Active class 5: How can I transport my application data reliably?

The learning objective of this class is to learn why we need reliable transport layer protocol, i.e., connection-oriented Transmission Control Protocol (TCP) and how TCP can provide reliable operation.

At the end of this activity, you should be able to:

1. Explain the operation of TCP (including the handshaking) and how it provides reliable communication.
2. build a simple client-server chat application that uses TCP as transport layer protocol.

This class activity is designed to be worked through active participation and collaborating with peers under the guidance of the teaching team in the class. The active classes are designed to be interactive, and they are here for you to extend your learning. However, these classes will only help you to enhance your learning if you come prepared. **To work on the class activities, you will be expected to have completed Transport Layer Module.** You need to have a basic understanding of layered model, TCP/IP, transport layer operation, and TCP. If you are not familiar with any of the above, please head back to the CloudDeakin unit site and complete the relevant modules before starting this active class.

The active classes are related to assessment tasks on OnTrack. After learning about different concepts from the content provided in the unit site, you will expand on this knowledge by working on activities designed to put these concepts in practice during the active classes and submit the completed tasks to OnTrack in the same week. The teaching team will guide and support your learning during these activities. This will help you manage your time and tasks better to avoid tasks piling up towards the deadlines. If you do not complete these activities in class, you will need to work on them in your own time, with limited support available from the teaching team.

The class activity is split into three parts. First, you will conduct a group discussion. Then, an analysis using Wireshark. Finally, you will build a simple client-server application using TCP.

Activity 1:

This activity is a group activity. You need a group of four people.

1. Assume one of your group members sent you a message. However, you have never received it. What went wrong? How do you make sure that the message is properly received?
2. Assume you are trying to access a web page containing an image. Now, your job is to make sure that you get the image properly. The image cannot be received in one packet, and it will be broken down to 10 packets.
3. Partner up with one of your group members and develop a transport layer protocol to guarantee that the image is properly received by the client. One can act as the server where the webpage/image is stored and the other can act as the client who wants to receive the image (this means your group has 2 pairs of server-client networks).
4. Write down protocol that you would use, including the communication between the client and server and all the messages that you are passing.
5. How do you guarantee that your protocol does the job as you expected?
6. Once you have finalized your protocol, check how other group members developed their protocols.
7. With your group, discuss whether your protocol valid when you need to access a large file that might be broken down into 1000 packets. If not, what are the changes you would like to make in your protocol?

Activity 2:

Now you are familiar with Wireshark. Therefore, you should be able to analyze protocols and packets with less instructions provided. Today, you are going to analyze TCP.

1. You can analyze the TCP three-way handshake in Wireshark packet capture. Analyze the order of segments sent/received by two end systems to establish a TCP connection.
2. Analyze the segment headers used, their purpose and sizes.
3. Make sure you take the screenshots of TCP segments that will be used for your task submissions.

If you require more guidance, please watch the following video.

https://www.youtube.com/watch?v=3Zb_EebU22o

Activity 3:

In this activity, you will build a simple client-server chat application that you built in Active Class 4 but now using TCP. TCP is a connection-oriented protocol where UDP is a connectionless protocol. Therefore, in TCP, the client and server need to handshake first and establish a TCP connection before they transmit data to each other.

You can partner up with a group member to build this simple client-server application. It would be good if you can partner up with the same person that you have done Activity 1.

You will build the following simple client-server application using Python.

- The client should be able to send a message (for example, “Hello SIT202”) to the server.
- Once the server receives the full message, it counts the number of characters it receives and displays the number in the terminal.
- The server responds to the client by sending the number of characters it received and the same message with uppercase letters.
- The client should receive the number and the message sent by the server.
- The client shows the received message on its terminal.

1. First you need to draw a diagram to explain the operation and communication of the client- server program that uses TCP.

2. Then one of you can develop the client side and the other can develop the server side of the program (you can also develop both and demonstrate the outcome if you chose to do so)

3. You need to show the output of the client- server program to demonstrate that your program works as expected. Please make sure to include the server and client codes and output (screenshots) in your lesson review.

Above and Beyond Tasks:

Those who are targeting for Credit and above can complete the following task as part of Task 4.1C and 5.2D to demonstrate your deeper understanding on TCP.

- Discuss the differences between the following three protocols and the reasons for using these protocols.
    - Stop-and-wait
    - Go-back-N
    - Selective repeat
- Why do we need congestion control in TCP? Explain your answer.