Lost in Simulation: Route Property in Mininet

Philipp Kern, Henning Stubbe*, Kilian Holzinger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: philipp.kern@tum.de, stubbe@net.in.tum.de, holzinger@net.in.tum.de

***Abstract*—The Mininet network emulator enables the comparison of speed, delay, jitter and packet loss across different topologies. It provides a Python API to instantiate almost arbitrary layouts of networks and connections with attributes like predetermined packet loss. We inspect linear and grid-like topologies and discover that both share similar performance characteristics.**

Setting up more paths from one host to another does not improve latency noticeably.

Emulating more consecutive switches in a network decreases it's throughput which would not be expected in a hardware implementation.

***Index Terms*—software-defined networks, measurement, hight-speed networks**

1. Introduction

Many fields require dependable behaviour in their network infrastructure. To analyze how different setups influence important properties, a network emulator like Mininet [1] can be used. Mininet is a program to test and deploy network configurations and therefore provides Software-defined Networking. While it is not possible to learn how real world hardware devices react in all situations this way, the most significant attributes can be mirrored in an emulated network through placement of nodes or link properties. We want to describe important properties of network routes and measure the impact different network configurations have on them.

Using Mininet A running Mininet instance can be controlled using a command line interface. It can be configured by command line arguments, but for our investigation we are using the Python API. Shell commands can be executed from the viewpoint of each node in the network. In this research we are using iperf3 and ping to garner information on interesting properties of specific topologies. Mininet can also be instructed to drop some packets to test the resilience of TCP/IP in overloaded network connections.

2. Related Work

In [2], Torres-Jr and Ribeiro researched how packet reordering influences TCP throughput and they established *Mean Displacement* and *Entropy* metrics as simple, universally applicable basis on which to compare network behaviour.

3. Properties of Routes commonly considered by the community

Certain internet connectivity properties are considered advantageous and in some cases critical for a variety of fields.

Mininet's Limitations The properties of hardware components and physical cables are difficult to predict before they are used, which makes them hard to account for in a simulation. On the other hand, we can - with little effort - introduce certain factors like latency, bandwidth, results of having multiple paths to choose from and consequences of overloading like packet loss in Mininet.

Delay Delay is also known as lag, ping rate and latency according to the IR Team [3], who also wrote [4] about jitter and [5] about packet loss. It is important to keep delay to a minimum for playing real time online video games as movement is often precisely timed. And video conferencing with low latency enables a more natural and spontaneous conversation of all participants.

Bandwidth Network bandwidth, commonly referred to as speed, is measured in $\mathrm{bit\,s^{-1}}$. Operating a file server benefits from high bandwidth in order to provide service to many users in a given time. It also enables users to stream higher quality video from streaming platforms.

Speed Ramp-up and Consistency Speed ramp-up is likely less important in many areas than bandwidth and latency consistency, but becomes crucial in serving small web pages quickly. Websites smaller than $5\,\mathrm{MB}$ could not benefit from a high bandwidth connection that reaches its peak after transmitting $10\,\mathrm{MB}$ and being significantly slower before that. Live-streaming services additionally need consistency in bandwidth to serve their users a certain quality of video without needing to change quality often. This behaviour depends mostly on the sender's TCP implementation, namely the pace at which it enlarges its congestion window at slow start as explained in [6] by Carle et al.

Packet Loss Packet loss occurs when network hardware receives more packets than it can handle, which it will then discard requiring the sender to resend them [5]. In online speech communication it is important to keep the need for re-sending packets to a minimum in order to avoid delays. Time sensitive fields are especially

doi: 10.2313/NET-2023-06-1_03

dependant on low packet loss rate.

Packet Reordering To speed up traffic flow through parallelization related internet packets are sometimes split up and later arrive in the wrong order at the client. Reordering of packets can have similar consequences for video and audio calls over the internet as packet loss: voice recordings not arriving in time makes for a choppy listening experience. Additionally it strains the receiving hardware to some extent. Ghasemirahni et al. saw an increase in web server performance by manually putting packets back in order in [7].

Jitter Jitter is the variance in ping time. If that variance is too high, voice over IP communications might sound stuttered or drop out [4].

4. Deploying Topologies in Mininet

In this section we provide a basic introduction on how to deploy networks with Mininet. Command line arguments make the deployment of basic topologies easier when compared to Python scripts. A small linear topology can be created like this `sudo mn --topo=linear,4`. It contains four hosts, each being connected to one switch which are in turn connected linearly.

For the measurements conducted in this research however just using the command line is too limiting. Using the Python API directly gives control over what program is run on the nodes, how exactly they are connected and which properties they have. This gives the additional advantage of being able to parameterize the topologies more easily and automate testing. The topologies are deployed entirely within one Python script available at [8]. Three different topologies were implemented and two of them tested, all containing two hosts:

- A linear one that places a certain number of switches between the hosts which differs from the preset variant that has hosts at every switch on the line
- One with switches connected in a grid-like fashion
- A circle of switches, two of which connect to the hosts

It is possible to set the length of the linear and circular topology and the width and height of the grid-like one.

If hosts are to be used as routers in Mininet, they have to be manually configured. That is, they have to be instructed to pass on IP traffic and use specific IP addresses. However, hosts do not act as routers so connections cannot be established across them, even if they are set to forward IP traffic (see 'hostline' and 'hostangle' topologies in [8]).

5. Emulating Environments for Testing Properties

To evaluate how different network configurations affect the requirements listed in 3, we now introduce custom topologies in Mininet. The topology and test code can be found in [8]. By launching a Wireshark instance from one of the switches or hosts it is possible to inspect packages going through the respective node. To find out the achievable bandwidth and speed ramp-up we use iperf3. The goal of this is to test how the availability of many different paths affect the speed and continuity of packet flows. As such, the circular topology mentioned in 4 is too similar to the other ones and evaluating it would not yield meaningful results.

5.1. Linear Topology

To identify the impact of having to simulate a large amount of switches, we construct a topology consisting of a variable amount of switches connected in a line (Figure 1). At both ends of this line we attach a host expecting them to be able to communicate with each other. We scale this network up by adding more switches into the line, parameterizing the network by the length of this connection. When measuring and comparing to the other topology, we refer to the amount of links from the first to the second host as the path length.

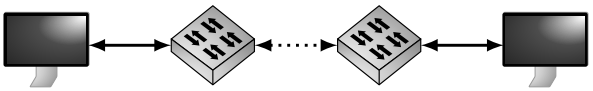

Figure 1: Linear Topology

5.2. Square Topology

Switches are placed in a two-dimensional grid and each is connected with the ones left, above, right and below itself (Figure 2). This is helpful for finding out whether alternative paths through switches can have an impact on relevant properties and how having to emulate many network devices strains the emulating system. In order for this to work at all, the Spanning Tree Protocol has to be enabled on all switches. The top left and the bottom right switch are each additionally connected to one of the two hosts. Here we define the minimum path length as the lowest number of links data, e. g. an Ethernet frame has to pass through to reach its destination which is the other host.

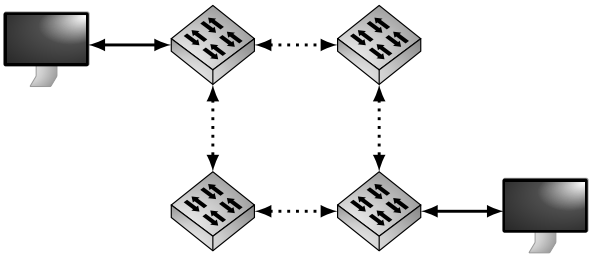

Figure 2: Square shaped Topology

5.3. Tools Used

For measuring delay, jitter and packet loss the `ping` [9] tool is used. The first ping request is sent separately from another ten as to not disturb the average times recorded from those. Bandwidth and speed ramp-up measurements are done with `iperf3` [10].

doi: 10.2313/NET-2023-06-1_03

6. Impact of Parameters on Route Properties

The following tests were conducted on an Intel® Core™ i5-2520M 2.50GHz processor using the Linux Kernel version 5.19.8. The test computer has 8 GB of DDR3 RAM clocked at 1333 MHz and was running a graphical desktop environment and the VSCode IDE at the time of measurement. Although existent, the additional system load was not influential to the results as it consists mostly of background services and amounting to only about four percent CPU utilization.

6.1. Delay

To mimic high distances between communicating parties which results in higher latency we can insert a variable amount of nodes in between the two. It was found that the the additional switches did not help or hinder the ping times.

The ping command is suitable here: it displays the time it takes to send packets back and forth between clients. In general, the larger the network the more time it needs to pass on the ICMP ECHO_REQUEST datagram sent by the ping command. To compensate for this behaviour a long pause is interjected in between a first ping command, the output of which is discarded, and the consecutive *real* ones. As well as multiple routes or paths to send packets between clients, round trip times and achievable bandwidth were measured.

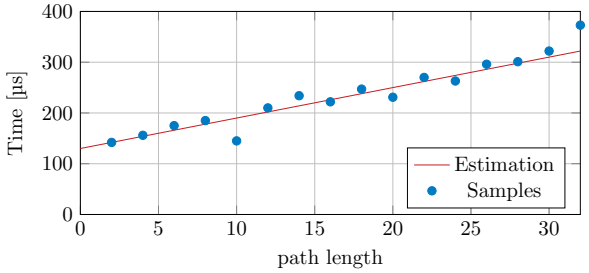

Figure 3: Average ping time in a linear topology

It appears that round trip times scale linearly with the amount of switches between the ping participants in Figure 3.

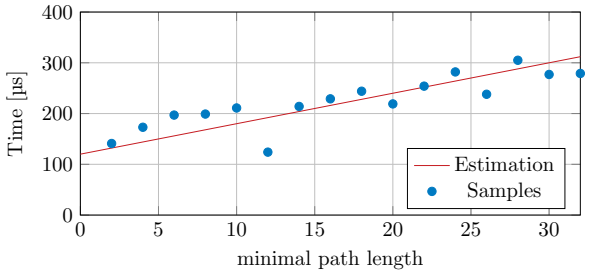

Figure 4: Average ping time in a square topology

When using a square topology with same height and width we can analyze the round trip times as well. Note that the minimal amount of links the packet has to traverse is double the amount of switches per dimension in this case. It can be observed that the additional switches in the network have not contributed to longer ping times. In fact the square shaped network turned out to be slightly faster than the linear one by on average $0.011\,63\,\mathrm{ms}$. In rough terms the formulas

$$d_l(x) = 0.13 + 0.006 \cdot x\mathrm{ms} \quad (1)$$

$$d_s(x) = 0.12 + 0.006 \cdot x\mathrm{ms} \quad (2)$$

present themselves as an estimate of the round trip time of linear and square topologies on the researcher's computer, where x is the amount of links between two hosts.

6.2. Speed Ramp-up and Consistency

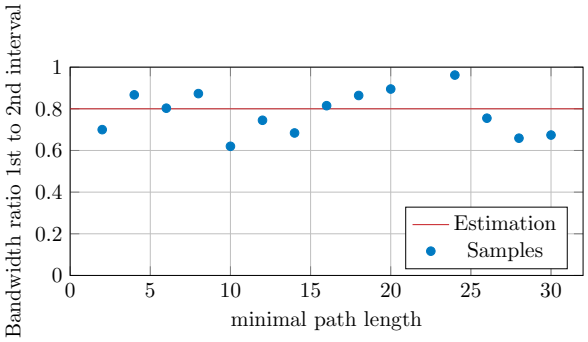

Figure 5: Speed of first 0.1 s interval relative to second interval

This test was performed in square shaped grid-of-switches-topology. Apart from two exceptions at minimum path lengths of 22 and 32, the bandwidth achieved in the first $0.1\,\mathrm{s}$ was always in the realm of $60\,\%$ to $100\,\%$ of the bandwidth of the second such interval (Figure 5). As a result, no significant delay would be noticeable by the user.

6.3. Packet Loss

Mininet has a setting for links to drop zero through 100 percent of packets going through. In the test scenario

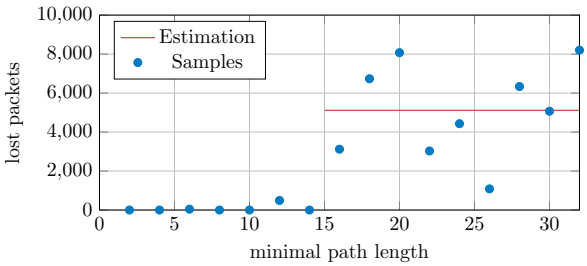

Figure 6: Packet loss in a square topology

all links were set to drop 0% of packets, but packet loss still occurred (Figure 6). When having to emulate a minimal path length of 16 or more links, on average 5119 packets were lost. This is possibly due to the emulating system not being able to handle more than a certain amount of switches. The irregularity of the specific amounts of lost packets also points toward that explanation, as system load by other applications is not uniform over time. With a correctly installed network, this should not happen in real

doi: 10.2313/NET-2023-06-1_03

hardware networks. The linear topology never exceeded the critical number of nodes, in this test between 52 and 66, and thus only seldomly lost a few hundred packets in two test rounds. Again other processes are to blame here.

6.4. Packet Reordering

Introducing multiple route options (e. g. hosts connected as a grid) could have led to reordered packets, but hosts were unable to connect to ones only reachable via other hosts in these tests.

6.5. Jitter

The first ping response takes longer to arrive, potentially due to having to initialize an IP connection between the hosts. Therefore, we begin our analysis of jitter at the second request, letting `ping` calculate the standard deviation of ten requests.

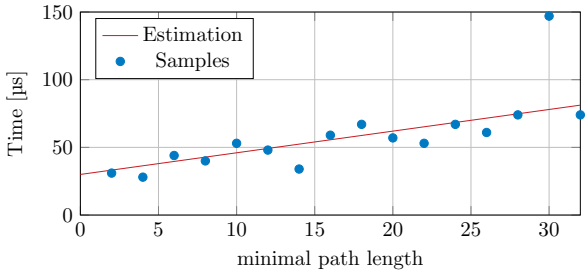

Figure 7: Standard deviation of ping in a linear topology

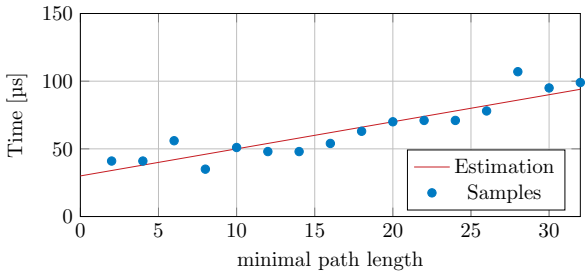

Figure 8: Standard deviation of ping in a square topology

Figures 7 and 8 indicate a linear increase in jitter with a growing number of switches with the linear topology presenting slightly more consistent round trip times.

6.6. Bandwidth

Similarly to the delay measurements, exchanging switches for hosts could have revealed changes in speed that result from having to route IP traffic in contrast to just passing on Ethernet frames. As the latter could happen without any intervention and therefore even delay in the simplest real-world case. The results of the iperf benchmarking hints toward an inversely proportional relation between path length and achievable bandwidth. The speed losses at higher node counts is best attributed to the overhead of simulation and should not have real world equivalences as switches operate independently.

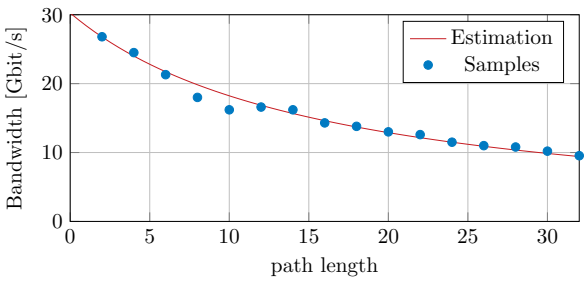

Figure 9: Average bandwidth in a linear topology

It seems that, looking at Figures 9 and 10, the availability of different paths consisting of switches from one host to another again has no great impact on speed. That is to expected, considering the simulated switches are likely incapable of rerouting traffic to avoid congestion - they are not routers after all.

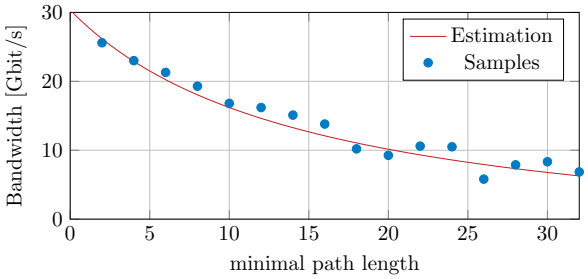

Figure 10: Average bandwidth in a square topology

The slightly better performance of the linear topology in contrast to the square one is best explained by the lower CPU load of fewer switches to simulate.

$$b_l(x) = 500 \cdot (x+16)^{-1} - 1\text{Gbit}\,\text{s}^{-1} \quad (3)$$

$$b_s(x) = 530 \cdot (x+15)^{-1} - 5\text{Gbit}\,\text{s}^{-1} \quad (4)$$

Formulas (3) and (4) estimate the expectable speed in $\text{Gbit}\,\text{s}^{-1}$ for linear and square, topologies. However, these are not the only possibilities to predict the bandwidth, as e. g. $b_{s2}(x) = 42 - (10 \cdot \ln(x+3))\text{Gbit}\,\text{s}^{-1}$ also matches the datapoints in Figure 10 closely.

7. Conclusion and Future Work

In summary, Mininet is a tool powerful in configuration options and yet simple to spin up for quick tests. Some difficulties arose from a sparsely annotated documentation of its Python API. On the other hand, by "running real kernel, switch and application code" [1], it offers the same tools as the host machine on all virtual nodes, which proved immensely helpful.

In this research we

- summarized important network properties
- presented a methodology to scale up networks for testing some of those properties
- benchmarked linear and square-shaped topologies with increasing size.

It turns out that the the ping and jitter properties of Mininet networks are reflective of hardware based networks, but that simulations sometimes lead to unrealistic

behavior. Throughput slowdown and sudden increase in packet loss are two examples for this which we found out in our research.
To find out more about certain behaviour of a networking infrastructure, Mininet supports the use of (external) controllers for topologies deployed inside it. In fixed-size topologies it would be feasible to make hosts act as routers by setting a default gateway for them manually. That allows for complex behaviour analysis of packet reordering and bandwidth and ping-time implications.

References

[1] M. P. Contributors. (2022) Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet. https://mininet.org/. [Online; accessed 5-September-2022].

[2] P. R. Torres-Jr and E. P. Ribeiro, "Packet Reordering Metrics to Enable Performance Comparison in IP-Networks," *Journal of Computer Networks and Communications*, vol. 2020, p. 8, 2020.

[3] I. Team. Network Latency - Common Causes and Best Solutions. https://www.ir.com/guides/what-is-network-latency. [Online; accessed 23-September-2022].

[4] ——. Network Jitter - Common Causes and Best Solutions. https://www.ir.com/guides/what-is-network-jitter. [Online; accessed 23-September-2022].

[5] ——. What Is Network Packet Loss? https://www.ir.com/guides/what-is-network-packet-loss. [Online; accessed 23-September-2022].

[6] G. Carle, S. Günther, M. Simon, and M. Sosnowski, "Grundlagen Rechnernetze und Verteilte Systeme (GRNVS)," 2022.

[7] H. Ghasemirahni, T. Barbette, G. P. Katsikas, A. Farshin, A. Roozbeh, M. Girondi, M. Chiesa, G. Q. M. Jr., and D. Kostić, "Packet Order Matters! Improving Application Performance by Deliberately Delaying Packets," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. Renton, WA: USENIX Association, Apr. 2022, pp. 807–827. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/ghasemirahni

[8] P. Kern. (2022) Lost in Simulation. https://gitlab.lrz.de/00000000014AB2A9/lost-in-simulation. [Online; accessed 25-September-2022].

[9] M. Muuss. (1983) ping. https://github.com/iputils/iputils. [Online; accessed 24-September-2022].

[10] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, and K. Prabhu. (2014) iperf3. https://software.es.net/iperf/. [Online; accessed 24-September-2022].