Understanding Linux Kernel to Build Software Routers
(Qualitative Discussion)

Shiv Kalyanaraman, Arvind Venkatesan, Satish Raghunath
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma

Resources
- Linux Source Code: http://lxr.linux.no/source/
- Linux Router Project (poor documentation): http://www.linuxrouter.org/
- MIT Click Modular Router: http://www.pdos.lcs.mit.edu/click/
- ICIR XORP (extensible router) project: http://www.xorp.org/

Focus
- What is a Kernel?
- What is a Software router?
- How do I program the kernel to make Routers?
- Issues in kernel programming
- Better ways to build routers
  - Example: Click Router Toolkit

What is a ‘software router’?
- Router in a PC
- Program that receives, processes and forwards packets to the next node
  - Example: ‘routed’, this routing daemon is a simple router that forwards the packet to the next machine

What is a Kernel?
- Brain of an Operating System
- Software that mediates between your programs and the hardware
- Performs
  - Memory management
  - Process and resource management
  - Device management
  - File management

Do I have to program a kernel?
- Not if you want to play games 😊
- Change some aspects of kernel functionality - like process scheduling etc
- Research OS performance with different settings
- Try better and innovative algorithms
Navigating the Linux Source Code

- Go to: http://lxr.linux.no/source/
- Choose kernel version (default = latest)
- Go through specific directories (eg: net)
- Navigate through the search feature (demo)
- Please navigate the following files:
  - tcp_output.c, tcp_input.c, ip_forward.c
  - Make high level observations on the code and how it implements the TCP/IP functions

Reading Linux Source Code: Exercise

Navigate the following code segments and roughly state how the packet receive is handled at the lower layers and handled by IP.

**A Packet Receiver**

Pseudo Code

1. The receive interrupt
   - a) net/core/dev.c:netif_rx(skb)
   - b) include/linux/interrupt.h:__cpu_raise_softirq()
2. The network RX softirq
   - a) net/core/dev.c:net_init()
3. The IPv4 packet handler
   - a) net/ipv4/ip_input.c:ip_rcv()

Kernel v/s High level Programming

**Kernel level**
- Fine grained control
- Lacks high level programming interface
- Complex debugging and deployment
- Requires kernel know how

**High Level**
- Coarse control
- High level data structures and constructs
- Easy to build and debug
- Requires knowledge of the toolkit/software stack that hides OS internals

**Kernel Modules**
- Easier way to extend Linux kernel functions!
  - set of functions and data types that can be compiled as an independent program
  - It is then linked into the kernel when the module is installed
  - Linux modules may be loaded in a static or dynamic (i.e. during runtime) form

Kernel Module vs. Application

**PROs**
- higher flexibility
- passing parameters
- easier development
- less memory consumption
- direct use of periphery
- less context switching

**CONS**
- overhead
- kernel stability / security
- limited to kernel functions
- kernel namespace
- size matters

Modules Requirements & Examples

- modules should have:
  - few external symbols (functions, variables)
  - register / unregister functionality
  - filesystems (ext2, vfat, ntfs, ...)
  - block devices (hard drives, floppy)
  - character devices (soundcard)
  - protocols (smb, ipx)

Click Router Toolkit - An alternative approach

- Modular software router
- The whole Click router itself is a Linux kernel module
- Click itself is modular!
  - i.e. we can remove/add components of Click (eg: queues etc)
  - interconnected collection of modules called elements
  - Elements are written in C++!!
- It is also extensible – we can prototype our own designs!
- Provides high level coding interface:
  - Eg: vectors, string buffers, ip address, queues etc
- Router configurations are written using a simple language (next slide)
Example: Intercepting Packets (for custom processing) with Click Toolkit

- FromDevice(eth0) -> Queue() -> ToDevice(eth1):
  1. FromDevice(eth0): Get a packet from eth0 interface
  2. Queue(): queue the packet
  3. ToDevice(eth1): Send the packet to eth1 interface

Looks like Object Oriented Programming!

The click/router toolkit provides a number of prewritten routing elements like Queues, Shapers, packet receptors/senders etc that can be used off the shelf to build new routers!

You can write your own elements! Or modify/customize existing elements.

Check out: http://www.pdos.lcs.mit.edu/click

Features of Click

- Provides a high level programming interface to the developer
- Hides Kernel Intricacies
- Easily installed and configured
- Provides a number of router building blocks that can be used/modified to build custom router configurations

Summary

- Kernel level coding is necessary for system level implementation
- Kernel Programming is fraught with complications and dangers
- High level Toolkits (like Click) provide a simpler interface for the developers to implement kernel level tasks