P4 OFFLOAD

Matty Kadosh

11/9/2017

Mellanox TECHNOLOGIES

Connect. Accelerate. Outperform.™

FLEXIBLE ETHERNET

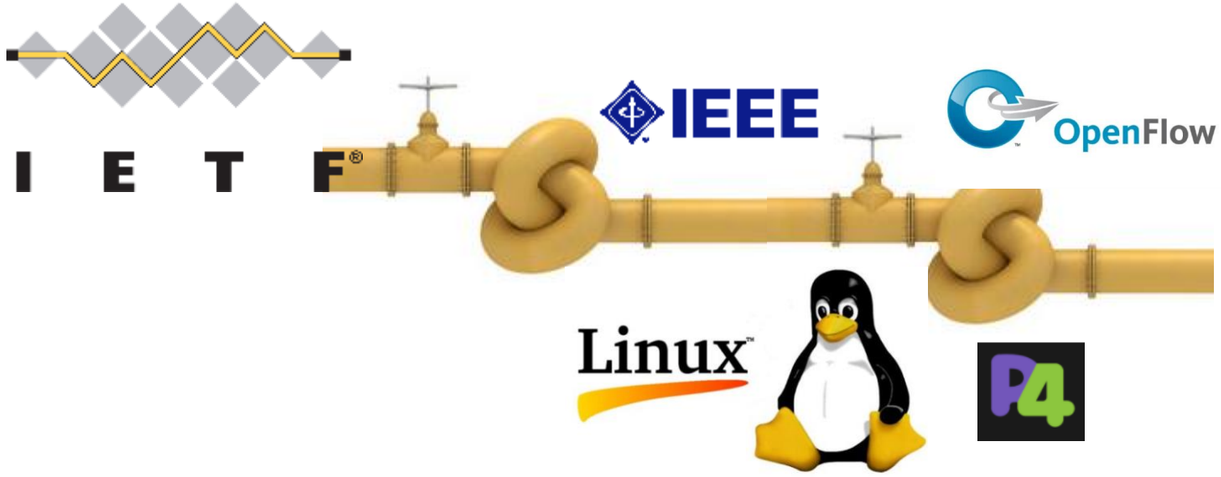
Old world

$$$ on legacy protocols
Best performance and stability
Low feature velocity

New world?

Write everything from scratch
Implement both standard and new applications
Variant feature velocity

Real world

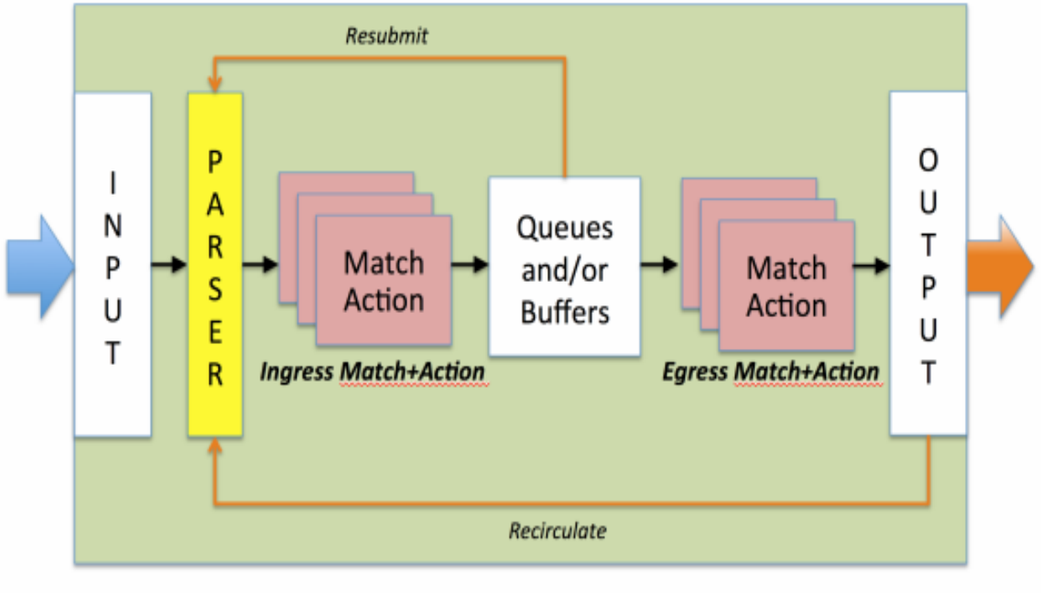


Legacy protocols don’t change
Application sand box for home grown needs
Extended HW longevity
High feature velocity

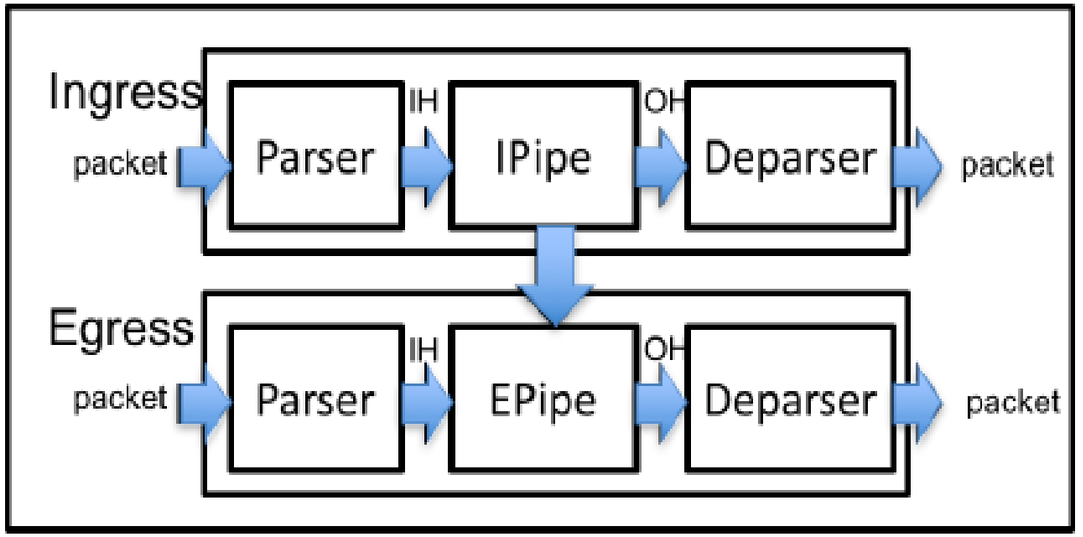
Mellanox TECHNOLOGIES SOFTWARE

P4

v14



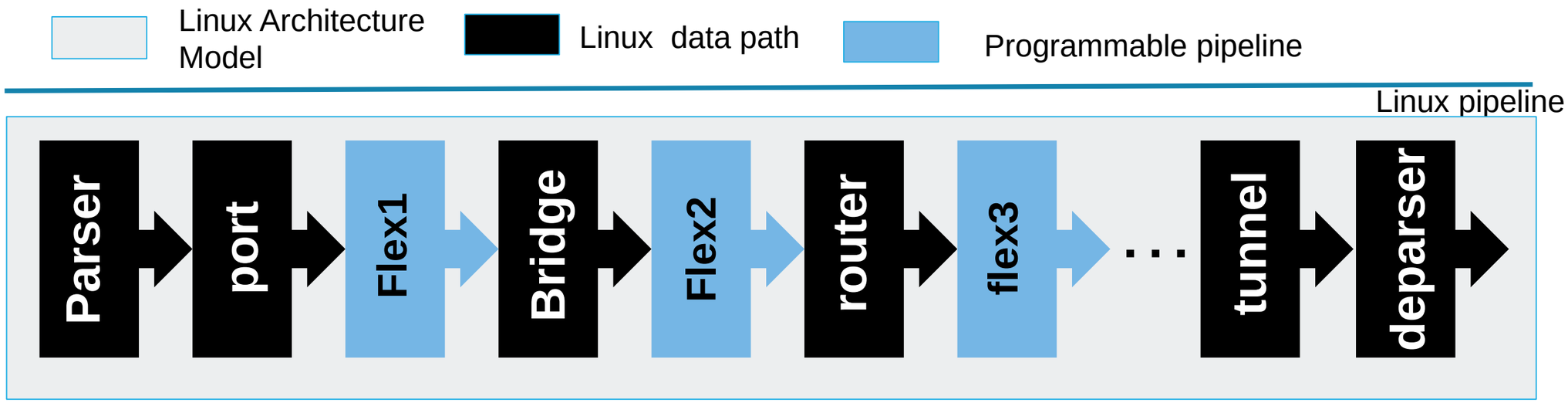
v16



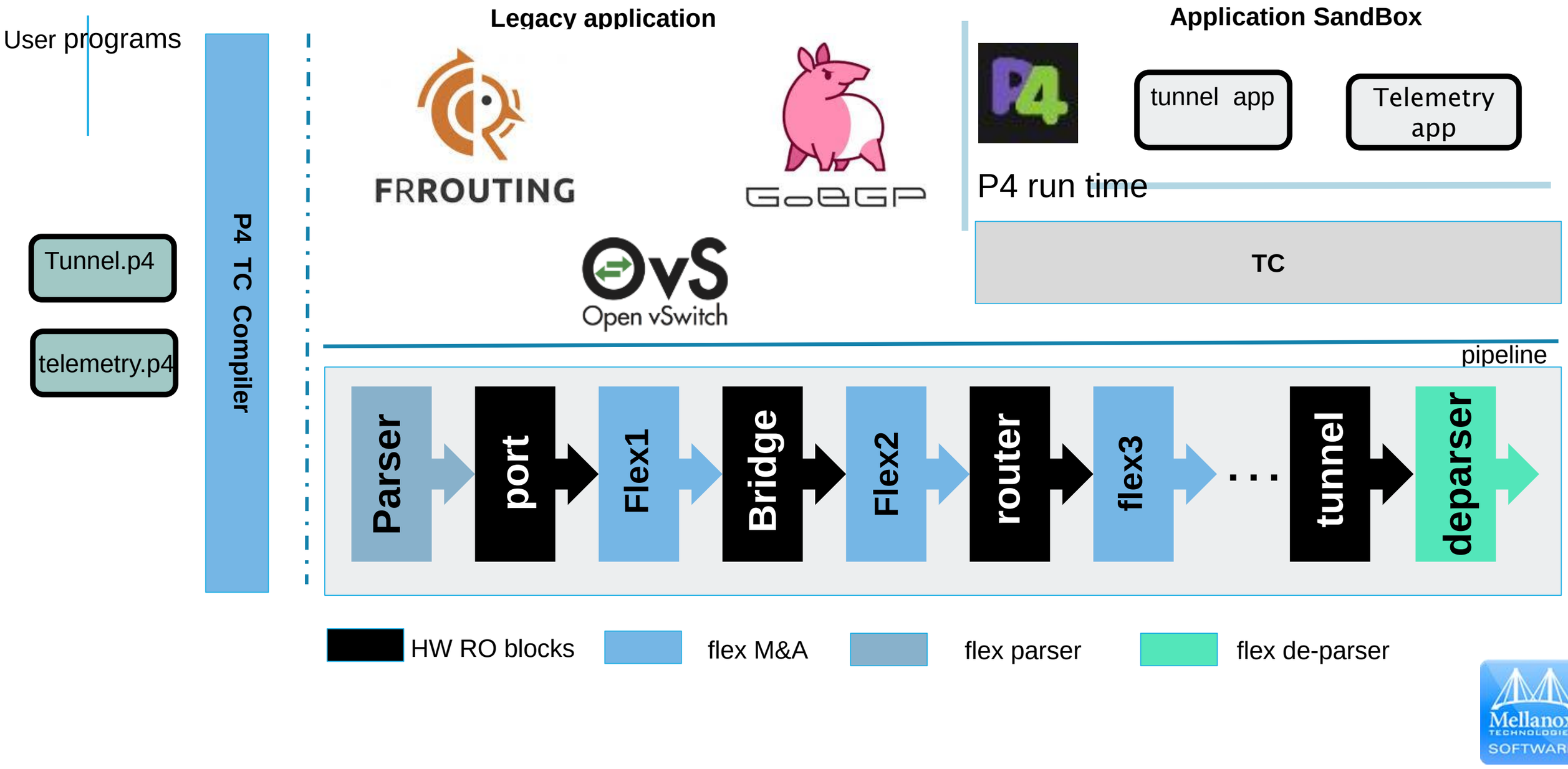
From the spec:

- Introducing P4 architecture description language
- “The P4 architecture can be thought of as a contract between the program and the target”
- “ Programmable blocks” i,.e. flexible blocks within a solid target
- “In general, P4 programs are not expected to be portable across different architectures”

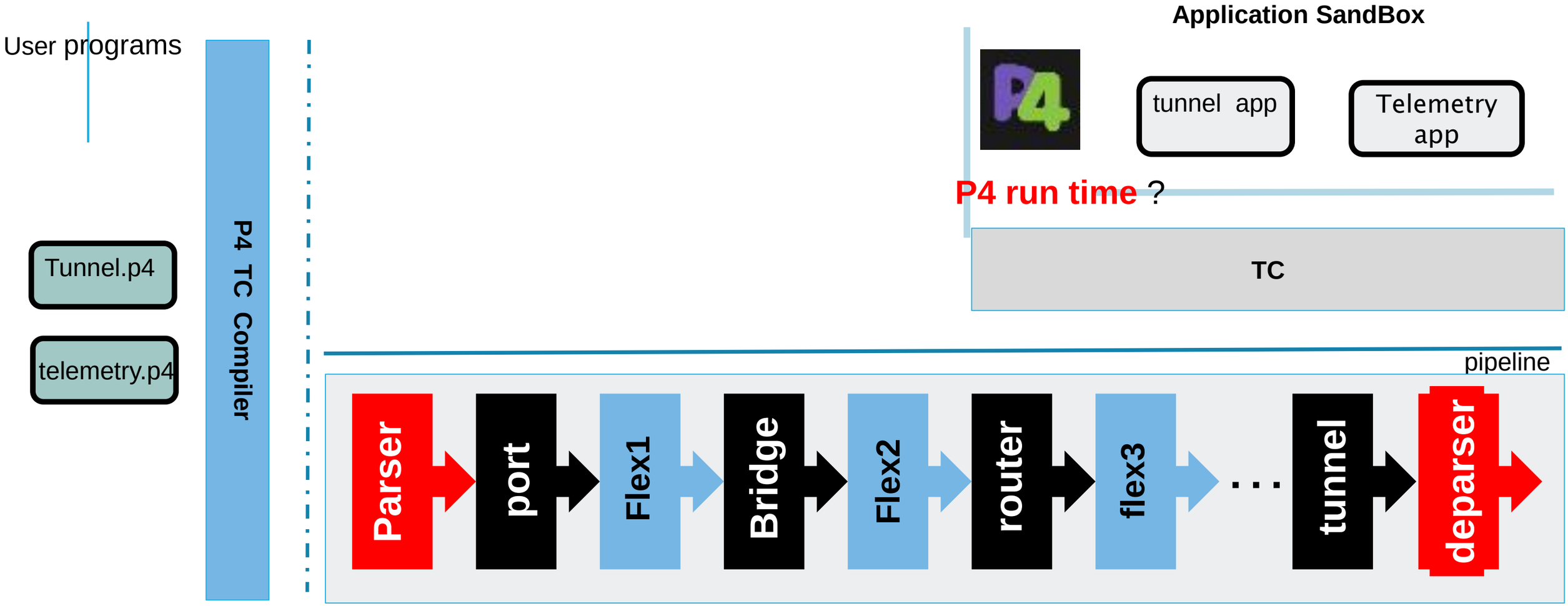
LINUX TARGET



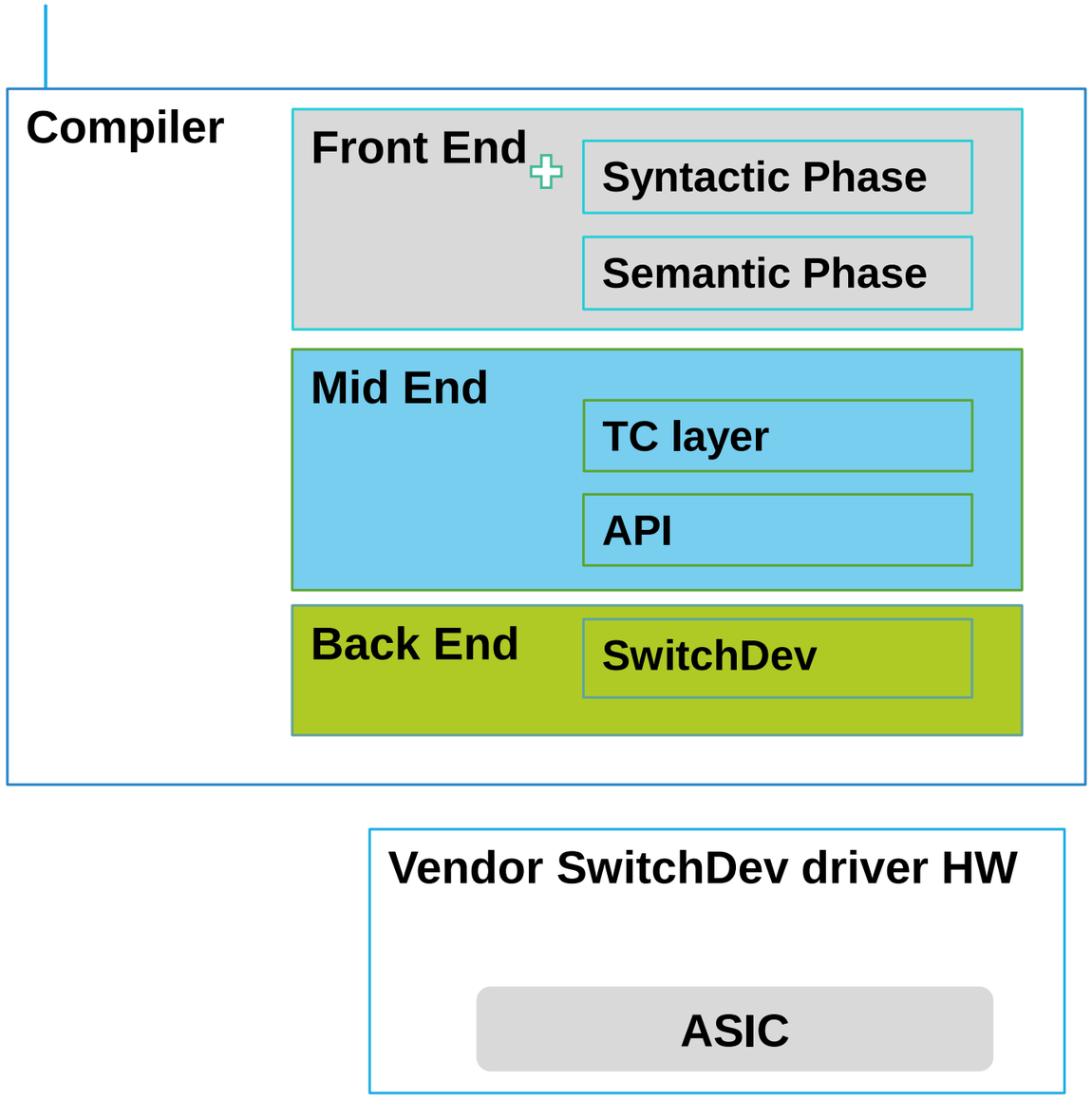
LINUX PROGRAMMABILITY



LINUX PROGRAMMABILITY – OPEN QUESTION



TC P4 COMPILER ARCHITECTURE



- **P4 version – $P4_{16}$,**
- **Front End** – Relates only to the language.
  - **Syntactic phase** – BNF based. Last time we read the source files. Output: Symbol tables.
  - **Semantic phase** – Verifies the Symbol tables. Extends the default semantic checks with platform specific ones.
- **Mid End**
  - **TC Pipe line configuration**
  - **P4 API**
- **Back End**
  - **Switchdev TC offload**

Mellanox TECHNOLOGIES SOFTWARE

Thank You

Mellanox TECHNOLOGIES

Connect. Accelerate. Outperform.™