

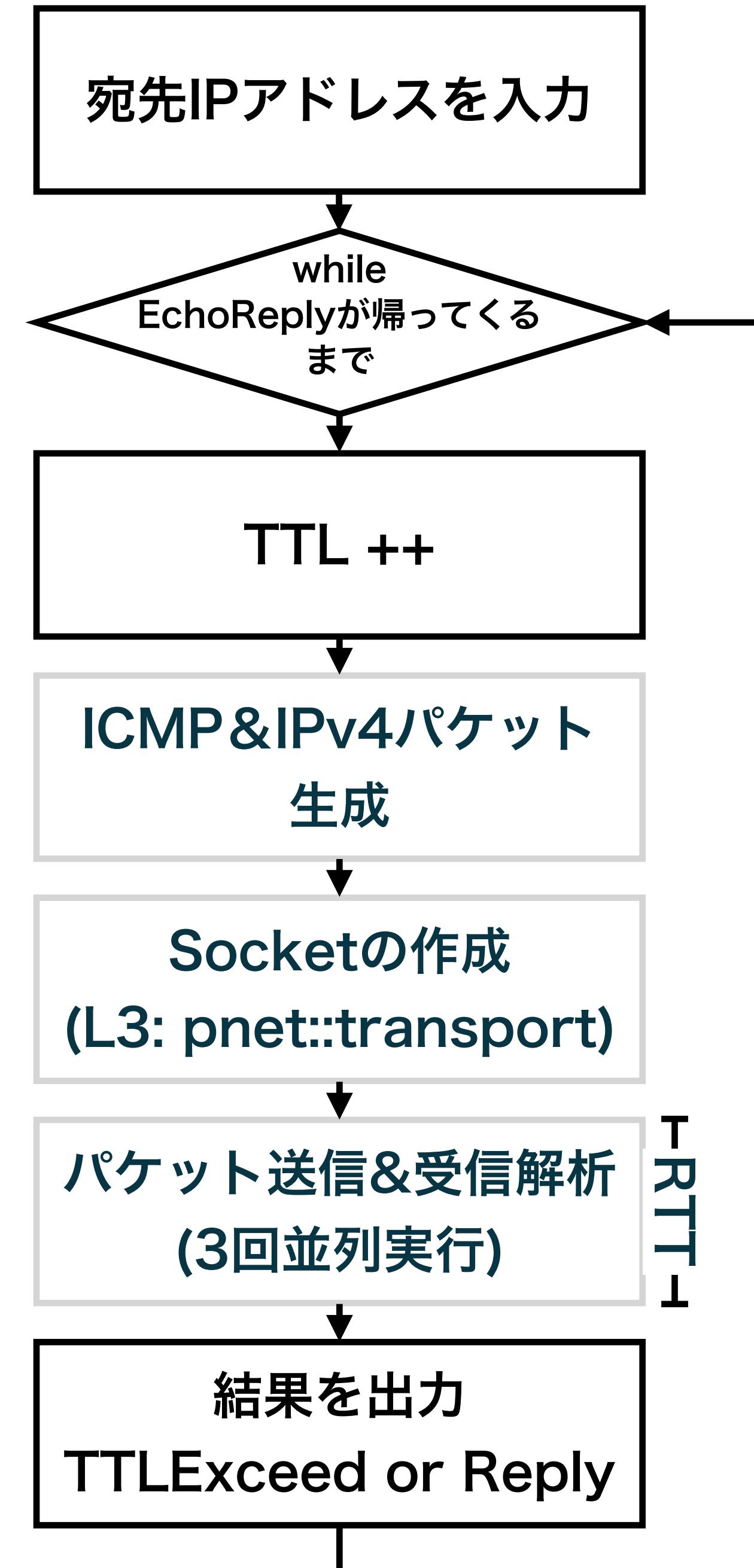
tracerouteの実装 2025-12-8

2455026 平野碧生

tracerouteの実装

- 前回のping作成時に実装した機能を流用
 - ICMPパケットヘッダ生成
 - IPv4パケットヘッダ生成
 - Socketの作成
 - パケット送信
 - パケット受信&解析

OSはMacOS、言語はRustで実装する



前回の問題調査

- 識別子が合わない問題
- ICMPのチェックサム検証失敗問題

識別子が合わない問題

ICMPタイプが11(TTLExceeded)のとき識別子が別のものに変わってしまう

- ルータはTTL=1のEchoRequestを受け取った際にパケットを破棄&返信用パケット生成を行うことからパケット自体が異なることがわかる

EchoRequest

タイプ	コード	チェックサム
識別子	シーケンス番号	
ペイロード		

TTLExceeded

タイプ	コード	チェックサム
0000		
EchoRequestのコピー		

実際に受信したパケット

ac	c4	bd	53	d5	fb	22	ed	27	9f	4a	6b	08	00	45	00
00	28	2c	c7	00	00	01	01	b9	3a	c0	a8	03	1c	08	08
08	08	08	00	13	ab	2c	c7	00	00	74	65	73	74	74	65
73	74	74	65	73	74										

タイプ・コード・チェックサム

22	ed	27	9f	4a	6b	ac	c4	bd	53	d5	fb	08	00	45	c0
00	44	8a	4e	00	00	40	01	68	3d	c0	a8	03	01	c0	a8
03	1c	0b	00	f4	ff	00	00	00	00	45	00	00	28	2c	c7
00	00	01	01	b9	3a	c0	a8	03	1c	08	08	08	08	08	00
13	ab	2c	c7	00	00	74	65	73	74	65	73	74	73	74	65
73	74														

同じ中身

見るべき箇所

これまで見ていた箇所

これまで参照していた識別子の箇所が00になっていた

識別子が合わない問題

ICMPタイプが11の時は参照する識別子の場所を正しい場所に変更

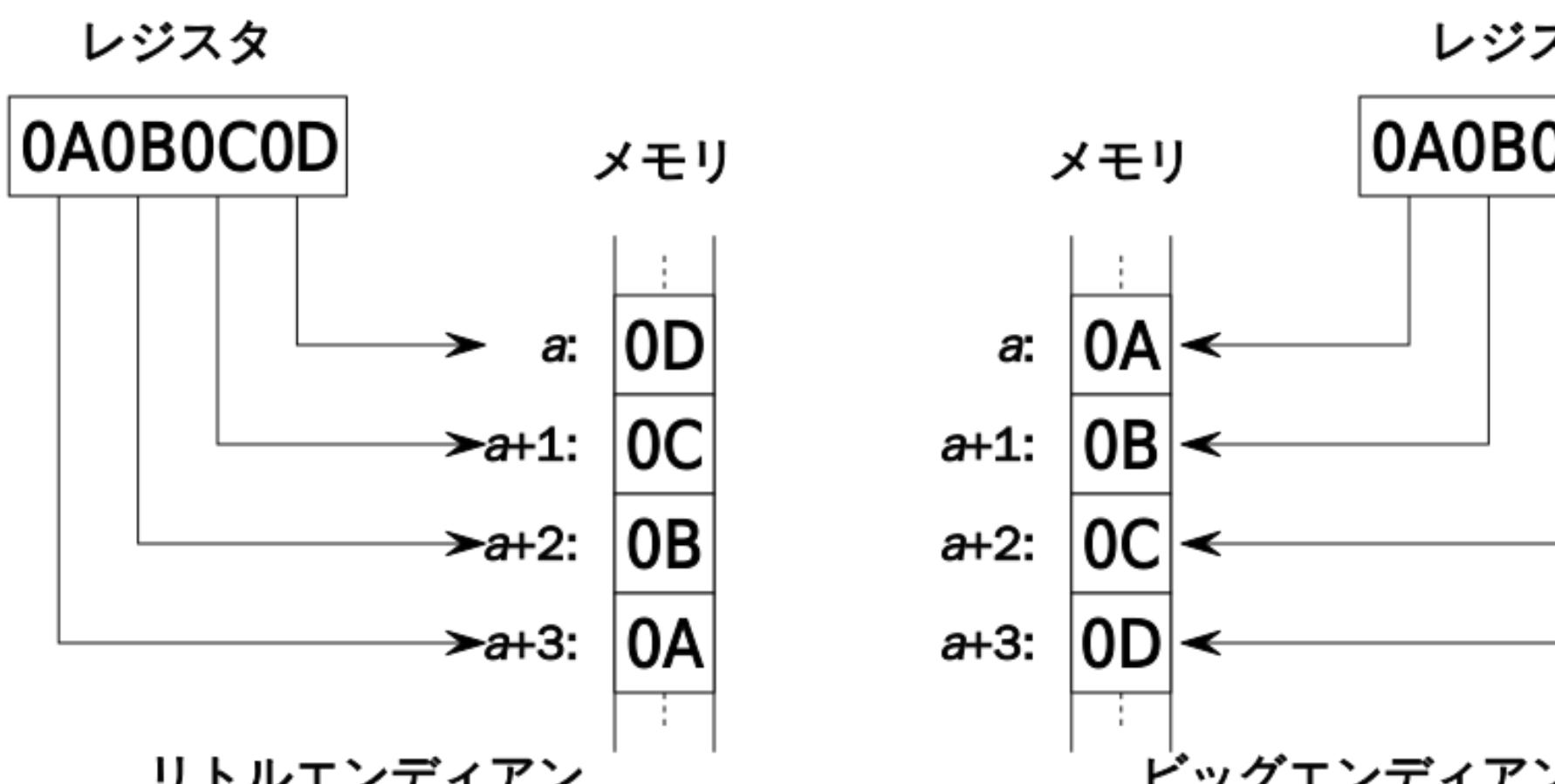
→ 正しい識別子を取得できていることを確認

```
Debug: original_icmp_identification=51683, send_icmp_identification=51683
Debug: original_icmp_identification=26746, send_icmp_identification=26746
1 192.168.3.1 4.609 ms
Debug: original_icmp_identification=12006, send_icmp_identification=12006
1 192.168.3.1 4.580 ms
1 192.168.3.1 4.684 ms
3 * (timeout)
2 *
2 *
2 *
Debug: original_icmp_identification=34472, send_icmp_identification=34472
3 172.16.128.130 23.814 ms
Debug: original_icmp_identification=31563, send_icmp_identification=31563
Debug: original_icmp_identification=38272, send_icmp_identification=38272
3 172.16.128.130 27.396 ms
3 172.16.128.130 27.576 ms
Debug: original_icmp_identification=24748, send_icmp_identification=24748
Debug: original_icmp_identification=50188, send_icmp_identification=50188
Debug: original_icmp_identification=35095, send_icmp_identification=35095
4 172.16.32.165 35.844 ms
4 172.16.32.165 35.782 ms
```

同じ識別子であることを確認

ICMPのチェックサム検証失敗問題

- リトルエンディアンとビッグエンディアンが原因
- レジスタに格納する際にOSに合わせて格納方式を変更
→ バイトの順序が逆になる



[Wikipedia エンディアン](#)

ここでエンディアンを変換している

```
655     changed_header = 1;
656     ip = mtod(m, struct ip *);
657     ip->ip_len = htons(ip->ip_len + (uint16_t)hlen);
658     ip->ip_off = htons(ip->ip_off);
659     ip->ip_sum = 0;
660     ip->ip_sum = ip_cksum_hdr_in(m, hlen);
661
```

[原因と思われるカーネルソース](#)

レジスタ格納時にスワップ→そのまま参照
によってパケット長がバイト列で入れ替わっていた

ICMPのチェックサム検証失敗問題

- リトルエンディアンとビッグエンディアンが原因
- レジスタに格納する際にOSに合わせて格納方式を変更
→ バイトの順序が逆になる

```
let mut fixed_payload = receive_payload.to_vec();
if fixed_payload.len() >= 11 {
    let tmp = fixed_payload[10];
    fixed_payload[10] = fixed_payload[11];
    fixed_payload[11] = tmp;
}
```

traceroute処理内のパケット

0B 00 F4 FF 00 00 00 00 45 00	20 00	0C AD 00 00 01 01 D9 5C C0 A8 03
1C 08 08 08 08 08 00 03 79 0C	AD 00 00 74 65 73 74	

Wiresharkで取得した生パケット

処理の際に入れ替わっている

22 ed 27 9f 4a 6b ac c4 bd 53 d5 fb 08 00 45 c0	00 3c 84 e4 00 00 40 01 6d af c0 a8 03 01 c0 a8
03 1c 0b 00 f4 ff 00 00 00 00 45 00 00 20 0c ad	00 00 01 01 d9 5c c0 a8 03 1c 08 08 08 08 00
03 79 0c ad 00 00 74 65 73 74	

ICMPのチェックサム検証失敗問題

- 強引に入れ替える処理を追加
→ チェックサムの検証成功を確認

```
let mut fixed_payload = receive_payload.to_vec();
if fixed_payload.len() >= 11 {
    let tmp = fixed_payload[10];
    fixed_payload[10] = fixed_payload[11];
    fixed_payload[11] = tmp;
}
```

対象の2バイトを入れ替える処理

```
checksum check: ipv4_header: FFFF, icmp_packet: FFFF
```

チェックサム実行結果

チェックサム検証問題2

- ホップ場所によってはチェックサムが失敗してしまう
- DF(Don't Fragment)がOSによって書き換えられている可能性. . .

45	00	00	38	1F	08	00	00	7C	01	4F	06	AC	10	20	E2	C0	A8	03	1C
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

22	ed	27	9f	4a	6b	ac	c4	bd	53	d5	fb	08	00	45	00				
00	38	1f	08	40	00	7c	01	4f	06	ac	10	20	e2	c0	a8				
03	1c	0b	00	1f	91	bd	48	00	00	45	00	00	20	cb	b4				
00	00	00	01	1b	55	c0	a8	03	1c	08	08	08	08	08	00				
44	6d	cb	b4	00	04														

DF情報がなぜか書き換えられている. . . ?

次週までにDF情報の書き換え問題への対処を行う
キリがなさそうだったらOSを変えることにします

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 Programming Language(恐れるな！並行性):
<https://doc.rust-jp.rs/book-jp/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>
- Wikipedia エンディアン:
<https://ja.wikipedia.org/wiki/%E3%82%A8%E3%83%B3%E3%83%87%E3%82%A3%E3%82%A2%E3%83%B3>

前回の資料による補足

識別子が合わない問題

- EchoReplyが帰ってきた際は識別子の照合は問題なかったが、 TTL Exceeded の場合はなぜか IP ヘッダの識別子が一致しない。 . . .

Time	Source	Destination	Protocol	Length	Identification	Info
339 2025-11-30 13:58:05.987196	192.168.3.2	8.8.8.8	ICMP	46	0x45ec (17900)	Echo (ping) request id=0x45ec, s
340 2025-11-30 13:58:05.987250	192.168.3.2	8.8.8.8	ICMP	46	0xc5bc (50620)	Echo (ping) request id=0xc5bc, s
341 2025-11-30 13:58:05.987280	192.168.3.2	8.8.8.8	ICMP	46	0xfc57 (64599)	Echo (ping) request id=0xfc57, s
342 2025-11-30 13:58:05.991599	192.168.3.1	192.168.3.2	ICMP	74	0xb1c4 (45508)..	Time-to-live exceeded (Time to l
343 2025-11-30 13:58:05.991601	192.168.3.1	192.168.3.2	ICMP	74	0xb1c5 (45509)..	Time-to-live exceeded (Time to l
344 2025-11-30 13:58:05.991602	192.168.3.1	192.168.3.2	ICMP	74	0xb1c6 (45510)..	Time-to-live exceeded (Time to l

送信時のもとの同じ(17900)

0000	72	e2	1d	05	65	5a	ac	c4	bd	53	d5	fb	08	00	45	c0
0010	00	3c	b1	c4	00	00	40	01	40	e9	c0	a8	03	01	c0	a8
0020	03	02	0b	00	f4	ff	00	00	00	00	45	00	00	20	45	ec
0030	00	00	01	01	a0	37	c0	a8	03	02	08	08	08	08	08	00
0040	ca	39	45	ec	00	00	74	65	73	74						

識別子をどう判別するのか？

→ 今週で調査＆実装を行う

IPヘッダ??

ICMPのチェックサム検証失敗問題

- ICMPのペイロードの中身が変更されたことを確認
 - 「TTLExceededのヘッダ」 + 「送信時のIPヘッダとICMPヘッダ」

0000	72	e2	1d	05	65	5a	ac	c4	bd	53	d5	fb	08	00	45	c0
0010	00	3c	f0	d2	00	00	40	01	01	db	c0	a8	03	01	c0	a8
0020	03	02	0b	00	f4	ff	00	00	00	00	45	00	00	20	d7	52
0030	00	00	01	01	0e	d1	c0	a8	03	02	08	08	08	08	08	00
0040	38	d3	d7	52	00	00	74	65	73	74						

チェックサム

WireSharkのTTLExceeded ICMPヘッダ

IPヘッダの「パケット長」が変更されている

0b	00	f4	ff	00	00	00	00	45	00	20	00	d7	52	00	00	01	01	0e	d1	c0	a8	03
02	08	08	08	08	08	08	00	38	d3	d7	52	00	00	74	65	73	74					

今回実装したスクリプトで解析したTTLExceeded ICMPヘッダ

これに関しては
原因がわからないです。