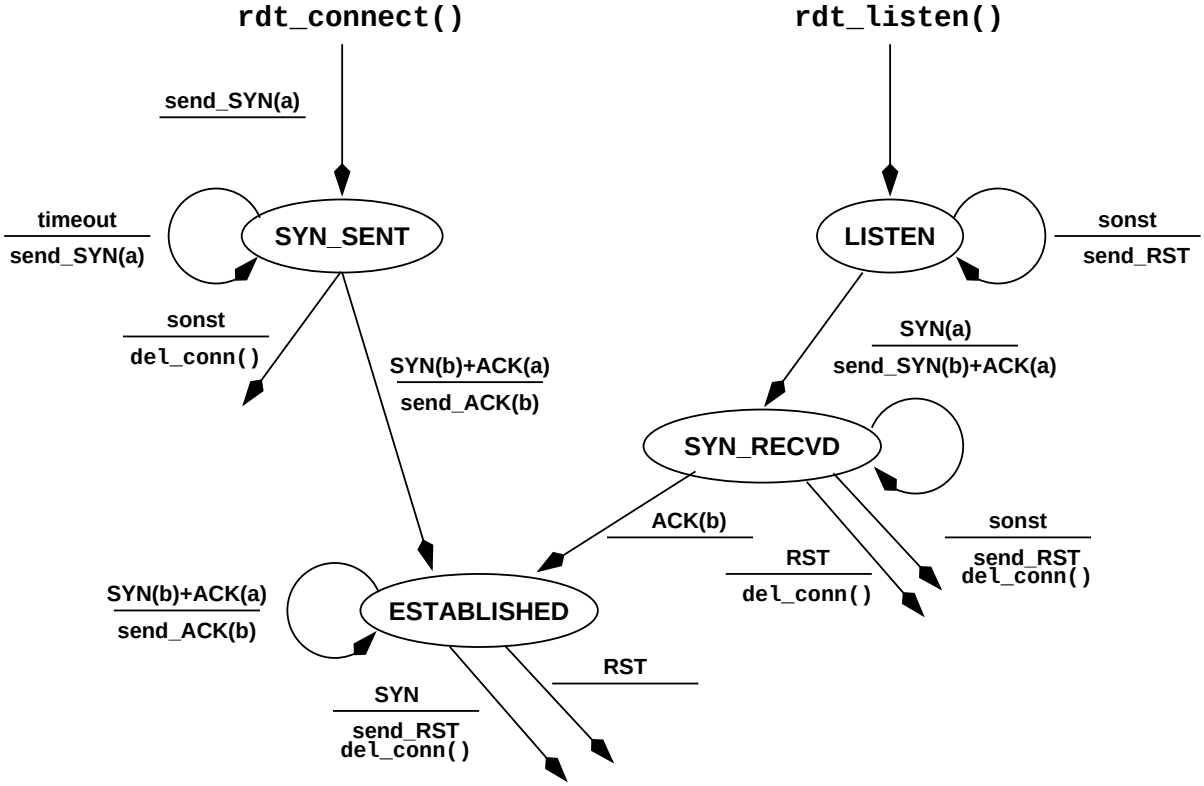
TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Intelligente Netze und Management verteilter Systeme
Prof. Anja Feldmann, Ph. D.
Harald Schiöberg, Bernhard Ager, Vlad Manilici

8. Übung zum Protokolldesign WS 08/09

Aufgabe 1: (50 Punkte) Verbindungsaufbau

In diesem Übungsblatt soll euere bestehende RDT Implementierung auf RDT 4.0 erweitert werden. Dieses kennt über RDT 3.0 hinaus die Funktionalität von Verbindungsauf- und -abbau. Zunächst wird der explizite Verbindungsaufbau eingeführt. Hierzu ist eine Erweiterung des RDT Headers um die 3 Flag Felder `SYN, FIN` und `RST` nötig. Aussderm wird das Magic auf `RDT4.0` geändert. Dazu ist folgender zusätzliche Zustandsautomat zu implementieren:



Im Zustand `ESTABLISHED` verhält sich RDT 4.0 genauso wie RDT 3.0. Der Aufruf von `rdt_listen` startet den Zustandsautomaten nun aber im Zustand `LISTEN`. In diesem Zustand wartet der Server auf ein Paket mit nur gesetztem `SYN` Flag. Jedes andere Paket wird mit einem `RST` beantwortet. Wird ein `SYN` empfangen, so antwortet der Server mit einem `SYN ACK` Paket und wechselt in den Zustand `SYN_RECVD`

Generell gilt: Bei einem `SYN` Paket darf die Sequenznummer frei gewählt werden (0 oder 1), dannach ist jedoch mit der darauffolgenden Sequenzenummer fortzufahren. `SYN`s sind mit der gleichen Sequenznummer zu bestätigen (genau wie normale Pakete).

Im Zustand `SYN_RECVD` erwartet der Server eine `ACK` Paket fuer sein verschicktes `SYN`. Trifft dieses ein, so wechselt er ohne weitere Aktion in den Zustand `ESTABLISHED`. Jedes andere Paket wird mit einem `RST` benantwortet.

Generell gilt: Nach dem versenden oder empfangen eines `RST` Packetes wird sofort, ohne weitere Aktion, der gesammte Zustand, der für eine Verbindung gespeichert ist gelöscht.

Clientseitig ist der Ast von `rdt_connect` zu implementieren. Wird `rdt_connect` aufgerufen, so wird sofort ein `SYN` Paket verschickt. Danach geht die Verbindung in den Zustand `SYN_SENT` über. In diesem Zustand wartet die Verbindung auf ein `SYN ACK` für das `SYN` Paket, wird dieses Erhalten, so bestätigt sie es, und geht in den Zustand `ESTABLISHED` über.

Generell gilt: In jedem dieser Zustände wird das zuletzt verschickte Paket beim Auftreten eines Timeouts neu verschickt. Ausserdem sollten die Aufrufe von rdt_send() und rdt_recv() einen Fehlercode an die Applikation liefern, wenn sich die Verbindung nicht im Zustand `ESTABLISHED` befindet.

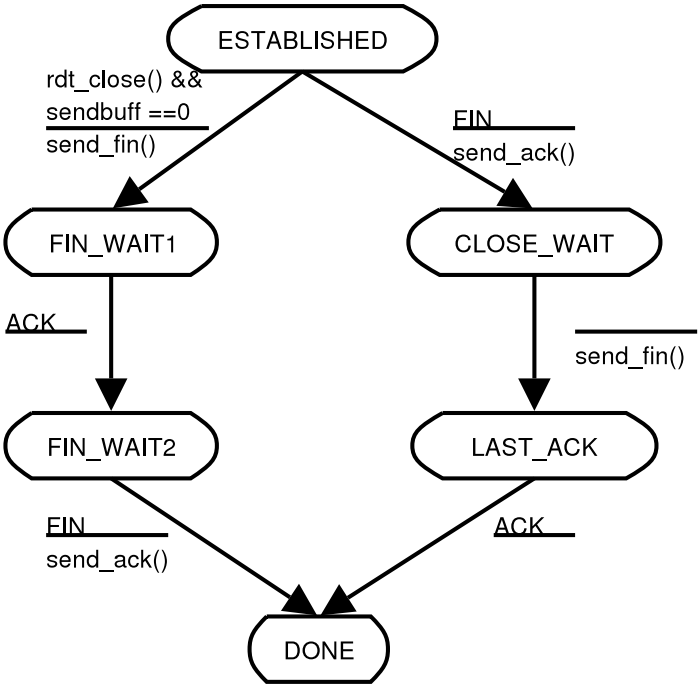
Aufgabe 2: (50 Punkte) Verbindungsabbau

Nun sollen Verbindungen auch wieder sauber beendet werden. Dazu ist die Funktion `rdt_close()` einzuführen. Diese setzt ein Flag (to_close) im Verbindungsstatus. Ist dieses gesetzt, so wird, sobald der Sendepuffer leer ist, ein Paket mit gesetztem `FIN` Flag verschickt, und die Verbindung geht in den Zustand `FIN_WAIT1` über. Wird ein `ACK` für diese `FIN` empfangen, so geht die Verbindung in den Zustand `FIN_WAIT2` über. In diesem Zustand wartet die Verbindung auf ein `FIN`, bestätigt dieses mit einem `ACK` und beendet sich danach.

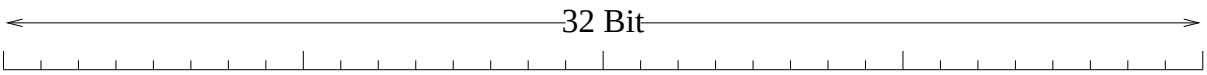
Diesem sogenannten *active close*, der durch den Aufruf von `rdt_close()` initiert wird steht der *passive close* gegenüber: Wird im Zustand `ESTABLISHED` ein Paket mit gesetztem `FIN` Flag empfangen, so wird dieses mit einem `ACK` bestätigt, und die Verbindung geht in den Zustand `CLOSE_WAIT` über. Wird in diesem Zustand ein `ACK` für das eben versandte `FIN` empfangen, so wird die Verbindung beendet.

Allgemein: Eine Verbindung zu beenden, heisst alle Daten, die zur Verbindung gehören aus dem Speicher zu löschen. Natürlich sind immer Prüfsummenchecks auf alle Pakete vorzunehmen, und bei allen verschickten Paketen der Timeout entsprechend zu setzen, um Neuübertragungen vornehmen zu können.

Hier ist die passende Statemachine:



Gennerell sollte man eine Funktion (z.B. statemachine_connection()) einführen, in der der state für den handshake und teardown bearbeitet wird. Die Funktionen statemachine_send() und statemachine_recv() werden dann nur noch angesprungen, wenn sich die Verbindung im Status `ESTABLISHED` befindet.
Hier nochmal der neue RDT-Header zum Überblick:



R	D	T	4
.	0	Reserviert	
Reserviert			
Reserviert			
Reserviert		Payload Length	
Seq number	Ack number	Reserviert Rst Fin Syn A	Checksum
Reserviert			
Reserviert			
Reserviert		Payload....	

Hier nochmal zur Klarstellung: Das `ACK` Flag hat den Wert 0x01, `SYN` den Wert 0x04, `FIN` den Wert 0x08 und `RST` den Wert 0x10. Die Bits 0x02, 0x20, 0x40 und 0x80 sind weiterhin reserviert.
Abzugeben sind:

- Der Quelltext Deines Programmes

—Ende der Aufgabenstellung für das 8. Aufgabenblatt.—

Abgabe: bis Mittwoch, 17.12.08 23:59h s. t.