

Ryan_Morris_1550957_CW1

by Ryan Morris

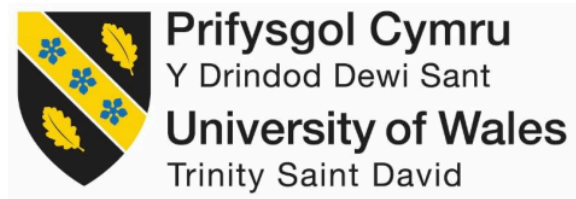
Submission date: 13-May-2025 12:12PM (UTC+0100)

Submission ID: 259047352

File name: 41185_Ryan_Morris_Ryan_Morris_1550957_CW1_664976_406632020.docx (1.15M)

Word count: 3240

Character count: 18679



Socket Programming and Concurrency
Course Work 1
ACCA6028

Ryan Morris
1550957
1550957@student.uwtsd.ac.uk

15th May 2025

Table of Contents

INTRODUCTION	2
TCP AND GUARANTEED DELIVERY	3
HOW THE TCP OPERATES.....	3
<i>The Two Generals Problem</i>	5
MTU AND DATA TRANSFER	6
<i>MTU and IP Fragmentation</i>	7
SYNCHRONOUS VS ASYNCHRONOUS PROTOCOLS	8
<i>Understanding Synchronous and Asynchronous Communication</i>	8
<i>Network Protocols and Synchronicity</i>	8
<i>I/O Dispatch Models</i>	9
INTEROPERABILITY	9
CONCLUSION	10
BIBLIOGRAPHY	10

Introduction

The need to develop reliable, efficient and interoperable networked applications is essential in the current era of computing. Understanding the underlying protocols and technologies used when creating anything from messaging apps to online multiplayer games is essential to modern day programmers, as understanding protocols and technologies used to govern communication between devices will inevitably lead to more stable and efficient applications. Socket programming and concurrency can be viewed as the backbone of these systems, allowing processes to interact across networks in both synchronous and asynchronous contexts. This assignment report will look at several core concepts within this domain, for each section there will be an emphasis on their practical implications for programmers and software development.

The first section will examine the Transmission Control Protocol (TCP), in particular, how the protocol guarantees the delivery of data and the different mechanisms it uses to achieve this, such as the three-way handshake, acknowledgements and retransmission strategies. The Two Generals Problem, which is essentially a conceptual model for understanding the limitations of absolute certainty, will also be analysed.

Following on from TCP, the impact of Maximum Transmission Unit (MTU) size on the efficiency of data transfer, looking at its place in the OSI model and also the challenges of IP fragmentation.

Synchronicity and asynchronicity in networking will then be examined, including looking at the suitability of each approach when used for different types of applications. Dispatch models

Finally interoperability will be researched to find the importance of creating interoperable applications and systems, and ways that programmers can endeavour to ensure the products they create are as interoperable as possible.

TCP and Guaranteed Delivery

700 words

The Transmission Control Protocol (TCP) is a protocol which along with the Internet Protocol (IP) form the foundation of internet communication, although we often see TCP/IP, the focus here will be solely on the TCP. TCP operates on the transport layer of the OSI model, whereas IP operates on the network layer to actually deliver packets.

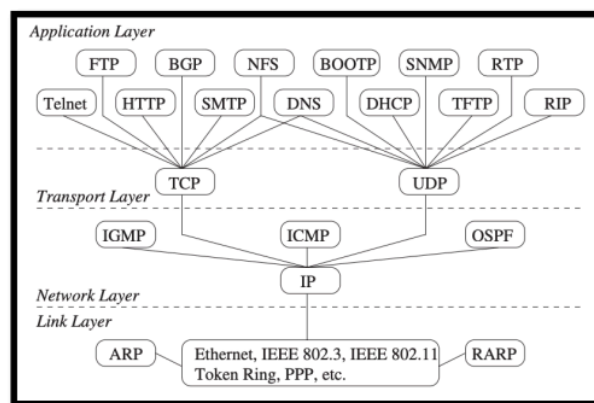


Figure 1 Shows relationship between protocols, including TCP/IP [1]

How the TCP Operates

TCP provides a connection-oriented and reliable service which connects two endpoints. It achieves this reliability through providing ordered, error-checked delivery of a stream of bytes. TCP incorporates mechanisms like sequence numbering, acknowledgements and retransmissions which ensure data is transferred fully and in order [1]. Prior to transferring any data, TCP establishes a connection between sender and receiver, to do this TCP utilises the three-way handshake, which also contributes to its reliability and confirms that both parties are ready to communicate.

The Three-Way Handshake

The three-way handshake is the secure method which TCP uses to establish a reliable connection for two end point to communicate with each other, such as a client and server. As we can see in Figure 2, the client initially sends a SYN (synchronise) packet, which contains a sequence number (x). The sequence number is essentially a reference point which will let the client and server track data which is to come. The server then responds with a SYN ACK (acknowledgement) packet, which both acknowledges the clients SYN by confirming its sequence number plus one ($x + 1$), and sends a SYN packet with its own sequence number (y). The client then finally responds with an ACK packet, which is the servers sequence number plus one ($y + 1$), at this point the connection is established [2] [3].

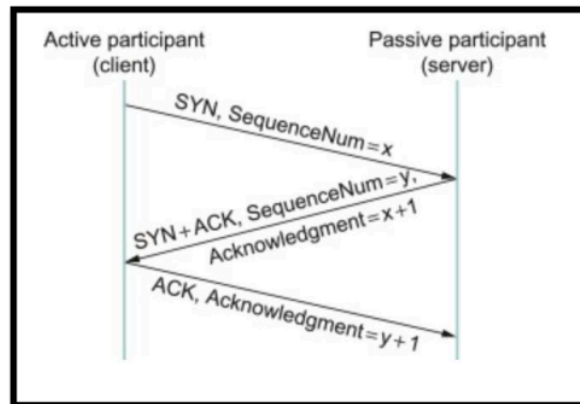


Figure 2 Three-Way handshake Sequence [4]

Once this is established, data can begin to be sent, to ensure reliability the synchronise/acknowledgement method is used to track the number of bytes to be sent [3]. Both client and server will now use sequence numbers and acknowledgements which were established during the handshake to track the information being transferred, to ensure completeness. Figure 3 below shows that after the connection is established, the server sends a "Hello Client" PSH (push) message, which includes a sequence number and length, the length tells the client how much data has been sent. The client then sends an ACK, confirming the data has been received in full. Each new piece of data sent increases the SEQ (sequence number) by the value of LEN, which is equivalent to the number of bytes sent. As seen in figure 3, a SEQ of 1 when combined with a LEN of 12, returns an ACK of 13, this confirms all the data which was sent has been received. This method ensures that no data is missing, and if data is missing, that exact data can be resent. Once the data has been transferred, the server sends a FIN (finished) message, which the client acknowledges before sending a FIN back. Once the server acknowledges the clients FIN, an ACK is sent, and the session is terminated.

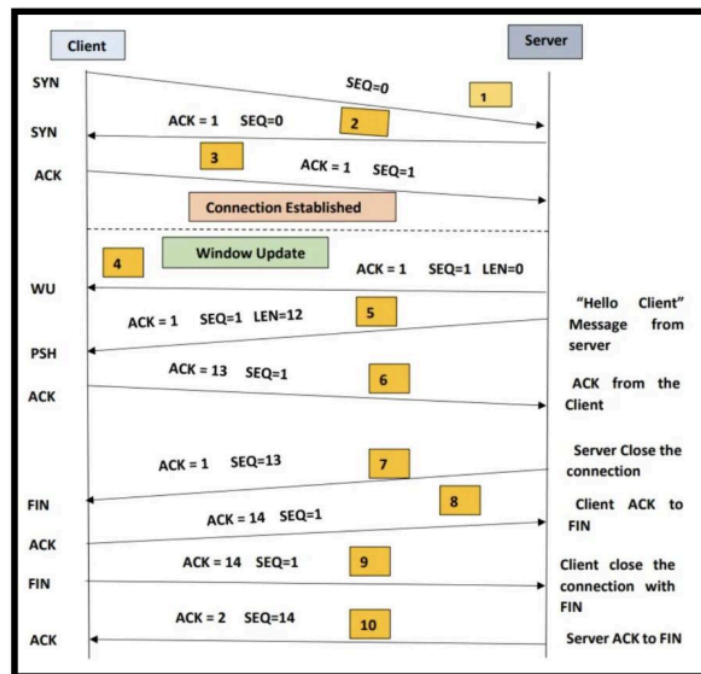


Figure 3 Three-Way Handshake and Data Transfer [2]

For programmers it's essential to have a basic understanding of how TCP works, programmers will be able to troubleshoot and rectify any errors, as well as build more reliable applications from the start [5].

The Two Generals Problem

To understand why the somewhat complicated three-way handshake is necessary, we can look at the Two Generals problem for a real-world example. The Two Generals problem is a fictional computer science scenario which, although is impossible to fully solve, perfectly demonstrates why the three-way handshake is necessary.

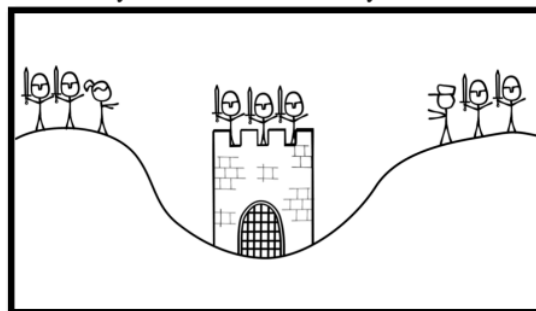


Figure 4 - Two Generals Problem [6]

In the scenario, two generals are trying to co-ordinate an attack, but they must agree on a time, as if only one attacks, their army will be defeated. The two generals are either side of a fortress and can't communicate directly with one another due to the distance and danger between them. To arrange the attack and decide upon a time, General A sends a messenger through the battlefield with a note to General B - "Attack at 10am". Herein lies the problem, General A doesn't know if General B received the message. The note is amended to "Attack at 10am, reply to confirm.". If General A receives a response, he knows General B knows the plan, but how does General B know that General A received the acknowledgement? He doesn't, and this loop could go on indefinitely.

In terms of TCP, the three-way handshake is the practical answer to this issue, once both sides send, receive and acknowledge a message, TCP deems the connection reliable enough for internet communication.

MTU and Data Transfer

MTU, or Maximum Transmission Unit, is the largest size of data or packet which can be sent over a network. The data link layer of the OSI model enforces MTU, and the MTU can vary depending on what technology is being used. For example, Ethernet usually has an MTU of 1500 bytes, if a packet is larger, it may be rejected or split up which can affect performance [7].

To understand how MTUs work in the process of transferring data, we need to look at where MTUs fit in the OSI model and how the Protocol Data Unit (PDU) handles data.

Layer	OSI Model	PDU
7	Application	Data
6	Presentation	
5	Session	
4	Transport	Segment
3	Network	Packet
2	Data Link	Frame
1	Physical	Bits

Figure 5 PDU at Each Layer [8]

Each layer of the OSI model passes data down in PDU format, depending on what level of the model the data is at, PDU is referred to differently. Figure 5 shows that at the application layer, it's called data, at the transport layer, it's called a segment, at the network layer, a

packet, at the data link layer, a frame and at the physical layer bits [9]. MTUs apply to frames at the data link layer and so controls what size a frame can be while on the physical network.

As programmers, MTUs are important to know about as any data which is being sent, such as a picture or a message, often need to be broken down as it moves through different layers of the OSI model. If MTU isn't considered, data being sent can exceed the MTU at the data link layer, if this happens the data could be dropped, or the data may be fragmented. Being aware of MTU allows programmers to create more efficient and reliable networked applications.

MTU and IP Fragmentation

When data is sent across a network, it needs to fit within the MTU size which is set by the data link layer. If the data is larger than the MTU limit, the network layer can break the packet into smaller pieces, in a process called IP fragmentation. Once the data has been fragmented, it can be sent across the network in smaller pieces and reassembled at the destination. If any of the fragments get lost in transit or delayed, the whole packet is considered lost and it may need to be resent, this can affect performance and reliability [10].

Fragmentation can happen at any of the routers a packet passes on its journey to the destination, if it reaches a router which has a smaller MTU size than what was set by the original router. Some drawbacks to fragmentation include increased overhead as fragmentation adds headers to each new segment and increase processing time at routers, with multiple segments, there's also an increased chance of one or more of the fragments becoming lost which requires retransmission of the entire original packet [10].

Original IP Datagram					
Sequence	Identifier	Total Length	DF May / Don't	MF Last / More	Fragment Offset
0	345	5140	0	0	0

IP Fragments (Ethernet)					
Sequence	Identifier	Total Length	DF May / Don't	MF Last / More	Fragment Offset
0-0	345	1500	0	1	0
0-1	345	1500	0	1	185
0-2	345	1500	0	1	370
0-3	345	700	0	0	555

Figure 6 Example of Fragmentation from Cisco [10]

Due to these drawbacks, a process called Path MTU Discovery (PMTUD) is often used, PMTUD tries to discover the smallest MTU along the path a packet will take and adjust the packet size accordingly to avoid fragmentation. If a packet which is too large reaches a router and has its Do Not Fragment (DF) flag set, the router will drop the packet and an ICMP (Internet Control Message Protocol) will send back a "Fragmentation Needed" message, which will prompt the sender to reduce the packet size and try again.

This isn't without its flaws, as PMTUD isn't always reliable, and some routers on the path could block ICMP messages from reaching the sender, this results in multiple packet drops with no feedback to the sender, causing failed or delayed connections [10].

Programmers can avoid IP fragmentation by being aware of and implementing a Maximum Segment Size (MSS) during TCP set up. An MSS defines how much data can be sent in a single packet. By implementing MSS, developers can ensure that data packets remain under the MTU and avoid fragmentation altogether.

Synchronous vs Asynchronous Protocols

Understanding Synchronous and Asynchronous Communication

Synchronous and asynchronous communication are the two main ways networked systems exchange data, they both have different use cases and methods of transmitting data [11]. Synchronous communication requires the sender to wait for a response from the receiver, this essentially blocks the sender, ensuring it stays idle until the data has been exchanged. With synchronous data transmission, everything happens in order, and the previous operation must finish before the next begins. Asynchronous data exchange on the other hand allows the sender to complete other tasks while the data is being processed by the receiver, making asynchronous non-blocking, this is the key difference between the two methods [11]. Asynchronous data exchange doesn't necessarily happen in order, similar operations can run concurrently, and we don't have to wait for tasks to finish before starting another [12].

A real-world analogy to simplify the difference between synchronous and asynchronous could be, if you are on a phone call, you begin the connection, exchange information, and then hang up, this could be compared to a synchronous exchange of data. If you are using MS Teams or texting, you can send a message, go about your day and receive a message back once the recipient of your message responds, this could be viewed as asynchronous.

Below is a basic table showing some general synchronous and asynchronous applications.

Synchronous	Asynchronous
Phone Call	Instant Messenger (text, Teams, WhatsApp etc)
Video Conference	Email
Telnet/SSH	Social media

Telnet is an old remote CLI controller, it has multiple safety concerns and has been widely replaced by SSH, which is also synchronous. Social media is under asynchronous as if we were to load Facebook for example, if every time we came across a larger piece of data, such as a longer video were to be synchronous, the whole UI would freeze while waiting for the entire video to load, it would destroy the usability of the site [13].

Network Protocols and Synchronicity

Various network protocols also follow either synchronous or asynchronous architecture. As mentioned SSH and Telnet are synchronous, when a command is sent, the client will wait for a response before continuing. HTTP is also synchronous, it only processes one request at a time, although other technologies can be used such as AJAX and promises in JavaScript which can do some asynchronous tasks on top of HTTP [14] [15]. The Simple Mail Transfer Protocol (SMTP) used for sending emails is asynchronous which allows us to send emails without immediately, or ever, receiving a response.

I/O Dispatch Models

In low level I/O dispatch models, synchronous I/O means that the program waits for every read and write operation to complete before continuing, in what Microsoft refer to as a wait-state [16]. While asynchronous I/O (known as AIO) allow operations like read and write to be called and run in the background on a thread and continue processing without waiting for a response, the kernel will notify the thread once the process is complete [16] [17].

Windows has ReadFile and WriteFile calls which can be either synchronous or asynchronous, depending on use case, when CreateFile is called, a 'FILE_FLAG_OVERLAPPED' flag is used to declare if the associated calls will be synchronous or asynchronous. This is where developer knowledge is required, as if the program needs to continue running without waiting for the thread to become free, the developer needs to know to use asynchronous I/O [16]. IBM refers to aio_read and aio_write which are asynchronous I/O relating to the Asynchronous I/O interface [17].

Understanding when to implement synchronous vs asynchronous dispatch models is a critical skill for a creating scalable networked applications, with IBM stating that AIO allows for overlapping processing and I/O which in turn improves throughput [17].

Interoperability

Interoperability when talking about software development and networking refers to the ability of different systems or applications to communicate and exchange data, even if they are built by different developers or operate on different operating systems, they need to work together [18]. A good example of interoperability would be iPhone running iOS communicating to a Linux server using the HTTP protocol. If these devices only had system specific protocols, they wouldn't be able to communicate outside of their ecosystem. To be interoperable, devices and systems need to ensure they are compatible, this can be achieved through using standard protocols, formats and standards [19].

Today's global reliance on interconnected devices means almost all applications, programs or devices need to be interoperable to connect with devices outside their ecosystem. If we take a generic web application, it will typically need to accept payments through a payment gateway, access external API's to fetch relevant data and even be accessible by crawlers to be indexed and appear on search engines such as Google [20]. If modern applications and systems weren't interoperable, communication and data exchange would be very different, with potentially large monopoly's appearing which could cause price increases, and a lack of innovation.

The format of information and data can increase interoperability, many systems and APIs will return data in JSON or XML format, these are widely known and interpretable file types, allowing applications or systems using a variety of platforms and programming languages to access the services. This is beneficial for developers, as it means that as long as these file types stay consistent, legacy applications will work and won't necessarily depreciate when technology advances.

If developers ensure they use widely accepted protocols, such as HTTP, TCP/IP, HTML and formats like XML and JSON, they can create interoperable applications and systems which will be able to communicate and exchange data, despite what operating system or programming language was used to create or host the application.

Conclusion

Throughout this report I have looked at various concepts which enable programmers to build reliable and efficient networked applications and systems, These included TCP – including the three-way handshake and sequence numbers and how they ensure reliability; MTU and IP fragmentation and their effect on data transfer and packet loss; the key differences between synchronous and asynchronous protocol and where each could be used, including how a synchronous application could lead to poor UX if used in the wrong scenario; And finally the importance of interoperability and the importance of using interoperable protocols and common data formats to ensure different platforms can work together.

If all of the areas covered were to be understood and applied correctly within a networked application, programmers can build networked applications which are reliable, efficient and scalable whilst also being easier to maintain and integrate with other platforms. With the correct level of understanding, many of the issues in networking can be anticipated, diagnosed and resolved effectively.

Bibliography

- [1] S. M. J. R. Y. L. S. Panwar, TCP/IP Essentials - A Lab Based Approach, Cambridge University Press, 2012.
- [2] S. Fernando, "TCP 3-Way Handshake," 06 2022. [Online]. Available: <https://medium.com/@wshehaniif/tcp-3-way-handshake-397d57baa8ee>. [Accessed 20 04 2025].
- [3] IBM, "Transmission Control Protocol (TCP)," [Online]. Available: <https://www.ibm.com/docs/en/zos-basic-skills?topic=4-transmission-control-protocol-tcp>. [Accessed 29 04 2025].
- [4] Science Direct, "Three-Way Handshake," [Online]. Available: <https://www.sciencedirect.com/topics/computer-science/three-way-handshake>. [Accessed 30 04 2025].
- [5] Checkly, "Understanding TCP for Application Developers," Checkly, [Online]. Available: <https://www.checklyhq.com/learn/monitoring/what-is-tcp/#:~:text=While%20TCP%20is%20often%20considered,and%20build%20more%20reliable%20applications..>
- [6] S. A. Brown, "The Two Generals Problem," 17 05 2024. [Online]. Available: <https://linuxblog.io/the-two-generals-problem/>.

- [7] G. Wright, "maximum transmission unit (MTU)," [Online]. Available: <https://www.techtarget.com/searchnetworking/definition/maximum-transmission-unit>. [Accessed 23 04 2025].
- [8] P. Singh, "OSI Model and TCP/IP Model," 02 10 2022. [Online]. Available: <https://prabhjeetlearning.medium.com/osi-model-and-tcp-ip-model-93f04f91ab8>. [Accessed 20 04 2025].
- [9] Geeks for Geeks, "Protocol Data Unit (PDU)," Geeks for Geeks, 29 04 2022. [Online]. Available: <https://www.geeksforgeeks.org/protocol-data-unit-pdu/>. [Accessed 29 04 2025].
- [10] Cisco, "Resolve IPv4 Fragmentation, MTU, MSS, and PMTUD Issues with GRE and IPsec," Cisco, 17 05 2023. [Online]. Available: <https://www.cisco.com/c/en/us/support/docs/ip/generic-routing-encapsulation-gre/25885-pmtud-ipfrag.html#toc-hId--1412502279>.
- [11] G. Lau, "Synchronous And Asynchronous Data Transmission: The Differences And How to Use Them," IEEE Computer Society , 01 10 2023. [Online]. Available: <https://www.computer.org/publications/tech-news/trends/synchronous-asynchronous-data-transmission>.
- [12] D. Bevans, "Asynchronous vs. Synchronous Programming: Key Similarities and Differences," Mendix, 14 10 2024. [Online]. Available: <https://www.mendix.com/blog/asynchronous-vs-synchronous-programming/>.
- [13] iAmDev, "Synchronous vs asynchronous in programming," YouTube, 12 05 2021. [Online]. Available: https://www.youtube.com/watch?v=poSVn_UP_vI.
- [14] "Is HTTP synchronous or asynchronous?," Quora, [Online]. Available: <https://www.quora.com/Is-HTTP-synchronous-or-asynchronous>.
- [15] "HTTP Synchronous nature," 2019. [Online]. Available: <https://stackoverflow.com/questions/52252800/http-synchronous-nature>.
- [16] Microsoft, "Synchronous and Asynchronous I/O," Microsoft, 20 03 2021. [Online]. Available: <https://learn.microsoft.com/en-us/windows/win32/fileio/synchronous-and-asynchronous-i-o>.
- [17] IBM, "Asynchronous I/O Subsystem," IBM, 24 03 2023. [Online]. Available: <https://www.ibm.com/docs/en/aix/7.1.0?topic=concepts-asynchronous-io-subsystem>.
- [18] Geek for Geeks, "Interoperability in Distributed Systems," Geek for Geeks, 09 09 2024. [Online]. Available: <https://www.geeksforgeeks.org/interoperability-in-distributed-systems/>.
- [19] M. Lindquist, "What Is Interoperability?," Oracle, 20 05 2024. [Online]. Available: <https://www.oracle.com/uk/interoperability/>.
- [20] Google Search Central, "Search Engine Optimization (SEO) Starter Guide," Google, [Online]. Available: <https://developers.google.com/search/docs/fundamentals/seo-starter-guide>.
- [21] ByteBite, "What is MTU, really? - Maximum Transmission Unit Explained - CCNA & CompTIA Network+," YouTube, 2025. [Online]. Available: <https://www.youtube.com/watch?v=xGK3EUAX8Xk>.
- [22] CloudFlare, "What is MTU (maximum transmission unit)," CloudFlare, [Online]. Available: <https://www.cloudflare.com/en-gb/learning/network-layer/what-is-mtu/>. [Accessed 08 05 2025].

FINAL GRADE

62/100

GENERAL COMMENTS

A fairly good report though use of sources needs a little work. You have used a very good number of sources, though a lot of these are non-authoritative web resources, including youtube and stackoverflow.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11