

國立交通大學

電機工程學系碩士班

碩士論文

基於軟體定義網路的雲端資料中心上  
網路效能和能源消耗最佳化研究

Flow-and-VM Migration for Optimizing  
Throughput and Energy in SDN-based Cloud  
Datacenters

研究生：林洧駐

指導教授：溫宏斌

中華民國

一百零二年七月

基於軟體定義網路的雲端資料中心上  
網路效能和能源消耗最佳化研究

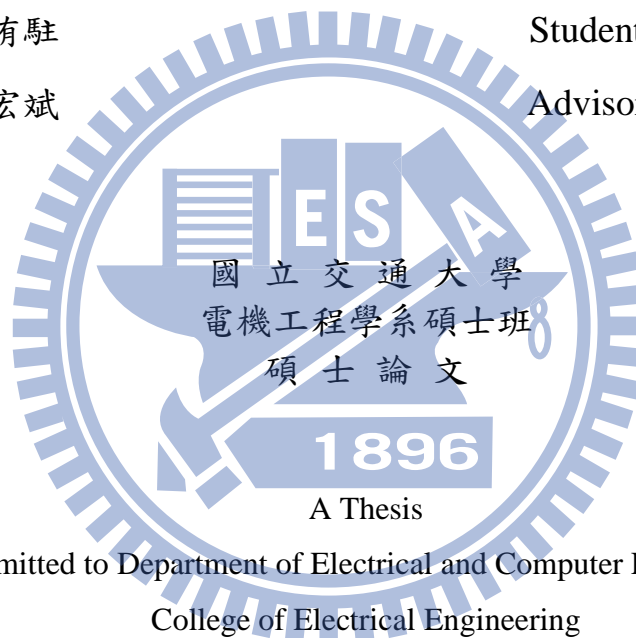
Flow-and-VM Migration for Optimizing  
Throughput and Energy in SDN-based Cloud  
Datacenters

研究生：林洵駐

Student : Wei-Chu Lin

指導教授：溫宏斌

Advisor : Hung-Pin Wen



A Thesis

Submitted to Department of Electrical and Computer Engineering

College of Electrical Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Electrical and Computer Engineering

July 2013

Hsinchu, Taiwan, Republic of China

中華民國 一 百 零 二 年 七 月

# 基於軟體定義網路的雲端資料中心上 網路效能和能源消耗最佳化研究

學生：林洵駐

指導教授：溫宏斌 教授

國立交通大學

電機學院電機工程學系碩士班

## 摘要

近年來，隨著雲端運算(Cloud Computing)的快速發展，資料中心扮演了越來越重要的角色提供大量的計算與服務。對雲端供應者而言，如何減少資料中心的能源消耗以及如何增加資料中心網路效能成為重要的課題用來節省開銷並得到最大利益。在這篇論文中，我們討論有關於節能以及提高網路效能的議題，提出「動態即時的流量和虛擬機器搬遷演算法」，在節能的條件下有效率的增加網路效能。

在本文，我們提出的「動態即時的流量和虛擬機器搬遷演算法」由兩個主要的概念所構成：(1) 藉由快速流量搬遷機制 (Traffic-aware Flow Migration, TA-FM) 及 (2) 考量節能及網路拓撲的虛擬機器搬遷機制 (Energy-and-Topology aware VM Migration, ETA-VMM)，前者在軟體定義網路裡提供了更快速的流量繞徑控制；後者則是在原本節能的虛擬機器搬遷演算法裡加入網路的考量。本文也結合兩個模擬器 NS2 和 CloudSim 的模擬數據來驗證我們提出的正確性及優越性。

# Flow-and-VM Migration for Optimizing Throughput and Energy in SDN-based Cloud Datacenters

Student : Wei-Chu Lin

Advisors : Hung-Pin Wen

Department of Electrical and Computer Engineering  
National Chiao Tung University

## ABSTRACT

Minimizing energy consumption and improving performance in datacenters are critical to cost-saving for cloud operators, but traditionally, these two optimization objectives are treated separately. VM migration policies are frequently provided by the cloud controller for energy saving, while a routing algorithm is adopted to improve network performance by the network controller. However, the interaction between the cloud controller and the network controller is not considered. Therefore, this paper presents an unified solution combining two strategies, *flow migration* and *VM migration*, to maximize throughput and minimize energy, simultaneously. Traffic-aware flow migration (FM) is first incorporated a dynamic reroute algorithm (DENDIST), evolving into **DENDIST-FM**, in a software-defined network (SDN) for improving throughput and avoiding congestion. Second, given energy and topology information, VM migration (**ETA-VMM**) can help reduce traffic loads and meanwhile save energy. Our experimental result indicates that compared to previous works, the proposed method can improve throughput by 50.0% on average with only 3.2% energy overhead. Finally, the unified flow-and-VM migration solution has been proven effective for optimizing throughput and energy in SDN-based datacenters.

## 誌 謝

能夠順利地完成此論文，首先要感謝我的指導教授溫宏斌老師。老師在我碩士的這幾年期間，不僅教導我有關課業和研究上的專業知識，還教導我許多做人處事和生活的道理。另外，老師熱心地經營實驗室的氣氛，讓學生們可以以愉悅的心情和舒適的環境下研究，並且把自己無私地奉獻在教學研究上，將會是我一輩子的楷模。

在整個碩士學涯中，影響我最深的莫過於是耕含，每次遭遇到困難時求助於他，他都會義氣相挺、拔刀而助，大部分程式工具的應用和研究的技巧多虧他的幫助。除此之外，耕含還是我運動和生活中最好的益友，謝謝讓我覺得在充實的研究生活外，還有健康的身心。

接著要感謝 CIA 實驗室裡所有的同學，謝謝佳伶時常關心我並且給我許多建議，謝謝在我寫論文時幫助我最多的宣銘、千慧和鈞堯，另外還有竣惟、昱澤、凱華、玗璇、冠辰、哲誠、宜武、芄銘、勝彥、索成、品維、以及同屆戰友鉉崴和菘昀，謝謝你們平常的照顧和幫忙。

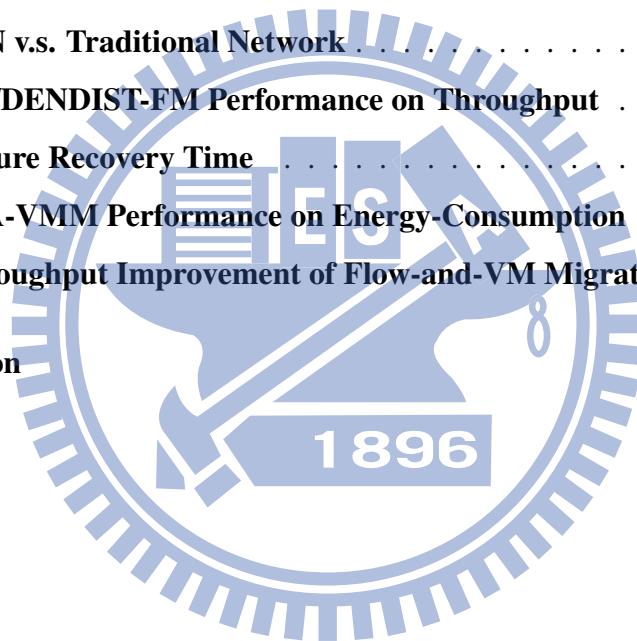
此外，還要感謝在 AIESEC 所認識的朋友們，落落、QQ、Saying、Joanne、嘉倫、幹兄、Dirk、穎瑄、阿迪以及冬季全國大會的朋友等，即使我在忙也願意跑來找我為我加油打氣，還有你們的紙條卡片，總是在我最無助時給我動力。還有早餐團的大家，辦斯、小竣、土豆、小 Q、阿嘎、阿軒、小臉等，讓我可以充滿勇氣迎接最辛苦的每一天。特別感謝小臉提供我住宿和生活上的各種幫助。

最後僅以此文獻給我最愛的父母和哥哥，謝謝你們總是可以體諒我必須常常忙碌而疏於與家人相處。

# Contents

<b>Abstract</b>	<b>i</b>
<b>Abstract-English</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>List of Tables</b>	<b>vi</b>
<b>List of Figures</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Thesis Outline . . . . .	3
<b>2 Background</b>	<b>6</b>
2.1 Dynamic Routing - D <sup>2</sup> ENDIST . . . . .	6
2.2 SDN and OpenFlow . . . . .	10
2.3 Literature Survey . . . . .	12
<b>3 System Model</b>	<b>14</b>
3.1 SDN Controller . . . . .	14
3.2 Cloud Controller . . . . .	15

<b>4</b>	<b>Dynamic Reroute with Flow Migration (DENDIST-FM) in SDN</b>	<b>16</b>
4.1	Problem of D <sup>2</sup> ENDIST in SDN . . . . .	17
4.2	Traffic-aware Flow Migration (FM) . . . . .	20
4.3	DENDIST-FM . . . . .	22
<b>5</b>	<b>Energy-and-Topology-aware VM Migration (ETA-VMM)</b>	<b>27</b>
5.1	Energy-aware VM Selection . . . . .	27
5.2	Topology-aware VM Placement . . . . .	28
<b>6</b>	<b>Experimental Results</b>	<b>30</b>
6.1	SDN v.s. Traditional Network . . . . .	31
6.2	FM/DENDIST-FM Performance on Throughput . . . . .	33
6.3	Failure Recovery Time . . . . .	35
6.4	ETA-VMM Performance on Energy-Consumption . . . . .	36
6.5	Throughput Improvement of Flow-and-VM Migration . . . . .	37
<b>7</b>	<b>Conclusion</b>	<b>40</b>
	<b>Bibliography</b>	<b>42</b>



## List of Tables

2.1	All Routing Path of Trad. Network: SPB . . . . .	8
2.2	All Routing Path of Trad. Network: D <sup>2</sup> ENDIST . . . . .	9
2.3	Flow Table Entry of OpenFlow . . . . .	11
4.1	All Routing Path of SDN: D <sup>2</sup> ENDIST . . . . .	18
4.2	All Routing Path of SDN: DENDIST + FM . . . . .	24
6.1	Different scales of topologies . . . . .	31
6.2	Energy comparison of different VM migration policies . . . . .	37
6.3	Experimental results of different scale topologies . . . . .	39



# List of Figures

1.1	Original routing and VM placement	4
1.2	Improved by dynamic re-route	4
1.3	Improved by VM re-placement	5
2.1	Example of Trad. Network: SPB	8
2.2	Example of Trad. Network: D <sup>2</sup> ENDIST	9
3.1	SDN-based Cloud Network Architecture	15
4.1	Example of SDN: D <sup>2</sup> ENDIST	18
4.2	The bottleneck of traditional routing	20
4.3	Example for flow migration	21
4.4	The drawback of D <sup>2</sup> ENDIST	22
4.5	Example of SDN: DENDIST + FM	24
4.6	Flowchart of <i>DENDIST-FM</i>	26
5.1	Regions in fat-tree topology	28
5.2	Flowchart of <i>ETA-VMM</i>	29
6.1	The adaptivity of SDN with imbalanced traffic loads	32
6.2	Different TIR	32
6.3	Throughput comparison of FM in different periods	33
6.4	Throughput comparison of DENDIST-FM	34

6.5 Recovery time for the network topology . . . . . 35

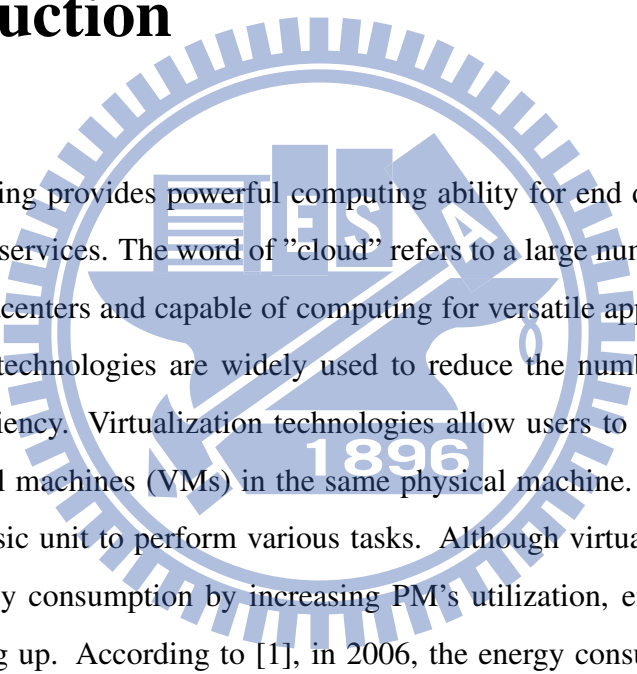
6.6 Throughput comparison of flow-and-VM migration . . . . . 38





# CHAPTER 1

## Introduction



Cloud computing provides powerful computing ability for end devices to access anytime and anywhere services. The word of "cloud" refers to a large number of servers, which are located in datacenters and capable of computing for versatile applications. In datacenters, virtualization technologies are widely used to reduce the number of physical machines (PM) for efficiency. Virtualization technologies allow users to run different applications through virtual machines (VMs) in the same physical machine. A virtual machine (VM) serves as a basic unit to perform various tasks. Although virtualization technologies can improve energy consumption by increasing PM's utilization, energy cost of datacenters is still growing up. According to [1], in 2006, the energy consumption of datacenters in U.S. was 61 billion kilowatt-hour (kWh), and will further becomes double between 2010 to 2015. As a result, improving energy efficiency allows of no delay to cloud datacenters.

Moreover, datacenters aims to support high-performance computing. However, tremendous traffics due to communication between VMs for versatile applications are generated, accompanying more energy consumption. Networking issues are often bottlenecks for providing services to clients. Therefore, providing high-quality and stable services for a large number of clients while minimizing operation costs to maximize profits are critical

for cloud operators. To sum up, two main issues arise for cloud operators: (1) **network-throughput optimization** and (2) **energy-saving improvement**.

For optimizing throughput and energy, we propose an unified solution consisting of flow migration and VM migration techniques for software-defined network (SDN) based cloud datacenters. SDN [2] is an evolution of computer networks and has been widely applied to datacenter networks. SDN decouples the control plane from the network devices and makes the controlling mechanisms centralized and programmable.

## 1.1 Motivation

To sum up, our proposed method aims at reducing unnecessary traffics in a datacenter network and excessive energy consumption originated from bad routing and improper VM allocation. Therefore, two aforementioned techniques running on the SDN controller and the cloud controller are combined, separately. Given link utilization of a datacenter network, our **dynamic reroute with flow migration** (DENDIST-FM) improves throughput and avoids congestion. **energy-and-topology aware VM migration** (ETA-VMM) leverages topological relationship to find best locations of VMs to move under energy and network constraints (i.e. SLAs).

Figure 1.1, 1.2, and 1.3 shows an example to illustrate the proposed method step by step. Such cloud architecture is a three-layer fat-tree with 10 switches (denoted by 0 to 9) and 8 host machines (denoted by A to H). 3 different kinds of applications (denoted as a to c) are running on VMs. Figure 1.1 shows the original routing and VM allocation. Assume that each host machine can run at most two VMs and the value on a link represents the number of traffic units. Figure 1.2 shows the result of applying DENDIST-FM to Figure 1.1, and thus the traffic on over-utilized link 2-6 is re-distributed to link 2-6 and link 3-6. Figure 1.3 show the result of applying **energy-and-topology aware VM migration** (ETA-VMM). The VM running application *a* on machine *E* is migrated to machine

*A*. Both the VM running application *b* on machine *A* and the VM running application *c* on machine *E* are migrated to machine *D*. As a result, the overall network loading is reduced<sup>1</sup>.

## 1.2 Thesis Outline

The rest of this paper is organized as follows. In Chapter 2, we discuss the background of D<sup>2</sup>ENDIST and SDN. Chapter 3 introduces a typical SDN-based cloud architecture where two key components, *SDN controller* and *cloud controller*, are detailed. Later, the first part (**dynamic reroute with traffic-aware flow migration** (DENDIST-FM) for SDN controller) of our proposed method is elaborated in Chapter 4. Then, Chapter 5 describes the second part (**energy-and-topology aware VM migration** (ETA-VMM) for cloud controller) of the proposed method. In Chapter 6, experimental results show the effectiveness of the proposed flow-and-VM migration method in SDN-based cloud datacenters. Last, Chapter 7 conclude this paper.

---

<sup>1</sup>Energy saving by ETA-VMM is not sufficiently shown in this example.

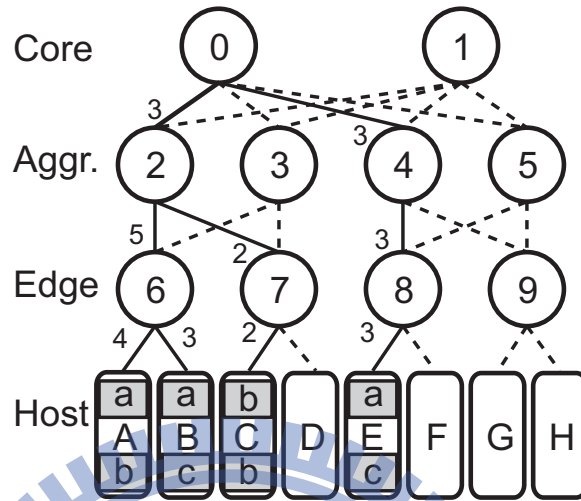


Figure 1.1: Original routing and VM placement

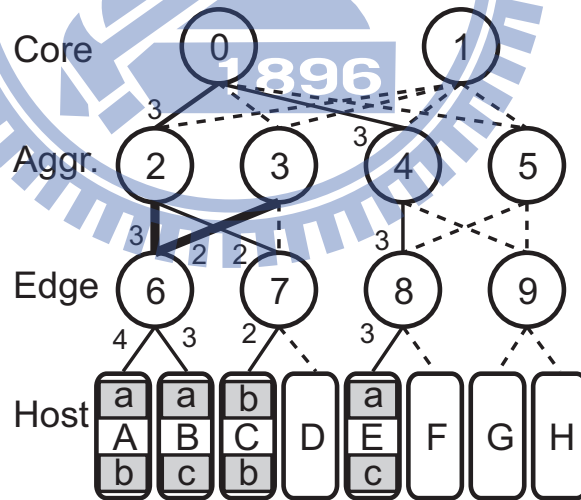


Figure 1.2: Improved by dynamic re-route

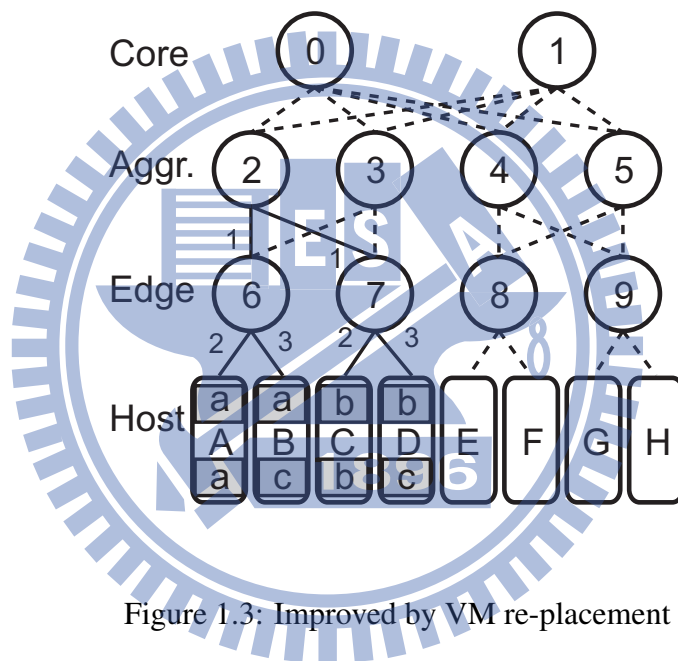


Figure 1.3: Improved by VM re-placement



# CHAPTER 2

## Background

Network utilization is one of the most critical issues in determining performance of a datacenter. Various techniques [3] [4] [5] are proposed to support multiple paths and to solve the oversubscription ratio for performance improvement. This work investigates various routing issues in a datacenter network (DCN), which software-define networking (SDN) is applied to. The routing issues and the concept of SDN are elaborated in the following sections, respectively.

### 2.1 Dynamic Routing - D<sup>2</sup>ENDIST

D<sup>2</sup>ENDIST was proposed to provide disjoint routing paths and served as a dynamic-reroute mechanism. One of the ideas originates from ENDIST [6]. ENDIST provides multiple selections from different divided edge nodes but may cause overlapping paths in a symmetric DCN topology. Disjoint ENDIST, an improved version of ENDIST, was built upon a spanning tree algorithm that divides weighted edge nodes. In this version, the routing candidates are totally disjoint. The other idea comes from the dynamic mechanism. Since the traffic pattern is time-invariant and under-determined, applying disjoint ENDIST will lower uti-

lization of links. Thus, disjoint and dynamic ENDIST (called D<sup>2</sup>ENDIST) was invented to alleviate load unbalancing during runtime.

Figure 2.1 illustrates the layer-two shortest-path-bridge (SPB) protocol [7] in a three-layer fat-tree with 10 switches (indexed from 0 to 9) and 8 hosts (indexed from A to H). In this example, 10 flows are generated in the topology and the respective routing paths are described in Table 2.1. New paths of SPB need to be recomputed by considering the weights of current routing paths. Otherwise, serious congestion may occur frequently. For simplicity, in our example, all of the flows are assumed with equal traffic volumes and the links between aggregate-level and edge-level switches can accommodate at most 3 flows at the same time. The oversubscription ratio between core-aggregate and aggregate-edge links is set as 2:1, meaning that the links between the aggregate level and the core level switches can accommodate at most 6 flows. Over-utilization occurs when the number of flows exceed its capacity. In this example, link 6 – 2, 7 – 2, 2 – 0, 0 – 4, and 4 – 8 are over-utilized links. Since two different spanning trees can be generated by D<sup>2</sup>ENDIST, the load is balanced on the fat-tree topology. Figure 2.2 shows that D<sup>2</sup>ENDIST outperforms the other routing algorithms due to disjoint paths. The rerouted paths are updated in Table 2.2 and the number of over-utilized links is reduced from 5 to 3. The underlined numbers indicate the differences between the original paths and new ones.

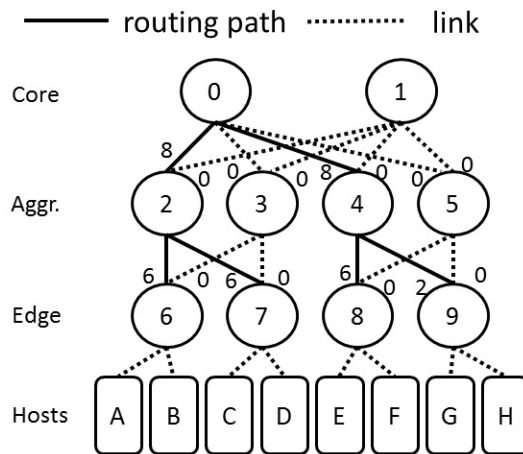


Figure 2.1: Example of Trad. Network: SPB

Table 2.1: All Routing Path of Trad. Network: SPB

	Source	Intermediate	Destination
Path#01	A	6 2 0 4 8	E
Path#02	B	6 2 0 4 8	E
Path#03	A	6 2 0 4 8	F
Path#04	B	6 2 0 4 8	F
Path#05	C	7 2 0 4 8	E
Path#06	C	7 2 0 4 8	F
Path#07	C	7 2 0 4 9	G
Path#08	C	7 2 0 4 9	H
Path#09	A	6 2 7	C
Path#10	A	6 2 7	D

Over-Utilization Links: 6-2, 7-2, 2-0, 0-4, 4-8

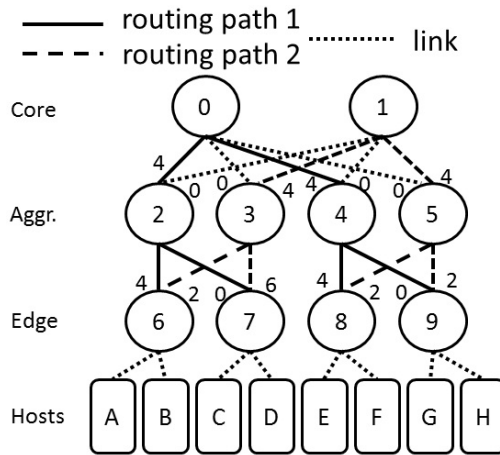


Figure 2.2: Example of Trad. Network: D<sup>2</sup>ENDIST

Table 2.2: All Routing Path of Trad. Network: D<sup>2</sup>ENDIST

	Source	Intermediate	Destination
Path#01	A	6 2 0 4 8	E
Path#02	B	6 2 0 4 8	E
Path#03	A	6 2 0 4 8	F
Path#04	B	6 2 0 4 8	F
Path#05	C	7 <u>3</u> <u>1</u> <u>5</u> 8	E
Path#06	C	7 <u>3</u> <u>1</u> <u>5</u> 8	F
Path#07	C	7 <u>3</u> <u>1</u> <u>5</u> 9	G
Path#08	C	7 <u>3</u> <u>1</u> <u>5</u> 9	H
Path#09	A	6 <u>3</u> 7	C
Path#10	A	6 <u>3</u> 7	D

Over-Utilization Links: 6-2, 7-3, 4-8

Comparing to previous works, D<sup>2</sup>ENDIST is the most effective one in traditional networks. However, when software-defined networking (SDN) comes into play, its effectiveness should be re-evaluated since the resolution of network operations can be improved by SDN. That is, traditional routing only considers the end-to-end routing whereas the routing can be configured in a flow-based sequence in SDNs [8] [9].

## 2.2 SDN and OpenFlow

Software-defined networking (SDN) is a new concept where SDN separates the control and data planes of networks devices. Each network device can be controlled by the user-defined topology or the routing algorithm by decoupling the intelligence from switches to the central controller. OpenFlow [10] is one of the most common implementations for a SDN controller.

OpenFlow network is a one prevailing SDN network and contains flow tables to control all types of flows. The definition of a flow can be a web application, all packets to one country, or traffic from one host. As indicated in Table 2.3, OpenFlow defines the instruction set as an entry in a flow table to allow operators to customize their flow management. Therefore, different types of flows constitute the flow space of their own. The controller makes the same type of flows take the same action in the flow space, such as allow/deny flow, make flow private, remove flow, or reroute flow. Although it is far from complete realization of SDN, we can still acquire many advantages from OpenFlow networks. Without OpenFlow, researchers need to design a NetFPGA or other hardware solution first, which increases the complexity of constructing an experimental environment. OpenFlow can simplify the problems into software-defined scenarios. In addition, several versions of network OS control (e.g. NOX [11], POX, and floodlight [20]) in different programming languages are available. We can add/delete the rule by defining a NOX or POX, more efficient than rebuilding the entire hardware system.

Table 2.3: Flow Table Entry of OpenFlow

Behavior	Switch port	MAC src	MAC dst	IP Src	TCP dport	...	Action
<b>Switching</b>	*	aa:aa:.	*	*	*		port5
<b>Flow Switching</b>	port2	cc:cc:.	ee:ee:.	*	80		port6
<b>Firewall</b>	*	*	*	*	22		drop
<b>Routing</b>	*	*	*	5.6.7.8	*		port5

Another advantage of SDN is that the resolution of network operations is higher than traditional networks. For example, SDN enables the cross-layer routing while traditional networks typically perform routing on one target layer. Namely, SDN provides more information to assist the router in making decisions. Under this situation, the routing is evolved from end-to-end to per-flow.

Although the experiment can be easily conducted through SDN, it still encounters several problems. The first one is the security issue. The security mechanism is composed of millions of codes to support various intrusion detections. However, the freedom of SDN makes itself prone to different type of invasions. Another problem is the stability. Although SDN is highly flexible in developing new routing schemes, it lacks a certification for stability. The last one is the overhead of the central controller. The central-control mechanism helps the switches make optimal decisions. However, since all types of packets need to send queries to the controller, tremendous bandwidth consumption as well as extra system overhead will occur accordingly.

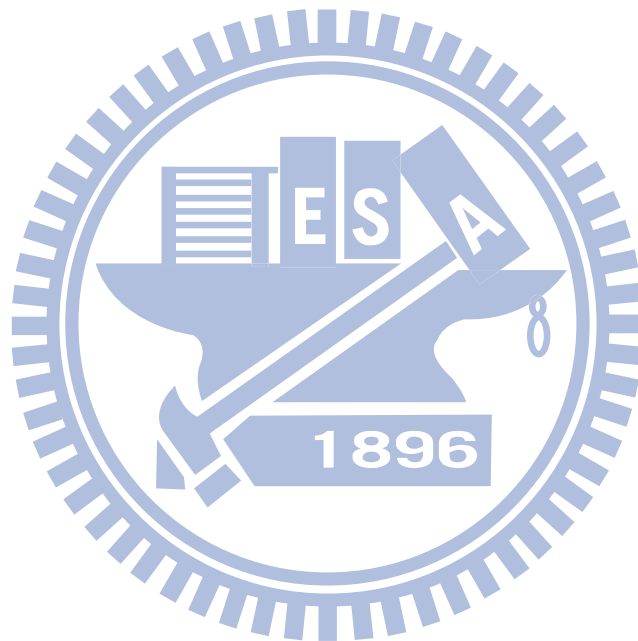
## 2.3 Literature Survey

SDN is used for enabling flexible per-flow routing where a flow can be defined across multiple network layers. Several previous works take advantages of SDN to compute paths with minimum cost on per-flow basis. With the support of SDN, CloudWatcher [12] considers locations of security devices (e.g. NIDS or firewalls) to find a path to transmit packets. OpenQoS [13] is a novel SDN controller that guarantees Quality of Service (QoS) for routing. However, *scalability* has not yet been properly addressed (only  $< 12$  routers were used in their experiments) and optimization of *network throughput* has also not been considered in current routing. Therefore, the first part of this work combines *traffic-aware flow migration* with a dynamic-reroute algorithm (DENDIST) into ***DENDIST-FM*** for SDN.

Furthermore, for datacenters, VM allocation typically helps to achieve high performance and meanwhile meets various service level agreements (SLAs). However, as the energy issue is becoming more important, VM allocation also needs to properly consider energy efficiency. Dynamic consolidation of VMs (through live migration and switching some PMs to sleep mode) is one of the solutions to save energy. As a result, novel VM management is proposed in [14] to increase performance of cloud applications and reduce operation cost of the computing infrastructure. Although the trade-off between performance and energy is discussed, linear programming used in [14] makes itself difficult to be scaled to large cases and thus limits its practicality.

Conventionally, VM migration for enhancing energy and network performances has been proposed separately in previous works. For example, an online algorithm [15] uses bounding techniques to enhance energy saving and machine utilization during VM migration. However, the network issue has not yet been addressed. Later, various VM allocation policies, such as [16] [17] [18], are also proposed to improve scalability of networks and to reduce unnecessary network traffic. Network topology and VM-to-VM communication are two keys to derive optimal solution. However, the energy issue is missing. As a result, the

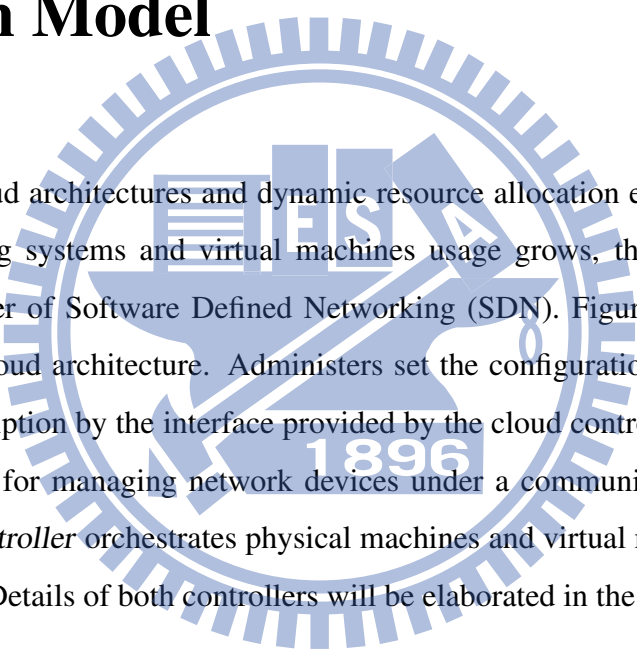
second part of this work proposes a technique called *energy-and-topology aware VM migration* (ETA-VMM) to improve energy saving and network performance, simultaneously.





## CHAPTER 3

### System Model



As elastic cloud architectures and dynamic resource allocation evolve while mobile computer operating systems and virtual machines usage grows, the need has arisen for an additional layer of Software Defined Networking (SDN). Figure 3.1 illustrates a typical SDN-based cloud architecture. Administrators set the configuration of high-level VMs and network description by the interface provided by the cloud controller. *The SDN controller* is responsible for managing network devices under a communications protocol whereas *The cloud controller* orchestrates physical machines and virtual machines to host versatile applications. Details of both controllers will be elaborated in the following sections.

#### 3.1 SDN Controller

A SDN controller manages network devices (e.g. switches, routers and etc.) by a communications protocol such as OpenFlow [10]. OpenFlow is the first standard interface between network controller and network devices. The OpenFlow protocol defines the flow types in a flow table to forward by matching rules like a instruction set of a CPU. The action sets in a flow table allows the flows to forward, drop, and so on, to control the behaviors of flows.

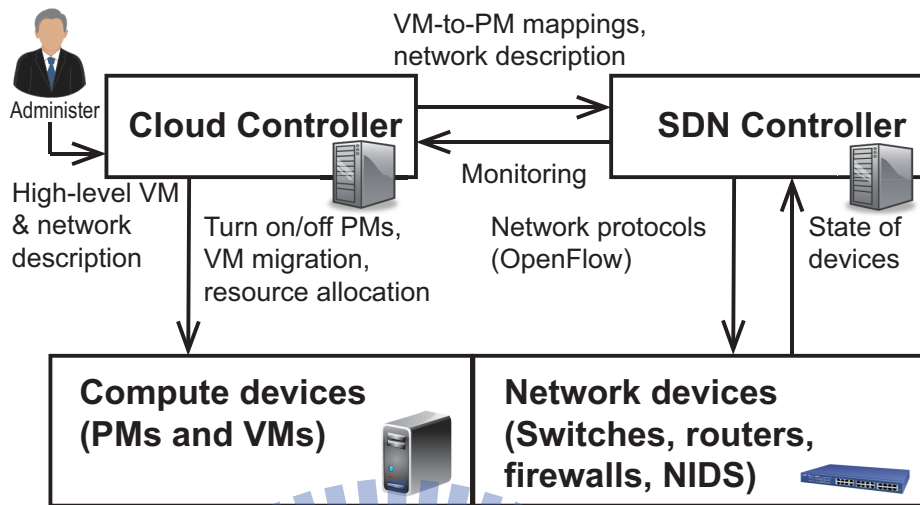


Figure 3.1: SDN-based Cloud Network Architecture

Besides, a network OS, such as NOX [11], Trema [19], floodlight [20] and OpenDaylight [21], modify flows as control management of SDN. SDN controller can support many functions with interaction of cloud controller and devices, including network monitoring, QoS support, flow-aware routing, and management of security devices.

## 3.2 Cloud Controller

Cloud controller serves as the hypervisor to manage VMs. Cloud controller is responsible for resource allocation (including CPU, memory, storage, and network bandwidth) for VMs according to their descriptions. Besides, turning on more physical machines (PMs) for fulfilling applications requests or turning them off for saving energy consumption are also the responsibility of cloud controller. As a result, administrators of cloud operators can retrieve these information from cloud controller of a datacenter. Moreover, cloud controller may also parse high-level network descriptions into commands to SDN controller to configure SDN-based switches [8].

## CHAPTER 4

# Dynamic Reroute with Flow Migration (DENDIST-FM) in SDN

As the demand of computing ability is rapidly increasing, tremendous and bursty traffics are generated between communicating VMs in datacenters. Network congestion on links will degrade performance of a datacenter network. As a result, there are two common aspects to improve performance: *topology* and *routing*. Topologies [3] [4] [5] of datacenters are categorized by the forms of multi-layers and multi-paths. Moreover, routing mechanism is important to avoid congestion and achieves load-balancing on a network topology. To the best of our knowledge, D<sup>2</sup>ENDIST [22] is one of the best routing algorithms to realize disjoint routing paths and dynamic reroute between hosts. However, when D<sup>2</sup>ENDIST is applied to SDN, its intimidating runtime complexity becomes a bottleneck for practicality. So, an alternative concept of flow control, *flow migration* (FM), is proposed for SDN. Later, flow migration (FM) and dynamic reroute (DENDIST) are merged and evolved into DENDIST-FM.

## 4.1 Problem of D<sup>2</sup>ENDIST in SDN

In traditional network architectures, switches need to exchange their own information and construct the forwarding table by sending hello-packets through the existing routing algorithms. Based on the forwarding table, the switches can transfer packets to the next hop. However, congestion may occur when massive flows with the same source and destination transmitted on the same path. For example, the routing path in Figure 2.2,  $6 - 2 - 0 - 4 - 8$ , starts from node 6 and ends at node 8. If a great number of flows (generated by the hosts under node 6 and node 8) run through this path, congestion may occur regardless of changing to any other back-up paths.

In a software-defined network, routing candidates are not statically determined only by switches. From the instruction set of OpenFlow, flows can be aggregated by the MAC/IP source address, destination address, VLAN ID, TCP ports and even the APP-ID (denoting the type of applications) [23] defined in the Open-Flow headers. A simple example is shown in Figure 4.1. The SDN controller can modify the address of the source or destination host in the flow table. In Table 2.2, path #1 to #4 share the same segments,  $6 - 2 - 0 - 4 - 8$ . In Table 4.1, such path is divided into two due to different source addresses. In this example, flows from the source address of host B (path #2 and #4) change to path:  $6 - 3 - 1 - 5 - 8$ . Similarly, path #5 and path #6 are modified due to different MAC addresses of the destinations (i.e. host E and F). As a result, the number of over-utilized links is reduced from 3 to 1.

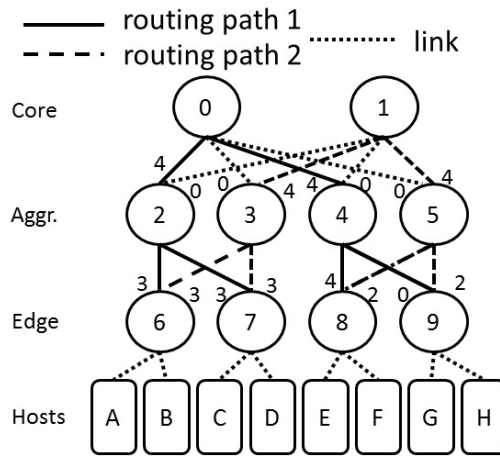


Figure 4.1: Example of SDN: D<sup>2</sup>ENDIST

Table 4.1: All Routing Path of SDN: D<sup>2</sup>ENDIST

	Source	Intermediate	Destination
Path#01	A	6 2 0 4 8	E
Path#02	B	6 <u>3</u> <u>1</u> <u>5</u> 8	E
Path#03	A	6 2 0 4 8	F
Path#04	B	6 <u>3</u> <u>1</u> <u>5</u> 8	F
Path#05	C	7 <u>2</u> <u>0</u> <u>4</u> 8	E
Path#06	C	7 <u>2</u> <u>0</u> <u>4</u> 8	F
Path#07	C	7 3 1 5 9	G
Path#08	C	7 3 1 5 9	H
Path#09	A	6 <u>2</u> 7	C
Path#10	A	6 3 7	D

Over-Utilization Links: 4-8

As mentioned before, SDN is a new network architecture, providing more flexible routing and finer traffic control over multiple network layers. So, D<sup>2</sup>ENDIST is not limited at the data-link layer in SDN. Moreover, traditional routing paths are typically fixed between the source switch (SS) and the destination switches (DS). If the hosts under SS and DS are mutually communicating, the path from SS to DS becomes paralyzed by massive data processing and serious packet loss will occur due to the limit of the buffer size.

As shown in Figure 4.2, when a dynamic-reroute algorithm is applied to SDN, the types of flows can be defined in many ways. First, the hosts under SS or DS consists of their own MAC addresses (Layer-2). Second, IP addresses (Layer-3) in host machines (PMs) or VMs are assigned by users. Third, different applications which consist of different TCP port numbers (Layer-4) can also be defined. Therefore, each types of flows can adjust its routing paths through dynamic reweight. As a result, the traffic loads in datacenter network are more balanced and the link utilization is improved.

Although the routing paths can be selected by different flow types, the computational complexity is increasing. Let  $N_s$  be the number of switches,  $N_h$  be the number of hosts in a topology, and  $N_f$  be the total number of flow types. The traditional routing algorithm - shortest-path bridge (SPB) [7], takes  $O(N_s^2)$  in time. D<sup>2</sup>ENDIST based on all-pair shortest paths also takes even  $O(N_s^3)$  in time. However, the number of flow types should be considered when the per-flow routing management in SDN is applied. The complexity of D<sup>2</sup>ENDIST in SDN becomes  $O(N_f * N_s^3)$ . If MAC addresses are taken as flow definition,  $N_f$  can be approximated as  $N_h * (N_h - 1)$ . Typically,  $N_h$  is much bigger than  $N_s$ . As a result, the total complexity of D<sup>2</sup>ENDIST can be larger than  $O(N_s^5)$ , which is computationally inefficient for real-world applications.

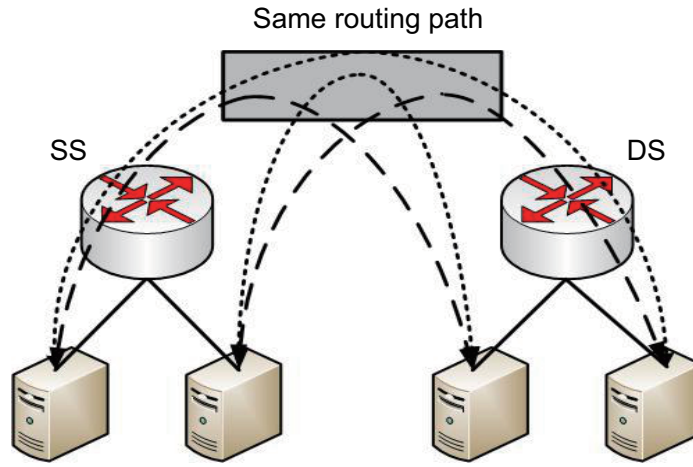


Figure 4.2: The bottleneck of traditional routing

## 4.2 Traffic-aware Flow Migration (FM)

Although D<sup>2</sup>ENDIST can achieve better throughput and balance traffic loads, the computation for routing paths per flows is too high to implement. As a result, we propose a scheme called traffic-aware flow migration (FM) as flow control on per-flow basis for SDN. It is proposed to solve the bottleneck of a traditional dynamic-reroute algorithm with the explosion of runtime complexity by the number of flow types.

Figure 4.3 shows the current link utilization of a network topology. To simplify the example, traffic load of each flow type is assumed the same where the number on each link denotes the number of flow types passing through the link (0 – 10). The upper-bound bandwidth on a link is 10. Over-utilized links refer to those links with the utilization rate exceeds 80% (i.e. more than 8 flows). In the example shown in Figure 4.3(a), link 2-0 is over-utilized detected by the SDN controller. The controller will choose one of flow types and retrieve its routing path in flow tables, If the routing path of this flow type is A-6-2-0-4-8-E, node 2 and 0 can be the substitution targets to avoid congestion on link 2-0. As a result, two possible reroute paths include (1) A-6-3-0-4-8-E or (2) A-6-2-1-4-8-E. The

path to replace depends on the comparison of traffic loads. In this example, path (2) will be chosen because path (1) has a higher utilization on link 3-0. The network status after flow migration is updated as Figure 4.3(b).

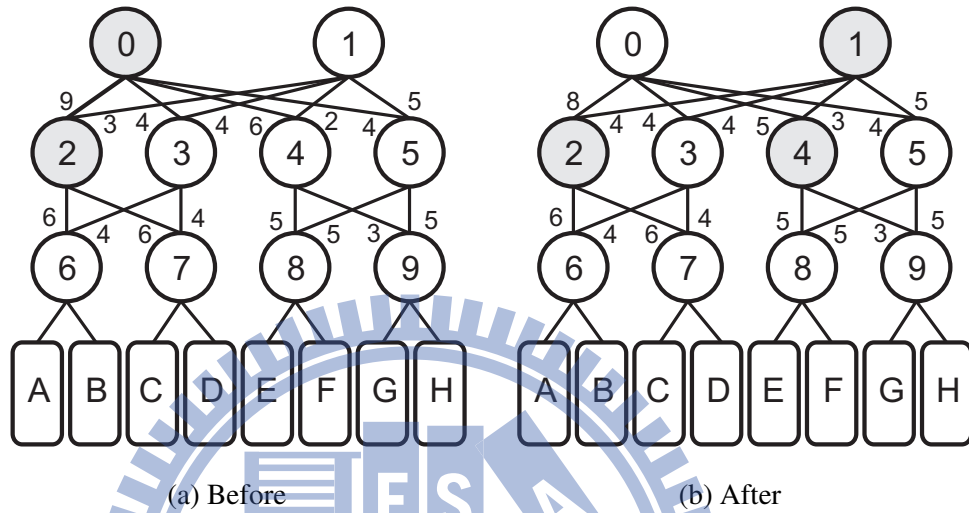


Figure 4.3: Example for flow migration

The main computational complexity of FM comes from finding one over-utilized links and choosing an alternative path. First, we are checking the total number  $N_l$  of links, iteratively. If the datacenter topology is a fully mesh structure,  $N_l$  is  $O(N_s^2)$ . However, for a fat-tree topology,  $O(N_l)$  is typically less than  $O(N_s^2)$ . Second, once one over-utilized link is detected, one of flow types (total number is  $N_f$ ) is chosen from the link with two possible edge nodes for substitution. Moreover, each edge node can only link to at most  $O(N_s)$  switches. Therefore, total runtime complexity for substitution becomes  $O(2*N_s)=O(N_s)$ , and makes FM efficient in practice.

Due to the symmetry in a fat-tree topology, an alternative path can always be found to balance traffic loads. In the traditional network, the routing paths are limited by the spanning tree where some of links are not used. Figure 4.4 shows two disjoint shortest path (SP) trees, where a routing path 6-2-0-4-8 (the primary path) needs to be switched to



the backup path due to the congestive link 6-2. However, link 1-5 is also congested and link 0-3, 0-5, 1-2, and 1-4 are wasted. After applying FM in SDN, those congested links can be quickly removed by substituting node 2 to node 3. Therefore, the new routing path 6-3-1-4-8 makes no unused link.

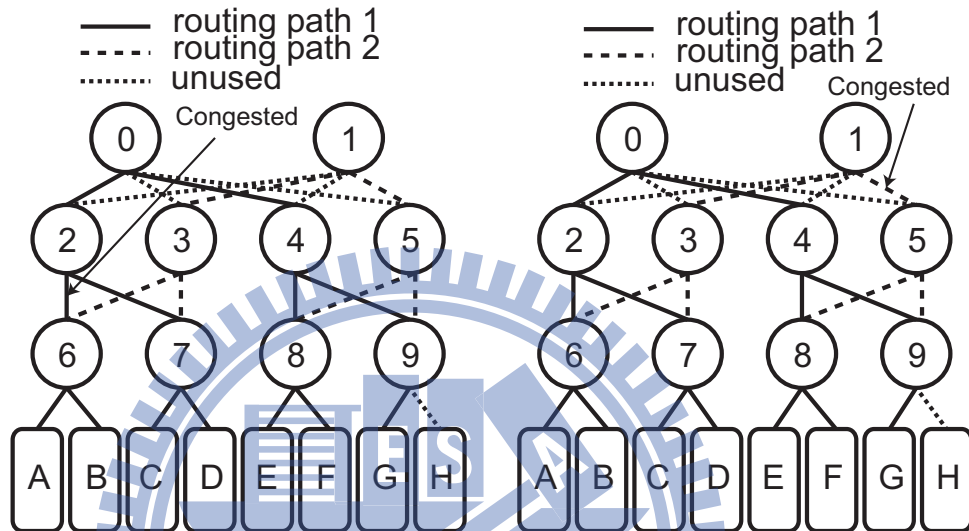


Figure 4.4: The drawback of D<sup>2</sup>ENDIST

### 4.3 DENDIST-FM

Traffic-aware flow migration (FM) is a more flexible and fast flow control algorithm in SDN. However, FM can also be benefited by disjoint reroute in balancing traffic loads at the beginning. As a result, *disjoint ENDIST* (a.k.a. DENDIST) is combined with FM, evolving into DENDIST-FM which takes less computation time than D<sup>2</sup>ENDIST.

Our flow-migration policy can be fully compatible with D<sup>2</sup>ENDIST. In Figure 4.1, link 4 – 8 remains over-utilized even if D<sup>2</sup>ENDIST is applied. Nevertheless, it can be quickly solved by flow migration. For example, path #01, #03, #05 and #06 all use the over-utilized link 4 – 8. Any one of these paths can be chosen to perform flow migration.

In Figure 4.5 and Table 4.2, the intermediate node 4 on path #05 is replaced to node 5. This method is intuitive but runs fast and effectively.



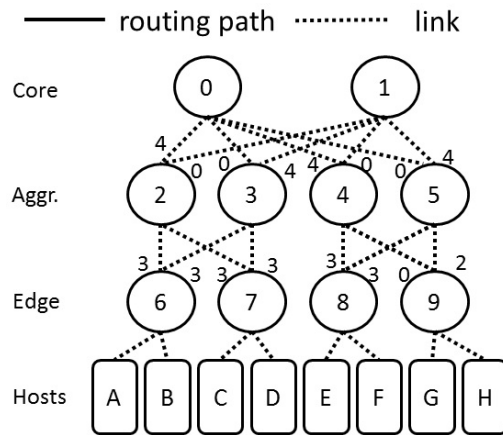


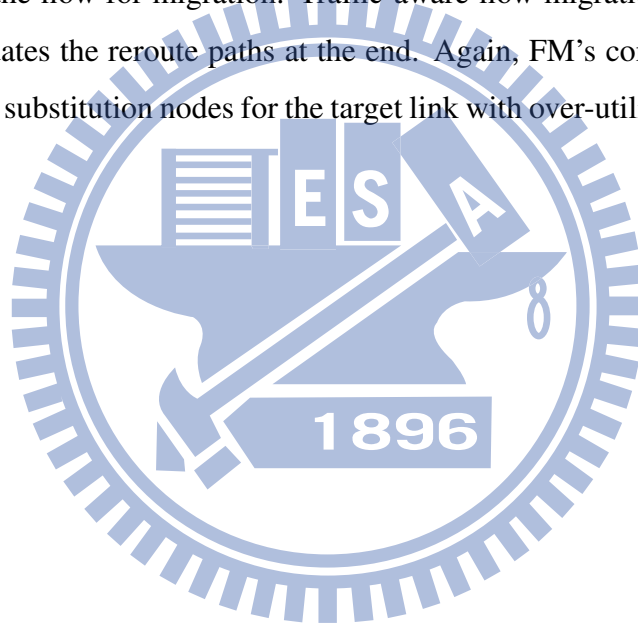
Figure 4.5: Example of SDN: DENDIST + FM

Table 4.2: All Routing Path of SDN: DENDIST + FM

	Source	Intermediate	Destination
Path#01	A	6 2 0 4 8	E
Path#02	B	6 3 1 5 8	E
Path#03	A	6 2 0 4 8	F
Path#04	B	6 3 1 5 8	F
Path#05	C	7 2 <u>0</u> <u>5</u> <u>8</u>	E
Path#06	C	7 2 0 4 8	F
Path#07	C	7 3 1 5 9	G
Path#08	C	7 3 1 5 9	H
Path#09	A	6 2 7	C
Path#10	A	6 3 7	D

Over-Utilization Links: none

DENDIST-FM is applied to a SDN-based cloud architecture as mentioned in Section II. Figure 4.6 shows the overall flow of DENDIST-FM. At the beginning, the SDN controller sends LLDP (Link Layer Discovery Protocol) [24] packets and waits for the response from switches to know their the neighbors. Therefore, the network topology is constructed after executing LLDP. Then, DENDIST is applied once to elevate throughput performance. Later, the SDN controller keeps polling the network status and collects link utilization periodically. Once any over-utilization is detected on a link, such link is targeted. Furthermore, the information about those flows running on this link are collected for choosing the flow for migration. Traffic-aware flow migration (FM) re-computes new paths and updates the reroute paths at the end. Again, FM's complexity is determined by the number of substitution nodes for the target link with over-utilization, and thus is  $O(N_s)$ .



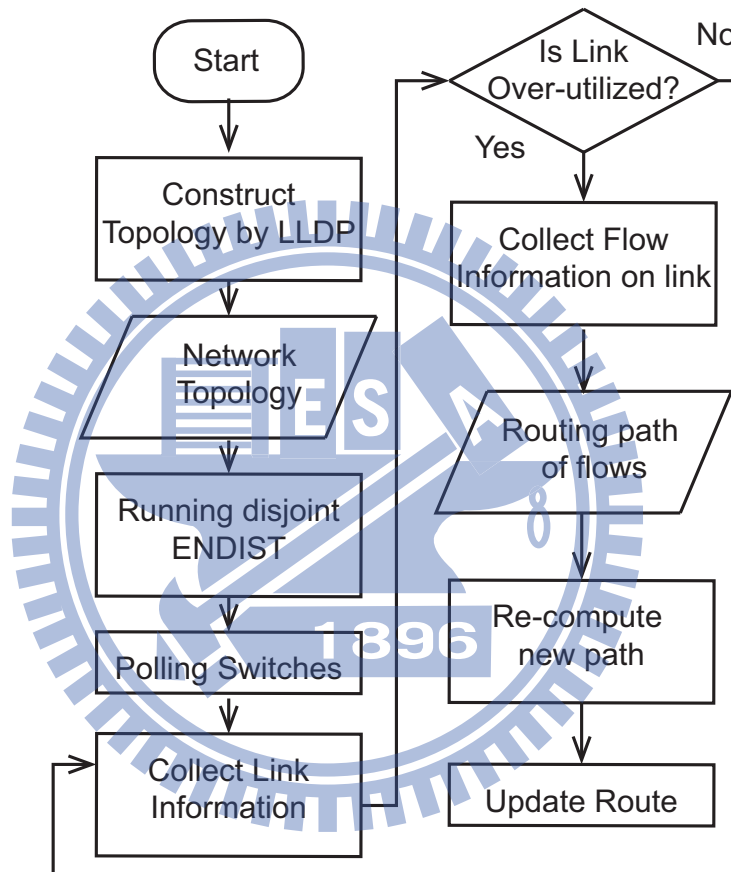


Figure 4.6: Flowchart of *DENDIST-FM*

## CHAPTER 5

# Energy-and-Topology-aware VM Migration (ETA-VMM)

Most of previous works on VM migration discuss energy saving of compute devices and throughput performance of networking, separately, in datacenters. Typically, the problem of VM migration is divided into two steps: (1) VM selection and (2) VM placement. Step (1) decides which VM needs to be migrated. Step (2) decides which host machine one picked VM is placed on. In this paper, we target both energy saving as well as throughput performance and propose *Energy-and-Topology Aware VM Migration* (ETA-VMM). In ETA-VMM, energy-aware VM selection first starts to detect over-/under-utilization of one host machine (PM). Second, network topology is also considered to facilitate VM placement.

### 5.1 Energy-aware VM Selection

Our energy-consumption reduction stems from migrating VMs on a under-utilized host to another and switching the empty host machine into the sleep mode. However, excessive

use of host machines may result in another reliability issue. Therefore, thresholds for over- and under-utilization are used to ensure the utilization of hosts in a reasonable range. The cloud controller keeps checking each host machine periodically with these thresholds.

## 5.2 Topology-aware VM Placement

The relationship between the datacenter topology and VM-to-VM communication is detailed in this section. The location of a selected VM is decided upon *network distance*, e.g. the number of hops or delay time between VMs. In this work, we take a different perspective. A fat-tree topology can be divided into four regions: *intra-host*, *intra-switch*, *intra-rack*, and *inter-rack*. As shown in Figure 5.1, different regions have different costs ranging from 1 (intra-host) to 4 (inter-rack). Once a VM is selected, the total cost of such VM is computed as the summation of costs from other communicating VMs.

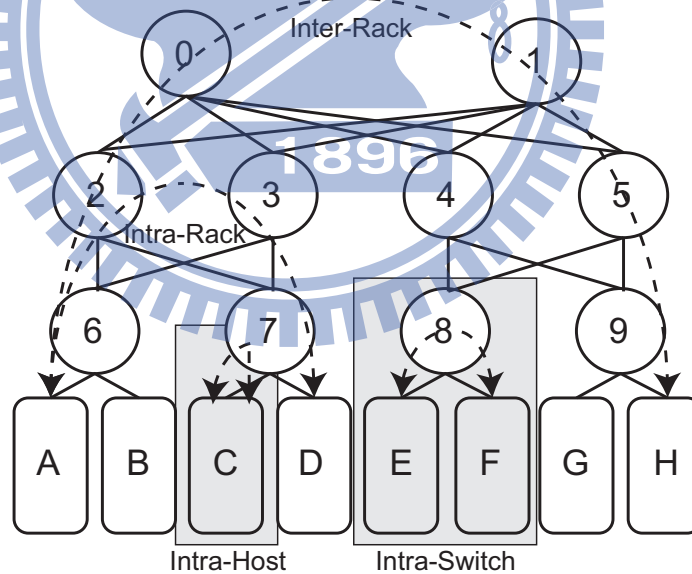


Figure 5.1: Regions in fat-tree topology

Figure 5.2 illustrates the overall flowchart of ETA-VMM. Upper and lower thresholds are checked to ensure the reasonable utilization of one machine. Once an over-utilized machine is found, the minimum migration policy [15] chooses one VM for migration. For the case of under-utilization, all VMs in the host machine are selected. Later, new host machines will be determined by VM placement on the basis of *network distance*. If no spare machine can be found to accommodate these VMs, the migration operation will be dismissed.

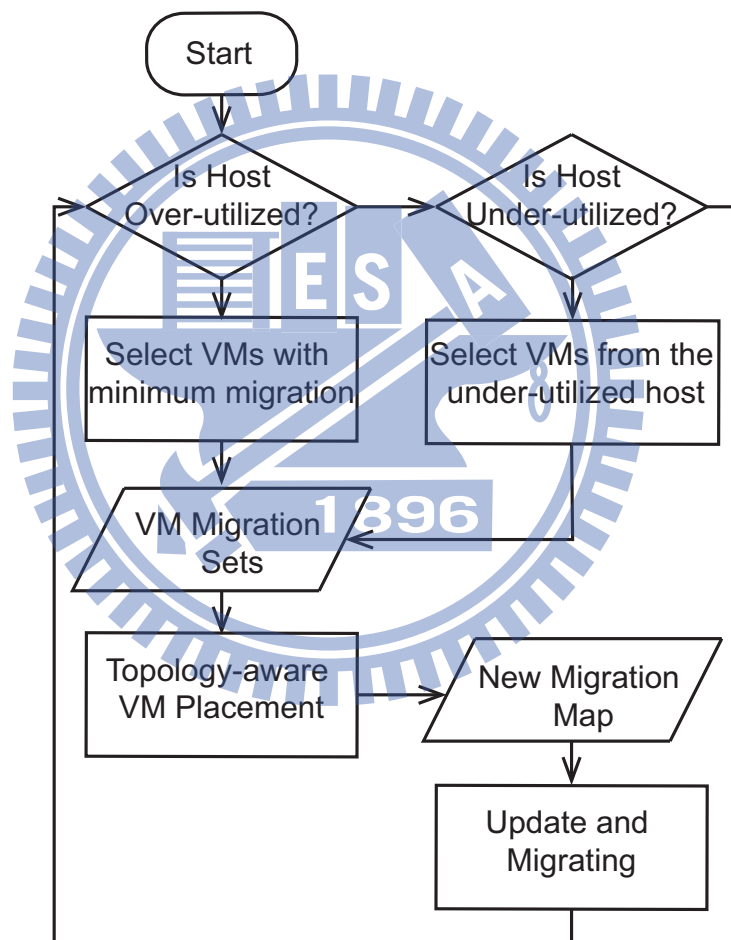


Figure 5.2: Flowchart of *ETA-VMM*



## CHAPTER 6

### Experimental Results

Network Simulator (NS2) v2.34 [25] and CloudSim v3.0 [26] are used in our experiments. NS2 provides the source routing to mimic the per-flow routing in SDN and evaluation of the network performance. CloudSim simulates the workload of host machines after VM deployment, and moreover, it can also calculate the energy consumption. The policy for VM deployment can be customized by users on CloudSim to evaluate SLA violation and energy consumption. The complete simulation environment also includes the interaction between NS2 and CloudSim.

Three fat-tree (FT) and BCube [5] topologies of different scales including small-scale, medium-scale and big-scale ones, are used in our experiments. The numbers of core-level, aggregate-level, and edge-level switches, host machines, and VMs are shown in Table 6.1. A large number of traffic flows are generated between communicating VMs where TCP traffic is modeled by a log-normal on-off model from [27] [28].

Table 6.1: Different scales of topologies

Scale	Core	Aggr.	Edge	Hosts	VMs
<b>FT-Small</b>	2	4	4	8	16
<b>FT-Medium</b>	5	10	10	20	40
<b>FT-Big</b>	5	10	50	500	1000
<b>BCube-Small</b>	4	4		16	32
<b>BCube-Medium</b>	8	8		64	128
<b>BCube-Big</b>	16	16		256	512

## 6.1 SDN v.s. Traditional Network

The first experiment shows performance difference between D<sup>2</sup>ENDIST in a traditional network and in SDN. One drawback of the traditional network is that it cannot prevent serious imbalance traffic from high-traffic volume. The traffic-imbalance ratio (TIR) is defined to represent the traffic volume from one specific path over the total traffic volume in the topology. Figure 6.2 shows the advantage of SDN compared to the traditional routing. SDN can have 60% improvement when the most of traffic is concentrated on some specific flows.

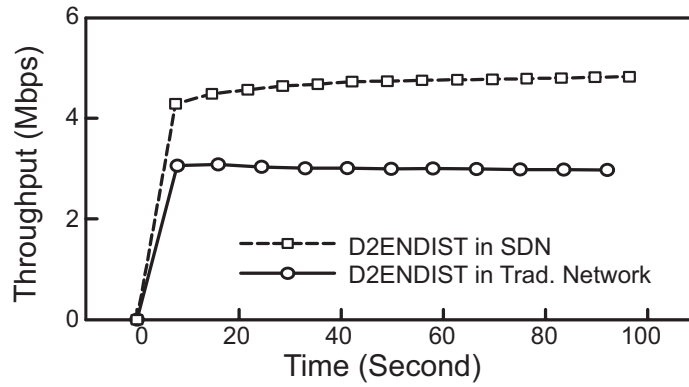


Figure 6.1: The adaptivity of SDN with imbalanced traffic loads

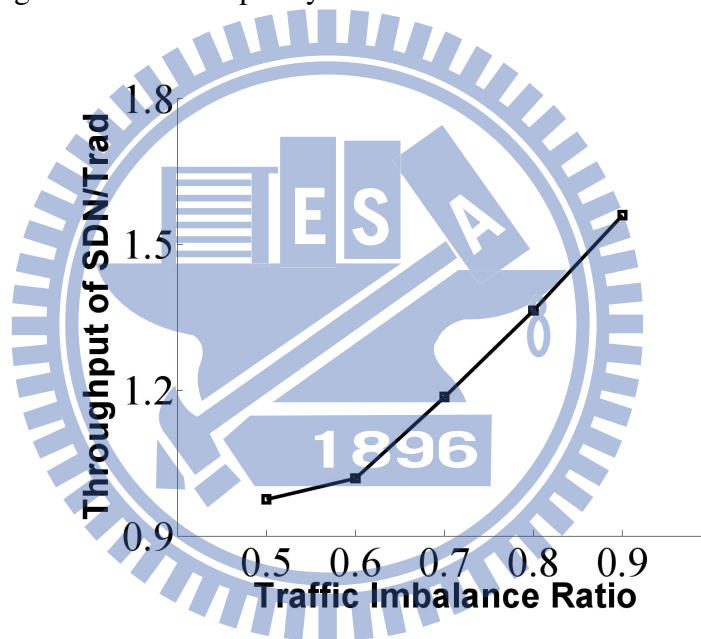


Figure 6.2: Different TIR

## 6.2 FM/DENDIST-FM Performance on Throughput

The second experiment compares the proposed traffic-aware flow migration (FM) with SPB and D<sup>2</sup>ENDIST. Figure 6.3 shows the throughputs of FM under two reroute periods. Compared to SPB, FM achieves throughput improvement by 96% and 113%, respectively, under 2-second and 1-second update periods. Note that the minimum period is determined by the polling time from the SDN controller. The shorter the polling time, the better the performance. However, FM can only achieve  $\sim 80\%$  throughput of D<sup>2</sup>ENDIST.

Therefore, we further conduct the experiment for DENDIST-FM, in which DENDIST is first applied for boosting throughput at the beginning followed by FM for throughput maintenance. Figure 6.4 shows the result of DENDIST-FM, comparing to D<sup>2</sup>ENDIST, FM alone and SPB. As one can see, performances of DENDIST-FM and D<sup>2</sup>ENDIST are comparable but DENDIST-FM uses a much lower runtime complexity than D<sup>2</sup>ENDIST does.

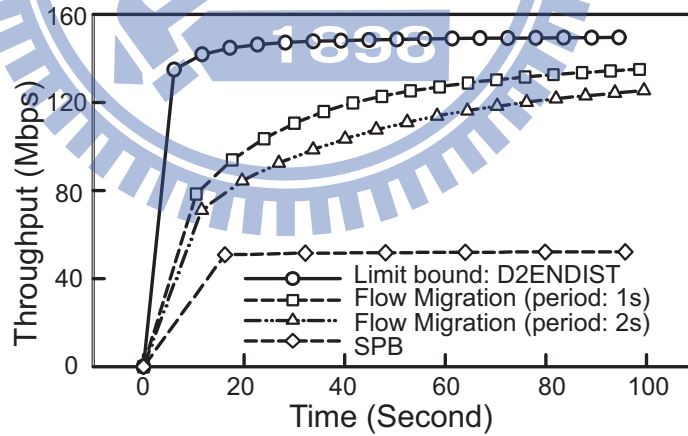


Figure 6.3: Throughput comparison of FM in different periods

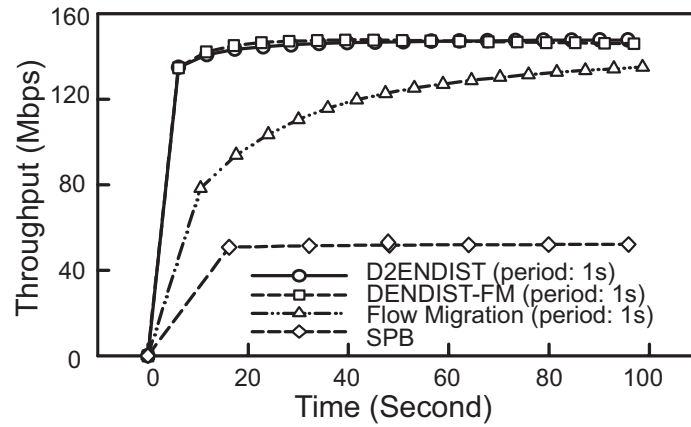
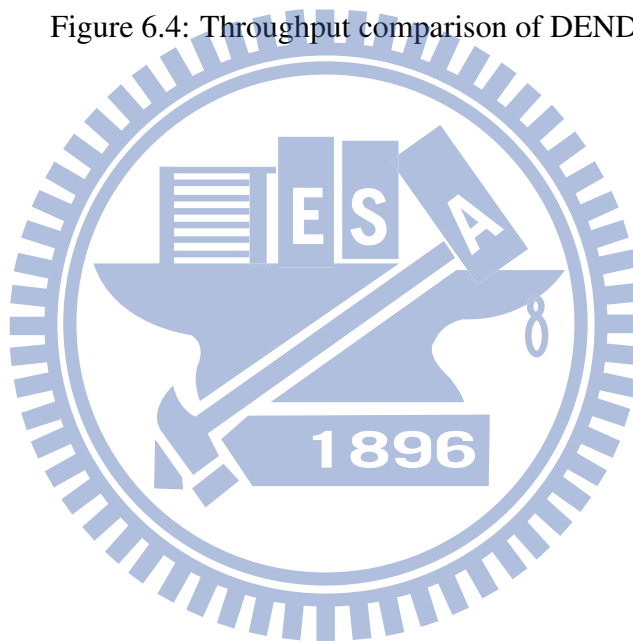


Figure 6.4: Throughput comparison of DENDIST-FM



### 6.3 Failure Recovery Time

Link failure is another critical issue in a datacenter network because it may lower the quality of service. Since the SDN controller keeps polling the status of switches to get responses from switches, it will quickly detect a link failure if one happens. With flow migration in SDN, the controller needs to neither frequently recompute the routing path nor collect all the information of topology for link-failure detection. However, it only modifies the routing path for flows with the failed link in the flow tables. According to our result, such process only takes 0.028 second to recover in a big-scale topology for one link failure. It is 2.1X faster than applying D<sup>2</sup>ENDIST [22] alone (which requires 0.06 second).

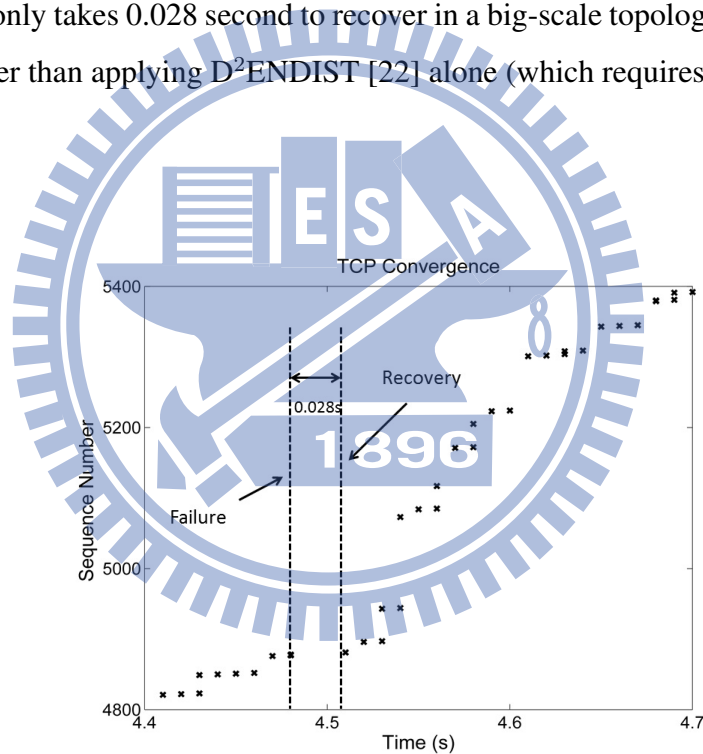


Figure 6.5: Recovery time for the network topology

## 6.4 ETA-VMM Performance on Energy-Consumption

In datacenters, energy consumption of a compute device is linearly proportional to its CPU utilization [15]. Energy-aware VM migration (EA-VMM) provided by CloudSim migrates VMs based on the utilization threshold. The host machine on which the migrating VMs are placed is determined by energy consumption. Our energy-and-topology aware VM migration (ETA-VMM) places the migrating VMs according to the communicating VMs. The analysis of energy consumption in CloudSim on a big-scale fat-tree topology is shown in Table 6.2. where the total simulation time is 1440 seconds. Energy consumption, the number of VM migrations, the number of host shutdowns, and SLA violation under different VM migration policies are shown. There are two kinds of SLA violations: performance degradation due to migration (migration costs) and violation time per active host (reliability of hosts). The former represents that performance of applications run in VMs will degrade during migration. The latter represents the ratio of hosts whose utilization exceeds the upper threshold. More details of SLA violation can be referred to [15].

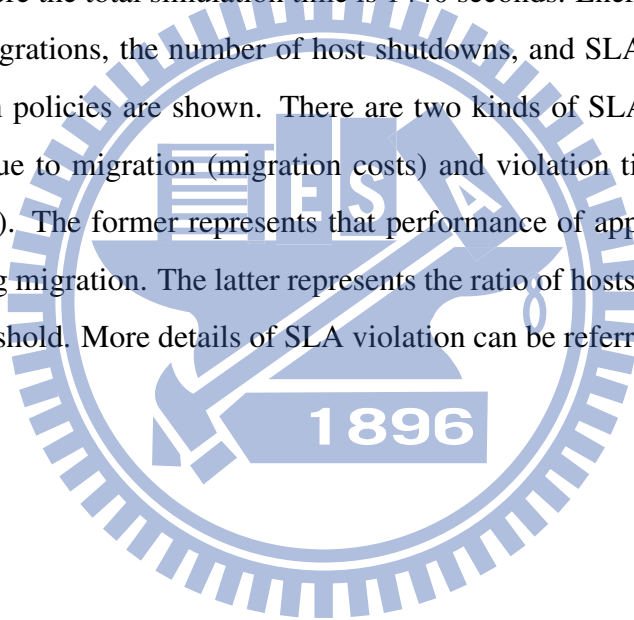


Table 6.2: Energy comparison of different VM migration policies

		<b>Without VMM</b>	<b>EA-VMM [15]</b>	<b>ETA- VMM</b>
<b>Energy consumption (kWh)</b>		10.47	7.18	7.44
<b>Number of VM migrations</b>		0	1605	1482
<b>Number of host shutdowns</b>		107	387	370
<b>SLA</b>	<b>Migration costs</b>	0%	0.45%	0.40%
<b>Vio.</b>	<b>Reliability of hosts</b>	0%	20.53%	19.14%

## 6.5 Throughput Improvement of Flow-and-VM Migration

NS2 is modified to adjust traffic for VM migration. Three scenarios, including SPB+EA-VMM, DENDIST-FM+EA-VMM and DENDIST-FM+ETA-VMM, are cross-compared in Figure 6.6 on a big-scale fat-tree topology. First, the original energy-aware VM migration (EA-VMM) with SPB results in the worst network performance because of ignoring communicating VMs. Second, for DENDIST-FM+EA-VMM, although DENDIST-FM is running, network throughput is limited because two communicating VMs may be placed distantly by EA-VMM. As a result, our combination (energy-and-topology aware VM migration (ETA-VMM) and DENDIST-FM) reaches the best network performance. Meanwhile, comparable energy-saving is also achieved by our flow-and-VM migration strategy (with only 3.2% energy overhead). Finally, simulation results of different VM migration policies and routing algorithms are summarized in Table 6.3.



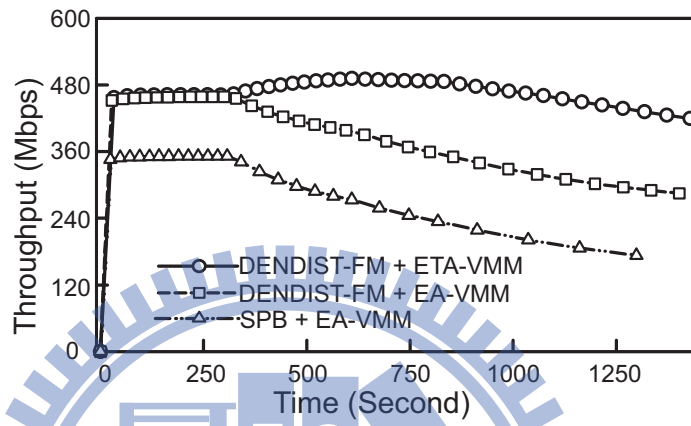


Figure 6.6: Throughput comparison of flow-and-VM migration

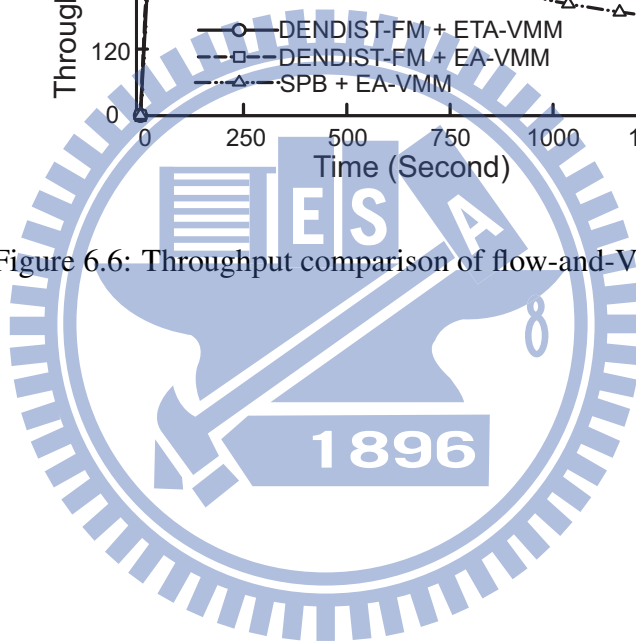


Table 6.3: Experimental results of different scale topologies

VMM	w/o VMM (a)		EA-VMM (b)		EA-VMM (c)		ETA-VMM (d)		Improvement Ratio			
	Energy Const. (kWh)	Avg. Thrght. (Mbps)	Energy Const. (kWh)	Avg. Thrght. (Mbps)	Energy Const. (kWh)	Avg. Thrght. (Mbps)	Energy Const. (kWh)	Avg. Thrght. (Mbps)	(a-d)/a	(b-d)/b	(d-a)/a	(d-b)/b
Routing	SPB											
Scale	Energy Const. (kWh)	Avg. Thrght. (Mbps)	Energy Const. (kWh)	Avg. Thrght. (Mbps)	Energy Const. (kWh)	Avg. Thrght. (Mbps)	Energy Const. (kWh)	Avg. Thrght. (Mbps)	Energy Const. (kWh)	Energy Const. (kWh)	Avg Thrght.	Avg Thrght.
	FT-S	9.54	0.15	9.74	0.15	10.99	0.15	13.41	11.8%	0%	40.6%	37.6%
FT-M	0.42	25.35	0.33	25.28	0.33	28.78	0.34	33.99	19.0%	-3.0%	34.1%	34.5%
FT-B	0.47	330.13	7.18	301.20	7.18	397.74	7.44	468.4	28.9%	-3.6%	41.9%	55.5%
BCube-S	0.34	13.11	0.23	13.27	0.23	13.74	0.25	19.91	26.5%	-8.7%	51.9%	50.0%
BCube-M	1.35	36.44	1.07	40.97	1.07	45.99	0.96	68.6	20.7%	10.3%	88.2%	67.4%
BCube-B	5.38	134.25	3.82	129.75	3.82	172.39	4.00	201.40	25.7%	-4.7%	50.0%	55.2%
Avg.									22.1%	-3.2%	51.1%	50.0%

## CHAPTER 7

### Conclusion

Traditionally, the VM migration policy and the routing algorithm are treated, separately. Therefore, this paper unifies flow migration and VM migration techniques into one efficient solution for optimizing throughput performance and maintaining comparable energy saving. Such solution is realized on a SDN-based cloud architecture, consisting of a Cloud controller and a SDN controller. As a result, our dynamic reroute with flow migration, DENDIST-FM, successfully reduces runtime complexity by four orders in the SDN controller. and meanwhile achieves comparable performance with D<sup>2</sup>ENDIST [22]. Besides, energy-and-topology aware VM migration (ETA-VMM) also demonstrates its effectiveness on energy saving and throughput improvement for the cloud controller. As a result, the unified solution including VM- and flow-migration strategies achieves throughput improvement by 50.0% with comparable energy saving (only 3.2% overhead) in our experiments.

The following issues are worthwhile for further investigation. First, the requirement of delay tolerance varies under different applications such as data integrity or real time streaming traffic. The Quality of Service (QoS) issue need to be considered so that the customers of cloud computing can choose suitable service for their applications. Second,

besides the energy consumption of compute devices, the energy consumption of network devices can be possibly reduced in datacenters. The global optimization of energy and network performance can be compared with the concurrent flow-and-VM migration scheme. Thirds, the implementation of the flow-control technique DENDIST-FM onto a network OS (e.g. NOX [11], Trema [19] or floodlight [20]) and the implementation of the VM-migration technique ETA-VMM onto a VM hypervisor (e.g. Hyper-V [29], Xen [30] or VMware's vCloud [31]) should be included in future work.



## Bibliography

- [1] K.T. Rao, P.S. Kiran, and L.S.S. Reddy, “Energy Efficiency in Datacenters through Virtualization: A Case Study,” in *Global Journal of Computer Science and Technology*, Vol 10, No 3, 2010.
- [2] *Software-Defined Networking: The New Norm for Networks*. <https://www.opennetworking.org/>
- [3] 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,” in *ACM SIGCOMM on Data communication*. New York, NY, USA: ACM, pp. 39–50, Aug 2009.
- [4] 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,” in *ACM SIGCOMM on Data communication*. New York, NY, USA: ACM, pp. 51–62, Aug 2009.
- [5] 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,” in *ACM SIGCOMM on Data communication*. New York, NY, USA: ACM, pp. 63–74, Aug 2009.
- [6] C. Suh, K. Kim, and J. Shin, “ENDIST: Edge node divided spanning tree,” in *10th International Conference on Advanced Communication Technology*, vol. 1, pp. 802–807, Feb 2008.
- [7] *Virtual Bridged Local Area Networks, Amendment 9: Shortest Path Bridging*, IEEE Std. 802.1aq/D0.3, 2006.
- [8] R. Raghavendra, J. Lobo, and K.-W. Lee, “Dynamic graph query primitives for sdn-based cloudnetwork management,” in *Proceedings of the first workshop on Hot topics in software defined networks*. New York, NY, USA: ACM, pp. 97–102, 2012.
- [9] S. Ghorbani, and M. Caesar, “Walk the line: consistent network updates with bandwidth guarantees,” in *Proceedings of the first workshop on Hot topics in*

- software defined networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 67–72. [Online]. Available: <http://doi.acm.org/10.1145/2342441.2342455>
- [10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: enabling innovation in campus networks,” in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 6974, 2008.
- [11] N. Gude, T. Koponen, J. Pettit, B. Pfaff, and M. Casado, “NOX: towards an operating system for networks,” in *ACM SIGCOMM Computer Communication Review* New York, NY, USA: ACM, pp. 105–110, July 2008.
- [12] S. Shin, and G. Gu, “Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?),” in *2012 20th IEEE International Conference on Network Protocols (ICNP)*, pp. 1–6, 2012.
- [13] H. Egilmez, S. Dane, K. Bagci, and A. Tekalp, “Openqos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks,” in *Signal Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pp. 1–8, 2012.
- [14] H. N. Van, F. Tran, and J.-M. Menaud, “Performance and power management for cloud infrastructures,” in *2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pp. 329–336, 2010.
- [15] A. Beloglazov, and R. Buyya, “Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers,” *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012. [Online]. Available: <http://dx.doi.org/10.1002/cpe.1867>
- [16] X. Meng, V. Pappas, and L. Zhang, “Improving the scalability of data center networks with traffic-aware virtual machine placement,” in *Proceedings IEEE INFOCOM*, pp. 1–9, 2010.
- [17] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, “Application-aware virtual machine migration in data centers,” in *Proceedings IEEE INFOCOM*, pp. 66–70, 2011.
- [18] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, “Application-aware virtual machine migration in data centers,” in *Proceedings IEEE INFOCOM*, pp. 2876–2880, 2012.
- [19] *Trena*. <http://trena.github.io/trema/>
- [20] *Floodlight*. <http://floodlight.openflowhub.org/>
- [21] *OpenDaylight*. <http://www.opendaylight.org/>

- [22] G.-H. Liu, H.-P. Wen, and L.-C. Wang, “D<sup>2</sup>ENDIST: Dynamic and disjoint endist-based layer-2 routing algorithm for cloud datacenters,” in *IEEE Global Communications Conference (GLOBECOM)*, pp. 1611–1616, Dec. 2012.
- [23] P. Lin, J. Bi, and H.Hu. “VCP: A virtualization cloud platform for SDN intra-domain production network,” in *IEEE International Conference on Network Protocols (ICNP)*, pp. 1-2, 2012.
- [24] *IEEE Standard. LLDP*. <http://standards.ieee.org/getieee802/download/802.1AB-2009.pdf>
- [25] *Network simulator ns-2* <http://www.isi.edu/nsnam/ns>
- [26] *ClousSim*: <http://www.cloudbus.org/cloudsim/>
- [27] T. Benson, A. Anand, A. Akella, and M. Zhang, “Understanding data center traffic characteristics,” *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 92–99, Jan. 2010.
- [28] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “The nature of data center traffic: measurements & analysis,” in *ACM SIGCOMM on Internet Measurement*. New York, NY, USA: ACM, pp. 202–208, Nov 2009.
- [29] *Hyper-V*. <http://www.microsoft.com/hyper-v-server/>
- [30] *Xen*. <http://www.xenproject.org>
- [31] *vCloud*. <http://www.vmware.com/products/datacenter-virtualization/vcloud-suite/>