# SilkRoad

## Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs


Rui Miao

James Hongyi Zeng, Jeongkeun Lee, Changhoon Kim, Minlan Yu

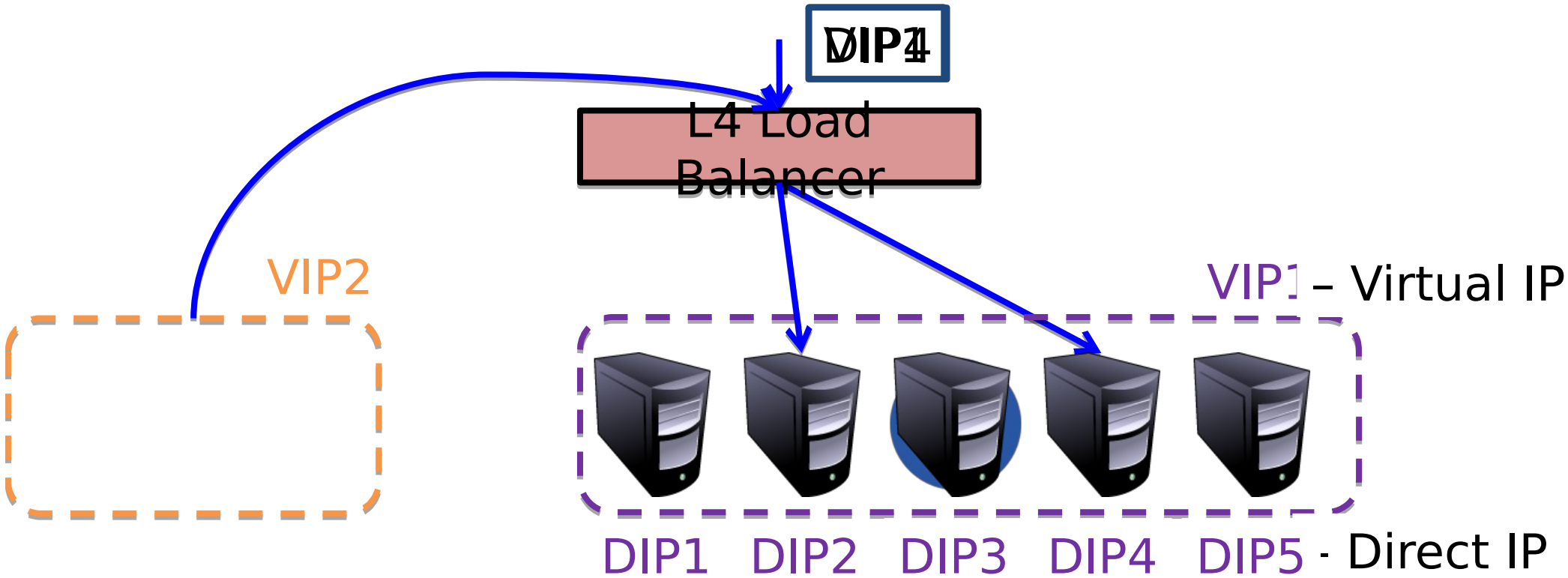
USC University of Southern California

facebook

BAREFOOT NETWORKS

Yale University

# Layer-4 Load Balancing




## Layer-4 load balancing is a critical function

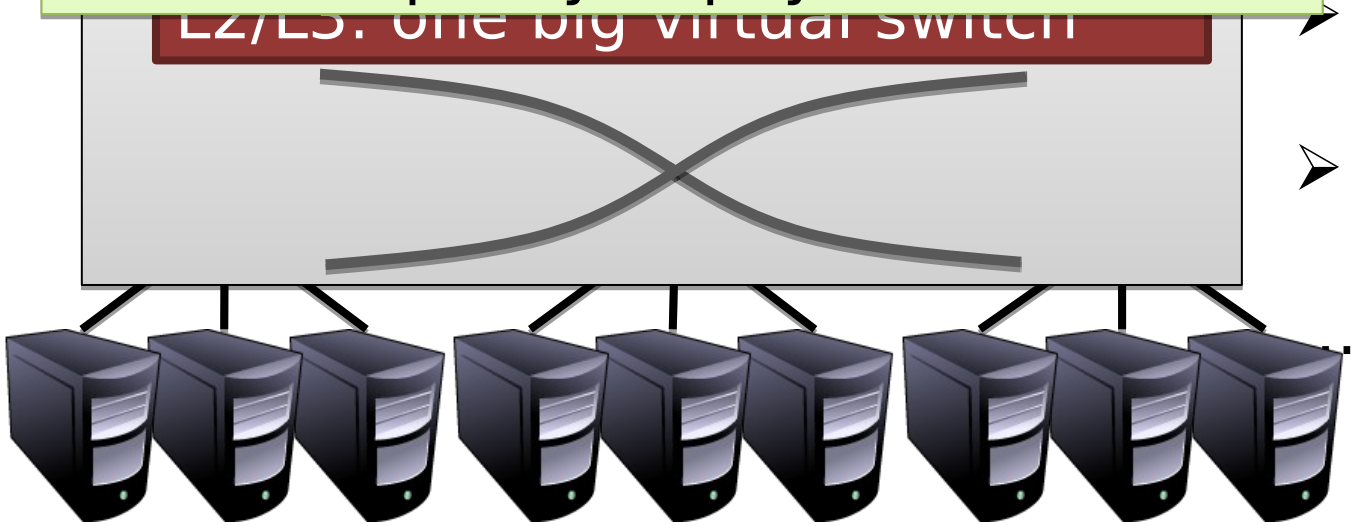
- handle both inbound and inter-service traffic
- >40%* of cloud traffic needs load balancing (Ananta [SIGCOMM'13])

# Scale to traffic growth

## Cloud traffic has a rapid growth

- doubling every year in Google, Facebook (Jupiter Rising [SIGCOMM'15])

L4: can we scale out load balancing to match
the capacity of physical network?




- Multi-rooted topology
- Datacenter transport

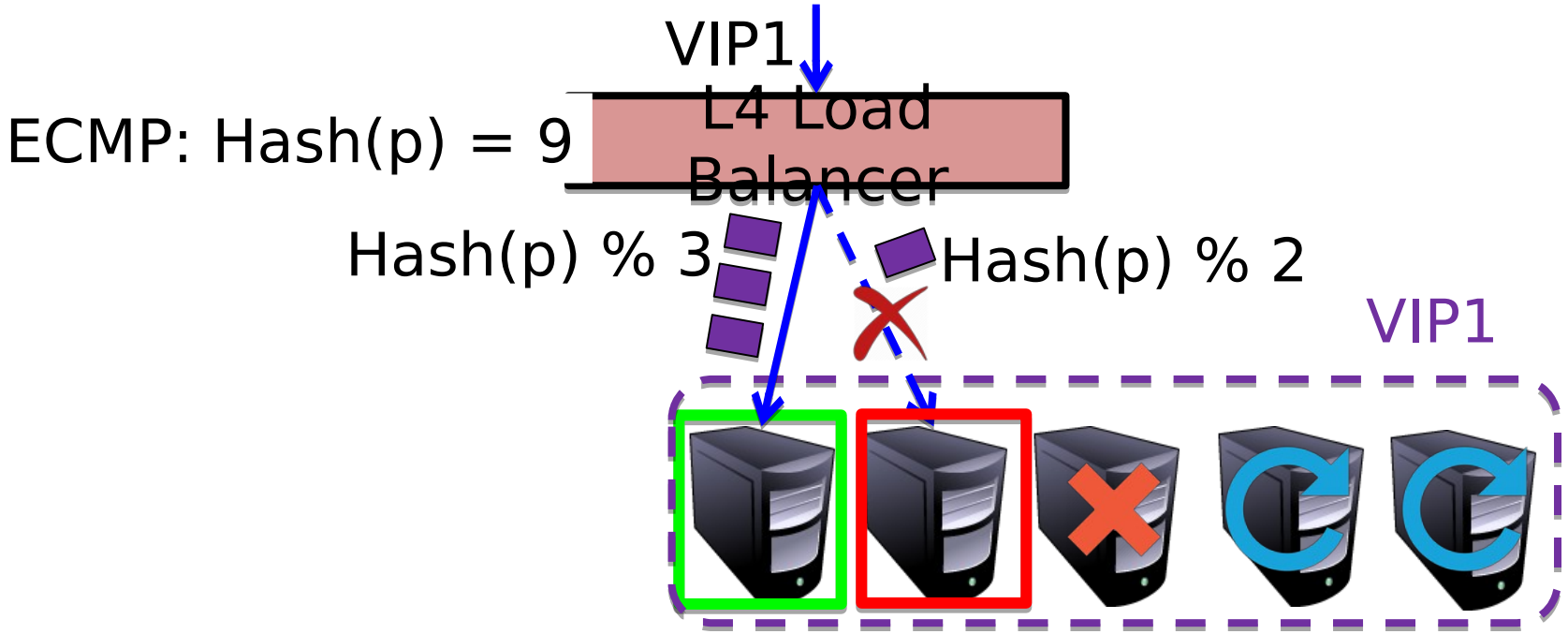
# Frequent DIP pool updates

## DIP pool updates

- failures, service expansion, service upgrade, etc.
- up to 100 updates per minute in a Facebook cluster

## Hash function changes under DIP pool updates

- packets of a connection get to different DIPs
- connection is broken



# Per-connection consistency (PCC)

Broken connections degrade the performance of cloud services

- tail latency, service level agreement, etc.

PCC: all the packets of a connection go to the same DIP

L4 load balancing needs connection states

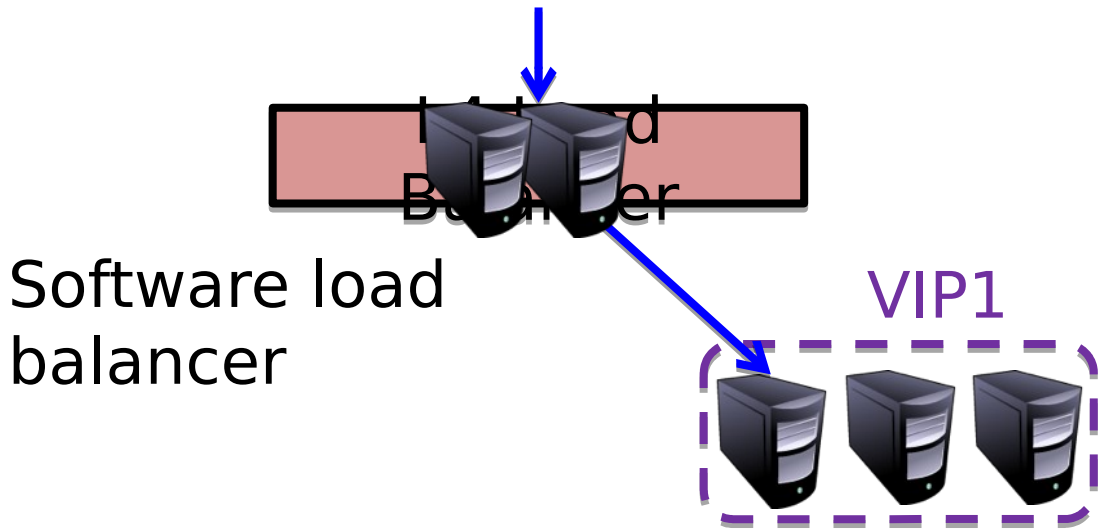
# Design requirements

Scale to traffic growth

While ensuring PCC under frequent DIP pool updates

# Existing solution 1: use software server

Ananta [SIGCOMM'13]
Maglev [NSDI'16]




✗ Scale to traffic growth
✓ PCC guarantee

## High cost

- 1K servers (~4% of all servers) for a cloud with 10 Tbps

## High latency and jitter

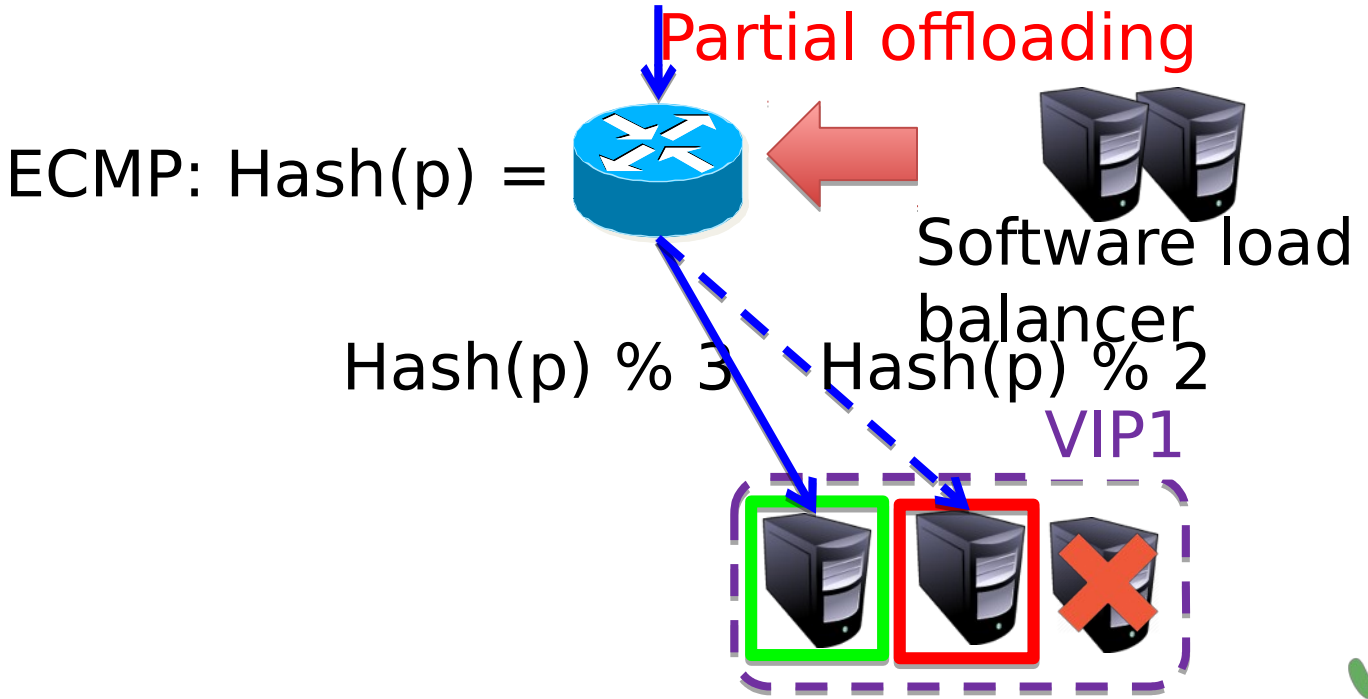
- add 50-300 µs delay for 10 Gbps in a server

## Poor performance isolation

- one VIP under attack can affect other VIPs

# Existing solution 2: partially offload to switches

Duet [SIGCOMM'14]
Rubik [ATC'15]



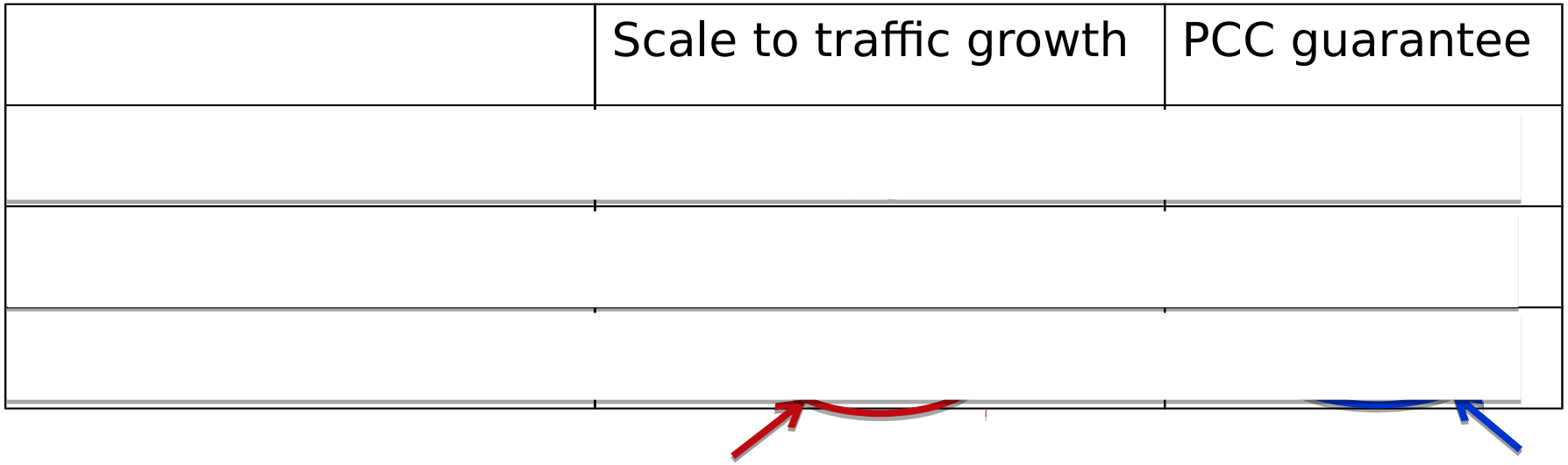

✓ Scale to traffic growt
✗ No PCC guarantee

## Hash function changes under DIP pool updates

- switch does not store connection states

# SilkRoad

Address such challenges using hardware primitives



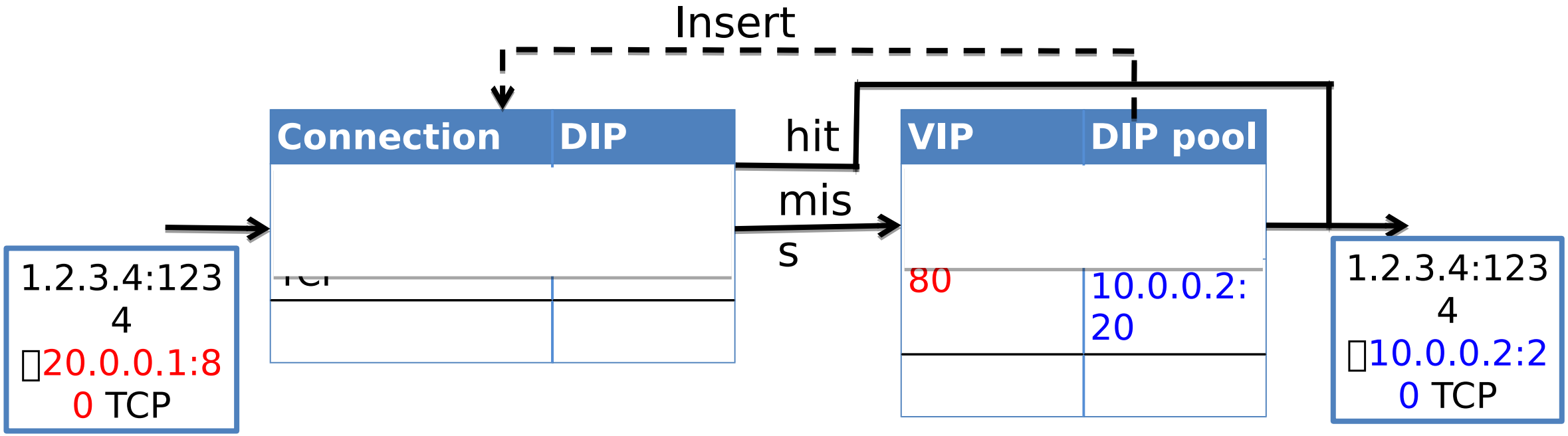

**Build on switching ASICs with multi-Tbps**

**Challenge: guarantee PCC under multi-Tbps**

# ConnTable in ASICs

**ConnTable**
store the DIP for each connection

**VIPTable**
store the DIP pool for each VIP



# Design challenges

Challenge 1: store millions of connections in ConnTable

Approach: novel hashing design to compress ConnTable

Challenge 2: do all the operations (e.g., PCC) in a few nanoseconds

Approach: use hardware primitives to handle connection state and its dynamics

# Many active connections in ConnTable

- Up to 10 million active connections per rack in Facebook traffic
  - a naïve approach: 10M * (37-byte 5-tuple + 18-byte DIP) = 550 MB

- ASIC features: storing all connection states just become possible
  - increasing SRAM size
  - emerging programmability allows to use SRAM flexibly

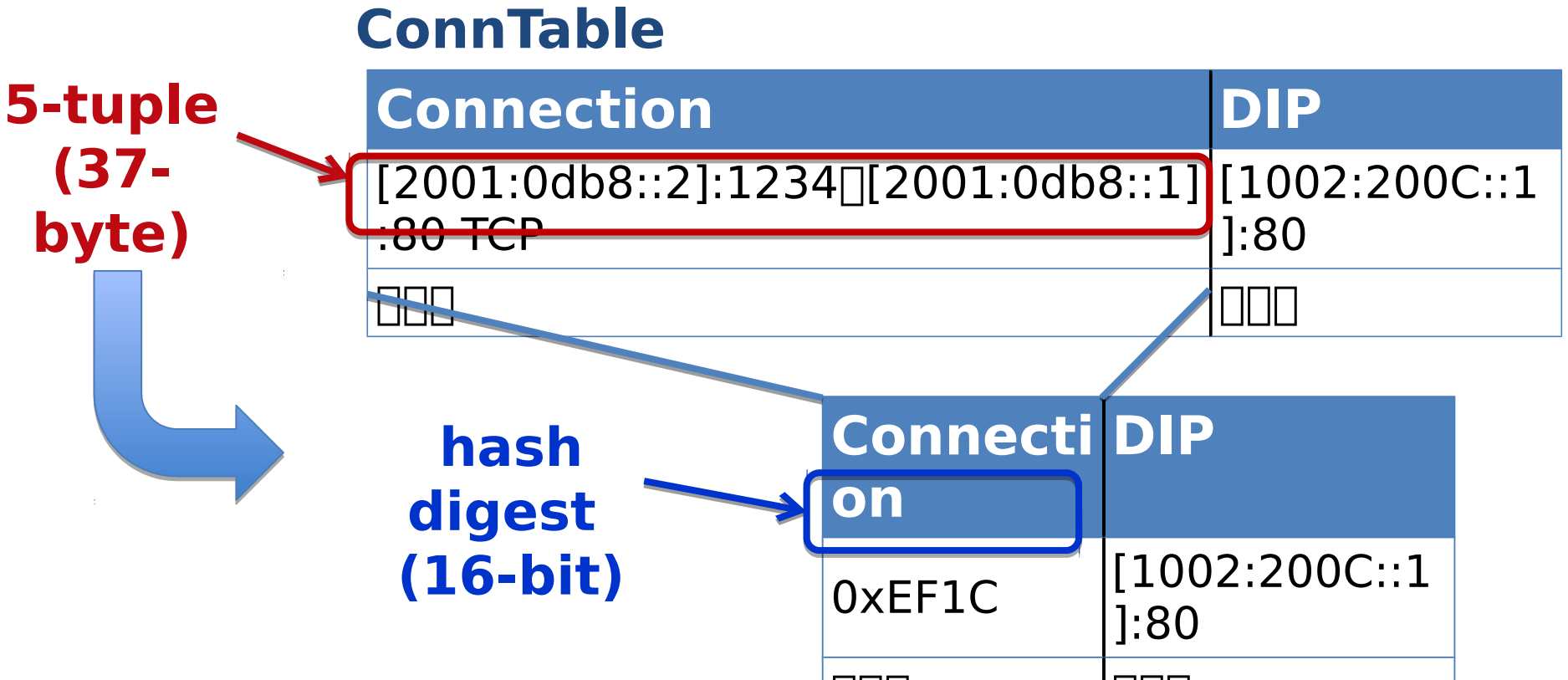
| Year | 2012 | 2014 | 2016 |
|---|---|---|---|
| SRAM (MB) | 10-20 | 30-60 | 50-100 |

# Approach: novel hashing design to compress ConnTable

Compact connection match key by hash digests

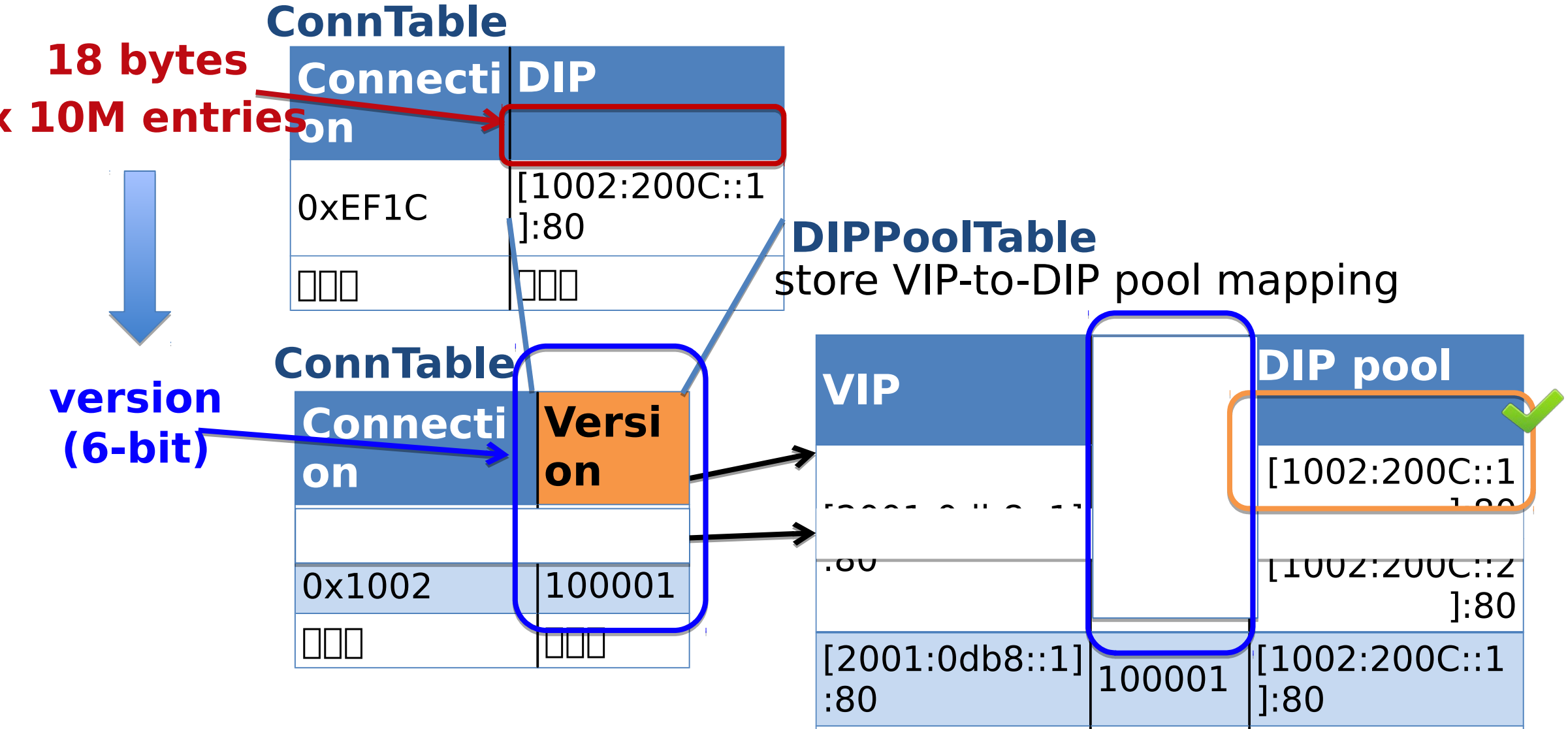
False positives caused by hash digests

- the chance is small (<0.01%)
- resolved via switch CPU (details in the paper)



# Approach: compress ConnTable

## Compact action data with DIP pool versioning



# Design challenges

Challenge 1: store millions of connections in ConnTable

Approach: novel hashing design to compress ConnTable

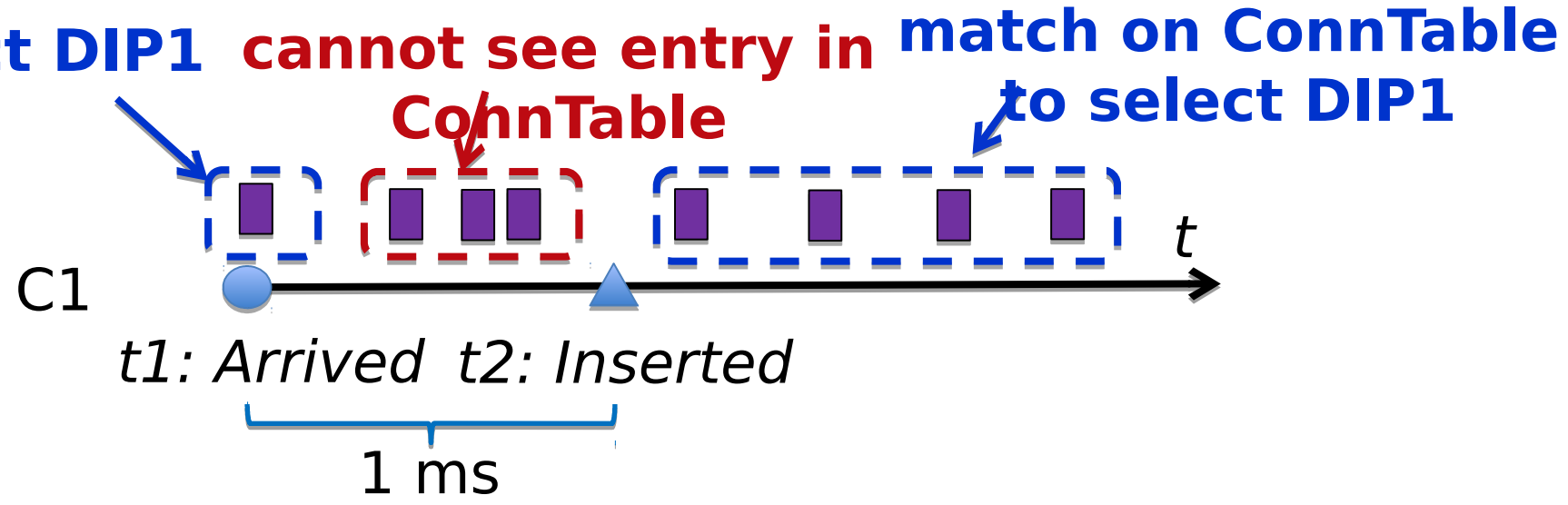
Challenge 2: do all the operations (e.g., PCC) in a few nanoseconds

Approach: use hardware primitives to handle connection state and its dynamics

# Entry insertion is not atomic in ASICs

## ASIC feature: ASICs use highly efficient hash tables

- fast lookup by connections (content-addressable)
- high memory efficiency
- but, require switch CPU for entry insertion, which is not atomic



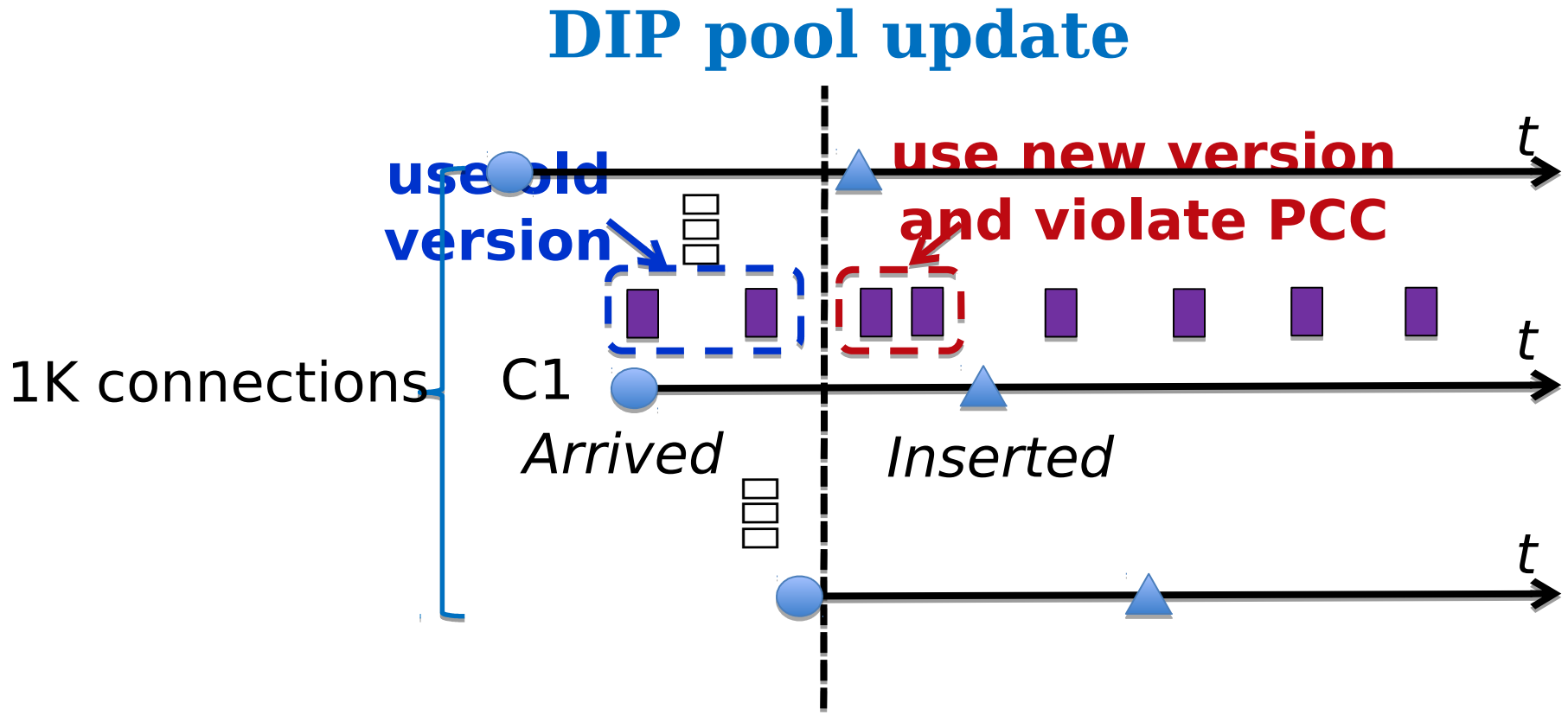

C1 is a pending connection between t1 and t2

# Many broken connections under DIP pool updates

DIP pool update breaks PCC for pending connections

Frequent DIP pool updates

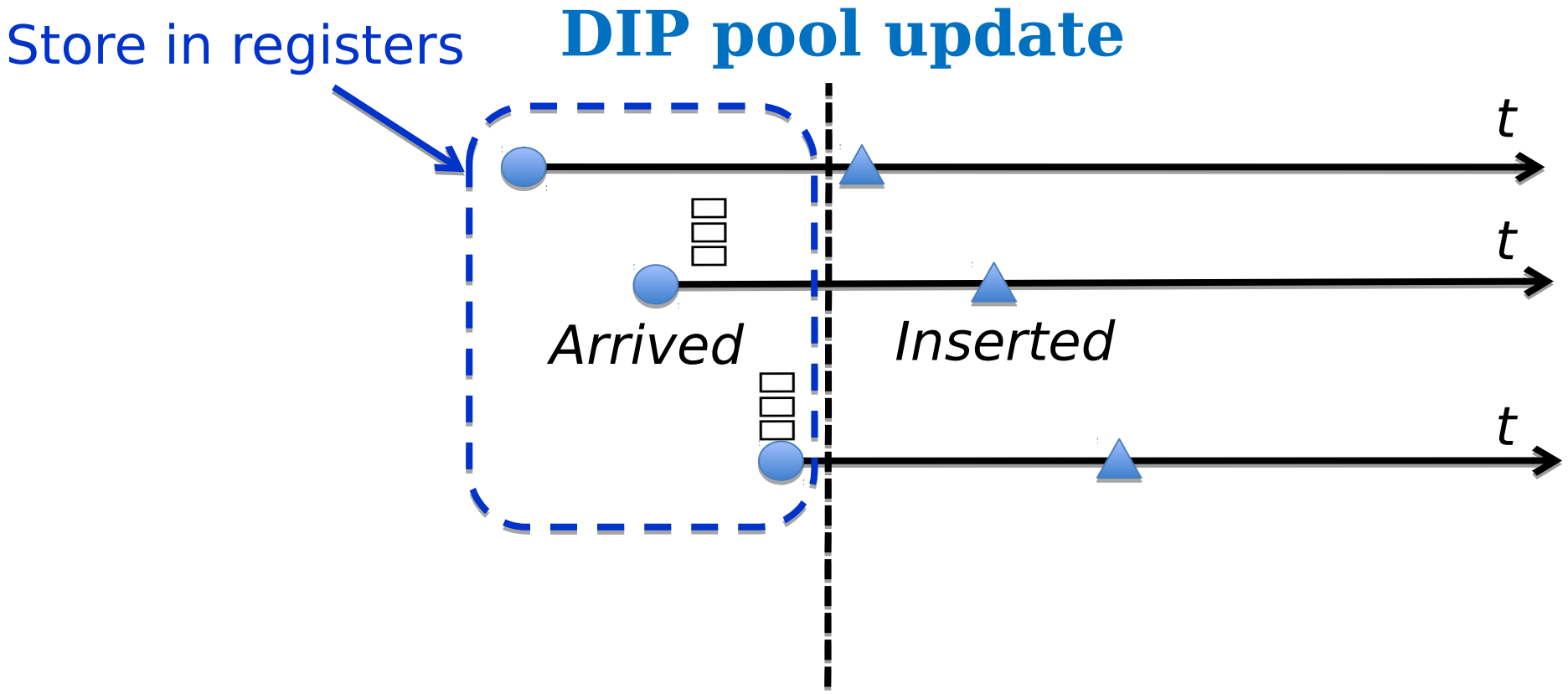
- a cluster has up to 100 updates per minute



# Approach: registers to store pending connections

## ASIC feature: registers

- support atomic update directly in ASICs
- store pending connections in registers



# Approach: registers to store pending connections

## Strawman: store connection-to-DIP mapping

- to look up connections, need *content addressable* memory
- but, registers are only *index-addressable*

## Key idea: use Bloom filters to separate old and new DIP pool versions

- store pending connections with old DIP pool version
- other connections choose new DIP pool version
- this is a membership checking, and only need index addressable

**Details in the paper**

# Prototype implementation

## Data plane in a programmable switching ASIC

- 400 lines of P4 code
- ConnTable, VIPTable, DIPPoolTable, Bloom filter, etc.

## Control plane functions in switch software

- 1000 lines of C code on top of switch driver software
- connection manager, DIP pool manager, etc.

# Prototype performance

## Throughput

- a full line rate of 6.5 Tbps
- one SilkRoad can replace up to 100s of software load balancers
- save power by 500x and capital cost by 250x

## Latency

- sub-microsecond ingress-to-egress processing latency

## Robustness against attacks and performance isolation

- high capacity to handle attacks
- use hardware rate-limiters for performance isolation

## PCC guarantee

# Simulation setup

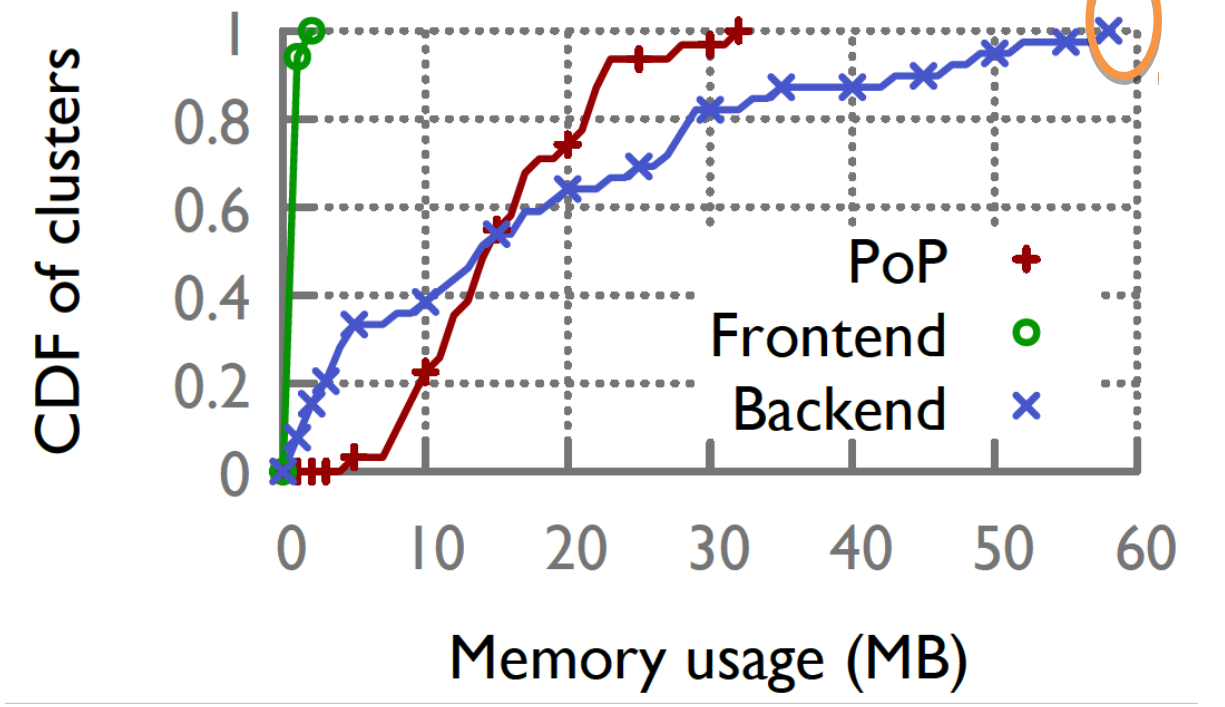
## Data from Facebook clusters

- about a hundred clusters from PoP, Frontend, and Backend
- One month of traffic trace with around 600 billion connections
- One month of DIP pool update trace with around three millions update

## Flow-level simulation

- run SilkRoad on all ToR switches
- 16-bit digest and 6-bit version in ConnTable

# SilkRoad can fit into switch memory




use up to 58MB SRAM
to store 15M connections

switching ASICs have 50-100 MB SRAM

# Conclusion

Scale to traffic growth with switching ASICs

High-speed ASICs make it challenging to ensure PCC

- limited SRAM and limited per-packet processing time

SilkRoad: layer-4 load balancing on high-speed ASICs

- a line rate of multi-Tbps
- ensure PCC under frequent DIP pool updates
- 100-1000x saving in power and capital cost

# Thank You!

## Please come and see our demo

- Implemented using P4 on Barefoot Tofino ASIC

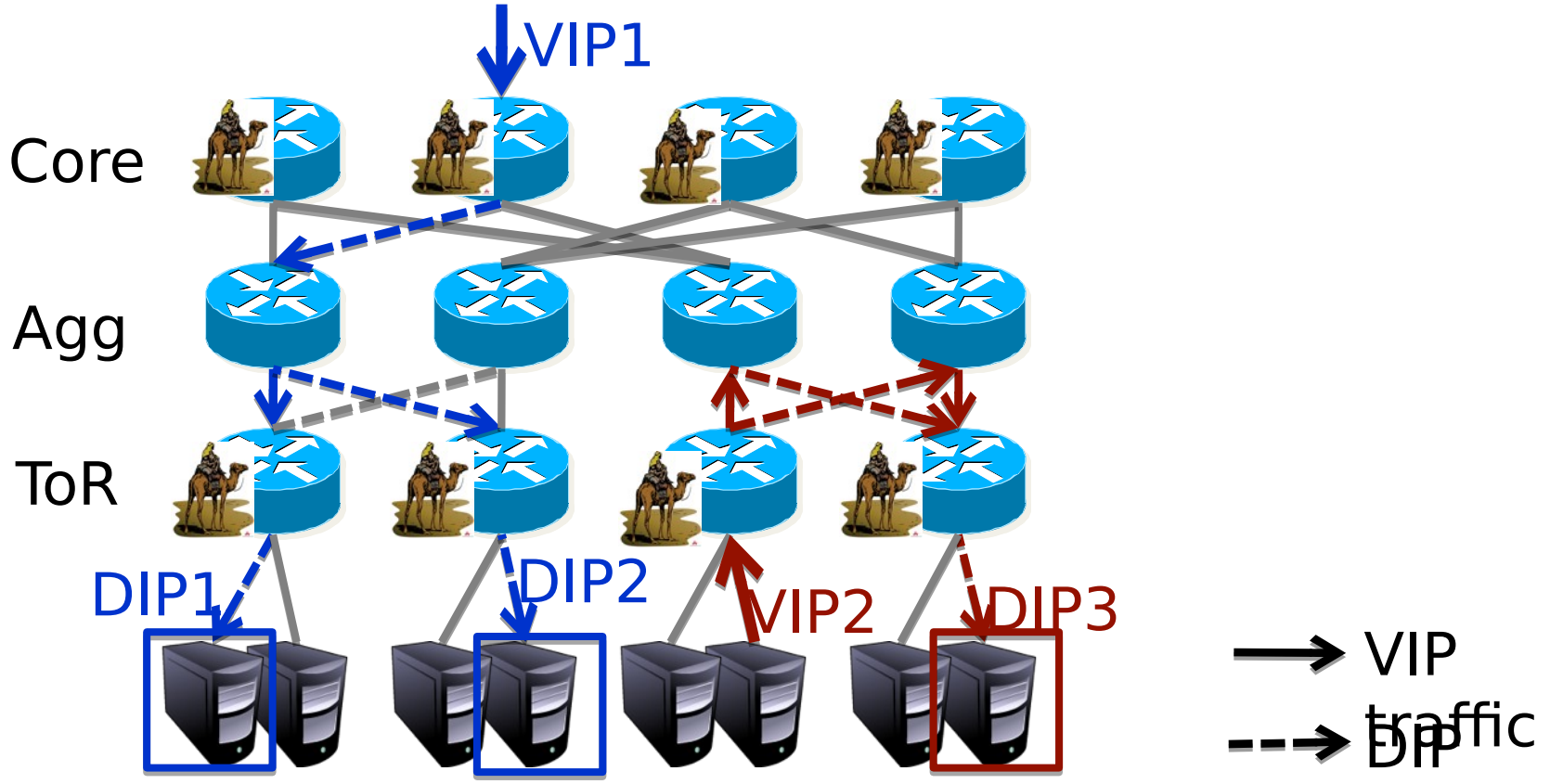
- Time: Tuesday (August 22), 10:45am - 6:00pm
- Location: Legacy Room

# BACK UP

# Network-wide deployment

## Simple scenario: at all the ToR switches and core switches

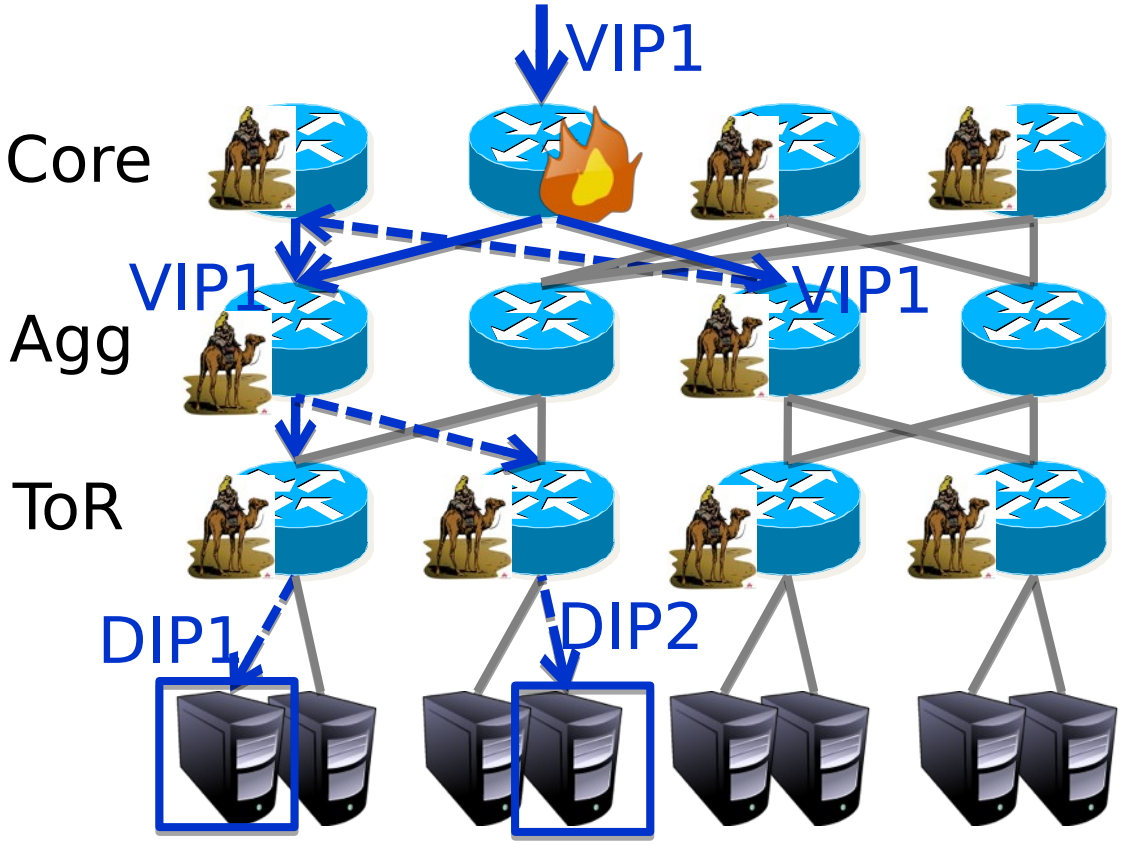
- each SilkRoad switch announces routes for all the VIPs
- all inbound and intra-datacenter traffic is load-balanced at its first hop



# Network-wide deployment

Harder scenarios: network-wide load imbalance, limited SRAM budget, incremental deployment, etc.

Approach: assign VIPs to different switch layers to split traffic




assign VIP1 to Agg switches

VIP assignment is a bin-packing problem