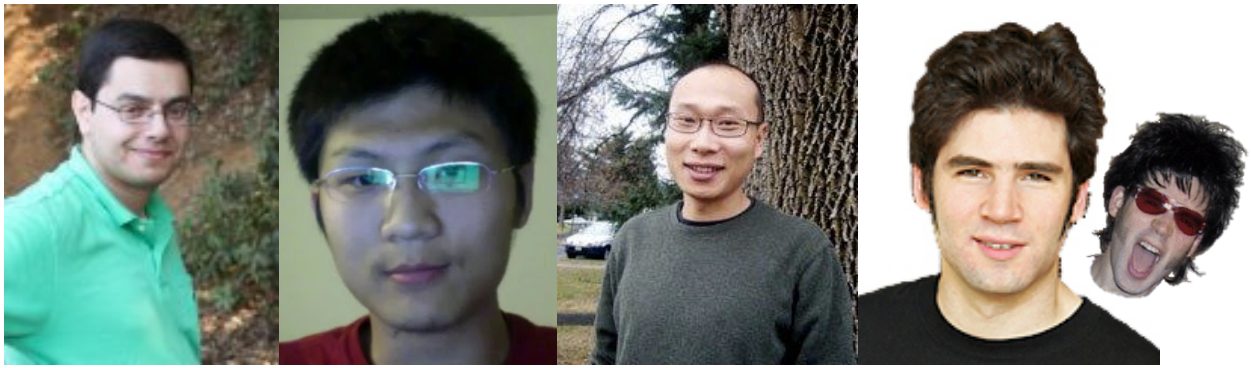
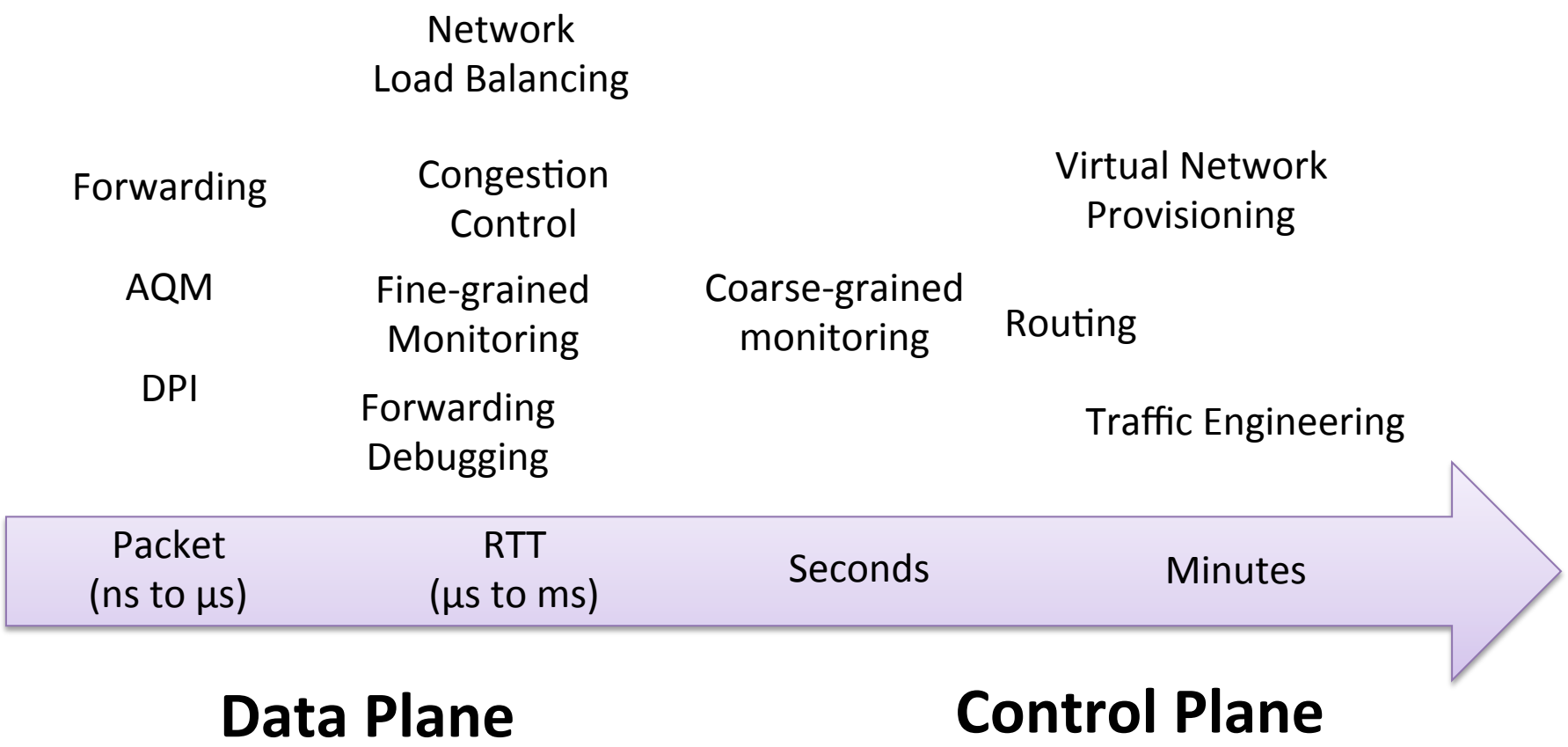
Tiny Packet Programs for low-latency network control

Vimal

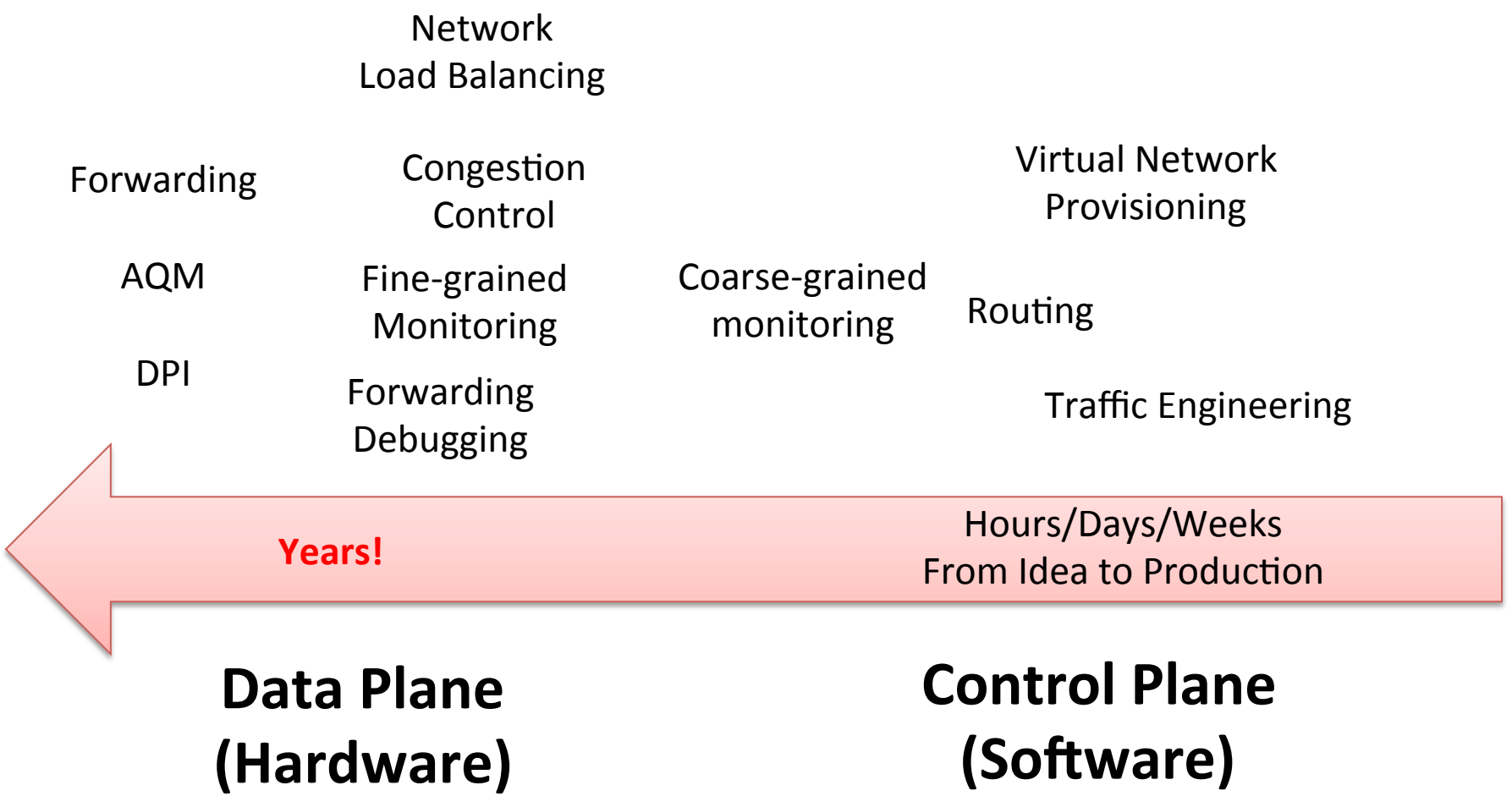
Mohammad Alizadeh, Yilong Geng, Changhoon Kim, David Mazières



Timescales of Network Functions



Timescales of Innovation/Deployment



Innovation: Where and How?

New Control Plane

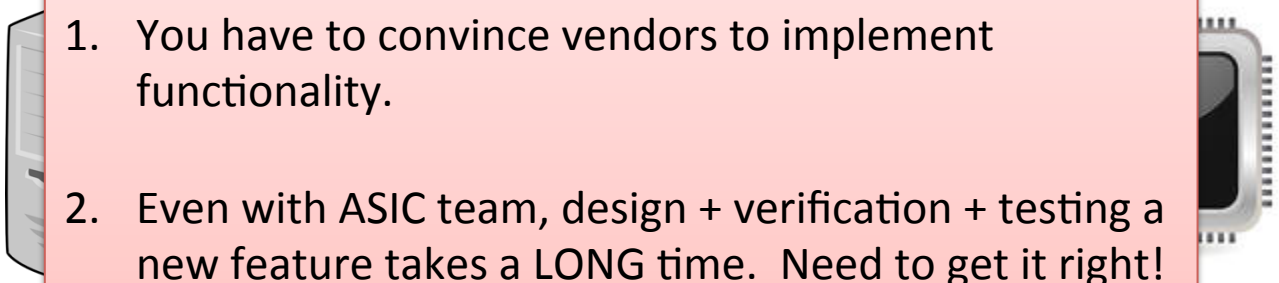
Fur

S

New Data Plane

ns in

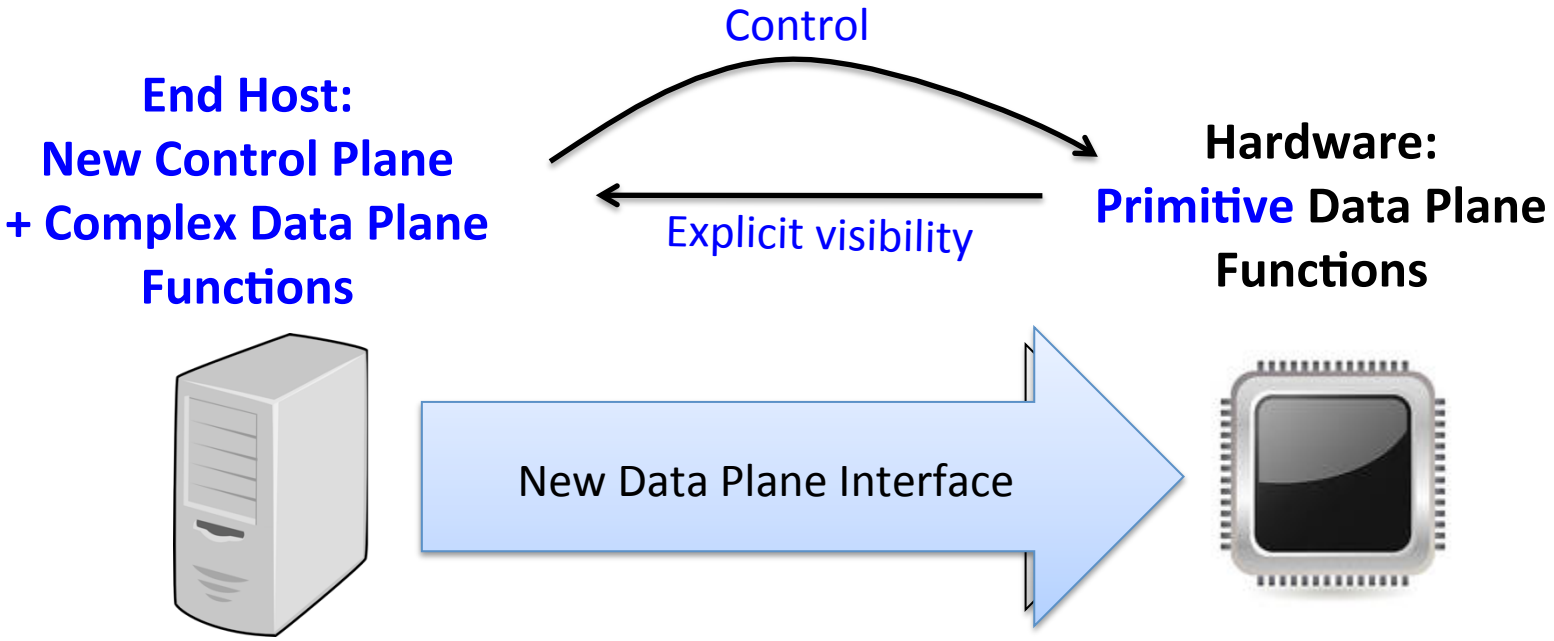
are



Data Plane Stagnation

1. You have to convince vendors to implement functionality.
2. Even with ASIC team, design + verification + testing a new feature takes a LONG time. Need to get it right!
3. Some functions have many "right" answers (e.g., congestion control).

This talk: Interface for programming Data Plane Functions



Key Idea: Empower end-hosts with low-latency, high throughput network visibility to carry out data plane functions at round-trip timescales.

Interface: Guiding Principles

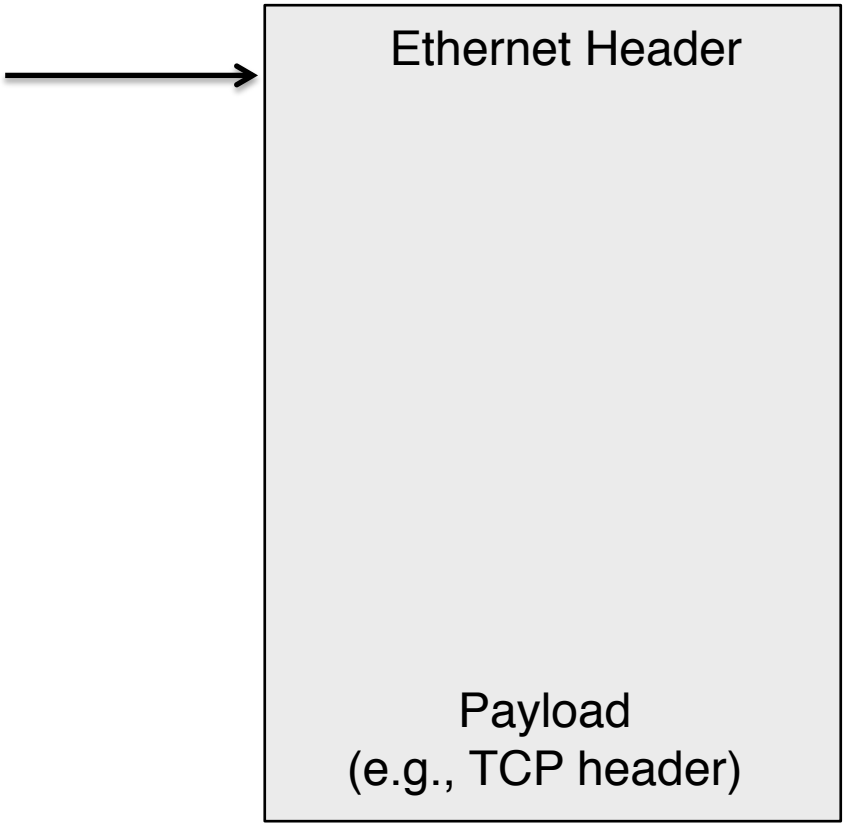
- Generic: Function-agnostic, increases chance of adoption across multiple switching vendors and platforms.

- In-band: Use data packets as your minions to tie network state *unambiguously* to specific packets (high throughput + low latency).

- Light-weight: Hardware is your friend, don't overwork it.

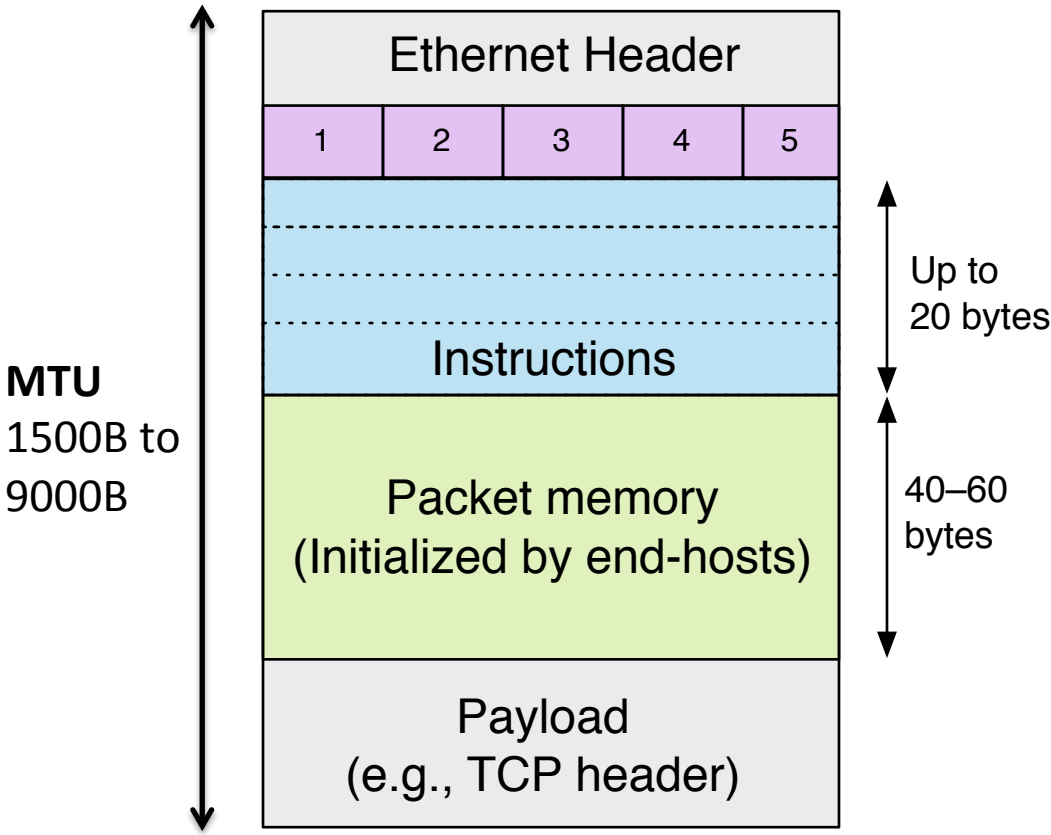
Interface: Tiny Packet Program

Regular packet, forwarded just like other packets.



Tiny Packet Program

Tiny relative to usual x86 programs.



(Not drawn to scale)

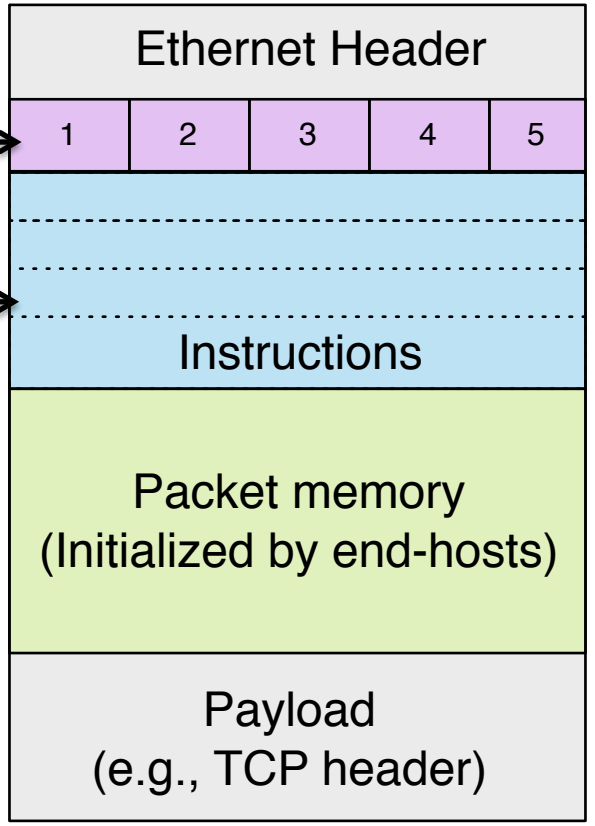
Tiny Packet Program

Control Fields (length, checksum, etc.)

Reads + Writes (optional)

Instruction	Purpose
LOAD/PUSH	Copy state from switch to packet
STORE/POP	Copy state from packet to switch
CSTORE	Conditional store
CEXEC	Conditional execute

Program = max 5 instructions with access to switch memory in hardware as described by a data sheet.



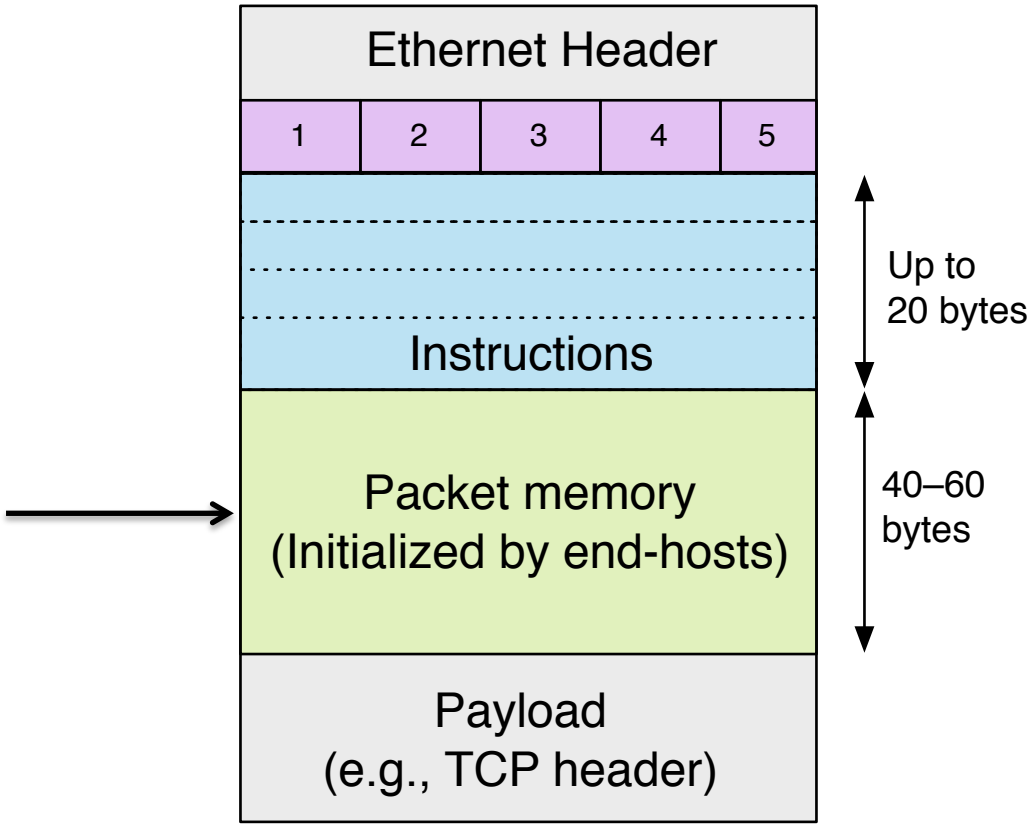
Useful State inside Switches

Namespace	Examples
Per-Switch	**Switch ID**, clock/uptime
Per-Port	**Link utilisation**, bytes sent/received, bytes enqueued, bytes dropped
Per-Queue	Bytes sent/received, **bytes enqueued**, bytes dropped
Per-Packet	**Input/output port**/queue, matched flow entry indices, packet fields

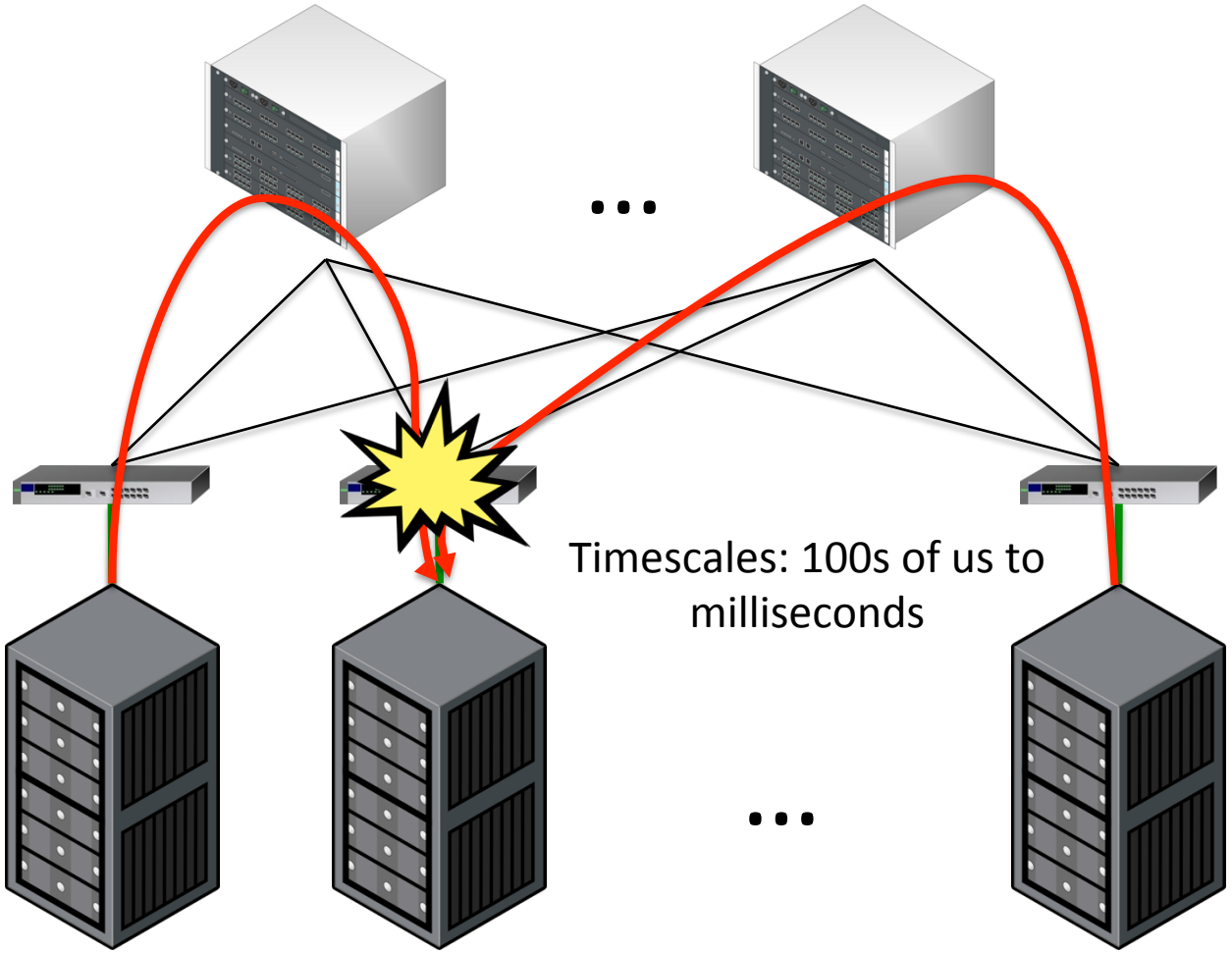
For more statistics, please check OpenFlow specification.

Tiny Packet Program

Packet memory contains space to load/store state.

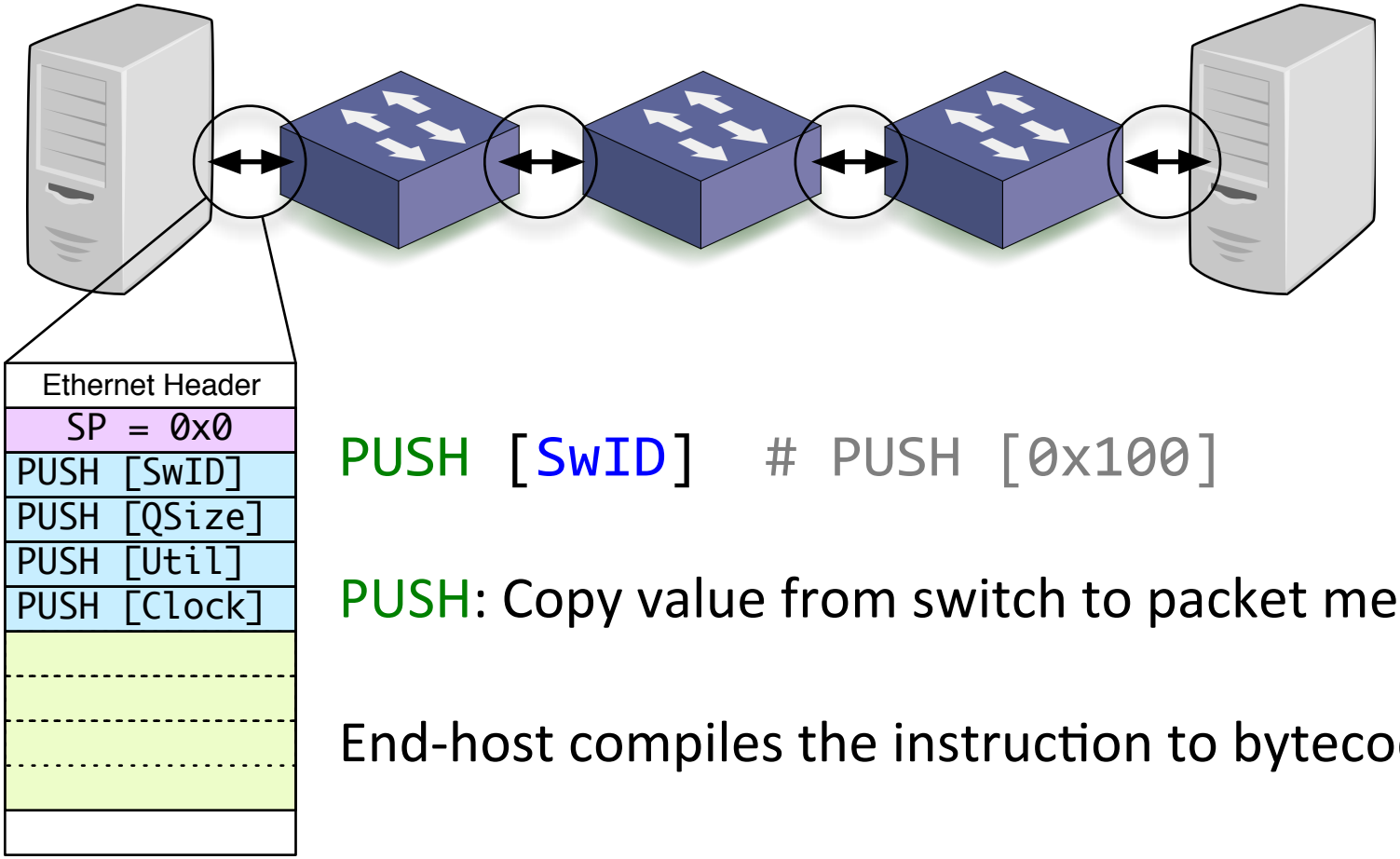


Localising Micro-Bursts*: Where and Which app?



*Incast: Vasudevan et al. SIGCOMM 2009

TPP workflow example

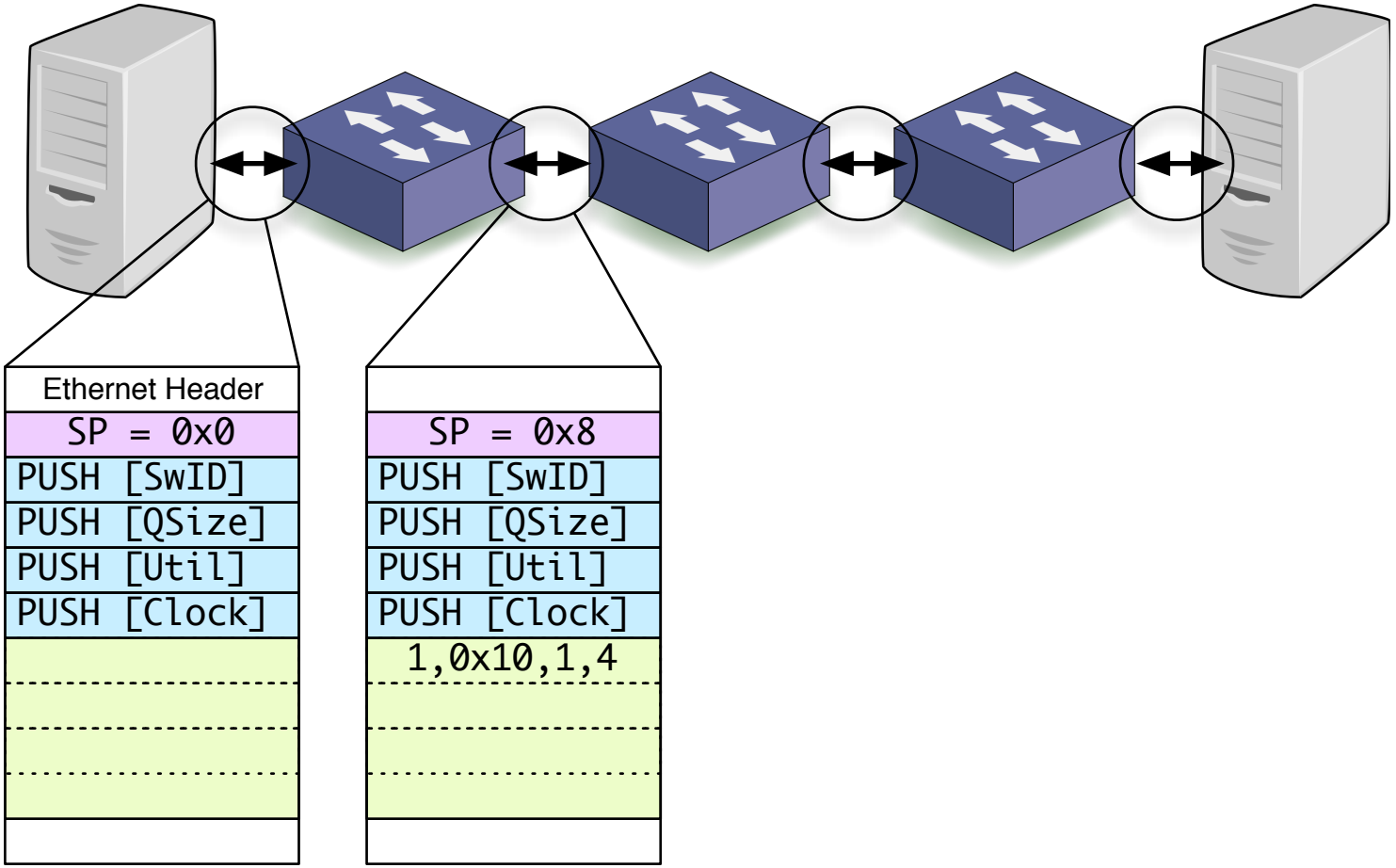



```
PUSH [SwID]   # PUSH [0x100]
```

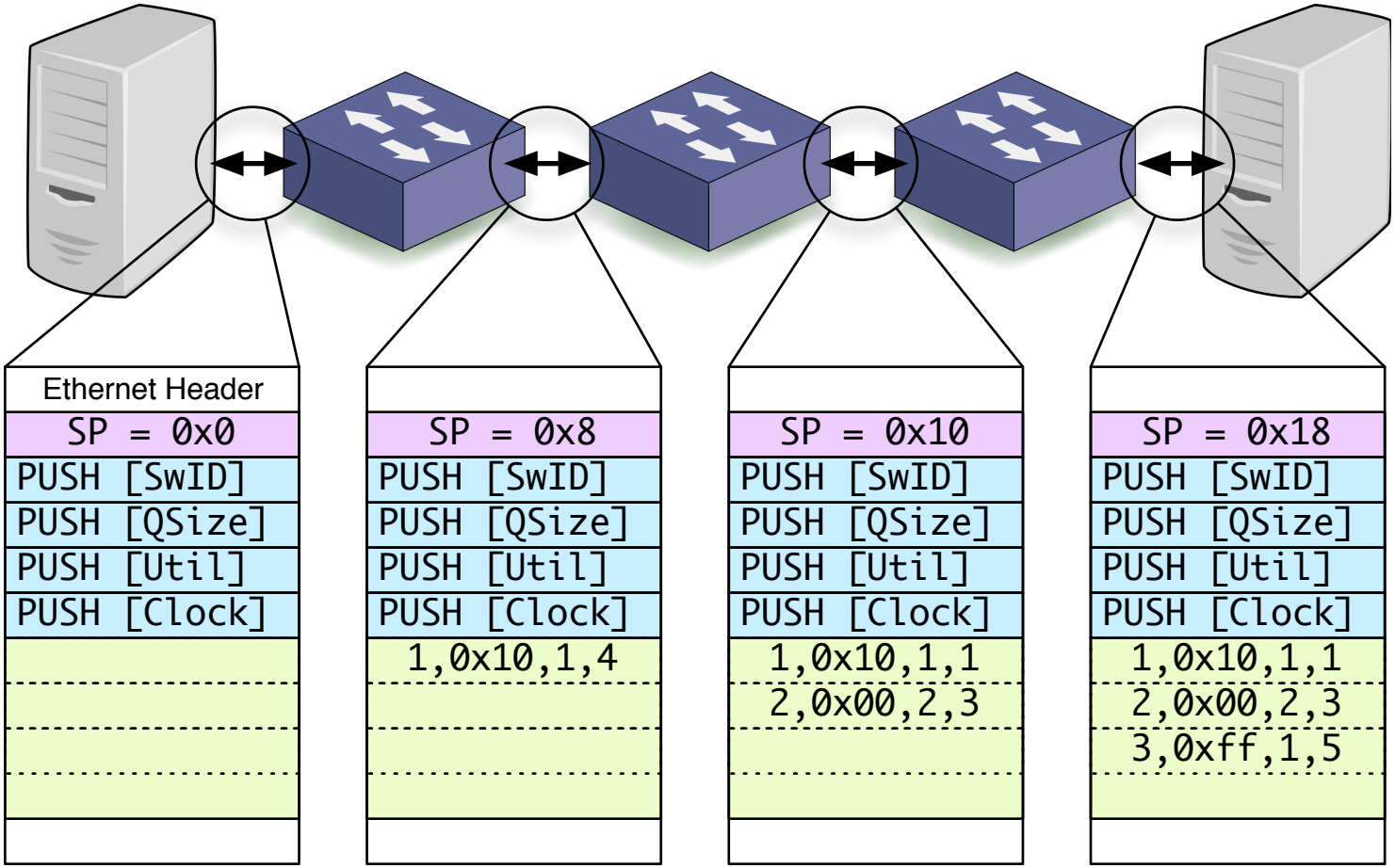
PUSH: Copy value from switch to packet memory.

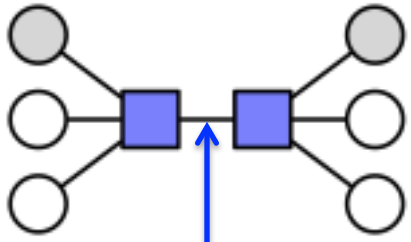
End-host compiles the instruction to bytecode.

Investigating Micro-Bursts



Investigating Micro-Bursts

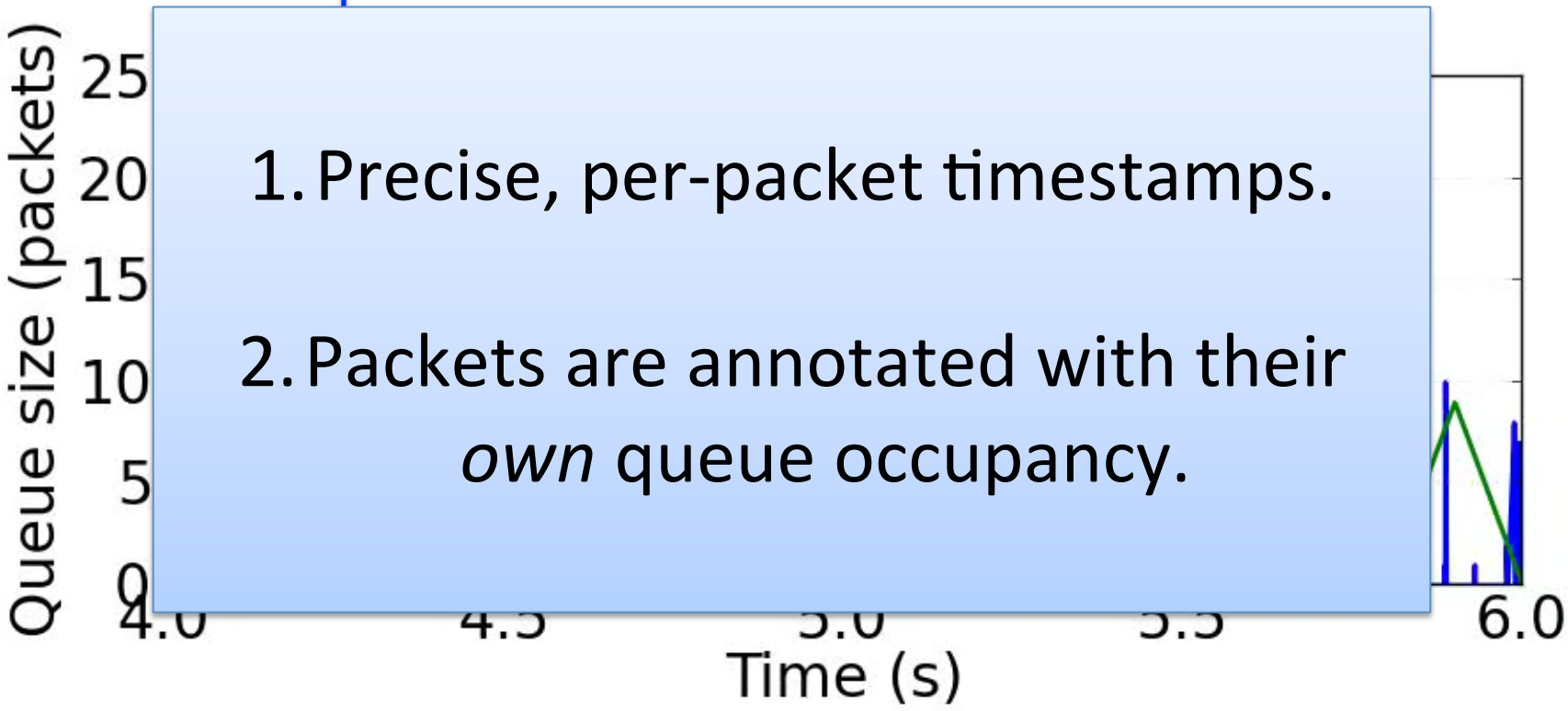




Mininet: All to all traffic at 30% load
Short (10kB) flows
TPP: 50 byte per packet for 5 hops.

1. Precise, per-packet timestamps.

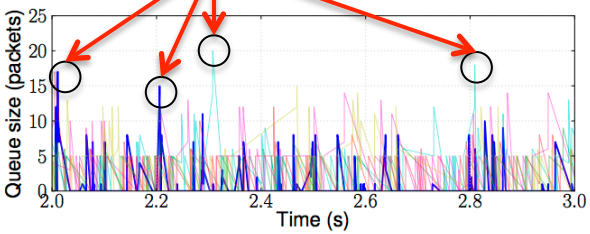
2. Packets are annotated with their *own* queue occupancy.



TPPs help Beyond Micro-Bursts

One interface, Many tasks

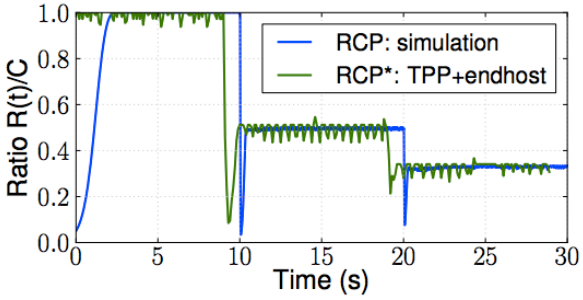
Micro-bursts



Identify competing applications

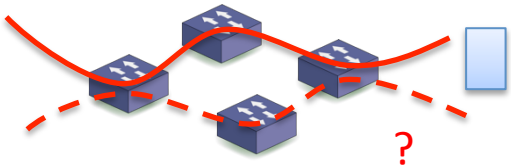
- Packets → Applications
- Understand traffic patterns

Congestion Control Algorithms: RCP



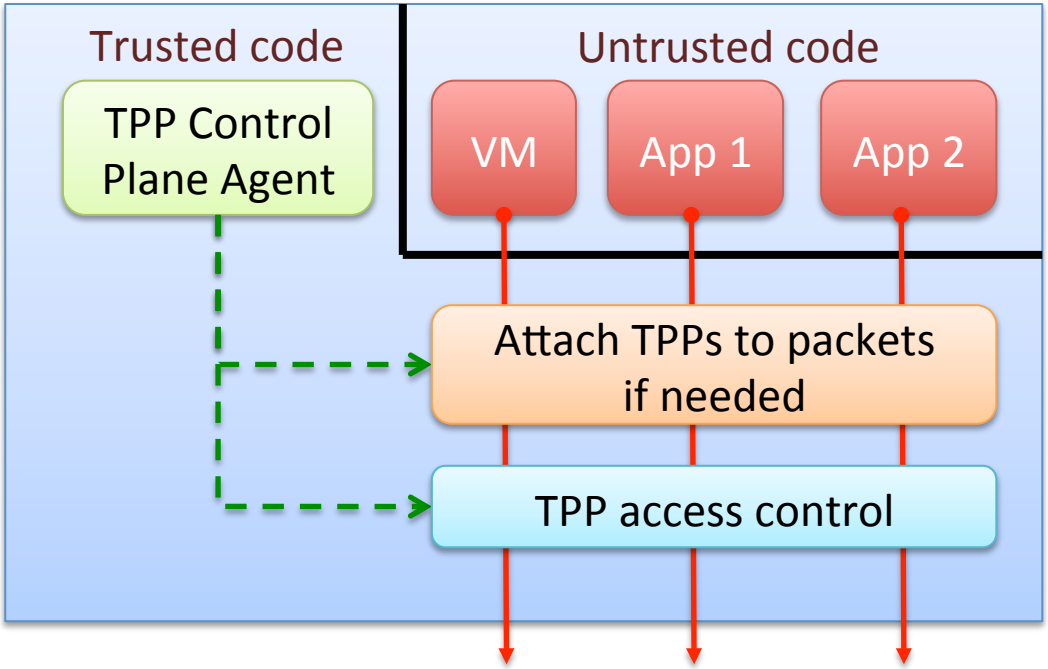
Use queue visibility for new congestion control algorithms. (Uses STORE instructions.)

Debugging Per-Packet Forwarding Behaviour: NetSight



Use per-packet visibility for detailed diagnosis.

TPP: End to End Picture



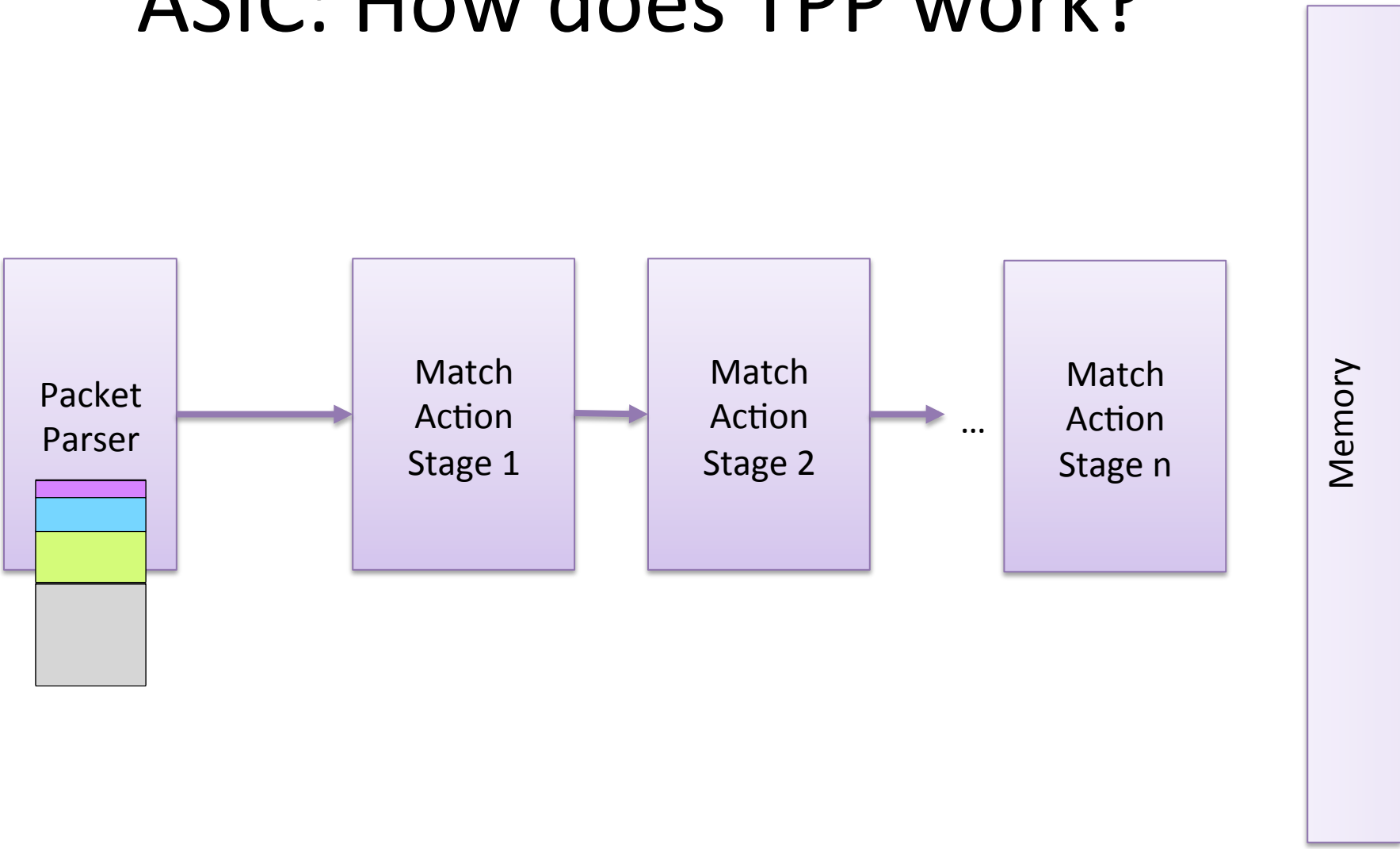
End Host Stack

- Create TPP apps
- Enforce fine-grained access control

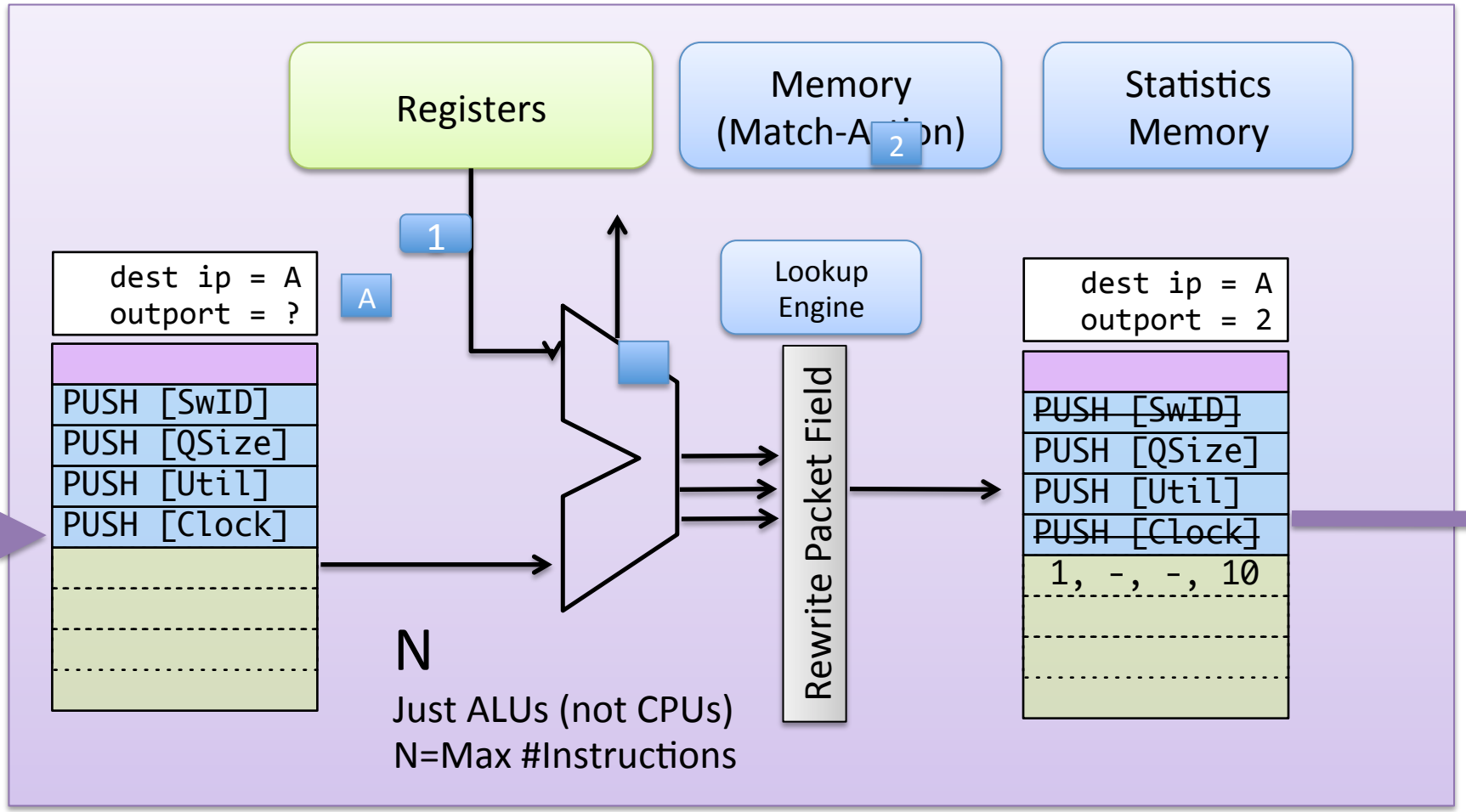
Switches/ASIC

Executes and forwards TPPs at line rate

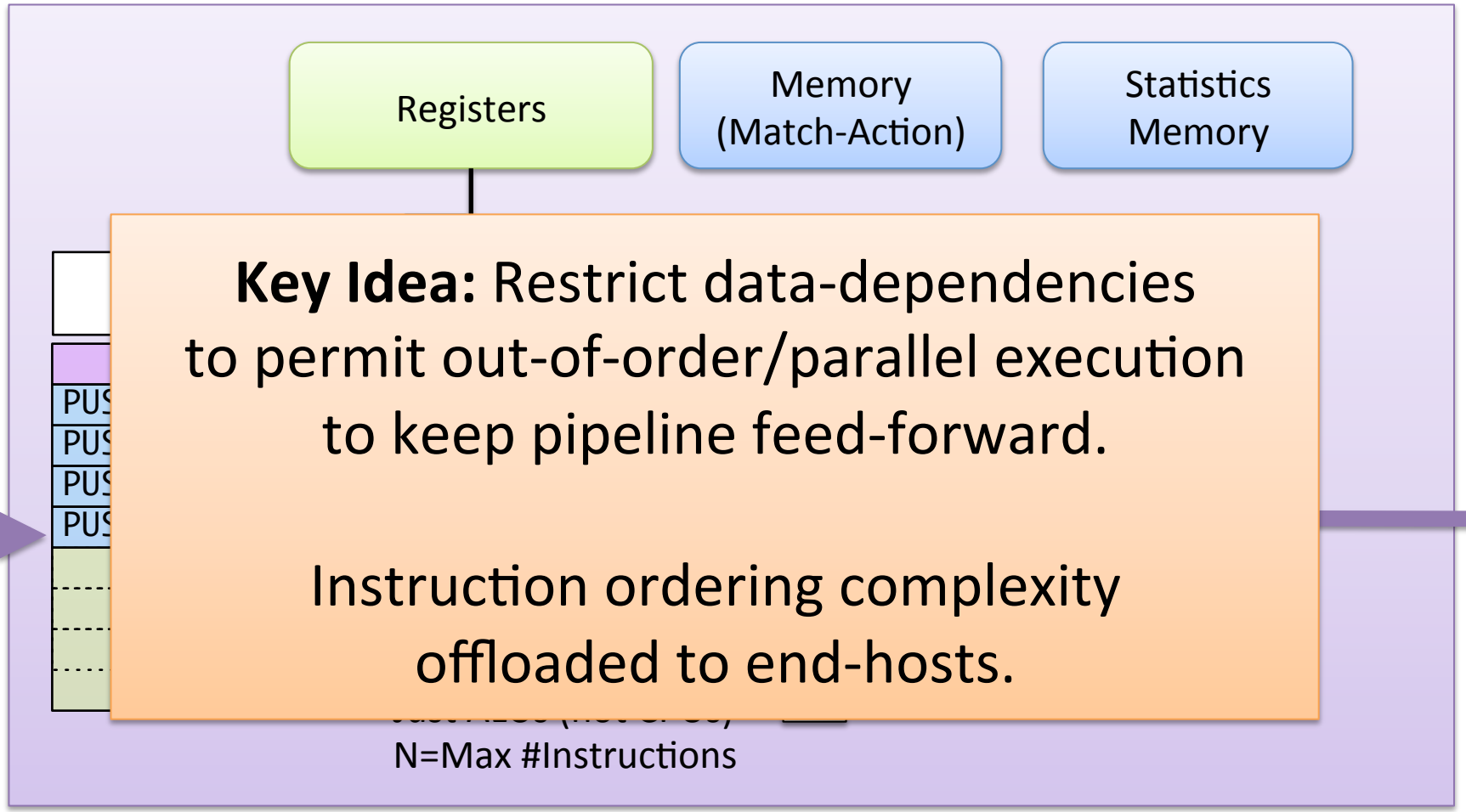
ASIC: How does TPP work?



Per-stage TPP processing



Per-stage TPP processing



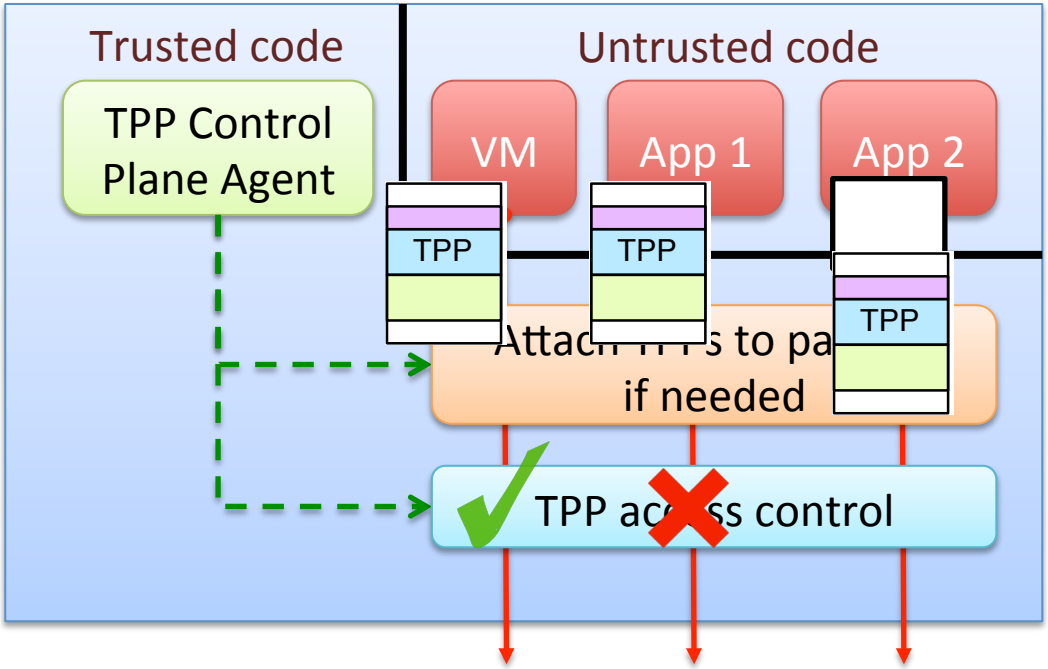
Feasibility Check: Hardware Costs

- NetFPGA-10G: 4 stages per port
  - 30% extra logic gates relative to simple IP router
  - ≤ 10 memory accesses per packet (by design)
    - Therefore, ≤ 10 cycle execution latency per packet

- Speculation from single-chip ASIC cost in [1]
  - < 3% additional instruction processors wrt to [1]
    - ≤ 50 nanoseconds @1GHz
    - Contrast this to unloaded 64B forwarding latency of 200—500ns

[1] Forwarding Metamorphosis (RMT) SIGCOMM 2013

TPP: End to End Picture



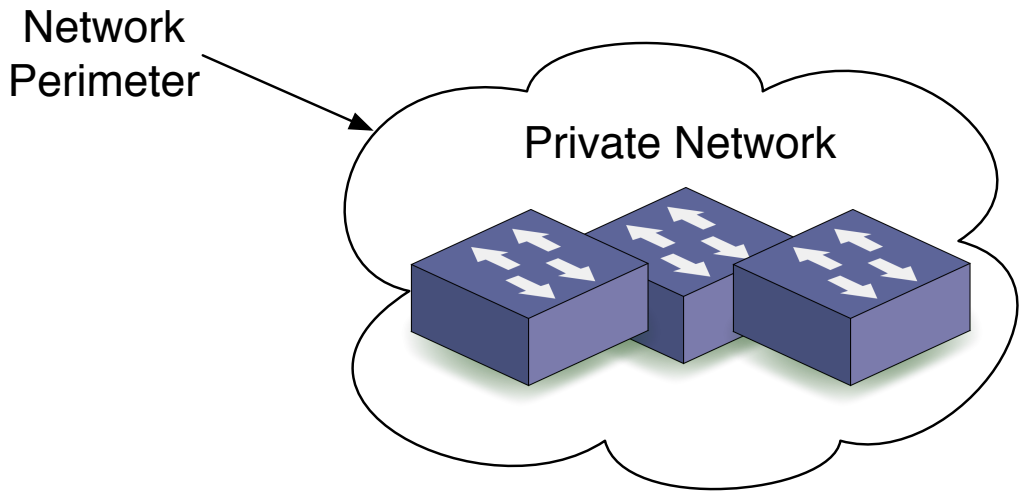
End Host Stack

- Create TPP apps
- Enforce fine-grained access control

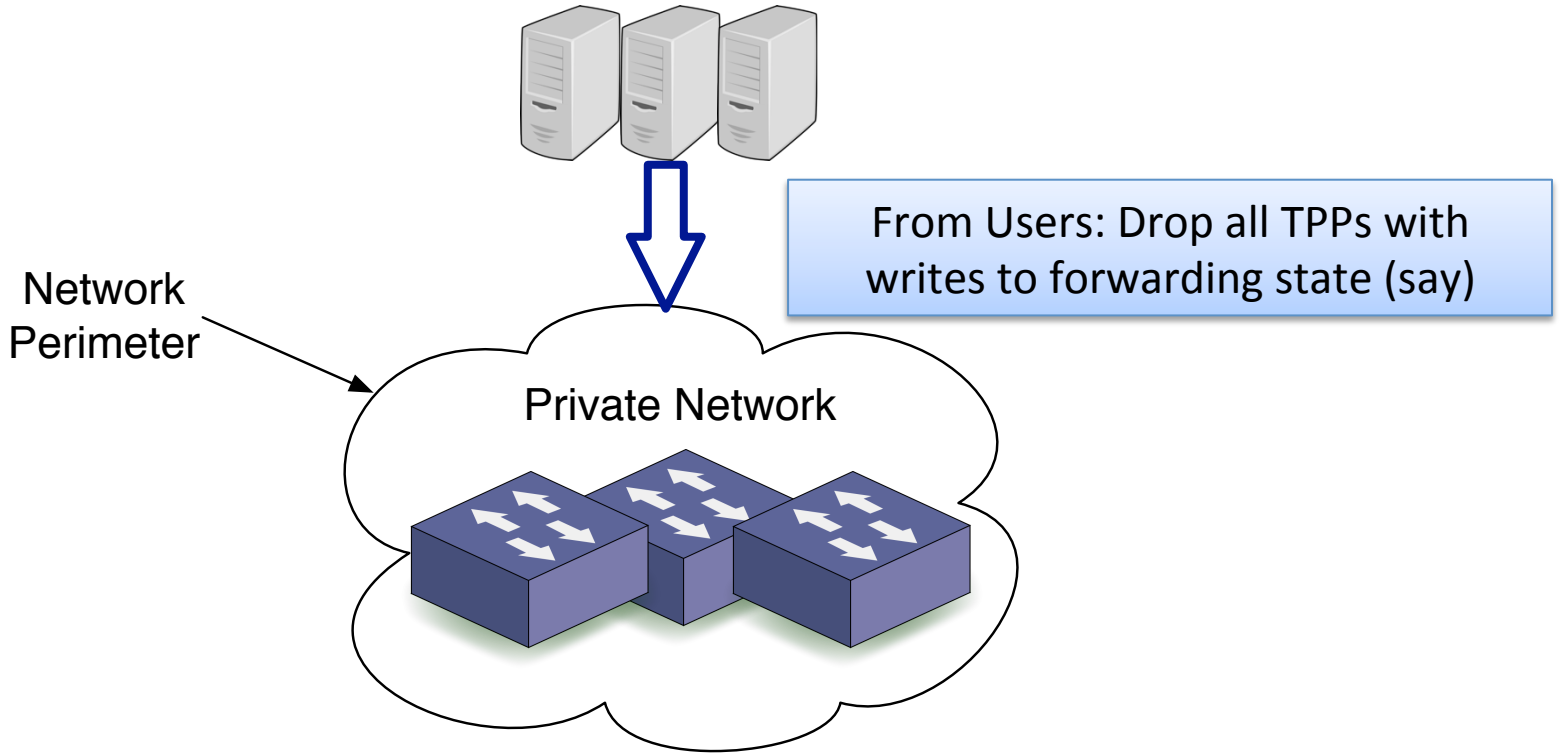
ASIC

Executes and forwards TPP at line rate

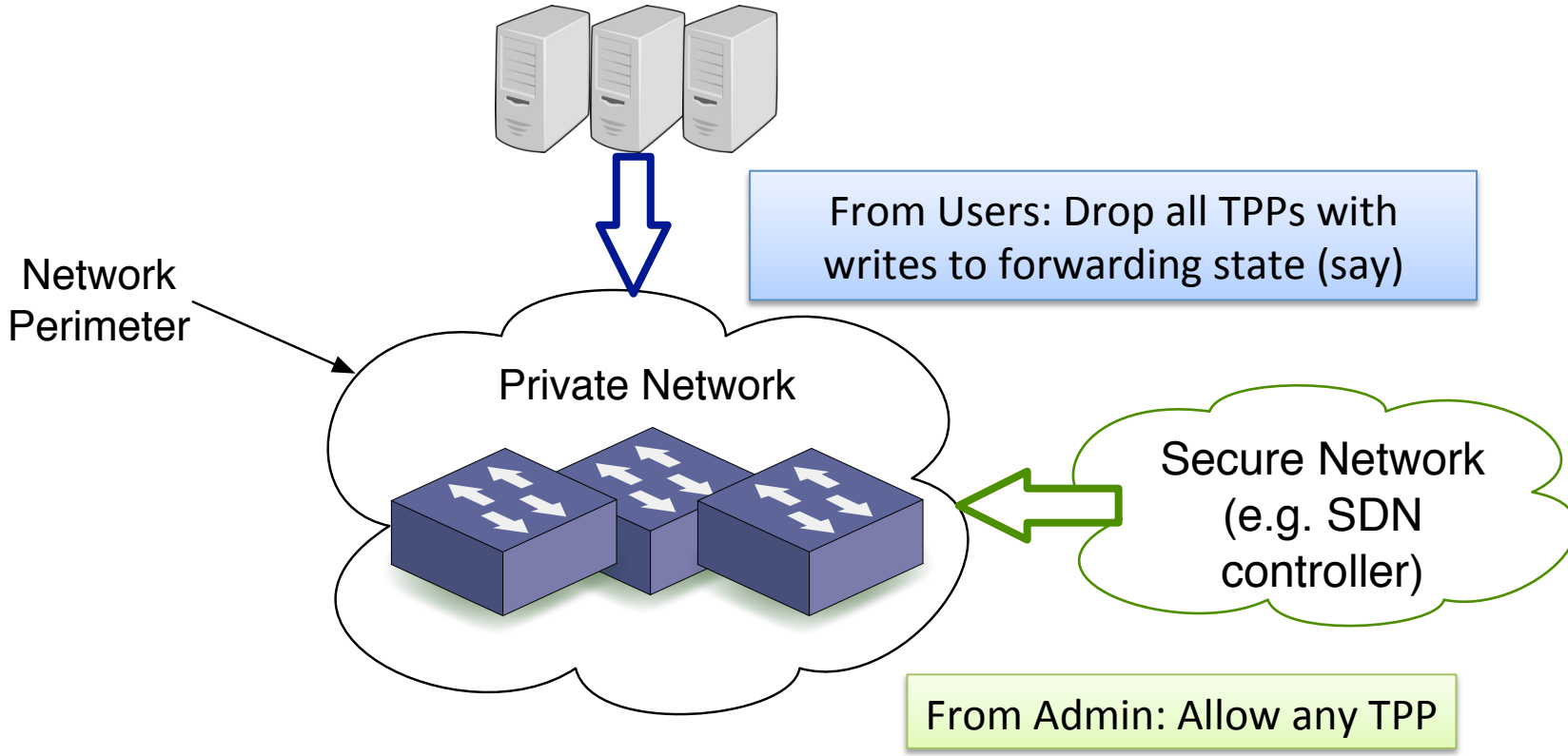
Sanity Check: TPPs can be made safe



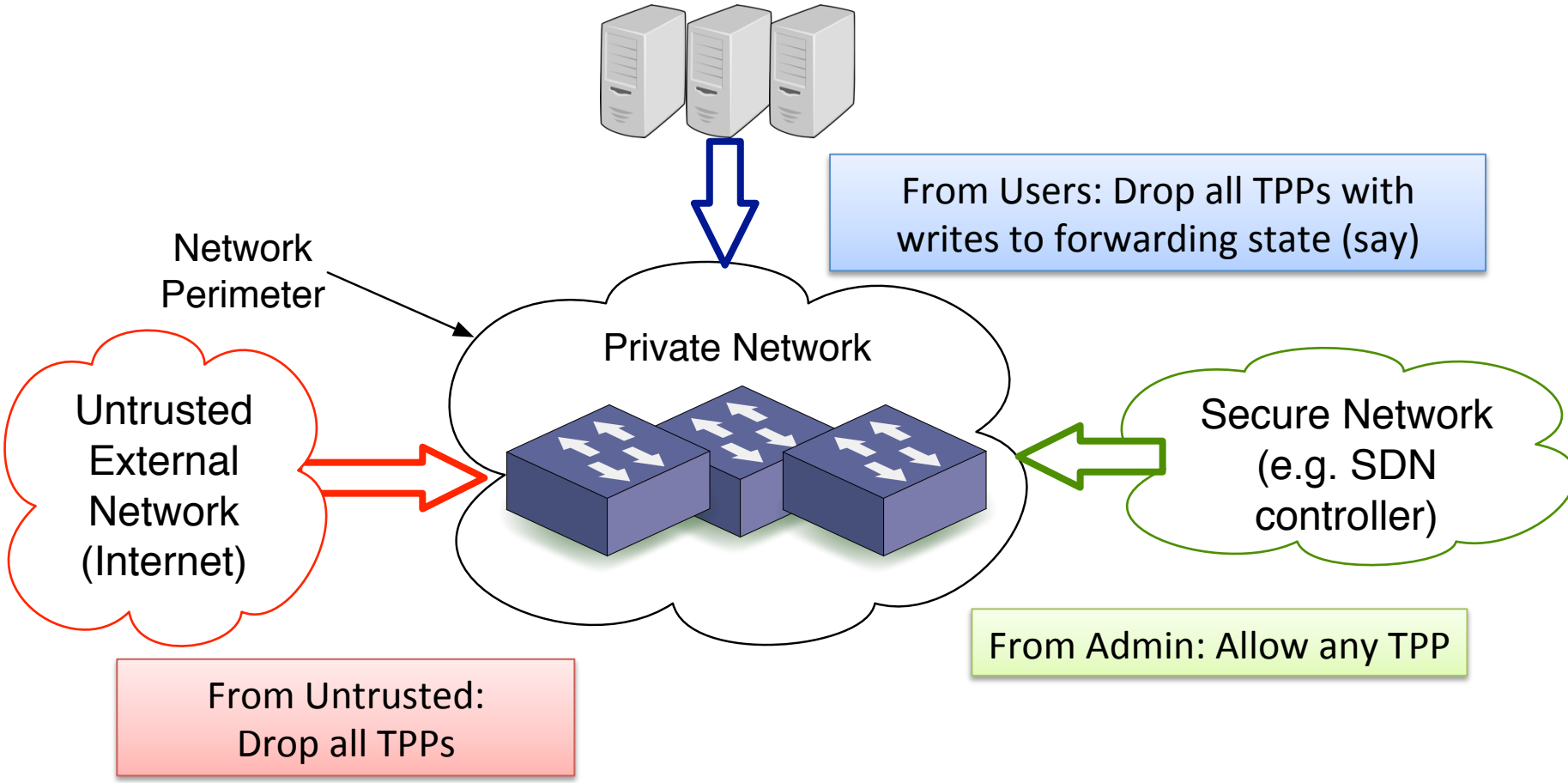
TPP Access Control Policy for Safety



TPP Access Control Policy for Safety



TPP Access Control Policy for Safety



Living on the Edge with TPPs

- What is TPP?
  - A data plane memory interface to switch memory
  - A highly simplified formulation of Active Networks
- Why is TPP useful?
  - Implement new data plane functions in software, without waiting years for specific hardware features

Pushing the limits of TPPs

What we have done
RCP: Maxmin and proportional fairness
Congestion localisation (micro-bursts)
Sketch-based measurement
Network debugging and troubleshooting

What we think is possible
XCP, QCP, QCN, MCP
Performance Isolation: EyeQ, Gatekeeper, FairCloud
Source Routing
Dynamic load-based packet routing (DeTail, etc.)

This year's best paper CONGA, on any TPP-capable network!

☺

(check our paper for spoilers.)

TPP isn't the right interface for:
Introducing new headers (e.g., IPv7, VXLAN, Geneve, etc.)
Event-driven functions (e.g., notify when link utilisation > 90%)
Per-packet queue management (e.g., pFabric, deficit round robin, etc.)