

# Eine kleine Tour durch Linux als Netzwerk-Gerät

jo

May 4, 2024

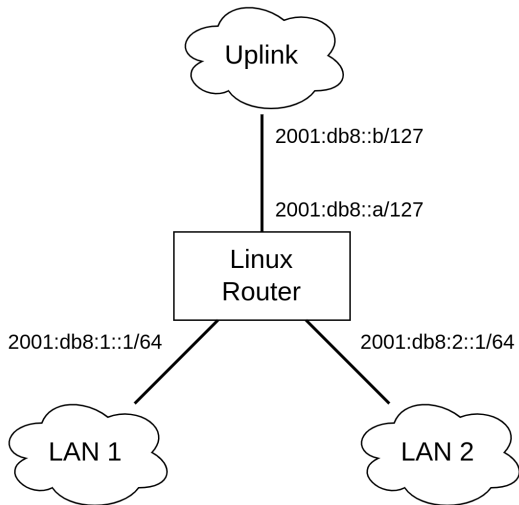
# iproute2

- ▶ ip address, ip route und ip link Befehle

```
ip \
  -6 \ # IPv6
  -c \ # color
  -d \ # details
  -br \ # brief
  -j \ # JSON output
  $COMMAND
```

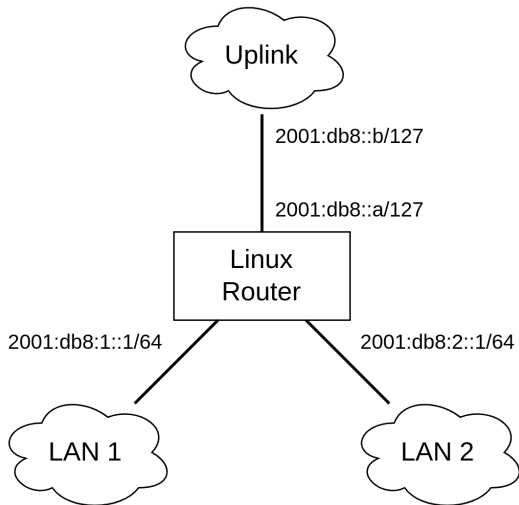
- ▶ Viele Beispiele: <https://baturin.org/docs/iproute2/>

# Beispiel Router



- ▶ Einfacher Router
- ▶ keine Firewall
- ▶ passende Config beim Uplink wird angenommen

# Routing in Linux

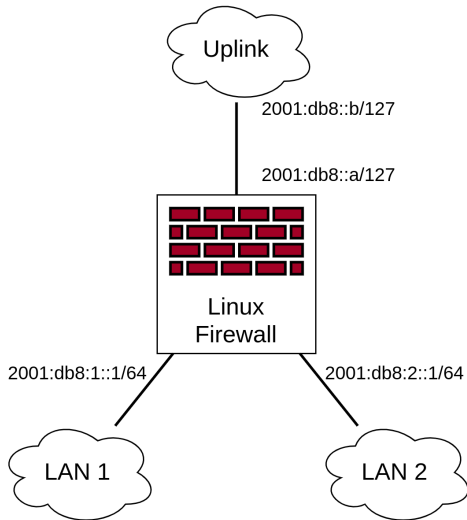


- ▶ IP Forwarding anschalten

```
# /etc/sysctl.conf
net.ipv6.conf.all.forwarding=1
net.ipv4.ip_forward=1
```
- ▶ ggf. Default-Route setzen

```
ip -6 route add ::/0 via
    ↪ 2001:db8::b dev uplink
```

# Simple Firewall



► Ziel: LAN 2 nur von LAN 1 erreichbar

# nftables

- ▶ Packet Classification Framework
- ▶ Ersatz für xtables (ip(6)tables, ebtables, arptables, ...)
- ▶ Pakete filtern, markieren, rate-limiten, Address-Translation, ...



# nftables

## Families

- ▶ bestimmen, welche Arten von Paketen verarbeitet werden
- ▶ i.d.R. braucht man `ip`, `ip6` und `inet` (IPv4+IPv6)
- ▶ 

```
table inet filter {  
    ...  
}
```
- ▶ für mehr siehe Doku (`man nft`)



# nftables

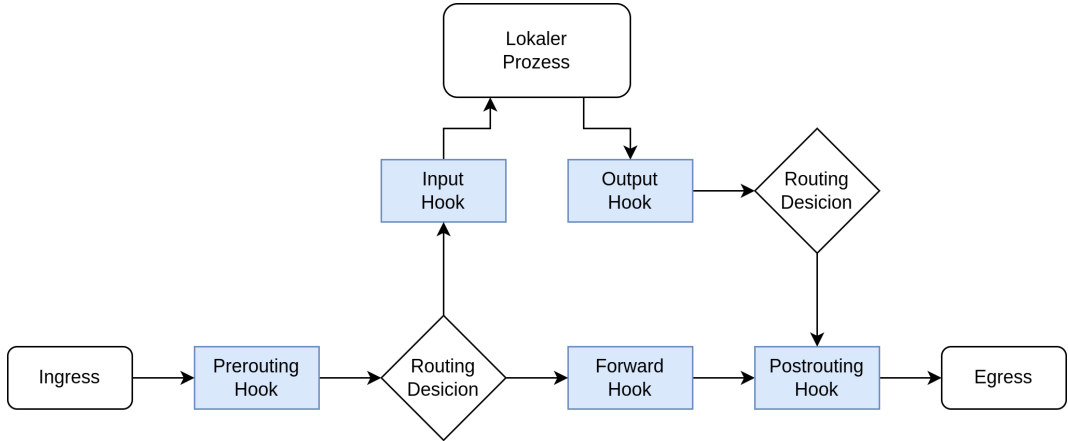
## Chain Types

- ▶ bestimmen, was in einer Chain gemacht werden kann
- ▶ i.d.R. filter, nat für NAT
- ▶ 

```
table inet filter {  
    chain input {  
        type filter hook input priority 0; policy drop;  
        ...  
    }  
}
```
- ▶ für mehr siehe Doku (man nft)

# nftables

## Hooks



<sup>0</sup>vereinfachte Version von

[https://wiki.nftables.org/wiki-nftables/index.php/Netfilter\\_hooks](https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks)

# nftables

## Priority

- Bestimmt wann eine Chain ausgeführt wird im Verhältnis zu anderen Funktionen

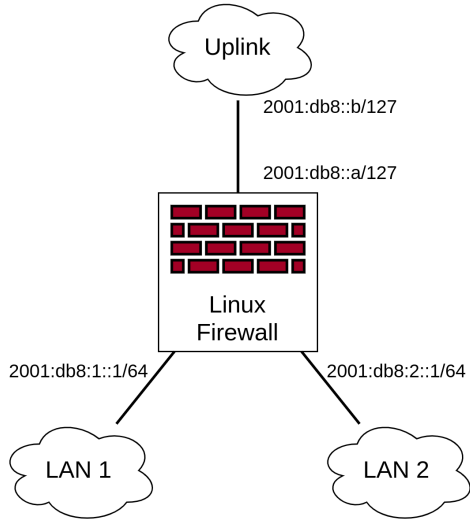
Priority	Name	Description
- 400		Defragmentation
- 200		Conntrack: associate packets with connections
- 100	dstnat	Destination NAT
0	filter	Filter Tables
100	srcnat	Source NAT
INT_MAX		Conntrack: adds new tracked connections

- Nur ein Ausschnitt <sup>1</sup>

---

<sup>1</sup>Mehr Infos: [https://wiki.nftables.org/wiki-nftables/index.php/Netfilter\\_hooks](https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks)

# Simple Firewall



► Ziel: LAN 2 nur von LAN 1 erreichbar

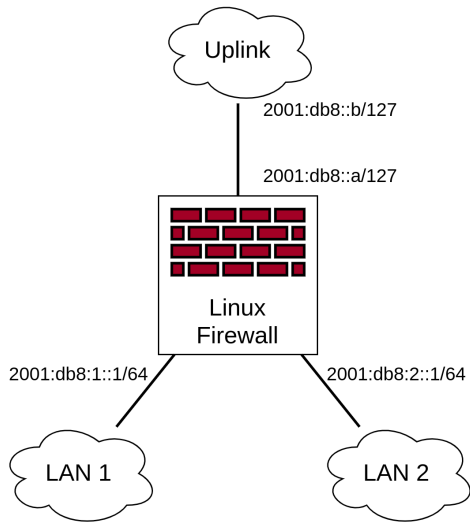
# Simple Firewall

## Config

```
flush ruleset
table inet filter {
    chain forward_filter {
        type filter hook forward priority 0; policy accept;

        ip6 saddr 2001:db8:1::/64 ip6 daddr 2001:db8:2::/64 accept;
        ip6 daddr 2001:db8:2::/64 reject with icmpx type admin-prohibited;
    }
}
```

# Simple Firewall



- ▶ Ziel: LAN 2 nur von LAN 1 erreichbar
- ▶ Problem: Pakete kommen zwar aus LAN 2 raus, der Rückweg funktioniert nicht (bis auf aus LAN 1).

# conntrack

- ▶ was macht conntrack?
- ▶ Ordnet Pakete einer Verbindung zu
- ▶ auch für Protokolle, die nicht Verbindungsorientiert sind (Bsp: UDP)
- ▶ viele Details zu den Innereien:

`https://thermalcircle.de/doku.php?id=blog:linux:connection\_tracking\_1\_modules\_and\_hooks`

# conntrack

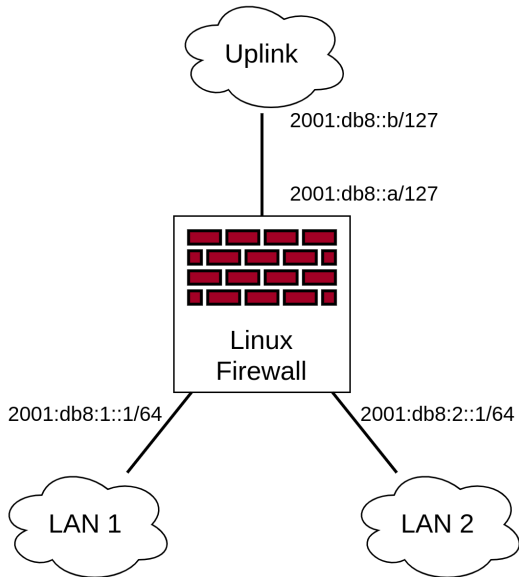
## States

- jedes Paket bekommt einen State

state	Description
new	Paket ist das erste einer Verbindung
established	Paket gehört zu einer bestehenden Verbindung
related	Paket steht in Zusammenhang mit einer bestehenden Verbindung (Bsp: ICMP Fehler)
invalid	Paket ist nicht das erste einer Verbindung, kann aber keiner bestehenden Verbindung zugeordnet werden
untracked	Paket soll nicht getrackt werden



# Simple Firewall



- ▶ Ziel: LAN 2 nur von LAN 1 erreichbar
- ▶ Problem: Pakete kommen zwar aus LAN 2 raus, der Rückweg funktioniert nicht (bis auf aus LAN 1).
- ▶ Die Antwortpakete haben Conntrack State established

# Simple Firewall

## Config

```
flush ruleset
table inet filter {
    chain forward_filter {
        type filter hook forward priority 0; policy accept;

        ct state {related, established} accept;

        ip6 saddr 2001:db8:1::/64 ip6 daddr 2001:db8:2::/64 accept;
        ip6 daddr 2001:db8:2::/64 reject with icmpx type admin-prohibited;
    }
}
```

# conntrack

## Tuning

- ▶ Maximale Anzahl der Verbindungen die getrackt werden

- ▶ Default-Settings i.d.R. relativ klein

```
cat /proc/sys/net/netfilter/nf_conntrack_max  
65536
```

- ▶ Aktuellen Wert anzeigen:

```
sudo conntrack -C  
11
```

- ▶ Wenn voll: Nachrichten in dmesg

```
[14008.628000] nf_conntrack: table full, dropping packet
```

---

<sup>1</sup>further details: [https://wiki.khnet.info/index.php/Conntrack\\_tuning](https://wiki.khnet.info/index.php/Conntrack_tuning)

# conntrack

## Tuning

- Größe der Conntrack-Table und Hash-Map hochdrehen

```
echo "$CONNTRACK_MAX" > \  
/proc/sys/net/netfilter/nf_conntrack_max  
echo "$HASHSIZE" > \  
/sys/module/nf_conntrack/parameters/hashsize
```

---

<sup>1</sup>further details: [https://wiki.khnet.info/index.php/Conntrack\\_tuning](https://wiki.khnet.info/index.php/Conntrack_tuning)

# conntrack

- ▶ Timeout-Werte für getrackte Sessions

```
# sysctl net.netfilter --pattern tcp_timeout
```

```
...
```

```
[...].nf_conntrack_tcp_timeout_established =\
```

```
432000 <-- 5 Tage
```

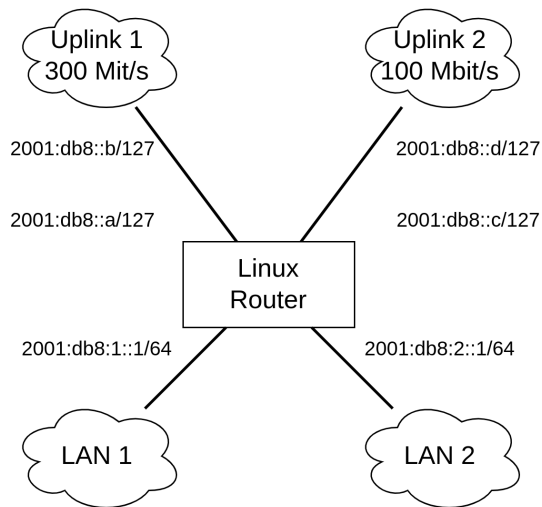
```
...
```

- ▶ Timeout sehr lang → abgebrochene Verbindungen füllen conntrack table
- ▶ Was ist ein guter Wert?
- ▶ irgendwas >2h damit TCP Keepalives funktionieren

# Mehrere Uplinks



# Mehrere Uplinks



► Ziel: Traffic über mehrere Uplinks verteilen

# Mehrere Uplinks

## Egress Load Balancing

- ▶ einfach: Multipath Route

```
ip route add ::/0 \  
    nexthop via 2001:db8::b dev uplink1 weight 3 \  
    nexthop via 2001:db8::d dev uplink2 weight 1
```

- ▶ verteilt Pakete über mehrere Nexthops nach Gewichtung
- ▶ Achtung: Pakete der selben Verbindung sollten den selben Pfad nehmen

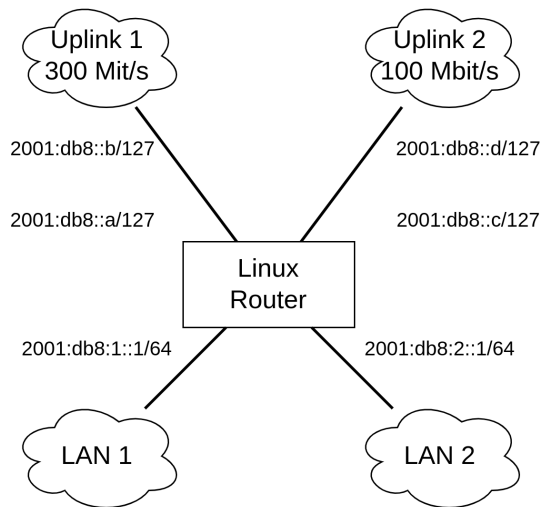


# Mehrere Uplinks

## Ingress Load Balancing

- Problem: Ingress-Pfad nicht direkt in unserer Kontrolle

# Policy-Based Routing



- Problem: verschiedener Traffic soll andere Wege nehmen
- Bsp: LAN 1 via Uplink 1, LAN 2 via Uplink 2

# Policy-Based Routing

- ▶ Normales Routing: Ziel-Adresse → Routingtabelle
- ▶ Policy-Based Routing: Regel-Liste → quasi arbiträres Matching → Routingtabelle

# Policy-Based Routing

- ▶ Routingtabellen werden implizit angelegt, wenn eine Route hinzugefügt wird
- ▶ Route in Routingtabelle 10 hinzufügen

```
ip route add ::/0 via 2001:db8::a/127 dev uplink1 table 10
ip route add 2001:db8:1::/64 dev lan1 table 10
ip route add 2001:db8:2::/64 dev lan2 table 10
```

- ▶ Routingtabelle anzeigen

```
# ip -6 route show table 10
2001:db8:1::/64 dev lan1 metric 1024 pref medium
2001:db8:2::/64 dev lan2 metric 1024 pref medium
default via 2001:db8::a dev uplink1 metric 1024 pref medium
```

- ▶ analog für table 20

# Policy-based Routing

- Policy Routing Regeln anlegen

```
ip -6 rule add from 2001:db8:1::/64 lookup 10
```

```
ip -6 rule add from 2001:db8:2::/64 lookup 20
```

- Regeln anzeigen:

```
# ip -6 rule
```

```
0:      from all lookup local
```

```
32764:   from 2001:db8:2::/64 lookup 20
```

```
32765:   from 2001:db8:1::/64 lookup 10
```

```
32766:   from all lookup main
```

# Policy-based Routing

## ► Test von lan1

```
# mtr 2001:db8:3::3 --report
```

```
Start: 2024-05-01T21:19:16+0200
```

HOST: lan1	Loss%	Snt	Last	Avg	Best	Wrst
1.  -- 2001:db8:1::1	0.0%	10	0.3	0.2	0.2	0.3
2.  -- 2001:db8::a	0.0%	10	0.2	0.2	0.2	0.2
3.  -- 2001:db8:3::3	0.0%	10	0.2	0.2	0.2	0.3

## ► Test von lan2

```
# mtr 2001:db8:3::3 --report
```

```
Start: 2024-05-01T21:20:58+0200
```

HOST: lan2	Loss%	Snt	Last	Avg	Best	Wrst
1.  -- 2001:db8:2::1	0.0%	10	0.2	0.2	0.2	0.3
2.  -- 2001:db8::c	0.0%	10	0.2	0.2	0.2	0.3
3.  -- 2001:db8:3::3	0.0%	10	0.2	0.2	0.2	0.3

# Network Namespaces

- ▶ komplett getrennte Netzwerk-Umgebung
  - ▶ eigene Routingtabellen
  - ▶ eigene Firewall-Regeln
  - ▶ eigene Interfaces
- ▶ hilfreich für Tests von Netzwerksoftware
- ▶ Netzwerk testen
- ▶ Baustein für Container

# Network Namespaces

- ▶ Network Namespace anlegen

```
# ip netns add $NAME
```

- ▶ Interface in den Namespace verschieben

```
# ip link set $IFACE netns $NAME
```

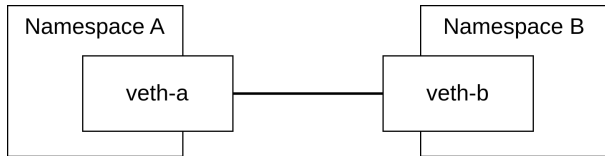
- ▶ Programm im Namespace ausführen

```
# ip netns exec $NAME $COMMAND
```



# Network Namespaces

veth-pairs



- Network-Namespaces mit einander verbinden

- Virtuelles Netzwerkkabel: veth-pair

```
# ip link add name veth-a type veth peer name veth-b
```

- ```
# ip a
```

```
4: veth-b@veth-a: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 ...  
    link/ether a2:75:8b:4c:00:f0 brd ff:ff:ff:ff:ff:ff
```

```
5: veth-a@veth-b: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 ...  
    link/ether 6a:70:06:18:bc:13 brd ff:ff:ff:ff:ff:ff
```

- ```
# ip link set veth-a netns A ...
```

# Referenzen I

- [1] *Connection tracking (conntrack) - Part 1: Modules and Hooks [Thermalcircle.de]*. URL: [https://thermalcircle.de/doku.php?id=blog:linux:connection\\_tracking\\_1\\_modules\\_and\\_hooks](https://thermalcircle.de/doku.php?id=blog:linux:connection_tracking_1_modules_and_hooks) (visited on 04/30/2024).
- [2] *Conntrack tuning – KHnetWiki*. URL: [https://wiki.khnet.info/index.php/Conntrack\\_tuning](https://wiki.khnet.info/index.php/Conntrack_tuning) (visited on 04/07/2024).
- [3] *Linux Advanced Routing & Traffic Control HOWTO*. URL: <https://tldp.org/HOWTO/Adv-Routing-HOWTO/> (visited on 04/28/2024).
- [4] *Netfilter hooks - nftables wiki*. URL: [https://wiki.nftables.org/wiki-nftables/index.php/Netfilter\\_hooks](https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks) (visited on 04/07/2024).
- [5] *Task-centered iproute2 user guide — Daniil Baturin*. URL: <https://baturin.org/docs/iproute2/> (visited on 05/01/2024).