

Assignment 1 – Socket Programming and Concurrency



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Student Name: Morgan Lloyd

Student Number: 296463

Table of Contents

1. Transmission Control Protocol (TCP)	3
2. Maximum Transmission Unit & Fragmentation	9
3. Synchronous & Asynchronous.....	11
4. Interoperability	12

1. Transmission Control Protocol (TCP)

TCP is a connection-oriented protocol which ensures that before data is sent a connection between the server and client is made. To establish this connection a process called the '**Three-way handshake**' is used. The process involves sending TCP headers in packets between the client and server. [1 pg. 282-294] [2]

Figure 1 shows a TCP header. Each header has a sequence number and an acknowledgement number. The sequence number identifies each byte of data being sent as well as the order. The acknowledgement number confirms the data it has received and what it expects from the next transfer of data. In the header, there are flags which are located in the red boxes below. These represent the following:

SYN = Synchronise – Sent to establish the TCP connection.

ACK = Acknowledgment – Sent to acknowledge the successful receipt of packages.

FIN = Finalise – Sent to end the TCP connection.

[1 pg. 282-294] [2]

TCP header also contains:

- Source port – To specify the sender
- Destination port – The port number used to send to the desired destination.
- Window – Used to specify the number of data octets.
- Checksum
- Urgent pointer
- Options

[1 pg. 282-294] [2]

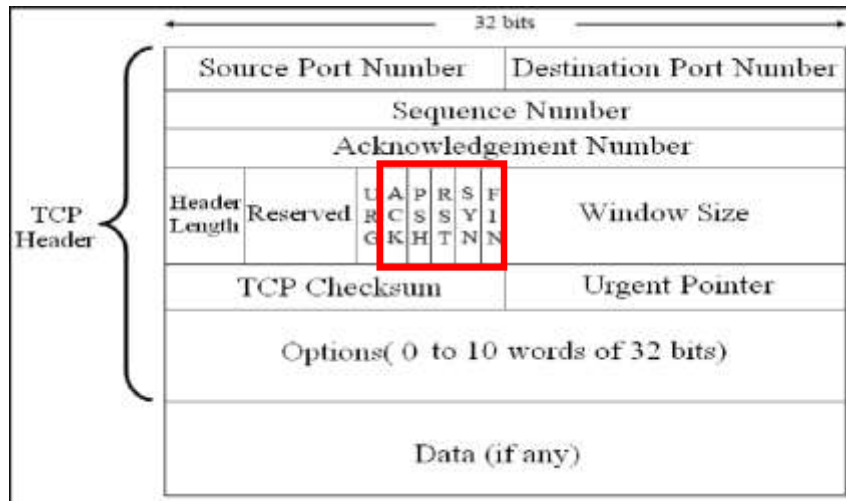


Figure 1 – TCP Header – [3]

Three-way handshake

Figure 2 shows the three-way handshake. This process establishes the connection between the client and server. First, the client sends a TCP header with the SYN flag set as well as a random initial sequence number. This is to establish a synchronised process of data exchange with the server. The server then acknowledges the client's sequence number by sending back an ACK flag and expects the next packet it receives to be $n+1$ of the last packet (501), this is the acknowledgement number. The server also sends the client an SYN flag with a random sequence number from which it will count. The client sends back an ACK telling the server it got the SYN and expects the next acknowledgement number to be 751. After this initial synchronisation, the client and server have established the TCP connection. [1 pg. 282-294] [2]

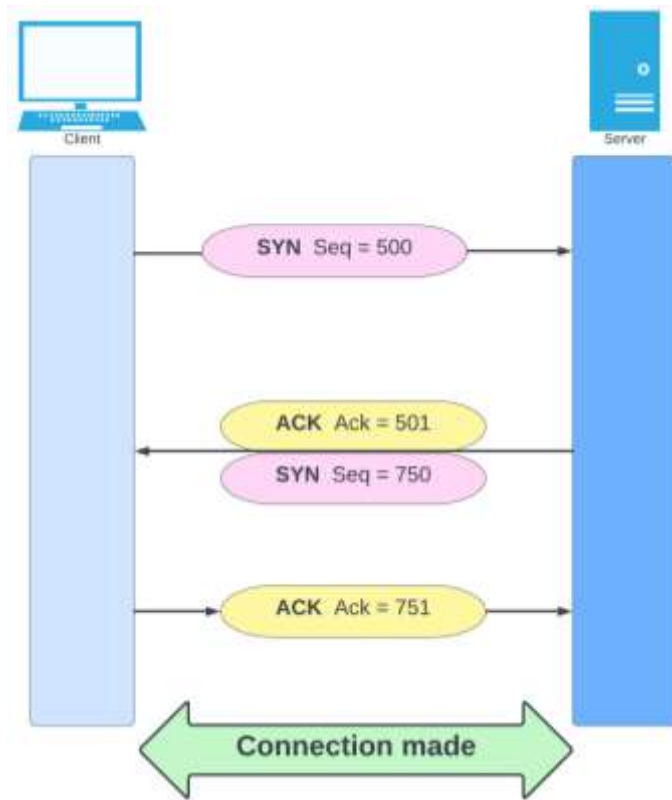


Figure 2 – Three-way handshake

Data transfer

Figure 3 shows the process after the three-way handshake called data transfer. The client sends the server a packet with the next sequence number it is expecting as well as some data in this request scenario. The server responds with an ACK and sends data to the client's request. The client then acknowledges this response and the server sends more data. This process repeats until either the server or client chooses to disconnect. [1 pg. 282-294] [2]

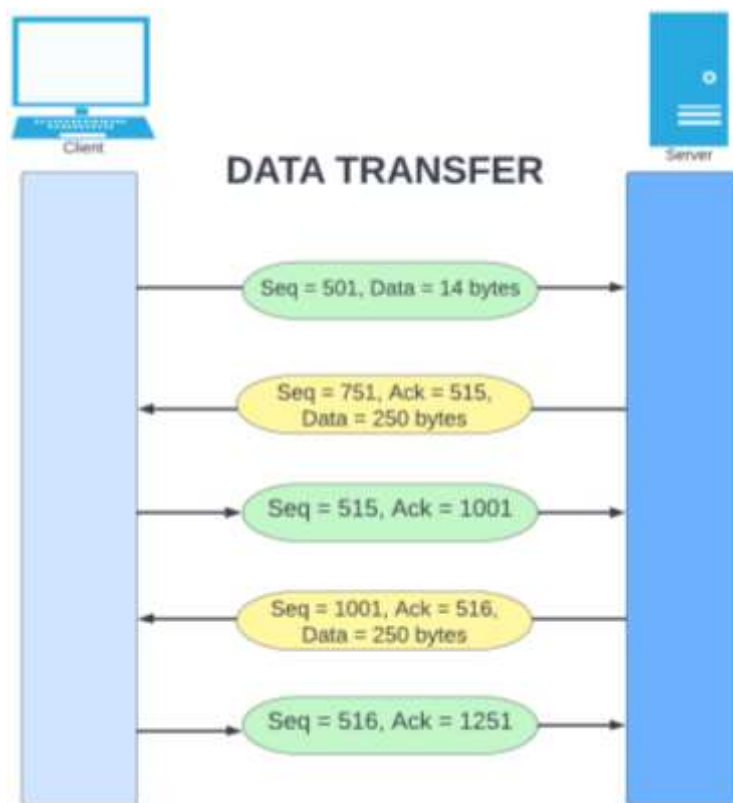


Figure 3 – Data transfer

Closing

The final process of TCP is the closing sequence as shown in *figure 4*. In this scenario, the server initialises the closing by sending a packet with the FIN flag set. The client sends back an ACK to acknowledge the receipt of the packet. The client then sends back a FIN to the server to close the process. Finally, the server acknowledges the client's FIN response. [1 pg. 282-294] [2]

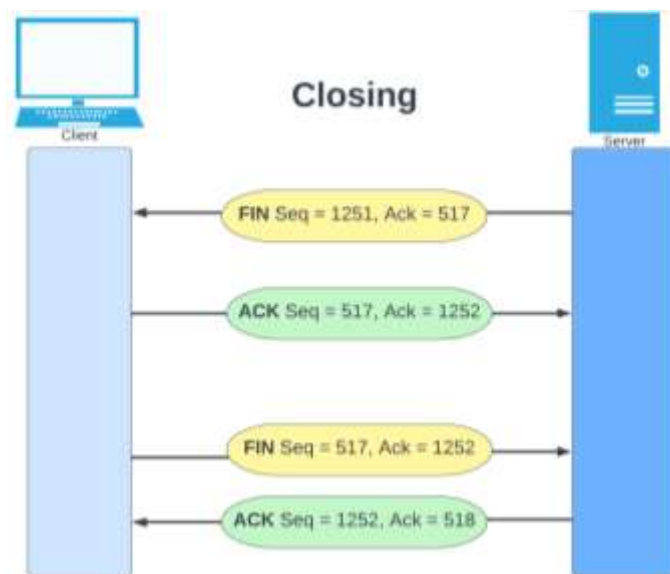


Figure 4 – Closing

TCP guaranteed delivery & Applications

TCP guarantees delivery due to the way it assigns a sequence number to each byte it transfers and an acknowledgement for receipt of the bytes. This is shown in *figure 5* below. Even if a byte is lost in the data transfer the acknowledgement number will tell the client/server it requires that particular byte on the next exchange ensuring all the data is received.

Additionally, the sequence numbers allow for correctly ordering the data if received out of sync.

TCP allows for flow control by using the 'window'. The window is an acceptable range of data determined by the receiver which is returned with each ACK to control the volume of data.

TCP transmission is reliable because of its use of sequencing, acknowledgements, flow control and the three-way handshake making it perfect for applications that require a guarantee of delivery such as messaging applications, online banking and web browsing. However, it falls down in applications that need fast delivery of data where reliability is not as important such as broadcasting, online gaming and streaming. It fails in these scenarios due to ensuring every byte is sent and received which takes much longer compared to the user datagram protocol (UDP). [1 pg. 282-294] [2] [4]

TCP is also used by protocols in the application layer including:

File Transfer Protocol – Used for transfer of data files between systems.

Hypertext transfer protocol – Used for transfer of web pages across systems.

Interactive mail access protocol – Used for email applications.

Post office protocol – Used for reading and removing emails on a server.

Secure shell (SSH)- Allows remote access to computers.[4]

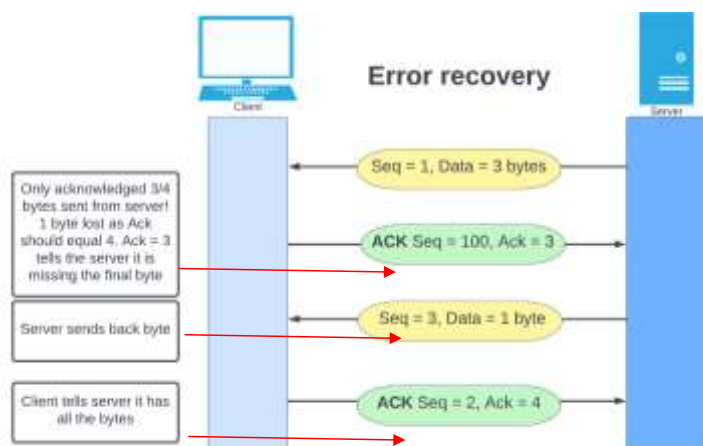


Figure 5 – TCP Error Recovery

2. Maximum Transmission Unit & Fragmentation

The maximum transmission unit determines the largest packet size that can be transmitted over a network link. Maximum transmission unit is measured in bytes and its size is typically determined by the type of network links and devices using a technique called path maximum transmission unit discovery or by determining the minimum size in the link manually.[5] For Ethernet, the maximum transmission unit is commonly 1500 bytes. *Figure 6* shows the maximum transmission unit.

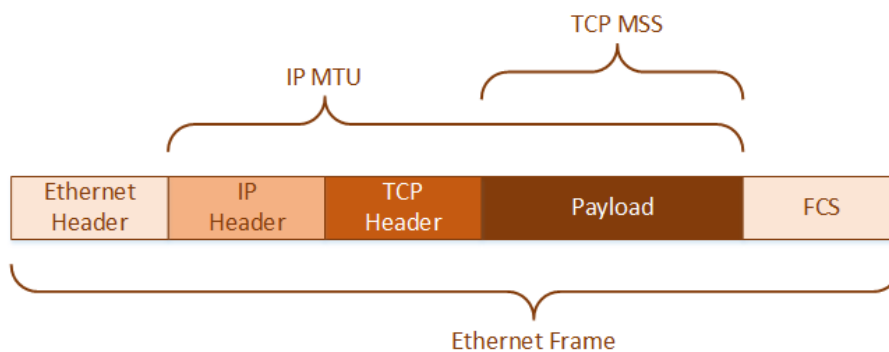
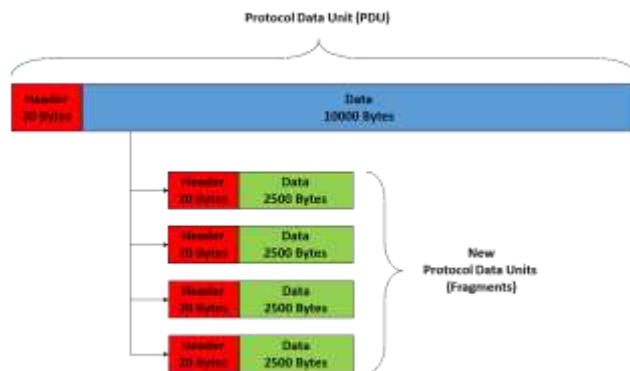


Figure 6 – MTU

Because the maximum transmission unit is concerned with packets which is the protocol data unit for the network layer of the OSI model it is important to see the effects of the protocol unit sizes in the other layers. Peer to peer communications in networking involves the processes of encapsulation and decapsulation of data to allow each layer of the system to understand the data in the correct format. Figure 7 shows this process against the OSI model as well as the protocol data unit for each layer in the table. [6]

OSI Model	PDU
Application	Data
Presentation	
Session	
Transport	Segment
Network	Packet
Data Link	Frame
Physical	Bits



[11]

3. Synchronous & Asynchronous

Client-server applications communication is based on requests and responses and it can either be synchronous or asynchronous. In asynchronous communication, a client will send a request to the server and then will perform another task while listening for the response back. In synchronous communication, the client will send a request to the server and will wait for the response without performing its next task.[12] As you can see from the request-response pattern the primary task of the server is to handle input and output (I/O) operations. POSIX defines synchronous and asynchronous I/O as:

- Synchronous I/O operations cause the requesting process to be blocked until that I/O operation completes.
- Asynchronous I/O operations do not cause the requesting process to be blocked. [13]

Blocking I/O

Blocking I/O model is the default for all sockets. In the blocking model, the system is stuck until the operation is complete. This model isn't ideal for handling multiple connections at the same time due to this behaviour. However, multi-threading can be used to segregate resources between client connections to enable multiple users.

Non-blocking I/O

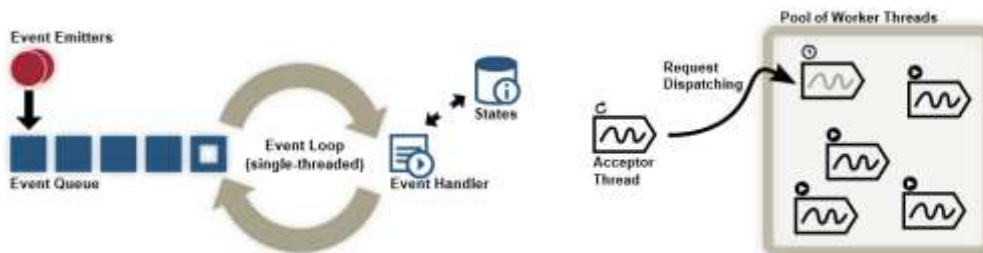
Non-blocking I/O model means an I/O request is queued straight away and the function is returned. The request then gets processed at a later point by the kernel. This functionality gives it an advantage over the blocking model as it will not block the process.

Asynchronous non-blocking I/O

Asynchronous non-blocking model returns from I/O calls straight away. After finishing it emits an event or executes a callback. This model does not block giving it an advantage over the blocking model. This model provides good performance according to Erb. [14]

Multi-Threaded vs Event-driven

Multi-threaded server architectures use a thread per client connection to enable the multiple connections without creating a deadlock. Multi-threaded servers use the blocking I/O operations and spawn a thread for each connection to prevent delays. Event-driven server architectures use an alternative to blocking I/O and thread per client connection used in the multi-threaded architecture. A single thread is used to handle multiple client connections. That single thread then manages all events emitting from I/O operations caused by connections. Below shows the event loop used in this event-driven architecture. The loop removes the events from the queue, processes the event and then moves on to the next one. The architecture does not use sequential operations it uses cascading asynchronous calls which can make fault finding and debugging difficult due to the **structure**. [14]



[14]

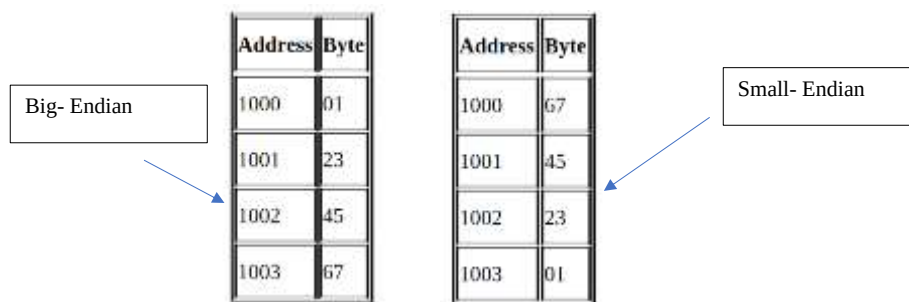
4. Interoperability

Interoperability is defined by the ISO/IEC 2382-01 as the “*capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units*”. [15]
Interoperability is the ability of computer systems to communicate with each other. To achieve interoperability systems need to be able to exchange, interpret and present data in a format that they agree upon.

To agree upon a medium of communication typically standard network protocols are used. A protocol is a set of rules for communication between two systems. Protocols that are used in the TCP/IP suite are commonly used for communication as they are open protocol standards available for anyone to develop network applications upon. The use of protocols found in the TCP/IP suite allows for interoperable communication between network applications as the data being sent is in an agreed-upon format defined by a protocol such as HTTP and FTP.

[16]

Interoperability is also achieved between different machines and operating systems through the TCP/IP suite through byte ordering. Some machines and operating systems use an order of bytes called 'little-endian' and some use 'big-endian', this is due to differences in architecture. TCP/IP uses a network byte order which is a big-endian representation. This means little-endian systems need to perform conversion to big-endian. Below shows the difference between these byte orders. [17]



References

1. Goralski W. The illustrated network : how TCP/IP works in a modern network. Amsterdam ; Boston: Elsevier/Morgan Kaufmann Publishers; 2009.
2. Doral P. An Introduction to Computer Networks Release 2.0.2 Peter L Dordal [Internet]. 2022 Feb [cited 2022 May 20] p. 385–91. Available from: <http://intronetworks.cs.luc.edu/current2/ComputerNetworks.pdf>
3. Masri A. Towards the distribution control of manufacturing systems : a component-based approach for taking into account the communication architecture in modeling [Internet]. 2009 [cited 2022 May 20]. Available from: https://www.researchgate.net/figure/TCP-Header-In-TCP-protocol-the-flags-field-has-in-important-role-in-the-connection-and_fig23_280681580
4. Kumar S, Rai S. Survey on Transport Layer Protocols: TCP & UDP. International Journal of Computer Applications [Internet]. 2012 May [cited 2022 May 20];46(7):20–5. Available from: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.734.7346&rep=rep1&type=pdf>
5. Determining the maximum transmission unit [Internet]. www.ibm.com. IBM; 2021 [cited 2022 May 20]. Available from: <https://www.ibm.com/docs/en/zos/2.3.0?topic=overview-determining-maximum-transmission-unit>

6. Fairhurst G. Encapsulation of PDUs [Internet]. Abdn.ac.uk. 2005 [cited 2022 May 20]. Available from: <https://erg.abdn.ac.uk/users/gorry/course/intro-pages/encapsulation.html>
7. ComputerNetworkNotes. Data Encapsulation and De-encapsulation Explained [Internet]. Available from: <https://www.computernetworkingnotes.com/ccna-study-guide/data-encapsulation-and-de-encapsulation-explained.html>
8. D G. Network Ports, Protocols, and Topologies [Internet]. [cited 2022 May 20]. Available from: <https://www.oreilly.com/library/view/comptia-network-certification/9781789340501/b3b52c5c-068e-4afb-a1ad-1492056f5c27.xhtml>
9. Alcock S, Nelson R. An Analysis of TCP Maximum Segment Sizes [Internet]. 2010 [cited 2022 May 20]. Available from: https://wand.net.nz/sites/default/files/mss_ict11.pdf
10. Wright G. What is the maximum transmission unit (MTU)? [Internet]. SearchNetworking. [cited 2022 May 20]. Available from: <https://www.techtarget.com/searchnetworking/definition/maximum-transmission-unit>
11. [cited 2022 May 20]. Available from: https://upload.wikimedia.org/wikipedia/commons/c/cd/PDU_Fragmentation-en.png
12. Lim M. Directly and Indirectly Synchronous Communication Mechanisms for Client-Server Systems Using Event-Based Asynchronous Communication Framework. IEEE Access [Internet]. 2019 [cited 2022 May 20];7:81969–82. Available from: <https://ieeexplore.ieee.org/document/8744225>
13. W Richard Stevens, Fenner B, Rudoff AM. UNIX network programming. Volume 1, The sockets networking API. 3rd ed. Vol. 1. Boston, Ma: Addison-Wesley; 2003.
14. Erb B. Concurrent Programming for Scalable Web Architectures [Internet]. [Ulm University]; 2012 [cited 2022 May 20]. p. 47–53. Available from: https://www.researchgate.net/profile/Benjamin-Erb/publication/236149101_Concurrent_Programming_for_Scalable_Web_Architectures/links/54edce0f0cf24a16e6080167/Concurrent-Programming-for-Scalable-Web-Architectures.pdf?origin=publication_detail
15. IEC I. ISO/IEC JTC1 SC36 Information Technology for Learning, Education and

Training [Internet]. web.archive.org. 2003 [cited 2022 May 20]. Available from:
<https://web.archive.org/web/20071129034634/http://old.jtc1sc36.org/doc/36N0646.pdf>

16. What is Interoperability? Definition and FAQs | OmniSci [Internet]. www.heavy.ai. [cited 2022 May 20]. Available from: <https://www.heavy.ai/technical-glossary/interoperability>

17. Reese RM. Learning network programming with Java : harness the hidden power of Java to build network-enabled applications with lower network traffic and faster processes [Internet]. Birmingham: Packt Pub; 2015 [cited 2022 May 20]. Available from:
<http://auhd.edu.ye/upfiles/elibrary/Azal2020-01-22-12-38-44-15941.pdf>