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

CSU33031 Computer Networks
Flow Forwarding

October 19, 2022

1 Introduction

The focus of this assignment is to learn about decisions to forward flows of packets and the information that is kept at network devices that make forwarding decisions. The design of forwarding mechanisms aims to reduce the processing required by network elements that forward traffic while providing scalability and flexibility.

2 Protocol Details

Due to the COVID pandemic, an increased number of employees in a company are working from home. As development team of a network provider, you are being asked to look at the concept of Software-Defined Wide-Area Networks and extend the concept to ADSL routers at the homes of individual employees. Your task is to develop an overlay that links implementations of forwarding mechanisms in ADSL routers to network elements of the provider to forwarding services at a cloud provider. The forwarding information in the forwarding tables of the implementation at the ADSL routers, the network elements at the provider and the cloud provider will be controlled from a central controller at the network provider (see figure 3).

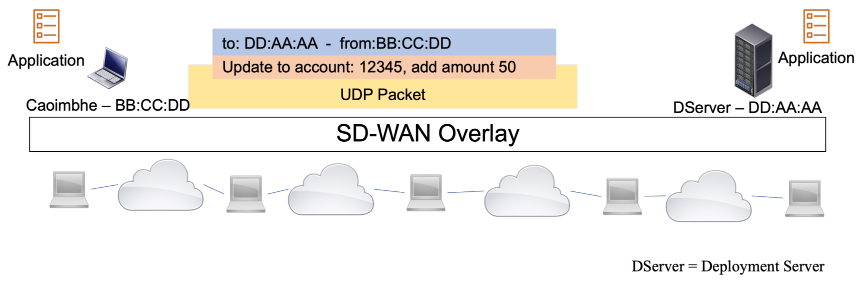
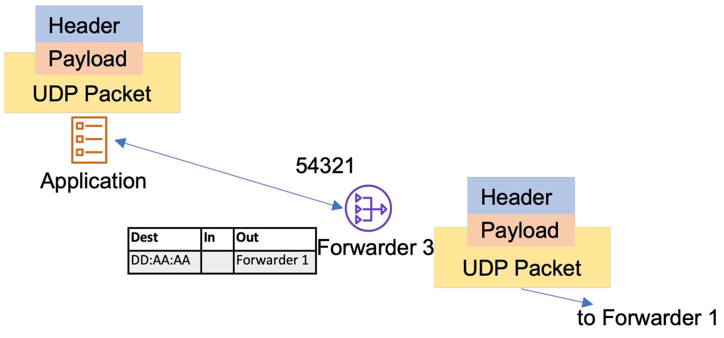

Application
Caoimbhe – BB:CC:DD
to: DD:AA:AA - from:BB:CC:DD
Update to account: 12345, add amount 50
UDP Packet
Application
DServer – DD:AA:AA
SD-WAN Overlay
DServer = Deployment Server

Figure 1: An application will send UDP datagrams to a forwarding service on the local host. The forwarding service will consult a table to determine from the header information of your protocol where to forward the datagram to. The table will provide it with the IP address of the network element that is the next hop and your implementation should forward it to another instance of the forwarding service on this IP address.

The implementation of the forwarding services for the overlay will use a UDP socket bound to port 54321 at every network element. An application at employees devices will be using your overlay by creating a header you design, attach the payload of the application to be transmitted, and send this data as payload to your forwarding service, for example bound to port 54321 on the default gateway, such as shown in figure 2. The forwarding service will examine the header and decide where to forward the information to, based on the header information and a forwarding table. The forwarding table will indicate instances of the service on other network elements.

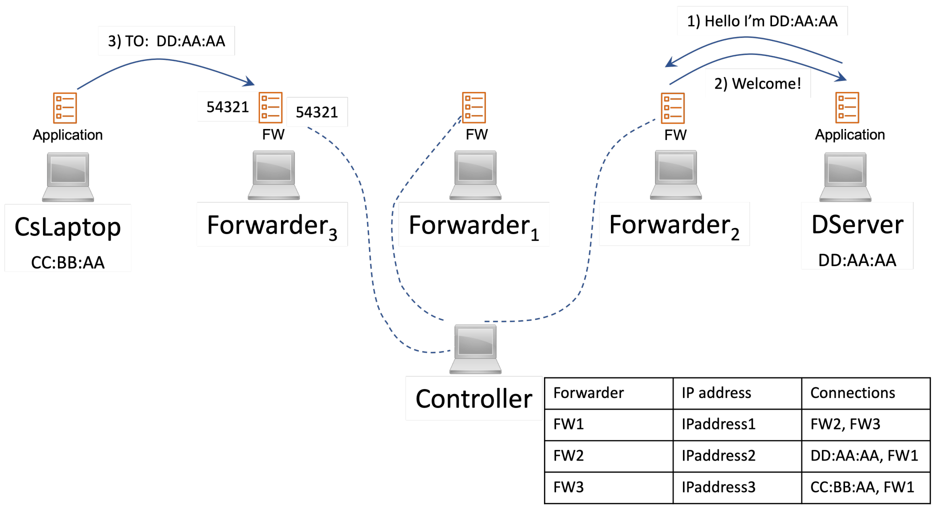
An application that intends to accept network traffic from the overlay will send a datagram to the forwarding service on the closest forwarding service, indicating that it intends to receive traffic for a given ID, e.g. 3 byte number[1]. The forwarding service will need to record the port number of that the application used and the string. Whenever datagrams arrive for the given ID, it should deliver these datagrams to the port number that is associated with the ID.



Dest	In	Out
DD:AA:AA		Forwarder 1

Figure 2: An application will transmit a header and payload to a forwarding service bound to port 54321 on the local host. The forwarding service will inspect the header and forward the header and payload to a forwarding service on another network element.

So, the basic functionality that a forwarding service has to provide is to accept incoming packets, inspect the header information, consult the forwarding table and forward the header and payload information to the destination. In the first place, if a destination is not known, a forwarding service would drop an incoming packet; once a controller has been implemented, the forwarding service needs to contact the controller to enquire about a forwarding information to the unknown destination and if it receives forwarding information, integrate this into its forwarding table.



Forwarder	IP address	Connections
FW1	IPaddress1	FW2, FW3
FW2	IPaddress2	DD:AA:AA, FW1
FW3	IPaddress3	CC:BB:AA, FW1

Figure 3: This example shows the forwarding of traffic with a label 'trinity' from an endpoint E1 to an endpoint E4. The names E1, E4, and R1 to R4 should be replaced by IPv4 addresses. The forwarding information should be supplied to the network elements by a central controller

In general, the network elements should be controlled by a central controller that initializes the flow tables of the network elements and would inform them about routes to destinations as the network changes. In a

[1]You could use 6 bytes or any length longer than 3 to identify an endpoint; I only choose 3 bytes to keep the example readable

first step though, in order to reduce complexity, the flow tables of the individual network elements should be hardcoded. The implementation of the controller element should only be pursued once the implementation of the forwarding functionality of the network elements is stable and allows for basic communication between two endpoints.
The final implementation should allow for a number of services to be started in the network of a cloud provider and a number of employees to use applications to communicate with these services at the same time i.e. your implementation of forwarding services needs to be able to handle to a number of concurrent flows originating from networks of end users and direct these flows to an from a number of services.
The OpenFlow protocol [5] is an example of a similar protocol that you could look at for inspiration of functionalities that may be interesting for you to include in your protocol. In the end, it is up to you what functionalities you implement in your protocol. In the deliverables, you need to describe these functionalities and justify why you included them in your protocol.
The following flow diagrams, figures 4 is an example of visualizations of network traffic between network elements. It shows a sequence of messages exchanged between the elements.

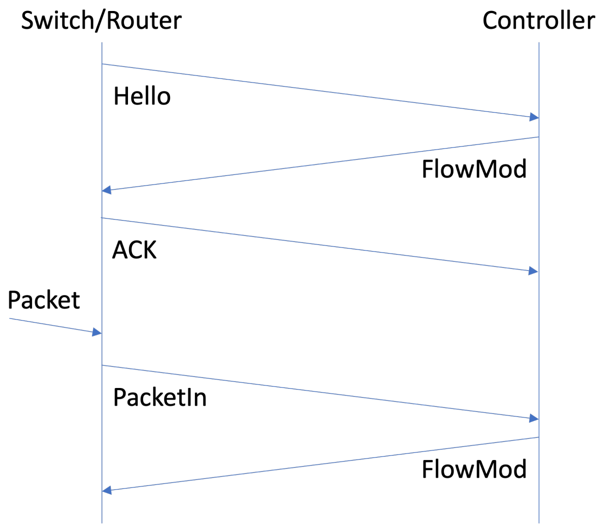


Figure 4: The diagram shows a router contacting the controller with an initial 'Hello' message and receives a modification message to its flow table. It replies to this with an acknowledgement. When a router receives a packet with an unknown destination, it will contact the controller and receive an additional modification to its flow table

Please use Flow Diagrams like these to document the communication between components of your implementation in your videos and in your final report.
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 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 hosts. There are a number of platforms that support the simulation of topologies or provide virtual infrastructures e.g. Docker [2], Mininet [6], Kubernetes [1], etc. For someone starting with socket programming and networking, I would suggest to use a platform such as Docker or Mininet; for someone already familiar with these concepts, I would suggest to implement their solution using Kubernetes. However, these are only suggestion and you need to make the decision how to implement your solution.
Possible topologies for Docker are shown in figure 5 and 6. Figure 5 shows a limited topology that can be used to test your implementation. The topology in figure 6 is more extensive and is shown to give you an idea to which extend you implementation should be stretched.

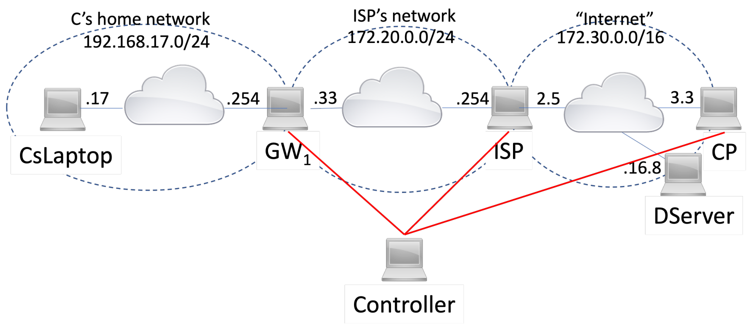


Figure 5: This figure shows a general topology that could be realised in Docker by setting up a number of containers and networks. This topology is kept to a minimum by showing a network of one employee, the network of a provider and a network for a cloud provider with an application server in the same network.

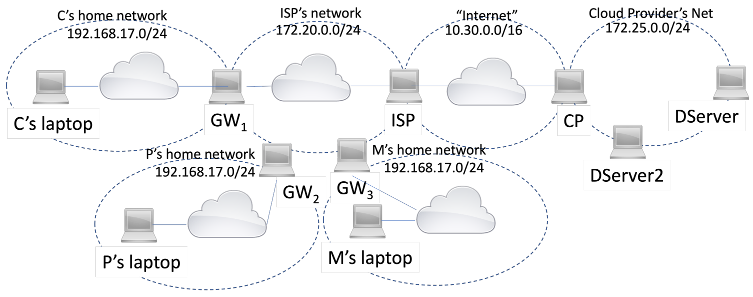


Figure 6: A more complex topology that woudl include a number of employees with their home networks and a number application servers in their own network

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

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 of 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 will 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 and the student number. 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 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] The Kubernetes Authors. Kubernetes project page. https://kubernetes.io, visited Sep 2021.

[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] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Computer Commununication Review*, 38(2):69–74, March 2008.

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