Weekly Log
Week 4

Jonathan Filbert
JARKOMDAT - A
1806191692

Topic: Transport Layer (TCP, UDP), Mux and Demux process

Transport layer adalah layer dari network stack yang memiliki tugas untuk membungkus packet yang dikirimkan dari application layer untuk nantinya dikirim oleh network layer.
Transport layer dapat berada pada sender dan receiver:

1. Sender
   Di sisi sender, TCP memiliki tugas untuk mempersiapkan packet yang ingin dikirim oleh application layer untuk dikirim. Packet ini akan dipecah menjadi beberapa segment dan nantinya akan dibungkus dan diberikan header sesuai dengan protokol pengirimannya, lalu kemudian akan dikirimkan ke network layer.
2. Receiver
   Di sisi penerima, transport layer memiliki tugas untuk mengumpulkan kembalii packet packet yang diterima oleh network layer untuk kemudian disatukan menjadi packet yang akan dikirim ke application layer untuk digunakan oleh user.

Transport Layer memiliki 2 protokol utama, yaitu TCP dan UDP.

1. TCP

TCP atau disebut *transmission control protocol* adalah sebuah protokol utnuk mengirimkan data dengan sifat **in order** dan **reliable.** Artinya TCP mengirimkan data dengan berurutan serta TCP akan menjamin bahwa semua packet yang dikirimkan akan diterima 100%.

TCP memiliki beberapa protokol yang akan dilakukan sehingga dapat menjamin reliabilitas:
1. Congestion Control
2. Flow Control
3. Connection Setup (Handshake)

Sifat dari TCP adalah:
1. One sender - one receiver
2. Reliable
3, Flow control
4. Bidirectional data flow
5. Connection-oriented (handshake needed)

TCP banyak dipakai jika kita membutuhkan koneksi yang datanya tidak boleh loss, contohnya adalah HTTP dan FTP.

2. UDP

UDP adalah protokol pembungkusan data yang dimana, berbeda dengan TCP, yang tidak mementingkan reliability, tetapi mementingkan **kecepatan.** Berbeda dengan TCP, karena fokus daripada UDP adalah kecepatan, maka berbagai protokol yang ada di TCP tidak ada di UDP, contohnya adalah flow control dan handshake.

UDP karena cepat dan ada kemungkinan data loss, maka cocok digunakan untuk streaming multimedia dan SNMP.

TCP dan UDP memiliki protokol checksum, yaitu dia akan menambahkan value dari data yang dikirim, kemudian akan dikomplemenkan dan dimasukan ke header dari packet yang dikirim. Kemudian, host yang menerima, akan menambahkan value dari data yang diteima, kemudian akan ditambahkan kepada value checksum yang ada di header (komplemennya), **bila hasilnya tidak 1 semua, maka ada data loss.**

Misal:
yang dikirim: 1010

Jonathan Filbert
JARKOMDAT - A
1806191692

Checksum yang dikirim: 0101 (**komplemen dari 1010**)

Yang diteima: 1000
Checksum yang direima: 0101
Penambahan:
1000
0101
———+
1101 => **ada yang tidak 0, artinya ada data loss.**

3. Mux and Demux

Multiplexing adalah proses yang terjadi pada host yang ingin mengirimkan data. Sebelum sebuah packet dikirimkan, maka packet tersebut akan dibungkus oleh header yang menyimpan berbagai informasi seperti asal port, asal IP, port tujuan, dan IP tujuan, lalu kemudian akan diteruskan ke network layer.

Demultiplexing adalah proses yang terjadi pada host yang menerima data dimana dia akan menerima sebuah packet yang sudah dibungkus dan kemudian akan membuka bungkusan data tersebut untuk mengirimkan dat tersebut ke socket proses yang benar.

Mux and Demux di TCP dan UDP berbeda karena natur dari koneksi yang dibuat oleh TCP dan UDP.

Di TCP, packet yang ditujukan kepada port yang sama, akan diterima oleh socket yang berbeda. Jadi bila ada 3 koneksi yang ditujukan kepada sebuah port HTTP di sebuah webserver (port 80), maka port 80 di mesin tersebut akan membuka 3 socket berbeda untuk memfasilitasi 3 koneksi tersebut. Karena natur ini, maka koneksi yang terjadi bersifat **bidirectional,** yaitu koneksi yang ada dari client dan socket yang ada di server nersifat dua arah.
Karena setiap request yang tujuan port nya sama, akan dibuatkan socket baru, maka setiap socket yang dibuat di TCP akan memiliki 4 tuple identifier:
source IP, source socket, destination IP, dan **destination socket.**

Di UDP, packet yang ditujukan kepada port yang sama, akan diterima oleh socket yang sama, sehingga koneksi yang terjadi adalah **unidirectional**, yaitu dari client ke server bersifat searah, sedangkan server ke client bersifat searah. Jadi misal ada 2 proses yang sama sama menargetkan sebuah port di server, maka kedua proses tersebut akan terjadi di socket yang sama.