NoRouter: instant multi-cluster & multi-cloud container networking

No routing configuration is required. No root privilege is required.

Akihiro Suda, NTT
What is NoRouter?

- Instant multi-cluster & multi-cloud networking for dev environments
- No public IP address is required
- No routing configuration is required
- No root privilege is required
- Just needs stdout (aka shell access)

https://norouter.io
What is NoRouter?
Goals and Non-Goals

Goals

• Facilitate working with heterogeneous dev environments e.g.,
  • Kubernetes cluster1 on GPU-enabled rich cloud
  • Kubernetes cluster2 on cheaper cloud
  • On-premise baremetal IoT devices
  • Laptop at home
Goals and Non-Goals

Goals

• UX
  • Human-friendly CLI and YAML

• Security
  • No need to sacrifice security with `docker run --privileged`

• Portability
  • Mostly for Docker/Kubernetes containers, but not only for them
  • Works with Docker, Podman, LXC, Kubernetes, SSH, and whatever, as long as stdio is available
Goals and Non-Goals

Non-Goals

- Production quality performance
  - Approximately 350 Mbps at maximum, with two Docker containers on same host

- Fault-tolerance
  - Could be achieved by running NoRouter with a distributed locker, e.g., Consul, though
Similar tools

- `ssh -L`, `ssh -R`
  - Depends on SSH
  - No connectivity across multiple remote hosts
Similar tools

- VDE (Virtual Distributed Ethernet)
  - Requires root to create TAP devices
    (VDE itself doesn’t require the root)

- SLiRP (c. 1995)
  - No connectivity across multiple remote hosts
### Demo: Laptop + GKE + AKS

#### Virtual network 127.0.42.0/24

- **127.0.42.100:8080**: port 80 of the local laptop
- **127.0.42.101:8080**: port 80 of “gkepod” on Kubernetes context “gke”
- **127.0.42.102:8080**: port 80 of “akspod” on Kubernetes context “aks”

```
hosts:
laptop:
  vip: "127.0.42.100"
gkepod:
  vip: "127.0.42.101"
  cmd: "kubectl --context=gke exec -i gkepod -- norouter"
akspod:
  vip: "127.0.42.102"
  cmd: "kubectl --context=aks exec -i akspod -- norouter"
hostTemplate:
  ports: ["8080:127.0.0.1:80"]
```

GKE: Google Kubernetes Engine, AKS: Azure Kubernetes Service
How it works

• Each of the hosts has `norouter` binary

• NoRouter manager process (on local laptop) launches NoRouter agent processes, e.g., `kubectl exec -i <POD> norouter`

• Agents send virtual L3 packets to the manager via stdio, and the manager works like a switch
How it works: Multi-loopback

- **Challenge**: How to create network devices without the root?

- TUN/TAP cannot be used because it requires the root (CAP_NET_ADMIN)

- **Solution**: Do not create devices at all

- NoRouter just uses the loopback interface with multiple IP addresses within 127.0.0.0/8
  - e.g. 127.0.42.100, 127.0.42.101, …
How it works: Multi-loopback

127.0.42.102

kubectl exec

127.0.42.100

$ norouter hosts.yaml

127.0.42.103

lxc exec

127.0.42.104

docker exec

ssh
How it works: TCP/IP stack

• TCP/IP is implemented in userspace using Netstack
  • Originates from gVisor and Fuchsia
  • Written in Go

• The current NoRouter implementation only supports TCP (v4)

• UDP support is on plan
How it works: Name resolution

• **Challenge:** `/etc/{resolv.conf, hosts}` cannot be modified without the root

• **Solutions:**
  - `$HOSTALIASES` file (`~/norouter/agent/hostaliases`)
    - Similar to `/etc/hosts` but customizable without the root
    - Not supported by all applications
  
  - HTTP proxy mode
    - NoRouter agent works as a HTTP proxy with built-in name resolver
    - Best fit for typical HTTP applications
  
  - SOCKS proxy mode
    - Similar to HTTP proxy mode but SOCKS
    - Supports both SOCKS4a and SOCKS5
VPN(-ish) using HTTP proxy mode

- HTTP proxy mode can be used as if it is a “VPN”

- Accesses to “http://<PRIVATE-IP>.eu-central-1.compute.internal” are routed via `ssh aws_bastion`

- Same applies to Azure and GCP addresses

```
hosts:
  local:
    vip: "127.0.42.100"
    http:
      listen: "127.0.0.1:18080"
  aws_bastion:
    cmd: "ssh aws_bastion -- norouter"
    vip: "127.0.42.101"
  azure_bastion:
    cmd: "ssh azure_bastion -- norouter"
    vip: "127.0.42.102"
  gcp_bastion:
    cmd: "ssh gcp_bastion -- norouter"
    vip: "127.0.42.103"
routes:
  - via: aws_bastion
to: ["*.compute.internal"]
  - via: azure_bastion
to: ["*.internal.cloudapp.net"]
  - via: gcp_bastion
to: ["*.example-123456.internal"]
```
How to get started

- Binaries are available for Linux, FreeBSD, NetBSD, OpenBSD, DragonFly BSD, macOS, and Windows: https://github.com/norouter/norouter/releases

- `norouter show-example` shows an example YAML

- `norouter -e` opens $EDITOR with an example YAML

- Docs: https://norouter.io/docs/
Future work

- Support UDP
- Support MASQUE (HTTP/3 VPN-ish)
- Support TUN/TAP (with root)
- Automatically generate mTLS certs with NoRouter virtual IP addresses
- …
Recap

• Instant multi-cluster & multi-cloud networking for dev environments

• No public IP address is required

• No routing configuration is required

• No root privilege is required

• Just needs stdio (aka shell access)

https://norouter.io