Note from Amulya (Module 2 lead)

- While implementing the TritonHTTP Server, if you choose to use the bufio package to write responses to clients, keep in mind that you would need to call flush() explicitly after writing (and before closing the connection).
- The class demo servers used conn.Write() to write responses to the client and this way does not require any explicit flushing.

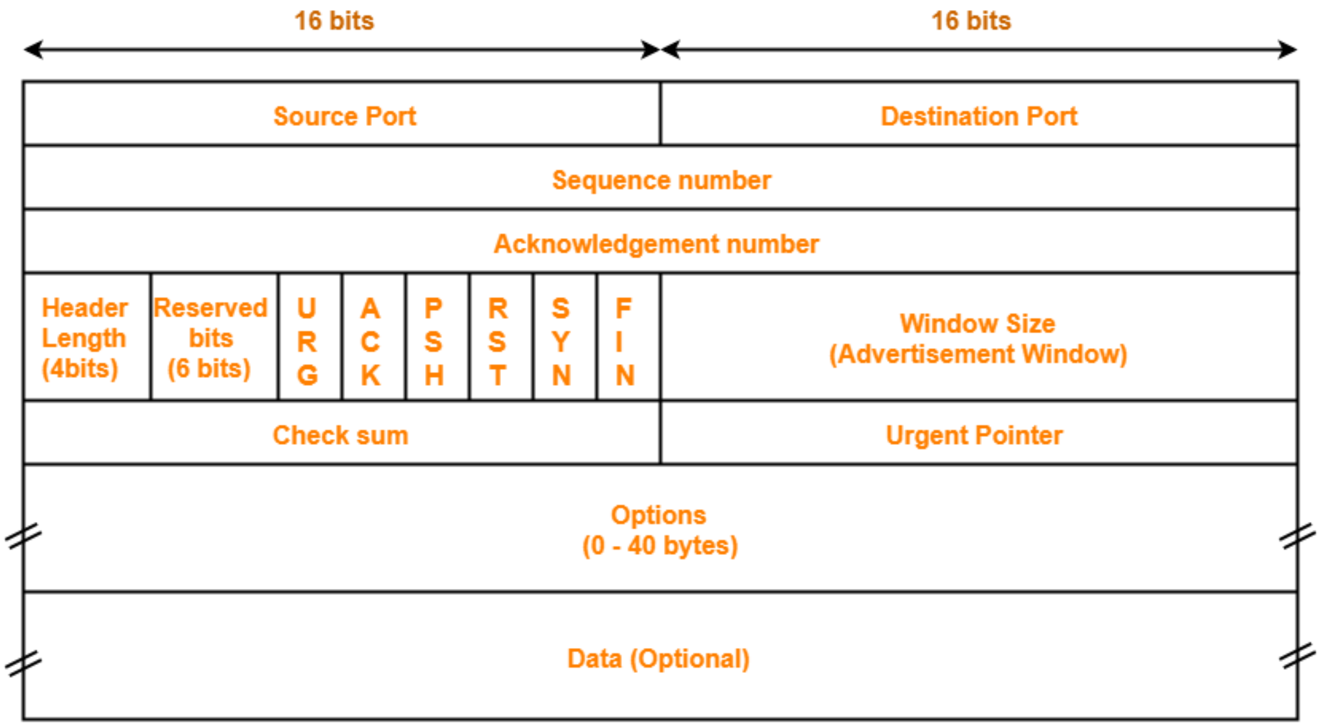
TCP and HTTP Timeline

Strategies for debugging

- When programming distributed systems, ***timing*** of an instruction is just as important as its ***correctness***
- Merely sending the right parameter is insufficient; if the server/client are not in a ***state*** where they can input the argument, the process will fail
- For instance, if you send a TCP connection close request ("FIN") to the client immediately after sending an HTTP 200, and the FIN arrives before the HTTP 200, the server may close the connection before the client has a chance to process the HTTP 200
  - The client may then throw a "connection closed by peer" error
- Note: in general we in the instructional staff have been instructed to assist at a very high level with debugging; the bulk of this project is to be completed independently

Which of these is not a property of the Transmission Control Protocol (TCP)?

- ~~Data arrives in the order in which it was transmitted~~
- ~~Data arrives reliably, even if the network is sometimes unreliable~~
- ~~Data arrives as an “infinite” stream of bytes~~
- **Data arrives as a sequence of fixed-sized messages**
- ~~Data arrives from only one sender~~


16 bits
16 bits
Source Port
Destination Port
Sequence number
Acknowledgement number
Header Length (4bits)
Reserved bits (6 bits)
URG
ACK
PSH
RST
SYN
FIN
Window Size (Advertisement Window)
Check sum
Urgent Pointer
Options (0 - 40 bytes)
Data (Optional)

TCP Header

Which of these is not a property of the Transmission Control Protocol (TCP)?

- Data arrives in the order in which it was transmitted
- Data arrives reliably, even if the network is sometimes unreliable
- Data arrives as an “infinite” stream of bytes
- **Data arrives as a sequence of fixed-sized messages**
- Data arrives from only one sender

- You and your partner are implementing a new protocol as part of a new machine learning system you're implementing. Your code needs to be able to read in a series of data updates, which are variable length, and separated by a carriage return (CR) character. You propose reading from the socket in 16KB chunks, appending data to a dynamic array as discussed during lecture. Your partner says that is too complicated, and instead argues for reading from the socket in 1-byte increments, and simply comparing the byte to the delimiter to detect the end of each update. Which approach is higher performance? Why?

- If a client does not get a response to an RPC request, what are three possible reasons why?

- Initially, RPCs were supposed to look and act “just like a local procedure”. What are two reasons why networked RPCs are fundamentally different than local procedures?

- RPC frameworks generally cannot support pointer arguments. Why is that? What would be a way to get around this limitation?