Chapter 4

A Virtual Routing Solution for IP Networks

Virgilius-Aurelian Minuță and Eugen Petac

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/65415

Abstract

The core of our paper is represented by the development of the HypeRSimRIP application that can be used for various networking purposes, such as designing a network or setting routing processes. Likewise, the application implements didactical functions useful for teaching networking-related concepts and experimental capabilities that enable its users to recreate famous experiments.

Keywords: Hypervisor, network, simulator, RIP

1. Introduction

The concept of virtual has different connotations, depending on the domain in which it is used: philosophy [1], current speech [2], creation of artificial ambients of virtual reality type [3], etc. Most of the time, the term usually refers to applications and services from the Internet. *The virtual experiment* represents an alternative, or better said a complementary, resource in the study of phenomena and processes.

The basis of the concept of *hardware virtualization* (also known as platform virtualization) consists in the use of software in order to simulate the existence of a hardware component. This allows the use of multiple independent Information and Communications Technology (ICT) systems within a single physical computational system. The term 'virtualization', in this sense, was used even in the 1960's mainframes, and it represented as a method of partitioning the resources of a mainframe to all its diverse applications [4].

A *virtual machine* (VM) is a software application that completely simulates all the functions of all hardware components of an informatics system (particularly, a computer). If on a real hardware system (further named as host) are installed multiple virtual machines, each one of them will function as an independent computational system, complete with its own processor,

its own RAM memory, its own Hard Disk Drive (HDD), etc. The operating system that allows the creation and continued simulation of more virtual machines is called a *Hypervisor* or a virtual machine manager.

At first, hypervisors can be classified into two distinct types: hypervisors of type 1 and hypervisors of type 2. Those of type 1 represent a hypervisor that runs directly on the host. Those of type 2 run within an operating system, which in turn runs on the host [5]. Hypervisors of type 1 are therefore more efficient, but because of this, their monetary cost increases considerably. Thus, in order to experience virtualization, it is often preferred to use a type 2 hypervisor, before entering in the possession of a type 1 hypervisor.

VMware vSphere (known also under the name of ESXi) [6] represents a type 1 hypervisor for servers. ESXi runs directly on the server (the host) and allows the creation and utilization of multiple virtual servers. VMware Workstation [7] is a type 2 hypervisor; it allows the user to create and run in parallel, multiple instances of virtual machines with operating systems compatible to x86 or x86-64 architectures on the same physical host.

Depending on the level of virtualization, hypervisors can be classified [7] as having:

- Full virtualization—the simulation of all the hardware components is almost complete, thus allowing an unaltered run of the operating system of the virtual machine.
- Partial virtualization—just a part of the hardware components will be simulated, thus applications need some changes for a proper use inside the virtual machine.
- Paravirtualization—the hardware medium is not simulated at all; the programs 'virtual machines' are executed in their own separated domain.

Desktop (work area) virtualization represents the concept of separation between the logical desktop and the physical machine. A form of desktop virtualization is represented by virtual desktop interface (VDI) that represents a more advanced form of hardware virtualization. Instead of having to interact with a host directly with a mouse, a keyboard and/or a monitor, the user can interact with the host via another machine—another desktop computer, or a smartphone, etc. This can be accomplished through a LAN network connection or a wireless LAN network connection or even the Internet. In this situation, the host becomes a server computer capable of running several virtual machines at the same time for multiple users. As a first example, the TeamViewer application, developed by TeamViewer GmbH [8], allows remote access and control of a desktop and file transfer between the two devices (a host PC (personal computer) and another device such as a PC or a smartphone). As a second example, more advanced than a connection to a single desktop (TeamViewer's situation), HP and IBM companies offer a VDI hybrid, with a series of virtualization software, in order to improve the computational limitations of a client [9].

Session virtualization allows multiple users to connect and authenticate simultaneously to a more powerful computer via a network. Each user has a personal desktop and a personal folder in which he can store his data [10]. With the multi-seat configuration, session virtualization can be achieved by connecting more input/output devices to a single personal computer.

Another form of virtualization consists in the movement of all the desktops to a cloud, thus creating hosted virtual desktops (HVDs) in which the desktops are stored and maintained by a hosting specialized business. Here, the benefits include a drastic decrease of the investment funds (in informatics equipment), which is replaced by a monthly 'rent' towards the hosting business [11].

In the second section of this paper, the application HypeRSimRIP ('hypervisor de Rețele cu simulator RIP' [Ro]—network hypervisor with integrated RIP simulator [Eng.]) is presented. This application is a type 2 hypervisor with full virtualization, which was developed by the authors in order to allow the construction, management and observation of a virtual system of Internet Protocol (IP) networks. The HypeRSimRIP application is useful in realizing virtual experiments dedicated to the study of IP networks. Moreover, the subject is of huge interest, considering that the world has embraced the Internet of Everything (IoE) concept—the connection of the Internet to all humans, processes, data and objects. The theoretical aspects, specific to IP networks, like the Open Systems Interconnection model (OSI) and TCP/IP models, the TCP/IP protocol suite, IPv4 addressing (sub-netting and variable length sub-net mask (VLSM) methods), the routing process, the RIPv1, RIPv2 and RIPng protocols are considered known, and they will not be presented in this paper. For further references, please consider [12–17].

In the third and final section of this paper, the functionality of the HypeRSimRIP application is described, alongside with some utilization examples. The application allows the simulation of notable experiments from the IP network domain. The results obtained by HypeRSimRIP are similar to the theoretical results obtained in the literature.

2. The HypeRSimRIP application

The interaction between the user and the HypeRSimRIP application takes place through a graphical user interface (GUI). HypeRSimRIP was developed by the authors in Visual Studio 2012, using the C# programming language, in the Microsoft's.Net Framework (version 4.5) programming platform.

2.1. Design of the main window

The main window of the program is composed of four main areas (see Figure 1):

- The first area (further named as the graphical zone) represents the area in which the network topology can be created, viewed and modified.
- The second area includes all the current routing information of all the routers.
- The third area (further named as the console) includes a history of text messages made for the user: help messages, tutorial messages, announcements about any change in the network topology, reports about sent data packages, etc. The font size of the second and third areas can be adjusted by using the mouse scroll whilst pressing the Ctrl key.

- The forth area includes a diagram that will plot the network usage from the last 60 sec of the simulation (the complete network usage is saved in a text file, at the end of the simulation). This area is detachable from the main window via a double click on it.

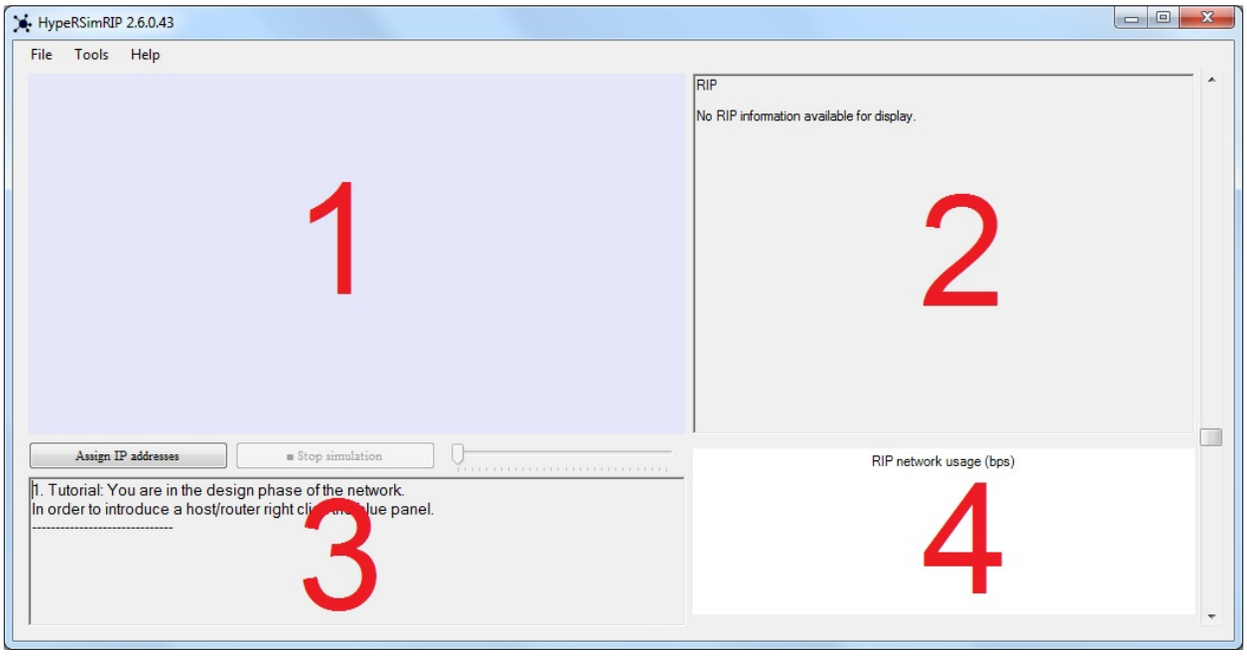

Figure 1. The main window divided into its four main areas.

The main window can be resized upwards of the minimum size of 1000 px (width) × 500 px (height). During the resize process, the following can be observed: the size of graphical zone remains unmodified (500 px × 300 px) for objective reasons (the devices have fixed representation sizes) and only the other areas modify their sizes.

The right-sided scrollbar of the main window (see Figure 1) allows changing the sizes of the routing information area and the network usage diagram area: The ratio determined by the scroll thumb in this scrollbar is equal to the ratio between the heights of the areas in question.

2.1.1. The graphical zone

The graphical zone represents one of the most complex parts of this application.

Its functionality changes depending on the stage in which the application currently is in. Two main stages distinguish themselves:

- The network design stage.
- The RIP protocol simulation stage.

In both stages, the mouse position is continuously monitored, more precisely: the coordinates of the cursor when left click is pressed, its position in the proceeding moments and the coordinates when left click is released. These three events are necessary for a single left click

because the user may want to drag some object from this area. By pressing right click, a contextual menu will appear; its elements will differ depending not only on the stage in which the application is in, but also on the type of the object that was right clicked. More clarification on this subject will be presented in the following sub-sections.

2.1.2. The network design stage

At the beginning of the application, in the network design stage, a right click on the graphical zone will give two device insertion options: host or router.

Programmatically, by choosing one of these options, an object of class type `Device` will be created in a list from the class `Topology`, which will have as coordinates, the upper-left position of the previous contextual menu. These devices will be represented on the graphical zone through discs of different dimensions and colours depending on their respective states. Next to the discs, a unique name will be shown for quick visual identification (see Figure 2).

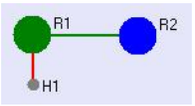

Figure 2. Device representation example: R1 is a selected router; R2 is an online router; H1 is an offline host; the segment [R1,R2] is a connection; the segment [R1,H1] is a deactivated connection.

Once a device has been inserted, the user can select this device or move it around, for repositioning, by dragging it with the mouse.

In order to detect if a device was selected, the application runs the following algorithm:

Step 1. All the devices `d` from the devices list from the `Topology` class are retrieved and checked one by one as in the following algorithm:

a. The radius of the disc representation of the device `d` is calculated depending on its type —5 px for hosts and 15 px for routers.

b. If the Euclidean distance between the coordinates of the device and the position of the left click is smaller than the calculated radius, then the device was selected. **STOP**

Step 2. If this step is reached, then no device has been selected. **STOP**

The same algorithm (with minor changes) is used for the right-click event, in order to determine the contents of contextual menu. The changes consist in:

- If the click was made at less than `Max (radius of host, radius of router)` from a device centre, the opening of the contextual menu will be cancelled because it is not desired to allow the user to insert overlapping devices.
- If the click was not made over any device, it is checked if the right click was made over a connection (as it will be seen in Section 2.3).

In the situation in which the left-click mouse button was clicked over a device, but it was not yet released, mouse movements are monitored as follows:

Step 3. If the Euclidean distance between `P_initial` (the initial point of click) and `P_current` (the current cursor location) is smaller than 6 px, it is considered that the user's hand just trembled when the button was pressed, thus no action will be taken.

Step 4. If the distance increases over 6 px, this means that the user actually wants to move the selected device. This movement will be done continuously until the left click is released (the 6 px protection from step 1 is deactivated). During this movement, two scenarios may occur:

- If the device starts to overlap with another (unwanted behaviour), both the devices are highlighted with yellow. When the overlap stops, they will revert to their original colours. If during the overlap scenario the left click button is released – which represents an illegal movement – the position of the displaced object will be re-established. Note that it will be re-established, not to the position of left click (`P_initial`) but to the original device position.
- If the object approaches or leaves the limits of the graphical zone, the object will be highlighted in red, and at the left-click release event, it will be deleted. Of course, if the object is placed back inside the margins of the graphical zone, without releasing the left-click button, its colour will be reverted to its original, and it will not be deleted at the button release event.

Step 5. If at the left-click button release event, the distance between `P_initial` and `P_current` has never exceeded 6 px, this means the user wanted to select the object, thus it will be highlighted as such (with green).

The management of these warning colours is done through the existence of multiple Boolean-type variables throughout the Device class. They determine the current status of a device. The order of colouring is given by the following method:

```
private void setNormalColor()
  {
   if (flag_deletion)
  {
    colour = Settings.getColor_Device_Dispose();
  }
 else if (flag_overlapp)
  {
    colour = Settings.getColor_Device_Overlap();
  }
 else if (flag_tracert)
  {
    colour = Settings.getColor_Device_Traced();
  }
 else if (flag_selected)
  {
    colour = Settings.getColor_Device_Selected();
  }
 else if (flag_online)
```

```
    {
      colour = Settings.getColor_Device_Online();
    }
   else
    {
      colour = Settings.getColor_Device_Offline();
    }
   }
```

The following order is distinguished: deletion colour, overlap colour, trace colour, selection colour and online/offline status colour.

The selection of two devices during the network design stage will create/delete a connection between the devices, and during the second stage, it will start a 'trace route' type command from the first device to the second (as it will be shown in Section 2.5).

2.2. Efficient interactive training

This version of the HypeRSimRIP application contains a new functionality of interactive training that allows the user to quickly learn how to use the program.

The training consists of a series of tutorial-type messages transmitted to the user via the console in some special identified moments (see Table 1). These messages are shown only once, each one of them having annexed a Boolean variable, which states if the message was previously shown, in which case the triggering event will be ignored.

Triggering event	Message
Start of the program	Tutorial: You are in the design phase of the network. In order to introduce a host/router, right click the blue panel.
Two devices inserted (with at least one being a router)	Tutorial: To create a connection, select two devices (green colour for selected devices).
Selection of a device	Tutorial: To deselect a device, click an empty place of the blue panel.
Creation of a connection	Tutorial: To delete a connection, select again its own two devices. In order to move to the next phase, press the button 'Assign IP addresses'.
Five devices inserted	Tutorial: You can delete a device by dragging it to the margins of the panel (red colour).
Device approaches graphical zone margins	
Overlap of two devices	Tutorial: It is forbidden to overlap devices. Devices in peril of overlapping will have yellow colour.

Triggering event	Message
The network simulation stage	Tutorial: In the beginning, a device is offline (grey colour, an online device is blue). Also, a device will run by default RIPv2 broadcast. To change these settings, right click the device.
Device activation Automatic start of a device	Tutorial: Use the play and pause button to play/pause the simulation.
Simulation starts with at least two online devices Activation of a second device during simulation	To test if there is a route between two online devices, select them. (Attention: There may or may not be a route depending on the order of the selection).
'Help' in the design stage	Help: You are in the design phase of the network. In order to introduce a host/router, right click the blue panel. To create a connection, select two devices (green colour for selected devices). To deselect a device, click an empty place of the blue panel. To delete a connection, select again its own two devices. In order to move to the next phase, press the button 'Assign IP addresses'. You can delete a device by dragging it to the margins of the panel (red colour). It is forbidden to overlap devices. Devices in peril of overlapping will have yellow colour.
'Help' in the simulation stage	Help: In the beginning, a device is offline (grey colour, an online device is blue). Also, a device will run by default RIPv2 broadcast. To change these settings, right click the device. Use the play and pause button to play/pause the simulation. To test if there is a route between two online devices, select them. (Attention: There may or may not be a route depending on the order of the selection). It is forbidden to overlap devices. Devices in peril of overlapping will have yellow colour.

Table 1. Messages and their triggering events

All these messages can be re-shown by choosing the option Help from the Help menu. Note that this action will print in the console only the help messages characteristic to the current stage of the simulation.

2.3. Programming elements

2.3.1. New devices

The HypeRSimRIP program currently includes two types of devices: routers and hosts. Programmatically, they represent instances of the same class—Device. They are differentiated just by their local variable:

```
private string type;
```

that can have the value of '`router`' or '`host`'. This was done having in mind the possibility of introducing in the application of new network-related devices such as switches, hubs, etc. Thus, the possibility of an application update that will include a new device type will not necessitate the complete rewriting of all the code, but instead it will just require some minor add-ons, after the model of the existent devices—hosts and routers.

2.3.2. The problem of selecting a connection

In the HypeRSimRIP program, it is necessary to select a connection between two devices, in order to see its properties or in order to activate/deactivate it. A connection between two devices is represented in the graphical zone by a 2 px wide line between the devices in question, which are in turn represented by discs of radiuses between 5 px and 15 px (see Figure 3).

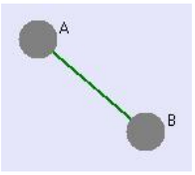

Figure 3. Connection example between the routers A and B.

Two devices (i.e. discs) cannot overlap, thus the length between two centres is at least 5 px + 5 px = 10 px (i.e. 5 px represents the minimum radius). Moreover, as it was previously seen, a selection cannot occur at less than 15 px of a device (i.e. 15 px represents the maximum device radius), in order to disallow the user the possibility of inserting overlapping devices.

In this problem, we not only know the coordinates of the device centres A and B, but also the coordinates of the selection point e.

In mathematics, to test if the point e is on the segment $[AB]$, it is first tested if A,e,B are collinear and if so, the second test verifies if e is between the segment end points.

In informatics, the problem that appears is that the plane (graphical zone) is discrete (it only has a finite set of points), the size of one point being of 1 px. Thus, it is almost impossible to select a point exactly on the $[AB]$ segment. To correct this problem, instead of verifying if the point e is on the segment, it must be checked if the point is near the segment, at maximum 5 px distance from the segment, which is 'somewhere' between points A and B and at least 20

px distance from these two points (5 px, because a selection inside the disc—the minimum length being 5 px—would represent the selection of the device and not of the connection, and another 15 px to compensate with the unselectable area around the device, as previously mentioned). By being 'somewhere' between points A and B, it is understood that the angles $\sphericalangle eAB$ and $\sphericalangle eBA$ must be acute angles (see Figure 4).

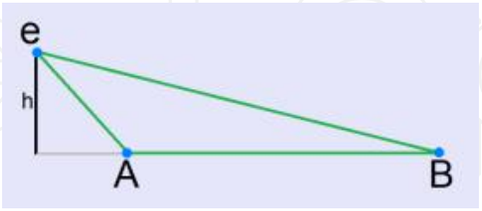

Figure 4. If $\sphericalangle eAB$ is obtuse, then clearly the selection was outside of the [AB] segment.

It can be easily proved that $\sphericalangle$AeB must be obtuse, by using the cosine theorem and the restrictions given by the problem. In a similar fashion, it can be proved that $\sphericalangle$AeB is obtuse, if and only if

$$AB^2 > Ae^2 + eB^2 \tag{1}$$

In order to calculate the distance h between e and the segment $[AB]$, by only knowing the coordinates of the A,e,B points, with simple surface formulae, it can be obtained that

$$h = \frac{\left| \begin{vmatrix} A.X & A.Y & 1 \\ B.X & B.Y & 1 \\ e.X & e.Y & 1 \end{vmatrix} \right|}{AB} \tag{2}$$

which gives the following evaluation statement:

```
Math.Abs(surface(e, A, B)) <= PointDistance(A, B)*5
```
(3)

where the surface method is defined by the Sarrus formula for the calculation of 3 × 3 determinants. Note that the division from the formula was not done in the code because when working with int-type variables (integer numbers), there are some precision issues in the event of division with remainder.

Thus, we obtain the following selection management algorithm:

http://dx.doi.org/10.5772/65415

Step 1. If the selection was made inside the disc of a device, the device will be selected.

Step 2. Else, if the selection was made at maximum 15 px around a device, the selection process will be cancelled.

Step 3. Else, if conditions (1) and (3) are satisfied, then it is considered that the connection between devices A and B was selected.

2.3.3. Device buffers, temporal acceleration and multi-threading

Each device has a series of buffers, which, during one second of the simulator, receive data packages coming from the directly connected devices. These data packages are processed at the end of the said second. These buffers, although existent even in real devices, have had their processing time increased from possibly several milliseconds to one full second, in order to allow a more clear observation of the changes that occur in the routing tables. As a consequence, although in reality, the routing tables of the routers of a mediocre size network reach converge in several milliseconds, here the convergence is slower. This duration can be modified quite quickly because all this temporal discussion really depends just on the interpretation of the trigger duration of an internal time counter. In the HypeRSimRIP program, it has already implemented a facility of time acceleration so that a second in reality can mean that only one second in the simulator has passed (the convergence being slow) or can be up to 30 sec in the simulator. This limit can be increased even higher, but keeping in mind the fact that we can simulate multiple accelerated devices on a single computer, the ratio 1:30 is to be considered as already large enough.

In terms of programming, these buffers are a series of list of objects that have a clear creation timestamp. These objects can be of the following types:

- RIPv1 Responses – instances of the class `ResponseV1`;
- RIPv2 Responses – instances of the class `ResponseV2`;
- RIP Requests – instances of the class `RequestFormat`.

At the end of each second, after the data packages transmissions, the simulator's second pass function will call the second pass function of every device, every time on another thread (hence multi-threading programming).

The second pass function of every device starts by processing every above-mentioned object, depending on their respective creation timestamp, as follows:

- If the creation time was 2 sec prior to the current time, then the object will be eliminated without processing, due to the fact that it is considered that the package was not processed on time and was therefore lost. This exception, following hundreds of simulations, has failed in occurring, but for safety reasons, it is still considered as a possibility.
- If the creation time was one second prior, then the object will be processed and afterwards, it will be eliminated from the buffer.
- If the creation time is the current time, then the object will be ignored, therefore it will be processed in the upcoming second.

- If the creation time is from a future point than the current time, the object will be ignored, and the user will be announced about this exception. Again, similar to the first exception, it has failed to occur yet.

Because each device processes its buffers on another thread, the main program will not wait for it to finish processing, as well all the other devices will not wait for each other. Thus, the execution order is non-deterministic, as in real life.

For didactical purposes, for a clearer observation of the simulation, the multi-threading facility can be disabled from the Settings windows of the program (see Figure 5).

2.4. Experimental facilities

The HypeRSimRIP program has a series of experimental facilities that allow modifications of some standard parameters of the RIP protocol. These include

- The possibility to change the value of infinity.
- The possibility to change the randomization interval of the automatic updates trigger period.

These changes can be done during the network design stage from the Settings window (see Figure 5).

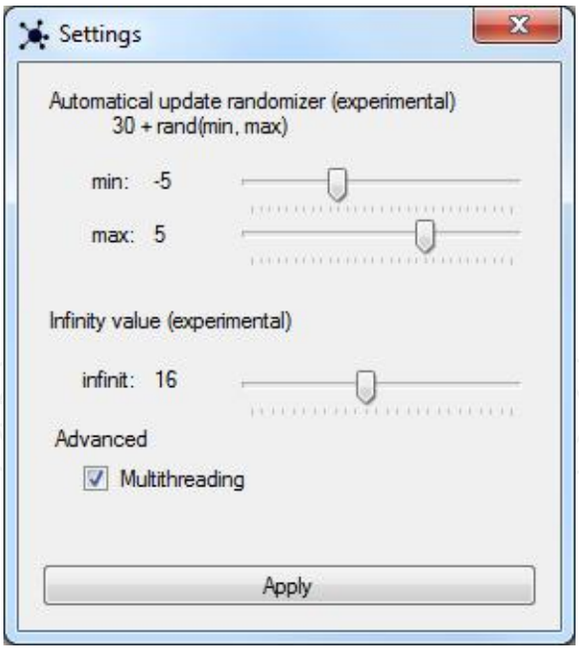

Figure 5. Settings window—Default-Start values.

For the infinite value change facility, the most suggestive examples are those with chain-linked routers. By default, the application comes with two such predefined examples: one with 20 chain-linked routers and another with 30 chain-linked routers.

For the facility in which you can change the randomization interval of the automatic updates trigger, the user can repeat rather interesting experiments, such as the one done by Floyd and Jacobson [17], which was the one that initially highlighted the need of a stochastic period. For this facility, an optimal experiment consists in the predefined network topology of 10 routers that generate a full graph. These experiments will be presented in detail in Section 3 of this paper.

2.5. Route highlight facility

The route highlight facility, characteristic to the HypeRSimRIP program, represents a highly useful tool int for the analysis of routing information. Its didactic scope is unmatched.

2.5.1. Usage

Once a network topology was designed in the program (see Figure 6), the IPv4 addresses were assigned (see Table 2) and the devices were activated (see Figure 7), all the devices start to transmit RIP datagrams and thus start to complete their individual routing tables.

Properties router 1st floor (00:00:00)
Name: 1st floor
Device type: router
Availability: offline
Local network
IPv4 network address: 192.168.0.64/28
IPv4 gateway address: 192.168.0.65/28
IPv4 broadcast address: 192.168.0.79/28
MAC gateway address: 70:11:16:91:AA:D9
Broadcast domain (BD) identifier: 5 external networks
External network towards central router
IPv4 network address: 192.168.0.80/28
IPv4 broadcast address: 192.168.0.95/28
IPv4 interface address: 192.168.0.82/28
Broadcast domain (BD) identifier: 6
RIP version: 2 broadcast

Table 2. Assigned parameters example—file automatically generated by the application

By selecting two online devices, the program will start to determine the route between them by using the routing data generated by each device at that given moment (see Figure 8). Note the fact that the utilized highlight algorithm is progressive, thus the topology or the routing information have time to change during the route tracing (see Figure 9).

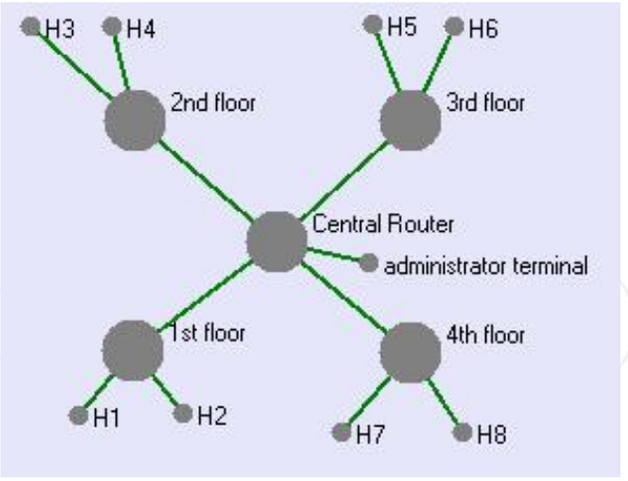

Figure 6. Network topology example.

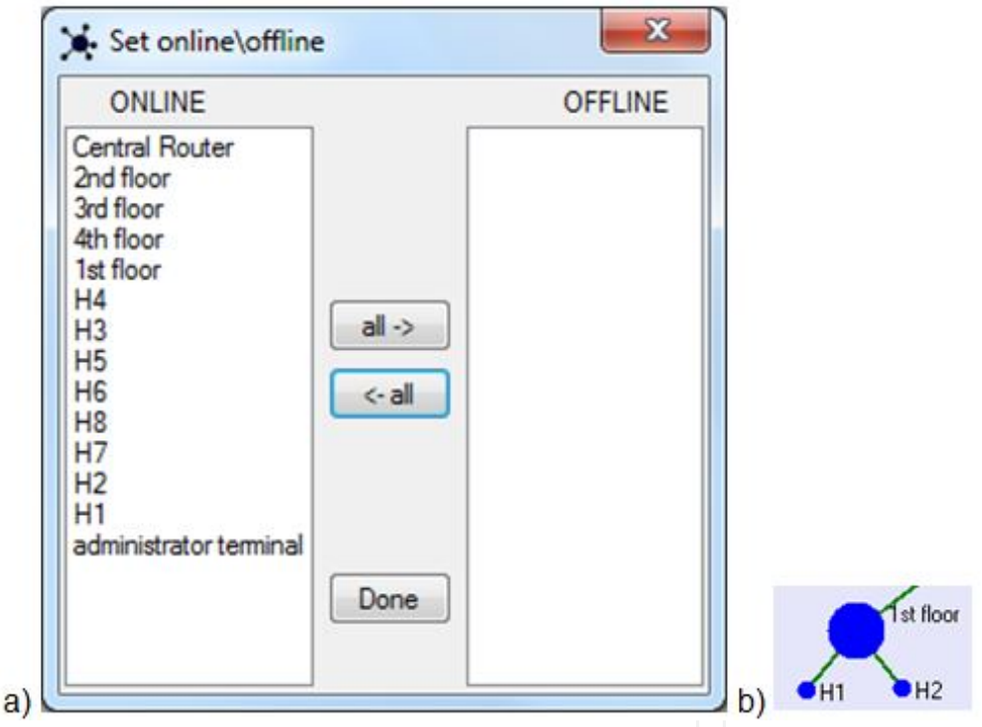

Figure 7. (a) Window that allows rapid multiple device activation/deactivation. (b) Online devices.

2.5.2. Programming

The programming of this tool relies on two main elements:

- The function

```
pinging(Device d1, Device d2): void
```

http://dx.doi.org/10.5772/65415

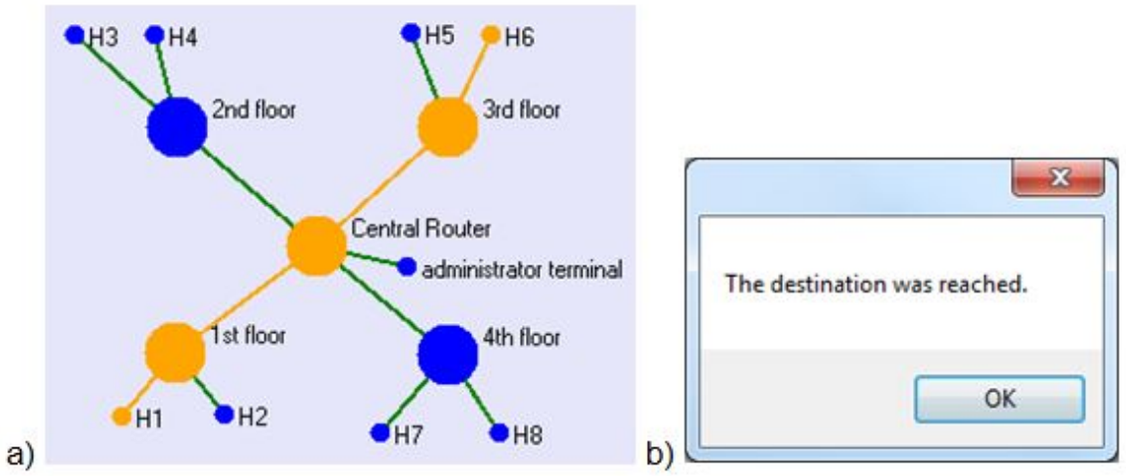

Figure 8. (a) The route from H1 to H6 is highlighted. (b) Destination reached alert message.

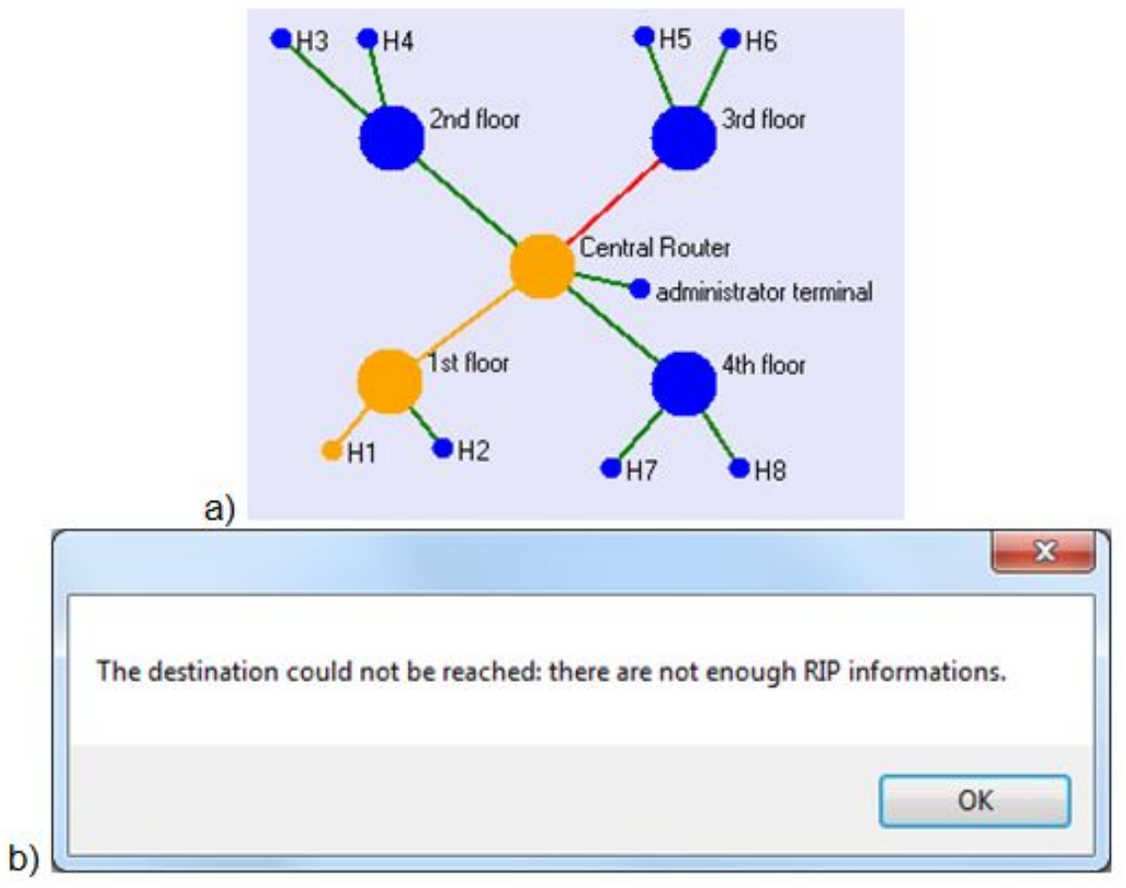

Figure 9. (a) In the event of the deactivation of the connection between the central router and the third floor, the route from H1 to H6 got blocked at the central router. (b) The message given by the program for this situation.

- The timer

```
timer_ping
```

The function `pinging` is called, once the program detects the selection of two devices (indifferent of the types: host-host, host-router, router-host or router-router). `d1` represents the first selected device and `d2` the latter, the order being of upmost importance because the transmission is unidirectional.

In the algorithm, two global variables `current_ping` and `end_ping` are initialized with the devices `d1` and `d2`, respectively. They represent the device in which the ping transmission has currently arrived and respectively the final destination of the ping transmission.

Furthermore, verification is in need to check if the sender is online; this is done in order to avoid having an offline device that is sending pings. If the result is positive, `current_ping` is highlighted and the `timer_ping` counter is started.

This functionality, as observed, has a distinct time counter, separated than that of the simulation time counter, in order to further simulate reality; in real life, there is not any dependence correlation between the two temporal measurements (the ping datagrams are different than the RIP datagrams). Didactically, the analysis of some well-defined situations, in which, for example, the update timers are frozen, is thus possible.

The `timer_ping` chronometer is set to 500 ms and after the `timer_ping.Start();` command the `'tick'` function

```
timer_ping_Tick(object sender, EventArgs e):void
```

will be periodically activated (every 500 ms).

At the beginning of this event management function, because of the complexity of the calculations, the timer is stopped and it will be restarted for the next step, after a new part of the route is highlighted.

The general form of a route is H-R-R-R-...-R-H, H-R-R-R-R-...-R, -R-R-R-...-R-H or R-R-R-...-R, where H means host and R means router. Thus, the following algorithm is:

Step 1. If `current_ping` is offline, then **STOP** – the device must have encountered a failure (i.e. went offline) during the tracing process.

Step 2. If `current_ping` is a host, then the existence of a connection between the current host and a router is checked. If the answer is negative, then the host must be isolated **STOP**. If the found connection is deactivated **STOP**. Otherwise, the found connection with its router is highlighted. If this router is in fact `end_ping`, the destination must have been thus reached, otherwise `timer_ping` is restarted.

The meaning of this step is that in order to discover another device, a host must first contact directly its connected router.

Step 3. In this moment of the algorithm, `current_ping` must be a router, but there still is a need to discover the nature of `end_ping` (host or router). If `end_ping` is a host, a local variable `ender_ping` is initialized with its directly connected router. If there is no such thing, then the final receiver is an isolated host **STOP**. If `end_ping` is a router, then `ender_ping` will be initialized with `end_ping` as a value.

This step has the following explication: the current router, by having the destination IP address, can calculate the IP address of the destination network, by doing so, it can calculate the transmission route via the routing table (as it will be seen in the next step).

Also, if `end_ping` was a host, it is important to check if `current_ping` is equal to `ender_ping`, which would mean that the destination is actually a host of the current router. In this eventuality, it remains to be verified if the host exists, if the connection is active and if the host is online, in which case the package is forwarded (i.e. the route and the host are highlighted and the user is announced that the destination was reached).

Step 4. At this point, only the route from `current_ping` to `ender_ping` needs to be determined. This route has the following general form: R-R-R-...-R, so the routing tables can finally be used. `current_ping` will search in his routing table the network IP address (calculated based upon the destination IP address) of `ender_ping`. If there is no such entry **STOP**. If it exists, from the entry, the transmission connection can be determined. If this connection is inactive **STOP**. Otherwise, the connection is highlighted. If the next hop (the next router) is offline **STOP**. Otherwise, the next hop is highlighted. `current_ping` is re-initialized with this next router and it will be verified if the destination was reached. If it was reached, the application will notify the user, and if not `timer_ping` is restarted.

3. Experimental results

3.1. Updates synchronization—classical experiment

Floyd Sally and Van Jacobson have made an experiment [17] in which they proved that without a randomization of the duration of the update counter, all the devices tended to synchronize their update periods, thus periodically overloading the network.

With the same goal, we can re-validate their experiment much faster, by using the HypeR-SimRIP application. Even though the concept of the application requires by default the randomization of the 30 sec update period with a random value between −5 and 5, the application allows the user to modify these limits.

In the HypeRSimRIP application, in order to deactivate the above-mentioned randomization, the user must set equal values for minimum and maximum randomization limits (preferably both values should be zero) in the Settings window (see Figure 10) before starting the simulation.

For the experiment, consider 10 routers, in the configuration given by Figure 11, with a full graph generated between them (connections between every router), generated by using the 'Full Graph Generation' options from the 'Diverse' menu of the application (see Figure 12).

After assigning the IPv4 addresses, all the routers are activated via the 'Set Online/Offline' window (Figure 7).

Commencing the simulation, devices start transmitting RIP datagrams and the application will start to show the network usage (see Figure 13). Periodically, huge network usage explosions

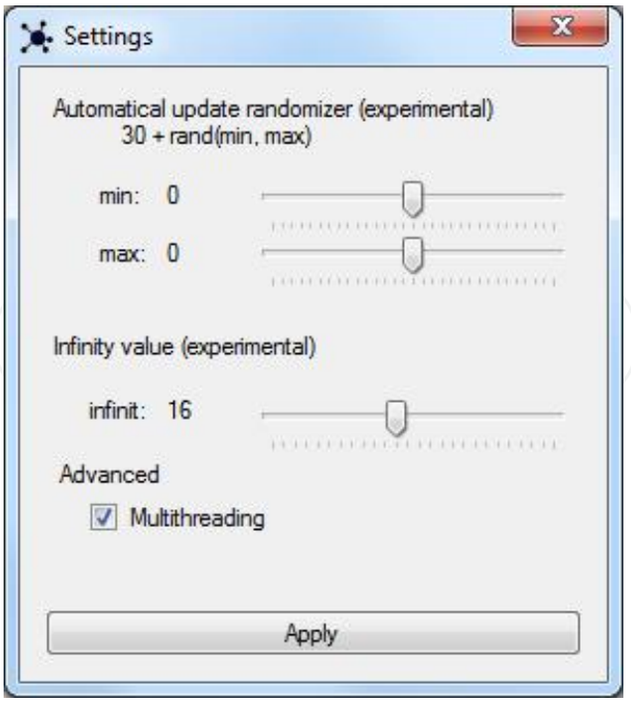

Figure 10. Settings window—randomization deactivation.

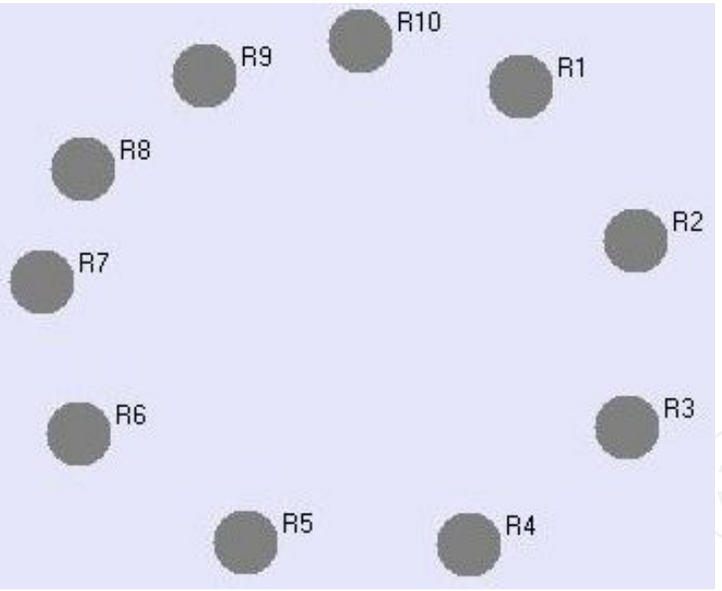

Figure 11. Ten router topology.

are observed: at first 484.496 bps, then periodically (every 30 sec) 171.360 bps. By using the temporal acceleration facility, it can be observed that this tendency will remain throughout the simulation of this experiment.

By repeating the experiment, with the randomization margin between −5 and 5, the network usage from Figure 14 was obtained.

http://dx.doi.org/10.5772/65415

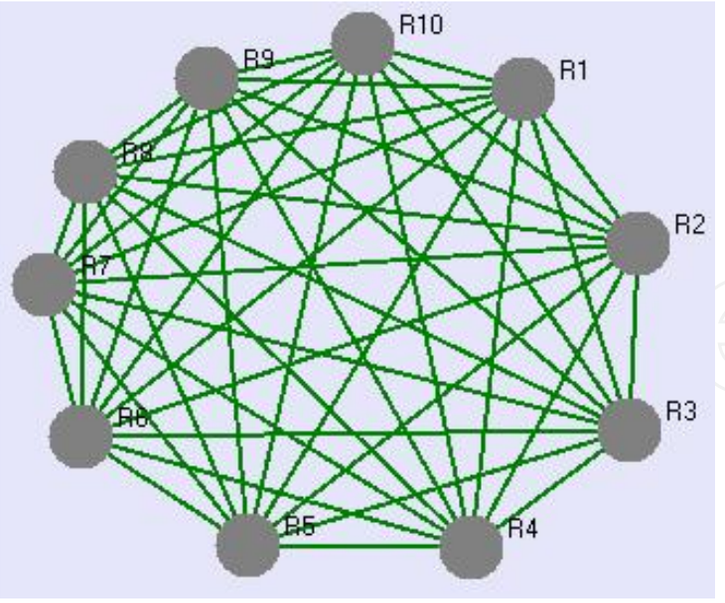

Figure 12. Topology determined by a full graph between 10 routers.

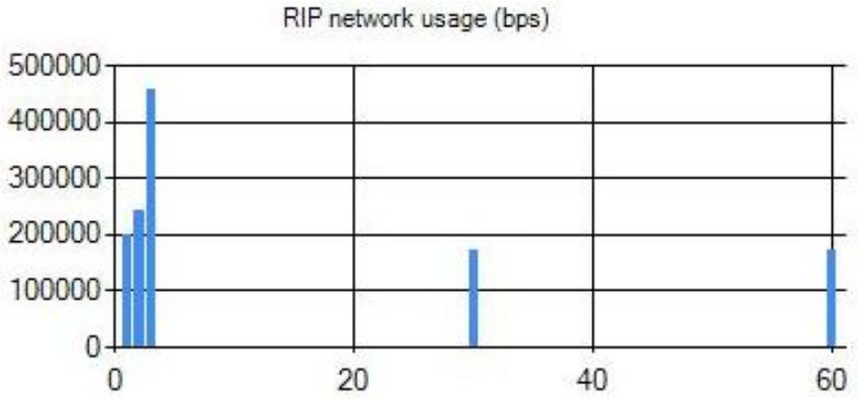

Figure 13. RIP network usage without randomization (first minute).

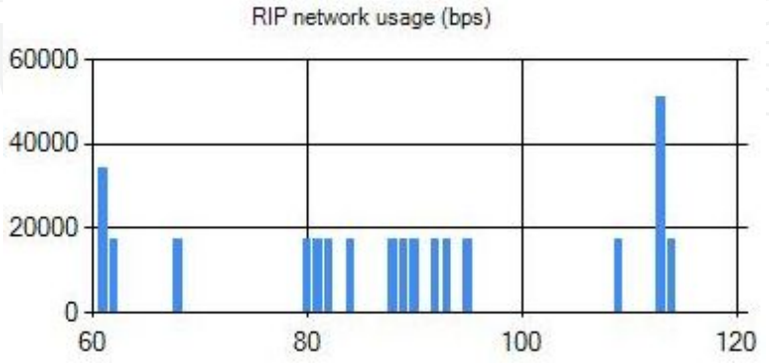

Figure 14. RIP network usage with randomization (second minute).

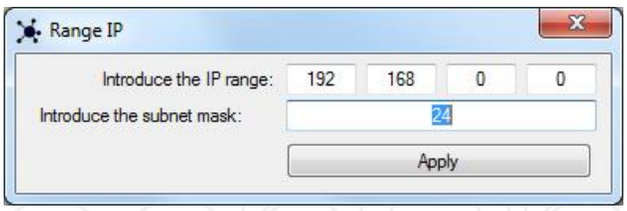

Figure 15. IP range window.

Most of the values during the update periods remained at 17.136 bps (10 times less than the minimum usage boom from the without randomization simulation), and in just a few exceptions, the network usage reached 51.408 bps (still half the minimum usage boom from the previous simulation).

In conclusion, because the transmissions are distributed, the network is not anymore disrupted significantly by the RIP datagrams, thus avoiding a possible network overload.

3.2. RIPv1-RIPv2 compatibility—practical example

Devices that implement RIP version 2 can either broadcast datagrams or multi-cast them. In the broadcast situation, devices that implement RIP version 1 can intercept transmitted messages and even respond to them, but because they were not configured to intercept multi-cast messages, they will not interact with multi-cast transmitted datagrams [14].

In order to utilize and highlight the above information, the following hypothetical situation will be presented and analysed using the HypeRSimRIP application:

A medium-sized company decides to hire a network administrator to create and ensure the maintenance of a network system inside their building. Assuming that the building has four floors and that on each floor there are two terminals (computers-hosts).

To ensure a good connectivity, the administrator will design a star-type topology (see Figure 6); on each floor, he will install a router, to which the respective two terminals will connect, and the four routers will inter-connect through a fifth router, which will provide the external network (e.g. Internet) connection. Obviously, to ensure full control of the corporate network, the administration terminal will be connected to this central router.

Thus, having designed the network, the administrator will continue by assigning the IP addresses provided by the Internet Service Provider (ISP), based on the sub-net mask and the requirements of the physical topology—suppose that the address is 192.168.0.0 with the standard mask /24 (255.255.255.0) (see Figure 15).

According to the VLSM technique of the HypeRSimRIP application, the administrator can now configure all routers with their corresponding parameters (static IPv4 addressing, for better security) (see Table 2). Finally, the network becomes operational (see Figure 16).

http://dx.doi.org/10.5772/65415

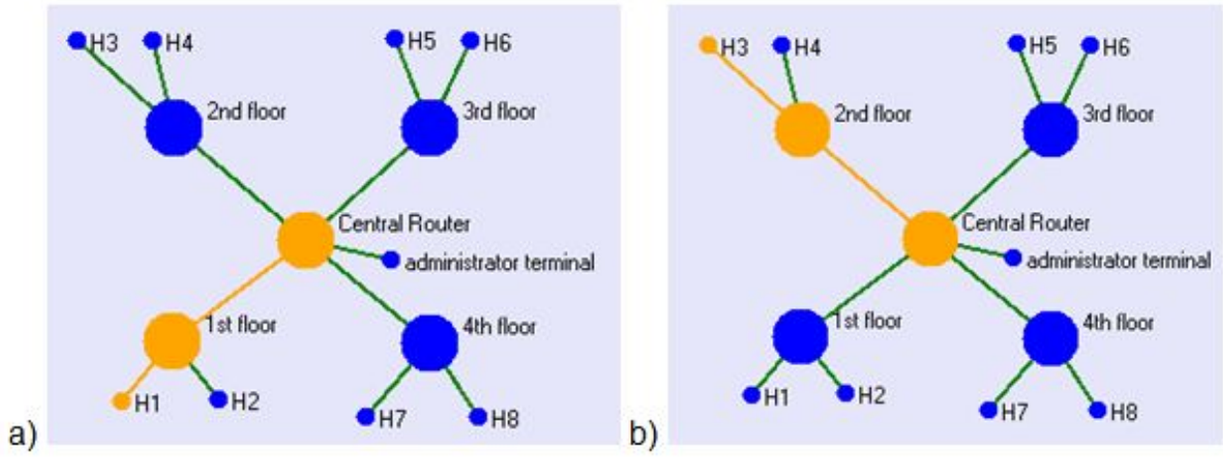

Figure 17. Highlighted routes: (a) from H1 to the central router and (b) from H3 to the central router.

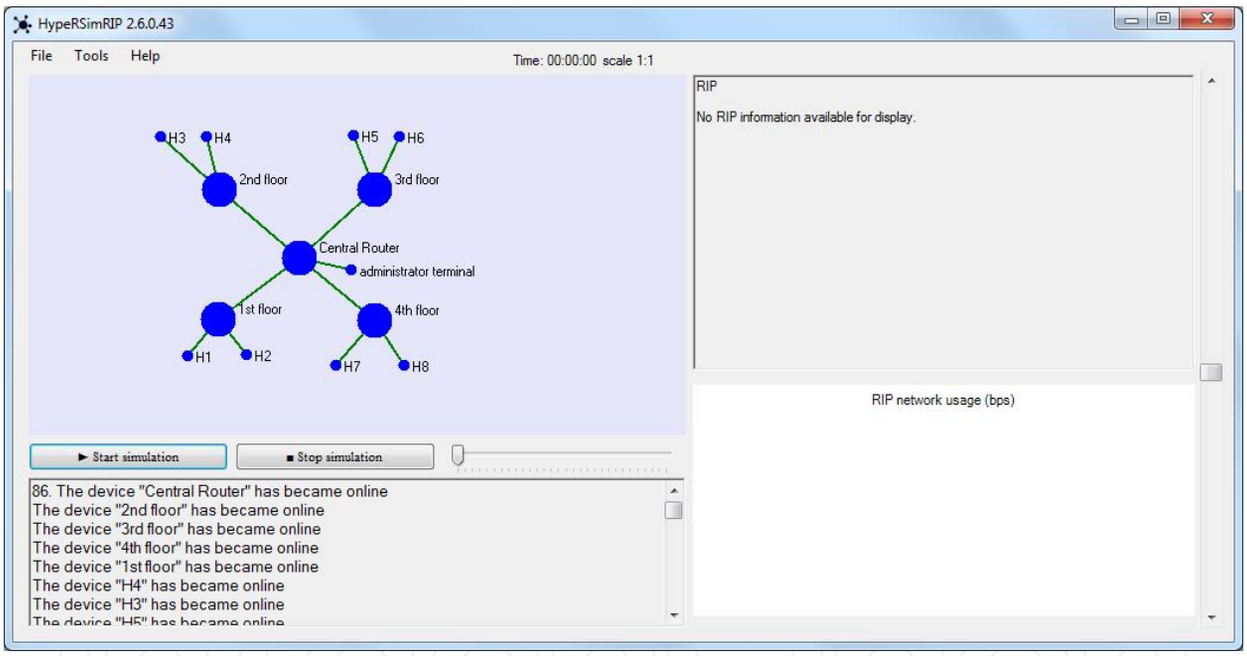

Figure 16. Online network.

Now, the administrator's design choice becomes obvious; any malicious information, any informatics attack that comes from the exterior, from the supernet, will be first intercepted by the administrator and dealt with. By using the route highlight facility of the program, the above-mentioned can be proven (see Figure 17).

By looking at Figure 18, it can be observed that any data package transmitted between two different floors will be routed through the central router. Thus, the administrator will have direct access to all the data transmitted inside the building. Such a perspective may have undesired consequences, such as the creation of direct routes between directly connected floors (to bypass the central router), which are undesired by the administration (see Figure 19).

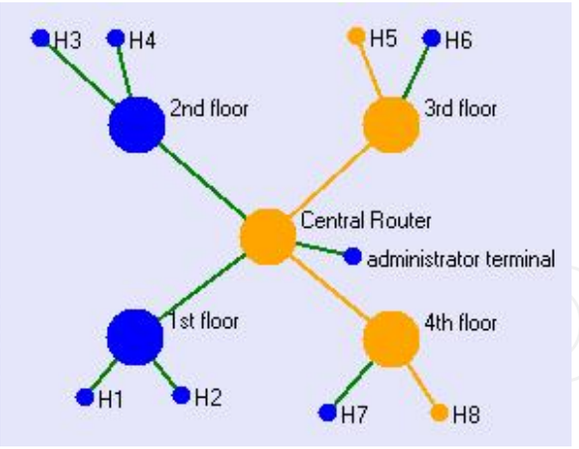

Figure 18. The route from H8 to H5 passes through the central router.

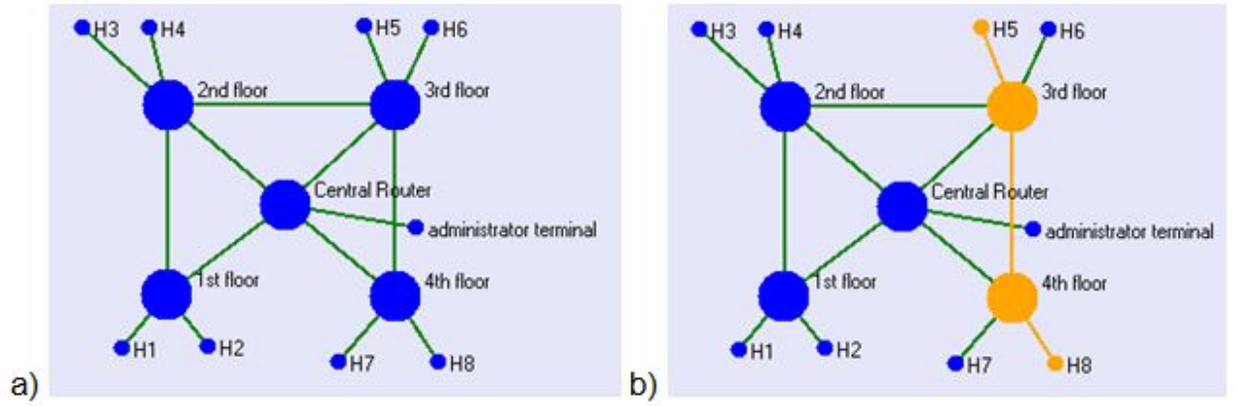

Figure 19. (a) The network topology after the undesired changes. (b) The route from H8 to H5 now bypasses the central router.

This problem can be immediately repaired without major efforts: the administrator can set the central router to use RIPv2 broadcast version (for backwards compatibility), and the four floor routers to alternate between RIPv1 and RIPv2 multi-cast: the routers on an odd floor to RIPv1 and the routers on even floors to utilize RIPv2 multi-cast.

In the HypeRSimRIP application, this can be done by using the 'RIP version switch' window (Figure 20), accessible from the context menu of every device. Once all the new settings have been introduced, it can be observed (Figure 21) that even though the illegal routes are still active, they have been avoided.

This example has thus presented a practical use of two Internal Gateway Protocols (IGPs) inside the same network and the interaction between the three RIP switches. Moreover, the following statement: 'A network system with multiple incompatible IGPs can have a full routing table, if and only if the transitioning entities can translate the routing information' [16] has been proven true, and also been given an example: here the transitioning entity is the central router.

Figure 20. The RIP version switch window.

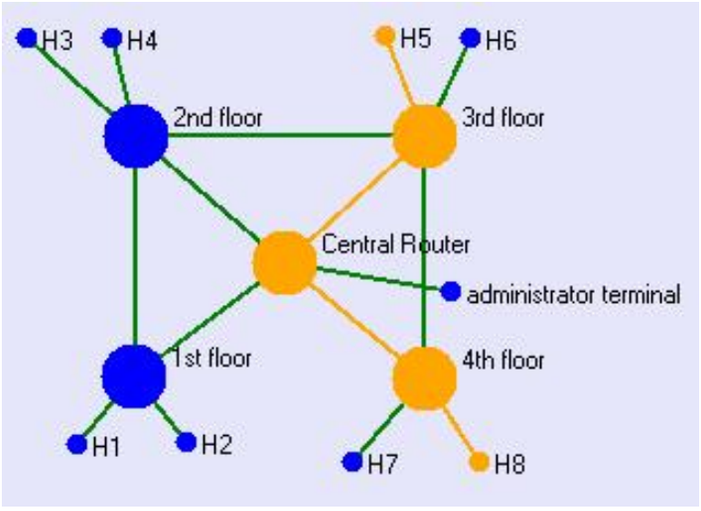

Figure 21. The route from H8 to H5 detours through the central router.

3.3. 'Changing infinity'—didactical experiment

Because the metric chosen for implementation by the RIP protocol is the hop count metric, and because for representing infinity, the value chosen is 16, the following drawback appears: If there exists a route longer than 15 hops, the protocol will not consider it (RIP uses the numeric value of 16 to mark unreachable devices) [13].

In this next experiment, a network system with routes of length 16 or bigger will be implemented, with the goal of highlighting the drawback. Assume a network system composed of 20 chain-linked routers as shown in Figure 22.

Starting the simulation, the network will converge after 18 sec. To illustrate the drawback, in Table 3, the routing table of the R1 router is presented.

The absence of connections with routers R17, R18, R19 and R20 can be observed. This takes place because they are situated at the distances of 16, 17, 18 and 19 hops, respectively, from R1.

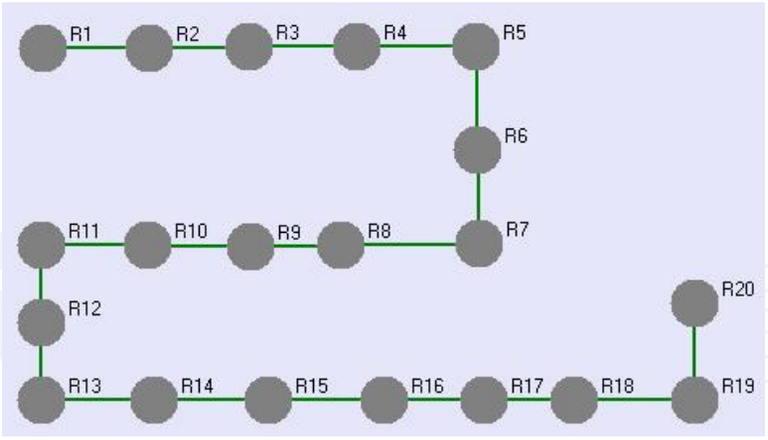

Figure 22. Network topology example in which the routing tables will be incomplete.

RIP Table - R1: 192.168.0.0/30 (Update in 6 sec.)
-> Connection with R2 via R2 at distance 1 (timeout:174)
-> Connection with R3 via R2 at distance 2 (timeout:174)
-> Connection with R4 via R2 at distance 3 (timeout:174)
-> Connection with R5 via R2 at distance 4 (timeout:174)
-> Connection with R6 via R2 at distance 5 (timeout:174)
-> Connection with R7 via R2 at distance 6 (timeout:174)
-> Connection with R8 via R2 at distance 7 (timeout:174)
-> Connection with R9 via R2 at distance 8 (timeout:174)
-> Connection with R10 via R2 at distance 9 (timeout:174)
-> Connection with R11 via R2 at distance 10 (timeout:174)
-> Connection with R12 via R2 at distance 11 (timeout:174)
-> Connection with R13 via R2 at distance 12 (timeout:174)
-> Connection with R14 via R2 at distance 13 (timeout:174)
-> Connection with R15 via R2 at distance 14 (timeout:174)
-> Connection with R16 via R2 at distance 15 (timeout:174)

Table 3. R1's routing table, after the convergence of the network—table automatically generated by the application

Although the initial traffic generated peaked at 117600 bps, it will stabilize at around 14016 bps in the absence of network topology changes and in the absence of some eventual (although improbable) synchronizations of automatic RIP updates (see Figure 23).

An interesting fact about this example is the initial form of the network usage chart—for the situation in which all routers have become simultaneously online. The reason for which the

http://dx.doi.org/10.5772/65415

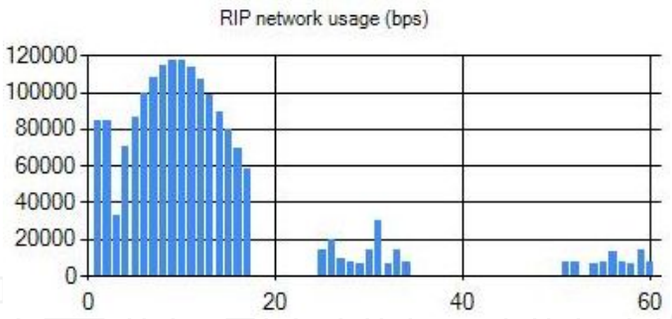

Figure 23. RIP network usage (first minute).

chart resembles a Gaussian bell can be explained mathematically: every router discovers one by one a new entity and thus forces RIP updates towards both its neighbours: Ri discovers Ri −1 and Ri+1, thus both Ri−1 and Ri+1 discover new routes so they each launch another two messages to their neighbours, and so on. An exponential ramification tendency is thus observed, but as soon as R1 and R20 are reached, they each send only one update, which will be ignored, because R2 and R19 were the gateways of all routes of R1 and R20, respectively, and so, because of the reverse poisoning [13], every route in the update will have a metric of 16. This transition from sending two messages simultaneously to only 1 creates a flattening effect on the chart—the chart had initially an exponential growth, and finally a quasi-linear drop.

The HypeRSimRIP application allows, for experimental purposes, to change infinity's numerical value. If, for example, it changed from 16 to 20, then all the routers could communicate between each other (Figure 24), but if it decreased to 5, a drastic decrease in the sizes of the routing tables can be observed (Table 4).

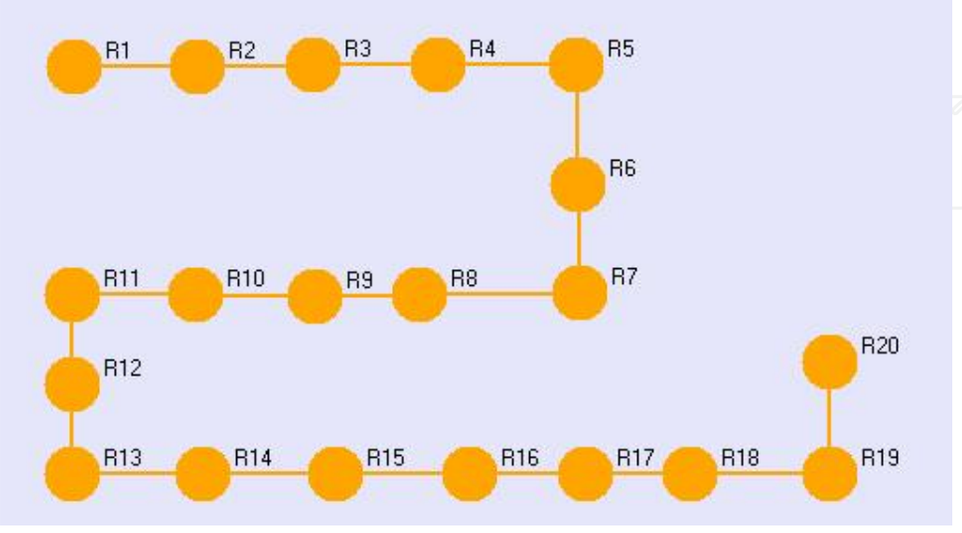

Figure 24. The route from R1 to R20 (infinity is represented through 20).

RIP Table - R1: 192.168.0.0/30 (update in 21 sec)
-> Connection with R2 via R2 at distance 1 (timeout:179)
-> Connection with R3 via R2 at distance 2 (timeout:179)
-> Connection with R4 via R2 at distance 3 (timeout:179)
-> Connection with R5 via R2 at distance 4 (timeout:179)
RIP Table - R2: 192.168.0.4/30 (update in 25 sec)
-> Connection with R1 via R1 at distance 1 (timeout:179)
-> Connection with R3 via R3 at distance 1 (timeout:179)
-> Connection with R4 via R3 at distance 2 (timeout:179)
-> Connection with R5 via R3 at distance 3 (timeout:179)
-> Connection with R6 via R3 at distance 4 (timeout:179)

Table 4. The routing tables of R1 and R2, after the convergence of the network (infinity is represented through 5)

4. Conclusions

The core of this article is represented by the development of the HypeRSimRIP application. HypeRSimRIP is a type 2 network hypervisor that allows the user to construct and manage an IP network in real time. It provides functional capabilities such as IPv4 addressing through sub-netting and VLSM, configuration of RIPv1 and RIPv2 protocols, connectivity test, etc. Moreover, the application implements several original educational facilities, useful for teaching network-related concepts: infinity value change, route highlighting. It is intended to also include in the application IPv6 addressing and a RIPng [16] implementation. Some of the personal contributions brought in this program are generation and display of IP datagrams, RIP datagrams, routing tables, charts of the network usage, ping and tracert-type commands, multi-threading (for a more realistic simulation), temporal acceleration, etc. Noteworthy are also the applications' experimental facilities, which allow the user to quickly recreate famous experiments such as the Floyd-Jacobson experiment [17].

Author details

Virgilius-Aurelian Minuță[1*] and Eugen Petac[2]

*Address all correspondence to: aurelmva@gmail.com

1 Babeș-Bolyai University, Cluj-Napoca, Department of Mathematics and Computer Science, Romania

2 'Ovidius' University of Constanța, Constanța, Department of Mathematics and Computer Science, Romania

References

[1] Baldwin J.M., editor. Dictionary of Philosophy and Psychology, Vol. II. London, UK: MacMillan and Co. Ltd.; 1902.

[2] Cambridge Advanced Learner's Dictionary. 3rd ed. Cambridge, UK: Cambridge University Press; 2008.

[3] Heim M. The Metaphysics of Virtual Reality. Oxford, UK: Oxford University Press; 1994.

[4] Graziano C.D. A Performance Analysis of Xen and KVM Hypervisors for Hosting the Xen Worlds Project [dissertation]. Ames, USA: Iowa State University; 2011.

[5] Popek J.P., Goldberg R.P. Formal requirements for virtualizable third generation architectures. Magazine Communications of the ACM. 1974;17(7):412-421

[6] VMware Inc. ESXi Hypervisor [Internet]. Available from: http://www.vmware.com/products/vsphere/features/esxi-hypervisor.html [Accessed: June 2014]

[7] VMware Inc. Understanding Full Virtualization, Paravirtualization, and Hardware Assist [Internet]. 15 October 2007. Available from: http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf [Accessed: April 2016]

[8] TeamViewer GmbH. Our History [Internet]. Available from: https://www.teamviewer.com/en/company/ [Accessed: April 2016]

[9] Chernicoff D. HP VDI Moves to Center Stage [Internet]. Available from: http://www.zdnet.com/blog/datacenter/hp-vdi-moves-to-center-stage/978 [Accessed: June 2014]

[10] Microsoft Corporation. Strategies for Embracing Consumerization [Internet]. Available from: http://download.microsoft.com/download/E/F/5/EF5F8B95-5E27-4CDB-860F-F982E5B714B0/Strategies%20for%20Embracing%20Consumerization.pdf [Accessed: June 2014]

[11] Intrinsic Technology. HVD: The Cloud's Silver Lining [Internet]. Available from: http://www.intrinsictechnology.co.uk/FileUploads/HVD_Whitepaper.pdf [Accessed: June 2014]

[12] Petac E., Musat B. Computer Networks. Experimental Studies. Constanta, Romania: Crizon; 2012.

[13] Hedrick C. Routing information protocol. Request for Comments. 1988;(1058):1-33

[14] Malkin G. RIP Version 2 – Carrying additional information. Request for Comments. 1994;(1723):1-9

[15] Malkin G. RIP Version 2. Internet Standard. 1998;(56):1-39

[16] Malkin G., Minnear I.R. RIPng for IPv6. Request for Comments. 1997;(2080):1-19

[17] Floyd S., Jacobson V. The synchronization of periodic routing messages. IEEE/ACM Transactions on Networking. 1994;2(2):122–136.

IntechOpen