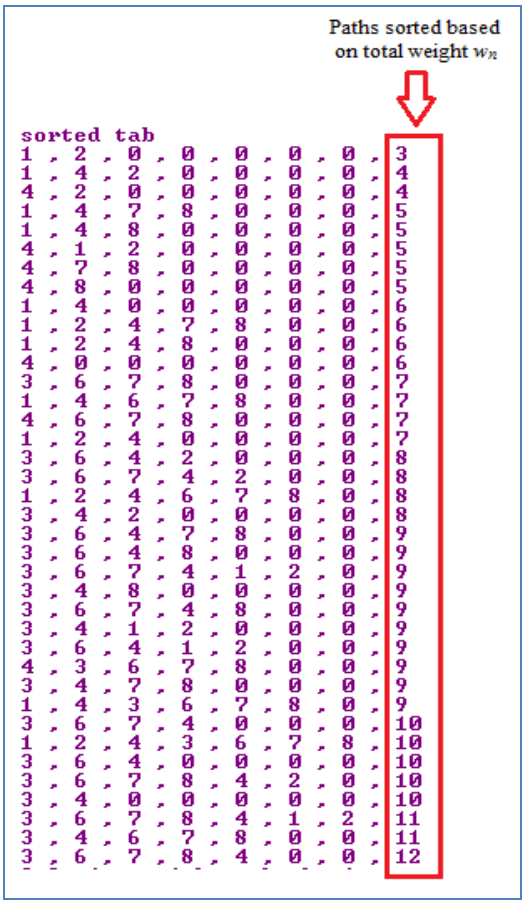

Figure 11. Pathfinder Output Sorted

Then, we apply the SLS algorithm to check line per line (path per path), applying the two conditions (condition 1 then condition 2).

Indeed, now that all inputs are available in proper way, we'll check if there is any intersection between all the paths that have same total weight. If there is no intersection, the algorithm print the solutions (application of first condition), starting with minimum total weight.

```
Solutions with paths having same weight with no intersections ...
---------------------------------
Solution 1
Path 1 =3 6 7 8 0 0 0 7
Path 2 =1 2 4 0 0 0 0 7
```

Figure 12. SLS Output in Respect of Condition 1

1st case: Condition 1 is applied:

If we consider Solution 1 above (the only solution available with first condition), we can use the set of the two following paths:

- Going via $node_3$, $node_6$, $node_7$ and $node_8$
- Or using $node_1$, $node_2$ and $node_4$

We remark that both paths have the same total weight which is "7". Hence, the load balancing is 50% on each path, and the most important thing, is that both paths are not using the same intermediate nodes, which limit the risk of compromising the confidentiality of the sent message; If one sniffer is located on node3, for example, the attacker will be able to get information transiting via Path1 (3-6-7-8) only, and the second

part of information going through Path2 (1-2-4) will not be possible for him to get it, and this is the major difference with normal routing and also with the per-packet load balancing already proposed by Cisco Router [5], using standard routing protocol such as RIP, RIPv2, EIGRP, OSPF, *etc*.

For the Cisco router case for example, Per-packet-load-balancing [1] is limited to the fact that router will send one packet for given destination over the first path, then the second packet for the same destination over the second path, and so on. Per-packet load balancing guarantees equal load across all links

2nd case: Condition 2 is applied:

If not possible to have two paths with the same weight, because, for example in "Figure 7", the link between $node_7$ and $node_8$ is down; other sets of paths will be displayed with different paths weights but still, using different intermediate nodes (application of second condition) as shown below.

Figure 13. SLS Output in Respect of Condition 2

As an example, we can take solution 2: the set of the two following paths can be used:

- Going via $node_1$ and $node_2$.
- Or using $node_4$ and $node_7$.

The SLS algorithm sorts solutions based on total weight; hence solutions with smallest total weight will be display first and so on.

The main difference between this case where we use Condition 2, and the first solutions, where Condition 1 is applied, is that the total weight is different. That's why a new parameter, called load m_n, will be defined for each path, and should be taken into consideration by the router while dispatching messages.

So using this set of paths (for Solution 2), we'll have:

m_1= 5 (62.5% load be considered on P_1)
w_1 = 3
m_2= 3 (37.5% load be considered on P_2)
w_2 = 5

Thus, we have presented, through this section, how the generic load sharing method proposed by routers could be enhanced, in order to accommodate security, and we have proposed a mechanism based on the SLS algorithm that can help stopping attacks based on sniffing, without any negative impact on routers performance.

4. Conclusion

In this paper, we proposed a new module inside routers to take care of the Confidentiality of information while dispatching messages, without incurring excessive load to the router. Initially, using graph theory, we built an algorithm able to find all possible paths from point A to point B. Then, using the proposed Secure Load Sharing algorithm (SLS algorithm), we were able to enhance the router task and add a new security issue, by dispatching segments of same packets via several paths having different intermediate nodes. Future work will focuses on enhancing the pathfinder algorithm to be able to handle more nodes in one area.

References

[1] Cisco," How Does Load Balancing Work?" Document ID: 5212, August **(2005).**
[2] T. Sneha Vasant, G. Deipali V., "Comparative Study of CIA and Revised-CIA Algorithm," Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on , vol., no., pp.713,718, 7-9 April **(2014).**
[3] J.C. Schlage-Puchta, « Graph Theory », CCSD, November **(2013).**
[4] K.Shuaib K, F. Sallabi, & A. Brooks, "Multi-area OSPF network design considerations", Proceedings of the IASTED International Conference on Wireless and Optical Communications, vol. 3, p. 555-559, **(2003)**.
[5] Cisco, "OSPF Design Guide", Document ID: 7039, August **(2005).**
[6] J. Moy, "OSPF Version 2", RFC2328, April **(1998).**

