bwNetFlow:

A Customizable Multi-Tenant Flow Processing Platform for Transit Providers

Daniel Nägele[a], Christopher B. Hauser[b], Leonard Bradatsch[b], Stefan Wesner[b]

SC19: INDIS Workshop, 11/17/2019

[a] BelWü Landeshochschulnetz Baden-Württemberg

[b] ulm university universität ulm

Introduction

- Daniel Nägele (`naegele@belwue.de`)
- Researcher *bwNetFlow* project

- Working at AS553 (BelWü)
  - Regional research and education network
  - Serving 9 universities, 46 colleges, among others
  - Aggregate transit capacity of ~1$\frac{\text{Tbit}}{\text{s}}$
  - A lot of peering

Assumptions and Goals

- Monitor traffic on all border interfaces
- Researchers have challenging flexibility requirements
  - Treat flows as discrete messages for maximum flexibility
  - Provide interested parties with **solely** their specific flows to…

enable operative and scientific insights

enhance applications with live data

visualize using simple dashboards

access the full data using an API

Apache Kafka as a core element for bwNetFlow

- **Apache Kafka**[1] is a distributed streaming platform
- *Topics* are ordered streams of **protobuf**-encoded[2] flow objects
- Topics are *consumed* and *produced*
- Built-in support for …
  - encryption
  - load balancing
  - retention policies
  - access control
  - partitioning
  - replication
- **goflow**[3] is a Netflow Collector for Apache Kafka

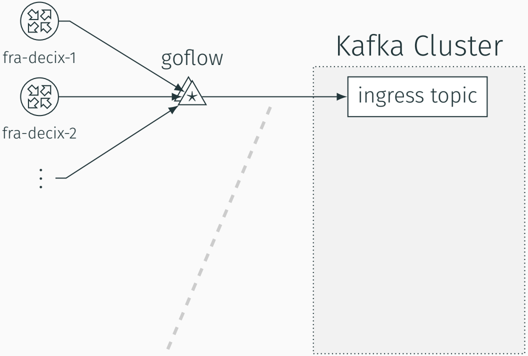
[1] `kafka.apache.org`
[2] `developers.google.com/protocol-buffers`
[3] `github.com/cloudflare/goflow`

Apache Kafka as a core element for bwNetFlow

- **Apache Kafka**[1] is a distributed streaming platform
- *Topics* are ordered streams of **protobuf**-encoded[2] flow objects
- Topics are *consumed* and *produced*
- Built-in support for …
  - encryption
  - load balancing
  - retention policies
  - access control
  - partitioning
  - replication
- **goflow**[3] is a Netflow Collector for Apache Kafka

[1] kafka.apache.org
[2] developers.google.com/protocol-buffers
[3] github.com/cloudflare/goflow

Project Architecture with Kafka at its core



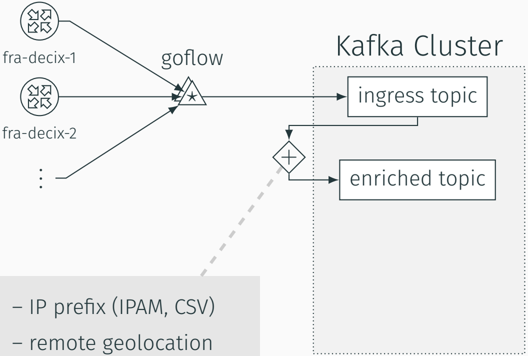
- protobuf format
- extensible and efficient

Routers

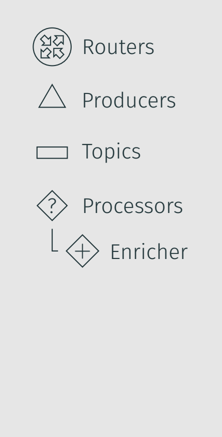
△ Producers

▭ Topics

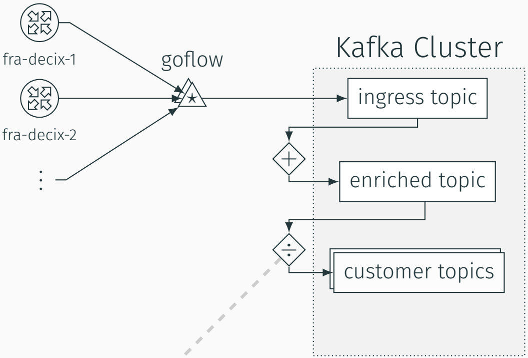
Project Architecture with Kafka at its core

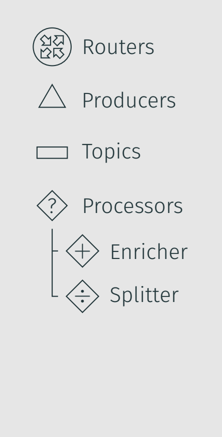


- IP prefix (IPAM, CSV)
- remote geolocation
- interface data (SNMP)
- routing info
- arbitrary lookups



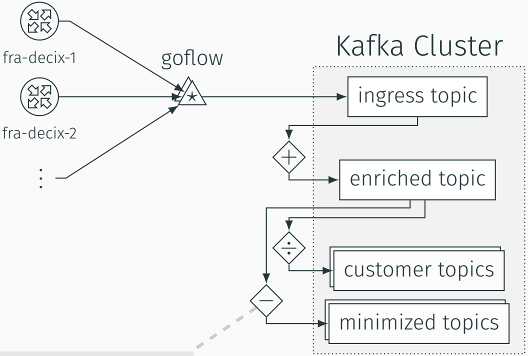
Project Architecture with Kafka at its core





- split on any field
- supports filters

Project Architecture with Kafka at its core



- remove fields
- drop flows
- anonymization

Routers

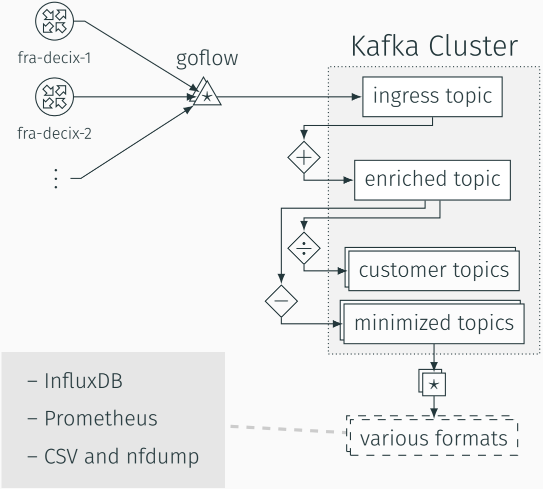
△ Producers

▭ Topics

◇ Processors

- ⊕ Enricher
- ÷ Splitter
- ⊖ Reducer

Project Architecture with Kafka at its core



fra-decix-1

fra-decix-2

goflow

Kafka Cluster

ingress topic

enriched topic

customer topics

minimized topics

various formats

- InfluxDB
- Prometheus
- CSV and nfdump

⊗ Routers

△ Producers

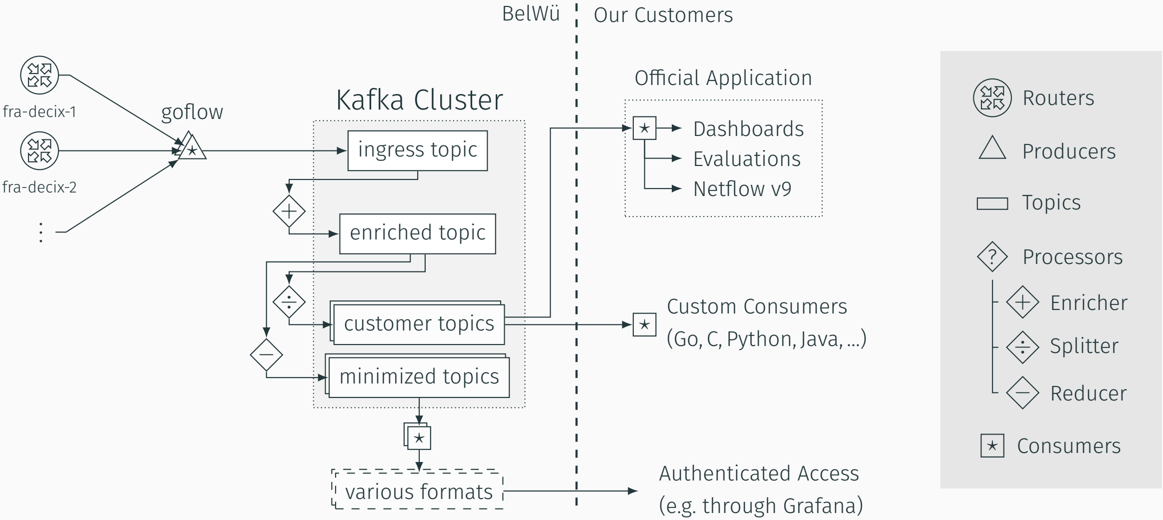
▭ Topics

◇ Processors

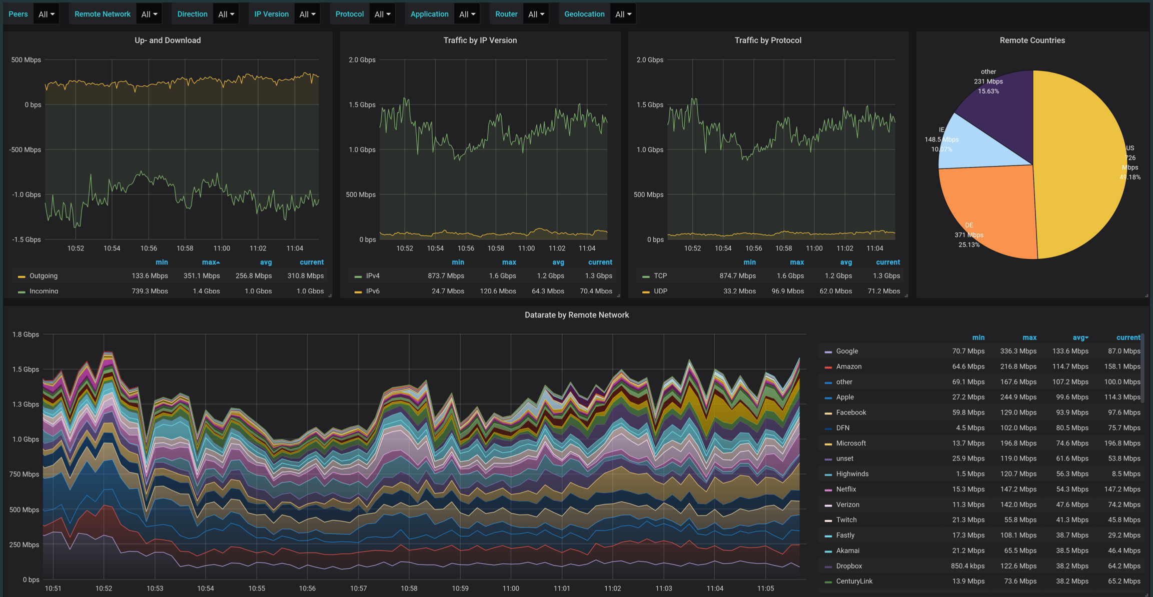
- ⊕ Enricher
- ÷ Splitter
- ⊖ Reducer

☆ Consumers

Project Architecture with Kafka at its core



Interactive Visualizations



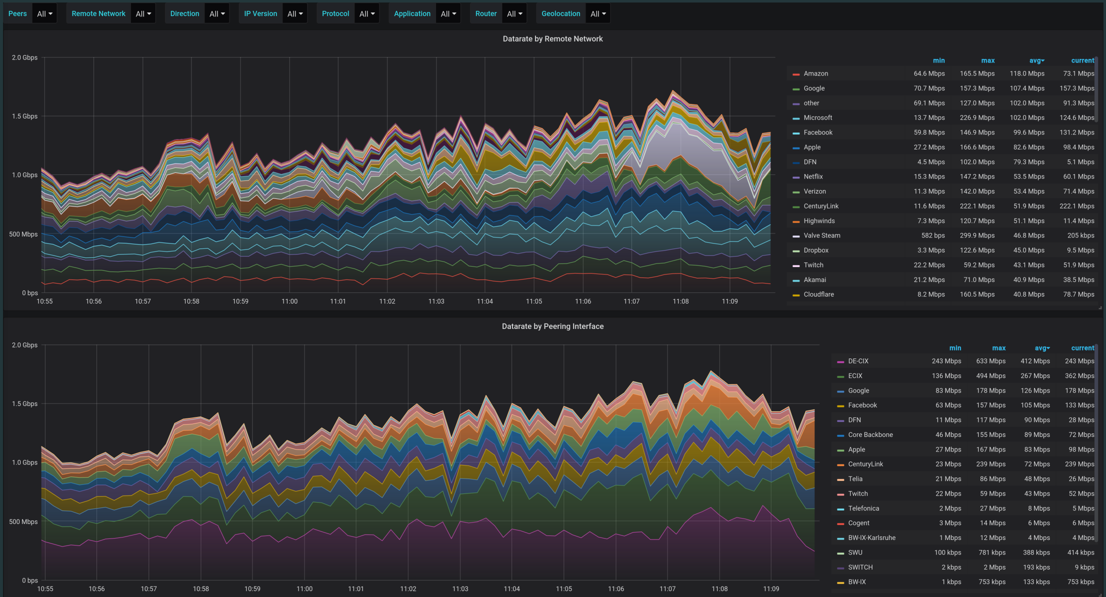
	min	max	avg	current
Outgoing	133.6 Mbps	351.1 Mbps	256.8 Mbps	310.8 Mbps
Incoming	739.3 Mbps	1.4 Gbps	1.0 Gbps	1.0 Gbps

	min	max	avg	current
IPv4	873.7 Mbps	1.6 Gbps	1.2 Gbps	1.3 Gbps
IPv6	24.7 Mbps	120.6 Mbps	64.3 Mbps	70.4 Mbps

	min	max	avg	current
TCP	874.7 Mbps	1.6 Gbps	1.2 Gbps	1.3 Gbps
UDP	33.2 Mbps	96.9 Mbps	62.0 Mbps	71.2 Mbps

	min	max	avg	current
Google	70.7 Mbps	336.3 Mbps	133.6 Mbps	87.0 Mbps
Amazon	64.6 Mbps	216.8 Mbps	114.7 Mbps	158.1 Mbps
other	69.1 Mbps	167.6 Mbps	107.2 Mbps	100.0 Mbps
Apple	27.2 Mbps	244.9 Mbps	99.6 Mbps	114.3 Mbps
Facebook	59.8 Mbps	129.0 Mbps	93.9 Mbps	97.6 Mbps
DFN	4.5 Mbps	102.0 Mbps	80.5 Mbps	75.7 Mbps
Microsoft	13.7 Mbps	196.8 Mbps	74.6 Mbps	196.8 Mbps
unset	25.9 Mbps	119.0 Mbps	61.6 Mbps	53.8 Mbps
Highwinds	1.5 Mbps	120.7 Mbps	56.3 Mbps	8.5 Mbps
Netflix	15.3 Mbps	147.2 Mbps	54.3 Mbps	147.2 Mbps
Verizon	11.3 Mbps	142.0 Mbps	47.6 Mbps	74.2 Mbps
Twitch	21.3 Mbps	55.8 Mbps	41.3 Mbps	45.8 Mbps
Fastly	17.3 Mbps	108.1 Mbps	38.7 Mbps	29.2 Mbps
Akamai	21.2 Mbps	65.5 Mbps	38.5 Mbps	46.4 Mbps
Dropbox	850.4 kbps	122.6 Mbps	38.2 Mbps	64.2 Mbps
CenturyLink	13.9 Mbps	73.6 Mbps	38.2 Mbps	65.2 Mbps

Interactive Visualizations: Transit Analysis



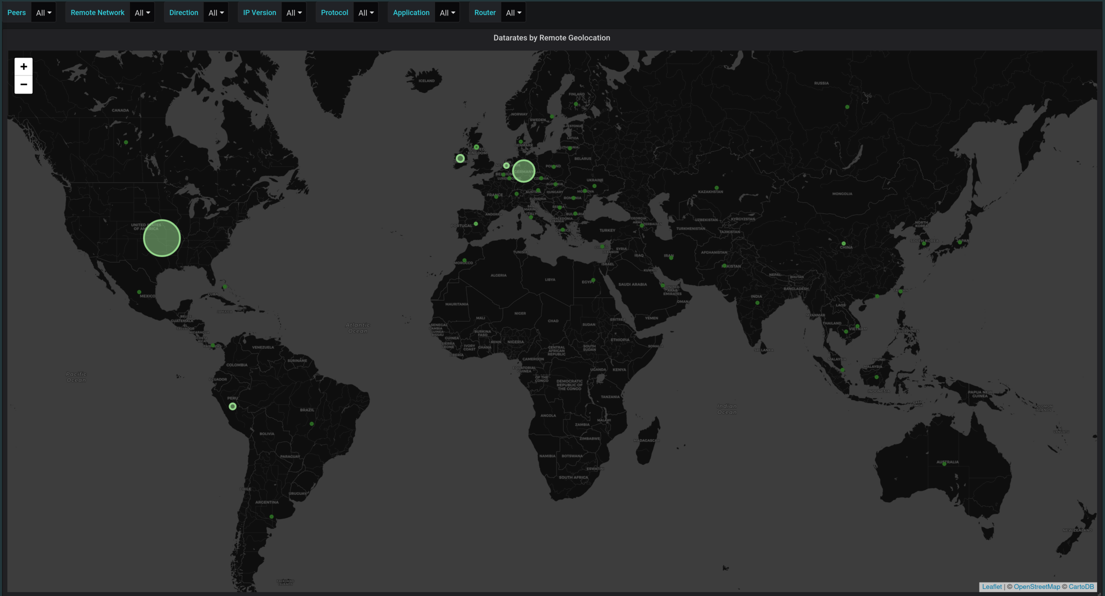
Datarate by Remote Network

	min	max	avg	current
Amazon	64.6 Mbps	165.5 Mbps	118.0 Mbps	73.1 Mbps
Google	70.7 Mbps	157.3 Mbps	107.4 Mbps	157.3 Mbps
other	69.1 Mbps	127.0 Mbps	102.0 Mbps	91.3 Mbps
Microsoft	13.7 Mbps	226.9 Mbps	102.0 Mbps	124.6 Mbps
Facebook	59.8 Mbps	146.9 Mbps	99.6 Mbps	131.2 Mbps
Apple	27.2 Mbps	166.6 Mbps	82.6 Mbps	98.4 Mbps
DFN	4.5 Mbps	102.0 Mbps	79.3 Mbps	5.1 Mbps
Netflix	15.3 Mbps	147.2 Mbps	53.5 Mbps	60.1 Mbps
Verizon	11.3 Mbps	142.0 Mbps	53.4 Mbps	71.4 Mbps
CenturyLink	11.6 Mbps	222.1 Mbps	51.9 Mbps	222.1 Mbps
Highwinds	7.3 Mbps	120.7 Mbps	51.1 Mbps	11.4 Mbps
Valve Steam	582 bps	299.9 Mbps	46.8 Mbps	205 kbps
Dropbox	3.3 Mbps	122.6 Mbps	45.0 Mbps	9.5 Mbps
Twitch	22.2 Mbps	59.2 Mbps	43.1 Mbps	51.9 Mbps
Akamai	21.2 Mbps	71.0 Mbps	40.9 Mbps	38.5 Mbps
Cloudflare	8.2 Mbps	160.5 Mbps	40.8 Mbps	78.7 Mbps

Datarate by Peering Interface

	min	max	avg	current
DE-CIX	243 Mbps	633 Mbps	412 Mbps	243 Mbps
ECIX	136 Mbps	494 Mbps	267 Mbps	362 Mbps
Google	83 Mbps	178 Mbps	126 Mbps	178 Mbps
Facebook	63 Mbps	157 Mbps	105 Mbps	133 Mbps
DFN	11 Mbps	117 Mbps	90 Mbps	28 Mbps
Core Backbone	46 Mbps	155 Mbps	89 Mbps	72 Mbps
Apple	27 Mbps	167 Mbps	83 Mbps	98 Mbps
CenturyLink	23 Mbps	239 Mbps	72 Mbps	239 Mbps
Telia	21 Mbps	86 Mbps	48 Mbps	26 Mbps
Twitch	22 Mbps	59 Mbps	43 Mbps	52 Mbps
Telefonica	2 Mbps	27 Mbps	8 Mbps	5 Mbps
Cogent	3 Mbps	14 Mbps	6 Mbps	6 Mbps
BW-IX-Karlsruhe	1 Mbps	12 Mbps	4 Mbps	4 Mbps
SWU	100 kbps	781 kbps	388 kbps	414 kbps
SWITCH	2 kbps	2 Mbps	193 kbps	9 kbps
BW-IX	1 kbps	753 kbps	133 kbps	753 kbps

Interactive Visualizations: Geolocation



Peers All ▾ | Remote Network All ▾ | Direction All ▾ | IP Version All ▾ | Protocol All ▾ | Application All ▾ | Router All ▾

Datarates by Remote Geolocation

Leaflet | © OpenStreetMap © CartoDB

API Access Example

```python
from confluent_kafka import Consumer
import flow_messages_enriched_pb2 as api

consumer = Consumer(config) # static config (host, ssl, sasl authentication)
consumer.subscribe(['flow-messages-enriched'])

while True:
    # Step 1: get data from kafka cluster
    raw = consumer.poll()

    # Step 2: decode using google protobuf
    flow = api.FlowMessage().ParseFromString(raw.value())

    # Step 3: work with the flow
    pass
```

Examples from our ops team’s Git

Which peers should fix some ACLs?

```
dst = ipaddress.ip_address(flow.DstAddr)
if not dst.is_global:
    print(flow.Peer)
```

Who has hosts talking to known Command & Control servers?

```
badguy = bytes([81,169,145,160]):
if badguy in (flow.SrcAddr, flow.DstAddr):
    print(f"{flow.Cid}: {flow.SrcAddr} -> {flow.DstAddr}")
```

Where do my users access my site from?

```
# distribution is a defaultdict: {'DE': 100, 'US': 70, ...}
if flow.DstAddr == bytes([129,143,232,10]):
    distribution[flow.RemoteCountry] += flow.Bytes
```

Examples from our ops team’s Git

Which peers should fix some ACLs?

```
dst = ipaddress.ip_address(flow.DstAddr)
if not dst.is_global:
    print(flow.Peer)
```

Who has hosts talking to known Command & Control servers?

```
badguy = bytes([81,169,145,160]):
if badguy in (flow.SrcAddr, flow.DstAddr):
    print(f"{flow.Cid}: {flow.SrcAddr} -> {flow.DstAddr}")
```

Where do my users access my site from?

```
# distribution is a defaultdict: {'DE': 100, 'US': 70, ...}
if flow.DstAddr == bytes([129,143,232,10]):
    distribution[flow.RemoteCountry] += flow.Bytes
```

Examples from our ops team’s Git

Which peers should fix some ACLs?

```
dst = ipaddress.ip_address(flow.DstAddr)
if not dst.is_global:
    print(flow.Peer)
```

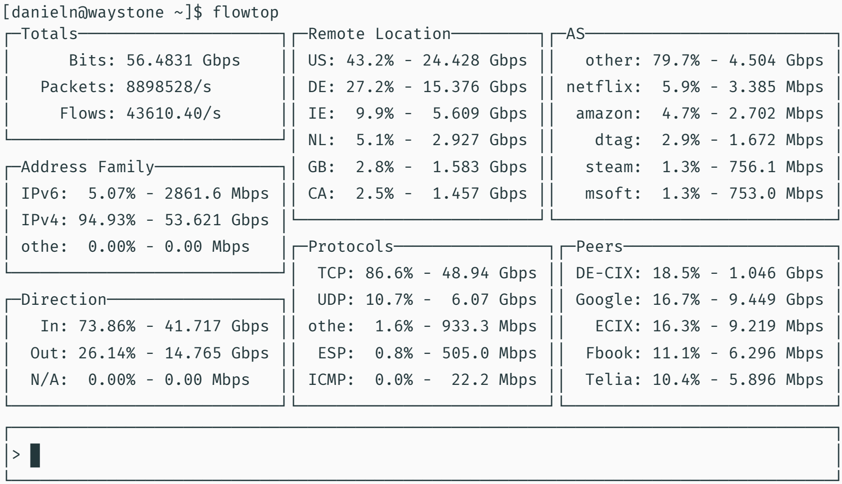
Who has hosts talking to known Command & Control servers?

```
badguy = bytes([81,169,145,160]):
if badguy in (flow.SrcAddr, flow.DstAddr):
    print(f"{flow.Cid}: {flow.SrcAddr} -> {flow.DstAddr}")
```

Where do my users access my site from?

```
# distribution is a defaultdict: {'DE': 100, 'US': 70, ...}
if flow.DstAddr == bytes([129,143,232,10]):
    distribution[flow.RemoteCountry] += flow.Bytes
```

Applications developed by our Customers: CLI Tools

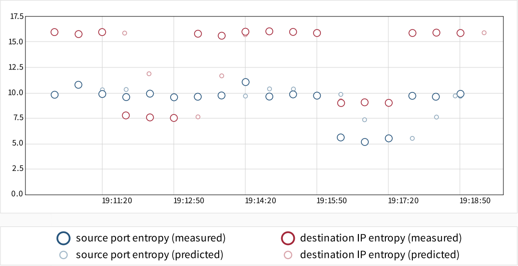


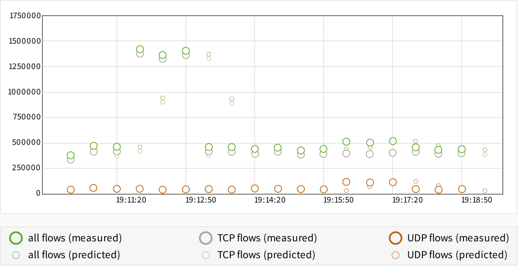
```
[danieln@waystone ~]$ flowtop
┌Totals──────────────────────┐┌Remote Location──────────┐┌AS───────────────────────────┐
│      Bits: 56.4831 Gbps    ││ US: 43.2% - 24.428 Gbps ││   other: 79.7% - 4.504 Gbps │
│   Packets: 8898528/s       ││ DE: 27.2% - 15.376 Gbps ││ netflix:  5.9% - 3.385 Mbps │
│     Flows: 43610.40/s      ││ IE:  9.9% -  5.609 Gbps ││  amazon:  4.7% - 2.702 Mbps │
└────────────────────────────┘│ NL:  5.1% -  2.927 Gbps ││    dtag:  2.9% - 1.672 Mbps │
┌Address Family─────────────┐│ GB:  2.8% -  1.583 Gbps ││   steam:  1.3% - 756.1 Mbps │
│ IPv6:  5.07% - 2861.6 Mbps ││ CA:  2.5% -  1.457 Gbps ││   msoft:  1.3% - 753.0 Mbps │
│ IPv4: 94.93% - 53.621 Gbps │└─────────────────────────┘└─────────────────────────────┘
│ othe:  0.00% - 0.00 Mbps   │┌Protocols─────────────────┐┌Peers───────────────────────┐
└────────────────────────────┘│  TCP: 86.6% - 48.94 Gbps ││ DE-CIX: 18.5% - 1.046 Gbps │
┌Direction───────────────────┐│  UDP: 10.7% -  6.07 Gbps ││ Google: 16.7% - 9.449 Gbps │
│   In: 73.86% - 41.717 Gbps ││ othe:  1.6% - 933.3 Mbps ││   ECIX: 16.3% - 9.219 Mbps │
│  Out: 26.14% - 14.765 Gbps ││  ESP:  0.8% - 505.0 Mbps ││  Fbook: 11.1% - 6.296 Mbps │
│  N/A:  0.00% - 0.00 Mbps   ││ ICMP:  0.0% -  22.2 Mbps ││  Telia: 10.4% - 5.896 Mbps │
└────────────────────────────┘└──────────────────────────┘└────────────────────────────┘
┌──────────────────────────────────────────────────────────────────────────────────────┐
│> █                                                                                   │
└──────────────────────────────────────────────────────────────────────────────────────┘
```

Applications developed by our Customers: DDoS Detection

KIT
BelWü
Uni Tübingen
Uni Ulm

Currently no attack in this network.





Future Work

- Improve Open Source presence and documentation
- Allow customers to influence their pipelines directly, without manual intervention
- Follow-up project bwNet2020 is approved, integrating both projects
    - major themes: Network Function Virtualization, Service Function Chaining
    - bwNetFlow as central component for the monitoring aspect as well as a service

Thank you!

Questions?

or contact me later: `naegele@belwue.de`

our code: `github.com/bwNetFlow`