第3回 ping実装

平野碧生 2455026 2025-11-10

実装内容（全体の流れ）

IPv4パケットヘッダの生成を一から実装

- 8ビット配列型式でヘッダを作成
- IPv4チェックサムの実装

受信したパケットのチェックサムを検証

- IPv4ヘッダのチェックサム
- ICMPヘッダのチェックサム

パケット長が合わない問題の調査

- WireSharkでパケットの調査
- ソースコードの調査

IPv4パケットの作成(create_ipv4_packet)

既存ライブラリ→ライブラリ使用せず自分で実装

1. ヘッダのフィールド中身を定義
2. 8ビットに変換後、配列に逐次格納
3. チェックサム結果を取得→該当indexに格納
4. ペイロードを格納

最終的にIPパケット全てにおいて、
外部のライブラリに依存しない形での実装を行なった

```
let tmp_version:u8 = 4;
let tmp_header_length: u8 = 5;
let tmp_dscp:u8 = 0;
let tmp_ecn:u8 = 0;
let tmp_packet_length:u16 = (header_len + payload.len()) as u16;
let tmp_df: u8 = 0;
let tmp_mf: u8 = 0;
let tmp_frags: u8 = tmp_df << 1 | tmp_mf;
let mut tmp_fragment_offset: u16 = 0;
let tmp_ttl: u8 = 64;
let tmp_protocol_num: u8 = 1; // ICMPプロトコル = 1
let mut tmp_checksum: u16 = 0;
let mut checksum_frag1: usize = 0;
let mut checksum_frag2: usize = 0;
let tmp_destination: u32 = destination_ipv4.to_bits();
let tmp_source: u32 = source_ipv4.to_bits();

ipv4_packet.push((tmp_version << 4) | tmp_header_length);
ipv4_packet.push((tmp_dscp << 6) | tmp_ecn);
ipv4_packet.push((tmp_packet_length >> 8) as u8);
ipv4_packet.push((tmp_packet_length & 0xff) as u8);
ipv4_packet.push((identification >> 8) as u8);
ipv4_packet.push((identification & 0xff) as u8);
    tmp_fragment_offset = tmp_fragment_offset & 0x1FFF;
ipv4_packet.push((tmp_frags << 5) | (tmp_fragment_offset >> 8) as u8);
ipv4_packet.push((tmp_fragment_offset & 0xff) as u8);
ipv4_packet.push(tmp_ttl);
ipv4_packet.push(tmp_protocol_num);
ipv4_packet.push((tmp_checksum >> 8) as u8);
    checksum_frag1 = ipv4_packet.len() - 1;
ipv4_packet.push((tmp_checksum & 0xff) as u8);
    checksum_frag2 = ipv4_packet.len() - 1;
ipv4_packet.extend_from_slice(&tmp_source.to_be_bytes().to_vec());
ipv4_packet.extend_from_slice(&tmp_destination.to_be_bytes().to_vec());
```

上部にてフィールドを定義
下部にて各フィールドを配列に格納

受信したパケットのチェックサムを検証(check_checksum)

ICMP、IPv4ヘッダの2つに対して検証

1. 8ビット配列のヘッダ情報を取得
2. 1の補数和を算出
3. 算出結果が0xffffであれば検証OK
4. 16ビット配列へ変換してから補数和の算出するのではなく、2要素ごとに配列要素を取り出し補数和を算出することで計算量を削減

IPv4のチェックサムにおいてヘッダー長が実際の長さが32byteのところ12byteとなっておりチェックサムに失敗する

```
↓IPv4ヘッダ
チェックサム検証 → 0xffeb

チェックサムの検証に失敗しました
↓ICMPヘッダ
チェックサム検証 → 0xffff
```

console output: 0xffebとなっており失敗していることがわかる

パケット長が合わない問題の調査&対応(analysis_packet)

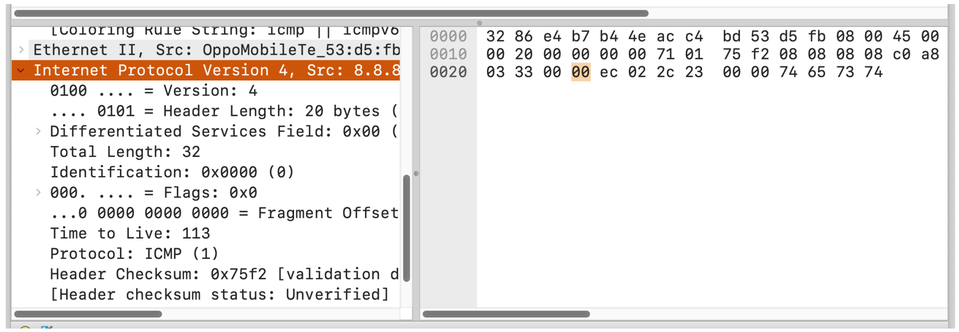
WireSharkでパケットorプログラムのどちらが原因なのかを調査

- 送信パケットと受信パケットの中身（実際のパケット長）を調べる

```
送信パケット(IPv4ヘッダ):[45, 0, 0, 20, 4b, 81, 0, 0, 40, 1, 5b, 71, c0, a8, 3, 33, 8, 8, 8, 8]
送信パケット(ICMPヘッダ):[8, 0, c4, a0, 4
                                          同じ配列要素数に対して異なるパケット長
受信パケット(IPv4ヘッダ):[45, 0, 0, c, 0, 0, 0, 0, 0, 71, 1, 75, f2, 8, 8, 8, 8, c0, a8, 3, 33]
受信パケット(ICMPヘッダ):[0, 0, cc, a0, 4b, 81, 0, 4, 74, 65, 73, 74]
```

- WireSharkで生のパケットの中身を調べる

  → パケットには問題なし



パケット自体では、正しいパケット長を
確認できた

ライブラリに原因がありそう（未調査）
→ 一旦パケット長を上書きする形で対応

参考資料

- 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