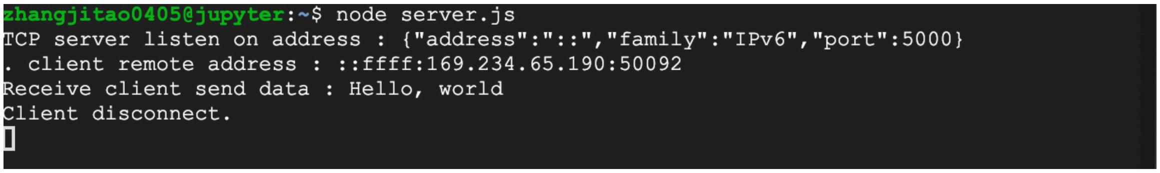
# CompSci-230: Homework 2
# Jitao Zhang

## 1. Programs(Written in Nodejs)

### Server side(in Google Cloud):

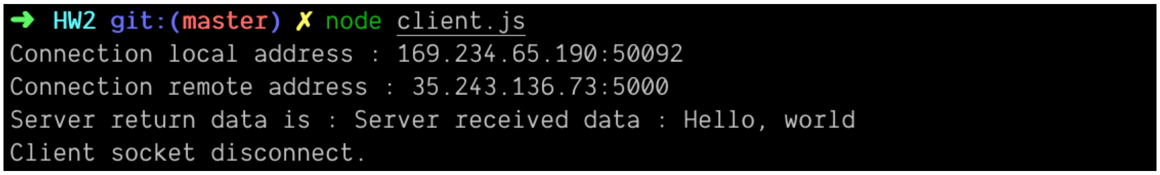
```
// Import builtin net module.
var net = require('net');

// Create and return a net.Server object
var server = net.createServer(function(socket) {

    console.log('. client remote address : ' + socket.remoteAddress +
':' + socket.remotePort);

    // set encoding type
    socket.setEncoding('utf-8');

    // When receive client data.
    socket.on('data', function (data) {

        // Print received client data
        console.log('Receive client send data : ' + data);

        // Server send data back to client use client net.Socket object.
        socket.end('Server received data : ' + data);
    });

    // When client send data complete.
    socket.on('end', function () {
        console.log('Client disconnect.');
    });

    // When client timeout.
    socket.on('timeout', function () {
        console.log('Client request time out. ');
    })
});

// Make the server a TCP server listening on port 5000.
server.listen(5000, function () {

    // Get server address info.
    var serverInfo = server.address();

    console.log('TCP server listen on address : ' +
JSON.stringify(serverInfo));

    server.on('close', function () {
        console.log('TCP server is closed.');
    });
});
```



```
zhangjitao0405@jupyter:~$ node server.js
TCP server listen on address : {"address":"::","family":"IPv6","port":5000}
. client remote address : ::ffff:169.234.65.190:50092
Receive client send data : Hello, world
Client disconnect.
```

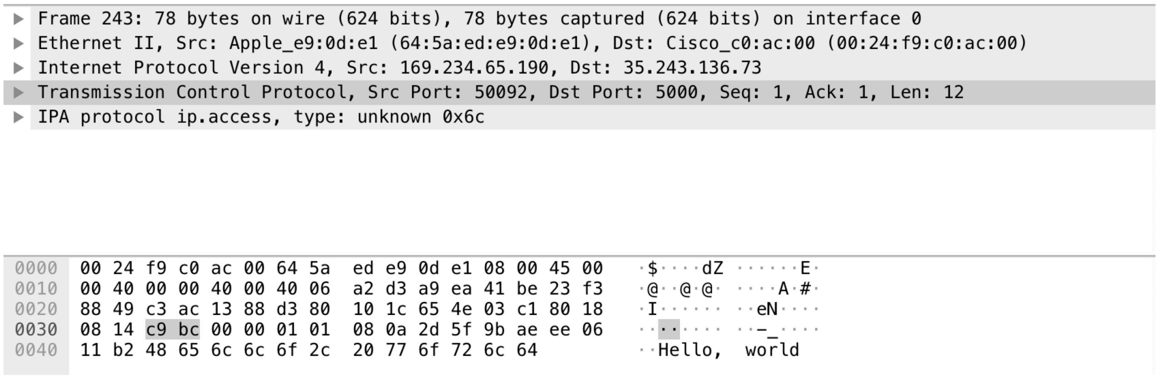
## Client Side(localhost):

```
// Import buildin net module.
var net = require('net');

var option = {
    host:'35.243.136.73',
    port: 5000
}

// Create TCP client.
var client = net.createConnection(option, function () {
    console.log('Connection remote address : ' + client.remoteAddress +
":" + client.remotePort);
    });

    // set the encoding type
    client.setEncoding('utf8');

    // When receive server send back data.
    client.on('data', function (data) {
        console.log('Server return data is : ' + data);
    });

    // When connection disconnected.
    client.on('end',function () {
        console.log('Client socket disconnect. ');
    });

    client.on('error', function (err) {
        console.error(JSON.stringify(err));
    });

    client.write('Hello, world');
```



```
➜ HW2 git:(master) ✗ node client.js
Connection local address : 169.234.65.190:50092
Connection remote address : 35.243.136.73:5000
Server return data is : Server received data : Hello, world
Client socket disconnect.
```

## 2. Packet from sender

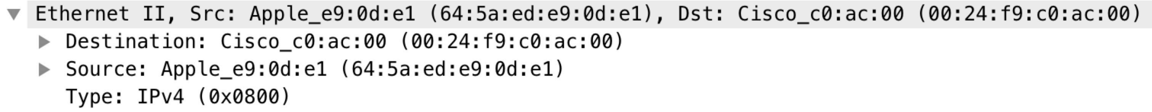


```
▶ Frame 243: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
▶ Ethernet II, Src: Apple_e9:0d:e1 (64:5a:ed:e9:0d:e1), Dst: Cisco_c0:ac:00 (00:24:f9:c0:ac:00)
▶ Internet Protocol Version 4, Src: 169.234.65.190, Dst: 35.243.136.73
▶ Transmission Control Protocol, Src Port: 50092, Dst Port: 5000, Seq: 1, Ack: 1, Len: 12
▶ IPA protocol ip.access, type: unknown 0x6c
```

```
0000  00 24 f9 c0 ac 00 64 5a  ed e9 0d e1 08 00 45 00   ·$····dZ ······E·
0010  00 40 00 00 40 00 40 06  a2 d3 a9 ea 41 be 23 f3   ·@··@·@· ····A·#·
0020  88 49 c3 ac 13 88 d3 80  10 1c 65 4e 03 c1 80 18   ·I······ ··eN····
0030  08 14 c9 bc 00 00 01 01  08 0a 2d 5f 9b ae ee 06   ········ ··-_····
0040  11 b2 48 65 6c 6c 6f 2c  20 77 6f 72 6c 64         ··Hello,  world
```

## 3. Questions

- What is encoded in bytes 0-5 and 6-11?



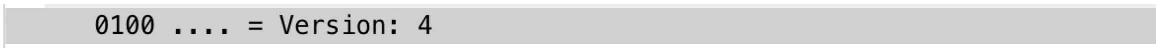
```
▼ Ethernet II, Src: Apple_e9:0d:e1 (64:5a:ed:e9:0d:e1), Dst: Cisco_c0:ac:00 (00:24:f9:c0:ac:00)
  ▶ Destination: Cisco_c0:ac:00 (00:24:f9:c0:ac:00)
  ▶ Source: Apple_e9:0d:e1 (64:5a:ed:e9:0d:e1)
    Type: IPv4 (0x0800)
```

0-5 bytes => Destination MAC address: 00:24:f9:c0:ac:00

6-11 bytes => Source MAC address: 64:5a:ed:e9:0d:e1

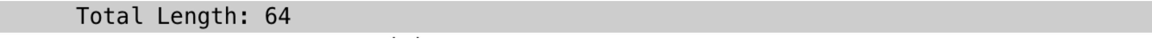
- What is encoded in, and what is the relationship between, byte 14 and the two bytes 16-17?

byte 14 is showing IP version4



```
    0100 .... = Version: 4
```

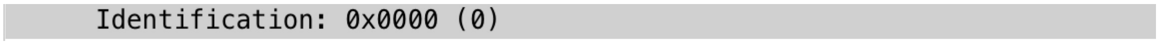
bytes 16 – 17 shows the total length is 64



```
    Total Length: 64
```

The relationship is that the IP packet maximum length is 64k bytes

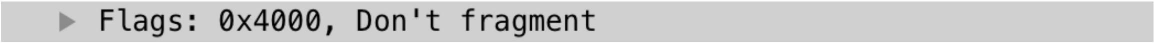
- What is encoded in bytes 18-19?



```
      Identification: 0x0000 (0)
```

because the data might be split up into many packets, the identification is to maintain the order of it. The 0 here means it is the first packet.

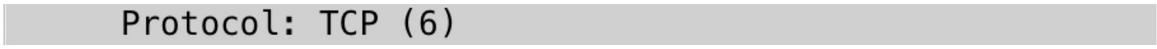
- What is encoded in bytes 20-21?



```
    ▶ Flags: 0x4000, Don't fragment
```

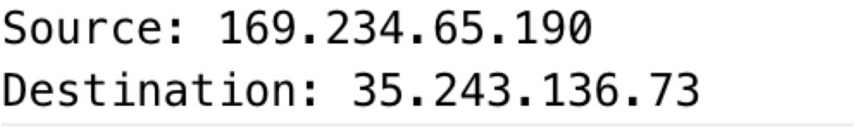
means this packet is not generated from fragment

- What is encoded in byte 23?



```
      Protocol: TCP (6)
```

- What is encoded in bytes 26-29 and 30-33?

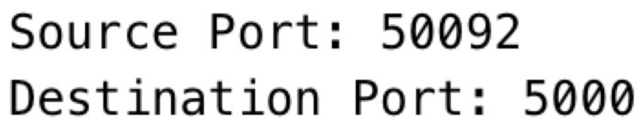


```
Source: 169.234.65.190
Destination: 35.243.136.73
```

26 – 29 source IP: 169.234.65.190

30 – 33 destination IP: 35.243.136.73

- What is encoded in bytes 34-35 and 36-37?



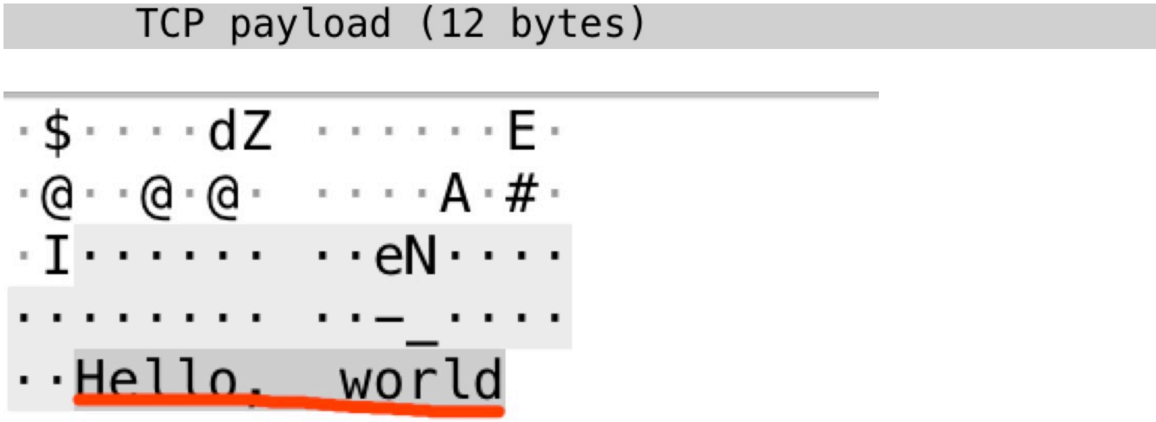
```
Source Port: 50092
Destination Port: 5000
```

34 – 35 source port 50092

36 – 37 Destination port 5000

- What is encoded after byte 65?

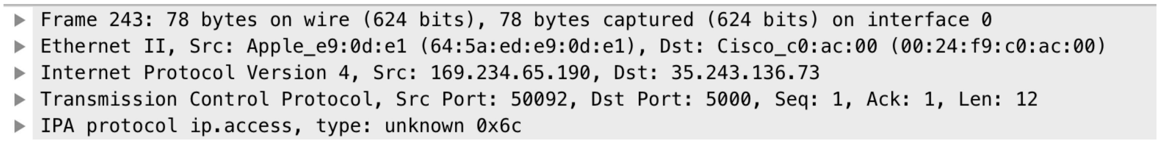
The TCP payload data



```
     TCP payload (12 bytes)
```

```
·$····dZ ······E·
·@··@·@· ····A·#·
·I······ ··eN····
········ ··__····
··Hello,  world
```

- Finally, split the raw dump of bits associating each block of bits to the layers 2 (Link), 3 (Network), 4 (Transport), and 7 (Application).



```
▶ Frame 243: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
▶ Ethernet II, Src: Apple_e9:0d:e1 (64:5a:ed:e9:0d:e1), Dst: Cisco_c0:ac:00 (00:24:f9:c0:ac:00)
▶ Internet Protocol Version 4, Src: 169.234.65.190, Dst: 35.243.136.73
▶ Transmission Control Protocol, Src Port: 50092, Dst Port: 5000, Seq: 1, Ack: 1, Len: 12
▶ IPA protocol ip.access, type: unknown 0x6c
```

78 bytes total:

0 – 13 link layer

14 – 33 network layer
34 – 65 transport layer
66 – 77 application layer