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

7th Work Sheet Praktikum Protokolldesign WS 07/08

Question 1: (50 points) Implementation of a Simple Forwarding Table

As flooding is a very inefficient way for forwarding messages, we add a simple mechanism for optimizing message forwarding. This mechanism consists of a simple lookup table. The entries of this table a generated using received messages and allow us to forward messages for known destinations in the 'correct' direction. The table is called a *forwarding table.*

The forwarding table has to constructed and used in the following way (learning phase):

- Extract the node ID of the originator (`FROM`) and the TTLs of all (non-handshake) messages. Then insert a new entry or, if the TTL hints at a better (shorter) path, update an existing entry in the forwarding table.

  The node ID of the originator is to be used as key to access the appropriate entry in the table. Thus, a table entry looks like this:

  Timestamp: when has this entry been created or updated last

  Direction: the neighbour (neighbour connection) the message was received from

  TTL: the TTL of the message used when adding/changing this entry. The TTL is a measure for the distance to the node, as it is always initialized with 3.

  Using this mechanism we get a table with the following property: For each node from which we have received a message, the table contains the neighbour that the originitaor used to reach us. This also means that we can reach the originator via the same neighbour!
- If a neighbour connection dies, all table entries that refer to this neighbour have to be removed. This might lead to nodes not reachable anymore through the forwarding table.
- Table entries have to be managed in a way, that they always contain the most up-to-date information about the shortest paths to other nodes. Paths of higher length must be ignored. Paths with the same length are more up-to-date and have to be used to update a table entry.

Using this forwarding table, it is now possible to optimize the forwarding of messages:

- Table entries older than 120 seconds must be ignored and removed.
- Messages with a specific destination (`FOR`) for which we have a current table entry are no longer flooded. Instead we forward the message only over the corresponding neighbour connection found in the forwarding table.
- If there is no matching entry for the destination node in the forwarding table, the message has to be flooded as before.
- No matter if a messages is forwarded by flooding or using the forwarding table, the TTL of the message has to be decremented and no forwarding is allowed if the decremented TTL is 0.
- As we will add a more intricate way to learn paths in one of the later assignments, the table logic has to be implemented in a way, that it is possible to manage (add, update, remove, lookup) entries using dedicated function calls.

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.

Question 2: (20 points) Automated Session Establishment

For this task, the nodes learned during the session establishment phase (HELLO handshake) of the protocol are to be used to enable the node software to automatically setup sessions to new neighbours. Therefore the following mechanisms have to be implemented:

- Store all peer node IDs learned via the `NEIGHBOURS` part of the `HELLO` handshake during the session initialization in a queue (FIFO).
- As long as there are less than the maximum number of 4 *actively established* sessions, new sessions have to be established. The peer nodes to connect are taken from the queue, i.e., removed from the queue and then used.
- If a session is closed or dies, the node ID of the neighbour has to be re-added to the queue. In addition, a new session has to be established, as long as there are less than 4 *actively established* sessions.
- If the attempt to connect a node fails, the node-ID must be ignored and is removed from the queue. Then the software has to attempt to connect the next node from the queue.

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.

Question 3: (30 points) Protocol Version 0.2 – Parameterized Messages

The protcol, as it is in version 0.1, is very limited in its abilities. Therefore we will extend our protocol in a way that it is possible to parameterize the messages we have so far, and also to add payload data to messages. The resulting protocol has the version number 0.2. In more detail, you have to implement the following changes:

- The version number is 0.2 now!
- A message consists of a header and a body.
- The header looks like this:
  - The first line of the header is the same as for version 0.1 but for the different version number.
  - Then there can follow an arbitrary number of additional non-empty lines that consist of a parameter name, a colon, and a parameter value:
    `<parameter> ':' <value> \r\n`
  - One of the parameter lines has to be a length specification for the body in terms of bytes
    `Content-Length: <n> \r\n`
    A message with an empty body has to specify a length of zero bytes:
    `Content-Length: 0`
  - The header ends with an empty line (which means `\r\n`)
- After the header a (possibly empty) body follows.

The following header parameters have to be added to all (non-handshake) messages and also have to be extracted upon receiving a message.

- `Application: <string>`
- `Content-Length: <body size in bytes>`

For this assignment, the `Application:` value is always `none`.

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 11., 2007, 11:59h s.t.