Evaluation study of virtual network embedding for short-lived virtual networks

A thesis

presented to

the Faculty of the Department of Engineering Technology

University of Houston

In partial fulfillment

of the Requirements for the Degree

Master of Science

in Engineering Technology

By

Mydhili Palagummi

December 2015

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Dr. Ricardo Lent for the continuous support of my study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank my thesis committee: Dr. Mequanint Moges and Dr. Xiaojing Yuan, for reviewing my work.

Finally, I would like to thank my husband for his immense support throughout my degree program.

Abstract

Cloud services are currently being extensively used for server hosting, data storage, scientific and research purposes. Virtualization technology is an essential element for these services. Virtualization enables the creation of multiple virtual instances on physical or hardware infrastructure. Network virtualization is a recent advancement in this field through which virtual networks can be created over real physical networks (also called as substrate networks). Such virtual networks facilitate testing and quick deployment of new technologies, better utilization of hardware and provide more flexibility to users.

A crucial element of network virtualization is the stage in which the virtual networks are created on the substrate network. This process is of critical nature as the number of virtual networks that are created on the substrate are high. Hence the placement of these virtual networks needs to be done in a strategic way. The creation of virtual networks on a substrate network is referred to as Virtual network embedding. Determining the best way to place or create multiple virtual networks on a substrate network while satisfying a given set of constraints is referred to as Virtual network embedding problem. The technique or algorithms used to solve this problem are known as virtual network embedding algorithms.

In this thesis we evaluate and compare six virtual network embedding algorithms for embedding short-lived virtual networks on substrate networks with fat-tree topology and UUNET topology. We discuss different metrics to evaluate the performance of embedding algorithms and compare the algorithms based on these metrics. In particular, we examine the probability of success of embedding a virtual network, average

substrate path length and the distribution pattern of virtual networks in the substrate network for six different algorithms. The aim of this thesis work is to compare the performance of virtual network embedding algorithms, observe the nuances of the approaches that contribute to optimal results and investigate the embedding for the case of short-lived virtual networks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Virtualization

Virtualization is a unique technology that contributed to a vast growth of cloud computing and helped in reducing costs and increasing hardware utilization, flexibility and scalability. A recent development in this area is Network virtualization where an entire network is virtualized by virtualizing every element in the network. Similar to storage virtualization and desktop virtualization, network virtualization gives an illusion of a real network to the user. The benefits that network virtualization offers are extremely valuable to the research community. For instance, experimenting new technologies, protocols or architectures will require huge amount of physical resources which is economically unreasonable and sometimes may not be feasible at all due to inflexibility of the existing infrastructure [2]. But by using virtualized resources, it is easy to test new technologies. The deployment of new technology or software is also very quick. Moreover, hardware utilization can be improved to a great extent by sharing the resources among many users. The isolation provided by virtualization ensures security and provides flexibility to the user to customize the virtual network. Technically, virtualization is defined as the process of creating virtual entities on physical entities. Popular version of virtualization are desktop virtualization, server virtualization, storage virtualization and network virtualization.

1.2 Motivation

One of the major challenge in realizing virtualized infrastructure is to embed the virtual networks in an efficient manner so that the substrate network is utilized effectively and also maximum number of virtual networks are accommodated in the substrate network. Also, depending on the specific application, the virtual network embedding may also have other objectives such as minimizing energy usage. And in other cases, there may be additional constraints such as additional security for certain virtual networks[12]. All these constraints and specific objectives make the virtual network embedding problem complex and difficult to solve. Many solutions have been proposed to a simplified version of the virtual network embedding problem. In this work, we have considered six different solutions which constitute simple approaches like randomly embedding nodes to solutions, greedy approaches and approaches that utilize topological characteristics to obtain an optimal embedding. Commonly, in a cloud computing scenario, the users may choose to operate their virtual networks only for a limited amount of time. In such a case, the virtual networks would be short-lived and there would be continuous arrivals and departures of virtual networks in the substrate network. We presume that there would be multiple gaps in the substrate network similar to fragmentation when the virtual network depart from the substrate network. To understand the performance of virtual network embedding algorithms in such a scenario, a simulation was carried out to study and evaluate the virtual network embedding algorithms for embedding short-lived virtual networks.

1.3 Contributions

The contributions of this thesis work are

- A brief overview of various approaches from the literature to solve virtual network embedding problem is presented
- Simulation and performance analysis of virtual network embedding algorithms

is conducted for short-lived virtual networks

- Virtual network embedding is simulated and analyzed for different levels of probability of departure of virtual networks
- Virtual network embedding is simulated and analyzed for two different substrate networks (tree based topology (Fat-tree) and a flat topology (UUNET))
- Various metrics for analyzing performance of virtual network embedding are discussed

1.4 Thesis organization

- In chapter 2, we lay the foundations for this thesis work by describing a generic business model, key features of virtual network embedding, objectives and constraints involved in virtual network embedding and a mathematical description of the virtual network embedding problem.
- Chapter 3 provides an outline of various categories of virtual network embedding approaches from the literature proposed in the recent years.
- Chapter 4 provides a synopsis of the six embedding algorithms that were analyzed in this work. The approach and strategy of each algorithm is discussed.
- Chapter 5 describes the experimental setup in this work. Specifically, the substrate networks and virtual networks used in the simulation are presented. The arrival and departure pattern of the virtual networks is described. The details of the simulation and the performance metrics used to evaluate the embedding are also provided in this chapter.
- Chapter 6 presents the results of the simulation and a few observations.
- Chapter 7 concludes the thesis and discusses possible future work in this area.

Chapter 2

Background

This chapter sets the stage for the discussion of virtual network embedding in this thesis. First, the business model is explained and then the detailed process of virtual network embedding is discussed. Later the objectives of virtual network embedding, steps involved in virtual network embedding and the mathematical representation of virtual network embedding are presented.

2.1 Business model

Nowadays, many cloud providers are providing Infrastructure-as-a-service for their customers. Fischer et al presented a futuristic Internet business model [9] where the business and network management roles are separated for effective operation of the service. For this work, we describe a similar simplified model to explain the virtual network embedding scenario. Figure 2.1 shows a simple business model with an infrastructure operator, virtual network operator and service provider. The infrastructure operator manages the network equipment and ensures proper operation of the substrate network. The service provider caters to the user's virtual network requests and is responsible for the interaction with the user. The service provider sends the virtual network requests to the virtual network operator who performs the virtual network embedding. The virtual network operator is responsible for creation and operation of virtual networks on the substrate network.

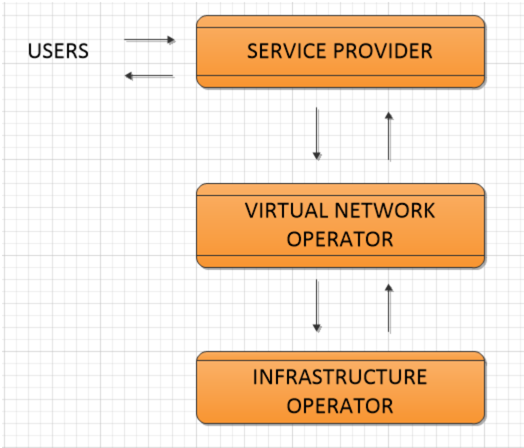


Figure 2.1: Business model

2.2 Virtual network embedding

Virtual Network Embedding(VNE) refers to the process of attending to virtual network requests and creating corresponding virtual networks on the substrate network. Virtual network requests are presented by the customer or users who want to use a virtual network on the operator's physical infrastructure. The virtual network requests consist CPU capacity demands for network nodes and bandwidth capacity demands for network links which represent the QoS constraints. Additional information like duration of hosting the virtual network, topology of the virtual network, priority or security constraints may also be provided. The virtual network operator is responsible for embedding the virtual network based on the request. Embedding involves creation of virtual network over the substrate network. Virtual network creation is achieved by creating virtual instances of network nodes and network links. Virtual nodes are realized by creating virtual machines on substrate nodes and virtual links are realized by creating virtual network connections between virtual nodes over substrate paths. VNE can be a success or a failure depending on the virtual network capacity demands and available resources on the substrate network. It is usually a two step process involving virtual node embedding and virtual link embedding. The virtual node embedding involves the creation of virtual nodes (virtual machine instances) on the substrate nodes and virtual link embedding involves determination of best possible routes (virtual links) between the substrate machines that host the

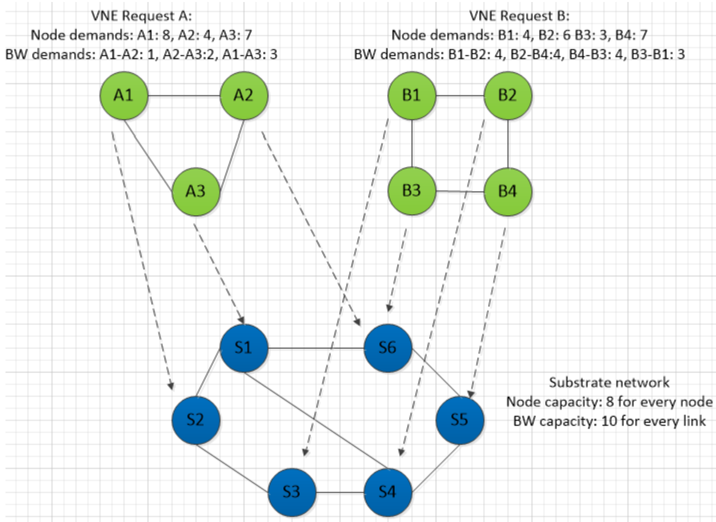


Figure 2.2: VNE: Embedding multiple virtual networks on substrate network

virtual nodes. Both these steps take the virtual node CPU capacity demands and virtual link bandwidth capacity demands into account. In essence, the virtual network operator is mapping each virtual node to a substrate node and each virtual link to a substrate path. Figure 2.2 shows an example of the embedding of two virtual networks on a substrate network. The virtual network request specifies the node and link demands of each virtual node and virtual link in the virtual network.

When the service provider receives huge amount of virtual network requests, the virtual network operator has to embed the virtual networks in an efficient way so that more number of virtual networks can be accommodated and the substrate also is efficiently utilized. For achieving such an efficient embedding, an algorithm is required that determines the best possible mapping for the virtual network request. This algorithm is usually referred to as virtual network embedding algorithm. The core of the solution to the virtual network embedding problem is implemented through the VNE algorithm. When the virtual network request is provided as input, the algorithm provides the best possible mapping based on the information about the substrate network. The virtual network embedding will result in a success if the substrate network has enough node capacities and link capacities to accommodate the virtual nodes and virtual links respectively. If the substrate network cannot satisfy either the virtual node capacity demand or the virtual link capacity demand, the embedding fails.

2.3 VNE objectives, parameters, methods

There are three main aspects for any virtual network embedding approach. The objective, optimization parameter and the method of embedding. There can be many ways to optimize the VNE process and that depends on the objective of the VNE which is decided by the virtual network operator. In general, the main objectives of embedding would be to maximize the revenue for the infrastructure operator and to maximize the number of virtual networks that can be accommodated. There are other specific cases where the objective is to optimize an additional feature. For instance, the objective may include the minimization of the amount of energy consumed by a virtual network.[16] In some cases, the virtual network operator may decide to perform VNE with an objective to provide additional security.[12] Embedding the networks in a way that ensures survivability and reliability are other examples of VNE objectives.[15] These specific objectives make the VNE problem more complex.

The optimization parameters are specific properties that are used to decide the best VNE mapping which makes the embedding more efficient. Many of the previous approaches use topological attributes like node centrality or node degree of the virtual nodes or substrate nodes in the embedding process. The topology design of the virtual network, estimated revenue, virtual network request priority are some other parameters that are used to optimize the embedding.

The methods of VNE can follow three possible approaches which provide heuristic, meta-heuristic and exact solutions to the VNE problem. Also, as indicated by Fischer et al in [9], the VNE methods can be coordinated or un-coordinated. In the coordinated method, the virtual node and virtual link embedding are performed at the same time. Whereas, in the uncoordinated method, the virtual node embedding and the virtual link embedding are performed separately one after the other. Various algorithms that have used different objectives, optimization parameters and methods are discussed in chapter 3.

2.4 Virtual network request

The virtual network request consists of the topology of the desired network, CPU capacity demands of the nodes in the network, bandwidth demands of the links in the network, priority preference of the network, security preference for the network and other constraints like geographical location. The virtual network operator has to satisfy all the constraints specified in the request. If the substrate network resources are insufficient and cannot satisfy the constraints in the request, the embedding fails and the virtual network request is rejected. If the substrate network has sufficient resources, the virtual network is embedded in the substrate network and can be used by the user.

2.5 Virtual node embedding

Virtual node embedding is the process of creating a virtual machine on a substrate node. This is the first step in the VNE process. To embed a virtual node, a candidate substrate node should be selected. Once a substrate node is selected, virtual machine can be created on that node. The method of the selection of candidate substrate node plays a pivotal role in the embedding process. Virtual node embedding is a many-to-one mapping of virtual nodes and substrate nodes. A virtual node can be mapped only on one substrate node. But multiple virtual nodes can be mapped on a single substrate node if the substrate node has sufficient resources.

2.6 Virtual link embedding

Virtual link embedding is the process of creating virtual links between the virtual nodes which are embedded in the virtual node embedding stage. Every virtual link has two virtual nodes connected to it. To embed a virtual link, the substrate nodes hosting these two virtual nodes are identified. Then, the best path between the two substrate nodes is identified and a virtual connection is made between these two substrate nodes. Usually, the best path is chosen based on shortest path algorithms. Virtual link embedding is a many-to-many mapping between virtual links and substrate links. A virtual link may be embedded on a single substrate link or multiple substrate links that form a path. And many virtual links can be embedded on a single substrate link. The substrate link should have sufficient bandwidth to satisfy the virtual link bandwidth demand. When a virtual link is embedded onto a substrate path(collection of connected substrate links), every link in the substrate path should satisfy the bandwidth demand of the virtual link. The substrate path is selected based on shortest path algorithms like Dijkstra's shortest path algorithm or Yen's K-shortest path algorithm.

2.7 Virtual network embedding problem

For a given substrate network and a set of virtual network requests, there will be many possible ways to map the virtual networks onto the substrate. The determination of the best possible mapping between the virtual network and substrate network is referred to as the virtual network embedding problem. Multiple objectives and constraints of VNE make it a complex problem to solve. This section presents the mathematical representation of virtual network embedding problem similar to the many VNE problem formulations in the literature.

The substrate or physical network can be represented by a weighted graph $G_p = (N_p, L_p)$ where N_p denotes the set of physical nodes and L_p denotes set of physical links. The virtual network is represented as $G_v = (N_v, L_v)$ where N_v denotes the set of virtual nodes and L_v denotes the set of virtual links in the virtual network. The mathematical notation of various other network parameters are given in Table 1.

In the process of virtual network embedding, the virtual nodes and links are mapped to substrate nodes and links. The virtual network requests have capacity

n_p	physical node
n_v	virtual node
l_p	physical link
l_v	virtual link
$c(n_p)$	CPU capacity of physical node
$c(n_v)$	CPU demand of virtual node
$bw(l_p)$	Bandwidth of physical link
$bw(l_v)$	Bandwidth demand of virtual link
$rev(G_v)$	Revenue generated by embedding virtual network G_v
$cost(G_v)$	Cost of embedding virtual network G_v
VN_s	Number of successfully embedded virtual networks
VNR	Number of virtual network requests
$P(G_v)$	set of physical paths allocated for virtual links of network G_v
$p(l_v, G_v)$	physical path allocated for the virtual link l_v of the network G_v
$hops(p(l_v, G_v)$	number of hops in a physical path p

Table 2.1: Network parameter notations

constraints that need to be satisfied to ensure quality of service. The virtual network embedding is said to be successful if the substrate network meets the CPU demand($c(n_v)$) and bandwidth demand($bw(l_v)$)) of the virtual network request.

The goal of virtual network embedding algorithms is to find the best mapping between the substrate and virtual network. The embedding can have multiple objectives such as maximizing revenue to cost ratio, maximizing acceptance ratio, minimizing energy usage, ensuring reliability and providing security[9]. In this work, maximizing revenue-to-cost ratio, acceptance ratio and providing quality of service are considered as the objectives for virtual network embedding. Quality of service is guaranteed by satisfying the capacity constraints of the virtual network requests.

Acceptance ratio is defined as the ratio of the number of successfully embedded virtual networks and number of virtual network requests. Mathematically, acceptance ratio is defined as

$$AR = VN_s/VNR$$

Revenue generated by embedding a virtual network is expressed in terms of the CPU capacity and bandwidth used by the virtual network. Hence, Revenue generated by a virtual network G_v is defined as

$$Rev = \sum_{n_v \epsilon N_v} c(n_v) + \sum_{l_v \epsilon L_v} bw(l_v)$$

Cost incurred for embedding a virtual network is expressed in terms of the CPU capacity and bandwidth allocated to the virtual network. Cost of embedding a virtual network G_v is defined as

$$Cost = \sum_{n_v \epsilon N_v} c(n_v) + \sum_{l_v \epsilon L_v)} hops(p(l_v, G_v)) * bw(l_v)$$

Chapter 3

Literature review

Many Virtual network embedding (VNE) algorithms with different objectives and approaches have been proposed in the recent years. Some of the algorithms that can be found in the literature are discussed in this chapter. The process of virtual network embedding has many variations. The embedding can be static or dynamic depending on the arrival of the virtual network requests. The node and link mapping can be done together or separately. An elaborate classification of various embedding algorithms is discussed in the survey by Fischer et al in [9]

3.1 Node ranking based methods

Node ranking is a popular technique used for VNE in many of the approaches. Node ranking may be based on topological attributes or network resources or both. Wang et al proposed an embedding algorithm [18] based on topological information and network resources. In this approach, a tree is constructed from the virtual network and it determines the order of embedding the virtual nodes. The highest resource demands are embedded first and the neighbors of a virtual node are embedded successively. The candidate substrate node is chosen based on its resources and distance to the substrate node that hosts the parent of the virtual node. The link embedding is done using the k-shortest path algorithm. The mapping tree algorithm focuses on embedding virtual nodes close to each other. Another such approach was proposed by Cui et al in [5].

The virtual nodes in each virtual network are ranked in decreasing order of resource demand. The candidate substrate node is selected based on convergence degree which is calculated using its resources and distance to the substrate nodes that host the virtual node's neighbors. For link embedding, multi-commodity flow algorithm or k-shortest path algorithm is used. Gong et al also proposed a similar approach[10] that performs the embedding based on a unique parameter called Global resource capacity (GRC). GRC of a node is computed using the resources of the node and the adjacent nodes and also the global resource capacity of the adjacent nodes. After that, a greedy algorithm based on GRC performs node mapping. Link mapping is done using Dijkstra's shortest path algorithm. Xiang Cheng et al proposed a VNE algorithm [3] in which the nodes are ranked based on the CPU capacity of the node, sum of link bandwidth of all outgoing links and the ranks of the reachable nodes. The nodes are mapped in a greedy manner based on ranks and the links are mapped using the k-shortest path algorithm.

3.2 Topological information based methods

Another common approach is to use the topological attributes of the virtual and/or the substrate network for a better embedding. Wang et al proposed an embedded algorithm based on closeness centrality [19]. This approach ranks the nodes based on closeness centrality. Closeness centrality is a characteristic of a node that represents the closeness of this node to all the other nodes. The virtual node with the maximum closeness centrality is mapped on the substrate node with the highest closeness centrality. Tao et al proposed a topology-cognitive algorithm in [17]. They formulated multiple algorithms and selected one of them based on the topology of the virtual network request. These algorithms were optimized for specific topologies. After the node mapping, link mapping is performed using a k-shortest path algorithm. A topology -aware approach based on degree of nodes was proposed by Feng et al in [8]. The virtual and substrate nodes are ranked based on node resources, link resources and the degree of the node. During the embedding, the virtual node with the highest rank is embedded onto the substrate node with the highest rank that satisfies the resource demands. An extension of this approach is proposed in [7]. Three topology-aware embedding algorithms based on seven topological characteristics are presented. In all the three algorithms the nodes are ranked based on the topological attributes and the node with the highest rank is mapped onto the substrate node with the highest rank. The three algorithms are based on degree, degree and farness, degree, farness and betweenness respectively.

3.3 Methods involving topology changes

Some embedding algorithms modify the structure of the networks to produce better embedding results. Lan Li et al propose a VNE algorithm based on sub-graph construction in [11]. To reduce the VNR processing time and improve load balancing, a sub-graph of the substrate graph is created and the VNE is done on the sub-graph. All the links with low load status are added to the sub-graph. After the sub-graph is constructed, the virtual nodes are mapped to the substrate node with the maximum capacity and the links are mapped based on a multi-commodity flow computation. Xue et al propose an approach [22] where they modify the virtual network request by dividing it into parts and embedding the sub-requests individually. Moreover, the decision of the candidate substrate node is affected by the geographical location of the substrate node.

3.4 Linear and non-linear programming methods

Some of the approaches follow unique mathematical formulations to solve the VNE problem. Whenzhi Liu et al proposed a one-step approach [13] to perform VNE. They used non-linear programming techniques to solve the VNE problem. Melo M et al presented an approach [14] for VNE using integer linear programming. Various constraints are defined for the number of virtual nodes per substrate node, number of substrate nodes mapped to a virtual node, CPU capacity, maximum distance between two virtual nodes and link bandwidth. And the objective function was formulated to minimize the substrate resource consumption.

3.5 Other methods

There are a few other approaches that perform VNE based on other parameters like periodic resource demands, user priority and energy conservation.

Xu et al proposed embedding algorithms [21] for virtual networks with static and periodic demands. It is based on the principle that many enterprise services have periodic demands that can be exploited for efficient allocation of resources. Su et al propose a VNE algorithm [16]that reduces the energy costs. The proposed algorithm maps the virtual nodes onto substrate nodes with low electricity price and which are already powered up and are active. Thereby, keeping some of the substrate nodes powered off which saves energy.

Some approaches use a variant of link embedding by performing path splitting. When path splitting is done, the virtual link is split into multiple flows and each flow is embedded onto a different path in the substrate network. Minlan et al propose such an algorithm [23] that performs virtual node mapping first and then during the link embedding process, path splitting and path migration is done to accommodate more requests and reduce the rejection rate of VN requests.

In most of the approaches, node and link embedding is done separately. Separating the two sub problems may not be the optimal way to embed the virtual network because by mapping the nodes without considering the links will restrict the solution space for the link mappings. Cheng et al propose an algorithm [3] that performs node and link mapping together.

In some approaches, to reduce the rejection rate, the virtual networks are reconfigured to make room for new requests. In one such approach [6], whenever there is a rejection of a VNE request, the network is analyzed for any possible reconfiguration.

The most suitable node for migration is chosen as the star moving candidate. This star moving candidate is migrated to another substrate node while keeping the virtual links intact. The reconfiguration is performed due to the fragmentation of network resources which causes VNE rejections. This is one of the major areas of focus in this thesis work. Six virtual network embedding algorithms were analyzed to study the effects of node ranking, and topological information in virtual network embedding. Also, by studying short-lived virtual networks, the distribution of virtual nodes was analyzed.

Chapter 4

Virtual network embedding algorithms

This chapter discusses the representative set of virtual network embedding approaches from the literature that were chosen for this performance evaluation. First, a simple approach that embeds virtual nodes randomly on the substrate network is taken to analyze random embedding of virtual nodes. Greedy approaches was the next category that was analyzed. In this category, Least Loaded First (LLF) and Most Loaded First (MLF) approaches were studied. The other category of approaches that were studied are approaches that exploit topological properties and node ranking strategies to embed the virtual networks. One such approach was proposed by Wang et al. by using topological attributes such as closeness centrality [19]. The evaluation in their work indicates that topological information can be exploited to improve the embedding. Another approach [18] performs the embedding by mapping neighbors closely and ranking the substrate nodes based on their available capacities. These two algorithms were chosen to test the effectiveness of topological information and node proximity for improving VNE. The evaluation study in this work aims to compare the performance of these algorithms and also study the effect of using these techniques in a substrate network which supports short-lived VNEs.

4.1 Random algorithm

Random algorithm is an embedding technique that randomly chooses the candidate substrate nodes for embedding the virtual node. The first step in embedding is to generate a mapping between virtual nodes and substrate nodes. Next, the node constraints are verified to check whether the selected substrate nodes have sufficient capacity to support the virtual node. Finally, the shortest paths between the virtual nodes are identified and the bandwidth constraints are also verified. If the node capacity and link bandwidth constraints are satisfied, then there would be a successful embedding. In the case where either the node capacity constraints or the bandwidth constraints are not fulfilled, the embedding will fail and there would be a rejection of virtual network request.

4.2 LLF algorithm

Least loaded first algorithm is one of the greedy techniques used in this evaluation study. During the virtual node embedding, the algorithm needs to choose the candidate substrate node to embed a given virtual node. The technique used to choose the candidate substrate node is one of the distinguishing factors among different algorithms. LLF algorithm chooses the substrate node with the least load as the candidate substrate node for each virtual node. Load of a substrate node is the amount of node capacity being used by the already embedded virtual nodes on the substrate node. Once the candidate substrate node is selected, the bandwidth constraints of the virtual links adjoining the virtual node are verified. After the virtual nodes are mapped onto the substrate nodes, shortest paths between the embedded virtual nodes are determined by using Dijkstra's shortest path algorithm. The virtual links are then mapped to the shortest paths in the substrate network.

4.3 MLF algorithm

Most loaded first algorithm is another greedy approach which works exactly the opposite way compared to the least loaded first algorithm. MLF algorithm chooses the substrate node with the most load as the candidate substrate node for each virtual node. In other words, it chooses the substrate node with the least available capacity as the candidate substrate node for each virtual node. In the virtual link embedding, Dijkstra's algorithm is used to determine the shortest path between the virtual nodes. In both LLF and MLF algorithms, the number of hops was used as the distance metric to determine the shortest paths.

4.4 Closeness centrality algorithm

The closeness centrality algorithm was proposed by Wang et al in [19]. Closeness centrality algorithm is a virtual network embedding algorithm that uses a topological property called closeness centrality to determine the best embedding. Closeness centrality is a property of a node in a graph. It is the measure of closeness of a node to all the other nodes in the network. Mathematically, The closeness centrality of a node n_i is defined as

$$cc_{n_i} = \frac{1}{\sum_{j=1}^{n} d(n_i, n_j)}$$

where n is the total number of nodes in the network and $d(n_i, n_j)$ denotes the shortest distance between the nodes n_i and n_j. Virtual network embedding is achieved in two steps - virtual node embedding and virtual link embedding. This algorithm takes a node ranking approach in embedding the virtual nodes. The virtual nodes are ranked in decreasing order of closeness centrality and the embedding is performed based on the rank. Specifically, the virtual node with the highest centrality is mapped onto the substrate node with the highest centrality that can satisfy the virtual node demands. After the virtual node embedding, the virtual link embedding is done using k-shortest path algorithm.

4.5 Improved closeness centrality algorithm

The improved closeness centrality algorithm is an enhanced version of closeness centrality algorithm. In this algorithm the closeness centrality property is modified by including the node and link capacities. This modified property is referred to as the improved closeness centrality. Mathematically, the improved closeness centrality of a node n_i is defined as

$$icc_{n_i} = \sum_{j=1}^{n} c(n_j) * e^{-(\frac{d(n_i,n_j)}{minBW(n_i,n_j)})^2} \tag{4.1}$$

where $c(n_j)$ is the capacity of the node n_j, minBW(n_i,n_j) is the minimum bandwidth on the path between the nodes n_i and n_j. After the node embedding, the links are mapped by using the k-shortest path algorithm.

4.6 Mapping tree algorithm

The mapping tree algorithm was proposed by Wang et al in [18]. This algorithm uses a node ranking approach and also maps the neighboring virtual nodes on substrate nodes which are closely located. Two unique parameters of the network play a key role in this algorithm. They are aggregate resource and selection factor. The aggregate resource is a property of a virtual or substrate node and indicates the amount of available node and link resources. The link resources refer to the total bandwidth capacity available on the outgoing links from the node. The aggregate resource of a node in a network is defined as

$$AR_{n_i} = c(n_i) * \sum_{l \epsilon L'_{n_i}} bw(l)$$

where L'_{n_i} refers to the set of outgoing links from the node n_i. The selection factor is a property of the substrate node and assists in the selection of the candidate substrate node for every virtual node. It is directly proportional to the aggregate resource of the substrate node and inversely proportional to the distance of the substrate node to the substrate node which hosts the neighbor of the virtual node which is being embedded. The selection factor of a substrate node is defined as

$$SF_{n_p} = \frac{AR_{n_p}}{d(n_p, n'_p)}$$

n'_p is the substrate node that hosts the parent node of virtual node that is being embedded

The mapping tree algorithm follows three steps for embedding a virtual network. First a mapping tree is constructed for the virtual network, virtual nodes are embedded and then the virtual links are embedded. As a first step in constructing the mapping tree, the virtual node with the maximum aggregate resource is chosen as the root of the tree. Then, its neighbors are added as child nodes successively. After the mapping tree is ready, the virtual nodes are ranked in breadth-first search order. The first choice of the candidate substrate node is the substrate node with the highest aggregate resource. Once this substrate node is saturated, the substrate node with the highest selection factor is selected as the candidate substrate node. The capacity constraint of the virtual node is always verified while choosing a candidate substrate node. Once the virtual nodes are embedded, the virtual links are embedded by using the k-shortest path algorithm. Again, the virtual links are mapped onto the substrate paths only after verifying the bandwidth constraints.

Chapter 5

Experiment setup

In this work, the performance of six virtual network embedding algorithms has been analyzed by simulating the process of virtual network embedding. The simulation was performed by coding the algorithms in Python and developing an object oriented framework representing the substrate and virtual networks. Multiple experiments were carried out by varying three parameters to study the short-lived virtual networks in different scenarios. The experiments have three parameters: Embedding algorithm, departure probability, substrate network. Each experiment is run over a period of 420 days. On each day, either a virtual network arrival or virtual network departure occurs with a given probability. This probability is referred to as departure probability. For the first 60 days, it is assumed that the consumers are charged a one time fee and hence there will not be any departures until the 60 day mark. After the 60 day mark, on each day there will be a single virtual network arrival or departure based on the departure probability. Six embedding algorithms were tested and compared in these experiments. Also, the embedding was observed in two different substrate topologies. This chapter describes the substrate and virtual networks used in the simulation, VNE request, determination of embedding result, VNE algorithms, VNE request arrivals and departures, performance metrics and the simulation.

5.1 Substrate networks

The substrate networks that were used to test virtual network embedding algorithms in this work are fat-tree topology and UUNET topology. This section gives a brief overview of each network.

5.1.1 Fat-tree topology

The fat tree topology was designed based on the clos topology which was developed by Charles Clos[1]. This hierarchical tree topology consists of two main elements called pods and core elements[4]. Pods consist of servers and two levels of switches. The servers are connected to the lower level switches(edge switches). The edge switches are connected to aggregation switches at the upper level. Core elements consist switches that interconnect pods. A k-ary fat tree consists of k pods. Each pod has k/2 aggregated switches and k/2 edge switches. And each k-port edge switch is connected to k/2 servers below and k/2 aggregation switches above. Figure 5.1 depicts a fat tree with k=4.The servers within a pod communicate with each other using the edge switches. The servers in different pods communicate through the core switches. The fat tree topology has redundant switches at the core and aggregation levels. Due to this the core switch is no longer a single point of failure. With a simple design, the fat tree provides redundancy and fault tolerance.

5.1.2 UUNET topology

UUNET is the first commericial Internet Service provider network. It is one of the most widely deployed IP networks in the world. In this work, the UUNET network topology in the United States is taken as the substrate network to test VNE. This topology has 49 network nodes and 135 network links.

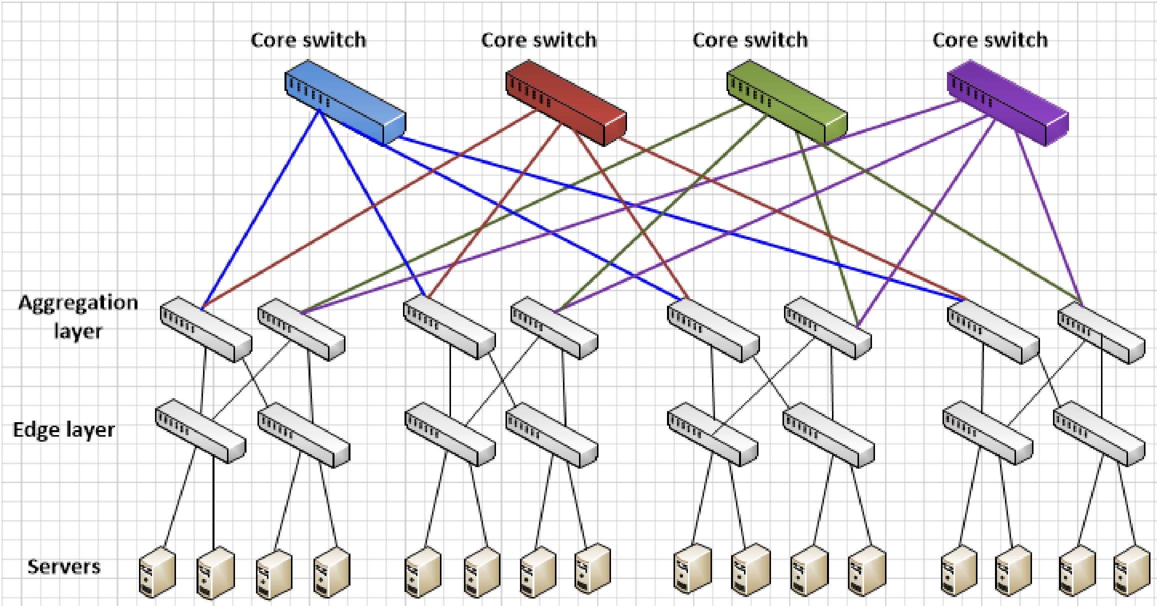


Figure 5.1: Fat-tree topology

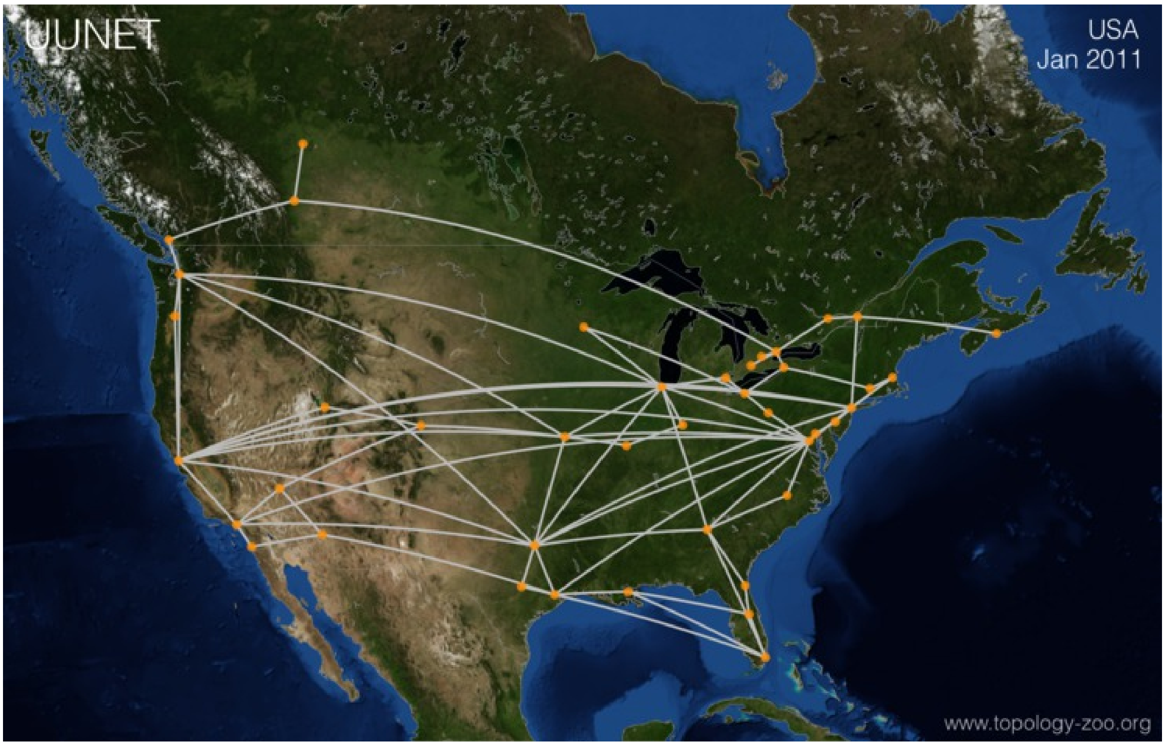


Figure 5.2: UUNET network topology Source:http://www.topology-zoo.org/maps/Uunet.jpg

5.2 Virtual networks

The virtual networks used for the simulation were generated using the Watts-Strogatz model. Watts and Strogatz put forward a small-world network topology model which is in-between regular and completely random topologies.[20]. This model generates small-world networks by starting with a regular fully connected graph and then performs a random re-wiring procedure on this graph. Each edge in the graph is rewired with a certain probability. In this work, the virtual networks are generated randomly using this model. The inputs given to the virtual network generator are the number of nodes(N), average node degree(D), rewiring probability(p), node demand and the bandwidth demand of the links. In this work, the number of nodes is equal to 1+V where V is a geometrically distributed random variable with an expected value of 10. Average node degree is a uniform random variable which is calculated based on N. The rewiring probability p is 0.1. For each virtual network, the node demand is set at 1 CPU capacity unit and the bandwidth demand is set at 100 bandwidth units for simplicity.

5.3 Virtual network requests

A virtual network request is a request for embedding a virtual network with a given set of demands. Prior to the embedding, 420 virtual networks are randomly generated. During the embedding simulation, on each day there will be an occurrence of virtual network arrival or virtual network departure based on the departure probability. The VNE was analyzed for three different values of departure probability: 0.25, 0.5, 0.75. In the beginning of each day, a decision is made about virtual network arrival or departure according to the departure probability. When there is a virtual network arrival, the embedding algorithm tries to embed the virtual network on the substrate network. When there is a virtual network departure, initially a virtual network is randomly chosen from the set of virtual networks already embedded on the substrate network. After selection, the substrate network resources being used by this virtual network are released to be used for future embeddings.

5.4 Embedding result

For every virtual network embedding request, the embedding algorithm tries to embed the virtual network onto the substrate network. The embedding would be successful if the constraints in the virtual network request are satisfied. In this work, the virtual node capacity demand and virtual link bandwidth demand are the constraints provided in the virtual network request. The embedding algorithm tests whether the node and link demands for all the virtual nodes and virtual links can be satisfied by the substrate network. If there are sufficient substrate resources to support the virtual network, the embedding would be successful. After every successful embedding, the substrate node and link capacities are updated to reflect the current available capacities. If there are not enough substrate resources to support the virtual network, the embedding will fail and the embedding algorithm rejects the virtual network embedding request.

5.5 Performance metrics

Performance metrics are needed to evaluate the quality of an embedding. Acceptance ratio and average revenue to cost ratio are two popular metrics that were used in the literature for evaluating virtual network embedding. In this work, the following metrics were used to evaluate the embedding produced by the different virtual network embedding algorithms.

- Acceptance ratio
- Revenue-to-cost ratio
- Average substrate path length
- Number of embedded virtual networks

- Substrate utilization

This section describes each metric and mathematical representations of these metrics.

5.5.1 Acceptance ratio

Acceptance ratio is the ratio of number of virtual network embeddings to the number of virtual network requests. It gives an idea about the number of virtual network requests being accepted. Mathematically, acceptance ratio is defined as

$$AR = VN_s/VNR$$

5.5.2 Revenue to cost ratio

The revenue to cost ratio is the ratio of revenue accrued and the cost incurred for embedding a virtual network. As each virtual network is embedded, additional revenue generated from that virtual network is added to the previous revenue. Similarly costs are computed on an additive basis as each network is embedded. The accrued revenue is calculated based on the amount of substrate resources consumed by the virtual networks. The cost is calculated based on the number of substrate resources allocated to the virtual network. The difference between the cost and revenue would depend on the substrate path length. The revenue and cost are represented mathematically as follows:

$$Rev = \sum_{n_v \epsilon N_v} c(n_v) + \sum_{l_v \epsilon L_v} bw(l_v)$$

$$Cost = \sum_{n_v \epsilon N_v} c(n_v) + \sum_{l_v \epsilon L_v)} hops(p(l_v, G_v)) * bw(l_v)$$

5.5.3 Average substrate path length

Substrate path length is the length of the path on which a virtual link is embedded. Average substrate path length of a virtual network is the average of the length of all substrate paths corresponding to all the virtual links in the virtual network. In this work, the substrate path length is measured in number of hops. The average substrate path length varies according to the placement of virtual nodes in the substrate network. Some algorithms may place virtual nodes closer to each other on the substrate network and therefore may have smaller substrate path length.

5.5.4 Number of virtual network embeddings

The number of virtual network embeddings is simply the number of virtual networks currently embedded on the substrate network. This may vary depending on the number of rejections of virtual network requests and also the departure probability.

5.5.5 Substrate utilization

Substrate utilization of a given substrate node is the ratio of substrate node capacity allocated for virtual nodes and the total substrate node capacity. It indicates the level of capacity usage by the virtual nodes. In this work, the substrate utilization of each substrate node is studied to analyze the distribution of virtual nodes in the substrate network.

5.6 Simulation

This section describes the simulation conducted for performance evaluation of VNE algorithms. Multiple experiments have been carried out by varying three parameters: Departure probability (0.25,0.5,0.75), substrate network (Fat-tree topology, UUNET) and the embedding algorithm (Random, LLF, MLF, Closeness centrality, Improved closeness centrality and mapping tree). In each experiment, the arrival and departure of virtual networks and then the actual embedding of the virtual network is simulated using a Python framework. The arrival and departure of the virtual networks is controlled by the departure probability. And the embedding is performed by the chosen embedding algorithm on the chosen substrate network. The aim of running the simulation is to obtain the acceptance ratio, revenue to cost ratio, substrate utilization and the other metrics which can be used to analyze the embedding process. The collected metrics represent the average value of 100 runs. The simulation runs for a length of 420 days with 1 virtual network arrival or departure per day. The departure of virtual networks starts only after the first 60 days. Once the virtual network departure starts, the embedding is analyzed roughly for a year (360 days). The virtual node CPU demand and virtual link bandwidth demand are set at 1 CPU capacity units and 100 bandwidth capacity units on an average. And the substrate node CPU capacity and link bandwidth capacity are set at 8 CPU capacity units and 1000 bandwidth capacity units.

Chapter 6

Results

This chapter presents the results and findings from the experiments. Overall, the mapping tree algorithm resulted in the best performance based on the metrics chosen in this work. Mainly mapping neighbors successively and onto substrate nodes that are closely located in the topology helped to achieve better results. VNE was simulated for the below six scenarios.

- VNE on Fat-tree topology, virtual networks depart with probabilty of 0.25
- VNE on Fat-tree topology, virtual networks depart with probabilty of 0.50
- VNE on Fat-tree topology, virtual networks depart with probabilty of 0.75
- VNE on UUNET topology, virtual networks depart with probabilty of 0.25
- VNE on UUNET topology, virtual networks depart with probabilty of 0.50
- VNE on UUNET topology, virtual networks depart with probabilty of 0.75

Six experiments are conducted for each scenario by using one of the six embedding algorithms for each experiment.

6.1 VNE on a fat-tree topology with departure probability 0.5

This section gives a general idea about the performance of each of the six embedding algorithms while embedding short-lived virtual networks on a fat-tree topology. The probability of a virtual network departing the substrate on any day is 0.5 in this case.

6.1.1 Acceptance ratio

The highest acceptance ratio was observed for the mapping tree algorithm as indicated by the Fig 6.1a. For the mapping tree approach, the acceptance ratio does not drop even though the virtual networks continue to arrive. Whereas in the case of the other five algorithms, there is a considerable drop in the acceptance ratio as the number of virtual networks increase. This indicates that the mapping tree can accommodate even more virtual networks compared to the other approaches. After time T=60, when the virtual networks begin to depart, the acceptance ratio goes up again and stabilizes after certain time. Especially, in the case of random algorithm, this variation can be seen clearly. The rise in acceptance ratio is due to the increase in substrate network's available capacity when the virtual networks depart. However, the virtual nodes may be distributed across the network increasing the substrate path length. The substrate path length is discussed next.

6.1.2 Average substrate path length

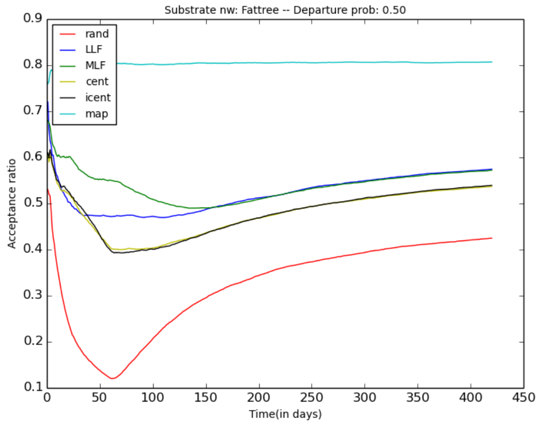
This metric gives an idea of the virtual network performance as it represents the average of the distances between every pair of virtual nodes in the substrate network. As shown in Fig 6.1c, this metric varies as the number of virtual networks in the substrate network increase. The algorithms that tend to embed virtual nodes in substrate nodes that are far from each other exhibit high average substrate path length. Random approach exhibits the highest value. As LLF algorithm tries to embed the virtual nodes on substrate nodes that have least load, the virtual nodes are distributed across the substrate network. Hence the distance between the virtual nodes increases, thereby increasing the substrate path length. As expected, average substrate path length of MLF is relatively low compared to LLF. This is because MLF tends to consolidate the virtual nodes in as few substrate nodes as possible. The closeness centrality based algorithms' performance with respect to this metric is similar to MLF, MLF being slightly better. Finally, mapping tree has the least average substrate path length which is due to embedding neighboring virtual nodes close to each other and the unique procedure it follows. As more and more virtual networks are embedded there will not be much choice and virtual nodes have to be embedded far from each other. In addition to this, when the virtual network leave the substrate, they may create fragmentation in the network. Due to this, virtual nodes have to be embedded significantly far from each other. This effect is seen after T=60 days. This is considerably less in the case of mapping tree and also in the case of MLF and centrality based algorithms to an extent.

6.1.3 Number of virtual networks

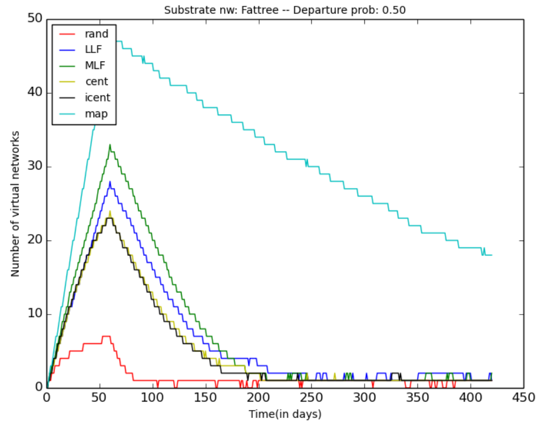
The number of virtual networks in the substrate network at a given time is depicted in Fig 6.1b for all the algorithms. This clearly indicates the number of virtual networks that the substrate network can accommodate. By using mapping tree approach, the highest number of virtual networks can be accommodated. By the 60 day mark, mapping tree approach embedded close to 50 virtual networks whereas the other algorithms embedded less than 35 virtual networks on the same substrate network. By testing the embedding of the same set of virtual networks on the same substrate network by using different algorithms, a clear picture of the performance of each algorithm is obtained.

6.1.4 Revenue-to-cost(RC) ratio

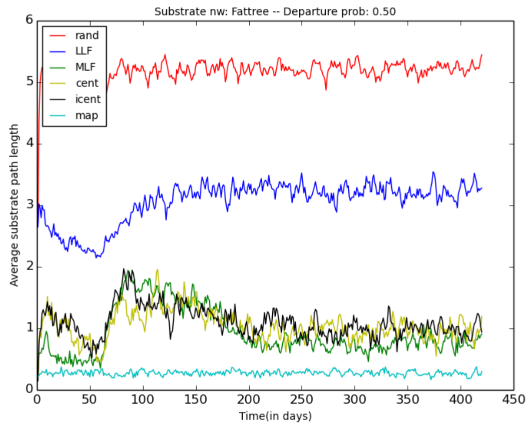
The RC ratio is indicated in the Fig 6.1d. This shows that the mapping tree not only accommodates more number of virtual networks but also ensures that this is done at



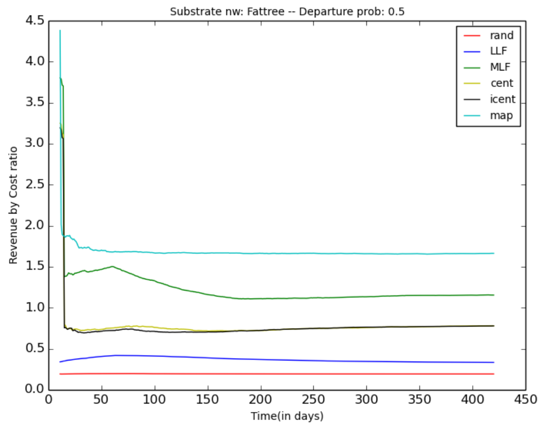
(a) Acceptance ratio



(b) Number of VNs in substrate



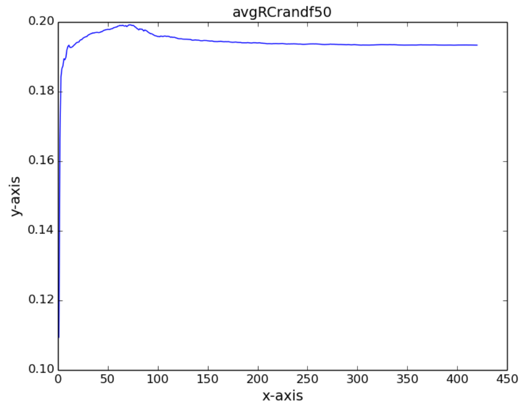
(c) Average substrate path length



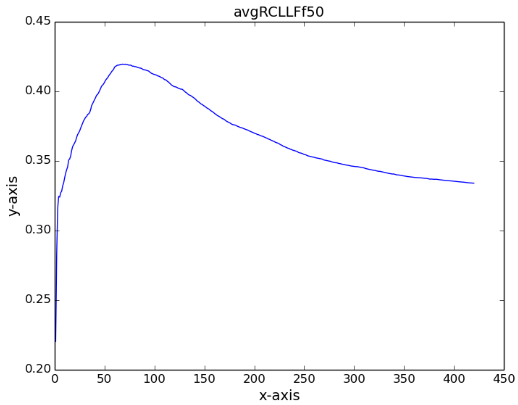
(d) Revenue by cost ratio

Figure 6.1: Plots for fat-tree topology with departure probability = 50%

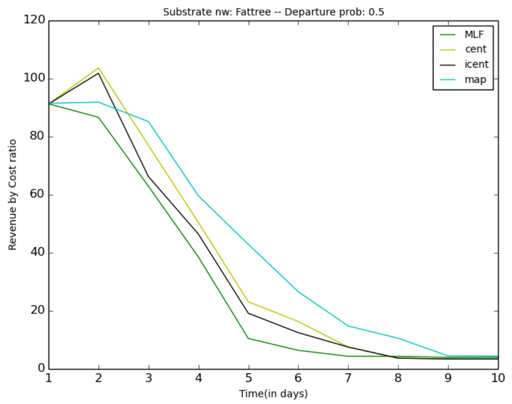
a reasonable revenue-to-cost ratio. A closer analysis of the RC ratio for random and LLF is depicted in Fig 6.2a and 6.2b. The revenue-to-cost ratio for random algorithm is at a low value and increases shortly after the 60 day mark when the virtual networks start to depart. Overall the ratio is below 1 indicating that the cost is always higher than the revenue accrued. However, the rise of the RC ratio until 60 is only due to the increase in number of embeddings as indicated by the Number of VNEs vs time graph in Fig 6.1b. LLF shows similar behavior but the ratio is higher in this case. The peak value of RC ratio for LLF is double that of random algorithm. Fig 6.2c indicates that the closeness centrality based algorithms, MLF and mapping tree have very high values of RC ratio approximately close to 90 in the first 10 days. This is because the initial mapping in these algorithms is consolidated to minimum number of substrate nodes which makes the substrate path length close to zero resulting in minimal cost and a high RC ratio. But as the number of virtual network embeddings increase, the substrate nodes start getting saturated and the virtual nodes have to be embedded in substrate nodes which may be far away from each other. This makes the substrate path length high and the cost slowly increase thereby bringing down the RC ratio.



(a) Random – RC ratio vs time



(b) LLF – RC ratio vs time



(c) MLF,cent,icent,map – RC ratio vs time

Figure 6.2: Closer look of the variation of RC ratio

6.1.5 Substrate utilization

The substrate utilization of the substrate nodes is measured at two instances (T= 60 days, 420 days) in the experiment. The substrate utilization at these two instances is depicted in Fig 6.3 and Fig 6.4. These figures indicate that the maximum utilization is seen in closeness centrality and mapping tree approach. It also indicates the trend of substrate utilization as the virtual networks leave the network. An interesting observation here is that even for the mapping tree approach, the maximum utilization is close to 50% only at T=60. This is potentially due to quick saturation of bandwidth capacities of the links before saturation of substrate node capacities. The figures 6.6 and 6.7 indicate the substrate utilization for the UUNET case. In this case, mapping tree performs better than the closeness centrality algorithm. This is probably due to the closeness centrality property which plays a better role in fat topology compared to tree topology.

6.2 VNE on a UUNET topology with departure probability 0.5

This section describes the performance of the six embedding algorithms on a UUNET topology. The probability of a virtual network departing the substrate on any day is 0.5 in this case.

6.2.1 Acceptance ratio

The study of VNE in UUNET topology also resulted in mapping tree producing the best results. The initial acceptance ratio is close to 1 and drops to 0.75. Once the virtual networks start departing, the acceptance ratio increases and stabilizes at a value around 0.8. The closeness centrality algorithms perform better than greedy algorithms. The MLF algorithm doesn't drop too low but at the end the acceptance ratio is lower than the closeness centrality based algorithms. LLF and random algorithms have the least acceptance ratio values in the experiment. Overall the algorithms that consolidate the virtual nodes seem to perform better than the other algorithms.

6.2.2 Average substrate path length

The average substrate path length exhibited by the mapping tree algorithm is the least and is less than 1. The smaller the value of average substrate path length, the better is the quality of embedding since the performance of the virtual network would be better. Again, the mapping tree and MLF algorithms have better substrate path lengths due to consolidation of virtual nodes. However, even the closeness centrality based algorithms have substrate path lengths close to MLF algorithm's values. LLF algorithm distributes the virtual nodes across the network and hence it has high substrate path lengths. And as expected, random algorithm exhibits the highest average substrate path length close to 3.5 on an average.

6.2.3 Number of virtual networks

The number of virtual networks is similar to the acceptance ratio metric in a way. Both indicate the number of virtual networks that can be supported by the substrate. However analyzing this metric would give us the number of virtual networks in the substrate network at different instances during the course of the experiment. The number of virtual networks for the mapping tree is the highest all throughout the experiment duration. MLF follows it closely until it reaches 60 day mark but after that the number of virtual networks falls down sharply. This indicates more virtual network rejections compared to mapping tree. Closeness centrality algorithms perform better than LLF but not as good as MLF to an extent. Random algorithm has the least number of virtual networks during the entire course of the experiment.

6.2.4 Revenue-to-cost ratio

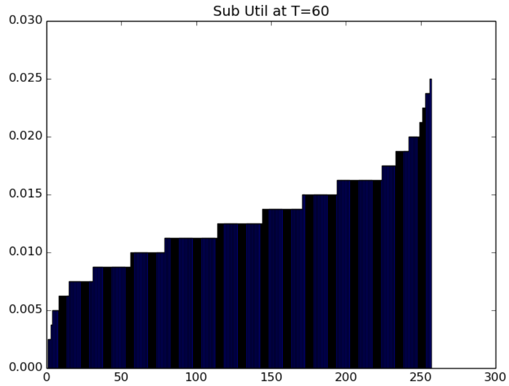
The revenue-to-cost ratio has very less variation when the departure of virtual networks continues. The revenue-to-cost ratio for mapping tree algorithm is the highest and is more than 1 all throughout the experiment. Whereas the revenue-to-cost ratio for LLF and random is less than 1 always. MLF again is close to mapping tree due to the common technique used in both algorithms(consolidation of virtual nodes). The closeness centrality based algorithms record a revenue-to-cost ratio that is lower than mapping tree but greater than 1 which is reasonable. Similar to the fat-tree case, this ratio is very high initially for Mapping tree and MLF.

6.3 VNE on a Fat-tree topology - varying departure probability

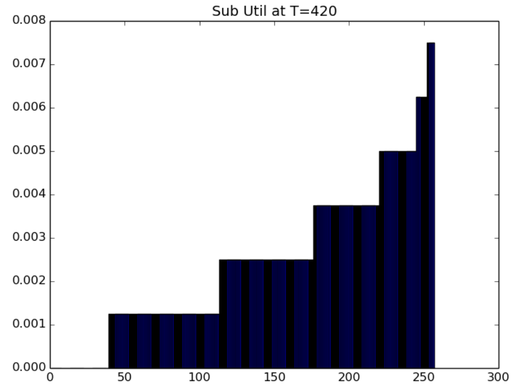
The acceptance ratio and revenue-to-cost ratio for varying departure probability is indicated in the figure 6.8. The acceptance ratio remains almost the same for different levels of departure probability. Whereas for the other algorithms we can see the difference in the acceptance ratio. As the departure probability increases, acceptance ratio increases. There is a sharp increase in acceptance ratio once the virtual networks begin to depart. In all cases, the mapping tree performs the best and random has the least acceptance ratio. There are slight variations in MLF and closeness centrality algorithms with varying departure probability. The MLF performs better than closeness centrality as the probability of departure increases. The revenue-to-cost ratio is similar for all levels of departure probability with only slight variations.

6.4 VNE on a UUNET topology - varying departure probability

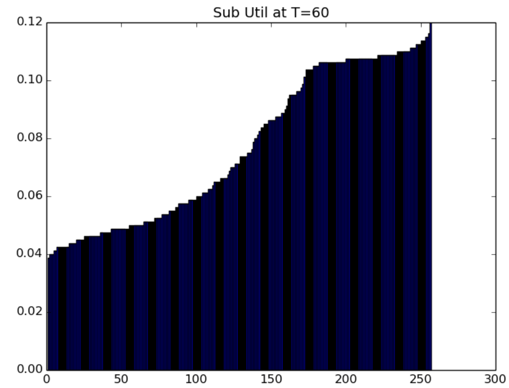
The acceptance ratio and Revenue-to-cost ratio for different values of departure probability is depicted in the figure 6.9. The acceptance ratio for mapping tree fluctuates for different values of departure probability unlike the fat-tree topology case. Even when the departure probability increases to 0.75, the mapping tree still has the best performance. As more and more networks leave, the LLF and random algorithms have an increase in the acceptance ratio almost close to the MLF algorithm in the case of UUNET topology. The revenue-to-cost ratio fluctuates for different levels of probability. Whereas in the case of fat-tree there were very slight changes. The structure of the topology makes a difference in the substrate path lengths which in turn changes the revenue-to-cost ratio. The closeness centrality based algorithms do better than MLF when the departure probability is 0.75. This probably indicates that MLF algorithm is slightly deteriorating due to fragmentation.



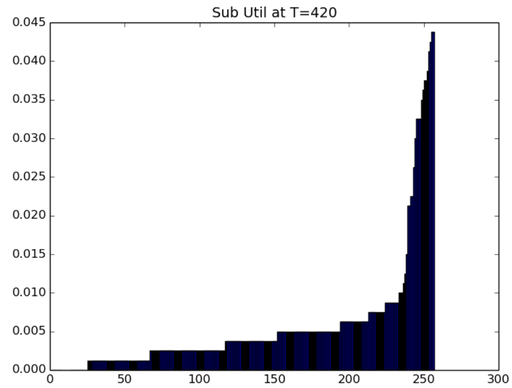
(a) Random – At T=60 days



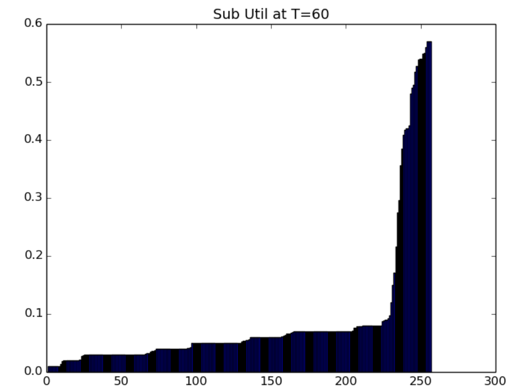
(b) Random – At T=420 days



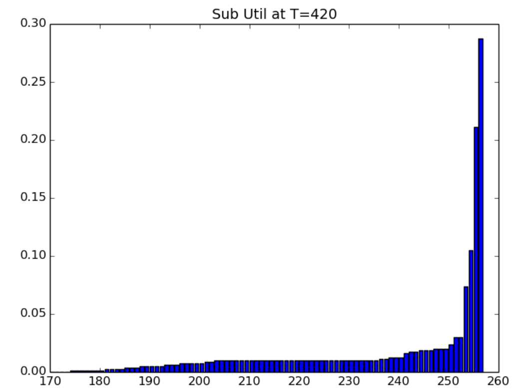
(c) LLF – T=60 days



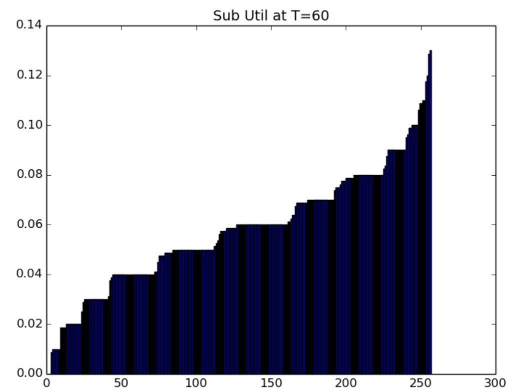
(d) LLF – T=420 days



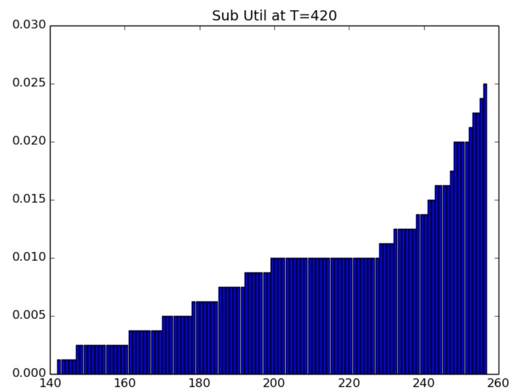
(e) MLF – T=60 days



(f) MLF – T=420 days

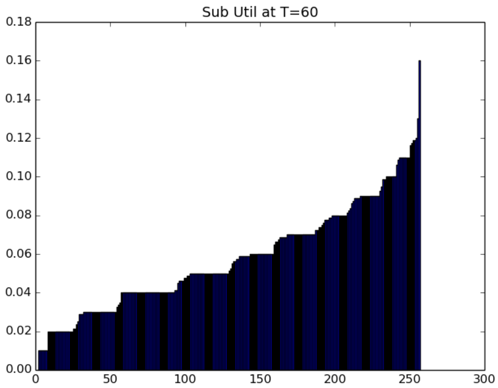


(g) Closeness centrality – At T=60 days

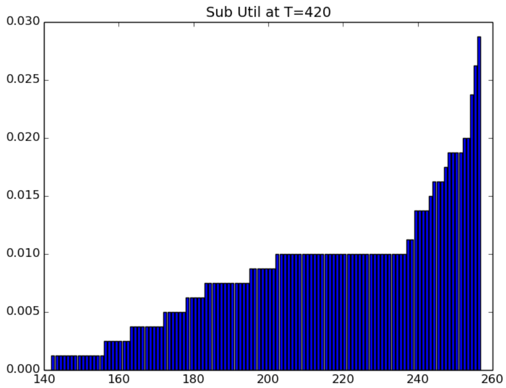


(h) Closeness centrality – At T=420 days

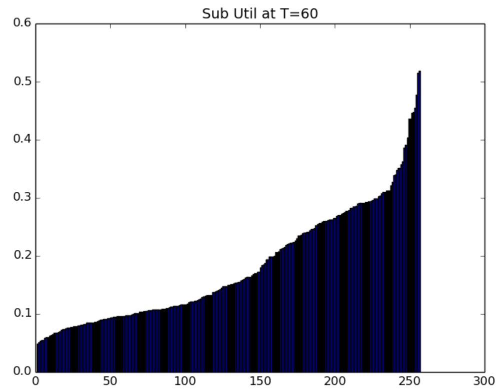
Figure 6.3: Substrate utilization vs substrate node rank in increasing order of utilization for fat-tree topology with departure probability = 50%



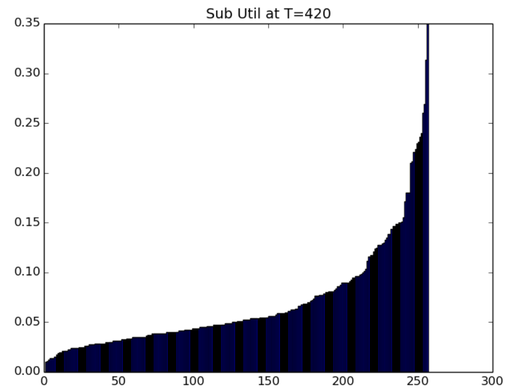
(a) icent – T=60 days



(b) icent – T=420 days

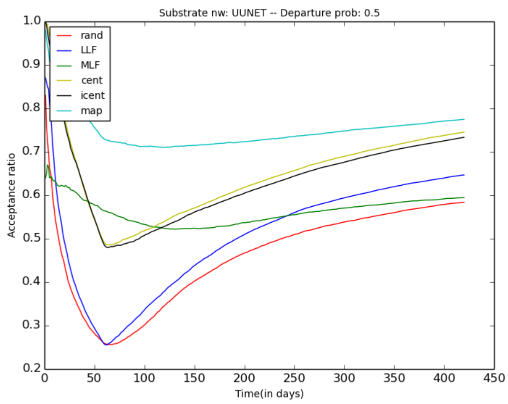


(c) Mapping tree – T=60 days

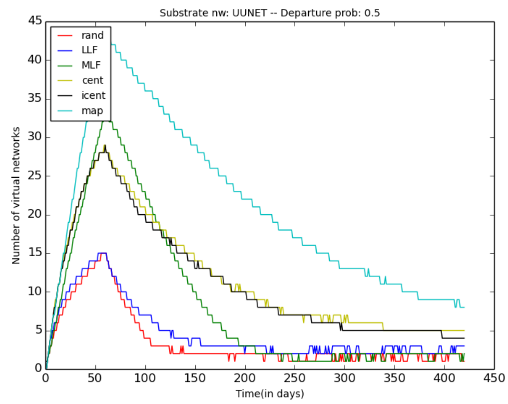


(d) Mapping tree – T=420 days

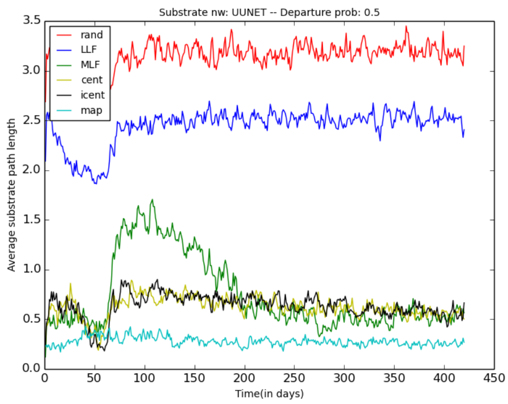
Figure 6.4: Contd: Substrate utilization vs substrate node rank in increasing order of utilization for fat-tree topology with departure probability = 50%



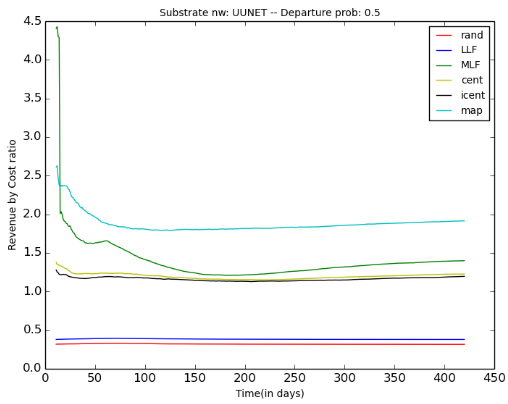
(a) Acceptance ratio



(b) Number of VNs in substrate

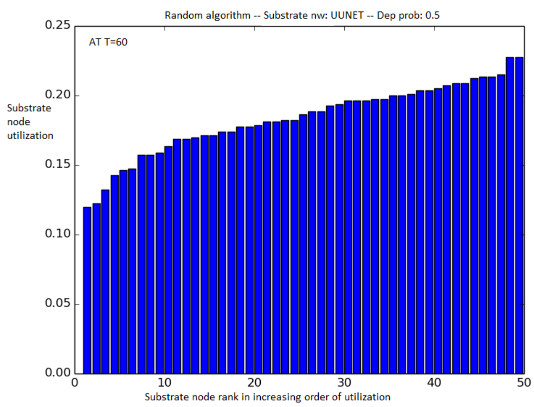


(c) Average substrate path length

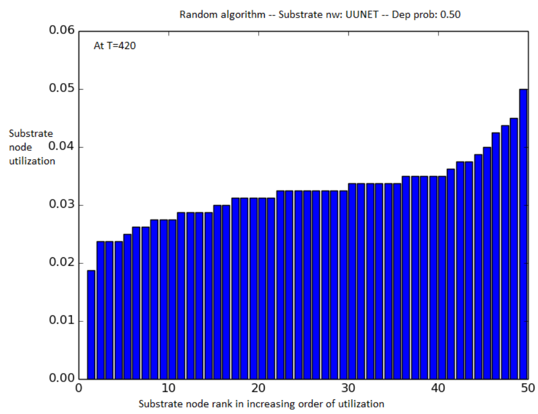


(d) Revenue by cost ratio

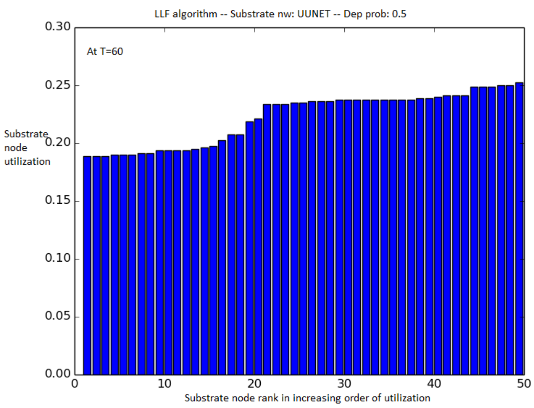
Figure 6.5: Plots for UUNET topology with departure probability = 50%



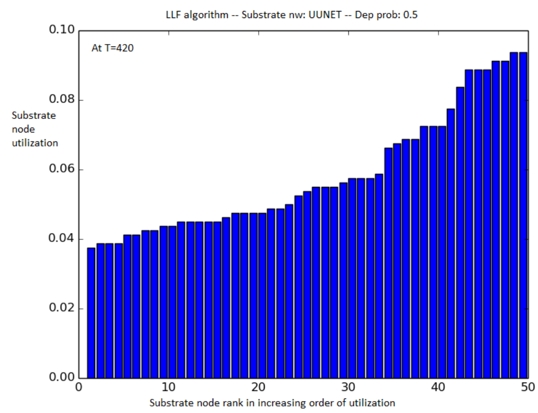
(a) Random – At T=60 days



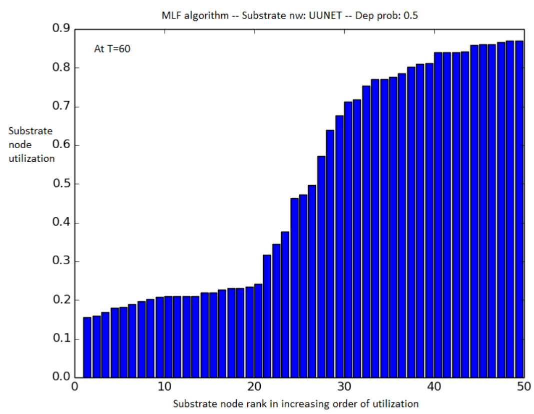
(b) Random – At T=420 days



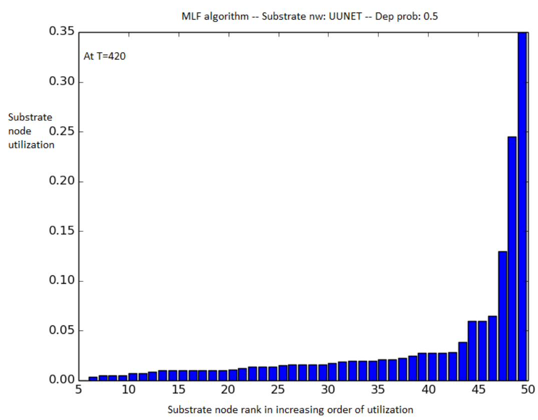
(c) LLF – T=60 days



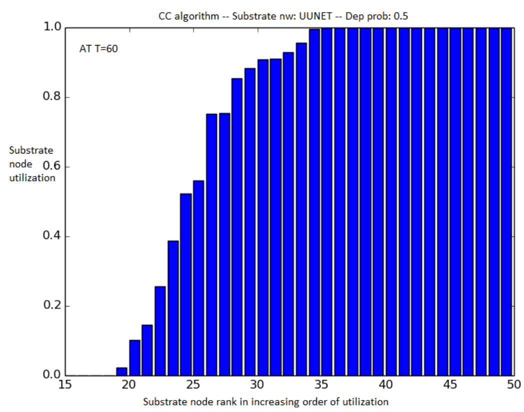
(d) LLF – T=420 days



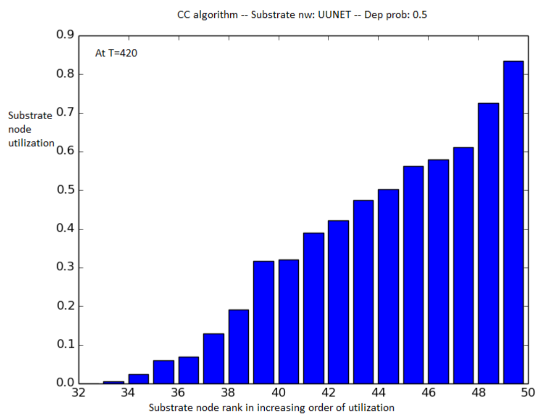
(e) MLF – T=60 days



(f) MLF – T=420 days

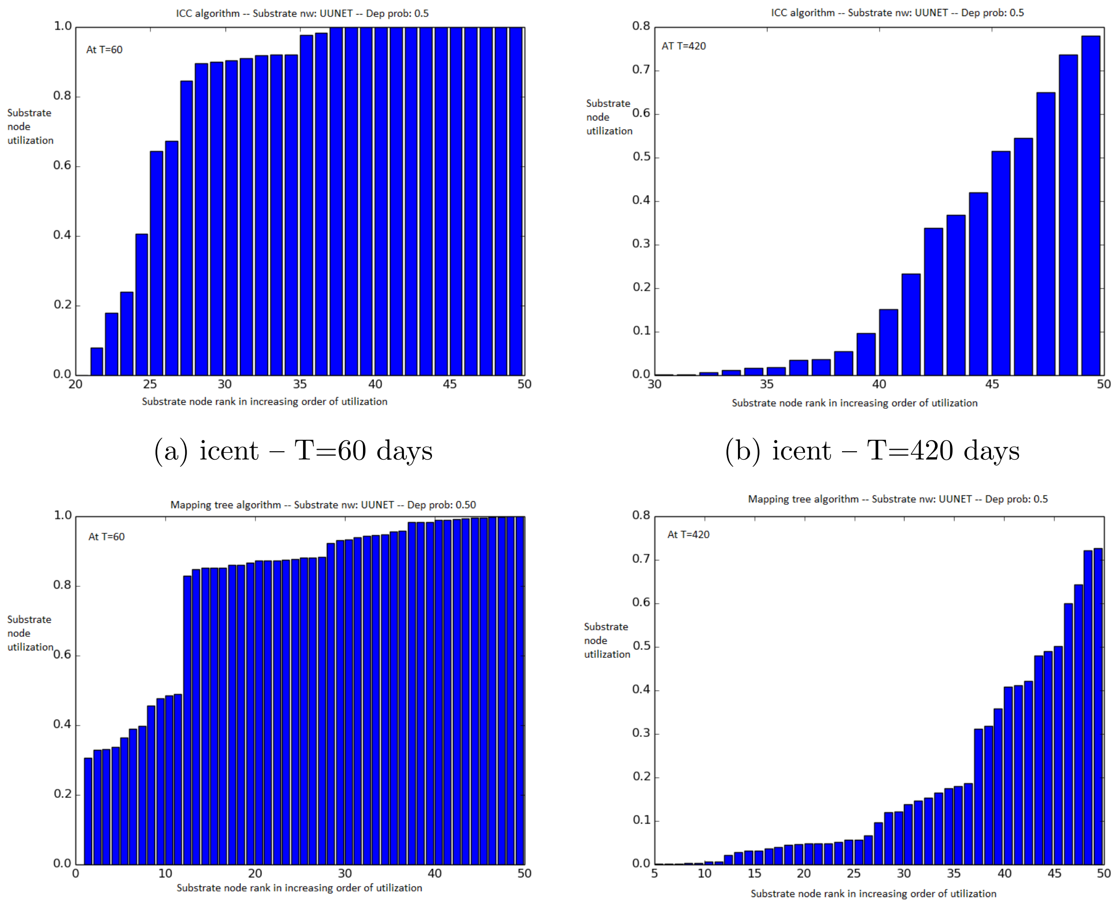


(g) Closeness centrality – At T=60 days



(h) Closeness centrality – At T=420 days

Figure 6.6: Substrate utilization vs substrate node rank in increasing order of utilization for UUNET topology with departure probability = 50%



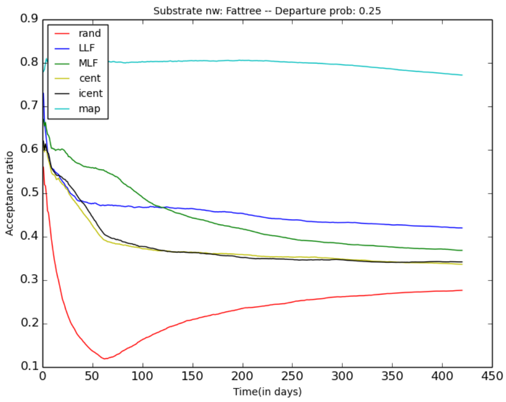
(a) icent – T=60 days

(b) icent – T=420 days

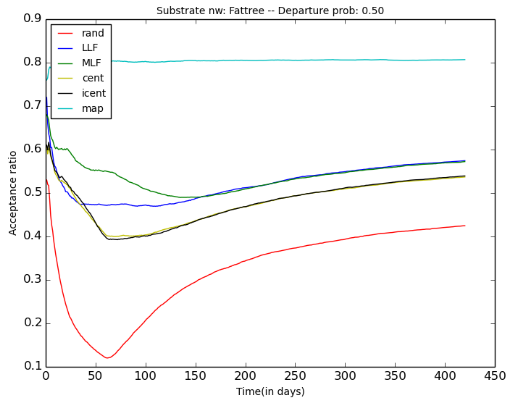
(c) Mapping tree – T=60 days

(d) Mapping tree – T=420 days

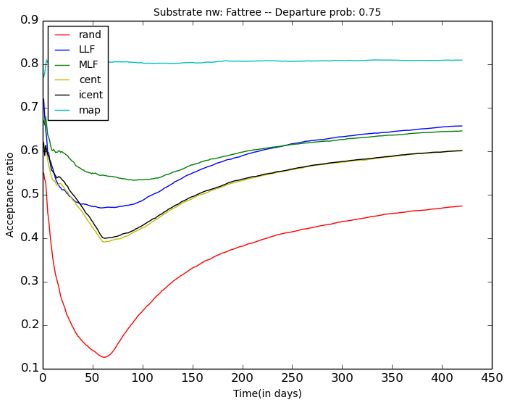
Figure 6.7: Contd: Substrate utilization vs substrate node rank in increasing order of utilization for UUNET topology with departure probability = 50%



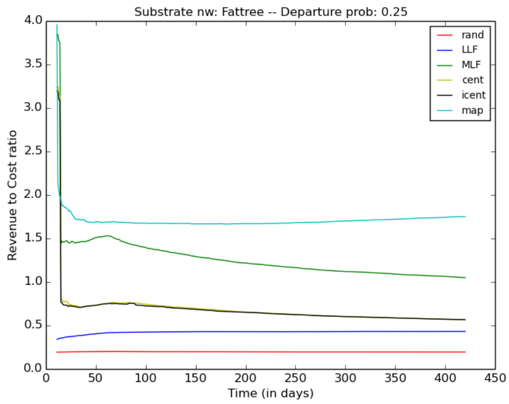
(a) Acceptance ratio-Dep Prob 25%



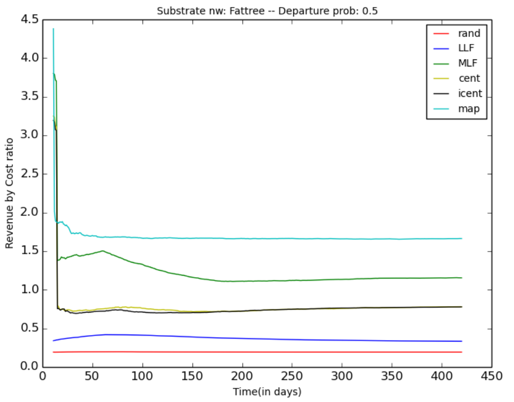
(b) Acceptance ratio-Dep Prob 50%



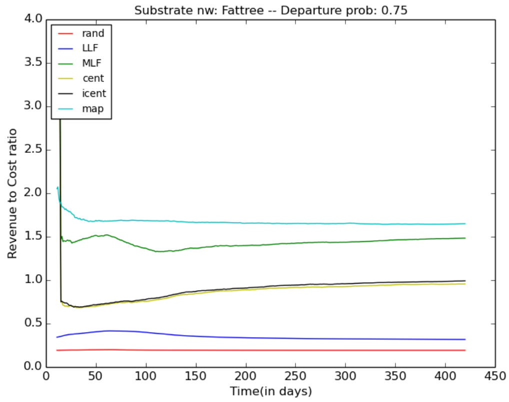
(c) Acceptance ratio-Dep Prob 75%



(d) Revenue-to-cost ratio-Dep Prob 25%

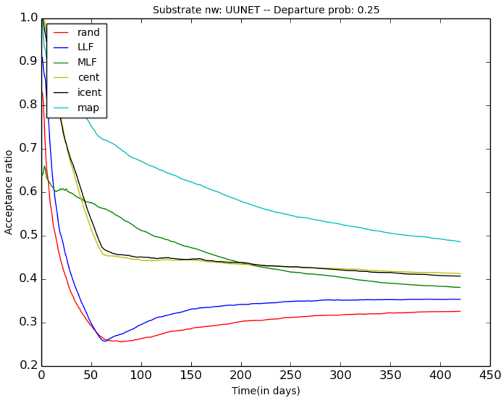


(e) Revenue-to-cost ratio-Dep Prob 50%

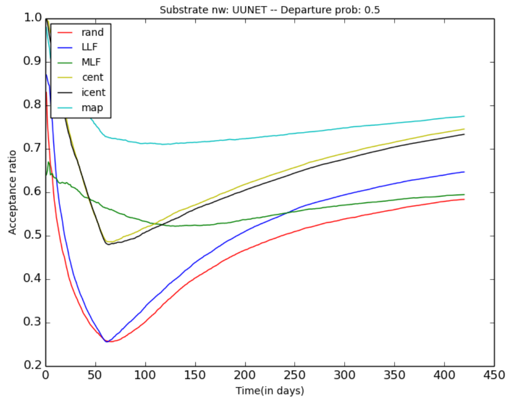


(f) Revenue-to-cost ratio-Dep Prob 75%

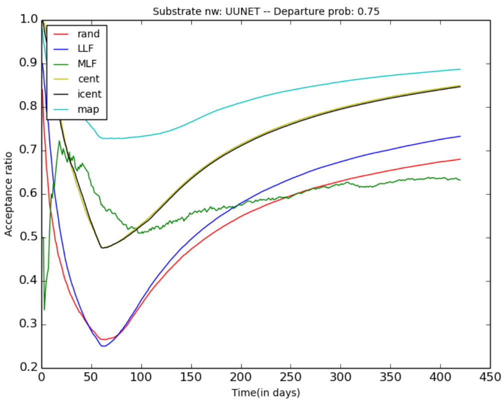
Figure 6.8: Acceptance ratio and Revenue-to-cost ratio for different values of departure probability in a Fat-tree topology



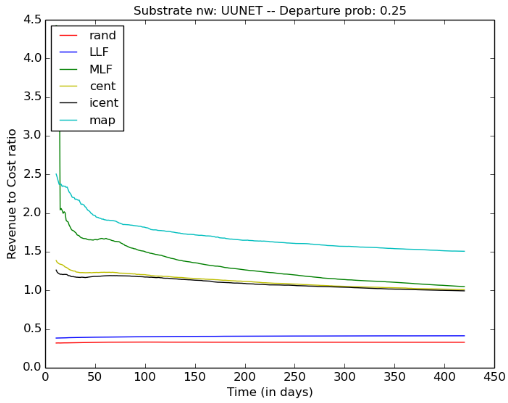
(a) Acceptance ratio-Dep Prob 25%



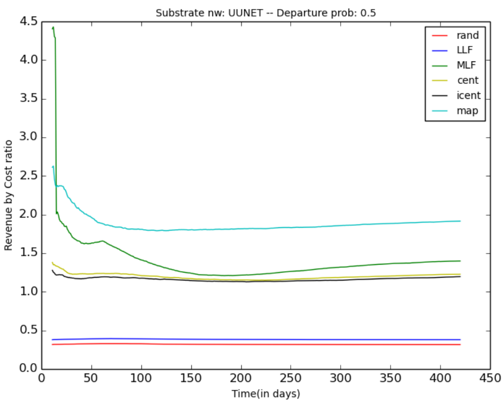
(b) Acceptance ratio-Dep Prob 50%



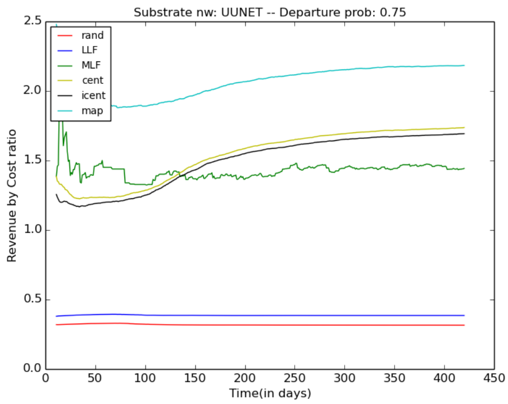
(c) Acceptance ratio-Dep Prob 75%



(d) Revenue-to-cost ratio-Dep Prob 25%



(e) Revenue-to-cost ratio-Dep Prob 50%



(f) Revenue-to-cost ratio-Dep Prob 75%

Figure 6.9: Acceptance ratio and Revenue-to-cost ratio for different values of departure probability in a UUNET topology

Chapter 7

Conclusion

7.1 Conclusion

Network virtualization has great benefits to offer to the networking community and cloud computing field. Virtual Network Embedding (VNE) is a crucial task in realizing network virtualization. It is necessary to have an effective VNE algorithm to achieve efficient operation of virtualized networks.

An evaluation and comparison of virtual network embedding algorithms in the specific case of short-lived networks was presented in this thesis. Through simulation, this thesis demonstrates that consolidating the virtual nodes improves the quality of embedding even when the virtual networks are short-lived.

In particular, VNE was closely studied in different scenarios to understand the various aspects that characterize an optimal embedding. Six VNE algorithms were implemented and VNE was simulated using these algorithms. Simulation was carried out with a Fat-tree topology (usually deployed in data centers) and also UUNET topology (which is a WAN-like flat topology). The virtual networks were randomly generated and short-lived. The life of the virtual networks in the substrate network was varied to study the level of fragmentation in the substrate network. A detailed performance evaluation was conducted for all the six algorithms. The results indicated low performance in the case of random algorithm where the virtual nodes are embedded randomly. The mapping tree algorithm which consolidated the virtual nodes and placed them close to each other in the substrate network exhibited the best performance and no adverse effects of fragmentation were observed in this case.

VNE problem is a complex problem due to multiple constraints and objectives. By simplifying it to an extent, various aspects of VNE can be studied that help in formulating better solutions to VNE. The evaluation of different approaches to VNE and the study of short-lived VNEs were the key contributions of this thesis work.

7.2 Future work

There are several areas of possible future work in the context of this thesis.

- The performance analysis can be extended to embedding algorithms that involve embedding in multiple connected substrate networks and also the embedding algorithms that use path splitting for improving the embedding.

- Another possible extension can be the formulation of a unique metric that measures the amount of fragmentation in the substrate network.

- Furthermore, this work can help in the development of embedding algorithms that perform reconfiguration of virtual networks to accommodate future virtual networks.

Appendix A

List of abbreviations

AR	Acceptance Ratio
CC	Closeness centrality
CPU	Central Processing Unit
ICC	Improved closeness centrality
LLF	Least Loaded First
MLF	Most Loaded First
RC	Revenue-to-Cost
VN	Virtual Network
VNE	Virtual Network Embedding
VNR	Virtual Network Request
WAN	Wide-area network

Bibliography

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM Computer Communication Review*, 38(4):63–74, 2008.

[2] M.F. Bari, R. Boutaba, R. Esteves, L.Z. Granville, M. Podlesny, M.G. Rabbani, Qi Zhang, and M.F. Zhani. Data center network virtualization: A survey. *Communications Surveys Tutorials, IEEE*, 15(2):909–928, Second 2013.

[3] Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo, and Jie Wang. Virtual network embedding through topology-aware node ranking. *SIGCOMM Comput. Commun. Rev.*, 41(2):38–47, April 2011.

[4] R.S. Couto, M.E.M. Campista, and L.H.M.K. Costa. A reliability analysis of datacenter topologies. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 1890–1895, Dec 2012.

[5] Hongyan Cui, Shaohua Tang, Xu Huang, Jianya Chen, and Yunjie Liu. A novel method of virtual network embedding based on topology convergence-degree. In *Communications Workshops (ICC), 2013 IEEE International Conference on*, pages 246–250, June 2013.

[6] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann. Vnr algorithm: A greedy approach for virtual networks reconfigurations. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6, Dec 2011.

[7] Min Feng, Jianxin Liao, Jingyu Wang, Sude Qing, and Qi Qi. Topology-aware virtual network embedding based on multiple characteristics. In *Communications (ICC), 2014 IEEE International Conference on*, pages 2956–2962, June 2014.

[8] Min Feng, Lei Zhang, Xiaomin Zhu, Jingyu Wang, Qi Qi, and Jianxin Liao. Topology-aware virtual network embedding through the degree. In *Information and Communications Technology 2013, National Doctoral Academic Forum on*, pages 1–6, Aug 2013.

[9] A. Fischer, J.F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4):1888–1906, Fourth 2013.

[10] Long Gong, Yonggang Wen, Zuqing Zhu, and T. Lee. Toward profit-seeking virtual network embedding algorithm via global resource capacity. In *INFOCOM, 2014 Proceedings IEEE*, pages 1–9, April 2014.

[11] Lan Li, Jianya Chen, Tao Huang, and Hongyan Cui. An efficient virtual network embedding algorithm based on subgraph. In *Wireless Personal Multimedia Communications (WPMC), 2014 International Symposium on*, pages 23–28, Sept 2014.

[12] Shuhao Liu, Zhiping Cai, Hong Xu, and Ming Xu. Security-aware virtual network embedding. In *Communications (ICC), 2014 IEEE International Conference on*, pages 834–840, June 2014.

[13] Wenzhi Liu, Yang Xiang, Shaowu Ma, and Xiongyan Tang. Completing virtual network embedding all in one mathematical programming. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 183–185, Sept 2011.

[14] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha. Optimal virtual network embedding: Node-link formulation. *Network and Service Management, IEEE Transactions on*, 10(4):356–368, December 2013.

[15] R.R. Oliveira, D.S. Marcon, L.R. Bays, M.C. Neves, L.S. Buriol, L.P. Gaspary, and M.P. Barcellos. No more backups: Toward efficient embedding of survivable virtual networks. In *Communications (ICC), 2013 IEEE International Conference on*, pages 2128–2132, June 2013.

[16] Sen Su, Zhongbao Zhang, A.X. Liu, Xiang Cheng, Yiwen Wang, and Xinchao Zhao. Energy-aware virtual network embedding. *Networking, IEEE/ACM Transactions on*, 22(5):1607–1620, Oct 2014.

[17] Huang Tao, Liu Jiang, Chen Jianya, and Liu Yunjie. A topology-cognitive algorithm framework for virtual network embedding problem. *Communications, China*, 11(4):73–84, April 2014.

[18] Gang Wang, Zhenmin Zhao, Zhaoming Lu, Yi Tong, and Xiangming Wen. A virtual network embedding algorithm based on mapping tree. In *Communications and Information Technologies (ISCIT), 2013 13th International Symposium on*, pages 243–247, Sept 2013.

[19] Zihou Wang, Yanni Han, Tao Lin, Hui Tang, and Song Ci. Virtual network embedding by exploiting topological information. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 2603–2608, Dec 2012.

[20] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of /'small-world/' networks. *Nature*, 393(6684):440–442, June 1998.

[21] Zichuan Xu, Weifa Liang, and Qiufen Xia. Efficient virtual network embedding via exploring periodic resource demands. In *Local Computer Networks (LCN), 2014 IEEE 39th Conference on*, pages 90–98, Sept 2014.

[22] Jiao Xue, Jiali You, Jinlin Wang, and Feng Deng. Nodes clustering and dynamic service balance awareness based virtual network embedding. In *TENCON 2013 - 2013 IEEE Region 10 Conference (31194)*, pages 1–4, Oct 2013.

[23] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29, March 2008.