XORP Inter-Process Communication Library Overview

Version 1.2

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

March 8, 2006

Abstract

Extensibility and robustness are key goals of the eXtensible Open Router Project (XORP). A step towards satisfying both of these goals is to separate the functional components of a router into independent tasks and to execute each of these tasks as separate processes. From an extensibility perspective this allows components of the router to be started, stopped, and exchanged dynamically, and to be distributed across a number of hosts. From a robusness perspective, the processes are afforded the protection mechanisms afforded by modern operating systems so a failure of one routing task does not necessarily bring the router to a halt. The routing tasks do need to communicate and we have developed an asynchronous remote procedure call mechanism that is capable of working with multiple transport protocols between remote hosts and can leverage existing IPC mechanisms within a single host. This document discusses aspects of the design and the directions it may take in future.

1 Introduction

Robustness and extensibility are two of the goals of the XORP project. One way a router can achieve robustness is to run routing protocols in protected environments, such as separate userland processes on a modern operating system. And one way a router can achieve extensibility is to be independent of the details about where those routing processes are running and what the composition of the forwarding plane is. The routing processes and network interfaces could be running on one machine or distributed across a cluster of machines that appear as single router. A necessary feature once routing protocols are running in distinct processes and potentially on distinct machines is an inter-process communication mechanism. In contrast to traditional inter-process communication schemes, the scheme employed in the XORP project can utilize multiple transport protocols and potentially communicate with unmodified components through these protocols, for instance SNMP or HTTP.

The lofty goals of XORP's Inter-Process Communication (XIPC) scheme are:

- to provide all of the IPC communication mechanisms that a router is likely to need, *e.g.*sockets, ioctl's, System V messages, shared memory.
- to provide a consistent and transparent interface irrespective of the underlying transport mechanism used.

- to transparently select the appropriate IPC mechanism when alternatives exist.
- to provide an asynchronous interface.
- to be efficient.
- to potentially wrapper communication with non-XORP processes, *e.g.*HTTP and SNMP servers.
- to be renderable in human readable form so XORP processes can read and write commands from configuration files.

The XIPC goals are realized through XORP Resource Locators (XRLs) and the XORP IPC (XIPC) library. XRLs are responsible for describing an inter-process calls and their arguments. An XRL may be represented in human readable form that allows for easy manipulation with editing tools and invocation from the command line during development.

XORP processes export XRL interfaces to a process known as the *Finder* and inform it of which IPC schemes are available to invoke each XRL. The Finder is then able to provide a resolution service for XORP processes. When a process needs to dispatch an XRL it first contacts the Finder, obtains details on which IPC mechanisms are available to contact the process, and then instantiates a suitable transport.

The XIPC library provides the framework for handling and manipulating XRLs, communicating with the Finder, and common protocol families for conducting IPC. In addition to the XIPC library, an interface definition language exists, together with tools to translate these into callable C++ interfaces and into a set of C++ handler routines for handling the receipt of XRL calls. These tools are described in document [1]. The tools reduce the amount of familiarity the working programmer needs to have with the internals of the XIPC library. This document provides an overview of the XIPC library and is the recommended starting point before using the library.

2 XORP Resource Locators (XRL's)

The mechanism we've settled on for IPC within XORP processes is mediated through *Xorp Resource Locators* (XRL's). An XRL describes a procedure call. It comprises the protocol family to be used for transport, the arguments for the protocol family, the interface of the target being called and its version, the method, and an argument list. Examples of XRLs in their human readable forms are shown in figure 1. The existence of a human readable form for XRLs is chiefly a convenience for humans who need to work with XRLs and not indicative of how they work internally.

Resolved and unresolved forms of the same are XRL are depicted in figure 1. The unresolved form is the starting point for the majority of inter-process communication. In the unresolved form the protocol family is set to "finder" and the protocol parameters set to the target name that the XRL call is intended for. A process wishing to dispatch an XRL for the first time passes the unresolved XRL to the Finder, which returns the resolved form with the appropriate protocol family and protocol family arguments. The finder may also modify the other components of the XRL, but doesn't usually do so. This functionality may be useful when supporting backwards compatibility of interfaces, *i.e.*the Finder could modify the interface number and method name.

The resolved forms of XRLs are typically maintained in a client side cache so the Finder need not be consulted for each XRL dispatch.

(a) Unresolved form:

```
finder://fea/fti/0.1/add_route?net:ipv4net=10.0.0.1/8&gateway:ipv4=192.150.187.1
+-----   +-- +-- +-- +-------- +-------------------------------------------------
|        |   |   |   |         |
|        |   |   |   Method    Arguments
|        |   |   |
|        |   |   Interface version
|        |   |
|        |   Interface Name
|        |
|        Protocol Parameters
|
Protocol Family
```

(b) Resolved form:

```
stcp://192.150.1.5:1992/fti/0.1/add_route?net:ipv4net=10.0.0.1&gateway:ipv4=192.150.1.1
+---   +--------------  +-- +-- +-------- +---------------------------------------------
|      |                |   |   |         |
|      |                |   |   Method    Arguments
|      |                |   |
|      |                |   Interface version
|      |                |
|      |                Interface Name
|      |
|      Protocol Parameters
|
Protocol Family
```

Figure 1: Human readable XRL forms.

3 XRL Targets and the Finder

An *XRL Target* is an entity that is capable of dispatching XRLs. The *Finder* is the entity in XIPC system that knows about XRL Targets and is responsible for directing IPC communication between XRL Targets. The IPC communication schemes that XRL Targets use to communicate XRLs between each other once they have been resolved by the Finder are known as *Protocol Families*.

Each XRL Target has an associated class name and an instance name. Class names indicate the functionality that the target implements and there may be multiple targets in the XIPC system with the same class name. Instance names are unique identifiers for each target in the XIPC system. XRLs to be resolved by the Finder may address a target by class name or by instance name. The Finder treats the first instance in a class as the primary instance of that class and XRLs directed to a class resolve equivalently to the primary instance.

The class of an XRL Target dictates which methods the target supports. Interfaces are collections of XRLs that relate to some aspect of functionality and objects in a given class implement handlers for one or more interfaces. Once an XRL Target has registered its presence with the Finder, it registers the XRLs associated with the interfaces it supports. XRLs are registered one at a time and only when all of the XRLs have been registered does the XRL Target become visible to the outside world through the Finder.

The Finder provides a one-to-many mapping service. Each XRL Target may specify multiple protocol families for each XRL they export. When a target requests the resolution of an XRL, the answer will contain the complete list of available resolutions and the resolver can decide which it would prefer to use. When registering the target indicates the XRL method name and an appropriate protocol family. so targets that support multiple protocol families perform a separate registration each XRL and protocol family pair. This provides a flexible system for implementing optimizations on the level of individual XRLs.

There is a slight subtlety with registration for the sake of security. This arises because some of the protocol families that can be used to communicate XRLs and responses to XRLs allow for remote access, examples being UDP and TCP protocol families. The source code to the XORP project is publicly available and the XRL interfaces and targets are included the package. With knowledge of the XRL interfaces and targets it would be relatively straightforward for external parties to opportunistically dispatch XRLs on a XORP router. However, if XRL communication can be coerced to use the Finder, access control can be enforced centrally. To achieve this, the Finder performs XRL method name transformation. When an XRL is registered with the Finder, the Finder transforms the resolved form of the XRLs method name, and instructs the registering target to dispatch the transformed name as if it were the original name. The method name transformation is designed to be hard to counter and each XRL method is transformed uniquely. The registering process therefore has to maintain a mapping table from the interface.

In addition to handle XRL registrations and resolutions, the Finder is capable of notifying XRL Targets about events it is aware of, like the birth and death of other XRL Targets. The Finder exposes an XRL interface for this purpose and is able to invoke XRLs on XRL Targets to perform the notification. A special tunneling mechanism exists in the communication protocol used to communicate with the Finder for this purpose. The details of this communication will be expanded upon later on [XXX in a later edit to this document].

4 Components of XRL Framework

XRL an inter-process call that is transparent to the underlying transport method.

Finder the process that co-ordinates the resolution of target names into a parseable form to find the XRL Protocol Family Listener.

XRL Router an object responsible for sending and receiving XRLs. They manage all the underlying interactions and are the interface that users are expected to use for XRL interactions.

Finder Client an object associated with an XRL Router that manages the communication with the Finder.

XRL Protocol Family a supported transport mechanism for the invoked XRL.

XRL Protocol Family Sender an entity that dispatches XRL requests and handles responses. Senders are created based on Finder lookup's of the appropriate communication mechanism.

XRL Protocol Family Listener an entity that listens for incoming requests, dispatches the necessary hook, and sends the responses. When Listeners are created they register the appropriate mapping with the Finder so that corresponding Senders can be instantiated to talk with them.

The kdoc documentation provides details of the particular classes.

References

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