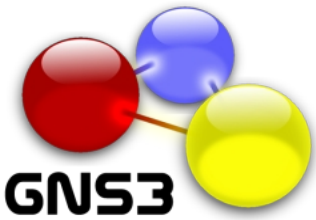


GNS3 Documentation

v0.3.2 beta

Parts of this tutorial are taken from the excellent Dynagen tutorial by Greg Anuzelli.

Introduction

GNS3 is a graphical network simulator that allows you to easily design network topologies and then run simulations on them. At the moment GNS3 supports IOS routers, ATM/Frame Relay/Ethernet switches and hubs. You can even extend your own network by connecting it to your virtual topology.

To do his magic, GNS3 is based on Dynamips and in part on Dynagen, it has been developed in python and through PyQt the GUI part is made with the powerful Qt library, famous for its use in the KDE project. GNS3 also uses the SVG technology (Scalable Vector Graphics) to provide high quality symbols for designing your network topologies.

About Dynamips

Dynamips is a Cisco router emulator written by Christophe Fillot. It emulates 1700, 2600, 3600, 3700, and 7200 hardware platforms, and runs standard IOS images. In Chris' own words:

> *This kind of emulator would be useful to:*
>
> - *Be used as a training platform, with software used in the real world. It would allow people to become more familiar with Cisco devices, Cisco being the world leader in networking technologies;*
> - *Test and experiment features of Cisco IOS;*
> - *Check quickly configurations to be deployed later on real routers.*
>
> *Of course, this emulator cannot replace a real router, it is simply a complementary tool to real labs for administrators of Cisco networks or people wanting to pass their CCNA/CCNP/CCIE exams.*

Although Dynamips provides a simple virtual switch, it does not emulate Catalyst switches (although it does emulate the NM-16ESW).

About Dynagen

Dynagen is a text-based front end for Dynamips written by Greg Anuzelli which provide a separate OOP API used by GNS3 for interfacing with Dynamips. GNS3 also uses the

Dynagen INI-like configuration file format and integrates Dynagen's management CLI which allows users to list devices, suspend and reload instances, determine and manage idle-pc values, perform packet captures, and more.

If somehow you have stumbled upon this tutorial without first finding the GNS3, Dynamips or Dynagen web sites, here they are along with some other important links:

GNS3: http://www.gns3.net/

Dynamips: http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator

Dynamips Blog (where most of the action is): http://www.ipflow.utc.fr/blog/

Dynagen (a text-based front-end to the emulator): http://dyna-gen.sourceforge.net/

GNS3 / Dynamips / Dynagen Bug tracking: http://www.ipflow.utc.fr/bts/

Hacki's Dynamips / Dynagen / GNS3 Forum: http://7200emu.hacki.at/index.php

Installing GNS3

GNS3 runs on Windows, Linux and Mac OS X (other platforms not tested) and requires the following dependencies to be installed if you want to use it from a source archive:

- Qt >= 4.3, available on http://trolltech.com/developer/downloads/qt/index
- Python >= 2.4, available on http://www.python.org
- Sip >= 4.5 if you need to compile PyQt, available on http://www.riverbankcomputing.co.uk/sip
- PyQt >= 4.1 available on http://www.riverbankcomputing.co.uk/pyqt

We have also put together an all-in-one Windows installer package that includes Winpcap, Dynamips, and a compiled version of GNS3, eliminating the need to install Python, PyQt and Qt. It also provides Explorer "integration" so you can double-click on network files in order to run them.

Windows users should install the all-in-one Windows installer package. This provides everything you need to run GNS3 on local or remote machines, except an IOS image (see the next section).

Linux users should download Dynamips and extract it to a suitable location. Install the GNS3 dependencies and then run GNS3. Users can also try our binary version for Linux, eliminating the need to install Python, PyQt and Qt.

Note: If you are running Dynamips on a RedHat or Fedora system, take a look at Dynamips FAQ item #2 if you are experiencing segfaults when you try to run Dynamips.

At the moment, Mac OS X users have to manually compile the dependencies. A binary version should be released soon.

IOS Images

Dynamips runs real Cisco IOS images. From the Dynamips FAQ:

> *Can you provide a Cisco IOS image for a 7200 to me?*
>
> *No, I am not allowed to distribute IOS images. You will have to find one by yourself. This should not be a problem if you are a Cisco customer.*

On Windows, drop the image in `C:\Program Files\Dynamips\images`. You can actually drop the images anywhere you want, but the sample labs are configured to look here. On Linux/Unix systems, designate a location to store your images and drop them there (I like to use /opt/images, but it's your system.)

Cisco IOS images are compressed. These compressed images will work just fine with Dynamips*, however the boot process is slowed significantly by this decompression process (just like on real routers). It is recommended that you decompress the images beforehand, so the emulator doesn't have to. You can do this with the "unzip" utility on Linux/Unix/Cygwin as follows:

```
unzip -p c7200-g6ik8s-mz.124-2.T1.bin > c7200-g6ik8s-
mz.124-2.T1.image
```

You will receive a warning from unzip, which you can safely ignore. On Windows you can use WinRAR to uncompress images. You may download a free copy of WinRAR at http://www.winrar.com.

Note that currently images for 2600 routers must be uncompressed to work with Dynamips.

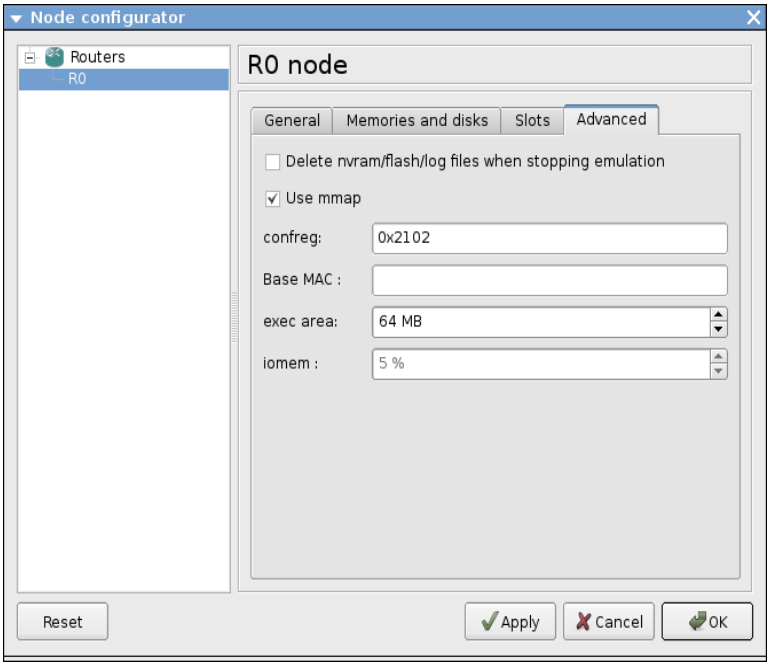
Please, always test your IOS image directly with Dynamips before using it in GNS3:

```
./Dynamips -P <chassis> <path-to-the-ios-image>
```

Resource Utilization

Dynamips uses a fair amount of RAM and CPU in order to accomplish its emulation magic. If you intend to run an IOS image that requires 256 MB of RAM on a real 7200 router, and you devote 256 MB of RAM to your virtual router instance, it will allocate 256 MB of working set memory. Dynamips also allocates (by default) 64 MB of RAM per instance on Unix systems (16 MB on Windows systems) to cache JIT translations. This will be the total working set size; by default the amount of your system's actual RAM used will typically be significantly less. This is because, by default, Dynamips uses memory mapped files for the routers' virtual memory. In the working directory you will see temporary "ram" files equal to the size of the virtual routers' RAM size. Your OS will naturally cache in RAM the sections of the mmap files that are being used. (See the Memory Usage Optimizations section for configuration options that can significantly reduce memory utilization).

If you have plenty of RAM, and you know what you are doing, uncheck "Use mmap" in the IOS router advanced settings. Right-click a device and choose Configure:

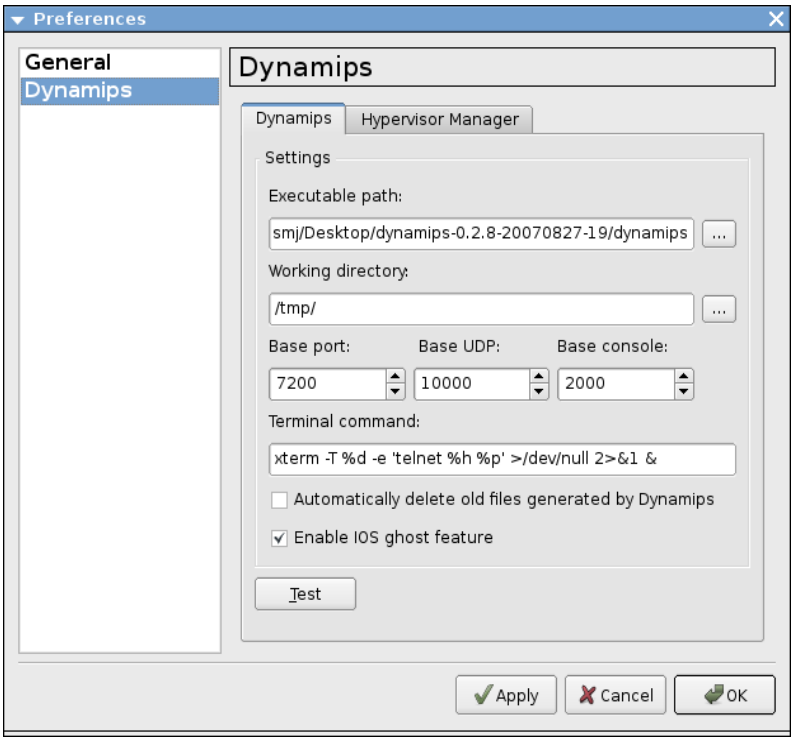


Dynamips also uses a lot of CPU power, because it is emulating a router's CPU instruction-by-instruction. It initially has no way of knowing when the virtual router's CPU

is idle, so it dutifully executes all the instructions that make up the IOS's idle routines - just as it would execute the instructions that perform "real" work. But once you have run through the "Idle-PC" process for a given IOS image, CPU utilization decreases drastically. More on this very important topic later.

Configuring Dynamips Preferences

To use Dynamips in GNS3, you must configure the path to it and the base port. These settings will be used by the Hypervisor Manager and to load .net files. Go to Preferences on the Edit menu:



The working directory is where all the files generated by Dynamips are stored, these include the NVRAM for the virtual router, as well as the bootflash, logfiles, and some other working files

Options:

- "Automatically delete old files generated by Dynamips" will delete old files like routers' nvrams (if you used mmap) etc... when returning to design mode.
- "Enable IOS ghost feature" is to globally use (or not) the Dynamips's ghost feature (see Memory Usage Optimizations section for more details)

To allow you to connect to the virtual router consoles, you also must configure the terminal command.

GNS3 will propose you a default command but you can set a custom one.

The following substitutions are performed:

%h = host

%p = port

%d = device name

Terminal Command Examples

- On Windows:

TerraTerm SSH users: `C:\progra~1\TTERMPRO\ttssh.exe %h %p /W=%d /T=1`

PuTTY users: `start C:\progra~1\PuTTY\putty.exe -telnet %h %p`

SecureCRT users: `start C:\progra~1\SecureCRT\SecureCRT.EXE /script c:\progra~1\dynamips\securecrt.vbs /arg %d /T /telnet %h %p & sleep 1`

- On Linux:

Gnome users: `gnome-terminal -t " + name + " -e 'telnet " + host + " " + str(port) + "' > /dev/null 2>&1 &`

- On Mac OS X:

With Terminal: `/usr/bin/osascript -e 'tell application "Terminal" to do script with command "telnet %h %p ; exit"' -e 'tell application "Terminal" to tell window 1 to set custom title to "%d"'`

With iTerm with named tabs: /`usr/bin/osascript -e 'tell app "iTerm"' -e 'activate' -e 'set myterm to the first terminal' -e 'tell myterm' -e 'set mysession to (make new session at the end of sessions)' -e 'tell mysession' -e 'exec command "telnet %h %p"' -e 'set name to "%d"' -e 'end tell' -e 'end tell' -e 'end tell'`

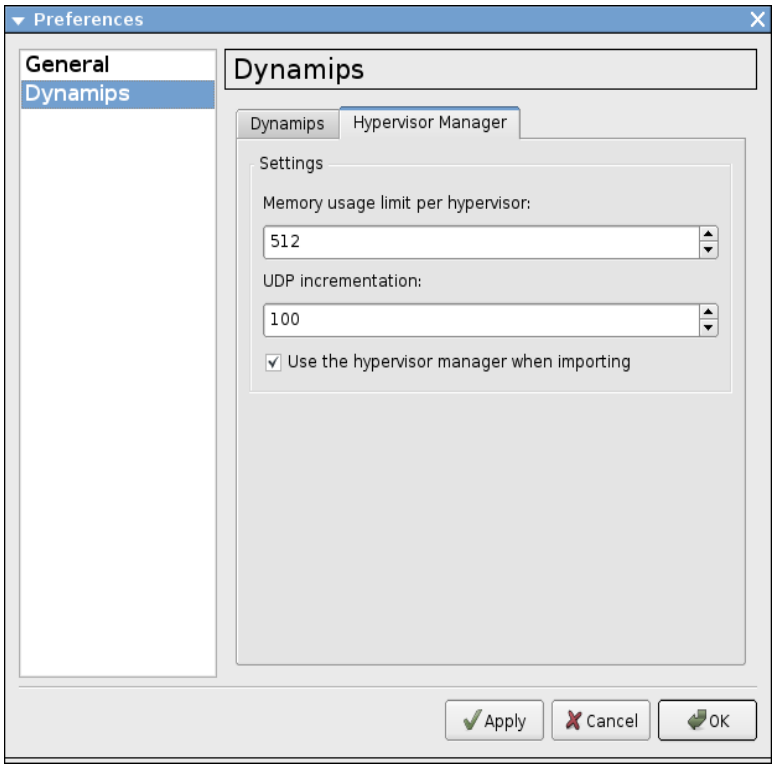
The Hypervisor Manager is used to run your hypervisors internally in GNS3, which means you don't need to start them manually. This manager also helps to address the memory usage limit per process problem when running many IOS instances on a single hypervisor (see FAQ to learn more about this issue) by “load-balancing” instances on multiple hypervisors.

Let's look at an example of how it works:

We want to run 5 IOS instances which uses 256 MB each and we configured the memory usage limit per hypervisor setting to 512 MB. When we start the lab, the hypervisor manager will create 3 hypervisor processes based on the following formula (you must round to the next highest whole number):

number of hypervisors = (256 * 5 / 512)

The Hypervisor Manager assigns the first 2 instances to the first hypervisor, the next 2 to the second hypervisor, and the last instance on the third hypervisor.



There are two other settings in the Dynamips preferences. “UDP incrementation” tells the hypervisor manager how to increment the Dynamips base udp port for each hypervisor process created (so if the base udp port is 10000 in Dynamips preferences and the incrementation is 100, then for 3 hypervisors this will set the base udp to 10000 for the first hypervisor, 10100 for the second and so on).

Note: more information on UDP issues can be found in the “Client / Server and Multi-server Operation” section.

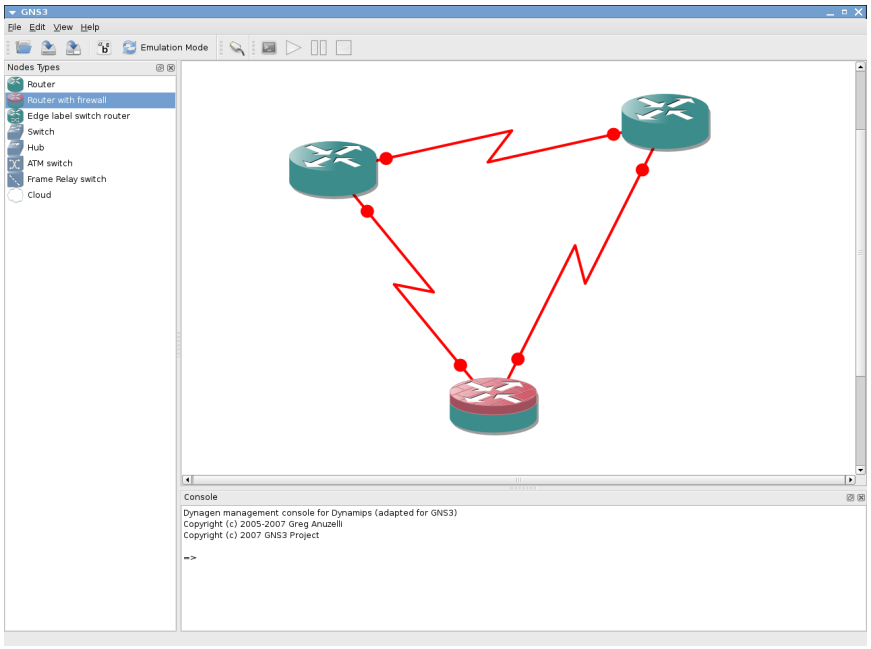
The option “Use the hypervisor manager when importing” is used when loading a topology file (.net) in GNS3. If this option is checked and in the .net file you have defined hypervisors to run on localhost, then GNS3 considers that those hypervisors need to be started by the Hypervisor Manager. If not checked, those hypervisors are recorded as external hypervisors and must be manually started.

Running a simple lab

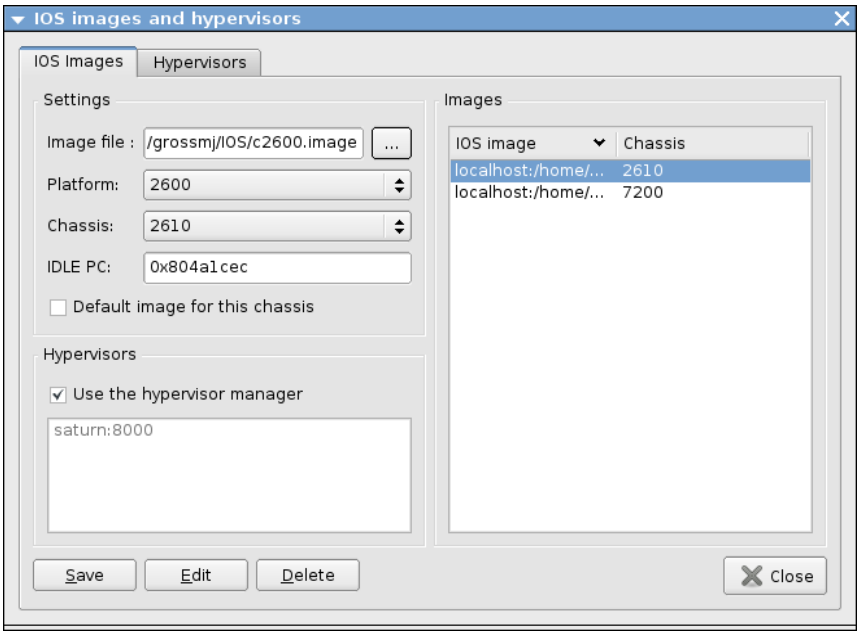
In this section we are going to guide you on how to run a three routers lab step-by-step.

The design mode

By default, GNS3 starts in “design mode”. In this mode you can create your network topology by drag 'n' dropping nodes from the list on the left to the scene on the right.



Next, record IOS images by selecting Edit -> IOS images from the menu (or pressing CTRL + SHIFT + I). Then set the path to an IOS image, choose the plaform and the chassis (if applicable) and input an IDLE PC value. By default, you are using the integrated hypervisor (the dynamips managed by GNS3) to run your IOS.



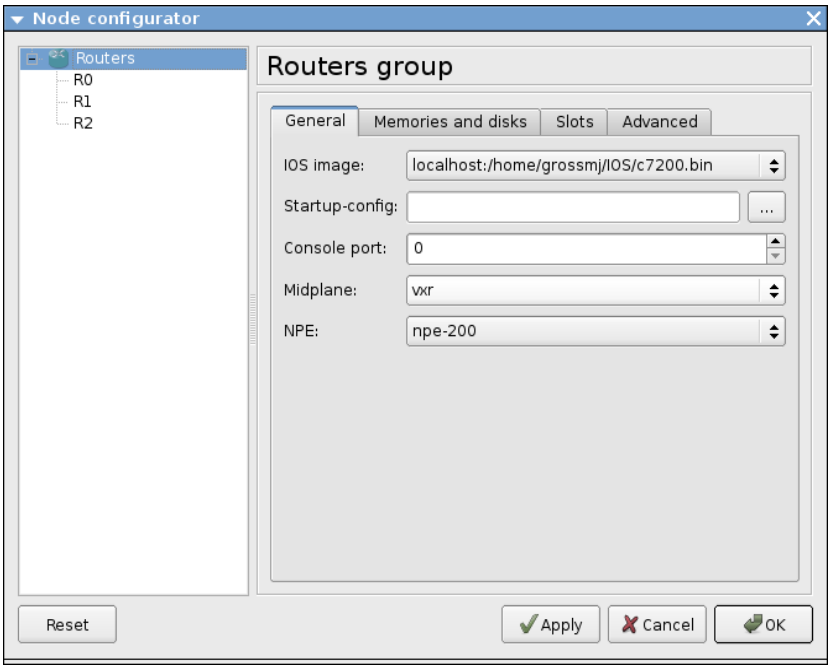
If you want to use external hypervisors (started by yourself), you can record them in the “Hypervisors” tab (see the “Client / Server and Multi-server Operation” section for more information).

All of these indicated IOS systems and hypervisors are saved in your gns3.ini file, so you just have to record them once.

Note: the gns3.ini file is located in `%APPDATA%` or `%COMMON_APPDATA%` on Windows and `$HOME/.config/` or `/etc/xdg/` or `/etc/qt4/` on Linux/Unix.

Once your IOS images are recorded, you may configure your nodes (right-click a node and choose configure).

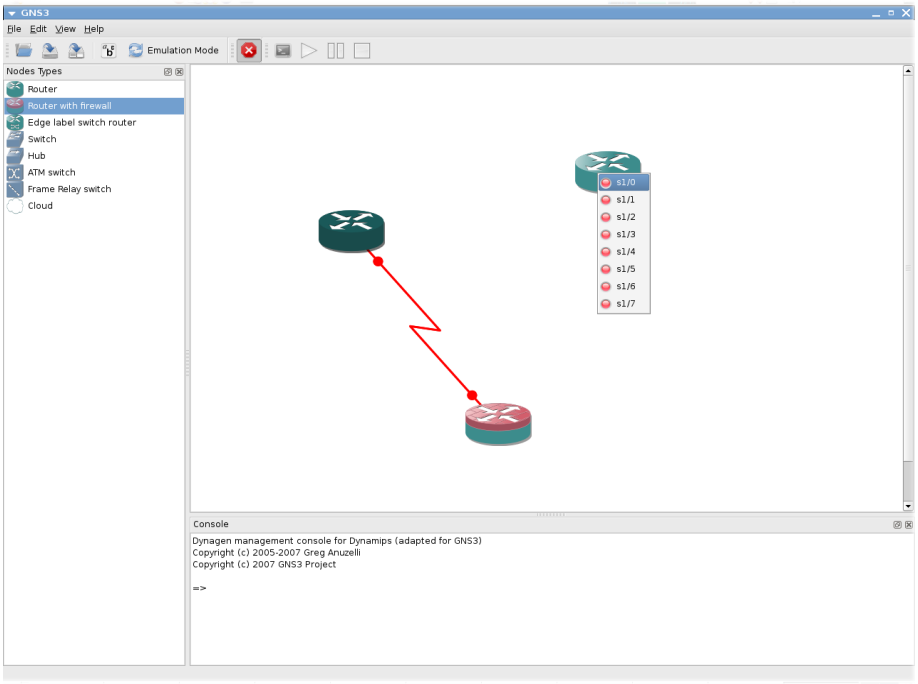
You may apply the same settings for every router by selecting “Routers” in the left tree or for specific routers by selecting their hostname in the tree.



In the node configurator you can select the IOS to use and configure many other things like the startup-config, the RAM size, the slots etc.

Next, add links between your nodes (click the “Add a link” button on the menu bar, select your source node and then your destination node). You will notice that you can choose the type of link (Ethernet, serial ...). If you do GNS3 assigns automatically the right modules for your corresponding link type in the slots of your routers and chooses the first available interface to connect the link.

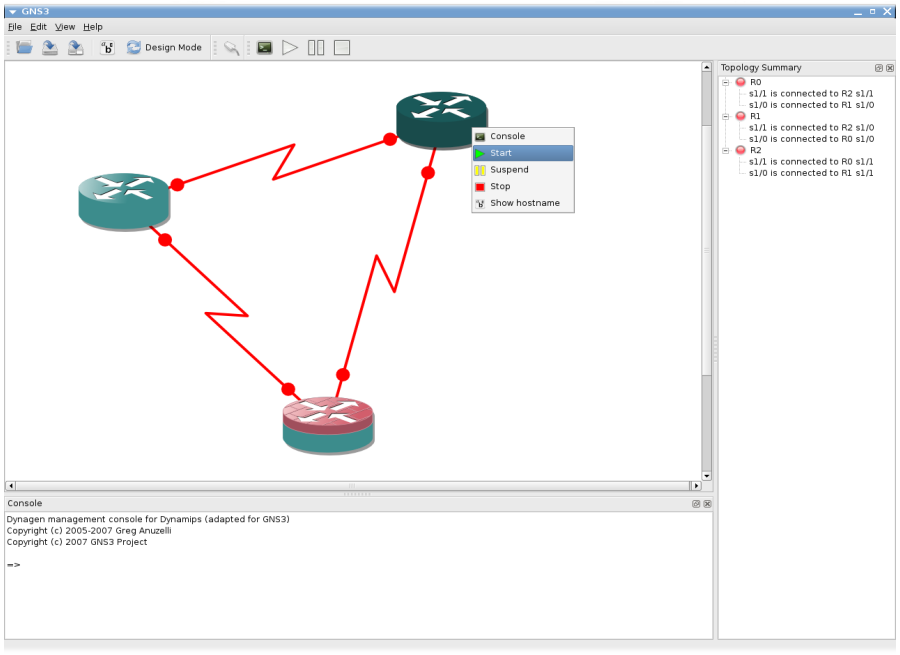
You may manually choose the interface to connect your link by choosing the manual method from the drop-down menu. But keep in mind that you must then configure the slots on your routers manually as well.



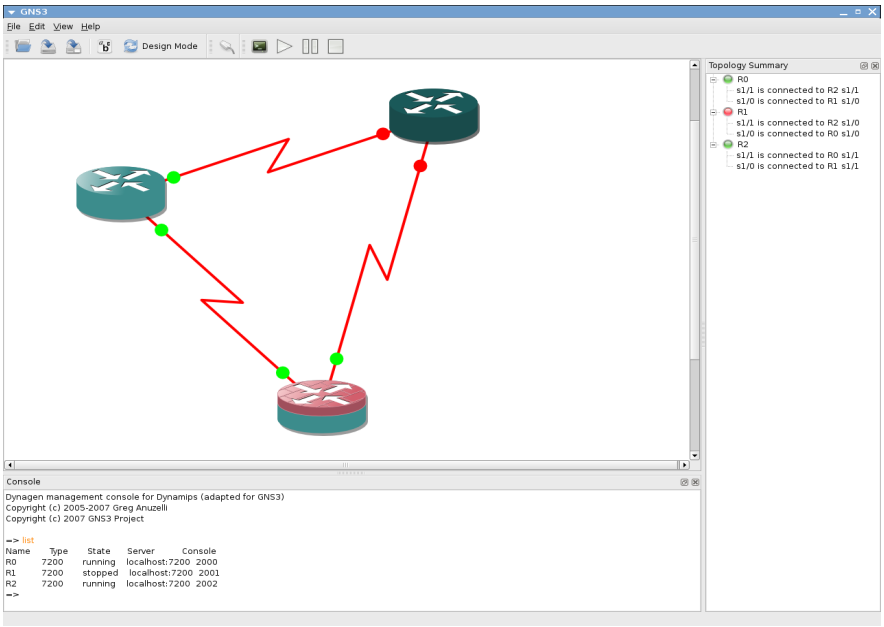
Note: used interfaces are marked in green; unused interfaces are marked in red.

The emulation mode

Your topology has been created. Now switch to the “emulation mode” (click on the Emulation Mode button in the menu bar).



Your network topology with all settings is then created on the hypervisor(s). You can start/stop/suspend an IOS instance by right-clicking on a node. If you have started the IOS, you may also go to a console session for the device. Note: multiple nodes may be selected in order to perform the operation on those nodes at the same time.

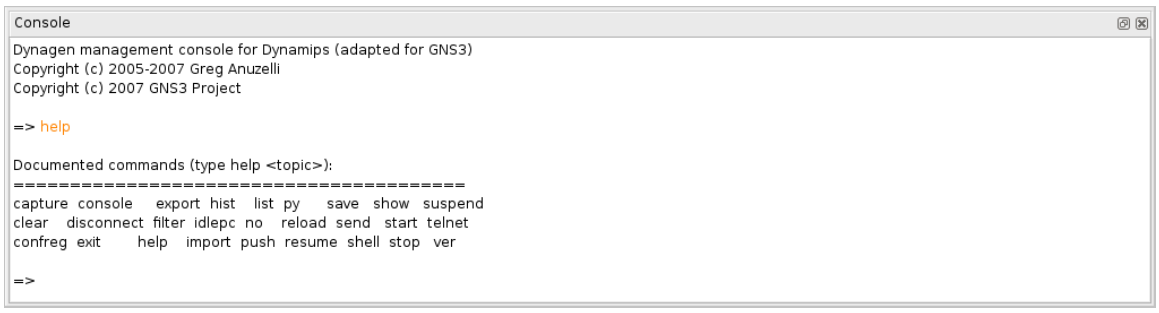


Once connected with by console to your routers, you can assign appropriate IP addresses to the serial interfaces (you can see the connected interfaces by looking at the topology summary on the right or by letting your mouse hover over a link), and “no shut” them, because they are indeed connected.

Working with the Console

Note: the Console pane at the bottom is only available when you are in emulation mode.

From the Console, use the **help** command to see a list of valid commands:

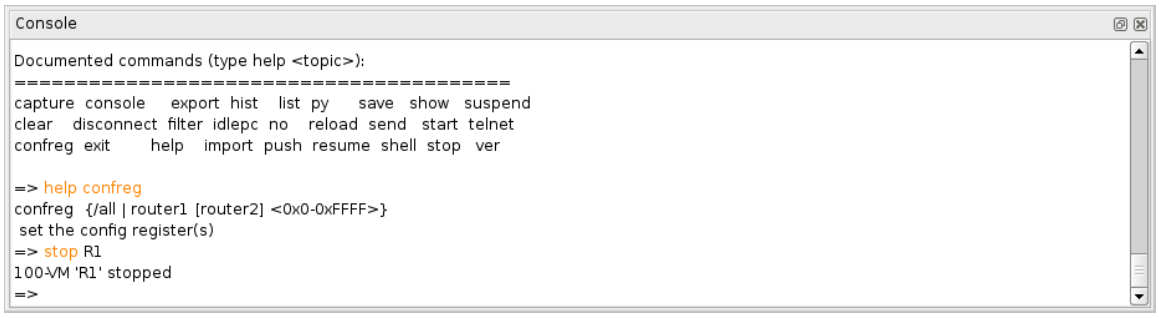


To get help on a particular command, either type **help *command*** or ***command* ?**. For example:

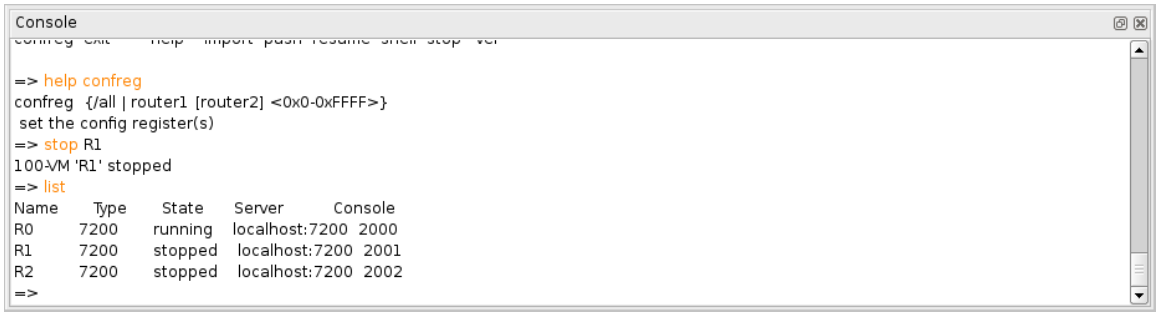
To “power off” a virtual router, use the **stop** command. Help shows the syntax as:

```
stop {/all | router1 [router2] ...}
```

To shut down a single router, type **stop *routername***:



And sure enough, the router is now stopped:



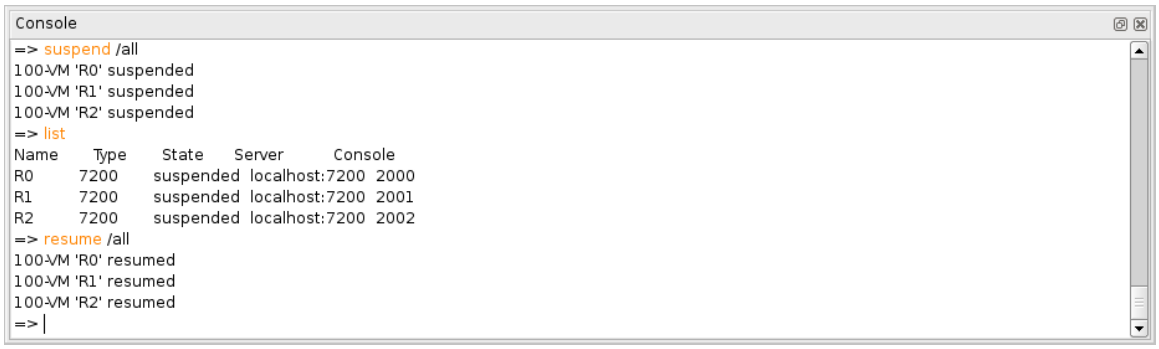
You can also provide a **list** of routers to stop, or issue **stop /all** to shut down all router instances.

To restart R1, use the **start** command:

```
start {/all | router1 [router2] ...}
```

The IOS reload command is not supported by Dynamips in virtual routers. So you can use the console **reload** command. It performs a stop, followed by a start. To reload all routers in the entire lab, issue a **reload /all** command:

The **suspend** and **resume** commands have a syntax similar to stop and start, but they temporarily pause the specified routers:



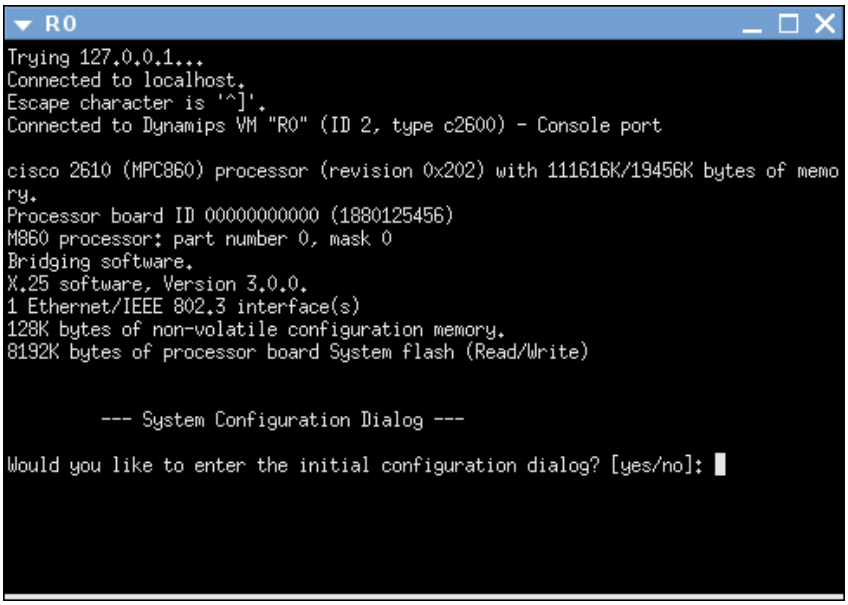
The **exit** command stops and deletes all devices from the network, and switches to design mode. If you exit the Console, your simulation will no longer be running.

Calculating Idle-PC values

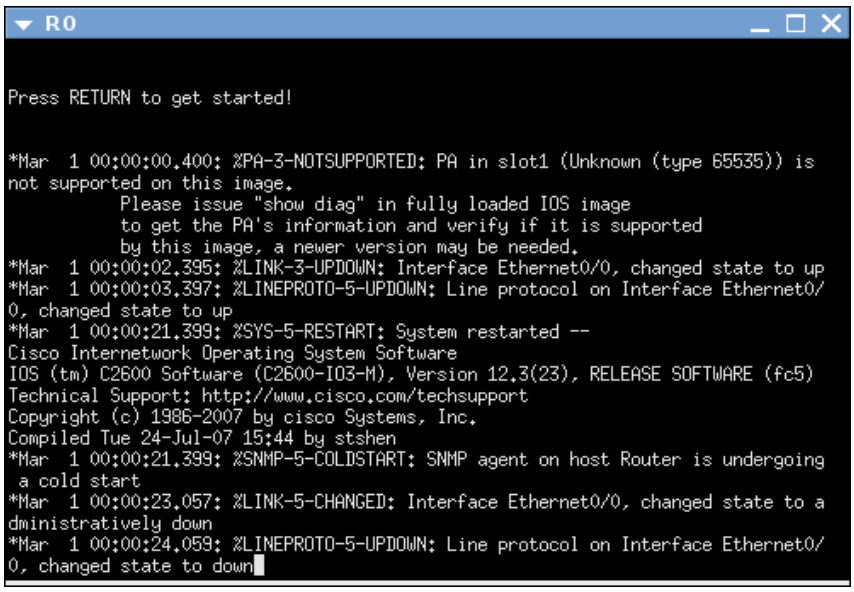
You may have noticed that your previous simulations caused your system's CPU to peg at 100% and stay there. This is because Dynamips does not know when the virtual router is idle and when it is performing useful work. The "idlepc" command performs analysis on a running image to determine the most likely points in the code that represent an idle loop in IOS. Once applied, Dynamips "sleeps" the virtual router occasionally when this idle loop is executed, significantly reducing CPU consumption on the host without reducing the virtual router's capacity to perform real work.

Here is how the process is performed. First, create a single router in design mode, choose the IOS image it will run, and then switch to emulation mode.

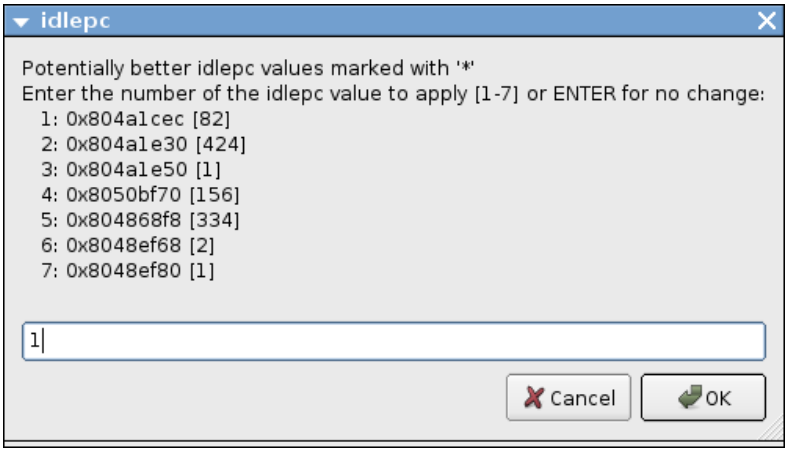
Then start your router and telnet to the running router instance. If you are presented with IOS autoconfig prompt, respond with "no". Otherwise, do not press anything:



Wait for all of the interfaces to initialize. Then wait a bit to ensure that the router is no longer booting and is idle. Your session should look something like this:



Now, switch back to the GNS3 Console, and issue the command **idlepc get *routername***. You will see a message that statistics are being gathered, and about 10 to 20 seconds later you should see a list of potential idlepc values:



Values that will most likely provide better results are marked with an asterisk. Select one of the values to try from the menu and press the OK button. You should notice your host (the one running the dynamips process) CPU utilization drop dramatically. If so, you've found a good idlepc value for this particular IOS image.

If your CPU usage did not drop, it's time to try a different value. Type **idlepc show *routername*** to show the list of values determined earlier, and select a different value.

Idle-PC values are particular to an IOS image. They will be different for different IOS versions, and even for different feature sets of the same IOS version. However Idle-PC values are not particular to your host PC, operating system, or to the version of dynamips.

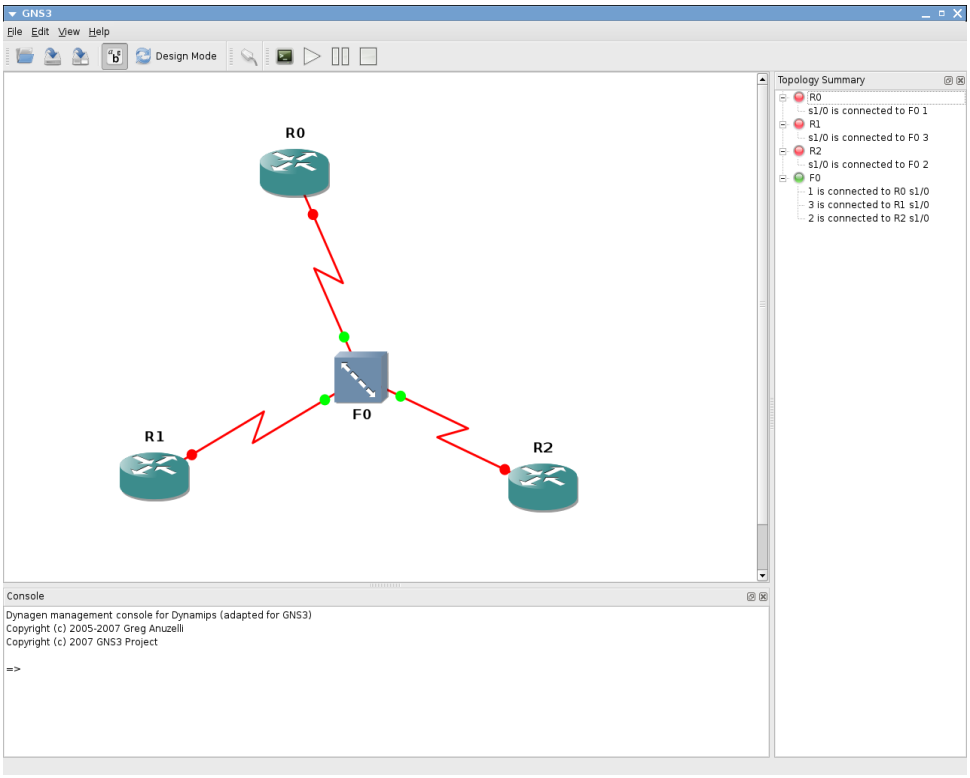
It is possible that dynamips will not be able to find and idlepc value for an image, or that the values it does find do not work. If this happens, try repeating the process again. Or you just might be out of luck with that particular image (however running into this situation is rare.)

After finding an idlepc value that works, write down the hexadecimal number (for example, 0x8048ef80). Go back to IOS Images on the Edit menu. Double-click on the IOS image in the right pane to bring up its settings in the left pane. Then type in the hexadecimal idlepc value, and click Save. This process only needs to be completed

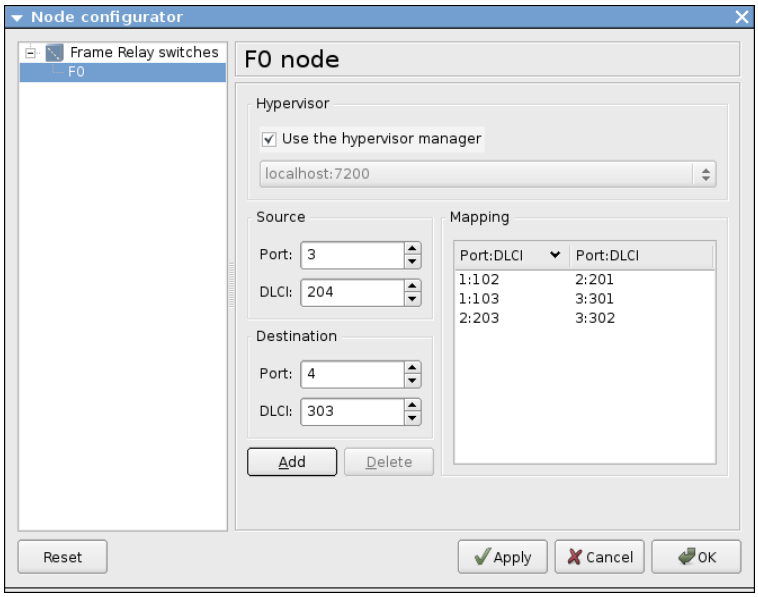
once for each IOS. Each time you open GNS3, the value will already be set. You may also enter the idlepc value while you are first setting up a new IOS version.

Using a Frame Relay device

Dynamips (and accordingly GNS3) provides support for an integrated frame relay switch. Looking at a frame relay lab:



We have connected the routers' serial interfaces to ports 1, 2, and 3 respectively on a Frame Relay switch named "F0".



Through the Node configurator we assigned a local DLCI of 102 on port 1, which maps to a DLCI of 201 on port 2. The other two entries are configured similarly, thereby creating a full mesh of PVCs between the three routers. (103 <-> 301, and 201 <-> 302).

Note: The Frame Relay switch emulated by Dynamips uses an LMI type of ANSI Annex D, not Cisco.

Launching the lab shows the following:

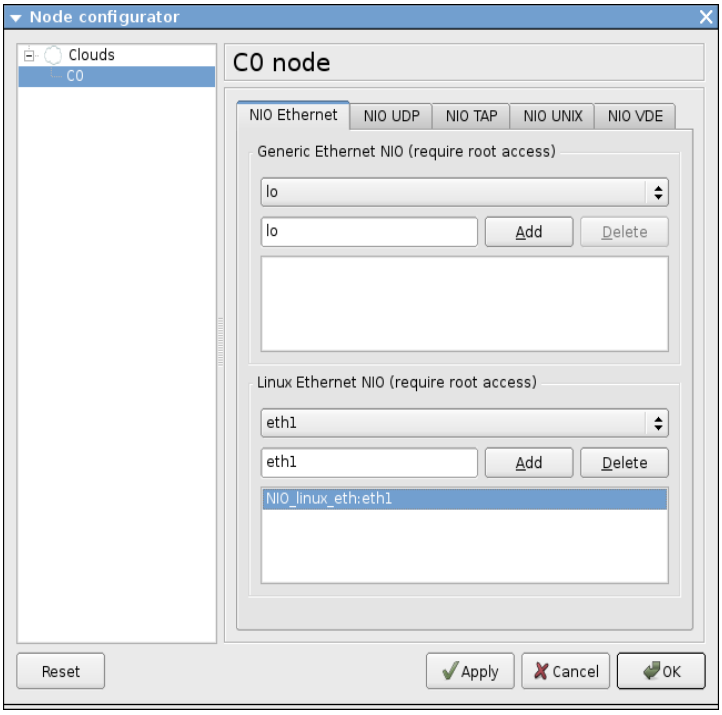
The Frame Relay switch F0 is listed, but you can't stop, start, suspend, or resume it like you can with virtual routers.

The ATM switch device can be configured in the same way.

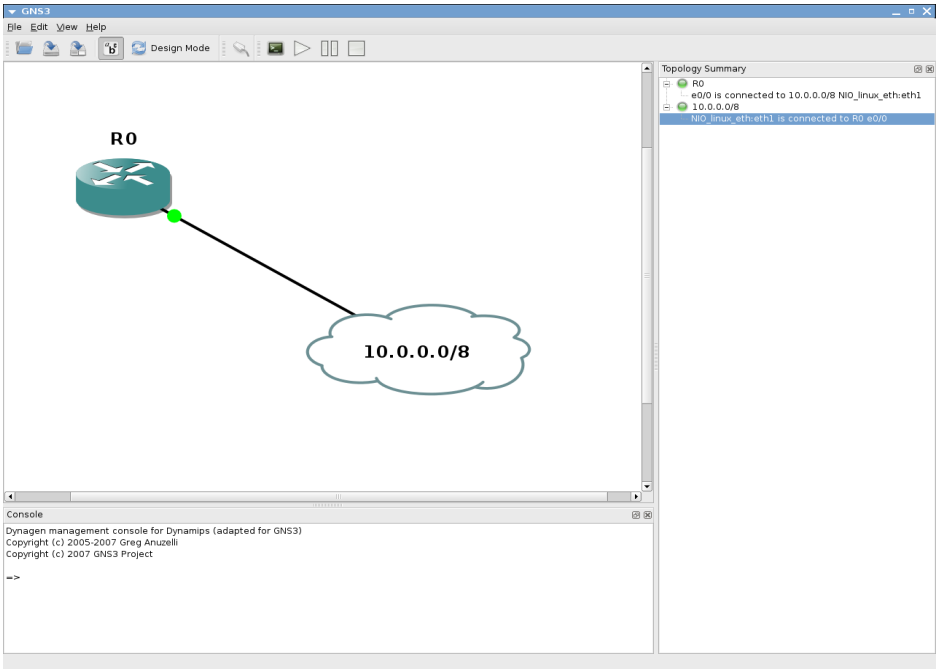
Communicating with Real Networks

Dynamips can bridge virtual router interfaces with real host interfaces, allowing your virtual network to communicate with the real world. On Linux systems, this is done with the NIO_linux_eth NIO (Network Input Ouput) descriptor.

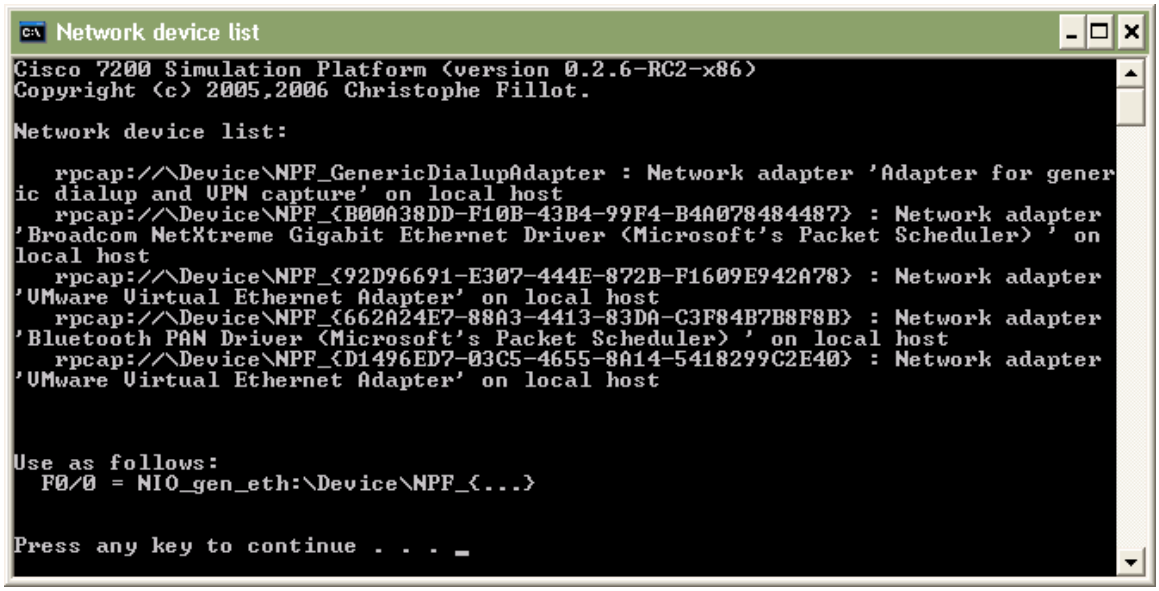
To use this feature with GNS3 you must create a “Cloud” device. A cloud represents your external connections. Then you must configure it. In this example we added the NIO_linux_eth1 NIO:



Then you can connect your cloud to a router or a Ethernet switch. In the following image we bridged the router’s e0/0 interface to the eth1 interface on the host. Packets that exit e0/0 are dumped out onto the real network through eth1, and return packets are forwarded back to the virtual router instance accordingly.



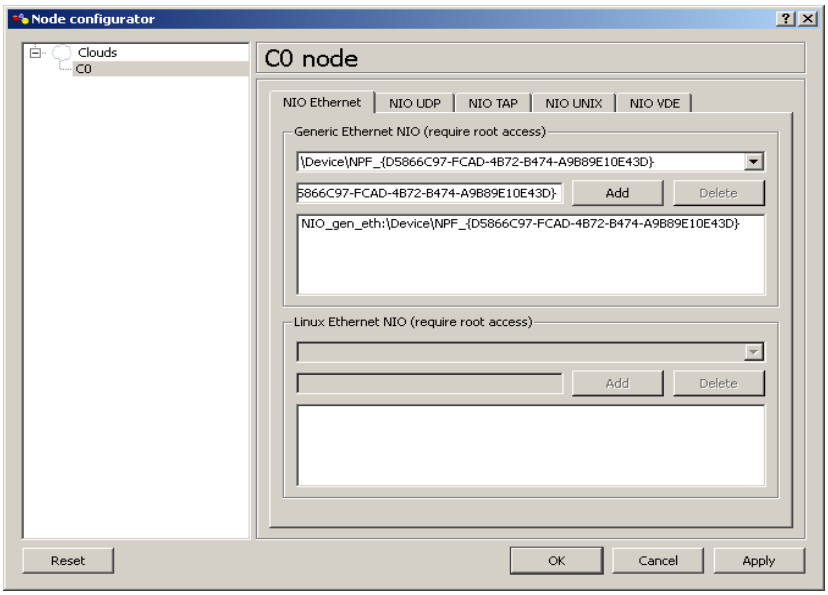
On Windows systems, the Winpcap library is used to accomplish this bridging. The interface format is a little more complex than on Linux systems. GNS3 will perform an auto-detection with the help of Dynamips to list the available interfaces. If the detection did not work, use the shortcut created by the GNS3 Windows installer (thanks to Dynagen). On the desktop, open the “Network Device List” shortcut:



So on my Windows system, I would use:

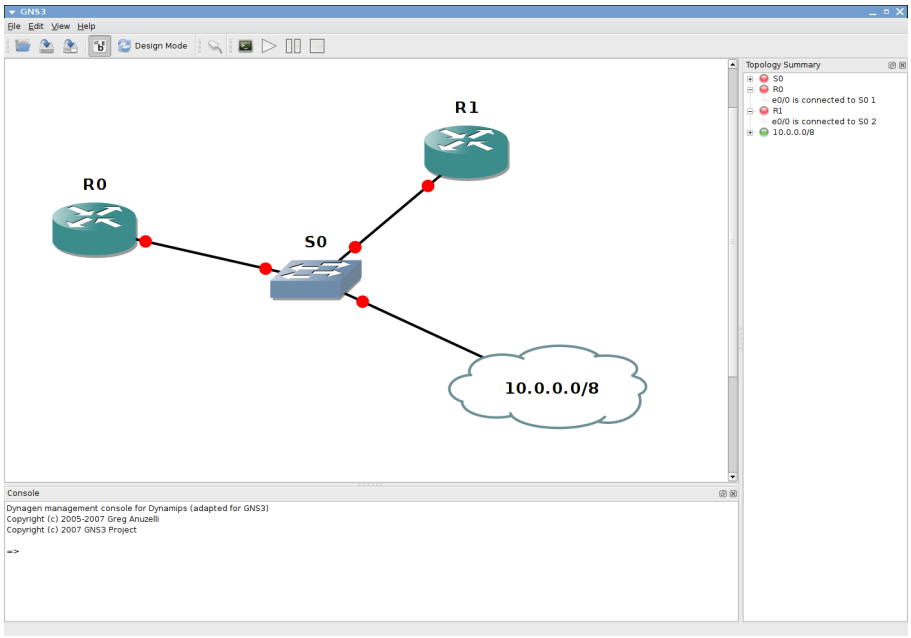
```
\Device\NPF_{B00A38DD-F10B-43B4-99F4-B4A078484487}
```

to bridge to my local Ethernet adapter. You just need to put this value in the Generic Ethernet NIO text box when configuring your “Cloud” device. Select the device from the drop down box, or paste it into the white box to the left of the Add button from the “Network Device List” shortcut. Then click the Add button to place the proper device into the white box below the Add button.

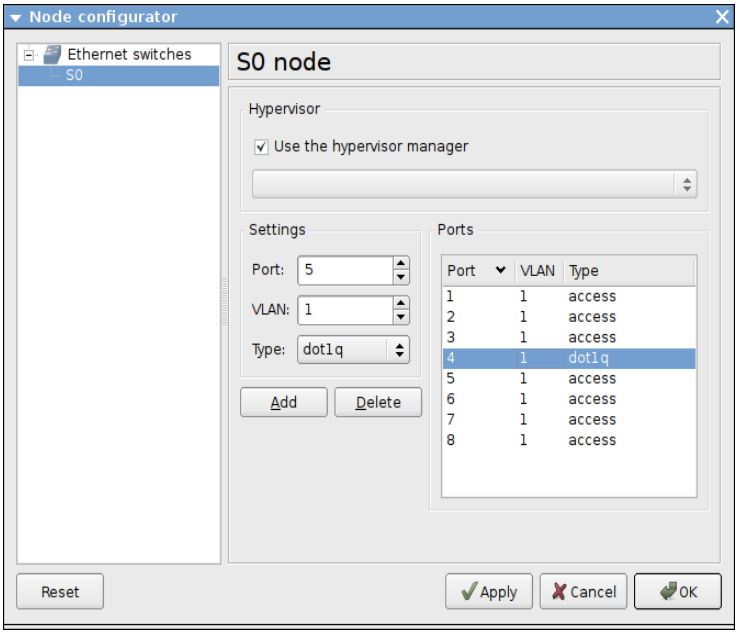


Using an Ethernet Switch device

Dynamips also provides an integrated virtual Ethernet switch that supports VLANs with 802.1q encapsulation. Looking at this lab:



Port 1 of the switch (connected to R1 e0/0) is an access port in VLAN 1. Port 2 is also an access port. Port 4 is a trunk port (specified with the dot1q keyword) with a native VLAN of 1. Trunk ports trunk all the VLANs known to the switch.



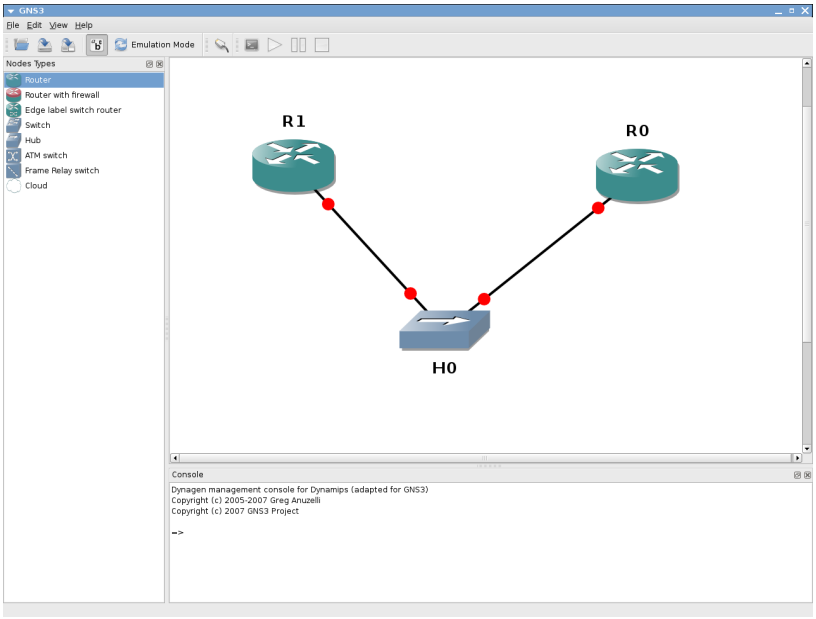
By default in GNS3, a switch has 8 access ports configured in VLAN 1.

You can also connect a switchport to the “real world” by connecting the switch to a “Cloud” device. Here we are connecting a trunk port (dotq1 encapsulation) with a native vlan of 1 to the host’s eth1, or Windows network device using the NIO_gen_eth Winpcap NIO (see Communicating with Real Networks section). If this host interface is connected to a real switch that is configured for trunking, you can now easily connect any router instance to any VLAN you wish.

The console includes CLI commands to show and clear the MAC address tables of virtual Ethernet switches. Those commands are s**how mac *Ethernet_switch_name*** and **clear mac *ethenet_switch_name***.

Using an Hub device

GNS3 allows you to create hub devices. Hub devices are the simplest devices to configure because you just need to choose the number of ports for each device (8 ports by default). Here is an example of a lab that use a hub:



In this example R0 and R1 are sharing the same media through the hub.

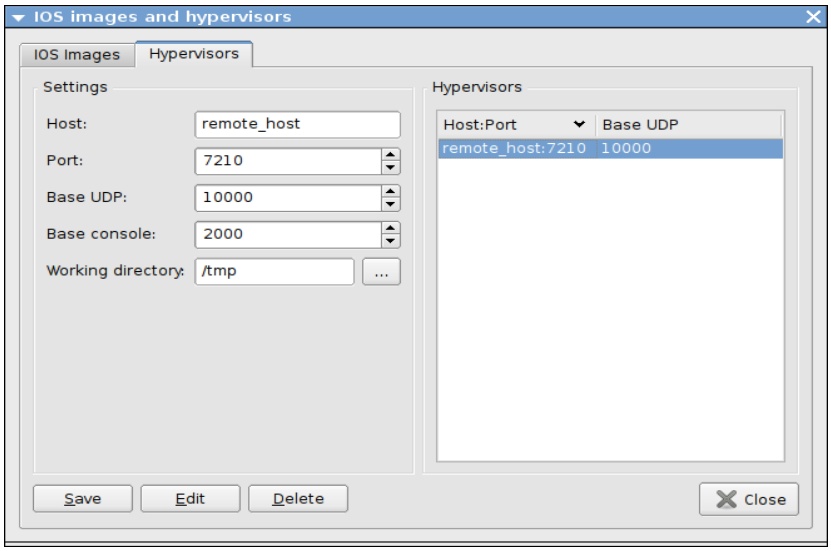
WIC Modules

Dynamips 0.2.8-RC1 added support for several WIC modules. Currently, these are the WIC-1T and WIC-2T on the 1700, 2600, 2691 and 3700 platforms, and the WIC-1ENET on the 1700. See the Hardware Currently Emulated section for specific model info and how many WIC slots are provided on each platform.

Note: at the moment GNS3 doesn't support WIC modules or the 1700 platform, but this will be integrated in a future release.

Client / Server and Multi-server Operation

The Dynamips “Hypervisor” mode that is used by GNS3 is a TCP/IP communications channel, so GNS3 can run on a different machine than the Dynamips emulator. This is done by manually specifying a hypervisor in the “IOS images and hypervisors” window.



Here we recorded a new hypervisor which will listen on port 7210 on the “remote_host” machine. You can also specify the fully qualified path to the working directory where the hypervisor will store all of its generated files on the Dynamips host. Be sure to use the correct directory separation character for the platform (here forward slashes for a Linux system).

Note: you must use a DNS name or an IP address in the host text box.

“Base UDP” is the base port for UDP NIOs that Dynamips uses to make the connections between your nodes. Dynamips will use a UDP port for each end of a link. For example, six UDP ports are used for a full meshed lab of 3 routers and a base udp of 10000.

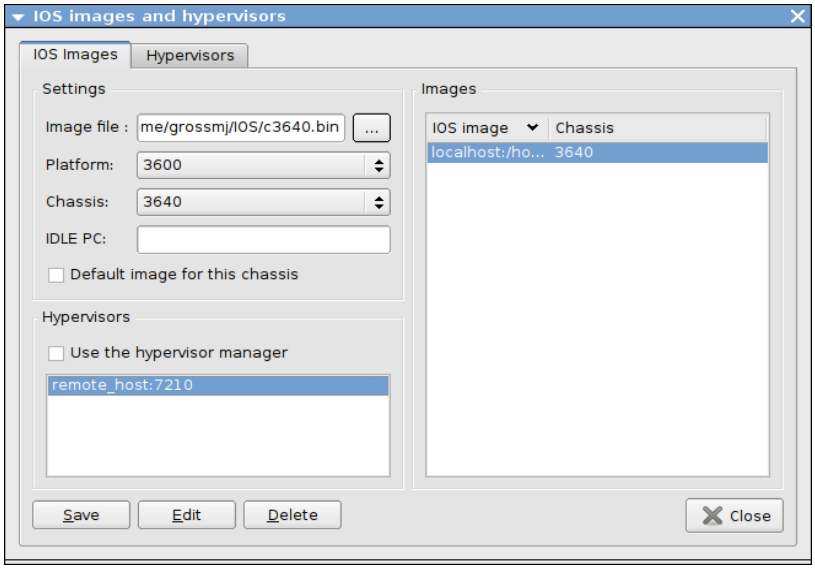
Here is an output of the netstat command that show you the connections between the nodes and the UDP port picked up by Dynamips:

```
udp        0      0 localhost:10000         localhost:10001         ESTABLISHED
udp        0      0 localhost:10001         localhost:10000         ESTABLISHED
udp        0      0 localhost:10002         localhost:10003         ESTABLISHED
udp        0      0 localhost:10003         localhost:10002         ESTABLISHED
udp        0      0 localhost:10004         localhost:10005         ESTABLISHED
udp        0      0 localhost:10005         localhost:10004         ESTABLISHED
```

Now let's imagine you want to run two connected IOS instances which are separately created on 2 different hypervisors on the same host and you choose the same UDP base for the hypervisors. Each hypervisor will try to take the same UDP port (10000 in the previous example) for each end of the connection and, of course, this will “collide” because Dynamips thinks they are different servers and therefore those UDP ports are safe to re-use.

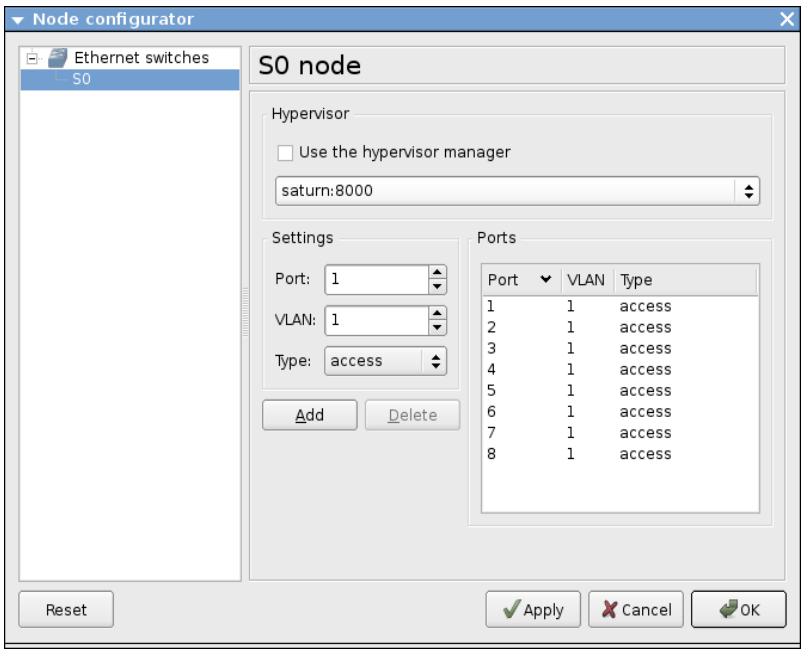
“Base console” is the TCP base port that the hypervisor will use when you open a console on an IOS. This value shouldn't “collide” because GNS3 uses the following formula to affect a console port to each router: “base port” + router id (which is unique). So it is safe to let the same value for every hypervisor.

Once your hypervisors are recorded, your can bind them to your IOS images. Select your image, uncheck the “Use the hypervisor manager” option, and then choose your hypervisor in the list. Now every router that is configured with this IOS image is able to communicate with that hypervisor.



Note: be sure that any host-based firewalls running on all your Dynamips servers (for example, XP SP2's firewall) are permitting the necessary traffic. This includes the Dynamips server port (defaults to TCP 7200), the console ports (e.g. TCP 2000, 2001, …) and the ports used by the NIO connections between interfaces, which start at UDP 10000 and work up from there.

You also have the possibility to choose on which hypervisor your non-IOS devices (switches and hubs) will run when configuring them. For example, on an Ethernet switch you can select the hypervisor in a list box.



Here we have chosen to run our switch on the hypervisor host named saturn which listens to port 8000.

Memory Usage Optimizations

As described in the Resource Utilization section, your labs can consume a large amount of real and virtual memory. The “ghostios” and “sparsemem” options were added to address both of these issues, respectively.

The Ghostios option can significantly reduce the amount of real host RAM needed for labs with multiple routers running the same IOS image. With this feature, instead of each virtual router storing an identical copy of IOS in its virtual RAM, the host will allocate one shared region of memory that they will all utilize. So for example, if you are running 10 routers all with the same IOS image, and that image is 60 MB in size you will save 9*60 = 540 MB of real RAM when running your lab. Enabling ghostios is as simple as checking a checkbox in the Dynamips preferences. This option is activated by default and applied to all router instances in the lab.

When enabled, you will notice additional files in the same directory as you router nvram files with names like “c3660-ik9o3s-mz.124-10.image.ghost”. This is the nmap’ed file that contains the shared memory region. The other files typically created with a router instance are created as well (log, nvram, and possibly bootflash files).

Measuring the amount of host memory saved with ghostios can be a little tricky due to the complexities of memory management in modern OSs. See this sticky post in the General section of Hacki’s Forum titled “Understanding memory usage and RAM Ghosting” for the gory details.

The “sparsemem” feature does not conserve real memory, but instead reduces the amount of virtual memory used by your router instances. This can be important, because OS limits a single process to 2 GB of virtual memory on 32-bit Windows, and 3 GB on 32-bit Linux. For example, on Windows, after the VM space is used by cygwin and other libraries dynamips depends on, this only leaves room for 4 router instances @ 256 MB each! Enabling sparsemem only allocates virtual memory on the host that is actually used by the IOS in that router instance, rather than the entire amount of RAM configured. This can allow you to run more instances per dynamips process before you have to resort to running multiple dynamips processes. See this FAQ item for more info on this issue.

The “sparsemem” feature is activated by default in GNS3 with no current option to disable it. Note: if users want to have an option to disabled this feature, please let us know.

Packet Capture

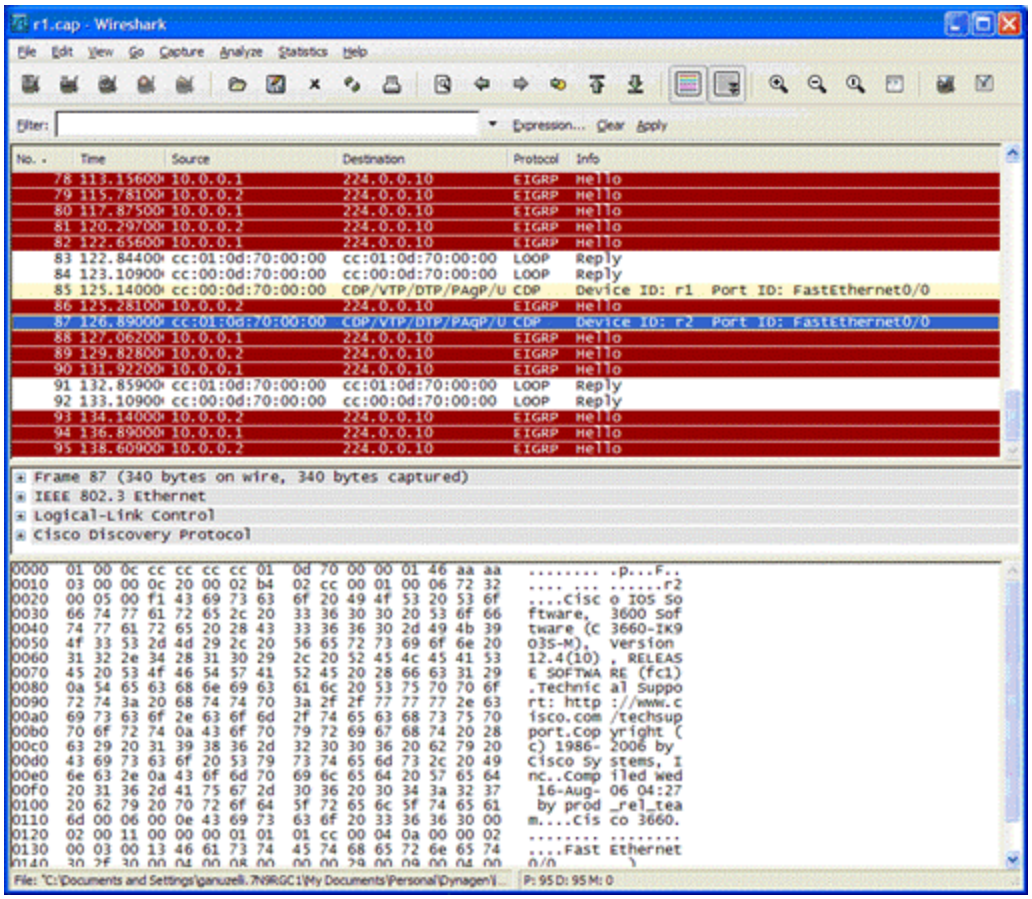
Dynamips / GNS3 can capture packets on virtual Ethernet or Serial interfaces and write the output to a capture file for use with applications like tcpdump, Wireshark, or any other application that can read the libpcap capture file format.

Consider three routers in series, “r1” and “r2” are connected via an Ethernet cable, and r2 connects to r3 via a point-to-point serial connection with HDLC encapsulation.

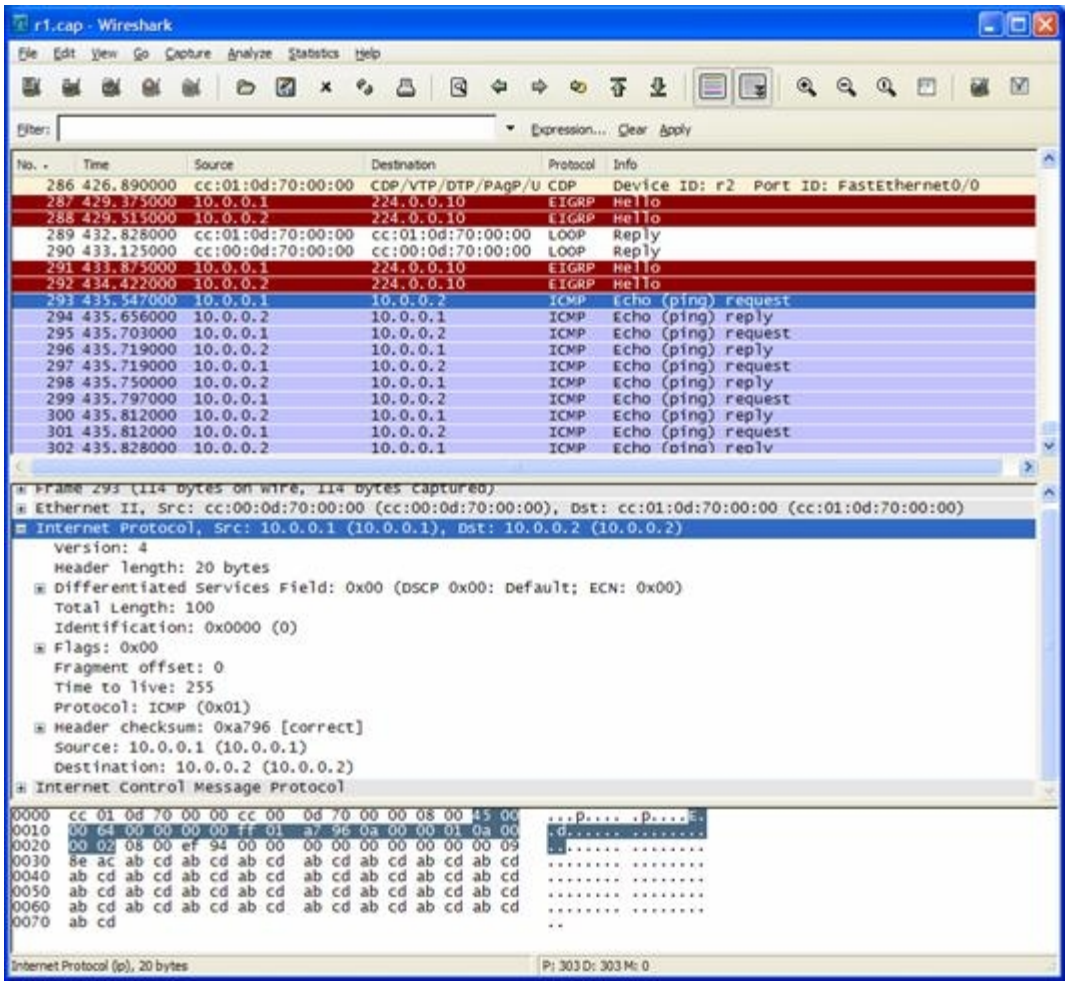
To begin capturing traffic at r1’s f0/0 interface and to write it to the file “r1.cap”, enter the following in the console window:

```
capture r1 f0/0 r1.cap
```

To view the traffic in real-time, open the file with Wireshark:



The capture is continuing to write packets to the output file. If we ping r2 from r1, then hit the “reload this capture file” icon we see:



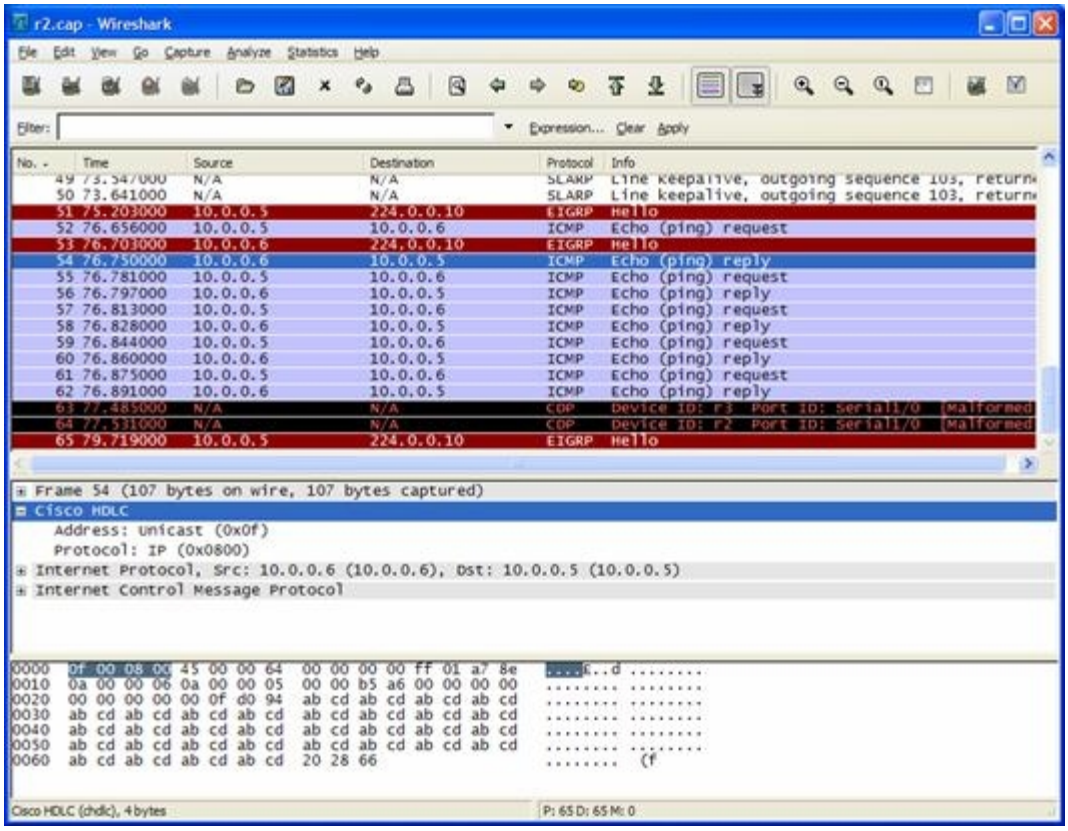
To stop capturing packets, enter:

```
no capture r1 f0/0
```

Dynamips / GNS3 can capture packets at serial interfaces too. In this case we must also specify the encapsulation we are using on our routers, so Wireshark will know how to decode the packets. Our encapsulation options are FR (Frame-Relay), HDLC, or PPP. To capture some traffic on our HDLC encapsulated r2 to r3 link use:

```
capture r2 s1/0 r2.cap HDLC
```

Now we can open r2.cap, and the decode looks like this:



Now end the capture with “no capture r2 s1/0”. Note that you can have multiple captures running simultaneously against different interfaces on different routers.

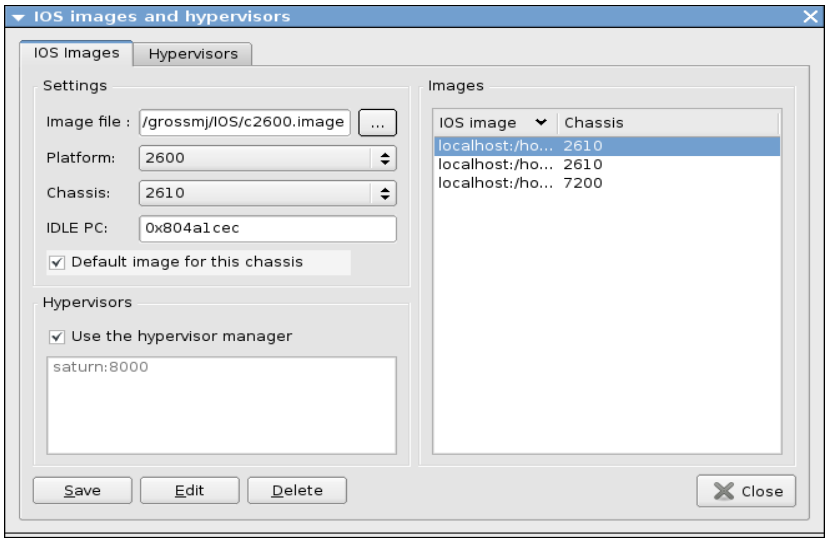
Save and load a topology

GNS3 can save and load your network topologies in the Dynagen INI-like configuration format (.net extension). This means that you can use the same files for both GNS3 and Dynagen. Because GNS uses Dynagen there are two restrictions concerning the use of .net files in GNS3:

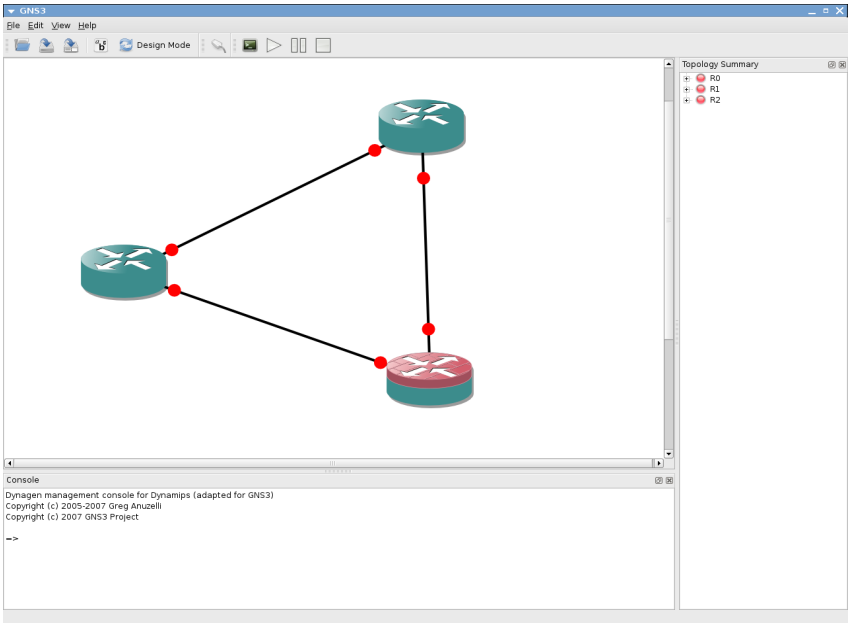
- To load your topology you must have a configured Dynamips path in the preferences dialog.
- To save your topology you must be in emulation mode.

Note: It is planned to make it possible to save your topologies in design mode in a future release of GNS3.

One thing you must be aware of is that GNS3 has a different way than Dynagen for handling .net files. For example, all the settings (ram, rom, nvram etc.) are recorded in device sub-sections and not in a default model section like Dynagen does. GNS3 will also record the idlepc and the IOS image filename in device sub-sections (because you have the possibility to use different IOS images for the same chassis on your routers). To override this behavior you can set an IOS image as default, then GNS3 will create a model sub-section where it will put the IOS filename and idlepc.



In the following example we created a topology of 3 routers (2610 chassis) and we saved it in emulation mode. A default IOS image for the 2610 chassis has been configured.



Here is the result saved in the .net file:

```
sparsemem = True
ghostios = True
[localhost:7200]
      udp = 10000
      console = 2000
      workingdir = /tmp/
      [[2610]]
            image = /home/grossmj/IOS/c2600.image
            idlepc = 0x804a1cec
      [[ROUTER R0]]
            model = 2610
            console = 2000
            ram = 128
            nvram = 128
            rom = 4
            disk0 = 8
            disk1 = 8
            mmap = True
            exec_area = 64
            slot0 = CISCO2600-MB-1E
            slot1 = NM-4E
            e0/0 = R2 e1/0
            e1/0 = R1 e1/0
            x = 15.0
            y = -151.0
      [[ROUTER R1]]
            model = 2610
```

```
        console = 2001
        ram = 128
        nvram = 128
        rom = 4
        disk0 = 8
        disk1 = 8
        mmap = True
        exec_area = 64
        slot0 = CISCO2600-MB-1E
        slot1 = NM-4E
        e0/0 = R2 e0/0
        x = -255.795418536
        y = -31.8822509939
[[ROUTER R2]]
        model = 2610
        console = 2002
        ram = 128
        nvram = 128
        rom = 4
        disk0 = 8
        disk1 = 8
        mmap = True
        exec_area = 64
        slot0 = CISCO2600-MB-1E
        slot1 = NM-4E
        x = 41.6776695297
        y = 107.837049096
```

You can now load this topology in Dynagen or GNS3.

Note: see Dynagen's tutorial to learn more about the .net file format.

Other Commands / Features

Here are some additional commands that can be used in the console that are not explained in this tutorial. Refer to the online help (`command /?` or `help command`) for usage:

- import / export – Imports and exports router configs from nvram to text files on your host. Can be used to get a copy of your current configs, or as a "snapshotting" feature to save your router configs before you make changes.
- push / save – Much like import and export, but the configs are stored as base64 encoded "blobs" right in your network file (specified with the "configuration" option). This allows you to distribute an entire lab with the network topology and IOS configs all in a single .net file
- filter – Applies a connection filter to an interface. Currently the only filter supported by dynamips is "freq_drop", which drops x out of every y packets across a link (simulating intermittent packet loss).
- send – Used to send raw hypervisor commands to dynamips (see README.hypervisor included with the dynamips source for documentation on hypervisor commands). These hypervisor commands are how GNS3 communicates with Dynamips. This command would typically only be used if developing new features in dynamips, experimenting, or simply curious.
- ver – outputs the version of Dynagen being used, as well as the versions of each dynamips instance Dynagen is connected to.
- hist – Console command history (like "history" in bash)

Also be sure to keep up on Dynamips development by following the technical blog at http://www.ipflow.utc.fr/blog/ for the latest developments.

Hardware Currently Emulated

Borrowed from ggee's excellent post on Hacki's forum:

==========1700s==========

1710

Slots: 0 (available)

WIC slots: 0

CISCO1710-MB-1FE-1E (1 FastEthernet port and 1 Ethernet port, automatically used)

Note, interfaces do not use a slot designation (e.g. “f0”)

1720

Note, interfaces do not use a slot designation (e.g. “f0”)

1721

Note, interfaces do not use a slot designation (e.g. “f0”)

1750

Note, interfaces do not use a slot designation (e.g. “f0”)

1751

1760

Slots: 0 (available)

WIC slots: 2

C1700-MB-1ETH (1 FastEthernet port, automatically used)

Cards:

- WIC-1T (1 Serial port)
- WIC-2T (2 Serial ports)
- WIC-1ENET (1 Ethernet ports)

==========2600s==========

2610

Slots: 1 (available)

WIC slots: 3

CISCO2600-MB-1E (1 Ethernet port, automatically used)

2611

Slots: 1 (available)

WIC slots: 3

CISCO2600-MB-2E (2 Ethernet ports, automatically used)

2620

Slots: 1 (available)

WIC slots: 3

CISCO2600-MB-1FE (1 FastEthernet port, automatically used)

2621

Slots: 1 (available)

WIC slots: 3

CISCO2600-MB-2FE (2 FastEthernet ports, automatically used)

2610XM

Slots: 1 (available)

WIC slots: 3

CISCO2600-MB-1FE (1 FastEthernet port, automatically used)

2611XM

Slots: 1 (available)

WIC slots: 3

CISCO2600-MB-2FE (2 FastEthernet ports, automatically used)

2620XM

Slots: 1 (available)

WIC slots: 3

CISCO2600-MB-1FE (1 FastEthernet port, automatically used)

2621XM

Slots: 1 (available)

WIC slots: 3

CISCO2600-MB-2FE (2 FastEthernet ports, automatically used)

2650XM

Slots: 1 (available)

WIC slots: 3

CISCO2600-MB-1FE (1 FastEthernet port, automatically used)

2651XM

Slots: 1 (available)

WIC slots: 3

CISCO2600-MB-2FE (2 FastEthernet ports, automatically used)

Cards:
- NM-1E (Ethernet, 1 port)
- NM-4E (Ethernet, 4 ports)
- NM-1FE-TX (FastEthernet, 1 port)
- NM-16ESW (Ethernet switch module, 16 ports)
- NM-NAM
- NM-IDS
- WIC-1T (1 Serial port)
- WIC-2T (2 Serial ports)

==========3600s==========

3660

Slots: 6 (available)

3640

Slots: 4

3620

Slots: 2

Cards:
- NM-1E (Ethernet, 1 port)
- NM-4E (Ethernet, 4 ports)
- NM-1FE-TX (FastEthernet, 1 port)
- NM-16ESW (Ethernet switch module, 16 ports)
- NM-4T (Serial, 4 ports)
- Leopard-2FE (Cisco 3660 FastEthernet in slot 0, automatically used)

==========3700s==========

2691 (The 2691 is essentially a 3700 with 1 slot)

Slots: 1 (available)

WIC slots: 3

3725

Slots: 2 (available)

WIC slots: 3

3745

Slots: 4 (available)

WIC slots: 3

Cards:

- NM-1FE-TX (FastEthernet, 1 port)
- NM-4T (Serial, 4 ports)
- NM-16ESW (Ethernet switch module, 16 ports)
- GT96100-FE (2 integrated ports, automatically used)
- NM-NAM
- NM-IDS
- WIC-1T (1 Serial port)
- WIC-2T (2 Serial ports)

==========7200s==========

7206

Slots: 6 (available)

Chassis types:

- STD
- VXR

NPEs:

- NPE-100
- NPE-150
- NPE-175
- NPE-200
- NPE-225
- NPE-300
- NPE-400
- NPE-G2 (Requires the use of NPE-G2 IOS images)

Cards:

- C7200-IO-FE (FastEthernet, slot 0 only)
- C7200-IO-2FE (FastEthernet, 2 ports, slot 0 only)
- C7200-IO-GE (GigabitEthernet, slot 0 only)
- PA-FE-TX (FastEthernet)
- PA-2FE-TX (FastEthernet, 2 ports)
- PA-4E (Ethernet, 4 ports)
- PA-8E (Ethernet, 8 ports)
- PA-4T+ (Serial, 4 ports)
- PA-8T (Serial, 8 ports)
- PA-A1 (ATM)
- PA-POS-OC3 (POS)
- PA-GE (GigabitEthernet)

FAQs

How do I determine idle pc values from GNS3?

See the “Calculating Idle-PC” values section in this tutorial.

When I try to run more than 4 router instances @ 256 MB each (or 6 instances @ 160 MB each) on Windows, or more than 7 instances @ 256 MB each (or 11 instances @ 160 MB each) on 32-bit Linux Dynamips crashes.

This problem shouldn't occur if you used the Hypervisor Manager (used by default) with correct settings This hypervisor is designed to “load-balance” your IOS instances on multiple Dynamips processes, but if you used hypervisors on your own on remote hosts, the problem is due to a per process memory limit.

By default, Windows has a 2 GB per process limit that (after including the memory used by the virtual router RAM, cygwin, libraries, and ‘scratch space”) that you are bumping up against. 32-bit Linux has a 3 GB per process limit by default.

The solution is to run multiple instances of Dynamips on the same system listening on different control ports like so:

On Windows:

start /belownormal /min "Dynamips" "dynamips.exe" -H 7200

start /belownormal /min "Dynamips" "dynamips.exe" -H 7201

On Linux/Unix:

nice dynamips –H 7200 &

nice dynamips –H 7201 &

Then correctly record those hypervisors in GNS3 (see Client / Server and Multi-server Operation)

I have a complex lab with several routers, and my serial interfaces are flapping, eigrp neighbor adjacencies are failing, show run and write mem takes forever.

This is most likely a performance issue with the host PC. Large labs consume lots of RAM and CPU. By default, the router's DRAM is simulated as a disk file of the same size as the allocated RAM. The host OS's caching features will naturally try to keep the most commonly access pages in RAM. But as your RAM runs low, disk thrashing will begin. The virtual routers then become "starved" for the CPU and start missing various hellos and such. There are several options for resolving this:

- Use a more powerful host (more RAM and / or a faster CPU)
- Distribute your lab across several hosts
- Use lower-end virtual routers where possible. For example, a 3620 running 12.2 IP base only needs 32 MB of RAM and could be used when you need to simulate a simple LAN router, or "the Internet".

There is a newer version of Dynamips available than the one bundled with the GNS3 Windows installer. How do I use it with GNS3 ? / How do I use GNS3 with Windows 2000 or Windows XP SP1?

The version of Dynamips included with the Windows GNS3 installer requires Windows XP SP2. In either of the above cases, download the Windows binaries from the

Dynamips site (http://www.ipflow.utc.fr/blog/). For Windows XP / 2003 use “dynamips-wxp.exe”. For Windows 2000, use the file “dynamips-w2000.exe” instead. Then copy both your executable and “cygwin1.dll” to “C:\Program Files\GNS3\Dynamips”, replacing the existing files.

On Linux / Unix / OS X, when I bridge a router or switch interface to my local host I can’t ping it from my host. But this works on Windows? What gives?

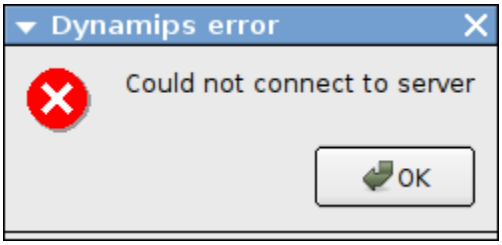
This does generally work on Windows (depending on your network card) but not on Linux / Unix. Most likely this is due to differences between libpcap and Winpcap, and the differences in the network stacks on Unix / Windows (e.g. NDIS). However, you should be able to ping your bridged interfaces from other systems on the bridged network. If this does not work on Windows for your particular NIC, try creating a Windows loopback adapter and bridging to that. See this thread for more info. On Linux you can use a tap interface and the NIO_tap NIO type. For OS X you can install tun/tap drivers as detailed in this thread.

Can I use GNS3 to make my network diagrams?

Yes but at the moment GNS3 only proposes symbols that are used in the emulation process, so you can made your diagrams with these symbols and export them as an image (jpeg, png, xpm, bmp are supported formats).

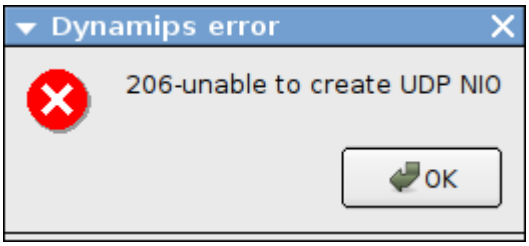
In a future release of GNS3, a package of decorative symbols and a way to write text on the drawing area will be implemented.

I get Dynamips error message boxes but I don't know what is wrong?



If you have this error message, that means that the hypervisor is not listening, you should look at the hypervisor settings in the Dynamips preferences if you are using the hypervisor manager or at the hypervisors section in the “IOS images and hypervisors” window. If the problem persists you can try to:

- Start the hypervisor manually and see his ouput
- Check if you have not another Dynamips process running on your system
- Clean the working directory of Dynamips (especially the lock file)



This error message means that Dynamips can't create an UDP NIO, this is often due to a collision when choosing UDP ports, check your hypervisors settings and see the “Client / Server and Multi-server Operation” for more information.

I have a question / I’m having a problem / I think I’ve found a bug. How do I submit a quality post on the forum or the bug tracking system thereby increasing the likelihood that someone will be able to help me out?

Be sure to note all the following in your post:

- The specific details of your issue
- Try to provide the simplest lab you can that recreates the issue
- Include the exception.log file if it exists.
- Any output from Dynamips (when using it externally)

Note: It is planned to provide a debug mode in a future release.