Assignment 3

Department of Computer Science,
University of Copenhagen (DIKU)
Datanet

Jonas Brunsgaard
Martin Bjerregaard Jepsen
Rasmus Wriedt Larsen

21 May, 2013

Department of Computer Science,
University of Copenhagen
Datanet

1 Theoretical part

1.1 The Internet Protocol

1.1.1 Addresses and network masks

Part 1.1: From a router's point of view the purpose of network masks is to reduce the size of the forwarding table. Because all IP addresses in the same network have the same prefix, a router only needs to store the prefix.

Part 1.2: Network masks may be expressed in the same dotted quad notation that IP addresses are expressed in (because they are also 4 bytes long). 255.225.255.0 is not a valid network mask, because it is not a continuous prefix of 1-bits. In the slash-notation we would interpret /28 as a prefix of 28 1-bits, or the network mask 255.255.255.240.

Part 1.3: The difference is that the network prefix was constrained to be of length 8, 16 or 24 bits in classful addressing, and can be of any length in classless interdomain routing.

Part 1.4: The network address is simply the IP anded with the network mask (byte form), and the broadcast address is the network address plus $2^{32-\text{prefixlength}} - 1$. The size of the network is $2^{\text{prefixlength}} - 2$ (subtracting 2 for the network and broadcast address). The available addresses are found in the interval between the network and broadcast address.

Part 2.1: Size of the network: 30
Network address: 130.225.165.0
Network mask: 255.255.255.224
Broadcast address: 130.225.156.31
First, fifth, last address: 130.225.165.1, 130.225.165.5, 130.225.165.30

Part 2.2: Size of the network: 510
Network address: 10.0.42.0
Network mask: 255.255.254.0
Broadcast address: 10.0.43.255
First, fifth, last address: 10.0.42.1, 10.0.42.5, 10.0.43.254

Part 2.3: Size of the network: 1
Network address: 4.2.2.1
Network mask: 255.255.255.255
Broadcast address: 4.2.2.1
First, fifth, last address: 4.2.2.1

Part 2.4: Size of the network: 16382
Network address: 192.38.64.0
Network mask: 255.255.192.0
Broadcast address: 192.38.127.255
First, fifth, last address: 192.38.64.1, 192.38.64.5, 192.38.127.254

1.1.2 Network Address Translation

Part 1: NAT-enabled routers handle multiple connections by maintaining a NAT translation table that maps pairs of internal IPs and ports to external ports. Each connection is given an external port in the router, such that traffic going back to the router can be de-multiplexed and sent to the appropriate internal IP and port.

Part 2: Because ports are limited to 16 bits, there are only 65536 possible ports to choose from. If 65536 long lived connections are opened by the internal hosts, no further mappings can be made in the NAT translation table. The routers ports are exhausted, and no further connections can be made.

Part 3: NAT breaks the layering principle of the Internet because it alters the contents of the transport layer port field. This is not allowed in the layering model, because a router only handles the layers up to the network layer.

1.2 Distributed Hash Tables

Part 1: The routing table for node 1 is as follows

i	$id + 2^i$	$succ$
0	2	3
1	3	3
2	5	6

Part 2: A query for item 5 (from node 1) will be received by node 6 directly.

Part 3: A query for item 5 (from node 7) will be received by node 3 and 6.

1.3 SSH Tunneling

1.3.1 ECHOSERVER ↔ CLIENT

Question 1: At this point the echo server is waiting for clients to connect to it. It is blocking on the accept call at line 24.

Question 2: The port number appears in both established sockets because it is in the source of the client's socket, and the destination of the server's socket (TCP if full duplex, so there is a connection in both directions). The port number was chosen randomly as the port for the client's outgoing connection.

Question 3: One can see all messages sent to and from the server in the TCP dump. This is because they are sent unencrypted over the network (so every hop on the way will also be able to read them). See subsection 3.1 for TCP dump.

Question 4: No it's not possible to establish a connection. The connection is blocked by a firewall on ask.diku.dk

Question 5: host = 'localhost', port = 6790. We can communicate this way, because the data sent to port 6790 on our localhost is forwarded to ask.diku.dk on port 6002 though our SSH tunnel.

Question 6: One can again see the messages sent to and from the server in the TCP dump. The data is not encrypted when sent between SSH tunnel and the echoclient, only when going though the SSH tunnel. See subsection 3.2 for TCP dump.

Question 7: One can only see scrambled data in the TCP dump. This is because the SSH traffic is encrypted. SSH runs on port 22. See subsection 3.3 for TCP dump.

Question 8: See Figure 1.

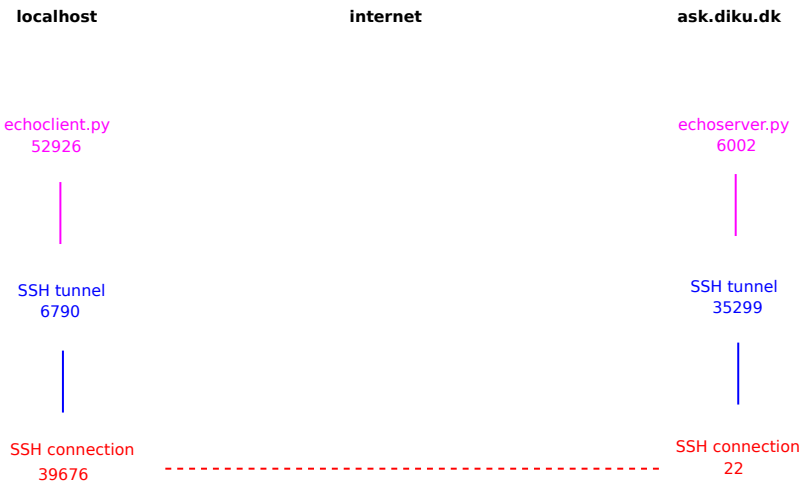

Figure 1: Overview of the connections. Dashes lines represents encrypted messages

2 Programming part

2.1 Question 1

There will be no problem in making a peer-to-peer connection between peer *A* and peer *B*.

The communication will go like this.

```
peer A (alice) <--> ns: HELLO alice alice_port
ns  <--> peer A (alice): 100 CONNECTED

peer A (alice) <--> ns: LOOKUP bob
ns  <--> peer A (alice): 200 INFO bob_ip bob_port

peer A (alice) <--> peer B (bob): HELLO alice
```

```
peer B (bob) <--> peer A (alice): 100 CONNECTED

peer A (alice) <--> peer B (bob): MSG Are you single
```

2.2 Question 2

It is not prudent to keep unused connections open, it just takes up system resources. On the other hand connections should be kept open to connections that are used frequently (because the whole connection and disconnection process add a considerable overhead to short transmissions.

We suggest implementing a peer timeout, such that connections only idle for a certain amount of time before they are automatically closed. This way only active connections takes up resources in the long run.

2.3 Question 3

We suggest a design where a peer is working as group master, and the name server only knows which peer is authorized to act as group master.

Other peers would be able to lookup and list the groups on the name server and thereafter connect to the groups peer master to get the member list.

The peer working as group master, would then send updates regarding member changes to all of its group members(peers joining and leaving the group).

All peers - which are members of the group - will then have an updated list of group members and could use a group broadcasting method to send messages to other group members.

On the name server we would need the following commands:

```
ADDGROUP <groupname>
DELGROUP <groupname>
LISTGROUPS <groupname>
LOOKUPGROUP <groupname>
```

On the peer side we would need these commands:

```
ADD_ME_TO_GROUP <nick> <port>
DEL_ME_FROM_GROUP
ADD_GROUP_MEMBER <nick> <port>
DEL_GROUP_MEMBER <nick>
MSG_GROUP <group> <msg>
```

2.4 Question 4

We can achieve $O(lg(n))$ with flooding. Figure 2, shows how $u0$ is broadcasting to the rest of the network consisting of the peers $u1, u2, u3, u4, u5, u6$.

$u0$ takes the broadcast-list and splits it in half, then $u0$ sends the message and one of the new half broadcasting lists (the receiver is subtracted) to the first member of the lists.

This implies that $u1$ will get the message and the broadcast list $u2, u3$.

$u4$ will also get the message, but will get the other half broadcasting list, containing $u5, u6$.

$u1$ again splits the remaining list in two and because each list now only contains one peer, only the message is send to $u2$ and $u3$. The exact same procedure is followed by $u4$. As mentioned above a graphical illustration is shown in Figure 2.

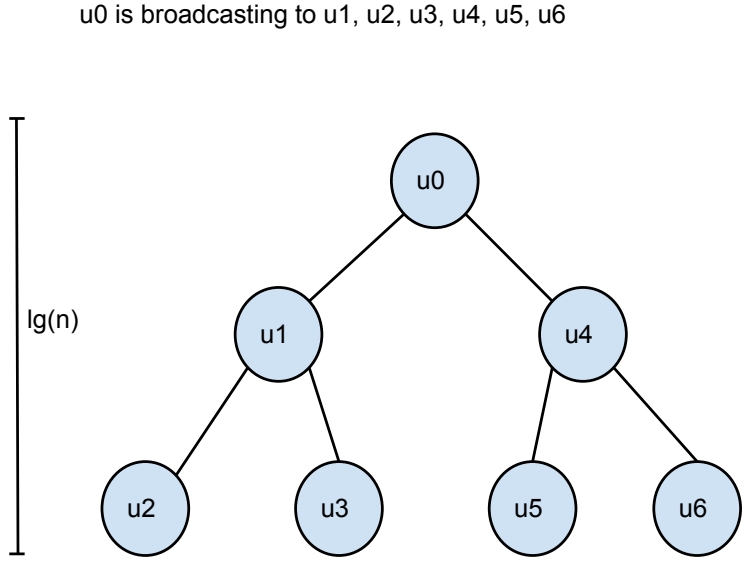

Figure 2: Flooding messages

2.5 Question 5

In regard to the name server we need the confirmation to assure that the name server indeed removed the peer from the userlist. Even though all packages are transmitted, the TCP protocol will not help us in case of an internal server error etc. In that regard a missing BYE from the server tells us, that something unexpected happened. The same is true for the MSG ACK, this also assure us that the recipient not only received the message, but also that it was in fact delivered to the receivers output device.

3 Appendix

3.1 Question 3 TCP dump

```
# after having established connection, message sent: hejhejhej

$ netstat -an | egrep '6789|Address'
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:6789          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:6789          127.0.0.1:33000
    ESTABLISHED
tcp        0      0 127.0.0.1:33000         127.0.0.1:6789
    ESTABLISHED

$ sudo tcpdump -Xs 1514 port 6789 -i lo
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 1514 bytes
13:55:53.351681 IP localhost.33000 > localhost.6789: Flags [P.], seq
    3618640162:3618640171, ack 2713362031, win 257, options [nop,nop,TS val
    2779685 ecr 2745738], length 9
    0x0000:  4500 003d f4ad 4000 4006 480b 7f00 0001  E..=..@.@.H.....
    0x0010:  7f00 0001 80e8 1a85 d7b0 1122 a1ba 9e6f  ............"...o
    0x0020:  8018 0101 fe31 0000 0101 080a 002a 6a25  .....1.......*j%
    0x0030:  0029 e58a 6865 6a68 656a 6865 6a          .)..hejhejhej
13:55:53.351733 IP localhost.6789 > localhost.33000: Flags [.], ack 9, win
    256, options [nop,nop,TS val 2779685 ecr 2779685], length 0
    0x0000:  4500 0034 7be5 4000 4006 c0dc 7f00 0001  E..4{.@.@.......
    0x0010:  7f00 0001 1a85 80e8 a1ba 9e6f d7b0 112b  ...........o...+
    0x0020:  8010 0100 fe28 0000 0101 080a 002a 6a25  .....(.......*j%
    0x0030:  002a 6a25                                .*j%
13:55:53.351817 IP localhost.6789 > localhost.33000: Flags [P.], seq 1:22,
    ack 9, win 256, options [nop,nop,TS val 2779685 ecr 2779685], length 21
    0x0000:  4500 0049 7be6 4000 4006 c0c6 7f00 0001  E..I{.@.@.......
    0x0010:  7f00 0001 1a85 80e8 a1ba 9e6f d7b0 112b  ...........o...+
    0x0020:  8018 0100 fe3d 0000 0101 080a 002a 6a25  .....=.......*j%
    0x0030:  002a 6a25 6672 6f6d 5f73 6572 7665 723a  .*j%from_server:
    0x0040:  6865 6a68 656a 6865 6a                   hejhejhej
13:55:53.351891 IP localhost.33000 > localhost.6789: Flags [.], ack 22, win
    257, options [nop,nop,TS val 2779685 ecr 2779685], length 0
    0x0000:  4500 0034 f4ae 4000 4006 4813 7f00 0001  E..4..@.@.H.....
    0x0010:  7f00 0001 80e8 1a85 d7b0 112b a1ba 9e84  ...........+....
    0x0020:  8010 0101 fe28 0000 0101 080a 002a 6a25  .....(.......*j%
    0x0030:  002a 6a25                                .*j%
```

3.2 Question 6 TCP dump

```
# after having established connection, message sent: hejhejhej

$ netstat -an | egrep '6790|Address'
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:6790          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:52926         127.0.0.1:6790
    ESTABLISHED
tcp        0      0 127.0.0.1:6790          127.0.0.1:52926
    ESTABLISHED
tcp6       0      0 ::1:6790                ::::*                   LISTEN

$ sudo tcpdump -Xs 1514 port 6790 -i lo
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 1514 bytes
21:57:40.447997 IP localhost.52926 > localhost.6790: Flags [P.], seq
    2905874139:2905874148, ack 4207445878, win 257, options [nop,nop,TS val
    862862 ecr 860254], length 9
    0x0000:  4500 003d 00c8 4000 4006 3bf1 7f00 0001  E..=..@.@.;.....
    0x0010:  7f00 0001 cebe 1a86 ad34 1edb fac8 8776  .........4.....v
    0x0020:  8018 0101 fe31 0000 0101 080a 000d 2a8e  .....1........*.
```

```
    0x0030:  000d 205e 6865 6a68 656a 6865 6a         ...^hejhejhej
21:57:40.468936 IP localhost.6790 > localhost.52926: Flags [P.], seq 1:22,
    ack 9, win 256, options [nop,nop,TS val 862868 ecr 862862], length 21
    0x0000:  4500 0049 9976 4000 4006 a336 7f00 0001  E..I.v@.@..6....
    0x0010:  7f00 0001 1a86 cebe fac8 8776 ad34 1ee4  ...........v.4..
    0x0020:  8018 0100 fe3d 0000 0101 080a 000d 2a94  .....=........*.
    0x0030:  000d 2a8e 6672 6f6d 5f73 6572 7665 723a  ..*.from_server:
    0x0040:  6865 6a68 656a 6865 6a                   hejhejhej
21:57:40.468973 IP localhost.52926 > localhost.6790: Flags [.], ack 22, win
    257, options [nop,nop,TS val 862868 ecr 862868], length 0
    0x0000:  4500 0034 00c9 4000 4006 3bf9 7f00 0001  E..4..@.@.;.....
    0x0010:  7f00 0001 cebe 1a86 ad34 1ee4 fac8 878b  .........4......
    0x0020:  8010 0101 fe28 0000 0101 080a 000d 2a94  .....(........*.
    0x0030:  000d 2a94                                ..*.
```

3.3 Question 7 TCP dump

```
# after having established connection, message sent: hejhejhej

# on local machine
$ netstat -an | egrep '130.225.96.225|Address'
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.0.11:39676      130.225.96.225:22
    ESTABLISHED

$ sudo tcpdump -Xs 1514 port 39676 -i wlan0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 1514 bytes
22:06:50.004458 IP RasmusWL-S430.local.39676 > ask.diku.dk.ssh: Flags [P.],
    seq 3066705582:3066705630, ack 1313069127, win 324, options [nop,nop,TS
    val 1000252 ecr 674476924], length 48
    0x0000:  4510 0064 1514 4000 4006 80fa c0a8 000b  E..d..@.@.......
    0x0010:  82e1 60e1 9afc 0016 b6ca 36ae 4e43 d847  ..`.......6.NC.G
    0x0020:  8018 0144 aeb3 0000 0101 080a 000f 433c  ...D..........C<
    0x0030:  2833 b37c 09a9 b044 ad14 e784 ae7b 59b7  (3.|...D.....{Y.
    0x0040:  9cce 0a7a 7b74 a51b f122 5519 a372 8803  ...z{t..."U..r..
    0x0050:  3520 d13b dc23 5a31 4fb8 66f3 acc9 48f1  5..;.#Z1O.f...H.
    0x0060:  f631 e476                                .1.v
22:06:50.072481 IP ask.diku.dk.ssh > RasmusWL-S430.local.39676: Flags [P.],
    seq 1:129, ack 48, win 125, options [nop,nop,TS val 674509440 ecr
    1000252], length 128
    0x0000:  4500 00b4 e3d6 4000 3206 bff7 82e1 60e1  E.....@.2.....`.
    0x0010:  c0a8 000b 0016 9afc 4e43 d847 b6ca 36de  ........NC.G..6.
    0x0020:  8018 007d b599 0000 0101 080a 2834 3280  ...}........(42.
    0x0030:  000f 433c df43 b086 3522 2a5d 881a 75f9  ..C<.C..5"*]..u.
    0x0040:  951a bbf3 4846 9f62 0653 b3a5 95f6 5721  ....HF.b.S....W!
    0x0050:  a9da f03f 1e73 2387 786a 6ca4 2773 5c6d  ...?.s#.xjl.'s\m
    0x0060:  384b 2086 e90a f137 49e4 82f2 75e0 9d53  8K.....7I...u..S
    0x0070:  144d 01a9 8c3a 8eec 1def 5d74 6826 1234  .M...:....]th&.4
    0x0080:  e425 6467 31fc 6def 8048 d864 644c f73f  .%dg1.m..H.ddL.?
    0x0090:  e1ab a94d 6654 d2e9 e7ae 3dc0 bd8e 93ad  ...MfT....=.....
    0x00a0:  6b08 97e8 157c dc10 36c1 d956 1348 3ef2  k....|..6..V.H>.
    0x00b0:  1cfa 36de                                ..6.
22:06:50.072527 IP RasmusWL-S430.local.39676 > ask.diku.dk.ssh: Flags [.],
    ack 129, win 323, options [nop,nop,TS val 1000269 ecr 674509440], length
     0
    0x0000:  4510 0034 1515 4000 4006 8129 c0a8 000b  E..4..@.@..)....
    0x0010:  82e1 60e1 9afc 0016 b6ca 36de 4e43 d8c7  ..`.......6.NC..
    0x0020:  8010 0143 a49c 0000 0101 080a 000f 434d  ...C..........CM
    0x0030:  2834 3280                                (42.


# on ask
$ netstat -an | egrep '2.111.93.104|Address'
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 130.225.96.225:22       2.111.93.104:39676
    ESTABLISHED

```

```
$ netstat -an | egrep 'Address|6002'
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:6002          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:6002          127.0.0.1:35299
    ESTABLISHED
tcp        0      0 127.0.0.1:35299         127.0.0.1:6002
    ESTABLISHED
```