Proposal to Develop an Extensible Open Router Platform

Mark Handley

November 30, 2000

1 Introduction

IP router software currently differs from software written for desktop computers in that it tends to be written for a specific manufacturer's router family rather than being portable across systems from many different manufacturers. Whilst this makes sense in the case of code that interacts closely with high-speed forwarding engines, this observation also holds true for higher-level software such as routing protocols and management software. In addition, the APIs that would enable such higher-level software to be written by third parties are typically also not made public.

We propose to develop an open and extensible software platform for routers that might change this router software development model. There are two main parts to such a platform:

- Higher-level routing code comprising the routing protocols, routing information bases, and management software necessary to exist in today's Internet.
- Low-level kernel code comprising an efficient forwarding path, together with the appropriate APIs to talk to routing code and to talk to additional higher-level functionality that is likely to be added later.

At the higher-level, the project needs to develop an architecture for interconnecting higher-level protocol modules in a manner that is efficient, but also modular and flexible. The APIs between these modules need to be carefully designed and well specified to allow third-parties to contribute new modules that extend the functionality of the router. It is important that such extensions can be in binary-only form so that a wide range of third-party business models are possible. It is also important that a router can use well-verified trusted routing modules at the same time as experimental third-party modules without unduly compromising the stability of the router. In this way such a router platform would stimulate early experimentation and deployment of new network functionality. For good operational reasons, this is extremely difficult in the current Internet.

At the low level the architecture needs to be capable of spanning a large range of hardware forwarding platforms, ranging from commodity PC hardware at the low end, through mid-range PC-based platforms enhanced with intelligent network interfaces such as Intel's IXP1200[1], to high-end hardware-based forwarding engines. Initially we plan to focus on a PC-based hardware platform as this allows for easy testing and early deployment, but great care will be taken to design a modular forwarding path architecture which can support some or all of the forwarding functions happening in hardware. We expect ideas taken from the Streams[2] network stack abstraction and from the Click[3] modular router to be of use in designing this part of the platform.

2 Target User Base

The Open Router platform is initially aimed at a number of different user-bases:

- Third-party router vendors that lack an existing credible software base.
- Third-party makers of intelligent network hardware for commodity platforms.
- Small organizations needing cheap low-to-medium power router solutions which are within the performance range achievable with commodity computer hardware.
- Research organizations needing an open platform on which to develop new experimental networking functionality.

3 Modular Routing Protocol Code

To be successful, such a router platform needs to be have good support for the routing and management protocols that are in widespread use today. This initial list includes (but is not limited to):

Routing Protocols:

- BGP4+
- OSPF
- RIP

- IS-IS
- PIM-SM/SSM
- IGMPv3/MLD

Management Interfaces:

- SNMP
- Cisco-compatible command interface over ssh and telnet
- HTTP interface

We expect that this list will probably change as a result of consultation with ISPs. Where relevant, these protocols should be both IPv4 and IPv6 capable.

To develop all these routing protocols from scratch is probably not necessary as much of the routing code from the open-source Merit MRTd routing daemon can be reused, although the basic architecture of MRTd is not itself suitable to satisfy the requirements of the Open Router platform.

We envisage an architecture into which new routing protocols and other network elements may be plugged at run time. Clearly routing protocols running on the same router need to interact with each other (eg BGP with OSPF, IGMP with PIM), and so a key part of the architecture will be a standard way of interconnecting such modules with the forwarding information base, and with each other where necessary. We plan to use off-the-shelf IPC/RPC mechanisms for this interconnection, possibly enhanced with shared memory in the few places where IPC performance is critical, but careful design will be required to avoid deadlock problems that are common with such architectures.

An important aspect of such an architecture is good support for policy management. Whilst this is a part of the platform that is likely to gain functionality relatively late in the development process, it is critical that all of the early work on routing protocol integration is done in the context of a policy management architecture.

4 Low-level Platform and APIs

The low-level platform needs to provide OS support to the routing protocols, including memory management and scheduling. This can best be provided by taking an existing stable open-source operating system as the base development platform rather than developing a complete OS from scratch. However, the packet forwarding part of existing operating systems is not sufficiently modular to effectively support the wide range of forwarding hardware that is required, nor is it sufficiently easily extended to add new functionality. In addition the existing API to the higher-level processes is also inappropriate, given that it will eventually need to provide a consistent interface to a rich assortment of forwarding mechanisms and hardware.

We plan to take an existing open-source Unix OS as our base operating system, although the forwarding code should be portable to other platforms with some effort. The original Unix forwarding code will be removed, and replaced with a new modular forwarding path, which will be designed with the concept of handing off some or all of the forwarding functionality to forwarding hardware on platforms where this is available. The forwarding path will also be designed with "distributed routers" in mind, where much of the forwarding work takes place between smart network interface cards. However, our *initial* implementation will probably not target advanced hardware platforms due to limited resources. The forwarding path should permit fast efficient forwarding of IPv4 and IPv6 packets, a range of queuing and queue management mechanisms, and have hooks for QoS mechanisms including Diffserv. We would plan to reuse good existing open-source code to construct such forwarding modules, where such code is available, but to restructure it so that it is consistent with our planned forwarding architecture.

The kernel API to the forwarding path will be replaced with one that is more appropriate for today's routing protocols. The routing/forwarding API will be supplemented with an appropriate API to allow extensions to be added allowing the future addition of non-routing extensions such as network intrusion detection, traffic normalizers, and caching. There is ongoing standardization work happening in the area of APIs for routers, including IEEE P.1520.3 "Standard for APIs for Internet Protocol Network Elements". Where appropriate, we will either include support for such APIs, or base our API around the appropriate standard.

Routing protocols make extensive use of timers to function correctly, and so providing enhanced support for application-level timers is also be a likely enhancement. No doubt a number of additional low-level changes will required to provide good support for the higher-level tasks, but it is not yet clear what these will be.

5 Project Staffing

This project is an ambitious undertaking — to produce a router platform that has sufficient features to be credible to ISPs, whilst being sufficiently stable to be trusted, is not an easy task. In principle, an open-source development model allows significant third-party enhancement and debugging and, if managed well, eventually results in very stable code such as FreeBSD and OpenBSD. However this only happens when such a project reaches

a critical mass, which comprises sufficient features and stability. This document outlines the basic functionality we believe is necessary to achieve critical mass. There are many additional features that such a platform would eventually require, and we hope that third-party coding support will add these features in time. Until we have a working, reasonably stable, acceptably featured platform, we cannot expect significant community coding involvement.

We expect that the project would require approximately 5 full-time employees to bring the platform to this basic level. Whilst we do not expect to have a one-to-one mapping of people to roles, these 5 positions are likely to breakdown in roughly the following way:

- Project management/systems architect: 0.5 FTE
- Unicast routing development and testing: 1.0 FTE
- Multicast routing development and testing: 0.75 FTE
- Network management/configuration development and testing: 1.0 FTE
- Low-level systems platform development and testing: 1.5 FTE
- Documentation: 0.25 FTE

We do not believe that all 5 FTEs need to be involved from the start of the project, but we do envisage needing to achieve this level of staffing within Year 1. We envisage the basic development phase of the project taking three years in total.

References

[1] Intel IXP1200 Network Processor. http://developer.intel.com/design/network/ixp1200.htm

[2] Dennis M. Ritchie, "A Stream Input-Output System", AT&T Bell Laboratories Technical Journal 63, No. 8 Part 2 (October, 1984), pp. 1897-1910. http://cm.bell-labs.com/cm/cs/who/dmr/st.html

[3] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek, "The Click Modular Router", In ACM Transactions on Computer Systems, to appear (August 2000). http://www.pdos.lcs.mit.edu/click/

[4] Merit Multi-threaded Routing Toolkit (MRTd). http://www.mrtd.net/

[5] IEEE Recommendation P.1520.3 "Standard for Application Programming Interfaces for Internet Protocol Network Elements", work in progress. http://www.ieee-pin.org