

---

# **IMUNES Manual**

---

3 April, 2019

---

# Contents

---

<b>1</b>	<b>Changelog</b>	<b>4</b>
<b>2</b>	<b>Introduction</b>	<b>5</b>
2.1	Document overview . . . . .	5
<b>3</b>	<b>User Interface Layout</b>	<b>6</b>
3.1	Toolbox . . . . .	6
3.2	Menubar . . . . .	8
3.2.1	File Menu . . . . .	8
3.2.2	Edit Menu . . . . .	9
3.2.3	Canvas Menu . . . . .	9
3.2.4	View Menu . . . . .	10
3.2.5	Tools Menu . . . . .	10
3.2.6	Topogen Menu . . . . .	11
3.2.7	Widgets Menu . . . . .	12
3.2.8	Events Menu . . . . .	12
3.2.9	Experiment Menu . . . . .	13
3.2.10	Help Menu . . . . .	13
<b>4</b>	<b>Quick Intro</b>	<b>14</b>
4.1	Simple Network Scenario . . . . .	14
4.1.1	Building a simple network . . . . .	14
4.1.2	Configuring a simple network . . . . .	16
4.1.3	Simulating a simple network . . . . .	23
4.2	Configuration files management . . . . .	26
4.2.1	Saving a virtual network configuration . . . . .	26
4.2.2	Opening a virtual network configuration . . . . .	27
<b>5</b>	<b>Advanced Usage</b>	<b>28</b>
5.1	Extended Network Scenario . . . . .	28
5.1.1	Canvas Management . . . . .	28
5.1.2	Attaching an external interface . . . . .	31
5.1.3	Attaching to a running experiment . . . . .	31
5.2	Additional Configuration . . . . .	32
5.2.1	Custom configuration . . . . .	32
5.2.2	Physical and logical interfaces . . . . .	33
5.3	Additional Tools . . . . .	35

5.3.1	Splitting a link . . . . .	35
5.3.2	Generating a network topology . . . . .	35
5.3.3	IPv4 address pool . . . . .	41
5.3.4	IPv6 address pool . . . . .	43
5.3.5	Routing protocol defaults . . . . .	43
5.4	Customizing Look . . . . .	43
5.4.1	Annotations . . . . .	43
5.4.2	Canvas background image . . . . .	46
5.4.3	Icons . . . . .	49
5.5	User-configurable Event Scheduling . . . . .	51
5.5.1	Principle of operation . . . . .	51
5.5.2	Configuring events with events editor . . . . .	52
5.5.3	Configuring events through configuration file . . . . .	53
5.6	Starting and terminating a simulation through CLI . . . . .	54
5.7	Managing virtual nodes (jails) - jls, jexec . . . . .	54
5.7.1	Examples . . . . .	55
5.8	Himage tool . . . . .	55
5.8.1	Examples . . . . .	56
5.9	Hcp tool . . . . .	56
5.9.1	Examples . . . . .	56
5.10	Example (himage and hcp) . . . . .	56
5.11	Vlink tool . . . . .	57
5.11.1	Examples . . . . .	57
<b>A</b>	<b>Installation</b>	<b>58</b>
A.1	Installation of IMUNES on FreeBSD 8 . . . . .	58
A.1.1	Installing FreeBSD . . . . .	58
A.1.2	Step by step guide through the FreeBSD installation . . . . .	58
A.1.3	Installing the FreeBSD X11 system - GUI . . . . .	59
A.1.4	Installing IMUNES . . . . .	60
A.1.5	Recompiling the FreeBSD kernel with VIMAGE support . . . . .	61
A.1.6	Running IMUNES on FreeBSD . . . . .	62
A.2	Installation of IMUNES on FreeBSD 9 . . . . .	62
A.2.1	Installing FreeBSD . . . . .	62
A.2.2	Step by step guide through the FreeBSD installation . . . . .	62
A.2.3	Installing the FreeBSD X11 system - GUI . . . . .	63
A.2.4	Installing IMUNES . . . . .	64
A.2.5	Recompiling the FreeBSD kernel with VIMAGE support . . . . .	65
A.2.6	Running IMUNES on FreeBSD . . . . .	65
A.3	Installation of IMUNES on FreeBSD 10 . . . . .	66
A.3.1	Installing FreeBSD . . . . .	66

A.3.2	Step by step guide through the FreeBSD installation . . . . .	66
A.3.3	Installing the FreeBSD X11 system - GUI . . . . .	67
A.3.4	Installing IMUNES . . . . .	68
A.3.5	Recompiling the FreeBSD kernel with VIMAGE support . . . . .	68
A.3.6	Running IMUNES on FreeBSD . . . . .	69
A.4	Running IMUNES with VMware Player . . . . .	69
A.4.1	Installing VMware . . . . .	69
A.4.2	Downloading the VMware image . . . . .	69
A.4.3	Running the VMware image . . . . .	69
A.5	Installation of the IMUNES GUI on Linux . . . . .	70
A.6	Installation of the IMUNES GUI on Windows . . . . .	71
<b>B</b>	<b>Troubleshooting</b>	<b>72</b>
B.1	Terminating all active experiments . . . . .	72
B.1.1	Cleaning up hanging ZFS mounts . . . . .	72
B.2	Restoring original ZFS snapshot . . . . .	72
B.3	Obtaining kernel panic traces . . . . .	72
<b>C</b>	<b>IMUNES network configuration file</b>	<b>74</b>

---

# 1 Changelog

---

2018-11-12 Changed images and fixed typos and errors.

2018-10-16 Preparation for newest IMUNES on FreeBSD-12.0

2015-07-22 Removed possible confusion when choosing optional dependencies: src needs to be installed to compile the kernel.

2015-07-13 Added Installation instructions for FreeBSD 10.1.

2014-09-15 Fixed textdump configuration instructions.

2014-09-09 Added a Changelog section.

2014-07-08 Reordering of enumerated items in the Install section.

2014-07-01 Updated Installation with 9.3 install steps. Added note that UnionFS is used instead of ZFS.

2013-12-12 Added/fixed some chapters in the manual and added new GUI images.

2013-12-10 Added installation instructions for FreeBSD 9.

---

## 2 Introduction

---

IMUNES is an Integrated Multiprotocol Network Emulator / Simulator of IP based networks. Virtual nodes in IMUNES are multiple network stack instances that are formed through special FreeBSD kernel modifications and Docker containers on Linux. Virtual nodes can be linked either with other virtual nodes or with the physical network interface through simulated links. All virtual nodes share a single place for their application binaries and libraries. The main strengths of this tool are high scalability, performance and fidelity.

### 2.1 Document overview

This document is intended to be a manual for users that are getting started with IMUNES, likewise for the ones that want to know more about its advanced features.

This manual is divided into three main parts: User Interface Layout, Quick Intro and Advanced Usage.

The first part, User Interface Layout, gives detailed description of IMUNES graphical user interface.

The second part, Quick Intro, is intended to prepare beginners to get a working network simulation in a short time. It gives detailed explanations for building, configuring and simulating a simple network. At the end it gives instructions related to IMUNES configuration files.

The third part, Advanced Usage, gives instructions for extending the network topology built in the first section. It also explains the usage of additional tools and configuration possibilities. It proceeds with features for customizing look, such as annotations, background image and icon size. At the end, it gives instructions related to event scheduling, starting and terminating simulation through command-line interface and `himage`, `hcp` and `vlink` commands.

---

## 3 User Interface Layout

---

IMUNES can be used either through the simple Tcl/Tk based graphical user interface (GUI) or through the command line interface (CLI).

We will run IMUNES on FreeBSD with some kind of X11 window manager and explain the main GUI components. If X11 is not running you can start it using `startx` command. To run IMUNES GUI use `imunes` command.

IMUNES GUI is a simple Tcl/Tk based management console, allowing specification and management of virtual network topologies. Its main parts are the work space in the middle, called canvas, the menubar on the top, the toolbox on the left side and the statusbar at the bottom (Figure 3.1)

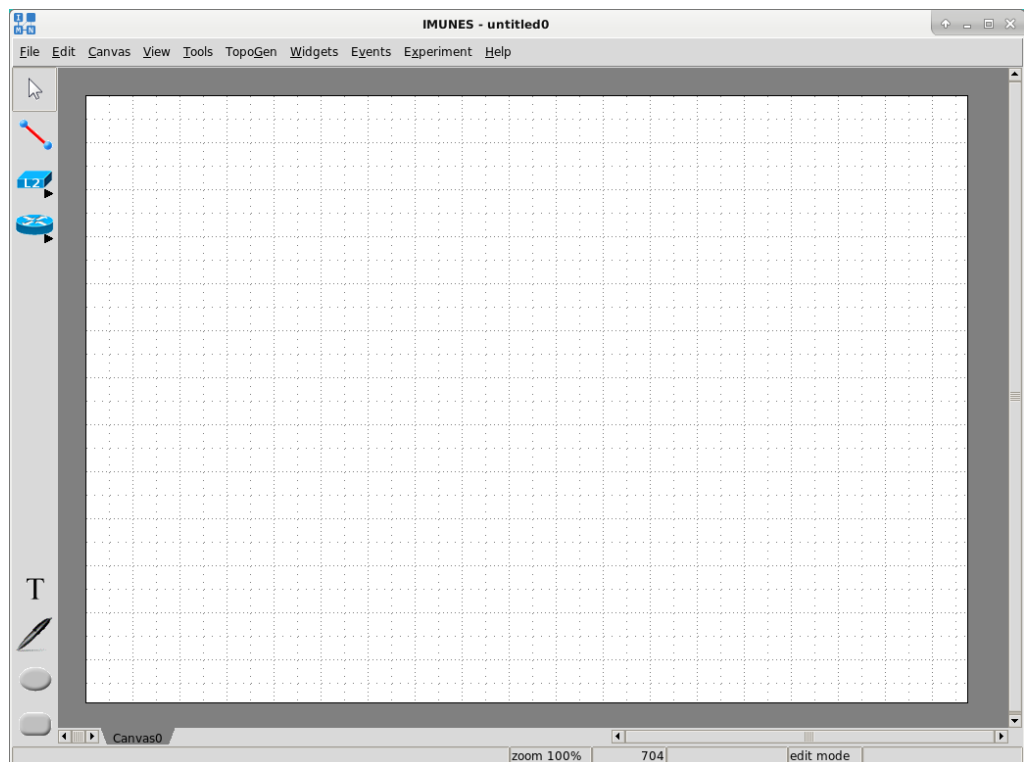


Figure 3.1: IMUNES GUI

The default working mode after the initial start (or after creating a new virtual network configuration file with the *File* → *New* option from the menubar) is edit mode. The edit mode is used to build and configure network topologies, contrary to the execute mode whose purpose is the network simulation. The network simulation will be explained later in the Section 4.1.3.

### 3.1 Toolbox

The toolbox, placed on the left side of the GUI, contains tools for building network topologies and tools for adding annotations (Figure 3.2). These tools are all available in the edit mode. In the execute mode, these tools (except for the *Select tool*) are shaded and cannot be used.

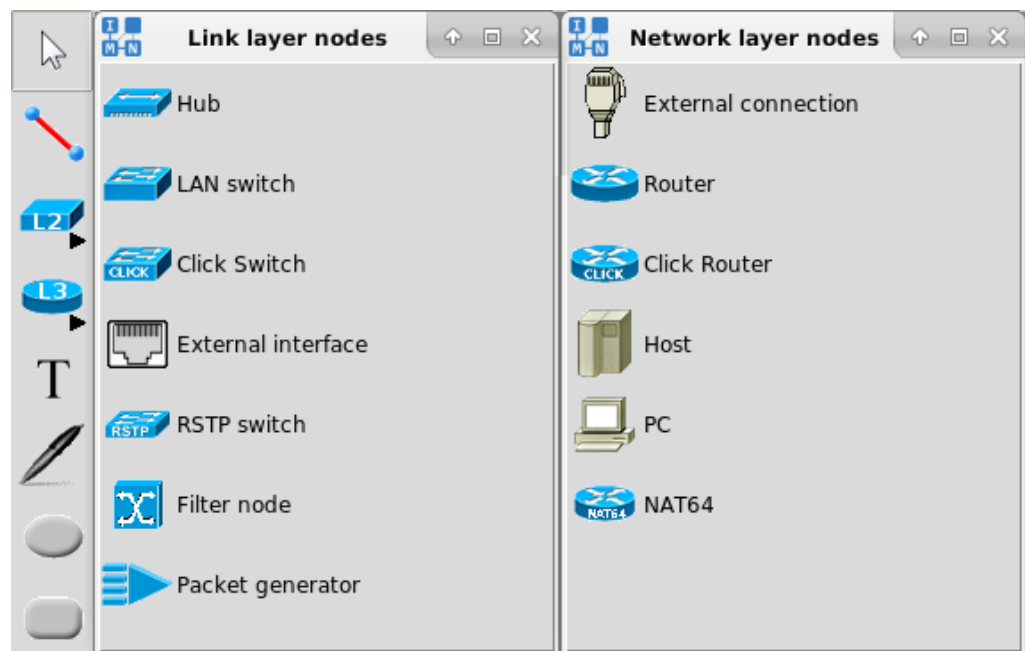


Figure 3.2: Toolbox tools

Each toolbox item shown in Figure 3.2, is described below.

#### Selecting elements:

- *Select tool* - The default tool for selecting and moving elements.

#### Building the network topology:

- *Link* - A tool that is used to create network links between nodes.

- **L2 - Link layer nodes (center column):**

- *Hub* - A link layer element that forwards every incoming packet to all of its ports and, thus, to every connected node.
- *LAN switch* - A link layer element that forwards incoming packets to connected nodes using the table of destination addresses and its ports.
- *Click switch* - A link layer element that forwards incoming packets to connected nodes using the table of destination addresses and its ports (using Click modular switch).
- *External interface* - A tool that provides the possibility to connect a virtual node with the physical interface (e.g. to give the node the access to the Internet).
- *RSTP switch* - A Rapid Spanning Tree Protocol switch that can prevent bridge loops and allow providing backup links if an active link fails.
- *Filter node* - A link layer element that can filter/divert/forward packets depending on their content.
- *Packet generator* - A link layer element to craft custom packets and send them with given packet rate.

- **L3 - Network layer nodes (right column):**

- *External connection* - A tool that provides the possibility to connect your host PC with a virtual node by creating an interface on your computer.
- *Router* - A network layer element that is capable of packet forwarding using the



routes obtained by dynamic routing protocols (available through quagga or xorp by default installation or any other standard FreeBSD routing daemon).

- *Click Router* - A network layer element that is capable of packet forwarding using the routes obtained by dynamic routing protocols (using Click modular router).
- *Host* - A network layer element that does not forward packets and has static routes. It starts standard network services, via portmap and inetd.
- *PC* - A network layer element that also does not forward packets and has static routes. Unlike host, it does not start any network services.
- *NAT64* - A router node which is capable to enable translation between IPv4 and IPv6 protocols using a form of network address translation (NAT).

#### Adding annotations:

- *Text* - A tool for adding new text on the canvas.
- *Oval* - A tool for adding a new oval shape on the canvas.
- *Rectangle* - A tool for adding a new rectangle shape on the canvas.
- *Freeform* - A tool for adding a new freeform shape on the canvas.

## 3.2 Menubar

The menubar consists of menus that provide access to various functions (Figure 3.3). Some options from the menubar are automatically disabled in the execute mode.

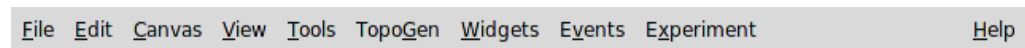


Figure 3.3: Menubar

### 3.2.1 File Menu

The *File* menu contains options for configuration files management (Figure 3.4).

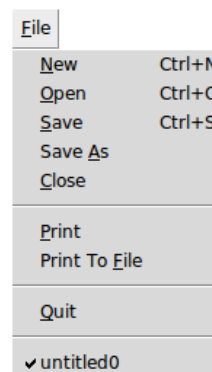


Figure 3.4: File menu

- *New* - Create a new virtual network configuration file.
- *Open* - Open an existing IMUNES network configuration file (.imn) by selecting it from the invoked *File Open* dialog.
- *Save, Save As* - Save the current virtual network topology in IMUNES network configuration file format (.imn).
- *Print* - Print the current canvas using Tcl/Tk PostScript and send it through the pipe to the default printing command (*lpr*) (that can also be changed, (e.g > *filename*)).

- *Print to file* - Print all canvases to PDF or PostScript file.
- *Close* - Close the virtual network configuration file. NOTE: If the experiment is not explicitly terminated it remains running.
- *Quit* - Exit the IMUNES GUI.
- *Recently used files* - A list of recently used files. Clicking on one of the files opens that configuration file.

### 3.2.2 Edit Menu

The *Edit* menu contains options for handling elements on the canvas.

Edit	
Undo	Ctrl+Z
Redo	Ctrl+Y
<hr/>	
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
<hr/>	
Select all	Ctrl+A
Select adjacent	Ctrl+N

Figure 3.5: Edit menu

- *Undo* - Undo the last change on the canvas reverting it to an older state.
- *Redo* - Reverse the undo command.
- *Cut, Copy, Paste* - Cut or copy elements from source and paste them to destination.
- *Select all* - Select a whole network topology.
- *Select adjacent* - Select nodes connected to the selected node(s). This feature is also available through the node menu.

### 3.2.3 Canvas Menu

The *Canvas* menu contains options for canvas management.

Canvas	
New	
Rename	
Delete	
<hr/>	
Resize	
Background image	
<hr/>	
Previous	PgUp
Next	PgDown
First	Home
Last	End

Figure 3.6: Canvas menu

- *New* - Create a new empty canvas.
- *Rename* - Rename the current canvas through the invoked dialog.
- *Delete* - Delete the current canvas.
- *Resize* - Resize the current canvas through the invoked dialog.

- *Background image* - Change background on the current canvas (see Section 5.4.2).
- *Previous, Next, First, Last* - Switch between available canvases.

### 3.2.4 View Menu

The *View* menu contains options for showing / hiding links and nodes parameters on the canvas, options for changing icon size, zooming options, etc.

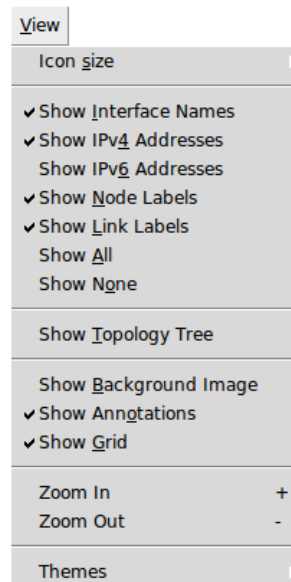


Figure 3.7: View menu

- *Icon size* - Change the size (normal or small) of all network elements (see Section 5.4.3).
- *Show [network element parameter]* - Show or hide information such as interface names, IPv4/IPv6 addresses, etc. These options are usually saved in the .imn files, providing consistent look of scenarios running on different computers.
- *Show Topology Tree* - Show or hide the tree with a list of all network topology elements.
- *Show Background Image* - Show or hide background image.
- *Show Annotations* - Show or hide annotations (text, oval, rectangle).
- *Show Grid* - Show or hide grid.
- *Zoom In, Zoom Out* - Magnify (*Zoom In*) or reduce (*Zoom Out*) the size of the display.
- *Themes* - Choose one of the themes from the submenu. Each theme represents a collection of styles, where a style describes the appearance (or appearances) of a Ttk widget class.

### 3.2.5 Tools Menu

The *Tools* menu contains the network topology management tools.

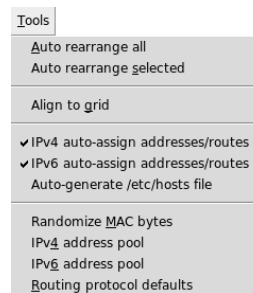


Figure 3.8: Tools menu

- *Auto rearrange all* - Automatically rearrange position of all network elements on canvas.
- *Auto rearrange selected* - Automatically rearrange position of the selected group of network elements.
- *Align to grid* - Arrange all network elements on canvas aligning them to grid.
- *IPv4 auto-assign addresses/routes* - Automatically assign IPv4 addresses and routes upon creating a new node.
- *IPv6 auto-assign addresses/routes* - Automatically assign IPv6 addresses and routes upon creating a new node.
- *Auto-generate /etc/hosts file* - Create a *hosts* file on every node and map every other node hostname with its address.
- *Randomize MAC bytes* - Randomizes the 4th and 5th byte of the automatically generated MAC address.
- *IPv4 address pool* - Set variable-mask IPv4 address pool through the invoked dialog in order to replace default 10.0.0.0/24 address pool (see Section 5.3.3). This will be applied to all the subsequently created network layer elements.
- *IPv6 address pool* - Set variable-mask IPv6 address pool through the invoked dialog in order to replace default fc00::/64 address pool (see Section 5.3.4). This will be applied to all the subsequently created network layer elements.
- *Routing protocol defaults* - Set the routing protocol defaults (routing model and protocols) through the invoked dialog (see Section 5.3.5). This will be applied to all selected routers (if any) at the time of change, as well as to all the subsequently created ones.

### 3.2.6 Topogen Menu

The *TopoGen* menu contains options for simple and fast specification of various network topologies (see Section 5.3.2).

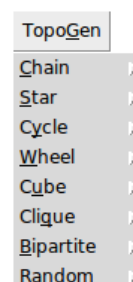


Figure 3.9: Topogen menu

### 3.2.7 Widgets Menu

The *Widgets* menu contains options for displaying information about the virtual network. To see these information, place the mouse pointer on the virtual node of interest after selecting a widget.

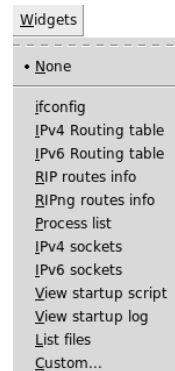


Figure 3.10: Widgets menu

- *None* - Do not show any information about the virtual network.
- *ifconfig* - Show network interfaces parameters.
- *IPv4 Routing table* - Show the IPv4 routing table.
- *IPv6 Routing table* - Show the IPv6 routing table.
- *RIP routes info* - Show the RIP routing information.
- *RIPng routes info* - Show the RIPng routing information.
- *Process list* - Show the running processes.
- *IPv4 sockets* - Show the IPv4 sockets.
- *IPv6 sockets* - Show the IPv6 sockets.
- *View startup script* - Show the startup script /boot.conf
- *View startup log* - Show the startup log /out.log
- *Custom...* - Allows the specification of the command that will be executed inside a virtual node upon placing the mouse pointer on the virtual node. The result of the command will be displayed inside the widget.

### 3.2.8 Events Menu

The *Events* menu - This menu is used to configure event scheduling. The event scheduling will be explained later in the Section 5.5.

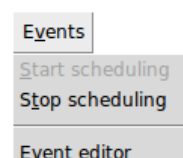


Figure 3.11: Events menu

- *Start scheduling* - Start the scheduling of events.
- *Stop scheduling* - Stop the scheduling of events.
- *Event editor* - Schedule events on the links through the opened editor.

### 3.2.9 Experiment Menu

The *Experiment* menu is used to start and terminate an experiment. It also enables to attach to a running experiment.

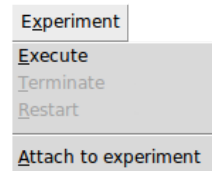


Figure 3.12: Experiment menu

- *Execute* - Start an experiment and switch to the execute mode. In the process of starting an experiment, IMUNES creates and configures the virtual network. All events during that process will be shown in the statusbar.
- *Terminate* - Terminate an experiment and switch to the edit mode. During the termination process, IMUNES will shut down all network elements and it will terminate active services on each node. The termination is finished when the message about the successful cleanup shows up in the statusbar.
- *Restart* - Terminate and restart the running experiment.
- *Attach to experiment* - This option opens a window with the list of running experiments on the current computer. It allows to resume running experiments that are shown in the *Attach to experiment* window shown in Figure 5.9.

### 3.2.10 Help Menu

The *Help* menu contains the option *About* that invokes the *About* dialog box for viewing version information.



Figure 3.13: Help menu

---

## 4 Quick Intro

---

### 4.1 Simple Network Scenario

In this section we will show how to build, configure and simulate the following simple network topology:

Personal computers (*office-pc1* and *office-pc2*) from the network 192.168.1.0/24 are connected to the LAN switch (*office-switch*) which is connected to the router (*office-router*). The server (*office-host*) from the network 192.168.2.0/24 is directly connected to the router (*office-router*). Personal computers from the first network have route only to the network 192.168.2.0/24. The server from the second network has the default route. Quagga routing is enabled on the router in order to be able to serve and receive dynamic route updates.

#### 4.1.1 Building a simple network

After running IMUNES on FreeBSD with some kind of X11 window manager (see Section 3), we will build previously described network using tools from the toolbox (see Section 3.1).

##### Adding and deleting network elements

To draw a node click on the corresponding node tool and then click on the workspace to place it. To connect nodes click on the *Link tool*, then click and hold on the source node and go to the destination node.

Now draw a router, a host, a LAN switch and two PCs. Using the *Link tool* connect the LAN switch to the router and then connect each PC to the LAN switch. Connect the host directly to the router. The created network topology should look like the one in Figure 4.1.

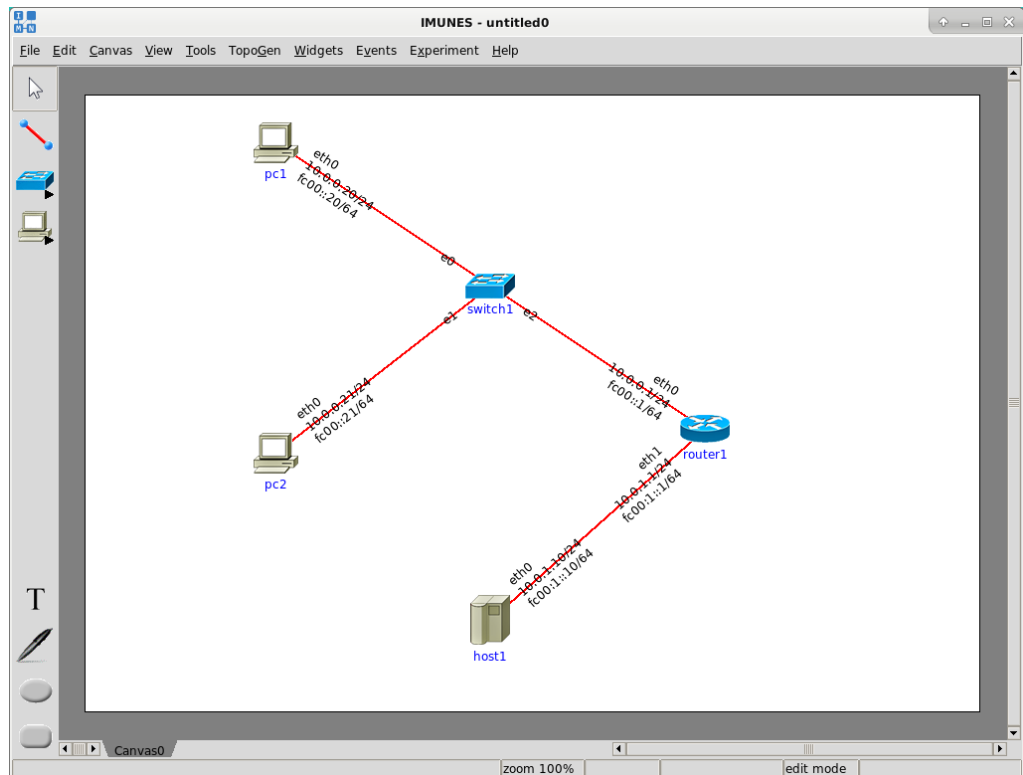


Figure 4.1: Simple network topology

When nodes are connected with the *Link tool* (the direction does not matter), the source node, the destination node and the link automatically get preconfigured parameters. When a mouse pointer is above a node or a link, some of the configured parameters are shown on the left side of the statusbar placed at the bottom of the window (Figure 4.2).



Figure 4.2: Node parameters in the statusbar

Some of the parameters can be visible on the canvas: interface names (link layer: e0, e1, e2 and network layer: eth0, eth1), IPv4/IPv6 addresses of network layer elements (PC, host, router), node names (router1, host1, switch1, pc1, pc2) and link labels (Bandwidth, Delay, BER or Duplicate if their values are not default).

You can manipulate with the visibility of nodes and links parameters from the View menu (Figure 4.3). In this simple scenario we do not want for IPv6 addresses to be visible, so we will turn the *Show IPv6 Addresses* option off.



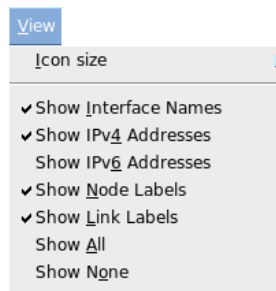


Figure 4.3: Show or hide nodes and links parameters

To delete the network element select it using the *Select tool* and then use the *Delete* keyboard button. You can also delete it by right clicking on it and clicking on the *Delete* label in the popped up menu. The node deletion is automatically followed by the deletion of associated links.

### Rearranging network elements

You can change position of the network element (node or link) and/or the node name. To move both the network element and its name select the network element with the *Select tool* and drag it to the designated position. To move only the node name select it with the *Select tool* and drag it to the designated position.

Using the *Select tool* you can also move around a group of connected nodes which can be selected using the *Ctrl* keyboard button in addition to the left click. To select the whole network topology use *Select All* option from the *Edit* menu.

For automatic rearranging of all network elements or rearranging the selected group of network elements use *Rearrange* and *Rearrange All* options from the *Tools* menu. To stop the rearranging process click with the *Select tool*.

#### 4.1.2 Configuring a simple network

Although preconfigured parameters of network elements are usually sufficient to start a simulation (automatically provided IPv4/IPv6 addresses, the default static route on the PC and the host and routing model and protocols parameters on the router as well), in this scenario we will set up our own parameters.

To open the network element configuration window:

- right click on the network element and select the *Configure* label from the popped up menu (Figure 4.4)
- or
- double click on the network element.

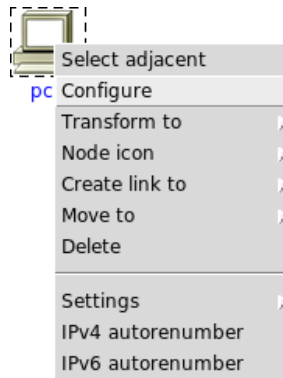


Figure 4.4: Configure a network element

Network elements configuration parameters can be also changed through the topology tree. To show the topology tree turn on the *Show Topology Tree* option from the *View* menu. The tree with a list of network topology elements (nodes and links) will be shown on the right side of the window (Figure 4.5). To open the network element configuration window double click or use the *Enter* keyboard button on node, interface or link label in the topology tree.

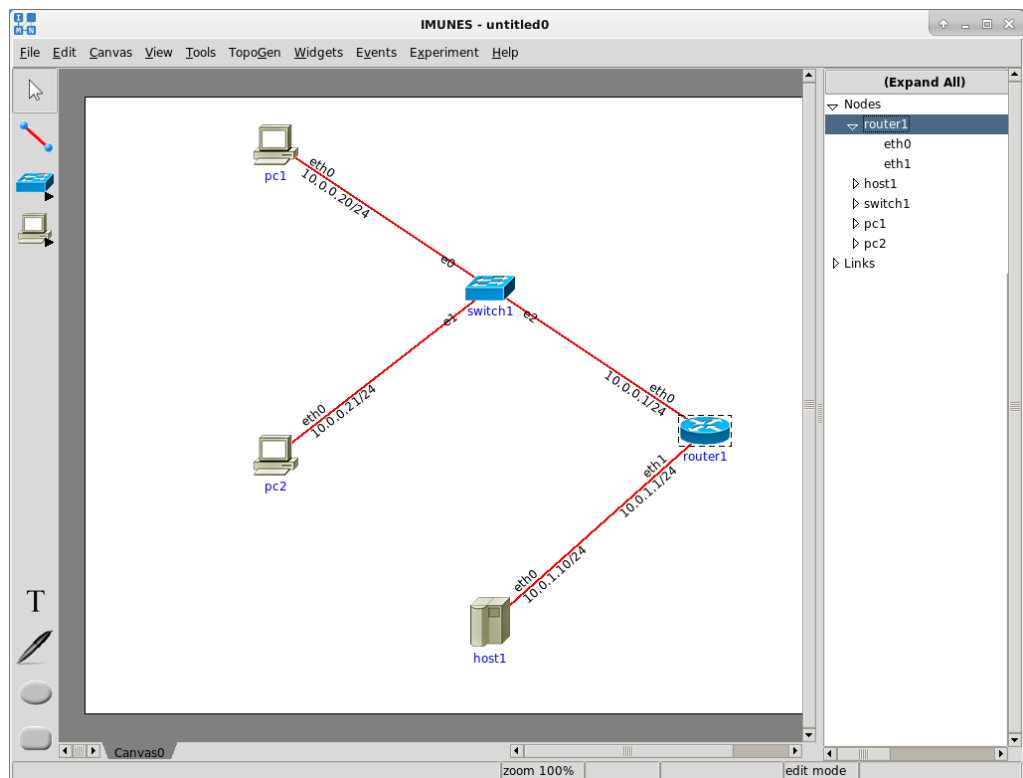


Figure 4.5: Changing configuration parameters through the topology tree

Depending on the type of a network element in our topology, there are 4 types of configuration windows:

- *Hub/LAN switch configuration window*
- *PC/Host/Click router configuration window*
- *Router configuration window*

- *Link configuration window*

There are also other types of configuration windows which are explained in other sections:

- *External interface configuration window*
- *External connection configuration window*
- *NAT64 configuration window*
- *RSTP switch configuration window*
- *Filter node configuration window*
- *Packet generator configuration window*
- *Click switch configuration window*

### Hub/LAN switch configuration window

The *hub/LAN switch configuration* window, as well as the configuration windows of other node types, contains a node name field. Besides that it contains only link layer interface parameters.

We will change the LAN switch name and data packet scheduling method (from preconfigured First In First Out (FIFO) data packet scheduling method to Weighted Fair Queuing (WFQ) method).

Change the node name to *office-switch*. To change data packet scheduling method select the link layer interface *e0* from the list of interfaces, choose *WFQ* option from the *Queue* menu and click on the *Apply* button (Figure 4.6).

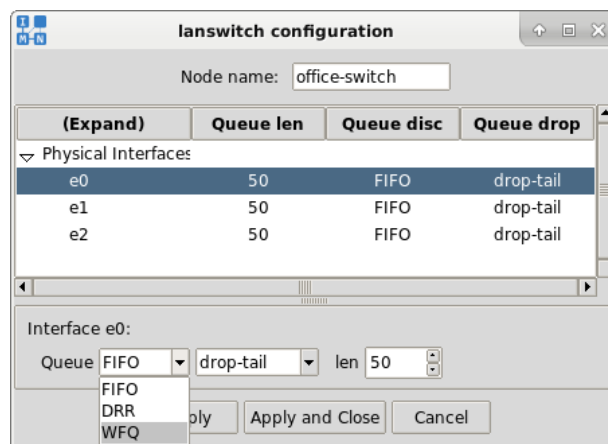


Figure 4.6: LAN switch configuration window

Packet scheduling method is now applied and you can see new queuing discipline for interface *e0* in the column *Queue disc* (Figure 4.7).

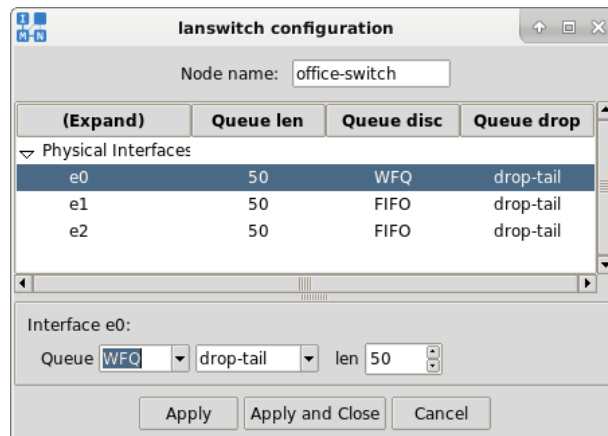


Figure 4.7: LAN switch configuration window with applied changes

Repeat the same procedure for the other link layer interfaces. Changed configuration is already applied so you can close the configuration window with the *Cancel* button but you can also use the *Apply and Close* button.

### PC/Host/Click router configuration window

The *PC/Host/Click router configuration* window consists of two subwindows. Each of them is associated with one of the following tabs: *Configuration* and *Interfaces* (Figure 4.8).

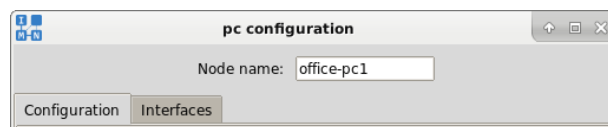


Figure 4.8: Tabs in the PC/Host/Click router configuration window

Besides a node name field, *PC/Host/Click router configuration* window contains startup services, routing parameters and custom configuration parameters (in the window associated with the *Configuration* tab) and network interface parameters (in the window associated with the *Interfaces* tab).

We will change the node name, network interface parameters and routing parameters.

Change the host node name to *office-host* and PC node names to *office-pc1* and *office-pc2*. To change IPv4 address left click on the *Interfaces* tab, select interface *eth0* from the list of interfaces, change the IPv4 address field and click on the *Apply* button (Figure 4.9). We will change the host IPv4 address field to 192.168.2.5/24 (now it belongs to 192.168.2.0/24) and PC IPv4 address fields to 192.168.1.5/24 and 192.168.1.7/24 (now they belong to network 192.168.1.0/24). IP address fields require the CIDR notation, so the IPv4 address is followed by a slash and a network length.

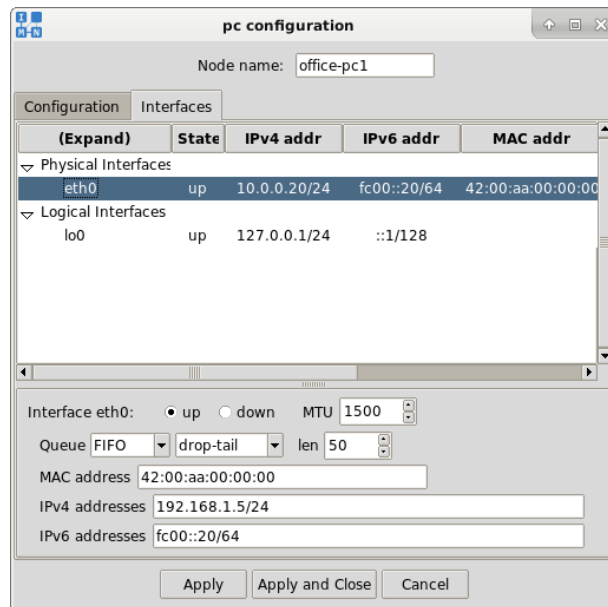


Figure 4.9: Changing IPv4 address

### Static routes

PCs and Hosts both use static routing. The preconfigured routing table contains only the default route. Every static route, as well as the default route, consists of:

1. the destination network: an IP address which is followed by a slash and a network prefix and
2. the next hop network interface IP address (which is an IP address without a slash and without a network prefix).

If the route syntax is wrong, that route will be silently ignored.

We will add the static route on *office-pc1* and *office-pc2* for the network 192.168.2.0/24 through the gateway 192.168.1.1 (Figure 4.10).

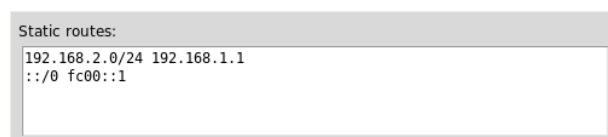


Figure 4.10: Adding the static route on the PC

On *office-host* we will change default gateway address to 192.168.2.1 (Figure 4.11).



Figure 4.11: Adding the static route on the PC

IPv6 addresses and default routes (placed below IPv4 addresses and routes) can be deleted.

To apply the changed configuration and close the configuration window click on the *Apply and Close* button.

## Router configuration window

The *router configuration window*, in addition to fields from *PC/Host/Click router configuration* window, contains the part for choosing the routing model and protocols, as well as an *IPsec* tab with IPsec parameters (See IPsec Section TODO).

We will only change the node name and network interface parameters.

Change the node name to *office-router* and IPv4 addresses on both network interfaces: 192.168.1.1/24 on the network interface *eth0* and 192.168.2.1/24 on the network interface *eth1*.

### Routing models and protocols

There are three possible routing models:

1. the quagga model
2. the xorp model (eXtensible Open Router Platform)
3. the static model

In the case of quagga and xorp routing models, there are options for enabling/disabling RIP, RIPng, OSPFv2 and OSPFv3. By default, all new quagga or xorp router instances will have both RIPv2 and RIPng enabled. The defaults can be changed with the *Tools* → *Routing protocol defaults* option from the menubar, which will be applied to all selected routers (if any) at the time of change, as well as to all the subsequently created ones (see Section 5.3.5). In the case of static routing model, the router uses routes from the static routes field that has the same syntax as the static routes field in the *PC/Host/Click router configuration* window.

We will leave the default router model - quagga with RIP and RIPng protocols enabled, and OSPFv2 and OSPFv3 protocols disabled (Figure 4.12).

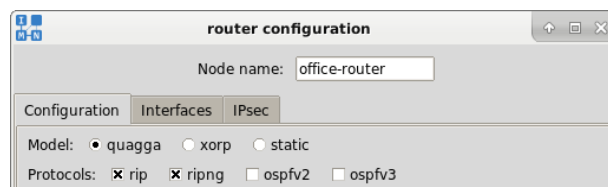


Figure 4.12: Routing models and protocols

## Link configuration window

The *link configuration* window offers the possibility to configure the link bandwidth (between 0 and  $10^9$  bps), the propagation delay (between 0 and  $10^7 \mu s$ ), the bit error rate (between 0 and  $10^{12}$ ) and the probability of package duplication (between 0 and 50%). There are also display properties: the link width (line thickness between 1 and 8) and the link color (red, green, blue, yellow, magenta, cyan or black).

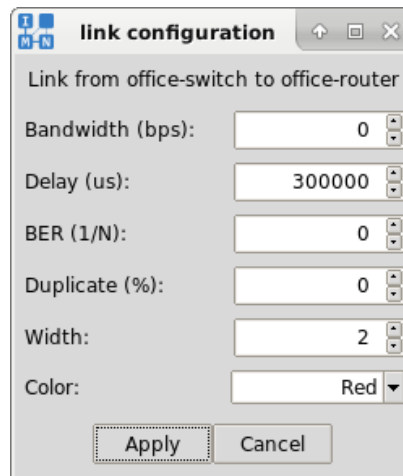


Figure 4.13: Link configuration window

Default values are as follows: the link which transmits packets without errors and without any possibility for the packet duplication with the unlimited link bandwidth and the zero propagation delay. The link width is set to value 2 and the link color is red.

We will leave default values on all links except on the link between *office-switch* and *office-router* (Figure 4.14). On that link we will set up the delay of 30000  $\mu$ s. Delay will be tested during the network simulation with the *traceroute* tool (see Section 4.1.3).

### Configured network topology

Configured network topology should look like the one in Figure 4.14.

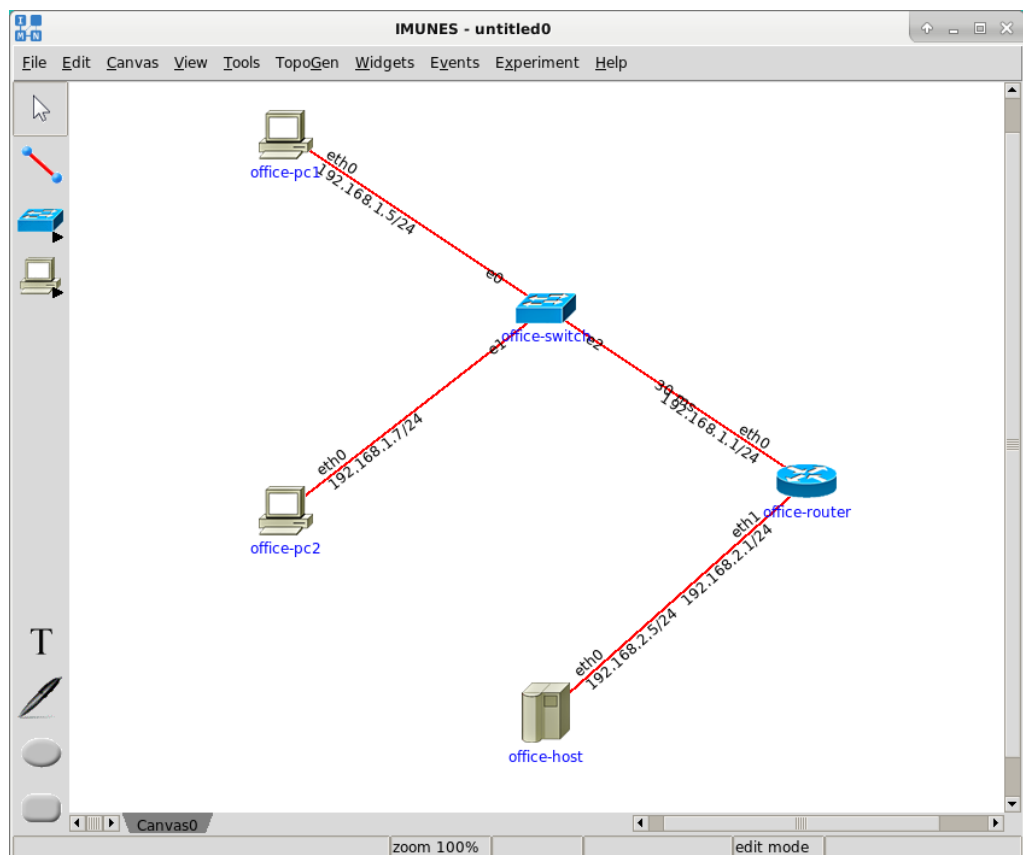


Figure 4.14: Configured network topology

### 4.1.3 Simulating a simple network

#### Starting an experiment

After the network topology is completely built and properly configured, we will start an experiment with the *Experiment* → *Execute* option from the menubar and IMUNES will switch from the edit mode to the execute mode. In the process of starting an experiment, IMUNES creates and configures the virtual network. That will take a few seconds and all events during that process will be shown in the statusbar placed at the bottom of the window.

**NOTE:** Although you can draw network topology on any system that supports Tcl/Tk (Linux, FreeBSD, Windows, Mac OS X, Solaris), an experiment can only be started on FreeBSD and Linux operating systems with root permissions (Figure 4.15 and Figure 4.16)!

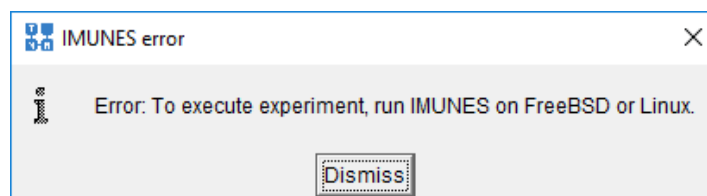


Figure 4.15: Starting an experiment in Windows





Figure 4.16: Starting an experiment in FreeBSD without root permissions

In addition to configured parameters, each node will be set with the loopback interface, a router will have the kernel forwarding enabled, and a host node will have portmap and inetd started.

Information about the time spent instantiating the network topology is shown in the statusbar (Figure 4.17).



Figure 4.17: Message about the instantiation of the network topology

In the right corner of the statusbar you can also see that IMUNES now works in the execute mode, as well as experiment unique identifier.

### Options from the node and the link menu

To open the node menu in the execute mode right click on the node. Note that the menu in the execute mode is different from the menu in the edit mode. It offers the possibility to select the node connected to this node (*Select adjacent*), to see the current configuration (*Configure*), to *Start / Stop / Restart* the network element, to start / stop / restart any of the possible *Services* or to *Import Running Configuration* from the *Settings* menu. The *Import Running Configuration* option copies the current MTU value and IPv4/IPv6 addresses from the running node to its configuration. It is also possible to open the *Shell window* (X terminal with a Unix shell), *Wireshark* or *tcpdump* network sniffers on any of the interfaces, *Firefox Web Browser* or a *Mail client*.

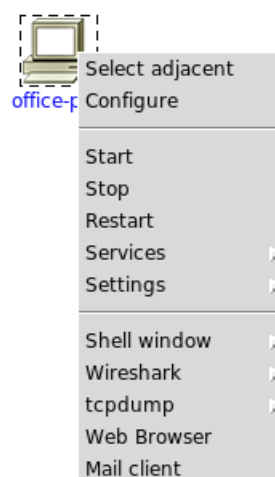


Figure 4.18: Network-layer node menu in the execute mode

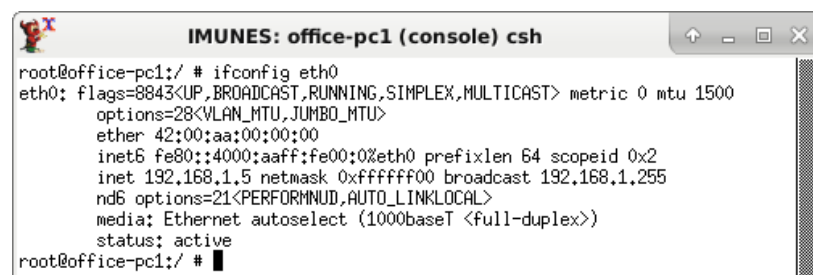
Note that both the node and the link menu in the execute menu offer the possibility to open the configuration window (*Configure* label).

From the node configuration window in the execute mode it is possible to change only the node name. Other node parameters such as link layer interface parameters, network interface parameters and routing parameters can be changed from the shell window on each node. To change those parameters from the node configuration window, stop the node (using the *Stop* label), change parameters and then start the node again (using the *Start* label).

On the other side, from the link configuration window in the execute mode it is possible to change the following link parameters: link bandwidth, the propagation delay, the bit error rate and the probability of package duplication. It is also possible to change display properties: the link width and the link color.

We will now check if the virtual network topology is properly configured. Open the shell window (e.g. *Shell window* → *csch* or simply double click on the node) on the network element (e.g. *office-pc1*).

- To check the network interface *eth0* parameters type the following command: `ifconfig eth0`. The result is shown in Figure 4.19.



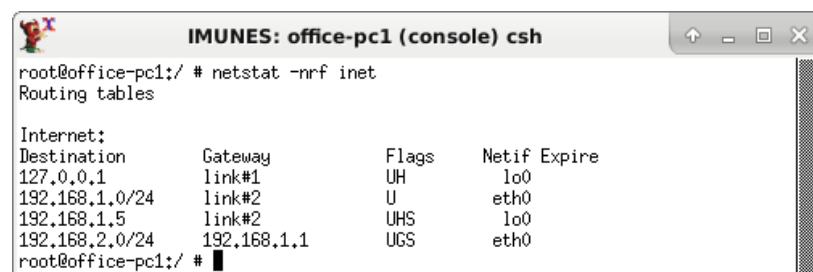
```

root@office-pc1:/ # ifconfig eth0
eth0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=28<VLAN_MTU,JUMBO_MTU>
ether 42:00:aa:00:00:00
inet6 fe80::4000:aaff:fe00:0%eth0 prefixlen 64 scopeid 0x2
inet 192.168.1.5 netmask 0xfffff00 broadcast 192.168.1.255
nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
media: Ethernet autoselect (1000baseT <full-duplex>)
status: active
root@office-pc1:/ #

```

Figure 4.19: Shell window on *office-pc1*, network interface parameters

- To check static routes type the following command: `netstat -nrf inet`. The result is shown in Figure 4.20.



```

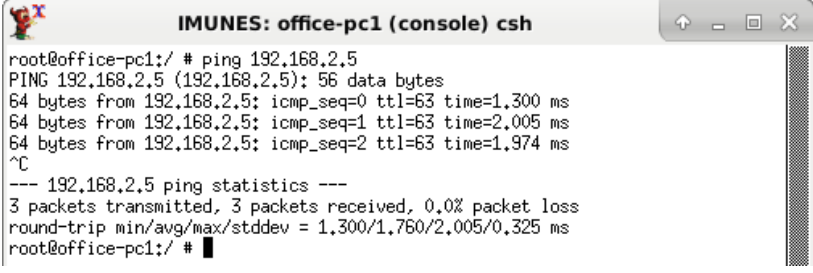
root@office-pc1:/ # netstat -nrf inet
Routing tables

Internet:
Destination      Gateway          Flags      Netif Expire
127.0.0.1         link#1          UH         lo0
192.168.1.0/24    link#2          U          eth0
192.168.1.5       link#2          UHS        lo0
192.168.2.0/24    192.168.1.1     UGS        eth0
root@office-pc1:/ #

```

Figure 4.20: Shell window on *office-pc1*, static routes

- To test if a particular network element is reachable (e.g. *office-host*) type the following command: `ping 192.168.2.5`. The result is shown in Figure 4.21. To stop transmitting packets press *Control-C* keyboard button.



```

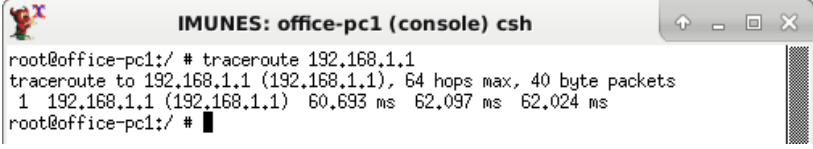
IMUNES: office-pc1 (console) csh
root@office-pc1:/ # ping 192.168.2.5
PING 192.168.2.5 (192.168.2.5): 56 data bytes
64 bytes from 192.168.2.5: icmp_seq=0 ttl=63 time=1.300 ms
64 bytes from 192.168.2.5: icmp_seq=1 ttl=63 time=2.005 ms
64 bytes from 192.168.2.5: icmp_seq=2 ttl=63 time=1.974 ms
^C
--- 192.168.2.5 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.300/1.760/2.005/0.325 ms
root@office-pc1:/ #

```

Figure 4.21: Shell window on office-pc1, pinging office-host

We will test delay on the link between *office-switch* and *office-router*, which is set to 30000  $\mu$ s (30 ms), by using the *traceroute* tool:

- In the shell window on *office-pc1* type the following command:  
`traceroute 192.168.1.1`. The result is shown in Figure 4.22.



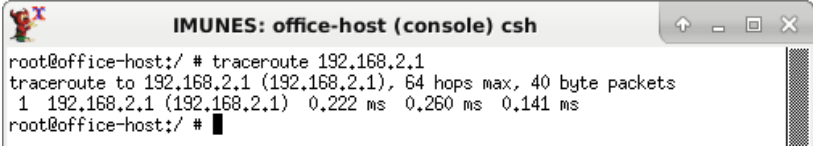
```

IMUNES: office-pc1 (console) csh
root@office-pc1:/ # traceroute 192.168.1.1
traceroute to 192.168.1.1 (192.168.1.1), 64 hops max, 40 byte packets
1 192.168.1.1 (192.168.1.1) 60.693 ms 62.097 ms 62.024 ms
root@office-pc1:/ #

```

Figure 4.22: Shell window on office-pc1, traceroute to office-router

- In the shell window on *office-host* type the following command:  
`traceroute 192.168.2.1`. The result is shown in Figure 4.23.



```

IMUNES: office-host (console) csh
root@office-host:/ # traceroute 192.168.2.1
traceroute to 192.168.2.1 (192.168.2.1), 64 hops max, 40 byte packets
1 192.168.2.1 (192.168.2.1) 0.222 ms 0.260 ms 0.141 ms
root@office-host:/ #

```

Figure 4.23: Shell window on office-host, traceroute to office-router

## Terminating an experiment

To terminate an experiment and switch from the execute mode to the edit mode use the *Experiment* → *Execute* option from the menubar. During the termination process, IMUNES will terminate active services on each node and shut down all network elements (links and nodes with both virtual and physical interfaces). The termination is finished when the message about the successful cleanup shows up in the statusbar (Figure 4.24).



Figure 4.24: Message about the successful cleanup

## 4.2 Configuration files management

### 4.2.1 Saving a virtual network configuration

After the virtual network is successfully built, configured and tested, it can be saved with *File* → *Save* or *File* → *Save As* options from the menubar. The virtual network topology is saved in IMUNES network configuration file format (.imn).

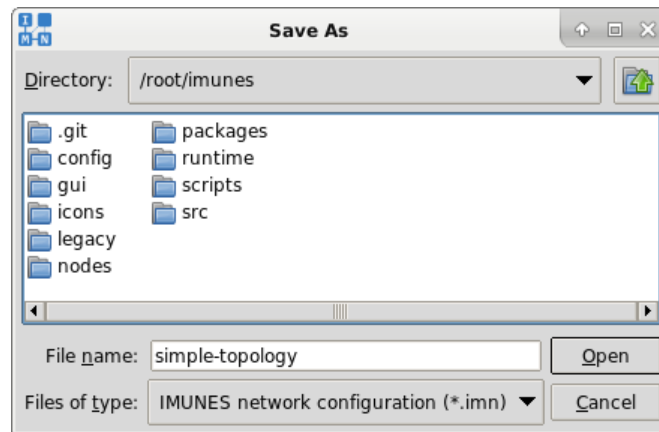


Figure 4.25: File Save dialog

The structure of the configuration file is simple and suitable for changing with a text editor (see Appendix C).

#### 4.2.2 Opening a virtual network configuration

To open an existing IMUNES network configuration file use the *File* → *Open* option from the menubar and select it from the invoked *File Open* dialog.

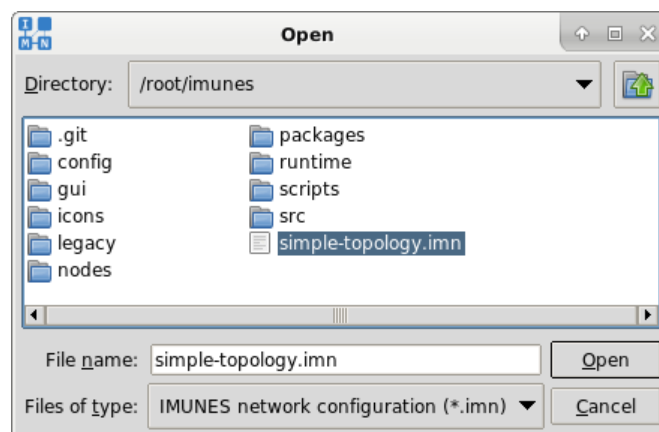


Figure 4.26: File Open dialog

The other way to open an imn file is to start IMUNES with that file as an argument: `imunes simple-topology.imn`

---

## 5 Advanced Usage

---

### 5.1 Extended Network Scenario

In this section we will show how to extend the simple network topology (see Section 4.1). We will explain the additional possibilities of building and configuring the extended network scenario:

The simple network topology is placed in the *office-canvas* while the additional elements are placed in the *roadwarrior-canvas*. The *roadwarrior-canvas* consists of a router (*roadwarrior-router*), a PC (*roadwarrior*) and a physical interface. The *roadwarrior-router* is connected to the *office-router* on the *office-canvas* through the 192.168.3.0/24 network, the PC is in the 161.53.19.0/24 network whereas the physical interface is in the 161.53.20.0/24 network.

We will now extend the simple network from the last chapter by adding the *roadwarrior-router*, *roadwarrior* and the physical interface all placed on another canvas. To open an existing IMUNES network configuration file use *File* → *Open* from the menubar or start IMUNES with the *imn* file as an argument: `imunes simple-topology.imn`. Check that you are in the edit mode. If not, switch with *Experiment* → *Terminate*.

#### 5.1.1 Canvas Management

To facilitate building of complex and large network topologies IMUNES lets you divide the network topology into a set of network layers. These network layers are called canvases.

Canvas management consists of two main elements:

- Canvas menu in the menubar (Figure 3.6)
- List of canvas tabs at the bottom of the main window, above the statusbar (Figure 5.1)

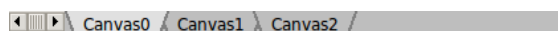


Figure 5.1: Canvas Tabs

To add a new canvas use the *Canvas* → *New* option from the menubar or double click on the empty space in the canvas tabs list at the bottom of the window.

You can rename the canvas with the *Canvas* → *Rename* option from the menubar or double click on the canvas tab in the canvas tabs list. (Figure 5.2) Similarly the *Canvas* → *Delete* option deletes the active canvas.

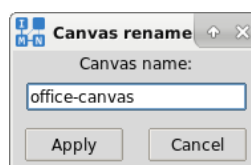


Figure 5.2: Canvas rename dialog

There is also the option *Canvas* → *Resize* that allows you to define a custom canvas size in pixels. The default canvas size is 900\*620 pixels. (Figure 5.3)

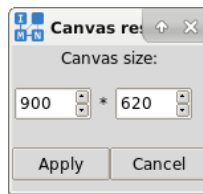


Figure 5.3: Canvas resize dialog

Canvas selection can be done with the options from the *Canvas* menu (*Previous*, *Next*, *First*, *Last*) or simply by clicking the tab with the canvas name on it.

Rename the existing canvas *Canvas0* into *office-canvas*. Add a new canvas, rename it into *roadwarrior-canvas* and select it as the active canvas.

This canvas is empty so we will add a router by selecting the router tool and clicking on the empty canvas. Rename this router into *roadwarrior-router*. Switch to the *office-canvas*. Now we will connect the *office-router* and the *roadwarrior-router*. To do that, right click on the *office-router* and select *Create link to* → *roadwarrior-canvas* → *roadwarrior-router* option (Figure 5.4) from the popped up menu. This will create a link between *roadwarrior-router* and *office-router*.

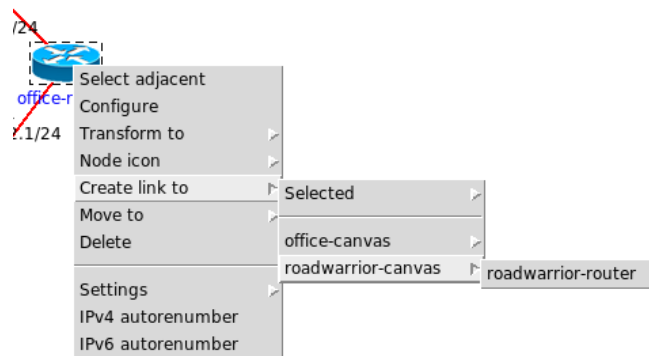


Figure 5.4: Create link to

On the *office-router* set the *eth2* interface IPv4 address to 192.168.3.1/24. On the *roadwarrior-router* set the *eth0* interface IPv4 address to 192.168.3.2/24. We will add another PC to the *roadwarrior-canvas*, name it *roadwarrior* and connect it with the *roadwarrior-router*. On the *roadwarrior* set the *eth0* IPv4 address to 161.53.19.100/24. On the *roadwarrior-router* set the *eth1* IPv4 address to 161.53.19.1/24.

The *roadwarrior-router* uses the same dynamic routing model (quagga) as the *office-router* and we do not need to configure anything else on the router. The *roadwarrior* uses static routes and we will need to change the default route gateway in static routes field of the *roadwarrior* configuration window to 0.0.0.0/0 161.53.19.1.

Finally, the configured network topology should look like the following (Figure 5.5 and Figure 5.6):

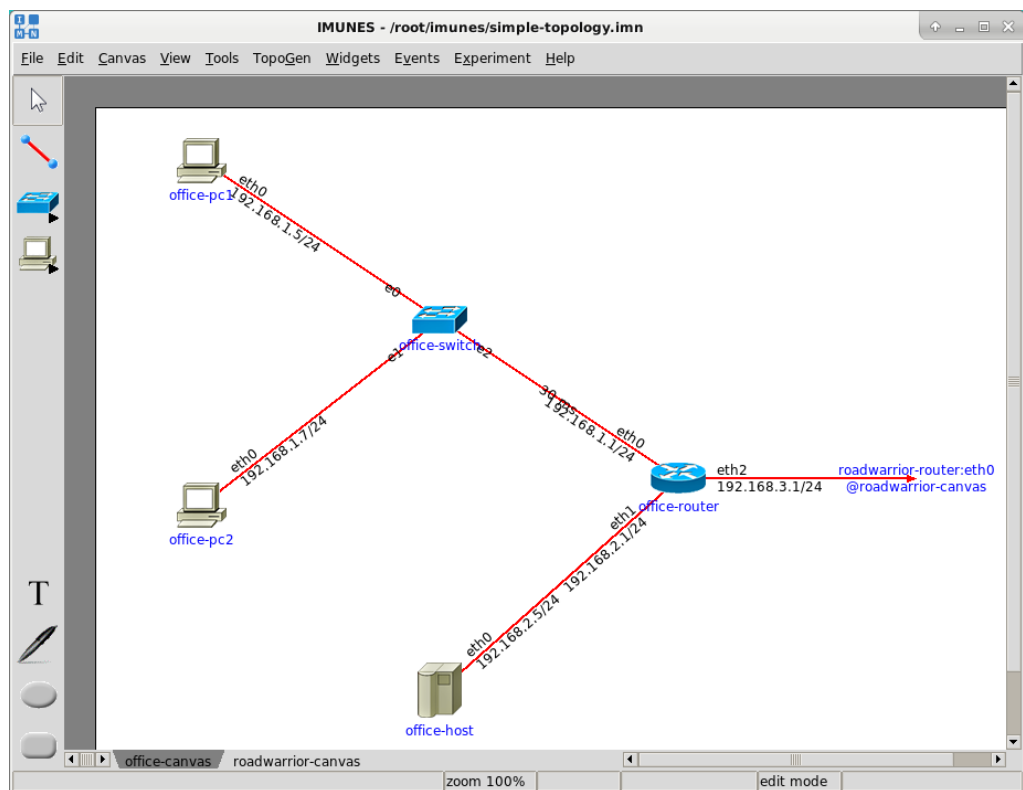


Figure 5.5: office-canvas

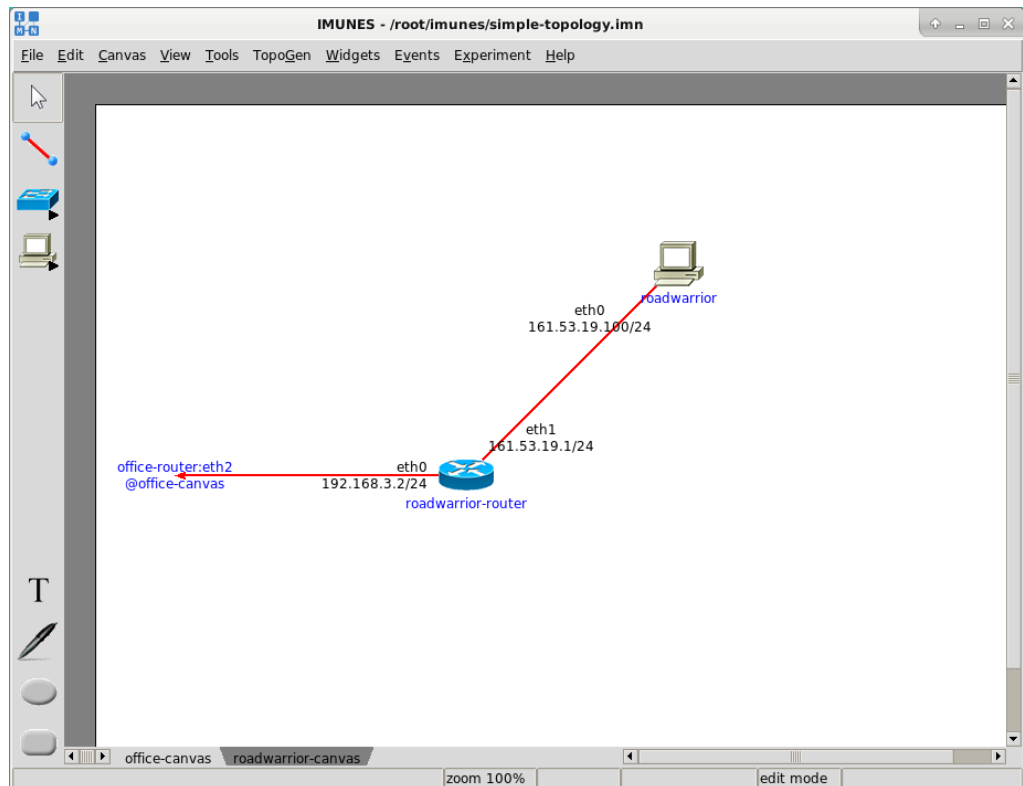


Figure 5.6: roadwarrior-canvas

Both the *roadwarrior* and *roadwarrior-router* can be easily moved from *roadwarrior-canvas*

to *office-canvas* with the *Move To* → *office-canvas* from the node menu. The link between *roadwarrior-router* and *office-router*, as well as any other link, can be deleted with the *Delete* option from the *Link* menu. To open the *link* menu, use the right click on the link and choose the *Delete* option.

When we are done with network configuration, we can start the experiment with *Experiment* → *Execute* from the menubar. We can now check that the *roadwarrior* can ping both networks (192.168.1.0/24, 192.168.2.0/24) and additionally, that the network 192.168.1.0/24 does not have an access to the *roadwarrior*, but it has access to the 192.168.2.0/24 network.

### 5.1.2 Attaching an external interface

The *External interface* tool from the toolbox provides the possibility to attach a physical interface to a virtual node. This way the virtual network is able to communicate with nodes from the external network.

We will now add the external interface to the *roadwarrior-canvas*. To add a external interface to the canvas select the *External interface* tool and click on the canvas.

*External interface* nodes can be connected either with the LAN switch or the router. Connect the newly created external interface with the virtual node *roadwarrior-router* using the *Link* tool from the toolbox or the *Create link to* option from the node menu.

The newly created external interface node is unassigned and in the node name field it contains *UNASSIGNED*. Open the external interface configuration window with a double click or right click on it and select the *Configure* option. Choose the name of the designated physical interface in the *Physical interface* drop-down list, e.g. *em1*, obtained from *ifconfig* –a command ran on the physical machine (outside IMUNES). (Figure 5.7) The name of the physical interface will appear in the node name label. It is also possible to assign a VLAN tag to it. (Figure 5.8)

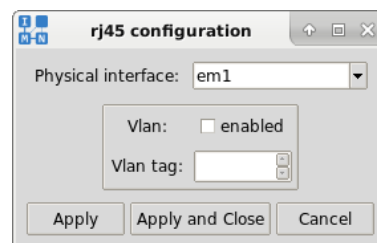


Figure 5.7: External interface configuration dialog



Figure 5.8: External interface node label

Check that *roadwarrior-router* has a properly configured IP address on the network interface connected to the physical interface. Additionally, check that routes which route packets between virtual network and the external network through the physical interface exist in both the external network and in the virtual network (on *roadwarrior-router*).

Save this configuration to a new file by selecting the *Save as* option from the *File* menu. Name the file *extended-topology.imn*.

### 5.1.3 Attaching to a running experiment

IMUNES gives you the possibility to run multiple independent experiments on one physical computer. Therefore, we added the possibility of attaching to running experiments through



the IMUNES GUI. In the *Experiment* menu, select the option *Attach to experiment*. A dialog similar to 5.9 is opened. Here you can select on which experiment you would like to attach. You can attach to all experiments, those that were started using batch mode and those that were executed from the GUI. The window shows the following experiment parameters:

- Experiment ID
- Filename of the topology
- Time when the experiment was started.
- Experiment screenshot (only if it was started through the GUI)

To attach to the you can double-click it's entry or use the *Resume selected experiment* button.

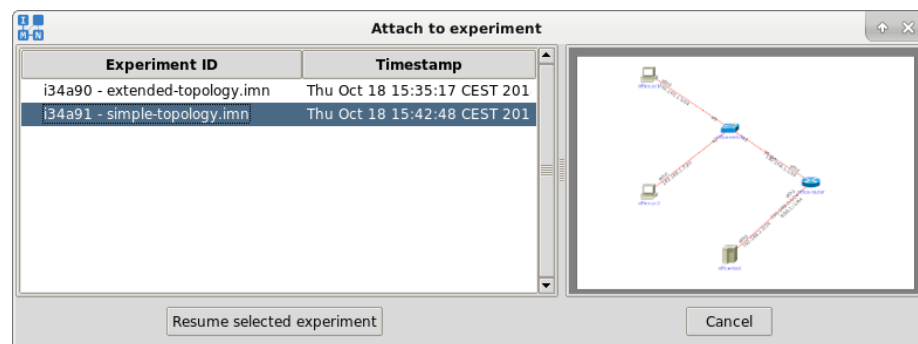


Figure 5.9: Attach to experiment window

## 5.2 Additional Configuration

### 5.2.1 Custom configuration

The configuration window of each network layer node (PC, Host, Router, NAT64 and Click router) also has the *Custom startup configuration* field. In order to view or edit the generated startup configuration click on the *Editor* button as shown in Figure 5.10.

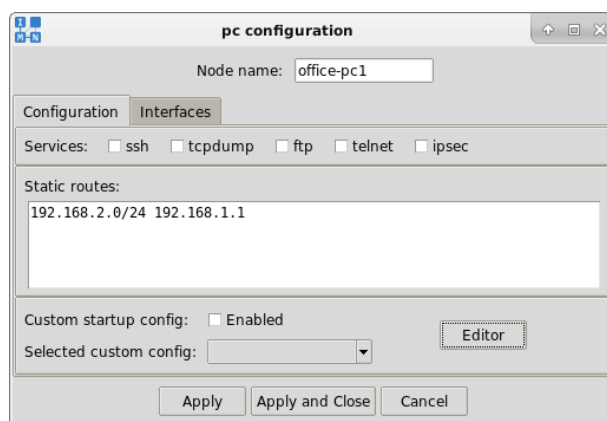


Figure 5.10: Configuration tab

In a newly opened window, write a new configuration name and click *Create*. An empty configuration file will be created. By clicking on *Fill defaults*, in case of a non-router network layer node or a router node with the static routing model, the default configuration consists of

`ifconfig` and `route` commands with the `/bin/sh` shell. Figure 5.11 shows the created default configuration.

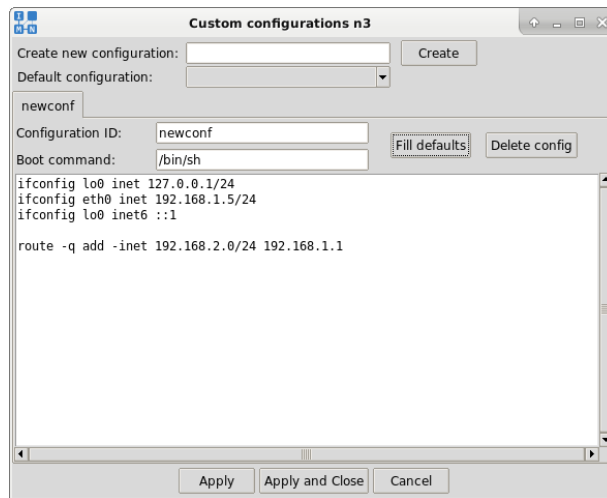


Figure 5.11: Custom configuration window

It is possible to add custom commands that will be executed at the boot time. When finished, click on the *Apply* or *Apply and Close* button to save the changes. When you have multiple configurations, you can choose the default one in the *Default configuration* drop-down menu. To delete a configuration, select its tab and click *Delete config* button.

**NOTE:** After starting the network simulation, the new/custom configuration will be considered only if *Custom startup configuration* is enabled. This is done by checking the *Enabled* check box in the *Custom startup config* field in Figure 5.10.

### 5.2.2 Physical and logical interfaces

When the Interfaces tab is opened, a configuration window similar to the one shown in Figure 5.12 is shown to the user. The Physical Interfaces list represents actual interfaces connected to links in IMUNES. Thus we can see that the selected node has one link in IMUNES.

IMUNES also offers the feature to configure additional logical interfaces. By default only one logical interface is added: `lo0` - the loopback interface with the following addresses: `127.0.0.1` for IPv4 and `::1` for IPv6.

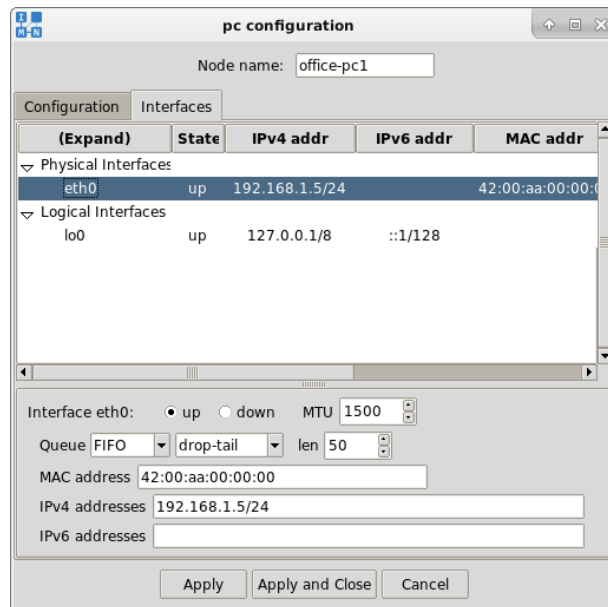


Figure 5.12: Physical and logical interfaces

Users can configure additional logical interfaces by clicking on the Logical Interfaces item in the Interfaces list. The Logical Interfaces configuration window is shown in Figure 5.13. At the moment IMUNES supports two types of logical interfaces: `lo` and `vlan`.

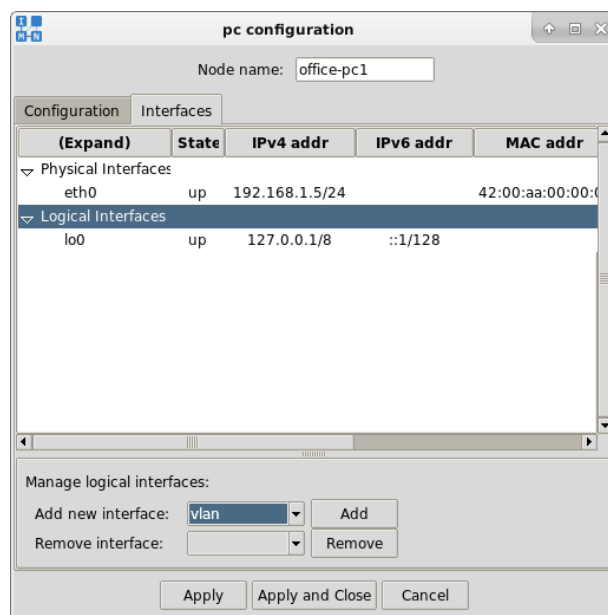


Figure 5.13: Logical interfaces management

Figure 5.14 shows an example for setting up a `vlan` logical interface `vlan0` on a physical interface `eth0` with an arbitrary tag 10 and an IPv4 address `10.0.0.1/24`.

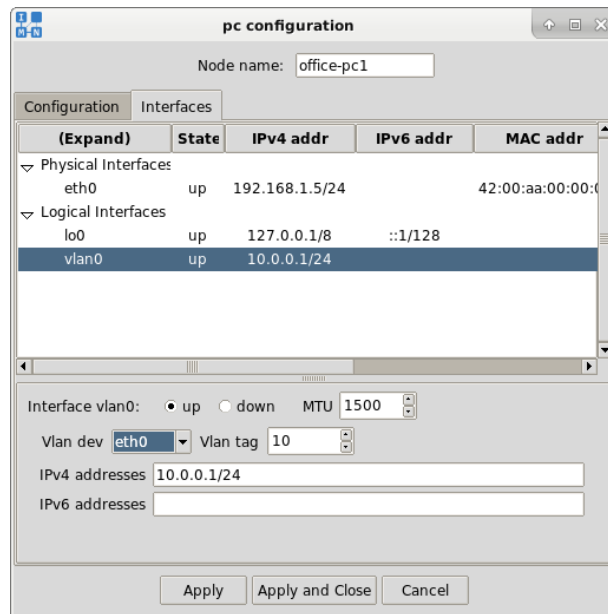


Figure 5.14: Logical interfaces management

## 5.3 Additional Tools

### 5.3.1 Splitting a link

Links can be split in two separate parts and it is possible to change location of both of its parts after selecting it with the *Select tool*. Right click on the link and then left click on the *Split label* in the pop-up menu to split the link in two halves. Separated link parts can be merged back with the *Merge* option from the link menu. This feature is shown in Figure 5.15.

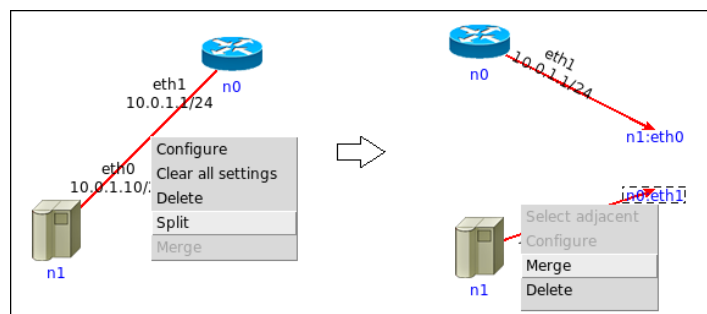


Figure 5.15: Split link

### 5.3.2 Generating a network topology

*TopoGen* menu from the menubar enables easier and faster generation of network topologies (Figure 5.16). This function can be used to generate following topologies: *Chain*, *Star*, *Cycle*, *Wheel*, *Cube*, *Clique*, *Bipartite* or *Random*.

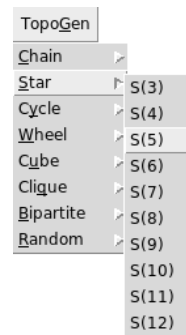


Figure 5.16: TopoGen menu

Some examples can be seen in Figure 5.17, Figure 5.18 and Figure 5.19. In order to generate a topology first select the network layer nodes (router, host or PC) from the toolbox and then the desired topology type e.g. bipartite graph  $K(2,3)$  (see Figure 5.18).

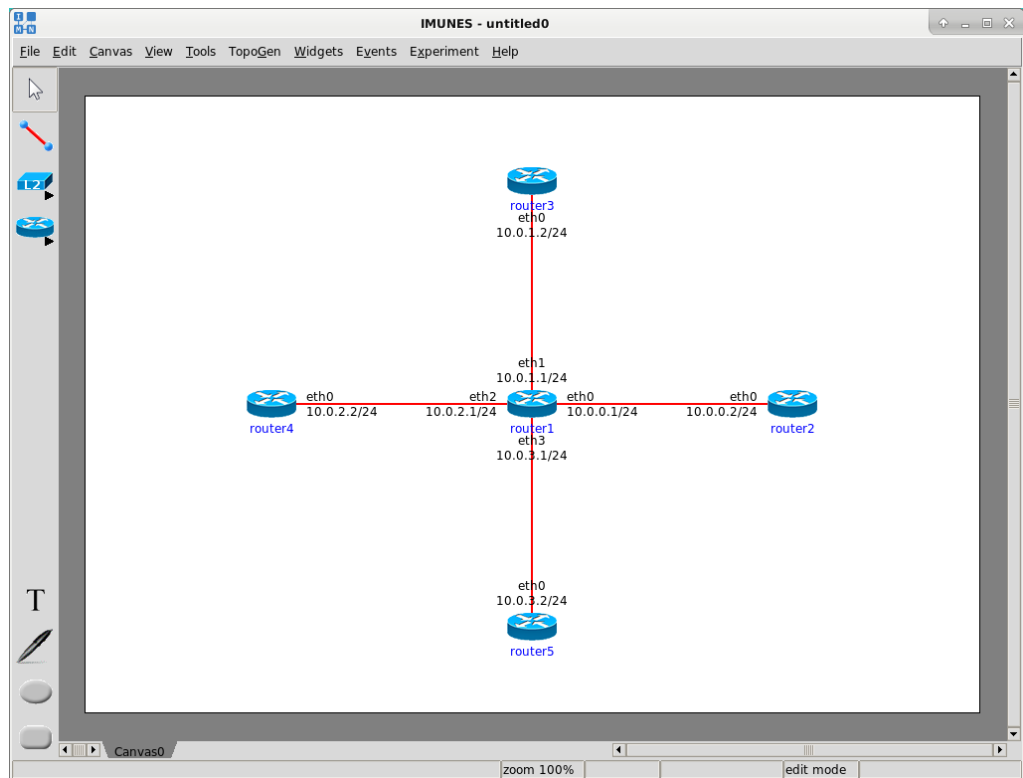


Figure 5.17: Star topology

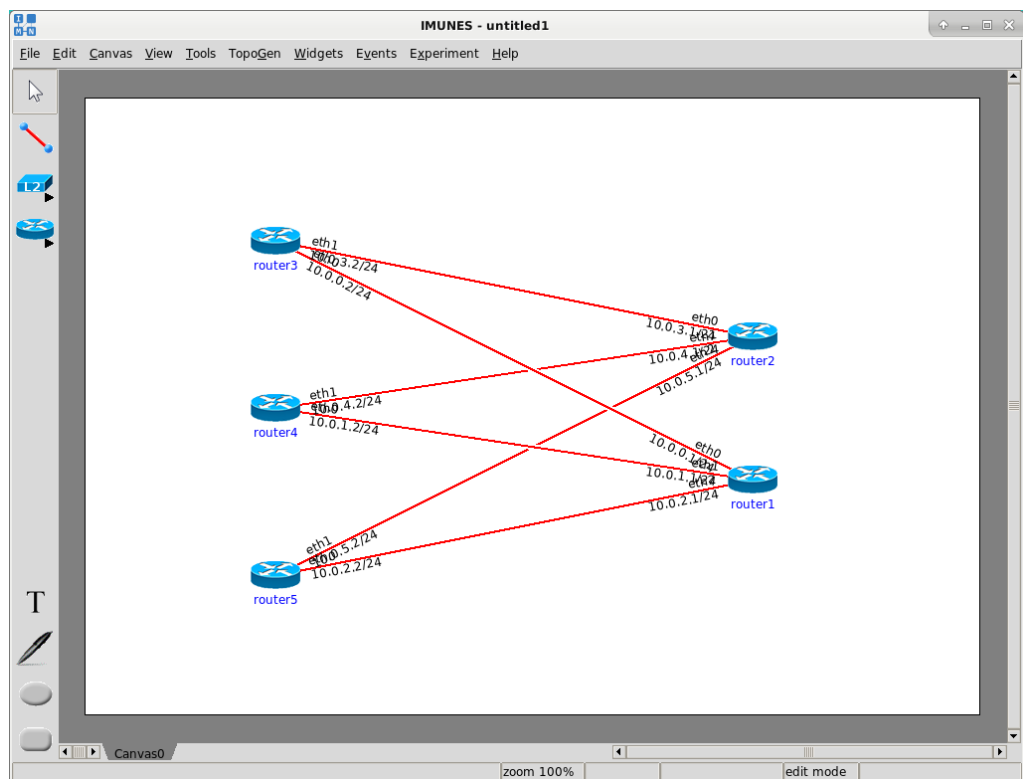


Figure 5.18: Bipartite topology

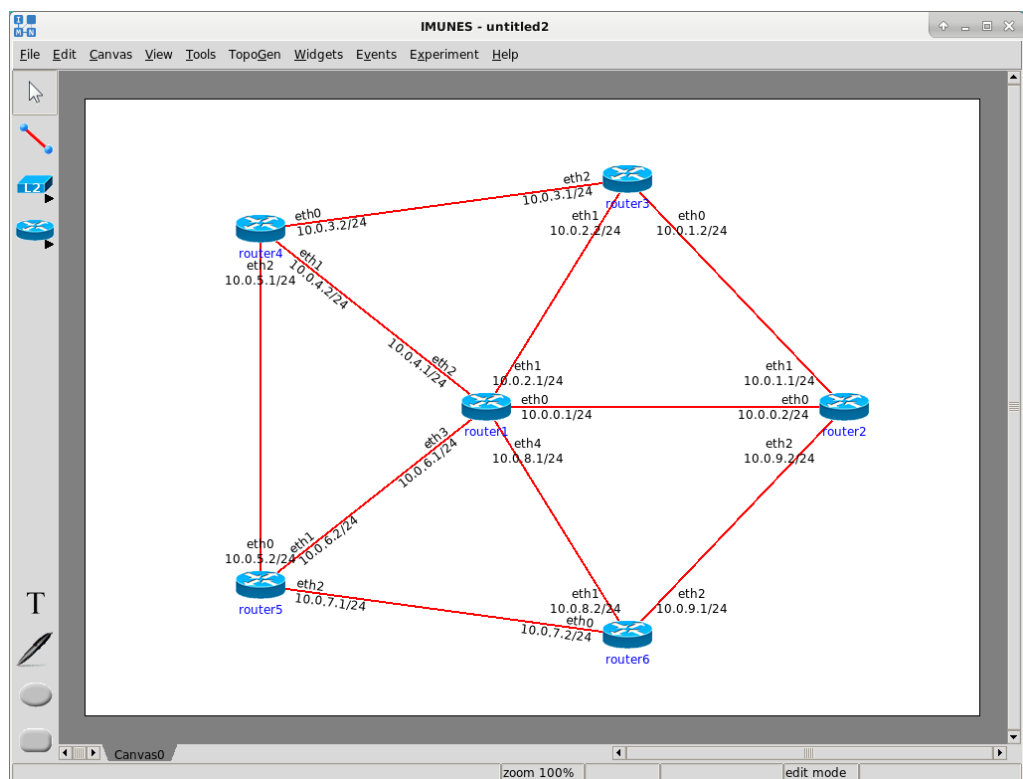


Figure 5.19: Wheel topology

In case of *random* topology an additional information is needed, so beside the number of

the nodes it is also necessary to specify the number of links. The nodes in the random topology will be randomly connected with the number of links specified before. An example of generating a *random* topology:

1. Select the router tool from the toolbox.
2. Choose the random topology: *TopoGen* → *Random*
3. Choose the desired number of nodes and links e.g.  $n = 6$ ;  $m = 5$ , where  $n$  is the number of nodes and  $m$  the number of links in the generated network topology (*Random* →  $R(6,m) \rightarrow R(6,5)$ ).

The result is shown in Figure 5.20.

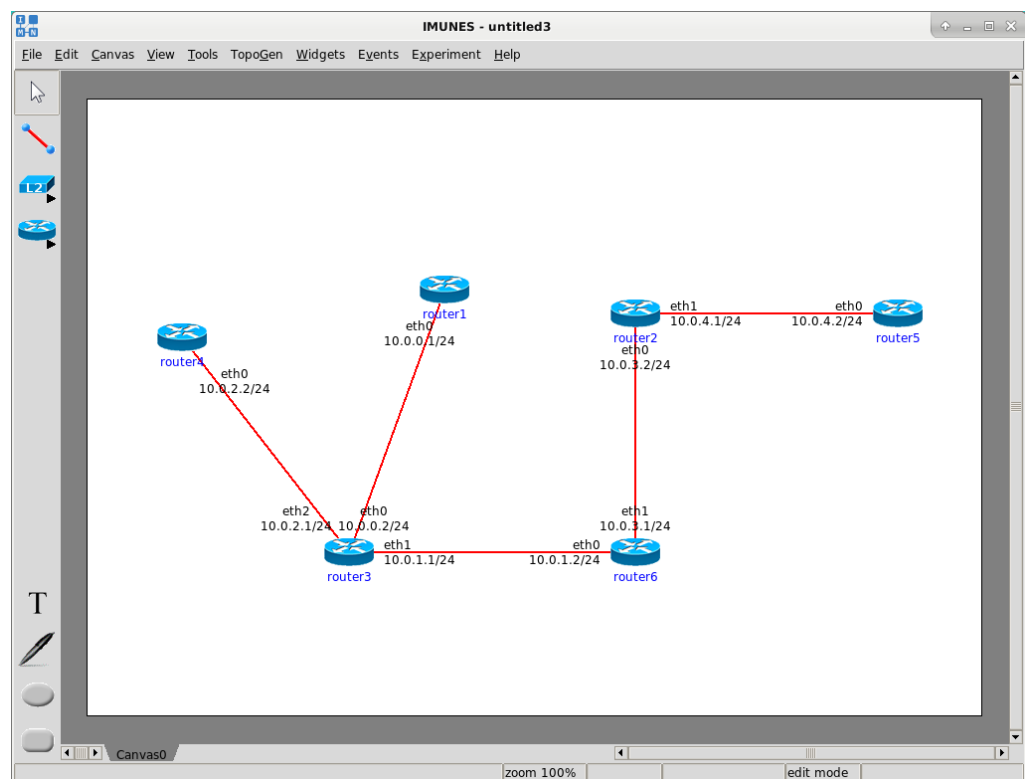


Figure 5.20: Random topology

Using the *TopoGen* tool you can generate topologies containing one type of node (router, host or PC). In that case, new nodes of the same type are created and placed on the canvas. Another option is to add new nodes to canvas and then connect them using the topology generator:

1. Add nodes to the canvas (don't have to be same type).
2. Select nodes that should be included in the new topology
3. Right click on one of the selected nodes and choose the option *Create Link to* from the menu. Choose the option *Selected* and select one of the offered topologies (*Chain*, *Star*, *Cycle*, *Clique* or *Random*). An example is shown in Figure 5.21.

In addition to that, it is also possible to transform existing nodes:

1. Select nodes that should be transformed
2. Right click on one of the selected nodes and choose the option *Transform to* from the menu. Select one of the offered options (*Router*, *PC* or *Host*). An example is shown in Figure 5.22.

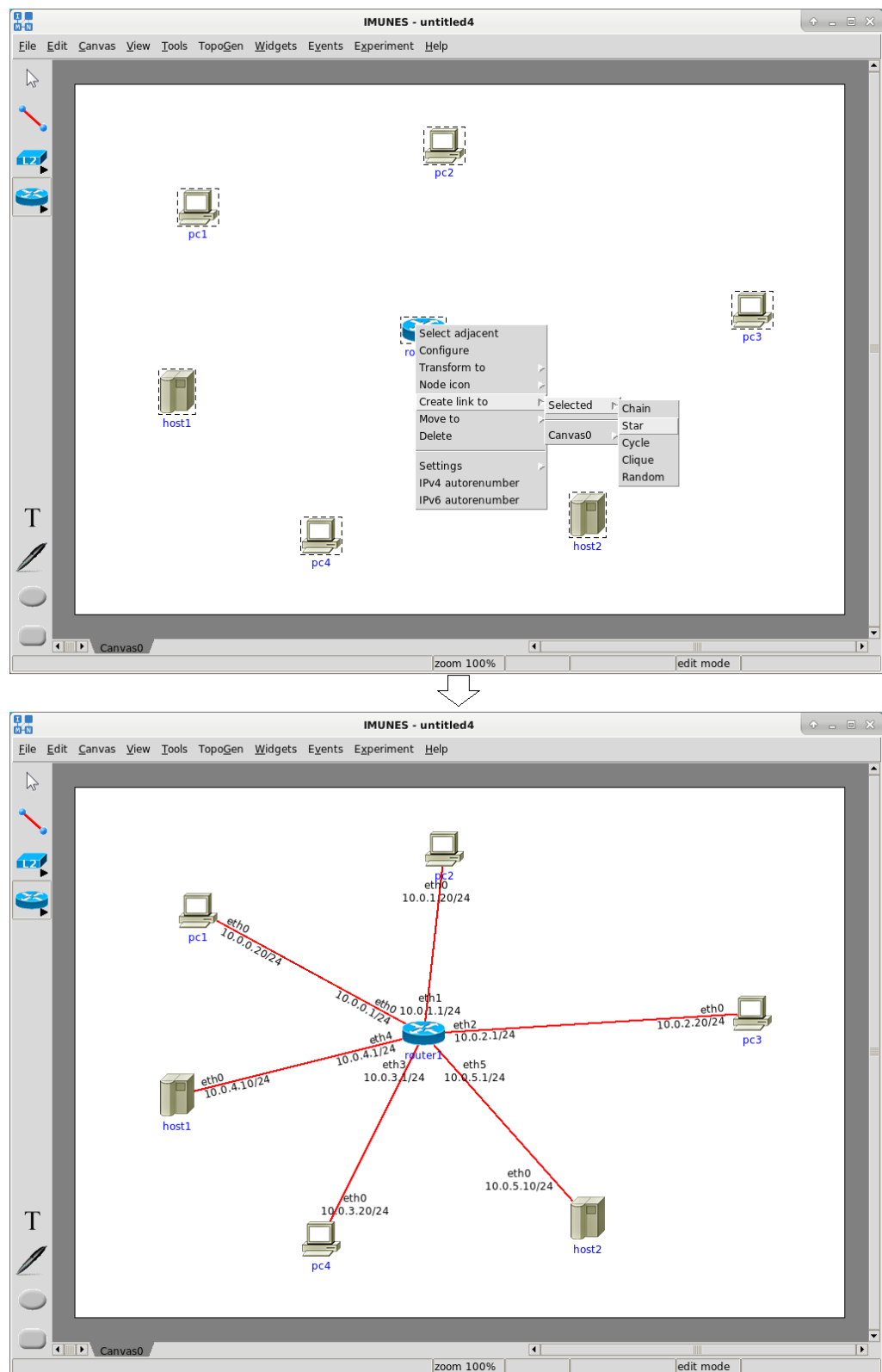


Figure 5.21: Example of creating a network topology with existing nodes



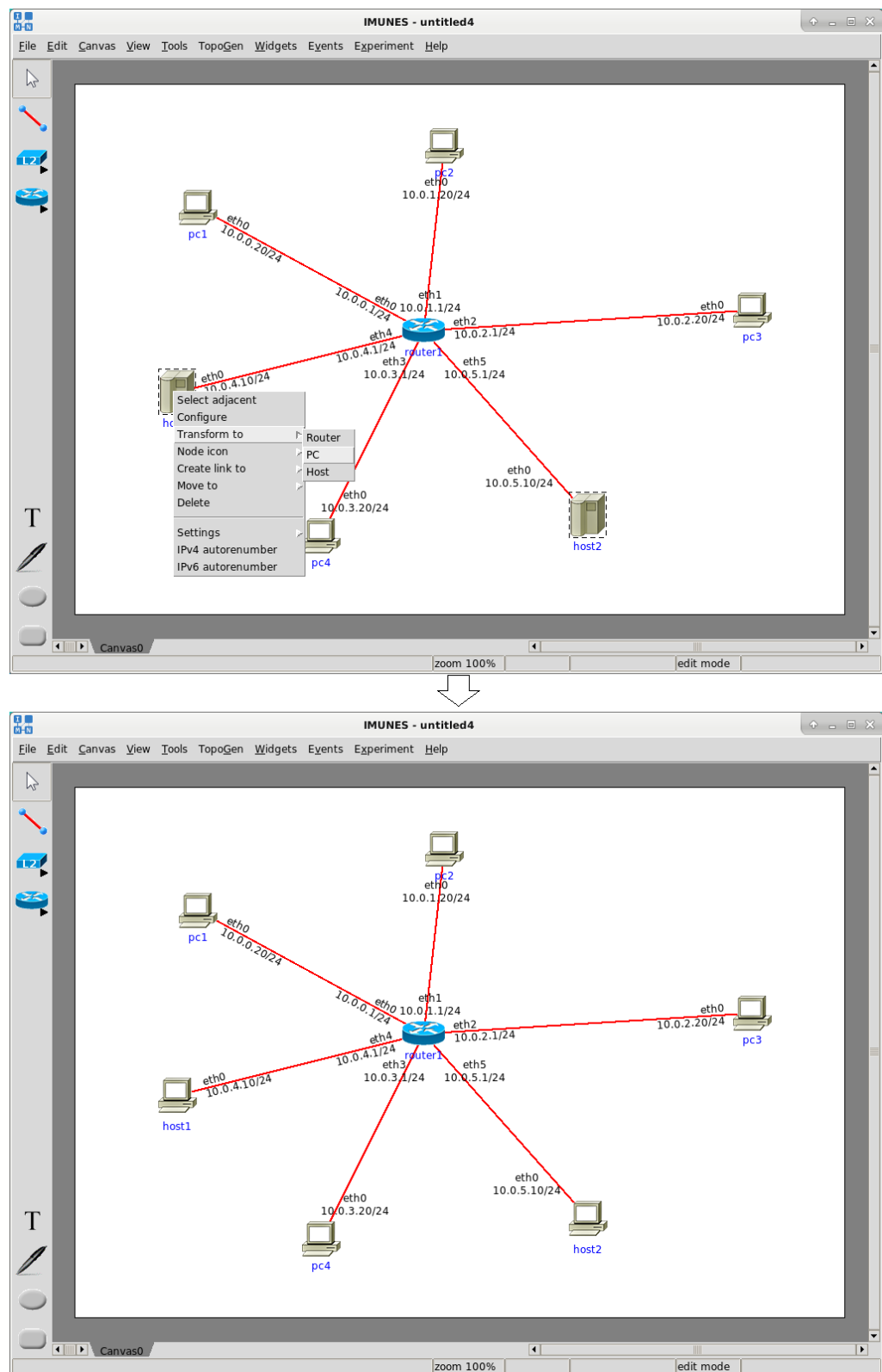


Figure 5.22: Example of transforming existing network layer nodes

### 5.3.3 IPv4 address pool

*IPv4 address pool* option from the *Tools* menu is used for replacing default 10.0.0.0/24 address pool. Choosing that option invokes a dialog shown in Figure 5.23.

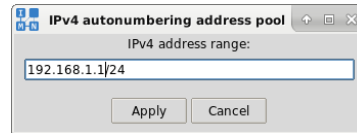


Figure 5.23: IPv4 address pool dialog

In order to replace default 10.0.0.0/24 address pool, set variable-mask IPv4 address pool through the invoked dialog. CIDR notation is required, so the IPv4 address needs to be followed by a slash and a network length. To apply changes click on the *Apply* button. The given address pool will be applied to all the subsequently created network layer elements (Figure 5.24).

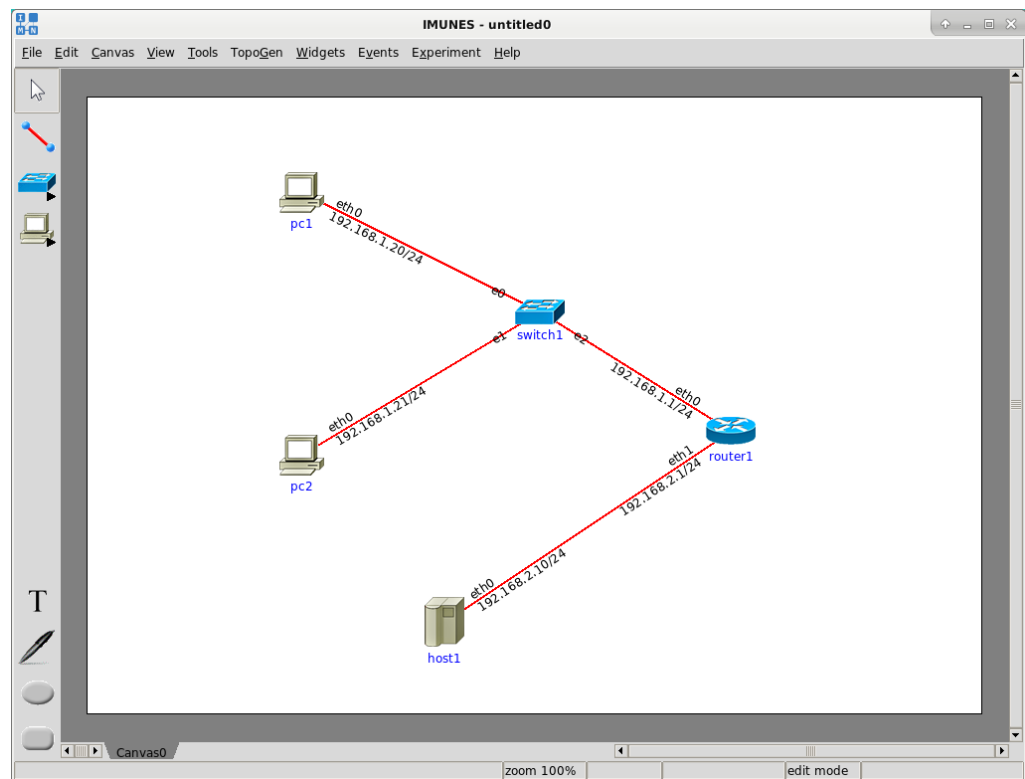


Figure 5.24: IPv4 address pool example

In order to apply the given address pool to selected elements, right click on the network layer element and choose the option *IPv4 autorenumber* from the popped up menu. In example shown in Figure 5.25 we have set IPv4 address pool to 160.153.1.1/24, selected all network elements and selected the option *IPv4 autorenumber* from the node menu to apply the given address pool to selected elements.

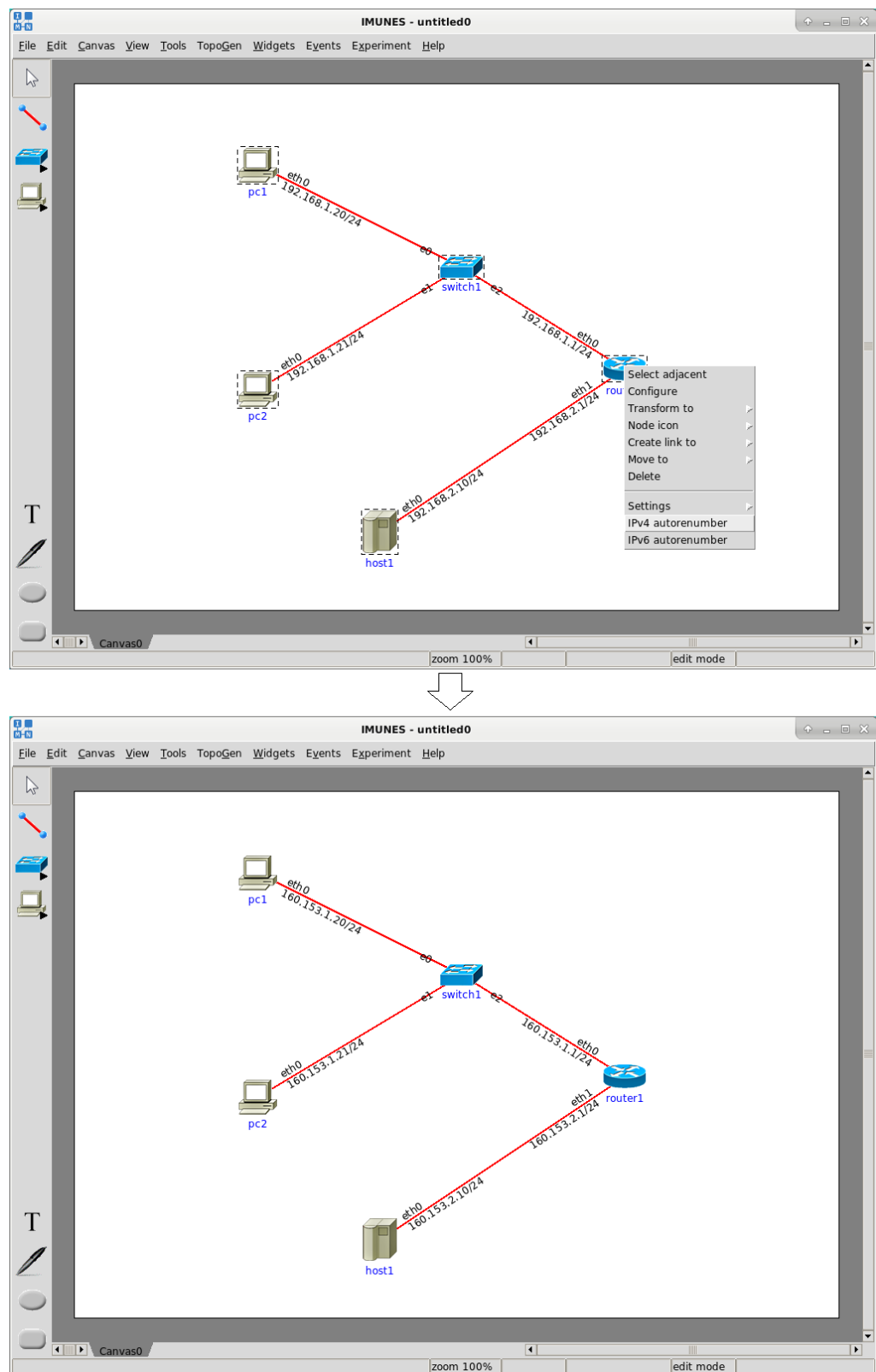


Figure 5.25: IPv4 autorenumber example

### 5.3.4 IPv6 address pool

*IPv6 address pool* option from the *Tools* menu is used for replacing default fc00::/64 address pool. The procedure for setting the IPv6 address pool is the same as for setting the IPv4 address pool.

### 5.3.5 Routing protocol defaults

In the *Tools* menu you can find the *Routing protocol defaults* option. Selecting this option invokes a popup window shown in Figure 5.26. Just as a reminder, settings referring to routing protocols can be also changed in the *router configuration* window. The difference is that changes made there are applied only to the router that is being configured. This tool in contrary makes it possible to apply changes to one or more selected routers. If there is no router selected than the changes will be applied to subsequently created routers. Note that this option will be disabled when the experiment starts.

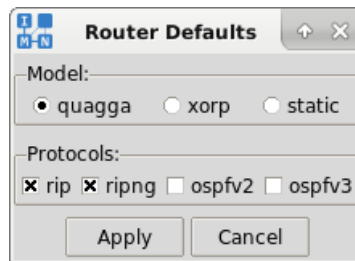


Figure 5.26: Router defaults window

## 5.4 Customizing Look

### 5.4.1 Annotations

To emphasize some parts of the network topology, you can add various graphic elements to the canvas. These elements are divided into three groups:

- Text
- Freeform
- Oval
- Rectangle

Each of these elements have their own tools in the toolbox: *Text* tool (Figure 5.27), *Freeform* tool (Figure 5.28), *Oval* tool (Figure 5.29) and *Rectangle* tool (Figure 5.30).



Figure 5.27: Text tool



Figure 5.28: Freeform tool



Figure 5.29: Oval tool



Figure 5.30: Rectangle tool

To add an annotation to the canvas, select the appropriate tool and click (and drag) where you want to add the annotation. A popup window will be shown. There you can define how will the annotation look.

When created, annotations can be moved around on the canvas. This is done by using the select tool. Click on the annotation and then drag it to its destination.

## Text

The text annotation lets you define the following options(Figure 5.31):

- *Text color* - color of the text in RGB values
- *Font* - which system font, size and style you want to use

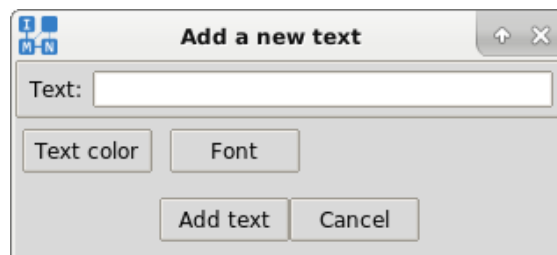


Figure 5.31: Text configuration window

## Freeform

The freeform configuration window includes:

- *Line color* - which line color to use
- *Width* - line width to use

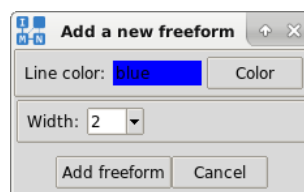


Figure 5.32: Freeform configuration window

## Oval

The size of the oval annotation is defined by dragging the cursor on the canvas while keeping the left mouse button pressed. When the annotation size seems to be fine, release the mouse button. The oval configuration window will popup (Figure 5.33).

The oval annotation lets you define the following additional options (Figure 5.33):

- *Fill color* - color of the annotation fill in RGB values.
- *Border color* - color of the annotation border in RGB values.
- *Border width* - width of the annotation border.

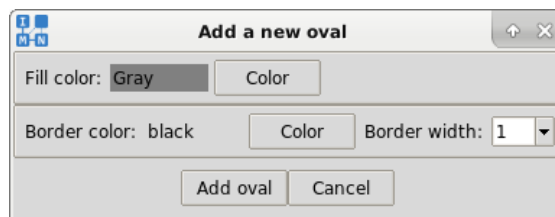


Figure 5.33: Oval configuration window

## Rectangle

The rectangle configuration window has the same options as the oval configuration window. The rectangle size is defined the same way as the size of the oval.

The rectangle annotation lets you define the following additional options (Figure 5.34):

- *Radius of the bend at the corners* - defines roundness of the rectangle edges.

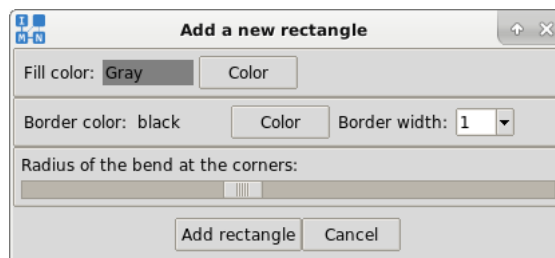


Figure 5.34: Rectangle configuration window

Example of the annotations usage is shown in Figure 5.35.

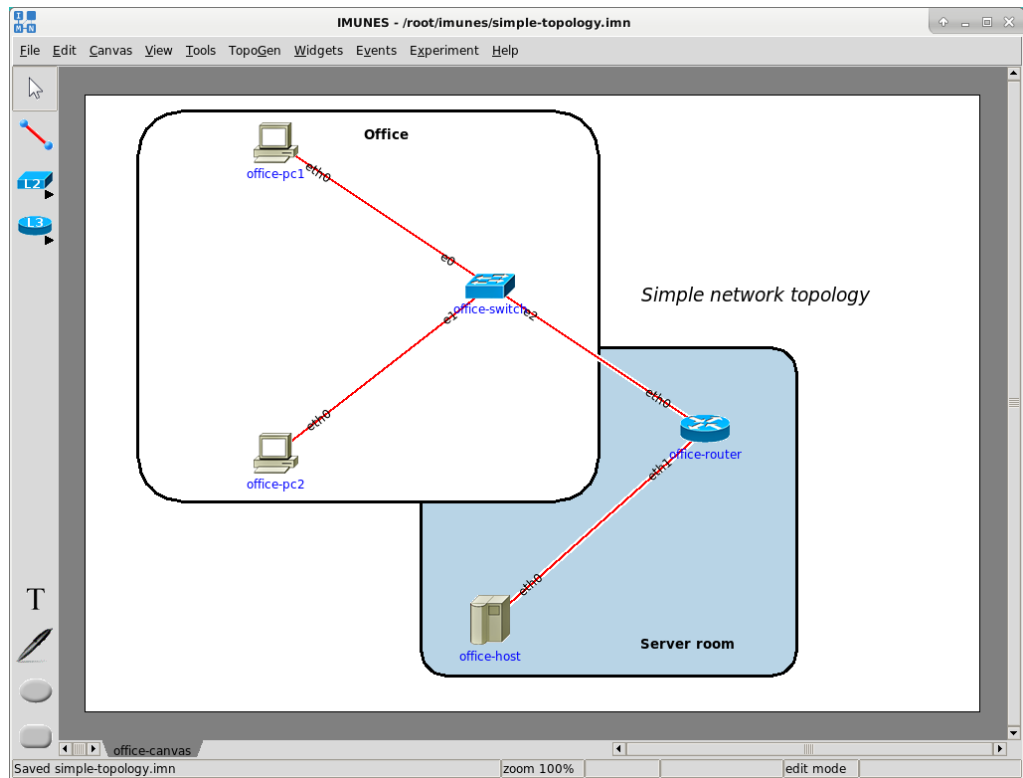


Figure 5.35: Annotations example

### 5.4.2 Canvas background image

The options for changing the canvas background image are accessible through the *Canvas* and *View* menu and through the menu that is opened with a right click on the empty canvas (Figure 3.6 and Figure 5.36).

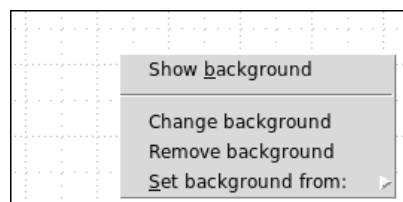


Figure 5.36: Background image menu

All options related to the canvas background are accessible in the *Background image* menu (Figure 5.36):

- *Show background* - Show or hide the canvas background.
- *Change background* - Opens the *Change canvas background* window (Figure 5.37).
- *Remove background* - Removes the background from the current canvas.
- *Set background from* - Sets the background from an another canvas.

To set a canvas background you need to open the *Change canvas background* window (Figure 5.37).

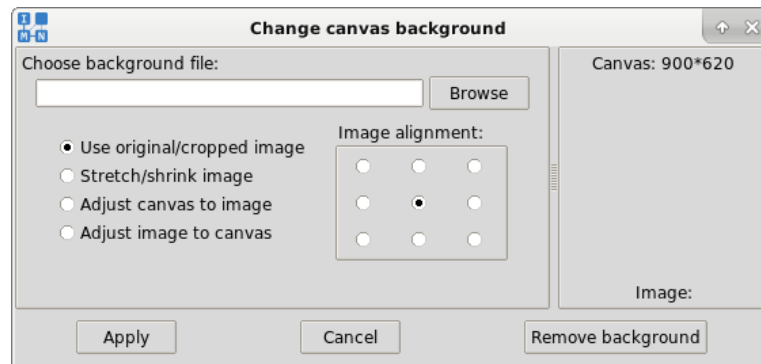


Figure 5.37: Change canvas background

This window is divided into two main parts:

- Left pane with the canvas background options
  - Field and button for choosing the background image
  - Image setting options
  - Image alignment options
  - Additional information if imagemagick is not present (Figure 5.38)
- Right pane with canvas and image information and preview
  - Canvas size information
  - Image preview
  - Image size information

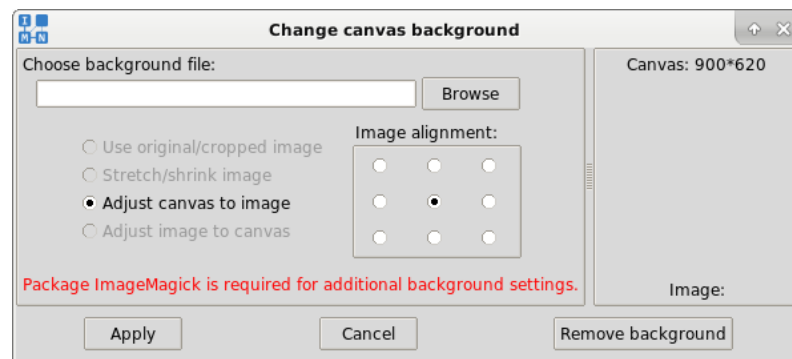


Figure 5.38: Warning if ImageMagick is not present

When the image is selected there are four image setting modes that can be chosen:

- *Use original/cropped image* - If the image is smaller it will be placed in the position defined by the image alignment. If the image is larger it will be cropped. The image alignment will define which part of the image will be taken as the background.
- *Stretch/shrink image* - If the image is smaller it will be stretched without changing the proportions. If the image is larger it will be shrunk without changing the proportions. The image alignment will define which part of the image will be taken as the background.
- *Adjust canvas to image* - The canvas will be resized to the image size and then the background image will be set.



- *Adjust image to canvas* - The image will be forcibly resized to the canvas size with changed proportions if needed.

## Settings canvas background images

We will use the extended topology example (*extended-topology.imn*) to set canvas background images. On the *office-canvas* we will set a background image. Then we will go to the *roadwarrior-canvas* and set that same image using the *Set background from* → *office-canvas* option (Figure 5.39).

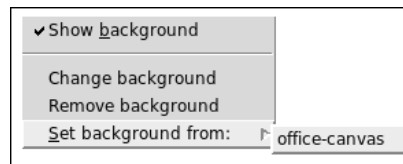


Figure 5.39: Set background from menu

The final result is shown in Figure 5.40 and Figure 5.41.

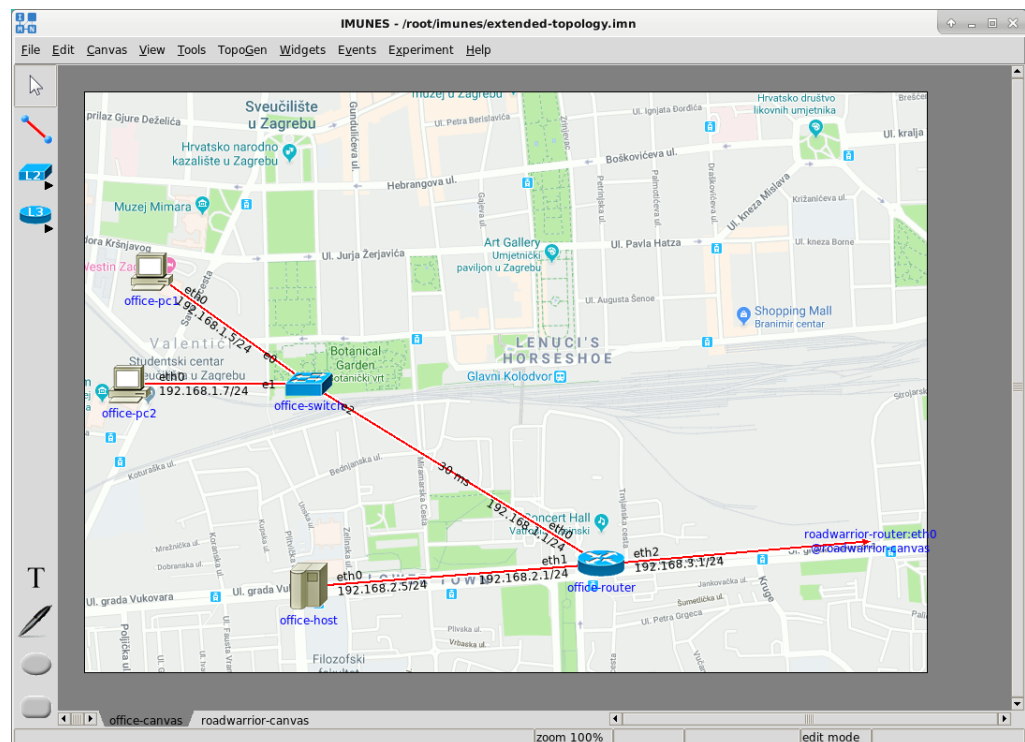


Figure 5.40: Canvas background example on office-canvas

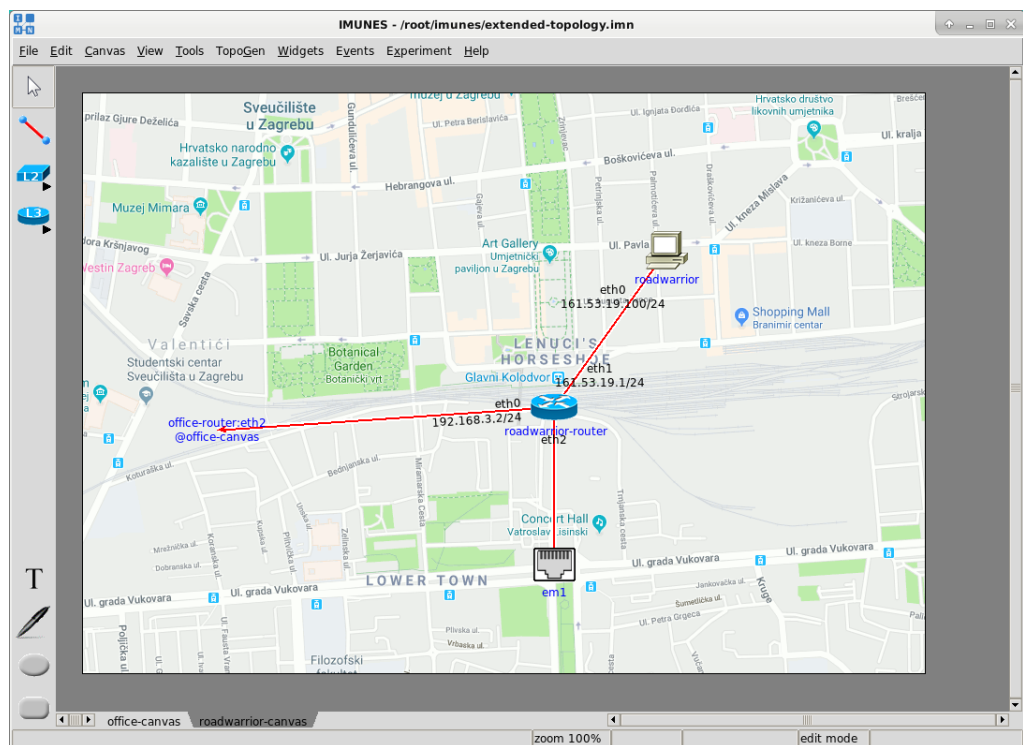


Figure 5.41: Canvas background example on roadwarrior-canvas

### 5.4.3 Icons

IMUNES lets you choose custom node icons. First, select the nodes whose icons you want to change. Then right click on the selection and then go to the *Node icon* → *Change node icons* option (Figure 5.42). This menu has also the *Set default icons* option that sets the default node icon for the selected icons.

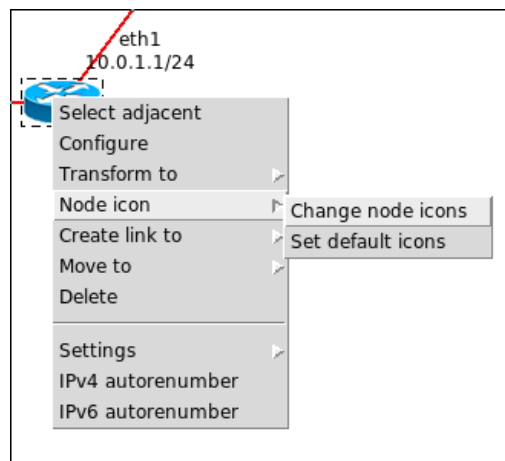


Figure 5.42: Node icon menu

The *Change node icons* option opens the *Set custom icon* window (Figure 5.43).

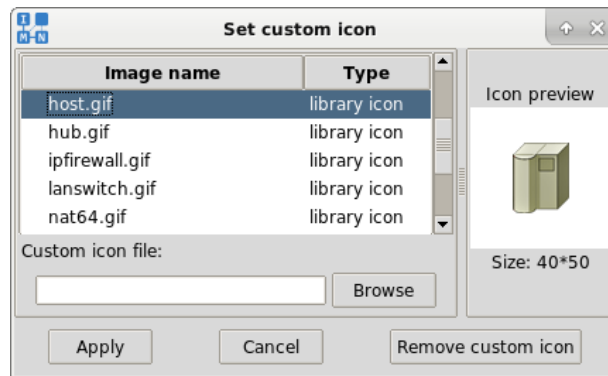


Figure 5.43: Set custom icon

This window is divided into two main parts (Figure 5.43):

- Left pane for choosing custom icons
  - List of library and custom icons on the top. Every icon that has been used or is being used in the project will be available at the end of the list, and will be given a generic name (e.g. img0, img5).
  - Field and button for choosing the custom icon
- Right pane with the icon preview and icon size information

We will now open the *extended-topology.imn* file and set a custom icon. We will choose the library icon *ipfirewall.gif* for the *roadwarrior-router*. The final result is shown on Figure 5.44.

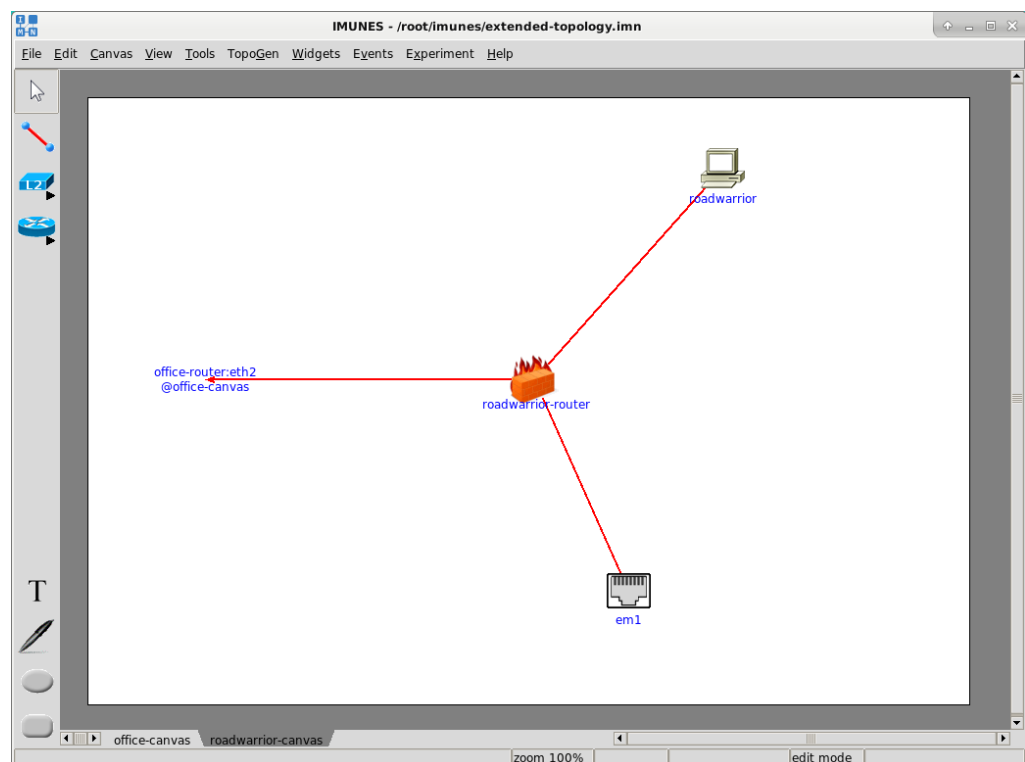


Figure 5.44: Changed roadwarrior-router icon

## Icon size

If you want to emphasize the information about nodes, interfaces and links instead of node icons you can change the icon size through the *View* → *Icon size* option (Figure 5.45).

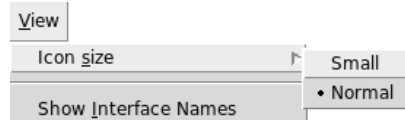


Figure 5.45: Icon size menu

Let's take the `simple-topology.imn` example and set the icon size to small. (Figure 5.46) Currently, only two sizes are available, normal and small. Otherwise, custom icons can be used.

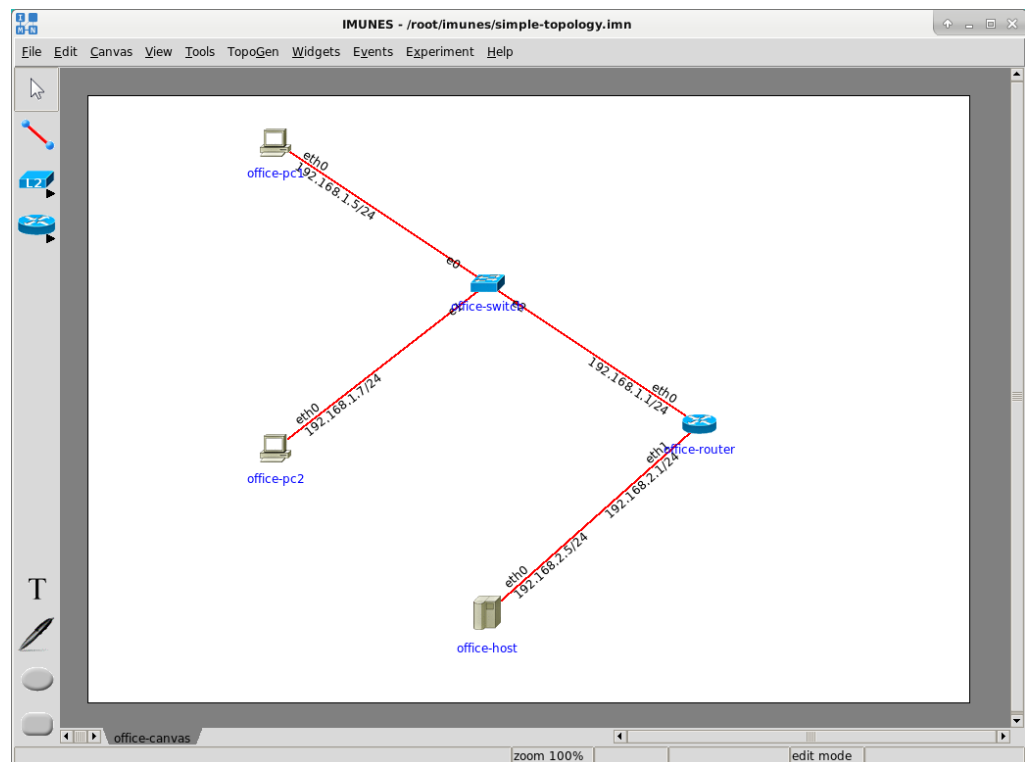


Figure 5.46: Icon size example

## 5.5 User-configurable Event Scheduling

This section describes a feature for scheduling of arbitrary deterministic and stochastic events during experiment execution.

### 5.5.1 Principle of operation

The control plane in IMUNES is extended to retain control over the experiment execution after initial topology instantiation, allowing for user-scheduled events to influence selected parameters during run time.

The current implementation allows for the event scheduler to assume control over selected link properties such as:

- bandwidth
- delay
- bit error rate
- packet duplication
- visual attributes
  - line width
  - line color

The event scheduler supports two general classes of schedulable events: one-time changes and periodic functions.

One-time events update the selected parameter at requested point in time, leaving that parameter constant throughout rest of the experiment execution, or until another event updates it to a new value at some later point in time.

Periodic functions allow for time-variant functions to be applied to selected parameters by specifying those functions as single events. Subsequently scheduled events acting upon the same parameter will cancel the current periodic function and replace it with either a constant value or another periodic function.

### 5.5.2 Configuring events with events editor

Events can be specified in the *Events editor* (Figure 5.47). To open the *Events editor* select the *Events* → *Events editor* option. The editor is divided into two elements. The left side element contains a list of all links. The right side element contains events configured on the selected link. This dialog allows you to edit events on that link. To save the event scheduling configuration you need to click on *Apply* button. The events editor gives you the possibility to start or stop the event scheduling during experiment execution. This can be also done from the *Events* menu in the menubar.

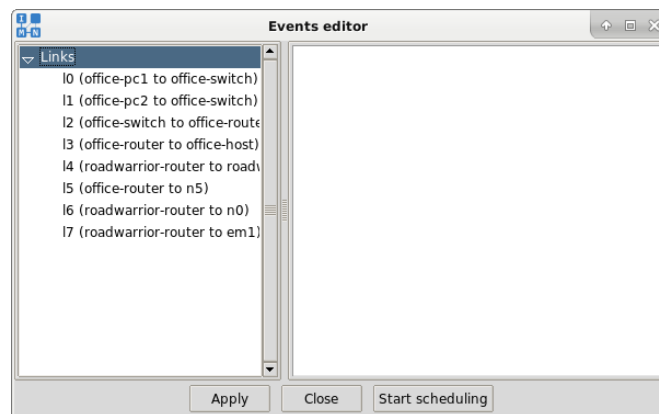


Figure 5.47: Events editor

Each event entry occupies a single line of text, consisting of four fields: deadline, target parameter, function and function parameters. The first field, deadline, is specified as an integer number of seconds since experiment instantiation. The second field, target parameter, may be one of the following: bandwidth, delay, ber, duplicate, width or color. The remainder of the line is further parsed as a function. The type of the function is determined by the leading keyword, which may be either const, ramp, rand or square. When saving the configuration through the events editor a syntax check will be performed. If the syntax is wrong the configuration will not be saved and a popup dialog will show the first line that has a syntax error.

## Const

The const function accepts one parameter, the target value. After the deadline time the parameter will constantly be equal to the target value.

## Ramp

Behavior of the ramp function is determined by three arguments: initial value, delta, and period, which are all integers. The first argument represents the initial function value. The second is the delta value, which may be both positive or negative, which is added to the previous value of the function at each period. The third argument is the period, determining how often will the delta value be added to the current function value.

## Rand

The rand function is determined by three arguments: lower bound, upper bound, and period; all of which are integers. The function will assume a random value between lower and upper bound after each period (in seconds) expires.

## Square

Finally, the square function has three arguments as well: low value, high value, and period; all integer numbers. The resulting function will flip from low value to high and vice versa after each period (in second) expires.

## Example

The following example illustrates a possible event scheduling scenario that can be entered in the Event editor window (Figure 5.47):

```
30 bandwidth ramp 128000 8000 2
30 delay rand 80000 120000 8
60 delay square 100000 200000 10
60 ber const 1
90 ber const 0
120 delay const 0
```

At  $t = 30$  s, bandwidth of the selected link will be set to 128 Kbps, and will continue to grow at a rate of 8 Kbps each 2 s.

Also at  $t = 30$  s, the delay will begin assuming a random value between 80 ms and 120 ms, and will continue to change the setting to new random values in the same range each 8 s.

At  $t = 60$  s, the delay will cease to assume random values, and instead it will begin to oscillate between two discrete values, 100 ms and 200ms, each 10 s.

Also at  $t = 60$  s, the bit error rate (BER) will be set to 1, resulting in all frames traversing the link to be silently dropped.

At  $t = 90$  s, the BER is reset back to 0, allowing for all frames to traverse the link without artificial losses.

At  $t = 120$  s, the oscillation of the delay parameter will stop, and delay will be reset to 0 ms for the rest of the experiment execution time.

The current time is shown on the status bar after the zoom value. (Figure 4.17)

### 5.5.3 Configuring events through configuration file

**NOTE:** It is advised to use the Events editor to configure event scheduling because it includes a syntax check. In the case of configuring events through configuration file, the wrong syntax will be silently ignored.

IMUNES experiments are defined via plain-text configuration files, which currently describe virtual nodes and links, as well as additional objects related only to GUI visual properties and annotations. The other way to configure events is manual editing of an existing configuration file.

The example below shows a configuration section describing a virtual link in an IMUNES experiment. The link l0 connects virtual nodes n0 and n1, has the bandwidth constraint set to 128 Kbps, and has the thickness of the line representing the link in the GUI set to 6 pixels. All of the mentioned properties are directly controllable via the IMUNES GUI. The configuration for link l0 also includes an empty placeholder for schedulable events, meaning that no events have been programmed for this link.

```
link l0 {
    width 6
    nodes {n0 n1}
    bandwidth 128000
    events {
    }
}
```

The events section in the configuration file accepts the same commands as does the Events editor in the GUI.

## 5.6 Starting and terminating a simulation through CLI

In addition to the *File*→*Open* option to open an .imn file and the *Experiment*→*Execute* option used to initiate the virtual network topology in the GUI, the simulation can be initiated through the command-line interface (CLI) with the following command:

```
# imunes -b simple-network.imn
```

Using IMUNES through GUI, the *Experiment*→*Terminate* option is used for shutting down the simulation and cleaning up the virtual network topology from the kernel. The CLI alternative for the latter is the following command:

```
# imunes -b -e experimentId
```

The parameter `experimentId` represents the experiment identifier. In order to get the experiment identifier you can use the `himage` command. With `himage -l` you will get a list of identifiers of all started experiments.

## 5.7 Managing virtual nodes (jails) - jls, jexec

The FreeBSD jail mechanism allows partitioning of a FreeBSD-based computer system into several independent smaller systems called jails. This mechanism enables creation of a safe environment, separate from the rest of the system. Processes created inside a jail are limited within that jail environment. Each jail is a virtual environment running on the host machine, having its own file system, processes, set of users, networking subsystem of the FreeBSD kernel and a few other things.

Two main commands exist in FreeBSD for managing and configuring previously created jails:

- `jexec` - executes a command inside an existing jail
- `jls` - lists jails.

A virtual image or `vimage` is a jail with its own independent network stack instance. Every process, socket and network interface present in the system is always attached to one, and only one, virtual network stack instance (`vnet`). During system bootup sequence a default `vnet` is created to which all the configured interfaces and user processes are initially attached.

The `jexec` command allows for execution of arbitrary processes in a targeted virtual image.

```
jexec jname command ...
```

The `jexec` command starts the selected command and its arguments in the jail `jname`.

To find out the names of started jails the `jls` command is used:

```
jls [-hnqsv] [parameter ...]
```

Since the default `jls` command doesn't list names of jails a better output is provided using the command:

```
jls -h jid name host.hostname
```

Also, the command `jls -v` gives a more detailed output.

### 5.7.1 Examples

Execute the `ifconfig` command in the jail with the jail name `n1`:

```
# jexec n1 ifconfig
```

Execute the `ssh` command in the IMUNES virtual node named `host1`:

First we need to find out the jail ID or jail name to execute the wanted command:

```
# jls -h jid name host.hostname | grep host1
```

The first parameter output is the jail ID, the second is the jail name and the last is the hostname. To execute the command you need the jail ID or jail name:

```
# jexec jid ssh
```

```
# jexec jname ssh
```

## 5.8 Himage tool

The `jls` and `jexec` commands can be impractical for creating scripts for topologies since the output of the `jls` command is needed for starting the `jexec` command. The `jexec` command can't take the hostname of the jail as an argument, it can only take the jail name or jail id. The jail name is created from the IMUNES experiment ID and the node identifier (not hostname). Every experiment started in IMUNES has a different randomly generated experiment ID to enable execution of multiple experiments at once.

IMUNES comes with the `himage` tool that enables the usage of hostnames when starting commands in virtual nodes. The `himage` tool starts the `jexec` command with the appropriate jail name so that the user doesn't have to search for it.

The `himage` command has the following options:

- `himage vi_hostname command ...` - executes the command in the virtual node with the specified hostname. If no command is specified, it starts an interactive shell.
- `himage -m vi_hostname` - executes the command in the experiment master jail. If no command is specified, it starts an interactive shell.
- `himage -v vi_hostname` - gets the jail node name of the virtual node with the specified hostname
- `himage -n vi_hostname` - gets the node identifier for the specified hostname.
- `himage -e vi_hostname` - gets the experiment ID in which the virtual node with the specified hostname is running.
- `himage -j/-i vi_hostname` - gets the jail ID in which the virtual node with the specified hostname is running.
- `himage -d` - gets the hostname jail filesystem path on the host machine.
- `himage -l` - gets the experiment list with the experiment data.
- `himage -ln` - gets the experiment list with node names.



### 5.8.1 Examples

Example of usage of the command `himage` on a node with the hostname "pc" to get a list of running processes:

```
# himage pc ps ax
```

If there are multiple experiments running and there are nodes with the same hostnames in these experiments the `himage` command accepts the following node specification where `vi_hostname` is specified as `hostname@eid`, where `eid` is the experiments' ID.

```
# himage hostname@eid command ...
```

i.e:

```
# himage pc@i3d05a ps ax
```

where `i3d05a` is the experiment ID of the running experiments. To find out which experiments are running the `himage -l` command can be used as well as `jls -h jid name host.hostname`.

Execute the `ifconfig` command the IMUNES node named `server`:

```
# himage server ifconfig
```

## 5.9 Hcp tool

While the `himage` command is used for running programs inside virtual nodes the `hcp` is used to copy files directly to the filesystem of running mobile nodes, thus simplifying deployment of configuration files for starting various services on virtual nodes.

Usage of the command `hcp`:

```
hcp [cp_command_options] [vi_hostname1:]filename [vi_hostname2:]filename
```

The `hcp` command invokes the `cp` command with the specified options. If the `vi_hostname1` is specified the script copies a file from the virtual node, otherwise it copies a file from the local folders. The second `vi_hostname2` specifies on which node the first file will be copied, if not specified the file is copied to the local folders.

`vi_hostname` is specified in the same way as in the `himage` command, `hostname` or `hostname@eid`.

### 5.9.1 Examples

Copy file `dhcpd.conf` from a local folder to the virtual node DHCP:

```
# hcp dhcpd.conf DHCP:/usr/local/etc/
```

Copy file `message.txt` from the virtual node PC to a local folder:

```
# hcp PC:/root/message.txt .
```

Copy file `index.html` from the virtual node HOST to the virtual node HTTP:

```
# hcp HOST:/usr/local/www/data/index.html HTTP:/usr/local/www/data/
```

## 5.10 Example (himage and hcp)

This is an example of starting an DHCP server through a script with the provided configuration file `dhcpd.conf`.

Kill the server if it's started:

```
# himage DHCP killall -9 dhcpd
```

Copy the configuration file:

```
# hcp dhcpd.conf DHCP:/usr/local/etc/
```

Create the `.leases` file defined in `dhcpd.conf`:

```
# himage DHCP touch /var/db/dhcpd.leases
```

Start the sever with the copied configuration file:

```
# himage DHCP dhcpd -cf /usr/local/etc/dhcpd.conf
```

## 5.11 Vlink tool

The `vlink` tool is used to change link parameters. The following link parameters are available:

- bandwidth (bps, bits-per-second)
- bit-error rate, BER (number of bits in which one error will occur)
- delay (microseconds)
- packet duplication (% , percentage of packets that will be duplicated)

A link is identified by the endpoint node names. The link between `pc1` and `pc2` is identified by `pc1:pc2` or `pc2:pc1`.

Usage of the command `vlink`:

```
vlink [options] link_name[@eid]
```

### 5.11.1 Examples

Setting the bandwidth to 10 Mb/s with a delay of 30 ms to the link connecting *router1* and *pc1*:

```
# vlink -bw 10000000 -dly 30000 router1:pc1
```

Generate an error on one bit in a million:

```
# vlink -BER 1000000 router1:pc1
```

Set packet duplication to 20

```
# vlink -dup 20 router1:pc1
```

Modifying a link in a specific experiment (i.e. Experiment ID = `i56ad1`):

```
# vlink -dup 20 router1:pc1@i56ad1
```

To reset the link settings to the default values the `-r` flag is used:

```
# vlink -r router1:pc1@i56ad1
```

Check all `vlink` arguments with:

```
# vlink -h
```

---

## Appendix A Installation

---

### A.1 Installation of IMUNES on FreeBSD 8

#### A.1.1 Installing FreeBSD

A comprehensive and explanatory guide for installing configuring and using FreeBSD can be found here:

<http://www.freebsd.org/doc/handbook/>

Section 2 of the handbook describes the installation of the FreeBSD operating system:

<http://www.freebsd.org/doc/handbook/install.html>

You can choose to install FreeBSD with two different architectures:

- i386 - 32-bit - works on most personal computers.
- amd64 - 64-bit - works on newer computers that support 64-bit processing. Adds support for more RAM.

#### A.1.2 Step by step guide through the FreeBSD installation

1. Insert the FreeBSD-8.2-RELEASE or FreeBSD-8.3-RELEASE (i386 or amd64) medium on startup. Boot from it.
2. Country Selection - Choose your country (e.g. United States). Press OK.
3. Main Menu - Choose the "Standard" installation. Press Select.
4. Message - Press OK.
5. FDISK Partition Editor - Press C to create a slice:
  - (a) A minimum of 20GB is needed for FreeBSD. Type 20G to create a 20GB slice. Press OK.
  - (b) A screen with the number 165 appears. Press OK.
  - (c) Position on the created partition (probably named ad0s1).
  - (d) Press S to set the created partition bootable.
6. FDISK Partition Editor - Position on the "unused" space beneath the created partition. Press C to create another slice:
  - (a) A minimum of 5GB is needed for ZFS. Type 5G to create a 5GB slice. Press OK.
  - (b) A screen with the number 165 appears. Press OK.
  - (c) Remember the name of this partition (probably named ad0s2). This name will be later used for configuring ZFS.
7. FDISK Partition Editor - Press Q to finish.
8. Boot Manager - Select the "Standard" option. Press OK.
9. Message - Press OK.
10. FreeBSD Disklabel Editor - While the partition "ad0s1" (the 20GB one) is selected press A to automatically distribute system partitions on it.
11. FreeBSD Disklabel Editor - The result should look similar to this:

Part	Mount	Size
------	-------	------

ad0s1a	/	1024MB
ad0s1b	swap	2006MB
ad0s1d	/var	5099MB
ad0s1e	/tmp	1024MB
ad0s1f	/usr	11326MB

12. FreeBSD Disklabel Editor - Press Q.
13. Choose Distributions - Select the "6 User" distribution.
14. FreeBSD Documentation Installation Menu - Select "en" and then "X Exit".
15. Ports collection installation - Select "Yes".
16. Choose Distributions - Select the "X Exit" to exit. Press OK.
17. Choose Installation Media - Select "1 CD/DVD Install from a FreeBSD CD/DVD".
18. Continue through the Installation - Select "Yes".
19. Wait until the installation is over.
20. Message - Press OK.
21. Configure any ethernet devices? - Select "Yes".
  - (a) Select the wanted ethernet device, e.g. em0.
  - (b) IPv6 configuration - Select "No".
  - (c) DHCP configuration - Select "Yes".
  - (d) Enter the hostname, e.g. IMUNES1.
  - (e) Select "OK".
22. Network gateway - Select "No".
23. Install inetd services - Select "No".
24. Allow SSH login - Select "Yes".
25. Allow anonymous FTP access - Select "No".
26. NFS server - Select "No".
27. NFS client - Select "No".
28. Customize system console settings - Select "No".
29. Set time zone - Select "No".
30. Configure mouse - Select "No".
31. Browse package collection - Select "No".
32. Add additional user accounts - Select "No".
33. Message - Set root password - Press "OK". Enter password. Re-enter password.
34. Visit the general configuration menu - Select "No".
35. Main Menu - Select the "X Exit Install" option by using the "Tab" key.
36. Are you sure you want to exit? - Select "Yes".
37. Message - Reboot - Press "OK".

### A.1.3 Installing the FreeBSD X11 system - GUI

Login into the machine as root. Edit the file `/etc/rc.conf` with an editor e.g. `vi` and add the following lines at the end of the file:

```
hald_enable="YES"
dbus_enable="YES"
```

```
zfs_enable="YES"
```

Save changes and exit.

```
# pkg_add -r xorg bash xpdf vim geany zip unzip xterm firefox
```

You can choose between multiple desktop environments:

- Gnome2-lite - complete, user-friendly desktop  
# pkg\_add -r gnome2-lite
- XFCE 4 - lightweight desktop environment  
# pkg\_add -r xfce4
- IceWM - extremely lightweight window manager, low resource usage  
# pkg\_add -r icewm

After the installation of one or multiple window manager we need to configure which will be used. Edit the file `.xinitrc` in the user folder and add a line depending on which window manager you want to use:

- Gnome2-lite  
exec gnome-session
- XFCE 4  
exec startxfce4
- IceWM  
exec icewm-session

Reboot the machine by issuing the `reboot` command:

```
# reboot
```

After the machine has rebooted, log in and issue the `startx` command to start the chosen window manager:

```
# startx
```

### A.1.4 Installing IMUNES

First we need to install the packages needed for IMUNES. To do this execute the following command:

```
# pkg_add -r tk86 ImageMagick tcllib libimg wireshark
```

After all the packages are installed we need to use the second partition we prepared during the installation (the one with the size 5GB, probably named `ad0s2`). This partition will be used to store the data that will be in the virtual nodes. For this purpose we will create a ZFS pool on the second partition with the command `zpool create` assuming that the second partition is named `ad0s2`:

```
# zpool create vroot /dev/ad0s2
```

Now we need to fetch the IMUNES source/tarball from the official imunes site:

```
http://imunes.tel.fer.hr/imunes-1.0.tar.gz
```

To fetch you can use the firefox browser that we installed earlier or use the `fetch` command:

```
# fetch http://imunes.tel.fer.hr/imunes-1.0.tar.gz
```

To extract the tarball use the following command:

```
# tar xf imunes-1.0.tar.gz
```

Enter the extracted directory:

```
# cd imunes
```

Now we need to install IMUNES and populate the ZFS file system with predefined and required data. To install imunes on the system execute:

```
# make install
```

To setup the ZFS file system execute:

```
# make vroot
```

Both of these commands can be executed together by executing:

```
# make
```

Now the IMUNES GUI can be ran just by typing the `imunes` command in the terminal:

```
# imunes
```

The GUI allows the specification of network topologies but to execute experiments the FreeBSD kernel must be compiled with a few additional options.

### A.1.5 Recompiling the FreeBSD kernel with VIMAGE support

First, the kernel sources need to be fetched, this can be done by using the `sysinstall` utility:

1. Start the `sysinstall` utility:

```
# sysinstall
```

2. Go to "Configure".

3. Go to "Distributions".

4. Go to "sys".

5. Mark "src" by pressing the Space key on it.

6. Select "X Exit".

7. Select "X Exit".

8. Select "Install from an FTP server".

9. Choose the "Main server" or a server closer to your location.

10. When the installation is done select press the Esc key until you exit the `sysinstall` utility.

Now the kernel source is available in the folder `/usr/src/sys`. Enter that folder:

```
# cd /usr/src/sys
```

Depending which distribution architecture you have decided to install, i386 or amd64 the next command differs.

For i386 enter the directory `i386/conf`:

```
# cd i386/conf
```

For amd64 enter the directory `amd64/conf`:

```
# cd amd64/conf
```

With an editor (`vi`, `vim` or `geany`) create a file named `VIMAGE`:

```
# vi VIMAGE
```

Insert the following configuration in the `VIMAGE` file:

```
include GENERIC
nooptions FLOWTABLE
options VIMAGE
options VNET_DEBUG
options MROUTING
```

```
options DDB
```

```
options KDB
```

Exit and save the file.

Now execute the following commands:

```
# config VIMAGE
# cd ../compile/VIMAGE
# make cleandepend && make depend
# make
# make install
```

When the last command ends use the `reboot` command to restart the computer:

```
# reboot
```

### A.1.6 Running IMUNES on FreeBSD

After login execute the `startx` command to start the window manager. Open a terminal and start `imunes`:

```
# imunes
```

Now you can create an arbitrary network topology or something like shown in Figure 4.1 and execute the experiment.

## A.2 Installation of IMUNES on FreeBSD 9

### A.2.1 Installing FreeBSD

A comprehensive and explanatory guide for installing configuring and using FreeBSD can be found here:

<http://www.freebsd.org/doc/handbook/>

Section 2 of the handbook describes the installation of the FreeBSD operating system:

<http://www.freebsd.org/doc/handbook/bsdinstall-start.html>

You can choose to install FreeBSD with two different architectures:

- i386 - 32-bit - works on most personal computers.
- amd64 - 64-bit - works on newer computers that support 64-bit processing. Adds support for more RAM.

### A.2.2 Step by step guide through the FreeBSD installation

1. Insert the FreeBSD-9.2-RELEASE or FreeBSD-9.3-RELEASE (i386 or amd64) medium on startup. Boot from it.
2. Welcome - Choose "Install".
3. Keymap Selection
  - (a) FreeBSD 9.3 - Press "Select" to "Continue with default keymap" or choose an alternative one.
  - (b) FreeBSD 9.2 - If you use a non-default key mapping for your keyboard, choose "Yes" and find your keyboard. Otherwise, choose "No".
4. Set Hostname - Set the hostname for your newly installed system (e.g. "imunes").
5. Distribution Select - Select "src" and "lib32" (only on amd64 architecture) and deselect "games" and "ports".
6. Partitioning - choose Guided or Manual
  - Guided - Choose a disk from the list (e.g. "ada0"). Select "Entire Disk". In the Confirmation window choose "Yes" to erase the entire disk. In the "Partition Editor" review the changes. After finishing partitioning, choose "Finish" and then "Commit".
  - Manual - example for manual partitioning for a 40GB slice:

Part	Type	Mount	Size
ada0p1	freebsd-boot		64kB
ada0p2	freebsd-ufs	/	2GB
ada0p3	freebsd-ufs	/var	8GB
ada0p4	freebsd-ufs	/tmp	4GB
ada0p5	freebsd-ufs	/usr	24GB
ada0p6	freebsd-swap	none	2GB

After finishing partitioning, choose "Finish" and then "Commit". NOTE: freebsd-boot boot partition will be created upon creation of the first freebsd-ufs partition.

7. Archive extraction - Wait for the installation to complete.
8. FreeBSD Installer - Enter your system administrator (root) password. Repeat the password.
9. Network Configuration - Choose a network interface to configure (if any available).
  - (a) Select "Yes" to configure IPv4 network if you use it. Then select "Yes" to configure DHCP if you use it.
  - (b) Select "Yes" to configure IPv6 network if you use it. Then select "Yes" to configure SLAAC if you use it.
  - (c) Resolver Configuration - If DHCP/SLAAC is used to configure the interface, some values may already be present. Otherwise, enter them yourself.
10. Timezones - If your timezone is not UTC, select "No" and choose your region and country (e.g. Europe->Croatia) and select "Yes" if asked for confirmation.
11. System Configuration
  - (a) FreeBSD 9.3 - Choose the "sshd" service to be started at boot and select "dumpdev" to enable crash dumps in /var.
  - (b) FreeBSD 9.2 - Choose the "sshd" service to be started at boot.
12. (FreeBSD 9.2 only) Dumpdev Configuration - Select "Yes" for enabling crash dumps in /var folder.
13. Add User Accounts - If necessary, add additional users now.
14. Final Configuration - Choose "Exit".
15. Manual Configuration - Nothing else is needed, so choose "No" to skip manual configuration.
16. Complete - Reboot. (Don't forget to remove your FreeBSD boot media.)

### A.2.3 Installing the FreeBSD X11 system - GUI

Login into the machine as root. Edit the file `/etc/rc.conf` with an editor e.g. `vi` and add the following lines at the end of the file:

```
hald_enable="YES"
dbus_enable="YES"
```

Save changes and exit.

```
# pkg_add -r xorg bash xpdf vim geany zip unzip xterm firefox subversion
```

You can choose between multiple desktop environments:

- Gnome2-lite - complete, user-friendly desktop
 

```
# pkg_add -r gnome2-lite
```
- XFCE 4 - lightweight desktop environment
 

```
# pkg_add -r xfce4
```



- IceWM - extremely lightweight window manager, low resource usage  
# pkg\_add -r icewm

After the installation of one or multiple window manager we need to configure which will be used. Edit the file `.xinitrc` in the user folder and add a line depending on which window manager you want to use:

- Gnome2-lite  
exec gnome-session
- XFCE 4  
exec startxfce4
- IceWM  
exec icewm-session

Reboot the machine by issuing the `reboot` command:

```
# reboot
```

After the machine has rebooted, log in and issue the `startx` command to start the chosen window manager:

```
# startx
```

## A.2.4 Installing IMUNES

First we need to install the packages needed for IMUNES. To do this execute the following command:

```
# pkg_add -r tk86 ImageMagick tcllib wireshark socat gmake
```

Now we need to fetch the IMUNES source/tarball from the official imunes site:

```
http://imunes.tel.fer.hr/imunes-1.0.tar.gz
```

To fetch you can use the firefox browser that we installed earlier or use the `fetch` command. Then extract the tarball and enter the directory:

```
# fetch http://imunes.tel.fer.hr/imunes-1.0.tar.gz
# tar xf imunes-1.0.tar.gz
# cd imunes_version
```

Alternatively, you could checkout the last fresh IMUNES source through the public SVN repository:

```
# svn co svn://imunes.tel.fer.hr/head/ imunes
# cd imunes
```

Now we need to install IMUNES and populate the virtual file system with predefined and required data. To install imunes on the system execute:

```
# make install
```

To setup the virtual file system execute:

```
# make vroot
```

Both of these commands can be executed together by executing:

```
# make
```

Now the IMUNES GUI can be ran just by typing the `imunes` command in the terminal:

```
# imunes
```

The GUI allows the specification of network topologies but to execute experiments the FreeBSD kernel must be compiled with a few additional options.

### A.2.5 Recompiling the FreeBSD kernel with VIMAGE support

If you didn't include the `src` option in step 4 while installing FreeBSD, the kernel sources need to be fetched. Use your selected architecture and release combination:

```
# fetch ftp://ftp.freebsd.org/pub/FreeBSD/releases/arch/release/src.txz
```

e.g. for FreeBSD 9.2 with amd64 architecture use:

```
# fetch ftp://ftp.freebsd.org/pub/FreeBSD/releases/amd64/9.2-RELEASE/src.txz
```

Extract it by using:

```
# tar xf src.txz -C /
```

Now the kernel source is available in the folder `/usr/src/sys`. Enter that folder:

```
# cd /usr/src/sys
```

Depending which distribution architecture you have decided to install, i386 or amd64 the next command differs.

For i386 enter the directory `i386/conf`:

```
# cd i386/conf
```

For amd64 enter the directory `amd64/conf`:

```
# cd amd64/conf
```

With an editor (`vi`, `vim` or `geany`) create a file named `VIMAGE`:

```
# vi VIMAGE
```

Insert the following configuration in the `VIMAGE` file:

```
include GENERIC
nooptions FLOWTABLE
options VIMAGE
options VNET_DEBUG
options MROUTING
```

```
options IPSEC
device crypto
options IPSEC_DEBUG
```

```
options DDB
options KDB
```

Exit and save the file.

Now execute the following commands:

```
# config VIMAGE
# cd ../compile/VIMAGE
# make cleandepend && make depend
# make
# make install
```

When the last command ends use the `reboot` command to restart the computer:

```
# reboot
```

### A.2.6 Running IMUNES on FreeBSD

After login execute the `startx` command to start the window manager. Open a terminal and start `imunes`:

```
# imunes
```

Now you can create an arbitrary network topology or something like shown in Figure 4.1 and execute the experiment.

## A.3 Installation of IMUNES on FreeBSD 10

### A.3.1 Installing FreeBSD

A comprehensive and explanatory guide for installing configuring and using FreeBSD can be found here:

<http://www.freebsd.org/doc/handbook/>

Section 2 of the handbook describes the installation of the FreeBSD operating system:

<http://www.freebsd.org/doc/handbook/bsdinstall-start.html>

You can choose to install FreeBSD with two different architectures:

- i386 - 32-bit - works on most personal computers.
- amd64 - 64-bit - works on newer computers that support 64-bit processing. Adds support for more RAM.

### A.3.2 Step by step guide through the FreeBSD installation

1. Insert the FreeBSD-10.x-RELEASE (i386 or amd64) medium on startup. Boot from it.
2. Welcome - Choose "Install".
3. Keymap Selection - Press "Select" to "Continue with default keymap" or choose an alternative one.
4. Set Hostname - Set the hostname for your newly installed system (e.g. "imunes").
5. Distribution Select - Select "src" and "lib32" (only on amd64 architecture) and deselect "games" and "ports".
6. Partitioning - choose Auto or Manual
  - Auto - Choose a disk from the list (e.g. "ada0"). Select "Entire Disk". In the Confirmation window choose "Yes" to erase the entire disk. In the "Partition Editor" review the changes. After finishing partitioning, choose "Finish" and then "Commit".
  - Manual - example for manual partitioning for a 40GB slice:

Part	Type	Mount	Size
ada0p1	freebsd-boot		64kB
ada0p2	freebsd-ufs	/	4GB
ada0p3	freebsd-ufs	/var	10GB
ada0p4	freebsd-ufs	/usr	24GB
ada0p5	freebsd-swap	none	2GB

After finishing partitioning, choose "Finish" and then "Commit". NOTE: freebsd-boot boot partition will be created upon creation of the first freebsd-ufs partition.

7. Archive extraction - Wait for the installation to complete.
8. FreeBSD Installer - Enter your system administrator (root) password. Repeat the password.
9. Network Configuration - Choose a network interface to configure (if any available).
  - (a) Select "Yes" to configure IPv4 network if you use it. Then select "Yes" to configure DHCP if you use it.
  - (b) Select "Yes" to configure IPv6 network if you use it. Then select "Yes" to configure SLAAC if you use it.

- (c) Resolver Configuration - If DHCP/SLAAC is used to configure the interface, some values may already be present. Otherwise, enter them yourself.
- 10. Timezones - If your timezone is not UTC, select "No" and choose your region and country (e.g. Europe->Croatia) and select "Yes" if asked for confirmation.
- 11. System Configuration - Choose the "sshd" service to be started at boot and select "dumpdev" to enable crash dumps in /var.
- 12. Add User Accounts - If necessary, add additional users now.
- 13. Final Configuration - Choose "Exit".
- 14. Manual Configuration - Nothing else is needed, so choose "No" to skip manual configuration.
- 15. Complete - Reboot. (Don't forget to remove your FreeBSD boot media.)

### A.3.3 Installing the FreeBSD X11 system - GUI

Login into the machine as root. Edit the file `/etc/rc.conf` with an editor e.g. `vi` and add the following lines at the end of the file:

```
hald_enable="YES"
dbus_enable="YES"
```

Save changes and exit.

If necessary, bootstrap pkg:

```
# env ASSUME_ALWAYS_YES=YES pkg bootstrap
```

Install necessary packages:

```
# pkg install xorg bash xpdf vim geany zip unzip xterm firefox wget
```

You can choose between multiple desktop environments:

- Gnome3-lite - complete, user-friendly desktop  
# `pkg install gnome3-lite`
- XFCE 4 - lightweight desktop environment  
# `pkg install xfce`
- IceWM - extremely lightweight window manager, low resource usage  
# `pkg install icewm`

After the installation of one or multiple window manager we need to configure which will be used. Edit the file `.xinitrc` in the user folder and add a line depending on which window manager you want to use:

- Gnome3-lite  
  `exec gnome-session`
- XFCE 4  
  `exec startxfce4`
- IceWM  
  `exec icewm-session`

Reboot the machine by issuing the `reboot` command:

```
# reboot
```

After the machine has rebooted, log in and issue the `startx` command to start the chosen window manager:

```
# startx
```

### A.3.4 Installing IMUNES

First we need to install the packages needed for IMUNES. To do this execute the following command:

```
# pkg install tk86 ImageMagick tcllib wireshark socat git gmake
```

Checkout the last fresh IMUNES source through the public github repository:

```
# git clone https://github.com/imunes/imunes.git
```

Now we need to install IMUNES and populate the virtual file system with predefined and required data. To install IMUNES on the system execute:

```
# cd imunes
# make install
```

To setup the virtual file system execute:

```
# imunes -p
```

Now the IMUNES GUI can be ran just by typing the `imunes` command in the terminal:

```
# imunes
```

The GUI allows the specification of network topologies but to execute experiments the FreeBSD kernel must be compiled with a few additional options.

### A.3.5 Recompiling the FreeBSD kernel with VIMAGE support

If you didn't include the `src` option in step 4 while installing FreeBSD, the kernel sources need to be fetched. Use your selected architecture and release combination:

```
# fetch ftp://ftp.freebsd.org/pub/FreeBSD/releases/arch/release/src.txz
```

e.g. for FreeBSD 10.1 with amd64 architecture use:

```
# fetch ftp://ftp.freebsd.org/pub/FreeBSD/releases/amd64/10.1-RELEASE/src.txz
```

Extract it by using:

```
# tar xf src.txz -C /
```

Now the kernel source is available in the folder `/usr/src/sys`. Enter that folder:

```
# cd /usr/src/sys
```

Depending which distribution architecture you have decided to install, i386 or amd64 the next command differs.

For i386 enter the directory `i386/conf`:

```
# cd i386/conf
```

For amd64 enter the directory `amd64/conf`:

```
# cd amd64/conf
```

With an editor (`vi`, `vim` or `geany`) create a file named `VIMAGE`:

```
# vi VIMAGE
```

Insert the following configuration in the `VIMAGE` file:

```
include GENERIC
nooptions FLOWTABLE
nooptions SCTP
options VIMAGE
options VNET_DEBUG
options DDB
```

```
options IPSEC
device crypto
options IPSEC_DEBUG
options IPSEC_NAT_T
```

Exit and save the file.

Now execute the following commands:

```
# config VIMAGE
# cd ../compile/VIMAGE
```

If your machine has multiple cores, you can use the `-j` flag to distribute the make process on a number of cores. For example, to use 4 cores, run:

```
# make -j4 depend
# make -j4
# make install
```

If you're running FreeBSD on a single core machine, just use the standard make commands:

```
# make depend
# make
# make install
```

When the last command ends use the `reboot` command to restart the computer:

```
# reboot
```

### A.3.6 Running IMUNES on FreeBSD

After login execute the `startx` command to start the window manager. Open a terminal and start imunes:

```
# imunes
```

Now you can create an arbitrary network topology or something like shown in Figure 4.1 and execute the experiment.

## A.4 Running IMUNES with VMware Player

We also have a VMware image containing the FreeBSD operating system including a complete and working installation of IMUNES.

### A.4.1 Installing VMware

To run the image VMware Player (or any other similar VMware product) needs to be installed. VMware player can be downloaded from the following address (after registration with VMware):

<http://www.vmware.com/products/player/>

The installation procedure is clearly explained in the VMware "Getting Started Guide" ([http://www.vmware.com/pdf/vmware\\_player310.pdf](http://www.vmware.com/pdf/vmware_player310.pdf))

### A.4.2 Downloading the VMware image

The VMware image can be downloaded from the following address:

<http://imunes.tel.fer.hr/imunes/dl/index.html>

There are two archives, one in the RAR format and other in the ZIP format. They both contain the same image but the RAR file is considerably smaller. Download one of them and then unpack the contents of the archive in a folder with enough space (The unpacked image is almost 2.3GB large, but for runtime execution 4GB is needed.)

### A.4.3 Running the VMware image

After the image is extracted go into the image directory. There should be two files, a `vmdk` file containing the disk image and a `vmx` file containing the VMware machine settings. Double-clicking the `vmx` file will open the VMware Player and start booting the image. A window will

pop-up asking you whether you moved or copied the image. Select the option "I copied it" and click on the "OK" button.

After a few minutes FreeBSD with the graphical interface will start. On the desktop there is an IMUNES icon to start IMUNES.

## A.5 Installation of the IMUNES GUI on Linux

**NOTE:** Although you can draw network topology on any system that supports Tcl/Tk (Linux, FreeBSD, Windows, Mac OS X, Solaris), an experiment can only be started on FreeBSD operation system with root permissions (Figure 4.15 and Figure 4.16)!

The first thing you need to do is install Tcl/Tk. The easiest way to do this is using ActiveState ActiveTcl 8.6. To download the installation files visit:

<http://www.activestate.com/activetcl/downloads>

Download the version 8.6, as IMUNES cannot run on earlier versions. Download the version that is designed for your operating system: x86 if you're using 32bit Linux or x86\_64 if you're using 64bit Linux. Select the option *AS package*.

When the package is downloaded extract it using an archive manager or through the shell with the command:

```
# tar xzvf filename.tar.gz
```

Now enter the extracted directory and execute the file `install.sh` with root permissions. In the shell this is done with the line

```
# sudo ./install.sh
```

and provide the superuser password. If you don't have superuser permissions you can install ActiveTcl in your user directory. Make a directory in which ActiveTcl will be installed, and during the installation provide the path to that directory. After the installation you just need to put the ActiveTcl bin directory in the PATH variable.

For a csh or compatible (tcsh) perform:

```
# setenv PATH "/opt/ActiveTcl/bin:$PATH"
```

For a sh or similar perform:

```
# PATH="/opt/ActiveTcl/bin:$PATH"
```

```
# export PATH
```

To make the changes permanent add this lines to your shell rc configuration file (e.g. `.bashrc` for bash, `.cshrc` for csh).

To run IMUNES with full functionality you need the `imagemagick` package. You can install it through package managers or through shell.

On Debian, Ubuntu and other Debian based distributions you will use:

```
# sudo apt-get install imagemagick
```

On Fedora, Red Hat and other Red Hat based distributions you will use:

```
# su
```

```
# yum install imagemagick
```

Now we can fetch the latest version of IMUNES through CVS. To do this you need to have the `cvs` package installed. This is done the same way as the ImageMagick installation:

```
# sudo apt-get install cvs
```

or

```
# su
```

```
# yum install cvs
```

The line for fetching IMUNES from CVS is as follows:

```
# cvs -d :pserver:anonymous@imunes.net/usr/local/src/cvsroot co imunes
```

Now enter the imunes directory and start IMUNES:

```
# cd imunes
```

```
# ./imunes
```

## A.6 Installation of the IMUNES GUI on Windows

**NOTE:** Although you can draw network topology on any system that supports Tcl/Tk (Linux, FreeBSD, Windows, Mac OS X, Solaris), an experiment can only be started on FreeBSD operation system with root permissions (Figure 4.15 and Figure 4.16)!

The first thing you need to do is install Tcl/Tk. The easiest way to do this is using ActiveState ActiveTcl 8.6. To download the installation files visit:

<http://www.activestate.com/activetcl/downloads>

Download the version 8.6, as IMUNES cannot run on earlier versions. Download the package by clicking on the *Windows Installer (EXE)* option. When the file is downloaded start the installation by double clicking on the file. Follow the installation instructions to the end.

Now, to enable the full functionality of IMUNES you need to download the ImageMagick program from:

<http://www.imagemagick.org/script/binary-releases.php>

Go to the *Windows Binary Release* section. Select the appropriate file. Download it and install it following the installation instructions.

To start IMUNES you need the latest IMUNES release. If you have access to a Unix/Linux system you can download it like it is shown at the end of the previous section (Section A.5). Then transfer it on the Windows machine.

Enter the imunes directory and double click the file `imunes.tcl` to start IMUNES.



---

## Appendix B Troubleshooting

---

### B.1 Terminating all active experiments

To terminate all active experiments a clean-up script is available in the IMUNES system. The script is invoked by issuing the command `cleanupAll` with root privileges. This script will terminate all running experiments.

```
# cleanupAll
```

#### B.1.1 Cleaning up hanging ZFS mounts

If a machine running an experiment is rebooted the experiment will not be available after boot but the ZFS mounts needed for network nodes will remain available after the restart. The `cleanupAll` tool is also used for destroying the remaining ZFS mounts.

### B.2 Restoring original ZFS snapshot

**NOTE:** New versions of IMUNES use Unionfs instead of ZFS.

The ZFS snapshot used for replicating on virtual nodes can be corrupted or deleted. To restore the ZFS snapshot to the original state we first need to destroy the existing default snapshot named `vroot/vroot@clean`. Before destroying the ZFS snapshot make sure that no experiments are currently running by using the command `himage -l` and use the `imunes -b eid` option or the `cleanupAll` tool to terminate them. removal of the ZFS snapshot is done by issuing the following command with root user privileges:

```
# zfs destroy -fR vroot/vroot@clean
```

After the procedure is complete download the IMUNES tarball from the IMUNES website (<http://imunes.tel.fer.hr/dl/imunes-1.0.tar.gz>) and unpack it. Then enter the directory and run the command `make vroot` with root privileges to restore the ZFS snapshot. The procedure done as follows:

```
# tar xf imunes-1.0.tar.gz
# cd imunes_version
# make vroot_zfs
```

**NOTE:** To restore the snapshot an Internet connection must be available.

### B.3 Obtaining kernel panic traces

In case a specific experiment configuration, workload type or hardware configuration triggers operating system crashes, obtaining traces of such events may be instrumental for successful debugging and resolving the observed operating system level bugs. The following procedure is recommended for collecting kernel panic traces:

Add the following line to the `/etc/rc.conf` file:

```
dumpdev="AUTO"
```

Create a new file `/usr/local/etc/rc.d/textdump` and populate it with the following script:

```
#!/bin/sh
#
# PROVIDE: textdump
```

```
# REQUIRE: LOGIN
#
```

```
ddb script kdb.enter.default="textdump set; capture on; bt;\nshow reg; show pcpu; show vnets; call doadump; reset"
```

Set execution bit on /usr/local/etc/rc.d/textdump file:

```
# chmod +x /usr/local/etc/rc.d/textdump
```

Once rebooted, the machine should be from now on configured to automatically store kernel panic traces in /var/crash directory. Here's an example of collection of kernel panic trace files:

```
# ls -l /var/crash/
total 96
-rw-r--r--  1 root  wheel      2 Sep 20 00:06 bounds
-rw-----  1 root  wheel    440 Sep 19 23:57 info.0
-rw-----  1 root  wheel    442 Sep 20 00:00 info.1
-rw-----  1 root  wheel    442 Sep 20 00:06 info.2
-rw-----  1 root  wheel   24576 Sep 19 23:57 textdump.tar.0
-rw-----  1 root  wheel   39424 Sep 20 00:00 textdump.tar.1
-rw-----  1 root  wheel   24576 Sep 20 00:06 textdump.tar.2
#
```

Such "textdump" tarballs should be submitted along with any reports of kernel crashes.

---

## Appendix C IMUNES network configuration file

---

Here is the example of IMUNES network configuration file for the network topology shown in Figure C.1

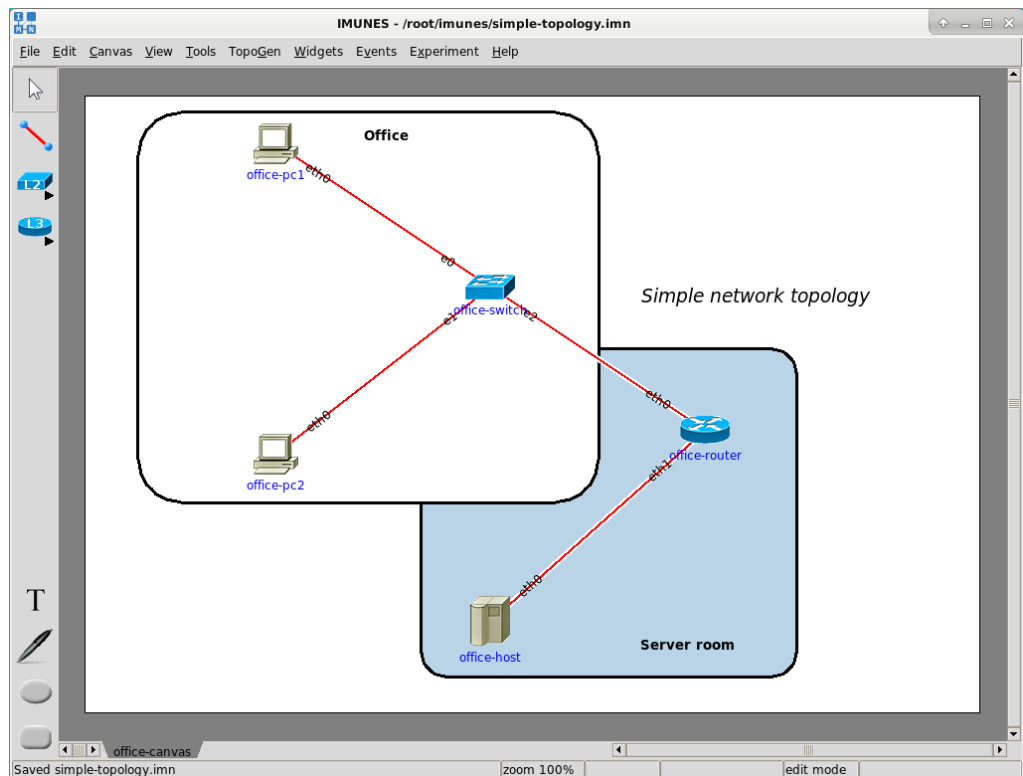


Figure C.1: Network topology

```
node n0 {
  type router
  model quagga
  network-config {
    hostname office-router
    !
    interface eth0
      mac address 42:00:aa:00:00:02
      ip address 192.168.1.1/24
    !
    interface eth1
      ipv6 address fc00:1::1/64
      mac address 42:00:aa:00:00:03
      ip address 192.168.2.1/24
    !
    interface lo0
      type lo
      ip address 127.0.0.1/8
  }
}
```

```

        ipv6 address ::1/128
        !
    router rip
        redistribute static
        redistribute connected
        redistribute ospf
        network 0.0.0.0/0
        !
    router ripng
        redistribute static
        redistribute connected
        redistribute ospf6
        network ::/0
        !
    }
    canvas c0
    iconcoords {624 336}
    labelcoords {624 361}
    interface-peer {eth0 n2}
    interface-peer {eth1 n1}
}

node n1 {
    type host
    network-config {
        hostname office-host
        !
        interface eth0
            mac address 42:00:aa:00:00:04
            ip address 192.168.2.5/24
            !
        interface lo0
            type lo
            ip address 127.0.0.1/8
            ipv6 address ::1/128
            !
        ip route 0.0.0.0/0 192.168.2.1
        !
        !
    }
    canvas c0
    iconcoords {408 528}
    labelcoords {408 564}
    interface-peer {eth0 n0}
}

node n2 {
    type lanswitch
    network-config {
        hostname office-switch
        !
        interface e2
            fair-queue
            !
        interface e1
            fair-queue
    }
}

```

```

        !
        interface e0
        fair-queue
        !
    }
    canvas c0
    iconcoords {408 192}
    labelcoords {408 215}
    interface-peer {e0 n3}
    interface-peer {e1 n4}
    interface-peer {e2 n0}
}

node n3 {
    type pc
    network-config {
        hostname office-pc1
        !
        interface lo0
        type lo
        ip address 127.0.0.1/8
        ipv6 address ::1/128
        !
        interface eth0
        mtu 1500
        mac address 42:00:aa:00:00:00
        ip address 192.168.1.5/24
        !
        !
        ip route 192.168.2.0/24 192.168.1.1
        !
    }
    canvas c0
    iconcoords {192 48}
    labelcoords {192 79}
    interface-peer {eth0 n2}
    custom-configs {
        custom-config-id newconf {
            custom-command /bin/sh
            config {
                ifconfig lo0 inet 127.0.0.1/8
                ifconfig eth0 inet 192.168.1.5/24
                ifconfig lo0 inet6 ::1

                route -q add -inet 192.168.2.0/24 192.168.1.1

                echo "Success!" > /tmp/log
                ifconfig vlan0
            }
        }
    }
    custom-enabled true
    custom-selected newconf
}

node n4 {

```

```

type pc
network-config {
    hostname office-pc2
    !
    interface eth0
        mac address 42:00:aa:00:00:01
        ip address 192.168.1.7/24
    !
    interface lo0
        type lo
        ip address 127.0.0.1/8
        ipv6 address ::1/128
    !
    ip route 192.168.2.0/24 192.168.1.1
    !
    !
}
canvas c0
iconcoords {192 360}
labelcoords {192 391}
interface-peer {eth0 n2}
}

link l0 {
    nodes {n3 n2}
}

link l1 {
    nodes {n4 n2}
}

link l2 {
    delay 30000
    nodes {n2 n0}
    bandwidth 0
}

link l3 {
    nodes {n0 n1}
    bandwidth 0
}

annotation a0 {
    type rectangle
    iconcoords {53 16 517 409}
    color #ffffff
    bordercolor black
    width 3
    rad 53.6
    canvas c0
}

annotation a1 {
    type text
    iconcoords {281 40}
    label {Office}
}

```

```
        labelcolor #000000
        font {-family {DejaVu Sans} -size 10 -weight bold -slant roman -underline 0 -overs
        canvas c0
    }

    annotation a2 {
        type rectangle
        iconcoords {338 254 716 584}
        color #b8d4e6
        bordercolor #000000
        width 3
        rad 30.78918918918919
        canvas c0
    }

    annotation a3 {
        type text
        iconcoords {559 552}
        label {Server room}
        labelcolor black
        font {-family {DejaVu Sans} -size 10 -weight bold -slant roman -underline 0 -overs
        canvas c0
    }

    annotation a4 {
        type text
        iconcoords {560 201}
        label {Simple network topology}
        labelcolor black
        font {-family {DejaVu Sans} -size 14 -weight normal -slant italic -underline 0 -ov
        canvas c0
    }

    canvas c0 {
        name {office-canvas}
    }

    option show {
        interface_names yes
        ip_addresses no
        ipv6_addresses no
        node_labels yes
        link_labels no
        background_images no
        annotations yes
        hostsAutoAssign no
        grid no
        iconSize normal
        zoom 1.0
    }
```