

## Re: Historical question

**Source:** <http://unix.derkeiler.com/Newsgroups/comp.unix.bsd.freebsd.misc/2005-06/0846.html>

---

**From:** Robert Bonomi ([bonomi\\_at\\_host122.r-bonomi.com](mailto:bonomi_at_host122.r-bonomi.com))

**Date:** 06/22/05

Date: Tue, 21 Jun 2005 23:39:53 -0000

In article <[slrndbfcgh.dja.mordor@fly.srk.fer.hr](mailto:slrndbfcgh.dja.mordor@fly.srk.fer.hr)>,

Zeljko Vrba <[mo.dor@fly.srk.fer.hr](mailto:mo.dor@fly.srk.fer.hr)> wrote:

>I'm wondering just out of curiosity.. Why was it decided that NICs don't have  
>an entry in /dev but to be configurable by ifconfig?

>

>And why don't all sockets (irrespective of domain) have some kind of a vnode  
>in a (synthetic) filesystem? It would be great for debugging sometimes just  
>to be able to inject data to socket by doing something like

>cat smth > /sockets/ipv4/... It doesn't even work for UNIX sockets which DO  
>have a file-system inode..

>

>Another use would be for sniffing traffic by doing cat on the socket. Without  
>such facility we need specialized tools like tcpdump.

>

>I'm not proposing an implementation. Just asking for the historical reason  
>for this unorthogonality.

"Because." \*grin\*

Get ahold of the book "The Design of the BSD 4.4 Operating System", if you want some insight into many of the design decisions.

I \*don't\* know, 'why', but I suspect that it had to do with the scale/complexity of things to be supported.

you have to have:

- 1) a destination address of some sort
- 2) a protocol specification, do determining 'content' formatting,
- 3) a message 'sub-type', and/or a 'port' address.

Over-and-above the 'gory details' of the physical interface layer.

contemplate the situation where a host may have multiple NICs, and you find it requires 5 elements to uniquely identify a 'connection' just in IP address-space.

- source IP address,
- source port,
- protocol,
- destination IP address,

destination port

IP addresses are 32 bits each. ports are 16 bits each, and there is 8 bits for 'protocol' identifier. This is 104 bits of 'identifier'.

Without considering 'what to do' if you want to speak something other than IP — e.g. DECnet, SNA, XNS, LU6.2, NetBios, etc.

(at the 'device driver' level, you've only got 'major ID', "minor ID" fields 'unit' and 'inode' to identify 'unique' items within the device. Postulating you use 'MajorID' to indicate addressing 'family' (IP, XNS, SNA, etc.) and 'minorID' for protocol family (e.g. for IP, UDP, TCP, ICMP, IGRP, etc.) you have to have at least 96 bits in the 'inode', to uniquely identify IP connection. For some of the other protocols, with larger address-spaces, it is even worse.

Enumerating that in a filesystem structure would consume *\*exorbitant\** amounts of resources.

Also, by standard implementation, filesystem resources are 'shared access' multiple users can read/write to the same file simultaneously.

'Sockets', on the other hand, are designed/implemented as 'exclusive access'.

Then there is the 'minor' question of what any of the standard filesystem operations would "mean", when applied to a 'socket'-type connection. What does a "rename" do? What happens if you "move" it to a different place in the file-system hierarchy? What does a "delete" do? If you 'delete' the socket from the filesystem, while you still have it open, can somebody else create a socket with the same name — and if so, do they get access to the same resource, or a different one?

The theory looks nice. implementation gets *\*very\** messy. <wry grin>