P4Debug:
A Framework for Debugging Programmable Data Planes

Pietro Giuseppe Bressana

Research Advisors:
Prof. Robert Soulé
Dr. Noa Zilberman
Pervasive Networks
Faulty Networks
Debugging Networks
Programmable Networks

New Switches:

• Programmable
• Reconfigurable
• Match fixed-function speed

Our goal:

leverage **Programmability** for **Debugging** network hardware
Outline:

✦ Debugging Networks
✦ A Use-Case
✦ Project Status
✦ Conclusions & Future Work
Programmed the switch with a custom P4 specification,

Sending traffic to switch, but don't see it getting to the end host…

How could we Find Out what went wrong Inside the switch?
Existing Projects

In-band Network Telemetry (INT)

Postcards
Limitations

• **Reactive**, packets *carry / trigger* debug info:
  ‣ No debug info going out of the pipeline!
  ‣ Cannot generate specific debug packets!

• **Embedded**, functionality *inside* the pipeline:
  ‣ Faulty pipeline => faulty debug!
  ‣ Cannot inspect each pipeline stage!
  ‣ In case surrounding HW is faulty…
  ‣ Cannot isolate the pipeline!
P4Debug

TEST PACKETS GENERATION ALGORITHM

+ 

INPUT

P4 PIPELINE

OUTPUT

INTERNAL PACKET GENERATOR MODULE

DEBUG MODULE
Fixed Infrastructure:

Functionality Changed by **Loading** New P4 **Program**

**No** Need to **Rebuild/Test** Hardware Every Time
P4Debug

Can generate (P4) **Specific** debug packets for **Exploratory** analysis

+  

Can run test generation **Algorithm**
Test packet generation

Algorithm:

Analyse user P4 specification
Test packet generation

Algorithm:
Identify header processing pipeline
Test packet generation

Algorithm:
Test data to trigger actions
Test packet generation

Debug Module:
Inject packets into P4 pipeline

Debug Module:
Collect debug events
P4Debug

Stand-alone:

Pipeline is Faulty?

Debug infrastructure Works Fine
P4Debug

**Stand-alone:**

Debug **Probes** + Debug module

Can **Inspect** each pipeline stage

TEST PACKETS GENERATION ALGORITHM
P4Debug

Pipeline is Isolated:

Surrounding HW is faulty?

Can debug the pipeline **From The Inside** of the device

TEST PACKETS GENERATION ALGORITHM

+
Speeding-up Lab Validation

**Hardware Testing:**

- Test
- Problem
- Add/Change HDL
  - Synthesis [Hrs]
- Add/Change Probes
  - Implementation [Hrs]

**P4Debug:**

- Test
- Problem
- Reprogram Switch Pipeline
  - P4 [mins]
- (Re)Program Debug Infrastructure
  - P4 [mins]
More Potential Use-Cases

- Basic Checks:
  - Switch port, checksum, packet size …

- More complex configurations:
  - NetFPGA SUME switch pipelines;
  - Custom network protocols.

- Real world applications:
  - P4 programs found in the literature.

- Performance:
  - Functional correctness, performance issues.
Language Extensions for Debugging

Specific Debug Packets

Debug Extensions

Advanced Features
Outline:

- Debugging Networks
- A Use-Case
- Project Status
- Conclusions & Future Work
Prototype: Technology

- P4
- P4-SDNet
- FPGA
  - P4-to-FPGA compiler
  - HW/SW FPGA Framework
Prototype: Architecture

TEST PACKETS GENERATION ALGORITHM

SW
Automatic generation tool

+ 

HW
FPGA-based Architecture
Prototype: Status

Host
  Trigger
  Collection

+ HW
  Packet generation
  Verification
Evaluation

- **Functionality:**
  - Number/type of bugs found;
  - Coverage of possible combinations.

- **Performance:**
  - Overhead of adding debug infrastructure.
  - Max packet rate that can be debugged;
  - Time for debugging all possible options;
  - Time Vs size of program (# of tables/rules/actions).

- **Resources:**
  - Impact of debug infrastructure;
  - Overhead when changing pipeline specification.
Outline:

✦ Debugging Networks
✦ A Use-Case
✦ Project Status
✦ Conclusions & Future Work
Conclusions

Networks are **Pervasive & Fault-prone:**

Tools & support **Not Enough** for Debugging

Reconfigurable switches: great **Programmability**

**P4Debug:** visibility into network devices
Future Work

Three **Steps**:
Packet Generator, Debug Module, P4 Extensions

Advanced **Techniques**:
PR, Header Space Analysis, …
Thank You for Your Attention !!!

pietrobressana.github.io

pietro.bressana@usi.ch