From XDP to Socket

Routing of packets beyond XDP with BPF

Udip Pant
Software Engineer

Martin Lau
Software Engineer

FACEBOOK Infrastructure

About

XDP: 1.5 years in production. Evolution and lessons learned.

Author: Nikita V. Shirokov

About

XDP: 1.5 years in production. Evolution and lessons learned.

Author: Nikita V. Shirokov

XDP enabled application

L4 load balancer:
https://github.com/facebookincubator/katran

Reason for L4 load balancing:
https://atscaleconference.com/videos/networking-scale-2018-layer-4-load-balancing-at-facebook/

About

XDP: 1.5 years in production. Evolution and lessons learned.

Author: Nikita V. Shirokov

XDP enabled application

L4 load balancer:
https://github.com/facebookincubator/katran

Reason for L4 load balancing:
https://atscaleconference.com/videos/networking-scale-2018-layer-4-load-balancing-at-facebook/

Operational Experience

Every packet toward facebook.com has been processed by XDP enabled application since May, 2017

Introduce BPF_MAP_TYPE_REUSEPORT_SOCKARRAY and BPF_PROG_TYPE_SK_REUSEPORT

From: Martin KaFai Lau <kafai-AT-fb.com>
To: <netdev-AT-vger.kernel.org>
Subject: [PATCH bpf-next 0/9] Introduce BPF_MAP_TYPE_REUSEPORT_SOCKARRAY and BPF_PROG_TYPE_SK_REUSEPORT
Date: Wed, 8 Aug 2018 00:59:17 -0700
Message-ID: <20180808075917.3009181-1-kafai@fb.com>
Cc: Alexei Starovoitov <ast-AT-fb.com>, Daniel Borkmann <daniel-AT-iogearbox.net>, <kernel-team-AT-fb.com>
Archive-link: Article

```
This series introduces a new map type "BPF_MAP_TYPE_REUSEPORT_SOCKARRAY"
and a new prog type BPF_PROG_TYPE_SK_REUSEPORT.

Here is a snippet from a commit message:

"To unleash the full potential of a bpf prog, it is essential for the
userspace to be capable of directly setting up a bpf map which can then
be consumed by the bpf prog to make decision.  In this case, decide which
SO_REUSEPORT sk to serve the incoming request.

By adding BPF_MAP_TYPE_REUSEPORT_SOCKARRAY, the userspace has total control
and visibility on where a SO_REUSEPORT sk should be located in a bpf map.
The later patch will introduce BPF_PROG_TYPE_SK_REUSEPORT such that
the bpf prog can directly select a sk from the bpf map.  That will
raise the programmability of the bpf prog attached to a reuseport
group (a group of sk serving the same IP:PORT).

For example, in UDP, the bpf prog can peek into the payload (e.g.
through the "data" pointer introduced in the later patch) to learn
the application level's connection information and then decide which sk
to pick from a bpf map.  The userspace can tightly couple the sk's location
in a bpf map with the application logic in generating the UDP payload's
connection information.  This connection info contact/API stays within the
userspace.

Also, when used with map-in-map, the userspace can switch the
old-server-process's inner map to a new-server-process's inner map
in one call "bpf_map_update_elem(outer_map, &index, &new_reuseport_array)".
The bpf prog will then direct incoming requests to the new process instead
of the old process.  The old process can finish draining the pending
requests (e.g. by "accept()") before closing the old-fds.  [Note that
deleting a fd from a bpf map does not necessary mean the fd is closed]"

Please see individual patch for details

Martin KaFai Lau (9):
```

[PATCH v3 bpf-next 0/9] BPF TCP header options

[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

- *Subject*: [PATCH v3 bpf-next 0/9] BPF TCP header options
- *From*: Martin KaFai Lau <kafai@xxxxxx>
- *Date*: Thu, 30 Jul 2020 13:56:57 -0700
- *Cc*: Alexei Starovoitov <ast@xxxxxxxxxx>, Daniel Borkmann <daniel@xxxxxxxxxxxxx>, Eric Dumazet <netdev@xxxxxxxxxxxxxxx>, Yuchung Cheng <ycheng@xxxxxxxxxx>
- *Smtp-origin-cluster*: ftw2c04
- *Smtp-origin-hostname*: devbig005.ftw2.facebook.com
- *Smtp-origin-hostprefix*: devbig

```
The earlier effort in BPF-TCP-CC allows the TCP Congestion Control
algorithm to be written in BPF.  It opens up opportunities to allow
a faster turnaround time in testing/releasing new congestion control
ideas to production environment.

The same flexibility can be extended to writing TCP header option.
It is not uncommon that people want to test new TCP header option
to improve the TCP performance.  Another use case is for data-center
that has a more controlled environment and has more flexibility in
putting header options for internal traffic only.
```

Overview

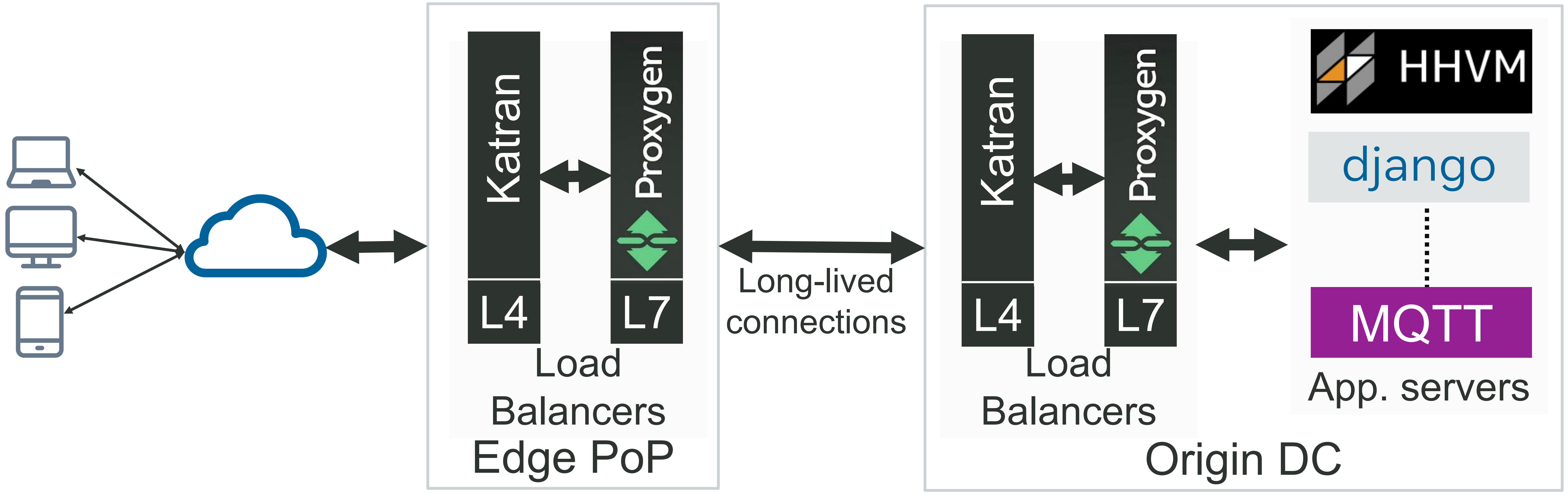
Part I: *Zero downtime restart* of L7 service

- Motivation
- Problems with existing approach
- `bpf_sk_reuseport` for efficiency and operational wins

Part II: Consistent and stateless routing of TCP Packets

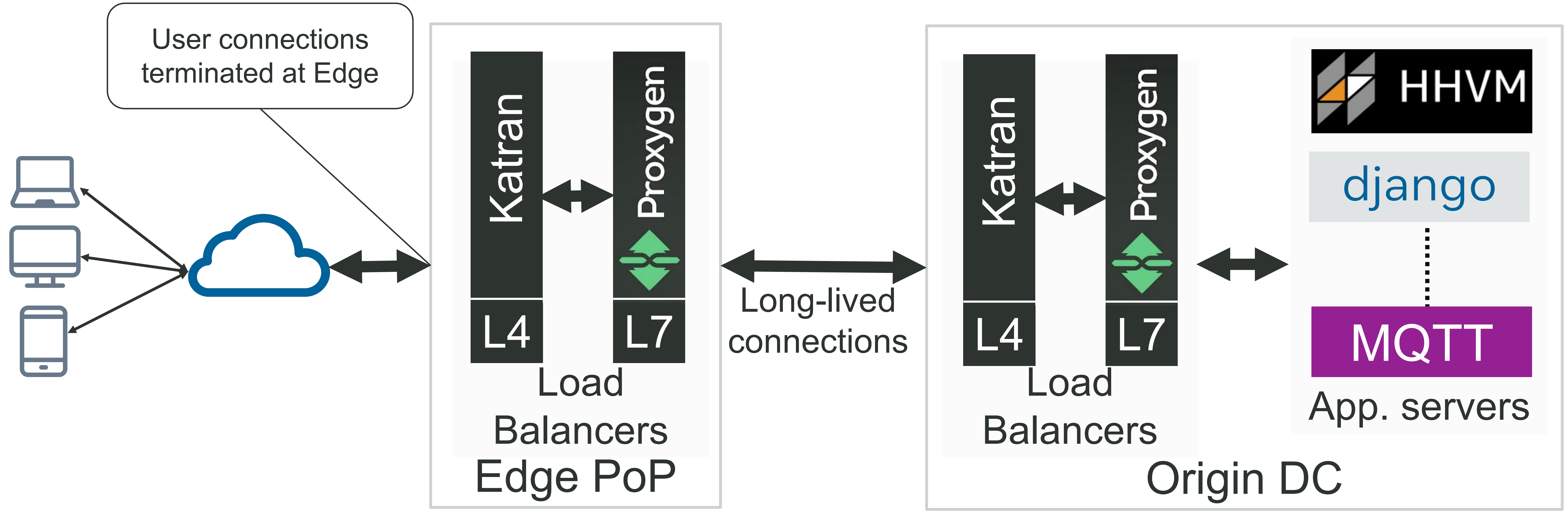
- Limitations of Consistent Hashing
- Embed server info with BPF TCP Header options (sock_ops)

Traffic Infrastructure @ FB



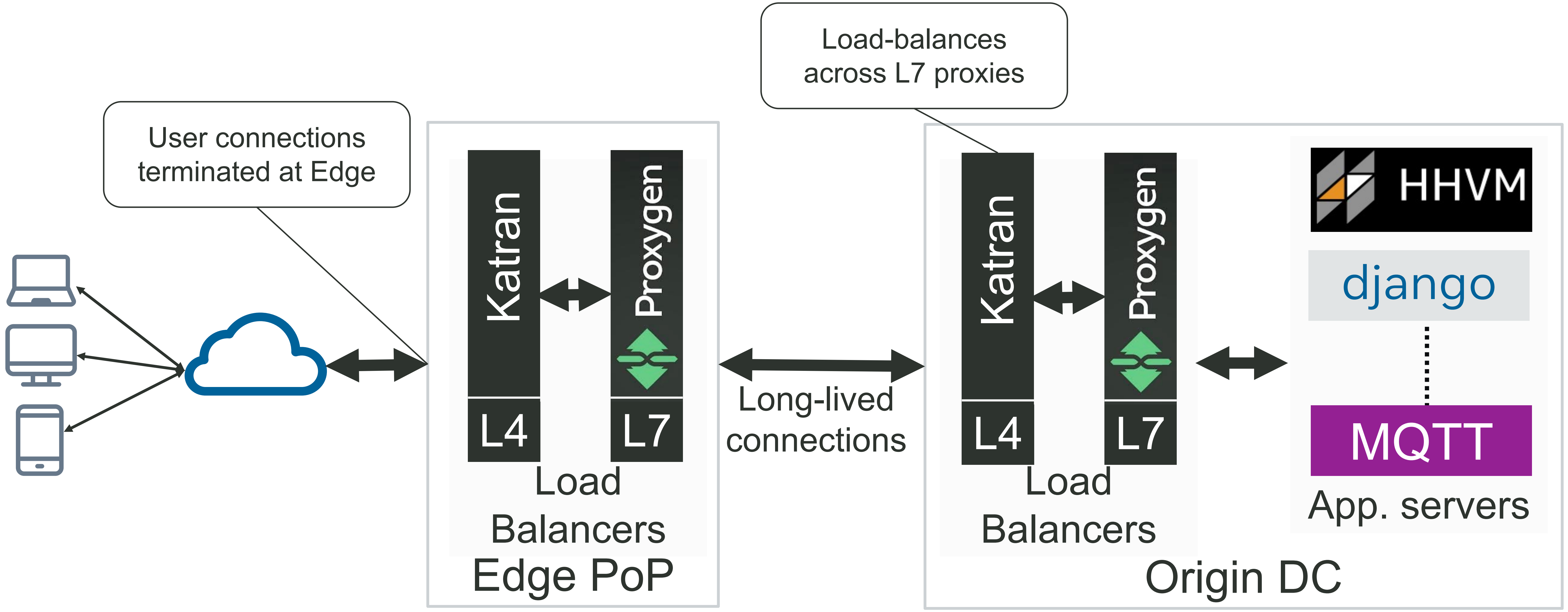
Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

Traffic Infrastructure @ FB



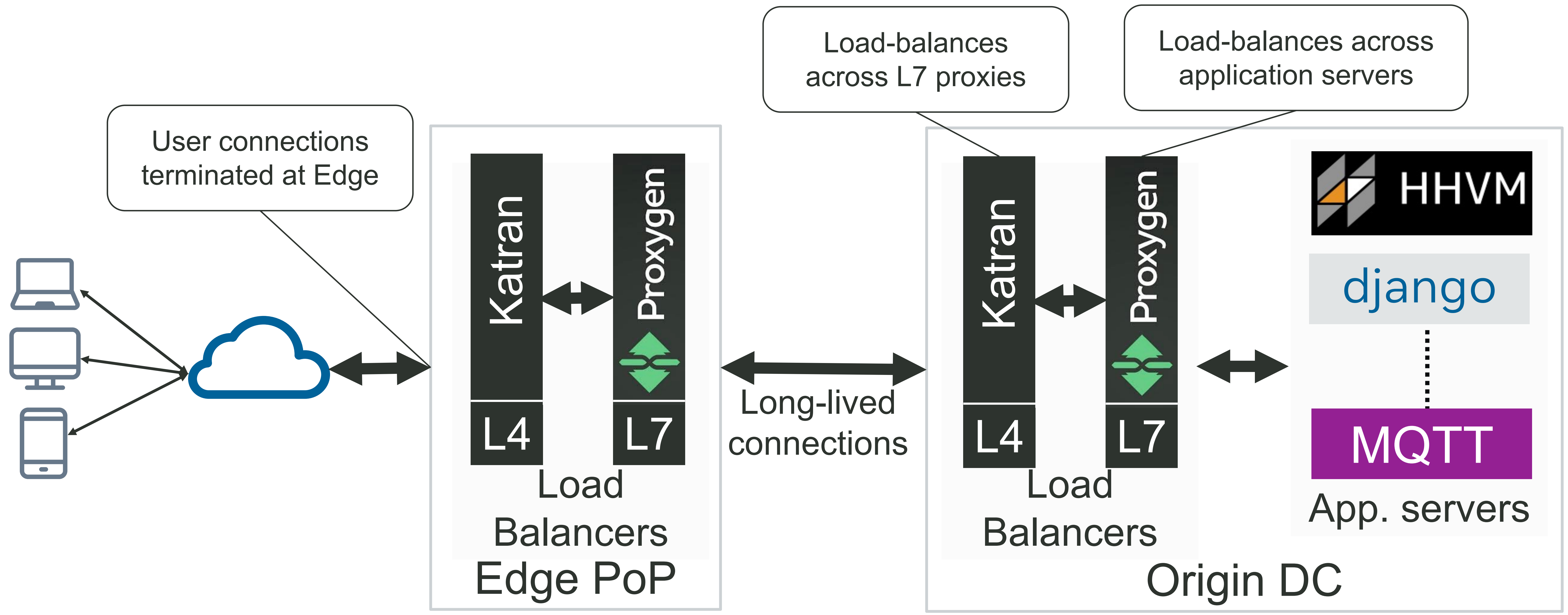
Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

Traffic Infrastructure @ FB



Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

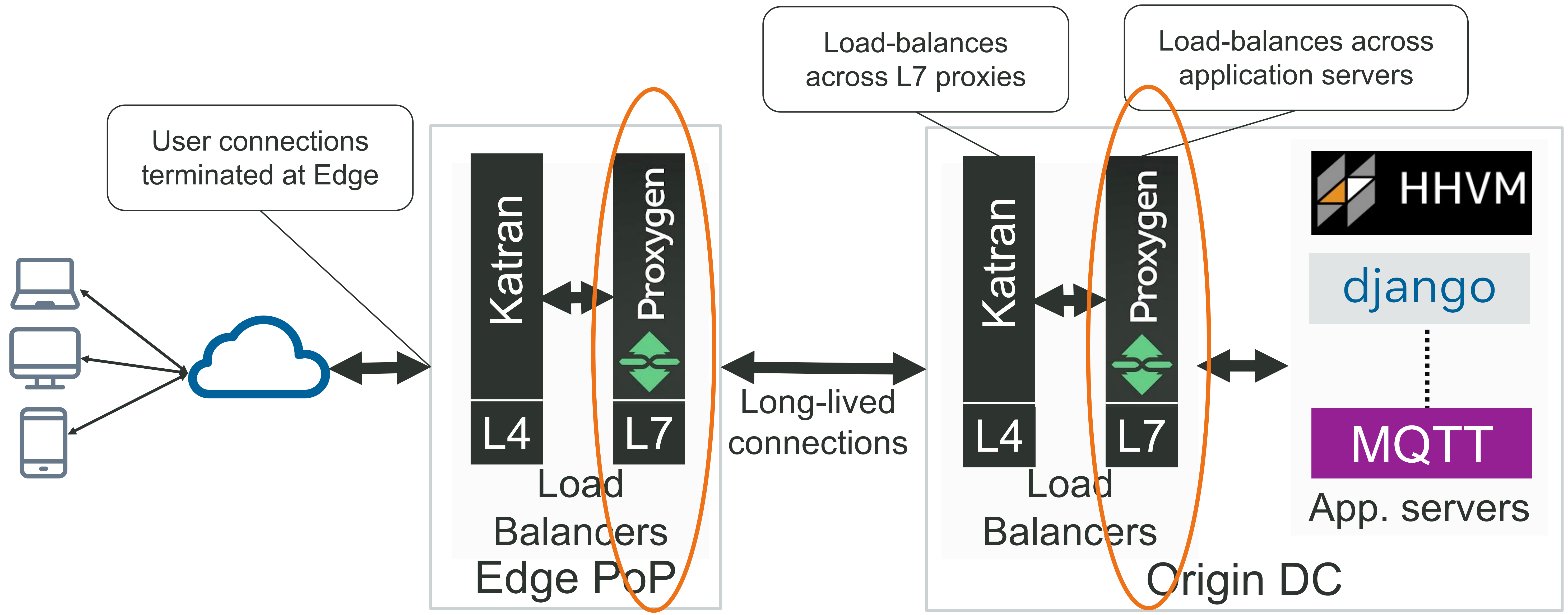
Traffic Infrastructure @ FB



Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

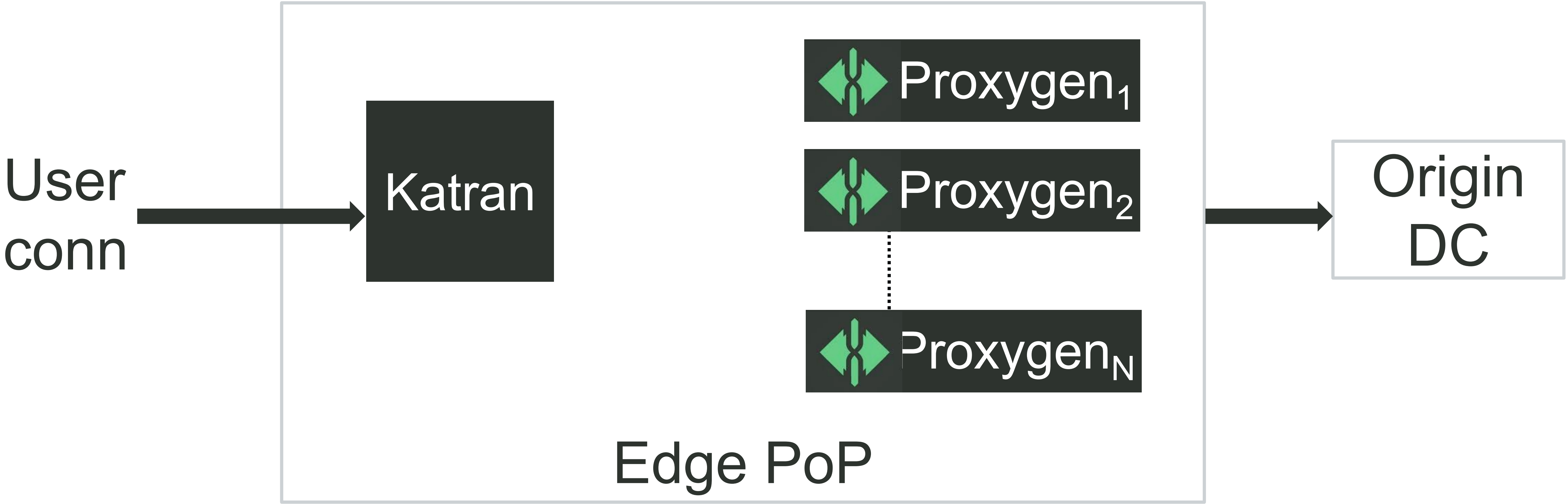
Part I: Routing of packets within a host for *Zero Downtime Restarts*

Traffic Infrastructure @ FB



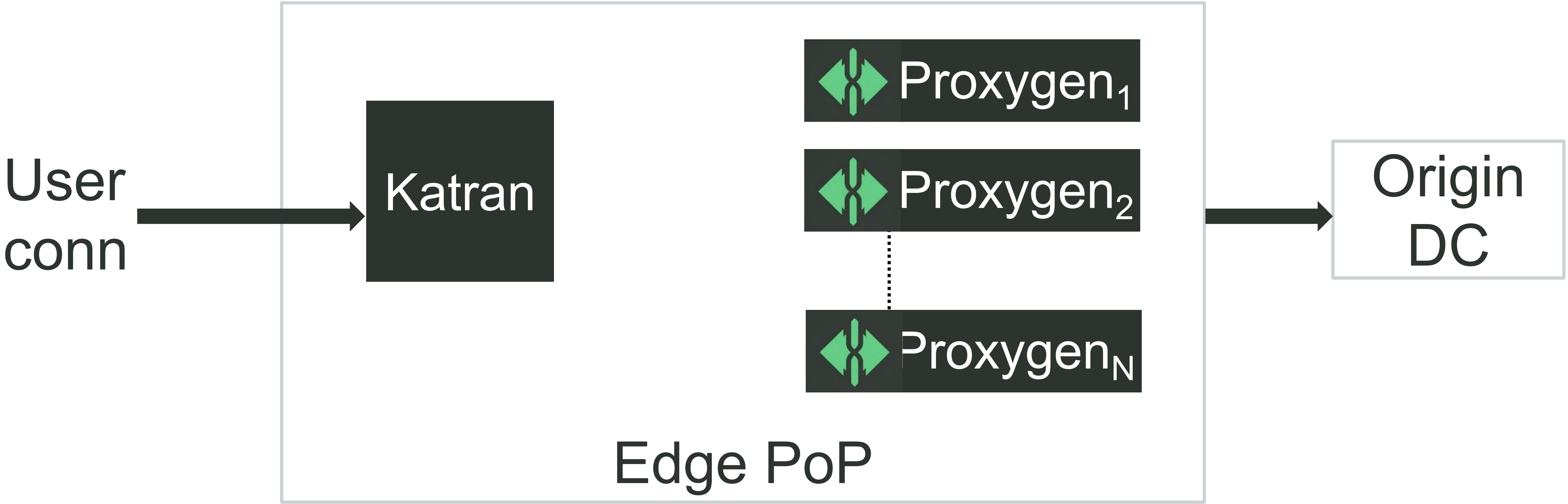
Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

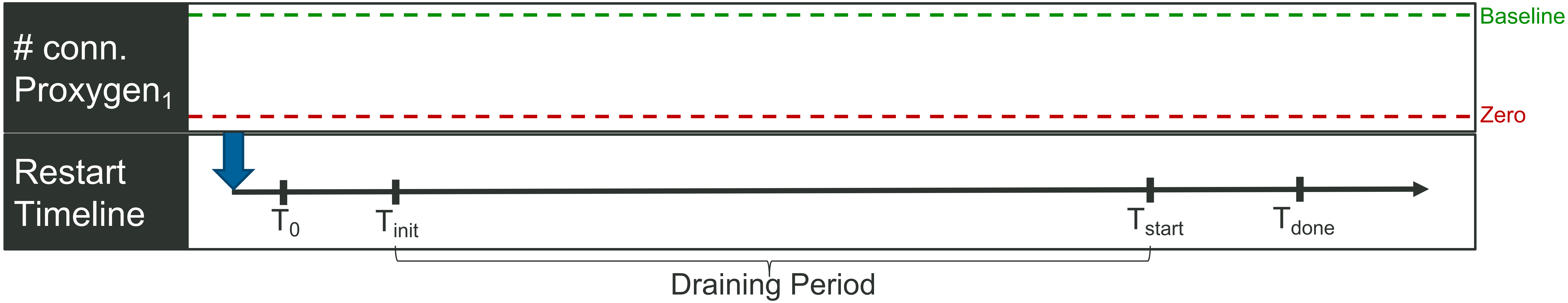
Traditional Restarts



Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-

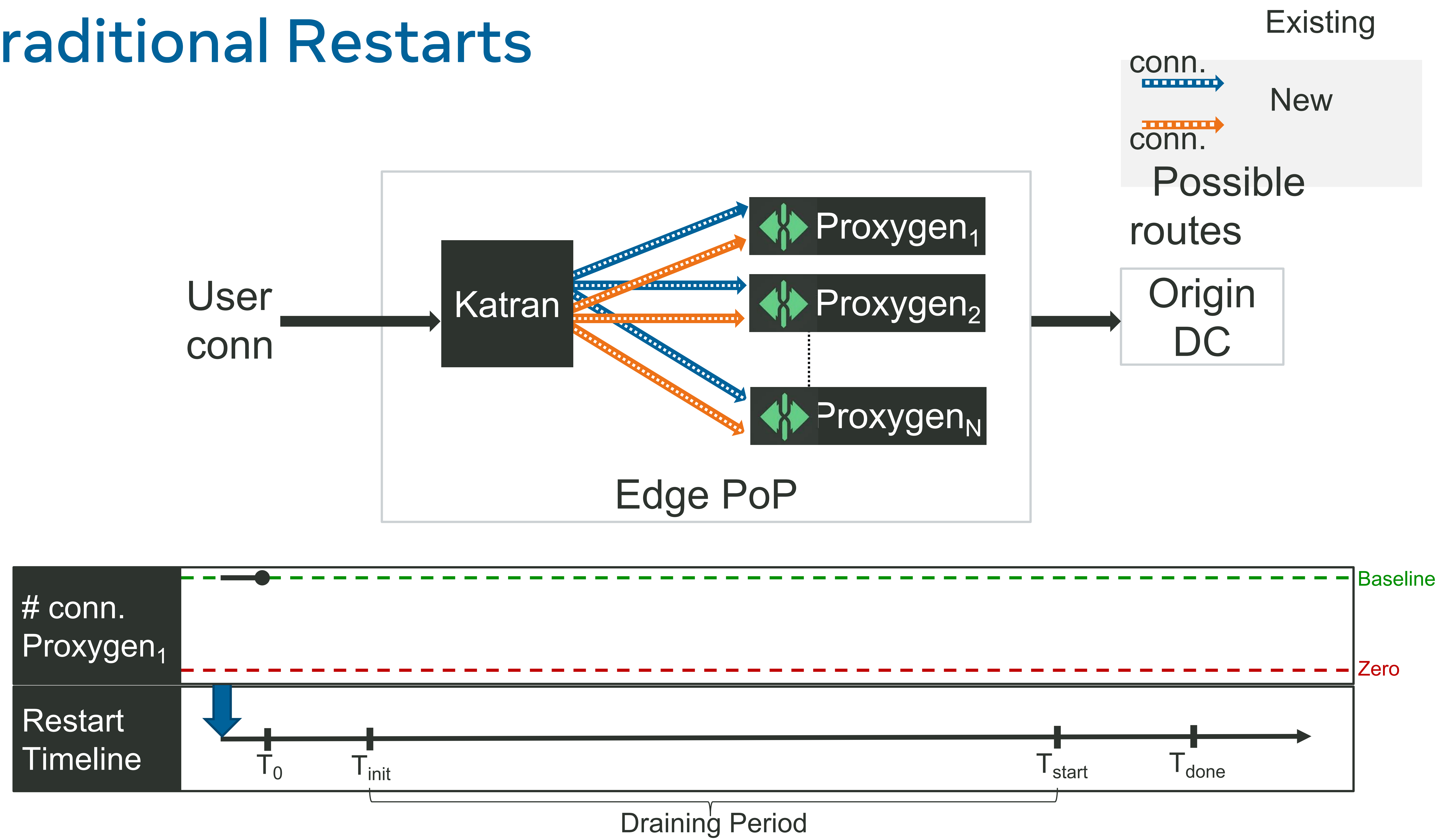
Traditional Restarts





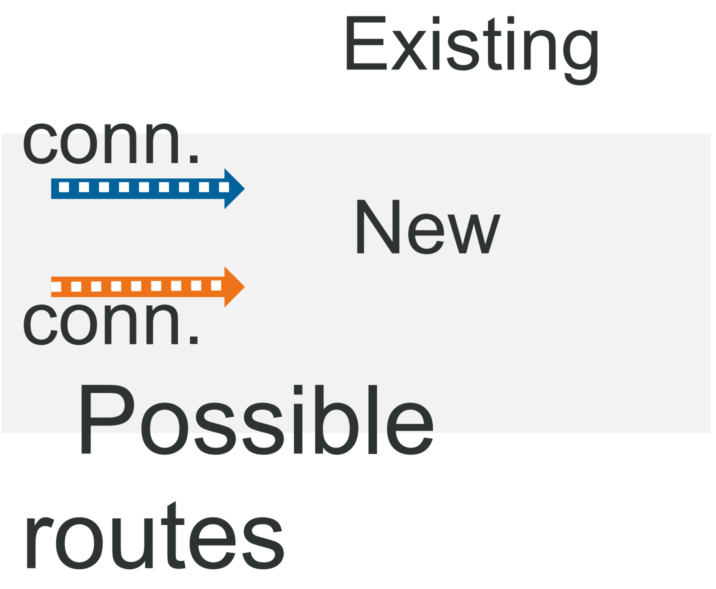
Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-

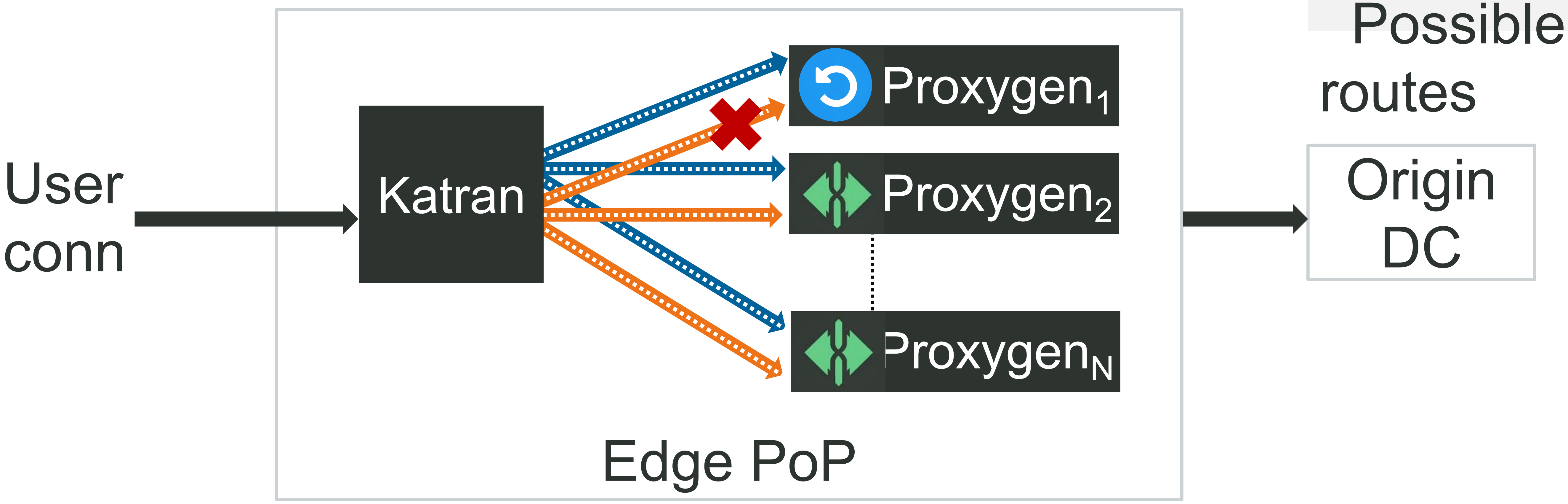
Traditional Restarts

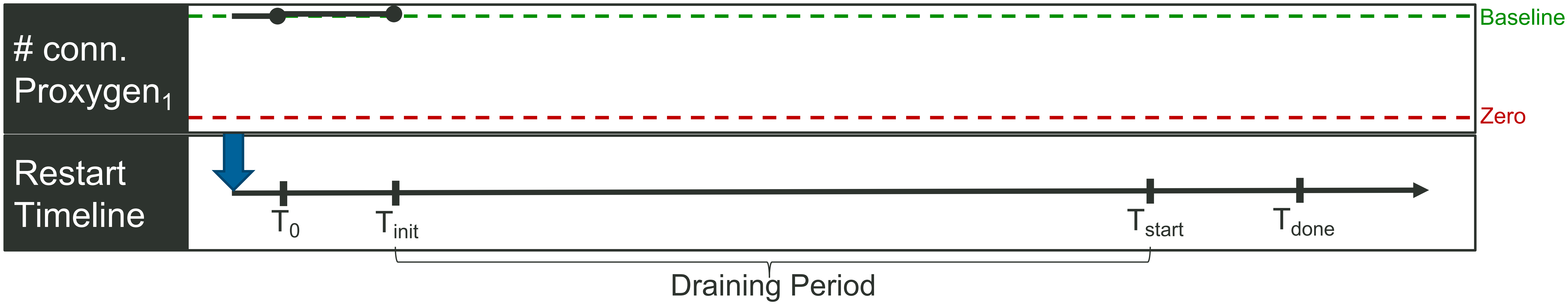


Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-

Traditional Restarts

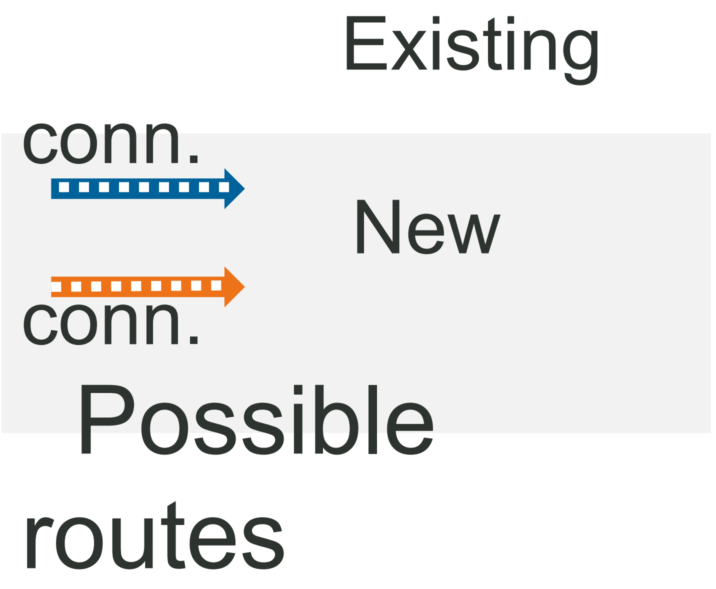


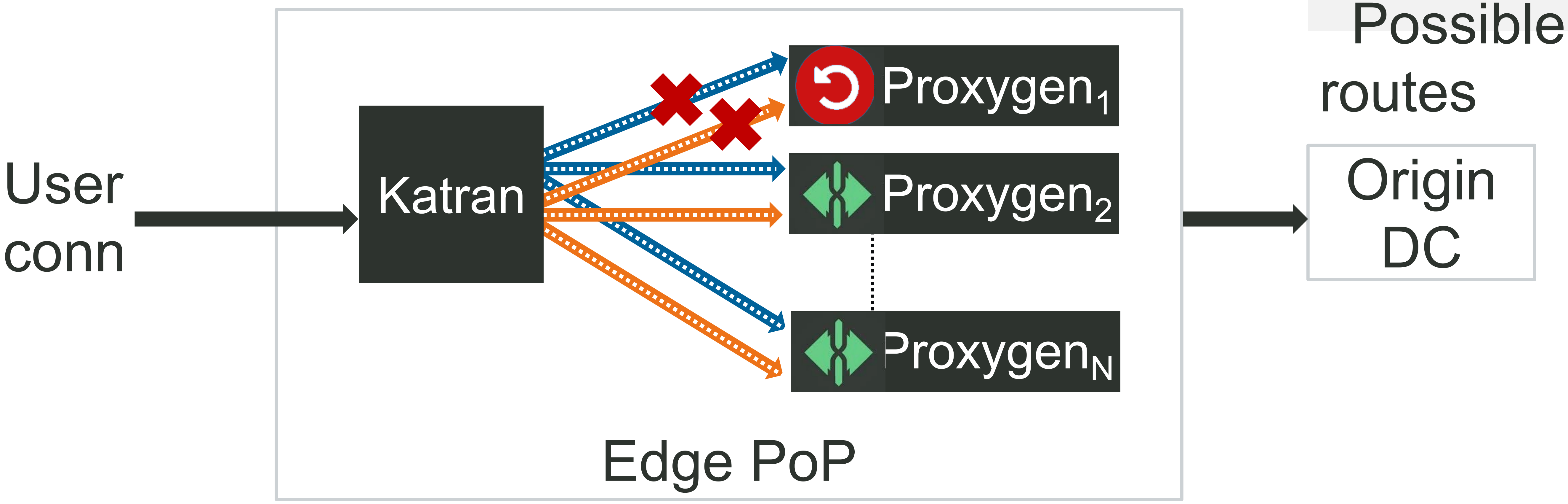


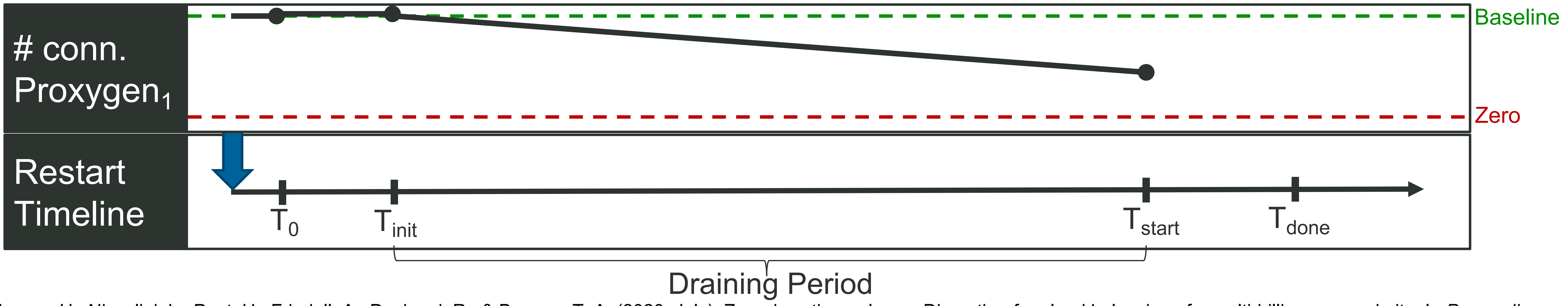


Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-

Traditional Restarts

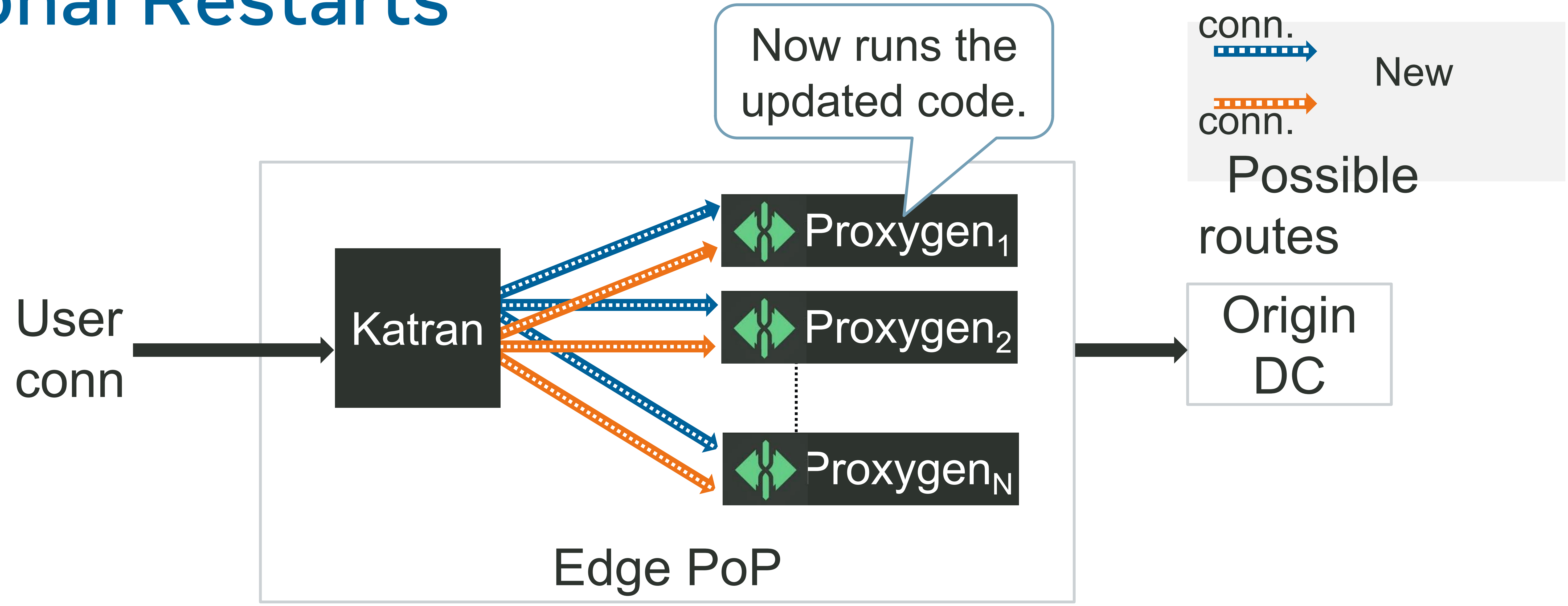


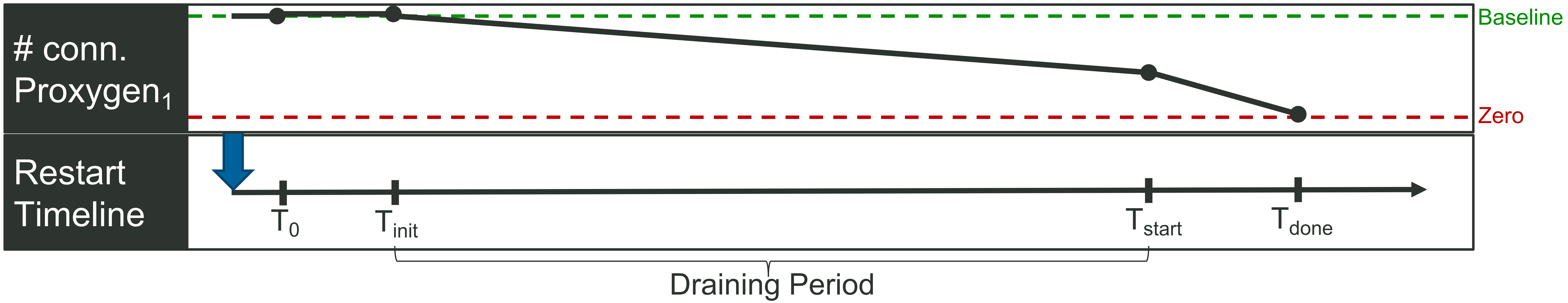




Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-

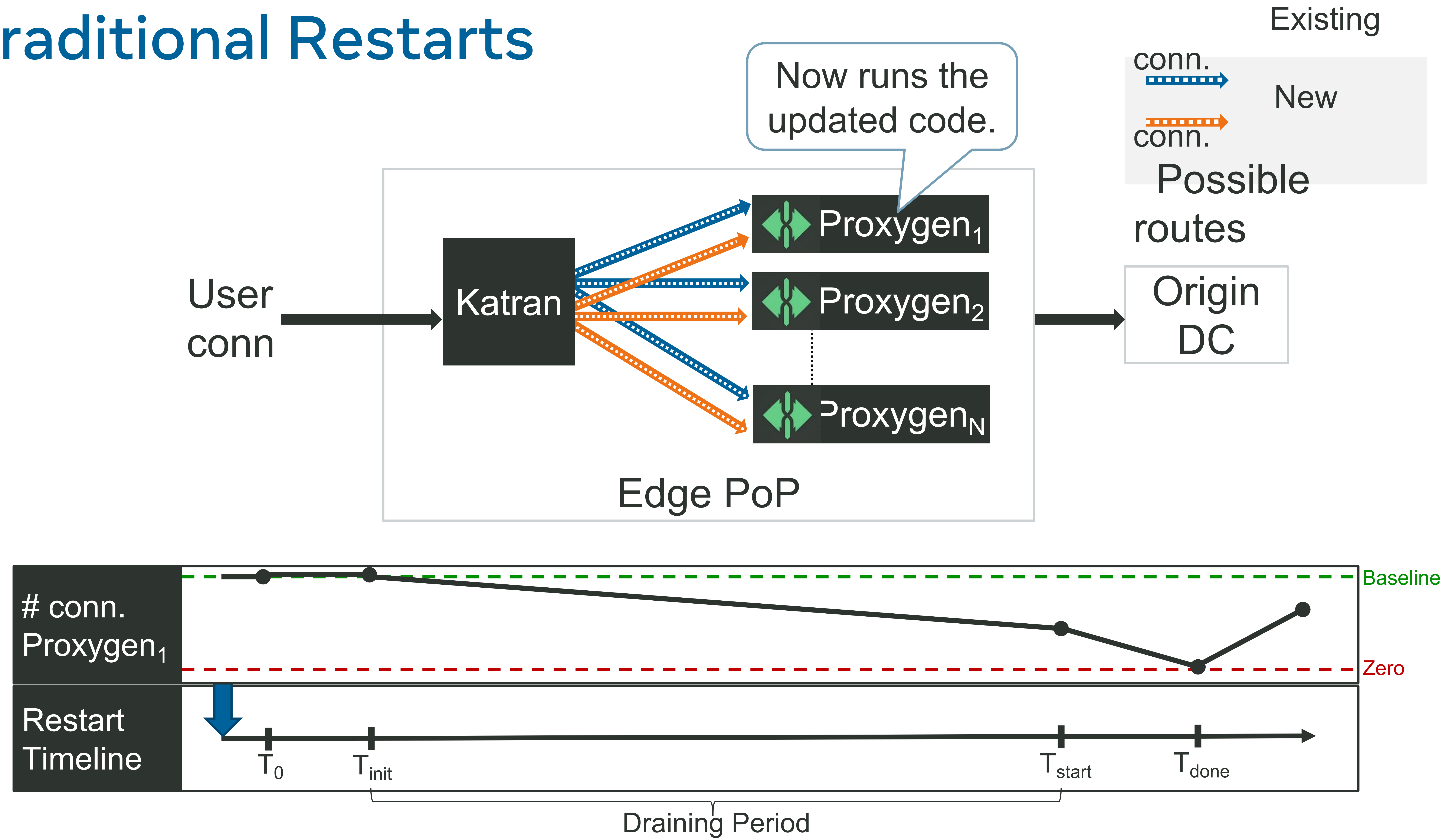
Traditional Restarts





Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-

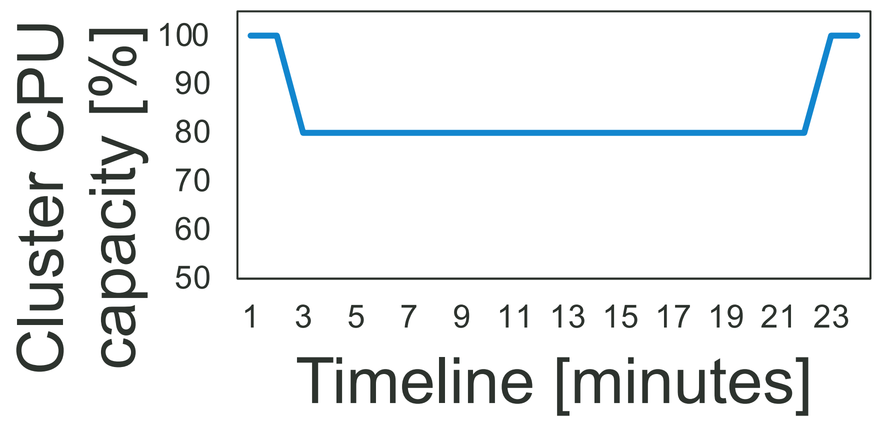
Traditional Restarts



Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-

Implications

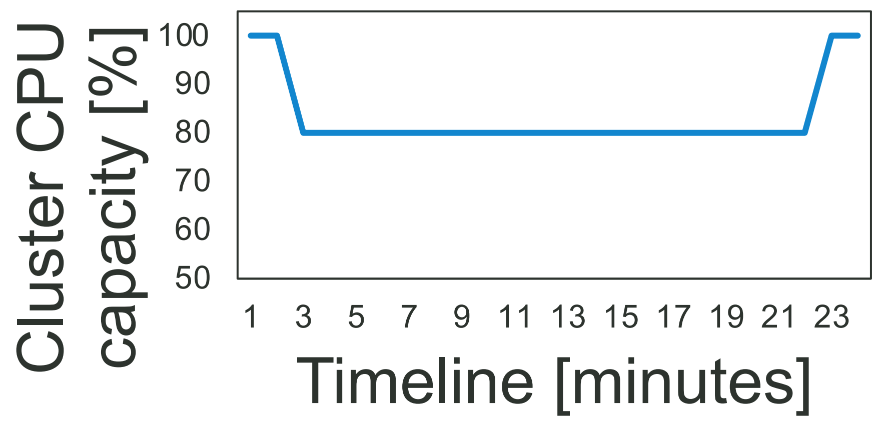
- **Reduced cluster CPU capacity.**
  - Lower # of instances available.



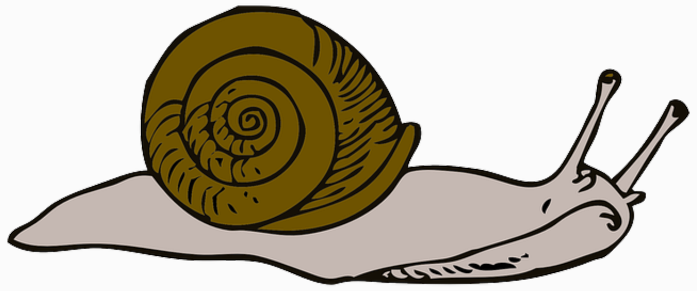
Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

Implications

- **Reduced cluster CPU capacity.**
  - Lower # of instances available.



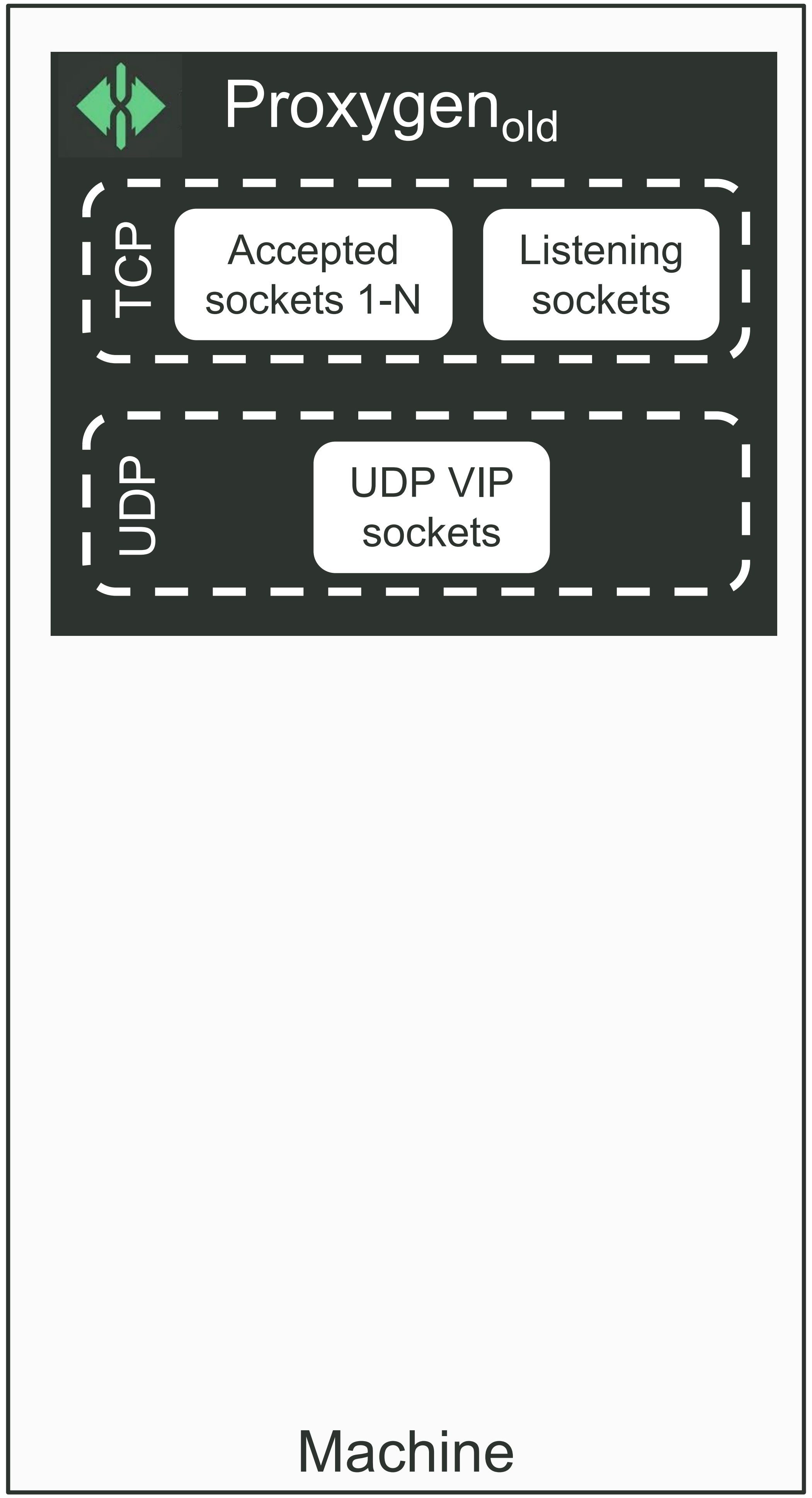
- **Slow update speed.**
  - Unable to “move fast”.



Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

How to release updates while ensuring no disruptions, zero downtime and fast iterations?

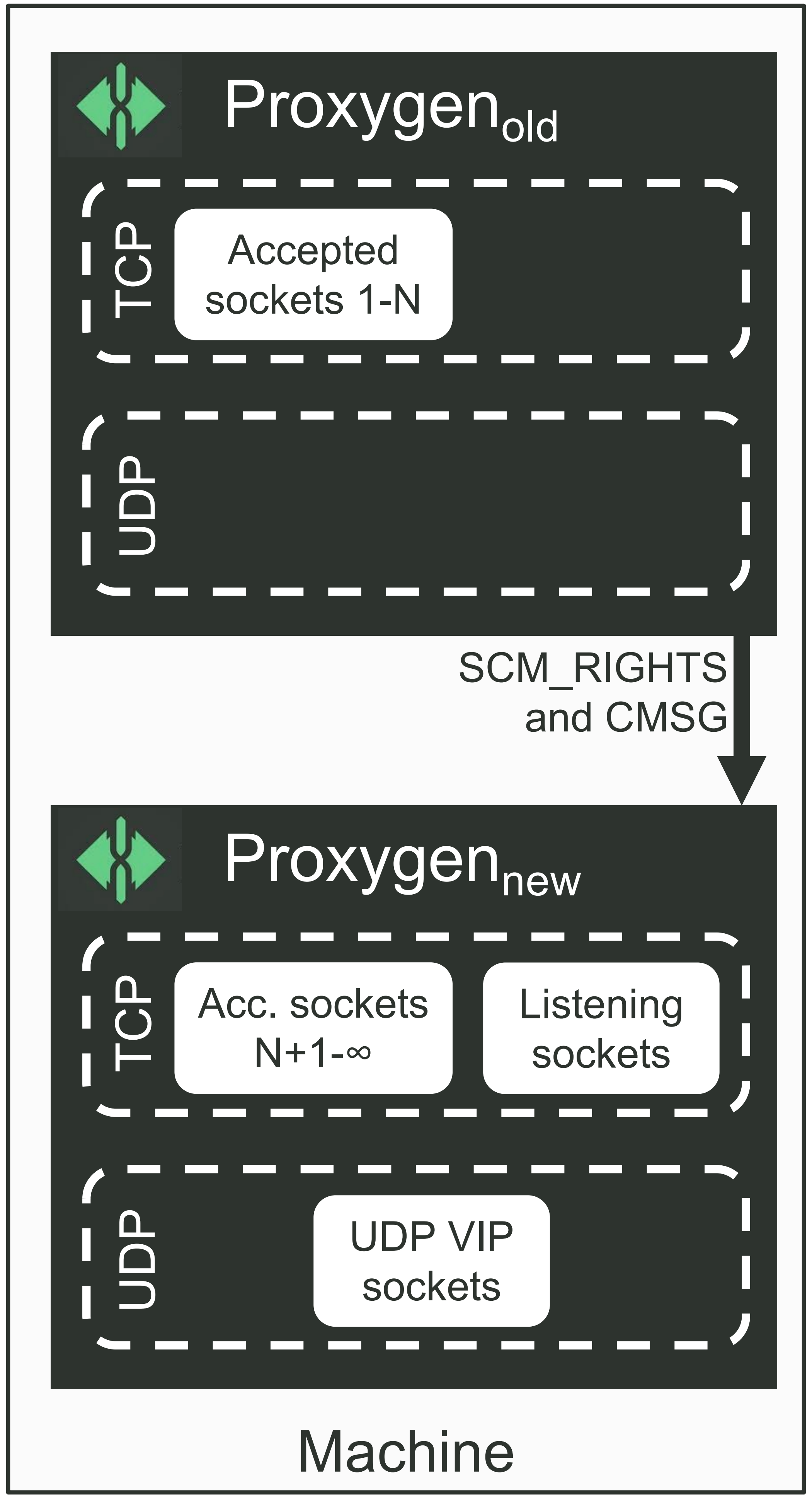
Socket Takeover (Proxygen restarts)



Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

Socket Takeover (Proxygen restarts)

- Takeover TCP listening and UDP VIP sockets.
  - Old instance drains, updated instance handles new connections.

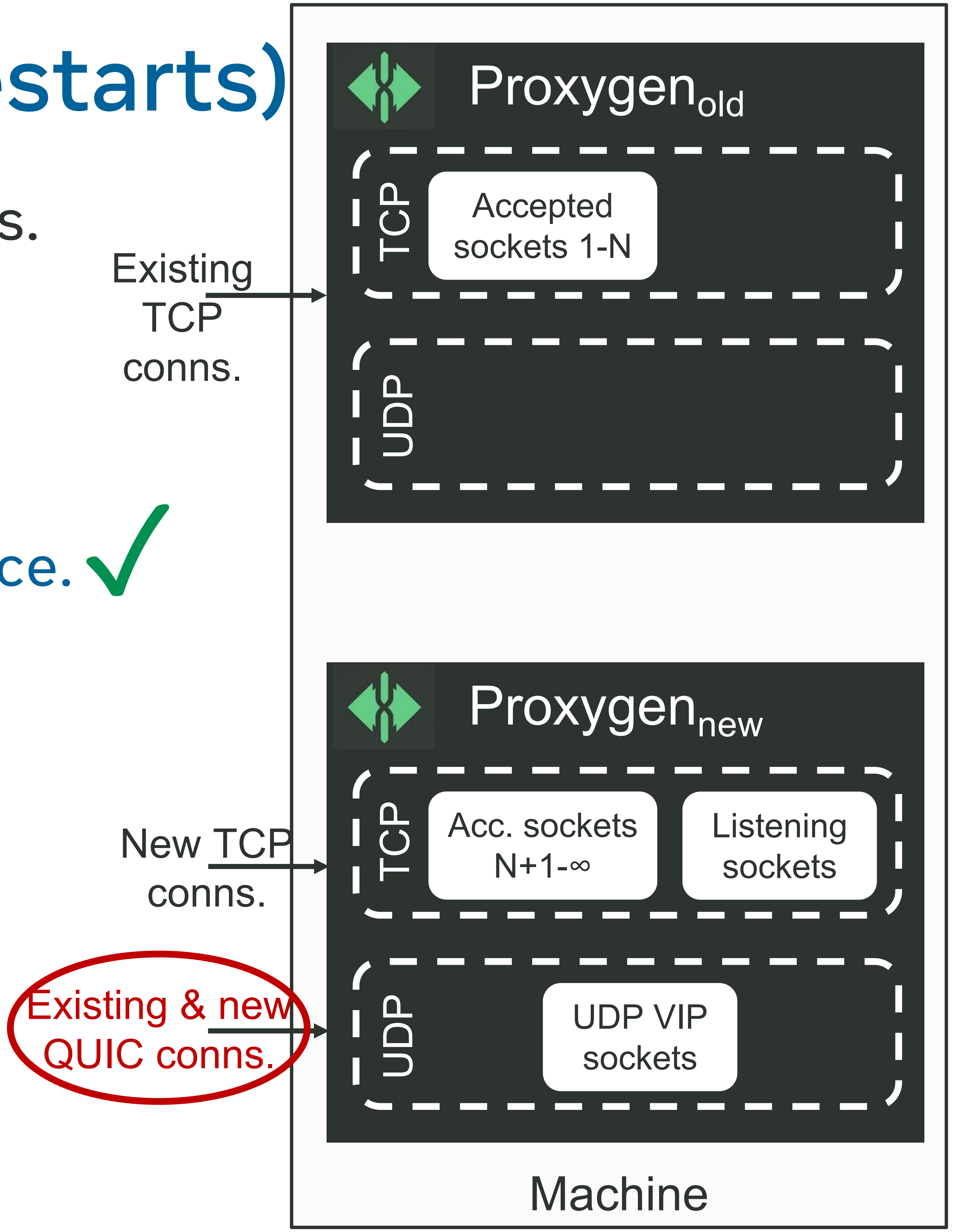


Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

Socket Takeover (Proxygen restarts)

- Takeover TCP listening and UDP VIP sockets.
  - Old instance drains, updated instance handles new connections.

- Connection state?
  - TCP -> Preserved in kernel and old instance. ✓
  - UDP -> Application level QUIC state. ✗



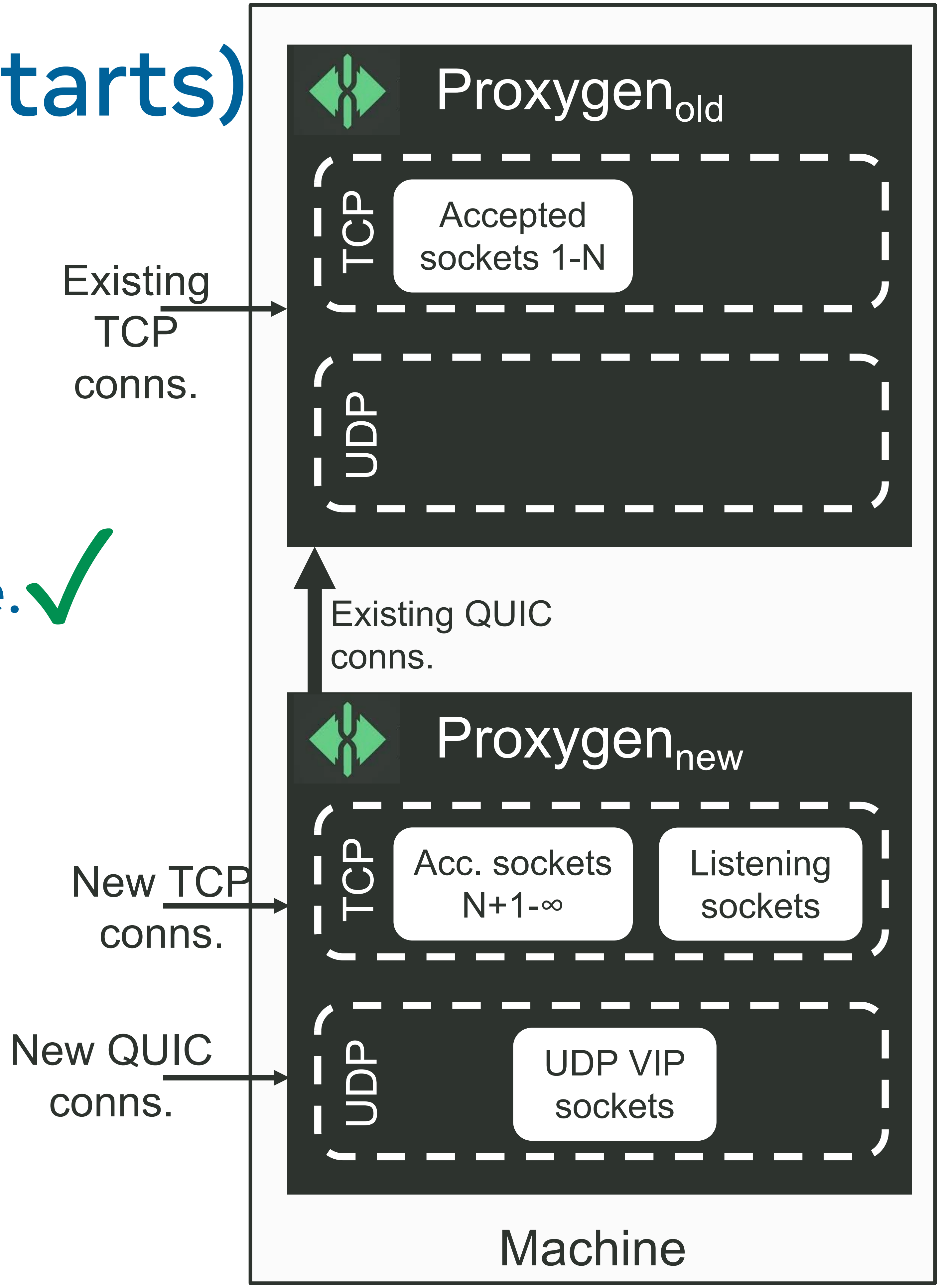
Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

Socket Takeover (Proxygen restarts)

- Takeover TCP listening and UDP VIP sockets.
  - Old instance drains, updated instance handles new connections.

- Connection state?
  - TCP -> Preserved in kernel and old instance. ✓
  - UDP -> Application level QUIC state. ✗

- User-space packet forwarding.
  - Coordination between Proxygens within machine.
  - QUIC "ConnectionID" based packet forwarding.



Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

Overall Socket-Takeover Mechanism

With support for UDP / QUIC

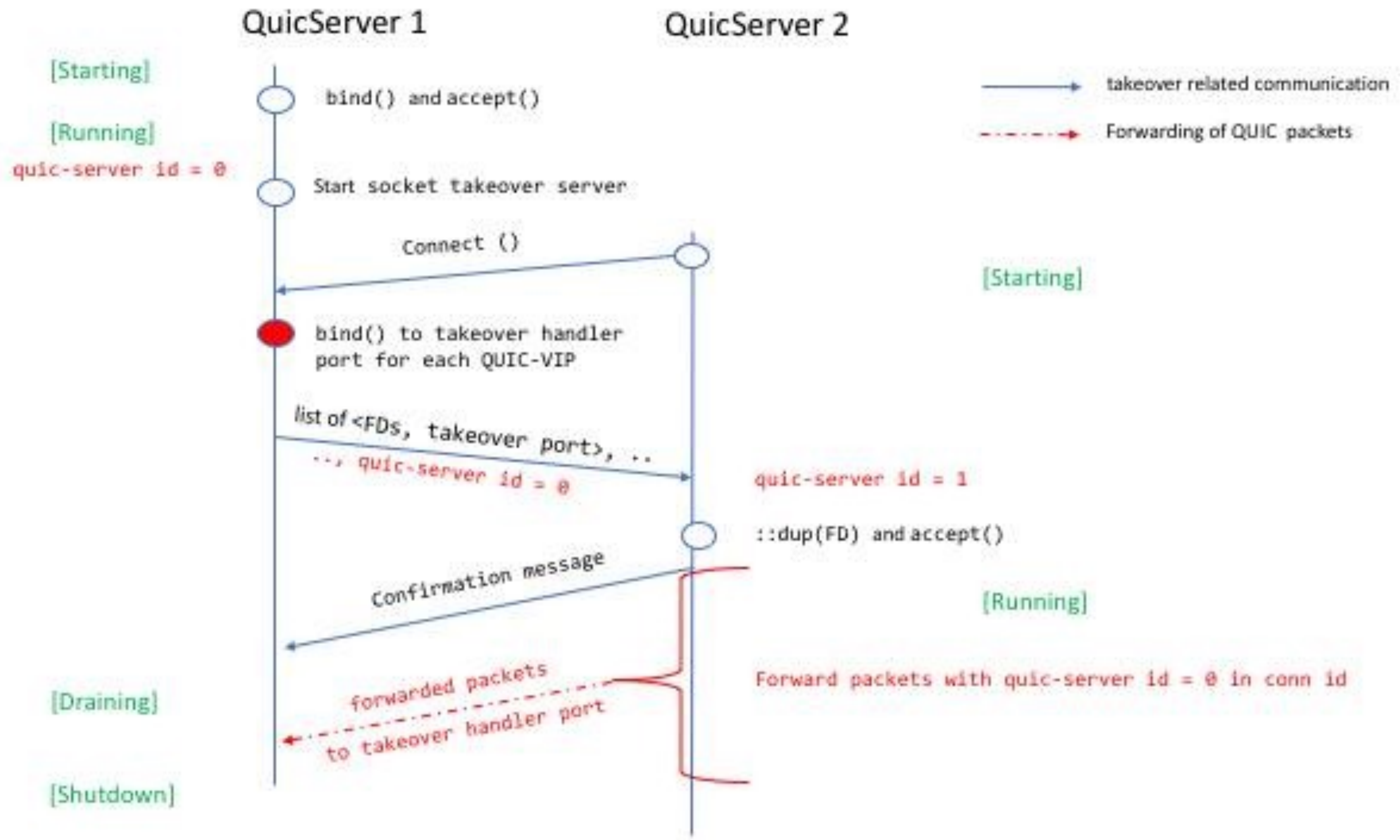


Figure: *Takeover* + packet forwarding mechanism for QUIC

Overall Socket-Takeover Mechanism

With support for UDP / QUIC

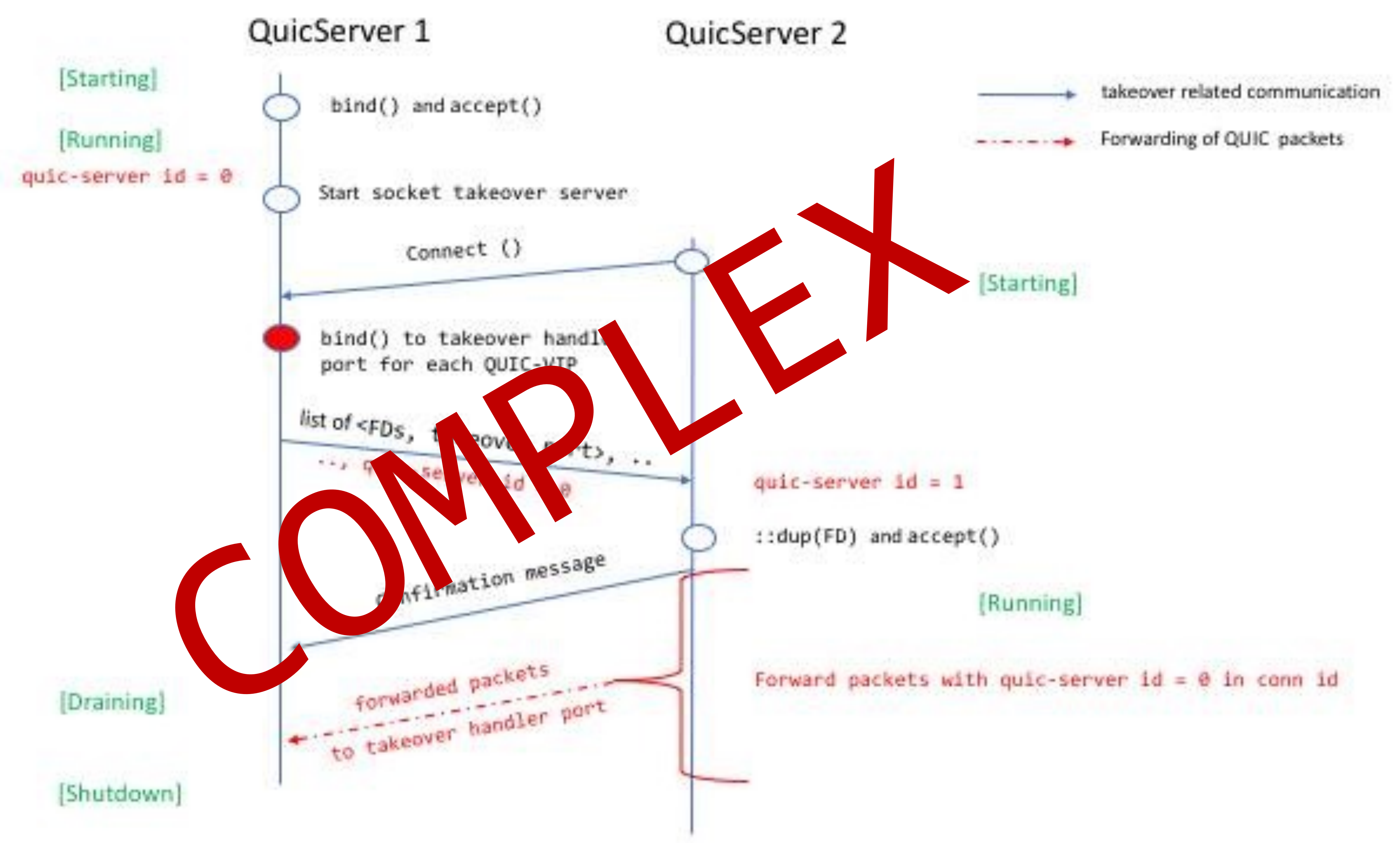
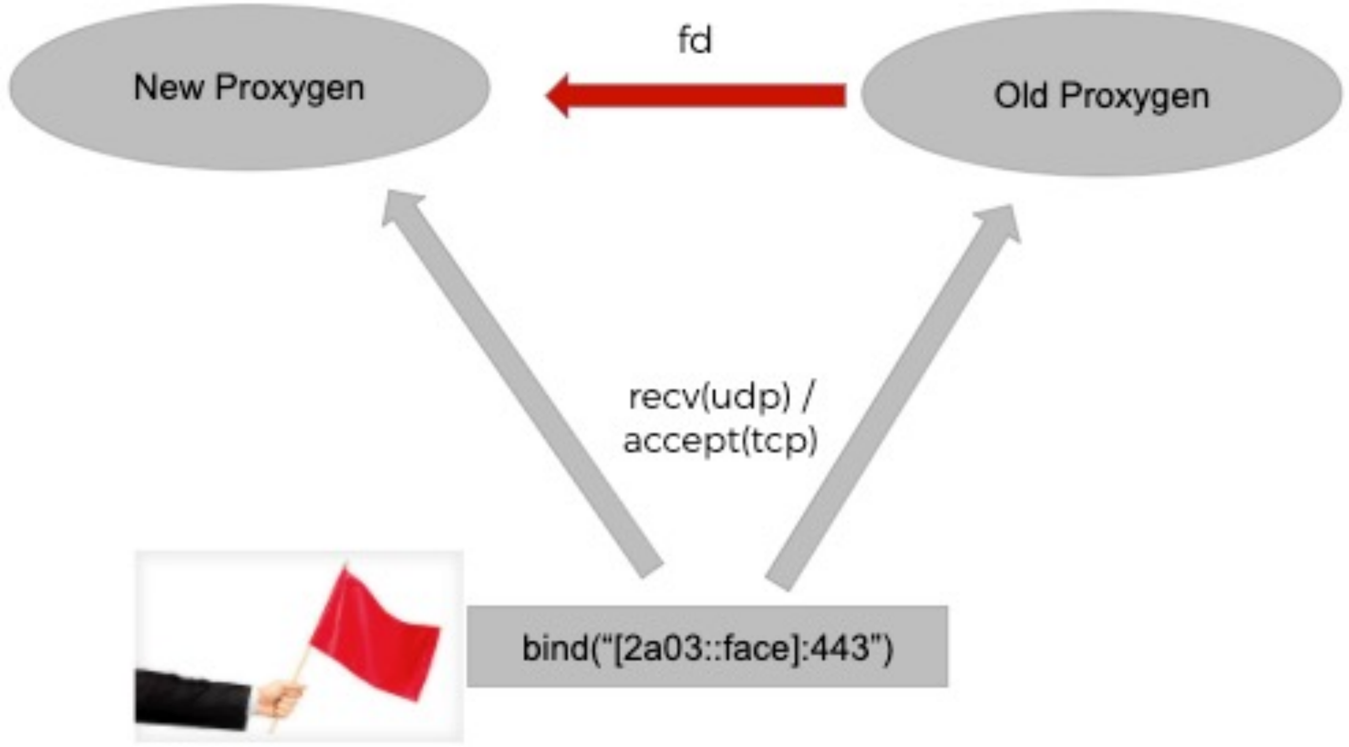


Figure: *Takeover* + packet forwarding mechanism for QUIC

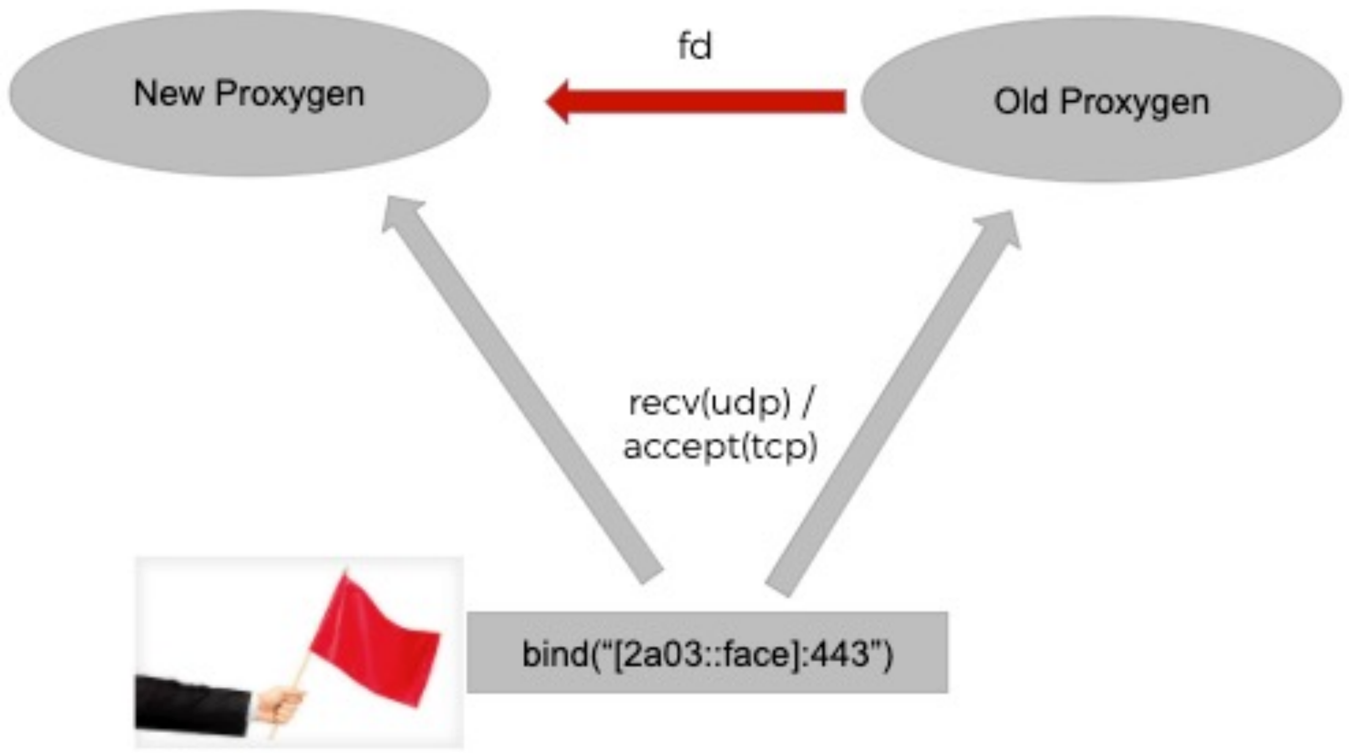
Issues with the existing Socket-Takeover

- Complex and Fragile process
  - A lot of interprocess communication (worse with UDP)
  - What if either of the process crashes?
  - Is this socket transferred?
  - Potential outages and vulnerabilities
- Root problem:
  - The sockets are shared between the old and the new process.



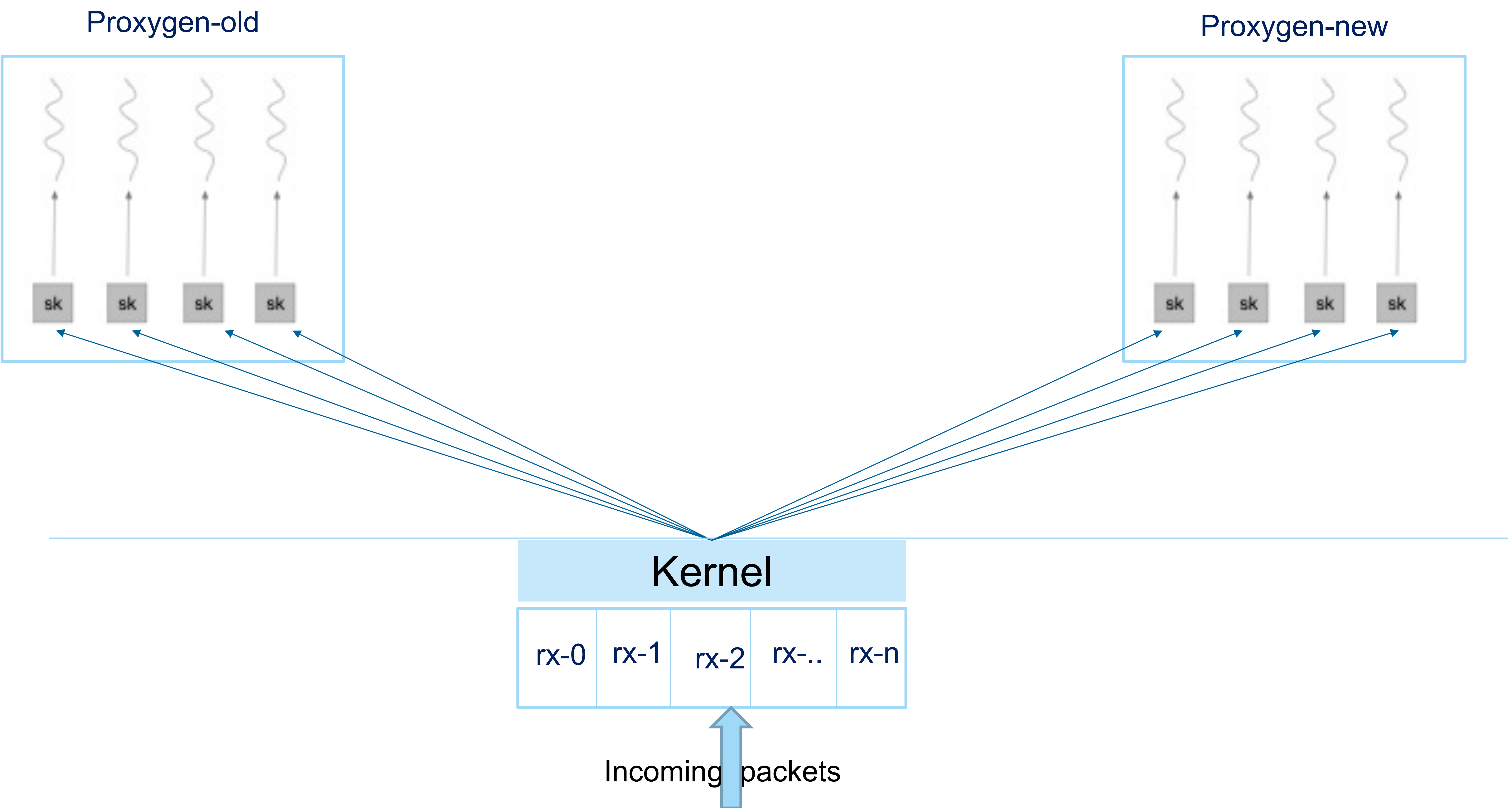
Issues with the existing Socket-Takeover

- Complex and Fragile process
  - A lot of interprocess communication (worse with UDP)
  - What if either of the process crashes?
  - Is this socket transferred?
  - Potential outages and vulnerabilities
- Root problem:
  - The sockets are shared between the old and the new process.
- 💡 **Can we avoid sharing the same sockets?**



Issues with the existing Socket-Takeover

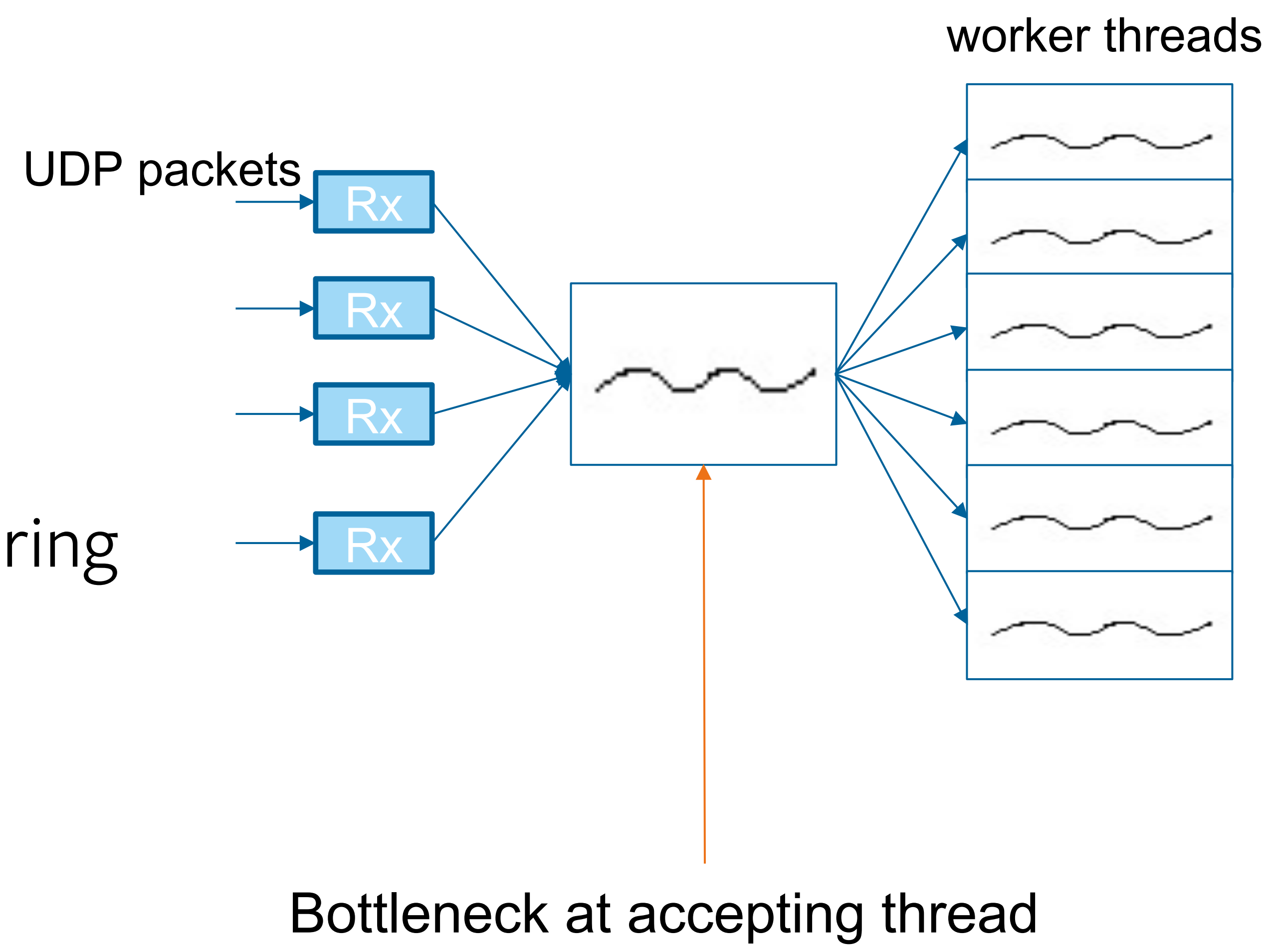
With SO_REUSEPORT: No consistent routing for UDP packets in a *connection* during restarts



Issues with the existing Socket-Takeover

Without SO_REUSEPORT: Single thread to multiplex _all_ UDP packets

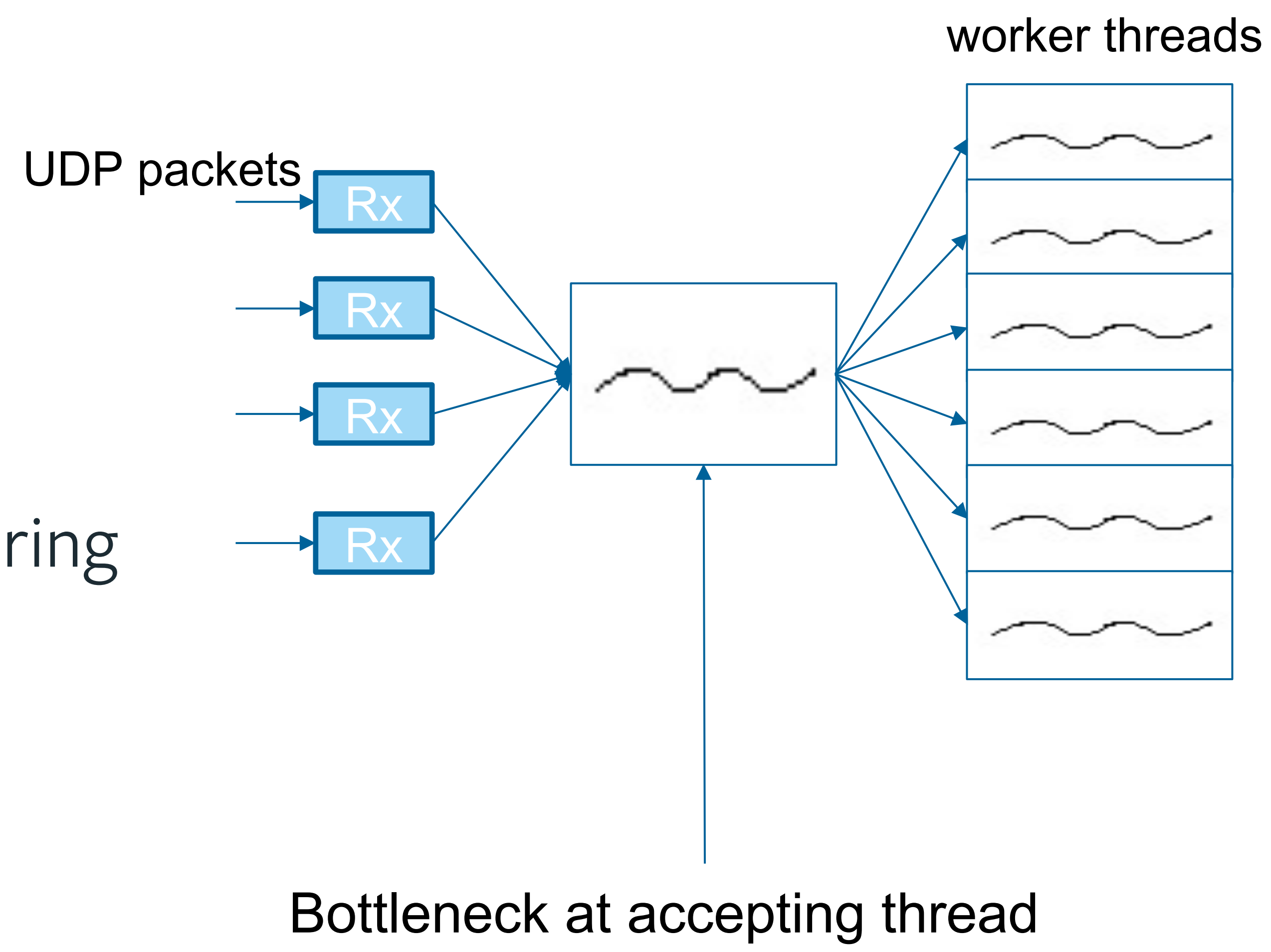
- Performance
  - Address scaling concerns – such as single threaded acceptor for UDP
- Root problem:
  - SO_REUSEPORT + UDP alone leads to lots of disruptions during proxygen restart.



Issues with the existing Socket-Takeover

Without SO_REUSEPORT: Single thread to multiplex _all_ UDP packets

- Performance
  - Address scaling concerns – such as single threaded acceptor for UDP
- Root problem:
  - SO_REUSEPORT + UDP alone leads to lots of disruptions during proxygen restart.
- 💡 How can we keep the inter-socket routing of UDP packets consistent?



Introducing SK-LB powered by SO_REUSEPORT_SOCKARRAY

- Taking a step back and thinking about a generic solution
- Attach a bpf program at socket level (`bpf_sk_reuseport`)

Introducing SK-LB powered by SO_REUSEPORT_SOCKARRAY

- Taking a step back and thinking about a generic solution
- Attach a bpf program at socket level (`bpf_sk_reuseport`)
    - Generically handle multiple protocols
    - Better control on the startup path for a new process on per vip level

Introducing SK-LB powered by SO_REUSEPORT_SOCKARRAY

- Taking a step back and thinking about a generic solution
- Attach a bpf program at socket level (`bpf_sk_reuseport`)
  - Generically handle multiple protocols
  - Better control on the startup path for a new process on per vip level
- Performance
  - Address scaling concerns – such as single threaded acceptor for UDP with SO_REUSEPORT

Introducing SK-LB powered by SO_REUSEPORT_SOCKARRAY

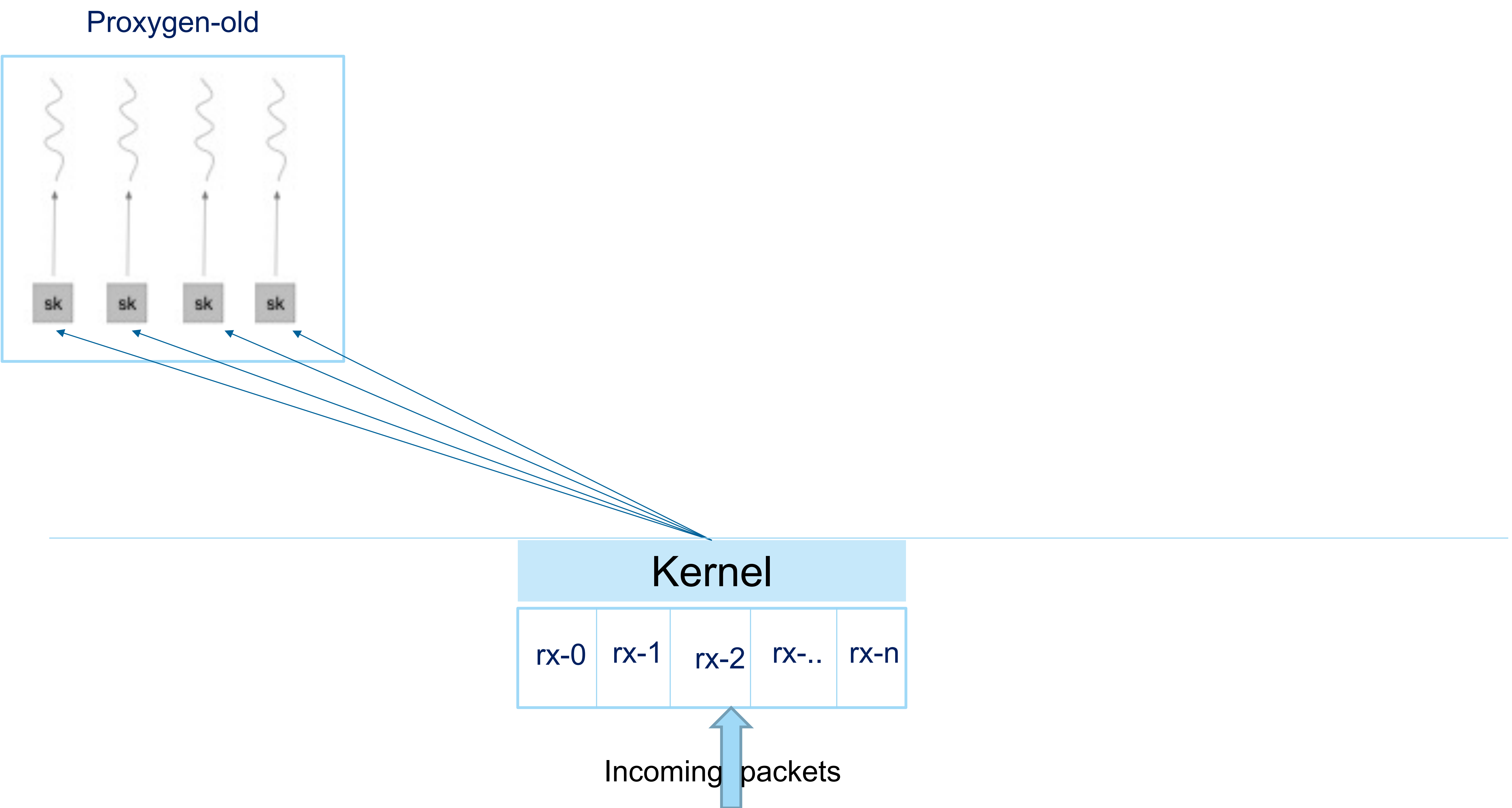
- Taking a step back and thinking about a generic solution
- Attach a bpf program at socket level (`bpf_sk_reuseport`)
  - Generically handle multiple protocols
  - Better control on the startup path for a new process on per vip level
- Performance
  - Address scaling concerns – such as single threaded acceptor for UDP with SO_REUSEPORT
- Routing control at packet level
  - Adjust weight of traffic per cpu

Introducing SK-LB powered by SO_REUSEPORT_SOCKARRAY

- Taking a step back and thinking about a generic solution
- Attach a bpf program at socket level (`bpf_sk_reuseport`)
    - Generically handle multiple protocols
    - Better control on the startup path for a new process on per vip level
- Performance
    - Address scaling concerns – such as single threaded acceptor for UDP with SO_REUSEPORT
- Routing control at packet level
    - Adjust weight of traffic per cpu
- Flexibility to iterate in future
    - Can keep each packet in same CPU, NUMA isolation?

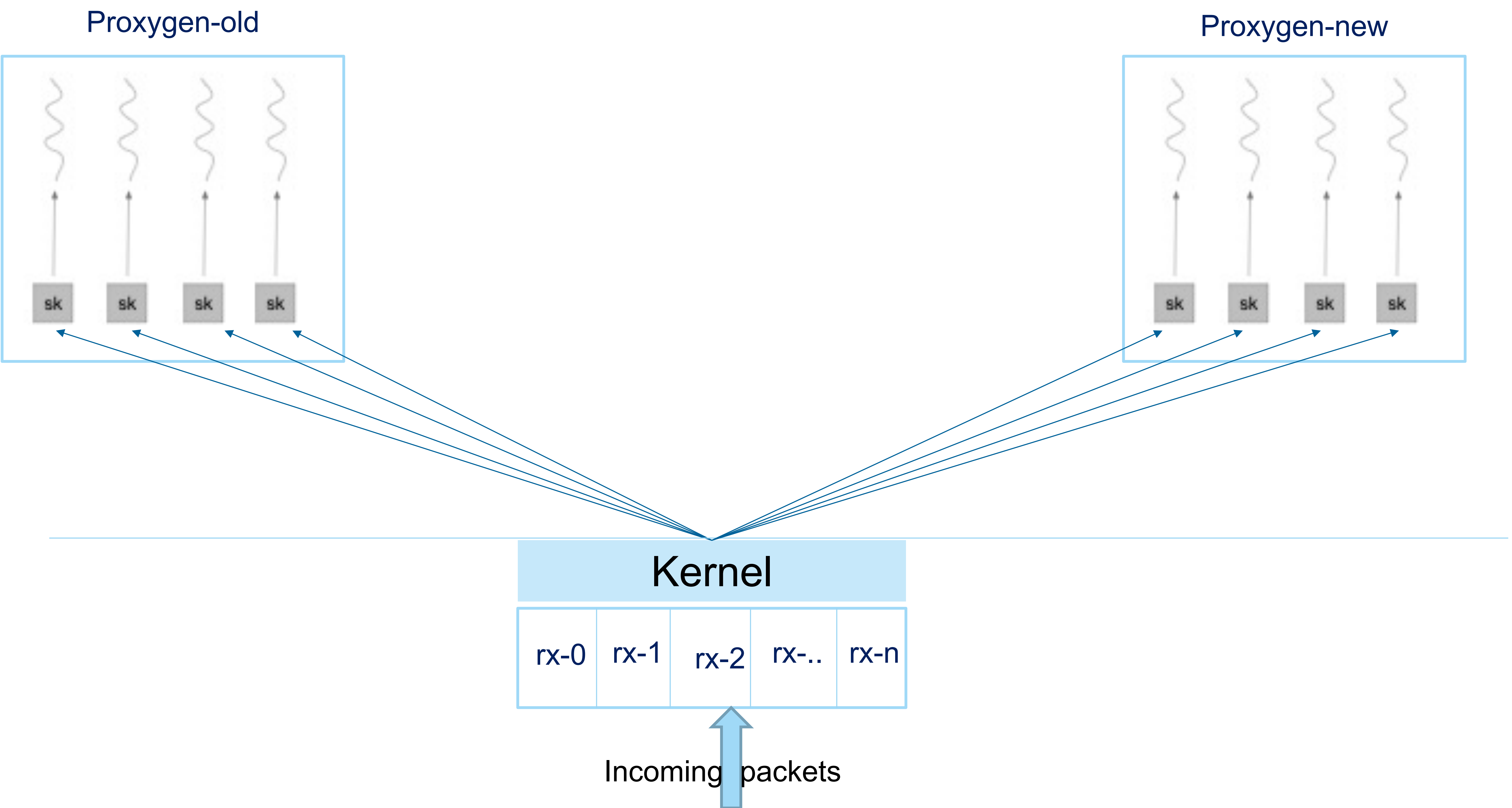
Introducing SK-LB powered by SO_REUSEPORT_SOCKARRAY

- Taking a step back and thinking about a generic solution



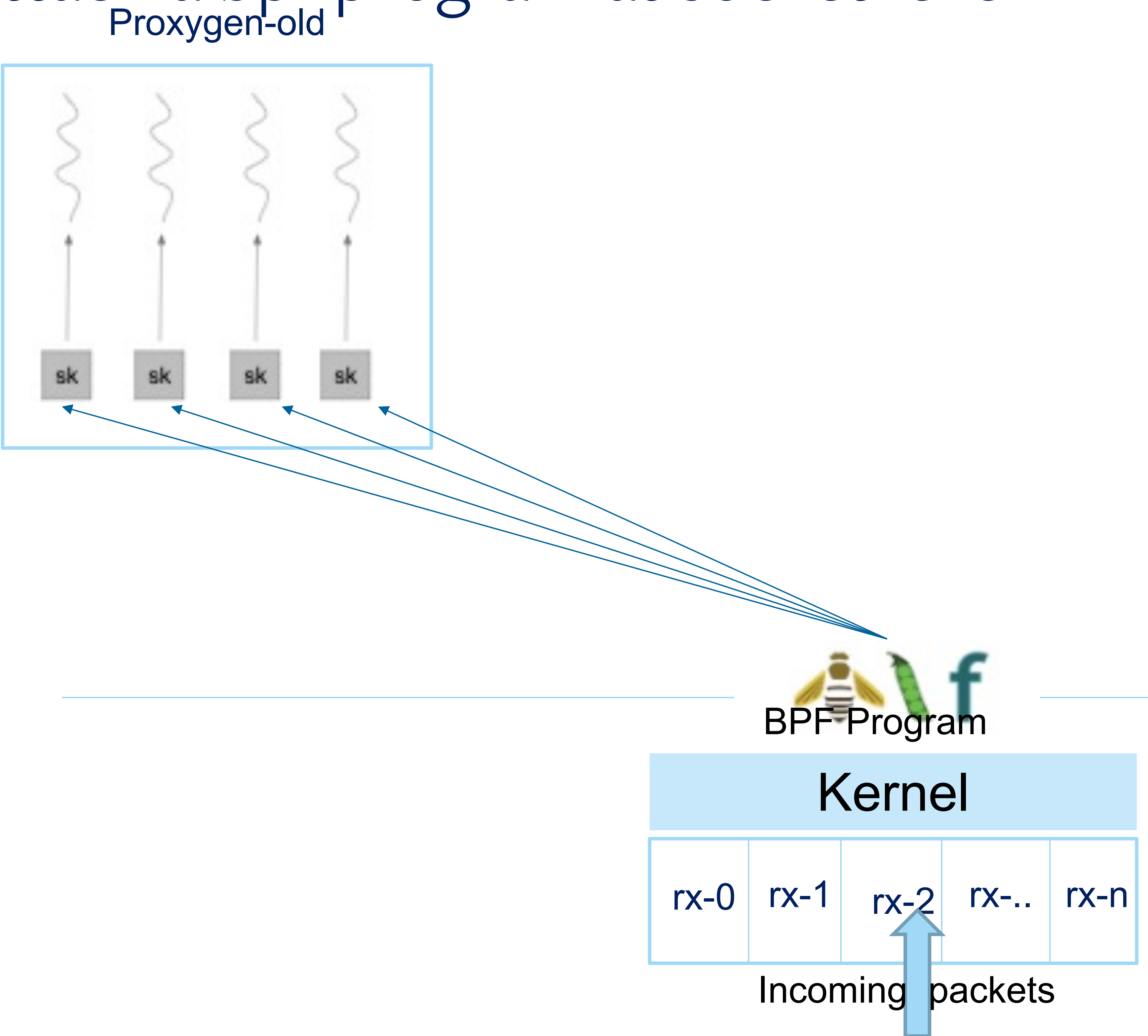
Introducing SK-LB powered by SO_REUSEPORT_SOCKARRAY

- Taking a step back and thinking about a generic solution



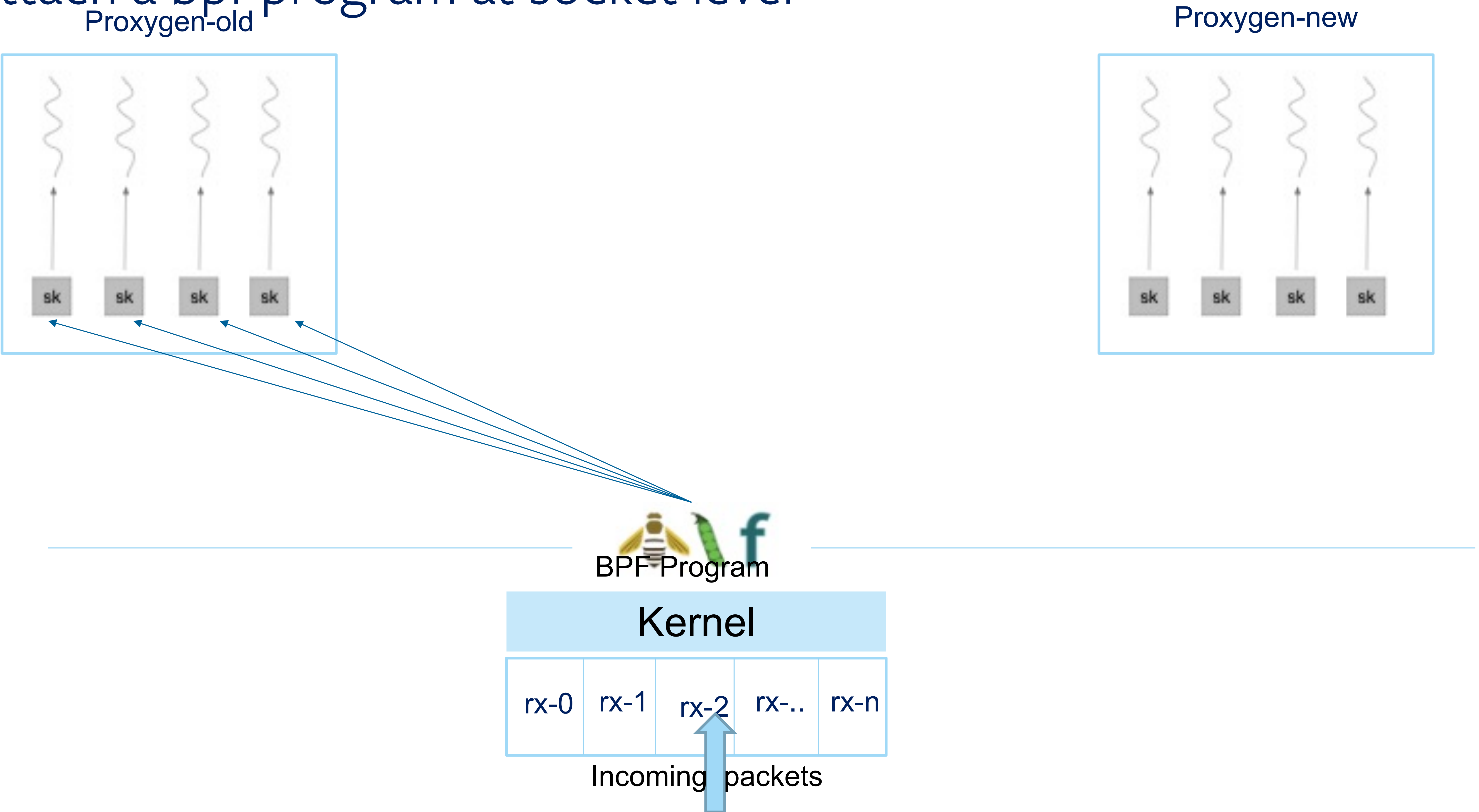
Introducing SK-LB powered by SO_REUSEPORT_SOCKARRAY

- Taking a step back and thinking about a generic solution
- Attach a bpf program at socket level



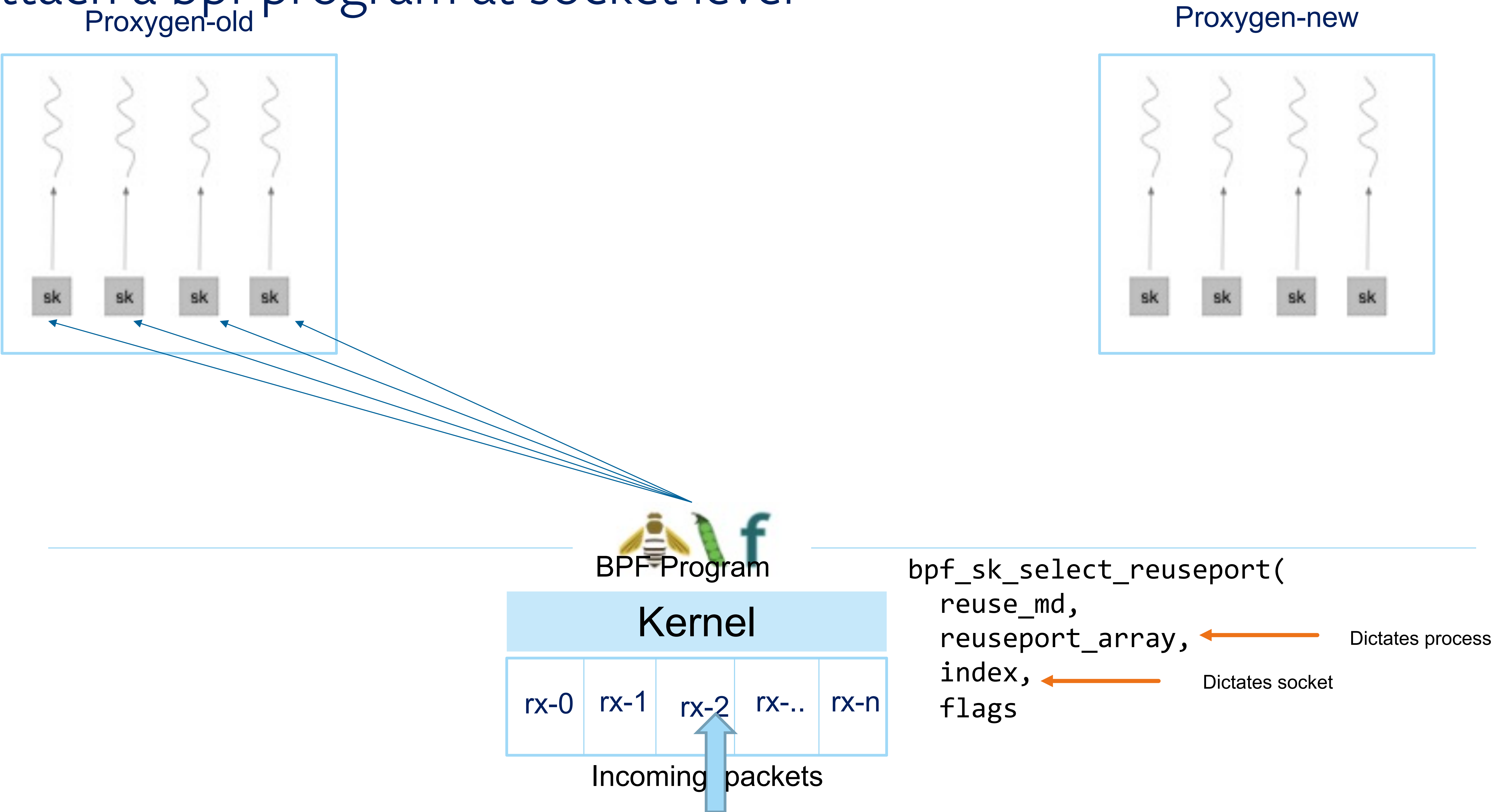
Introducing SK-LB powered by SO_REUSEPORT_SOCKARRAY

- Taking a step back and thinking about a generic solution
- Attach a bpf program at socket level



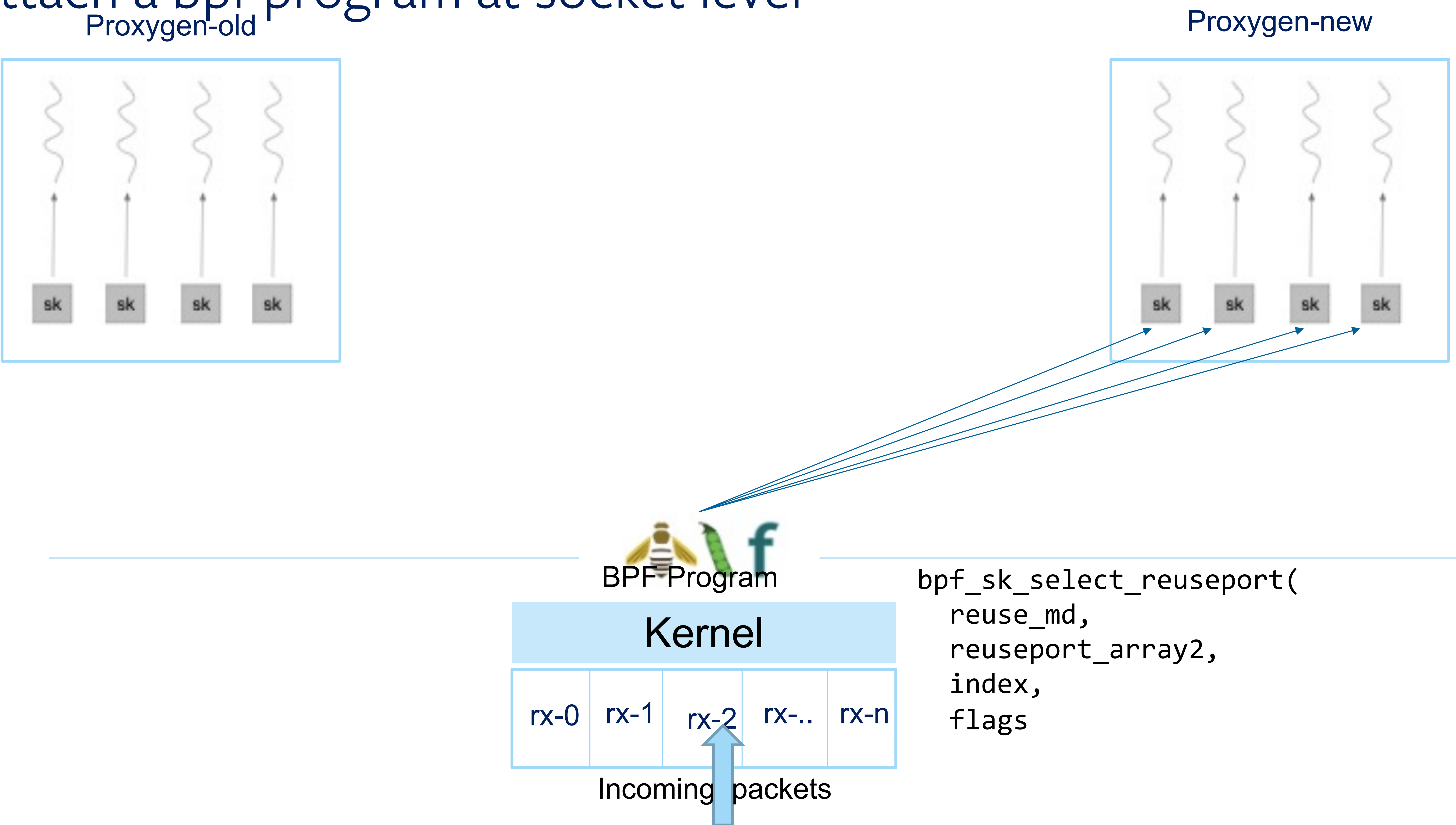
Introducing SK-LB powered by SO_REUSEPORT_SOCKARRAY

- Taking a step back and thinking about a generic solution
- Attach a bpf program at socket level



Introducing SK-LB powered by SO_REUSEPORT_SOCKARRAY

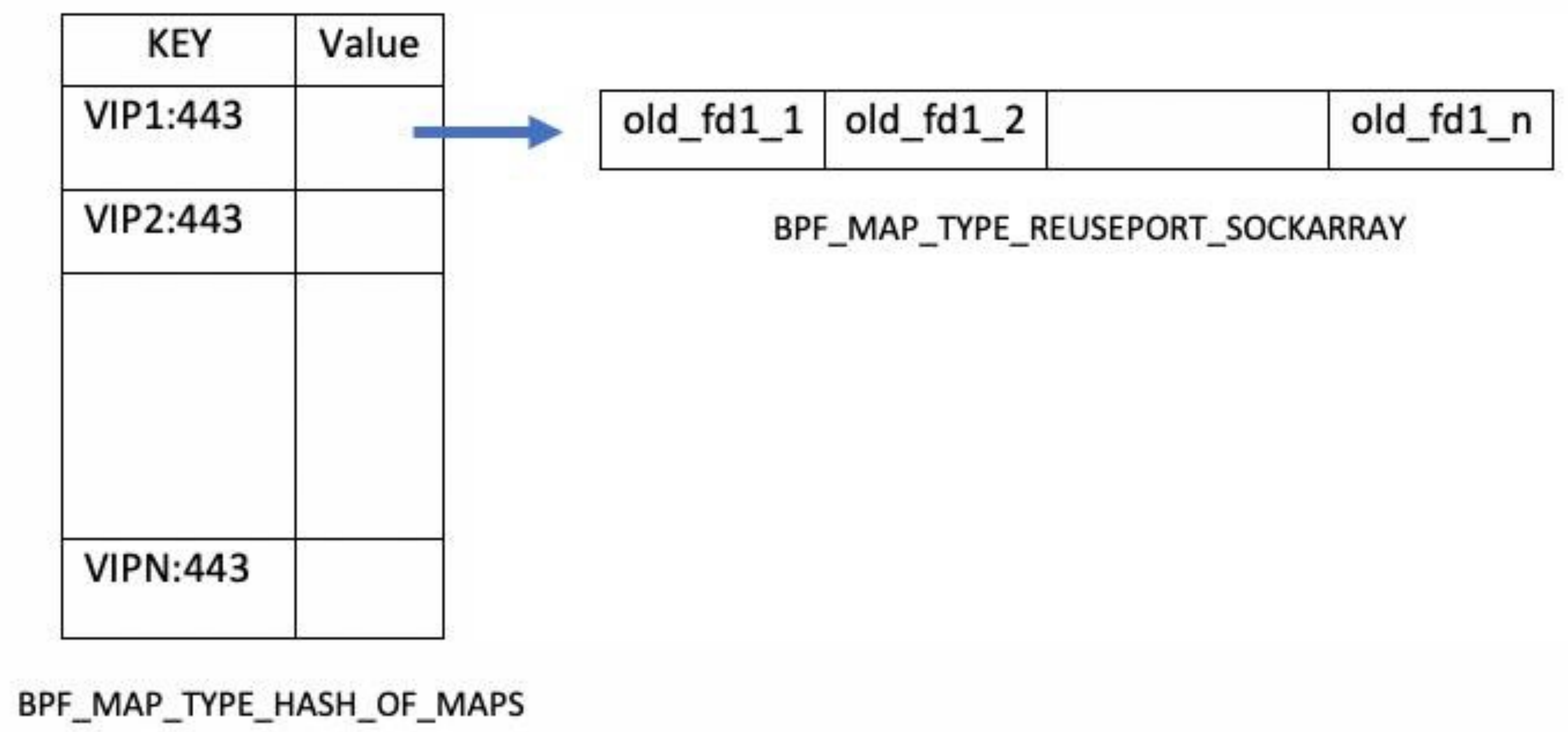
- Taking a step back and thinking about a generic solution
- Attach a bpf program at socket level




```
bpf_sk_select_reuseport(
  reuse_md,
  reuseport_array2,
  index,
  flags
```

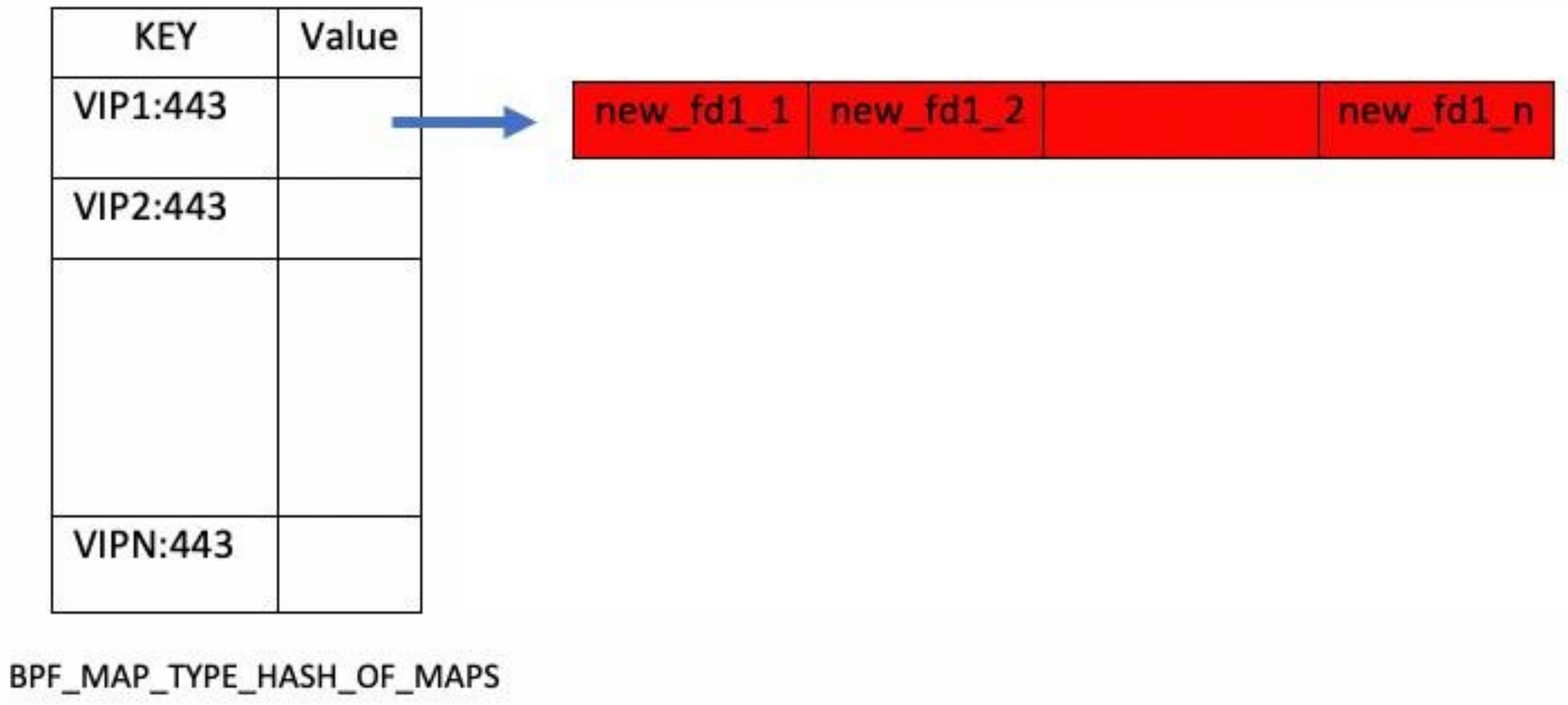
SK-LB powered by SO_REUSEPORT_SOCKARRAY

Better control on the startup path for a new process on per vip level



SK-LB powered by SO_REUSEPORT_SOCKARRAY

Better control on the startup path for a new process on per vip level

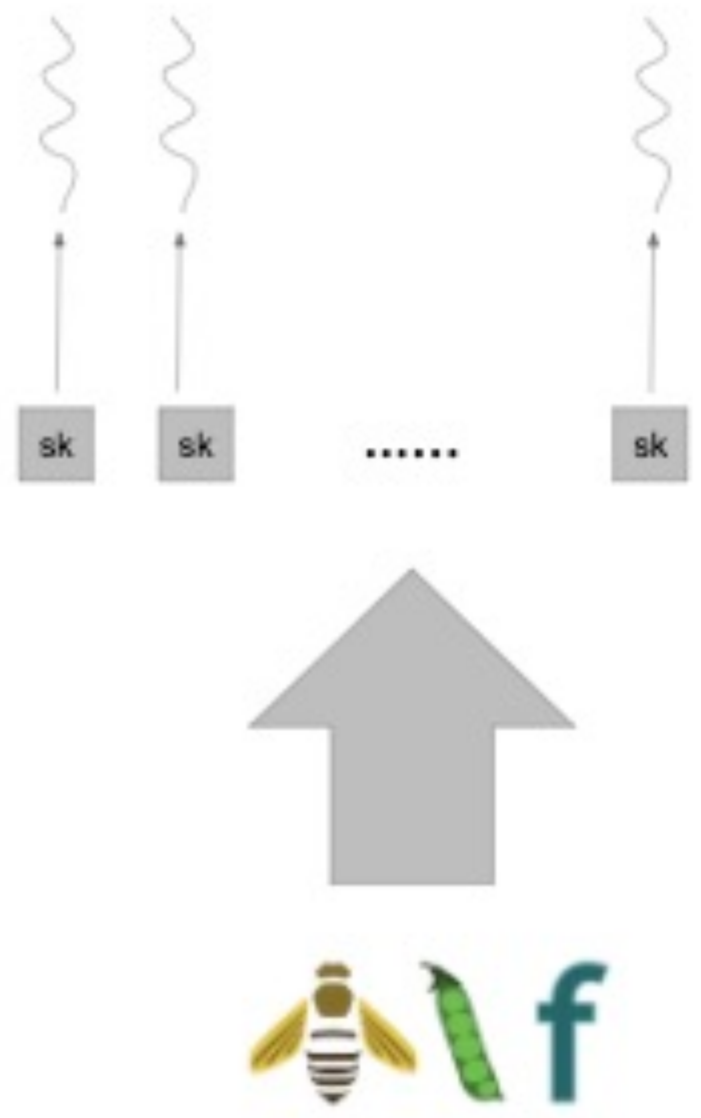


One socket per thread

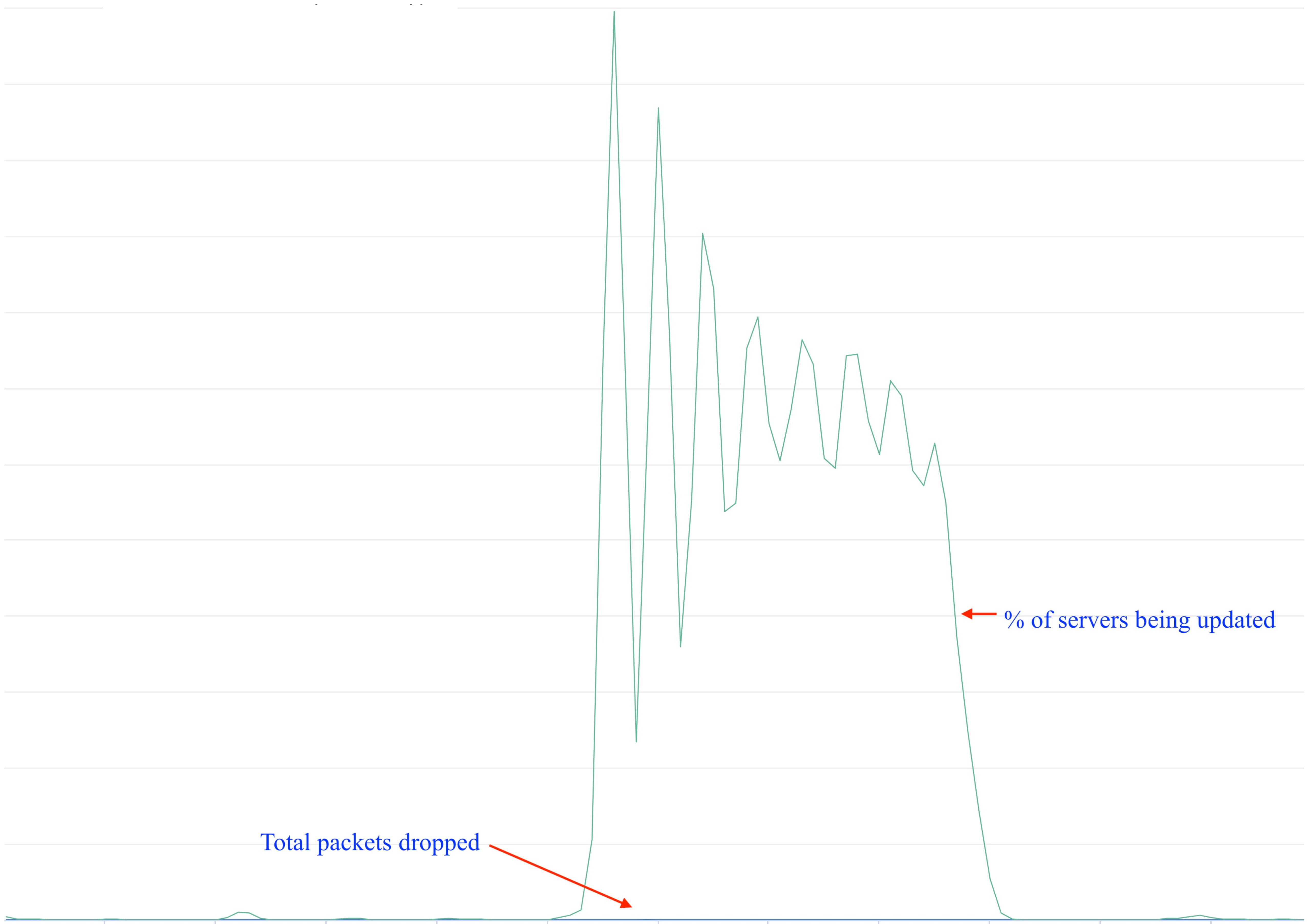
Address scaling concerns – such as single threaded acceptor for UDP

- Each thread bind its own socket to port
- No more sharing of sockets!

- Primary consideration is for UDP which does not have the concept of “one-new-connection => one-new-socket” like TCP

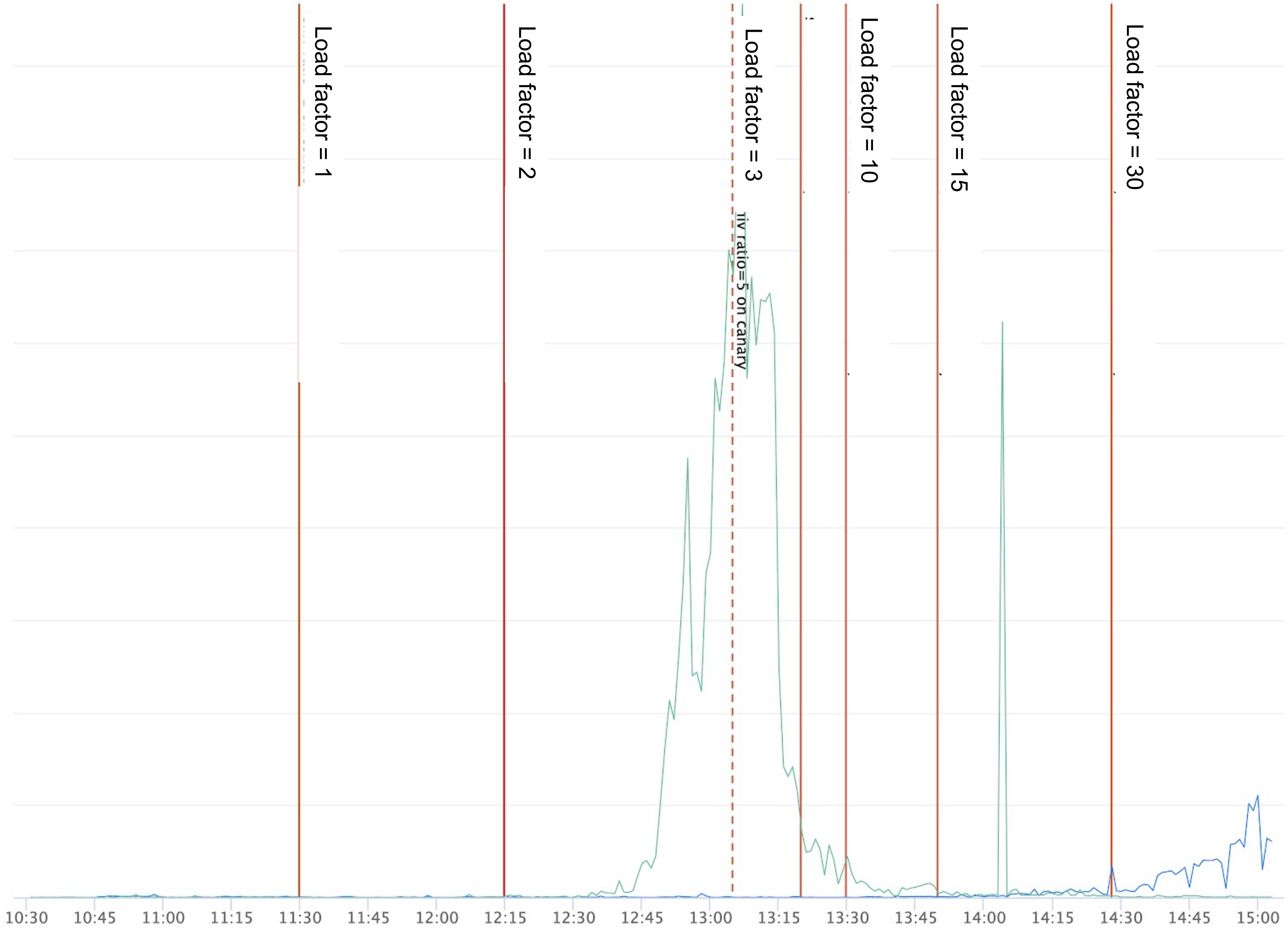


No packet drops during restarts



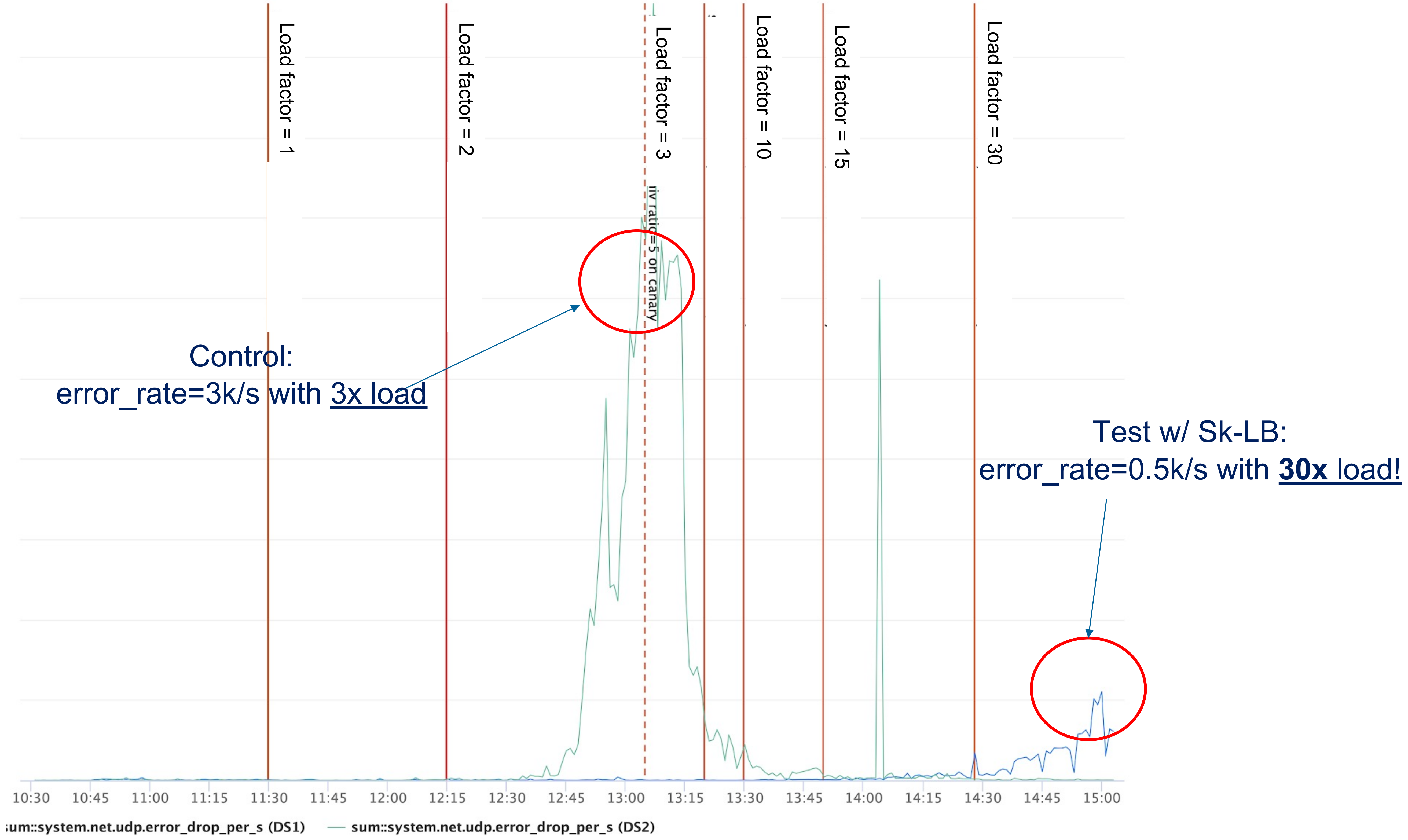
10x scaling of UDP packet processing ability

Control host hits limit at 3x traffic; test scales well to > 20x (until CPU saturates)



10x scaling of UDP packet processing ability

Control host hits limit at 3x traffic; test scales well to > 20x (until CPU saturates)



BPF to rescue

With `bpf_sk_reuseport` + `SO_REUSEPORT_SOCKARRAY`

Operational wins

simplified the overall process,
no IPC => less failures

Efficiency wins

10x more efficient for UDP load

Reliability wins

no packet drops due to misrouting of packets, or race during TCP 3WH

Experience deploying it

CPU spikes due to spin_lock in bind() path

- Issues in multi-tenant environment with large number of sockets in a *netns*
- Led to spikes in CPU and even host locking

Experience deploying it

CPU spikes due to spin_lock in bind() path

- Issues in multi-tenant environment with large number of sockets in a *netns*
- Led to spikes in CPU and even host locking
- `bind()` impl takes a spin lock to walk a long hashtable bucket with just port as key alone (where 443 and 80 are common ports)

/net/ipv4/inet_connection_sock.c

```
head = inet_csk_find_open_port(sk, &tb, &port);
if (!head)
  return ret;
…
head = &hinfo->bhash[inet_bhashfn(net, port,
hinfo->bhash_size)];

spin_lock_bh(&head->lock);

inet_bind_bucket_for_each(tb, &head->chain)
if (net_eq(ib_net(tb), net) && tb->l3mdev == l3mdev &&
tb->port == port)
  goto tb_found;

…
tb_found:
if (!hlist_empty(&tb->owners)) {
  if (sk->sk_reuse == SK_FORCE_REUSE)
    goto success;

  if ((tb->fastreuse > 0 && reuse) ||
    sk_reuseport_match(tb, sk))
      goto success;
  if (inet_csk_bind_conflict(sk, tb, true, true))
    goto fail_unlock;
}
```

Experience deploying it

CPU spikes due to spin_lock in bind() path

- Bug with caching of SO_REUSEPORT in the bind-address cache

/net/ipv4/inet_connection_sock.c

```
spin_lock_bh(&head->lock);
. . .

if ((tb->fastreuse > 0 && reuse) ||
sk_reuseport_match(tb, sk))
  goto success;
// ^^ returned true for ::

if (inet_csk_bind_conflict(sk,
tb, true, true))
  goto fail_unlock;
}
```

```
bind("[::1]:443");  /* without SO_REUSEPORT. Succeed. */
bind("[::2]:443");  /* with SO_REUSEPORT. Succeed. */
bind("[::]:443");   /* with SO_REUSEPORT. Still Succeed  */
```
[1]

[1] Bug fixed in https://lore.kernel.org/lkml/20200601174049.377204943@linuxfoundation.org/

Experience deploying it

CPU spikes due to spin_lock in bind() path

- Bug with caching of SO_REUSEPORT in the bind-address cache

```
bind("[::1]:443");  /* without SO_REUSEPORT. Succeed. */
bind("[::2]:443");  /* with SO_REUSEPORT. Succeed. */
bind("[::]:443");   /* with SO_REUSEPORT. Still Succeed  */
```

[1]

- Needed to ensure the cache was cleared
- Workaround with `bind(*:443)` with SO_REUSEPORT enabled

[1] Bug fixed in https://lore.kernel.org/lkml/20200601174049.377204943@linuxfoundation.org/

/net/ipv4/inet_connection_sock.c

```
spin_lock_bh(&head->lock);
. . .
if ((tb->fastreuse > 0 && reuse) ||
sk_reuseport_match(tb, sk))
  goto success;
// ^^ always true

if (inet_csk_bind_conflict(sk,
tb, true, true))
  goto fail_unlock;
}
```

bpf_sk_select_reuseport vs bpf_sk_lookup

- sk_lookup: also allows to pickup a TCP listening or unconnected UDP socket
- https://lwn.net/Articles/825103/

- Overlap in some of the motivations

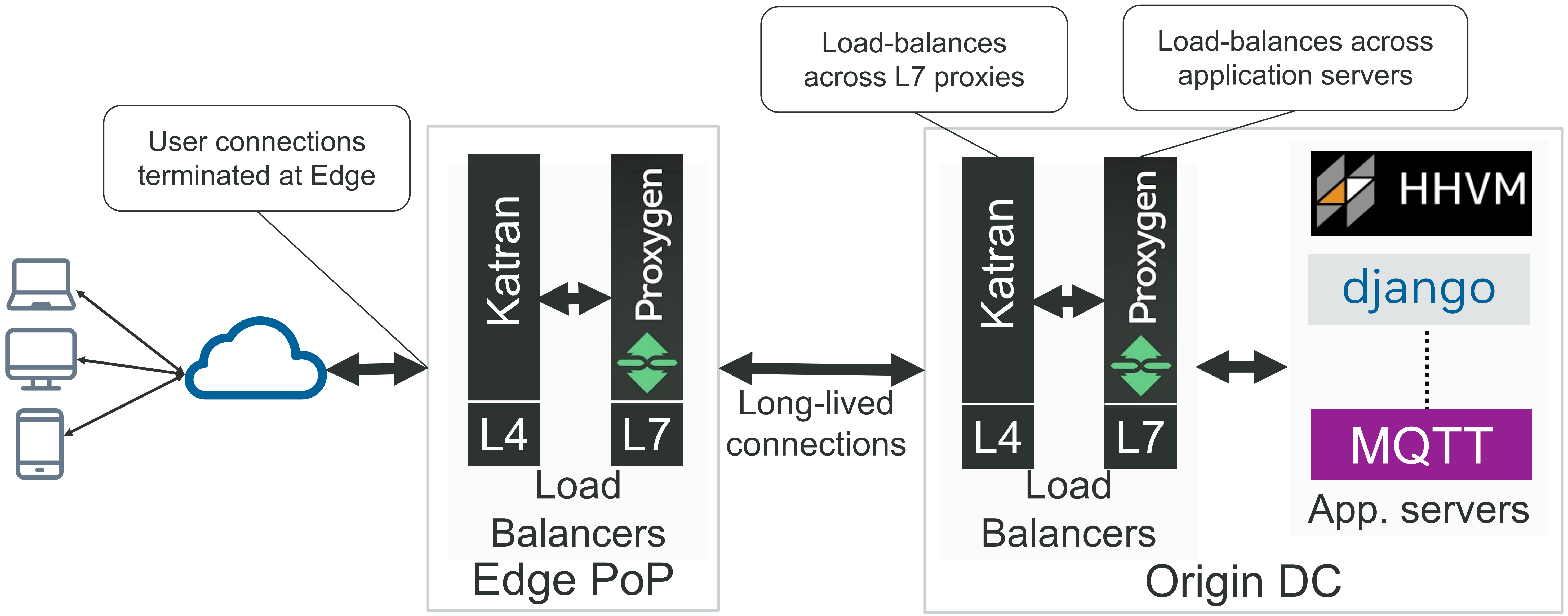
- sk_select_reuseport IS associated with the address for the socket family

- sk_lookup on the other hand decouples IP from Socket – lets it pick *any* / netns

Marwan Fayed, Lorenz Bauer, Vasileios Giotsas, Sami Kerola, Marek Majkowski, Pavel Odintsov, Jakub Sitnicki, Taejoong Chung, Dave Levin, Alan Mislove, Christopher A. Wood, and Nick Sullivan. 2021. The ties that un-bind: decoupling IP from web services and sockets for robust addressing agility at CDN-scale. In Proceedings of the 2021 ACM SIGCOMM 2021 Conference (SIGCOMM '21). Association for Computing Machinery, New York, NY, USA, 433–446.

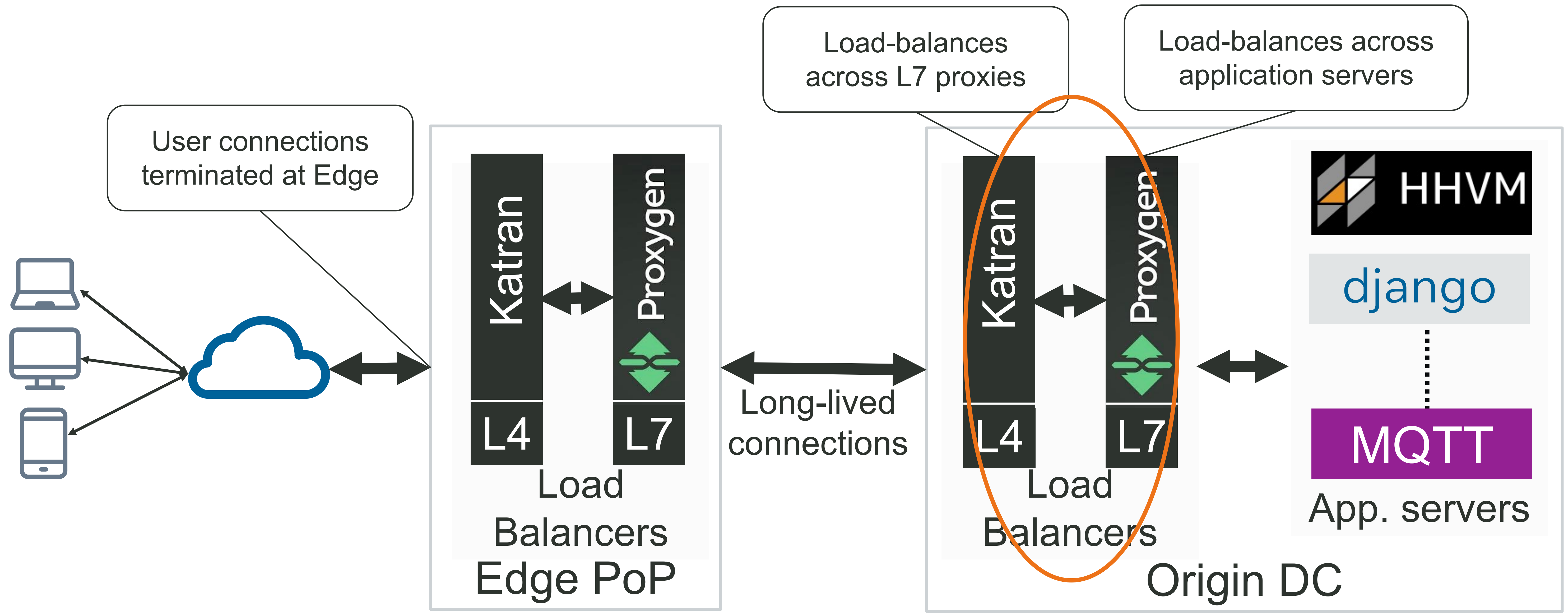
Part II: Stateless routing of TCP packets from XDP to L7 applications

Traffic Infrastructure @ FB



Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

Traffic Infrastructure @ FB



Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R., & Benson, T. A. (2020, July). Zero downtime release: Disruption-free load balancing of a multi-billion user website. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (pp. 529-541).

Routing mechanism within Katran (L4 LB)

Powered by Consistent Hashing

- Employs a variation of the Maglev Hash for Consistent Hashing

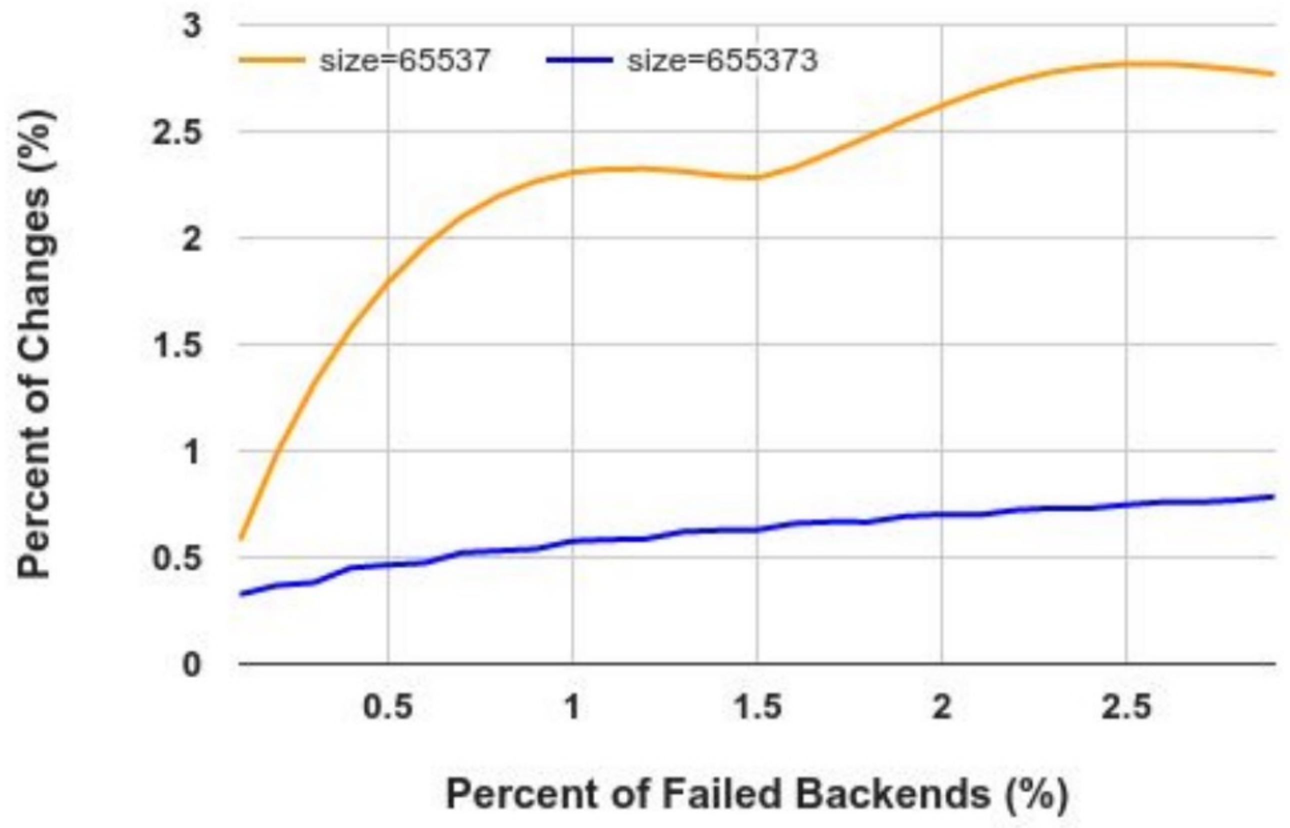
- Locally tracks connections for resiliency against backend server changes

```
int pick_host(packet* pkt)
  if (is_in_local_cache(pkt))
      return local_cache[pkt]
  return consistent_hash(pkt) % server_ring
```

- Highly effective and efficient

Limitations of Consistent Hashing

Tradeoffs between reliability and complexity



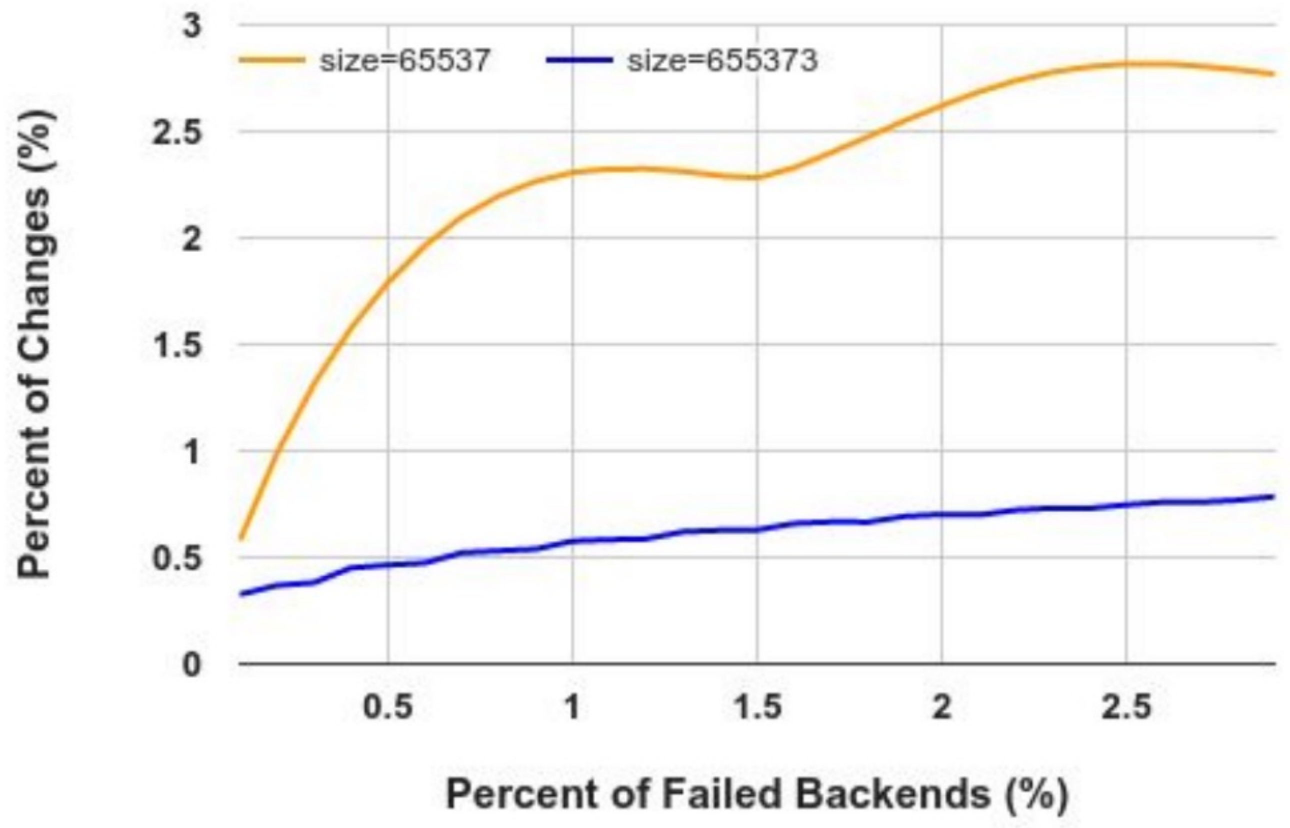
(From the Maglev paper [1])

[1] Daniel E Eisenbud, Cheng Yi, Carlo Contavalli, Cody Smith, Roman Kononov, Eric Mann-Hielscher, Ardas Cilingiroglu, Bin Cheyney, Wentao Shang, and Jinnah Dylan Hosein. 2016. Maglev: A fast and reliable software network load balancer. In 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16). 523–535.

Limitations of Consistent Hashing

Tradeoffs between reliability and complexity

- *Highly* effective != 100% effective
    - For long-lived TCP connections, e.g. videos



(From the Maglev paper [1])

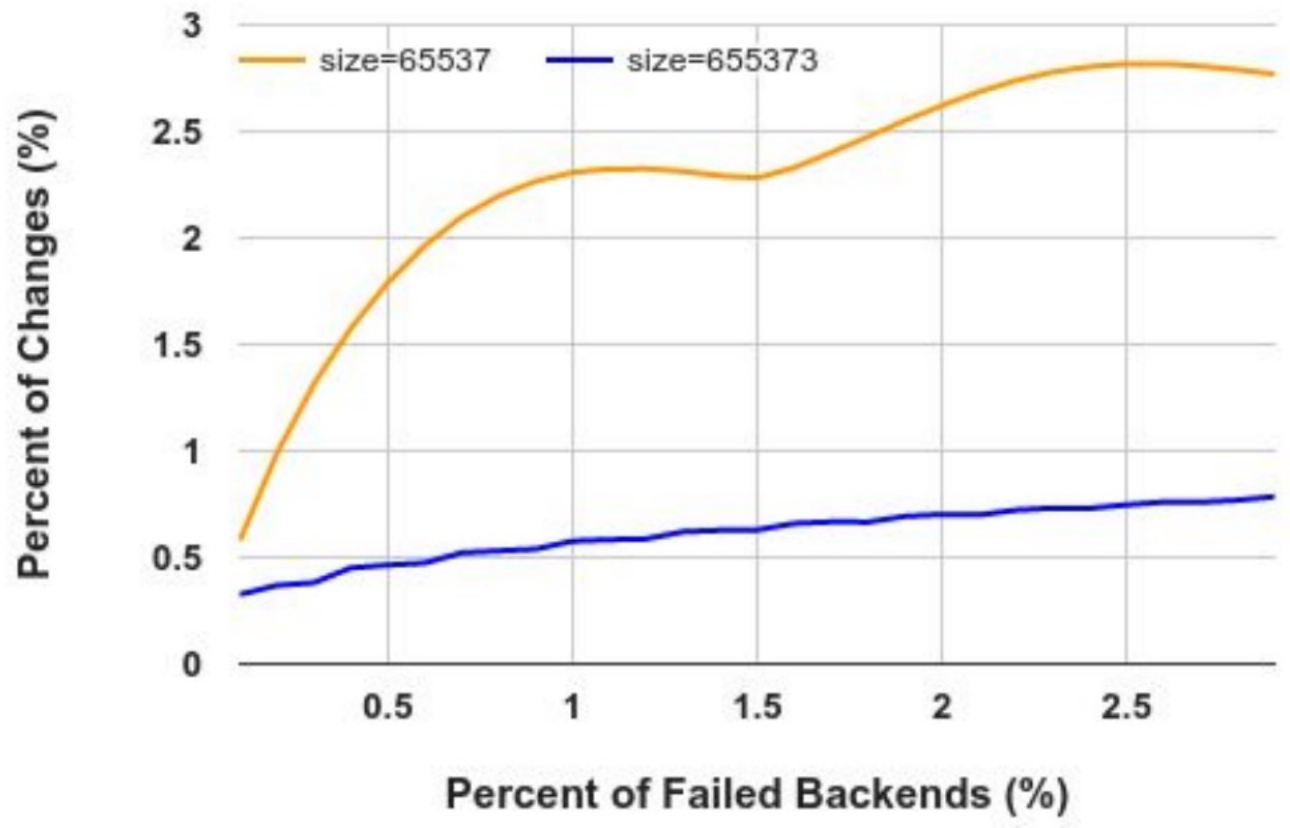
[1] Daniel E Eisenbud, Cheng Yi, Carlo Contavalli, Cody Smith, Roman Kononov, Eric Mann-Hielscher, Ardas Cilingiroglu, Bin Cheyney, Wentao Shang, and Jinnah Dylan Hosein. 2016. Maglev: A fast and reliable software network load balancer. In 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16). 523–535.

Limitations of Consistent Hashing

Tradeoffs between reliability and complexity

- *Highly* effective != 100% effective
    - For long-lived TCP connections, e.g. videos

```
if (is_in_local_cache(pkt))
      return local_cache[pkt]
return consistent_hash(pkt) % server_ring
```



(From the Maglev paper [1])

[1] Daniel E Eisenbud, Cheng Yi, Carlo Contavalli, Cody Smith, Roman Kononov, Eric Mann-Hielscher, Ardas Cilingiroglu, Bin Cheyney, Wentao Shang, and Jinnah Dylan Hosein. 2016. Maglev: A fast and reliable software network load balancer. In 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16). 523–535.

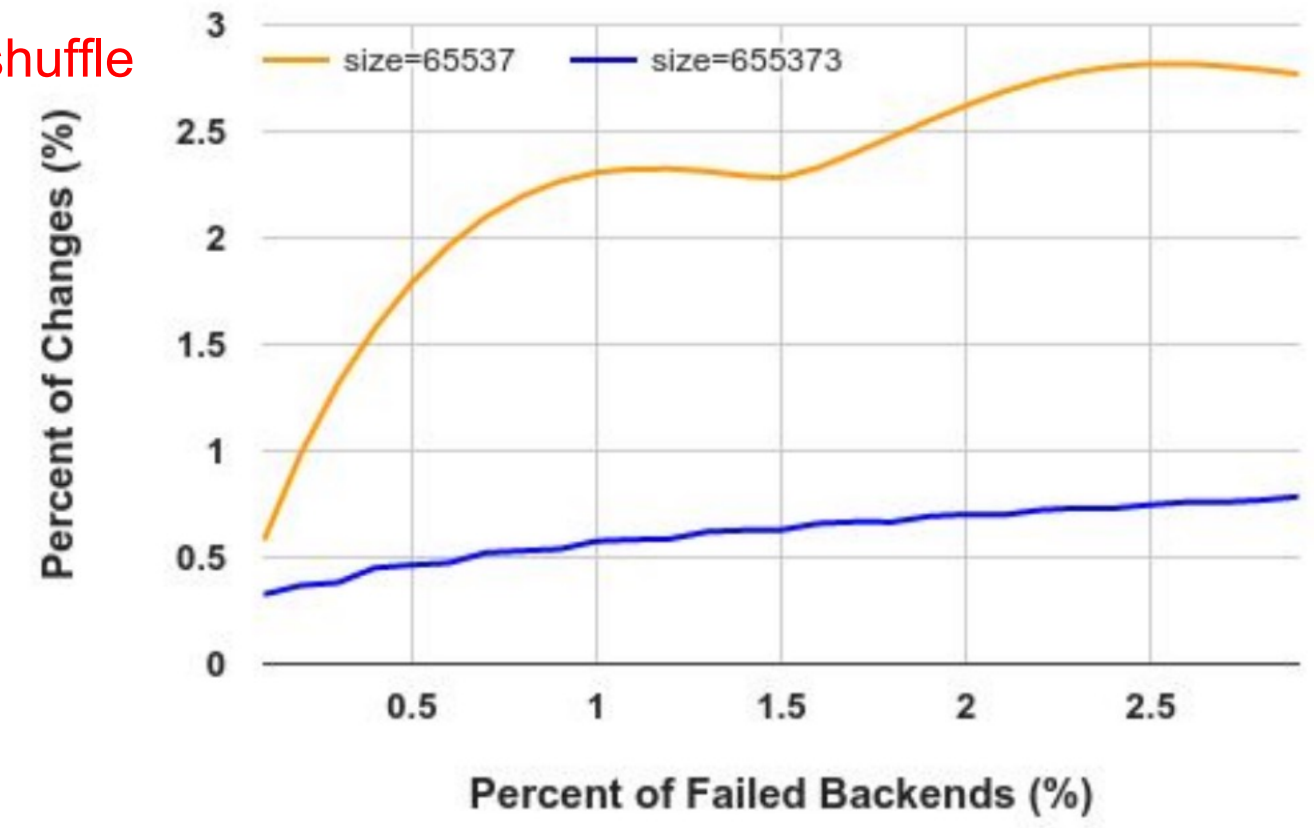
Limitations of Consistent Hashing

Tradeoffs between reliability and complexity

- *Highly* effective != 100% effective
  - For long-lived TCP connections, e.g. videos

```
if (is_in_local_cache(pkt))
      return local_cache[pkt]
return consistent_hash(pkt) % server_ring
```

Miss on ECMP shuffle



(From the Maglev paper [1])

[1] Daniel E Eisenbud, Cheng Yi, Carlo Contavalli, Cody Smith, Roman Kononov, Eric Mann-Hielscher, Ardas Cilingiroglu, Bin Cheyney, Wentao Shang, and Jinnah Dylan Hosein. 2016. Maglev: A fast and reliable software network load balancer. In 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16). 523–535.

Limitations of Consistent Hashing

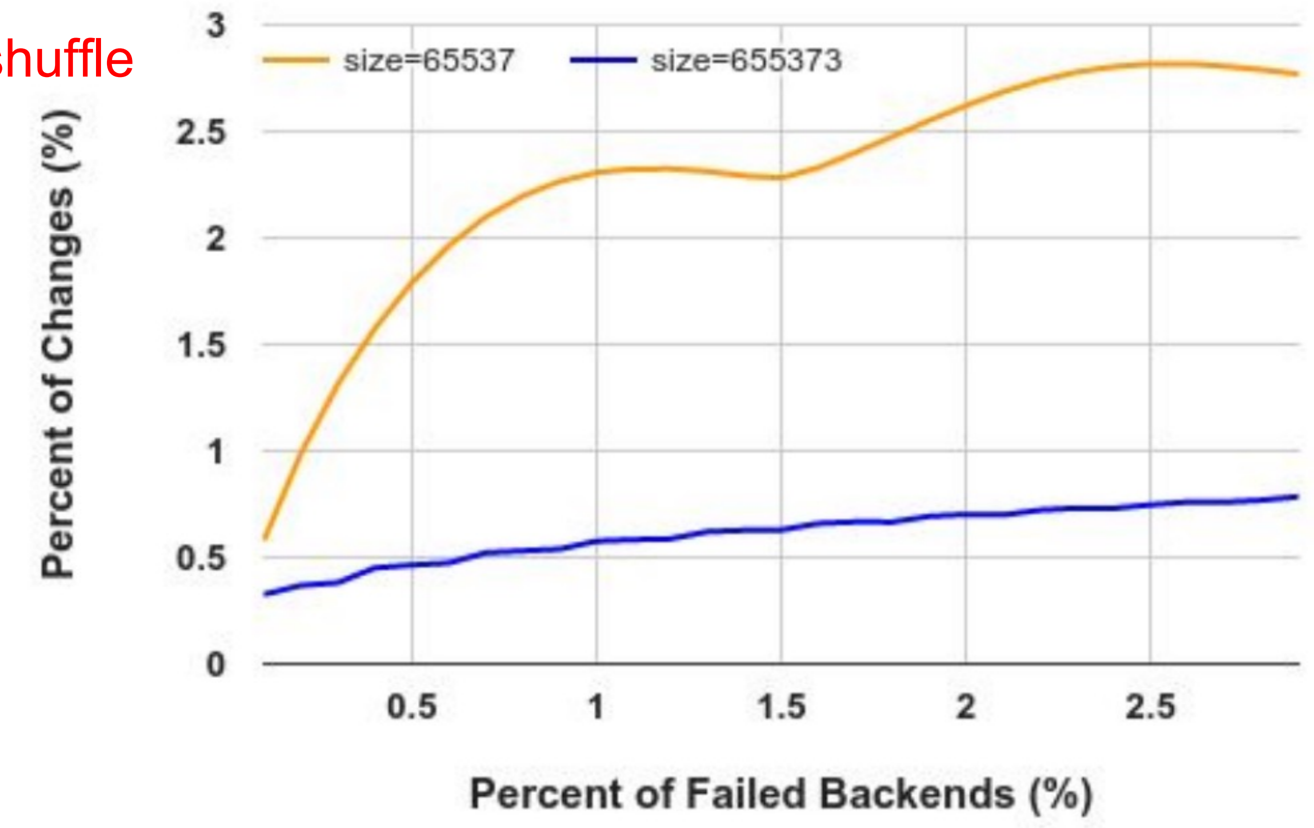
Tradeoffs between reliability and complexity

- *Highly* effective != 100% effective
    - For long-lived TCP connections, e.g. videos

```
if (is_in_local_cache(pkt))
      return local_cache[pkt]
return consistent_hash(pkt) % server_ring
```

Miss on ECMP shuffle

Small chance of different server if `server_ring` changes



(From the Maglev paper [1])

[1] Daniel E Eisenbud, Cheng Yi, Carlo Contavalli, Cody Smith, Roman Kononov, Eric Mann-Hielscher, Ardas Cilingiroglu, Bin Cheyney, Wentao Shang, and Jinnah Dylan Hosein. 2016. Maglev: A fast and reliable software network load balancer. In 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16). 523–535.

Limitations of Consistent Hashing

Tradeoffs between reliability and complexity

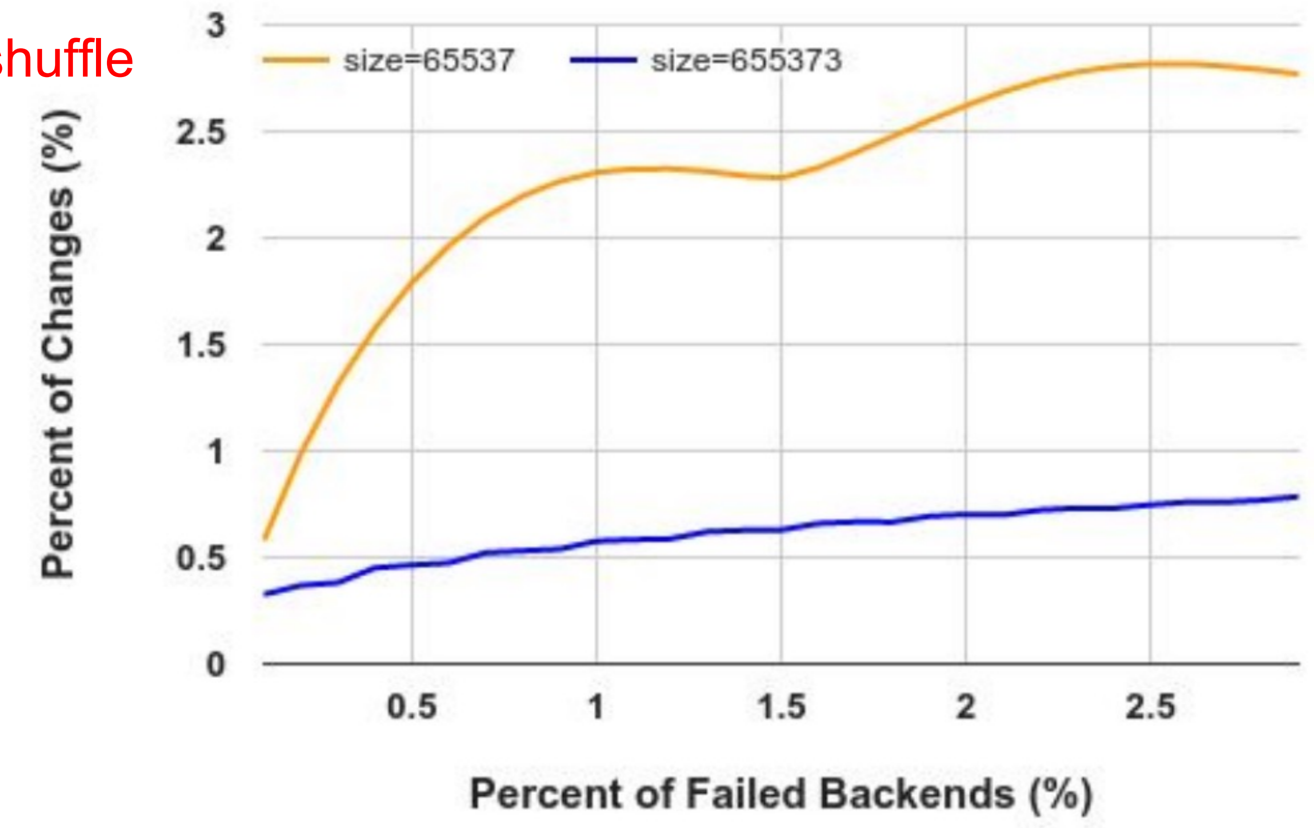
- *Highly* effective != 100% effective
  - For long-lived TCP connections, e.g. videos

```
if (is_in_local_cache(pkt))
     return local_cache[pkt]
return consistent_hash(pkt) % server_ring
```

Miss on ECMP shuffle

Small chance of different server if `server_ring` changes

- “Continuous release” of L4 and L7 hurts overall reliability ☹
- Sharing connection states across hosts adds complexity



(From the Maglev paper [1])

[1] Daniel E Eisenbud, Cheng Yi, Carlo Contavalli, Cody Smith, Roman Kononov, Eric Mann-Hielscher, Ardas Cilingiroglu, Bin Cheyney, Wentao Shang, and Jinnah Dylan Hosein. 2016. Maglev: A fast and reliable software network load balancer. In 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16). 523–535.

Not an issue for QUIC

Embed routing info in the packet

- QUIC specification [RFC 9000] allows servers to choose arbitrary `connection_id`
    - Servers can embed routing info in the `connection_id`
    - Clients MUST echo it back
    - Enables completely stateless routing in L4

Not an issue for QUIC

Embed routing info in the packet

- QUIC specification [RFC 9000] allows servers to choose arbitrary `connection_id`
    - Servers can embed routing info in the `connection_id`
    - Clients MUST echo it back
    - Enables completely stateless routing in L4

- 💡 **What if we could do the same for TCP?**

Stateless routing of TCP packets

Use BPF TCP header options

[PATCH v3 bpf-next 0/9] BPF TCP header options

- *Subject*: [PATCH v3 bpf-next 0/9] BPF TCP header options
- *From*: Martin KaFai Lau <kafai@xxxxxx>
- *Date*: Thu, 30 Jul 2020 13:56:57 -0700
- *Cc*: Alexei Starovoitov <ast@xxxxxxxxxx>, Daniel Borkmann <daniel@xxxxxxxxxxxxx>, Eric Dumazet <netdev@xxxxxxxxxxxxxxx>, Yuchung Cheng <ycheng@xxxxxxxxxx>
- *Smtp-origin-cluster*: ftw2c04
- *Smtp-origin-hostname*: devbig005.ftw2.facebook.com
- *Smtp-origin-hostprefix*: devbig

```
The earlier effort in BPF-TCP-CC allows the TCP Congestion Control
algorithm to be written in BPF.  It opens up opportunities to allow
a faster turnaround time in testing/releasing new congestion control
ideas to production environment.

The same flexibility can be extended to writing TCP header option.
It is not uncommon that people want to test new TCP header option
to improve the TCP performance.  Another use case is for data-center
that has a more controlled environment and has more flexibility in
putting header options for internal traffic only.
```

Stateless routing of TCP packets

Use BPF TCP header options

- sock_ops program attached to *cgroup*
  - Gets callback on events such as *LISTEN, CONNECT, CONN_ESTD etc*
  - Can read and write TCP header options on each end point

[PATCH v3 bpf-next 0/9] BPF TCP header options

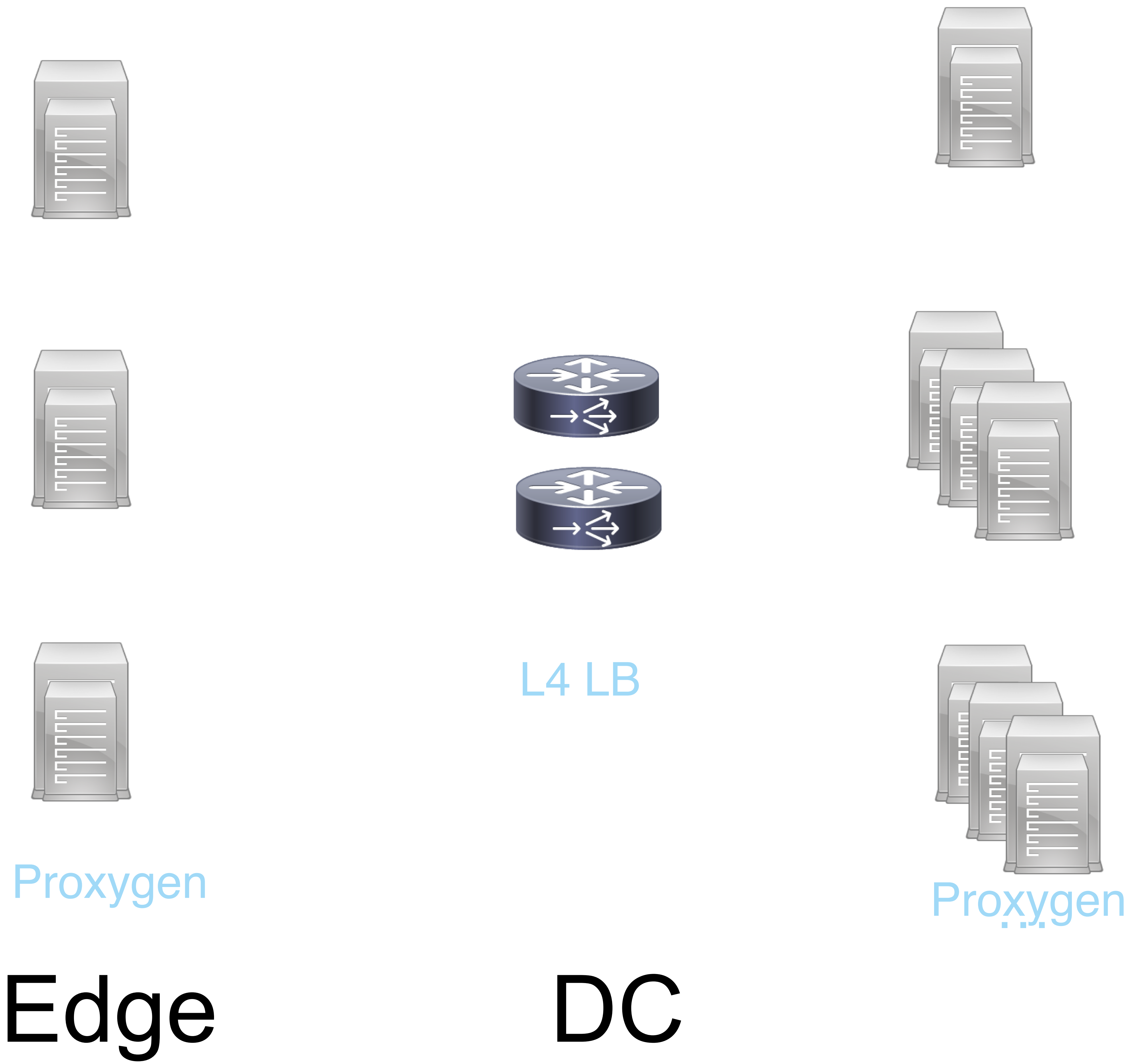
[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

- *Subject*: [PATCH v3 bpf-next 0/9] BPF TCP header options
- *From*: Martin KaFai Lau <kafai@xxxxxx>
- *Date*: Thu, 30 Jul 2020 13:56:57 -0700
- *Cc*: Alexei Starovoitov <ast@xxxxxxxxxx>, Daniel Borkmann <daniel@xxxxxxxxxxxxx>, Eric Dumazet <netdev@xxxxxxxxxxxxxxx>, Yuchung Cheng <ycheng@xxxxxxxxxx>
- *Smtp-origin-cluster*: ftw2c04
- *Smtp-origin-hostname*: devbig005.ftw2.facebook.com
- *Smtp-origin-hostprefix*: devbig

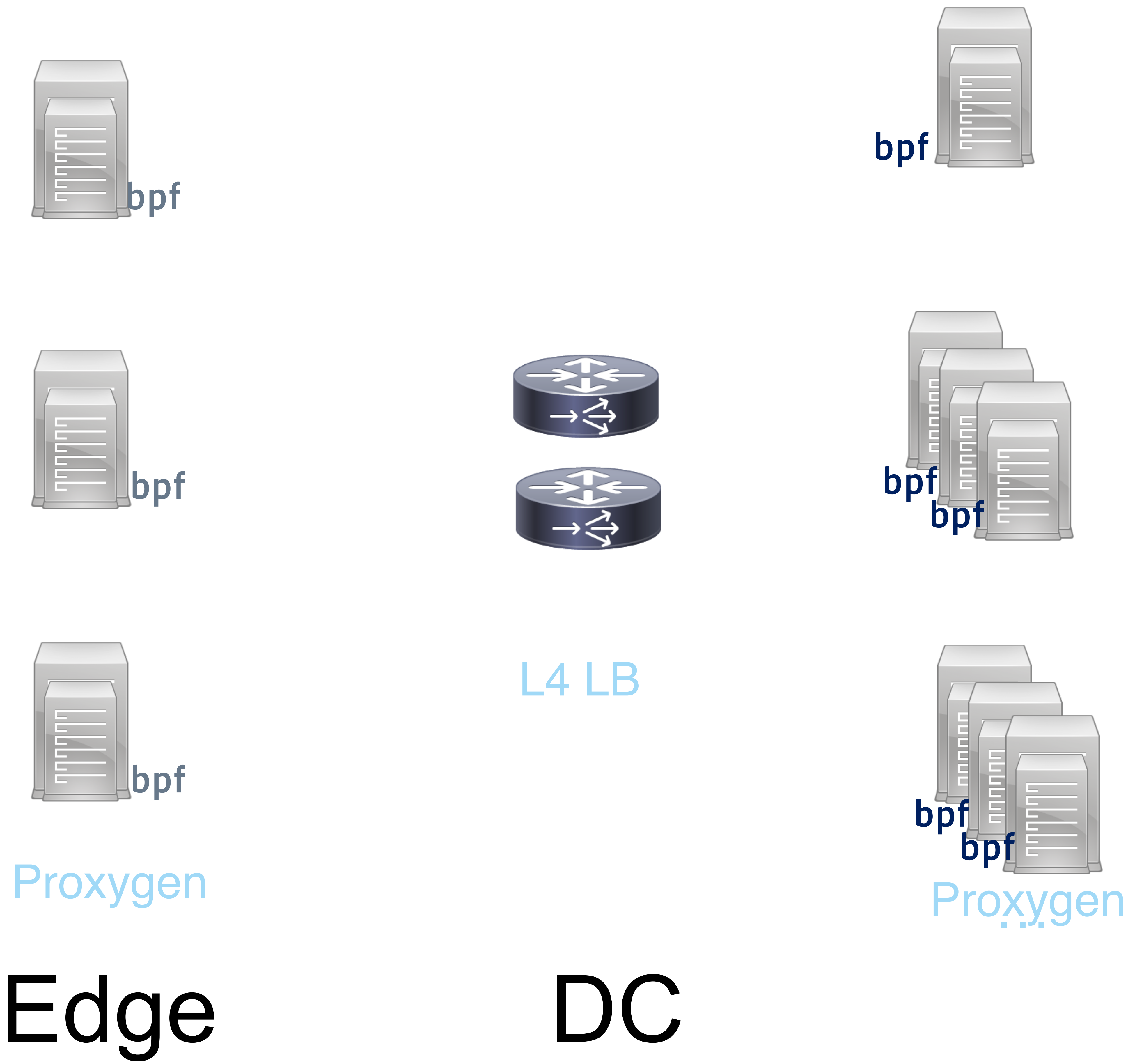
```
The earlier effort in BPF-TCP-CC allows the TCP Congestion Control
algorithm to be written in BPF.  It opens up opportunities to allow
a faster turnaround time in testing/releasing new congestion control
ideas to production environment.

The same flexibility can be extended to writing TCP header option.
It is not uncommon that people want to test new TCP header option
to improve the TCP performance.  Another use case is for data-center
that has a more controlled environment and has more flexibility in
putting header options for internal traffic only.
```

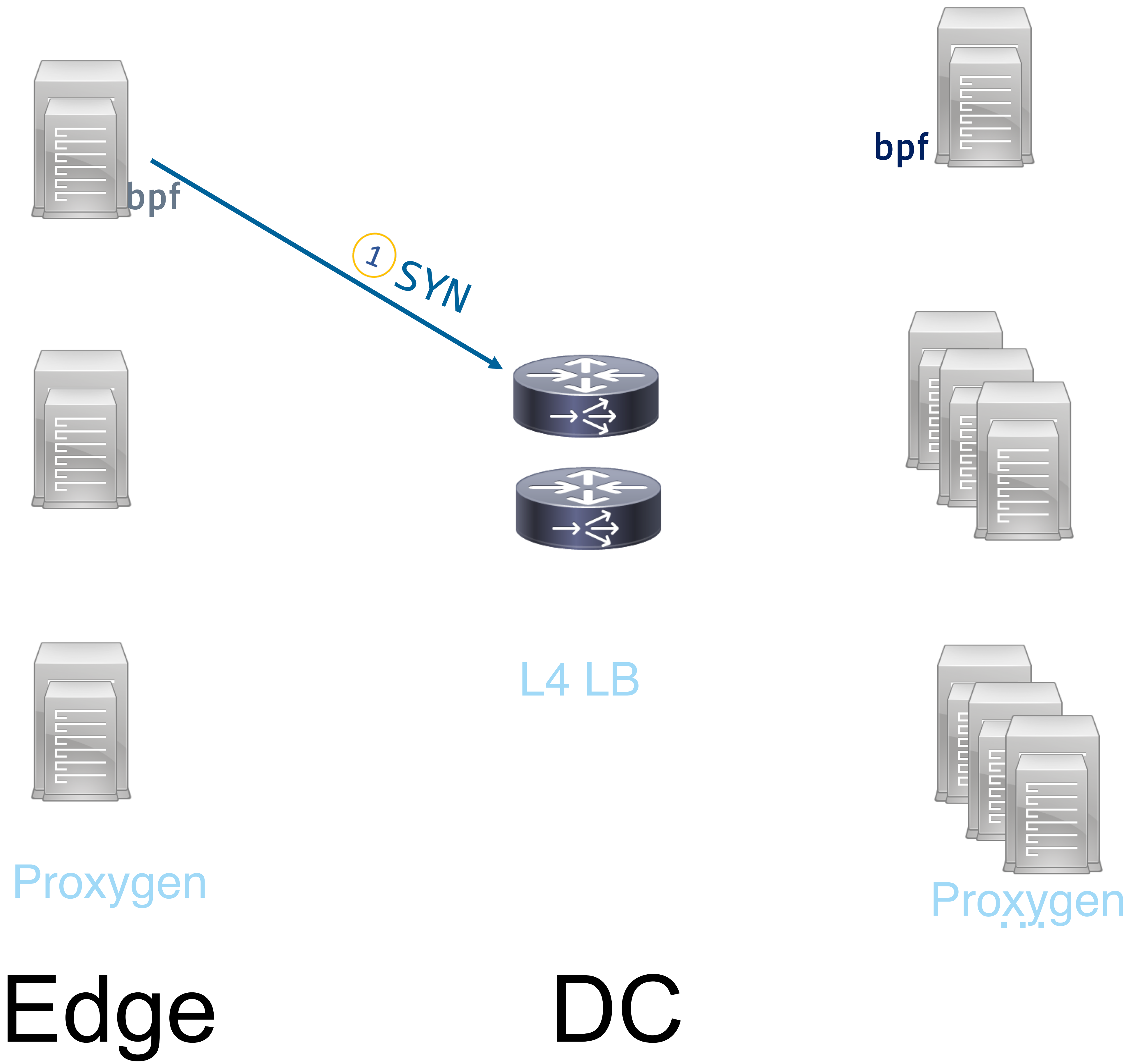
Execution in the datapath



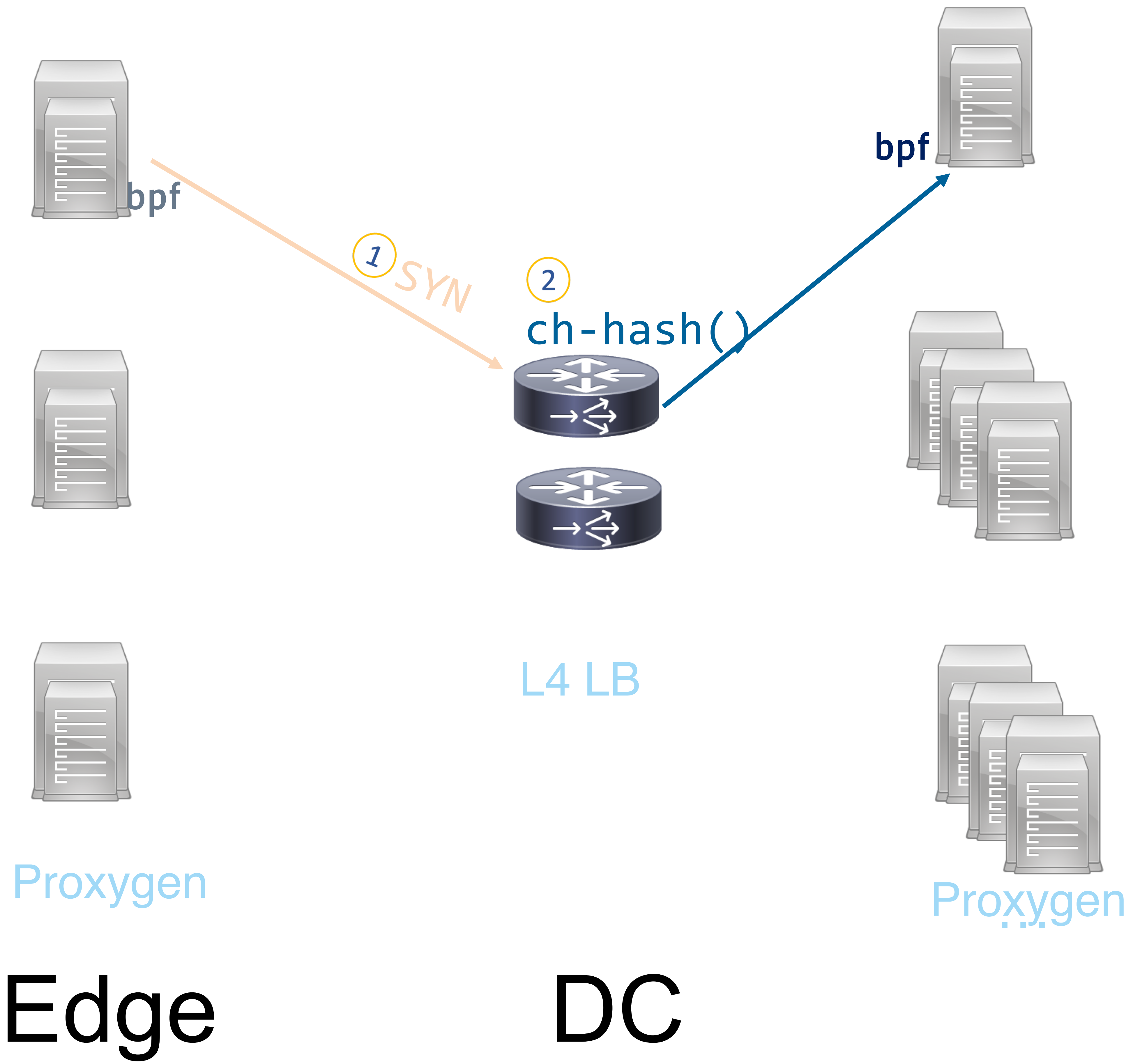
Execution in the datapath



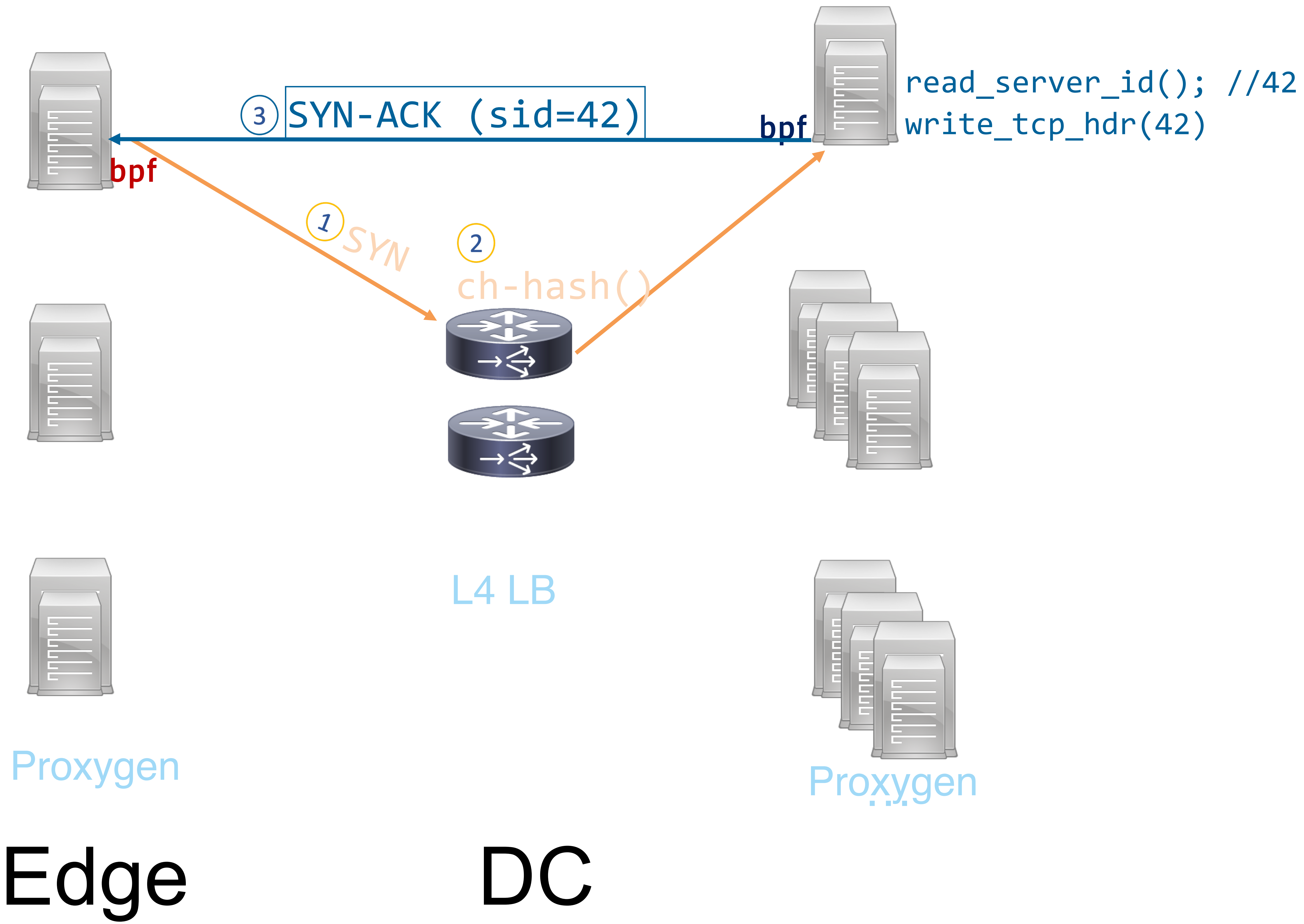
Execution in the datapath



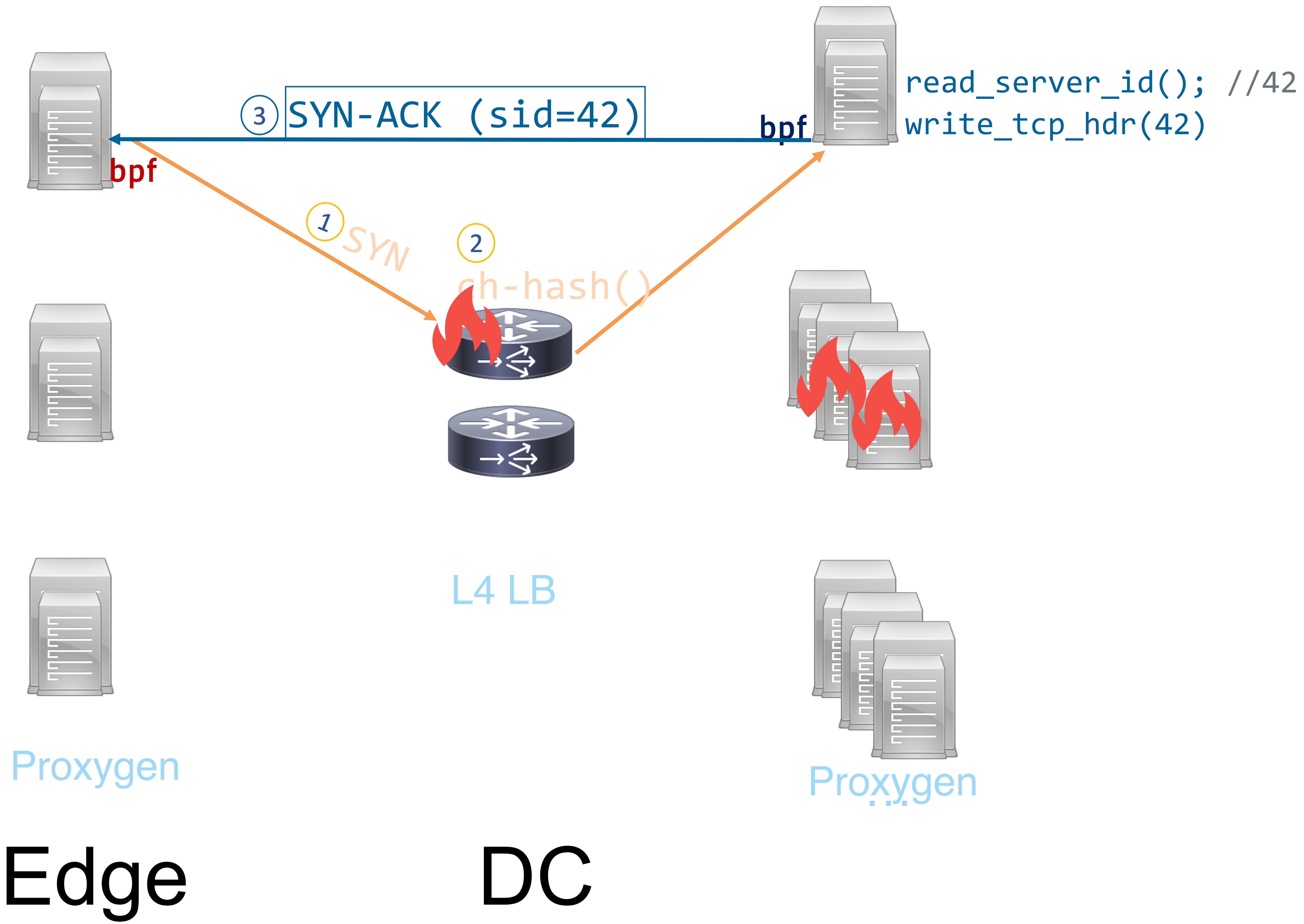
Execution in the datapath



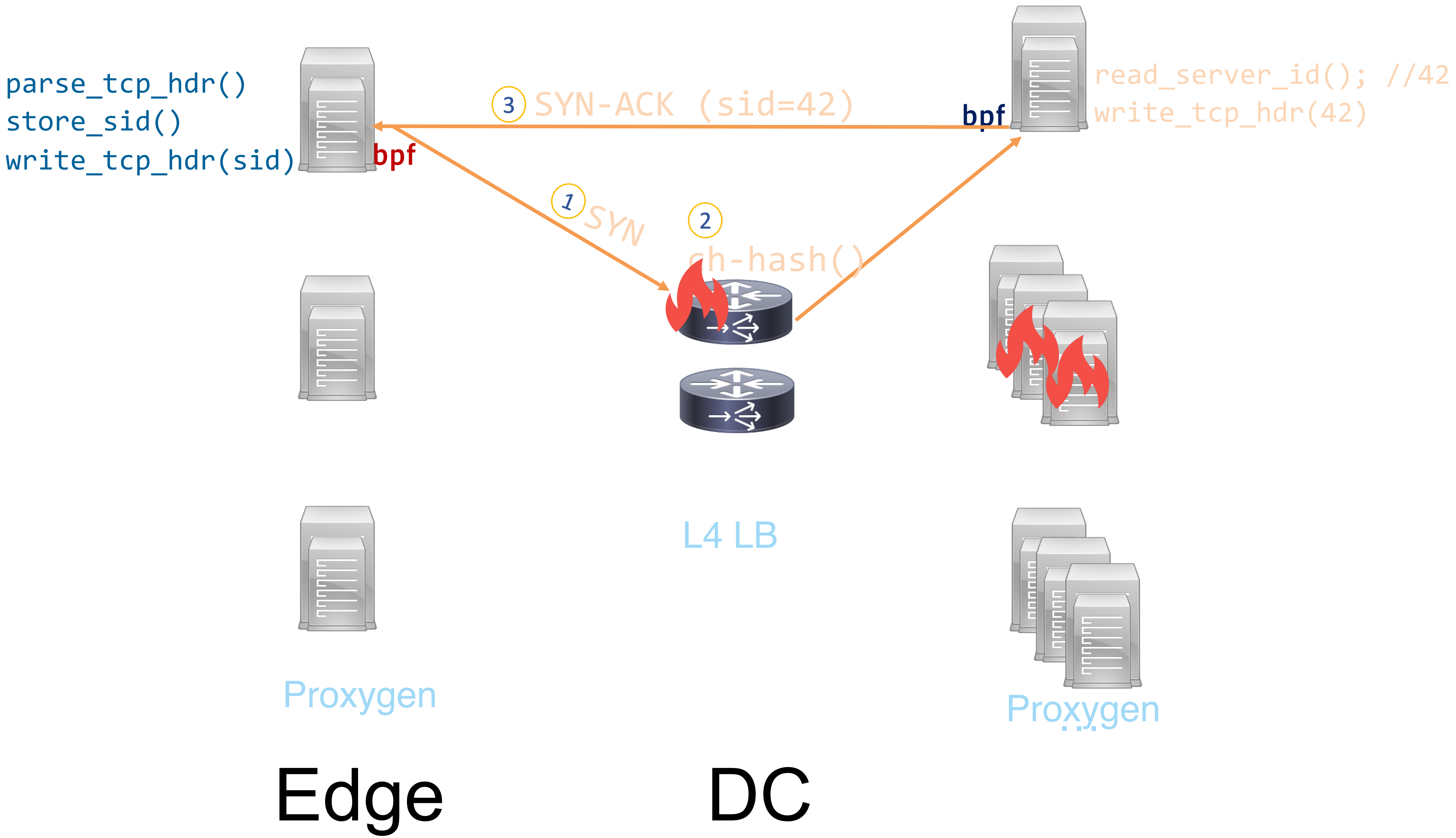
Execution in the datapath



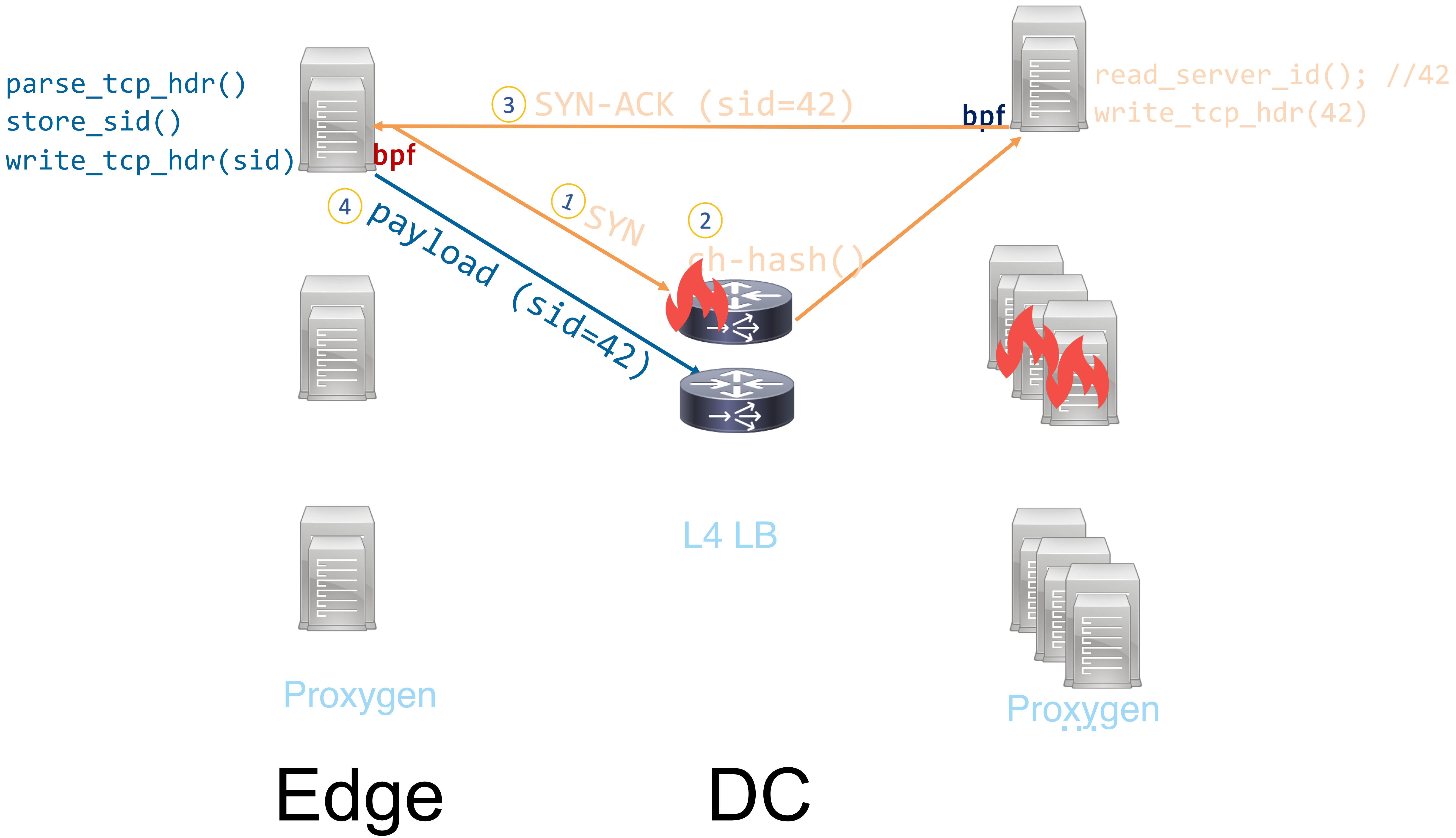
Execution in the datapath



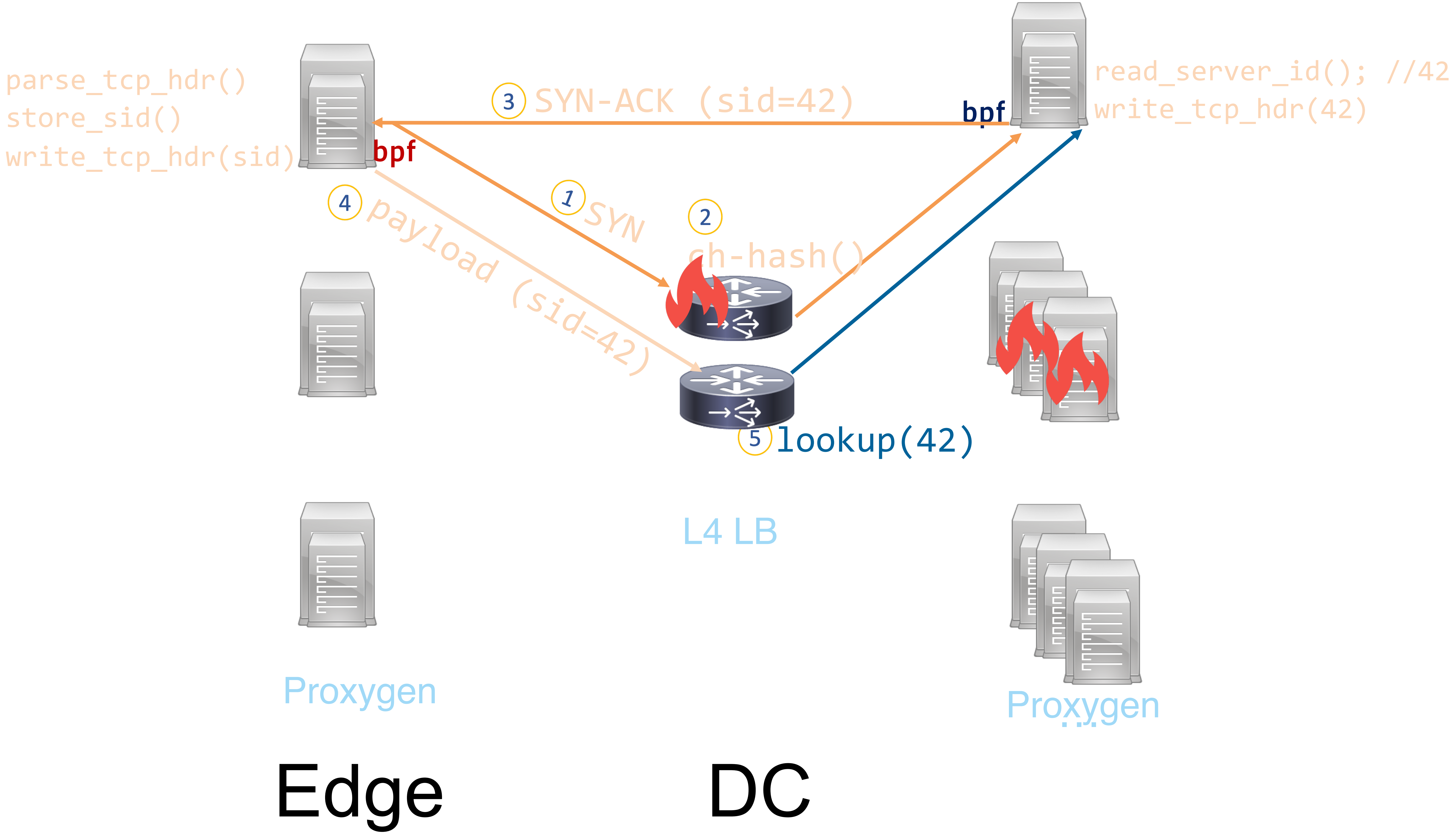
Execution in the datapath



Execution in the datapath



Execution in the datapath



Overhead in the data-path

Data overhead

```
struct tcp_opt {
  uint8_t kind;
  uint8_t len;
  uint32_t server_id;
}; // 6-bytes total
```

Runtime overhead: Parse TCP header for possible `server_id` in Katran (L4)

Implementation details

Operations

```
switch (skops->op) {
    case BPF_SOCK_OPS_TCP_LISTEN_CB:
    case BPF_SOCK_OPS_PASSIVE_ESTABLISHED_CB:
    case BPF_SOCK_OPS_TCP_CONNECT_CB:
    case BPF_SOCK_OPS_ACTIVE_ESTABLISHED_CB:
    case BPF_SOCK_OPS_PARSE_HDR_OPT_CB:
    case BPF_SOCK_OPS_HDR_OPT_LEN_CB:
    case BPF_SOCK_OPS_WRITE_HDR_OPT_CB:
    . . .
}
```

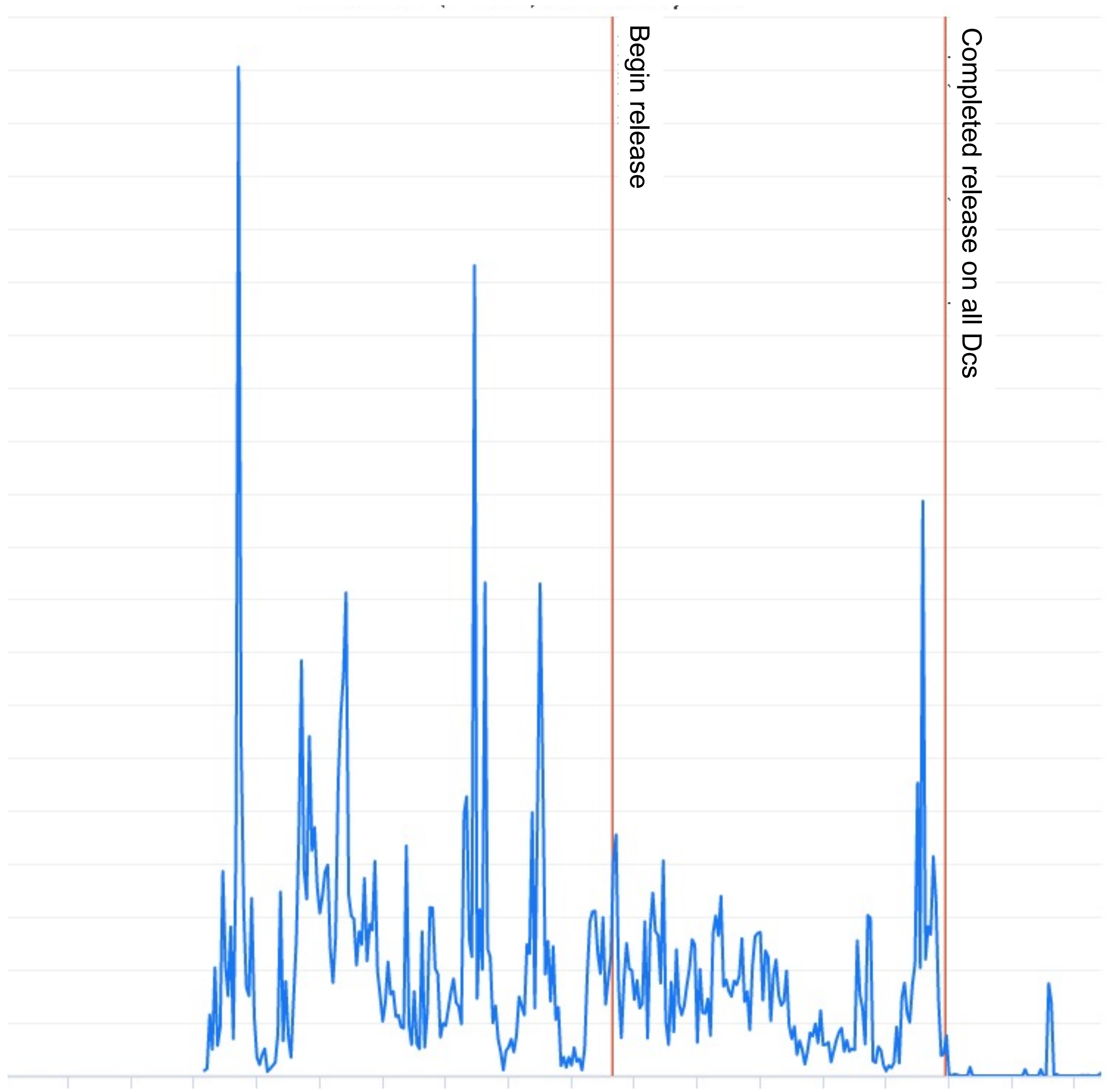
Storage: use `bpf_sk_storage` to store `server_id` per flow within each end-point

Assignment and propagation of `server_id`

- An offline workflow assigns and propagates `server_id`
- Control planes of Katran and Proxygen load them onto their data planes
- Same pipeline for both QUIC and TCP

Results

Total errors due to connection resets for an application with long lived connections



Limitations

- Only feasible if you control both end points
- Useful for typical setup in Data centers
  - Requires embedding the `server_id` in each TCP packet
- Typically not feasible in external clients for TCP
  - Middleboxes and firewalls could drop it as well

Recap

Embed with `server_id` in TCP hdr for stateless routing

- Completely stateless solution
- No tangible extra cost in terms of CPU / memory
- Alternatives are quite complex
    - Share states between hosts
    - Embed `server_id` in fields such as ECR

Questions?

Thank you!