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

CSU33031 Computer Networks
Flow Forwarding

October 17, 2021

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

Your are given the task to develop a replacement for IPv4 or IPv6 by designing a protocol that forwards payloads based on a collection of strings that identify the source and destination of traffic. The forwarding information will be kept in forwarding tables located at the individual network elements and the content of these tables will be controlled by a central controller.

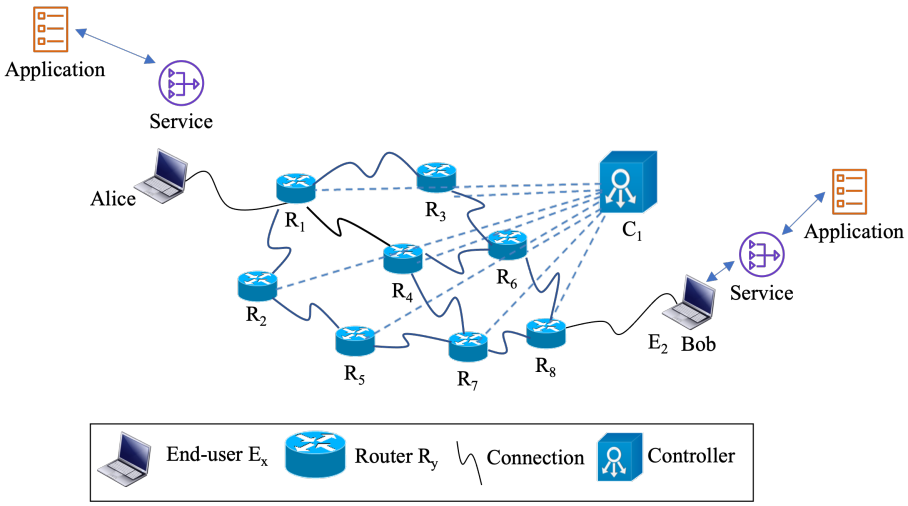

Figure 1: An application will send UDP datagrams to a forwarding service on the local host. The forwarding service will consult a table to determine from the header information of your protocol where to forward the datagram to. The table will provide it with the IP address of the network element that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

The initial implementation of the forwarding mechanism will use a UDP service bound to port 51510 at every network element. Someone in future may use the same information that is currently transferred using

UDP and send it, using Ethernet frames as a complete replacement to current IP protocols, as shown in figure 2 b) (see also the use of faces in NDN [1]).
An application using your protocol will create a header you design, attach the payload to be transmitted, and send it as payload to a service you develop, bound to port 51510 on the local host, such as shown in figure 2 a). The 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 receive traffic will send a datagram to the forwarding service on its local host, indicating that it intends to receive traffic for a given string. The forwarding service will need to record the port number of that the application used and the string. Whenever datagrams arrive for the given string, it should deliver these datagrams to the port number that is associated with the string.

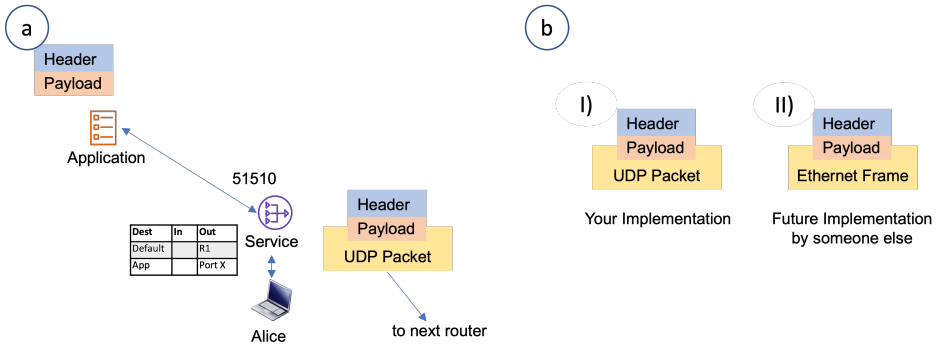

Figure 2: An application will transmit a header and payload to a forwarding service bound to port 51510 on the local host. The forwarding service will inspect the header and forward the header and payload to a forwarding service on another network element.

The header information should be encoded as type-length-value (TLV) format. Figure /refTLVExample shows an example where the first number of the encoding gives the type of the information that follows, the second number gives the length of the information and the third item of the field provides the value of the information itself. The first example shows a field that could represent a network ID, the network has the length of 7 characters and the value 'trinity'. The second example combines two network IDs by introducing a type combination that can hold any combination of further TLVs, in this case two network IDs. This example allows for construction of hierarchical networkIDs such as 'trinity.scss'.

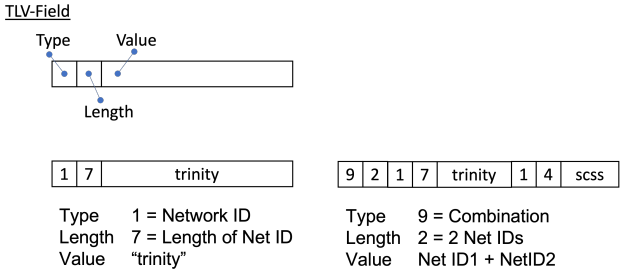

Figure 3: The 1st number of the encoding gives the type of the information that follows, the 2nd number gives the length of the information and the 3rd item of the field provides the value of the information itself. The 1st example shows a field that could represent a network ID, the network has the length of 7 characters and the value 'trinity'. The 2nd example combines two network IDs by introducing a type combination that can hold any combination of further TLVs, in this case two network IDs.

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.

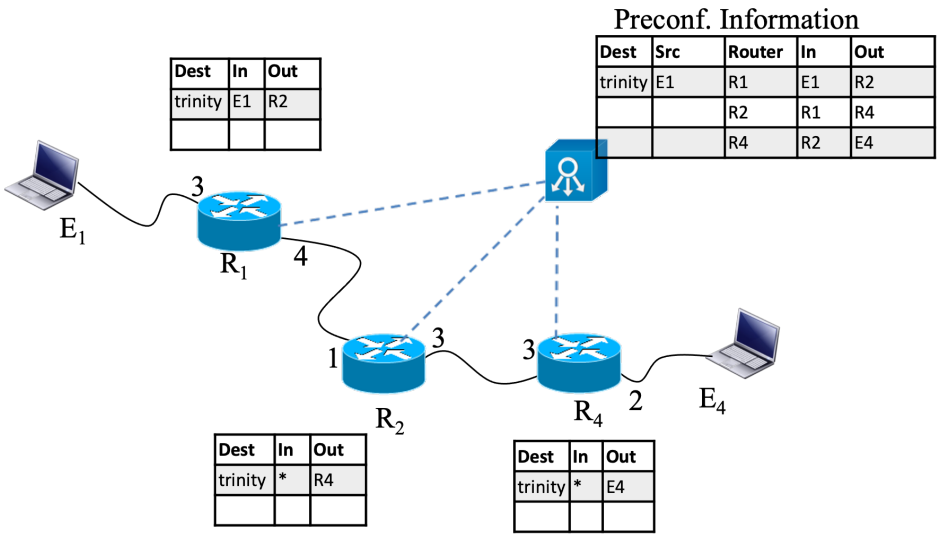

Figure 4: This example shows the forwarding of traffic with a label 'trinity' from an endpoint E1 to an endpoint E4. The names E1, E4, and R1 to R4 should be replaced by IPv4 addresses. The forwarding information should be supplied to the network elements by a central controller

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 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 Named Data Networking (NDN) protocol [1] and the OpenFlow protocol [6] are examples of similar protocols that you could look at for 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 5 is an examples of visualizations of network traffic between network elements. It shows a sequence of messages exchanged between the elements.
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 [3], Mininet [7], Kubernetes [2], 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.

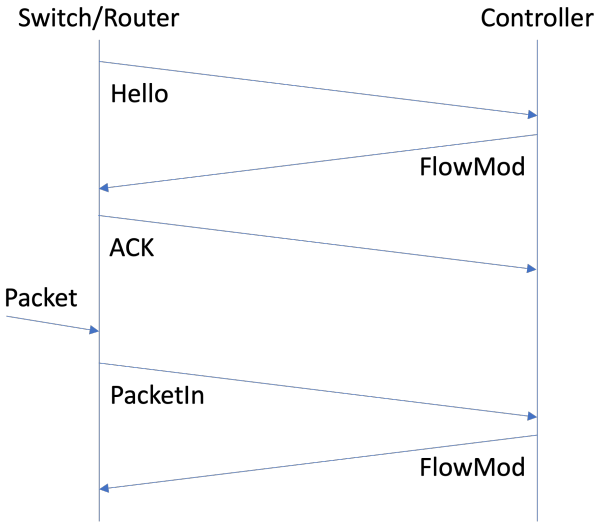

Figure 5: The diagram shows a router contacting the controller with an initial 'Hello' message and receives a modification message to its flow table. It replies to this with an acknowledgement. When a router receives a packet with an unknown destination, it will contact the controller and receive an additional modification to its flow table

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 [4]. Programs such as Wireshark [5], 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 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 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] Alexander Afanasyev, Xiaoke Jiang, Yingdi Yu, Jiewen Tan, Yumin Xia, Allison Mankin, and Lixia Zhang. Ndns a dns-like name service for ndn. In *In Proceedings of the 26th International Conference on Computer Communication and Networks(ICCCN)*, pages 1–9, July 2017.

[2] The Kubernetes Authors. Kubernetes project page. https://kubernetes.io, visited Sep 2021.

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

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

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

[6] 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.

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