

## 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 ( $\text{Payload} > 0$ ) หลังจากทำ Three-way Handshake เสร็จสิ้น โดยส่งจากฝั่ง Client ไปยัง Server (เช่น HTTP GET, Database Query)
- Response: คือแพ็กเก็ตที่ส่งข้อมูลตอบกลับมาจาก Server ไปยัง Client (เช่น HTTP 200 OK, ผลลัพธ์จากการ Query) โดยมักจะมีเลข Acknowledgment Number ที่ตรงกับเลข Next Sequence Number ของฝั่ง Request

### การตัดสินว่า Request และ Response อยู่ในการสนทนาระหว่างกัน

เป็นขั้นตอนที่สำคัญเพื่อใช้ในการจำแนกว่าแพ็กเก็ต Response ที่เจอนั้นอยู่ในการสนทนาระหว่างกับแพ็กเก็ต Request ก่อนหน้าหรือไม่ โดยมีวิธีคือสร้างตาราง ดังนี้

- นำข้อมูล Source IP, Destination IP, Source Port, Destination Port, Protocol จากทั้ง Request และ Response มาตรวจสอบ
- สร้าง hash ที่มี Key คือ Source IP, Destination IP, Source Port, Destination Port, Protocol ตามข้อมูลแพ็กเก็ต ซึ่งเป็น Key แบบสองทิศทาง เช่น A ไป B และ B ไป A ต้องระบุได้ว่าเป็นค่าเดียวกัน
- 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]

- นำผลลัพธ์จากข้อ 3 เข้าสู่ฟังก์ชัน Jenkins Hash จะได้ค่า Hash ออกมา (เช่น 0x1A2B3C4D)

หากแพ็กเก็ตที่ตรวจสอบมีค่า Hash ตรงกับค่าที่มีอยู่ในระบบ สามารถระบุได้ว่าแพ็กเก็ตตั้งกล่าวอยู่ในสายการสนทนา (Flow) เดียวกัน แต่หากไม่พบข้อมูล จะทำการสร้าง รายการใหม่ในตารางสถานะเพื่อรับแพ็กเก็ตตัดไปที่มีค่า Hash ตรงกัน

### การตัดสินว่า Response นั้นเป็นการตอบกลับของ Request ก่อนหน้าหรือไม่

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

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

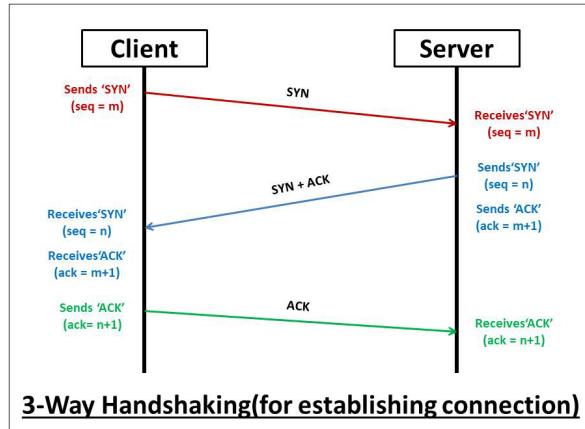
ทั้ง 3 ช่วงมีวิธีการจับคู่ Request และ Response บางส่วนคล้ายคลึงกันและบางส่วนแตกต่างกัน โดยมีรายละเอียดดังนี้

#### ขั้นตอนการสร้าง (Handshake)

ในการจับคู่ช่วงเริ่มการติดต่อ ตัวข้อมูล (Payload) นั้นยังไม่มี แต่ละฝ่ายจะใช้การบวกเลข 1 เข้าไปใน Acknowledgment Number เพื่อสื่อถึงการได้รับแล้ว โดยสมมุติการติดต่อ กันระหว่าง A และ B ดังนี้

- ครั้งที่ 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 เพื่อเริ่มการส่งหน้าแบบ 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 โดยสมมุติการติดต่อ กันระหว่าง A และ B ดังนี้

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

สามารถสรุปเป็นสูตรคำนวนได้ดังนี้

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

### ขั้นตอนการปิด (Close)

คล้ายกับขั้นตอนการสร้าง แม้ไม่มีข้อมูล แต่การส่ง Flag FIN ถือเป็นการใช้เลข Sequence Number ไป 1 หน่วย

- คู่ที่ 1: A ต้องการปิดการส่งหน้าจึงส่ง 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 ถือเป็นอันจบการส่งหน้า หากเป็นไปตามขั้นตอนนี้สรุปได้ว่า ACK นี้เป็น Response ของ FIN ก่อนหน้า

## การตรวจสอบความผิดปกติของการทำงานในโปรโตคอล TCP

### 1. การส่งข้าม Retransmission

ผู้ส่งมีการส่งแพ็กเก็ตซ้ำ เนื่องจาก 2 กรณีดังนี้

1. **Fast Retransmission** การส่งข้ามที่เกิดจากผู้ส่งได้รับ ACK ที่มี Acknowledgment Number เมื่อ晚กัน 3 ครั้งจากผู้รับซึ่งบ่งบอกว่าทางผู้รับนั้นกำลังรอข้อมูลลำดับนั้นๆ อยู่
2. **Timeout Retransmission** การส่งข้ามที่เกิดจากเวลาที่ผู้ส่งนั้นรอการตอบรับ ACK จากผู้รับหมดลง ผู้ส่งจะตัดสินว่าแพ็กเก็ตที่ส่งนั้นไปไม่ถึงและจะทำการส่งข้ามอีกครั้ง

เมื่อพบแพ็กเก็ตใหม่ทำการตรวจสอบเงื่อนไขดังนี้

1. ค่า Sequence Number ของ Packet ปัจจุบันน้อยกว่า Expected Sequence และอยู่ในช่วง Sequence ที่เคยบันทึกก่อนหน้า
2. ต้องเป็นพิกัดเดียวกับแพ็กเก็ตที่เคยบันทึกไว้ก่อนหน้า
3. แพ็กเก็ตนั้นมีความยาวข้อมูลมากกว่า 0

หากเป็นไปตามเงื่อนไขเหล่านี้จะตัดสินว่าเป็นแพ็กเก็ตที่มีการส่งซ้ำ

### 2. Previous Segment Not Found

ตรวจสอบความไม่ต่อเนื่องของลำดับข้อมูล โดยค่าหมายเลขลำดับที่ได้รับมีค่ามากกว่าหมายเลข Sequence Number ที่คาดหวัง ซึ่งบ่งบอกว่ามีข้อมูลบางส่วนสูญหายระหว่างการรับส่ง หรือเกิดความผิดพลาดในการดักจับแพ็กเก็ตของระบบ

เมื่อพบแพ็กเก็ตใหม่ทำการตรวจสอบเงื่อนไขดังนี้

1. ค่า Seq ของแพ็กเก็ตปัจจุบันมากกว่า ค่าที่บันทึกไว้

### 3. Zero Window

เมื่อบาฟเฟอร์ของผู้รับเต็ม ผู้รับจะทำการแจ้งค่าขนาดหน้าต่างตอบรับ (Window Size) เป็น 0 เพื่อระงับการส่งข้อมูลจากผู้ส่งชั่วคราว จนกว่าทรัพยากรในระบบจะพร้อมรับข้อมูลใหม่อีกครั้ง

เมื่อพบแพ็กเก็ตใหม่ทำการตรวจสอบเงื่อนไขดังนี้

1. ค่าในฟิลด์ Window Size ใน TCP Header มีค่าเท่ากับ 0
2. ไม่รวมแพ็กเก็ตที่เป็น SYN, FIN หรือ RST

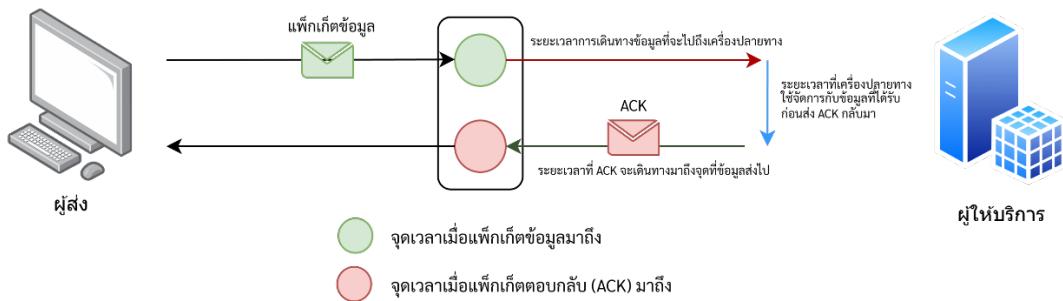
## ความผิดปกติของการทำงานในโปรโตคอล TCP อ้างอิงจาก

[https://www.wireshark.org/docs/wsug\\_html\\_chunked/ChAdvTCPAnalysis.html](https://www.wireshark.org/docs/wsug_html_chunked/ChAdvTCPAnalysis.html)

## ค่าดัชนีการวัดประสิทธิภาพ (Metrics)

### ACK Response Time

ค่าดัชนีนี้คือส่วนต่างของเวลาระหว่าง แพ็คเก็ตข้อมูลที่ส่งไป กับ แพ็คเก็ตยืนยัน (ACK) ที่ตอบกลับมา ซึ่งบอกระยะเวลาการเดินทางข้อมูลที่จะไปถึงเครื่องปลายทาง ระยะเวลาที่เครื่องปลายทางใช้จัดการกับข้อมูลที่ได้รับก่อนส่ง ACK กลับมา และระยะเวลาที่ ACK จะเดินทางมาถึงจุดที่ข้อมูลส่งไป



### การคำนวณ

ตรวจสอบแพ็คเก็ตที่มีข้อมูล จากนั้นตรวจสอบแพ็คเก็ตที่อยู่ในสายการสันหนาเดียวกัน (Flow) ที่เป็นการยืนยัน (ACK) ว่าได้รับข้อมูลนี้แล้วนำ Timestamp ของทั้ง 2 มาหาส่วนต่างตามสูตรด้านล่าง

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

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

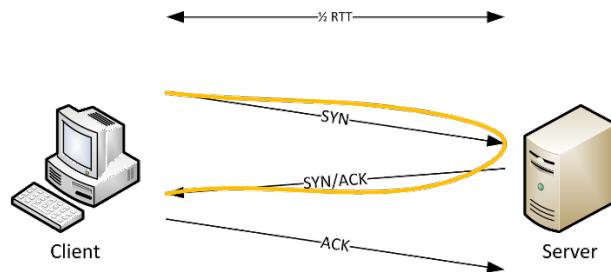
### Handshake Time/Initial Round Trip

คือ ระยะเวลาในการทำ TCP 3-way Handshake ซึ่งมีขนาดแพ็คเก็ตที่เล็กและคงที่ (ปกติไม่เกิน 74 bytes) ทำให้ค่า Transmission Delay ต่ำ และยังไม่มีการประมวลผลของแอปพลิเคชัน ทำให้ไม่มี Delay ในส่วนนี้ ดังนั้นจึงบอกได้ว่าเป็นค่า Latency พื้นฐาน ของเส้นทางนั้นๆ (Baseline Latency) สามารถใช้เป็นเกณฑ์อ้างอิงในการเปรียบเทียบเมื่อเกิดปัญหาได้

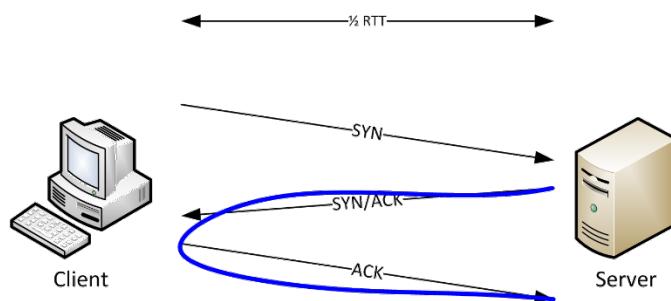
### การคำนวณ

วัดช่วงเวลาไปและกลับของการทำงาน TCP 3-way Handshake โดยมีเงื่อนไขดังนี้

หากเราวัดจาก Client จะเป็นการวัดระยะเวลาระหว่าง SYN ถึง SYN/ACK



หากเราดูจาก Server จะเป็นการวัดระยะเวลาระหว่าง SYN/ACK ถึง ACK



โดยการกำหนดว่าใครเป็น Server หรือ Client จะดูว่าฝ่ายไหนเป็นผู้เริ่มสื่อสาร ในบริบทนี้คือการส่งแพ็คเก็ต SYN เพื่อเริ่ม TCP 3-way Handshake

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