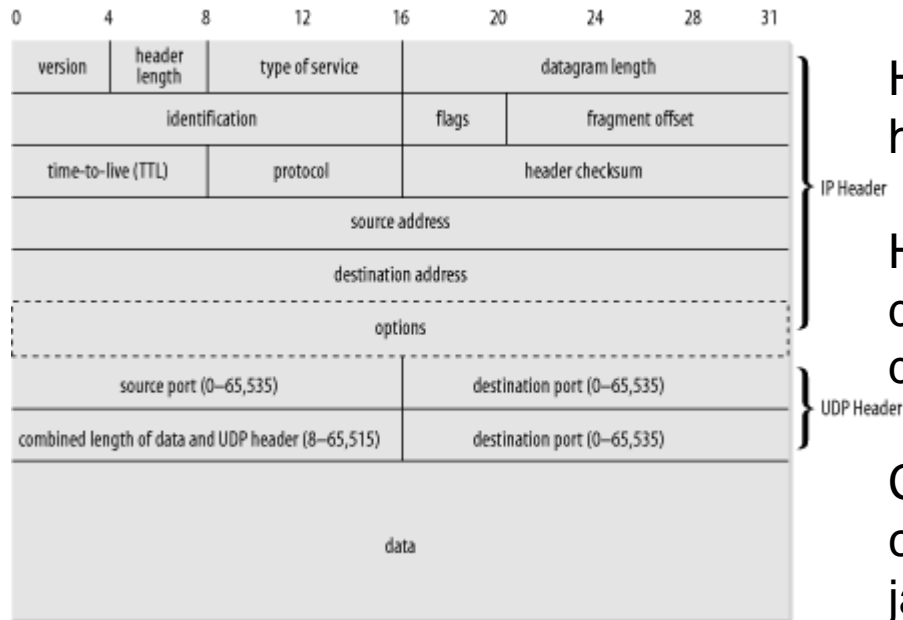


Pemrograman Jaringan 9

Connectionless Oriented Programming

UDP



Header UDP menambah 8 bytes di header IP.

Header UDP berisi source dan destination nomor port, panjang data, dan checksum yang bersifat opsional.

Checksum bersifat opsional sebab jika checksum tidak sesuai maka program jaringan akan menolak datagram tanpa memberitahukan kepada penerima maupun pengirim.

UDP merupakan protokol yang tidak reliable.

Kelas java.net.DatagramSocket

- Kelas ini mengirim dan menerima DatagramPacket dari atau ke jaringan.

Constructor:

- DatagramSocket(int port)
 - Kelas ini dapat digunakan untuk menyatakan penggunaan suatu nomor port sebagai "pintu" untuk menerima koneksi dari client.
- DatagramSocket(int port, InetAddress laddr)
 - Kelas ini membentuk koneksi dengan protokol UDP pada alamat IP lokal tertentu dan pada nomor port tertentu.
- DatagramSocket()
 - Kelas ini membentuk koneksi dengan protokol UDP pada alamat IP lokal host dengan penentuan nomor portnya secara random berdasar tersedianya nomor port yang dapat digunakan.

Kelas java.net.DatagramPacket

- Merupakan kelas yang menyatakan atau mewakili sebuah paket informasi, yaitu sebuah array byte yang dipertukarkan pada jaringan.

Constructor:

- DatagramPacket(byte[] buf, int length)
 - Kelas ini dapat digunakan untuk mengambil informasi. Constructor ini membutuhkan sebuah array byte yang menjadi parameter pertama, yang berfungsi untuk menyimpan data dan informasi ukuran data yang diterima.
- DatagramPacket(byte[] buf, int length, InetAddress address, int port)
 - Constructor ini digunakan untuk membuat paket Datagram yang akan mengirim data. Constructor ini memerlukan informasi array byte yang akan dikirim dan panjangnya, serta alamat dan port yang dituju.

Methods

- `getData()` untuk mengambil informasi data
- `getLength()` untuk mengambil panjang datagram
- `getAddress()` untuk mengambil alamat IP
- `getPort()` untuk mengambil alamat port

Methods

- `send(DatagramPacket data)` akan mengirim `DatagramPacket` ke host dan port yang dituju
- `receive(DatagramPacket data)` akan memblok eksekusi sampai suatu paket lengkap diterima

Exception

- `public class SocketException extends IOException`
 - Kelas ini merupakan kelas yang diturunkan dari kelas `IOException`. Kelas exception ini dipanggil atau dipicu ketika ada kegagalan dalam pemakaian socket, sebagai contoh adalah kegagalan dalam protokol TCP. Salah satu penyebabnya yang mungkin terjadi adalah ketika port yang akan digunakan sudah digunakan sebelumnya pada localhost. Penyebab yang lain adalah user tidak dapat melakukan bind ke port yang dituju. Misalnya saja, Anda ingin menggunakan port 80 untuk aplikasi Anda, namun ternyata pada komputer Anda tersebut sudah berjalan HTTP Server yang juga menggunakan port 80. Bila hal ini terjadi, maka JVM akan melemparkan kegagalan yang ada ke kelas exception `SocketException`.

Exception (2)

- `public class BindException extends`
 - SocketException Kelas ini akan dipanggil ketika ada port lokal yang akan digunakan sudah terpakai oleh yang lain, atau ada kegagalan dalam permintaan untuk menggunakan alamat.
- `public class ConnectException extends`
 - SocketException Kelas ini akan dipanggil ketika sebuah koneksi ditolak oleh host yang dituju, oleh karena tidak ada proses yang siap menerima data pada port yang dituju.

Exception (3)

- public class NoRouteToHostException extends SocketException
 - Koneksi yang akan dibangun tidak dapat dipenuhi oleh karena melebihi waktu timeout yang tersedia atau host yang dituju tidak dapat dicapai (unreachable).
- public class ProtocolException extends IOException
 - Terjadi ketika data diterima dari network menyalahi aturan TCP/IP

Contoh

- DatagramExample.java

UDP Characteristics

- Tidak ada inisialisasi koneksi, sehingga tidak diperlukan welcoming socket, seperti accept di server dan penginisialisasian socket di client.
- Tidak ada input dan output stream yang dibentuk dari atau ke socket.
- Pengirim akan membuat datagram paket dan mengirimkannya ke proses yang dituju. Setiap paket yang dibentuk akan diberikan informasi mengenai alamat IP dan port untuk setiap deretan byte yang akan dikirim.
- Penerima harus menguraikan paket datagram tersebut agar tahu informasi yang dikirimkan.

Datagram Sockets

SERVER:

1. Create a `DatagramSocket` object

```
DatagramSocket dgramSocket =  
    new DatagramSocket(1234);
```

2. Create a buffer for incoming datagrams

```
byte[] buffer = new byte[256];
```

3. Create a `DatagramPacket` object for the incoming datagram

```
DatagramPacket inPacket =  
    new DatagramPacket(buffer,  
        buffer.length);
```

4. Accept an incoming datagram

```
dgramSocket.receive(inPacket)
```

Datagram Sockets

SERVER:

5. Accept the sender's address and port from the packet

```
InetAddress clientAddress = inPacket.getAddress();  
int clientPort = inPacket.getPort();
```
6. Retrieve the data from the buffer

```
String message =  
    new String(inPacket.getData(), 0,  
inPacket.getLength());
```
7. Create the response datagram

```
DatagramPacket outPacket =  
    new DatagramPacket(  
        response.getBytes(),  
response.length(),  
clientAddress, clientPort);
```
8. Send the response datagram

```
dgramSocket.send(outPacket)
```
9. Close the *DatagramSocket*: *dgram.close()*;

Datagram Sockets

CLIENT:

1. Create a DatagramSocket object

```
DatagramSocket dgramSocket = new  
DatagramSocket;
```

2. Create the outgoing datagram

```
DatagramPacket outPacket = new  
DatagramPacket(message.getBytes(),  
message.length(), host, port);
```

3. Send the datagram message

```
dgramSocket.send(outPacket)
```

4. Create a buffer for incoming datagrams

```
byte[] buffer = new byte[256];
```

Datagram Sockets

CLIENT:

5. Create a *DatagramPacket* object for the incoming datagram

```
DatagramPacket inPacket =  
    new DatagramPacket(buffer,  
        buffer.length);
```

6. Accept an incoming datagram
dgramSocket.receive(inPacket)
7. Retrieve the data from the buffer
*string response = new
String(inPacket.getData(), 0,
inPacket.getLength());*
8. Close the *DatagramSocket*:
dgram.close();

Prinsip-prinsip yang dilakukan oleh InfoClient

- Buat DatagramSocket
- Lakukan loop sampai user mengetikkan QUIT.
 - Baca masukkan dari user
 - Tampung pada buffer array byte
 - Buat obyek DatagramPacket untuk dikirimkan ke server
 - Kirimkan DatagramPacket ke server
 - Siapkan packet datagram untuk mengambil informasi dari client
 - Baca DatagramPacket yang dikirim dari client
- Setelah Client QUIT, tutup DatagramSocket.

Praktikum

- InfoClientUDP.java
- InfoServerUDP.java