XORP Inter-Process Communication Library Overview

Version 0.1

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

December 11, 2002

Abstract

Extensibility and robustness are key goals of the eXtensible Open Router Project (XORP). A step towards these goals is separating the tasks of router in multiple userland programs. The programs may need to communicate with the kernel and may need to communicate across multiple hosts in the case of a distributed router. 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 note documents the initial design and directions it may take.

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, sysV messages and 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). An XRL describes a procedure call. A resolver process, the *Finder*, translates XRLs into methods for inter-process communication. Each XRL has a textual description and each XORP process supports multiple XRL call interfaces, each comprised of a group of related XRLs that implement some aspect of the processes exported functionality. When a XORP process starts it registers its supported XRLs and transport protocols with the Finder. The Finder is then able to inform processes with sufficient privileges how to resolve to resolve those XRLs.

The XIPC library consists of a set of routines for building and manipulating XRLs, for dispatching and handling the responses from XRLs, and converting XRLs to and from human readable text. This document describes XRLs, the XIPC library, and the issues involved in it's design. The internals of the library are subject to change and some portions, such as access control, are not yet implemented in the library.

In addition to the XIPC library, an interface definition language exists, and 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 [**?**].

The tools reduce the amount of familiarity the working programmer need 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 for IPC within XORP processes is mediated through *Xorp Resource Locators* (XRL's). An XRL describes a procedure call. It comprises of the protocol family to be used for transport, the arguments for the protocol family, the interface of the target being called and it's 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, ie 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:

```
xudp://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 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 dispatching and receiving XRL requests. 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.