



UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea triennale in *Ingegneria Informatica e dell'Automazione*

Implementazione di OpenVPN su router 4G per site-to-site vpn in ambiente CG-NAT

Study and configuration of a site-to-site VPN in CG-NAT environment

Relatore:

Prof. Ennio Gambi

Correlatore:

Ing. Adelmo De Santis

Laureando:

Simone Viozzi

Prefazione

#TODO da riscrivere

Nell'ambito del mio percorso universitario ho avuto modo di approfondire le tematiche relative al mondo delle reti e del networking, a tal proposito grazie alla possibilità offerta dal Dipartimento di Ingegneria dell'Informazione, dal Prof. Ennio Gambi e dall'Ing. Adelmo De Santis ho conseguito con successo la certificazione "*HUAWEI HCIA Routing and Switching*". Successivamente, grazie alle competenze acquisite, ho collaborato con alcuni miei colleghi per progettare e realizzare una implementazione di una VPN site-to-site attraverso una connessione radiomobile per conto dell'azienda Esse-ti S.r.l.

In questo elaborato verranno esposte le principali fasi del progetto realizzato, ponendo un particolare focus sulle problematiche iniziali affrontate e all'architettura di rete nel cui ambito è stata realizzata la comunicazione tramite un canale sicuro.

Indice

Prefazione	ii
Indice	iii
Elenco delle figure	v
1 Introduzione	1
1.1 Intro su ip/tcp	1
1.2 openvpn	1
1.3 openwrt	1
2 Overview dell'architettura e delle componenti utilizzate	2
2.1 Obbiettivo da ottenere	2
2.2 Specifiche dei componenti	3
2.2.1 Esse-ti 4G.Router	4
2.2.2 VPS OVHCloud	5
2.2.3 Host domotico	6
2.2.4 Macchina del cliente	6
3 Configurazione del server	7
3.1 Overview della configurazione e prerequisiti	7
3.2 Creazione della Public key infrastructure Certificate Authority (PKI CA)	8
3.3 Configurazione della PKI di OpenVPN	10
3.4 Firma del certificato opnevpn dalla CA	11
3.5 Generazione della <i>tls-crypt pre-shared key</i>	12
3.6 Generazione delle chiavi per i clients	13
3.7 Creazione del file di configurazione del server OpenVPN	14
3.8 Configurazioni sulla network stack del server openvpn	14

3.9	Configurazione del firewall	15
3.9.1	Configurazione del NAT	15
3.9.2	Configurazione del packet forwarding	16
3.9.3	Conclusione della configurazione del firewall	17
3.10	Avvio del server OpenVPN	17
3.11	Script per la creazione delle configurazioni dei client	18
3.12	Test della configurazione	20
4	Configurazione Router	22
4.1	Overview della configurazione	22
4.2	Introduzione a Luci	22
4.3	Accesso ssh al Router	23
4.4	Creazione della configurazione e test	24
4.5	Abilitazione del Client-to-Client nel server OpenVPN	25
4.6	Assegnazione ip statico al Router	27
	Bibliografia	28

Elenco delle figure

2.1	Schema concettuale dell'obiettivo da raggiungere. [1]	2
2.2	Schema concettuale dell'architettura che si dovrà implementare. [1]	3
2.3	Topologia virtuale. [1]	3
2.4	4G.Router	4
3.1	Configurazione di partenza e di obiettivo per questo capitolo. [1]	7
4.2	Interfaccia grafica LuCI	23
4.3	Configurazione della VPN tramite LuCI	25

Capitolo 1

Introduzione

#TODO da scrivere da 0

1.1 Intro su ip/tcp

1.2 openvpn

1.3 openwrt

Capitolo 2

Overview dell'architettura e delle componenti utilizzate

2.1 Obbiettivo da ottenere

In una collaborazione tra il Dipartimento di Ingegneria dell'Informazione e l'azienda **Esse-ti S.R.L.** ci è stato esposto un progetto che consiste nel:

- fornire a dei clienti un router 4G, su cui possono essere connessi vari dispositivi, ad es. di tipo domotico.
- rendere questi dispositivi accessibili ai clienti attraverso internet

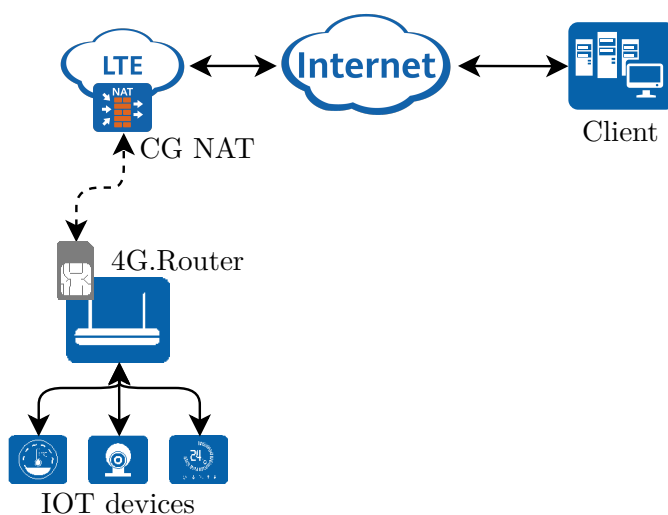


Figura 2.1: Schema concettuale dell'obiettivo da raggiungere. [1]

Data la presenza del CG-NAT si vede subito che non è realizzabile a meno che il cliente non abbia un'IP pubblico e la sua macchina venga configurata opportunamente. Questo però non è possibile nel caso generale, quindi per risolvere efficacemente questa topologia si deve necessariamente

introdurre una terza macchina provvista di IP pubblico e che funga da ponte tra il 4G.Router e il cliente.

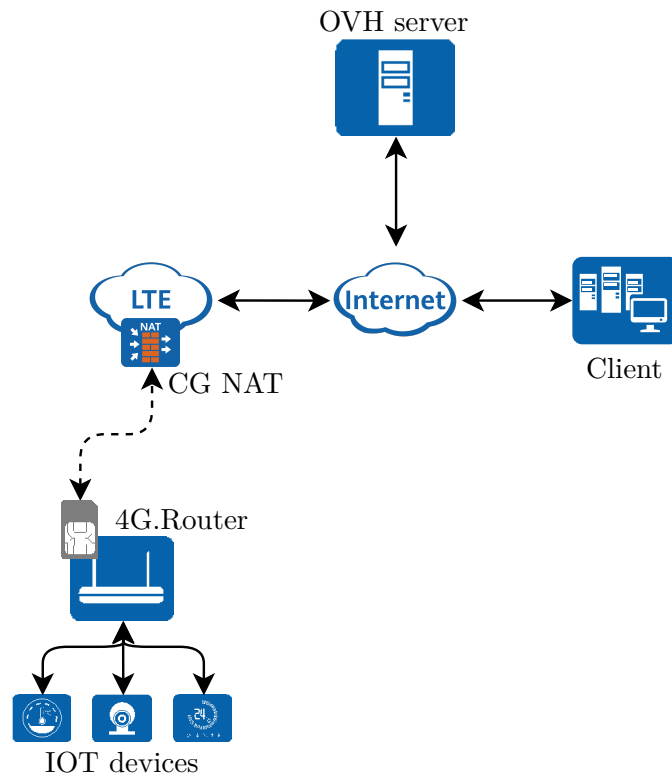


Figura 2.2: Schema concettuale dell'architettura che si dovrà implementare. [1]

In questo modo si può configurare una VPN sul server OVH e connettersi sia il 4G.Router che la macchina del cliente. In questo modo l'unica configurazione che il cliente dovrà fare è l'installazione di un cliente VPN, ciò è il minimo possibile di configurazione.

La configurazione virtuale vista dal 4G.Router e dai clienti sarà quindi:

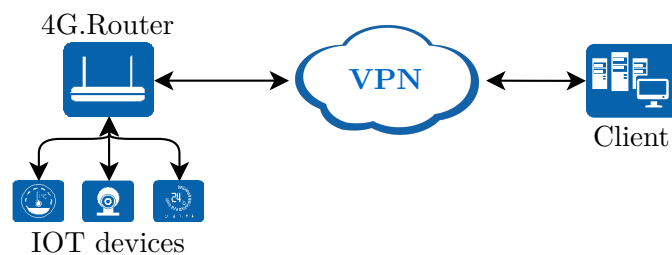


Figura 2.3: Topologia virtuale. [1]

2.2 Specifiche dei componenti

i componenti necessari sono:

- Esse-ti 4G.Router
- Server

- Host domotico
- Macchina del cliente

vediamo le caratteristiche minime che i componenti dovranno avere:

2.2.1 Esse-ti 4G.Router

Ci è stato fornito dall'azienda Esse-ti, consiste in un gateway 4G con funzionalità di router. Le specifiche complete possono essere trovate sul sito del produttore ([link](#))



Figura 2.4: 4G.Router

Per l'implementazione di questa architettura sono necessarie solo un sub-set delle specifiche:

- Access Point wireless per offrire connettività Internet Wi-Fi a dispositivi wireless
- Client Dynamic DNS per consentire all'utente di raggiungere da remoto, tramite Internet, il router stesso e tutti i dispositivi connessi via Wi-Fi o porta LAN
- Gateway telefonico per consentire l'invio e la ricezione di chiamate attraverso la rete 4G LTE/UMTS/GSM a telefoni fissi, combinatori o altri dispositivi telefonici collegati all'ingresso FXS

Presenta inoltre come sistema operativo una versione personalizzata di OpenWrt.

La configurazione del dispositivo puo' essere fatta sia da terminale, entrando in ssh, sia da interfaccia web:



(a) Schermata di autenticazione



(b) Grafico del traffico



(c) Schermata con stato riassuntivo

L'interfaccia web e' una versione personalizzata di [Luci](#).

Per semplicita' si fara' riferimento all'*Esse-ti 4G.Router* chiamandolo semplicemente router.

2.2.2 VPS OVHCloud

La VPS ha il solo vincolo di dover avere un'ip pubblico e una connessione a internet abbastanza veloce. Dovra' infatti sopportare un traffico simmetrico in upload / download.

Per la realizzazione della topologia e' stata selezionata una macchina una VPS del provider OVHCloud, con le seguenti caratteristiche:

- 2 core virtuali
- 4Gb di memoria ram
- 80Gb di storage NVMe
- 500Mbps simmetrici di banda
- ipv4 pubblico
- Ubuntu 16.04

Per semplicita' si fara' riferimento alla *VPS OVHCloud* come server.

2.2.3 Host domotico

Per effettuare le varie operazioni di testing e' stato aggiunta *raspberry pi* che ha svolto il ruolo di "host domotico". Sono state fatti test con ping e iperf per testare che tutta la topologia sia stata configurata correttamente.

2.2.4 Macchina del cliente

Deve poter essere una qualunque macchina, non ha vincoli di sistema operativo

Necessita di avere il client openvpn installato:

- con sistema operativo Windows si deve scaricare l'eseguibile dal [sito ufficiale](#)
- su linux e' sufficiente cercare nei repository ufficiali della distribuzione che si sta usando.

Capitolo 3

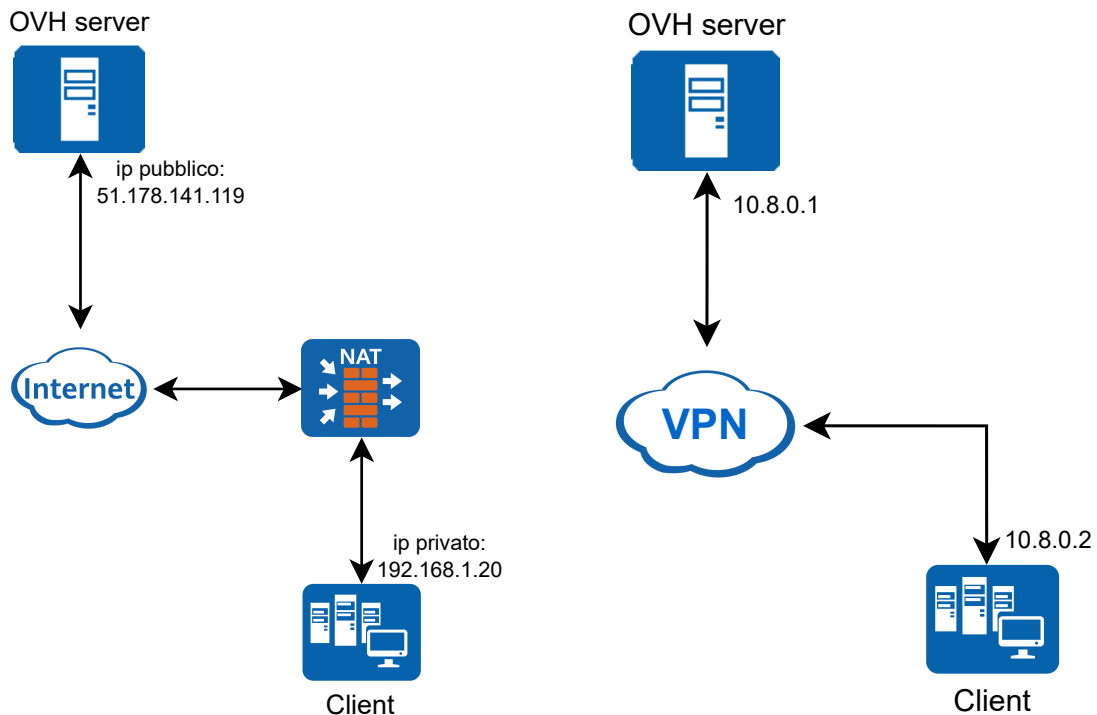
Configurazione del server

3.1 Overview della configurazione e prerequisiti

In questa sezione andremo a installare e configurare OpenVPN server sulla VPS di OVHCloud.

Supponiamo di partire da una configurazione di base, che contiene solo il server openvpn ed un generico client, che supponiamo sia sotto un NAT.

Supponiamo inoltre che l'ip pubblico del *server* sia `51.178.141.119`, si avra' quindi una configurazione come in figura [3.1a](#).



(a) Configurazione di partenza per questo capitolo. (b) Configurazione virtuale da raggiungere.

Figura 3.1: Configurazione di partenza e di obiettivo per questo capitolo. [\[1\]](#)

Per instaurare una comunicazione bidirezionale tra il server e il client, si dovra' configurare

oppportunamente una rete vpn la cui configurazione e' rappresentata in figura [3.1b](#).

I pacchetti necessari sono `openvpn` ed `easy-rsa`, che possono essere installati con:

```
Server
1 $ sudo apt-get update
2 $ sudo apt-get install -y openvpn easy-rsa
```

E' inoltre necessario avere un editor di testo, ad es. `nano` o `vim`

3.2 Creazione della Public key infrastructure Certificate Authority (PKI CA)

La CA puo' essere configurata sulla stessa macchina dove e' stato installato openvpn, ma cio' e' sconsigliato per motivi di sicurezza, supponiamo quindi di usare un secondo server chiamato *server CA*

La utility `easy-rsa` mette a disposizione il comando `make-cadir`, che permette di creare una cartella pronta ad ospitare la Certificate Authority.

Andiamo quindi a crearla, nella home ad esempio:

```
Server CA
1 $ mkdir ~/openvpn-ca
2 $ ln -s /usr/share/easy-rsa/* ~/openvpn-ca/
3 $ chmod 700 /home/ubuntu/openvpn-ca/
4 $ cd openvpn-ca/
5 $ ./easyrsa init-pki
6
7 init-pki complete; you may now create a CA or requests.
8 Your newly created PKI dir is: /home/ubuntu/openvpn-ca/pki
9
10 $ la
11 easyrsa openssl-easyrsa.cnf pki vars.example x509-types
```

Ora si devono personalizzare le variabili `vars`, si puo' sia partire da un file vuoto oppure modificare `vars.example` per poi rinominarlo `vars`. Andiamo quindi a creare un nuovo file `vars`:

Server CA

```
1 $ vim vars
2 set_var EASYRSA_REQ_COUNTRY "IT"
3 set_var EASYRSA_REQ_PROVINCE "MC"
4 set_var EASYRSA_REQ_CITY "Recanati"
5 set_var EASYRSA_REQ_ORG "Esse-ti"
6 set_var EASYRSA_REQ_EMAIL "s.gasparrini@esse-ti.it"
7 set_var EASYRSA_REQ_OU "Esse-ti"
8 set_var EASYRSA_REQ_CN "openvpn-ca"
9
10 set_var EASYRSA_ALGO "ec"
11 set_var EASYRSA_DIGEST "sha512"
```

Le variabili nel primo blocco determinano i dati che poi verranno registrati nei certificati.

Le ultime 2 sono opzioni di sicurezza, in particolare si setta l'algoritmo di cifratura

A questo punto si deve lasciare il comando `build-ca` per costruire la CA:

Server CA

```
1 $ ./easyrsa build-ca
2
3 Note: using Easy-RSA configuration from: ./vars
4
5 Using SSL: openssl OpenSSL 1.1.1f 31 Mar 2020
6
7 Enter New CA Key Passphrase:
8 Re-Enter New CA Key Passphrase:
9 read EC key
10 writing EC key
11
12 You are about to be asked to enter information that will be incorporated
13 into your certificate request.
14 What you are about to enter is what is called a Distinguished Name or a DN.
15 There are quite a few fields but you can leave some blank
16 For some fields there will be a default value,
17 If you enter '.', the field will be left blank.
18 -----
19 Common Name (eg: your user, host, or server name) [Easy-RSA CA]:
20
21 CA creation complete and you may now import and sign cert requests.
22 Your new CA certificate file for publishing is at:
23 /home/ubuntu/openvpn-ca/pki/ca.crt
24
```

Eseguendo il comando verrà chiesto di inserire una passphrase, che verrà usata per criptare la chiave privata appena generata. Il secondo prompt è relativo al nome da dare alla certificazione, in questo caso è stato lasciato il valore di default `Easy-RSA CA`.

3.3 Configurazione della PKI di OpenVPN

Il procedimento e' simile al precedente, ma questa volta va eseguito sul *server*.

Creiamo quindi una cartella per ospitare la PKI, es `/openvpn-pki`, e linkiamo `easy-rsa`. Inoltre limitiamo i permessi all'utente non root che stiamo usando, in questo caso "ubuntu".

Server

```
1 $ mkdir ~/openvpn-pki
2 $ ln -s /usr/share/easy-rsa/* ~/openvpn-pki/
3 $ sudo chown ubuntu ~/openvpn-pki/
4 $ chmod 700 ~/openvpn-pki/
5 $ cd ~/openvpn-pki/
```

Andiamo a creare un file `vars`:

Server

```
1 $ vim vars
2 set_var EASYRSA_ALGO      "ec"
3 set_var EASYRSA_DIGEST    "sha512"
```

Concludiamo la creazione della PKI con il comando:

Server

```
1 $ ./easyrsa init-pki
2
3 Note: using Easy-RSA configuration from: ./vars
4
5 init-pki complete; you may now create a CA or requests.
6 Your newly created PKI dir is: /home/ubuntu/openvpn-pki/pki
7
```

A questo punto il server openvpn ha tutti i prerequisiti per creare una sua chiave privata e relativa *Certificate Signing Request*.

Come nome e' stato scelto "server":

Server

```
1 $ ./easysrsa gen-req server nopass
2
3 Note: using Easy-RSA configuration from: ./vars
4
5 Using SSL: openssl OpenSSL 1.1.1f 31 Mar 2020
6 Generating an EC private key
7 writing new private key to '/home/ubuntu/openvpn-pki/pki/private/server.key.438W2xM0g9'
8 -----
9 You are about to be asked to enter information that will be incorporated
10 into your certificate request.
11 What you are about to enter is what is called a Distinguished Name or a DN.
12 There are quite a few fields but you can leave some blank
13 For some fields there will be a default value,
14 If you enter '.', the field will be left blank.
15 -----
16 Common Name (eg: your user, host, or server name) [server]:
17
18 Keypair and certificate request completed. Your files are:
19 req: /home/ubuntu/openvpn-pki/pki/reqs/server.req
20 key: /home/ubuntu/openvpn-pki/pki/private/server.key
21
```

La chiave `server.key` va copiata nell'apposita cartella.

Server

```
1 $ sudo cp /home/ubuntu/openvpn-pki/pki/private/server.key /etc/openvpn/server/
```

Il secondo file creato, `server.req`, corrisponde ad una *Certificate Signing Request (CSR)* che va firmata e validata dalla CA. In questo modo ogni client che si fida della CA si fiderà di conseguenza del server OpenVPN

3.4 Firma del certificato openvpn dalla CA

Dobbiamo quindi copiare il file `server.req` nel *server CA*, possiamo qualunque metodo purché sia sicuro, ad esempio con `scp`:

Server

```
1 $ scp -3 ubuntu@openvpn_server:/home/ubuntu/openvpn-pki/pki/reqs/server.req
   ↪ ubuntu@ca_server:/tmp
```

Dobbiamo quindi spostarci sul server CA e importare la *certificate request* e firmarlo:

Server CA

```
1 $ cd ~/openvpn-ca
2 $ ./easyrsa import-req /tmp/server.req server
3 $ ./easyrsa sign-req server server
4 Using configuration from /home/ubuntu/openvpn-ca/pki/safessl-easyrsa.cnf
5 Check that the request matches the signature
6 Signature ok
7 The Subject's Distinguished Name is as follows
8 commonName      :ASN.1 12:'server'
9 Certificate is to be certified until Mar 11 15:50:45 2025 GMT (1080 days)
10
11 Write out database with 1 new entries
12 Data Base Updated
```

Verrà creato un file in `/openvpn-ca/pki/issued` chiamato `server.crt` che conterrà la chiave pubblica che verrà usata dal server openvpn e inoltre la firma della CA.

Ora si devono copiare i file `ca.crt` e `server.crt` dal *server CA* al *server OpenVPN*:

Server

```
1 $ scp -3 ubuntu@ca_server:/home/ubuntu/openvpn-ca/pki/issued/server.crt
   ↪ ubuntu@openvpn_server:/tmp
2 $ scp -3 ubuntu@ca_server:/home/ubuntu/openvpn-ca/pki/ca.crt ubuntu@openvpn_server:/tmp
```

Possiamo quindi tornare sul *server OpenVPN* e copiare i 2 file da `/tmp` a `/etc/openvpn/server`:

Server

```
1 $ sudo cp /tmp/server.crt /etc/openvpn/server
2 $ sudo cp /tmp/ca.crt /etc/openvpn/server
```

3.5 Generazione della *tls-crypt pre-shared key*

Per aumentare ulteriormente la sicurezza del nostro *server OpenVPN* possiamo creare un'ulteriore chiave, che consiste in una chiave *preshared* che verrà inserita in tutte le configurazioni e serve a offuscare il certificato in fase di validazione. Quindi in caso di attacco si dovrà conoscere anche questa chiave.

La creazione va fatta sul *server OpenVPN*:

Server

```
1 $ cd ~/openvpn-pki/
2 $ openvpn --genkey --secret ta.key
```

il file generato `ta.key` dovrà essere copiato nella directory del server openvpn:

Server

```
1 $ sudo cp ta.key /etc/openvpn/server
```

3.6 Generazione delle chiavi per i clients

Creiamo una cartella nella *home* che ospiterà le chiavi dei *client* e le configurazioni openvpn:

Server

```
1 $ mkdir -p ~/client-configs/keys
2 $ chmod -R 700 ~/client-configs
```

Creiamo quindi un certificato per un *client*:

Server

```
1 $ cd ~/openvpn-pki/
2 $ ./easysrsa gen-req client1 nopass
```

Ora dobbiamo copiare `client1.key` nella directory appena creata, e `client1.req` va copiato nel server CA per essere firmato:

Server

```
1 $ cp pki/private/client1.key ~/client-configs/keys/
2 $ scp -3 ubuntu@openvpn_server:/home/ubuntu/openvpn-pki/pki/reqs/client1.req
   ↪ ubuntu@ca_server:/tmp
```

Dobbiamo quindi spostarci sul server CA e importare la *certificate request* e firmarla:

Server CA

```
1 $ cd ~/openvpn-ca
2 $ ./easysrsa import-req /tmp/client1.req client1
3 $ ./easysrsa sign-req client client1
4 Using configuration from /home/ubuntu/openvpn-ca/pki/safessl-easysrsa.cnf
5 Check that the request matches the signature
6 Signature ok
7 The Subject's Distinguished Name is as follows
8 commonName          :ASN.1 12:'client1'
9 Certificate is to be certified until Mar 16 13:15:09 2025 GMT (1080 days)
10
11 Write out database with 1 new entries
12 Data Base Updated
```

Per poi ricopiare dal server CA al server openvpn il certificato firmato:

Server

```
1 $ scp -3 ubuntu@ca_server:/home/ubuntu/openvpn-ca/pki/issued/client1.crt  
   ↪ ubuntu@openvpn_server:/tmp
```

Quindi ci dobbiamo spostare sul server OpenVPN e copiare le chiavi nella cartella `client-configs/keys`, in modo da prepararla per la creazione delle configurazioni OpenVPN. E' necessario inoltre cambiare i permessi dei file rendendoli accessibili all'utente Ubuntu:

Server

```
1 $ cp /tmp/client1.crt ~/client-configs/keys/  
2 $ cp ~/openvpn-pki/ta.key ~/client-configs/keys/  
3 $ sudo cp /etc/openvpn/server/ca.crt ~/client-configs/keys/  
4 $ sudo chown ubuntu:ubuntu ~/client-configs/keys/*
```

3.7 Creazione del file di configurazione del server OpenVPN

Il server openvpn viene configurato attraverso `/etc/openvpn/server/server.conf`, per non partire da una configurazione vuota si puo' copiare la configurazione di esempio:

Server

```
1 $ cd /etc/openvpn/server/  
2 $ sudo wget "https://raw.githubusercontent.com/OpenVPN/openvpn/  
3     master/sample/sample-config-files/server.conf"
```

Dobbiamo quindi modificare il file e cambiare alcune configurazioni, per facilitare la lettura sara' incluso il numero riga modificato:

Server

```
1 $ sudo vim server.conf  
2 85 dh none           # non sonos stati usati i parametri Diffie-Hellman  
3 92 topology subnet   # topologia raccomandata  
4 244 ;tls-auth ta.key 0 # This file is secret  
5 245 tls-crypt ta.key  # selezione della preshared key  
6 253 cipher AES-256-GCM # selezione della cifratura scelta  
7 275 user nobody       # utente che eseguirà il server openvpn, in modo da  
8                        # restringere i permessi  
9 276 group nogroup     # stassa cosa per il gruppo  
10 318 auth sha256       # selezione del metodo di autenticazione
```

3.8 Configurazioni sulla network stack del server openvpn

Per abilitare l'*ip forwarding* si dovrà modificare il file `/etc/sysctl.conf`, il comando successivo serve a ricaricare le configurazioni dai file:

Server

```
1 $ sudo vim /etc/sysctl.conf
2 69 net.ipv4.ip_forward = 1
3 $ sudo sysctl -p
4 net.ipv4.ip_forward = 1
```

3.9 Configurazione del firewall

Sulla VPS scelta e' presente il firewall *firewalld*, ma per una piu' semplice configurazione e' consigliato di disattivarlo e installare *ufw*:

Server

```
1 $ sudo systemctl mask firewalld
2 $ sudo systemctl stop firewalld
3 $ sudo apt-get install ufw
4 $ sudo ufw allow ssh
5 Rule added
6 Rule added (v6)
7 $ sudo ufw enable
```

E' importantissimo ricordarsi di consentire l'SSH prima di abilitare il firewall, altrimenti si perdera' l'accesso alla VPS.

3.9.1 Configurazione del NAT

Per far si che i pacchetti provenienti dalla *VPN* entrino nella network stack del *server* si deve aggiungere una regola di *NAT* nel firewall. Per farlo si deve conoscere quale e' l'interfaccia di rete del *server*, cioe' quella che ha come ip il suo ip pubblico:

Server

```
1 $ ip addr
2 [...]
3 2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen
   ↪ 1000
4     link/ether a6:23:5f:48:ba:de brd ff:ff:ff:ff:ff:ff
5     inet 51.178.141.119/20 brd 51.178.141.255 scope global dynamic ens3
6         valid_lft 1857sec preferred_lft 1857sec
7     inet6 fe80::23:bfff:ac24:aace/64 scope link
8         valid_lft forever preferred_lft forever
9 [...]
```

In questo caso il nome dell'interfaccia di rete e' *ens3*, possiamo quindi procedere con la configurazione del firewall, si andra' a modificare il file `/etc/ufw/before.rules` e aggiungere la regola di NAT:

```
Server
1 $ sudo vim /etc/ufw/before.rules
2 # ## rules.before
3 # ## Rules that should be run before the ufw command line added rules. Custom
4 # rules should be added to one of these chains:
5 # ufw-before-input
6 # ufw-before-output
7 # ufw-before-forward
8 #
9
10 # START OPENVPN RULES
11 # NAT table rules
12 *nat
13 :POSTROUTING ACCEPT [0:0]
14 # Allow traffic from OpenVPN client to ens3
15 -A POSTROUTING -s 10.8.0.0/24 -o ens3 -j MASQUERADE
16 COMMIT
17 # END OPENVPN RULES
18
19
20 # Don't delete these required lines, otherwise there will be errors
21 *filter
22 . . .
```

Nella modifica del file si deve stare attenti a inserire la nuova regola in cima al file e sotto i commenti iniziali, e' inoltre importante inserire i commenti nella regola.

3.9.2 Configurazione del packet forwarding

Next, you need to tell UFW to allow forwarded packets by default as well. To do this, open the `/etc/default/ufw` file:

```
Server
1 sudo nano /etc/default/ufw
```

Inside, find the `DEFAULT_FORWARD_POLICY` directive and change the value from DROP to ACCEPT: `/etc/default/ufw`

```
Server
1 DEFAULT_FORWARD_POLICY="ACCEPT"
```

Save and close the file when you are finished.

3.9.3 Conclusione della configurazione del firewall

Per concludere la configurazione si deve abilitare la porta relativa alla vpn, in questo caso `1194` , e riavviare il firewall:

```
Server
1 $ sudo ufw allow 1194/udp
2 $ sudo ufw reload
3 $ sudo ufw status
4 Status: active
5 To Action From
6 --
7 22 ALLOW Anywhere
8 1194/udp ALLOW Anywhere
9 22 (v6) ALLOW Anywhere (v6)
10 1194/udp (v6) ALLOW Anywhere (v6)
```

3.10 Avvio del server OpenVPN

Ora che la configurazione del server e' in una situazione stabile possiamo avviarlo:

Server

```
1 $ sudo systemctl enable openvpn-server@server.service
2 $ sudo systemctl start openvpn-server@server.service
3 $ sudo systemctl status openvpn-server@server.service
4 • openvpn-server@server.service - OpenVPN service for server
5   Loaded: loaded (/usr/lib/systemd/system/openvpn-server@.service; enabled; vendor
        ↪ preset: disabled)
6   Active: active (running) since Mon 2022-04-18 13:08:44 CEST; 4h 22min ago
7     Docs: man:openvpn(8)
8           https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
9           https://community.openvpn.net/openvpn/wiki/HOWTO
10  Main PID: 436 (openvpn)
11    Status: "Initialization Sequence Completed"
12     Tasks: 1 (limit: 9488)
13    Memory: 4.8M
14     CPU: 199ms
15    CGroup: /system.slice/system-openvpn\x2dserver.slice/openvpn-server@server.service
16            └─436 /usr/bin/openvpn --status /run/openvpn-server/status-server.log
17              ↪ --status-version 2 --suppress-timestamps --config server.conf
18 Apr 18 13:08:44 server openvpn[436]: /sbin/ip addr add dev tun0 local 10.8.0.1 peer
19 ↪ 10.8.0.2
20 Apr 18 13:08:44 server openvpn[436]: /sbin/ip route add 10.8.0.0/24 via 10.8.0.2
21 Apr 18 13:08:44 server openvpn[436]: UDPv4 link local (bound): [AF_INET][undef]:1194
22 Apr 18 13:08:44 server openvpn[436]: UDPv4 link remote: [AF_UNSPEC]
23 Apr 18 13:08:44 server openvpn[436]: MULTI: multi_init called, r=256 v=256
24 Apr 18 13:08:44 server openvpn[436]: IFCONFIG POOL: base=10.8.0.4 size=62, ipv6=0
25 Apr 18 13:08:44 server openvpn[436]: IFCONFIG POOL LIST
26 Apr 18 13:08:44 server openvpn[436]: Initialization Sequence Completed
```

Il comando `systemctl enable` abilita il servizio per essere avviato all'avvio della macchina, mentre `systemctl start` lo avvia immediatamente. Con il comando `systemctl status` si può verificare lo stato del servizio, si vede che il servizio è *active (running)*.

3.11 Script per la creazione delle configurazioni dei client

Per facilitare la creazione dei file di configurazione dei client, `clientX.conf`, andremo a creare un apposito script bash. Per prima cosa si deve scaricare e personalizzare la configurazione base del client:

Server

```
1 $ cd ~/client-configs/
2 $ wget "https://raw.githubusercontent.com/OpenVPN/openvpn\
3     /master/sample/sample-config-files/client.conf" \
4     -O base.conf
5 $ vim base.conf
6 42  remote 51.178.141.119 1194      # va messo l'ip e la porta del server OpenVPN
7 88  ;ca ca.crt                    # non useremo i file esterni ma ingloberemo
8 89  ;cert client.crt              # questi file in un file direttamente nella
9 90  ;key client.key               # configurazione del client
10 108 ;tls-auth ta.key 1           # stessa cosa per la preshared key
11 116 cipher AES-256-GCM          # cifratura usata
12 117 auth SHA256                  # autenticazione usata
13 118 key-direction 1
14                                     # le seguenti righe sono delle direttive al dns dell'host
15 120 ; script-security 2
16 121 ; up /etc/openvpn/update-resolv-conf
17 122 ; down /etc/openvpn/update-resolv-conf
18
19 125 ; script-security 2
20 126 ; up /etc/openvpn/update-systemd-resolved
21 127 ; down /etc/openvpn/update-systemd-resolved
22 128 ; down-pre
23 129 ; dhcp-option DOMAIN-ROUTE .
```

Ora creiamo lo script bash `make_config.sh` :

Server

```
1 $ vim make_config.sh
2 #!/bin/bash
3
4 # First argument: Client identifier
5
6 KEY_DIR=~/.client-configs/keys
7 OUTPUT_DIR=~/.client-configs/files
8 BASE_CONFIG=~/.client-configs/base.conf
9
10 cat ${BASE_CONFIG} \
11     <(echo -e '<ca>' \
12     ${KEY_DIR}/ca.crt \
13     <(echo -e '</ca>\n<cert>' \
14     ${KEY_DIR}/${1}.crt \
15     <(echo -e '</cert>\n<key>' \
16     ${KEY_DIR}/${1}.key \
17     <(echo -e '</key>\n<tls-crypt>' \
18     ${KEY_DIR}/ta.key \
19     <(echo -e '</tls-crypt>' \
20     > ${OUTPUT_DIR}/${1}.ovpn
21 $ chmod 700 make_config.sh
```

Lo scopo di questo script e' di aggiungere al file `base.conf` il certificato della CA, `ca.crt` ,

il certificato e chiave relativi al client per cui si sta creando la configurazione, passato come argomento allo script, e la *preshared key*. Il tutto viene scritto in un file che ha lo stesso nome del *client* per cui si sta creando la configurazione ma `.conf`.

Quindi per creare la configurazione di *client 1*:

```
Server
1 $ ./make_config.sh client1
```

Nella cartella `client-configs/files/` si troverà il file di configurazione per il client `client1.ovpn`.

3.12 Test della configurazione

Ora che abbiamo un file di configurazione per il client, possiamo testare che la configurazione fino a questo punto sia corretta. Per farlo ci spostiamo su una macchina client, con SO Linux ad esempio, e si avvia il *client* con la configurazione creata al passo precedente:

```
Client
1 $ sudo openvpn --config client1.ovpn
2 [...]
3 Thu Apr 21 12:53:04 2022 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 256
  ↳ bit key
4 Thu Apr 21 12:53:04 2022 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 256
  ↳ bit key
5 Thu Apr 21 12:53:04 2022 ROUTE_GATEWAY 192.168.1.20/255.255.255.0 IFACE=eth0
  ↳ HWADDR=02:42:0a:00:04:03
6 Thu Apr 21 12:53:04 2022 /sbin/ip route add 10.8.0.1/32 via 10.8.0.2
7 Thu Apr 21 12:53:04 2022 WARNING: this configuration may cache passwords in memory -- use
  ↳ the auth-nocache option to prevent this
8 Thu Apr 21 12:53:04 2022 Initialization Sequence Completed
```

Se la configurazione fino a questo punto è corretta si avrà il messaggio `Initialization Sequence Completed`.

Nel *client* si avrà una nuova interfaccia di rete chiamata `tun0`, questa è l'interfaccia virtuale creata dalla vpn.

```
Client
1 2: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN
  ↳ group default qlen 500
2   link/none
3   inet 10.8.0.2/24 scope global tun0
4       valid_lft forever preferred_lft forever
```

Si può vedere come l'ip assegnato al *client* dalla vpn è `10.8.0.2`.

Per testare che la connessione sia instaurata correttamente si puo' usare la utility `ping`, ad esempio possiamo fare il ping dal *client* verso l'ip interno alla vpn del *server*:

Client

```
1 $ ping -c2 10.8.0.1
2 PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
3 64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=0.250 ms
4 64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.220 ms
```

Se nel frattempo si esegue la utility `tcpdump` sul server si potranno vedere i pacchetti *echo request* ed *echo reply*:

Server

```
1 $ sudo tcpdump
2 listening on tun0, link-type RAW (Raw IP), capture size 262144 bytes
3 13:11:12.018615 IP 10.8.0.2 > 10.8.0.1: ICMP echo request, id 11, seq 1, length 64
4 13:11:12.018640 IP 10.8.0.1 > 10.8.0.2: ICMP echo reply, id 11, seq 1, length 64
5 13:11:13.039993 IP 10.8.0.2 > 10.8.0.1: ICMP echo request, id 11, seq 2, length 64
6 13:11:13.040018 IP 10.8.0.1 > 10.8.0.2: ICMP echo reply, id 11, seq 2, length 64
```

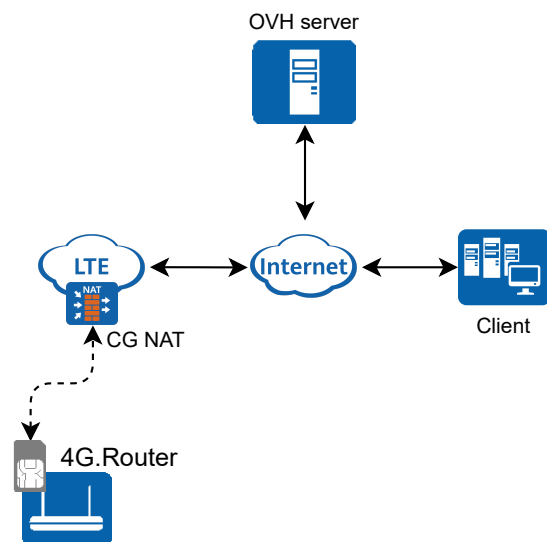
Si vede quindi che e' possibile una comunicazione bidirezionale tra *client*, `10.8.0.2`, e *server*, `10.8.0.1`.

Capitolo 4

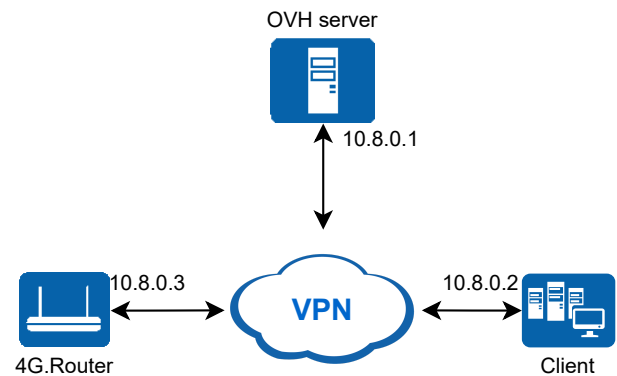
Configurazione Router

4.1 Overview della configurazione

In questo capitolo andremo a configurare il *router 4g* e connetterlo alla VPN.



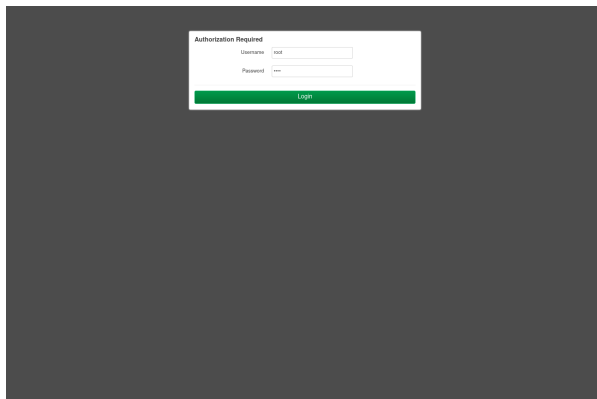
(a) Diagramma di stato del router



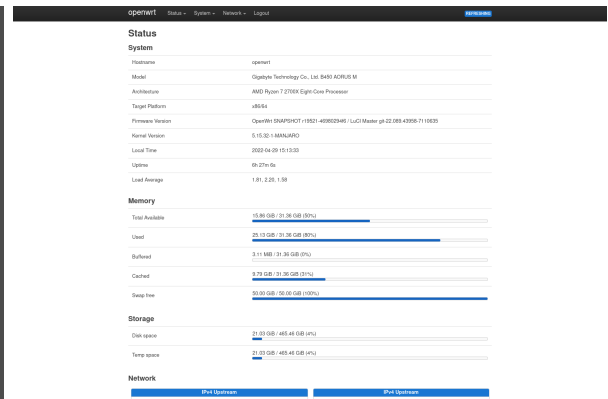
(b) Diagramma di stato del router

4.2 Introduzione a Luci

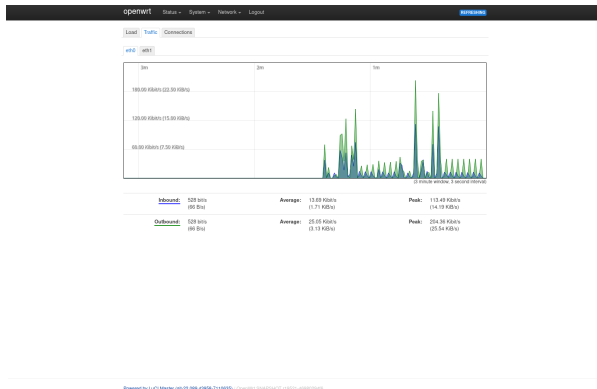
Luci è la web interface ufficiale di *openwrt*, dovrebbe essere già installata e raggiungibile, in caso contrario può essere installata e configurata seguendo la guida ufficiale di *OpenWrt* [2].



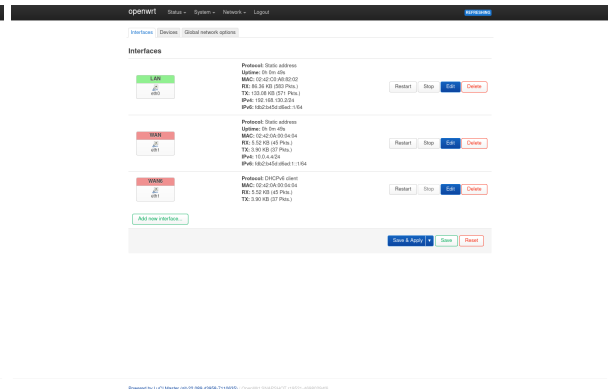
(a) Login page



(b) Status page



(c) Graphs page



(d) Interfaces page

Figura 4.2: Interfaccia grafica LuCI

Le credenziali di default sono **Username:root** **Password:root**, come mostrato in fig. 4.2a.

L'homepage, fig. 4.2b, mostra un riepilogo dello stato del router, ad esempio sono presenti: informazioni sull'hardware, informazioni sulla memoria e storage, sono presenti inoltre informazioni riassuntive sulle interfacce di rete e sul *DHCP*.

L'interfaccia e' estensiva e permette di configurare quasi ogni aspetto del funzionamento del router, compreso il firewall, il *DHCP*, i processi in esecuzione, etc.

4.3 Accesso ssh al Router

Oltre all'interfaccia grafica *LuCI* si puo' accedere al router tramite ssh.

Router 4g

```
1 BusyBox v1.35.0 (2022-04-24 21:09:51 UTC) built-in shell (ash)
2
3  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
4  | - |   |   |   |   |   |   |   |   |   |   |   |   |   |
5  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
6  |   | W I R E L E S S   F R E E D O M
7  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
8  OpenWrt SNAPSHOT, r19521-46980294f6
9  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
```

Non esiste un account utente quindi viene effettuato il login come root.

4.4 Creazione della configurazione e test

Si devono seguire gli step descritti in sezione 3.6, quindi creare la certificate request e firmarla nel *server CA*. Per poi usare lo script creato in sezione 3.11 per costruire il file di configurazione:

Server

```
1 $ ./make_config.sh router
```

Dopodiché si deve spostare il file `router.ovpn` dal *Server* al *Router 4g*, supponiamo di averlo copiato nella cartella `/configs`.

Di default non e' presente *OpenVPN* nel *Router 4g*, lo si puo' installare con:

Router 4g

```
1 $ opkg update
2 $ opkg install openvpn
3 $ opkg install luci-app-openvpn
```

Ora possiamo avviare il client openvpn:

Router 4g

```
1 $ openvpn --config /configs/router.ovpn
2 2022-04-29 17:26:37 OpenVPN 2.5.6 x86_64-openwrt-linux-gnu [SSL (mbed TLS)] [LZ4] [EPOLL]
   ↪ [MH/PKTINFO] [AEAD]
3 [...]
4 2022-04-29 17:26:37 VERIFY ECU OK
5 2022-04-29 17:26:37 VERIFY OK: depth=0, CN=server
6 2022-04-29 17:26:37 Control Channel: TLSv1.2, cipher
   ↪ TLS-ECDHE-RSA-WITH-AES-256-GCM-SHA384, 2048 bit key
7 2022-04-29 17:26:37 [server] Peer Connection Initiated with [AF_INET]10.0.4.2:1194
8 2022-04-29 17:26:37 net_addr_ptp_v4_add: 10.8.0.10 peer 10.8.0.9 dev tun0
9 2022-04-29 17:26:37 Initialization Sequence Completed
```

Se il file di configurazione e' stato creato correttamente si vedra' il messaggio **Initialization Sequence Completed**.

Comparira' inoltre l'interfaccia **tun0** a cui e' stato assegnato l'indirizzo **10.8.0.10**.

Per abilitare l'autostart di openvpn per il router si deve, per prima cosa, modificare il file **/etc/config/openvpn** in modo che faccia riferimento alla config corretta:

```
Router 4g
1 $ vim /etc/config/openvpn
2 20 option config /configs/router.ovpn
```

Ora possiamo abilitarla usando luci:

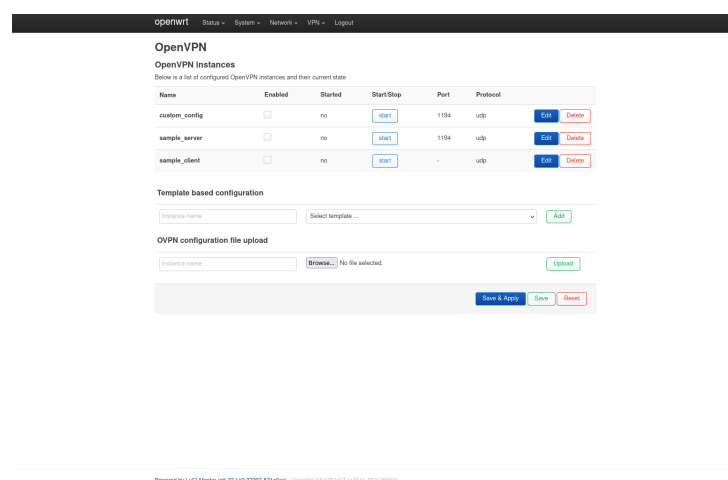


Figura 4.3: Configurazione della VPN tramite LuCI

Si deve mettere il check su *enabled* e premere start, per poi salvare le modifiche. In questo modo il router si connettera' automaticamente alla VPN anche se venisse riavviato.

4.5 Abilitazione del Client-to-Client nel server OpenVPN

In questo momento i client della VPN, *client1* e *router*, possono comunicare tra loro, ma lo fanno passando per la network stack del *server*. Infatti:

```
Router 4g
1 $ ping -c2 10.8.0.2 # client1
2 PING 10.8.0.2 (10.8.0.2): 56 data bytes
3 64 bytes from 10.8.0.2: seq=0 ttl=63 time=0.519 ms
4 64 bytes from 10.8.0.2: seq=1 ttl=63 time=0.501 ms
5
6 --- 10.8.0.2 ping statistics ---
7 2 packets transmitted, 2 packets received, 0% packet loss
8 round-trip min/avg/max = 0.501/0.510/0.519 ms
```

Dal server possiamo vedere i pacchetti con *tcpdump*:

```
Server
1 $ sudo tcpdump -i tun0
2 listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
3 16:20:50.791063 IP 10.8.0.3 > 10.8.0.2: ICMP echo request, id 1759, seq 0, length 64
4 16:20:50.791098 IP 10.8.0.3 > 10.8.0.2: ICMP echo request, id 1759, seq 0, length 64
5 16:20:50.791273 IP 10.8.0.2 > 10.8.0.3: ICMP echo reply, id 1759, seq 0, length 64
6 16:20:50.791285 IP 10.8.0.2 > 10.8.0.3: ICMP echo reply, id 1759, seq 0, length 64
7 16:20:51.791153 IP 10.8.0.3 > 10.8.0.2: ICMP echo request, id 1759, seq 1, length 64
8 16:20:51.791174 IP 10.8.0.3 > 10.8.0.2: ICMP echo request, id 1759, seq 1, length 64
9 16:20:51.791365 IP 10.8.0.2 > 10.8.0.3: ICMP echo reply, id 1759, seq 1, length 64
10 16:20:51.791374 IP 10.8.0.2 > 10.8.0.3: ICMP echo reply, id 1759, seq 1, length 64
```

Si vede che ogni richiesta viene duplicata, la prima e' in entrata sulla network stack del *server* e la seconda in uscita.

Per evitare questo traffico possiamo abilitare l'opzione `client-to-client` nel file di configurazione del server. In questo modo il layer *openvpn* effettuera' direttamente il forwarding tra i client della vpn [3].

```
Server
1 $ vim /etc/openvpn/server/server.conf
2 209 client-to-client
3 $ sudo systemctl restart openvpn-server@server.service
```

Possiamo quindi rieseguire gli stessi test fatti sopra:

```
Router 4g
1 $ ping -c2 10.8.0.2
2 PING 10.8.0.2 (10.8.0.2): 56 data bytes
3 64 bytes from 10.8.0.2: seq=0 ttl=64 time=0.351 ms
4 64 bytes from 10.8.0.2: seq=1 ttl=64 time=0.307 ms
5
6 --- 10.8.0.2 ping statistics ---
7 2 packets transmitted, 2 packets received, 0% packet loss
8 round-trip min/avg/max = 0.307/0.329/0.351 ms
```

Ma questa volta la network stack del *server* non vede nessun pacchetto:

```
Server
1 $ sudo tcpdump -i tun0
2 listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
```


4.6 Assegnazione ip statico al Router

Dato che non possiamo sapere a priori quanti client si connetteranno contemporaneamente alla VPN, per sapere sempre quale ip viene assegnato al router e' necessario assegnargliene uno statico. Questo viene fatto usando la `client-config-dir`.

```
Server
1 $ sudo mkdir /etc/openvpn/server/ccd # creo la client-config-dir
2 $ vim /etc/openvpn/server/server.conf # abilito l'opzione nella config del server
3 167 client-config-dir ccd
4 $ vim /etc/openvpn/server/ccd/router
5 ifconfig-push 10.8.0.254 255.255.255.0 # impongo l'ip per questo common name
6 $ sudo systemctl restart openvpn-server@server.service
```

Così facendo al router gli verrà sempre assegnato l'ip `10.8.0.254`, indipendentemente dall'ordine in cui gli host si connettono alla vpn. Ciò ci permette di sapere sempre e a priori qual'è l'ip del router.

Bibliografia

- [1] Huawei. Product image gallery. https://info.support.huawei.com/network/imagelib/getImagePartList?product_family=Router&product_type=Access%20Router%7CIOT%20Gateway&domain=&lang=en.
- [2] OpenWrt. Luci essentials. <https://openwrt.org/docs/guide-user/luci/luci.essentials>.
- [3] ServerFault. Descrizione openvpn client-to-client. <https://serverfault.com/a/738558/558773>.