



Socket Programming and Concurrency



Steph Challenger
133155

Contents

Transmission Control Protocol (TCP) 2

Maximum Transmission Unit Size 3

Synchronicity and Asynchronicity 5

Interoperability 7

Design Decisions 10

 Overview 10

 Diagrams 11

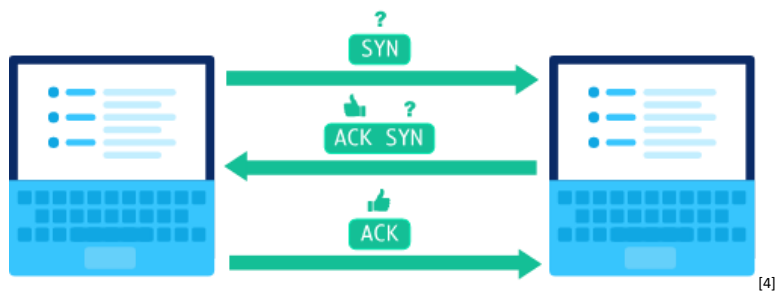
 Features 12

References 14

Socket programming report and implementation

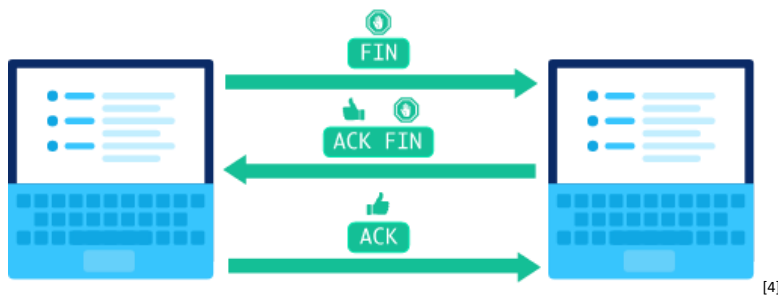
Transmission Control Protocol (TCP)

TCP is a component of the transport layer that connects applications to network services. It is used on top of IP to ensure that packets are transmitted reliably. Many of the issues that emerge from packet-based messaging, such as missing packets, out-of-order packets, duplicate packets, and damaged packets, are addressed by TCP. TCP is a connection-oriented protocol that delivers data transport reliability services and flow control. ^[1 & 2] Two computers set up a connection to exchange data when a client connects to a remote server through TCP, to do this, they must first use a three-way handshake to establish a connection. The three-way handshake is a three-step process in which the client and server swap synchronisation (SYN) and acknowledgment (ACK) packets before the actual data transmission begins. SYN is a connection protocol that is used to start and maintain a connection. It also aids in the synchronisation of sequence numbers across several devices and the ACK is used to notify the opposite side that the SYN has been received. It is built in such a manner that both ends assist you in concurrently initiating, negotiating, and separating TCP socket connections. It allows you to concurrently transmit numerous TCP socket connections in both directions. ^[3 & 4]



Computer 1 sends a packet to computer 2 with the SYN set to 1 and the ACK set to 0. Computer 2 returns a packet to computer 1 with both the SYN and ACK set to 1. Computer 1 responds back with an ACK. The next step is to transfer data packets, computer 1 sends a packet containing data and a sequence number, and computer 2 acknowledges that number by incrementing it by the length of the received data. The sequence and acknowledgment numbers assist the computers in keeping track of which data was retrieved correctly, which data was missed, and which data was transmitted twice by mistake. Sequence numbers are

numbers assigned to packets so that the destination can rearrange them and detect whether or not one is missing. The connection is terminated once data has been transmitted and they have finished sending or receiving data. This is accomplished by one computer sending FIN (finish) set to 1 to the other computer, the other computer responding with ACK and FIN, the first computer responding with ACK, and the connection is then terminated. [3 & 4]



Commented [AG1]: Correct

TCP, despite its high reliability, does not solve the Two Generals' Problem. For example, when one computer attempts to terminate a connection by sending FIN to the other, but the FIN sent back from the other computer gets lost, the connection remains half-open, with one side not knowing that the connection has been terminated. The Two Generals Problem shows how two processes cannot agree on a decision via an unstable network and is an unsolvable problem in computer science. [5]

Commented [AG2]: Maybe but there are strategies

However, TCP does perform error checks and generates reports. It ensures that data arrives in the same order at the receiving end as it did at the sending end. TCP uses a timeout to detect lost packets. The sender starts a timer after sending a packet and places it in a retransmission queue. If the timeout expires and the sender has not had an acknowledgment from the recipient, the packet is resent. If the packet was simply late rather than lost, this may result in duplicate packets, if this is the case, it can be deleted. [4]

Commented [AG3]: Good understanding, not so much critical thought.

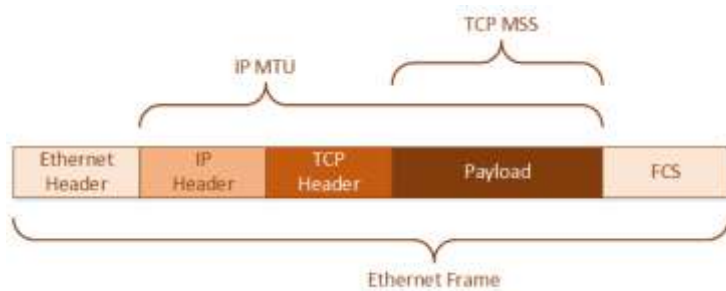
Maximum Transmission Unit Size

The MTU (Maximum Transmission Unit) for a protocol determines the maximum size of its packets or frames, although network devices, encapsulation, and other considerations can make matters difficult. The MTU is the maximum amount of bytes that a single datagram can contain without being split into smaller datagrams or dropping along the way among its source and destination. It is most commonly used to describe the size of packets on an Ethernet network that uses the Internet Protocol (IP). The default MTU size may vary

Commented [AG4]: Why?

depending on the network type, selecting the appropriate MTU size will help improve internet connectivity and performance. Ethernet, for example, has an MTU size of 1500 bytes, while other networks may have higher or lower MTU sizes. The higher the MTU, the more data can be passed into fewer packets, resulting in data being transmitted over the network quicker and more efficiently. [6 & 8]

Commented [AG5]: Usually - this is not absolute



The overall frame size is increased when packets or frames from one protocol are encapsulated inside another one. Because encapsulation includes a protocol header, any 1500-byte packets that are thereafter encapsulated will exceed the network's MTU limit. The amount of bytes added by encapsulation varies depending on the protocol. A packet may start off as a typical IPv4 packet with a 1500-byte MTU, but depending on its location, it may undergo encapsulation that increases its size beyond the MTU. [6 & 8]

One of the most prominent issues with MTU is that higher-level protocols can occasionally produce packets that are bigger than the link can handle, requiring changes to make it function. The solution is fragmentation, which separates the datagram into multiple parts, each of which is small enough to go through the single link using the MTU value set for that interface. The fragmentation occurs at layer 3 in the OSI model (IP layer) and labels the fragmented packets to ensure that the destination host's IP layer understands that the packets should be reassembled into the original datagram. Packet fragmentation is only possible with IPv4. If a packet is sent via IPv4 that is bigger than the MTU, it will be fragmented except if the do-not-fragment flag is set. Fragmentation is not possible with IPv6. In IPv6, if a packet increases its size beyond the MTU, it is dropped. [6, 7 & 8]

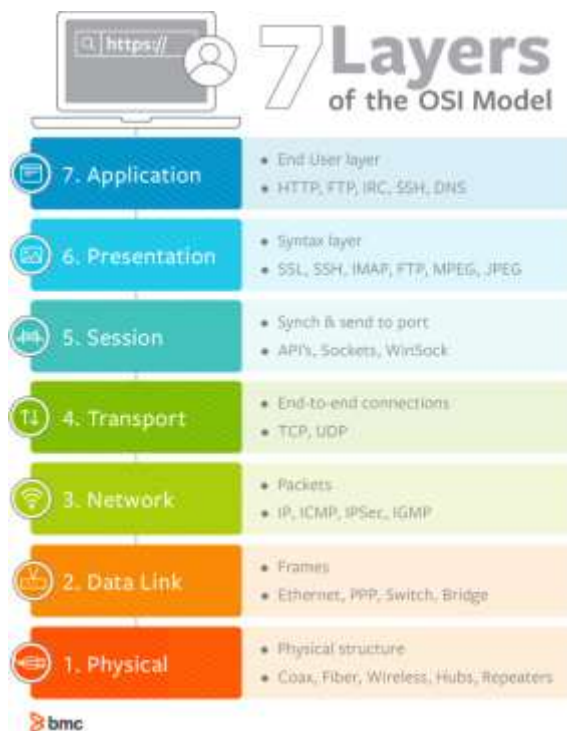
Commented [AG6]: Interesting!

Even though fragmentation can be a solution, it does come with its drawbacks as it reduces performance by adding latency and more data resulting in more traffic being sent. Adjusting the maximum segment size or TCP MSS (Maximum Segment Size) before getting to the data link layer is the most effective technique to avoid fragmentation. MSS is similar to MTU, however, it is utilised at layer 4 in the OSI model with TCP. It is the maximum size that the payload can be, after removing the header size from the IP and TCP. This is sent in the SYN packet as part of the TCP three-way handshake. [6, 7 & 8]

Commented [AG7]: Absolutely - good critical understanding

Referring to the OSI model, the Maximum Transmission Unit (MTU) is the biggest data block that can be processed at layer 3. Layer 3's limit is derived straight from layer 2. Layer 2 employs frames, each of which has a maximum size threshold. The Ethernet frame is on layer 2: data link, the IP protocol is on layer 3: network, and the Maximum Segment Size (MSS) is on layer 4: transport. [6 & 9]

Commented [AG8]: No discussion of PDU



Synchronicity and Asynchronicity

There are two types of transmissions in computer networking: synchronous and asynchronous transmissions. A method of exchanging data between two devices that are connected via a network.

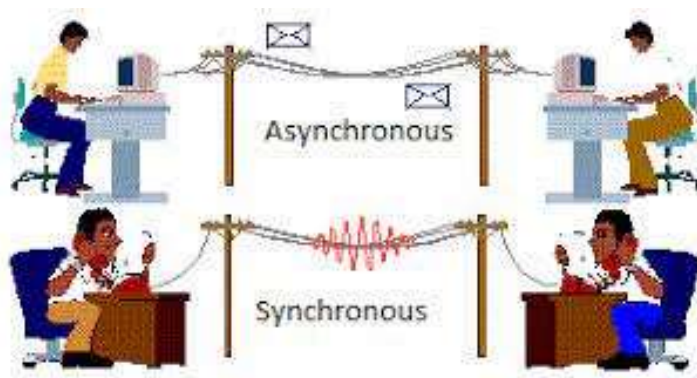
Synchronous data transmission is a form of data transfer that delivers a continuous stream of data signals along with time signals. Data can be transferred at specified intervals in the form of frames or blocks using the synchronous transmission. It assists with ensuring that the transmitter and receiver are in sync with one another. Simply put, synchronisation occurs in real time when at least two devices exchange data at the same time. This communication technique is typically used when large volumes of data need to be moved

from one area to another. Synchronous transmission has several advantages, which include assisting in the transfer of large amounts of data, it provides real-time communication between connected devices allowing for a more in-depth interaction and reduces timing problems as there is immediate feedback with real-time resolutions. Although there is a drawback in that the transmitter and receiver must operate concurrently with the same clock frequency due to time zone differences. Synchronous communication is the way to go when needing an immediate response. Some examples of synchronous transmission are chatrooms, video conferences, phone calls, and face-to-face interactions. ^[10 & 11]

Commented [AG9]: Good examples

Asynchronous data transmission is a flow control method for transferring data from the transmitter to the receiver, also known as the start/stop transmission. Each character or byte transmitted consists of a start and a stop bit with an irregular period of time between them. It transmits one character or eight bits at a time. Each character transmits the start bit before the transmission process has begun. It also sends the stop bit after transmitting the character. The overall amount of bits is ten, which includes the character bits as well as the start and stop bits. It entails the transmission of data between two or more devices without using live or instant interactions. There is no real-time interaction. Asynchronous communication occurs when there is a delay between the time a message is transmitted and the time it is interpreted by the recipient. This form of communication is not usually done in person, and it is also not something that is planned or scheduled. The benefits of asynchronous transmission are that it is time zone friendly, meaning that the transmitter and receiver can operate at different clock frequencies, it allows for flexibility as it gives freedom of when to respond rather than having to respond right away, and it allows for higher quality due to the slower communication, giving time to think and respond when ready. The drawback, however, is that the delay between a message may leave someone hanging and waste precious time. Asynchronous communication is the way to go when a response is not needed immediately. Some examples of asynchronous transmission are emails, letters, text messages, and video recordings. ^[10 & 11]

Commented [AG10]: Fine, this is a reasonable discussion of this.



The key differences between synchronous and asynchronous transmissions are that synchronous sends data in the form of frames or blocks whereas asynchronous sends data in characters or bytes. Synchronous transmission has a fast speed while asynchronous transmission has a slow speed. Synchronous transmission can be costly unlike asynchronous transmission is cost-effective. The synchronous transmission uses constant time intervals and asynchronous uses random time intervals. Lastly, unlike asynchronous transmission, synchronous transmission does not include gaps between the data. ^[10 & 11]

Synchronous and asynchronous transmission both have their benefits and drawbacks. Small quantities of data are sent using asynchronous transmission, whereas large volumes of data are sent using the synchronous transmission. ^[10 & 11]

Input/Output (I/O) requests are handled in a variety of ways by applications. The majority of I/O operations performed by apps are synchronous. While the device executes the data operation, the application thread waits and provides a status code when the I/O is finished. After then, the software can proceed and instantly access the data that has been sent. The ReadFile and WriteFile methods in Windows are performed synchronously when used in their most basic form. They finish the I/O operation before handing control back to the caller. Asynchronous I/O allows an application to send several I/O requests to a device while it conducts the I/O operation. This sort of I/O can boost an application's throughput due to it enabling the application thread to carry on working on other tasks while an I/O operation is being performed. When calling the Windows CreateFile method, a FILE_FLAG_OVERLAPPED flag must be provided to perform the asynchronous I/O. After an asynchronous I/O operation has been initiated, the thread will need to wait for the device driver to complete the data operation before accessing any data from the I/O operation then return to the I/O system. The timing differs for the return from the I/O system to the caller based on the handle being opened for synchronous or asynchronous I/O. ^[13 & 14]

Commented [AG11]: Not sure that we need to know about IO

Interoperability

Interoperability refers to the capability of systems, applications, and databases to work together in a coordinated manner without the intervention of the end user, even if they were produced by different manufacturers in different sectors. To put it another way, data can be exchanged across two or more different components or devices, both by software and hardware. It meets the end user's expectations for quality while having no adverse effects on the transmitting and receiving networks and it allows for unlimited data and resource sharing between devices over LANs (local area networks) and WANs (wide area networks). ^[15] To be interoperable, two systems must be able to communicate data and then present it in a way that a user can understand. The more the systems communicate

with one another via interoperability, the more efficient the company will be. The flow of information from one department to another will be sped up, vital data will be recorded faster, and changing trends and demands will be adapted and reacted to more quickly with seamless integrations. It helps businesses to improve their efficiency and performance while also giving them a more comprehensive overview of all their data. Interoperability can be applied to any type of data sharing. ^[15 & 18]

There are two main types of interoperability, syntactic interoperability is when two or more systems can communicate and share information. It enables different software components to work together despite the fact that their interfaces and programming languages are different. And semantic interoperability is where each system understands the data transmitted between two or more systems. This information transmitted should be relevant as semantic interoperability necessitates useful outcomes that are specified by the users of the systems included in the transmission. ^[15]

How interoperability works is that two or more systems must be able to communicate, interpret, and deliver data in a manner that the other can understand. This is achieved by establishing syntactic interoperability, which entails implementing a standard data format and data structure protocols, it is then followed by semantic interoperability, which entails adding metadata to each data component that links it to a managed, shared vocabulary. There are links to an ontology within this shared vocabulary, which is a data model that defines a collection of ideas in a domain and their relationships. ^[15 & 18]

Commented [AG12]: Correct

When it comes to achieving end-to-end connectivity, network interoperability becomes a must. The more different networks there are, the more important it is to ensure that they can communicate with one another. The benefits of interoperability include increased productivity as businesses improve their efficiency by decreasing the time it takes to process data, lower costs because fewer resources are needed, reduced errors since the data quality improves, and having additional sources due to the data systems being synchronised, resulting in more unified information. ^[16 & 17]

To comprehend the importance of interoperability, it helps to consider what can occur when systems fail to interact effectively with one another. Overspending on applications which are no longer active, such as software that serves only one purpose and is not connected with other apps goes unnoticed for years, resulting in wasted expenses. Incompatible software can also make collaboration harder. Simultaneously, the large number of various systems and solutions that are utilised on a daily basis can be overwhelming, as it requires navigating and switching between applications multiple times during the day. Interoperability does not need to take a long time to be successful, even a minor adjustment can save hours of frustration and hassle. Interoperability is important in software because it ensures that several programmes may share files, communicate information, and utilise the same protocols. ^[16 & 17]

There are two approaches to achieve network interoperability, one way is to require the two networks to agree to a common protocol standard or to create a standard interface where all networks must comply, or the other way is to provide a gateway that interprets between the two protocols. The following strategies can help to facilitate interoperability at various stages of development:

1. Multiple barriers to interoperability can be mitigated or even prevented before any standard is implemented.
2. The next stage is to concentrate on a suitable standardisation strategy. The architecture, needs, and protocols should all be fully covered by the standards specifications.
3. The final step is testing: compliance and interoperability testing to ensure that the equipment and network meet the standards. ^[17]

The testing for interoperability is a process where systems or apps are officially tested in a production setting to ensure that standards are clear and any inconsistencies are discovered. Easy-to-use features, compliance with syntax and data format compatibility, and suitable logical and physical connection mechanisms are all elements in interoperability testing. The Plan, Do, Check, Act (PDCA) Cycle is a standard interoperability testing approach. ^[15]

Commented [AG13]: 2770 - within tolerance

Design Decisions

Overview

A network multiplayer game using sockets has been developed.

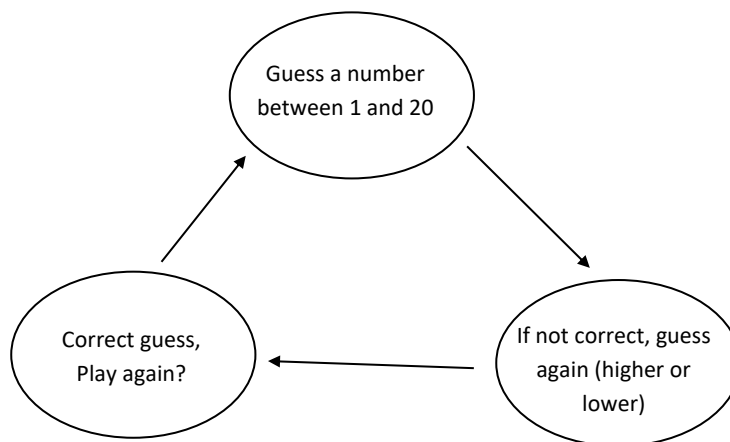
Title: Higher or Lower

Genre: Puzzle game

Environment: Visual studio console-driven. This was chosen as it is the simplest and quickest way to develop the game within the time frame given.

Aim: The aim is to guess the correct number. The server will generate a random number between 1 and 20, and the client (player) has to guess it. If the client's guessed number is higher than the generated number, the server will inform the client that they need to guess a higher number, and if the client's guessed number is lower than the generated number, the server will inform the client that they need to guess a lower number. This will continue until the correct number has been guessed. The client will then have the option to either end the game or play it again.

Game loop: Guess a number between 1 and 20 > If not correct, guess again (higher or lower) > Correct guess > Play again?

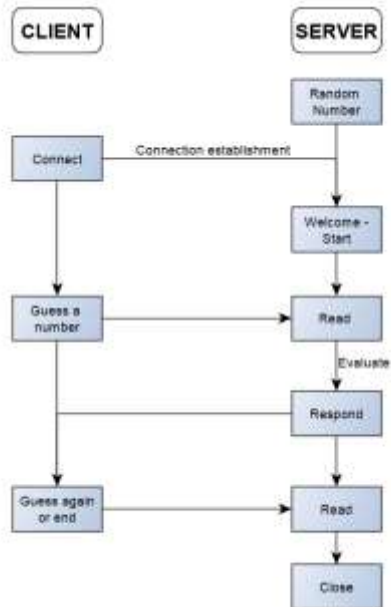


The game loop shows the overall flow control of the game. In most games, the player will keep doing a series of actions over and over again until the player quits. This is an easier way to visualise and understand.

Commented [AG14]: Clear description of interaction planned

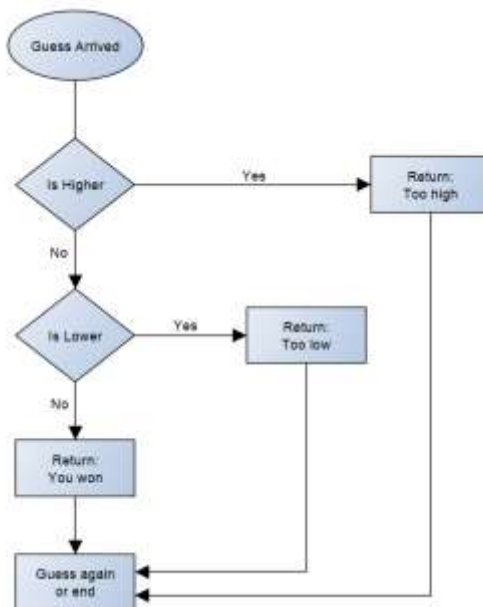
Diagrams

Below displays a diagram showing the game play interactions between the server and the client.



The flowchart diagram below shows the flow once the client has guessed a number.

Commented [AG15]: These are reasonably demonstrative but not correctly formatted.



Features

TCP (transmission control protocol) will be used to develop the network game rather than UDP (user datagram protocol). The reason for this is TCP offers a lot more features and provides a more reliable and error-checked connection between two hosts. TCP is used in several popular games, including World of Warcraft and Minecraft.^[19] Whereas UDP is an unreliable protocol with no certainty that packets will arrive in the correct order, which is crucial in this higher or lower game, because if it were out of order, the game would be broken, and no one would want to play it.^[19] For example, if the server's random number is 5 and the client guesses the number 8, then the server responds that the guess is too low, when it is actually too high, the client will then guess a higher number, denying the client a chance to win and leaving them feeling confused.

Status codes will be implemented into the development. These will be used for the same purpose that HTTP status codes are used, to provide a better understanding and define the type of response. The server side will store these status codes so that they can communicate and record which message types are being received. To keep the overhead to a minimum, the status codes will be set as integers using only one byte. These will be made into constants as they will be used several times throughout the development.^[20]

The data will be compressed using the bit packing technique to achieve quicker refresh rates and reduced bandwidth. It will consist of only using the number of bits that will be needed.^[19] The number of bytes will be put into a buffer size that will be set at 100, which is considerably lower than the usual maximum transmission unit (MTU) of 1400 to 1500 bytes per packet. This higher or lower game will not need to be any higher than 100 bytes as it will only be communicating guessed numbers and the server will respond to the client whether they need to guess higher or lower or if they have guessed the correct number, so 100 bytes per packet will be more than enough. The smaller the packet, the faster the transmission time resulting in better performance.^[21]

A separate class will be developed on the server side called the 'IPSorter' which will automatically discover the best IP address to connect to for a given hostname. This will also be added on the client side, without having to make any changes. It will first search for available IP addresses for an IPv6 address, then repeat the process for an IPv4 address. If an IPv6 address is discovered, it will be returned and used, if not, an IPv4 address will be returned and used.

For better visibility and clarity, a messaging scheme will be established to specify what will be conveyed and how it will be displayed, for example: [status code] : [user] : [data] <EOF>. This will display the status code to determine the type of response, the user's name (whether that will be the client's name or the server), and the content being sent. The EOF at the end stands for End of File, which will be used to signify when a certain data send has been completed as the data cannot be trusted to always fit in a 100 byte buffer.

Commented [AG16]: Why is this?

Commented [AG17]: Clear messaging scheme - good choice.

Commented [AG18]: Good choice - good justification.

Commented [AG19]: It will

Commented [AG20]: My code

Commented [AG21]: Also my code.

Race conditions and messaging queues will be implemented since the development will be a multithreaded process, allowing concurrent processes such as data send, data receive and user input to execute at the same time, as well as using asynchronous methods. The reason for implementing these is to prevent incomplete messages and duplication, as well as to improve the message flow. A race condition occurs when the behaviour of a program is determined by the relative timing or interleaving of numerous threads or processes.^[22] And messaging queues are where messages are added to a queue and sent in the order that they were received. The send process will take the message at the front of the queue and sends it, repeating this process until the queue is empty.^[23]

The features that will be included, as well as the explanations of why they will be used and how important they are, all demonstrate that they follow best practice, as evidenced by their literature references.

Commented [AG22]: I hope not!

Commented [AG23]: No it isn't

Commented [AG24]: 974 - within tolerance

References

1. Corporation P. Progress KB - What is TCP (Transmission Control Protocol). Progress Software Knowledgebase. 2020. [accessed 31 Mar 2022] Available from: <https://knowledgebase.progress.com/articles/Article/P73433>
2. Forouzan B. TCP/IP Protocol Suite | Guide books. Guide books. 2005. [accessed 31 Mar 2022] Available from: <https://dl.acm.org/doi/book/10.5555/1215247>
3. Corporation P. Progress KB - How does a TCP session work. Progress Software Knowledgebase. 2020. [accessed 31 Mar 2022] Available from: <https://knowledgebase.progress.com/articles/Knowledge/P73431>
4. Fox P. Transmission Control Protocol (TCP) (article) | Khan Academy. Khan Academy. 2020. [accessed 31 Mar 2022] Available from: <https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/transmission-control-protocol--tcp>
5. Hash M. "The Byzantine Generals Problem" - Martin Hash.com. Martinhash.com. 2016. [accessed 31 Mar 2022] Available from: <https://martinhash.com/forums/viewtopic.php?t=914>
6. Direction N. How MTU and MSS Affect You Network | Network Direction. Network Direction. 2022. [accessed 2 Apr 2022] Available from: <https://networkdirection.net/articles/network-theory/mtu-and-mss/>
7. Wu C. Resolve IPv4 Fragmentation, MTU, MSS, and PMTUD Issues with GRE and IPsec. Cisco. 2019. [accessed 2 Apr 2022] Available from: <https://www.cisco.com/c/en/us/support/docs/ip/generic-routing-encapsulation-gre/25885-pmtud-ipfrag.html>
8. Hogg S. MTU size issues, fragmentation, and jumbo frames. Network World. 2021. [accessed 2 Apr 2022] Available from: <https://www.networkworld.com/article/2224654/mtu-size-issues.html>
9. Fraihat A. Computer Networking Layers Based on the OSI Model. ResearchGate. 2020. [accessed 2 Apr 2022] Available from: https://www.researchgate.net/profile/Ahmad-Fraihat/publication/348183262_Computer_Networking_Layers_Based_on_the_OSI_Model/links/5ff2dd0192851c13fee7e2b4/Computer-Networking-Layers-Based-on-the-OSI-Model.pdf
10. Priscila. Synchronous vs. asynchronous learning: what's the difference. EasyLMS. 2020. [accessed 2 Apr 2022] Available from: <https://www.easy-lms.com/knowledge-center/lms-knowledge-center/synchronous-vs-asynchronous-learning/item10387>
11. Science. Synchronous and Asynchronous Data Transmission | Computer Science. Teach Computer Science. 2022. [accessed 2 Apr 2022] Available from: <https://teachcomputerscience.com/asynchronous-and-synchronous-data-transmission/>
12. Peterson A. Synchronous and Asynchronous Discussions: Effects on Cooperation, Belonging, and Affect. Files.eric.ed.gov. 2018. [accessed 2 Apr 2022] Available from: <https://files.eric.ed.gov/fulltext/EJ1202382.pdf>

13. Pearson. Understanding the Windows I/O System | Microsoft Press Store. Microsoftpressstore.com. 2012. [accessed 2 Apr 2022] Available from: <https://www.microsoftpressstore.com/articles/article.aspx?p=2201309&seqNum=3>
14. Ashcraft A. Synchronous and Asynchronous I/O - Win32 apps. Docs.microsoft.com. 2021. [accessed 2 Apr 2022] Available from: <https://docs.microsoft.com/en-us/windows/win32/fileio/synchronous-and-asynchronous-i-o>
15. Al H. What is Interoperability? Definition and FAQs | HEAVY.AI. Heavy.ai. 2022. [accessed 9 Apr 2022] Available from: <https://www.heavy.ai/technical-glossary/interoperability>
16. MUELLER H. What is Software Interoperability and How Can it Boost Profits and Productivity? | Formstack Blog. Formstack.com. 2021. [accessed 9 Apr 2022] Available from: <https://www.formstack.com/resources/blog-software-interoperability>
17. Comviva M. All You Need To Know About Network Interoperability - Comviva. Comviva. 2016. [accessed 9 Apr 2022] Available from: <https://www.comviva.com/blog/references/all-you-need-to-know-about-network-interoperability-comviva/>
18. O'Sullivan D. GDPR Data Interoperability Model. Tara.tcd.ie. 2018. [accessed 9 Apr 2022] Available from: http://www.tara.tcd.ie/bitstream/handle/2262/91569/2018_conference_GDPR_data_in_teroperability_model.pdf?sequence=1
19. Pierre. Beginner's Guide to Game Networking. pvigier's blog. 2019. [accessed 15 May 2022] Available from: <https://pvigier.github.io/2019/09/08/beginner-guide-game-networking.html>
20. IONOS. HTTP status codes and their meaning. IONOS Digitalguide. 2019. [accessed 17 May 2022] Available from: <https://www.ionos.co.uk/digitalguide/hosting/technical-matters/the-most-important-http-status-codes-at-a-glance/#:~:text=HTTP%20status%20codes%20play%20an,as%20automatically%20generated%20HTML%20pages.>
21. GameSparks. Real-Time Best Practices - GameSparks Learn. Docs.gamesparks.com. 2021. [accessed 17 May 2022] Available from: <https://docs.gamesparks.com/tutorials/real-time-services/real-time-best-practices.html>
22. Baeldung. What is a Race Condition?. baeldung. 2020. [accessed 17 May 2022] Available from: <https://www.baeldung.com/cs/race-conditions>
23. Dhanushka D. The Stuff That Every Developer Should Know About Message Queues. Medium. 2020. [accessed 17 May 2022] Available from: <https://medium.com/event-driven-utopia/the-stuff-that-every-developer-should-know-about-message-queues-a9452ac9c9d>