|  |
| --- |
| Georgia Institute of Technology  CS 3251 – A |
| Reliable Transfer Protocol (RxP) |
| Homework 4:  Protocol Specification |

|  |
| --- |
| Tyler M. Smith  Kyle Rabago-Banjo  November 4, 2015 |

In the first page of the report (after the cover page), please provide clear answers to the following questions: 

* -  Is your protocol non-pipelined (such as Stop-and-Wait) or pipelined (such as Selective Repeat)?
  + Pipelined: selective repeat. Send a window-size number of packets. Resend ones that are NACKed or time out
* -  How does your protocol handle lost packets?
  + Lost packets are caught by a timeout and retransmitted
* -  How does your protocol handle corrupted packets?
  + Corrupted packets detected by checksum. Receiver sends a NACK packet
* -  How does your protocol handle duplicate packets?
  + Duplicate packets are ACKed again. If more than one ACK is received by the sender, they are ignored.
* -  How does your protocol handle out-of-order packets?
  + Packets are buffered according to sequence number. Once all packets are received, they are combined into the file
* -  How does your protocol provide bi-directional data transfers?
  + Either side can send or receive DATA, ACK, or NACK packets. Only difference in server and client is during connection
* -  Does your protocol use any non-trivial checksum algorithm (i.e., anything more sophisticated than the IP checksum)?
  + Uses alg. Similar to IPv4: sum 16-bit values, take one’s complement (or different. Doesn’t matter)

REQUIREMENTS:

a. RxP must be as reliable as TCP

b. RxP must be connection-oriented

c. RxP must provide window-based flow control

d. RxP must provide byte-stream communication semantics (as TCP does).

DESIGN SPECIFICATION MUST INCLUDE:

High-Level Description of RxP

How it works

Any special features

Description (and diagram) of RxP header structure

Fields. Sizes

* Sequence number
* Total number of segments
* Total size of file
* Type (DATA, ACK, NACK)
* Payload size
* Header size
* Checksum (header and payload)

Finite state-diagram for client and server

States, system calls, packets received and sent (similar to TCP state diagram)

Definition of API (think JavaDocs)

Available (visible) functions, values, etc.

Algorithms used: Corruption detection, anything that isn’t trivial

NOTE: Protocol spec must support all features for full credit.