Exercise for Lecture Software Defined Networking

Prof. Dr. David Hausheer
Julius Rückert, Leonhard Nobach

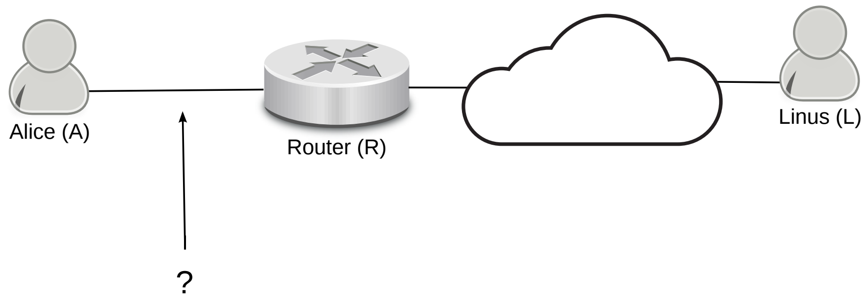
TECHNISCHE UNIVERSITÄT DARMSTADT

Winter Term 2015/16
Exercise No. 1
Published at: 20.10.2015
Submission exclusively via Moodle, Deadline: 27.10.2015 before 4pm
Contact: `[rueckert|lnobach]@ps.tu-darmstadt.de`
Web: `http://www.ps.tu-darmstadt.de/teaching/ws1516/sdn/`
Submission: `https://moodle.tu-darmstadt.de/enrol/index.php?id=6349`
– Example Solution –

Problem 1.1 - Cross-Layer Networking Basics

Note: This task is not directly related to the content in the slides, however you should be able to solve it with your previous knowledge of communication networks. Depending on your previous skills, it might require some online research. Nevertheless, the understanding of this topic will definitely help you with solving future exercises, labs, and exam questions related to SDN and OpenFlow!
Tip: Recap the following topics: the ISO/OSI layer model, Ethernet, MAC addresses, the Internet Protocol (IP, at least IPv4), ARP, ICMP, TCP (only the basics) and UDP.

Alice has a Linux PC. She connects to the Router via Ethernet, and her first action is to "ping" Linus's PC.



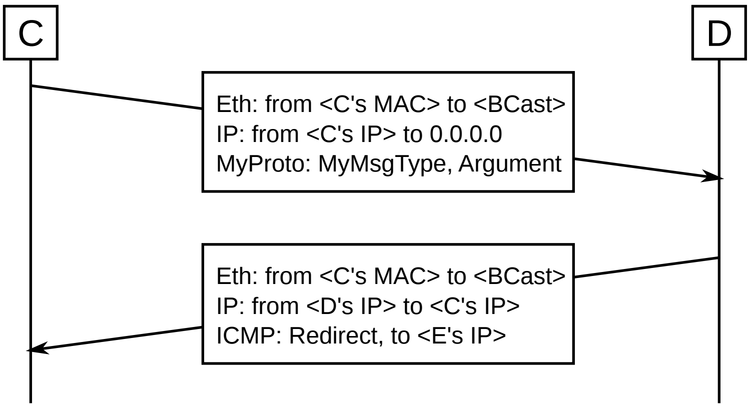
Assume the following:

- Alice uses no domain name, she just enters Linus's IP address for pinging
- Linus answers the ping, and the (Internet) connection between Linus and the Router is working as expected.
- Alice does not use DHCP, her IP address and default route was entered manually.
- No Layer 2 information between Alice and the router is known priorly.
- No other communication unrelated to the ping is taking place.

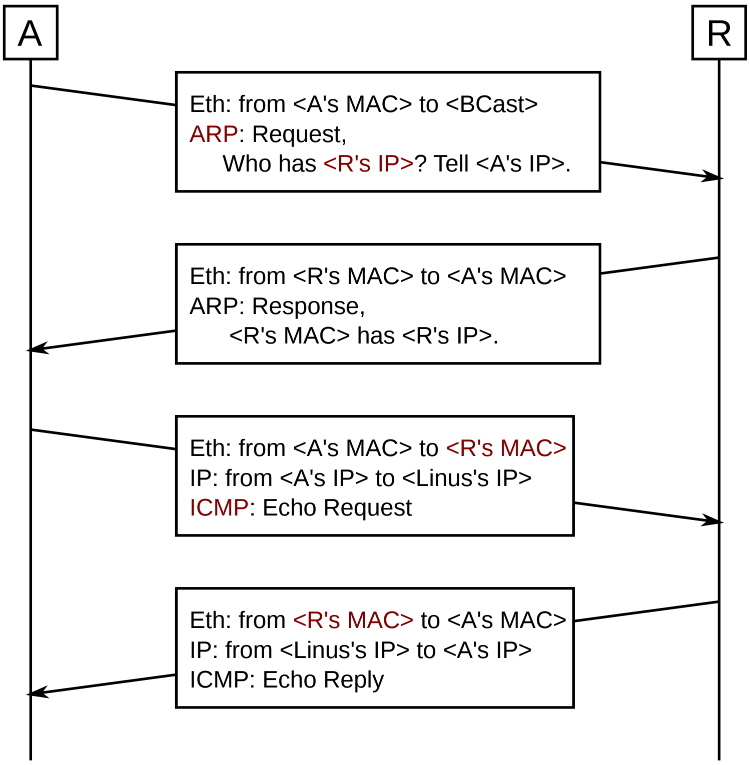
Draw the communication on the Ethernet link between Alice and the Router in a sequence diagram.

- Every message exchanged in your diagram shall contain all protocols of the ISO/OSI layer model that were used, starting with the data link layer.
- For every protocol, specify **data link and network addresses** that are used. As you don't know the actual adresses of anyone, you may write e.g. <IP address of X> or <X's IP> as a placeholder. For broadcast addresses, you also may use <BCast>.
- For the highest protocol used, also write down the message type/opcode that is used, e.g. ICMP **Redirect**.

Below is an **example** diagram, explaining the idea of the format that you shall use (Note: The example diagram makes no sense beyond explaining the *format*). Stop after the first ping answer has been received by Alice.



Solution:



- As Alice has statically configured her IP addressing information, she knows the IP address of her **default gateway**, which is the interior IP address of R.

- All traffic not belonging to the local subnet is commonly forwarded to this default gateway, so is the traffic for Linus. However, to start sending IP packets for Linus to R, she requires Layer 2 information about R (its MAC address).

- Alice therefore **sends an ARP request** to ask who has the default gateway's IP address, and gets an ARP response from R, containing R's MAC address.

- Alice now knows the MAC address of its default gateway, which is required for further IP communication.

- To communicate with Linus, Alice now sends a packet with Linus's IP address in the IP destination field, **but with R's MAC address** in the Ethernet destination field.

- Returning packets from Linus also always contain R's MAC address in the Ethernet source field, but Linus's IP address in the IP source field.

- *Ping* sends out ICMP Echo request packets. Linus therefore answers with an ICMP Echo Reply.

Problem 1.2 - Routing and Basics

a) Routing vs. Forwarding

Briefly explain the difference between **routing** and **forwarding**.

Solution:

•*Routing* (algorithm): A successive exchange of connectivity information between routers. Each router builds its own routing table based on collected information.

•*Forwarding* (process): A switch- or router-local process which forwards packets towards the destination using the information given in the local routing table.

b) Examples for Routing Algorithm Classes

Name two examples (each) of **link state** and **distance vector** routing algorithms.
(Note: You should know the abbreviations as well as the full names.)

Solution:

Link state:

•OSPF (Open Shortest Path First)

•IS-IS (Intermediate System to Intermediate System)

Distance vector:

•RIP (Routing Information Protocol)

•IGRP (Interior Gateway Routing Protocol)

c) Link State vs. Distance Vector in a Nutshell

Explain the main difference between the two routing algorithm classes.
(Limit your answer to a few sentences!)

Solution:

Example answer: "Link-state algorithms (also known as shortest path first algorithms) flood routing information to all nodes in the internetwork. Each router, however, sends only the portion of the routing table that describes the state of its own links. In link-state algorithms, each router builds a picture of the entire network in its routing tables. Distance vector algorithms (also known as Bellman-Ford algorithms) call for each router to send all or some portion of its routing table, but only to its neighbors. In essence, link-state algorithms send small updates everywhere, while distance vector algorithms send larger updates only to neighboring routers. Distance vector algorithms know only about their neighbors."

Source: `http://docwiki.cisco.com/wiki/Routing_Basics`

d) What is the name of the most widely used routing protocol that exchanges routes **between** autonomous systems? Do not abbreviate the name.

Solution: Border Gateway Protocol (Exterior Gateway Protocol would be okay, too.)

e) Distance Vector Routing: Count-to-Infinity Problem

Assume a simple distance vector routing protocol that only uses the "poison reverse" mechanism to avoid the count-to-infinity problem. Besides, assume for this task that in every round(/time step), every node sends a routing update to its direct neighbors. Please complete the routing table entries for the given topology after round 3 and 4 in the following scheme.

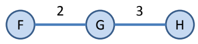
Solution: In general, poison reverse is only one mechanisms to tackle the count-to-infinity problem. In practice it is used together with other mechanisms, such as *split horizon*. Therefore, studying poison reverse alone can still lead to cases where the count-to-infinity problem is not completely solved, which is also an important lesson to learn in this task.

Poison reverse: “If a router A learns about unreachable routes through one of its interfaces, it advertises those routes as unreachable through the same interface.”
Source: https://www.juniper.net/techpubs/software/jseries/junos93/jseries-config-guide-basic/split-horizon-and-poison-reverse-efficiency-techniques.html.

Using this description, three basic cases can be derived for round 3 and 4, depending on when routers sends/receive updates during a round:
(1) the first case assumes that G receives F’s update before G generates and sends out an update itself. In this case, as we only use poison reverse, the count-to-infinity problem is not solved. False routes are further propagated.
(2) In the second case, one could assume (the other way round) that F receives G’s update before F generates and sends out and update itself. In this case, F receives the information on the unreachable route to H and, according to poison reverse, reports the unreachability back. Therefore, after round 3, the information already converged.
(3) In the third case, it is assumed that routing updates are send concurrently by F and G. This is a rather theoretical case but possible. Assuming that updates are always send out concurrently in all further rounds this leads to an unstable and still count-to-infinity-affected situation. Here, alternating, always one of the routers knows about the unreachability, while the other has an step-wise increasing hop count.

More cases could be constructed as combination of the three cases.

Round 1

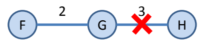


Node F		
Destination	Distance	Next node
G	2	G
H	5	G

Node G		
Destination	Distance	Next node
F	2	F
H	3	H

Node H		
Destination	Distance	Next node
F	5	G
G	3	G

Round 2



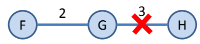
Node F		
Destination	Distance	Next node
G	2	G
H	5	G

Node G		
Destination	Distance	Next node
F	2	F
H	N.E.	-

Node H		
Destination	Distance	Next node
F	N.E.	-
G	N.E.	-

Case 1

Round 3

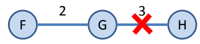


Node F		
Destination	Distance	Next node
G	2	G
H	9	G

Node G		
Destination	Distance	Next node
F	2	F
H	7	F

Node H		
Destination	Distance	Next node
F	N.E.	-
G	N.E.	-

Round 4



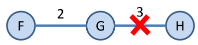
Node F		
Destination	Distance	Next node
G	2	G
H	13	G

Node G		
Destination	Distance	Next node
F	2	F
H	11	F

Node H		
Destination	Distance	Next node
F	N.E.	-
G	N.E.	-

Case 2

Round 3

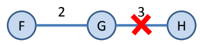


Node F		
Destination	Distance	Next node
G	2	G
H	N.E.	-

Node G		
Destination	Distance	Next node
F	2	F
H	N.E.	-

Node H		
Destination	Distance	Next node
F	N.E.	-
G	N.E.	-

Round 4



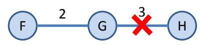
Node F		
Destination	Distance	Next node
G	2	G
H	N.E.	-

Node G		
Destination	Distance	Next node
F	2	F
H	N.E.	-

Node H		
Destination	Distance	Next node
F	N.E.	-
G	N.E.	-

Case 3

Round 3

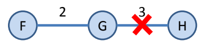


Node F		
Destination	Distance	Next node
G	2	G
H	N.E.	-

Node G		
Destination	Distance	Next node
F	2	F
H	7	F

Node H		
Destination	Distance	Next node
F	N.E.	-
G	N.E.	-

Round 4



Node F		
Destination	Distance	Next node
G	2	G
H	9	-

Node G		
Destination	Distance	Next node
F	2	F
H	N.E.	-

Node H		
Destination	Distance	Next node
F	N.E.	-
G	N.E.	-

Problem 1.3 - Load Balancing with OpenFlow (Case Study 1)

This task is a case study. You are supposed to demonstrate theoretical concepts defined in the lecture in an applied setting. Only the problem and its rough context is defined. The context may be extended, if necessary. You are intended to define processes and procedures to solve the problem. Your solution should be defined to an extent allowing a team of skilled staff to implement your solution, i.e., details may be omitted, if they do not have a large impact on your solution. The solution should be presented in a text-based form. Additional literature may be used and it is highly recommended to discuss your solution in a team.

Scenario and Setting:

Friendlist is a start-up social network whose user base increases from day to day. The engineers found that their single webserver cannot take the load of requests anymore. However, for several technical reasons, it is required that the website is reachable behind a single IP address. The engineers therefore plan to purchase a load balancer, besides new servers and a network switch to connect them to the load balancer.

Friendlist requests several quotes from different vendors. Most vendors offer a dedicated load balancer appliance, which is very expensive. You are a software developer preparing a quote for Friendlist, and you found out that if the switch supports OpenFlow, you can solve the problem without special load balancer hardware. Like common load balancers, your solution guarantees that the same customer IP address is always serviced by the same server to maintain sessions.

Briefly sketch your system architecture to implement the discussed mechanism. More elaborately, describe how the SDN controller behaves, and which OpenFlow rules are pushed to the switch.