

Network Theory P144089

by Kevin John Moses

FILE	4913_KEVIN_JOHN_MOSES_NETWORK_THEORY_P144089_227510_1457846215.DOCX (204.57K)		
TIME SUBMITTED	16-JAN-2020 11:19PM (UTC+0000)	WORD COUNT	3678
SUBMISSION ID	118531728	CHARACTER COUNT	19367

Network Theory

NETWORKED APPLICATIONS ACCA6015

KEVIN MOSES P144089

Table of Contents

1. Introduction	2
2. Transmission Control Protocol	2
2.1. TCP Connection Initiation (The Three-way Handshake)	2
2.2. TCP Connection Closure (Four-way Handshake).....	3
2.3. TCP Connection State.....	3
2.4. Packet Delivery.....	5
2.5. The Two Generals Problem	6
3. The Impact of The Maximum Transmission Unit (MTU) Size on The Transfer of Data	6
3.1. MTU Definition.....	6
3.2. What Affects the MTU Size?	6
3.3. MTU's Impact on Transfer of Data.....	7
3.4. IP Fragmentation.....	8
4. Synchronicity and Asynchronicity In Network Protocols	8
4.1. Synchronicity and Asynchronicity in Network Protocols	8
4.2. Application Types for Synchronous and Asynchronous.....	8
5. Interoperability for Networked Applications.....	9
5.1. Definition	9
5.2. Achieving Interoperability for Networked Applications	9
6. Conclusion	10
References	11

Table of Figures

Figure 1 - TCP Three-way Handshake	3
Figure 2- TCP Four-way Handshake	3
Figure 3 - Endpoint States During the TCP Connection Lifecycle.....	5
Figure 4 - OSI Model.....	7
Figure 5 - Movement of Information from Sender Application to Receiver Host	8

Table of Tables

Table 1 - Example of a Potential Shared Set of Network Protocols for Networked Applications	9
---	---

1. Introduction

In this paper, the author will discuss topics that relate to network theory and how those topics apply to design decisions in the development of networked applications. In particular, the author will consider the topics of transmission control protocol (TCP), the impact of the maximum transmission unit (MTU) size on the transfer of data, synchronicity and asynchronicity in network protocols and interoperability for networked applications.

2. Transmission Control Protocol

The TCP standard is defined in the Internet Engineering Task Force (IETF) Request for Comments RFC 793 [1].

In their paper Performance Analysis of Different TCP Variants in Wireless Ad-hoc Networks [2], Shenoy et al state:

“TCP is a transport layer protocol that ensures ordered delivery of packets which complements the Internet Protocol (IP) at network layer.”

As mentioned by A. M. Alotaibi et. Al in their paper Security issues in Protocols of TCP/IP Model at Layers Level [3], the TCP/IP Reference Model was developed by the Department of Defence’s Project Research Agency. The paper goes on to say that:

“One formal system that has been present as a conceptual model is the TCP/IP Protocol Suite that was formed in 1980 as an internetworking solution...”

and

“The TCP/IP Protocol Suite is a group of different communication protocols working through the Internet and other private communication networks, and it carries most of the essential services running over the network.”

TCP is used as the Transport Layer for HTTP/1.1 and HTTP/2, which are defined by the IETF under RFC 7230 [4] and RFC 7540 [5] respectively. However, HTTP/3, which is defined in the IETF draft proposal draft-ietf-quic-http-23 [6] uses QUIC (Quick UDP Internet Connections) instead of TCP as the underlying Transport layer.

2.1. TCP Connection Initiation (The Three-way Handshake)

As described in Evaluating The Stream Control Transmission Protocol Using UPPAAL [7], a connection between two endpoints is established via a three-way handshake (see Figure 1 below).

Endpoint A (the initiating endpoint) sends a (SYN) message to the Endpoint B (a second endpoint).

Once Endpoint B receives the SYN message, it responds with its own SYN message along with an acknowledgement message (ACK).

After Endpoint A receives the SYN+ACK replies, it responds with its own ACK message, completing the handshake.

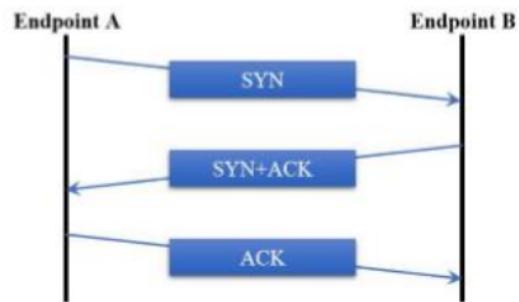


Figure 1 - TCP Three-way Handshake [7]

2.2. TCP Connection Closure (Four-way Handshake)

A TCP connection is closed by a series of messages, known as the four-way handshake (see Figure 2 below) exchanged between two endpoints as described in User-side Evil Twin Attack Detection Using Time-delay Statistics of TCP Connection Termination [8].

To begin closing the TCP connection, the client (Endpoint A), sends a notification to the server (Endpoint B) that it has no more data to send (FIN).

Endpoint B replies to the FIN notification with an ACK response.

Endpoint B then sends its own FIN notification (if it doesn't have any more data to send).

Endpoint A will respond to the FIN notification from Endpoint B with its own ACK message.

Timeouts notwithstanding, the TCP connection between the two endpoints will only be closed on both sides once each has sent a FIN notification and received an ACK response.

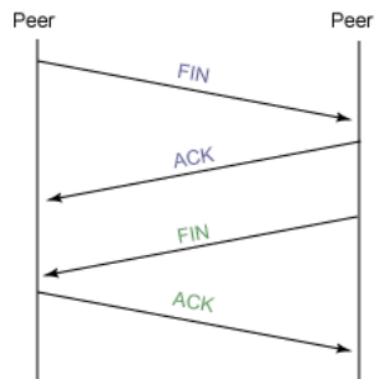


Figure 2- TCP Four-way Handshake [9]

2.3. TCP Connection State

Each endpoint in a TCP connection goes through a series of states which are defined in RFC 793 [1]. These states change during the different stages of the three-way handshake, during data exchange and during the closing of the connection.

The LISTEN state represents an endpoint awaiting a connection request. In regard to the example used above in the *TCP Initiation (The Three-way Handshake)*, this is the state of Endpoint B before it receives a SYN message.

The SYN-SENT state represents the state of an endpoint after sending a connection request, but not yet having had a response. This is the state that Endpoint A moves into after sending a SYN message while it is awaiting a response from Endpoint B.

The SYN-RECEIVED represents an endpoint having received a connection request and having sent an acknowledgement. This is the state that Endpoint B moves into after having send a SYN+ACK message and is awaiting an ACK response from Endpoint A.

The ESTABLISHED state is the state of an endpoint after receiving an acknowledgement response from another server. It is only in ESTABLISHED state where endpoints can send and receive data. This is the state that Endpoint A moves into once it has received a SYN+ACK response from Endpoint B, but Endpoint B will only move into this state once it has received the ACK response from Endpoint A. Both endpoints remain in this state until the Four-way Handshake is initiated.

The FIN-WAIT-1 state is entered by the client once it has sent a close request and is awaiting an acknowledgment of this. This is the state that Endpoint A moves into once it has sent a FIN request and is waiting for an ACK from Endpoint B.

The CLOSE-WAIT state is the state that a server endpoint enters when it has received a close request from the client and responded with an acknowledgement. It will remain in this state until it sends its own close request to the client. This is the state that Endpoint B enters when it sends an ACK in response to a FIN request from Endpoint A. Endpoint B will remain in this state until it has finished sending any remaining data to Endpoint A.

The FIN-WAIT-2 state occurs when the client endpoint has received an acknowledgement of the close request it sent and is waiting for a close request from the server endpoint. Endpoint A would enter into this state when it has received an ACK from Endpoint B in response to its own FIN request. It will remain in this state until it receives a FIN request from Endpoint B.

The LAST-ACK state is the state the server endpoint enters once it has sent its own close request after it has finished sending any remaining data. This is the state Endpoint B moves into once it has sent a FIN request to Endpoint A and is awaiting a final ACK response.

The TIME-WAIT state is the state the client endpoint enters once it has sent the acknowledgement of the close request of the server endpoint. The client endpoint usually remains in this state for a set period of time in case there are any remaining duplicate network packets that may be received in case they interfere with a new TCP connection. This is the state that Endpoint A enters once it has sent an ACK in response to a FIN request from Endpoint B.

The Closed state is the state that both endpoints enter once the TCP connection is fully closed. For the server endpoint this occurs after receiving an acknowledgement of its close request and for the client endpoint this is the state entered after a defined time spent in the TIME-WAIT state. Endpoint B would enter this state once it has received an ACK in response to its FIN request and Endpoint A would enter this state after a certain amount of time spent in the TIME-WAIT state.

An example of how the TCP connection and closing procedures can be seen in Figure 3 below.

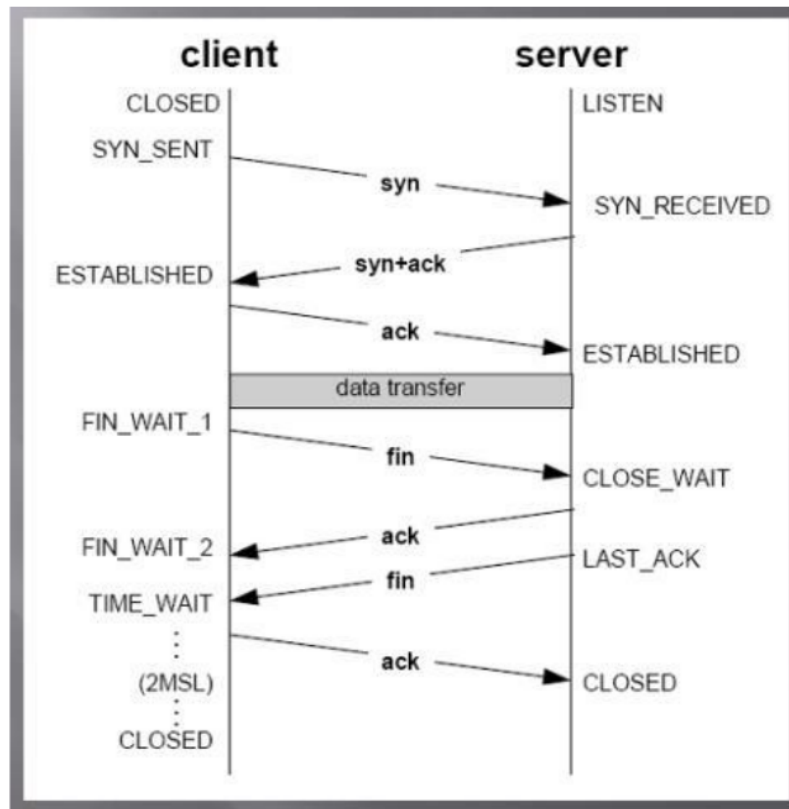


Figure 3 - Endpoint States During the TCP Connection Lifecycle [10]

2.4. Packet Delivery

TCP divides data into segments for transmission over the network. Each segment is encapsulated in a TCP header which forms the packets that are sent. Each packet is assigned a sequence number in the TCP header. This means that the receiving endpoint can take these segments of data, arrange them in order, regardless of the order they are received, and pass the whole message back up to the application layer

Other relevant data held within a TCP header include whether the packet is the last packet sent from the sender, a checksum to ensure the packet itself has not been corrupted during transmission over the network, the amount of data that the sender is currently willing to receive (known as window size) and the port numbers of both the sender and receiver.

The window size is especially important when the receiving endpoint is sending ACKs in acknowledgement of the segments received. It allows the sender to control the rate at which it is sending packets to avoid them being lost and having to be resent which results in network congestion.

In the paper Performance of TCP Variants Over LTE Network [11], TCP's reliability in transmitting packets of data is attributed to its ability to control congestion.

Every segment of data must be accounted for with a corresponding ACK from the receiver, although one ACK can acknowledge the receipt of several segments. Because each packet is numbered, TCP

can recognise packets of data that have not been acknowledged and they be resent. Because of this ability to resend missing data, TCP can also handle any duplicate packets that may be received. This can happen when a packet (either a segment of data or an ACK) is delayed to a point where the sending endpoint resends the packet.

2.5. The Two Generals Problem

The Two Generals Problem relates to a scenario where two entities are trying to achieve consensus by sending messages to each other that may or may not be received. To enable the sender to know that their message has been sent and understood, they expect an acknowledgement message in receipt of their original message [12].

In some instances of this scenario, the generals are planning an attack on an enemy target, but neither will move until they are sure their counterpart will do the same. This leads to a scenario where both generals want to receive an acknowledgment of their acknowledgment leading to an infinite loop of acknowledgements being sent and resent when they don't get an acknowledgement of their acknowledgement [12].

In regard to TCP, this describes the problem of packets and their corresponding acknowledgements being lost during transmission. The sender of data segments will resend these segments if an ACK response is not received within a given time. This lack of an ACK response from a receiver could be down to either a packet from the sender being lost or the ACK response for that packet sent from the receiver being lost.

This can lead to duplication of data packets being sent from the sender, which in turn leads to duplicate ACKs from the receiver as they cannot be sure that their original ACK was not lost. This can especially be a problem in noisy or unreliable networks and the duplication of messages itself adds to network congestion. Impact of The Maximum Transmission Unit Size on The Transfer of Data

3. The Impact of The Maximum Transmission Unit (MTU) Size on The Transfer of Data

3.1. MTU Definition

Maximum Transmission Unit (MTU) is defined in Service Quality Assurance in The IP Networks for Smart Grids [13] as:

“... the size of the largest IP data unit that can be communicated in a single network transaction.”

3.2. What Affects the MTU Size?

The MTU size at Layer 4 (Transport Layer) of the OSI Model (see Figure 4 below) is restricted by the frame size at Layer 2 (Data Link) and the capabilities of the physical devices at Layer 1 (Physical).

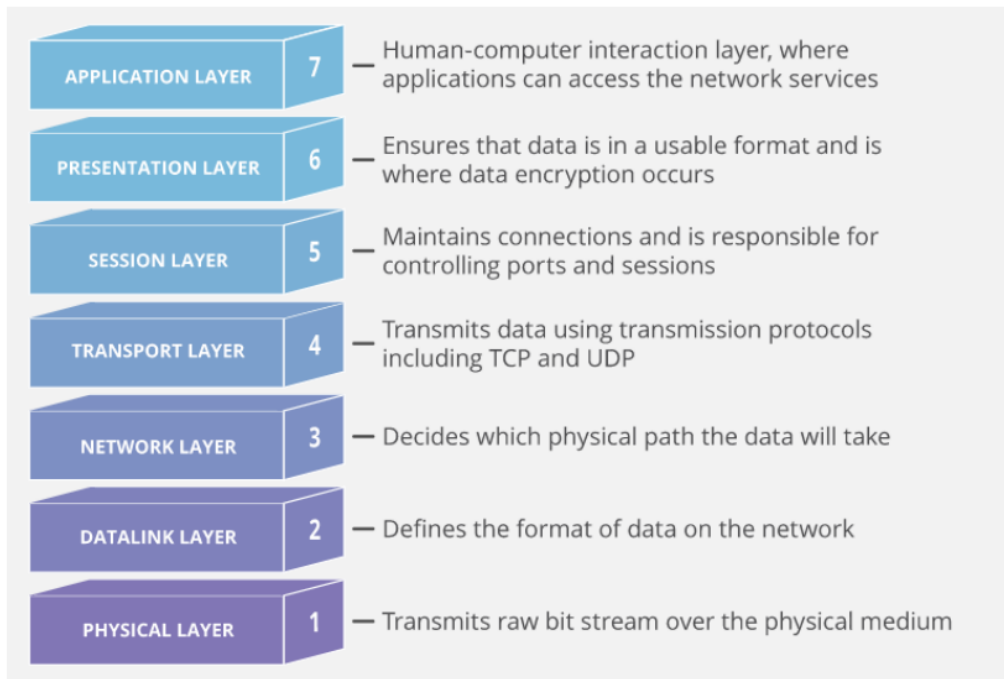


Figure 4 - OSI Model [14]

Layer 2 restricts the MTU by having Maximum Frame Sizes and header overheads. For example, The Ethernet standard originally set the Maximum Frame Size as 1518 bytes and has a header of 18 bytes. This means that an MTU over Ethernet can be at maximum 1500 bytes [15]. The Maximum Frame Size for Ethernet was later increased to 1522 bytes to allow for VLAN tagging [16].

The Physical devices at Layer 1 may fragment IP packets if they are larger than they can handle. The MTU is therefore affected by the maximum size of IP packets that can be sent along a network without being fragmented.

3.3. MTU's Impact on Transfer of Data

The larger the MTU, the larger the chunks/fragments of data can be sent in a packet. This means that one payload is divided into less chunks/fragments and therefore incur less overhead in the form of headers at the various levels of the network.

For example, TCP will break down a data payload from an application into fragments and each fragment will have a TCP header. Each TCP Packet will then have header added at Layers 2 and 3 of the network, e.g. IP headers and Ethernet Frames respectively. Example shown in Figure 5 below.

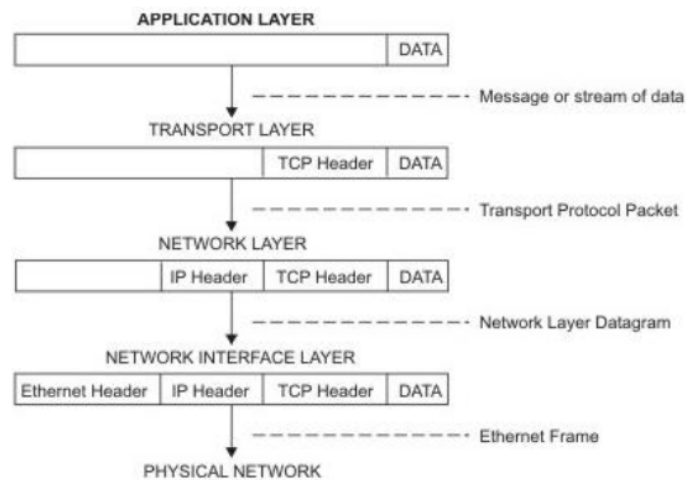


Figure 5 - Movement of Information from Sender Application to Receiver Host [17]

By limiting the number of fragments application data is split into, the overhead of transmitting header data is reduced.

3.4. IP Fragmentation

If a physical device receives a Protocol Data Unit (PDU) which is greater than the MTU of the next hop along the network it can either notify the original sender that the PDU needs to be made smaller, or it may fragment the PDU itself [18].

If Fragmentation occurs, then the packet data is fragmented and headers added to each fragment (creating several PDUs from the original one), for the next receiving device to reassemble the original packet.

Not only does this process incur the overhead of extra header data going over the network, it may be fragmented and reassembled several times along its journey to its destination, further congesting the network.

4. Synchronicity and Asynchronicity In Network Protocols

4.1. Synchronicity and Asynchronicity in Network Protocols

Synchronicity in terms of network protocols means waiting for a sent message to be acknowledged before sending the next message. Essentially making sure that message has been received by the target endpoint before sending the next message [19]. This can also be called guaranteed delivery.

Asynchronicity on the other hand means that messages are sent without waiting for an acknowledgement that the previous message has been received. This scenario is usually used when it is acceptable to lose some messages [19].

4.2. Application Types for Synchronous and Asynchronous

REST APIs are based on the premise that a client will send a request and receive a response, even if the request is malformed or the cannot be fulfilled [20]. This is an example of a synchronous application, as the application (or a thread within the application) will block until a response is received or a time-out occurs.

While with media streaming, it may be preferable to lose some data packets rather than hold up the stream.

The preference of using synchronous or asynchronous communication could be determined the I/O dispatch model of the application sending the data.

In some cases, it may not be preferable to have a single threaded application block waiting for a response or acknowledgement that a message/payload has been received. Also it may not be performant to have multiple threads open all waiting for a response/acknowledgement for inconsequential messages. This will lead to the applications host machine assigning CPU time to threads not doing anything and increase network congestion with requests being duplicates and acknowledgments that message have been received by the target host.

5. Interoperability for Networked Applications

5.1. Definition

The Cambridge Dictionary [21] defines Interoperability as:

“the degree to which two products, programs, etc. can be used together, or the quality of being able to be used together”.

Interoperability is also used to describe the ability of computer systems to share and exchange data [22].

5.2. Achieving Interoperability for Networked Applications

To enable two or more computer systems to communicate over a network they need to be able to be able to send messages that can be understood and know what to expect in response.

This means that all the computer systems need to share a common set of protocols (OSI Model Layers 3 - 7).

For instance, an application may wish to use HTTPS to communicate with another system. Layers 3 - 7 may be as described in the Table 1 below.

Layer	Protocol
7	HTTPS
6	TLS
5	Session Established in TCP
4	TCP
3	IP

Table 1 - Example of a Potential Shared Set of Network Protocols for Networked Applications

Both the sending and receiving computer systems will need access to the Layer 2 of the network, but these do not necessarily need to be the same for the sending and receiving computer systems, e.g. the sending system may be connected to the network via Wi-Fi and the receiving system may be connected via Ethernet.

If all the computer systems are under the control of one body, then they can be built with mirroring protocol stacks to ensure interoperability.

Where this is not the case, then the protocol stack or stacks that a computer system is able to communicate over should be known to the other computer systems that wish to communicate with

it. Using more common protocols that are well supported is likely to make development easier for those designing computer systems to communicate with other computer systems.

Also providing several protocols to communicate over could also help interoperability. For example, a file server could be able to communicate over HTTP, FTP & SSH. That means that another computer system that wishes to communicate with the file server only needs to be able to communicate over one of those Layer 7 protocols.

6. Conclusion

In this paper the author has given an overview of TCP including how it establishes & closes a connection and how it ensures delivery of data. The author has also described how MTU is defined and how it affects data transfer over a network and how IP fragmentation is used when the MTU is set to high. The author has explained synchronicity and asynchronicity in network protocols and discussed what types of applications could make best use of synchronicity and asynchronicity. Lastly the author gave a definition of interoperability between computer systems and how it can be achieved.

References

- [1] J. B. Postel, "RFC 793: Transmission Control Protocol," September 1981. [Online]. Available: <http://www.ietf.org/rfc/rfc0793.txt>. [Accessed 31 October 2019].
- [2] S. U. Shenoy, M. S. Kumari, U. K. K. Shenoy and N. Anusha, "Performance analysis of different TCP variants in wireless ad hoc networks," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, 2017.
- [3] A. M. Alotaibi, B. Alrashidi, S. Naz and Z. Parveen, "Security issues in Protocols of TCP/IP Model at Layers Level," *International Journal of Computer Networks and Communications Security*, vol. 5, no. 5, pp. 96-104, 2017.
- [4] R. Fielding and J. Reschke, "RFC7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing," June 2015. [Online]. Available: <http://www.ietf.org/rfc/rfc7230.txt>. [Accessed 31 October 2019].
- [5] M. Belshe, R. Peon and M. Thomson, "RFC 7540: Hypertext Transfer Protocol Version 2 (HTTP/2)," May 2015. [Online]. Available: <https://tools.ietf.org/html/rfc7540>. [Accessed 31 October 2019].
- [6] M. Bishop, "Hypertext Transfer Protocol Version 3 (HTTP/3) draft-ietf-quic-http-23," 12 September 2019. [Online]. Available: <https://tools.ietf.org/html/draft-ietf-quic-http-23>. [Accessed 31 October 2019].
- [7] S. Saini and A. Fehnker, "Evaluating the Stream Control Transmission Protocol Using Uppaal," in *Proceedings 2nd Workshop on Models for Formal Analysis of Real Systems*, Uppala, 2017.
- [8] E. Kuo, M. Chang and D. Kao, "User-side Evil Twin Attack Detection Using Time-delay Statistics of TCP Connection Termination," in *20th International Conference on Advanced Communication Technology (ICACT)*, Chuncheon-si Gangwon-do, 2018.
- [9] M. Jones, "Better networking with SCTP," IBM, 28 February 2006. [Online]. Available: <https://www.ibm.com/developerworks/library/l-sctp/>. [Accessed 2 January 2020].
- [10] aeguana, "TCP Socket States," 12 July 2015. [Online]. Available: <http://blog.aeguana.com/2015/07/12/tcp-socket-states/>. [Accessed 2 January 2020].
- [11] S. A. Nor and A. N. Maulana, "Performance of TCP variants over LTE network," *AIP Conference Proceedings*, vol. 1761, no. 1, pp. 20-21, 2016.
- [12] J. Gray, "Notes on Data Base Operating Systems," in *Operating Systems, An Advanced Course*, New York, Springer-Verlag, 1978, p. 393-481.
- [13] P. Ceferin, Z. Toroš, R. Đukić, I. Štih and B. Zupan, "Service Quality Assurance In The IP Networks For Smart Grids," *CIREN - Open Access Proceedings Journal*, vol. 2017, no. 1, pp. 1459-1462, 2017.

- [14] Cloudflare, "What Is The OSI Model?," [Online]. Available: <https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/>. [Accessed 12 January 2020].
- [15] A. Khalaf, M. Salam and K. Ahmed, "Investigation of different ethernet wiring and different frame size to enhance the performance of LAN," in *19th International Conference on Advanced Communication Technology (ICTACT)*, Bongpyeong, 2017.
- [16] IEEE, "IEEE Std 802.3ac-1998," [Online]. Available: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=729403&isnumber=15720>. [Accessed 15 January 2020].
- [17] IBM, "TCP/IP Protocols," [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/ssw_aix_71/network/tcpip_protocols.html. [Accessed 12 January 2020].
- [18] R. Bonica, F. Baker, G. Huston, R. Hinden, O. Troan and F. Gont, "IP Fragmentation Considered Fragile," 10 October 2018. [Online]. Available: <https://tools.ietf.org/id/draft-ietf-intarea-frag-fragile-01.html>. [Accessed 12 January 2020].
- [19] M. Lim, "Supporting synchronous and asynchronous communications in event-based," *International Journal of Advanced Computer Research*, vol. 9, pp. 11-19, 2019.
- [20] Microsoft, "Web API design," 1 12 2018. [Online]. Available: <https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design>. [Accessed 12 January 2020].
- [21] Cambridge University Press, "interoperability," [Online]. Available: <https://dictionary.cambridge.org/dictionary/english/interoperability>. [Accessed 2 January 2020].
- [22] A. Valatavicius and S. Gudas, "Measuring Enterprise Application Software Interoperability Capability," in *Doctoral Consortium/Forum@DB&IS*, Trakai, 2018.

FINAL GRADE

81 / 100

GENERAL COMMENTS

Instructor

A very good selection of sources used, mostly of high academic quality. Number 10 is an outlier - similar sources should be avoided for future assessments.

Quotes are used in quite a few places. Quotes are perfectly acceptable academic practice, however you should try to avoid them if you can. As a rule, you should use a quote only when the exact wording of something is extremely important - paraphrasing is always better. In this report, every instance of a quote would have been better if it were paraphrased instead.

The work overall is of a high quality - you clearly understand the topic well and have a good written style. Well done, Kevin.

Engagement with Literature Skills (20): 15

Knowledge and Understanding Skills (30): 25

Cognitive and Intellectual Skills (30): 24

Transferable Skills for Life and Professional Practice (20): 17

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
