TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Intelligente Netze und Management verteilter Systeme
Prof. Anja Feldmann, Ph.D.
Gregor Maier, Jörg Wallerich, Andi Wundsam

6th Work Sheet Praktikum Protokolldesign WS 07/08

In this assignment we will improve our node software from the previous assignment by adding a peer search mechanism, search requests and the basic foundations for the ability to transfer files over the P2P overlay. Below you will find a more precise specification of how to implement the additional functionality.

Processing Protocol Messages: For this task, messages are to be forwarded by simple flooding, no matter if they are addressed to a specific node or not. Whenever messages are newly generated, e.g., replies for requests, such messages have to be sent to *all* neighbours (broadcasting). If a received message is forwarded, then it is sent to all neighbours except the one that message was received from (flooding). When flooding is used to forward messages, it will happen that messages arrive multiple times at the same node over different paths. In order to maintain the stability of the system, a node must react only once to a message, even if the message is received multiple times. To achieve this behaviour the node has to remember which messages it has already seen. Therefore we use message IDs that, together with the node ID of the originator, build a globally unique identifier for messages. Message IDs will be forwarded without change, but newly created messages always get a new message ID.
There are several ways how to generate such message IDs and to provide a mechanism to check if a message is new or not. One of the most simple ones is a per-node message counter to generate message IDs. These message IDs can be checked by managing ranges of already seen message IDs, e.g., $1-11, 13-14$ (we have seen all messages from ID 1 to 14, *except* the message with ID 12).
In order to avoid overflowing the overlay network with messages, we will limit the maximum distance that messages are allow to travel. This is accomplished by using a time-to-live (`TTL`) field in the messages. This field gives the remaining distance that the message is still allowed to travel. The distance is measured in hops (the path from a node to each one of its neighbours is one hop). The TTL is initialized with 3, allowing it to reach all nodes up to a distance of 3 hops.
Whenever a message arrives, its TTL counter is to be decreased by on. After the message is processed (e.g., sending a `PONG` reply for a `PING` request), the TTL counter is checked and as long as the TTL is greater than 0, the message can be forwarded to the nodes neighbours with the changed TTL value. If the TTL is equal to 0, the message is not forwarded. Note that this scheme changes messages while forwarding!

Features to implement: Extend the node software by implementing the following features and protocol messages:

1. *Peer Search*: Pear search is done by sending `PING` messages and collecting `PONG` replies (see protocol specifivation on worksheet 5). When a pear search is initiated, the node has to generate a `PING` message and send it to all its neighbours. A node that receives a `PING` message first generates a `PONG` messages and sends it only to the neighbour from which the `PING` message has been received (no flooding at this point!). Then the node must flood the `PING` message with regard to the TTL counter of the message. Note that `PONG` messages have an explicit destination node ID, the ID of the node from that the `PING` message originated. `PONG` messages have to be flooded unless it has arrived at its destination.

   The node that started the peer search has to display the sender node ID and the TTL value of all incoming `PONG` replies to its request.

2. *Search Requests*: A search for a file is handled by the `SEARCH` and `FOUND` messages. Search requests work similar to `PING` and `PONG` except that `FOUND` messages are only generated if the queried file is available in a special download directory. Otherwise, the request is forwarded only.

The node that initiates a search has to keep information about all nodes that have replied with a `FOUND` message. The request contains the file name in the `KEY` field, so that it is possible to to match search requests with replies. This way it is possible to know to what node a download request message for a given file can be sent.

3. *File Transfer*: When a node wants to download a file, it sends a `GET` message into the overlay network using a node as destination node, that is known to have the requested file ready for download.

   Because our protocol ist not yet powerful enough to support actual file transfers, `GET` messages have to be answered with an error message of type `510 NOT IMPLEMENTED`.

   Uploading files works very similar to downloading, except that the request message for uploading is of type `PUT`. The node for now has to answer `PUT` messages with the same error as for `GET` messages.

Protocol messages that are used to implement features like peer search or file transfers are *not* allowed before the `HELLO` handshake between two neighbours has finished. Only *after* the handshake has finished successfully, it is allowed to transfer other messages. On the other hand, `HELLO` messages are not allowed after the handshake. If an illegal message is received, we have a protocol violation. Protocol violations are always reacted to with an appropriate error message and the termination of the connection.

User Interface: In order to initiate peer search or file transfers, the node software has to be able to read short commande from the keyboard and act accordingly. Such commands ***might*** be:

ping tells the node to start a peer search by sending a *PING* into the overlay network.

search <**filename**> tells the node to send a file search request.

get <**filename**> tells the node to download a file.

put <**filename**> <**node-id**> tells the node to upload a file onto a given node.

Question 1: (100 points) **P2P System, Part 2**

Extend the node software by adding peer detection, information search and data transfer. This includes:

(a) (10 points) Command input

(b) (10 points) New message types and reaction to protocol violations

(c) (40 points) Recognition of already received messages

(d) (40 points) Forwarding of messages by flooding with regard to the TTL

Deliverables:

- Your program/script (both source code and binary if written in C, C++ or Java)
- A short description of how to use the software, and what things are not quite working yet.

For details concerning the submission of your solutions see FAQ
(`http://www.net.t-labs.tu-berlin.de/teaching/ws0708/PD_labcourse/`).
Due: Tuesday, December 4., 2007, 11:59h s.t.