# wireshark dissector with lua

2013/06/05

@team_eririn

https://www.ainoniwa.net/ssp/

# 資料概要

- Luaプラグインを用いて、Wiresharkにデコード可能なプロトコルを追加する手法について記載します。

- 今回は、ネットワークベンチマークソフトウェアである、 iperf パケットを題材にします。

# 想定環境

- OS
  - Windows XP, Vista, 7
- Wireshark
  - Version : 1.6.x or 1.8.x
  - http://www.wireshark.org/download.html
- iperf
  - Version : 2.0.5
  - http://sourceforge.jp/projects/sfnet_iperf/

# 補記

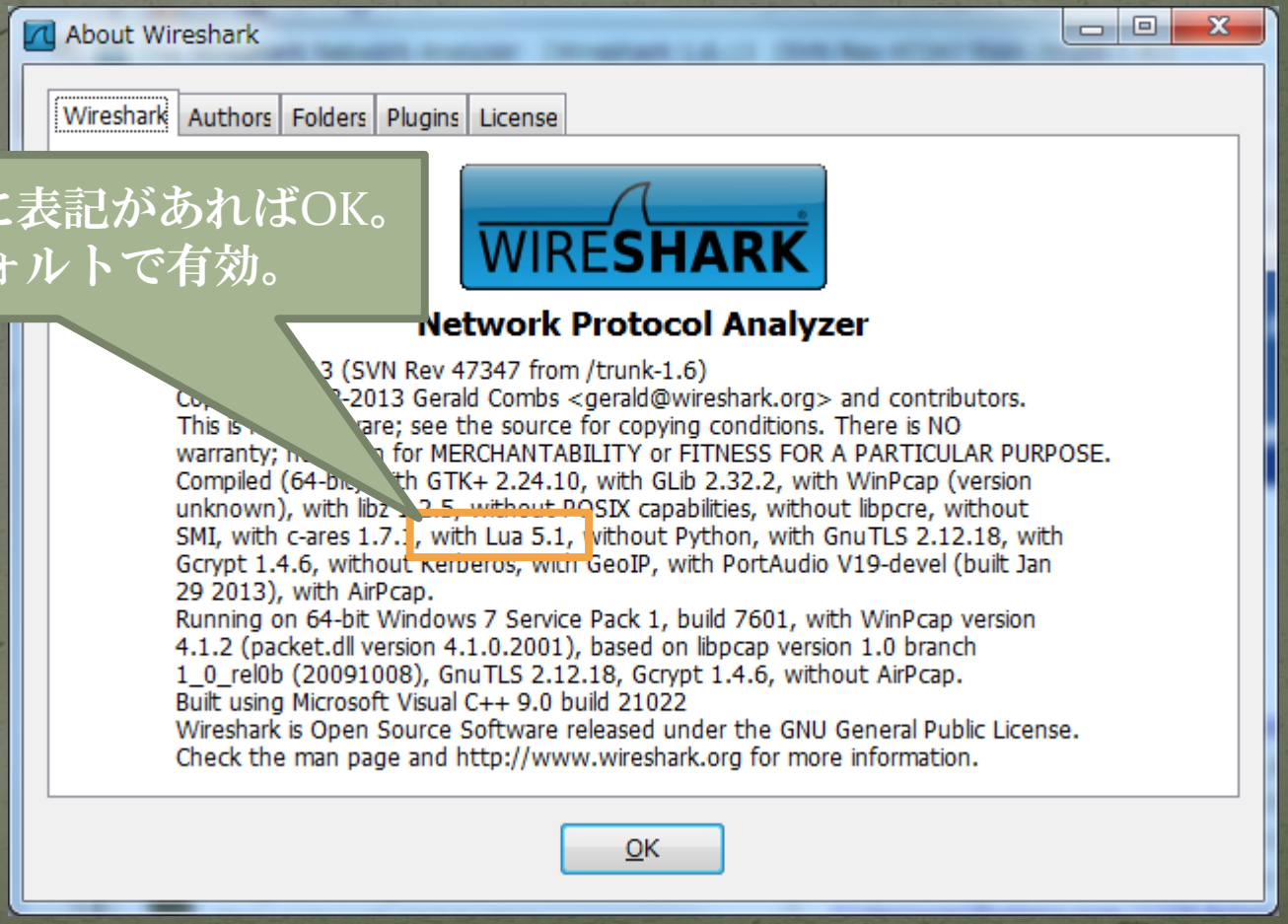
- Windowsを主な対象として記載しているものの、LuaによるPlugin作成に関しては、Wiresharkを利用するWindows以外の各種OSで共通です

- 資料中ではWireshark 1.6系をベースに記載しているため、1.8系と細部が異なる可能性があります

# プラグイン作成の前に

# Lua enable ?

- Help -> About Wireshark




"with Lua 5.1"のように表記があればOK。
Windowsはデフォルトで有効。

# Lua scriptの読み込み方法

AもしくはBの方法を選択します。
本資料はAにて進行します。

A) ${wireshark_install_dir}/plugins/${wireshark_version}/ に *.lua ファイルを置く。
- 例)
  C:¥Program Files¥Wireshark¥plugins¥1.6.14¥iperf.lua
- ※wiresharkをバージョンアップすると、${wireshark_version}は消えるので、作成中などは注意

B) ${wireshark_install_dir}/init.lua に、以下のように書き加える。
- dofile(DATA_DIR..”your_script.lua”)

# 補記

${wireshark_install_dir}/init.lua の中で、

- disable_lua = true

という記載がある場合は、以下のように変更します。

- disable_lua = false

または

- -- disable_lua = true

古いWiresharkの場合（もしくはLinuxパッケージのポリシーに基づく場合）は、上記の記載が残っている可能性があります。

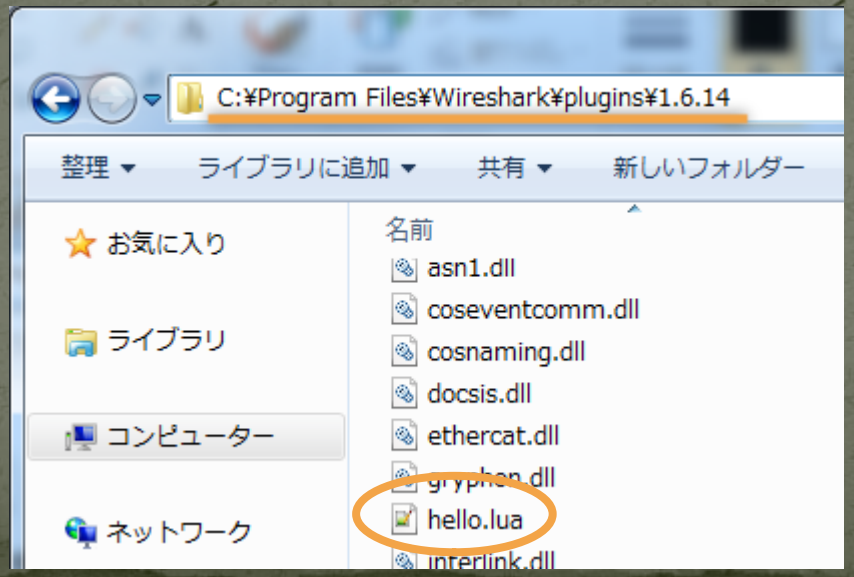
# 試しに読み込めるか確かめよう

1. 作ります（※）。
   - hello.lua

```
local hello_lua = "Hello."
```

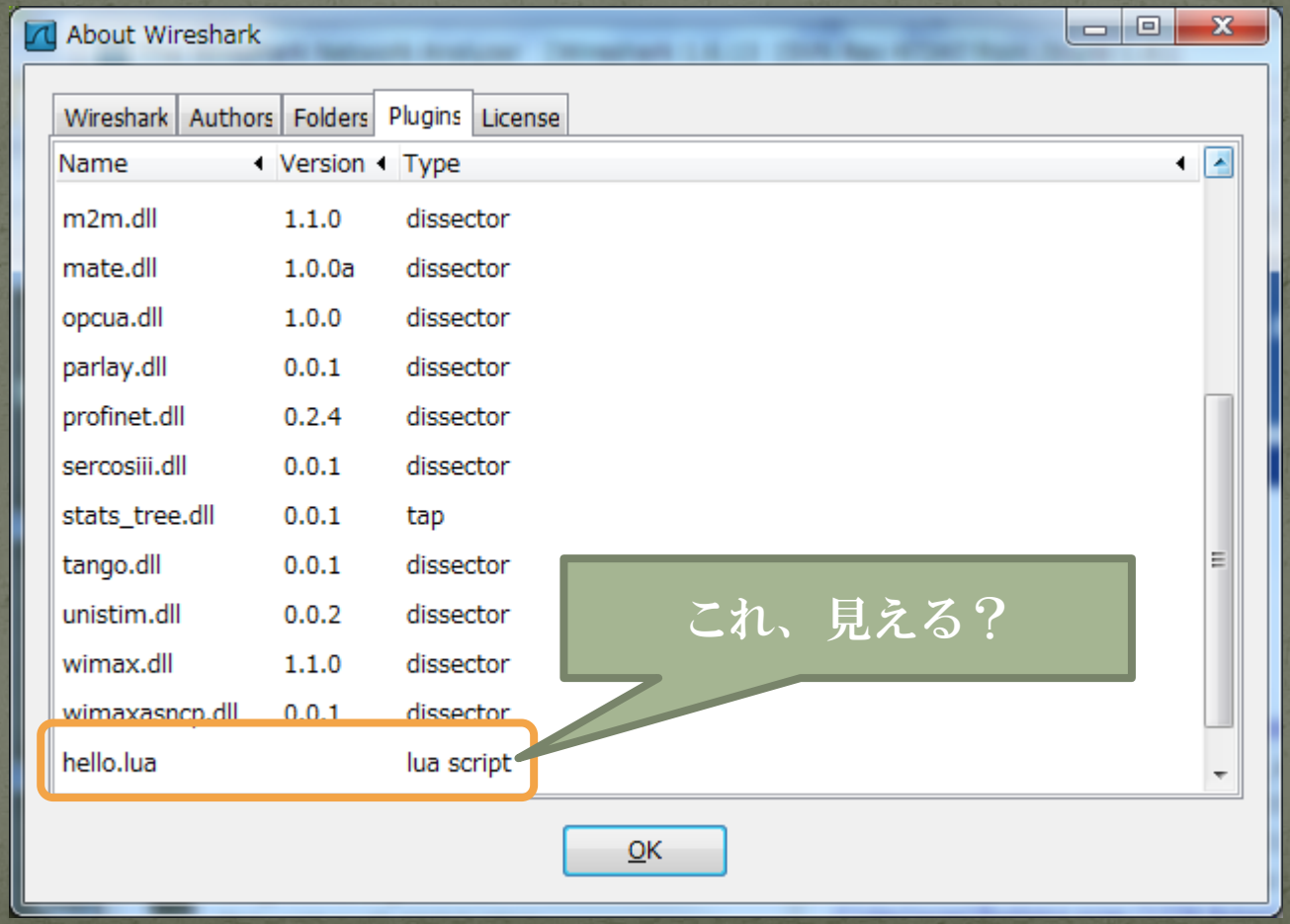
※Shift-JISでないと動作しない可能性があります

2. 置きます。



# wiresharkを起動

- Help -> About Wireshark -> Plugins



# プラグイン作成例

# では実際にプラグイン作成へ

- 必要となる知識
  - デコード対象プロトコルの構造体に対する知識
    - 今回はiperfになります
  - lua プラグインのwiresharkでの作法

# ('A`)

※本資料では iperf プロトコルを全部解剖して解説するなんてことはしません。
極々限られた範囲のみをデコード出来るようにして、とりあえずの達成感を得ましょう。

# まずは検体の取得

1. wiresharkを起動して適当なインタフェースでキャプチャ開始
2. iperf -c 192.168.122.150 -u
3. wiresharkで検体を保存

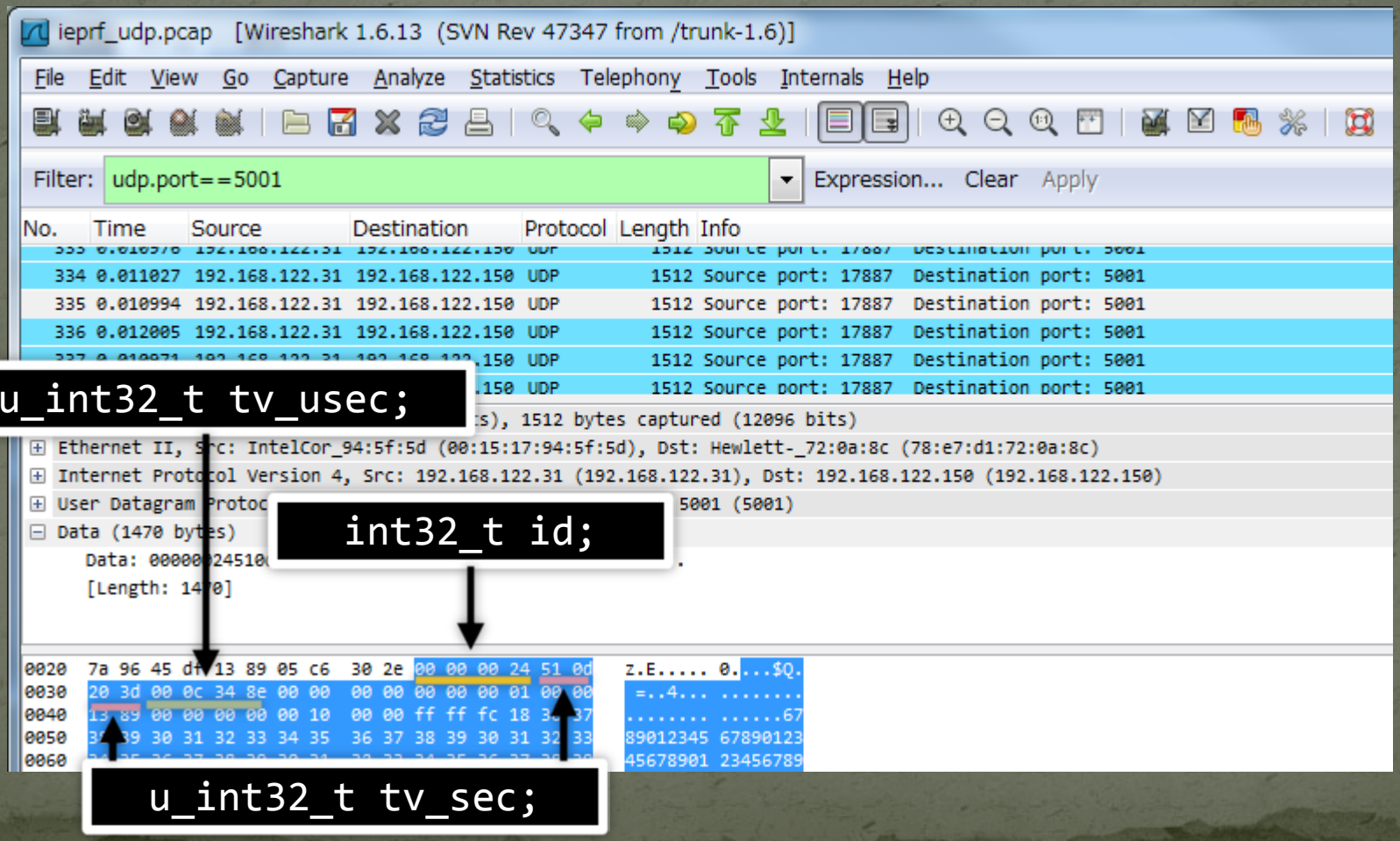
# 対象とするiperfプロトコル部位

- 以下の条件で進行します。
  - UDPのペイロード先頭部位のみ。つまり以下の部分。
    - iperf-2.0.5¥include¥Settings.hpp // line : 292-304

```
// used to reference the 4 byte ID number we place in UDP datagrams
// use int32_t if possible, otherwise a 32 bit bitfield (e.g. on J90)
typedef struct UDP_datagram {
#ifdef HAVE_INT32_T
    int32_t id;
    u_int32_t tv_sec;
    u_int32_t tv_usec;
#else
    signed   int id      : 32;
    unsigned int tv_sec  : 32;
    unsigned int tv_usec : 32;
#endif
} UDP_datagram;
```

# つまりどこ？

- この辺（Hex View参照）



# 暫定結論

- iperf のデフォルトポート番号は5001(TCP/UDP両方)
- iperf のデータグラムの先頭は、以下のフォーマット

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| id | | | | tv_sec | | | | tv_usec | | | | | | | |

- iperf のプロトコルフォーマットを少しかじったところで、 wireshark の lua プラグインのお作法へ

# Protocol dissector step

1. プロトコルの宣言
2. フィールドの宣言
3. 実際のデータ処理
4. プロトコルツリーへの追加
5. プロトコルの登録

# コードで見る概要

- 基本のステップはこれだけ。（--はコメントアウト）

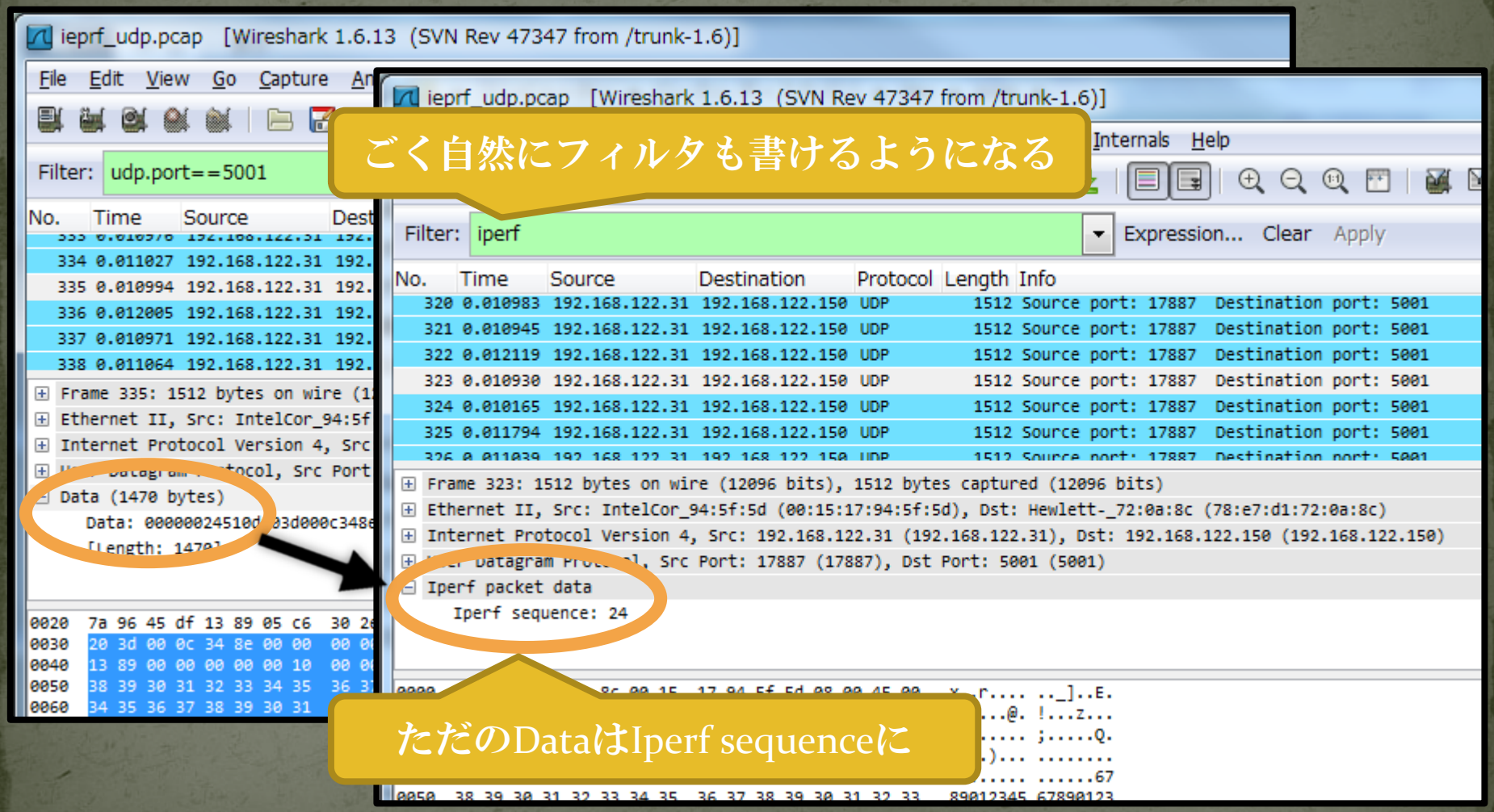
```
-- *** Step 1 : プロトコルの宣言 ***
iperf_proto = Proto("iperf","Iperf UDP packet")
-- *** Step 2 : フィールドの宣言 ***
iperf_seq_F = ProtoField.uint32("iperf.seq", "Iperf sequence")
iperf_proto.fields = {iperf_seq_F}

function iperf_proto.dissector(buffer,pinfo,tree)
    -- *** Step 3 : 実際のデータ処理 ***
    local iperf_seq_range = buffer(0,4)
    local iperf_seq = iperf_seq_range:uint()
    -- *** Step 4 : プロトコルツリーに追加 ***
    local subtree = tree:add(iperf_proto, buffer(), "Iperf packet data")
    subtree:add(iperf_seq_F, iperf_seq_range, iperf_seq)
end

-- *** Step 5 :プロトコルの登録 ***
DissectorTable.get("udp.port"):add(5001, iperf_proto)
```

- ファイルに保存してpluginsディレクトリに置く。

# ここまでだけでこうなる



# あとは、肉付けしていく

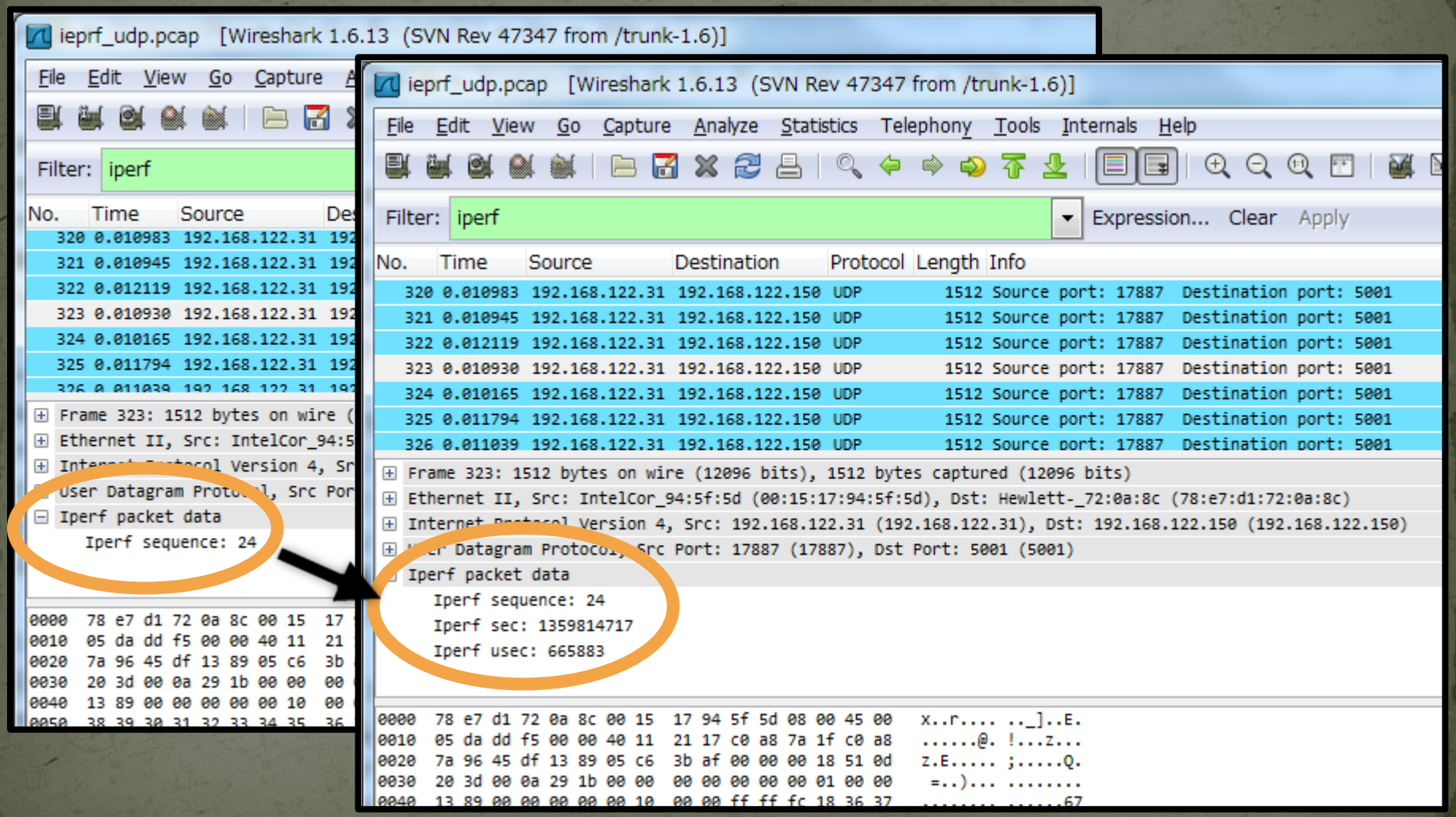
- さっきの基本ステップに書き足していく（赤字部）。

```
-- *** Step 1 : プロトコルの宣言 ***
iperf_proto = Proto("iperf","Iperf UDP packet")
-- *** Step 2 : フィールドの宣言 ***
iperf_seq_F  = ProtoField.uint32("iperf.seq",  "Iperf sequence")
iperf_sec_F  = ProtoField.uint32("iperf.sec",  "Iperf sec")
iperf_usec_F = ProtoField.uint32("iperf.usec", "Iperf usec")
iperf_proto.fields = {iperf_seq_F, iperf_sec_F, iperf_usec_F }

function iperf_proto.dissector(buffer,pinfo,tree)
    -- *** Step 3 : 実際のデータ処理 ***
    local iperf_seq_range  = buffer(0,4)
    local iperf_sec_range  = buffer(4,4)
    local iperf_usec_range = buffer(8,4)
    local iperf_seq  = iperf_seq_range:uint()
    local iperf_sec  = iperf_sec_range:uint()
    local iperf_usec = iperf_usec_range:uint()
    -- *** Step 4 : プロトコルツリーに追加 ***
    local subtree = tree:add(iperf_proto, buffer(), "Iperf packet data")
    subtree:add(iperf_seq_F,  iperf_seq_range,  iperf_seq)
    subtree:add(iperf_sec_F,  iperf_sec_range,  iperf_sec)
    subtree:add(iperf_usec_F, iperf_usec_range, iperf_usec)
end

-- *** Step 5 :プロトコルの登録 ***
DissectorTable.get("udp.port"):add(5001, iperf_proto)
```

# さらにこうなる


ieprf_udp.pcap [Wireshark 1.6.13 (SVN Rev 47347 from /trunk-1.6)]
File Edit View Go Capture
Filter: iperf
No. Time Source
320 0.010983 192.168.122.31
321 0.010945 192.168.122.31
322 0.012119 192.168.122.31
323 0.010930 192.168.122.31
324 0.010165 192.168.122.31
325 0.011794 192.168.122.31
Frame 323: 1512 bytes on wire
Ethernet II, Src: IntelCor_94:5
Iperf packet data
Iperf sequence: 24
0000 78 e7 d1 72 0a 8c 00 15
0010 05 da dd f5 00 00 40 11
0020 7a 96 45 df 13 89 05 c6
0030 20 3d 00 0a 29 1b 00 00
0040 13 89 00 00 00 00 00 10
ieprf_udp.pcap [Wireshark 1.6.13 (SVN Rev 47347 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: iperf
Expression... Clear Apply
No. Time Source Destination Protocol Length Info
320 0.010983 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
321 0.010945 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
322 0.012119 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
323 0.010930 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
324 0.010165 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
325 0.011794 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
326 0.011039 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
Frame 323: 1512 bytes on wire (12096 bits), 1512 bytes captured (12096 bits)
Ethernet II, Src: IntelCor_94:5f:5d (00:15:17:94:5f:5d), Dst: Hewlett-_72:0a:8c (78:e7:d1:72:0a:8c)
Internet Protocol Version 4, Src: 192.168.122.31 (192.168.122.31), Dst: 192.168.122.150 (192.168.122.150)
Src Port: 17887 (17887), Dst Port: 5001 (5001)
Iperf packet data
Iperf sequence: 24
Iperf sec: 1359814717
Iperf usec: 665883
0000 78 e7 d1 72 0a 8c 00 15 17 94 5f 5d 08 00 45 00 x..r.... .._]..E.
0010 05 da dd f5 00 00 40 11 21 17 c0 a8 7a 1f c0 a8 ......@. !...z...
0020 7a 96 45 df 13 89 05 c6 3b af 00 00 00 18 51 0d z.E..... ;.....Q.
0030 20 3d 00 0a 29 1b 00 00 00 00 00 00 00 01 00 00 =..)... ........

# ('∀`)

- iperf(UDP)のシーケンス番号、タイムスタンプがデコードできるようになったよ！

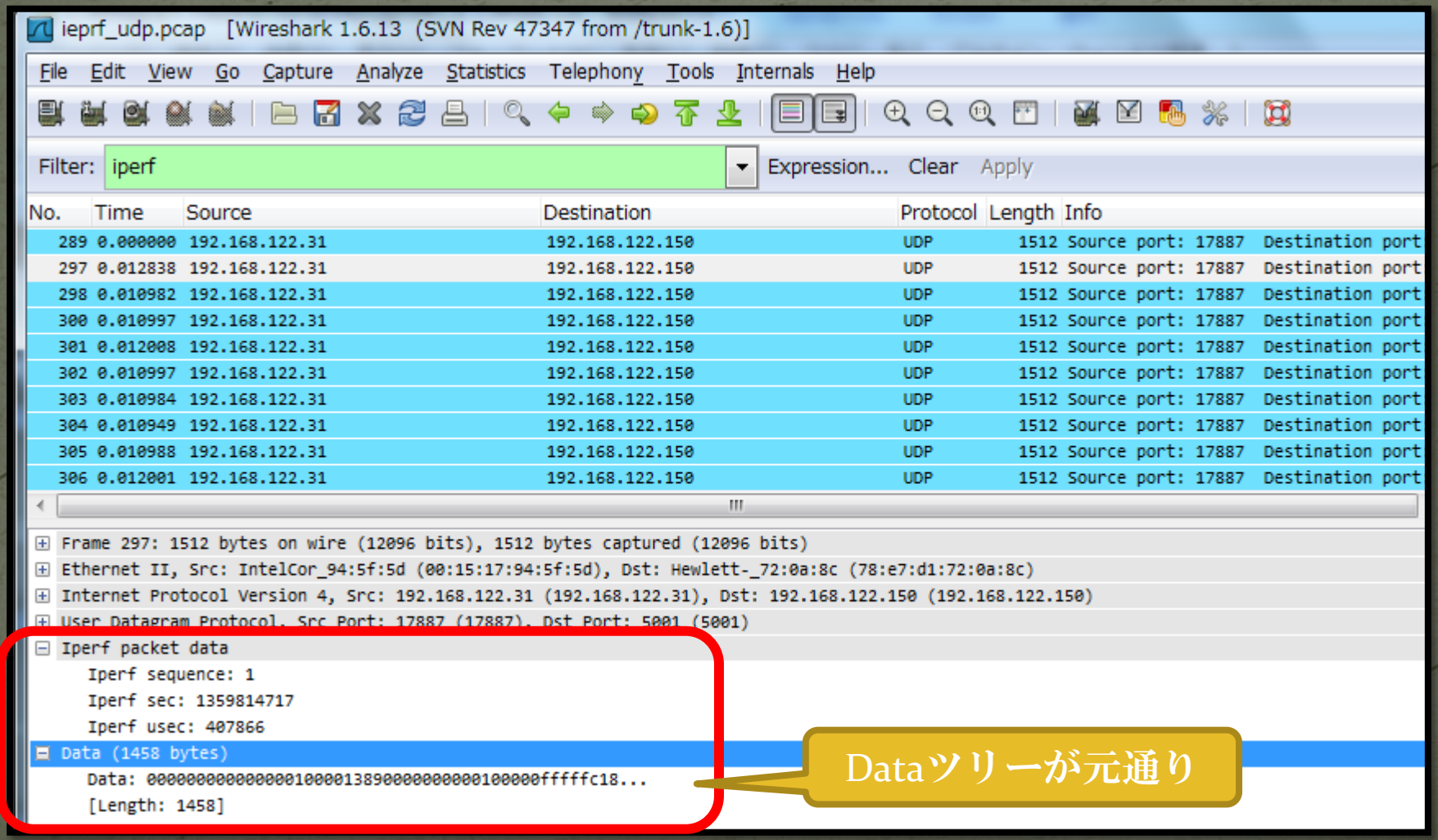
# Q&A

# Q. Dataはどこいっちゃったの？

- A. Dataをツリーに追加していないので見えません
  - 必要であれば、WiresharkのData Dissectorに再度残ったデータを入れることで、一般的な表現形に戻すことができます。
  - 赤字部を追加。

```
--前略
    -- *** Step 3 : 実際のデータ処理 ***
    local iperf_seq_range  = buffer(0,4)
    local iperf_sec_range  = buffer(4,4)
    local iperf_usec_range = buffer(8,4)
    local iperf_seq  = iperf_seq_range:uint()
    local iperf_sec  = iperf_sec_range:uint()
    local iperf_usec = iperf_usec_range:uint()
    -- *** Step 4 : プロトコルツリーに追加 ***
    local subtree = tree:add(iperf_proto, buffer(), "Iperf packet data")
    subtree:add(iperf_seq_F,  iperf_seq_range,  iperf_seq)
    subtree:add(iperf_sec_F,  iperf_sec_range,  iperf_sec)
    subtree:add(iperf_usec_F, iperf_usec_range, iperf_usec)
    Dissector.get("data"):call(buffer(12,buffer:len()-12):tvb(), pinfo, tree)
--以下略
```

# 出力例


ieprf_udp.pcap [Wireshark 1.6.13 (SVN Rev 47347 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: iperf
Expression... Clear Apply
No. Time Source Destination Protocol Length Info
289 0.000000 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port
297 0.012838 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port
298 0.010982 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port
300 0.010997 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port
301 0.012008 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port
302 0.010997 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port
303 0.010984 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port
304 0.010949 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port
305 0.010988 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port
306 0.012001 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port
Frame 297: 1512 bytes on wire (12096 bits), 1512 bytes captured (12096 bits)
Ethernet II, Src: IntelCor_94:5f:5d (00:15:17:94:5f:5d), Dst: Hewlett-_72:0a:8c (78:e7:d1:72:0a:8c)
Internet Protocol Version 4, Src: 192.168.122.31 (192.168.122.31), Dst: 192.168.122.150 (192.168.122.150)
User Datagram Protocol, Src Port: 17887 (17887), Dst Port: 5001 (5001)
Iperf packet data
Iperf sequence: 1
Iperf sec: 1359814717
Iperf usec: 407866
Data (1458 bytes)
Data: 0000000000000001000013890000000000100000fffffc18...
[Length: 1458]
Dataツリーが元通り

# Q. Bit Fieldを取得できる？

- A. できます。
  - 単にbitの値を切り出したい場合
    - local iperf_seq  = iperf_seq_range:bitfield(30,2)
  - Treeに追加するためのフィールド宣言
    - iperf_flag_bit_F   = ProtoField.uint32("iperf.flag_bit","bit", base.HEX, None, 0x00008000)
  - Treeに実際追加する場合
    - subflagatree:add(iperf_flag_bit_F, iperf_flags_range, iperf_flags)
    - 第一引数のフィールド指定以外は、参照元の値を使う

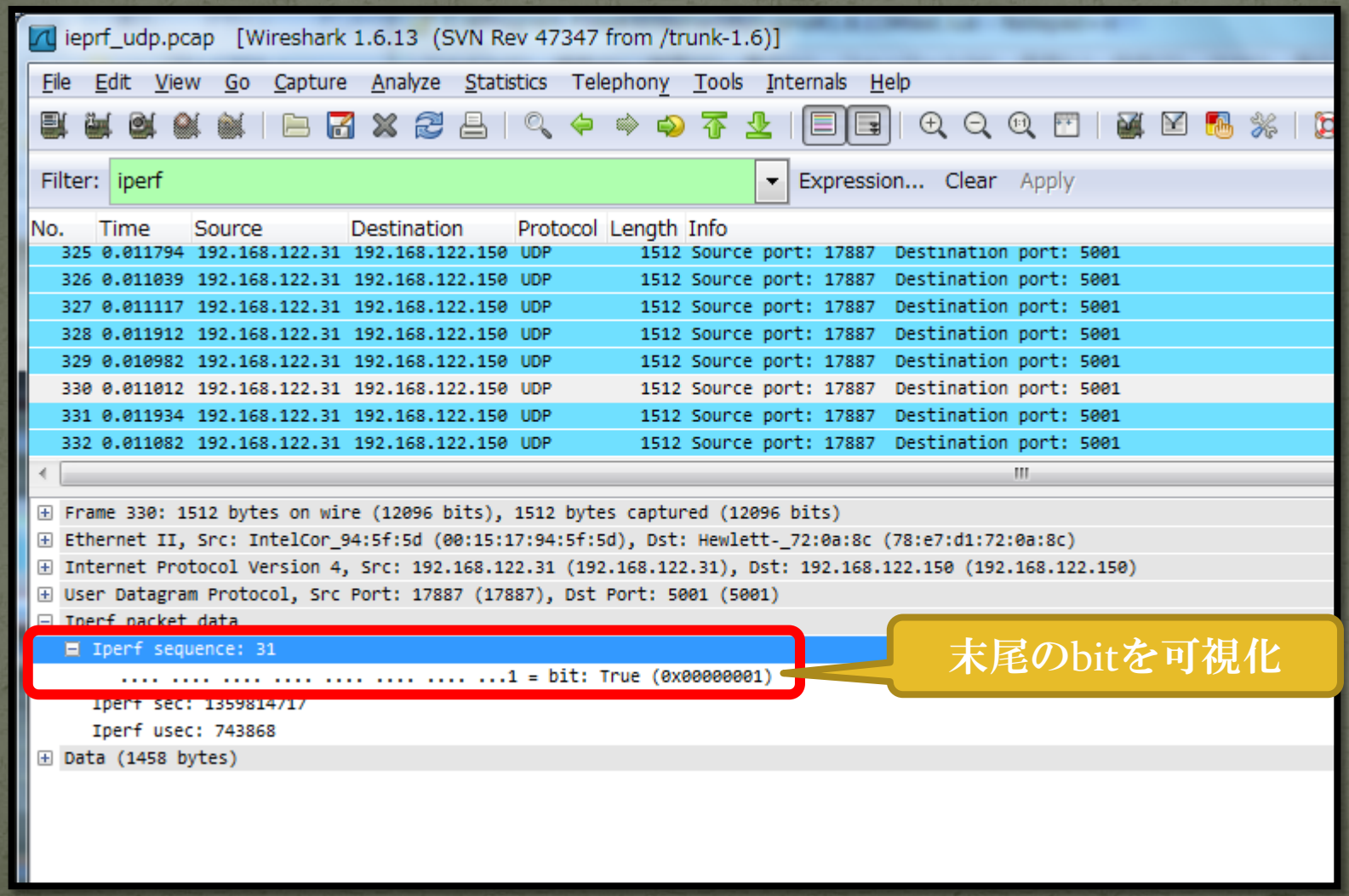
# 例えばこう

```
-- *** Step 1 : プロトコルの宣言 ***
iperf_proto = Proto("iperf","Iperf UDP packet")
-- *** Step 2 : フィールドの宣言 ***
iperf_seq_F  = ProtoField.uint32("iperf.seq",  "Iperf sequence")
iperf_sec_F  = ProtoField.uint32("iperf.sec",  "Iperf sec")
iperf_usec_F = ProtoField.uint32("iperf.usec", "Iperf usec")
local VALS_BOOL = {[0] = "False", [1] = "True"}
iperf_flag_bit_F = ProtoField.uint32("iperf.flag_bit","bit", base.HEX, VALS_BOOL, 0x00000001)
iperf_proto.fields = {iperf_seq_F, iperf_sec_F, iperf_usec_F, iperf_flag_bit_F }

function iperf_proto.dissector(buffer,pinfo,tree)
    -- *** Step 3 : 実際のデータ処理 ***
    local iperf_seq_range  = buffer(0,4)
    local iperf_sec_range  = buffer(4,4)
    local iperf_usec_range = buffer(8,4)
    local iperf_seq  = iperf_seq_range:uint()
    local iperf_sec  = iperf_sec_range:uint()
    local iperf_usec = iperf_usec_range:uint()
    -- *** Step 4 : プロトコルツリーに追加 ***
    local subtree = tree:add(iperf_proto, buffer(), "Iperf packet data")
    --subtree:add(iperf_seq_F,  iperf_seq_range,  iperf_seq)
    local subflagatree = subtree:add(iperf_seq_F, iperf_seq_range, iperf_seq)
    subflagatree:add(iperf_flag_bit_F, iperf_seq_range, iperf_seq)
    subtree:add(iperf_sec_F,  iperf_sec_range,  iperf_sec)
    subtree:add(iperf_usec_F, iperf_usec_range, iperf_usec)
    Dissector.get("data"):call(buffer(12,buffer:len()-12):tvb(), pinfo, tree)
end

-- *** Step 5 :プロトコルの登録 ***
DissectorTable.get("udp.port"):add(5001, iperf_proto)
```

# で、こうなる


ieprf_udp.pcap [Wireshark 1.6.13 (SVN Rev 47347 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: iperf
Expression... Clear Apply
No. Time Source Destination Protocol Length Info
325 0.011794 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
326 0.011039 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
327 0.011117 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
328 0.011912 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
329 0.010982 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
330 0.011012 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
331 0.011934 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
332 0.011082 192.168.122.31 192.168.122.150 UDP 1512 Source port: 17887 Destination port: 5001
Frame 330: 1512 bytes on wire (12096 bits), 1512 bytes captured (12096 bits)
Ethernet II, Src: IntelCor_94:5f:5d (00:15:17:94:5f:5d), Dst: Hewlett-_72:0a:8c (78:e7:d1:72:0a:8c)
Internet Protocol Version 4, Src: 192.168.122.31 (192.168.122.31), Dst: 192.168.122.150 (192.168.122.150)
User Datagram Protocol, Src Port: 17887 (17887), Dst Port: 5001 (5001)
Iperf packet data
Iperf sequence: 31
.... .... .... .... .... .... .... ...1 = bit: True (0x00000001)
Iperf sec: 1359814717
Iperf usec: 743868
Data (1458 bytes)
末尾のbitを可視化

# Q. Expert Infosとか使える？

- A. 使えます。
  - Treeにアイテムを追加する際は、Itemオブジェクト、SubTreeを追加する際は、Treeオブジェクトが返ってくるので、それぞれに設定する
  - Itemの場合
    - bit_item:add_expert_info(PI_MALFORMED, PI_WARN, 'seq bit on')
  - Treeの場合
    - subtree:set_expert_flags(PI_MALFORMED, PI_WARN)
  - 詳しくはこの辺のドキュメントを見る。
    - http://www.wireshark.org/docs/wsug_html_chunked/lua_module_Tree.html#lua_fn_treeitem_set_expert_flags__group___severity__

# 例) 特定bitが1だったら登録する

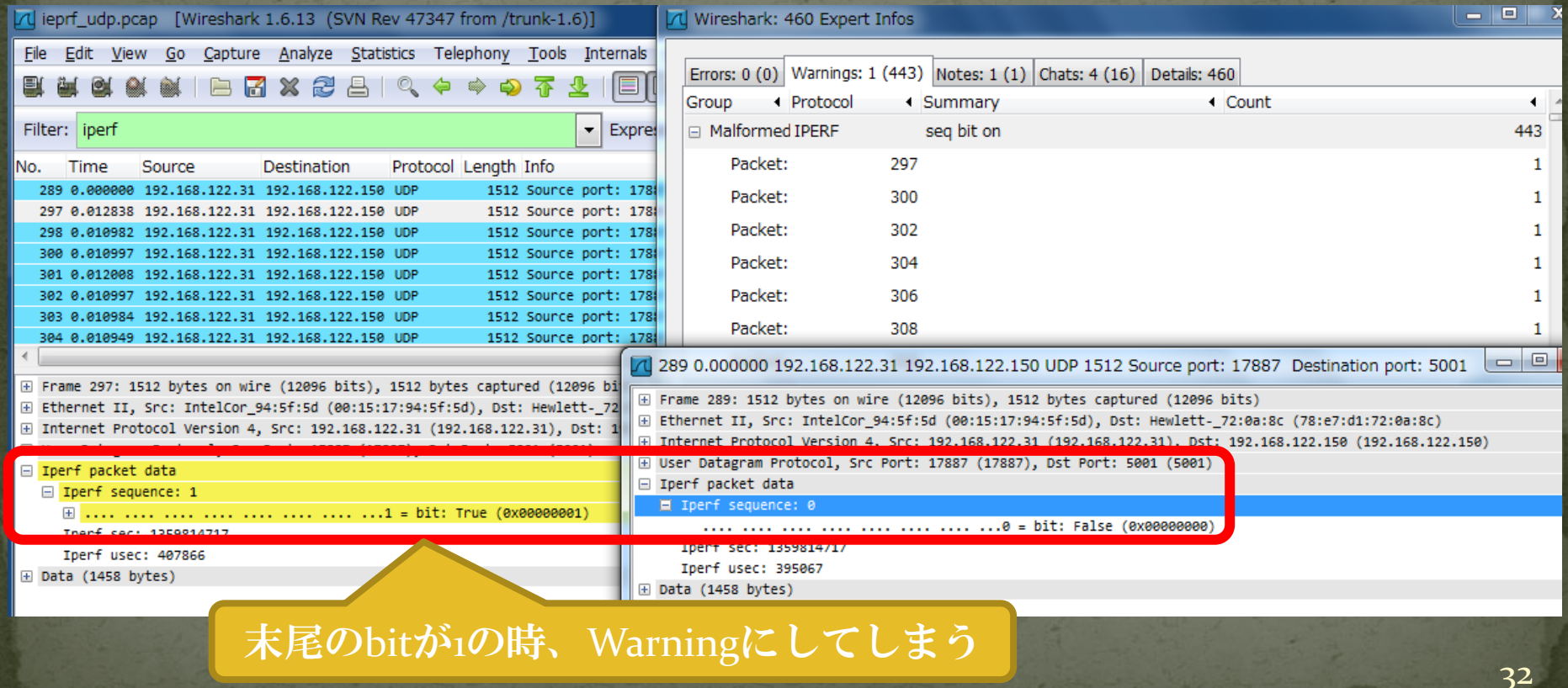
```
iperf_proto = Proto("iperf","Iperf UDP packet")
iperf_seq_F  = ProtoField.uint32("iperf.seq",  "Iperf sequence")
iperf_sec_F  = ProtoField.uint32("iperf.sec",  "Iperf sec")
iperf_usec_F = ProtoField.uint32("iperf.usec", "Iperf usec")
iperf_flag_bit_F = ProtoField.uint32("iperf.flag_bit","bit", base.HEX, None, 0x00000001)
iperf_proto.fields = {iperf_seq_F, iperf_sec_F, iperf_usec_F, iperf_flag_bit_F }

function iperf_proto.dissector(buffer,pinfo,tree)
    local iperf_seq_range  = buffer(0,4)
    local iperf_sec_range  = buffer(4,4)
    local iperf_usec_range = buffer(8,4)
    local iperf_seq  = iperf_seq_range:uint()
    local iperf_bit  = iperf_seq_range:bitfield(31,1)
    local iperf_sec  = iperf_sec_range:uint()
    local iperf_usec = iperf_usec_range:uint()
    local subtree = tree:add(iperf_proto, buffer(), "Iperf packet data")
    local subflagatree = subtree:add(iperf_seq_F, iperf_seq_range, iperf_seq)
    bit_item = subflagatree:add(iperf_flag_bit_F, iperf_seq_range, iperf_seq)
    if iperf_bit == 1 then
        subtree:set_expert_flags(PI_MALFORMED, PI_WARN)
        bit_item:add_expert_info(PI_MALFORMED, PI_WARN, 'seq bit on')
    end
    subtree:add(iperf_sec_F,  iperf_sec_range,  iperf_sec)
    subtree:add(iperf_usec_F, iperf_usec_range, iperf_usec)
end

DissectorTable.get("udp.port"):add(5001, iperf_proto)
```

# こうなる

- Expert Infosは、Analyze -> Expert Info Compositeから表示できます



# 参考文献


- Chapter 11. Lua Support in Wireshark
  - http://www.wireshark.org/docs/wsug_html_chunked/wsluarm.html