# The Challenges of Dynamic Network Interfaces

by Brooks Davis

brooks@{aero,FreeBSD}.org

The Aerospace Corporation

The FreeBSD Project

EuroBSDCon 2004

October 29-31, 2004

Karlsruhe, Germany

## Introduction

- History of Dynamic Interfaces
- Problems
- Possible Solutions
- Advice to Implementors
- Future Work

## Early UNIX

- Totally static.
- All devices must be compiled in to kernel
- Fast and easy to program
- Difficult to maintain as the number of devices grows

## Autoconfiguration

- Introduced in 4.1BSD (June 1981)
- One kernel can serve multiple hardware configurations
- Probe
  - Test for existence of devices, either using stored addresses or matching devices on self-identifying buses
- Attach
  - Allocate a driver instance (as of 6.0, this must be fully dynamic)

# Kernel Modules

- Allows drivers to be added at run time
- LKM (Loadable Kernel Modules)
  - Introduced in 2.0 by Terry Lambert
  - Modeled after the facility in SunOS
- KLD (dynamic kernel linker)
  - Introduced along with newbus in 3.0 by Doug Rabson
  - Added a generic `if_detach()` function

## PC Card & CardBus

- Initial PC Card (PCMCIA) support via PAO in 2.0
- Fairly good support in 3.0
- Most PAO changes merged in 4.0
  - PAO development ceased
- CardBus support in 5.0

## Other Removable Devices

- USB Ethernet (4.0)
- Firewire (fwe(4) in 4.8, fwip(4) in 5.3)
- Bluetooth (5.2)
- Hot plug PCI
- Compact PCI
- PCI Express
- Express Card

## Netgraph

- Node implement network functions
- Arbitrary connection of nodes allowed
- ng_iface(4) node creates interfaces on demand

## Interface Cloning

- Handles most pseudo interface creation
  - `ifconfig vlan create`
- Imported from NetBSD in 4.4
- Extended to support more complex names in 5.3
  - `ifconfig fxp0.10 create`
    - Creates vlan(4) interface handling tag 10 on fxp0

## Interface Renaming

- Introduced in 5.2
- Allows interfaces to be given logical names
- Aids hardware independence
  - devd(8) should eventually support things like setting interface names based on slot number

## Problems

- Userland
  - Network management systems
  - Network monitoring applications
- Kernel
  - Stale references
  - Hardware races

# Not Detecting Arrival and Departure

- Many (most?) applications do not detect arrival or departure of interfaces
- Only interfaces that were attached at startup are shown
- Mostly Harmless

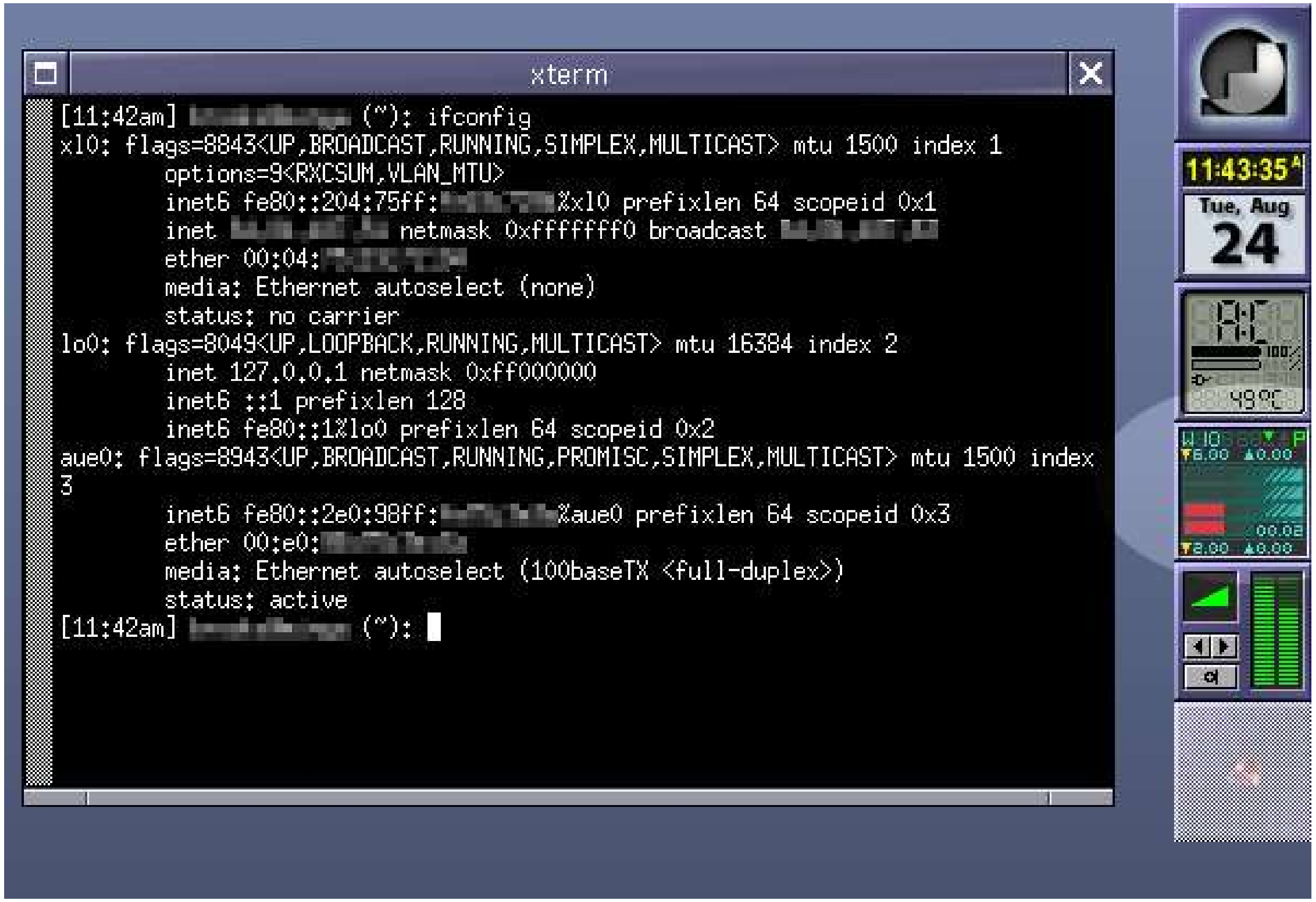
## Solution: Not Detecting Arrival and Departure

- Detect Arrival and Departure
  - Periodic rescan
    - Beware races
  - Routing socket
    - Renaming looks like departure plus arrival
  - /dev/net
    - Use kqueue(2) or dir(3)

## Inconsistent Interfaces Indexes

- The index is the stable handle to an interface
- Indexes are reused and are allocated in a non-sparse manner
- Interfaces come and go in arbitrary order
- Indexes are only good for the life of the interface instance

# Inconsistent Interfaces Indexes



```
[11:42am] [redacted] (~): ifconfig
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500 index 1
        options=9<RXCSUM,VLAN_MTU>
        inet6 fe80::204:75ff:[redacted]%xl0 prefixlen 64 scopeid 0x1
        inet [redacted] netmask 0xfffffff0 broadcast [redacted]
        ether 00:04:[redacted]
        media: Ethernet autoselect (none)
        status: no carrier
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384 index 2
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x2
aue0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500 index
3
        inet6 fe80::2e0:98ff:[redacted]%aue0 prefixlen 64 scopeid 0x3
        ether 00:e0:[redacted]
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
[11:42am] [redacted] (~): 
```

## Solutions: Inconsistent Interfaces Indexes

- Out of band notification (routing socket or kqueue(2))
- Check the interface epoch
  - `struct if_data` member, `ifi_epoch` is a `time_t` containing the time the interface was created (or its statistics were reset)
  - Will not work in the case of sub-second create-destroy-create cycles (insufficient space to use higher resolution time)

## SNMP `ifIndex` vs `if_index`

- RFC2233:
  - A previously-unused value of `ifIndex` must be assigned to a dynamically added interface if an agent has no knowledge of whether the interface is the “same” or “different” to a previously incarnated interface

## SNMP `ifIndex` vs `if_index`

- Removing and replacing an interface will work since the same index will generally be allocated
- Hacks in place to maintain indexes across driver reloads
- Does not meet this requirement in general
  - What is the “same” interface on a tunnel server?

# Solutions: SNMP `ifIndex` vs `if_index`

- Handle `ifIndex` values in the agent
  - May require application specific daemons or hooks to allow external management
- Allow applications to manage `if_index` assignments
  - Easy to implement in kernel
  - Very wasteful of memory
  - Small index space ($2^{15}$-1)

## SNMP Counters

- RFC2233:
  - A management station, not noticing that an interface has gone away and another has come into existence, must not be confused when calculating the difference between the counter values retrieved on successive polls of a particular `ifIndex` value

## Solutions: SNMP Counters

- Detect interface departure and arrival and produce synthetic counters
  - Inaccurate since interfaces do not have a zombie state
- Set the ifCounterDiscontinuityTime variable defined in RFC2233
  - The new `ifi_epoch` member of `struct if_data` may be used

## Stale `struct ifnet` Pointers

- `struct ifnet` is the device independent part of the interface
- In-flight packets hold references to it in `struct mbuf`
- In most cases, the packets are drained from queues before the `struct ifnet` is destroyed
- Dummynet prevents this by holding the packets elsewhere

## Solution: Stale `struct ifnet` Pointers

- Refcount `struct ifnet`
  - Expensive
  - Since the hardware may really be gone, delay in removing the struct ifnet should be avoided
- Do not refer to `struct ifnet` via a pointer in long-lived references
  - Use the index and use `ifnet_byindex()`
  - Return special `dead_if` when gone

## Advice to Implementers

- Remember that interfaces are dynamic
  - In general, assume devices will be dynamic
- Do not assume interface names are stable handles to interfaces
  - Use indexes instead
- Avoid kvm(3)

## Conclusions

- Dynamic interfaces are here to stay
  - More and more buses support hot swapping
- SNMP agent enhancements are needed
- Solutions to stale struct ifnet references are needed

## Questions?

http://people.FreeBSD.org/~brooks/pubs/eurobsdcon2004/