ECMP-ER

Extending ECMP toward A Practical Hardware Load Balancer

Ryo Nakamura[1], **Kentaro Ebisawa**[2], Tomoko Okuzawa[2], Chunghan Lee[2], Yuji Sekiya[1]

1 Information Technology Center, The University of Tokyo

2 Toyota Motor Corporation

About

- This slide describes ECMP-ER, which was published at IOTS 2022 on 9th Dec, 2022
  - IOTS 2022: IPSJ SIG IOT (Internet and Operation Technology) Symposium 2022
  - Program (Japanese): https://www.iot.ipsj.or.jp/symposium/iots2022-program/

- Objective of this slide is to provide information about ECMP-ER in English
  - IOTS 2022 paper is written in Japanese whose English abstract is referenced in the next slide

Abstract

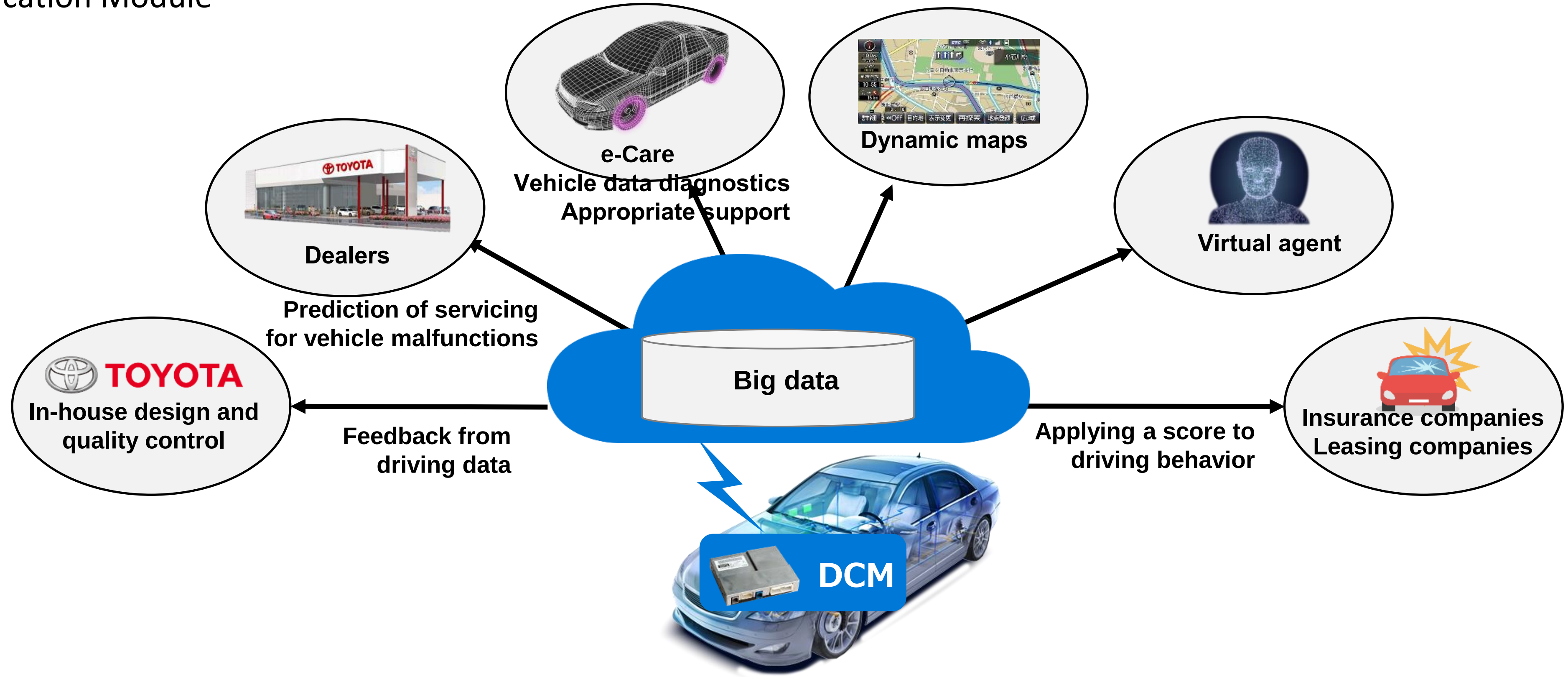
This paper proposes an enhanced Equal Cost Multi-path (ECMP) to address a drawback of ECMP as a load balancer. ECMP functionality of commodity hardware routers can distribute traffic to multiple equal-cost next-hops on a per-flow basis. Therefore, if we could use ECMP of hardware routers as load balancers, it is better than introducing dedicated load balancers from investment and operational costs. However, ECMP as a load balancer has an issue; when next-hops for an ECMP entry increase or decrease, existing connections would be transferred to another server and be disrupted. In this paper, we propose ECMP with Explicit Retransmission (ECMP-ER) to tackle this issue. ECMP-ER is based on traditional layer-3 ECMP; thus it runs with only traditional layer-3 routing mechanisms. Moreover, ECMP-ER maintains next-hops of the previous state for each ECMP entry. When a packet is transferred to a different server due to next-hops change, the server re-transmits the packet, and the ECMP-ER router forwards it with the previous next-hops. As a result, the packet arrives at the correct server. We prototyped ECMP-ER on a P4 switch, and the evaluation shows that ECMP-ER transfers all traffic without disruption in a situation where ECMP lost 20% of connections.

Reference: https://ipsj.ixsq.nii.ac.jp/ej/?action=pages_view_main&active_action=repository_view_main_item_detail&item_id=222743&item_no=1&page_id=13&block_id=8

Motivation: Growth of upstream traffic from connected cars

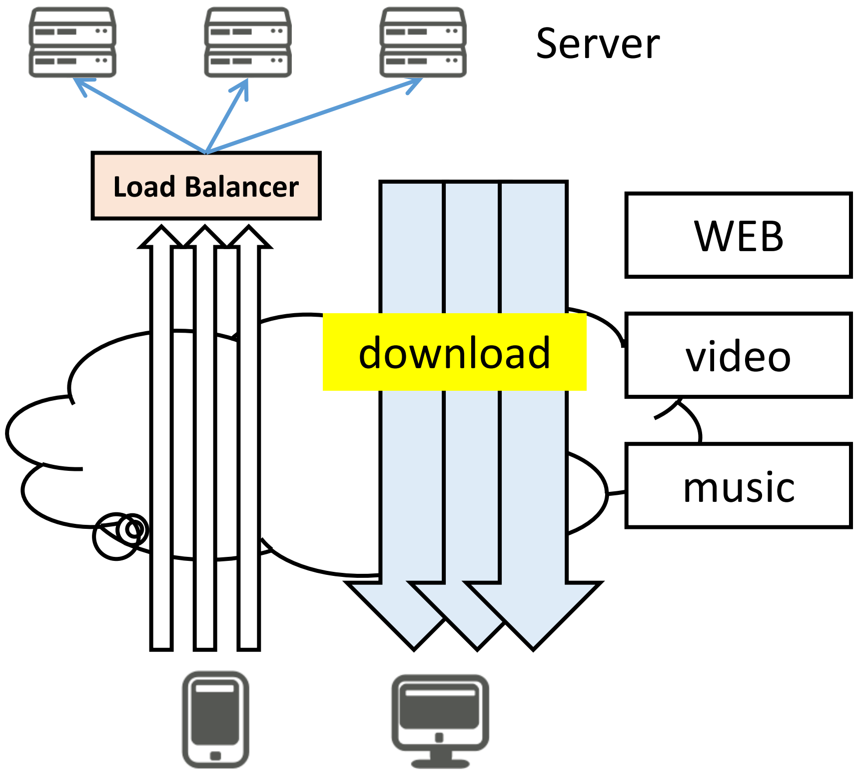
- Text data (CAN) to photo/video data to enable rich services (e.g. dynamic maps)
- More sensors, more data (e.g. autonomous car)
- Increase of number of cars connected to network (DCM[*] as default option)

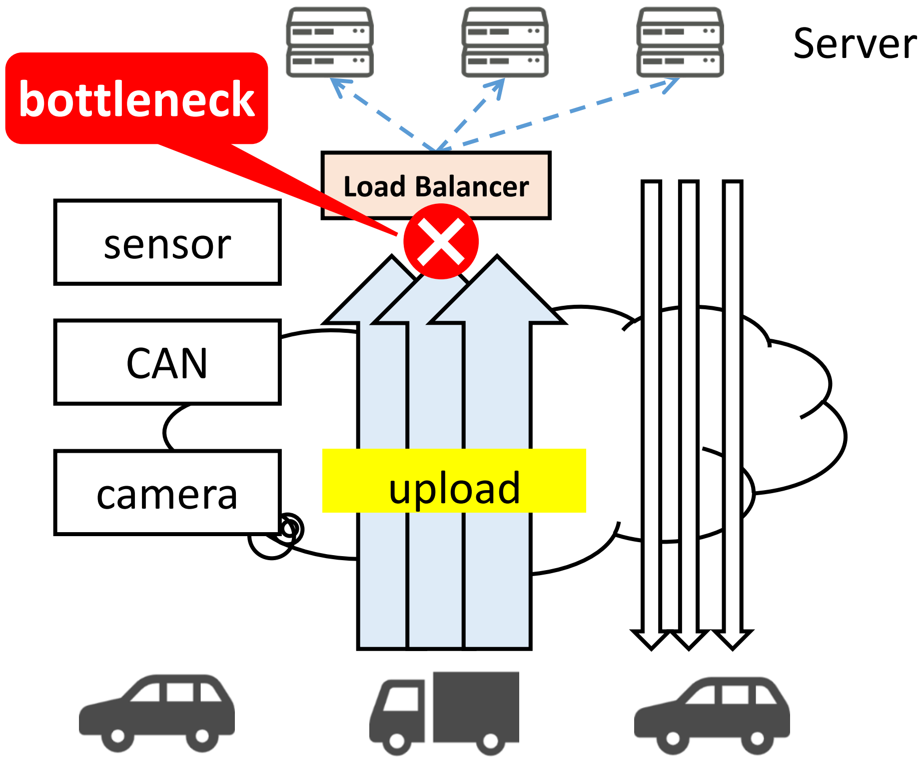
[*] DCM ... Data Communication Module



Lack of simple and scalable way to balance upstream traffic

- Load Balancers (LBs) focused for web and video streaming traffic (data center to consumer)
- Direct Server Return (DSR) only scales downstream traffic
- Direction agnostic LBs using NPU/FPGA exist, but comes with a big price tag





State Less vs. State Full

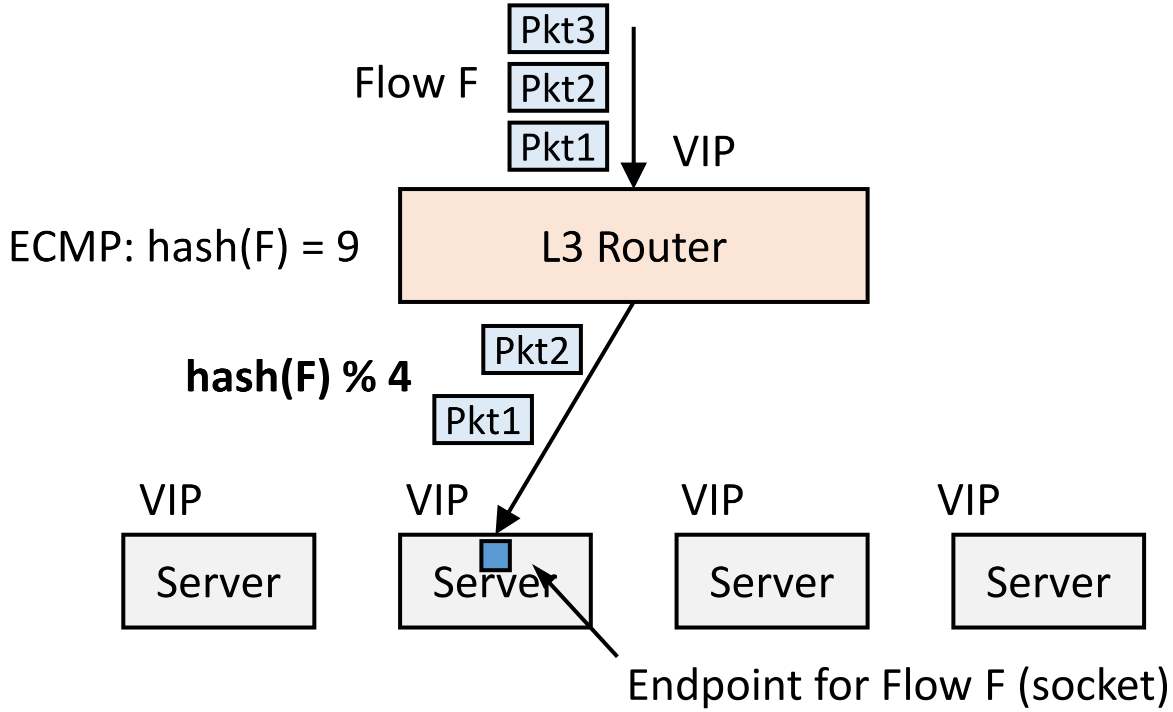
- State Less Load Balancer (e.g. ECMP)
  - ECMP ... Equal Cost Multi Path
  - identify destination server by calculating Hash of flow info (5 tuple)
  - easy to scale by using hardware with less resource
  - vulnerable to adding / removing servers (Hash result change impacting existing connections)
  - => need additional method to ensure **Per-connection Consistency (PCC)**

- State Full Load Balancer
  - Easy to ensure PCC by maintaining flow and server mapping
  - Lack of Scalability
    - requires memory to store mapping table
      - hardware based load balancers stores this table in expensive TCAM/SRAM
      - software based load balancers could hold huge table, but difficult to achive high throughput
    - need to sync table among LBs for active/active clustering
    - vulnerable to SYN-flood attach

Problem of State Less Load Balancer using ECMP

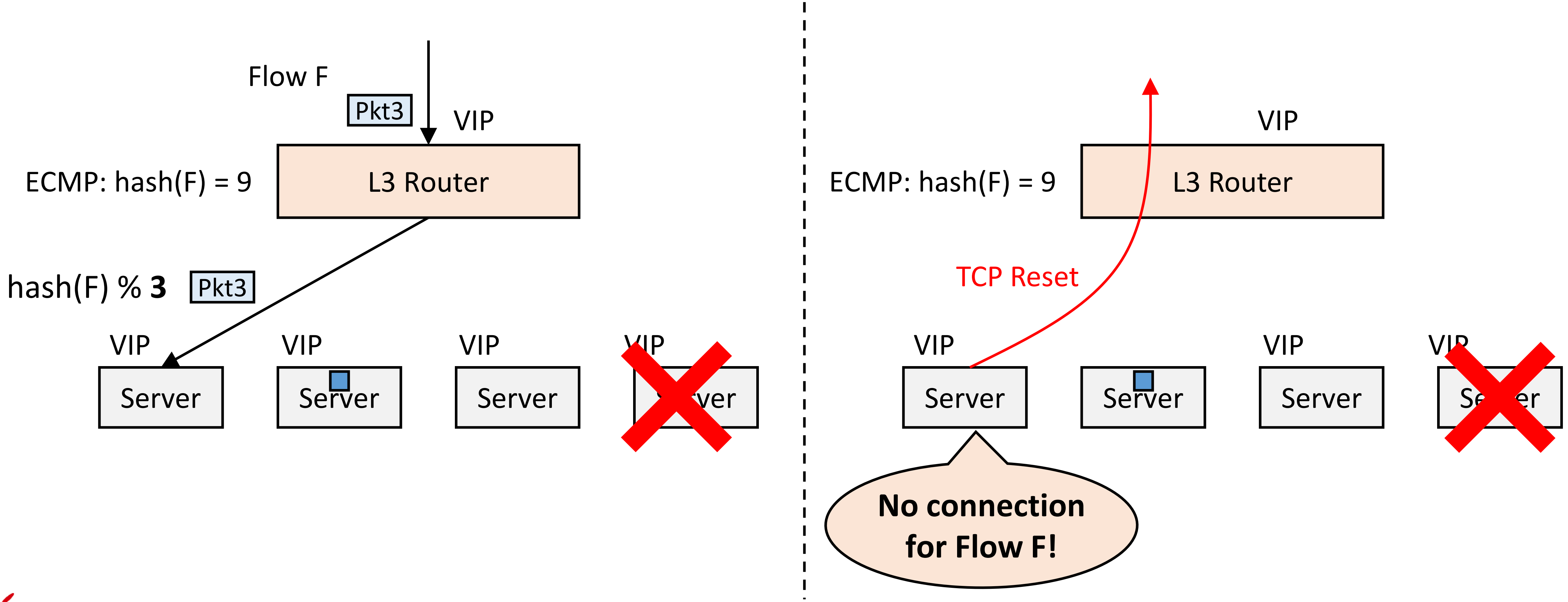
- Decide destination Server based on hash calculation result

e.g. Destination Server ID = hash (Flow_info) % <number_of_servers>



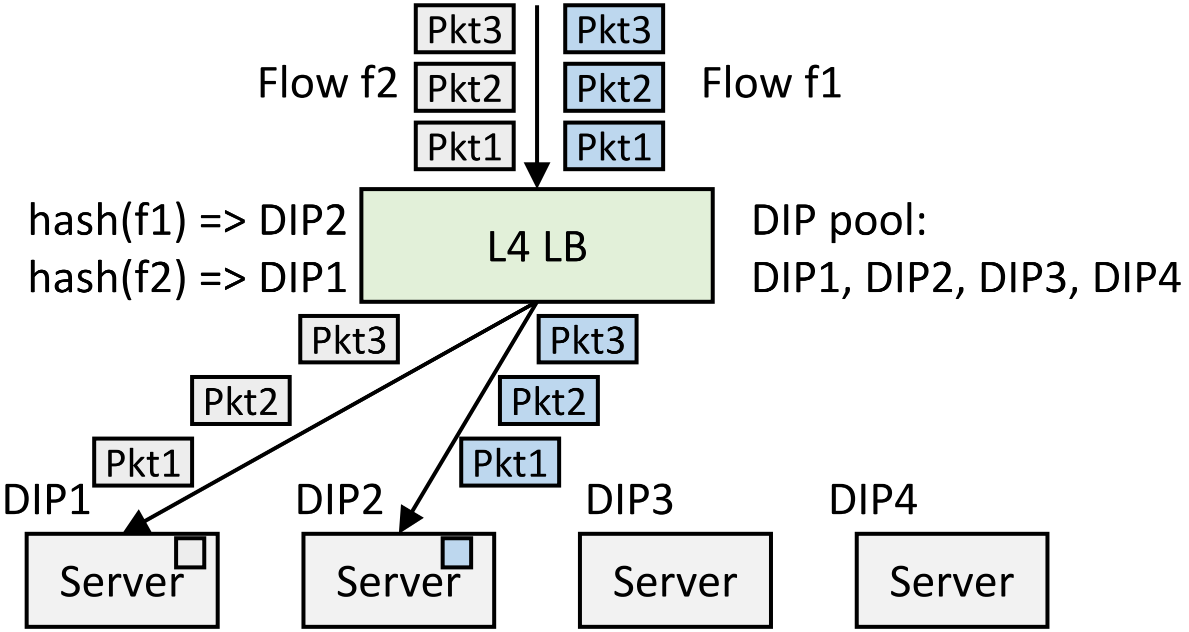
- When adding/removing server and <num_of_servers> changes, destination address for the same flow (TCP connection) will change resulting to TCP reset even for connection running on unchanged servers

=> need additional method to ensure **Per-connection Consistency (PCC)**



State Full Load Balancer

- Easy to ensure PCC by maintaining flow and server mapping
- Lack of Scalability
  - requires memory to store mapping table
    - hardware based load balancers stores this table in expensive TCAM/SRAM
    - software based load balancers could hold huge table, but difficult to achive high throughput
  - need to sync table among LBs for active/active clustering
  - vulnerable to SYN-flood attach



State Less vs. State Full (summary)

State Less Load Balancer

- CAN NOT ensure PCC
- implementation based on hardware
- High Throughput

State Full Load Balancer

- CAN ensure PCC
- implementation based on software
- Low Throughput

	PCC	Data Plane	Throughput
Stateless		Hardware	**High (Tbps)**
Stateful	✔	Software	**Low (Gbps)**

Approches to research Load Balancer

- Seek for better method for State Full LB on software
- Seek for a method to implement State Full LB on hardware
- Seek for a method to ensure PCC on hardware based State Less LB

Previous Works

	Stateless	Stateful
Software		MS Ananta (SIGCOMM'13), Google Maglev (NSDI'16), Facebook katran
Hardware	Beamer (NSDI'18), Fastly Failed (NSDI'18)	Facebook SilkRoad (SIGCOMM'17), Tiara (NSDI'22)

Issues common to previous works

Maglev is a highly complex distributed system that has been serving Google for over six years. We have learned a lot while operating it at a global scale.

From Section 4 Operational Experience in the Maglev paper (NSDI'16)

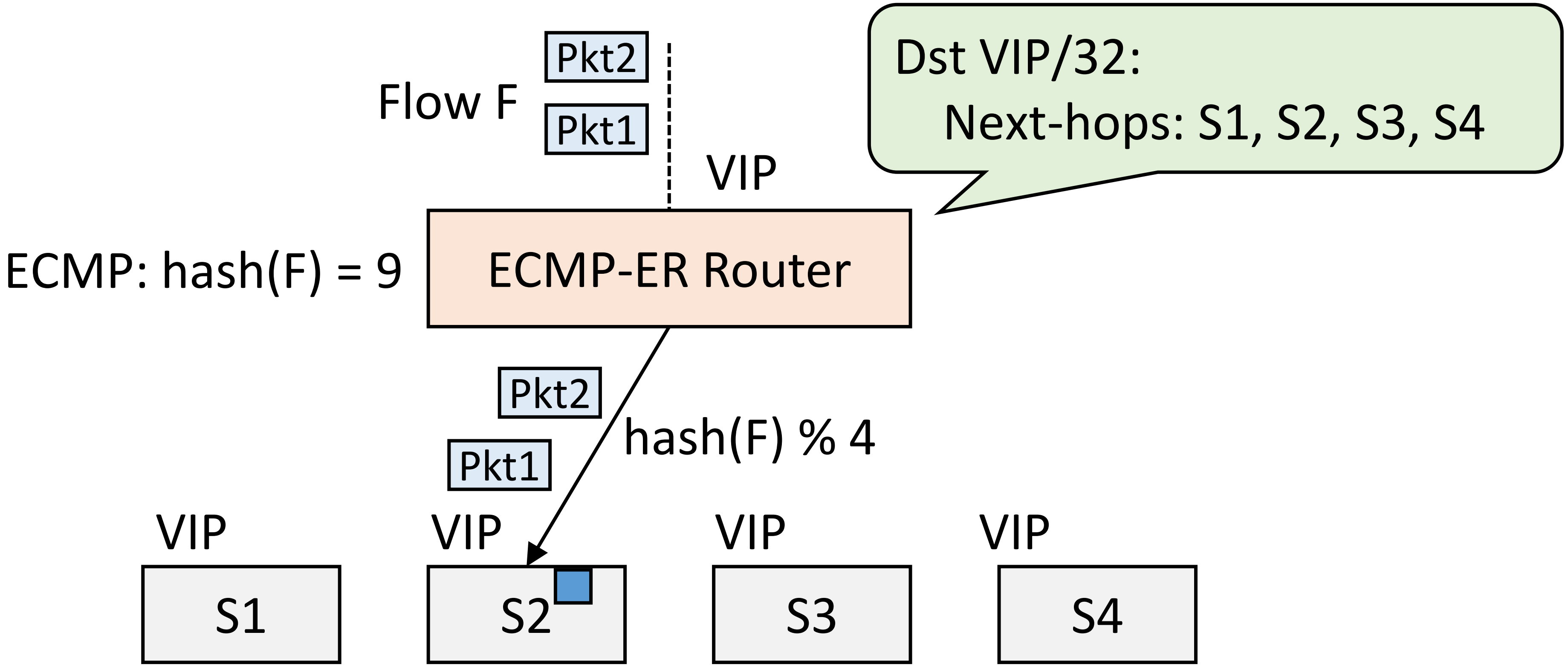
- How to control Load Balancer and operation?
  - updating VIP and DIP, sync state among LBs, how to connect with DC networks, etc.
- These are essential for real operations, but method is not self-evident
- Below methods are mentioned in paper, but not detail enough to actually operate
  - use cloud controller (Ananta)
  - read external file (Maglev)
  - use Zookeeper (Beamer)
  - run agent on switch (Faild)

Not many can operate like how hyper giants do.

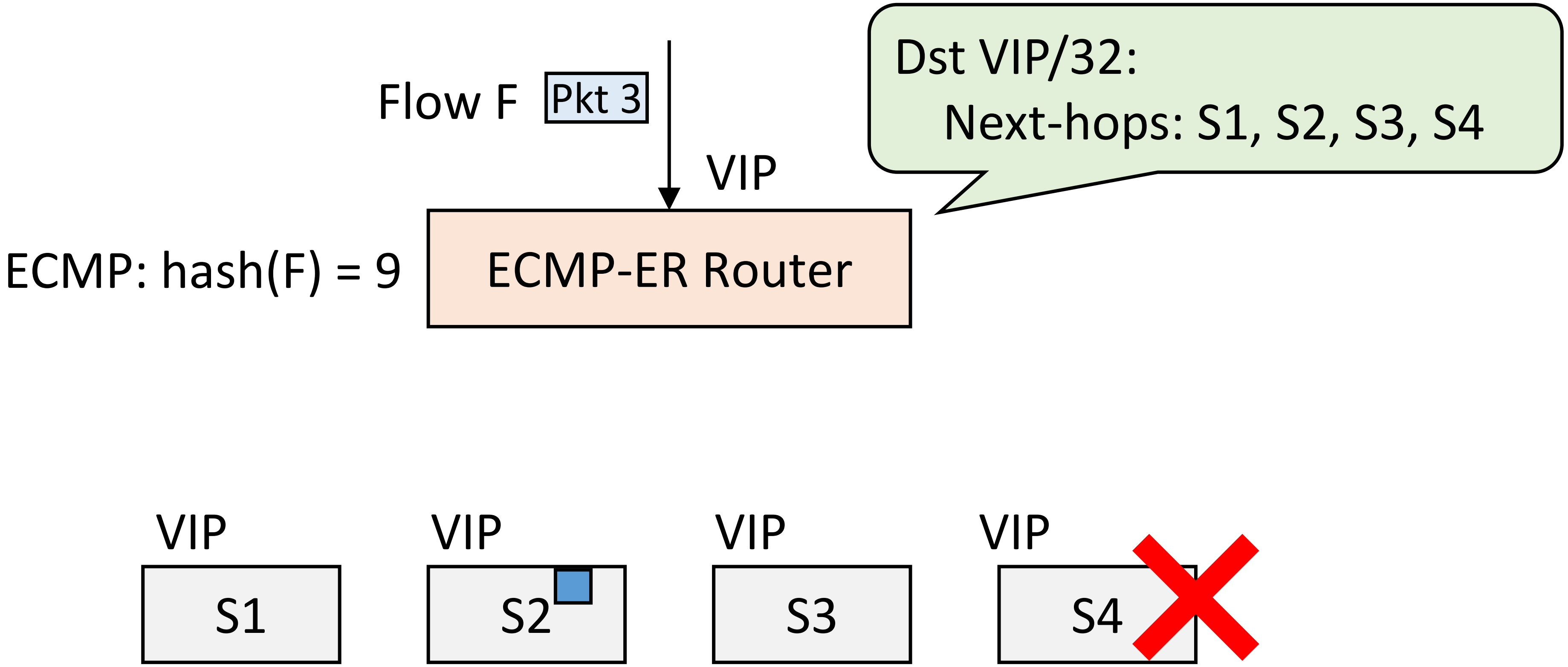
ECMP-ER: ECMP with Explicit Retransmission

- Approach
  - Ensure PCC with State Less and Hardware based Load Balancer
  - Server forwards' packet to another server, if the packet does not belong to the server
- No extra control mechanism required for Load Balancer
  - Works with existing routing protocols
  - (because we are using ECMP)
- Ensure PCC with minial change to ECMP and Server functionality
  1. **Router** will maintain ECMP route information (nexthops) up to one generation ago
  2. **Server** will forward packet to router if the connection does not below to the server
  3. **Router** will forward packet to the nexthop one generation ago, if it was sent back from server

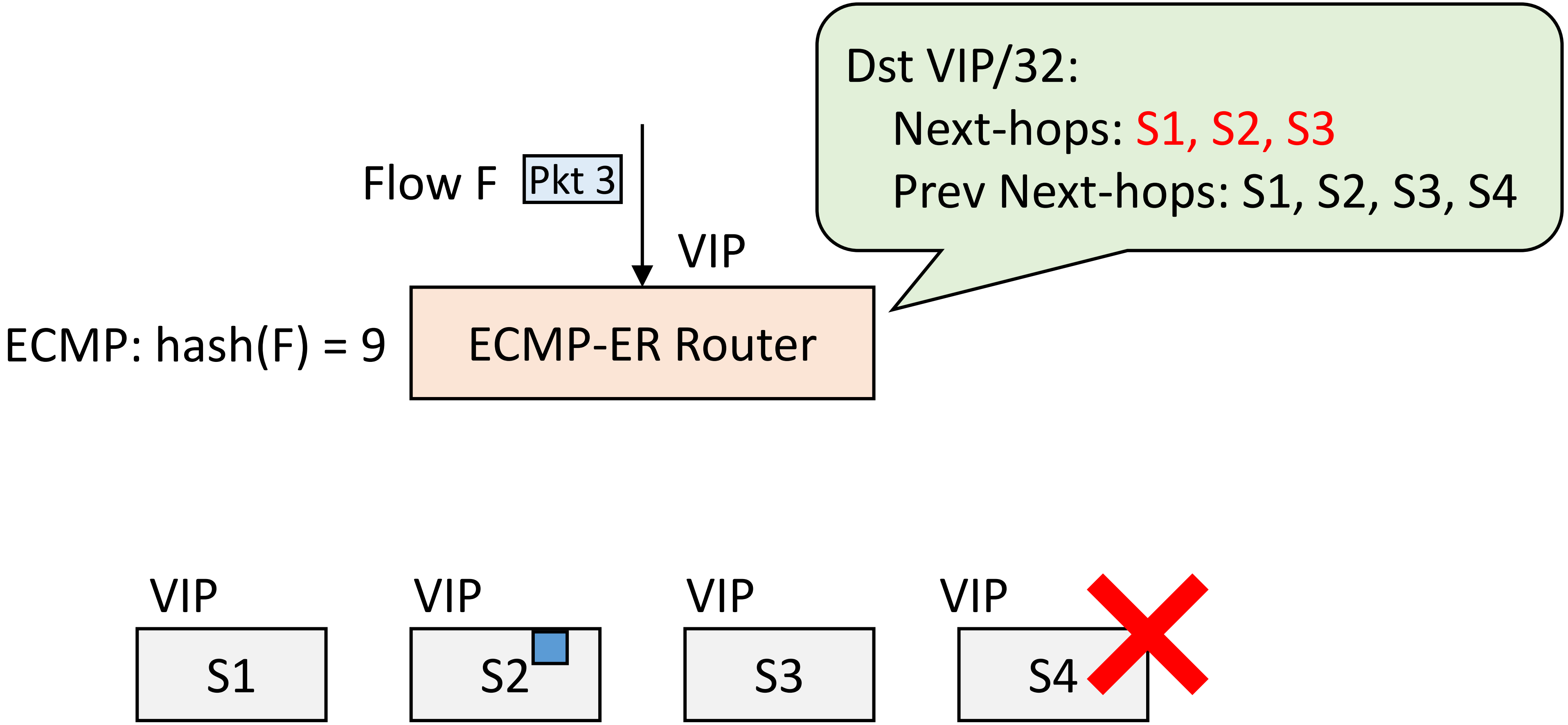
Example of How ECMP-ER Works (1 / 6)



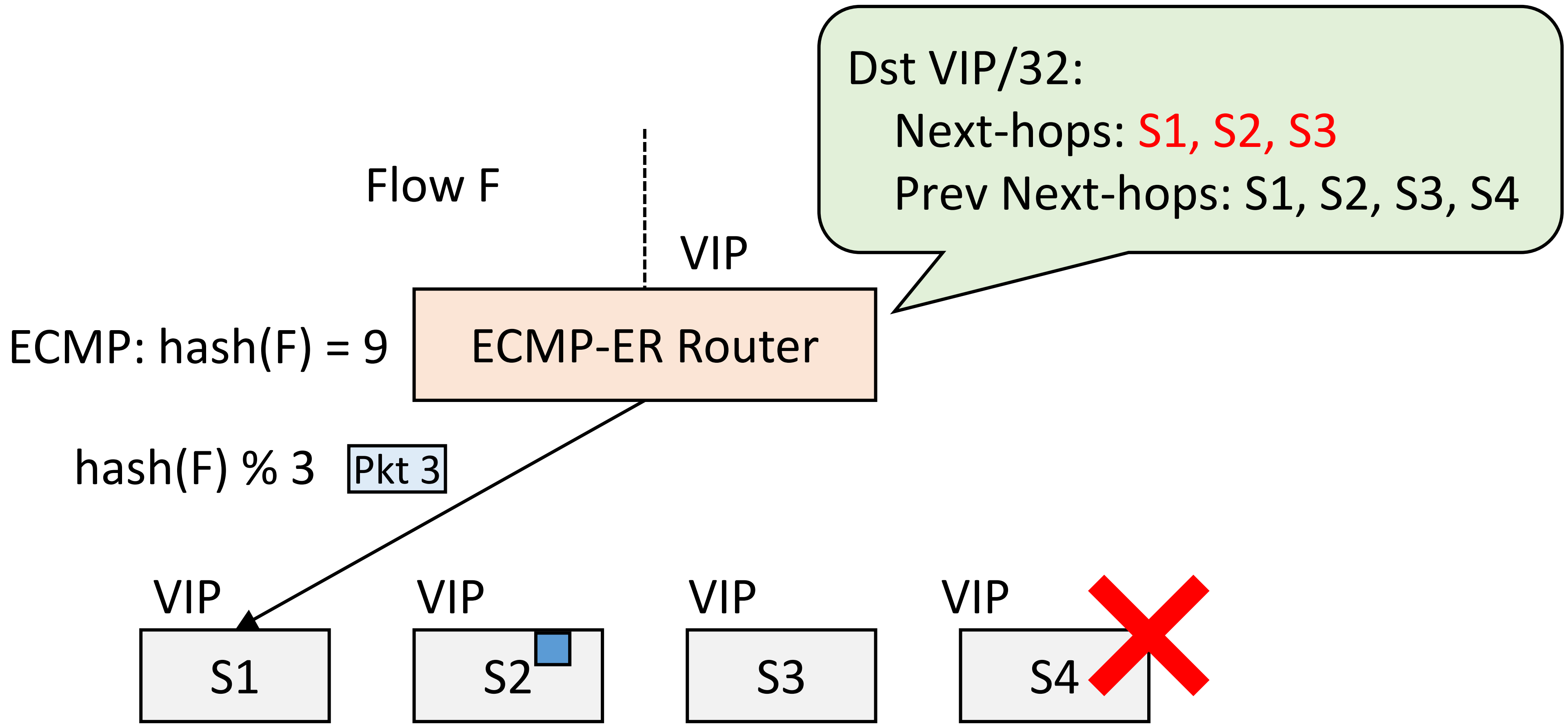
Example of How ECMP-ER Works (2 / 6)



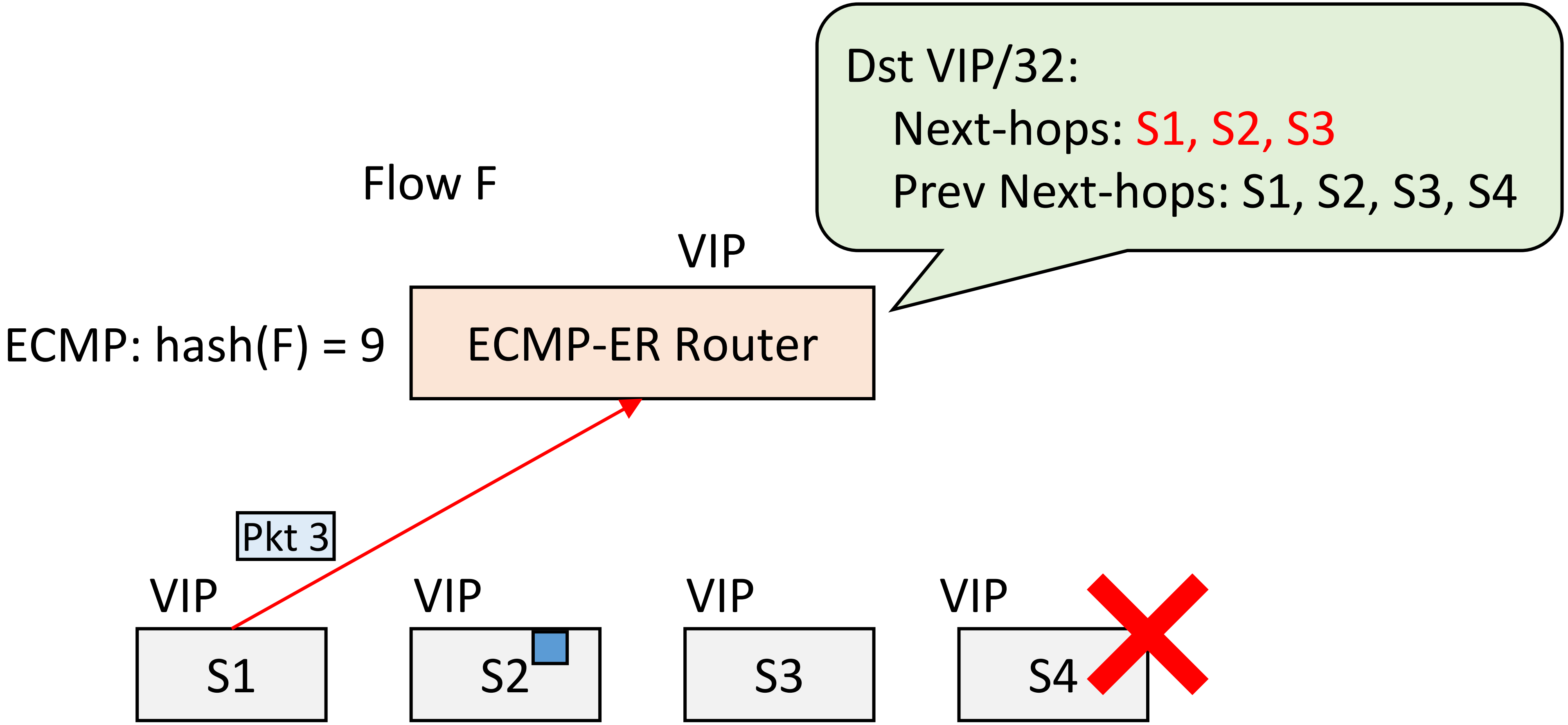
Example of How ECMP-ER Works (3 / 6)



Example of How ECMP-ER Works (4 / 6)



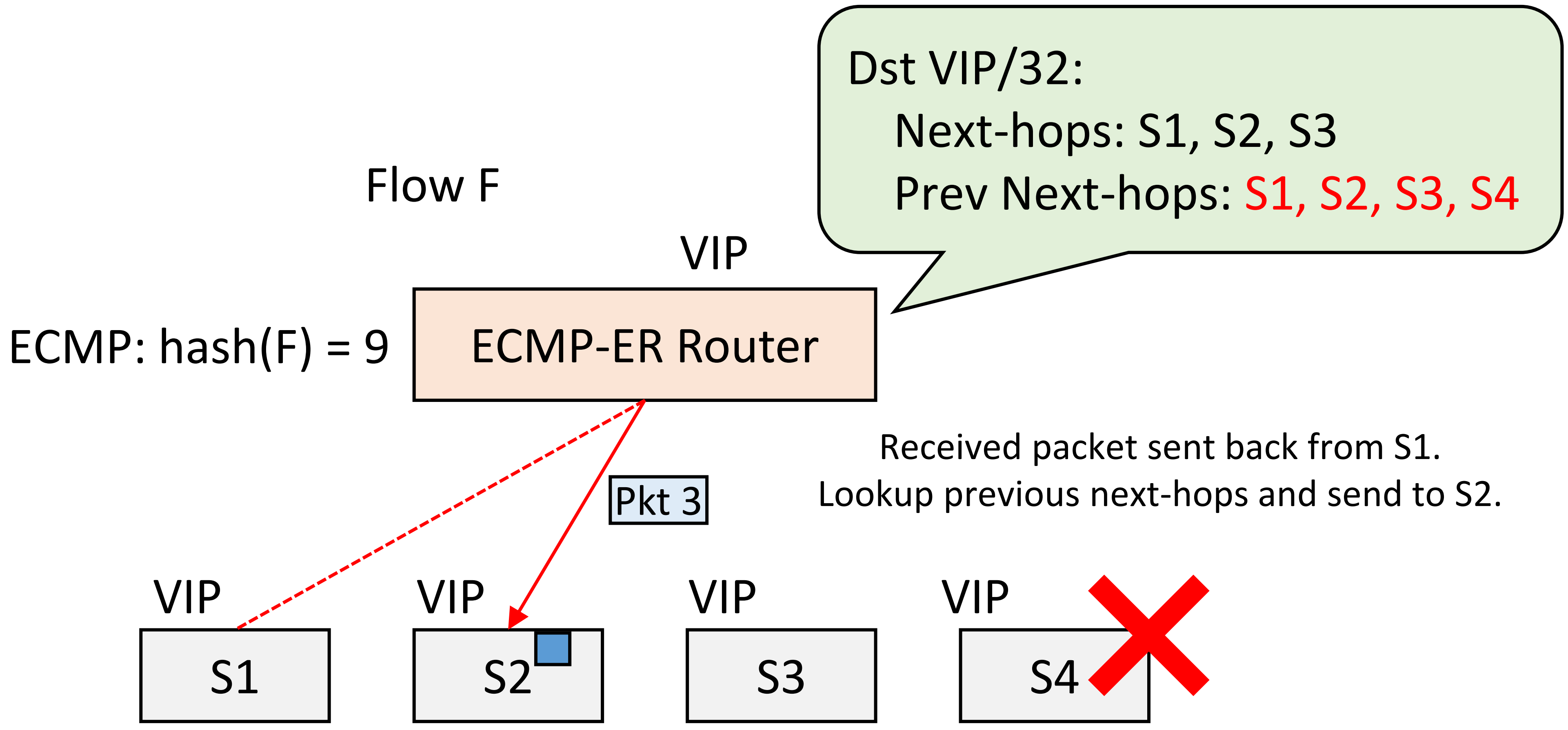
Example of How ECMP-ER Works (5 / 6)



S1 does not own Flow F

Send the packet back to the ECMP-ER Router

Example of How ECMP-ER Works (6 / 6)



Server retransmission logic operates autonomously

Control Plane is NOT required

1. If received packet is TCP and not a SYN packet (== part of existing Flow), and if the server does not own established socket for the packet, server sends the packet to ECMP-ER router
2. Else, it will receive packet

```
if packet is TCP and packet is not SYN then
    if find_socket(packet) is None then
        return retransmit(packet)
return receive(packet)
```

Pros and Cons of ECMP-ER

Pros

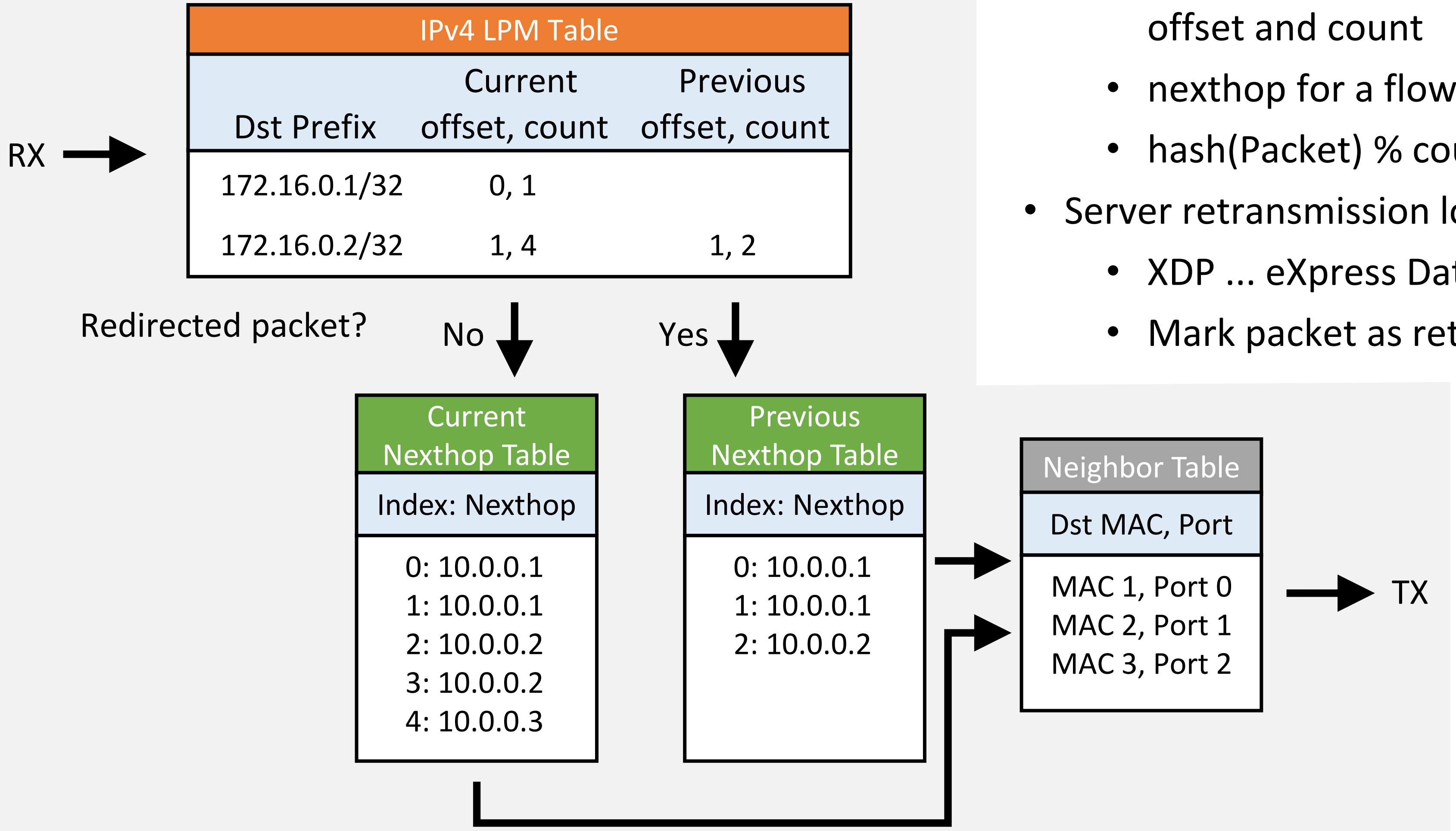
- Ensure PCC on Hardware without State (State Less)
  - High throughput utilizing Ethernet ASIC / NPU
- Special control plane is NOT required
  - Only route exchange (or configuration) between routers and servers is required
  - Easy to operate

Cons

- Cannot distribute traffic based on server load etc.
  - Could be done to some extent using Weighted ECMP

Prototype Implementation of ECMP-ER

ECMP-ER Router implementation using P4



- ECMP-ER Router was prototyped using P4 on Tofino ASIC
  - nexthops to VIP (Dst Prefix) are stored in a table using offset and count
  - nexthop for a flow is decided with below formula
  - hash(Packet) % count + offset
- Server retransmission logic was prototyped using XDP
  - XDP ... eXpress Data Path
  - Mark packet as retransmitted using ToS field

Evaluation of ECMP-ER

Eval#1 Operational Verification

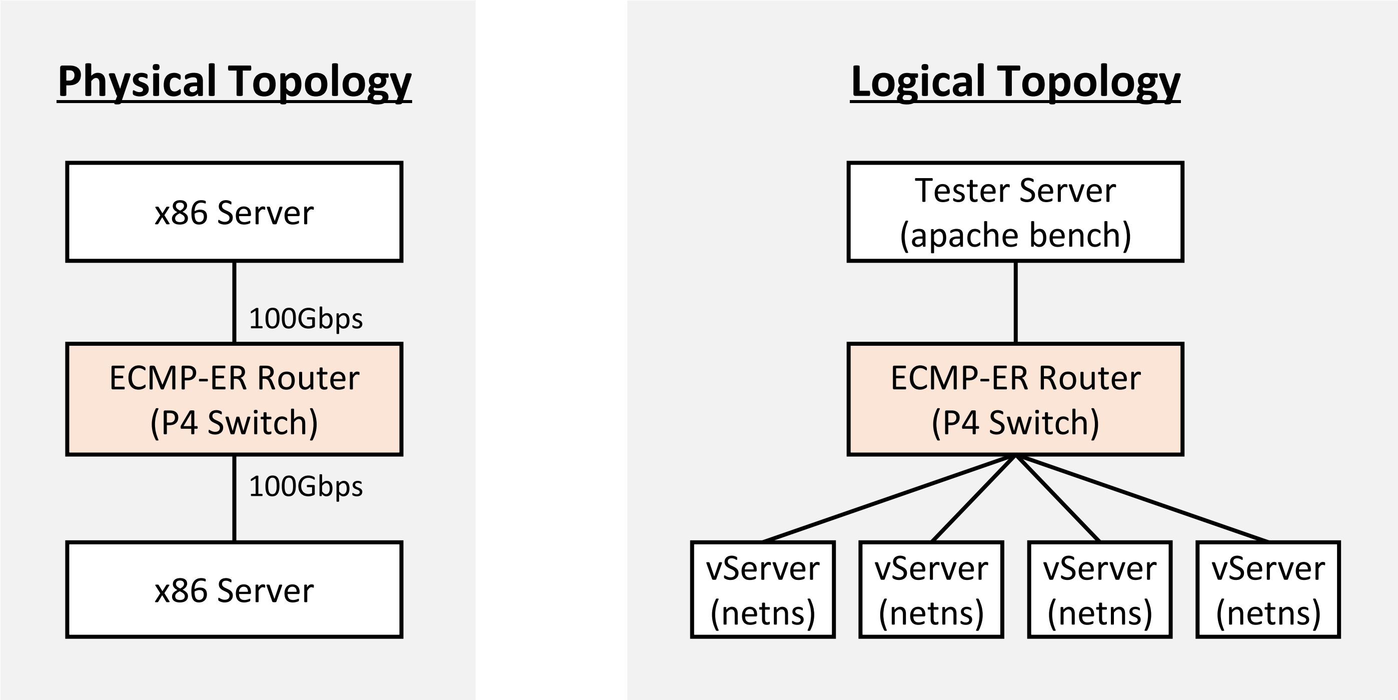
Evaluate if ECMP-ER works as expected

Eval#2 Churn Resistance

Server add/remove frequency with PCC ensured

Evaluation Environment

- Two x86 Servers + Wedge100BF-32X (Tofino)
- Run HTTP server inside virtual Server (netns running nginx)
  - SR-IOV ethernet dev is attached to each vServer (netns/nginx)
  - Send HTTP Request from another server using apache bench

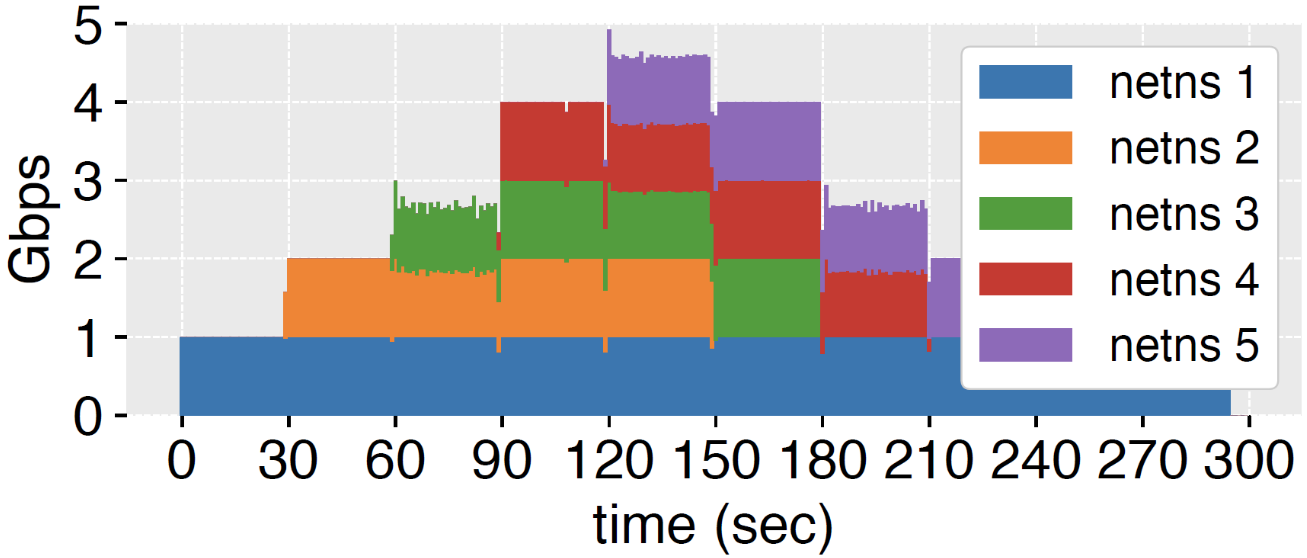


Eval #1 Operational Verification (with ECMP-ER)

1. Apply load to VIP from test server using apache bench
2. Add / Remove vServer (netns running nginx) every 30 sec

Notes

- Load from apache bench was HTTP GET to 1MB file
- 1Gbps rate limit was applied to SR-IOV VF to simulate server performance limit

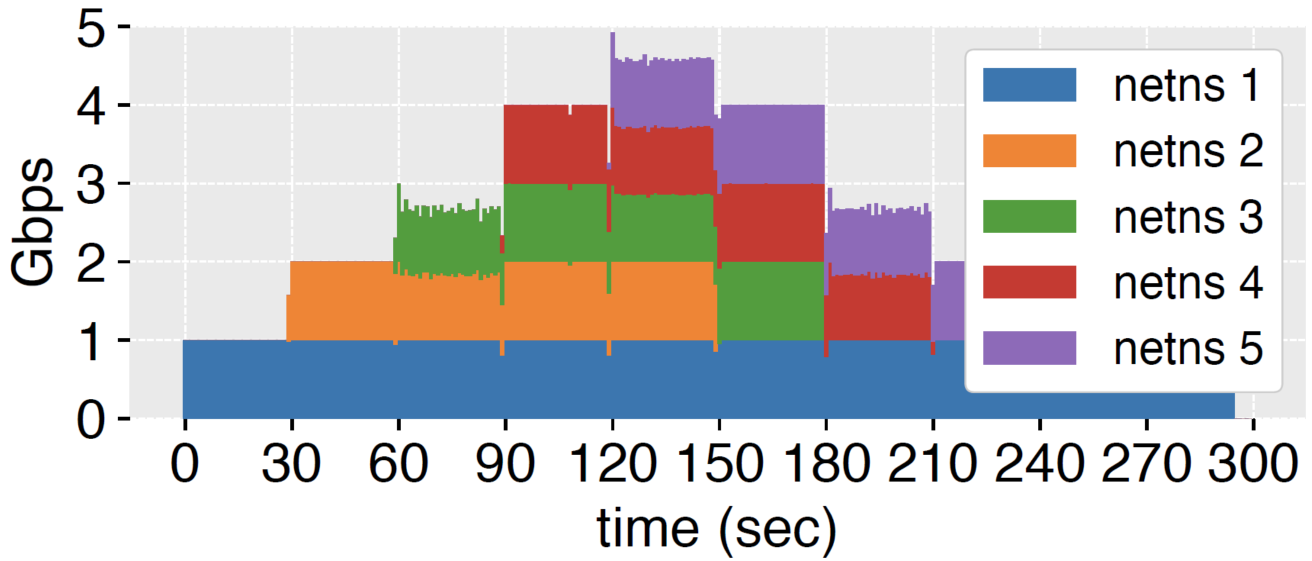



```
Concurrency Level:      96
Time taken for tests:   300.001 seconds
Complete requests:      85678
Failed requests:        0
```

Failed requests was zero (0) even when server was dynamically added / removed

Eval #1 Operational Verification (with ECMP)

- Run same test with ECMP-ER disabled
  - Remove XDP program attached to VFs
  - No retransmit by server
- Connections whose server (nexthop) changed will be disconnected
  - RST sent from Server




```
Concurrency Level:      96
Time taken for tests:   300.001 seconds
Complete requests:      86043
Failed requests:        1092
```

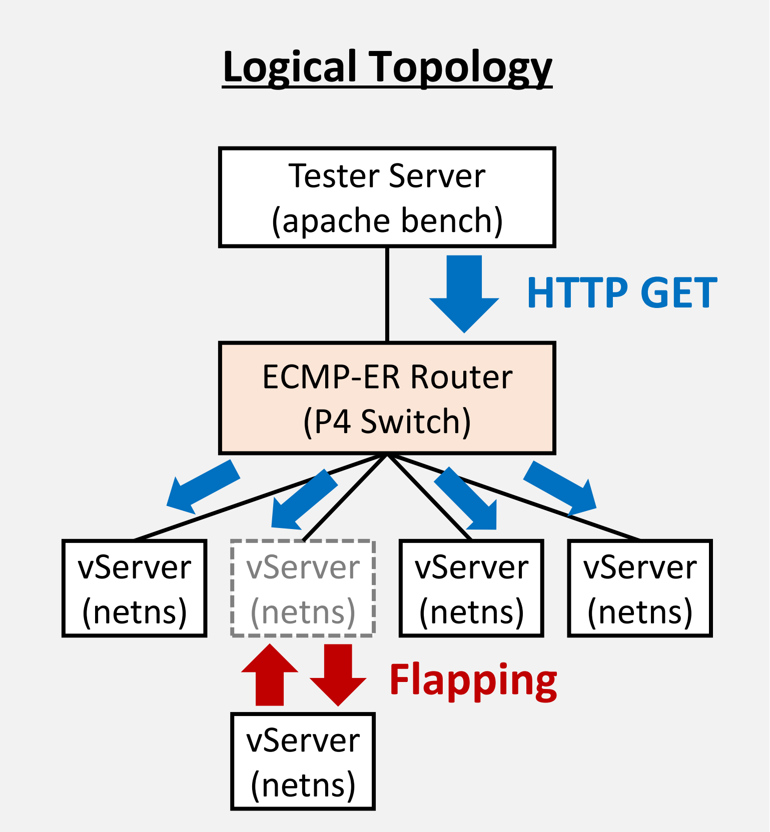
Request will fail due to TCP RST when server was dynamically added / removed

Eval#2 Churn Resistance

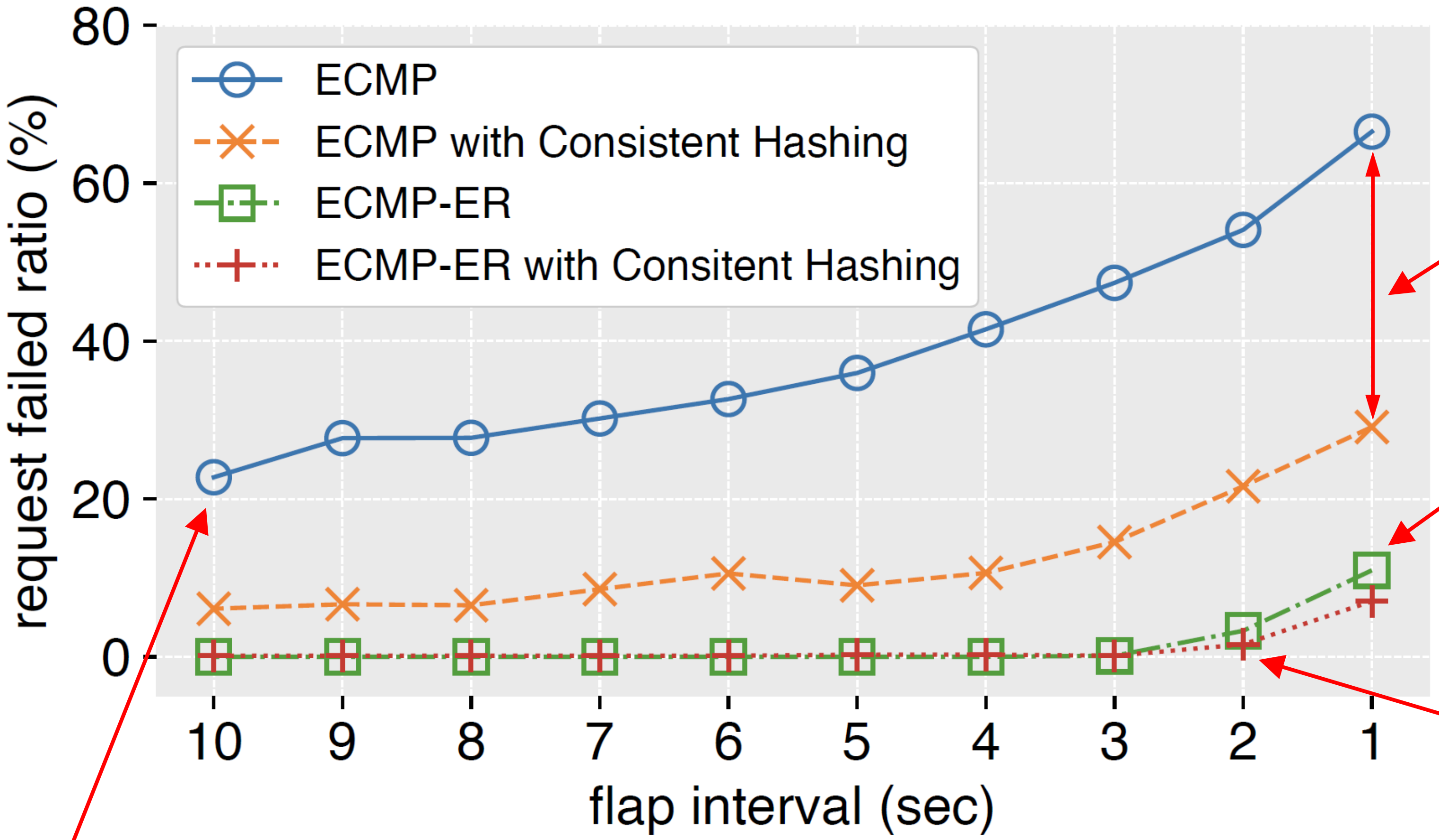
- Frequently up/down (flapping) server and observe Failure rate of HTTP Request
  - repeate removing randomly selected server (nginx) and restore after certain period of time

<u>Two types of ECMP Hash algorithm</u>

- Trafitional Hasing (modulo)
  - nexthop of all flow will change after server add/remove
- Consistent Hashing
  - nexthop is less likely to change after server add/remove



Eval#2 Churn Resistance (result)



Consistent Hash was effective
66% (modulo) -> 29% (Consistent)

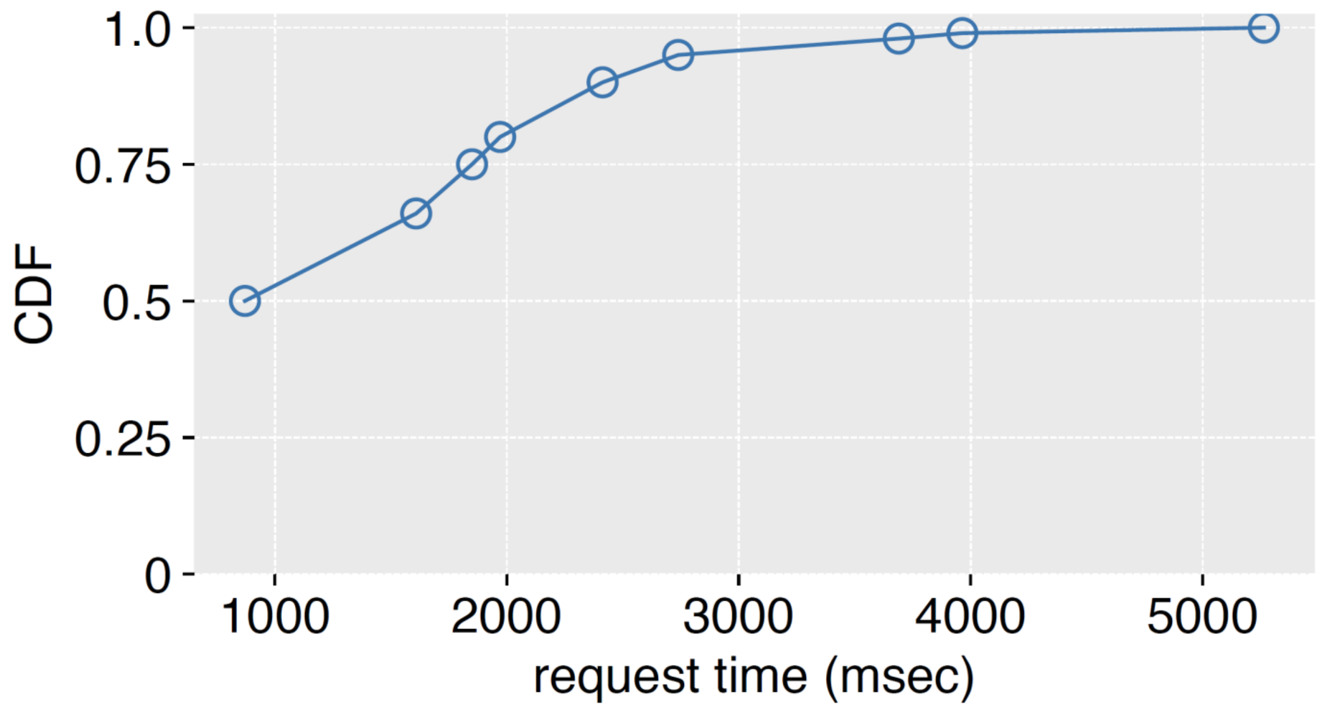
Consistent Hash was effective for ECMP-ER as well

Request did not fail for if flap interval was equal to or more than 3 sec for ECMP-ER

More than 20% of GET request has failed for Normal ECMP when flap happens every 10sec.
This rate will increase as flap happens more frequently (shorter interval).

Case when TCP was disconnected for ECMP-ER

- ECMP-ER can protect connections that can be forwarded correctly using a previous generation of the forwarding table
- Connection will be disconnected if nexthop changes more than two times within connection lifetime (duration to complete GET request)

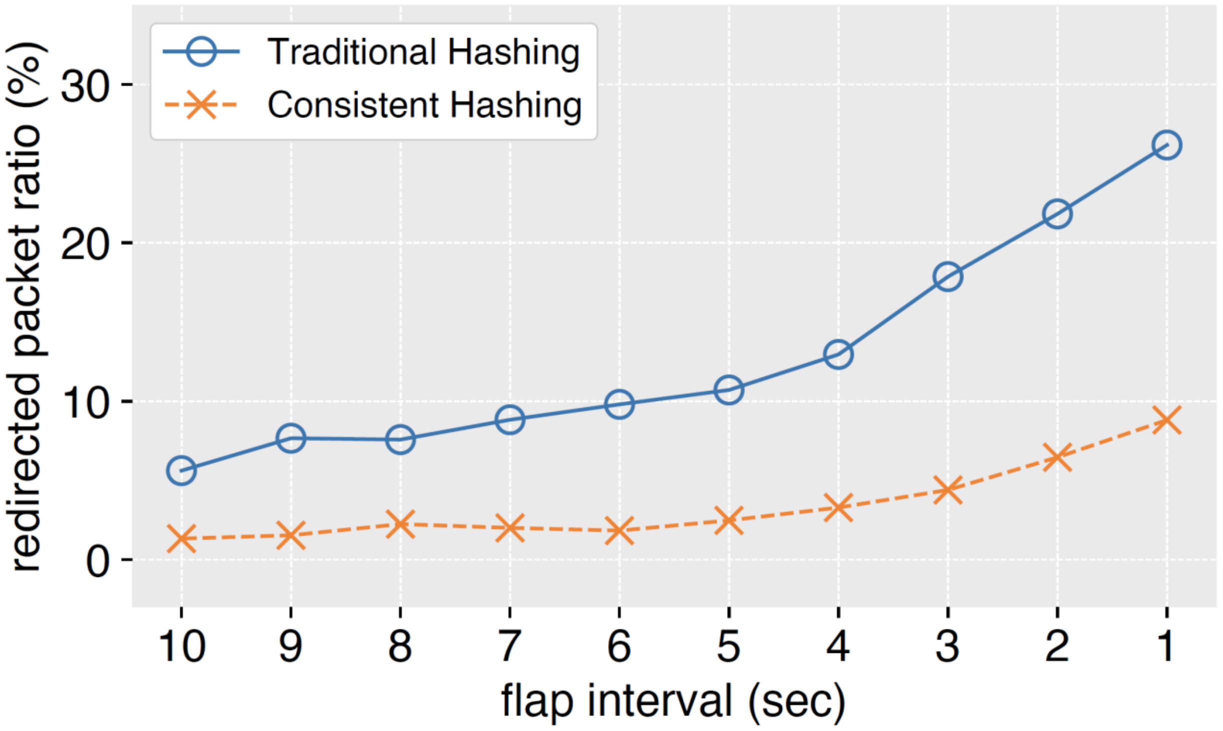


Distribution of time required to complete GET request

- 50% requests will be completed within 1 sec
- Top 1% of requests takes more than 3.9 sec
- Connection will fail if a request duration is longer than "nexthop change interval" times "number of generations (2 for this evaluation)"

Impact of retransmit packet by ECMP-ER enabled Server

Consistent Hashing can reduce number of retransmitted packets



Raito of retransmitted packet from Server to ECMP-ER router

- raito = retransmitted packet / received packet
- Traditional Hashing: 5.6% (10sec) ~ 26% (1sec)
- Consistent Hashing: 1.3% (10 sec) ~ 8.8% (1 sec)

Conclusion

- ECMP-ER
  - Load Balancing method which can ensure PCC on hardware (ASIC/NPU) without state
  - Redirect packet to correct server using nexthop information of past generation(s)
  - No extra control mechanism required for Load Balancer

- Evaluation Result
  - ECMP-ER can maintain connection if interval of nexthop change (server add/remove) is longer than connection duration.
  - Even in situation where 20% of connections are disconnected with ECMP, ECMP-ER was able to forward packet without any loss.

- Future Work
  - Application to larger scale network and detailed evaluation
  - Implemtation to platforms other than P4 capable platform