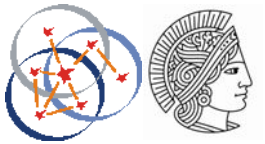
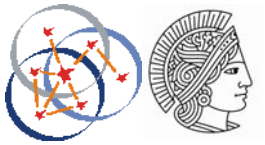


P2P Networks
Programming Exercise # 2

Dominik Fischer, P2P Networks Group,
TU Darmstadt

Date: Nov. 1th, 2011

Solving Exercise 1



```
final ByteBuffer buffer = ByteBuffer.allocate(
    m.length() + Integer.SIZE / 8
);

buffer.order(ByteOrder.BIG_ENDIAN);

buffer.putInt(message.length());
buffer.put(message.getBytes());

buffer.rewind();

DatagramChannel channel = DatagramChannel.open(
    StandardProtocolFamily.INET
);

channel.send(buffer, address);
channel.close();
```

Road Blocks

- Network Byte Order.

- `send (sock, "message", 7);`
  `int packlen = read (sock, buffer, 7);`
- `packlen == 7?`
- `False` – in case of TCP.
  - The stream can be split arbitraryly.
- `True` – in case of UDP.
  - But UDP comes with a maximum packet size instead.

Network Debugging

- Tstat
  - analyzes networks statistically.
- GNU netcat
  - sends and receives simple messages and
  - provides one of the shortest solution to exercise 1:

```
echo "message" | netcat –u localhost 1337
```

- Wireshark
  - inspects and dissects packets in detail.

Next Exercise – Multicast

- Implement a message distributing system.
- It consists of a client that
  - can send arbitrary string messages input by the user to a given address and
  - displays all string messages it receives via a given port
- and a server that
  - receives string messages on a given port and
  - redistributes them to a set of addresses.
- The server should consume the following messages and treat their senders accordingly:
  - “hello” - send a copy of all subsequently received messages to the sender of this message,
  - “bye” - stop sending anything to the sender of this message.
  - “kill” - shut down the server.