Design and Analysis of Schedules for Virtual Network Migration

Samantha Lo
School of Computer Science,
Georgia Tech, Atlanta, GA
30332, USA
samantha@cc.gatech.edu

Mostafa Ammar
School of Computer Science,
Georgia Tech, Atlanta, GA
30332, USA
ammar@cc.gatech.edu

Ellen Zegura
School of Computer Science,
Georgia Tech, Atlanta, GA
30332, USA
ewz@cc.gatech.edu

ABSTRACT

The Internet faces well-known challenges in realizing modifications to the core architecture. To help overcome these limitations, virtual networks run over physical networks and use Internet paths and protocols as essentially a link layer in the virtual network. Effective use of the underlying network requires intelligent placement of virtual networks so that underlying resources do not incur over-subscription. Additionally, because virtual networks may come and go over time, and underlying networks may experience their own dynamic changes, virtual networks may need to be migrated—re-mapped to the physical network during active operation—to maintain good performance. In this paper we consider the problem of scheduling the sequence of node moves that take a virtual network from an original placement to a new placement. We build on prior work that achieves migration of a single node with minimal disruption to develop a model for the migration cost and latency for a given network migration schedule. We then develop algorithms for determining a single-node-at-a-time sequence of moves to minimize migration cost, and further consider multiple node moves in parallel to minimize migration time and cost. Our algorithms are the first we are aware of to systematically address the virtual network migration scheduling problem.

1. INTRODUCTION

The Internet faces well-known challenges in realizing modifications to the core architecture. The use of network virtualization has been proposed to help overcome these limitations (e.g.,[6, 15, 5, 7, 12]). Virtual networks run over physical networks and use Internet paths and protocols as essentially a link layer in the virtual network. We focus in this work on the specific form of virtualization where virtual routers are instantiated in physical router hardware and where multiple virtual routers belonging to different virtual networks may share the same physical router. The structure of such networks is illustrated in Figure 1, where two virtual networks are sharing a common physical network. On each physical router, the figure indicates which virtual routers are instantiated; virtual links are denoted by dashed lines that may traverse more than one physical link. This type of virtualization allows a substrate network provider to be decoupled from (and offer services to) overlay network providers.[1]

Virtual networks are attractive because they provide significant flexibility in operations and in mapping their requirements to physical network resources. Numerous studies have investigated mapping virtual networks to physical substrate resources in a manner that makes effective use of the physical network [18, 13, 17, 8]. Virtualization also allows flexibility in changing the mapping of a virtual network over time. Work in this area has explored reasons for network reconfiguration (or migration) such as changes in the traffic carried on the virtual network [9] or because of other virtual networks that share the same physical infrastructure have arrived or departed and the substrate resources should be remapped [18, 13, 17]. Additionally, virtual network migration can form the foundation of a "moving target defense" [4], where a virtual network evades detection or attack by changing its location in the physical network.

The work cited above considers questions of *policy* in virtual network migration and determines when to initiate the migration of an existing virtual network as well as the target new placement based on performance or security objectives. Our focus in this work is on questions of *mechanism*, i.e., how to actually move the network. These are largely unaddressed in prior work.

A virtual network migration mechanism needs to take into account the specifics of the router virtualization technology. It also needs to be guided by desired performance objectives such as: 1) how to manage disruption to existing traffic on the virtual network, 2) what is the additional overhead incurred by the physical substrate while the migration is in progress and 3) how long the migration process lasts.

Our work starts by leveraging prior work that designs a *single virtual router* live migration mechanism [16].

[1]This form of virtualization is distinguished from peer-to-peer virtual overlay networks, e.g. [7], where end users establish a network by using physical paths between end systems.

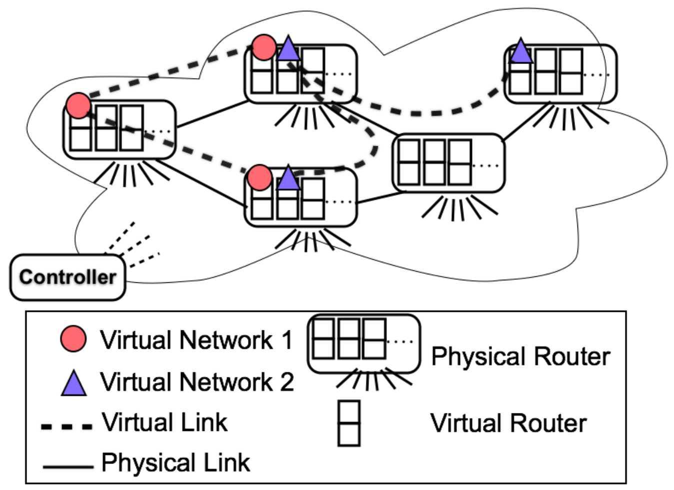


Figure 1: The system architecture for virtual network migration

That work aims to perform virtual router migration in a manner that does not disrupt current data flows using the migrating router.

We build on this single virtual router migration mechanism (outlined in more detail in the next section) and consider how to use it as a subroutine in the live migration of an *entire* virtual network. Specifically, we are interested in determining, given a new target location within the substrate for a virtual network, the best *schedule* of virtual router migrations that implement the desired network move. We maintain the goal of not disrupting current network traffic during the migration process and consider how to design a schedule that minimizes the overhead (cost) of network migration or the time it takes to complete the migration task. Our work is the first we are aware of that considers the question of scheduling a live network migration.

Our work starts with a model in Section 3 that describes the cost and duration of a network migration sequence. Our model is applicable to general schedules where multiple nodes may be moved in a single migration step. Then in Section 4, we develop algorithms for determining an optimal single-node-at-a-time sequence of moves to minimize migration overhead (cost) and schedules that allow for multiple node moves in parallel to minimize migration cost or time. We evaluate these algorithms in Section 5 with simulations of virtual networks on a physical network. We conclude our work in Section 6 with future work. The following section we give an overview of a single virtual router migration and virtual network migration process.

2. BACKGROUND

In this section, we first review the process of migrating a single virtual router. Then we describe the process of migrating a virtual network and the system architecture that supports this functionality.

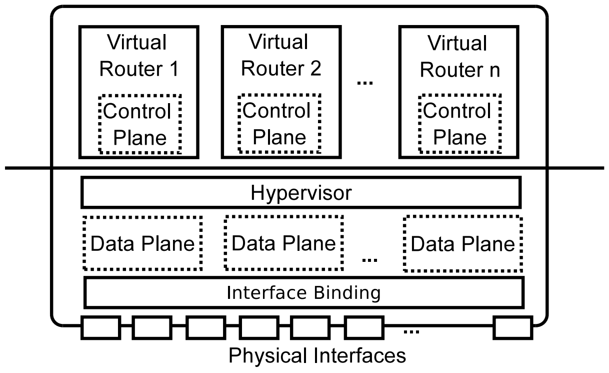


Figure 2: A simple model of a physical router with multiple virtual routers

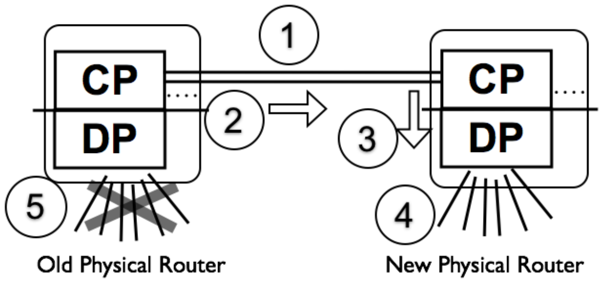


Figure 3: Virtual router migration steps 1-5. The old and new physical routers contain multiple virtual routers. (CP=Control Plane, DP=Data Plane)

2.1 Live Migration of a Single Virtual Router

Our work leverages the research in [16], which develops a mechanism for the live migration of a single router without disrupting current traffic traversing the virtual network. That work assumes a router architecture as shown in Figure 2. In the figure multiple virtual routers are instantiated on a physical router. Three features of virtual router architecture make it "migrateable": independence of the virtual router through OS virtualization support, separation of data and control planes, and the ability to dynamically bind a router's data plane to physical substrate interfaces.

With this router architecture, the work in [16] describes the following sequence of actions that are required to move a virtual router. The numbers in Figure 3 correspond to each step:

1. Setup tunnel: Setup a tunnel between the original and the final physical locations of the virtual

router.

2. Migrate control plane and copy memory: Create an image of the virtual router's control plane and transfer the image through the tunnel to the new physical router. At the same time, collect the routing update messages for this virtual router and forward them to the virtual router after the image transfer is completed such that they can be processed in the new location.

3. Clone data plane: Clone the data plane based on the existing control plane on the new physical node and repopulate the FIB. After this is completed, both old and new data planes are running at the same time.

4. Duplicate forwarding links between the virtual router and its neighbors: Set up the outgoing virtual links from the new physical router to its neighbors. Once these are ready, set up original incoming virtual links to redirect traffic from the neighbors of the original virtual router to the virtual router in the new physical machine. The new traffic flows are assigned to the new forwarding links.

5. Remove old forwarding links: Once old traffic flows at the old forwarding links are complete, remove the data plane instance in the original physical location to complete the move.

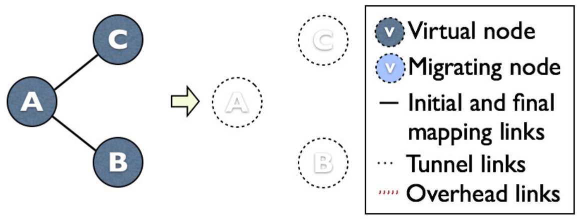
We use this single virtual router migration process as a subroutine to migrate each virtual router on a virtual network, as described next.

In the rest of the paper, we use the term "virtual node" and "virtual router" interchangeably.

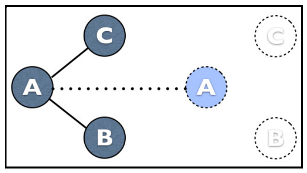
2.2 Network Migration Process

The example shown in Figure 4 demonstrates a virtual network migration. The example network has three virtual nodes, A, B and C, which are moved one at a time, in three stages, across the substrate. Each stage has two parts. During the *preparation part*, the control plane moves to its new physical node and the data plane is cloned (Step 1-3 in Section 2.1). After the preparation part, during the *migration part*, the duplicate forwarding virtual links are created and eventually the old forwarding links are removed (Steps 4-5 in Section 2.1). In the figure, the preparation part of a stage is illustrated in the left column, while the migration part is illustrated in the right.

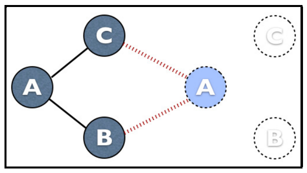
The migration sequence in the figure is (A, C, B). After A's control plane is migrated in the preparation part of Stage 0 in Figure 4(b), virtual links between the physical nodes of A and its neighbors, B and C are set up and are working in parallel with the original virtual links in the migration part in Figure 4(c). After the migration of node A, the virtual links between the



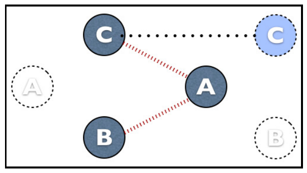
(a) A virtual network with node A, B and C on their initial mappings



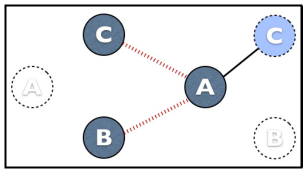
(b) Stage 0: Preparation part—copying the control plane of A



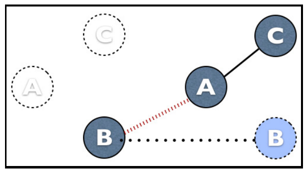
(c) Stage 0: Migration part—duplicating forwarding links between A and its neighbors



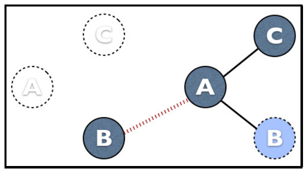
(d) Stage 1: Preparation part—copying the control plane of C



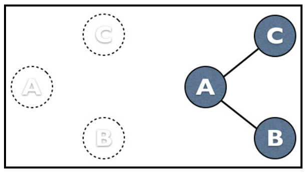
(e) Stage 1: Migration part—duplicating forwarding links between C and its neighbor



(f) Stage 2: Preparation part—copying the control plane of B



(g) Stage 2: Migration part—duplicating forwarding links between B and its neighbor



(h) Stage 2: Finished migration

Figure 4: Migrating a virtual network with virtual node A, B and C. (Physical network is not shown.) The migration sequence is A, C and then B.

migrated node A and its neighbors are not finalized until all the neighbors have been migrated. We will return to this issue when we analyze the cost of a move, but for now we simply note that there are migration overhead links that must be maintained in support of a migration in progress. The length (physical network hop count) and the duration (time maintained) of any overhead link is influenced by the migration sequence. Our algorithms to decide on a migration schedule will aim in part to reduce length and duration of overhead links.

2.3 System Architecture for Virtual Network Migration

To enable the network to migrate and determine the migration sequence, we adopt a simple system architecture. Figure 1 shows our system architecture which is similar to some existing overlay network architectures such as GENI [10]. In our architecture, the physical network has a controller to initiate and synchronize the virtual network migrations. The controller connects to all of the physical routers through in-band or out-of-band tunnels. When a request for a virtual network migration is initiated based on reconfiguration goals, the controller is given initial and final virtual network mappings. The controller decides the node migration sequence using algorithms running locally or remotely, and that are discussed in Section 4. The controller initiates the steps for each single node migration to accomplish the network migration.

3. PROBLEM STATEMENT

We now turn to a formal statement of the migration problem and a model for the cost and time associated with a given virtual node move sequence.

3.1 The Migration Problem

Let $P = (R, E)$ be the physical network where R is a set of physical routers and E is a set of physical links. Let $G = (V, L)$ be a virtual network where V is a set of virtual nodes and L is a set of virtual links.

Let $m : V \to R$ be a function that maps the set of the virtual nodes in G to a set of physical routers in P, i.e. $m(v) = r$ means a virtual node $v \in V$ is mapped to a physical router $r \in R$. A virtual link is mapped to a physical path (an ordered list of connected physical routers) where the endpoints of the virtual link are mapped to the first and last routers on the path and each pair of physical routers in the list are joined by a physical link.

We are given an initial mapping $m_0 : V \to R$ and a final target mapping $m_f : V \to R$. The goal of migration is to determine the most efficient schedule to move virtual nodes so as to accomplish the virtual network move from m_0 to m_f, where efficiency may be measured by overhead cost of migration, total migration time, or a combination of the two.

As explained earlier, network migration consists of a sequence of stages, where each stage migrates a subset of the virtual nodes. Let $S = (S_0, S_1, S_2, \ldots)$ be a sequence of node migrations that move the virtual network from m_0 to m_f. At stage i, the set of virtual nodes moved is $S_i \subseteq V$ such that $\forall i \neq j \; S_i \cap S_j = \emptyset$ and $\forall i \; S_i \neq \emptyset$ and $\bigcup_i S_i = V$.

There is an additional constraint on the sets S_i, namely that a given stage, we cannot move two nodes that are neighbors in the virtual network. This constraint arises because of the way that links are maintained in the single node move procedure. If two neighbors are moved at the same time, the virtual network will enter an inconsistent state. When we devise algorithms to move multiple nodes at a time, we are careful to satisfy the no-neighbors constraint.

We are interested in two important performance metrics: 1) the cost of a network migration representing the bandwidth impact on the physical network of the migration process, and 2) the completion time of a virtual network migration. We describe each of these formally in the next two subsections, starting with the time.

3.2 Network Migration Time

Recall that each stage consists of a preparation part and a migration part, as illustrated in Figure 5. The preparation part consists of setting up the tunnels, migrating the control planes and cloning the data planes by repopulating the FIB. The migration part consists of duplicating the forwarding links, waiting for all old data flows to complete, and finally removing the old forwarding links. After stage i, the network has fully completed all moves in the sequence S up to and including S_i.

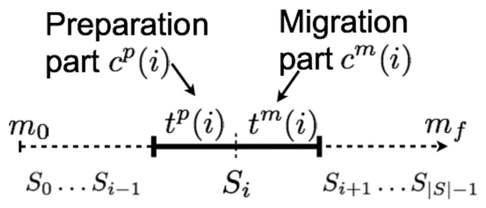


Figure 5: Illustration of the time of a virtual network migration

We denote the time taken in the *preparation part* and the *migration part* at stage i by $t^p(i)$ and $t^m(i)$, respectively. The total time for a particular migration sequence S is then:

$$\sum_{i=0}^{|S|-1} (t^p(i) + t^m(i)) \tag{1}$$

The time spent in the preparation part $t^p(i)$ depends on the following characteristics of the migrating nodes in S_i: the available bandwidth of the tunnels, the size of the control plane images, and the time to repopulate each FIB. For simplicity of exposition, we assume for a given virtual network that the control plane image sizes are the same for all the nodes. When the sizes of the control planes are the same and the data plane technology of the physical routers are the same, the FIB repopulation times for all the nodes are also the same. An interesting effect takes place, however, when nodes migrate together at the same stage. As more nodes move together, there is a higher probability that the tunnels used to move the control planes may share a bottleneck physical link. Thus while it is generally advantageous to move multiple nodes at a time, too much interference in the control plane tunnels can reduce or eliminate the benefit.

We model the time in the preparation part $t^p(i)$ as a function of the nodes in the set S_i and the time to migrate a control plane and repopulate a FIB, $f(S_i, \alpha)$, where α is the time for migrating the control plane and populating the FIB of a single node. We explore different functions $f(S_i, \alpha)$ in the evaluation section.

The time in the migration part $t^m(i)$ mainly depends on the traffic flows on the old forwarding links of the migrating nodes. Once the flows are over, the migrating nodes can completely switch to the new forwarding links. We assume that the time for these flows to finish is constant over all the stages and independent of the number of nodes moved at a stage.

Given the time computation, it should be clear that the primary method for reducing the migration time is to move as many nodes as possible together, thus reducing the number of stages, while taking care not to overload physical links that are needed to move the control planes.

3.3 Network Migration Cost

We model the impact of the virtual network migration on the physical network as a cost function that is divided into the cost during the preparation part ($c^p(i)$) and the cost during the migration part ($c^m(i)$) of each migration stage i. Our measure of cost is bandwidth consumed on physical links (bits/sec) multiplied by duration of consumption (sec), hence our cost units are bits. In the preparation part, there is a bandwidth impact on the physical network resulting from the transfer of the control plane across the tunnel established from the original location to the final location. Regardless of the migration sequence, each control plane must be transferred, and the cost is simply the sum of all control plane sizes. Hence this cost component is a constant.

We are more interested in the cost implications that derive from overhead virtual links. In both the preparation part and the migration part, there are a set of virtual links that are present only because migration is

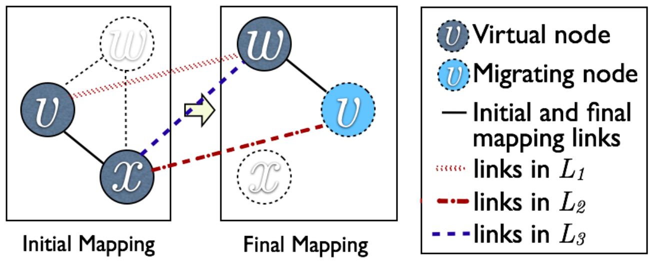


Figure 6: Illustration of types of links at stage i when v is migrating after w has moved at stage $i-1$.

taking place and that are dependent on the migration sequence. To understand the impact of these overhead virtual links, we divide the nodes into three sets at stage i:

1. Nodes that will move in this stage (i.e., $S_i \in V$),
2. Nodes that have already moved in prior stages, denoted $A_i \in V$, and
3. Nodes that have not yet moved, denoted $Y_i \in V$.

where $S_i \cup A_i \cup Y_i = V$ and $S_i \cap A_i \cap Y_i = \emptyset$.

We classify the overhead virtual links at stage i into three sets:

1. $L_1(i)$: Virtual links between the original mappings of the moving nodes (S_i) and final mappings of their already moved neighbor nodes (A_i), i.e. $l = (v, w)$ which the physical path is $(m_0(v), m_f(w))$ where $v \in S_i, w \in A_i$,
2. $L_2(i)$: Virtual links between the final mappings of the moving nodes (S_i) and the original mappings of their not moved neighbor nodes (Y_i), i.e. $l = (v, x)$ which the physical path is $(m_f(v), m_0(x))$ where $v \in S_i, x \in Y_i$, and
3. $L_3(i)$: Virtual links between the final mappings of the already moved nodes (A_i) and the original mappings of their not moved neighbor nodes (Y_i), i.e. $l = (w, x)$ which the physical path is $(m_f(w), m_0(x))$ where $w \in A_i, x \in Y_i$.

Figure 6 illustrates a simple three-node virtual network with nodes x, v, and w. In the stage illustrated, node v is moving, node w has already moved, and node x has yet to move. The virtual link (v, w) which is mapped to the physical path $(m_0(v), m_f(w))$ is of type L_1, the virtual link (x, v) which is mapped to the physical path $(m_f(v), m_0(x))$ is of type L_2, and the virtual link (w, x) which is mapped to the physical path $(m_f(w), m_0(x))$ is of type L_3.

The links in $L_1(i)$ and $L_3(i)$ are present throughout both the preparation and migration parts of stage i. The links in $L_2(i)$, on the other hand, are present only during the migration part of stage i, indeed they are created to maintain connectivity when the nodes in S_i are moving. We use $p(l)$ to denote the bandwidth impact of virtual link l. Later we simplify by assuming $p(l)$ is the number of physical links in path l.

The cost of the virtual network in the preparation part of stage i is then:

$$c^p(i) = t^p(i) \sum_{l \in (L_1(i) \cup L_3(i))} p(l) \tag{2}$$

The cost of the virtual network in the migration part of stage i is:

$$c^m(i) = t^m(i) \sum_{l \in (L_1(i) \cup L_2(i) \cup L_3(i))} p(l) \tag{3}$$

The total cost for a particular migration sequence S is the sum of the cost in all preparation and migration stages:

$$\sum_{i=0}^{|S|-1} c^p(i) + c^m(i) \tag{4}$$

4. ALGORITHMS FOR VIRTUAL NETWORK MIGRATION

We present three algorithms for migrating virtual networks. These algorithms can be divided into two types: moving one node at a time and moving multiple nodes at a time. We design the algorithms with two goals. One of the goals is to minimize the total migration time of the migration, which is described in Equation 1. The other goal is to minimize the total cost, which is described in Equation 4.

In the move-one-node-at-a-time scheme, a virtual node can start moving to its final placement only when the previous virtual node has finished its move. We find a move sequence $S^* = (S_0, S_1, \ldots)$ where $\forall i \; |S_i| = 1$.

In the move-multiple-nodes-at-a-time scheme, multiple nodes can be migrated at the same stage. However, we are constrained not to move nodes that are neighbors in the virtual network at the same stage.

In this section, we first introduce the algorithm Local Minimum Cost First (LMCF) which produces a move-one-node-at-a-time schedule to minimize the migration cost. Then we present two move-multiple-nodes-at-a-time algorithms: Maximal Independent Set-Size Sequence (MIS-SS) and Maximal Independent Set-Local Minimum Cost First (MIS-LMCF) which minimize the time and the cost of the migration. Note that we do not have an algorithm in the category of move-one-node-at-a-time with the goal of minimizing the time of migration because the number of stages required for any one-node-at-a-time scheme is the same.

4.1 Local Minimum Cost First (LMCF)

The goal of Local Minimum Cost First (LMCF) algorithm is to minimize the migration cost under the move-one-node-at-a-time scheme.

LMCF is described in Algorithm 1. First we compute the initial migration cost of the migration part $c^m(0)$ of stage 0 for each virtual node and choose the cheapest one to migrate first. Then from the rest of the unmoved virtual nodes, we recompute the migration cost c^m for each of them based on the current link mapping at that stage. Again, we choose the one that gives the cheapest cost and migrate it. We repeat until the network is completely migrated. If there are nodes that give the same cost, we pick the node with the lowest cost of the preparation part c^p. If both of their migration and preparation parts' costs are the same, we break the tie by randomly picking one of the cheapest nodes to be moved next.

Algorithm 1 Local Minimum Cost First

```
 1: function LMCF($G = (V, P)$, $P = (R, E)$, $m_0, m_f$)
 2:     set sequence $S = [\,]$                ▷ Result sequence
 3:     for $i = 0 \to |V| - 1$ do
 4:         $j = 0$
 5:         for $v \in V - (S_0 \ldots S_{i-1})$ do
 6:             $S_i = v$
 7:             $cost[v] = c^m(i)$   ▷ Find the cost for $v$ to
    be migrated at stage $i$
 8:             if $j = 0$ then
 9:                 $minv = v$
10:                 $mincost = cost[v]$
11:             else if $cost[v] < mincost$ then
12:                 $minv = v$
13:                 $mincost = cost[v]$
14:             end if
15:             $j = j + 1$
16:         end for
17:         $S_i = minv$             ▷ Return the node with
    minimum cost to move at stage $i$.
18:     end for
19:     return $S$
20: end function
```

4.2 Maximal Independent Set-Size Sequence (MIS-SS)

The goal of MIS-SS is to minimize the migration time. In MIS-SS, we simplify the time as the number of stages involved in a virtual network migration. We consider minimizing the number of migration stages by migrating multiple nodes at a time. We first determine the nodes which can be migrated at the same stage. We call the set of nodes to be migrated at a stage as *migration* *set*. Because of the no-neighbor constraint, the nodes to be migrated at the same stage cannot be neighbor among each other. The set of these non-neighbor nodes is called *independent set*. We find the maximum sets among these independent sets such that each set is not a subset of another set. These sets are called *maximal independent sets* (MISs).

We use an algorithm from [2] to generate MISs. For each node, this MIS algorithm generates a maximal independent set with inputs of a graph and a node of the graph. We use this algorithm and input each node to the algorithm to generate the maximal independent sets of the graph.

With the migration sets produced by the MIS algorithm, we design an algorithm to determine the migration sequence of these sets to minimize the number of stages. MISs are not maximum independent sets and they can have intersections. Since these sets can have intersections, when we migrate a set, some of the nodes in other sets may have been migrated as well. Thus, the remaining sets shrink. The number of stages involved in a virtual network migration can be highly reduced from LMCF, to the number of MISs, then to the number of migration sets in a set sequence which eliminates more nodes at the earlier stages.

In MIS-SS(Algorithm 2), first we use the MIS algorithm to determine migration sets of a virtual network. After the grouping, MIS-SS determines the migration sequence by picking the largest maximal independent set to be migrated first. At each stage, MIS-SS picks the largest remaining set to be migrated. After each stage, MIS-SS removes the nodes which have been migrated from the remaining sets. The remaining sets shrink when more nodes are migrated. If the sets are highly intersected, the time for migration can be reduced significantly as well.

4.3 Maximal Independent Set-Local Minimum Cost First (MIS-LMCF)

In MIS-LMCF, we attempt to minimize a combination of cost and time. Because when we shorten the time of the migration, we can minimize the time of the overhead links staying at the migrating state and further minimize the cost of the migration. In Algorithm 3, we move multiple nodes at a time to minimize the migration time. We also consider the migration cost of each migration sets to schedule the migration sequence.

We start with the sets of nodes $MSet[\,]$ from the MIS algorithm. Then we use a modified version of LMCF to determine which set should be migrated at each stage. We migrate the set with the lowest cost first. After each stage, we remove the migrated nodes from the remaining sets and recompute the cost for the migrating each remaining set. Again, we choose the set with the lowest cost to be moved next.

Algorithm 2 MIS-Size Sequence (MIS-SS)

```
1: function MIS-SS($G = (V, P)$, $P = (R, E)$, $m_0$, $m_f$)
2:     $MSets[\,] = FindMIS(G = (V, P))$ ▷ Call the MIS algorithm to generate all maximal independent sets
3:     $S_0 = MaxSize(MSets[\,])$ ▷ First migrate the largest maximal independent set
4:     for $i = 1 \to |MSets[\,]| - 1$ do
5:         Remove $\forall v \in S_0 \dots S_{i-1}$ from all $MSets[\,]$ ▷ Remove the migrated nodes from $MSets[]$
6:         if $MaxSize(MSets[\,]) = \emptyset$ then
7:             break
8:         end if
9:         $S_i = MaxSize(MSets[])$ ▷ Migrate the set with the most number of nodes left. If there are equal number of nodes in more than one sets, pick one set randomly.
10:    end for
11:    return $S$
12: end function
```

Algorithm 3 MIS-Local Minimum Cost First (MIS-LMCF)

```
1: function MIS-LMCF($G = (V, P)$, $P = (R, E)$, $m_0$, $m_f$)
2:     $MSets[] = FindMIS(G = (V, P))$ ▷ Call the MIS algorithm to generate all maximal independent sets
3:     for $i = 0 \to |MSets[\,]| - 1$ do
4:         $j = 0$
5:         Remove $\forall v \in S_0 \dots S_{i-1}$ from all $MSets[\,]$ ▷ Remove the migrated nodes from $MSets[]$
6:         if $MaxSize(MSets[\,]) = \emptyset$ then
7:             break
8:         end if
9:         for $set \in MSets[\,] - \{S_0, \dots S_{i-1}\}$ do
10:            $S_i = set$
11:            $cost[set] = c^m(i)\ when\ S_i = set$ ▷ Find the cost for the set of nodes to be migrated at stage $i$
12:            if $j = 0$ then
13:                $minset = set$
14:                $mincost = cost[set]$
15:            else if $cost[set] < mincost$ then
16:                $minset = set$
17:                $mincost = cost[set]$
18:            end if
19:        end for
20:        $S_i = minset$ ▷ Return the set of nodes with minimum cost to move at stage $i$
21:    end for
22:    return $S$
23: end function
```

5. EVALUATIONS

We use simulation experiments to evaluate the migration algorithms. We are interested first in a set of basic questions about algorithm performance relative to optimal, relative to worst case, and relative to one another. Specifically, (Q1) how important is it to have an intelligent migration algorithm? (Q2) how close do the proposed algorithms come to optimal? (Q3) how do the algorithms compare to one another? We next turn to the influence of two characteristics of the virtual and physical networks that our intuition suggests play a role in algorithm performance. (Q4) What is the influence on cost of migration distance, i.e., how far apart in the physical network are the original and final virtual network locations? (Q5) What is the influence of virtual network node degree on the time and cost of migration? We use these results to make some preliminary recommendations about the operation of networks that support migration.

5.1 Simulation Setup

We evaluate the algorithms with 50 different 10-node virtual networks and a 100-node physical network generated by GT-ITM [11] with the transit-stub model. The 10-node virtual networks are randomly generated by the Python NetworkX package [3] with different node degree distributions. Table 1 shows the average node degrees and diameters of the 100-node physical network and five of the 10-node virtual networks. These five virtual networks are chosen to have a wide range of average node degrees and we focus on them for some of the experimentation.

Table 1: Five of the Virtual Networks in the simulations

Network	Average Node Degree	Diameter
Physical (100-node)	3.7	11
Virtual 1	2.4	7
Virtual 2	2.8	5
Virtual 3	3.2	3
Virtual 4	3.6	3
Virtual 5	5.4	2

In our simulations, each virtual network is mapped to different initial and final mappings on the physical network. These mappings are randomly chosen with the constraint that there are no physical nodes in common in the initial and final mappings. We simulate the migration sequences using the three algorithms. We also exhaustively try every one-node-at-a-time move sequence. We calculate the cost and time for each migration sequence generated by the exhaustive search and the three algorithms.

In our simulations, for the preparation part ($t^p = f(S_i, \alpha)$), we use the control plane transfer time from [16]. We assume that each of the virtual routers on the 10-node virtual networks has less than 1k routes. The memory copy time is around 1 second with 10k routes. The FIB repopulating time of the software data plane is 2.1 seconds for 1k routes. Thus, $\alpha = 1 + 2.1 = 3.1$ *seconds*. We experiment with two choices for the function $f(S_i, \alpha)$. In the first case, we assume we will migrate one control plane at a time, hence $f_1(S_i, \alpha) = \alpha|S_i|$. In the second, we assume that we can migrate all control planes in parallel with no bandwidth interference, hence $f_2(S_i, \alpha) = \alpha$.

For the time in the migration part, we need to model the longest finishing time of any flow currently traversing the router(s) that have moved. This is not easy to estimate, since in general it depends on the uses of the virtual network, the capacities of the physical and virtual links, and the degree of sharing among flows and virtual networks. Further, if there are very long running flows, that might lead to policy decisions to avoid migration altogether or to disrupt these flows so that migration can be accomplished more quickly. In the absence of specific information, and consistent with our desire to use values derived from [16], we use the longest control plane transfer time as our longest flow finishing time. Figure 8 in [16] shows a time of about 3 seconds to transfer a 124 MB control plane. We choose to use $t^m = 3.5 seconds$, providing a bit of cushion above the 3 second measurement. In future work, we will examine better models for flow finishing time, as well as adaptations to the migration process when some flows are very long.

5.2 Baseline Results

We start with questions Q1-Q3 regarding basic algorithm performance, starting with one-node-at-a-time migration sequences. While a special case of muliple nodes at a time, one node at a time may have advantages in simplicity of operation. Table 2 shows the optimal and worst case costs of all possible one node at a time migration sequences for virtual networks (VN) 1-5, as well as the ratio of worst cost to optimal cost. As is clear in the table, the ratio of worst cost to optimal cost can be substantial, exceeding 2.5 for all five networks and as high as 3.37 for VN 1, which has lowest average node degree and highest diameter. We conclude from this experiment that there is value to investigating migration algorithms that can achieve cost that drives towards optimal.

We next compare the results from our greedy LMCF algorithm with the best cost. The rightmost column of Table 2 shows that LMCF does very well, finding migration sequences with cost close to the optimal solution on all five virtual networks. Without an exhaustive search on all $10! = 3.6M$ combinations of migration sequences, LMCF can give a migration sequence with cost between

Table 2: The costs of the best cases, the worst cases and LMCF of virtual network (VN) 1, 2, 3, 4 and 5 (move one-node-at-a-time sequences) with $t^p = f_1(S_i, \alpha = 3.1)$ sec and $t^m(i) = 3.5$ sec

VN	Optimal S^*	Worst	Ratio	LMCF
1	890.42	3000.64	3.37	890.42
2	1116.06	3263.55	2.92	1179.68
3	1413.76	3585.74	2.53	1635.49
4	1739.87	4864.23	2.80	1897.57
5	2751.34	6984.34	2.54	2985.42

Table 3: Results of MIS-SS, MIS-LMCF and LMCF with five of the 10-node VNs with $t^p = f_1(S_i, \alpha = 3.1) = |S_i|\alpha$ (the best costs are underlined)

	MIS-SS		MIS-LMCF		LMCF	
VN	Cost	Time	Cost	Time	Cost	Time
1	1087.42	44.88	1093.18	44.88	<u>890.42</u>	65.88
2	1415.38	41.38	<u>1162.36</u>	48.38	1179.68	65.88
3	1714.10	44.88	<u>1402.41</u>	51.88	1635.49	65.88
4	1603.54	44.88	<u>1482.29</u>	44.88	1897.57	65.88
5	<u>1710.82</u>	44.88	3335.73	44.88	2985.42	65.88

1 and 1.16 times the optimal migration sequence.

We next turn to the performance of the two algorithms that can move multiple nodes at a time. Recall that MIS-SS attempts to minimize time by moving as many nodes at a time in each stage and hence reducing the total number of stages. MIS-LMCF attempts to minimize a combination of cost and time by moving large sets but choosing which set to move based on cost considerations. Both algorithms rely on the MIS algorithm to generate candidate migration sets. It is computationally prohibitive to exhaustively try all possible multiple-node-at-a-time sequences, hence we focus our attention on the comparison of these algorithms to one another and then to the LMCF algorithm.

Table 3 shows the results assuming one control plane is migrated at a time, while Table 4 shows the results assuming all control planes in a stage can migrate together without interference. In each table, the lowest cost solution is underlined.

The goal of the MIS-SS algorithm is to minimize the migration time. It successfully achieves the lowest migration time of any of the three algorithms on each virtual network, and it improves on the LMCF algorithm by about 1.5 times when the control planes move one at a time and by more than 2x when the control planes move in parallel. If migration time is the key consideration, the MIS-SS algorithm provides a good solution.

The goal for MIS-LMCF is also to reduce the number of stages but to do so taking cost into consideration. The MIS-LMCF algorithm successfully achieves relatively low migration time as compared to the LMCF algorithm, and frequently matches the time performance of MIS-SS. Turning to cost, the MIS-LMCF algorithm achieves the lowest cost more often than MIS-SS or LMCF, however it does not always have lowest cost. For example, looking at Table 3, for VN 1, the lowest cost is achieved by the LMCF algorithm rather than either of the MIS algorithms, while for VN 5 the lowest cost is achieved by MIS-SS. If migration cost is the key consideration, it is important to understand whether control planes can move in parallel with limited interference. If they can, then one of the MIS-based algorithms will improve cost. If they cannot, then the MIS algorithms generally offer only modest improvement over LMCF, though there is a notable exception for VN 5 where LMCF cost is twice the cost of the MIS-based algorithms.

Table 4: Results of MIS-SS, MIS-LMCF and LMCF with five of the 10-node virtual networks with $t^p = f_2(S_i, \alpha = 3.1) = \alpha$ (the best costs are underlined)

VN	MIS-SS Cost	Time	MIS-LMCF Cost	Time
1	797.15	29.44	<u>657.78</u>	29.44
2	881.15	25.94	<u>872.09</u>	32.94
3	<u>1102.67</u>	32.53	1105.96	39.53
4	1090.94	32.53	<u>1056.14</u>	32.53
5	1710.82	26.35	<u>1501.46</u>	26.35

5.3 Effect of Migration Distance

We now dig deeper into the effect of virtual and physical network characteristics on the performance of the algorithms. In particular, we examine the effect of migration distance and then the effect of virtual network node degree (Q4). Migration distance is determined by the initial and final mappings of the virtual network to the physical network. We use virtual network 2 and simulate 90 migrations with 10 different mappings on the 100-node network. For each pair of mappings m_0 and m_f, we calculate the average migration distance by averaging the physical path lengths of the virtual node locations between $m_0(v)$ and $m_f(v)$. For our virtual network locations and physical network characteristics, we see average migration distances in the range of roughly 3.5 to 6.5 physical node hops. Recall that the physical network has diameter 11, so these distances are consistent with that constraint on the maximum distance a node can move.

Our model for time is based on the time to move one control plane, the interference between control plane moves, and the time to complete the longest outstanding flow. None of these depend on migration distance in our current model, though a refinement of the model might take into consideration that longer distance moves may take more time and that distance might influence how much interference exists between control plane moves.

We expect cost to be influenced by migration distance, since overhead links must traverse the physical network between original node locations to final node locations. In Figure 7, we show the costs of the migration sequences from LMCF, MIS-SS and MIS-LMCF as a function of the average migration distance. The LMCF plot contains 90 points, one for each initial-final pair, while the MIS-SS and MIS-LMCF plots contain two sets of 90 points, one for each model of control plane interference ($f_1(S_i, \alpha) = \alpha|S_i|$ and $f_2(S_i, \alpha) = \alpha$). In all three cases, the cost increases with average migration distance, as expected. The MIS-SS and MIS-LMCF algorithms tend to achieve comparable cost that improves noticeably over LMCF when the control planes can move in parallel. The impact of higher average migration distance is somewhat modest—the highest migration distance settings have cost approximately 1.4x the lowest migration distances. The cost advantage of MIS-LMCF is more pronounced when control plane moves can interfere than when they cannot, and this persists across migration distances.

5.4 Effect of Node Degree

Finally we turn to Q5, the question of the impact of virtual network node degree on the time and cost performance of migration. In our initial experiments, we observed variability in cost across the five virtual networks, with cost growing as virtual network node degree increased. Higher node degree means more neighbors, with two cost effects. First, more dense networks have more smaller size independent sets, hence more stages and more time is required. Second, more dense networks require more overhead links to connect nodes that have moved to neighbors that have not. Both effects will increase cost. We saw very little variability in time as average node degree increased, however we had only one virtual network representative for each average node degree value.

To explore more deeply, we use all 50 different 10-node virtual networks. We migrate each from and to the same initial and final mappings so as to eliminate the effect of migration distance. We run each algorithm and compare the time for the two MIS-based algorithms and the cost for all three algorithms. Figure 8 shows the migration time of MIS-SS and MIS-LMCF. With increasing average node degree, fewer nodes can be grouped into the same migration set, thus the time for migration increases with the average node degree, albeit slowly. Overall, MIS-SS achieves its goal to minimize the migration time for networks with different average node degrees. However, because the time is dominated by the migration sets, both algorithms give similar resulting migration times. For both algorithms, the time grows more slowly when the control planes interfere than when they do not.

Figure 9 shows the costs of sequences from LMCF, MIS-SS, and MIS-LMCF. As expected, an increase in average node degree has a substantial effect on the cost. The LMCF algorithm is most significantly affected by the increase in average node degree, since there are more overhead links and they must remain in place until the affected nodes have been moved. The MIS-based algorithms also see increases in cost with average node degree, with slightly slower cost growth when control planes do not interfere. In the limit, when the virtual network is a full mesh, the MIS-based algorithms must move single nodes at a time and hence cannot do any better than the LMCF algorithm.

5.5 Summary

From our simulations, all algorithms perform reasonably well. They all give migration sequences with reasonable costs and time. We also observe that no one algorithm consistently exhibits the lowest cost across all the average distance and node degree cases considered in our experiments. With the MIS algorithms, they both perform well in saving cost and time.

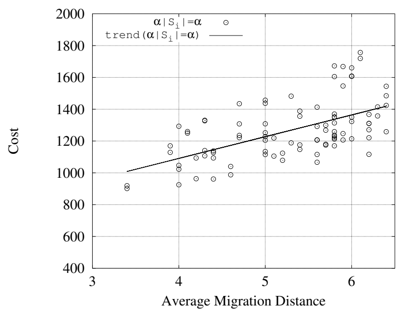
6. CONCLUSION AND FUTURE WORK

We believe our work makes the application of remapping policies in virtual networks possible, especially in architectures where virtual networks require reconfiguration. Our work considers the problem of the mechanism to migrate an entire virtual network from its initial mapping to its final mapping. We design scheduling algorithms for virtual network migrations that minimize the disruption to the current data traffic and the overhead traffic in the migration process.

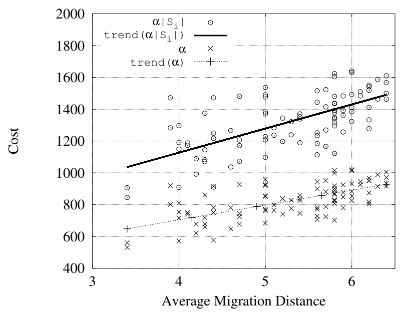
We propose three algorithms, LMCF, MIS-SS, and MIS-LMCF to discover the virtual router migration sequences for migrating an entire virtual network with lower overhead (cost) and shorter time. The LMCF algorithm produces a move-one-node-at-a-time schedule that exhibits cost close to the optimal solution. We then consider move-multiple-nodes-at-a-time schedules to minimize the time and the cost of migration. Our simulations of the MIS-SS and MIS-LMCF algorithms show that move-multiple-nodes-at-a-time schedules effectively minimize the time and cost.

Currently, we focus on the scheduling of virtual router migration based on the cost of the overhead links and the time of migration. Future work involves the introduction of another metric, data loss for modeling the time to wait to remove the old forwarding links. If data loss is allowed in the migration, we can migrate the network in a shorter time.

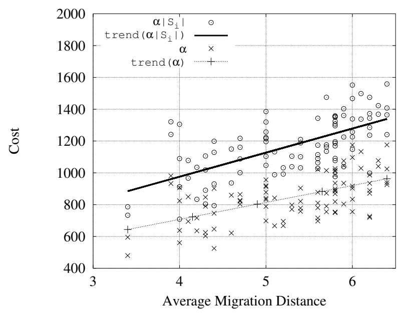
Another extension to our work is the scheduling of



(a) LMCF

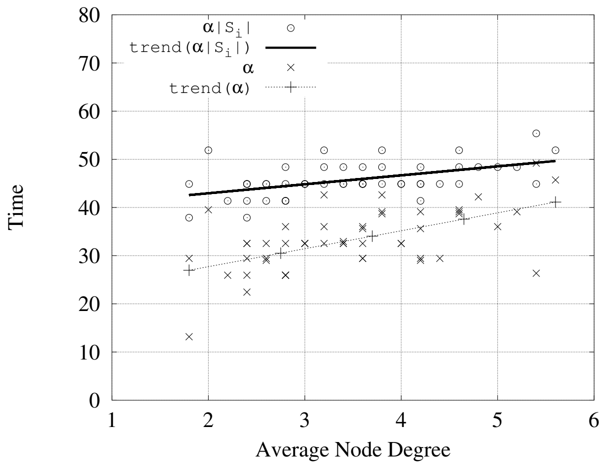


(b) MIS-SS

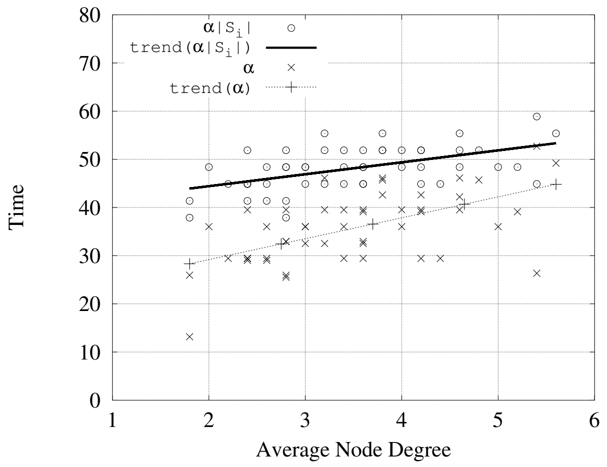


(c) MIS-LMCF

Figure 7: Cost for migrating VN 2 among 10 different initial and final mappings (90 migrations) with algorithm LMCF, MIS-SS and MIS-LMCF

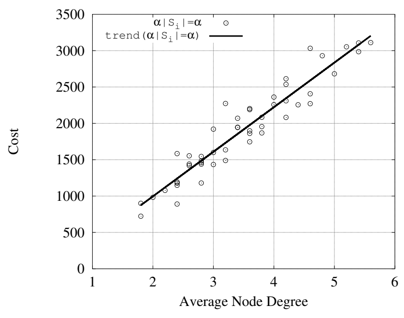


(a) MIS-SS

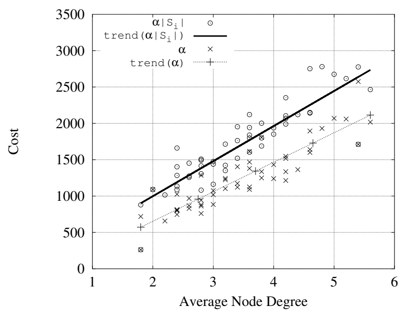


(b) MIS-LMCF

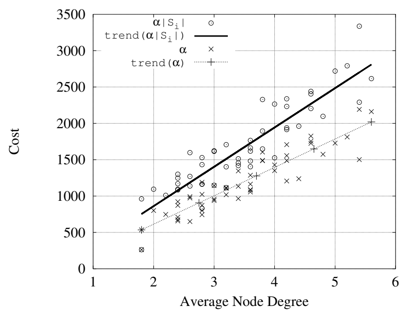
Figure 8: Time of migrating 50 different 10-node VNs on the same initial and final mappings with algorithm MIS-SS and MIS-LMCF



(a) LMCF



(b) MIS-SS



(c) MIS-LMCF

Figure 9: Cost of migrating 50 different 10-node VNs on the same initial and final mappings with algorithm LMCF, MIS-SS and MIS-LMCF

the control plane migrations in each stage such that the total time of virtual network migration is minimized. Currently we model the time of migration as moving the control planes one at a time or all at once. In practical systems, the control plane migration traffic may share the same bottleneck, interrupting the data plane traffic. Thus, the control plane migration scheduling should be considered as another problem in virtual network migration.

The interactions among multiple virtual network migrations have not been studied in this paper. In future work, the scheduling problem of migrating multiple virtual networks that share the same substrate becomes important. The scheduling problem of multiple virtual network migrations involves the dynamic of migrating multiple virtual routers belong to different virtual networks. When multiple virtual networks are migrating at the same time, some of the physical links and physical routers are fully occupied and may exceed the capacities of the links and physical routers. As a result, scheduling multiple virtual network migration is more complicated than a single virtual network migration.

Our work further raises the questions of how we design the applications and system architectures for virtual networks. A possible application of virtual network migration is defending attack traffic on virtual networks. Future work on extending our virtual network migration scheduling to other technologies such as achitecture with FlowVisor [1] and OpenFlow enabled switch networks [14] will be very useful as well.

7. REFERENCES

[1] FlowVisor. `http://www.openflowswitch.org/wk/index.php/FlowVisor`.

[2] Maximal independent set algorithm implementation. `https://networkx.lanl.gov/trac/ticket/282`. URL retrieved Apr 2012.

[3] Networkx: A python language software package. `http://networkx.lanl.gov/`. URL retrieved Apr 2012.

[4] E. Al-Shaer. Toward network configuration randomization for moving target defense. In *Moving Target Defense*, volume 54 of *Advances in Information Security*, pages 153–159. Springer New York, 2011.

[5] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. 18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 131–145, Banff, Canada, Oct. 2001.

[6] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.

[7] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proc. 5th USENIX OSDI*, Boston, MA, Dec. 2002.

[8] M. Demirci and M. Ammar. Fair allocation of substrate resources among multiple overlay networks. *International Symposium on Modeling, Analysis, and Simulation of Computer Systems*, 0:121–130, 2010.

[9] J. Fan and M. H. Ammar. Dynamic topology configuration in service overlay networks: A study of reconfiguration policies. In *Proc. IEEE INFOCOM*, 2006.

[10] GENI: Global Environment for Network Innovations. `http://www.geni.net/`.

[11] GT-ITM. `http://www.cc.gatech.edu/projects/gtitm/`.

[12] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *Proc. ACM SIGCOMM*, pages 61–72, Pittsburgh, PA, Aug. 2002.

[13] J. Lu and J. Turner. Efficient mapping of virtual networks onto a shared substrate. *Department of Computer Science and Engineering Washington University in St Louis Technical Report WUCSE2006*, 35(2006-35):1–11, 2006.

[14] OpenFlow Switch Consortium. `http://www.openflowswitch.org/`, 2008.

[15] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz. Overqos: offering internet qos using overlays. *SIGCOMM Comput. Commun. Rev.*, 33:11–16, January 2003.

[16] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford. Virtual routers on the move: Live router migration as a network-management primitive. In *Proc. ACM SIGCOMM*, Seattle, WA, Aug. 2008.

[17] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Computer Communications Review*, Apr. 2008.

[18] Y. Zhu and M. Ammar. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. In *Proc. IEEE INFOCOM*, Barcelona, Spain, Mar. 2006.