3.2 libnet

Libnet is an API that provides high-level interface to low-level network packet handling. Libnet was written in the C programming language, and consists of functions that access to different layers of the Internet protocol architecture. The functions in the library are platform independent, because the platform dependent codes are taken care of by Libnet.

The following shows some of the supported protocols and the corresponding Internet protocol architecture layers:

`Application layer`	`RIP OSPF DNS DHCP`
`Transport layer`	`TCP UDP`
`Network layer`	`IP ICMP OSPF`
`Link layer`	`ARP CDP Ethernet`

There are a few functions that need to be called to create the libnet environment and to free up the resources. `libnet_init()` should be called first to create a libnet environment, and it returns a libnet context that is used in other libnet functions.

```
libnet_t * libnet_init(int injection_type, char *device, char *err_buf);
```

The `injection_type` parameter accepts `LIBNET_LINK, LIBNET_LINK_ADV, LIBNET_RAW4, LIBNET_RAW4_ADV, LIBNET_RAW6,` and `LIBNET_RAW6_ADV`. These primitives provides two methods to build packets. When `LIBNET_LINK` or `LIBNET_LINK_ADV` is used then packets are built starting at the data-link layer. Other primitives are used to built raw packets at IP layer. The `device` parameter accepts the network interface device name. The `err_buf` parameter is a buffer for holding error message, and it should be declared statically as `char err_buf[LIBNET_ERRBUF_SIZE]`.

```
void libnet_destroy(libnet_t *l);
```

Another required function that need to be called is `libnet_destroy()`. This function should be called when done with libnet for closes the network interface and free up the resources. It accepts the value that was returned by `libnet_init()`.

```
u_int32_t libnet_get_ipaddr4(libnet_t *l);
```

`libnet_get_ipaddr4()` returns the IP address of the device that was initialized.

```
struct libnet_ether_addr *libnet_get_hwaddr(libnet_t *l);
```

`libnet_get_hwaddr()` returns the MAC address for the device that was initialized.

```
libnet_ptag_t libnet_build_ipv4(u_int16_t len, u_int8_t tos, u_int16_t id,
                                u_int16_t frag, u_int8_t ttl, u_int8_t prot,
                                u_int16_t sum, u_int32_t src, u_int32_t dst,
                                u_int8_t *payload, u_int32_t payload_s,
                                libnet_t *l, libnet_ptag_t ptag);
```

`libnet_build_ipv4()` is for building IP version 4 header. It returns the protocol tag value on success, and -1 on error. The following are explanation of the parameters:

`len` is the total length of the IP packet including all subsequent data

`tos` is the type of service bits

`id` is the IP identification number

`frag` is the fragmentation bits and offset

`ttl` is the time to live in the network

`prot` is the upper layer protocol

`sum` is the checksum (0 for libnet to autofill)

`src` is the source IPv4 address (little endian)

`dst` is the destination IPv4 address (little endian)

`payload` is the optional payload or set to NULL

`payload_s` is the payload length or 0

`l` is the pointer to a libnet context

`ptag` is the protocol tag to modify an existing header, 0 to build a new one