



Current Topics in Linux Networking

Stephen Hemminger
<shemminger@osdl.org>

Senior Staff Engineer
OSDL Japan Linux Symposium
13-14 June 2006

Current Topics in Linux Networking

- Focus – what I know:
 - Network drivers
 - Ethernet Bridging
 - TCP congestion control
- Futures



Network Drivers API

- Network device driver API is complex
 - don't have to implement all of it
 - no driver gets it right the first time
- Driver's carefully reviewed
 - Long term support has to be the community
- Some areas undergoing construction
 - wireless
 - Faster devices will likely cause API to change



Network drivers

- About 350 network drivers in kernel
 - Sometimes we need to fix all of them
 - This is a good thing!
- Out of mainline kernel drivers are ignored
- Conversion and cleanup for sysfs took over 2 man years



Marvell/SysKonnect Gigabit Ethernet

- Existing driver was “vendor maintained”
 - Later versions not accepted by mainline
 - not incremental
 - regression missing previous fixes
 - Many unused features
 - private ioctl's and MIB
 - private failover
 - remote monitoring
 - Not the same style as other drivers



Skge, sky2: New drivers

- Smaller (20% of original)
 - better performance
 - more complete support of standard API's
- Wide community involvement
 - testing exposed many problems in hardware and old driver
 - Now have support from Intel and vendor



Ethernet Bridging

- Widely used for firewalling and virtualization
- Network locking changes
 - network in 2.6 uses RCU instead of brlock
- Same basic code since 2.4
 - Original author (student) did not have enough time
 - Adopted maintainer



Ethernet Bridging – new features

- Automatic speed and carrier detection
 - code reuse for VLAN, bonding, etc
- Variable MTU
 - still need netfilter module for fragmentation
- Sysfs API
 - necessitated by need to support mixed 32/64bit configuration
- Spanning Tree
 - add infrastructure to allow RSTP as application



TCP congestion control

- Changes for supporting high speed wide area networks
 - Large Bandwidth Delay Product
- Large academic research community
 - web100 and others
 - integrated started with 2.6.0



TCP congestion control

- Many variants proposed
- All changed same code locations
- Logic errors showing up during the code integration process



Step 1: TCP congestion control API

- Factor out hooks used by each congestion control
- Provide API and infrastructure support
- Add socket options
- Object oriented style design
- Change existing CC to match API



Step 2: More TCP

- Availability of API led to:
 - more researchers
 - congestion control choices
 - cleaner code
 - more review
- NS-2 support for Linux TCP CC API
 - allows validation in known scenario



Step 3: Testing tools

- Emulation – add delay loss to an existing interface
 - Existing emulation tool:
 - NIST net
 - Dummynet (FreeBSD)
 - New network emulator using QOS scheduling
 - Part of standard kernel
 - Allows building complex scenarios



Step 4: Pickup the pieces

- Some changes broke things
 - Example:Java debugger
 - lots of small writes
 - now fixed
 - Configuration options exist to force old behavior
- Need to improve testing
 - Small errors still getting through
 - Hard to test
 - Researchers don't agree



Conclusion: TCP

- Open Source works!
 - Most of the work was done by others:
 - researchers
 - developer community
- Linux is years ahead of any other OS
 - base platform for research



Near term changes

- Many 10 Gigabit drivers under review
- Better wireless stack
 - full softmac
 - bridging (WDS) needed
- Virtualization
 - Network namespace virtualization
 - Xen adoption



Long term speculation

- Changes for higher performance
 - net channels – fix cache intensive data structures
 - socket API limits DMA
 - AIO API is non-standard and complex
 - RDMA is too specialized
 - TCP/IP in user space??
 - incoming packet processing too slow
 - real world 100's of netfilter rules
 - flow cache scale issues
 - unified flow cache?



Unlikely to be adopted

- TCP Offload Engine
 - intrusive
 - loss of control of firewalling and TCP behavior
 - performance benefit unsure
- Open RDMA
 - vendor driven kernel API
 - overlap with Infiniband

