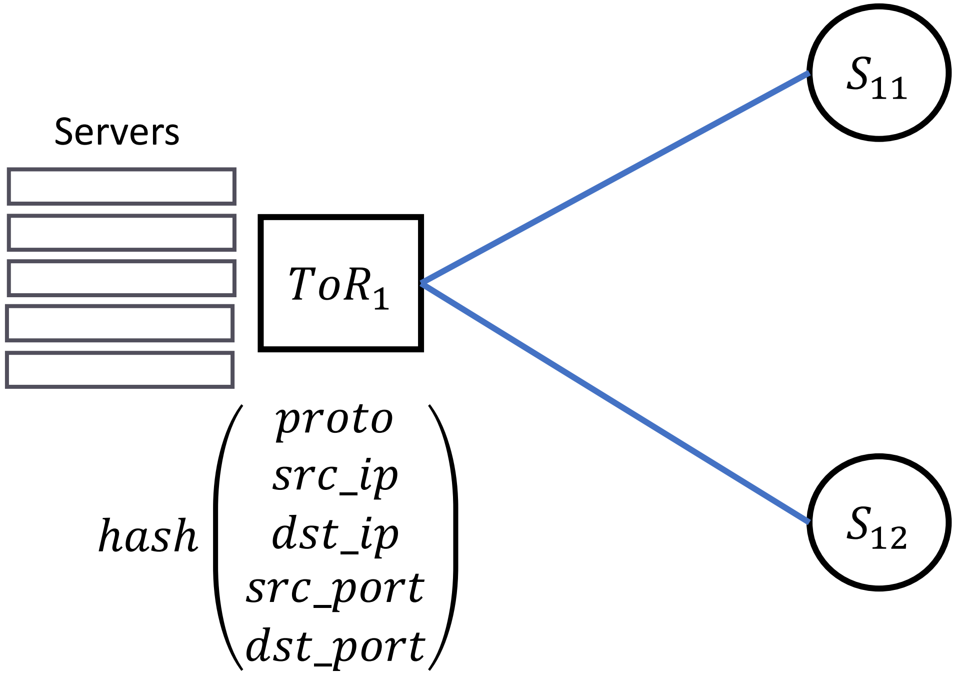
Self-healing Networking with Flow Label

Alexander Azimov mitradir@yandex-team.ru

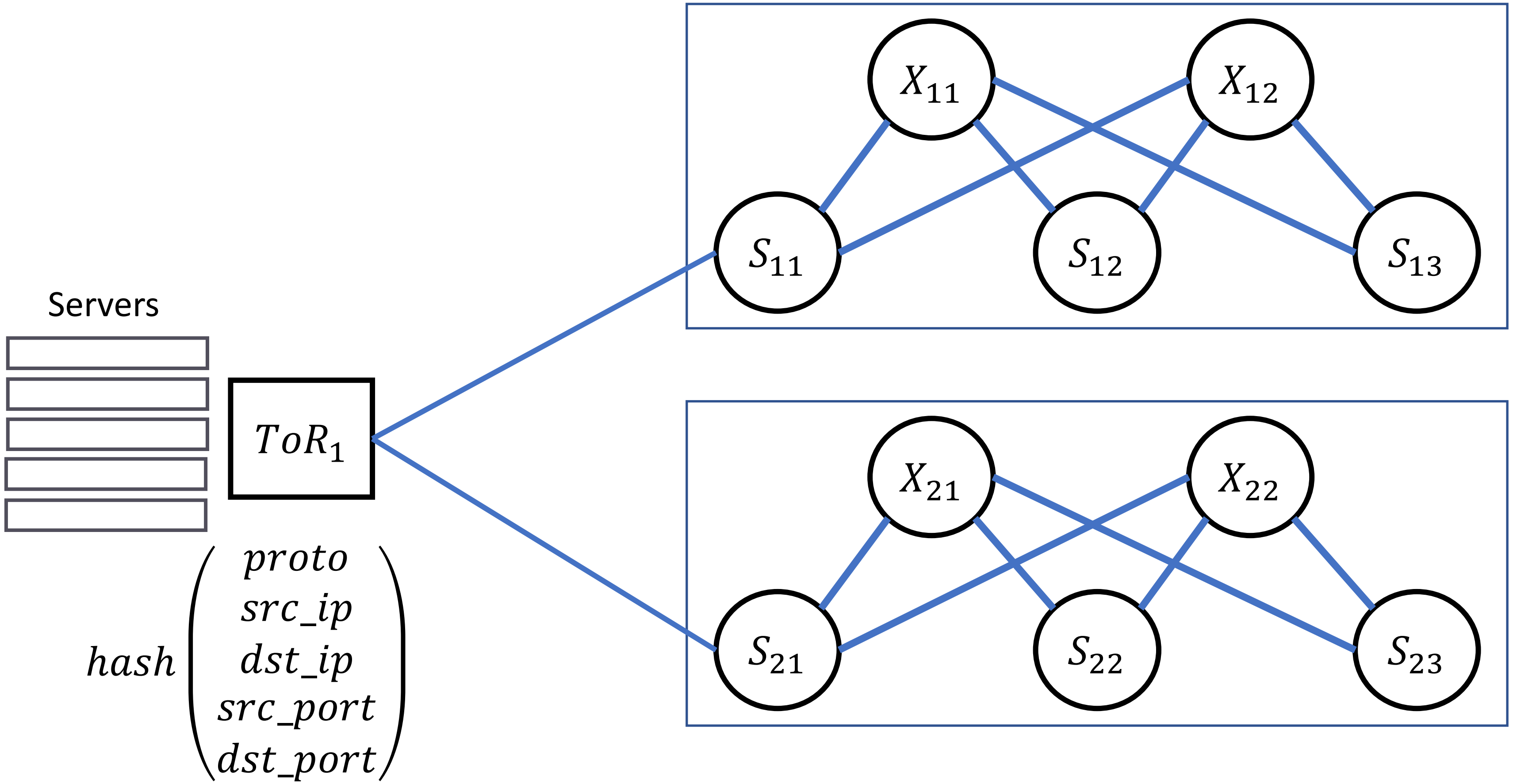
Dmitry Yakunin zeil@yandex-team.ru

Just a Top of Rack Switch (ToR)



$$hash\begin{pmatrix} proto \\ src\_ip \\ dst\_ip \\ src\_port \\ dst\_port \end{pmatrix}$$

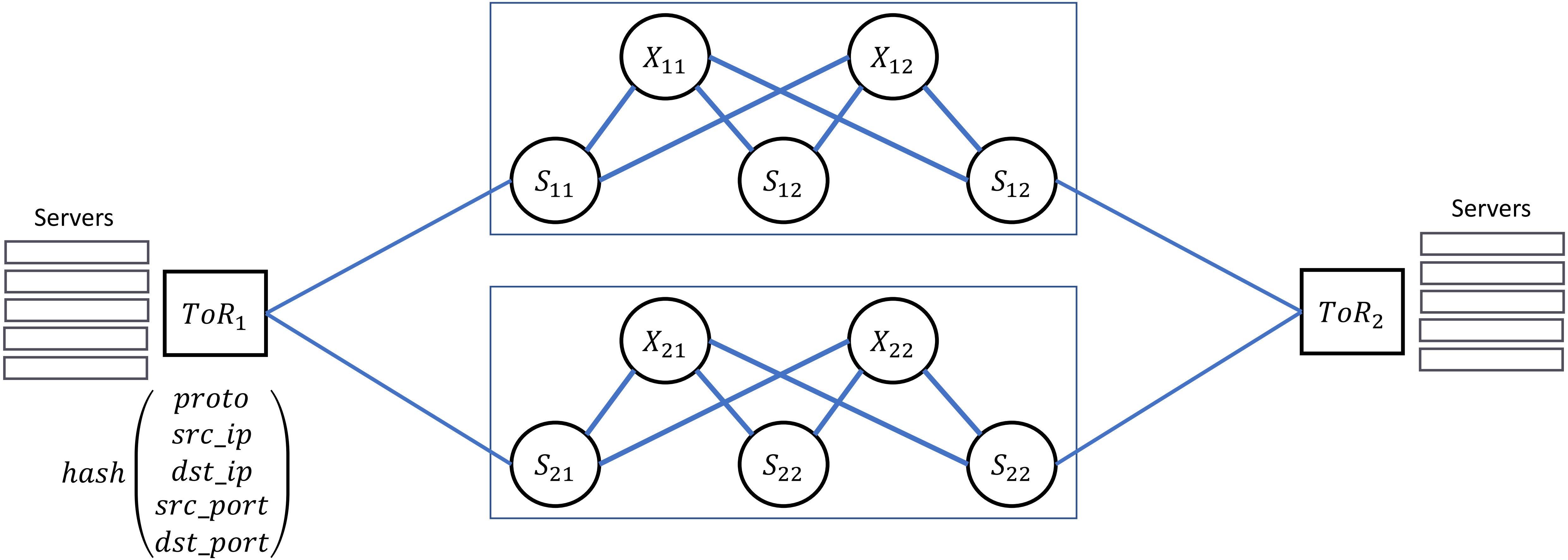
ToR + 2xPlanes



$$hash\begin{pmatrix} proto \\ src\_ip \\ dst\_ip \\ src\_port \\ dst\_port \end{pmatrix}$$

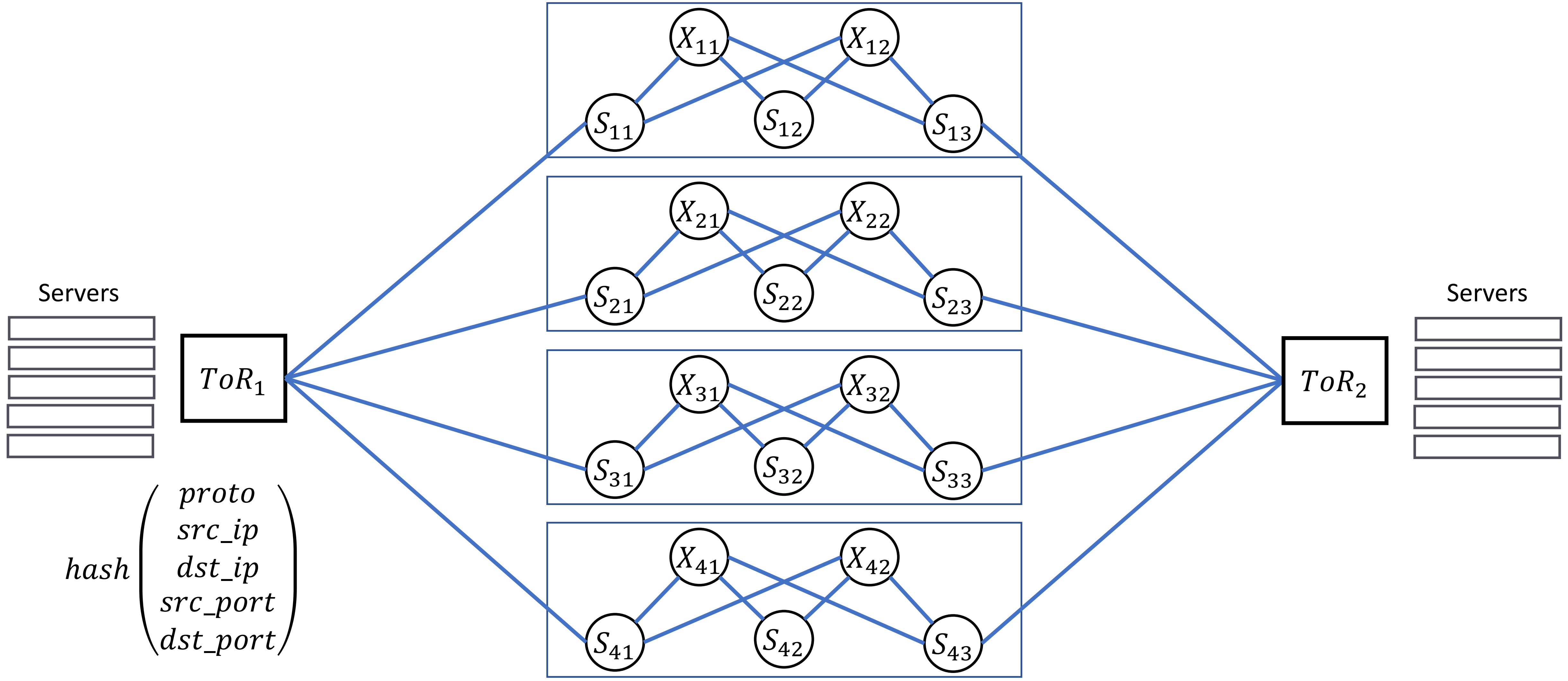
S – spine;
X – super spine;

ToR + 2xPlanes + ToR



$$hash\begin{pmatrix} proto \\ src\_ip \\ dst\_ip \\ src\_port \\ dst\_port \end{pmatrix}$$

ToR + 4xPlanes + ToR



$$hash\begin{pmatrix} proto \\ src\_ip \\ dst\_ip \\ src\_port \\ dst\_port \end{pmatrix}$$

Theory DC: Many-Many Paths

N_PLANES: Number of planes in DC;

N_X_SPINES: Number of super spines (X) in each plane;

- Inside ToR: 1
- Inside PoD: N_PLANES
- Between PoDs: N_PLANES x N_X_SPINES

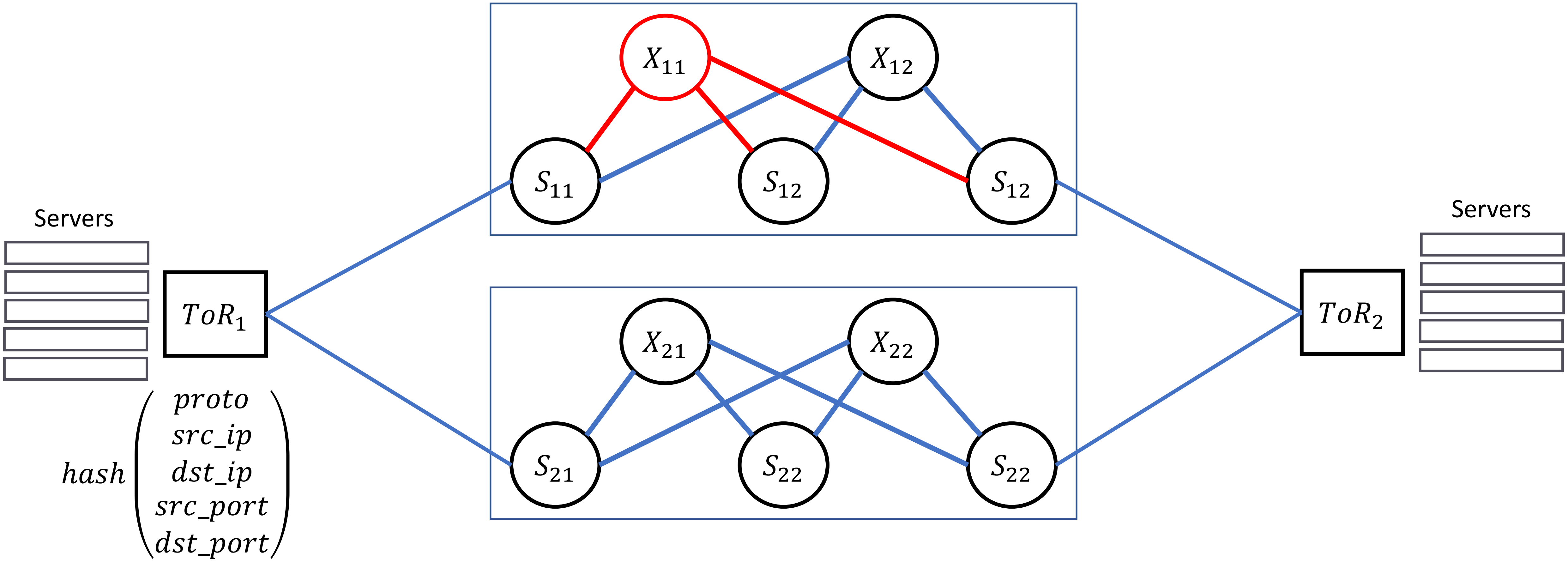
Real DC: Many-Many Paths

N_PLANES: Number of planes in DC; (8)

N_X_SPINES: Number of super spines (X) in each plane; (32)

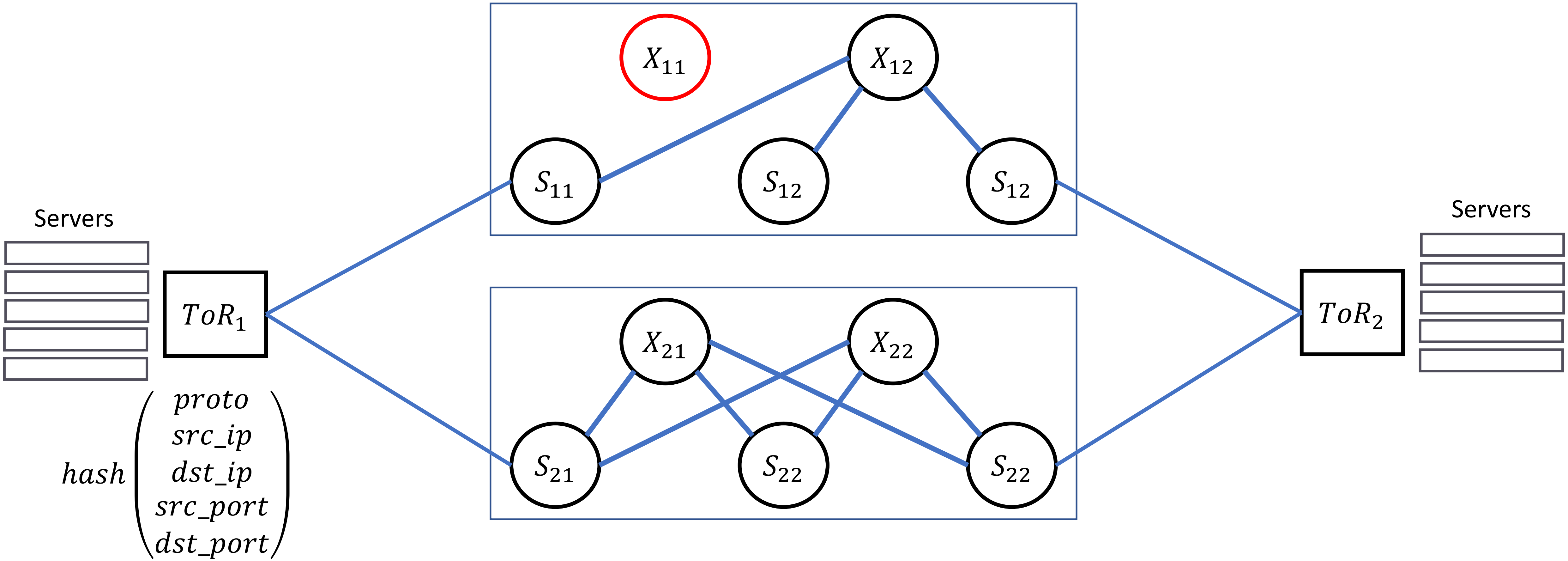
- Inside ToR: 1
- Inside PoD: N_PLANES = 8
- Between PoDs: N_PLANES x N_X_SPINES = 256

X_{11} is Broken



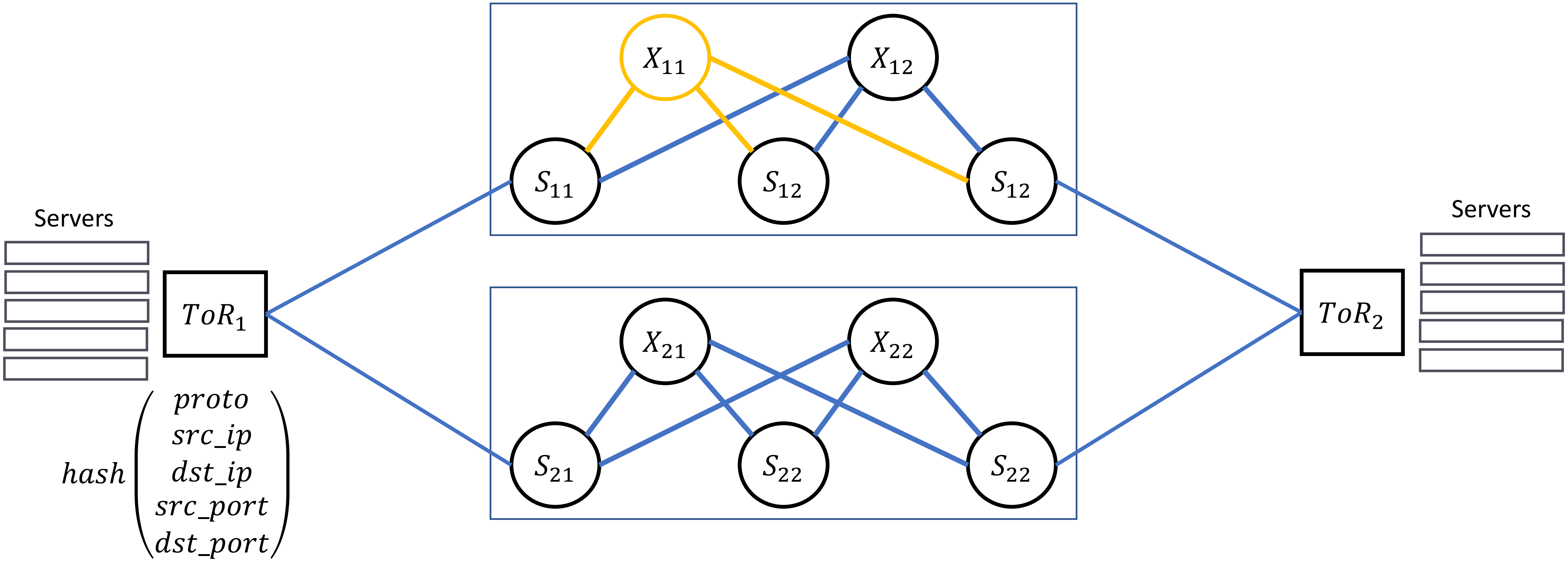
$$hash\begin{pmatrix} proto \\ src\_ip \\ dst\_ip \\ src\_port \\ dst\_port \end{pmatrix}$$

X_{11} is Broken: No link, No Problem

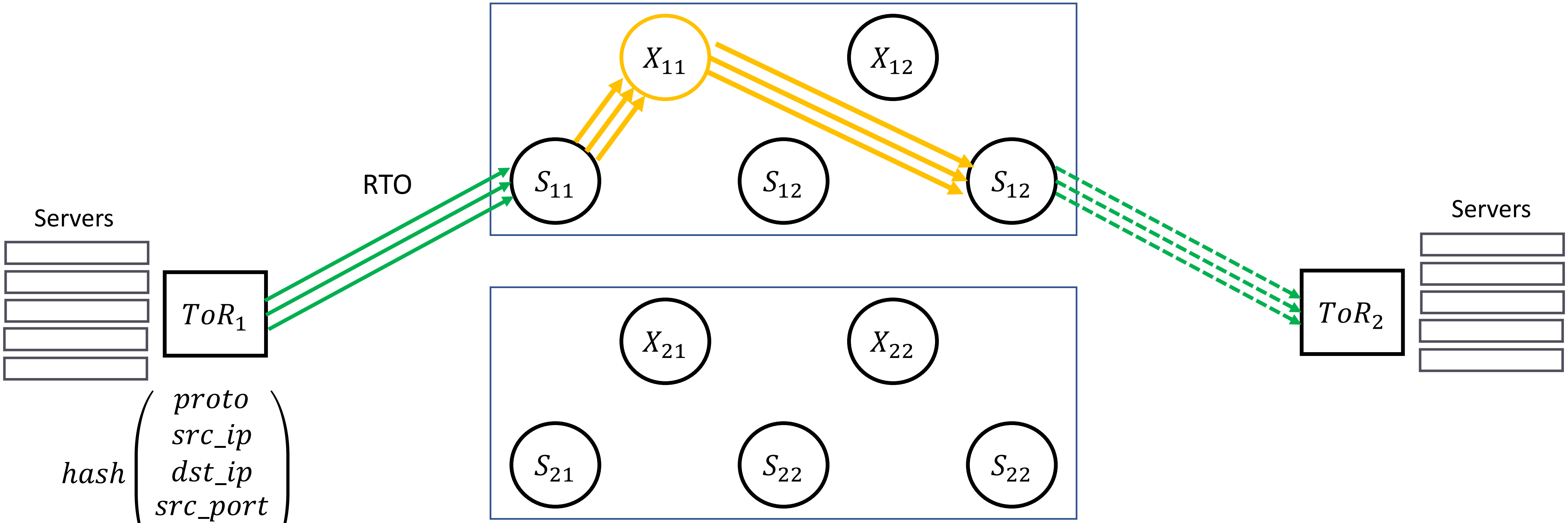


$$hash\begin{pmatrix} proto \\ src\_ip \\ dst\_ip \\ src\_port \\ dst\_port \end{pmatrix}$$

X_{11} is Broken: Constant Loss

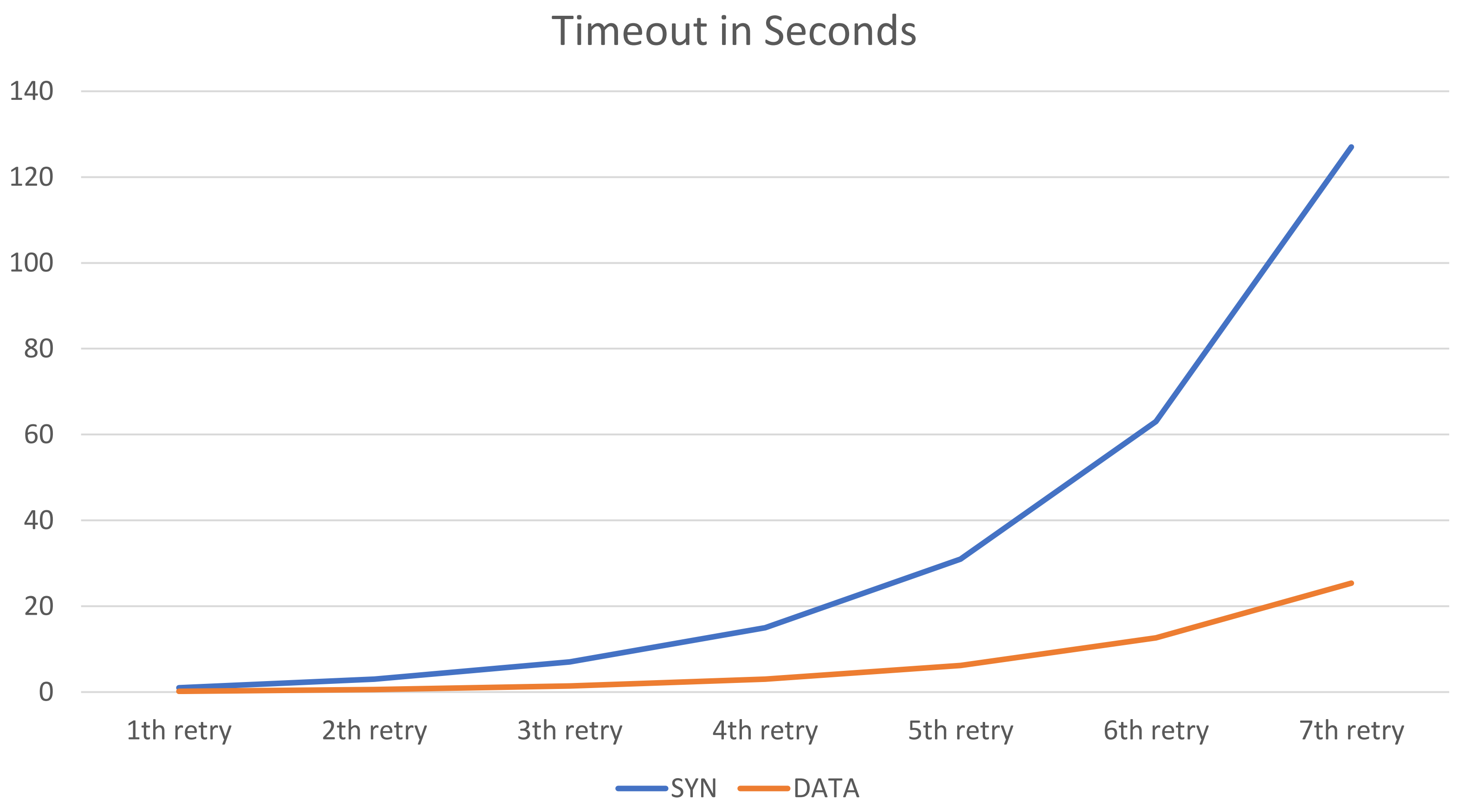


Unhappy TCP Flow



$$hash\begin{pmatrix} proto \\ src\_ip \\ dst\_ip \\ src\_port \\ dst\_port \end{pmatrix}$$

RTO & SYN_RTO Timeouts



Timeouts

RTO_MIN	SYN_RTO
200ms	1s

Real RTT

<1ms

RTO = MAX(RTO_MIN, RTT)

The Old Way: Services

- Configure TCP options using sysctl;
- Configure application timeouts;
- TCP sessions reuse with software defined health checks;
- None of these methods are properly evaluated;

The Old Way: NOC

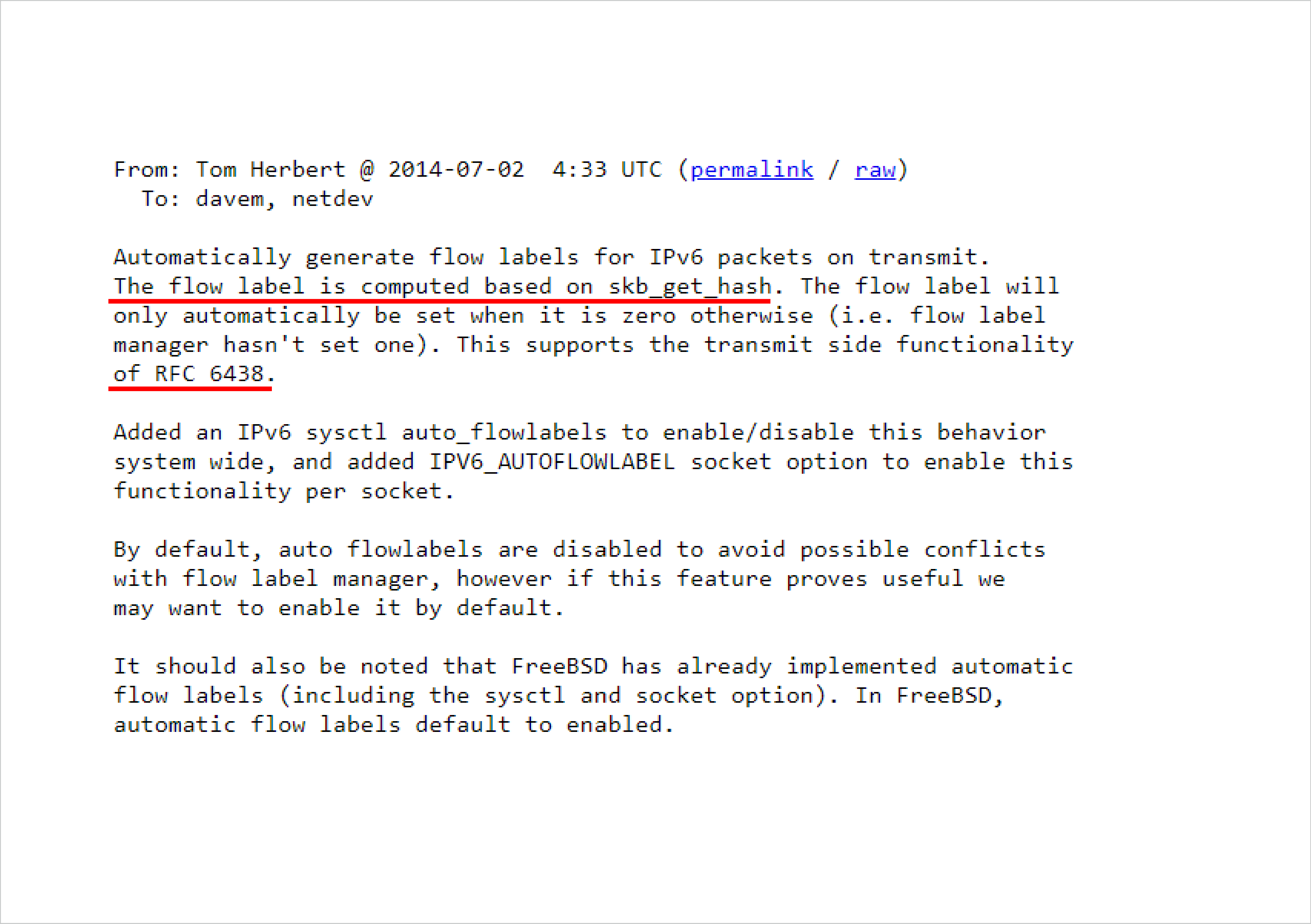
The Old Way: NOC

- Outage!
- Detection (1-5 minutes);
- Isolation (5-15 minutes);

Total: 5-20 minutes of service degradation.

Linux Kernel

2014



From: Tom Herbert @ 2014-07-02 4:33 UTC (permalink / raw)
  To: davem, netdev

Automatically generate flow labels for IPv6 packets on transmit. The flow label is computed based on skb_get_hash. The flow label will only automatically be set when it is zero otherwise (i.e. flow label manager hasn't set one). This supports the transmit side functionality of RFC 6438.

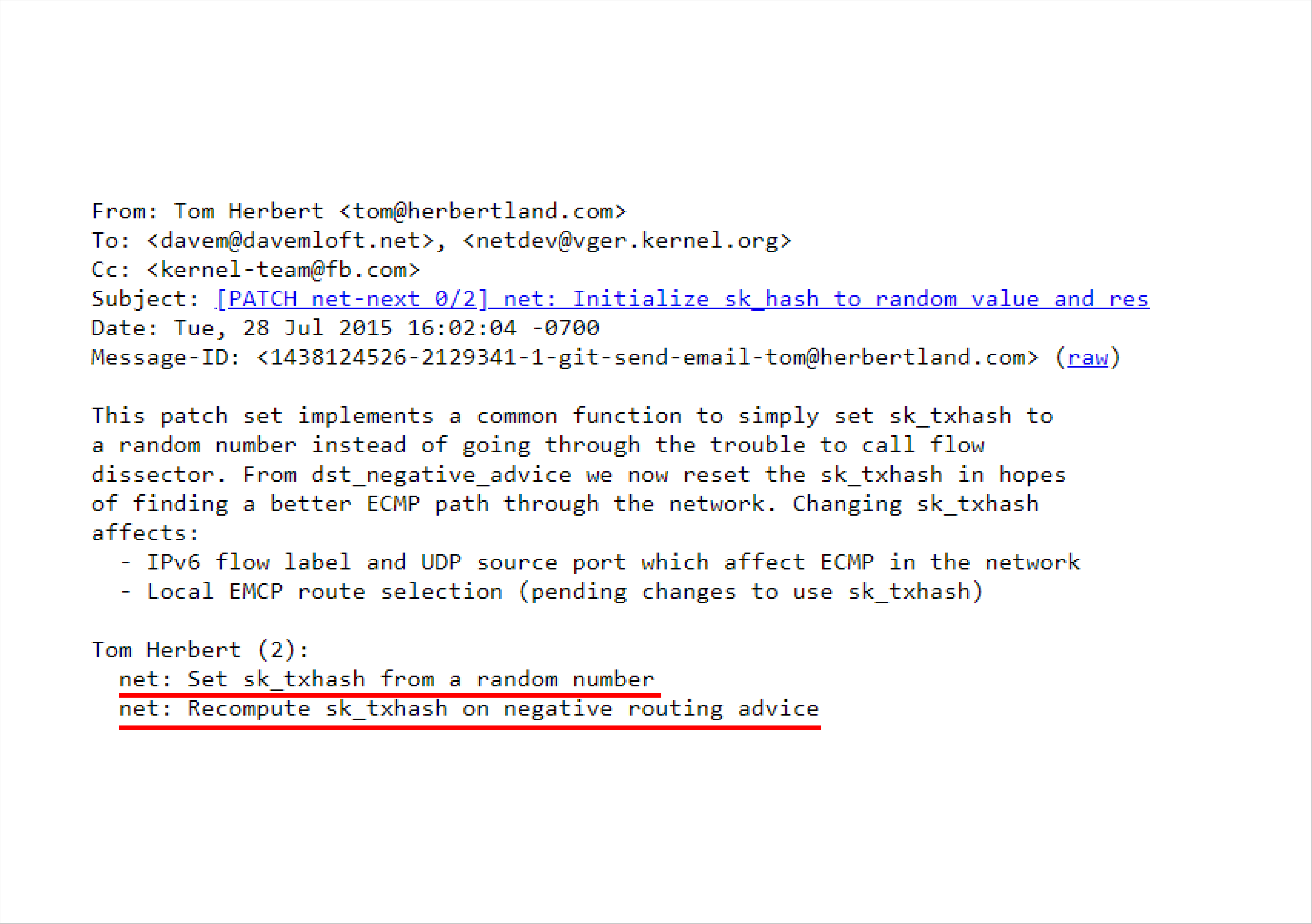
Added an IPv6 sysctl auto_flowlabels to enable/disable this behavior system wide, and added IPV6_AUTOFLOWLABEL socket option to enable this functionality per socket.

By default, auto flowlabels are disabled to avoid possible conflicts with flow label manager, however if this feature proves useful we may want to enable it by default.

It should also be noted that FreeBSD has already implemented automatic flow labels (including the sysctl and socket option). In FreeBSD, automatic flow labels default to enabled.

Linux Kernel

2015



From: Tom Herbert <tom@herbertland.com>
To: <davem@davemloft.net>, <netdev@vger.kernel.org>
Cc: <kernel-team@fb.com>
Subject: [PATCH net-next 0/2] net: Initialize sk_hash to random value and res
Date: Tue, 28 Jul 2015 16:02:04 -0700
Message-ID: <1438124526-2129341-1-git-send-email-tom@herbertland.com> (raw)

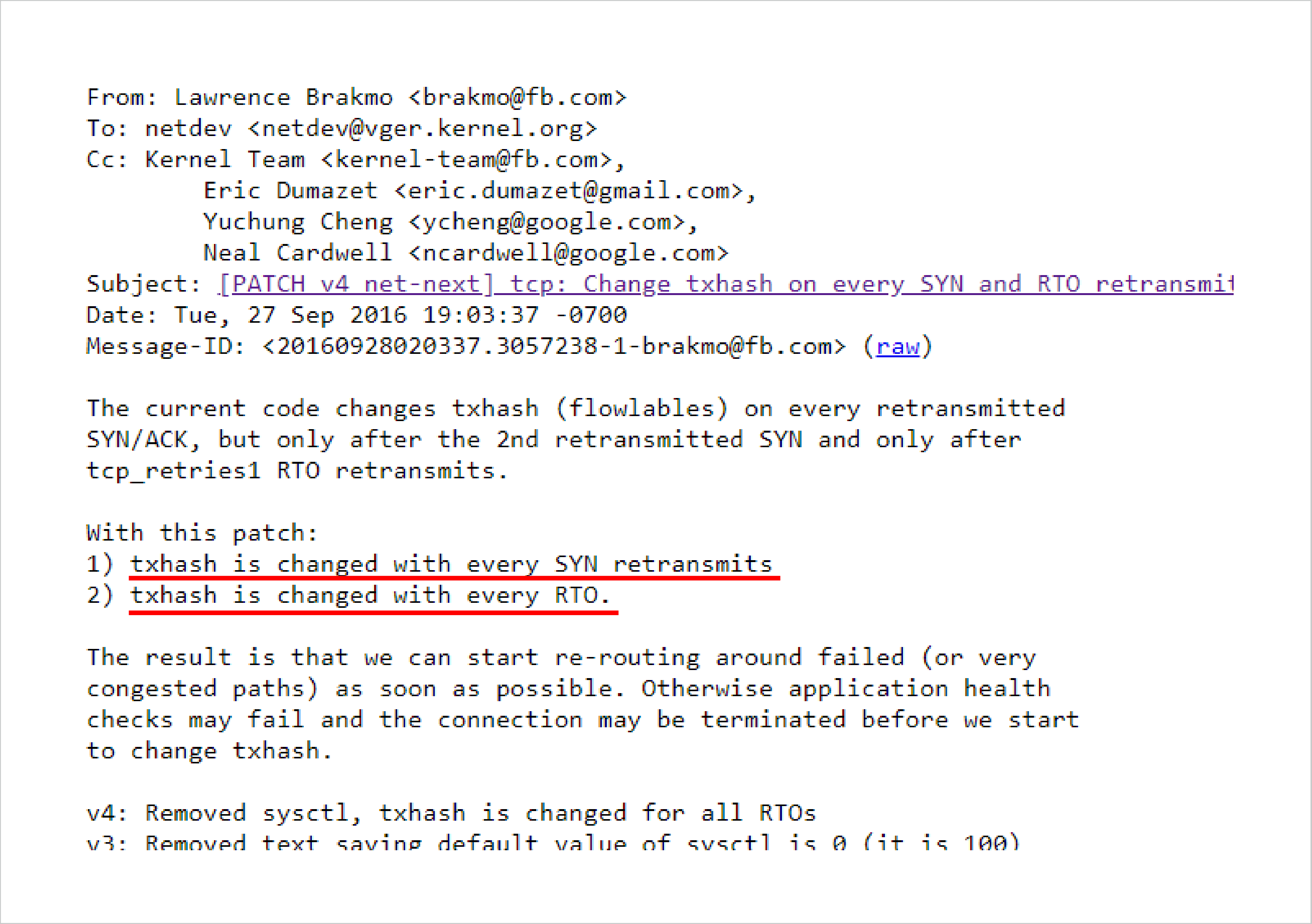
This patch set implements a common function to simply set sk_txhash to a random number instead of going through the trouble to call flow dissector. From dst_negative_advice we now reset the sk_txhash in hopes of finding a better ECMP path through the network. Changing sk_txhash affects:

- IPv6 flow label and UDP source port which affect ECMP in the network
- Local EMCP route selection (pending changes to use sk_txhash)

Tom Herbert (2):
  net: Set sk_txhash from a random number
  net: Recompute sk_txhash on negative routing advice

Linux Kernel

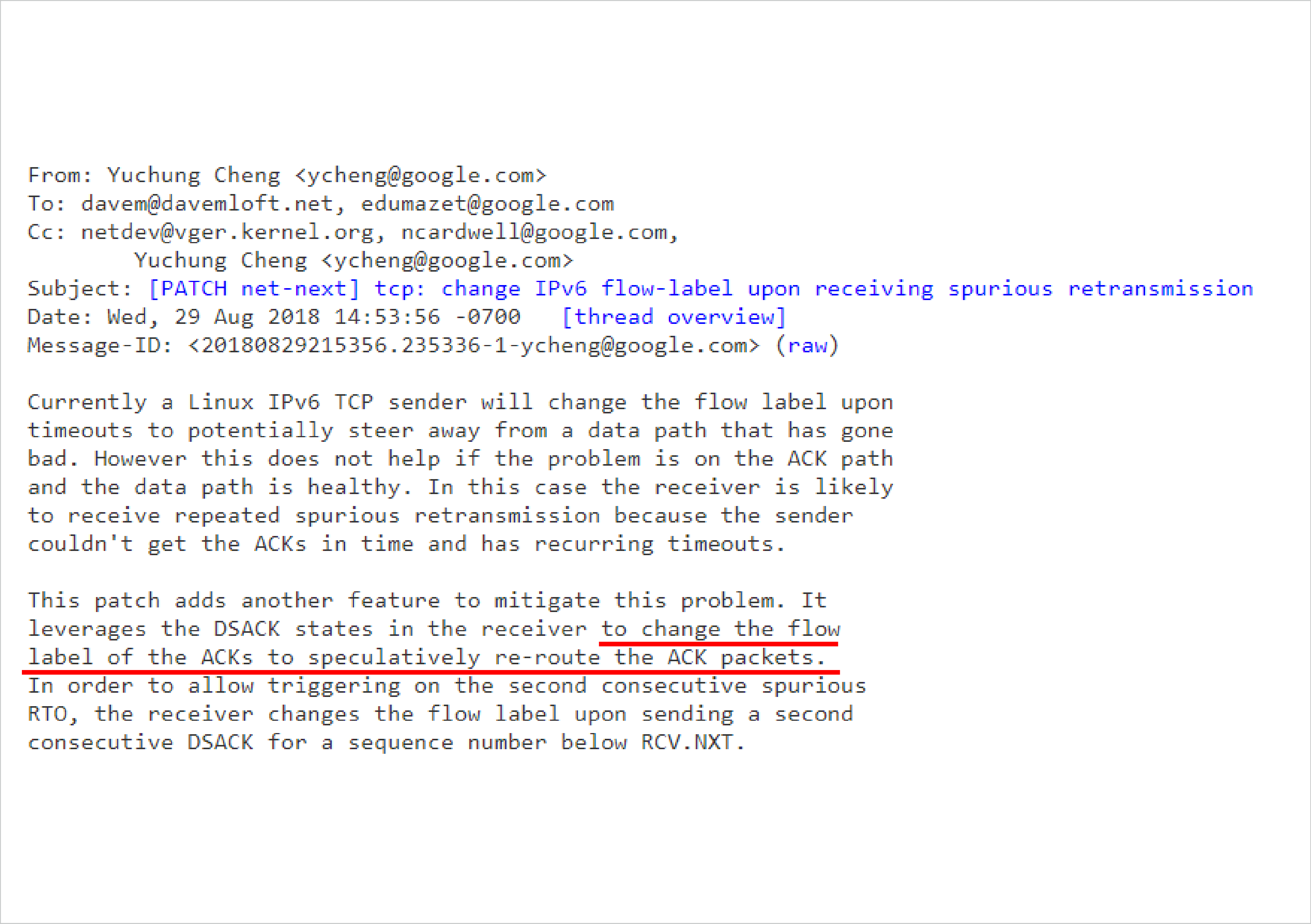
2016



```
From: Lawrence Brakmo <brakmo@fb.com>
To: netdev <netdev@vger.kernel.org>
Cc: Kernel Team <kernel-team@fb.com>,
        Eric Dumazet <eric.dumazet@gmail.com>,
        Yuchung Cheng <ycheng@google.com>,
        Neal Cardwell <ncardwell@google.com>
Subject: [PATCH v4 net-next] tcp: Change txhash on every SYN and RTO retransmit
Date: Tue, 27 Sep 2016 19:03:37 -0700
Message-ID: <20160928020337.3057238-1-brakmo@fb.com> (raw)

The current code changes txhash (flowlables) on every retransmitted
SYN/ACK, but only after the 2nd retransmitted SYN and only after
tcp_retries1 RTO retransmits.

With this patch:
1) txhash is changed with every SYN retransmits
2) txhash is changed with every RTO.

The result is that we can start re-routing around failed (or very
congested paths) as soon as possible. Otherwise application health
checks may fail and the connection may be terminated before we start
to change txhash.

v4: Removed sysctl, txhash is changed for all RTOs
v3: Removed text saying default value of sysctl is 0 (it is 100)
```

Linux Kernel

2018

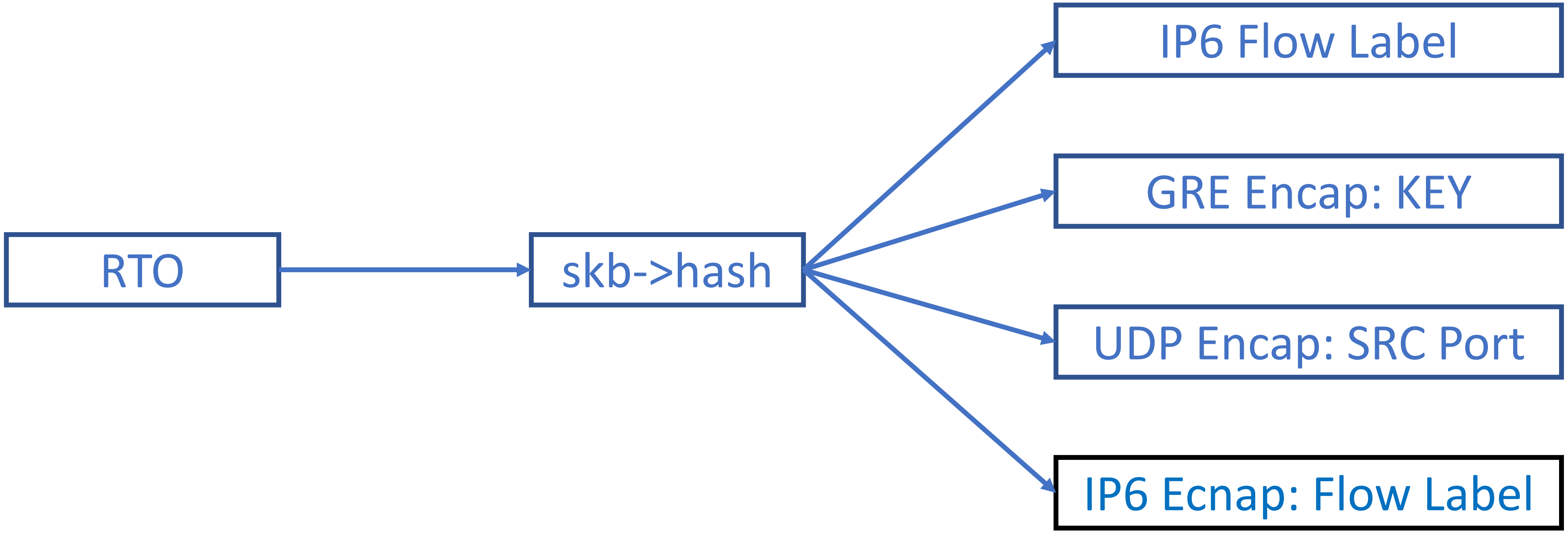


From: Yuchung Cheng <ycheng@google.com>
To: davem@davemloft.net, edumazet@google.com
Cc: netdev@vger.kernel.org, ncardwell@google.com,
Yuchung Cheng <ycheng@google.com>
Subject: [PATCH net-next] tcp: change IPv6 flow-label upon receiving spurious retransmission
Date: Wed, 29 Aug 2018 14:53:56 -0700 [thread overview]
Message-ID: <20180829215356.235336-1-ycheng@google.com> (raw)

Currently a Linux IPv6 TCP sender will change the flow label upon timeouts to potentially steer away from a data path that has gone bad. However this does not help if the problem is on the ACK path and the data path is healthy. In this case the receiver is likely to receive repeated spurious retransmission because the sender couldn't get the ACKs in time and has recurring timeouts.

This patch adds another feature to mitigate this problem. It leverages the DSACK states in the receiver to change the flow label of the ACKs to speculatively re-route the ACK packets. In order to allow triggering on the second consecutive spurious RTO, the receiver changes the flow label upon sending a second consecutive DSACK for a sequence number below RCV.NXT.

TCP RTO & skb->hash



RTO

skb->hash

IP6 Flow Label

GRE Encap: KEY

UDP Encap: SRC Port

IP6 Ecnap: Flow Label

net.ipv6.auto_flowlabels

0: automatic flow labels are completely disabled

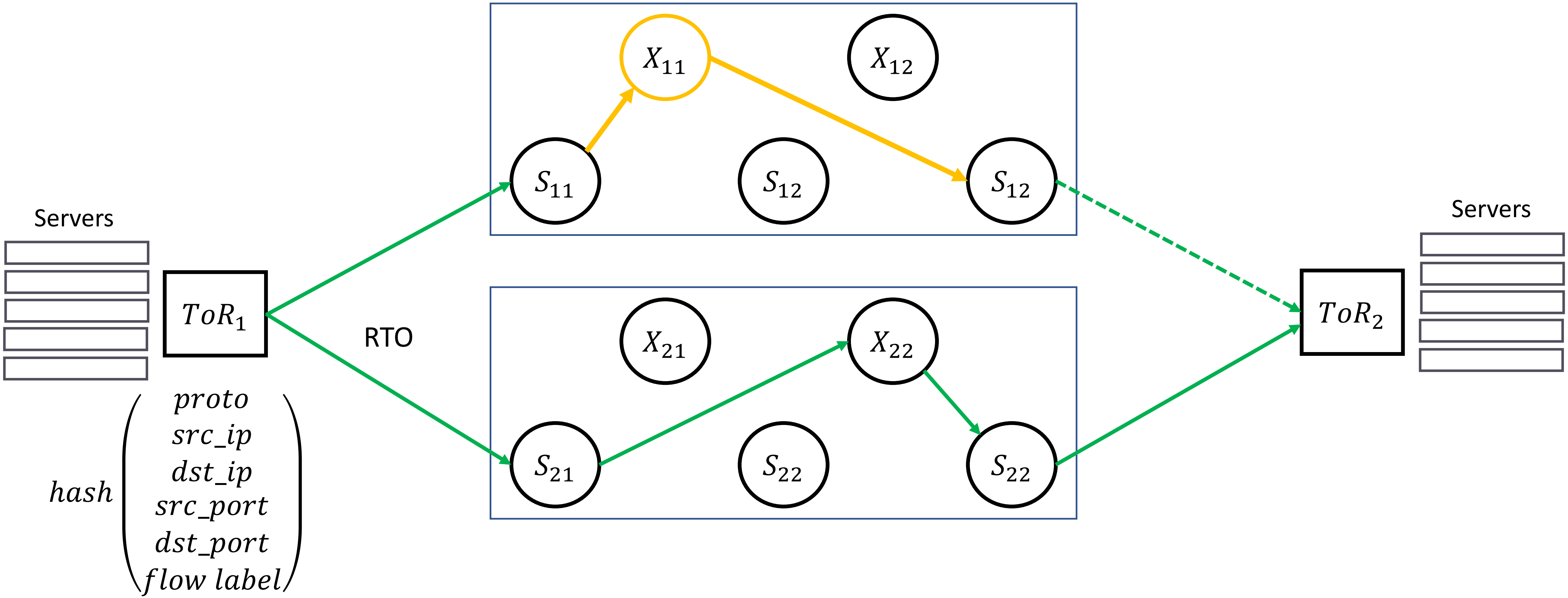
1: automatic flow labels are enabled by default, they can be disabled on a per socket basis using the IPV6_AUTOFLOWLABEL socket option

2: automatic flow labels are allowed, they may be enabled on a per socket basis using the IPV6_AUTOFLOWLABEL socket option

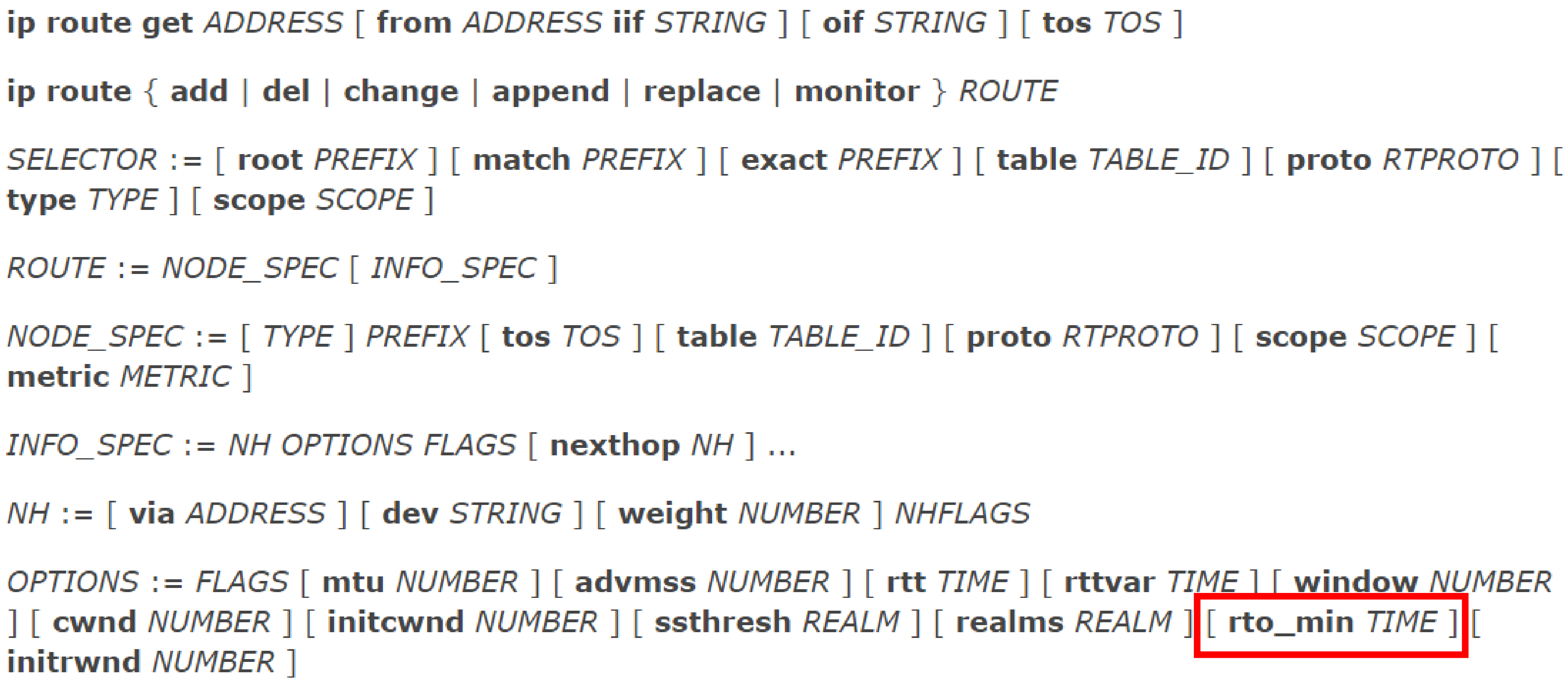
3: automatic flow labels are enabled and enforced, they cannot be disabled by the socket option

Default: 1

Unhappy TCP Flow Becomes Happier



How to Reduce RTO Timeouts?



ip route get *ADDRESS* [**from** *ADDRESS* **iif** *STRING*] [**oif** *STRING*] [**tos** *TOS*]

ip route { **add** | **del** | **change** | **append** | **replace** | **monitor** } *ROUTE*

SELECTOR := [**root** *PREFIX*] [**match** *PREFIX*] [**exact** *PREFIX*] [**table** *TABLE_ID*] [**proto** *RTPROTO*] [**type** *TYPE*] [**scope** *SCOPE*]

ROUTE := *NODE_SPEC* [*INFO_SPEC*]

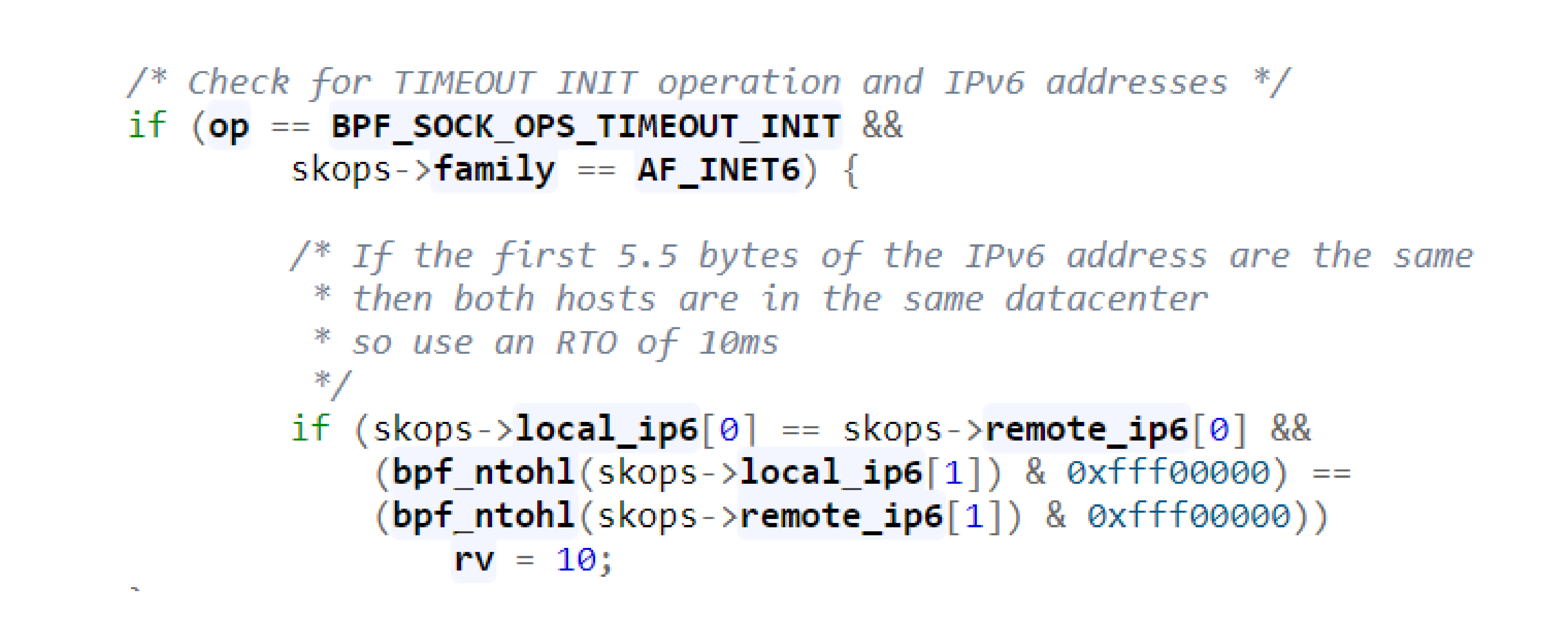
NODE_SPEC := [*TYPE*] *PREFIX* [**tos** *TOS*] [**table** *TABLE_ID*] [**proto** *RTPROTO*] [**scope** *SCOPE*] [**metric** *METRIC*]

INFO_SPEC := *NH OPTIONS FLAGS* [**nexthop** *NH*] ...

NH := [**via** *ADDRESS*] [**dev** *STRING*] [**weight** *NUMBER*] *NHFLAGS*

OPTIONS := *FLAGS* [**mtu** *NUMBER*] [**advmss** *NUMBER*] [**rtt** *TIME*] [**rttvar** *TIME*] [**window** *NUMBER*] [**cwnd** *NUMBER*] [**initcwnd** *NUMBER*] [**ssthresh** *REALM*] [**realms** *REALM*] [**rto_min** *TIME*] [**initrwnd** *NUMBER*]

SYN_RTO is Different

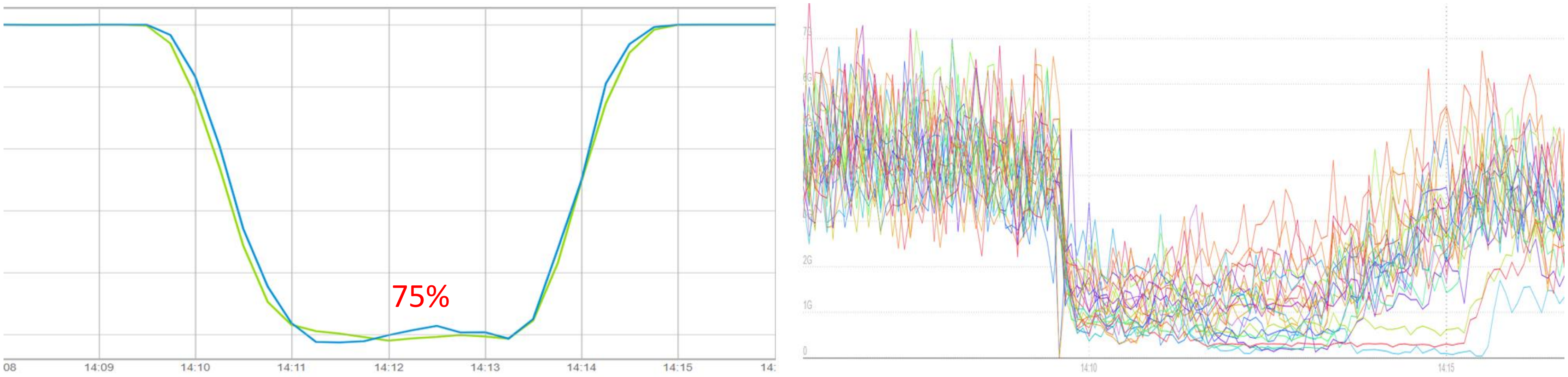


```
/* Check for TIMEOUT INIT operation and IPv6 addresses */
if (op == BPF_SOCK_OPS_TIMEOUT_INIT &&
        skops->family == AF_INET6) {

        /* If the first 5.5 bytes of the IPv6 address are the same
         * then both hosts are in the same datacenter
         * so use an RTO of 10ms
         */
        if (skops->local_ip6[0] == skops->remote_ip6[0] &&
            (bpf_ntohl(skops->local_ip6[1]) & 0xfff00000) ==
            (bpf_ntohl(skops->remote_ip6[1]) & 0xfff00000))
                rv = 10;
```

Changing SYN RTO

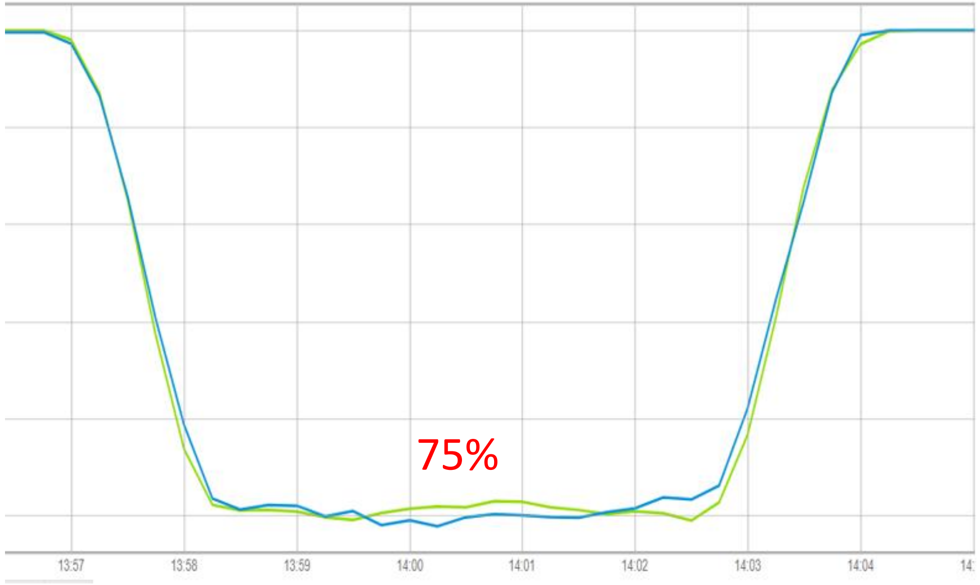
https://elixir.bootlin.com/linux/latest/source/samples/bpf/tcp_synrto_kern.c

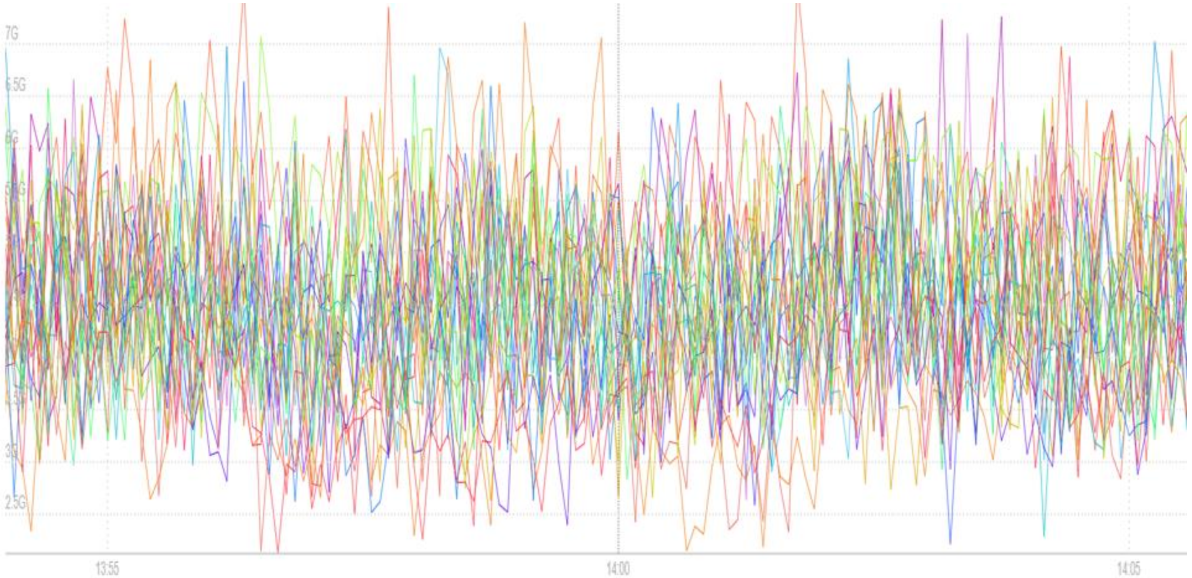
Evaluation: Without Flow Label



One of four ToR uplinks drops packets, significant service degradation

Evaluation: Flow Label + eBPF

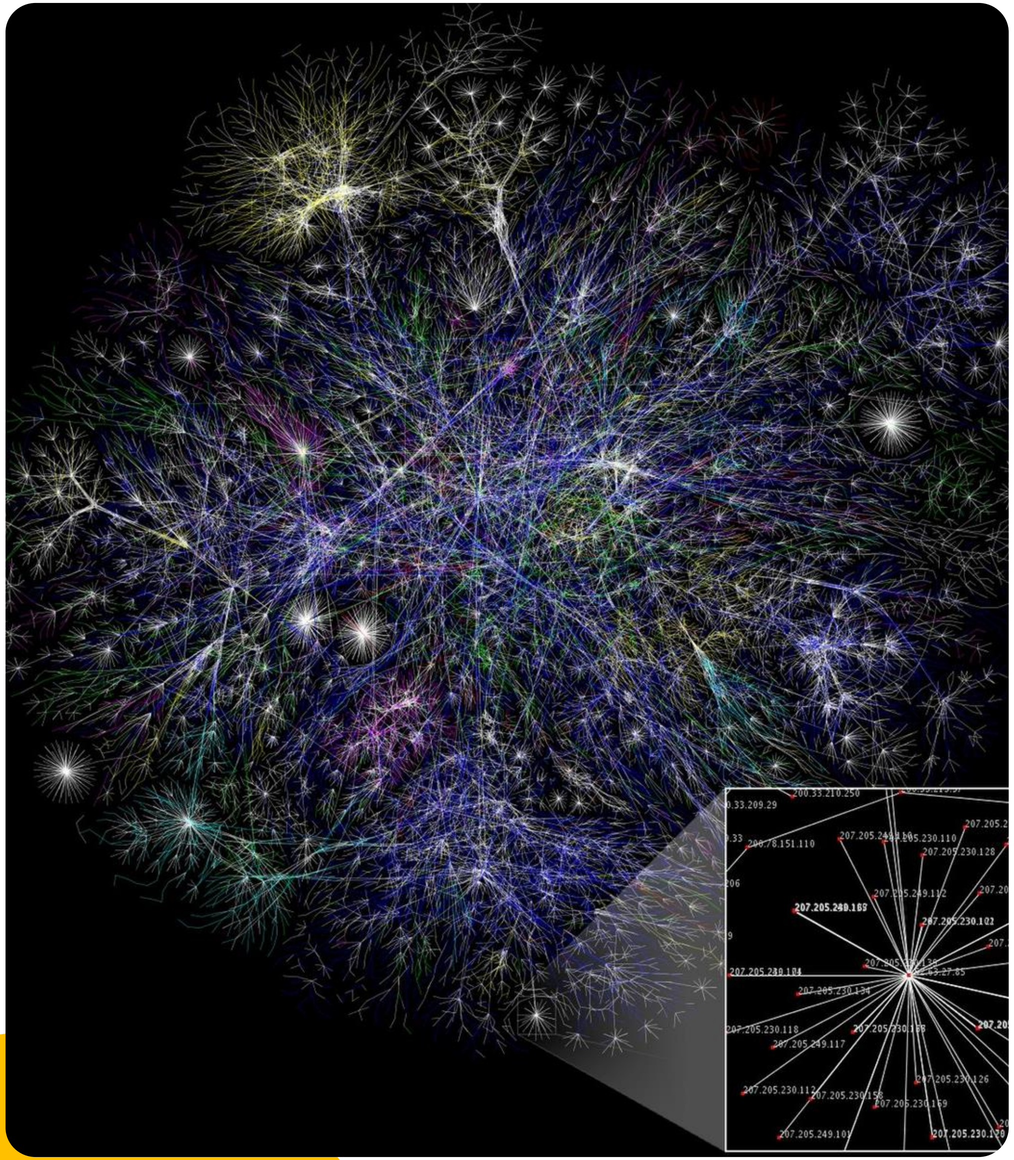




One of four ToR uplink drops packets, no effect on the service!

Self-healing Datacenter: Cookbook

- Does it scale? **Yes!**
- Does it have many paths? **Yes!**
- Does it have fault tolerance? **Use IPv6! Use flow label!**
- How do I change RTO? **eBPF is the answer!**
- Without documentation!



Theory Internet: Many-Many Paths

Multihomed at the edge;

Multiple connections between peers;

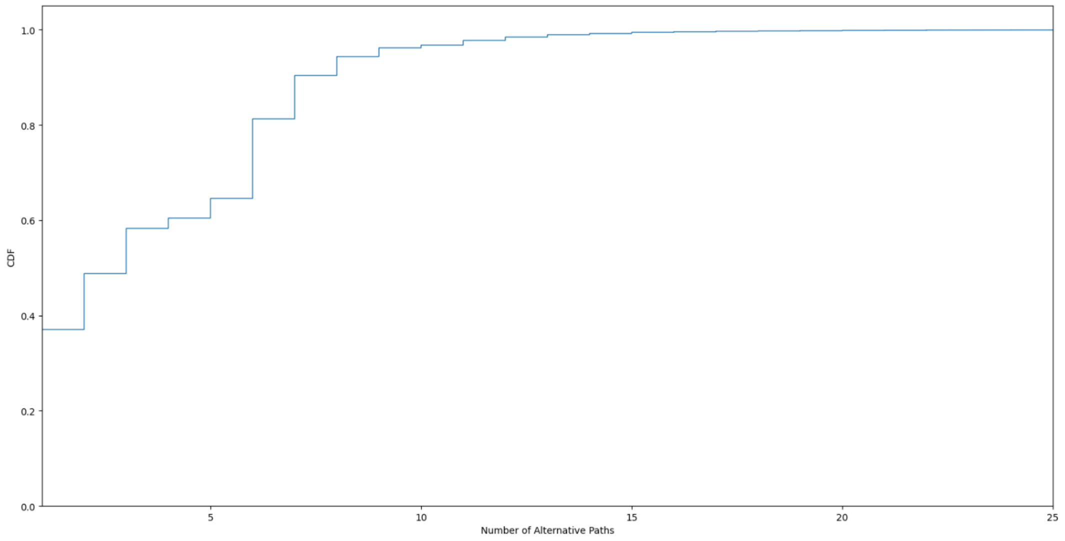
Multiple connection with upstreams;

Real Internet: Many-Many Paths

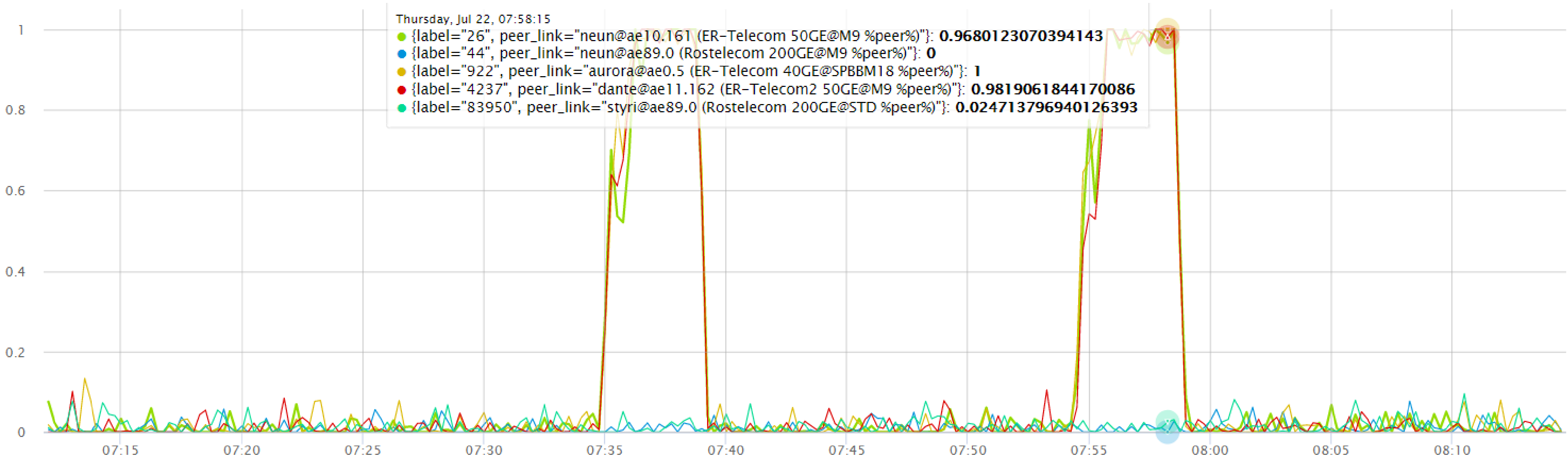
Average number of best paths: 3.8

Maximum number of best paths: 44

>60% of prefixes have more then 1 path



A Real Outage



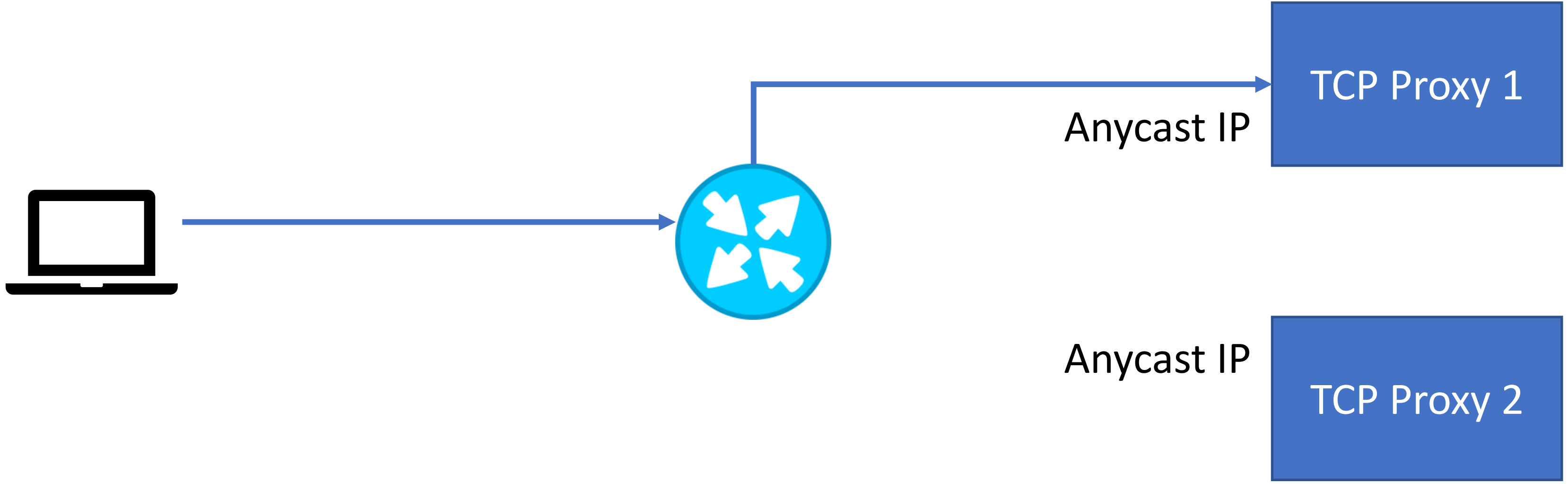
Thursday, Jul 22, 07:58:15

- ● {label="26", peer_link="neun@ae10.161 (ER-Telecom 50GE@M9 %peer%)"}: **0.9680123070394143**
- ● {label="44", peer_link="neun@ae89.0 (Rostelecom 200GE@M9 %peer%)"}: **0**
- ● {label="922", peer_link="aurora@ae0.5 (ER-Telecom 40GE@SPBBM18 %peer%)"}: **1**
- ● {label="4237", peer_link="dante@ae11.162 (ER-Telecom2 50GE@M9 %peer%)"}: **0.9819061844170086**
- ● {label="83950", peer_link="styri@ae89.0 (Rostelecom 200GE@STD %peer%)"}: **0.02471379694012639 3**

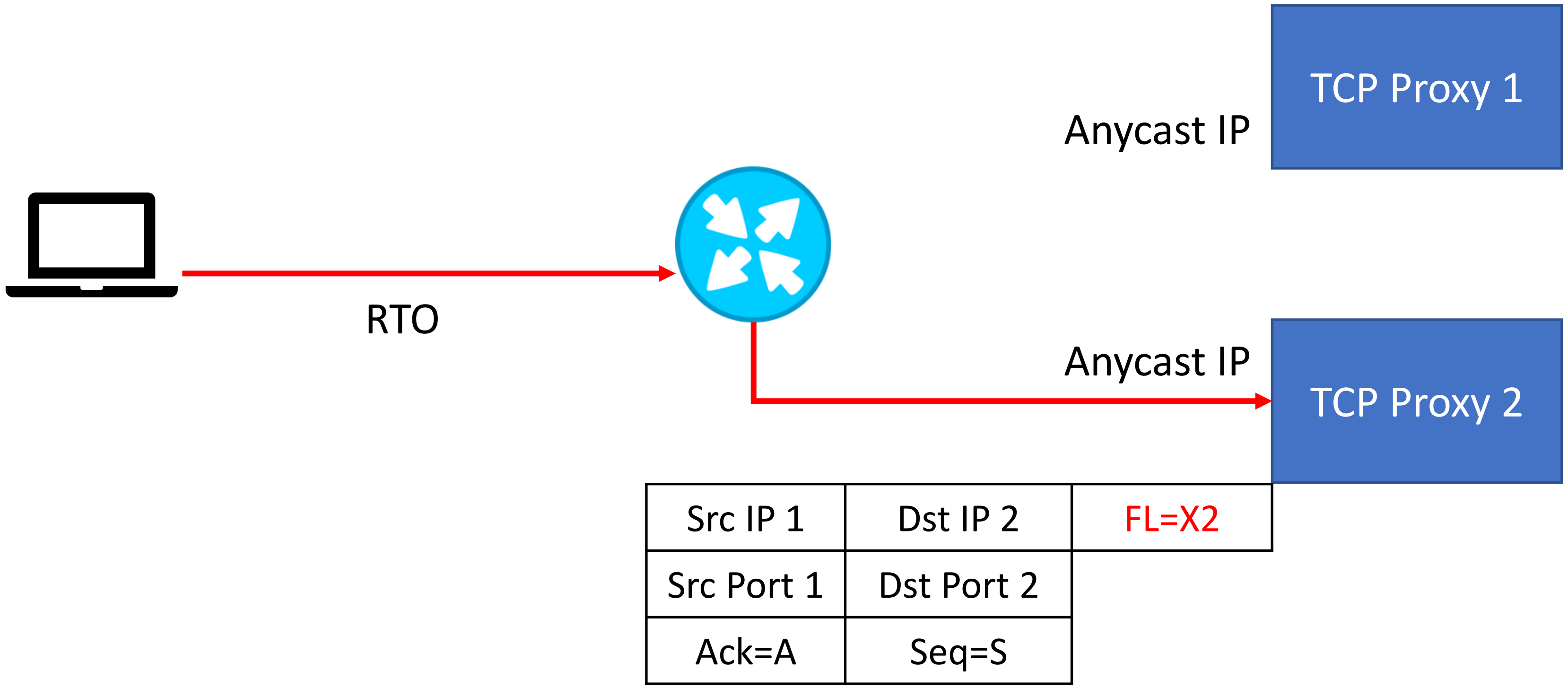
IT'S AN ANYCAST
imgflip.com

RTO & Anycast

Src IP 1	Dst IP 2	FL=X1
Src Port 1	Dst Port 2	
Ack=A	Seq=S	



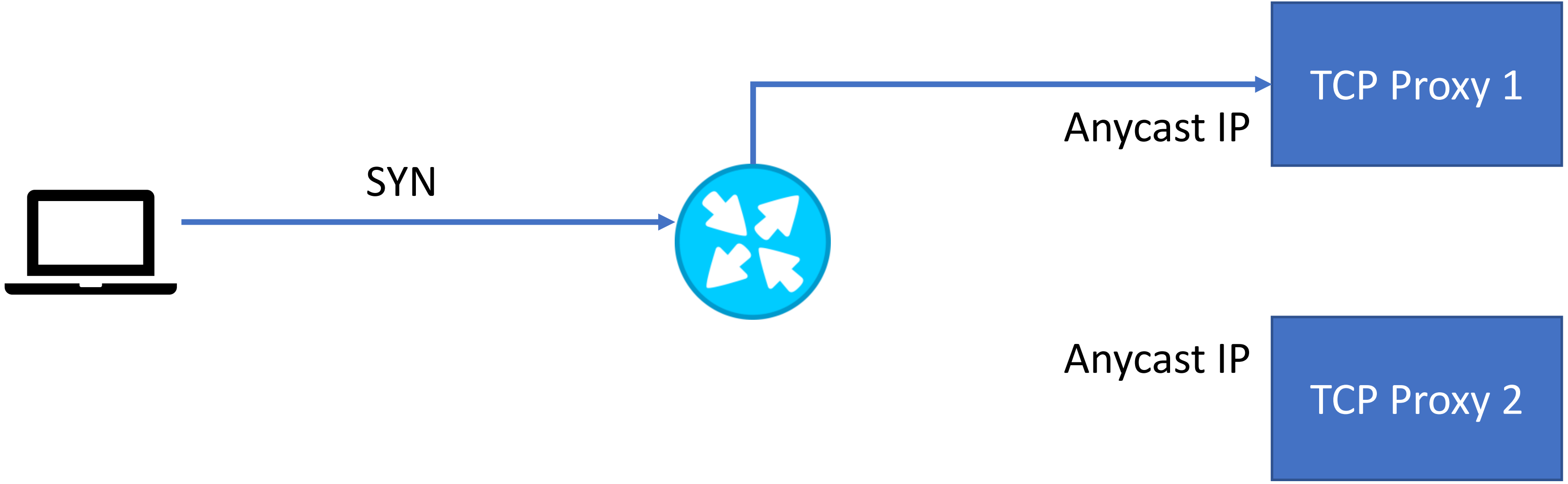
RTO & Anycast



Src IP 1	Dst IP 2	FL=X2
Src Port 1	Dst Port 2	
Ack=A	Seq=S	

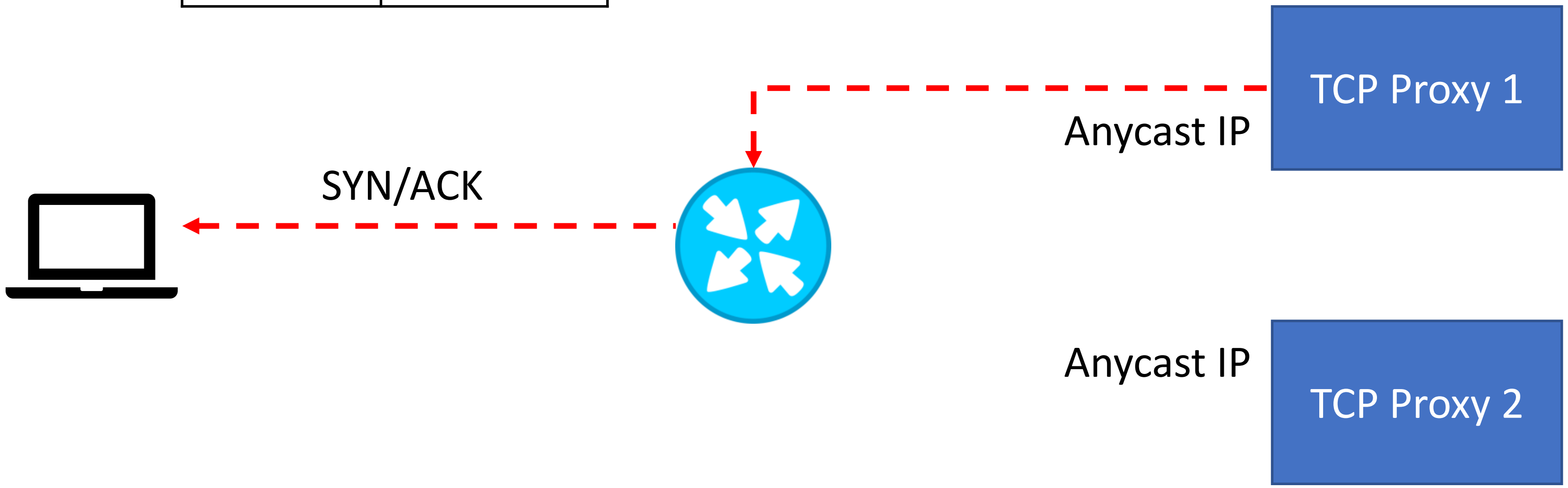
SYN RTO & Anycast

Src IP 1	Dst IP 2	FL=X1
Src Port 1	Dst Port 2	
Ack=0	Seq=S1	



SYN RTO & Anycast

Src IP 2	Dst IP 1	FL=Y1
Src Port 2	Dst Port 1	
Ack=S1+1	Seq=S2	



SYN/ACK

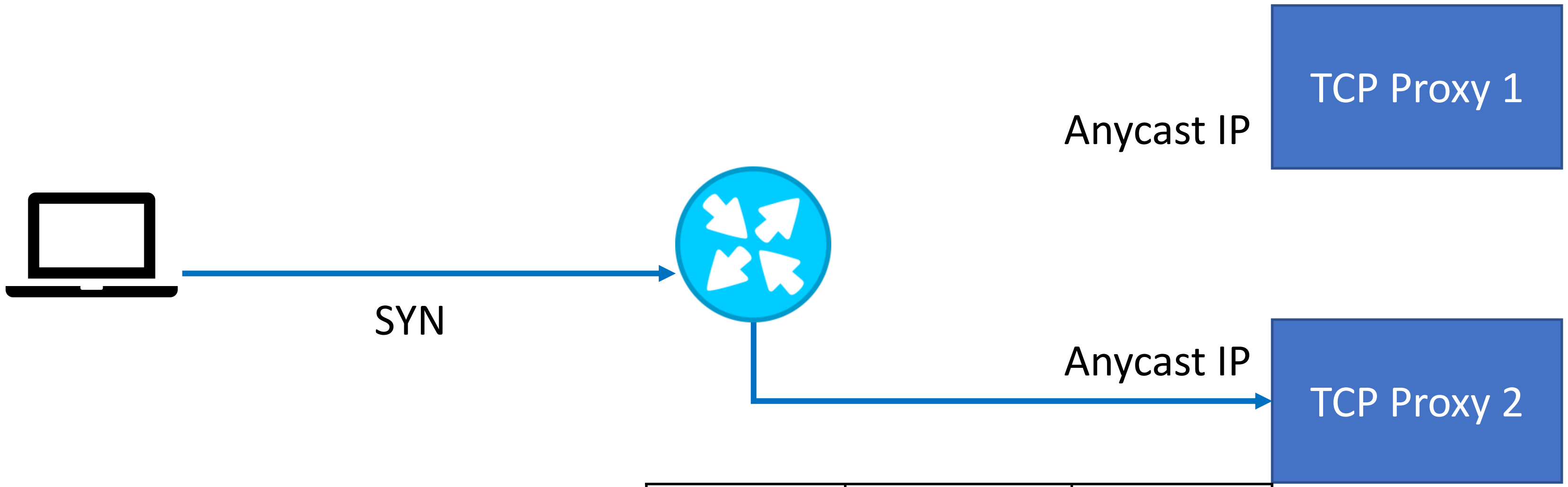
Anycast IP

TCP Proxy 1

Anycast IP

TCP Proxy 2

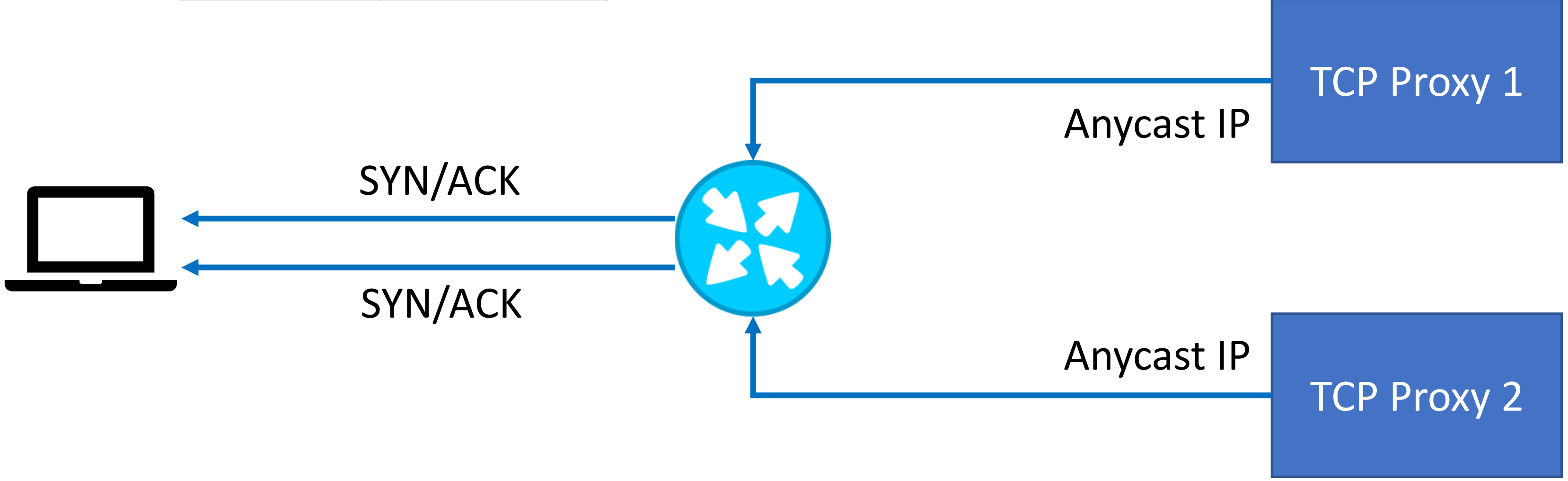
SYN RTO & Anycast



Src IP 1	Dst IP 2	FL=X2
Src Port 1	Dst Port 2	
Ack=0	Seq=S1	

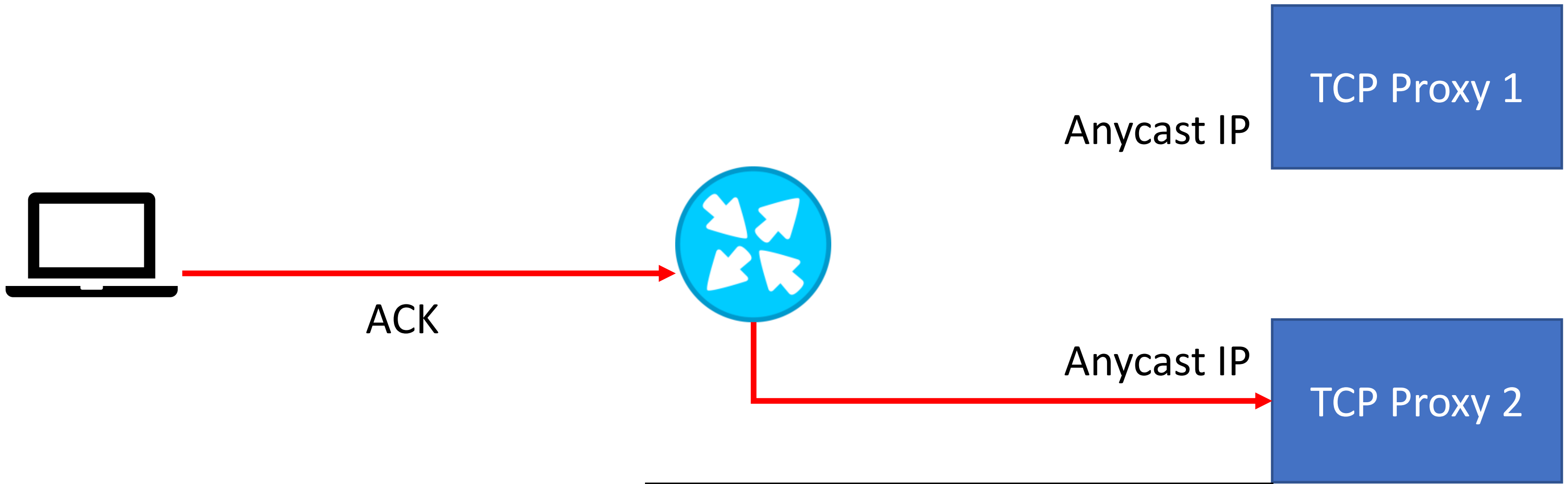
SYN RTO & Anycast

Src IP 2	Dst IP 1	FL=Y1
Src Port 2	Dst Port 1	
Ack=S1+1	Seq=S2	



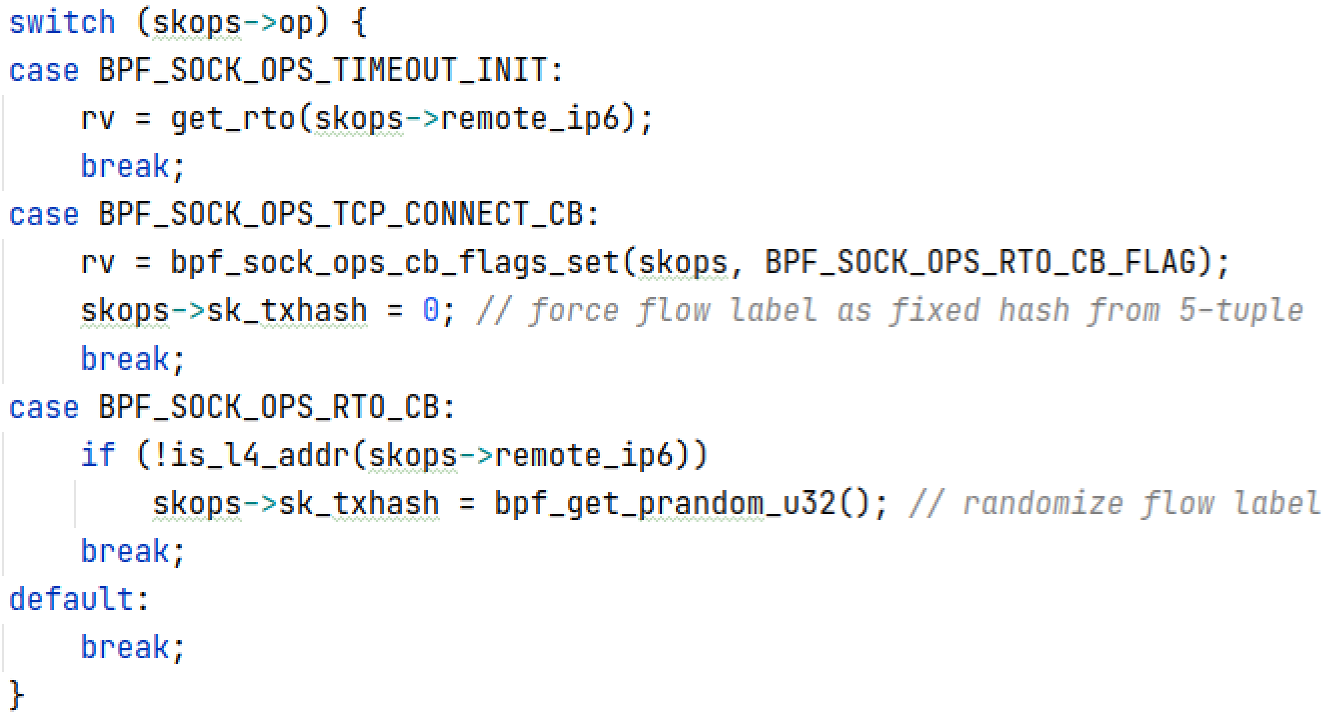
Src IP 2	Dst IP 1	FL=Z1
Src Port 2	Dst Port 1	
Ack=S1+1	Seq=S3	

SYN RTO & Anycast



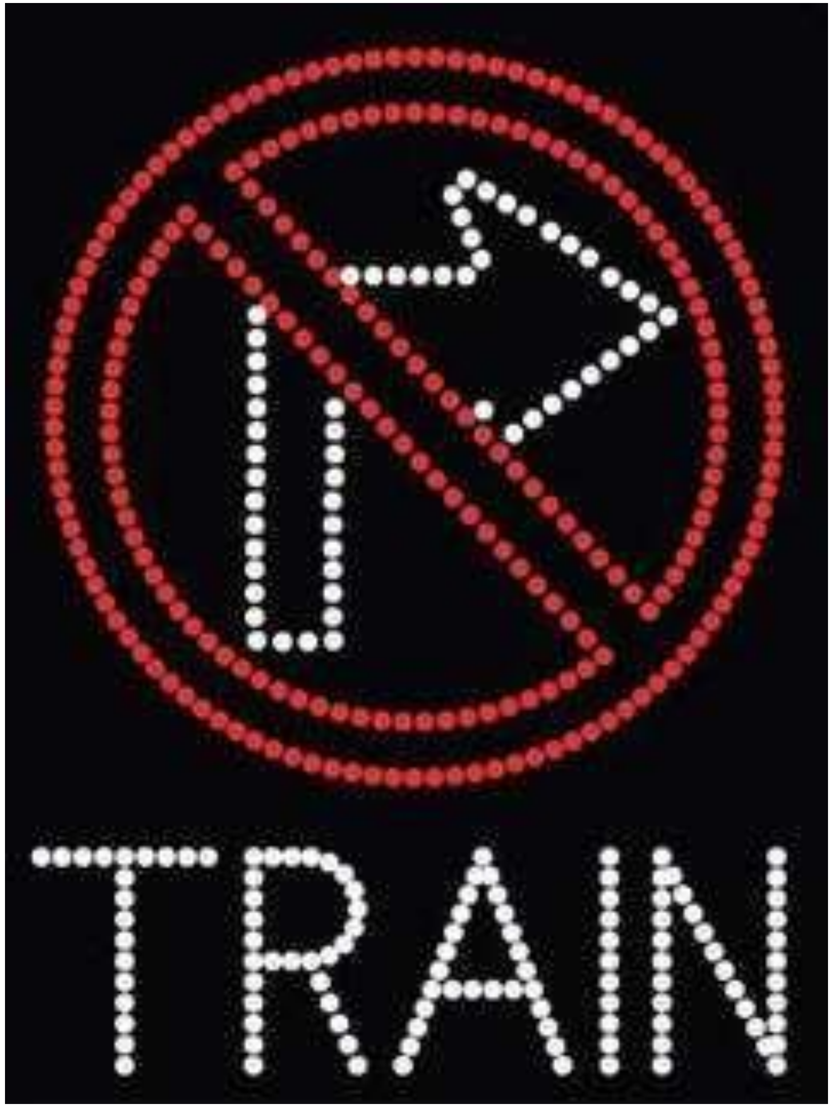
Src IP 1	Dst IP 2	FL=X2
Src Port 1	Dst Port 2	
Ack=S2 + 1	Seq=S1 + 1	

Temporary Workaround



```c
switch (skops->op) {
case BPF_SOCK_OPS_TIMEOUT_INIT:
    rv = get_rto(skops->remote_ip6);
    break;
case BPF_SOCK_OPS_TCP_CONNECT_CB:
    rv = bpf_sock_ops_cb_flags_set(skops, BPF_SOCK_OPS_RTO_CB_FLAG);
    skops->sk_txhash = 0; // force flow label as fixed hash from 5-tuple
    break;
case BPF_SOCK_OPS_RTO_CB:
    if (!is_l4_addr(skops->remote_ip6))
        skops->sk_txhash = bpf_get_prandom_u32(); // randomize flow label
    break;
default:
    break;
}
```

SRv6



Toward Correct Solution

- Holy war against ‘state’ at anycast services (L4 balancers!);
- Changing TCP behavior to safe mode;
- Keeping the knobs, we love knobs!

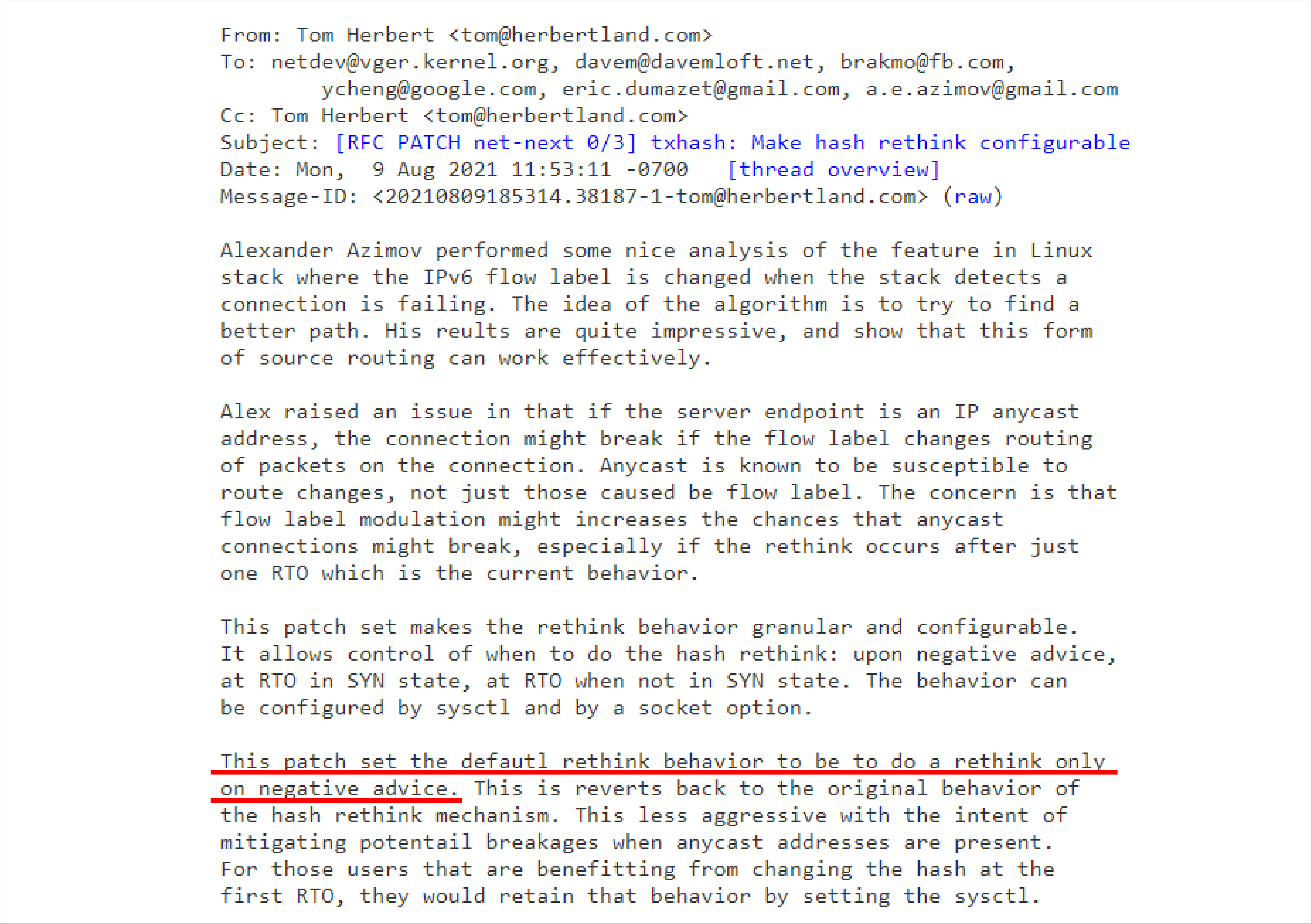
Flow Label: Safe Mode

Client – sends SYN, Server – responds with SYN&ACK

- In case of SYN_RTO or RTO events Server SHOULD recalculate its TCP socket hash, thus change Flow Label. This behavior MAY be switched on by default;
- In case of SYN_RTO or RTO events Client MAY recalculate its TCP socket hash, thus change Flow Label. This behavior MUST be switched off by default;

Linux Kernel

2021



From: Tom Herbert <tom@herbertland.com>
To: netdev@vger.kernel.org, davem@davemloft.net, brakmo@fb.com,
        ycheng@google.com, eric.dumazet@gmail.com, a.e.azimov@gmail.com
Cc: Tom Herbert <tom@herbertland.com>
Subject: [RFC PATCH net-next 0/3] txhash: Make hash rethink configurable
Date: Mon, 9 Aug 2021 11:53:11 -0700 [thread overview]
Message-ID: <20210809185314.38187-1-tom@herbertland.com> (raw)

Alexander Azimov performed some nice analysis of the feature in Linux stack where the IPv6 flow label is changed when the stack detects a connection is failing. The idea of the algorithm is to try to find a better path. His reults are quite impressive, and show that this form of source routing can work effectively.

Alex raised an issue in that if the server endpoint is an IP anycast address, the connection might break if the flow label changes routing of packets on the connection. Anycast is known to be susceptible to route changes, not just those caused be flow label. The concern is that flow label modulation might increases the chances that anycast connections might break, especially if the rethink occurs after just one RTO which is the current behavior.

This patch set makes the rethink behavior granular and configurable. It allows control of when to do the hash rethink: upon negative advice, at RTO in SYN state, at RTO when not in SYN state. The behavior can be configured by sysctl and by a socket option.

This patch set the defautl rethink behavior to be to do a rethink only on negative advice. This is reverts back to the original behavior of the hash rethink mechanism. This less aggressive with the intent of mitigating potentail breakages when anycast addresses are present. For those users that are benefitting from changing the hash at the first RTO, they would retain that behavior by setting the sysctl.

Self-healing ~~Datacenter~~ TCP: Cookbook

- Flow label provides is a way to ‘jump’ from a failing path;
- Already works in controlled environment;
- Can disrupt TCP connection with stateful anycast services;
- We need to change Linux defaults!
- This time we need to document it!