

Self-healing Network or The Magic of Flow Label

Alexander Azimov mitradir@yandex-team.ru



Yandex.Music

AI-powered streaming platform
serving over 20 million users in
12 countries



Yandex.Direct

Ad platform serving 53% of the
Russian digital advertising
market



Yandex Zen

Personalized, AI-powered content
feed with 40 million monthly
users



Yandex.Taxi

Ride-hailing in 17 countries with
over 1 billion rides



Alice

Russian-speaking intelligent
assistant with 35 million monthly
users

Yandex

Facts and figures from some of the intelligent products and services in the Yandex ecosystem



Yandex.Drive

Carsharing with a fleet of 11,500
vehicles



Yandex.Eats

Food delivery from over 10,000
restaurants across 30+ cities



Search

56.3% search share in Russia
67.7% desktop, 50% mobile

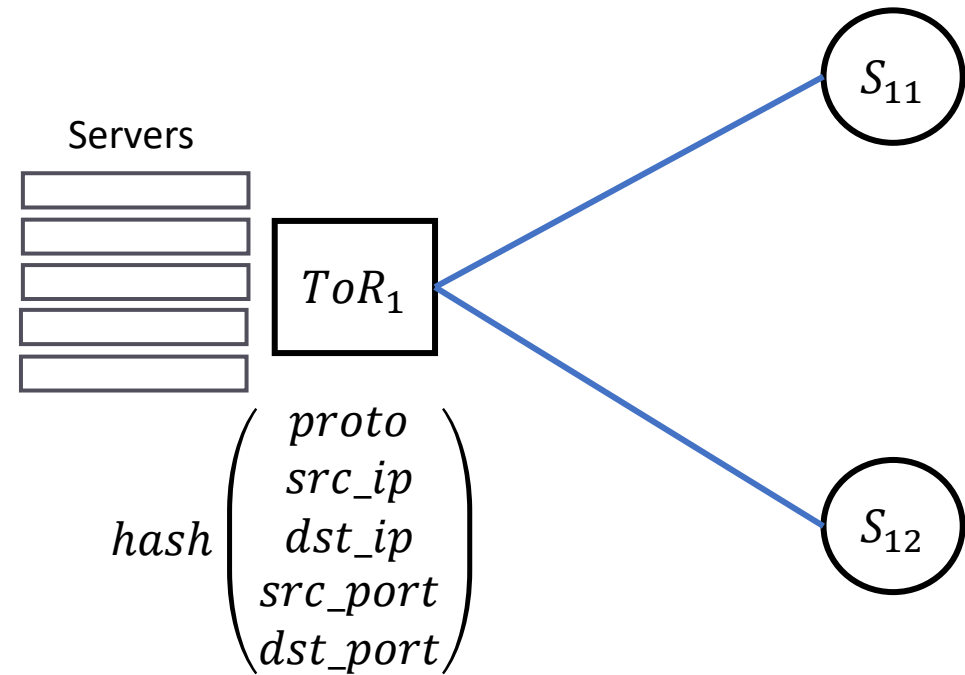


Self-Driving Car

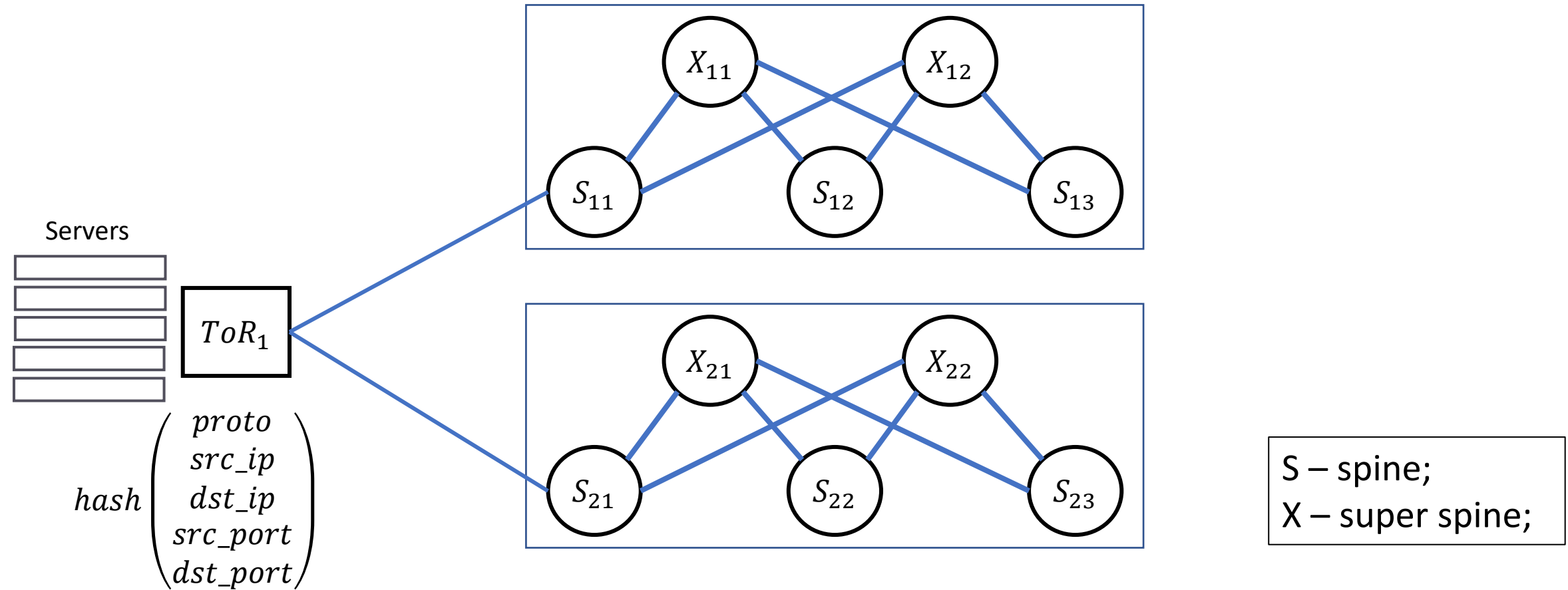
Tested on public roads in Russia,
Israel, and the US
Robotaxi service with 3,000+ rides

All figures as of August 2019

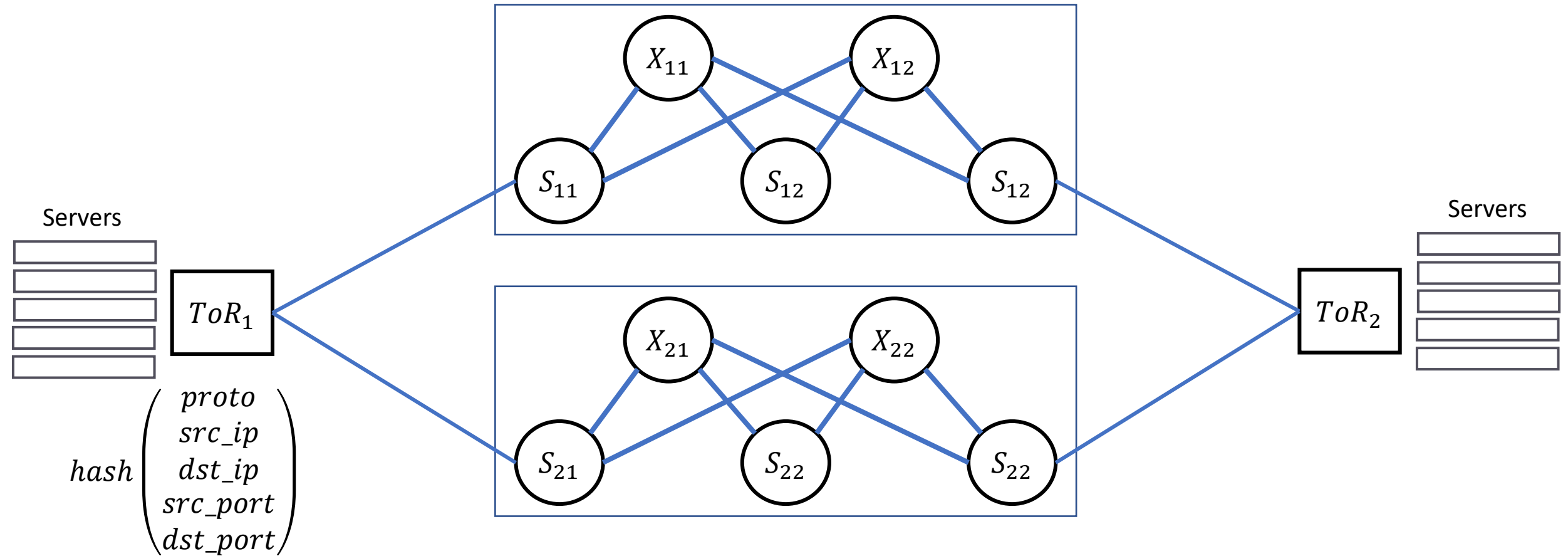
Just a Top of Rack Switch (ToR)



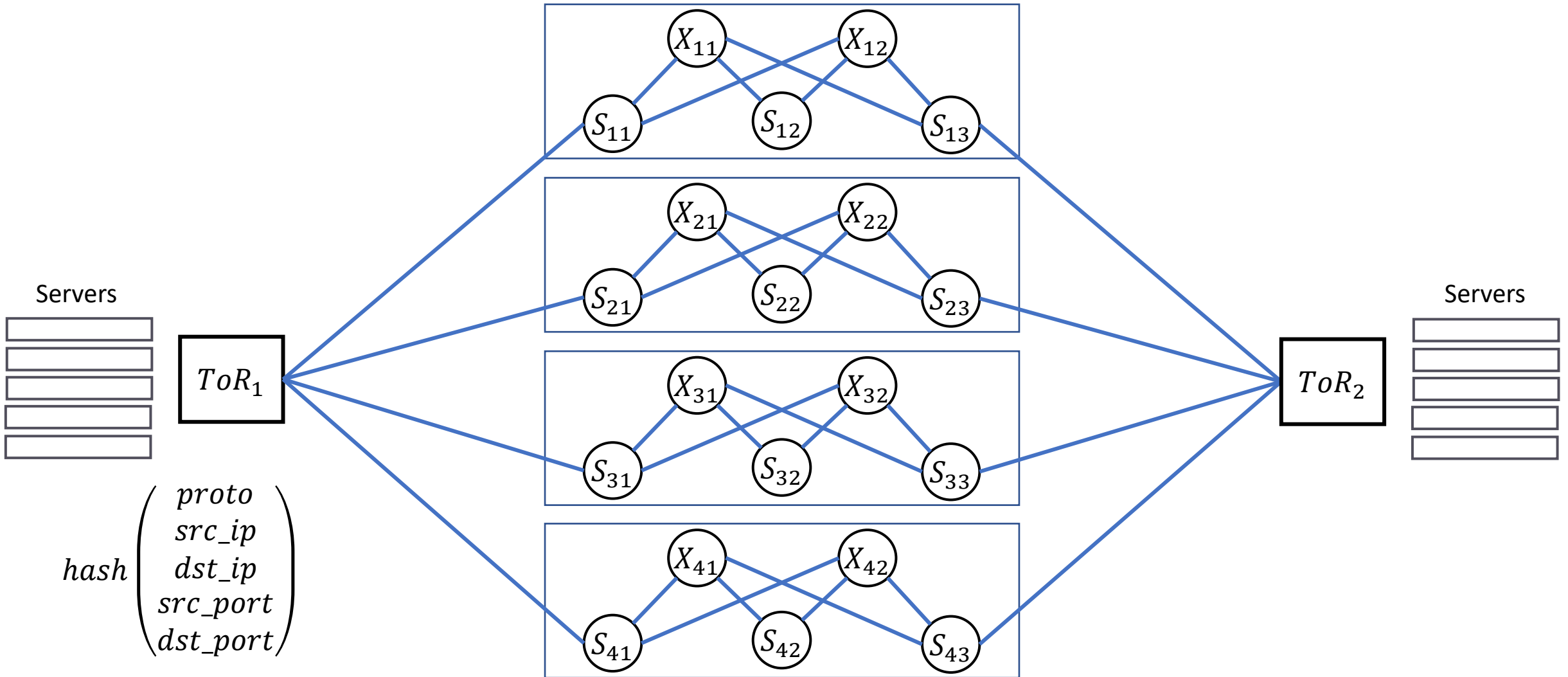
ToR + 2xPlanes



ToR + 2xPlanes + ToR



ToR + 4xPlanes + ToR



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: $N_PLANES \times N_X_SPINES$

Many-Many Paths

N_PLANES: Number of planes in DC; (8)

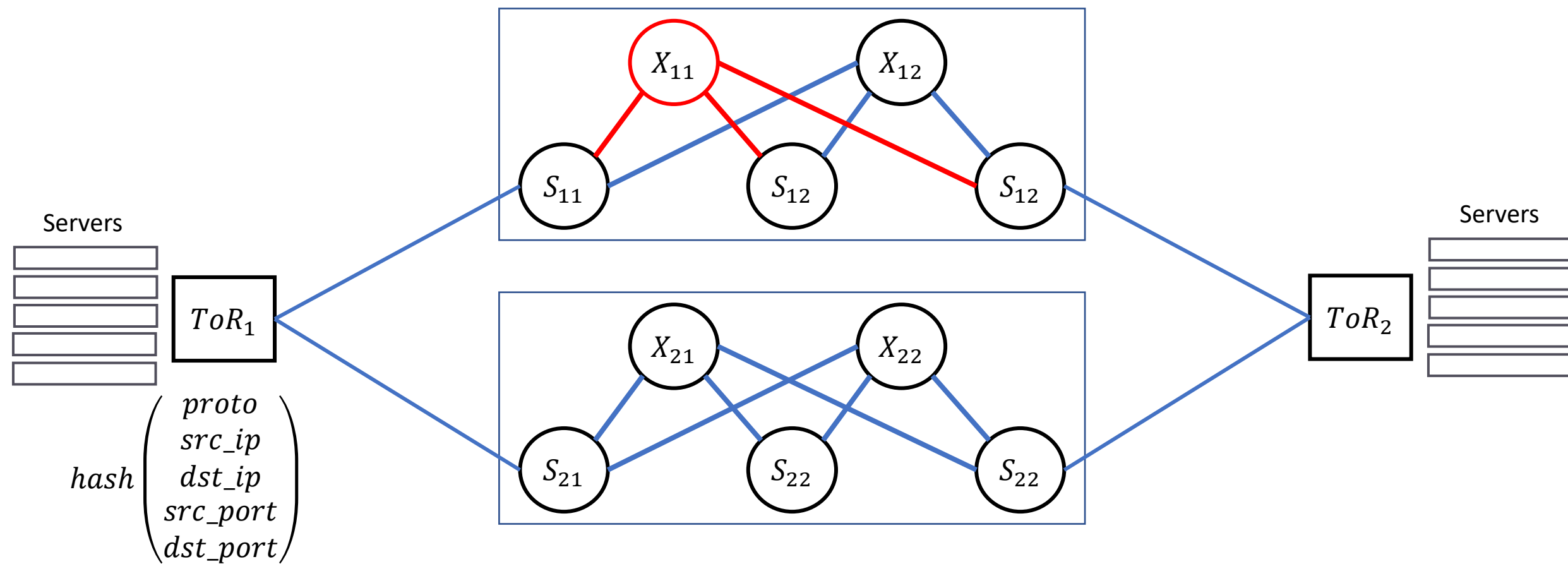
N_X_SPINES: Number of super spines (X) in each plane; (32)

- Inside ToR: 1
- Inside Pod: $N_PLANES = 8$
- Between Pods: $N_PLANES \times N_X_SPINES = 256$

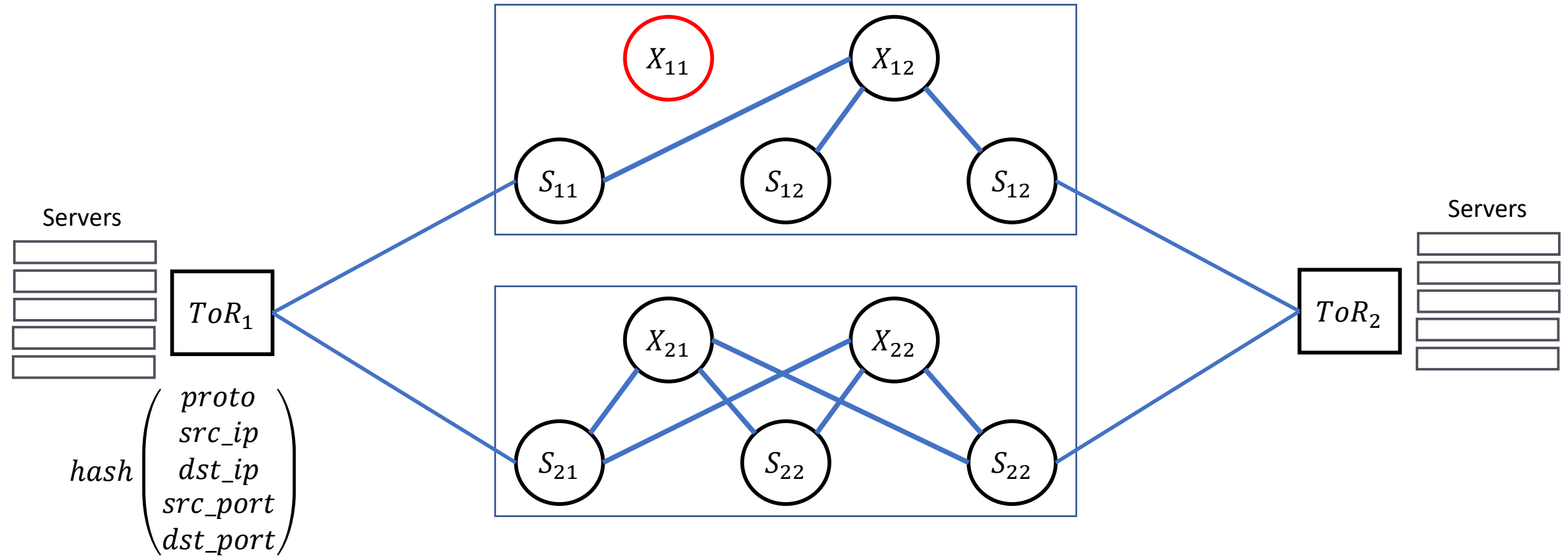
Self-healing Datacenter: Cookbook

- Does it scale? **Yes!**
- Does it have many paths? **Yes!**
- Does it have fault tolerance?

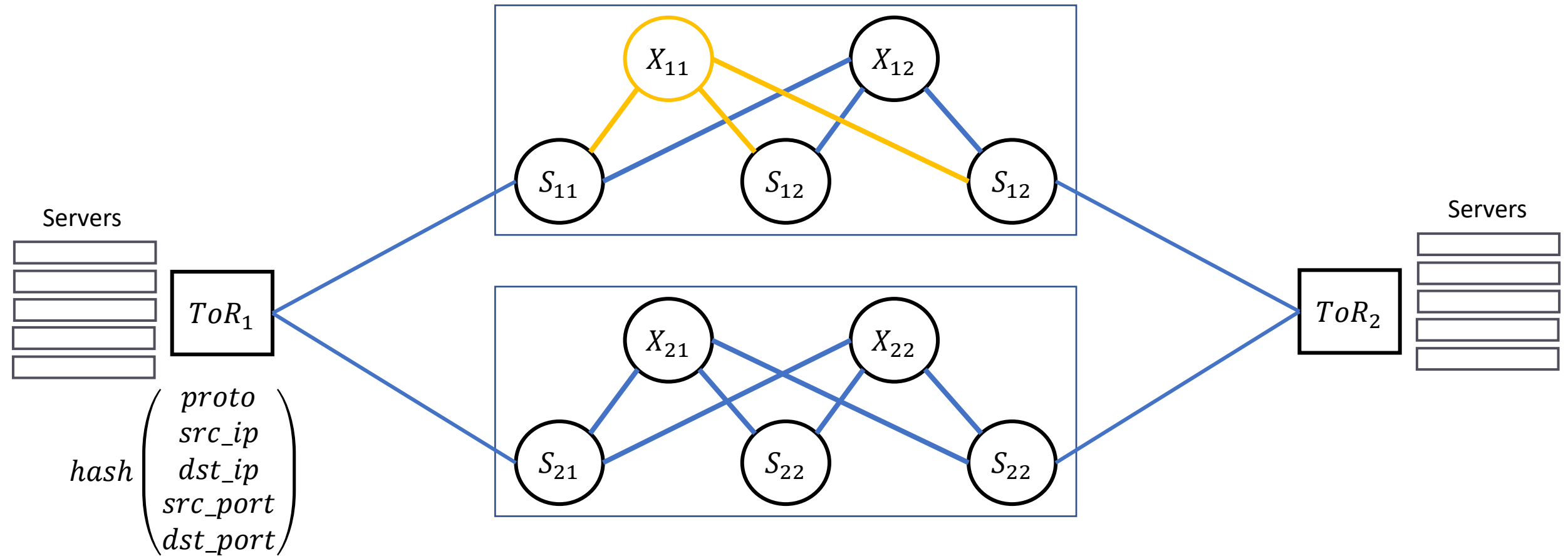
X_{11} is Broken



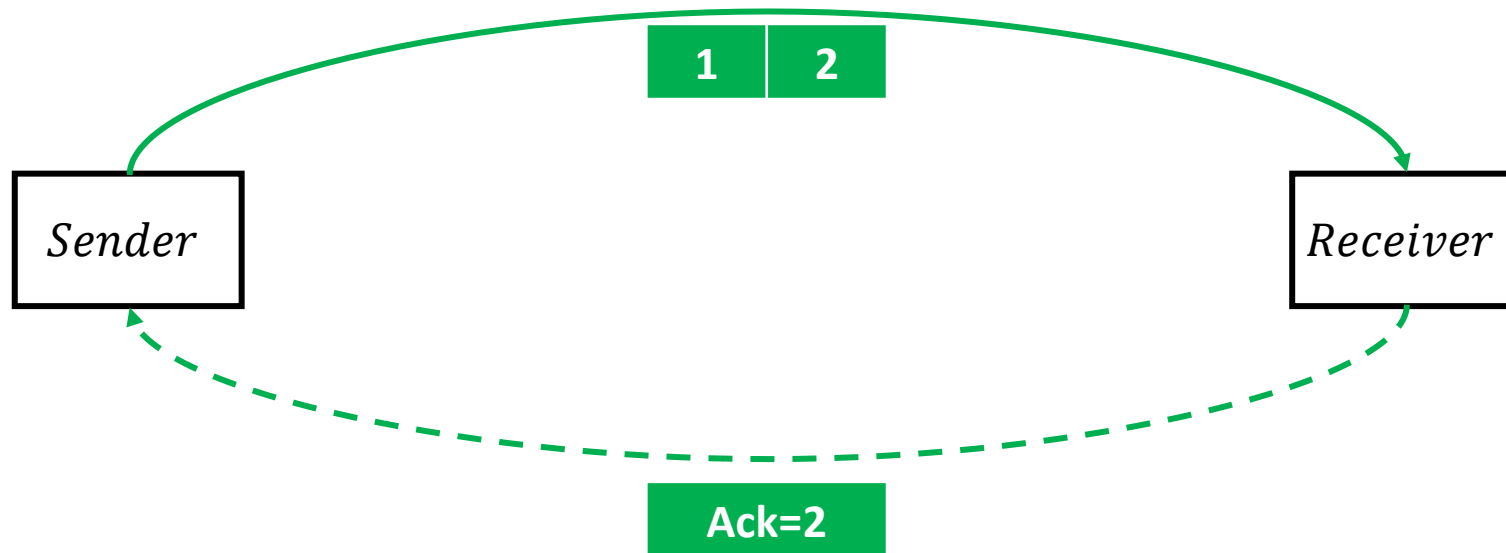
X_{11} is Broken: No link, No Problem



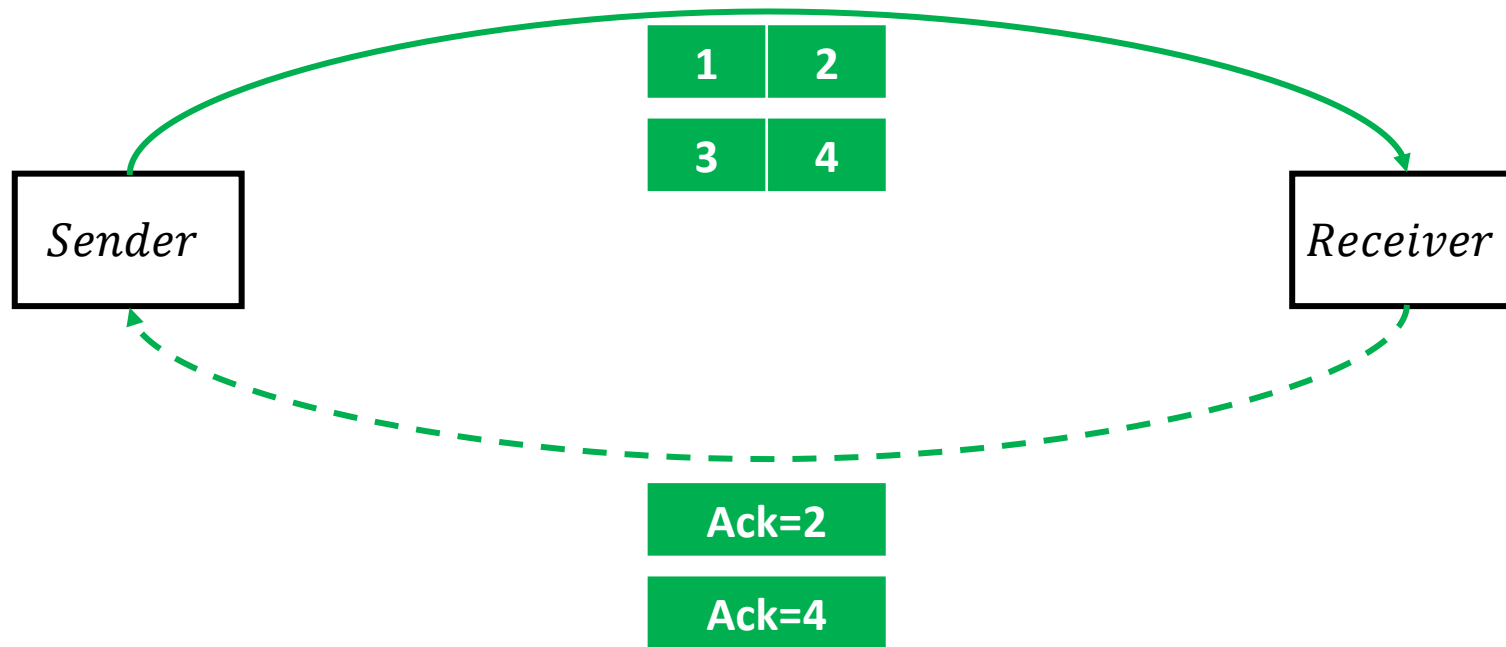
X_{11} is Broken: Constant Loss



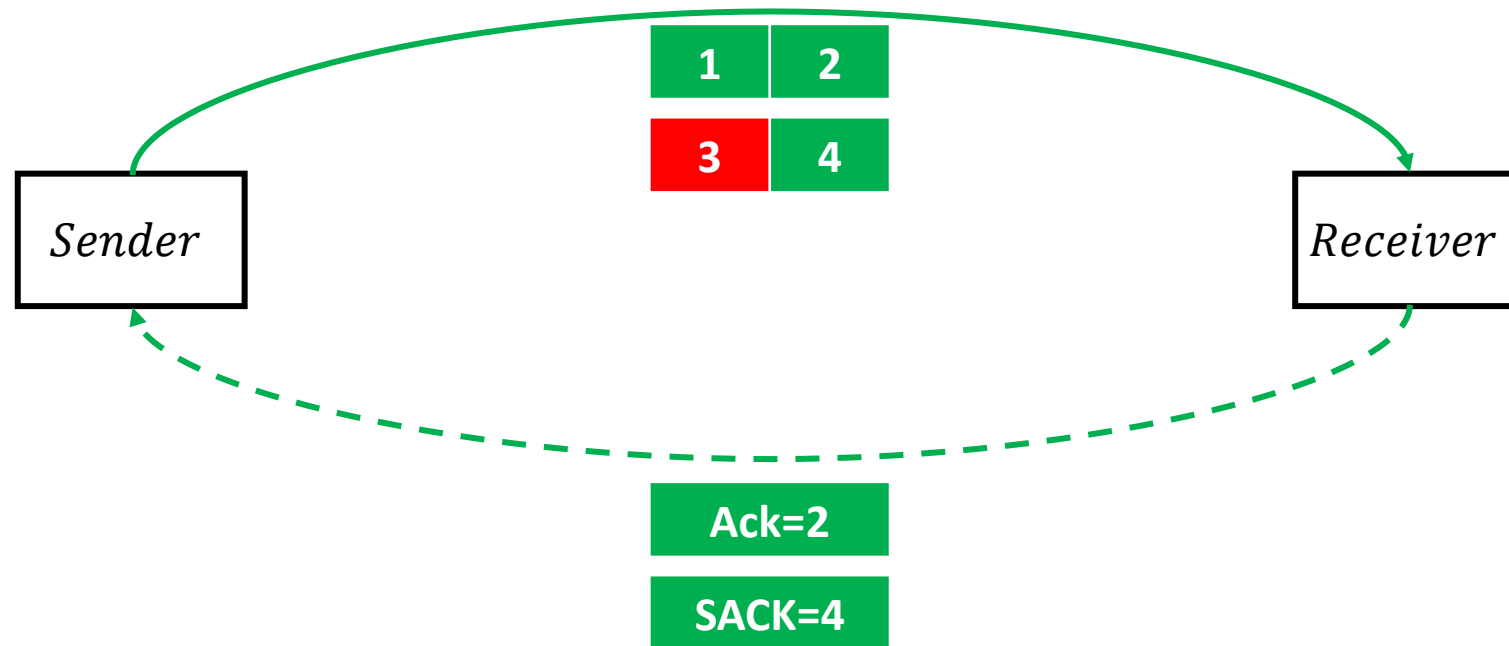
TCP Acknowledgment



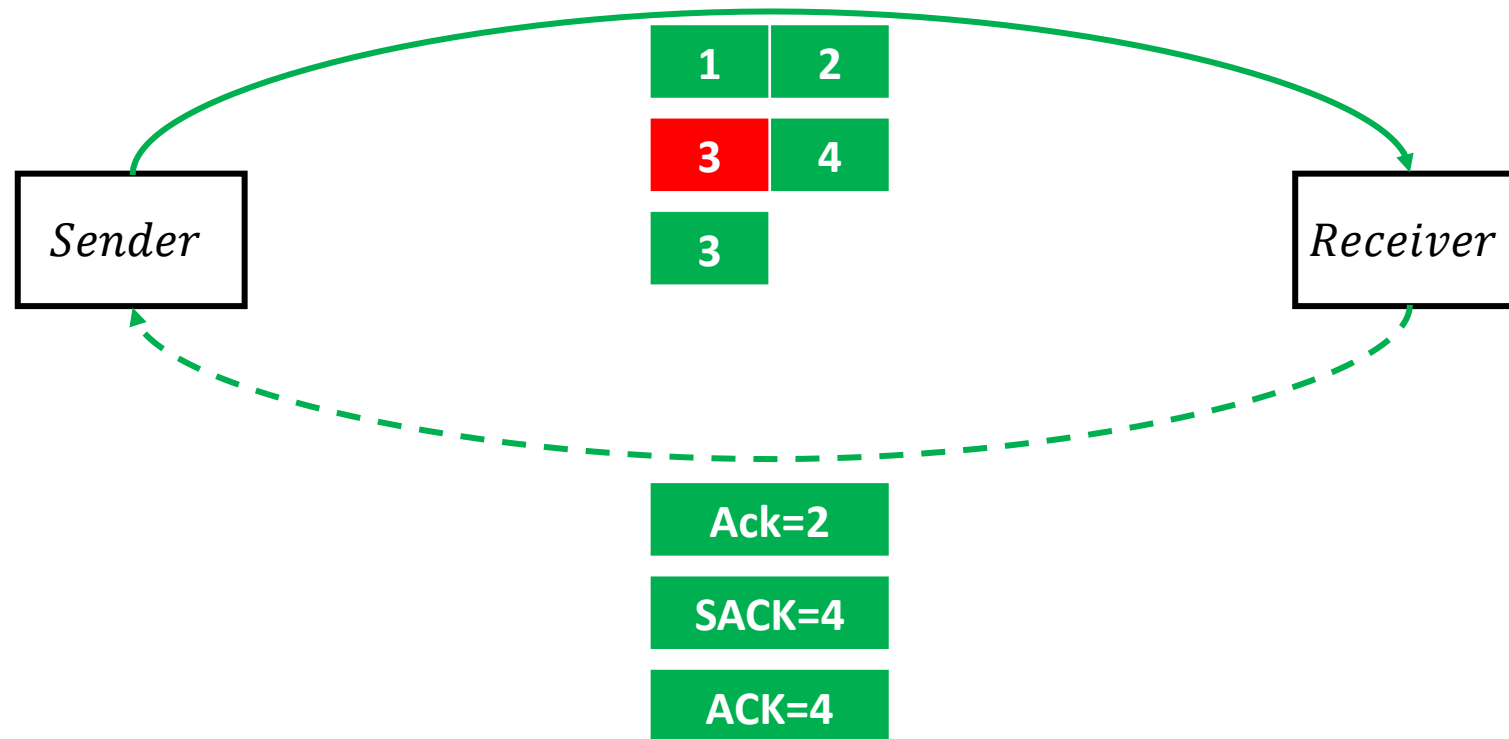
TCP Acknowledgment



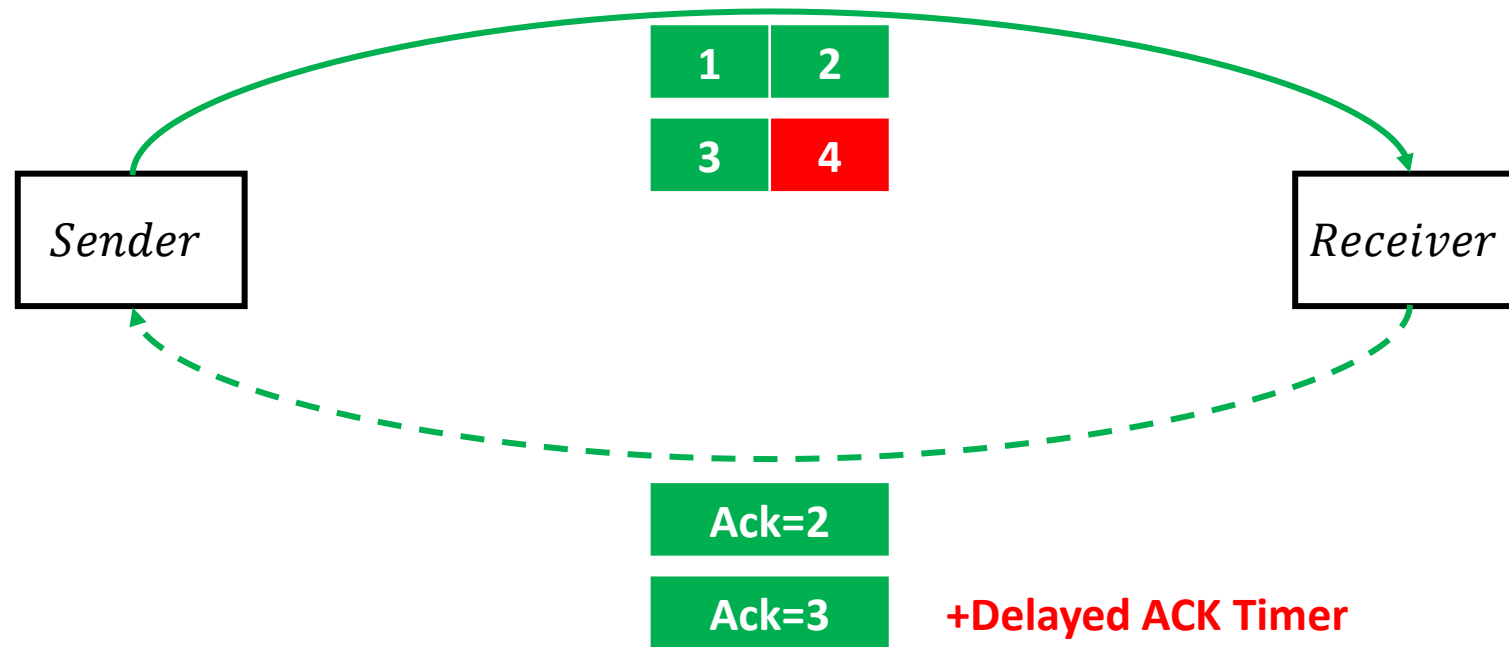
TCP Retransmits: SACK



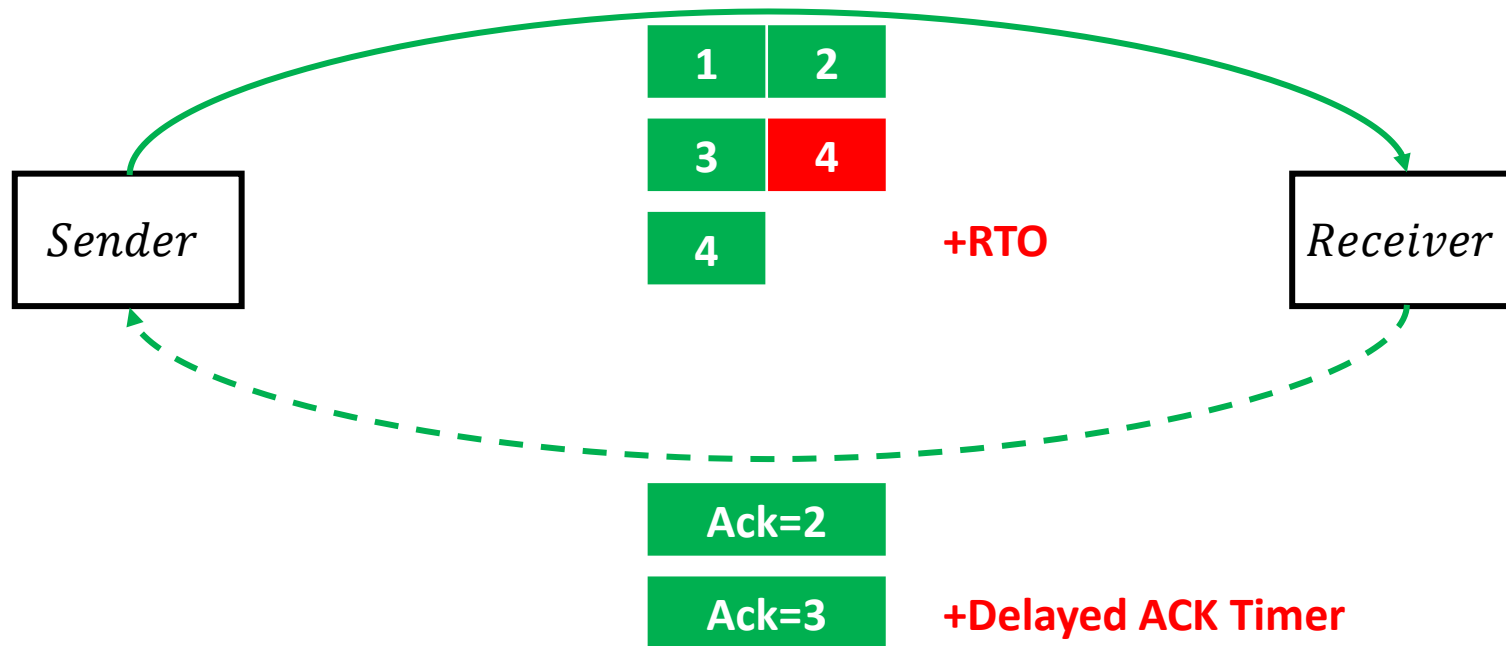
TCP Retransmits: SACK



TCP Retransmits: RTO

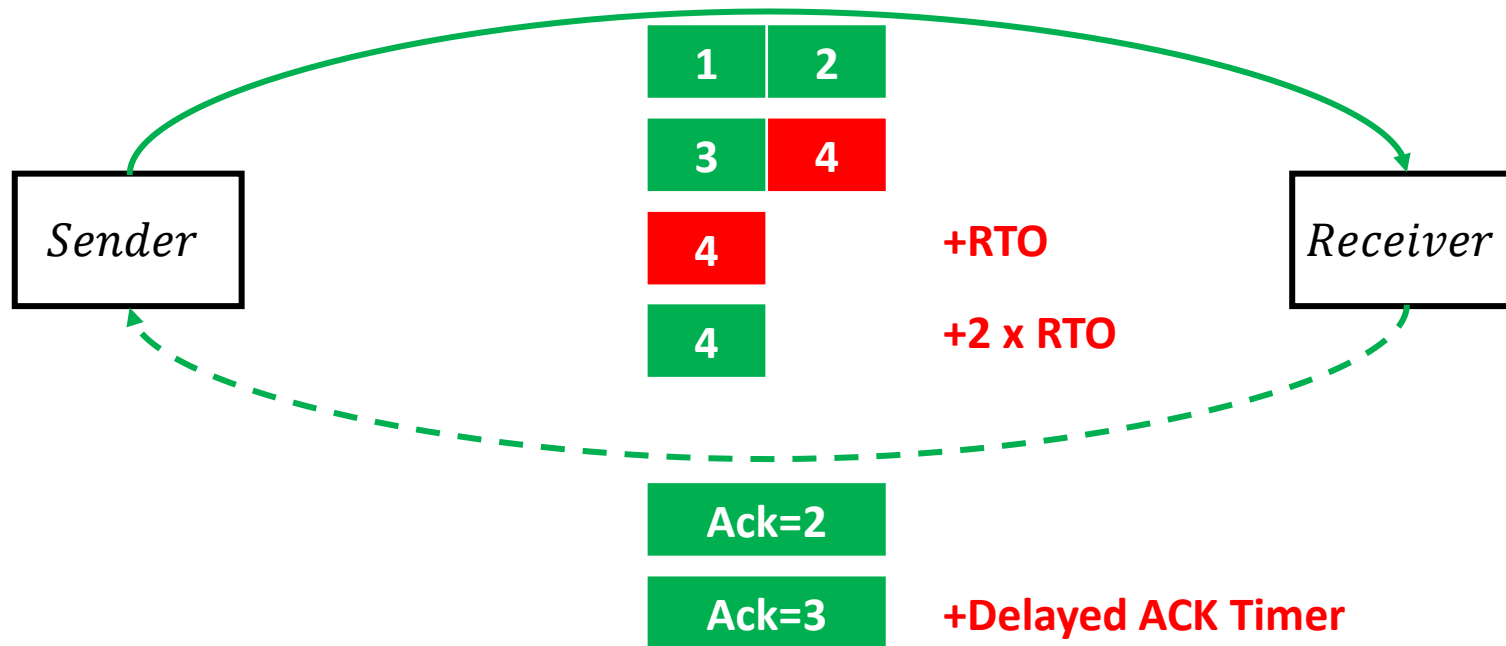


TCP Retransmits: RTO



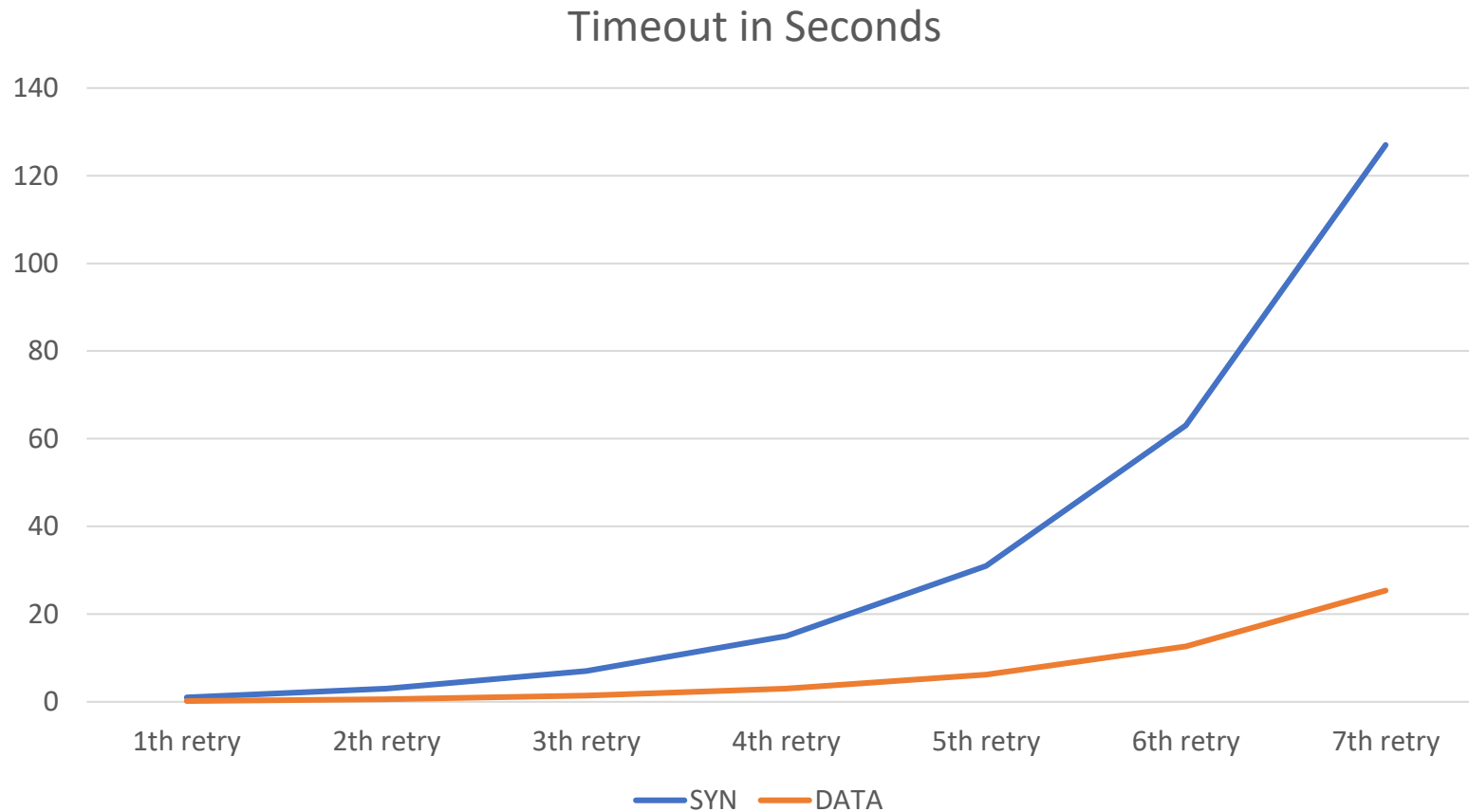
$$RTO = \text{MAX}(RTO_MIN, RTT)$$

TCP Retransmits: RTO



$$RTO = \text{MAX}(RTO_MIN, RTT)$$

RTO & SYN_RTO Timeouts

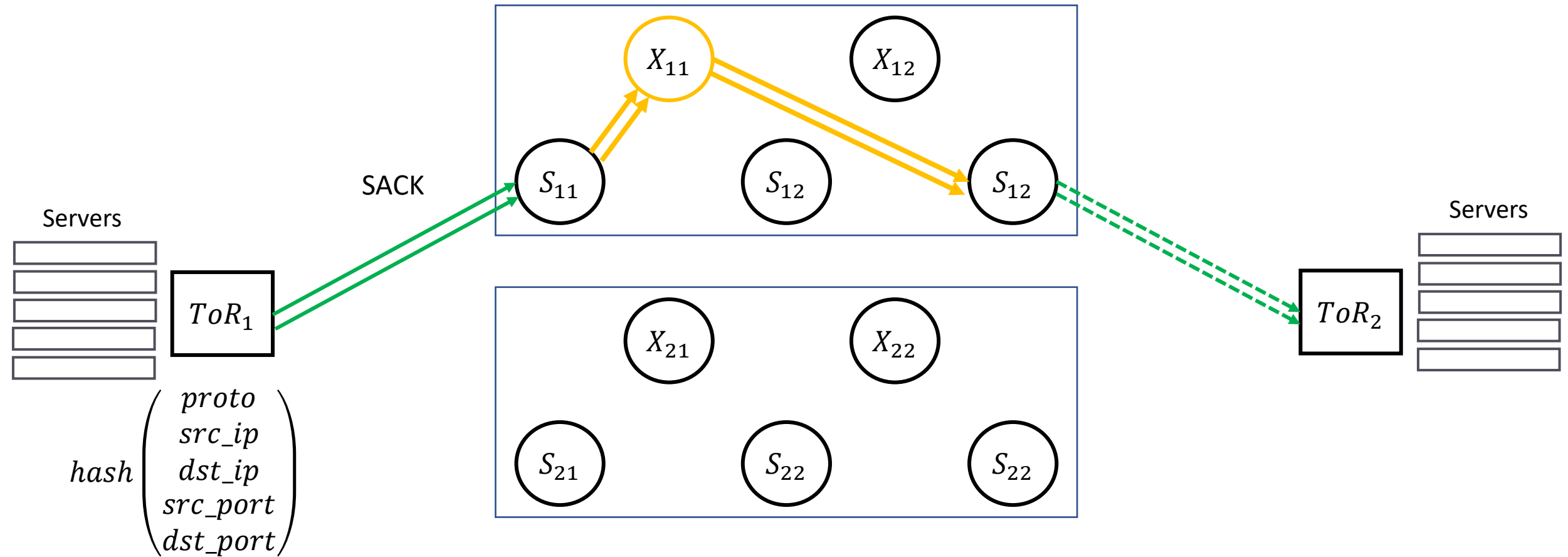


$$\text{RTO} = \text{MAX}(\text{RTO_MIN}, \text{RTT})$$

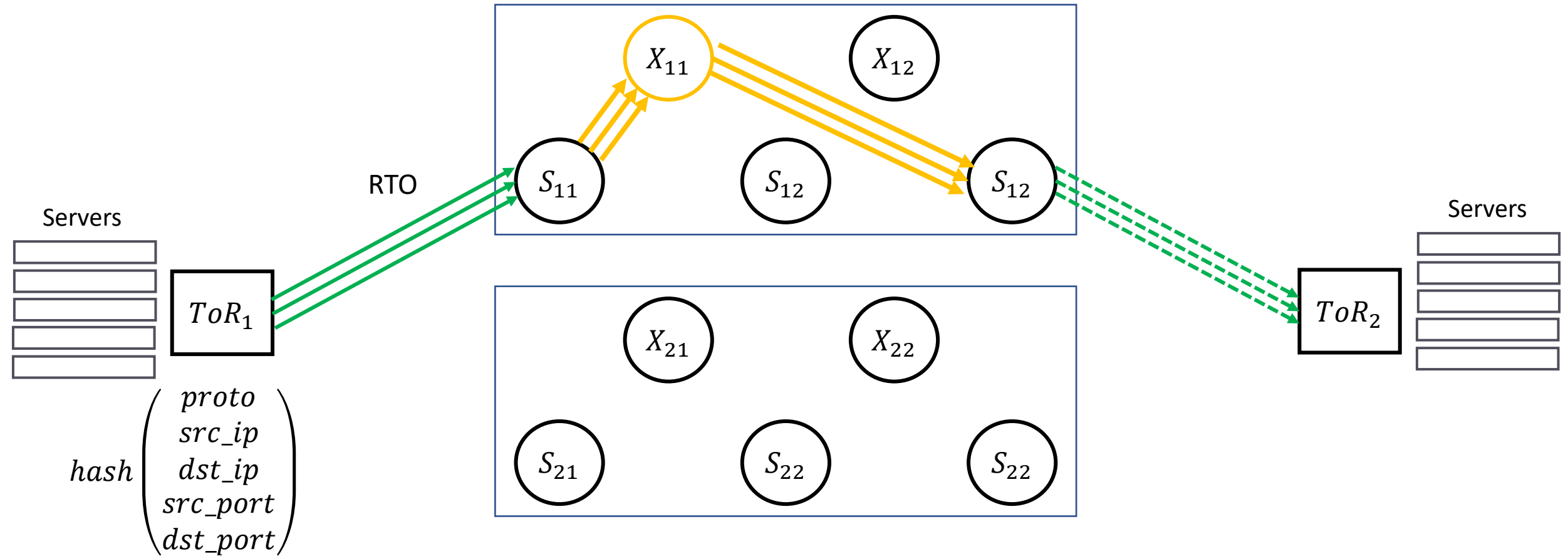
Timeouts	
RTO_MIN	SYN_RTO
200ms	1s

Real RTT
1ms

Unhappy TCP Flow



Unhappy TCP Flow



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;



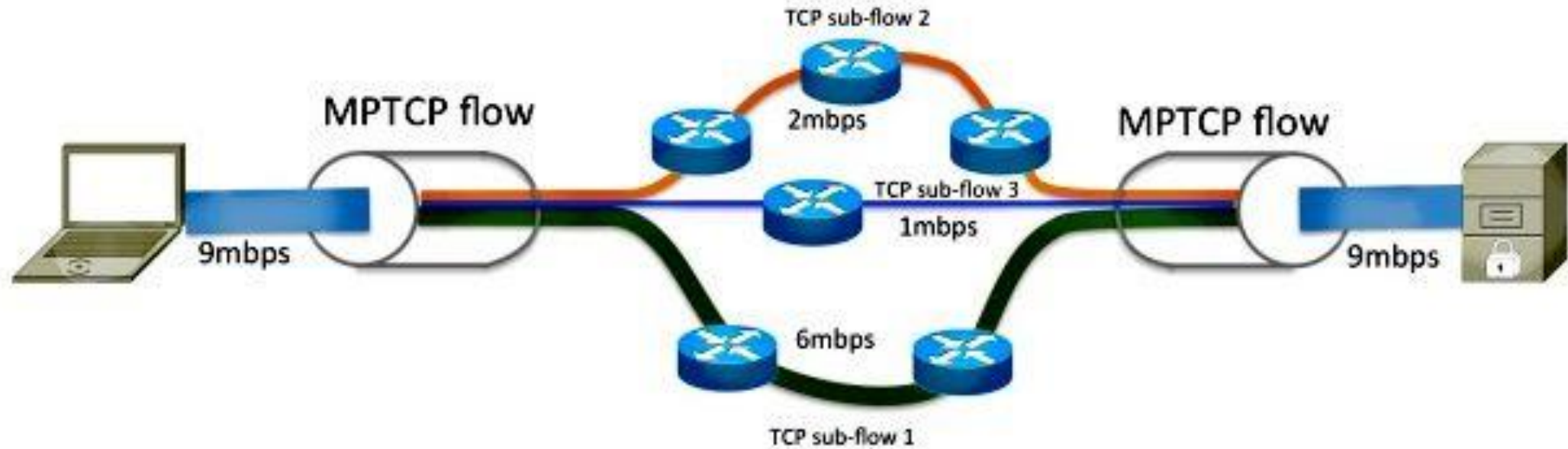
The Old Way: NOC

The Old Way: NOC

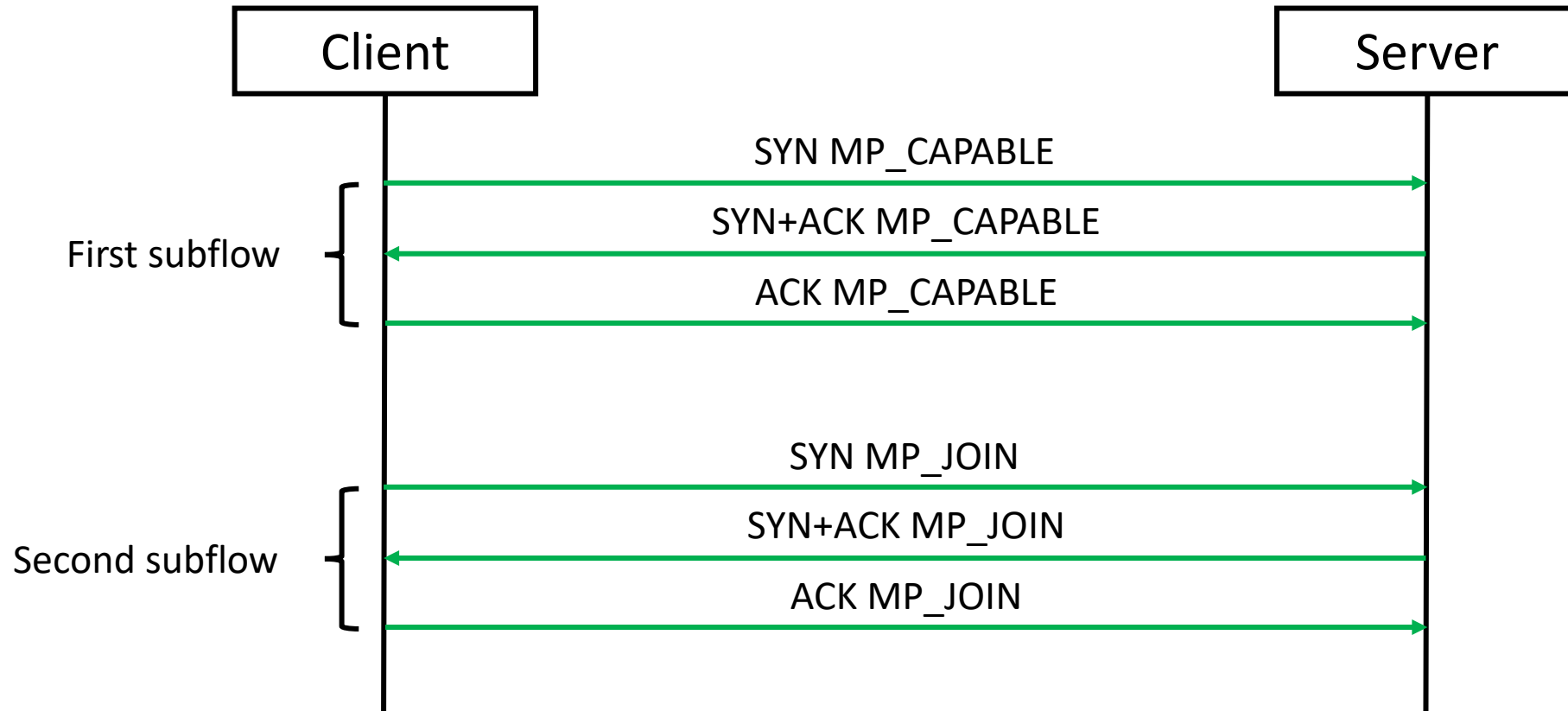
- Outage!
- Detection (1-5 minutes);
- Isolation (5-15 minutes);

Total: 5-20 minutes of service degradation.

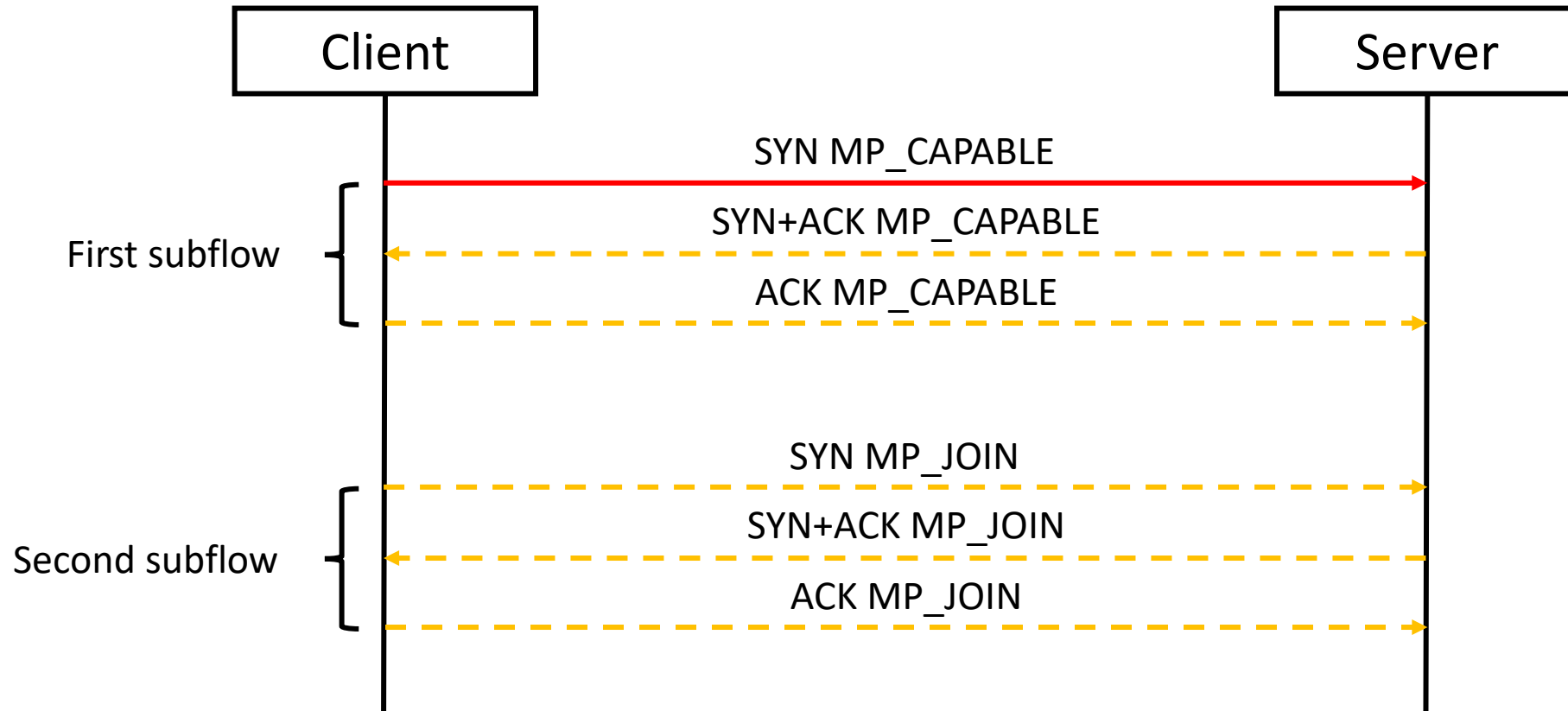
Can Single TCP Flow Use Multiple Paths?



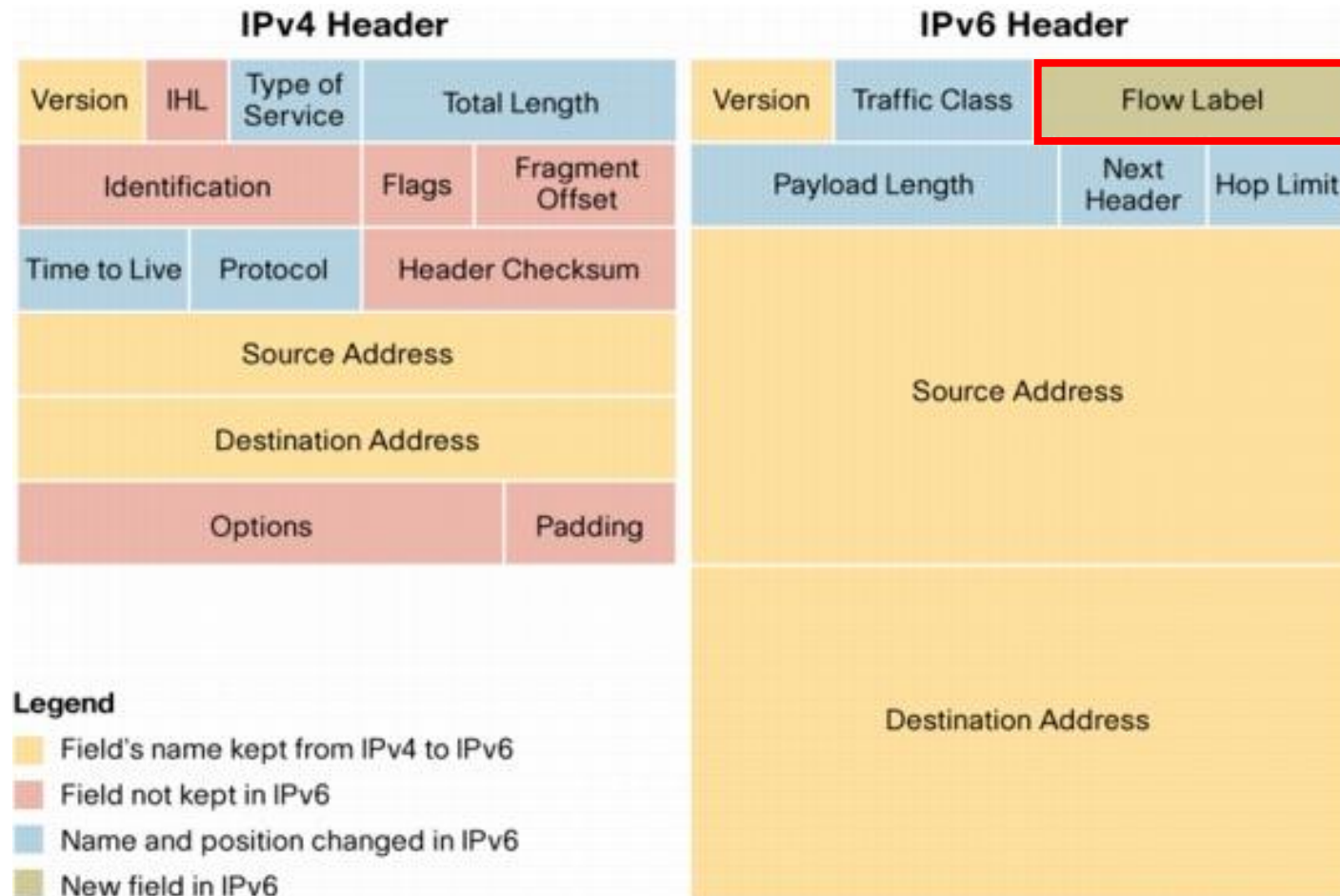
MPTCP – Ordered Subflows



MPTCP – Doesn't Give False Tolerance



IP Headers





CAUTION

**Conspiracy Theory
Ahead**



Flow Label

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 retransmits](#)
Date: Tue, 27 Sep 2016 19:03:37 -0700
Message-ID: <20160928020337.3057238-1-brakmo@fb.com> ([raw](#))

The current code changes txhash (flowlabels) 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)

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

Flow Label: Yet Some Search Engine

The screenshot shows a Google Scholar search interface. The search bar contains the text "flow label tcp". Below the search bar, the results are listed under the heading "Статьи" (Articles). The first result is titled "A Collaborated IPv6-Packets Matching Mechanism Base on Flow Label in OpenFlow" by W Sun, H Wei, Z Ji, Q Zhang, and C Lin, published in the International Conference on ... in 2015 by Springer. The second result is titled "TCP-GEN framework to achieve high performance for HAIPE-encrypted TCP traffic in a satellite communication environment" by Y Kim, JY Jo, and R Harkanson, published in the 2018 IEEE International ... in 2018 by IEEE. The third result is titled "OpenTCP: Combining congestion controls of parallel TCP connections" by S Islam, M Weizl, and S Gjessing, published in the 2016 IEEE Advanced ... in 2016 by IEEE. The fourth result is titled "System and method for conveying the reason for TCP reset in machine-readable form" by JM Smith, CF Lai, AR Carlini, and AS Chittenden, published as US Patent 8,891,532 in 2014 by Google Patents. On the left side of the search results, there are filters for "За все время" (All time), "С 2020" (From 2020), "С 2019" (From 2019), and "С 2016" (From 2016). There is also a "Выбрать даты" (Select dates) button with a date range of "2014" to "...". Below these filters, there are checkboxes for "включая патенты" (including patents) and "показать цитаты" (show citations), and a button "Создать оповещение" (Create notification).

Google Scholar

flow label tcp

Статьи

Результатов: примерно 18 200 (0,08 сек.)

За все время
С 2020
С 2019
С 2016
Выбрать даты
2014 —
Поиск

По релевантности
По дате

☒ включая патенты
☒ показать цитаты
☐ Создать оповещение

A Collaborated IPv6-Packets Matching Mechanism Base on Flow Label in OpenFlow
W Sun, H Wei, Z Ji, Q Zhang, C Lin - International Conference on ..., 2015 - Springer
... and compare the latency, the jitter and the size of **flow** table between **flow** table with **flow label** and **flow** table without **flow label**, and proved that ... The first edition of OpenFlow focused on IPv4 and did not support IPv6 **flow** [7]. ONF started to consider how IPv6 **flows** could be ...
☆ Цитируется: 3 Похожие статьи Все версии статьи (3)

TCP-GEN framework to achieve high performance for HAIPE-encrypted TCP traffic in a satellite communication environment
Y Kim, JY Jo, R Harkanson... - 2018 IEEE International ..., 2018 - ieeexplore.ieee.org
... Even if a collision occurs, no data is lost, but an interference will occur between the merged **TCP flows**. TABLE II ... IP, Source Port, Dest. Port}) TABLE III. IPV6 HEADER FIELDS TO ENCODE **FLOW** ID Old Fields **Flow Label** (20 bits) New Fields **TCP** Marker (1 bit) **TCP** flags (3 ...
☆ Цитируется: 4 Похожие статьи Все версии статьи (2)

OpenTCP: Combining congestion controls of parallel TCP connections
S Islam, M Weizl, S Gjessing... - 2016 IEEE Advanced ..., 2016 - ieeexplore.ieee.org
... life experiments to prove that OpenTCP can efficiently control several concurrent end-to-end **flows** ... 1) S. Amante, B. Carpenter, S. Jiang, and J. Rajahalme, "IPv6 **Flow Label** Specification," RFC ... rfc/rfc6437.txt [2] L. Andrew, S. Floyd, and G. Wang, "Common **TCP** evaluation suite ...
☆ Цитируется: 4 Похожие статьи Все версии статьи (3)

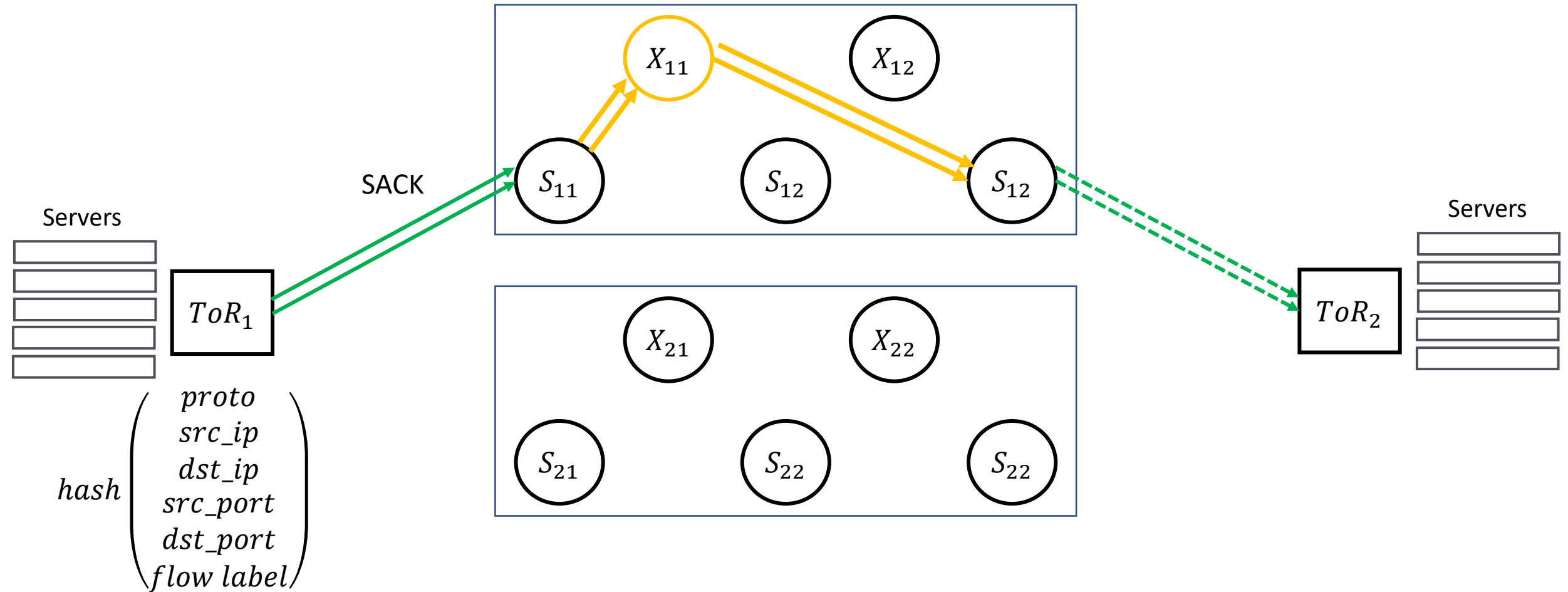
System and method for conveying the reason for TCP reset in machine-readable form
JM Smith, CF Lai, AR Carlini, AS Chittenden - US Patent 8,891,532, 2014 - Google Patents
US8891532B1 - System and method for conveying the reason for **TCP** reset in machine-readable form - Google Patents. System and method for conveying the reason for **TCP** reset in machine-readable form. Download PDF Info ... **TCP** segments have the following general format ...
☆ Цитируется: 35 Похожие статьи Все версии статьи (2)

There is nothing...

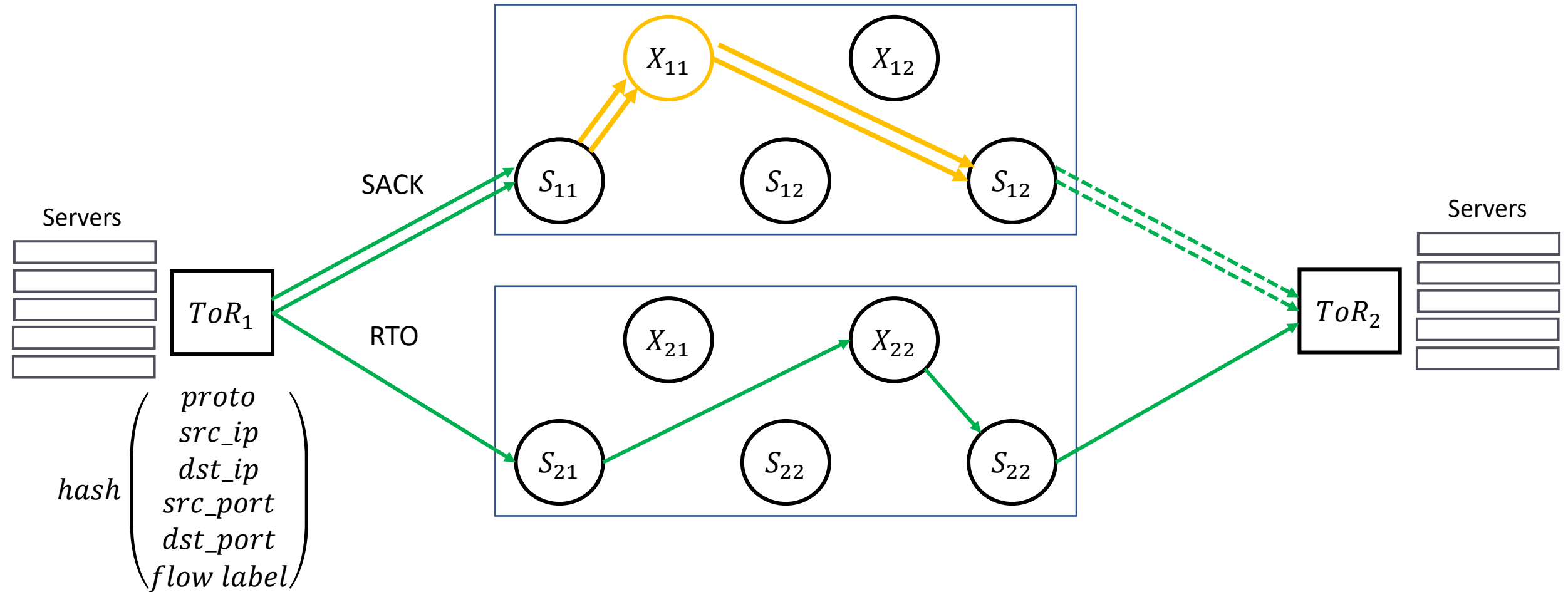


**I KNOW WHAT YOU
ATE LAST SUMMER**

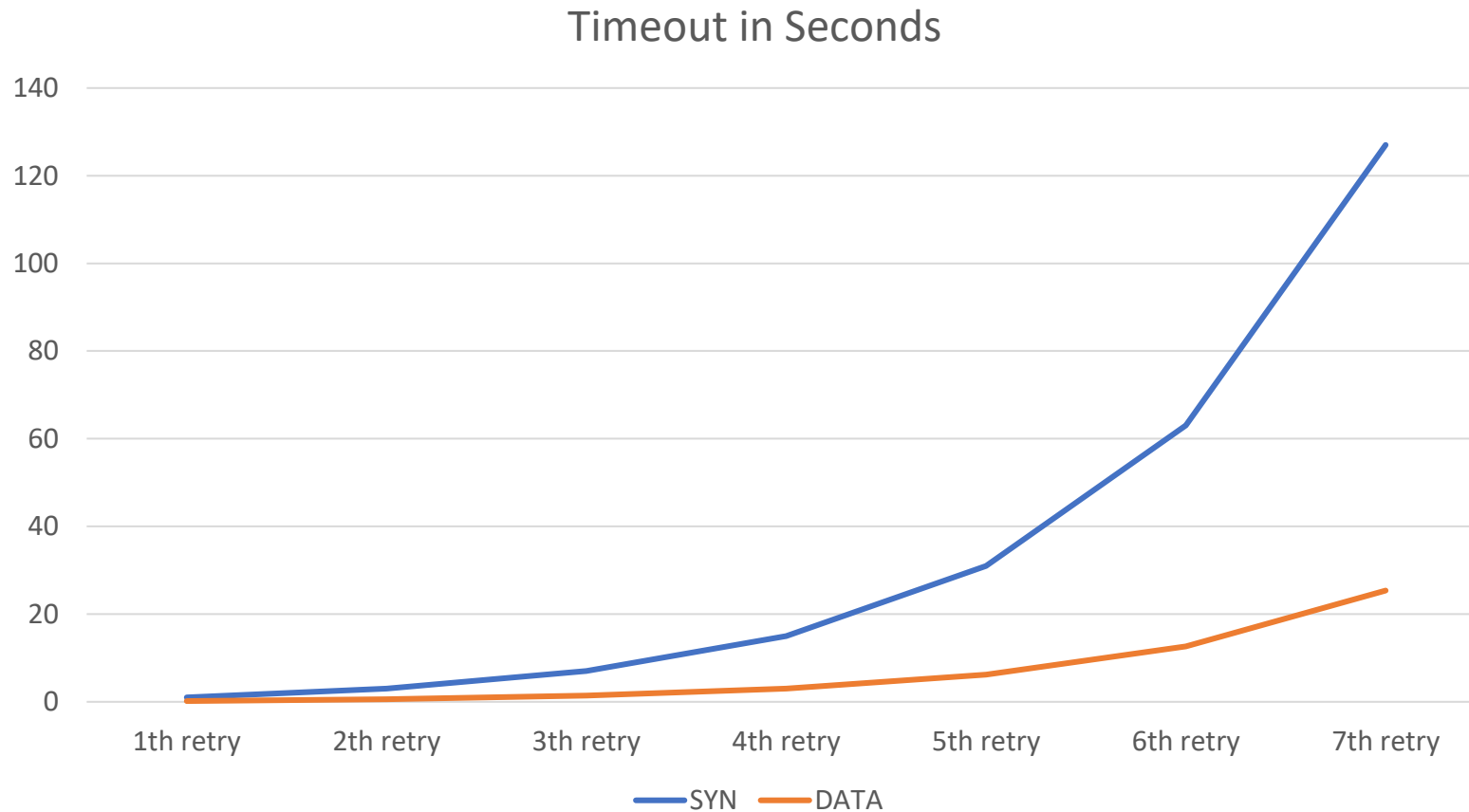
Unhappy TCP Flow



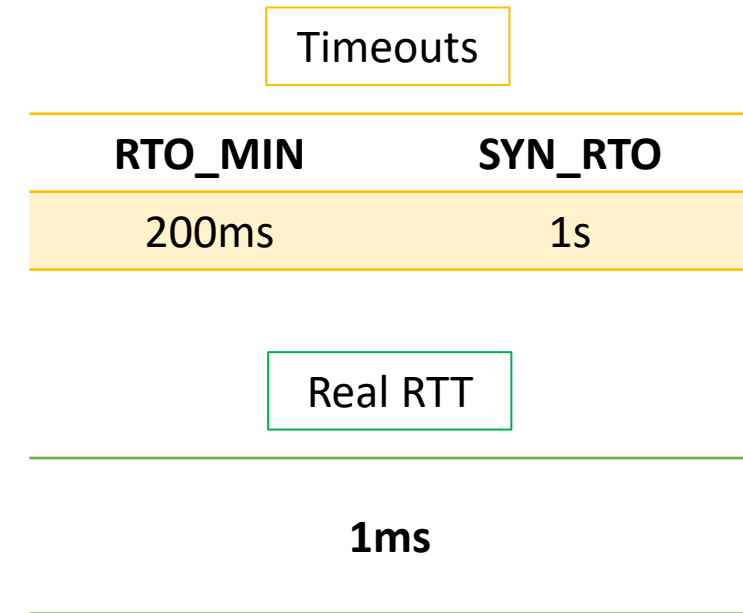
Unhappy TCP Flow Becomes Happier



RTO & SYN_RTO Timeouts



$$\text{RTO} = \text{MAX}(\text{RTO_MIN}, \text{RTT})$$



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
```

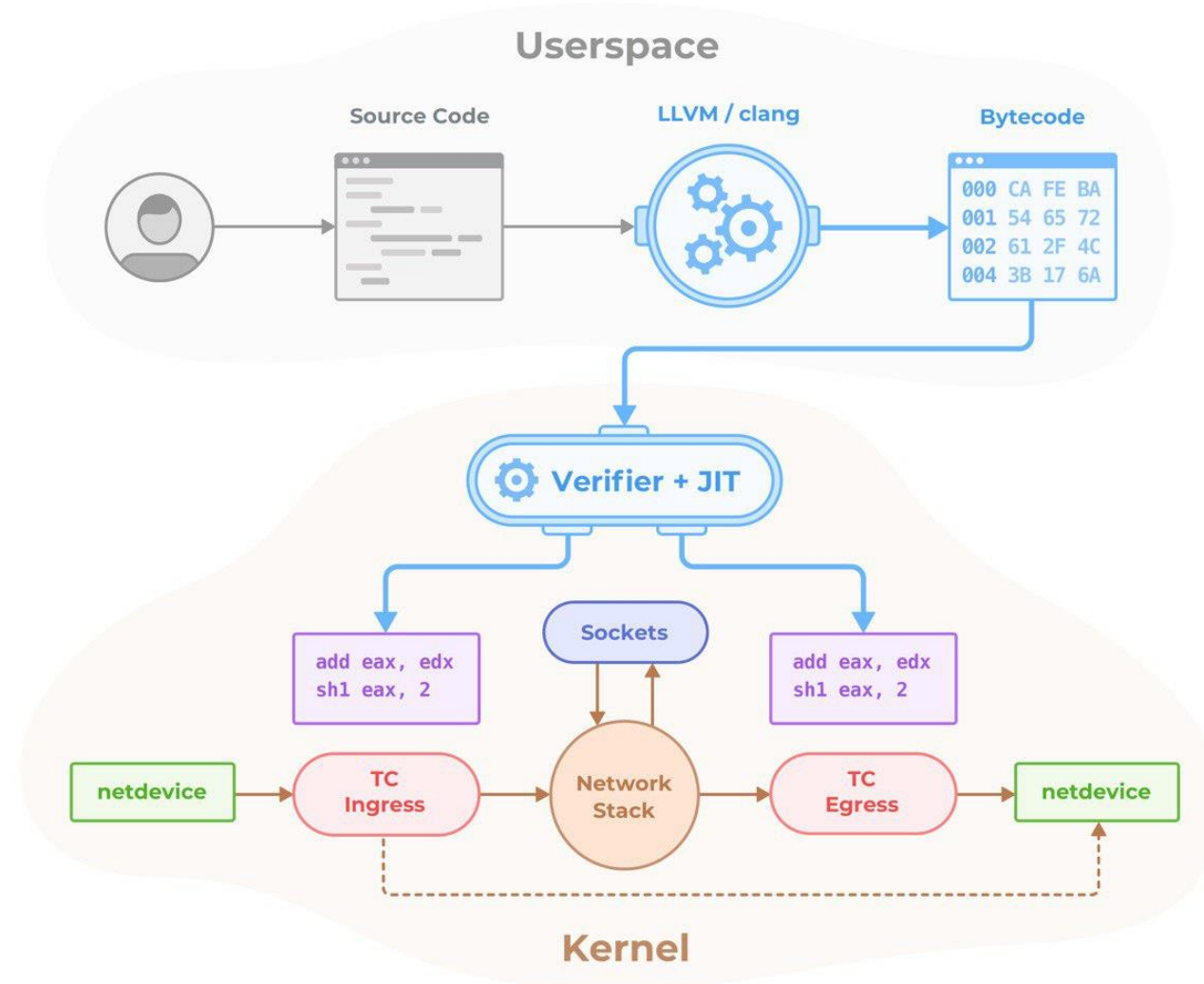
```
OPTIONS := FLAGS [ mtu NUMBER ] [ advms NUMBER ] [ rtt TIME ] [ rttvar TIME ] [ window NUMBER ] [ cwnd NUMBER ] [ initcwnd NUMBER ] [ ssthresh REALM ] [ realms REALM ] [ rto_min TIME ] [ initrwnd NUMBER ]
```




SYN_RTO is
Different



eBPF



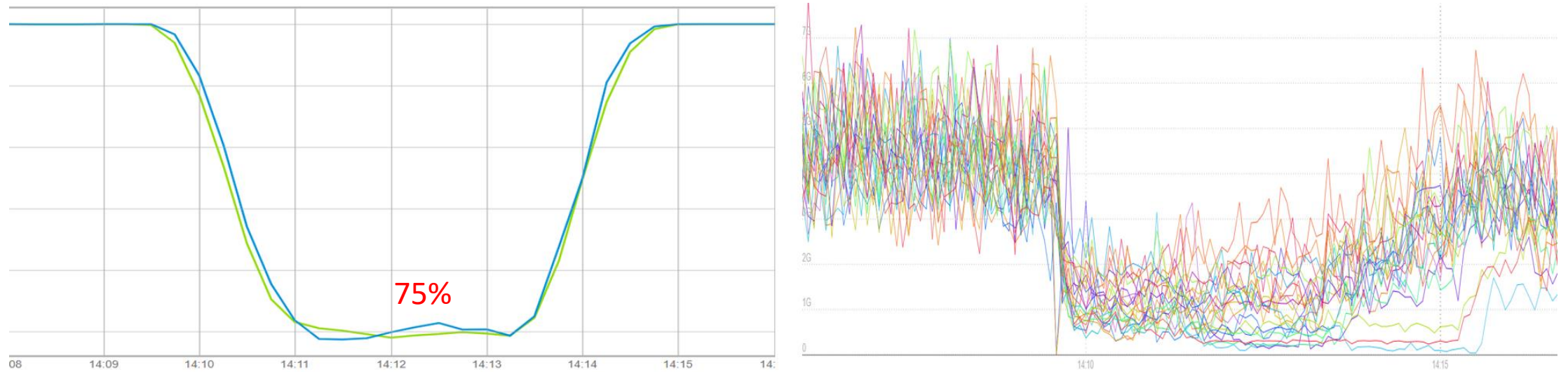
```
/* Check for TIMEOUT INIT operation and IPv6 addresses */
if (op == BPF SOCK OPS TIMEOUT INIT &&
    skops->family == AF_INET6) {

    /* 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]) & 0xffff0000) ==
        (bpf_ntohl(skops->remote_ip6[1]) & 0xffff0000))
        rv = 10;
}
```

Changing SYN RTO

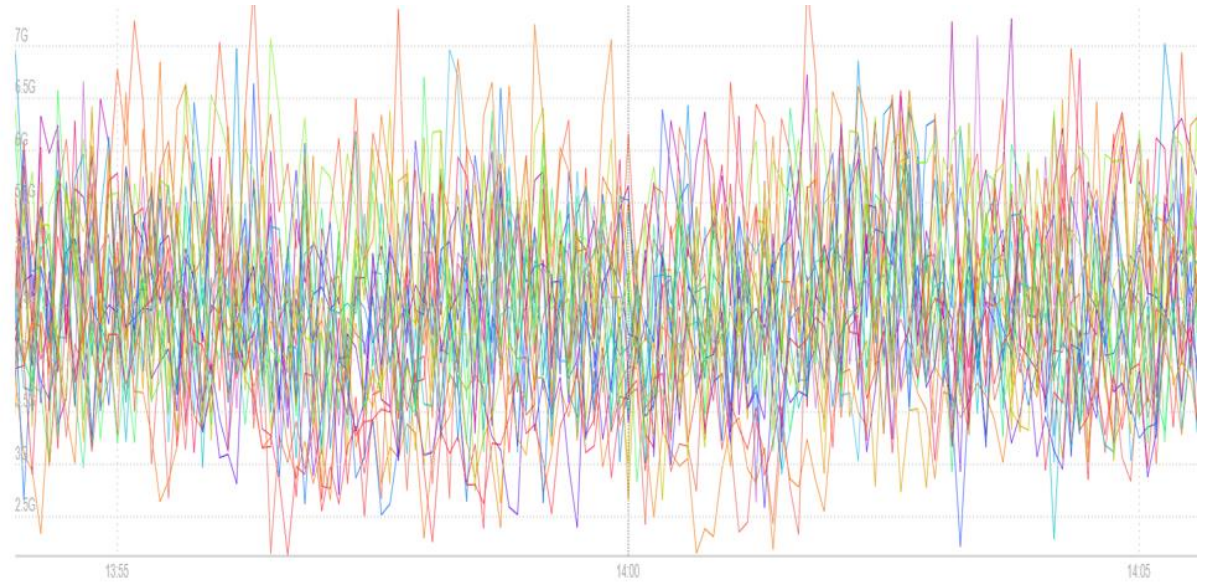
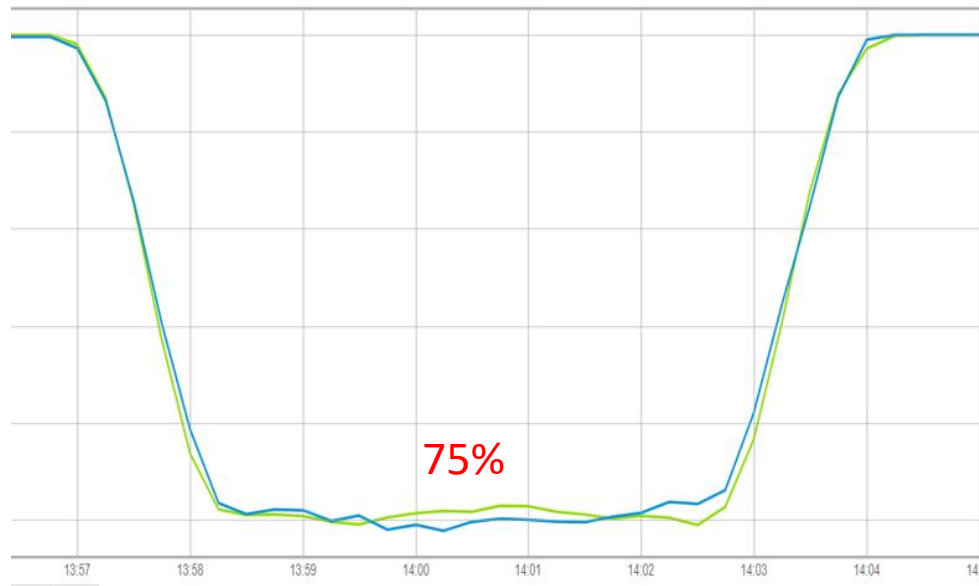
https://elixir.bootlin.com/linux/latest/source/samples/bpf/tcp_synrto_kern.c

Evaluation: Without Flow Label



One of four ToR uplinks drops packets, significant service degradation

Evaluation: Flow Label + eBPF

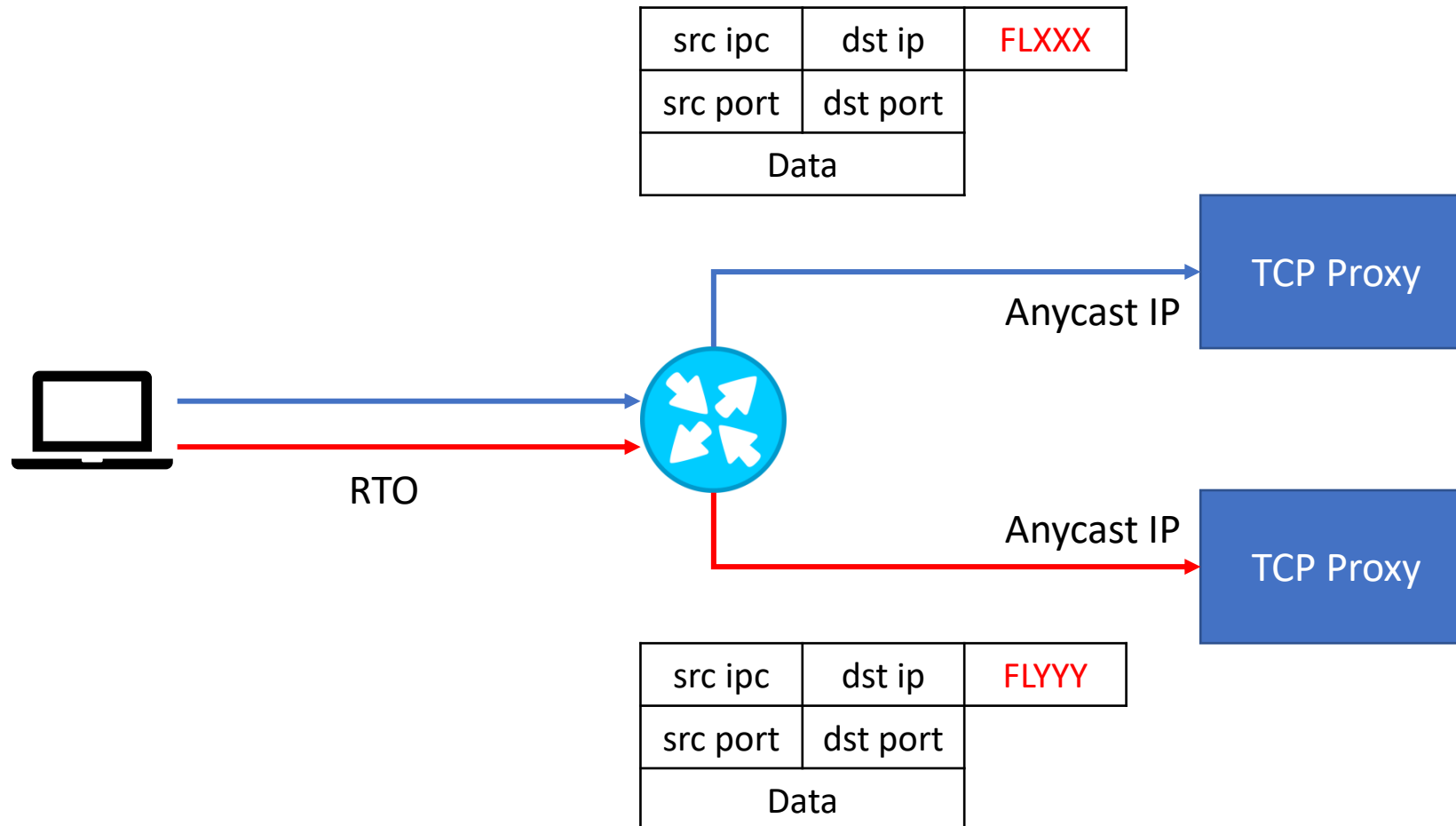


One of four ToR uplink drops packets, no effect on the service!

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!**

Side Effect



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;

TCP

Self-healing ~~Datacenter~~: Cookbook

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