

CS380 — Project 4

February 6, 2017

Due: Wednesday, February 15, 2017 by midnight (50 points)

Description

In this project, you'll be applying the same techniques you used in project 3 to implement IPv6. IPv6 packet headers may be larger than IPv4, but are actually quite a bit simpler! Like project 3, here's a table of things that must be done:

Version	Implement
Traffic Class	Don't implement
Flow Label	Don't implement
Payload length	Implement
Next Header	Implement and set to UDP protocol value
Hop Limit	Implement and set to 20
SourceAddr	Implement assuming it is a valid IPv4 address that has been extended to IPv6 for a device that does not use IPv6
DestinationAddr	Implement assuming it is a valid IPv4 address that has been extended to IPv6 for a device that does not use IPv6 using the IP address of the server you are connecting to.

Send the packets as bytes through the socket's output stream. The host will be the same as the previous project: `codebank.xyz` on port 38004.

For more information about the implementation of IPv6, see your textbook, the RFC specification at <https://www.ietf.org/rfc/rfc2460.txt>, or the Wikipedia article about IPv6: <https://en.wikipedia.org/wiki/IPv6>.

Like project 3, you should generate 12 packets with data size starting at 2 bytes and doubling for each packet.

Unlike project 3, you will not be getting back text messages indicating what went wrong or if the packet was correct. Instead, I'll be sending back a specific response code for each packet you send consisting of a four byte [magic number](#). Here's a table indicating which codes will be sent under various circumstances:

Code	Reason
0xCAFEBAFE	Packet was correct
0xCAFE0000	Incorrect version
0xDEAD0000	One of the "do not implement" fields was not all 0s
0xBBADBEEF	Payload length is wrong
0xFEEDFACE	Next header is wrong
0xFEE1DEAD	Hop limit is wrong
0xDEADCODE	Bad source address
0xABADCAFE	Bad destination address

Therefore, you should be aiming for 0xCAFEBAFE after each packet.

Sample Output

Your program should print each response code in hexadecimal after sending each packet such as:

```
$ java Ipv6Client
data length: 2
Response: 0xCAFEBAFE
```

```
data length: 4
Response: 0xCAFEBAFE
```

```
data length: 8
Response: 0xCAFEBAFE
```

```
data length: 16
Response: 0xCAFEBAFE
```

```
data length: 32
Response: 0xCAFEBAFE
```

```
data length: 64
Response: 0xCAFEBAFE
```

```
data length: 128
Response: 0xCAFEBAFE
```

```
data length: 256
Response: 0xCAFEBAFE
```

```
data length: 512
Response: 0xCAFEBAFE
```

```
data length: 1024
Response: 0xCAFEBAFE
```

```
data length: 2048
Response: 0xCAFEBAFE
```

```
data length: 4096
Response: 0xCAFEBAFE
```

Submission

1. On codebank.xyz, create a project as your user named CS380-P4. Follow this naming convention precisely including case.
2. Make sure you add the reference to the remote repository in your local project with:

```
$ git remote add origin https://codebank.xyz/username/CS380-P4.git
```

3. Your project should have a main class named `Ipv6Client` in a file named `Ipv6Client.java`. You can have other files or classes, but it should successfully compile and run by simply using:

```
$ javac Ipv6Client.java
```

```
$ java Ipv6Client
```