



P2P Networks – Exercise # 2

Ikram M. Khan, P2P Networks Group,
TU Darmstadt

Date: Nov. 8th, 2011



Announcements

How to submit your assignments?

- Theory assignment
 - Hand-over in the class or drop in the box nearby S1|02 A110
- Programming assignment
 - p2pws11submission@tk.informatik.tu-darmstadt.de



Exercise # 2: Overlay Networks

What is an overlay network?

- A network built on the top of another network e.g., IP

How is it different from other networks?

- Direct neighbours in the overlay network may not be direct neighbours in the underlying network.
- Therefore the “shortest path” between two nodes in the overlay may actually not be the true shortest path in the original network.



Exercise # 2 : Bootstrapping

What is a bootstrapping process?

- Example in P2P: we want to join a network, but do not know the peers
- Solution: find at least one node in the network
- Connecting to that node to retrieve a list of peers allows us to join the network

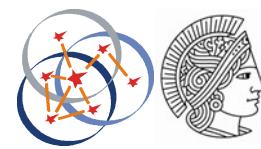
Define name, identifiers, and references.

- Names, identifiers and references are attributes of a resource
- Names describe a resource, but may not be unique
- Identifiers are unique
- References give the location of the resource



Exercise # 2: Epidemics

- How fast do we need to deal with the zombie outbreak?
 - As quick as possible, while their numbers are still low
- What are the possible solutions to deal with the outbreak?
 - Treatment, quarantine, and attacking the zombies
- Depending upon the rate of outbreak what could be the best course for human survival? Justify your answer.
 - If the outbreak is quite strong and quick, then attacking/isolating the zombies massively while trying to quickly find a cure.
 - This is subjective, your justification is decisive here ;-)
- In the case of P2P networks, what is the real lesson to be learned from this paper?
 - Highly “infectious” situations get out of control quickly, are hard to revert
 - Massive sharing of a resource → very hard to remove all the copies
 - It's better to act very early to prevent distribution



Exercise # 2: Important considerations

- Do not submit answers that are not yours
 - Result: 0 points and colloquium
- If you **based** your answer on something that is not in the lecture slides or really need to make a citation:
 - Refer your sources! (where sources ≠ your classmates)
- Max of 2 people for written assignment
- Bullet points are OK as long as they are not ambiguous:
 - If I am not able to understand your idea, I prefer a full sentence.



Exercise # 2: Theory Exercise (11 P.)

- Overlay networks (2 P.)
 - What is an overlay network?
 - How is it different from other networks?
- Bootstrapping (3 P.)
 - What is a bootstrapping process?
 - Define and explain:
 - Name
 - Identifiers
 - References



Exercise # 2: Theory Exercise (11 P.)

- Epidemics (6 P.)
 - How fast do we need to deal with the zombie outbreak?
 - What are the possible solutions to deal with the outbreak?
 - Depending upon the rate of outbreak what could be the best course for human survival? Justify your answer.
 - In the case of P2P networks, what is the real lesson to be learned from this paper?



Exercise # 2: Programming Exercise (8 P.)

- Implement a UDP based chat service for distributing arbitrary string (ASCII) messages among participants. Your chat service should do the following:
 - The client should be able to send and receives messages to and from the server, respectively.
 - The server should redistribute all messages it receives on port 1337 to all clients it knows.
 - A client could make itself known to the server by sending it the string message "HELLO".
 - When the server receives the message "BYE" it shouldn't send any further messages to that client.
 - When the server receives the message "KILL" it should shutdown itself.



Exercise # 2 - Hints For Those Using C

- Server:
 - Keep a list of `struct sockaddr`.
- Client:
 - To send and receive simultaneously
 - Use `select()` over `stdin` and the socket.

Exercise # 2 - Hints For Those Using Java



- Server:

```
for (SocketAddress destination : destinations) {  
    buffer.rewind();  
    channel.send(buffer, destination);  
}
```

- Client:

- Send and receive simultaneously using threads.
- Java is unable to *select()* on *System.in*.



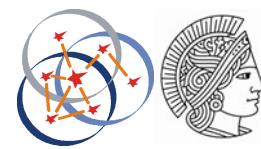
Exercise # 2: Hints For Those Using C

- *gethostbyname* is outdated.
 - *getaddrinfo* does not depend on IPv4.
 - *struct sockaddr_storage* keeps all addresses.
- *sizeof()* is not *strlen()*!



Exercise # 2: Hints For Those Using Java

- Use `java.nio`.
- Buffer
 - Abstraction for blocks of data.
 - Use them to compose and dissect packets.
 - Handles byte order, data types, character encoding...
- Channels
 - More efficient than streams,
 - More abstract.



Exercise # 2: Hints For Those Using Python

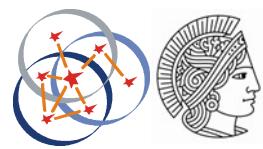
- *socketserver*
 - Ready-to-use framework.
 - Forking and threading variants through Mixins.
- Various server and client classes
 - Using the *socketserver* base classes.
 - Best practice examples for protocol handling.

python -m http.server



Exercise # 3 - File Transfer

- Write a file transfer client and server application that copies files of arbitrary size via UDP
- The application shall be able to copy files of multiple Gigabytes size
 - Hint: you could divide large files into chunks and transmit them using UDP packets (datagrams)
- If a file, or part of it, is lost the transfer shall be repeated automatically
 - Hint: use mechanism(s) to handle packet loss(es)
- Please document your assignment
 - Document packet formats used by your application
 - Include sample tests
 - Write sample scripts to run your code



Thanks and Good Luck!