Department of Informatics
Technical University of Munich

TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

MASTER'S THESIS IN INFORMATICS

Efficient and Accurate Hop-by-Hop Capacity Estimation

Bakar Andguladze

Technical University of Munich

Department of Informatics

Master's Thesis in Informatics

Efficient and Accurate Hop-by-Hop Capacity Estimation

Effiziente und Genaue Hop-by-Hop Kapazitätsabschätzungen

Author:	Bakar Andguladze
Supervisor:	Prof. Dr.-Ing. Georg Carle
Advisor:	Simon Bauer, M. Sc.
	Benedikt Jaeger, M. Sc.
Date:	November 15, 2021

I confirm that this Master's Thesis is my own work and I have documented all sources and material used.

Garching, November 15, 2021

Location, Date

Signature

Abstract

Abstract of the thesis will be written afterwards

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This master thesis presents the capacity estimation method for hop-by-hop measurements. Our goal is to create and test a method of calculating the capacity (i.e. maximum transmission rate) in a given network path and locating the narrow link. We will test the developed tool in regard to various metrics, such as packet size, intrusion rate, cross-traffic and flow-interference and finally, analyze the test results in order to conclude whether our approach is actually capable of delivering accurate results efficiently.

1.1 Motivation

As the Internet is becoming increasingly essential part of our day-to-day lives, it is ever more important for the Internet providers to enhance the quality of networks in order to create a better user experience. One of the means to achieve this goal is to have a better picture of the network they aim to improve. Therefore, we are going to create a tool that measures the capacity of the path between two hosts. The intended field of application is, for instance, enhancing the performance of network, traffic analysis, network monitoring, etc.

There are quite a few capacity estimation methodologies and tools available, that will be discussed in chapters below. However, the current State-of-the-Art methods have some significant flaws and limitations regarding our measurement goals, such as

- High intrusion, which can lead to network overload,
- Dependence on ongoing traffic, which can be unpredictable at times,
- Inability to locate the narrow link on the path.

Therefore, our goal is to develop a new solution that tries to minimize the influence of these limitations in regard to our measurement objectives. We will try to implement the least possible intrusion without compromising the accuracy. Also our tool will be able to find the first narrow link of the path by measuring the capacity of each hop in the network.

This new solution will be tested and evaluated in comparison to the results of existing capacity measurement tools, i.e. PPrate implementation by Patryk Brzoza[x], but in contrast to Brzoza's passive approach our tool will be based on an active measurement methodology. Moreover Brzoza's measurement tool was designed to estimate end-to-end capacity, while this thesis is concerned about measuring the capacity of each hop in the network and finding the narrow link, as hop-by-hop measurements provide a better picture of a network and enable to take a closer look at potential issues.

1.2 Research Questions

This thesis is supposed to answer the following research questions:

- **How to measure network capacity hop-by-hop?**
  In order to measure the path capacity hop-by-hop, we need to estimate capacities to each router on the path until the destination host is reached.
- **How to optimize the trade-off between accuracy and intrusiveness regarding large-scale measurements?**
  Certain level of intrusion into the network will be necessary for the measurements. However there is an important factor to consider: too high intrusion could disrupt the traffic in the network and too low could lead to unreliable results. Therefore an optimal middle ground has to be found: What will be the optimal amount of packets to send to each router to get correct results?
- **How robust is the proposed solution regarding the handling of cross-traffic, flow-interference and sudden path parameter changes?**
  Real networks are usually quite complex and different challenges might arise when we are trying to measure the path capacity. We need to find out whether our solution is feasible when it faces cross-traffic, flow-interference or when the path parameters suddenly change.
- **Are we able to locate the capacity bottlenecks of a network?**
  We are interested to find the location of the weakest link in the given network. This can be achieved by finding the capacities to each hop. The weakest link will

1.3 Outline

This section introduces the structure of the thesis.
Chapter 2 defines the necessary terminology for understanding this thesis. Namely, capacity, TCP, ICMP, Raw sockets, etc.
Chapter 3 describes the related work - what has been done in regard to capacity estimations and some drawbacks and limitations to the existing approaches
Chapter 4 describes the approach and the tool that we have developed to implement it.
Chapter 5 reviews the test setup and test environment in which the experiments are conducted.
Chapter 6 evaluates our approach based on several parameters. It reviews the different factors that might affect the measurements and to what extent. Namely, how packet length influences the measurement results, what is the optimal rate of intrusion in the network and whether the cross-traffic and the flow-interference cause higher inaccuracy. Based on how the proposed approach handles these challenges we can state whether it is reliable or not.
Finally, chapter 7 concludes our thesis and subsequently discusses the future work - what can be done afterwards to further extend our methodology.

Chapter 2

Background

The following chapter provides the background information and defines the important terminology that is useful to understand our work.

2.1 Terminology

Certain terms in our area of research can be used with different meanings, therefore we first need to state that this thesis will be using the definitions provided by Prasad et al. **prasad** in their paper "Bandwidth estimation: metrics, measurement, techniques and tools", as it appears to be more widespread and accepted.

Prasad et al. **prasad** introduce the following three metrics: capacity, bandwidth and bulk transfer capacity(BTC), capacity being the main focus of our work. Moreover, they distinguish between segments and hops. The former being the link at the data link layer (L2) and the latter - the links at the IP layer (L3).

[THIS IS A COPY] A segment normally corresponds to a physical point-to-point link, a virtual circuit, or to a shared access local area network (e.g., an Ethernet collision domain, or an FDDI ring). In contrast, a hop may consist of a sequence of one or more segments, connected through switches, bridges, or other layer-2 devices. We define an end-to-end path from an IP host (source) to another host (sink) as the sequence of hops that connect to

Capacity

[ALSO COPY] A layer-2 link, or segment, can normally transfer data at a constant bit rate, which is the transmission rate of the segment. For instance, this rate is 10Mbps on

a 10BaseT Ethernet segment, and 1.544Mbps on a T1 segment. The transmission rate of a segment is limited by both the physical bandwidth of the underlying propagation medium as well as its electronic or optical transmitter/receiver hardware. At the IP layer a hop delivers a lower rate than its nominal transmission rate due to the overhead of layer-2 encapsulation and framing. Specifically, suppose that the nominal capacity of a segment is

The transmission time for an IP packet of size bytes is

Available Bandwidth

Another important metric is the available bandwidth of a link or end-to-end path. The available bandwidth of a link relates to the unused, or "spare", capacity of the link during a certain time period. So even though the capacity of a link depends on the underlying transmission technology and propagation medium, the available bandwidth of a link additionally depends on the traffic load at that link, and is typically a time-varying metric.

2.2 OSI Model

2.3 TCP

https://datatracker.ietf.org/doc/html/rfc793

2.4 ICMP

https://datatracker.ietf.org/doc/html/rfc792

2.5 Traffic Generation

iPerf

Raw Sockets

https://www.binarytides.com/raw-sockets-c-code-linux/

`https://www.wireshark.org/docs/man-pages/tshark.html`

2.6 Capacity Estimation

general explanation

Chapter 3

Related Work

3.1 Existing Approaches

Active vs Passive End-to-End vs Hop-by-hop

Variable Packet Size Probing
Packet Pair/Train Probing
PPrate

Chapter 4

Implementation

4.1 Analysis

4.2 Approach

4.3 Traffic Generation

4.4 Estimation of End-to-End Capacity

4.5 Estimation of Capacities Hop-by-Hop

parameters(?)

4.6 Locating the Narrow Link

Chapter 5

Evaluation

This chapter presents the results of the evaluation of our approach. The experiments have been conducted based on several test parameters and their combinations and we will be analyzing the baseline parameter sets as well as key combinations.

We evaluate based on the path length, capacity range, packet size, number of packets. For the better insight we also apply artificial packet loss and icmp rate limit. The tests are conducted on the empty network as well as with significant amount of cross-traffic load.

The following parameters must be passed to the program via the config file:

- `topo_size` - The number of routers between the source host and the destination host.
- `capacity_range` - The range of numbers from which a capacity for each link will be generated
- `packet_size` - The size of each packet (in bytes) that will be sent from the source host to the destination.
- `packets_per_hop` - The amount of packets that will be targeted at each hop
- `icmp_ratelimit` - The artificial ICMP rate limit that will be assigned to each router. The default value is 0.
- `packet_loss` - The artificial packet loss that will be assigned to each router. The default value is 0.

- `cross_traffic` - The coefficient of the cross traffic load. The Optimal value is recommended to be assigned from 0 to 1.0, the value of 0 meaning an empty network without any cross traffic.

5.1 Test Environment and Setup

For the testing purposes we decided to build a virtual network using the network emulator - Mininet[1]. It provides the necessary tools to create an artificial network and has a very handy Python API that enables us to build custom topologies.

The figure below represents the template of the network that we have used in our experiments. It consists of the source and destination hosts, the top and bottom hosts and routers that connects them with each other.

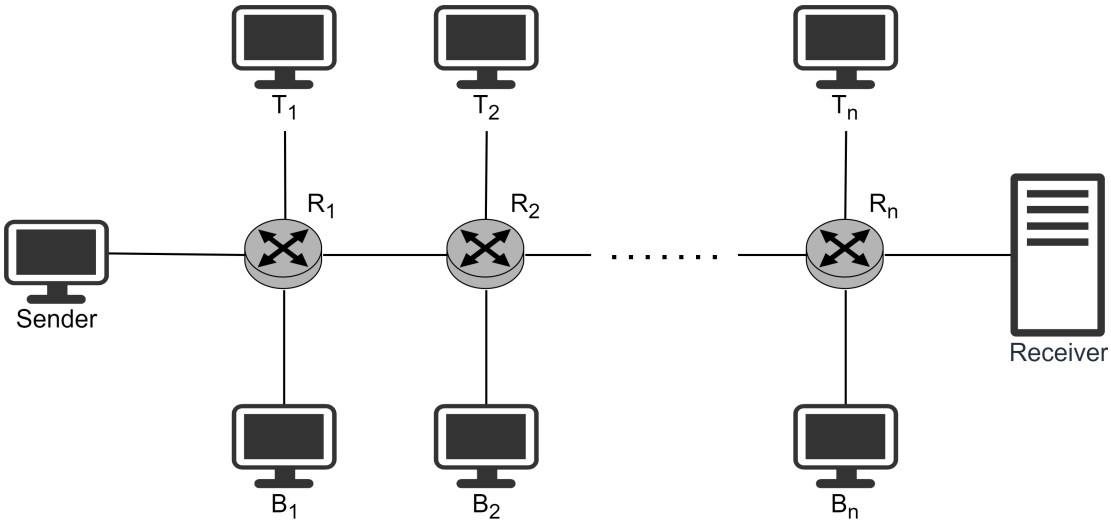

Figure 5.1: The template network topology

As a reader can see from the Figure 5.1, the main hosts (sender and receiver) are connected with a sequence of `n` routers, each consisting of four interfaces and each having its own subnetwork. The number of routers and, therefore, the number of top and bottom hosts is dynamic and configurable. The routes from and to all hosts are statically configured.

The measurements are conducted based on the main path, which is the one that connects the sender to the receiver. Top and bottom hosts are used for generating cross-traffic in later experiments. Cross-traffic interferes with the main flow of packets and has a significant influence on the accuracy of capacity estimations.

The measuring node is "Sender", as using our tool only makes sense if the measuring node is the one that generates the traffic. All the necessary traffic is captured at this node with the `tcpdump` command and subsequently analyzed in order to deliver the result.

Finally, it is also important to mention that Mininet has certain limitations, namely, it is limited with the computational power of the machine it runs on. Therefore we have decided to execute the estimation experiments on the testbed of the chair. Each set of parameters have been tested 20 times to ensure the reliable results.

5.2 Evaluation in Empty Network

5.2.1 Path Length

The first parameter to be discussed is the path length from the source host to the sink, i.e. number of hops on the path. In order to analyze the impact that a path length can have on the measurement accuracy, we have to execute the test runs on different values of the given parameter.
We decided to conduct measurements on arbitrary numbers of routers, namely 3, 8, 20 and 32. They paint a good picture of how our estimation tool reacts on networks with different sizes.

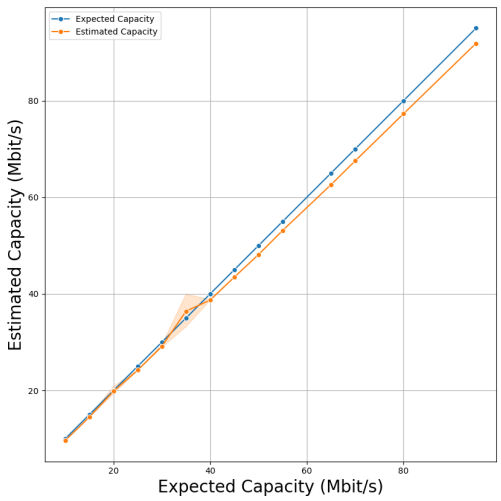

Figure 5.2: Path length: 3 Routers

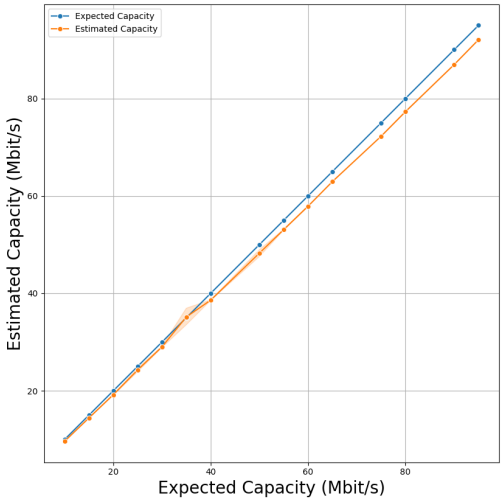

Figure 5.3: Path length: 32 Routers

Conclusion

We can conclude that the path length doesn't affect the accuracy of the estimation tool. After testing the path length we decided to take 8 routers as a default path length for

Path Length	Standard Deviation	Average	Min Error	Max Error
3	3.74%	4.78%	1.78%	19.18%
8	3.45%	4.94%	0.4%	20.17%
20	2.93%	4.17%	0.64%	28.79%
32	2.45%	4.21%	0.11%	20.23%
63	4.06%	5.02%	0.02%	24.97%

Table 5.1: Error Statistics of paths with different lengths

the posterior estimations
Notes
gets weirdly accurate around 35-40
description of table is required

5.2.2 Packet Size

One of the key parameters in our evaluation is the packet size. As the packet size along with the amount of packets that have to be injected into the network can be the cause of the network overload, we have to find an optimal amount of intrusion so that we don't have to sacrifice the accuracy.

Based on the test results we were able to observe that the optimal packet size depends on the capacity range of the network.

Estimation Error

Conclusion

5.2.3 Train Length

Conclusion

5.2.4 Optimal Intrusion

5.2.5 Capacity Range

One of the most important parameters is the capacity range of the links that we conduct our measurements on. After testing the tool on several different networks with varying capacity ranges, such as [10, 100], as well as [500, 600] and [1000, ..] it appears that our methodology has a significant limitation when measuring the networks that contain links with relatively high capacities.

Conclusion

In conclusion, we can deduce that the accuracy of the tool greatly depends on the capacity range. We can argue that on an empty network the accuracy is good for the paths that have the capacity value up to XXX Mbits. We expect that the tool accuracy when measuring the links with capacities over XXX will be less reliable during cross-traffic. In other words the accuracy threshold will be lower. We will discuss this in the section about cross-traffic.

5.2.6 Packet Loss

5.2.7 ICMP Rate Limiting

Provide the information about icmp rate limiting and then how it affects your experiment results

5.2.8 Summary

5.3 Evaluation during Cross-Traffic

We have conducted the previous experiments on empty networks without any cross-traffic. However this is not the case in the real Internet that we use on daily basis. Cross traffic seems to affect the test results significantly. What does 1.0 mean?

5.3.1 Path Length

5.3.2 Packet Size

5.3.3 Train Length

5.3.4 Optimal Intrusion during Cross-Traffic

5.3.5 Capacity Range

5.3.6 Packet Loss

5.3.7 ICMP Rate Limiting

5.3.8 Combination of Cross-Traffic, Artificial Packet Loss and ICMP Rate Limiting

5.3.9 Summary

5.4 Data Replication

One option to try to see if it can help

Chapter 6

Conclusion

In this thesis we have implemented a hop-by-hop capacity estimation methodology and evaluated it based on various parameters.

The following sections will shortly summarize our work throughout the timespan of the thesis and address all the concerns and questions that we had in the beginning.

6.1 Evaluation Results

Sum up whether this method can actually be used in practice

6.2 Answers to the Research Questions

After the careful and thorough research we can already answer the questions we were aiming to address in the beginning of this thesis:

- **How to measure network capacity hop-by-hop?**
- **How to optimize the trade-off between accuracy and intrusiveness regarding large-scale measurements?**
- **How robust is the proposed solution regarding the handling of cross-traffic, flow-interference and sudden path parameter changes?**

- **Are we able to locate the capacity bottlenecks of a network?**
  The answer to this question greatly depends on the accuracy of the hop-by-hop estimation. As we have seen in the Evaluation chapter, there are situations when we get accurate results and there are cases when the results are somewhat inaccurate or not reliable at all. In case of the reliable results, the answer will be 'yes'. We can locate the capacity bottlenecks based on the first smallest capacity value from the sequence of results.

  There is a catch, however: if the path consists of multiple hops and there are several bottlenecks, we are able to locate only one, namely the first one. For example, if there are 15 hops on the path and the bottleneck is located on the first hop, but the capacities of several other links equals to that of the first link, we will not be able to see it. In such situations, we can only assume that the subsequent capacities are more than or equal of the capacity of the first hop.

6.3 Future Work

As our evaluation has shown, there is a lot of room for improvement for our methodology.
First and foremost, we have not conducted any test runs on a real network, which can be done in order to see how our tool handles the real traffic in the Internet.
Moreover, further research is required to handle the inaccuracy problem with cross-traffic as it is the key factor when it comes to the real-life usage of our framework.

Chapter A

Appendix

This chapter describes the source code repository and provides the information and instructions of how to use, alter and/or improve the current version of the Hop-by-hop measurement framework.

A.1 The Source Code Repository

The source code repository for this thesis is located on the GitLab instance belonging to the TUM. The `ma-andguladze/App` directory contains the whole source code.

The tool consists of the following files:

- `PPrate.py` - The pprate algorithm implementation by Patryk Brzoza[2]
- `TrafficGenerator.c` - As the name suggests, this program generates TCP traffic from one host to another. It takes the IP addresses of two hosts as arguments and creates the traffic via raw sockets. The file should be compiled before the first run.
- `mininet_topo.py` - Builds an instance of the network in Mininet considering all the necessary parameters passed, conducts the traffic generation, captures the traffic and saves the results in pcap files.
- `prepare_test.py` - Contains several helper functions, most notably the config file parser and the capacity generator.
- `process_tcp_csv.py` - Converts the Calculates the end-to-end capacity of the network. Implementation by Brzoza[2]

- `process_icmp_csv.py` - Converts the `icmp.pcap` file into csv and delivers the hop-by-hop capacity estimations based on the latter
- `run_test.py` - The main file that bootstraps the tool and runs the experiments in one setting. It requires a `JSON` configuration file to be passed as an argument with the help of which it is possible to manipulate several key parameters necessary for testing.

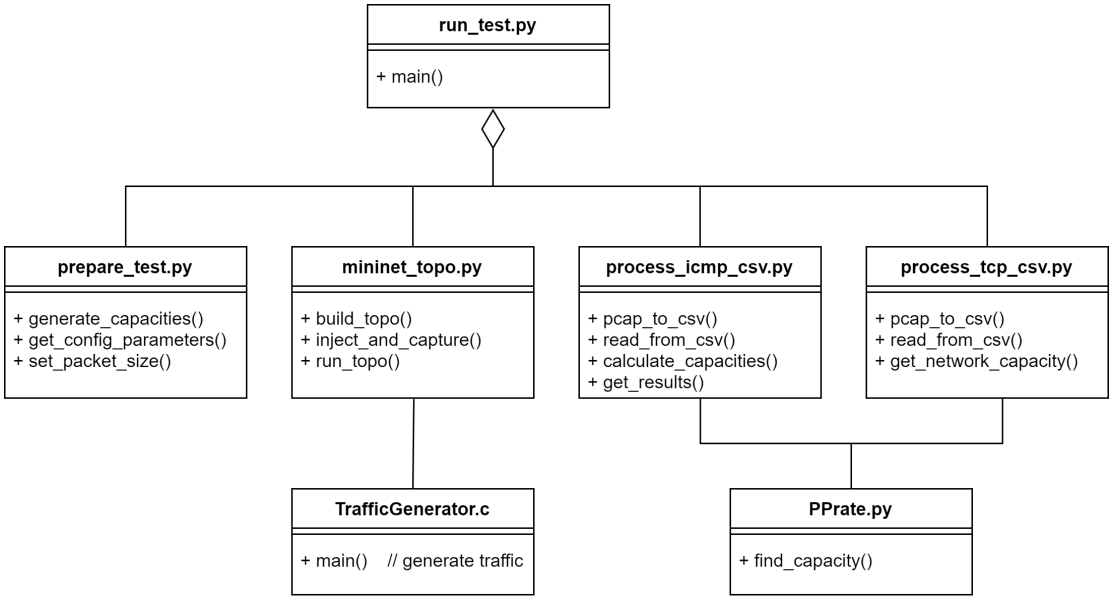

Figure A.1: The architecture of the framework

A.2 Setup

In order to run the test measurements with our tool, several dependencies and software should be installed on the Linux machine.

The experiments conducted by us were run only on Mininet - a network emulator tool which creates a realistic virtual network of virtual hosts, switches, controllers and links and runs a real kernel and application code[1]. To run the experiments on Mininet, it is necessary to install the emulator first. It is highly recommended to install Mininet from the source GitHub repository instead of packages, as the Mininet homepage[3] states that packages could give an older version.

Mininet installation:

```
1  $ git clone git://github.com/mininet/mininet
2  $ mininet/util/install.sh [options]
```

The default branch normally will install the latest stable version, however in case a user wishes to have any other version, it also possible to check out the branch with the following script before running `install.sh`:

```
$ cd mininet
$ git tag  # list available versions
$ git checkout -b mininet-2.3.0 2.3.0  # or any desired version from the list
$ cd ..
```

Note that Mininet must run as a root.

As for the rest, the following dependencies are required to be installed to run experiments:

- pandas
- NumPy
- SciPy
- tshark
- iPerf

For setting up these and compiling the `TrafficGenerator.c` program, run `install.sh` script in the `ma-andguladze/App` directory.

A.3 Running Tests

The measurements must be run with a `JSON` config file containing all the parameters that are interesting and important for our measurements. Manipulating these parameters gives us a better picture about strengths and weaknesses of our approach.

The following script executes the experiment:

```
$ sudo python run_test.py <config file>
```

The following `JSON` serves as an example of a configuration file used for experiments. It contains default parameter values, each of which will be manipulated respectively in upcoming sections:

```
{
  "topo_size": 3, // path length
  "capacity_range": [10, 100],
  "capacity_delta": 5
  "packet_size": 1400,
  "packets_per_hop": 300,
  "icmp_ratelimit": 0,
  "packet_loss": 0,
  "cross_traffic": 0.0,
```

```
  "output": "results/results.csv"
}
```

Chapter B

List of acronyms

Bibliography

[1] *Mininet Home*, `http://mininet.org//`, [Online; accessed 2021].

[2] P. Brzoza, "Implementation and Evaluation of Passive Capacity Estimation", B.Sc. Thesis, Munich, Germany, 2019.

[3] *Mininet Installation*, `http://mininet.org/download//`, [Online; accessed 11.2021].