UNIVERSITÀ DEGLI STUDI DI CAMERINO

SCUOLA DI SCIENZE E TECNOLOGIE

Corso di Laurea in Informatica Classe L-31

VPN con isolamento degli utenti e gestione avanzata della banda

LAUREANDO

Lorenzo Lapucci

RELATORE

Fausto Marcantoni

Anno Accademico 2022/2023

Abstract

La crescente consapevolezza sull'importanza della privacy online ha spinto sempre più utenti a ricorrere ai servizi di VPN come strumento per proteggere le proprie comunicazioni e dati personali. Tuttavia, molte soluzioni VPN attualmente disponibili sul mercato presentano limitazioni nell'isolamento degli utenti, nella gestione della banda e nelle prestazioni complessive. Questa tesi si propone di affrontare tali sfide attraverso la progettazione e l'implementazione di una VPN client-server, focalizzandosi su tre obiettivi principali: garantire l'isolamento degli utenti, gestire automaticamente la banda e fornire prestazioni elevate. Per raggiungere questi obiettivi, viene adottato il protocollo WireGuard, noto per la sua semplicità, velocità e sicurezza. La VPN sviluppata offre un'architettura che assicura l'isolamento completo degli utenti, utilizzando il firewall, insieme a tecniche di traffic shaping per ottimizzare l'utilizzo della banda disponibile. Grazie a WireGuard, la soluzione offre prestazioni elevate, minimizzando la latenza e massimizzando l'utilizzo della banda.

Sommario

1. Introduzione

1.1. Contesto e Motivazioni

Le azioni intraprese per ridurre il divario digitale sottolineano l'importanza fondamentale di internet per la partecipazione alla società dell'informazione[1][2]. Per la fruizione dei contenuti online si richiede la condivisione delle nostre informazioni. Anche se non si effettua il login con il proprio account, un sito web o applicazione potrebbe comunque profilare l'utente, spesso per scopi di statistica e marketing.

Il tracciamento di un utente può avvenire tramite diversi identificatori univoci che vengono raccolti per formare un'impronta digitale dell'utente. Queste informazioni possono comprendere, ad esempio, la risoluzione del monitor che si utilizza, il fuso orario impostato, i font installati nel sistema e l'indirizzo IP della connessione internet che si sta utilizzando[3][4].

Dalla raccolta e l'utilizzo di queste informazioni da parte dei servizi online, nasce un bacino di utenti preoccupati per la propria privacy. Questi utenti cercano strumenti per ridurre la quantità di informazioni personali che condividono, intenzionalmente o meno.

Tra questi strumenti vediamo un aumento nell'uso delle VPN (Virtual Private Network) da parte di più di un terzo degli utenti in internet[5], il cui scopo è "mascherare" l'indirizzo IP reale dell'utente accedendo a internet tramite indirizzi IP di server terzi; indirizzi che sono condivisi con più utenti e che quindi rendono più difficile tracciare il singolo utente.

Un server di un servizio VPN può quindi reindirizzare pacchetti di migliaia di utenti contemporaneamente e se non si pongono dei limiti e non si gestisce la banda in modo adeguato può portare a basse prestazioni della connessione internet.

Figura 1.1: Come una VPN cripta il traffico fino al servizio

L’indirizzo IP di una connessione può anche essere utilizzato per geolocalizzare l’utente, consentendo di bloccare determinati contenuti in alcuni stati o aree geografiche.
Non tutte le VPN offrono le stesse prestazioni e funzionalità, nasce quindi la necessità di un servizio VPN che permetta agli utenti di sfruttare la propria connessione in tutte le attività che non comprende solo la navigazione sui siti internet ma anche il download di contenuti e l’online gaming, che richiede una buona stabilità della connessione.

1.2. Obiettivi e Sfide Tecniche

Questa tesi si propone di sviluppare una VPN client-server con le seguenti caratteristiche:

- isolamento degli utenti;
- gestione avanzata della banda;
- prestazioni elevate.

1.2.1. Isolamento degli utenti

Lo stesso servizio di VPN consente a più utenti di connettersi e utilizzare lo stesso indirizzo IP. È essenziale impedire l’accesso alle reti private di altri utenti connessi allo stesso server ed evitare che possano comunicare direttamente.
L'utilizzo di firewall e tool avanzati come `iptables` possono aiutare durante l'implementazione di un sistema di isolamento.

1.2.2. Gestione avanzata della banda

In una situazione ideale, la banda totale del server corrisponde alla banda dedicata per utente moltiplicata per il numero di utenti connessi; per un server la banda rappresenta uno dei costi maggiori e si aggira a circa $8 per ogni Mb/s di larghezza di banda dedicata[6].
Dato l’elevato costo della connessione internet, operatori di telecomunicazione e servizi di VPN condividono la banda totale a loro disposizione con più utenti, partendo dall’idea che non tutti i clienti consumano contemporaneamente il 100% della banda a loro disposizione.

Per ridurre l’impatto della VPN al minimo e fornire una connessione stabile a tutti gli utenti si possono utilizzare delle tecniche di Traffic Shaping per gestire in tempo reale l’utilizzo di banda da parte di ogni utente, così da poter garantire una quantità minima di banda ed evitare situazioni classiche di una connessione instabile e satura come la perdita dei pacchetti[7].

1.2.3. Prestazioni elevate

Utilizzo di un protocollo VPN moderno ed efficiente per ridurre al minimo la latenza e consentire a connessioni con velocità elevate di sfruttare al massimo la banda messa a disposizione dal server.

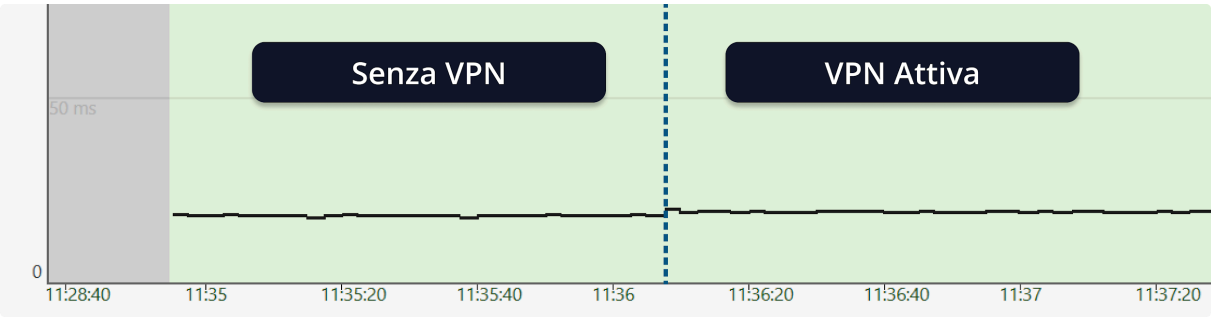

Figura 1.2: Grafico della latenza con e senza VPN.

Il raggiungimento di questi obiettivi permetterà di creare una VPN che sia un'alternativa valida alle soluzioni attualmente disponibili sul mercato.

2. Protocolli VPN

2.1. Perché una nuova VPN

Esistono già diverse VPN sul mercato, che utilizzano i protocolli più diffusi[8]. Tuttavia, molti di questi servizi, che spesso offrono alcune delle funzionalità desiderate come l'isolamento degli utenti e sono efficaci nel nascondere l'indirizzo IP dei client, non offrono altre funzionalità come la gestione avanzata della banda ma aggregano molti più utenti su un singolo server a scapito delle prestazioni del servizio.

L'obiettivo di questa tesi è di sviluppare una nuova VPN che offra tutte queste funzionalità, utilizzando i protocolli VPN più recenti e performanti.

2.2. Introduzione ai protocolli VPN

I protocolli VPN definiscono le procedure utilizzate per stabilire, gestire e proteggere le connessioni tra client e server. Esistono diversi protocolli VPN, ognuno con caratteristiche uniche in termini di sicurezza e prestazioni. Alcuni dei protocolli VPN più comuni includono:

- OpenVPN;
- WireGuard;
- L2TP/IPsec.

2.2.1. OpenVPN

OpenVPN è un protocollo open-source che permette di autenticarsi tramite l'utilizzo di chiavi private condivise, certificati o credenziali. OpenVPN basa la cifratura della connessione su OpenSSL tramite il protocollo TLS (Transport Layer Security), lo stesso utilizzato per la crittografia dei dati su siti web tramite HTTPS (HyperText Transfer Protocol over Secure Socket Layer). Può essere configurato per funzionare tramite TCP (Transmission Control Protocol) o UDP (User Datagram Protocol)[9].

2.2.2. WireGuard

Meno diffuso di OpenVPN, WireGuard è un protocollo più recente che si distingue per la semplicità, la velocità e la sicurezza. Le dimensioni ridotte della codebase aiutano la community open-source nel verificare la presenza di bug di sicurezza. Il codice, inoltre, è stato formalmente verificato dai ricercatori Benjamin Dowling e Kenneth G. Paterson[10].

WireGuard stabilisce dei tunnel sicuri con i client tramite la chiave pubblica, unica per ogni utente collegato. Superato il riconoscimento tramite chiave pubblica il protocollo

procede a condividere le chiavi di cifratura nella fase di handshake, che avviene circa ogni due minuti[11]. A differenza di OpenVPN, nell'implementazione standard, supporta solo UDP.

2.2.3. L2TP/IPsec

L2TP/IPsec (Layer 2 Tunneling Protocol / IP Security) rappresenta una buona soluzione nelle situazioni in cui le performance non sono fondamentali e i client non supportano altri protocolli come OpenVPN o WireGuard. Questo protocollo incapsula i dati due volte, prima tramite L2TP che crea un tunnel client-server e dopo con IPsec che crea una connessione sicura tramite la crittografia dei dati[12][13]. A differenza di WireGuard, L2TP/IPsec, come OpenVPN, mantiene una connessione attiva tra client e server[14].

2.3. Confronto tra VPN

Nel mercato dei servizi VPN esistono diverse opzioni, ciascuna con le proprie caratteristiche e punti di forza; in questo confronto ci concentriamo su quattro macro-aree: modello di business, prestazioni, facilità d'uso e manutenzione delle VPN più diffuse sul mercato, confrontandole con la nostra proposta per evidenziarne i punti di forza e, eventualmente, le limitazioni, utilizzando una valutazione da 1 a 10, dove 10 indica le migliori prestazioni, la massima facilità d'uso e la difficoltà minima di manutenzione, mentre il modello di business si distingue tra "Gratuita" (se è possibile installare il software sul proprio server o se il servizio offre un piano gratuito) e "A pagamento" (se il servizio è gestito e disponibile tramite pagamento). Un servizio VPN può offrire contemporaneamente un piano gratuito e a pagamento.

VPN	Modello di Business	Prestazioni [15]	Facilità d'uso	Manutenzione	Disponibilità del software
OpenVPN	Gratuita / A pagamento	8	6	6	Open Source
WireGuard	Gratuita	10	7	7	Open Source
L2TP/IPsec	Gratuita	7	5	6	Open Source
NordVPN	A pagamento	8	9	9	Proprietario
Surfshark	A pagamento	8	10	9	Proprietario
FastestVPN	A pagamento	6	10	9	Proprietario
Proton VPN	Gratuita / A pagamento	8	9	9	Proprietario
Mullvad VPN	A pagamento	8	10	9	Proprietario
SaferVPN (interrotto)	A pagamento	6	9	7	Proprietario
VPN Proposta	A pagamento	9	8	9	Open Source

OpenVPN e WireGuard sono un’ottima scelta per gli utenti esperti che desiderano la massima flessibilità. Tuttavia, sono più difficili da installare, configurare e mantenere rispetto ad altri servizi VPN che sono gestiti e offrono client per la connessione estremamente semplificati.

Tra le alternative, teniamo in considerazione servizi a pagamento già esistenti, che spesso aggregano un numero elevato di utenti su un solo server per poter condividere un indirizzo IP con più utenti, aumentando la difficoltà di profilazione. Questo può portare a un degrado della connessione se la banda messa a disposizione dal server non è sufficiente per il numero di utenti connessi[16][17].

Per offrire una buona combinazione di prestazioni, semplicità d’uso, sicurezza e minima manutenzione da parte dell’utente finale, il servizio di VPN proposto utilizzerà il protocollo WireGuard, configurato per l’isolamento degli utenti e gestione della banda per ogni utente.

3. WireGuard in dettaglio

3.1. Architettura del protocollo

WireGuard è un protocollo relativamente nuovo, progettato per essere semplice da implementare, sicuro e performante[18]. WireGuard è implementato direttamente nel kernel Linux, per evitare le molteplici copie dei pacchetti con conseguente calo delle performance, ma mette a disposizione anche implementazioni alternative per compatibilità con altri sistemi[19].

Il protocollo non mantiene una connessione attiva (Connection-less Protocol), le chiavi simmetriche, usate per crittografare i pacchetti, vengono scambiate ogni due minuti circa; questo scambio avviene in base al tempo, e non in base ai dati precedenti, perché il protocollo è progettato per gestire in modo adeguato la perdita di pacchetti (packet loss). La fase di handshake utilizza una coda di pacchetti per ogni host/client, questo per minimizzare il packet loss e mantenere le altre connessioni attive performanti[11].

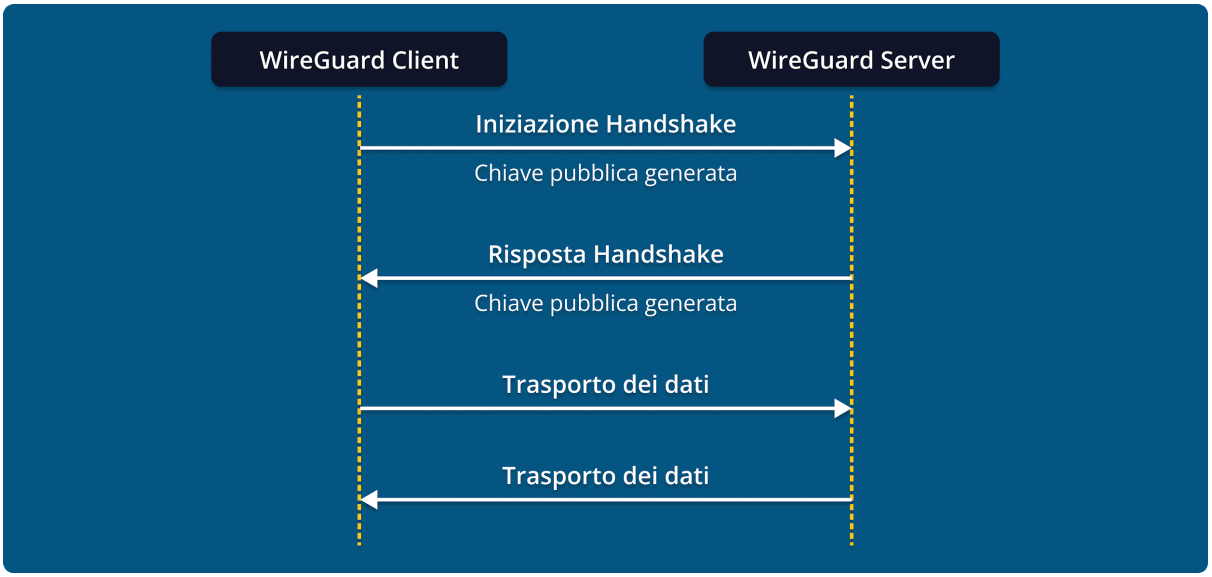

Figura 3.1: Diagramma di sequenza dell'handshake in WireGuard.
Il diagramma illustra i passi necessari per completare l'handshake del protocollo WireGuard, che richiede un solo round trip (1-RTT).

Per mitigare possibili attacchi DoS (Denial of Service), il protocollo non risponde in nessun modo a client non autorizzati, il comportamento rimane silenzioso e invisibile ad eccezione per i pacchetti autenticati e per l'handshake iniziale[11].

Il concetto di Cryptokey Routing è fondamentale per WireGuard e consente di associare le chiavi pubbliche dei peer ad una lista di IP consentiti all'interno del tunnel creato dal protocollo. Nel protocollo i peers sono identificati unicamente tramite la chiave pubblica, questo consente di creare un semplice mapping tra le chiavi pubbliche e un insieme di indirizzi IP consentiti al peer[19].

Quando un pacchetto in uscita dal server viene trasmesso ad un determinato IP, WireGuard utilizza questa tabella per trovare la giusta chiave pubblica da utilizzare per la crittografia del pacchetto; allo stesso modo, se un pacchetto in entrata non corrisponde agli IP consentiti per la chiave utilizzata per la crittografia, questo viene ignorato[19].

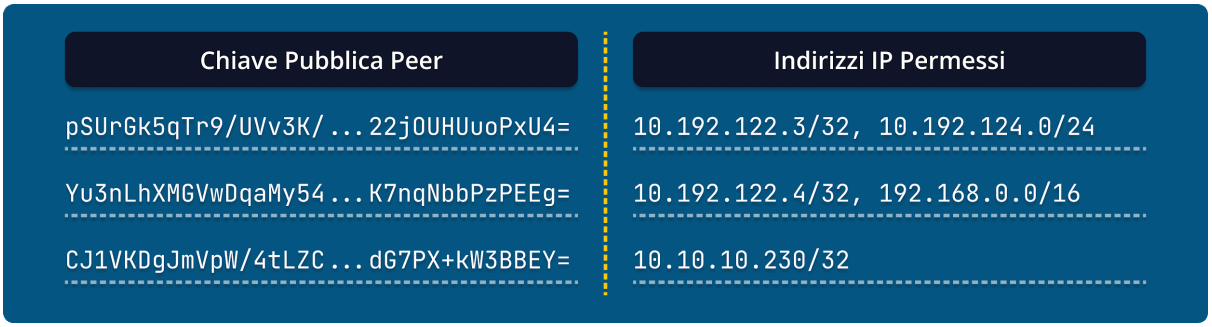

Figura 3.2: Esempio di una mappatura tra chiavi pubbliche e indirizzi IP permessi

3.2. Autenticazione e crittografia

WireGuard utilizza un modello di crittografia a chiave asimmetrica per autenticare e proteggere le comunicazioni tra i peer. Questo modello si basa sull'uso di chiavi pubbliche e private per ciascun peer.

Durante la fase di handshake iniziale, i peer scambiano le rispettive chiavi pubbliche, per poi procedere con lo scambio delle chiavi di sessione simmetriche. Queste chiavi vengono utilizzate per cifrare e decifrare i dati trasmessi tra i peer. Poiché le chiavi di sessione vengono generate in modo casuale per ogni sessione e vengono scambiate periodicamente, WireGuard offre una robusta protezione contro attacchi di tipo replay o di compromissione di quest'ultime[19].

Per la crittografia dei dati, WireGuard utilizza una combinazione di algoritmi[19]:

- **Curve25519**: algoritmo di scambio di chiavi basato su curve ellittiche per la generazione di chiavi simmetriche condivise[20];
- **ChaCha20**: algoritmo di stream cipher per la crittografia dei dati[21];
- **Poly1305**: algoritmo di autenticazione del messaggio per la verifica dell'integrità dei dati[22].

Questa combinazione di algoritmi offre un elevato livello di sicurezza e prestazioni.
Nel caso in cui la chiave privata venisse compromessa, l'utilizzo di una chiave di crittografia simmetrica diversa per ogni sessione rende impossibile decrittografare le comunicazioni passate[19].

4. Configurazione di un server WireGuard

4.1. Premessa

Per procedere con l'installazione e la configurazione del server WireGuard, prendiamo in considerazione una macchina virtuale con sistema operativo Ubuntu 22.04, larghezza di banda di 200 Mb/s e indirizzo IPv4 statico e pubblico; la quantità di memoria RAM o le prestazioni del processore non sono rilevanti per la configurazione. WireGuard supporta sia IPv4 che IPv6, per semplicità configuriamo il server tramite IPv4.
Fissiamo i seguenti valori e caratteristiche:

Sistema operativo	Ubuntu 22.04
Larghezza di banda disponibile	200 Mb/s
Banda minima per utente	5 Mb/s
Numero di utenti massimo	32
Larghezza di banda necessaria	160 Mb/s (5 Mb/s * 32)

4.2. Installazione

Una volta stabilita la connessione SSH alla macchina virtuale con un account root o con accesso al comando `sudo`, è possibile eseguire i seguenti comandi per aggiornare la lista dei pacchetti disponibili e installare WireGuard con le dipendenze necessarie:

```
$ apt -y update
$ apt -y install wireguard coreutils iptables iproute2 nano systemd ufw
```

4.3. Configurazione

Successivamente, utilizzando il comando `wg`, procediamo alla generazione della coppia chiave pubblica e privata per il server. Tramite il comando `wg genkey` generiamo la chiave privata che viene salvata nel file `/etc/wireguard/private.key`, tramite il comando `chmod` rimuoviamo tutti i permessi di lettura e scrittura da parte degli utenti non-root:

```
$ wg genkey | tee /etc/wireguard/private.key
$ chmod go= /etc/wireguard/private.key
```

Per la generazione della chiave pubblica, che andrà derivata dalla chiave privata e inserita in tutte le configurazioni dei peer che vorranno connettersi al server, utilizziamo il comando `wg pubkey`.
La chiave pubblica viene salvata nel file `/etc/wireguard/pub.key`:

```
$ cat /etc/wireguard/private.key | wg pubkey | tee /etc/wireguard/pub.key
```

In questo step definiamo il range di indirizzi IPv4 da assegnare ai peer e all'interfaccia che creiamo con WireGuard. La specifica RFC 1918 ha riservato i seguenti blocchi di indirizzi IPv4 per l'uso privato[23]:

Inizio blocco	Fine blocco	Prefisso
`10.0.0.0`	`10.255.255.255`	`10.0.0.0/8`
`172.16.0.0`	`172.31.255.255`	`172.16.0.0/12`
`192.168.0.0`	`192.168.255.255`	`192.168.0.0/16`

È possibile scegliere un qualsiasi range di indirizzi all'interno di uno dei tre blocchi; per i prossimi passi, a scopo esemplificativo, andiamo ad utilizzare il prefisso `10.0.0.0/24`, appartenente al primo blocco. Questo prefisso consentirà di assegnare fino a 255 indirizzi IP (massimo 254 se escludiamo l'indirizzo di broadcast per questioni di compatibilità).
Assegnamo al server WireGuard l'indirizzo privato `10.0.0.1` e la porta `8787`.

Per il prossimo step è necessario copiare la chiave privata che è stata generata in precedenza e che verrà inserita nel file di configurazione:

```
$ cat /etc/wireguard/private.key
```

Per concludere la configurazione, utilizziamo un editor di testo per andare a creare e modificare il file `/etc/wireguard/wg0.conf`:

```
$ nano /etc/wireguard/wg0.conf
```

Nel file inseriamo le seguenti righe, sostituendo `<PRIVATE_KEY>` con la chiave privata copiata in precedenza:

```
[Interface]
PrivateKey = <PRIVATE_KEY>
Address = 10.0.0.1/24
ListenPort = 8787
SaveConfig = true
```

In alternativa alla creazione manuale del tunnel che stiamo configurando tramite il comando `wg` ogni volta che è necessario, WireGuard, può essere configurato per l'avvio automatico come servizio all'avvio del sistema operativo tramite `systemd`. Questo è possibile grazie allo script `wg-quick`.
Prima di avviare il servizio, è necessario abilitarlo tramite questo comando, dove `wg0` è il nome del file di configurazione creato in precedenza:

```
$ systemctl enable wg-quick@wg0.service
```

Una volta abilitato il servizio, però essere avviato tramite il comando:

```
$ systemctl start wg-quick@wg0.service
```

Alcune macchine virtuali hanno il firewall disattivato come modalità predefinita, in ogni caso tramite il comando `ufw` aggiungiamo una nuova regola per permettere l'accesso dall'esterno tramite la porta `8787` UDP:

```
$ ufw allow 8787/udp
```

Se il firewall `ufw` è abilitato, tramite questo comando è possibile controllare le regole che sono state aggiunte:

```
$ ufw status
```

Il risultato di questo comando indica se il firewall è abilitato o meno e, in caso negativo, non mostrerà alcuna regola, anche nel caso di esito positivo del comando precedente.

Infine, per chiudere il tunnel WireGuard e procedere con le prossime configurazioni, eseguiamo questo comando, eseguire questo comando è fondamentale per evitare errori durante la configurazione del firewall:

```
$ systemctl stop wg-quick@wg0.service
```

4.4. Forwarding dei pacchetti

Per instradare tutto il traffico utente attraverso il server WireGuard, è necessario configurare il sistema operativo per abilitare il forwarding dei pacchetti IP. Questa configurazione avviene a livello di kernel e per farlo utilizziamo il file messo a disposizione dal software `sysctl`.

Quando l'IP forwarding è abilitato, il sistema operativo può agire esattamente come un router, indirizzando i pacchetti provenienti dai peer in internet e viceversa.

Il file `/etc/sysctl.conf` può essere modificato tramite un editor di testo per aggiungere il parametro relativo all'IP forwarding:

```
$ nano /etc/sysctl.conf
```

Sarà sufficiente aggiungere questa linea in fondo al file di configurazione:

```
1 net.ipv4.ip_forward=1
```

Infine, dopo aver salvato e chiuso il file, tramite il comando sysctl andiamo ad applicare la nuova configurazione:

```
$ sysctl -p
```

In questo modo il server WireGuard sarà in grado di reindirizzare il traffico in entrata ai relativi peer o in internet, utilizzando quindi l'indirizzo IP del server e nascondendo quello dei peer.

L'ultimo step per configurare questo comportamento è impostare il firewall per abilitare il NAT (Network Address Translation) e implementare correttamente il routing dei peer tramite il singolo indirizzo IP del server.
Conoscere il nome dell'interfaccia utilizzata dal server per la connessione a internet è necessario per i comandi successivi, per trovarlo si può utilizzare il comando `ip`:

```
$ ip route list default
```

L'output del comando include il nome dell'interfaccia immediatamente dopo la parola "dev", in questo caso `eth0`:

```
1 default via 85.120.0.1 dev eth0 onlink
```

Una volta riconosciuto il nome dell'interfaccia connessa ad internet andiamo a modificare la configurazione del server WireGuard creata in precedenza:

```
$ nano /etc/wireguard/wg0.conf
```

Nel file di configurazione, le istruzioni `PostUp` e `PreDown` servono per eseguire dei comandi dopo l'avvio e prima della chiusura del tunnel WireGuard, per la VPN proposta possono essere utilizzate per aggiungere e rimuovere regole del firewall:

```
    ...
7   PostUp = ufw route allow in on %i out on eth0
8   PostUp = iptables -t nat -I POSTROUTING -o eth0 -j MASQUERADE
9   PreDown = ufw route delete allow in on wg0 out on eth0
10  PreDown = iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
```

In questa configurazione, la linea 7 e 8 aggiungono due regole al firewall, che vengono eliminate quando il tunnel viene chiuso tramite la linea 9 e 10.

La prima regola (`ufw route allow in on %i out on eth0`) permette di reindirizzare i pacchetti provenienti dall'interfaccia del tunnel all'interfaccia `eth0`; la regola implica che il traffico di risposta viene reindirizzato automaticamente al tunnel.

Il secondo comando, invece, aggiunge una regola tramite `iptables` per configurare il mascheramento, ovvero la riscrittura dei pacchetti IP provenienti dal tunnel tramite NAT; il metodo "masquerade" indica che il firewall modifica automaticamente l'IP sorgente per i pacchetti in uscita in modo da agire come gateway.

4.5. Aggiunta di peer a WireGuard

Per permettere ad altri dispositivi di connettersi al server WireGuard, è necessario creare un file di configurazione per ciascun peer, da utilizzare nel proprio dispositivo. Questo file include le informazioni necessarie per stabilire la connessione VPN, come l'indirizzo IP del server, la chiave pubblica del server e la chiave privata del peer.
È possibile aggiungere un numero illimitato di peer al server, purché rientri nel range di indirizzi IP scelto per il tunnel.

L'aggiunta di un peer al server è un processo simile a quello della generazione delle chiavi e della configurazione del server stesso[24].

In WireGuard ogni peer è identificato dalla propria chiave pubblica, la coppia di chiavi pubblica e privata può essere generata utilizzando il comando `wg genkey`, l'ultimo output è la chiave pubbliche e va copiata per inserirla nella configurazione del server:

```
$ wg genkey | tee peer0-private.key
$ cat peer0-private.key | wg pubkey | tee peer0-public.key
```

Una volta ottenuta la coppia di chiavi è possibile aggiungere il peer alla configurazione del tunnel WireGuard, sostituire `<PEER_PUBLIC_KEY>` con la chiave copiata dal comando precedente. Per il primo peer, a scopo esemplificativo, assegniamo l'IP privato `10.0.0.2`:

```
$ nano /etc/wireguard/wg0.conf
```

```
...
[Peer]
PublicKey = <PEER_PUBLIC_KEY>
AllowedIps = 10.0.0.2/32
```

Per configurare il server è sufficiente aggiungere queste righe in fondo al file. La configurazione dei client è descritta nella sezione dedicata.

5. Isolamento degli utenti

5.1. Comunicazione tra peer

Ogni peer collegato al server WireGuard ha il proprio indirizzo IP privato, tramite questo indirizzo è possibile comunicare con gli altri peer connessi al server, esattamente come se si trovassero all'interno della stessa rete locale. Questa caratteristica rende possibile la comunicazione diretta e sicura tra i client connessi alla rete WireGuard, consentendo loro di scambiarsi dati protetti dalla crittografia[18].

Nel caso di configurazioni ad hoc, necessarie in contesti aziendali dove si vuole accedere a risorse remote dislocate in uffici e strutture diverse, con connessioni diverse, è possibile collegare logicamente le reti private tramite WireGuard e accedere alle risorse aziendali da qualsiasi luogo, come se fossero fisicamente presenti all'interno della rete locale dell'azienda.

Questa possibilità, per quanto utile, rappresenta un problema di sicurezza in un servizio di VPN che ha, tra gli scopi principali, quello di migliorare e proteggere la privacy dei propri utenti.

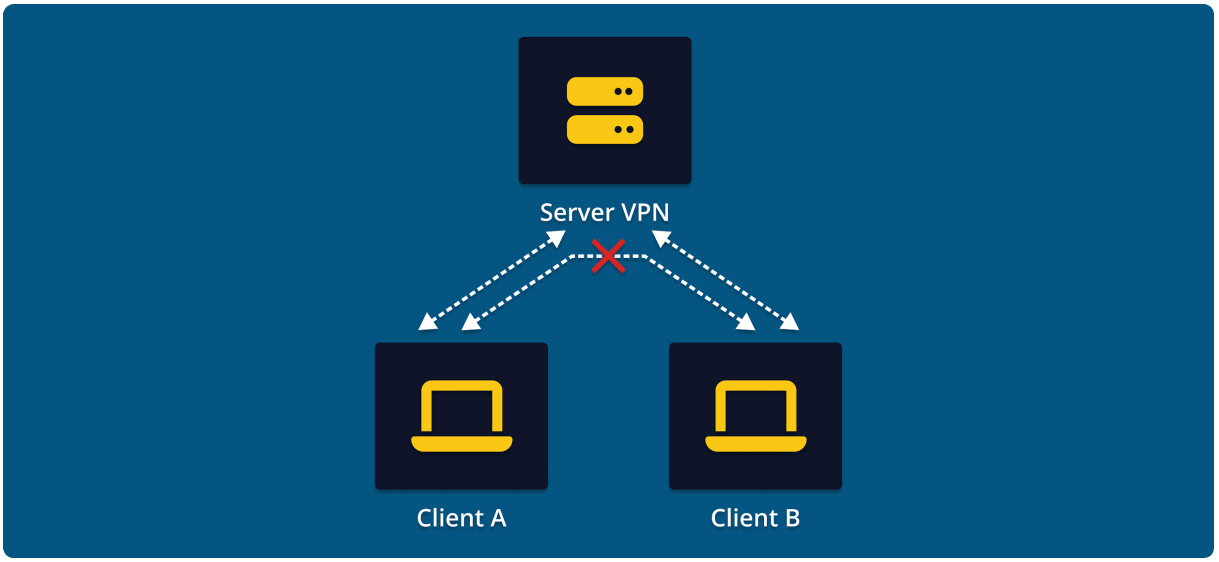

Figura 5.1: La comunicazione tra i client non è consentita

5.2. Utilizzo di iptables per isolare i peer

Per mitigare questo problema e garantire l'isolamento dei peer, è necessario implementare un meccanismo di sicurezza. L'utilizzo di un firewall come `iptables` è un'opzione efficace per configurare delle regole specifiche che blocchino eventuali tentativi di comunicazione tra i peer connessi al server lasciando la possibilità agli utenti di comunicare solo con il server VPN e non tra loro[25].

Come per gli altri comandi relativi al firewall, è necessario modificare il file di configurazione del server WireGuard e utilizzare le istruzioni `PostUp` e `PreDown`[26]:

```
$ nano /etc/wireguard/wg0.conf
```

La prima riga da aggiungere inserisce una regola tramite `iptables` al firewall che blocca ogni comunicazione che ha come sorgente e destinazione l'interfaccia del tunnel WireGuard.
La seconda riga, invece, rimuove la regola precedente dal firewall.

Questo è il contenuto finale del file di configurazione, modificato aggiungendo le righe 9 e 12:

```
1  [Interface]
2  PrivateKey = <PRIVATE_KEY>
3  Address = 10.0.0.1/24
4  ListenPort = 8787
5  SaveConfig = true
6
7  PostUp = ufw route allow in on %i out on eth0
8  PostUp = iptables -t nat -I POSTROUTING -o eth0 -j MASQUERADE
9  PostUp = iptables -A FORWARD -i %i -o %i -j REJECT
10 PreDown = ufw route delete allow in on wg0 out on eth0
11 PreDown = iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
12 PreDown = iptables -D FORWARD -i %i -o %i -j REJECT
13
14 [Peer]
15 PublicKey = <PEER_PUBLIC_KEY>
16 AllowedIps = 10.0.0.2/32
```

Infine, dopo aver salvato il file di configurazione, è possibile avviare il tunnel tramite `systemctl`:

```
$ systemctl start wg-quick@wg0.service
```

6. Gestione della banda

6.1. Tecniche di assegnazione della banda

La gestione della banda è cruciale per garantire un'esperienza di rete ottimale per tutti gli utenti connessi alla VPN. Le tecniche di assegnazione della banda mirano a distribuire equamente (o con priorità) la larghezza di banda disponibile tra i peer connessi. Questo capitolo esplorerà le principali tecniche utilizzate dai servizi di telecomunicazioni e dai servizi di VPN commerciali per gestire in modo efficace la banda.

In generale, un metodo comune utilizzato dagli ISP (Internet Service Provider) per gestire la banda in modo efficiente è l'oversubscription[27]. Questa pratica si basa sull'idea che non tutti gli utenti utilizzeranno simultaneamente la massima quantità di banda a loro disposizione, per questa ragione si consente a più utenti di condividere una quantità limitata di larghezza di banda. Ad esempio, se un ISP dispone di una larghezza di banda di 10 Gbps, potrebbe offrire più di 10 connessioni da 1 Gbps l'una ai propri clienti.

Per garantire un'esperienza utente ottimale e prevenire situazioni di congestione della rete, le VPN implementano una serie di tecniche per la gestione della banda[17]. Nel seguito di questo capitolo, approfondiremo due di queste tecniche, la prioritizzazione basata su classi di traffico e la limitazione della banda per utente.

6.1.1. Prioritizzazione basata su classi di traffico

Una delle tecniche più comuni è quella di suddividere il traffico in diverse categorie, o classi, e assegnare la priorità a ogni classe in base alle necessità del servizio[28].
Ad esempio, il traffico VoIP (Voice over IP), che comprende le chiamate vocali, potrebbe essere assegnato a una classe con priorità alta per garantire una comunicazione fluida e senza ritardi eccessivi, mentre il resto del traffico potrebbe essere assegnato a una classe con priorità più bassa.

Questo assicura che le applicazioni sensibili alla latenza e alla perdita di pacchetti abbiano sempre risorse sufficienti per funzionare in modo ottimale, anche in condizioni di carico elevato.

6.1.2. Limitazione della banda per utente

Per prevenire situazioni in cui un singolo utente può consumare eccessivamente la banda a discapito degli altri utenti, è possibile impostare dei limiti per ogni peer, in modo da limitare la quantità di dati che un utente può trasmettere o ricevere entro un

determinato periodo di tempo. Ad esempio è possibile impostare un limite di 100 Mb/s per ogni peer connesso al server.

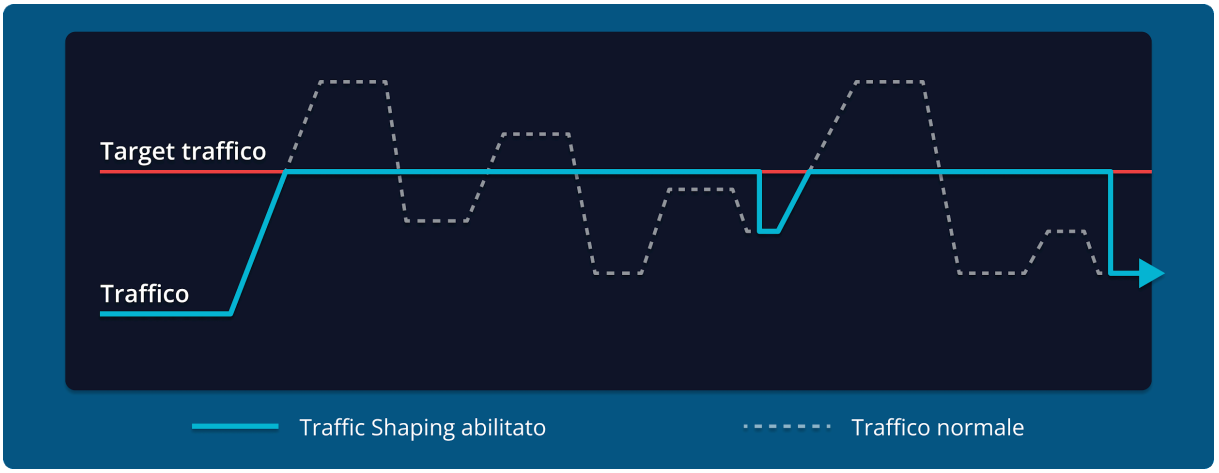

Figura 6.1: Confronto tra traffico normale e traffico modellato e limitato dal Traffic Shaping

6.2. Tecniche di traffic shaping

Il traffic shaping è una tecnica avanzata utilizzata nella gestione della banda all'interno delle reti che mira a controllare il flusso di traffico in modo da ottimizzare le prestazioni della rete e garantire un utilizzo efficiente delle risorse disponibili[29].

Questa tecnica fa uso di politiche di gestione del traffico e possono includere la limitazione della velocità di trasmissione, la prioritizzazione del traffico e la gestione delle code di trasmissione dei pacchetti. L'obiettivo principale del traffic shaping è evitare congestione di rete e garantire prestazioni simili per tutti gli utenti. L'utilizzo del traffic shaping consente di implementare le tecniche di assegnazione della banda come la prioritizzazione basata su classi di traffico e la limitazione della banda per utente, introdotti nel capitolo precedente.

Le tecniche di traffic shaping più comuni includono:

- Token Bucket;
- Hierarchical Token Bucket (HTB);
- Class-Based Queueing (CBQ);
- Common Applications Kept Enhanced (Cake).

Nel prossimo sottocapitolo, esamineremo due approcci al traffic shaping: Cake e HTB. Queste tecniche offrono un livello avanzato di flessibilità e controllo per una gestione più precisa del traffico degli utenti.

6.3. Traffic shaping: tecniche Cake e HTB

Cake e HTB sono due delle tecniche di traffic shaping più utilizzate per la gestione avanzata della banda in una rete.

Cake è un algoritmo di gestione del traffico relativamente nuovo, progettato per la condivisione equa della banda e la minimizzazione della latenza. Si basa sul principio di "fair queuing", che mira a fornire a tutti gli utenti un accesso equo alla larghezza di banda disponibile. Cake è particolarmente efficace in scenari con un elevato numero di utenti connessi e diverse esigenze di traffico, è il risultato dell'applicazione e dell'evoluzione degli algoritmi di gestione del traffico precedenti come lo Smart Queue Management (SQM) e il CoDel (Controlled Delay). La sua capacità di adattarsi dinamicamente alle variazioni del traffico lo rende una scelta preferita per le reti moderne ad alta densità di utenti e con requisiti di latenza sensibili[30].

Cake opera in "deficit mode", una modalità di funzionamento che prevede l'assegnazione di quote di banda basate sulle necessità effettive degli utenti in quel momento anziché allocare una quantità fissa di larghezza di banda a ciascun utente. Cake monitora costantemente il traffico ed è in grado di adattarsi rapidamente, un approccio dinamico che consente di ottimizzare l'utilizzo delle risorse di rete e di mantenere bassa la latenza anche in presenza di picchi di traffico improvvisi[31].

HTB è un algoritmo di traffic shaping più consolidato, basato su un sistema di token bucket; l'algoritmo assegna a ciascun utente un certo numero di token a intervalli regolari, un utente può trasmettere dati solo se possiede token sufficienti. HTB è una tecnica che può essere utilizzata per diversi scopi, come la limitazione della banda e la prioritizzazione del traffico[32].

Le classi di traffico in HTB possono essere organizzate in un albero gerarchico, con classi superiori che possono influenzare il comportamento delle classi sottostanti. Questo permette di priorizzare o limitare il traffico in base a criteri come il protocollo di trasporto, l'indirizzo IP di origine o destinazione e così via.

Per il servizio di VPN ci concentriamo sull'utilizzo di HTB per la configurazione di regole per definire la banda totale del server e la banda minima e massima per ogni peer.

6.4. Configurazione di HTB per la condivisione di banda

Per garantire un'allocazione equa della larghezza di banda disponibile è fondamentale suddividere la banda in classi e sottoclassi, assegnando a ogni classe un limite appropriato. L'obiettivo è assegnare un tetto massimo e un limite inferiore di banda sempre disponibile per ogni utente. Il primo passo consiste nella creazione di una struttura gerarchica che rifletta le esigenze.

La classe principale (o root) rappresenta la larghezza di banda totale disponibile per la VPN e sarà suddivisa in classi figlie.

Le sottoclassi (o classi figlie) rappresentano i vari peer connessi al server WireGuard, tramite un filtro basato sull'indirizzo IP del peer. Ogni sottoclasse avrà assegnati dei limiti di banda appropriati, in questo caso, come stabilito dal capitolo 4.1, il limite inferiore sarà impostato a 5 Mb/s mentre quello superiore a 200 Mb/s.

Per l'assegnazione dei limiti di banda esistono due parametri principali: Burst e Rate[32].
Il burst rappresenta la quantità massima di dati che un peer può trasmettere in un determinato periodo di tempo senza incorrere in penalità o restrizioni.
Il rate rappresenta la larghezza di banda, specificata, nel nostro caso, in megabit al secondo.

In questo caso il parametro burst non verrà configurato, in modo da applicare le regole su tutti i pacchetti, immediatamente.

La configurazione di HTB, sul kernel Linux, avviene tramite le utility fornite da Traffic Control. Il comando `tc` fa parte del pacchetto `iproute2` installato in precedenza e consente di configurare lo scheduler di rete. La configurazione sarà simile sia per il download sia per l'upload, ma per quest'ultimo è necessario creare un'interfaccia virtuale IFB (Intermediate Functional Block) che farà da tramite per filtrare i pacchetti in ingresso al server.

La creazione dell'interfaccia virtuale, con nome "vpninbound", avviene tramite questi comandi:

```
$ ip link add name vpninbound type ifb
$ ip link set vpninbound up
```

Successivamente, tramite Traffic Control, indirizziamo i pacchetti in ingresso alla nuova interfaccia:

```
$ tc qdisc replace dev wg0 handle ffff: ingress
$ tc filter add dev wg0 parent ffff: matchall action mirred egress \
  redirect dev vpninbound
```

A questo scopo utilizziamo due delle funzionalità principali di Traffic Control: qdisc e filter. Le Queueing Disciplines (qdisc) sono un meccanismo che determina come i pacchetti sono messi in coda, priorizzati e pianificati per la trasmissione; i filtri, invece, sono regole utilizzate per selezionare e classificare tipi specifici di pacchetti e consentono di eseguire delle operazioni su questi pacchetti[33].

Il primo comando è utilizzato per impostare una nuova disciplina della coda in ingresso sull'interfaccia `wg0`, identificata con "`ffff:`".
Il secondo comando aggiunge un filtro, sempre sull'interfaccia `wg0`, che ha come parente il qdisc creato in precedenza; questo filtro agisce su tutti i pacchetti (matchall) e definisce l'azione che in questo caso è la copia dei pacchetti sull'interfaccia virtuale `vpninbound`.

Dopo aver creato l'interfaccia virtuale e aver indirizzato i pacchetti in ingresso, l'ultimo passaggio è quello di applicare i limiti di banda utilizzando HTB, per farlo aggiungeremo un qdisc di tipo "htb" configurato per ogni indirizzo IP della VPN, sia per l'interfaccia `wg0` (download) che per l'interfaccia `vpninbound` (upload).

Per impostare HTB come disciplina della coda è sufficiente eseguire il comando `tc qdisc add`, ogni comando va eseguito per entrambe le interfacce:

```
$ tc qdisc add dev wg0 root handle 1: htb default 10
$ tc qdisc add dev vpninbound root handle 1: htb default 10
```

La classe principale di HTB, con banda minima e massima di 200 Mb/s, viene configurata tramite il comando `tc class` dove `rate 200mbit` indica la banda minima e `ceil 200mbit` la banda massima:

```
$ tc class add dev wg0 parent 1: classid 1:1 htb rate 200mbit ceil 200mbit
$ tc class add dev vpninbound parent 1: classid 1:1 htb rate 200mbit ceil \
  200mbit
```

Gli ultimi comandi da eseguire per completare la configurazione di HTB sono le sottoclassi e i relativi filtri per ogni IP privato della VPN, in modo da impostare una banda minima di 5 Mb/s e una banda massima di 200 Mb/s.

Per semplicità utilizziamo un `for` tramite il linguaggio `bash`, per eseguire i comandi a partire dall'indirizzo IP `10.0.0.2` fino all'indirizzo `10.0.0.254`:

```
for i in {2..254}
do
  CLIENT_IP="10.0.0.$i"
  # download
  tc class add dev wg0 parent 1:1 classid 1:1$i htb rate 5mbit ceil \
  200mbit
  tc filter add dev wg0 protocol ip parent 1:0 prio 1 u32 match ip \
  dst ${CLIENT_IP} flowid 1:1$i

  # upload
  tc class add dev vpninbound parent 1:1 classid 1:1$i htb rate \
  5mbit ceil 200mbit
  tc filter add dev vpninbound protocol ip parent 1:0 prio 1 u32 \
  match ip src ${CLIENT_IP} flowid 1:1$i

done
```

Al termine della configurazione, ogni peer che si connette al server WireGuard rispetterà le regole di banda stabilite sopra, garantendo un'allocazione equa e controllata della larghezza di banda sia in download che in upload, senza superare i limiti imposti e senza compromettere le prestazioni complessive della rete.

7. Configurazione dei client

Per potersi connettere al server WireGuard precedentemente configurato, è necessario configurare anche i client, a partire dall'installazione di WireGuard sui dispositivi che si desidera utilizzare. L'installazione varia a seconda del sistema operativo in uso, di seguito elenchiamo i passaggi principali per la configurazione dei client sui principali sistemi operativi come Windows, Linux e Android.

Per tutti i client la configurazione del peer è in comune, ma è necessario sostituire `<PEER_PRIVATE_KEY>` e `<SERVER_PRIVATE_KEY>` con i valori presenti nei file `peer0-private.key` e `/etc/wireguard/private.key`, `<SERVER_IP>` con l'indirizzo IP pubblico del server WireGuard. L'indirizzo privato utilizzato, come per gli esempi precedenti, è `10.0.0.2`:

```
[Interface]
PrivateKey = <PEER_PRIVATE_KEY>
Address = 10.0.0.2/32

[Peer]
PublicKey = <SERVER_PRIVATE_KEY>
AllowedIPs = 0.0.0.0/0
Endpoint = <SERVER_IP>:8787
PersistentKeepalive = 15
```

7.1. Windows

Tramite il sito ufficiale è possibile scaricare il file di installazione del client ufficiale di WireGuard: https://www.wireguard.com/install/. Il file, in formato `exe` o `msi`, una volta aperto copia e configura automaticamente tutti i file necessari per il client; al termine dell'installazione viene aperta la finestra principale (Figura 7.1) dove è possibile aggiungere, modificare e rimuovere i tunnel WireGuard.

Tramite il menù a tendina alla destra di "Add Tunnel" oppure la scorciatoia da tastiera `Ctrl + N` è possibile aggiungere la configurazione del server.
Nel campo "Name" è possibile inserire il nome del tunnel che non deve contenere spazi o iniziare con un numero, nell'area di testo, invece, rimpiazziamo il contenuto con la configurazione del peer.

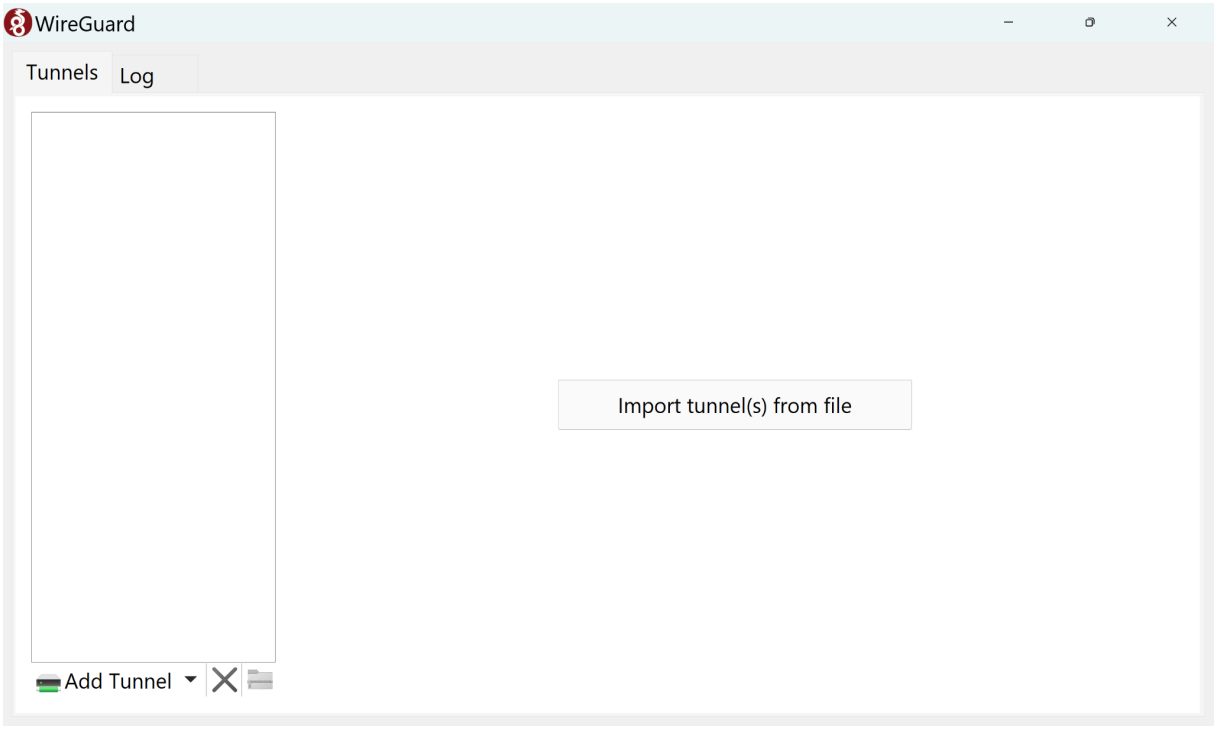

Figura 7.1: Interfaccia principale del client WireGuard per Windows

Una volta salvato il tunnel è possibile connettersi tramite il pulsante “Activate”:

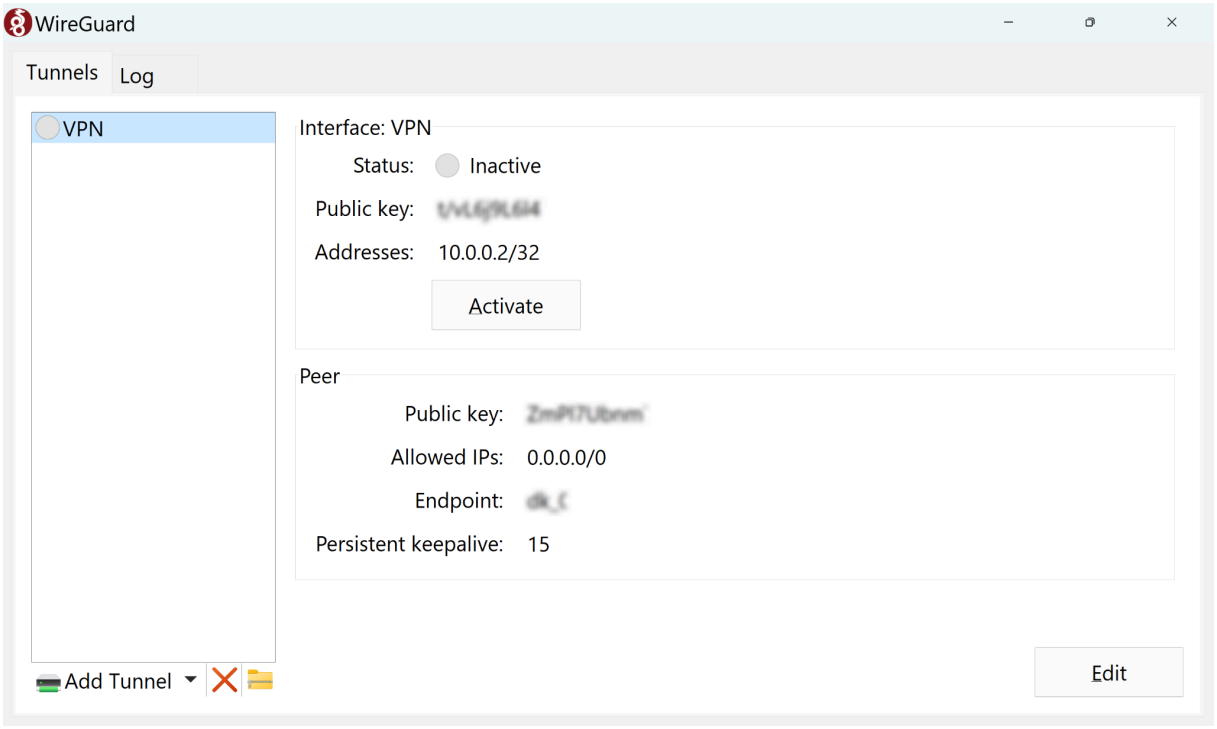

Figura 7.2: Interfaccia principale con il tunnel configurato

7.2. Linux

L'installazione e configurazione del client su Linux è molto simile a quella del server. Alcune versioni del sistema operativo mettono a disposizione un'interfaccia grafica, solitamente nelle impostazioni, per poter aggiungere un tunnel WireGuard tramite il proprio network manager; tutte le altre, invece, devono essere configurate a linea di comando o tramite applicazioni di terze parti. In ogni caso, per trovare il comando adatto è consigliabile controllare il sito ufficiale contenente le istruzioni di installazione: https://www.wireguard.com/install/

Il primo passo consiste nell'installazione di WireGuard, il nome del pacchetto o il comando utilizzato dipende dalla distribuzione del sistema operativo che si utilizza, per Debian e derivati il comando è il seguente:

```
$ apt update
$ apt install wireguard
```

Su distribuzione Fedora il comando invece è il seguente:

```
$ dnf install wireguard-tools
```

Arch utilizza un altro package manager, quindi il comando da eseguire è diverso:

```
$ pacman -S wireguard-tools
```

Una volta installato WireGuard, con un qualsiasi editor di testo, è necessario copiare la configurazione del peer nel file `/etc/wireguard/wg0.conf`, per utilizzare l'editor nano è sufficiente eseguire questo comando, come per il server:

```
$ nano /etc/wireguard/wg0.conf
```

Infine, dopo aver salvato il file di configurazione, la utility `wg-quick` può essere utilizzata per attivare il tunnel:

```
$ wg-quick up wg0
```

Questo comando avvia il tunnel specificato nel file di configurazione `wg0.conf`. Una volta eseguito il comando, il tunnel WireGuard sarà attivo e pronto per l'uso.

7.3. Android

Per configurare il client su Android è necessario installare l'applicazione ufficiale, scaricabile direttamente dal sito di WireGuard in formato APK o tramite il Google Play Store. L'applicazione è identificata dal pacchetto `com.wireguard.android`.

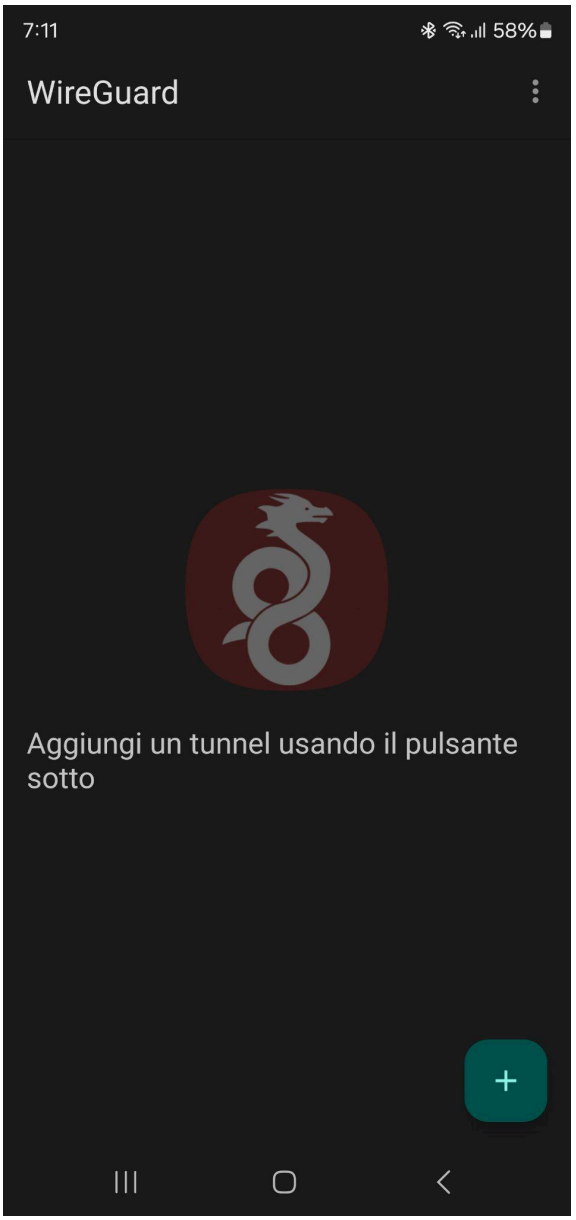

Figura 7.3: Applicazione ufficiale WireGuard per Android

Tramite il pulsante "+" in basso a destra è possibile configurare il nuovo tunnel, utilizzando la configurazione creata in precedenza.

Figura 7.4: Il tunnel aggiunto è visibile nella schermata principale dell'applicazione

7.4. Altri sistemi operativi

Per gli altri sistemi operativi che non sono stati elencati, ma compatibili con WireGuard, la configurazione può essere gestita in modo simile. Tuttavia, l'approccio specifico può variare a seconda delle implementazioni disponibili per il sistema operativo in questione.

Ad esempio, su macOS, la configurazione di WireGuard è simile a quella di Windows. L'applicazione ufficiale di WireGuard è scaricabile dall'App Store di macOS. Una volta installata, l'interfaccia utente presenta opzioni simili a quelle di Windows, consentendo agli utenti di aggiungere, configurare e gestire i tunnel WireGuard con facilità.

Per altre piattaforme, non supportate ufficialmente, potrebbero esistere implementazioni di terze parti. In generale, la configurazione prevede le stesse informazioni, specificando la chiave privata del peer, l'indirizzo IP del server, la chiave pubblica etc. per poi procedere con l'attivazione del tunnel.

In ogni caso si consiglia di fare riferimento al sito ufficiale per ulteriori informazioni.

8. Conclusione

In questa tesi è stata sviluppata una VPN client-server con un focus specifico su tre obiettivi principali: isolamento degli utenti, gestione avanzata della banda e prestazioni elevate. È stato anche introdotto il protocollo WireGuard, fondamentale per la soluzione proposta.

La VPN sviluppata si è dimostrata un'alternativa valida alle soluzioni attualmente disponibili sul mercato offrendo un servizio efficiente e sicuro.

L'isolamento degli utenti è stato garantito attraverso una configurazione del firewall del server con l'uso di strumenti come `iptables`. La gestione avanzata della banda ha consentito di ottimizzare l'utilizzo delle risorse, garantendo una connessione stabile e prevenendo situazioni di sovraffollamento. Infine, l'adozione di WireGuard ha permesso di ottenere prestazioni elevate, riducendo al minimo la latenza e massimizzando l'utilizzo della banda disponibile.

8.1. Sviluppi futuri

Per quanto la VPN proposta soddisfi gli obiettivi stabiliti, ci sono ancora opportunità di miglioramento e sviluppo futuro. Ad esempio, potrebbero essere esplorate ulteriori tecniche di ottimizzazione della banda e della gestione del traffico per migliorare ulteriormente le prestazioni della rete. Inoltre, potrebbero essere esaminate nuove funzionalità o integrazioni per rendere la soluzione ancora più versatile e adatta alle esigenze specifiche degli utenti.

Tra le varie aree di sviluppo è possibile approfondire:

- La prioritizzazione del traffico, basato sul riconoscimento dei pacchetti e il QoS (Quality of Service).
- L'utilizzo di più server WireGuard per implementare un bilanciamento del carico (load balancing) dei client e approfondire la capacità di roaming del protocollo.
- Quali legislazioni sono più adatte ad un servizio di VPN che punta alla privacy dei propri utenti. Sono richiesti i log o è possibile praticare una politica no-log?
- Supporto per IPv6, per permettere agli utenti di sfruttare i vantaggi del protocollo più recente.
- Sviluppo di un'interfaccia utente, per facilitare l'utilizzo della VPN da parte dell'utente finale.
- Integrazione con un sistema di autenticazione, per facilitare la gestione dei client.

9. Bibliografia

[1] European Commission - “Shaping Europe's digital future” - https://digital-strategy.ec.europa.eu/en

[2] Marco Gui (2015) - “Le trasformazioni della disuguaglianza digitale tra gli adolescenti: evidenze da tre indagini nel Nord Italia” - https://journals.openedition.org/qds/515 - DOI 10.4000/qds.515

[3] Google - “Norme sulla privacy” - https://policies.google.com/privacy

[4] Am I Unique? - https://amiunique.org/

[5] Forbes - “VPN Statistics & Trends In 2024” - https://www.forbes.com/advisor/business/vpn-statistics/

[6] Cloudflare - “The Relative Cost of Bandwidth Around the World” - https://blog.cloudflare.com/the-relative-cost-of-bandwidth-around-the-world/

[7] Fastweb - “Traffic shaping, cos’è e a cosa serve” - https://www.fastweb.it/fastweb-plus/digital-magazine/traffic-shaping-cos-e-e-a-cosa-serve/

[8] Security.org - “The Best VPN Services” - https://www.security.org/vpn/best/

[9] OpenVPN - “Reference manual for OpenVPN 2.6” - https://openvpn.net/community-resources/reference-manual-for-openvpn-2-6/

[10] Benjamin Dowling, Kenneth G. Paterson (2018) - “A Cryptographic Analysis of the WireGuard Protocol” - https://link.springer.com/chapter/10.1007/978-3-319-93387-0_1 - DOI 10.1007/978-3-319-93387-0_1

[11] WireGuard - “Protocol & Cryptography” - https://www.wireguard.com/protocol/

[12] IETF - RFC 2661 - https://www.ietf.org/proceedings/50/I-D/l2tpext-svctype-00.txt

[13] IETF - “SIPP Encapsulating Security Payload” - https://web.archive.org/web/20160909031941/http://www.toad.com/gnu/draft-ietf-sip-esp-00.txt

[14] Tailscale - “IPsec vs. WireGuard” - https://tailscale.com/compare/ipsec

[15] paperplans5 - “VPN Comparison Table” - https://docs.google.com/spreadsheets/d/1ijfqfLrJWLUVBfJZ_YalVpstWsjw-JGzkvMd6u2jqEk/edit

[16] Eglė Juodytė, NordVPN - “Does a VPN decrease internet speed?” - https://nordvpn.com/it/blog/does-vpn-slow-down-internet/

[17] Shweta, Forbes - “Does A VPN Slow Down Your Internet?” - https://www.forbes.com/advisor/business/does-vpn-slow-down-internet/

[18] WireGuard - https://www.wireguard.com/

[19] Jason A. Donenfeld (2017) - “WireGuard: Next Generation Kernel Network Tunnel” - https://www.wireguard.com/papers/wireguard.pdf - DOI 10.14722/ndss.2017.23160

[20] Daniel J. Bernstein (2006) - “A state-of-the-art Diffie-Hellman function” - https://cr.yp.to/ecdh.html

[21] Daniel J. Bernstein (2008) - “ChaCha, a variant of Salsa20” - https://cr.yp.to/chacha/chacha-20080128.pdf

[22] Daniel J. Bernstein (2005) - “The Poly1305-AES Message-Authentication Code” - https://link.springer.com/chapter/10.1007/11502760_3 - DOI 10.1007/11502760_3

[23] IETF - RFC 1918 - https://datatracker.ietf.org/doc/html/rfc1918

[24] WireGuard - “Quick Start” - https://www.wireguard.com/quickstart/

[25] netfilter - “The netfilter.org iptables project” - https://www.netfilter.org/projects/iptables/index.html

[26] Debian Manpages - “wg-quick(8) – wireguard-tools” - https://manpages.debian.org/bookworm/wireguard-tools/wg-quick.8.en.html

[27] Doug Dawson (2021) - “Demystifying ISP Oversubscription” - https://circleid.com/posts/20210914-demystifying-isp-oversubscription

[28] Oracle - “Using Classes of Service to Prioritize Traffic” - https://docs.oracle.com/cd/E36784_01/html/E36826/ipqos-intro-70.html

[29] Cisco - “Traffic Shaping” - https://www.cisco.com/c/it_it/support/docs/smb/wireless/CB-Wireless-Mesh/2076-traffic-shaping.html

[30] Bufferbloat.net - “Cake” - https://www.bufferbloat.net/projects/codel/wiki/Cake/

[31] Bufferbloat.net - “Cake Technical Information” - https://www.bufferbloat.net/projects/codel/wiki/CakeTechnical/

[32] MikroTik - “HTB (Hierarchical Token Bucket)” - https://help.mikrotik.com/docs/pages/viewpage.action?pageId=137986076

[33] The Linux Documentation Project - “Components of Linux Traffic Control” - https://tldp.org/HOWTO/Traffic-Control-HOWTO/components.html

Ringraziamenti

Ringrazio i miei genitori e mio fratello, per avermi sostenuto nel mio percorso e per esserci stati in ogni momento. Ringrazio i miei amici per avermi accompagnato in questi anni, aver creato con me moltissimi ricordi e condiviso gioie e dolori. Ringrazio la mia grande famiglia, per avermi ispirato fin da piccolo.