Instalación y Configuración de OpenVPN 2.0 para Cómputo Paralelo Intercluster

Fernando G. Tinetti* Walter Aróztegui

III-LIDI, Facultad de Informática, UNLP
50 y 115, 1900, La Plata
Argentina

CeTAD, Facultad de Ingeniería, UNLP
48 y 116, 1900, La Plata
Argentina

Reporte Técnico PLA-002-2006[1]
Julio 2006

Resumen

Este reporte está orientado básicamente a documentar la instalación y configuración de una VPN (Virtual Private Network) con OpenVPN para llevar a cabo cómputo paralelo intercluster. Esto implica que, por un lado un porcentaje de la documentación puede verse como información "oficial" previamente publicada, y por el otro tiene todos los detalles de una instalación y configuración específica, es decir que normalmente tiene las referencias a las cosas propias de una instalación en particular que no pueden ni deben tener los documentos oficiales de las distribuciones de software. En este caso en particular, la mayor parte de la documentación útil sobre OpenVPN está en http://openvpn.net y en su corresponiente howto: http://openvpn.net/howto.html. Se debe notar que estas referencias ahora mismo son del OpenVPN 2.0, pero la versión posiblemente se actualizará con el paso del tiempo. La idea básica de tener OpenVPN es la de tener una forma técnicamente sustentable para correr aplicaciones paralelas intercluster. Se intenta que este reporte documente *todos* los pasos necesarios para tener una VPN con OpenVPN: desde la instalación misma hasta la configuración y los detalles de seguridad que deben estar definidos al menos a nivel de firewalls.

1. Introducción

En el contexto de las herramientas (software), metodología y configuración (incluyendo seguridad) necesarias para cómputo paralelo intercluster, se puede llegar a la necesidad de definir y mantener una VPN (Virtual Private Network). Está claro que esto no es necesario en una red local o cluster para cómputo paralelo, y quizás hayan otras alternativas para cómputo paralelo intercluster. En este sentido, si bien se podría hacer referencia a ssh como la herramienta básica a utilizar, se torna bastante poco sustentable debido a la tendencia de bibliotecas y aplicaciones paralelas de abrir conexiones TCP y UDP que pueden ser muy probablemente bloqueadas por los firewalls de los clusters intervinientes o, peor aún, bloqueadas por firewalls *intermedios* en los routers de entrada y salida de las instituciones en las cuales están instalados los clusters. El caso más clásico de los ambientes universitarios es el de clusters instalados en grupos dentro de departamentos en facultades de universidades que se intercomunican. Si bien los firewalls de cada cluster se pueden *adaptar* a las exigencias de interconexión para cómputo intercluster, normalmente no se puede hacer lo mismo con los respectivos firewalls de las redes de los departamentos, de las facultades y de las universidades involucradas.

Desde una perspectiva general y por su propia definición, VPN provee una forma de comunicar las computadoras en dos o más redes locales como si estuvieran en una única red local. Esto en sí mismo puede ser considerado útil para cómputo paralelo intercluster. Sin embargo, avanzando en el nivel de detalle de las característcas de cómputo y comunicaciones de las aplicaciones paralelas, también provee una forma de "encauzamiento" o multiplexación de todas las comunicaciones intercluster a través de una única interconexión física entre los clusters. Es decir: aunque haya varias máquinas de uno de los clusters transfiriendo datos con varias máquinas de otro cluster, todas las transferencias de datos se llevarán a cabo utlizando una única interconexión, que además es conocida a priori y por lo tanto puede ser mantenida a nivel de seguridad de los firewalls.

---

* Investigador Asistente CICPBA

[1] PLA: sigla de Parallel Linear Algebra

2. Instalación del Software

Todo el software puede se puede obtener de su home page, http://openvpn.net, donde también están los links para obtener los archivos de fuentes y paquetes rpm para su instalación tanto en Windows como en Linux, y se puede acceder a algunas dependencias como openssl, lzo y pam.

Las indicaciones de instalación que se darán aquí corresponden sólo a Linux, los detalles para hacerlo en Windows, pueden consultarse en el manual de OpenVPN o en OpenVPN-HOWTO también en la página indicada.

El ejecutable OpenVPN puede ser instalado indistintamente sobre las máquinas que efectuarán las tareas de servidor o de clientes, pues tal ejecutable provee ambas funciones. De hecho, lo único que cambia en cada máquina (sevidor o cliente) es la configuración, dado que el software (ejecutable) es exactamente el mismo.

2.1. Usando (y/o Creando) RPM

Si se tiene una distribución de Linux que soporta paquetes RPMs (SuSE, Fedora, Red Hat,etc.) es el mecanismo de instalación que se recomienda pues genera scripts de inicio automático. El método más sencillo consiste en encontrar un paquete adecuado para la distribución utilizada y una vez que se tiene el rpm se instala de la manera convencional

rpm -ivh openvpn[detalles].rpm

o actualizando una instalación existente

rpm -Uvh openvpn[detalles].rpm

Esto creará un directorio /etc/openvpn, los scripts de inicio automático, y algunos archivos útiles para la configuración posterior del *servicio* en el directorio /usr/share/doc/openvpn-2.0.2/sample-config-files/ y en el directorio /usr/share/doc/openvpn-2.0.2/easy-rsa/.

En caso de no tener un RPM binario a utilizar directamente, se puede construir el paquete binario RPM, que como toda operación con RPMs tiene sus propias dependencias. Aparte de las dependencias de openVPN mencionadas previamente:

- openssl
- lzo
- pam

la construcción del rpm necesitará también

- openssl-devel
- lzo-devel
- pam-devel

con versiones adecuadas de acuerdo a la distribución de linux correspondiente. La creación propiamente dicha a partir del archivo de fuente comprimido, se realiza de la siguiente manera:

rpmbuild -tb openvpn[versión].tar.gz

lo que genera el archivo rpm en /usr/src/redhat/RPMS/i386/openvpn-[detalles].rpm que es suficiente para los ejecutables pero no para la etapa de configuración, por lo que no es recomendable eliminar el RPM que contiene los fuentes.

2.2. Instalación en RH 8 y en RH 9

No fue posible encontrar un .rpm con los binarios para OpenVPN en Linux RedHat 8 ni RedHat 9, por lo que es necesario instalar el software desde los .rpm con los fuentes. En el caso de la instalación en RedHat 9, los pasos que se siguieron fueron:

1) Instalación de lzo y lzo-devel:

rpm -i lzo-1.08-4.0.rh9.rf.i386.rpm

rpm -i lzo-devel-1.08-fr2.i386.rpm

2) Generación del paquete de OpenVPN propiamente dicho a partir del .tar.gz con los fuentes:

rpmbuild -tb openvpn-2.0.2.tar.gz

que genera el paquete en
/usr/src/redhat/RPMS/i386/openvpn-2.0.2-1.i386.rpm

3) Instalación del paquete de OpenVPN a partir del RPM con los binarios generado:
rpm -i /usr/src/redhat/RPMS/i386/openvpn-2.0.2-1.i386.rpm

En este punto, se está en las condiciones que se describieron previamente, es decir con un directorio /etc/openvpn, los scripts de inicio automático del *servicio* OpenVPN y algunos archivos útiles para la configuración posterior del *servicio* en el directorio /usr/share/doc/openvpn-2.0.2/sample-config-files/ y en el directorio /usr/share/doc/openvpn-2.0.2/easy-rsa/.

Luego, se intentó reutilizar este paquete en RH 8 y se instala sin problemas. Sin embargo, cuando se intenta ejecutar, los binarios no funcionan porque fueron creados para RH 9 y, por lo tanto, las bibliotecas de carga dinámica de RH 8 son anteriores a las que necesitan los binarios generados en RH 9. Finalmente, se generó nuevamente el .rpm a partir de los fuentes, pero ahora para RH 8. En realidad, convenía generar el .rpm en RH 8 y seguramente habría sido posible utilizarlo directamente en RH 9 sin inconvenientes. Los pasos de la generación del .rpm en RH 8 son similares a los que se siguieron en RH 9:

1) Instalación de lzo y lzo-devel:
rpm -i lzo-1.08-3.0.rh8.dag.i386.rpm
rpm -i lzo-devel-1.08-fr1.i386.rpm

2) Generación del paquete de OpenVPN propiamente dicho a partir del .tar.gz con los fuentes:
rpmbuild -tb openvpn-2.0.2.tar.gz
que genera el paquete en
/usr/src/redhat/RPMS/i386/openvpn-2.0.2-1.i386.rpm

3) Instalación del paquete de OpenVPN a partir del RPM con los binarios generado:
rpm -i --force /usr/src/redhat/RPMS/i386/openvpn-2.0.2-1.i386.rpm
Se debió usar la opción --force porque, *por alguna razón*, no se pudo desinstalar la versión instalada antes (la generada en RH 9) y, por supuesto, rpm -i "solo" no funcionaba dado que ya había una versión de OpenVPN instalada.

2.3. Instalación sin paquete RPM

También es posible instalar OpenVPN sobre Linux utilizando el método *universal* de ./configure, primero expandiendo el archivo comprimido .tar.gz con:
tar xfz openvpn-[versión].tar.gz
y cambiando al directorio extraído (cd openvpn-[versión]), tipeando:
./configure
make
make install
No se eligió este método porque solamente genera los binarios, no genera los scripts de inicio del *servicio*. Por otro lado, de esta manera también quedan "instalados" los fuentes, algo que no sucede en el caso de generar el .rpm a partir del .tar.gz que *contiene* los fuentes.

3. Configuración: *Servidor y Cliente*

En la sección anterior en realidad se explicaron las diferentes formas en que se pueden obtener los binarios de OpenVPN y dos ejemplos concretos de esa tarea en Linux RH 8 y Linux RH 9. Como en muchos otros casos y en particular con servicios relacionados con interconexiones sobre Internet y/o TCP/IP, se deben configurar los servicios o, más específicamente el software a utilizar para tener operativo el servicio en las computadoras a utilizar. Una vez más la mayor parte de la información que se incluye en esta sección no hace más que repetir y/o *adaptar* la información existente en el sitio *oficial* de OpenVPN.

El primer paso para *construir* una configuración de OpenVPN es establecer una infraestructura de clave pública (PKI), que consiste de:

- Un certificado separado (también conocido como clave pública) y una clave privada para el servidor y cada cliente.

- Un certificado/clave de autoridad maestro (CA), que es usado para firmar cada uno de los certificados de clientes y servidor.

OpenVPN soporta autenticación bidireccional basada en certificados, lo que significa que el cliente debe autenticar el certificado del servidor y el servidor debe autenticar el certificado del cliente antes de establecer una confianza mutua.

Siguiendo con el proceso de configuración, una vez que se tienen los certificados y claves, se deben editar los archivos de configuración propiamente dichos para arrancar el servicio de OpenVPN tanto *del lado del servidor* como *del lado del cliente*. Por lo tanto, en las subsecciones que siguen se describirán estos dos pasos generales: generación de claves y certificados y edición de los archivos de configuración.

3.1. Generación de Claves y Certificados

Tal como se explicó antes, la instalación de los binarios implica también la "instalación" de archivos útiles para la configuración *posterior* en los directorios /usr/share/doc/openvpn-2.0.2/sample-config-files/ y /usr/share/doc/openvpn-2.0.2/easy-rsa/. El primero de éstos que se utilizará es el easy-rsa, que normalmente se copia a /etc/openvpn para trabajar en él y dejar el contenido *definitivo* que utilizarán los binarios del servicio provisto por OpenVPN. Por otro lado, esto evita que una futura actualización de OpenVPN sobreescriba las modificaciones hechas. Por lo tanto, lo primero que se hace es copiar el directorio completo:

```
cp -R easy-rsa /etc/openvpn
```

Usualmente, todo lo referente a la generación de claves y certificados se lleva a cabo en la computadora que será el servidor.

Generando el certificado/clave maestro. La inicialización se lleva a cabo con (en el directorio en el que se copió, /etc/openvpn/easy-rsa/)

```
. ./vars
./clean-all
./build-ca
```

Si vars no es *ejecutable*, se debería hacer antes

```
chmod 755 vars
```

(no es útil sh vars porque, justamente, define variables que se usan en los demás comandos).

El comando build-ca construirá el certificado de autoridad invocando el comando interactivo de openssl, del cual se incluye aquí una muestra de su salida por pantalla cuando se utilizó la computadora con RH 9 que será la *servidora* para OpenVPN:

```
easy-rsa # ./build-ca
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'ca.key'
---
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
---
Country Name (2 letter code) [KG]:ar
State or Province Name (full name) [NA]:ba
Locality Name (eg, city) [BISHKEK]:lp
Organization Name (eg, company) [OpenVPN-TEST]:intercluster
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:lidipar86
Email Address [me@myhost.mydomain]:
```

Los parámetros anteriores pueden ser obviados, salvo el correspondiente a Common Name que es el único que debe ser entrado explícitamente.

Generando certificado y clave para el servidor. Un certificado y clave privados para el servidor se generarán con:

```
./build-key-server server
```

También en este caso, la mayoría de los parámetros que pide pueden ser obviados, salvo otra vez Common Name en el que habrá que entrar "server" y además dos preguntas deben contestarse afirmativamente:

"Sign the certificate? [y/n]" y

"1 out of 1 certificate requests certified, commit? [y/n]:" y

En el caso de la instalación hecha en RH 9, las respuestas fueron las mismas que las dadas para el comando anterior para las mismas preguntas.

Generando certificado y clave para clientes. Si se supone que hay 3 clientes para usar, se hace en el servidor:

./build-key cliente1

./build-key cliente2

./build-key cliente3

Para cada cliente se debe incluir el Common Name apropiado, por ejemplo "cliente1", "cliente2" y "cliente3" en este caso. Se debe recordar que hay que usar siempre un único Common Name para cada cliente. En el caso de la instalación realizada, en la máquina con RH 9 y hostname lidipar86, se hizo

./build-key lidipar14

dado que la máquina lidipar14 será la computadora *cliente* de lidipar86. Toda la información requerida por los comandos anteriores se ingresó con los mismos datos, excepto en Common Name, para el que se ingresó lidipar14.

Generando parámetros Diffie Hellman. Deben ser generados para el servidor con (recordar que todo esto se está haciendo en el servidor):

./build-dh

lo que dará una salida:

```
easy-rsa # ./build-dh
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
.....+.....
.....+.....+.....+.....
.....
```

El paso final en la generación de claves consiste en copiar todos los archivos a las máquinas que los necesiten, de acuerdo con la siguiente tabla:

Nombre	Computadora/s	Propósito	Permisos
ca.crt	servidor+todos los clientes	certificado maestro de Root	-rw-r--r--
ca.key	sólo servidor	clave maestra de root	-rw-------
dh{n}.pem	sólo servidor	parámetros Diffie Hellman	-rw-r--r--
server.crt	sólo servidor	certificado de servidor	-rw-r--r--
server.key	sólo servidor	clave del servidor	-rw-------
cliente1.crt	sólo cliente1	certificado de cliente1	-rw-r--r--
cliente1.key	sólo cliente1	clave de cliente1	-rw-------
cliente2.crt	sólo cliente2	certificado de cliente2	-rw-r--r--
cliente2.key	sólo cliente2	clave de cliente1	-rw-------
cliente3.crt	sólo cliente3	certificado de cliente3	-rw-r--r--
cliente3.key	sólo cliente3	clave de cliente1	-rw-------

Tabla 1: Distribución de Archivos.

De acuerdo con la instalación realizada en lidipar86, que será el *servidor*, con RH 9, el contenido del directorio /etc/openvpn relacionado con los archivos de claves y certificados es

```
-rw-r--r--    1 root     root         1172 sep 21 17:21 ca.crt
-rw-r--r--    1 root     root          245 sep 21 17:19 dh1024.pem
drwxr-xr-x    5 root     root         4096 sep 20 18:57 easy-rsa
-rw-r--r--    1 root     root         3537 sep 21 17:21 server.crt
-rw-------    1 root     root          887 sep 21 17:21 server.key
```

De acuerdo con la instalación realizada en lidipar14, que será el *cliente*, con RH 8, el contenido del directorio /etc/openvpn relacionado con los archivos de claves y certificados es

```
-rw-r--r--    1 root     root         1172 Sep 21 15:25 ca.crt
-rw-r--r--    1 root     root         3437 Sep 21 15:24 lidipar14.crt
-rw-------    1 root     root          887 Sep 21 15:24 lidipar14.key
```

Toda esta tarea se realiza en el servidor y luego se distribuyen los archivos en las demás computadoras, que serían los clientes.

3.2. Edición/Adaptación de Archivos de Configuración

Como se explicó antes, en el directorio /usr/share/doc/openvpn-2.0.2/sample-config-files/ se pueden encontrar archivos útiles para la configuración. Más específicamente, se tienen ejemplos comentados de los archivos

server.conf
client.conf

que *solamente* se deben adaptar para el caso específico del servidor y de los clientes. Normalmente, éste es el último paso de la configuración y a partir de aquí se llega a obtener un OpenVPN completamente operativo. Se presentan a continuación estos archivos, modificados/editados a partir de server.conf y client.conf incluidos dentro de la distribución .rpm de OpenVPN. Los detalles propios corresponden a la instalación en las máquinas mencionadas antes: lidipar86 (*servidor*) y lidipar14 (*cliente*).

Server.conf:

```
########################################################
# Sample OpenVPN 2.0 config file for                   #
# multi-client server.                                 #
#                                                      #
# This file is for the server side                     #
# of a many-clients <-> one-server                     #
# OpenVPN configuration.                               #
#                                                      #
# OpenVPN also supports                                #
# single-machine <-> single-machine                    #
# configurations (See the Examples page                #
# on the web site for more info).                      #
#                                                      #
# This config should work on Windows                   #
# or Linux/BSD systems.  Remember on                   #
# Windows to quote pathnames and use                   #
# double backslashes, e.g.:                            #
# "C:\\Program Files\\OpenVPN\\config\\foo.key" #
#                                                      #
# Comments are preceded with '#' or ';'                #
########################################################

# Which local IP address should OpenVPN
# listen on? (optional)
local 163.10.22.86

# Which TCP/UDP port should OpenVPN listen on?
# If you want to run multiple OpenVPN instances
# on the same machine, use a different port
# number for each one.  You will need to
# open up this port on your firewall.
port 1194

# TCP or UDP server?
proto tcp
;proto udp

# "dev tun" will create a routed IP tunnel,
```

```
# "dev tap" will create an ethernet tunnel.
# Use "dev tap" if you are ethernet bridging.
# If you want to control access policies
# over the VPN, you must create firewall
# rules for the the TUN/TAP interface.
# On non-Windows systems, you can give
# an explicit unit number, such as tun0.
# On Windows, use "dev-node" for this.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel if you
# have more than one.  On XP SP2 or higher,
# you may need to selectively disable the
# Windows firewall for the TAP adapter.
# Non-Windows systems usually don't need this.
;dev-node MyTap

# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key).  Each client
# and the server must have their own cert and
# key file.  The server and all clients will
# use the same ca file.
#
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys.  Remember to use
# a unique Common Name for the server
# and each of the client certificates.
#
# Any X509 key management system can be used.
# OpenVPN can also use a PKCS #12 formatted key file
# (see "pkcs12" directive in man page).
ca /etc/openvpn/ca.crt
cert /etc/openvpn/server.crt
key /etc/openvpn/server.key  # This file should be kept secret

# Diffie hellman parameters.
# Generate your own with:
#   openssl dhparam -out dh1024.pem 1024
# Substitute 2048 for 1024 if you are using
# 2048 bit keys.
dh /etc/openvpn/dh1024.pem

# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.8.0.0 255.255.255.0

# Maintain a record of client <-> virtual IP address
```

```
# associations in this file.  If OpenVPN goes down or
# is restarted, reconnecting clients can be assigned
# the same virtual IP address from the pool that was
# previously assigned.
;ifconfig-pool-persist ipp.txt

# Configure server mode for ethernet bridging.
# You must first use your OS's bridging capability
# to bridge the TAP interface with the ethernet
# NIC interface.  Then you must manually set the
# IP/netmask on the bridge interface, here we
# assume 10.8.0.4/255.255.255.0.  Finally we
# must set aside an IP range in this subnet
# (start=10.8.0.50 end=10.8.0.100) to allocate
# to connecting clients.  Leave this line commented
# out unless you are ethernet bridging.
;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100

# Push routes to the client to allow it
# to reach other private subnets behind
# the server.  Remember that these
# private subnets will also need
# to know to route the OpenVPN client
# address pool (10.8.0.0/255.255.255.0)
# back to the OpenVPN server.
;push "route 192.168.10.0 255.255.255.0"
;push "route 192.168.20.0 255.255.255.0"

# To assign specific IP addresses to specific
# clients or if a connecting client has a private
# subnet behind it that should also have VPN access,
# use the subdirectory "ccd" for client-specific
# configuration files (see man page for more info).

# EXAMPLE: Suppose the client
# having the certificate common name "Thelonious"
# also has a small subnet behind his connecting
# machine, such as 192.168.40.128/255.255.255.248.
# First, uncomment out these lines:
;client-config-dir ccd
;route 192.168.40.128 255.255.255.248
# Then create a file ccd/Thelonious with this line:
#   iroute 192.168.40.128 255.255.255.248
# This will allow Thelonious' private subnet to
# access the VPN.  This example will only work
# if you are routing, not bridging, i.e. you are
# using "dev tun" and "server" directives.

# EXAMPLE: Suppose you want to give
# Thelonious a fixed VPN IP address of 10.9.0.1.
# First uncomment out these lines:
;client-config-dir ccd
;route 10.9.0.0 255.255.255.252
# Then add this line to ccd/Thelonious:
#   ifconfig-push 10.9.0.1 10.9.0.2

# Suppose that you want to enable different
# firewall access policies for different groups
```

```
# of clients.  There are two methods:
# (1) Run multiple OpenVPN daemons, one for each
#     group, and firewall the TUN/TAP interface
#     for each group/daemon appropriately.
# (2) (Advanced) Create a script to dynamically
#     modify the firewall in response to access
#     from different clients.  See man
#     page for more info on learn-address script.
;learn-address ./script

# If enabled, this directive will configure
# all clients to redirect their default
# network gateway through the VPN, causing
# all IP traffic such as web browsing and
# and DNS lookups to go through the VPN
# (The OpenVPN server machine may need to NAT
# the TUN/TAP interface to the internet in
# order for this to work properly).
# CAVEAT: May break client's network config if
# client's local DHCP server packets get routed
# through the tunnel.  Solution: make sure
# client's local DHCP server is reachable via
# a more specific route than the default route
# of 0.0.0.0/0.0.0.0.
;push "redirect-gateway"

# Certain Windows-specific network settings
# can be pushed to clients, such as DNS
# or WINS server addresses.  CAVEAT:
# http://openvpn.net/faq.html#dhcpcaveats
;push "dhcp-option DNS 10.8.0.1"
;push "dhcp-option WINS 10.8.0.1"

# Uncomment this directive to allow different
# clients to be able to "see" each other.
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
;client-to-client

# Uncomment this directive if multiple clients
# might connect with the same certificate/key
# files or common names.  This is recommended
# only for testing purposes.  For production use,
# each client should have its own certificate/key
# pair.
#
# IF YOU HAVE NOT GENERATED INDIVIDUAL
# CERTIFICATE/KEY PAIRS FOR EACH CLIENT,
# EACH HAVING ITS OWN UNIQUE "COMMON NAME",
# UNCOMMENT THIS LINE OUT.
;duplicate-cn

# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
```

```
# Ping every 10 seconds, assume that remote
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120

# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
#
# Generate with:
#   openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
# on the server and '1' on the clients.
;tls-auth ta.key 0 # This file is secret

# Select a cryptographic cipher.
# This config item must be copied to
# the client config file as well.
;cipher BF-CBC        # Blowfish (default)
;cipher AES-128-CBC   # AES
;cipher DES-EDE3-CBC  # Triple-DES

# Enable compression on the VPN link.
# If you enable it here, you must also
# enable it in the client config file.
;comp-lzo

# The maximum number of concurrently connected
# clients we want to allow.
;max-clients 100

# It's a good idea to reduce the OpenVPN
# daemon's privileges after initialization.
#
# You can uncomment this out on
# non-Windows systems.
;user nobody
;group nobody

# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun

# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log

# By default, log messages will go to the syslog (or
# on Windows, if running as a service, they will go to
# the "\Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
```

```
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it.  Use one
# or the other (but not both).
log         openvpn.log
;log-append  openvpn.log

# Set the appropriate level of log
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
verb 7

# Silence repeating messages.  At most 20
# sequential messages of the same message
# category will be output to the log.
;mute 20
```

client.conf:

```
################################################
# Sample client-side OpenVPN 2.0 config file #
# for connecting to multi-client server.     #
#                                            #
# This configuration can be used by multiple #
# clients, however each client should have   #
# its own cert and key files.                #
#                                            #
# On Windows, you might want to rename this  #
# file so it has a .ovpn extension           #
################################################

# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client

# Use the same setting as you are using on
# the server.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel
# if you have more than one.  On XP SP2,
# you may need to disable the firewall
# for the TAP adapter.
;dev-node MyTap

# Are we connecting to a TCP or
# UDP server?  Use the same setting as
# on the server.
proto tcp
```

```
;proto udp

# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote 163.10.22.86 1194
;remote my-server-2 1194

# Choose a random host from the remote
# list for load-balancing.  Otherwise
# try hosts in the order specified.
;remote-random

# Keep trying indefinitely to resolve the
# host name of the OpenVPN server.  Very useful
# on machines which are not permanently connected
# to the internet such as laptops.
;resolv-retry infinite

# Most clients don't need to bind to
# a specific local port number.
nobind

# Downgrade privileges after initialization (non-Windows only)
;user nobody
;group nobody

# Try to preserve some state across restarts.
persist-key
persist-tun

# If you are connecting through an
# HTTP proxy to reach the actual OpenVPN
# server, put the proxy server/IP and
# port number here.  See the man page
# if your proxy server requires
# authentication.
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]

# Wireless networks often produce a lot
# of duplicate packets.  Set this flag
# to silence duplicate packet warnings.
;mute-replay-warnings

# SSL/TLS parms.
# See the server config file for more
# description.  It's best to use
# a separate .crt/.key file pair
# for each client.  A single ca
# file can be used for all clients.
ca /etc/openvpn/ca.crt
cert /etc/openvpn/lidipar14.crt
key /etc/openvpn/lidipar14.key

# Verify server certificate by checking
# that the certicate has the nsCertType
# field set to "server".  This is an
```

```
# important precaution to protect against
# a potential attack discussed here:
#  http://openvpn.net/howto.html#mitm
#
# To use this feature, you will need to generate
# your server certificates with the nsCertType
# field set to "server".  The build-key-server
# script in the easy-rsa folder will do this.
;ns-cert-type server

# If a tls-auth key is used on the server
# then every client must also have the key.
;tls-auth ta.key 1

# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
;cipher x

# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
;comp-lzo

# Set log file verbosity.
verb 7

# Silence repeating messages
;mute 20
```

En el caso del cliente, lo que cambió (o se aseguró que fuera de una manera en particular) fue básicamente:

proto tcp
remote 163.10.22.86
;resolv-retry infinite
;comp-lzo
los paths para .ca .crt y .key

En el caso del servidor, lo que se aseguró/modificó a partir del erchivo de ejemplo que viene con la distribución fue:

local 163.10.22.86
proto tcp
;comp-lzo
server 10.8.0.0 255.255.255.0

3.3. Arranque de OpnVPN, Reglas de Firewall y tun

Si la instalación se realizó desde un paquete rpm, OpenVPN se cargará como un servicio desde el inicio, escribiendo archivos .log de inicio y status en las posiciones especificadas en la configuración. También se puede iniciar el servidor desde la línea de comandos haciendo

openvpn server.conf

que es muy recomendado en la primera prueba del servicio hasta que se pueda confirmar que funciona correctamente. A partir de allí sí conviene que arranque automáticamente junto con los demás servicios del sistema operativo/red. Si no hay problemas de arranque, el comando anterior dará una salida en pantalla semejante a

```
Sun Feb 6 20:46:38 2005 OpenVPN 2.0_rc12 i386-Red Hat 8.0-linux [SSL] [LZO] [EPOLL]
                        built on Feb
Sun Feb 6 20:46:38 2005 Diffie-Hellman initialized with 1024 bit key
Sun Feb 6 20:46:38 2005 TLS-Auth MTU parms [ L:1542 D:138 EF:38 EB:0 ET:0 EL:0 ]
Sun Feb 6 20:46:38 2005 TUN/TAP device tun1 opened
```

```
Sun Feb 6 20:46:38 2005 /sbin/ifconfig tun1 10.8.0.1 pointopoint 10.8.0.2 mtu 1500
Sun Feb 6 20:46:38 2005 /sbin/route add -net 10.8.0.0 netmask 255.255.255.0 gw
                        10.8.0.2
Sun Feb 6 20:46:38 2005 Data Channel MTU parms [ L:1542 D:1450 EF:42 EB:23 ET:0 EL:0
                        AF:3/
Sun Feb 6 20:46:38 2005 UDPv4 link local (bound): [undef]:1194
Sun Feb 6 20:46:38 2005 UDPv4 link remote: [undef]
Sun Feb 6 20:46:38 2005 MULTI: multi_init called, r=256 v=256
Sun Feb 6 20:46:38 2005 IFCONFIG POOL: base=10.8.0.4 size=62
Sun Feb 6 20:46:38 2005 IFCONFIG POOL LIST
Sun Feb 6 20:46:38 2005 Initialization Sequence Completed
```

De la misma manera, se puede inicializar el cliente desde la línea de comandos con (y, nuevamente, se recomienda, hasta que se asegure que el servicio funciona correctamente)

openvpn client.conf

que dará una salida similar a la anterior del servidor, finalizada si no hay ningún problema con "Initialization Sequence Completed".

Sin embargo, el arranque satisfactorio de OpenVPN tanto en el lado del cliente como del lado del servidor no implica que se tenga una VPN *realmente operativa*. De hecho, lo más probable es que no sea así dado que normalmente hay firewalls intermedios cuyas reglas se deben *adaptar* para el tráfico de la VPN. Más específicamente, se deben incluir las reglas que dejen acceder el tráfico VPN en el lado del servidor (hacia el port del servidor). Esto es que el puerto de acceso del servidor esté abierto desde el firewall, que en este caso sería el 1194 TCP y que en linux estén habilitados el IP forwarding y la interfaz tun o tap según corresponda, en caso de que no sea así, IP forwarding se habilita con

echo 1 > /proc/sys/net/ipv4/ip_forward

y la carga del módulo de la interfaz tun (que a partir de RH9 está automáticamente cargada, pero no en RH 8) con

modprobe tun

Ahora, se puede comprobar si efectivamente funciona la VPN con un ping a través del VPN desde el cliente a la dirección *virtual* del servidor

ping 10.8.0.1

y si esto funciona bien, se puede suponer que la VPN también está funcionando correctamente (al menos esto es lo que afirma taxativamente el howto correspondiente).

4. VPN como LAN (*Realmente Operativa*)

Aunque el comando ping funcione, rápidamente se puede comprobar que no se está en las mismas condiciones que en una LAN. Más específicamente, se puede notar que los servicios de red más comunes, como el ssh, rsh, rcp, scp, etc. realmente no funcionan. Normalmente se comprueba de la *peor* manera: simplemente no responden. Esto se puede solucionar rápida y sencilla, porque se debe a los firewalls, normalmente del lado del servidor, así que hay que incluir las reglas de firewall apropiadas para que el tráfico "extra" (además del específico de VPN) se mantenga y no se elimine por las reglas del firewall definidas a priori para eliminar todo lo que no se incluya de manera explícita. Aunque puede parecer complejo no lo es, porque todo el tráfico de servicios de red que se utilicen a través de VPN (con las direcciones de la VPN) en realidad se lleva a cabo con la interfase tun. Por lo tanto, todo lo que se debe agregar son las reglas asociadas a tun, tal como se indica en el howto:

###Para habilitar entrada y salida VPN
Allow TUN interface connections to OpenVPN server
/sbin/iptables -A INPUT -i tun+ -j ACCEPT
Allow TUN interface connections to be forwarded through other interfaces
/sbin/iptables -A FORWARD -i tun+ -j ACCEPT

y a partir de ahora sí se tiene una VPN que funciona *realmente* como una LAN... o eso parece...

5. Clusters de/para Cómputo Paralelo

En el caso de los clusters instalados y utilizados para cómputo paralelo es usual que aparezca al menos un problema más, que es propio de estas plataformas de cómputo paralelo aunque no es exclusivo. Una de las

formas más sencillas de proveer SSI (Single System Image) al menos en el ámbito de *filesystem* es utilizar NFS (Network File System). Además, también es común usar el NAT (NEtwork Address Translation) del firewall de una máquina en el cluster para salir al exterior en el caso de usar Internet, por ejemplo (por el sólo hecho de navegar). En todos los casos, estos servicios se consideran básicos e independientes de la instalación de VPN. El caso usual es tener una red intranet (10.10... ó 192.168...) y una computadora con dos placas de red y con firewall instalado de forma tal que la *segunda* placa de red tenga IP pública y, de hecho, se lleve a cabo la el ruteo de IP con esta computadora (normalmente denominada "servidor"). También es bastante usual que esta computadora tenga el sistema de archivos a "montar" y compartir con las demás del cluster vía NFS.

Ni bien se instala y se pone en marcha la VPN, toda la configuración y forma de funcionamiento que se describe antes deja de ser *operativo*. De hecho, si uno de los directorios que se utilizan vía NFS es el /home, *automáticamente* los usuarios pierden el acceso a sus directorios personales. La razón básicamente tiene relación con la forma en que se lleva a cabo el ruteo de paquetes a partir del arranque de OpenVPN. Este cambio es hecho directamente por OpenVPN y no parece sencillo cambiarlo (al menos no aparece de manera inmediata en la documentación disponible). Aunque parece bastante complicado no lo es, dado que lo único que se necesita mantener es el ruteo que "estaba" antes del arranque de OpenVPN. La secuencia de comandos que sigue es un ejemplo de lo que se utiliza para evitar esta *interferencia* de OpenVPN con el ruteo previo al arranque del servicio:

```
modprobe tun
openvpn /etc/openvpn/client.conf &
sleep 1
route add -host 10.10.0.254 eth0
```

(10.10.0.254 es el NFS server en este ejemplo).

Y de esta forma no solamente se tiene una VPN con los servicios de rsh y demás necesarios para cómputo paralelo funcionando correctamente, sino que también siguen funcionando los servicios *anteriores y básicos* como NFS.