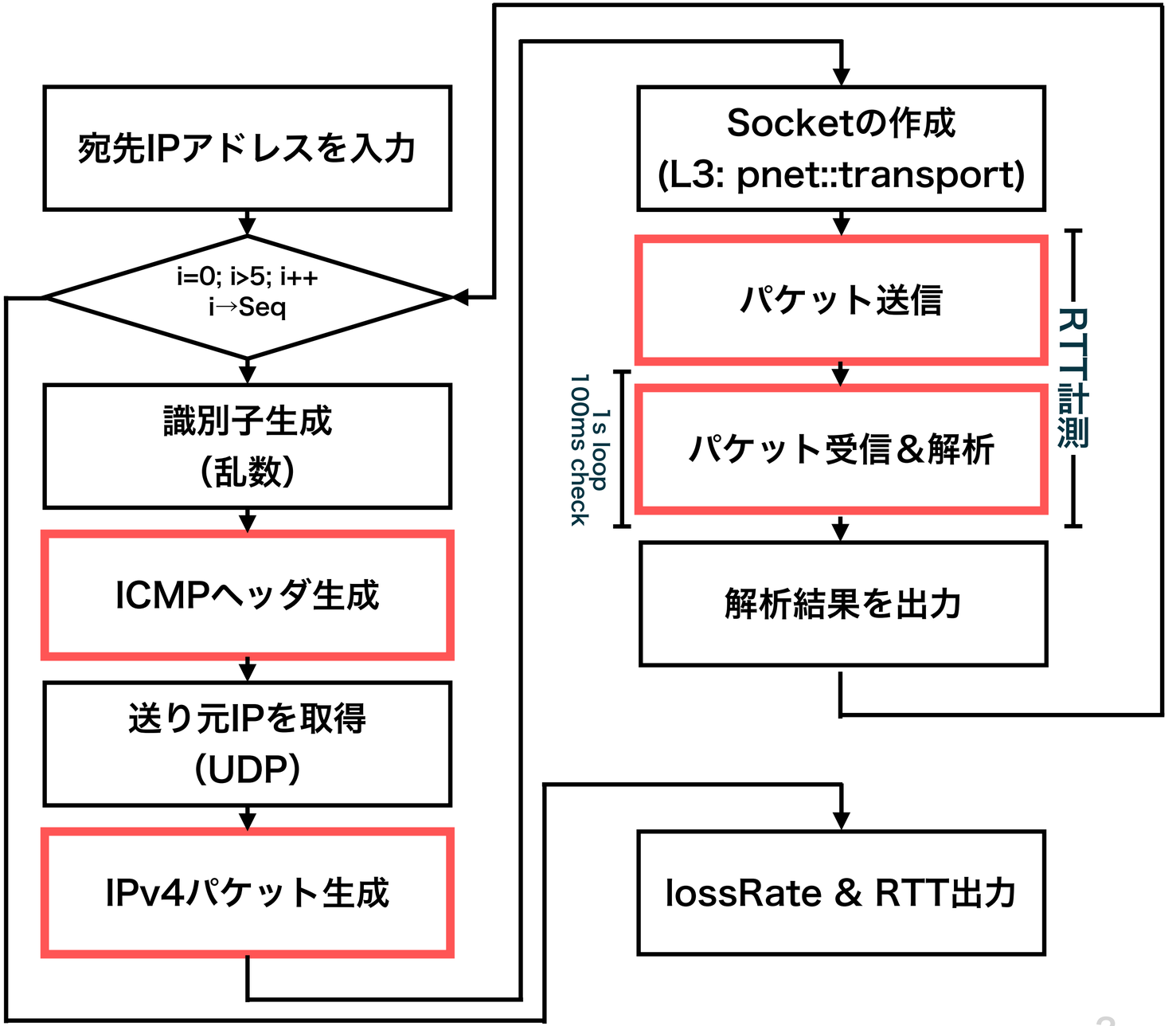
情報メディア通信工学 ping 最終報告

2455026 平野碧生 2025-11-17

実装物の全体像

- ICMPヘッダの生成
- IPv4ヘッダの生成
- パケットの送信＆受信監視
- パケット解析：ping結果
- 補足
  - パケット長問題の原因調査



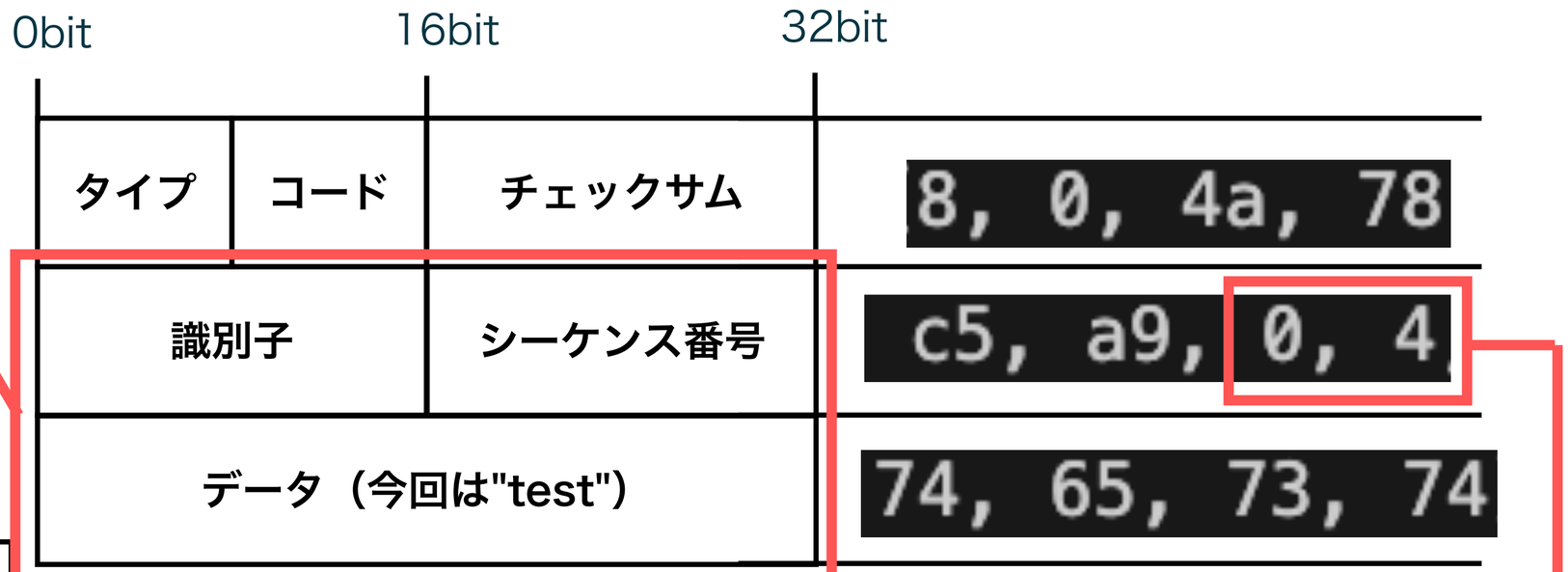
ICMPヘッダの生成

データ部分に

- 識別子
- シーケンス番号
- メッセージ("test")

を追加

シーケンス番号は5回のうち何番目かを判別する用に設定

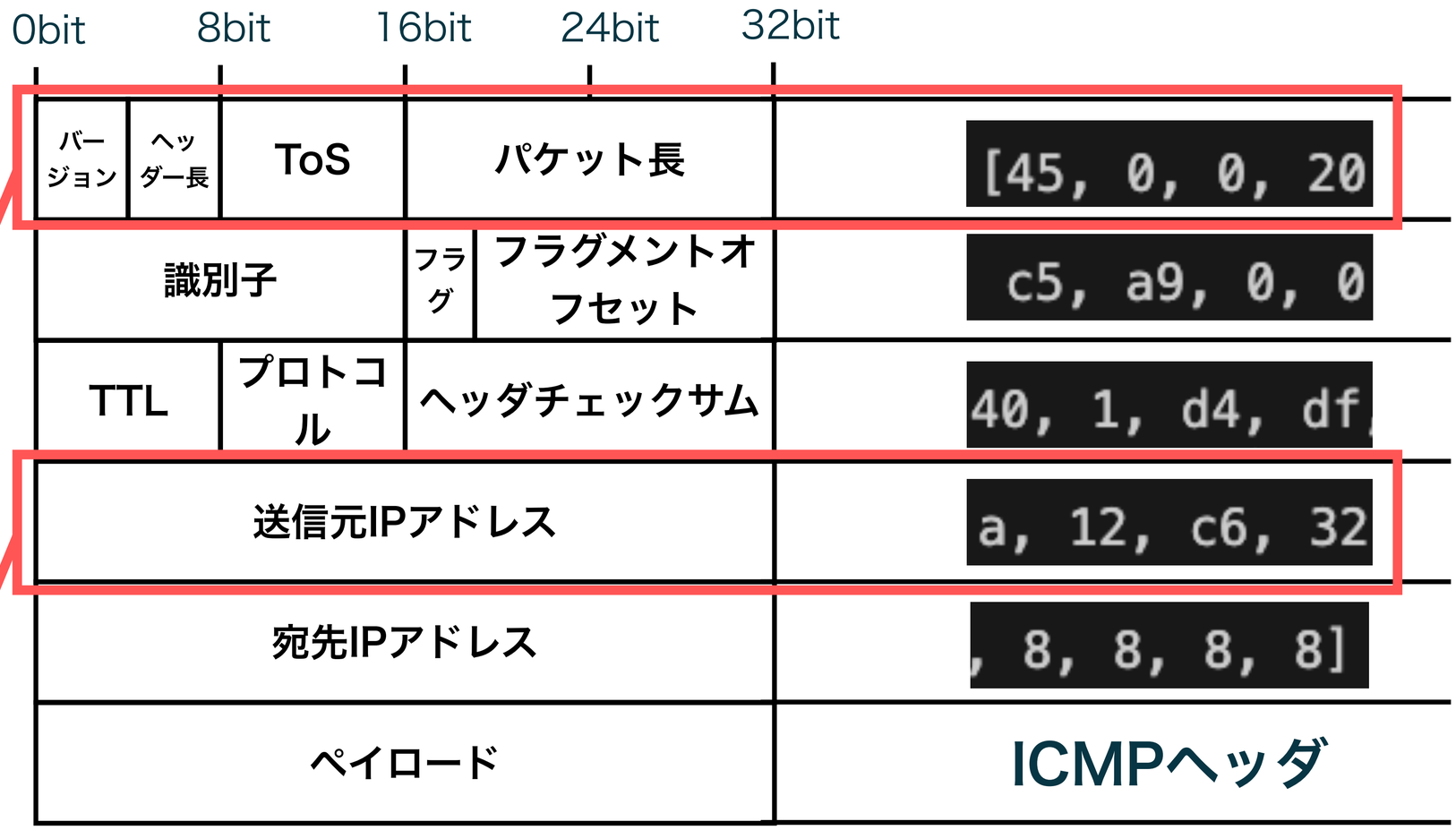


ICMPパケットフォーマット　　実際に送信したパケット

IPv4ヘッダの生成

- header_len=20bit
- total_len=
  header_len
  +
  payload_len

- 送信元IPアドレス
  - UDP通信で取得



IPv4パケットフォーマット　　実際に送信したパケット

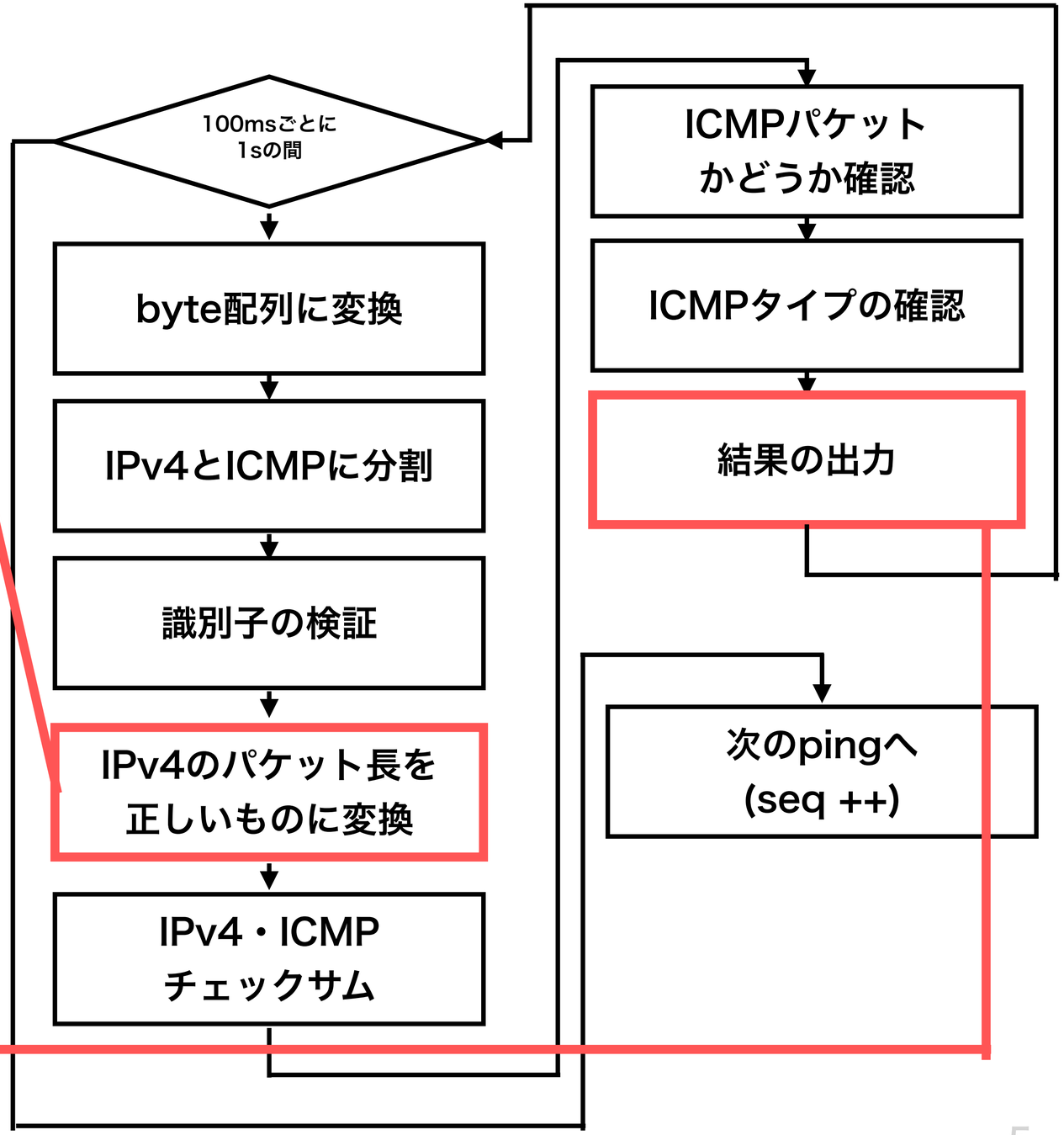
パケットの送信＆受信監視

パケット長が実際のものと異なるため、
正しいものになるように上書き

127.0.0.1の場合はチェックサムをパス

```
----- パケット受信 -----

受信パケット(IPv4ヘッダ):[45, 0, 0, c, 0, 0, 0, 0, 75, 1, 65, 89, 8, 8, 8, 8, a, 1
受信パケット(ICMPヘッダ):[0, 0, b1, 95, 66, 8c, 0, 4, 74, 65, 73, 74]
受信パケットを修正(パケット長→32):[45, 0, 0, 20, 0, 0, 0, 0, 75, 1, 65, 89, 8, 8,

↓IPv4ヘッダ
チェックサム検証 → 0xffff

↓ICMPヘッダ
チェックサム検証 → 0xffff

↓ICMP Echo Reply
type = 0 code = 0 checksum =  0xb195 identification = 0x668c seq = 4 data = test
time = 35.5034 ms

----------------------
```

結果の出力の様子



パケット解析：ping結果

- 外部

  (→8.8.8.8)

```
PING 8.8.8.8 (8.8.8.8): 32 data bytes
32 bytes from 8.8.8.8: icmp_seq=0 ttl=113 time=25.829 ms
32 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=35.895 ms
32 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=39.776 ms
32 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=33.590 ms
32 bytes from 8.8.8.8: icmp_seq=4 ttl=113 time=34.211 ms
loss_rate:0.0%
round-trip min/avg/max/stddev = 25.829/33.860/39.776/4.557 ms
```

- ローカル

  (→デフォルトゲートウェイ)

```
PING 192.168.3.1 (192.168.3.1): 32 data bytes
32 bytes from 192.168.3.1: icmp_seq=0 ttl=64 time=3.226 ms
32 bytes from 192.168.3.1: icmp_seq=1 ttl=64 time=3.886 ms
32 bytes from 192.168.3.1: icmp_seq=2 ttl=64 time=3.650 ms
32 bytes from 192.168.3.1: icmp_seq=3 ttl=64 time=4.595 ms
32 bytes from 192.168.3.1: icmp_seq=4 ttl=64 time=5.579 ms
loss_rate:0.0%
round-trip min/avg/max/stddev = 3.226/4.187/5.579/0.825 ms
```

- 自身

  （→ループバックアドレス）

```
PING 127.0.0.1 (127.0.0.1): 32 data bytes
32 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.110 ms
32 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.131 ms
32 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.106 ms
32 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.110 ms
32 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.139 ms
loss_rate:0.0%
round-trip min/avg/max/stddev = 0.106/0.119/0.139/0.013 ms
```

補足：パケット長問題の原因調査

1. pnet: rawバイト列

   12byte ← 解析ミスではない

2. Wireshark: rawパケット

   32byte ← 受信パケットは正常

3. C: pnet libを使わずに解析

   12byte ← lib依存ではない

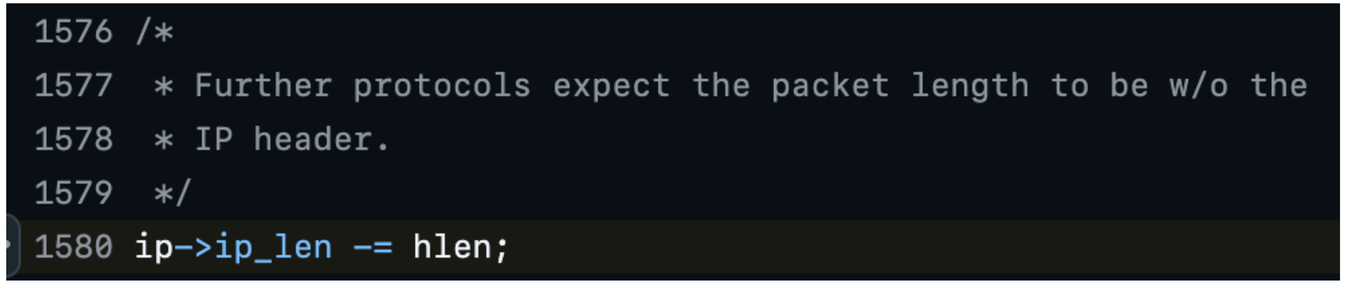
4. OS: OSでのpacket処理を確認

   →MacOSのカーネルソースを確認

```
Waiting for ICMP packets... (run ping in another terminal)
=== received packet ===
recv_len            = 32
ip_header_len       = 20
ip_total_length     = 3072
raw[2:4]            = 0x0c 0x00
protocol            = 1
ICMP payload bytes (first 16): 00 00 b0 5a 67 cb 00 00 74 65 73 74

=== received packet ===
recv_len            = 32
ip_header_len       = 20
ip_total_length     = 3072
raw[2:4]            = 0x0c 0x00
protocol            = 1
ICMP payload bytes (first 16): 00 00 b3 c0 64 64 00 01 74 65 73 74

=== received packet ===
recv_len            = 32
ip_header_len       = 20
ip_total_length     = 3072
raw[2:4]            = 0x0c 0x00
protocol            = 1
ICMP payload bytes (first 16): 00 00 01 79 16 ab 00 02 74 65 73 74

=== received packet ===
recv_len            = 32
ip_header_len       = 20
ip_total_length     = 3072
raw[2:4]            = 0x0c 0x00
protocol            = 1
ICMP payload bytes (first 16): 00 00 6b 0c ad 16 00 03 74 65 73 74
```

補足：パケット長問題の原因調査

- MacOSのカーネルソースを確認



```
1576 /*
1577  * Further protocols expect the packet length to be w/o the
1578  * IP header.
1579  */
1580 ip->ip_len -= hlen;
```

xnu-11417.121.6におけるカーネルソース

上位プロトコルへ渡す際はペイロード長を渡すようになっている

- 今回使用したソケットはレイヤー3だったので、レイヤー2の処理はOS依存
  - レイヤー2→3の際に「パケット長」から「ペイロード長」に置き換えられた
    → **修正は厳しいため、今回はパケット長を上書きする形で実装**

参考文献

- MacOSのカーネルソース仕様：https://github.com/apple-oss-distributions/xnu/blob/xnu-11417.121.6/bsd/netinet/ip_input.c
- チェックサム仕様：https://www.mew.org/~kazu/doc/bsdmag/cksum.html
- ICMP, IPv4仕様：「図解入門TCP/IP」第2版 みやたひろし著
- 各ライブラリ：https://docs.rs/
  - pnet(パケット周りの処理)：https://docs.rs/pnet/latest/pnet/
  - rng（乱数生成）：https://docs.rs/rand/latest/rand/trait.Rng.html
  - regex（正規表現）：https://docs.rs/regex/latest/regex/