JARINGAN KOMPUTER – TUGAS PENDAHULUAN MODUL
TCP (TRANSMISSION CONTROL PROTOCOL)

Nama : Muhammad Hamzah Haifan Ma'ruf

NIM : 2311102091

Kelas : S1IF-11-07

Kode Dosen : AIZ

JAWABAN MODUL

A. Teori

1. Saat ini anda seharusnya sudah mengetahui apa itu OSI Model dan TCP/IP Model. Tuliskan dan jelaskan hubungan layer pada OSI Model dengan layer pada TCP/IP Model (contoh Transport Layer pada OSI Model setara dengan Transport Layer pada TCP/IP Layer) buatkan secara visual. Serta sebutkan protokol HTTP, DNS, UDP dan TCP berada pada layer mana untuk setiap model

OSI Model	TCP/IP Model	Contoh Protokol
7. Application	4. Application	HTTP, DNS
6. Presentation	4. Application	(ditangani aplikasi)
5. Session	4. Application	(ditangani aplikasi)
4. Transport	3. Transport	TCP, UDP
3. Network	2. Internet	IP, ICMP
2. Data Link	1. Network Access	Ethernet, Wi-Fi
1. Physical	1. Network Access	Kabel, sinyal fisik

Application Layer (OSI & TCP/IP)

- Fungsinya menangani interaksi dengan aplikasi dan layanan jaringan.
- Protokol: HTTP, DNS

Transport Layer (OSI & TCP/IP)

- Bertanggung jawab mengatur koneksi end-to-end dan memastikan data sampai dengan benar.
- Protokol: TCP (koneksi andal), UDP (koneksi cepat tanpa jaminan).

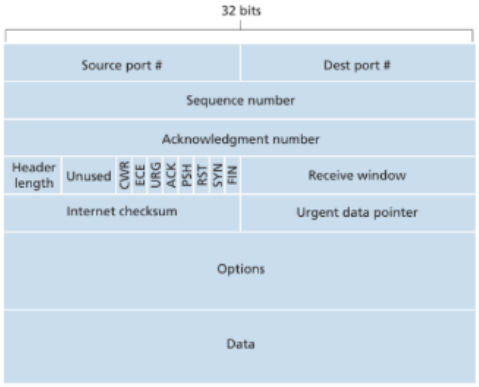
Network Layer (OSI) & Internet Layer (TCP/IP)

- Mengelola pengalamatan dan routing antar jaringan.
- Protokol: IP, ICMP (ping).

Data Link & Physical (OSI) & Network Access (TCP/IP)

- Mengurus koneksi fisik, frame data, dan transfer bit antar perangkat.
- Protokol: Ethernet, Wi-Fi.

2. Perhatikan gambar dibawah ini:Berdasarkan gambar tersebut jawablah pertanyaan dibawah ini:



a. Apa yang dimaksud dari gambar diatas?

Gambar tersebut adalah header dari segmen TCP (Transmission Control Protocol). Ini adalah struktur data yang digunakan saat mentransfer data melalui jaringan dengan protokol TCP di Transport Layer (OSI & TCP/IP). Header ini berisi berbagai informasi penting seperti port, urutan data, konfirmasi penerimaan, flag, ukuran window, dan lainnya yang membantu memastikan data terkirim dengan benar dan berurutan.

b. Jelaskan fungsi dari Source port dan Destination port dalam gambar tersebut! Sebutkan port-port yang digunakan untuk protocol HTTP, HTTPS, UDP dan TCP

- Source Port: Menandakan port asal dari pengirim agar penerima tahu dari aplikasi mana data datang.
- Destination Port: Menentukan port tujuan di perangkat penerima agar data sampai ke aplikasi yang tepat.

Port yang sering digunakan:

- HTTP: Port 80
- HTTPS: Port 443
- UDP: Bergantung pada layanan, contoh 53 untuk DNS
- TCP: Fleksibel, contoh 21 (FTP), 22 (SSH), 80 (HTTP)

c. Apa yang dimaksud dengan sequence number dan acknowledgment number? Bagaimana keduanya digunakan dalam transfer data? Apa yang terjadi jika sequence number bernilai 1000 dan acknowledgement number bernilai 2000?

- Sequence Number: Nomor urut byte pertama dalam segmen data saat ini. Berguna untuk menjaga urutan data. Contoh: Jika Sequence Number = 1000, artinya data saat ini dimulai dari byte ke-1000.
- Acknowledgment Number: Memberi tahu pengirim bahwa data sampai dengan baik. Angka ini berisi byte selanjutnya yang diharapkan penerima. Contoh: Jika Acknowledgment Number = 2000, artinya penerima sudah menerima hingga byte ke-1999, dan menunggu byte ke-2000.

Jika Sequence Number = 1000 dan Acknowledgment Number = 2000, artinya:

- Pengirim mengirim data dari byte ke-1000.
- Penerima mengakui (acknowledge) bahwa dia sudah menerima sampai byte 1999 dan menunggu byte ke-2000.

Kalau data rusak/hilang, tidak ada acknowledgment, dan pengirim akan retransmit (kirim ulang).

d. Jelaskan peran flags (SYN, ACK, FIN, RST, PSH, URG, ECE, CWR) pada gambar tersebut! Berikan contoh penggunaan dari flags tersebut!

Header ini berisi informasi penting seperti port, nomor urut data, konfirmasi penerimaan, flag, dan kontrol lainnya agar data terkirim dengan benar dan berurutan.

Source Port menunjukkan port asal pengirim, sedangkan Destination Port menandakan port tujuan di penerima. Contoh port yang sering digunakan adalah 80 untuk HTTP, 443 untuk HTTPS, dan 53 untuk UDP (DNS). Sequence Number menandakan byte pertama dalam segmen data, sementara Acknowledgment Number memberi tahu pengirim bahwa data sudah diterima hingga byte tertentu.

Misalnya, jika Sequence Number = 1000 dan Acknowledgment Number = 2000, artinya penerima menunggu byte ke-2000 selanjutnya.

Flags bertindak sebagai kontrol koneksi, seperti SYN untuk memulai koneksi, ACK untuk mengonfirmasi data, FIN menutup koneksi, dan RST mereset koneksi. Ada juga PSH untuk mengirim data langsung ke aplikasi, URG menandakan data mendesak, serta ECE dan CWR untuk kontrol kemacetan. Window Size menentukan berapa banyak data yang bisa dikirim tanpa menunggu konfirmasi. Checksum berfungsi mendeteksi error pada data, sedangkan Urgent Pointer menandai data yang harus segera diproses.

e. Apa fungsi Window Size dari gambar tersebut? Serta jelaskan perbedaan antara checksum dan urgent pointer dalam gambar tersebut!

- Window Size

  Menentukan jumlah data (dalam byte) yang bisa dikirim tanpa harus menunggu acknowledgment. Semakin besar window, semakin cepat transfernya, tapi juga lebih berisiko kalau ada data rusak.

- Checksum

  Digunakan untuk mendeteksi error dalam data yang dikirim. Pengirim membuat checksum, penerima menghitung ulang dan membandingkan. Kalau beda, data dianggap rusak dan diminta ulang.

- Urgent Pointer

  Menunjukkan bahwa bagian tertentu dari data bersifat mendesak (urgent). Biasanya dipakai dengan URG flag untuk memberitahu aplikasi agar segera memproses data tersebut (misalnya dalam sesi Telnet saat tekan Ctrl+C).

3. Jelaskan apa yang dimaksud dengan:

   a. TCP Flow Control dan TCP Congestion Control

   TCP Flow Control menghindari pengirim mengirim data lebih cepat daripada yang bisa diproses penerima. Ini dilakukan dengan Window Size yang memberi tahu pengirim seberapa banyak data yang bisa dikirim tanpa membebani penerima. TCP Congestion Control mencegah kemacetan jaringan. TCP menyesuaikan kecepatan

pengiriman berdasarkan kondisi jaringan dengan algoritma seperti Slow Start, Congestion Avoidance, Fast Retransmit, dan Fast Recovery.

b. RTT, sampleRTT, estimatedRTT dan devRTT

- RTT (Round Trip Time): Waktu yang dibutuhkan data untuk pergi dari pengirim ke penerima dan kembali lagi.
- sampleRTT: RTT aktual yang diukur untuk setiap segmen tertentu.
- estimatedRTT: RTT yang diperkirakan dengan merata-ratakan beberapa sampleRTT sebelumnya.
- devRTT: Perbedaan antara sampleRTT dan estimatedRTT, digunakan untuk menghitung seberapa besar variasi delay.

c. TimeoutInterval

TimeoutInterval adalah waktu tunggu maksimal sebelum pengirim menganggap paket hilang dan mengirim ulang. Ini dihitung dari estimatedRTT + 4 * devRTT agar responsif tapi tidak terlalu cepat timeout.

d. TCP Throughput dan avg TCP Throughpu

- TCP Throughput: Jumlah data yang berhasil ditransfer per detik.
- avg TCP Throughput: Rata-rata throughput selama koneksi berlangsung. Dipengaruhi oleh RTT, window size, dan congestion control.

4. Buatkan ilustrasi dan jelaskan perbedaan antara:

a. 2-way handshake scenarios

Proses koneksi dengan dua langkah:

- Client → Server: SYN (Permintaan koneksi)
- Server → Client: ACK (Menyetujui koneksi)

Kekurangan tidak memastikan server siap menerima data, rentan serangan Half-Open Connection.

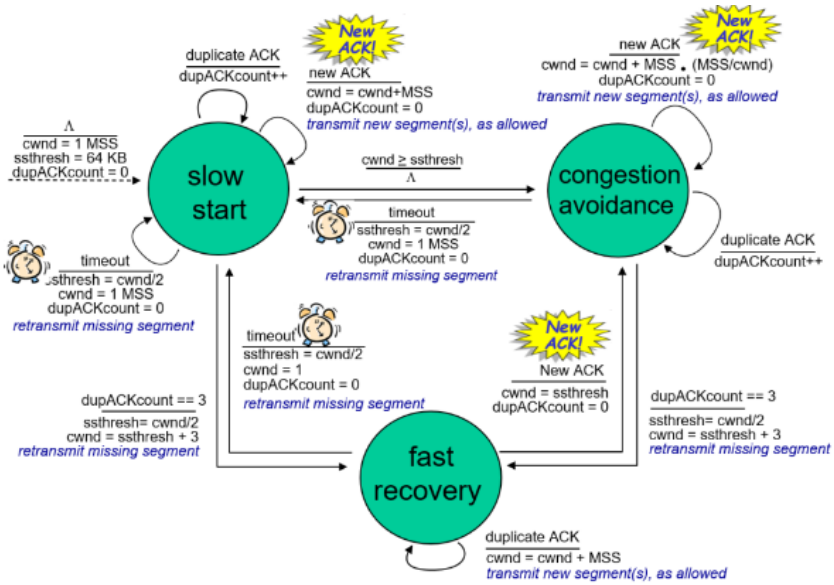
b. 3-way handshake scenarios

Proses koneksi dengan tiga langkah yang lebih andal:

- Client → Server: SYN (Permintaan koneksi)
- Server → Client: SYN-ACK (Menyetujui dan membuka koneksi)
- Client → Server: ACK (Konfirmasi akhir)

Lebih aman dan memastikan kedua sisi siap bertukar data.

5. Perhatikan gambar berikut yang merupakan berkaitan TCP Congestion Control: Berdasarkan alur pada gambar di atas, jawablah pertanyaan dibawah ini



a. Jelaskan fase slow start dalam mekanisme congestion control TCP seperti yang ditunjukkan pada gambar! Apa tujuan dari fase ini dan bagaimana TCP mengatur laju pengiriman data selama fase ini?

Fase Slow Start bertujuan untuk perlahan-lahan meningkatkan laju pengiriman data agar tidak langsung membebani jaringan. Awalnya, ukuran Congestion Window (cwnd) dimulai dari 1 MSS (Maximum Segment Size). Setiap kali pengirim menerima ACK baru, ukuran cwnd bertambah 2 kali lipat (eksponensial), yaitu cwnd = cwnd + MSS. Fase ini berlanjut hingga mencapai nilai ssthresh (slow start threshold) atau terjadi packet loss.

b. Bagaimana TCP beralih daro fase slow start ke fase congestion avoidance? Apa kondisi yang menyebabkan transisi ini, dan bagaimana perilaku TCP berubah setelah transisi tersebut?

   TCP beralih dari Slow Start ke Congestion Avoidance saat cwnd mencapai ssthresh. Setelah itu, pertumbuhan laju pengiriman berubah dari eksponensial menjadi linear (lebih lambat dan terkendali). Di fase ini, cwnd bertambah MSS/cwnd setiap kali menerima ACK baru untuk menghindari kemacetan jaringan.

c. Jelaskan mekanisme fast recovery dalam TCP seperti yang ditunjukkan pada gambar! Apa tujuan tujuan dari fast recovery dan bagaimana TCP merespons packet loss selama fase ini?

   Fast Recovery bertujuan mempercepat pemulihan setelah mendeteksi packet loss tanpa harus kembali ke Slow Start. Ketika pengirim menerima 3 duplicate ACK (ACK yang sama diulang 3 kali), TCP menganggap segmen hilang. ssthresh disetel ke cwnd/2, lalu cwnd dinaikkan sebesar ssthresh + 3 untuk menjaga aliran data. Pengirim langsung mengirim ulang segmen yang hilang tanpa menunggu Timeout, dan saat ACK baru datang, cwnd bertambah 1 MSS (linear growth) dan kembali ke Congestion Avoidance.

d. Apa peran "New ACK!" dalam mekanisme congestion control TCP seperti yang ditunjukkan pada gambar? Bagaimana TCP menggunakan informasi dari ACK untuk mengatur laju pengiriman data dalam berbagai fase?

   "New ACK!" menandakan penerima berhasil menerima segmen baru. Setiap New ACK yang diterima di Slow Start melipatgandakan cwnd (eksponensial), di Congestion Avoidance menambah cwnd secara linear, dan di Fast Recovery membantu memastikan pemulihan berjalan baik. TCP mengandalkan ACK untuk menyesuaikan laju pengiriman: semakin banyak ACK baru diterima, semakin besar cwnd, sehingga throughput meningkat.

B. TCP Performance Analysis

Sekarang anda memahami rumus-rumus yang akan digunakan menilai suatu performa TCP, jawablah pertanyaan berikut:

1. Seorang administrator jaringan ingin menganalisis performa TCP pada koneksi antara dua host (Client dan Server). Data diambil menggunakan Wireshark selama

proses transfer file berukuran 10 MB melalui protokol TCP. Tujuannya adalah untuk:

a. Menghitung nilai RTT , EstimatedRTT , dan TimeoutInterval

RTT (Round Trip Time)
RTT didapat dari:

$$RTT = WaktuACKDiterima - WaktuPaketDikirim$$

Dari tabel:

- RTT1 = 0.050 ms
- RTT2 = 0.051 ms
- RTT3 = 0.055 ms
- RTT4 = 0.052 ms

EstimatedRTT
Rumus:

$$EstimatedRTT = (1-\alpha) \times EstimatedRTTsebelumnya + \alpha \times SampleRTT$$

Diketahui **$\alpha = 0.125$**:

- EstimatedRTT pertama = 0.050 ms (sama dengan SampleRTT pertama)
- EstimatedRTT kedua = $(1-0.125) \times 0.050 + 0.125 \times 0.051 = 0.050125$ms
- EstimatedRTT ketiga = $(1-0.125) \times 0.050125 + 0.125 \times 0.055 = 0.0507$ms
- EstimatedRTT keempat = $(1-0.125) \times 0.0507 + 0.125 \times 0.052 = 0.0508$ms

DevRTT
Rumus:

$$DevRTT = (1-\beta) \times DevRTTsebelumnya + \beta \times |SampleRTT - EstimatedRTT|$$

Dengan **$\beta = 0.25$**:

- DevRTT pertama = 0 (karena belum ada perbandingan)
- DevRTT kedua = $(1-0.25)\times 0+0.25\times|0.051-0.050|=0.00025$ms
- DevRTT ketiga = $(1-0.25)\times 0.00025+0.25\times|0.055-0.0507|=0.0013$ms
- DevRTT keempat = $(1-0.25)\times 0.0013+0.25\times|0.052-0.0508|=0.0012$ms

TimeoutInterval
Rumus:

$$TimeoutInterval=EstimatedRTT+4\times DevRTT$$

TimeoutInterval terakhir:

$$TimeoutInterval=0.0508+4\times 0.0012=0.0556ms$$

b. Mengestimasi TCP Throughput.

Rumus throughput:

$$TCP\ Throughput=Window\ Size/RTT$$

Diketahui:

- Window Size = 65,536 bytes (64 KB)
- RTT rata-rata = (0.050 + 0.051 + 0.055 + 0.052) / 4 = 0.052 ms

Maka:

$$TCP\ Throughput=65{,}536\times 8/0.052=10{,}086{,}461\ bps\approx 10\ Mbps$$

c. Menilai apakah koneksi memiliki performa yang optimal.

- RTT stabil → Iya, karena nilainya konsisten di sekitar 0.05 ms.
- Throughput tinggi → Iya, karena mencapai 10 Mbps, sesuai ekspektasi koneksi cepat.
- TimeoutInterval wajar → Iya, karena 0.0556 ms cukup responsif dan tidak terlalu cepat atau lambat.

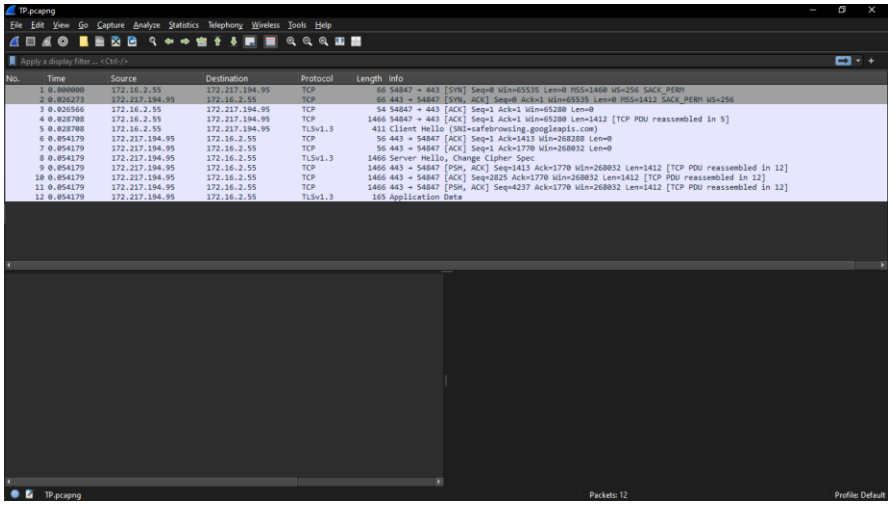
Dengan data yang dikumpulkan setelah melakukan capture pada Wireshark adalah sebagai berikut:

Informasi tambahan:

No	Waktu Paket Dikirm (s)	Waktu ACK Diterima (s)	SampleRTT (ms)
1	0,000	0,050	50
2	0,051	0,102	51
3	0,103	0,155	52
4	0,156	0,209	53

- Ukuran file yang ditransfer: 10 MB
- Total waktu transfer: 2 detik .
- Window size TCP: 64 KB = 65,536 byte.
- Faktor bobot (α): 0.125 dan Faktor bobot (β): 0.25

2. Saat ini seharusnya anda sudah paham bagaimana perhitungan berkaitan performansi TCP. Sekarang buka file TP.pcapng disana terdapat 12 paket. Lakukan hal perhitungan yang mirip dengan nomor 1 untuk menjawab pertanyaan berikut (pastikan time sudah dalam format Times of Days):



a. Menghitung nilai RTT, EstimatedRTT, dan TimeoutInterval.

Paket 1 (SYN) dikirim pada waktu 0.000000

Paket 2 (SYN, ACK) diterima pada waktu 0.022673

RTT = 0.022673 - 0.000000 = 0.022673 detik (22.673 ms)

$$EstimatedRTT=(1-0.125)\times EstimatedRTTsebelumnya+0.125\times SampleRTT$$

Kita pakai SampleRTT = 22.673 ms:

$$EstimatedRTT=(0.875\times 0)+(0.125\times 22.673)=2.834ms$$

Lalu hitung **DevRTT**:

$$DevRTT=(1-0.25)\times DevRTTsebelumnya+0.25\times|SampleRTT-EstimatedRTT|$$

$$DevRTT=(0.75\times 0)+0.25\times|22.673-2.834|=4.96ms$$

$$TimeoutInterval=EstimatedRTT+4\times DevRTT$$

$$TimeoutInterval=2.834+4\times 4.96=22.67ms$$

Hint:

- Gunakan nilai RTT eksplisit yang tercatat dalam packet content, misalnya [the RTT to ACK the segment was: ….]
- Untuk menhitung EstimatedRTT dan DevRTT menggunakan nilai faktor bobot a = 0.125 dan β = 0.25
- Hanya terdapat dua sampleRTT

b. Mengestimasi TCP Throughput.

$$Throughput=RTTWindow/Size$$

Dari paket Wireshark, kita lihat Window Size = 268032 bytes, RTT = 22.673 ms:

$$Throughput=268032/0.022673=11.82\ MB/s$$

Hint:

- Windows Size: menentukan nilai yang akan digunakan pada windows size memiliki tiga opsti sebagai berikut:

  - Rata-Rata: rata-rata dari nilai [calculated window size: …]pada packet content yang berbeda
  - Minimum: nilai terkecil dari nilai [calculated window size: …]pada packet content yang berbeda
  - Maksimum: nilai terbesar dari nilai [calculated window size: …] pada packet content yang berbeda

- Ukuran file yang Ditransfer: Total penjumlahan tcp.len (TCP payload) dari semua packet. Gunakan filter tcp.len > 0 di Wireshark untuk mempermudah
- MSS: Nilai terkecil dari Maximum segment size yang ditemukan pada handshake TCP
- Probabilitas Kehilangan paket (p): menggunakan internet umum

c. Menilai apakah koneksi memiliki performa yang optimal.

a. Window Size vs RTT

Window Size yang besar (268032 bytes) menunjukkan koneksi mencoba mengirim data dalam jumlah besar per RTT. RTT yang stabil di 22.673 ms termasuk rendah — artinya koneksi cukup responsif. Koneksi sejauh ini terlihat efisien, karena kombinasi window size besar dan RTT kecil mendukung throughput tinggi.

b. Loss Detection

Tidak ada Retransmission atau Duplicate ACK yang muncul di tangkapan paket, yang menandakan tidak ada paket loss. Koneksi stabil dan tidak mengalami kemacetan.

- Bandingkan throughput untuk masing-masing opsi window size (rata-rata, minimum, maksimum).
- Evaluasi apakah nilai RTT dan window size mendukung efisiensi transfer data.

C. TCP Cognition Control

Jawab pertanyaan dibawah ini serta lakukan setiap langkah dibawah ini dan berikan screenshot-nya:

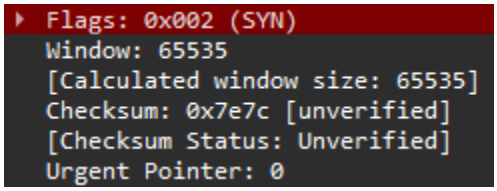
Sebagai seorang administrator jaringan, Anda diminta untuk menganalisis komunikasi TCP antara Host A (172.16.2.55) sebagai client dan Server B (23.33.184.237) sebagai server menggunakan Wireshark. Data capture mencakup komunikasi TCP melalui port 80 (HTTP). Fokus analisis adalah memahami bagaimana mekanisme TCP Congestion Control bekerja dalam komunikasi ini, serta memvisualisasikan aliran data menggunakan Time-Sequence Graph (Stevens) dari sudut pandang Client-to-Server dan Server-to-Client. Data capture didapatkan pada file TP2.pcapng

Lakukan Instruksi dan Jawab pertanyaannya disertai screenshot:

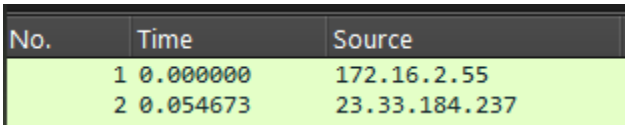
1. Identifkasi Parameter Awal TCP

   a. Temukan paket SYN pertama yang digunakan untuk memulai koneksi TCP! Berapa nomor paketnya?

      Nomor paket SYN pertama adalah 1.



   b. Catat nilai MSS yang dikirim oleh Client dan Server



      MSS Client (172.16.2.55): 1460 bytes

      MSS Server (23.33.184.237): 1460 bytes

   c. Apakah ada perbedaan nilai MSS? Jika iya jelaskan kenapa bisa terjadi!

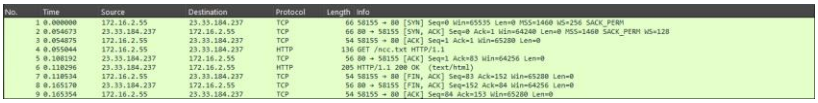
      Tidak ada perbedaan MSS antara client dan server.

      Penjelasan:

MSS (Maximum Segment Size) sama-sama 1460 bytes, yang merupakan ukuran umum di jaringan Ethernet (MTU 1500 - 40 bytes header TCP/IP).

Perbedaan MSS bisa terjadi kalau salah satu perangkat punya jaringan dengan MTU lebih kecil (misal, koneksi mobile atau VPN).

2. Analisis Congestion Contro



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.2.55	23.33.184.237	TCP	66	58155 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.054673	23.33.184.237	172.16.2.55	TCP	66	80 → 58155 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
3	0.054875	172.16.2.55	23.33.184.237	TCP	54	58155 → 80 [ACK] Seq=1 Ack=1 Win=65280 Len=0
4	0.055044	172.16.2.55	23.33.184.237	HTTP	136	GET /esc.txt HTTP/1.1
5	0.108192	23.33.184.237	172.16.2.55	TCP	56	80 → 58155 [ACK] Seq=1 Ack=83 Win=64256 Len=0
6	0.118296	23.33.184.237	172.16.2.55	HTTP	205	HTTP/1.1 200 OK (text/html)
7	0.118534	172.16.2.55	23.33.184.237	TCP	54	58155 → 80 [FIN, ACK] Seq=83 Ack=152 Win=65280 Len=0
8	0.165170	23.33.184.237	172.16.2.55	TCP	56	80 → 58155 [FIN, ACK] Seq=152 Ack=84 Win=64256 Len=0
9	0.165354	172.16.2.55	23.33.184.237	TCP	54	58155 → 80 [ACK] Seq=84 Ack=153 Win=65280 Len=0

a. Identifikasi apakah ada indikasi packet loss (misalnya, duplikasi ACK atau retransmisi) baik dari sisi client maupun server!

   Dari capture, tidak ada indikasi packet loss seperti:

   Duplicate ACK

   Retransmission

   Koneksi berjalan lancar dengan ACK normal di setiap paket.

b. Jika terjadi packet loss, bagaimana TCP merespons? Apakah Client mengurangi ukuran congestion window (cwnd)? Jelaskan proses recovery yang dilakukan!

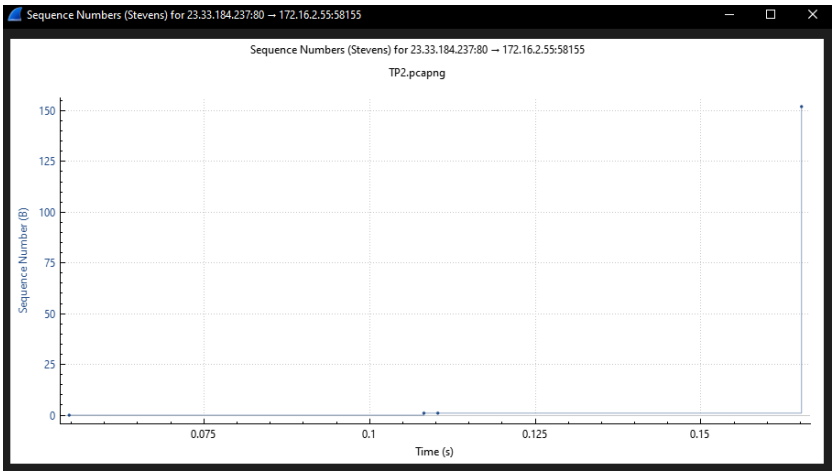
   Karena tidak ada loss di capture ini, TCP tidak melakukan mekanisme recovery seperti Fast Retransmit atau Congestion Window Reduction (cwnd).

c. Temukan pola Slow Start, Congestion Avoidance dan Fast Retransmit pada file tersebut (jika ada), Jika tidak ada, jelaskan kenapa hal tersebut tidak dapat terjadi!

   - Slow Start: Tidak terlihat eksplisit, karena jumlah paket kecil dan koneksi langsung stabil.
   - Congestion Avoidance: Koneksi berjalan langsung stabil tanpa penurunan throughput.
   - Fast Retransmit: Tidak ada karena tidak terjadi packet loss.

3. Visualisasi Time-Sequence Graph (Stevens)

   Untuk memunculkan Time-Sequence Graph (Stevens) pada file TP2.pcapng, klik file TCP paling awal lalu ikuti seperti gambar berikut:



a. Pada dialog box memiliki fitur jelaskan apa yang dimaksud dengan Stream dan Switch Direction

- Stream adalah satu sesi komunikasi TCP spesifik antara client dan server. Dalam kasus ini, stream merepresentasikan data dari 172.16.2.55 (client) ke 23.33.184.237 (server) dan sebaliknya.
- Switch Direction membalik arah grafik, jadi kalau semula dari Client → Server, akan berbalik menjadi Server → Client. Berguna untuk menganalisis kedua arah komunikasi.

b. Tampilkan Time-Sequence Graph (Stevens) untuk Client to Server, lalu amati pola aliran data pada grafik tersebut lalu jawablah:

- Slow Start: Tidak terlihat jelas karena koneksi kecil dan pendek. Biasanya terlihat sebagai lonjakan eksponensial, tapi di sini hanya beberapa paket kecil.
- Congestion Avoidance: Tidak terlihat karena koneksi selesai sebelum fase ini tercapai.
- Anomali: Tidak ada retransmisi atau stagnasi sequence number. Koneksi terlihat bersih.

c. Tampilkan Time-Sequence Graph (Stevens) untuk Server to Client, lalu amati pola aliran data pada grafik tersebut lalu jawablah:

- Slow Start: Tidak terlihat, hanya beberapa paket kecil.
- Congestion Avoidance: Tidak muncul karena durasi koneksi sangat singkat.
- Anomali: Tidak ada retransmisi atau stagnasi sequence number, data mengalir lancar.

d. Bandingkan pola aliran data antara aktivitas pada poin b dan poin c. Apakah ada perbedaan signifikan dalam jumlah data yang dikirim? Apakah ada perbedaan dalam kecepatan pengiriman data? Jelaskan kenapa hal tersebut dapat terjadi

- Jumlah data server mengirim lebih banyak data dibanding client. Ini wajar karena client hanya mengirim request kecil, sedangkan server membalas dengan data lebih besar (response HTTP).
- Kecepatan pengiriman server lebih cepat karena data HTTP langsung dikirim dalam beberapa paket besar.

4. Kesimpulan

a. Berdasarkan analisis TCP Congestion Control dan visualisasi Time Sequence Graph dari sudut pandang client-to-server dan server-to-client , apakah mekanisme TCP berjalan optimal dalam komunikasi ini?

Iya, cukup optimal

- Tidak ada retransmisi atau packet loss.
- Koneksi singkat dan cepat, tanpa indikasi kemacetan.
- Flow data terlihat stabil meskipun singkat.

b. Jika tidak, identifikasi masalah utama dan berikan rekomendasi untuk meningkatkan performa

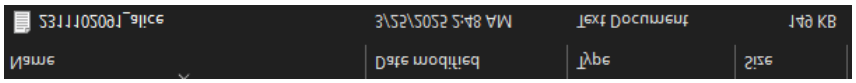
Walaupun berjalan lancar, ada beberapa rekomendasi untuk koneksi lebih besar atau kompleks

- Aktifkan TCP Window Scaling: Supaya koneksi besar lebih efisien.
- Gunakan HTTP/2 atau HTTP/3: Protokol ini lebih baik dalam menangani banyak request sekaligus.
- Monitoring jaringan: Kalau jumlah paket besar, perhatikan indikator packet loss atau latensi tinggi.

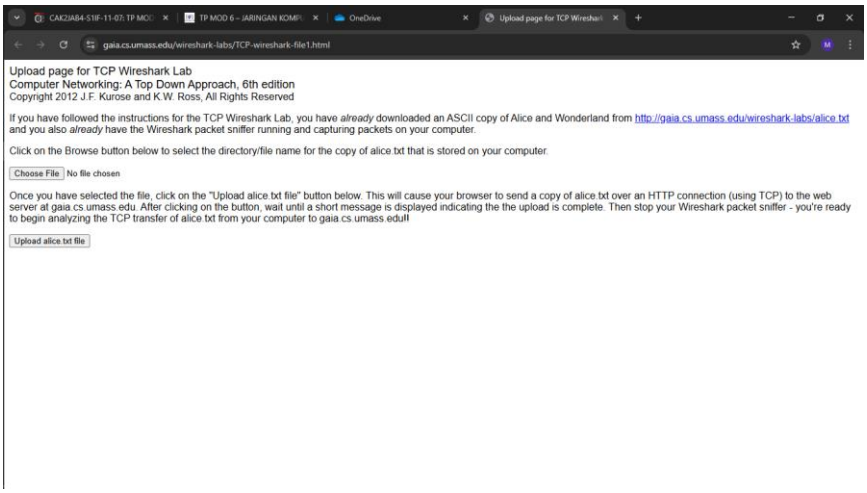
D. Wireshark

Lakukan setiap langkah dibawah ini dan berikan screenshot-nya:

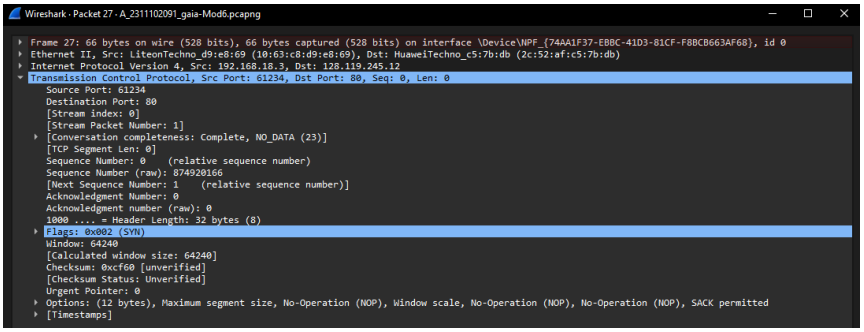
1. Buka browser. Buka http://gaia.cs.umass.edu/wireshark-labs/alice.txt dan copy ASCII dari Alice in Wonderland. Simpan ini sebagai file .txt di suatu tempat di komputer/laptop dengan format nama “nim_file.txt”.

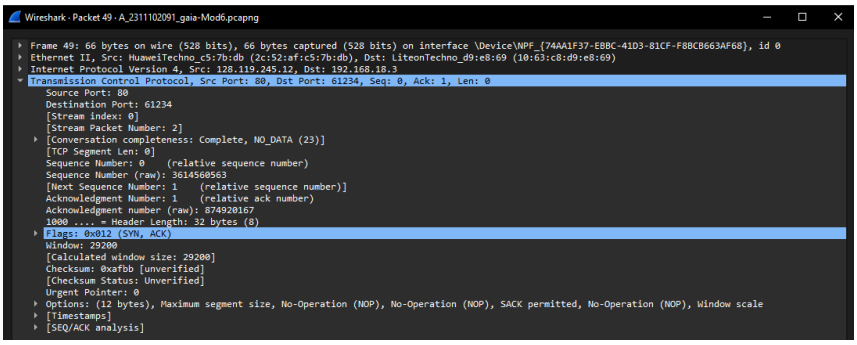


2. Selanjutnya buka http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark file1.html



3. Anda akan melihat layar seperti gambar berikut
4. Gunakan tombol browse dalam formulir ini ( ) ke file di komputer anda yang baru saja anda buat yang berisi Alice in Wonderland. Jangan dulu tekan tombol “Upload file alice.txt” ( ).
5. Sekarang jalankan Wireshark dan mulai capture paket!
6. Kembali ke browser, tekan tombol “Upload file alice.txt” untuk mengupload file, pesan singkat akan ditampilkan di jendela browser anda!
7. Hentikan capture paket Wireshark
8. Save dengan format nama A_NIM_gaia-Mod6.pcapng (contoh:A_130*******_gaia-Mod6.pcapng)
9. Tunjukkan TCP pada paket request dan response dari hasil tangkapan anda!





Berdasarkan hasil pekerjaan anda yang dilakukan di atas jawablah pertanyaan

berikut!

1. Bagaimana cara mengidentifikasi paket TCP dalam hasil tangkapan Wireshark? Apa perbedaan antara paket TCP dan protokol lainnya (misalnya UDP) dalam hasil tangkapan? Tunjukkan baris pada Wireshark yang menunjukkan paket TCP berdasarkan aktivitas upload file Alice in Wonderland. Apakah ada indikasi bahwa koneksi TCP ditutup setelah proses upload selesai? Jelaskan.

Untuk mengidentifikasi paket TCP di Wireshark, kita bisa melihat kolom "Protocol" yang menampilkan "TCP" atau menggunakan filter "tcp" agar hanya paket TCP yang muncul. Perbedaan utama antara TCP dan UDP terletak pada cara pengiriman data. TCP bersifat connection-oriented, yang berarti harus melalui proses handshake dan memiliki mekanisme kontrol aliran serta error checking, sedangkan UDP lebih ringan dan cepat karena connectionless, namun tidak menjamin data sampai dengan urutan yang benar. Pada hasil tangkapan Wireshark, aktivitas upload file Alice in Wonderland ditandai dengan paket TCP yang memiliki flag "PSH" (Push), seperti yang terlihat pada paket nomor 51 dan seterusnya. Indikasi bahwa koneksi TCP ditutup terlihat dari paket yang memiliki flag "FIN" atau "RST" di akhir sesi, yang menunjukkan bahwa salah satu pihak (client atau server) meminta koneksi diakhiri.

2. Jelaskan struktur header TCP dan identifikasi field penting yang berkaitan dengan aktivitas upload file. Bagaimana mekanisme kontrol kesalahan (error control) dalam TCP diimplementasikan melalui header TCP? Apakah ada indikasi penggunaan checksum dalam header TCP? Jelaskan kegunaannya.

Struktur header TCP memiliki beberapa field penting yang mendukung aktivitas upload file. Di antaranya adalah Source dan Destination Port yang menentukan aplikasi tujuan, Sequence Number untuk menandai urutan data, Acknowledgment Number sebagai konfirmasi penerimaan, Flags seperti SYN, ACK, dan FIN untuk mengontrol koneksi, Window Size untuk pengaturan aliran data, serta Checksum untuk mendeteksi kerusakan data. Mekanisme kontrol kesalahan dalam TCP

diimplementasikan melalui checksum ini. Setiap paket TCP membawa checksum yang akan diverifikasi di sisi penerima. Jika checksum tidak sesuai dengan data yang diterima, paket dianggap rusak dan dibuang, lalu pengirim akan melakukan retransmission. Checksum ini penting untuk memastikan integritas data dan mencegah kerusakan informasi saat pengiriman berlangsung.

3. Bagaimana data besar (file Alice in Wonderland) dikirim melalui TCP segment?Apakah data file dikirim dalam satu segment atau dibagi menjadi beberapa segment? Jelaskan. Tunjukkan contoh segment TCP dalam hasil tangkapan Wireshark dan jelaskan isi payload-nya. Bagaimana TCP segment menangani masalah seperti packet loss atau delay? Apakah ada indikasi adanya retransmission dalam hasil tangkapan Anda? Jika ya, jelaskan penyebabnya. Apa yang terjadi jika salah satu segment gagal diterima oleh penerima? Bagaimana TCP memastikan integritas data?

Data besar seperti file Alice in Wonderland tidak dikirim dalam satu segment utuh, melainkan dipecah menjadi beberapa segment kecil sesuai dengan Maximum Segment Size (MSS). Di hasil tangkapan Wireshark, kita bisa melihat adanya "PDU reassembled" yang menandakan beberapa segment bergabung menjadi satu data besar. TCP memiliki mekanisme untuk menangani packet loss dan delay. Jika ada segment yang hilang atau terlalu lama sampai, pengirim akan mendeteksi hal ini lewat Acknowledgment Number yang tidak berubah, kemudian melakukan retransmission. Jika penerima tidak mendapatkan segment tertentu, ia juga akan mengirim ulang ACK untuk segment terakhir yang diterima dengan benar (DUP ACK), memicu pengirim untuk segera mengirim ulang segment yang hilang. Selain itu, TCP memastikan integritas data dengan memaksa pengirim mengulang segment yang gagal sampai, sehingga data yang diterima selalu lengkap dan sesuai urutan.

4. Jelaskan proses three-way handshake yang terjadi sebelum pengiriman data dimulai. Tunjukkan paket-paket yang membentuk three-way handshake dalam hasil tangkapan Wireshark. Apa peran flag SYN , SYN-ACK , dan ACK dalam proses ini? Jelaskan secara rinci. Bagaimana three-way handshake memastikan koneksi yang andal antara client dan server? Apakah ada indikasi bahwa koneksi TCP ditutup setelah proses upload selesai? Jelaskan bagaimana proses penutupan koneksi dilakukan

Sebelum pengiriman data dimulai, TCP menjalankan proses three-way handshake untuk memastikan koneksi stabil dan siap digunakan. Proses ini dimulai dengan client mengirim paket SYN sebagai permintaan koneksi ke server. Server kemudian membalas dengan SYN-ACK sebagai tanda persetujuan, dan client menutup proses dengan mengirim ACK sebagai konfirmasi akhir. Peran flag SYN adalah untuk memulai koneksi, SYN-ACK sebagai balasan dari server, dan ACK sebagai persetujuan akhir dari client bahwa koneksi berhasil dibuat. Handshake ini memastikan kedua belah pihak siap bertukar data dan menyepakati parameter koneksi seperti sequence number. Setelah data terkirim, koneksi ditutup dengan proses empat langkah (four-way handshake), di mana salah satu pihak mengirim FIN untuk memulai penutupan, pihak lain

membalas ACK, kemudian mengirim FIN kembali, lalu diakhiri dengan ACK terakhir. Jika koneksi terputus mendadak, Wireshark akan menunjukkan flag RST sebagai tanda reset koneksi.