WireGuard Capability Testing

Lincoln Bryant
Judith Stephen
University of Chicago

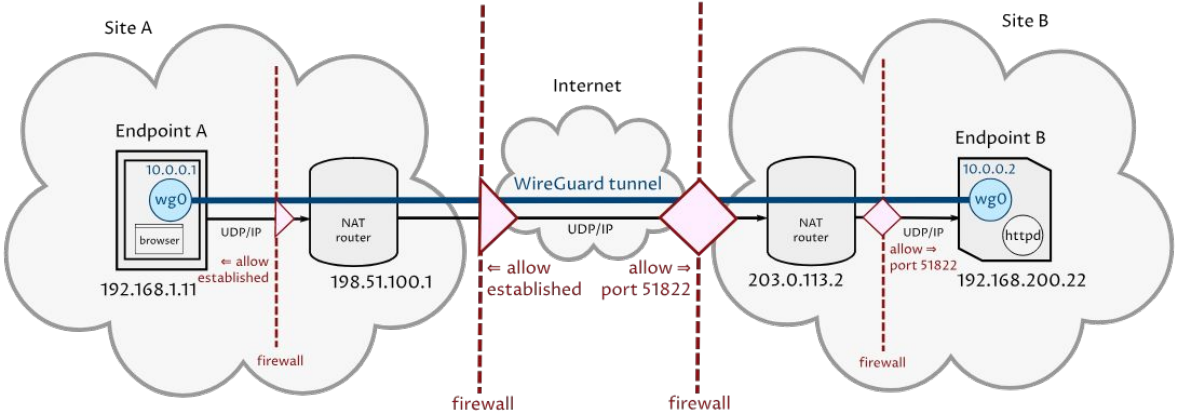
Aidan Rosberg
Indiana University

HTC 2025
June 5, 2025

About WireGuard

- Radically simple VPN software
  - Key exchange inspired by SSH, no X.509 or PKI
  - Static IP assignment to a Linux tunnel interface
  - Small codebase (~4K LoC vs 100K+ for OpenVPN or 400K+ for IPsec)
- Modern cryptographic standards
  - X25519 key change (Diffie–Hellman) for authentication
  - ChaCha20–Poly1305 for data transfer
- Operates at Layer 3
  - All data transport over UDP
- Merged into the mainline Linux Kernel (5.6+)
  - Available in EL9 as a Tech Preview
- Userspace implementations also exist, such as Cloudflare's BoringTun

What is it good for?



- Various modes of integration
  - Fully connected VPN mesh, every node has WireGuard installed and connected to the mesh
  - Strategically placed WireGuard nodes to act as gateways into the network
  - Direct application integration, with WireGuard lib compiled in

Ideas for applications

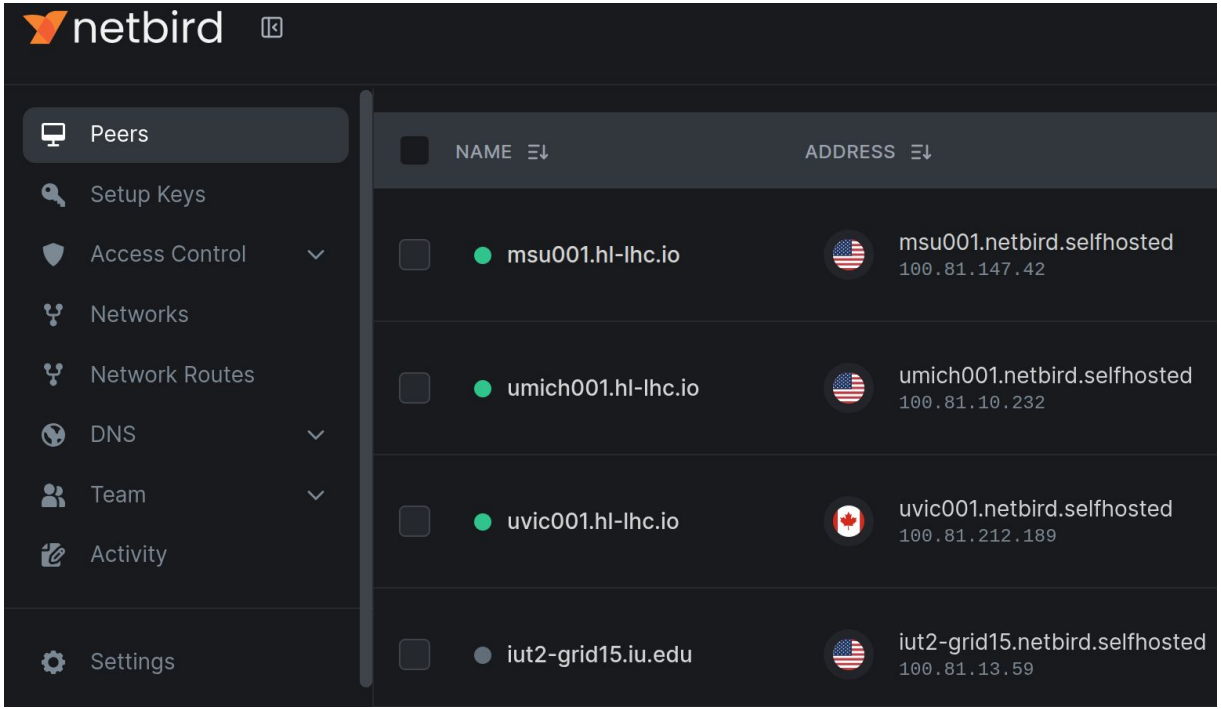
- Flat, private networks for cluster applications
  - Application only sees a private network, the underlying network topology is invisible
- Reasonably secure, wide-area filesystem mounts
  - Protocols like NFS encrypted in transport by WireGuard
- Distributed analysis facilities
  - Contribute resources from T1s, T2s, even your laptop :)

Control plane management

- WireGuard has no built-in notion of a control plane
  - YOU are responsible for allocating IPs to every node in the mesh
  - Key exchange in a fully connected mesh from every peer, to every peer, is not scalable!
- A few options: Tailscale, Netbird, others
  - Tailscale is perhaps the most popular:
    - Semi-open source, with a proprietary control plane
    - Alternative, open source control plane exists, support unclear
  - Netbird doesn't market nearly as well, but it is **fully open source** , so we went with it

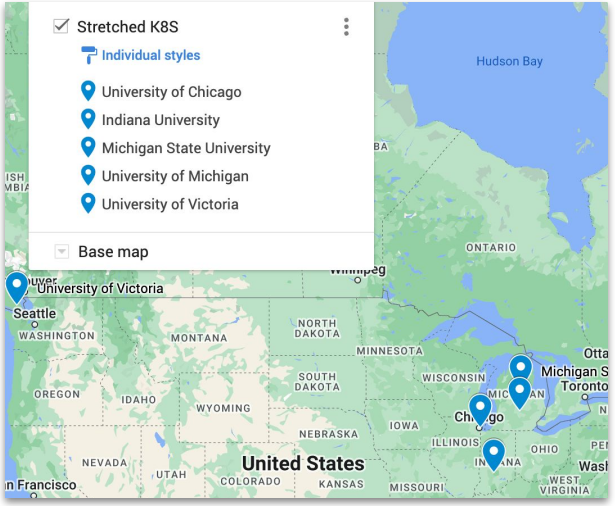
Netbird Features

- Peer management
  - Including ephemeral peers
- OAuth2 integration
- Internal DNS nameservers
- Liveness probing
- Networks, routes, etc



Kubernetes on top of WireGuard

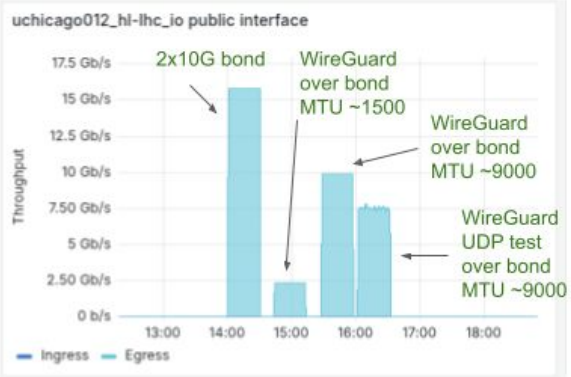
- The WireGuard networking model is powerful and flexible enough that you can build a working Kubernetes cluster on top of it
- WireGuard as an underlayment technology is simplifies the networking by being separate from, and effectively invisible to, Kubernetes
- Convenient for adding resources from sites that may not have a fully flexible firewall at the WAN.
    - For instance, some sites can't provide WAN-facing network service capability, but can provide disk, compute, etc

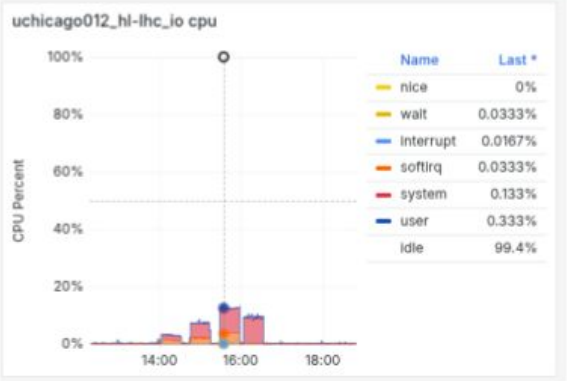


US ATLAS / ATLAS Canada Kubernetes Mesh

Some performance considerations & measurements

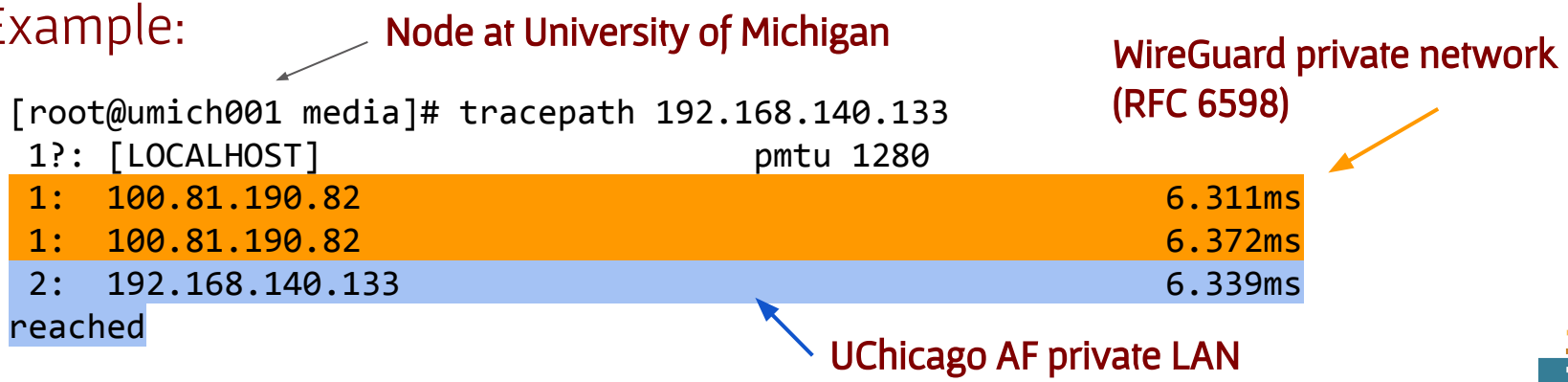
- The encryption used by WireGuard is not (today) offloadable to hardware
  - Non-negligible CPU usage encrypting/decrypting traffic
- WireGuard encapsulation requires 60 bytes of overhead for IPv4, 80 bytes for IPv6
- WireGuard performance with the kernel module in EL9 at ~MTU 1500 (minus overhead) is fairly bad compared to line rate
- Increasing the MTU to near-9000 improves things significantly, but still not line rate on a 2x10G bond
- CPU usage is reasonably high on a Sandybridge CPU with 28 hyperthreads





Gateway nodes

- Running WireGuard on all nodes can be challenging or undesirable
  - Access can still be provided via gateway nodes
- Doesn't provide the same level of full connectivity, but does allow applications to reach across the WireGuard Network to access resources
- Example:



WireGuard and Privilege

- Can we have containerized applications join or leave the mesh in an ephemeral way, without privilege? (Think glideins/pilots)
- Yes, with some specific requirements:
  - User namespaces >0, network namespaces >0
  - CAP_NET_ADMIN, CAP_NET_RAW
    - For creating tunnel devices and manipulating raw packets
  - CAP_SYS_MODULE
    - For loading the WireGuard kernel module (userspace implementation will work if needed)

WireGuard under Podman

- At SLAC, we successfully joined a proof-of-concept container to the mesh as an unprivileged user via:

```
podman run --cap-add=NET_ADMIN \
           --cap-add=NET_RAW \
           --cap-add=SYS_MODULE \
           --sysctl="net.ipv4.conf.all.src_valid_mark=1" \
           --sysctl="net.ipv4.conf.all.forwarding=1" \
           -v /lib/modules:/lib/modules \
           -v /dev/net:/dev/net
```

WireGuard under OpenShift

- OpenShift's security model presents additional challenges
  - All of the capabilities need to be included in a not-quite-root service account
- Issues mounting `/dev/tun` into the container
  - This seems to inescapably require rootly privileges
    - `hostPath` mounts in Kubernetes are dangerous and leaky
- Since OKD effectively requires WireGuard to run as root, perhaps a gateway node configuration is more appropriate
  - Cluster admin runs the gateway
  - Clients just 'use' the network

Cybersecurity Considerations

- In especially security-conscious environments, policy could be challenging
- Cybersecurity experts tell us that it's important for them to be able to inspect the unencrypted traffic
- If there's value from using WireGuard in these environments, perhaps a configuration with dedicated routing nodes would be appropriate

Conclusions/Summary

- WireGuard enables lightweight VPN meshes between sites, which may enable novel workloads
- WireGuard's design is robust enough to facilitate complex applications including Kubernetes
- Linux's containerization facilities are advanced enough to generally allow us to run WireGuard unprivileged on workers and other resources
- Performance is okay when tuned, but needs to be studied further, especially on newer hardware