TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Intelligente Netze und Management verteilter Systeme
Prof. Anja Feldmann, Ph. D.
Harald Schiöberg, Bernhard Ager, Vlad Manilici

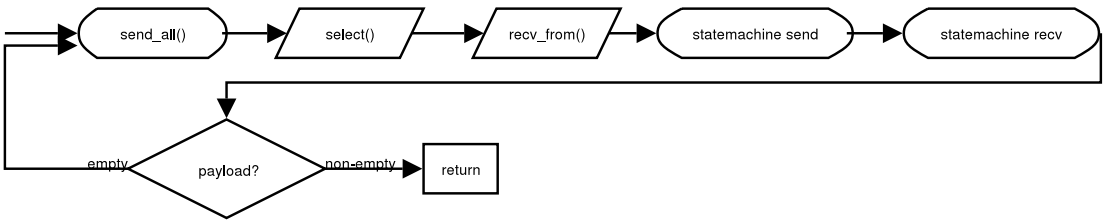
5. Übung zur Protokolldesign WS 08/09

Vorbemerkung

Da diese Aufgabe über mehrere Blätter geht, sollte man sich gut überlegen, welche Programmiersprache man verwenden will. Unabhängig von der Programmiersprache müssen allerdings einige vorgegebene Rahmenfunktionen implementiert werden.
Die Aufgabenstellung für dieses Blatt: Implementiere einen Sender und Empfänger für RDT 1.0. Passe deinen Server und Client-Dienst vom 2. Übungsblatt so an, dass er als Transportprotokoll RDT 1.0 verwendet. Der Sinn dieses Blattes ist es hauptsächlich, dass ihr ein gut strukturiertes Programm-Grundgerüst für die nächsten Blätter bekommt. Daher wird euch relativ genau vorgeschrieben, *wie* ihr manche Dinge programmieren sollt. Haltet euch genau daran, auch wenn es oft einfachere Wege gibt. Ihr werdet es dafür bei den kommenden Blättern viel einfacher haben.

Aufgabe 1: (70 Punkte) Funktionen Die folgenden Funktionen müssen implementiert werden. Natürlich könnt ihr weitere Hilfsfunktionen benutzen, aber die hier gennanten Funktionen müssen implementiert sein, und auch das tun, was in der Angabe steht.

- `rdt_connect()` Diese Funktion setzt alle Parameter, die notwendig sind, um sich mit einem Server zu verbinden. Ausserdem generiert sie den UDP-Socket.
- `rdt_listen()` Diese Funktion ist das serverseitige Gegenstück zu rdt_connect(). Sie öffnet den UDP-Socket für eingehende Verbindungen, und setzt alle Notwendigen Parameter.
- `rdt_send()` Diese Funktion nimmt eine beliebige Menge Daten als Parameter an, und schreibt sie in einen Sender-Puffer. Die Daten werden in dieser Funktion *nicht* wirklich auf das Netzwerk geschrieben.
- `rdt_recv()` Diese Funktion liefert den gesamten Inhalt des Empfänger-Puffers zurück. Sie liest *nicht* wirklich Daten vom Netzwerk, sondern nur aus dem Empfänger-Puffer.
- `rdt_select()` Diese Funktion macht die Hauptarbeit. Sie verhalt sich wie der Systemaufruf `select()`, d.h. sie blockiert so lange, bis Daten im Empfänger-Puffer vorliegen, die mit `rdt_recv()` gelesen werden können. Hier ist ein Flussdiagramm der Funktion:



  - `send_all()` Diese Funktion sendet alle Daten aus dem Senderpuffer an den Empfänger über das Netzwerk. Sollten mehr als 500 Bytes Daten im Sendepuffer sein, so müssen mehrere Pakete verschickt werden, da die MSS (Maximum Segment Size) eines RDT Packetes 534 Bytes ist.
  - `select()` Der bekannte Systemaufruf select, der wartet, bis Daten anliegen.
  - `recv_from()` Der bekannte Systemaufruf recv_from, der Daten vom Netzwerk liest.
  - `statemachine_send()` Das ist ein Funktionsaufruf, der das eben empfangene Packet bearbeitet. In RDT 1.0 passiert hier gar nichts, da der Sender keine (ACK-) Packete vom Receiver erwartet.

– `statemachine_recv()` Dieser Funktionsaufruf bearbeitet angekommene Datenpakete. Hier wird in RDT 1.0 nur der Header entfernt, und das Packet in den Empfängerpuffer kopiert.
– *Payload* Sollten Daten im Empfängerpuffer vorliegen, so kommt `rdt_select()` an dieser Stelle zurück. Andernfalls springt man wieder nach oben in die Funktion.

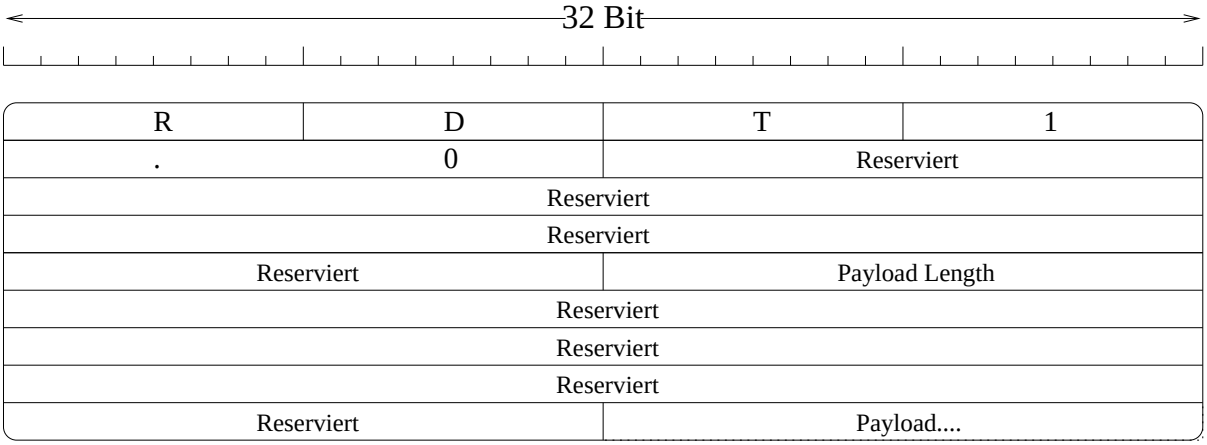
Aufgabe 2: (30 Punkte) Fileserver/Client Schreibe nun den Fileserver und Client aus Übungsblatt 2 so um, dass er statt der normalen Systemaufrufe die entsprechenden RDT Aufrufe benutzt.

Die Struktur eines RDT Paketes

Der Header von RDT 1.0 sieht wie folgt aus:

- 6 bytes RDT magic. Diese Bytes enthalten in ASCII-Codierung die Zeichen „RDT1.0“.
- 12 bytes reserviert. Diese Bytes enthalten immer den Wert 0.
- 2 bytes Payload length. Diese Bytes enthalten in Network Byte Order die Länge der Nutzdaten des Pakets in Bytes.
- 14 bytes reserviert. Diese Bytes enthalten immer den Wert 0.

Gesamtlänge des Headers ist damit 34 bytes. Danach folgen maximal 500 Bytes Nutzdaten.



Details zur Abgabe der Aufgaben:
siehe FAQ: `http://www.net.t-labs.tu-berlin.de/teaching/ws0809/PD_labcourse/faq.shtml`
Abgabedatum: Mittwoch, 26.11.2008, 23:59 s.t.