Energy-Efficient VNF Replication in Virtualized Data Centers

Master's Project – Fall 2017

By

Janani Janardhanan

Faculty Advisor: Dr. Bin Tang

Committee Member: Dr. Mohsen Beheshti

Committee Member: Dr. Jianchao Jack Han

Overview

Abstract

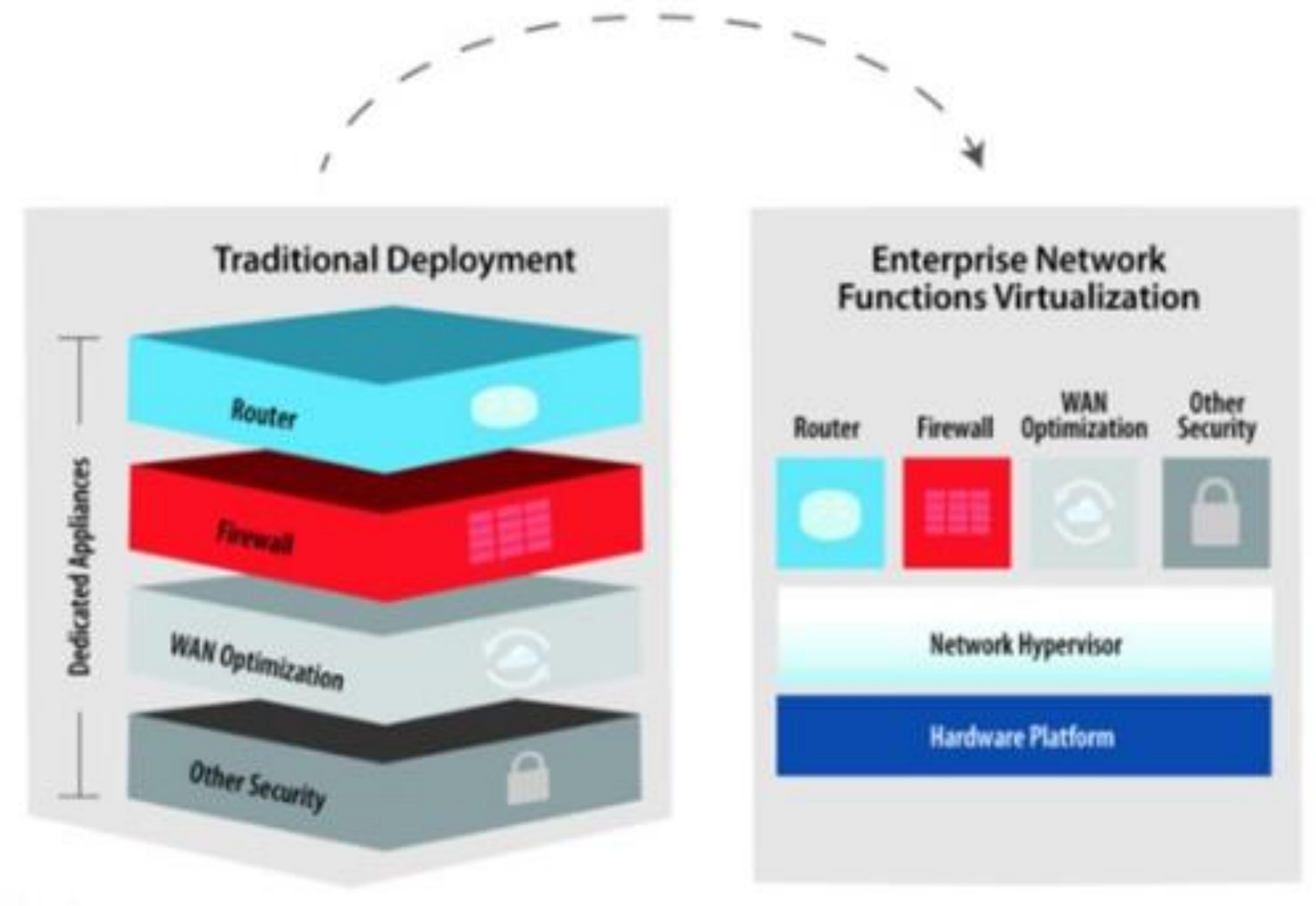
- VNF – Virtual Network Function.
- Implementation of network functions/middleboxes – Eg., Firewall, Intrusion Detection System, WAN optimizer etc.
- Most of the existing researches focus only on optimal placement of VNFs.
- This project provides effective solutions to VNF/middlebox replication problem for Fat-tree data centers.
- Heuristic algorithms: Closest Next Middlebox First (CNMF), Exhaustive MiddleBox Replication(EMBR), Traffic-Aware VNF Replication (TAVR).
- EMBR and TAVR accomplish better energy-efficiency.
- TAVR outperforms EMBR by approximately 12% with increase in middlebox types and communicating VM pairs .

Introduction

- What is NFV ?
  - ❑ Network function virtualization (NFV) is an innovative network architecture paradigm.
  - ❑ Consolidates many network equipment types onto industry standard high volume servers, switches, and storages.
  - ❑ NFV is based on the concept of Virtual network Functions(VNF).

- What is VNF?
  - ❑ Abstract building block to process network traffic to accomplish a task. Eg., firewall, IDS etc.

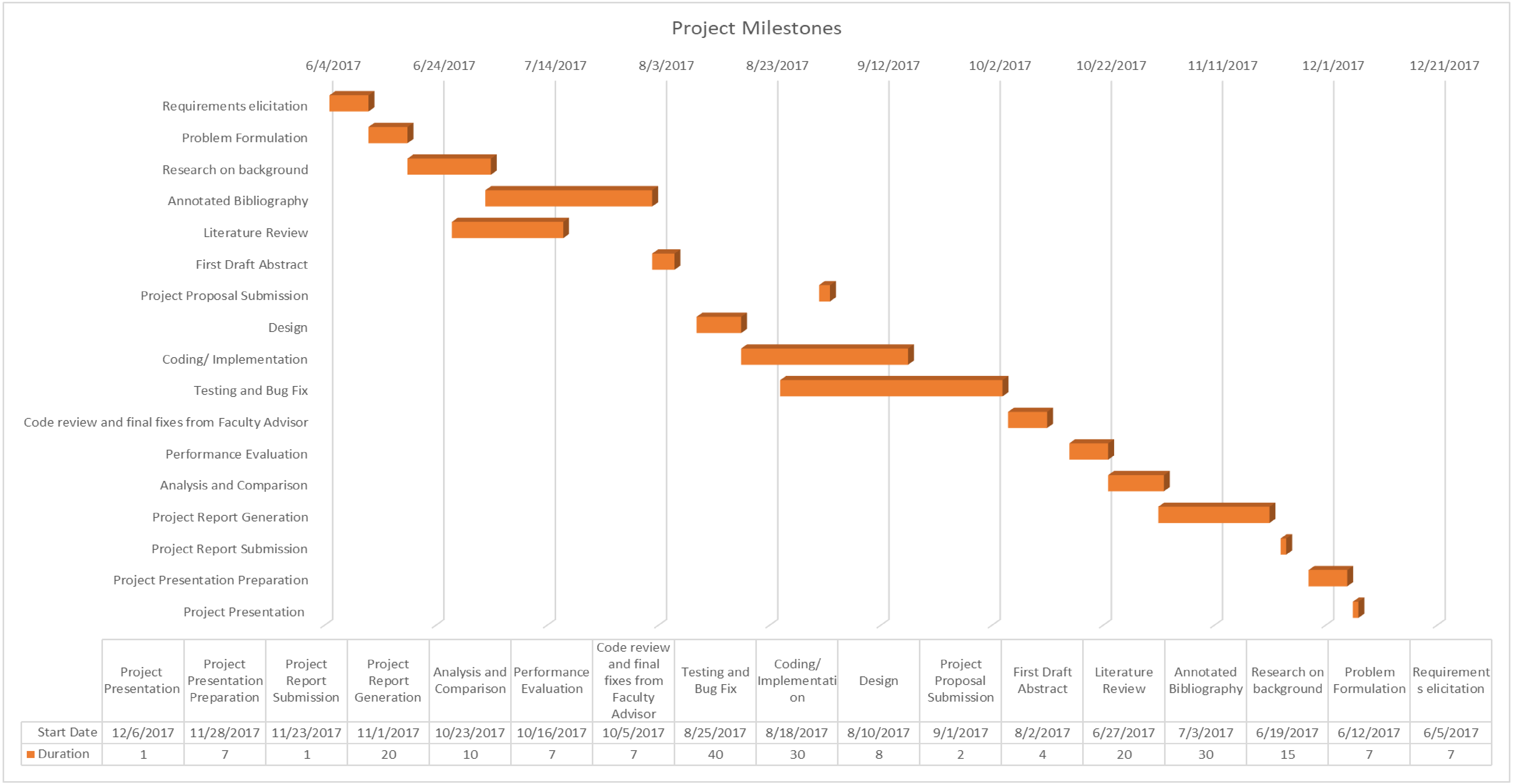
- Why are virtualizing network functions significant?
  - ❑ VNFs were previously dedicated hardware
  - ❑ Cost-effective, open interface, flexibility, energy-efficient, rapid service.



Source: "youdailytech.com"

- What is service chaining?
  - ❑ An ordered list of network functions to serve a network traffic.
  - ❑ With VNFs, easy implementation, on-time recovery, more automation and quick software upgrades are possible.

Project Schedule



	Project Presentation	Project Presentation Preparation	Project Report Submission	Project Report Generation	Analysis and Comparison	Performance Evaluation	Code review and final fixes from Faculty Advisor	Testing and Bug Fix	Coding/ Implementation	Design	Project Proposal Submission	First Draft Abstract	Literature Review	Annotated Bibliography	Research on background	Problem Formulation	Requirements elicitation
Start Date	12/6/2017	11/28/2017	11/23/2017	11/1/2017	10/23/2017	10/16/2017	10/5/2017	8/25/2017	8/18/2017	8/10/2017	9/1/2017	8/2/2017	6/27/2017	7/3/2017	6/19/2017	6/12/2017	6/5/2017
🟧 Duration	1	7	1	20	10	7	7	40	30	8	2	4	20	30	15	7	7

Specification of Requirements

- NFV allows us to organize network functions, like building blocks to create communication services that can be deployed quickly and allow increased growth.
- Software Defined Networking (SDN) paired with NFV can reduce costs for service providers.
- Placing replicas of the service chain in the network greatly helps to load balance as well as serve as backups.
- The ultimate goal of this project is to design and implement efficient algorithms to create multiple copies of an ordered sequence of virtual network functions in the Data Center Network such that minimum cost flow is ensured along with providing dynamic provisioning, load balancing and high availability.

Middlebox Replication Problem (MRP)

- There are m middleboxes (of different types) $M = \{mb_1, mb_2, \ldots, mb_m\}$, where mb_j $(1 < j < m)$ is located at switch $SW_j \in V_s = \{SW_1, SW_2, \ldots, SW_{|Vs|}\}$.
- V_s is the set of switches holding the replicas of the middlebox instances distributed across the network.
- Each switch has a capacity, indicating number of middleboxes it can store. The capacity of switch SW_i is *cap(k)*.
- The objective of MRP is to replicate middleboxes and place them onto switches such that the capacity constraint is satisfied and also when each communicating VM pairs traverse to one instance of mb_1, mb_2, …mb_m, each in that order, it results in minimum communication cost .

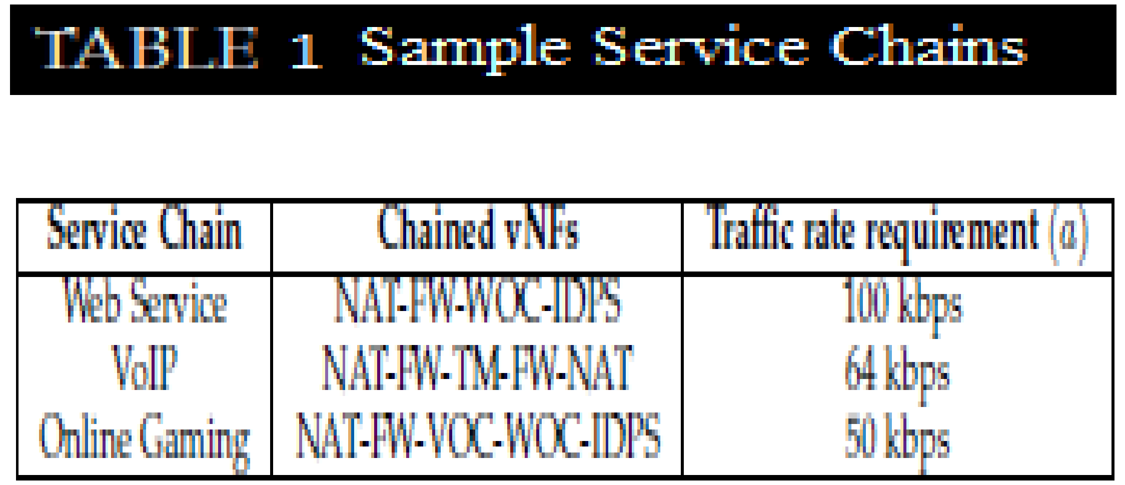


TABLE 1 Sample Service Chains

Service Chain	Chained vNFs	Traffic rate requirement (α)
Web Service	NAT-FW-WOC-IDPS	100 kbps
VoIP	NAT-FW-TM-FW-NAT	64 kbps
Online Gaming	NAT-FW-VOC-WOC-IDPS	50 kbps

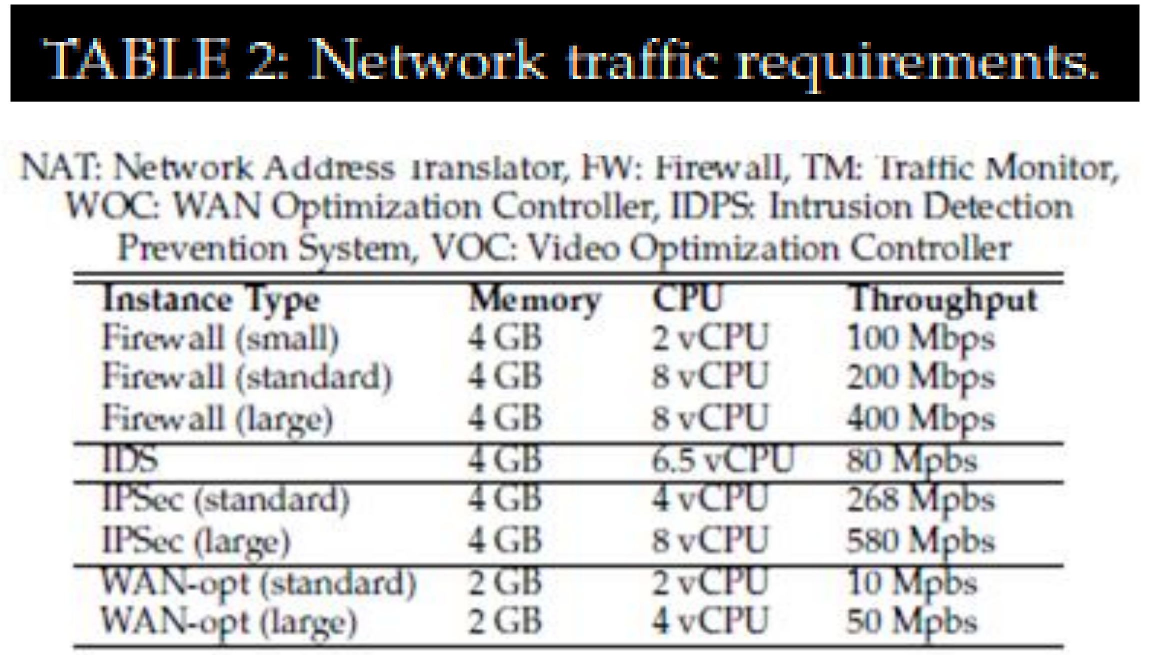


TABLE 2: Network traffic requirements.

NAT: Network Address Translator, FW: Firewall, TM: Traffic Monitor, WOC: WAN Optimization Controller, IDPS: Intrusion Detection Prevention System, VOC: Video Optimization Controller

Instance Type	Memory	CPU	Throughput
Firewall (small)	4 GB	2 vCPU	100 Mbps
Firewall (standard)	4 GB	8 vCPU	200 Mbps
Firewall (large)	4 GB	8 vCPU	400 Mbps
IDS	4 GB	6.5 vCPU	80 Mpbs
IPSec (standard)	4 GB	4 vCPU	268 Mpbs
IPSec (large)	4 GB	8 vCPU	580 Mpbs
WAN-opt (standard)	2 GB	2 vCPU	10 Mpbs
WAN-opt (large)	2 GB	4 vCPU	50 Mpbs

Source: [5]

MRP Problem Formulation

- Phase 1: Efficient Replication
  - Select a set of switches S_j = {S_1, S_2, S_3.. S_m}, where S_j is the set of switches that stores an instance of mb_j.
  - The objective of MRP is to find host switches under the constraint that switch SW_k ($1 < k < | V_s |$) does not store more than *cap(k)* middleboxes.

- Phase 2: Choosing best middlebox sequences for each Virtual Machine(VM) pair
  - For each VM pair (v_i, v_i'), find the sequence of switches $mb_{i,1} \in S_1 \cup \{SW(1)\}$, $mb_{i,2} \in S_2 \cup \{SW(2)\}$, etc. and finally, $mb_{i,m} \in S_m \cup \{SW(m)\}$ to traverse in that order to visit each middlebox instance, such that total communication cost is minimized.

- Expected solution:
  - Communication cost for one VM Pair:

$$C_i^r = c(S(v_i), mb_{i,1}) + \sum_{j=1}^{m-1} c(mb_{i,j}, mb_{i,j+1}) + c(mb_{i,m}, S(v_i'))$$

  - For 'p' Vm pairs, the communication cost is :

$$C^r = \sum_{i=1}^{p} C_i^r = c(S(v_i), mb_{i,1}) + \sum_{j=1}^{m-1} c(mb_{i,j}, mb_{i,j+1}) + c(mb_{i,m}, S(v_i'))$$

  - The objective is to obtain the middlebox distribution under capacity constraints and with C^r_{min} .

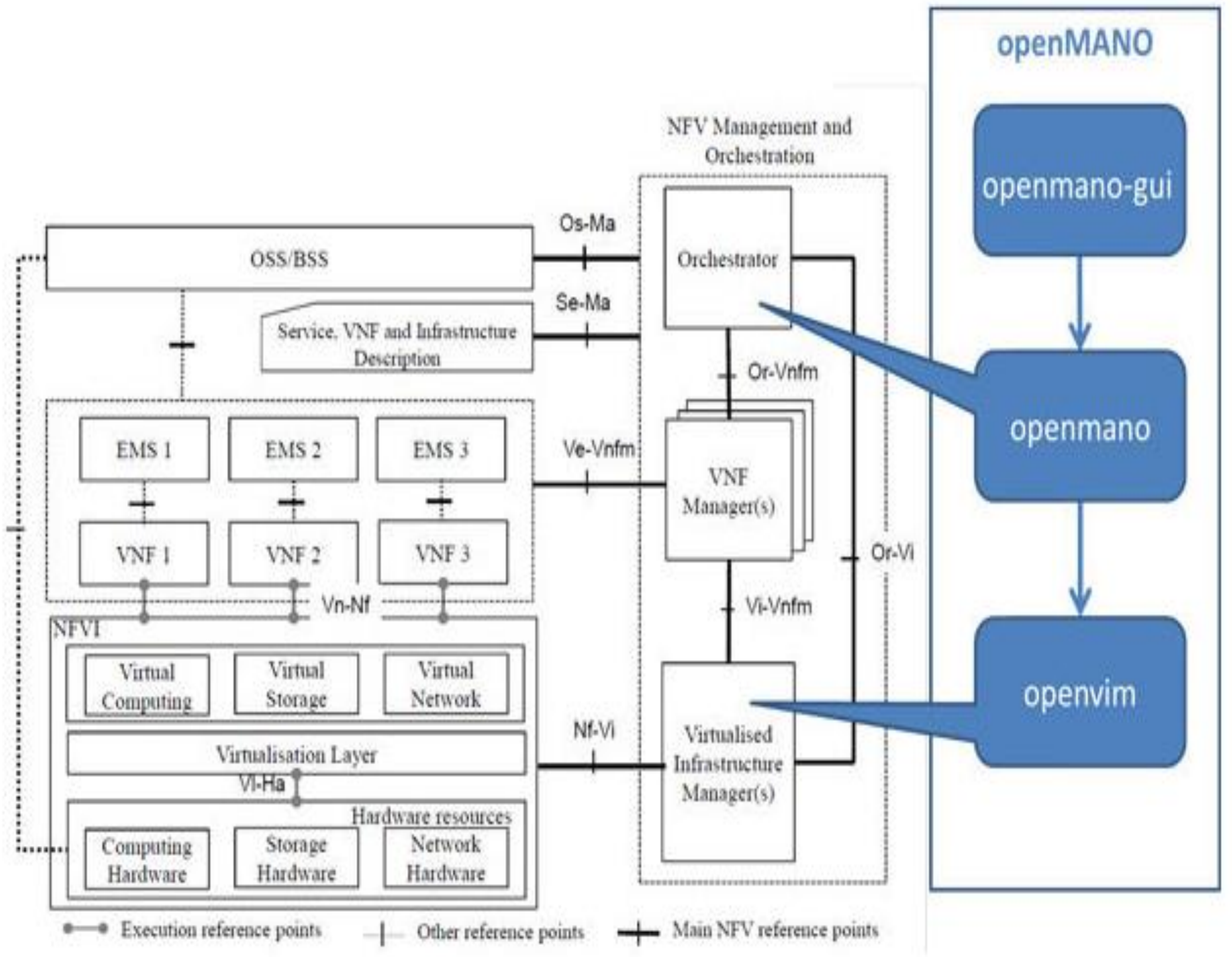
Background/Literature Review

- Industrial/ Practical applications:
  - ❑Communication Service Providers spend huge amounts of money buying and maintaining specialized network hardware; thus, companies such as AT&T, Sprint, CenturyLink and other global CSPs have been receiving much of the attention from vendors who are working on NFV solutions [8].

- Related Existing Researches on VNFs:
  - ❑ Optimal VNF Placement [5]:
    - ➢ Sampling based approach using markov chains.
    - ➢ Reducing state space of feasibilities.
  - ❑ VNF replication for providing load balancing [4]:
    - ➢ Focus only on load balancing and not in optimized use of resources and link cost.
  - ❑ Optimized VNF replication across distributed data center for mobile networks [6]:
    - ➢ Very similar intention like ours but optimization is considered across data centers.
    - ➢ Algorithms are suitable only for mobile networks.

Network Architecture

1. NFV Architecture

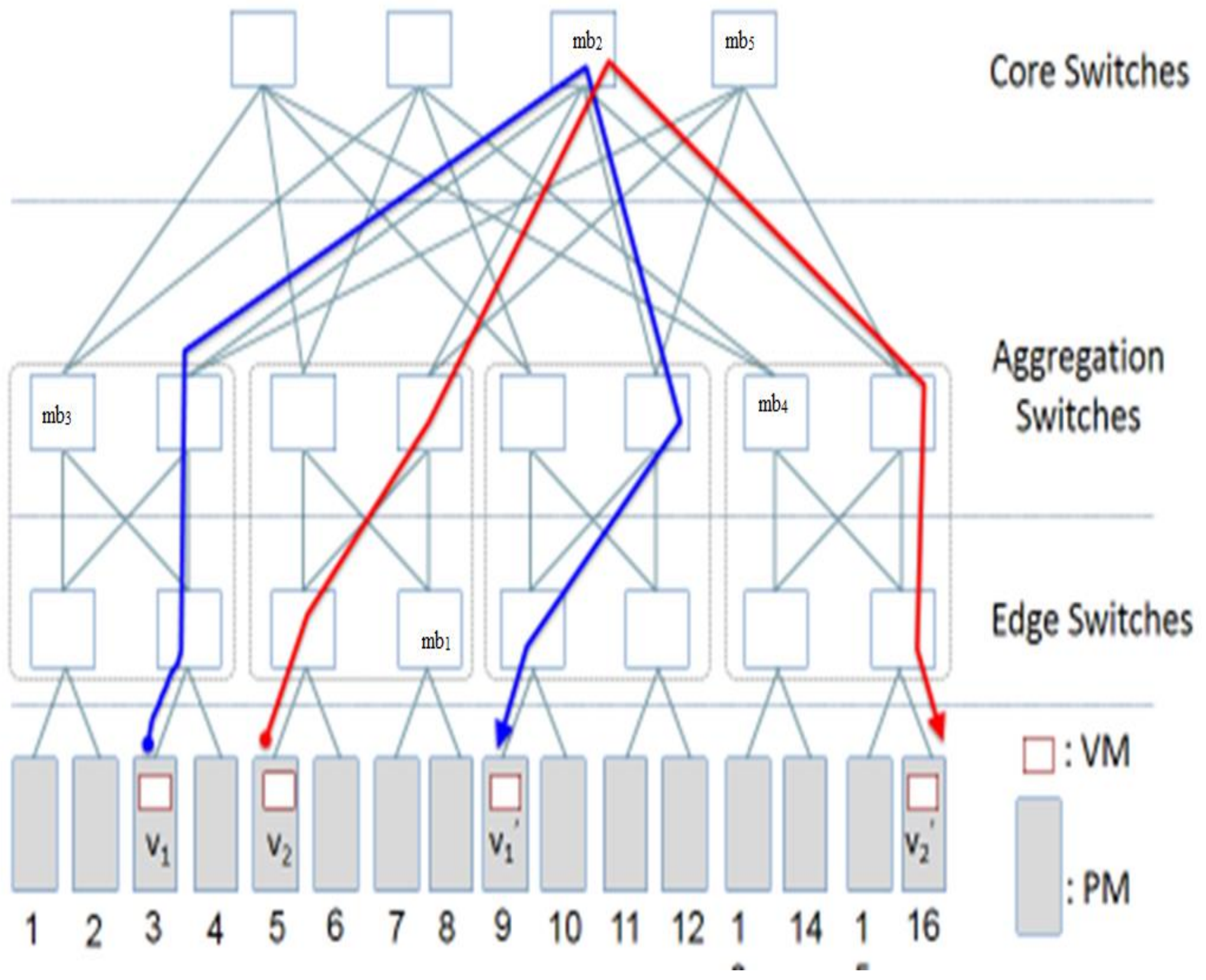


Source: "www.sdxcentral.com"

- The NFV architecture is basically described by three components: Services, NFV Infrastructure (NFVI) and NFV Management and Orchestration (NFV-MANO).
- A Service is the composition of VNFs that can be implemented in virtual machines running on operating systems or on the hardware directly.
- The hardware and software resources are provided by the NFVI that includes computing, storage, networking etc.
- NFV-MANO is composed by the orchestrator, VNF managers and Virtualized Infrastructure Manager.

Network Architecture (contd.)

2. Architecture of Fat-tree topology



- Fat Tree topologies are popular for their nonblocking nature, providing many redundant paths between any 2 hosts.
- A Fat Tree consists of k pods, each containing two layers of k/2 switches namely edge switches and aggregation switches.
- Each k-port switch in the lower layer (edge switch) is directly connected to k/2 hosts.
- Each of the remaining k/2 ports is connected to k/2 of the k ports in the aggregation layer of the hierarchy.
- There are $(k/2)^2$ K-port core switches. Each core switch has one port connected to each of the k pods.
- Thus, in total there are $5k^2/4$ switches in the network. Also, fat-tree topology supports connecting $k^3/4$ physical machines or hosts to the edge switches.

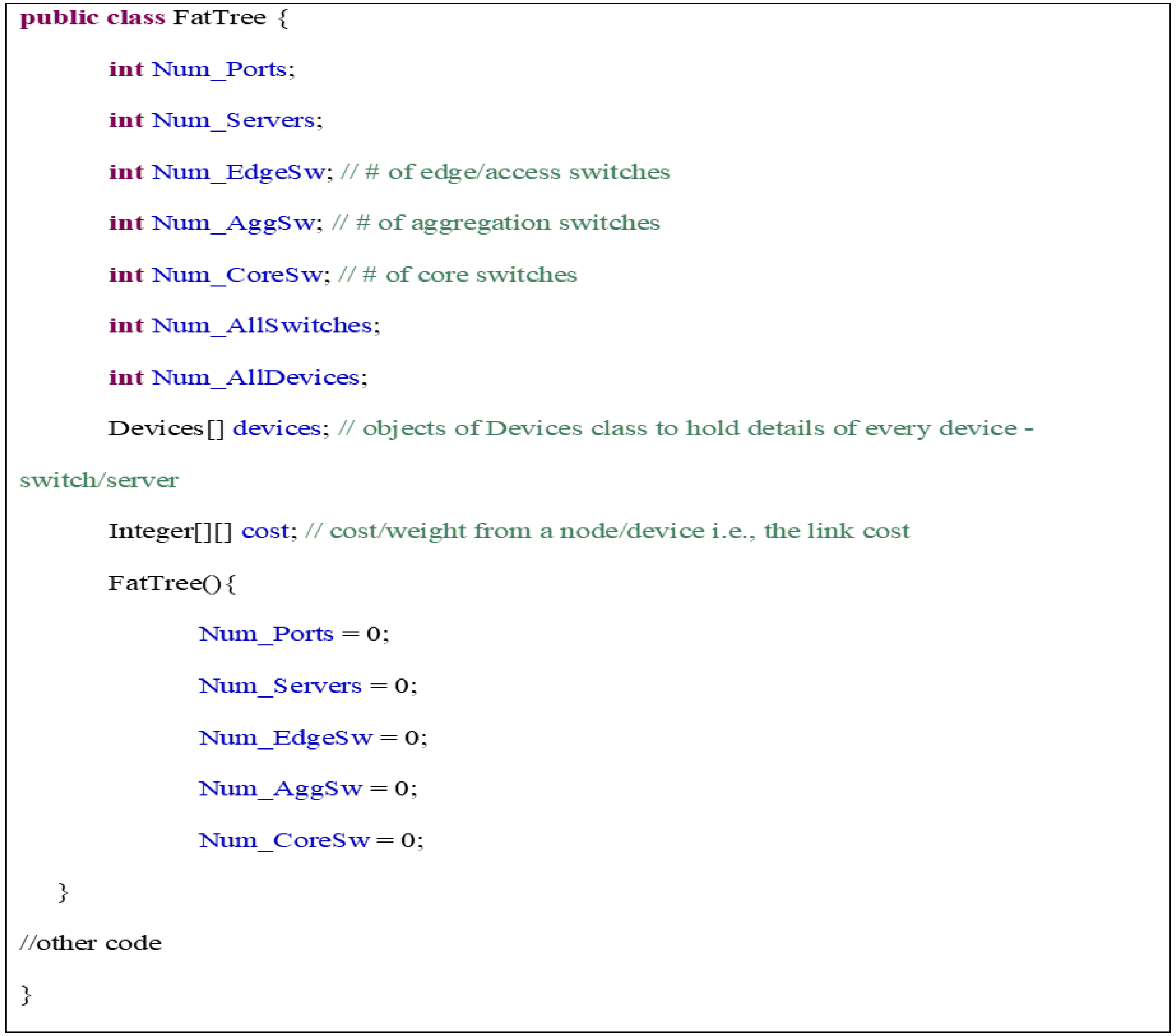
Design

- Methodology – Object Oriented Design(OOD)
- Reasons – To avail various OO paradigms like
  - Encapsulation, Inheritance, Polymorphism
  - Aggregation
- Classes or Entities
  - Device
    - A common class/entity that might be instantiated for a server/switch.
  - Fat-Tree
    - The class whose object is the object of the fat-tree network based on ‘K’ value from user input. The Fat-Tree class aggregates an array of Device class objects.
  - Proposed Algorithm
    - Every proposed algorithm is implemented as a separate class aggregating Fat-Tree class.

The Device class

```
class Devices{
        int DeviceID;
        int capacity;
        boolean isServer;
        int podID;
//      boolean isVirtual;
        ArrayList<Integer> VM;
        ArrayList<Integer> MB;
        ArrayList<Integer> mb_preference_list;
        ArrayList<Integer> neighbors;
        final static int Server_Capacity = 10; //# of VMs a server holds
        final static int Switch_Capacity = 1;  //# of MBs a switch holds
        Devices(int id, int capacity, boolean isServer){
                this.DeviceID = id;
                this.capacity = capacity;
                this.isServer = isServer;
                this.neighbors = new ArrayList<Integer>();
                if(this.isServer){
                        VM = new ArrayList<Integer>();
                                mb_preference_list = new ArrayList<Integer>();
                                MB = null;
                        }
                        else{
                                VM = null;
                                mb_preference_list=new ArrayList<Integer>();
                                MB = new ArrayList<Integer>();
                        }
                }
        }
```

The Fat-tree Class



```
public class FatTree {
        int Num_Ports;
        int Num_Servers;
        int Num_EdgeSw; // # of edge/access switches
        int Num_AggSw; // # of aggregation switches
        int Num_CoreSw; // # of core switches
        int Num_AllSwitches;
        int Num_AllDevices;
        Devices[] devices; // objects of Devices class to hold details of every device -
switch/server
        Integer[][] cost; // cost/weight from a node/device i.e., the link cost
        FatTree(){
                Num_Ports = 0;
                Num_Servers = 0;
                Num_EdgeSw = 0;
                Num_AggSw = 0;
                Num_CoreSw = 0;
    }
//other code
}
```

Then, the Possible operations on the fat-tree network were implemented as the methods of FatTree class. For eg.,

- Create a barebone fat-tree network based on the given 'K'.
- Randomly distribute the virtual machines across the servers.
- Randomly pair up different virtual machines.
- Place one original sequence of middlebox instances on the network.
- Calculate the cost or distance of every node from every other node in the network.
- Calculate traffic flow cost when traffic flows between one VM and another in a VM pair.
- Reset the fat-tree network to its initial state.

The Proposal Framework

- Proposed Algorithms:

1. Random Replication Algorithm
2. Closest Next Middlebox First Algorithm
3. Exhaustive Middlebox Replication Algorithm
4. Traffic-Aware VNF Replication algorithm

- Pre-requisites/constraints:

1. Expects a fat-tree network with three tiers and (5 k^2 /4) switches and (k^3 /4)servers.
2. Expects network functions to be service chains.
3. Maximum possible replications R_{max} is set to $5k^2/4m$. This can be changed as needed.

Random Replication (RR) Algorithm

- **Input**:
  1. K – Number of ports
  2. F – An object of FatTree network
  3. M – Number of middlebox types
  4. C – The original sequence of the service chain
  5. P – The VM pairs placed on the physical machines of the network

- **Algorithm:**
  1. For every middlebox type in the service chain {mb_1, mb_2 ,…mb_m},
  2. If the current middlebox type mb_x’s replica count has not reached $R_{max,}$
  3. Randomly choose a switch as host for $mb_{x.}$
  4. If the chosen switch’s capacity satisfies the capacity constraints of mbx, place the replica copy of mb_x on that switch.
  5. Else, go to Step 3.
  6. If all middlebox types have R_{max} replicas, stop the algorithm.

Random Replication (RR) Algorithm (contd.)

- **Explanation:**
  - ❑ Every middlebox type is ensured to have R_{max} replicas in the network provided all switches satisfy the capacity constraints.
  - ❑ The host is randomly chosen by only considering the capacity of the host.
  - ❑ Once, random replica copies of all middlebox types are thus placed across the network, every VM pair can choose a random service chain to send traffic from source to destination.
  - ❑ Though random procedures can work well at times, they are not always reliable.
  - ❑ Random Replication algorithm can only be used in scenarios where VM pairs communicate very rarely and energy conservation is not significant.

- **Time-Complexity:**
  - ❑ O (R_{max} * M*5K^2/4) => O (K^4). This is the worst-case execution time for the Random Replication algorithm. In the best case, where every switch it randomly chooses for the first time is the correct host for a middlebox type mb_{m}, the time complexity is O (K^2).

Closest Next Middlebox First(CNMF) Algorithm

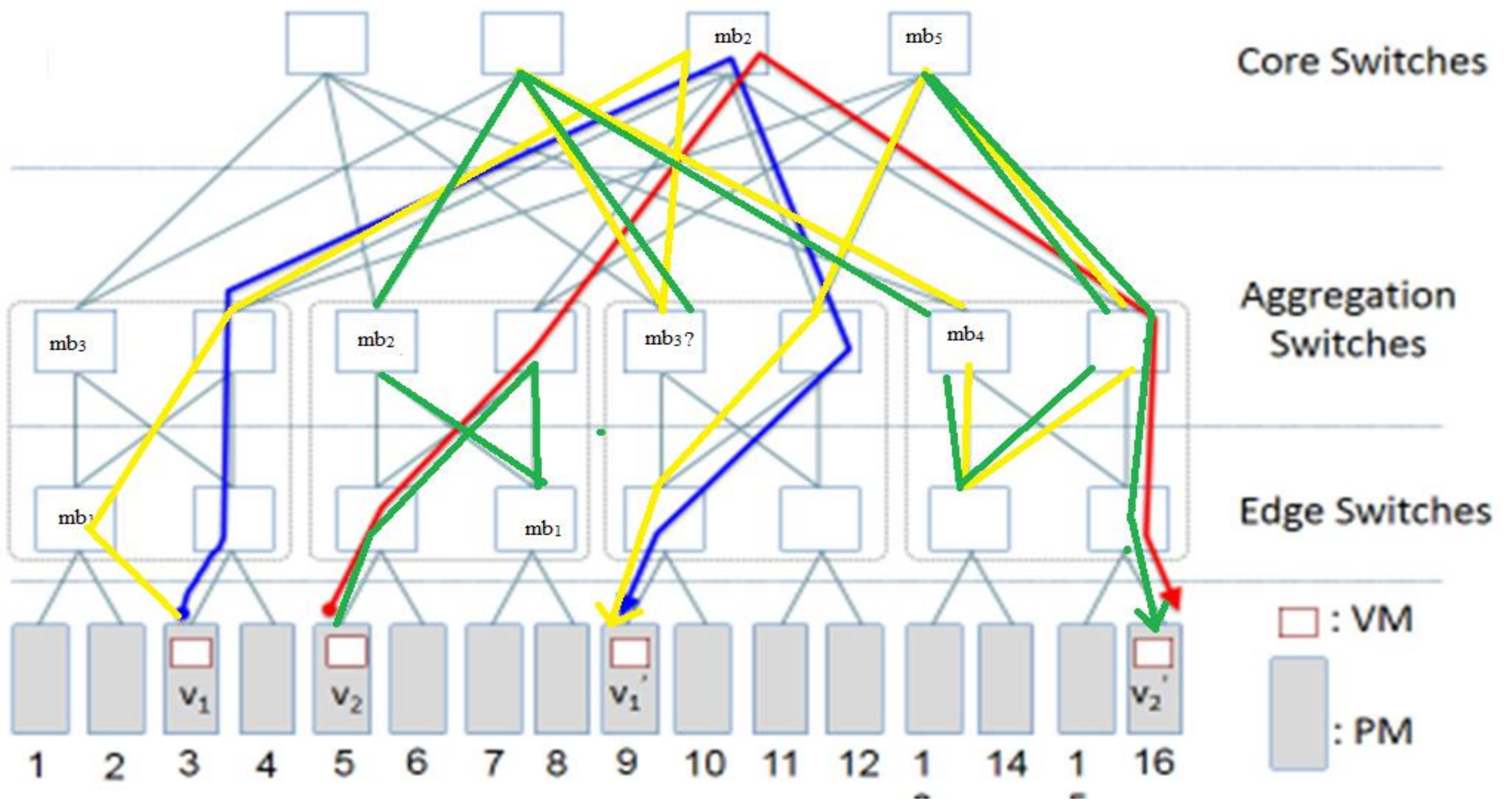
- **Input:**

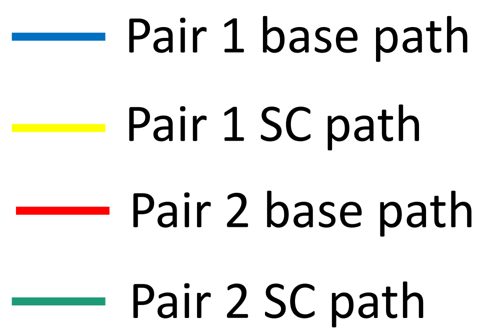
1. K – Number of ports
2. F – An object of FatTree network
3. M – Number of middlebox types
4. C – The original sequence of the service chain
5. P – The VM pairs placed on the physical machines of the network

- **Algorithm:**

```
1.  Initialize a property called next closest middlebox to every VM pair as original $mb_1$.
2.  Initialize next closest middlebox to every middlebox up to $_{mbm-1}$ in the original sequence. That is for $mb_1$, set the closest next middlebox as $mb_2$, for $mb_2$ the closest next middlebox is $mb_3$ etc.
3.  For placing every replica copy 'R' from {1, 2 ,....$R_{max\},}$
4.  For every middlebox type 'M' in the service chain {$mb_1$, $mb_2$ ,...$mb_m$},
5.  For every switch 'S' as host in the fat-tree network,
6.  If the chosen switch's capacity 'cap' satisfies the capacity constraints,
7.  For All 'P' VM pairs in the network,
8.  Choose closest next middlebox of every device up to $mb_{x.}$
9.  From all available $mb_{x+1}$, choose closest $mb_{x+1}$ to current $mb_{x.}$
10. Choose closest next middlebox from chosen $mb_{x+1}$ to $mb_{m.}$
11. Send traffic via all 'P's using the service chain obtained from step 8-10.
12. If the switch 'S' yields the minimum overall cost for that middlebox type 'M', place 'M' on 'S' and decrease its available capacity.
13. If $mb_x$ is $mb_{1,}$
        For all 'P', check if current $mb_x$ can be set as closest next $mb_1$
14. Else,
        For all 'R' replicas of $mb_{x-1}$, check and set if $mb_x$ is the closest next
```

Comparison of base and SC(Service Chain) paths during replication process





Closest Next Middlebox First Algorithm(contd.)

- Explanation
  - ❑ The algorithm replicates middlebox instances one by one by placing a middlebox instance (mb_x) in a node closest to one of the copies of mb_{x-1} instances.
  - ❑ The node that hosts an mb_x is chosen to yield the lowest overall traffic-flow cost on the network.
  - ❑ VNF replication using this method successfully places at least one copy of a middlebox type on every node on the network.
  - ❑ When all the nodes in the network have a copy of a VNF instance, the replication is done.
  - ❑ Then, each VM pair is assigned to its closest service chain for relaying traffic.
  - ❑ Shortest path may not be the best solution in all cases. This algorithm can be tremendously useful when quick set up is required.

- Time Complexity
  - ❑ $O(R_{max} * M*5K^2/4 *(2P+R_{max})) \Rightarrow O(PK^4+K^6)$ which is approximately $O(K^6)$. This is the execution time for the algorithm.

Exhaustive MiddleBox Replication (EMBR) Algorithm

- **Input:**
  1. K – Number of ports
  2. F – An object of FatTree network
  3. M – Number of middlebox types
  4. C – The original sequence of the service chain
  5. P – The VM pairs placed on the physical machines of the network

- **Algorithm:**
  1. For placing every replica copy ‘R’ from {1, 2, ...R_{max}},
  2. For every middlebox type ‘M’ in the service chain {mb_1, mb_2 ,...mb_m},
  3. For every switch ‘S’ as host in the fat-tree network,
  4. If the chosen switch’s capacity ‘cap’ satisfies the capacity constraints of $mb_{x,}$
  5. For All ‘R’ middlebox replica copies of {$mb_{1,}$ mb_2...$mb_{x-1\},}$
  6. For All ‘R-1’ middlebox replica copies of {$mb_{x+1,}$ $mb_{x+2,}$...$mb_{m\},}$
  7. For All ‘P’ VM pairs in the network,
  8. If the switch ‘S’ yields the minimum cost for that middlebox type ‘M’, place ‘M’ on ‘S’ and decrease its available capacity.

Exhaustive MiddleBox Replication (EMBR) Algorithm (contd.)

- **Explanation:**
  - ❑In this algorithm, we exhaust all possible combinations of middlebox instances so as to achieve the ideal or perfect result.
  - ❑The only drawback of this algorithm is its convergence time.
  - ❑However, it is commonly known that network orchestration for Quality of Service (QoS) services is time consuming during the initial set up, but once it is set up and is running, the service remains unaffected until disabled deliberately by the network administrator.

- **Time Complexity:**
  - ❑$O\ (R_{max} * M*5K^2/4 *R_{max}*R_{max\ *}P) \Rightarrow O\ (PK^8/M^2)$. Although the execution time is longer than CNMF, reduction in traffic cost is greatly achieved.

Traffic-Aware VNF Replication (TAVR) Algorithm

- **Input:**

1. K – Number of ports
2. F – An object of FatTree network
3. M – Number of middlebox types
4. C – The original sequence of the service chain
5. P – The VM pairs placed on the physical machines of the network

- **Algorithm:**

```
1. For all 'P' VM pairs associate them to their respective traffic frequency group in {0,1,2,3} based on their frequency of communication per time unit.
2. Calculate the probability distribution for each traffic group as follows:
3. Probability distribution of a group G = (Number of VM pairs in G/ P) where P is the total number of VM pairs available in the network.
4. For every group G, calculate the number of replications that can be allocated to that group by using the following formula:
5. Number of replicas($R_g$) for a group G = Probability distribution of G * $R_{max}$
6. Thus, for G={0,1,2,3}, $R_0 + R_1 + R_2 + R_3 = R_{max.}$
7. For every group G,
8. For every possible replica 'R' within the group from {1,2…$R_g$},
9. For every middlebox type 'M' in service chain from {$mb_{1,}$ $mb_2$ …. $mb_{m\}}$ ,
10. For every switch 'S' as host in the fat-tree network,
11. If the chosen switch's capacity 'cap' satisfies the capacity constraints of $mb_{x,}$
12. If $mb_x$ is $mb_{1,}$ create a temporary service chain from original service chain with $mb_1$ being $mb_{x.}$
13. Else, create a service chain from {$mb_1$,$mb_2$…$mb_{x-1}$} from the current replication 'R', retain $mb_x$ and choose {$mb_{x+1,....}$$mb_m$} from original service chain.
14. For all $P_g$ VMpairs belonging to that group G,
15. If current $mb_x$ yields the minimum overall traffic cost which is expected to be lesser than or equal to the original cost yielded by the service chain before replication, place 'M' on 'S'.
```

Traffic-Aware VNF Replication Algorithm (TAVR) (contd.)

- **Explanation:**
  - ❑ The VM pairs after being placed on their respective host servers are associated to a traffic class group based on their rate or frequency of communication.
  - ❑ This algorithm categorizes the VM pairs under 4 groups namely ‘Very Frequent Communicators’, ‘Frequent Communicators’, ‘Medium Communicators’ and ‘Rare communicators’.
  - ❑ Each traffic group has its own distribution count as well. For example, one of the frequency distribution is [40%,45%,12%,3%].
  - ❑ Number of replications allocated in favor of a traffic group is determined by the probability distribution of that traffic group and by the frequency of communication between each VM pair in the traffic group.
  - ❑ This replication is done in the order of priority of the traffic group; most frequently communicating VM Pairs are given the highest priority.
  - ❑ The primary advantage of this algorithm is the efficient replication of VNFs based on expected traffic flow.

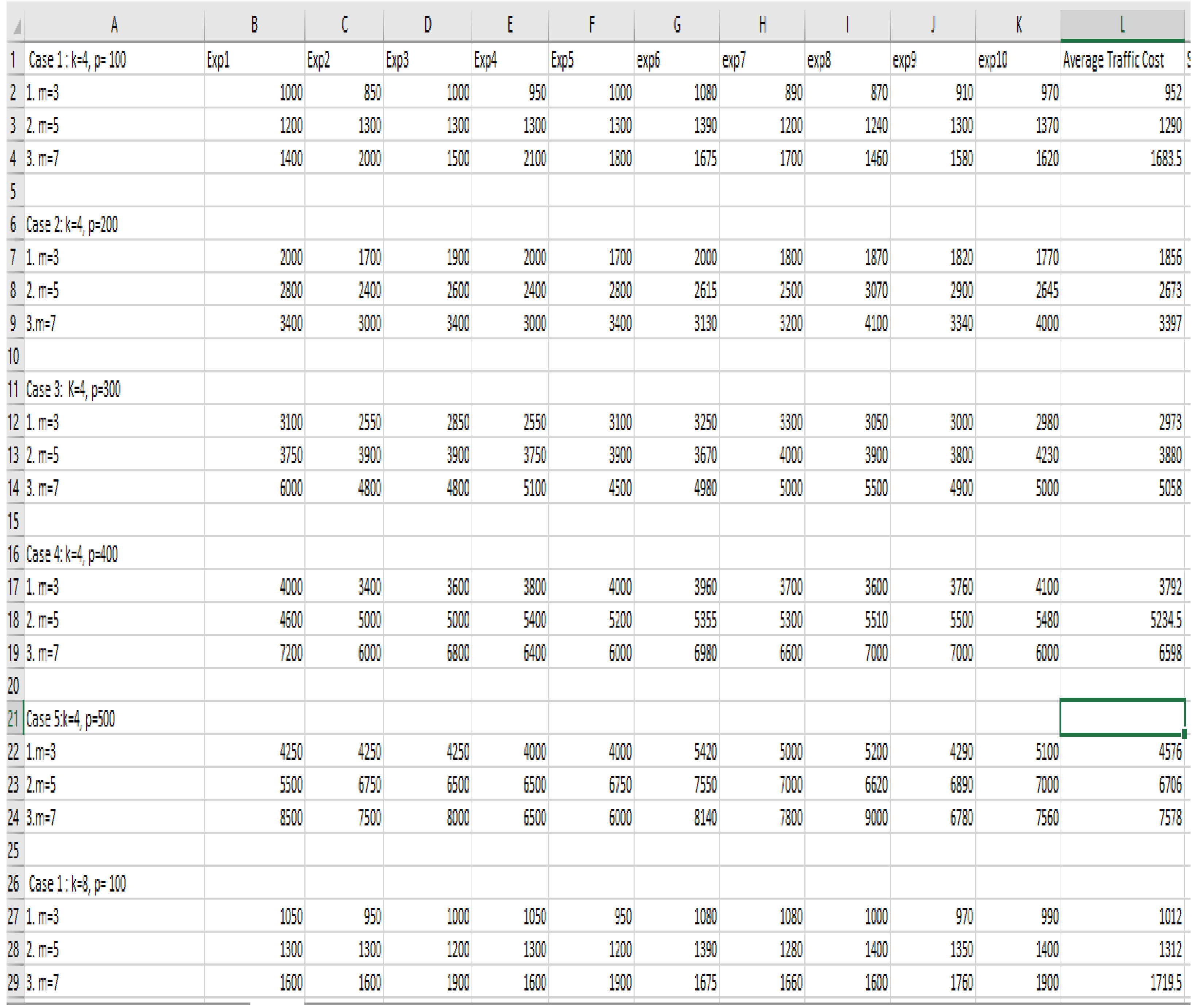
- **Time-Complexity:**
  - ❑ $O(G * R_{max} * M * 5K^2/4 * P) \Rightarrow O(G * (5K^2/4M) * M * (5K^2/4) * P) \Rightarrow O(GPK^4)$. This algorithm performs better than all proposed algorithms. Once the replicas are set up, a service chain preference list can be created for all VM pairs to choose a best service chain for each VM pair. To do that the execution time would be O(PRmax). Instead, it could also be set in Step 11-12 by checking if the current traffic cost is the minimum traffic cost yielded so far for the pair ‘p’.

Performance Evaluation

The parameters that are configured in the network during the simulations are as follows:

- ❑ *K = 4 and 8*
- ❑ *P = 100, 200, 300, 400, and 500*
- ❑ *m = 3, 5 and 7.*

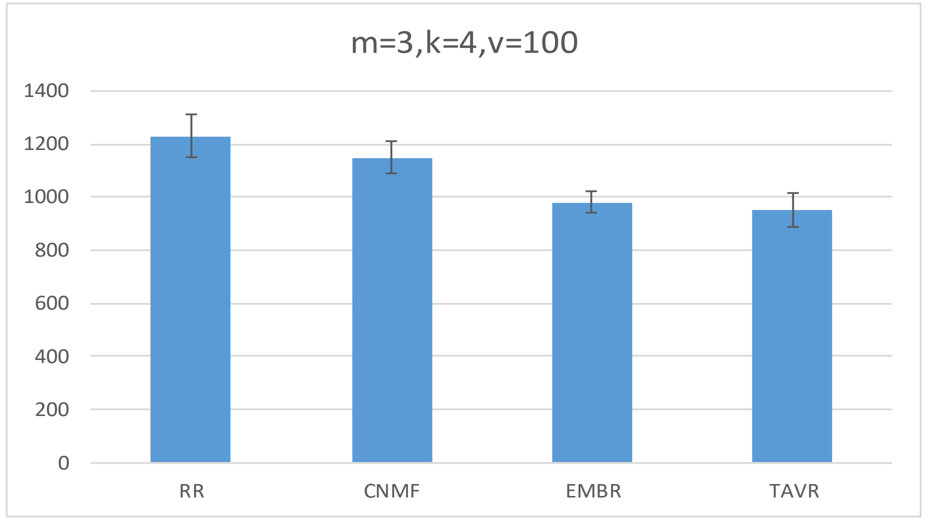
- ‘K’ is the number of switch ports in a k-ary fat-tree.
- ‘P’ is the number of VM pairs residing on the physical machines connected to the edge switches.
- ‘m’ is the number of middlebox types in the service chain. The average traffic cost for each case are plotted as graphs (column charts).
- Each algorithm was run ten times and average traffic cost were logged in Microsoft Excel’s worksheet as shown.
- Excel’s inbuilt function STEDEV.S was used to compute the standard deviation in the trials. CONFIDENCE is Excel’s inbuilt function to compute the Confidence Interval (CI).
- CI = CONFIDENCE (alpha, standard_dev, size).

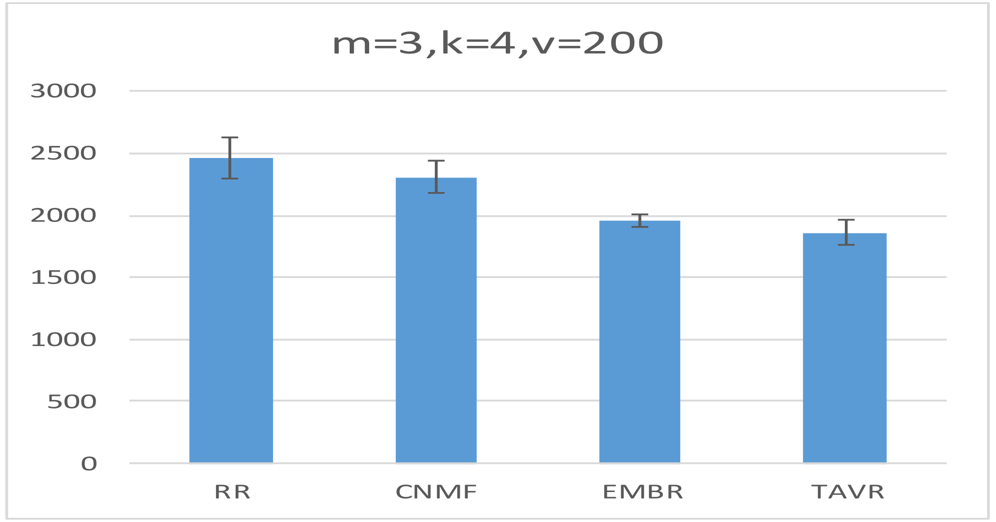


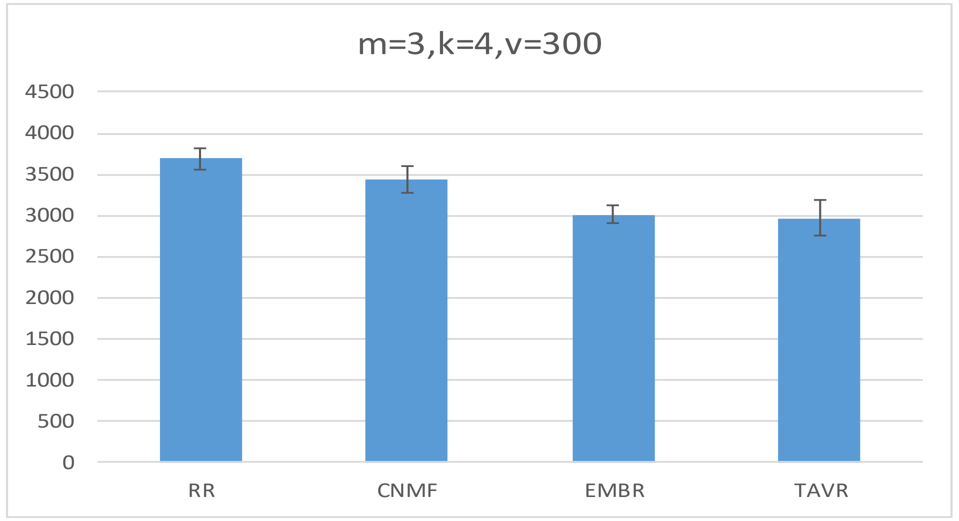
	A	B	C	D	E	F	G	H	I	J	K	L
1	Case 1 : k=4, p= 100	Exp1	Exp2	Exp3	Exp4	Exp5	exp6	exp7	exp8	exp9	exp10	Average Traffic Cost
2	1. m=3	1000	850	1000	950	1000	1080	890	870	910	970	952
3	2. m=5	1200	1300	1300	1300	1300	1390	1200	1240	1300	1370	1290
4	3. m=7	1400	2000	1500	2100	1800	1675	1700	1460	1580	1620	1683.5
5												
6	Case 2: k=4, p=200											
7	1. m=3	2000	1700	1900	2000	1700	2000	1800	1870	1820	1770	1856
8	2. m=5	2800	2400	2600	2400	2800	2615	2500	3070	2900	2645	2673
9	3.m=7	3400	3000	3400	3000	3400	3130	3200	4100	3340	4000	3397
10												
11	Case 3: K=4, p=300											
12	1. m=3	3100	2550	2850	2550	3100	3250	3300	3050	3000	2980	2973
13	2. m=5	3750	3900	3900	3750	3900	3670	4000	3900	3800	4230	3880
14	3. m=7	6000	4800	4800	5100	4500	4980	5000	5500	4900	5000	5058
15												
16	Case 4: k=4, p=400											
17	1. m=3	4000	3400	3600	3800	4000	3960	3700	3600	3760	4100	3792
18	2. m=5	4600	5000	5000	5400	5200	5355	5300	5510	5500	5480	5234.5
19	3. m=7	7200	6000	6800	6400	6000	6980	6600	7000	7000	6000	6598
20												
21	Case 5:k=4, p=500											
22	1.m=3	4250	4250	4250	4000	4000	5420	5000	5200	4290	5100	4576
23	2.m=5	5500	6750	6500	6500	6750	7550	7000	6620	6890	7000	6706
24	3.m=7	8500	7500	8000	6500	6000	8140	7800	9000	6780	7560	7578
25												
26	Case 1 : k=8, p= 100											
27	1. m=3	1050	950	1000	1050	950	1080	1080	1000	970	990	1012
28	2. m=5	1300	1300	1200	1300	1200	1390	1280	1400	1350	1400	1312
29	3. m=7	1600	1600	1900	1600	1900	1675	1660	1600	1760	1900	1719.5

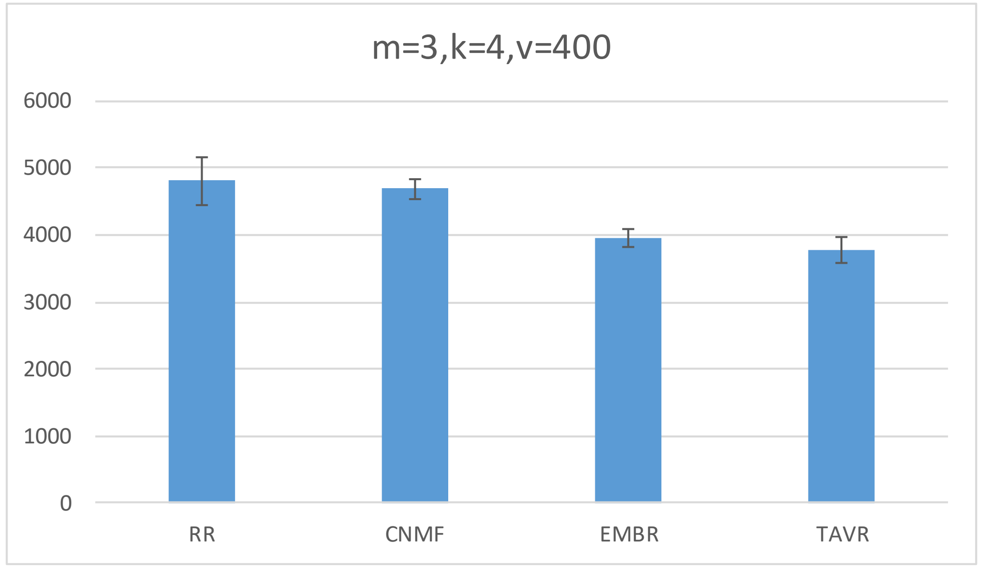
Comparative Evaluation

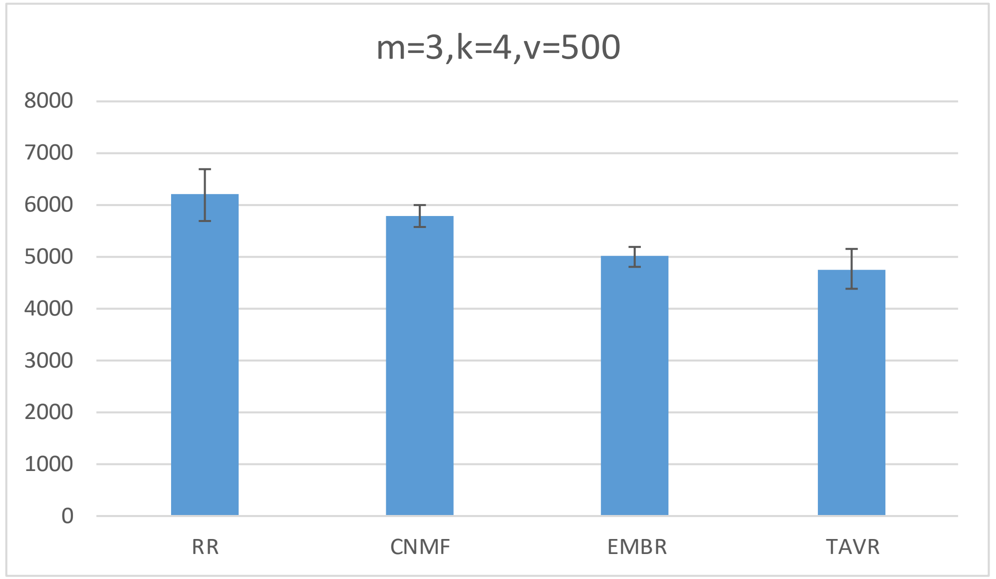
1. Performances of algorithms with the least values of m and k ; m=3 and k=4





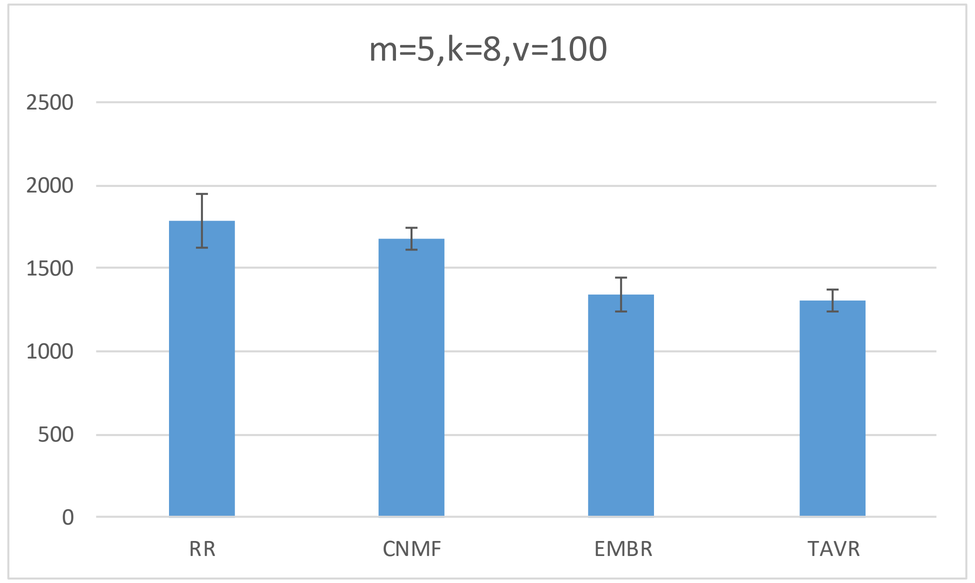


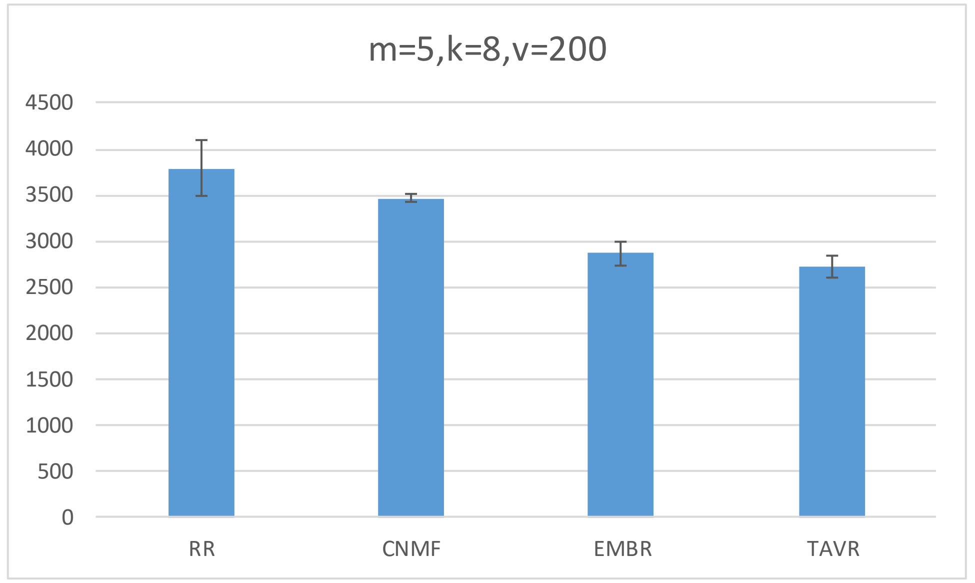


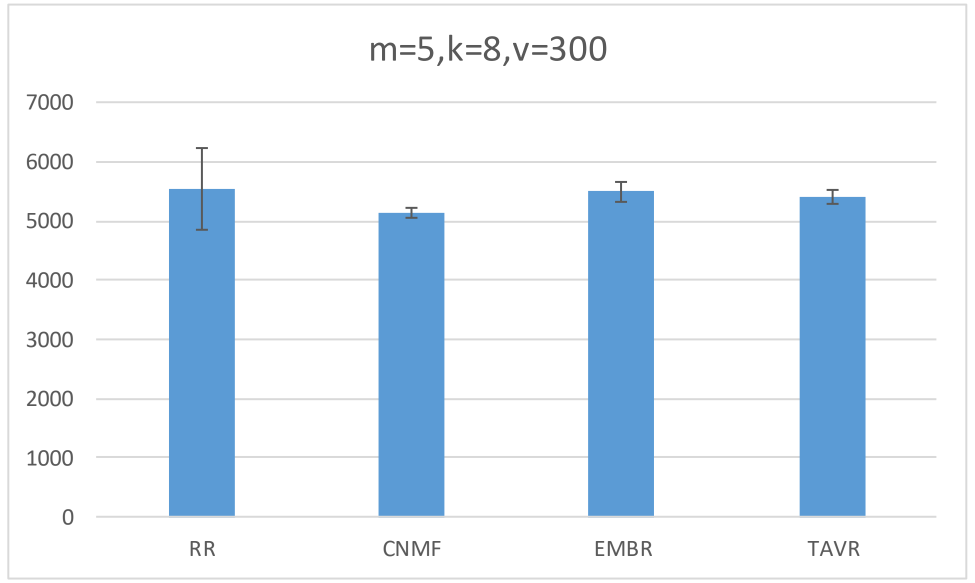


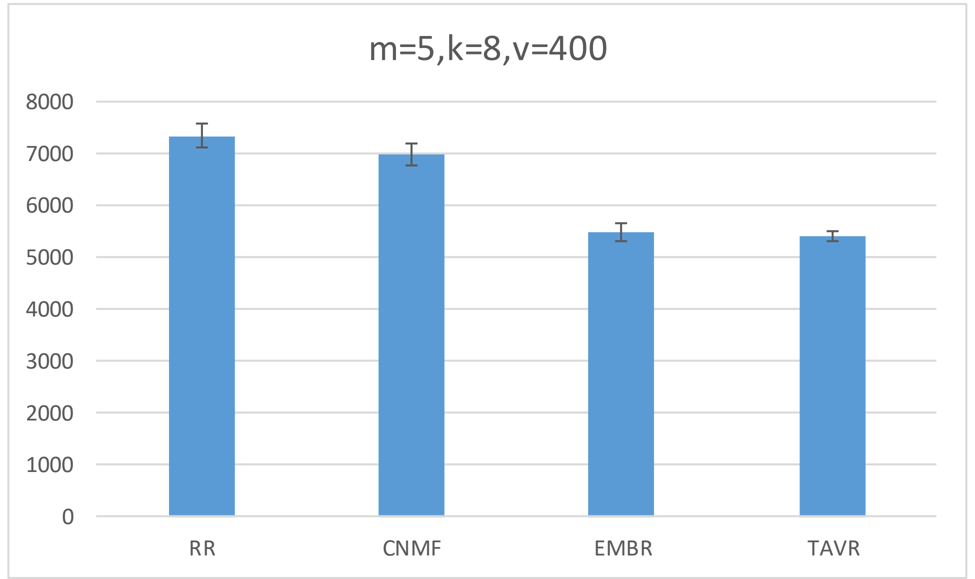
Comparative Evaluation (Contd.)

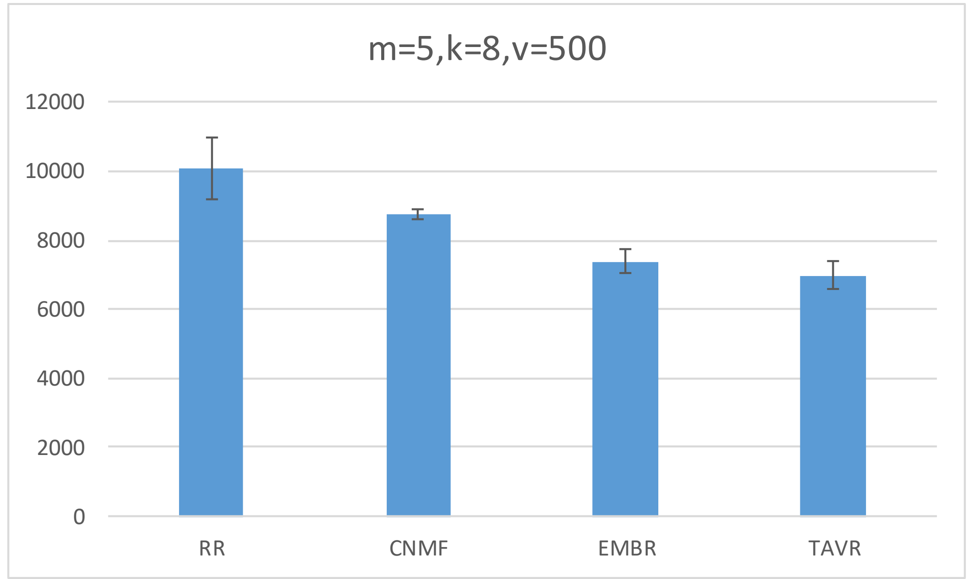
2. Performances of algorithms with an average values of m and k ; m=5 and k=8.





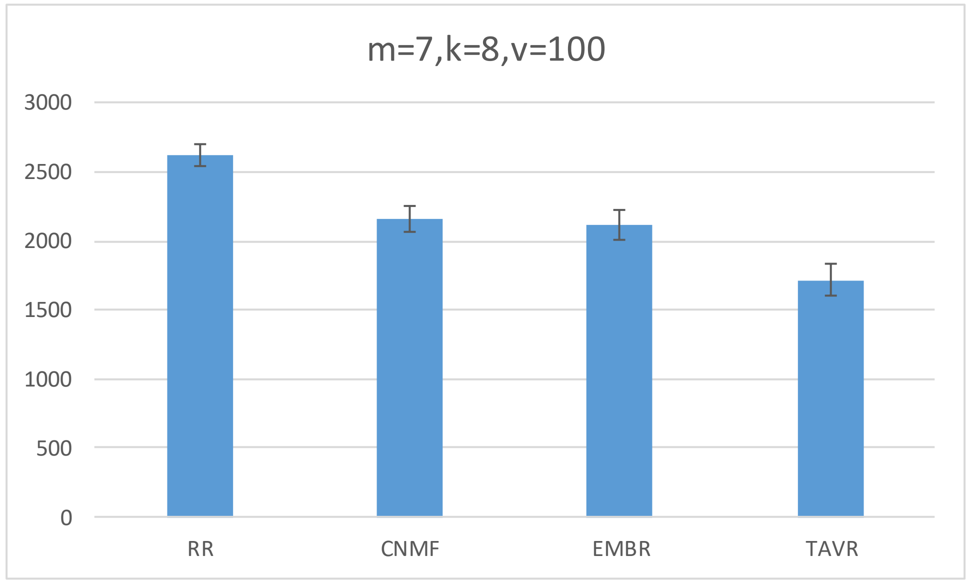


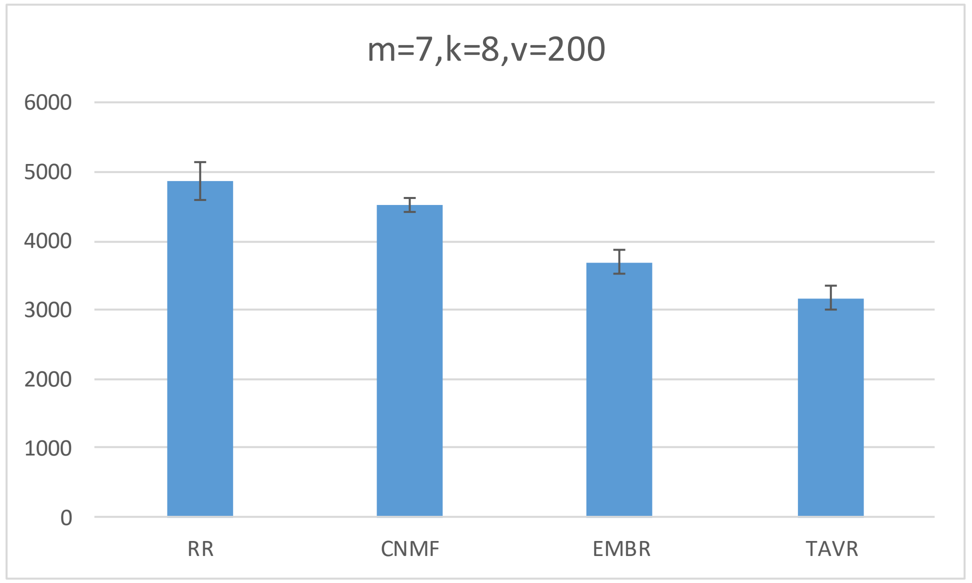


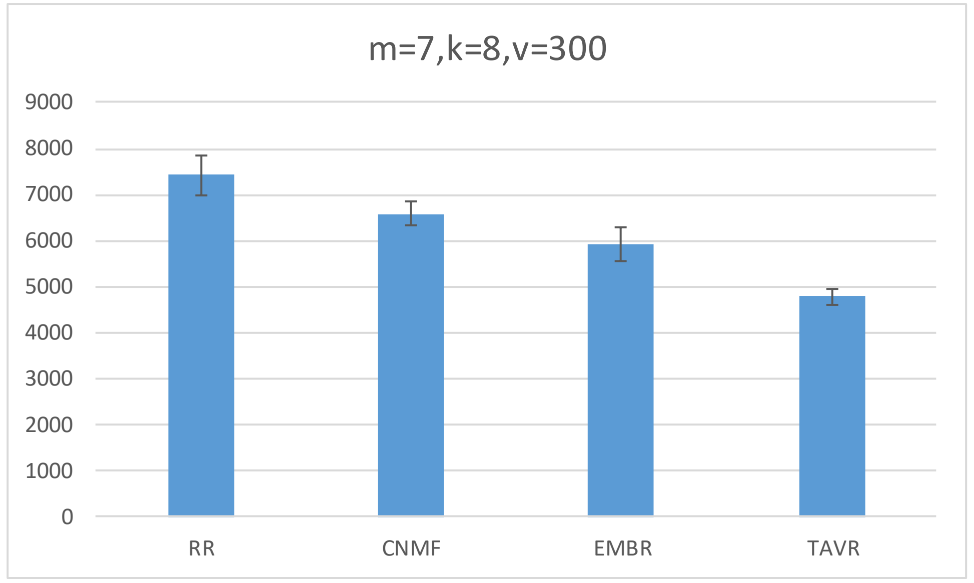


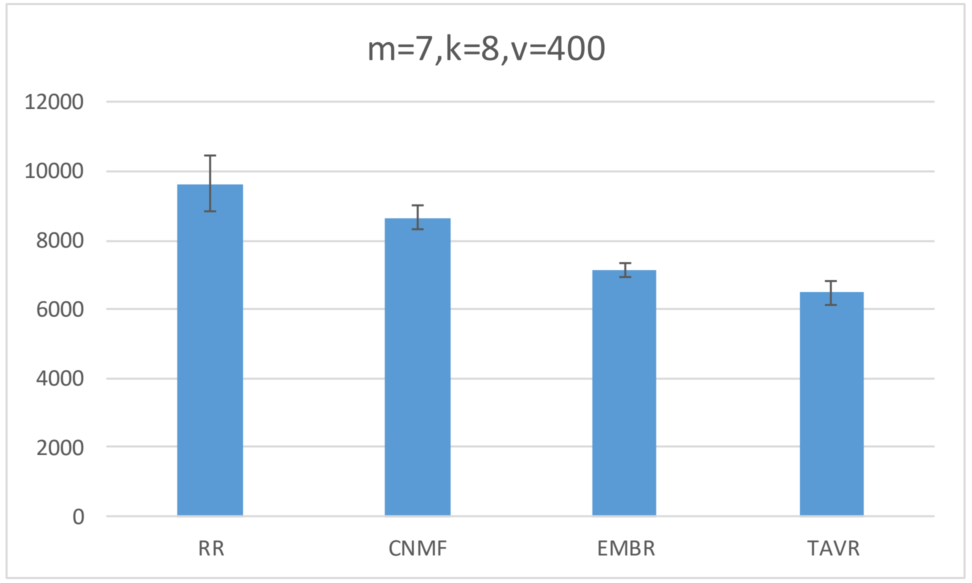
Comparative Evaluation (Contd.)

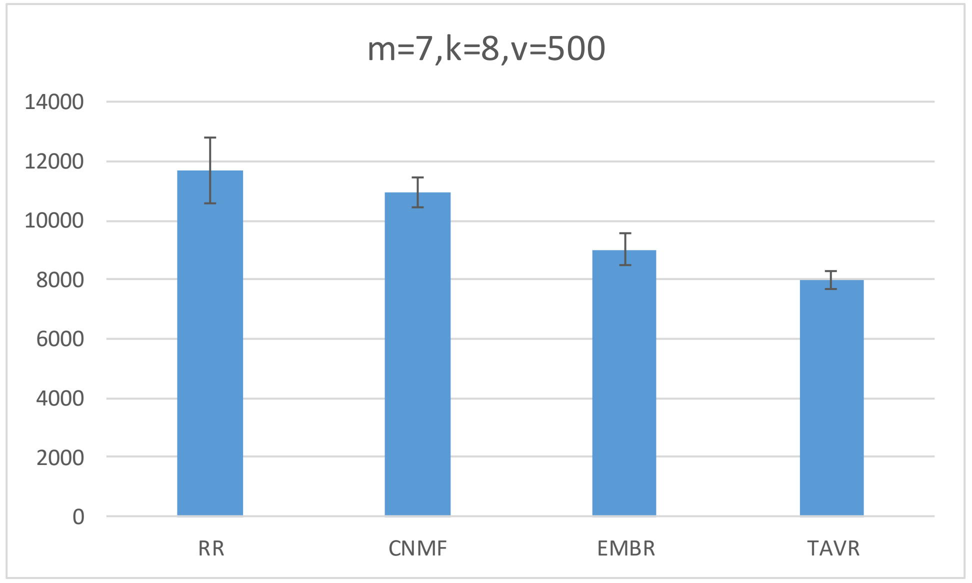
3. Performances of algorithms with large values of m and k ; m=7 and k=8.



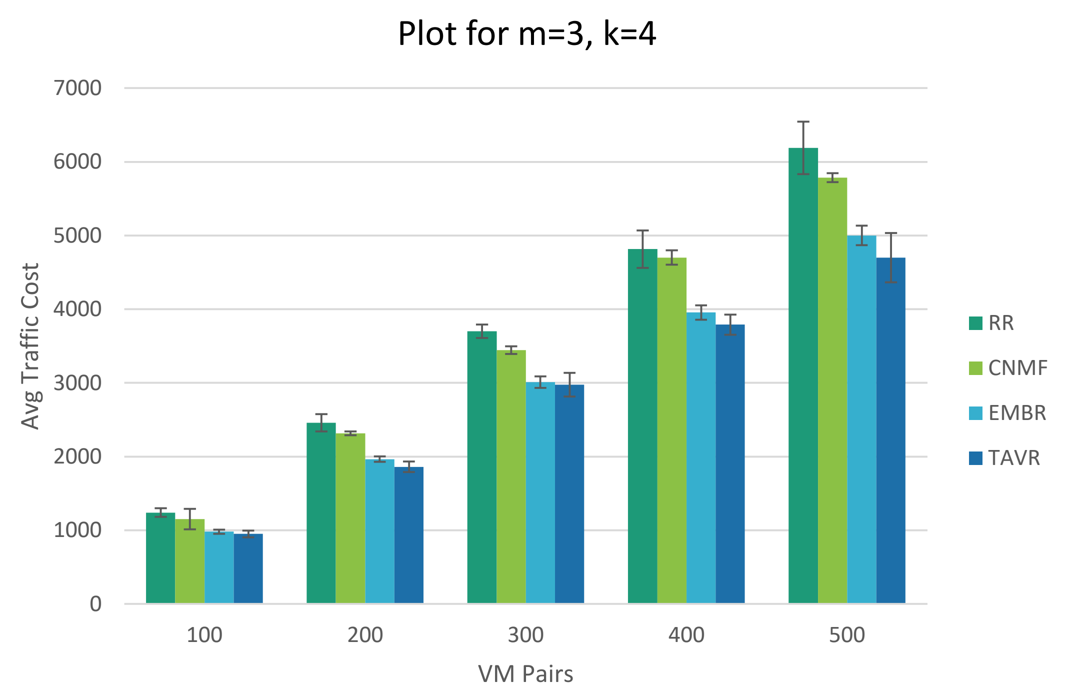


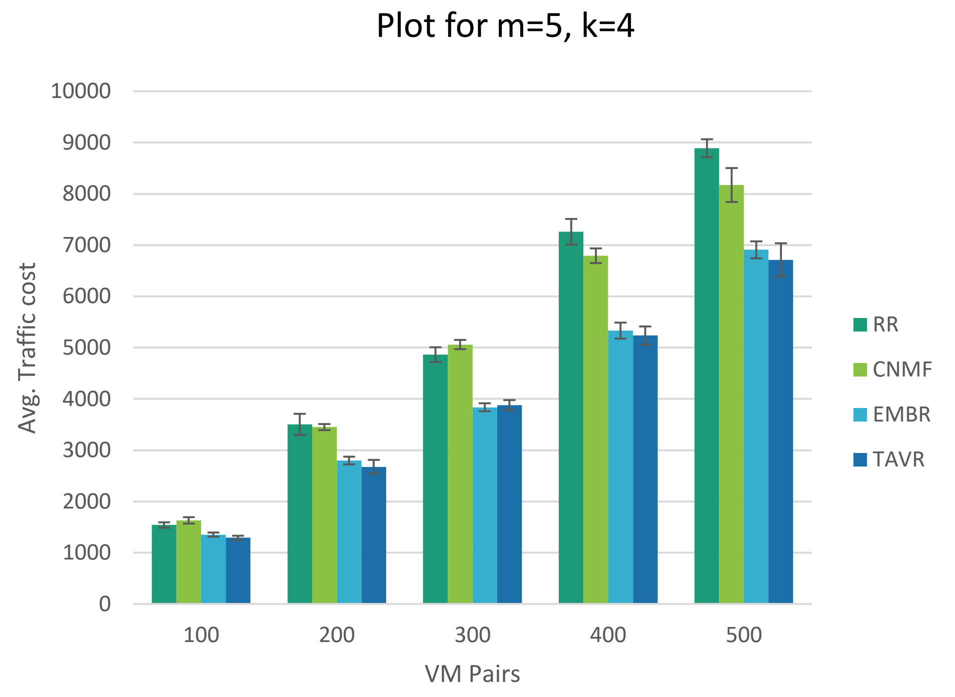


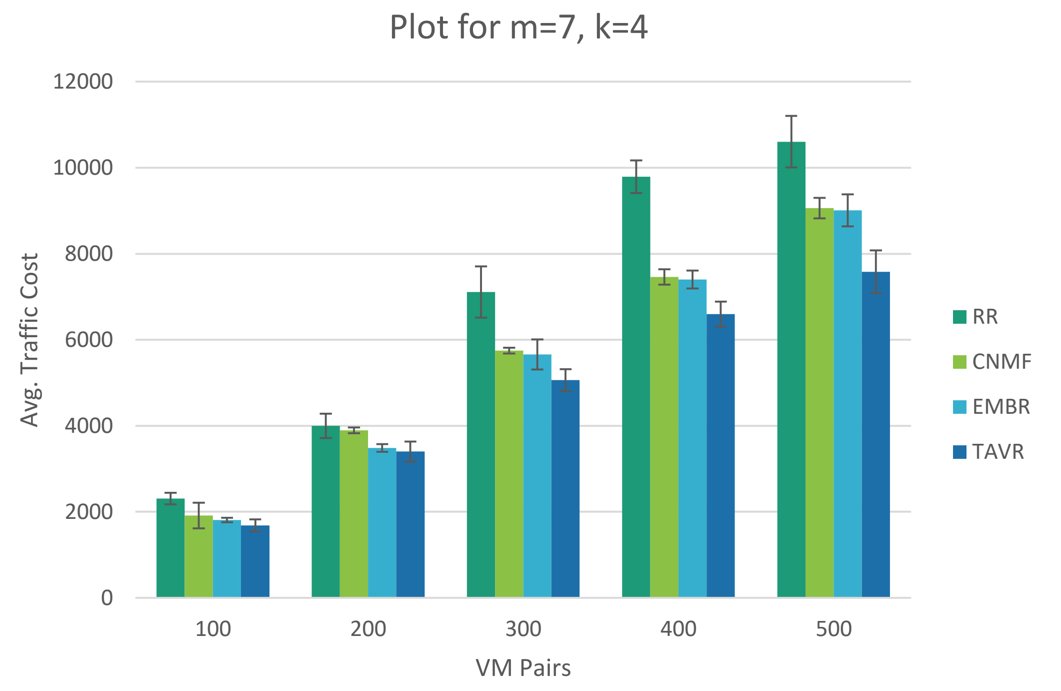




Plots for K=4





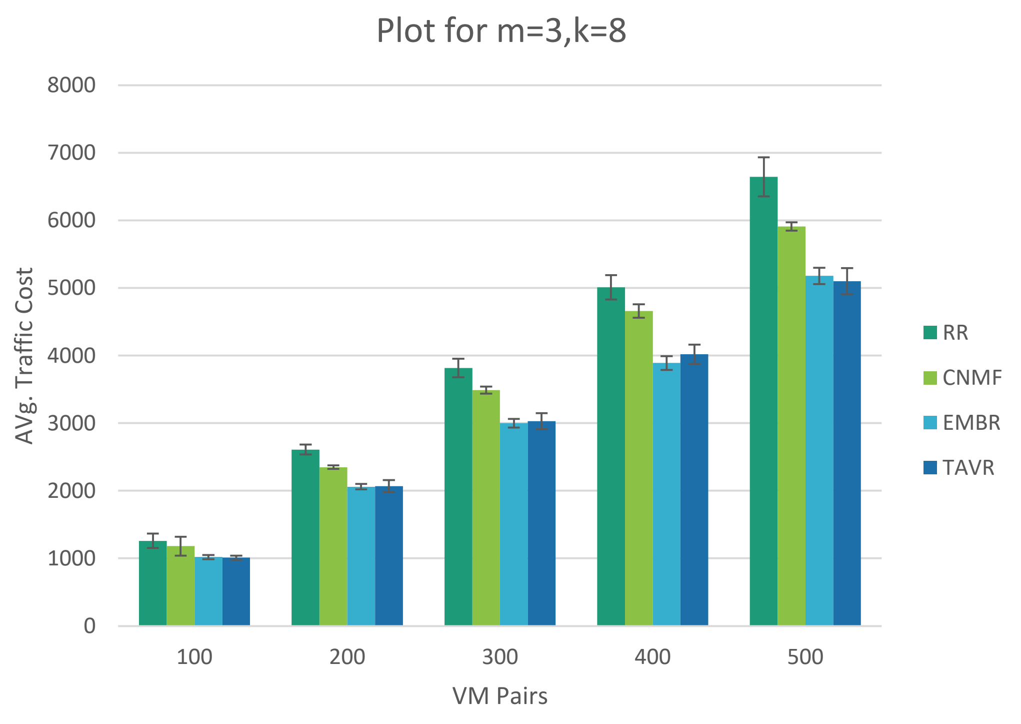


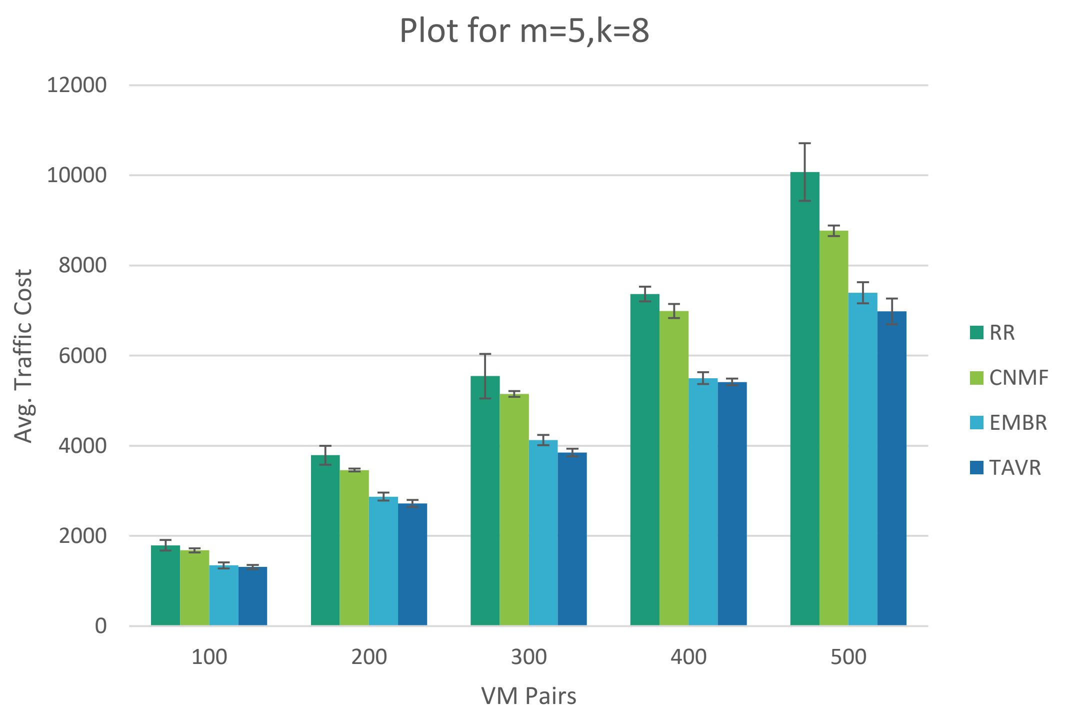
Computations from Simulation Parameters:

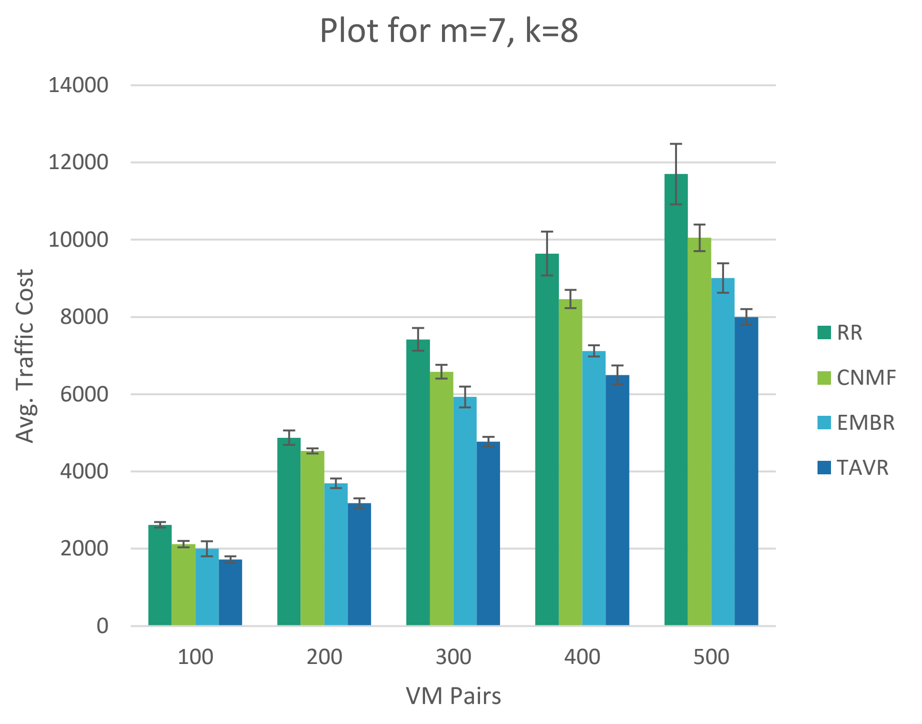
- K: 4
- Number of switches: $5k^2/4 = 5*16/4 = 20$
- Number of servers: $k^3/4 = 4*4*4/4 = 16$
- Traffic Cost in the network: $C = \sum_{i=1}^{p} C_i = c(S(v_i), mb_{i,1}) + \sum_{j=1}^{m-1} c(mb_{i,j}, mb_{i,j+1}) + c(mb_{i,m}, S(v_i'))$, where

  p={100,200,300,400,500}, m={3,5,7}

Plots for K=8







Computations from Simulation Parameters:

- K: 8
- Number of switches: $5k^2/4 = 5*64/4 = 80$
- Number of servers: $k^3/4 = 8*8*8/4 \quad = 128$
- Traffic Cost in the network: $C = \sum_{i=1}^{p} C_i = c(S(v_i), mb_{i,1}) + \sum_{j=1}^{m-1} c(mb_{i,j}, mb_{i,j+1}) +$

  $c\,(mb_{i,m}, S(v_i'))$, where p={100,200,300,400,500}, m={3,5,7}

Analysis on Individual Performance

The “Random” one

Random Replication(RR) is useful in cases where only load balancing is important and over all traffic cost can be compromised, i.e., having replica copies just for the purpose of high availability. It doesn’t guarantee reduced traffic cost.

The “Quick” one

While Closest Next Middlebox First (CNMF) has a low convergence time and it does provide reasonable results, choosing the shortest path within the service chain doesn’t always yield the overall traffic cost.

The “Ideal” one

There is no chance Exhaustive MiddleBox Replication(EMBR) misses the best cost yielding service chain because all combinations are explored. The only drawback of this algorithm is convergence time. But the algorithm has to be done for only initial set up or during a change in the network. It doesn’t require to be run in an everyday basis.

The “Efficient” one

Traffic-Aware VNF replication algorithm is very efficient in scenarios where expected traffic flow among the VM pairs is already known. It outperforms EMBR by 12%-15% with an increase in m and p. If there are rare cases where no other distribution of VNFs can yield a cost lesser than or equal to the original cost, then TAVR doesn’t place any replica in the network and that is the only drawback with this algorithm.

Comparative Analysis Report

Attributes/Algorithms	RR	CNMF	EMBR	TAVR
1. Execution time (w.r.to K)	O (K^4).	O(K^6).	O(K^8).	O(K^4).
Advantages	1. Quick and easy way. 2. Load balancing achieved.	1. Reliable in cases where the closest VNFs serve as the best service chain	1. Ideal algorithm that doesn't miss the best cost for overall traffic flow cost as all combinations of VNFs are explored to form a service chain.	1.Best result yielding algorithm in typical networks were traffic flow is known already.
Disadvantages	1. Unreliable in terms of energy efficiency.	1. Not consistent results	1. Long convergence time.	1. Replicas cannot be placed if none of the possible replicas can yield a cost lesser than original traffic cost.
Performance	1. Average traffic cost keeps getting larger with increase in m, k and p.	1. Although it doesn't perform as good as EMBR/TAVR, with the increase in m and k, it performs better and closer to EMBR because with more middlebox types and switches that hold these middleboxes, the shortest path is more often the best path.	1. EMBR performs the best among all algorithms. It performs slightly lower than TAVR in few cases as EMBR can have replicas of service chain which may produce a cost greater than original service chain. Also, for every replica, it is checked if it is optimal for all VM pairs. EMBR provides consistent results with increase in m, k and p values.	1. TAVR performs close to EMBR or at times better, as TAVR places replicas which always yield traffic cost lesser than original service chain's traffic cost. Also, every replica has to be evaluated only for the traffic group of VM pairs it belongs to. So, with increase in m, k an p, TAVR yields the best result.

Future Work

- All algorithms discussed in this project primarily works for fat-tree network. The authors of [6] have worked on core mobile network. There are other widely used Data Center topologies like DCell, Leaf-Spine, Butterfly, Jellyfish etc. These algorithms can be improved to make them more generalized.
- Also, the four algorithms that were implemented, only performed VNF replication for service chain scenarios. As discussed in design and analysis, there could be cases where the middleboxes do not have to be visited in a particular order.
- Non-Sequential Middlebox Replication Algorithm is implemented although out of scope of service chain scenario but it is not extensively tested for efficiency unlike other service-chain algorithms. Also, instead of basing the non-sequential replication on node preference, there can be better solutions as well.
- The algorithms discussed here also do not include scenarios where different communicating VM pairs have different service chains of different lengths. If the VNFs are not combined as service chains, then a middlebox prioritization scheme is to be used to prioritize middlebox instances based on their demand on the network and the replication must be done accordingly.
- As we keep bringing in different dimensions like the above mentioned to the replication problem, there is indeed a vast scope of extension and improvement to this project.

Conclusion

- With the ever-growing Data Centers, it is very important to maintain the quality of service and reduce the operational and network cost as well.
- The algorithms developed and tested during this project are highly efficient in ensuring minimum cost flow in Data Center Networks in which the traffic flow of any traffic type between the communicating VM pairs must be processed by several network functions.
- Pertaining to reduced overall cost, Traffic-Aware VNF replication outperforms others in a network in which expected traffic flow is given.
- Exhaustive Middlebox Replication algorithm is more generalized and an ideal solution to achieve the optimal average traffic cost in the network.
- EMBR and TAVR perform very closely in most cases but with increase in number of middlebox types and number of communicating VM pairs, TAVR outperforms EMBR by 12%-15%.
- These algorithms when implemented with other proposed solutions in future research directions, add more value to the future of Network Function Virtualization coupled with Software Defined Networking.

References

[1] Sevil Mehraghdam, Matthias Keller, Holger Karl, “Specifying and Placing Chains of Virtual Network Functions”, IEEE 3rd International Conference on Cloud Networking (CloudNet), 2014.

[2] Francisco Carpio and Jukan, “Balancing the Migration of Virtual Network Functions with Replications in Data Centers”, arXiv:1705.05573v1 [cs.NI], 16 May 2017.

[3] Rami Cohen, “Near Optimal Placement of Virtual Network Functions”, IEEE Conference on Computer Communications (INFOCOM), 2015.

[4] Francisco Carpio, Samia Dhahri and Admela Jukan, “VNF Placement with Replication for Load Balancing in NFV Networks”, arXiv:1610.08266v1 [cs.NI], 26 October 2017.

[5] Pham, Nguyen H. Tran, Shaolei Ren, Walid Saad, Choong Seon Hong, “Traffic-aware and Energy-efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach”, IEEE Transactions on Services Computing, 2017.

[6] Francisco Carpio, Wolgang Bziuk and Admela Jukan, “Replication of Virtual Network Functions: Optimizing Link Utilization and Resource Costs”, arXiv:1702.07151v1 [cs.NI] 23 Feb 2017.

[7] http://www.tomsitpro.com/articles/nfv-network-functions-virtualization-telecom,1-1756.html

[8] https://www.sdxcentral.com/nfv/definitions/nfv-elements-overview/

[9] https://community.fs.com/blog/sdn-nfv-the-future-of-network.html

[10] *yourdailytech.com*

[11] Brian Lebiednik, Aman Mangal, Niharika Tiwari, ” A Survey and Evaluation of Data Center Network Topologies”, arXiv:1605.01701v1 [cs.DC] 5 May 2016.

Acknowledgements

I would first like to thank my project advisor Dr. Bin Tang for sharing his ideas about such an interesting topic involving cutting-edge technologies like software defined networking, virtualization etc. and also for his very helpful feedback throughout all phases of the project. He was always ready to meet in person and clarify my doubts when needed and encouraged me to work harder to achieve better results.

I would like to gratefully and sincerely thank Dr. Mohsen Beheshti, Professor, Department Chair of Computer Science; I must express my very profound gratitude for his wonderful support and encouragement.

I would also like to thank my committee member, Dr. Jianchao 'Jack' Han, professor of Computer Science, for all of his guidance and valuable advising through my studying years.

I would like to thank all the faculty and my fellow students of Department of Computer Science for their direct and indirect support and for their significant contribution for my academic growth and excellence.

I would like to thank CAHSI for sponsoring to present this project in HENAAC Poster Competition which helped me win first place in such an esteemed event.

Finally, and most importantly, I would like to thank my husband and my son for their unfailing support, understanding and patience during the past two years.