Impulse Talk NC4:
Modelling Programmable Device Behavior

Max Helm
October 13, 2021
Academic Salon on Time-Sensitive Networking and Deterministic Applications
Munich, Germany

Chair of Network Architectures and Services
Department of Informatics

Technical University of Munich
**Motivation**

**Programmable device workflow**

```
Generic Programmable Device + Program = Programmed Device
```

- **Software-Defined Networking**
- **OpenFlow or P4**
- **Device for specific use case**
  - e.g. latency requirement

**Dynamic modeling workflow**

```
Generic Device Model + Selection of Functionality Models = Model of Programmed Device
```

- Collection of Network Calculus models
- Service curves for logical functions
- Derive worst-case latency bounds
Motivation

Programmable device workflow

Generic Programmable Device + Program = Programmed Device

Software-Defined Networking

OpenFlow or P4

Device for specific usecase e.g. latency requirement

Dynamic modeling workflow

Generic Device Model + Selection of Functionality Models = Model of Programmed Device

Collection of Network Calculus models

Service curves for logical functions

Derive worst-case latency bounds
Background
Software-defined Networking (SDN)

- Separation of concern for networks
- Three distinct planes with specific tasks:
  - Management and configuration
  - High-level network algorithms
  - Packet forwarding tasks
- Two well-known implementations of the SDN concept
  - OpenFlow (on the control plane)
  - P4 (on the data plane)
Background
OpenFlow vs. P4

OpenFlow
- Introduces programmability to the control plane
- Used for the manipulation of existing protocols
- Allows comparatively high-level packet manipulation

P4
- Introduces programmability to the data plane
- Creation of entirely new protocols
- Allows low-level packet manipulation

Shared design between P4 & Openflow
- Packet processing pipeline applies the match-action principle:
  - User define patterns (matches) to execute packet processing tasks (actions)

Challenges
- Device performance changes significantly depending on the programmed network task
- Conceptual differences between both languages hinder their direct comparison
Background
Performance Bounds in Networks

Network Calculus

- Calculate worst-case delay bounds in networks
- Represents nodes and data flows as wide-sense increasing functions
- Combines these functions to calculate bounds

Service Curve

- Wide-sense increasing function describing a node, depends on arrival and departure times of flow data
- Multiple nodes can be combined into one node by convolving their service curves

Convolution

\[
\beta_{f_0} \ast \beta_{f_1} \ast \beta_{f_0,f_1} = \beta_{f_0} \ast \beta_{f_1} = \beta_{f_0} \ast \beta_{f_1}
\]

Time [s]

Data [bit]

\[
R_{f_0} = T_{f_0} \quad R_{f_1} = T_{f_1}
\]

\[
R_{f_0,f_1} = \min(R_{f_0}, R_{f_1})
\]

\[
T_{f_0,f_1} = T_{f_0} + T_{f_1}
\]
Modeling Concept
Modeling Concept

Device Model

- Logical functions $f_n$ in the Device under Test (DuT)
- Baseline function $f_0$ needed to operate device
- Feed-forward network of additional functions
Modeling Concept

Device Model

- Logical functions $f_n$ in the Device under Test (DuT)
- Baseline function $f_0$ needed to operate device
- Feed-forward network of additional functions

Measurements

- Goal: measure each logical function in isolation
- Measure baseline function $f_0$
- Measure each logical function pair $f_0 + f_i$
Modeling Concept

Device Model
- Logical functions $f_n$ in the Device under Test (DuT)
- Baseline function $f_0$ needed to operate device
- Feed-forward network of additional functions

Measurements
- Goal: measure each logical function in isolation
- Measure baseline function $f_0$
- Measure each logical function pair $f_0 + f_i$

Service Curve Model
- Approximate service curve parameters for each logical function using measurements of function pairs
- Subtract influence of baseline function
- Latency parameter for service curve of $f_1$: $T_{f_1} = T_{f_0 + f_1} - T_{f_0}$
Modeling Concept

Device Model
- Logical functions $f_n$ in the Device under Test (DuT)
- Baseline function $f_0$ needed to operate device
- Feed-forward network of additional functions

Measurements
- Goal: measure each logical function in isolation
- Measure baseline function $f_0$
- Measure each logical function pair $f_0 + f_i$

Service Curve Model
- Approximate service curve parameters for each logical function using measurements of function pairs
- Latency parameter for service curve of $f_1$: $T_{f1} = T_{f0} + f_1 - T_{f0}$

Model any combination of logical functions while minimizing required measurements
Evaluation
Evaluation

Investigated Platforms

- 4 × 100 Mbit/s Ethernet ports
- low-cost, embedded hardware
- supports OpenFlow (realized as software)

- 4 × 10 Gbit/s Ethernet ports
- powerful hardware
- supports P4 programming language
Evaluation

Setup

OpenFlow / Zodiac FX
- OpenFlow controller required for switch management
- External timestamp monitoring network traffic via splitter

P4 / NetFPGA
- Standalone P4 implementation using pre-filled tables
- External timestamp monitoring network traffic via fiber-optical splitter
Evaluation

Differences between Platforms

Why did we choose the different platforms?

- Demonstrate the applicability of our framework, despite obvious differences:
  - OpenFlow (control plane programmability) vs. P4 (data plane programmability)
  - 100 Mbit/s vs. 10 000 Mbit/s
  - Embedded platform (Zodiac FX) vs. high-performance platform (NetFPGA)

Goal:

- Apply NC to programmable network devices
- Find a common framework applicable to vastly different platforms
- Therefore, we create and measure common test scenarios for both platforms
Evaluation

Investigated Test Scenarios

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>num. rules</td>
<td>1</td>
</tr>
<tr>
<td>packet size</td>
<td>64 B</td>
</tr>
<tr>
<td>match types</td>
<td>port, tp-dst, dl-dst, masked-nw-dst, five-tuple, all</td>
</tr>
<tr>
<td>action types</td>
<td>output, set-dl-src, strip-vlan, set-vlan-id, set-nw-src, set-nw-tos, set-tp-src</td>
</tr>
</tbody>
</table>

Table 1: Investigated match-action scenarios

- We use the match-action principle of P4 and OpenFlow as a common foundation for our comparison
- We investigate different match types and action types separately
- We start with the most basic forwarding scenarios (port & output) and gradually increase the complexity of the forwarder selecting the given match and action types
Evaluation

Comparison of Match Performance

- Variable match, fixed action
- Latency measurements and their comparison to the baseline function

OpenFlow / Zodiac FX

- Latencies scale with amount of data to be matched
- Maximum deviation from baseline is \( \approx 6 \mu s \)
Evaluation

Comparison of Match Performance

- Variable match, fixed action
- Latency measurements and their comparison to the baseline function

OpenFlow / Zodiac FX
- Latencies scale with amount of data to be matched
- Maximum deviation from baseline is \( \approx 6 \mu s \)

P4 / NetFPGA
- Maximum deviation from baseline is \( \approx 0.01 \mu s \)
- Time resolution of hardware is 0.0125 \( \mu s \)
Evaluation

Comparison of Action Performance

- Variable action, fixed match

![Latency Comparison Graph]

OpenFlow / Zodiac FX

- Deviations of 2 µs to 5 µs for lower layer manipulations (MAC, VLAN)
- Deviations of ≈ 9 µs for network and transport layer manipulations
Evaluation

Comparison of Action Performance

- Variable action, fixed match

OpenFlow / Zodiac FX
- Deviations of 2 µs to 5 µs for lower layer manipulations (MAC, VLAN)
- Deviations of ≈ 9 µs for network and transport layer manipulations

P4 / NetFPGA
- Maximum deviations ≈ 0.01 µs for any action
Evaluation

Evaluating the Predictive Power of our Model

- Use measurements to derive model of other logical function combinations for both devices
- Calculate latencies for the combinations
- Perform measurements for the new combinations
- Compare them to the model results and calculate the relative error
Evaluation

Evaluating the Predictive Power of our Model

- Use measurements to derive model of other logical function combinations for both devices
- Calculate latencies for the combinations
- Perform measurements for the new combinations
- Compare them to the model results and calculate the relative error
Evaluation

Predictive Quality Evaluation (Worst Case)

OpenFlow / Zodiac FX
- Relative error below 1%
- Relatively high variance between function combinations

P4 / NetFPGA
- Relative error below 0.75%
- Comparatively low variance
Model exhibits a reasonable predictive power.

No high correlation between error and types of function in combinations indicates good overall performance.
Conclusion

Summary & Contributions

Summary

- Dynamic performance model for programmable devices, requiring less measurements than resulting models
- Measurements demonstrate (expected) performance gaps between platforms
- We applied the same methodology to entirely different classes of programmable network devices
- Dynamic models show low error for both platforms respectively

ITC 33 paper:

- Details on measurement & modeling methodology as well as gathered data

Future Work

- Exact service curve derivation based on inversion of the min-plus convolution
- More complex service curve shapes
Backup Slides
OpenFlow / Zodiac FX
- Similar behavior

P4 / NetFPGA
- Similar behavior
Backup Slides

Predictive Quality Evaluation (Median Case)

OpenFlow / Zodiac FX
- Similar behavior

P4 / NetFPGA
- More variance between different function combinations