In order to capture packets (with Wireshark/TShark, tcpdump, or any other libpcap-based packet capture program) on a Linux system, the "packet" protocol must be supported by your kernel. If it is not, you may get error messages such as

> modprobe: can't locate module net-pf-17

in "/var/adm/messages", or may get messages such as

> socket: Address family not supported by protocol

from applications using libpcap.

Most recent Linux distributions will have this configured in by default. If it is not configured in with the default kernel, and if it is not a module loaded by default, you must configure the kernel with the CONFIG_PACKET option for this protocol; the following note is from the Linux "Configure.help" file for the 2.0[.x] kernel:

> Packet socket
> CONFIG_PACKET
> The Packet protocol is used by applications which communicate directly with network devices without an intermediate network protocol implemented in the kernel, e.g. tcpdump. If you want them to work, choose Y.
>
> This driver is also available as a module called af_packet.o (= code which can be inserted in and removed from the running kernel whenever you want). If you want to compile it as a module, say M here and read Documentation/modules.txt; if you use modprobe or kmod, you may also want to add "alias net-pf-17 af_packet" to /etc/modules.conf.

and the note for the 2.2[.x] kernel says:

> Packet socket
> CONFIG_PACKET
> The Packet protocol is used by applications which communicate directly with network devices without an intermediate network protocol implemented in the kernel, e.g. tcpdump. If you want them to work, choose Y. This driver is also available as a module called af_packet.o (= code which can be inserted in and removed from the running kernel whenever you want). If you want to compile it as a module, say M here and read Documentation/modules.txt. You will need to add 'alias net-pf-17 af_packet' to your /etc/conf.modules file for the module version to function automatically. If unsure, say Y.

In addition, there is an option that, in 2.2 and later kernels, will allow packet capture filters specified to programs such as tcpdump to be executed in the kernel, so that packets that don't pass the filter won't be copied from the kernel to the program, rather than having all packets copied to the program and libpcap doing the filtering in user mode.

Copying packets from the kernel to the program consumes a significant amount of CPU, so filtering in the kernel can reduce the overhead of capturing packets if a filter has been specified that discards a

significant number of packets. (If no filter is specified, it makes no difference whether the filtering isn't performed in the kernel or isn't performed in user mode. :-))

Most recent Linux distributions will have this configured in by default. If it is not configured in with the default kernel, you must configure the kernel with the CONFIG_FILTER option; the "Configure.help" file says:

> Socket filtering
> CONFIG_FILTER
> The Linux Socket Filter is derived from the Berkeley Packet Filter. If you say Y here, user-space programs can attach a filter to any socket and thereby tell the kernel that it should allow or disallow certain types of data to get through the socket. Linux Socket Filtering works on all socket types except TCP for now. See the text file linux/Documentation/networking/filter.txt for more information. If unsure, say N.

An additional problem, on Linux, with older versions of libpcap, is that capture filters do not work when snooping loopback devices; if you're capturing on a Linux loopback device, do not use a capture filter, as it will probably reject most if not all packets, including the packets it's intended to accept - instead, capture all packets and use a display filter to select the packets you want to see. Most recent Linux distribution releases will not have this problem.

In addition, older versions of libpcap will, on Linux systems with a 2.0[.x] kernel, or if built for systems with a 2.0[.x] kernel, not turn promiscuous mode off on a network device until the program using promiscuous mode exits, so if you start a capture with Wireshark on some Linux distributions, the network interface will be put in promiscuous mode and will remain in promiscuous mode until Wireshark exits. There might be additional libpcap bugs that cause it not to be turned off even when Wireshark exits; if your network is busy, this could cause the Linux networking stack to do a lot more work discarding packets not intended for the machine, so you may want to check, after running Wireshark, whether any network interfaces are in promiscuous mode (the output of "ifconfig -a" will say something such as

```
eth0      Link encap:Ethernet  HWaddr 00:00:66:66:66:66
          inet addr:66.66.66.66  Bcast:66.66.66.255  Mask:255.255.255.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:6493 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3380 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:18 Base address:0xfc80
```

with "PROMISC" indicating that the interface is in promiscuous mode), and, if any interfaces are in promiscuous mode and no capture is being done on that interface, turn promiscuous mode off by hand with

```
ifconfig <ifname> -promisc
```

where "<ifname>" is the name of the interface.

Newer versions of libpcap shouldn't have this problem, even on 2.0[.x]

kernels; no version of libpcap should have that problem on systems with 2.2 or later kernels.