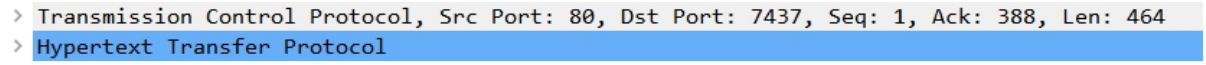
กิจกรรมที่ 4 : HTTP

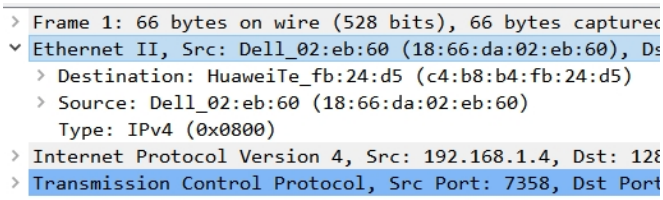
ในกิจกรรมที่ผ่านมา จะเป็นการแนะนำการใช้งาน Wireshark เป็นส่วนใหญ่ในกิจกรรมครั้งนี้ จะเริ่มทำความรู้จักกับ protocol ใน Application Layer โดย protocol แรก คือ HTTP (Hypertext Transport Protocol)

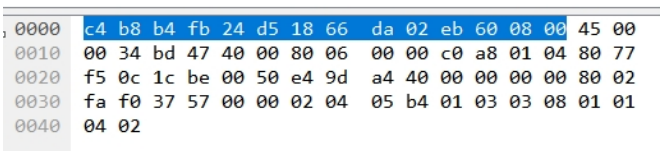
1. ให้ใช้ Wireshark เริ่มทำการ Capture และป้อน url : http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html เสร็จแล้วให้หยุด
2. ให้ใช้ display filter : http เพื่อให้แสดงเฉพาะ HTTP (ที่ถูกต้องควรจะมีแค่ 2 แพ็กเกต ในกรณีที่มีเกิน 2 แพ็คเกต อาจมาจากกรณี favicon ติดมาด้วย แต่ไม่ต้องไปสนใจแพ็คเกตที่เกินมา)
   (กรณีบรรทัดที่ 2 (Response) เป็น 304 Not Modified ให้เคลียร์ cache ของ browser แล้วทำใหม่)
3. ใน Packet List Pane ให้เลือก packet ที่เป็น HTTP Response และหาว่ามีความยาวของทั้ง frame เป็นเท่าไร

__กรณี Request มีความยาวไม่แน่นอน ขึ้นกับ Browser กรณี Response ยาว 5xx ไบต์
เช่น TCP payload = 464 + Header Ethernet 14 ไบต์ + Header IP 20 ไบต์ + Header TCP 20 ไบต์__

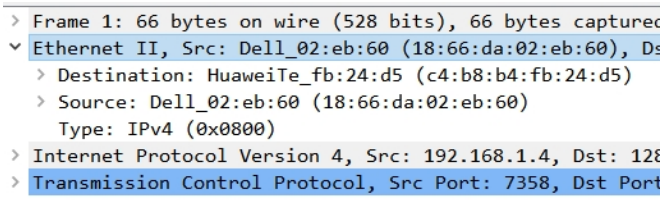


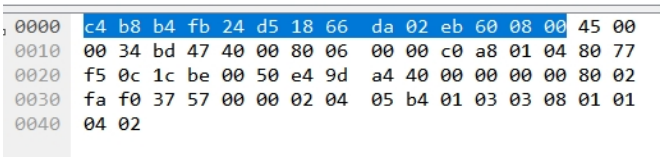
4. ใน packet ตามข้อ 3 ความยาวเฉพาะส่วน header ของ Ethernet II เป็นเท่าไร __14 ไบต์__ ให้บันทึก screenshot หน้าจอส่วนที่แสดงความยาวมาแสดง (Hint: หาข้อมูลจาก Packet Byte Pane)





5. ใน packet ตามข้อ 3 ความยาวเฉพาะส่วน header ของ Transmission Control Protocol เป็นเท่าไร __20 ไบต์__ ให้บันทึก screenshot หน้าจอส่วนที่แสดงความยาวมาแสดง

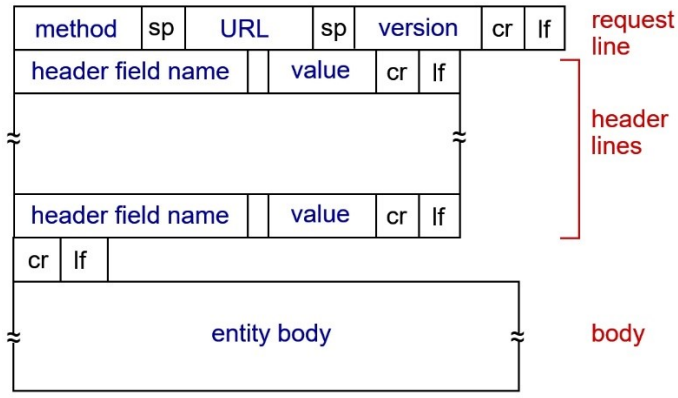




6. เพราะเหตุใด header ของ packet ต้องซ้อนเป็นชั้นๆ จงอธิบายเหตุผล

__ใน Internet Protocol Suite ได้แบ่งความรับผิดชอบของแต่ละชั้น (Layer) เอาไว้ชัดเจน ซึ่งช่วยให้นักออกแบบโปรโตคอลสามารถให้ความสำคัญกับโปรโตคอลในชั้นใดชั้นหนึ่งโดยไม่ต้องสนใจถึงชั้นอื่นๆ ในการทำงานจึงเป็นการห่อหุ้ม (Encapsulation) เป็นลำดับชั้น โดยส่งต่อลงมาทีละ Layer ในแต่ละ Layer ก็จำเป็นต้องมีข้อมูลสำหรับใช้ใน Layer เดียวกันที่ปลายทาง จึงมีการซ้อนเป็น Layer โดยเมื่อถึงปลายทาง ก็จะมีการถอด Header ออกไป คล้ายกับเปิดซองจดหมาย__

7. จากรูปแบบของ HTTP Message ตามรูป และ HTTP Request และ Response ที่ดักจับได้ ให้ตอบคำถามต่อไปนี้ (สามารถใช้วิธี capture แล้ว highlight ข้อมูลเพื่อตอบคำถามได้)



- browser และ server ใช้ HTTP version ใด __HTTP/1.1__
- browser เป็นโปรแกรมอะไร __โปรแกรมอะไรก็ได้ที่เป็นชื่อ Browser __
- server เป็นโปรแกรมอะไร __ Apache 2.4.6__
- ภาษาที่ browser ระบุว่าสามารถรับจาก server ได้ __ไม่แน่นอนขึ้นกับ Browser __
- status code ที่ส่งกลับมาจาก server มายัง browser __200 OK __
- ค่าของ Last-Modified ของไฟล์ที่ server __ขึ้นกับวันที่ดึงข้อมูล__
- มีข้อมูลกี่ไบต์ที่ส่งมายัง browser __128 ไบต์ (Content Length)__

- ให้สรุปว่า header field name ตาม HTTP message format ของข้อมูลที่ส่งกลับมีอะไรบ้าง

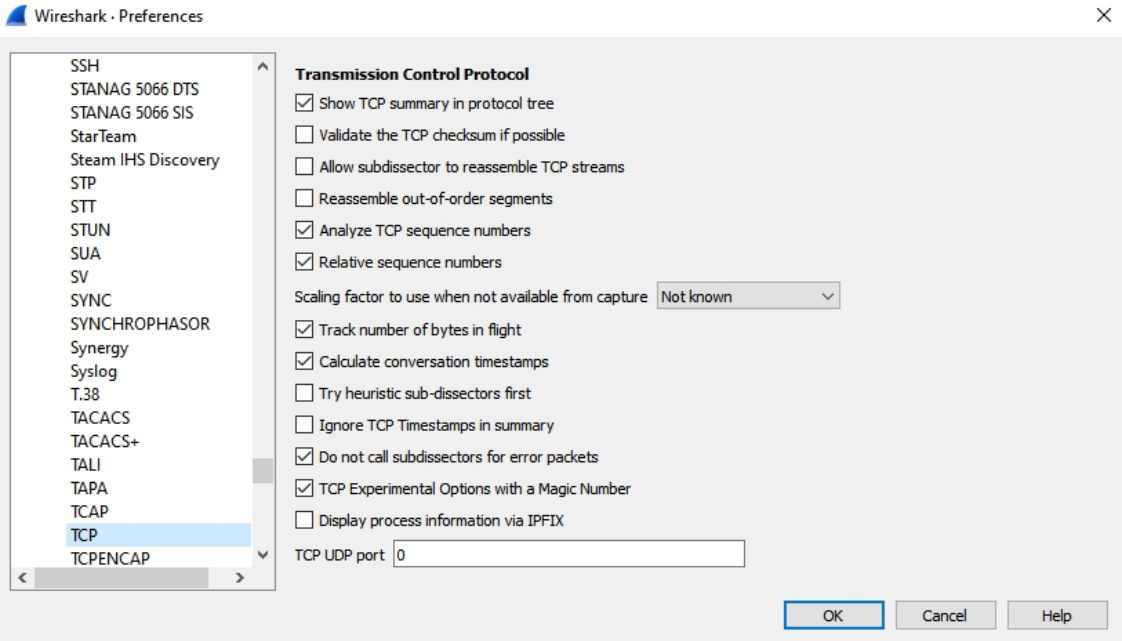
  __Date, Server, Last-Modified, ETAG, Accept-Ranges, Content-Length, Keep-Alive, Connection, Content-Type__

```
Date: Tue, 09 Feb 2021 12:36:14 GMT\r\n
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.14 mod_perl/2.0.11 Perl/v5.16.3\r\n
Last-Modified: Tue, 09 Feb 2021 06:59:01 GMT\r\n
ETag: "80-5bae1d2479c57"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 128\r\n
Keep-Alive: timeout=5, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=UTF-8\r\n
```

8. ให้นักศึกษาหาวิธี clear cache ของ browser ที่ตนเองใช้อยู่ แล้วจัดการ clear ให้เรียบร้อย
9. เปิด Wireshark ใหม่แล้ว capture การเรียกหน้าเว็บเพจไปยัง url http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html จากนั้นให้กด refresh เพื่อโหลดหน้าอีกครั้ง จากนั้นให้หยุด capture
10. ให้ใช้ display filter : http เพื่อให้แสดงเฉพาะ HTTP (ที่ถูกต้องควรจะมีแค่ 4 แพ็กเกต ในกรณีที่มีเกิน 4 แพ็คเกต อาจมาจากกรณี favicon ติดมาด้วย แต่ไม่ต้องไปสนใจแพ็คเกตที่เกินมา) และตอบคำถามต่อไปนี้
    - ใน HTTP GET ครั้งที่ 1 มีคำว่า IF-MODIFIED-SINCE หรือไม่ __ไม่มี__
    - ใน HTTP GET ครั้งที่ 2 มีคำว่า IF-MODIFIED-SINCE หรือไม่ __มี__
    - **(ถ้ามี) ข้อมูลที่ต่อจาก** IF-MODIFIED-SINCE มีความหมายอย่างไร

      __ Browser จะมีการ Cache ข้อมูลเอาไว้ ดังนั้นเมื่อ Browser ทราบว่าเป็นการโหลดหน้าเดิม ก็จะส่งวันและเวลาที่มีการโหลดหน้าเดิมไปให้กับ Server เพื่อให้ Server ตัดสินใจว่าจะส่ง Content มาให้ใหม่หรือไม่__
    - ในการตอบกลับของ server ครั้งที่ 2 มีการส่งไฟล์มาด้วยหรือไม่ สามารถอธิบายได้ว่าอย่างไร

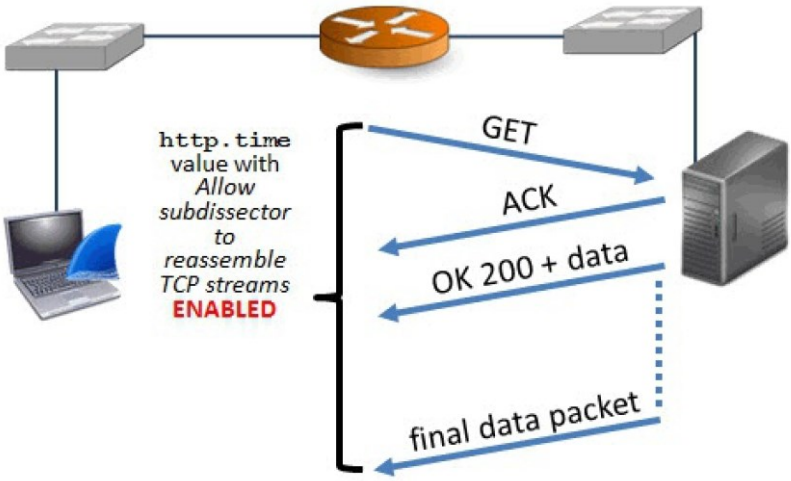
      __ไม่มีการส่งไฟล์ เนื่องจากนับจากเวลาใน IF-MODIFIED-SINCE ยังไม่มีการเปลี่ยนแปลง Content ใหม่ ยังเป็น Content เดิม เมื่อ Server ตรวจสอบพบจึงรู้ว่าข้อมูลเป็นข้อมูลเดิม จึงไม่มีการส่งมาใหม่__
11. ให้ไปที่ Edit | Preference… | Protocol | TCP ตามรูป



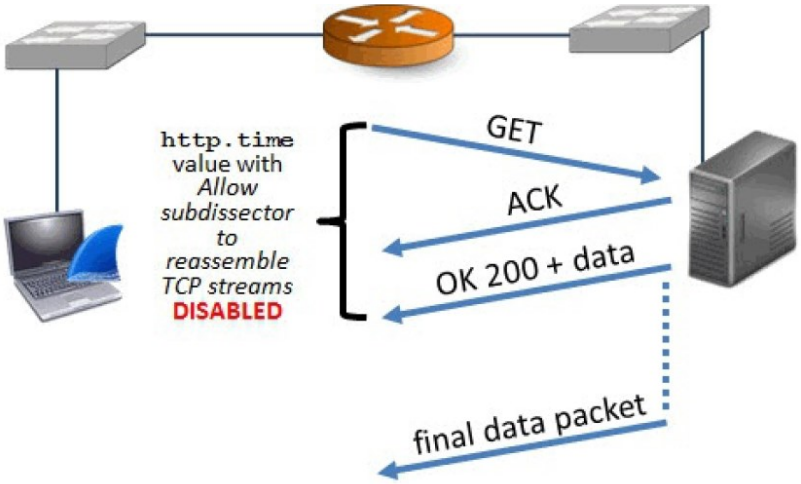
ให้แน่ใจว่า ไม่ติ๊กที่ **Allow subdissector to reassemble TCP streams**

12. ให้ทำตามข้อ 8 อีกครั้ง และเปิด Wireshark ใหม่แล้ว capture การเรียกหน้าเว็บเพจไปยัง url http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html จากนั้นให้หยุด capture
13. ให้ใช้ display filter : http เพื่อให้แสดงเฉพาะ HTTP (ถ้าทำถูกจะมี 5 บรรทัด) ซึ่งจะเห็นว่าหลังจากข้อมูล HTTP/1.1 200 OK แล้ว ยังมีข้อมูลตามมาอีก เนื่องจากไฟล์ html มีความยาวมาก (มากกว่า 4000 ไบต์) ทำให้ไม่สามารถส่งมาใน 1 packet ได้ จึงมีการแบ่งเป็นหลายๆ ส่วน (โดย TCP) ดังนั้นใน Wireshark จึงแสดงคำว่า Continuation ให้นักศึกษาตอบคำถามต่อไปนี้
    - มี HTTP GET กี่ครั้ง และมี packet ใดบ้างที่มี Status Code และเป็น Status Code ใด
      __มี HTTP GET 1 ครั้ง และ packet ที่2 ที่มี status code คือ Code 200 ok __
14. ให้ทำตามข้อ 8 อีกครั้ง และเปิด Wireshark ใหม่แล้ว capture การเรียกหน้าเว็บเพจไปยัง url http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html จากนั้นให้หยุด capture
    - ให้ใช้ display filter : http เพื่อให้แสดงเฉพาะ HTTP และให้ตอบคำถามต่อไปนี้
      มี HTTP GET กี่ครั้ง และไปยัง url ใดบ้าง
      __3 ครั้งจาก
      http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark file4.html.
      http://gaia.cs.umass.edu/pearson.png,
    - http://kurose.cslash.net/8E_cover_small.jpg __
    - ผู้เรียนคิดว่า ภาพทั้ง 2 ภาพในไฟล์ ถูกทำการ download ทีละไฟล์ (serialize) หรือถูก download ไปพร้อมๆ กัน (parallelize) ให้อธิบาย
      __ parallel ซึ่งถ้าดูจาก Wireshark จะเห็นว่ามี Request ในเวลาใกล้ๆ กัน และมี Response กลับมาไม่เป็นไปตามลำดับ (อาจต้องโหลดดูหลายๆ ครั้งจึงจะเห็นว่าบางครั้งภาพที่ 2 มาก่อนภาพแรก)__
    - ให้คลิ๊กขวาที่ Transmission Control Protocol | Protocol Preferences แล้วติ๊กที่ **Allow subdissector to reassemble TCP streams** เกิดอะไรขึ้น

__Wireshark จะไม่แสดงคำว่า Continue และจะรวมข้อมูลของ Response ใน Stream เดียวกันเป็น packet เดียว__



ค่า http.time เมื่อ Enable Allow subdissector to reassemble TCP streams



ค่า http.time เมื่อ Disable Allow subdissector to reassemble TCP streams

ในการตรวจสอบความล่าช้าในการทำงานของ Web Server เราจะใช้ค่า RTT (Round Trip Time) ซึ่งเป็นค่าเวลาตั้งแต่ GET จนถึงตอบกลับ (OK 200) ซึ่งจะบอกได้ถึงการตอบสนองต่อการเรียกใช้ของ Web Server ตัวนั้น ซึ่งสำหรับ Wireshark จะมีผลกระทบจาก การกำหนดค่า **Allow subdissector to reassemble TCP streams** ตามรูป คือ หาก disable จะคิดเฉพาะ packet HTTP OK 200 แต่ถ้า Enable ก็จะเป็นเวลาที่นับรวมถึงการโหลดข้อมูลทั้งหมด ดังนั้นให้ disable **Allow subdissector to reassemble TCP streams** ก่อน

15. ให้ไปที่ บรรทัดที่เป็น 200 OK แล้วไปที่ Hypertext Transfer Protocol แล้วขยาย subtrees ออกมาทั้งหมด แล้วไปที่บรรทัด **Time since request** แล้วเลือก **Apply as Column** ให้ตั้งชื่อว่า HTTP Delta จากนั้นให้ sort เพื่อหา packet ที่มีเวลา HTTP Delta มากที่สุด

16. ให้นักศึกษาตรวจสอบ RTT ของ 3 เว็บดังนี้ 1) http://example.com/ 2) http://www.http2demo.io/ 3) http://www.vulnweb.com/ และเว็บอื่นอีก 1 เว็บ (ผู้เรียนเลือกเอง) ให้บอกว่าค่า RTT ของแต่ละเว็บมีค่าใด ให้เรียงลำดับน้อยไปมาก ให้นักศึกษาแสดงขั้นตอนการทำงาน (เขียนอธิบายย่อๆ และบันทึก screenshot ประกอบ) และเปรียบเทียบค่ากับเพื่อนอีก 1 คน ว่าลำดับเหมือนกันหรือไม่ อย่างไร
__ข้อมูลของแต่ละคนจะไม่เท่ากันขึ้นกับปัจจัยหลายอย่าง เช่น ความช้าเร็ว ใกล้ไกล__