

## Open Networking Conference 2019

# あなた好みの中継ルータ

2019/10/31

古河ネットワークソリューション株式会社

羽田野 文也

# 自己紹介

## ■ 氏名

- 羽田野 文也

## ■ 所属

- FNS 開発本部 技術企画開発部 先端技術開発グループ
- 色々と新しいことをやっています

## ■ 主に今までやってきたこと

- ルータ装置の運用関連機能の開発
- 構成管理用サーバシステムの開発/運用管理

# ルータ上でFLOSSなアプリを使うメリット

## ■ ユーザ価値の観点から考えると...

価値	アプリの種類	具体的なメリット
設備コスト 低減	<ul style="list-style-type: none"><li>システム管理ツール</li><li>FW</li></ul>	<ul style="list-style-type: none"><li>TCOを下げられる。機器代・ソフト代に加え電気代・ラック費用・保守費等</li><li>管理者が少なくて済む</li></ul>
運用コスト 低減	<ul style="list-style-type: none"><li>構成管理ツール</li><li>ZTP</li><li>IDS</li></ul>	<ul style="list-style-type: none"><li>自動化して人員を減らせる</li><li>人的ミスを減らせる</li><li>セキュリティ対策が可能</li></ul>
トラブル 防止	<ul style="list-style-type: none"><li>ネットワークフロー解析</li><li>キャプチャ</li></ul>	<ul style="list-style-type: none"><li>現地に行かずトラブルシュートが可能</li><li>回線利用状況が把握できる</li><li>EUに報告しやすい</li></ul>
カスタマイズ	<ul style="list-style-type: none"><li>見える化</li><li>SE/キャリア開発アプリ</li></ul>	<ul style="list-style-type: none"><li>より高度なサービスを提供できる</li><li>機器を追加することなくカスタマイズ</li></ul>

- どんなアプリを動かしてみたいですか？
- どんなことをルータにやらせてみたいですか？



# いろいろなアプリケーションを動かしてみました

## ■ パケットキャプチャ

- tcpdump

## ■ ネットワークフロー解析

- ntopng, sflowd, hsfowd, packetbeat

## ■ プロキシサーバ

- squid

## ■ IDS

- snort

## ■ ログ解析

- filebeat

# コンテナ型仮想環境の操作

## ■ Ubuntuベースのシステムコンテナ

```
Router#container attach
root@container:~#
root@container:~# cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.3 LTS"
root@container:~# pstree -A
systemd+-accounts-daemon---2*[{accounts-daemon}]
|-agetty
|-atd
|-chronyd
|-cron
|-dbus-daemon
|-filebeat---8*[{filebeat}]
|-hsflowd---{hsflowd}
|-networkd-dispat---{networkd-dispat}
|-polkitd---2*[{polkitd}]
|-redis-server---3*[{redis-server}]
|-rsyslogd---2*[{rsyslogd}]
|-snort---{snort}
|-softflowd
|-squid---squid+-log_file_daemon
|   |-5*[{security_file_c}]
|   \-unlinkd
|-sshd
|-systemd-journal
|-systemd-logind
|-systemd-network
|-systemd-resolve
|-systemd-udev
\--unattended-upgr---{unattended-upgr}
root@container:~#
```

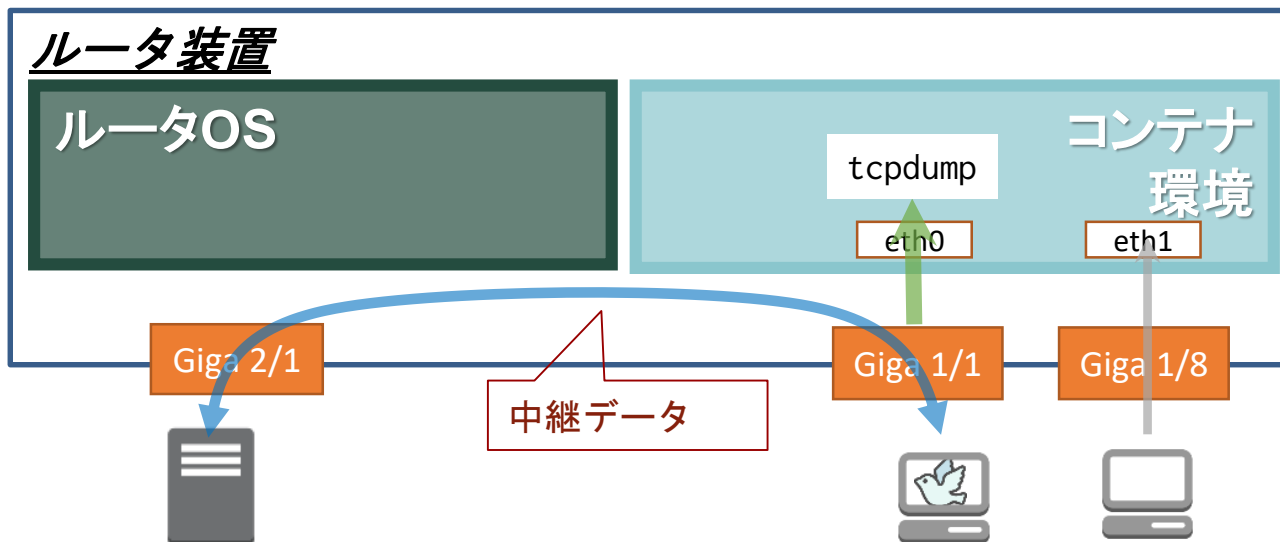
initデーモンを含めたOS全体が動作

複数のデーモンプロセスを同じコンテナ内で動作させる

# パケットキャプチャ

- ルータのポートモニタリング機能を利用すれば中継データのキャプチャも可能

```
root@container:~# tcpdump -i eth0
06:29:48.611087 STP 802.1d, Config, Flags [none], bridge-id 807f.b8:be:bf:06:dc:00.8004,
length 43
06:29:49.691734 IP 192.168.127.30.65474 > nrt12s13-in-f195.1e100.net.http: Flags [R.], seq
1679839507, ack 3054297526, win 0, length 0
06:29:49.691797 IP 192.168.127.30.65457 > 117.18.237.29.http: Flags [R.], seq 0, ack 1, win 0,
length 0
06:29:49.691911 IP 192.168.127.30.65455 > 151.139.128.14.http: Flags [R.], seq 4273249763, ack
2972801286, win 0, length 0
06:29:50.610004 STP 802.1d, Config, Flags [none], bridge-id 807f.b8:be:bf:06:dc:00.8004,
length 43
:
```



# パケットキャプチャ

## ■ SSHサービスを利用すればWiresharkでみることも可能

- コンテナ環境にGUIはないので外部端末から実行

```
wireshark -k -i <(ssh root@192.168.127.21 "tcpdump -U -n -w - -i eth0 vlan and not port ssh")
```

The screenshot displays the Wireshark interface with a packet capture of an HTTP response. The packet list pane shows a packet of type HTTP with status 200 OK. The packet details pane shows the structure of the packet, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The packet bytes pane shows the raw hex and ASCII data.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	158.202.234.31	10.10.30.10	TCP	70	9200 → 48078 [ACK] Seq=1 Ack=1 Win=2171 Len=0 TSval=769346823 TSecr=2645896832
2	0.000036	158.202.234.31	10.10.30.10	TCP	70	9200 → 48078 [ACK] Seq=1 Ack=2649 Win=2164 Len=0 TSval=769346823 TSecr=2645896832
3	0.000054	158.202.234.31	10.10.30.10	TCP	70	9200 → 48078 [ACK] Seq=1 Ack=3113 Win=2161 Len=0 TSval=769346823 TSecr=2645896832
4	0.028299	158.202.234.31	10.10.30.10	HTTP	423	HTTP/1.1 200 OK (application/json)
5	0.533585	Cisco_06:dc:04	Spanning-tree-(for-...	STP	64	Conf. Root = 32768/127/b8:be:bf:06:dc:00 Cost = 0 Port = 0x8004
6	0.890754	192.52.127.30	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
7	0.913001	10.10.30.1	10.10.30.10	SNMP	442	get-response 1.3.6.1.2.1.31.1.1.1.1.10.401070000 1.3.6.1.2.1.31.1.1.1.11.401070000 1.3.6.1.2.1.31.1.1.
8	1.692692	158.202.234.31	10.10.30.10	TCP	70	9200 → 47232 [ACK] Seq=1 Ack=1 Win=1432 Len=0 TSval=769348515 TSecr=2645898525
9	1.720373	158.202.234.31	10.10.30.10	HTTP	243	HTTP/1.1 200 OK (application/json)
10	1.952522	158.202.234.31	10.10.30.10	HTTP	405	HTTP/1.1 200 OK (application/json)

Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0  
Ethernet II, Src: Furukawa\_f0:5a:72 (00:80:bd:f0:5a:72), Dst: Furukawa\_f0:5a:76 (00:80:bd:f0:5a:76)  
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 50  
Internet Protocol Version 4, Src: 158.202.234.31, Dst: 10.10.30.10  
Transmission Control Protocol, Src Port: 9200, Dst Port: 48078, Seq: 1, Ack: 1, Len: 0  
Source Port: 9200  
Destination Port: 48078  
[Stream index: 0]  
[TCP Segment Len: 0]  
Sequence number: 1 (relative sequence number)  
[Next sequence number: 1 (relative sequence number)]  
Acknowledgment number: 1 (relative ack number)

```
0000 00 80 bd f0 5a 76 00 80 bd f0 5a 72 81 00 00 32  ...Zv...Zr...2
0010 08 00 45 00 00 34 65 98 40 00 3e 06 26 2e 9e ca  ...E...4e...@->.&...
0020 ea 1f 0a 0a 1e 0a 23 f0 bb ce 58 16 0a b9 53 bd  ...#...X...S...
0030 48 2d 80 10 08 7b 97 b3 00 00 01 01 08 0a 2d db  H...{...
0040 4d 07 9d b5 2e 80
```

# ネットワークフロー解析(ntopng)

- DPI機能を使用して様々なネットワークプロトコルを判別可能

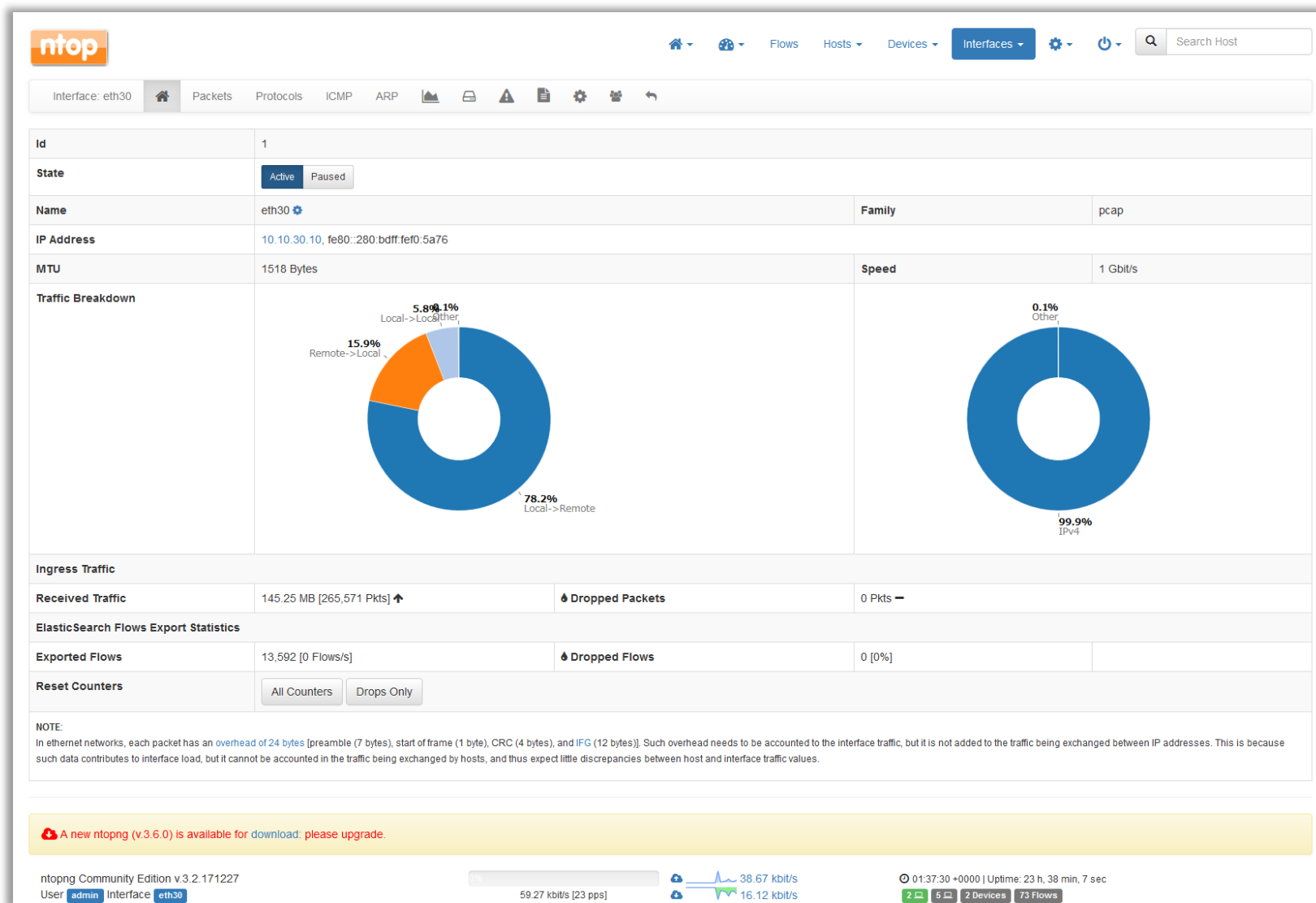
```
root@container:~# apt show ntopng
Package: ntopng
Version: 3.2+dfsg1-1
Priority: extra
Section: universe/net
Origin: Ubuntu
(省略)
root@container:~# apt install ntopng
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
(省略)
Setting up ntopng (3.2+dfsg1-1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
root@container:~# systemctl start ntopng
root@container:~#
```





# ネットワークフロー解析(ntopng)

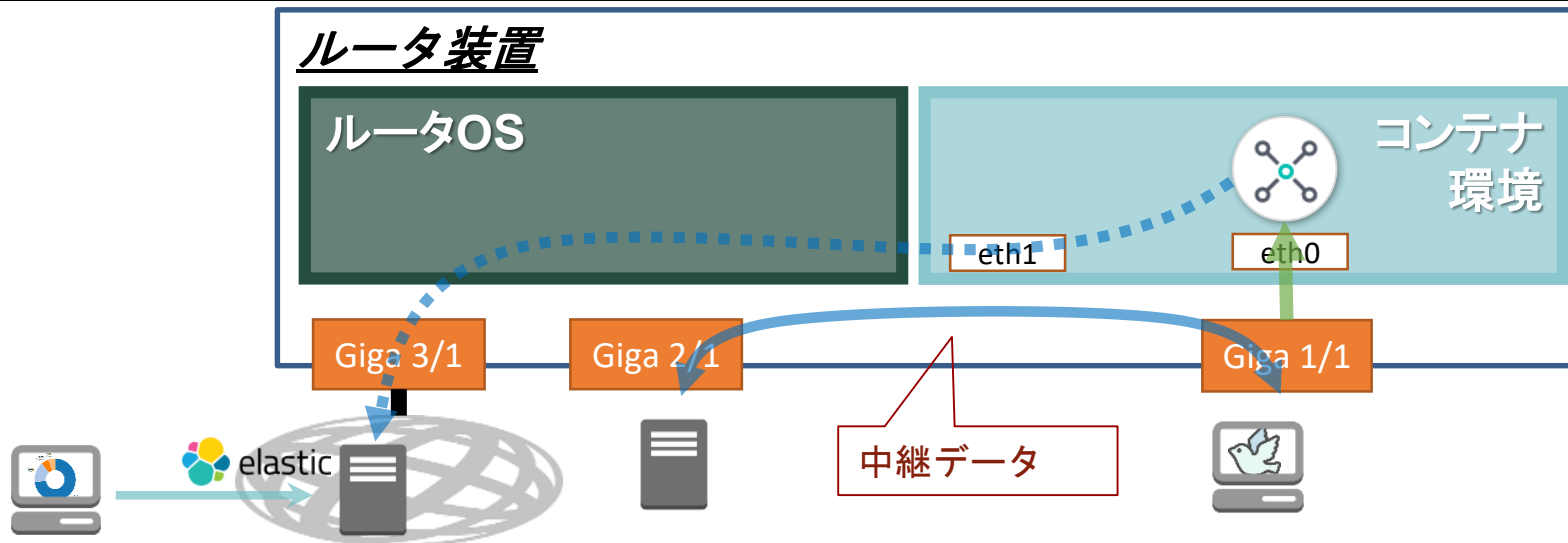
## ■ NtopngのWebUIで解析結果を確認



# ネットワークフローの見える化(packetbeat)

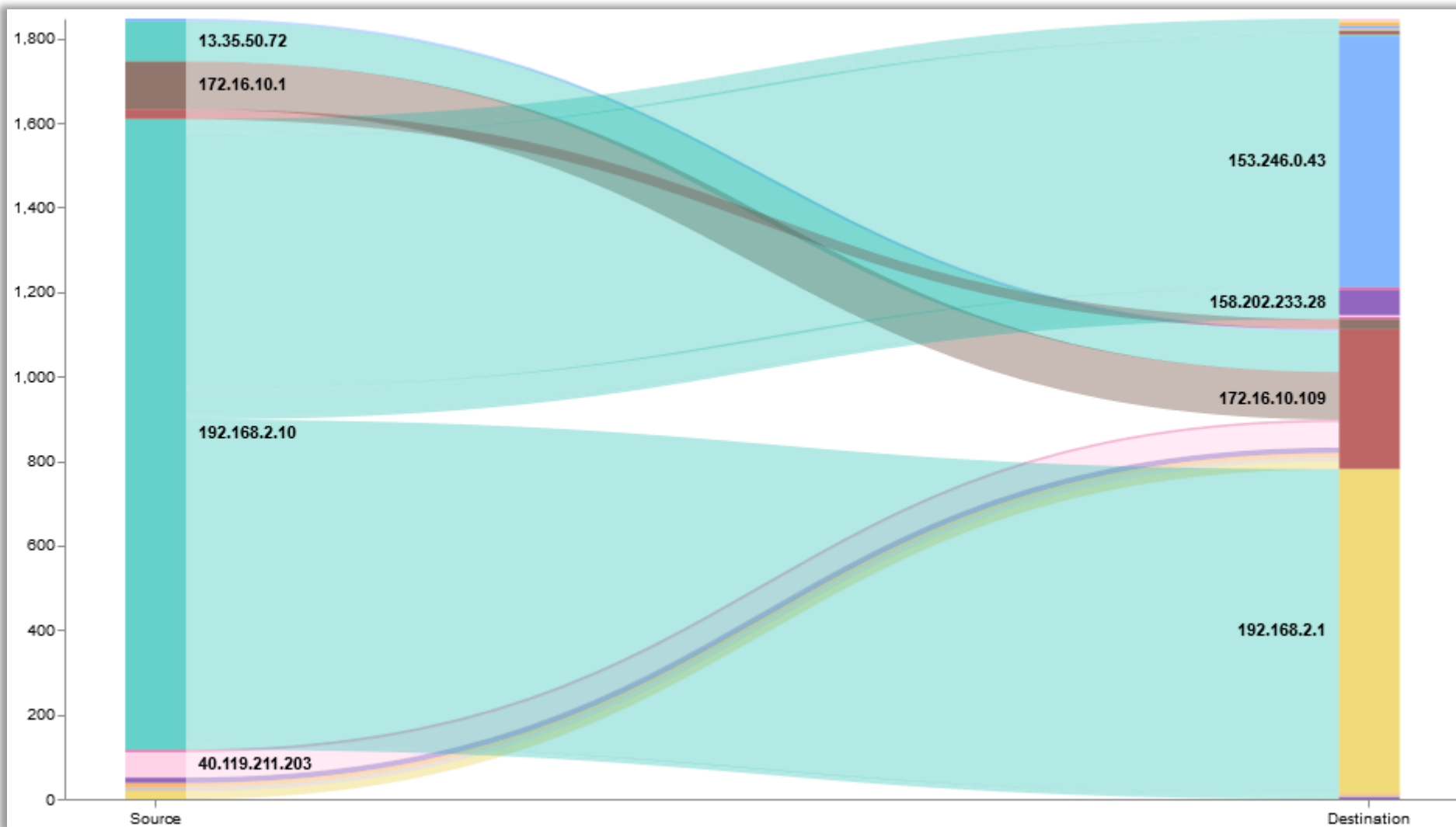
- Elastic社により開発されているパケットアナライザー
  - 外部サーバで動作するElasticsearchにデータを送信
  - Kibanaによるデータの可視化がリアルタイムに可能

```
root@container:~# systemctl start packetbeat
root@container:~# systemctl status packetbeat
packetbeat.service - Real-Time Packet Analyzer
  Loaded: loaded (/usr/lib/systemd/system/packetbeat.service; disabled; vendor preset:
  enabled)
  Active: active (running) since Sun 2019-10-20 09:04:14 UTC; 23h ago
  Docs: https://www.elastic.co/guide/en/beats/packetbeat/current/index.html
  Main PID: 7256 (packetbeat)
  Tasks: 9 (limit: 4275)
  CGroup: /system.slice/packetbeat.service
          mq7256 /usr/bin/packetbeat -c /etc/packetbeat/packetbeat.yml -path.home
  /usr/share/packetbeat -path.config /etc/packetbeat -path.data /var/lib/packetbeat -path.logs
  /var/log/packetbeat
```



# Packetbeatによるフローの見える化

■ Packetbeatから受信したデータをもとに可視化

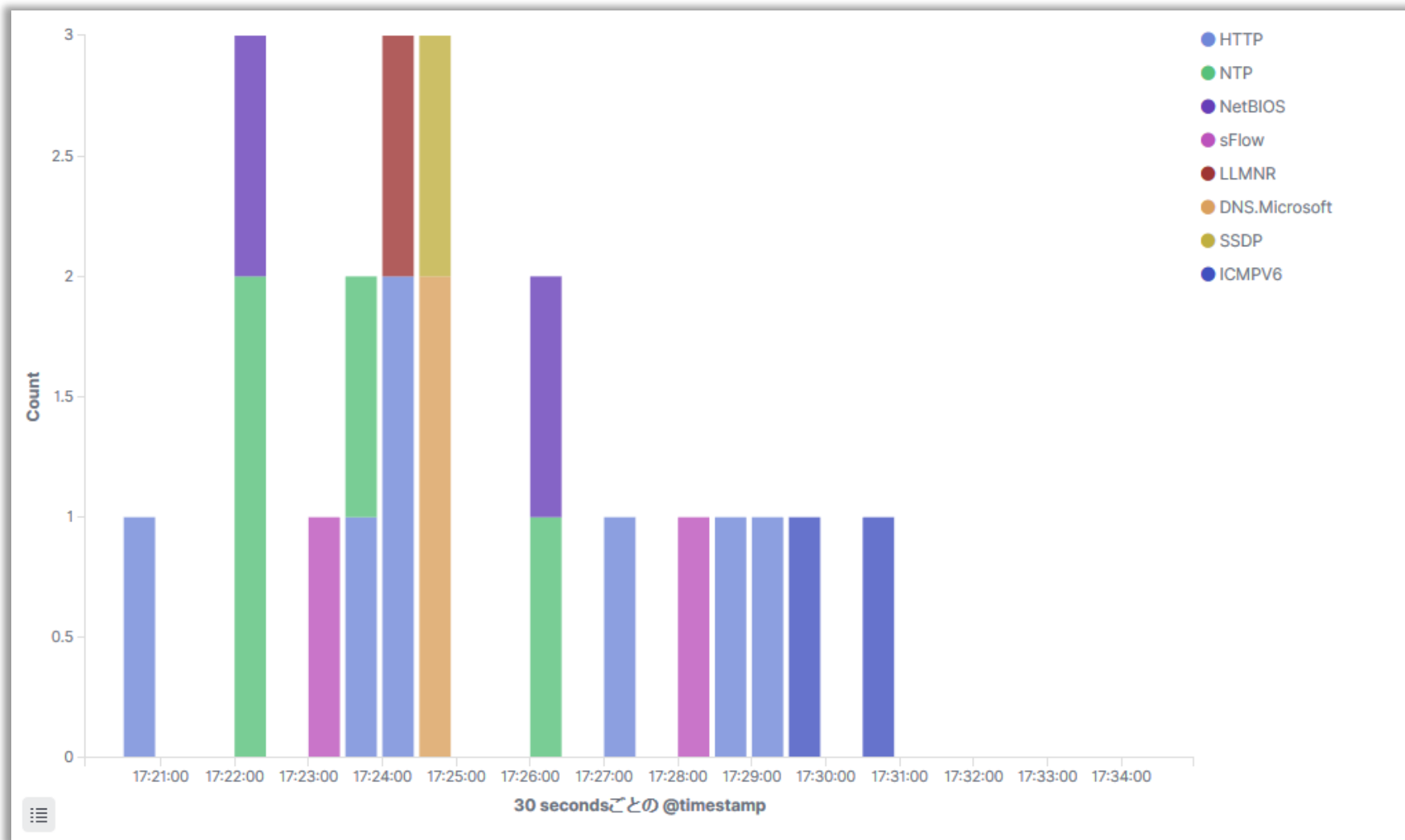


# ntopngによるフローの見える化

## ■ Ntopngの解析結果もKibanaで可視化

Dump expired flowsの設定

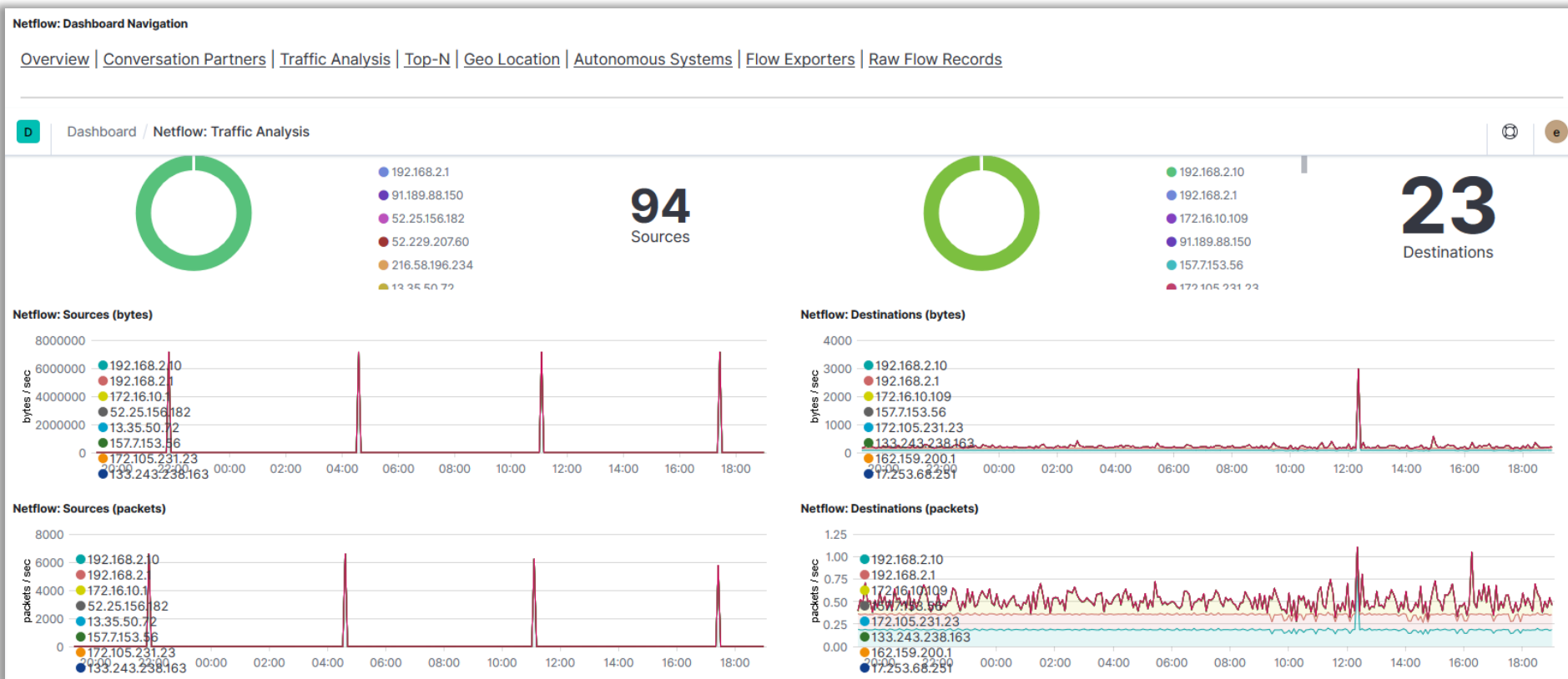
```
-F=es;_doc;ntopng-3.2;http://xxx.xxx.xxx.xxx:9200/_bulk;username:password
```



# Netflowデータの可視化

## ■ Softflowdを使用してNetflowのデータを外部サーバへ送信

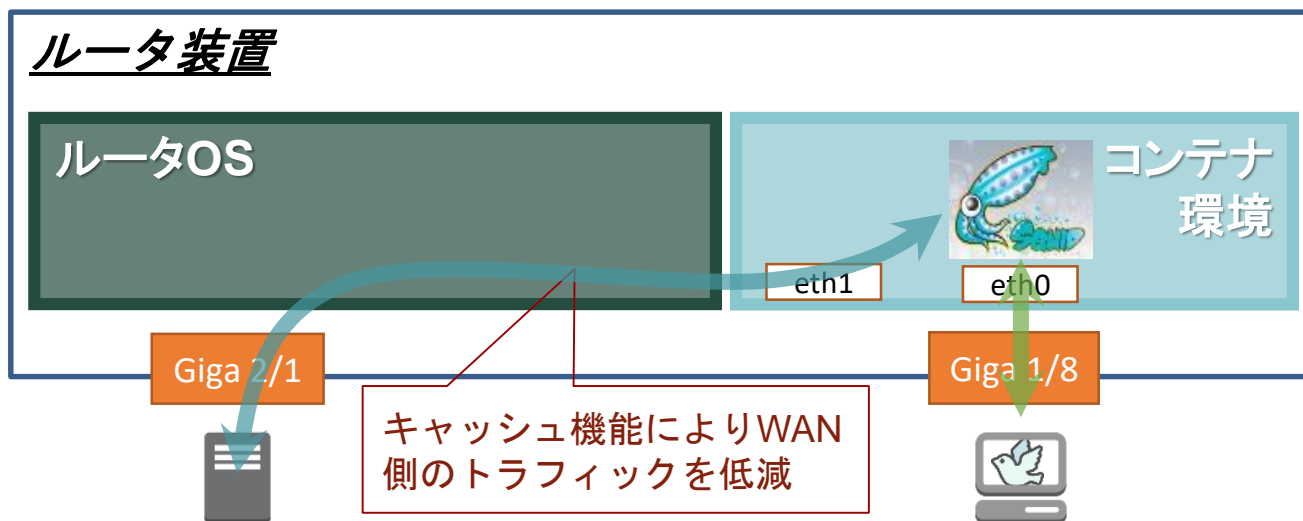
```
root@container:~# softflowd -T ether -i eth0 -v 9 -n xxx.xxx.xxx.xxx:2055
```



# プロキシサーバ

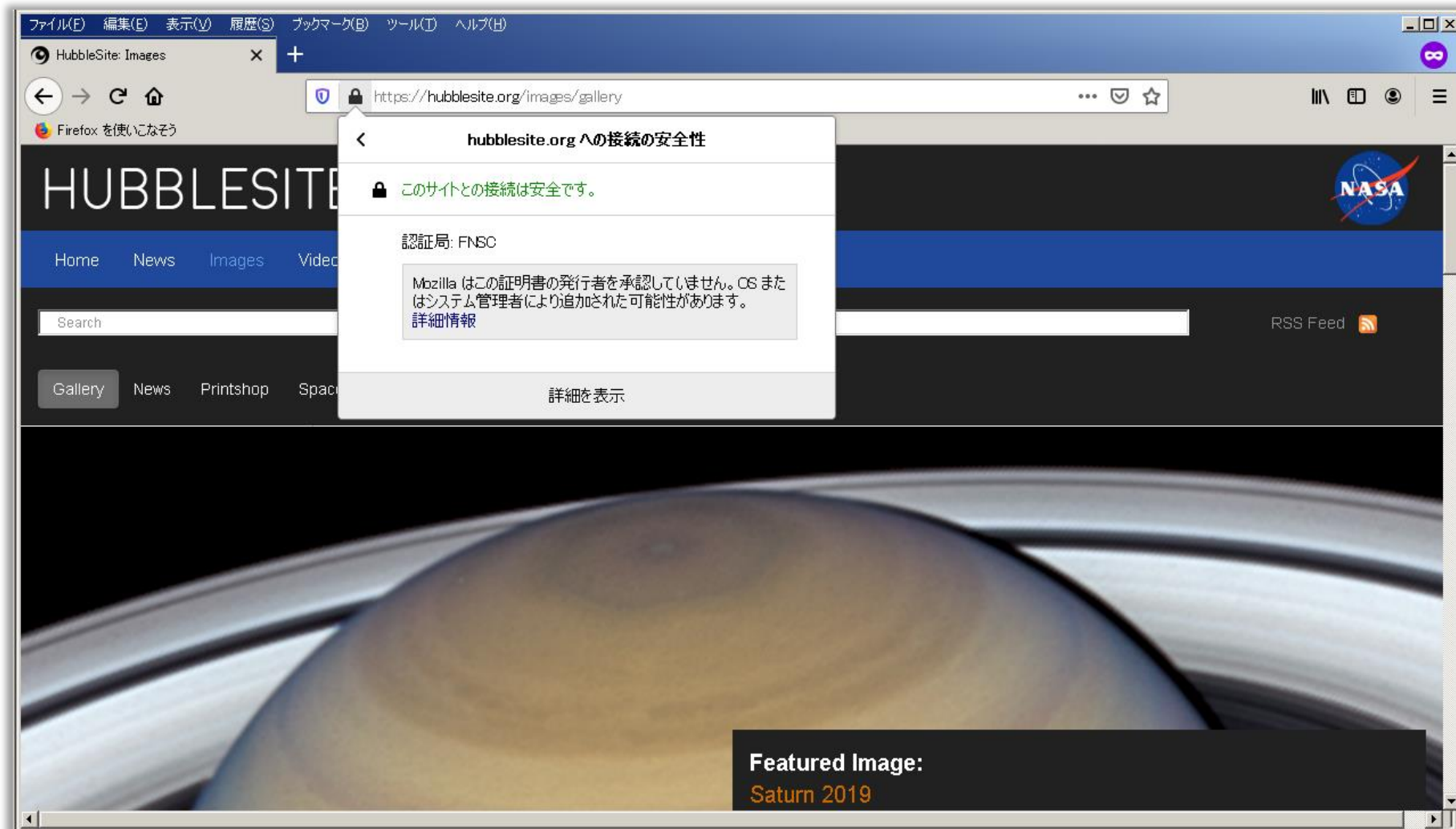
## ■ Squidを使用したプロキシ & キャッシュサーバ

```
root@container:~# systemctl start squid
root@container:~# iptables -t nat -A PREROUTING -i eth1 -p tcp -m tcp --dport 80 -j
REDIRECT --to-ports 3128
root@container:~# iptables -t nat -A PREROUTING -i eth1 -p tcp -m tcp --dport 443 -j
REDIRECT --to-ports 3128
root@container:~# tail /var/log/squid/access.log
1572202844.819      60 192.168.127.30 NONE/200 0 CONNECT 172.217.161.36:443 --
ORIGINAL_DST/172.217.161.36 -
1572202844.823    1021 192.168.127.30 TCP_MISS/200 49334 GET
https://media.stsci.edu/uploads/image/thumbnail/4316/low_STScI-H-1912b-t407x400.png -
ORIGINAL_DST/130.167.167.5 image/png
1572202844.965     67 192.168.127.30 TCP_MISS/302 867 GET https://www.google.com/ads/ga-
audiences? - ORIGINAL_DST/172.217.161.36 text/html
1572202845.028    805 192.168.127.30 TCP_MISS/200 62557 GET
https://media.stsci.edu/uploads/image/thumbnail/4291/low_STScI-H-p1856a-t-400x400.png -
ORIGINAL_DST/130.167.167.5 image/png
```



# 透過型プロキシ

## ■ HTTPS通信はSquidとPC間でTLSセッションを確立



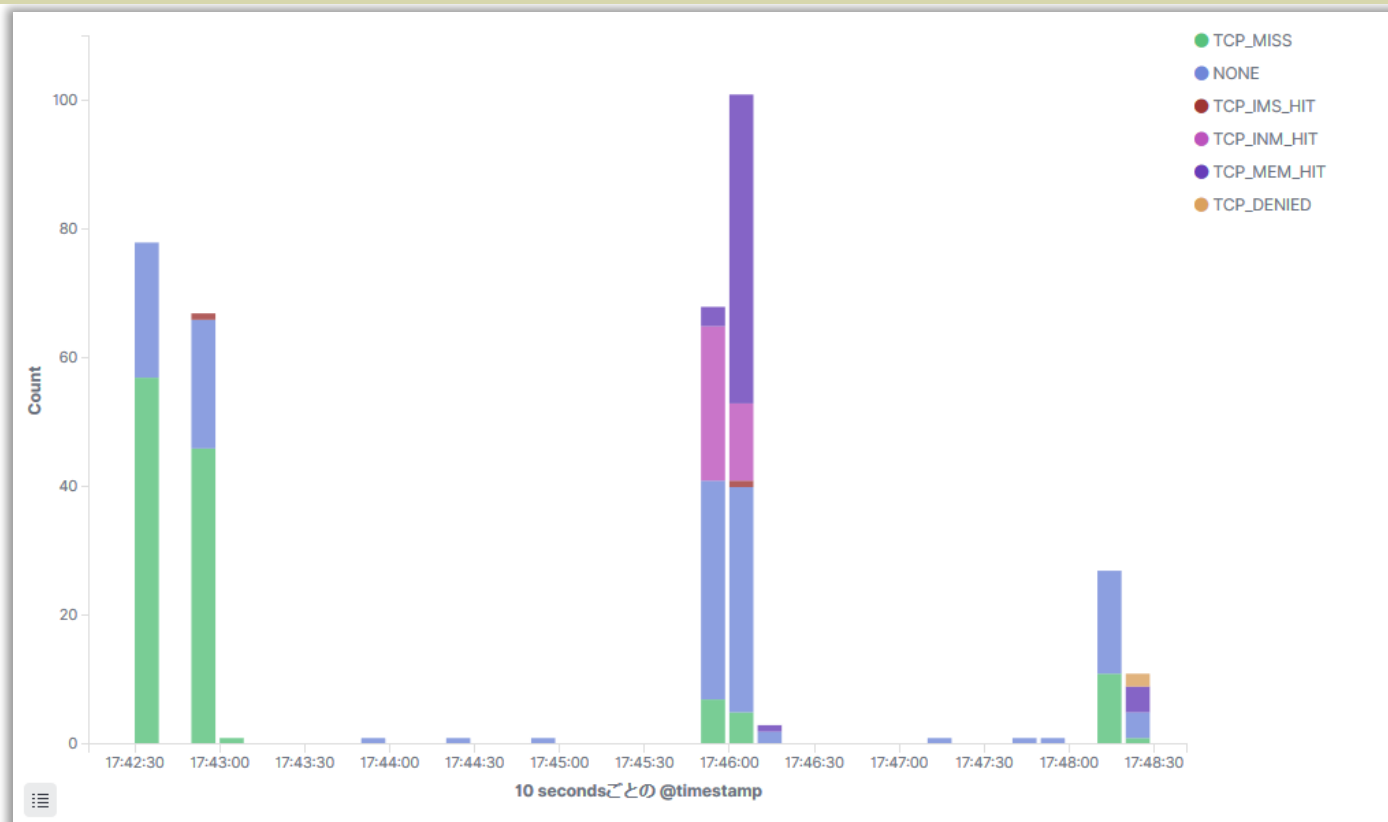
# Squidのアクセスログを可視化

## ■ Filebeatを使用してログファイルを外部サーバへ送信

```
root@container:~# systemctl start filebeat
```

### Grokフィルタの設定

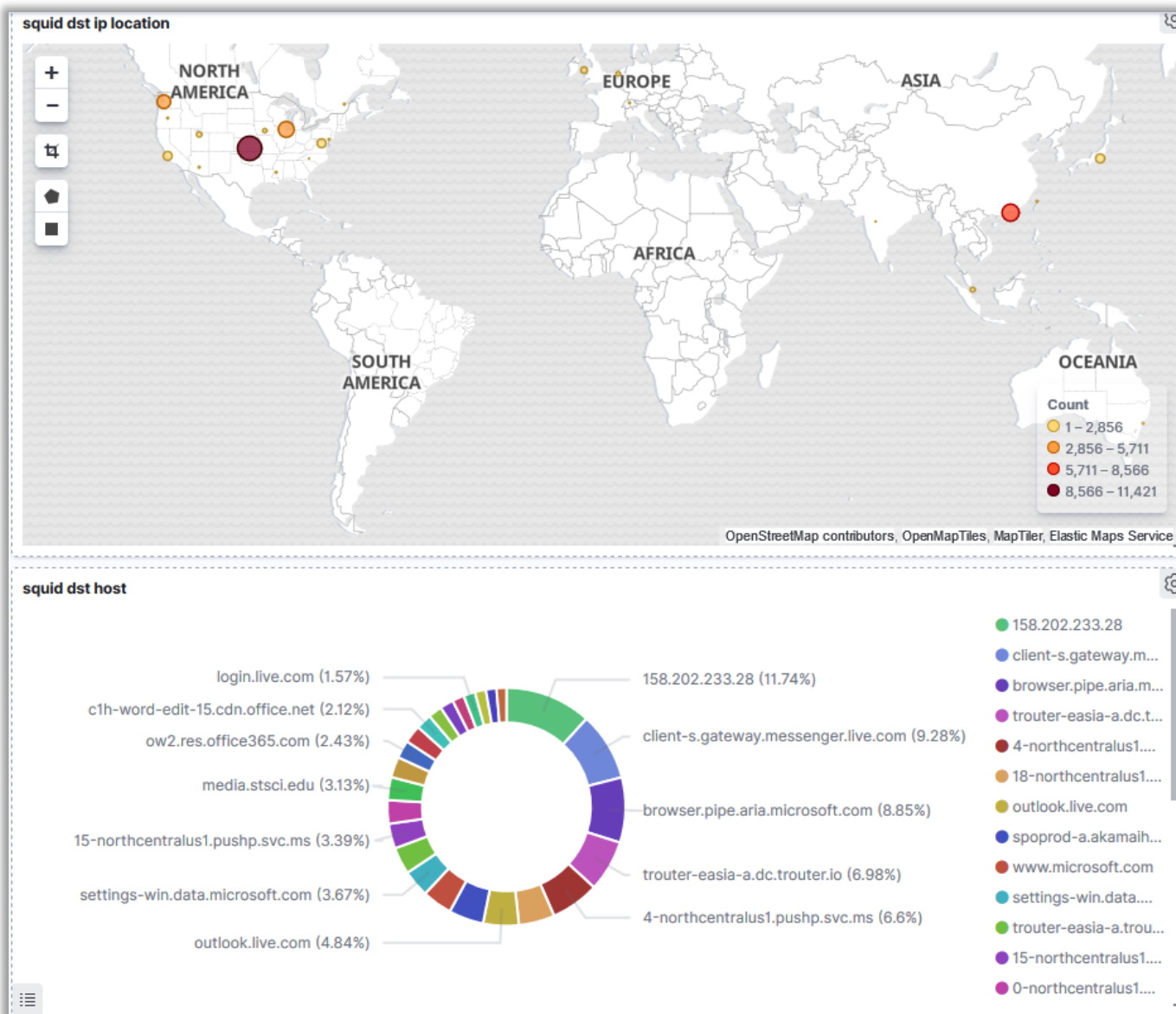
```
"%{NUMBER:squid_date} %{SPACE} %{NUMBER:squid_res_time} (?-  
| %{IP:squid_src_ip}) %{DATA:squid_req_stat} / %{NUMBER:squid_http_status} %{NUMBER:squid_repl  
y_size} %{DATA:squid_http_method} %{DATA:squid_http_protocol} : // %{IPORHOST:squid_dst_host} %  
{NOTSPACE:squid_req_url} (?- | %{NOTSPACE:squid_user}) %{DATA:squid_hier_code} / (?-  
| %{IP:squid_dst_ip}) %{NOTSPACE:squid_content_type}"
```





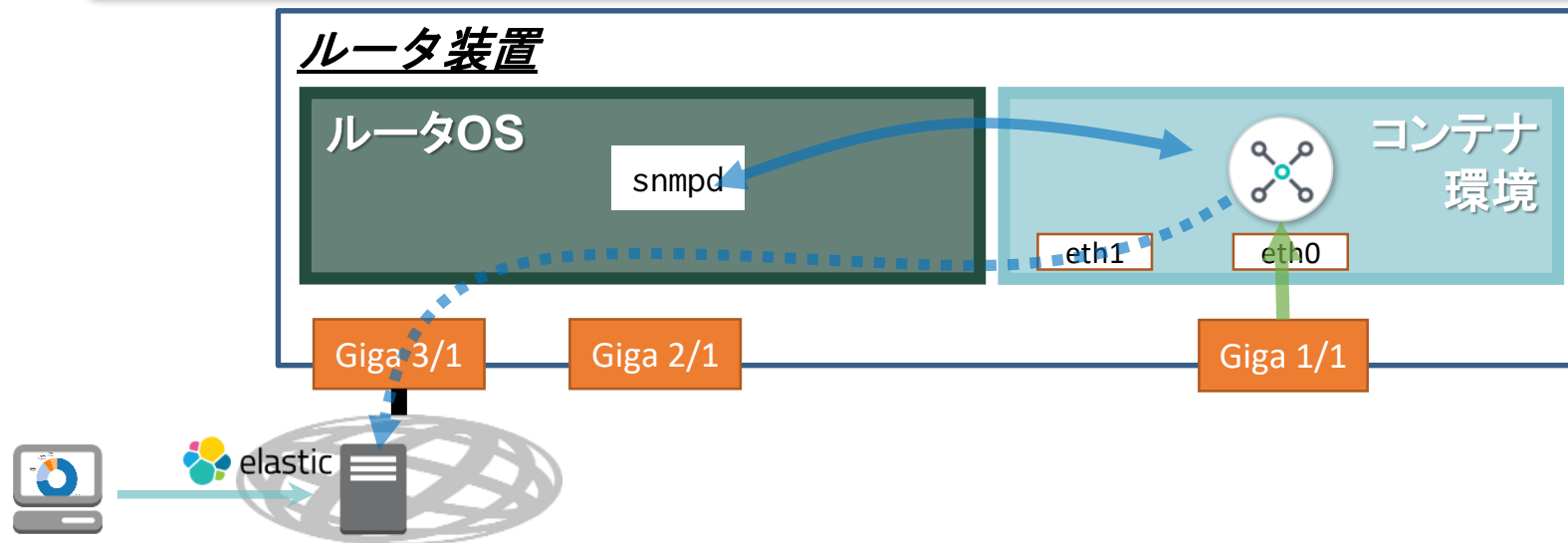
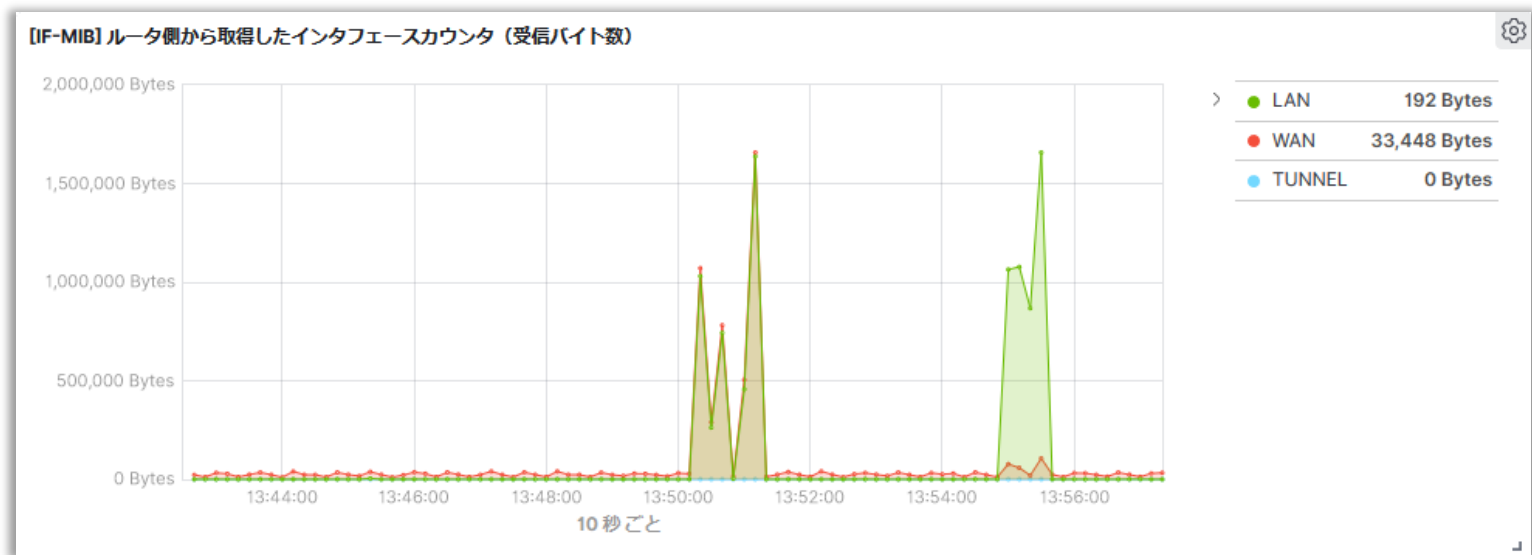
# Squidのアクセス先を可視化

## ■ GeoIPフィルタのプラグインを使用して地図上に可視化



# インターフェースカウンタの可視化

- ルータOS側から取得したMIBデータをもとに可視化
  - データはfilebeatで送信



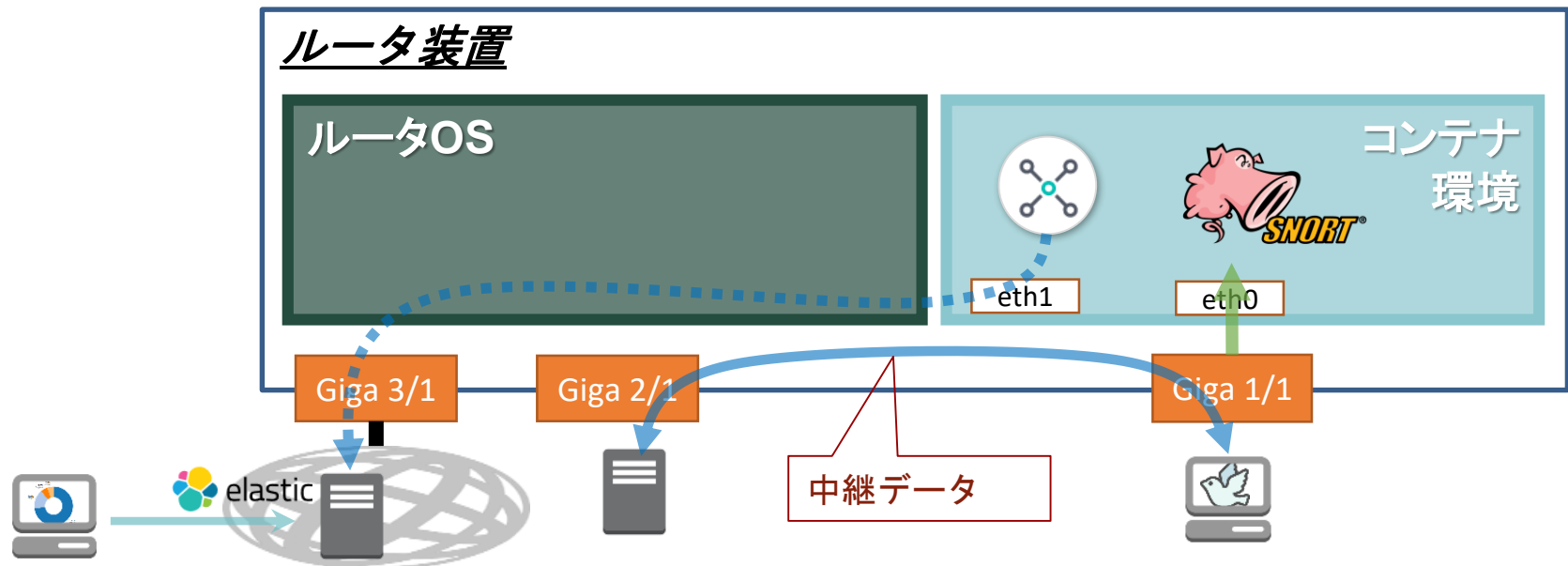
# ネットワーク型IDS

## ■ Snortによる中継データの解析、不正検知のアラート

```
root@container:~# snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/snort3-community.rules --plugin-path /usr/local/lib/snort_extra -i eth0 -H -y -k none
```

### snort.luaの設定

```
alert_json =  
{  
  file = true,  
  fields = 'timestamp pkt_num proto pkt_gen pkt_len dir src_addr src_port dst_addr dst_port  
service rule priority class action b64_data'  
}
```

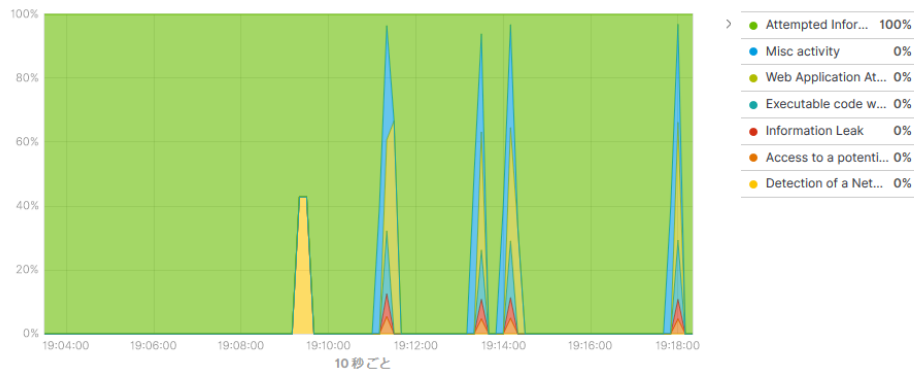


# Snortの解析結果の可視化

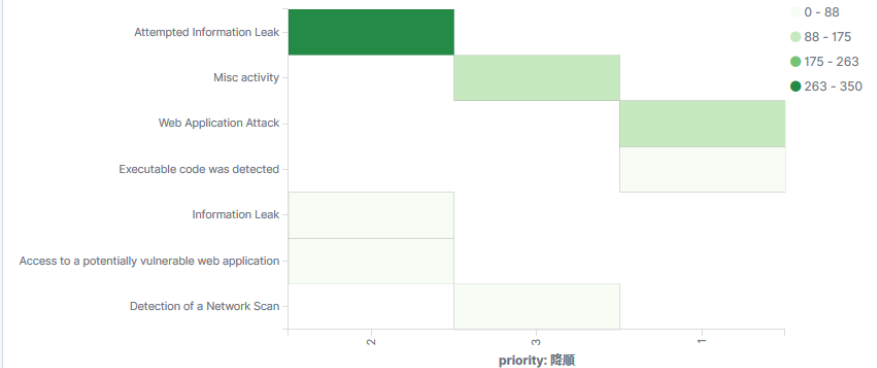
## ■ SnortのイベントログをFilebeatで外部サーバへ送信

```
root@container:~# tail /run/alert_json.txt
{ "timestamp" : "19/10/21-10:33:58.967749", "pkt_num" : 16366, "proto" : "UDP", "pkt_gen" :
"raw", "pkt_len" : 395, "dir" : "C2S", "src_addr" : "10.10.30.10", "src_port" : 56888,
"dst_addr" : "10.10.30.1", "dst_port" : 161, "service" : "unknown", "rule" : "1:1411:19",
"priority" : 2, "class" : "Attempted Information Leak", "action" : "allow", "b64_data" :
"MIIBawIBAQQGcHVibGljoIIBXAIESAj51gIBAAIBAÐCCA(中略)AAFAA==" }
```

[snort3] アクティビティ数の内訳



[snort3] event priority vs classification

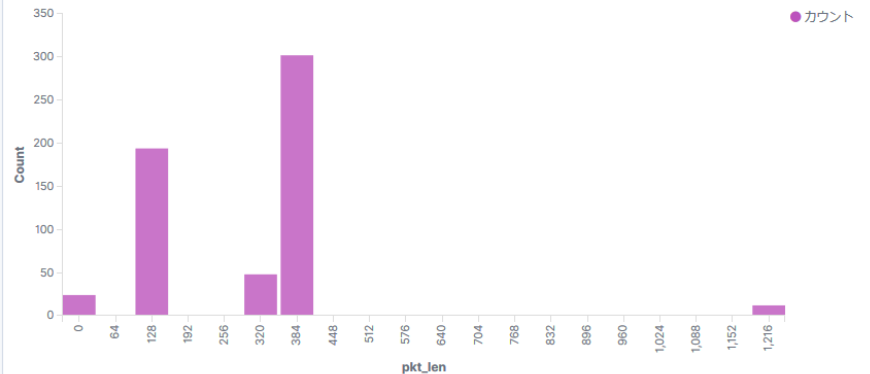


[snort3] アクティビティの概要

Executable code was detected  
Web Application Attack  
Misc activity  
Information Leak  
Access to a potentially vulnerable web application  
Detection of a Network Scan

class.keyword: 降順 - カウント

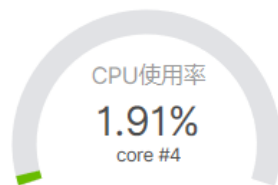
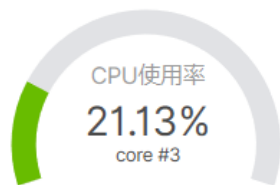
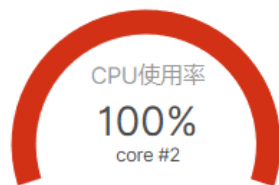
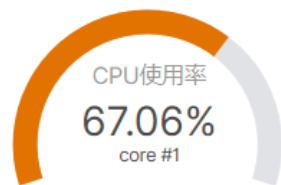
[snort3] Histogram of logged packet / buffer sizes



# CPU使用率の可視化

- ルータOS側で取得したデータをもとに可視化
  - データはfilebeatで送信

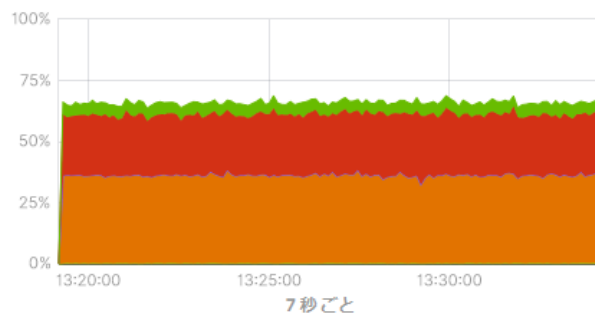
CPUコア毎のCPU使用率(mpstat)



データプレーンのCPU使用率

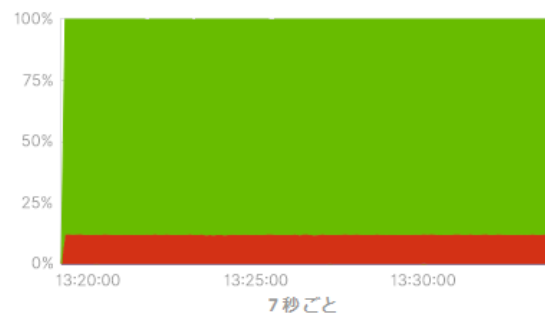


CPU使用率内訳[Core#1]



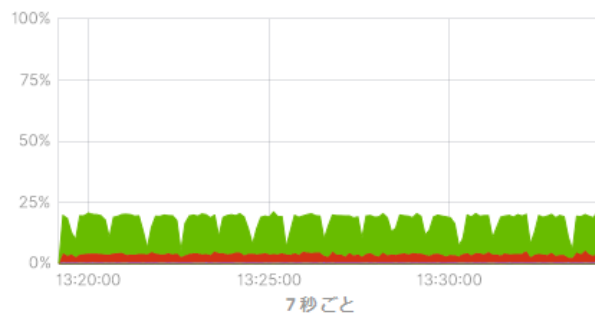
user	4.98%
sys	24.88%
nice	0%
irq	36.97%
softirq	0.24%
iowait	0%
steal	0%
guest	0%

CPU使用率内訳[Core#2]



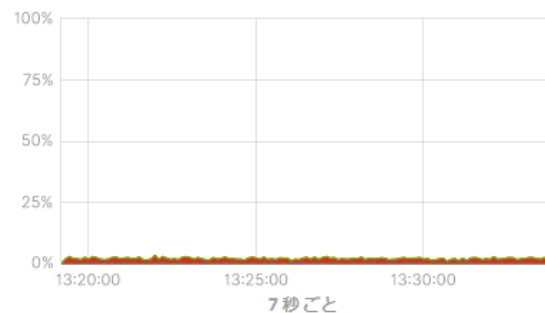
user	88.2%
sys	11.8%
nice	0%
irq	0%
softirq	0%
iowait	0%
steal	0%
guest	0%

CPU使用率内訳[Core#3]



user	16.7%
sys	3.62%
nice	0%
irq	0.4%
softirq	0.4%
iowait	0%
steal	0%
guest	0%

CPU使用率内訳[Core#4]

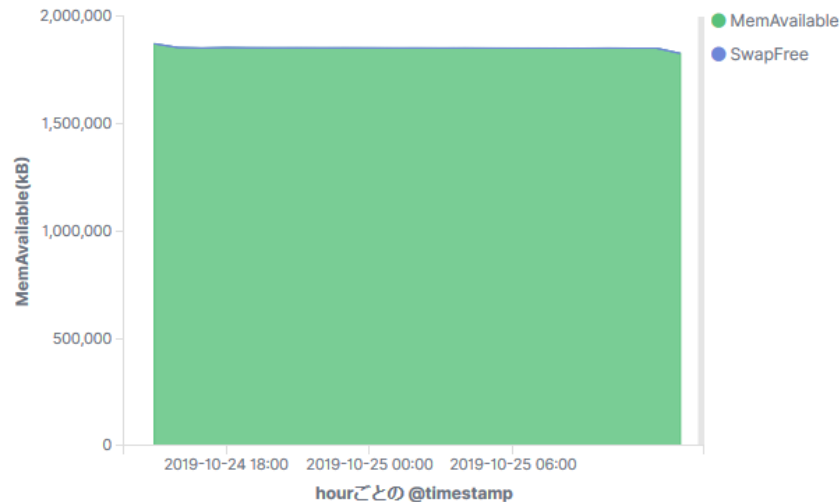


user	0%
sys	1.7%
nice	0%
irq	0.21%
softirq	0%
iowait	0%
steal	0%
guest	0%

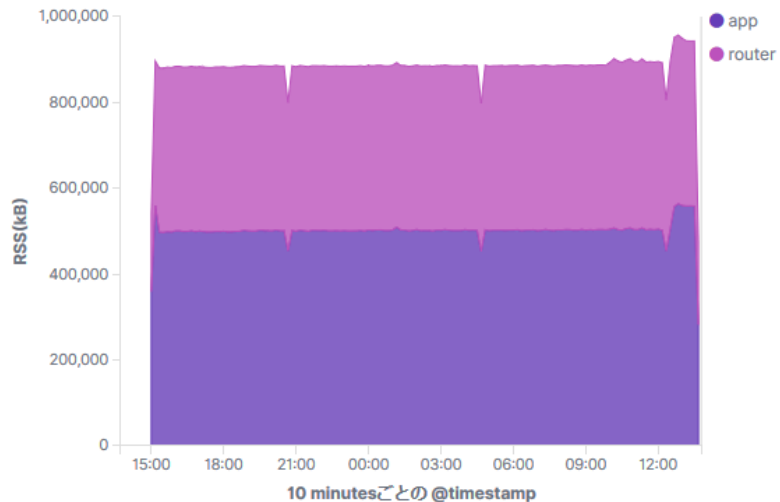
# メモリ使用量やディスクI/Oの可視化

## ■ 同様にメモリ使用量やディスクI/Oの可視化も可能

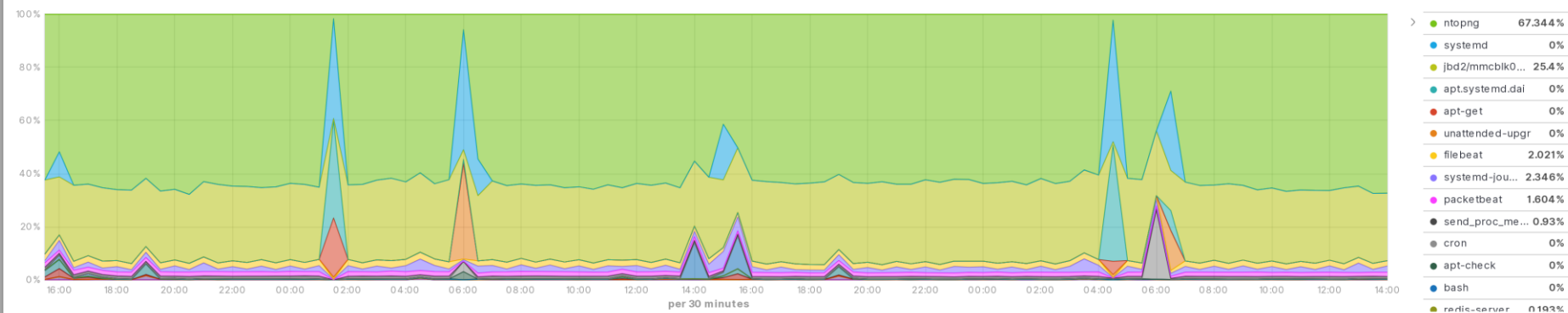
[meminfo] メモリ使用可能量



[ps] 種別毎のメモリ使用量(RSS)



[pidstat] Number of kilobytes the task has caused to be read from disk per second



# こんなこともできるかも（例）

- Webカメラ
  - USBタイプのカメラを装着してルータから画像を配信
- ソフトウェアAP
  - USBタイプの無線LAN親機を装着してルータを無線LANアクセスポイントにする
- ローカル5G
  - OSSのEPCを動作させてプライベートLTEを構築
- ChatOps
  - チャットサービスを利用したルータ装置の情報共有
- 構成管理自動化
  - Cloud-initやAnsibleなどと連携してルータやサーバ環境の構築を自動化
- 独自開発アプリのポーティング
  - Ubuntuベースのシステムで動作可能なら動くはず

**皆様のご意見をお聞かせください！**

