LOAD BALANCING MIT DNSDIST

(DDI User Group 2021)

CARSTEN STROTMANN

Created: 2021-06-18 Fri 09:49

WER SPRICH HIER?

Carsten Strotmann

dnsworkshop.de

blog.defaultroutes.de

DNS(SEC)/DANE/DHCP/IPv6 Trainer und Helfer

RIPE/IETF

WAS IST DNSDIST

- *dnsdist* ist ein Open-Source DNS load-balancer
  - Homepage: https://dnsdist.org
  - Lizenz: GPL Version 2
- *dnsdist* wird von PowerDNS.COM B.V erstellt und betreut
  - *dnsdist* ist unabhängig vom PowerDNS autoritativen DNS server und vom PowerDNS Recursor (die Produkte teilen sich Quellcode)
  - *dnsdist* kann zusammen mit anderen DNS Servern benutzt werden (BIND 9, NSD, Windows DNS, Unbound …)

DNSDIST BESONDERHEITEN (1)

- *dnsdist* empfängt DNS Anfragen und leitet diese an nachfolgende DNS Resolver oder autoritative DNS Server weiter
  - fail-over oder load-balancing Regeln steuern diese Weiterleitung
- Antworten können im *dnsdist* gecached werden
- *dnsdist* kann Angriffe und DNS-Missbrauch erkennen und abwehren
- DNS-over-TLS und DNS-over-HTTPS Unterstützung
- DNScrypt Unterstützung

DNSDIST BESONDERHEITEN (2)

- eBPF Socket Filter (Linux)
- Einfache aber mächtige Konfigurations-Möglichkeiten mit der Programmiersprache Lua
- Kann im Betrieb dynamisch umkonfiguriert werden (ohne Neustart)
- Remote HTTP API
- Eingebauter Web-Server für API-Zugriffe, Monitoring und Statistiken

INSTALLATION UND KONFIGURATION

BETRIEBSSYSTEM PAKETE

- *dnsdist* ist in vielen Paket-Repositories von Unix/Linux Systemen vorhanden
  - Debian/Ubuntu
  - Fedora
  - Red Hat EL / CentOS (via EPEL)
  - Arch Linux (AUR)
  - NixOS
  - FreeBSD / NetBSD / OpenBSD / DragonFlyBSD
  - pkgsrc (Cross-Platform https://www.pkgsrc.org)
- Die Betriebssystem-Repositories haben jedoch nicht immer die aktuelle Programmversion!

POWERDNS REPOSITORIES

- PowerDNS.COM B.V. bietet Repositories mit den aktuellen Versionen und der Entwicklungsversion für …
  - Debian 9/10
  - Raspbian/RaspberryOS 9/10
  - Ubuntu LTS 16.04/18.04/20.04
  - CentOS 7/8 (benötigt Abhängigkeiten aus den EPEL Repositories)
- Informationen über diese Repositories finden sich unter https://repo.powerdns.com/
- PowerDNS.COM B.V. (Teil von Open Xchange https://www.open-xchange.com) bietet kommerziellen Support für *dnsdist*

INSTALLATION AUS DEM QUELLEN

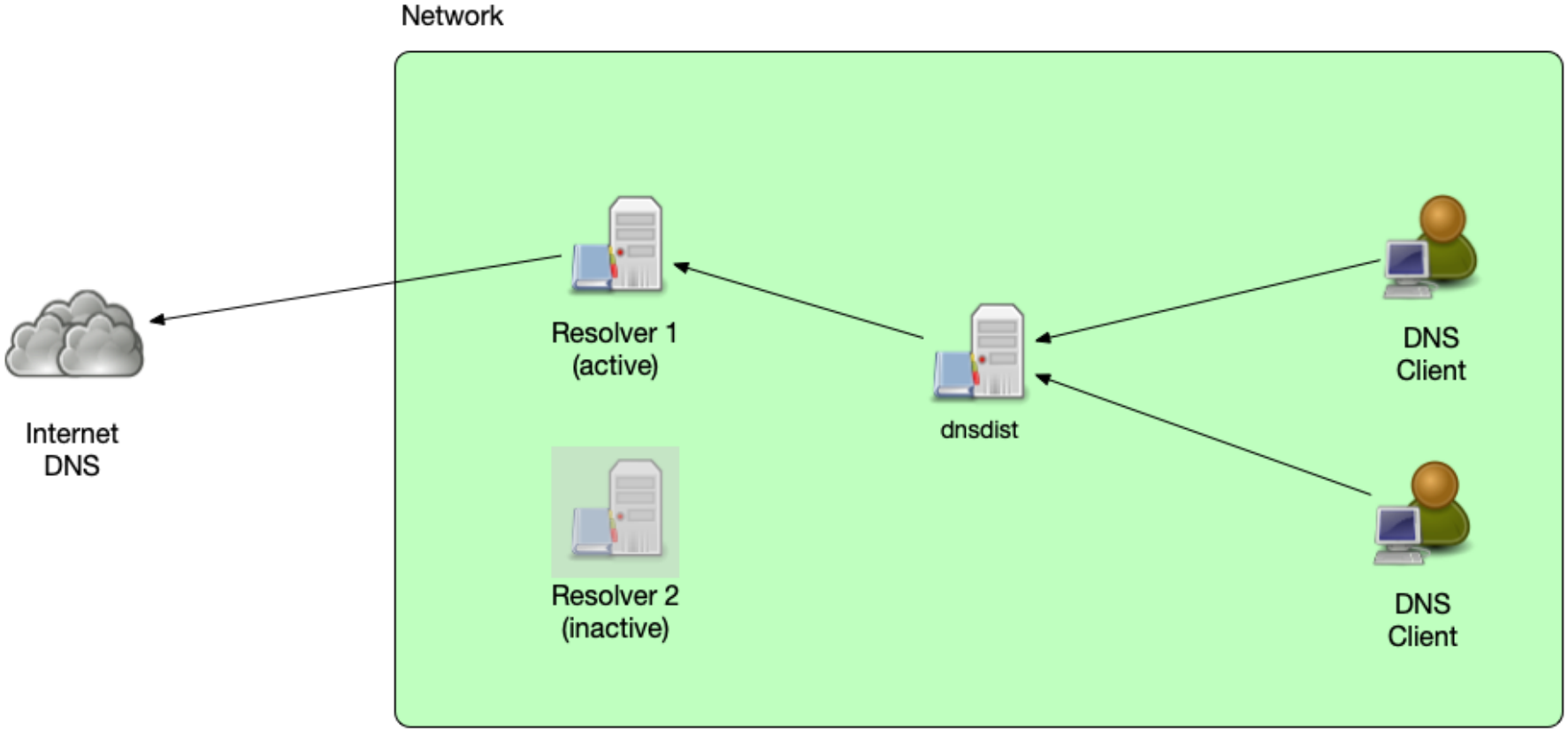
- *dnsdist* kann aus den Programmquellen übersetzt und installiert werden
- Abhängigkeiten:
  - Boost
  - Lua 5.1+ oder LuaJit
  - Editline (libedit)
  - libsodium (Optional)
  - protobuf (Optional, ab 1.6.0 nicht mehr benötigt)
  - re2 (Optional)
- *dnsdist* (und andere Software) sollte **nicht** auf einem DNS Server in Produktion kompiliert werden
- Installations-Anleitungen befinden sich unter https://dnsdist.org/install.html

ANWENDUNGSSZENARIEN FÜR DNSDIST

FAIL-OVER

- *dnsdist* kann DNS Anfragen basierend auf der Verfügbarkeit von Backend-Servern im Pool verteilen
  - Hierzu wird die Policy `firstAvailable` benutzt
  - Bei dieser Policy haben die Server im Pool eine Reihenfolge: der erste verfügbare Server in der Reihenfolge bekommt alle Anfragen
  - Diese Policy kann zusätzlich mit einem "Anfragen pro Sekunde" Limit konfiguriert werden
    - Wird dieses Limit überschritten, so werden die überzähligen Anfragen an den nächsten DNS Server in der Reihe geleitet

FAIL-OVER


Network
Internet
DNS
Resolver 1
(active)
Resolver 2
(inactive)
dnsdist
DNS
Client
DNS
Client

LOAD-BALANCING

- *dnsdist* kann DNS Anfragen auf verschiedene Back-End-Server (oder Back-End-Server Pools) verteilen. Hierbei gibt es verschiedene Load-Balancing Konfigurationen:
  - `leastOutstanding`: benutze den/die Server mit den wenigsten noch ausstehenden Anfragen
  - `chashed`: verteile die DNS Anfragen basierend auf dem Hash des Domain-Namens der Anfrage
  - `whashed`: verteile die DNS Anfragen basierend auf dem Hash des Domain-Namens der Anfrage, aber mit einer Gewichtung der Back-End-Server
  - `wrandom`: zufällige Verteilung der Anfragen, jedoch mit einer Gewichtung der Back-End-Server (die Back-End-Server bekommen einen Anteil der Anfragen basierend auf der Gewichtung)
  - `roundrobin`: Anfragen werden auf Basis eines Round-Robin Algorithmus verteilt

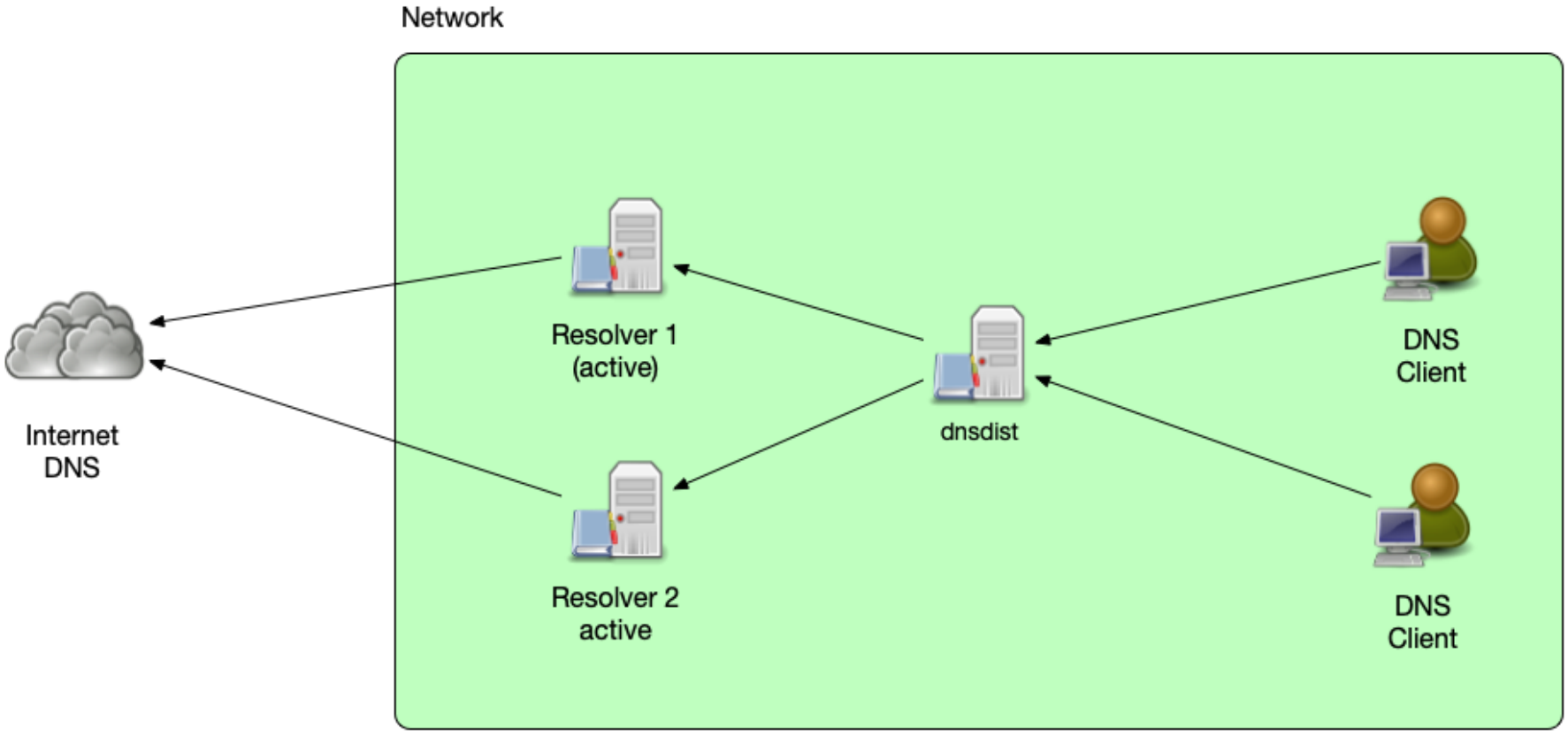
LOAD-BALANCING

- Die Load-Balancing Regeln können durch eigene, in der Programmiersprache Lua (https://www.lua.org/) programmierten, Regeln erweitert werden
    - Beispiel einer einfachen Round-Robin Policy:

```
counter=0
function luaroundrobin(servers, dq)
     counter=counter+1
     return servers[1+(counter % #servers)]
end

setServerPolicyLua("luaroundrobin", luaroundrobin)
```

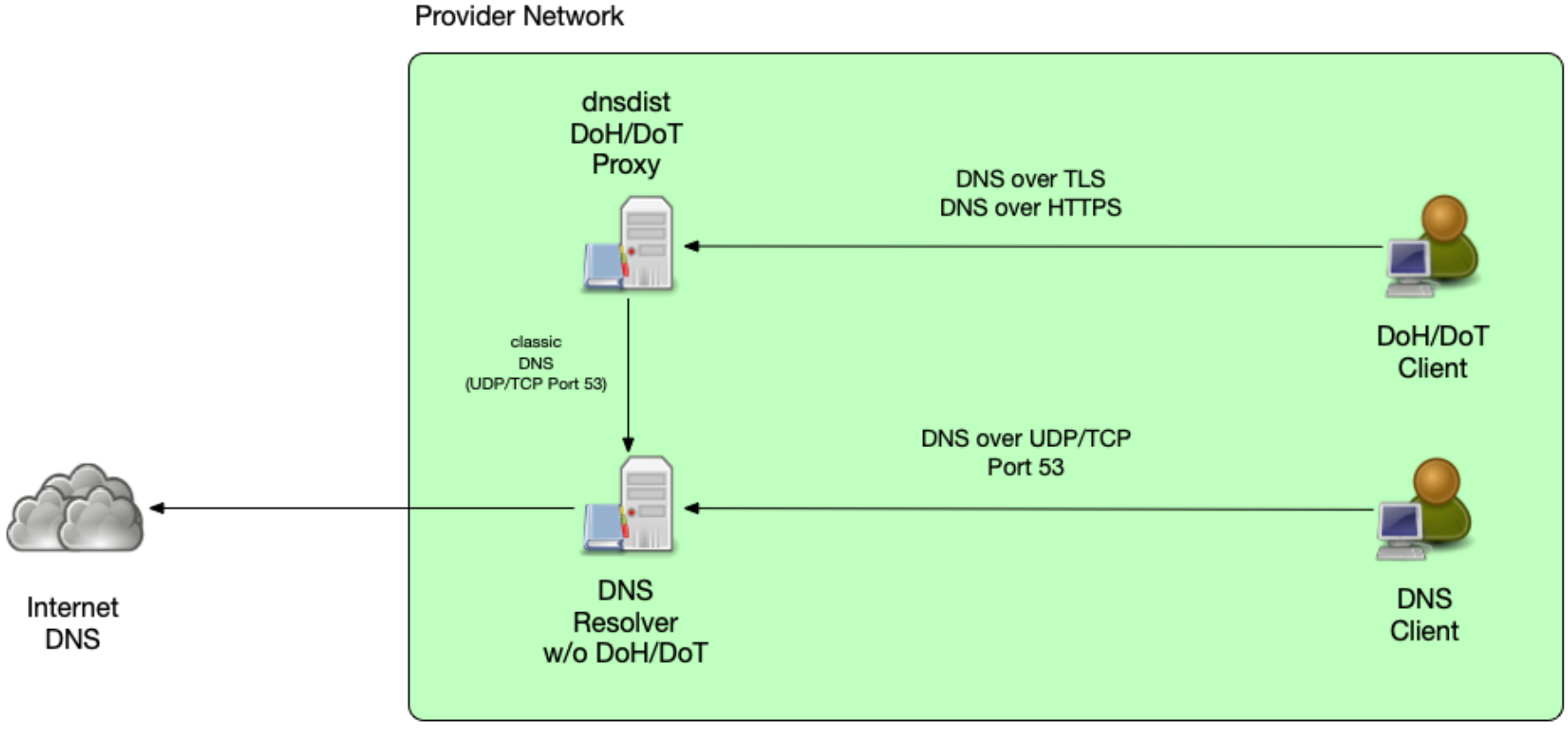
LOAD-BALANCING



DOH/DOT PROXY, DDOS UND MALWARE ABSICHERUNG

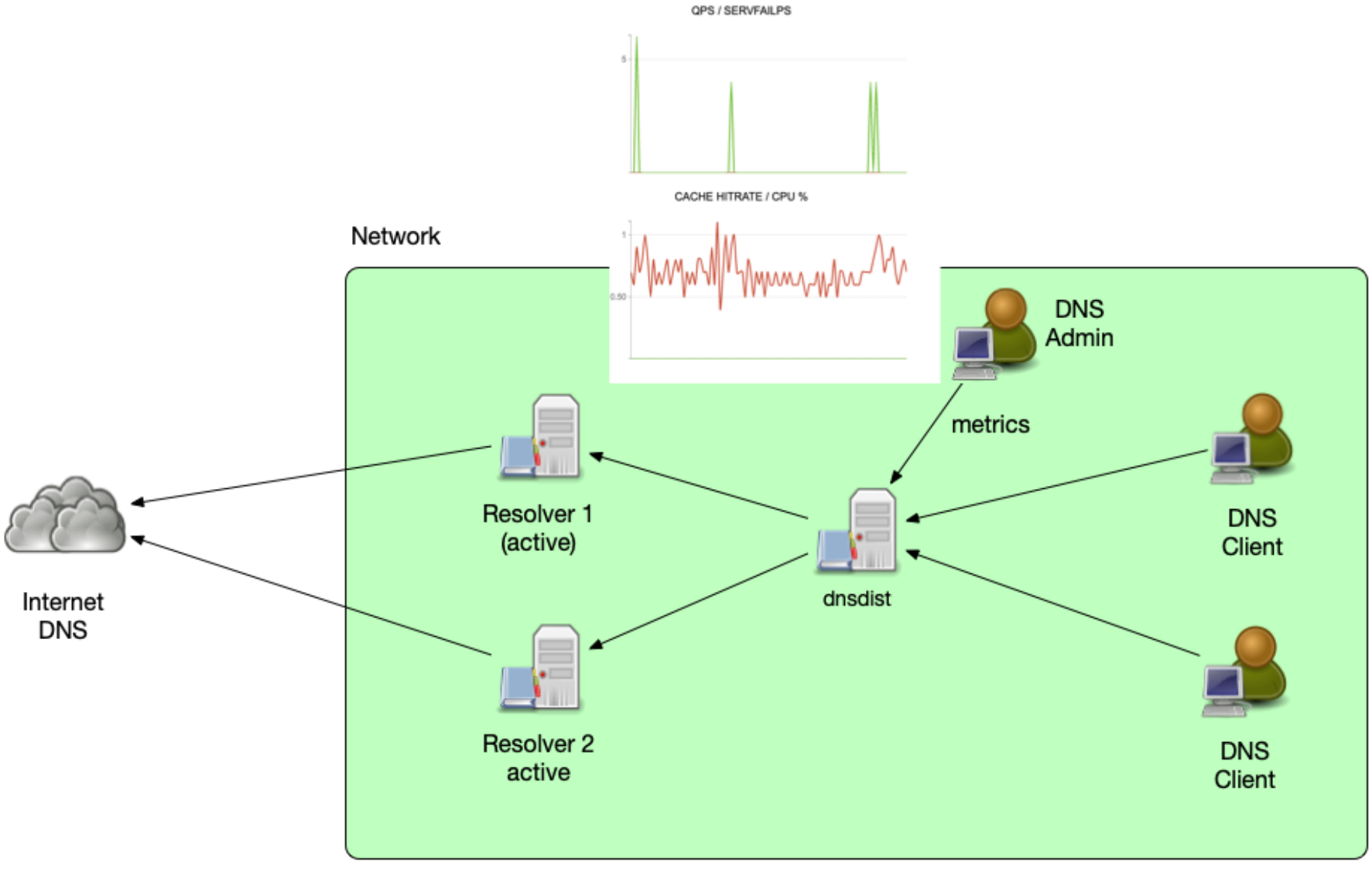
- *dnsdist* kann neue Funktionen zu bestehenden autoritativen DNS Server und DNS Resolvern hinzufügen, ohne das Änderungen an diesen Backend-Servern notwendig sind
  - Schutz vor Denial-of-Service Angriffen
  - Filtern von bekannten Malware-Domains (aka *DNS-Firewall*)
  - DNS Transport-Verschlüsselung (DNS-over-TLS und DNS-over-HTTPS)

DOH/DOT TRANSPORT VERSCHLÜSSELUNG



MESSDATEN IN EINEM DNS-SERVER-CLUSTER ZUSAMMENFÜHREN

- Da *dnsdist* als zentrales System die Anfragen an die Server eines DNS-Clusters weiterleitet, kann *dnsdist* ein Monitoring für einen DNS-Cluster bereitstellen
  - für mehrere DNS Resolver
  - für mehrere autoritative DNS Server

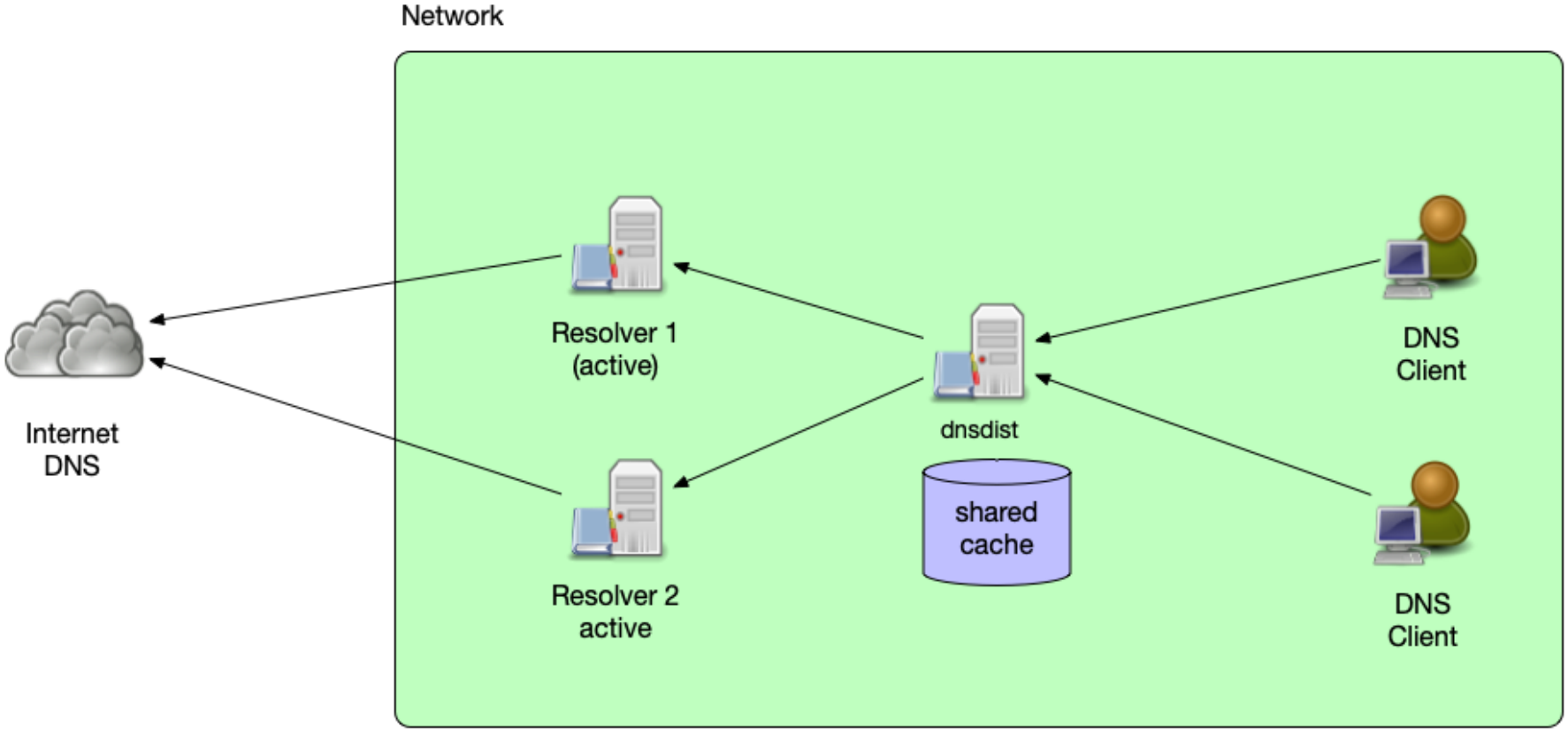

QPS / SERVFAILPS
CACHE HITRATE / CPU %
Network
DNS
Admin
metrics
Resolver 1
(active)
DNS
Client
Internet
DNS
dnsdist
Resolver 2
active
DNS
Client

ZENTRALER DNS CACHE

- *dnsdist* ist kein DNS-Resolver, es kann keine DNS-Delegationen verfolgen und DNS Namen selbstständig auflösen
  - *dnsdist* kann jedoch die Antworten von DNS-Resolvern im Cache speichern und von dort beantworten
  - *dnsdist* kann optional abgelaufene DNS-Informationen senden (TTL expired), wenn keiner der konfigurierten Back-End-Server erreichbar ist

```
> getPool(""):getCache():printStats()
Entries: 122/10000
Hits: 9147
Misses: 10147
Deferred inserts: 1
Deferred lookups: 0
Lookup Collisions: 0
Insert Collisions: 0
TTL Too Shorts: 0
```

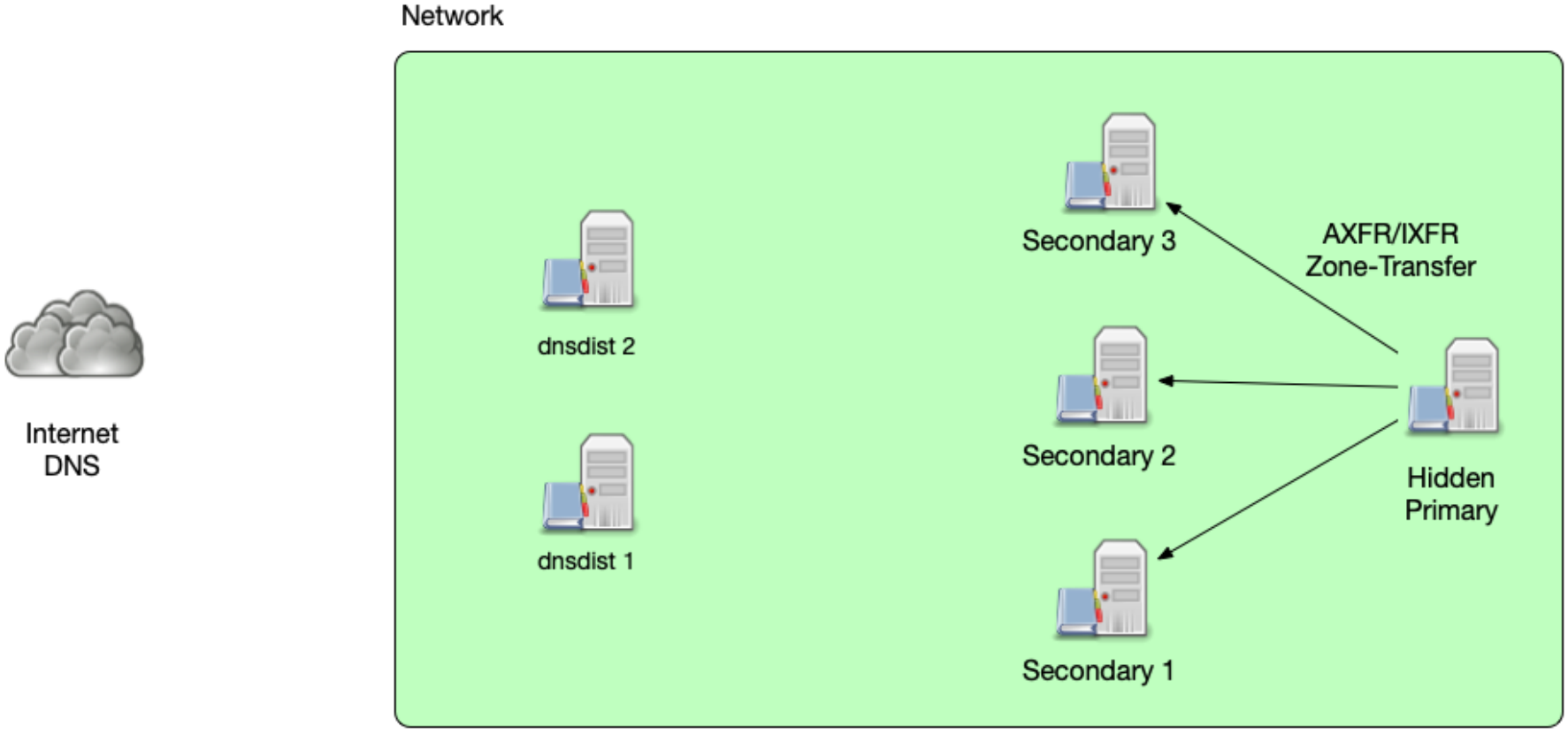
ZENTRALER DNS CACHE



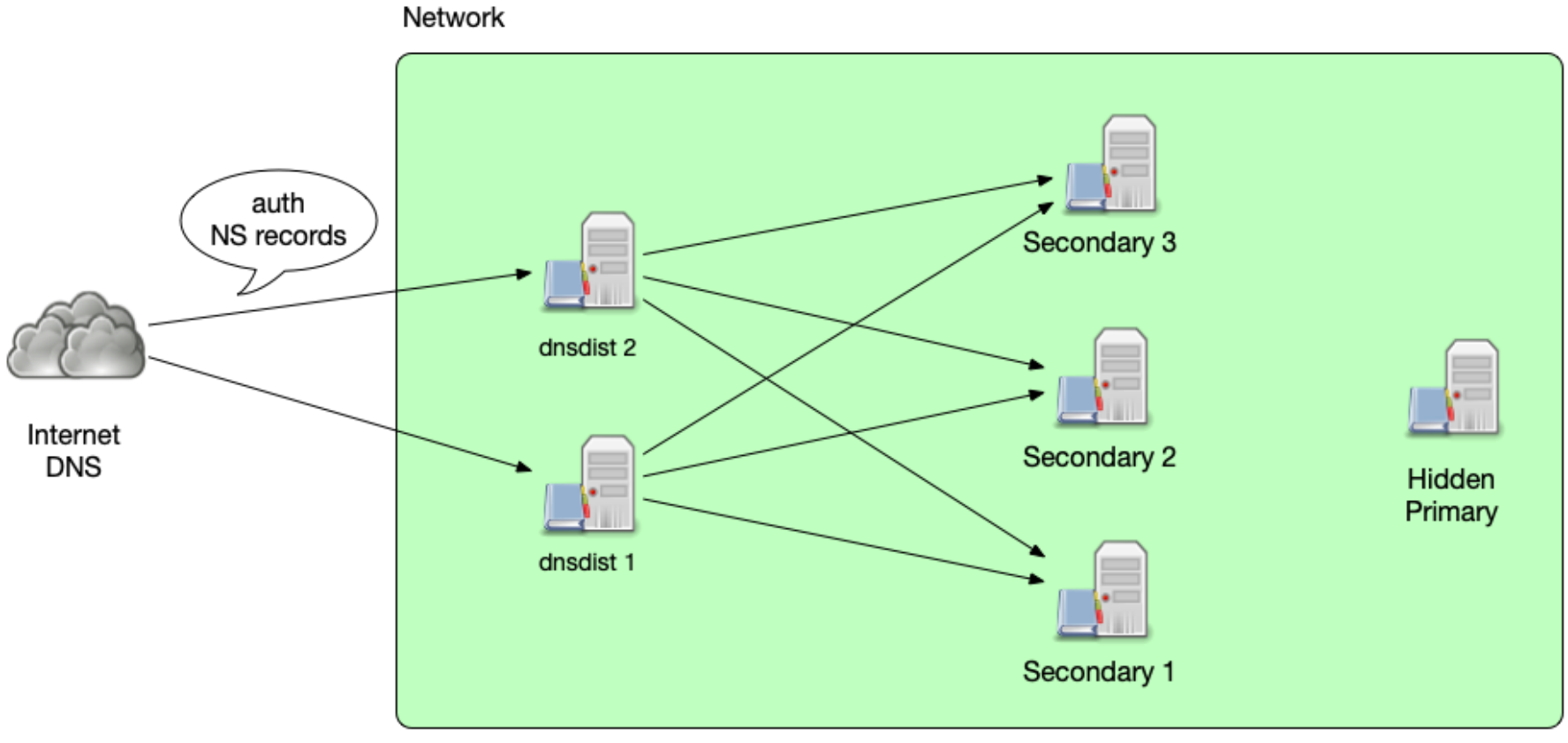
SCHUTZ VON AUTORITATIVEN DNS SERVERN

- *dnsdist* kann als front-end Load-Balancer vor autoritativen DNS Servern eingesetzt werden
  - Mittels spezieller Regeln kann *dnsdist* die autoritativen DNS Server vor bestimmen bösartigen DNS Anfragen schützen
  - Back-End Server können aus dem Cluster genommen werden (für Wartung oder zur Analyse von Vorfällen), ohne das der DNS-Dienst beeinträchtigt wird

SCHUTZ VON AUTORITATIVEN DNS SERVERN



SCHUTZ VON AUTORITATIVEN DNS SERVERN



KONFIGURATION UND INSTALLATION

HOCHVERFÜGBARKEIT MIT DNSDIST

- Beim Einsatz von *dnsdist* ist darauf zu achten das *dnsdist* kein (neuer) "Single-Point-of-Failure" wird
- Mögliche Lösungen, um *dnsdist* ausfallsicher zu betreiben:
  - *dnsdist* vor DNS Resolvern: mehrere *dnsdist* Instanzen via DHCP oder IPv6 RA an die Client-Rechner verteilen
  - *dnsdist* vor autoritativen DNS-Servern: per DNS-Delegation auf mehrere *dnsdist* Instanzen verweisen
  - Die *dnsdist* Instanzen durch eine Hochverfügbarkeits-Lösung eines Betriebssystems ausfallsicher gestalten (z.B. Heartbeat/Pacemaker)

DNSDIST KONFIGURATION

- *dnsdist* liest die initiale Konfiguration aus der Datei `dnsdist.conf` (meist im Verzeichnis `/etc/dnsdist`)
  - diese Konfigurationsdatei ist Lua-Quellcode welcher in einer eingebetteten Lua-VM ausgeführt wird

KONFIGURATIONS-DATEI

- Beispiel `dnsdist.conf` Konfigurationsdatei:

```
---- Listen addresses
addLocal('192.0.2.1:53',     { reusePort=true })
addLocal('127.0.0.1:53',     { reusePort=true })
addLocal('[::1]:53',         { reusePort=true })
addLocal('[2001:db8::1]:53', { reusePort=true })
---- Back-end server
newServer({address="192.0.2.100",        qps=10000, order=1})
newServer({address="2001:db8:100::5353", qps=100,   order=3})
newServer({address="2001:db8:200::6312", qps=100,   order=2})
---- Policy
setServerPolicy(whashed)
setACL({'192.0.2.0/24', '2001:db8::/64'})
---- Cache
pc = newPacketCache(10000, {maxTTL=86400, minTTL=0, temporaryFailureTTL=60, staleTTL=60, dontAge=false})
getPool(""):setCache(pc)
---- Web-server
webserver("192.0.2.1:8083")
setWebserverConfig({acl="192.0.2.10/32",password="dnsdist-is-great"})
--- Console
controlSocket('127.0.0.1:5199')
setKey("2ux3QDmpdDAzYjspexaspAdqnXF8jXFU5qhd/BqXV8ag=")
---- Filter Rules
addAction(RegexRule(".*\\.facebook\\..*$"), RCodeAction(DNSRCode.REFUSED))
addAction(RegexRule(".*\\.doubleclick\\..*$"), RCodeAction(DNSRCode.REFUSED))
```

DNSDIST KONSOLE

- Mittels des *dnsdist* Programms kann eine interaktive Kommandozeilen-Konsole zum laufenden *dnsdist* Prozess aufgebaut werden
  - von dieser Konsole aus kann der laufende *dnsdist* Prozess dynamisch konfiguriert werden, ohne das der *dnsdist* Prozess neu gestartet werden muss

```
$ /bin/dnsdist -c
> showServers()
198.51.100.12#   Name                 Address                      State     Qps    Qlim Ord Wt    Queries   Drops
0   192.0.2.53:53        192.0.2.53:53                   up      1.0   10000   1  1      10088     132   0.0   9.2
1   198.51.100.12:53     198.51.100.12:53                up      0.0     100   2  1       1391       2   0.0  44.5
2   203.0.113.11:53      203.0.113.11:53                 up      0.0     100   3  1        318       0   0.0  65.3
All                                                              0.0                    11797     134
> newServer({address="1.1.1.1",        qps=10000, order=1})
1.1.1.1:53
> showServers()
#   Name                 Address                      State     Qps    Qlim Ord Wt    Queries   Drops Drate   Lat
0   192.0.2.53:53        192.0.2.53:53                   up      0.0   10000   1  1      10103     132   0.0   8.9
1   1.1.1.1:53           1.1.1.1:53                      up      0.0   10000   1  1          3       0   0.0   0.3
2   198.51.100.12:53     198.51.100.12:53                up      0.0     100   2  1       1392       2   0.0  44.4
3   203.0.113.11:53      203.0.113.11:53                 up      0.0     100   3  1        319       0   0.0  65.2
All                                                              0.0                    11817     134
>
```

DNSDIST WEB-SERVER

- *dnsdist* kann interne Statistiken über einen eingebauten Web-Server anzeigen
  - dieser Web-Server muss in der *dnsdist* Konfiguration angeschaltet werden

```
---- Webserver
webserver("192.0.2.1:8083")
setWebserverConfig({acl="192.0.2.10/32",password="dnsdist-is-great"})
```

DNSDIST WEB-SERVER

POWERDNS

dnsdist 1.6.0-alpha3
dnsdist comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to redistribute it according to the terms of the GPL version 2.

Uptime: 4 hours, Number of queries: 22900 (1.00 qps), ACL drops: 0, Dynamic drops: 0, Rule drops: 0
Average response time: 6.79 ms, CPU Usage: 0.90%, Cache hitrate: 100.00%, Server selection policy: leastOutstanding
Listening on: 127.0.0.1:53, 172.22.1.8:53, [::1]:53, [fd75:8765:1d2a:0:a90a:6c20:75a4:d5dd]:53, ACL: 0.0.0.0/0, ::/0

QPS / SERVFAILPS

CACHE HITRATE / CPU %

#	Name	Address	Status	Latency	Queries	Drops	QPS	Out	Weight	Order	Pools
0	127.0.0.11:53	127.0.0.11:53	up	33.54	10186	140	0.00	0	1	1	
1	1.1.1.1:53	1.1.1.1:53	up	12.35	69	3	0.00	0	1	1	
2	172.22.1.1:53	172.22.1.1:53	up	49.72	1413	2	0.00	0	1	2	

#	Rule	Action	Matches
0	Regex: .*\.facebook\..*$	set rcode 5	0
1	Regex: .*\.doubleclick\..*$	set rcode 5	2

#	Response Rule	Action	Matches
No response rules defined			

Dyn blocked netmask	Seconds	Blocks	Reason
No dynamic blocks active			

Kernel-based dyn blocked netmask	Seconds	Blocks
No eBPF blocks active		

DNSDIST WEB-API

- Mittels des eingebauten Web-Servers bietet *dnsdist* eine Web-API. Über diese Web-API können …
  - … Metriken im JSON Format abgefragt werden
  - … Metriken im Prometheus Format abgefragt werden
  - … die laufenden Konfigurationen des *dnsdist* Server Prozesses gesichert werden
- Die Web-API ist über einen API-Schlüssel gesichert, welcher bei jeder Anfrage übermittelt werden muss

METRIKEN EINES DNS-SERVER CLUSTERS MIT DNSDIST ZUSAMMENFÜHREN

GRAPHITE MONITORING

- Graphite ist eine in Python geschriebene Monitoring Software (Open Source) https://graphiteapp.org/
- *dnsdist* kann Metriken mittels des Graphite eigenen *carbon* Protokolls an einen Graphite-Server senden
- Beispiel-Konfiguration:

```
carbonServer('192.0.2.210', 'dnsdist.isp.example', 30, 'dnsdist', 'main')
```

- siehe https://dnsdist.org/guides/carbon.html

DNSDIST UND PROMETHEUS

- Prometheus ist eine populäre Monitoring Lösung: https://prometheus.io
- Prometheus kann die *dnsdist* Metriken mittels der Web-API vom `/metrics` URL-Endpunkt auslesen

ZENTRALER DNS CACHE

- *dnsdist* kann ein oder mehrere Caches für DNS Antworten aufbauen
    - Unterschiedliche Server-Pools können auf separate Caches zugreifen
- Die Caches speichern die Antworten von den Back-End-Servern (DNS Resolver oder autoritative DNS Server)
- Beispiel:

```
pc = newPacketCache(10000, --- create a new pool cache "pc" with 10.000 entries
  {
  maxTTL=86400,            --- maximum TTL cache time
  minTTL=0,                --- minimum TTL cache time
  temporaryFailureTTL=60,  --- TTL used for server failures or "refused"
  staleTTL=60,             --- TTL for stale cache entries
  dontAge=false            --- cache entries "age", their TTL is decremented in cache
})
getPool(""):setCache(pc)   --- assign the cache to the default pool
```

LASTVERTEILUNG FÜR AUTORITATIVE DNS SERVER

KONFIGURATION DES HEALTH-CHECK

- *dnsdist* prüft die Verfügbarkeit eines Back-End-Server mit einer DNS-Anfrage des A-Records für `a.root-servers.net`
  - diese Anfrage funktioniert bei DNS-Resolvern, jedoch nicht bei einigen Konfiguration für autoritative Server
  - der *dnsdist* 'Health-Check' kann in der Konfiguration angepasst werden. In diesem Beispiel werden die autoritativen Server für `dnsworkshop.de` geprüft:

```
newServer({address="5.45.109.212",                          checkType="SOA", checkType=DNSClass.IN, checkName="dnsw
newServer({address="185.92.221.212",                        checkType="SOA", checkType=DNSClass.IN, checkName="dnsw
newServer({address="2001:19f0:5001:df:76d7:5703:ba0a:e220", checkType="SOA", checkType=DNSClass.IN, checkName="dnsw
newServer({address="2a03:4000:6:2115::2",                   checkType="SOA", checkType=DNSClass.IN, checkName="dnsw
setServerPolicy(leastOutstanding)
setLocal("192.0.2.123:53")
[...]
```

SOA ANFRAGEN UND IXFR/AXFR

- Abfragen des `SOA` Records und `IXFR` / `AXFR` Anfragen für Zonentransfers sollten per *dnsdist* Regel auf einen dedizierten "primary" autoritativen Back-End-Server geleitet werden
  - dies stellt sicher das Zonen-Transfers aus der gleichen Zone beantwortet werden wie die SOA-Anfrage
  - die nachfolgende Beispiel-Konfiguration sendet alle SOA/AXFR und IXFR Anfragen zum Pool `primary`, welcher nur einen einzigen autoritativen Server enthält

```
newServer({
  address="192.0.2.123",
  name="primary",
  pool={"primary", "otherpool"}
})
addAction(
  OrRule({
    QTypeRule(DNSQType.SOA),
    QTypeRule(DNSQType.AXFR),
    QTypeRule(DNSQType.IXFR)}),
  PoolAction("primary")
)
```

SOA ANFRAGEN UND IXFR/AXFR

- Die Back-End autoritativen DNS Server sehen die Anfragen (SOA, Zonen-Transfers) von der IP-Adresse des *dnsdist* Servers
  - IP-Adressen basierte Access-Control-Listen müssen entsprechend angepasst werden (noch besser ist es TSIG basierte Authentisierung zu benutzen)
  - Der Parameter `source` definiert die von *dnsdist* für ausgehende DNS Anfragen benutzte IP-Adresse

```
newServer({address="192.0.2.1", source="192.0.2.127"})
newServer({address="192.0.2.1", source="eth1"})
newServer({address="192.0.2.1", source="192.0.2.127@eth1"})
```

DYNAMISCHE DNS UPDATES

- Dynamische Updates (RFC 2136) sollten direkt zu einem der autoritativen DNS-Server, und nicht zu der *dnsdist* Adresse, gesendet werden
  - Der Domain-Name eines echten autoritativen DNS Servers sollten im `mname` Feld des SOA-Records stehen
  - Alternativ kann der Sender der dynamischen Updates (z.B. `nsupdate`) konfiguriert werden, so das die Updates an die dedizierte IP-Adresse des primary autoritativen DNS-Servers gesendet werden:

```
nsupdate
> ttl 3600
> server 192.0.2.221
> add www.example.com. IN A 192.0.2.212
> send
```

NOTIFY

- Ein Update einer DNS Zone auf einem autoritativen primary DNS Server erzeugt eine NOTIFY Nachricht an alle secondary DNS Server, welche als NS-Records in der DNS Zone definiert sind
    - der Empfänger kann eine *dnsdist* Instanz sein, welche vor dem autoritativen DNS Server positioniert ist und das NOTIFY an den Back-End-Server weiterleitet
    - IP basierte ACLs auf dem Back-End-Server müssen entsprechend angepasst werden, so das ein NOTIFY von der *dnsdist* Adresse verarbeitet wird
    - ACLs in *dnsdist* können benutzt werden, um NOTIFY Meldungen nur von vertrauenswürdigen IP-Adressen zu erlauben
    - ACLs mit TSIG Schlüsseln müssen nicht angepasst werden, denn diese sind unabhängig von den IP-Adressen der Server
    - Als Alternative kann NOTIFY in einigen DNS-Servern explizit konfiguriert werden, z.B. im BIND 9:

```
zone "example.com" {
   type primary;
   file "example.com";
   notify explicit;
   also-notify { 192.0.2.53; 198.51.100.12; };
};
```

RATE-LIMITING

- *dnsdist* kann DNS Anfragen auf Basis der Inhalte filtern oder per Rate-Limit beschränken:
  - DNSSEC angefragt oder nicht
  - EDNS Optionen
  - Anfragen pro Sekunde pro Quell-Subnetz
  - Quell-Netzwerk
  - DNS Opcodes
  - DNS Netzwerk Klassen
  - Domain-Name in der Anfrage (per Regular Expression)
  - Anzahl der Label im Domain-Name der Anfrage
  - Return Code
  - RD-Flag (Recursion Desired)
  - Anzahl der Records / Anzahl der Record-Typen in einer DNS Antwort
  - und vieles mehr

RATE-LIMITING

- Diese Regeln können von *dnsdist* dynamisch auf Basis der gemessenen Anfragen erstellt werden (dynamische Regeln)
  - Durch die Lua-Programmiersprache können die Regeln sehr individuell vom Betreiber des *dnsdist* angepasst werden
- Regeln können automatisch auslaufen und entfernt werden um ein Überblockieren zu verhindern

```
local dbr = dynBlockRulesGroup()
dbr:setQueryRate(30, 10, "Exceeded query rate", 60)
dbr:setRCodeRate(DNSRCode.NXDOMAIN, 20, 10, "Exceeded NXD rate", 60)
dbr:setRCodeRate(DNSRCode.SERVFAIL, 20, 10, "Exceeded ServFail rate", 60)
dbr:setQTypeRate(DNSQType.ANY, 5, 10, "Exceeded ANY rate", 60)
dbr:setResponseByteRate(10000, 10, "Exceeded resp BW rate", 60)

function maintenance()
  dbr:apply()
end
```

- Dynamische Regeln: https://dnsdist.org/guides/dynblocks.html

RATE-LIMITING

- Mittels der eingebauten Regeln kann *dnsdist* viele Arten von bösartigen DNS-Anfragen blockieren oder umleiten
  - Unter Linux kann *dnsdist* mit Hilfe der eBPF Kernel-VM bestimmte DNS-Anfragen schon im Linux-Kernel blockieren, bevor diese Pakete durch den TCP/IP Stack gegangen sind
    - dies kann die Last auf dem System bei einem Denial-of-Service Angriff signifikant senken
- eBPF Socket Filter: https://dnsdist.org/advanced/ebpf.html

LOAD-BALANCING FÜR RESOLVER

FAIL-OVER KONFIGURATION

- Wird die *dnsdist* Load-Balancing Policy auf `firstAvailable` gesetzt, so ist eine einfache Fail-Over Konfiguration aktiv
  - alle DNS Anfragen werden zum ersten verfügbaren DNS Server gesendet, welcher den konfigurierten Schwellwert (Max. Anfragen pro Sekunde) noch nicht erreicht hat

```
---- Back-end server
newServer({address="192.0.2.100",        qps=1000, order=1})
newServer({address="2001:db8:100::5353", qps=500,  order=2})
newServer({address="2001:db8:200::6312", qps=500,  order=3})
---- Policy
setServerPolicy(firstAvailable)
setACL({'192.0.2.0/24', '2001:db8::/64'})
---- Cache
pc = newPacketCache(10000, {maxTTL=86400, minTTL=0, temporaryFailureTTL=60, staleTTL=60, dontAge=false})
getPool(""):setCache(pc)
[...]
```

LOAD-BALANCING KONFIGURATION

- Weitere Policy Einstellungen erzeugen aktive Load-Balancing oder Load-Distribution Konfiguration

```
---- Back-end server
newServer({address="192.0.2.100",         order=1})
newServer({address="2001:db8:100::5353", order=3})
newServer({address="2001:db8:200::6312", order=2})
---- Policy
setServerPolicy(leastOutstanding)
setACL({'192.0.2.0/24', '2001:db8::/64'})
---- Cache
pc = newPacketCache(10000, {maxTTL=86400, minTTL=0, temporaryFailureTTL=60, staleTTL=60, dontAge=false})
getPool(""):setCache(pc)
[...]
```

- Eigene Load-Balancing Policies können in der Lua Programmiersprache hinzugefügt werden
- Dokumentation: "Loadbalancing and Server Policies" https://dnsdist.org/guides/serverselection.html

SERVER POOLS

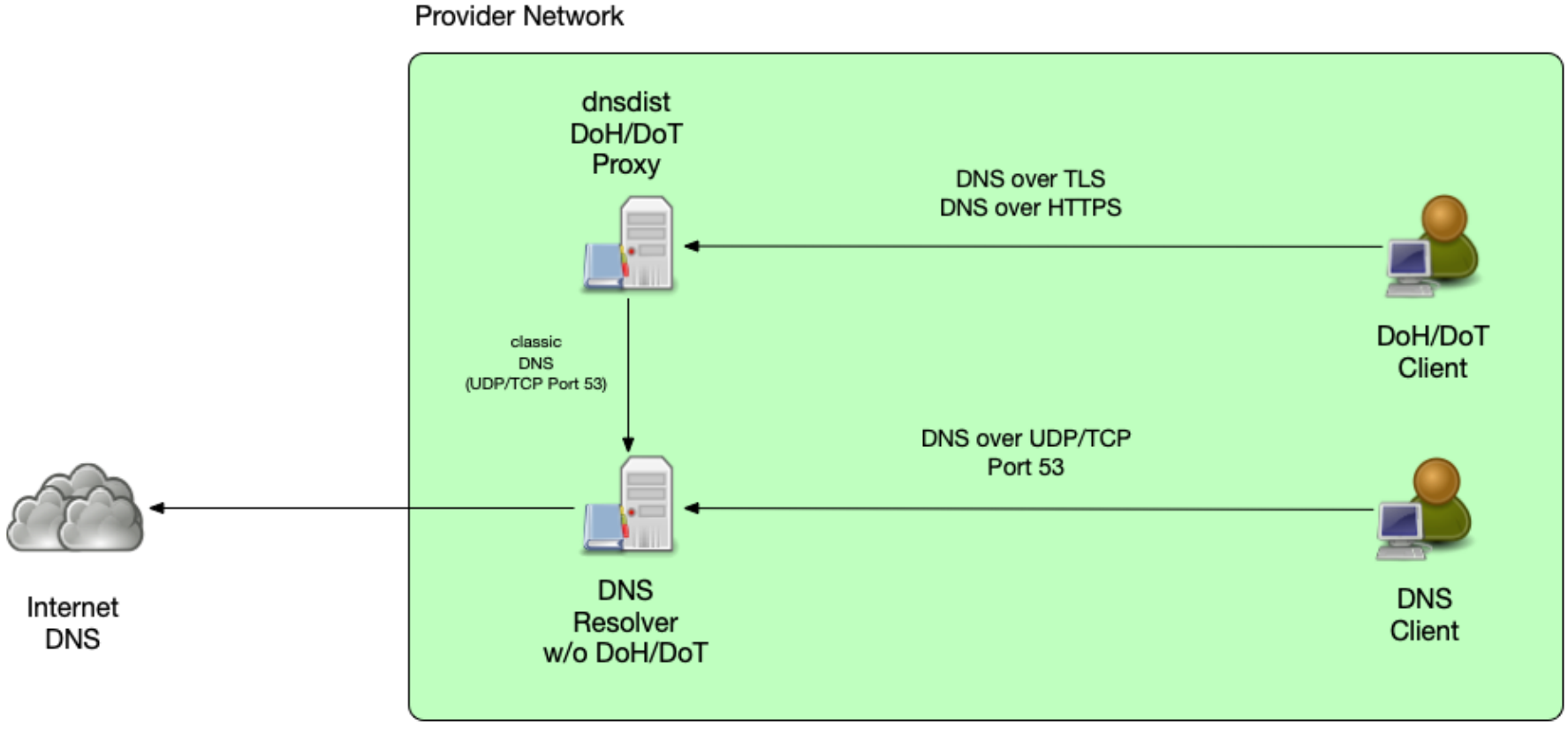
- *dnsdist* gruppiert Back-End-Server in einem oder mehreren "Pools"
  - Es gibt immer den Pool mit dem leeren Namen ""
  - Neue Pools können beim Hinzufügen von neuen Back-End-Servern erzeugt werden
  - Regeln und Aktionen (Rules and Action) können benutzt werden im Anfragen bestimmten Pools zuzuweisen
- Pools können benutzt werden um *bösartige* Anfragen (DDoS, Malware) zu isolieren

```
-- Add a backend server with address 192.0.2.3 and assign it to the "abuse" pool
newServer({address="192.0.2.3", pool="abuse"})
-- Send all queries for "bad-domain1.example." and "bad-domain2.example" to the "abuse" pool
addAction({'bad-domain1.example', 'bad-domain2.example.'}, PoolAction("abuse"))
```

DOH/DOT TERMINATION

- *dnsdist* kann DNS-over-TLS (DoT) und DNS-over-HTTPS (DoH) Verbindungen terminieren
  - *dnsdist* erstellt einen DoH/DoT "proxy", welcher DoH/DoT Anfragen von DNS Clients annimmt und via klassischem DNS (UDP/TCP Port 53) an einen Back-End-Resolver weiterleitet

DOH/DOT PROXY



MOTIVATION FÜR EINEN DOH/DOT PROXY?

- Einfache Installation
- Die existierenden DNS-Resolver müssen nicht geändert werden
- Über separate Hardware/Server lässt sich diese Lösung einfach skalieren

FRAGEN UND ANTWORTEN