Open Network Platform

Andrew McRae

amcrae@employees.org

Abstract

Much research in networking takes place outside of routers and network switches, mostly in hosts. This can cause constraints due to the fact that the researcher cannot effectively incorporate new applications or test facilities into proprietary equipment, because of the closed nature of the equipment.

Alternatives such as using dedicated Linux hosts and the like are not always possible due to the lack of performance or lack of sophisticated networking features.

This paper presents a framework that exists on the Netdevices mid-range router designed to allow customers and researchers much closely integration of their applications and research tools to the heart of the packet processing engine. The intent is to allow customers and researchers to run their code actually within the router, for maximum effect, but in a way that is secure and robust.

Introduction

Traditionally, routers and standalone network devices are black boxes; configuration and management is performed through typical console or network connected management services such as a CLI, SNMP, embedded web servers or proprietary device management services or programs. The level of configuration usually extends to the explicit configuration and control of pre-programmed features and applications that are part of the device e.g interface attributes, protocol addresses, routing protocol configuration etc.
However, users sometimes desire a greater level of access via their network devices, for a variety of reasons:

- Network or protocol debugging. Sometimes users require low level capture of specific types of packets, or wish to selectively monitor traffic, or track and maintain protocol specific context. This is usually done for low level network topology debugging, or when developing, testing or tuning specific network protocols.
- More dynamic or flexible configuration options. Routers generally have a static configuration, and often changing parameters or attributes of the configuration will cause some disruption or reload of existing configuration. Users who have a need to dynamically modify parameters based on some user-specific algorithm or external source may have limited options on how to best incorporate fast changing parameters into the static configuration. For example, an ISP may wish to dynamically modify user profiles according to the network load, or time of day, or current available bandwidth. This may require reloading of QoS configuration on a fast changing basis.
- Dedicated user-specific applications. Users may develop applications specific to their network requirements such as tracking certain connections, or monitoring certain hosts.

Existing Facilities

Over the development of routers, more facilities have been introduced to provide for user requirements so that greater monitoring or access is allowed. These facilities include:

- RMON, a framework whereby packets on interfaces can be counted or traced and the results uploaded to a remote host.
- Netflow, where TCP/UDP sessions are maintained in a flow table, and summary information uploaded or stored for the sessions.
- Access Control Lists, where classification of packets for filtering allows logging and traps of particular identified packets.
- Introduction of dedicated features such as Intrusion Detection, where in-built or downloaded rulesets are used to track and log suspect packets.

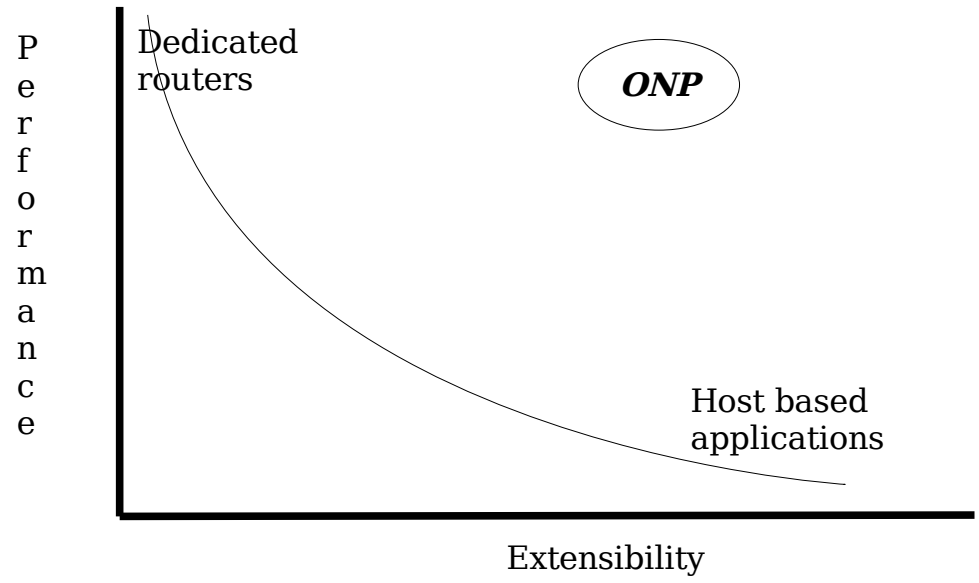
A common event in the networking device market is the emergence of niche dedicated devices performing some function that has not yet been incorporated in mainstream devices. Examples include stateful firewalls, IDS (Intrusion Detection Systems), QoS or other traffic management systems etc. The usual progression is that niche devices meet the initial requirements, and then eventually the functionality gets incorporated into mainstream devices. Mainstream routers often suffer a considerable lag of new functionality due to the complexity of developing new features, software release and stability

requirements, or the fact that specialised hardware is sometimes used and the feature requires some significant development to incorporate it into the hardware architecture of the router.
Ultimately, whilst these facilities meet the need of a significant portion of requirements, a key problem is that routers or dedicated devices are closed devices that do not allow the user to incorporate any of their own applications (or if so, in a very limited fashion). There is little or no *extensibility* in these devices.

However, for many years alternatives to dedicated routers have existed i.e the use of standard workstations and computers that can be used as packet forwarders. Since the initial development of networking on Unix through software such as the BSD networking releases, Unix systems have very often been used as flexible and highly programmable routers, often used for the development of new features and applications for networking. In fact, a large range of facilities have been widely available to support this type of application, such as the Berkeley Packet Filter, tunnel and tap interfaces, and even the standard socket API for capturing raw packets. Within the Linux kernel, a variety of different APIs are available to allow developers to develop sophisticated networking applications. However, this approach suffers from a number of limitations:

- One challenge is how to incorporate these applications within the existing kernel based packet processing path in a way that is efficient and robust, without having to resort to complex kernel modules. A common solution here is to employ the packet tap APIs that exist in the kernel, and perform all the processing in user space.
- The performance of a Linux based system is often quite a bit lower than dedicated routers. This may be due to a combination of poor hardware architecture of NICs, lack of hardware offload support, inefficiencies in kernel/user packet processing handoff etc.
- If the application is a monitoring one, the use of a Linux system often means this has to be added as a separate device in the network. If many networks are to be monitored, then a separate system has to be added to each network.

In summary, routers and dedicated devices are fast, but not extensible, and Linux based applications are very programmable, but not fast or not close to the heart of the network:



The goal of the Open Network Platform (ONP) is to provide a framework for user applications as part of a high performance mid-range router. The ONP's goal is to bridge the gap between highly programmable, user-application supporting devices, and high performance networking.
An alternative arrangement is also presented, describing how this framework can operate on off-the-shelf platforms.

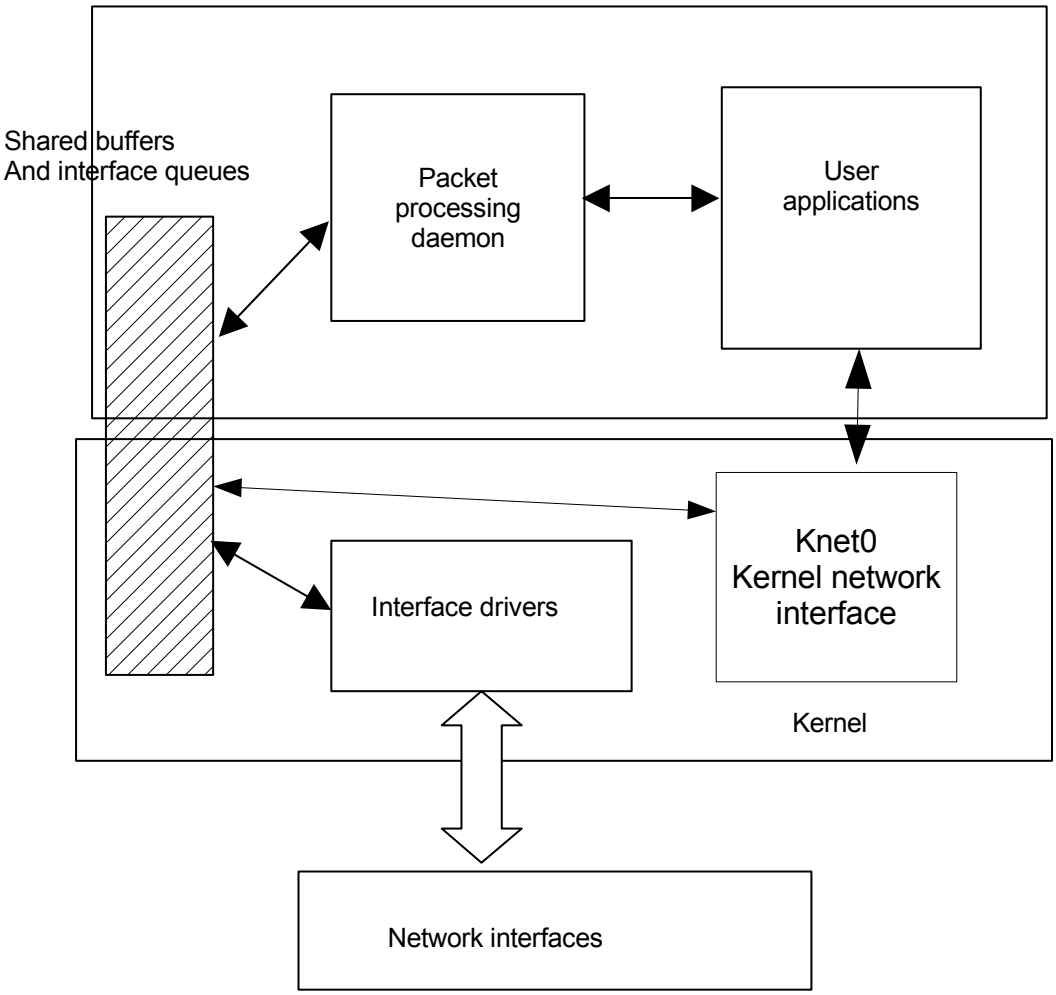
Hardware Platform

The NetDevices SG-8 is a state-of-the-art services gateway that incorporates many different networking applications in a single integrated device, offering a high degree of manageability, performance and robustness.

Physically, the SG-8 consists of a number of separate hardware hot-swappable modules, interconnected with a high speed industry standard PCI-Express bus. A unique feature is the

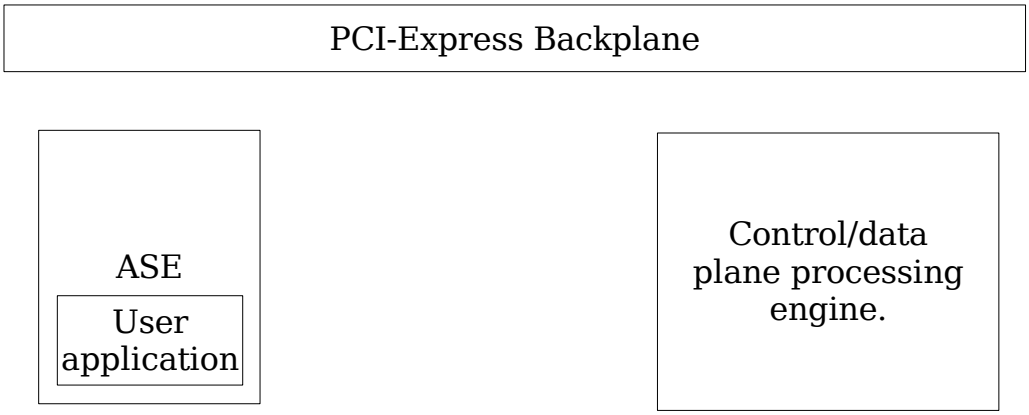
introduction of a cluster of management orientated CPUs, separating the management functions of the device from the control and data plane.

The control and data plane runs on a dedicated Opteron CPU, using a Netdevices designed user level networking architecture.



The user level networking architecture provides a high performance processing path for packet forwarding.

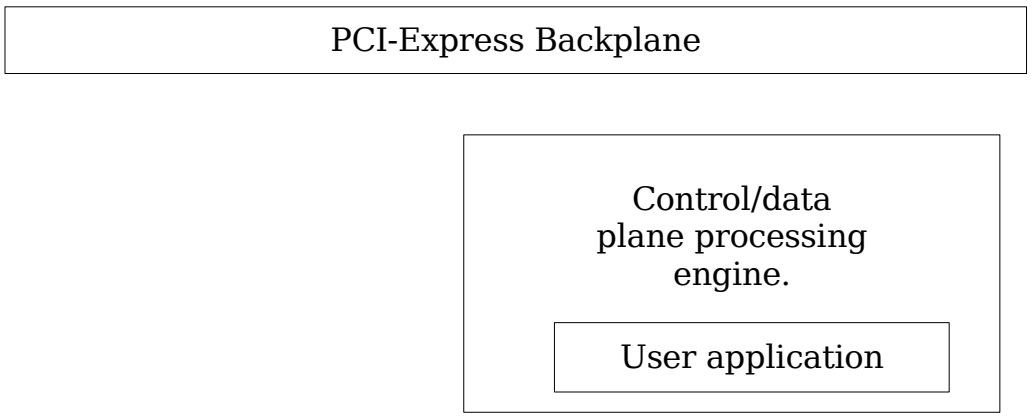
In the hardware architecture of the Netdevices proprietary platform, there are two complementary approaches to running user applications; the first is via a dedicated hardware card (an Application Support Engine) containing a separate Opteron CPU that is used to execute the application directly:



In this arrangement, an agent on the data plane engine will configured to redirect selected traffic to the ASE, so that the user application can process the packets. This arrangement can be thought of as a

dedicated host connected directly via an internal high speed network to the router. Typical applications would include DNS, Radius servers etc.

In the second arrangement, a network application can be directly executed on the data plane engine itself:



This avoids the use of a separate card, but requires more porting modifications to the application so that it integrates with the dedicated data plane engine.

Alternative Platform

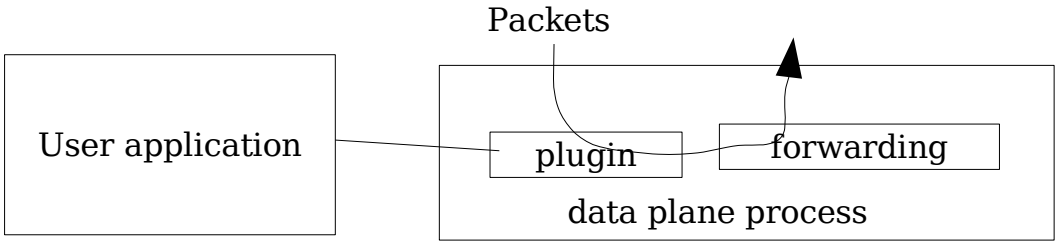
Aside from the proprietary hardware platform, it is possible to run the software on a standard, off-the-shelf PC platform using Linux. In this environment, it is possible to also run existing network applications without change (depending on the type of application), or with little modification.

Application Types

Different classes of applications may have different levels of integration, depending on the requirement of the application to connect to the data or control plane of the system.
To this end, it is possible to categorise applications in the following manner:

1. Standard networking aware applications. Applications that do not require visibility of separate network interfaces. Most networking applications fall into this category, such as ssh, telnet, http etc. It is expected that these applications would work unchanged by using the standard socket API.
2. Network applications that are required to be interface aware, or perhaps need to access packet based I/O independent from standard protocols (e.g DHCP etc.). For these applications, which often use extended socket option processing to gather extra context information, changes may be necessary to connect to the data plane of the system.
3. Router aware applications, where even if the packet I/O can be standardised sockets, the application needs to be interfacing with the router and it's associated components (e.g routing protocols etc). For these applications, it is expected that they would connect to the control plane IPC of the router to interact with the various control tables of the system.
4. Data plane aware, where applications require access or tapping of the packet stream through the system, but can achieve this via an external API (such as an API to tap a particular interface's packet stream). Packet sniffing applications and specialised monitoring applications may fall into this category.
5. Data plane inline, where the application is implementing some particular packet processing, and requires this processing to be part of the packet data plane path. For example, voice applications that required in-line codec processing of packets would greatly benefit from being able to perform the processing in-line in the data plane, rather than traversing the data through multiple kernel/user crossings.

A typical application that requires data-plane integration appears in the following diagram:



The data plane process allows loadable object modules to be installed. These plugins integrate closely with the forwarding packet path, providing a very optimal in-line path for packet processing.
The user application provides a small portion of code as the plugin (only code that needs to directly interact with the packet data need be in this plugin).
For other portions of the user application, a normal socket based API can be used for packets that are addressed to the router itself.

Benefits

The stated goal of the ONP is provide a framework for network applications within a router environment, rather than indirectly on a separate host. What are the benefits of running applications in this way?

- Performance, since applications can be partake in the data plane packet processing path directly, without having to recopy the packet, or have it undergo multiple kernel/user boundaries.
- Applications can reside directly on the network device, without having to exist on a separate device in the network.
- Applications can exert fine-grained control over the configuration and parameters of the device, dynamically modifying the network packet processing.
- Given the right APIs and framework, existing network applications can be readily ported.
- The data-plane processing for the application can be run at user level, with the subsequent benefits of flexibility, robustness, restartability etc.

Conclusion

There is much discussion in the Linux community concerning user level networking, or the need to migrate the complex network packet processing out from the kernel to user space, and to provide a new framework for higher performance network processing in Linux. The ONP framework in the Netdevices product is an example of what can be achieved if a suitable and efficient user level networking framework can be constructed. Given that change is in the air, consideration should be given to the NetIO framework developed by Netdevices as an example of this, and ONP as a framework for integrating network orientated applications.