Paul Masch und Clemens Heffner

**Netzwerke I Praktium: Blatt 7-8**

Datum: 14.12.2015

**Aufgabe 7.1: Implementierung einer zuverlässigen Datenübertragung über UDP**

**b)**

Das Paket braucht eine **Checksum**, um Bitfehler erkennen zu können.

Der Empfänger muss den Sender über Bitfehler mit einer **Bestätigung(ACK)** bzw. Nicht-Bestätigung(NAK) informieren können. Der Empfänger muss wissen ob das gesendete Paket eine Retransmission ist oder nicht also braucht es eine

**Sequenznummer**.

struct packet {

int seqNr;

int ackNr

bool isAck;

int checksum;

}

**g)**

Um eine Datei mit 669.097 Bytes zu übertragen braucht das Programm, ohne das s sFehler und Retransmission auftreten, zwischen 0,229s-0,235s.

**h)**

Im Vergleich dazu braucht TCP zur Übertragung der selben Datei mit 669.097 Bytes 0,0092s -> TCP ist 24 mal schneller als das Programm mit Stop-and-Wait

Das Programm wird wohl nie schneller sein als TCP, da es ähnlich wie TCP auch ein RTT zur Verbindung braucht und da es Stop-und-Wait als Protokoll verwendet und nicht auf Betriebssystemebene arbeitet sondern auf einer virtuellen maschine (c#).

Auch der Overhead von TCP 20Bytes (ohne Optionen) ist kleiner als der von mir entwickelte Header (19Bytes+UDP header = 27Bytes).

**Aufgabe 7.2: TCP Flow Control**

**a)**

Da der Rechner B die Daten langsamer verarbeiten kann als sie empfangen werden, wird der Empfangspuffer je nach Größe nach einiger Zeit voll sein. Rechner B teilt Rechner A über den restlichen Speicherplatz im Empfangspuffer mit jedem ACK mit. Ist kein Speicher im Empfangspuffer von Rechner B vorhanden, drosselt Rechner A daraufhin die Verbindung, solange bis der Empfangspuffer wieder Platz hat.

**b)**

Bei Rechner A stellt sich eine Senderate von 20Mbit/s ein.

**c)**

Rechner A würde in dieser Zeit die Übertragung der Datei stoppen, wenn der Empfangspuffer von B bereits voll ist und stattdessen 1 Byte Pakete an B schicken bis dieser den Empfangspuffer leert und im ACK des 1 Byte Paketes Rechner A dies mitteilt, sodass A mit der Übertragung fortsetzt.