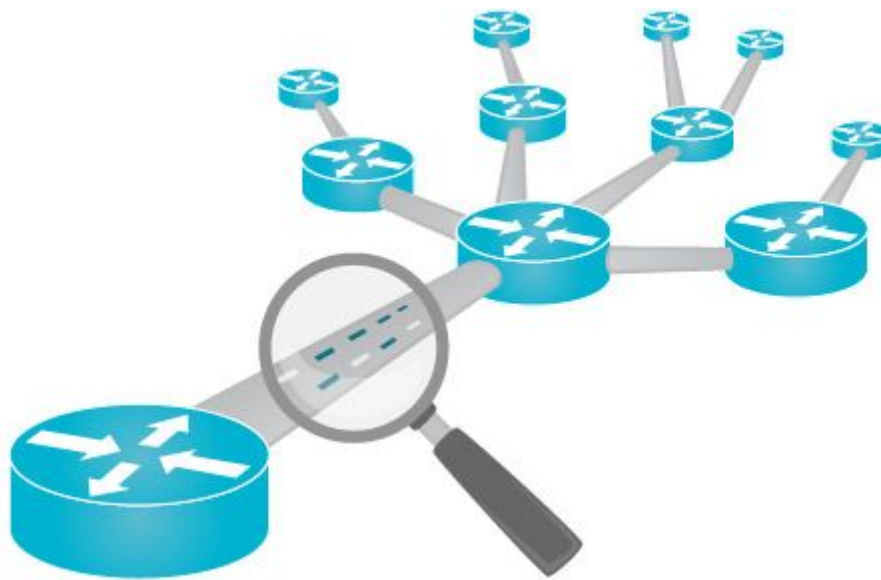


University Of Pisa

Advanced Network Architecture and Wireless Systems

Network Traffic Monitoring



Authors:

De Bianchi Luigi
Micheloni Giulio
Piscione Pietro

Index

- [1. Introduction](#)
- [2. Assumptions](#)
- [3. Tools](#)
 - [3.1 Quagga](#)
 - [3.1.1 Introduction](#)
 - [3.1.2 Installation](#)
 - [3.1.3 Configuration](#)
 - [3.1.4 Use](#)
- [4. Functionalities](#)
 - [4.1 Topology](#)
 - [4.1.1 Differences](#)
 - [4.1.2 Algorithm](#)
- [5. Utilization](#)
 - [5.1.1 Preliminar study](#)
 - [5.1.2 Implementation](#)
 - [5.1.3 Modus operandi](#)
 - [5.1.3.1 Synchronous mode](#)
 - [5.1.3.2 Asynchronous mode](#)
 - [5.1.3.3 Synchronous vs asynchronous mode](#)
 - [5.1.3.4 Tests](#)
 - [5.1.3.5 Example](#)
- [6. Tunnel](#)
- [7. How to use](#)
 - [7.1 First screen](#)
 - [7.2 GUI](#)
- [8. Appendix](#)
 - [8.1 VFiles](#)
 - [8.2 SNMP tunnel retrieving algorithm](#)
- [9. References](#)

1. Introduction

The Internet is a big and complex network by which is possible to exchange data across the all world. The time to deliver packets from one end of the world mainly depends from network status. In particular, the condition of each router along the path traversed by the packets determine the delay and throughput end-to-end.

The aim of this project is to realize a software to monitor a network status and provides the information topology, network usage and instaured MPLS tunnels, i.e. subset of network in which some resources are explicitly reserved within the tunnels themselves.

The network taken in consideration could be an Autonomous System which implements OSPF as routing protocol.

2. Assumptions

The software is only an analysis tool so some assumptions over other components of the network are required:

- The Autonomous Systems uses only one routing area. In case of more than one routing area, then the software is capable to reconstruct only one area topology
- All routers are Cisco and they are already SNMP-configured
- In general at least one address of the network must be available in order to build the topology but in case we use Quagga software this limit can be overcome

3. Tools

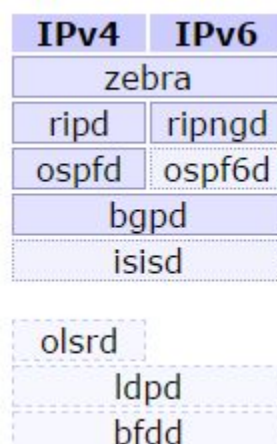
The tools used are the following:

- Quagga
 - A routing software suite that provide implementations of OSPFv2 and v3
 - install with `configure_quagga.sh`
- Telnet
 - The popular application layer protocol used when quagga is not available
 - Normally present in every linux distro but in case is missing `sudo apt-get install telnetd`
- pySNMP
 - A python library to manage SNMP protocol
 - install with `pip install pysnmp`
- Networkx and matplotlib
 - python libraries to build and show in the GUI the network topology
 - install with `pip install networkx`
 - install with `sudo apt-get install python-matplotlib`
- py2-ipaddress
 - python library to interact with IP address
 - install with `pip install https://pypi.python.org/packages/source/p/py2-ipaddress/py2-ipaddress-3.4.tar.gz`
- GNS3
 - As network emulator

3.1 Quagga

3.1.1 Introduction

Quagga is an open source routing software based on the Zebra router, for which development was stopped in 2003. It supports the main standardised routing protocols such as RIP, OSPF or BGP and can be installed on any Linux system with a 2.4 or higher kernel.



As it's possible to see in the image quagga is composed of several daemons, one per routing protocol and another one called Zebra acting as the kernel routing manager. Each daemon has its own configuration file and terminal interface which can be accessed by telnet.

The vtysh tool is provided to configure the Quagga router from the localhost, in a unique interface.

Quagga works independently from the operating system (OS) over which it is installed.

It must be emphasised that Quagga owns only routing capabilities and functionalities associated with it, such as access lists or route maps. It does not provide "non-routing" functionalities such as DHCP server, NTP server/client or ssh access but it is often possible to enable them on the operating system supporting your Quagga router.

The Quagga Command line Interface is very similar to the Cisco IOS software allows it to be configured very easily for those who are familiar with Cisco.

3.1.2 Installation

Quagga can be installed through normal packet manager in any linux system with:

```
sudo apt-get install quagga quagga-doc
```

3.1.3 Configuration

[The file **quagga_setup.sh** in the main folder automatically execute most of the configuration described here]

Configuration is entirely manual;

First of all the user must create and configure .conf files under "/etc/quagga/" for each daemon he wants to run. The configuration possibility are almost endless so we recall for quagga_setup.sh for actual configuration in our case.

Moreover such files need to be set up such that user "quagga" is owner and with 640 as octal permits.

After daemon configuration the user need to set zebra to start the daemons with the ip of the interface connected to the network.

Once every file is ready daemons can be run with:

```
sudo /etc/init.d/quagga [start. restart, stop]
```

If with

```
ps -ef | grep quagga
```

are shown all the daemons everything went well.

3.1.4 Use

In quagga packets is present "vtysh" tool that can be used either in prompt mode which open a prompt exactly like IOS or in command mode which accept one IOS command at the time like:

```
vtysh -c "show ip ospf database"
```

4. Functionalities

In this section the main features of the software are described, they are logically divided by provided network services.

4.1 Topology

In this case we are interested in the OSPF topology that can be obtained in two ways:

1. Quagga
2. Telnet

Our software support both mode and each one has advantages and drawbacks.

Quagga need more software installation and a root access but since is local no delay are added to collect information when Telnet is limited by the network.

4.1.1 Differences

Since either systems use IOS like commands the algorithm to rebuild the topology doesn't change but quagga register itself as a new OSPF router so the obtained topology is different in the two case.

The presence of one more router isn't a critical factor because we are sure that nothing in the route decision is affected since Quagga router is an empty branch in the topology.

4.1.2 Algorithm

With either Quagga or Telnet we need to obtain the "OSPF Database" with the IOS command:

```
show ip ospf database
```

and the queried router return something like this where we are interested in the id of the routers in the the ospf area that is clearly visible in the leftmost column of the output under the "*Router Link State*" section.

```

user@instant-contiki: ~
File Edit View Search Terminal Help
user@instant-contiki:~$ sudo vtysh -c "show ip ospf database"

    OSPF Router with ID (192.168.0.100)

      Router Link States (Area 0.0.0.0)

Link ID      ADV Router   Age  Seq#       CkSum  Link count
1.1.1.1      1.1.1.1      66  0x80000004 0x2aa0  6
2.2.2.2      2.2.2.2      504 0x80000004 0x8959  5
3.3.3.3      3.3.3.3      506 0x80000003 0xb8ef  3
4.4.4.4      4.4.4.4      510 0x80000003 0xc707  5
5.5.5.5      5.5.5.5      514 0x80000002 0x7151  6
192.168.0.100 192.168.0.100 61  0x80000005 0x8747  1

      Net Link States (Area 0.0.0.0)

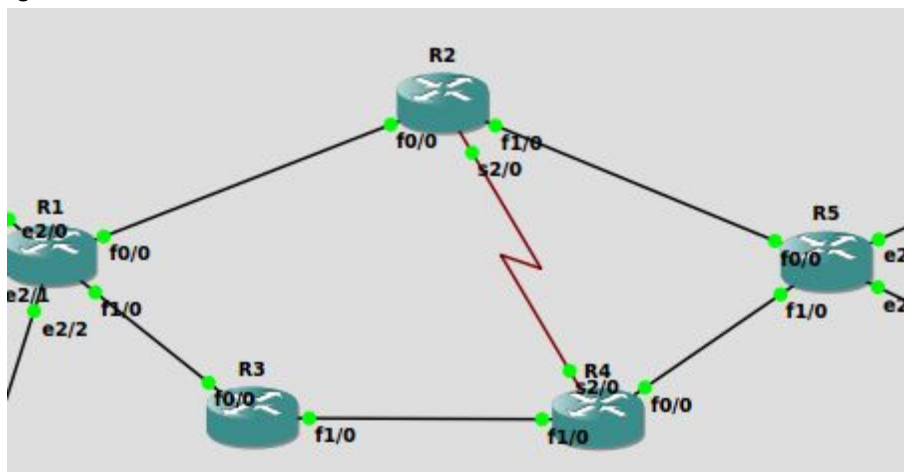
Link ID      ADV Router   Age  Seq#       CkSum
10.1.1.2     2.2.2.2      504 0x80000001 0x1a01
10.2.2.2     3.3.3.3      506 0x80000001 0x070a
10.3.3.2     4.4.4.4      515 0x80000001 0x58a6
10.4.4.2     5.5.5.5      514 0x80000001 0x7779
10.5.5.2     5.5.5.5      514 0x80000001 0xfbfa
192.168.3.1  1.1.1.1      67  0x80000001 0x7482

```

Once we know every router we need to understand how they are connected and this can be done with another IOS command:

```
show ip ospf database router [ ID ]
```

Suppose in gns3 we have a structure like this



the output corresponding to the router r2 gives


```

user@instant-contiki:~$ sudo vtysh -c "show ip ospf database router 2.2.2.2"

    OSPF Router with ID (192.168.0.100)

          Router Link States (Area 0.0.0.0)

LS age: 828
Options: 0x22 : *|-|DC|-|-|E|*
LS Flags: 0x6
Flags: 0x0
LS Type: router-LSA
Link State ID: 2.2.2.2
Advertising Router: 2.2.2.2
LS Seq Number: 80000004
Checksum: 0x8959
Length: 84
  Number of Links: 5

    Link connected to: Stub Network
      (Link ID) Net: 2.2.2.2
      (Link Data) Network Mask: 255.255.255.255
      Number of TOS metrics: 0
      TOS 0 Metric: 1

    Link connected to: another Router (point-to-point)
      (Link ID) Neighboring Router ID: 4.4.4.4
      (Link Data) Router Interface address: 12.2.2.1
      Number of TOS metrics: 0
      TOS 0 Metric: 64

    Link connected to: Stub Network
      (Link ID) Net: 12.2.2.0
      (Link Data) Network Mask: 255.255.255.252
      Number of TOS metrics: 0
      TOS 0 Metric: 64

    Link connected to: a Transit Network
      (Link ID) Designated Router address: 10.5.5.2
      (Link Data) Router Interface address: 10.5.5.1
      Number of TOS metrics: 0
      TOS 0 Metric: 1

    Link connected to: a Transit Network
      (Link ID) Designated Router address: 10.1.1.2
      (Link Data) Router Interface address: 10.1.1.2
      Number of TOS metrics: 0
      TOS 0 Metric: 1

```

There are three possible Network link that can obtained:

- Stub
- Transit
- point-to-point

Stub networks are those from which the router doesn't receive any ospf messages. On the other side Transit and point-to-point Network are those connected to ospf router; point-to-point refer to serial link and Transit to ethernet link.

So we are only interested in Transit and point-to-point Networks to build our topology.

To build our matrix of connection we are interested in the rows with "(Link ID)" and "(Link Data)" field.

- “Link ID”
 - in case of a Transit Network specify the “*Designated Router*” that is the elected router during the “*Hello phase*” in the segment; since the same information is held on the two ends of the segment the two interfaces that have the same Designated router are connected
 - in case of a point-to-point link this field directly held the ID of the connected router
- “Link Data” field has the same value in Transit and point-to-point Networks and specify the IP assigned to the local interface for the specified network

So once all these informations are collected from all the routers building the topology is straightforward since for each interface we know the next router (point-to-point case) or the common Designated router (Transit Network case).

5. Utilization

The aim of this part is to retrieve, through some specific informations, the network usage. The network usage is one of most important information about the network itself. It informs a user (typically a system administrator) from a possible bottleneck presence.

5.1.1 Preliminar study

In the preliminary study what we focus on is how to get the utilization from a link. In the networking environment the packets go through the router interfaces. The packets number that across a specific interface depend from a lot of factor: network traffic, interface speed, scheduling algorithm and so on.

From a mathematical point of view, the normalized usage of an interface is the following:

$$\frac{(\Delta IfInOctecs + \Delta IfOutOctecs) * 8}{(number\ of\ seconds\ in\ \Delta) * ifSpeed}$$

- $\Delta IfInOctecs$: expressed in byte, is the difference of ingoing octets number in the Δ interval of time
- $\Delta IfOutOctecs$: expressed in byte, is the outgoing octets number in the Δ interval of time
- $ifSpeed$: is the speed of the interface expressed in bit/sec
- $number\ of\ seconds\ in\ \Delta$: is the number of seconds between two measurements. It is expressed in seconds.

The result of this formula has not measurement unit. The minimum value is zero and the maximum is one.

- Zero means no bits pass through the interface in the Δ interval.
- One means that the maximum number of bits pass in the Δ interval. The maximum number of bits depends from interface speed.

The above formula is multiplied by 100 and we can express the above formula in percentage.

What we have obtained is the utilization on a link: giving a temporal interval Δ in which an amount of bits pass into and out from an interface and giving its interface speed $ifSpeed$, the utilization of the link attached to that interface is giving by above formula.

In detail, is possible to compute the ingoing and outgoing utilization for a specific interface.

$$\frac{(\Delta IfInOctecs) * 8}{(number\ of\ seconds\ in\ \Delta) * ifSpeed}$$

$$\frac{(\Delta IfOutOctets)*8}{(number\ of\ seconds\ in\ \Delta)*ifSpeed}$$

They are the ingoing utilization formula and outgoing utilization formula, respectively.
From now the ingoing utilization and outgoing utilization are called simply utilizations for simplicity reason.

5.1.2 Implementation

The SNMP (Simple Network Management Protocol) is widely used for monitoring and control networks. It is a request-response protocol and it can either read or (in some cases) write variables (called more specifically entities) present in a router that is SNMP-configured.

The SNMP protocol is, for its simplicity, a good solution to monitoring the router resources. In our case what we need to obtain is the utilization on the links.

The SNMP manager is the node in the network, by which all requests to the routers are performed.

It can obtain the above informations from one or more routers and then is possible to compute, using above formula, the utilization(s).

The routers organizes information about itself (routing protocol and tables, interfaces names, speeds, tunnels and so on) hierarchically (tree-structured) in a MIB (Management Information Base), a database used for managing the entities. Each entity is identified through a OID (Object Identifier).

For instance the interface status would have an OID of 1.3.6.1.2.1.2.2.1.8.1 The OID values depends from the device and, in some cases, from the interface ID. The final number is the interface ID.

E.g.

```
snmpwalk -v2c -c public 172.16.99.1 1.3.6.1.2.1.2.2.1.8.
```

This requests has the following option:

the version is the 2c (c stands for community)

the community name is "public" (the common used)

the request is made to 172.16.99.1 address

the OID requested is 1.3.6.1.2.1.2.2.1.8. which meaning has been specified before

For our purposes we need the OID of the following entities:

- ifSpeed: the interface speed expressed in bit/sec
- IfInOctets: the ingoing octets number expressed in bit
- IfOutOctets: the outgoing octets number expressed in bit
- SysTimeUp: the router system time up expressed in hundredth of seconds

ifSpeed, IfInOctets and IfOutOctets are easily to infer what they represent.

The SysTimeUp is used for compute the temporal interval between two requests and has been opportunely converted in seconds.

For simplicity we don't report all OIDs of above entities. They can be found online.

Once obtained the above entities is not possible to compute the utilization. We need to obtain twice the above entities and then is possible to compute the utilization because the delta values are needed. Moreover, the interface speed is retrieved only once because it is a static quantity that doesn't change over the time. Meanwhile the ingoing octets, outgoing octets and the system time up change over the time. This procedure is applied to all interfaces of all routers in the network taken in consideration. In this way is possible to have the utilizations values on all links.

5.1.3 Modus operandi

The SNMP manager can request either in asynchronous or synchronous mode the informations to the routers about the utilizations.

5.1.3.1 Synchronous mode

In the synchronous mode the manager asks periodically to routers about entities to compute the links utilizations. The polling period can be set manually by the user.

From an implementation point of view a thread can ask periodically through SNMP protocol the information to compute the utilizations.

5.1.3.2 Asynchronous mode

In the asynchronous mode the manager can asks when obtain the network status. In this way is not necessary a periodic polling. The user can decide whenever gets the links utilizations.

5.1.3.3 Synchronous vs asynchronous mode

These two modes are implemented because a system administrator, or more in general an end user, would have either a freshness or not network status through links utilizations. The data reported to him\her are shown in a topology in which from green (low) to red (high) is reported the link utilizations.

In synchronous and asynchronous cases the network status will be available but with different method. In the first case, taken a low polling period time, i.e. high polling frequency, the network could suffer from flooding due to high requests number. Instead, in the second case the user can decide when obtain the network status. A frequent refresh can lead to a network flooding.

A good tradeoff between network traffic and network status freshness could be either a "not high" periodic polling or a refresh made "not more" frequently. In both case is up to the user decides when update the network status.

5.1.3.4 Tests

In order to verify the links consistency utilizations, i.e. the network status, some tests have been performed.

The tests have been made with *iperf* and the *ping test*.

With iperf the traffic was sent from one host to another for a certain amount of time and then the utilization on involved links was measured. The data obtained from the software was consistency what iperf sent through those links.

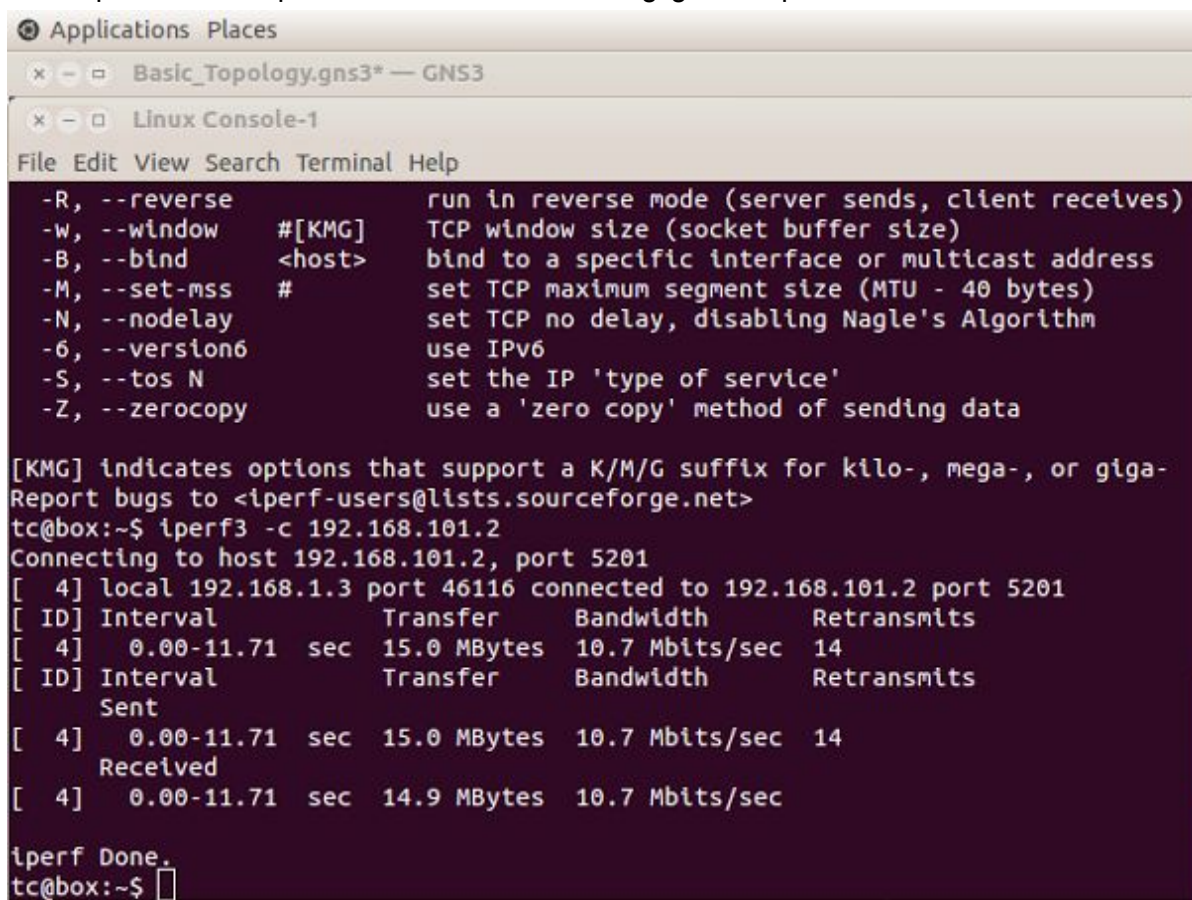
Is was also used the ping test with a specific ping packet size to test the formula correctness. Also in this case the results was coherent with what we expect.

5.1.3.5 Example

The following example shows the iperf test.

A traffic of around 10.7 Mbit/sec is sent from Host1 to host2. Since that the FastEthernet Interface speed is 100 Mbit/sec the normalized use of the link attached to that interface is around 10.7%. The computed utilization through the software is 10,755%. The slightly difference with expected values is due to SNMP requests and responses that passes through that link.

The requests and responses sizes of SNMP is negligible respect to the traffic in the link.



```
Applications Places
Basic_Topology.gns3* — GNS3
Linux Console-1
File Edit View Search Terminal Help
-R, --reverse          run in reverse mode (server sends, client receives)
-w, --window           #[KMG]   TCP window size (socket buffer size)
-B, --bind             <host>   bind to a specific interface or multicast address
-M, --set-mss          #        set TCP maximum segment size (MTU - 40 bytes)
-N, --nodelay           set TCP no delay, disabling Nagle's Algorithm
-6, --version6         use IPv6
-S, --tos N            set the IP 'type of service'
-Z, --zerocopy         use a 'zero copy' method of sending data

[KMG] indicates options that support a K/M/G suffix for kilo-, mega-, or giga-
Report bugs to <iperf-users@lists.sourceforge.net>
tc@box:~$ iperf3 -c 192.168.101.2
Connecting to host 192.168.101.2, port 5201
[ 4] local 192.168.1.3 port 46116 connected to 192.168.101.2 port 5201
[ ID] Interval           Transfer     Bandwidth       Retransmits
[ 4]   0.00-11.71   sec   15.0 MBytes   10.7 Mbits/sec    14
[ ID] Interval           Transfer     Bandwidth       Retransmits
Sent
[ 4]   0.00-11.71   sec   15.0 MBytes   10.7 Mbits/sec    14
Received
[ 4]   0.00-11.71   sec   14.9 MBytes   10.7 Mbits/sec

iperf Done.
tc@box:~$
```

TE Dashboard			
Text information			
Router Name	Speed	Utilization %	Connected to
▼ R1			R2, R3,
FastEthernet0/0	100 Mbit/s	IN: 0.207% OUT: 0.0%	R2
FastEthernet1/0	100 Mbit/s	IN: 0.0% OUT: 10.338%	R3
Ethernet2/2	10 Mbit/s	IN: 0.067% OUT: 0.07%	
▷ R2			R1, R5, R4
▼ R3			R1, R4
FastEthernet0/0	100 Mbit/s	IN: 10.755% OUT: 0.006%	R1
FastEthernet1/0	100 Mbit/s	IN: 0.0% OUT: 9.739%	R4
▼ R4			R5, R3, R2
FastEthernet0/0	100 Mbit/s	IN: 0.0% OUT: 7.142%	R5
FastEthernet1/0	100 Mbit/s	IN: 7.144% OUT: 0.006%	R3
Serial2/0	1 Mbit/s	IN: 0.025% OUT: 0.018%	R2
▼ R5			R2, R4
FastEthernet0/0	100 Mbit/s	IN: 0.001% OUT: 0.189%	R2
FastEthernet1/0	100 Mbit/s	IN: 8.446% OUT: 0.0%	R4
	192.168.0.100		

In the appendix 8.1 there is another method to retrieve the link utilizations, it relies on the VFiles mechanism. This part actually has not been completely implemented yet. The sending VFiles mechanism works correctly with an already TFTP server implemented. The software manager is not still interact with the VFiles mechanism. Further works can be done on this part.

6. Tunnel

The aim of this part is the retrieving and collecting of informations about MPLS traffic engineering tunnels configured and instaured in the network among routers.

For this purpose the SNMP protocol offers many different OIDs, which are specified by name in this part, but for more details see reference [1] and [6].

Now the algorithm executed by the class TeTunnels.py is briefly explained.

The inputs of this algorithm are the router ip address and the community name for SNMP, meanwhile the outputs are two table, in form of dictionary, where the former describes the information regarding each configured tunnel, and the latter instead contains information belonging to each LSP in the router.

Since these are tables, each row is implemented through the class Container.py, that is a wrapper for a dictionary with utility functions for adding and retrieving attributes in form of pairs <key, value>, each attribute is a table column.

The table itself is implemented as a dictionary with entries like this: <key, Container Object>, where the key may be either the tunnel or LSP name.

The information associated to each row of the two tables are described below:

Configured tunnels table:

- Number of bytes ingoing/outgoing.
- Configured path (this field is empty if the path is configured as dynamic).
- Source address.
- Destination address.
- Descriptive name (that is the LSP's name associated to this tunnel configuration).

LSP table:

- Number of bytes forwarded by the tunnel (It is zero if the router is either the source or the destination. OID: mplsTunnelPerfHCBytes).
- Source address.
- Destination address.
- Computed path.
- Holding priority (OID: mplsTunnelHoldingPrio).
- Setup priority (OID: mplsTunnelHoldingPrio).
- Mean rate in bit/sec, as part of TSpec (OID: mplsTunnelResourceMeanRate)
- Maximum rate in bit/sec, as part of TSpec (OID: mplsTunnelResourceMaxRate)
- Maximum burst size in bytes (OID: mplsTunnelResourceMaxBurstSize)
- Administrative status (OID: mplsTunnelAdminStatus)
- Operative status (OID: mplsTunnelOperStatus)
- Tunnel role (OID: mplsTunnelRole)

Now that we have defined the output format, we can introduce the conceptual algorithm. The main task is to scan the SNMP OID "*mplsTunnelTable*" whose entries contain most of the information needed to fill the output data structures. The rest of information, like the path

followed by each tunnel and the network resources desired for each of them, lie in the OID *"mplsTunnelHopTable"* and *"mplsTunnelResourceTable"*, respectively.

First of all the algorithm has to know how many tunnels are configured in the router, moreover it needs to create a mapping between the tunnel names and their unique identifiers, for the purpose of future references, thus in order to do this it scans the OID resource called *"mplsTunnelName"* and it saves into internal data structures the name of each tunnel and the relative mapping with its identifier.

The second step concerns the retrieving of installed LSP names, and creating the mapping between LSP names and their unique identifiers. In this step it also creates the associations among tunnel configuration names and their relative LSP names.

Now that it has all of these information, it can directly access to all SNMP resources, for each tunnel configurations and LSPs, using the unique identifiers mapping.

The final step is to collect all needed information, i.e. the fields listed above, from the *"mplsTunnelTable"*. For each LSP this procedure scans one by one the interested OIDs, and it stores the result. The only difference is when the algorithm needs retrieve the information relative to the LSP path and desired network resources. In the first case it has to previously obtain the index of the given LSP for the *"mplsTunnelHopTable"*, this can be done by querying the OID *"mplsTunnelHopTableIndex"*, and then it can access to the table itself. For the desired network resources the algorithm has to do the same procedure, that is it must previously retrieve the index for the *"mplsTunnelResourceTable"* through accessing the OID *"mplsTunnelResourcePointer"*, and then it is able to access to the table itself.

7. How to use

7.1 First screen

TE Dashboard

Traffic Engineering Dashboard

Router IP address:

SNMP community name:

Retrieve network topology using: ☒ Telnet ☐ Quagga

Here it's possible to select to use Quagga or Telnet; in case of quagga the field IP address is useless since the router is local but the user must know the SNMP Community name essential for every SNMP request.

7.2 GUI

The screenshot shows the TE Dashboard GUI with three main sections highlighted by red boxes and numbered 1, 2, and 3.

Section 1 (Test information): A table showing router details.

Router Name	IP address	Subnet mask	Connected to
> R1	1.1.1.1		R2, R3,
> R2	2.2.2.2		R1, R5, R4
> R3	3.3.3.3		R1, R4
> R4	4.4.4.4		R5, R3, R2
> R5	5.5.5.5		R3, R4
	192.168.0.100		

Section 2 (Commands): A vertical list of buttons for controlling the dashboard.

- refresh information
- Settings
- Show link utilization
- Show TE tunnels
- Show network topology

Section 3 (Network topology): A diagram showing the network topology with routers R1 through R6 and their connections.

This is the core of the program where the user can see all the information.

1. Router list is always visible with all the the information per interface except the utilization.

Text information			
Router Name	IP address	Subnet mask	Connected to
▼ R1	1.1.1.1		R2, R3,
FastEthernet0/0	10.1.1.1	255.255.255.252	R2
FastEthernet1/0	10.2.2.1	255.255.255.252	R3
Ethernet2/0	192.168.1.1	255.255.255.0	
Ethernet2/1	192.168.2.1	255.255.255.0	
Ethernet2/2	192.168.3.1	255.255.255.0	
Loopback0	1.1.1.1	255.255.255.255	
▼ R2	2.2.2.2		R1, R5, R4
FastEthernet0/0	10.1.1.2	255.255.255.252	R1
FastEthernet1/0	10.5.5.1	255.255.255.252	R5
Serial2/0	12.2.2.1	255.255.255.252	R4
Loopback0	2.2.2.2	255.255.255.255	
▷ R3	3.3.3.3		R1, R4
▷ R4	4.4.4.4		R5, R3, R2
▷ R5	5.5.5.5		R2, R4
	192.168.0.100		

2. These are the feature implemented

- Refresh Information:

This is self explanatory, if the user need updated information can force the update.

- Settings:

The Settings dialog box has a title bar with standard window controls. It contains two main sections: 'Working mode' and 'Parameters'. In the 'Working mode' section, there are two radio buttons: 'Synchronous mode' (which is selected) and 'Asynchronous mode'. In the 'Parameters' section, there is a 'Refresh time' label followed by a text input field containing the value '1'. Below this, there are two radio buttons: 'Show all interfaces' and 'Show only active interfaces' (which is selected). At the bottom of the dialog, there are two buttons: 'Save' and 'Cancel'.

From this window the user can set some parameters, like the working mode (asynchronous or synchronous), the refresh time of link utilizations, and if to show all router interfaces or just the active ones.

- Show link utilization:

Text information			
Router Name	Speed	Utilization %	Connected to
▼ R1			R2, R3,
FastEthernet0/0	100 Mbit/s	IN: 0.0% OUT: 0.0%	R2
FastEthernet1/0	100 Mbit/s	IN: 0.0% OUT: 0.001%	R3
Ethernet2/2	10 Mbit/s	IN: 0.043% OUT: 0.045%	
▼ R2			R1, R5, R4
FastEthernet0/0	100 Mbit/s	IN: 0.004% OUT: 0.004%	R1
FastEthernet1/0	100 Mbit/s	IN: 0.0% OUT: 0.0%	R5
Serial2/0	1 Mbit/s	IN: 0.02% OUT: 0.02%	R4
▷ R3			R1, R4
▷ R4			R5, R3, R2
▷ R5			R2, R4
192.168.0.100			

Update the view with newly evaluated utilization information.

- Show TE tunnels

Text information			
Router Name	Speed	Utilization %	Connected to
▼ R1			R2, R3,
FastEthernet0/0	100 Mbit/s	IN: 0.0% OUT: 0.0%	R2
FastEthernet1/0	100 Mbit/s	IN: 0.0% OUT: 0.001%	R3
Ethernet2/2	10 Mbit/s	IN: 0.043% OUT: 0.045%	
▼ R2			R1, R5, R4
FastEthernet0/0	100 Mbit/s	IN: 0.004% OUT: 0.004%	R1
FastEthernet1/0	100 Mbit/s	IN: 0.0% OUT: 0.0%	R5
Serial2/0	1 Mbit/s	IN: 0.02% OUT: 0.02%	R4
▷ R3			R1, R4
▷ R4			R5, R3, R2

Name	Source	Destination	Path	Max rate	Max burst	Mean rate
R2						
▼ R1						
R1_t1	1.1.1.1	5.5.5.5	R1 R2 R5	1024.0 Kbit/s	1000 bytes	1024.0 Kbit/s
R1_t0	1.1.1.1	5.5.5.5	R1 R3 R4 R5	400.0 Kbit/s	1000 bytes	400.0 Kbit/s

Add a new view with all the tunnel that begin from the selected router. Since the protocol to obtain such information is pretty slow the can request tunnels from one router at the time.

- Show network topology

It is shown the network topology graph like in the third red box inside the first image of this section.

3. Network topology is present from the beginning with all the interfaces. In case the user close the graph is always possible to have a new one with the “Show network topology” button.

8. Appendix

8.1 VFiles

The SNMP manager can request either in asynchronous or synchronous mode the informations to the routers about the utilizations.

In the synchronous mode the manager asks periodically to routers about entities to compute the links utilizations. The polling period can be set manually by the user.

From an implementation point of view a thread can ask periodically through SNMP protocol the information to compute the utilizations.

In the asynchronous mode the manager can asks when obtain the network status. In this way is not necessary a periodic polling. The user can decide whenever gets the links utilizations.

These two modes are implemented because a system administrator, or more in general an end user, would have either a freshness or not network status through links utilizations. The data reported to him\her are shown in a topology in which from green (low) to red (high) is reported the link utilizations.

In synchronous and asynchronous cases the network status will be available but with different method. In the first case, taken a low polling period time, i.e. high polling frequency, the network could suffer from flooding due to high requests number. Instead, in the second case the user can decide when obtain the network status. A frequent refresh can lead to a network flooding.

A good tradeoff between network traffic and network status freshness could be either a “not high” periodic polling or a refresh made “not more” frequently. In both case is up to the user decides when update the network status.

Another modus operandi chosen to monitoring the network status through links utilizations is through the VFile. The VFile is a file created within a router in which are stored the OIDs values. The VFiles are sent periodically to a server that uses either the TFTP or FTP protocol. The routers must be corrected configured in order to create, store and send to the TFTP\FTP server the VFile.

The protocol chosen for the file transport is the TFTP. The TFTP protocol is softer than FTP. The TFTP protocol doesn't require two connections (data and control) as FTP protocol. Moreover, for our purposes is neither needed the authentication nor session establishment between the TFTP server and the routers. What the routers simply do is to send VFiles to the TFTP server.

To realize the above functionality a TFTP server was implemented and the routers has been correctly configured. Moreover, in the routers configurations is possible to set, in minutes, the interval time for the VFile sending. The minimum values is one minute. For most details see the Cisco website documentation [2].

Once the VFiles arrives to the SNMP manager, the information in the VFile are extracted and processed for compute the utilization. This procedure is done for all routers in the network.

In the start phase the utilizations computation requires at least two minutes because is needed two VFiles from the same routers to compute the utilizations. In the next cases the utilizations computation requires, in the best case, one minute due to VFiles waiting.

The polling and VFiles methods both obtain the information to compute the network status. In both cases is used SNMP protocol, but in the former there is request-response mechanism, in the latter the routers, once configured, send VFile to the TFTP server.

Initially, the software has its data structure empty. Independently from the modus operandi what we need is to know are the static informations about the routers like, for instance, the interfaces speeds, their hostnames, its IP interfaces addresses and so on. Such values are requested through SNMP protocol to initialize the data structure within the software itself. Then the dynamic information (number outgoing and ingoing octets and the delta time value) can be retrieved how is preferred.

In the polling case there is an SNMP manager since that SNMP request are performed from it. It sends either asynchronously or synchronously the SNMP requests.

In the VFiles case the SNMP manager is not present. Is present a TFTP server that listens for incoming VFiles. These files are stored, processed and then deleted.

The SNMP requests and responses have less size than VFiles in which the informations are aggregated.

8.2 SNMP tunnel retrieving algorithm

Part 1: Find the configured tunnels in the router.

1.1. Data structures:

<NameIdMap> is a dictionary for mapping the unique identifier inside the "mplsTunnelTable" to the respective tunnel configuration name.

<ConfiguredTunnels> is instead one of the output data structure of the algorithm.

1.2. SNMP walk on OID "mplsTunnelName". Output rows format:

<"OID.tunnelID.tunnelInstance.sourceAddress.destAddress" = "tunnelName">

1.3. For each row regarding tunnel configurations, that is <tunnelInstance> == 0:

1.3.1. Create an entry in the <NameIdMap> dictionary with the concatenation <tunnelID.tunnelInstance> as key and <tunnelName> as value.

1.3.2. Create an entry in the <ConfiguredTunnels> dictionary with <tunnelName> as key and an instance of the object Container as value.

- 1.3.3. Add to the this given entry the attributes source and destination ip address using the values <sourceAddress> and <destAddress> respectively.

Part 2: Find the LSPs instaured in the router.

- 2.1. Data structure:
 - <LspTable> is the last output data structure of the algorithm.
 - <LspldMap > is a dictionary that keeps the mapping between the unique identifier inside the “mplsTunnelTable” and the respective LSP instance name.
- 2.2. SNMP walk on OID “mplsTunnelDescr”. Output rows format:
<”OID.tunnelID.tunnelInstance.sourceAddress.destAddress” = “tunnelDescr”>
- 2.3. For each row regarding tunnel configurations, that is <tunnelInstance> == 0:
 - 2.3.1. Retrieve the associated tunnel name using <tunnelID.tunnelInstance> as key in the dictionary <NameIdMap>.
 - 2.3.2. Use this tunnel name as key in the dictionary <ConfiguredTunnels> to obtain the container object and add to it the LSP descriptive name <tunnelDescr>.
- 2.4. For each row regarding LSP instance, that is <tunnelInstance> != 0:
 - 2.4.1. Create an entry in the <LspldMap> dictionary with the concatenation <tunnelID.tunnelInstance> as key and <tunnelDescr> as value.
 - 2.4.2. Create an entry in the <LspTable> dictionary with <tunnelDescr> as key and an instance of the object Container as value.
 - 2.4.3. Add to the this given entry the attributes source and destination ip address using the values <sourceAddress> and <destAddress> respectively.

Part 3: Retrieve the configured path relative to each tunnel configuration. Here we need to access another table starting from the main one, the “mplsTunnelTable”, for this reason we must retrieve two indices used to directly access the “mplsTunnelHopTable” entries.

- 3.1. For each key, called <K>, in the dictionary <NameIdMap>:
 - 3.1.1. Execute an SNMP walk on the OID produced by the concatenation of “mplsTunnelHopTableIndex” and the key <K>, the result is the primary index.
 - 3.1.2. Execute an SNMP walk on the OID produced by the concatenation of “mplsTunnelPathInUse” and the key <K>, the result is the secondary index.
 - 3.1.3. Access to the “mplsTunnelHopTable” through a SNMP walk adding to the OID also the two indices, and obtain the configured path.

- 3.1.4. Store the path in the dictionary <ConfiguredTunnels> using as key the tunnel name given by the <NameIdMap> with the key <K>

Part 4: Populate the LSP table with every needed information.

For the sake of simplicity the main loop is here divided in three different loops, but in the implementation there is just one loop.

- 4.1. For each key, called <L>, in the dictionary <LspIdMap>:
 - 4.1.1. Initialize a list with every OID that must be retrieved.
<InfoVect> = ["mplsTunnelSetupPrio",
"mplsTunnelHoldingPrio", "mplsTunnelRole",
"mplsTunnelAdminStatus", "mplsTunnelOperStatus",
"mplsTunnelPerfHCBytes"].
 - 4.1.2. For each element <E> in the <InfoVect>:
 - 4.1.2.1. Execute a SNMP get on the OID <E> concatenated with the LSP identifier <L>.
 - 4.1.2.2. Store the result in the <LspTable> entry, indexed by the LSP name, specified in the dictionary <LspIdMap> using as key <L>.
- 4.2. For each key, called <L>, in the dictionary <LspIdMap>:
 - 4.2.1. Execute a SNMP walk on the OID
"mplsTunnelCHopTableIndex" concatenated with the LSP id <L>.
The result is the primary index used to access directly to the rows in the table "mplsTunnelCHopTable" regarding the given LSP.
 - 4.2.2. With the primary index obtain the computed path for the LSP.
 - 4.2.3. Store the retrieved path in the <LspTable> entry, indexed by the LSP name, specified in the dictionary <LspIdMap> using as key <L>.
- 4.3. For each key, called <L>, in the dictionary <LspIdMap>:
 - 4.3.1. Execute a SNMP walk on the OID
"mplsTunnelResourcePointer" concatenated with the LSP id <L>.
The response format is the the concatenation of an OID and the needed index.
 - 4.3.2. Initialize a list of OID strings.
<InfoVect> = ["mplsTunnelResourceMaxRate",
"mplsTunnelResourceMeanRate",
"mplsTunnelResourceMaxBurstSize"].
 - 4.3.3. For each element <E> in <InfoVect>:
 - 4.3.3.1. Execute a SNMP walk on the OID <E> concatenated with the index previously obtained.
 - 4.3.3.2. Store the result in the <LspTable> entry, indexed by the LSP name, specified in the dictionary <LspIdMap> using as key <L>.

9. References

- [1]: <http://snmp.cloudapps.cisco.com/Support/SNMP/do/BrowseOID.do>
- [2]: http://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/gdatacol.html
- [3]: <https://supportforums.cisco.com/blog/154821>
- [4]: <http://pysnmp.sourceforge.net/>
- [5]: <http://www.cisco.com/c/en/us/support/docs/ip/simple-network-management-protocol-snmp/8141-calculate-bandwidth-snmp.html>
- [6]: http://www.cisco.com/c/en/us/td/docs/ios/12_2s/feature/guide/temib2.html#wp1068713