

New Forwarding Strategy for Metro Ethernet Networks Based on Hierarchical Addressing

Farhad Faghani*, Ghasem Mirjalily**, Reza Saadat**, Farmarz Hendessi***

*Department of Electrical Engineering, Najafabad Branch, Islamic Azad University, Isfahan, Iran.

** Faculty of Electrical and Computer Engineering, Computer and Communication Networks Research Group, Yazd University, Yazd, Iran.

*** Faculty of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran.

faghani@iaun.ac.ir, mirjalily@yazduni.ac.ir, rsaadat@yazduni.ac.ir, hendessi@cc.iut.ac.ir

Abstract— Current Ethernet technology relies on Spanning Tree Protocol (STP). As the name suggests, STP creates a spanning tree within a network of connected layer-2 Ethernet switches, and disables those links that are not part of the spanning tree, leaving a single active path between any two network nodes. Because of lack of any traffic engineering mechanism in STP, there is not any load balancing on links and switches. This causes a huge load on some links and switches especially close to the root node. In our previous works, we have suggested some forwarding strategies such as: BST and SSS to overcome this problem. In this paper, we propose a new forwarding strategy based on hierarchical addressing named Hierarchical Forwarding Strategy (HFS), to improve the performance of the previous forwarding strategies. Simulation results show the improvements of HFS in comparison to STP and SSS.

Keywords— Ethernet Networks; Hierarchical Addressing; Spanning Tree; Traffic Engineering.

I. INTRODUCTION

Ethernet is widely accepted technology used in many local area networks. The significant advancement of Ethernet technology is pushing Ethernet from the local area network environment to metropolitan and wide area network environments. The major challenge for deploying Ethernet in metro area networks is to guarantee the quality of service and traffic engineering requirements. One of the main tasks of traffic engineering is load balancing, the idea of which is uniform resource utilization by moving traffic from congested links to other parts of the network in a well-controlled way [1].

Current Ethernet technology relies on IEEE Spanning Tree Protocol (STP), which provides a loop-free connectivity across various network nodes. STP does this task by reducing the topology of a switched network to a tree topology where redundant links are pruned. This action ensures there is a unique path from any node to every other node. Disabled links are then kept in a standby mode of operation until a network failure occurs. One of the main disadvantages of STP is the lack of load balancing. For example, if two leaf nodes want to communicate with each other, their traffic should pass a long way through the root. As a result, there is usually a huge

traffic on the links and switches close to the root that results in bottlenecks. Traffic engineering of Ethernet using spanning tree is a widely researched topic and several enhancements have been proposed in the literature in order to solve this problem by mitigating congestion near the root [2]. Some well-known solutions are SmartBridge [3], AMSTP [4] and Viking [5].

In our previous works, we have considered the problem of traffic engineering in Ethernet networks and we have proposed some solutions [6-7]. In [6], we proposed a strategy to select the best spanning tree (BST) in a given network topology based on load balancing on links and switches instead of using shortest path selection. The basic problem of BST is its high computational complexity in large-scale networks.

In [7], we proposed a new simple forwarding strategy named Shortcut Switching Strategy (SSS) that uses some of the blocked links to forward some traffic. We analytically proved that SSS has a better load balancing on links and switches versus STP. Although SSS improves the performance of metro Ethernet networks; but, it is not a complete solution. One of the main reasons is the few amount of information about network topology, which each node uses to forward frame. This lack of information redounds to weak performance of many proposed algorithms.

By using hierarchical addressing, each node can gather some information about the frame forwarding ability of their neighbors. Using hierarchical addressing is common in some wireless networks, but never used in Ethernet networks. For example, ZigBee uses hierarchical addressing to facilitate its tree routing [8]. In this paper, we propose a new forwarding strategy based on hierarchical addressing to improve the load balancing capabilities of metro Ethernet networks. Our new proposed strategy is named Hierarchical Forwarding Strategy (HFS).

The rest of this paper is organized as follows: Section II describes IEEE STP and SSS briefly. The proposed algorithm is explained in Section III. In Section IV, some simulation results are presented and conclusions are drawn in the last section.

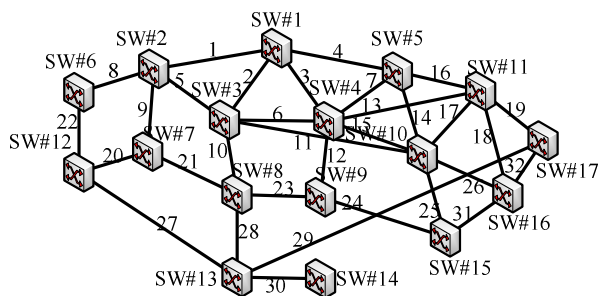


Figure 1. A typical network before running STP (Topology #1).

II. SPANNING TREE PROTOCOL AND SHORTCUT SWITCHING STRATEGY

In STP, the construction of minimum routing cost spanning tree is through exchanges of switch IDs and link costs carried by Bridge Protocol Data Units (BPDUs). BPDUs are control messages that are exchanged across the switches within a local area network.

BPDUs carry information about switches, ports, costs, and priorities. The STP uses the BPDU information to elect the Root switch and Root ports for the switched network. In this protocol, each of the ports of a switch assumes one of the five states, namely: Disabled, Blocked, Listening, Learning, and Forwarding. In addition to port states, STP assigns port roles or functions to each port namely: Root port, Designated port, and Blocked port [9].

Figure 1 shows a typical random network topology and Figure 2 shows this topology after running STP. Black Solid lines are active links and red dashed lines are blocked links. Note that in Figure 1, the links are identified using number labels (not weights or cost variables).

In STP, each switch maintains a forwarding list for each active port attached to the network. The list indicates the node addresses for which frames should be forwarded through that port. When a switch receives a frame on its port x , searches the forwarding table to determine if the destination address is listed for any port except port x . If the destination address is not found, forwards frame out all ports except the one from which it was received. If the destination address is in the forwarding table for some port y , transmits the frame through it onto the network.

A simple scheme for constructing the forwarding table is based on the use of the source address field in each data frame. Thus, a switch can update its forwarding table for that port on the basis of the source address field of each incoming frame [9].

As you can see from Figure 2, by using STP, many links and ports are blocked. So, using STP is not efficient and there is an implicit talent to improve it. In our previous work, we proposed a forwarding strategy named Shortcut Switching Strategy (SSS) to enhance the performance of STP [7]. Shortcut Switching Strategy is based on running STP to construct the spanning tree. In SSS, each node needs to learn the addresses of its blocked neighbor nodes.

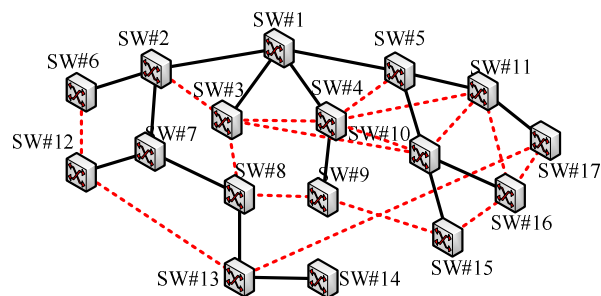


Figure 2. Typical network after running STP.

These neighbors are connected via blocked links; we call them Blocked Neighbor Nodes (BN-Nodes). Each node registers its BN-Nodes in a BN-Nodes list.

In summary, the address learning mechanism and forwarding table construction of SSS is the same as STP; but, in this new approach, each port learns the addresses of its blocked neighbor nodes (BN-Nodes) by using the source address field in each received BPDU.

In SSS, when a switch receives a data frame, it searches its BN-Nodes list and applies these forwarding rules:

- If the destination address is in its BN-Nodes list, it forwards the frame directly to that BN-Node via its blocked port.
- If the address is not available in its BN-Nodes list, it forwards the frame as usual by using its traditional STP forwarding table.

In next section, we will describe our new forwarding algorithm named Hierarchical Forwarding Strategy (HFS).

III. HIERARCHICAL FORWARDING STRATEGY

Our new proposed strategy consists of three major steps:

1. Running Standard Spanning Tree Protocol.
2. Re-Addressing the nodes by using hierarchical addresses.
3. Using new forwarding strategy.

A. Running Standard Spanning Tree Protocol

In this step, we must use the STP or its improved versions such as BST protocol to construct the traditional minimum-cost spanning tree by selecting the proper root and designated ports as usual.

B. Re-addressing the nodes by using hierarchical addressing

In conventional Ethernet, we use MAC address (Media Access Control address) that is a unique identifier assigned to network interfaces for communications on the physical network segment. MAC addresses are 48 bits often assigned by the manufacturer of a network interface and are stored in its hardware, read-only memory, or some other firmware mechanism. If assigned by the manufacturer, a MAC address usually encodes the manufacturer's registered identification number and may be referred to as the burned-in address.

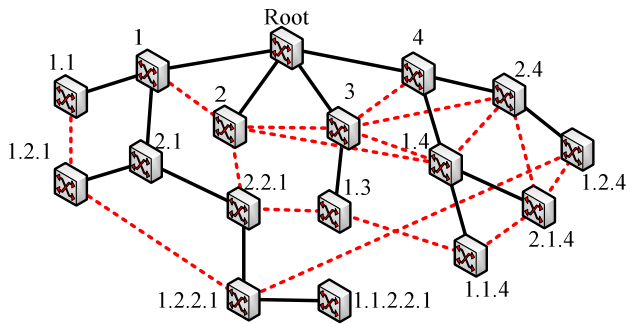


Figure 3. New node addresses after using Hierarchical addressing

It may also be known as an Ethernet hardware address, hardware address or physical address. A network node may have multiple interfaces and will then have one unique MAC address per interface.

In HFS, after running STP, all nodes need to be re-addressed in hierarchical manner. In other words, in HFS, we do not use the standard MAC addresses; instead, we use a new hierarchical addressing, in which, each node allocates the addresses of its children. This can be done by using BPDU control frames.

In order to achieve a feasible addressing scheme, address allocation will be based on the following assumptions:

- All addresses are 48bits.
- Maximum depth of the spanning tree is 8. In other words, the levels are numbered from 0 to 8. Root node is located on level 0.
- Maximum number of ports for each switch is 63 numbered from 1 to 63.

The 48-bits address consists of eight 6-bits numbered right to left from 1 to 8. Each 6-bit is related to the corresponding level. In other words, 6-bits i will be used by a parent node located on level $(i-1)$ to allocate the addresses of its children located on level i .

Each child has a unique ID number that is the parent's port number that the child is connected to parent via it. Therefore, each parent has maximum 63 children numbered from 1. The Root address is zero (48 bits of zero). Root as a parent assigns addresses to its children. To do this, it inserts in the rightmost 6-bits, the child's port number. Therefore, the addresses of nodes located on level 1 are from 1 to 63. Now, each node on level 1 assigns addresses to its own children. To do this, it inserts in the second 6-bits the corresponding child's port number. Therefore, the addresses of nodes located on level 2 are from $1 \times 2^6 + 1 = 65$ to $63 \times 2^6 + 63 = 4095$. This process will be continued level by level.

In the rest of this paper, to make the addresses more compact and easier to read, we write them in decimal form with a decimal point (dot) separating the 6-bits. Furthermore, we abbreviate the address by omitting the leftmost zeros. As an example, see Figure 3.

By using hierarchical addressing, each switch can identify its corresponding level by investigating its allocated address. Also, each switch knows the allocated addresses of its children, its parents and its grand-parents.

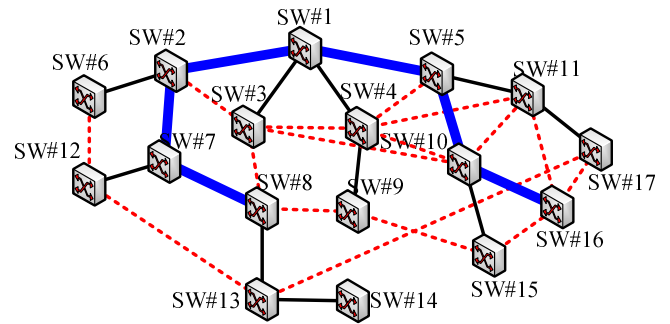


Figure 4. Forwarding path for STP .

Here, we define Common Ancestor (CA) node as the first common ancestor of two nodes when we move towards the Root. It is the rightmost common numbers of two nodes' addresses. For example, the common ancestor of nodes 1.2.1 and 1.3 is the node 0 (Root), the common ancestor of nodes 1.2.1 and 1.1 is node 1 and the common ancestor of nodes 2.1 and 1.2.2.1 is node 2.1.

C. Using new forwarding strategy

After re-addressing the nodes, frame forwarding can be easier than before. To do this, each switch generates and updates a Neighbor Nodes (NN) list. It consists of all accessible nodes via its active and blocked ports. When a switch receives a frame of data, searches its own NN list to select the best next hop node. The next hop node will be the node with lowest number of hop counts to the destination. Actually, when a frame is received, the following strategy will be applied:

- If the destination address is available in NN list, the next hop will be the destination.
- If the destination address is not available in NN list, the number of hops to destination for all of the neighbors will be calculated. Then, the nearest neighbor to the destination will be selected as next hop.

To calculate the hop counts, we use the concept of Common Ancestor (CA). It is clear that, the number of hops from any node i to destination node j , is equal to:

$$H(i, j) = H_{CA}(i) + H_{CA}(j) \quad (1)$$

where $H_{CA}(i)$ and $H_{CA}(j)$ are the number of hop counts from nodes i and j to their Common Ancestor (CA), respectively. For example for two nodes $i=1.1$ and $j=2.2.1$, the CA is node 1; therefore: $H_{CA}(i) = 1$, $H_{CA}(j) = 2$ and $H(i, j) = 3$ hops.

Now, suppose that node 2.2.1 wants to send a frame of data to node 2.1.4. The NN list of 2.2.1 is: {2.1, 2, 1.3, 1.2.2.1}. The destination node (node 2.1.4) is not in the NN list. Also, $H(2.1, 2.1.4) = 5$, $H(2, 2.1.4) = 4$, $H(1.3, 2.1.4) = 5$ and $H(1.2.2.1, 2.1.4) = 8$; therefore, node 2 will be selected as the next hop.

In a similar manner, the NN list of node 2 is: {1, Root, 3, 1.4, 2.2.1}. The destination node (node 2.1.4) is not in the NN list. Also, $H(1, 2.1.4) = 4$, $H(Root, 2.1.4) = 3$, $H(3, 2.1.4) = 3$, $H(1.4, 2.1.4) = 1$ and $H(2.2.1, 2.1.4) = 6$; therefore, node 1.4 will be selected as the next hop. For node 1.4, the NN list is: {2, 3, 4, 2.4, 1.1.4, 2.1.4}. As the destination node (node

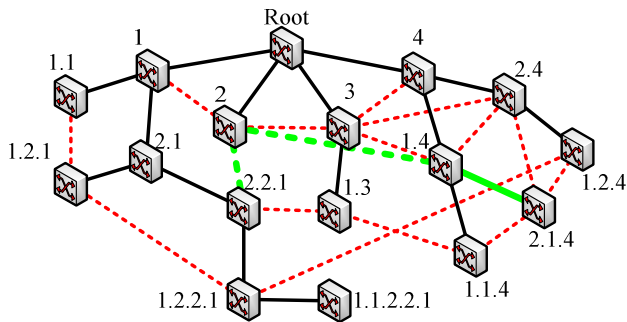


Figure 5. Forwarding path for HFS

2.1.4) is in the NN list, the next hop will be the final destination node. For the above example, forwarding path of the STP is shown in Figure 4 by blue ticked lines and forwarding path in HFS is shown in Figure 5 by green ticked lines. As you can see, in this example the number of hops for STP is 6 hops, while for HFS is only 3 hops.

As we will see in the next section, by using our new approach in Ethernet networks, we can improve the load balancing on links and switches. Also, using HFS can prevent the huge traffic on links and switches close to the Root.

IV. SIMULATION RESULTS

We implemented our proposed algorithm in MATLAB. For simulation, we used two different network topologies: Topology #1 is the typical network topology shown in Figure 1. Topology #2 shown in Figure 6 is FM-PM topology that is a popular topology for metro Ethernet networks. FM-PM topology which is described in [10], is with Full-Mesh structure in core region and Partial-Mesh structure in aggregation region. In simulated networks, the bandwidths of links are assumed the same and equal to 1Gbps and the switching capacities of switches are also assumed the same and equal to 100 Gbps. In our simulations, we set the link costs all the same, and the traffic between each node pair is assumed to be constant bit rate (1Mbps).

To compare the results, we use four parameters: average and variance of the normalized link loads (σ_l^2), average and variance of the normalized switch loads (σ_s^2) [6]. σ_l^2 is a useful parameter that indicates the degree of link load balancing in the network and σ_s^2 is a useful parameter that indicates the degree of switch load balancing [10].

The simulation results are shown in Figures (7-10) and Tables 1 and 2. As you can see in Figures 7 and 8, load balancing on links in HFS is better than STP and SSS in both topologies. Load balancing in network topology #2 is better than network topology #1. This is because of having more blocked links in topology #2. In this topology, the core region is full-mesh connected.

Also, as it is shown in Figures 9 and 10, we have better load balancing on switches in HFS in both topologies. Numerical results shown in Tables 1 and 2 indicate that HFS can achieve better results in comparison to STP and SSS in both topologies.

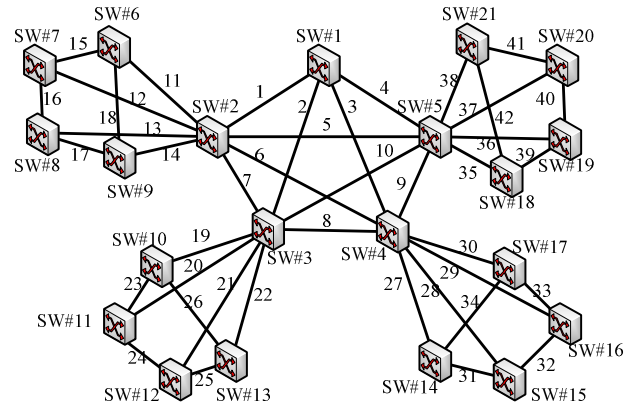


Figure 6. A typical FM-PM topology (Topology #2)

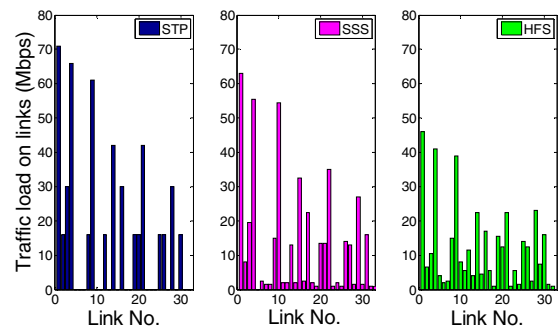


Figure 7. Traffic loads on links in Topology #1

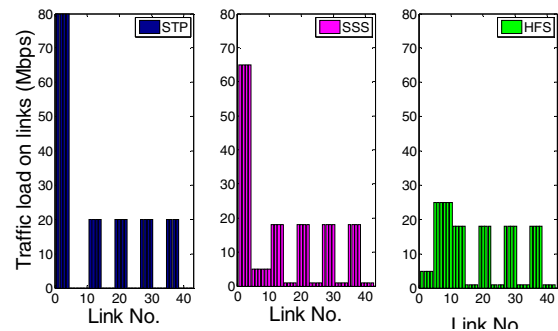


Figure 8. Traffic loads on links in Topology #2

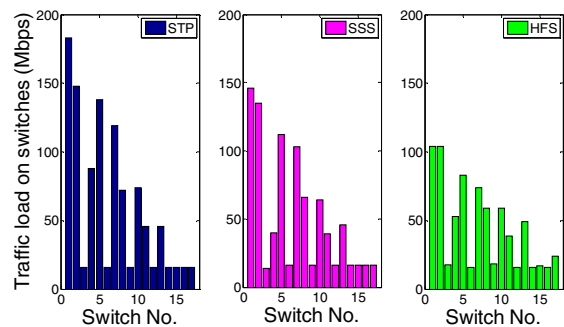


Figure 9. Traffic loads on switches in Topology #1

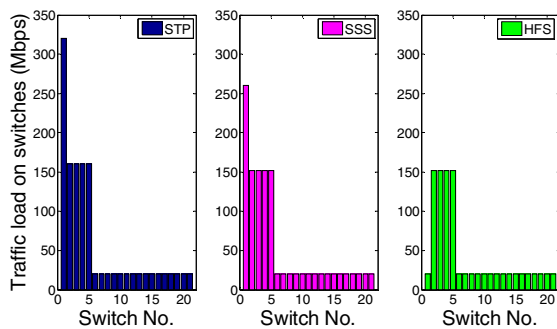


Figure 10. Traffic loads on switches in Topology #2

V. CONCLUSIONS

In this paper, we introduced a new forwarding algorithm named *Hierarchical Forwarding Strategy* that uses hierarchical addressing in Metro Ethernet Networks. In the proposed strategy, each node selects the neighbor with lowest number of hop counts to the destination as the next hop. To show the effectiveness of the proposed approach, we simulated a typical topology and the well-known FM-PM topology. Simulation results showed that HFS decreases the traffic on links and switches and has a good load balancing on links and switches.

REFERENCES

- [1] M. Huynh, and P. Mohapatra, "Metropolitan Ethernet Network: A Move from LAN to MAN," *Journal of Computer Networks, Elsevier*, Vol. 51, No. 17, pp. 4867-4894, 2007.
- [2] Rute C. Sofia, "A Survey of Advanced Ethernet Forwarding Approaches," *IEEE Communications surveys & tutorials*, Vol. 11, No. 1, pp. 92-115, First Quarter 2009.
- [3] T. Rodeheffer, C. Thekkat, and D. Anderson, "Smartbridge: A scalable bridge architecture," *ACM Computer Communication Review*, vol. 30, no. 4, pp. 205-218, 2000.
- [4] G. Ibanez, A. Garcia-Martinez, and A. Azcorra, "Alternative Multiple Spanning Tree Protocol (AMSTP) for Optical Ethernet Backbones," *Proc. of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, 2004.
- [5] S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh, "Viking: A Multi-Spanning-Tree Ethernet Architecture for Metropolitan Area and Cluster Networks," *INFOCOM*, Vol. 4, pp. 2283-2294, 2004.
- [6] G. Mirjalily, H. Karimi, and S. Rajaei, "Load Balancing in Metro Ethernet Networks by Selecting the Best Spanning Tree," *Journal of Information Science and Engineering*, vol. 27, pp. 1747-1759, 2011.
- [7] F. Faghani, G. Mirjalily, "Shortcut Switching Strategy in Metro Ethernet networks," *Journal of Computer Communications, Elsevier*, Vol. 34, No. 8, pp. 1022-1032, 2011.
- [8] W. Qiu, E. Skafidas, and P. Hao, "Enhanced Tree Routing for Wireless Sensor Networks," *Journal Ad Hoc Networks, Elsevier*, No. 8, pp. 638-650, 2009.
- [9] R. Seifert, and J. Edwards, *The All-New Switch Book*, Wiley Publication, 2008.
- [10] F. Faghani, G. Mirjalily, R. Saadat, and F. Hendessi, "Toward Benchmarks for Evaluation of Forwarding Strategies in Metro Ethernet Networks," *Proc. of International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN)*, pp. 114-119, Bali, Indonesia, 2011.

TABLE 1. COMPARISON OF PERFORMANCE METRICS IN TOPOLOGY #1

Parameter	STP	SSS	HFS	Improvement of HFS vs SSS	Improvement of HFS vs STP
Average of the Link Loads	15.62	13.81	11.95	13.46%	23.50%
Average of the Switch Loads	58.82	52.01	45.03	13.46%	23.50%
Variance of the Link Loads	42.4×10^{-5}	34.2×10^{-5}	13.6×10^{-5}	59.98%	67.76%
Variance of the Switch Loads	2.90×10^{-7}	2.15×10^{-7}	0.92×10^{-7}	56.34%	68.08%
Average of Hops	3.67	3.25	2.81	13.46%	23.50%

TABLE 2. COMPARISON OF PERFORMANCE METRICS IN TOPOLOGY #2

Parameter	STP	SSS	HFS	Improvement of HFS vs SSS	Improvement of HFS vs STP
Average of the Link Loads	15.23	14.14	11.28	20.20%	25.94%
Average of the Switch Loads	62.5	55.25	47.81	20.20%	25.94%
Variance of the Link Loads	52.9×10^{-5}	32.9×10^{-5}	8.81×10^{-5}	73.28%	83.37%
Variance of the Switch Loads	6.37×10^{-7}	5.02×10^{-7}	2.68×10^{-7}	46.53%	57.65%
Average of Hops	3.04	2.82	2.25	20.20%	25.94%