Something

IN3260

Oskar Haukebøe

December 4, 2023

1 Introduction

Network problems can often be obscure and challenging to diagnose for end users. When experiencing problems, it can be difficult to know where the problems emerged. The bottleneck could be at the local WiFi, or it could be that one is just accessing a distant server, and therefore is experiencing slowdowns.

One project which aims in helping to detect whether the bottleneck is on the local WiFi, or somewhere else is NETHINT [1]. This project aims to passively listen to all WiFi traffic and to be able to compare the latencies and packet losses for all devices on the WiFi network, as described in [2]. By doing this, it should be able to help detect where the bottleneck is.

In this assignment, the aim is to verify whether the data collected by NETHINT can determine whether the bottleneck is on the local WiFi. This is done by using a TEACUP [3] testbed to perform automated experiments on actual machines with different network configurations.

2 IN-PROGRESS The testbed

In order to test whether the NETHINT program can determine where the bottleneck is situated, it is necessary to have it listen to various traffic with the bottleneck at different places. For this reason, we have set up the testbed as shown in Figure **??**. This represents a situation where there is one user Client 2 who is streaming a video from Server 2, and another user Client 1 who is doing something else that is consuming bandwidth from Server 1. The goal is to be able to determine whether the traffic between Server 1 and Client 1 is affecting the traffic between *Server 2* and Client 2.

All nodes in Figure fig:topology are physical computers running Debian Linux, except for the Switch, and are being controlled by another computer not in the figure, using TEACUP. This makes it possible to automate the generation of network traffic, the throughput, and delay at the routers, as well as logging the network traffic.

We are using a switch between the two clients and the first router in order to have the clients both be accessed through the same network interface at Router 1 when running over the wired connection. This allows us to easily limit the bandwidth to both clients at the router and have them share the bandwidth.

The WiFi AP is also connected to the switch and hosts an unencrypted WiFi network. The network is unencrypted because the NETHINT program does not support decrypting the network the WiFi traffic. The two clients are connected to this WiFi

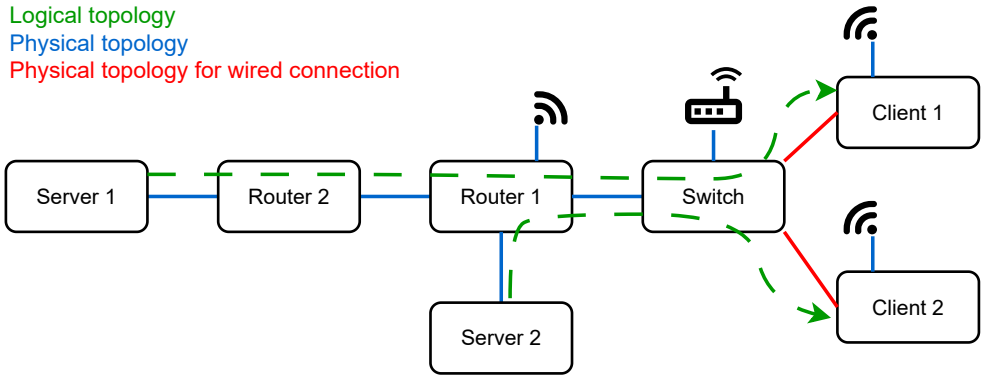

Figure 1: Topology of the testbed

network and they can reach the two Servers over it. The WiFi card at Router 1 is set to monitor mode and is used by NETHINT to listen to all the WiFi traffic between the clients and the WiFi AP. In order for the router to listen to the WiFi traffic, the WiFi interface must be set to use the same channel/frequency as the router. Both the WiFi interface at Router 1 and the WiFi AP have therefore been set to use channel 3.

In order to tell the machines how to reach each other, they had to be configured to know where the packets should be sent. As an example Client 2 got configured as follows in the file `/etc/network/interfaces.d/vlan11-iface`:

```
auto enp36s0
iface enp36s0 inet static
      address 172.16.12.5/24
      up route add -net 172.16.10.0 netmask 255.255.255.0 gw
         ↪ 172.16.12.254 || true
      up route add -net 172.16.11.0 netmask 255.255.255.0 gw
         ↪ 172.16.12.254 metric 10 || true
      up route add -net 10.10.12.0 netmask 255.255.255.0 gw
         ↪ 172.16.12.254 metric 10 || true
```

where `172.16.12.254` is the IP address of the network interface of Router 1 facing the switch. `172.16.11.0` is the IP address of Server 2, and `10.10.12.0` is the IP interface of Server 1. This essentially tells the client that it can reach the two servers through Router 1. The address `172.16.10.0`, which is the interface on Router 1 facing Router 2, had to be added for TEACUP to run as it first wants some of the computers/interfaces to ping each other, and so it had to be possible for Client 2 to ping this interface on Router 1.

The configuration above sets up the routes for running the tests over the wired connection. But we also wanted to run the tests over a wireless connection, which is why `metric 10` is also specified in the configuration above.

As the two clients now can reach the servers through both the WiFi AP and through the cables, they need to be configured to use the correct network interface. This makes it possible to run the command

```
sudo ip route add 172.16.11.0/24 via 172.16.13.1 metric 2
```

on the client in order to send the traffic over the router, which has IP address `172.16.13.1`, instead of over the cable. Similarly, one can then run

```
sudo ip route del 172.16.11.0/24 via 172.16.13.1 metric 2
```

in order to use the wired connection again.

2.1 The bottlenecks

The bottlenecks are placed at either Router 1, Router 2, or at the WiFi AP. This corresponds to the two users at Client 1 and Client 2 either having a common bottleneck or not having a common bottleneck. When the bottleneck is at Router 2, the user at Client 2 should not be affected too much, as Router 1 should still have room for more bandwidth than is being used.

In order to configure the bottlenecks, we change the throughput at Router 1 and Router 2. The throughput at the WiFi AP is set to the lowest value it supports which is 54 Mbps. The values at Router 1 and Router 2 are then set either higher than this or lower, depending on where the bottleneck should be. The values used at the routers for the three different bottleneck configurations are listed in Table **??**.

Table 1: Throughput at the two routers in the different bottleneck configurations

Bottleneck at:	Router 1 (Mbps)	Router 2 (Mbps)
Router 1	15	70
Router 2	70	15
WiFi AP	70	70

2.2 Traffic

We emulate VoIP traffic, for which we send 20 UDP packets per second with a packet size of 100 bytes (this mimics Skype, which will use TCP when UDP does not work and was found to send at roughly this rate and packet size with occasional outliers [4]).

References

[1] P. J. Barhaugen, "NETwork Home INTerference." May 08, 2023. Accessed: Nov. 30, 2023. [Online]. Available: `https://github.com/petternett/NETHINT`

[2] P. Juterud Barhaugen, "Home Network Interference Detection with Passive Measurements," University of Oslo, 2023.

[3] S. Zander and G. Armitage, "CAIA Testbed for TEACUP Experiments Version 2," 2015.

[4] M. Mazhar Rathore, A. Ahmad, A. Paul, and S. Rho, "Exploiting encrypted and tunneled multimedia calls in high-speed big data environment," *Multimed tools appl*, vol. 77, no. 4, pp. 4959–4984, Feb. 2018, doi: 10.1007/s11042-017-4393-7.