# Diplomvorprüfung in

# **Technische Informatik 4**

## Sommersemester 2008

12. September 2008

**Name :** _____

**Matrikelnummer :** _____

**Geburtsdatum:** _____

**Studienfach:** _____

**Fachsemester:** _____

- **Bitte verwenden Sie einen blauen oder schwarzen Kugelschreiber (kein rot, keinen Bleistift).**
- **Schriftliche Aufzeichnungen (sowohl eigene Aufzeichnungen wie auch Bücher) sind als Hilfsmittel zugelassen. Auch ein Taschenrechner ist erlaubt und hilfreich. Nicht zugelassen sind dagegen Computer, PDAs, Mobiltelefone und sonstige Kommunikationsmittel.**
- **Legen Sie den Ausweis (mit Lichtbild) griffbereit auf den Platz.**
- **Bitte überprüfen Sie, ob Sie alle 17 Blätter erhalten haben.**
- **Schreiben Sie die Antworten jeweils in den freien Raum hinter den Fragen. Sollte dieser nicht ausreichen, steht noch freier Raum am Ende der Klausur zur Verfügung. Bitte kennzeichnen Sie dort deutlich, welche Aufgabe Sie bearbeiten. Gesondert beigelegte Blätter werden nicht bewertet!**
- **Schreiben Sie deutlich! Unleserliche Antworten gehen nicht in die Bewertung ein!**

**Ich habe die Hinweise auf dieser Seite zur Kenntnis genommen und alle 17 Blätter der Klausur und die Arbeitskopie empfangen:**

_____

**Unterschrift**

Bewertung:

| 1 | 2 | 3 | 4 | 5 | Σ |
|---|---|---|---|---|---|
| | | | | | |

## *Aufgabe 1: Allgemeine Fragen (20 Punkte)*

Bitte antworten Sie kurz, möglichst in ganzen Sätzen.

a) (3 Punkte) Geben Sie die Protokollschichten des Internet hierarchisch von oben nach unten an und ordnen Sie die folgenden typischen Aufgaben der entsprechenden Schicht zu: Wegewahl, Dateitransfer, Medienzugriff, Modulation, Überlastkontrolle

| Nr. | Protokollschicht | Aufgabe |
|---|---|---|
| 5 | | |
| 4 | | |
| 3 | | |
| 2 | | |
| 1 | | |

b) (1 Punkt) Nennen Sie eine Anwendung (oder ein Anwendungsprotokoll), die/das UDP benutzt!

c) (2 Punkte) Wie viele TCP-Verbindungen sind jeweils beim Abruf einer Basis-HTML-Datei mit zwei eingebetteten Objekten in den folgenden Fällen nötig?

| | |
|---|---|
| bei nicht-persistentem HTTP ohne parallele Verbindungen | |
| bei nicht-persistentem HTTP mit parallelen Verbindungen | |
| bei persistentem HTTP ohne Pipelining | |
| bei persistentem HTTP mit Pipelining | |

Tragen Sie die Anzahl der benötigten TCP-Verbindungen in die rechte Spalte ein.

d) (2 Punkte) Wer führt womit ein Longest-Prefix-Match durch und wozu?

e) (2 Punkte) Wird ein Schiebefensterprotokoll mit gegebener Fenstergröße über ein lokales Netz oder über eine Satellitenverbindung einen höheren Durchsatz erreichen? Begründen Sie Ihre Antwort!

f) (2 Punkte) Geben Sie Klasse, Netzwerk- und Hostanteil der IP-Adresse 128.128.255.255 an! Handelt es sich um eine Ziel- oder Quelladresse? Begründen Sie Ihre Antwort.

g) (1 Punkt) Was versteht man unter Speichervermittlung?

h) (1 Punkt) Was sagt die Kanalpuffergröße aus?

i) (4 Punkte)

I) Wie berechnet sich der Durchsatz des Multi-Token-Ring, wenn jede Station bei Erhalt eines freien Tokens nicht immer, sondern nur mit Wahrscheinlichkeit p einen Rahmen sendet? Modifizieren Sie dazu die Ihnen bekannte Formel

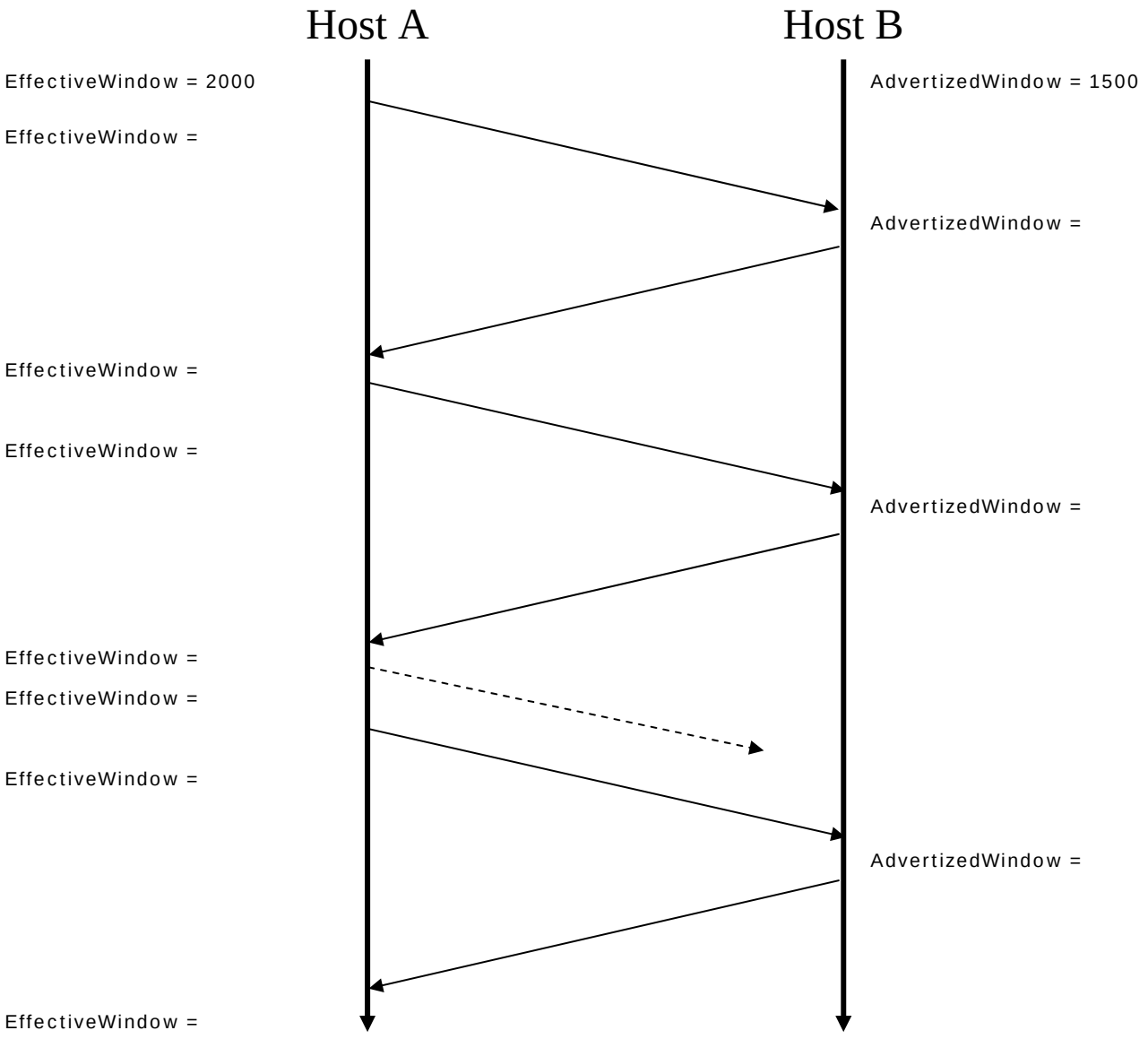
$$S_{max}=(NL/R)/(NL/R + \tau)$$

II) Unter welcher Bedingung an Ringlatenz $\tau$ des Multi-Token-Ring und an die maximale Ausbreitungsverzögerung D des CSMA/CD-Protokolls, kann CSMA/CD einen größeren Durchsatz erreichen? Nehmen Sie dazu für den Multi-Token Ring dieselbe Sendewahrscheinlichkeit für jede Station wie bei CSMA/CD an (und benutzen Sie Ihre modifizierte Formel aus der vorigen Teilaufgabe).

j) (2 Punkte) Wie müssen sich Signal- und Rauschleistung zueinander verhalten, wenn die maximale Datenrate in einem rauschbehafteten Kanal größer als dessen Bandbreite sein soll?

## *Aufgabe 2: Transportschicht (25 Punkte)*

a) (17 Punkte) Nehmen Sie an, dass Host A Daten an Host B schicken will und dazu eine TCP-Verbindung zu Host B aufbaut. Inklusive des Verbindungsaufbaus sieht die Segmentfolge wie unten dargestellt aus, wobei ein Segment verloren geht (gestrichelter Pfeil!). Wann immer möglich (in dieser Segmentfolge), schickt Host A 500 Bytes an Host B. Host A wählt als Startsequenznummer 1000, Host B wählt 10. Tragen Sie für alle Segmente die Sequenzummern (sqn), Acknowledgment-Nummern (ack), die Größe der Nutzdaten sowie eventuell gesetzte TCP-Flags in das Diagramm ein. Es sind keine Pfeile hinzuzufügen. Dabei soll auch die Flusskontrolle von Host B auf Host A berücksichtigt werden: Host A startet mit einem EffectiveWindow der Größe 2000, Host B mit einem AdvertizedWindow der Größe 1500. Fügen Sie für beide Seiten die entsprechenden Fenstergrößen an den vorgesehenen Stellen hinzu, wobei Sie annehmen, dass Host B in dem dargestellten Zeitabschnitt keine Bytes an die Anwendung ausliefert.






Bitte umseitige Zusatzfrage beachten!

Zusatzfrage zu Teil a):

Wie reagieren Sender und Empfänger am Ende der oben dargestellten Segmentfolge? Beschreiben Sie kurz das jeweilige Verhalten!

b) (8 Punkte) Eine Webseite soll mit nicht-persistentem HTTP übertragen werden. Die Webseite besteht aus einer HTML-Basisseite der Größe **7 S** mit einem einzigen eingebetteten Objekt der Größe **14 S**, wobei **S** die Größe eines TCP-Segments in Bits ist. Die dynamische Fenstergröße (in Anzahl von Segmenten) einer TCP-Verbindung soll sich ausschließlich nach dem Slow-Start-Mechanismus entwickeln. Die Übertragungsrate **R** und die Round Trip Time **RTT** werden als idealisiert konstant angenommen.

- Berechnen Sie die Antwortzeit der Übertragung (in Abhängigkeit von **S** und **R**) für den Fall, dass **RTT = 4 S/R** gilt.
- Veranschaulichen Sie das Übertragungsverhalten an Hand einer Skizze, aus der hervorgeht, wann Wartezeiten auftreten und in welchen Fenstern wieviele Segmente übertragen werden!
- In welchem Verhältnis stehen die Latenzen für das eingebettete Objekt und die halb so große HTML-Basisseite?

## *Aufgabe 3: Socket-Programmierung (25 Punkte)*

Ein HTTP-Proxy ist ein Vermittler zwischen einem Web-Client und einem HTTP-Server. Er nimmt die HTTP-Requests der Browser entgegen, extrahiert aus dem HTTP-Request den Server, baut eine Verbindung zu diesem auf und leitet den HTTP-Request an diesen weiter. Die Antwort wird byteweise (ohne Berücksichtigung des Inhalts) an den Client zurückgegeben.

**Aufgabe:**

Implementieren Sie die oben geschilderte Funktionalität des HTTP-Proxys in JAVA. Bauen sie Ihre Implementierung auf dem nachfolgend gegebenen Code auf, leiten Sie die Klasse **ProxyRequest** von der Klasse Request ab und implementieren sie die Methode **handle()**. Berücksichtigen Sie außerdem folgende Punkte:

- Client und Server verwenden HTTP/1.0 .
- Verwenden Sie die in der Klasse Request stehenden Variablen.
- Der Client fragt immer den Server auf Port 80, d.h. der Port muss nicht aus dem HTTP-Request extrahiert werden.
- Tritt bei der Verbindungsaufnahme zum Server ein Fehler auf, ist eine entsprechende Fehlermeldung an den Client zurückzugeben (HTTP-Response). Anschließend ist die Anfrage „sauber“ zu beenden.
- Ansonsten sind keine Fehler zu behandeln.

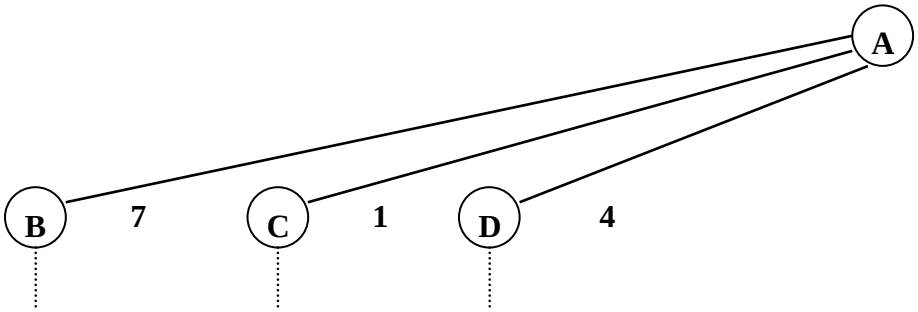
Gegeben ist folgender Quellcode:

```
// Hauptklasse des HTTP-Proxys
import java.net.*;

public class Proxy {
  public static void main(String args[]) throws Exception {
    ServerSocket s = new ServerSocket(8000);
    Socket socket;

    while(true) {
     socket = s.accept();
     Request r = new ProxyRequest();    // Ihre Klasse
     r.setClientSocket(socket);
     Thread t = new Thread(r);
     t.start();
    }
  }
}

// Abstrakte Klasse des Proxy-Requests
// Leiten Sie hiervon Ihre ProxyRequest-Klasse ab
```

```
import java.io.*;
import java.net.*;

public abstract class Request implements Runnable {
  public final static int BUFFER = 1048576;

  // Verwenden Sie nachfolgende Variablen
  protected Socket client, server;  // Socket zum Client/Server
  protected OutputStream     cOut;  // Verbindung zum Client
  protected BufferedReader   cIn;   // Verbindung vom Client
  protected DataOutputStream sOut;  // Verbindung zum Server
  protected InputStream      sIn;   // Verbindung vom Server

  public void setClientSocket(Socket s) {
    client = s;
  }

  public void run() {
    try {
       handle();
    } catch(Exception e) {System.out.println(e);};

    try {
        cIn.close(); cOut.close(); client.close();
        sIn.close(); sOut.close(); server.close();
    } catch(Exception e1) {System.out.println(e1);}
  }

  // Von Ihnen in der abgeleiteten Klasse zu implementieren !
  protected abstract void handle() throws Exception;
}
```

## *Aufgabe 4: Routingverfahren (15 Punkte)*

Der Knoten A hat in einem Netzwerk die Nachbarn B, C und D und kennt die angegebenen Kosten für die jeweiligen direkten Verbindungen:




Weiterhin gibt es die Knoten E und F im Netz, die A nur über seine direkten Nachbarn erreichen kann. Führen Sie den Distanz-Vektor-Algorithmus aus der Sicht des Knoten A durch. Geben Sie zunächst die Initialisierung der Distanztabelle von Knoten A an.

| **von A zu** | $\mathbf{D_A(\cdot)}$ | $\mathbf{nh_A(\cdot)}$ |
|---|---|---|
| **A** | | |
| **B** | | |
| **C** | | |
| **D** | | |
| **E** | | |
| **F** | | |

In den folgenden Schritten senden die Knoten B, C und D simultan die Distanzinformationen aus ihren Distanztabellen an A (siehe Spalten 2, 3 und 4 auf den nächsten Seiten). Die Notation ist wie in der Vorlesung gewählt (d.h., $\mathbf{nh_A(\cdot)}$ bezeichnet den Vektor der aktuellen nächsten Knoten auf den kürzesten Wegen von A zu den anderen Knoten). Füllen Sie die leeren Felder der nachfolgenden Tabellen: dabei sollen in den Spalten $\mathbf{D_A(\cdot)}$ und $\mathbf{nh_A(\cdot)}$ die vom Knoten A errechnete Distanztabelle nach Erhalt der Nachbarinformationen des jeweiligen Schrittes stehen. Fügen Sie außerdem in den jeweiligen letzten Spalten die Distanzinformationen ein, die Knoten A im nächsten Schritt an seinen Nachbar C schicken würde, wenn er Poisoned Reverse anwenden würde.

Markieren Sie die Elemente in den Tabellen, die sich von einem Schritt auf den nächsten verändert haben.

**Schritt 1:**

| | $D_B(\cdot)$ von B | $D_C(\cdot)$ von C | $D_D(\cdot)$ von D | von A zu | $D_A(\cdot)$ | $nh_A(\cdot)$ | $D_A(\cdot)$ an C |
|---|---|---|---|---|---|---|---|
| **A** | 7 | 1 | 4 | **A** | | | |
| **B** | 0 | ∞ | ∞ | **B** | | | |
| **C** | ∞ | 0 | 2 | **C** | | | |
| **D** | ∞ | 2 | 0 | **D** | | | |
| **E** | 3 | ∞ | 1 | **E** | | | |
| **F** | 1 | 5 | ∞ | **F** | | | |

**Schritt 2:**

| | $D_B(\cdot)$ von B | $D_C(\cdot)$ von C | $D_D(\cdot)$ von D | von A zu | $D_A(\cdot)$ | $nh_A(\cdot)$ | $D_A(\cdot)$ an C |
|---|---|---|---|---|---|---|---|
| **A** | 7 | 1 | 3 | **A** | | | |
| **B** | 0 | 6 | 4 | **B** | | | |
| **C** | 6 | 0 | 2 | **C** | | | |
| **D** | 4 | 2 | 0 | **D** | | | |
| **E** | 2 | 3 | 1 | **E** | | | |
| **F** | 1 | 5 | 2 | **F** | | | |

**Schritt 3:**

| | $D_B(\cdot)$ von B | $D_C(\cdot)$ von C | $D_D(\cdot)$ von D | von A zu | $D_A(\cdot)$ | $nh_A(\cdot)$ | $D_A(\cdot)$ an C |
|---|---|---|---|---|---|---|---|
| **A** | 7 | 1 | 3 | **A** | | | |
| **B** | 0 | 6 | 3 | **B** | | | |
| **C** | 6 | 0 | 2 | **C** | | | |
| **D** | 3 | 2 | 0 | **D** | | | |
| **E** | 2 | 3 | 1 | **E** | | | |
| **F** | 1 | 4 | 2 | **F** | | | |

**Schritt 4:**

| | $D_B(\cdot)$ von B | $D_C(\cdot)$ von C | $D_D(\cdot)$ von D | von A zu | $D_A(\cdot)$ | $nh_A(\cdot)$ | $D_A(\cdot)$ an C |
|---|---|---|---|---|---|---|---|
| **A** | 7 | 1 | - | **A** | | | |
| **B** | 0 | 5 | - | **B** | | | |
| **C** | 5 | 0 | - | **C** | | | |
| **D** | 3 | 2 | - | **D** | | | |
| **E** | 2 | 3 | - | **E** | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **F** | 1 | 4 | - | **F** | | | |

**Schritt 5:**

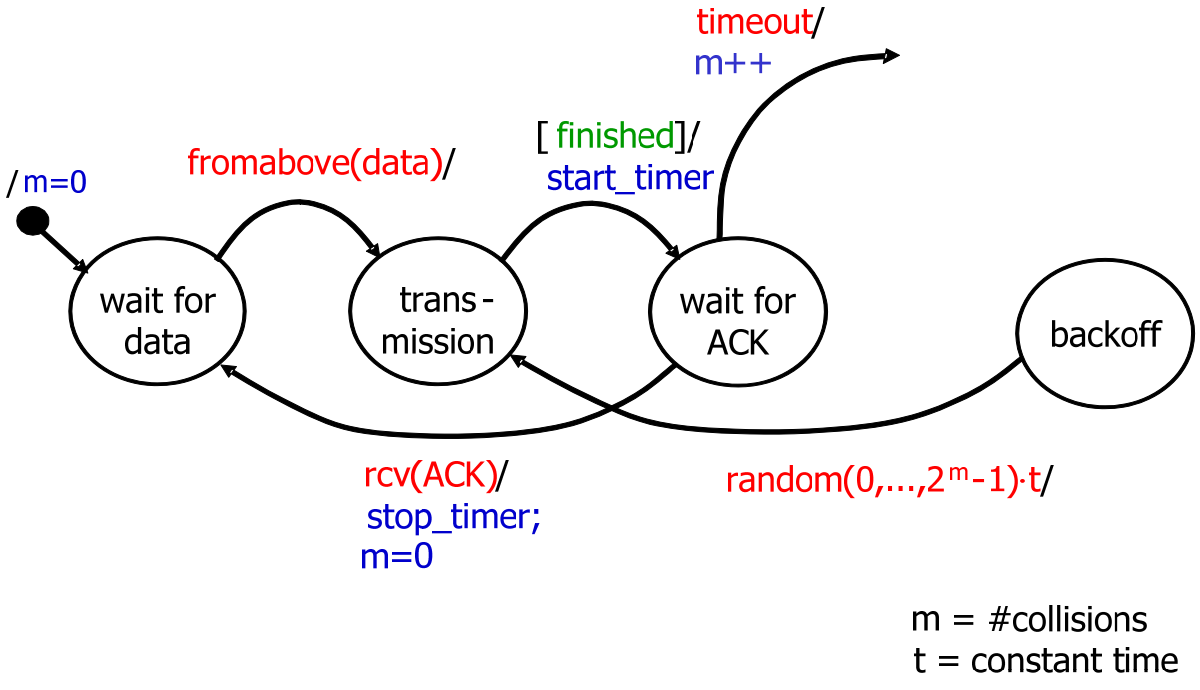
| | $D_B(\cdot)$ von B | $D_C(\cdot)$ von C | $D_D(\cdot)$ von D | von A zu | $D_A(\cdot)$ | $nh_A(\cdot)$ | $D_A(\cdot)$ an C |
|---|---|---|---|---|---|---|---|
| **A** | <u>6</u> | - | - | **A** | | | |
| **B** | 0 | - | - | **B** | | | |
| **C** | 5 | - | - | **C** | | | |
| **D** | 3 | - | - | **D** | | | |
| **E** | 2 | - | - | **E** | | | |
| **F** | 1 | - | - | **F** | | | |

**Schritt 6:**

| | $D_B(\cdot)$ von B | $D_C(\cdot)$ von C | $D_D(\cdot)$ von D | von A zu | $D_A(\cdot)$ | $nh_A(\cdot)$ | $D_A(\cdot)$ an C |
|---|---|---|---|---|---|---|---|
| **A** | - | - | - | **A** | | | |
| **B** | - | - | - | **B** | | | |
| **C** | - | - | - | **C** | | | |
| **D** | - | - | - | **D** | | | |
| **E** | - | - | - | **E** | | | |
| **F** | - | - | - | **F** | | | |

## *Aufgabe 5: Verbindungsschicht (15 Punkte)*

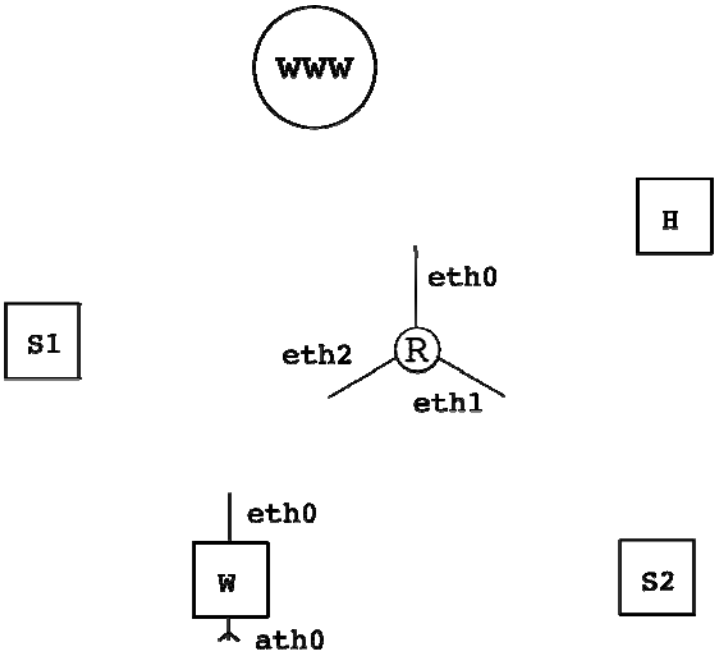
a) (6 Punkte) Gegeben ist das aus Vorlesung und Übung bekannte Statechart des ALOHA–Protokolls (Sender). Erweitern Sie es so, dass der Sender nach ***M*** Fehlversuchen (Timeouts) einen Fehler an die obere Schicht meldet und ansonsten den Backoff wie gewohnt ausführt.



b) (9 Punkte) Verbinden Sie die unten erläuterten Komponenten so, dass alle folgenden Kriterien erfüllt sind:

- Die vier PCs pc1, pc2, pc3, pc4 besitzen einen Internetzugang (www).
- pc5 kann ausschließlich den von pc1 und pc2 erzeugten Verkehr ins www auf der physikalischen Schicht überwachen, d.h. mithören.
- pc1 und pc2 sind in derselben Broadcast-Domäne (können von einem Broadcast auf Schicht 2 erreicht werden), befinden sich aber in unterschiedlichen Kollisionsdomänen.
- pc3 und pc4 befinden sich zusammen in einem Subnetz, das von dem, in dem sich pc1 und pc2 befinden, verschieden ist.

Tragen Sie pc1 bis pc5 sowie alle nötigen Verbindungen in die Abbildung ein!




**Komponenten:**

| | | |
|---|---|---|
| pc1, pc2, pc3, pc5 | : | PC mit Ethernetschnittstelle eth0 |
| pc4 | : | PC mit WLAN-Schnittstelle ath0 |
| S1, S2 | : | Switch |
| H | : | Hub |
| R | : | Router mit Ethernetschnittstellen eth0, eth1, eth2 |
| W | : | WLAN-Accesspoint mit<br>- Funkschnittstelle ath0<br>- Ethernetschnittstelle eth0 |
| www | : | Zugang zum Internet über Ethernetschnittstelle eth0 |

# Arbeitskopie - Quellcode Aufgabe 3:

```
// Hauptklasse des HTTP-Proxys
import java.net.*;

public class Proxy {
  public static void main(String args[]) throws Exception {
    ServerSocket s = new ServerSocket(8000);
    Socket socket;

    while(true) {
     socket = s.accept();
     Request r = new ProxyRequest();  // Ihre Klasse
     r.setClientSocket(socket);
     Thread t = new Thread(r);
     t.start();
    }
  }
}


// Abstrakte Klasse des Proxy-Requests
// Leiten Sie hiervon Ihre ProxyRequest-Klasse ab
import java.io.*;
import java.net.*;

public abstract class Request implements Runnable {
  public final static int BUFFER = 1048576;

  // Verwenden Sie nachfolgende Variablen
  protected Socket client, server;  // Socket zum Client/Server
  protected OutputStream     cOut;  // Verbindung zum Client
  protected BufferedReader   cIn;   // Verbindung vom Client
  protected DataOutputStream sOut;  // Verbindung zum Server
  protected InputStream      sIn;   // Verbindung vom Server

  public void setClientSocket(Socket s) {
    client = s;
  }

  public void run() {
    try {
       handle();
    } catch(Exception e) {System.out.println(e);};

    try {
        cIn.close(); cOut.close(); client.close();
        sIn.close(); sOut.close(); server.close();
    } catch(Exception e1) {System.out.println(e1);}
  }

  // Von Ihnen in der abgeleiteten Klasse zu implementieren !
  protected abstract void handle() throws Exception;
}
```