รายงานความก้าวหน้าวิชา CE Project

ครั้งที่ 2

ระหว่างวันที่ 26 ส.ค. 65 ถึงวันที่ 9 ก.ย 65

1. ชื่อโครงงาน (อังกฤษ) Performance Improvement Mechanism in Software-defined Network

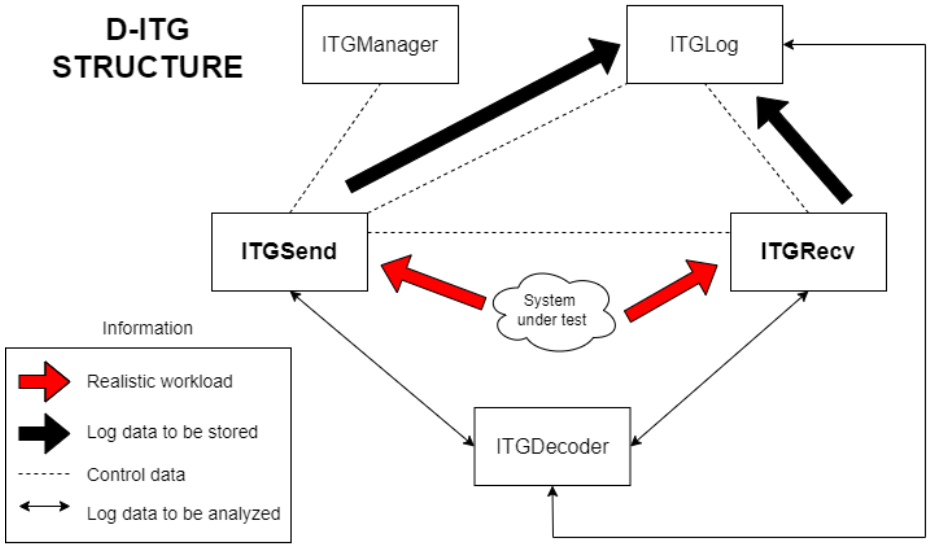
2. การดำเนินงานมีความก้าวหน้า 30 % (ใช้ค่า **% Complete** จาก MS Project)

   มีความก้าวหน้าเพิ่มขึ้นจากรายงานความก้าวหน้า ครั้งก่อน 5 %

   ☐ เร็วกว่าแผน ..... วัน ■ ช้ากว่าแผน 5 วัน

3. รายละเอียดความก้าวหน้า

   3.1 รายละเอียดเพิ่มเติมของโปรแกรม D-ITG Traffic Generator



รูปที่ 1.1 แสดงโครงสร้างของ D-ITG Traffic Generator

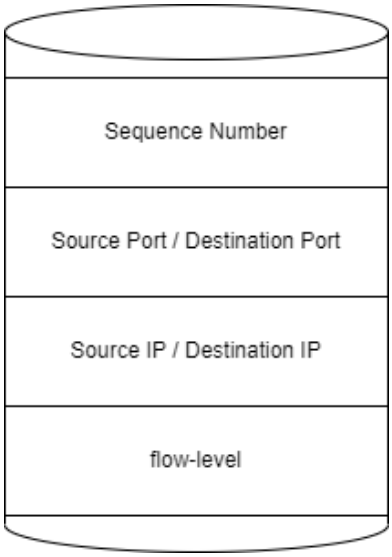
จากรูปที่ 1.1 เราจะนำ D-ITG ทำการจำลองสร้างข้อมูลขึ้นมาและส่งภายในเครือข่ายที่เรากำหนดขึ้น โดยเราจะให้ ITGSend และ ITGRecv เป็นโหนดทั้งคู่เปรียบเสมือนอุปกรณ์ภายในเครือข่าย (Switch) โดย ITGSend จะเป็นฝั่งการเริ่มต้นการส่งข้อมูล (Source) และ ITGRecv จะเป็นฝั่งคอยรับข้อมูล (Destination) ทั้งคู่จะส่งแพ็คเก็ตหากันโดยตรง ในแต่ละโหนดของทั้งคู่นั้น ทั้งคู่จะเก็บข้อมูลในการส่งไว้ที่ใน ITGLog ซึ่งจะเก็บอยู่ในรูปแบบของ log file และเราสามารถที่จะเรียกดูการไหลของข้อมูลได้ ก่อนที่จะเรียกดู log file ในด้านของ ITGDecoder จะทำการวิเคราะห์ข้อมูลในการส่งและประมวลผล

ให้กับ ITGLog ไม่ว่าจะเป็น จำนวนการส่งแพ็คเก็ต อัตราการส่งข้อมูล เป็นต้น ซึ่งข้อมูลเหล่านี้สามารถนำไปแปลงเป็น csv ได้ เพื่อที่จะนำข้อมูลไปเทรนในขั้นตอนต่อไปในการทำโครงงาน

ภายใน D-IGT Traffic Generator ที่เราได้ใช้เพื่อทำการจำลองการส่งข้อมูล (Packet) นั้น จะมีโพรโทคอลให้ใช้งานอยู่ 5 รูปแบบ ได้แก่ UDP (User Datagram Protocol), TCP (Transport Control Protocol), ICMP (Internet Control Messaging Protocol), SCTP (Session Control Transport Protocol) และ DCCP (Datagram Congestion Control Protocol) โดยโพรโทคอลทั้งหมดนี้ จะสามารถใช้งานได้เพียงในระบบปฏิบัติการลีนุกซ์ (Linux OS) เท่านั้น ซึ่งก็เป็นสาเหตุที่เลือกที่จะใช้ Ubuntu OS ที่ใช้งานผ่านตัวเครื่องจักรเสมือน (Virtual Machine)

โพรโทคอล (Protocol)	ระบบปฏิบัติการที่รองรับ (Supported OS)
UDP	Linux OS, Windows OS
TCP	Linux OS, Windows OS
ICMP	Linux OS (root), Windows OS
SCTP	Linux OS
DCCP	Linux OS

ตารางที่ 1.1 แสดงการเปรียบเทียบระบบปฏิบัติการที่รองรับโพรโทคอลต่างๆ

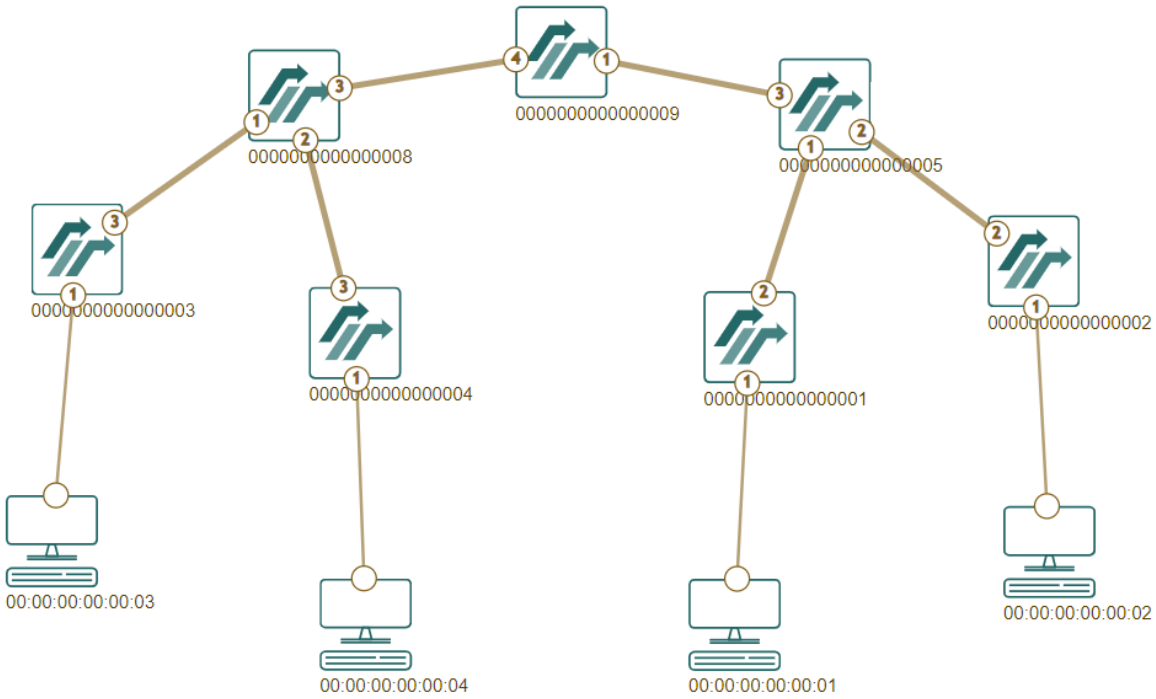


รูปที่ 1.2 แสดง Header fields ที่เราสามารถปรับเปลี่ยนได้ภายในแต่ละโหนด

ภายในโปรแกรม D-IGT Traffic Generator เราสามารถทำให้ ITGSend ส่งข้อมูลไปหา ITGRecv ได้หลายทิศทาง โดยเราสามารถปรับเปลี่ยน Source / Destination IP, Source / Destination Port และ Sequence number ได้ ซึ่งการปรับเปลี่ยนข้อมูลเหล่านี้จะทำให้เราสามารถกำหนดเส้นทางในการส่งข้อมูลได้

3.2 การใช้โปรแกรม D-ITG ร่วมกับ Mininet

D-ITG จะทำหน้าที่จำลองการส่งข้อมูลภายในเครือข่าย ส่วนด้าน Mininet จะรับหน้าที่ในการสร้างเครือข่ายขึ้นมา หรือ โทโพโลยี (Topologies) ในการส่งข้อมูลจากต้นทางไปยังปลายทางนั้น เราจำเป็นที่จะต้องกำหนด IP Address ให้กับโหนดแต่ละโหนดภายในเครือข่าย เพื่อที่จะให้ข้อมูลเดินทางได้อย่างถูกต้องและรู้จุดหมายปลายทาง นอกจากนี้เราสามารถเรียก Flowmanager เข้ามาช่วยแสดงแผนภาพเครือข่ายผ่านเว็บบราวเซอร์ได้ เช่นเดียวกับที่เราได้ทำการทดลองในรายวิชา CEPP ที่ผ่านมา



รูปที่ 1.3 แสดงแผนภาพเครือข่ายที่ได้ทำการจำลองผ่าน Mininet และแสดงผ่าน Flowmanager Browser

โดยภายในเครือข่ายที่จำลองขึ้นประกอบไปด้วยสวิตช์ 10 เครื่อง และ คอมพิวเตอร์ 4 เครื่อง โดย IP Address v4 ที่ใช้ภายในเครือข่ายก็จะสามารถถูกกำหนดผ่าน Python-Script file ได้ โดยอ่านข้อมูลผ่าน Mininet อีกทีหนึ่ง

```python
from mininet.topo import Topo
from mininet.node import Host, OVSKernelSwitch
from mininet.link import TCLink

class Fat( Topo ):

    def build( self ):
        "Create custom topo."

        # Add hosts and switches
        h1 = self.addHost( 'h1', mac="00:00:00:00:00:01",ip="192.168.69.1/24", cls=Host,)
        h2 = self.addHost( 'h2', mac="00:00:00:00:00:02",ip="192.168.69.2/24", cls=Host,)
        h3 = self.addHost( 'h3', mac="00:00:00:00:00:03",ip="192.168.69.3/24", cls=Host,)
        h4 = self.addHost( 'h4', mac="00:00:00:00:00:04",ip="192.168.69.4/24", cls=Host,)

        sw1 = self.addSwitch( 'sw1', cls=OVSKernelSwitch, protocols='OpenFlow13', )
        sw2 = self.addSwitch( 'sw2', cls=OVSKernelSwitch, protocols='OpenFlow13', )
        sw3 = self.addSwitch( 'sw3', cls=OVSKernelSwitch, protocols='OpenFlow13', )
        sw4 = self.addSwitch( 'sw4', cls=OVSKernelSwitch, protocols='OpenFlow13', )
        sw5 = self.addSwitch( 'sw5', cls=OVSKernelSwitch, protocols='OpenFlow13',)
        sw8 = self.addSwitch( 'sw8', cls=OVSKernelSwitch, protocols='OpenFlow13', )
        sw9 = self.addSwitch( 'sw9', cls=OVSKernelSwitch, protocols='OpenFlow13', )

        # Add links
        self.addLink( h1, sw1, 1, 1, cls=TCLink, bw=100, )
        self.addLink( h2, sw2, 1, 1, cls=TCLink, bw=100, )
        self.addLink( h3, sw3, 1, 1, cls=TCLink, bw=100, )
        self.addLink( h4, sw4, 1, 1, cls=TCLink, bw=100,  )
        self.addLink( sw1, sw5, 2, 1, cls=TCLink, bw=100,  )
        self.addLink( sw2, sw5, 2, 2, cls=TCLink, bw=100,  )
        self.addLink( sw3, sw8, 3, 1, cls=TCLink, bw=100,  )
        self.addLink( sw4, sw8, 3, 2, cls=TCLink, bw=100,  )
        self.addLink( sw9, sw5, 1, 3, cls=TCLink, bw=100,  )
        self.addLink( sw9, sw8, 4, 3, cls=TCLink, bw=100,  )

topos = { 'mytopo': ( lambda: Fat() ) }
```

รูปที่ 1.4 แสดง Python-Script file ที่ทำการกำหนดองค์ประกอบภายในเครือข่ายจำลอง

ในการอัพเดทความคืบหน้าครั้งที่ (ครั้งที่ 2) พวกเราได้ทำการประกอบ D-ITG และ Mininet เข้าด้วยกันได้แล้ว แต่ปัญหาที่เกิดขึ้นก็คือ ทำการทดลองจากอุปกรณ์หนึ่งไปยังอีกอุปกรณ์หนึ่งไม่ได้ ซึ่งไม่ตรงกับทางทฤษฎีเพียงอย่างใด จึงทำให้เรายังไม่ได้เริ่มการเทรนข้อมูลที่ได้จากการส่งข้อมูลภายในเครือข่าย ปัญหาต่อมาคือ Flowmanager ซึ่งมีส่วนประกอบของ “Eventlet” ก็คือไลบรารี่สำหรับปรับปรุงองค์ประกอบเครือข่ายต่างๆ โดยใช้ภาษาไพท่อน (Python) แต่สามารถแก้ไขได้แล้ว

4. ปัญหาที่เกิดขึ้นและแนวทางการแก้ไข

ปัญหาที่ 1: Eventlet Networking Library

สถานะ: แก้ไขได้แล้ว

สาเหตุ: เนื่องจาก Flowmanager ได้ทำการอัพเดทแพทช์ใหม่และใช้ เวอร์ชัน 0.31.4 เป็นเวอร์ชันล่าสุด ซึ่งทำให้เกิดบัคการเรียกไฟล์เกิดขึ้น พวกเราได้ทำการตรวจเวอร์ชันจนกระทั่งพบว่าหลังจากเวอร์ชัน 0.30.2 เป็นต้นไปไม่สามารถใช้งานได้ จึงทำให้พวกเราตัดสินใจที่จะใช้ Eventlet version 0.30.2

ปัญหาที่ 2: ไม่สามารถทำการปิง (Ping) และส่งข้อมูลจากอุปกรณ์ไปอีกอุปกรณ์ได้ภายในเครือข่าย

สถานะ: กำลังดำเนินการ

แนวทางในการแก้ไข: ทำการทดลองและศึกษาไปด้วยว่าเหตุผลเกิดจากอะไร และทำการแก้ไข

5. สิ่งที่จะดำเนินการต่อไป

ทำการทดลองต่างๆ ภายในเครือข่ายเช่น การกำหนด IPv4 Address / Mac Address การกำหนดเส้นทาง ฯลฯ เมื่อทำการทดสอบการปิงผ่านแล้ว หรือ ส่งข้อมูลผ่านแล้ว พวกเราก็จะเริ่มทำการเพิ่มอุปสรรค (Noise) ให้กับเครือข่าย เช่น ดีเลย์, จิตเตอร์ และ การกำหนดขนาดข้อมูล เป็นต้น เพื่อให้เป็นการส่งข้อมูลอย่างสมจริงมากยิ่งขึ้น หลังจากนั้นเราก็จะนำข้อมูลที่เราได้จากการทดลอง (Log file) นำไปเทรนต่อ โดยใช้โมเดล Bidirectional LSTM และ โมเดล LSTM ในการเทรนข้อมูล