Równoważenie obciążenia

Kurs DevOps – Wykład 5

Kuba Nowak

Uniwersytet Wrocławski

3 grudnia 2025

Model OSI

Model OSI
7. Warstwa aplikacji
NNTP · SIP · SSI · DNS · FTP · Gopher · HTTP · NFS · NTP · SMPP · SMTP · SNMP · Telnet · DHCP · Netconf · RTP · (więcej)
6. Warstwa prezentacji
MIME · XDR · TLS · SSL
5. Warstwa sesji
Potoki (Unix/Linux) · NetBIOS · SAP · L2TP · PPTP · SPDY
4. Warstwa transportowa
TCP · UDP · SCTP · DCCP · SPX
3. Warstwa sieciowa
IP (IPv4, IPv6) · ICMP · IPsec · IGMP · IPX · AppleTalk · Router
2. Warstwa łącza danych
ATM · SDLC · HDLC · ARP · CSLIP · SLIP · GFP · PLIP · IEEE 802.3 · Frame Relay · ITU-T G.hn · PPP · X.25 · Przełącznik sieciowy
1. Warstwa fizyczna
RS-232 · RS-449 · ITU-T V-Series · I.430 · I.431 · POTS · PDH · SONET · PON · OTN · DSL · IEEE 802.3 · IEEE 802.11 · IEEE 802.15 · IEEE 802.16 · IEEE 1394 · USB · Bluetooth · Koncentrator sieciowy

Źródło: `https://pl.wikipedia.org/wiki/Model_OSI`

Proxy

Definicja

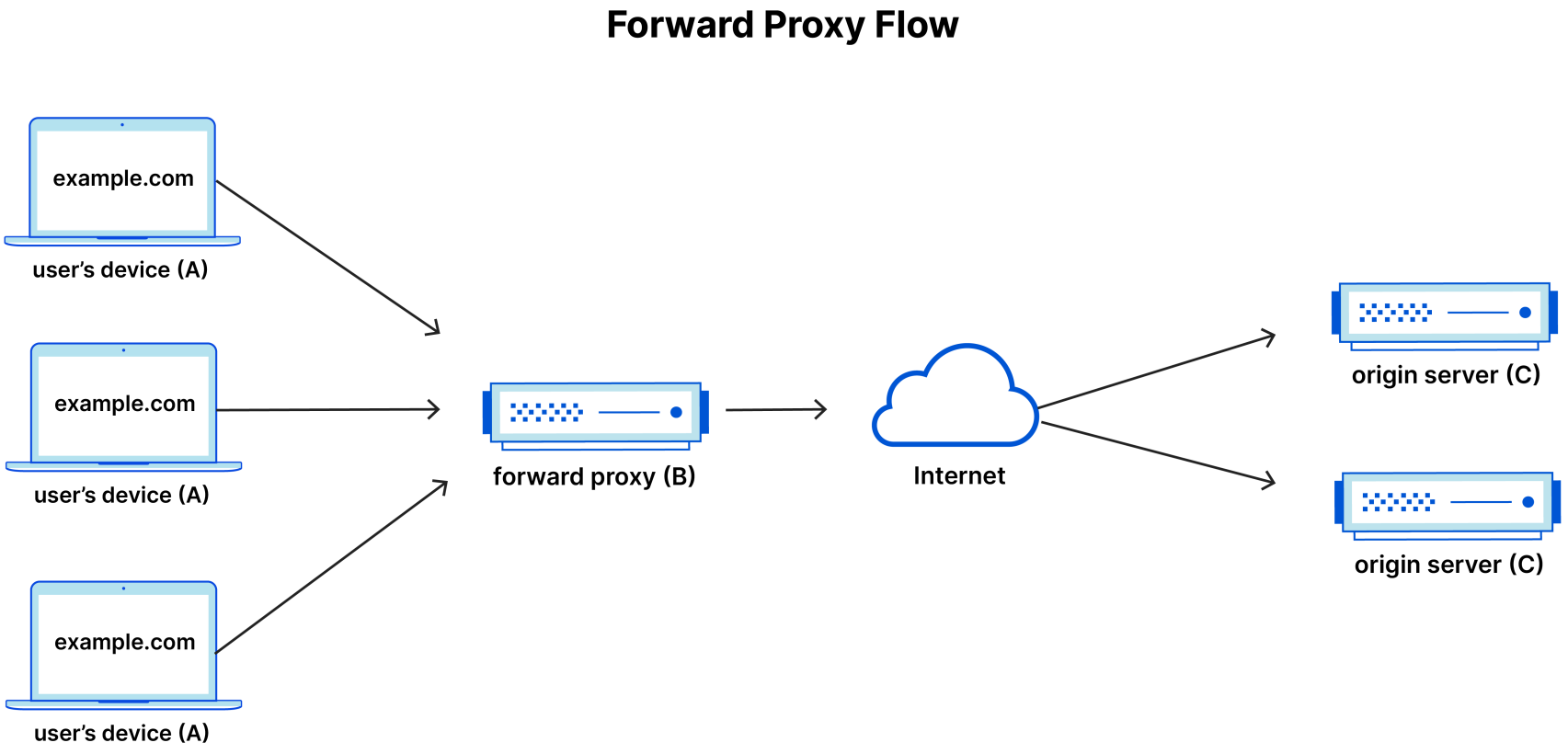
Serwer pośredniczący (ang. proxy) – oprogramowanie lub serwer z odpowiednim oprogramowaniem, które dokonuje pewnych operacji (zwykle nawiązuje połączenia) w imieniu użytkownika. —

`https://pl.wikipedia.org/wiki/Serwer_po%C5%9Brednicz%C4%85cy`

Typy proxy:

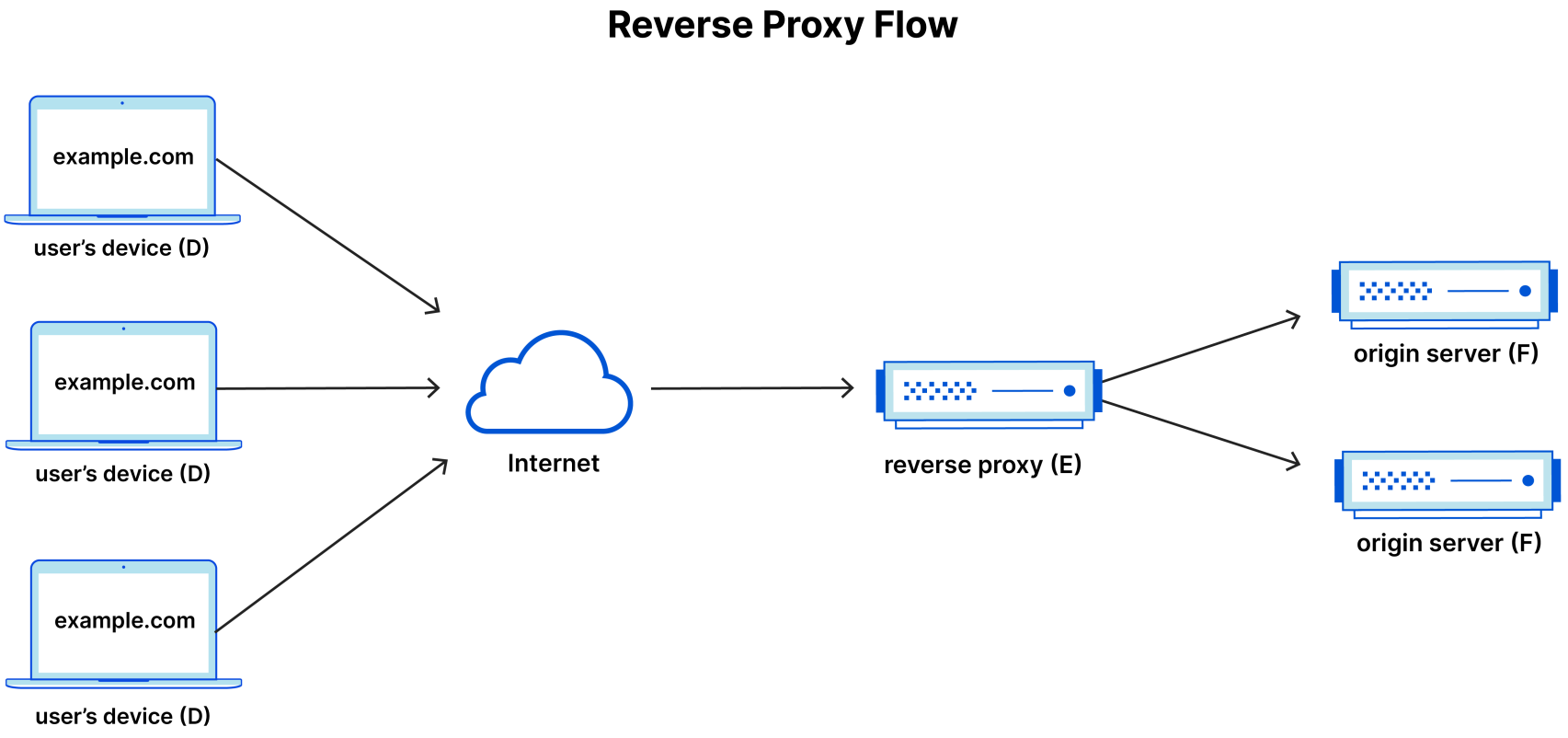
- forward proxy,
- reverse proxy.

Forward Proxy



Źródło: `https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/`

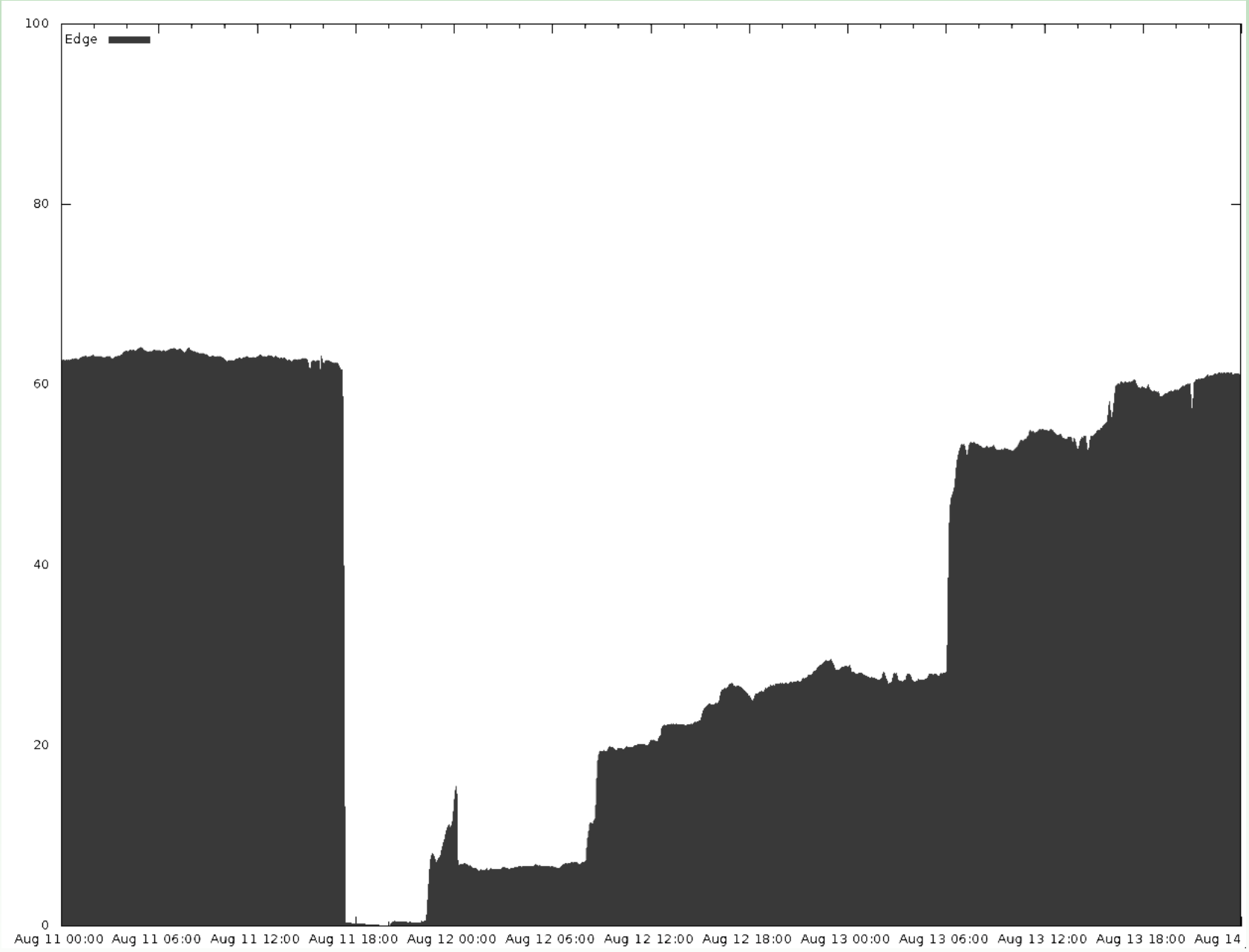
Reverse Proxy



Źródło: `https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/`

Rola proxy

- Kierowanie ruchem
- Standaryzowanie (ang. sanitize) wiadomości
- Aplikowanie reguł bezpieczeństwa
- Kontrola dostępu
- Anonimizacja
- Cachowanie
- Równoważenie obciążenia



Źródło: Beyer i in. 2018

Warstwy równoważenia obciążenia

Warstwa 7 HAProxy, NGINX

Warstwa 4 HAProxy, Linux Virtual Server (LVS or IPVS)

Warstwa 3 ECMP

HAProxy

- Jeden z najszybszych serwerów równoważących obciążenie.
- Open source.
- Pierwsza wersja z 2001.
- Oryginalny autor: Willy Tarreau.
- Ma wersję komercyjną (np. z GUI).
- Wielowątkowy, bazujący na zdarzeniach.

HAProxy

Wybrane strony używające HAProxy:

- Google
- Reddit
- GitHub
- Instagram
- kernel.org
- StackOverflow
- Twitter
- Wikipedia

HAProxy

Funkcjonalności:

- Wspiera HTTP (warstwa 7) i TCP (warstwa 4)
- Wsparcie dla UDP w wersji płatnej
- Przede wszystkim load balancer
- Pozwala narzucać ograniczenia dostępu/przesyłu
- Modyfikacja nagłówków wiadomości
- Odczyt i modyfikacja ciasteczek
- Ograniczenie przeciw DDoS
- Cache małych obiektów
- Monitorowanie stanu serwerów
- Kontrola poprzez socket unixowy

HAProxy

Wspierane protokoły:

- TCP
- HTTP
- FastCGI
- gRPC
- Passive FTP
- Syslog
- WebSocket

Konfiguracja

Cztery podstawowe sekcje:

- global
- frontend
- backend
- defaults

Access Control List (ACL)

Reguły, które można definiować, a następnie ewaluować podczas analizy pakietu, by określić zachowania warunkowe.

Zmienne

Zakresy widoczności zmiennych:

proc	globalna
sess	prywatna dla sesji TCP
txn	prywatna dla wymiany (transakcji) req-resp HTTP
req	dostępne tylko podczas przetwarzania requestu HTTP
res	dostępne tylko podczas przetwarzania odpowiedzi HTTP
check	prywatna dla danego wykonania sprawdzenia żywotności serwera

```
http-request set-var(txn.path) path
```

Demonstracja

Konfiguracja HAProxy

Algorytmy

roundrobin	Po kolei każdy serwer dostaje pakiety zgodnie z przydzieloną wagą.
static-rr	Round robin, ale ze statycznymi wagami.
leastconn	Połączenie jest przekazywane serwerowi obsługującemu najmniejszą liczbę klientów.
first	Wybiera serwer, mający wolne połączenia, o najmniejszym ID.
hash	Ekstrahuje podane dane z zapytania i wylicza na tej podstawie hasz.
source	Hasz jest wyliczany na podstawie adresu IP źródła. Zapewnia ciągłe forwardowanie do tego samego serwera.
uri	Hasz jest wyliczany na podstawie URI.
url_param	Wyszukuje parametr w zapytaniu HTTP i haszuje jego wartość.
hdr	Haszuje wartość danego nagłówka.
random	Losuje klucz do zahaszowania.
rdp-cookie	Haszuje identyfikator sesji RDP[1] wyciągany z pakietów TCP.
log-hash	Wyciąga wartość z loga, haszuje ją i następnie wysyła log do odpowiedniego serwera logów.
sticky	Wysyła logi ciągle do tego samego serwera, jeśli zmieni stan na DOWN, a potem na UP, dodawany jest na koniec listy.

[1] Remote Desktop Protocol

Obsługa stanu

Problem

Co jak połączenia klienta powodują wygenerowanie stanu po stronie serwera?

Potencjalne rozwiązania:

- Nie mieć stanu po stronie serwera – przekazać go do zewnętrznej bazy danych.
- Stan przechowywać po stronie klienta (ciasteczka).
- Przy pomocy ciasteczek identyfikować gdzie ostatnio wysłaliśmy tego klienta.
- Routowanie na podstawie pochodzenia zapytania (np. adres IP).

Bezpieczeństwo

Po uruchomieniu:

- Chroot
- Dropowanie roota

Wpływ na działanie HAProxy:

- Brak zapisu logów na dysk
- Po odczycie danych/konfiguracji z dysku na startupie, nie ma możliwości ponownego dostępu do nich w runtimie
- HAProxy nie jest informowane o tym, że jakiś socket unixowy został ponownie otwarty (np. do pisania logów)
- W runtimie brak dostępu do `resolv.conf`

Wydajność

- Automatyczne wykrywanie topologi procesora (rdzenie, cziplety, SoC-i, współdzielone L3, NUMA, podłączenia NIC)
- Minimalne zależności między wątkami
- Kernel Linuksa zajmuje większość czasu obsługi zapytania (70%-95%).
- Kernel Linuksa średnio się skaluje (np. liczba wątków korzystających z soketa)
- Własna implementacja zegara systemowego.
- Zarządzanie pamięcią przy pomocy pooli

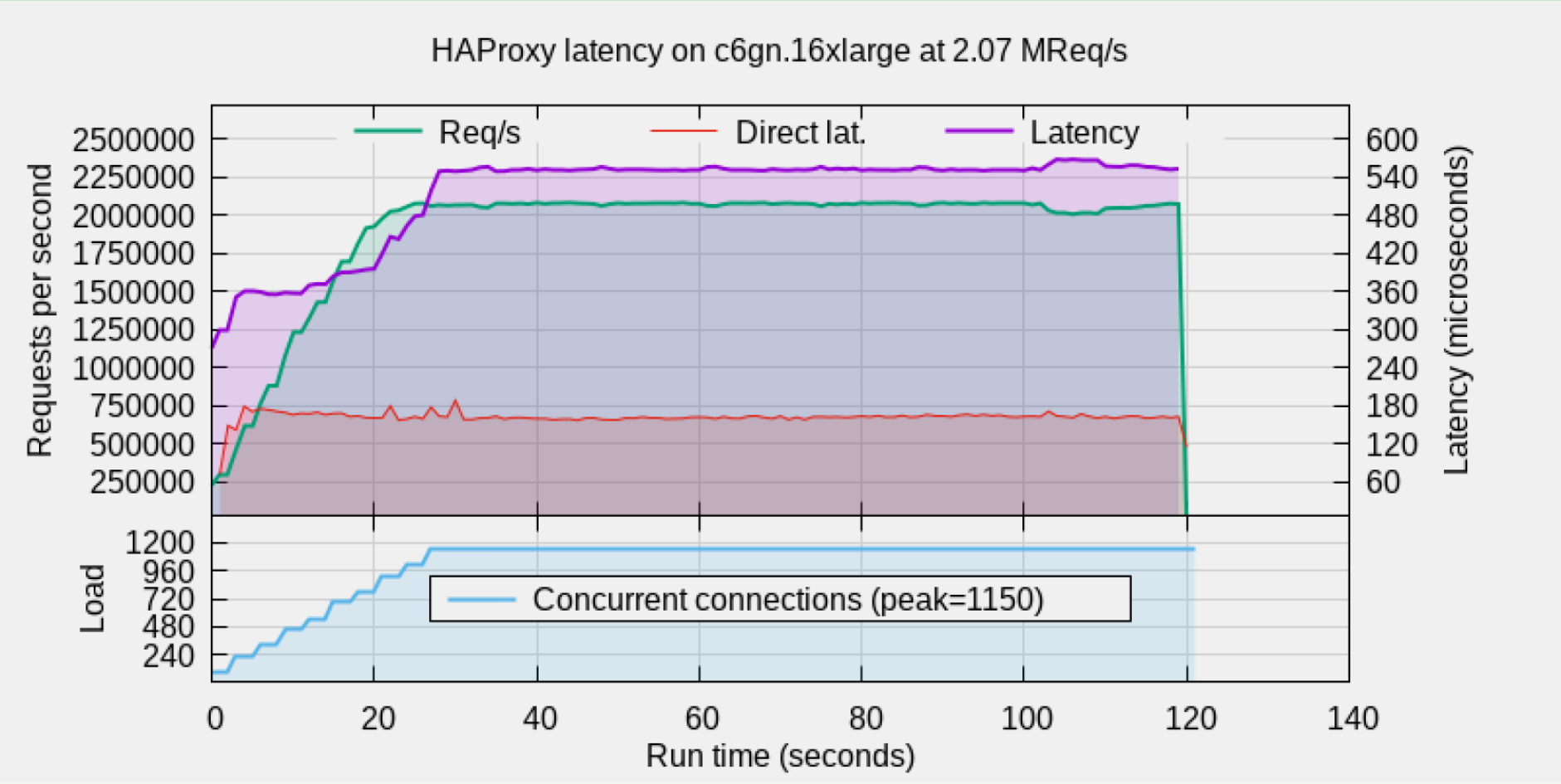
Skalowanie

- Głównie skalowanie horyzontalne.
- Możliwe jest współdzielenie stanu:
  - Model active-standby
  - Model active-active

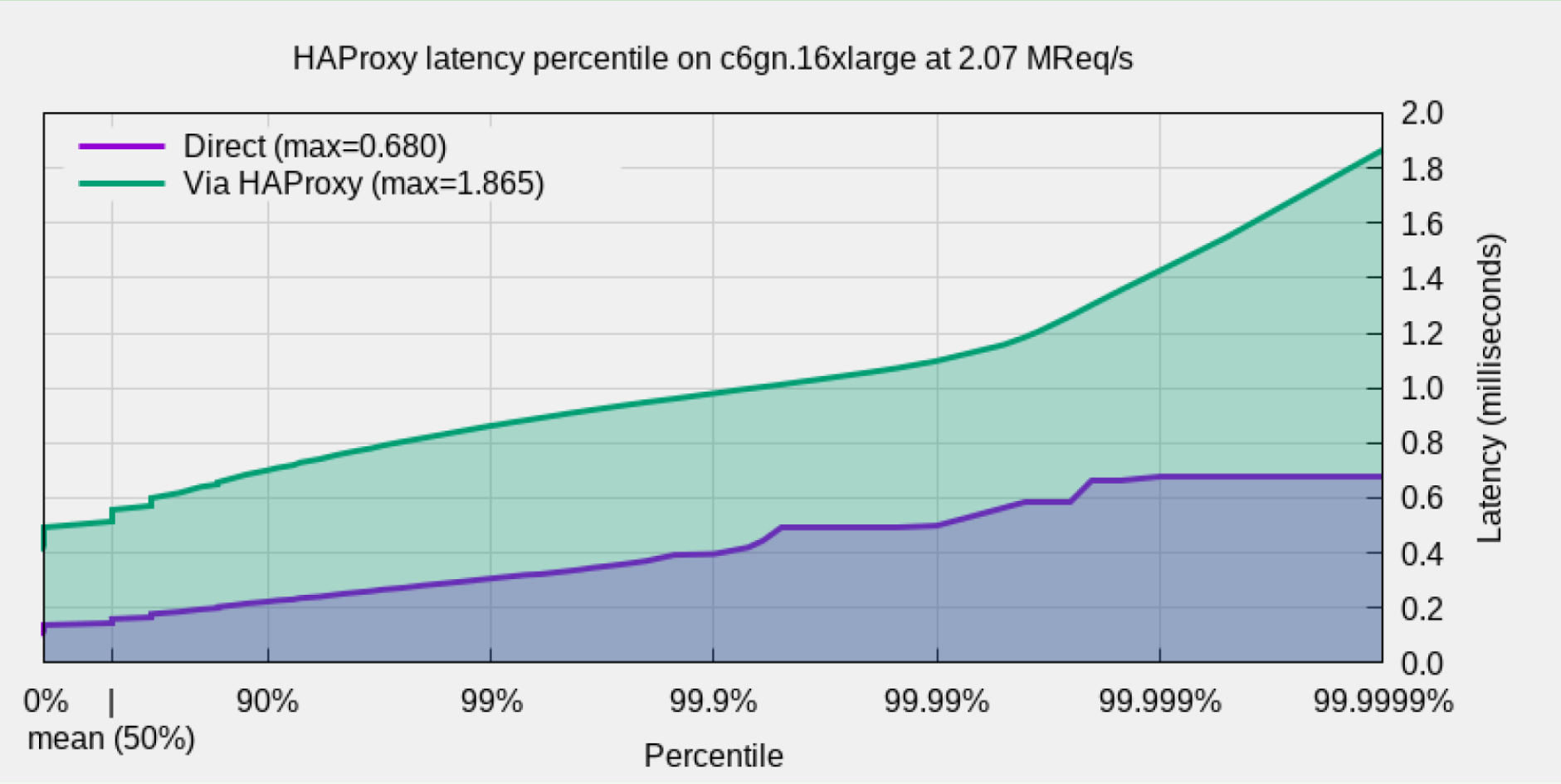
Wydajność proxy/load balancera

Jak jest mierzona wydajność proxy/load balancerów?

- Liczba obsługiwanych sesji na sekundę.
- Liczba obsługiwanych sesji równolegle.
- Przepustowość linka.
- Opóźnienie.



Źródło: `https://www.haproxy.com/blog/haproxy-forwards-over-2-million-http-requests-per-second-on-a-single-aws-arm-instance`



Źródło: `https://www.haproxy.com/blog/haproxy-forwards-over-2-million-http-requests-per-second-on-a-single-aws-arm-instance`

Skalowanie

Limit HAProxy

Aktualnie HAProxy jest w stanie obsłużyć 2mln requestów na sekundę. A co jak mamy ich więcej? Jak zrobić load balancing na load balancerach?

- DNS round-robin
- Podejście warstwowe:
  - Najpierw równoważenie ruchu na poziomie IP (L3).
  - Następnie równoważenie TCP (L4).
  - Równoważenie obciążenia na podstawie zawartości HTTP (L7).

LVS

Linux Virtual Server

Serwer równoważący obciążenie na warstwie 4, wbudowany w jądro Linuksa.

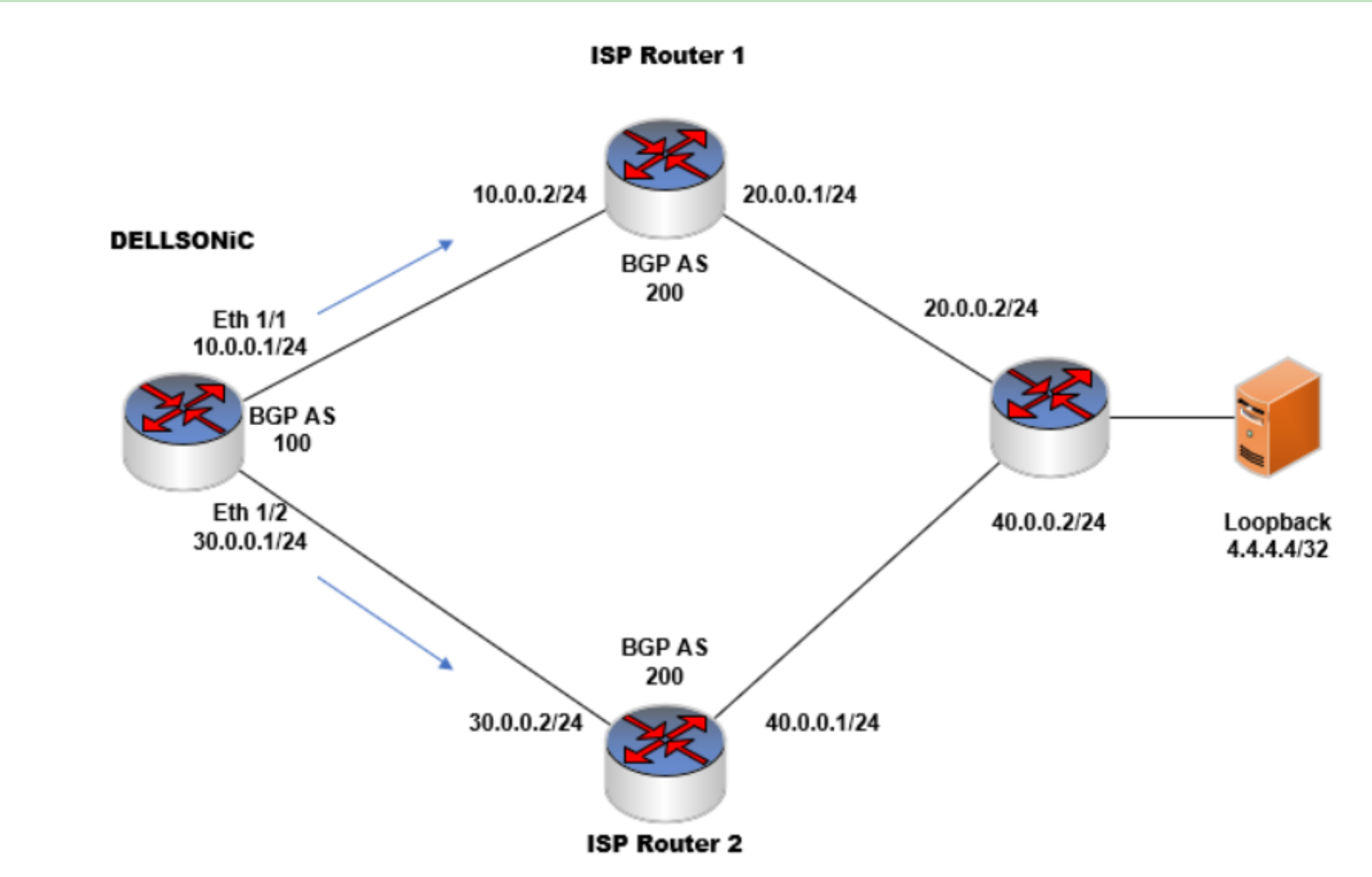
- rr Round Robin
- wrr Weighted Round Robin
- lc Least-Connection
- wlc Weighted Least-Connection (default)
- lblc Locality-Based Least-Connection
- lblcr Locality-Based Least-Connection with Replication
- dh Destination Hashing
- sh Source Hashing
- sed Shortest Expected Delay
- nq Never Queue
- fo Weighted Failover
- ovf Weighted Overflow
- mh Maglev Hashing

ECMP

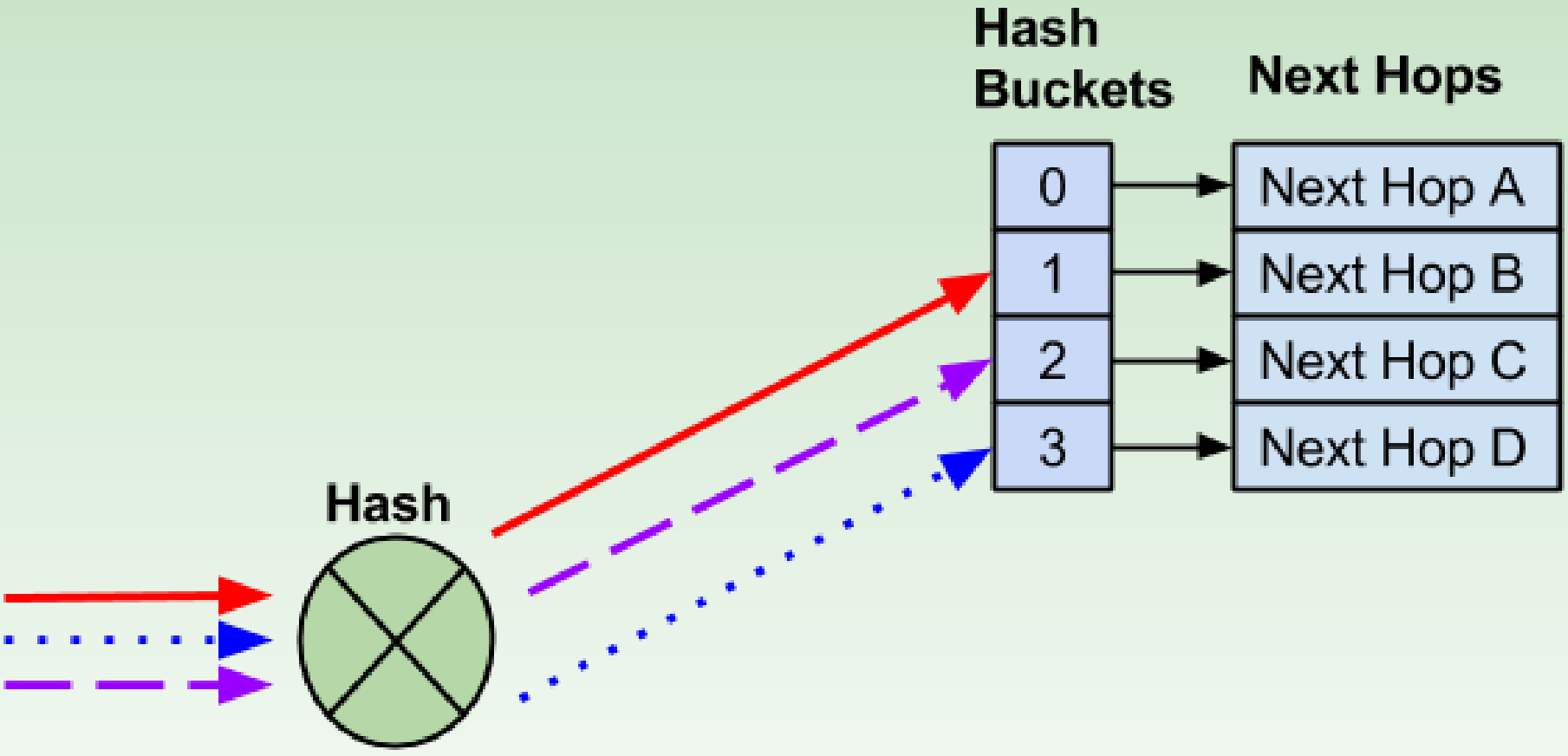
Equal-cost multi-path routing

Sposób równoważenia obciążenia na warstwie 3 i 4, gdzie różne strumienie danych, mogą być wysyłane do tego samego hosta różnymi ścieżkami.

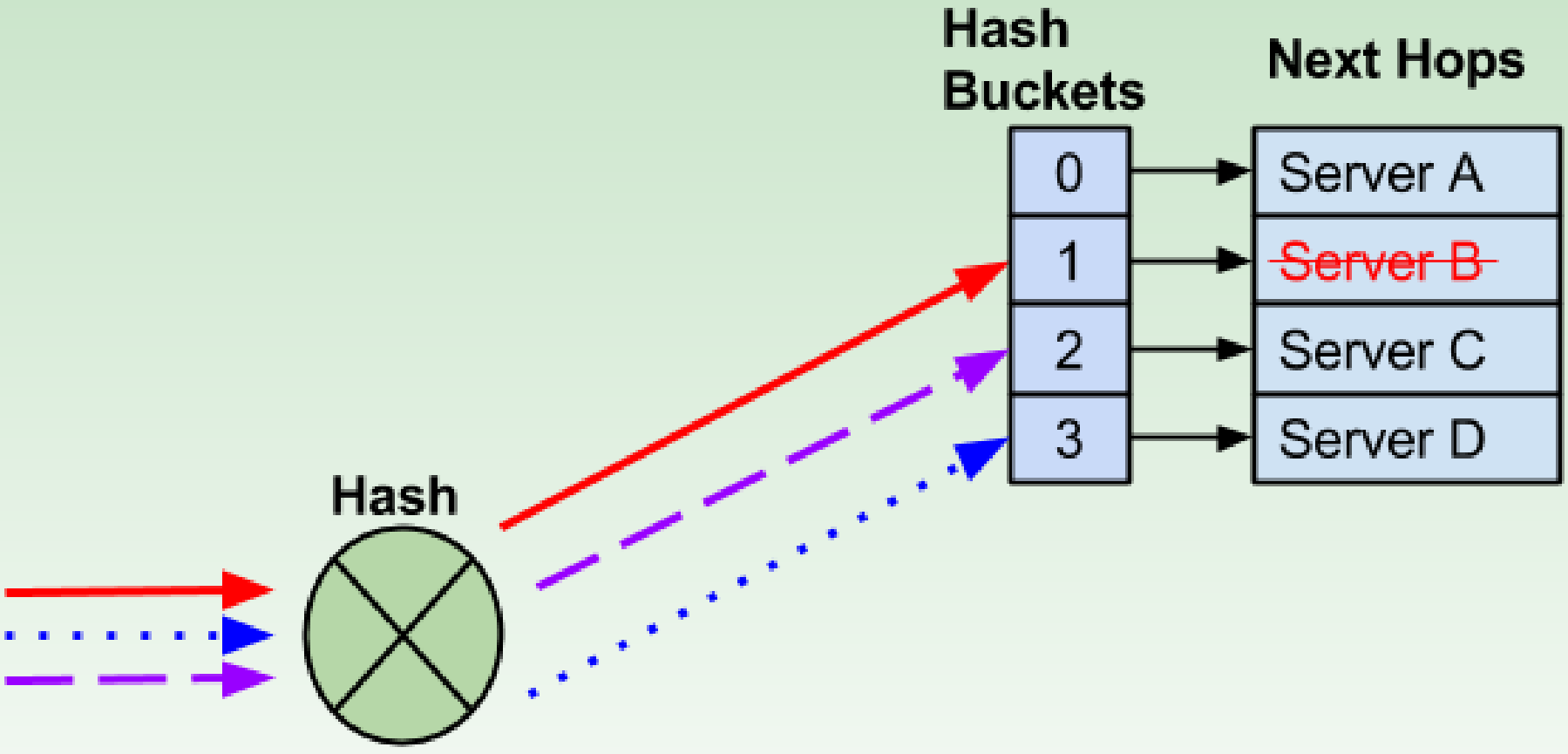
- Losowy wybór ścieżki dla każdego pakietu (przestarzałe)
- Wybór ścieżki na podstawie hasza (per strumień):
  - L3 IP_src, IP_dst
  - L4 IP_src, IP_dst, port_src, port_dst
- Algorytmy:
  - Modulo-N
  - Hash-threshold
  - Resilient hashing



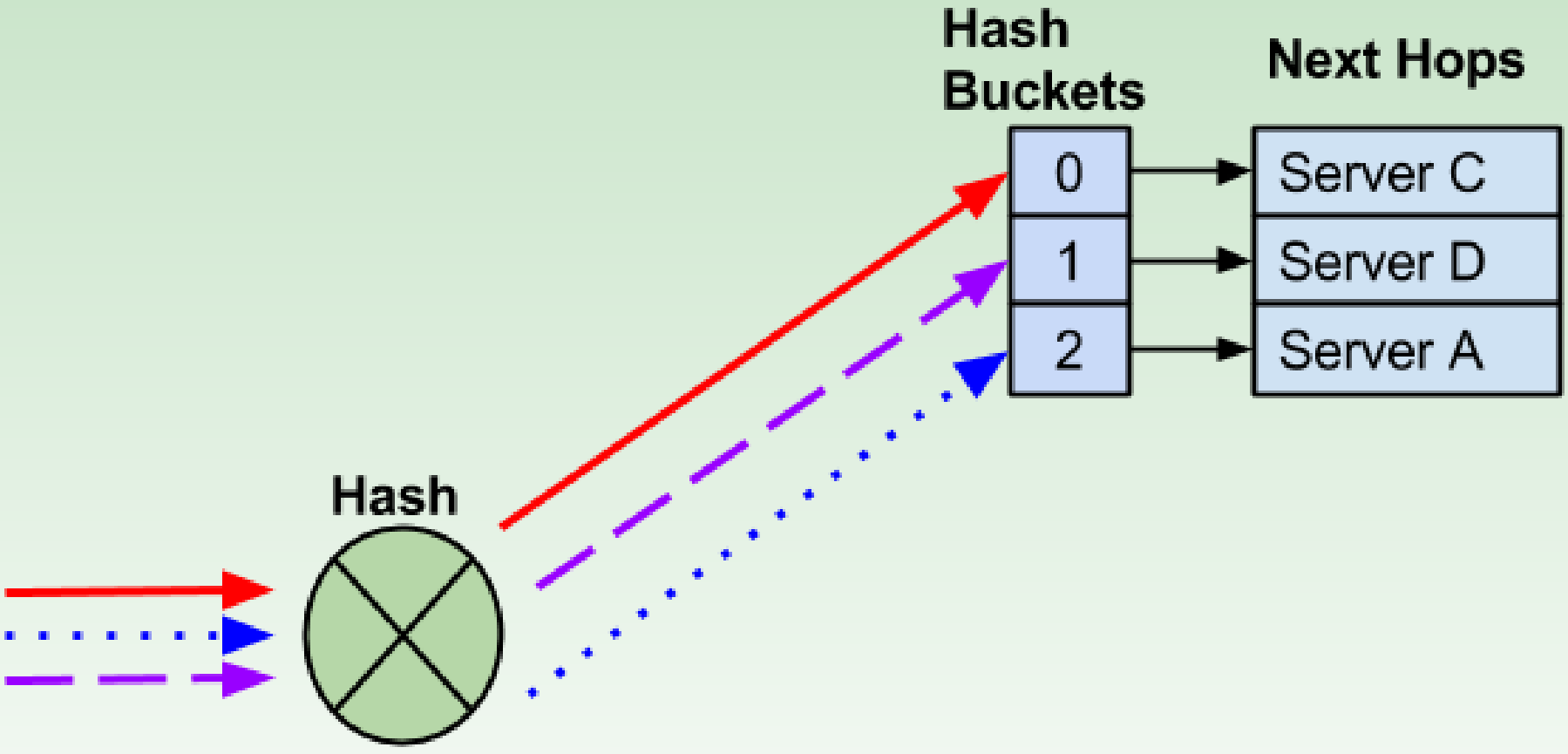
Źródło: https://www.dell.com/support/kbdoc/pl-pl/000220698/dell-networking-sonic-how-to-configure-use-equal-cost-multi-path-in-bgp-with-a-bas



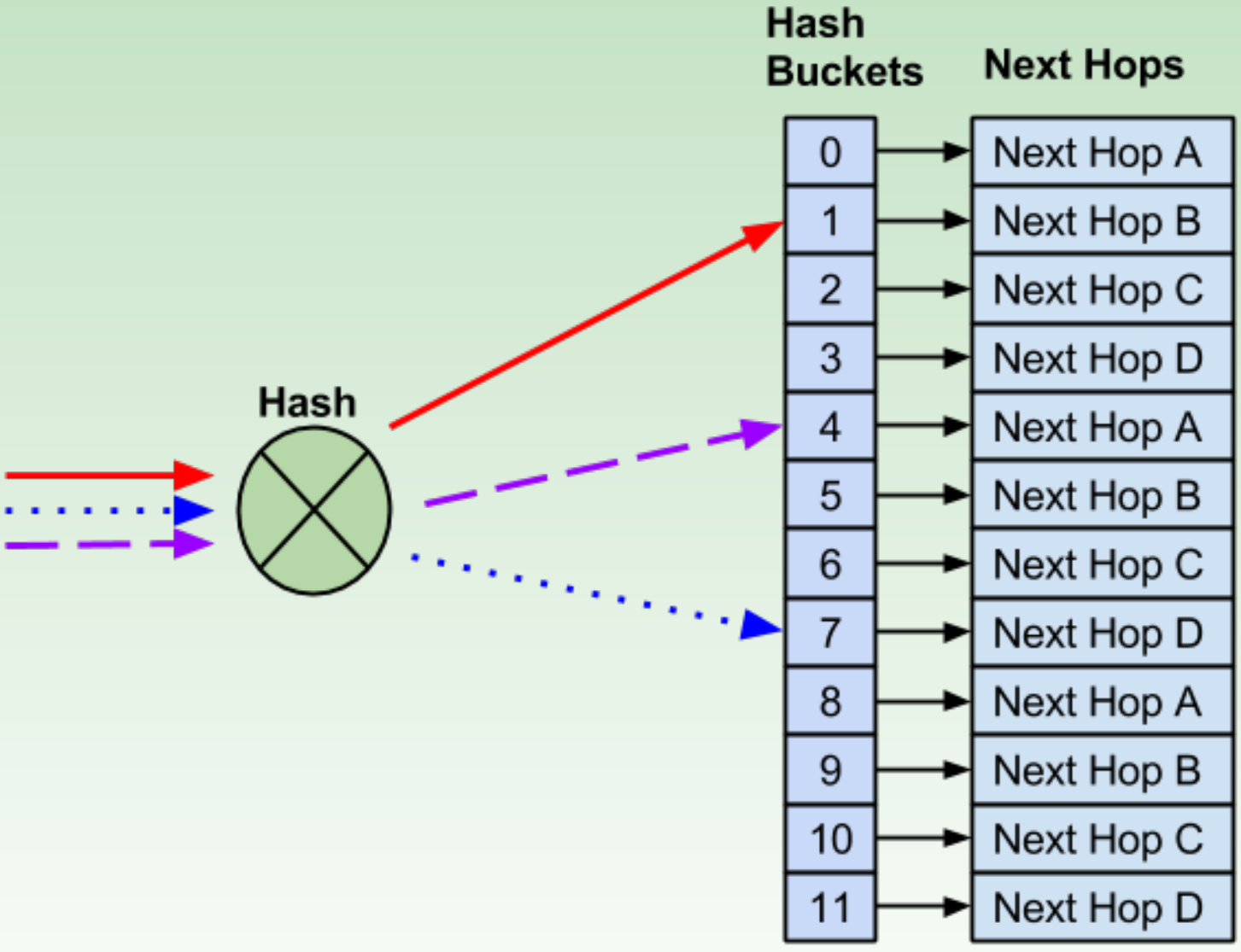
Źródło: https://docs.nvidia.com/networking-ethernet-software/cumulus-linux-44/Layer-3/Routing/Equal-Cost-Multipath-Load-Sharing-Hardware-ECMP/



Źródło: https://docs.nvidia.com/networking-ethernet-software/cumulus-linux-44/Layer-3/Routing/Equal-Cost-Multipath-Load-Sharing-Hardware-ECMP/



Źródło: https://docs.nvidia.com/networking-ethernet-software/cumulus-linux-44/Layer-3/Routing/Equal-Cost-Multipath-Load-Sharing-Hardware-ECMP/



Źródło: https://docs.nvidia.com/networking-ethernet-software/cumulus-linux-44/Layer-3/Routing/Equal-Cost-Multipath-Load-Sharing-Hardware-ECMP/

```
cumulus@switch:~$ ip route show 10.1.1.0/24
10.1.1.0/24  proto zebra  metric 20
  nexthop via 192.168.1.1 dev swp1 weight 1 onlink
  nexthop via 192.168.2.1 dev swp2 weight 1 onlink
```

Źródło: https://docs.nvidia.com/networking-ethernet-software/cumulus-linux-44/Layer-3/Routing/Equal-Cost-Multipath-Load-Sharing-Hardware-ECMP/

Haszowanie z minimalnymi zakłóceniami

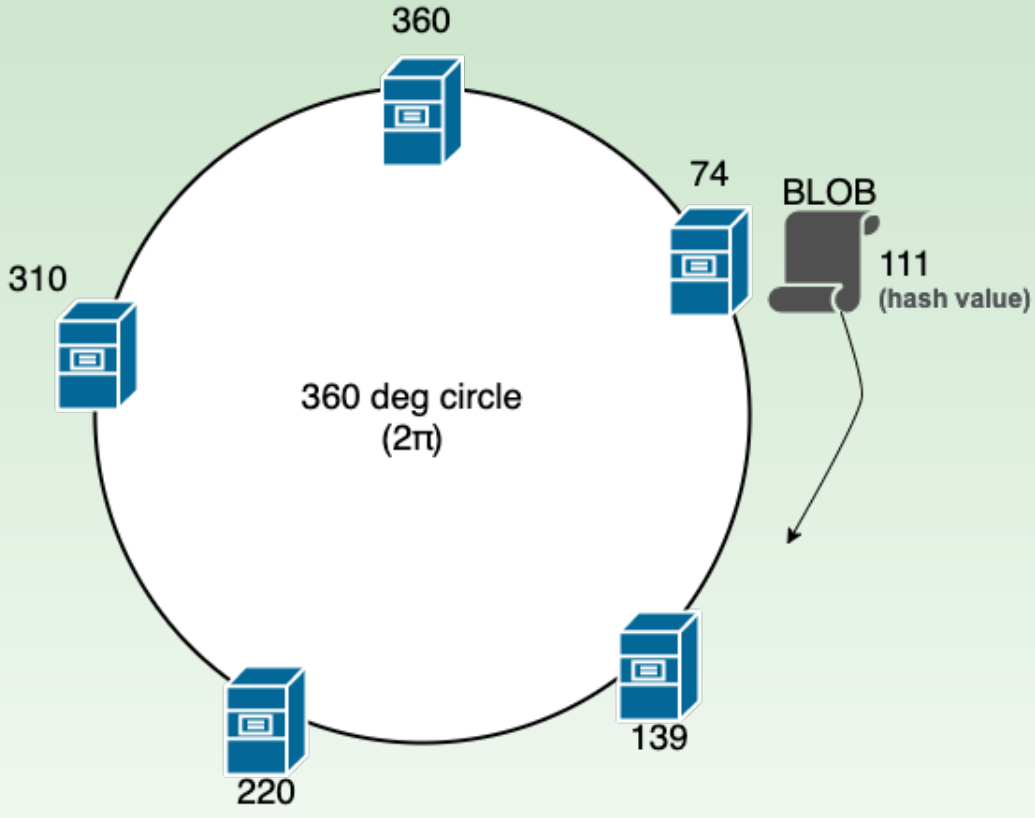
Cel

Po dodaniu/usunięciu dowolnego kubełka z tablicy haszującej, tylko $O(n/m)$ elementów musi zmienić miejsce (gdzie n, to liczba elementów, a m to liczba kubełków).

- Consistent hashing
- Rendezvous (albo highest random weight) hashing

Consistent hashing

Rozmieszczamy kubełki na kole jednostkowym, przedmioty haszujemy na to koło i wsadzamy do następnego kubełka zgodnie z ruchem wskazówek zegara.

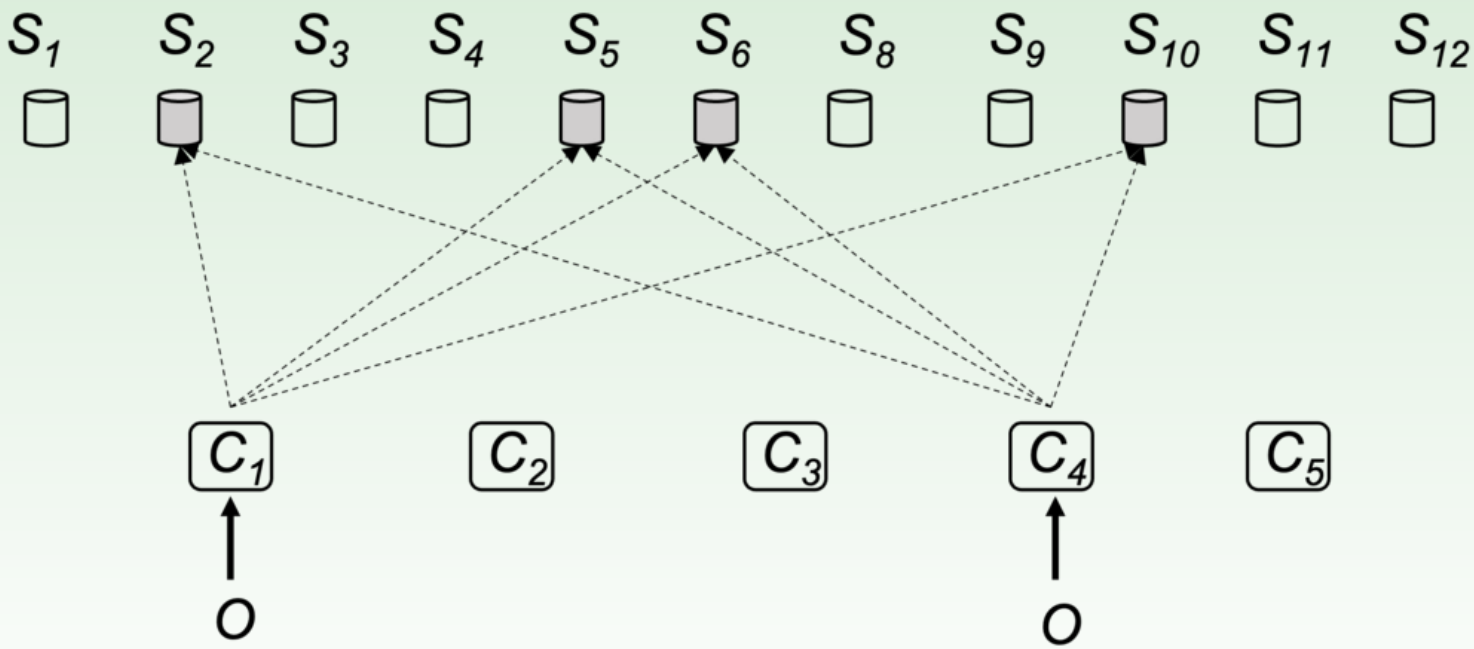


Źródło: `https://commons.wikimedia.org/wiki/File:Consistent_Hashing_Sample_Illustration.png`

Rendezvous hashing

Algorytm

Dla każdego kubełka ustalamy inne ziarno, następnie haszujemy obiekt z ziarnem każdego kubełka i umieszczamy go w k kubełkach o największym haszu. Dla ustalonej funkcji haszującej i ziaren otrzymujemy rozwiązanie problemu rozproszonej tablicy haszującej.



Źródło: https://en.wikipedia.org/wiki/File:Rendezvous_Hash_Schematic.png


Beyer, Betsy i in. (2018). *The Site Reliability Workbook: Practical Ways to Implement SRE*. 1st. O'Reilly Media, Inc. ISBN: 1492029505. URL: https://sre.google/workbook/table-of-contents/.