

# MODUL PRAKTIKUM

# JARINGAN KOMPUTER

## S1 INFORMATIKA



## Lembar Pengesahan

Saya yang bertanda tangan di bawah ini:

Nama : Dr. Fazmah Arif Yulianto, S.T., M.T.  
NIK : 99750034  
Koordinator Mata Kuliah : Jaringan Komputer  
Prodi : S1 Informatika

Menerangkan dengan sesungguhnya bahwa modul ini digunakan untuk pelaksanaan praktikum di Semester Genap Tahun Ajaran 2022/2023 di Laboratorium Informatika Fakultas Informatika Universitas Telkom.

Bandung, 20 Februari 2023



Mengesahkan,  
Koordinator Mata Kuliah Jaringan Komputer



Dr. Fazmah Arif Yulianto, S.T., M.T.

Mengetahui,  
Kaprodi S1 Informatika



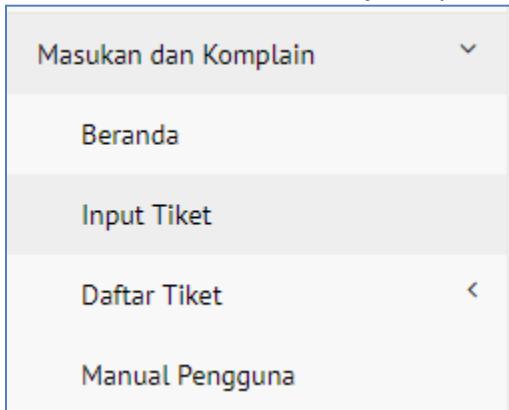
Dr. Erwin Budi Setiawan, S.Si., M.T.

## **Peraturan Praktikum Laboratorium Informatika 2022/2023**

1. Praktikum diampu oleh dosen kelas dan dibantu oleh asisten laboratorium dan asisten praktikum.
2. Praktikum dilaksanakan di Gedung Telkom University Lanmark Tower (TULT) Lantai 6 dan Lantai 7 sesuai jadwal yang ditentukan.
3. Praktikan wajib membawa modul praktikum, KTM, dan alat tulis.
4. Praktikan wajib mengisi daftar hadir *rooster* dan BAP praktikum dengan bolpoin bertinta hitam.
5. Durasi kegiatan praktikum S-1 = 2 jam (100 menit).
6. Jumlah pertemuan praktikum sebanyak 16 kali.
7. Praktikan wajib hadir minimal 75% dari seluruh pertemuan praktikum di lab. Jika total kehadiran kurang dari 75% maka nilai UAS/ Tugas Besar = 0.
8. Praktikan yang datang terlambat :
  - <= 30 menit : diperbolehkan mengikuti praktikum tanpa tambahan praktikum
  - >30 menit : tidak diperbolehkan mengikuti praktikum
9. Saat praktikum berlangsung, asisten praktikum dan praktikan:
  - Wajib menggunakan seragam sesuai aturan institusi.
  - Wajib mematikan/ mengkondisikan semua alat komunikasi.
  - Dilarang membuka aplikasi yang tidak berhubungan dengan praktikum yang berlangsung.
  - Dilarang mengubah pengaturan *software* maupun *hardware* komputer tanpa ijin.
  - Dilarang membawa makanan maupun minuman di ruang praktikum.
  - Dilarang memberikan jawaban ke praktikan lain.
  - Dilarang menyebarkan soal praktikum.
  - Dilarang membuang sampah di ruangan praktikum.
  - Wajib meletakkan alas kaki dengan rapi pada tempat yang telah disediakan.
10. Setiap praktikan dapat mengikuti praktikum susulan maksimal dua modul untuk satu mata kuliah praktikum.
  - Praktikan yang dapat mengikuti praktikum susulan hanyalah praktikan yang memenuhi syarat sesuai ketentuan institusi, yaitu: sakit (dibuktikan dengan surat keterangan medis), tugas dari institusi (dibuktikan dengan surat dinas atau dispensasi dari institusi), atau mendapat musibah atau keduakan (menunjukkan surat keterangan dari orangtua/wali mahasiswa.)
  - Persyaratan untuk praktikum susulan diserahkan sesegera mungkin kepada asisten laboratorium untuk keperluan administrasi.
  - Praktikan yang diijinkan menjadi peserta praktikum susulan ditetapkan oleh Asman Lab dan Laboran Informatika dan tidak dapat diganggu gugat.

## Tata Cara Komplain Praktikum IFLAB Melalui IGRACIAS

1. Login iGgracias.
2. Pilih Menu **Masukan dan Komplain**, pilih **Input Tiket**.



3. Pilih Fakultas/Bagian: **Bidang Akademik (FIF)**.
4. Pilih Program Studi/Urusan: **Urusan Laboratorium/Bengkel/Studio (FIF)**.
5. Pilih Layanan: **Praktikum**.
6. Pilih Kategori: **Pelaksanaan Praktikum**, lalu pilih **Sub Kategori**.
7. Isi **Deskripsi** sesuai komplain yang ingin disampaikan.

The form is titled 'Input Keluhan'. It contains the following fields:

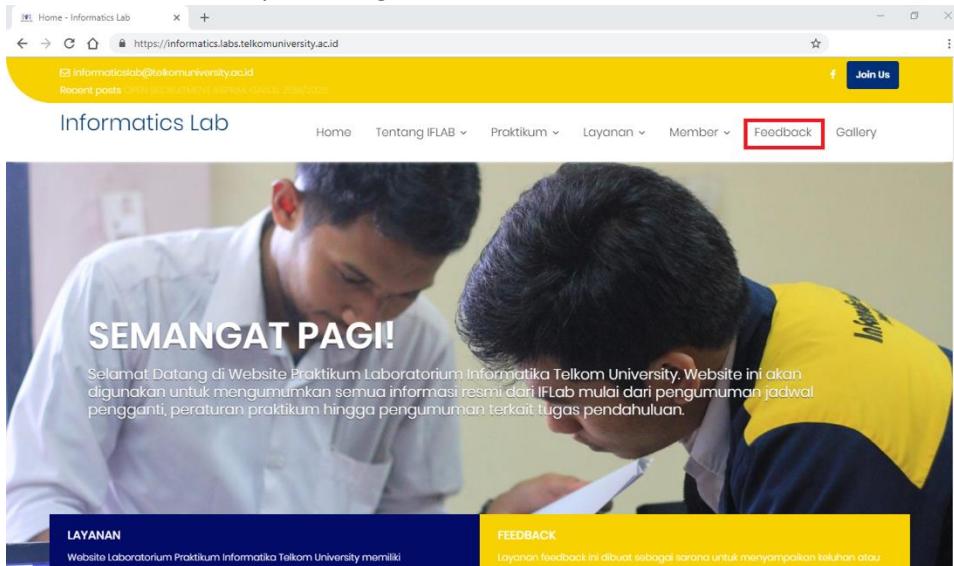
- Fakultas / Bagian : BIDANG AKADEMIK (FIF)
- Program Studi / Urusan : URUSAN LABORATORIUM/BENGKEL/STUDIO (FIF)
- Pelapor : RIZQILLAH ZAHRA LESTARI
- Layanan : PRAKTIKUM
- Kategori : Pelaksanaan Praktikum
- Sub Kategori : Please Select...
- Tipe Masukan :  Komplain  Masukan
- Deskripsi : (Rich text editor toolbar below)

A watermark 's lab' is visible on the right side of the form area.

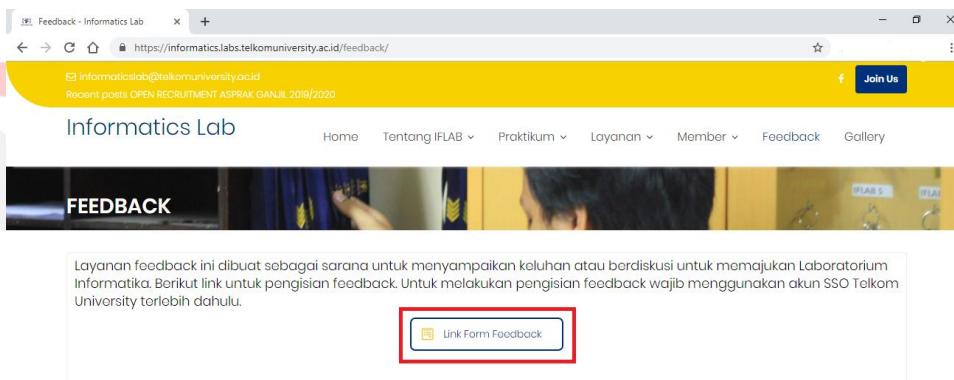
8. Lampirkan *file* jika perlu. Lalu klik Kirim.

## Tata Cara Komplain Praktikum IFLAB Melalui Website

1. Buka website <https://informatics.labs.telkomuniversity.ac.id/> melalui browser.
2. Pilih menu **Feedback** pada navigation bar website.



3. Pilih tombol **Link Form Feedback**.



4. Lakukan *login* menggunakan akun **SSO Telkom University** untuk mengakses *form feedback*.
5. Isi *form* sesuai dengan *feedback* yang ingin diberikan.

## Attribution

This lab is an integral part of the Computer Networking course using the textbook “Computer Networking: A Top-Down Approach”, 8th edition, written by J.F. Kurose and K.W. Ross.

The module uses supplementary material from the textbook. The original material can be accessed online via the URL: [http://gaia.cs.umass.edu/kurose\\_ross/index.php](http://gaia.cs.umass.edu/kurose_ross/index.php)

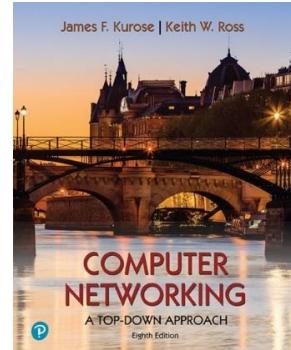
---

## Wireshark Lab

Supplement to *Computer Networking: A Top-Down Approach*, 8<sup>th</sup> ed., J.F. Kurose and K.W. Ross

“Tell me and I forget. Show me and I remember. Involve me and I understand.” Chinese proverb

© 2005-2021, J.F Kurose and K.W. Ross, All Rights Reserved



## Daftar Isi

Lembar Pengesahan.....	i
Peraturan Praktikum Laboratorium Informatika 2022/2023 .....	ii
Tata Cara Komplain Praktikum IFLAB Melalui IGRACIAS .....	iii
Tata Cara Komplain Praktikum IFLAB Melalui <i>Website</i> .....	iv
Daftar Isi.....	vi
Daftar Gambar .....	ix
Modul 1    RUNNING MODUL.....	11
Modul 2    PENGENALAN TOOLS .....	12
2.1    Pengantar.....	12
2.2    Wireshark.....	13
2.3    Menjalankan Wireshark.....	14
2.4    Menggunakan Wireshark Untuk Test Run .....	16
Modul 3    HTTP .....	19
3.1    Pengantar.....	19
3.2    Basic HTTP GET/response interaction.....	19
3.2.1    HTTP CONDITIONAL GET/response interaction .....	20
3.3    Retrieving Long Documents .....	21
3.4    HTML Documents dengan Embedded Objects .....	21
3.5    HTTP Authentication.....	22
Modul 4    DNS.....	23
4.1    Pengantar.....	23
4.2    Nslookup .....	23
4.3    DNS cache pada Komputer Anda .....	25
4.4    Tracing DNS dengan Wireshark .....	26
Modul 5    UDP .....	27
5.1    Pengantar.....	27
5.2    Tugas .....	27
Modul 6    TCP .....	29
6.1    Pengantar.....	29
6.2    Menangkap Transfer TCP dalam Jumlah Besar dari Komputer Pribadi ke Remote Server .....	29
6.3    Tampilan Awal pada Captured Trace .....	30
6.4    TCP Congestion Control .....	32
Modul 7    SOCKET PROGRAMMING: MEMBUAT APLIKASI JARINGAN.....	34
7.1    Pengantar.....	34
7.2    Program Socket dengan UDP .....	35
7.2.1    UDPClient.py .....	36

7.2.2	UDPServer.py .....	38
7.3	Program Socket dengan TCP .....	39
7.3.1	TCPClient.py .....	41
7.3.2	TCPServer.py .....	43
Modul 8	WEB SERVER DAN PEMBAGIAN TOPIK TUGAS BESAR .....	45
8.1	Pengantar.....	45
8.2	Kode .....	45
8.3	Menjalankan Server .....	45
8.4	Yang harus dikerjakan.....	45
8.5	Skeleton Kode Python untuk Web Server.....	45
8.6	Latihan Tambahan.....	46
8.7	Pembagian Topik Tugas Besar .....	47
Modul 9	IP .....	48
9.1	Pengantar.....	48
9.2	Menangkap paket dari eksekusi traceroute .....	48
9.2.1	Bagian 1: IPv4 Dasar.....	49
9.2.2	Bagian 2: Fragmentasi.....	51
9.2.3	Bagian 3: IPv6.....	51
Modul 10	DHCP .....	53
10.1	Pengantar.....	53
10.2	Mengumpulkan Jejak Paket .....	53
Modul 11	Asistensi Tugas Besar .....	55
Modul 12	ICMP .....	56
12.1	Pengantar.....	56
12.2	ICMP dan Ping .....	56
12.3	ICMP dan Traceroute .....	59
Modul 13	Ethernet and ARP .....	62
13.1	Pengantar.....	62
13.2	Menangkap dan menganalisis frame Ethernet.....	62
13.3	Address Resolution Protocol.....	64
13.3.1	Caching ARP .....	64
13.3.2	Mengamati Aksi ARP .....	65
13.3.3	Kredit Tambahan.....	66
Modul 14	Presentasi Tugas Besar.....	68
Modul 15	802.11 WiFi .....	69
15.1	Pengantar.....	69
15.2	Starting.....	69
15.3	Beacon Frames.....	70
15.4	Data Transfer .....	71
15.5	Association/Disassociation .....	71

Modul 16 ASISTENSI 2 (PRAKTIKUM SUSULAN) .....	72
Daftar Pustaka.....	73
Daftar Perubahan.....	74



## Daftar Gambar

Figure 2.1 Stuktur paket sniffer .....	13
Figure 2.2 tampilan awal wireshark .....	14
Figure 2.3 wireshark window saat dan setelah melakukan capture paket .....	15
Figure 2.4 tampilan jendela wireshark capture interface pada OS windows .....	16
Figure 2.5 tampilan jendela wireshark capture interface pada OS mac.....	16
Figure 2.6 details dari pesan HTTP yang berisi GET dari <a href="http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html">http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html</a> .....	18
Figure 3.1 tampilan jendela wireshark setelah <a href="http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html">http://gaia.cs.umass.edu/wireshark-labs/ HTTP-wireshark-file1.html</a> diambil oleh browser.....	20
Figure 4.1 dasar nslookup command.....	23
Figure 4.2 menggunakan nslookup untuk menemukan nama server authoritative untuk domain nyu.edu .....	24
Figure 4.3 menggunakan nslookup untuk melakukan reverse DNS lookup .....	25
Figure 4.4 using nslookup to find the IP address of www.iitb.ac.in and the names of the .....	25
Figure 5.1 dasar nslookup command.....	27
Figure 6.1 halaman untuk upload file alice.txt dari komputer anda ke gaia.cs.umass.edu .....	30
Figure 6.2 Success! Kamu sudah mengupload file gaia.cs.umass.edu.....	30
Figure 6.3 penjabaran pesan HTTP POST di upload dari file alice.txt dari komputer anda ke gaia.cs.umass.edu .....	31
Figure 6.4 TCP segments yang terlibat dalam pengiriman pesan HTTP POST (termasuk file alice.txt) ke gaia.cs.umass.edu .....	32
Figure 6.5 A sequence-number-versus-time plot (Stevens format) dari TCP segments.....	33
Figure 7.1 client-server application menggunakan UDP.....	36
Figure 7.2 UDPClient.py .....	36
Figure 7.3 UDPServer.py .....	38
Figure 7.4 The TCPServer process has two sockets .....	41
Figure 7.5 client-server application menggunakan TCP.....	42
Figure 9.1 Tangkapan layar Wireshark, menunjukkan paket UDP dan ICMP di tracefile ip-wireshark-trace1-1.pcapng .....	50
Figure 9.2 Tangkapan layar Wireshark, menampilkan segmen di tracefile ip-wireshark-trace1- 1.pcapng menggunakan filter tampilan ip.src==192.168.86.61 dan ip.dst==128.119.245.12 dan udp dan !icmp .....	51
Figure 9.3 Tangkapan layar Wireshark, menunjukkan paket pertama yang diambil dalam ip-wireshark-trace2- 1.pcapng. Jika Anda melihat kolom Sumber, Anda akan melihat beberapa alamat IPv6 .....	52
Figure 10.1 Tampilan Wireshark, menampilkan tangkapan DHCP Discover, Offer, Request, dan pesan ACK.....	54
Figure 12.1 Jendela Command Prompt setelah memasukkan perintah Ping .....	57
Figure 12.2 Output Wireshark untuk program Ping dengan Protokol Internet diperluas.....	58
Figure 12.3 Wireshark capture of ping packet with ICMP packet expanded. .....	59
Figure 12.4 Jendela Command Prompt menampilkan hasil program Traceroute.....	60
Figure 12.5 Jendela wireshark bidang ICMP diperluas untuk satu paket kesalahan ICMP. .....	61
Figure 13.1 Tampilan Wireshark menampilkan pesan HTTP GET untuk <a href="http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html">http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html</a> .....	63
Figure 13.2Tampilan Wireshark yang menampilkan detail frame Ethernet yang berisi permintaan HTTP GET .....	64
Figure 13.3 Menjalankan perintah “arp -a” dari salah satu komputer penulis .....	65
Figure 13.4 Permintaan ARP yang disiarkan dari salah satu komputer .....	66

Figure 15.1 Wireshark window, setelah membuka file Wireshark\_802\_11.pcap .....70



## Modul 1 RUNNING MODUL

### Tujuan Praktikum

1. Mahasiswa mengetahui aturan dan sistem pelaksanaan praktikum
2. Mahasiswa mengetahui tools yang akan digunakan dan memastikan tools berfungsi dengan baik selama pelaksanaan praktikum

Pada modul ini, Asisten praktikum akan memberikan penjelasan mengenai aturan, dan sistem pelaksanaan praktikum yang mencakup sistem kegiatan praktikum, sistem penilaian, dan sanksi untuk setiap pelanggaran yang dilakukan selama pelaksanaan praktikum. Selain itu Asisten praktikum juga akan menjelaskan modul yang akan dipelajari selama 16 pekan praktikum, dan tools yang akan digunakan selama praktikum untuk mata kuliah praktikum jaringan komputer.

Bagi para praktikan dapat memastikan bahwa tools yang akan digunakan telah terinstall di komputer lab masing-masing dan tidak ada komputer yang mengalami kerusakan sehingga dapat mengurangi kendala teknis selama pelaksanaan praktikum.

Pastikan tools berikut telah terinstall di komputer lab masing-masing:

1. Wireshark  
Jika belum terinstall dapat di download pada link berikut <http://www.wireshark.org/>
2. Python  
Jika belum terinstall dapat di download pada link berikut <https://www.python.org/downloads/>

Setelah memastikan tools pada komputer anda telah terinstall silahkan coba untuk membuka file soal1.pcap yang terlampir pada LMS kelas praktikum di wireshark masing masing, dan asisten praktikum akan menjelaskan mengenai fitur dasar pada wireshark yang akan digunakan pada praktikum selanjutnya.

## Modul 2 PENGENALAN TOOLS

### Tujuan Praktikum

1. Mahasiswa dapat melakukan instalasi *tool* yang digunakan (Wireshark).
2. Mahasiswa dapat menggunakan tool (Wireshark) untuk menangkap dan mengidentifikasi paket data

### 2.1 Pengantar

Pemahaman seseorang tentang protokol jaringan dapat diperdalam dengan "melihat protokol bekerja" dan dengan "mencoba menggunakan protokol", mengamati urutan pesan yang dipertukarkan antara dua entitas protokol, menggali rincian operasi protokol, dan menyebabkan protokol melakukan tindakan tertentu dan kemudian mengamati tindakan tersebut dan konsekuensinya. Ini dapat dilakukan dalam skenario simulasi atau dalam lingkungan jaringan "nyata" seperti Internet. Pada modul ini, Anda akan menjalankan berbagai aplikasi jaringan dalam berbagai skenario. Anda akan mengamati protokol jaringan di komputer Anda bekerja, berinteraksi dan bertukar pesan dengan entitas protokol yang dijalankan di tempat lain di Internet.

Pada modul ini, Anda akan berkenalan dengan **Wireshark**, dan membuat beberapa penangkapan dan pengamatan paket sederhana.

Alat dasar untuk mengamati pesan yang dipertukarkan antara entitas protokol disebut **packet sniffer**. Packet sniffer menangkap ("sniffs") pesan yang dikirim/diterima dari/oleh komputer; dan biasanya akan menyimpan dan/atau menampilkan konten dari berbagai bidang protokol dalam pesan yang diambil ini. Packet sniffer sendiri bersifat **pasif**. packet sniffer mengamati pesan yang dikirim dan diterima oleh aplikasi dan protokol yang berjalan di komputer, tetapi tidak dapat mengirim paket itu sendiri. Demikian pula, paket yang diterima tidak pernah secara eksplisit ditujukan ke packet sniffer. Sebagai gantinya, packet sniffer menerima salinan paket yang dikirim/diterima dari/oleh aplikasi dan protokol yang dijalankan pada komputer anda.

Figure 2.1 menunjukkan struktur packet sniffer. Di sebelah kanan Figure 2.1 adalah protokol (dalam hal ini, protokol Internet) dan aplikasi (seperti browser web atau klien email) yang biasanya berjalan di komputer. Paket sniffer, yang ditunjukkan dalam kotak putus-putus pada Figure 2.1 adalah tambahan untuk perangkat lunak biasa di komputer Anda, dan terdiri dari dua bagian. **packet capture library** menerima salinan dari setiap *frame link layer* yang dikirim dari atau diterima oleh komputer Anda melalui antarmuka tertentu (link layer, seperti Ethernet atau WiFi). (Figure 2.1) bahwa pesan yang dipertukarkan oleh protokol lapisan yang lebih tinggi seperti HTTP, FTP, TCP, UDP, DNS, atau IP semua akhirnya dikemas dalam *frame link layer* yang ditransmisikan melalui media fisik seperti kabel Ethernet atau radio WiFi 802.11 . Menangkap semua *frame link layer* dengan demikian memberi Anda semua pesan yang dikirim/diterima di seluruh tautan yang dipantau dari/oleh semua protokol dan aplikasi yang dijalankan di komputer Anda.

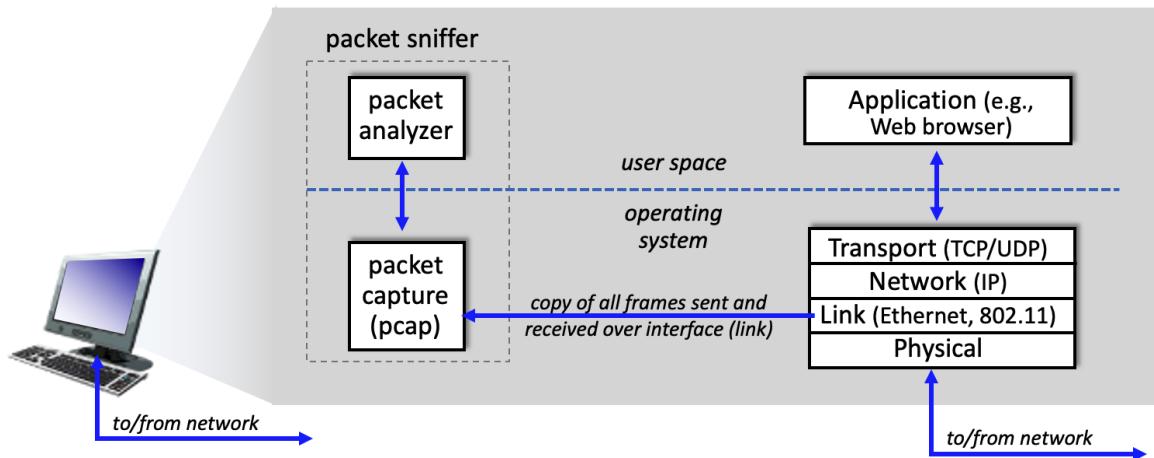


Figure 2.1 Stuktur paket sniffer

Komponen kedua dari packet sniffer adalah **packet analyzer**, yang menampilkan isi dari semua bidang dalam pesan protokol. Untuk melakukannya packet analyzer harus "memahami" struktur semua pesan yang dipertukarkan oleh protokol. Misalnya, kita tertarik untuk menampilkan berbagai bidang dalam pesan yang dipertukarkan oleh protokol HTTP pada Figure 2.1. Penganalisis paket memahami format frame Ethernet, dan juga dapat mengidentifikasi datagram IP dalam frame Ethernet. Ia juga memahami format datagram IP, sehingga dapat mengekstrak segmen TCP dalam datagram IP. Akhirnya, ia memahami struktur segmen TCP, sehingga dapat mengekstrak pesan HTTP yang terkandung dalam segmen TCP, dan memahami protokol HTTP, misalnya mengetahui bahwa byte pertama dari pesan HTTP akan berisi string "GET," "POST," or "HEAD".

Kita akan menggunakan sniffer paket Wireshark [<http://www.wireshark.org/>] untuk modul ini, ini memungkinkan untuk menampilkan konten pesan yang dikirim/diterima dari/oleh protokol di berbagai tingkat tumpukan protokol. (Secara teknis, Wireshark adalah packet analyzer yang menggunakan *packet capture library* di komputer Anda. Secara teknis, Wireshark menangkap *frame link layer* seperti yang ditunjukkan pada Figure 2.1, tetapi menggunakan istilah umum "paket" untuk merujuk ke *frame link layer*, datagram lapisan jaringan, segmen lapisan transportasi, dan pesan lapisan aplikasi, jadi kita akan menggunakan istilah "paket" yang kurang tepat di sini untuk melanjutkan dengan konvensi Wireshark).

**Wireshark** adalah penganalisa protokol jaringan gratis yang berjalan di komputer Windows, Mac, dan Linux/Unix. Wireshark adalah penganalisa paket yang ideal, stabil, memiliki basis pengguna yang besar dan dukungan yang terdokumentasi dengan baik yang mencakup panduan pengguna ([http://www.wireshark.org/docs/wsug\\_html\\_chunked/](http://www.wireshark.org/docs/wsug_html_chunked/)), halaman manual (<http://www.wireshark.org/docs/man-pages/>), dan FAQ yang mendetail (<http://www.wireshark.org/faq.html>), wireshark juga memiliki fungsionalitas yang kaya yang mencakup kemampuan untuk menganalisis ratusan protokol, dan antarmuka pengguna yang dirancang dengan baik. Wireshark beroperasi di komputer yang menggunakan Ethernet, serial (PPP), LAN nirkabel, 802.11 (WiFi), dan banyak teknologi lapisan tautan lainnya.

## 2.2 Wireshark

Untuk menjalankan Wireshark, Anda harus memiliki akses ke komputer yang mendukung Wireshark dan libpcap atau WinPCap packet capture library. Perangkat lunak libpcap akan diinstal jika belum terinstal dalam sistem operasi Anda, saat Anda menginstal Wireshark. Lihat

<http://www.wireshark.org/download.html> untuk daftar sistem operasi yang didukung dan situs unduhan.

Unduh dan instal perangkat lunak Wireshark:

- Masuk ke halaman <http://www.wireshark.org/download.html> dan unduh dan instal binary Wireshark pada komputer Anda.

FAQ Wireshark memiliki sejumlah petunjuk bermanfaat dan informasi menarik, terutama jika Anda kesulitan menginstal atau menjalankan Wireshark.

## 2.3 Menjalankan Wireshark

Saat Anda menjalankan program Wireshark, Anda akan mendapatkan tampilan awal yang terlihat seperti pada figure 2.2. Versi Wireshark yang berbeda akan memiliki tampilan awal yang berbeda.

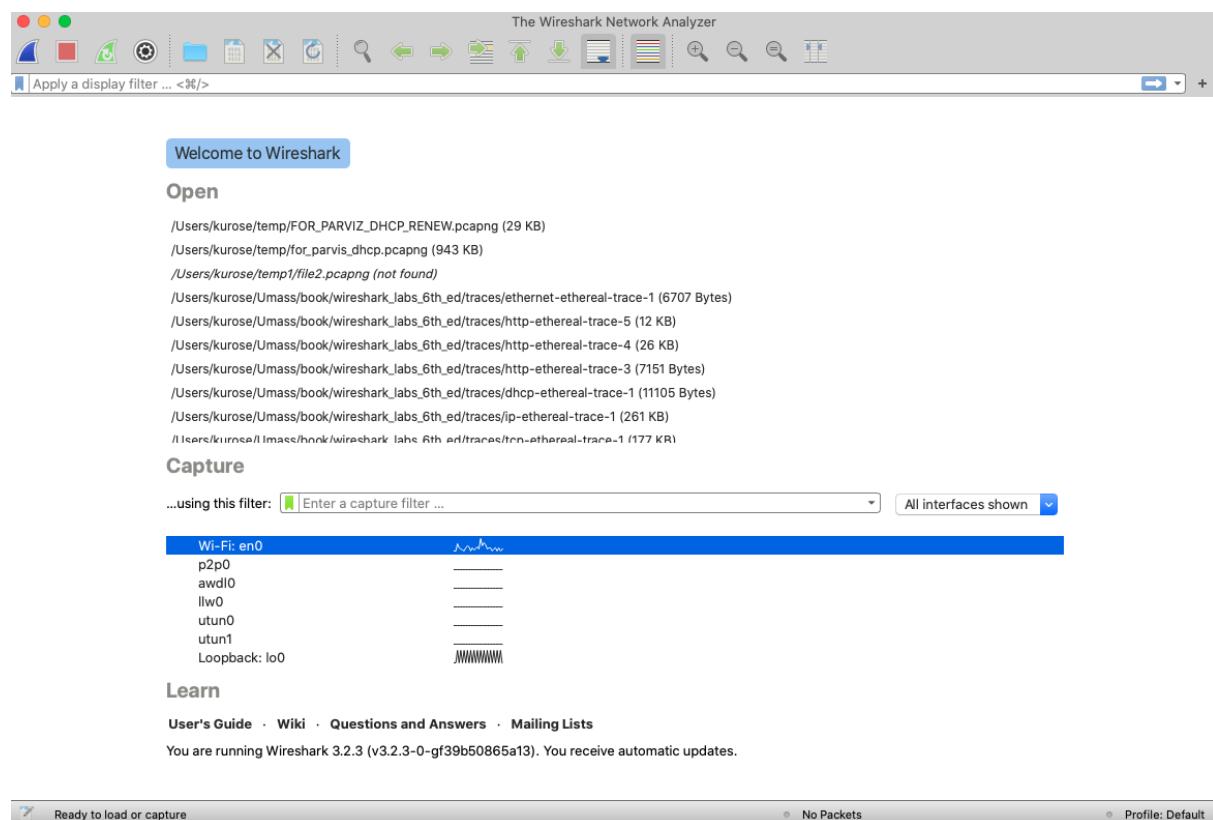


Figure 2.2 tampilan awal wireshark

Perhatikan bahwa di bawah bagian Capture, ada daftar yang disebut *interfaces*. Komputer Mac pada gambar diatas hanya memiliki satu interfaces – “Wi-Fi en0” (diarsir dengan warna biru pada figure 2.2) yang merupakan interfaces untuk akses Wi-Fi. Semua paket ke/dari komputer ini akan melewati interfaces Wi-Fi, jadi di sinilah kita akan menangkap paket. Lalu klik dua kali pada interfaces ini.

Jika Anda mengklik interfaces ini untuk memulai pengambilan paket (yaitu, agar Wireshark mulai menangkap semua paket yang dikirim ke/dari interfaces itu), tampilan seperti figure 2.3 akan ditampilkan, dan menampilkan informasi tentang paket yang diambil. Setelah memulai pengambilan paket, Anda dapat menghentikannya dengan menggunakan menu **pull down Capture** dan memilih **Stop** (atau dengan mengklik tombol kotak merah di sebelah sirip Wireshark pada figure 2.2).

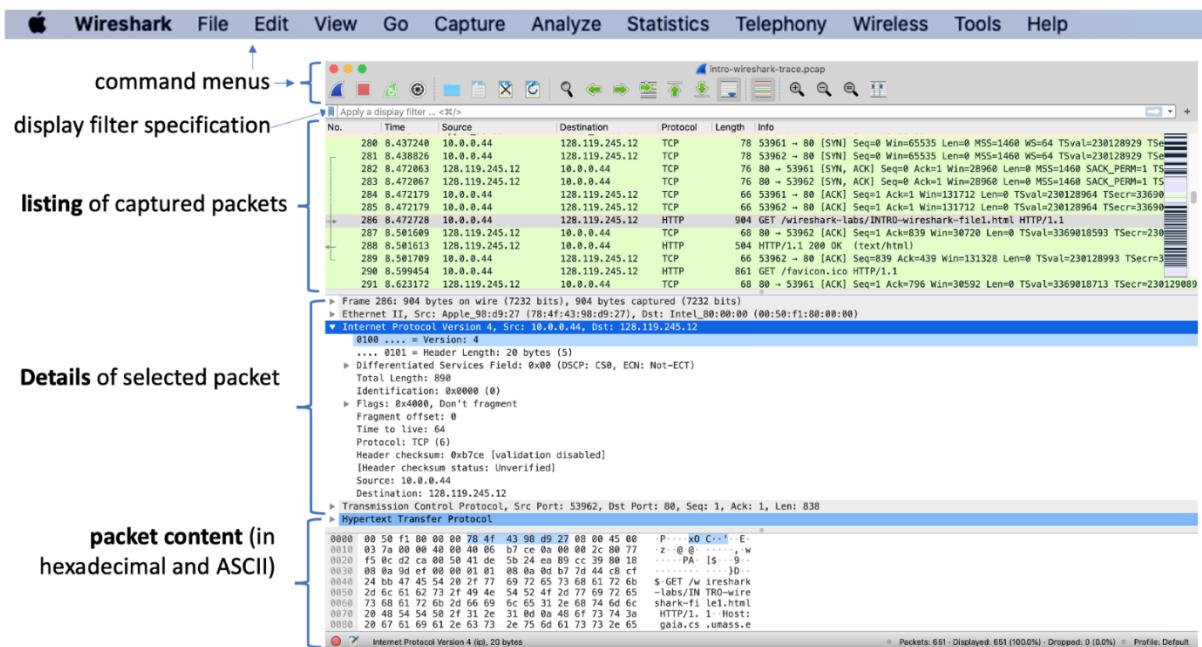


Figure 2.3 wireshark window saat dan setelah melakukan capture paket

Antarmuka Wireshark memiliki lima komponen utama:

- **command menu** adalah menu pull-down standar yang terletak di bagian atas jendela Wireshark. Menu File memungkinkan Anda untuk menyimpan data paket yang diambil atau membuka file yang berisi data paket yang diambil sebelumnya dan keluar dari aplikasi Wireshark. Menu Tangkap memungkinkan Anda untuk memulai pengambilan paket.
- **packet-listing window** menampilkan ringkasan satu baris untuk setiap paket yang diambil, termasuk nomor paket saat paket ditangkap, sumber paket dan alamat tujuan, jenis protokol, dan informasi khusus protokol yang terkandung dalam paket. Daftar paket dapat diurutkan menurut salah satu kategori ini dengan mengklik nama kolom. Bidang jenis protokol mencantumkan protokol tingkat tertinggi yang mengirim atau menerima paket ini, yaitu protokol yang merupakan sumber atau sink utama untuk paket ini.
- **packet-header details window** memberikan rincian tentang paket yang dipilih (disorot) di jendela daftar paket. (Untuk memilih paket di jendela daftar paket, letakkan kursor di atas ringkasan satu baris paket di window daftar paket dan klik dengan tombol kiri mouse.). Rincian ini mencakup informasi tentang frame Ethernet (dengan asumsi paket dikirim/diterima melalui antarmuka Ethernet) dan datagram IP yang berisi paket ini. Jumlah detail Ethernet dan lapisan IP yang ditampilkan dapat diperluas atau diminimalkan dengan mengklik kotak plus/minus atau segitiga yang mengarah ke kanan/bawah di sebelah kiri frame Ethernet atau garis datagram IP di window detail paket. Jika paket telah dibawa melalui TCP atau UDP, rincian TCP atau UDP juga akan ditampilkan, yang juga dapat diperluas atau diperkecil. Terakhir, rincian tentang protokol tingkat tertinggi yang mengirim atau menerima paket ini juga disediakan.
- **packet-contents window** menampilkan seluruh isi frame yang diambil, baik dalam format ASCII maupun heksadesimal.
- Pada atas antarmuka pengguna grafis Wireshark, adalah **packet display filter field**, di mana nama protokol atau informasi lain dapat dimasukkan untuk menyaring informasi yang ditampilkan di jendela daftar paket (dan karenanya windows header-paket dan isi-paket). Pada contoh di bawah ini, kita akan menggunakan bidang packet-display filter untuk

menyembunyikan (bukan menampilkan) paket Wireshark kecuali paket yang sesuai dengan pesan HTTP.

## 2.4 Menggunakan Wireshark Untuk Test Run

Diasumsikan bahwa komputer Anda terhubung ke Internet melalui antarmuka Ethernet kabel atau antarmuka WiFi 802.11 nirkabel. Lakukan hal berikut:

1. Jalankan browser web Anda, yang akan menampilkan beranda anda.
2. Jalankan Wireshark. Anda awalnya akan melihat jendela yang mirip dengan yang ditunjukkan pada Figure 2.2. dimana Wireshark belum mulai menangkap paket.
3. Untuk memulai pengambilan paket, pilih menu **pull down Capture** dan pilih **Interfaces**. Ini akan menyebabkan windows "*Wireshark: Capture Interfaces*" ditampilkan. Anda akan melihat daftar interfaces, seperti yang ditunjukkan pada Figure 2.4 (Windows) dan 2.5 (Mac).

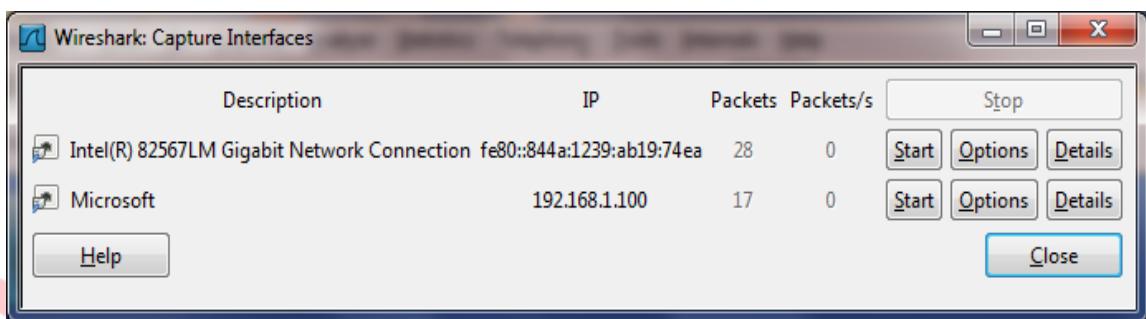


Figure 2.4 tampilan jendela wireshark capture interface pada OS windows

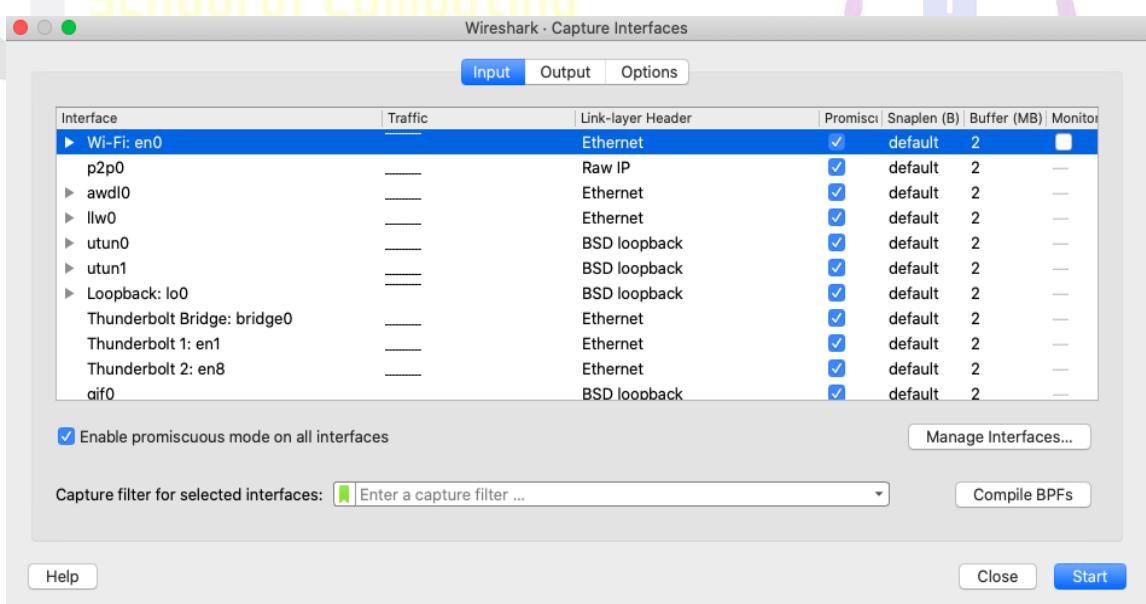


Figure 2.5 tampilan jendela wireshark capture interface pada OS mac

4. Anda akan melihat daftar antarmuka di komputer Anda serta jumlah paket yang telah diamati pada antarmuka itu sejauh ini. Pada mesin Windows, klik Mulai untuk antarmuka di mana Anda ingin memulai pengambilan paket (dalam kasus di figure 2.4 , Koneksi jaringan Gigabit). Pada Windows, pilih antarmuka dan klik Mulai di bagian bawah jendela. Pengambilan paket sekarang akan dimulai - Wireshark akan menangkap semua paket yang dikirim/diterima dari/oleh komputer Anda

5. Setelah Anda mulai menangkap paket, sebuah jendela yang mirip dengan yang ditunjukkan pada figure 2.3 akan muncul. Jendela ini menunjukkan paket yang ditangkap. Dengan memilih menu *drop down* Tangkap dan pilih Berhenti, atau dengan mengklik kotak Stop merah, Anda dapat menghentikan pengambilan paket. Mari kita ambil beberapa paket menarik terlebih dahulu. Untuk melakukannya, kita perlu menghasilkan beberapa lalu lintas jaringan. Mari kita lakukan dengan menggunakan browser web, yang akan menggunakan protokol HTTP.
6. Saat Wireshark sedang berjalan, masukkan URL: <http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html> dan tampilkan halaman tersebut di browser Anda. Untuk menampilkan halaman ini, browser Anda akan menghubungi server HTTP di gaia.cs.umass.edu dan bertukar pesan HTTP dengan server untuk mendownload halaman ini. Frame Ethernet atau WiFi yang berisi pesan HTTP ini (serta semua frame lain yang melewati adaptor Ethernet atau WiFi) akan ditangkap oleh Wireshark.
7. Setelah browser Anda menampilkan halaman INTRO-wireshark-file1.html (ini adalah satu baris ucapan selamat yang sederhana), hentikan pengambilan paket Wireshark dengan memilih berhenti di jendela pengambilan Wireshark. Jendela utama Wireshark sekarang akan terlihat seperti figure 2.3. Anda sekarang memiliki data paket langsung yang berisi semua pesan protokol yang dipertukarkan antara komputer Anda dan entitas jaringan lainnya! Pertukaran pesan HTTP dengan server web gaia.cs.umass.edu akan muncul di suatu tempat dalam daftar paket yang diambil. Tetapi akan ada banyak jenis paket lain yang ditampilkan juga. Meskipun satu-satunya tindakan yang Anda lakukan adalah mengunduh halaman web, ternyata ada banyak protokol lain yang berjalan di komputer Anda yang tidak terlihat oleh pengguna.
8. Ketik "http" (tanpa tanda kutip, dan dalam huruf kecil – semua nama protokol dalam huruf kecil di Wireshark, dan pastikan untuk menekan tombol enter/return Anda) ke jendela spesifikasi filter tampilan di bagian atas halaman utama jendela Wireshark. Kemudian pilih Terapkan (di sebelah kanan tempat Anda memasukkan "http") atau tekan saja kembali. Ini akan menyebabkan hanya pesan HTTP yang ditampilkan di jendela daftar paket. figure 2.6 di bawah ini menunjukkan tangkapan layar setelah filter http diterapkan ke jendela pengambilan paket yang ditunjukkan sebelumnya pada figure2.3. Perhatikan juga bahwa di jendela detail paket yang dipilih, kami telah memilih untuk menampilkan konten terperinci untuk pesan aplikasi Hypertext Transfer Protocol yang ditemukan dalam segmen TCP, yang ada di dalam datagram IPv4 yang ada di dalam bingkai Ethernet II (WiFi). Berfokus pada konten pada pesan, segmen, datagram, dan tingkat bingkai tertentu memungkinkan kita fokus pada apa yang ingin kita lihat (dalam hal ini pesan HTTP).

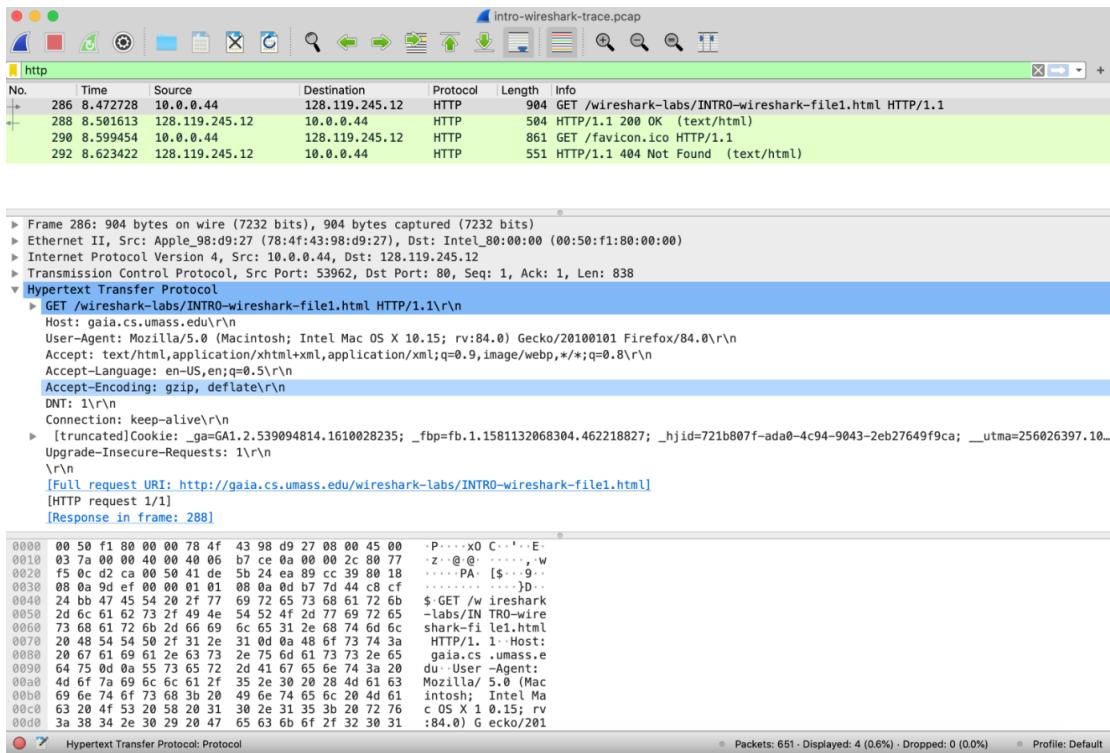


Figure 2.6 details dari pesan HTTP yang berisi GET dari <http://gala.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html>

9. Temukan pesan HTTP GET yang dikirim dari komputer Anda ke server HTTP [gaia.cs.umass.edu](http://gala.cs.umass.edu). (Cari pesan HTTP GET di bagian "daftar paket yang diambil" dari jendela Wireshark (lihat Figure 2.3 dan 2.6) yang menunjukkan "GET" diikuti oleh URL [gala.cs.umass.edu](http://gala.cs.umass.edu) yang Anda masukkan. Saat Anda memilih pesan HTTP GET, frame Ethernet, datagram IP, segmen TCP, dan informasi header pesan HTTP akan ditampilkan di jendela header-paket 3. Dengan mengklik '+' dan '-' dan panah mengarah ke kanan dan ke bawah ke sisi kiri jendela detail paket, meminimalkan jumlah informasi Frame, Ethernet, Internet Protocol, dan Transmission Control Protocol yang ditampilkan.
- Maksimalkan jumlah informasi yang ditampilkan tentang protokol HTTP. Tampilan Wireshark Anda sekarang akan terlihat kira-kira seperti yang ditunjukkan pada Figure 2.3. (Perhatikan, khususnya, jumlah informasi protokol yang diminimalkan untuk semua protokol kecuali HTTP, dan jumlah informasi protokol yang dimaksimalkan untuk HTTP di jendela header paket).

## 10. Keluar dari Wireshark

## Modul 3 HTTP

### Tujuan Praktikum

1. Mahasiswa dapat menginvestigasi cara kerja protokol HTTP menggunakan Wireshark.

### 3.1 Pengantar

Kita sekarang siap menggunakan Wireshark untuk menyelidiki protokol yang sedang beroperasi. Di modul ini, kita akan mempelajari beberapa aspek protokol HTTP: interaksi dasar GET/response, format pesan HTTP, mengambil file HTML besar, mengambil file HTML dengan objek yang disematkan, serta autentikasi dan keamanan HTTP.

### 3.2 Basic HTTP GET/response interaction

Mari kita mulai penjelajahan HTTP dengan mengunduh file HTML yang sangat sederhana - file yang sangat pendek, dan tidak berisi objek yang disematkan. Lakukan hal berikut:

1. Jalankan browser web Anda.
2. Jalankan packet sniffer Wireshark. Masukkan "http" (hanya huruf, bukan tanda kutip, dan dalam huruf kecil) di jendela spesifikasi filter tampilan, sehingga hanya pesan HTTP yang diambil yang akan ditampilkan nanti di jendela daftar paket.
3. Tunggu sedikit lebih dari satu menit, dan kemudian mulai pengambilan paket Wireshark.
4. Masukkan berikut ini ke browser Anda <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html>, Browser Anda akan menampilkan file HTML satu baris yang sangat sederhana.
5. Hentikan pengambilan paket Wireshark.

Jendela Wireshark Anda akan terlihat mirip dengan jendela yang ditunjukkan pada Gambar 2.1. Jika Anda tidak dapat menjalankan Wireshark pada koneksi jaringan langsung, Anda dapat mengunduh trace paket yang dibuat saat langkah-langkah di atas diikuti.

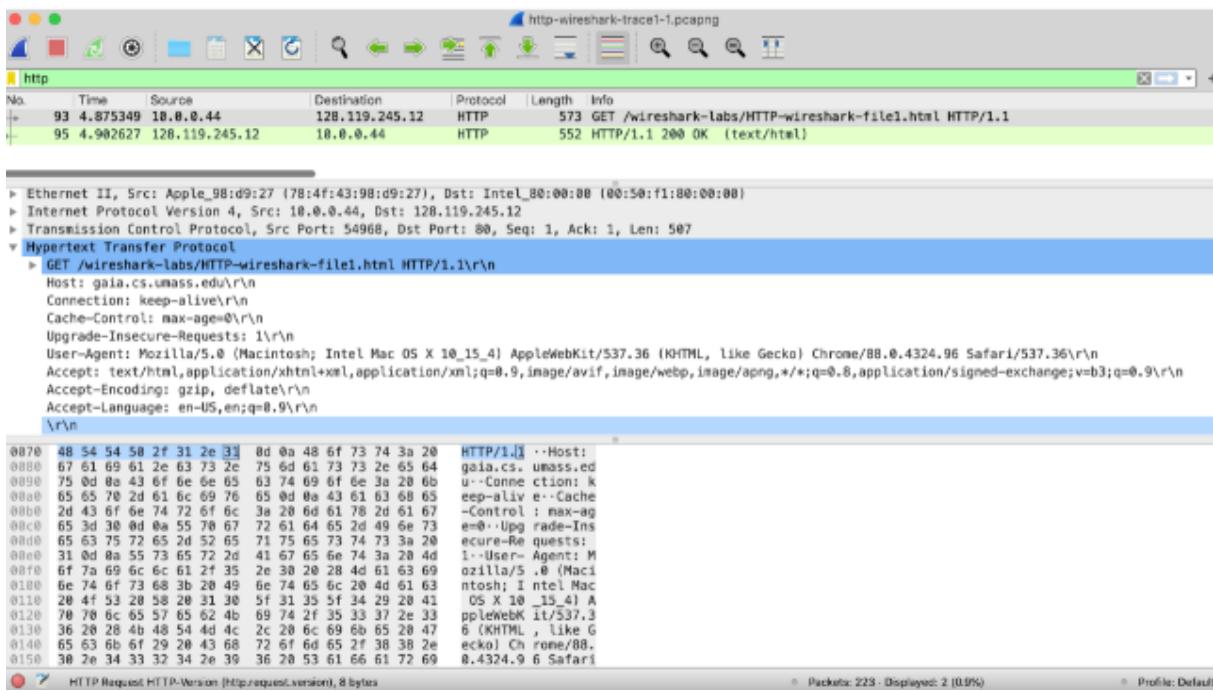


Figure 3.1 tampilan jendela wireshark setelah <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html> diambil oleh browser

Contoh pada Gambar 3.1 menunjukkan di jendela daftar paket bahwa dua pesan HTTP ditangkap: pesan GET (dari browser Anda ke server web gaia.cs.umass.edu) dan pesan respons dari server ke browser Anda.

Jendela isi-paket menunjukkan rincian pesan yang dipilih (dalam hal ini pesan HTTP OK, yang disorot di jendela daftar-paket). Ingatlah bahwa karena pesan HTTP dibawa di dalam segmen TCP, yang dibawa di dalam datagram IP, yang dibawa dalam bingkai Ethernet, Wireshark juga menampilkan informasi paket Frame, Ethernet, IP, dan TCP. Kita ingin meminimalkan jumlah data non-HTTP yang ditampilkan, jadi pastikan kotak di paling kiri Frame, Ethernet, IP dan Informasi TCP memiliki tanda plus atau segitiga siku-siku (yang berarti ada informasi yang tersembunyi dan tidak ditampilkan), dan garis HTTP memiliki tanda minus atau segitiga yang mengarah ke bawah (yang berarti bahwa semua informasi tentang pesan HTTP ditampilkan).

### 3.2.1 HTTP CONDITIONAL GET/response interaction

Sebagian besar browser web melakukan caching objek dan dengan demikian sering melakukan GET bersyarat saat mengambil objek HTTP. Sebelum melakukan langkah-langkah di bawah ini, pastikan cache browser Anda kosong. Sekarang lakukan hal berikut:

- Jalankan browser web Anda, dan pastikan cache browser Anda dibersihkan, seperti yang dibahas di atas.
- Mulai sniffer paket Wireshark
- Masukkan URL berikut ke browser Anda <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html> Browser Anda akan menampilkan file HTML lima baris yang sangat sederhana.
- Masukkan kembali URL yang sama ke browser Anda dengan cepat (atau cukup pilih tombol segarkan di browser Anda)

- Hentikan pengambilan paket Wireshark, dan masukkan "http" di jendela spesifikasi filter tampilan, sehingga hanya pesan HTTP yang diambil yang akan ditampilkan nanti di jendela daftar paket.

Jika Anda tidak dapat menjalankan Wireshark pada koneksi jaringan langsung (atau tidak dapat membuat browser Anda mengeluarkan bidang If-Modified-Since pada permintaan HTTP GET kedua), Anda dapat mengunduh jejak paket yang dibuat saat langkah-langkah di atas diikuti.

### **3.3 Retrieving Long Documents**

Dalam contoh kita sejauh ini, dokumen yang diambil adalah file HTML sederhana dan pendek. Mari kita lihat apa yang terjadi ketika kita mengunduh file HTML yang panjang. Lakukan hal berikut:

- Jalankan browser web Anda, dan pastikan cache browser Anda dibersihkan.
- Mulai sniffer paket Wireshark
- Masukkan URL berikut ke browser Anda <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html> Browser Anda seharusnya menampilkan Bill of Rights AS yang agak panjang.
- Hentikan pengambilan paket Wireshark, dan masukkan "http" di jendela tampilan-filter-spesifikasi, sehingga hanya pesan HTTP yang diambil yang akan ditampilkan.

Di jendela daftar paket, Anda akan melihat pesan HTTP GET Anda, diikuti oleh respons TCP multi-paket untuk permintaan HTTP GET Anda. Pastikan filter tampilan Wireshark Anda dihapus sehingga respons TCP multi-paket akan ditampilkan dalam daftar paket.

Respons multi-paket ini membutuhkan sedikit penjelasan. Dalam kasus HTTP GET kita, badan entitas dalam respons adalah seluruh file HTML yang diminta. Dalam kasus kita di sini, file HTML agak panjang, dan pada 4500 byte terlalu besar untuk muat dalam satu paket TCP. Di Wireshark versi terbaru, Wireshark menunjukkan setiap segmen TCP sebagai paket terpisah, dan fakta bahwa respons HTTP tunggal terfragmentasi di beberapa paket TCP ditunjukkan oleh "segmen TCP dari PDU yang dipasang kembali" di kolom Info pada tampilan Wireshark .

### **3.4 HTML Documents dengan Embedded Objects**

Sekarang kita telah melihat bagaimana Wireshark menampilkan lalu lintas paket yang diambil untuk file HTML besar, kita dapat melihat apa yang terjadi ketika browser Anda mengunduh file dengan objek yang disematkan, yaitu file yang menyertakan objek lain (dalam contoh di bawah, file gambar ) yang disimpan di server lain.

Lakukan hal berikut:

- Jalankan browser web Anda, dan pastikan cache browser Anda dibersihkan, seperti yang dibahas di atas.
- Mulai sniffer paket Wireshark
- Masukkan URL berikut ke browser Anda <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html>
- Browser Anda harus menampilkan file HTML pendek dengan dua gambar. Kedua gambar ini direferensikan dalam file HTML dasar. Artinya, gambar itu sendiri tidak terdapat dalam HTML; alih-alih URL untuk gambar terdapat dalam file HTML yang diunduh. Browser Anda harus mengambil logo ini dari situs web yang ditunjukkan. Logo penerbit kita diambil dari situs web gaia.cs.umass.edu.

- Hentikan pengambilan paket Wireshark, dan masukkan "http" di jendela tampilan-filter-spesifikasi, sehingga hanya pesan HTTP yang diambil yang akan ditampilkan.

### 3.5 HTTP Authentication

Terakhir, mari kita coba mengunjungi situs web yang dilindungi kata sandi dan memeriksa urutan pesan HTTP yang dipertukarkan untuk situs tersebut. URL-nya [http://gaia.cs.umass.edu/wireshark-labs/protected\\_pages/HTTP-wireshark-file5.html](http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html) dilindungi kata sandi. Nama pengguna adalah "wireshark-students" (tanpa tanda kutip), dan kata sandinya adalah "network" (sekali lagi, tanpa tanda kutip). Jadi mari kita akses situs yang dilindungi kata sandi "secure" ini. Lakukan hal berikut:

- Pastikan cache browser Anda dibersihkan, dan tutup browser Anda. Kemudian, mulai browser Anda
- Mulai sniffer paket Wireshark
- Masukkan URL berikut ke browser Anda [http://gaia.cs.umass.edu/wireshark-labs/protected\\_pages/HTTP-wireshark-file5.html](http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html) Ketik nama pengguna dan kata sandi yang diminta ke dalam kotak pop up.
- Hentikan pengambilan paket Wireshark, dan masukkan "http" di jendela spesifikasi filter tampilan, sehingga hanya pesan HTTP yang diambil yang akan ditampilkan nanti di jendela daftar paket.

Catatan: Jika Anda tidak dapat menjalankan Wireshark pada koneksi jaringan langsung, Anda dapat menggunakan jejak paket http-ethereal-trace-5 "classic", atau trace tambahan lainnya.

Sekarang mari kita periksa output Wireshark. Anda mungkin ingin membaca terlebih dahulu tentang otentikasi HTTP dengan meninjau materi yang mudah dibaca tentang "Kerangka Otentikasi Akses HTTP" di [http://frontier.userland.com/stories/storyReader\\$2159](http://frontier.userland.com/stories/storyReader$2159)

Nama pengguna (wireshark-students) dan kata sandi (network) yang Anda masukkan dikodekan dalam string karakter (d2lyZXNoYXJrLXN0dWRlbnRzOm5IdHdv cms=) mengikuti header "Otorisasi: Dasar" di pesan HTTP GET klien. Meskipun tampaknya nama pengguna dan kata sandi Anda dienkripsi, mereka hanya dikodekan dalam format yang dikenal sebagai format Base64. Nama pengguna dan kata sandi tidak dienkripsi!

Untuk melihat ini, buka <http://www.motobit.com/util/base64-decoder-encoder.asp> dan masukkan string yang disandikan base64 d2lyZXNoYXJrLXN0dWRlbnRz dan decode. Anda telah menerjemahkan dari penyandian Base64 ke penyandian ASCII, dan dengan demikian akan melihat nama pengguna Anda. Untuk melihat kata sandi, masukkan sisa string Om5IdHdv cms= dan tekan decode. Karena siapa pun dapat mengunduh alat seperti Wireshark dan mendapatkan paket (bukan hanya milik mereka sendiri) yang melewati adaptor jaringan mereka, dan siapa pun dapat menerjemahkan dari Base64 ke ASCII, Anda harus memahami bahwa kata sandi sederhana di WWW situs tidak aman kecuali tindakan tambahan diambil.

## Modul 4 DNS

### Tujuan Praktikum

1. Mahasiswa dapat menginvestigasi cara kerja DNS menggunakan Wireshark

### 4.1 Pengantar

Domain Name System (DNS) menerjemahkan nama host ke alamat IP, memenuhi peran penting dalam infrastruktur Internet. Ingatlah bahwa peran klien dalam DNS relatif sederhana – klien mengirimkan kueri ke server DNS lokalnya, dan menerima tanggapan kembali, karena server DNS memiliki hierarkis berkomunikasi satu sama lain untuk menyelesaikan kueri DNS klien secara rekursif atau berulang. Namun, dari sudut pandang klien DNS, protokolnya cukup sederhana – kueri diformulasikan ke server DNS lokal dan respons diterima dari server tersebut.

Secara khusus, Anda mungkin ingin meninjau materi di server DNS lokal, cache DNS, catatan dan pesan DNS, dan bidang JENIS dalam catatan DNS.

### 4.2 Nslookup

Mari mulai tentang DNS dengan memeriksa perintah nslookup, yang akan memanggil layanan DNS yang mendasarinya untuk mengimplementasikan fungsinya. Perintah nslookup tersedia di sebagian besar sistem operasi Microsoft, Apple IOS, dan Linux. Untuk menjalankan nslookup Anda cukup mengetikkan perintah nslookup pada baris perintah di jendela DOS, jendela terminal Mac IOS, atau shell Linux.

Dalam operasinya yang paling dasar, nslookup memungkinkan host yang menjalankan nslookup untuk menanyakan server DNS tertentu untuk catatan DNS. Server DNS yang ditanyakan dapat berupa server DNS root, top-level-domain DNS server (TLD), server DNS otoritatif, atau server DNS perantara. Misalnya, nslookup dapat digunakan untuk mengambil catatan DNS “Tipe=A” yang memetakan nama host (contohnya www.nyu.edu) ke alamat IP-nya. Untuk menyelesaikan tugas ini, nslookup mengirimkan permintaan DNS ke server DNS yang ditentukan (atau server DNS lokal default untuk host yang menjalankan nslookup, jika tidak ada server DNS tertentu yang ditentukan), menerima respons DNS dari server DNS tersebut, dan menampilkan hasilnya.

Pertama-tama kita akan menjalankan nslookup pada baris perintah Linux di host newworld.cs.umass.edu yang terletak di Departemen CS di kampus University of Massachusetts (UMass), di mana server nama lokal bernama primo.cs.umass.edu (yang memiliki alamat IP 128.119.240.1). Mari kita coba nslookup dalam bentuknya yang paling sederhana:

```
[newworld.cs.umass.edu] nslookup www.nyu.edu
Server:      128.119.240.1
Address:     128.119.240.1#53

Non-authoritative answer:
www.nyu.edu      canonical name = WEB.GSLB.nyu.edu.
Name:   WEB.GSLB.nyu.edu
Address: 216.165.47.12
Name:   WEB.GSLB.nyu.edu
Address: 2607:f600:1002:6113::100
```

Figure 4.1 dasar nslookup command

Dalam contoh ini perintah nslookup diberikan satu argumen, nama host (www.nyu.edu). Dengan kata lain, perintah ini mengatakan “tolong kirimkan saya alamat IP untuk host www.nyu.edu.”

Seperti yang ditunjukkan pada tangkapan layar, respons dari perintah ini memberikan dua informasi:

(1) nama dan alamat IP server DNS yang memberikan jawabannya – dalam hal ini server DNS lokal di UMass; dan

(2) nama host kanonik dan alamat IP [www.nyu.edu](http://www.nyu.edu).

Anda mungkin telah memperhatikan bahwa ada dua pasangan nama/alamat yang disediakan untuk [www.nyu.edu](http://www.nyu.edu). Yang pertama (216.165.47.12) adalah alamat IPv4 dalam notasi desimal bertitik yang terlihat familiar; yang kedua (2607:f600:1002:6113::100) adalah alamat IPv6 yang tampak lebih panjang dan lebih rumit. Kita akan belajar tentang IPv4 dan IPv6 dan dua skema pengalamatan yang berbeda nanti.

Untuk saat ini, mari kita fokus pada dunia IPv4 kita yang lebih nyaman.

Meskipun tanggapan datang dari server DNS lokal (dengan alamat IP 128.119.240.1) di UMass, sangat mungkin bahwa server DNS lokal ini berulang kali menghubungi beberapa server DNS lain untuk mendapatkan jawabannya.

Selain menggunakan nslookup untuk mengkueri record DNS “Type=A”, kita juga dapat menggunakan nslookup to nslookup untuk query record “TYPE=NS”, yang mengembalikan nama host (dan alamat IP-nya) dari server DNS otoritatif yang tahu cara mendapatkan alamat IP untuk host di domain server otoritatif.

```
newworld.cs.umass.edu> nslookup -type=NS nyu.edu
Server:      128.119.240.1
Address:     128.119.240.1#53

Non-authoritative answer:
nyu.edu nameserver = ns2.nyu.org.
nyu.edu nameserver = ns4.nyu.edu.
nyu.edu nameserver = ns1.nyu.net.

Authoritative answers can be found from:
ns2.nyu.org      internet address = 128.122.0.76
ns1.nyu.net      internet address = 128.122.0.8
ns4.nyu.edu      internet address = 216.165.87.102
ns4.nyu.edu      has AAAA address 2607:f600:2001:6100::135
```

Figure 4.2 menggunakan nslookup untuk menemukan nama server authoritative untuk domain nyu.edu

Pada contoh di Figure 4.2, kita telah memanggil nslookup dengan opsi “-type=NS” dan domain “nyu.edu”. Ini menyebabkan nslookup mengirim kueri untuk catatan tipe-NS ke server DNS lokal default. Dengan kata lain, kueri tersebut mengatakan, “tolong kirimkan saya nama host dari DNS otoritatif untuk nyu.edu”. (Bila opsi –type tidak digunakan, nslookup menggunakan default, yaitu untuk menanyakan record tipe A.) Jawabannya, ditampilkan pada screenshot di atas, pertama-tama menunjukkan server DNS yang memberikan jawaban (yang merupakan default lokal Server DNS UMass dengan alamat 128.119.240.1) bersama dengan tiga server nama DNS NYU.

Masing-masing server ini memang merupakan server DNS otoritatif untuk host di kampus NYU. Namun, nslookup juga menunjukkan bahwa jawabannya adalah “non-otoritatif”, artinya jawaban ini berasal dari cache beberapa server, bukan dari server DNS NYU yang berwenang. Akhirnya, jawabannya juga termasuk alamat IP dari server DNS otoritatif di NYU. (Meskipun kueri tipe-NS yang

dihasilkan oleh nslookup tidak secara eksplisit meminta alamat IP, server DNS lokal mengembalikan ini "gratis" dan nslookup menampilkan hasilnya.)

nslookup memiliki sejumlah opsi tambahan selain "-type=NS" yang mungkin ingin Anda jelajahi. Berikut adalah situs dengan tangkapan layar dari sepuluh penggunaan nslookup yang populer: <https://www.cloudns.net/blog/10-most-used-nslookup-commands/> dan berikut adalah "halaman manual" untuk nslookup: <https://linux.die.net/man/1/nslookup>.

Terakhir, terkadang kita mungkin tertarik untuk menemukan nama host yang terkait dengan alamat IP tertentu, yaitu kebalikan dari pencarian yang ditunjukkan pada Figure 4.1 (di mana nama host diketahui/ditentukan dan alamat IP host dikembalikan). nslookup juga dapat digunakan untuk melakukan apa yang disebut "pencarian DNS terbalik". Pada Figure 4.3, misalnya, kita menetapkan alamat IP sebagai argumen nslookup (128.119.245.12 dalam contoh ini) dan nslookup mengembalikan nama host dengan alamat tersebut (gaia.cs.umass.edu dalam contoh ini)

```
[kurose@MacBook-Pro-6 ~ % nslookup 128.119.245.12
Server:      75.75.75.75
Address:     75.75.75.75#53

Non-authoritative answer:
12.245.119.128.in-addr.arpa    name = gaia.cs.umass.edu.

Authoritative answers can be found from:
```

Figure 4.3 menggunakan nslookup untuk melakukan reverse DNS lookup

```
[kurose@MacBook-Pro-6 ~ % nslookup www.iitb.ac.in
Server:      75.75.75.75
Address:     75.75.75.75#53

Non-authoritative answer:
Name:   www.iitb.ac.in
Address: 103.21.124.10

[kurose@MacBook-Pro-6 ~ % nslookup -type=NS iitb.ac.in
Server:      75.75.75.75
Address:     75.75.75.75#53

Non-authoritative answer:
iitb.ac.in      nameserver = dns1.iitb.ac.in.
iitb.ac.in      nameserver = dns2.iitb.ac.in.
iitb.ac.in      nameserver = dns3.iitb.ac.in.
```

Figure 4.4 using nslookup to find the IP address of www.iitb.ac.in and the names of the

### 4.3 DNS cache pada Komputer Anda

Dari deskripsi resolusi kueri DNS iteratif dan rekursif, Anda mungkin berpikir bahwa server DNS lokal harus dihubungi setiap waktu aplikasi perlu menerjemahkan dari nama host ke alamat IP. Itu tidak selalu benar dalam praktik!

Sebagian besar host (misalnya, komputer pribadi Anda) menyimpan cache dari catatan DNS yang baru saja diambil (kadang-kadang disebut cache resolver DNS), sama seperti banyak browser Web

menyimpan cache objek yang baru saja diambil oleh HTTP. Ketika layanan DNS perlu dipanggil oleh sebuah host, host tersebut akan terlebih dahulu memeriksa apakah catatan DNS yang diperlukan ada di cache DNS host ini; jika catatan ditemukan, tuan rumah bahkan tidak akan repot-repot menghubungi server DNS lokal dan sebagai gantinya akan menggunakan catatan DNS yang di-cache ini.

Anda juga dapat secara eksplisit menghapus catatan dalam cache DNS Anda. Itu berarti bahwa komputer Anda perlu memanggil layanan DNS terdistribusi saat berikutnya perlu menggunakan layanan resolusi nama DNS, karena tidak akan menemukan catatan dalam cache. Pada komputer Mac, Anda dapat memasukkan perintah berikut ke jendela terminal untuk menghapus cache DNS resolver Anda:

```
sudo killall -HUP mDNSResponder
```

Pada komputer Windows Anda dapat memasukkan perintah berikut di command prompt:

```
ipconfig /flushdns
```

and on a Linux computer, enter:

```
sudo systemd-resolve --flush-caches
```

#### 4.4 Tracing DNS dengan Wireshark

Sekarang kita sudah familiar dengan nslookup dan membersihkan cache DNS resolver. Pertama-tama mari kita tangkap pesan DNS yang dihasilkan oleh aktivitas penjelajahan web biasa.

- Kosongkan cache DNS di host Anda, seperti dijelaskan di atas.
- Buka browser Web Anda dan bersihkan cache browser Anda.
- Buka Wireshark dan masukkan ip.addr == <your\_IP\_address> ke dalam filter tampilan, di mana <your\_IP\_address> adalah alamat IPv4 komputer Anda. Dengan filter ini, Wireshark hanya akan menampilkan paket yang berasal dari, atau ditujukan ke, host Anda.
- Mulai pengambilan paket di Wireshark.
- Dengan browser Anda, kunjungi halaman Web: [http://gaia.cs.umass.edu/kurose\\_ross/](http://gaia.cs.umass.edu/kurose_ross/)
- Hentikan pengambilan paket.

Jika Anda tidak dapat menjalankan Wireshark pada koneksi jaringan langsung, Anda dapat mengunduh file pelacakan paket yang diambil saat mengikuti langkah-langkah di atas di salah satu komputer pembuatnya.

# Modul 5 UDP

## Tujuan Praktikum

1. Mahasiswa dapat menginvestigasi cara kerja protokol UDP menggunakan Wireshark

## 5.1 Pengantar

Di modul ini, kita akan melihat sekilas protokol transport UDP. **UDP adalah** protokol yang disederhanakan dan tanpa embel-embel.

Pada tahap ini, Anda harus sudah bisa menggunakan Wireshark. Jadi, kami tidak akan menjelaskan langkah-langkah secara eksplisit seperti di modul sebelumnya. Secara khusus, kami tidak akan memberikan contoh tangkapan layar untuk semua langkah.

## 5.2 Tugas

Mulailah menangkap paket di Wireshark dan kemudian lakukan sesuatu yang akan menyebabkan host Anda mengirim dan menerima beberapa paket UDP. Kemungkinan juga hanya dengan tidak melakukan apa-apa (kecuali menangkap paket melalui Wireshark) beberapa paket UDP yang dikirim oleh orang lain akan muncul di trace Anda. Secara khusus, protokol Domain Name System (DNS) biasanya mengirimkan permintaan DNS dan pesan respons di dalam UDP, jadi kemungkinan Anda akan menemukan beberapa pesan DNS (dan karenanya paket UDP) di trace Anda.

Secara khusus Anda dapat mencoba perintah nslookup, yang memanggil protokol DNS yang mendasarinya, yang pada gilirannya akan mengirim segmen UDP dari/ke host yang mengeluarkan nslookup. nslookup tersedia di sebagian besar sistem operasi Microsoft, Apple IOS, dan Linux. Untuk menjalankan nslookup Anda cukup mengetikkan perintah nslookup pada baris perintah di jendela DOS, jendela terminal Mac IOS, atau shell Linux.

Figure 5.1 adalah screenshot menjalankan nslookup pada baris perintah Linux pada host newworld.cs.umass.edu yang terletak di Departemen CS di kampus University of Massachusetts (UMass), untuk menampilkan alamat IP www.nyu.edu.

```
[newworld.cs.umass.edu]# nslookup www.nyu.edu
Server:          128.119.240.1
Address:         128.119.240.1#53

Non-authoritative answer:
www.nyu.edu      canonical name = WEB.GSLB.nyu.edu.
Name:             WEB.GSLB.nyu.edu
Address:          216.165.47.12
Name:             WEB.GSLB.nyu.edu
Address:          2607:f600:1002:6113::100
```

Figure 5.1 dasar nslookup command

Setelah memulai pengambilan paket di Wireshark, jalankan nslookup untuk nama host yang sudah lama tidak Anda kunjungi. Kemudian hentikan pengambilan paket, atur filter paket Wireshark Anda sehingga Wireshark hanya menampilkan segmen UDP yang dikirim dan diterima di host Anda. Pilih segmen UDP pertama dan perluas bidang UDP di jendela detail. Jika Anda tidak dapat menemukan

segmen UDP dalam pelacakan Anda atau tidak dapat menjalankan Wireshark pada koneksi jaringan langsung, Anda dapat mengunduh trace paket yang berisi beberapa segmen UDP.



Fakultas Informatika  
School of Computing  
Telkom University



# Modul 6 TCP

## Tujuan Praktikum

1. Mahasiswa dapat menginvestigasi cara kerja protokol TCP menggunakan Wireshark

## 6.1 Pengantar

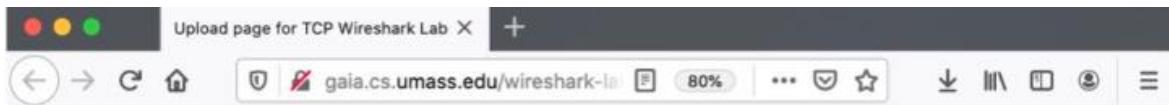
Di modul ini, kita akan menyelidiki perilaku protokol TCP secara mendetail. Kita akan melakukannya dengan menganalisis jejak segmen TCP yang dikirim dan diterima dalam mentransfer file 150KB (berisi teks Alice's Adventures in Wonderland karya Lewis Carroll) dari komputer Anda ke server jauh. Kami akan mempelajari penggunaan nomor urut dan pengakuan TCP untuk menyediakan transfer data yang andal; kita akan melihat algoritma congestion control TCP – mulai lambat dan congestion avoidance; dan kita akan melihat mekanisme kontrol aliran yang ditawarkan penerima TCP. Kami juga akan secara singkat mempertimbangkan penyiapan koneksi TCP dan kami akan menyelidiki kinerja (throughput dan waktu pulang pergi) koneksi TCP antara komputer Anda dan server.

## 6.2 Menangkap Transfer TCP dalam Jumlah Besar dari Komputer Pribadi ke Remote Server

Sebelum memulai mempelajari TCP, kita harus menggunakan Wireshark untuk mendapatkan trace paket transfer TCP file dari komputer Anda ke remote server. Anda akan melakukannya dengan mengakses halaman Web yang memungkinkan Anda memasukkan nama file yang disimpan di komputer Anda (yang berisi teks ASCII Alice in Wonderland), dan kemudian mentransfer file ke server Web menggunakan HTTP POST metod. Kita menggunakan metode POST daripada metod GET karena kita ingin mentransfer sejumlah besar data dari komputer Anda ke komputer lain. Tentu saja, kita akan menjalankan Wireshark selama proses ini untuk mendapatkan trace segmen TCP yang dikirim dan diterima dari komputer Anda.

Lakukan hal berikut:

- Mulai browser web Anda. Buka <http://gaia.cs.umass.edu/wireshark-labs/alice.txt> dan ambil salinan ASCII Alice in Wonderland. Simpan ini sebagai file .txt di suatu tempat di komputer Anda.
- Selanjutnya buka <http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html>.
- Anda akan melihat layar seperti Figure 6.1.



Upload page for TCP Wireshark Lab  
Computer Networking: A Top Down Approach, 6th edition  
Copyright 2012 J.F. Kurose and K.W. Ross, All Rights Reserved

If you have followed the instructions for the TCP Wireshark Lab, you have already downloaded an ASCII copy of Alice and Wonderland from <http://gaia.cs.umass.edu/wireshark-labs/alice.txt> and you also already have the Wireshark packet sniffer running and capturing packets on your computer.

Click on the Browse button below to select the directory/file name for the copy of alice.txt that is stored on your computer.

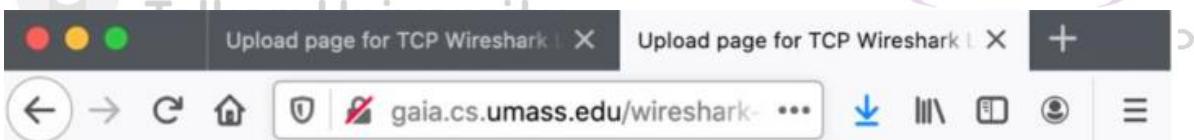
Browse... No file selected.

Once you have selected the file, click on the "Upload alice.txt file" button below. This will cause your browser to send a copy of alice.txt over an HTTP connection (using TCP) to the web server at gaia.cs.umass.edu. After clicking on the button, wait until a short message is displayed indicating the the upload is complete. Then stop your Wireshark packet sniffer - you're ready to begin analyzing the TCP transfer of alice.txt from your computer to gaia.cs.umass.edu!!

Upload alice.txt file

Figure 6.1 halaman untuk upload file alice.txt dari komputer anda ke gaia.cs.umass.edu

- Gunakan tombol Browse dalam formulir ini ke file di komputer Anda yang baru saja Anda buat yang berisi Alice in Wonderland. Jangan tekan tombol "Unggah file alice.txt".
- Sekarang jalankan Wireshark dan mulai pengambilan paket.
- Kembali ke browser Anda, tekan tombol "Upload file alice.txt" untuk mengupload file ke server gaia.cs.umass.edu. Setelah file diunggah, pesan ucapan selamat singkat akan ditampilkan di jendela browser Anda.
- Hentikan pengambilan paket Wireshark. Jendela Wireshark Anda akan terlihat mirip dengan jendela yang ditunjukkan pada Figure 6.2



## Congratulations!

You've now transferred a copy of alice.txt from your computer to gaia.cs.umass.edu. You should now stop Wireshark packet capture. It's time to start analyzing the captured Wireshark packets!

Figure 6.2 Success! Kamu sudah mengupload file gaia.cs.umass.edu

Jika Anda tidak dapat menjalankan Wireshark pada koneksi jaringan langsung, Anda dapat mengunduh jejak paket yang diambil saat mengikuti langkah-langkah di atas pada salah satu komputer. Selain itu, Anda mungkin merasa perlu mengunduh trace ini meskipun Anda telah menangkap trace Anda sendiri dan menggunakan

### 6.3 Tampilan Awal pada Captured Trace

Sebelum menganalisis perilaku koneksi TCP secara rinci, mari kita lihat trace tingkat tinggi.

Mari kita mulai dengan melihat pesan HTTP POST yang mengunggah file alice.txt ke gaia.cs.umass.edu dari komputer Anda. Temukan file itu di jejak Wireshark Anda, dan perluas pesan

HTTP sehingga kita dapat melihat pesan HTTP POST lebih hati-hati. Layar Wireshark Anda akan terlihat seperti Figure 6.3.

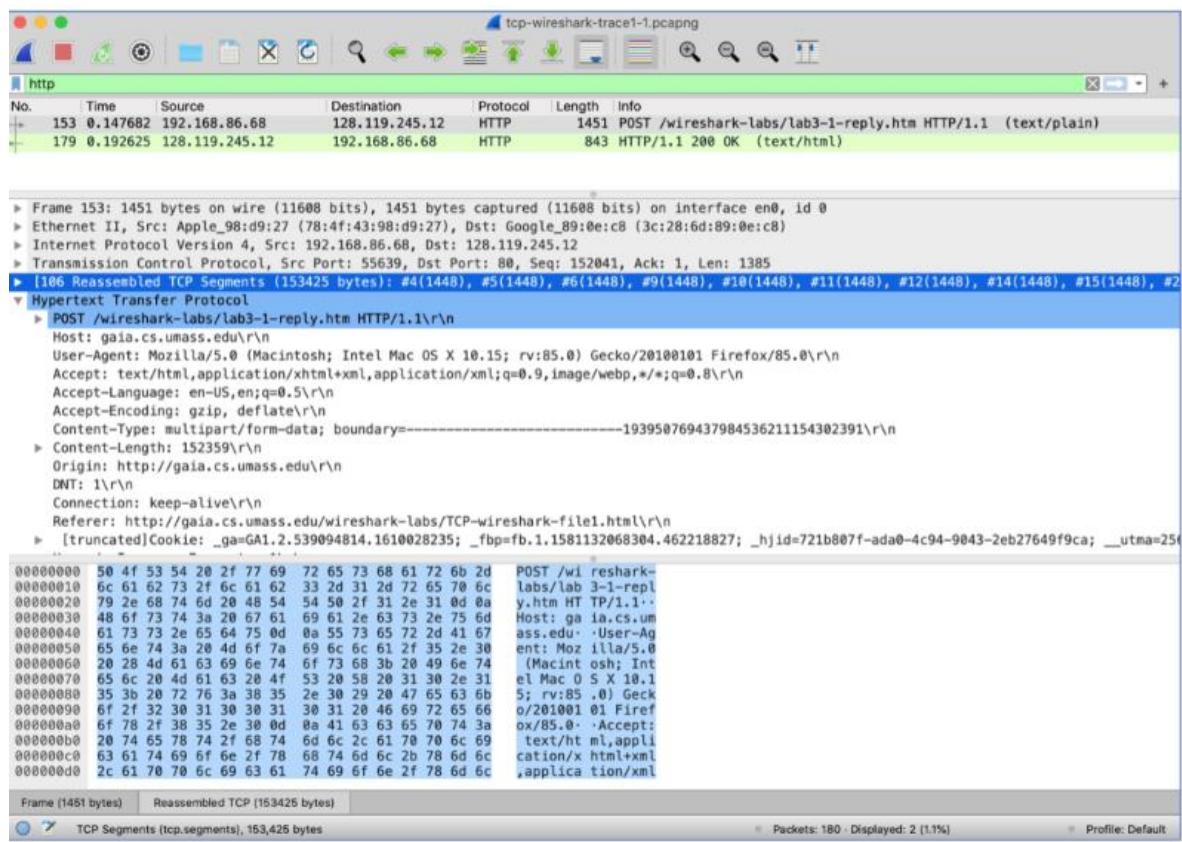


Figure 6.3 penjabaran pesan HTTP POST di upload dari file alice.txt dari komputer anda ke gaia.cs.umass.edu

Ada beberapa hal yang perlu diperhatikan di sini:

- Isi pesan HTTP POST application-layer Anda berisi konten file alice.txt, yang merupakan file besar lebih dari 152K byte. OK – tidak terlalu besar, tetapi akan terlalu besar untuk pesan HTTP POST yang satu ini untuk dimuat hanya dalam satu segmen TCP!
- Faktanya, seperti yang ditunjukkan pada jendela Wireshark pada Figure 6.3 kita melihat bahwa pesan HTTP POST tersebar di 106 segmen TCP. Ini ditunjukkan di mana panah merah ditempatkan pada Figure 6.3 Jika Anda melihat lebih teliti di sana, Anda dapat melihat bahwa Wireshark juga sangat membantu Anda, memberi tahu Anda bahwa segmen TCP pertama yang berisi awal dari pesan POST adalah paket #4 dalam trace tertentu untuk contoh di Figure 6.3 , yang merupakan trace tcp-wireshark-trace1-1. Segmen TCP kedua yang berisi pesan POST dalam paket #5 dalam trace, dan seterusnya.

Sekarang mari kita melihat beberapa segmen TCP.

- Pertama, filter paket yang ditampilkan di jendela Wireshark dengan memasukkan “tcp” ke jendela spesifikasi filter tampilan di bagian atas jendela Wireshark. Tampilan Wireshark Anda akan terlihat seperti Figure 6.4. Pada Figure 6.4, kami telah mencatat segmen TCP yang memiliki bit SYN yang ditetapkan – ini adalah pesan TCP pertama dalam jabat tangan tiga arah yang mengatur koneksi TCP ke gaia.cs.umass.edu yang pada akhirnya akan membawa HTTP POST pesan dan file alice.txt. Kami juga telah mencatat segmen SYNACK

(langkah kedua dalam three-way handshake TCP), serta segmen TCP (paket #4, seperti yang dibahas di atas) yang membawa pesan POST dan awal file alice.txt. Tentu saja, jika Anda mengambil file trace Anda sendiri, nomor paket akan berbeda, tetapi Anda akan melihat perilaku yang serupa dengan yang ditunjukkan pada Figure 6.3 dan 6.4.

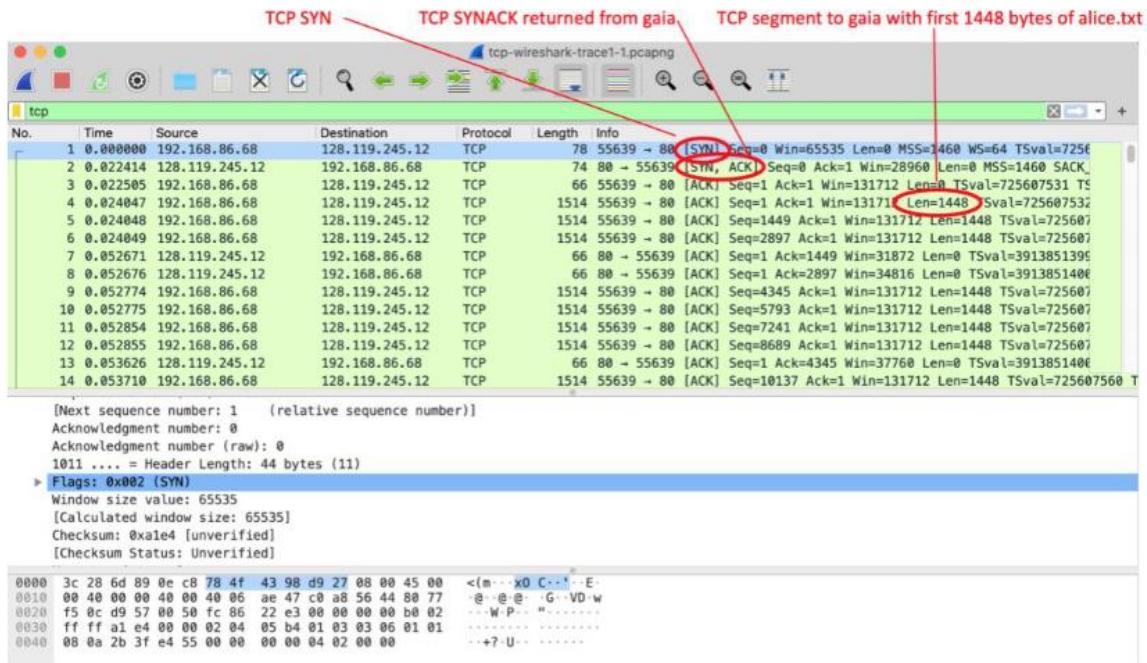


Figure 6.4 TCP segments yang terlibat dalam pengiriman pesan HTTP POST (termasuk file alice.txt) ke gaia.cs.umass.edu

Karena modul ini tentang TCP daripada HTTP, sekarang ubah jendela "daftar paket yang diambil" Wireshark sehingga menampilkan informasi tentang segmen TCP yang berisi pesan HTTP, bukan tentang pesan HTTP, seperti pada Figure 6.4 di atas. Inilah yang kita cari—serangkaian segmen TCP yang dikirim antara komputer Anda dan gaia.cs.umass.edu!

## 6.4 TCP Congestion Control

Sekarang mari kita periksa jumlah data yang dikirim per satuan waktu dari klien ke server. Kita akan menggunakan salah satu utilitas grafik TCP Wireshark-Time-Sequence-Graph(Stevens)—untuk memplot data.

- Pilih segmen TCP yang dikirim klien di jendela "daftar paket yang diambil" Wireshark yang sesuai dengan transfer alice.txt dari klien ke gaia.cs.umass.edu. Kemudian pilih menu: Statistics->TCP Stream Graph-> Time-Sequence-Graph(Stevens8). Anda akan melihat plot yang terlihat mirip dengan plot pada Figure 6.5, yang dibuat dari paket yang ditangkap di trace paket tcp-wireshark-trace1-1. Anda mungkin harus memperluas, mengecilkan, dan mengutak-atik interval yang ditunjukkan pada sumbu agar grafik Anda terlihat seperti Figure 6.5.

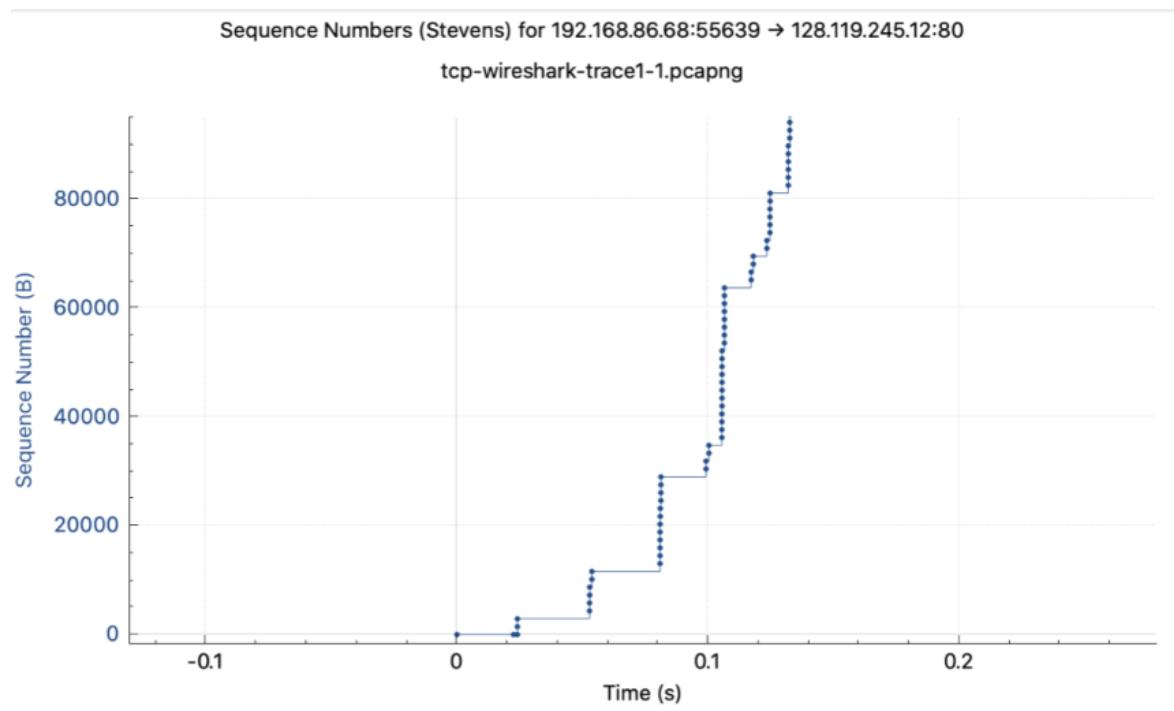
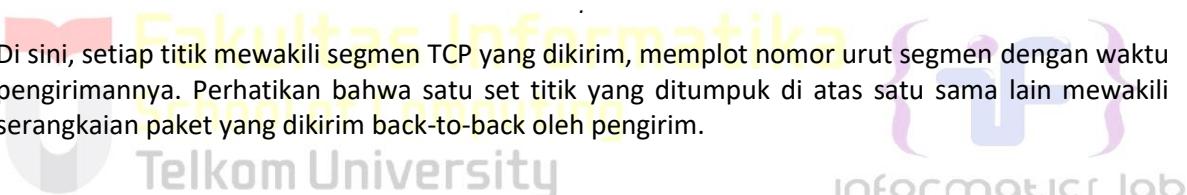


Figure 6.5 A sequence-number-versus-time plot (Stevens format) dari TCP segments

Di sini, setiap titik mewakili segmen TCP yang dikirim, memplot nomor urut segmen dengan waktu pengirimannya. Perhatikan bahwa satu set titik yang ditumpuk di atas satu sama lain mewakili serangkaian paket yang dikirim back-to-back oleh pengirim.



# Modul 7 SOCKET PROGRAMMING: MEMBUAT APLIKASI JARINGAN

## Tujuan Praktikum

1. Mahasiswa bisa membuat program berbasis socket UDP
2. Mahasiswa bisa membuat program berbasis socket TCP

### 7.1 Pengantar

Sekarang setelah kita melihat sejumlah aplikasi jaringan yang penting, mari kita pelajari bagaimana program aplikasi jaringan sebenarnya dibuat. *typical network application* terdiri dari sepasang program—program klien dan program server—yang berada di dua sistem akhir yang berbeda. Ketika kedua program ini dijalankan, proses klien dan proses server dibuat, dan proses ini berkomunikasi satu sama lain dengan **membaca dari, dan menulis ke, soket**. Saat membuat aplikasi jaringan, tugas utama developer adalah menulis kode untuk program klien dan server.

Ada dua jenis network applications. Pertama adalah **implementasi** yang operasinya ditentukan dalam standar protokol, seperti RFC atau beberapa dokumen standar lainnya; aplikasi semacam itu kadang-kadang disebut sebagai "terbuka", karena aturan yang menentukan operasinya diketahui semua orang. Untuk implementasi seperti itu, program klien dan server harus sesuai dengan aturan yang ditentukan oleh RFC. Sebagai contoh, program klien dapat menjadi implementasi dari sisi klien dari protokol HTTP dan didefinisikan secara tepat dalam RFC 2616; program server dapat menjadi implementasi dari protokol server HTTP, juga didefinisikan secara tepat dalam RFC 2616. Jika satu developer menulis kode untuk program klien dan developer lain menulis kode untuk program server, dan kedua developer dengan hati-hati mengikuti aturan dari RFC, maka kedua program akan dapat saling beroperasi. Memang, banyak aplikasi jaringan saat ini melibatkan komunikasi antara program klien dan server yang telah dibuat oleh developer independen—misalnya, browser Google Chrome yang berkomunikasi dengan server Web Apache, atau klien BitTorrent yang berkomunikasi dengan pelacak BitTorrent.

Jenis jaringan lainnya aplikasi adalah **proprietary network application**. Dalam hal ini, program klien dan server menggunakan protokol an application-layer yang belum dipublikasikan secara terbuka di RFC atau di tempat lain. developer membuat program klien dan server, dan developer memiliki kendali penuh atas apa yang ada dalam kode. Tetapi karena kode tersebut tidak mengimplementasikan *open protocol*, developer independen lainnya tidak akan dapat mengembangkan kode yang beroperasi dengan aplikasi tersebut. Anda akan merusak kode yang mengimplementasikan aplikasi client-server yang sangat sederhana.

Selama fase pengembangan, salah satu keputusan pertama yang harus dibuat pengembang adalah apakah aplikasi akan dijalankan di atas TCP atau di atas UDP. Ingatlah bahwa **TCP** berorientasi pada koneksi dan menyediakan saluran aliran byte yang andal di mana data mengalir di antara dua sistem akhir. **UDP** tidak memiliki koneksi dan mengirimkan paket data independen dari satu sistem ujung ke ujung lainnya, tanpa jaminan tentang pengiriman. Ingat juga bahwa ketika program klien atau server mengimplementasikan protokol yang ditentukan oleh RFC, itu harus menggunakan nomor port terkenal yang terkait dengan protokol; sebaliknya, ketika mengembangkan *proprietary application*, developer harus berhati-hati untuk menghindari penggunaan nomor port yang terkenal seperti itu.

Kami memperkenalkan pemrograman soket UDP dan TCP melalui aplikasi UDP sederhana dan aplikasi TCP sederhana. Kami menyajikan aplikasi UDP dan TCP sederhana dalam Python 3. Kami dapat menulis kode dalam Java, C, atau C++, tetapi kami memilih Python terutama karena Python dengan jelas memaparkan konsep soket key. Dengan Python ada lebih sedikit baris kode, dan setiap baris dapat dijelaskan kepada programmer pemula tanpa kesulitan.

## 7.2 Program Socket dengan UDP

Di bagian ini, kita akan menulis program client-server sederhana yang menggunakan UDP dan menulis program serupa yang menggunakan TCP. Proses yang berjalan pada mesin yang berbeda berkomunikasi satu sama lain dengan mengirimkan pesan ke dalam soket. Kita mengatakan bahwa setiap proses dianalogikan dengan sebuah rumah dan soket proses dianalogikan dengan sebuah pintu. Aplikasi berada di satu sisi pintu di rumah; protokol transport-layer berada di sisi lain pintu di dunia luar. Developer aplikasi memiliki kendali atas segala sesuatu di sisi lapisan aplikasi soket; namun, ia memiliki sedikit kontrol dari sisi transport-layer.

Sekarang mari kita lihat lebih dekat interaksi antara dua proses komunikasi yang menggunakan soket UDP. Sebelum proses pengiriman dapat mendorong paket data keluar dari pintu soket, saat menggunakan UDP, terlebih dahulu harus melampirkan **alamat tujuan** ke paket. Setelah paket melewati soket pengirim, Internet akan menggunakan alamat tujuan ini untuk merutekan paket melalui Internet ke soket dalam proses penerima. Ketika paket tiba di soket penerima, proses penerima akan mengambil paket melalui soket, dan kemudian memeriksa isi paket dan mengambil tindakan yang tepat.

Alamat IP host tujuan adalah bagian dari alamat tujuan. Dengan memasukkan alamat IP tujuan dalam paket, router di Internet akan dapat merutekan paket melalui Internet ke host tujuan. Tetapi karena sebuah host mungkin menjalankan banyak proses aplikasi jaringan, masing-masing dengan satu atau lebih soket, maka perlu juga untuk mengidentifikasi soket tertentu di host tujuan. Ketika soket dibuat, pengidentifikasi, yang disebut nomor port, diberikan padanya. Alamat tujuan paket juga menyertakan nomor port soket. Singkatnya, proses pengiriman melampirkan ke paket alamat tujuan, yang terdiri dari alamat IP host tujuan dan nomor port soket tujuan. Selain itu, alamat sumber pengirim—terdiri dari alamat IP host sumber dan nomor port soket sumber—juga dilampirkan ke paket. Namun, melampirkan alamat sumber ke paket biasanya tidak dilakukan oleh kode aplikasi UDP; alih-alih dilakukan secara otomatis oleh sistem operasi yang mendasarinya.

Kita akan menggunakan aplikasi client-server sederhana berikut untuk mendemonstrasikan pemrograman soket untuk UDP dan TCP:

1. Klien membaca sebaris karakter (data) dari keyboardnya dan mengirimkan datanya ke server.
2. Server menerima data dan mengubah karakter menjadi huruf besar.
3. Server mengirimkan data yang dimodifikasi ke klien.
4. Klien menerima data yang dimodifikasi dan menampilkan baris pada layarnya.

Gambar 7.1 menyoroti aktivitas utama yang berhubungan dengan soket dari klien dan server yang berkomunikasi melalui layanan transportasi UDP.

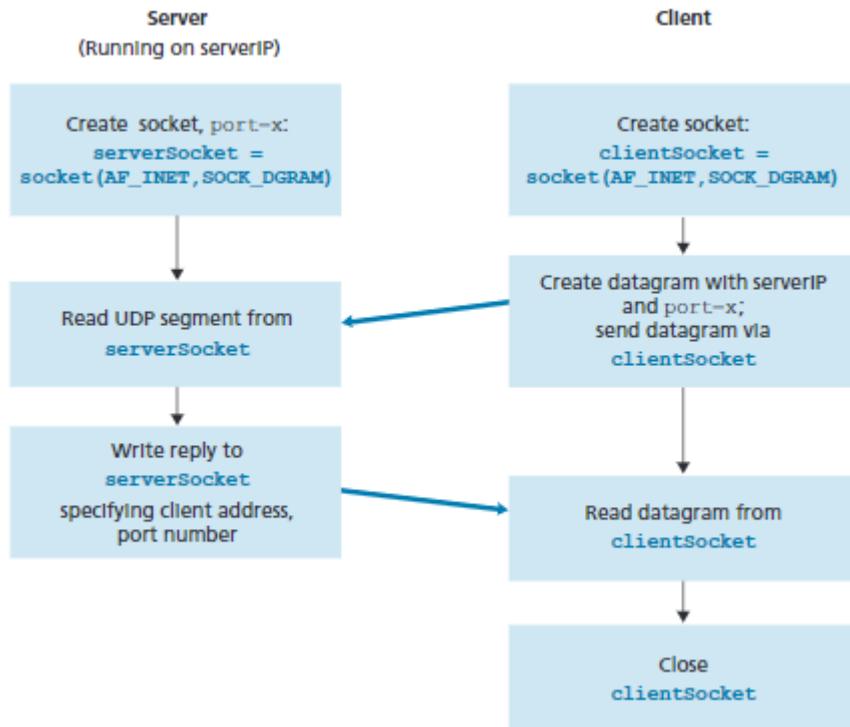


Figure 7.1 client-server application menggunakan UDP

Sekarang lihat pasangan program client-server untuk implementasi UDP dari aplikasi sederhana ini. Kami juga menyediakan analisis baris demi baris yang terperinci setelah setiap program. Kami akan mulai dengan klien UDP, yang akan mengirim pesan tingkat aplikasi sederhana ke server. Agar server dapat menerima dan membalas pesan klien, itu harus siap dan berjalan—yaitu, itu harus berjalan sebagai proses sebelum klien mengirim pesannya.

**Program klien disebut UDPClient.py, dan program server disebut UDPServer.py.** Untuk menekankan isu-isu kunci, kami sengaja memberikan kode yang minimal. "Kode yang baik" pasti akan memiliki beberapa baris tambahan, khususnya untuk menangani kasus kesalahan. Untuk aplikasi ini, kami memilih 12000 untuk nomor port server.

### 7.2.1 UDPClient.py

Berikut adalah kode untuk sisi klien aplikasi:

```

from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = input('Input lowercase sentence:')
clientSocket.sendto(message.encode(), (serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()

```

Figure 7.2 UDPClient.py

Sekarang mari kita lihat berbagai baris kode di UDPClient.py.

```
from socket import *
```

Modul soket membentuk dasar dari semua komunikasi jaringan dengan Python. Dengan memasukkan baris ini, kita akan dapat membuat soket dalam program kita.

```
serverName = 'hostname'  
serverPort = 12000
```

Baris pertama menetapkan variabel serverName ke string 'hostname'. Di sini, kita menyediakan string yang berisi alamat IP server (contohnya "128.138.32.126") atau nama host server (contohnya "cis.poly.edu"). Jika kita menggunakan nama host, maka pencarian DNS akan secara otomatis dilakukan untuk mendapatkan alamat IP. Baris kedua mengatur serverPort variabel integer ke 12000.

```
clientSocket = socket(AF_INET, SOCK_DGRAM)
```

Baris ini membuat soket klien, yang disebut clientSocket. Parameter pertama menunjukkan keluarga alamat; khususnya, AF\_INET menunjukkan bahwa jaringan yang mendasarinya menggunakan IPv4. Parameter kedua menunjukkan bahwa soket bertipe SOCK\_DGRAM, yang berarti soket tersebut adalah soket UDP (bukan soket TCP).

Perhatikan bahwa kita tidak menentukan nomor port soket klien saat kita membuatnya; kita malah membiarkan sistem operasi melakukan ini untuk kita. Sekarang pintu proses klien telah dibuat, kami ingin membuat pesan untuk dikirim melalui pintu.

```
message = input('Input lowercase sentence:')
```

input() adalah fungsi bawaan dalam Python. Ketika perintah ini dijalankan, pengguna di klien diminta dengan kata-kata "Masukkan kalimat huruf kecil:" Pengguna kemudian menggunakan keyboardnya untuk memasukkan inputan yang dimasukkan ke dalam variabel message. Sekarang kita memiliki soket dan pesan, kita ingin mengirim pesan melalui soket ke host tujuan.

```
clientSocket.sendto(message.encode(), (serverName, serverPort))
```

Pada baris di atas, pertama-tama kita mengubah pesan dari tipe string ke tipe byte, karena kita perlu mengirim byte ke dalam soket; ini dilakukan dengan metode encode(). Metode sendto() melampirkan alamat tujuan (serverName, serverPort) ke pesan dan mengirimkan paket yang dihasilkan ke soket proses, clientSocket. Setelah mengirim paket, klien menunggu untuk menerima data dari server.

```
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
```

Dengan baris di atas, ketika sebuah paket tiba dari Internet di soket klien, data paket tersebut dimasukkan ke dalam variabel modifiedMessage dan alamat sumber paket tersebut dimasukkan ke dalam variabel serverAddress.

Variabel serverAddress berisi alamat IP server dan nomor port server. Program UDPClient sebenarnya tidak memerlukan informasi alamat server ini, karena program ini sudah mengetahui alamat server sejak awal; tetapi baris Python ini memberikan alamat server. Metod recvfrom juga mengambil ukuran buffer 2048 sebagai input. (Ukuran buffer ini berfungsi untuk sebagian besar tujuan.)

```
print(modifiedMessage.decode())
```

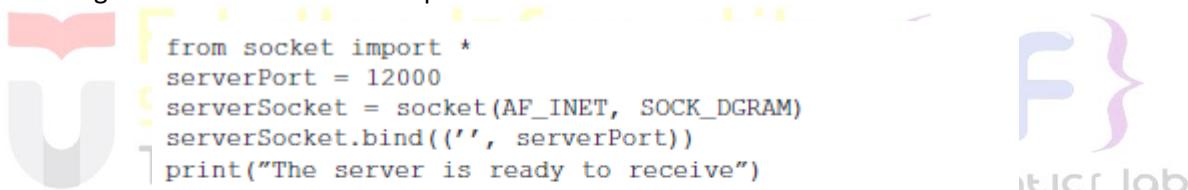
Baris ini mencetak ModifiedMessage pada tampilan pengguna, setelah mengubah pesan dari byte menjadi string. Seharusnya baris asli yang diketik pengguna, tetapi sekarang menggunakan huruf kapital.

```
clientSocket.close()
```

Baris ini menutup soket. Proses kemudian berakhir.

### 7.2.2 UDPServer.py

Sekarang mari kita lihat sisi server aplikasi:



```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print("The server is ready to receive")
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(),
    clientAddress)
```

Figure 7.3 UDPServer.py

Perhatikan bahwa awal UDPServer mirip dengan UDPClient. Itu juga mengimpor modul soket, juga menetapkan serverPort variabel integer ke 12000, dan juga membuat soket tipe SOCK\_DGRAM (soket UDP). Baris kode pertama yang secara signifikan berbeda dari UDPClient adalah:

```
serverSocket.bind(('', serverPort))
```

Baris di atas mengikat (yaitu, menetapkan) nomor port 12000 ke soket server. Jadi, di UDPServer, kode (ditulis oleh developer aplikasi) secara eksplisit menetapkan nomor port ke soket. Dengan cara ini, ketika seseorang mengirim paket ke port 12000 di alamat IP server, paket itu akan diarahkan ke soket ini. UDPServer kemudian memasuki while loop yang akan memungkinkan UDPServer untuk menerima dan memproses paket dari klien tanpa batas. Dalam while loop, UDPServer menunggu paket tiba.

```
message, clientAddress = serverSocket.recvfrom(2048)
```

Baris kode ini mirip dengan apa yang kita lihat di UDPClient. Ketika sebuah paket tiba di soket server, data paket dimasukkan ke dalam pesan variabel dan alamat sumber paket dimasukkan ke dalam variabel clientAddress. Variabel clientAddress berisi alamat IP klien dan nomor port klien. Di sini, UDPServer akan menggunakan informasi alamat ini, karena memberikan alamat pengirim, mirip dengan alamat pengirim pada surat pos biasa. Dengan informasi alamat sumber ini, server sekarang tahu ke mana ia harus mengarahkan balasannya.

```
modifiedMessage = message.decode().upper()
```

Baris ini adalah inti dari aplikasi sederhana kita. Dibutuhkan baris yang dikirim oleh klien dan, setelah mengonversi pesan menjadi string, menggunakan metode upper() untuk menggunakan huruf kapital.

```
serverSocket.sendto(modifiedMessage.encode(), clientAddress)
```

Baris terakhir ini melampirkan alamat klien (alamat IP dan nomor port) ke pesan dengan huruf kapital (setelah mengubah string menjadi byte), dan mengirimkan paket yang dihasilkan ke soket server. (Seperti yang disebutkan sebelumnya, alamat server juga dilampirkan ke paket, meskipun ini dilakukan secara otomatis daripada secara eksplisit oleh kode.) Internet kemudian akan mengirimkan paket ke alamat klien ini. Setelah server mengirim paket, paket tetap berada di loop while, menunggu paket UDP lain tiba (dari klien mana pun yang berjalan di host mana pun). Untuk menguji pasangan program, Anda menjalankan UDPClient.py pada satu host dan UDPServer.py di host lain. Pastikan untuk menyertakan nama host atau alamat IP server yang tepat di UDPClient.py.

Selanjutnya, Anda menjalankan UDPServer.py, program server terkompilasi, di host server. Ini menciptakan proses di server yang menganggur sampai dihubungi oleh beberapa klien. Kemudian Anda menjalankan UDPClient.py, program klien yang dikompilasi, di klien. Ini menciptakan proses di klien.

Terakhir, untuk menggunakan aplikasi di klien, Anda mengetikkan kalimat diikuti dengan carriage return. Untuk mengembangkan aplikasi server klien UDP Anda sendiri, Anda dapat mulai dengan sedikit memodifikasi program klien atau server. Misalnya, alih-alih mengubah semua huruf menjadi huruf besar, server dapat menghitung berapa kali huruf s muncul dan mengembalikan nomor ini. Atau Anda dapat memodifikasi klien sehingga setelah menerima kalimat dengan huruf kapital, pengguna dapat terus mengirim lebih banyak kalimat ke server.

### 7.3 Program Socket dengan TCP

Tidak seperti UDP, TCP adalah protokol berorientasi koneksi. Ini berarti bahwa sebelum klien dan server dapat mulai mengirim data satu sama lain, mereka harus terlebih dahulu handshake dan membuat koneksi TCP. Salah satu ujung koneksi TCP terpasang ke soket klien dan ujung lainnya terpasang ke soket server. Saat membuat koneksi TCP, kita mengaitkannya dengan alamat soket klien (alamat IP dan nomor port) dan alamat soket server (alamat IP dan nomor port). Dengan

koneksi TCP dibangun, ketika satu sisi ingin mengirim data ke sisi lain, itu hanya memasukkan data ke dalam koneksi TCP melalui soketnya.

Ini berbeda dengan UDP, di mana server harus melampirkan alamat tujuan ke paket sebelum memasukkannya ke dalam soket. Sekarang mari kita lihat lebih dekat interaksi program klien dan server di TCP. Klien memiliki tugas memulai kontak dengan server. Agar server dapat bereaksi terhadap kontak awal klien, server harus siap. Ini menyiratkan dua hal. Pertama, seperti dalam kasus UDP, server TCP harus berjalan sebagai proses sebelum klien mencoba untuk memulai kontak. Kedua, program server harus memiliki pintu khusus—lebih tepatnya, soket khusus—yang menyambut beberapa kontak awal dari proses klien yang berjalan pada host arbitrer. Menggunakan analogi rumah/pintu kita untuk proses/soket, terkadang kita akan merujuk ke kontak awal klien sebagai “mengetuk pintu penyambutan.” Dengan proses server berjalan, proses klien dapat memulai koneksi TCP ke server. Ini dilakukan dalam program klien dengan membuat soket TCP.

Ketika klien membuat soket TCP, itu menentukan alamat soket penerima di server, yaitu, alamat IP dari host server dan nomor port soket. Setelah membuat soketnya, klien memulai three-way handshake dan membuat koneksi TCP dengan server. three-way handshake, yang terjadi di dalam lapisan transport, sama sekali tidak terlihat oleh program klien dan server. Selama three-way handshake, proses klien mengetuk pintu penyambutan proses server. Ketika server "mendengar" ketukan, itu menciptakan pintu baru — lebih tepatnya, soket baru yang didedikasikan untuk klien tertentu. Dalam contoh kita di bawah ini, pintu penyambutan adalah objek soket TCP yang kita sebut `serverSocket`; soket yang baru dibuat yang didedikasikan untuk klien yang membuat koneksi disebut `connectionSocket`.

Siswa yang menemukan socket TCP untuk pertama kalinya terkadang bingung dengan soket penerima (yang merupakan titik kontak awal untuk semua klien yang ingin berkomunikasi dengan server), dan setiap soket koneksi sisi server yang baru dibuat yang kemudian dibuat untuk berkomunikasi dengan setiap klien. Dari perspektif aplikasi, soket klien dan soket koneksi server terhubung langsung dengan pipa. Proses klien dapat mengirim byte sewenang-wenang ke dalam soketnya, dan TCP menjamin bahwa proses server akan menerima (melalui soket koneksi) setiap byte dalam urutan yang dikirim.

TCP dengan demikian menyediakan layanan yang andal antara proses klien dan server. Selanjutnya, seperti halnya orang dapat masuk dan keluar dari pintu yang sama, proses klien tidak hanya mengirim byte ke dalam tetapi juga menerima byte dari soketnya; sama halnya, proses server tidak hanya menerima byte dari tetapi juga mengirim byte ke soket koneksinya. Kita menggunakan aplikasi client-server sederhana yang sama untuk mendemonstrasikan pemrograman soket dengan TCP: Klien mengirimkan satu baris data ke server, server mengkapitalisasi baris dan mengirimkannya kembali ke klien. Figure 7.4 menyoroti aktivitas utama yang berhubungan dengan soket dari klien dan server yang berkomunikasi melalui layanan trans-port TCP.

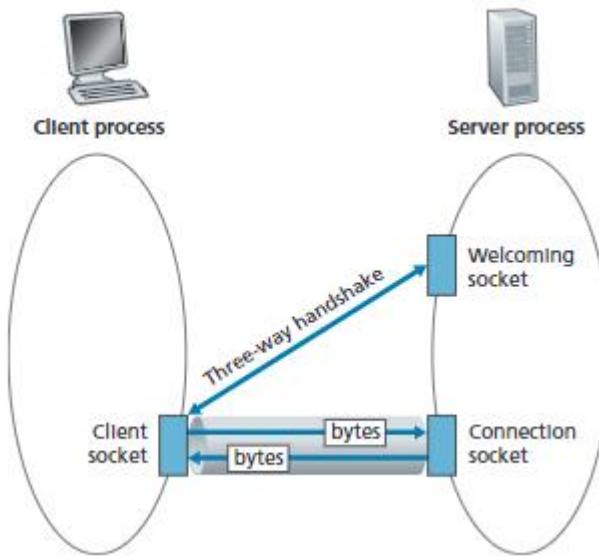


Figure 7.4 The TCPserver process has two sockets

### 7.3.1 TCPClient.py

Berikut adalah kode untuk sisi klien aplikasi:

```

from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input('Input lowercase sentence:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print('From Server: ', modifiedSentence.decode())
clientSocket.close()

```

Figure 7.5 TCPClient.py

Sekarang mari kita lihat berbagai baris dalam kode yang berbeda secara signifikan dari implementasi UDP. Baris pertama adalah pembuatan soket klien.

```
clientSocket = socket(AF_INET, SOCK_STREAM)
```

Baris ini membuat soket klien, yang disebut clientSocket. Parameter pertama sekali lagi menunjukkan bahwa jaringan yang mendasarinya menggunakan IPv4. Parameter kedua

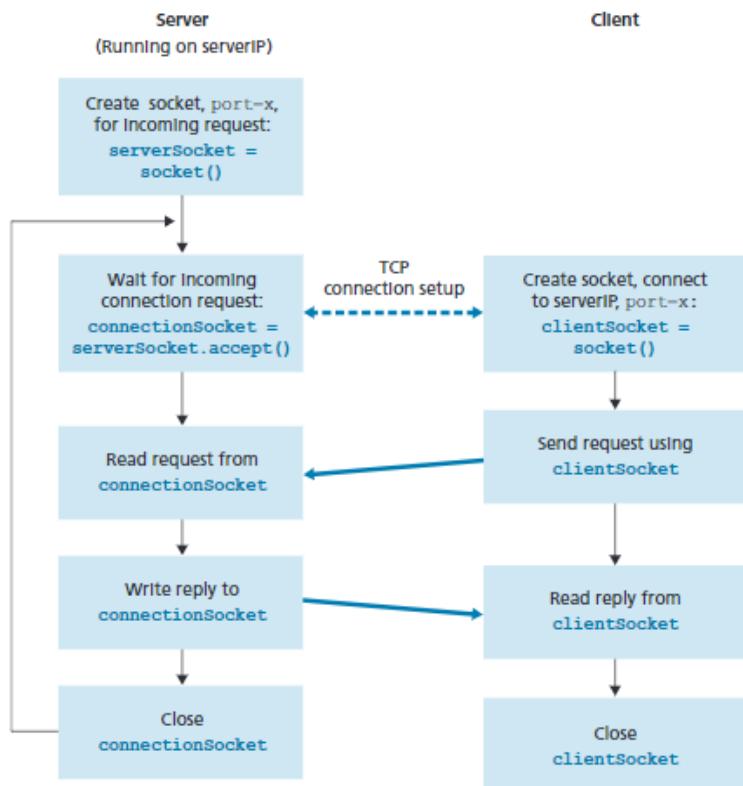


Figure 7.5 client-server application menggunakan TCP

menunjukkan bahwa soket bertipe SOCK\_STREAM, yang berarti soket TCP (bukan soket UDP). Perhatikan bahwa kita sekali lagi tidak menentukan nomor port soket klien saat kita membuatnya; kita malah membiarkan sistem operasi melakukan ini untuk kita. Sekarang baris kode berikutnya sangat berbeda dari apa yang kita lihat di UDPClient:

```
clientSocket.connect((serverName, serverPort))
```

Ingatlah bahwa sebelum klien dapat mengirim data ke server (atau sebaliknya) menggunakan soket TCP, koneksi TCP harus dibuat terlebih dahulu antara klien dan server. Baris di atas memulai koneksi TCP antara klien dan server. Parameter metode connect() adalah alamat sisi server koneksi. Setelah baris kode ini dieksekusi, three-way handshake dilakukan dan koneksi TCP dibuat antara klien dan server.

```
sentence = input('Input lowercase sentence:')
```

Seperti halnya UDPClient, di atas memperoleh kalimat dari pengguna. Kalimat string terus mengumpulkan karakter sampai pengguna mengakhiri baris dengan mengetik carriage return. Baris kode berikutnya juga sangat berbeda dari UDPClient:

```
clientSocket.send(sentence.encode())
```

baris di atas mengirimkan kalimat melalui soket klien dan ke koneksi TCP. Perhatikan bahwa program tidak secara eksplisit membuat paket dan melampirkan alamat tujuan ke paket, seperti halnya dengan soket UDP. Sebaliknya program klien hanya menjatuhkan byte dalam kalimat string ke dalam koneksi TCP. Klien kemudian menunggu untuk menerima byte dari server.

```
modifiedSentence = clientSocket.recv(2048)
```

Ketika karakter tiba dari server, mereka ditempatkan ke dalam string yang dimodifikasiSentence. Karakter terus terakumulasi dalam ModifiedSentence sampai baris berakhir dengan karakter carriage return. Setelah mencetak kalimat dengan huruf kapital, kita menutup soket klien:

```
clientSocket.close()
```

Baris terakhir ini menutup soket dan, karenanya, menutup koneksi TCP antara klien dan server. Ini menyebabkan TCP di klien mengirim pesan TCP ke TCP di server.

### 7.3.2 TCPServer.py

Sekarang mari kita lihat program server.

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print('The server is ready to receive')

while True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.encode())
    connectionSocket.close()
```

Figure 7. 7 TCPServer.py

Sekarang mari kita lihat baris yang berbeda secara signifikan dari UDPServer dan TCP-Client. Seperti halnya TCPClient, server membuat soket TCP dengan:

```
serverSocket=socket (AF_INET, SOCK_STREAM)
```

Mirip dengan UDPServer, kita mengaitkan nomor port server, serverPort, dengan soket ini:

```
serverSocket.bind(('',serverPort))
```

Tetapi dengan TCP, serverSocket akan menjadi soket penerima kita. Setelah membangun pintu masuk ini, kami akan menunggu dan mendengarkan beberapa klien untuk mengetuk pintu:



```
serverSocket.listen(1)
```

Baris ini membuat server mendengarkan permintaan koneksi TCP dari klien. Parameter menentukan jumlah maksimum koneksi antrian (setidaknya 1).

```
connectionSocket, addr = serverSocket.accept()
```

Ketika klien mengetuk pintu ini, program memanggil metode accept() untuk serverSocket, yang membuat soket baru di server, yang disebut connectionSocket, yang didedikasikan untuk klien khusus ini. Klien dan server kemudian menyelesaikan handshake, membuat koneksi TCP antara clientSocket klien dan connectionSocket server. Dengan koneksi TCP dibuat, klien dan server sekarang dapat mengirim byte satu sama lain melalui koneksi. Dengan TCP, semua byte yang dikirim dari satu sisi hanya dijamin tiba di sisi lain tetapi juga dijamin tiba secara berurutan.

```
connectionSocket.close()
```

Dalam program ini, setelah mengirim kalimat yang dimodifikasi ke klien, kita menutup soket koneksi. Tetapi karena serverSocket tetap terbuka, klien lain sekarang dapat mengetuk pintu dan mengirim server sebuah kalimat untuk dimodifikasi.

Ini melengkapi diskusi kita tentang pemrograman soket di TCP.

Anda didorong untuk menjalankan dua program di dua host yang berbeda, dan juga memodifikasinya untuk mencapai tujuan yang sedikit berbeda. Anda harus membandingkan pasangan program UDP dengan pasangan program TCP dan melihat perbedaannya.

## Modul 8 WEB SERVER DAN PEMBAGIAN TOPIK TUGAS BESAR

### Tujuan Praktikum

1. Mahasiswa bisa membuat program web server sederhana berbasis TCP socket programming

### 8.1 Pengantar

Pada modul ini, Anda akan mempelajari dasar-dasar pemrograman soket untuk koneksi TCP dengan Python: cara membuat soket, mengikatnya ke alamat dan port tertentu, serta mengirim dan menerima paket HTTP. Anda juga akan mempelajari beberapa dasar format header HTTP. Anda akan mengembangkan server web yang menangani satu permintaan HTTP pada satu waktu. Server web Anda harus menerima dan mengurai permintaan HTTP, mendapatkan file yang diminta dari sistem file server, membuat pesan respons HTTP yang terdiri dari file yang diminta yang didahului oleh baris header, dan kemudian mengirim respons langsung ke klien. Jika file yang diminta tidak ada di server, server harus mengirim pesan HTTP "404 Not Found" kembali ke klien.

### 8.2 Kode

Di bawah ini Anda akan menemukan kode kerangka untuk server Web. Anda harus melengkapi kode kerangka. Tempat-tempat di mana Anda perlu mengisi kode ditandai dengan #Fill in start dan #Fill in end. Setiap tempat mungkin memerlukan satu atau lebih baris kode.

### 8.3 Menjalankan Server

Letakkan file HTML (contoh: HelloWorld.html) di direktori yang sama dengan tempat server berada. Jalankan program server. Tentukan alamat IP host yang menjalankan server (contoh: 128.238.251.26). Dari host lain, buka browser dan berikan URL yang sesuai. Misalnya: <http://128.238.251.26:6789>HelloWorld.html> adalah nama file yang Anda tempatkan di direktori server. Perhatikan juga penggunaan **nomor port** setelah titik dua. Anda perlu mengganti nomor port ini dengan port apa pun yang Anda gunakan dalam kode server. Pada contoh di atas, kita telah menggunakan nomor port **6789**. Browser kemudian akan menampilkan isi HelloWorld.html. Jika Anda menghilangkan ":6789", browser akan menganggap port 80 dan Anda akan mendapatkan halaman web dari server hanya jika server Anda listening di port 80. Kemudian cobalah untuk mendapatkan file yang tidak ada di server. Anda akan mendapatkan pesan "**404 Not Found**".

### 8.4 Yang harus dikerjakan

Anda akan mengumpulkan kode server lengkap bersama dengan tangkapan layar browser klien Anda, memverifikasi bahwa Anda benar-benar menerima konten file HTML dari server.

### 8.5 Skeleton Kode Python untuk Web Server

```
#import socket module
from socket import *

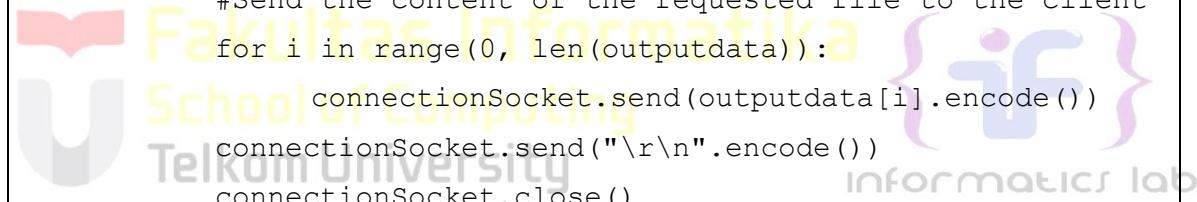
import sys # In order to terminate the program

serverSocket = socket(AF_INET, SOCK_STREAM)
```

```

#Prepare a sever socket
#Fill in start
#Fill in end
while True:
    #Establish the connection
    print('Ready to serve...')
    connectionSocket, addr = #Fill in start #Fill
    in end
    try:
        message = #Fill in start #Fill in end
        filename = message.split()[1]
        f = open(filename[1:])
        outputdata = #Fill in start #Fill in end
        #Send one HTTP header line into socket
        #Fill in start
        #Fill in end
        #Send the content of the requested file to the client
        for i in range(0, len(outputdata)):
            connectionSocket.send(outputdata[i].encode())
        connectionSocket.send("\r\n".encode())
        connectionSocket.close()
    except IOError:
        #Send response message forfile not found
        #Fill in start
        #Fill in end
        #Close client socket
        #Fill in start
        #Fill in end
    serverSocket.close()
    sys.exit()#Terminate the program after sending the corresponding
    data

```



## 8.6 Latihan Tambahan

Saat ini, server web hanya menangani satu permintaan HTTP dalam satu waktu. Menerapkan server multithreaded yang mampu melayani beberapa permintaan secara bersamaan. Dengan menggunakan threading, pertama-tama buat thread utama di mana server Anda yang dimodifikasi mendengarkan klien di port tetap. Ketika menerima permintaan koneksi TCP dari klien, itu akan

mengatur koneksi TCP melalui port lain dan melayani permintaan klien di thread terpisah. Akan ada koneksi TCP terpisah di thread terpisah untuk setiap pasangan permintaan/respons.2. Daripada menggunakan browser, tulis klien HTTP Anda sendiri untuk menguji server Anda. Klien Anda akan terhubung ke server menggunakan koneksi TCP, kirim Permintaan HTTP ke server, dan menampilkan respons server sebagai output. Anda dapat mengasumsikan bahwa permintaan HTTP yang dikirim adalah metod GET. Klien harus mengambil argumen baris perintah yang menentukan alamat IP server atau nama host, port tempat server listening berada, dan jalur di mana objek yang diminta disimpan di server. Berikut ini adalah format perintah input untuk menjalankan klien

```
client.py server_host server_port filename
```

## 8.7 Pembagian Topik Tugas Besar

Selanjutnya di kelas praktikum akan dilakukan plotting untuk menentukan pembagian kelompok tugas besar untuk praktikum Jaringan Komputer sesuai dengan topik tugas besar yang telah disediakan dengan spesifikasi yang telah ditentukan oleh dosen koordinator dan sesuai dengan arahan dari asisten praktikum di kelas.



Fakultas Informatika  
School of Computing  
Telkom University



# Modul 9 IP

## Tujuan Praktikum

1. Mahasiswa dapat menginvestigasi cara kerja protokol IP menggunakan Wireshark

## 9.1 Pengantar

Di modul ini, kita akan mempelajari protokol IP yang terkenal, dengan fokus pada datagram IPv4 dan IPv6. Modul ini memiliki tiga bagian. Pada bagian pertama, kita akan menganalisis paket dalam jejak datagram IPv4 yang dikirim dan diterima oleh program traceroute (program traceroute itu sendiri dieksplorasi lebih detail di lab Wireshark ICMP). Kita akan mempelajari fragmentasi IP di bagian 2 modul ini, dan melihat sekilas IPv6 di bagian 3 modul ini.

## 9.2 Menangkap paket dari eksekusi traceroute

Untuk menghasilkan jejak datagram IPv4 untuk dua bagian pertama modul ini, kita akan menggunakan program traceroute untuk mengirim datagram dengan dua ukuran berbeda ke [gaia.cs.umass.edu](http://gaia.cs.umass.edu). Ingat bahwa traceroute beroperasi dengan terlebih dahulu mengirim satu atau lebih datagram dengan bidang time-to-live (TTL) di header IP yang disetel ke 1; kemudian mengirimkan serangkaian satu atau lebih datagram menuju tujuan yang sama dengan nilai TTL 2; kemudian mengirimkan serangkaian datagram ke tujuan yang sama dengan nilai TTL 3; dan seterusnya.

Ingatlah bahwa router harus mengurangi TTL di setiap datagram yang diterima sebesar 1 (sebenarnya, RFC 791 mengatakan bahwa router harus mengurangi TTL setidaknya satu). Jika TTL mencapai 0, router mengembalikan pesan ICMP (tipe 11 – TTL-melebihi) ke host pengirim. Sebagai hasil dari perilaku ini, datagram dengan TTL 1 (dikirim oleh host yang mengeksekusi traceroute) akan menyebabkan router satu hop dari pengirim mengirim pesan ICMP TTL-exceeded kembali ke pengirim; datagram yang dikirim dengan TTL 2 akan menyebabkan router dua hop untuk mengirim pesan ICMP kembali ke pengirim; datagram yang dikirim dengan TTL 3 akan menyebabkan router tiga hop untuk mengirim pesan ICMP kembali ke pengirim; dan seterusnya. Dengan cara ini, host yang mengeksekusi traceroute dapat mempelajari alamat IP router antara dirinya dan tujuan dengan melihat alamat IP sumber dalam datagram yang berisi pesan ICMP TTL-exceeded.

Mari kita jalankan traceroute dan kirimkan datagram dengan dua ukuran berbeda. Yang lebih besar dari dua panjang datagram akan membutuhkan pesan traceroute untuk difragmentasi di beberapa datagram IPv4.

- Linux/MacOS. Dengan perintah traceroute Linux/MacOS, ukuran datagram UDP yang dikirim ke tujuan akhir dapat diatur secara eksplisit dengan menunjukkan jumlah byte dalam datagram; nilai ini dimasukkan dalam baris perintah traceroute segera setelah nama atau alamat tujuan. Misalnya, untuk mengirim datagram traceroute 2000 byte ke [gaia.cs.umass.edu](http://gaia.cs.umass.edu), perintahnya adalah:

```
%traceroute gaia.cs.umass.edu 2000
```

- Windows. Program tracert yang disediakan dengan Windows tidak mengizinkan seseorang untuk mengubah ukuran pesan ICMP yang dikirim oleh tracert. Jadi tidak mungkin menggunakan mesin Windows untuk menghasilkan pesan ICMP yang cukup besar untuk memaksa fragmentasi IP. Namun, Anda dapat menggunakan tracert untuk menghasilkan

paket kecil dengan panjang tetap untuk melakukan Bagian 1 lab ini. Pada prompt perintah DOS masukkan:

```
>tracert gaia.cs.umass.edu
```

Jika Anda ingin melakukan bagian kedua dari modul ini, Anda dapat mengunduh file jejak paket yang ditangkap di salah satu komputer.

Lakukan hal berikut:

- Mulai Wireshark dan mulai pengambilan paket. (Capture->Start atau klik tombol sirip hiu biru di kiri atas jendela Wireshark).
- Masukkan dua perintah traceroute, menggunakan gaia.cs.umass.edu sebagai tujuan, yang pertama dengan panjang 56 byte. Setelah perintah itu selesai dijalankan, masukkan perintah traceroute kedua untuk tujuan yang sama, tetapi dengan panjang 3000 byte.
- Hentikan pelacakan Wireshark.

Jika Anda tidak dapat menjalankan Wireshark pada koneksi jaringan langsung, Anda dapat menggunakan file jejak paket, ip-wireshark-trace1-1.pcapng. Anda mungkin merasa perlu mengunduh jejak ini bahkan jika Anda telah menangkap jejak Anda sendiri dan menggunakannya, serta jejak Anda sendiri, saat Anda menjelajahi pertanyaan di bawah ini.

### 9.2.1 Bagian 1: IPv4 Dasar

Dalam penelusuran Anda, Anda seharusnya dapat melihat rangkaian segmen UDP (untuk MacOS/Linux) atau pesan ICMP Echo Request (Windows) yang dikirim oleh traceroute di komputer Anda, dan pesan ICMP TTL-exceeded yang dikembalikan ke komputer Anda oleh router perantara. Dalam pertanyaan di bawah, kami akan menganggap Anda menggunakan komputer MacOS/Linux; pertanyaan terkait untuk kasus mesin Windows harus jelas. Layar Anda akan terlihat mirip dengan tangkapan layar pada Gambar 9.1, di mana kami telah menggunakan filter tampilan “udp || icmp” (lihat bidang filter tampilan yang diisi hijau muda pada Gambar 9.1) sehingga hanya protokol UDP dan/atau ICMP paket ditampilkan.

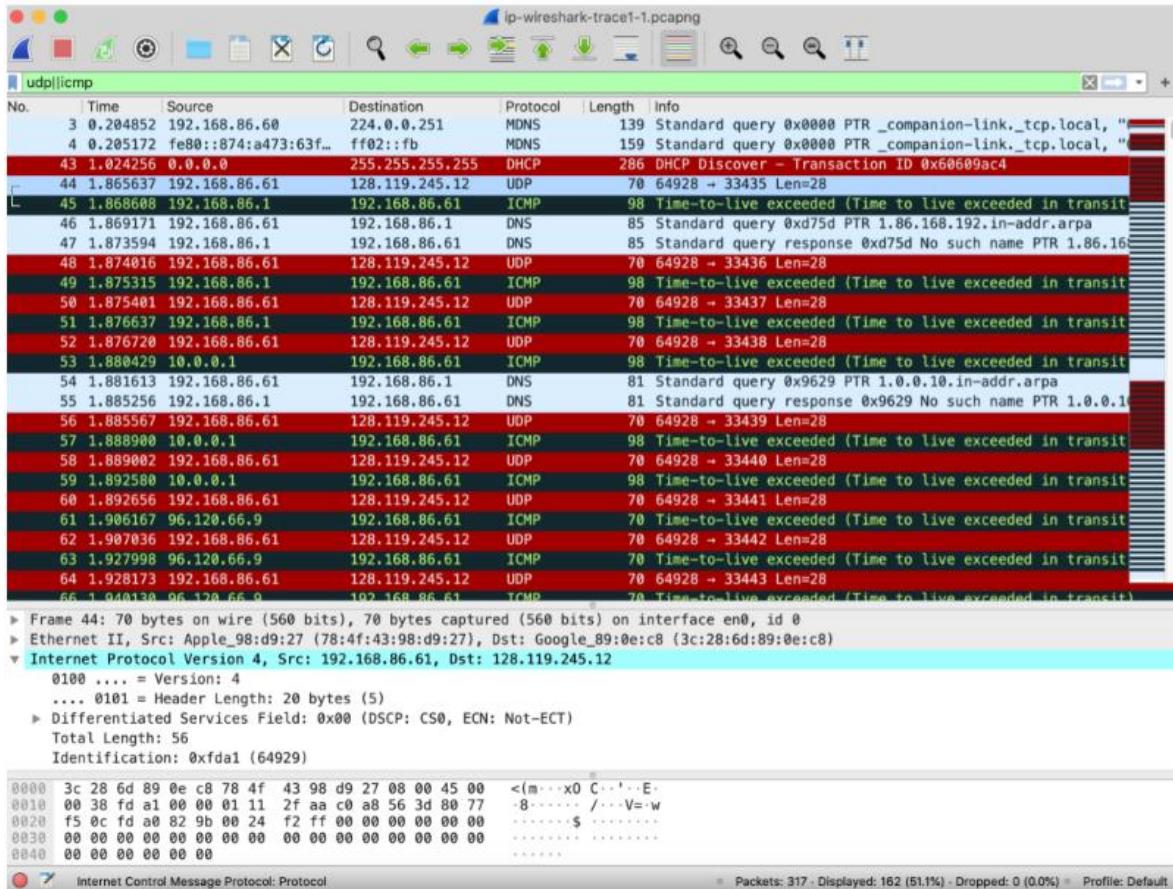


Figure 9.1 Tangkapan layar Wireshark, menunjukkan paket UDP dan ICMP di tracefile ip-wireshark-trace1-1.pcapng

Selanjutnya, mari kita lihat urutan segmen UDP yang dikirim dari komputer Anda melalui traceroute, ditujukan ke 128.119.245.12. Filter tampilan yang dapat Anda masukkan untuk melakukan ini adalah “ip.src==192.168.86.61 dan ip.dst==128.119.245.12 dan udp dan !icmp”. Ini akan memungkinkan Anda untuk dengan mudah bergerak secara berurutan hanya melalui datagram yang hanya berisi segmen-segmen ini. Layar Anda akan terlihat seperti Gambar 9.2.

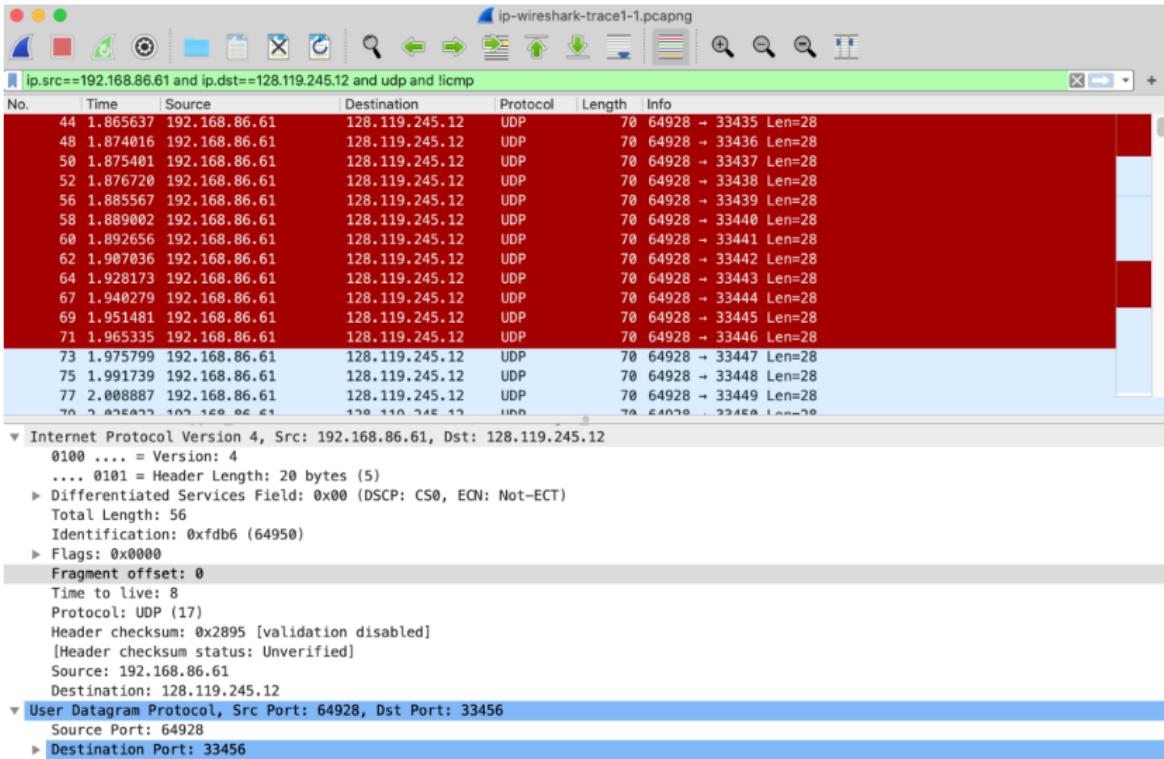


Figure 9.2 Tangkapan layar Wireshark, menampilkan segmen di tracefile ip-wireshark-trace1-1.pcapng menggunakan filter tampilan ip.src==192.168.86.61 dan ip.dst==128.119.245.12 dan udp dan !icmp

Sekarang mari kita lihat paket ICMP yang dikembalikan ke komputer Anda oleh router intervensi di mana nilai TTL dikurangi menjadi nol (dan karenanya menyebabkan pesan kesalahan ICMP dikembalikan ke komputer Anda). Filter tampilan yang dapat Anda gunakan untuk menampilkan paket-paket ini saja adalah “ip.dst==192.168.86.61 dan icmp”.

### 9.2.2 Bagian 2: Fragmentasi

Di bagian ini, kita akan melihat segmen UDP besar (3000-byte) yang dikirim oleh program traceroute yang dipecah menjadi beberapa datagram IP. Kami menyarankan Anda terlebih dahulu membaca materi tentang fragmentasi IP dari edisi ke-7 buku teks kami (dan sebelumnya) di [http://gaia.cs.umass.edu/kurose\\_ross/Kurose\\_Ross\\_7th\\_edition\\_section\\_4.3.2.pdf](http://gaia.cs.umass.edu/kurose_ross/Kurose_Ross_7th_edition_section_4.3.2.pdf)

Urutkan daftar paket dari Bagian 1, dengan filter tampilan apa pun dihapus, menurut waktu, dengan mengklik kolom Waktu

### 9.2.3 Bagian 3: IPv6

Di bagian terakhir ini kita akan melihat sekilas datagram IPv6 menggunakan Wireshark. Untuk menghasilkan jejak ini, browser web kami membuka beranda youtube.com. Youtube (dan Google) memberikan dukungan yang cukup luas untuk IPv6.

Buka file ip-wireshark-trace2-1.pcapng dalam file .zip trace yang diberikan pada catatan kaki 2. Tampilan Wireshark Anda akan terlihat seperti Gambar 9.3.

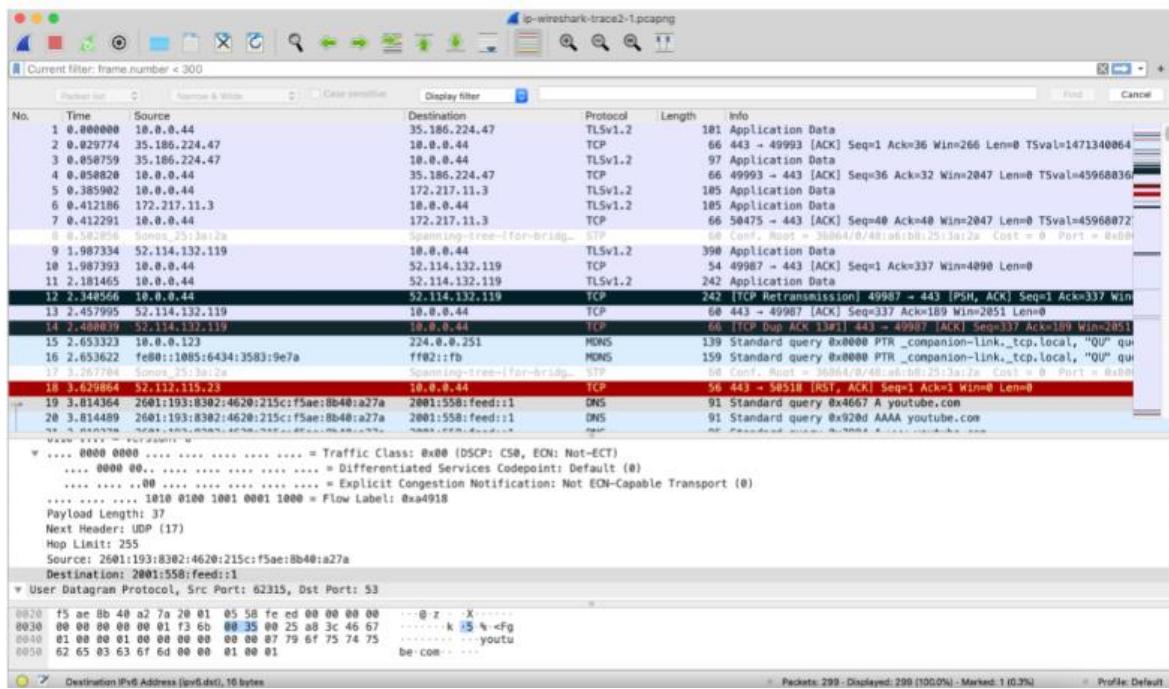


Figure 9.3 Tangkapan layar Wireshark, menunjukkan paket pertama yang diambil dalam ip-wireshark-trace2-1.pcapng. Jika Anda melihat kolom Sumber, Anda akan melihat beberapa alamat IPv6

Mari kita mulai dengan melihat lebih dekat pada paket ke-20 dalam jejak ini, dikirim pada t=3.814489. Ini adalah permintaan DNS (terkandung dalam datagram IPv6) ke server DNS IPv6 untuk alamat IPv6 youtube.com. Jenis permintaan DNS AAAA digunakan untuk menyelesaikan nama ke alamat IP IPv6.

# Modul 10 DHCP

## Tujuan Praktikum

1. Mahasiswa dapat menginvestigasi cara kerja protokol DHCP menggunakan Wireshark.

## 10.1 Pengantar

Di modul ini, kita akan melihat sekilas Dynamic Host Configuration Protocol, DHCP. Ingat bahwa DHCP digunakan secara luas di perusahaan, universitas dan LAN kabel dan nirkabel jaringan rumah untuk secara dinamis menetapkan alamat IP ke host, serta untuk mengkonfigurasi informasi konfigurasi jaringan lainnya.

Seperti yang telah kami lakukan di lab Wireshark sebelumnya, Anda akan melakukan beberapa tindakan di komputer Anda yang akan menyebabkan DHCP beraksi, dan kemudian menggunakan Wireshark untuk mengumpulkan dan kemudian jejak paket yang berisi pesan protokol DHCP.

## 10.2 Mengumpulkan Jejak Paket

Misalnya, alamat IP baru diperlukan, atau alamat DHCP yang ada akan diperbarui); pesan Permintaan dan ACK tidak. Untuk mengumpulkan jejak yang akan berisi keempat jenis pesan DHCP, kita perlu melakukan beberapa tindakan baris perintah di Mac, Linux, atau PC.

Pada Mac:

- di jendela terminal/Shell masukkan perintah berikut:

```
% sudo ipconfig set en0 none
```

Di mana en0 (dalam contoh ini) adalah antarmuka tempat Anda ingin menangkap paket menggunakan Wireshark. Anda dapat dengan mudah menemukan daftar nama antarmuka di Wireshark dengan memilih opsi Capture->options. Perintah ini akan menghapus konfigurasi antarmuka jaringan en0.

- Mulai Wireshark, ambil paket pada antarmuka yang Anda de-konfigurasi pada Langkah 1.
- Di jendela terminal/shell masukkan perintah berikut:

```
% sudo ipconfig set en0 dhcp
```

Ini akan menyebabkan protokol DHCP meminta dan menerima alamat IP dan informasi lain dari server DHCP.

- Setelah menunggu beberapa detik, hentikan pengambilan Wireshark.

Pada Linux:

1. Di jendela/shell terminal, masukkan perintah berikut:

```
sudo ip addr flush en0  
sudo dhclient -r
```

di mana en0 (dalam contoh ini) adalah antarmuka tempat Anda ingin menangkap paket menggunakan Wireshark. Anda dapat dengan mudah menemukan daftar nama antarmuka di Wireshark dengan memilih Capture -> Options. Perintah ini akan menghapus alamat IP antarmuka yang ada, dan melepaskan sewa alamat DHCP yang ada.

2. Mulai Wireshark, ambil paket di antarmuka yang Anda de-konfigurasi di Langkah 1.
3. Di jendela terminal/shell, masukkan perintah berikut:

```
sudo dhclient en0
```

di mana, seperti di atas, en0 adalah antarmuka di mana Anda saat ini menangkap paket. Ini akan menyebabkan protokol DHCP meminta dan menerima alamat IP dan informasi lain dari server DHCP.

4. Setelah menunggu beberapa detik, hentikan pengambilan Wireshark.

Di PC:

1. Di jendela baris perintah, masukkan perintah berikut::

```
> ipconfig /release
```

Perintah ini akan menyebabkan PC Anda memberikan alamat IP-nya.

2. Mulai Wireshark.

3. Di jendela baris perintah, masukkan perintah berikut:

```
> ipconfig /renew
```

Ini akan menyebabkan protokol DHCP meminta dan menerima alamat IP dan informasi lainnya dari server DHCP.

4. Setelah menunggu beberapa detik, hentikan pengambilan Wireshark.

Setelah menghentikan pengambilan Wireshark pada langkah 4, Anda harus melihat di jendela Wireshark Anda untuk memastikan Anda benar-benar menangkap paket yang kami cari. Jika Anda memasukkan "dhcp" ke dalam bidang filter tampilan (seperti yang ditunjukkan pada bidang hijau muda di kiri atas Figure 10.1), layar Anda (pada Mac) akan terlihat seperti Gambar 10.1.

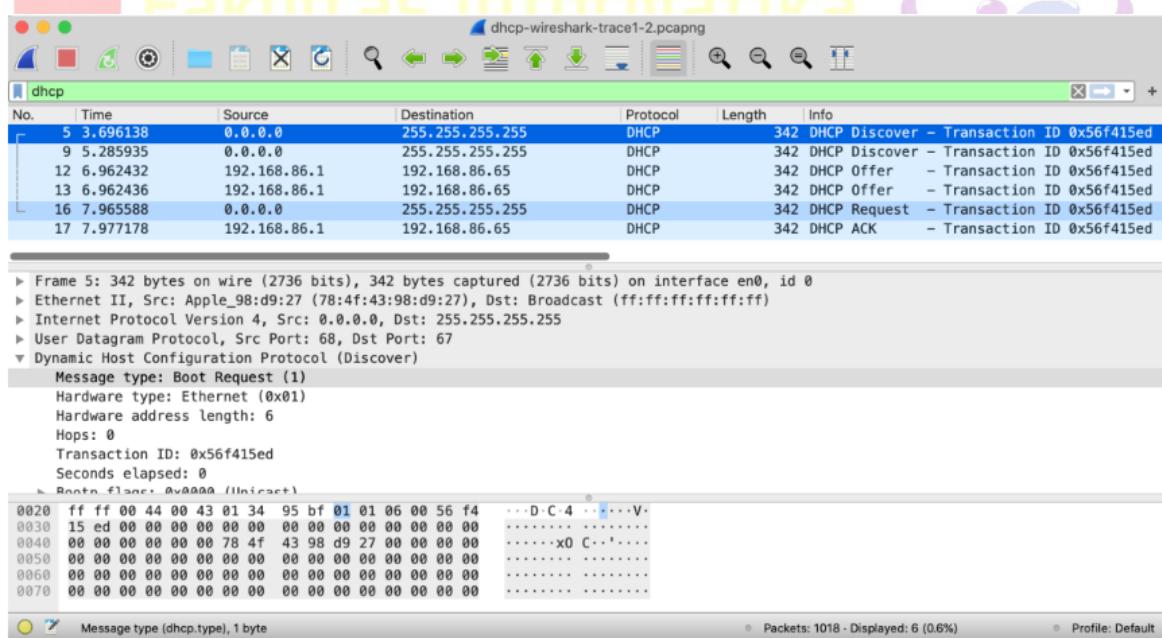


Figure 10.1 Tampilan Wireshark, menampilkan tangkapan DHCP Discover, Offer, Request, dan pesan ACK

Jika Anda tidak dapat menjalankan Wireshark pada koneksi jaringan langsung, tidak dapat menangkap keempat pesan DHCP, atau ditugaskan untuk melakukannya oleh instruktur, Anda dapat menggunakan file pelacakan Wireshark, dhcp-wireshark-trace1-1.pcapng2 yang telah kami kumpulkan mengikuti langkah-langkah di atas di salah satu komputer penulis. Anda mungkin merasa berharga untuk mengunduh jejak ini bahkan jika Anda telah menangkap jejak Anda sendiri dan menggunakannya, serta jejak Anda sendiri, saat Anda menjelajahi pertanyaan di bawah ini.

## Modul 11 Asistensi Tugas Besar

Tujuan Praktikum
------------------

- |   |
|---|
| 1. Melakukan asistensi dan laporan progress penggerjaan tugas besar |
|---|

Pada modul 11 ini, silahkan laporkan progress penggerjaan tugas besar Jaringan Komputer yang telah dikerjakan dan lanjutkan asistensi untuk menyelesaikan penggerjaan tugas besar di waktu praktikum kelas masing-masing.



Fakultas Informatika  
School of Computing  
Telkom University



## Modul 12 ICMP

### Tujuan Praktikum

1. Mahasiswa dapat menginvestigasi cara kerja protokol ICMP menggunakan Wireshark
2. Mahasiswa dapat membuat program ICMP Pinger

### 12.1 Pengantar

Di modul ini, kita akan mengeksplorasi beberapa aspek dari protokol ICMP:

- Pesan ICMP yang dihasilkan oleh program Ping;
- Pesan ICMP yang dihasilkan oleh program Traceroute;
- format dan isi pesan ICMP.

Kami menyajikan lab ini dalam konteks sistem operasi Microsoft Windows. Namun, sangat mudah untuk menerjemahkan lab ke lingkungan Unix atau Linux.

### 12.2 ICMP dan Ping

Mari kita mulai pembelajaran CMP kita dengan menangkap paket yang dihasilkan oleh program Ping. Anda mungkin ingat bahwa program Ping adalah alat sederhana yang memungkinkan siapa saja (misalnya, administrator jaringan) untuk memverifikasi apakah sebuah host aktif atau tidak. Program Ping di host sumber mengirimkan paket ke alamat IP target; jika target hidup, program Ping di host target merespons dengan mengirimkan paket kembali ke host sumber. Seperti yang mungkin sudah Anda duga (mengingat lab ini tentang ICMP), kedua paket Ping ini adalah paket ICMP.

Lakukan hal berikut:

- Mari kita mulai petualangan ini dengan membuka aplikasi Windows Command Prompt (yang dapat ditemukan di folder Aksesoris Anda).
- Jalankan packet sniffer Wireshark, dan mulai pengambilan paket Wireshark.
- Perintah ping ada di c:\windows\system32, jadi ketikkan “ping -n 10 hostname” atau “c:\windows\system32\ping -n 10 hostname” di baris perintah MS-DOS (tanpa tanda kutip), di mana nama host adalah host di benua lain. Jika Anda berada di luar Asia, Anda mungkin ingin memasukkan www.ust.hk untuk server Web di Universitas Sains dan Teknologi Hong Kong. Argumen “-n 10” menunjukkan bahwa 10 pesan ping harus dikirim. Kemudian jalankan program Ping dengan mengetikkan return.
- Saat program Ping berakhir, hentikan pengambilan paket di Wireshark.

Di akhir percobaan, Jendela Command Prompt Anda akan terlihat seperti Gambar 12.1. Dalam contoh ini, program ping sumber berada di Massachusetts dan program Ping tujuan berada di Hong Kong. Dari jendela ini kita melihat bahwa program ping sumber mengirim 10 paket kueri dan menerima 10 tanggapan. Perhatikan juga bahwa untuk setiap respons, sumber menghitung waktu pulang pergi (round-trip time (RTT)), yang untuk 10 paket rata-rata 375 mdtk.

```
C:\> Command Prompt  
C:\WINDOWS\SYSTEM32>ping -n 10 www.ust.hk  
Pinging www.ust.hk [143.89.14.34] with 32 bytes of data:  
Reply from 143.89.14.34: bytes=32 time=415ms TTL=231  
Reply from 143.89.14.34: bytes=32 time=425ms TTL=231  
Reply from 143.89.14.34: bytes=32 time=318ms TTL=231  
Reply from 143.89.14.34: bytes=32 time=314ms TTL=231  
Reply from 143.89.14.34: bytes=32 time=336ms TTL=231  
Reply from 143.89.14.34: bytes=32 time=359ms TTL=231  
Reply from 143.89.14.34: bytes=32 time=381ms TTL=231  
Reply from 143.89.14.34: bytes=32 time=401ms TTL=231  
Reply from 143.89.14.34: bytes=32 time=400ms TTL=231  
Reply from 143.89.14.34: bytes=32 time=409ms TTL=231  
  
Ping statistics for 143.89.14.34:  
    Packets: Sent = 10, Received = 10, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
    Minimum = 314ms, Maximum = 425ms, Average = 375ms  
C:\WINDOWS\SYSTEM32>  
C:\WINDOWS\SYSTEM32>  
C:\WINDOWS\SYSTEM32>
```

Figure 12.1 Jendela Command Prompt setelah memasukkan perintah Ping

Gambar 12.2 memberikan screenshot dari output Wireshark, setelah "icmp" telah dimasukkan ke dalam jendela tampilan filter. Perhatikan bahwa daftar paket menunjukkan 20 paket: 10 kueri Ping yang dikirim oleh sumber dan 10 respons Ping yang diterima oleh sumber. Perhatikan juga bahwa alamat IP sumber adalah alamat pribadi (di belakang NAT) dengan format 192.168/12; alamat IP tujuan adalah alamat web server di HKUST. Sekarang mari kita perbesar paket pertama (dikirim oleh klien); pada gambar di bawah, area isi paket memberikan informasi tentang paket ini. Kita melihat bahwa datagram IP dalam paket ini memiliki nomor protokol 01, yang merupakan nomor protokol untuk ICMP. Ini berarti bahwa muatan datagram IP adalah paket ICMP.

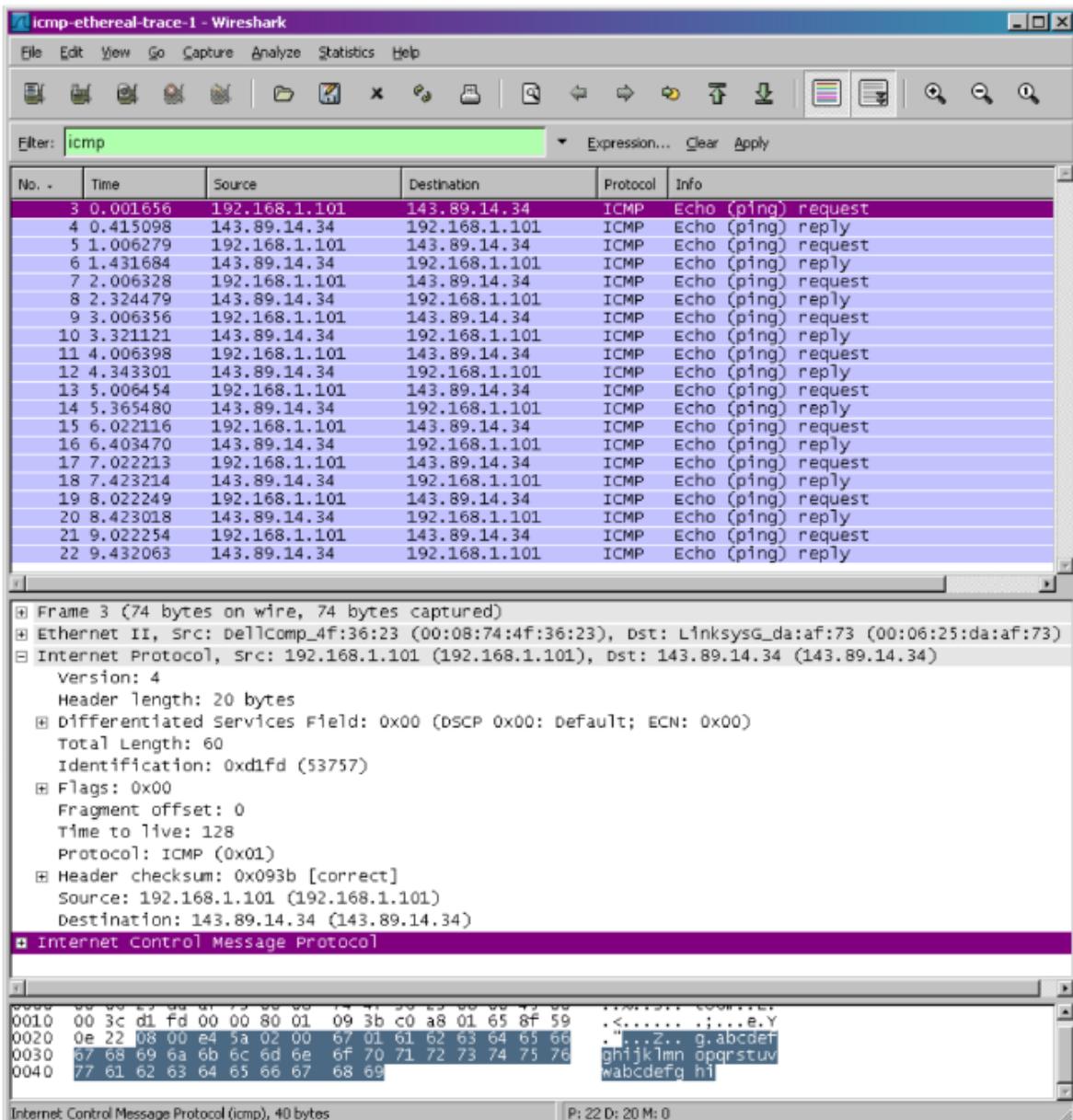


Figure 12.2 Output Wireshark untuk program Ping dengan Protokol Internet diperluas.

Gambar 12.3 berfokus pada ICMP yang sama tetapi telah memperluas informasi protokol ICMP di jendela isi paket. Perhatikan bahwa paket ICMP ini adalah Tipe 8 dan Kode 0 - yang disebut paket "permintaan gema" ICMP. Perhatikan juga bahwa paket ICMP ini berisi checksum, identifier, dan sequence number.

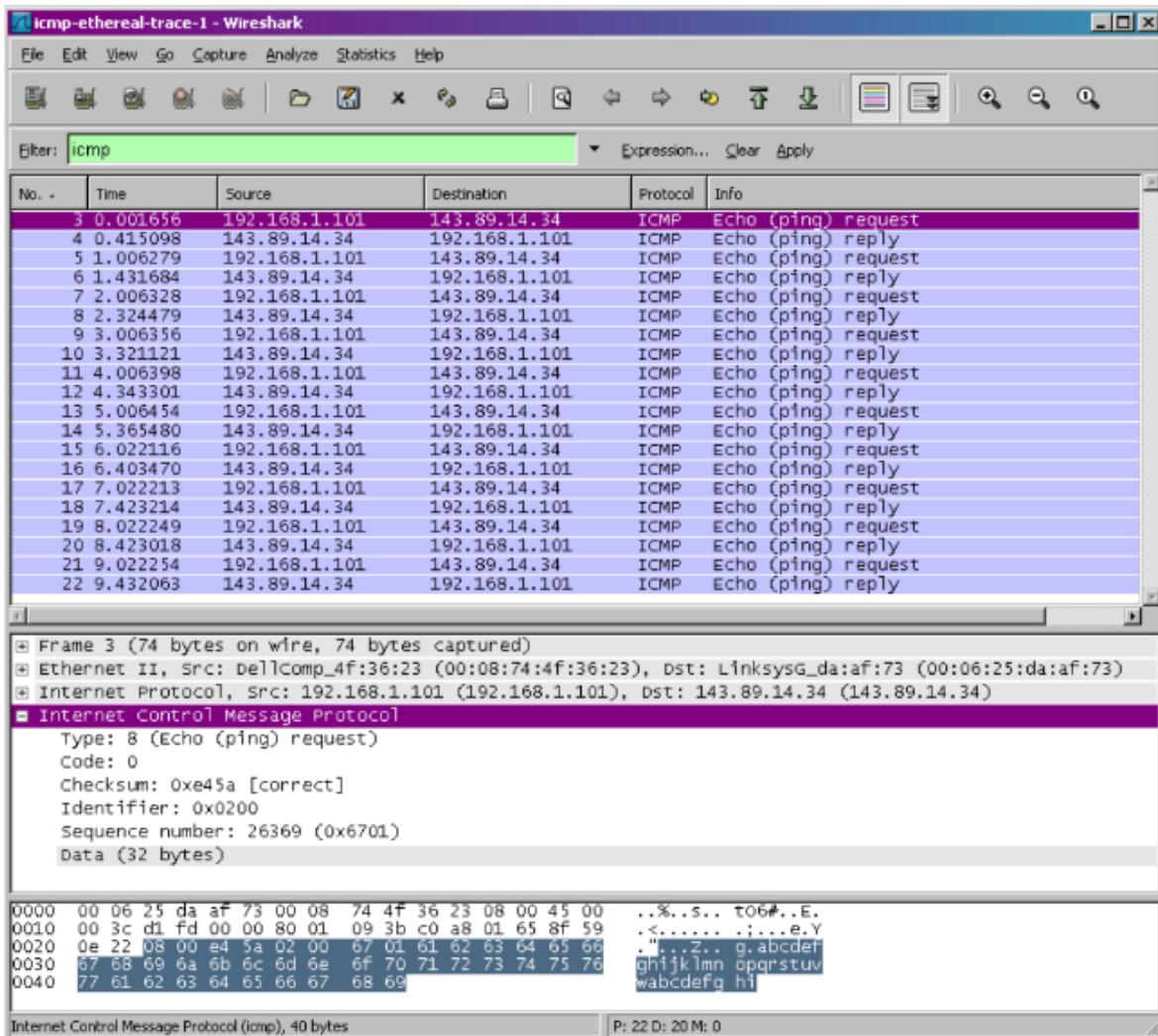


Figure 12.3 Wireshark capture of ping packet with ICMP packet expanded.

### 12.3 ICMP dan Traceroute

Sekarang mari kita lanjutkan petualangan ICMP kita dengan menangkap paket yang dihasilkan oleh program Traceroute. Anda mungkin ingat bahwa program Traceroute dapat digunakan untuk mengetahui jalur yang diambil paket dari sumber ke tujuan.

Traceroute diimplementasikan dengan cara yang berbeda di Unix/Linux/MacOS dan di Windows. Di Unix/Linux, sumber mengirimkan serangkaian paket UDP ke tujuan target menggunakan nomor port tujuan yang tidak mungkin; di Windows, sumber mengirimkan serangkaian paket ICMP ke tujuan target. Untuk kedua sistem operasi, program mengirimkan paket pertama dengan TTL=1, paket kedua dengan TTL=2, dan seterusnya. Ingatlah bahwa router akan mengurangi nilai TTL paket saat paket melewati router. Ketika sebuah paket tiba di router dengan TTL=1, router mengirimkan paket kesalahan ICMP kembali ke sumbernya. Berikut ini, kami akan menggunakan program tracert Windows asli. Versi shareware dari program Windows Traceroute yang sangat bagus adalah pingplotter ([www.pingplotter.com](http://www.pingplotter.com)). Kami akan menggunakan pingplotter di lab IP Wireshark kami karena ini menyediakan fungsionalitas tambahan yang kami perlukan di sana.

Lakukan hal berikut:

- Mari kita mulai dengan membuka aplikasi Windows Command Prompt (yang dapat ditemukan di folder Aksesoris Anda).
- Jalankan packet sniffer Wireshark, dan mulai pengambilan paket Wireshark.
- Perintah tracert ada di c:\windows\system32, jadi ketikkan "tracert hostname" atau "c:\windows\system32\tracert hostname" di baris perintah MS-DOS (tanpa tanda kutip), di mana nama host adalah host di benua lain. (Perhatikan bahwa pada mesin Windows, perintahnya adalah "tracert" dan bukan "traceroute".) Jika Anda berada di luar Eropa, Anda mungkin ingin memasukkan www.inria.fr untuk server Web di INRIA, sebuah penelitian ilmu komputer institut di Prancis. Kemudian jalankan program Traceroute dengan mengetikkan return.
- Saat program Traceroute berakhir, hentikan pengambilan paket di Wireshark.

Di akhir percobaan, Jendela Command Prompt Anda akan terlihat seperti Gambar 12.4. Pada gambar ini, program Traceroute klien berada di Massachusetts dan tujuan target berada di Prancis. Dari gambar ini kita melihat bahwa untuk setiap nilai TTL, program sumber mengirimkan tiga paket probe. Traceroute menampilkan RTT untuk setiap paket probe, serta alamat IP (dan mungkin nama) dari router yang mengembalikan pesan ICMP TTL-exceeded

```

C:\ Command Prompt
C:\WINDOWS\SYSTEM32>
C:\WINDOWS\SYSTEM32>
C:\WINDOWS\SYSTEM32>tracert www.inria.fr

Tracing route to www.inria.fr [138.96.146.2]
over a maximum of 30 hops:

 1  13 ms    12 ms    13 ms  10.216.228.1
 2  21 ms    14 ms    13 ms  24.218.0.153
 3  12 ms    11 ms    13 ms  bar01-p4-0.wsfldhei.ma.attbb.net [24.128.190.197]
 4  16 ms    16 ms    15 ms  bar02-p6-0.ndhmhei.ma.attbb.net [24.128.0.101]
 5  15 ms    15 ms    15 ms  12.125.47.49
 6  17 ms    17 ms    17 ms  12.123.40.218
 7  22 ms    23 ms    22 ms  tbr2-cl1.n54ny.ip.att.net [12.122.10.22]
 8  23 ms    23 ms    23 ms  ggr2-p3120.n54ny.ip.att.net [12.123.3.1091]
 9  26 ms    21 ms    25 ms  att-gw.nyc.opentransit.net [192.205.32.138]
10  98 ms    98 ms    96 ms  P4-0.PASCR1.Pastourelle.opentransit.net [193.251.241.133]
11  97 ms    98 ms    98 ms  P9-0.AUUCR1.Aubervilliers.opentransit.net [193.251.243.29]
12  98 ms    98 ms    108 ms  P6-0.BAGCR1.Bagnolet.opentransit.net [193.251.241.93]
13  104 ms   106 ms   103 ms  193.51.185.30
14  114 ms   114 ms   117 ms  grenoble-pos1-0.cssi.renater.fr [193.51.179.238]
15  114 ms   115 ms   114 ms  nice-pos2-0.cssi.renater.fr [193.51.180.34]
16  129 ms   114 ms   118 ms  inria-nice.cssi.renater.fr [193.51.181.137]
17  113 ms   114 ms   112 ms  www.inria.fr [138.96.146.2]

Trace complete.

C:\WINDOWS\SYSTEM32>

```

Figure 12.4 Jendela Command Prompt menampilkan hasil program Traceroute

Gambar 12.5 menampilkan jendela Wireshark untuk paket ICMP yang dikembalikan oleh router. Perhatikan bahwa paket kesalahan ICMP ini berisi lebih banyak bidang daripada pesan Ping ICMP.

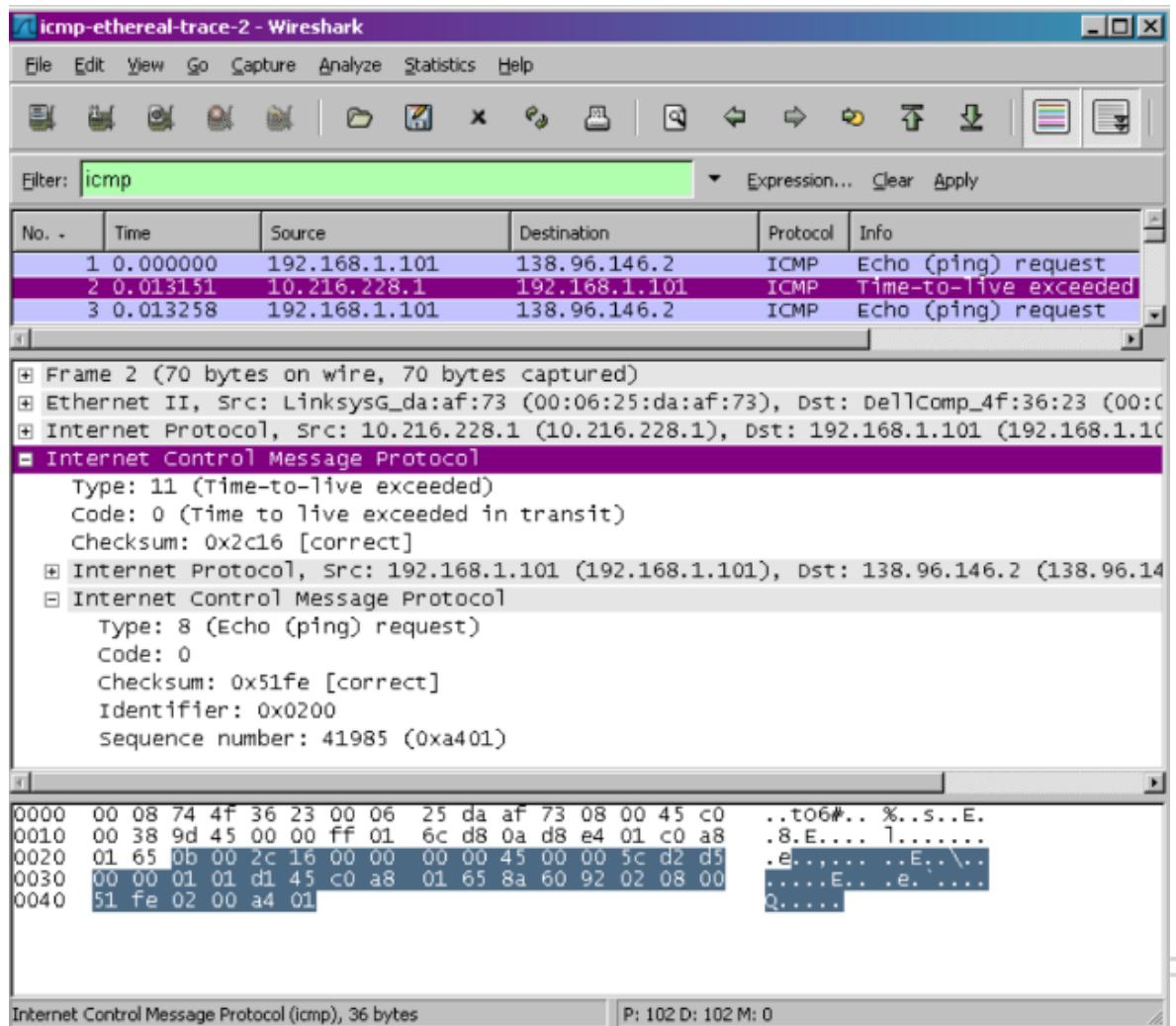


Figure 12.5 Jendela wireshark bidang ICMP diperluas untuk satu paket kesalahan ICMP.

# Modul 13 Ethernet and ARP

## Tujuan Praktikum

1. Mahasiswa dapat menginvestigasi cara kerja Ethernet dan ARP menggunakan Wireshark

## 13.1 Pengantar

Di lab ini, kami akan menyelidiki protokol Ethernet dan protokol ARP. RFC 826 (<ftp://ftp.rfc-editor.org/in-notes/std/std37.txt>) berisi detail mengerikan dari protokol ARP, yang digunakan oleh perangkat IP untuk menentukan alamat IP dari antarmuka jarak jauh yang alamat Ethernetnya diketahui.

## 13.2 Menangkap dan menganalisis frame Ethernet

Mari kita mulai dengan menangkap satu set frame Ethernet untuk dipelajari. Untuk melakukan ini, tentu saja, Anda memerlukan akses ke koneksi Ethernet kabel untuk PC atau Mac Anda. Jika Anda tidak dapat menjalankan Wireshark pada koneksi Ethernet langsung, Anda dapat mengunduh jejak paket yang diambil saat mengikuti langkah-langkah di bawah ini di salah satu komputer.

Lakukan hal berikut:

- Pertama, pastikan cache browser Anda dari dokumen yang diunduh sebelumnya kosong.
- Jalankan Wireshark dan masukkan URL berikut ke browser Anda: <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html>. Browser Anda akan menampilkan Bill of Rights AS yang agak panjang.
- Hentikan pengambilan paket Wireshark.

Pertama, temukan nomor paket (kolom paling kiri di jendela Wireshark atas) dari pesan HTTP GET yang dikirim dari komputer Anda ke [gaia.cs.umass.edu](http://gaia.cs.umass.edu), serta awal dari pesan respons HTTP yang dikirim ke komputer Anda oleh [gaia.cs.umass.edu](http://gaia.cs.umass.edu). Anda akan melihat layar yang terlihat seperti ini (di mana paket 126 pada tangkapan layar di bawah ini berisi pesan HTTP GET)

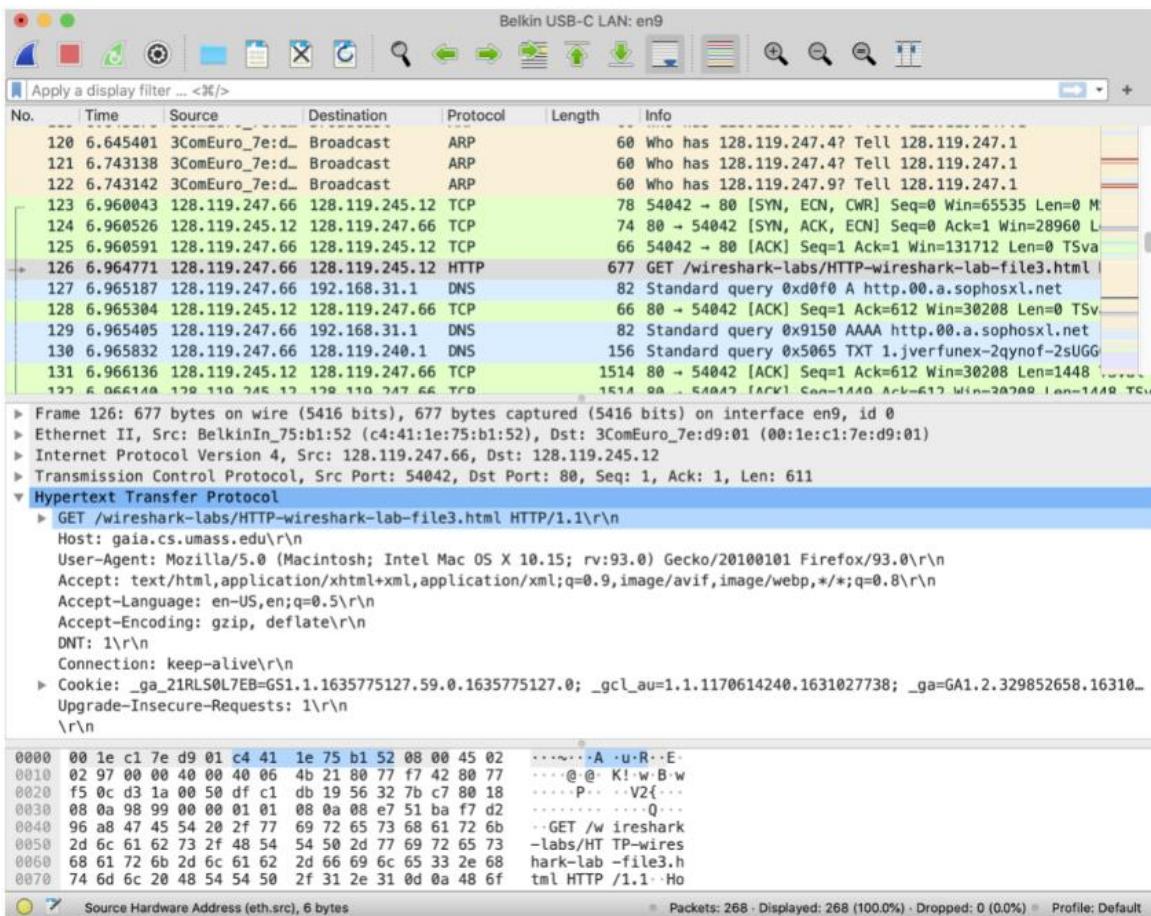


Figure 13.1 Tampilan Wireshark menampilkan pesan HTTP GET untuk <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-lab-file3.html>

Karena lab ini membahas tentang Ethernet dan ARP, kami tidak tertarik dengan protokol tingkat tinggi seperti IP, TCP, atau HTTP. Kami tertarik dengan frame Ethernet dan pesan ARP!

Mari kita mulai dengan melihat frame Ethernet yang berisi pesan HTTP GET. (Ingat bahwa pesan HTTP GET dibawa di dalam segmen TCP, yang dibawa di dalam datagram IP, yang dibawa di dalam bingkai Ethernet; Perluas informasi Ethernet II di jendela detail paket. Perhatikan bahwa isi dari Frame Ethernet (header serta payload) ditampilkan di jendela isi paket. Tampilan Anda akan terlihat seperti yang ditunjukkan pada Gambar 13.2.

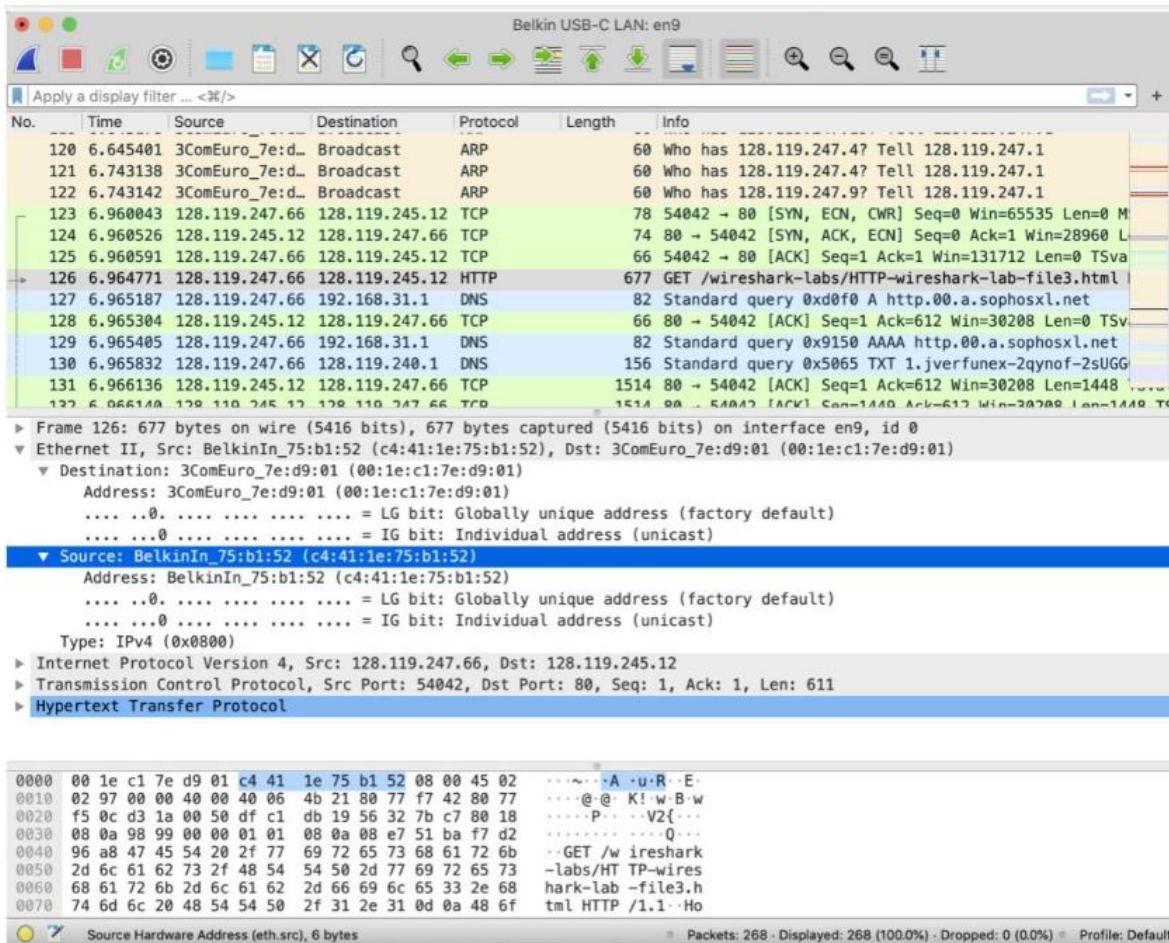


Figure 13.2 Tampilan Wireshark yang menampilkan detail frame Ethernet yang berisi permintaan HTTP GET

### 13.3 Address Resolution Protocol

Di bagian ini, kita akan mengamati protokol ARP bekerja.

#### 13.3.1 Caching ARP

Ingatlah bahwa protokol ARP biasanya menyimpan cache pasangan terjemahan alamat IP-ke-Ethernet di komputer Anda. Perintah arp (dalam DOS, MacOS dan Linux) digunakan untuk melihat dan memanipulasi isi cache ini. Karena perintah arp dan protokol ARP memiliki nama yang sama, sangat mudah untuk membingungkan mereka. Namun perlu diingat bahwa mereka berbeda - perintah arp digunakan untuk melihat dan memanipulasi isi cache ARP, sedangkan protokol ARP mendefinisikan format dan arti dari pesan yang dikirim dan diterima, dan mendefinisikan tindakan yang diambil pada pengiriman dan penerimaan pesan ARP .

Mari kita lihat isi cache ARP di komputer Anda. Di DOS, MacOS, dan Linux, perintah "arp -a" akan menampilkan konten cache ARP di komputer Anda. Jadi pada baris perintah, ketik "arp -a". Hasil memasukkan perintah ini pada salah satu komputer penulis ditunjukkan pada Gambar 13.3.

```
[kurose@noho4 ~ %
[kurose@noho4 ~ %
[kurose@noho4 ~ % arp -a
gw-vlan-2471.cs.umass.edu (128.119.247.1) at 0:1e:c1:7e:d9:1 on en9 ifscope [ethernet]
sammac.cs.umass.edu (128.119.247.19) at (incomplete) on en9 ifscope [ethernet]
robomac.cs.umass.edu (128.119.247.79) at 78:7b:8a:ac:ad:e1 on en9 ifscope [ethernet]
[kurose@noho4 ~ %
[kurose@noho4 ~ %
[kurose@noho4 ~ %
[kurose@noho4 ~ %
```

Figure 13.3 Menjalankan perintah “arp -a” dari salah satu komputer penulis

Untuk mengamati komputer Anda mengirim dan menerima pesan ARP, kami perlu mengosongkan cache ARP, karena jika tidak, komputer Anda kemungkinan besar akan menemukan pasangan terjemahan alamat IP-Ethernet yang diperlukan dalam cache-nya dan akibatnya tidak perlu mengirimkan ARP pesan. Perintah “arp -d -a” akan menghapus cache ARP Anda menggunakan baris perintah. Untuk menjalankan perintah ini di mesin Mac atau Linux, Anda memerlukan hak akses root atau menggunakan sudo. Jika Anda tidak memiliki hak akses root dan tidak dapat menjalankan Wireshark pada mesin Windows, Anda dapat melewati bagian pengumpulan jejak lab ini dan menggunakan file jejak ethernet-wireshark-trace1 yang dibahas sebelumnya.

### 13.3.2 Mengamati Aksi ARP

Lakukan hal berikut:

- Kosongkan cache ARP Anda, seperti dijelaskan di atas dan pastikan cache browser Anda dibersihkan dari dokumen yang diunduh sebelumnya.
- Mulai sniffer paket Wireshark.
- Masukkan URL berikut ke dalam peramban Anda: <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-lab-file3.html> Peramban Anda akan kembali menampilkan Bill of Rights AS yang agak panjang.
- Hentikan pengambilan paket Wireshark.

Sekali lagi, kami tidak tertarik dengan IP atau protokol lapisan yang lebih tinggi, jadi mari kita lihat paket ARP. Tampilan Anda akan terlihat seperti yang ditunjukkan pada Gambar 14.4 (perhatikan bahwa kita telah memasukkan "arp" ke jendela filter tampilan di bagian atas layar Wireshark).

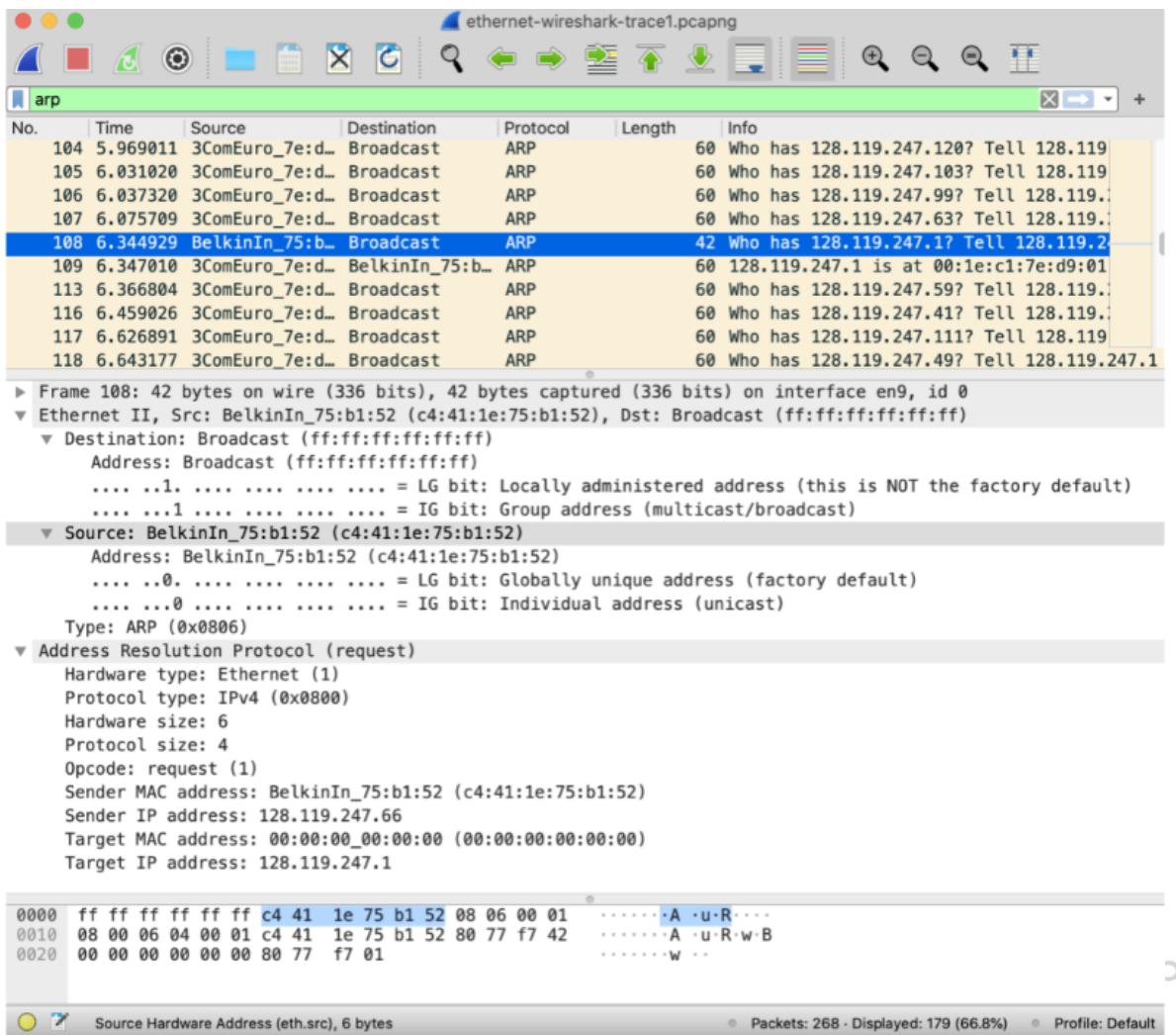


Figure 13.4 Permintaan ARP yang disiarkan dari salah satu komputer

Mari kita mulai dengan melihat frame Ethernet yang berisi pesan ARP

Sekarang mari kita gali lebih dalam lagi ke dalam pesan ARP itu sendiri. Untuk menjawab pertanyaan ini, Anda perlu menggali ARP. RFC asli (<https://datatracker.ietf.org/doc/html/rfc826>) yang mendefinisikan ARP agak sulit dibaca. Entri Wikipedia untuk ARP cukup bagus: [https://en.wikipedia.org/wiki/Address\\_Resolution\\_Protocol](https://en.wikipedia.org/wiki/Address_Resolution_Protocol).

### 13.3.3 Kredit Tambahan

EX-1. Perintah arp:

```
arp -s InetAddr EtherAddr
```

memungkinkan Anda untuk secara manual menambahkan entri ke cache ARP yang menyelesaikan alamat IP InetAddr ke alamat fisik EtherAddr. Apa yang akan terjadi jika, ketika Anda menambahkan entri secara manual, Anda memasukkan alamat IP yang benar, tetapi alamat Ethernet yang salah untuk antarmuka jarak jauh itu? Serangan keamanan yang dikenal sebagai "keracunan ARP" (<https://www.varonis.com/blog/arp-poisoning/>) memalsukan pesan ARP dan menyebabkan entri yang salah dibuat ke dalam tabel ARP!

EX-2. Berapa lama waktu default entri tetap berada di cache ARP Anda sebelum dihapus? Anda dapat menentukan ini secara empiris (dengan memantau isi cache) atau dengan melihat ini di dokumentasi sistem operasi Anda. Tunjukkan bagaimana/di mana Anda menentukan nilai ini.



Fakultas Informatika  
School of Computing  
Telkom University



## Modul 14 Presentasi Tugas Besar

<b>Tujuan Praktikum</b>
1. Mahasiswa dapat mempresentasikan hasil penggerjaan tugas besar kepada dosen / asisten



Pada modul 14, kalian akan diminta mendemokan dan mempresentasikan hasil aplikasi yang telah dibuat kepada dosen/asisten praktikum yang telah ditetapkan, serta mengumpulkan laporan tugas besar yang telah dibuat.



**Fakultas Informatika**  
School of Computing  
Telkom University



# Modul 15 802.11 WiFi

## Tujuan Praktikum

1. Mahasiswa dapat menginvestigasi cara kerja WiFi menggunakan Wireshark

## 15.1 Pengantar

Di lab ini, kami akan menyelidiki protokol jaringan nirkabel 802.11. Karena kita akan menggali lebih dalam 802.11 daripada yang tercakup dalam teks, Anda mungkin ingin melihat "Tutorial Teknis tentang Protokol 802.11," oleh Pablo Brenner (Breezecom Communications), [http://www.sss-mag.com/pdf/802\\_11tut.pdf](http://www.sss-mag.com/pdf/802_11tut.pdf), dan "Memahami Jenis Bingkai 802.11," oleh Jim Geier, <http://www.wi-fiplanet.com/tutorials/article.php/1447501>. Dan, tentu saja, ada "Alkitab" 802.11 - standar itu sendiri, "ANSI/IEEE Std 802.11, Edisi 1999 (R2003)," <http://gaia.cs.umass.edu/wireshark-labs/802.11-1999.pdf>. Secara khusus, Anda mungkin menemukan Tabel 1 pada halaman 36 dari standar yang sangat berguna ketika melihat melalui jejak nirkabel. Di semua lab Wireshark sejauh ini, kami telah menangkap frame pada koneksi Ethernet kabel. Di sini, karena 802.11 adalah protokol lapisan tautan nirkabel, kami akan menangkap bingkai "di udara." Sayangnya, beberapa driver perangkat untuk NIC 802.11 nirkabel masih tidak menyediakan kait untuk menangkap/menyalin frame 802.11 yang diterima untuk digunakan di Wireshark. Jadi, di lab ini, kami akan memberikan jejak dari frame 802.11 yang diambil untuk Anda analisis dan asumsikan dalam pertanyaan di bawah ini bahwa Anda menggunakan jejak ini. Jika Anda dapat menangkap bingkai 802.11 menggunakan versi Wireshark, Anda dapat melakukannya.

## 15.2 Starting

Download file zip <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip> dan ekstrak file Wireshark\_802\_11.pcap. Jejak ini dikumpulkan menggunakan AirPcap dan Wireshark yang berjalan di komputer di jaringan rumah salah satu penulis, yang terdiri dari titik akses/router gabungan Linksys 802.11g, dengan dua PC berkabel dan satu PC host nirkabel yang terpasang ke titik

akses/router. Dalam file jejak ini, kita akan melihat bingkai ditangkap di saluran 6. Aktivitas host nirkabel yang diambil dalam file jejak adalah:

- Host sudah diasosiasikan dengan 30 Munroe St AP saat pelacakan dimulai.
- Pada t = 24,82, host membuat permintaan HTTP ke <http://gaia.cs.umass.edu/wireshark-labs/alice.txt>. Alamat IP gaia.cs.umass.edu adalah 128.119.245.12.
- Pada t=32.82, host membuat permintaan HTTP ke <http://www.cs.umass.edu>, dengan alamat IP 128.119.240.19.
- Pada t = 49,58, host memutuskan sambungan dari 30 Munroe St AP dan mencoba menyambung ke linksys\_ses\_24086. Ini bukan titik akses terbuka, dan host akhirnya tidak dapat terhubung ke AP ini.
- Pada t=63.0 tuan rumah menyerah mencoba untuk mengasosiasikan dengan linksys\_ses\_24086 AP, dan mengasosiasikan lagi dengan titik akses 30 Munroe St.

Setelah Anda mengunduh jejak, Anda dapat memuatnya ke Wireshark dan melihat jejak menggunakan menu tarik-turun File, pilih Buka, lalu pilih file jejak Wireshark\_802\_11.pcap. Tampilan yang dihasilkan akan terlihat seperti Figure 15.1.

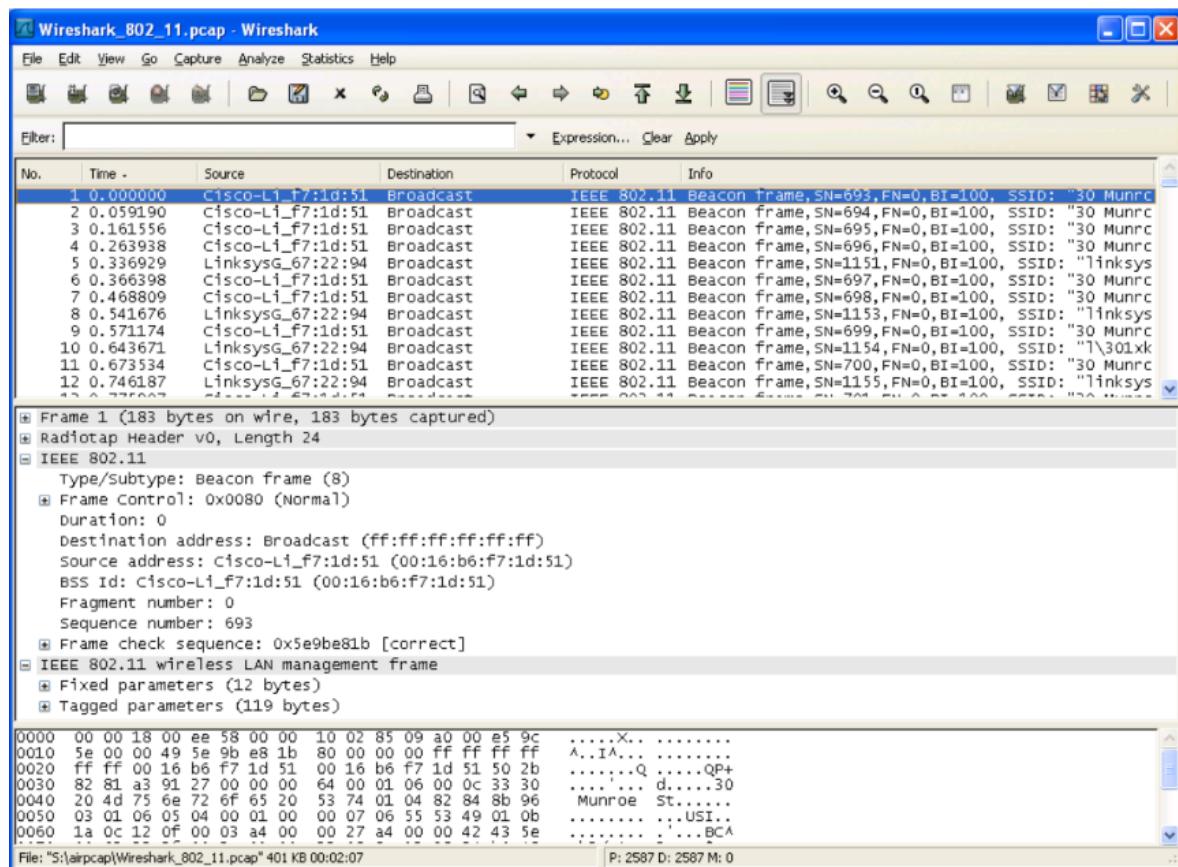


Figure 15.1 Wireshark window, setelah membuka file Wireshark\_802\_11.pcap

### 15.3 Beacon Frames

Ingatlah bahwa bingkai suar digunakan oleh 802.11 AP untuk mengiklankan keberadaannya. Untuk menjawab beberapa pertanyaan di bawah ini, Anda dapat melihat detail frame dan subbidang “IEEE 802.11” di jendela Wireshark tengah. Bila memungkinkan, saat menjawab pertanyaan di bawah, Anda harus menyerahkan cetakan paket dalam jejak yang Anda gunakan untuk menjawab pertanyaan yang diajukan. Beri anotasi pada printout2 untuk menjelaskan jawaban Anda. Untuk

mencetak paket, gunakan File->Print, pilih Selected packet only, pilih Packet summary line, dan pilih jumlah minimum detail paket yang Anda perlukan

## 15.4 Data Transfer

Karena pelacakan dimulai dengan host yang sudah diasosiasikan dengan AP, pertama-tama mari kita lihat transfer data melalui asosiasi 802.11 sebelum melihat asosiasi/disasosiasi AP. Ingat bahwa dalam jejak ini, pada  $t = 24,82$ , host membuat permintaan HTTP ke <http://gaia.cs.umass.edu/wireshark-labs/alice.txt>. Alamat IP gaia.cs.umass.edu adalah 128.119.245.12. Kemudian, pada  $t = 32.82$ , host membuat permintaan HTTP ke <http://www.cs.umass.edu>.

## 15.5 Association/Disassociation

Asosiasi di 802.11 dilakukan menggunakan frame ASSOCIATE REQUEST (dikirim dari host ke AP, dengan tipe frame 0 dan subtipe 0) dan frame ASSOCIATE RESPONSE (dikirim oleh AP ke host dengan tipe frame 0 dan subtipe 1, di tanggapan atas PERMINTAAN ASOSIASI yang diterima). Untuk penjelasan rinci setiap bidang dalam bingkai 802.11, lihat halaman 34 (Bagian 7) dari spesifikasi 802.11 di <http://gaia.cs.umass.edu/wireshark-labs/802.11-1999.pdf>.



Fakultas Informatika  
School of Computing  
Telkom University



## Modul 16 ASISTENSI 2 (PRAKTIKUM SUSULAN)

Tujuan Praktikum
1. Mahasiswa dapat menginvestigasi cara kerja WiFi menggunakan Wireshark

Pada modul ini, praktikan yang tidak hadir pada praktikum di modul tertentu diberikan kesempatan untuk melakukan praktikum susulan sesuai dengan kehadiran selama mengikuti kelas praktikum.



Fakultas Informatika  
School of Computing  
Telkom University



## Daftar Pustaka

- [1] James F. Kurose, Keith Ross, Computer Networking: A Top-Down Approach, 8th ed, Pearson, 2021
- [2] Computer Networking: A Top-Down Approach Companion website  
[http://gaia.cs.umass.edu/kurose\\_ross/index.php](http://gaia.cs.umass.edu/kurose_ross/index.php)



Fakultas Informatika  
School of Computing  
Telkom University



## Daftar Perubahan

### 22 September 2022

Penerbitan Modul Praktikum Jaringan Komputer – S1 PJJ Informatika Ganjil 22/23 berbasis materi dan buku *Computer Networking: A Top-Down Approach*

### 22 Februari 2023

Penerbitan Modul Praktikum Jaringan Komputer – S1 Informatika Genap 22/23 berbasis materi dan buku *Computer Networking: A Top-Down Approach*



Fakultas Informatika  
School of Computing  
Telkom University





**Kontak Kami :**

-  @fiflab
-  Praktikum IF LAB
-  informaticslab@telkomuniversity.ac.id
-  informatics.labs.telkomuniversity.ac.id