ProxyGuard

WireGuard behind a reverse proxy

Jeroen Wijenbergh, GÉANT

Me

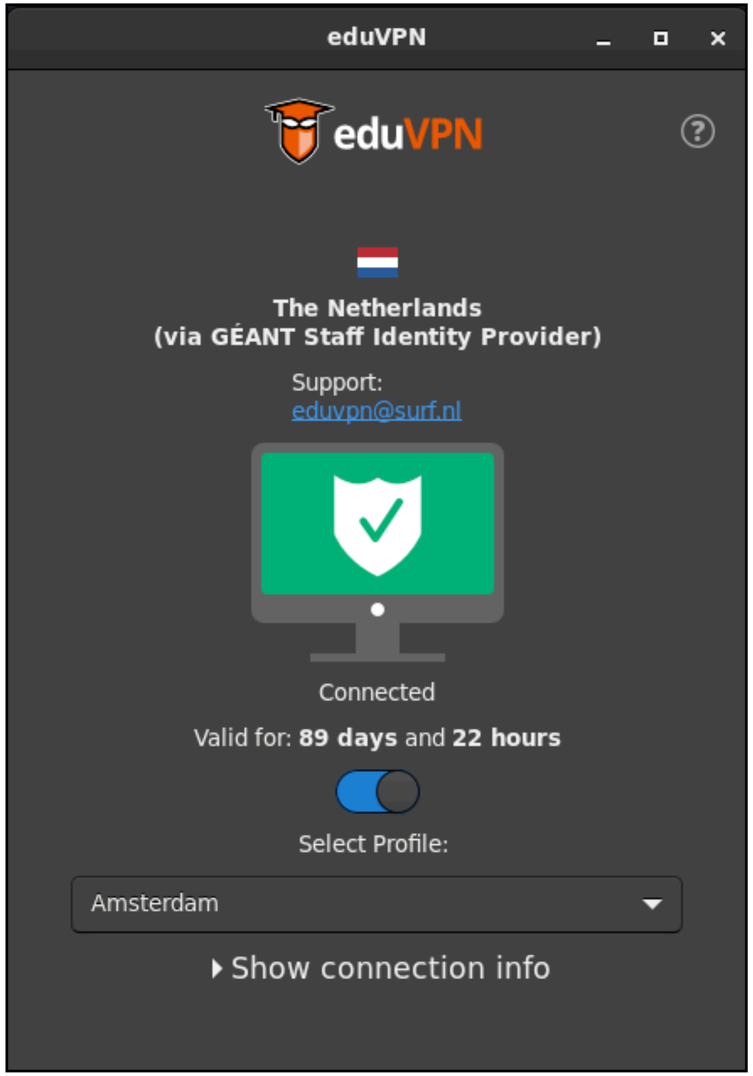
GÉANT
Networks • Services • People

eduVPN

eduroam

eduVPN

- Free & open-source VPN
- Focused on research & education
- Easy to self-host: https://docs.eduvpn.org/server/v3/
- Supports OpenVPN & WireGuard (since version 3)
- OpenVPN has UDP + TCP out of the box
- WireGuard only UDP



So what?

- UDP can be blocked ☹️
- MTU issues
- 443/53 not the holy grail

WIREGUARD

FAST, MODERN, SECURE VPN TUNNEL

- Less is more: 4,000 LOC [1], compared to OpenVPN's 70,000 [2]
- Missing features: TCP
- Quote: *Transforming WireGuard's UDP packets into TCP is the job of an upper layer of obfuscation* https://wireguard.com/known-limitations/

1: https://www.wireguard.com/papers/wireguard.pdf

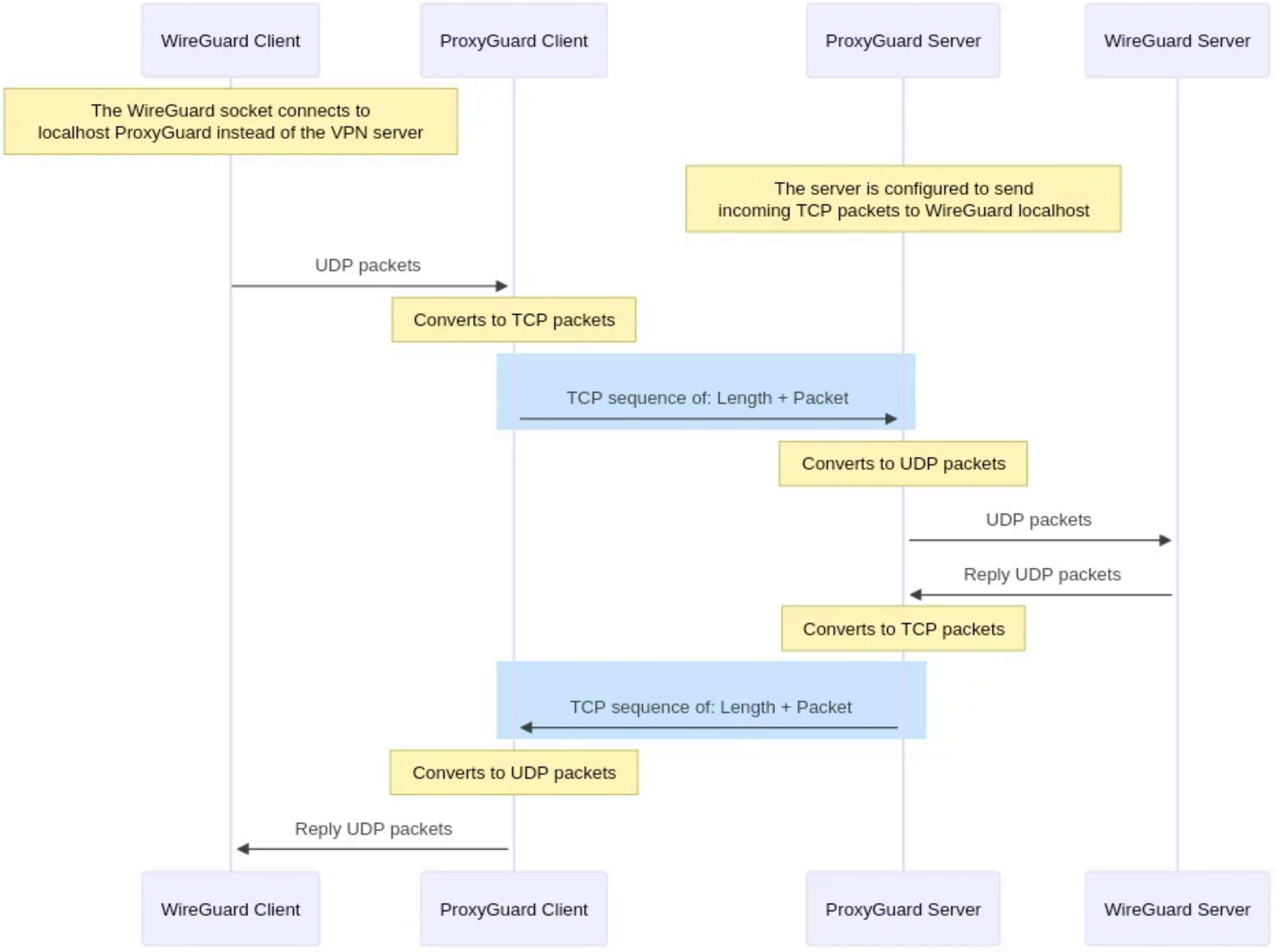
2: https://blog.openvpn.net/what-is-cloudflare-vpn/

Finding a tool

- Must: Client & Server implementation
- Nice to have:
  - Go implementation
  - Run behind reverse proxy (port sharing)
  - TLS

ProxyGuard

- https://codeberg.org/eduvpn/proxyguard
- First version in Go
- Simple UDP wrapper over TCP
- How to deal with cut-off packets?


WireGuard Client
ProxyGuard Client
ProxyGuard Server
WireGuard Server
The WireGuard socket connects to localhost ProxyGuard instead of the VPN server
The server is configured to send incoming TCP packets to WireGuard localhost
UDP packets
Converts to TCP packets
TCP sequence of: Length + Packet
Converts to UDP packets
UDP packets
Reply UDP packets
Converts to TCP packets
TCP sequence of: Length + Packet
Converts to UDP packets
Reply UDP packets
WireGuard Client
ProxyGuard Client
ProxyGuard Server
WireGuard Server

Checking the requirements

- ✅ Client & Server implementation
- Nice to have:
  - ✅ Go implementation
  - ❌ Run behind reverse proxy (port sharing)
  - ❌ TLS

Reverse proxy?

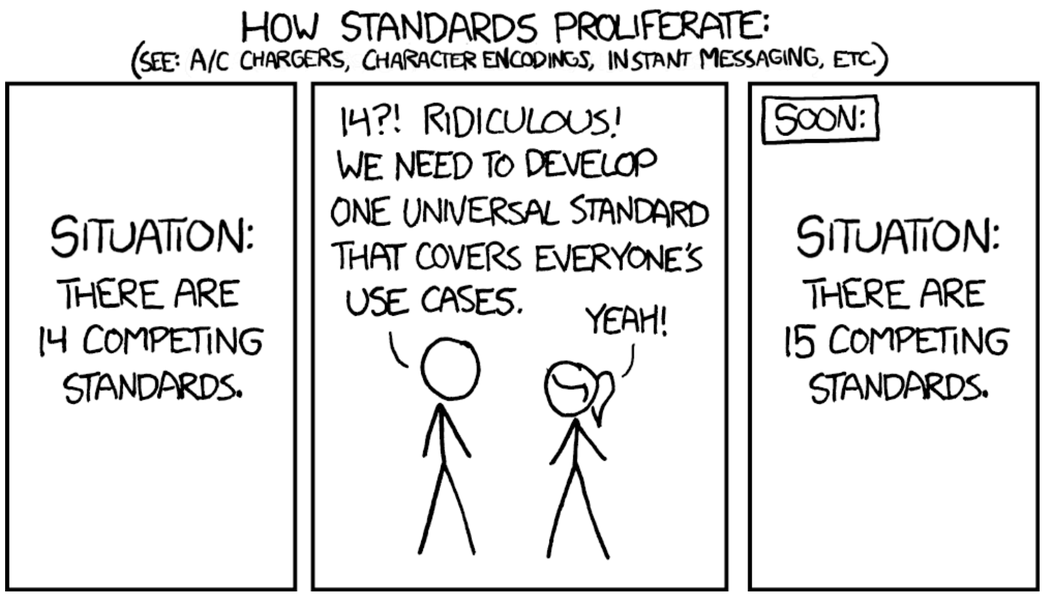
- Need to somehow proxy the connection?
- WebSockets?
  - Complex packet format
  - Does more than we need

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-------+-+-------------+-------------------------------+
|F|R|R|R| opcode|M| Payload len |    Extended payload length    |
|I|S|S|S|  (4)  |A|     (7)     |             (16/64)           |
|N|V|V|V|       |S|             |   (if payload len==126/127)   |
| |1|2|3|       |K|             |                               |
+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
|     Extended payload length continued, if payload len == 127  |
+ - - - - - - - - - - - - - - - +-------------------------------+
|                               |Masking-key, if MASK set to 1  |
+-------------------------------+-------------------------------+
| Masking-key (continued)       |          Payload Data         |
+-------------------------------- - - - - - - - - - - - - - - - +
:                     Payload Data continued ...                :
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                     Payload Data continued ...                |
+---------------------------------------------------------------+
```

https://datatracker.ietf.org/doc/html/rfc6455

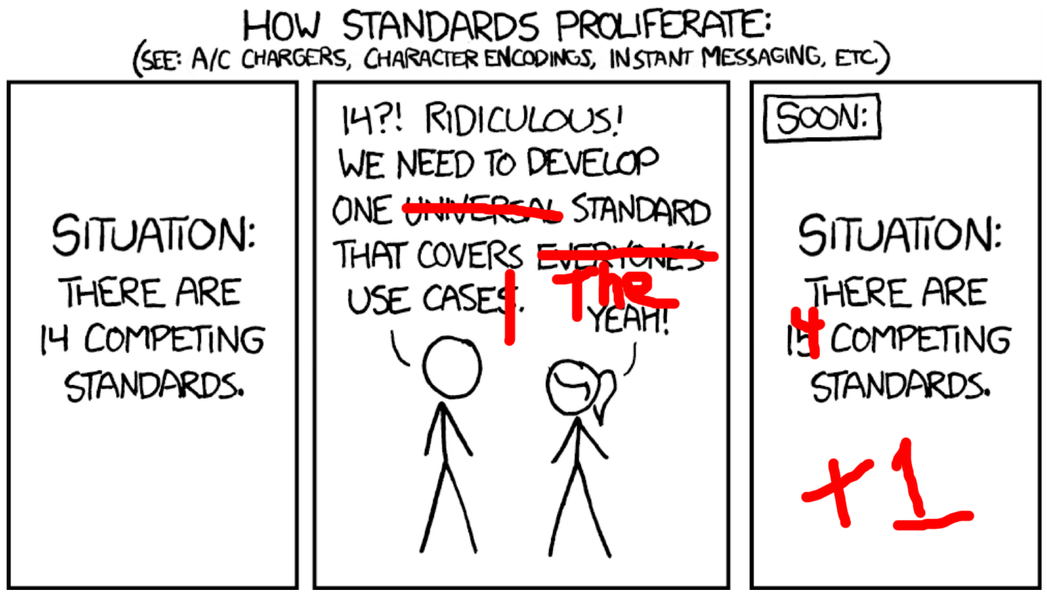
Fine, our own protocol?!

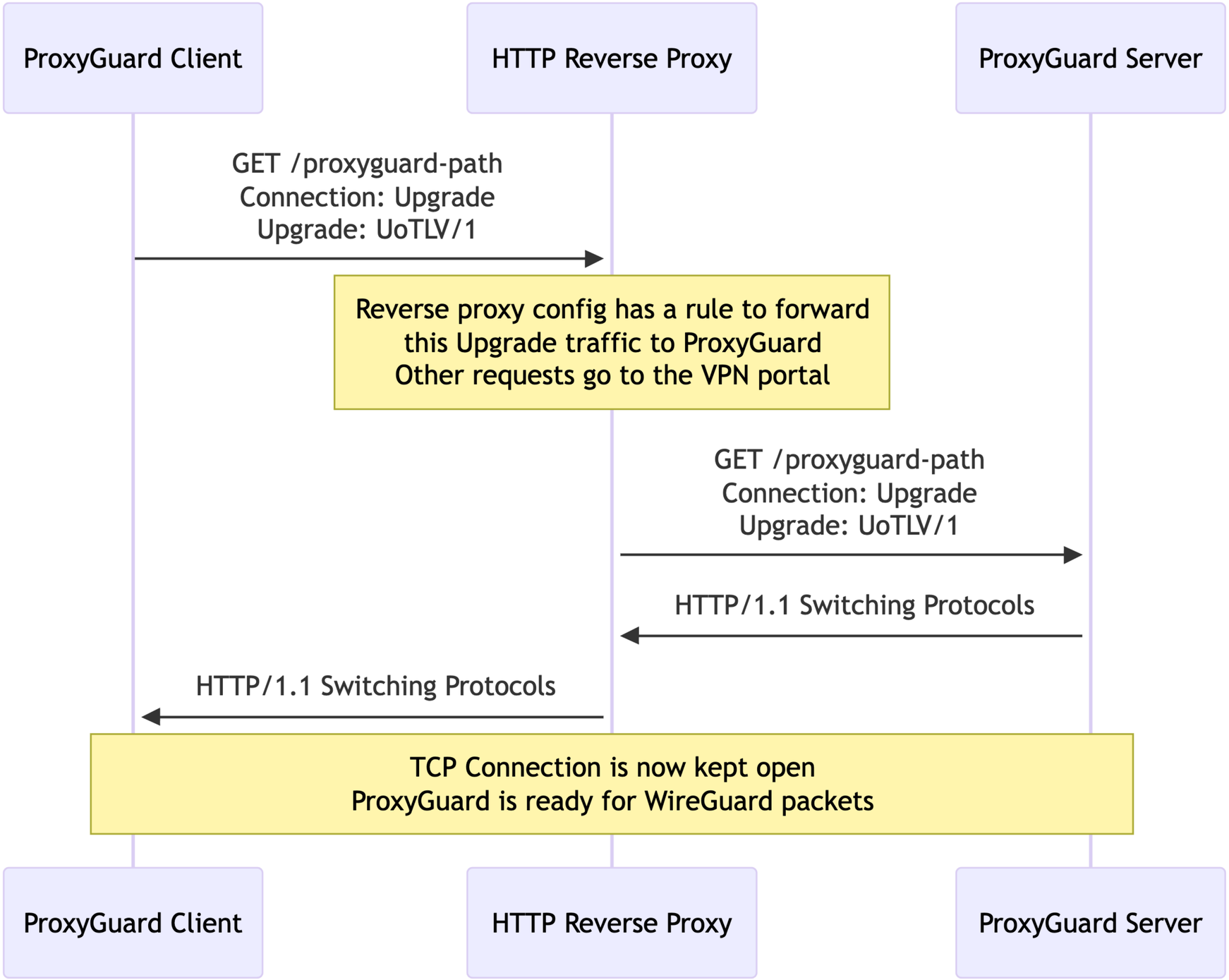
- Idea:
  - Re-use the handshake of web-sockets
  - Keep the packet format length + value



Fine, our own protocol?!

- Idea:
  - Re-use the handshake of web-sockets
  - Keep the packet format length + value




ProxyGuard Client
HTTP Reverse Proxy
ProxyGuard Server
GET /proxyguard-path
Connection: Upgrade
Upgrade: UoTLV/1
Reverse proxy config has a rule to forward
this Upgrade traffic to ProxyGuard
Other requests go to the VPN portal
GET /proxyguard-path
Connection: Upgrade
Upgrade: UoTLV/1
HTTP/1.1 Switching Protocols
HTTP/1.1 Switching Protocols
TCP Connection is now kept open
ProxyGuard is ready for WireGuard packets
ProxyGuard Client
HTTP Reverse Proxy
ProxyGuard Server

dev@dev-QEMU-Virtual-Machine: ~

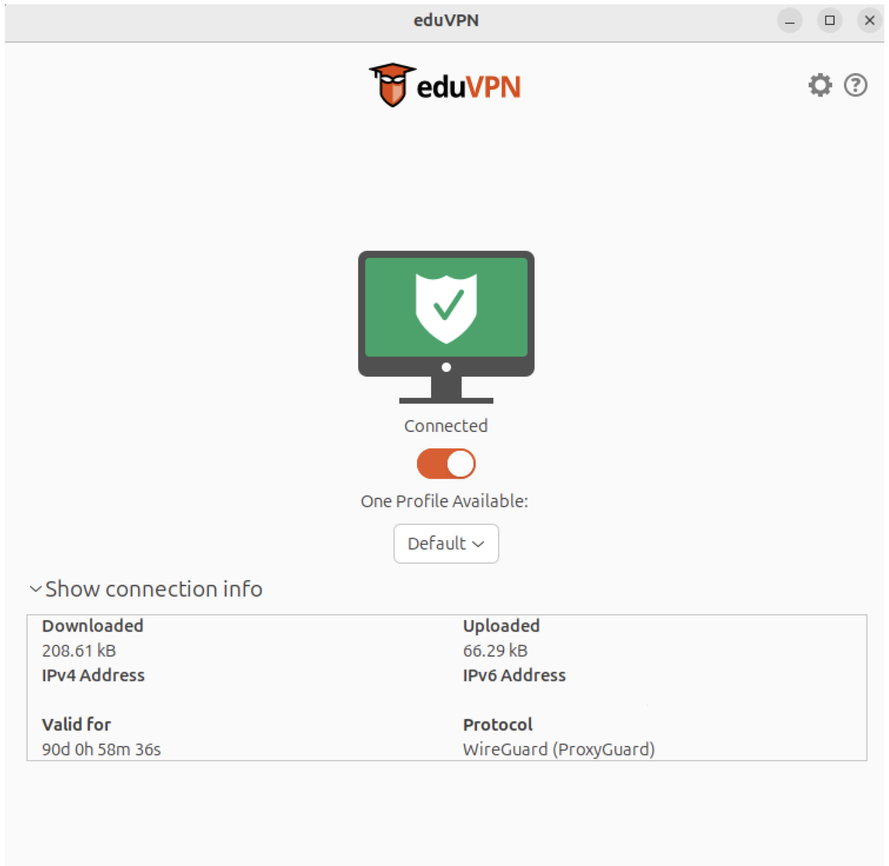
dev@dev-QEMU-Virtual-Machine: ~ | dev@dev-QEMU-Virtual-Machine: ... | dev@dev-QEMU-Virtual-Machine: ... | debian@debian: ~

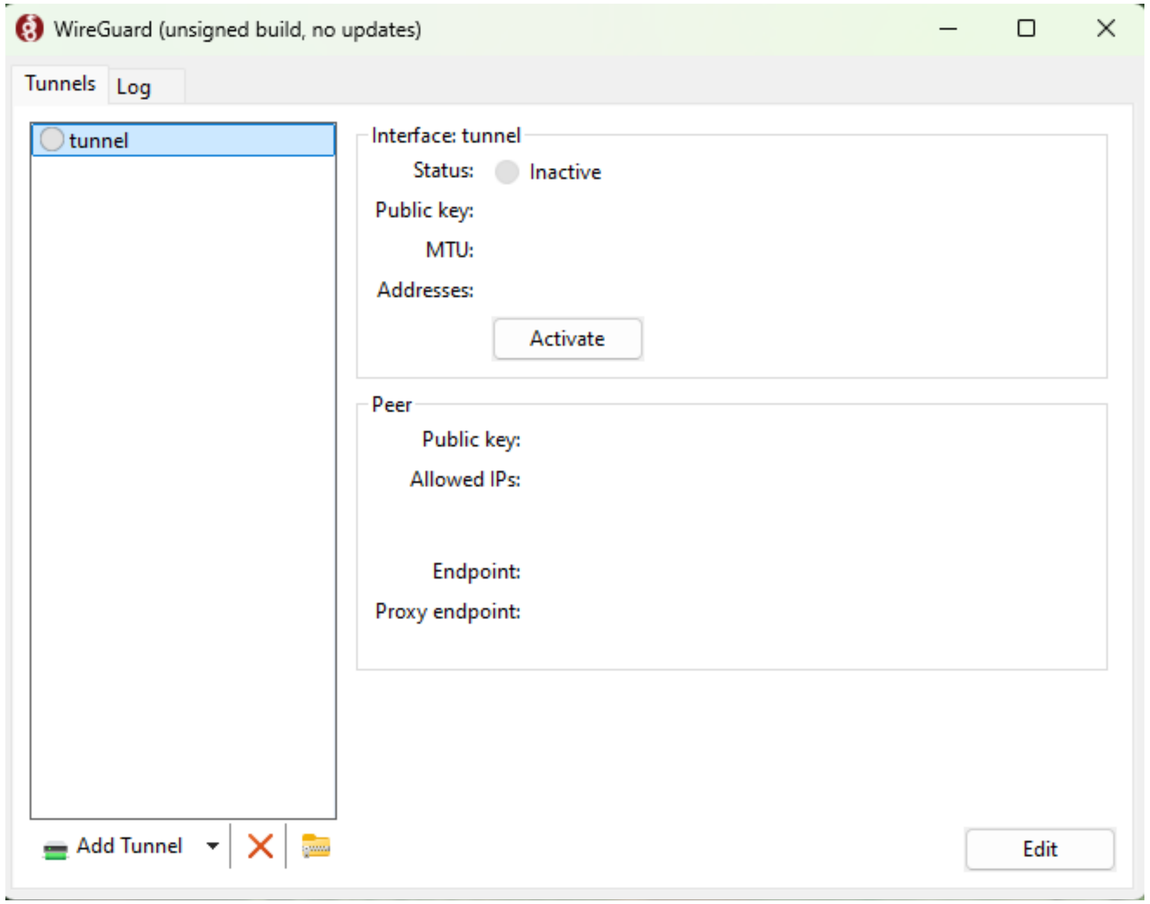
```
[Interface]
PrivateKey = ...
Address = ...
DNS = ...
ListenPort = 51820
Fwmark = 51820

[Peer]
PublicKey = ...
AllowedIPs = 0.0.0.0/0,::/0
PersistentKeepalive = ...
# ProxyGuard client listener
Endpoint = 127.0.0.1:51821
~
~
~
                                                    7,0-1         All
```

https://www.youtube.com/watch?v=ns1EcNeab-c

ProxyGuard: Client implementations

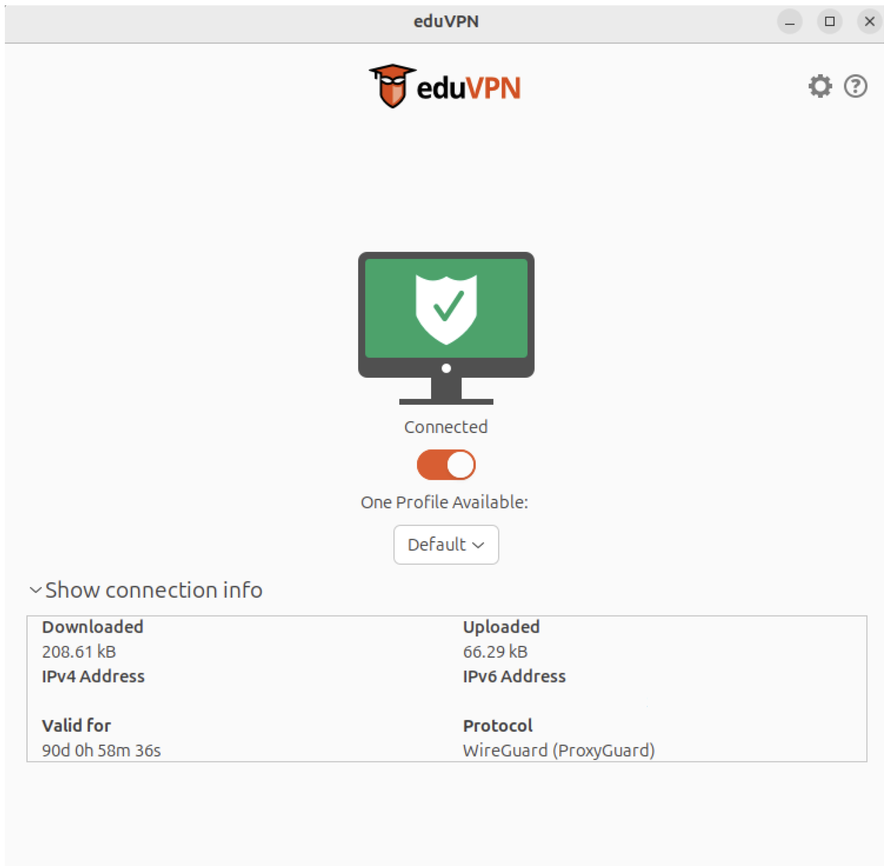


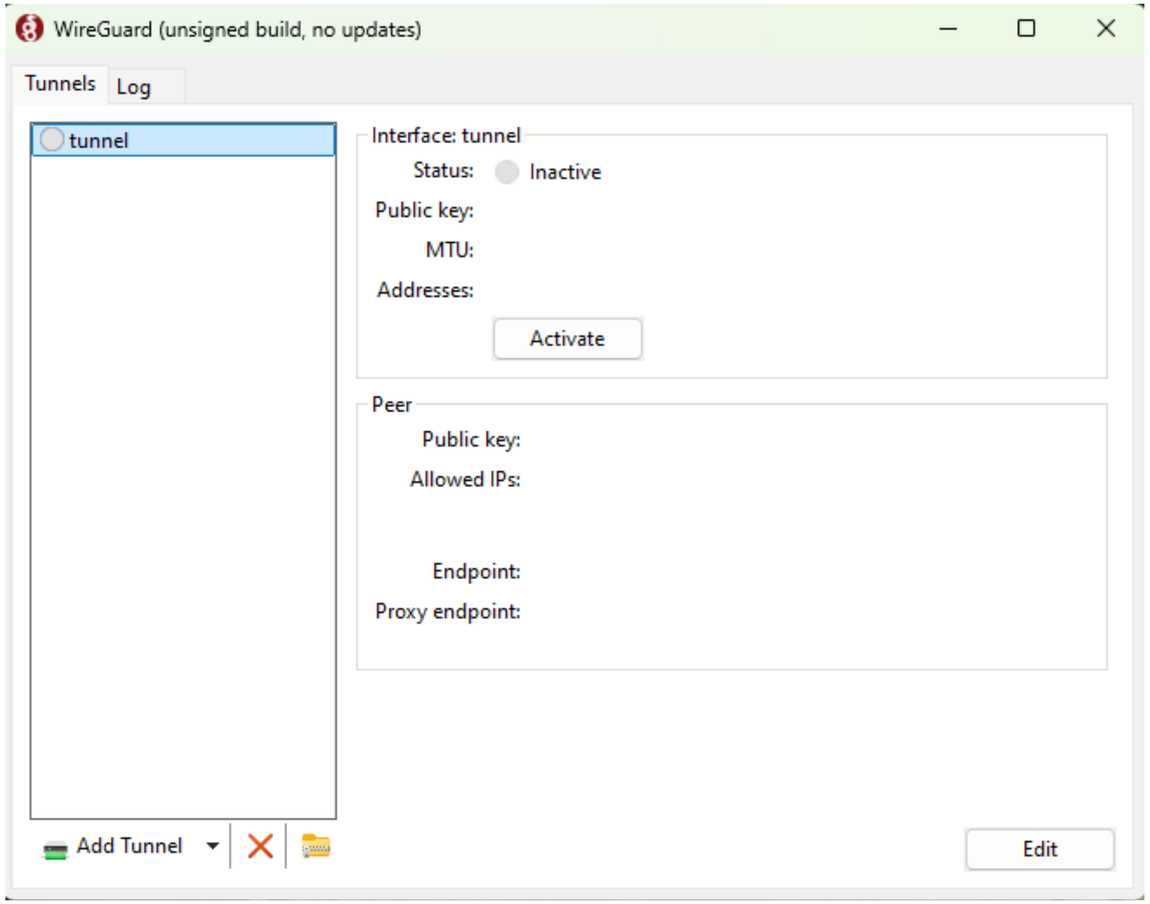


https://codeberg.org/amebis/wireguard-windows

ProxyGuard: Client implementations

🍴 https://codeberg.org/eduvpn/wireguard-go





https://codeberg.org/amebis/wireguard-windows

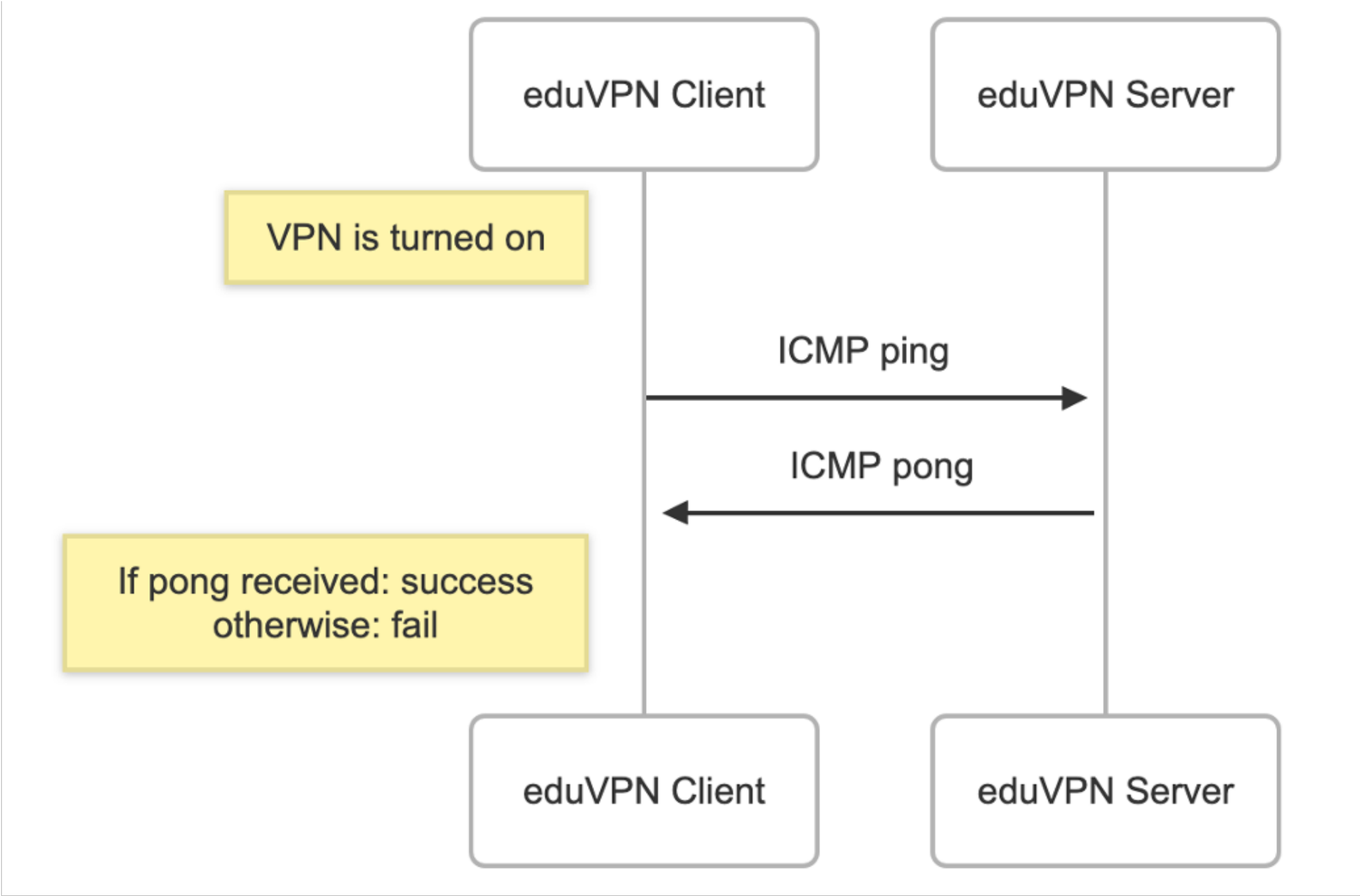
ProxyGuard: Challenges & Future plans

- Initial impl. easy
- Performance testing and improvements
- Roaming solution not ideal ☹️
- Linux daemon
  - Automatically restart on sleep, roaming
  - Get WireGuard properties dynamically

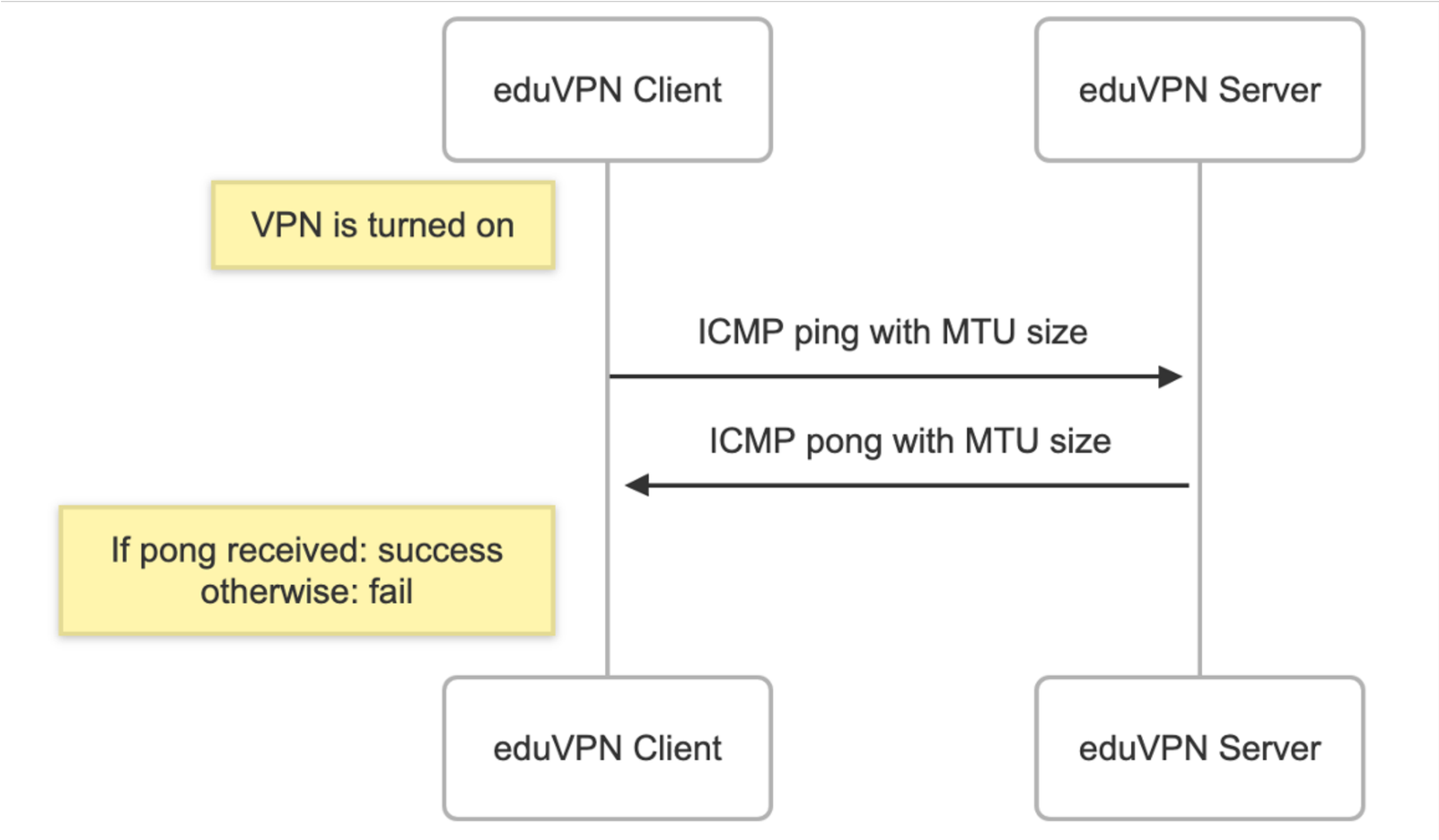
eduVPN: Automatic fallback

- ✅ TCP option for WireGuard
- How to determine when TCP is needed?
- ❌ Global option in the client
- Instead: Try to detect network issues

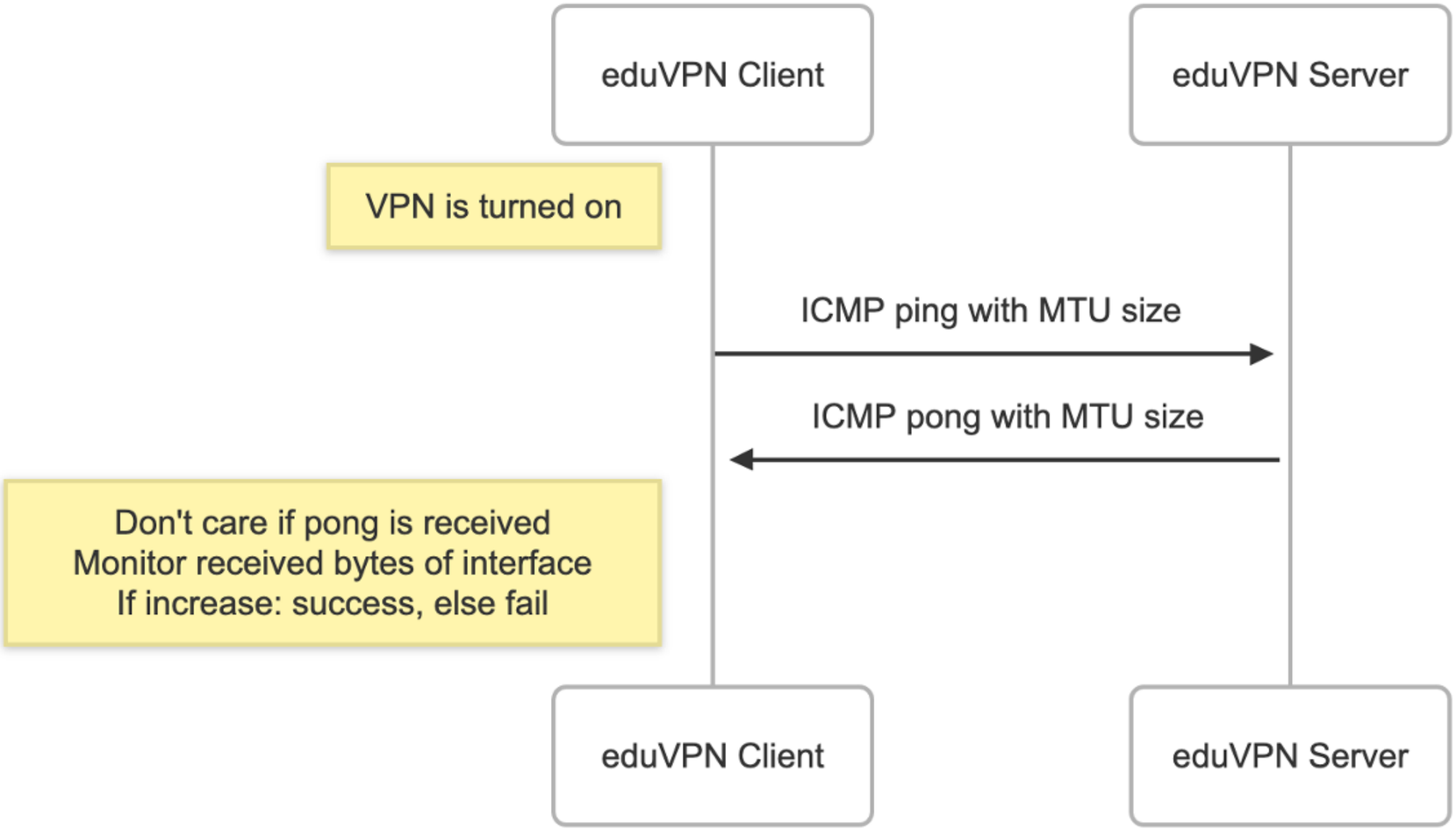
eduVPN: online detection



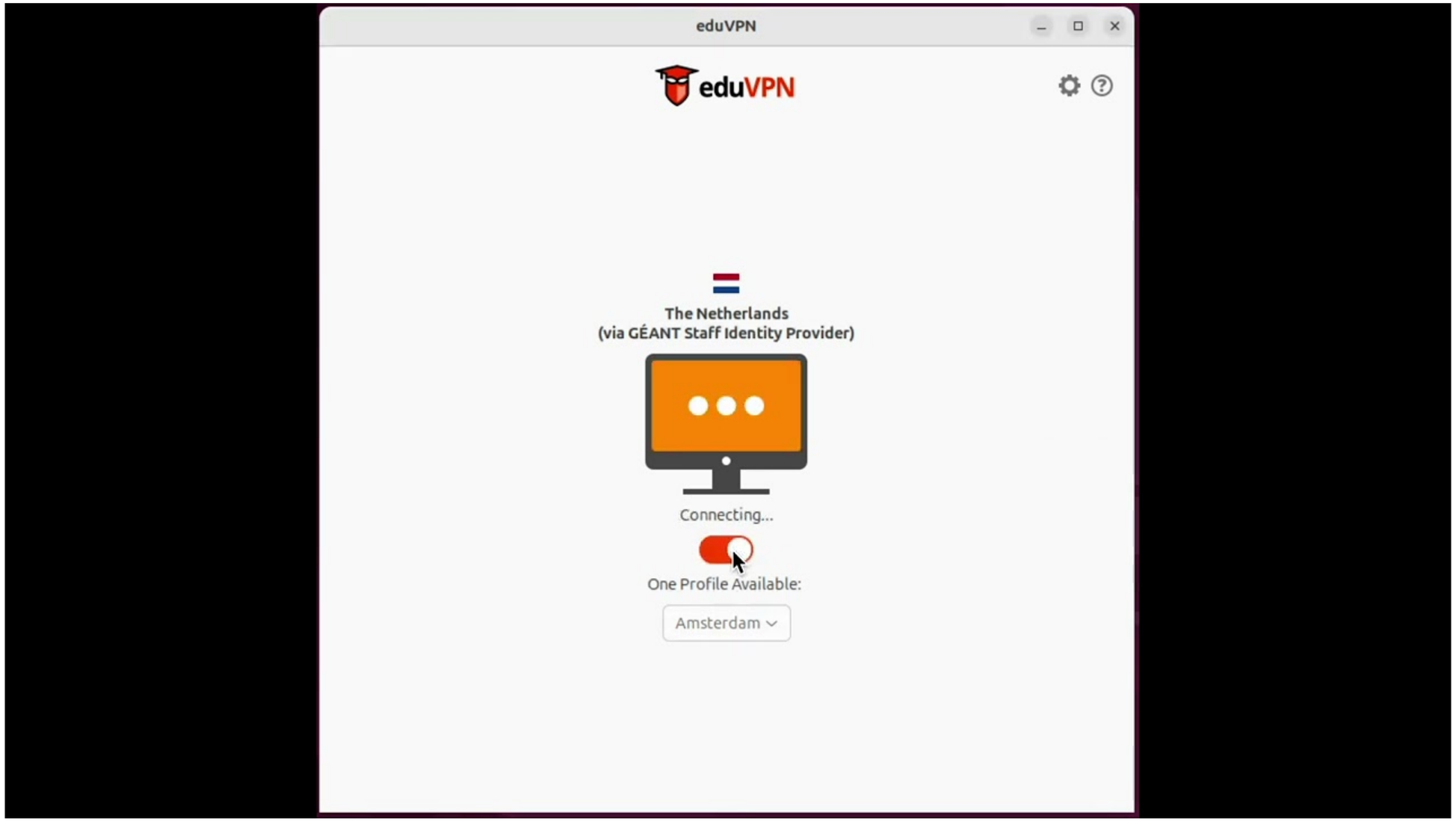
eduVPN: online detection



eduVPN: online detection



eduVPN: online detection



https://www.youtube.com/watch?v=FYxFI0r471Q

eduVPN: online detection

- Small traffic in the background
  - Phone 🏠
- ICMP ping needs root
  - Can do echo pings on Linux
  - Doesn’t work on all distributions ☹️

eduVPN: online detection

- Small UDP daemon: Sends UDP packets (‘pong’) of same size as request (‘ping’)
  - Wouldn’t require root ✅
  - Tests MTU ✅
  - Separate component ☹️
- Make online detection more dynamic?

Questions?

- 🏡 https://codeberg.org/eduvpn/proxyguard
- 📧 jeroen.wijenbergh@geant.org