

Socket Programming Report

by Ava Faye Kennedy

Submission date: 14-May-2025 08:01AM (UTC+0100)

Submission ID: 259167259

File name: 25473_Ava_Faye_Kennedy_Socket_Programming_Report_664976_3840427.docx (46.62K)

Word count: 3587

Character count: 20542

Socket Programming Report

Ava Faye Kennedy - **Socket Programming and Concurrency ACCA6028**

Table of Contents

Contents

Table of Contents	2
Introduction	3
Transmission Control Protocol	4
Flow Control	4
Error Handling	4
Congestion Control.....	5
Three-way handshake	5
Two generals' problem.....	5
Maximum Transmission Unit	7
The MTU.....	8
Fragmentation.....	8
Optimal MTU size.....	8
Synchronicity and Asynchronicity in network protocols.....	9
Synchronous data transmission	9
Asynchronous data transmission	9
I/O Dispatch Models	10
Interoperability	11
Conclusion.....	12
References.....	13

Introduction

This paper will explore transfer control protocol and surrounding topics. Covering TCP itself as well as the maximum transmission unit, synchronicity and asynchronicity and interoperability. It will seek to explain and through exploration of available research demonstrate the relevance of these topics.

This exploration of topics considers both theoretical frameworks and practical considerations and through a synthesis of available research aims to provide insight into the vast and interconnected field of network programming, with a particular focus on networks constructed to make connections via TCP.

Transmission Control Protocol

Transmission Control Protocol is a connection-oriented protocol for connection and data exchange between devices in networks. Within the OSI model TCP operates at the Transport layer. [1]

TCP ensures that a continuous, reliable connection is held between two devices on a network until the transfer of data between them has finished. It also handles the termination of the connection when communication between devices has finished. Whilst TCP can be less than optimal for real time connections, there are tools that seek to mitigate this flaw[2].

TCP uses a variety of tools to ensure a reliable connection between server and client, when navigating development of networks developers can keep these tools and options in mind to ensure success.[3]

Flow Control

Flow control ensures that the sender on a network does not overwhelm the receiver by transmitting data at speeds that it cannot handle.

Sliding window protocol is a flow control method that is part of TCP protocol. The sliding window protocol determines the volume of bytes that can be sent to a system unacknowledged. The amount of unacknowledged bytes that can be sent is determined by the size of the buffer on the sender and the available space in the buffer for the receiver. [4]

Error Handling

TCP has ways of dealing with errors, mainly through a set of three methods. These methods are checksum, acknowledgement and retransmission.

Checksum: Every segment has a checksum field that is used to find corrupt segments. If a segment is found to be corrupted, then it is summarily discarded by the destination TCP and considered lost.[5] The checksum value is calculated by the sender, the receiver performs the same calculation and the values are compared, this allows receiver to determine if there is an error or not by if these values match or if they do not.[6] The checksum is determined based on the 'TCP Header, TCP body and Pseudo IP header'[6].

Acknowledgement: This is a method for determining if data segments have been received. Control segments that do not contain data that is being sent but instead contain numbers that help to tell the sender if packets have been received.[5] In TCP the receiver acknowledges that data has been received, this allows the sender to continue advancing the process by sending new data. Seq and ack numbers allow ordered and reliable transmission of data via TCP for a network, between a sender and a receiver. These numbers are critical during the TCP Handshake and during data transfer. [7]

Retransmission: When a packet is lost, delivery of a packet is delayed or corrupted when it is checked then that packet is retransmitted. Packets are retransmitted under two conditions. When the sender receives a duplicate acknowledgement or when a certain amount of time passes.[5] When a packet is sent and a certain amount of time passes without an acknowledgement being received then retransmission occurs, this is fairly common on a network. Frequent packet retransmission can occur due to network congestion, dropped packets can mean that the packet is lost on its way to the receiver or in turn that the acknowledgement is lost. [8] [9]

Congestion Control

Congestion control is what the TCP protocol does to attempt to control data flow over a network and mitigate issues with congestion. It achieves this via a congestion window and congestion policy that seek to avoid a network becoming congested during data transfer. If the network cannot process data as fast as it is being sent, then it can instruct the sender to lower the rate of data transfer. Congestion control ensures that the network is not overloaded. The slow start algorithm is one way in which congestion is controlled. The slow start algorithm investigates the network slowly, the amount of data sent is increased until the limit of the network is determined, This happens via the adjustment of the congestion window, which directly influences the number of packets TCP can send without an acknowledgement. [10] [11]

Three-way handshake

TCP uses a three-step process to establish a connection between the sender and receiver on a network. In the first part of this process the sender provides a segment a synchronise sequence number, this tells the server that communication is going to be starting over the network. It also tells the network the sequence number used in segments.

Next the server responds with an acknowledgement of the segment that was sent to it as well as the sequence number the segments are going to use.

Finally, the client acknowledges the server's response, and a two-way connection is established, data transfer can then start. TCP then employs flow control and error handling to ensure that the connection that has been established is reliable [12] [13]

Two generals' problem

The two generals' problem is an analogy for describing how data loss can occur on a network and illustrates that how, because of this TCP is not perfect and even connections established with it can have issues.

In essence, since there is a chance that data loss can happen, the last party to acknowledge is not aware if the acknowledgement was received. The analogy describes two generals, commanding allied armies, they must attack at the same time if they are to be victorious. The only way they can communicate is via messengers, these messengers have a chance to be captured, ergo data loss. General A sends a message informing general B that they are ready to attack, General B sends an acknowledgement back. Herein is where the issue arises, General B does not attack now, they have no way of knowing their acknowledgement was received, as the messenger could have been captured. Thus, the generals will never attack, instead opting to continuously send back acknowledgements to each other's messengers, never knowing when to attack.

Of course, the answer is that one party must be uncertain, hence the analogy demonstrating that the connection cannot be a perfect one. However, you can do things to reduce this uncertainty. Let's say that instead General A simply sends a message indicating that in a week exactly they will attack they wait for a response acknowledging their message, if they receive one, they stop sending the message, otherwise they send their message again, and repeat this process until they receive a message, or until exactly a week when they attack either way. This relaxed two generals problem is analogous to retransmission in TCP. [14]

Retransmission is a process in TCP wherein the negatives effects of packet loss are at least partially mitigated, this is done by resending the packet if it is deemed necessary by either a timeout or a receiving a duplicate packet. [8]

The OSI Model

In Networking the OSI is a theoretical framework by which the standard for internet communication is described. The OSI is made up of layers that a packet traverses to send be sent from one application to another. This model is made up of seven such layers: the physical layer, the data link layer, the network layer, the transport layer, the session layer, the presentation layer and the application layer. These layers 1-7 define how the process of sending data should be.

The physical layer of the OSI is concerned with hardware, such as interfaces on the devices in the network and the cards in pcs that allow connectivity.

The data link of the OSI layer ensures that there is error free, node to node communication by managing frame creation, mac addresses and switch level forwarding.

The network layer of the OSI routes packets between networks using logical addressing and determines optimal paths.

The transport layer guarantees end to end data delivery by segmentation, flow control and protocols like TCP and UDP.

The session layer begins, ends and maintains connections between applications.

The presentation layer of the OSI translates, encrypts and formats data for compatibility and safety in applications.

Finally the application layer provides user services and interfaces for network access, these services include HTTP, Email and others. [15]

Maximum Transmission Unit

The MTU

The maximum transmission unit or MTU is a value that determines the maximum size of the packets that can be sent and received on the network. If a packet is larger than the MTU then it will not be able to be sent without fragmentation, wherein the packet is broken up in order to be sent across the network. [15], [16]

The MTU resides in layer three of the OSI model. The MTU resides primarily on the third layer, the network layer. There it sets the maximum packet size, fragmenting the packets that exceed this. However, it also interacts with layer 2, the data link layer, where it defines the largest size a frame can be. It does this by limiting the payload a frame can carry. On the third layer the MTU defines the largest size a packet can be to be sent across the network. [15], [18], [19]

Fragmentation

During Fragmentation a packet is split into parts and then those parts are sent across the network and reassembled when they are received.

The MTU being too small can lead to frequent fragmentation and wasted bandwidth, fragmentation being undesirable because the packets need to be reconstructed when they are received. This inefficiency gets worse with higher quantity of packets. Fragmentation is necessary for the network to process larger packets but ideally to avoid a reduction in performance fragmentation should not be happening. Another impracticality of IP fragmentation is that each part of a fragmented packet is treated as if it were a single packet, this increases the chance of issues like packet loss.

IP packet fragmentation can cause issues due to out of order arrival and fragment loss. Out of order arrival increases latency and memory usage as incoming fragmented packets have to be reordered and fragment loss is an issue, as a single lost fragment causes the whole packet to be non-functional, which then requires retransmission. Older systems may struggle with data transfer if there is fragmentation, as older hardware can discard fragments. For these reasons, where possible, fragmentation should be avoided.

Optimal MTU size

The MTU should be the same size or a little bigger than the packets in the network layer and travel on the network so that fragmentation can be avoided. The MTU also can't be too big compared to the PDU as this can also cause performance issues [16]. [20]

Path MTU discovery is a way of avoiding fragmentation, it achieves this by discovering the MTU of a path between two hosts. [21] When setting up a network the MTU size must be considered to ensure that there is a lack of fragmentation and that the network can function well. Programs like Wireshark and Traceroute can be used to identify issues with the MTU.

When configuring a network, developers can use MTU diagnostic and discovery tools to ensure their network configuration lends itself to smooth and efficient data transfer without the need for fragmentation.

Synchronicity and Asynchronicity in network protocols

When developing networks and systems that use them it is important to consider the form of data transmission that suits the needs of the system. Different systems have different requirements in terms of synchronous vs asynchronous transmission.

Synchronous data transmission

Synchronous data sends a large volume of data in real time. It is considered a reliable way to transfer data and is an effective method of data transfer across networks. Synchronous data transfer uses a set clock speed. Data is transmitted continuously, when data is transmitted it is separated into parts, these parts are then transferred to the receiver. In synchronous transfer there are no spaces, the character 'syn' prepends the data being sent and 'syn' characters are placed between the parts of data and assist in timing.

The beginning of the data frame is a set of synchronisation characters, these let the receiver know that data is being received. For the data transfer to be successful, both the sender and the receiver need to be in sync, hence synchronous data transfer [22].

Because of the lack of start and end bits, the rate of transfer is fast, however there is a greater chance of encountering an error. Over time, the clocks can become out of sync and the receiver could have the time wrong, causing bytes to become damaged due to lost bits. This issue can be mitigated by resynchronising the clocks regularly. Synchronous data transfer can have lower overhead costs that asynchronous despite generally being more expensive [23], [24]

Some examples of synchronous data transmission are chatrooms and phone and video calls; these services utilise synchronous data transfer to ensure an uninterrupted flow of data.

Asynchronous data transmission

Asynchronous data does not utilise a clock speed, it is non-continuous data transfer. Data is sent without waiting for a real time response. Data is transmitted in packets that can be handled separately. In asynchronous data transfer each character is headed by a start bit and concluded with an end bit, gaps between characters can be present. Data is sent one byte at a time.

Asynchronous data transfer is more scalable than synchronous data transfer as the sender and receiver are less intrinsically linked with each other. Asynchronous transfer is more simple, is more cost-effective and doesn't require two-way communication to function. [23], [25]

Some examples of asynchronous data transmission are emails and forums; these services use asynchronous data transfer, because the services do not need a continuous flow of data, data can be uploaded and sent without much input from the receiver.

I/O Dispatch Models

Modern network applications use various I/O models to handle concurrency and optimise resource consumption. These models influence how developers create servers and clients to manage connections effectively. One of these I/O models is blocking, during blocking a thread is paused, or blocked, until an I/O operation completes. Blocking is simple to implement as it is linear but wastes CPU resources during wait times. It is also not particularly scalable. This model is best for applications with low levels of concurrency and simple single client applications. Another model is non-blocking, this model does not wait on operations, they return immediately regardless of if data is ready. Developers should check readiness themselves using polling. This model avoids thread blocking but needs manual state management, this is also very taxing on the CPU due to constant polls. It also has better scaling than blocking but is still inefficient for larger systems. A use case for this model are applications with mixed I/O. Finally, event driven models allow the monitoring of multiple sockets. This allows the single threading of thousands of connections, efficient use of the CPU and allows for complex state management. This method is useful in high performance systems and systems that need to operate in real time.

When designing systems, I/O dispatch models should be considered based on the scope of the system being created and the requirements of the networks being used. It is vital to choose the correct model for the system to ensure system performance.

Interoperability

Interoperability is an essential feature of modern networks. Networks having interoperability means that the networks and devices on them can interact with each other seamlessly regardless of the differences between them. Two devices from different vendors that use different technologies could communicate and function on a network without issue.

Interoperability is important to ensure that networks that are made up of hardware from multiple vendors function, that networks remain scalable and that the communication on these networks are seamless, maintaining the end user experience.

Achieving interoperability on a network is a multifaceted issue, including adherence to existing standards and standardisation in general. Additionally, in cases where there may be issues with compatibility middleware can act as mediators for networks, assuring that all parts of the network can communicate. TCP/IP is one of the standardised protocols used to ensure network interoperability. Others include Ethernet and Wi-Fi as well as HTTP/HTTPS and other standardised application layer protocols. Organisations such as IEEE define standards by which networking should be conducted and this helps to ensure devices can work with other well on diverse networks. Additionally, products are tested to ensure that they comply with network standards. An example for a standard that seeks to achieve interoperability is the '3rd Generation Partnership Project' that aims to maintain interoperability for phone networks, it does this by setting out how these networks should be setup and maintained. Interworking is another approach to interoperability that uses middleware to help software or hardware that are not natively compatible with each other.

When designing networks developers should ensure they adhere to the standards that have been decided by the industry, and where needed utilise middleware to ensure a seamless user experience via interoperability of their network or system. If these standards inform the development of systems it will not only improve the final result in terms of network function, the system will also be more scalable. Other technologies will be compatible with the system and can be used to expand.

Conclusion

This report has explored the fundamentals of reliable and efficient network communication, discussing TCP, MTU optimisation, synchronicity and asynchronicity, I/O dispatch models and interoperability. These elements shape modern networking, influencing developer decisions and system performance. TCP is still the cornerstone of reliable data transfer, using mechanisms like error handling, flow control and congestion avoidance to mitigate issues on the network. Developers should consider congestion control a priority and optimise MTU and its setup to ensure their networks function effectively. These principles remain the foundation of modern networking.

Developers need to consider an array of different standards and tools when developing a network, and being up to date and informed on these standards and development within the industry is paramount to the development of systems that function effectively in the current technological ecosystem. Whilst there are a wealth of factors that need to be considered there is also a wealth of available information on the topic, developers should seek to familiarise themselves with this information.

References

- [1] 'What is TCP (Transmission Control Protocol)? | Definition from TechTarget', Search Networking. Accessed: Jan. 29, 2025. [Online]. Available: <https://www.techtarget.com/searchnetworking/definition/TCP>
- [2] D. Rossi and M. Mellia, 'Real-Time TCP/IP Analysis with Common Hardware', in *2006 IEEE International Conference on Communications*, Jun. 2006, pp. 729–735. doi: 10.1109/ICC.2006.254794.
- [3] K. R. Fall and W. R. Stevens, *TCP/IP Illustrated*. Addison-Wesley Professional, 2012.
- [4] 'IBM Storage Protect'. Accessed: May 09, 2025. [Online]. Available: <https://www.ibm.com/docs/en/storage-protect/8.1.24?topic=tuning-tcp-flow-control>
- [5] 'Error Control in TCP', GeeksforGeeks. Accessed: May 09, 2025. [Online]. Available: <https://www.geeksforgeeks.org/error-control-in-tcp/>
- [6] 'Calculation of TCP Checksum', GeeksforGeeks. Accessed: May 09, 2025. [Online]. Available: <https://www.geeksforgeeks.org/calculation-of-tcp-checksum/>
- [7] madpacket, 'TCP Sequence and Acknowledgement Numbers Explained', MadPackets. Accessed: May 09, 2025. [Online]. Available: <https://madpackets.com/2018/04/25/tcp-sequence-and-acknowledgement-numbers-explained/>
- [8] 'Retransmission Rules for TCP | Baeldung on Computer Science'. Accessed: Jan. 29, 2025. [Online]. Available: <https://www.baeldung.com/cs/tcp-retransmission-rules>
- [9] O. Ergun, 'Troubleshooting TCP Retransmission Errors: Causes, Prevention, and Solution'. Accessed: May 09, 2025. [Online]. Available: <https://orhanergun.net/resolving-tcp-retransmission-troubleshooting>
- [10] 'TCP Congestion Control', GeeksforGeeks. Accessed: May 09, 2025. [Online]. Available: <https://www.geeksforgeeks.org/tcp-congestion-control/>
- [11] O. Ergun, 'Mastering TCP Congestion Control: A Guide for IT Professionals'. Accessed: May 09, 2025. [Online]. Available: <https://orhanergun.net/tcp-congestion>
- [12] 'TCP 3-Way Handshake Process', GeeksforGeeks. Accessed: Jan. 29, 2025. [Online]. Available: <https://www.geeksforgeeks.org/tcp-3-way-handshake-process/>
- [13] S. S. Panwar, S. Mao, J. Ryoo, and Y. Li, *TCP/IP Essentials: A Lab-Based Approach*, 1st ed. Cambridge University Press, 2004. doi: 10.1017/CBO9781139167246.
- [14] S. A. Brown, 'The Two Generals Problem', LinuxBlog.io. Accessed: Jan. 29, 2025. [Online]. Available: <https://linuxblog.io/the-two-generals-problem/>
- [15] M. M. Alani, 'OSI Model', in *Guide to OSI and TCP/IP Models*, M. M. Alani, Ed., Cham: Springer International Publishing, 2014, pp. 5–17. doi: 10.1007/978-3-319-05152-9_2.
- [16] 'What is MTU(Maximum Transmission Unit)?', GeeksforGeeks. Accessed: Mar. 04, 2025. [Online]. Available: <https://www.geeksforgeeks.org/what-is-mtumaximum-transmission-unit/>
- [17] 'Linux on IBM Systems'. Accessed: Apr. 05, 2025. [Online]. Available: <https://www.ibm.com/docs/en/linux-on-systems?topic=results-network-setup-mtu-size>
- [18] J. D. Day and H. Zimmermann, 'The OSI reference model', *Proc. IEEE*, vol. 71, no. 12, pp. 1334–1340, Dec. 1983, doi: 10.1109/PROC.1983.12775.
- [19] 'What is the OSI Model?' Accessed: May 12, 2025. [Online]. Available: <https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/>
- [20] 'From Data to Frame: The Evolution of PDUs Across the OSI Model', DEV Community. Accessed: Apr. 05, 2025. [Online]. Available: <https://dev.to/stungnet/from-data-to-frame-the-evolution-of-pdus-across-the-osi-model-21gd>
- [21] 'IP Fragmentation in Detail'. Accessed: May 10, 2025. [Online]. Available: <https://packetpushers.net/blog/ip-fragmentation-in-detail/>
- [22] J. Roller, 'Synchronous And Asynchronous Data Transmission: The Differences And How to Use Them', IEEE Computer Society. Accessed: May 08, 2025. [Online]. Available:

<https://www.computer.org/publications/tech-news/trends/synchronous-asynchronous-data-transmission/>

- [23] 'Synchronous and Asynchronous Data Transmission | Computer Science'. Accessed: May 08, 2025. [Online]. Available: <https://teachcomputerscience.com/asynchronous-and-synchronous-data-transmission/>
- [24] 'Synchronous vs. Asynchronous'. Accessed: May 08, 2025. [Online]. Available: https://www.clear.rice.edu/comp310/JavaResources/sync_vs_async.html
- [25] 'Asynchronous Data Transfer', GeeksforGeeks. Accessed: May 08, 2025. [Online]. Available: <https://www.geeksforgeeks.org/asynchronous-data-transfer/>

Socket Programming Report

GRADEMARK REPORT

FINAL GRADE

60/100

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. Additionally, some sections of the report have been cited very well, while a few others (e.g. the interoperability section) haven't been cited at all.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14