การออกแบบและพัฒนาหน้าต่างบนเว็บสำหรับอิมูเลเตอร์เครือข่าย
A Design and Development of a Web Interface for Network Emulators

ธงชัย เจือจันทร์[1], นพรัตน์ โพธิ์สิงห์[1], เอกราวี คำแปล[1], สมสวัสดิ์ นิลดำ[1]
Thongchai Chuachan[1], Nopparat Posing[1], Ekrawee Kamplae[1], Somsawut Nindam[1]
Received: 23 March 2016; Accepted: 12 June 2016

บทคัดย่อ
อิมูเลเตอร์เป็นหนึ่งในเครื่องมือที่สำคัญสำหรับการทดลองด้านเครือข่าย แต่การใช้อิมูเลเตอร์เครือข่ายยังคงมีปัญหาของการติดตั้ง คอนฟิก และสนับสนุนได้เพียงบางระบบปฏิบัติการ งานวิจัยนี้จึงจะพัฒนาเว็บอิมูเลเตอร์ที่มีคุณสมบัติพื้นฐานเดียวกันกับอิมูเลเตอร์ที่ทำงานแบบบนเครื่องคอมพิวเตอร์เนทีฟ โดยการพัฒนาหน้าต่างที่ทำงานบนเว็บ ต่อยอดการสร้างและจัดการโหนดทางเครือข่ายด้วยอิมูเลเตอร์ CORE ซึ่งใช้ Node.js พัฒนาให้มีการทำงานแบบ RESTful แล้วทำการทดสอบประสิทธิภาพและประสิทธิผลทั้งความล่าช้าต่อการตอบสนอง และประสิทธิภาพของระบบ จากผลการทดลองพบว่าโปรแกรมจากงานวิจัยนี้สามารถตอบสนองต่อผู้ใช้ได้รวดเร็ว สามารถรองรับการสร้างและจัดการโหนดทางเครือข่ายได้จำนวนมาก และทำงานบนบราวเซอร์ได้อย่างมีประสิทธิภาพ

คำสำคัญ: อิมูเลเตอร์ เทสเบด เครื่องมือทดลองเครือข่าย

Abstract
Network emulation tools are essential for network learning and testing. However, early network emulation tools have some issues, particularly with installation and configuration. Moreover, they depend on operating systems. In this research a web-based emulation has been developed to work the same as previous prominent network emulation tools. So, a Common Open Research Emulator (CORE) has been selected to provide a basis network node. We experimented to measure delay and efficiency of our prototype. Experimental results have shown that the web-based emulator has effectively provided for a number of users and network nodes.

Keywords: Emulator, test-bed, Network testing tools

บทนำ
การจำลองระบบเครือข่าย (Network Simulation)[1] ถูกใช้เป็นเครื่องมือสำหรับลองด้านเครือข่าย เพื่อพิสูจน์แนวคิดจากการออกแบบและพัฒนาโปรแกรมเครือข่ายต้นแบบ โดยวิธีจำลองระบบเครือข่ายช่วยให้การติดตั้ง การกำหนดโทโพโลยี (Topology) และการแบ่งเครือข่าย (Network Separation) ทำได้ง่าย ทั้งในเครือข่ายขนาดเล็ก (Small Scale Network) และเครือข่ายขนาดใหญ่ (Large Scale Network) แต่ต้องอาศัยการควบคุมค่าพารามิเตอร์ (Parameter) ที่เหมาะสม และมีความเข้มงวดทางสถิติดัง Millman และคณะ[2] ได้นำเสนอไว้ สำหรับการทดสอบเครือข่ายแบบ test-bed จะเป็นใช้วิธีนำโปรแกรมต้นแบบไปทดสอบกับระบบเครือข่ายจริงและได้รับความน่าเชื่อถือจากการการถูกทดสอบบนกลุ่มฮาร์ดแวร์จริง เช่น PlanetLab[3] แต่อาจมีราคาสูง รองรับจำนวนผู้ใช้ได้น้อย และยากต่อการติดตั้งเพื่อดำเนินการทดสอบอุปกรณ์จำนวนมากได้ ทำให้ทั้ง Network Simulation และ test-bed จึงมีทั้งจุดดีและจุดด้อยที่แตกต่างกันสูง

[1] *อาจารย์, สาขาวิทยาการคอมพิวเตอร์, ภาควิชาวิทยาศาสตร์พื้นฐาน, คณะวิทยาศาสตร์และเทคโนโลยี, มหาวิทยาลัยราชภัฏสุรินทร์.*
E-mail: {thongchai.c@srru.ac.th, n_posing@hotmail.com, civiclove0326@hotmail.co.th, somsawut.nindam@gmail.com}
[1] *Lecturer, Department of Computer Science, Faculty of Science and Technology, Surindra Rajabhat University. E-mail: {thongchai.c@srru.ac.th, n_posing@hotmail.com, civiclove0326@hotmail.co.th, somsawut.nindam@gmail.com}*

หลายหน่วยวิจัยจึงมีแนวคิดที่จะใช้การเลียนแบบ (Emulation)[4–7] ขึ้นจากการใช้เทคโนโลยี Virtualization มาช่วยแก้ปัญหาต่างจากทั้งการใช้ Network Simulation และ test-bed โดย Emulation นี้จะสามารถประมวลผลจากฟังก์ชันจริงของโปรแกรมต้นแบบ (Functional Realism) เวลาในการประมวลผลเป็นไปตามจริง (Timing Realism) การใช้เครือข่ายจริง (Traffic Realism) การติดตั้งและกำหนด Topology เครือข่ายมีความยืดหยุด (Topology Flexibility) และมีราคาต่ำ (Low Cost) Emulation จึงเป็นแนวทางที่สมดุลต่อการทดลองระบบเครือข่าย โดยใช้ติดตั้งโปรแกรมต้นแบบได้จริง แยกการเชื่อมต่อเครือข่ายภายในระบบ และทดสอบด้วยวิธี test-bed ได้

มีหลายซอฟต์แวร์ที่ถูกพัฒนาจากเทคนิค Emulation เช่น Mininet[8], Cloonix[9], Netkit[10], Integr- ated Multiprotocol Network Emulator/Simulator (IMMUNES)[11] และ Common Open Research Emu- lator (CORE)[7] ด้วยการใช้วิธีสร้างโหนดทางเครือข่ายจากเครื่องมือต่าง ๆ เช่น Linux Container[12] และ Kernel-based Virtual Machine (KVM)[13] แต่ยังยากที่นักวิจัยติดตั้ง คอนฟิก และจัดการระบบพื้นฐาน ก่อนนำไปใช้เพื่อการทดลองและการศึกษาด้านเครือข่าย เพราะต้องใช้เวลาในการติดตั้งและจัดการระบบสูง

งานวิจัยนี้จึงพัฒนา Graphical User Interface (GUI) บนเว็บสำหรับ Emulation ที่มีความสามารถใกล้เคียงกับระบบ User Interface (UI) แบบผู้ใช้เดี่ยว เช่น CORE เป็นต้น โดยการพัฒนาเว็บแอพพลิเคชัน ช่วยให้การจัดการ การทดลองด้านเครือข่าย มีความสะดวกต่อการนำไปใช้ สามารถนำไปทดสอบโพรโท-คอล ทางเครือข่ายที่มีอยู่ในปัจจุบัน และโพรโทคอลที่พัฒนาขึ้นใหม่ได้ เช่น เครือข่ายสถาปัตยกรรม TCP/IP และรวมถึงเครือข่ายแบบใหม่อย่าง Named Data Networking (NDN)[14] ได้

การพัฒนาหน้าต่าง Emulator บนเว็บ (Web Emulator) นี้ ใช้ Node.js[15] เป็นเครื่องมือเชื่อมต่อระหว่างหน้าต่างบนเว็บแอพพลิเคชันกับอิมูเลเตอร์ CORE โดยได้พัฒนากลไกการแปลงรูปแบบการคอนฟิกโหนดทางเครือข่ายของ CORE และหน้าต่าง Web Emulator ที่ทำงานแบบ RESTful[16] แล้วประเมินประสิทธิภาพและประสิทธิผลของระบบไม่ต่ำกว่า 30 ครั้ง ที่ความเชื่อมั่นร้อยละ 95 พบว่า Web Emulator จากงานวิจัยนี้ช่วยเพิ่มประสิทธิภาพและประสิทธิผลของ Emulator ได้เป็นอย่างดี

วัตถุประสงค์

1. เพื่อออกแบบและพัฒนากลไกเชื่อมโยงการทำงานระหว่างหน้าต่างอิมูเลเตอร์บนเว็บกับโหนดทางเครือข่ายของอิมูเลเตอร์ CORE

2. เพื่อพัฒนาหน้าต่างเว็บอิมูเลเตอร์ที่มีประสิทธิภาพใกล้เคียงกับโปรแกรมบนเดสก์ท็อป

งานวิจัยที่เกี่ยวข้อง

โปรแกรมจำลองเครือข่าย

โปรแกรมจำลองเครือข่าย (Network Simulation Tools) ถูกใช้เป็นเครื่องมือที่สำคัญสำหรับการศึกษาและวิจัยด้านเครือข่าย และได้มีการพัฒนาให้มีความแม่นยำสูงขึ้น เพื่อให้ผลการทดลองเครือข่ายมีความน่าเชื่อถือ เช่น Network Simulator 2 (ns2)[17], Network Simulator 3 (ns3)[18] และ Packet Tracer[19] เป็นต้น แต่อย่างไรก็ตามโปรแกรมเหล่านี้ยังมีข้อบกพร่องหลายด้าน เช่น การใช้ ns2 และ ns3 ในการทำงานวิจัยด้านเครือข่าย ต้องมีการพัฒนาต้นแบบที่อยู่ในรูปของการจำลอง การควบคุมค่าพารามิเตอร์ที่ซับซ้อน และซอฟต์แวร์ที่พัฒนาขึ้นถูกสั่งงานบนระบบที่ถูกจำลองแบบคร่าว ๆ ซึ่งไม่สามารถนำไปปรับใช้เป็นซอฟต์แวร์ที่ใช้งานได้จริง สำหรับโปรแกรมที่ใช้เรียนรู้เครือข่ายอย่าง Packet Tracer เป็นเพียงการจำลองแบบเบื้องต้น ไม่สามารถพัฒนาซอฟต์แวร์ต้นแบบและนำไปทดลองได้ การนำโปรแกรมจำลองเครือข่ายมาใช้จึงต้องควบคุมการทดลองเครือข่ายอย่างเข้มงวด งานวิจัยนี้จึงมีแนวคิดที่จะพัฒนาโปรแกรมที่จะช่วยให้เพิ่มประสิทธิภาพเครื่องมือการเรียนรู้และการทำงานวิจัยด้านเครือข่ายที่ดีขึ้น

การทดสอบแบบ test-bed

การทดลองด้วยวิธีใช้ test-bed ก่อนหน้านี้จะถูกใช้ก็ต่อเมื่อแนวคิดของเครือข่ายมีความเชื่อมั่นสูง จึงพัฒนาโปรแกรมต้นแบบและทดลองกับระบบเครือข่ายจริง ผลการทดสอบระบบจึงได้จากการทำงานบนระบบปฏิบัติการ และฮาร์ดแวร์จริง และการทดลองสามารถควบคุมไหลของแพ็กเก็ตได้หลายแบบ เช่น ทดสอบแบบปิด และทดสอบแบบเปิด เป็นต้น แต่การทดสอบด้วยวิธี test-bed ยังมีต้นทุนสูง ยุ่งยากต่อการจัดการทั้งด้านซอฟต์แวร์และฮาร์ดแวร์ เปลี่ยนทอพอโลยีของเครือข่ายได้ยาก และใช้เวลาคอนฟิกสูง เช่น PlanetLab[3] และ ORBIT[20] เป็นต้น จึงเกิดความยุ่งยากต่อการนำไปใช้กับกลุ่มผู้วิจัยที่มีทรัพยากรณ์ไม่เพียงพอ งานวิจัยนี้จึงหาแนวทางที่จะทำให้ระบบการทดสอบและเรียนรู้เครือข่ายโดยใช้ test-bed ได้ และมีประสิทธิภาพที่ดีขึ้น

การเลียนแบบเครือข่าย

การเลียนแบบเครือข่าย (Network Emulation) เป็นวิธีเลียนแบบอุปกรณ์คอมพิวเตอร์ เช่น เครื่องคอมพิวเตอร์ เร้าเตอร์ และอื่น ๆ ที่มีระบบปฏิบัติการ พร้อมที่จะทำงานโดย

มีฟังก์ชันทำงานเหมือนกับของจริงทุกประการ และเทคนิคนี้ถูกใช้โดยทั่วไปในหน่วยของ Virtualization โดยการใช้ Network Emulation นี้จะสร้างโหนดทางเครือข่ายเป็น Virtual Machine (VM) มีระบบปฏิบัติการรองรับการทำงานของโปรแกรมจริง เช่น การใช้ Linux Container (LXC)[12] และ Kernel-based Virtual Machine (KVM)[13] เป็นต้น ช่วยให้ระบบปฏิบัติการใน VM ถูกเชื่อมต่อกับเครือข่ายทั้งจากการจำลองและเครือข่ายจริงได้

การนำโปรแกรม Network Emulation มาใช้สำหรับการทดลองและเรียนรู้ด้านเครือข่ายจึงสามารถช่วยลดต้นทุนด้านฮาร์ดแวร์ ความยากต่อการติดตั้ง และสามารถทดสอบเครือข่ายด้วยวิธี test-bed ได้ ก่อนหน้านี้มีโปรแกรม Network Emulation ที่ถูกพัฒนาขึ้น เพื่อใช้สำหรับงานวิจัยหลายโปรแกรม เช่น Integrated Multiprotocol Network Emulator (IMMUNE)[11], Cloonix[9], Mininet[8], Netkit[10], Open Network Emulator[5] และ Common Open Research Emulator (CORE)[7] เป็นต้น โดยการทดสอบเครือข่ายส่วนใหญ่ในปัจจุบันโปรแกรม Network Emulation ถูกนำมาใช้อย่างกว้างขวาง เช่น การพัฒนา Software Defined Network (SDN)[21] ถูกนำไปติดตั้งและทดสอบบนโปรแกรม Mininet โดยกลุ่มนักวิจัย SDN เป็นต้น การนำโปรแกรม Network Emulation มาใช้จึงมีความสำคัญกับกลุ่มนักวิจัยและกลุ่มผู้เรียนด้านเครือข่ายในปัจจุบัน

Common Open Research Emulator (CORE)

CORE[7] เป็นหนึ่งในโปรแกรม Network Emulator ที่สามารถสร้างและจัดการเครือข่ายสำหรับการทดลองและการเรียนรู้ด้านเครือข่ายได้ดี ซึ่งพัฒนาโดยใช้กลไกจัดการเครือข่ายของ FreeBSD สำหรับเชื่อมต่อกับเครือข่ายจำลองและอุปกร์เครือข่ายจริง และช่วยให้ออกแบบและแบ่งส่วนของเครือข่ายที่เชื่อมต่อได้ในระดับเฟรมแพ็กเก็ตและสูงขึ้น โดยใช้โพรโทคอลทางเครือข่ายจริง แต่ CORE คงมีปัญหาหลายด้าน เช่น ทำงานได้เพียงบนระบบปฏิบัติการลินุกส์ การคอนฟิกเพื่อใช้ในการเรียนรู้และทดลองทางเครือข่ายแบบกลุ่ม และการคอนฟิก CORE เบื้องต้นทำได้ยาก ซึ่งงานวิจัยนี้ได้เล็งเห็นและแก้ประเด็นปัญหาเหล่านี้

Table 1 CORE and Web Emulator configuration format

CORE imn	WebEmu
node n3 {	version: '0.2',
type router	details: ' JSON WebEmu : Configuration file',
model router	nodes:[
network-config {	{
hostname n3	"id":"n3",
interface eth1	"type":"router",
ip address 10.0.0.20/24	"model":"router",
interface eth0	"peer":["n2","n6"],
ip address 192.168.6.240/24	"coords":{"x":316,"y":128},
}	"lbcoords":{"x":316,"y":160},
canvas c1	"infs":[
iconcoords {316.0 128.0}	{"interface":"eth1","ip":"10.0.0.20","ifnet":"10.0.0.20/24"},
labelcoords {316.0 160.0}	{"interface":"eth0","ip":"192.168.6.240","ifnet":"192.168.6.240/24"}
services {IPForward}	]
interface-peer {eth0 n2}	}]
interface-peer {eth1 n6}	
}	

การออกแบบและพัฒนา Web Emulator

การเลือก Emulator พื้นฐาน

การพัฒนาหน้าต่าง Web Emulator ในงานวิจัยนี้เลือกใช้ Common Open Research Emulator (CORE) เป็นระบบพื้นฐาน จากที่ Ahrenholz[7] ได้แสดงผลการทดสอบประสิทธิภาพของการทดลองเครือข่ายได้ดี เพิ่มโหนดทางเครือข่ายได้จำนวนมาก CORE จึงเป็นตัวเลือกที่สำคัญในการนำมาพัฒนาหน้าต่าง Web Emulator ต่อยอดได้ดี

โครงสร้างระบบ

โครงสร้างของ Web Emulator ดังแสดงใน (Figure 1) จะประกอบด้วย 3 ส่วน คือ หน้าต่าง Web Emulator สำหรับผู้ใช้ (WebEmu Interface) ส่วนการเชื่อมต่อระหว่างโหนดทางเครือข่ายกับหน้าต่างสำหรับผู้ใช้ (WebEmu Plane) และส่วนของโหนดทางเครือข่าย (VMs Plane)

โครงสร้างของ Web Emulator นี้ต้องมีการ์ดเครือข่าย (Network Interface) อย่างน้อย 2 การ์ด โดยการ์ดแรกดัง Interface 0 ใน (Figure 1) จะเป็นส่วนติดต่อกับหน้าต่าง WebEmu Interface ซึ่งเป็นหน้าต่างหลักสำหรับควบคุมโหนดทางเครือข่าย ส่วนการ์ดเครือข่าย Interface 1 ใน (Figure 1) ถูกเชื่อมต่อกับเครือข่าย Bridge เพื่อทำหน้าที่แบ่งส่วนของเครือข่ายและรูปแบบการเชื่อมต่อของเครือข่าย ซึ่งแต่ละส่วนของ WebEmu Interface, WebEmu Plane และ VM Configurations จะมีขั้นตอนการทำงานดังนี้

1. WebEmu Interface เป็นหน้าต่างติดต่อกับผู้ใช้ที่มีความสามารถอัพโหลดคอนฟิกของ CORE แล้วแปลง (Convert) ค่าคอนฟิกให้สนับสนุนการทำงานของ Web Emulator ได้ และสามารถเฝ้าสังเกต (Monitoring) สถานะของโหนดทางเครือข่าย และโฮสของเครื่องทดสอบได้ โดยการเลือกโหนดทางเครือข่ายสามารถแสดงผลการประมวลผลเพื่อสังเกตคำสั่ง ip route, ifconfig, arp -a, ps, tc qdisc show, tc filter show, tc class show, iptables -L และ ipset -L

2. WebEmu Plane เป็นเครื่องมือสำหรับการเชื่อมต่อระหว่าง WebEmu Interface และ VMs Plane เข้าด้วยกัน โดยมีหน่วยให้บริการ (Service) เชื่อมต่อทั้ง 2 ส่วนดังนี้

1) Emulator GUI เป็นหน้าต่างสำหรับผู้ใช้แสดงผลโดยใช้ RESTful ทำให้การเชื่อมต่อกับโหนดทางเครือข่ายภายในระบบ Emulator และจะไม่โหลดหน้าเว็บใหม่เมื่อถูกเรียกใช้โดยผู้ใช้งาน ซึ่งเป็นการแสดงผลที่ไม่แตกต่างจากการใช้โปรแกรมหน้าต่างเดี่ยวของโปรแกรมอิมูเลเตอร์ CORE

2) Monitoring Services เป็น Service รองรับคำสั่งจากผู้ใช้ในหน้าต่าง Emulator GUI แล้วเชื่อมต่อไปยังโหนดทางเครือข่าย ก่อนนำผลการประมวลผลที่ได้ส่งต่อให้กับ Emulator GUI

3) VM Configurations ทำหน้าที่จัดการโหนดทางเครือข่ายเมื่อมีการสั่งให้ระบบทำงาน โดยผู้ใช้สามารถเตรียมข้อมูลเบื้องต้นสำหรับแต่ละโหนด เช่น ประเภทอุปกรณ์ ค่าคอนฟิกเบื้องต้น และคำสั่งที่ต้องทำงานเมื่อ VMs เริ่มทำงาน

3. VMs Plane เป็นส่วนของโหนดทางเครือข่ายที่ใช้ Linux Container ทำหน้าที่เป็น Virtual Machine (VM) ในแต่ละ VM จะมีเซอร์วิส WebTerm รอรับการเข้าถึงหน้าต่าง Terminal ของระบบปฏิบัติการลินุกส์ผ่านเว็บบราวเซอร์ และมีหน่วยจัดการเครือข่ายแบบ Bridge ที่ถูกเชื่อมต่อกับการ์ดเครือข่าย Interface 1 ใน (Figure 1)

การแปลงรูปแบบการคอนฟิก

การคอนฟิกของอิมูเลเตอร์ CORE จะอยู่ในรูปแบบเฉพาะของ CORE งานวิจัยนี้จึงได้แปลงให้อยู่ในรูปแบบ JSON ซึ่งเป็นรูปแบบที่เป็นมาตรฐาน และสามารถนำไปใช้ได้กับหลายส่วน เช่น หน้าต่างของซอฟต์แวร์ Web Emulator จากงานวิจัยนี้ และเซอร์วิสที่เชื่อมต่อระหว่าง VM กับหน้าต่าง WebTerm เป็นต้น

การปรับเปลี่ยนการคอนฟิกแบบรูปแบบเฉพาะของ CORE มีดัง (Table 1) โดยได้เพิ่มรุ่นและคำอธิบายเป็นส่วนแรกของการคอนฟิก และมีการปรับเปลี่ยนค่าคอนฟิกจาก CORE เป็น Web Emulator ดังนี้

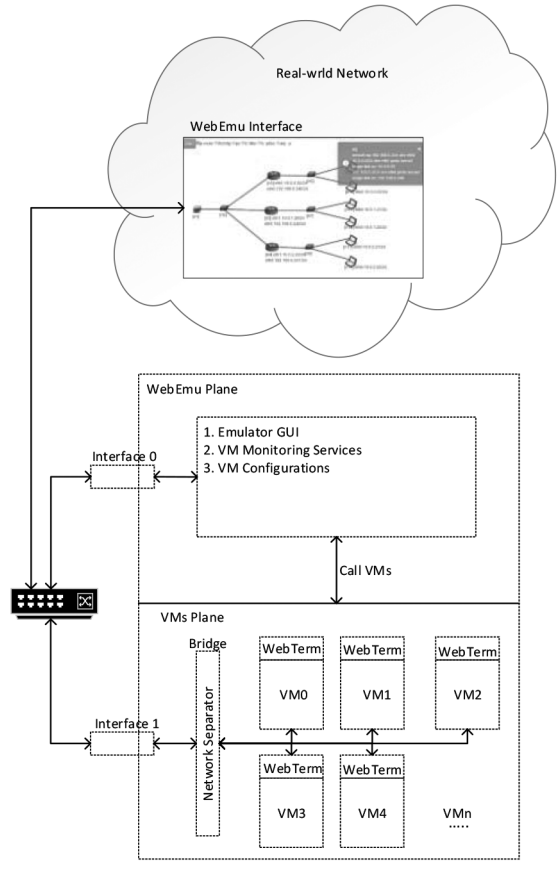

Figure 1 Web Emulator Structure

1. หมายเลขระบุโหนดทางเครือข่ายจาก node *<id>* ให้เป็นวัตถุอะเรย์ในรูปแบบของ JSON คือ “id”:*<id>* โดย*<id>* คือหมายเลขระบุโหนดทางเครือข่าย

2. ประเภทของอุปกรณ์เครือข่ายจาก type *<router/switch/RJ45>* เป็น “type”: *<router/switch/ RJ45>* โดยโหนดทางเครือข่ายจะมีคุณสมบัติ 3 ประเภท คือ Router Switch และ RJ45 ซึ่งระบุในตัวแปร model และtype เพื่อเรียก Icon แสดงผลในหน้าต่าง Web Emulator

3. ตำแหน่งของรูปอุปกรณ์และป้ายระบุข้อมูลจะนำมารวมเก็บไว้ใน *cords* และ *lbcoords* ตามลำดับ ซึ่งหมายถึง *iconcoords* และ *labelcoords* ในอิมูเลเตอร์ CORE

4. รายการ *infs* ของ Web Emulator จะเก็บข้อมูลของอุปกรณ์เครือข่าย เช่น ชื่อของการ์ดเครือข่าย หมายเลขไอพี และหมายเลขอุปกรณ์เครือข่าย ซึ่งมาจากตัวแปร *interface* และ *ip address* ของ CORE

การพัฒนาและทดลองโปรแกรมต้นแบบ

การพัฒนาโปรแกรมต้นแบบ

การพัฒนา Web Emulator จากแนวคิดของงานวิจัยนี้ ใช้หลายเครื่องมือมาใช้ร่วมกัน คือ

1. ส่วนของ VM จะใช้ CORE เป็นระบบพื้นฐาน แล้วแปลงให้เป็นไฟล์คอนฟิกในรูปแบบ JSON และส่งต่อให้กับผู้ใช้ผ่านทางโพรโทคอล HTTP สำหรับการคอนฟิกหน้าต่างผู้ใช้บนบราวเซอร์ โดยการแปลงคอนฟิกนี้จะใช้วิธีการอ่านไฟล์และให้บริการเว็บเซิฟเวอร์ด้วย Node.js ใช้ไลบรารี Express สำหรับให้บริการบนเว็บ และ fs สำหรับการอ่านและเขียนไฟล์ โดยลักษณะของการทำงานจะเป็นแบบ RESTFul ซึ่งตอบสนองกับผู้ใช้เหมือนกับการใช้โปรแกรมบนหน้าต่างเดสก์ท็อปทั่วไป

2. ส่วนการเชื่อมต่อระหว่างเซอร์วิสของ CORE กับหน้าต่างผู้ใช้ Web Emulator ทำได้โดยใช้ไลบรารี shelljs ของ Node.js ซึ่งเมื่อผู้ใช้เรียกคำสั่ง เช่น ifconfig ของ VM0 ระบบจะเรียกคำสั่ง vnode ของ CORE และส่งคำสั่ง ifconfig เข้าไปยังโหนดดังกล่าวแล้วรอการตอบกลับ ก่อนที่จะส่งต่อไปยังหน้าต่าง Web Emulator และเมื่อได้รับข้อความระบบจะแสดงเป็น popup ดังแสดงใน (Figure 3) ด้านบนขวา

3. ส่วนของ WebTerm พัฒนาโดยใช้ Node.js เป็นส่วนของให้บริการหน้าต่าง Terminal บนเว็บ ซึ่งใช้ไลบรารี Express เพื่อทำหน้าที่เป็นเซิฟเวอร์ แล้วใช้ไลบรารี term.js, pty.js และ socket.io ในการติดต่อกับระบบปฏิบัติการในแต่ละ VM และรอรับการเรียกใช้ ผ่านหน้าต่าง WebTerm แบบ Real-time

4. การจัดการด้านความมั่นคงของ Web Emulator ในงานวิจัยนี้จะมีหน้าต่างล็อกอินเพียงครั้งเดียวที่หน้าต่างหลัก ใน (Figure 3) ส่วนการเข้าถึง VM ต่าง ๆ จะเข้าใช้ได้โดยไม่ต้องทำการล็อกอิน

การทดลองและประเมินผล

(Figure 2) เป็นภาพการทดลองจากงานวิจัยนี้ แบ่งเครือข่ายออกเป็น 2 กลุ่ม คือ 1) เครือข่ายหลัก 192.168.6.0/24 ถูกเชื่อมต่อกับผู้ใช้ Web Emulator และ 2) กลุ่มของเครือข่ายทดลองที่ VM ถูกเชื่อมต่ออยู่ คือ 10.0.0.0/24 10.0.1.0/24 และ 10.0.2.0/24 ซึ่งเครือข่ายทั้ง 2 กลุ่มถูกเชื่อมกันด้วยเร้าเตอร์ R0 R1 และ R2 ตามลำดับ โดยการทดลองต้องมีการคอนฟิกค่าเบื้องต้นดังนี้

1. ที่เครื่องผู้ใช้จะใช้ระบบปฏิบัติการ Windows และต้องเพิ่มตารางเร้าแพ็กเก็ตแบบถาวร (static route) เพื่อให้เครือข่ายหลักมองเห็นเครือข่ายของ VM โดยใช้คำสั่ง route add เครือข่าย 10.0.0.0/24 ผ่านทางเร้าเตอร์ R0 ที่หมายเลขไอพี *192.168.6.241* ตามคำสั่งต่อไปนี้

#> route add 10.0.0.0 mask 255.255.255.0 192.168.6.241 metric 2

สำหรับการส่งแพ็กเก็ตไปยังเครือข่าย 10.0.1.0/24 ให้ส่งไปยังเร้าเตอร์ R1 ที่หมายเลขไอพี *192.168.6.242* โดยใช้คำสั่งต่อไปนี้

#> route add 10.0.1.0 mask 255.255.255.0 192.168.6.242 metric 2

การส่งแพ็กเก็ตไปยังเครือข่ายสุดท้าย 10.0.2.0/24 ให้ส่งไปยังเร้าเตอร์ R2 ที่หมายเลขไอพี *192.168.6.243* โดยใช้คำสั่งต่อไปนี้

#> route add 10.0.2.0 mask 255.255.255.0 192.168.6.243 metric 2

2. ผู้ใช้เข้าถึง R0, R1 และ R2 แล้วทำการคอนฟิกค่า default gateway โดยให้ชี้ไปที่เกตเวย์ของเครือข่าย 192.168.6.0/24 แล้วทำการรีโมทย์ไปยัง VM0 ถึง VMn และเพิ่มค่า default gateway ให้กับ VM0 ถึง VMn ก่อนที่จะทำการเชื่อมต่อไปยังเครือข่ายภายนอกมหาวิทยาลัย เช่น การเข้าเว็บไซต์ต่าง ๆ การทดลองเครือข่ายประเภทต่าง ๆ เป็นต้น

การทดสอบ Web Emulator ได้ประเมินแง่ของ Delay ของการตอบสนองต่อผู้ใช้ในหน้าต่างใช้งาน ทั้งส่วนหน้าต่าง Web Emulator และหน้าต่างของแต่ละ WebTerm สำหรับการสนับสนุนรูปแบบการสื่อสารได้ทดสอบการ

โพรโทคอลที่ทำงานในระดับชั้น DataLink, Internet Protocol (IP) และ Transport ซึ่งประกอบด้วย Address Resolution Protocol (ARP), IP Transmission Control Protocol (TCP), User Datagram Protocol (UDP) และการสื่อสารประเภทใหม่ Named Data Networking (NDN) และฟังก์ชันการเร้าติ้ง โดยการทดลองได้เพิ่ม VM ครั้งละ 10 โหนดจาก 1, 10, 20 และ 30 โหนด แล้วประเมินประสิทธิภาพของระบบ ซึ่งการทดลองได้ทำซ้ำจำนวน 30 ครั้ง ที่ระดับความเชื่อมั่นร้อยละ 95

เครื่องทดสอบในงานวิจัยนี้ ส่วนของเซิฟเวอร์ให้บริการหน้าต่าง Web Emulator และ VM ใช้หน่วยประมวลผล Intel(R) Core(TM)2 Duo CPU E6550 @2.33GHz หน่วยความจำหลัก 2GB และหน่วยความจำสำรอง 70GB ทำงานบนระบบปฏิบัติการลินุกส์ CentOS รุ่น 7 ส่วนผู้ทดลองใช้คอมพิวเตอร์เดสก์ท็อประบบปฏิบัติการ Windows หน่วยประมวลผล Intel(R) Core(TM) i5-3337U @1.80GHz หน่วยความจำหลัก 8GB และหน่วยความจำสำรอง 240GB เข้าถึงหน้าต่าง Web Emulator ด้วยเบราเซอร์ Chrome

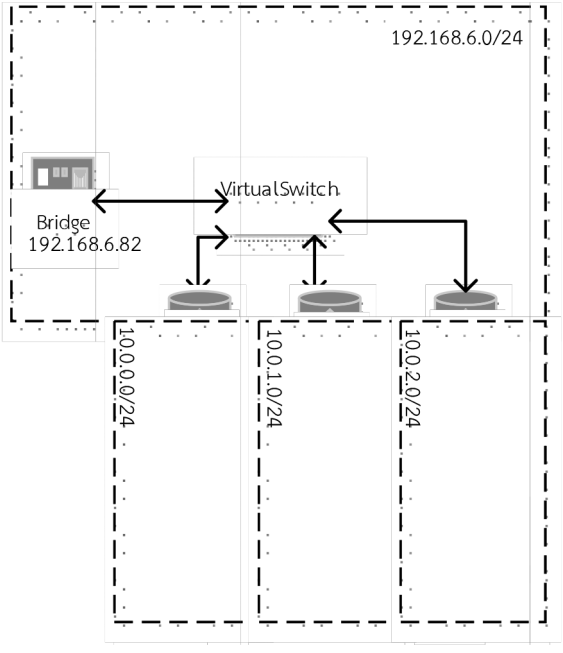

Figure 2 an evaluation scenario

ผลการทดลอง Web Emulator

ผลการพัฒนาโปรแกรมต้นแบบ

หน้าต่าง Web Emulator จากงานวิจัยนี้แสดงใน (Figure 3) ซึ่งประกอบด้วยส่วน Info คือข้อมูลของโปรแกรม Upload คือ การอัพโหลดการเชื่อมต่อเครือข่าย ส่วน Monitor คือ การแสดงสถานะเครือข่าย และส่วนตรวจสอบสถานะของ VM จากคำสั่งต่าง ๆ สำหรับส่วนหลักของโปรแกรมจะเป็นรูปแบบการเชื่อมต่อเครือข่าย ซึ่งผู้ใช้สามารถเลือกเพื่อเข้าคอนฟิกอุปกรณ์ เช่น เร้าเตอร์ และ คอมพิวเตอร์ได้

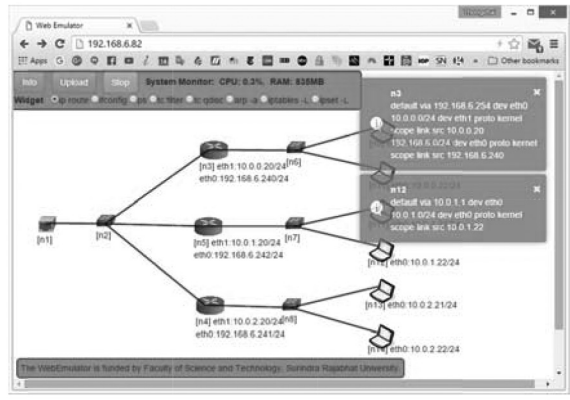

Figure 3 A main window of Web Emulator

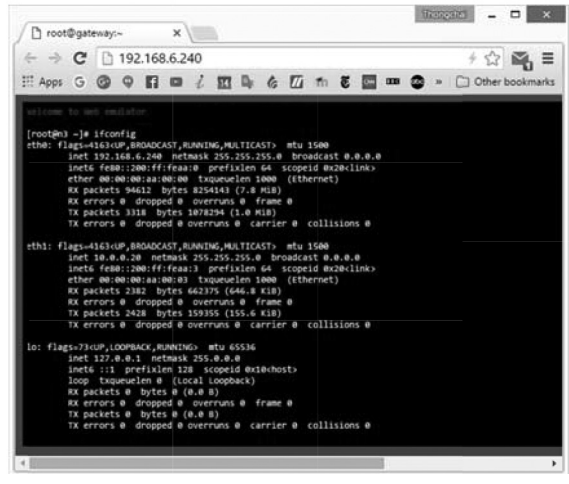

Figure 4 A WebTerm window

(Figure 4) เป็นหน้าต่าง WebTerm ในการเข้าถึงอุปกรณ์เร้าเตอร์ โดยผู้ใช้สามารถใช้คำสั่งได้เหมือนกับการรีโมทย์เข้าถึงผ่านโพรโทคอล Secured Shell (SSH) ซึ่งผู้ใช้สามารถใช้โปรแกรมที่สนับสนุนการใช้โพรโทคอล SSH ในการเข้าถึง VM ได้ โดยหน้าต่าง WebTerm สามารนำไปใช้เชื่อมต่อกับเครือข่ายอื่น ๆ โดยใช้แพ็กเก็ตประเภท TCP UDP ARP ฟังก์ชัน Routing และเครือข่าย NDN ได้

สำหรับความทัดเทียมของฟังก์ชันการทำงานระหว่าง CORE และ Web Emulator จากงานวิจัยนี้ ดัง (Table 2) จะเห็นว่า Web Emulator ยังมีฟังก์ชันบางส่วนที่ยังไม่สามารถเทียบเคียงกับ CORE ได้ เช่น การเพิ่มโหนดทางเครือข่ายบนหน้าต่าง การสร้างสคริปการเคลื่อนที่ของโหนด แต่ Web Emulator ช่วยให้ไม่เกิดปัญหาเรื่อง Platform ของระบบปฏิบัติการ Web Emulator จึงสามารถทำงานได้บนทุกระบบปฏิบัติการ มือถือ และแท็บเล็ตได้ ขณะที่ CORE จะทำงานได้เฉพาะบนระบบปฏิบัติการลินุกส์เท่านั้น

Table 2 functionality comparision of CORE and Web Emulator

	CORE	Web Emulator
Adding and configuring nodes to an software GUI	/	/
Mobility models	/	x
Monitoring features	/	/
Multi-platform software	x	/

ผลการทดสอบประสิทธิภาพ Web Emulator

การประเมินประสิทธิภาพการใช้หน้าต่างหลัก Web Emulator เทียบกับ CORE ดังแสดงใน (Figure 5) จะเห็นว่า Web Emulator มี Delay สูงกว่าการเข้าใช้ CORE อยู่ประมาณ 10ms โดยการใช้ Web Emulator อยู่ที่ 194.9±1.7 ms และการใช้ CORE จะอยู่ที่ 185.1 ±1.1 ms ดังนั้นการพัฒนาหน้าต่างใช้ CORE ถูกใช้งานบนเว็บจึงไม่เกิดความล่าช้ากว่าการใช้โปรแกรมบนหน้าต่างเดสก์ท็อปแต่อย่างใด

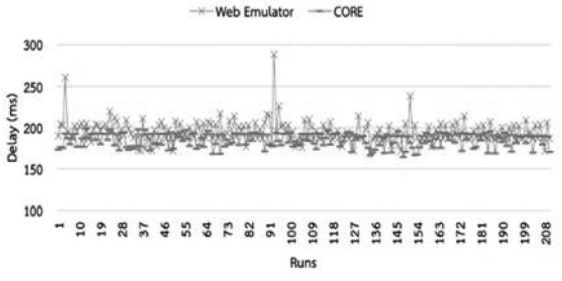

Figure 5 Processing delay between CORE and Web Emulator

เมื่อประเมินประสิทธิภาพระหว่าง CORE และ Web Emulator แยกตามคำสั่งที่ประกอบด้วย ps, tc iproute, ifconfig, arp, iptables และ ipset ดังแสดงใน (Figure 6) ซึ่งถูกเรียกใช้ผ่านหน้าต่างหลัก โดย CORE มีค่า Delay อยู่ที่ 185.94±2.81, 185.49±3.14, 185.89± 2.91, 186.02±2.46, 182.72±3.43 184.64±3.07 และ 185.31 ± 2.91 ms ตามลำดับ ส่วน Web Emulator จะมี Delay อยู่ที่ 200.60±5.11, 191.13±4.10, 197.37±3.64, 198.37±7.26, 189.17±3.58, 194.70±4.17 และ 193.03±3.11 ms ตามลำดับ ซึ่งจะเห็นว่าการเรียกใช้คำสั่งต่าง ๆ ใน VM ยังมีค่า Delay ต่ำกว่า 20 ms มีความแตกต่างกันเพียงเล็กน้อยจากการเรียกใช้จากหน้าต่าง Web Emulator เมื่อเทียบกับ CORE

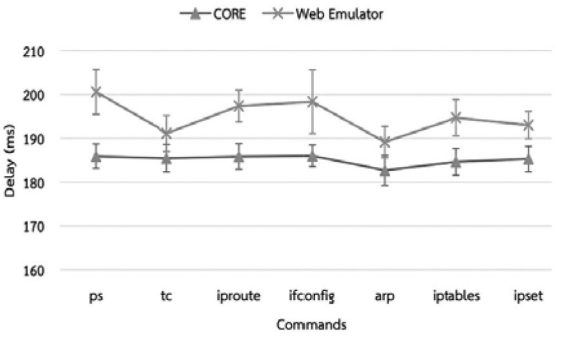

Figure 6 CORE and Web Emulator delay of the different unix commands

(Figure 7) แสดงอัตราการเพิ่มขึ้นของการ Delay เมื่อเพิ่มจำนวนผู้ใช้ Web Emulator จาก 1, 10, 20 และ 30 คน โดยค่า Delay อยู่ที่ 240.6±7.7, 294.5±13.0, 297.6±13.3 และ 328.1±15.7 ms ตามลำดับ ซึ่งจะเห็นว่าอัตราการเพิ่ม Delay เมื่อเพิ่มจำนวนผู้ใช้มีอัตรา Delay เพิ่มขึ้นเพียงเล็กน้อย

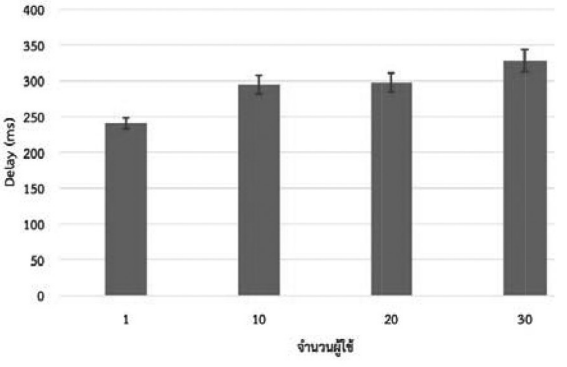

Figure 7 WebTerm delay

ผลการประเมินสมรรถนะเครื่องคอมพิวเตอร์ทดสอบ

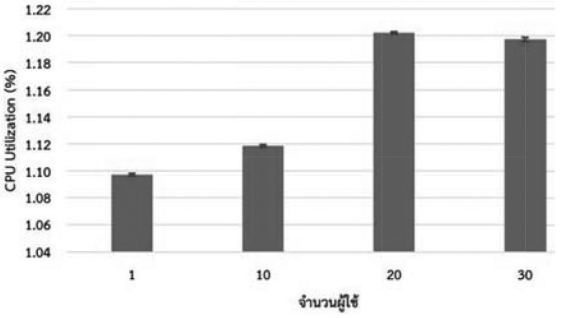

Figure 8 CPU utilizations

การประเมิน Utilization ดัง (Figure 8) พบว่าการใช้ซีพียูของเครื่องทดลองถูกใช้ไม่เกินร้อยละ 2 และจำนวนผู้ใช้ที่ทดสอบประกอบด้วย 1, 10, 20 และ 30 คน มีค่าเท่ากับร้อยละ 1.10±0.0009, 1.12±0.0008, 1.20±0.0007 และ 1.20±0.001 ms ตามลำดับ จะเห็นได้ว่าการเพิ่มจำนวนโหนด

ทางเครือข่ายมีอัตราการใช้ซีพียูเพิ่มขึ้นเพียงเล็กน้อย ไม่ส่งผลกระทบให้สมรรถนะของเครื่องทดลองลดลง

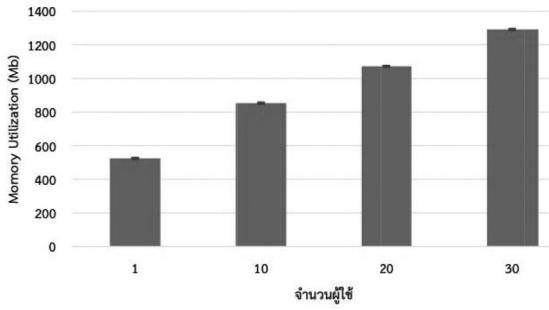

Figure 9 Memory utilizations

(Figure 9) แสดง Memory Untilization จากการทดลองเพิ่มจำนวนผู้ใช้จาก 1, 10, 20 และ 30 คน เพื่อสังเกตสมรรถนะของการจัดการ Memory ซึ่งจากการทดลองมีสัดส่วนการใช้ Memory อยู่ที่ 524.7±1.4, 853.2±1.7, 1072.7±0.8 และ 1292.2±1.5 ตามลำดับ ดังนั้นการเพิ่มจำนวน VM ส่งผลให้ Memory ถูกใช้เพิ่มขึ้นอย่างมีนัยยะ

สรุปและอภิปรายผล

การทดลองเครือข่ายเป็นเครื่องมือที่สำคัญที่ช่วยให้เกิดความเข้าใจในหลักการของเครือข่าย เครื่องมือที่ใช้ทดลองกลายเป็นปัจจัยสำคัญที่จะใช้สำหรับการเรียนรู้ เช่น สำหรับนักเรียน นักศึกษา สำหรับกลุ่มนักวิจัยเครือข่าย เป็นต้น ความเหมือนจริงและความแม่นยำจึงเป็นสิ่งสำคัญที่จะช่วยให้เกิดความเข้าใจต่อเหตุการณ์ต่าง ๆ ในระบบเครือข่ายได้ ก่อนหน้านี้การใช้ Simulation ถูกใช้อย่างกว้างขวาง เช่น OPNET NS2 และ NS3 และการใช้อุปกรณ์เครือข่ายจริงมีราคาสูงและยากต่อการจัดการ Emulation จึงมีบทบาทสำคัญที่จะตอบสนองทั้งการทดสอบแบบ test-bed และใช้เครือข่ายจริง แต่ก่อนหน้านี้ Emulation เป็นโปรแกรมที่ถูกใช้แบบเดี่ยว เช่น CORE Mininet และ อื่น ๆ การใช้ในลักษณะแบบกลุ่มทำได้ยาก งานวิจัยนี้จึงพัฒนาหน้าต่างสำหรับทดลองทางเครือข่ายที่ทำงานบนเว็บ ช่วยให้การศึกษาและทดลองด้านเครือข่ายทำได้เร็วขึ้น เหมาะกับการนำไปใช้เพื่อการเรียนการสอน และการทดลองสำหรับนักวิจัยเป็นกลุ่ม การพัฒนาระบบนี้ใช้ การต่อยอดจากโปรแกรม CORE ในการจัดการ VM และใช้ Node.js พัฒนาเป็นหน้าต่าง Web Emulator และทำงานแบบ RESTFul ซึ่งโปรแกรม Web Emulator ถูกพัฒนาเป็นโอเพ่นซอร์สและดาวน์โหลดได้จาก https://github.com/thongchaic/WebEmu.git โดยโปรแกรมสามารถตอบสนองต่อผู้ใช้แบบ Real-time จากการทดลองทั้งในแง่ของการตอบสนองต่อผู้ใช้ และสมรรถนะในการรองรับผู้ใช้จำนวนมาก โดย Web Emulator จากงานวิจัยนี้ มีค่า Delay อยู่ระหว่าง 194.9±1.7 ms แตกต่างจากการใช้ CORE ต่ำกว่า 20 ms ได้

ในอนาคตมีแนวคิดที่จะพัฒนาหน้าต่างที่มีคุณสมบัติเพิ่มขึ้นให้ทัดเทียมกับโปรแกรม CORE เช่น การสร้างเครือข่ายจากหน้าต่างโปรแกรม การกำหนดการเคลื่อนที่ของอุปกรณ์เครือข่ายได้ และการแสดงผลแบบสามมิติได้

กิตติกรรมประกาศ

งานวิจัยนี้ได้รับเงินสนับสนุนจาก คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏสุรินทร์

เอกสารอ้างอิง

1. Breslau L, Estrin D, Fall K, Floyd S, Heidemann J, Helmy A, et al. Advances in network simulation. Computer. 2000 May;33(5):59–67.
2. Millman E, Arora D, Neville S. STARS: A Framework for Statistically Rigorous Simulation-Based Network Research. In Biopolis, Spain; 2011. p. 733–739.
3. PlanetLab [Internet]. 2007 [cited 2016 Jan 10]. Available from: https://www.planet-lab.org/about
4. Koizumi M, Ebata T, Tsutsumi T, Ohshima K, Terada M. Design and Implementation of Scalable Distributed Wireless Network Emulator for High-Speed Mobility. In Bali; 2012. p. 302–307.
5. Duggirala V, Varadarajan S. Open Network Emulator: A Parallel Direct Code Execution Network Simulator. In Zhangjiajie; 2012. p. 101–110.
6. Cong J, Wang Z, Jia Y. WINPLEM: A Windows-Based Packet-Level Network Emulator. In Wuhan; 2010. p. 1–6.
7. Ahrenholz J. Comparison of CORE Network Emulation Platforms. In San Jose, California, USA; 2010. p. 166–171.
8. Mininet Team. Mininet: An Instant Virtual Network on your Laptop [Internet]. 2016 [cited 2016 Jan 12]. Available from: http://mininet.org/
9. Cloonix team. Cloonix [Internet]. 2016 [cited 2016 Jan 20]. Available from: http://www.cloonix.fr/
10. Battista G, Patrignani M, Pizzonia M, Rimondini M. Netkit [Internet]. 2011 [cited 2016 Jan 12]. Available from: http://wiki.netkit.org/index.php/Main_Page

11. Cetušić G, Mikuc M, Salopek D, Vasić V, Zec M. Integrated Multiprotocol Network Emulator/Simulator [Internet]. 2015 [cited 2016 Jan 10]. Available from: http://imunes.net/
12. LXC team. Linux Containers [Internet]. 2014 [cited 2016 Jan 7]. Available from: https://linuxcontainers.org
13. KVM team. Kernel Virtual Machine [Internet]. 2015 [cited 2016 Jan 7]. Available from: http://www.linux-kvm.org/page/Main_Page
14. Zhang L, Afanasyev A, Burke J, Jacobson V, claffy kc, Crowley P, et al. Named Data Networking. ACM SIGCOMM Computer Communication Review. 2014 Jul;44(3):66–73.
15. Linux Foundation. Node.js [Internet]. 2015 [cited 2015 Jul 15]. Available from: https://nodejs.org/
16. Richardson L, Ruby S. RESTful Web Services. O'Reilly Media; 2007.
17. The Network Simulator - ns-2 [Internet]. [cited 2014 May 17]. Available from: http://www.isi.edu/nsnam/ns/
18. Network Simulator 3 [Internet]. [cited 2014 May 17]. Available from: http://www.nsnam.org
19. Cisco System. Packet Tracer [Internet]. 2015 [cited 2016 Jan 7]. Available from: https://www.netacad.com/about-networking-academy/packet-tracer/
20. Open-Access Research Testbed for Next-Generation Wireless Networks [Internet]. 2014 [cited 2016 Jan 10]. Available from: http://www.orbit-lab.org/
21. Haleplidis E, Pentikousis K, Denazis S, Salim J, Meyer D. Software-Defined Networking (SDN): Layers and Architecture Terminology [Internet]. 2015 Jan. Report No.: 7426. Available from: https://tools.ietf.org/html/rfc7426