Tiny Packet Programs
for low-latency network control

Vimal

Mohammad Alizadeh, Yilong Geng, Changhoon Kim, David Mazières
Timescales of Network Functions

- Forwarding
  - AQM
  - DPI
- Congestion Control
  - Fine-grained Monitoring
  - Forwarding Debugging
- Network Load Balancing
- Virtual Network Provisioning
  - Routing
  - Traffic Engineering

Packet (ns to µs)  RTT (µs to ms)  Seconds  Minutes

Data Plane  Control Plane
Timescales of Innovation/Deployment

- Forwarding
  - Network Load Balancing
  - Congestion Control
  - Fine-grained Monitoring
  - Forwarding Debugging
  - AQM
  - DPI

- Virtual Network Provisioning
- Routing
- Traffic Engineering

Data Plane (Hardware)

Control Plane (Software)

Years!

Hours/Days/Weeks
From Idea to Production

August 19, 2014
Innovation: Where and How?

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.

MTU
1500B to 9000B

Payload
(e.g., TCP header)

Packet memory
(Initialized by end-hosts)

Instructions

Ethernet Header

(Not drawn to scale)
Tiny Packet Program

Control Fields (length, checksum, etc.)

Reads + Writes (optional)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD/PUSH</td>
<td>Copy state from switch to packet</td>
</tr>
<tr>
<td>STORE/POP</td>
<td>Copy state from packet to switch</td>
</tr>
<tr>
<td>CSTORE</td>
<td>Conditional store</td>
</tr>
<tr>
<td>CEXEC</td>
<td>Conditional execute</td>
</tr>
</tbody>
</table>

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

August 19, 2014
# Useful State inside Switches

<table>
<thead>
<tr>
<th>Namespace</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per-Switch</td>
<td><strong>Switch ID</strong>, clock/uptime</td>
</tr>
<tr>
<td>Per-Port</td>
<td><strong>Link utilisation</strong>, bytes sent/received, bytes enqueued, bytes dropped</td>
</tr>
<tr>
<td>Per-Queue</td>
<td>Bytes sent/received, <strong>bytes enqueued</strong>, bytes dropped</td>
</tr>
<tr>
<td>Per-Packet</td>
<td><strong>Input/output port</strong>/queue, matched flow entry indices, packet fields</td>
</tr>
</tbody>
</table>

For more statistics, please check OpenFlow specification.
Packet memory contains space to load/store state.

Tiny Packet Program

- Ethernet Header
- Instructions
- Packet memory (Initialized by end-hosts)
- Payload (e.g., TCP header)
Localising Micro-Bursts*: Where and Which app?

Timescales: 100s of us to milliseconds

*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

<table>
<thead>
<tr>
<th>Ethernet Header</th>
<th>SP = 0x0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUSH [SwID]</td>
<td></td>
</tr>
<tr>
<td>PUSH [QSize]</td>
<td></td>
</tr>
<tr>
<td>PUSH [Util]</td>
<td></td>
</tr>
<tr>
<td>PUSH [Clock]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ethernet Header</th>
<th>SP = 0x8</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUSH [SwID]</td>
<td></td>
</tr>
<tr>
<td>PUSH [QSize]</td>
<td></td>
</tr>
<tr>
<td>PUSH [Util]</td>
<td></td>
</tr>
<tr>
<td>PUSH [Clock]</td>
<td>1,0x10,1,4</td>
</tr>
</tbody>
</table>

August 19, 2014
Investigating Micro-Bursts

<table>
<thead>
<tr>
<th>Ethernet Header</th>
<th>SP = 0x0</th>
<th>SP = 0x8</th>
<th>SP = 0x10</th>
<th>SP = 0x18</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUSH [SwID]</td>
<td>PUSH [SwID]</td>
<td>PUSH [SwID]</td>
<td>PUSH [SwID]</td>
<td></td>
</tr>
<tr>
<td>PUSH [QSize]</td>
<td>PUSH [QSize]</td>
<td>PUSH [QSize]</td>
<td>PUSH [QSize]</td>
<td></td>
</tr>
<tr>
<td>PUSH [Clock]</td>
<td>PUSH [Clock]</td>
<td>PUSH [Clock]</td>
<td>PUSH [Clock]</td>
<td></td>
</tr>
<tr>
<td>1,0x10,1,4</td>
<td>1,0x10,1,1</td>
<td>1,0x10,1,1</td>
<td>1,0x10,1,1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2,0x00,2,3</td>
<td>2,0x00,2,3</td>
<td>3,0xff,1,5</td>
<td></td>
</tr>
</tbody>
</table>
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

Identify competing applications
- Packets → Applications
- Understand traffic patterns

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

Debugging Per-Packet Forwarding Behaviour: NetSight

Use per-packet visibility for detailed diagnosis.

August 19, 2014
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

Registers

Memory (Match-Action)

Statistics Memory

dest ip = A
outport = ?

PUSH [SwID]
PUSH [QSize]
PUSH [Util]
PUSH [Clock]

N
Just ALUs (not CPUs)
N=Max #Instructions

PUSH [SwID]
PUSH [QSize]
PUSH [Util]
PUSH [Clock]
1, -, -, 10

Rewrite Packet Field

Lookup Engine

dest ip = A
outport = 2

August 19, 2014
Per-stage TPP processing

Registers

Memory (Match-Action)

Statistics Memory

Key Idea: Restrict data-dependencies to permit out-of-order/parallel execution to keep pipeline feed-forward.

Instruction ordering complexity offloaded to end-hosts.

PUSH [QSize]
PUSH [SwID]
PUSH [Util]
PUSH [Clock]

Rewrite Packet Field

N = Max #Instructions

August 19, 2014
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

From Users: Drop all TPPs with writes to forwarding state (say)
TPP Access Control Policy for Safety

From Users: Drop all TPPs with writes to forwarding state (say)

From Admin: Allow any TPP

Secure Network (e.g. SDN controller)
TPP Access Control Policy for Safety

From Untrusted: Drop all TPPs
From Admin: Allow any TPP
From Users: Drop all TPPs with writes to forwarding state (say)

Network Perimeter

Untrusted External Network (Internet)

Private Network

Secure Network (e.g. SDN controller)
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

<table>
<thead>
<tr>
<th>What we have done</th>
<th>What we think is possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCP: Maxmin and proportional fairness</td>
<td>XCP, QCP, QCN, MCP</td>
</tr>
<tr>
<td>Congestion localisation (micro-bursts)</td>
<td>Performance Isolation: EyeQ, Gatekeeper, FairCloud</td>
</tr>
<tr>
<td>Sketch-based measurement</td>
<td>Source Routing</td>
</tr>
<tr>
<td>Network debugging and troubleshooting</td>
<td>Dynamic load-based packet routing (DeTail, etc.)</td>
</tr>
</tbody>
</table>

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.)