Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Networked Systems
ETH Zürich — seit 2015

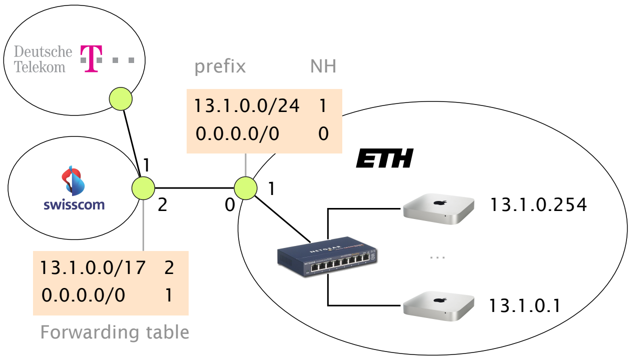
# Communication Networks

Prof. Laurent Vanbever

**Solution:** Exercise 5 – Forwarding & Routing

## 5.1 The Art of Defaulting Properly (Exam Style Question)

Consider this simple network configuration between ETH and Swisscom. Assume that ETH owns a large IP prefix 13.1.0.0/17, but only uses 13.1.0.0/24 to address its internal hosts. For simplicity, we assume that ETH and Swisscom operators configure their forwarding table statically and rely on the use of a default route (0.0.0.0/0).



Where are my IP packets going?

**a)** How many IP addressable addresses does ETH "own" in total?

**Solution:** $2^{(32-17)} - 2$

**b)** Give the first and last IP address that ETH can use for addressing a host.

**Solution:** 13.1.0.1 and 13.1.127.254

**c)** Suppose Swisscom receives a packet for 13.1.0.66 from Deutsche Telekom. What is the path taken by this IP packet?

**Solution:** Swisscom/1 → Swisscom/2 → ETH/0 → ETH/1

**d)** Suppose Swisscom receives a packet for 13.1.66.1 from Deutsche Telekom. What is the path taken by this IP packet?

**Solution:** Swisscom/1 → Swisscom/2 → ETH/0 → Swisscom/2 → ETH/0 → ...

**e)** What eventually happens to the packet for 13.1.66.1? As an attacker observing this, could you use this observation to congest the ETH-Swisscom link more easily? Explain why (or why not).

**Solution:** It will eventually be dropped as the TTL reaches 0. Permanent forwarding loops can be used to perform a Denial of Service (DoS) attack with few resources. Here an attacker can simply start sending fake traffic to 13.1.66.1 which will start “pilling up” on the Swisscom ↔ ETH link. The actual damages will depend on: *i)* the rate at which the attacker can send; *ii)* the TTL of the packets; as well as *iii)* the actual capacity of the link. Observe that the induced congestion negatively impact *all* traffic, including traffic destined to 13.1.0.0/24.

## 5.2 Detective work

You just started your first job as a network operator of a small network. To get more familiar with the network, you look at a packet trace captured at a switch. The trace contains packets from multiple hosts and one router connected by a (layer 2) switch. The router acts as default gateway, providing access to the Internet and is assigned the first IP address in the subnet: `179.168.8.1`. Each row in the following table represents one packet observed at the switch.

| SRC MAC Address | DST MAC Address | SRC IP Address | DST IP Address |
|---|---|---|---|
| 6a:00:02:49:a1:a0 | 11:05:ab:59:bb:02 | 179.168.11.1 | 179.168.8.2 |
| 6a:00:02:49:a1:a0 | da:15:00:00:01:11 | 179.168.11.1 | 179.168.16.1 |
| da:15:00:00:01:11 | 11:05:ab:59:bb:02 | 129.132.103.40 | 179.168.8.2 |
| 11:05:ab:59:bb:02 | 40:34:00:7a:00:01 | 179.168.8.2 | 179.168.15.254 |
| 11:05:ab:59:bb:02 | ac:00:0a:aa:10:05 | 179.168.8.2 | 179.168.9.99 |
| ac:00:0a:aa:10:05 | 01:05:3c:34:00:02 | 179.168.9.99 | 179.168.13.255 |
| 6a:00:02:49:a1:a0 | da:15:00:00:01:11 | 179.168.11.1 | 179.168.8.1 |

**a)** Can you identify all the hosts that are part of the local network?

**Solution:** The local hosts are all the sources and destinations that do not have to go through the default gateway (e.g., their MAC address is not replaced by the MAC address of the router):

- 179.168.11.1
- 179.168.8.2
- 179.168.9.99
- 179.168.15.254
- 179.168.13.255

**b)** Can you reconstruct the IP subnet used to address the hosts within that local network?

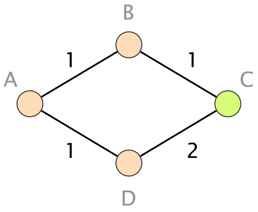
**Solution:** First, we should note, that the router MAC address is only used for IP sources or destinations outside the local subnet (router is used as gateway) or for packets from/towards the router. With this in mind, we can identify the lowest subnet address from the packets (179.168.11.1 -> 179.168.8.1) and (179.168.11.1 -> 179.168.8.2) as 179.168.8.1. Furthermore, we can infer that 179.168.15.254 still belongs to the local subnet (179.168.8.2 -> 179.168.15.254) but 179.168.16.1 is a destination outside of the network (179.168.11.1 -> 179.168.16.1). We can therefore identify the used subnet as 179.168.8.0/21.

## 5.3 Convergence (Exam Style Question)

Consider this simple network running OSPF as link-state routing protocol. Each link is associated with a weight that represents the cost of using it to forward packets. Link weights are bi-directional.

Assume that routers A, B and D transit traffic for an IP destination connected to C and that link $(B, C)$ fails. Which nodes among A, B and D could potentially see their packets being stuck in a transient forwarding loop? Which ones would not?

**Solution:** Nodes A and B could see their packets stuck in a forwarding loop if B updates its forwarding table before A, which is likely to happen as B would be the first to learn about an adjacent link failure. On the other hand, D would not see any loop as it uses its direct link with C to reach any destination connected beyond it.



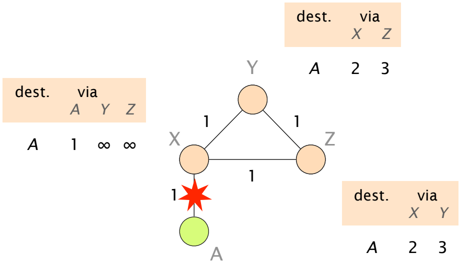
Loopy or not?

Assume now that the network administrator wants to take down the link $(B, C)$, *on purpose*, for maintenance reasons. To avoid transient issues, the administrator would like to move away all traffic from the link *before* taking it down and this, without creating any transient loop (if possible). What is the minimum sequence of increased weights setting on link $(B, C)$ that would ensure that *no packet* destined to C is dropped?

**Solution:** One example of a minimum sequence of weight settings is [1, 3, 5].

*Note:* The problem highlighted above happens because B shifts traffic to A before A shifts traffic to D, hence creating a forwarding loop. By setting the $(B, C)$ link weight to 3, (only) A shifts from using $(A, B, C)$ to using $(A, D, C)$. Once A has shifted, it is safe to shift B by setting the link weight to 5 (or higher). Once B has shifted has well, the link can be safely torn down.

## 5.4 Convergence with Poisoned Reverse



| dest. | via A | Y | Z |
|---|---|---|---|
| A | 1 | ∞ | ∞ |

| dest. | via X | Z |
|---|---|---|
| A | 2 | 3 |

| dest. | via X | Y |
|---|---|---|
| A | 2 | 3 |

Consider the network on the left which uses distance vector routing with poisoned reverse. Each link is associated with a weight that represents the cost of using it to forward packets. Link weights are bi-directional.

Assume that the link between X and A fails (as shown in the figure) and use the table below to show the first 8 steps of the convergence process. How many steps does it take until the network has converged to a new forwarding state? Explain your observations.

**Solution:** The network does not converge as the maximum link weight is increased by one in each round ("count to infinity problem"). Poisoned reverse does not solve the problem of counting to infinity if three or more nodes are involved. One possible workaround is to define ∞ as a small value (e.g. ∞ := 16).

**Solution:**

| | X | | | Y | | Z | |
|---|---|---|---|---|---|---|---|
| dst=A | via A | via Y | via Z | via X | via Z | via X | via Y |
| $t = 0$ before the failure | 1 | ∞ | ∞ | 2 | 3 | 2 | 3 |
| $t = 1$ after X sends its vector | ★ | ∞ | ∞ | ∞ | 3 | ∞ | 3 |
| $t = 2$ after Y sends its vector | ★ | 4 | ∞ | ∞ | 3 | ∞ | ∞ |
| $t = 3$ after Z sends its vector | ★ | 4 | ∞ | ∞ | ∞ | ∞ | ∞ |
| $t = 4$ after X sends its vector | ★ | 4 | ∞ | ∞ | ∞ | 5 | ∞ |
| $t = 5$ after Y sends its vector | ★ | ∞ | ∞ | ∞ | ∞ | 5 | ∞ |
| $t = 6$ after Z sends its vector | ★ | ∞ | ∞ | ∞ | 6 | 5 | ∞ |
| $t = 7$ after X sends its vector | ★ | ∞ | ∞ | ∞ | 6 | ∞ | ∞ |
| $t = 8$ after Y sends its vector | ★ | 7 | ∞ | ∞ | 6 | ∞ | ∞ |

Add the distance vectors to this table