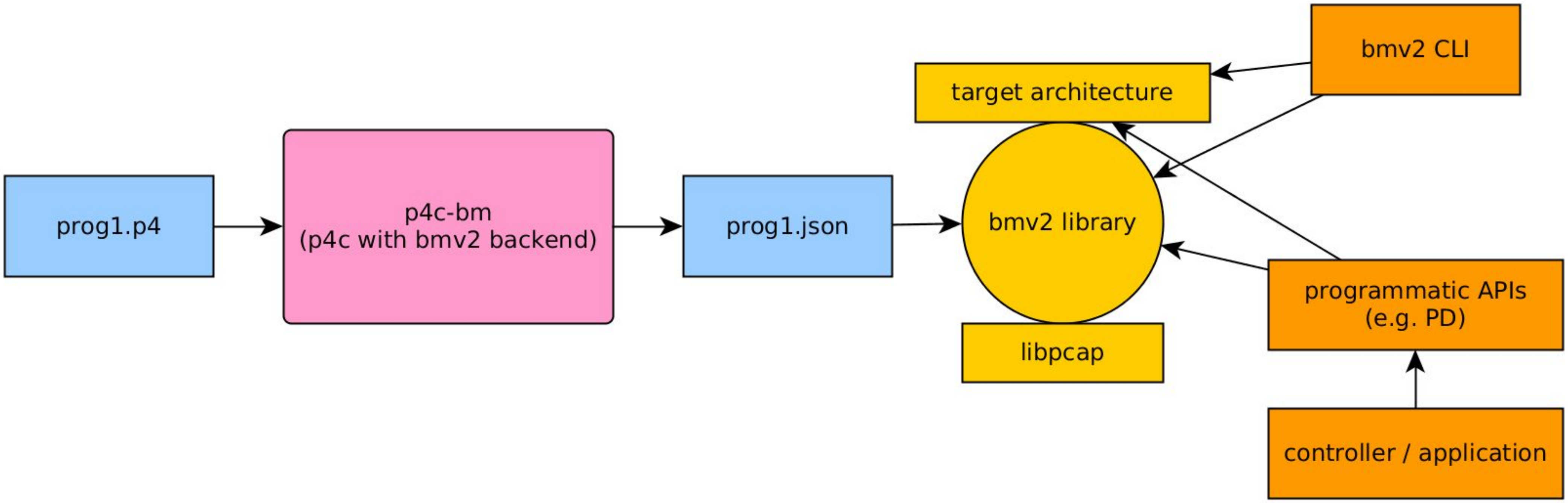
Enabling fast P4 development with bmv2

Antonin Bas, Barefoot Networks

What is bmv2?

- bmv2 = behavioral-model version 2
- A C++ user-space software switch to emulate your P4 dataplane
- Aimed at being 100% conformant to the P4 specification
- No C/C++ (re)compilation required when switching P4 programs
- Architecture-independent: mostly generic code which can be used to implement any device model (that can be described by P4 v1.2)
- A set of tools to make development easier: a runtime CLI to program & inspect the dataplane, a GDB-like debugger

bmv2 workflow



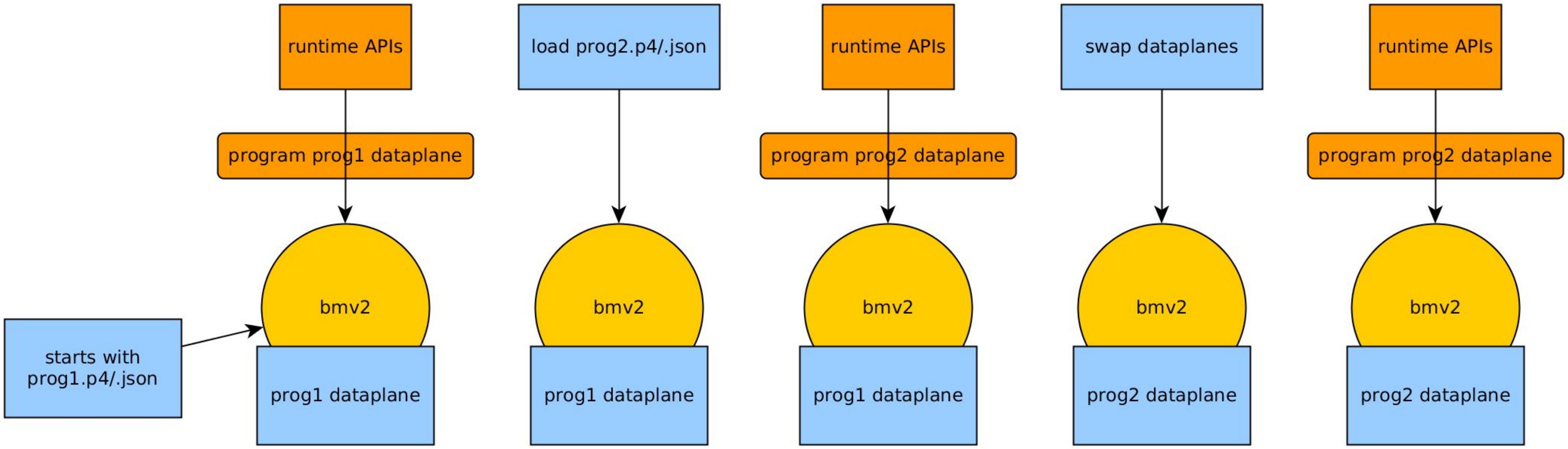
Target-independence: implementing your device

- bmv2 comes with a *simple_switch* target, which implements the P4 v1.0 / v1.1 abstract device model using the bmv2 library: ~600 LOC
- Vendors with a P4 programmable device can develop their own software switch / simulator using the bmv2 building blocks: parser, deparser, match-action pipeline…
- Each new target can define: its own primitive actions, its own hash functions, *and soon its own extern types*

The bmv2 runtime CLI

- Makes it easy to perform simple, individual table operations:
  - `table_add ipv4_lpm set_nhop 10.0.0.1/24 => 10.0.0.1 1`
  - `table_delete ipv4_lpm <handle returned by table_add>`
  - `table_dump ipv4_lpm`
- Also support for P4 stateful objects: counters, meters, registers
- Final goal is to have a standard P4 CLI, built on top of the standard P4 runtime APIs, which will work for all P4 targets

One cool feature: live swapping of P4 programs



- Can be done through the CLI: `load_new_config_file`, `swap_config_files`
- Packets see only an atomic swap

Introducing a P4 debugger

- bmv2 supports a GDB-like P4 debugger
- Lets you block packet processing and print information about packet headers / metadata / match-action tables
- Because of the bmv2 threading model, multiple packets can exist in the switch simultaneously; each packet is identified by a unique id
- We tried to give the debugger a “GDB feel” with familiar commands:
  - **`w`** sets a watchpoint on a header field
  - **`b`** sets a breakpoint on a P4 object (e.g. parse state, table)
  - **`p`** prints the value of a header field for a given packet
  - **`bt`** prints the backtrace of P4 objects entered by a packet
  - **`c`** starts / resumes packet processing

Introducing a P4 debugger

- Because multiple packets can exist in a switch at a given time, events can get intertwined
- Several commands to make things more readable and “follow” a given packet
  - `` `filter_notifications` `` to restrict the received notifications to a subset of packets
  - `` `stop_packet_in` `` to prevent the switch from accepting more packets
  - `` `skip` `` to skip all future notifications for a given packet
- Integrates with the bmv2 runtime CLI
- User guide available on p4lang
- Work in progress: better integration with P4 in the future (e.g. ability to break using line numbers)

Recap

- A P4 conformant software switch that can be used
  - to test your P4 programs
  - to test your P4 applications / control planes
  - as a base to implement a simulator for your P4-enabled hardware
- Input is a JSON file, which will remain the same as the P4 language evolves
- Many useful development features already (intuitive CLI, packet-level logging, debugger) and more to come!
- All the code is available on p4lang: https://github.com/p4lang/behavioral-model