Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CSU33031 Computer Networks
Publish/Subscribe Protocol for Audio/Video

September 8, 2023

1 Introduction

The focus of this assignment is to learn about protocol development and the information that is kept in a header to support functionality of a protocol. Protocol design is always a balancing act between introducing functionality that relies on additional header information and the overhead that the additional header information introduces.

2 Protocol Details

The aim of the protocol is to provide a distribution mechanism for video, audio and text content using a publish-subscribe mechanism based on *UDP datagrams*. The encoding of the header information of this protocol should be implemented in a *binary format*.
The protocol involves a number of actors: One or more subscribers, a broker, and one or more producers. A subscribers issues subscriptions for content to a broker and receives published content from this broker. The broker processes subscriptions and publications, keeps records of publications and forwards content from a publishers to potential subscribers. The header information that you included in your packets has to support the identification of subscriptions and published content - potentially consisting of a number of packets and the management of subscriptions and publications by the broker.

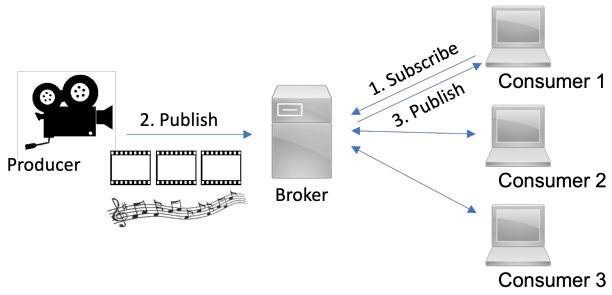

Figure 1: As a very simplistic starting point, a producer e.g. software of a video camera sends a frames to a broker. This broker distributes the frames to consumers that previously subscribed to content at the broker

As motivation, assume that you have been tasked to developed a new protocol for IRL[1] streamers where content will be created by the streamer, send to your broker and distributed from there to an audience. LiveU [2] for example uses a propriatary protocol to a server that then can use the Secure Reliable Transport (SRT) protocol [6] to distribute video streams to streaming services. SRT[1][3] is a good example to look at

[1] "**IRL = In Real Life**"
[2] `https://cdn-liveutv.pressidium.com/wp-content/uploads/2021/07/Solo-Pro-brochure_July23.pdf`
[3] `https://en.wikipedia.org/wiki/Secure_Reliable_Transport`

for a simplistic header design.
The basic functionality that your protocol has to provide is to support subscriptions for content by consumers, publication of content by publishers and distribution of content by the broker to interested consumers. Streams are made up of individual frames. The producer will send individual frames to the broker and the broker will forward the framees to the respective consumers. Frames may arrive out of order or not at all - a frame missing is not much an issue for a stream; however displaying an older frame after a newer would result in a strange viewing experience. Your implementation needs to take this into account.
The assignment focusses on protocol design and communication i.e. you do not have to implement frontends that capture and display frames, print-outs to a screen or logfile are sufficient. The assignment information on Blackboard includes a number of frames as an example, you could read
Your protocol should support multiple producers and multiple consumers at any given time i.e. there may be 3 producers, each with 2 consumers. Producers should be identified by a 3-byte code e.g. "ABCD99" and their current streams are given a number, encoded in another byte e.g. "ABCD9901" for the stream 1 by producer "ABCD99". Consumers may elect to subscribe to a specific stream or to all streams from a given producer. This information has to be encoded in binary in the headers and be discussed as part of your header design.
With every protocol design, there are a variety of options that influence the amount of your header information and functionality that can be added to a protocol. For example, a producer may send out frames as well as audio or producers and consumers may send text related to a stream that is distributed to subscribers. The proprietary protocool used by LiveU allows for a streamer to send traffic over 4 different connections to a server and the server combines this information into a single stream to the consumer e.g. byte 1, 5, 9 of a frame are send over connection 1, byte 2, 6, 10 are send over connection 2, etc and if a part of a frame is dropped by one connection, the consumers would still receive the remaining parts. It is up to you to decide on the functionalities you intent to implement and how In the deliverables, you need to discuss the functionalities that you choose to implement and justify why you included them in your protocol.
The following flow diagrams, figure 2 is an example of visualizations of network traffic between components. They show the sequence of messages exchanged between components with the start of their transmission at the sender and their arrival at the receiver. Please use Flow Diagrams like these to document the communication between components of your implementation in your videos and in your final report.

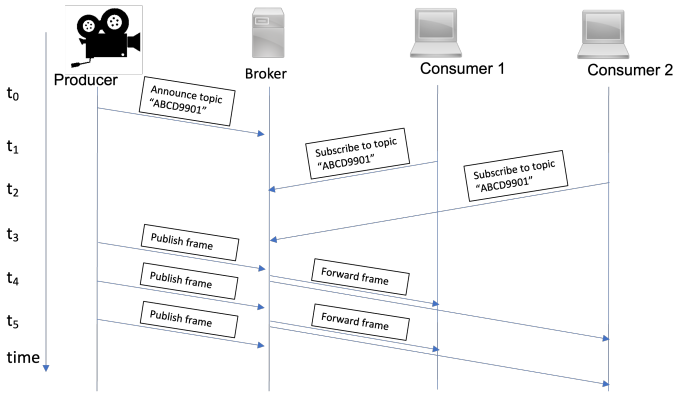

Figure 2: The producer would contact the broker to announce a specific stream. The consumers send subscriptions to the broker for a stream and when the producer transmits frames of a stream to a broker, the broker will forward these to consumers that have subscribed

The protocol can be implemented in a programming language of your choosing. One of the conditions is that the communication between the processes is realized using **UDP sockets and UDP Datagrams**. Please avoid Python 2.7 because the implementation of Datagram sockets in the obsolete versions is based the transfer of strings instead of binary arrays.
The easiest way to start with the development of your solution is possibly to connect your components through the localhost interface of a machine; however, at the end, you will need to be able to demonstrate that your protocol can connect components located at a number of individual hosts. There are a number

of platforms that support the simulation of topologies or provide virtual infrastructures e.g. Docker [2], Mininet [5], etc. For someone starting with socket programming and networking, I would suggest to use a platform such as Docker or Mininet. However, these are only suggestions and you need to make the decisions how to implement your solution.
You can use Generative AI tools such as GitHub Copilot[4] or OpenAI's ChatGPT[5]. You have to indicate in your videos and reports material that has been created with the help of these tools and you are responsible for any potential inaccuracies that this may introduce.

3 Deliverables & Submission Details

The deliverables for this assignment are split into 3 parts: 2 videos, a report describing your solution and the files making up the implementation of your solution. The deadline for the submission of these deliverables are given in Blackboard.
One component of the deliverables at every step is the submission of captures of network traffic. These captures should be in the form of PCAP files [3]. Programs such as Wireshark [4], tshark, etc offer functionality to capture network traffic from interfaces. The videos and reports should NOT contain your student ID.

3.1 Part 1: Video & PCAP file

The video of part 1 should demonstrate the initial design of your solution and a capture of network traffic between the components of your solution and the files of your traffic capture in pcap format. In the video, you should explain the setup of the topology that you are using and the information that makes up the header information in your traffic captures.
The submission process for this part consists of two steps: 1) Submitting the PCAP file or files that you captured from your network traffic, and 2) submitting the video for this step.
The video is to be no longer than 4 minutes; content past the 4 minute-mark will be ignored during marking. Videos with voice-over using text-to-speech (TTS) will not be accepted and marked with 0.

3.2 Part 2: Video & PCAP file

The video of part 2 should demonstrate the current state of your solution, the functionality that you have implemented so far, and a capture of network traffic between the components of your solution and the files of your traffic capture in pcap format. In the video, you should explain the basic implementation fo your protocol and the information that is being exchanged between the components of your solution.
The submission process for this part consists of two steps: 1) Submitting the PCAP file or files that you captured from your network traffic, and 2) submitting the video for this step.
Videos with voice-over using text-to-speech (TTS) will not be accepted and marked with 0. The video is to be no longer than 4 minutes; content past the 4 minute-mark may be ignored during marking.

3.3 Final Deliverable

The final deliverable should include a report that describes the components of your solution and their functionality, the protocol that you implemented and the communication between the components of your solution, the topology that you used to run you solution and how your solution was executed.
The submission process for this part consists of three steps: 1) Submitting the PCAP file or files that you captured from your network traffic, 2) submitting the source code and any files that may be necessary to execute your solution, and 3) submitting the report about your solution.
The files that contain the implementation and the report should be submitted through Blackboard. Every file should contain the name of the author. The source files of the implementation should be submitted as an archived file e.g. ".zip" or ".tar.gz". The report should be submitted as either word- or pdf-document. The deadline for the submission is given in Blackboard.
The report may be submitted to services such as TurnItIn for plagiarism checks.

[4] `https://github.com/features/copilot`
[5] `https://openai.com/chatgpt`

4 Marking Scheme

The contribution of the assignment to the overall mark for the module is 30% or 30 points. The submission for part 1 and 2 will be each marked out of 5 points and the submission for the final part will be marked out of 20 points. The mark for the final deliverable will be split into 50% for the functionality of your solution and 50% for the documentation through the report.

References

[1] Wikipedia Authors. Secure reliable transport. Wikipedia Page, 2023.

[2] Docker. Docker project page. https://www.docker.com, visited Sep 2021.

[3] Wikipedia Editors. pcap - wikipedia page. https://en.wikipedia.org/wiki/Pcap, visited Aug 2021.

[4] Wireshark Foundation. Wireshark project page. https://www.wireshark.org, visited Sep 2021.

[5] Mininet. Mininet project page. http://mininet.org, visited Sep 2021.

[6] Maria Sharabayko, Maxim Sharabayko, Jean Dube, Joonwoong Kim, and Jeongseok Kim. The srt protocol - draft-sharabayko-mops-srt-01. Technical report, Internet Engineering Task Force (IETF), sep 2020.