柔軟で規模追従可能な
トラフィック解析基盤SF-TAPついて

北陸先端科学技術大学院大学
高信頼ネットワークイノベーションセンター
井上朋哉

2016年01月12日

自己紹介

井上朋哉

- 前職
  - PFU
  - ネットワークアプライアンスのＨＷ開発業務等
  - Clwit
  - ネットワークソフトウェアの開発業務等
- 現職
  - 北陸先端科学技術大学院大学
高信頼ネットワークインベーションセンター
  - 情報通信研究機構
北陸StarBED技術センター

- 研究
  - パケット制御，テストベッド，名前解決，P2P

自己紹介

井上朋哉
- PFU
  - ネットワークア
- Clwit
  - ネットワークソ

最近はこんなのやってました．

解決，P2P等

SF-TAP

Scalable and Flexible Traffic Analysis Platform

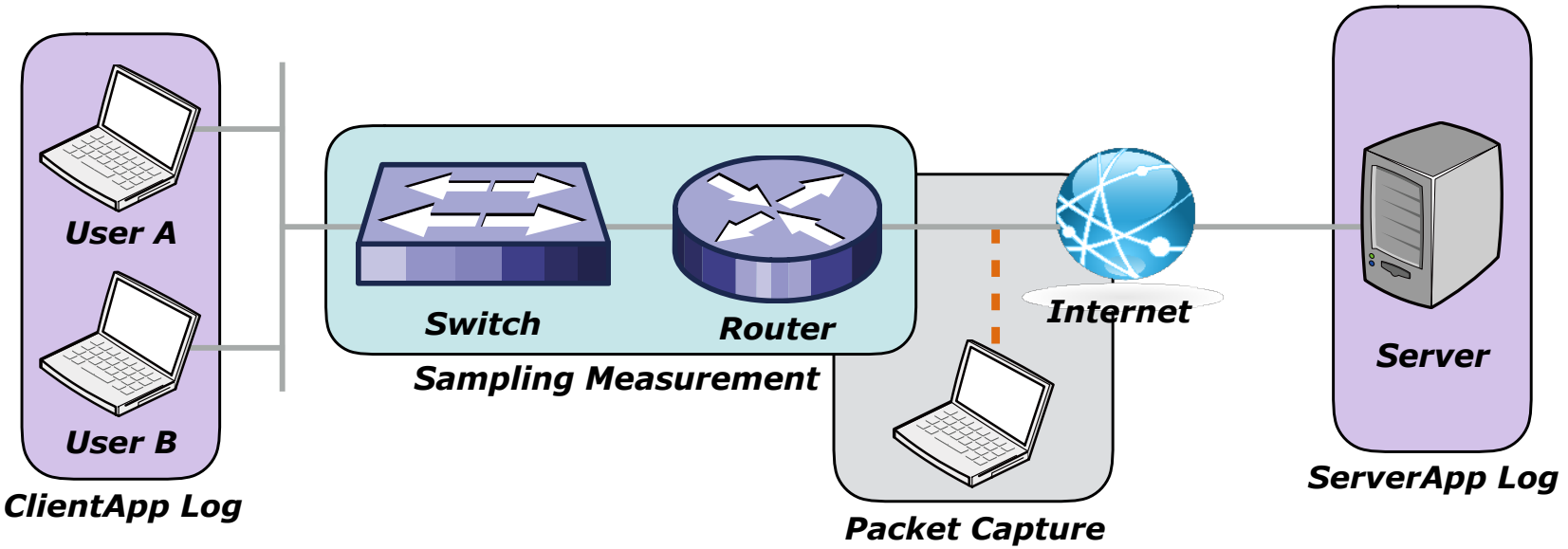
- 開発メンバー
  - 高野祐輝，三浦良介，安田真悟，明石邦夫，井上朋哉
  - JAIST篠田研究室の学生とOBのグループ

- Source Code
  - https://github.com/SF-TAP/

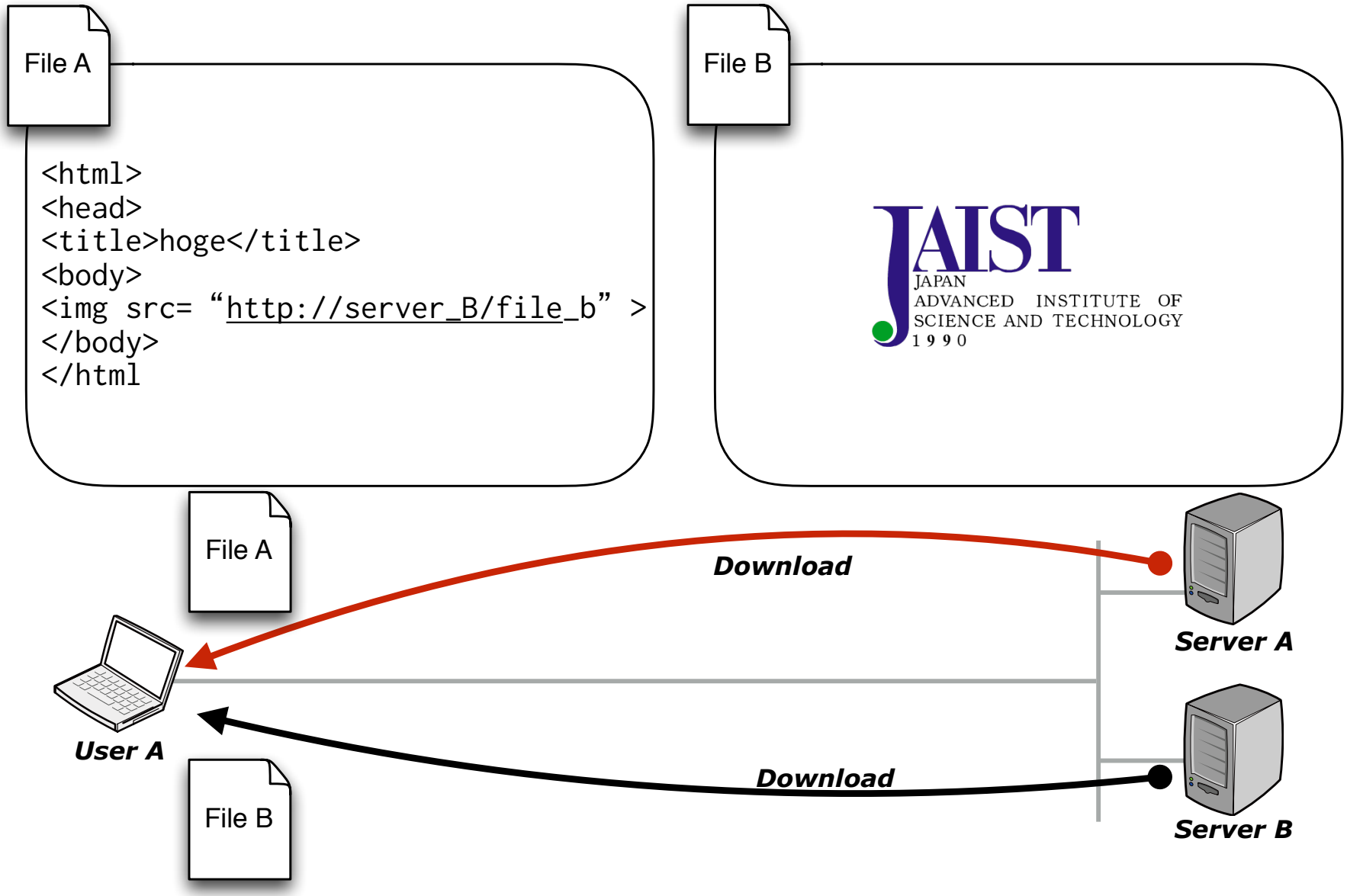
- 今までの活動
  - WIDE, INTEROP2015, LISA2015等で発表

開発背景

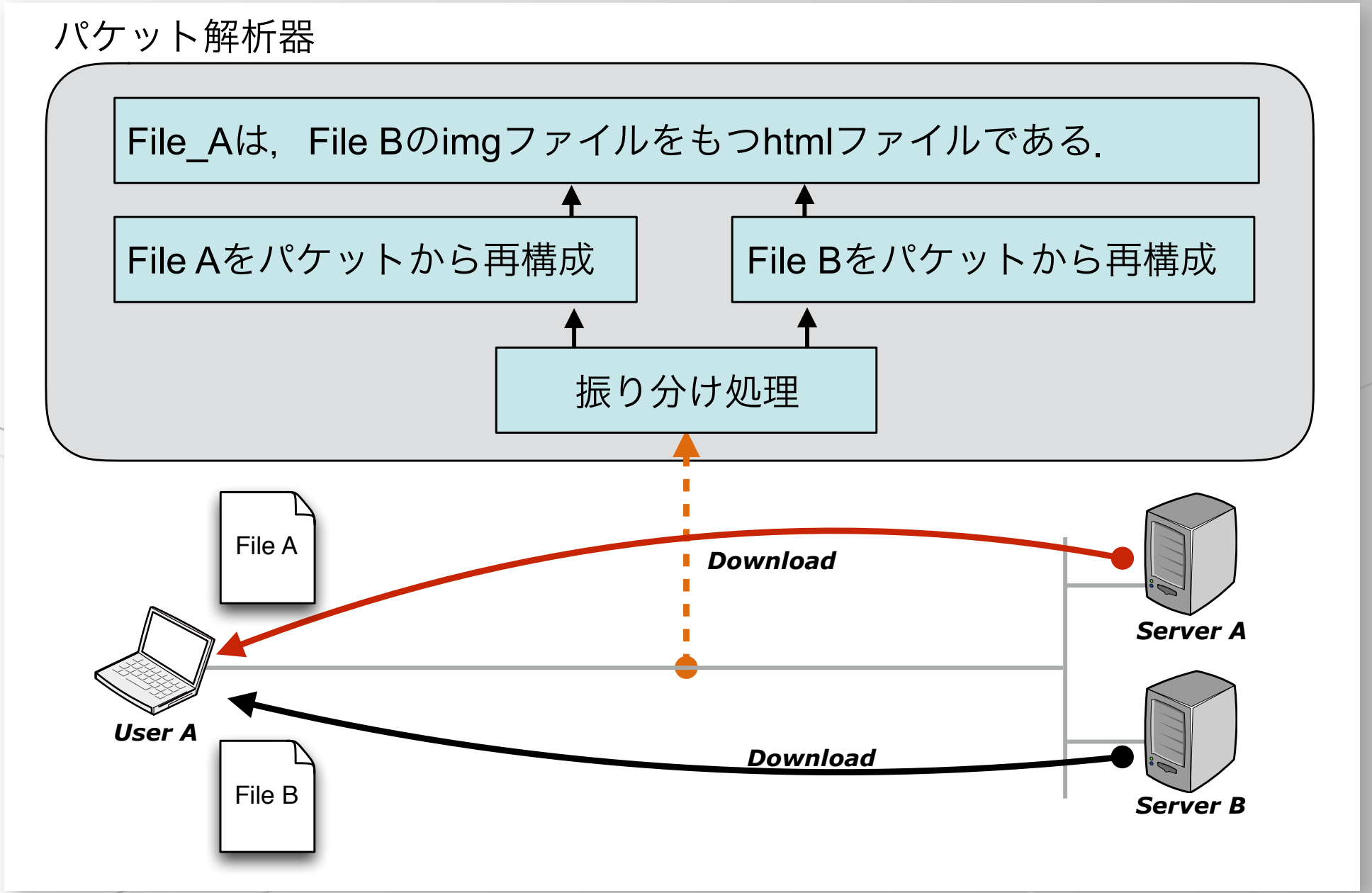
- 研究開発でのトラフィックのモニタや解析
  - どんな通信が行われているか？間違った通信をおこなっていないか？等の確認がおこないたい．
- 方法
  - クライアントやサーバのアプリケーションログ
  - 転送機器(L2/L3)の統計情報やサンプリング計測
  - パケットキャプチャによるトラフィック解析



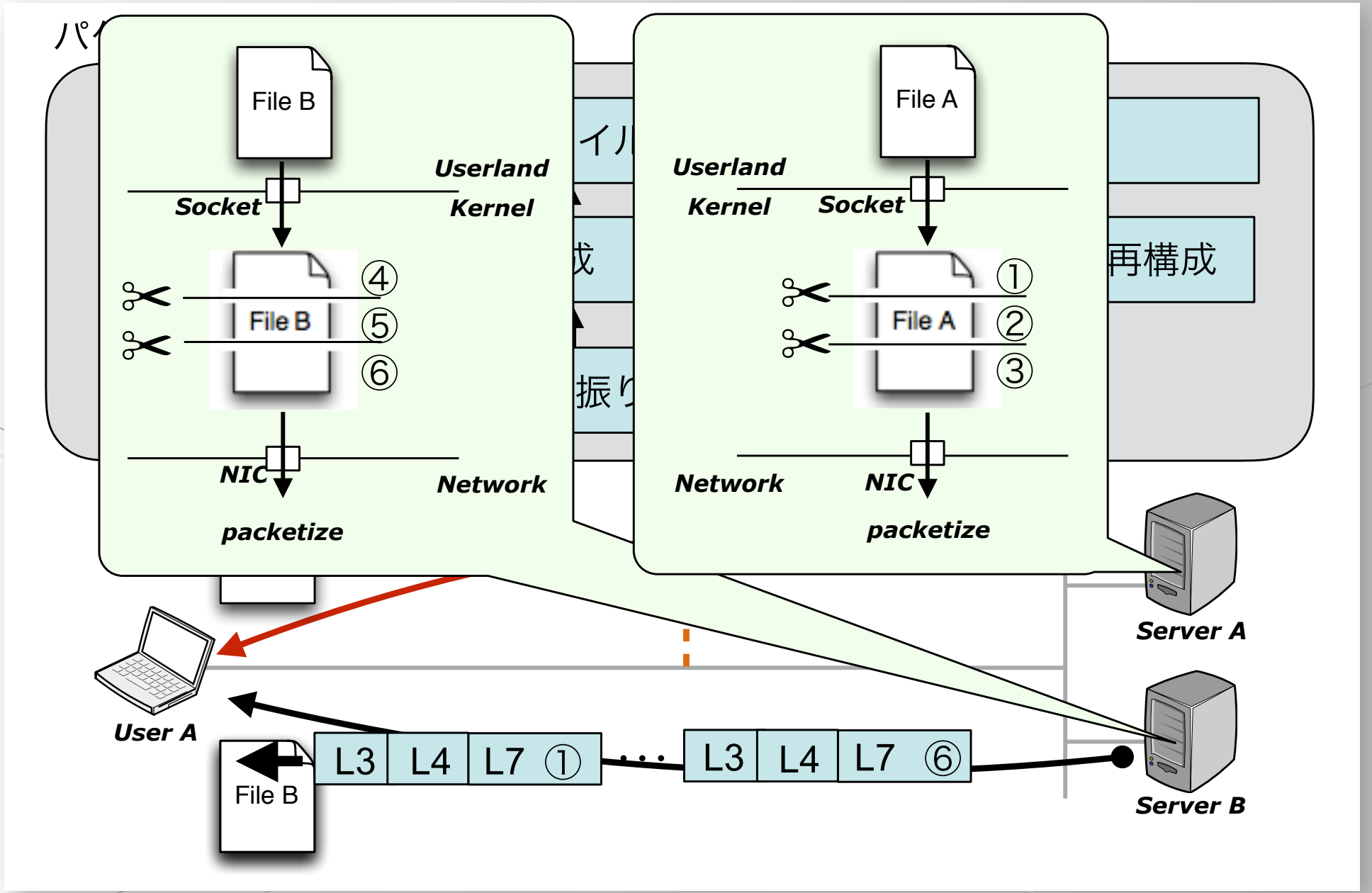
L7トラフィックを解析？



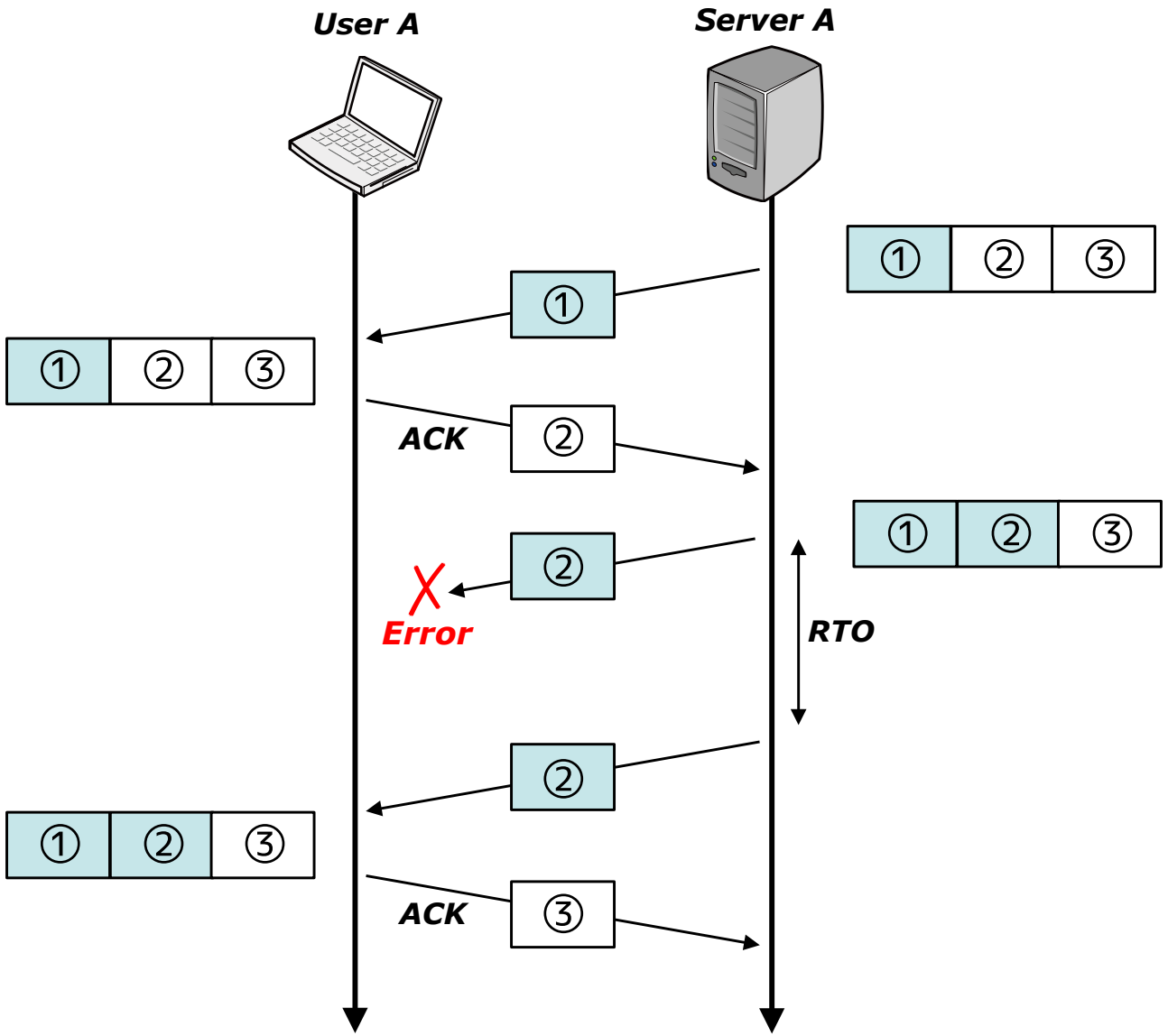
L7トラフィクを解析するには？



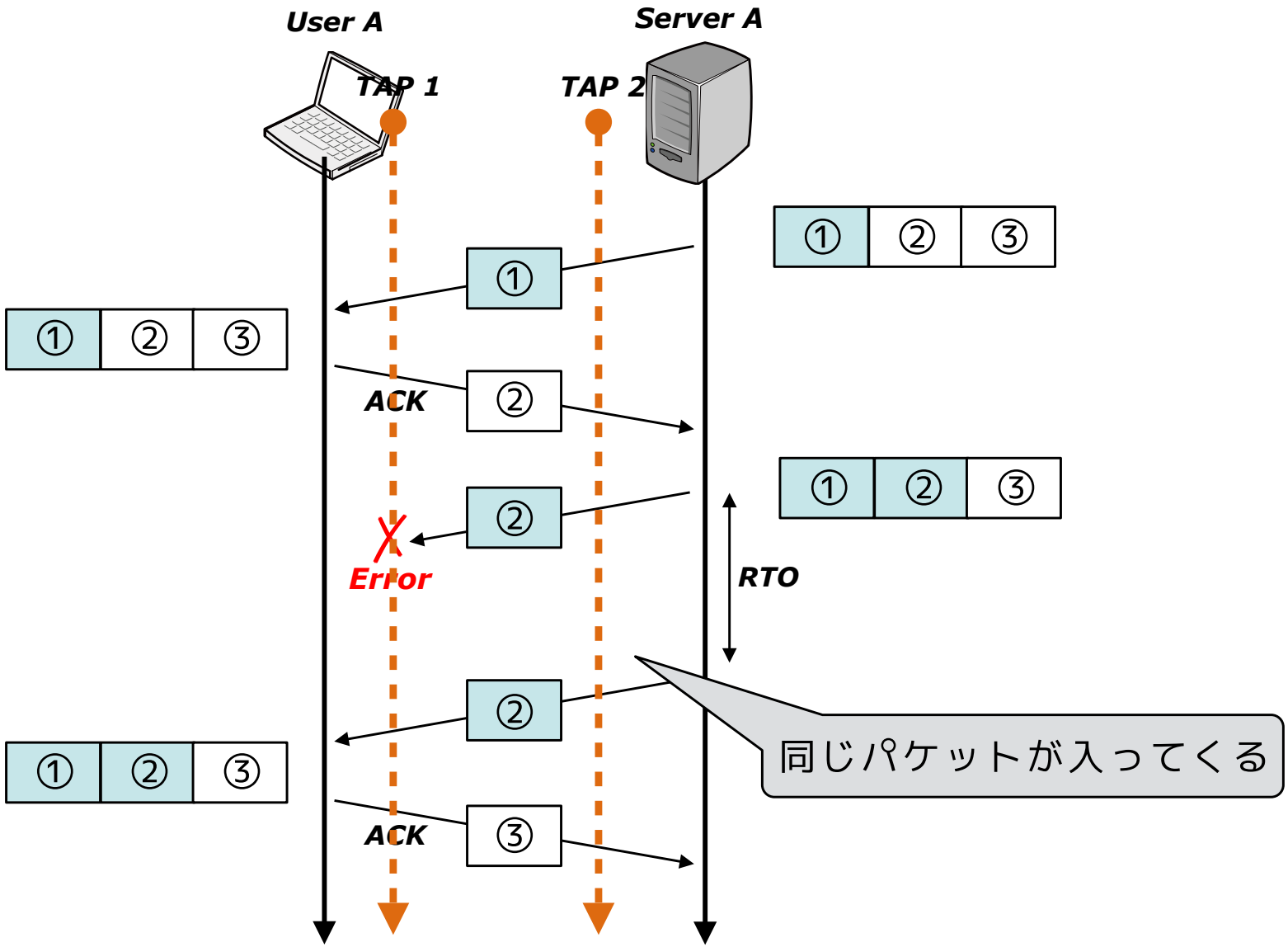
L7トラフィックを解析するには？



TCPの再送制御？



TCPの再送制御？

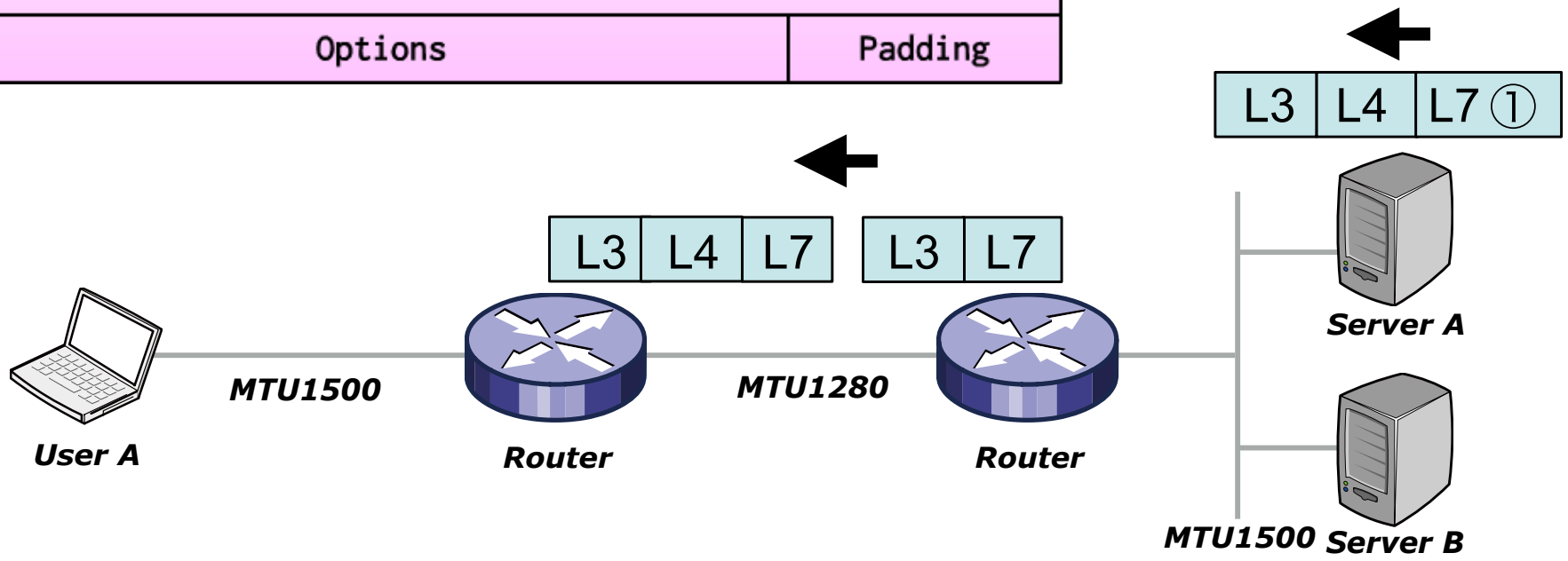


IP Fragment ?

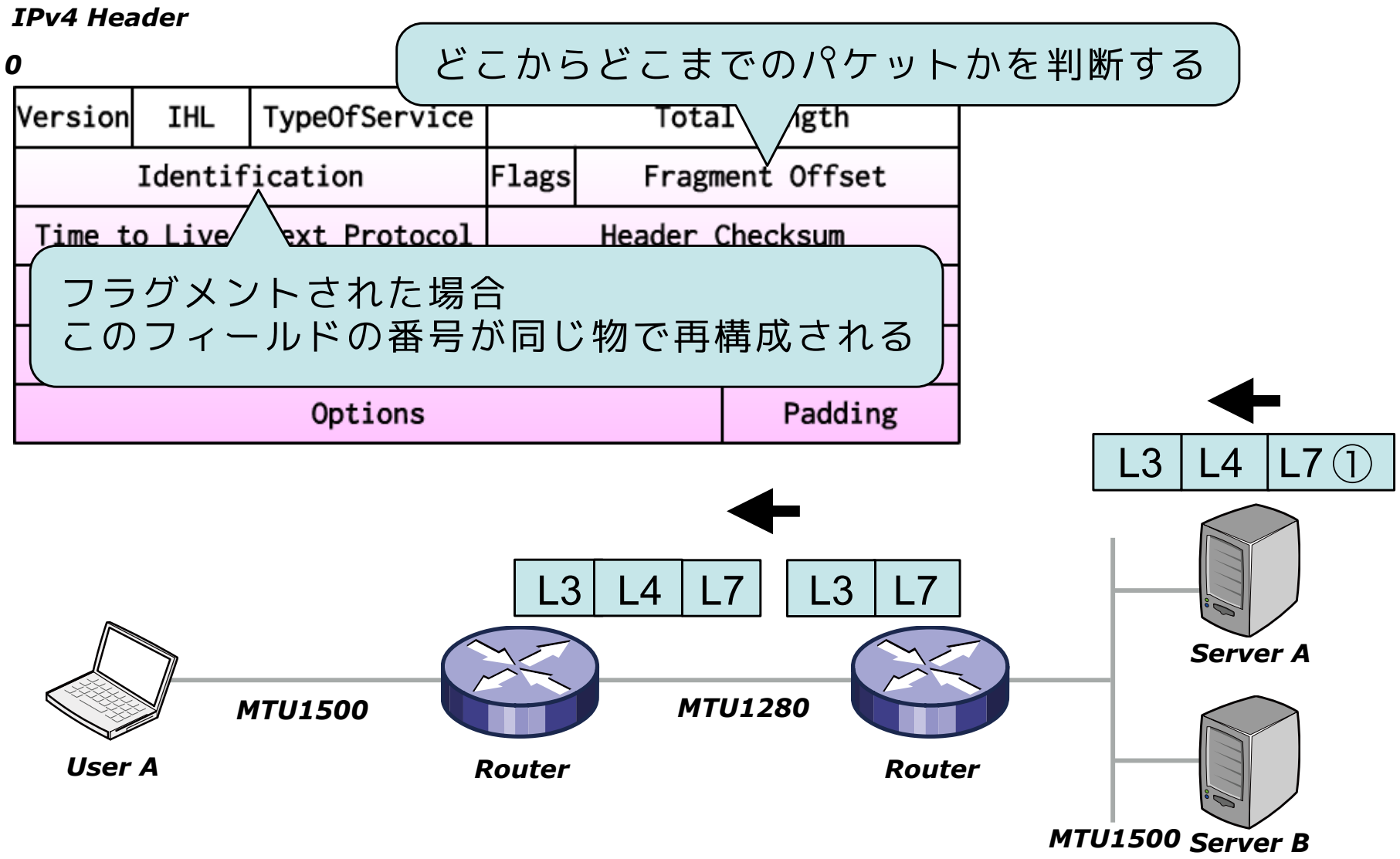
IPv4 Header

0 ***31***

Version	IHL	TypeOfService	Total Length
Identification		Flags	Fragment Offset
Time to Live	Next Protocol	Header Checksum	
Source IP Address			
Destination IP Address			
Options			Padding



IP Fragment ?


IPv4 Header
0
どこからどこまでのパケットかを判断する
Version
IHL
TypeOfService
Total
gth
Identification
Flags
Fragment Offset
Time to Live
ext Protocol
Header Checksum
フラグメントされた場合
このフィールドの番号が同じ物で再構成される
Options
Padding
L3
L4
L7 ①
L3
L4
L7
L3
L7
Server A
User A
MTU1500
Router
MTU1280
Router
MTU1500
Server B

パケットからL7情報をとりだすために！

- パケットからL7情報を再構築する必要がある
  - どこ当てのパケットであるかの振り分け
    - {src,dst} IP adress,tcp/udp,{src,dst} Port number等
  - TCPリアセンブル
    - 再送制御を考慮してデータグラムを再構築
  - IPデフラグメント
    - IP識別子とフラグメントオフセットからパケットを再構築
  - その他
    - TAPのトラフィックデータはその他いろいろな使わない情報も含まれているので，落としてやる必要がある．

パケットからL7情報をとりだすために！

- パケットからL7情報を再構築する必要がある
  - どこ光ての　　分け
    - 　　mber等

これを必要になる度にコーディングしたくない！

  - TCPリアセ　ンブル
    - 再送制御を考慮してデータグラムを再構築
  - IPデフラグメント
    - 　　トオフセットかパケットを再構築

ライブラリがあるんじゃ？

ネットワークアプライアンス機器つかえば？

  - その他
    - TAPのトラフィックデータはその他いろいろな使わない情報も含まれているので，落す必要がある．

ライブラリ？ソフトウェア？アプライアンス？

- ソフトウェア
  - wireshark,nDPI/OpenDPI,binpack/BRO,snort, いろいろ
- ライブラリ
  - libnids(c/c++),dpkt(python),net::frame/
    net::packet(perl), いろいろ
- ネットワークアプライアンス
  - ixia anue, panatech, いろいろ

ライブラリ？ソフトウェア？アプライアンス？

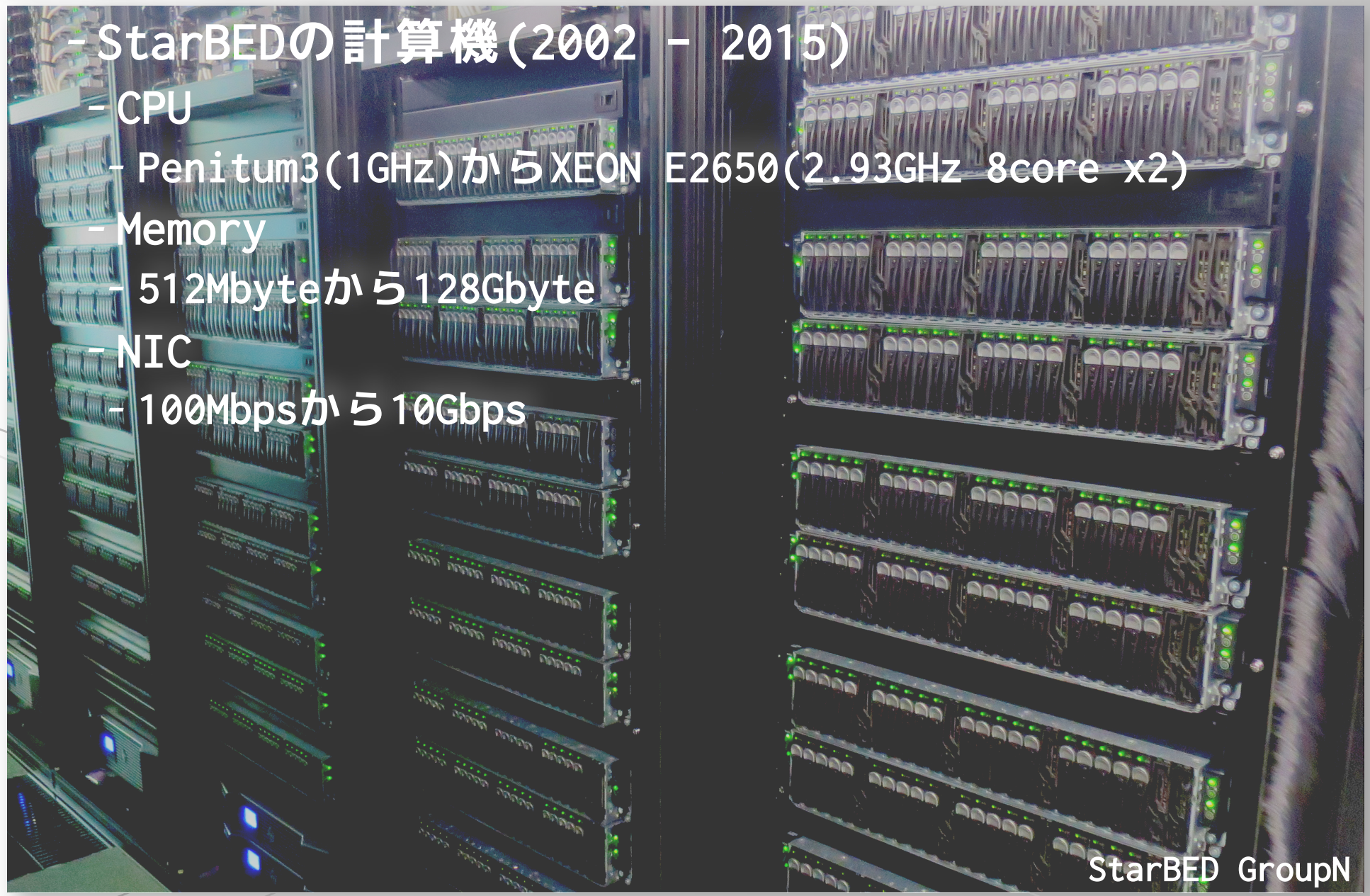
- ソフトウェア
  - /BRO,snort, いろいろ
- ライブラリ
  - libnids(
    net::pac
- ネットワークア
  - ixia anue, panat

解析ソフトウェアだと結果をもう一度プログラムに付け替える必要がある

みんなすきな言語ばらばらでしょ？

好きな解析ロジックをくみこむことがむずかしい

計算機とネットワークの高機能化



- StarBEDの計算機(2002 - 2015)
  - CPU
    - Penitum3(1GHz)からXEON E2650(2.93GHz 8core x2)
  - Memory
    - 512Mbyteから128Gbyte
  - NIC
    - 100Mbpsから10Gbps

トラフィック解析器基盤の開発目的

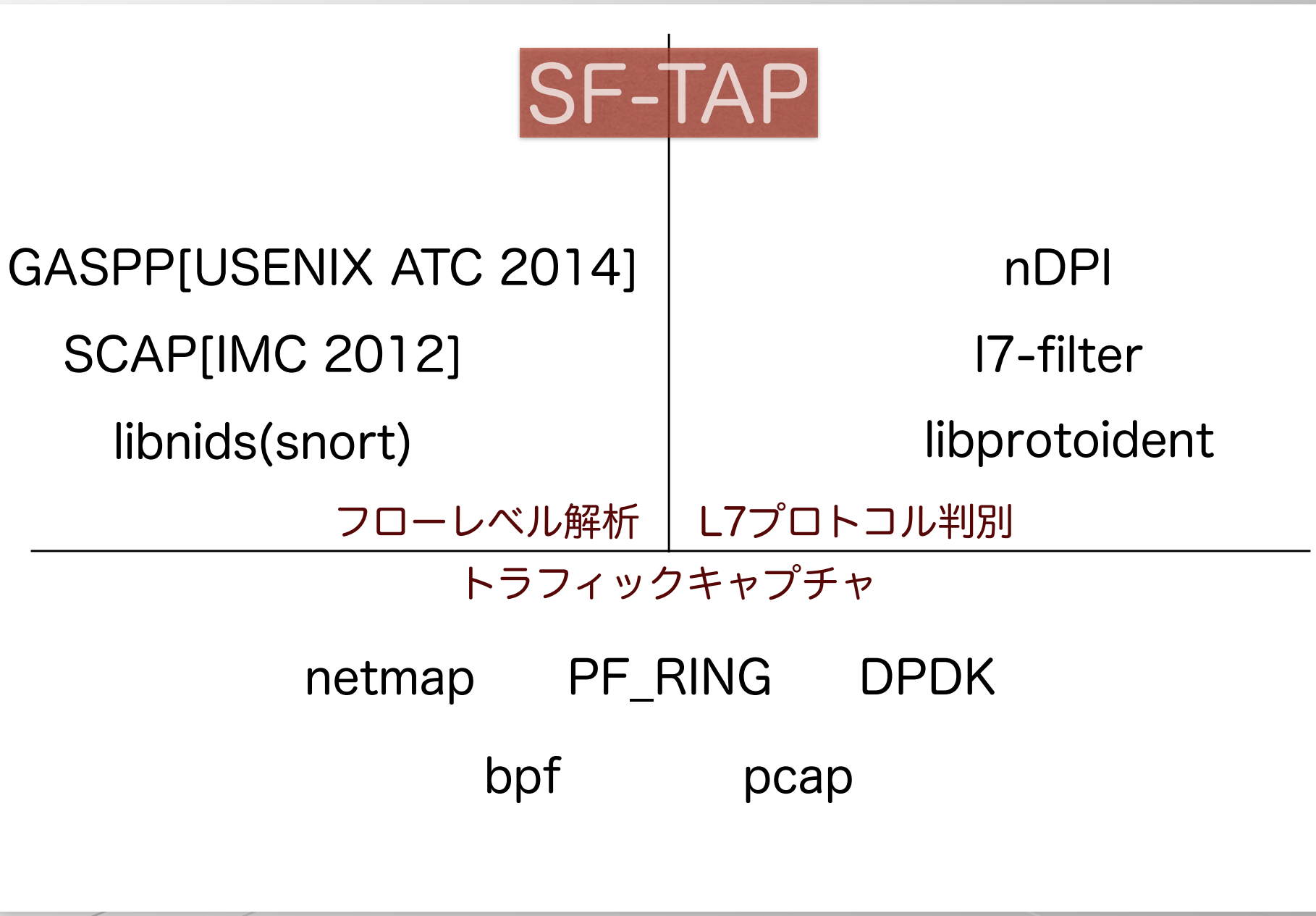
- 目的に応じた解析器が実装可能な柔軟性をもつこと

動的なロジック部の付け替えが可能なモジュラ化

- 高負荷なL7トラフィック解析を台数効果で捌ける規模追従性もつこと

トラフィックフロー分割による多段処理分散

関連技術・関連研究


SF-TAP
GASPP[USENIX ATC 2014]
SCAP[IMC 2012]
libnids(snort)
nDPI
l7-filter
libprotoident
フローレベル解析
L7プロトコル判別
トラフィックキャプチャ
netmap
PF_RING
DPDK
bpf
pcap

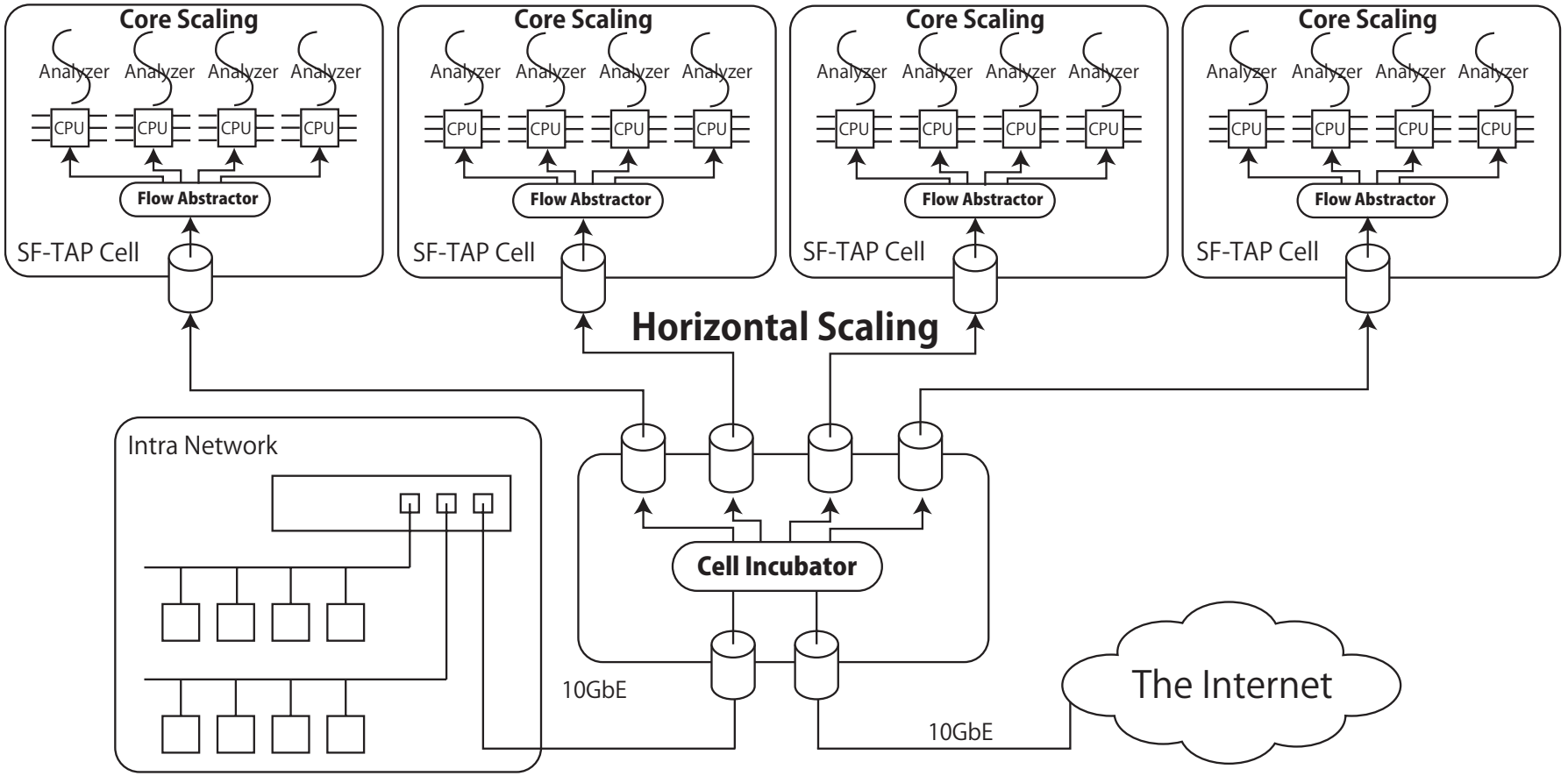
設計課題

課題１：
キャプチャの高速化
そもそも既存のPCのパケットキャプチャが遅くパケットを落とすことへの対応
- libpcap, bpf, sock_packet, etc…

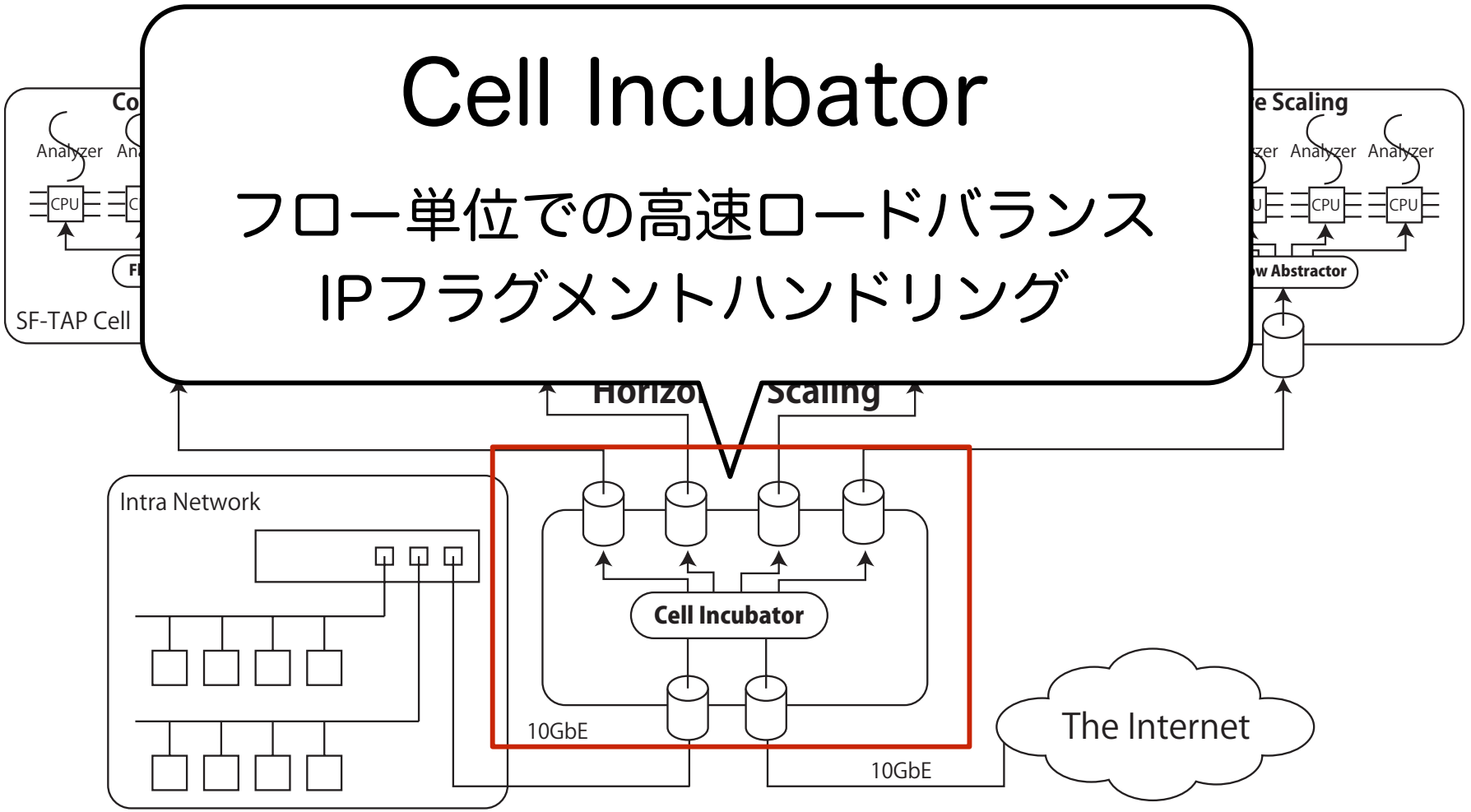
課題２：
解析プログラムへの負荷分散対応
- 機器分散, CPU分散,etc…

課題３：
OSの汎用的なIPC-IFを解析プログラムとの界面にすることでどの言語からも利用できるよう考慮
- pipe, shared memory, mmap, af_unix, etc…

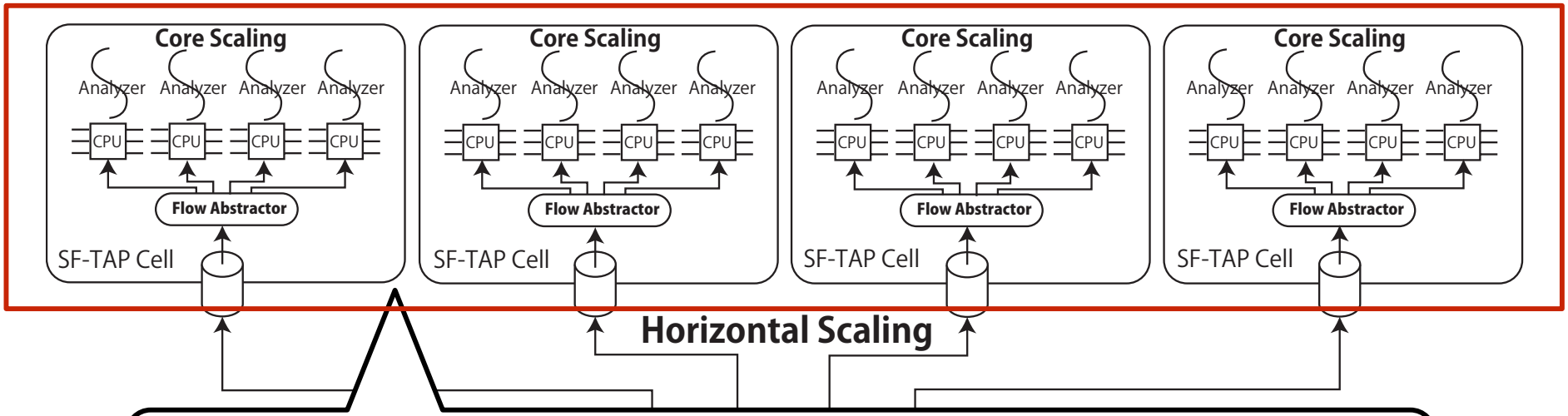
構成


Core Scaling
Analyzer
CPU
Flow Abstractor
SF-TAP Cell
Horizontal Scaling
Intra Network
Cell Incubator
10GbE
10GbE
The Internet

構成


Cell Incubator
フロー単位での高速ロードバランス
IPフラグメントハンドリング
Analyzer
CPU
SF-TAP Cell
Scaling
Intra Network
Cell Incubator
10GbE
10GbE
The Internet

構成



Flow Abstractor

L7プロトコル判別

IPフラグメント＆TCPストリーム再構成

フロー抽象化による非言語依存のIF提供

Cell Incubator（フロー単位での高速ロードバランス）

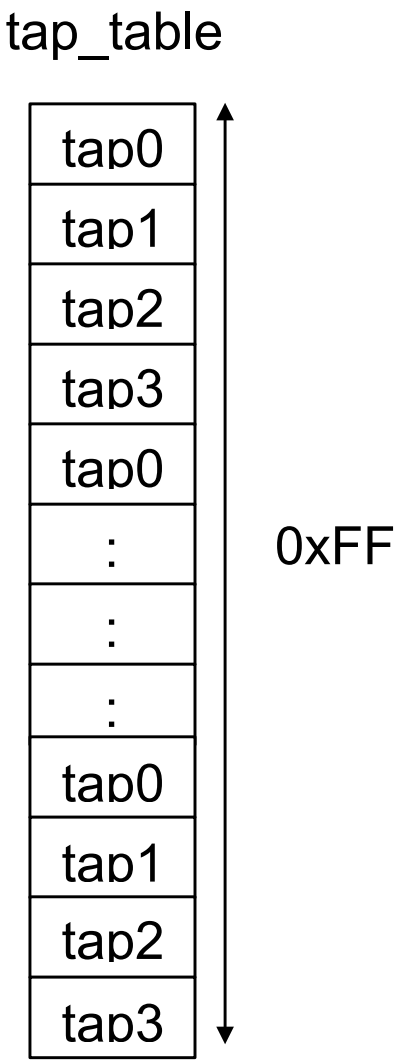
- ネットワーク機器の高速化
  - 10Mbps→100Mbps→1Gbps→10Gbps→40Gbps→100Gbps
  - 拾いきれないパケット，packet drop

- 高速パケットI/Oのためのソフトウェア
  - DPDK,PF-RING,NETMAP
    - Interrupt Coalescing
    - Pollingモデルの専用パケットI/O API
- フローロードバランシング２通り
  - L4のトラフィックフローを単位としたバランシング
    - srcPort,dstPort
  - L3のアドレスを単位としたバランシング
    - srcIP,dstIP

Cell Incubator（IPフラグメントハンドリング）

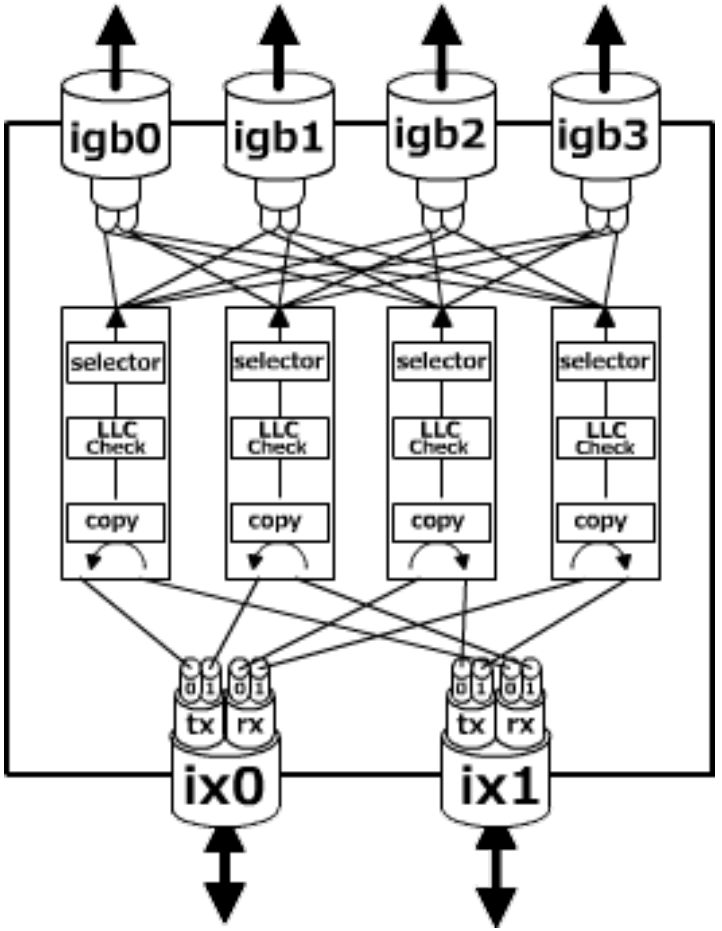
- 擬似コード

```
uint8_t selector(struct ether* frame) {
  select_value = hash(ether_frame);
  return tap_table[select_value];
}

uint8_t hash(struct ether* frame) {
  struct ip* ip_ptr = ipheader(frame);
  if ( !fragment ) {
       return l4hash(ip_ptr);
  } else {
       //フラグメントパケット
      uint16_t key = hash16(ip_ptr->id^ip_ptr->src^ip_ptr->dst);
      if ( top_of_fragment_packet ) {
         //先頭
         fragment_table[key] = l4hash(ip_ptr);
         return fragment_table[key];
      } else {
         //途中のパケット
         return fragment_table[key];
      }
  }
}
```

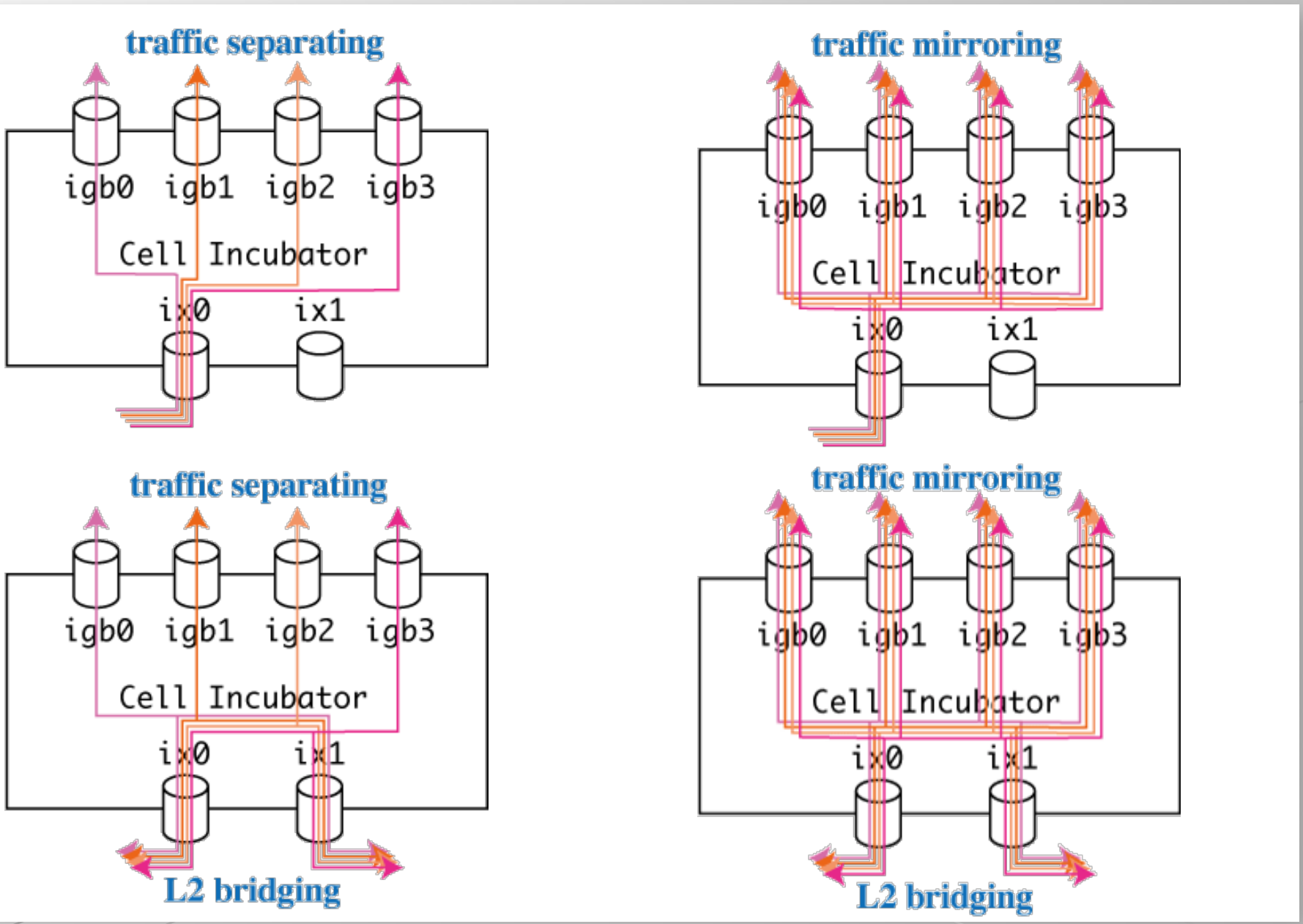


Cell Incubator

- トラフィックの受け側IF
  - NICのtxq_rxq対のthreadを持つ
  - 各スレッドが独立したFlowバランステーブルをもつ
- TAP側IF
  - cas spin lock
  - tap-ifのtxqへバランスさせる
  - LLC FrameはTAP側へ転送しない
- プログラム
  - c++11

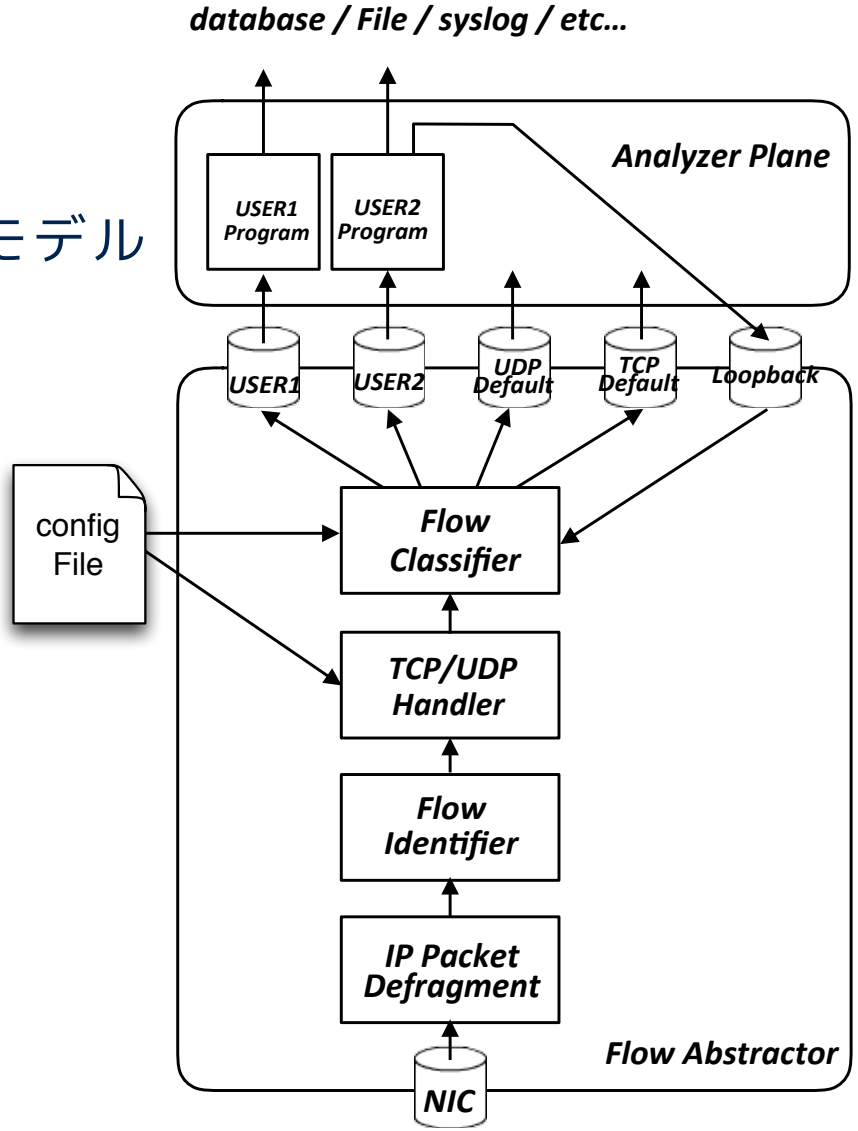


Cell Incubator



SF-TAP Cell

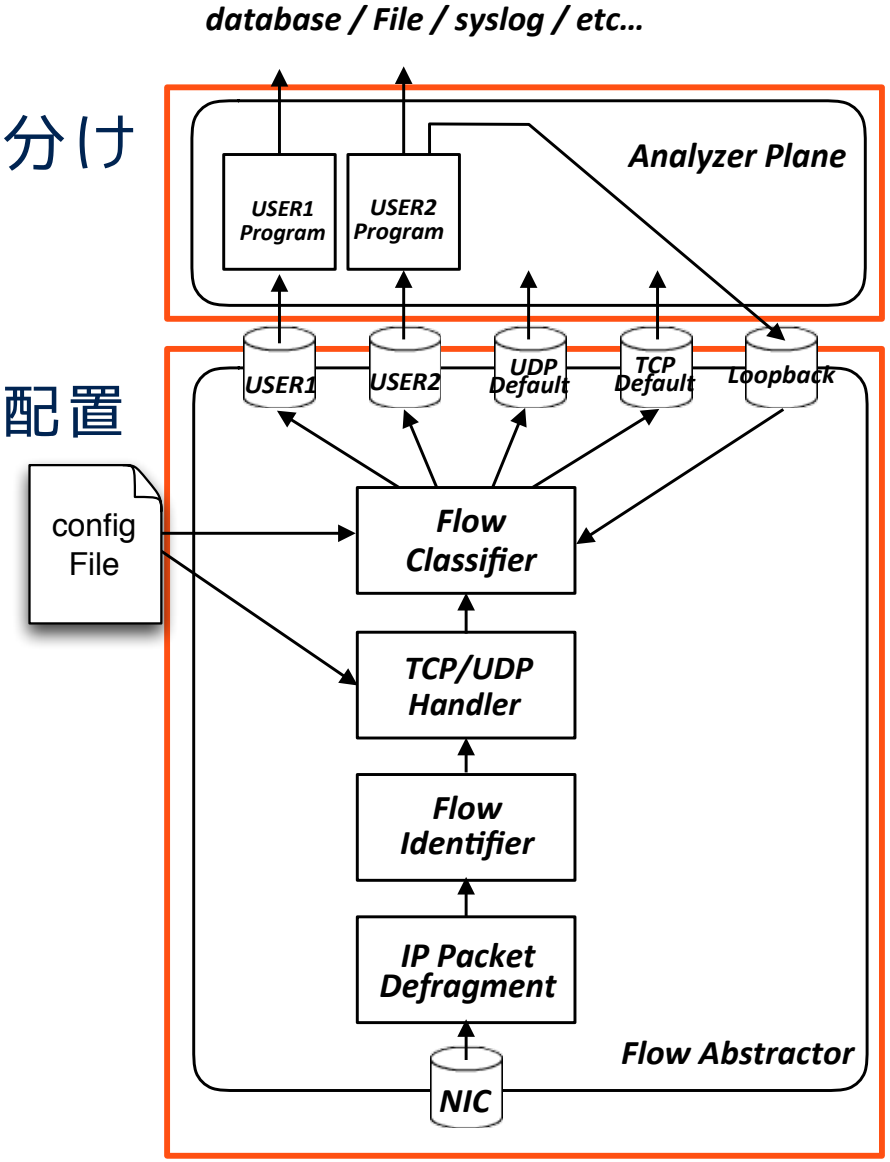
- プログラム
  - C++11
  - コンシューマ／プロデューサモデル
  - 各要素ごとにthread化
  - ライブラリ
    - re2,event,boost,yaml-cpp



SF-TAP Cell

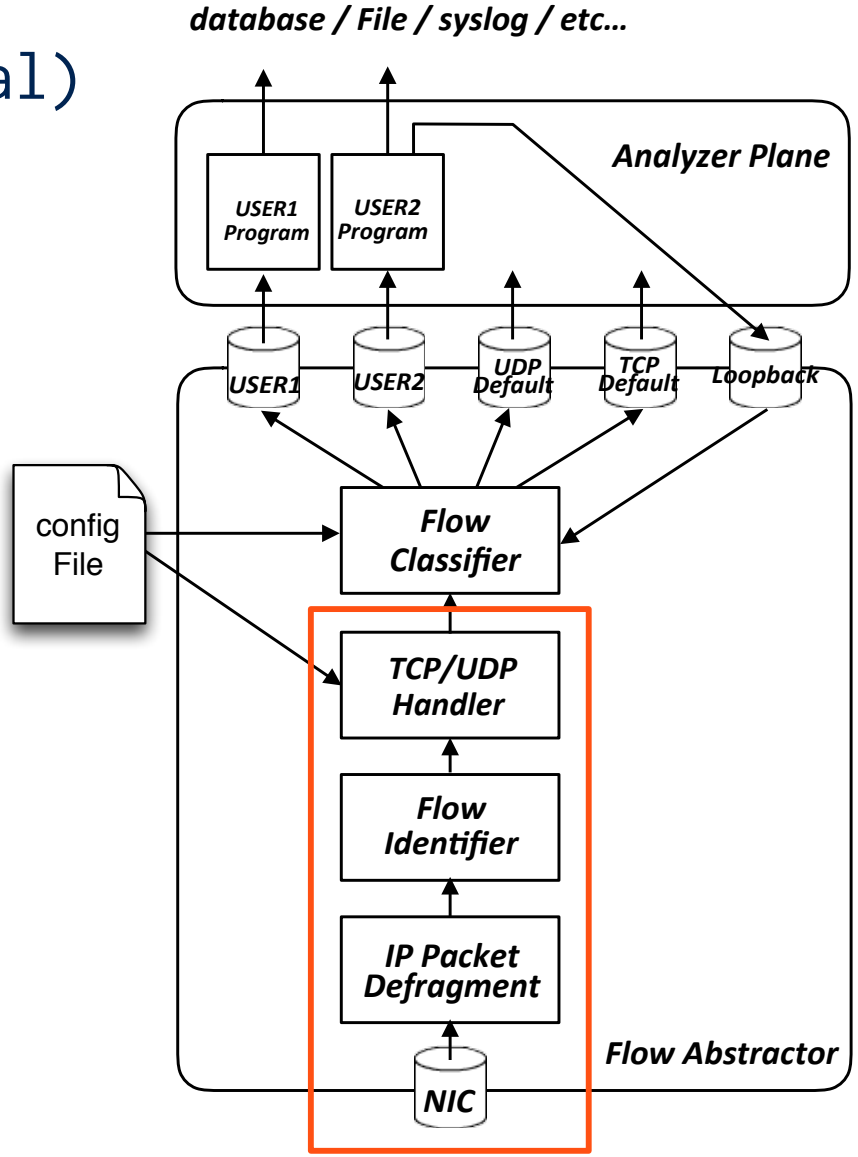
- Flow-Abstractor
  - 解析プログラムへの
    抽象化トラフィック振り分け

- Analyser Plane
  - ユーザの解析プログラムを配置



SF-TAP Cell (Flow Abstractor)

- パケットキャプチャ
  - pcap,netmap(experimental)
- フローの再構成
  - TCPの再構成
  - フロー識別
  - IPパケットの再構成
- 処理間はRINGを持つ
  - cas spin lock
  - condition lock



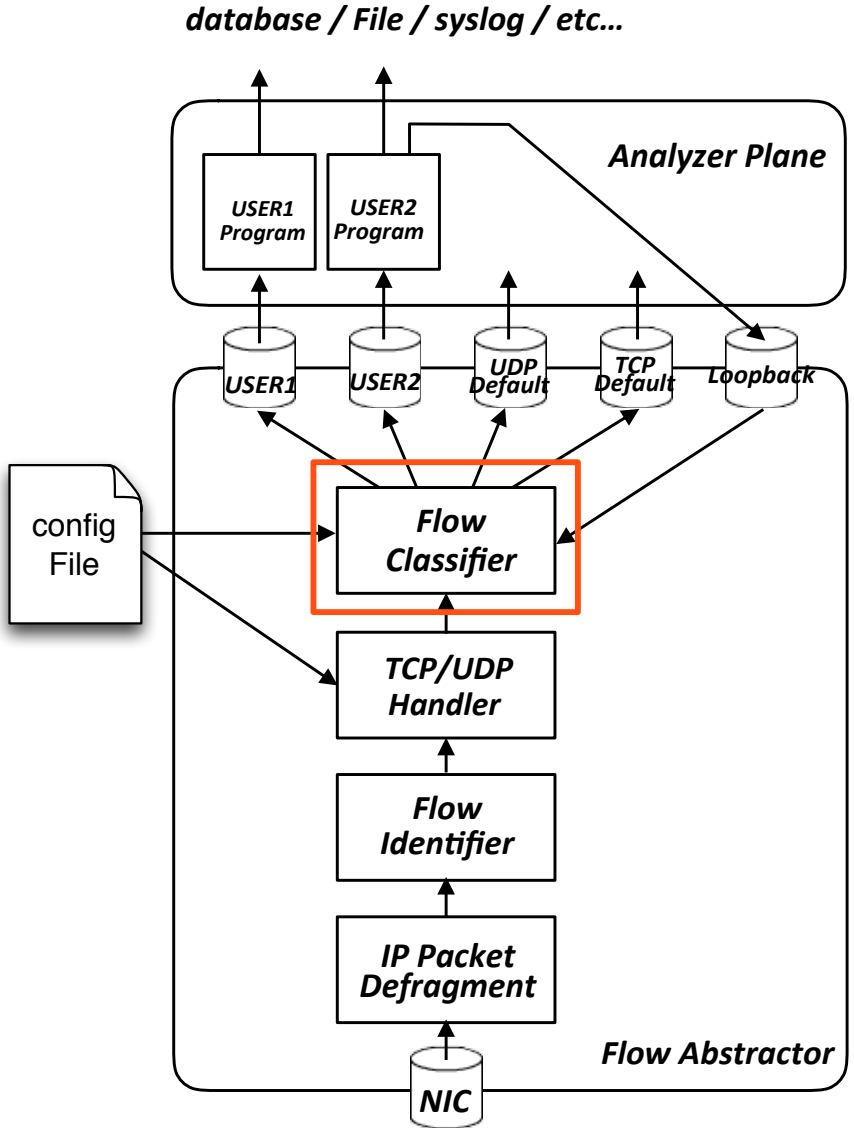
SF-TAP Cell (Flow Abstractor)

- 抽象化IFへの転送

- Flow-ID マッチング
  - IP address
  - port
  - hop count

- stream マッチング
  - up stream
  - down stream

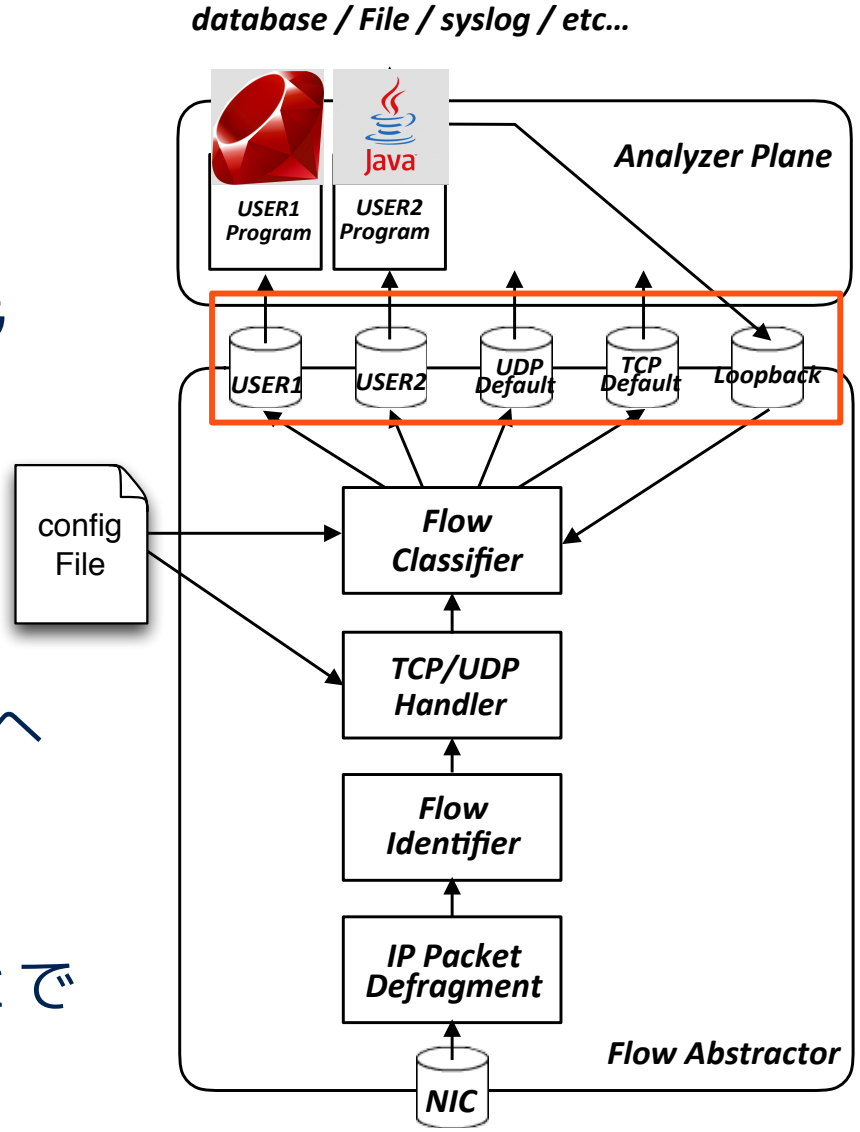
- ヘッダパターンマッチ
  - 正規表現



SF-TAP Cell (Flow Abstractor)

- Unix Domain Socket
  - ユーザ定義IF
    - config fileにより定義
  - Default IF
    - ユーザ定義から外れたトラフィックを出力する
      - TCP
      - UDP
  - Loopback IF
    - もう一度flow-abstractorへ

  - 使えない場合
    - netcat/socat等を使うことでpipeで標準出力へ



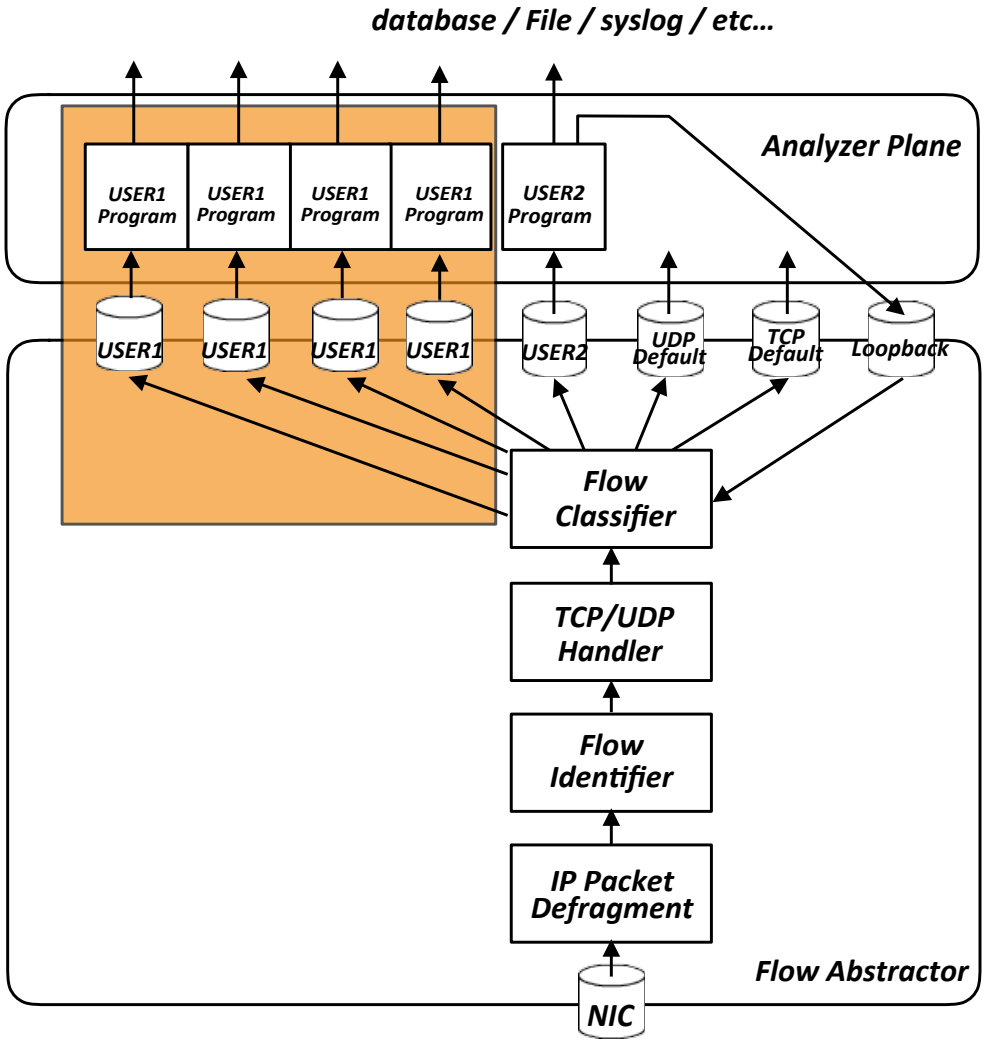
SF-TAP Cell (Flow Abstractor Config File)

```
# global configuration
global:
  home:    /tmp/sf-tap
  timeout: 30  # close long-lived (over 30[s]) but do-nothing connections
  lru:     yes # bring the least recently used pattern to front of list
  cache:   yes # use cache for regex
  tcp_threads:   2
  regex_threads: 2

loopback7:
  if:     loopback7
  format: text

tcp_default:
  if:     default # for every flow that wasn't matched by any rules
  proto:  TCP
  format: text
  body:   yes

udp_default:
  if:     default # for every flow that wasn't matched by any rules
  proto:  UDP
  format: text
  body:   yes

http:
  up:     '^[-a-zA-Z]+ .+ HTTP/1\.(0\r?\n|1\r?\n([-a-zA-Z]+: .+\r?\n)+)'
  down:   '^HTTP/1\.[01] [1-9][0-9]{2} .+\r?\n'
  proto:  TCP  # TCP or UDP
  if:     http
  format: text # text or binary
  body:   yes  # if specified 'no', only header is output
  nice:   100  # the smaller a value is, the higher a priority is
  utf8:   no   # treat data as UTF8 or latin1 (binary). used for regex
#  balance: 4   # flows are balanced by 4 interfaces
```

flow-abstractor/examples
から一部抜粋

SF-TAP Cell (Flow Abstractor ユーザ定義IFの負荷分散)

- ユーザ定義インターフェースを任意の数にFlowを維持するようにロードバランシング

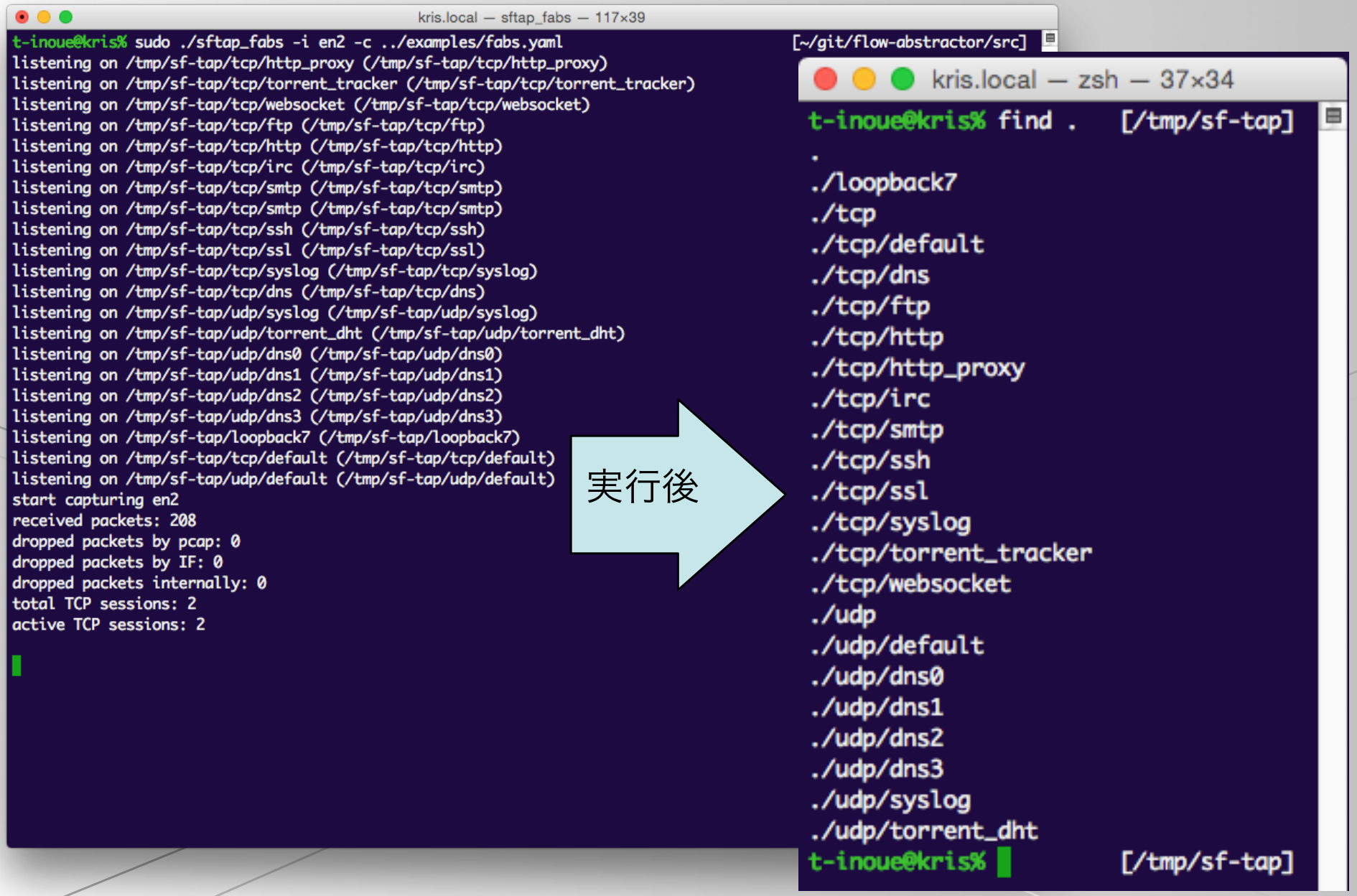


SF-TAP Cell (Flow Abstractor Port Number)

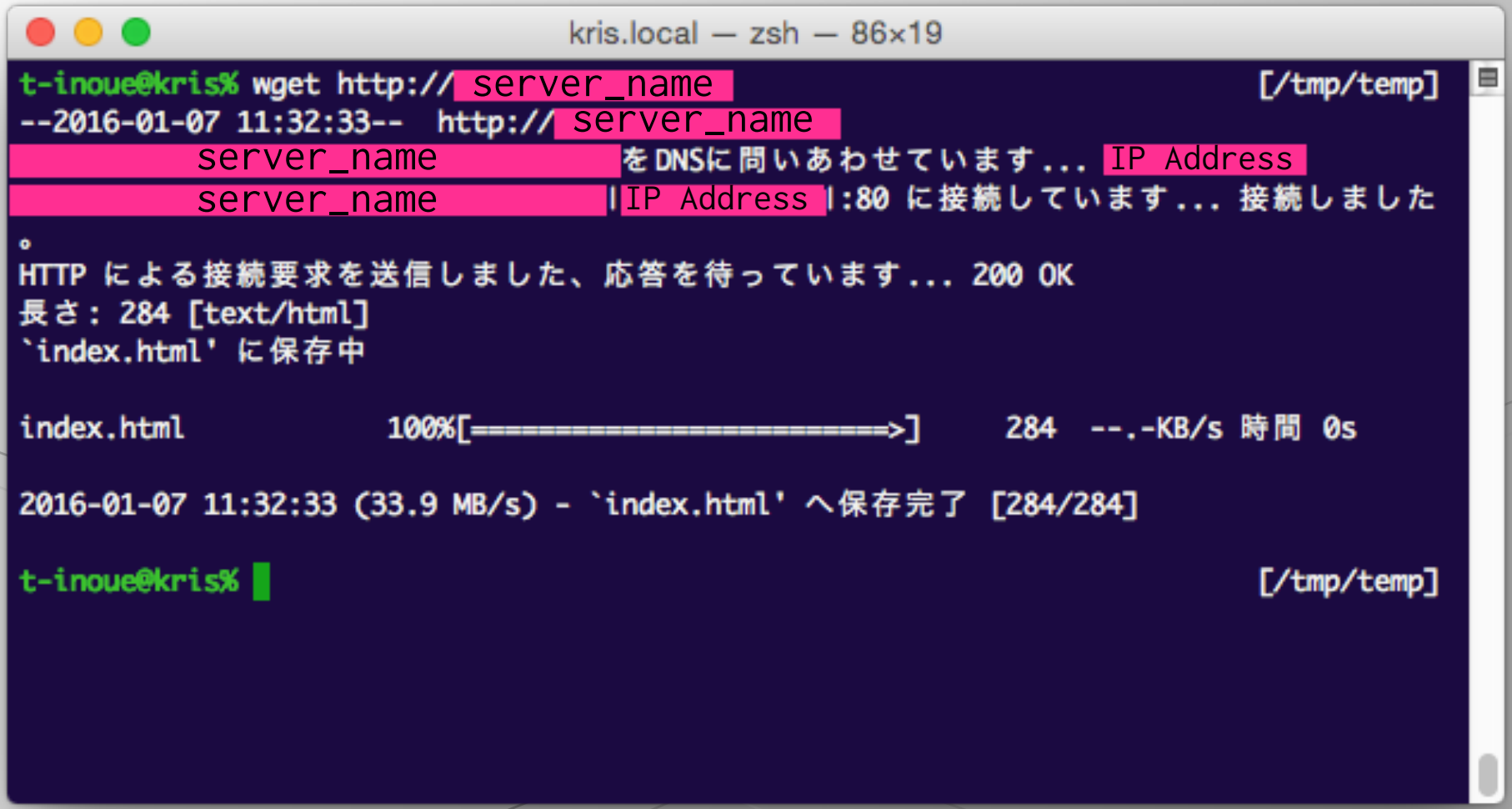
```
syslog_udp:
  up:     '^<([0-9]|[1-9][0-9]|1[0-8][0-9]|19[01])>'
  proto:  UDP
  if:     syslog
  format: text
  port:   514
  nice:   100
  utf8:   no

syslog_tcp:
  up:     '^([1-9][0-9]* )?<([0-9]|[1-9][0-9]|1[0-8][0-9]|19[01])>'
  down:   '^.?'
  proto:  TCP
  if:     syslog
  format: text
  port:   514
  nice:   100
  utf8:   no

dns_udp:
  proto:  UDP
  if:     dns
  port:   53,5353,5355 # 53: UDP DNS, 5353: multicast DNS, 5355: LLMNR
  format: text
  nice:   200
  utf8:   no

dns_tcp:
  proto:  TCP
  if:     dns
  port:   53
  format: text
  nice:   200
  utf8:   no
```

flow-abstractor/examples
から一部抜粋

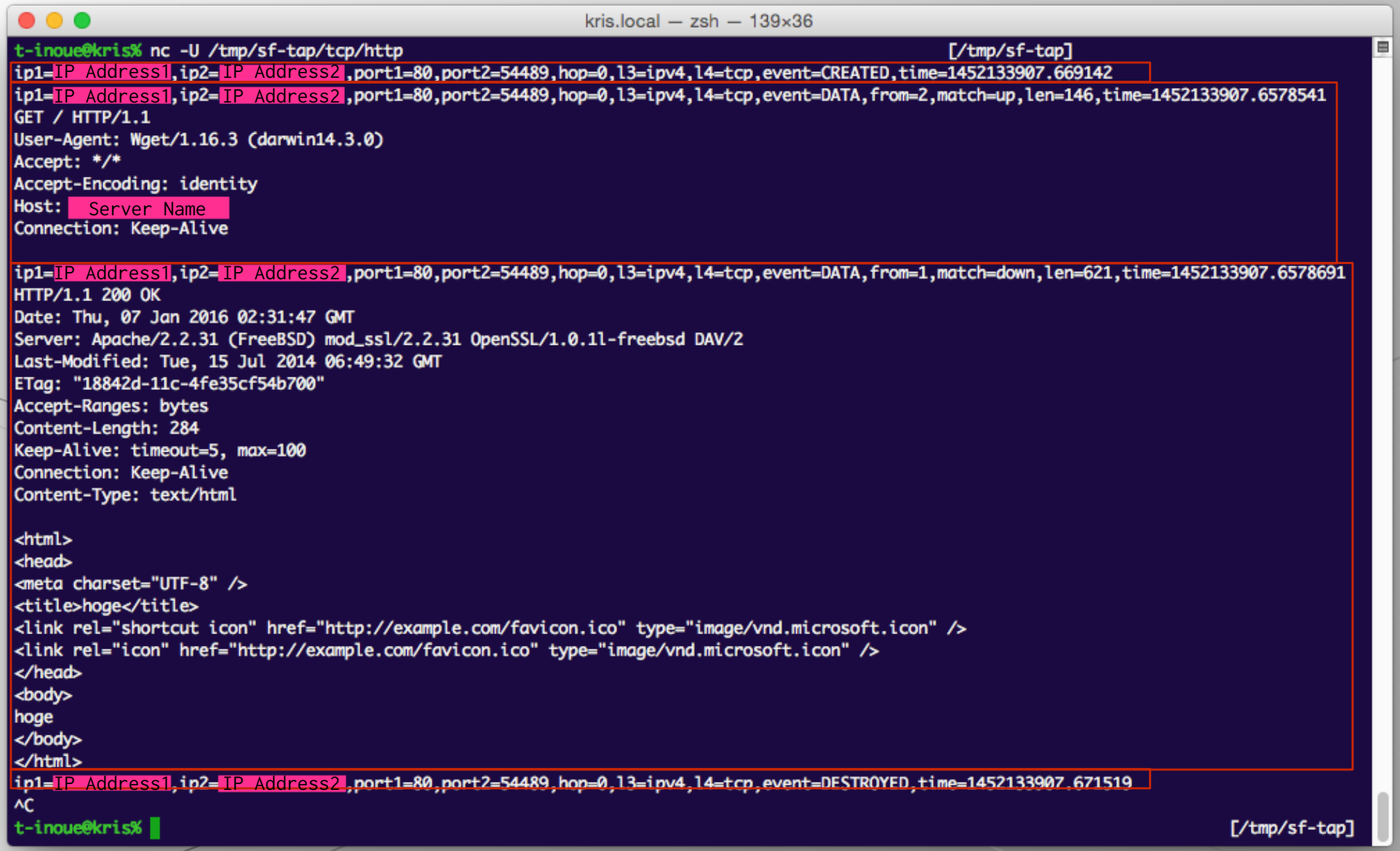
実行画面（サンプル）


kris.local — sftap_fabs — 117×39
t-inoue@kris% sudo ./sftap_fabs -i en2 -c ../examples/fabs.yaml
listening on /tmp/sf-tap/tcp/http_proxy (/tmp/sf-tap/tcp/http_proxy)
listening on /tmp/sf-tap/tcp/torrent_tracker (/tmp/sf-tap/tcp/torrent_tracker)
listening on /tmp/sf-tap/tcp/websocket (/tmp/sf-tap/tcp/websocket)
listening on /tmp/sf-tap/tcp/ftp (/tmp/sf-tap/tcp/ftp)
listening on /tmp/sf-tap/tcp/http (/tmp/sf-tap/tcp/http)
listening on /tmp/sf-tap/tcp/irc (/tmp/sf-tap/tcp/irc)
listening on /tmp/sf-tap/tcp/smtp (/tmp/sf-tap/tcp/smtp)
listening on /tmp/sf-tap/tcp/smtp (/tmp/sf-tap/tcp/smtp)
listening on /tmp/sf-tap/tcp/ssh (/tmp/sf-tap/tcp/ssh)
listening on /tmp/sf-tap/tcp/ssl (/tmp/sf-tap/tcp/ssl)
listening on /tmp/sf-tap/tcp/syslog (/tmp/sf-tap/tcp/syslog)
listening on /tmp/sf-tap/tcp/dns (/tmp/sf-tap/tcp/dns)
listening on /tmp/sf-tap/udp/syslog (/tmp/sf-tap/udp/syslog)
listening on /tmp/sf-tap/udp/torrent_dht (/tmp/sf-tap/udp/torrent_dht)
listening on /tmp/sf-tap/udp/dns0 (/tmp/sf-tap/udp/dns0)
listening on /tmp/sf-tap/udp/dns1 (/tmp/sf-tap/udp/dns1)
listening on /tmp/sf-tap/udp/dns2 (/tmp/sf-tap/udp/dns2)
listening on /tmp/sf-tap/udp/dns3 (/tmp/sf-tap/udp/dns3)
listening on /tmp/sf-tap/loopback7 (/tmp/sf-tap/loopback7)
listening on /tmp/sf-tap/tcp/default (/tmp/sf-tap/tcp/default)
listening on /tmp/sf-tap/udp/default (/tmp/sf-tap/udp/default)
start capturing en2
received packets: 208
dropped packets by pcap: 0
dropped packets by IF: 0
dropped packets internally: 0
total TCP sessions: 2
active TCP sessions: 2
[~/git/flow-abstractor/src]
実行後
kris.local — zsh — 37×34
t-inoue@kris% find . [/tmp/sf-tap]
.
./loopback7
./tcp
./tcp/default
./tcp/dns
./tcp/ftp
./tcp/http
./tcp/http_proxy
./tcp/irc
./tcp/smtp
./tcp/ssh
./tcp/ssl
./tcp/syslog
./tcp/torrent_tracker
./tcp/websocket
./udp
./udp/default
./udp/dns0
./udp/dns1
./udp/dns2
./udp/dns3
./udp/syslog
./udp/torrent_dht
t-inoue@kris% [/tmp/sf-tap]

http（サンプル）


kris.local — zsh — 86×19
t-inoue@kris% wget http:// server_name [/tmp/temp]
--2016-01-07 11:32:33-- http:// server_name
server_name をDNSに問いあわせています... IP Address
server_name |IP Address|:80 に接続しています... 接続しました
。
HTTP による接続要求を送信しました、応答を待っています... 200 OK
長さ: 284 [text/html]
`index.html' に保存中
index.html 100%[=========================>] 284 --.-KB/s 時間 0s
2016-01-07 11:32:33 (33.9 MB/s) - `index.html' へ保存完了 [284/284]
t-inoue@kris% [/tmp/temp]

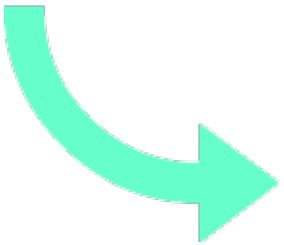
http （サンプル）


kris.local — zsh — 139×36
t-inoue@kris% nc -U /tmp/sf-tap/tcp/http [/tmp/sf-tap]
ip1=IP Address1,ip2=IP Address2,port1=80,port2=54489,hop=0,l3=ipv4,l4=tcp,event=CREATED,time=1452133907.669142
ip1=IP Address1,ip2=IP Address2,port1=80,port2=54489,hop=0,l3=ipv4,l4=tcp,event=DATA,from=2,match=up,len=146,time=1452133907.6578541
GET / HTTP/1.1
User-Agent: Wget/1.16.3 (darwin14.3.0)
Accept: */*
Accept-Encoding: identity
Host: Server Name
Connection: Keep-Alive
ip1=IP Address1,ip2=IP Address2,port1=80,port2=54489,hop=0,l3=ipv4,l4=tcp,event=DATA,from=1,match=down,len=621,time=1452133907.6578691
HTTP/1.1 200 OK
Date: Thu, 07 Jan 2016 02:31:47 GMT
Server: Apache/2.2.31 (FreeBSD) mod_ssl/2.2.31 OpenSSL/1.0.1l-freebsd DAV/2
Last-Modified: Tue, 15 Jul 2014 06:49:32 GMT
ETag: "18842d-11c-4fe35cf54b700"
Accept-Ranges: bytes
Content-Length: 284
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
<html>
<head>
<meta charset="UTF-8" />
<title>hoge</title>
<link rel="shortcut icon" href="http://example.com/favicon.ico" type="image/vnd.microsoft.icon" />
<link rel="icon" href="http://example.com/favicon.ico" type="image/vnd.microsoft.icon" />
</head>
<body>
hoge
</body>
</html>
ip1=IP Address1,ip2=IP Address2,port1=80,port2=54489,hop=0,l3=ipv4,l4=tcp,event=DESTROYED,time=1452133907.671519
^C
t-inoue@kris% [/tmp/sf-tap]

SF-TAPのヘッダフォーマット

- CSVによるkey-valueペア

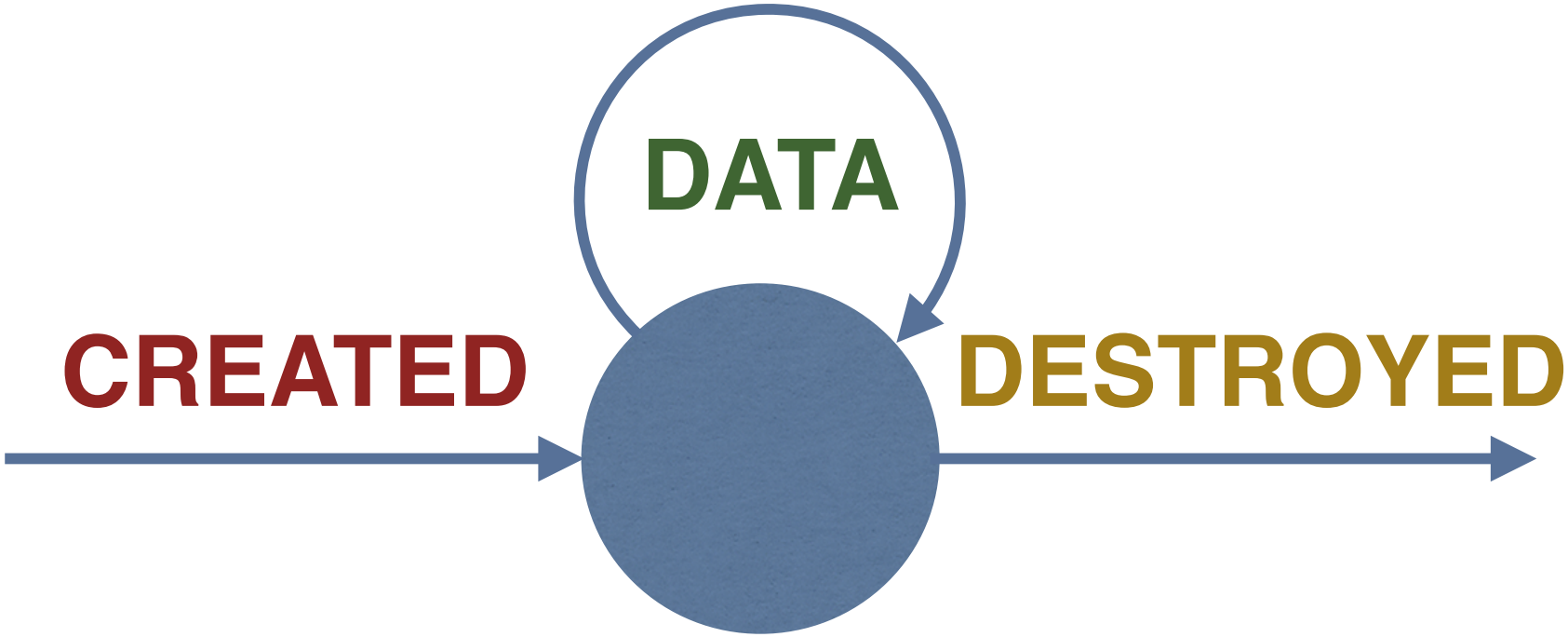
ip1=192.168.24.54,ip2=216.58.221.196,port1=59547,port2=80,hop=0,l3=ipv4,l4=tcp,event=CREATED



```
{
  “ip1”: “192.168.24.54”,
  “ip2”: “216.58.221.196”,
  “port1”: 59547,
  “port2”: 80,
  “hop”: 0,
  “l3”: “ipv4”,
  “l4”: “tcp”,
  “event”: “CREATED”
}
```

Event フォーマット

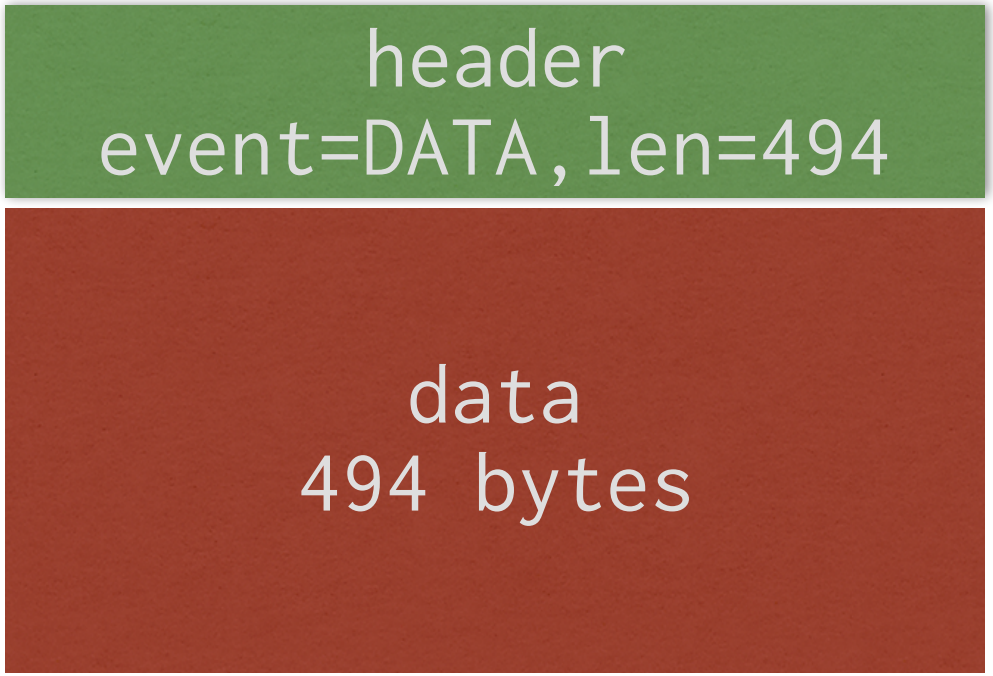
When arriving data, DATA event is invoked.

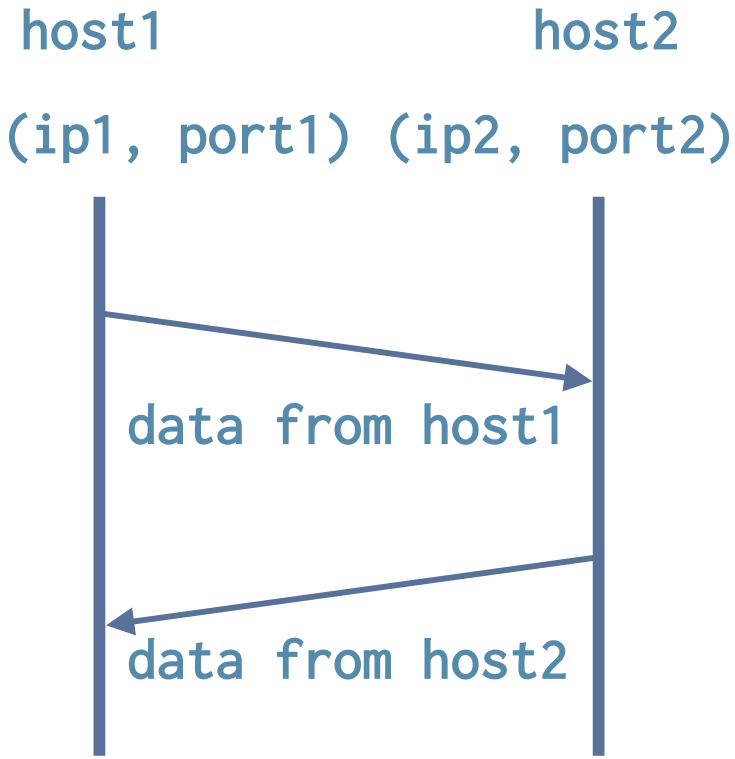


CREATED	When TCP connection is established (performed 3-way handshake), CREATED event is invoked.
DESTROYED	When TCP connection is destroyed (received FIN/RST, or timeout), DESTROYED event is invoked.

Data フォーマット 1 (from)

```
ip1=192.168.24.54,ip2=216.58.221.196,port1=59547,port2=80,h
op=0,l3=ipv4,l4=tcp,event=DATA,from=2,match=down,len=494
```





Data フォーマット2 (upstream / downstream)

Configuration

```
http:
  up:     '^[-a-zA-Z]+ .+ HTTP/1\.(0\r?\n|1\r?\n([-a-zA-Z]+: .+\r?\n)+)'
  down:   '^HTTP/1\.[01] [1-9][0-9]{2} .+\r?\n'
  proto:  TCP  # TCP or UDP
  if:     http # file name of UNIX domain socket
  format: text # text or binary
  body:   yes  # if specified 'no', only header is output
```

Matched with the pattern of downstream

```
ip1=192.168.24.54,ip2=216.58.221.196,port1=59547,port2=80,hop=0,l3=ipv4,l4=tcp,ev
ent=DATA,from=2,match=down,len=494
```

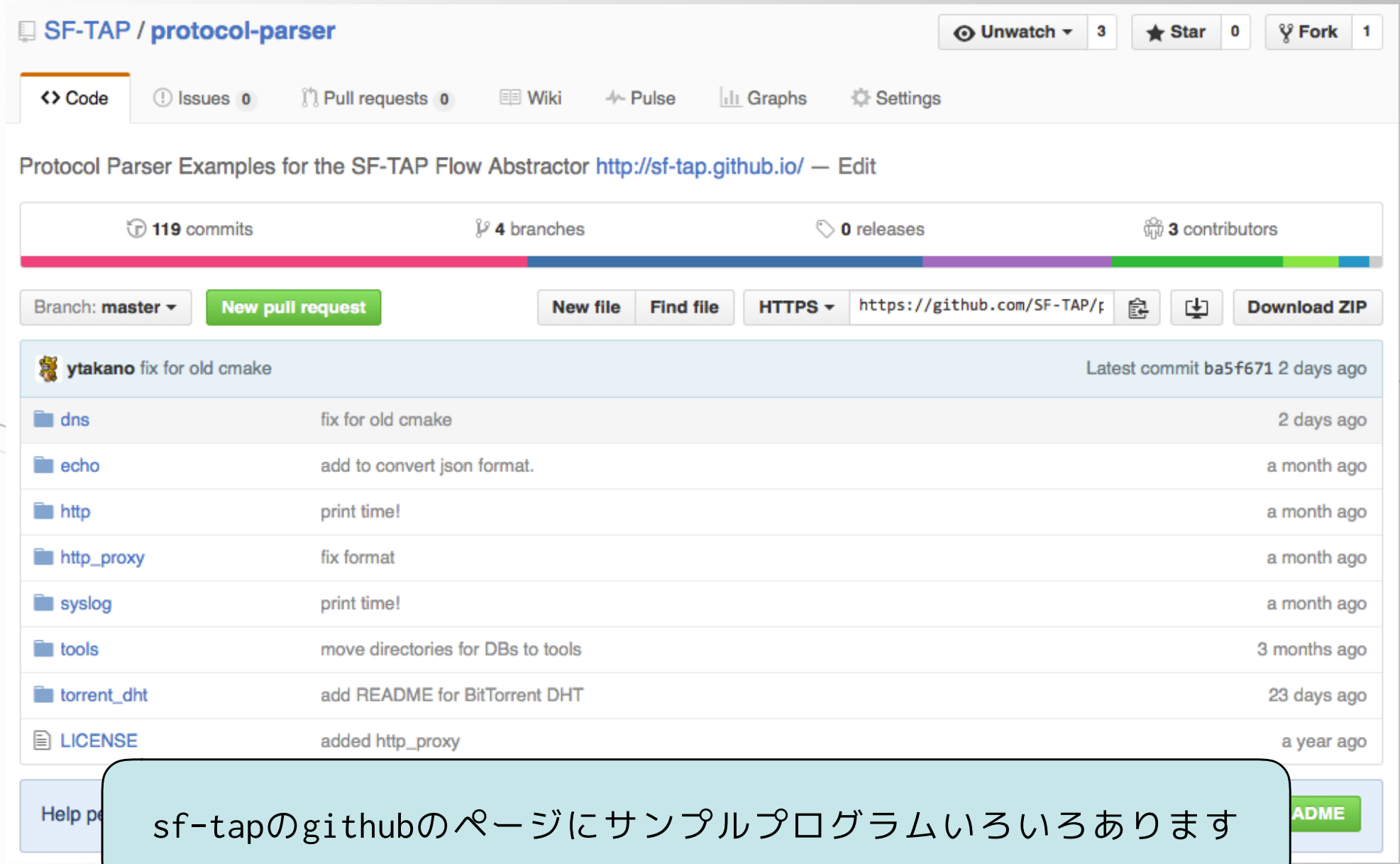
Matched with the pattern of upstream

```
ip1=192.168.24.54,ip2=216.58.221.196,port1=59547,port2=80,hop=0,l3=ipv4,l4=tcp,e
vent=DATA,from=1,match=up,len=78
```

解析プログラムの擬似コード

```
// connect to socket
s = socket();
connect(s, “/tmp/sf-tap/tcp/http” );

for (;;) {
  // read header
  readline(s, line);
  h = parse_header(line);

   // generate session ID
  sid = new sessionID(h[ “ip1” ], h[ “ip2” ],
                      h[ “port1” ], h[ “port2” ], h[ “hop” ]);

  if (h[ “event” ] ==  “DATA” ) {
    read(s, buf, h[ “len” ]);
    analysis_process(buf, h[ “len” ]);
  }
}
```

解析プログラム（プロトコルパーサ）



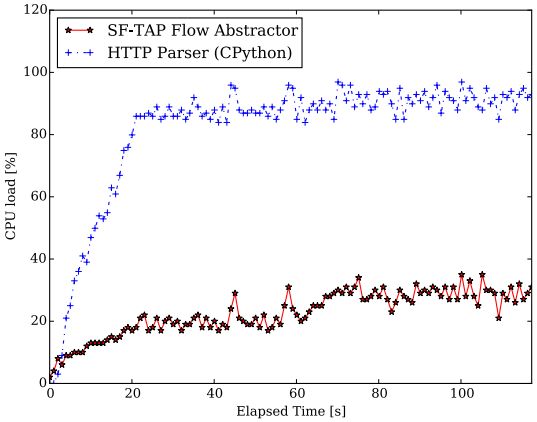
sf-tapのgithubのページにサンプルプログラムいろいろあります

Flow Abstractor の計測

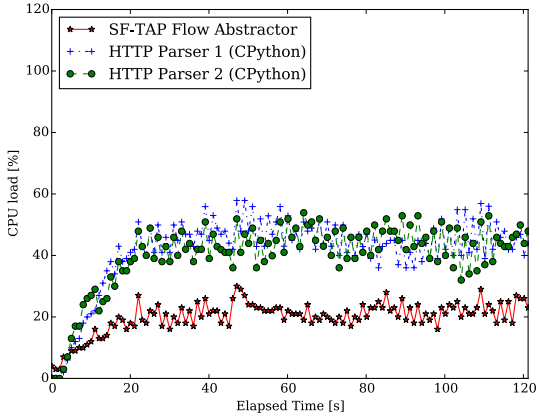
- 複数IFに対するロードバランシング

CPU : Xeon E7 4830 v2 (10core 2.2Ghz)
Memory: DDR3 1.5TB
OS : Ubuntu 14.10 (kernel 3.16)

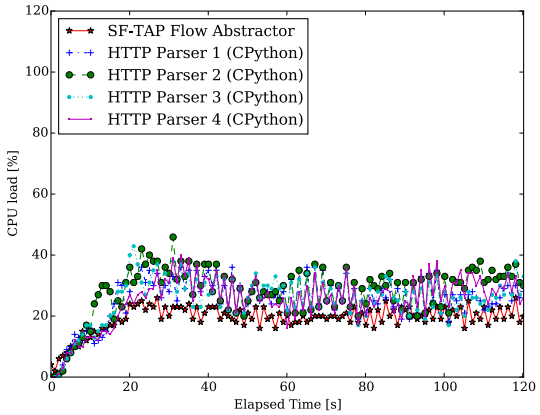
負荷器: curl-loader



(a) HTTP Analyzer x 1



(b) HTTP Analyzer x 2

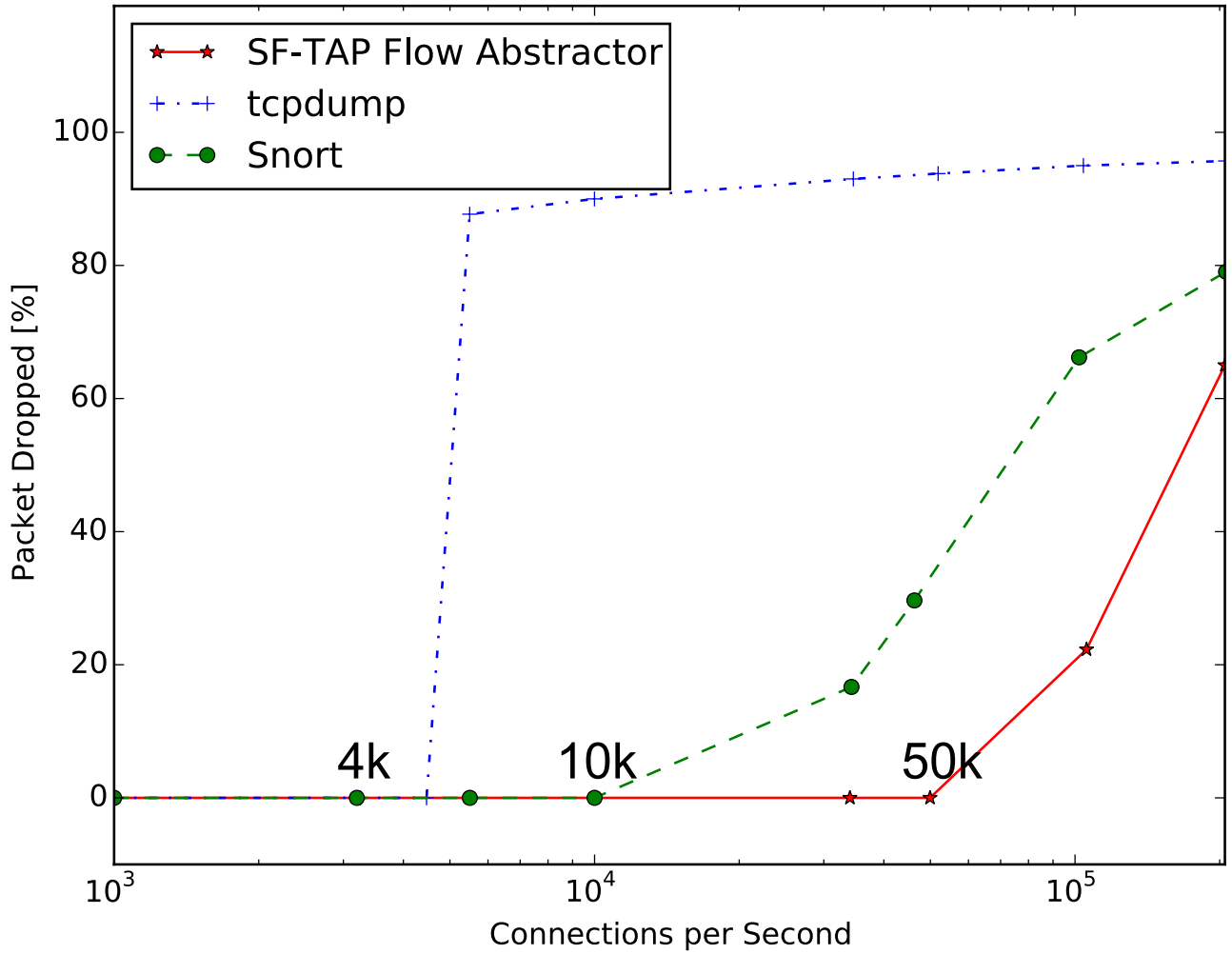


(c) HTTP Analyzer x 4

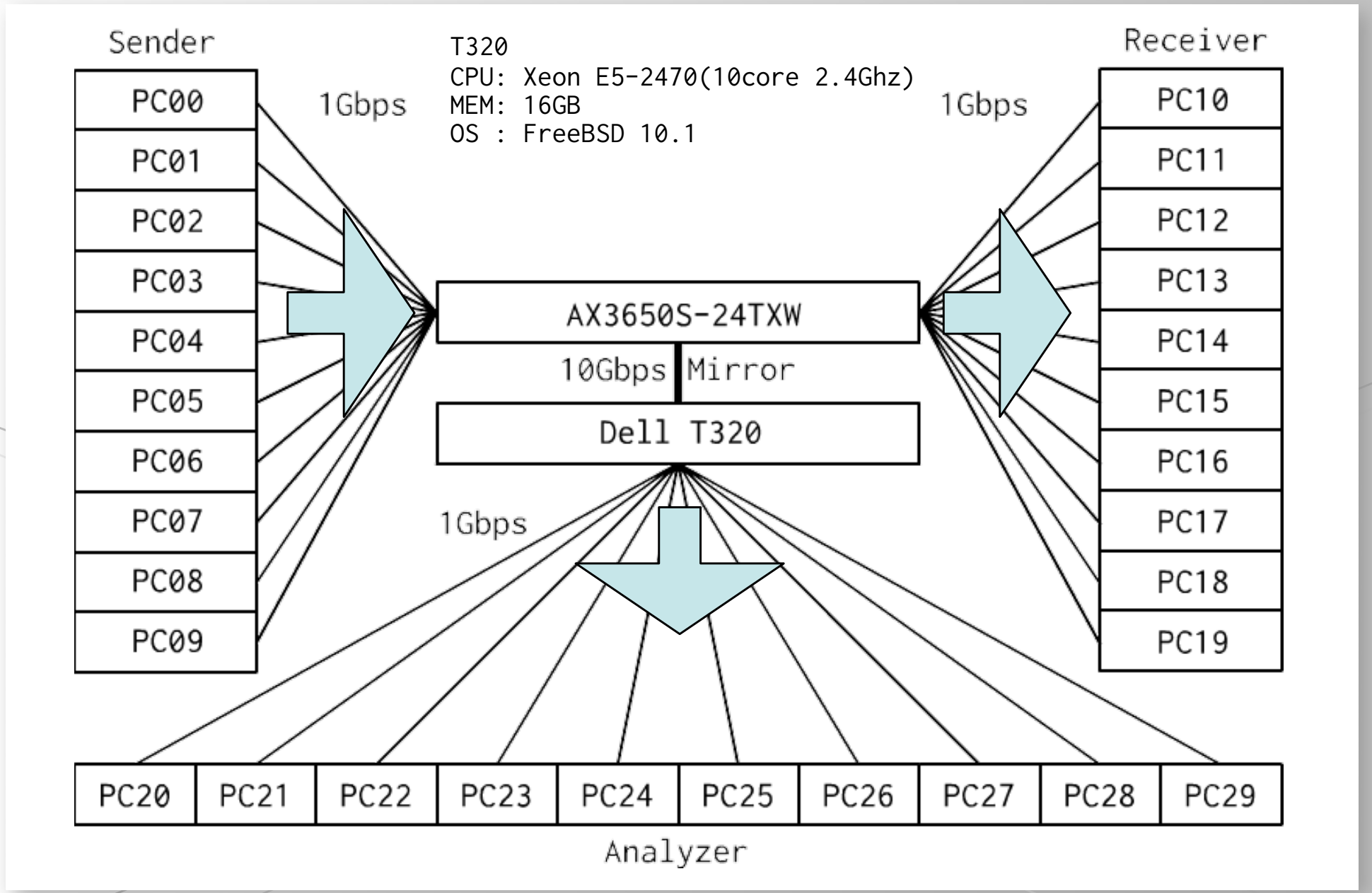
generate 50 clients / sec, 1000 clients maximum, 2500 requests / sec on average

Flow Abstractor の計測

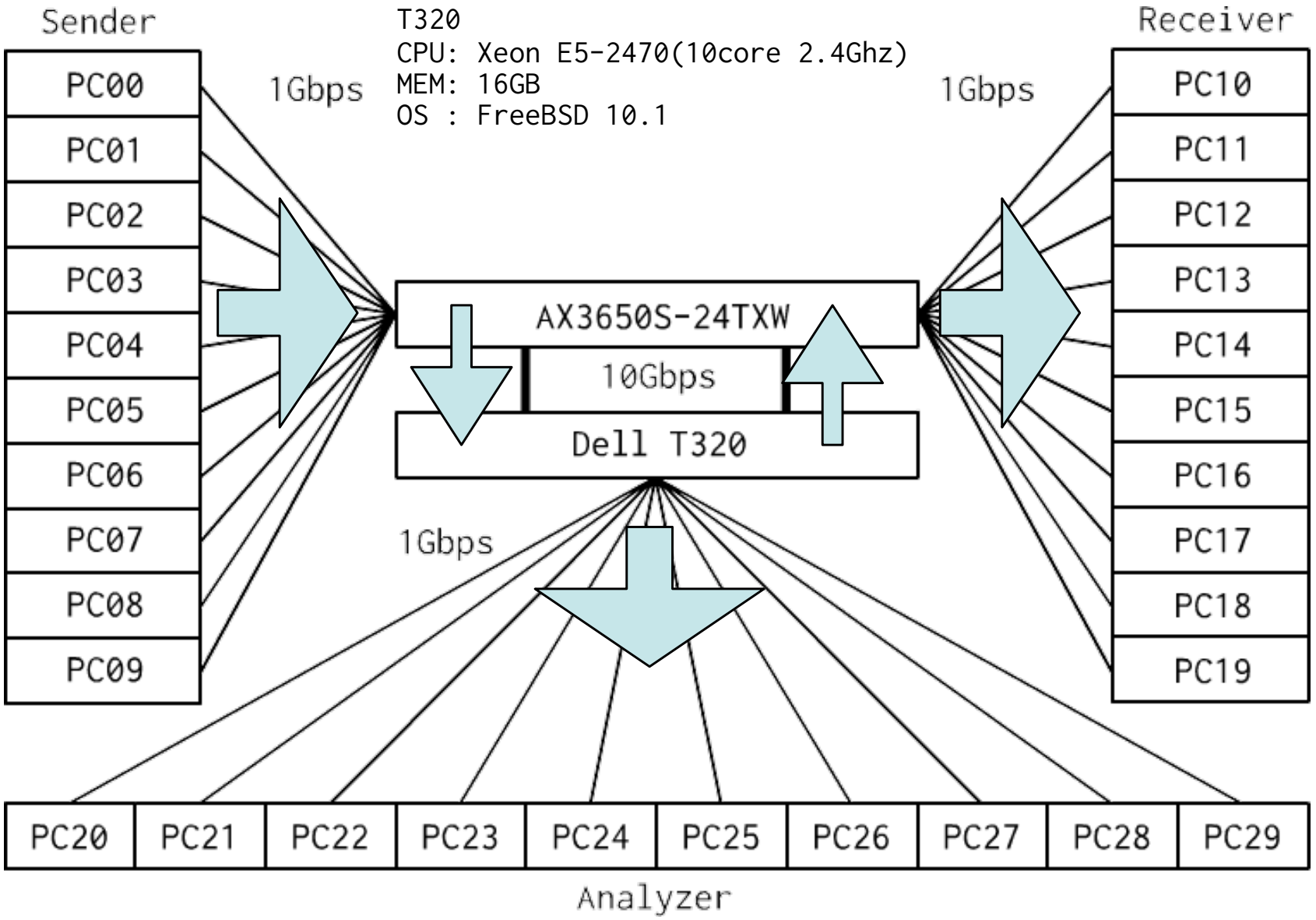
- コネクション/Secに対するパケットドロップ率(pcap)



Cell Incubator の計測

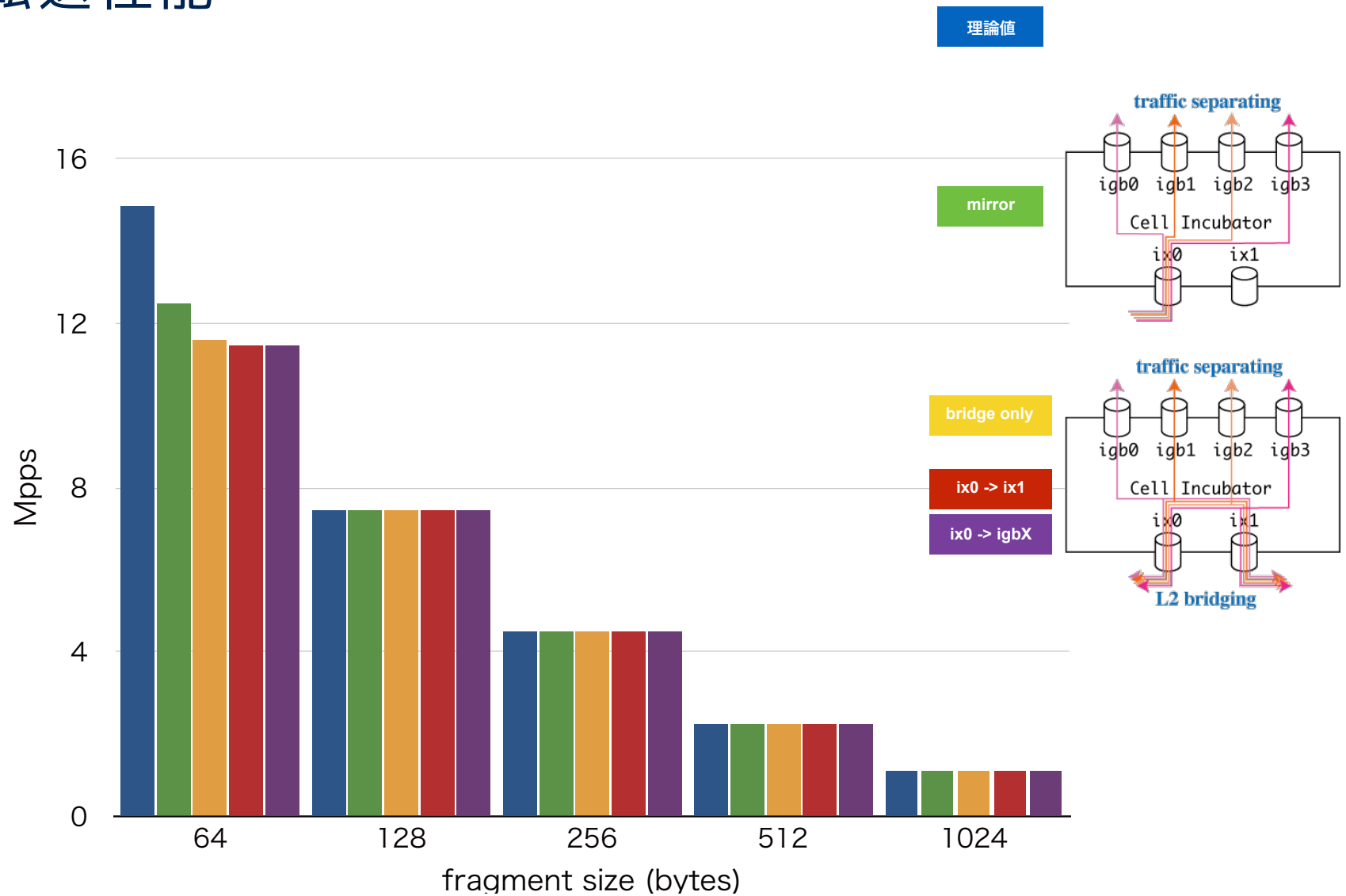


Cell Incubator の計測



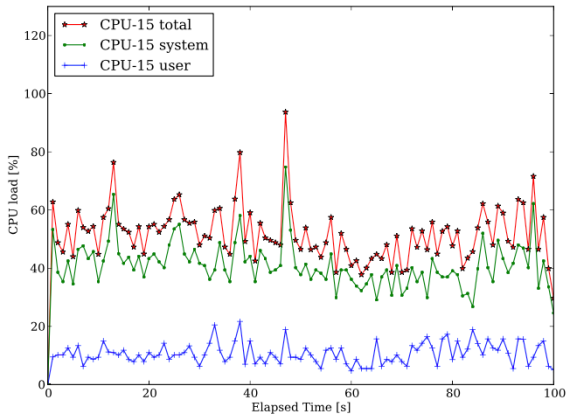
Cell Incubator の計測

- 転送性能

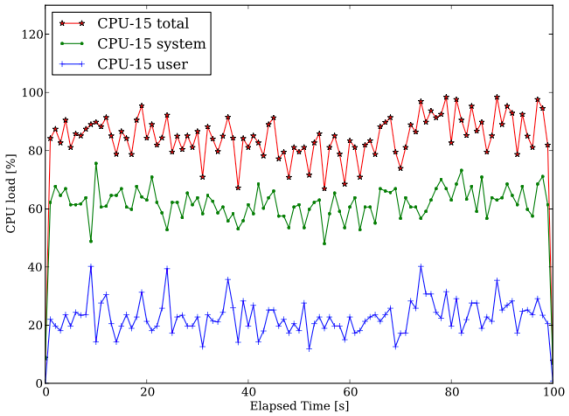


Cell Incubator の計測

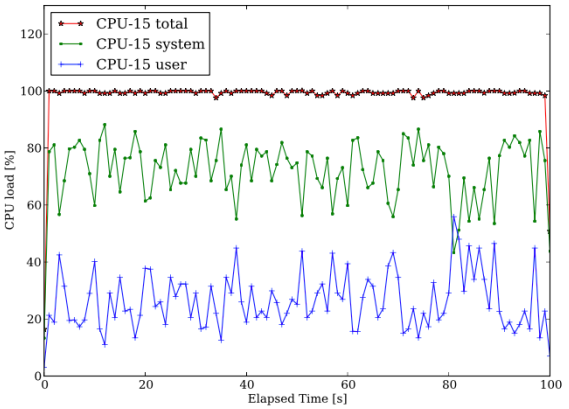
- CPU使用率
  - bridge＆flow-separate時のCPU15を抜粋



(a) 5.95 Mpps



(b) 10.42 Mpps



(c) 14.88 Mpps

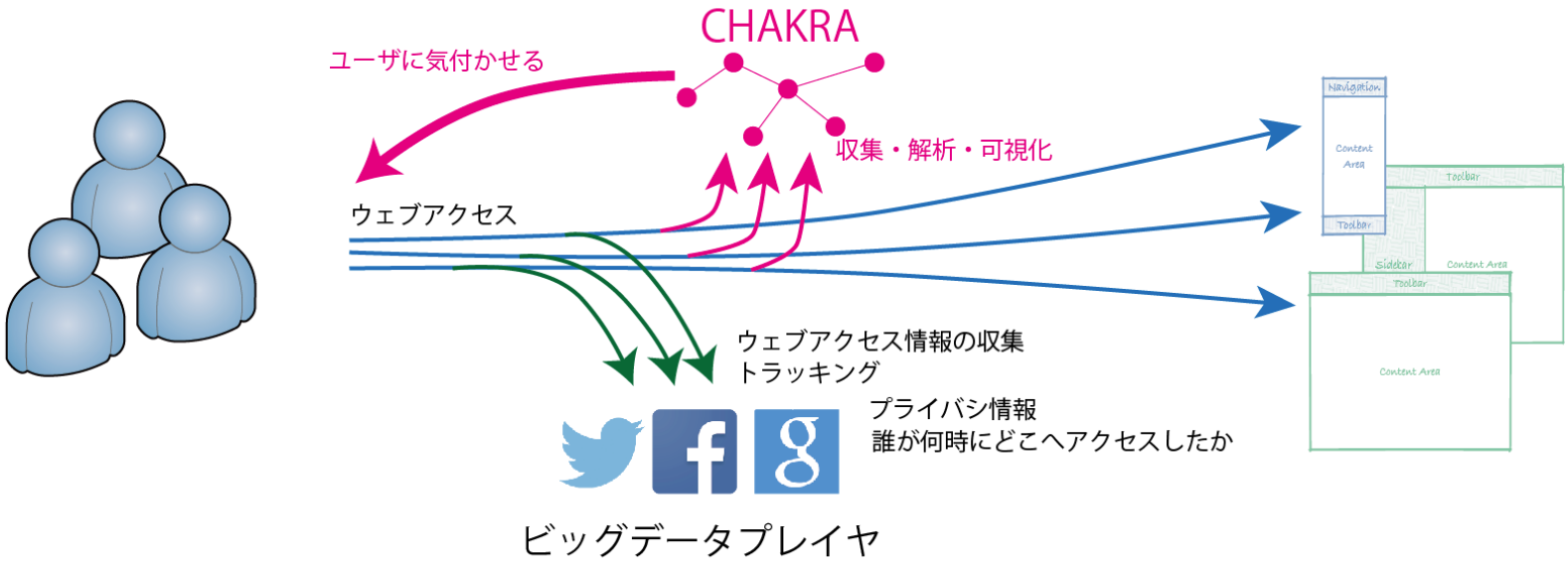
SF-TAPの利用事例

- CHAKRA
  - Third-party Web Trackingの可視化システム
  - PST2014, INTEROP2015

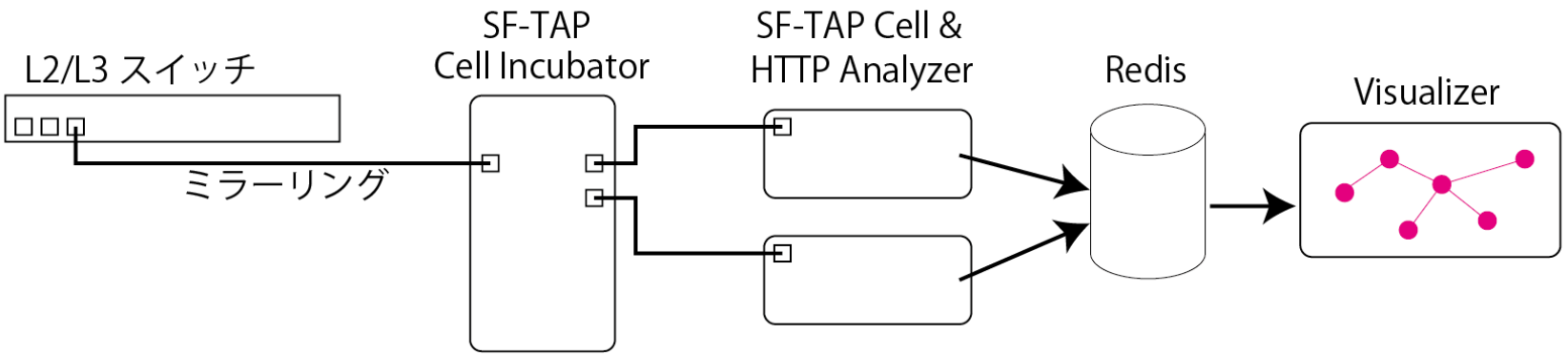
- 7 senses
  - L7コンテンツベースの経路制御システム
  - 沖縄オープンラボプログラミングコンテスト
  - 共同開発者
    - 栗原良尚（NTTコミュニケーション株式会社）

- トラフィック可視化
  - elastic searchへ情報を保持させkibanaで表示

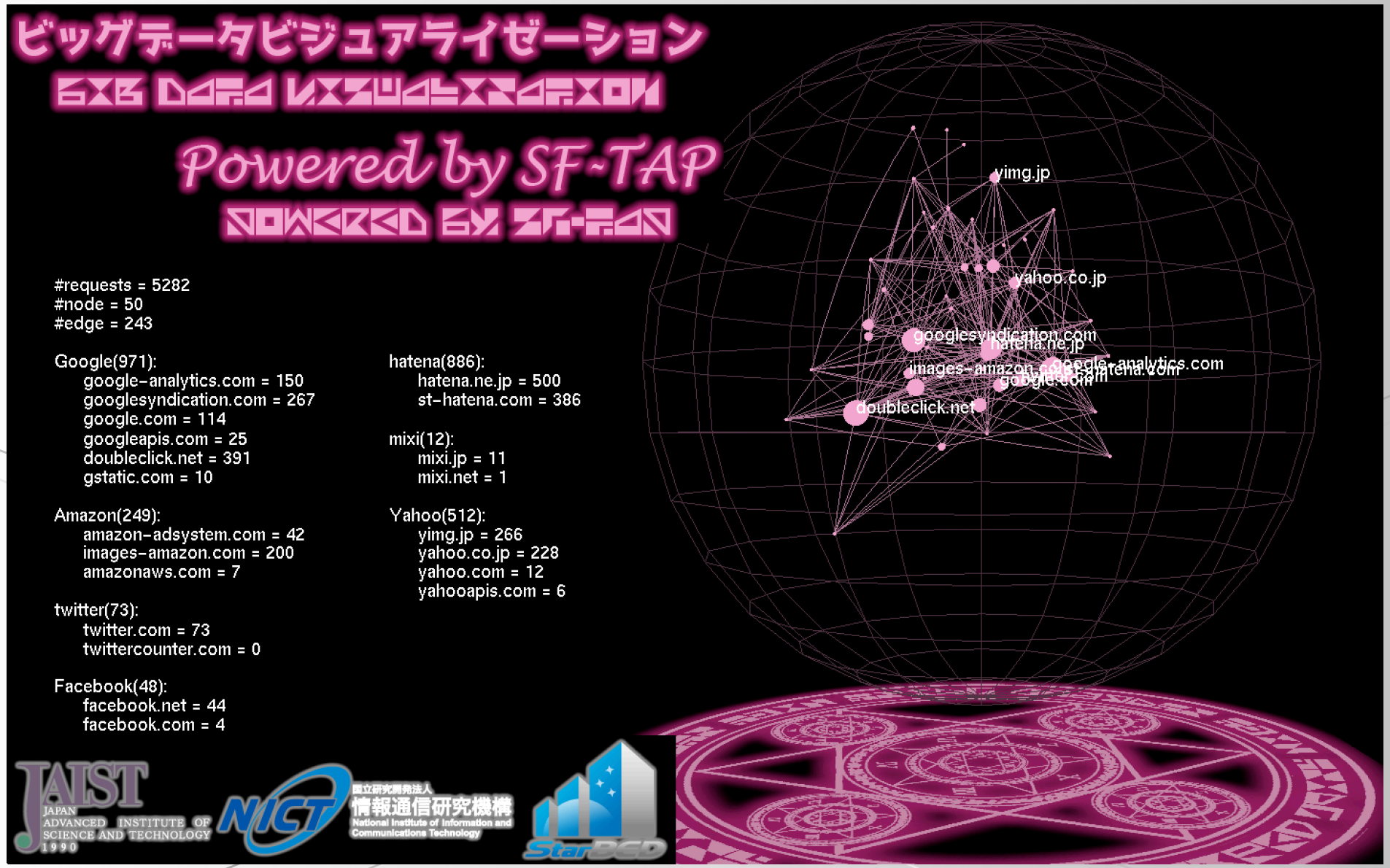
-Third-party Web Trackingの可視化システム



構成

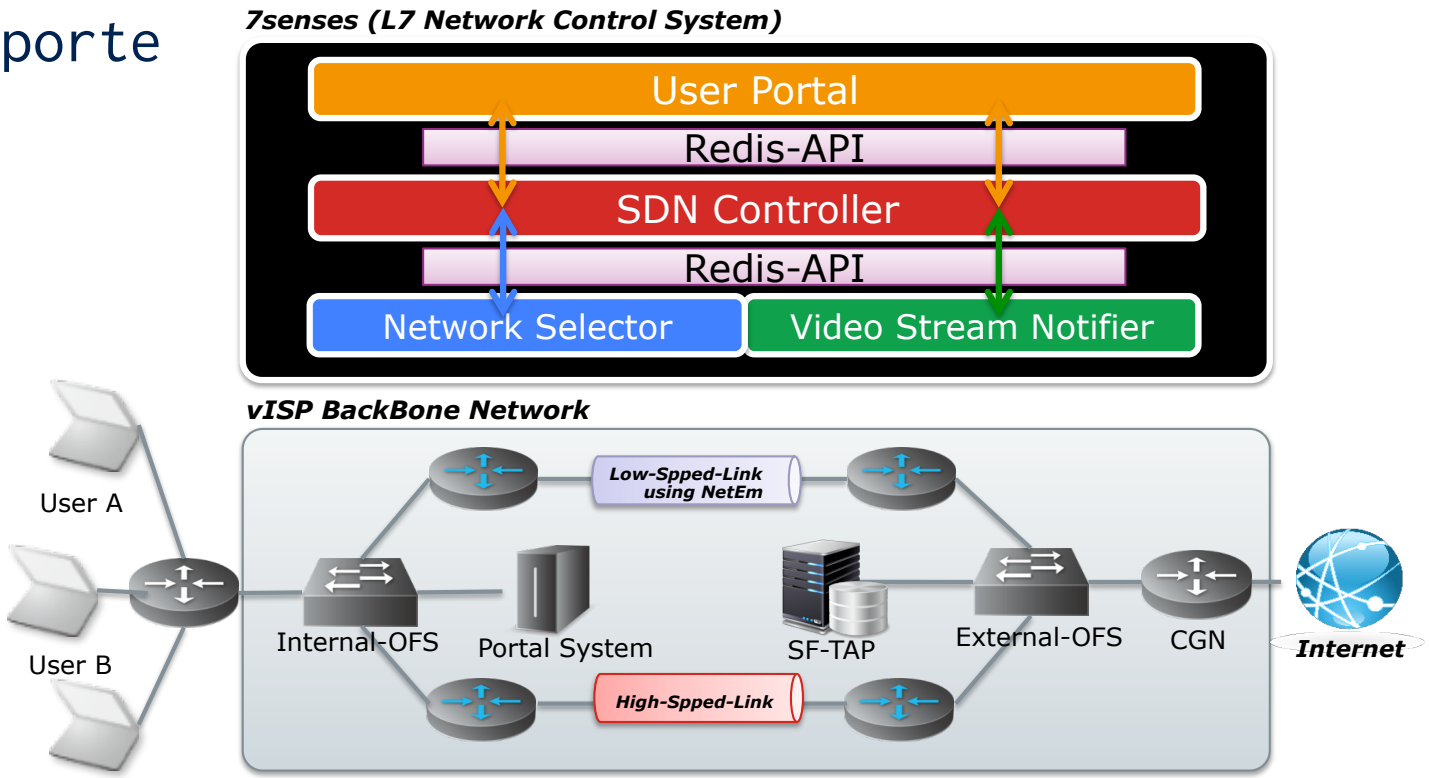


CHAKRA(movie)

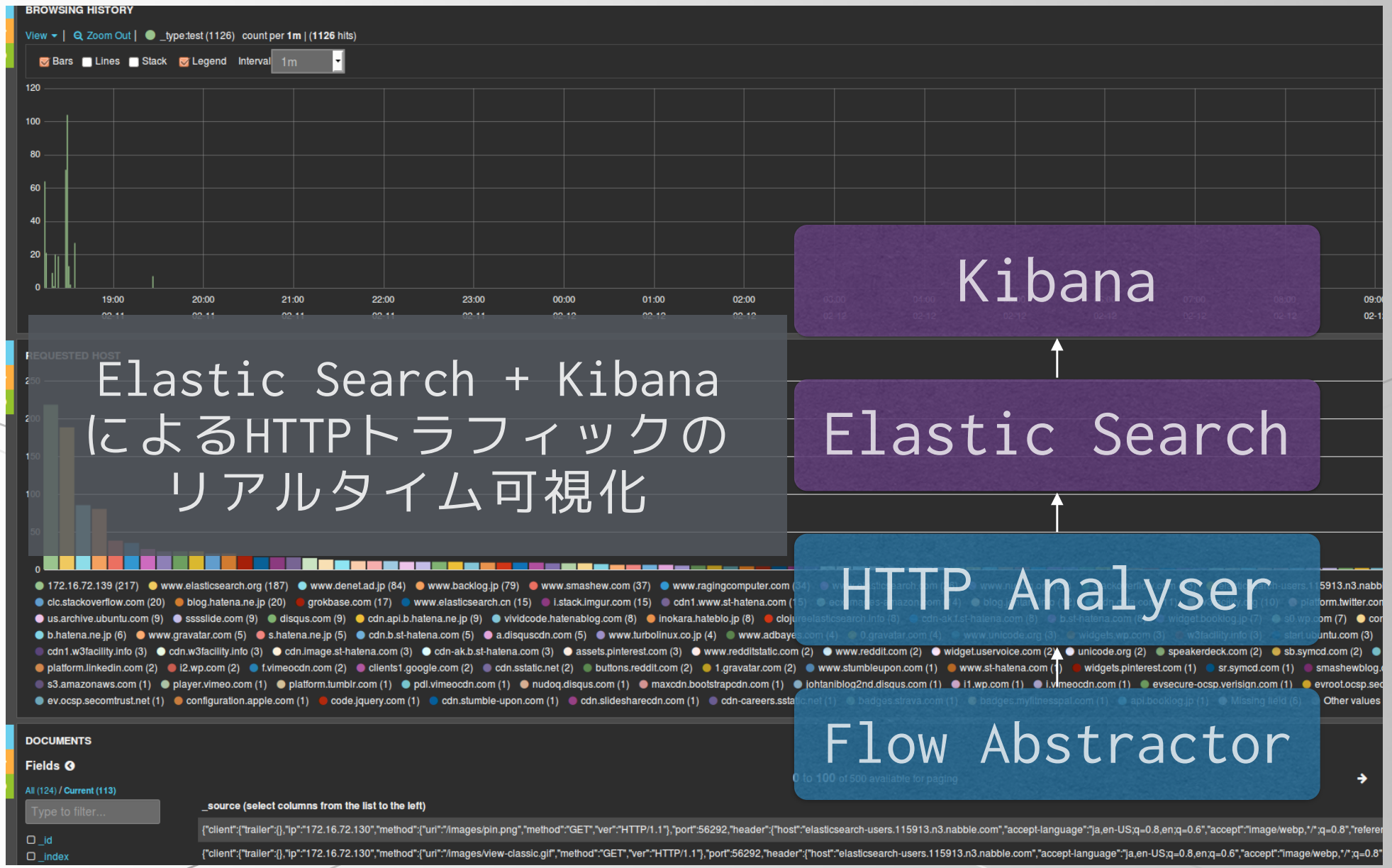


7Senses

- L7コンテンツベースの経路制御システム
  - Movieファイルのパス切り替え
    - OpenFlow : Open vSwitch
    - controller : ryu
    - DPI : sf-tap
    - CGN : natporte



WIDE合宿


BROWSING HISTORY
Kibana
Elastic Search + Kibana
によるHTTPトラフィックの
リアルタイム可視化
Elastic Search
HTTP Analyser
Flow Abstractor
DOCUMENTS
Fields
_source (select columns from the list to the left)

いま開発中なこと

- SSL対応
  - 指定したノード内の暗号通信をフックし
    flow-abstractorへ転送するプログラムを作成
- 自律的な負荷分散
  - 負荷を見ながら解析器と経路を調節する機構
- ロードバランサ
  - PC内でNetworkIFへのフローロードバランサ
  - DSRロードバランサ
- プログラムの高速化
  - SIMD化，TSX化，slab-allocator，copy回数を減らす

おわり

- ホームページあります！
  - http://sf-tap.github.io
  - 環境設定や使い方の説明等

- その他いろいろメールまってます！