Remote Access via WireGuard

Homelab Club 11/4/2025

Topics

- Remote access
- What is a VPN? What does it solve?
- Wireguard config
- Small network architecture tangent
- Limitations and solutions

Remote access

Some people are just into homelabbing for tinkering, but some also find tools they want to actually run and use.

Some that speak to me personally:

- Personal website
- Game servers
- Media server
- Network storage
- Password manager

As an aside: Exposing services to the Internet

Instead of exposing a VPN, I could expose “X” service directly to the Internet. However, exposing a service to the public is not a great idea unless you are purposefully expecting the public to use it.

- Not all services you host may be well audited and free of known vulnerabilities. Think hobby-project repos that take user input, built using old versions of popular libraries, and haven’t been updated in a couple years (but it does the one cool thing you want to do, so you really really really want to host it!)
- Even with well-maintained apps, if a vulnerability is found and fixed in an update, you need to keep on top of updates. The more services you host, the harder it may be to keep on top of updates for each individual service.
- Exposing services directly can make you a target when getting port-scanned, as tools can usually return what software is being hosted on a port (and sometimes the version!!)

Instead: Maintain one well-vetted endpoint

- Port scanners will only see a single endpoint and won’t be able to make guesses to what software you’re running inside unless they manage to break in.
- As long as you don’t accidentally backdoor yourself some other way (installing malicious software), you shouldn’t have to worry about direct attacks from outside.

Two common options

SSH bastion with SSH keys

VPN (e.g. OpenVPN, IPSec, WireGuard) ← Today’s focus

What is a VPN?

What is a VPN?

It’s an overlay network that allows you to access a private network over a public network.

Traffic is sent to the private network over an encrypted tunnel established between two public points.

- If all traffic is sent over the tunnel, it is a full tunnel (this is what you get with privacy VPN products)
- If only traffic to specific networks is sent over the tunnel (e.g. private IPv4 space), it is a split tunnel. Traffic outside of those specific networks gets sent normally

Why should you care about using a VPN?

- You can keep your self-hosted services inaccessible from the public Internet, and use a VPN to send traffic designated to these private services.
- If you want to ensure that any unencrypted protocols aren’t sniffable on public networks, you can run a full tunnel VPN.
  - This only really hides this information from your immediate gateway (like your Starbucks WiFi network, or eduroam). If the destination is still somewhere on the public Internet, the traffic will still get sent unencrypted after it leaves your VPN’s gateway).
    - Privacy VPN providers really aren’t clear enough about this limitation!

WireGuard

VPN protocol that has been gaining popularity recently.

- Goal: Be as easy to set up as SSH
- Asymmetric keys are used to validate node identity and are used to encrypt traffic.
- Designed deliberately to be barebones. This makes it relatively easy to audit, though OOTB it has a limited featureset.
- Supported on basically all platforms (Windows, macOS, Linux (both in-kernel and user-mode implementations) FreeBSD, iOS, Android)

WireGuard Pros/Cons (not comprehensive)

Pros

- Performant! Much more so than OpenVPN (at least with the userland implementation), similar to IPsec
- Relatively small attack surface. Less code to audit, less functionality to add bugs to during development.

Cons

- Barebones is a double-edged sword.
  - No native support for more traditional user management (username/password logins, ACLs)

Example configuration file (star configuration)

Central server

```
[Interface]
Address = 172.16.121.1/24
ListenPort = 51820
PrivateKey = WAkUMyCoolServerPrivateKeyDoNotSteal
#PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
#PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE


[Peer] # My MacBook Air
PublicKey = uvdNbSJv82JosnlTCG1DKPmxSs9NQ0fr0lZN2e8SeU4=
AllowedIPs = 172.16.121.4/32

[Peer] # My iPhone
PublicKey = 7yZEu2TtgTPrrcZM8SqTeV7FHDpp1SnKN46rWG0/ty0=
AllowedIPs = 172.16.121.6/32
```

Client (My MacBook Air)

```
[Interface]
PrivateKey = OLIlMyCoolClientPrivateKeyDoNotSteal
Address = 172.16.121.4/32
DNS = 172.16.121.1
[Peer]
PublicKey = I+kaYCKCTUR7D9CnHpyPWb5BYhZsBi18EPhEJoaXjEk=
AllowedIPs = 10.121.2.0/24, 10.121.3.0/24, 10.121.20.0/24, 10.121.121.0/24, 172.16.121.0/24
Endpoint = vpn.mycooldomain.com:51820
```

Possible issues

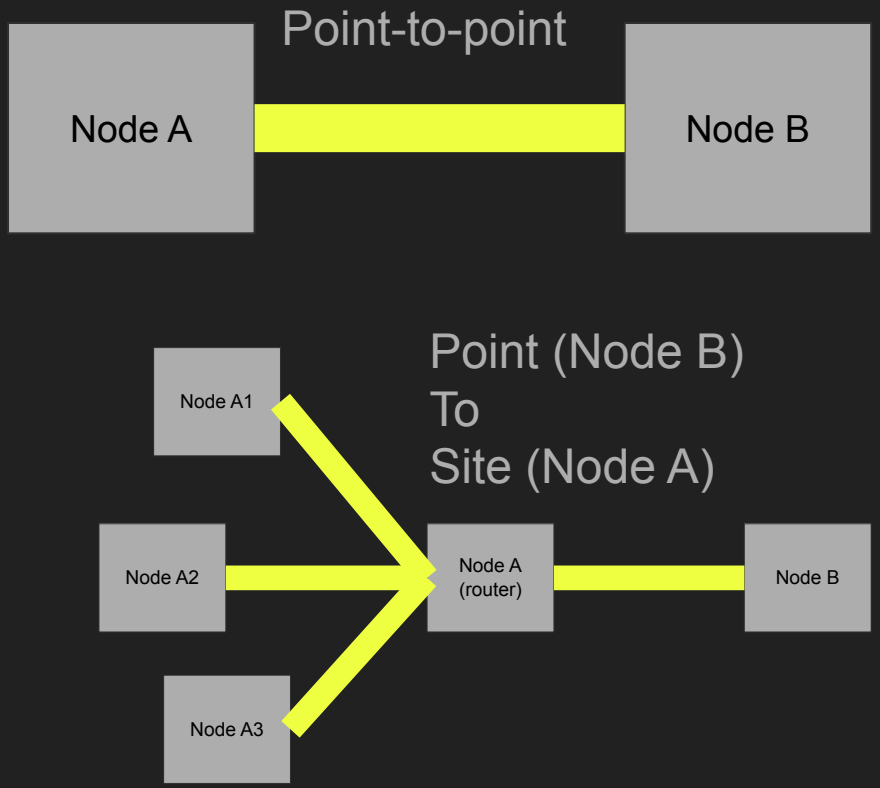
- What if my home IP address changes, common for residential ISPs?
  - Dynamic DNS. If you buy a domain name and set up a Dynamic DNS service, you can update your DNS periodically to the new IP.
- I want to limit client access to specific services
  - Firewall rules on the endpoint node can work, or on the service hosts themselves.
  - Depending on the platform, these rules might not scale easily to lots of devices.
- What if I can’t forward a port? (e.g. CGNAT, no IPv6)
  - If both sides are static endpoints, you might be able to UDP hole punch by defining the endpoint on both the client and server ends. Very dependent on firewall setups though.
- What if I UDP outbound traffic is blocked?
  - ...

Tangent: Network topologies and VPN setups

Point to point

- Node A and Node B know about each other.
- A and B could route traffic for other devices on their own network (point-to-site, or site-to-site)

Most performant, as connections between nodes are direct. However, scaling to more points/sites makes configuration exponentially more complex to configure. There are tools to handle this (foreshadowing).

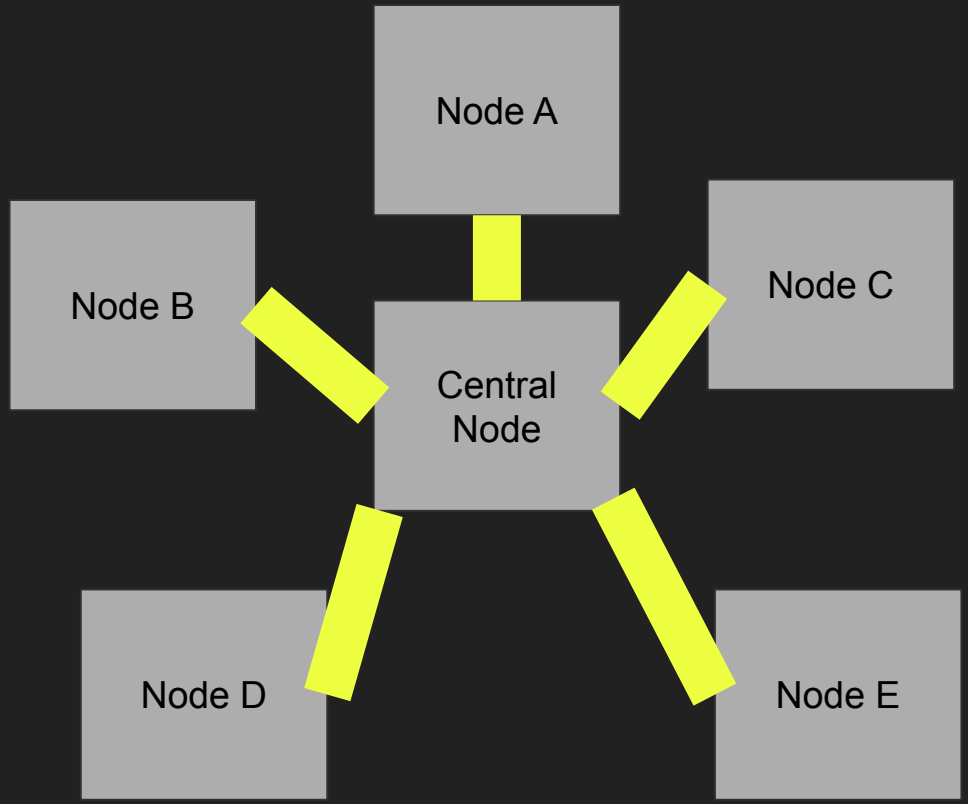


Tangent: Network topologies and VPN setups

Star

- All nodes route traffic through a central node.
- The central node can arbitrate access to other nodes in the star.
- Typically the central node needs an exposed port.

Simple to manage (nodes don’t need to know about each other’s keys), but latency can be an issue if endpoints are far from the central node.



Mesh networking solutions that use WireGuard

Mesh Networking is a term used nowadays to describe some services that establish point-to-point connections between devices. There is a publicly hosted coordinator that handles the public keys for these devices and helps them negotiate connections, even in nonoptimal network conditions.

- Because it knows the public IPs of both ends, UDP hole punching can potentially be done, firewalls permitting.
- If outbound UDP is blocked or source port randomization is done, they can relay UDP traffic inside of TCP.
- This makes connections possible even behind CGNAT!
- A potentially useful tool for those aiming for the aspirational “zero-trust” network.

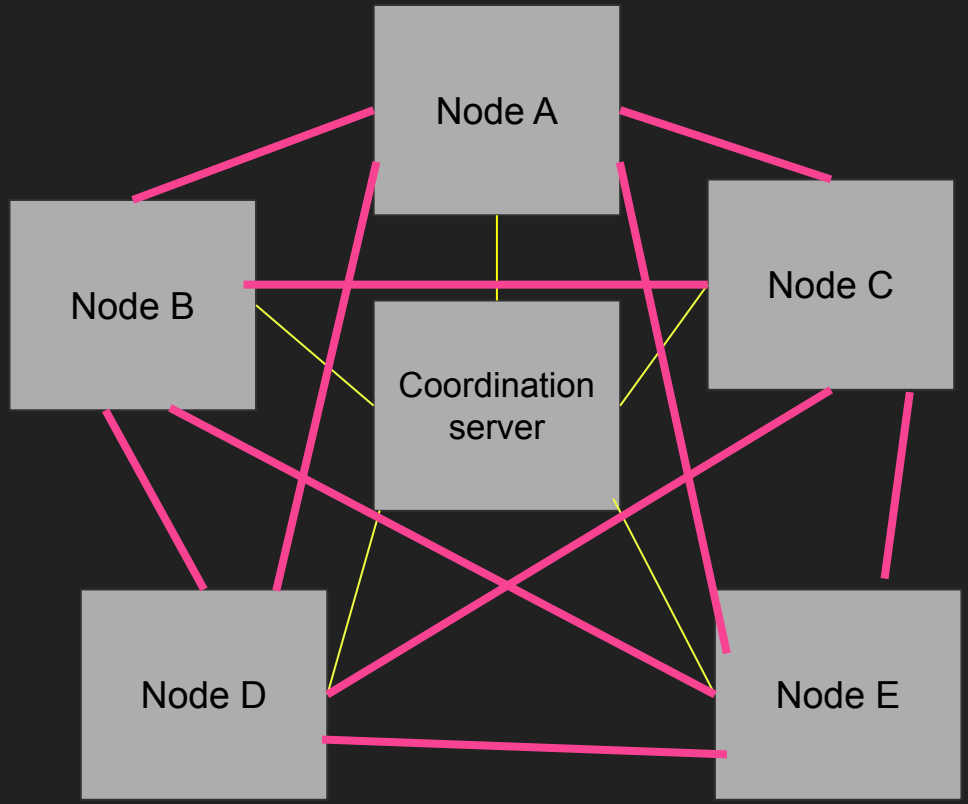
Mesh network setup

On the surface it may look like a star, but ideally the only data that goes to/from the coordination server is information about a node's public keys.

- Nodes establish direct VPN connections to each other.
- The coordination node can arbitrate access to other nodes by choosing not to share keys.
- The coordination node needs to be publicly accessible.

Low latency between nodes, and the coordination server handles key distribution, so the best of both worlds!

Yellow - coordination data
Pink - network traffic



Well-known WireGuard-based mesh network services

Tailscale - Cloud-hosted coordination service.

- More well-known than the others
- Probably the best for a set-and-forget, or if you are trying to share with other friends.
- Ties with existing SSO services pretty easily.

Headscale - Open reimplementation of Tailscale's coordination service, compatible with the Tailscale client

- Can be fully self-hosted
- CLI only.
- Can use pre-shared keys, otherwise you need to connect to external SSO or roll your own SSO (tl;dr authentication can be a pain).

Pangolin - Relatively new kid on the block

- Self-hosted or cloud-hosted
- Comes with its own authentication implementation.
- Can relay traffic over TCP (I think? It sounds like it's experimental)