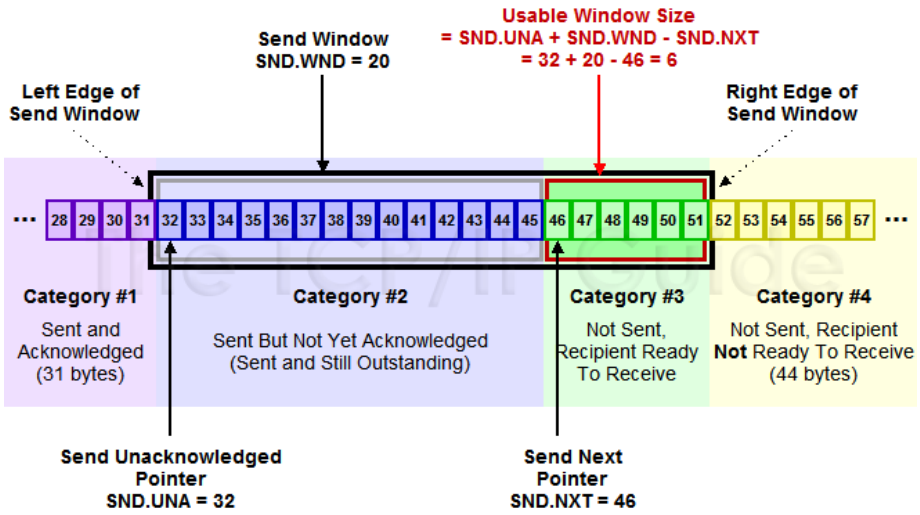
กิจกรรมที่ 8 : TCP Window

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

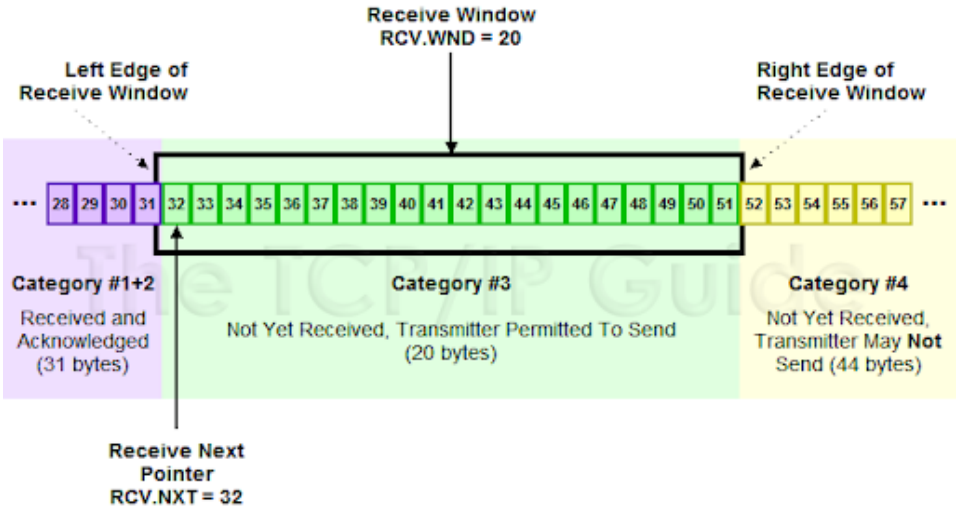
ใน send window จะแบ่งออกเป็น 4 ส่วน คือ

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

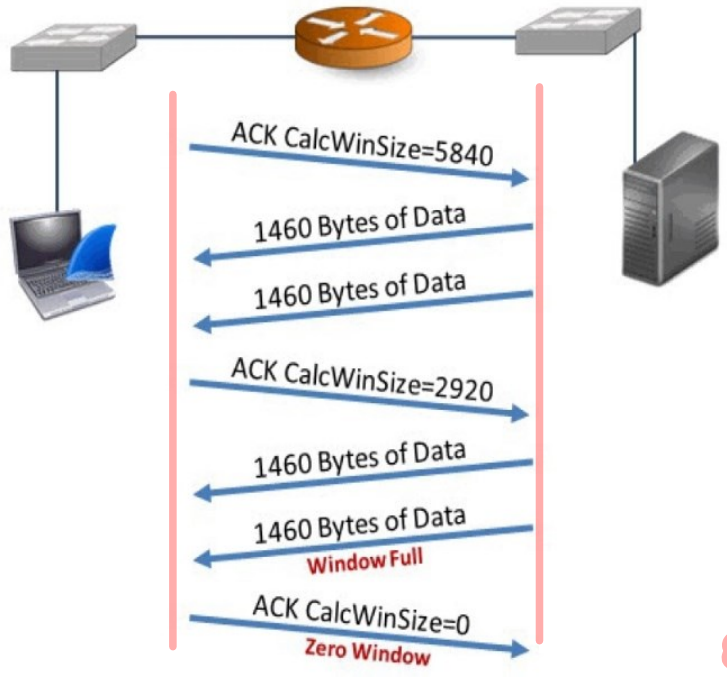


ใน receive window จะแบ่งเป็น 2 ส่วน

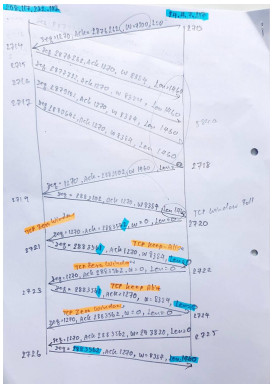
- ข้อมูลที่รับแล้วและ acknowledge ไปแล้ว
- ข้อมูลพร้อมจะรับ



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 หมายเลข 2718 ซึ่งเป็น packet ที่ host 24.4.7.217 ส่ง ACK กลับมา โดยมี window size เหลือเพียง 1,460 ไบต์
   - ต่อมาใน packet หมายเลข 2719 พบว่า host 208.117.232.102 มีการส่งข้อมูลไปอีก 1,460 ไบต์ ซึ่งจะทำให้เต็ม receive window พอดี และทำให้ wireshark สร้างข้อมูลแจ้งเตือนว่า window full
   - เมื่อถึง packet หมายเลข 2720 พบว่า host 24.4.7.217 ส่ง packet ACK กลับมา โดยมีค่า window size เป็น 0 ทำให้ wireshark สร้างข้อมูลแจ้งเตือนว่า zero window
   - ให้สังเกตช่วงเวลาระหว่าง packet หมายเลข 2720 และ 2721 จะเห็นว่ามีระยะห่างมากกว่าปกติ หมายความว่าฝั่งผู้ส่งเมื่อพบ zero window ก็จะรอฝั่งผู้รับให้เคลียร์ receive window เสียก่อน
   - ใน packet หมายเลข 2721 จะมีการส่ง packet keep alive (คือ packet ACK ที่ไม่มีข้อมูล จากฝั่งผู้ส่ง ซึ่งจะเกิดขึ้นเมื่อ keepalive timer expire)
   - จากนั้นใน packet หมายเลข 2722 ผู้รับจะส่ง ACK กลับมา โดยมี window size เป็น 0 เช่นเดิม และเกิดซ้ำอีกครั้งใน packet หมายเลข 2723 และ 2724
   - จนกระทั่ง packet หมายเลข 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

2. ให้นักศึกษาตรวจสอบ zero window ระยะที่ 2 แล้วตอบคำถาม ต่อไปนี้

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

Window full 4021 , zero window 4022 , window update 4035

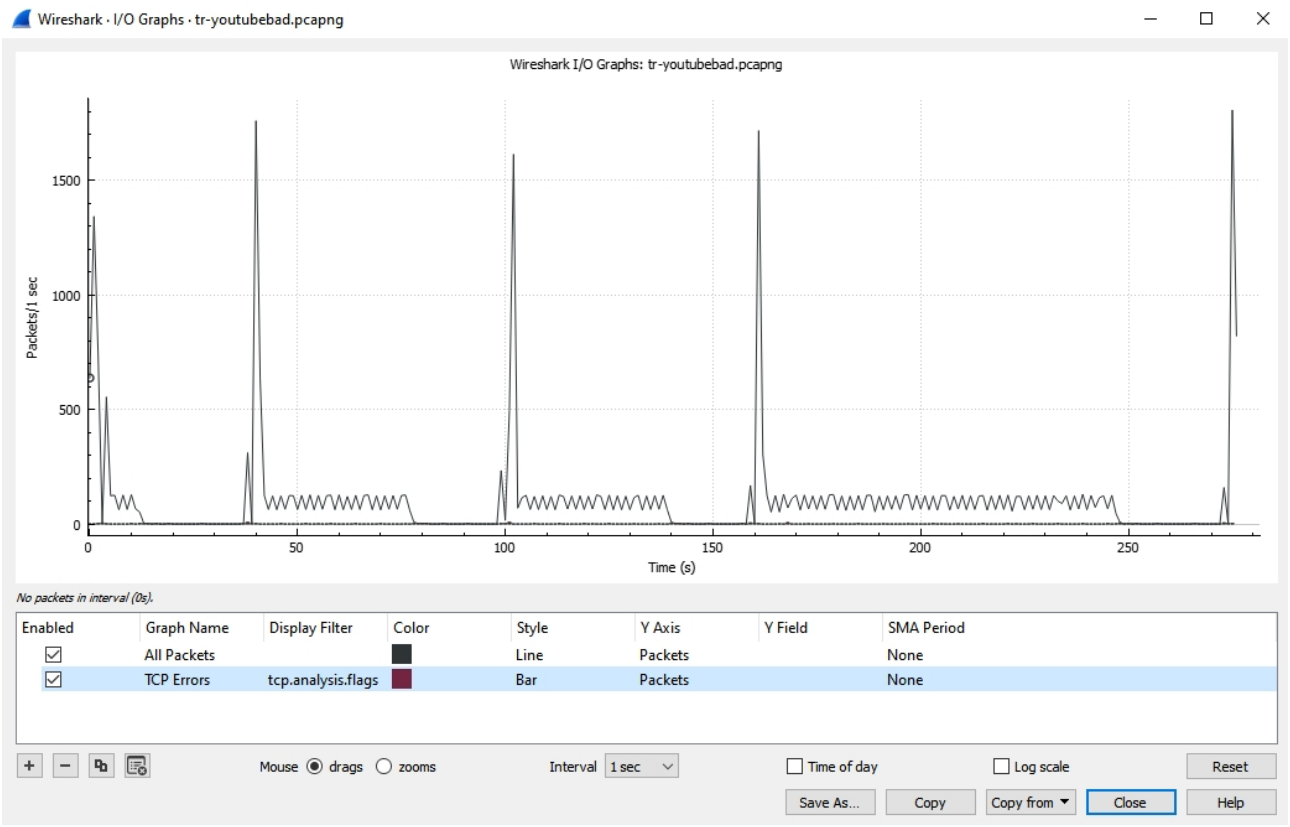
- หลังจากมีการทำ keep alive กี่ครั้ง มีช่วงระยะเวลาเท่าไรบ้าง นับจาก zero window ครั้งก่อน

keep alive 6 times >> 0.477622 , 0.995377 , 1.878101 , 3.704205 , 7.398856 , 10.020023

- ระยะเวลาตั้งแต่เกิด zero window ครั้งแรกจนถึง window update ใช้เวลาเท่าไร

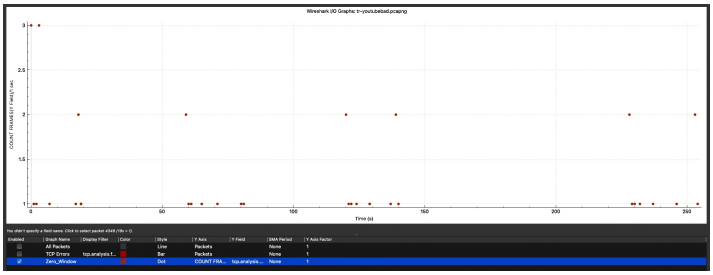
25.430224

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

- ให้ Disable กราฟเดิมที่มีอยู่ทุกกราฟ โดยคลิกที่ช่องสี่เหลี่ยมในคอลัมน์ Enabled ของแต่ละกราฟเพื่อนำเครื่องหมายถูกออก
- 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



- กราฟใหม่ที่พึ่งสร้างขึ้นบอกข้อมูลอะไร

บอกจำนวน packet zero window เทียบกับ เวลา

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

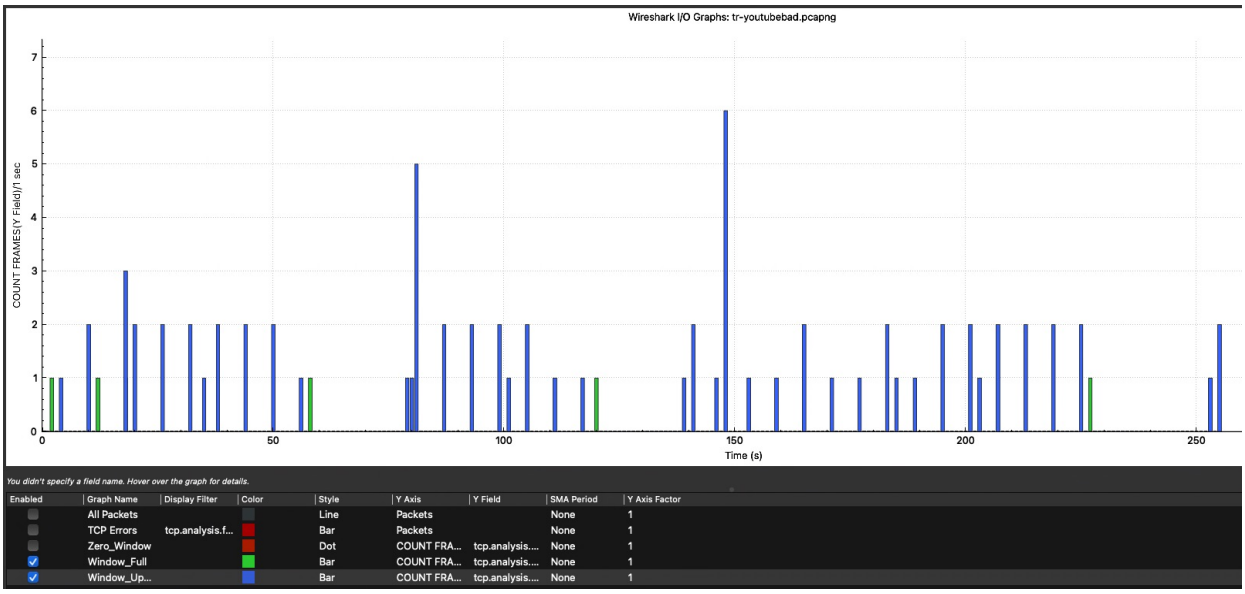
เขียว แสดง จำนวน packet Window full

น้ำเงิน แสดง ——— Window update

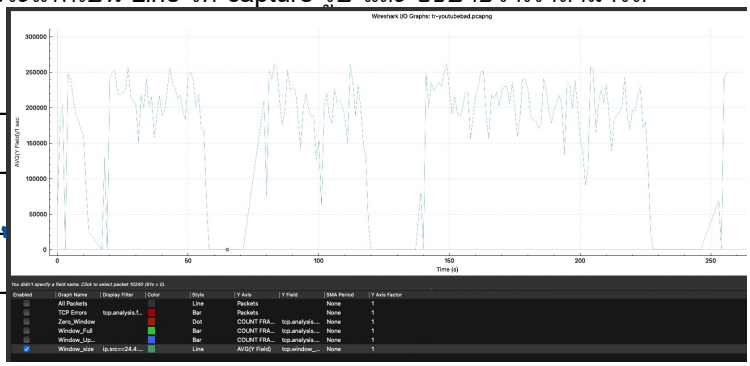
เทียบกัน เวลา (s)

- จากกราฟสามารถบอกได้หรือไม่ว่ามี window full กี่ครั้ง ให้บันทึกภาพ screenshot ประกอบด้วย

บอกได้ มี 5 ครั้ง

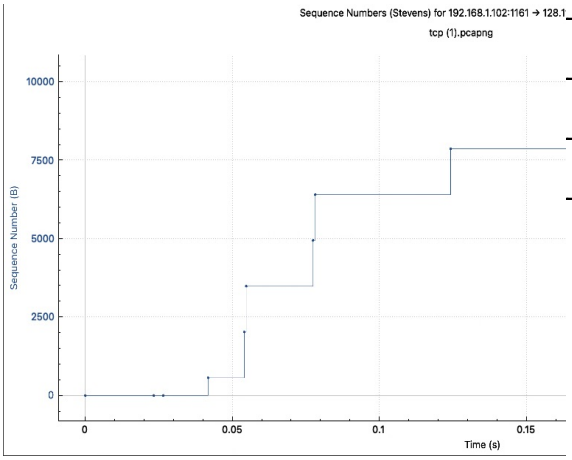


5. ให้สร้าง I/O Graph ใหม่ โดยในช่อง Display Filter ให้ใส่ ip.src==24.4.7.217 ใน Y(AXIS) ใช้ AVG(*) และช่อง Y Field ใช้ tcp.window_size กำหนดประเภทเป็น Line ให้ capture รูป และ อธิบายว่าเราสามารถวิเคราะห์ข้อมูลอะไรจากกราฟนี้



แสดง window size โดยเฉลี่ย
และแสดงเฉพาะ ip 24.47.217
ซึ่งสามารถนำมาวิเคราะห์หา window full, zero win, window update

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) จากนั้นคลิกที่ปุ่ม Switch Direction เพื่อเปลี่ยนทิศทางให้เป็นทิศที่ส่งจาก host 192.168.1.102 ส่งไปยัง host 128.119.245.12 โดยแต่ละจุดแสดงถึงการส่งในแต่ละ TCP segment ให้พิจารณากราฟนี้ร่วมกับกราฟจาก Statistics-> Flow Graph นักศึกษาสามารถบอกได้หรือไม่ว่า Slow Start เริ่มต้นและสิ้นสุดที่ใด และมี Congestion Avoidance เกิดขึ้นหรือไม่



เกิด slow start ที่วินาทีที่ 0.0-0.1
Congestion Avoidance ตั้งแต่วินาทีที่ 0.1