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

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

Reliable Transfer Protocol (RxP) is designed to allow users to reliably transfer files across a network. As such, it behaves as an Application Layer extension to TCP, even though it is built upon UDP. Consequently, it must handle data loss and corruption in the following ways:

1. Lost packets are retransmitted after a timeout period (calculated in a similar manner to TCP)
2. Corrupted packets are detected by a checksum value transmitted within each packet. Upon receipt of a corrupted packet, the receiver notifies the sender with a NACK packet (see Packet Types).
3. All non-corrupt packets are ACKnowledged by the receiver. Only one ACK is accepted per sent packet. All subsequent ACKnowledgments for the same packet are ignored.
4. All packets are sent with an order number. The receiver buffers packets according to their order numbers as they are received.
5. Once connected, both the server and client can send DATA, ACK, and NACK packets. No distinction is made between client and server after connection.
6. Each packet has a checksum attached which provides protection against corruption. See Appendix A: Algorithms for the Checksum Algorithm. (Give basic description here)

RxP uses the Sliding Window, Selective Repeat ARQ to transfer data. The window size can be specified by the user (see Commands), and specific packets are retransmitted upon receipt of a NACK.

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:

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.

ENCODING PACKETS AND SENDING: (Python3)

a = '1110011011011010100001110101010101010110101010101011010101010101010'

>>> h = hex(int(a,2))

>>> h

'0x736d43aaab555aaaa'

>>> h[2:].encode()

b'736d43aaab555aaaa'

>>> s = h[2:].encode()

>>> h2 = s.decode()

>>> h2

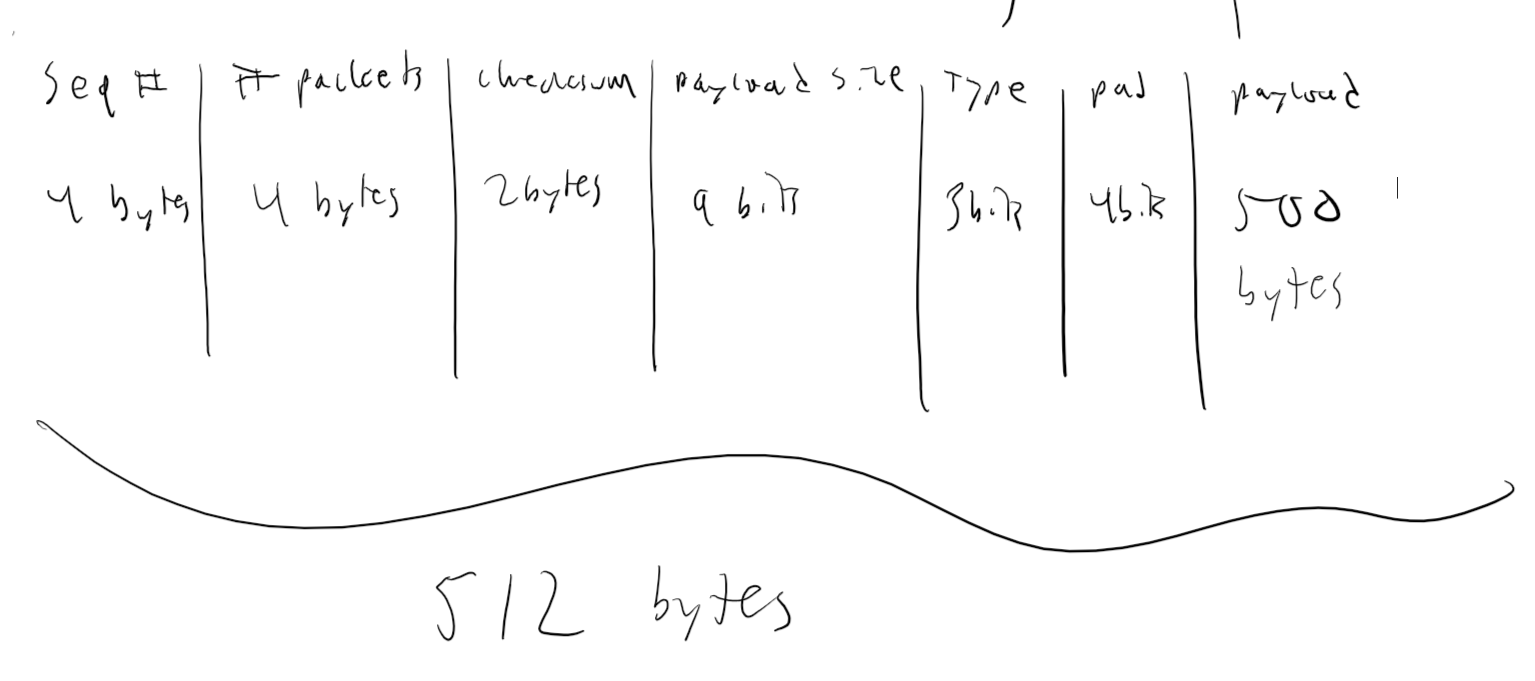
'736d43aaab555aaaa'

>>> i = int(h2,16)

>>> b2 = bin(i)

>>> b2

'0b1110011011011010100001110101010101010110101010101011010101010101010'

Packet Header:  


Need 2 bytes for Window Size. Take from payload and put before payload size.