パケットを作成するライブラリ libnet について

2015/7/3

藤田将輝

1 はじめに

現在，デバッグ支援環境の機能であるパケットジェネレータの実装のため，パケットを作成するライブラリを調査している．本資料では，任意のプロトコルを作成できるライブラリである libnet について説明する．libnet では libnet コンテキストという独自のデータを用いて，パケットを作成する．libnet コンテキストにそれぞれのレイヤにおけるプロトコルのヘッダを作成することでパケットを作成する．また，libnet を用いてパケットを作成できることを確認した．しかし，作成したパケットがデバイスドライバにおいて正常に処理されるかについては確認していない．以降の章では，libnet を用いた場合のパケットの作成方法について述べる．

2 libnet

libnet は任意のプロトコル，レイヤのパケットを作成するライブラリである．libnet を用いて，パケットを作成する際は，まず libnet コンテキストという，ライブラリ固有のデータを初期化する必要がある．このデータに，ヘッダ等の情報を付与していく形でパケットが作成される．例として，libnet を使用した際の，UDP を作成する手順を図 1 に示し，以下で説明する．

(1) `libnet_init()` を実行し，libnet コンテキストを初期化する．
(2) `libnet_build_udp()` を実行し，UDP ヘッダと payload を作成する．
(3) `libnet_build_ipv4()` を実行し，IPv4 ヘッダを作成する．
(4) `libnet_build_ethernet()` を実行し，Ether ヘッダを作成する．
(5) `libnet_destroy()` を実行し，libnet コンテキストを開放する．

3 パケットの作成に必要なデータの作成

パケットを作成する際に必要なデータとして，以下のデータが挙げられる．

(1) Ether ヘッダ
　（A）宛先 MAC アドレス
　（B）送信元 MAC アドレス
(2) IPv4 ヘッダ
　（A）宛先 IP アドレス
　（B）送信元 IP アドレス

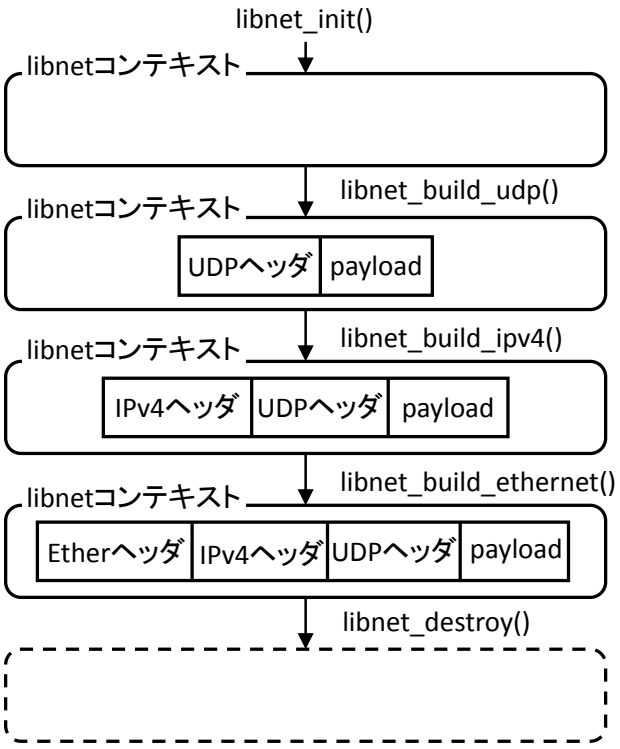

図 1　パケット作成流れ

(3) UDP ヘッダ

（A）宛先ポート

（B）送信元ポート

libnet で定義されている関数を用いることで，送信元の MAC アドレス，IP アドレスを容易に作成できる．また，ユーザが宛先アドレスを指定する際に，バイトオーダ等の情報を変換できる関数も定義されている．さらに，各ヘッダを定義する関数を用いることで，全てのヘッダの情報を自動的に作成できる．

4　ヘッダを作成する関数

libnet で定義されている関数の中で，UDP パケットを作成する際に使用される関数について以下に示し説明する．

(1) `libnet_build_udp()`

【形式】

```
libnet_ptag_t libnet_build_udp(uint16_t sp, uint16_t dp, uint16_t len,
uint16_t sum,t uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t
ptag)
```

【引数】

`uint16_t sp`: 送信元ポート番号

uint16_t dp: 宛先ポート

uint16_t len: UDP パケット全体のサイズ

uint16_t sum: チェックサム (0 を指定すると自動で決定する．)

uint8_t *payload: payload

uint32_t payload_s: payload のサイズ libnet_t *l: libnet コンテキスト

libnet_ptag_t ptag: libnet コンテキストに付与されるタグ

【戻り値】

成功: ptag

失敗: -1

【機能】

引数をもとに，UDP ヘッダを作成し，libnet コンテキストに付与する．

(2) libnet_build_ipv4()

【形式】

libnet_ptag_t libnet_build_ipv4(u_int16_t ip_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)

【引数】

u_int16_t ip_len: IP パケット全体のサイズ

u_int8_t tos: サービスビットのタイプ

u_int16_t id: IP を識別する番号

u_int16_t frag: フラグメンテーションビットとそのオフセット

u_int8_t ttl: IP パケットの寿命

u_int8_t prot: 次に続くプロトコルの識別

u_int16_t sum: チェックサム

u_int32_t src: 送信元 IP アドレス

u_int32_t dst: 宛先 IP アドレス

u_int8_t * payload: payload

u_int32_t payload_s: payload のサイズ

libnet_t* l: libnet コンテキスト

libnet_ptag_t ptag: libnet コンテキストに付与されるタグ

【戻り値】

成功: ptag

失敗: -1

【機能】

引数をもとに，IPv4 ヘッダを作成し，libnet コンテキストに付与する．

(3) `libnet_build_ethernet()`

【形式】

`libnet_ptag_t libnet_build_ethernet(u_int8_t* dst, u_int8_t * src, u_int16_t type, u_int8_t * payload, u_int32_t payload_s, libnet_t * l, libnet_ptag_t ptag)`

【引数】

`u_int8_t* dst`: 宛先 MAC アドレス

`u_int8_t * src`: 送信元 MAC アドレス

`u_int16_t type`: 次に続くプロトコルの識別

`u_int8_t * payload`: payload

`u_int32_t payload_s`: payload のサイズ

`libnet_t* l`: libnet コンテキスト

`libnet_ptag_t ptag`: libnet コンテキストに付与されるタグ

【戻り値】

成功: `ptag`

失敗: -1

【機能】

引数をもとに，Ether ヘッダを作成し，libnet コンテキストに付与する．

5 パケットの抽出方法

作成されたパケットは libnet コンテキストに格納されており，これをパケットとして取り出す必要がある．`libnet_adv_cull_packet()` を使用することで，パケットを取り出せる．引数に，`u_int8_t` の配列と `u_int32_t` の変数を与えることで，作成したパケットとこのサイズを取得できる．詳細について以下に示し，説明する．

【形式】

`int libnet_adv_cull_packet(libnet_t* l, u_int8_t** packet, u_int32_t* packet_s)`

【引数】

`libnet_t* l`: libnet コンテキスト

`u_int8_t** packet`: パケットが格納される配列

`u_int32_t* packet_s`: パケットのサイズが格納される変数

【戻り値】

成功: 1

失敗: -1

【機能】

作成したヘッダ等の情報をパケットに格納する．また，このサイズを格納する．

6 課題

linnet を用いることで，指定したポート，IP アドレス，および MAC アドレスでパケットを作成できることを確認した．次の課題として以下の課題が考えられる．

(1) libnet を用いて作成したパケットが正常に処理されるかの調査
libnet を用いることでパケットを作成できることを確認したが，このパケットが正常に処理されるかどうかを確認できていない．このため，これを調査する．

(2) 各種のヘッダを作成する関数の処理流れの調査
`libnet_build_udp()` などのヘッダを作成する関数の内部処理について調査する．また，これを参考にして，パケットジェネレータの実装をすすめる．

7 おわりに

本資料では libnet を用いたパケットの作成方法について述べた．また，libnet を用いることで，パケットを作成できることを確認した．今後の課題として，作成したパケットが本デバッグ支援機構において正常に処理されるかどうかを確認することと，`libnet_build_udp()` などのヘッダを作成する関数内の処理を調査し，これを参考にパケットジェネレータを作成することが挙げられる．