

TEMPORAL BANDWIDTH-INTENSIVE VIRTUAL NETWORK ALLOCATION
OPTIMIZATION IN DATA CENTERS

A Thesis
in
Computer Science

Presented to the Faculty of the University
of Missouri–Kansas City in partial fulfillment of
the requirements for the degree

MASTER OF SCIENCE

by
MATTHEW OWENS

B. A., University of Kansas, Lawrence, KS USA, 2010

Kansas City, Missouri
2014

© 2014
MATTHEW OWENS
ALL RIGHTS RESERVED

TEMPORAL BANDWIDTH-INTENSIVE VIRTUAL NETWORK ALLOCATION
OPTIMIZATION IN DATA CENTERS

Matthew Owens, Candidate for the Master of Science Degree
University of Missouri–Kansas City, 2014

ABSTRACT

In this paper, we consider bandwidth-intensive services for customers that want virtual networks (VN) in a data center environment. In particular, we consider this problem in a temporal context where bandwidth-intensive requests from each VN may arrive randomly at a review point, which may last for a certain duration. Thus, at each review point, the data center network provider must optimally allocate resources for the demand requests. For this problem, we present a mixed-integer programming (MIP) problem formulation where any request from a VN customer may be assigned to any virtual machine so that network resource availability is optimized. We present an overbooking strategy that may be employed to allow for some demands not met in the first try. For comparison, we also consider a base case where the allocation is pinned to a specific destination. Through our study, we show the comparative gains of different schemes.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering, have examined a thesis titled “Temporal Bandwidth-Intensive Virtual Network Allocation Optimization in Data Centers,” presented by Matthew Owens, candidate for the Master of Science degree, and hereby certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Deep Medhi, Ph.D., Committee Chair
Department of Computer Science & Electrical Engineering

Baek-Young Choi, Ph.D.
Department of Computer Science & Electrical Engineering

Kenneth Mitchell, Ph.D.
Department of Computer Science & Electrical Engineering

CONTENTS

ABSTRACT	iii
ILLUSTRATIONS	vi
TABLES	vii
ACKNOWLEDGEMENTS	viii
Chapter	
1 Introduction	1
2 Literature Survey	4
3 Model	6
3.1 Strategy and an Algorithmic Framework	9
4 Results	12
4.1 Minimization Heuristic Results	15
4.2 Dynamic Destination Compared with Selection Model	22
5 Conclusion	25
A Node Numbering	27
Appendix	
B NetworkSimulation	29
REFERENCE LIST	30
VITA	31

ILLUSTRATIONS

Figure		Page
1	High level architecture	13
2	4 Head Topology Demands Blocked	16
3	8 Head Topology Demands Blocked	17
4	4 Head Topology Time Periods Blocked	18
5	8 Head Topology Time Periods Blocked	19
6	4 Head Topology Nodes Added	20
7	4 Head Topology Nodes Added	21
8	4 Head Topology Time Peroid Blocked with Selection Model	23
9	4 Head Topology Demands Blocked with Selection Model	24
10	Node Numbering	28

TABLES

Tables		Page
1	Notations in Formulation	7
2	Notations used in Algorithm 1	9

ACKNOWLEDGEMENTS

I would like to thank my academic advisor Deep Medhi, Ph.D.

I would like to thank my parents, David and Helen, for making this possible.

This work has been partially supported by the National Science Foundation Grant
CNS-0916505.

CHAPTER 1

INTRODUCTION

Data center networking has seen tremendous growth in recent years with many services now being offered through data centers. Also, many different uses have been found for data centers for different customers. One such use for data centers is the ability to provide virtualized network (VN) services for different customers, especially where new requests on demand arrive, and for which network resources are needed to be required at review points. The need for such virtual networks in a data center environment is necessitated by customers wanting full control and view of their own network; secondly, from security and privacy aspects, such a logical separation of data center resources is also necessary for certain customers. Furthermore, customers may request resources for a virtual network in two different phases: 1) one where they want a VN to be initially created with certain basic requirements for a customer, 2) request resources to be added to their VN based on their own internal demand.

In this setting, we address the problem faced by a data center network provider (DCNP) to allocate resources to different VNs on demand for bandwidth-intensive services.

From the perspective of a DCNP, the problem is to ensure that it can provide both the host and network resources to their customers. However, a DCNP may have some flexibility on resource allocation based on service level agreements. For example, while

maintaining bandwidth guarantee for customers, a DCNP may allow overbooking of its resources by a certain amount. This translates to less resources for its customers as long as the overbooking is within a tolerable level permitted in the service level agreement. Thus, by *bandwidth-intensive* services, we refer to service that requires bandwidth guarantee, but may allow some overbooking. This may result in a number of usage policies in place, depending on what may be allowable. We consider this problem in a temporal context in that a resource request from a customer arrives at random to be fulfilled at a review point, while the customer uses this resource for a certain duration. Then, an arriving request at an entry point to the data center network may be assigned to any one of the virtual machines at the data center, based on the availability of both the VM resources and data center network resources. Thus, over a time horizon, we wish to understand network resource utilization based on policies in places. Furthermore, in a data center setting, a request for a resource may be directed from an entry point to the network and to *any* of the possible hosts that can serve the request, instead of being tied to a particular host, as long as this is allowable at the time of the virtual network creation. A DCNP certainly wants to take advantage of this flexibility in optimizing its resource allocation.

We present a mixed-integer programming (MIP) problem formulation from a traffic engineering perspective that optimizes the network resources by considering newly arriving requests at review points. We then present a strategy based on overbooking to understand the tradeoffs in allocating services over a time horizon compared to the based case. Thus, the contribution of this work is to 1) present the MIP model to be used at each review point for allocation of requests from each VN, 2) to discuss strategies and show

that they fit into an algorithmic framework with the optimization model, and 3) present tradeoff results on how the strategies impact service requests.

The rest of the thesis is organized as follows. After discussing related work in Chapter 2, we present our optimization model in Chapter 3.1. In Chapter 3.2, we present strategies, including consideration for overbooking for bandwidth-intensive services; we also present an algorithmic framework that incorporates the strategies along with the optimization model. Finally, in Chapter 4, we present our study to understand the tradeoffs due to strategies considered.

CHAPTER 2

LITERATURE SURVEY

Data center networking has been receiving a lot of attention in recent years. A number of early works on data center networking has focused on architectural design of data center networks [1–4, 6, 9] and their scalability and performance.

Data center traffic engineering for virtual machine placement and routing is considered in [7]. While we also take a traffic engineering perspective, we do not consider the virtual machine placement problem; rather, we focus on servicing bandwidth-intensive virtual network service in a temporal context. In [8], the authors considered traffic off-balancing from an energy perspective in data center networks.

The work that is closest to our work is [5]. While [5] considered virtual network allocation with bandwidth guarantee, on the surface, it looks as though we are addressing the same problem, there are a number of differences. Broadly, they focus their work from an architectural perspective while we focus our work from a traffic engineering perspective. While they provide a heuristic for their allocation problem, we present an explicit optimization formulation for traffic engineering since we consider the problem of allocation at each review point. Unlike their work, our work does not consider the issue of time taken for job migration to another virtual machine as the time context of such movement is typically in seconds while the review points for traffic engineering are in the order of several minutes. Furthermore, we factor in handling of overbooking with

dynamic allocation at review points, which was not addressed in [5].

CHAPTER 3

MODEL

We consider new request arrivals from customers to be random, for which the resource allocation is to be done at every review point. We assume the review point to be every τ minutes. Typically, τ is every five to fifteen minutes. The duration a new request stays on is assumed to be random. Note that since the data center is set to serve VN customers, at any time, there are existing VN tunnels setup for prior requests. Thus, any job that needs immediate access to resources, that is, jobs that cannot wait until the review points, are assumed to be served by existing VN channels for the customers (that were set up at earlier review points). Since such immediate jobs are served through existing VN tunnels, they are not modeled in our case. In other words, the scope of our work is to consider new requests at review points that are major requests requiring allocation of new bandwidths and virtual network tunnels.

Consider review point t (for clarity, notations are summarized in Table 1). We need to look at the number of new requests received from different VN customers at the entry point to the data center network, which are to be assigned through the network if resources are available. Thus, consider $d_i^v(t; \mu)$ to be the bandwidth demand for VN v entering at entry point i at time t for duration μ ; in general, μ is not known at the time of arrival of the request. If J_i^v is the set of hosts that can serve entry point i for virtual network v at review point t , then an arrival request can be assigned to one of the hosts. We

Table 1: Notations in Formulation

V :	Set of Virtual Network (VN) Customers
I :	Set of Entry points to the Data Center
$J_i^v(t)$:	Set of hosts associated with entry point $i \in I$ for VN $v \in V$ at review point t
$K(t)$:	Number of virtual machines at host j at review point t
L :	Set of links in the data center network
$P_{ij}^v(t)$:	Set of paths from entry point i to virtual host j for VN v at time t
$c_\ell(t)$:	Available capacity on link $\ell \in L$ at review point t in the data center network
$\delta_{ijp,\ell}^v(t)$:	Link-path indicator: 1 if path p for request at entry point i for virtual host j for VN v at review point t uses link ℓ ; 0 otherwise
Variables:	
$y_{ij}^v(t)$:	Binary decision variable of a request for VN v at entry point i is to be sent to virtual host j at review point t
$x_{ijp}^v(t)$:	Binary decision variable on if a request from i to j will use path p at review point t
$z_\ell^v(t)$:	bandwidth needed on link ℓ for VN v at review point t

can model this by defining a binary variable $y_{ij}^v(t)$ that satisfies the following requirement

$$\sum_{j \in J_i^v(t)} y_{ij}^v(t) = 1, \quad i \in I, v \in V. \quad (3.1)$$

If a particular host j is selected, then a path needs to be selected from entry point i to j from a list of possible paths through the data center network. This can shown as follows

$$\sum_{p \in P_{ij}^v(t)} x_{ijp}^v(t) = y_{ij}^v(t), \quad j \in J_i^v(t), i \in I, v \in V. \quad (3.2)$$

Note that since only one j is chosen for each request entering i for VN v at review point t , then all other y s will be set to zero, which will automatically ensure from the above equation that no path is chosen to unselected host j .

Bandwidth needed on link ℓ of the data center network for VN v to satisfy the

associated demand at review point t is given by

$$\sum_{i \in I} \sum_{j \in J_i^v(t)} \sum_{p \in P_{ij}^v(t)} \delta_{ijp,\ell}^v(t) x_{ijp}^v(t) d_i^v(t; \mu) = z_\ell^v(t), \quad (3.3)$$

$$\ell \in L, v \in V. \quad (3.4)$$

Since the total bandwidth available on link ℓ at review point t is c_ℓ , then the total bandwidth needed for all VNs must be satisfied, which can be expressed as:

$$\sum_{v \in V} z_\ell^v(t) \leq c_\ell(t), \quad \ell \in L. \quad (3.5)$$

In addition to the basic constraints listed above, there are possibly additional requirements. Each VN customer may impose that a minimum bandwidth is to be allocated to them on any of the links

$$z_\ell^v(t) \geq \varepsilon \quad (\text{where } \varepsilon \geq 0), \quad \ell \in L, v \in V. \quad (3.6)$$

Finally, each host j may have a limit of $K(t)$ virtual machines at review point t for all the connections

$$\sum_{v \in V} \sum_{i \in I} y_{ij}^v(t) \leq K(t). \quad (3.7)$$

We consider the objective of using the least number of network resources at current review point t . Thus, we consider the following minimization objective at time t .

$$\text{Minimize } \sum_{\ell \in L} \sum_{v \in V} z_\ell^v(t). \quad (3.8)$$

This objective is motivated by the fact that if the least amount of resources are used in the network at review point t , then it will be able to support as many requests as possible at the next review point. This is a mixed-integer linear programming model.

Table 2: Notations used in Algorithm 1

$D_{new}(t)$:	New demands at review point t
$D_{cont}(t)$:	Prior demand continuing to be served at review point t
$D_{left}(t)$:	Demand left (not allocated by the model) at review point t
$D(t)$:	Demand to be served at review point t
$C(t)$:	Capacity at review point t
$C_{left}(t)$:	Capacity left at review point t (after allocation)
$C_{OB}(t)$:	Available capacity due to $OB(t, \Delta)$ -strategy at review point t
$\hat{Z}(t)$:	Link Flow allocated at review point t by the model

3.1 Strategy and an Algorithmic Framework

We consider an overbooking strategy that may be applied over a time horizon. Through this considerations, we can understand how a strategy can make an impact on service delivery for VN services.

3.1.1 Overbooking Strategy

The overbooking strategy, the $OB(t; \Delta)$ -strategy, allows overbooking at review point t by a certain amount Δ if some demands are not met. A DCN provider may consider serving a request at review point t even if it does not have full resources. This is possible if a DCN provider is willing to overbook its resources by a certain amount, which may result in some degraded service perception for its customers. As long as such degradation is acceptable (within a tolerable level) or is supported under the service level agreements, a DCN provider may opt for overbooking at review point t .

Algorithm 1 Strategy Execution

```
1:  $t \leftarrow 0$ 
2: while  $t \leq T$  do
3:    $D(t) \leftarrow D_{new}(t) + D_{cont}(t)$ 
4:    $(\widehat{Z}(t), D_{left}(t)) \leftarrow \mathcal{M}(D(t), C(t), K(t))$ 
5:    $C_{left}(t) \leftarrow C(t) - \widehat{Z}(t)$ 
6:   if  $D_{left}(t) > 0$  then
7:     if  $OB(t; \Delta)$ -strategy then
8:        $C_{OB}(t) \leftarrow C_{left}(t) + \Delta$ 
9:        $(\widehat{Z}(t), D_{left}(t)) \leftarrow \mathcal{M}(D_{left}(t), C_{OB}(t), K(t))$ 
10:    end if
11:  end if
12:   $t \leftarrow t + 1$ 
13: end while
```

3.1.2 An Algorithmic Framework

We now present an algorithmic framework that incorporates the strategies along with the model presented in Section 3 (see Algorithm 1). This allows us to conduct a study of their effectiveness over a time horizon T . In describing this framework, we use a compact notation. For example, $C(t)$ refers to the capacity at review point t for all links without identifying each link, i.e., it uses a vector representation of capacity at review point t . Notations used in Algorithm 1 are summarized in Table 2.

In this framework, at review point t , the model described in Chapter 3, which we will now refer to as $\mathcal{M}(D(t), C(t), K(t))$, is solved. The output of the model is indicated as two pieces of information: $\widehat{Z}(t)$ for the link flow imposed by solving the model, and $D_{left}(t)$ is any demand that could not be satisfied by the model in its first try. Any demand not satisfied at review point t is then subject to consideration in the $OB(t; \Delta)$ -strategy for overbooking.

Note that a number of metrics after solving the model at a review point t are of interest such as the number of requests blocked. These measures are also collected and will be discussed later in Chapter 4.

In regard to employing the $OB(t; \Delta)$ -strategy, we take a two-phase approach. We first check if the currently available capacity can accommodate the demands. If this is not possible, we increment the capacity by Δ , and solve the model again. In other words, the increment for overbooking is not automatic; rather, it takes a conservative approach.

CHAPTER 4

RESULTS

Data center topologies typically have certain patterns. In Fig. 1, we show a DCN topology that we considered in our study. In this topology, there are four entry points (shown at the north end of the figure) for external customers to the data center network. They are served by any of the virtual machines, shown at the south end of the figure. Clearly, we can see that from any entry point, i , to a destination host j , there are a number of possible paths. In our case, we allow the destination to be randomly selected because the service provided by the end machines is the same.

In our study, we consider that there are four VN customers, where each VN customer has one or more random arrivals that may enter at any of the entry points at review point t . A request may stay on for a random duration of time, during which the path from i to j is enforced. In our study, we set $K(t) = 2$, since our primary goal is to understand how the options and strategies perform in the core of the data center network. The core of the optimization model is solved using CPLEX.

In addition to the two strategies, we consider a simplified version of $\mathcal{M}(D(t), C(t), K(t))$, in which the demand request arriving at an entry point is directly pinned to a destination j . We refer to this simplification as ‘fixed’ model, i.e.,

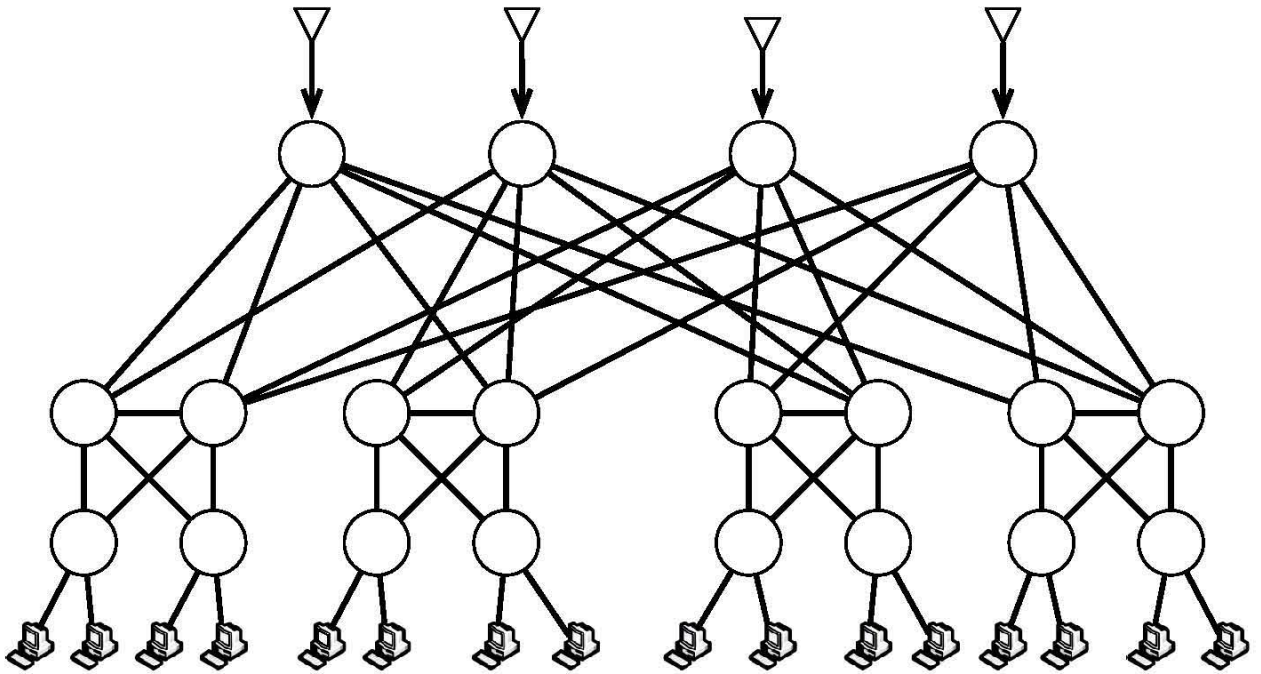


Figure 1: High level architecture

$\mathcal{M}_{fixed}(D(t), C(t), K(t))$. To distinguish between the two, $\mathcal{M}(D(t), C(t), K(t))$ is referred to as ‘dynamic’, i.e., $\mathcal{M}_{dynamic}(D(t), C(t), K(t))$. Thus, we take a two-cycle approach to solve the model. When a request is generated at review point t , it was first solved as $\mathcal{M}_{fixed}(D(t), C(t), K(t))$; this way, we now have a base case, which is static. If this does not result in an assignment for all requests, then a second cycle is done to assign unallocated requests randomly to another destination j' to consider the dynamic case, $\mathcal{M}_{dynamic}(D(t), C(t), K(t))$. Note that the dynamic case may not result in allocation for all unassigned requests (depending on the network capacity) – this is meant as a way to improve over the base case and helps us in our comparison. Altogether, we then consider four *situations*: base case (“Base”), dynamic case (“Dynamic Destination”), base case with overbooking (“Overbooking”), and dynamic with overbooking (“Dynamic Destination with Overbooking”). In this study, we set overbooking to allow 5% additional capacity over the base capacity.

For our study, we consider two topologies. In the first topology (shown in Fig. 1), there are four entry points, and it is thus labeled as a 4-head topology. The second topology (figure not shown here) has the exact same structure as the 4-head topology; the difference is that there are eight entry points instead (that is, an 8-head topology), and thus, it has double the nodes in the bottom parts also, compared with the 4-head topology.

4.1 Minimization Heuristic Results

We generated ten sets of demand randomly that have the same average arrival rate, average duration, and the average demand volume. The arrivals follow the Poisson process and the duration is assumed to be exponentially distributed. The results reported here is the average of the ten independent runs. In Fig. 2, we present the fraction of requests blocked as the network capacity (listed as the fixed capacity on each link) is increased. Naturally, as the capacity is increased, the blocking goes down. Secondly, if there is sufficient capacity in the network, there is little blocking with any of the situations. On the other hand, when the capacity is tight, we see a significant difference between the four situations. Specifically, at a capacity of a 50 units on each link, we see that compared to the base situation, overbooking reduces the average number of requests blocked by 11%, the dynamic destination reduces by 22%, and the dynamic destination with overbooking reduces blocking by 32%. Thus, the dynamic selection is a good option to use compared to the base situation, if no penalty is to be incurred for overbooking. On the other hand, if the penalty for overbooking is tolerable, then significant gain can be achieved by combining dynamic destination with overbooking. The observation is similar when we consider the 8-head topology (see Fig. 3).

Another measure we considered is the number of times during the entire simulation run, we have observed a request being blocked at any review point. The time fraction on review point blocking is labeled as time blocking (see Fig. ?? and Fig. 5). The general pattern of observation is similar to request blocking with the dynamic destination with overbooking having the least time blocking when the capacity is tight. However, when

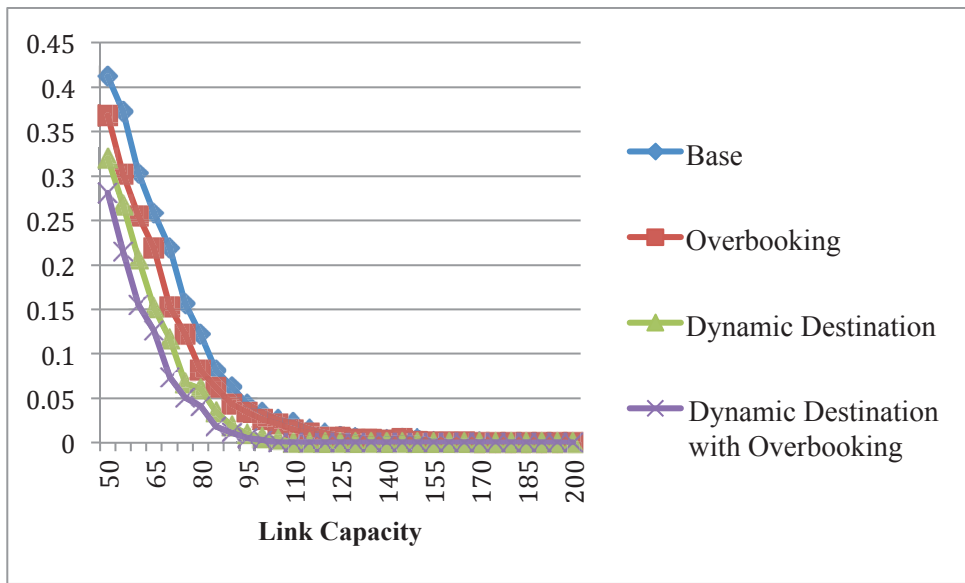


Figure 2: 4 Head Topology Demands Blocked

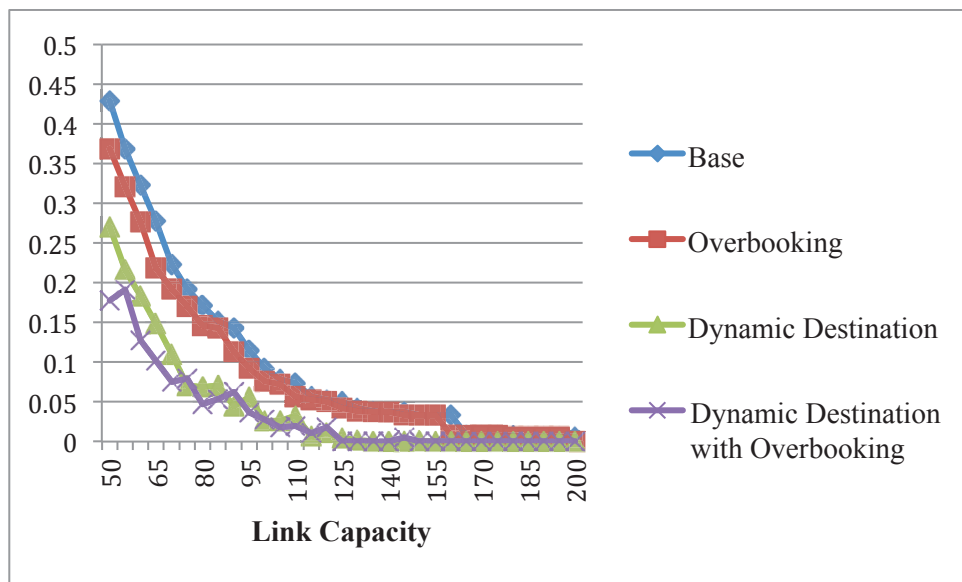


Figure 3: 8 Head Topology Demands Blocked

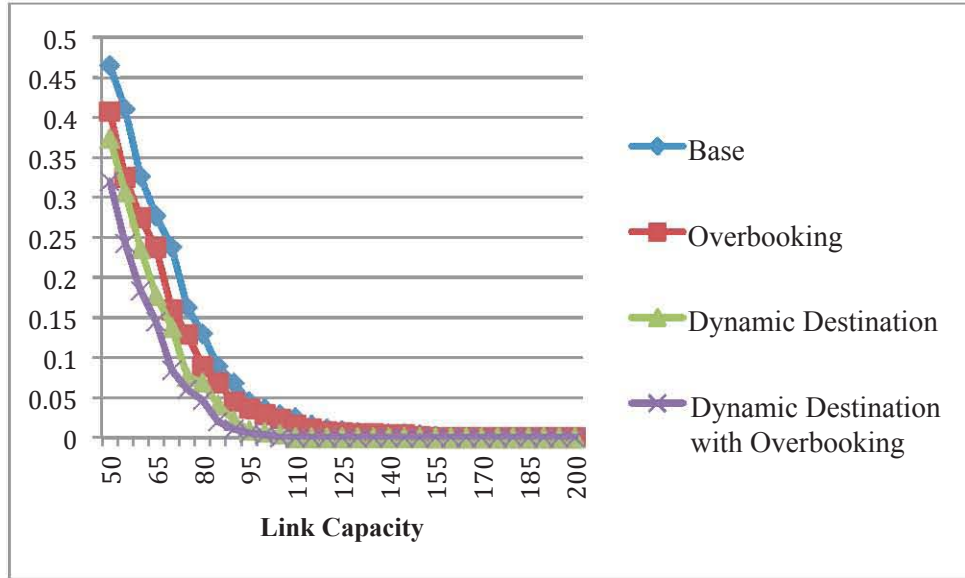


Figure 4: 4 Head Topology Time Periods Blocked

the capacity is not as tight, there are cases where we found the dynamic destination to be comparable to the dynamic destination with overbooking (the overlap is within the margin of error). That is, in a larger topology and with abundant capacity, there may not be the need to do overbooking all the time.

Finally, in Fig. 6 and Fig. 7, we show the opportunity to consider additional paths due to dynamic allocation to any destination compared to the static case. This is another way to see why we see a drop in request blocking with the dynamic case compared to the static case. In general, the number of additional paths considered did not appear to be

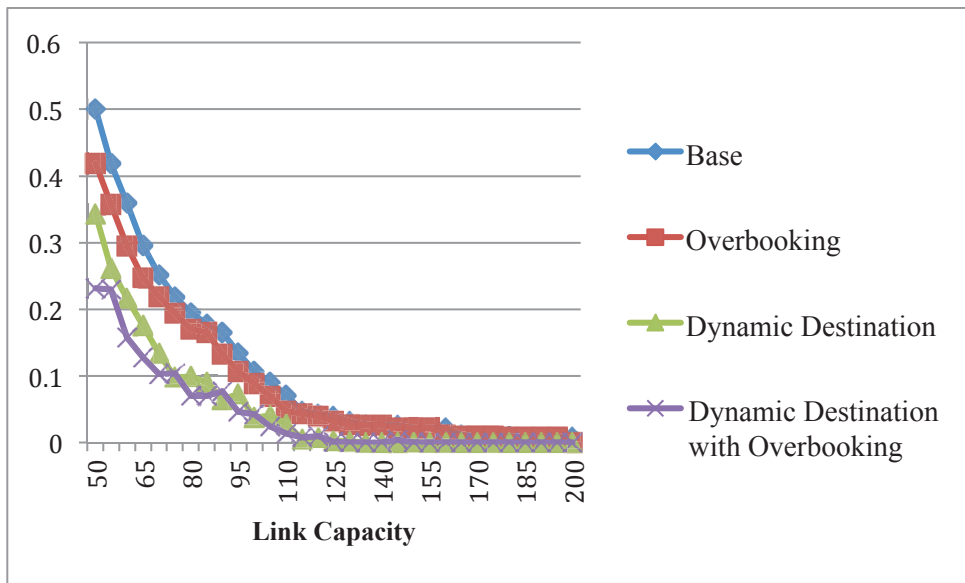


Figure 5: 8 Head Topology Time Periods Blocked

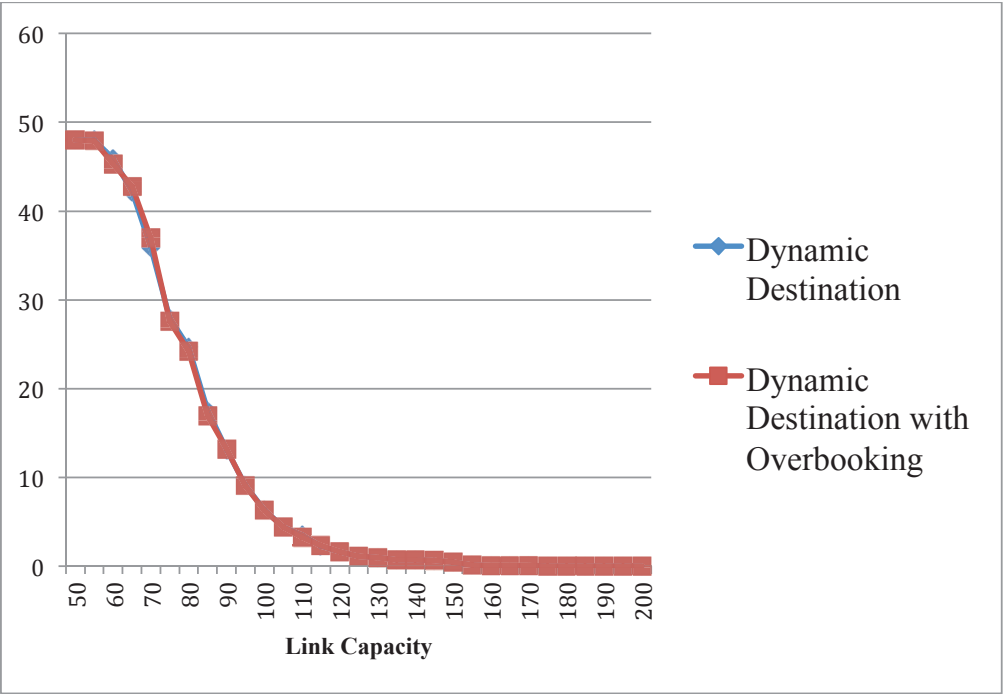


Figure 6: 4 Head Topology Nodes Added

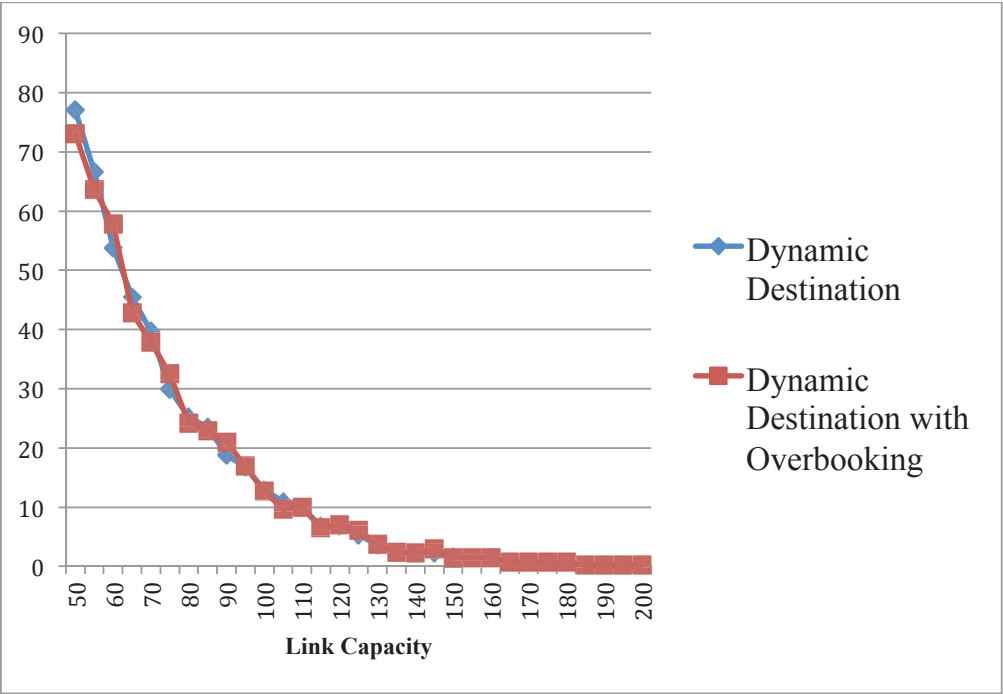


Figure 7: 4 Head Topology Nodes Added

any different between the dynamic situation and the dynamic situation with overbooking enforced.

4.2 Dynamic Destination Compared with Selection Model

In this study, we consider the differences between the Dynamic Destination heuristic and the optimization model outlined in chapter 3. We use the 4-head topology shown above. The virtual networks follow the same pattern shown above. The results reported here is the average of the ten independent runs. As above, the demand decreased as the capacity increase. When capacity is tight, we can see there is a difference between Fig. 8 and figure Fig. 9. While the capacity is 50, the dynamic destination heuristic and optimization model follow a similar pattern in figure Fig. 8, in figure Fig. 9 there is an 4% difference between the two. So while both networks are block a similar amount of times, the optimization model is better at placing the current demands.

Another difference is the optimization model is not adding any new path to the Virtual Network. It is performing better than the Dynamic Destination heuristic without using more paths. However after the capacity is 150 on the graph, the Dynamic Destination has less blocks. This may be because the capacity gains of adding another path become greater than balancing of the demands done by the selection model.

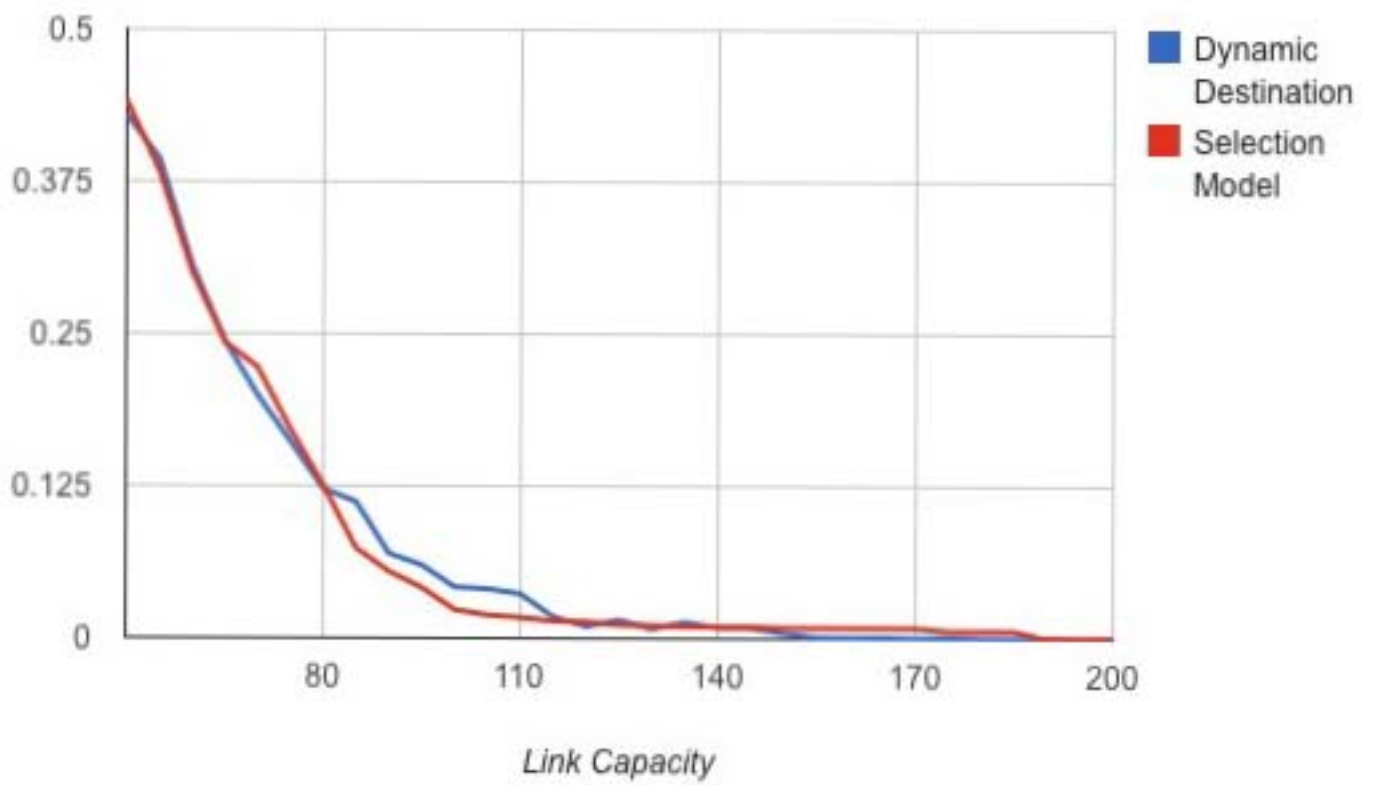


Figure 8: 4 Head Topology Time Peroid Blocked with Selection Model

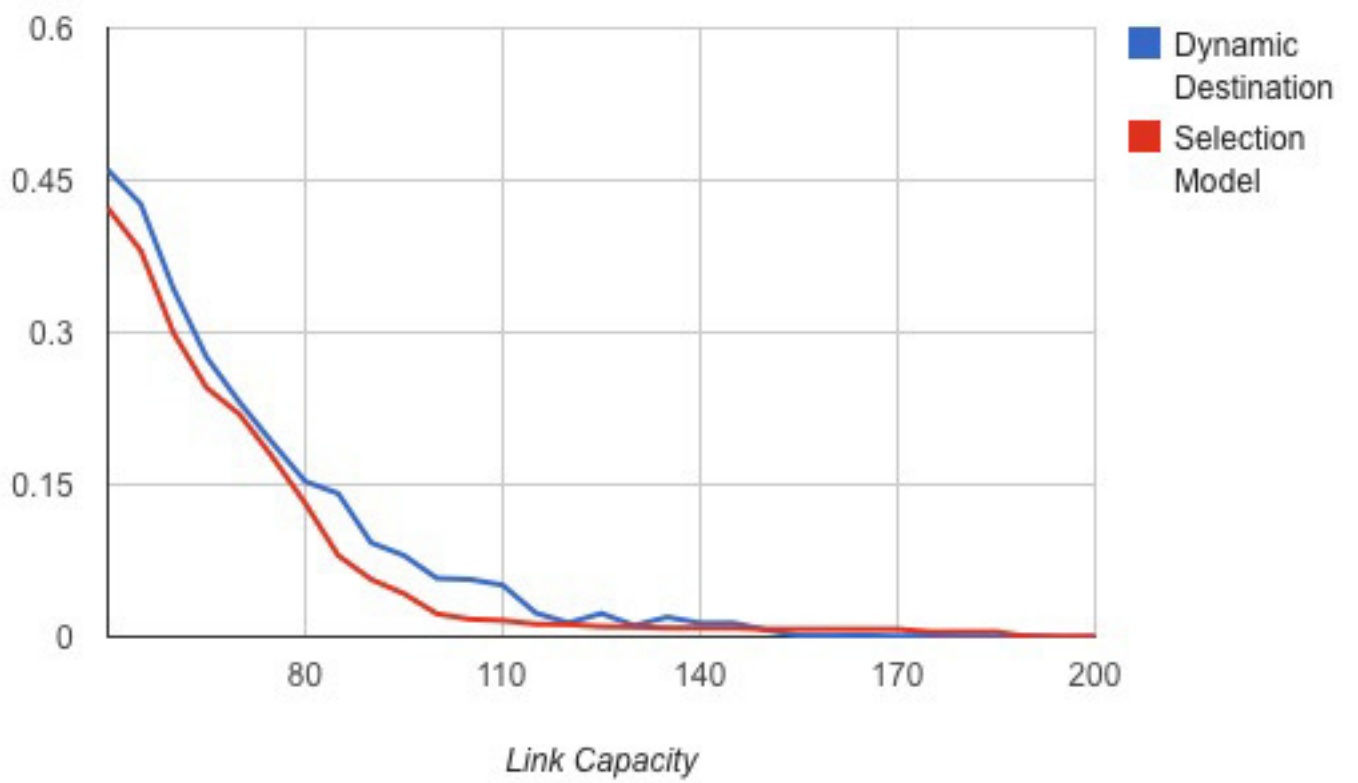


Figure 9: 4 Head Topology Demands Blocked with Selection Model

CHAPTER 5

CONCLUSION

As datacenter use continues to grow, looking at how demand travels through a network become more important. By looking at the demand on the network, the performance of the end machine can increase because of the bottleneck of the network. There are a few example like bank transactions that support this mode as mentioned earlier in the paper. With new uses of datacenters growing, network utilization will become more important.

Virtual Machine placement based on network capacity random helps utilize data-center resources. In bandwidth intensive datacenters, utilization of network resources is crucial to optimization. Most studies look at host capacity for VMs without looking at the network capacity. Changing placement of the virtual machines increase the ability for the utilization of network. A smaller datacenter network is able to support more hosts. By looking at the network capacity, it will decrease the round trip time during periods of hight demand.

This thesis looks at a network in the context of time. Demands come and go based on the demands of the VM in the network. Migrating the VM to other hosts in the data-center allows for greater utilization of the network while different customers experience different levels of demand. By looking at different time snapshots in time, the network can migrate the VM to different host in order to create this efficiencies. Time is important because networks face different demands at different times.

By making MIP allows us to see how this network behaves over the time. The MIP shows where the VM should be placed in a virtual network. The model is not optimizing the demand, but MIP shows where the VM should be placed in their respected virtual networks. This allows for datacenter providers to maintain QOS, and the ability to scale their services without having to upgrade their network in order to meet more demand.

The results show how the MIP problem work compared to random placement throughout the network. The results show that the difference between random placement of VMs to any host and in a virtual network created created for a datacenter client there is little difference between the two. By migrating virtual machines around a virtual network in order to meet demand on the basis of a network demand, it creates an efficient way to place VM on hosts in a network.

So in conclusion, by adjusting VM placement in a datacenter while maintaining QOS provided by Virtual Network, the datacenter network is able to support more demand. This provides greater utilization of resources through adjusting routing by placement rather than upgrading the capacity of the network. This shows how a novel routing technique can improve network performance by up to 10

APPENDIX A

NODE NUMBERING

The nodes are number as in Fig. 10

The entry points are nodes 1-4. The paths from the entry points to the machines are named with the this formula: $16(e_i - 1) + (vm_i - 20)$. e is the entry point node number. vm is the node number of the virtual machining. An example is e_2 to vm_{22} would be path x_{18} in the optimization model.

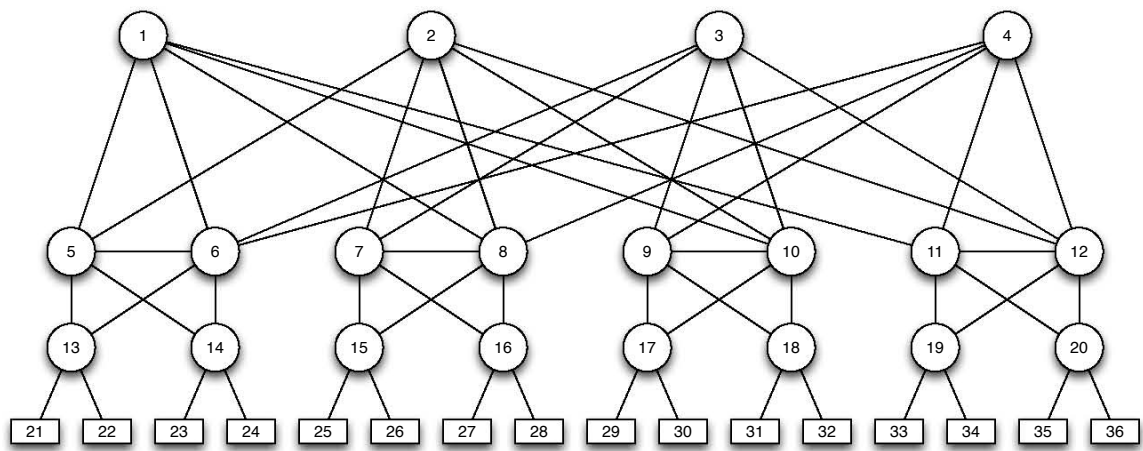


Figure 10: Node Numbering

APPENDIX B

NETWORKSIMULATION

The simulation is a python script which interfaces with IBM CPLEX Optimizer. The script generates the LP files and solves them based on the input to the program. The input is network topology, the list of paths, the demands, and the virtual networks. The code located in at <https://github.com/matao/NetworkSimulation>.

REFERENCE LIST

- [1] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. *Proc. of ACM SIGCOMM 2008*, Seattle, WA, USA, 2008, 63–74.
- [2] G. Alkmim, D. Batista, and N. da Fonseca. Approximated algorithms for mapping virtual networks on network substrates. *Proc. of IEEE International Conference on Communications (ICC2012)*, June 2012, 1460–1465.
- [3] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. *Proc. of ACM SIGCOMM 2009*, Barcelona, Spain, 2009, 51–62.
- [4] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: a high performance, server-centric network architecture for modular data centers. *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, 63–74, Aug. 2009.
- [5] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. Sec-ondNet: a data center network virtualization architecture with bandwidth guarantees. *Proc. of ACM CoNEXT 2010*, Philadelphia, PA, 2010, 15:1–15:12.
- [6] F. Hao, T. V. Lakshman, S. Mukherjee, and H. Song. Enhancing dynamic cloud-based services using network virtualization. *Proc. of 1st ACM workshop on Virtualized*

infrastructure systems and architectures, ser. VISA '09, Barcelona, Spain, 2009, 37–44.

- [7] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, Joint VM placement and routing for data center traffic engineering. *Proc. of IEEE INFOCOM 2012*, march 2012, 2876–2880.
- [8] C. Lee and J. Rhee. Traffic off–balancing algorithm: Toward energy proportional datacenter network. *OptoElectronics and Communications Conference (OECC)*, 2012 17th, july 2012, 409–410.
- [9] R. Niranjana Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, PortLand: a scalable fault–tolerant layer 2 data center network fabric. *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, 39–50, Aug. 2009.

VITA

Matthew Owens was born on March 31, 1989 in Kansas City, Missouri.

He graduated from University of Kansas in 2010 with a degree in the History of Art.