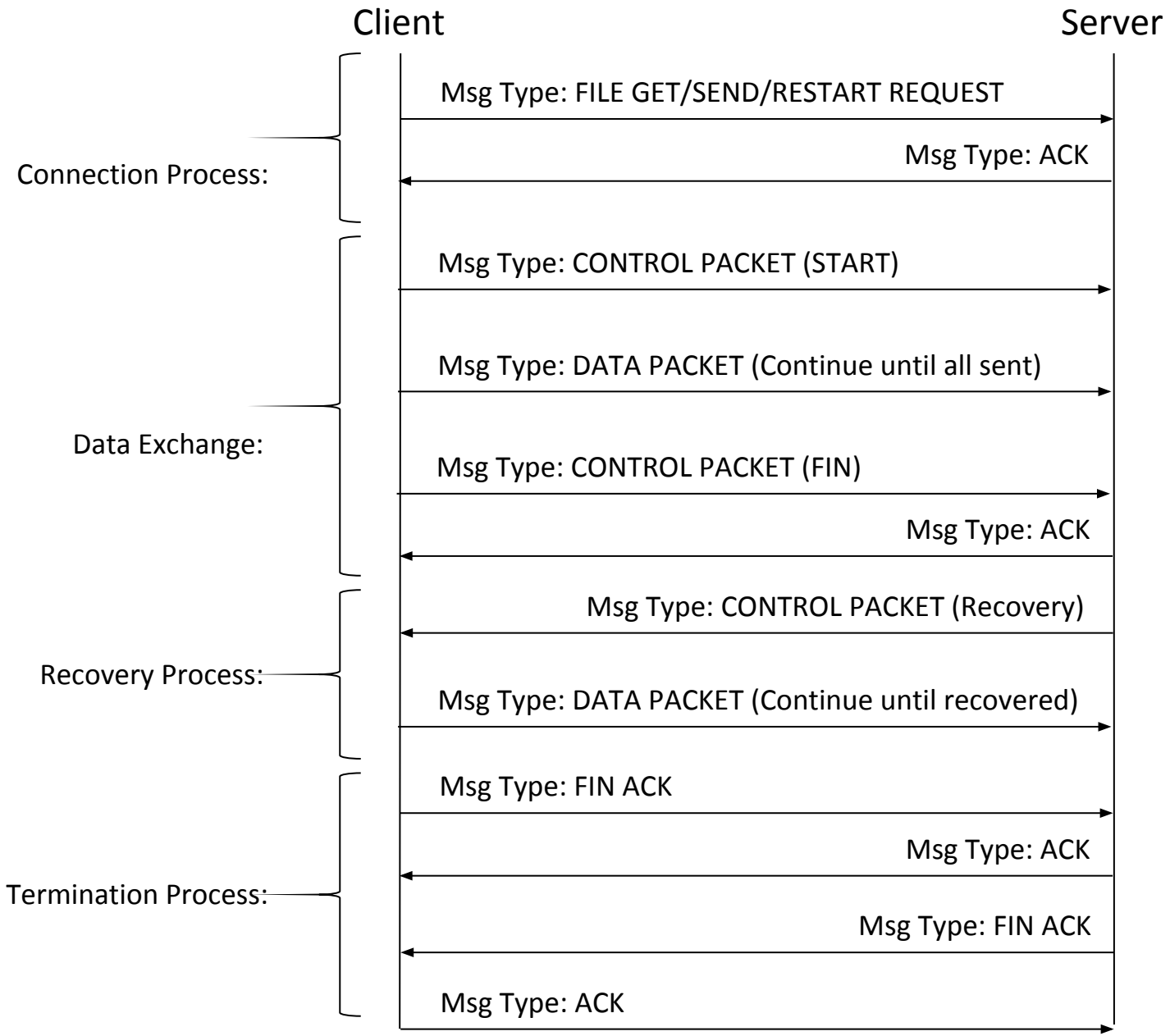
PROTOCOL DESIGN TEMPLATE

Dan and Jon's Wings (and Thodoris and Josh)

Message Flow Template



Software Implementation Considerations

The software client must implement both the file sending and receiving parts. Both should listen on ports 10512 (control packets) and 10256 (data packets) by default (or can be configured otherwise).

Connection initiation

- Randomly generate a transaction ID, which will be sent on the initial file transfer request. The receiver of the file transfer request may indicate that this transaction ID is not unique, and this repeats until the transaction ID is indeed unique.

Software Implementation Considerations

ACKs:

- ACKs are sent on response of all control packets (which are not as numerous).
- ACKs are sent from file receiver every **n** data packets (**n** will be fixed based on recovery attempt number; i.e., initially **n**=2^8, then **n**=2^7, … **n**=2^1). Specific implementation of strict congestion control can be left up to the sender implementation (for example, compare time to send **n** number packets vs getting an ACK back from the receiver regarding those **n** packets, or slow down after a few lost ACKs).
- One way for the connection termination is after **z** lost consecutive ACKs (which would indicate that the receiver is not still connected).

Restart mechanism:

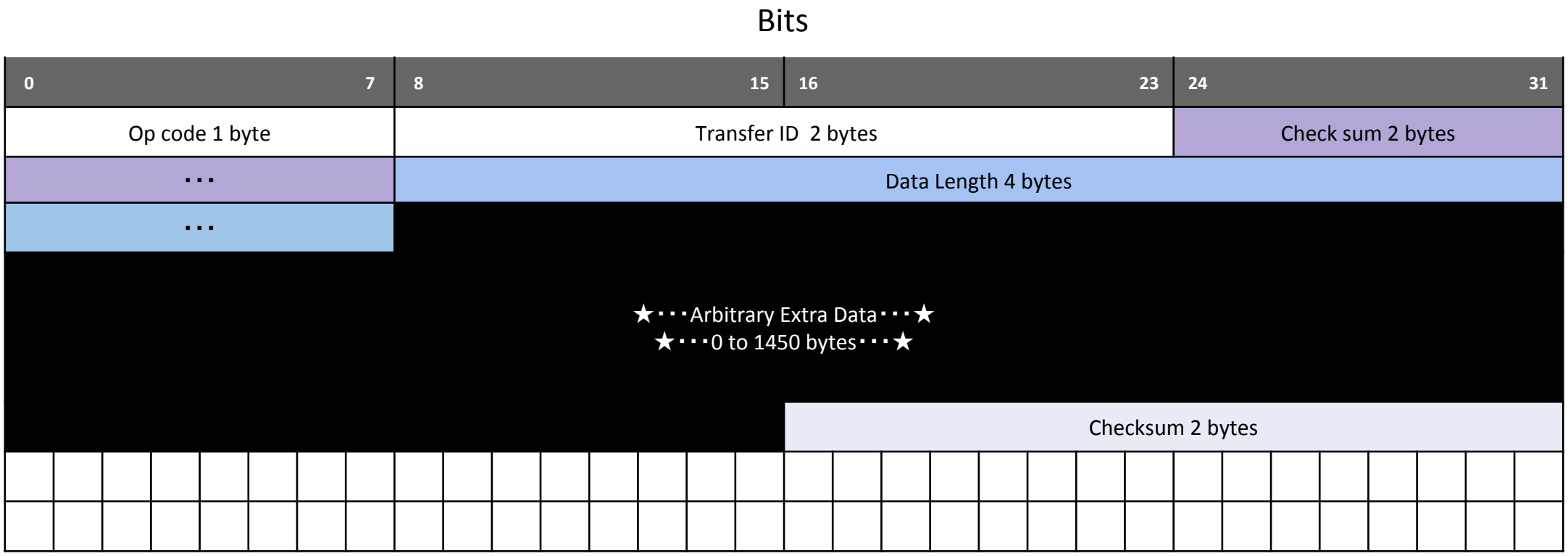
- Implemented purely software-side on file receiver side; if connection closed with some error, can retry seamlessly and just request the remaining packets with special restart opcode (basically starting file request and then immediately starting recovery with specified packets).

Recovery Process

- After all data packets sent, if any packets are lost, receiver sends a control packet specifying what packets were lost.
  - Flow loops back to sender sending data packets until all lost data packets sent.
  - If packets are still lost, flow loops back again to sending data packets.
  - This process loops **y** number of times, depending on the size of the file being sent.
  - If packets are still lost after **y** tries, terminate connection.

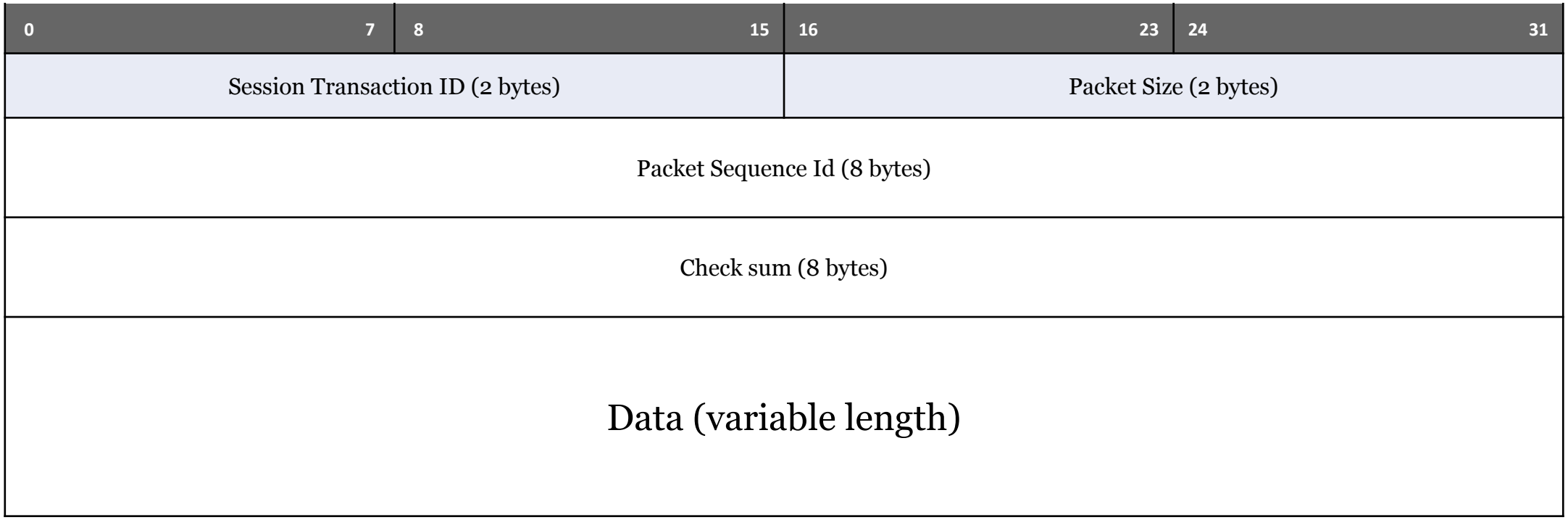
We can let **n** (number of packets between ACKs), **y** (number of recovery attempts), **z** (number of consecutive ACKs before connection is lost), be determined by the specific software implementation (or tweaked manually when starting the implementation).

Control Header Template ≪port:10512≫



Data Header Template ≪port:10256≫

Bits



Control Header Values

Header	Values	Meaning/Derivation/Definition
TID	2 bytes	Transfer ID
Opcode	1 byte	Opcode
Data length	4 bytes	Length of data (in bytes)
Data	0-1450 bytes	Arbitrary length extra data (dependent on operation)
Checksum	2 bytes	Checksum

Control Header Opcodes

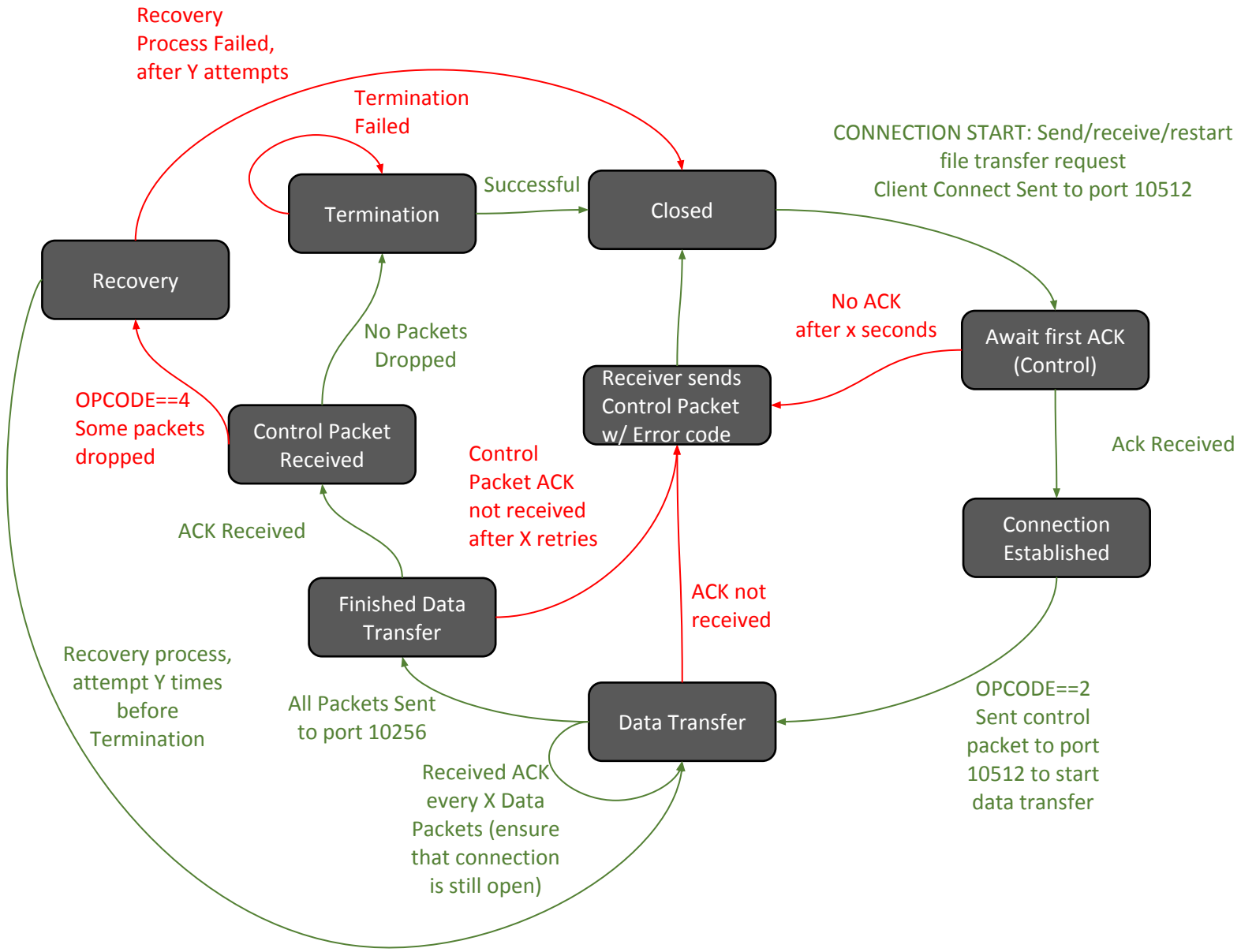
Opcode	Value	Extra data	Initiator	Meaning
Acknowledgement	0	Opcode of packet acking to	Either	Received packet, used in basic flow termination
Get file request	1	File path	Connection initiator	Request file to be sent over to initiator
Send file request	2	File path	Connection initiator	Request file to be sent from initiator
Finished sending	3	Retries	File sender	Current batch of packets sent
Bad packets	4	List of missing packets (sequence order; specific numbers or ranges)	File receiver	Not all packets received
Error	5	Error code	Either	Error in file transfer
Stop file sending	6		Either	Stop request
Restart request	7	List of missing packets (sequence order)		

Data Header Values

Header	Values	Meaning/Derivation/Definition
Packet Size	0-65535	Size of the Header Packet
Packet sequence index: 8 bytes	$0-[2^{64}-1]$	indexes each individual packet
Session/Transaction ID	0-65535	Unique identification for file transaction
Checksum	0-255	detects errors

Client Behavior

State	Description
Closed	Client is not sending any packets
Await first ACK (control)	Wait for the first ACK after sending control packets
Connection Established	Send starting control packet to start data transfer
Data Transfer	Make sure ACKs are received after every x data packets, if not assume connection is dead.
Finished Data Transfer	After Data Transfer, determine if ACK was received
Control Packet Received	If no packets dropped, proceed to termination, else proceed to recovery
Recovery	Proceed with recovery for dropped packets, then proceed to termination
Termination	FIN ACK and ACK responses.
Send Control Packet w/ Error Code	Close connection with ERROR OPCODE; sender pings receiver for error Control Packet, restart connection w/ proper restart opcode, handled via software.



Message Flow and State Diagram Questions

- How does the client initiate a connection? **Get/Send/Resend file request opcode.**
- Are you going to have authentication? **Nay**
- Are you going to have encryption? **Absolutely none**
- How will you confirm when a file has completed transfer? **Finish sending opcode**
- How will you ensure the integrity of the message in transit? **Checksums**
- How will you handle dropped packets? **After sender sends “finish sending” opcodes, receiver sends ack (on success) or sends “bad packets” opcode and lists sequence numbers of bad packets**
- Who initiates the closing of the connection? **Either with “stop file sending” opcode**
- Will you have error handling? **Basic — send error code and message in variable length**

Message Header and Value Questions

•How many header types do you need? Does it make sense to define one header and have an opcode to differentiate message types? Or does it make sense to have multiple header types? **Two header types.**

•How many bytes will the header(s) be? **Data header- fixed 20 bytes + DATA**

Control header- Fixed 11 bytes + Arbitrary Data

•Do you want the header to be a fixed size or do you want to enable variability? **We have variability in the packet size field**

•Do you want to make your header as small and efficient as possible, only catering to what is possible currently, or do you want to future-proof your header by making it larger than currently necessary? **Small headers, large variable for arbitrary data (control header)**

•Do you want to add options that allow you to introduce variability and future proof mechanisms?

Variable length extra data field and a whole byte for opcodes allows for expandability in control header.

Creative Thinking Process