TRAFFIC PRIORITY MAXIMIZATION IN POLICY ENABLED TREE BASED DATA CENTERS

Alexander Ing
California State University Dominguez Hills
Department of Computer Science

Overview

- Introduction
- Related Works
- Problem model
- Problem statement
- Proposed solutions
- Results
- Conclusion / Future work

Introduction

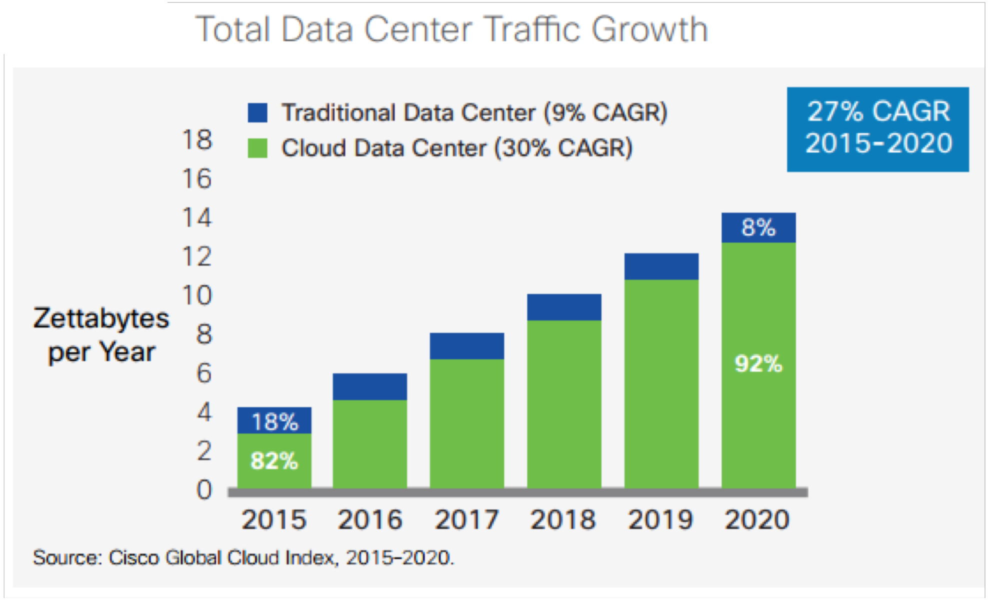
Introduction

Increase of internet activities

27% compound annual growth rate

Datacenter traffic estimated to increase by 10 zettabytes in 2018

SDN and NFV introduced to datacenters



Cisco Global Cloud Index: Forecast and Methodology, 2015-2020 White Paper

Introduction

Centralized controller with global view of whole network

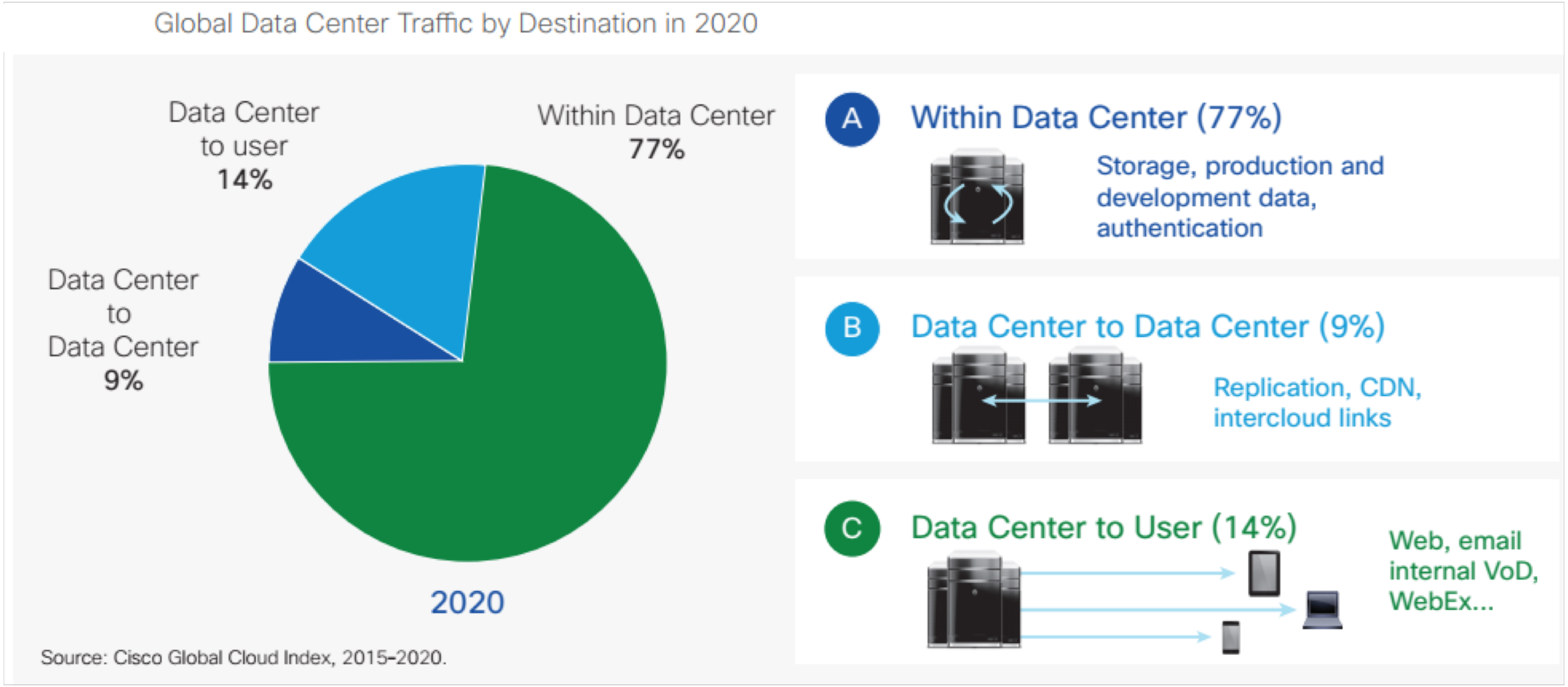
- Global view of network allows for optimization

Network function virtualization

- Functions provided by hardware can now be virtualized
- Virtual machines rented to customers
- Infrastructure as a service (IaaS)

44% of traffic will be supported by SDNs and NFV technology by 2020

Introduction



Cisco Global Cloud Index: Forecast and Methodology, 2015-2020 White Paper

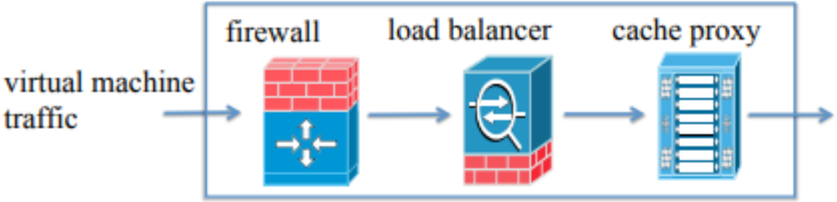
Introduction

Middleboxes

- Middlebox: Firewall, Web cache, Load Balancer, etc.
- Packets or data flow pass through middleboxes before reaching their destination

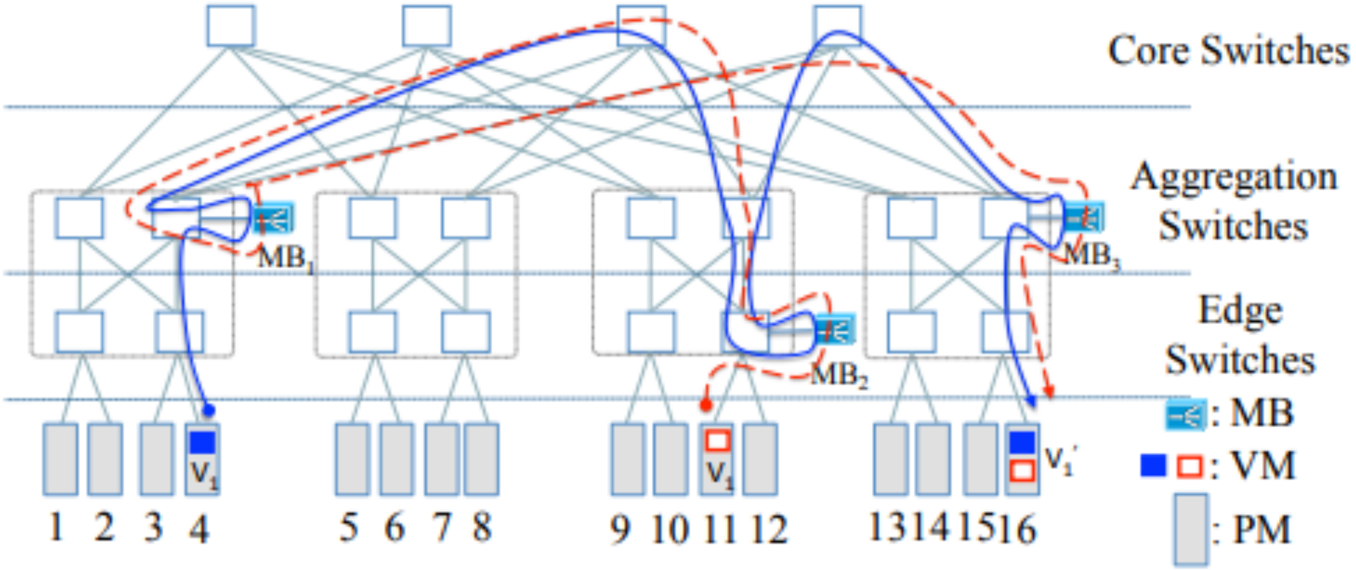
Policy Chain

- Ordered sequence of Middleboxes that packets must follow before reaching their destination
- Increased security, performance, etc.
- Creates longer communication paths



Introduction

Why is this a problem?



Related Work

Multi-commodity flow with in-network processing by Charikar et al.

- Traffic optimization in data center networks
- General graph topology
- Policy enforcement
- Virtual middleboxes placed on compute nodes

Multicut and Integer Multicommodity Flow in Trees by Vazirani

- Maximizing flows in tree graphs
- Proposed a 2 approximation algorithm
- Does not consider middleboxes or policy chain

Related Work

SIMPLE-fying middlebox policy enforcement using SDN By Qazi et al.

- Propose a SDN based traffic steering model
- Ensures enforcement of policy chains
- Does not consider flow optimization

Flowtags: Enforcing network-wide policies By Fayazbakhsh et al.

- Flows are tagged upon processing completion
- Ensure policy enforcement
- Does not take into consideration flow maximization

Problem Models

Datacenter Model

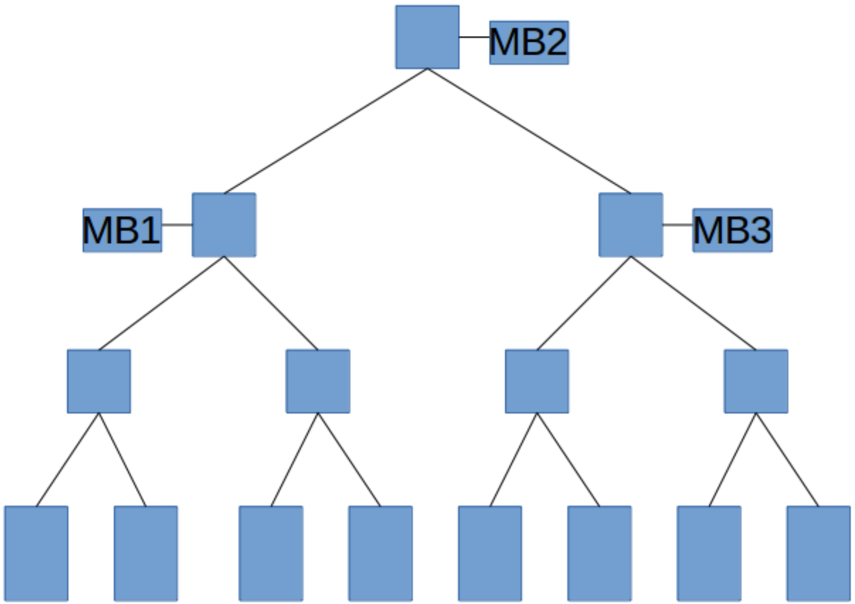
The data center is modeled as an undirected graph G (V, E)

Where $V = V_P \cup V_S$

Physical machines V_P and the network switches V_S

E is the set of edges

VM pairs placed on V_P which are the leaf nodes in tree data centers.



VM communication pair model

VM communication pair $P(V_i, V_i`)$ consist of a source VM $V_i$ and destination VM $V_i`$

Each pair has three properties: communication frequency, priority, and demand

Each pair $(V_i, V_i`)$ has the following:

- frequency F_i is a random number from [1, F]
- priority T_i is a random number from [1, T]
- demand D_i is a random number from [1, D]

Edge model

Each edge connects at most two vertices V

Each edge E in graph G has a capacity K to its available bandwidth

Edge (u,v) has capacity K(u,v), indicating the bandwidth available at (u,v)

The total communication demand D across an edge must not exceed capacity, thus $D \leq K$

Middlebox Model

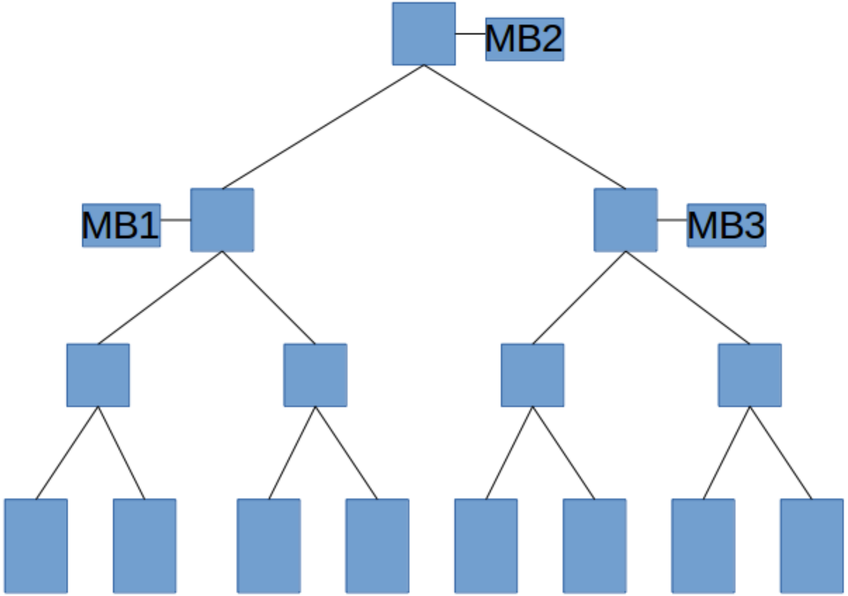
The set of middleboxes M are placed within the datacenter

$M = \{m_1 \ . \ . \ . \ m_i\}$

i = Total number of middleboxes placed, where $i \geq 2$

The policy to be followed is $\{m1 \ ... \ m_i\}$

The path needed to traverse the policy chain make up what we call the 'spine'



Network Spine

Tree topologies have one path to a vertex

The spine S is $S \subseteq E$

$S_i = E(v_i, v_{i+1})$

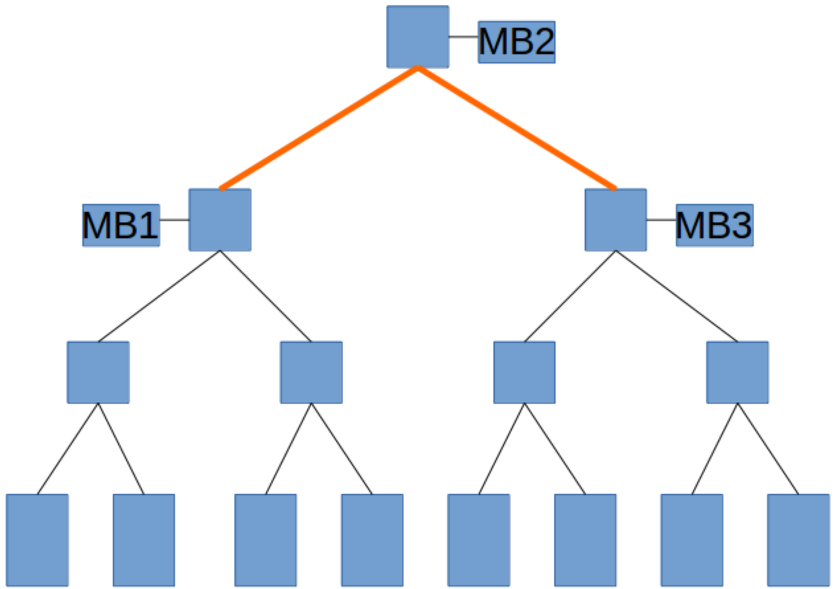
Where i is the middlebox in set M

Ingress switch – first MB in chain

Egress switch – last MB in chain

Example:

E(MB1, MB2) and E(MB2, MB3)



Problem Statement

Given a graph G (V, E) with placed set of middleboxes M and VM pairs P

Each pair in the set of commination pairs P sends traffic with demand D that needs to traverse m1 ... m_i

Goal:

1. Is it possible to satisfy all the VM communication pairs with the given resources
2. choose VM pairs to satisfy in order to maximize priority while ensuring policy is enforced and demand does not exceed edge capacity

Proposed solutions

Feasibility Check

VM communications satisfiable

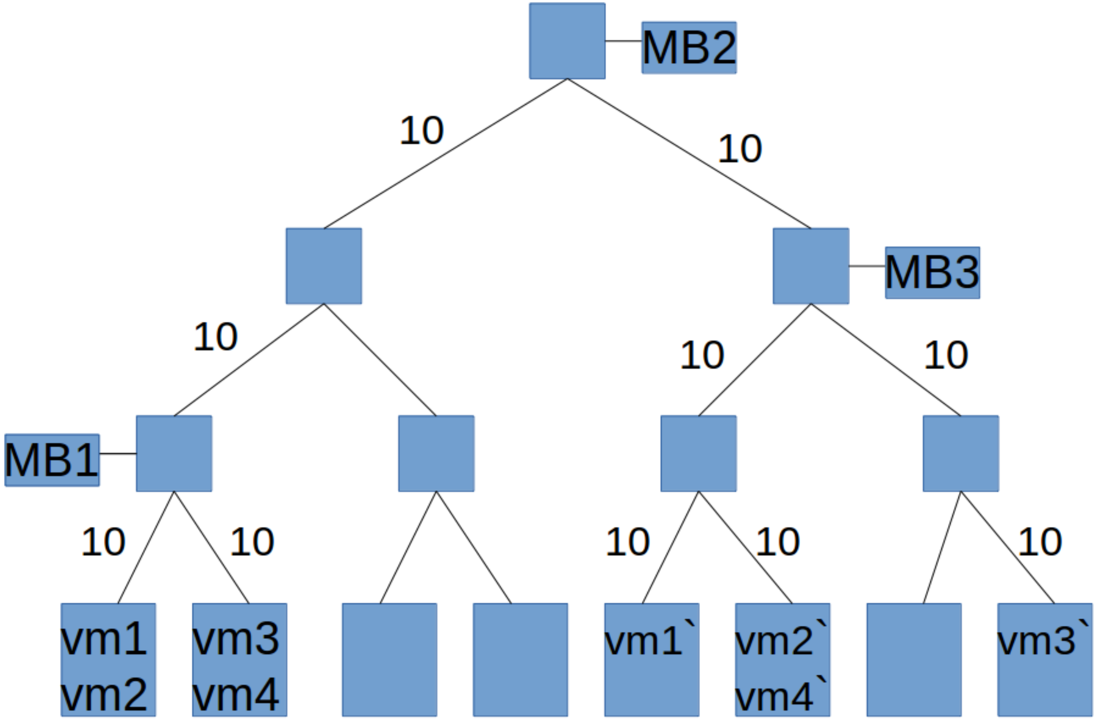
If all VMs are satisfiable, no decisions needed

All VM pairs’ demand is subtracted from link cap. along its path

If any link cap. < 0, not feasible

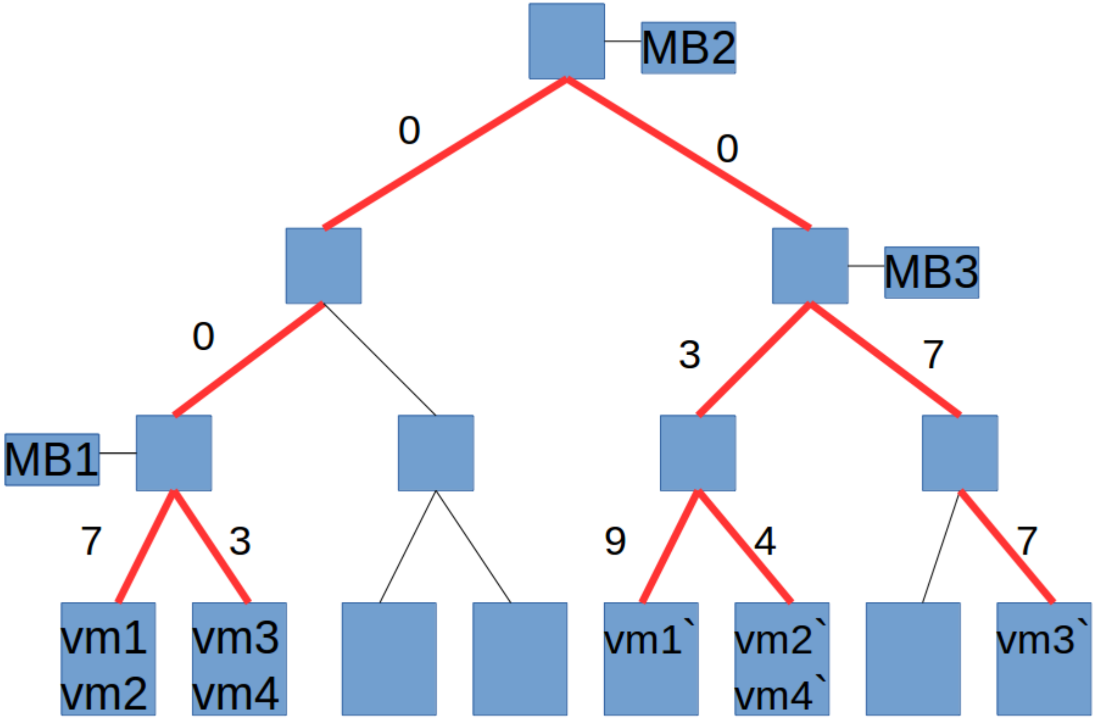
Feasibility Check Example

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4



Feasibility Check Example

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4



Lowest Demand First

VM communication pairs are ordered in non-decreasing order by demand

Edges from source to spine

Edges through the spine

Edges from spine egress to destination

Time complexity is $O(V^2 * n)$

Algorithm 1 Lowest Demand First

Input: data center G(V, E) with placed VM pairs and MBs

Output: communication priority serviced

Notation:

P = unsorted list of VM pairs

$pair.Demand$ = VM pair's demand

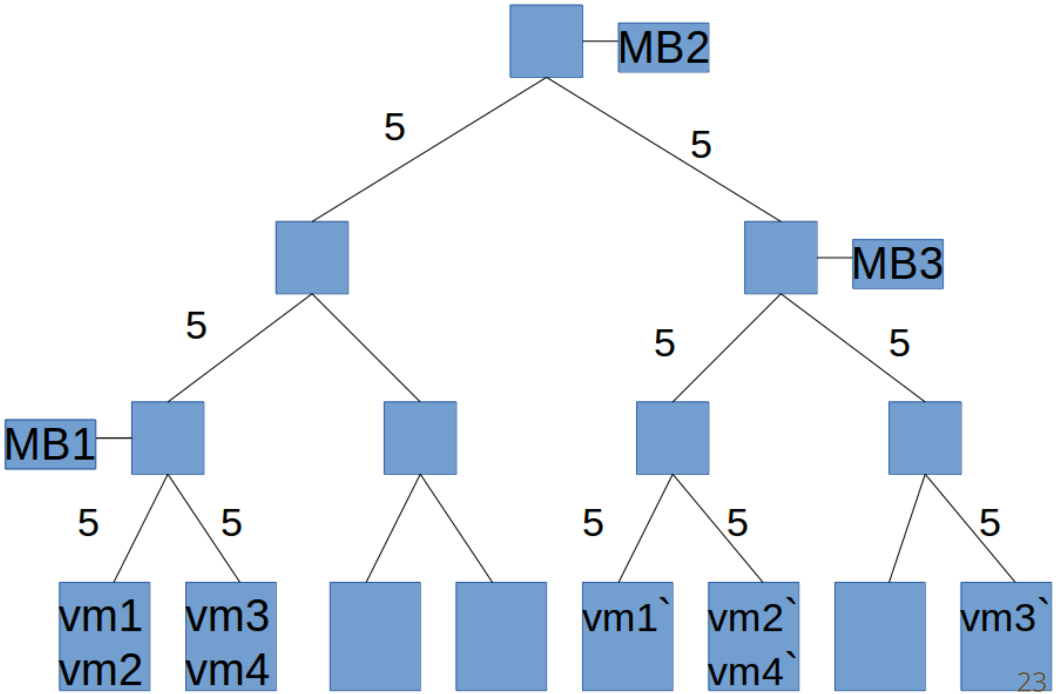
$pair.tPriority$ = VM pair's priority * frequency

$edge.capacity$ = edge's capacity

```
Priority = 0
for i in length(P) do
  for j in range(0, length(P) - i - 1) do
    if P[j].Demand > P[j + 1].Demand then
      swap P[j] with P[j + 1]
    end if
  end for
end for
for pair in P do
  vm1, vm2 ← VMs in pair
  fullFlag = 0
  ES ← edge set from vm1 to vm2 following policy
  for edge in ES do
    if pair.Demand > edge.capacity then
      fullFlag = 1
      break
    end if
  end for
  if fullFlag == 0 then
    establish pair connection
    update edges capacity ES
    Priority += pair.tPriority
  end if
end for
Return Priority
```

Lowest Demand First Example

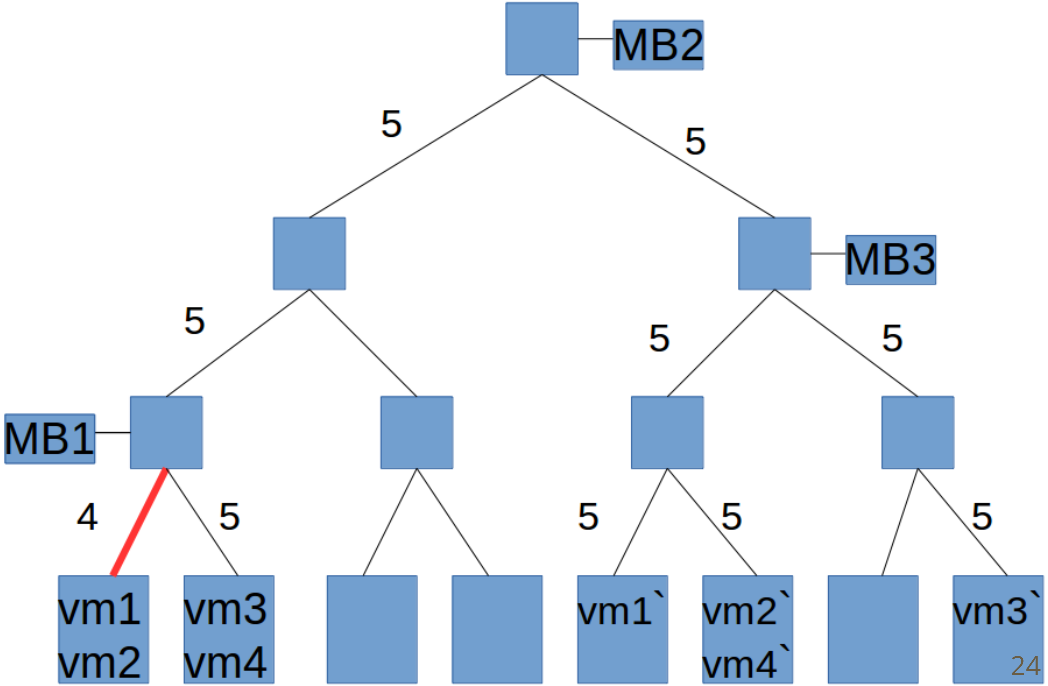
Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4



Lowest Demand First Example

VM1 Selected, Check path to ingress

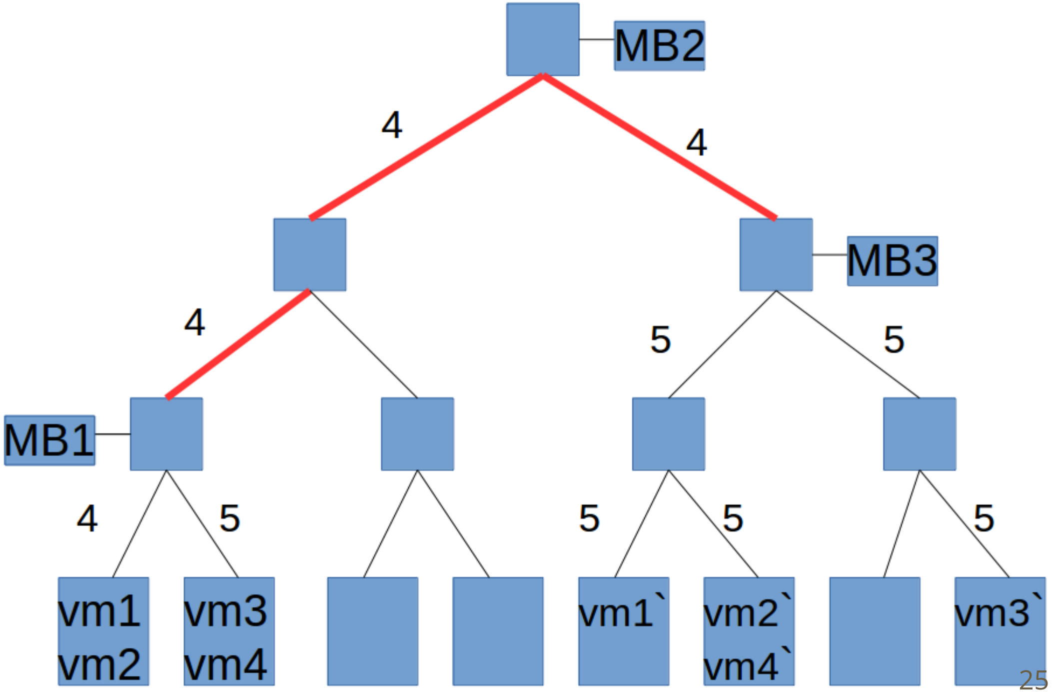
Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4



Lowest Demand First Example

Check path in spine

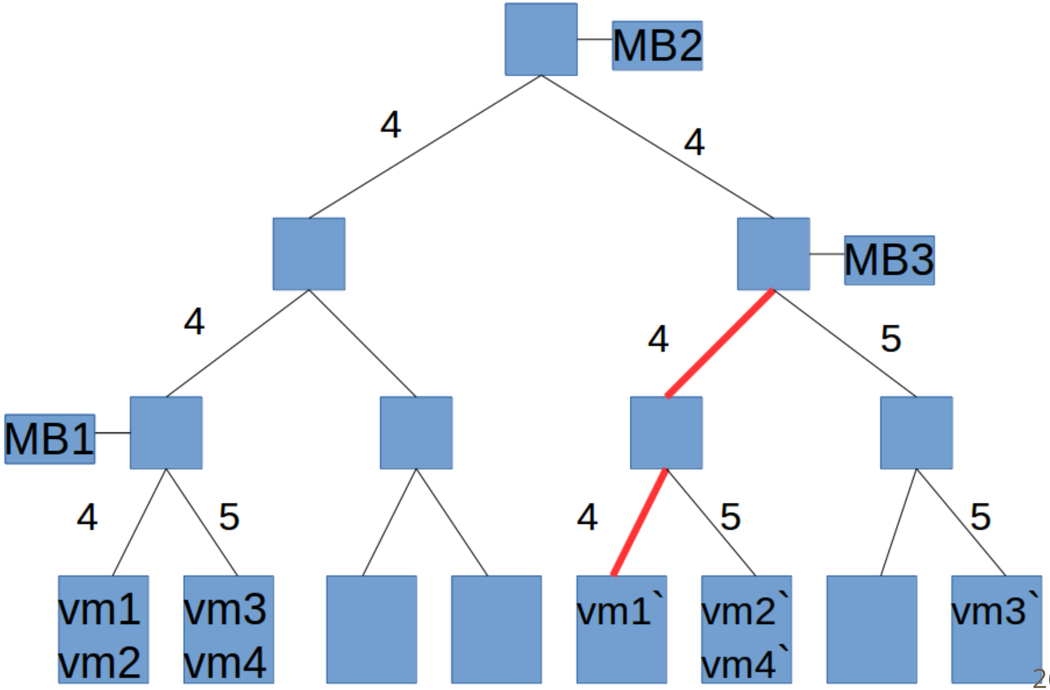
Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4



Lowest Demand First Example

Check path from egress to destination

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4

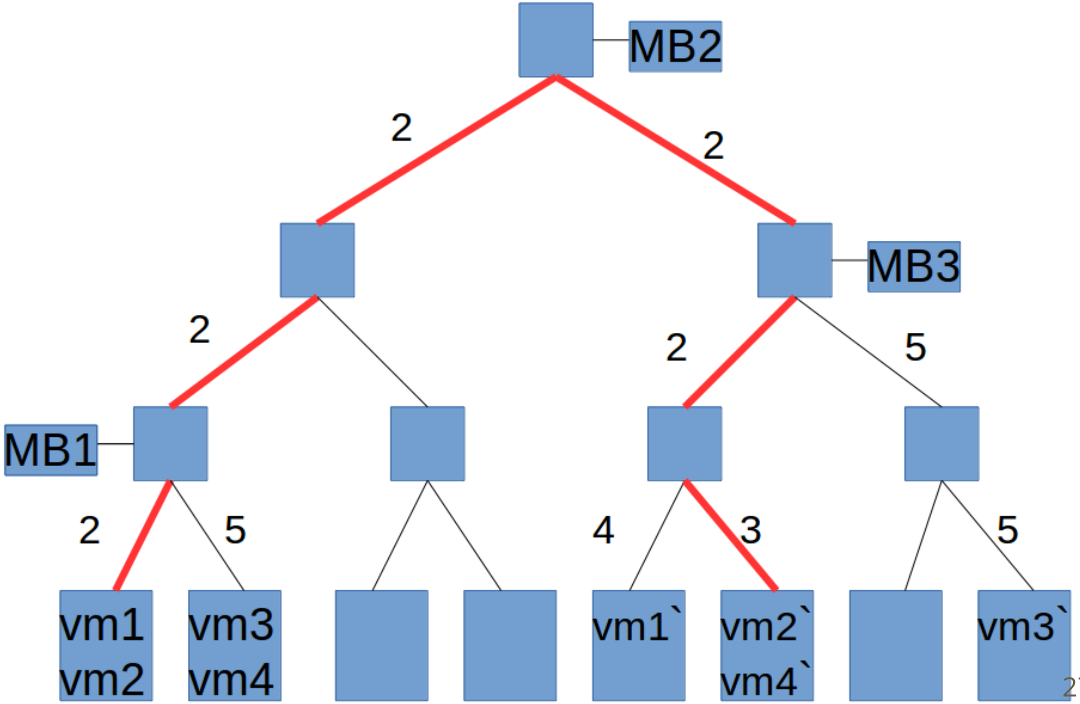


Lowest Demand First Example

VM2 is selected

Same path check is performed

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4

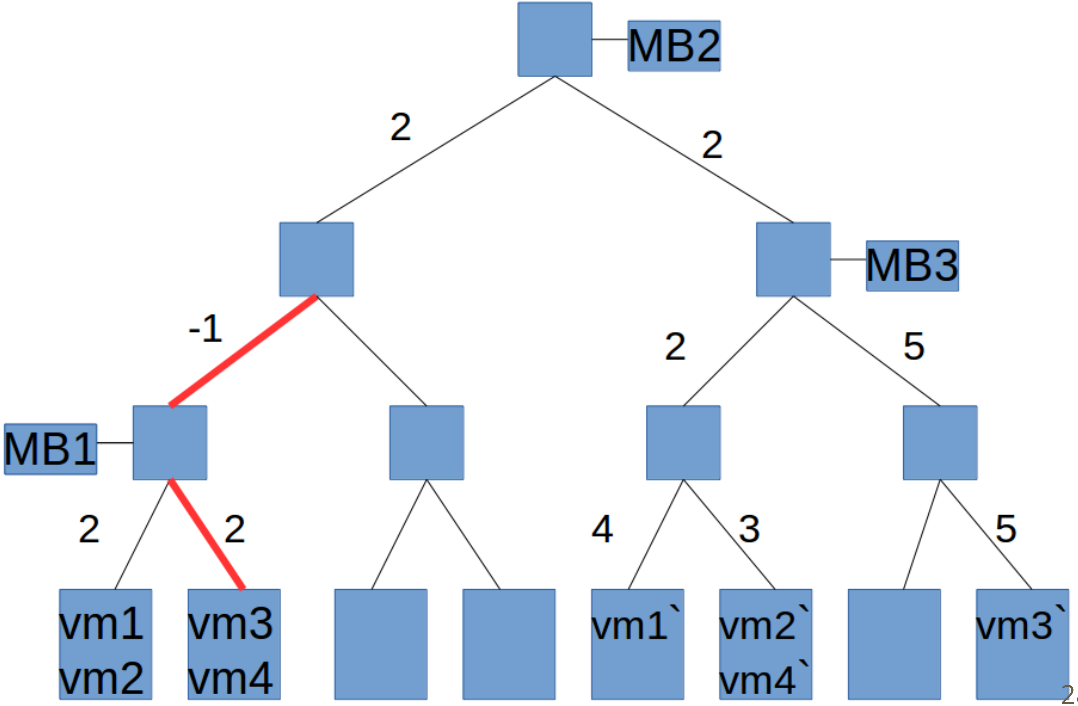


Lowest Demand First Example

VM3 is selected, not enough capacity

Total priority = 4

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4



Highest Priority First

VM are ordered in non-increasing order by their priority

The path for the communication pair is checked

Time complexity is $O(V^2 * n)$

Algorithm 2 Highest Priority First
Input: data center G(V, E) with placed VM pairs and MBs
Output: communication priority serviced
Notation:
P = unsorted list of VM pairs
$pair.Demand$ = VM pair's demand
$pair.tPriority$ = VM pair's priority * frequency
$edge.capacity$ = edge's capacity

```
Priority = 0
for i in length(P) do
    for j in range(0, length(P) - i - 1) do
        if P[j].tPriority < [j + 1].tPriority then
            swap P[j] with P[j + 1]
        end if
    end for
end for
. . .
Return Priority
```

Highest Average Priority

VM priority with respect to its demand

VMs ordered in non-decreasing order by their average priority

The path for the communication pair is checked

Time complexity is $O(V^2 * n)$

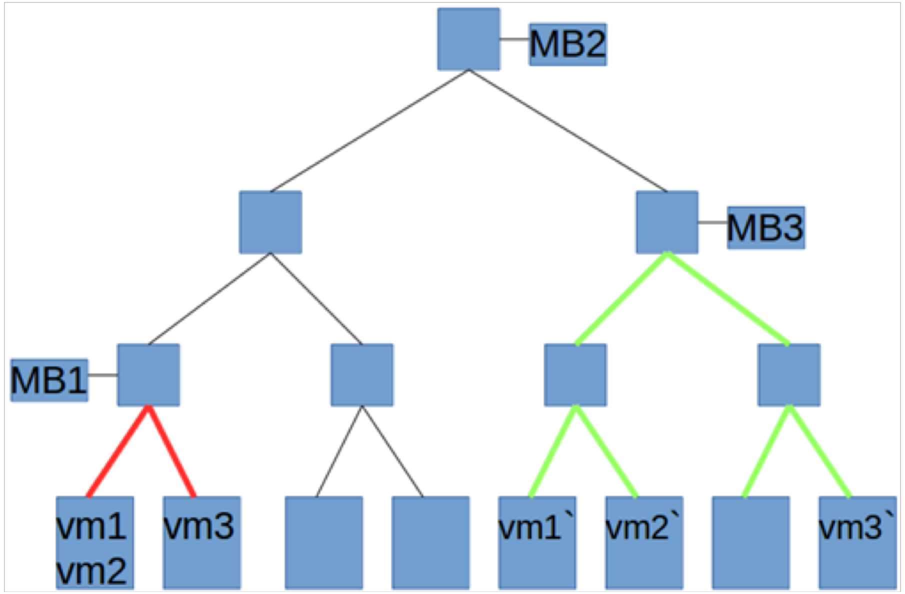
Algorithm 3 Highest Average Priority First
Input: data center G(V, E) with placed VM pairs and MBs
Output: communication priority serviced
Notation:
P = unsorted list of VM pairs
$pair.Demand$ = VM pair's demand
$pair.tPriority$ = VM pair's priority * frequency
$edge.capacity$ = edge's capacity

```
Priority = 0
for i in length(P) do
    for j in range(0, length(P) - i - 1) do
        if P[j].tPriority / P[j].Demand < P[j + 1].tPriority / P[j + 1].Demand then
            swap P[j] with P[j + 1]
        end if
    end for
end for
. . .
Return Priority
```

Special Case

Source VMs hosted in PMs located in subtree of ingress switch

Destination VMs hosted in PMs in subtree of egress switch



1:0 Knapsack Problem

Optimization problem:

- Given a knapsack with capacity C
- Set of items with attributes weight W and value V
- Items can not be split

Which items to choose in order to maximize V such that $W \leq C$

Solution: Dynamic programming

Dynamic Programming

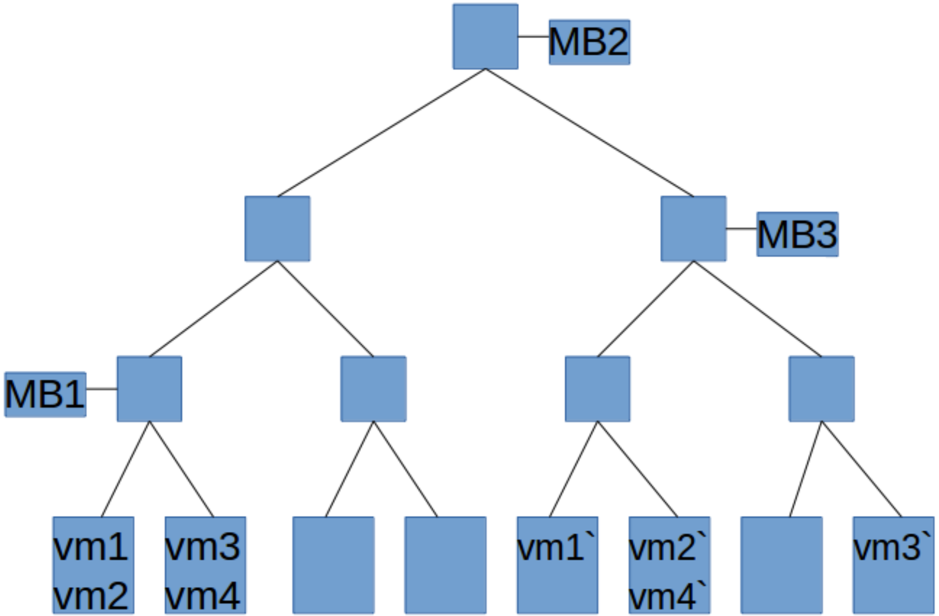
The total capacity restricted by spine

Items are the VM pairs:

Item weight is VM demand

Item value is VM priority

Time complexity is O(V * C + n)

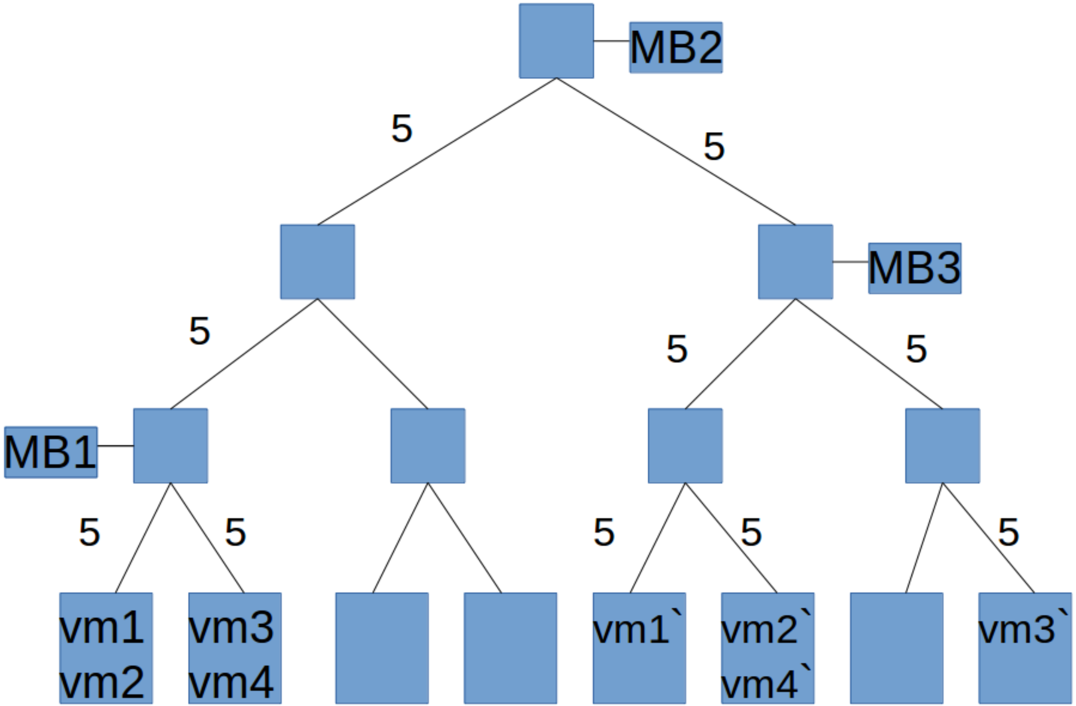


Dynamic Programming Example

All links in spine traversed once

Max Capacity = 5/1 = 5

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4



Dynamic Programming Example

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4

Items \ Weight	0	1	2	3	4	5
0	0	0	0	0	0	0
VM 1						
VM 2						
VM 3						
VM 4						

Dynamic Programming Example

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4

Items \ Weight	0	1	2	3	4	5
0	0	0	0	0	0	0
VM 1	0	2	2	2	2	2
VM 2						
VM 3						
VM 4						

Dynamic Programming Example

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4

Items \ Weight	0	1	2	3	4	5
0	0	0	0	0	0	0
VM 1	0	2	2	2	2	2
VM 2	0	2	2	4	4	4
VM 3						
VM 4						

Dynamic Programming Example

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4

Items \ Weight	0	1	2	3	4	5
0	0	0	0	0	0	0
VM 1	0	2	2	2	2	2
VM 2	0	2	2	4	4	4
VM 3	0	2	2	5	7	7
VM 4						

Dynamic Programming Example

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4

Items \ Weight	0	1	2	3	4	5
0	0	0	0	0	0	0
VM 1	0	2	2	2	2	2
VM 2	0	2	2	4	4	4
VM 3	0	2	2	5	7	7
VM 4	0	2	2	5	7	8

Dynamic Programming Example

Link Cap. = 5	Priority	Demand
VM 1	2	1
VM 2	2	2
VM 3	5	3
VM 4	6	4

VM4 and VM1 Chosen

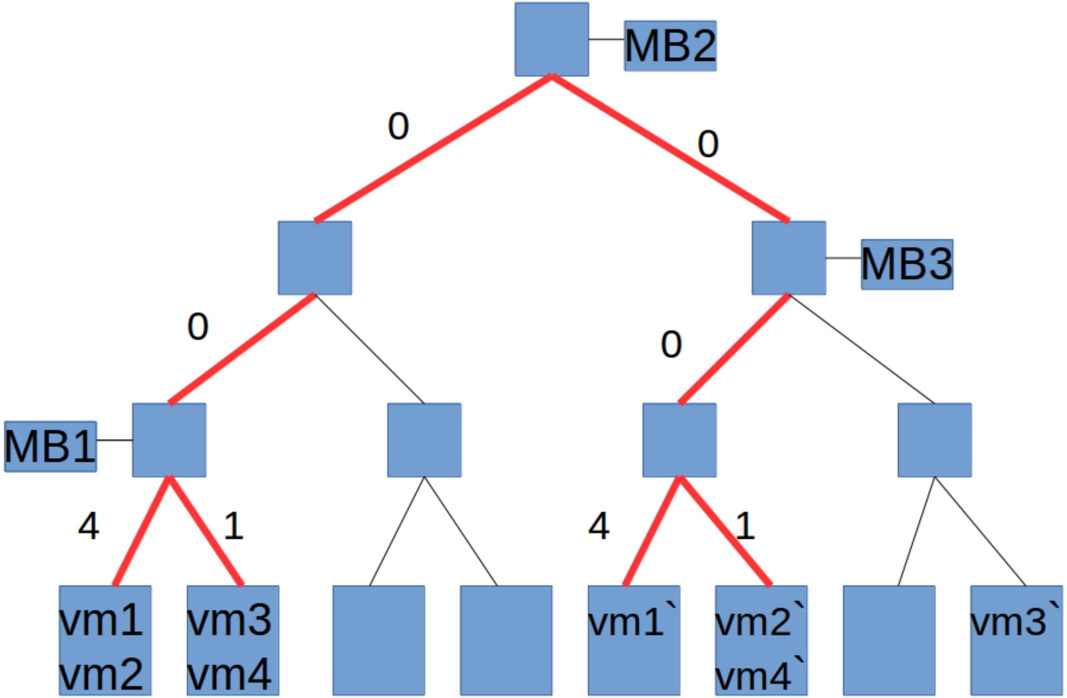
Items \ Weight	0	1	2	3	4	5
0	0	0	0	0	0	0
VM 1	0	2	2	2	2	2
VM 2	0	2	2	4	4	4
VM 3	0	2	2	5	7	7
VM 4	0	2	2	5	7	8

Dynamic Programming

VM4 and VM1 selected

VM communication sent

Total of 8 priority



Simulation and Results

Simulation Parameters

Tree data center with 84 nodes each node with 4 children

- 21 switches
- 64 physical machines

VM communication attributes randomized

- Communication priority in the range of 1-100
- Communication frequency in the range of 1-10
- Communication demand in the range of 1-10

Variables checked: Amount of Mbs, link capacity, and amount of VMs

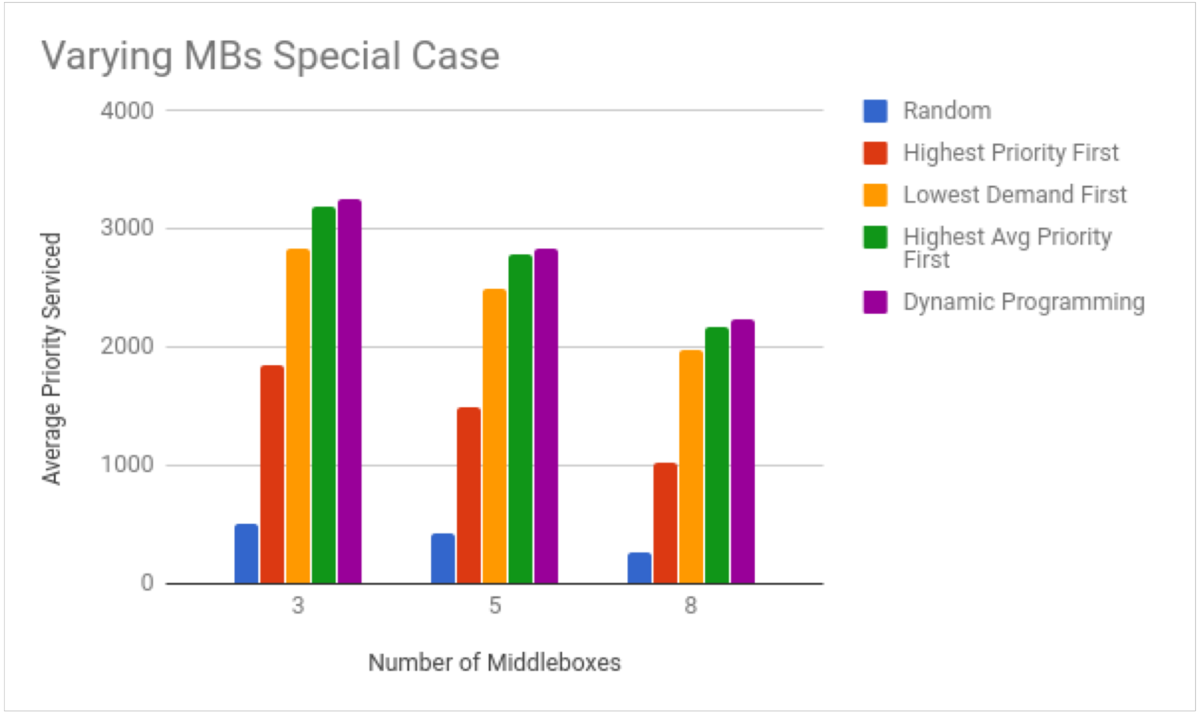
Varying Amount of Middleboxes

Amount of MBs = 3, 5, 8

Link Cap. = 200

Amount of pairs = 200

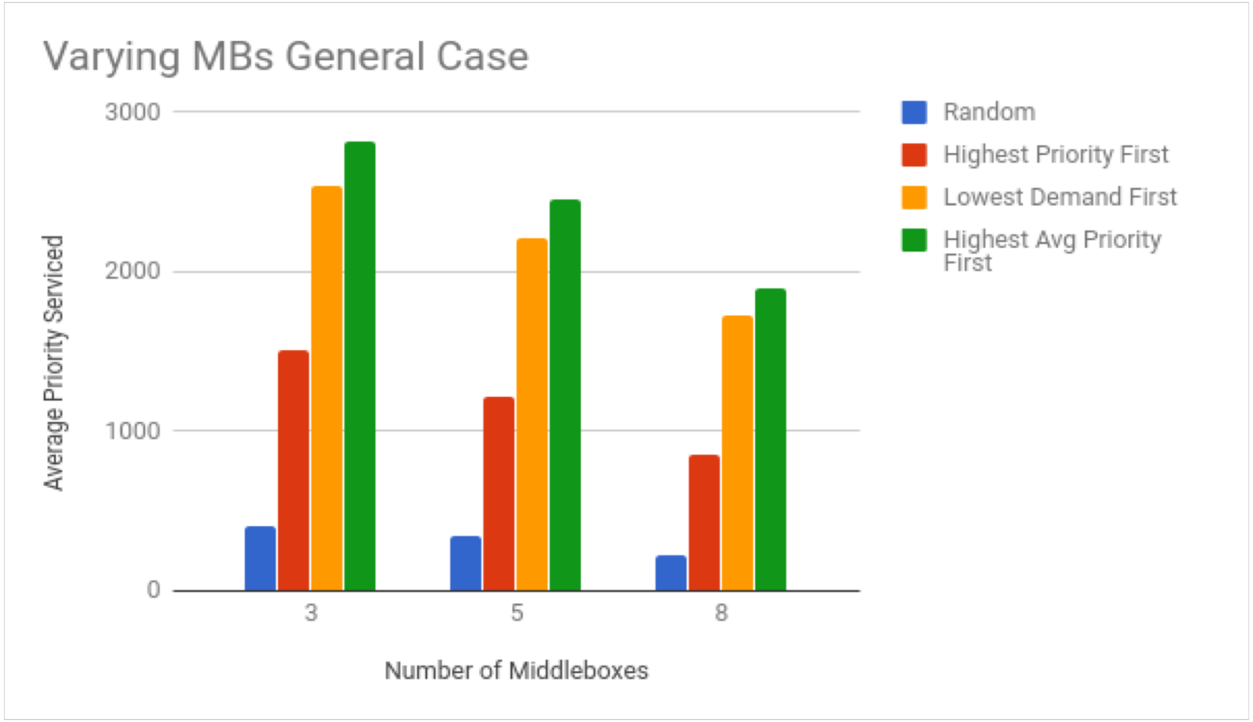
Dynamic programming outperforms in all cases



Varying Amount of Middleboxes

General case, VM placement unrestricted

Highest Average Priority First performs the best



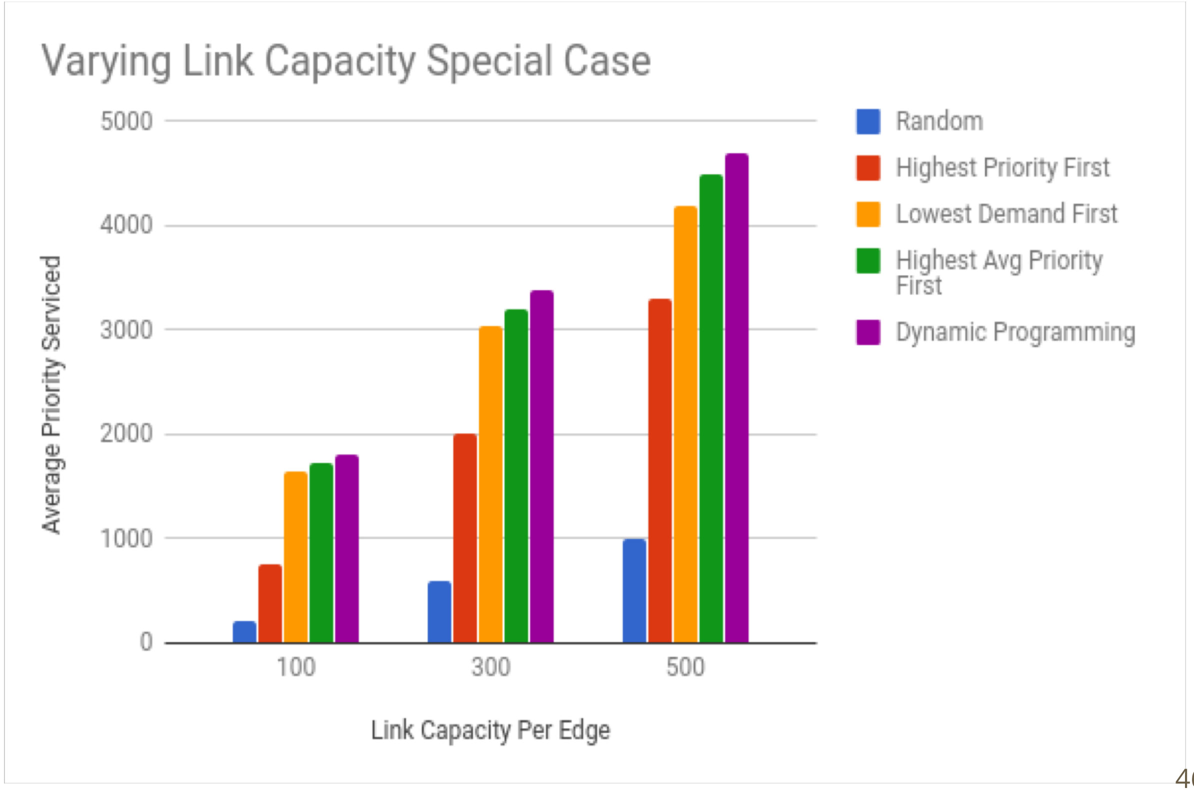
Varying Amount of Link Capacity

Link Cap. = 100, 300, and 500

Amount of MBs = 5

Amount of pairs = 200

Dynamic programming outperforms in all cases

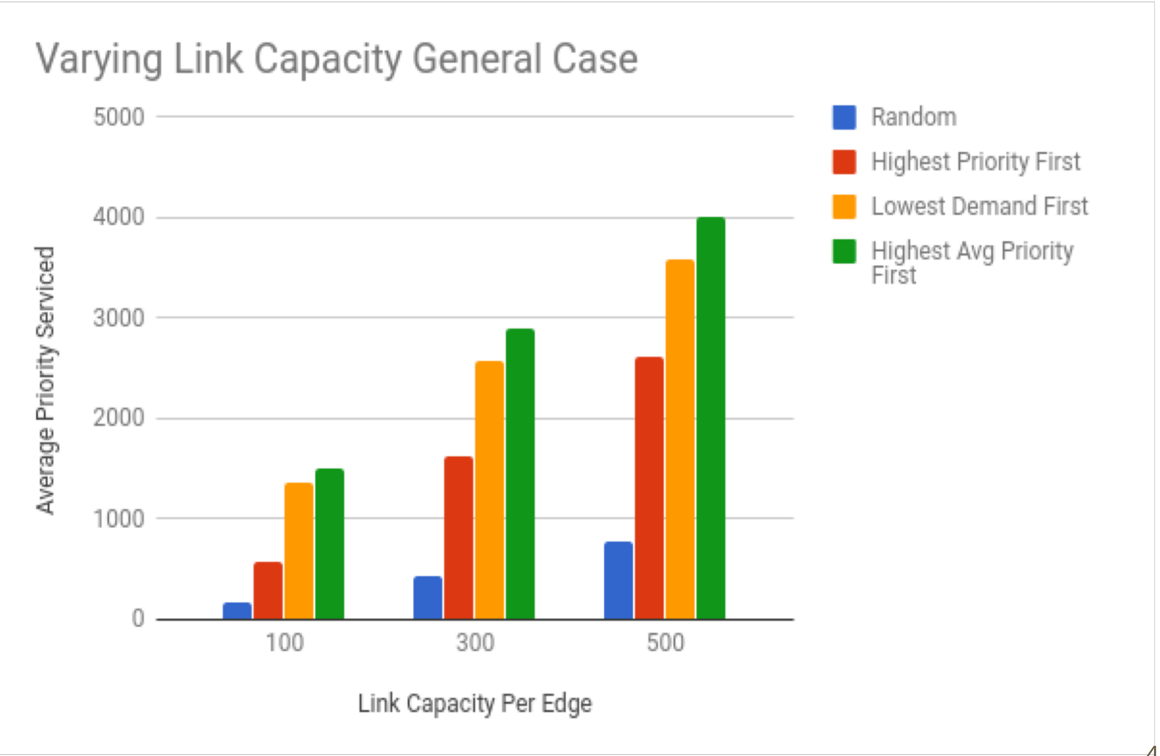


Varying Amount of Link Capacity

General case, VM placement unrestricted

Performance increase with link capacity

Highest Average Priority First performs the best



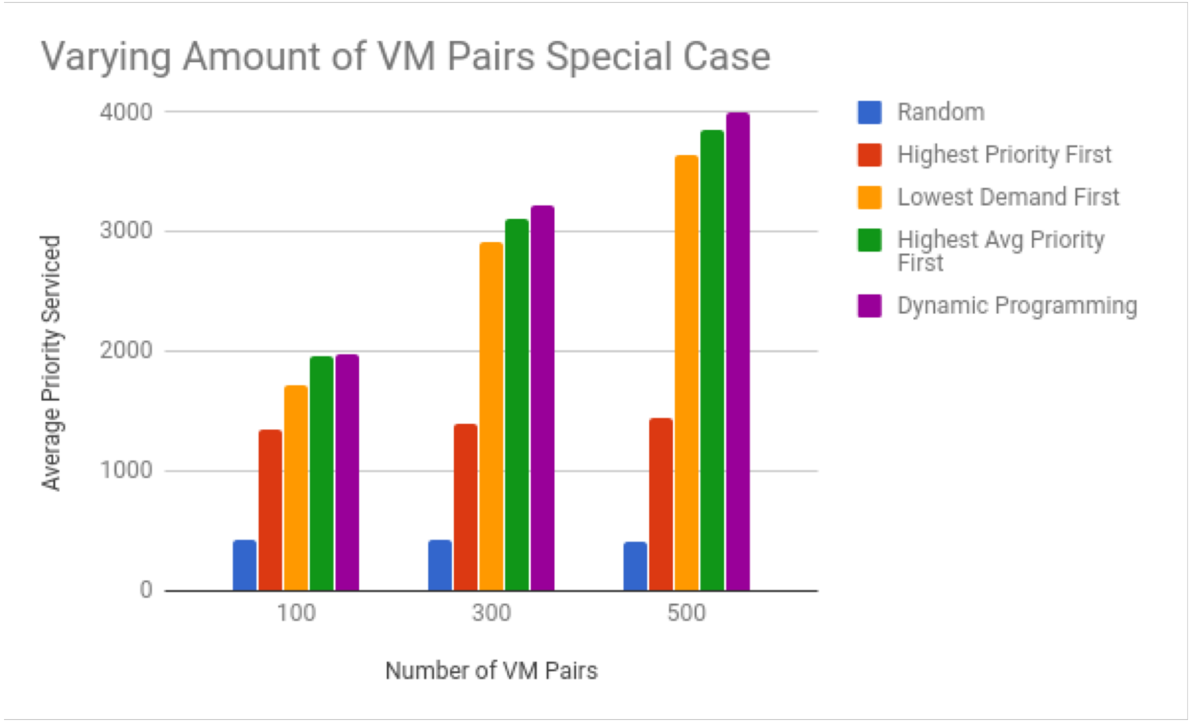
Varying Amount of Communication Pairs

Amount of pairs = 100, 300, and 500

Amount of MBs = 5

Link Cap. = 200

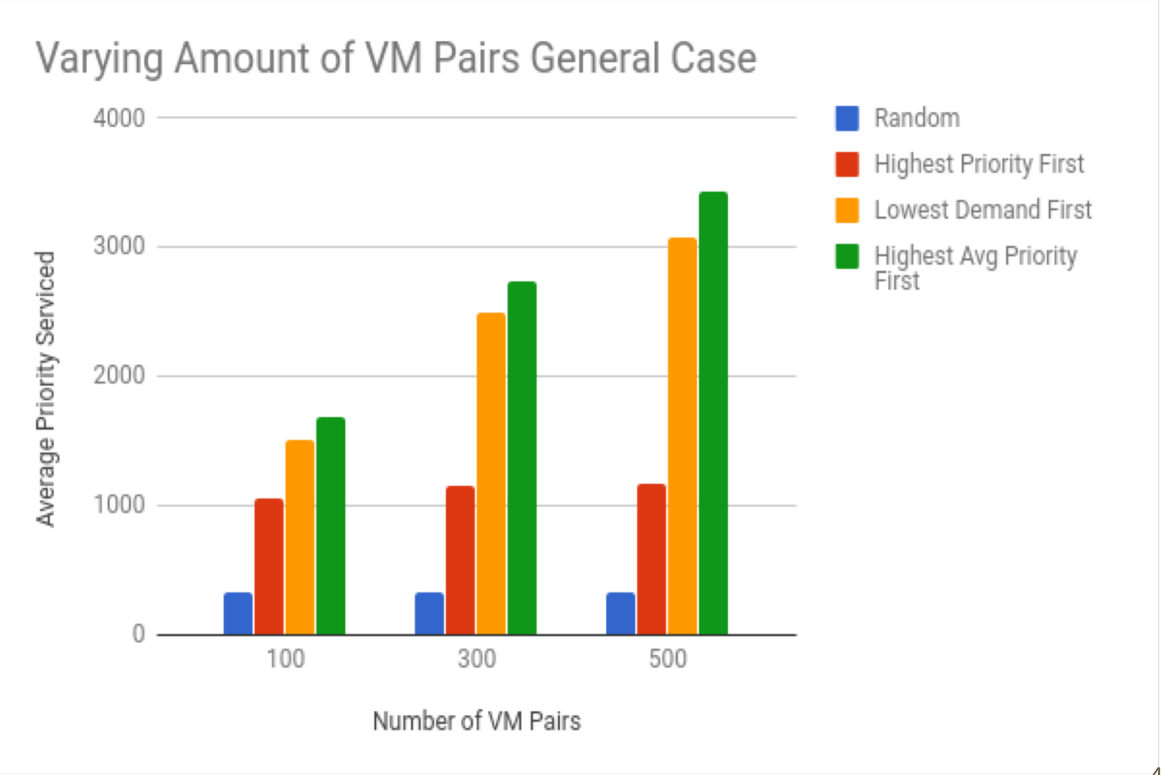
Dynamic programming outperforms in all cases



Varying Amount of Communication Pairs

General case, VM placement unrestricted

Highest Average Priority First performs the best



Future work and Conclusion

Future Work

Development and testing of general solution

Multiple middleboxes with multiple instances

Addition of other NFV technologies such as VM replication

Testing proposed solutions in emulated environment

Conclusion

Four algorithms proposed for priority maximization

Three heuristics and dynamic programming

Showed the tree network can be modeled as 1/0 knapsack under special conditions

Showed Dynamic programming approach performed the best, with the highest average demand performing the best in the general case