Opportunities and Challenges in In-Network Computing

Tutorial
CUSO Winter School – January 2020
Alberto Lerner (UNIFR)
Joint work with Robert Soulé (Yale) and Noa Zilberman (Oxford)
Warning

• The tutorial is still work in progress!
What is a switch?

Network switch silicon  Left / Right

Source: Noa Zilberman
Simplified switch architecture

To achieve high throughput, packet switches are pipelined.

Source: Noa Zilberman
How fast is a network switch?
100Gb/sec
Network switches

Switch box
Arista 7280

Switch silicon
Arista 7508R

Scale: 25Tbps, 64×400GE

Source: Noa Zilberman
Network switch-systems

Scale: Petabit / second

Cisco CRS-X

Source: Noa Zilberman

Huawei 12800
Programmable Networks?

- Traditionally, hardware network devices – NICs, switches, etc – have been black-boxes.
- There’s a new generation of devices that are “open,” as in, they can be (re-)programmed.
- Networking protocols can now be encoded as software.
So, Software-Defined Networks, right?

• Close, but, no
• SDN allows controlling different manufacturer’s devices uniformly, abstracting away their implementations

• However,
  • It is a “choose from a fixed menu” approach when it comes to functionality
  • It does not cover all ranges of devices (e.g., NICs)
What are the headers in the packet?

What is the processing algorithm?

What should the output packet look like?

Simplified programmable packet processing

Source: p4.org
In-network computing

The execution of native host applications within the network using standard network devices

Source: Noa Zilberman
In-network computing

The execution of native host applications within the network using standard network devices

Source: Noa Zilberman
In-network Computing: benefits

\( \times 10 - \times 100 \) Latency
\( \times 10,000 \) Throughput
\( \times 1,000 \) Power efficiency

Performance

Tokusahi ... & Zilberman, “The case for in-network computing on demand”, Eurosys 2019
Opportunities

• In-Network Computing
  • Programmable networks can support application logic, to some extent

• Uses include
  • Performance Management
  • Application-aware routing*
  • Caching (semi-offloading)
  • Full Offload
“To Some Extent”

• NICs and switches are not (yet) general purpose programming platforms
  • Deterministic speed as a requirement (e.g., in switches)
  • Limited “instruction set”

• Varied characteristics across different devices
  • But there’s an emerging programming model that shields programmers from hardware details
Agenda

• Part I - Programming Model
  • PISA
  • P4 cameo
  • Higher-Level Alternatives

• Part II – Example PISA targets
  • Software emulation
  • Programmable hardware switches
  • Reconfigurable NICs

• Part III – Use cases
  • Performance management
  • Services Coordination
  • Caching
  • Offloading

• Part IV – Discussion
  • Pitfall and Fallacies
Part I - Programming Model

PISA
P4 cameo
High-Level Alternatives
Protocol-Independent Switch Architecture

• A packet forwarding device can have four programmable components
  • Parser
  • Match-action
  • Header/metadata bus
  • Deparser
Parser/Deparser
Match-Action Table

Source: Noa Zilberman
Match-Action Example

- Forward only to 10.0.1.1
- Table layout defined at compilation time
- Table content manipulated at runtime

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
<th>Action Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.1.1/32</td>
<td>ipv4_forward</td>
<td>dstAddr=00:00:00:00:01:01 port=1</td>
</tr>
<tr>
<td>10.0.1.2/32</td>
<td>drop</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>NoAction</td>
<td></td>
</tr>
</tbody>
</table>

Source: Noa Zilberman
Action Capabilities

• Simple Arithmetic
• CRC calculation
• Hardware-dependent extensions
Header/Metadata Bus
Different Renderings of the Model

• NICs
  • One TX “pipeline,” one RX one

• Switches data planes
  • Ingress and egress pipelines
  • Parallel instances

• Others
  • Including in software
Programming against PISA machines

• P4 is emerging as a de-facto standard
  • Match-action table as a first-class citizen
  • Large number of possible targets
  • Compiler accepts hardware capabilities description
• Hardware-specific SDK

```c
table ipv4_lpm {
  key = {
    hdr.ipv4.dstAddr: lpm;
  }
  actions = {
    ipv4_forward;
    drop;
    NoAction;
  }
  size = 1024;
  default_action = NoAction();
}
```
Higher-Level Compilers

• P4 may be too low-level a language for certain domains

• E.g.,
  • DSL (e.g. Telemetry)
  • C-subset
  • SQL
Part II- Example PISA Targets

Software emulation
Programmable hardware switches
Reconfigurable NICs
Behavioral Model v2

• Software implementation of a PISA switch
• "Network-in-a-box" with quick setup
• Much more powerful than hardware targets
• Not the fastest speeds
Under the hood

- Compiler
- Virtual Interfaces
- BMV2
- P4Runtime

Figure from https://github.com/p4lang/tutorials/blob/master/P4_tutorial.pdf
Tofino

• Hardware implementation of a PISA switch
  • Ingress and Egress pipelines
  • “RMT” stages
  • Traffic Manager
Switching Silicon

Source: wired.com
Programmable Data Plane

### Table

<table>
<thead>
<tr>
<th>field: dest MAC addr</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>06:05:04:03:02:01</td>
<td>forward(port1)</td>
</tr>
<tr>
<td>01:02:03:04:05:06</td>
<td>forward(port4)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Diagram

- **Packets**: parser \(\rightarrow\) ingress MAUs \(\rightarrow\) traffic manager \(\rightarrow\) egress MAUs \(\rightarrow\) deparser

- **Traffic Manager**:\(X\)
Match-Action Unit

- Input/Output
  - PHV
  - Metadata
- Match engine
  - Constant time lookup
  - Different types of matches (and thus memories)
- Action Engine
  - Simple transformations

Adapted from Bosshart et al. *Forward Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN* 
SIGCOMM’13
Switch Programming Workflow

Source: Noa Zilberman
NetFPGA

- Line rate
- Flexible
- Open-Platform

Source: Noa Zilberman

Reference Designs

- Network Interface Card
- Hardware Accelerated Linux Router
- IPv4 Reference Router
- Traffic Generator
- Openflow Switch
- More Projects
- Add Your Project
Reference Switch

• Five stages:
  • Input port
  • Input arbitration
  • Forwarding decision and packet modification
  • Output queuing
  • Output port

• Packet-based module interface
• Pluggable design

Source: Noa Zilberman
P4 Simple SUME Switch

Source: Noa Zilberman
Part III – Use Cases

Performance management
Services Coordination
Caching
Offloading
Performance Management

• “TCP Incast” and data skew can cause the network to drop packets
• In-Band Telemetry can query the instantaneous network state
• Traffic policy can communicate the application guidelines it wishes to use
Semantic Routing

• Networking has been "address centric"
  • Servers are supposed to know the address with which they wish to communicate
• Networking can also be "data centric"
  • The network choses a destination according to a packet’s contents
Caching

• The network holds part of the "state" a server (or servers) is maintaining
• Request for that content can be server without travelling all the way to the servers
Offloading

• The network is one of the servers
  • A particularly well placed one at that
Part IV – Discussion

Pitfalls and fallacies

Research agenda
Fallacies

• The number of applications that can use programmable networks is limited.

• Programmable-network devices are expensive to purchase and operate, despite the performance benefits
Pitfalls

• CPU-based middle-boxes replace in-network program- ming devices

• Network devices do not have capacity enough for net- working and application functionalities to co-exist.
Research Agenda

• Stateful computations are still problematic
• Instruction set is still very limited
  • Packet slicing
  • Ordered tables
  • Etc
• Space for more higher-level DSLs
Conclusion
Conclusion

• Programmable networks is off-the-shelf equipment (at least for switches)
  • Programming model
  • Compiler tool chain
  • Development community

• In-Network Computing is actively being explored (“X on Tofino”)

• This is just the first generation of the platform
Q&A

Thank you!