XORP Design Overview

Version 0.1

XORP Project
International Computer Science Institute
Berkeley, CA 94704, USA
feedback@xorp.org

December 11, 2002

1 Introduction

This document provides a brief overview of the XORP (eXtensible Open Router Platform) architecture. It is intended both for people who are interested in the architecture itself, and as a starting point for developers wishing to modify the software. People who are interested in the XORP multicast routing architecture should read also [7].

1.1 XORP Motivation

A gap exists between network research and Internet practice due to the nature of the Internet router software business. It is really hard for researchers to introduce new protocols and mechanisms into operational networks, and to study existing protocols in the wild. The primary goal of XORP to fill this gap. To succeed it must be both a research tool and a stable deployment platform that can be used in production networks. It must also place a strong emphasis on extensibility, while coping with the impact of this on robustness, security and performance.

For more details about the XORP design *philosophy*, see [12].

2 Target

The goal for XORP is to become a suitable software platform that can be used as the core of practically any router. However, the *initial* focus is on an edge router running on commodity PC hardware with relatively low port density and moderate size routing tables [1].

However, given the flexibility of the XORP architecture, in the future we should not be limited to edge-router scenarios or PC hardware. For example:

- We can easily imagine using multiple PCs as forwarding engines, with a single control element. This would allow greater port density.
- Another useful target would be a PC with a number of network processors (*e.g.,* Intel IXP1200) doing the bulk of the forwarding.

[1] A comparable production router might be a Cisco 7206VXR (or pretty much anything smaller than this).

- Finally, in the long run, the code base might be run the control processor of high-performance ASIC-based routers.

3 Functionality

XORP is designed to support both IPv4 and IPv6. However, even though most of the code is written to support both, as of Release 0.1 (December 2002) only IPv4 has been tested.

Below is our initial target list of protocols and features to be supported by XORP. Those in bold are already implemented or supported to some degree. Not all of the implemented features have been tested yet.

Unicast Routing Protocols

- **BGP4+** (**IPv4** and IPv6)
- OSPF (**IPv4** and IPv6)
- RIPv2 (IPv4), RIPng (IPv6)
- IS-IS

Multicast Routing Protocols

- **PIM-SM**, PIM-SSM, Bidir-PIM (IPv4 and IPv6)
- **IGMPv1, v2**, v3 (IPv4 only)
- **MLDv1**, v2 (IPv6 only)

Network Management

- **Command Line Interface** (similar to Juniper)
- SNMP
- WWW

Forwarding Path

- **Traditional UNIX forwarding path**
- Click forwarding path
- User-level simulation-like environment

4 Architecture

4.1 Design Philosophy

The XORP design philosophy stresses *extensibility*, *performance* and *robustness*.

For routing and management modules, the primary goals are extensibility and robustness. These goals are achieved by carefully separating functionality into independent modules, running in separate Unix processes, with well-defined APIs between them. Clearly there are performance penalties to pay for such an architecture, but we believe that so long as careful attention is paid to computational complexity, the costs associated with inter-process communication will be acceptable given the obvious extensibility and robustness benefits.

For the forwarding path, the primary goals are extensibility and performance. Robustness here is primarily achieved through simplicity and a modular design that encourages re-use of well-tested components.

4.2 Design Overview

XORP can be divided into two subsystems. The higher-level ("user-space") subsystem consists of the routing protocols and management mechanisms. The lower-level ("kernel") provides the forwarding path, and provides APIs for the higher-level to access.

User-level XORP uses a multi-process architecture with one process per routing protocol, and a novel inter-process communication mechanism known as XORP Resource Locators (XRLs) [4]. XRL communication is not limited to a single host, and so XORP can in principle run in a distributed fashion. For example, we can have a distributed router, with the forwarding engine running on one machine, and each of the routing protocols that update that forwarding engine running on a separate control processor system.

The lower-level subsystem can use traditional UNIX kernel forwarding, or the Click modular router [1]. The modularity and minimal dependencies between the lower-level and user-level subsystems allow for many future possibilities for forwarding engines. As standards such as those being developed in the IETF ForCES Working Group emerge, we expect to support them to provide true forwarding engine interchangeability.

Figure 1 shows the processes in XORP, although it should be noted that some of these modules use separate processes to handle IPv4 and IPv6. For simplicity, the arrows show only the main communication flows used for routing information. Control flows are not shown - for example, the FEA may need to inform the routing processes when an interface goes down. These processes are further described in Section 4.3.

We should note that even though our design philosophy is that each XORP component in Figure 1 will be run as a separate process, it would also be possible to compile most of them together to run as one single process. Obviously, the robustness of such a router would suffer because the crash of a single component would bring down the whole router. However, the XORP architecture is designed to be flexible, and other developers building on this software can choose to use it in different ways.

4.3 XORP Processes Description

4.3.1 FEA (Forwarding Engine Abstraction)

The FEA provides a platform independent interface to the basic routing and network interface management functionality. For example, get or set information about network interfaces, install or modify unicast forwarding entries, etc.

If the router is distributed, in the sense that some Forwarding Engines (FEs) are not in the same chassis as the control software, then the FEA will also handle control communications with the remote FEs. Note that strictly speaking it is not required that all communication with the FE must go through the FEA. For example, Click modules can communicate directly with an user-space process. However, the FEA abstracts all the details about the underlying system from the user-level XORP processes; therefore by using the common API provided by the FEA we can greatly simplify the rest of the XORP components.

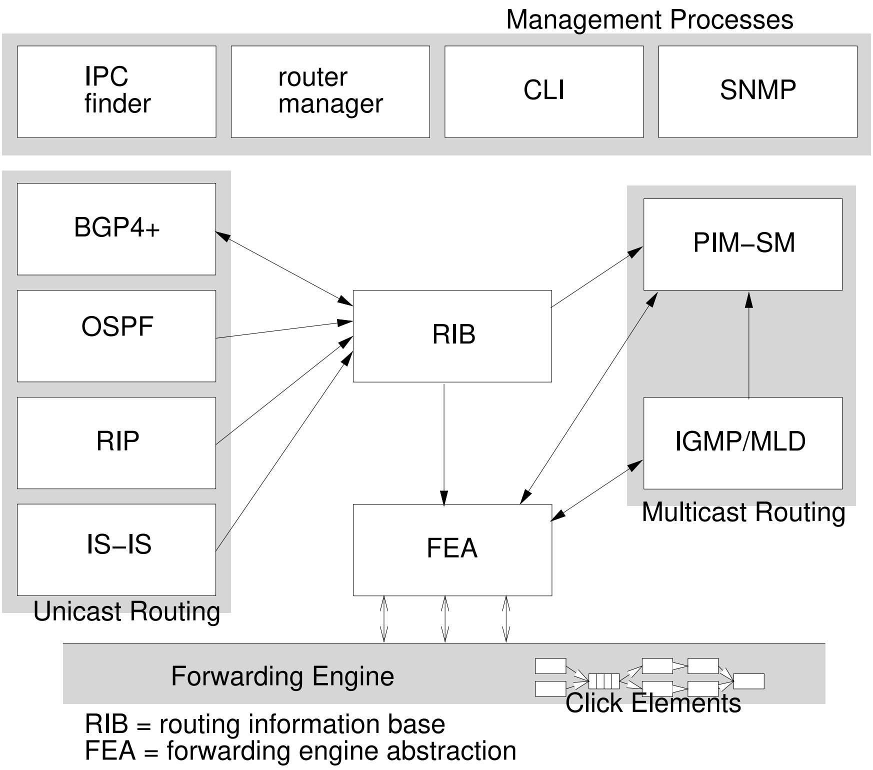

Figure 1: XORP Process Model

Note that in release 0.1 (December 2002), the multicast-related functionalities are separated from the unicast-based functionalities in the MFEA (Multicast Forwarding Engine Abstraction).

For more information about the FEA see [3]. For more information about the MFEA see [6].

4.3.2 RIB (Routing Information Base)

The RIB holds a user-space copy of the entire routing/forwarding table, complete with information about where each route came from (*e.g.,* which protocol, and when). It communicates with the routing protocols such as BGP, RIP and OSPF to instantiate routes, and with the FEA to install the appropriate forwarding entries in in the FEs.

The RIB also holds the routing information for multicast-capable routes (MRIB) to be used for Reverse-Path Forwarding (RPF) information. etc. For example, PIM-SM uses the MRIB information to route joins/prunes and to determine the RPF interface for sources and for RPs.

The MRIB is populated by whatever "unicast" protocol is used to provide multicast capable path information. Typically this is MBGP for inter-domain paths, where it is possible to tell the difference between unicast-capable and multicast-capable routers. For intra-domain routing, this usually is the regular unicast FIB information.

Note that for unicast, routing and forwarding tables are practically the same. In case of multicast, the RIB provides the RPF information, while the forwarding information (the incoming and outgoing interfaces) is computed by the particular multicast routing protocol, but this information is kept by the multicast routing protocol itself, and installed directly through the FEA. In the future, when there is more than one multicast routing protocols, XORP may have the multicast equivalent of RIB that would be responsible for coordinating among the different multicast routing protocols running on the same router.

On a router with multiple FEs, the RIB is responsible for splitting up the routing table amongst the FEs and for figuring out how to forward between FEs.

For more information about the RIB see [10].

4.3.3 BGP4+

This is the BGP routing daemon. It implements IPv4 and IPv6 unicast routing, as well as MBGP for both IPv4 and IPv6 multicast RIBs for multicast routing purpose.

For more information about the XORP BGP implementation, see [2].

4.3.4 OSPF

This is the OSPF routing daemon. There are separate IPv4 and IPv6 daemons, because unlike BGP there is no real need to tie them together.

Currently (December 2002), we intend to use a port of an existing OSPF implementation (http://www.ospf.org/) to the XORP architecture. However this does not support IPv6.

4.3.5 RIP

This is the RIP routing daemon. Similarly to OSPF, the IPv4 and IPv6 daemons are separate.

Currently (December 2002), we intend to port existing RIP implementations (*r*outed and *r*oute6d from FreeBSD) to the XORP architecture.

4.3.6 IGMP/MLD

This is Internet Group Management Protocol/Multicast Listener Discovery handler. It implements the router-side part of IGMP and MLD. Its main purpose is to discover local multicast members and propagate this information to multicast routing daemons such as PIM-SM.

For more information about the XORP IGMP/MLD implementation see [5].

4.3.7 PIM-SM

This is the PIM-SM multicast routing daemon. Similar to OSPF and RIP, the IPv4 and IPv6 daemons are separate. The PIM-SM protocol requires information about local multicast members, and Reverse-Path Forwarding to operate properly. The former is obtained from the IGMP/MLD process; the latter is obtained from the RIB.

For more information about the XORP PIM-SM implementation see [8].

4.3.8 RTRMGR: XORP Router Manager

The RTRMGR is the process responsible for starting all components of the router, to configure each of them, and to monitor and restart any failing process. It also provides the interface for the CLI to change the router configuration.

For more information about the RTRMGR see [9].

4.3.9 CLI: Command Line Interface

The CLI is intended to be used by an user to access the router, view its internal state, or to configure it on-the-fly. Its functionality is closely related to the RTRMGR. However, because the robustness of the RTRMGR itself is extremely important, all functionality that can be run as a separate CLI process are separated from the RTRMGR.

For more information about the CLI see [9].

4.3.10 Inter-Process Communication Finder

The IPC finder is needed by the communication method used among all XORP components, *i.e.,* the XRLs. Each of the XORP components registers with the IPC finder. The finder assists the XRL communications (more specifically, it knows the location of each XRL target), therefore a XORP process does not need to know explicitly the location of all other processes, or how to communicate with them.

For more information about the IPC finder and XRLs see [4] and [11].

4.3.11 SNMP

This is the SNMP management process. It is used for SNMP access to the router. For example, it can be used to translate SNMP requests into XRL requests. Currently (December 2002) it has not been implemented yet.

A Modification History

- December 11, 2002: Version 0.1 completed.

References

[1] The Click Modular Router Project. http://www.pdos.lcs.mit.edu/click/.

[2] XORP BGP Routing Daemon. XORP technical document. http://www.xorp.org/.

[3] XORP Forwarding Engine Abstraction. XORP technical document. http://www.xorp.org/.

[4] XORP Inter-Process Communication Library. XORP technical document. http://www.xorp.org/.

[5] XORP MLD/IGMP Daemon. XORP technical document. http://www.xorp.org/.

[6] XORP Multicast Forwarding Engine Abstraction. XORP technical document. http://www.xorp.org/.

[7] XORP Multicast Routing Design Architecture. XORP technical document. http://www.xorp.org/.

[8] XORP PIM-SM Routing Daemon. XORP technical document. http://www.xorp.org/.

[9] XORP Router Manager Process (rtrmgr). XORP technical document. http://www.xorp.org/.

[10] XORP Routing Information Base (RIB) Process. XORP technical document. http://www.xorp.org/.

[11] XRL Interfaces: Specification and Tools. XORP technical document. http://www.xorp.org/.

[12] Mark Handley, Orion Hodson, and Eddie Kohler. XORP: An Open Platform for Network Research. In *Proceedings of HotNets-I Workshop*, Princeton, New Jersey, USA, October 2002.