Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Bandwidth Limit Enforcement for SCIONLab

Bachelor Thesis

Manuel Meinen

Advisors: Prof. Dr. Adrian Perrig, Mr. Matthias Frei
Department of Computer Science, ETH Zürich

Abstract

Today's society is heavily dependant on the internet and therefore also on it's underlying architecture. However, protocols like IP[1] and BGP[2] were not designed to withstand the threads that the internet is facing today. To address this issue, a research group around Prof. Adrian Perrig invented a novel internet architecture named SCION, which stands for *Scalability, Control and Isolation on Next-Generation Networks*.
For ISPs[3], research institutions or in general administrators of ASes[4] to test out SCION, the Network Security Group of ETH Zurich manages a distributed testbed called SCIONLab. Customers of SCIONLab can customize and download the configuration for setting up a SCION-AS. This SCION-AS connects then to one of ETH's Attachment Points and can therefore communicate via the SCIONLab infrastructure.
At the moment there are no bandwidth limitations in place other than the physical ones, meaning that neither the SCIONLab administrators nor the customers can set any upper limits for the bandwidth they have available between the User-AS and the SCIONLab infrastructure. However, such an upper limit is desirable for both customers and ETH for different reasons. Customers might want to test out the behaviour of a certain SCION based application when having limited bandwidth available. And ETH, which pays for the bandwidth SCIONLab is using, has an interest in enforcing an upper bandwidth limit to gain control over the expenses they have.
Since SCIONLab is an overlay network, meaning that whatever looks like a physical link from the perspective of a SCIONLab-AS is in fact a connection over the traditional IP based internet, bandwidth limitations per link can be enforced on an IP-level using existing tools.
The goal of this bachelor thesis project is to design and implement an automated mechanism to enforce a per IP-connection bandwidth limit between the User-ASes and the Attachment Points of SCIONLab. This is realized by using a tool called TC[5], which is part of the iproute2 utility collection.

[1] Internet Protocol
[2] Border Gateway Protocol
[3] Internet Service Providers
[4] Autonomous Systems
[5] Traffic Control

Contents

Introduction

Chapter 1

SCION/SCIONLab

1.1 SCION

SCION is a new internet architecture invented by Prof. Adrian Perrig and his fellow researchers. It's goal is to deliver scalability, control and isolation on next-generation networks. SCION differentiates between a control plane and a data plane, which are completely separated from each other to increase robustness. Failures in the control plane therefore don't immediately affect the availability of the data plane and vice versa.

1.1.1 Control Plane

Since SCION is an internet architecture it is designed to replace both IP and BGP and therefore fundamentally redesign the internet on OSI[1]-Layer 3 (Network Layer) and 4 (Transport Layer). Like the traditional internet, SCION is organized in Autonomous Systems. But unlike the traditional internet, SCION organizes multiple ASes in so called Isolation Domains. Per ISD[2], ASes have the same Trust Root Configuration.

Path Construction

Path construction in SCION happens in two phases, path exploration and path registration. In the path exploration phase the core-ASes send out PCBs[3]. The ASes can then decide to whom they want to forward the PCB. Each AS can then decide over which paths they want to be reached and register these paths in the path servers of the core-ASes. These processes happen both within an ISD as well as across ISDs. The final path is then constructed out of three partial paths, an up-path, a down-path and a core

[1] Open Systems Interconnection
[2] Isolation Domain
[3] Path-Segment Construction Beacons

path. How this final path is constructed is up to the AS itself, which gives it control over the path a certain package takes to reach it's destination. For further details I recommend reading chapter 7 of *SCION: A Secure Internet Architecture* [1]

1.1.2 Data Plane

The data plane consists mainly of path combination and data forwarding. Each AS possesses multiple up-paths. The down-paths are registered in the path servers of the core-ASes of the destination's ISD and can be requested by the AS. The core paths are stored in the path servers of the core-ASes. Therefore to get the possible paths to reach a destination AS, the source AS chooses an up-path to reach one of its core-ASes and requests the paths to one of the destination's core-ASes. Then it contacts the destination's core-AS to request the down-paths to the destination AS. Out of these three path segments the source AS can then construct the final path. By doing so it can also take short cuts over peering-links or if the destination AS lies on the up-path then it doesn't even need to contact any core-AS.
More details on how paths can be combined can be found in chapter 8 of the SCION-book [1]

1.1.3 Joining a SCION Network

For an AS to join a SCION network is quite easy. All the AS has to do is setting up at least one path server, one beacon server, one certificate server and one SCION border router. Then it buys connectivity from an AS that is already in the ISD that the AS wants to join as well. By joining the ISD the AS accepts the TRC configured for this ISD.

1.1.4 Topology

1.2 SCIONLab

SCIONLab is the distributed testbed for SCION. Most of the network is managed by the Network Security Group of ETH Zurich. The ASes that are not managed by ETH Zurich are called user-ASes. Each of them is typically attached to one of the few Attachment Points that are part of the network infrastructure managed by ETH. The APs are the only ASes that allow direct connections to user-ASes.
The SCIONLab network consists of multiple ISDs, most of which are grouping together ASes based on their geographic location or membership of a political union. However, one ISD has the purpose of building a backbone for the entire SCIONLab network. It is heavily interconnected and is hosted on Amazon Web Services.

1.2.1 SCIONLab as an Overlay Network

Each SCIONLab-AS is based on a Ubuntu-16.04 machine. Either it is a Virtual Machine or a dedicated SCION system. Each SCIONLab-AS has to at least have the following services available:

1. A **Beacon Server** that sends and receives the PCBs
2. A **Path Server** that stores the path segments and disseminates them to the customers.
3. A **Certificate Server** that holds the certificates which are used to validate the paths.
4. A **Border Router** that routes (on a SCION-level) the traffic leaving the AS.

As shown in figure 1.1 the Border Router is also responsible for wrapping the SCION-traffic into IP-traffic. This is simply done by setting up an IP-connection to the corresponding Attachment Point and then send the SCION packets over that connection. It is important to note that the Border Router doesn't do any routing on an IP-level.

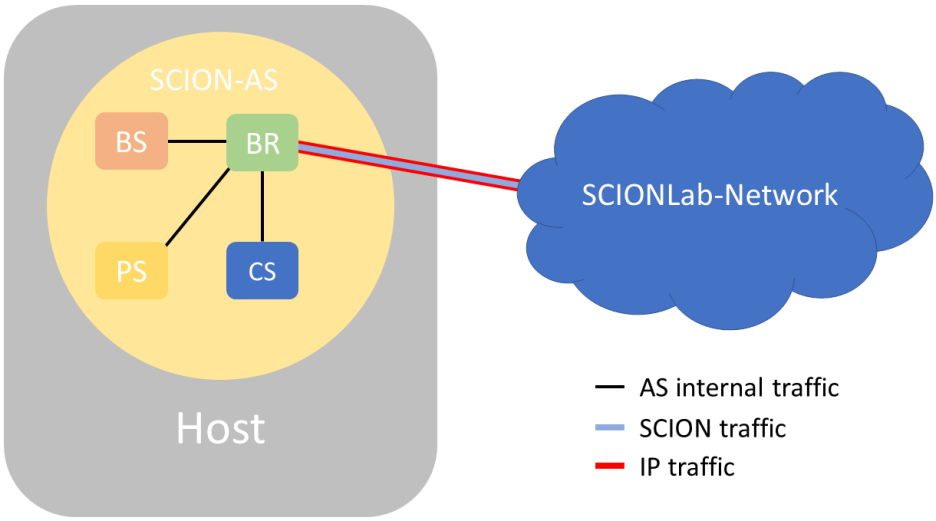

Figure 1.1: Schematic representation of a minimal SCIONLab-AS set-up

1.2.2 SCIONLab AS

There are two main ways to set up a SCIONLab-AS. The most common one is to set it up in a VM. The other option is to set it up on a dedicated SCION

system. Either way, you can do that by creating an account on scionlab.org and use the web interface to configure your AS to your needs. This web portal is called SCIONLab Coordination Service. After you configured your AS, you can download the files that are used to either set up the VM or the dedicated SCION system. The Attachment Point needs some time to receive the new configuration files to set up the connection to the user-AS. When the AP is ready, you will receive an e-mail that informs you about it. As visible in figure 1.2 the current version of the SCIONLab Coordination Service doesn't allow you to configure any bandwidth limits.

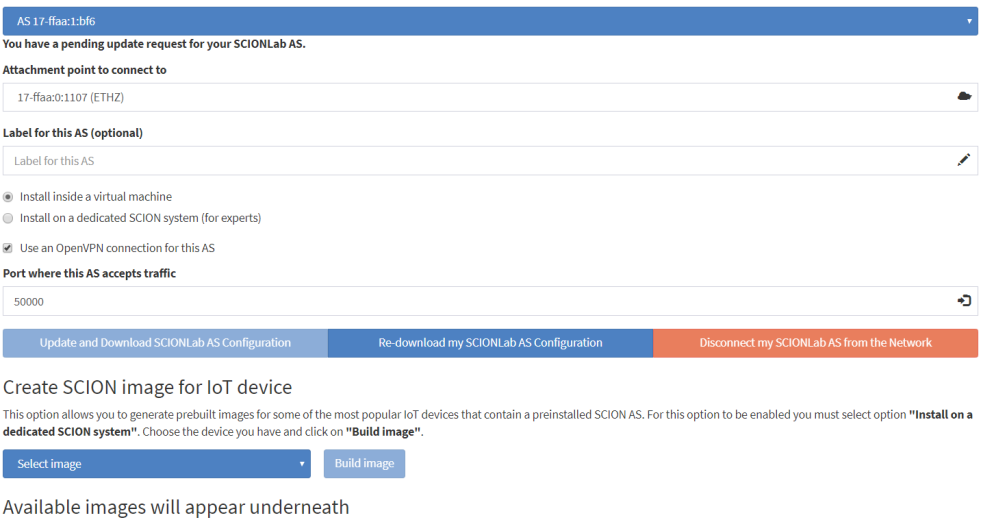

Figure 1.2: Web Interface of the SCIONLab Coordination Service

1.2.3 Topology

Chapter 2

Traffic Control

Traffic control in Linux is realized using a tool named TC. It consists of four basic techniques.

1. **Shaping**: This is the technique we will be using in order to enforce an upper bandwidth limit. But in general, it is the process of manipulating the bandwidth. It can also be used to smooth out bursts in order to improve the quality of service. Shaping is done on egress traffic.

2. **Scheduling**: This is the process of staging packets according to a schedule. Like this reordering of packets can be achieved. Scheduling as well happens with egress traffic.

3. **Policing**: This is the equivalent to shaping but for ingress traffic. It is worth noticing that traffic policing is more limited than traffic shaping, since there is no ingress queue.

4. **Dropping**: This is a quite primitive approach of just dropping traffic that exceeds a given bandwidth. This is applicable for both ingress and egress traffic.

For our needs, traffic shaping fits best. But since we need to limit ingress traffic as well, and shaping only operates on egress traffic, we need a workaround. More about that in section 2.1.3. Traffic control using TC is implemented using tree basic building blocks: QDISCs[1], classes and filters. They will be discussed in section 2.2.

[1] Queuing Disciplines

2.1 Theory

2.1.1 Traffic Shaping

As previously stated, traffic shaping happens with egress traffic. However, there is a special QDISC that is handling ingress traffic. Since this QDISC is the only one applicable on ingress traffic, it is called *ingress*. The term Queuing Discipline, in that case, is a bit misleading, because there is no such thing as an ingress queue. All the other QDISCs handle egress traffic and are quite diverse. Each QDISC has a shaping algorithm that shapes the traffic passing the corresponding QDISC. QDISCs can be either classful or classless (such as the ingress QDISC). Classful QDISCs have a shaping algorithm that can handle traffic, which is classified in different classes, differently. Both the classes and the classifiers, which are filters that determine which kind of traffic belongs to which class, are attached at the QDISC. Classless QDISCs can obviously not handle classes. However they can handle filters, which can either do policing on the traffic or i.e. redirect it to a different interface.

2.1.2 Egress Traffic

2.1.3 Ingress Traffic

Traffic Policing using Filters

Traffic Shaping using Virtual Interfaces

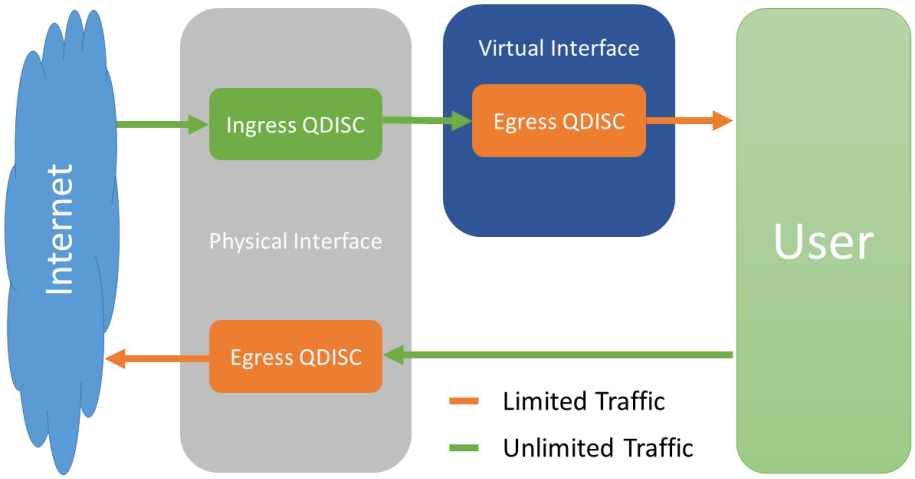

Figure 2.1: Interface set-up

2.2 TC

2.2.1 Queuing Disciplines

2.2.2 Classes

2.2.3 Filters

Chapter 3

Conception

3.1 Front end

3.2 Back end

Chapter 4

Implementation

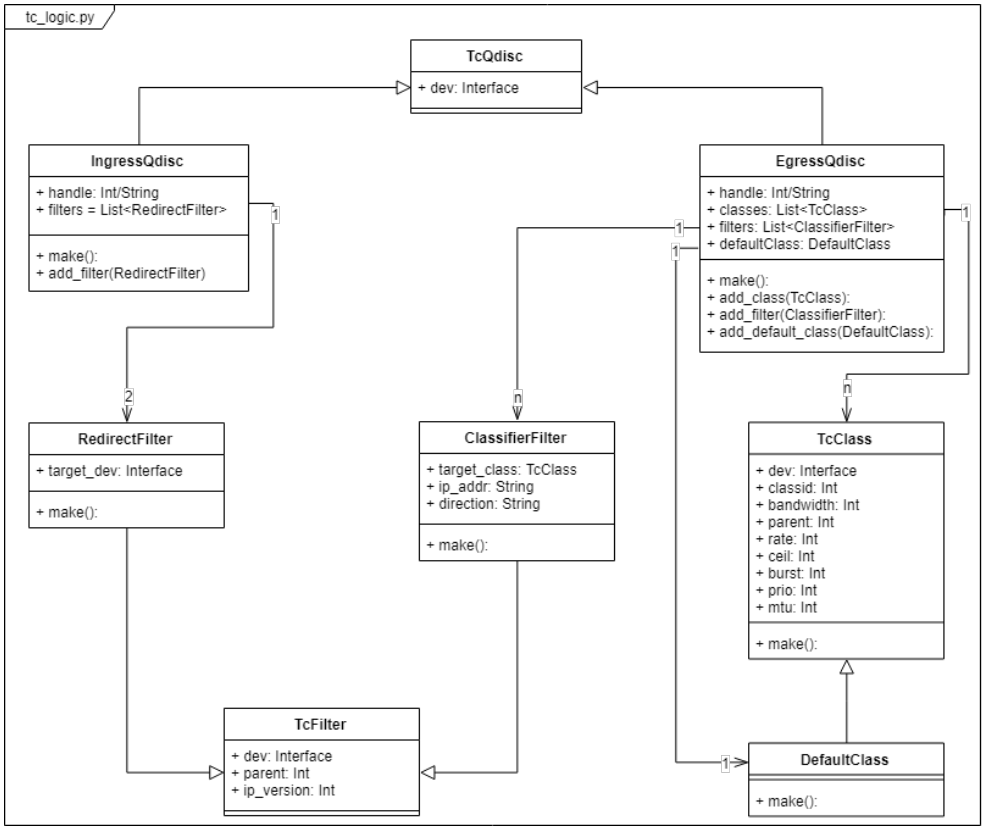

Figure 4.1: UML-Model of tc_logic.py

Chapter 5

Evaluation

5.1 iperf3

Chapter 6

Conclusion

Appendix A

Abbreviations

AP	Attachment Point
AS	Autonomous System
AWS	Amazon Web Services
BGP	Border Gateway Protocol
BR	Border Router
BS	Beacon Server
CS	Certificate Server
ETH	Eidgenössische Technische Hochschule (Swiss Federal Institute of Technology)
IP	Internet Protocol
ISD	Isolation Domain
ISP	Internet Service Provider
OSI	Open Systems Interconnection
PCB	Path-Segment Construction Beacon
PS	Path Server
QDISC	Queuing Discipline
SCION	Scalability, Control and Isolation on Next-Generation Networks
SCIONLab	Testbed for SCION
TC	Traffic Control
TRC	Trust Root Configuration
VM	Virtual Machine

Bibliography

[1] Adrian Perrig et al. *SCION: a secure Internet architecture*. Springer, 2017.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.