

# API Design Challenges for Open Router Platforms on Proprietary Hardware

Jeff Mogul -- [Jeff.Mogul@hp.com](mailto:Jeff.Mogul@hp.com)

Praveen Yalagandula, Jean Tourrilhes, Rich McGeer, Sujata Banerjee, Tim Connors, Puneet Sharma

HP Labs, Palo Alto



Open router platforms need a programming interface that gives software access to platform hardware features at the right level of detail.

Open router platforms need a programming interface that gives software access to platform hardware features at the right level of detail:

- Why do we need open router platforms?
- What hardware features?
- What software uses them?
- What do we propose as the API?

# Why open router platforms?

Open Router Platforms allow third parties to develop software extensions for proprietary router hardware, supporting (e.g.):

- Faster deployment of novel features
- Consolidation of multiple vendors' functions into one physical box
- Reduced lock-in
- Creation of SW ecosystems
  - Router vendor gains features without NRE costs
- Improved programming discipline even within a vendor's engineering team

# What about software-only routers?

Software routers have their place:

- Cheap if you use recycled PC hardware
- Easy to program, extend
- Well-supported by Open Source

but:

- Cannot match top performance of HW routers
- Cannot match price/performance of commodity Ethernet switches
- Hard to get lots of ports in dense package
- Probably less energy-efficient

# Orphal

a.k.a. the “Open Router Proprietary-Hardware  
Abstraction Layer”

We explain:

- Why we want it
- What it would look like

but we haven't built it yet

# What hardware features are interesting?

Orphal can expose (for example):

- TCAMs (Ternary Content Addressable Memories)
- Hardware hash tables
- Programmable header extractors
- Deep Packet Inspection (DPI) ASICs
- Programmable packet-header rewriters
- Power monitoring and power-scaling controls

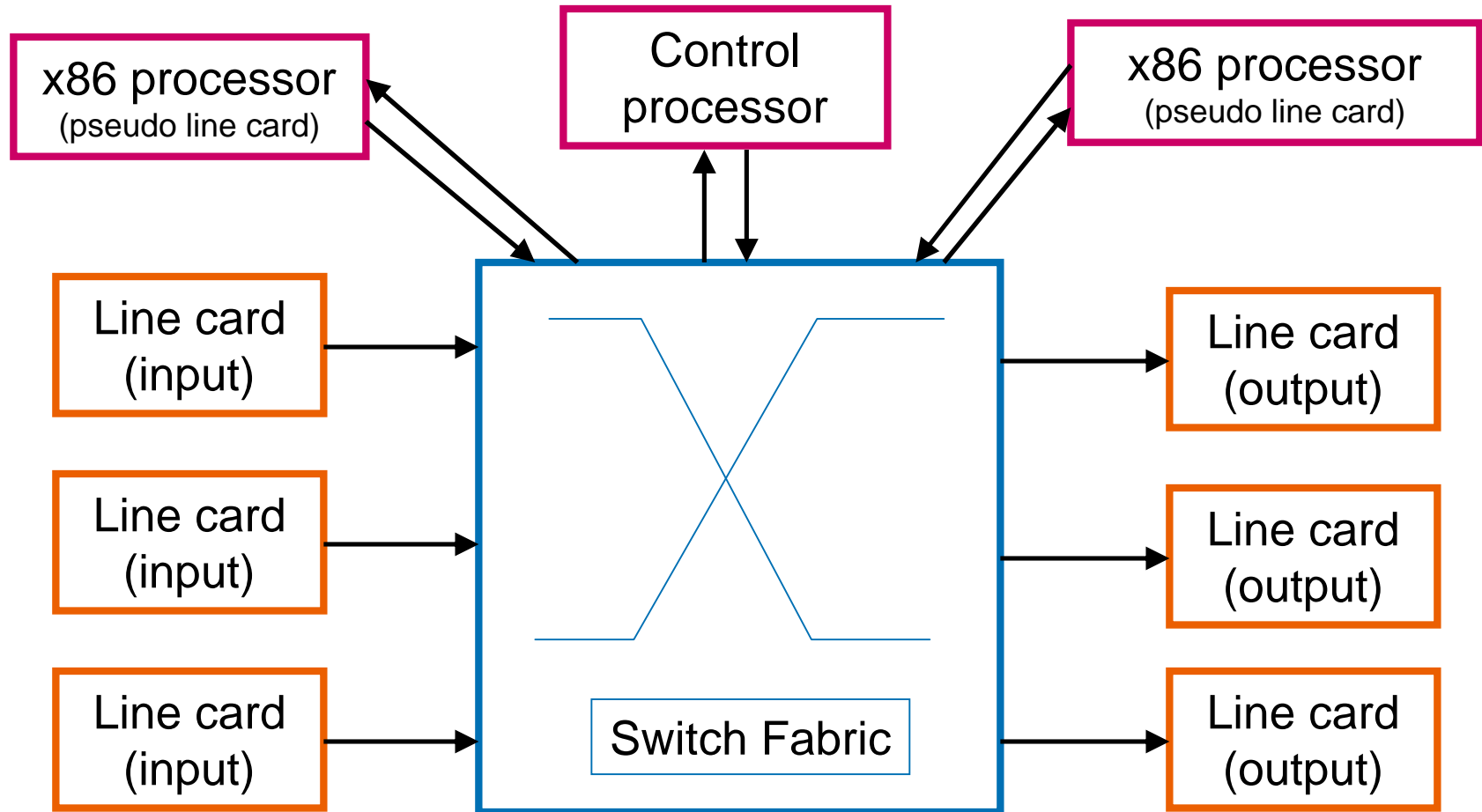
Most of these features are on the line cards

Base OS (e.g., Linux) exposes CPU, RAM, storage

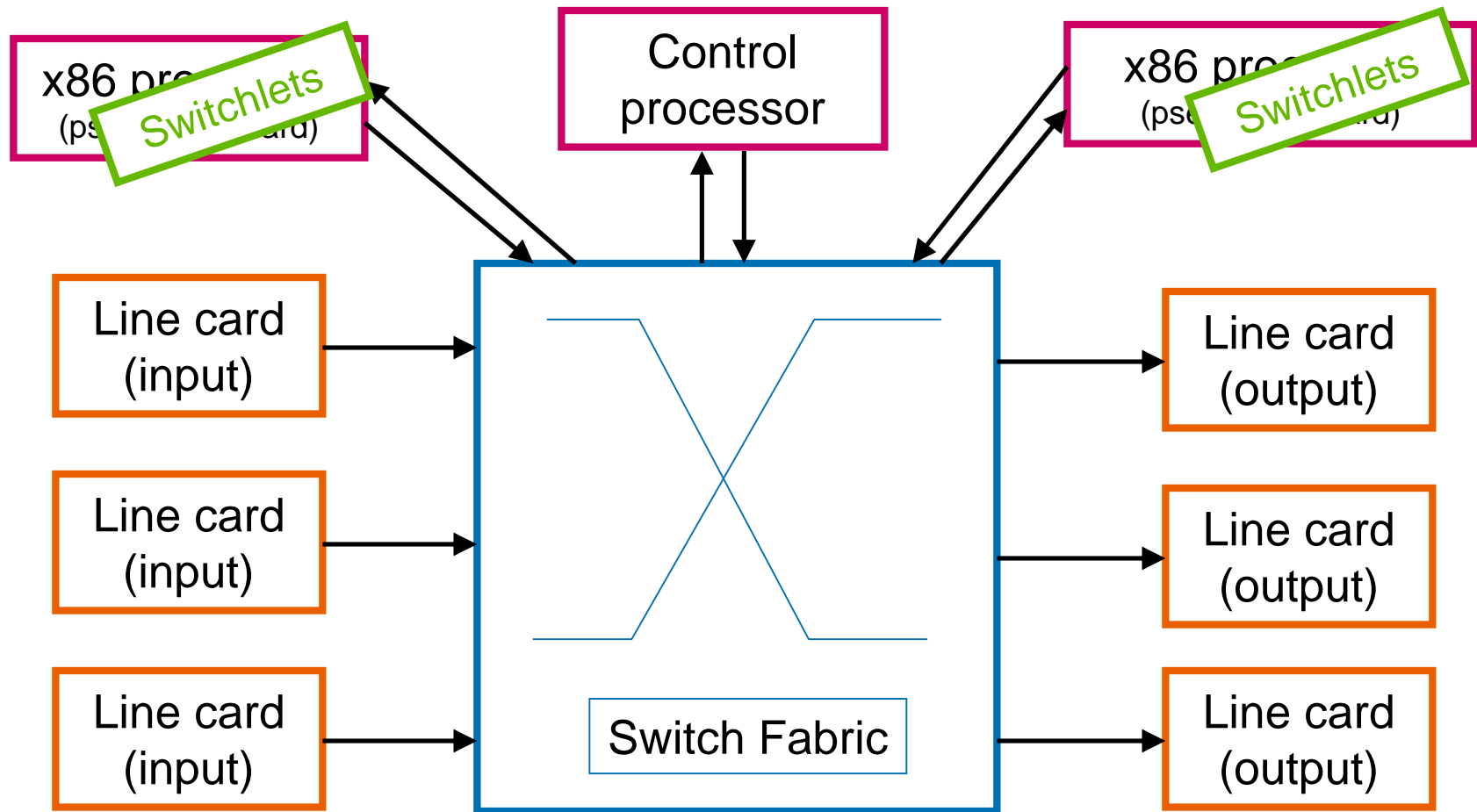
# What **software** runs on open routers?

1. Platform-specific software
2. General-purpose OS (e.g., Linux or Windows)
3. Vendor-specific software framework
4. **“Switchlets”**
  - **Modules with their own address spaces and thread(s)**
  - **Run on x86 HW**
  - Run on top of an API such as
    - Click
    - XORP
    - Orphal

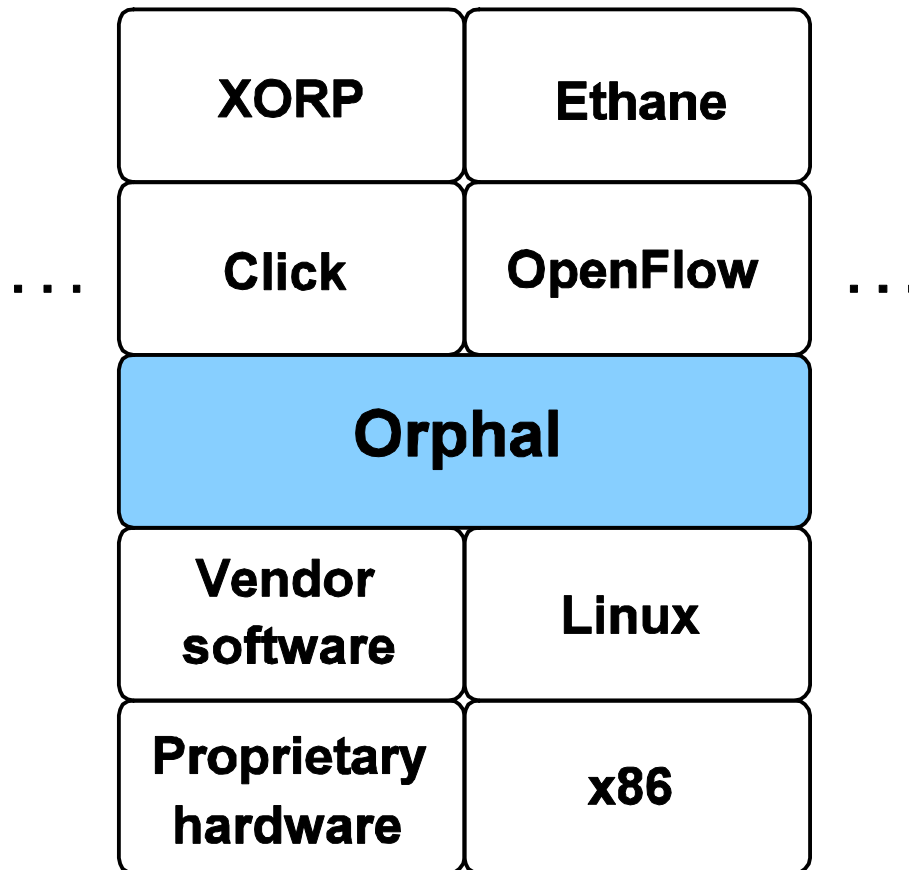
# What does a typical open router platform look like?



# What does a typical open router platform look like?



# Layering in an open router platform



# Examples of switchlets

- Specialized firewalls
  - Triggered by DPI HW to check unusual flows
- Specialized monitoring
  - to report on unusual patterns of flow creations
- NAT
- Dynamic VLAN
- OpenFlow
  - More detail later
- Click

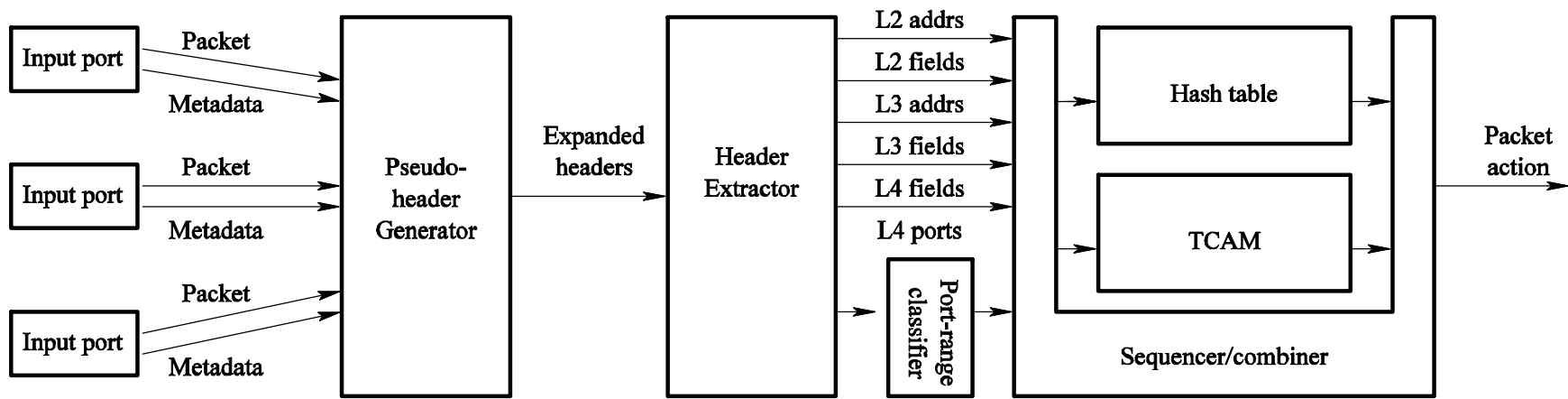
# Categories of switchlets

- Per-packet switchlets
  - Like Click elements
  - For performance reasons, limited to exceptional cases
- Per-flow switchlets
  - Especially for monitoring, firewalling
- Control-plane functions
  - Like XORP supports (e.g., routing protocols)
- Optimizer/helper modules (invoked by upcalls)
  - e.g., TCAM-rule optimizer; backing store for HW caches

# What are the design challenges for an API?

- Resource management
  - allocation of shared resources among switchlets
- Controlled sharing
  - when multiple switchlets must share a decision
- Isolation between switchlets
  - to avoid unexpected “feature interactions”
- Portability (more or less) between platforms
- Manageability
  - allow router admin to create & preserve a stable config

# Idealized TCAM-based lookup path (input side of a line card)



# What **API** are we proposing?

## Example TCAM-user API:

- tcamAddRow(tag, action, ordering)
- tcamDeleteRow(handle)
- tcamGetRow(handle)
- tcamRegisterInterest(handle, callbackFunction)
  - Allows a switchlet to receive packets and/or discover flows
- tcamConflictCallback(handle, callbackFunction)
  - Informs a switchlet that a conflict has been detected

## Example TCAM-optimizer API functions:

- Load a set of TCAM rows
- Obtain the abstract state of the TCAM database
- Get TCAM usage statistics

# Example: OpenFlow

- Some of us are porting OpenFlow
  - On an HP ProCurve 5406ZL switch (goal: line rate)
  - Not using Orphal yet!
- Challenges:
  - Limited number of TCAM rows
    - Use wild-card TCAM entry to match low-rate flows
    - Could implement OpenFlow as switchlet that forwards no-match packets to a controller node
  - Limited tag-field size
    - 144 bits is large enough for IP/TCP 5-tuple
    - But OpenFlow 10-tuple requires multiple TCAM lookups/packet
      - fortunately, the switch can do multi-stage lookups at line rate
      - Could use helper switchlet to do the pattern translation

# Related Work

- Other open architectures:
  - Click [Kohler et al. '00], XORP [Handley et al. '3]
- Interesting router hardware designs:
  - NetFPGA [Naous et al. '08], Casado et al. '08
- Lots of work on TCAMs
  - See paper for some of these
- Related work we had to leave out of the paper:
  - Router plugins [Decasper et al. '98]; SoftRouter [Laksman et al. '07]; IEEE P1520
  - All at higher levels of abstraction than Orphal
- Earlier uses of the term “switchlet”:
  - van der Merwe & Leslie '96; ALIEN [Alexander & Smith '99]; da Fonseca et al. '02

# Backup slides

# TCAM conflicts

- Kinds of conflicts include:
  - Two switchlets that “own” the same row
  - Rows from two switchlets that match the same packet
- Possible approach: define inter-switchlet ranking
  - Low-ranking switchlet “loses” its row(s) when conflict is detected
- Problems:
  - “Conflict” is hard to define
  - conflict-checking is NP-complete [McGeer & Yalagandula]
- Orphal can support plug-in conflict-checkers using helper switchlets
  - Rather than building these algorithms into the framework