

Part 1

by Michael Simmonds

Submission date: 01-May-2023 07:15PM (UTC+0100)

Submission ID: 204152326

File name: 18297_Michael_Simmonds_Part_1_499749_2068222156.docx (235.81K)

Word count: 4003

Character count: 23939

Assessment 1 Part 1

Socket Programming and Concurrency

1908090

Michael Simmonds

Contents

Contents.....	1
Introduction	2
Transmission Control Protocol (TCP)	3
How does TCP work?	3
Two general problem	5
Maximum Transmission Unit (MTU).....	6
Synchronicity & Asynchronicity	8
Interoperability	10
Conclusion.....	11
Bibliography	12

Introduction

With most of daily life now revolving around the internet and networks the need for reliable data transmission has increased. The Transmission Control Protocol (TCP) is one of the key protocols used to allow for this by supplying end to end connection between devices/applications.

Within this research driven assignment TCP is researched, specifically how it operates and how it almost always guarantees delivery of data. The assignment will look at both how the three-way handshake works along with how the two general problems could have an affect on how data is transmitted.

The assignment will consider the impact of the maximum transmission unit (MTU) size on data transfer, taking into account factors such as the size of the protocol data unit (PDU), their relative positions in the OSI model, and the practicalities of IP fragmentation.

Synchronicity and asynchronicity in network protocols will also be explored look at what they mean and what sort of applications use each. The assignment will also consider I/O dispatch models and their implications for networked applications. Finally, we will explore the meaning of interoperability for networked applications and how it can be best achieved.

As programmers, understanding the technical details of TCP, MTU, synchronicity, asynchronicity, and interoperability is crucial for the design and development of networked applications.

Transmission Control Protocol (TCP)

With the rise of technology, the way data is exchanged has become more vital. Transmission Control Protocol (TCP) is a standard that defines how to establish and maintain a network conversation by which applications can exchange data. [1] TCP is widely used and is known for its reliability of delivering data over the internet. TCP establishes a connection between two endpoints and guarantees delivery of data even if there is packet loss or congestion. It provides reliable, ordered, and error-checked delivery of data between two endpoints.

How does TCP work?

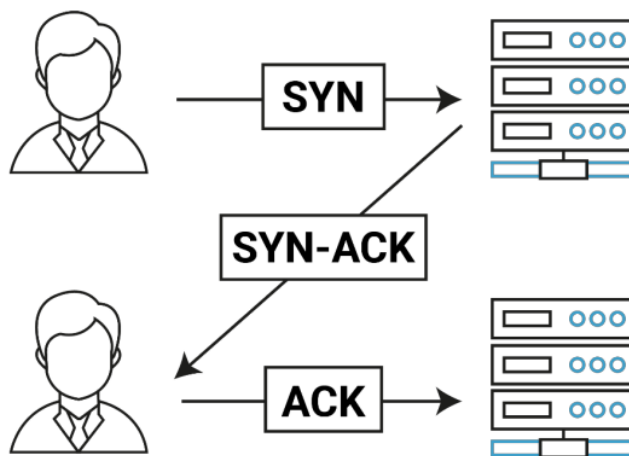
TCP allows a user or application send messages to other users or applications, the protocol splits these messages into small chunks called packets, TCP also ensures that the connection between two clients is stable enough to exchange messages. [2]

Before data is transmitted there needs to be a connection created between the sender and receiver, to do this TCP uses the "Three-way handshake".

The three-way handshake is a three-step process that requires both the client and server to exchange synchronization and acknowledgment packets before the real data communication process starts. [3]

There are three types of TCP messages and all three are used within the three-way handshake when establishing a connection:

1. **Syn (Synchronise)** – This message is used to create a connection between two application/devices, it also sends the receiving device a packet which contains a sequence number which allows the server to know how many packets to expect.
2. **Syn-Ack (Synchronization and Acknowledgement)** – When the receiving application/device receives the Syn packet, it sends a Syn-Ack packet back, this packet contains the same sequence number as the packet it received along with its own sequence number.
3. **Ack (Acknowledge)** – Once an application/device receives a Syn, Ack is used to send a response to advise the other application/device that it received it.



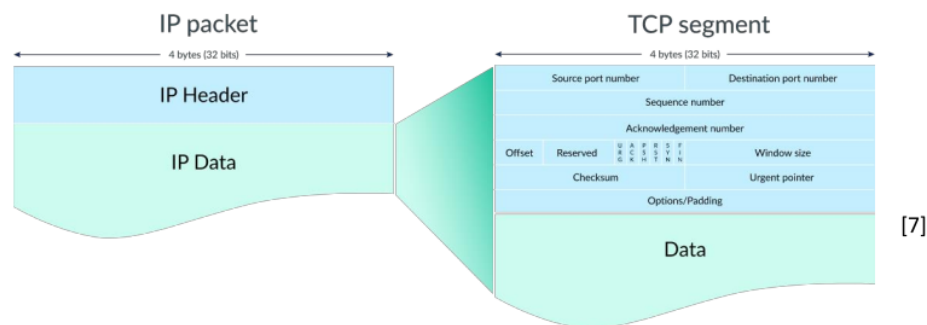
[4]

Within the connection establishment phase, applications can announce their Maximum Segment Size (MSS), which defines the largest TCP segment they will exchange. [5]

Once the three-way handshake is completed there is a stable connection and data transmission can commence.

Once a connection is made, TCP segments the data due to be transmitted into packets to be transmitted. Each TCP packet is given a header, this header contains several information fields, these fields include:

1. Source port (16 bits)
2. Destination port (16 bits)
3. Sequence Number (32 bits)
4. Acknowledgment number (32 bits)
5. Data offset (4 bits)
6. Reserved (6 bits)
7. Window (16 bits)
8. Checksum (16 bits)
9. Urgent pointer (16 bits)
10. Options (variable) [6]



Each device/application is different TCP uses flow control, flow control manages the flow of data between the device/application and makes sure that the flow of data can be received at a pace capable by both. If one device/application is transmitting data to quickly for the other TCP can slow it down. TCP uses a sliding window algorithm for flow control. [8]

TCP can also combat network congestion using Congestion Control, too much traffic on a network can lead to packet loss or loss of performance, TCP can detect this increase of traffic and can respond by reducing the transmission rate of data.

Errors can occur by TCP uses a number of techniques to detect and correct these errors. TCP's main error checking technique is the use of checksums, as you can see above in the makeup of a TCP packet, each packet contains a checksum. The receiving side calculates the checksum on the data that is received using the same algorithm as the sender and compares its value to the checksum passed in the header. If the values do not match, the packet is rejected. [9]

Another error checking technique that TCP uses is called selective acknowledgment, the receiving device/application indicates to the sender which packets its received and which packets it still needs to receive. This make sure that only packets that need sending are sent reducing the number of unnecessary retransmissions, improving overall performance.

TCP's flow control, congestion control and error checking mechanisms are critical for ensuring reliable and efficient data transfer over networks and reducing packet loss, particularly over the Internet where traffic can be unpredictable and varied. [10]

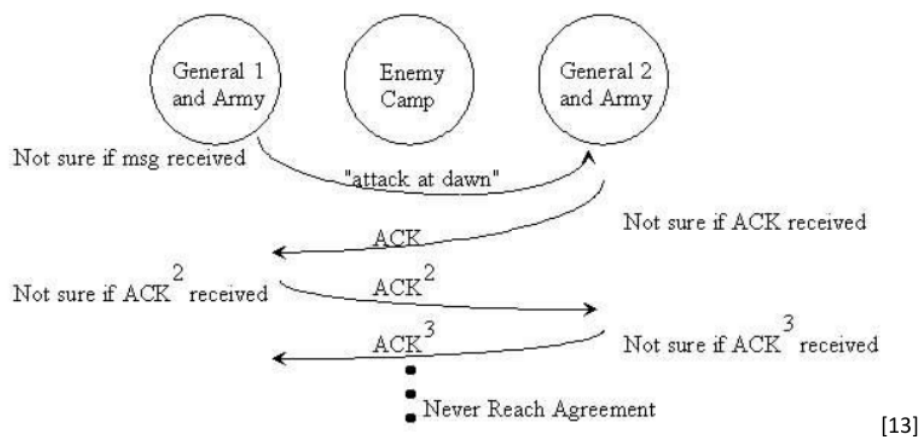
Now that all the data has been transmitted the connection between the two devices/application now needs to be terminated. To terminate the connection another message is sent called a Fin (Finish) packet, the device that wants to terminate the connection sends this packet and the other device/application sends back an Ack packet to acknowledge that its received the Fin packet and also send its own Fin packet to let the original device/application that it too has finished transmitting data, the original device/application then sends an Ack packet to acknowledge it too has received the Fin packet. This process is known as the four-way handshake. [11]

Two general problem

The "two generals" problem is a theoretical problem in computer networking and distributed computing, first introduced by Leslie Lamport in 1982. The problem concerns the difficulty of achieving consensus between two nodes in a network that may be subject to communication failures or delays. [12]

The two general problem refers to the need for acknowledgment and retransmission in network transmission. The two general problem is often described with an army example where there are two sets of troops one set either side of a set of enemy troops, too defeat the enemy both sets of troops must attack at the same time. However, the problem occurs they can only communicate through messengers who may be intercepted by the enemy. The generals need to agree on a time to attack, but they cannot be sure whether their messages have been received by the other general or not.

This is illustrated using the 3 way hand shake below:



Using the acknowledgment and retransmission mechanism with TCP can solve the two general problem. When the sending device/application send a packet it waits for acknowledgement from the receiver before sending the next packet. If the sender does not receive the ACK message in a certain amount of time (captured by the enemy in the analogy) the sender re sends the first packet. The receiving device/application will always send an ACK message for every packet it receives. As mentioned above if the if the receiving device/application detects a missing packet it will request the missing packet to be retransmit. This should always get around the two general problem and make sure that all packets are reliably delivered, even if some of them are lost or delayed.

Maximum Transmission Unit (MTU)

The maximum packet size that can be sent through a network without being fragmented is known as the Maximum Transmission Unit. There is a maximum packet size that each device/application can receive and transmit. Before transmitting a packet, the Maximum Transmission Unit is calculated. The packet is split into numerous packets and transmitted if the next receiving device cannot handle the size of the original packet. We refer to this as fragmentation. The MTU size is influenced by a number of related elements, including the size of the protocol data unit (PDU), its location in the OSI model, and the realities of IP fragmentation. [14]

The MTU size is measured in bytes, and it varies with different network technologies, it is an important factor because it determines the efficiency of data transfer across the network. If a device/applications MTU is too small and fragmentation occurs, it can cause delays in the data getting to the receiving device on the other hand though if the MTU size is too big the packet could struggle to be transmitted over a network which can cause other issues including data loss.

For example, an ethernet connection has an MTU size is 1518 bytes, 18 bytes of which are overhead (header and frame check sequence), resulting in an MTU of 1500 bytes [15] so any packets larger than 1500 bytes will be fragmented and sent in a number of packets.

If a packet is fragmented the original packet keeps the original IP header while the subsequent fragments have a modified IP header that includes information about the fragment's position in the original packet. [16] When the data transmission is complete the fragments are put back together at the receiving end using the information within the IP headers, using this information the receiving device/application is able to figure out if there are any missing fragment, if there are any missing the device/application can request these missing fragments from the original sender.

As previously mentioned, fragmentation can introduce delays and packet loss so it's a good idea to avoid it wherever possible by ensuring packets are smaller than the MTU of the network its being transmitted on.

When determining the size of the MTU its important to factor in the Protocol Data Unit (PDU). In networking, a protocol data unit (PDU) is the basic unit of exchange between entities that communicate between two layers in the OSI model. [17] PDU's provide a common language and structure for exchanging data.

PDUs size and structure depending on what protocol is being used and what layer of the OSI model its being used in. For example, the PDU size for the application layer is different from the PDU size for the transport layer

The structure normally consists of a header information such as the source and destination addresses, the protocol version, and any other control information as well as the data actually being transmitted.

If the MTU is not large enough for the to contain the PDU it leads to the MTU being fragmented and cause delays to the speeds of the data transfer.

The OSI model does play its part in the size of the PDU, the Open Systems Interconnection (OSI) model describes seven layers that computer systems use to communicate over a network. It was the first standard model for network communications, adopted by all major computer and telecommunication companies in the early 1980s:

7	Application Layer	Human-computer interaction layer, where applications can access the network services
6	Presentation Layer	Ensures that data is in a usable format and is where data encryption occurs
5	Session Layer	Maintains connections and is responsible for controlling ports and sessions
4	Transport Layer	Transmits data using transmission protocols including TCP and UDP
3	Network Layer	Decides which physical path the data will take
2	Data Link Layer	Defines the format of data on the network
1	Physical Layer	Transmits raw bit stream over the physical medium

[18]

Each layer affects the size of the PDU, the lower the layer the larger the PDU size which in turn affects the MTU size. This means that the MTU size for the application layer will be smaller than the MTU size for the transport layer.

Each layer adds its own header and footer to the data received from the layer above it, creating a PDU for that layer. The PDUs are:

1. Physical Layer: Bits
2. Data Link Layer: Frames
3. Network Layer: Packets
4. Transport Layer: Segments
5. Session Layer: Data
6. Presentation Layer: Data
7. Application Layer: Data

[19]

So, in conclusion, the MTU size is a critical factor that affects the transfer of data across a network. The size of the PDU, the position of the PDU in the OSI model, and the practicalities of IP fragmentation are all associated factors that have an impact on the MTU size.

To guarantee effective data transport throughout the network, it is crucial to make sure the MTU size is suitable for the size of the PDU and its location in the OSI model.

Synchronicity & Asynchronicity

In network protocols, synchronicity and asynchronicity refer to the timing and coordination of data transmission between networked devices/applications. A synchronicity protocol indicates that the communication is communicated in a specific order and timing, on the other hand asynchronicity communicates in no order or timing.

Network protocols have evolved over time, with many modern protocols now utilizing both synchronous and asynchronous modes of communication to provide optimal performance and flexibility. [20]

Synchronous communication requires each device/application to follow a specific set of rules that sets out when each device/application can communicate with each other. This means that both the sender and the receiver must be present and actively engaged in the communication process at the same time. Normally, when using synchronous each device/application takes in turns communicating with one another using a pre-determined order this allows data to be transmitting a specific real time sequence.

Synchronous communication relies on the presence of a clocking system at both ends of the transmission. These clocks must be synchronized at the beginning of the session [21]

Applications that use synchronous communication are normally time sensitive applications that need real and precise timings where even a small delay or failure could have massive consequences applications like:

- Finance applications
- Voice and video call applications
- Gaming
- Emergency services applications
- Military applications

Asynchronicity does not need a specific order of communication, devices/application are free to transmit data as soon as they are ready, and they do not need to wait for the other device/application to finish. This means that the sender can transmit data without requiring the receiver to be present or actively engaged at the same time.

Input/output (I/O) dispatch models are used in operating systems to manage the flow of data between the CPU, memory, and input/output devices. Performance of a device/application can be affected by I/O operations.

Data can be sent between devices over a network. Without I/O, computers would not be able to communicate to other systems or devices. [22]

Synchronous I/O dispatch model allows the operating system to block an application until the I/O operation is complete. This means that an application cannot perform any other task until the I/O operation has finished. As the device/application cannot move on until the previous task had finish this can lead to reduced performance.

You can combat the recused performance using the synchronous I/O dispatch model by using multiple threads which will allow you to run I/O tasks asynchronously. This means that if a device/application is using multiple threads it can run more than one I/O task at a time.

Because of how relatively simple and fast synchronous I/O tasks action, applications that need real-time communication normally use this model, applications like video and audio communication. Synchronous I/O is also used in many server-side applications, such as web servers, where the server needs to process requests and respond to clients in a timely manner. [23]

The **Asynchronous I/O dispatch model** does not block the execution of tasks, this model allows new I/O tasks to start while other I/O tasks are in progress. This allows more efficient use of the device/application resources. This model is more complex to implement than the synchronous model, but it can lead to improved system performance by allowing multiple I/O requests to be processed concurrently. [24]

Asynchronous I/O model works slightly different to the synchronous I/O model as it uses call-backs and promises to handle I/O tasks, call-backs and promises are sort of a promise to the device/application that when the current I/O task running is complete, this allows the application to continue processing other tasks while waiting for the I/O operation to complete. When an I/O event occurs, the operating system notifies the application by invoking the registered call back function or resolving the promise. The application can then handle the I/O event and continue with other tasks.

The asynchronous I/O dispatch model is widely used in network programming, where it is necessary to handle multiple simultaneous connections efficiently. It is also used in graphical user interfaces, where it allows the application to remain responsive while performing long-running operations, such as file I/O or network communication. [25]

In conclusion, both synchronous and asynchronous communication protocols have their own advantages and disadvantages, and their usage depends on the requirements of the application. Synchronous communication provides real-time communication and precision, making it suitable for time-sensitive applications such as finance, emergency services, and voice/video call applications. Asynchronous communication, on the other hand, provides flexibility and efficiency, allowing multiple I/O requests to be processed concurrently, making it ideal for network programming and graphical user interfaces. Understanding the differences between these protocols can help developers choose the right protocol for their application and optimise their performance.

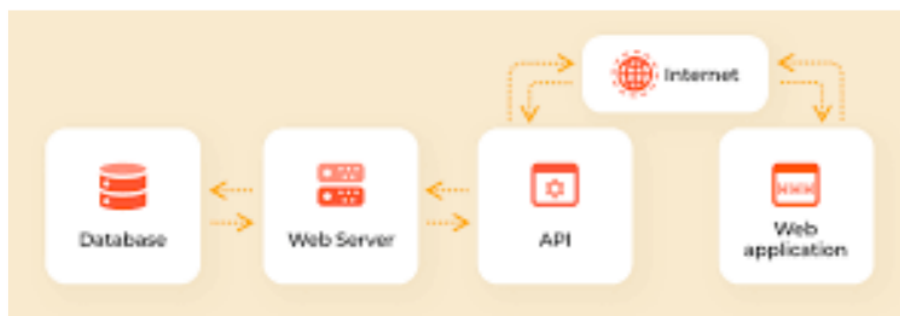
Interoperability

The way organisations work is changing and the need for systems to exchange data has rapidly increased. Interoperability refers to this ability of systems and applications being able to exchange data. Interoperability brings many benefits including data accuracy and real time data sharing. It ensures that different applications, devices, and platforms can work together and exchange information without any loss of functionality or data accuracy.

Data exchange between applications, databases and IT systems is crucial for the growth of modern technology, such as the IoT. Approaches to improving or achieving interoperability include conducting compatibility tests, engineering products with a common standard and using the same technology, coding language or syntax across multiple systems when appropriate. [26]

To achieve interoperability in networked systems/applications both sender and receiver have to be using common standards, protocols and interfaces. Its these standards that define the rules of data exchange making sure that the data is interpreted correctly and consistently.

There are a number of ways of achieving interoperability, one way is through the use of Application Programming Interfaces (API's). API's provide a way for different applications to interact with each other as there are a set number of rules and protocols that each device/application have to follow in order to gain access to the others data.



[27]

Middleware is another approach to achieving interoperability. Middleware refers to a layer of software that sits between different applications and enables them to communicate with each other. The software enables one or more kinds of communication or connectivity between applications or application components in a distributed network. By making it easier to connect applications that weren't designed to connect with one another, and providing functionality to connect them in intelligent ways, middleware streamlines application development and speeds time to market. [28]

Another approach which is widely is using a common language across devices/application, for instance, the HTTP protocol is a widely adopted standard for exchanging data between web servers and clients. Similarly, the TCP/IP protocol is a standard for transmitting data over the internet. Adhering to standards ensures that different systems can communicate and exchange data without any loss of functionality or data integrity.

Interoperability can bring many benefits including:

- Increased efficacy – data exchanges can eliminate data duplication and it allows different systems to work together seamlessly, resulting in faster and more efficient processes.
- Better collaboration – with the exchange of data between systems teams to work together more effectively by providing a common platform to exchange information.
- Improved Data Analytics: Interoperability allows different systems to exchange data seamlessly, making it easier to analyse and gain insights from large amounts of data. It enables organizations to make informed decisions based on data-driven insights. [29]
- Error reduction – as well as increased efficacy reducing data duplication will reduce the amount of human error as the data is only being input once reducing the changes of error.

To summarise, interoperability refers to the ability of systems and applications to exchange data seamlessly, enabling different applications, devices, and platforms to work together and exchange information without any loss of functionality or data accuracy. Achieving interoperability requires common standards, protocols, and interfaces, and can be achieved through approaches such as compatibility tests, using common technology and coding language, and middleware. Interoperability can bring benefits such as increased efficacy, better collaboration, improved data analytics, and error reduction.

Conclusion

In conclusion, as programmers, it is crucial to understand TCP operations, MTU size, synchronicity and asynchronicity in network protocols, and interoperability when developing networked applications. By considering these factors and their impact on networked applications, programmers can design efficient, reliable, and interoperable systems that meet the needs of users.

Bibliography

- [1] - B. Lutkevich. "What is Transmission Control Protocol (TCP)? Definition from SearchNetworking." Networking. <https://www.techtarget.com/searchnetworking/definition/TCP> (accessed Apr. 1, 2023).
- [2] S. Nonis. "What is TCP? How Does it Work?" MonoVM.com. <https://monovm.com/blog/what-is-tcp-protocol/> (accessed Apr. 1, 2023).
- [3] L. Williams, "TCP 3-way handshake (syn, syn-ack,ACK)," Guru99, 04-Mar-2023. [Online]. Available: <https://www.guru99.com/tcp-3-way-handshake.html>. [Accessed: 02-Apr-2023].
- [4] "Three-Way Handshake." Techopedia. <https://www.techopedia.com/definition/10339/three-way-handshake> (accessed Apr. 25, 2023).
- [5] C. Henke. "What Is Transmission Control Protocol (TCP)? | IoT Glossary." emnify | IoT Solution Provider. <https://www.emnify.com/iot-glossary/transmission-control-protocol> (accessed Apr. 01, 2023).
- [6] Postel, J. (1981) Transmission Control Protocol. RFC 793, Internet Engineering Task Force (IETF). (accessed Apr. 1, 2023).
- [7] "Transmission Control Protocol (TCP) (article) | Khan Academy." Khan Academy. <https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/transmission-control-protocol-tcp> (accessed Apr. 02, 2023).
- [8] R. Sheldon. "What is the sliding window technique and how does it work?" Networking. <https://www.techtarget.com/searchnetworking/definition/sliding-windows> (accessed Apr. 04, 2023).
- [9] "IBM Documentation." IBM - Deutschland | IBM. <https://www.ibm.com/docs/en/zvm/7.2?topic=verification-tcpip-checksum-testing> (accessed Apr. 04, 2023).
- [10] "How to modify the TCP/IP maximum retransmission time-out - Microsoft Support." Microsoft Support. <https://support.microsoft.com/en-us/topic/how-to-modify-the-tcp-ip-maximum-retransmission-time-out-7ae0982a-4963-fa7e-ee79-ff6d0da73db8> (accessed Apr. 04, 2023).
- [11] "TCP Connection Termination - IP With Ease." IP With Ease. <https://ipwithease.com/tcp-connection-termination/#:~:text=The%20connection%20termination%20phase%20uses,acknowledges%20with%20an%20ACK%20packet.> (accessed Apr. 04, 2023).
- [12] "Two Generals' Problem – Finematics." Finematics – decentralized finance education. <https://finematics.com/two-generals-problem/> (accessed Apr. 06, 2023).
- [13] "Body." Computer Science | College of Humanities Arts and Sciences. <http://www.cs.uni.edu/~fienup/cs146s01/in-class-overheads-and-activities/lecture10.lwp/lecture10.htm> (accessed Apr. 06, 2023).
- [14] G. Wright. "What is the maximum transmission unit (MTU)?" Networking. <https://www.techtarget.com/searchnetworking/definition/maximum-transmission-unit> (accessed Apr. 05, 2023).
- [15] "MTU Size: What Is It and What Should It Be Set To? - Home Network Geek." Home Network Geek. <https://homenetworkgeek.com/mtu-size/> (accessed Apr. 06, 2023).
- [16] "IP Fragmentation in Detail - Packet Pushers." Packet Pushers - Where Too Much Technology Would Be Barely Enough. <https://packetpushers.net/ip-fragmentation-in-detail/> (accessed Apr. 06, 2023).

- [17] P. Loshin. "What is a protocol data unit, and how does it work?" Networking. <https://www.techtarget.com/searchnetworking/definition/protocol-data-unit-PDU> (accessed Apr. 06, 2023).
- [18] "What is OSI Model | 7 Layers Explained | Imperva." Learning Center. <https://www.imperva.com/learn/application-security/osi-model/> (accessed Apr. 06, 2023).
- [19] "Protocol Data Units - CompTIA Network+ N10-007 - 1.3 - Professor Messer IT Certification Training Courses." Professor Messer IT Certification Training Courses. <https://www.professormesser.com/network-plus/n10-007/protocol-data-units/> (accessed Apr. 06, 2023).
- [20] "Allen-Bradley Communication Types and Protocols | Do Supply Tech Support." Do Supply Tech Support | Information and Troubleshooting on Allen-Bradley, Eaton, GE Fanuc, and more. <https://www.dosupply.com/tech/2020/08/14/communication-types-and-protocols/> (accessed Apr. 06, 2023).
- [21] "Synchronous Communication (Networking)." what-when-how — In Depth Tutorials and Information. <http://what-when-how.com/networking/synchronous-communication-networking/> (accessed Apr. 06, 2023).
- [22] C. Trick. "What is I/O (Input/Output)?" Trusted Computing Innovator | Trenton Systems. <https://www.trentonsystems.com/blog/what-is-i/o> (accessed Apr. 07, 2023).
- [23] "I/O Request Handling." oracle.com. <https://docs.oracle.com/cd/E19253-01/816-4854/6mb1o3ar6/index.html> (accessed Apr. 07, 2023).
- [24] "Synchronous and Asynchronous I/O - Win32 apps." Microsoft Learn: Build skills that open doors in your career. <https://learn.microsoft.com/en-us/windows/win32/fileio/synchronous-and-asynchronous-i-o> (accessed Apr. 07, 2023).
- [25] N. Medhat. "What is `Asynchronous I/O`?" Medium. <https://medium.com/nerd-for-tech/what-is-asynchronous-i-o-b37994359471#:~:text=or%20handed%20off,-,Examples,the%20alert%20and%20then%20resume.> (accessed Apr. 07, 2023).
- [26] S. Lewis. "What is interoperability? | Definition from TechTarget." App Architecture. <https://www.techtarget.com/searchapparchitecture/definition/interoperability> (accessed Apr. 07, 2023).
- [27] A. Rawat. "What is an API? API for beginners | Ayushi Rawat." Ayushi Rawat's Blog. <https://ayushirawat.com/what-is-an-api-api-101> (accessed Apr. 07, 2023).
- [28] "What is Middleware? | IBM." IBM - Deutschland | IBM. <https://www.ibm.com/topics/middleware> (accessed Apr. 07, 2023).
- [29] S. O'Brien. "Interoperability | RingCentral UK Blog." RingCentral UK Blog. https://www.ringcentral.com/gb/en/blog/definitions/interoperability/#Benefits_of_interoperability (accessed Apr. 07, 2023).

Part 1

GRADEMARK REPORT

FINAL GRADE

44/100

GENERAL COMMENTS

Instructor

A reasonable report, though the use of sources could be improved - these trend towards being a little overly web-based.

A limited attempt made at an implementation, a somewhat viable server (though heavily derived from my code). Some consideration given to a C# client, but this is effectively a blank windows forms project. A report detailing some of this included.

Part 1 (report): 63

Part 2 (implementation): 25

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

