This document is a user’s manual document of an SDN OpenFlow controller based on already development OSS project trema-edge (<http://github.com/trema-edge.git>).

# Overview

SDN(Software Defined Networking define an abstract model of controlling network devices that are either hosts or switches connected together to form a controlled network.

Switch is a device and in our case a software device with ports and tables. Hosts connected to ports and transmit packets to ports. Packets arrived at switch are indexed against a table. A table consists of a classifier and a set of instructions/actions. A controller sits above governs and controls the actions. The following diagrams below depicts this basic idea.

cONTROLLER

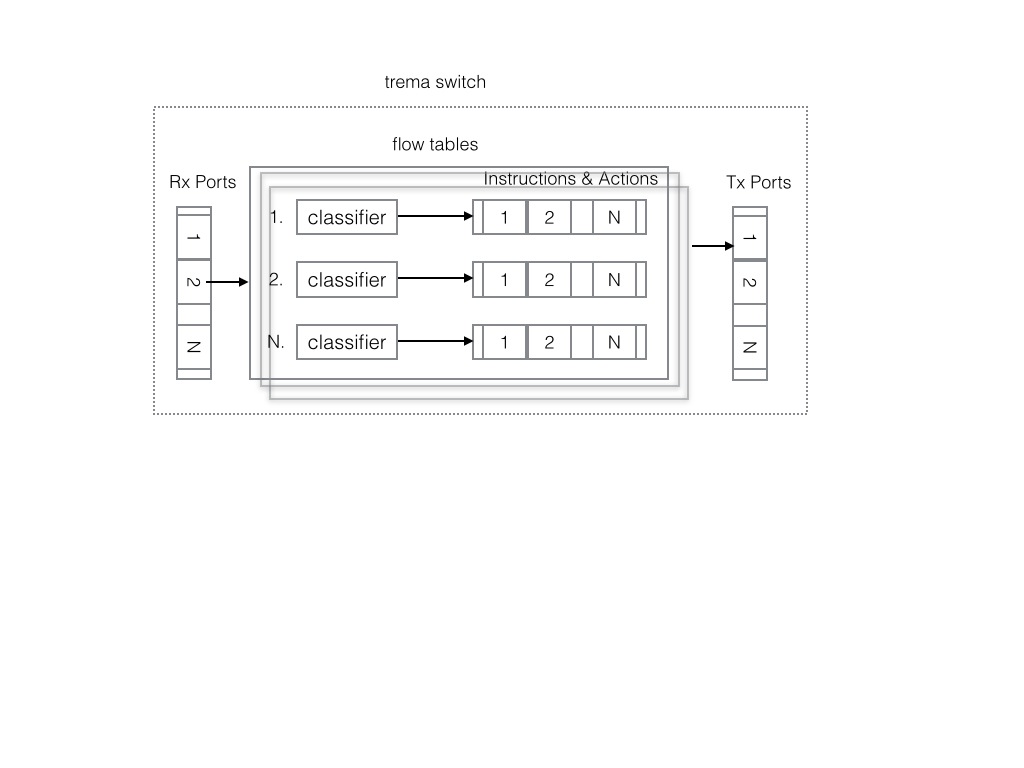
TREMA-SWITCH

virtual

host

virtual

host



SDN Controller

core switch

core switch

host3

edge switch

edge switch

host1

core switch

core switch

host4

host2

This project aims to demonstrate an OpenFlow controller capable of assigning bandwidth from edge to core switches using a fair-share algorithm. We name this project bandwidth enforcer traffic steering.

core

switch

core mesh network

SDN Controller

core

switch

core

switch

edge switch

core

switch

edge switch

bandwidth

enforcer

traffic

steering

traffic

statistics

SDN

controller

virtual hosts

traffic sources

virtual hosts

traffic sources

The above diagram depicts the main components of the SDN controller. A packet transmitted from a virtual traffic source would end up in the SDN controller which will attempt to assign a bandwidth that is equally distributed among the rest of the hosts entering the edge switch. All edge to core switches would be examined and the maximum possible allocated bandwidth would be assigned. All the above depicted components are software processes running in a single VM running Ubuntu with 4GBytes of memory with two or more dedicated CPUs. A user interacts or controls certain aspects of the software using a Chrome Web browser.

Ubuntu VM

SDN Controller



# Starting/Stopping the system

Using a terminal window login into the system using the provided password.

At the terminal window change directory to:

~$cd trema-edge

~$./trema run src/examples/bw\_enforcer/bw-enforcer.rb -c src/examples/bw\_enforcer/sample.conf

Wait for the system to start up approximately 10 second

~$cd trema-edge

~$./trema send\_packets -s host1 -d host5

~$./trema show\_stats host5

~$./trema send\_packets -s host5 -d host1

~$./trema show\_stats host1

~$./trema send\_packets -s host1 -d host5 —pps=100

~$./trema send\_packets -s host5 -d host1 —pps=100

The first trema command sends a single packet from a source host 1 to destination host 5. The show\_stats command would display statistics that verifies that successfully the packet has been transmitted and received by the destination host. The extra option “pps” packets per second sends 100 packets per second from a source host to destination host.

To be able to use the web browser two steps are needed. The first step starts the redis server that allows communication of events and control between the SDN controller and the web server. To start the redis server type the following command:

~$cd redis-server

~$redis-server &

It is recommended to run the above command in a separate window to avoid clogging the terminal with output messages.

Type the next command to start the web server on the Ubuntu.

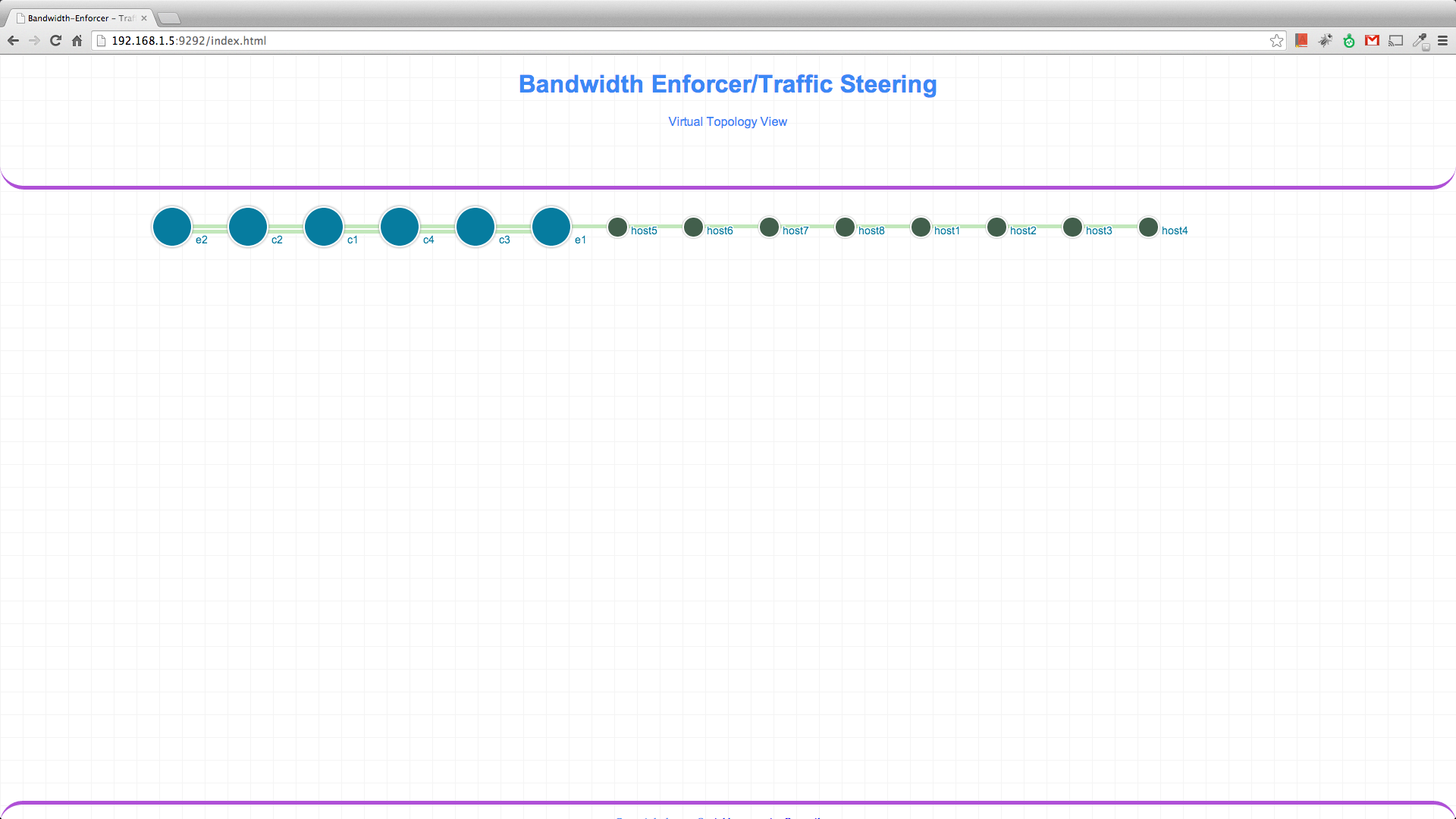
~$cd trema-edge/src/examples/bw-enforcer

~$rackup config

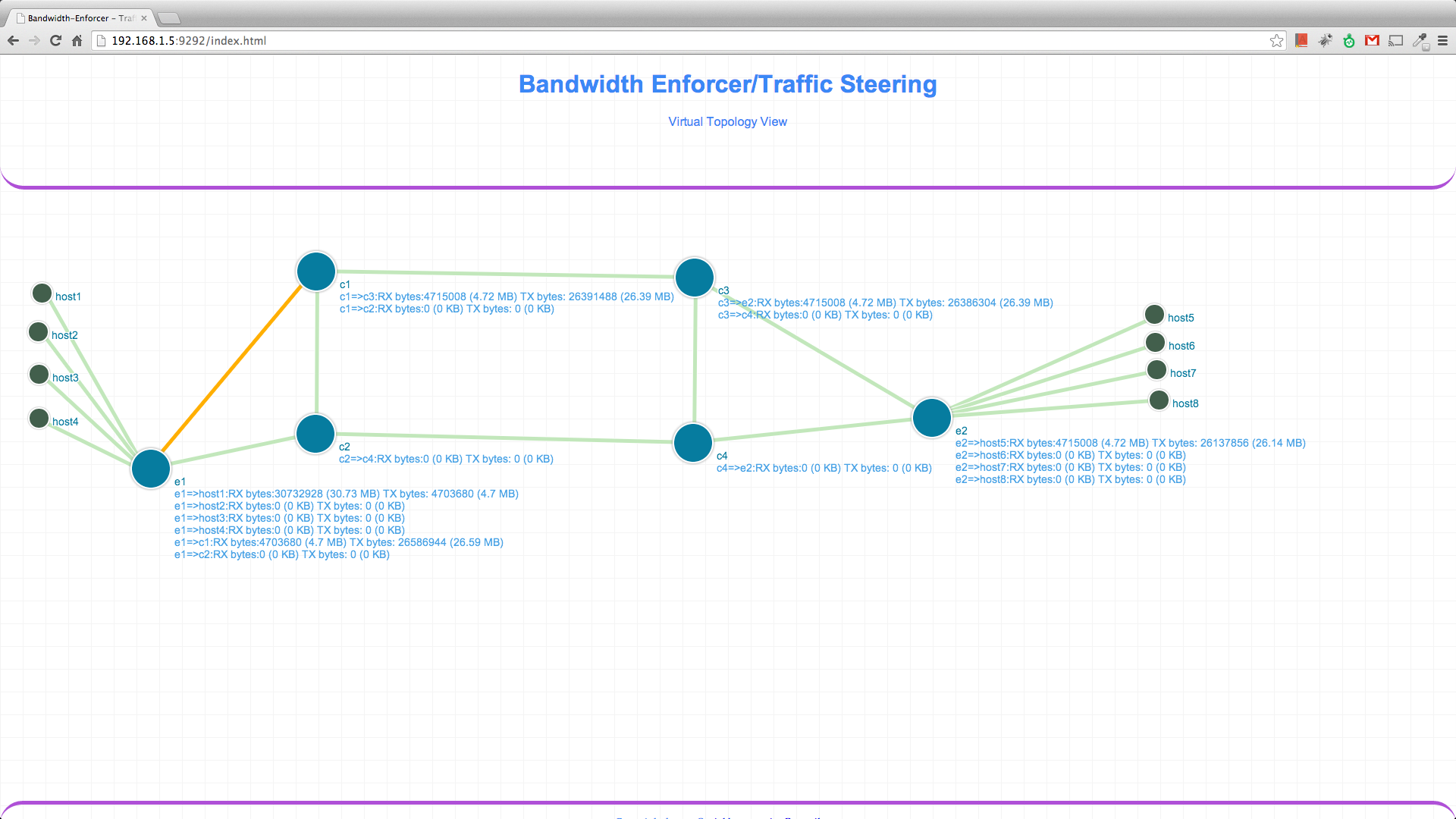
The above command starts the web server running on the same Ubuntu PC. The user points to browser using the following url:

http://localhost:9292/index.html

The web server would read the configured topology and display all nodes in a straight line. Connected nodes are joined with line segments indicated in green color. The following screen shows the initial configuration.



The next screenshot indicates that the link from edge switch e1 to core switch c1 is 1/3 of its capacity indicated by the orange colored line.

If the edge to core switch capacity is nearly full the user can double click at the link ie (e1=>c1) and assign a new bandwidth value. Then on any subsequent new packet the fair-share algorithm would take into consideration the new link capacity value and would try to reassign all current paths to a new calculated value.