



Advances in Solaris Networking

Darren Reed

Staff Engineer (Kernel Networking)

Sun Microsystems

Solaris Networking

- IPFilter
- Packet Filtering Hooks
- Stack Instances
- Crossbow
- NEMO

IPFilter

- History
- Platforms
- Design
- Use
- Solaris IPFilter

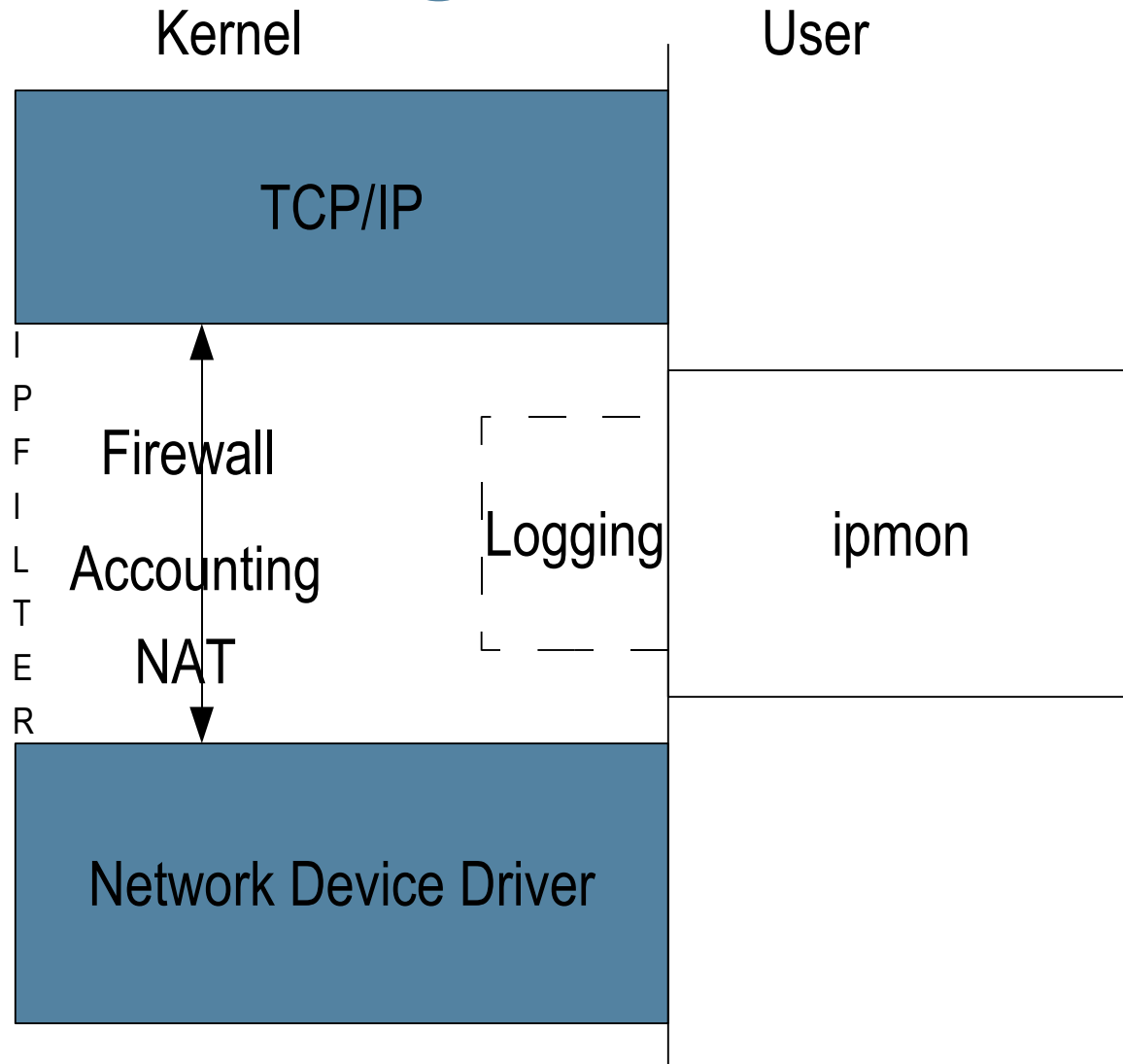
IPFilter History

- Inception, circa 1993
- Logging - 1994
- Stateful Filtering - 1995
- NAT - 1996
 - > Proxies
- STREAMS module for Solaris – 2000
 - > Integrated into Solaris 10 – December 2003
- Address pools - 2002

IPFilter platforms

- AIX 5.3 ML05
- BSDs
 - > FreeBSD
 - > NetBSD
 - > OpenBSD
 - > others...
- HP-UX 11
- IRIX 6.2, 6.5
- Linux Kernels 2.4, 2.6
 - > SuSE, RedHat, others...
- QNX 6
- Solaris
- SunOS 4.1
- Tru64 5.1b

IPFilter design



IPFilter use

- Managing filter rules
 - > ipf
 - Stateful filtering
- Managing NAT
 - > ipnat
 - Transparent proxies, internal kernel proxies
- Logging
 - > ipmon
 - From kernel buffer to syslog or file
- Managing address pools
 - > ippool

IPFilter on Solaris

- IPFilter prior to 4.1 (3.x)
 - > Interposed itself below IP by modifying pointers
- IPFilter 4.1.0 and beyond
 - > Uses STREAMS module, pfil
- OpenSolaris, build 52 and beyond
 - > Packet Filtering Hooks API

Solaris Networking

- IPFilter
- Packet Filtering Hooks
- Stack Instances
- Crossbow
- NEMO

Packet Filtering Hooks

- STREAMS approach
- Performance issues
- Interaction with zones
- Project Goals
- Filtering Points
- API
- Future work

Packet Filtering - STREAMS

- Arrange for STREAMS module to be between driver and IP
 - > autopush
 - > Ifconfig modinsert
- Can *snoop* on conversation between driver and IP
- Requires a large amount of support code
- No supported method to access IP internal data

Packet Filtering Performance

- STREAMS module Prevents IP from enabling new driver-IP optimisations
 - > Total cost on end host – 20%-30%
- Elimination of the STREAMS module resulted in greater than 90% of the performance bump from IPFilter being eliminated.

Packet Filtering Hooks – Goals

- Support packet filtering between zones
- Eliminate the performance problems with STREAMS modules
- Provide an interface to networking functionality/data supported by network protocols

Packet Filtering and Zones

- Hooks inserted into IP codepaths
- Intercepts *all* loopback traffic
- Packets associated with physical network interface
- Inter-zone filtering currently relies on addresses
 - > Cannot specify/match on zone names
- Filtering is all handled by the global zone

Packet Filtering – Interception Points

- Currently present in OpenSolaris:
 - > Physical Out
 - > Physical In
 - > Loopback Out
 - > Loopback In
 - > Forwarding

Packet Filtering - API

- Access to hooks is provided through kernel networking API
 - > Private, unstable, interface – ie not ready for people to use in applications but ready for hacking on :-)
- Provide access to networking data structures and functions
 - > Getting interface/address information
 - > Callbacks for notification of change

Packet Filtering – Future work

- Allowing multiple callouts to modify data
- Support filtering at other layers
- Add new intercept points for IP:
 - > Local Out
 - > Local In
 - > Others?
- OpenSolaris Community
 - > networking-discuss@opensolaris.org

Solaris Networking

- IPFilter
- Packet Filtering Hooks
- **Stack Instances**
- Crossbow
- NEMO

IP Instances

- Project Goals
- Zones Networking
- Shared vs Exclusive stacks
- Delegation of control

IP Instances – Project Goals

- Provide a more robust architecture for zones networking
 - > Provides per-zone routing tables
 - > Provides per-zone ARP table
 - > i.e. provide an instance of IP per-zone

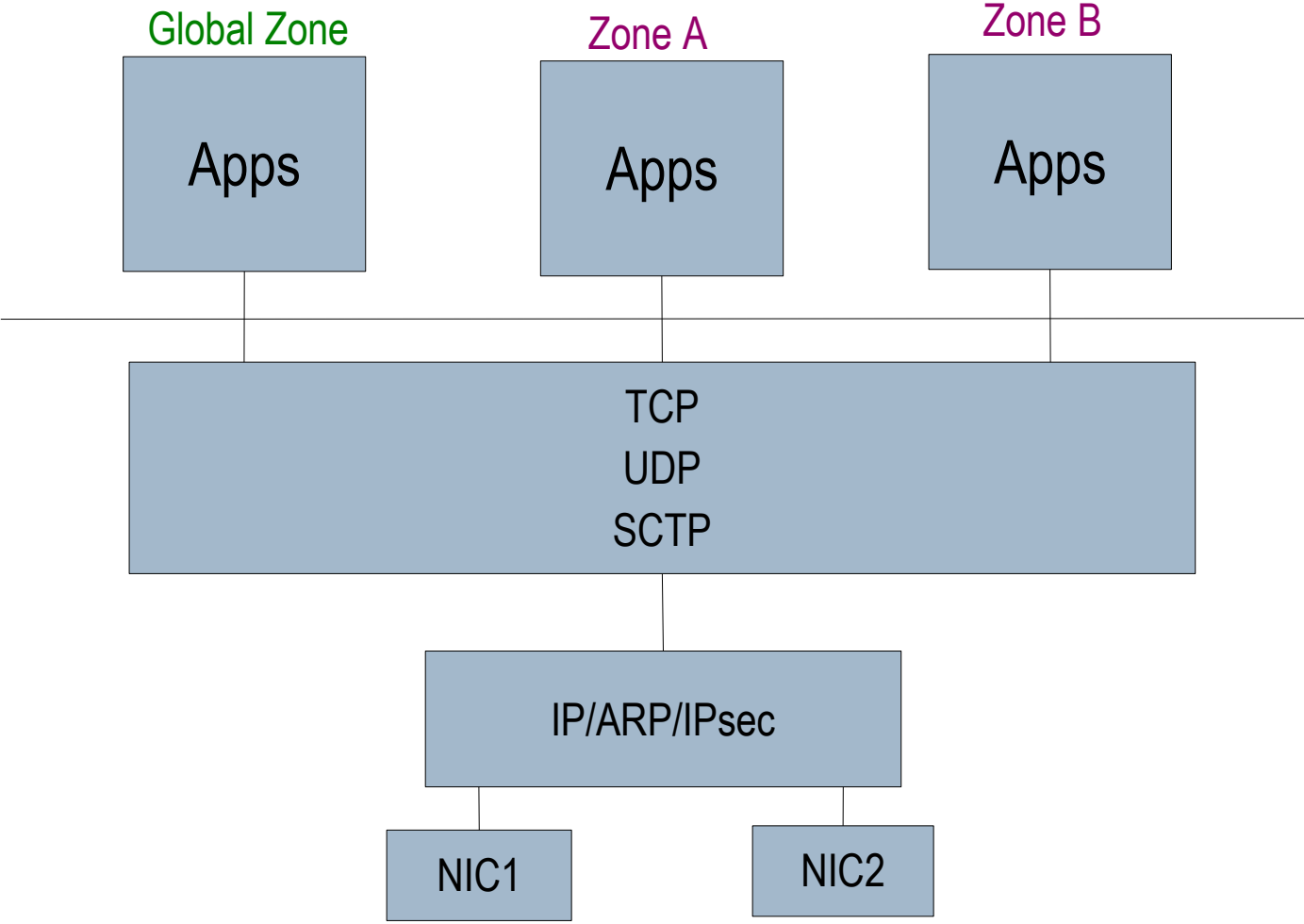
Aside – Zones/containers introduction

- Virtualisation Technology
- Provides
 - > Seperate process space
 - Inter-zone communication must be via TCP/IP
 - > Seperate user space
 - > Confined (i.e. chroot'd) disk space
- Global Zone and Local zones
 - > Visibility of the system
- Zone administration
 - > Subset of tasks required for Global Zone

Zones Networking

- Restricted visibility
 - > Routing table = associated with network interface(s)
- Administered from the global zone
 - > Cannot use DHCP
- Has its own socket port number space
 - > Can listen on “0.0.0.0 port 80” in every zone
- Limited firewall capabilities

Zones Networking – Today



Shared vs Exclusive Instances

- Shared stack
 - > Uses global routing table
 - > Global interface management, performance tuning, etc
 - > Short-cut routing
 - > Packet filtering applied to the entire machine
- Exclusive Instances
 - > Routing table per zone
 - > Each zone can tweak TCP, etc, settings in its own way
 - > Each zone decides what filtering it wants

Networking – Delegation of control

- An exclusive instance cannot be managed from the global zone
- Local zone root has full control over IP
 - > Use of ndd to tune IP is allowed and is private
- Network interfaces delegated for exclusive use are not visible in the global zone

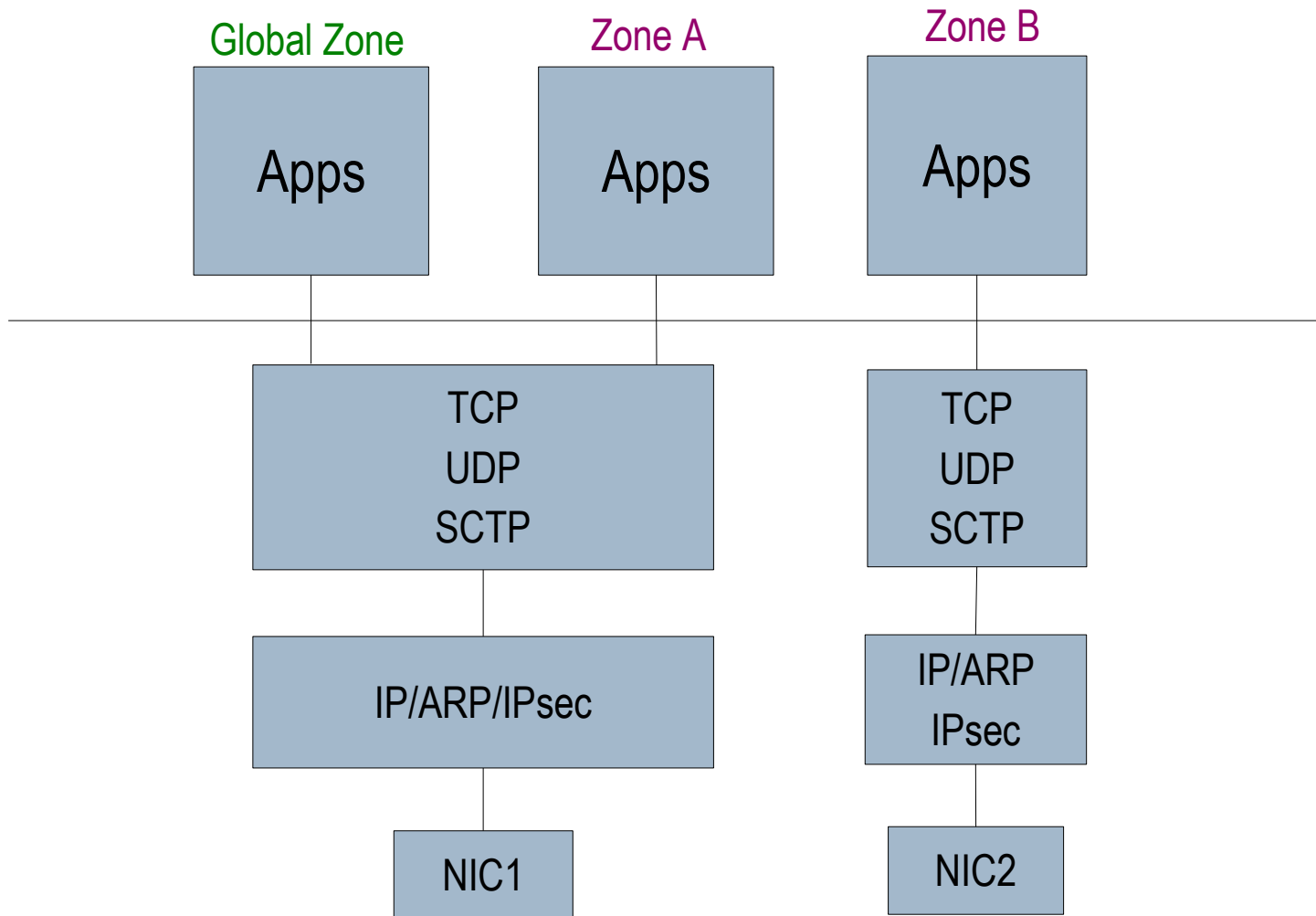
IP Instances – Changes to security

- Privilege split of SYS_NET_CONFIG
 - > SYS_IP_CONFIG (new) for zones
- Can snoop from inside a zone
 - > *But can still snoop using the interface in global too!*
- Security Threat
 - > Zone can forge ethernet packets

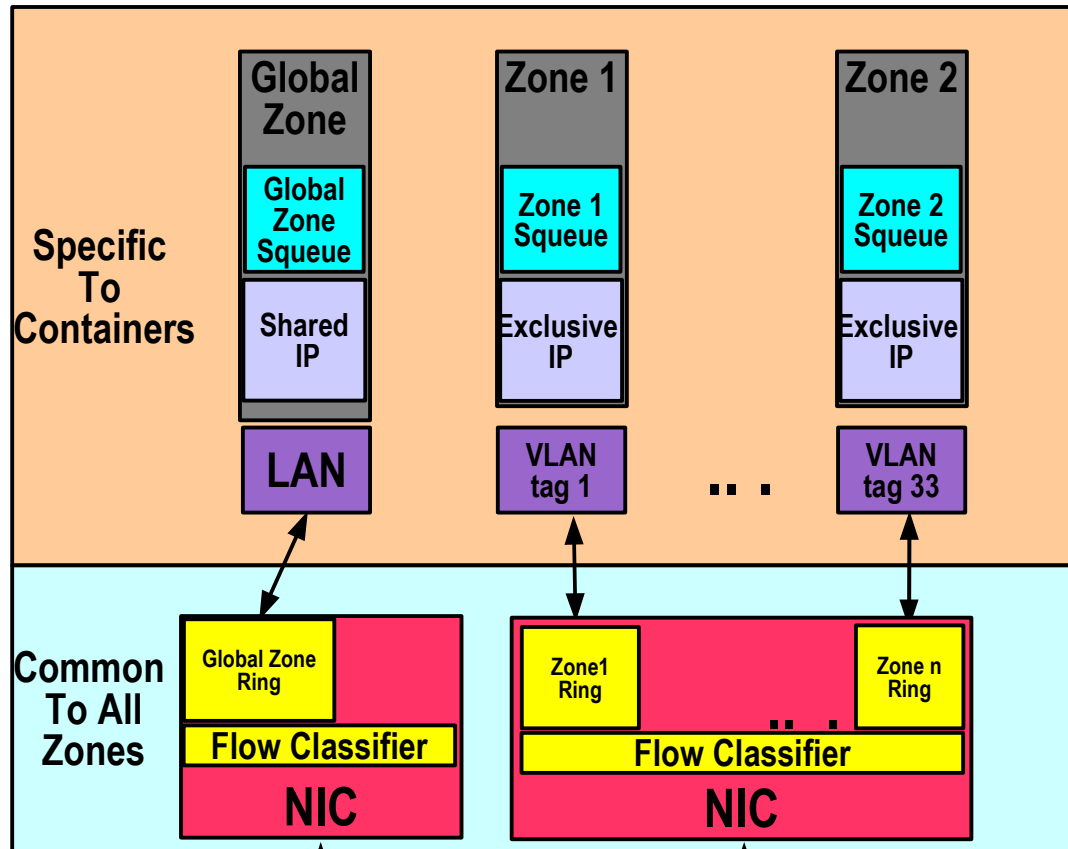
IP Instances – Implementation

- No more global data in IP
- Each instance is tied to **one** zone
 - > Cannot have more than one instance per zone
 - > Cannot have an instance without a zone
 - > Are enabled with zonecfg when creating a zone
 - set ip-type=exclusive
- On OpenSolaris as part of the Crossbow project

Zones Networking – IP Instances



IP Instances – Separation of VLANs



Management network - bge0

Data network – bge1 with VLANs bge10001 and bge33001

Solaris Networking

- IPFilter
- Packet Filtering Hooks
- Stack Instances
- **Crossbow**
- NEMO

Crossbow

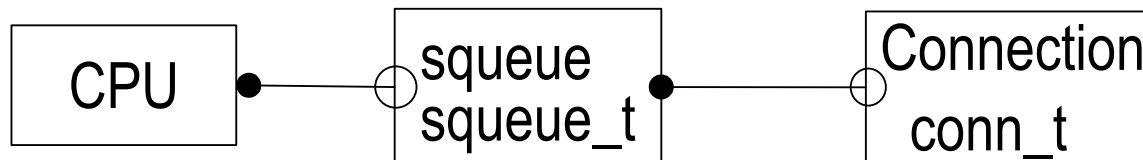
- Background - Fire Engine (S10) and queues
- Crossbow Project
 - > Bandwidth Control
 - > VNICs

Fire Engine – Solaris 10

- Goal: Make Solaris's TCP/IP stack perform competitively
- Merge IP and TCP
 - > Some code refactoring
 - > Use of /dev/ip and /dev/tcp preserved
 - > Reduce context switching, increase cache hits
- Implementation of queues
 - > *Vertical Perimeter for Solaris networking*

Fire Engine – queues

- queue = serialisation queue (FIFO)
- Per-CPU for all inbound/outbound packets
- Provides mutual exclusion to TCP without locks
 - > 1 thread at a time works on an queue
- Packet keeps the same CPU going:
 - > From top to bottom
 - > From bottom to top



Crossbow

“Crossbow provides the building blocks for network virtualization and resource control by creating virtual stacks around any service (HTTP, HTTPS, FTP, NFS, etc.), protocol (TCP, UDP, SCTP, etc.), or Virtual machines like Containers, Xen and Idoms.” - *Sunay Tripathi*

- Bandwidth Control
- Virtual Network Interfaces

Crossbow Bandwidth Control

- Goal: manage and share bandwidth on a NIC
 - > Reserve x Mbs/second
 - > Quota of x Mb/s
- Packet dropping
 - > RED (Random Early Detection)
- Traffic Shaping
 - > TBF (Token Bucket Flow)
- Queueing
 - > CBQ (Class Based Queuing), WFQ (Weighted Fair Queuing), DRR (Differential Round Robin)

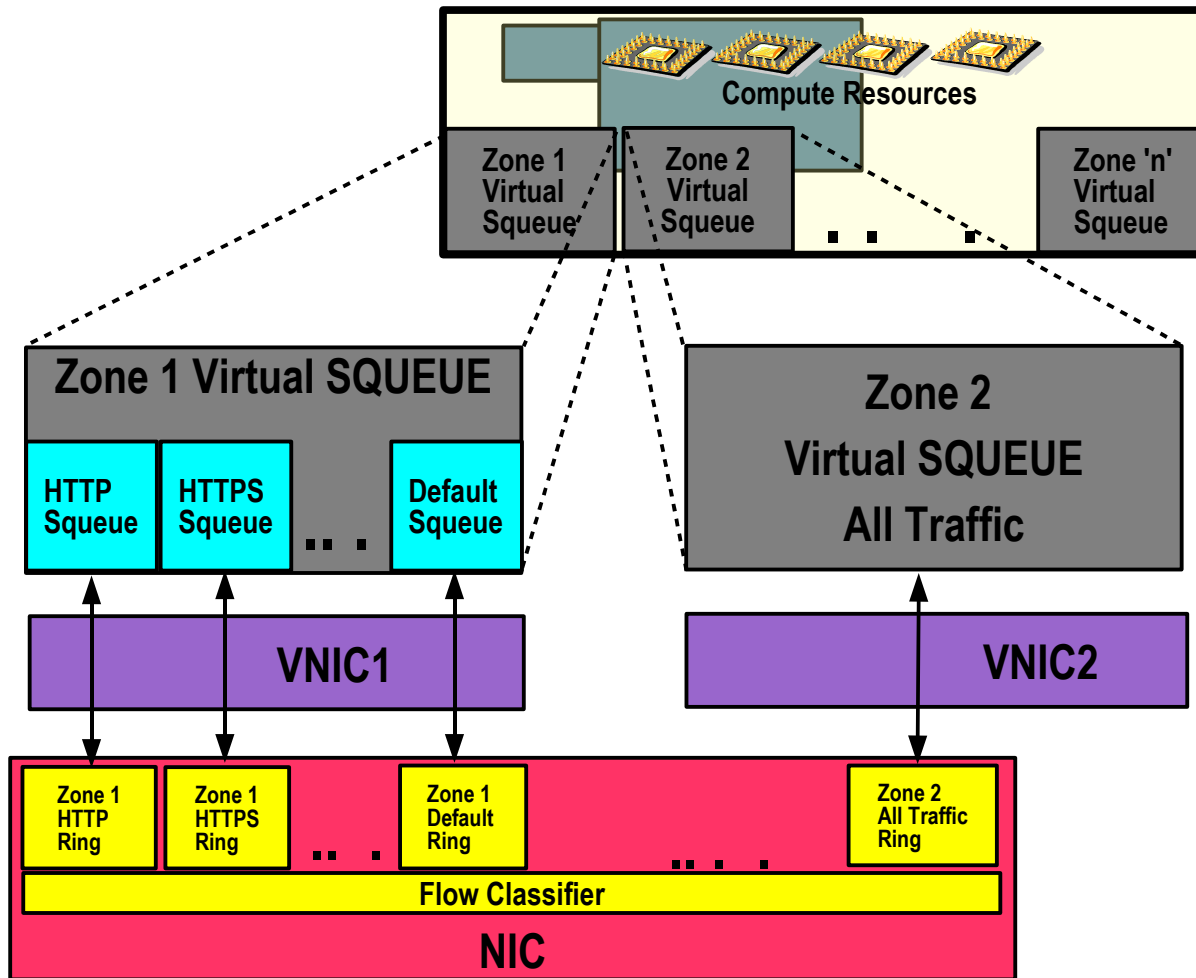
Crossbow – Virtual Interfaces

- Has its own virtual resources
 - > Tx/Rx descriptor rings
 - > DMA channels
 - > MAC addresses
 - > Kernel threads & queues
- Can be defined above any NEMO networking device
 - > `dladm show-link`
 - e.g aggr (VLAN), bge, etc.

Crossbow – VNIC details

- Carve up 1Gb/s and 10Gb/s hardware NIC into multiple virtual NICs
- Implemented as a Nemo/GLDv3 MAC driver.
- Assign NIC hardware resources (interrupts, rings, etc) to virtual NICs
- Rely on hardware-based flow classification to steer traffic to VNICs and maximize performance
- Assign VNICs to Zones or Xen domains

CrossBow Virtual NICs Example



Crossbow – Development community

- OpenSolaris Community
 - > crossbow-discuss@opensolaris.org
- Project within the networking community
 - > <http://www.opensolaris.org/os/project/crossbow/>
- Further ideas:
 - > Virtual switch

Solaris Networking

- IPFilter
- Packet Filtering Hooks
- Stack Instances
- Crossbow
- **NEMO**

NEMO – Network Driver Interface

- GLDv3 – Network device driver framework
- Interrupt handling
- Trunking

Nemo – GLDv3

- GLD – Generic LAN Driver
 - > Evolving interface:
 - Version 1 in Solaris 7
 - Version 2 in Solaris 9
 - > Current drivers: bge, nge, xge, rge, ixgb, e1000g
- High performance framework that IP can use to control all activities/resources without burdening the device driver writer.
- Work in progress to make the API robust and “*future proof*”

Nemo – GLDv3 Framework

- Defines interfaces provided by Solaris
- Defines the interface a driver must supply
- MAC-type plug-in framework
 - > Ethernet, IP tunnel, WiFi, Infiniband

Nemo – Interrupt Moderation

- Networking interrupts are bad because writers get pinned, context switches, etc.
- Bind a NIC to a queue and let the queue own the NIC and have the ability to turn off interrupts
- If queue becomes backlogged, it turns the NIC interrupts off
- More CPU is available to process backlog
- The packets that were received by the NIC while interrupts were disabled are sent up as a chain as soon as interrupts resume.

Nemo: Interrupt Moderation (cont.)

- Expect another 20% improvement in web workloads
- Sample mpstat output:

> *Mpstat (older driver)*

intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
10818	8607	4558	1547	161	1797	289	19112	17	69	0	12

> *Mpstat (GLDv3 based driver)*

intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
2823	1489	875	151	93	261	1	19112	17	57	0	27

- Notice the decrease in interrupts, context switches, mutex contentions, etc. and the increase in idle time.

Nemo: Trunking

- Create trunks (link aggregations) of 1Gb NICs or 10GB NICs
- IEEE 802.3ad compliant, including LACP (Link Aggregation Protocol)
- Each member of the trunk is owned by an individual queues that controls the packet arrival rate
- No specific support needed from device driver
- Near linear scalability for a trunk of 4 1Gb NICs



Darren Reed
Staff Engineer (Kernel Networking)
Sun Microsystems