Open_pcap: The function open_pcap is used to open a live capture on a network interface using the libpcap library.

Working:
- It initializes a variable *on* to 1. This variable is used later to set the BIOCIMMEDIATE option if the operating system supports it.
- If the debug option is enabled (checked by *opt_debug*), it prints a debug message showing the parameters that will be passed to *pcap_open_live*.
- It calls pcap_open_live to open a live capture on the network interface specified by ifname. The parameters to pcap_open_live specify a snapshot length of 99999 bytes, promiscuous mode off (0), a read timeout of 1 millisecond, and a pointer to an error buffer.

char errbuf[PCAP_ERRBUF_SIZE];
pcap_t *pcap_open_live(const char *device, int snaplen, int promisc, int to_ms, char *errbuf);

where snaplen: snapshot length -> length in bytes to be captured.
promisc: non-zero will make it run in promiscuous mode.
to_ms: packet buffer timeout: read more:
errbuf: If NULL is returned by the call, errbuf is filled in with an appropriate error message. errbuf may also be set to warning text when pcap_open_live() succeeds; we can display the warning/error message using this.

ifname, errbuf are global variables.

-If pcap_open_live fails (returns NULL), it prints an error message and returns -1.

-If the operating system is not Linux or Solaris (checked by #if (!defined OSTYPE_LINUX) && (!defined __sun__)), it sets the BIOCIMMEDIATE option on the pcap descriptor. If this ioctl call fails, it prints an error message.
Pcap descriptor using -> pcap_fileno
BIOCIMMEDIATE: Enables or disables "immediate mode", based on the truth value of the third argument. When immediate mode is enabled, reads return immediately upon packet reception. Otherwise, a read will block until either the kernel buffer becomes full or a timeout occurs.
For this the third argument of the ioctl must be of type u_int. And we choose a int on = 1 for this and pass its address.

close_pcap: uses pcap_close call to close the pcap file.

pcap_recv: This function reads a packet from the pcap capture into the buffer specified by packet. It specifies a char pointer p to hold the packet data and runs a while loop until p is not null. It uses the libpcap function pcap_next to get the next packet from the capture.
- If the packet is smaller than the size of the buffer, it adjusts the size to match the packet size. It then copies the packet data into the packet char pointer (packet)[buffer] and returns the size of

the packet.

const u_char *pcap_next(pcap_t *pcapfp, struct pcap_pkthdr *hdr);
- both pcapfp and hdr are globally defined.