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

7. Blatt Praktikum Protokolldesign WS 07/08

Aufgabe 1: (50 Punkte) Implementation einer einfachen Forwarding Tabelle

Da Flooding eine sehr ineffiziente Methode zum Weiterleiten von Nachrichten ist soll in dieser Aufgabe ein einfacher Mechanismus zur Optimierung der Nachrichten-Weiterleitung implementiert werden. Dieser Mechanismus besteht aus einer einfachen Tabelle, deren Einträge aus empfangenen Nachrichten generiert werden und es erlauben, Nachrichten an bekannte Ziele direkt in die korrekte 'Richtung' weiterzuleiten. Diese Tabelle wird auch *Forwarding-Table* genannt.

Die Tabelle soll folgendermaßen aufgebaut werden (Lern-Phase):

- Aus jeder empfangenen Nachricht sind die Knoten-ID des ursprünglichen Absenders (`FROM`) sowie die TTL zu extrahieren und ein entsprechender Eintrag in der Tabelle zu erstellen bzw. zu aktualisieren falls die TTL einen besseren (kürzeren) Weg anzeigt.

  Die Knoten-ID des Absenders soll als Zugriffs-Schlüssel in die Tabelle genutzt werden. Ein Tabelleneintrag sieht dann folgendermassen aus:

  Timestamp: wann wurde dieser Eintrag zuletzt aktualisiert

  Direction: zu belegen mit der Verbindung oder dem Nachbarn, über den die Nachricht empfangen wurde.

  TTL: zu belegen mit der TTL der Nachricht. Dies ist ein Mass für die Entfernung zum Knoten, da die TTL immer mit 3 initialisiert wird.

  Diese Tabelle enthält also für jeden Knoten, von dem man eine Nachricht empfangen hat, die Nachbarverbindung, über die man für den Sender der Nachricht erreichbar ist und somit auch, wie man selbst den Sender erreichen kann.

- Sollte eine Nachbarverbindung wegfallen, so sind alle Einträge die diese Verbindung referenzieren, zu löschen! Dies kann dazu führen, dass kein Eintrag zu einem Ziel mehr existiert.

Mit Hilfe dieser Forwarding-Tabelle kann nun das Weiterleiten von Nachrichten optimiert werden. Dabei ist folgendermassen vorzugehen:

- Tabelleneinträge die älter als 120 Sekunden sind, sollen ignoriert und gelöscht werden.
- Nachrichten für die ein Empfänger (`FOR`) angegeben ist und für die ein aktueller Tabellen-Eintrag existiert sollen nicht mehr geflooded sondern nur noch über die Verbindung, die in der Tabelle angegeben ist, weitergeleitet bzw. versendet werden.
- Wenn in der Forwarding-Table kein aktueller Eintrag für ein gegebenes Ziel zu finden ist, dann soll die Nachricht wie vorher geflooded werden.
- Unabhängig davon, ob eine Nachricht durch Flooding oder unter Benutzung der Forwarding-Tabelle weitergeleitet wird, ist die TTL der Nachricht zu erniedrigen und die Nachricht wird nicht weitergeleitet, wenn die TTL 0 ist.
- Die Tabellen-Logik sollte so gestaltet werden, daß das Pflegen (Einfügen, Ersetzen, Löschen) über Funktionsaufrufe möglich ist, da der Mechanismus zum Erlernen von Wegen in einem der nächsten Blätter noch weiter verfeinert werden wird.

Abzugeben sind:

- Dein Programm (sowohl Quelltext als auch ggf. compiliert im Falle von C, C++ und Java)

- Ggf. eine Beschreibung, was funktionieren sollte, dies leider aber immer noch nicht tut...

Aufgabe 2: (20 Punkte) Automatisierter Verbindungsaufbau

Für diese Aufgabe sollen die in der Protokoll-Initialisierungsphase (HELLO handshake) erlernten Nachbarknoten benutzt werden, um die Software vollautomatisch Peer-Verbindungen aufbauen zu lassen. Dazu soll folgendes implementiert werden:

- Merken aller wärend der Handshake-Phase durch die `NEIGHBOURS`-Liste erlernten P2P-Knoten in einer Queue (FIFO).
- Solange weniger als die maximal 4 erlaubten *aktiv aufgebauten* Verbindungen bestehen, sollen neue Verbindungen aufgebaut werden. Die Knoten, zu dem die Verbindungen aufgebaut werden sollen, sind aus der Queue zu entnehmen, d.h. aus der Queue zu streichen und dann zu benutzen.
- Wenn eine Verbindung stirbt, so soll der entsprechende Nachbarknoten wieder in die Queue eingefügt werden. Außerdem soll der nächste Knoten aus der Queue für einen neuen Verbindungsaufbau genutzt werden falls nun weniger als 4 *aktiv aufgebaute* Verbindungen bestehen.
- Wenn der Verbindungsaufbau zu einem Knoten scheitert, soll die Knoten-ID verworfen werden, d.h. *nicht* wieder in die Queue eingefügt werden. Stattdessen soll versucht werden, eine Verbindung zum nächsten Knoten aus der Queue aufzubauen.

Abzugeben sind:

- Dein Programm (sowohl Quelltext als auch ggf. compiliert im Falle von C, C++ und Java)
- Ggf. eine Beschreibung, was funktionieren sollte, dies leider aber immer noch nicht tut...

Aufgabe 3: (30 Punkte) Protokoll-Version 0.2 – Parametrisierte Nachrichten

Das Protokoll, wie es in der Version 0.1 vorliegt, ist in seinen Fähigkeiten stark eingeschränkt. Deshalb soll das Protokoll in dieser Aufgabe so erweitert werden, daß es möglich ist bereits spezifizierte Nachrichten zu parametrisieren und zusätzlich Nutzdaten zu übertragen. Das so erweiterte Protokoll erhält dann die Versionsnummer 0.2. Im Detail ist folgendes zu tun:

- Die Versionsnummer lautet jetzt 0.2 !
- Eine Nachricht besteht aus einem Header und einem Body.
- Der Header sieht wie folgt aus:
  - Die erste Zeile jeder Nachricht sieht genau so aus wie bei Version 0.1, jedoch mit anderer Versions-Nummer.
  - Danach kommen beliebig viele nichtleere Zeilen, die aus einem Parameternamen, einem ':' und einem Wert bestehen:
    `<parameter> ':' <value> \r\n`
  - Der Header muß einen Parameter mit einer Längenangabe für den Body enthalten:
    `Content-Length: <n> \r\n`
    wobei n die Länge des Bodys in Bytes ist. Für eine Nachricht ohne Body ist die Länge mit 0 anzugeben, also
    `Content-Length: 0`
  - Der Header wird durch eine Leerzeile (also `\r\n`) beendet.
- Nach dem Header folgt der (evt. leere) Body.

Folgende Header-Zeilen *müssen* in alle (nicht-Handshake) Nachrichten eingbaut und bei Empfang extrahiert werden:

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

Für dieses Aufgabenblatt soll der `Application:`-Wert immer `none` sein.

Abzugeben sind:

- Dein Programm (sowohl Quelltext als auch ggf. compiliert im Falle von C, C++ und Java)

- Ggf. eine Beschreibung, was funktionieren sollte, dies leider aber immer noch nicht tut...

Details zur Abgabe der Aufgaben: siehe FAQ
(unterhalb `http://www.net.t-labs.tu-berlin.de/teaching/ws0708/PD_labcourse/`**).**
Abgabedatum: Dienstag, 11. Dezember 2007, 11:59h s.t.