

# tracerouteの実装 2025-12-15

2455026 平野碧生

# tracerouteで必要な情報

```
$ traceroute 8.8.8.8
Traceroute to 8.8.8.8 (8.8.8.8), 64 hops max, 40 byte packets
 1 133.55.72.1 (133.55.72.1) 8.481 ms 6.007 ms 3.738 ms
 2 133.55.105.3 (133.55.105.3) 4.652 ms 4.396 ms 4.693 ms
 3 133.55.104.200 (133.55.104.200) 4.428 ms 4.700 ms 5.839 ms
 4 150.99.189.189 (150.99.189.189) 5.874 ms 14.596 ms 22.334 ms
 5 150.99.13.181 (150.99.13.181) 11.627 ms 12.380 ms 12.457 ms
 6 210.173.184.57 (210.173.184.57) 11.844 ms 11.208 ms 11.280 ms
 7 216.239.59.81 (216.239.59.81) 11.488 ms
 108.170.255.229 (108.170.255.229) 11.173 ms
 192.178.110.81 (192.178.110.81) 14.086 ms
 8 142.250.236.35 (142.250.236.35) 11.162 ms
 142.250.239.221 (142.250.239.221) 41.000 ms
 142.251.69.231 (142.251.69.231) 11.271 ms
 9 dns.google (8.8.8.8) 13.559 ms 12.482 ms 12.369 ms
```

tracerouteの出力結果

ホップの設定ごとに 3 つのプローブを送信

① ホップ値 (何ホップ目か)

② ゲートウェイのアドレス

→ アドレスが異なる場合は別行で出力

③ 正常なプローブごとの往復時間

# tracerouteの実装

## 前回のping作成時に実装した機能を流用

- ICMP&IPv4パケットの自前作成

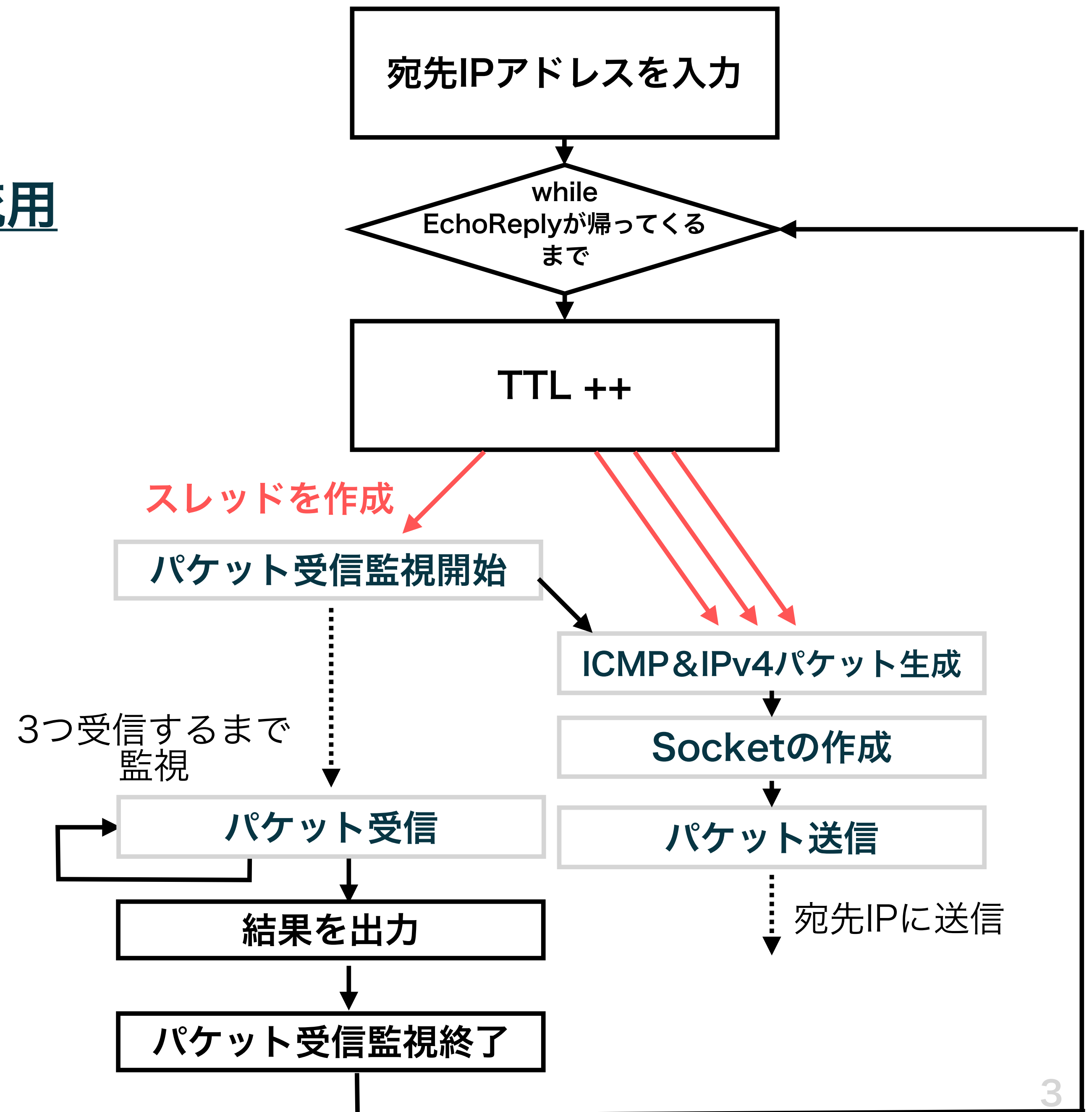
## 自前でパケット解析

- 生パケットを自分で解析

## 既存のtracerouteと異なる部分

- 並列処理での実行

OSはMacOS、言語はRustで実装



# 前回の問題調査

## これまでに起こった問題点

- 識別子が合わない問題
- ICMPのチェックサム検証失敗問題（パケット長）
- ICMPのチェックサム検証失敗問題（DF）



## 今回やったこと

- これまで使っていたpnetクレートではなくlibpcapを使ったパケット解析に差し替える
  - 生パケットから自分で解析
- 各TTLごとに受信用のスレッドを新規作成し、送信パケットの応答を監視
  - 動的なパケット受信となったので、ICMPペイロードにRTT計測開始時間を格納するよう変更

# libpcap

Rustでlibpcapを叩くラッパーがあったのでそれを利用

Wiresharkと同じパケットを取得できたことを確認

→ pnetからlibpcapに変更

- チェックサム検証失敗問題1&2
- 識別子が合わない問題

の2つを解消

The image shows a comparison of packet data between three sources: Wireshark, libpcap, and a previous packet. The data is presented in hexadecimal format. Red boxes highlight the packet length (00 38) and header length (00 20) fields. Red lines connect these fields to labels above the comparison: 'パケット長' (Packet Length) and 'ヘッダー長' (Header Length).

パケット長

ヘッダー長

Wireshark:

```
45 00 00 38 9c 90 40 00 7c 01 d0 fd ac 10 20 e2
c0 a8 03 9c 0b 00 1f 91 bd 48 00 00 45 00 00 20
48 3d 00 00 00 01 9e 4c c0 a8 03 9c 08 08 08 08
08 00 c7 e8 48 3d 00 00
```

libpcap:

```
45 00 00 38 9c 90 40 00 7c 01 d0 fd ac 10 20 e2
c0 a8 03 9c 0b 00 1f 91 bd 48 00 00 45 00 00 20
48 3d 00 00 00 01 9e 4c c0 a8 03 9c 08 08 08 08
08 00 c7 e8 48 3d 00 00
```

previous packet:

```
45 00 00 24 9c 90 00 00 7c 01 d0 fd ac 10 20 e2
c0 a8 03 9c 0b 00 1f 91 bd 48 00 00 45 00 20 00
48 3d 00 00 00 01 9e 4c c0 a8 03 9c 08 08 08 08
08 00 c7 e8 48 3d 00 00
```



# My tracerouteの成果

ほとんど標準ライブラリしか使わずにtracerouteを実装

```
// --- 自作ライブラリ ---
use utility;
use utility::check_checksum;

// --- 外部ライブラリ ---
use rand::{rng, Rng};

// --- 標準ライブラリ --- You, 1 秒前 • Uncomm
use std::time::{Duration, Instant};
use std::net::Ipv4Addr;
use std::thread;
use std::sync::{Arc, Mutex};
use std::collections::HashMap;
use std::sync::atomic::{AtomicBool, Ordering};
use std::io::Write;
```

```
[package]
name = "traceroute"
version = "0.1.0"
edition = "2024"

[dependencies]
libpcap = "0.1.7"
rand = "0.9.2"

[dependencies.utility]
path = "../utility"
```

外部のライブラリは、libpcapとrandのみ

- libpcap: パケット解析
- rand: 乱数生成

# My tracerouteの実行結果

既存のtracerouteと同じような出力をするように実装

```
traceroute to 8.8.8.8 (8.8.8.8), 64 hops max
1 192.168.3.1 8.080 ms 8.226 ms 8.246 ms
* * *
3 172.16.128.130 37.623 ms 37.885 ms 37.918 ms
4 172.16.32.165 37.914 ms 37.898 ms 37.977 ms
5 172.16.32.226 40.075 ms 39.953 ms 51.312 ms
6 10.0.16.161 37.359 ms 37.587 ms 37.712 ms
7 10.216.21.65 39.574 ms 39.708 ms 39.653 ms
8 10.216.21.66 43.259 ms 43.411 ms 43.518 ms
9 10.9.203.6 60.672 ms 62.651 ms 61.590 ms
10 74.125.146.69 38.139 ms 37.266 ms 37.251 ms
11 209.85.245.105 40.173 ms 40.657 ms 41.026 ms
12 142.251.251.3 37.242 ms 37.307 ms 48.282 ms
13 8.8.8.8 36.911 ms 37.031 ms 36.983 ms
```

My traceroute

```
traceroute to 8.8.8.8 (8.8.8.8), 64 hops max, 40 byte packets
1 192.168.3.1 (192.168.3.1) 6.155 ms 3.076 ms 4.129 ms
2 * * *
3 172.16.128.130 (172.16.128.130) 29.642 ms 30.007 ms 25.007 ms
4 172.16.32.165 (172.16.32.165) 28.412 ms 32.392 ms 30.076 ms
5 172.16.32.226 (172.16.32.226) 27.903 ms 27.511 ms 25.391 ms
6 10.0.16.161 (10.0.16.161) 26.462 ms 34.875 ms 23.856 ms
7 10.0.11.125 (10.0.11.125) 25.082 ms
  10.0.11.113 (10.0.11.113) 27.544 ms
  10.216.21.73 (10.216.21.73) 30.266 ms
8 10.216.21.78 (10.216.21.78) 40.063 ms
  10.216.21.70 (10.216.21.70) 29.629 ms
  10.216.21.78 (10.216.21.78) 32.525 ms
9 10.9.203.2 (10.9.203.2) 31.551 ms 32.279 ms 31.201 ms
10 74.125.146.69 (74.125.146.69) 33.623 ms 30.046 ms 33.802 ms
11 * * *
12 dns.google (8.8.8.8) 37.486 ms 23.979 ms 25.093 ms
```

既存のtraceroute

# 参考文献

- IBM Power10 tracerouteコマンド:  
<https://www.ibm.com/docs/ja/power10/9028-21B?topic=commands-traceroute-command>
- The Rust Programing Language(恐れるな！並行性):  
<https://doc.rust-jp.rs/book-ja/ch16-00-concurrency.html>
- rust mpsc:  
<https://doc.rust-lang.org/std/sync/mpsc/index.html>
- RFC 777:  
<https://datatracker.ietf.org/doc/html/rfc777>
- rust libpcap:  
<https://docs.rs/libpcap/latest/libpcap/index.html>