หลักการวัดประสิทธิภาพในชั้น Transport Layer (TCP)

การตัดสินว่าแพ็กเก็ตใดคือ Request และ Response

เพื่อวัดประสิทธิภาพของเป้าหมายในชั้น Transport Layer ระบบควรจำแนกได้ว่าแพ็กเก็ตใดคือ Request จาก Client ที่มายังเป้าหมายและแพ็กเก็ตใดคือ Response จากเป้าหมายไปยัง Client

โดยในโปรโตคอล TCP นั้นจะตรวจสอบได้จากทิศทางการไหลของข้อมูลและ Payload

- Request: มักจะเป็นแพ็กเก็ตแรกที่ส่งข้อมูล (Payload > 0) หลังจากทำ Three-way Handshake เสร็จสิ้น โดยส่งจากฝั่ง Client ไปยัง Server (เช่น HTTP GET, Database Query)
- Response: คือแพ็กเก็ตที่ส่งข้อมูลสวนทางกลับมาจาก Server ไปยัง Client (เช่น HTTP 200 OK, ผลลัพธ์จากการ Query) โดยมักจะมีเลข Acknowledgment Number ที่ตรงกับเลข Next Sequence Number ของฝั่ง Request

การตัดสินว่า Request และ Response อยู่ในการสนทนาเดียวกัน

เป็นขั้นตอนที่สำคัญเพื่อใช้ในการจำแนกว่าแพ็กเก็ต Response ที่เจอนั้นอยู่ในการสนทนาเดียวกันกับแพ็กเก็ต Request ก่อนหน้าหรือไม่ โดยมีวิธีคือสร้างตาราง ดังนี้

1. นำข้อมูล Source IP, Destination IP, Source Port, Destination Port, Protocal จากทั้ง Request และ Response มาตรวจสอบ
2. สร้าง hash ที่มี Key คือ Source IP, Destination IP, Source Port, Destination Port, Protocal ตามข้อมูลแพ็กเก็ต ซึ่งเป็น Key แบบสองทิศทาง เช่น A ไป B และ B ไป A ต้องระบุได้ว่าเป็นค่าเดียวกัน
3. เพื่อให้ได้ Key ตามข้อ 2 จะนำ IP ทั้ง 2 มาหา Min-Max และทำแบบเดียวกับ Port ทั้ง 2 เช่นกัน จากนั้นนำมาจัดทำให้อยู่ในรูปแบบนี้

[Address_Low][Address_High][Port_Low][Port_High][Protocol]

4. นำผลลัพธ์จากข้อ 3 เข้าสู่ฟังก์ชัน **Jenkins Hash** จะได้ค่า Hash ออกมา (เช่น 0x1A2B3C4D)

หากแพ็กเก็ตที่ตรวจสอบมีค่า Hash ตรงกับค่าที่มีอยู่ในระบบ สามารถระบุได้ว่าแพ็กเก็ตดังกล่าวอยู่ในสายการสนทนา (Flow) เดียวกัน แต่หากไม่พบข้อมูล จะทำการสร้าง รายการใหม่ในตารางสถานะเพื่อรอรับแพ็กเก็ตถัดไปที่มีค่า Hash ตรงกัน

การตัดสิดว่า Response นั้นเป็นการตอบกลับของ Request ก่อนหน้าหรือไม่

เมื่อทำการยืนยันแล้วว่าแพ็กเก็ตทั้งคู่นั้นในอยู่ในการสนทนาเดียวกัน ขั้นต่อมาคือการตรวจสอบว่า Response ที่มาถึงนั้นคือ Response ที่ตอบกลับ Request ก่อนหน้าจริงหรือไม่ โดยจากการทำงานโปรโตคอล TCP จะสามารถแบ่งช่วงการสนทนาได้ 3 ช่วงคือ

- **ขั้นตอนการสร้าง (Handshake)**
- **ขั้นตอนการส่งข้อมูล (Data)**
- **ขั้นตอนการปิด (Close)**

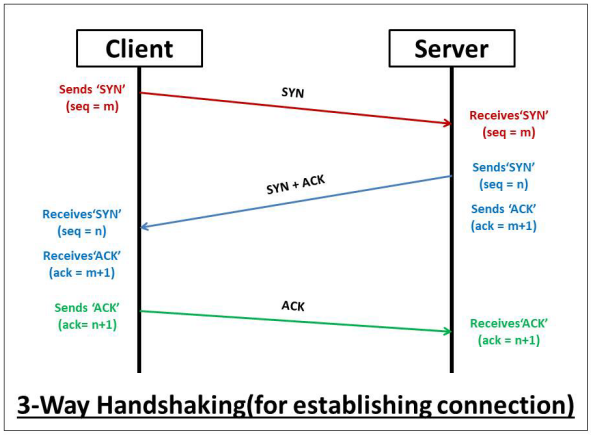
ทั้ง 3 ช่วงมีวิธีการจับคู่ Request และ Response บางส่วนคล้ายคลึงกันและบางส่วนแตกต่างกัน โดยมีรายละเอียดดังนี้

ขั้นตอนการสร้าง (Handshake)

ในการจับคู่ช่วงเริ่มการติดต่อ ตัวข้อมูล (Payload) นั้นยังไม่มี แต่ละฝั่งจะใช้การบวกเลข 1 เข้าไปใน Acknowledgment Number เพื่อยืนยันการได้รับแล้ว โดยสมมุติการติดต่อกันระหว่าง A และ B ดังนี้

- **คู่ที่ 1** A ส่ง SYN ที่มี Sequence Number X ไปยัง B จากนั้น B ตอบกลับด้วย SYN-ACK ที่มี Acknowledgment Number X+1 หากเป็นไปตามขั้นตอนดังนี้สรุปได้ว่า SYN-ACK นี้เป็น Response ของ SYN ก่อนหน้า

- **คู่ที่ 2** SYN-ACK ที่ B ส่งนั้น มีการเริ่ม Sequence Number Y เพื่อเริ่มการสนทนาแบบ Full Duplex จาก A ตอบกลับด้วย ACK ที่มี Acknowledgment Number Y+1 หากเป็นไปตามขั้นตอนดังนี้สรุปได้ว่า ACK นี้เป็น Response ของ SYN-ACK ก่อนหน้า



https://static.afteracademy.com/images/what-is-a-tcp-3-way-handshake-process-three-way-handshaking-establishing-connection-6a724e77ba96e241.jpg

ขั้นตอนการส่งข้อมูล (Data)

ในขั้นตอนนี้จะมีการส่งข้อมูล Payload เกิดขึ้น วิธีการจับคู่ในขั้นตอนนี้จะตรวจสอบ Ack ที่ตอบกลับมาจะต้องเท่ากับ Sequence Number + ขนาดของ Payload โดยสมมุติการติดต่อกันระหว่าง A และ B ดังนี้

- A ส่ง Data โดยมี Sequence Number 100 และขนาดของข้อมูลคือ 500 bytes ไปที่ B
- B ยืนยันโดยตอบกลับแพ็กเก็ตที่มี Acknowledgment Number 600 (100+500) หากเป็นไปตามขั้นตอนนี้สรุปได้ว่า ACK นี้เป็น Response ของ Data ก่อนหน้า

สามารถสรุปเป็นสูตรคำนวณได้ดังนี้

$$\text{Expected ACK} = \text{Sequence Number} + \text{TCP Payload Length}$$

เงื่อนไขอื่นในการจับคู่

ในหลายครั้งที่การจราจรของเครือข่ายไม่เป็นไปตามลำดับและหลักการ จึงมีเงื่อนเพิ่มเติมสำหรับการจับคู่ Request และ Response ดังนี้

1. หากพบแพ็กเก็ตที่มี Sequence Number เดิมส่งมาอีกครั้ง (Retransmission) จะไม่นำไปจับคู่เพื่อตามกฎของ Karn เพื่อป้องกันค่า RTT ที่ผิดเพี้ยน
2. หากมีแพ็กเก็ตข้อมูลส่งมามากกว่า 1 ต้องคำนวณค่า ACK Number ที่คาดหวังจากแพ็กเก็ตข้อมูลตัวสุดท้ายที่มาถึง เงื่อนไขนี้จะรวมถึงการข้ามลำดับ ซึ่งสามารถตรวจสอบที่ Sequence Number ของแพ็กเก็ตว่ามีการไล่ลำดับหรือไม่

ขั้นตอนการปิด (Close)

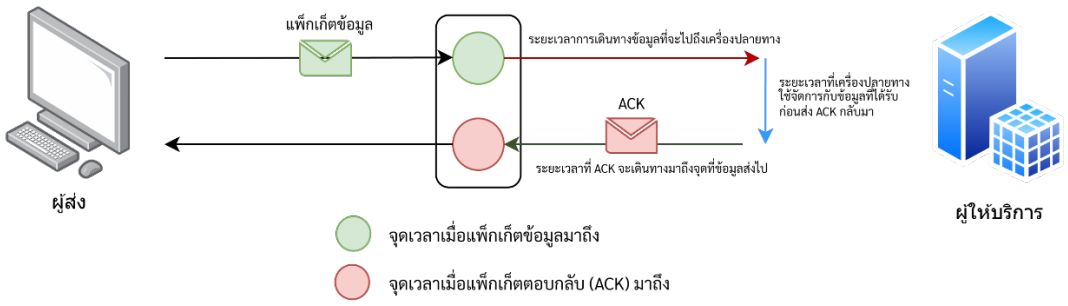
คล้ายกับขั้นตอนการสร้าง แม้ไม่มีข้อมูล แต่การส่ง Flag FIN ถือเป็นการใช้เลข Sequence Number ไป 1 หน่วย

- คู่ที่ 1: A ต้องการปิดการสนทนาจึงส่ง FIN ที่มี Sequence Number Z จากนั้น B ตอบกลับด้วย ACK ที่มี Acknowledgment Number Z+1 หากเป็นไปตามขั้นตอนนี้สรุปได้ว่า ACK นี้เป็น Response ของ FIN ก่อนหน้า
- คู่ที่ 2: B ส่ง FIN ที่มี Sequence Number W จากนั้น A ตอบกลับด้วย ACK ที่มี Acknowledgment Number W+1 ถือเป็นอันจบการสนทนา หากเป็นไปตามขั้นตอนนี้สรุปได้ว่า ACK นี้เป็น Response ของ FIN ก่อนหน้า

ค่าดัชนีการวัดประสิทธิภาพ (Metrics)

ACK Response Time

ค่าดัชนีนี้คือส่วนต่างของเวลาระหว่าง **แพ็กเก็ตข้อมูลที่ส่งไป** กับ **แพ็กเก็ตยืนยัน (ACK) ที่ตอบกลับมา** ซึ่งบอกระยะเวลาการเดินทางข้อมูลที่จะไปถึงเครื่องปลายทาง ระยะเวลาที่เครื่องปลายทางใช้จัดการกับข้อมูลที่ได้รับก่อนส่ง ACK กลับมา และระยะเวลาที่ ACK จะเดินทางมาถึงจุดที่ข้อมูลส่งไป



การคำนวณ

ตรวจหาแพ็กเก็ตที่มีข้อมูล จากนั้นตรวจหาแพ็กเก็ตที่อยู่ในสายการสนทนาเดียวกัน (Flow) ที่เป็นการยืนยัน (ACK) ว่าได้รับข้อมูลนี้แล้วนำ Timestamp ของทั้ง 2 มาหาส่วนต่างตามสูตรด้านล่าง

$$\text{ACK Response Time} = T_{ACK} - T_{Segment}$$

- $T_{Segment}$: เวลาที่ส่ง TCP Segment ที่มีข้อมูล (Payload) ออกไป
- T_{ACK}: เวลาที่ได้รับ ACK Packet ที่ระบุหมายเลข Acknowledgment Number ตรงกับ Next Sequence Number ของ Segment นั้นพอดี

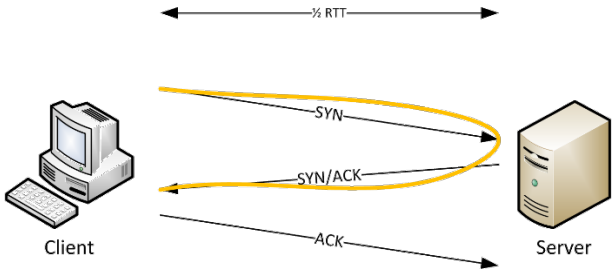
Handshake Time/Initial Round Time Trip

คือ **ระยะเวลาในการทำ TCP 3-way Handshake** ซึ่งมีขนาดแพ็กเก็ตที่เล็กและคงที่ (ปกติไม่เกิน 74 bytes) ทำให้ค่า Transmission Delay ต่ำ และยังไม่มีการประมวลผลของแอปพลิเคชัน ทำให้ไม่มี Delay ในส่วนนี้ ดังนั้นจึงบอกได้ว่าเป็นค่า Latency พื้นฐานของเส้นทางนั้นๆ (Baseline Latency) สามารถใช้เป็นเกณฑ์อ้างอิงในการเปรียบเทียบเมื่อเกิดปัญหาได้

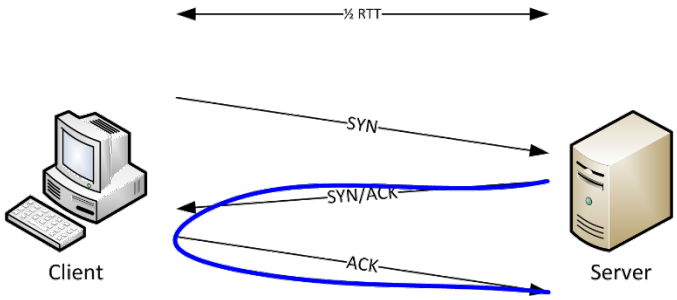
การคำนวณ

วัดช่วงเวลาไปและกลับของการทำงาน TCP 3-way Handshake โดยมีเงื่อนไขดังนี้

หากเราวัดจาก Client จะเป็นการวัดระยะเวลาระหว่าง SYN ถึง SYN/ACK



หากเราวัดจาก Server จะเป็นการวัดระยะเวลาระหว่าง SYN/ACK ถึง ACK



โดยการกำหนดว่าใครเป็น Server หรือ Client จะดูว่าฝั่งไหนเป็นผู้เริ่มสื่อสาร ในบริบทนี้คือการส่งแพ็กเก็ต SYN เพื่อเริ่ม TCP 3-way Handshake

ขอบคุณรูปภาพจาก https://blog.packet-foo.com/2014/07/determining-tcp-initial-round-trip-time/