Lab Class
Protocol-Design

P2P-Overlay, Part II
**P2P-Protokol, Version 0.1**

- Extending the P2P Node Software by Adding 'Standard' Features:
  - Peer Search / Overlay Mapping
  - Information Lookup
  - (Information Transfer)
P2P-Protokol, Version 0.1

- New message concepts:
  - Originator spec.: FROM
  - Destination spec.: FOR
  - Message-IDs: MESSAGE-ID
  - Message reach: TTL
  - Matching of replies with responses: KEY

- Order of fields is predefined!
  (see Assignment 5)
**P2P-Protokol, Version 0.1**

- Peer Search (1):
  - Node only knows neighbours
  - Want to know more about other nodes (peers)
  - Send information request (ping) into network
  - Collect responses (pongs)
  - But: might result in huge number of replies!
**P2P-Protokol, Version 0.1**

- Peer Search (2):
  - New message type: 
    
    `PING FROM viper:2000 MESSAGE-ID 1 TTL 3 P2P/0.1`
  - Contains originator spec: `FROM VIPER:2000`
    - needed to send replies
    - used for duplicate message detection
  - Message ID: `MESSAGE-ID 1`
    - used for duplicate message detection
    - (Node-ID, Message ID) globally unique!
**P2P-Protokol, Version 0.1**

- Peer Search (3):
  
  PING FROM viper:2000 MESSAGE-ID 1 TTL 3 P2P/0.1

- Time-to-Live Counter: TTL 3
  
  - limits message reach...
  - ... and thus overlay network load
Peer Search (5):

Reply message:

```
P2P/0.1 PONG FOR viper:2000 FROM boa:2000 MESSAGE-ID 24 TTL 3
```

☐ Contains destination node ID:

```
FOR VIPER:2000
```

☐ Contains new message ID!!

☐ Reply message with both originator and destination:

```
first destination spec (FOR),
then originator spec (FROM)!
```
Information Lookup (1):

- 'Information' usually means file names :-)
- Can have multiple simultaneous lookup requests in progress:
  - what reply belongs to what request?
- Again potentially many replies.
P2P-Protokol, Version 0.1

- Information Lookup (2)
  - New message type:
    
    SEARCH FROM viper:2000 KEY readme.txt MESSAGE-ID 2 TTL 3 P2P/0.1

  - Contains sender spec, message ID, TTL
  - Contains search term spec: KEY readme.txt
P2P-Protokol, Version 0.1

- Information Lookup (3)
  - New reply message type:
    ```plaintext
    P2P/0.1 FOUND FOR viper:2000 FROM boa:2000 MESSAGE ID 10
    TTL 3 KEY readme.txt
    ```
  - Reply message, so first destination (FOR), then originator (FROM)
  - New message ID!
  - Contains search term (KEY) to enable matching of requests to replies.
P2P-Protokol, Version 0.1

- Downloading Information (1):
  - Need to know where to find information (SEARCH/FOUND)
  - Request with explicit destination!
P2P-Protokol, Version 0.1

- Downloading Information (2):
  - New message type:
    GET FROM viper:2000 FOR boa:2000 KEY readme.txt MESSAGE-ID 101 TTL 3 P2P/0.1
  - Request message, so
    - first originator (FROM)
    - then destination (FOR)
  - Note: opposite order of orig. and dest.!
P2P-Protokol, Version 0.1

- Downloading Information (3):
  - Protocol not yet powerful enough for information transfer
  - Respond with error message:
    P2P/0.1 510 NOT IMPLEMENTED FOR viper:2000 FROM boa:2000 MESSAGE-ID 13 TTL 3 KEY readme.txt
P2P-Protokol, Version 0.1

- Uploading Information (1):
  - Works like downloading:
    PUT FROM viper:2000 FOR boa:2000 KEY readme.txt
    SIZE 1024 MESSAGE-ID 19 TTL 3
  - New field to warn receiver about size:
    SIZE 1024
  - For now, reply with same error message as for downloading (510 NOT IMPLEMENTED)
**P2P-Protokol, Version 0.1**

- Newly generated (non-handshake) messages:
  - request: broadcast
  - reply: send only to neighbour from which the request has been received

- Forwarding:
  - ignore duplicate messages
  - flood with regard to TTL
**P2P-Protokol, Version 0.1**

*Duplicate Message Detection*

- With flooding, message may arrive multiple times
  - React to message once only!
  - Need to recognize duplicates
  - Use tuple (Node-ID, Message-ID)
  - For each originator node:
    - store message IDs already seen as ordered list
    - check new messages against list
    - generate Message IDs using counter
P2P-Protokol, Version 0.1
TTL Handling
**P2P-Protokol, Version 0.1**

*TTL Handling*

- Node generates message
  1. send new message: ...TTL 3...
  2. receive, decrement (TTL now 2)
  3. process message
  4. check TTL > 0
     i. Yes: flood as ...TTL 2...
     ii. No: discard

- Decrement TTL *after* receiving, *before* checking!

- Forwarding changes Message!!!
Additional Commands

- Reading short commands from keyboard:
  - ping
  - search legal.mp3
  - get legal.mp3 viper:2100
  - put legal.mp3 boa:2200