XORP: An eXtensible Open Router Platform

Atanu Ghosh Mark Handley Orion Hodson
Eddie Kohler Pavlin Radoslavov
International Computer Science Institute

Adam Greenhalgh
University College London

Luigi Rizzo
University of Pisa

Networking research: divorced from reality?

Gap between research and practice in routing and forwarding

Most of the important Internet protocols originated in research, often at universities.

> It used to be that researchers designed systems, *built implementations*, *tried them out*, and standardized the ones that *survived and proved useful.*
>
> What happened?

Networking research: why the divorce?

The commercial Internet

- Network stability is critical, so experimentation is difficult
- Major infrastructure vendors not motivated to support experimentation

Network simulators

- High-quality simulators make a lingua franca for research

Simulation is not a substitute for experimentation

Many questions require real-world traffic and/or routing information

Most grad students:

- Give up, implement their protocol in *ns*
- Set *ns* parameters based on guesses, existing scripts
- Write a paper that may or may not bear any relationship to reality

We need to be able to run experiments when required!

The state of the art

Open APIs facilitate end-system protocol development

WWW, RTP, SIP, RTSP, . . .

Open-source OSes do the same for kernel changes

TCP SACK, IGMPv3, . . .

Also a history of experimentation in commercial OSes (affiliated labs)

Overlay networks may help with end-system/network interactions

Field in its infancy

What about protocols that affect the routers?

Option 1:

Persuade Cisco to implement your protocol;

Persuade ISPs that your protocol won't destabilize their networks;

Conduct experiment.

Option 2:

Implement routing protocol part in MRTd, GateD, or Zebra;

Implement forwarding part in FreeBSD, Linux, Click, Scout, . . . ;

Persuade network operators to replace their Ciscos with your PC;

Conduct experiment.

Likelihood of success?

Possible solutions

Solution 1:

A router vendor opens their development environment and APIs.

Thus, new router applications can be written and deployed by third parties.

Basic router functionality cannot be changed.

Solution 2:

Someone (*hint, hint*) builds a complete open-source router software stack explicit designed for extensibility *and* robustness.

Adventurous network operators deploy this router on their networks; it develops a reputation for stability and configurability.

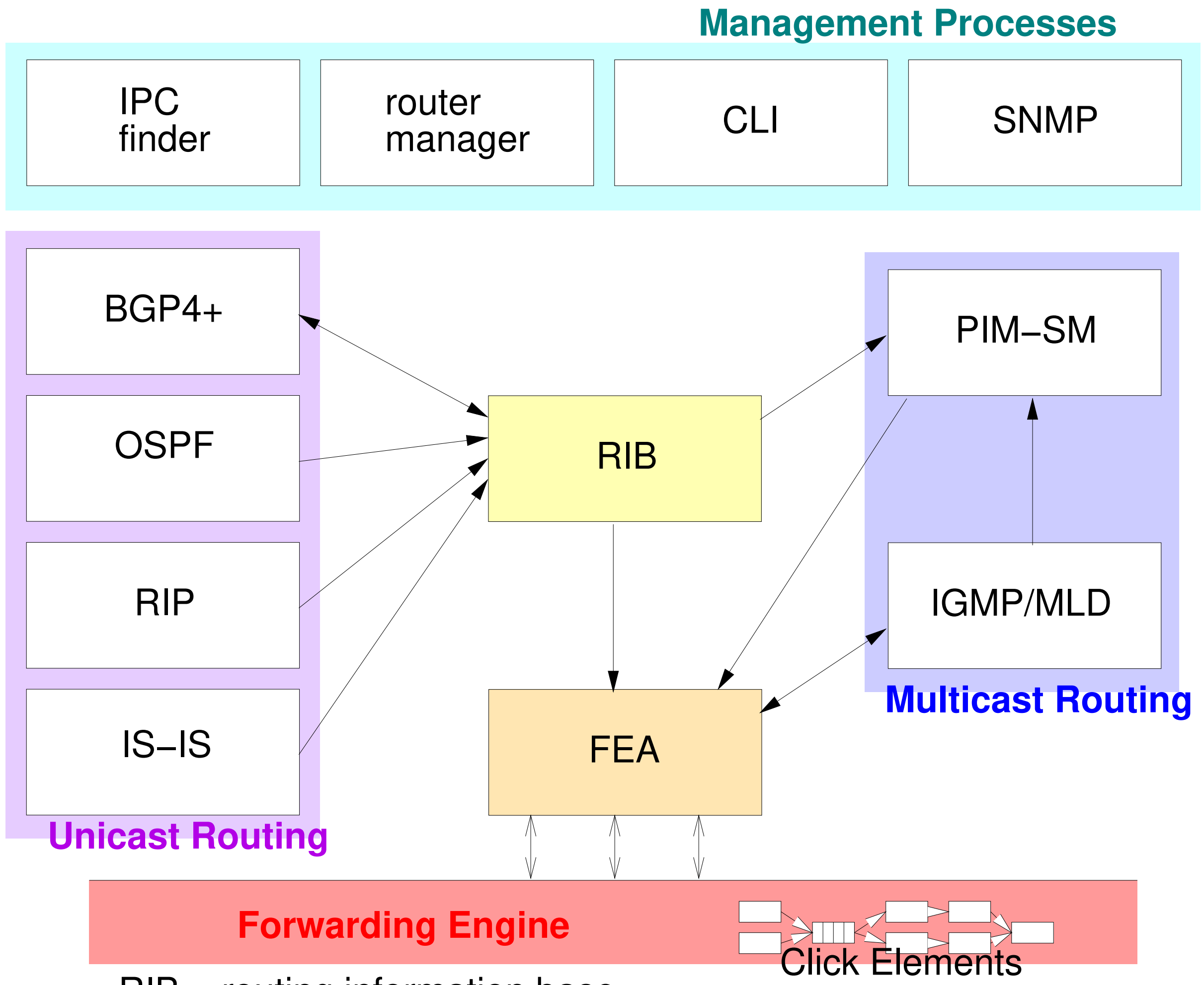
Result: a fully extensible platform suitable for research *and* deployment.

XORP

eXtensible Open Router Platform

Complete software stack for an IP router, including routing protocols, management interfaces, and forwarding path

Architecture



RIB = routing information base
FEA = forwarding engine abstraction

Four challenges

Features

- Real-world routers must support a long feature list

Extensibility

- Every aspect of the router should be extensible
- Multiple extensions should be able to coexist

Performance

- Not core routers, but edge routing is hard enough
- Raw forwarding performance, scalability in routing table size

Robustness

- Must not crash or misroute packets

Features

IPv4 and IPv6

Unicast routing protocols

BGP4+, OSPF, RIPv2/RIPng, Integrated IS-IS

Multicast

PIM-SM/SSM, IGMPv3/MLD

DHCP, PPP

Management

SNMP, command line, WWW

Forwarding elements

Route lookup, filter/firewall, ARP, AQM, encapsulation

Extensibility: Intra-router APIs

Separate abstract request (API) from concrete request (which process? which arguments? which version?)

In particular, the caller:

- Should not care about IPC mechanism
- Should not know in advance which process is relevant
- . . . unless required

Extensibility: XORP Resource Locators

XORP IPC mechanism

- Like URLs for IPC

```
finder://fea/fea/1.0/add_address4?vif:txt=fxp0&addr:ipv4=10.0.0.1
```

Extensibility: XORP Resource Locators

XORP IPC mechanism

Like URLs for IPC

`finder://fea/fea/1.0/add_address4?vif:txt=fxp0&addr:ipv4=10.0.0.1`

IPC mechanism: `finder`, `xudp`, `snmp`, ...

Extensibility: XORP Resource Locators

XORP IPC mechanism

- Like URLs for IPC

```
finder://fea/fea/1.0/add_address4?vif:txt=fxp0&addr:ipv4=10.0.0.1
```

Module/process name: `fea`, `rib`, `bgp`, ...

Extensibility: XORP Resource Locators

XORP IPC mechanism

Like URLs for IPC

`finder://fea/fea/1.0/add_address4?vif:txt=fxp0&addr:ipv4=10.0.0.1`

Interface name: `fea`, `routing-process`, ...

Extensibility: XORP Resource Locators

XORP IPC mechanism

- Like URLs for IPC

```
finder://fea/fea/1.0/add_address4?vif:txt=fxp0&addr:ipv4=10.0.0.1
```

Version number

Extensibility: XORP Resource Locators

XORP IPC mechanism

Like URLs for IPC

`finder://fea/fea/1.0/add_address4?vif:txt=fxp0&addr:ipv4=10.0.0.1`

Method name: `delete_address4`, `get_mtu`, ...

Extensibility: XORP Resource Locators

XORP IPC mechanism

Like URLs for IPC

```
finder://fea/fea/1.0/add_address4?vif:txt=fxp0&addr:ipv4=10.0.0.1
```

Arguments

Extensibility: XORP Resource Locators

XORP IPC mechanism

Like URLs for IPC

```
finder://fea/fea/1.0/add_address4?vif:txt=fxp0&addr:ipv4=10.0.0.1
```

Library marshals arguments, implements transport, handles responses

Redirection into a single XRL or an XRL sequence

Programmer explicitly handles failure

Using XRLs

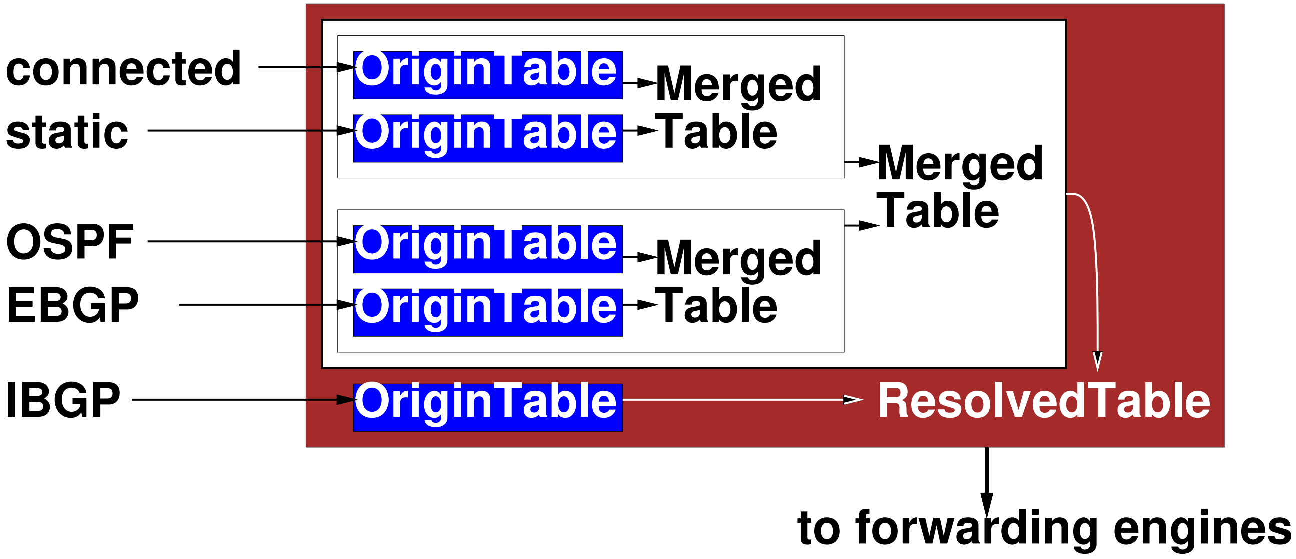
Interface files map (Juniper-style) configuration syntax . . .

```
protocols ospf {
   router-id: 128.16.64.1
   area 128.16.0.1 {
      interface xl0 {
         hello-interval: 30
}  }  }
```

. . . to XRLs

```
protocols.ospf {
   area.ADDR {
      interface.IFNAME {
         hello-interval {
            %set: xrl "ospfd/set_interface_param ? area_id:ipv4=ADDR
                       & interface:txt=IFNAME
                       & ospf_if_hello_interval:i32=VALUE";
}  }  }  }
```

Extensibility: RIB



Object-oriented routing table design

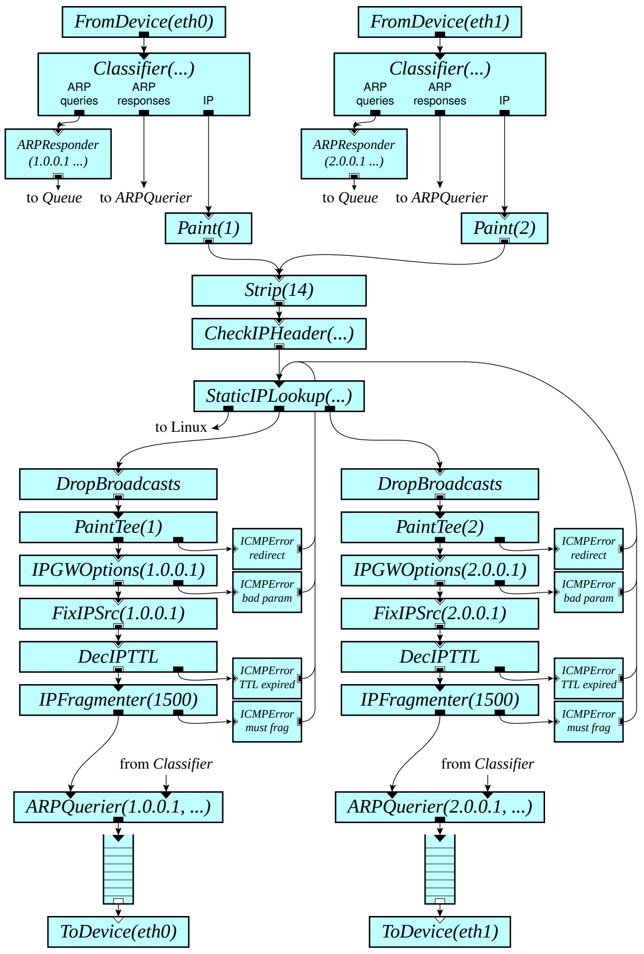
Add new merged tables implementing new merging policies, . . .

Extensibility/performance: Click forwarding path

Fast kernel forwarding

Easy to write extensions

XORP also supports native FreeBSD forwarding



Robustness

Policy decision: Strong robustness for user-level processes

- Difficult to get performance, robustness, and extensibility simultaneously
- Kernel robustness through inspection of extensions

Facilitated by multi-process design

- Automatically restart processes that crash
- Defensive programming for shared XORP processes like RIB

XRL sandboxes

- All interaction with router through XRLs, packets
- Redirect XRLs to run new protocols in a sandbox

Status

Core design, IPC, RIB, Click complete

- Routing tables, multicast, IPv6, Click integration in progress

All-new BGP, PIM-SM, IGMP in progress

Adapted OSPF, RIP in progress

First preliminary release within a month

- Check it out! Please help!

WWW.XORP.ORG