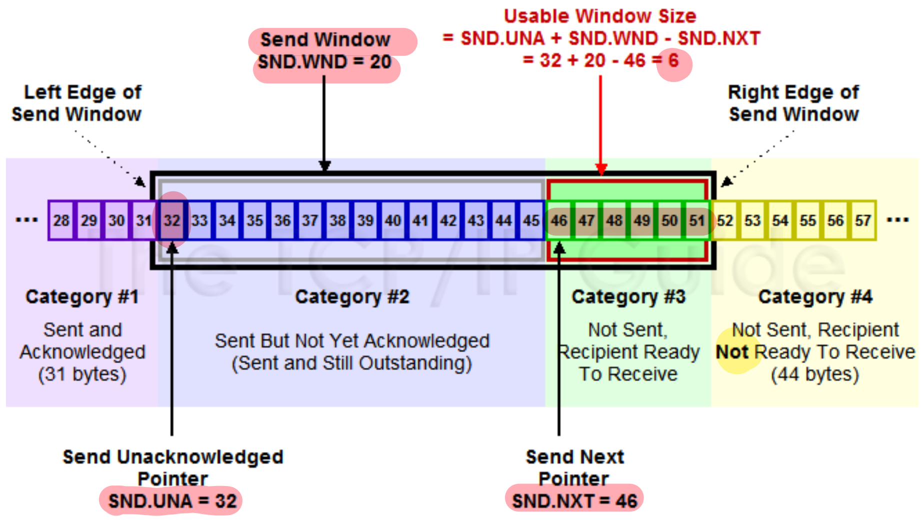
นายปัณณวิชญ์ ธนศักดิ์ดาภา 63010584
01076010 เครือข่ายคอมพิวเตอร์ : 2/2564
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

กิจกรรมที่ 8 : TCP Window

กิจกรรมครั้งนี้จะเป็นการทำความเข้าใจกับโปรโตคอล TCP (Transmission Control Protocol) ให้มากยิ่งขึ้น โดยเน้นเรื่องของ TCP Window โดย TCP Window จะแบ่งออกเป็น send Window และ receive Window

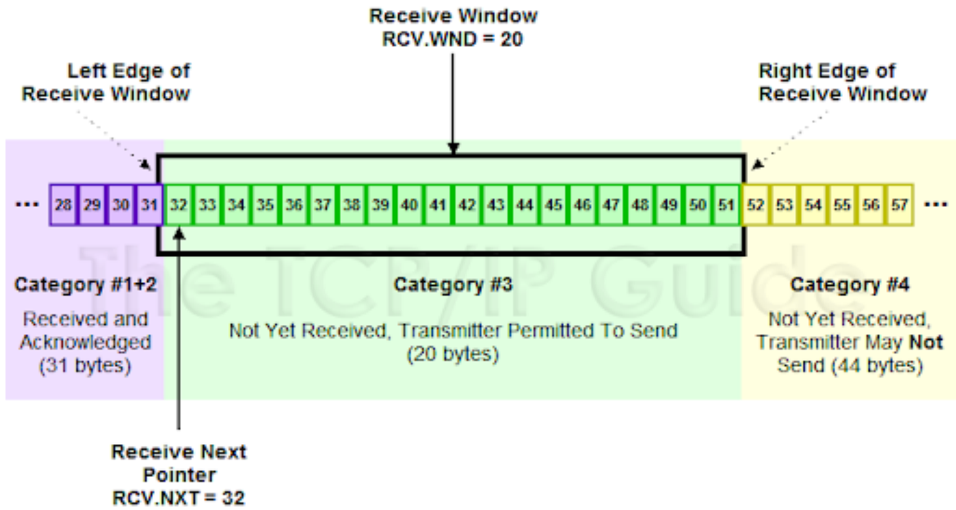
ใน send window จะแบ่งออกเป็น 4 ส่วน คือ

- ข้อมูลที่ส่งแล้วและได้รับ Acknowledge ไปแล้ว
- ข้อมูลที่ส่งไปแล้วแต่ยังไม่ได้รับ Acknowledge (ใน Wireshark จะเรียกว่า byte in flight)
- ข้อมูลที่ยังไม่ได้ส่ง และ ฝั่งรับสามารถรับได้ (ตามขนาดของ receive window)
- ข้อมูลที่ยังไม่ได้ส่ง และ ฝั่งรับไม่พร้อมจะรับเนื่องจากขนาดของ receive window

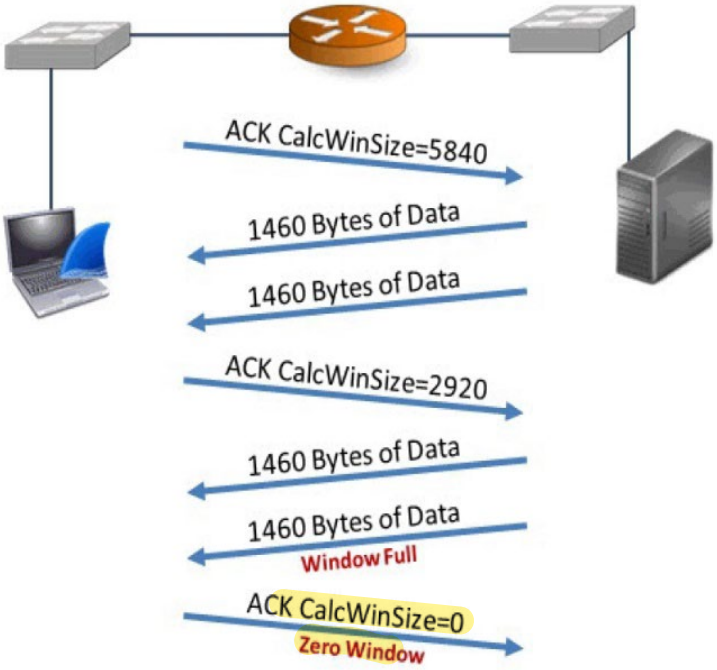


ใน receive window จะแบ่งเป็น 2 ส่วน

- ข้อมูลที่รับแล้วและ Acknowledge ไปแล้ว
- ข้อมูลพร้อมจะรับ



ในระหว่างการสื่อสารทั้ง 2 ด้านจะมีการแจ้งขนาดของ window size ที่เหลือที่ยังรับข้อมูลได้มาใน header ของ TCP โดยมีขนาด 2 ไบต์ โดยมีค่าสูงสุด คือ 65,535 ไบต์ โดยมี Scaling Factor เป็นตัวคูณ ซึ่งหากฝั่งรับไม่สามารถนำข้อมูลออกจาก receive window ได้เร็วพอจะทำให้ Buffer เต็มและเกิด zero window ตามรูป (หมายเหตุข้อมูล window full และ zero window นี้เป็นข้อมูลที่ wireshark สร้างขึ้น เพื่อให้สะดวกต่อการใช้งาน)



1. ให้เปิดไฟล์ tr-youtubebad.pcapng จากนั้นให้ค้นหาเหตุการณ์ zero window โดยใช้ display filter tcp.analysis.zero_window จะเห็นว่ามี zero window เกิดขึ้นจำนวนมาก ให้เลือกบรรทัดแรก แล้วยกเลิก filter โปรแกรม wireshark จะแสดงบริเวณ packet ที่เกิด zero window ครั้งแรก ให้ขยาย TCP หาฟิลด์ calculated window size แล้วสร้างเป็นคอลัมน์ โดยกำหนดให้ Align Center และตั้งชื่อเป็น **WinSize**
   - ให้สังเกตที่ packet 2718 ซึ่งเป็น packet ที่ host 24.4.7.217 ส่ง ACK กลับมา โดยมี window size เหลือเพียง 1,460 ไบต์
   - ต่อมาใน packet 2719 host 208.117.232.102 มีการส่งข้อมูลไปอีก 1,460 ไบต์ ซึ่งจะทำให้เต็ม receive window พอดี และทำให้ wireshark สร้างข้อมูลแจ้งเตือนว่า window full
   - เมื่อถึง Packet 2720 host 24.4.7.217 ก็ส่ง Packet ACK กลับมา โดยมีค่า window size เป็น 0 ทำให้ wireshark สร้างข้อมูลแจ้งเตือนว่า zero window
   - ให้สังเกตช่วงเวลาระหว่าง packet 2720 และ 2721 จะเห็นว่ามีระยะห่างมากกว่าปกติ หมายความว่าฝั่งผู้ส่งเมื่อพบ zero window ก็จะรอฝั่งผู้รับให้เคลียร์ receive window เสียก่อน
   - ใน packet 2721 จะมีการส่ง packet keep alive (คือ packet ACK ที่ไม่มีข้อมูล จากฝั่งผู้ส่ง ซึ่งจะเกิดขึ้นเมื่อ keepalive timer expire)
   - จากนั้นใน packet 2722 ผู้รับจะส่ง ACK กลับมา โดยมี window size เป็น 0 เช่นเดิม และเกิดซ้ำอีกครั้งใน packet 2723 และ 2724

- จนกระทั่ง packet 2725 ฝั่งผู้รับจึงส่ง packet ACK ซึ่งมีขนาดของ window size = 243820 ซึ่งไม่เท่ากับ 0 ซึ่งหมายความว่า receive window ของฝั่งผู้รับว่างแล้ว พร้อมรับข้อมูลใหม่ ณ จุดนี้ ถือว่าเหตุการณ์ zero window สิ้นสุดลง โดย wireshark จะสร้างข้อมูลแจ้งเตือน window update

packet ที่ 4022

2. ให้นักศึกษาตรวจสอบ zero window ระยะที่ 2 แล้วตอบคำถาม ต่อไปนี้

- เกิด window full, zero window (เฉพาะครั้งแรก) และ window update ที่ packet ใด

window full ; packet ที่ 4022 window update ; packet ที่ 4036

zero window ; packet ที่ 4023

- หลังจากมีการทำ keep alive กี่ครั้ง มีช่วงระยะเวลาเท่าไรบ้าง นับจาก zero window ครั้งก่อน ให้แสดงรูป capture จาก wireshark ที่แสดงเวลาของ keep alive แต่ละครั้ง มาด้วยใน 1 รูป

มี keep alive 6 ครั้ง

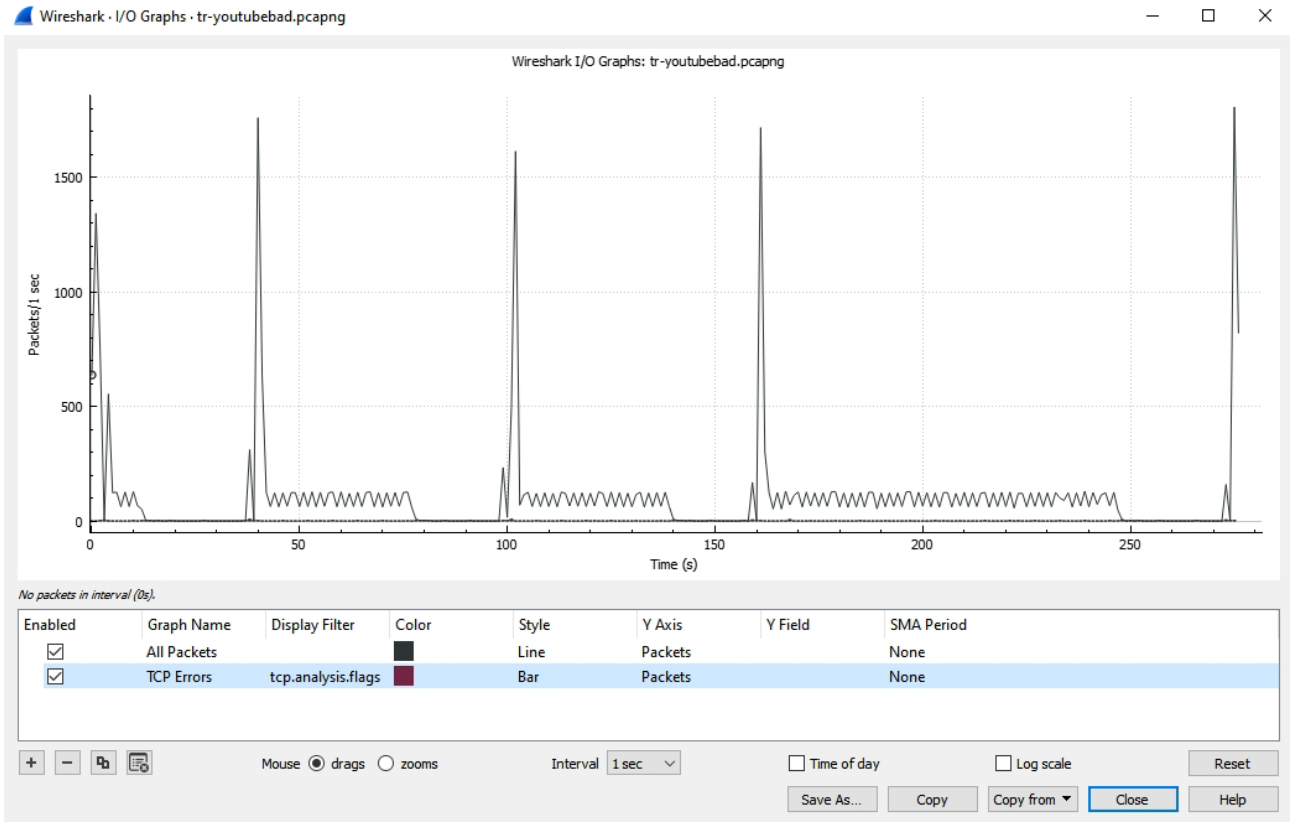
เวลา 0.477 , 0.995 , 1.878 , 3.704 , 7.398 , 10.02 ตามลำดับ.

4022	12.6792…	208.117.232.102	24.4.7.217	TCP	382	4248122	4248450	1270	8384	0.362283000	[TCP Window Full] 80 → 56770 [PSH, A
4023	12.8890…	24.4.7.217	208.117.232.102	TCP	54	1270	1270	4248450	0	0.209752000	[TCP ZeroWindow] 56770 → 80 [ACK] Se
4024	13.3666…	208.117.232.102	24.4.7.217	TCP	60	4248449	4248449	1270	8384	0.477622000	[TCP Keep-Alive] 80 → 56770 [ACK] Se
4025	13.3666…	24.4.7.217	208.117.232.102	TCP	54	1270	1270	4248450	0	0.000046000	[TCP ZeroWindow] 56770 → 80 [ACK] Se
4026	14.3620…	208.117.232.102	24.4.7.217	TCP	60	4248449	4248449	1270	8384	0.995377000	[TCP Keep-Alive] 80 → 56770 [ACK] Se
4027	14.3621…	24.4.7.217	208.117.232.102	TCP	54	1270	1270	4248450	0	0.000057000	[TCP ZeroWindow] 56770 → 80 [ACK] Se
4028	16.2402…	208.117.232.102	24.4.7.217	TCP	60	4248449	4248449	1270	8384	1.878101000	[TCP Keep-Alive] 80 → 56770 [ACK] Se
4029	16.2402…	24.4.7.217	208.117.232.102	TCP	54	1270	1270	4248450	0	0.000063000	[TCP ZeroWindow] 56770 → 80 [ACK] Se
4030	19.9451…	208.117.232.102	24.4.7.217	TCP	60	4248449	4248449	1270	8384	3.704824000	[TCP Keep-Alive] 80 → 56770 [ACK] Se
4031	19.9452…	24.4.7.217	208.117.232.102	TCP	54	1270	1270	4248450	0	0.000141000	[TCP ZeroWindow] 56770 → 80 [ACK] Se
4032	27.3441…	208.117.232.102	24.4.7.217	TCP	60	4248449	4248449	1270	8384	7.398856000	[TCP Keep-Alive] 80 → 56770 [ACK] Se
4033	27.3442…	24.4.7.217	208.117.232.102	TCP	54	1270	1270	4248450	0	0.000100000	[TCP ZeroWindow] 56770 → 80 [ACK] Se
4034	37.3642…	208.117.232.102	24.4.7.217	TCP	60	4248449	4248449	1270	8384	10.020053000	[TCP Keep-Alive] 80 → 56770 [ACK] Se
4035	37.3643…	24.4.7.217	208.117.232.102	TCP	54	1270	1270	4248450	0	0.000052000	[TCP ZeroWindow] 56770 → 80 [ACK] Se
4036	38.3192…	24.4.7.217	208.117.232.102	TCP	54	1270	1270	4248450	166440	0.954932000	[TCP Window Update] 56770 → 80 [ACK]

- ระยะเวลาตั้งแต่เกิด zero window ครั้งแรกจนถึง window update ใช้เวลาเท่าไร

ประมาณ 25.4302 sec

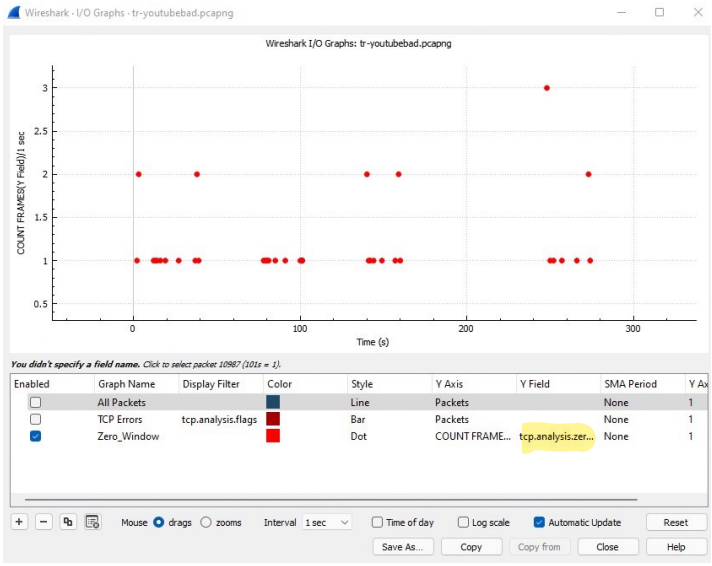
3. การวิเคราะห์ข้อมูลนอกจากจะทำในหน้าต่าง Packet List และ Packet Detail แล้ว ใน wireshark ยังให้เครื่องมือประเภทกราฟมาด้วย จากไฟล์เดิม ให้นักศึกษาเรียกเมนู Statistics | I/O Graph จะปรากฏหน้าจอดังนี้



- ข้อมูลแกน Y คือ packet/sec แกน x คือเวลา ซึ่งจะเห็นว่าข้อมูลมีการส่งได้ดี (กราฟพุ่งสูง จำนวน 5 ครั้ง) จากนั้นก็ลดลงอย่างมาก
- ในช่องด้านล่าง เราสามารถสร้างกราฟขึ้นมาใหม่ได้ ให้กด + แล้วกำหนดข้อมูลดังนี้
  - Graph Name : Zero_Window
  - Display filter : ว่าง
  - Color : แดง
  - Style : Dot
  - Y Axis : COUNT FRAMES(Y Field)
  - Y Field : tcp.analysis.zero_window
- ให้ Disable กราฟเดิมทั้ง 2 กราฟ

- กราฟบอกข้อมูลอะไร (แสดงรูป capture ของกราฟด้วย)

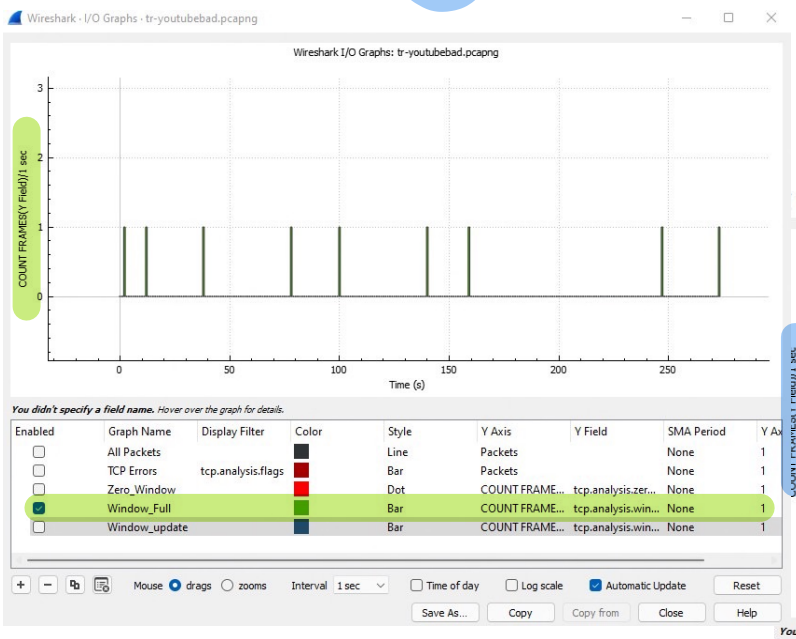
ขณะที่ส่งมีการเกิด zero-window ในรูปแบบ Dot ตามช่วงเวลาต่างๆ

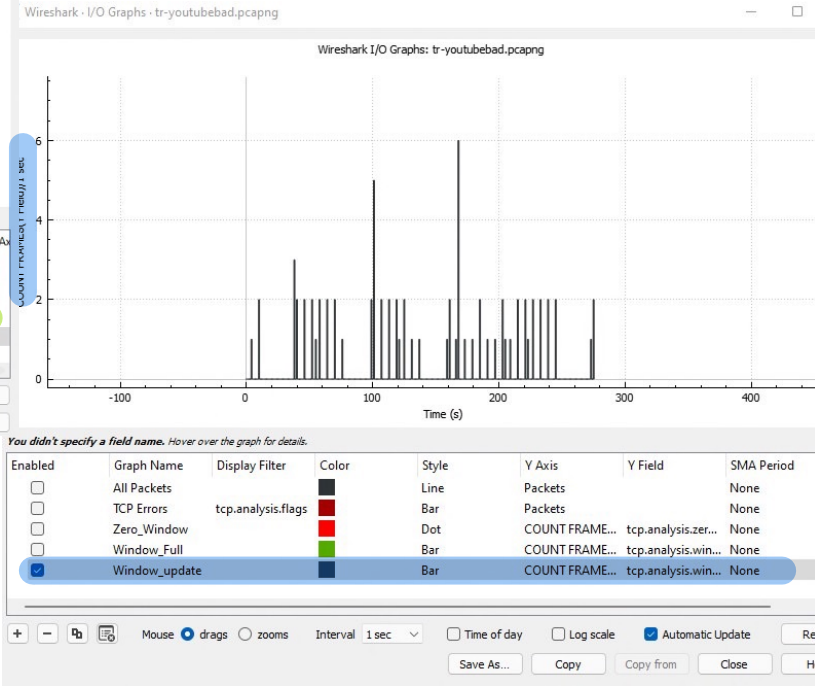


4. ให้สร้างกราฟเพิ่มอีก 2 กราฟ ดังนี้
   - ชื่อ Window_Full โดยใน Y(AXIS) ใช้ COUNT FRAMES(Y Field) และช่อง Y Field ใช้ tcp.analysis.window_full กำหนดประเภทเป็น Bar สีเขียว
   - ชื่อ Window_Update โดยใน Y(AXIS) ใช้ COUNT FRAMES(*) และช่อง Y Field ใช้ tcp.analysis.window_update กำหนดประเภทเป็น Bar สีน้ำเงิน
   - กราฟแสดงอะไร (แสดงรูป capture ของกราฟด้วย)

กราฟสีเขียวแสดงจำนวนครั้ง window full ต่อ 1 sec

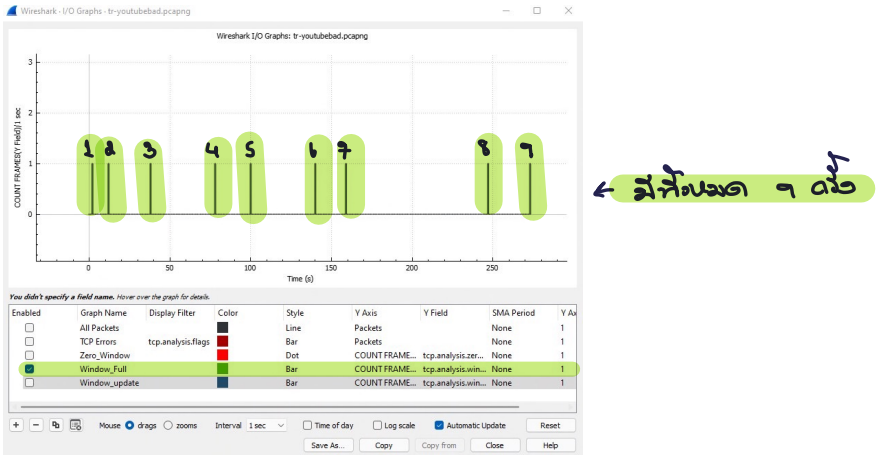
กราฟสีน้ำเงินแสดงจำนวนครั้ง window-update ต่อ 1 sec





- จากกราฟสามารถบอกได้หรือไม่ว่ามี window full กี่ครั้ง ให้ Capture รูปประกอบด้วย

สังเกตได้ว่ามี window full ทั้งหมด 7 ครั้ง

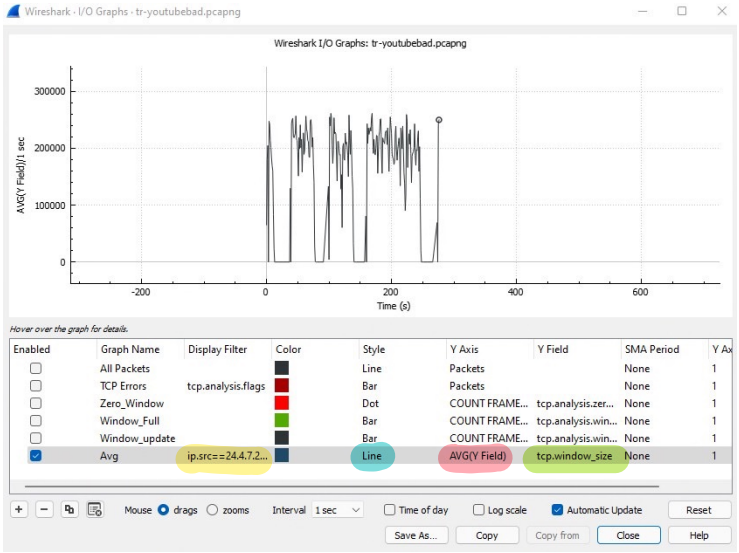


5. ให้สร้าง I/O Graph ใหม่ โดยในช่อง Display Filter ให้ใส่ ip.src==24.4.7.217 ใน Y(AXIS) ใช้ AVG(*) และช่อง Y Field ใช้ tcp.window_size กำหนดประเภทเป็น Line ให้ capture รูป และ อธิบายว่าเราสามารถวิเคราะห์ข้อมูลอะไรจากกราฟนี้ ให้ Capture รูปประกอบด้วย

วิเคราะห์ขนาดของ window size ตั้งแต่ 0 ถึง 300,000 เมื่อกราฟลงต่ำสุด

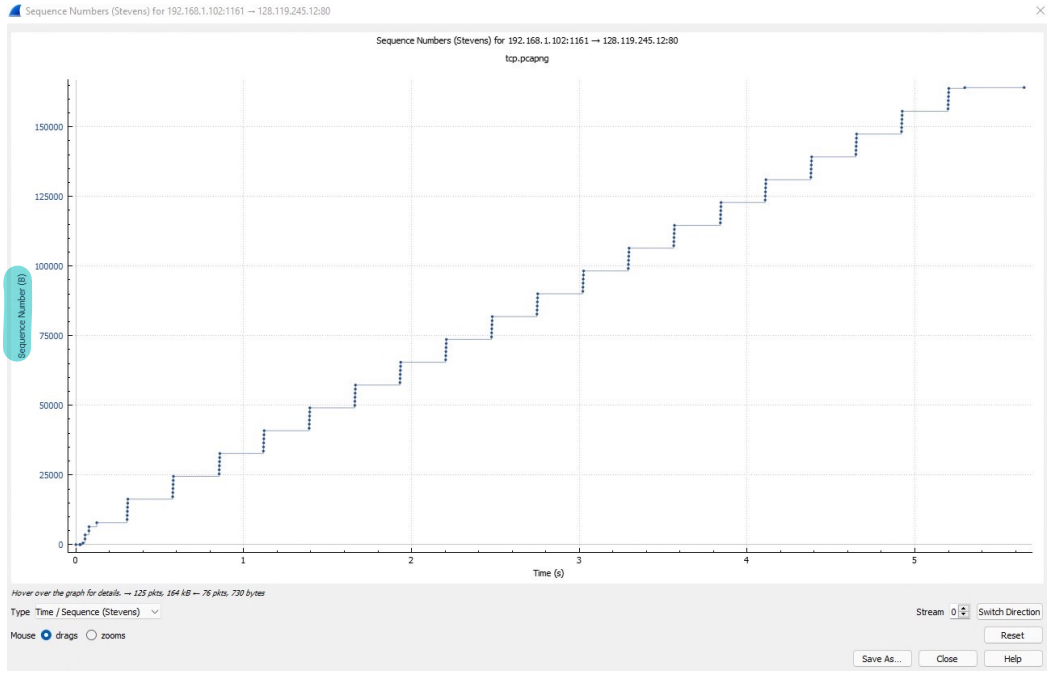
คือค่า window size = 0 และเมื่อ window update จะกระโดดขึ้นไปแนวดิ่งไปที่ประมาณ

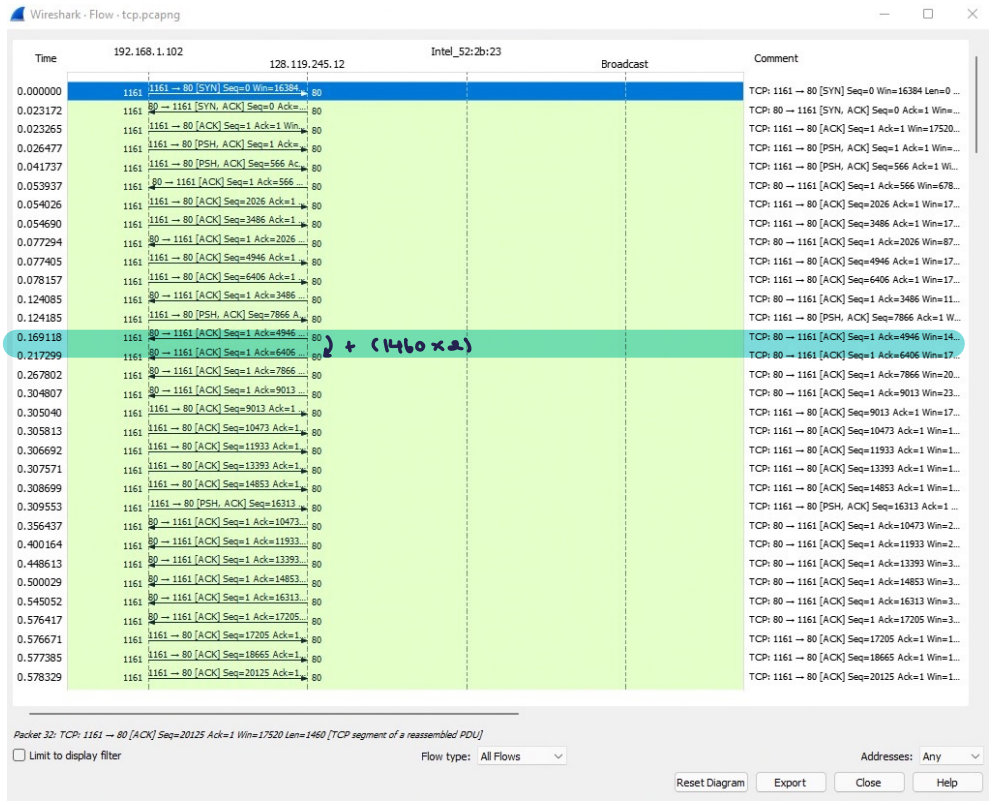
240,000



6. ในการควบคุม congestion control ของ TCP จะมีหลักอยู่ 2 ข้อ คือ Slow Start และ Congestion Avoidance ให้เปิดไฟล์ tcp.pcapng แล้วดูที่ Statistics->TCP Stream Graph-> Time-Sequence-Graph(Stevens) โดยแต่ละจุดแสดงถึงการส่งในแต่ละ segment ร่วมกับ Statistics-> Flow Graph นักศึกษาสามารถบอกได้หรือไม่ว่า Slow Start เริ่มต้นและสิ้นสุดที่ใด และมี Congestion Avoidance เกิดขึ้นหรือไม่ ให้อธิบาย พร้อมรูปประกอบ

ช่วงที่ slow start เริ่มส่ง packet ที่ 2 ขึ้นมา (Congestion window)
มีค่าเพิ่มขึ้นหนึ่งเท่า ของ window size (1460 byte) ขึ้นไปถึง 62,780 bytes
ถึงจะหยุดการเพิ่มขนาด window size และไปสู่ Congestion Avoidance ต่อไป





งานครั้งที่ 8

- การส่งงาน เขียนหรือพิมพ์ลงในเอกสารนี้ และส่งโดยเป็นไฟล์ PDF เท่านั้น
- ตั้งชื่อไฟล์โดยใช้รหัสนักศึกษา และ _Lab8 เช่น 63010789_Lab6.pdf
- กำหนดส่ง ภายในวันที่ 23 มีนาคม 2565