pmacct and network analytics at AS2914

Paolo Lucente

NTT Communications | pmacct

PTT Forum 11, Sao Paulo – Dec 2017

whoami

Paolo Lucente
GitHub: [paololucente]
LinkedIn: [plucente]

Digging data out of networks worldwide for fun and profit for more than 10 years

Round of introductions

Who is NTT Communications (NTT Com)?

NTT Communications

Global IP Network

AS 2914

global carrier Awards 2017

Winner

Best Global Wholesale Carrier - Data

NTT Communications

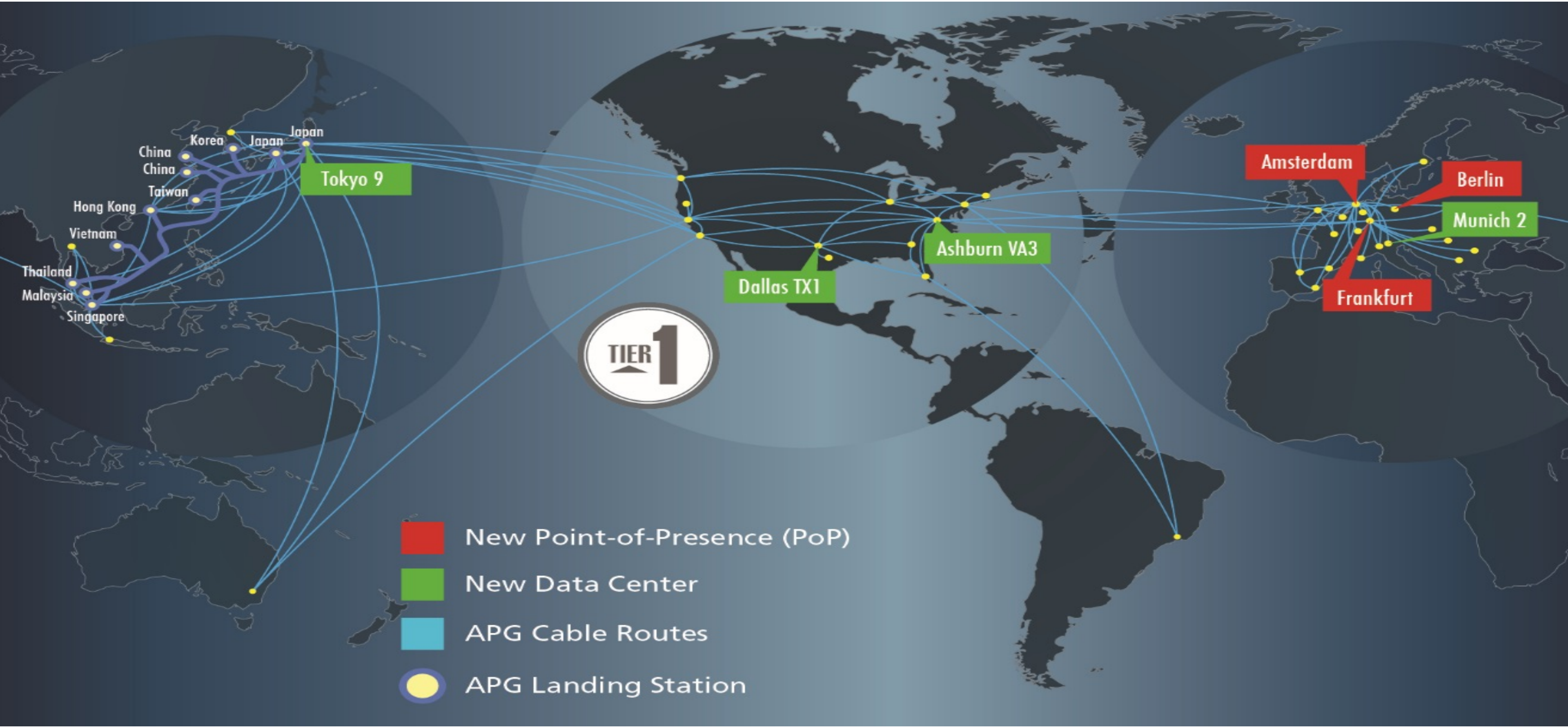
wca

2017 Winner

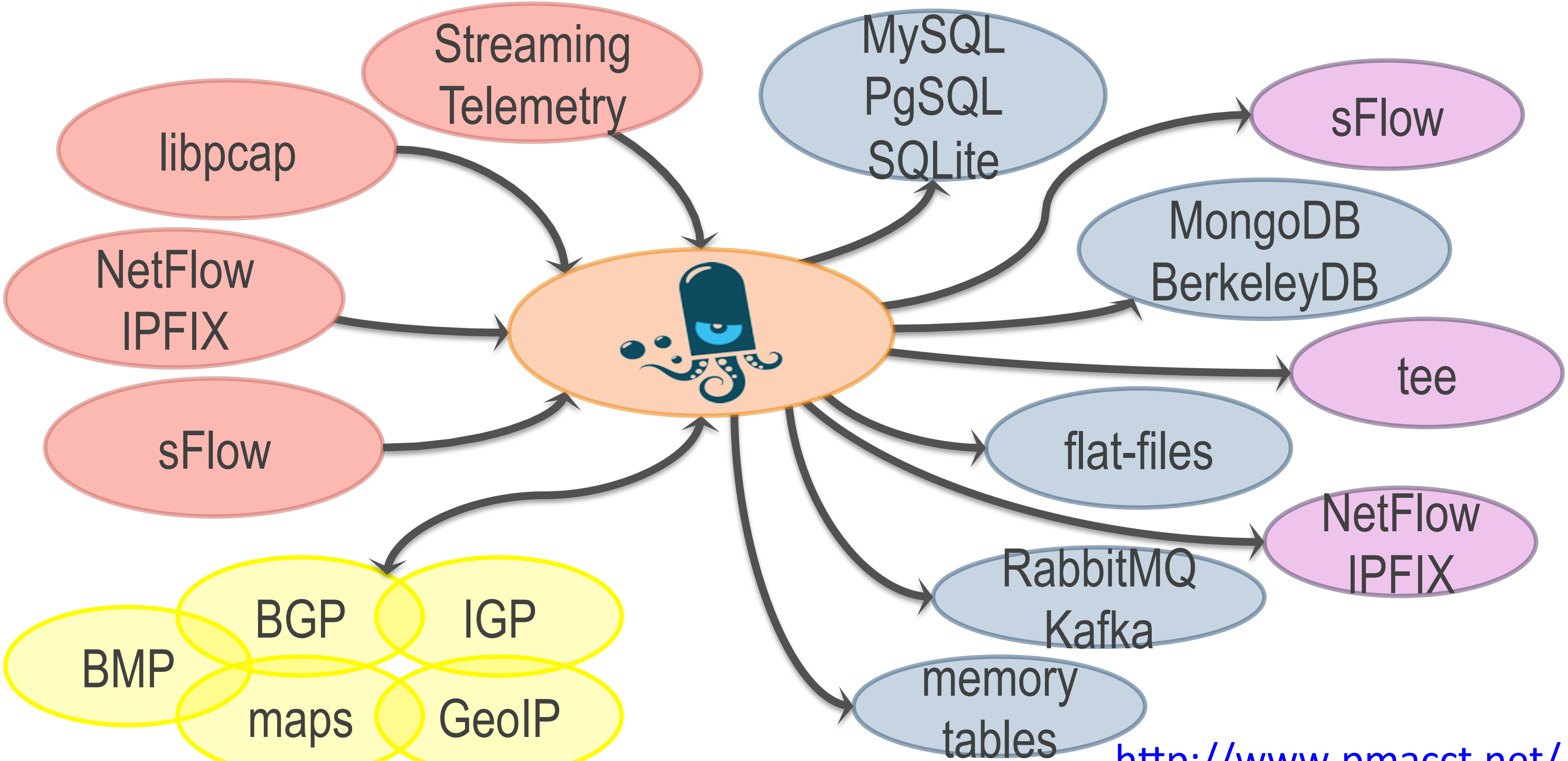
Best Wholesale Operator

- A subsidiary of NTT, one of the largest telecom companies in the world
- NTT Com has 18,000 employees worldwide
- One of the top IP providers in the world
- Infrastructure includes Global IP Network and 140+ data centers in 196 countries/regions

NTT Com's Global IP Network AS2914

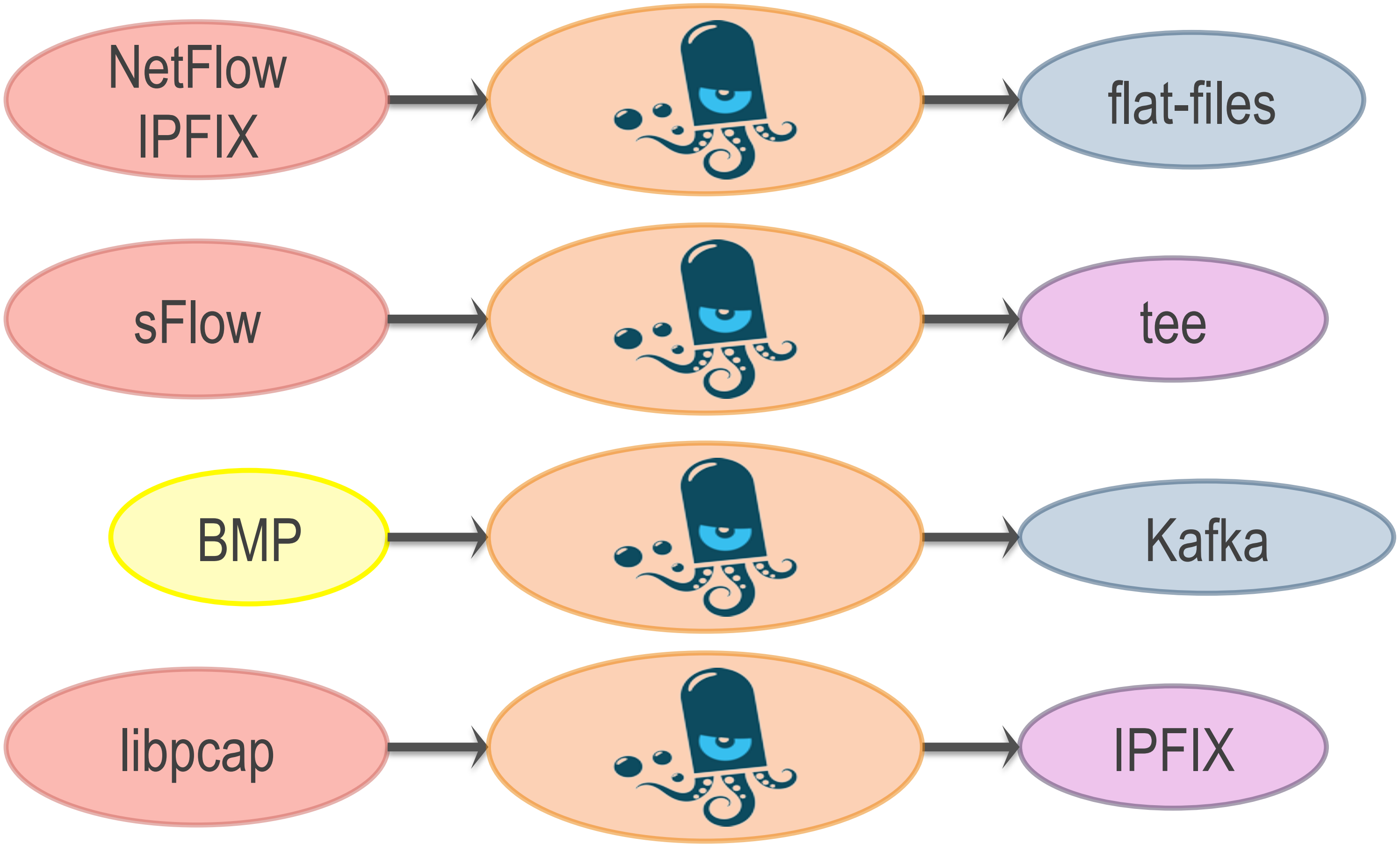


pmacct is open-source, free, GPL'ed software

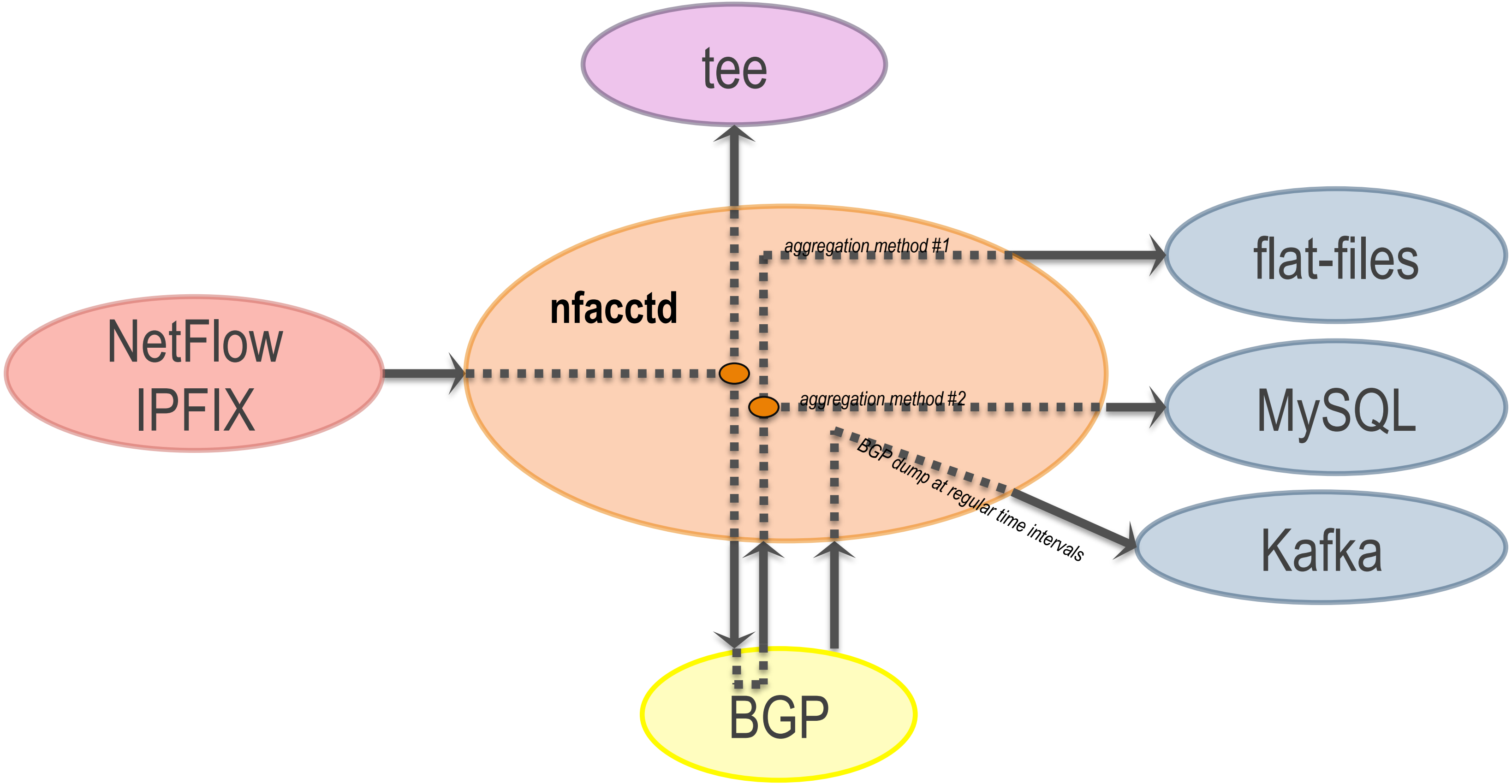


http://www.pmacct.net/

pmacct: a few simple use-cases

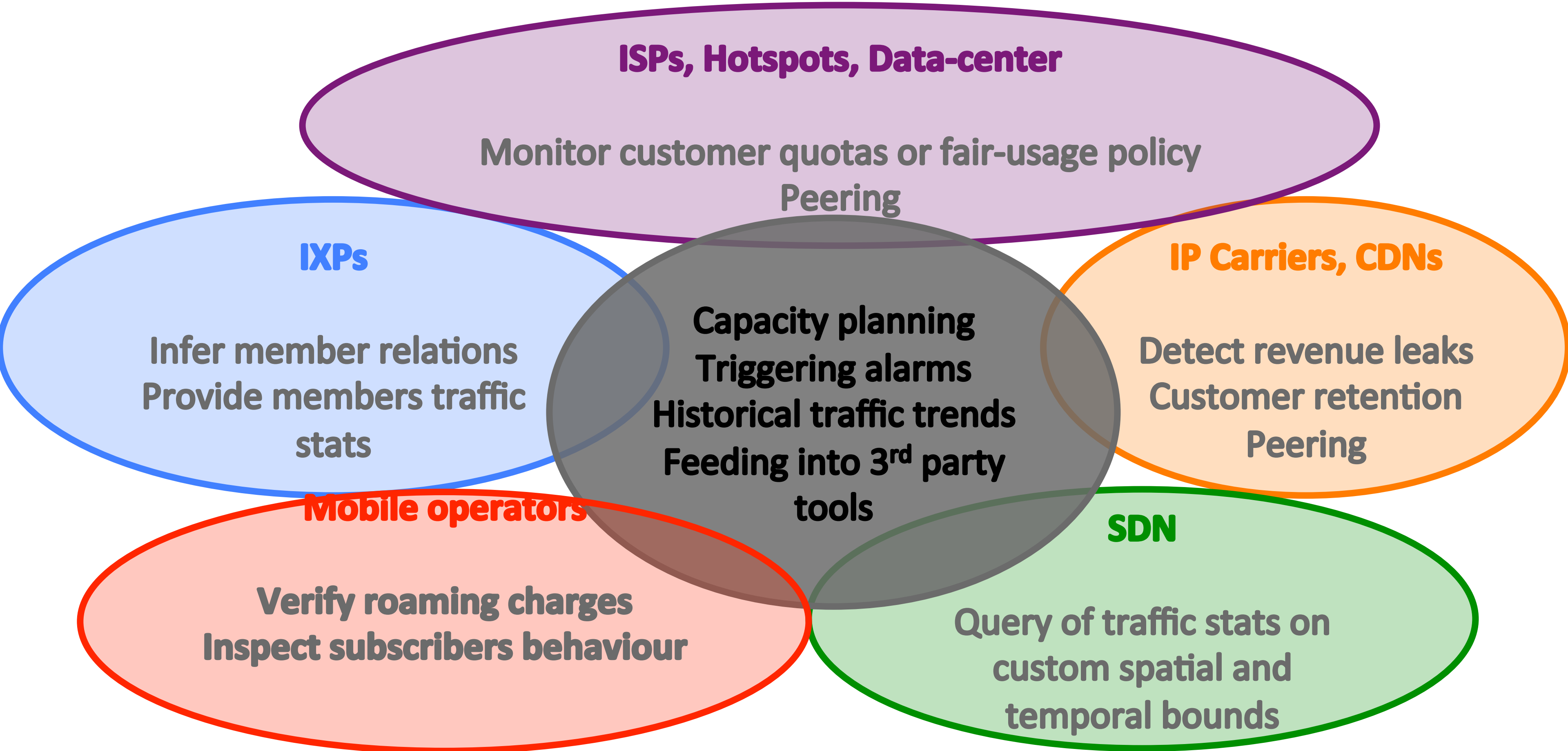


pmacct: a slightly more complex use-case



The use-case for message brokers

Use cases by industry



Key pmacct non-technical facts

- 10+ years old project
- Can’t spell the name after the second drink
- Free, open-source, independent
- Under active development
- Innovation being introduced
- Well deployed around, also in large SPs/IXPs
- Close to the SP/IXP community needs

Some technical facts (1/2)

- Pluggable architecture:
  - Can easily add support for new data sources and backends
- Correlation of data sources:
  - Natively supported data sources (ie. flow telemetry, BGP, BMP, IGP, Streaming Telemetry)
  - Enrich with external data sources via tags and labels
- Enable analytics against each data source:
  - Stream real-time
  - Dump at regular time intervals (possible state compression)

Some technical facts (2/2)

- Build multiple views out of the very same collected network traffic, ie.:
  - Unaggregated to flat-files for security and forensics; or to message brokers (RabbitMQ, Kafka) for Big Data
  - Aggregated as [<ingress router>, <ingress interface>, <BGP next-hop>, <peer destination ASN>] and sent to a SQL DB to build an internal traffic matrix for capacity planning purposes
- Pervasive data-reduction techniques, ie.:
  - Data aggregation
  - Filtering
  - Sampling, Re-Sampling

Building a Network Analytics pipeline

PTT Forum 11, Sao Paulo – Dec 2017

Typical key goals for Network Analytics

- Business Intelligence
- Insight in traffic patterns
- Support peering decisions
- Support forensics of network events
- Capacity planning
- Traffic Engineering

Sample pipeline for Network Analytics

- Input data (BGP, NetFlow, SNMP, …)
- Collection (pmacct, homegrown SNMP poller)
- Data encoding (JSON, Apache Avro, etc.)
- Distribution (Kafka)
- Enrichment (homegrown glue in $language)
- Ingestion (RDBMS vs TSDB)
- Visualization

Getting BGP to the collector

- Needed for technical reasons:
  - Flow exporters use NetFlow v5, ie. no BGP next-hop
  - Flow exporters are unaware of BGP
  - Libpcap is used to collect traffic data
- Needed for topology or traffic related reasons:
  - Transiting traffic to 3rd parties
  - Dominated by outbound traffic

Getting BGP to the collector (cont.d)

- Let pmacct collector BGP peer with all PE devices: facing peers, transit and customers
  - No best-path computation at the collector: scalability preferred to optimizing memory usage
  - Count some 50MB of memory per full-routing table
- Set the collector as iBGP peer at the PE devices:
  - Configure it as a RR client
  - Collector acts as iBGP peer across (sub-)AS boundaries

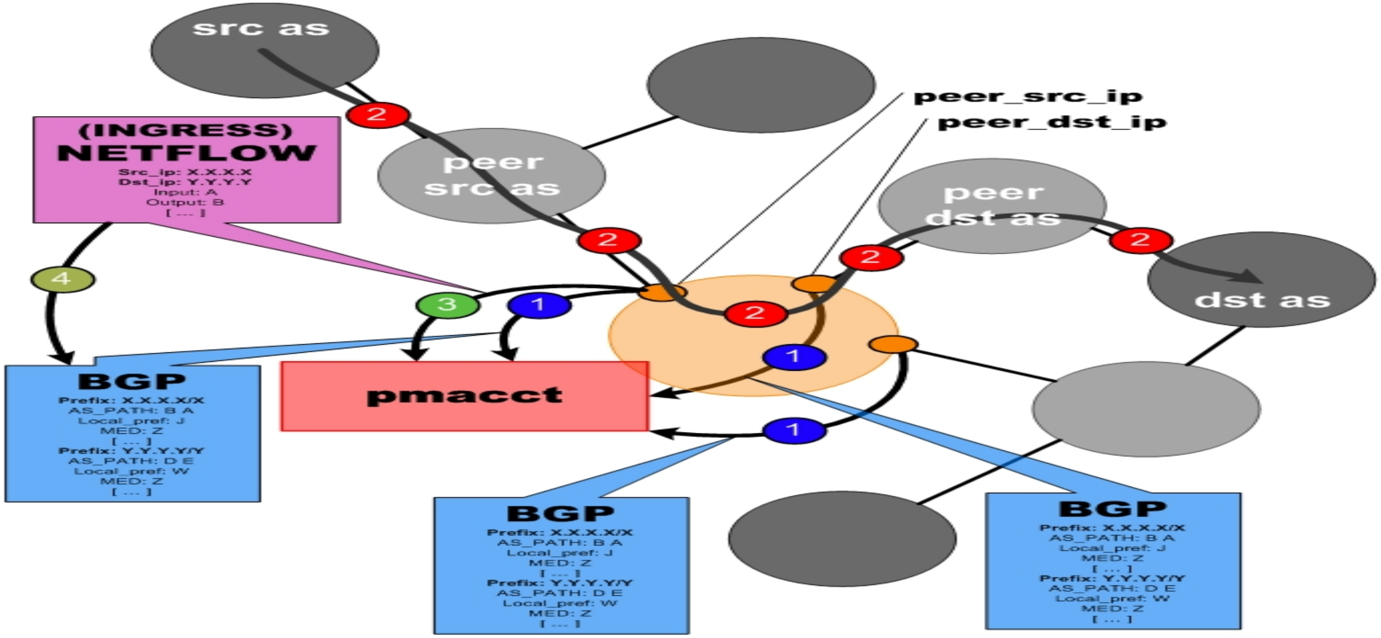
Getting telemetry to the collector

- Export ingress-only measurements at all PE devices: facing peers, transit and customers.
  - Traffic is routed to destination, so plenty of information on where it's going to
    - True, some eBGP multi-path scenarios may get challenging
  - It's crucial instead to get as much as possible about where traffic is coming from, ie.:
    - input interface at ingress router
    - source MAC address
- Perform data reduction at the PE (ie. sampling)

Getting telemetry to the collector (cont.d)

- Multiple flow collectors can be in use, ie. for different purposes. Typical export models:
  - Single tier, unicast: PE devices perform N exports
  - Multiple tiers: PEs perform export to transparent replicators in active/standby fashion; these in turn stream telemetry data to the actual collectors
- It's crucial flow collectors can tag, aggregate, filter, etc. telemetry data:
  - … might be not all data is for every collector

Telemetry data/BGP correlation



1 Edge routers send full BGP tables to pmacct
2 Traffic flows
3 NetFlow records are sent to pmacct
4 pmacct looks up BGP information: NF src addr == BGP src addr

Data Encoding

- JSON
  - Schemaless
  - Can be compressed successfully end-to-end
  - Simple, easy to troubleshoot and debug
  - Often that is the encoding supported at ingestion time
  - Similars: BSON, MsgPack
- Apache Avro
  - With schema
  - Binary format (when things go wrong ..)
  - Similars: Thrift, Capt'n Proto

Distribution

- Kafka: de-facto standard for data shipping
  - Easy to model different producer-consumer architectures
  - pmacct has a plugin to produce to Kafka
  - Most TSDBs can consume from Kafka
- People in the need for raw data can tap into this layer to consume directly
- Intuitive (that does not mean straightforward ..) to scale-out, balance and replicate

Storing data persistently

- Data need to be aggregated both in spatial and temporal dimensions before being written down:
  - Optimal usage of system resources
  - Avoids expensive consolidation of micro-flows
- Build project-driven data set(s):
  - No shame in multiple partly overlapping data-sets
  - Optimize computing

Storing data persistently (cont.d)

- "noSQL" databases (Big Data ☺):
  - Able to handle large time-series data-sets
  - Meaningful subset of SQL query language
  - Innovative storage and indexing engines
  - Scalable: clustering, spatial and temporal partitioning
  - UI-ready: ie. ELK and TICK stacks
- Open-source RDBMS:
  - Able to handle large data-sets
  - Flexible and standardized SQL query language
  - Solid storage and indexing engines
  - Scalable: clustering, spatial and temporal partitioning

Enriching data

Tag

BGP Fields

Counters

Time

```
CREATE TABLE acct_bgp (
    tag INT(4) UNSIGNED NOT NULL,
    as_src INT(4) UNSIGNED NOT NULL,
    as_dst INT(4) UNSIGNED NOT NULL,
    peer_as_src INT(4) UNSIGNED NOT NULL,
    peer_as_dst INT(4) UNSIGNED NOT NULL,
    peer_ip_src CHAR(15) NOT NULL,
    peer_ip_dst CHAR(15) NOT NULL,
    comms CHAR(24) NOT NULL,
    as_path CHAR(21) NOT NULL,
    local_pref INT(4) UNSIGNED NOT NULL,
    med INT(4) UNSIGNED NOT NULL,
    packets INT UNSIGNED NOT NULL,
    bytes BIGINT UNSIGNED NOT NULL,
    stamp_inserted DATETIME NOT NULL,
    stamp_updated DATETIME,
    PRIMARY KEY (…)
);
```

```
shell> cat pretag.map
id=100  peer_src_as=<customer>
id=80   peer_src_as=<peer>
id=50   peer_src_as=<IP transit>
[ … ]
```

```
shell> cat peers.map
id=65534 ip=X in=A
id=65533 ip=Y in=B src_mac=J
id=65532 ip=Z in=C bgp_nexthop=W
[ … ]
```

Post-processing and reporting

- Traffic delivered to a BGP peer, per location:

```
mysql> SELECT  peer_as_dst, peer_ip_dst, SUM(bytes), stamp_inserted \
          FROM acct_bgp \
          WHERE peer_as_dst = <peer | customer | IP transit> AND
                stamp_inserted = < today | last hour | last 5 mins > \
          GROUP BY peer_as_dst, peer_ip_dst
```

- Aggregate AS PATHs to the second hop:

```
mysql> SELECT SUBSTRING_INDEX(as_path, '.', 2) AS as_path, bytes \
          FROM acct_bgp \
          WHERE local_pref = < IP transit pref> AND
                stamp_inserted = < today | yesterday | last week > \
          GROUP BY SUBSTRING_INDEX(as_path, '.', 2)
          ORDER BY SUM(bytes)
```

- Focus peak hour (say, 8pm) data:

```
mysql> SELECT … FROM … WHERE … \
         stamp_inserted LIKE '2010-02-% 20:00:00' \
          …
```

Post-processing and reporting (cont.d)

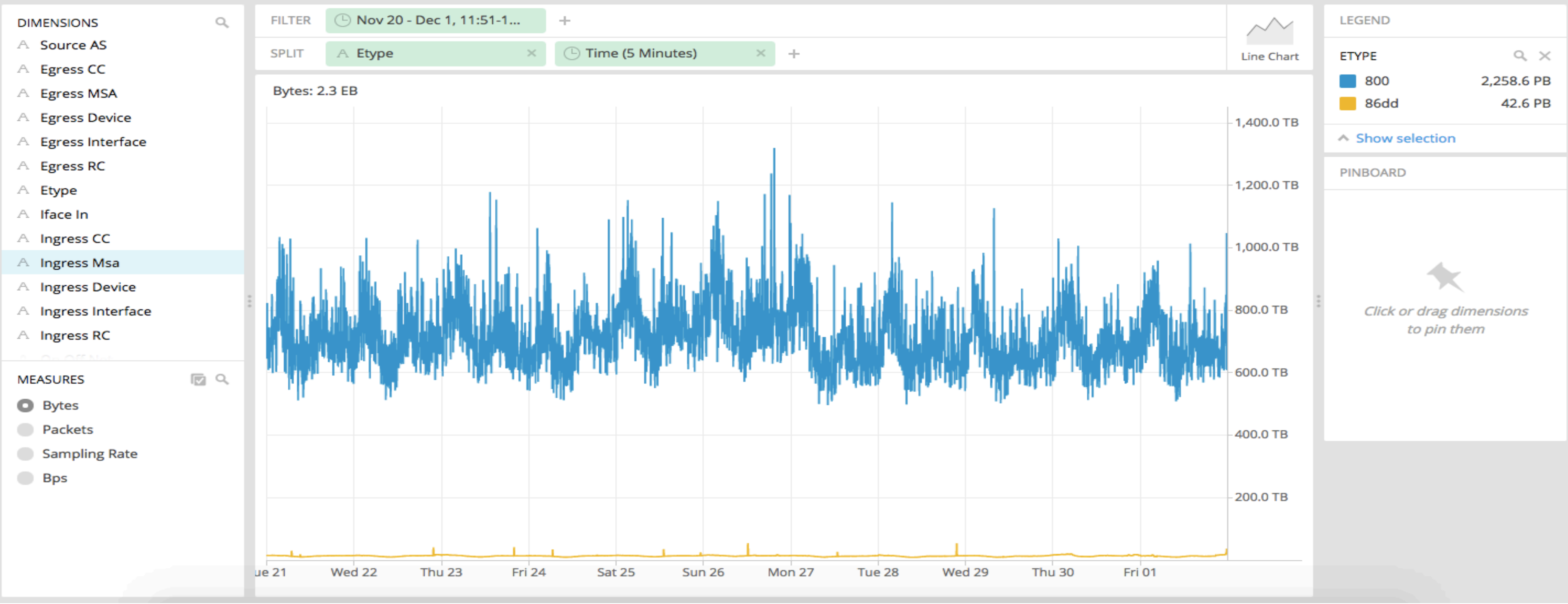
- Traffic breakdown, ie. top N grouping BGP peers of the same kind (ie. peers, customers, transit):

```
mysql> SELECT … FROM … WHERE … \
       local_pref = <<peer | customer | IP transit> pref> \
       …
```

- Fetch a traffic matrix (or a subset of it):

```
mysql> SELECT peer_ip_src, peer_ip_dst, bytes, stamp_inserted \
       FROM acct_bgp \
       WHERE [ peer_ip_src = <location A> AND \
             peer_ip_dst = <location Z> AND \ ]
             stamp_inserted = < today | last hour | last 5 mins > \
       GROUP BY peer_ip_src, peer_ip_dst
```

UI example



DIMENSIONS
Source AS
Egress CC
Egress MSA
Egress Device
Egress Interface
Egress RC
Etype
Iface In
Ingress CC
Ingress Msa
Ingress Device
Ingress Interface
Ingress RC
MEASURES
Bytes
Packets
Sampling Rate
Bps
FILTER
Nov 20 - Dec 1, 11:51-1...
SPLIT
Etype
Time (5 Minutes)
Line Chart
Bytes: 2.3 EB
1,400.0 TB
1,200.0 TB
1,000.0 TB
800.0 TB
600.0 TB
400.0 TB
200.0 TB
ue 21
Wed 22
Thu 23
Fri 24
Sat 25
Sun 26
Mon 27
Tue 28
Wed 29
Thu 30
Fri 01
LEGEND
ETYPE
800 2,258.6 PB
86dd 42.6 PB
Show selection
PINBOARD
Click or drag dimensions to pin them

pmacct and network analytics at AS2914

Paolo Lucente paolo@ntt.net
http://www.gin.ntt.net

Paolo Lucente paolo@pmacct.net
http://www.pmacct.net/ | https://github.com/pmacct/pmacct

PTT Forum 11, Sao Paulo – Dec 2017

Bonus slides

Telemetry data correction

- Telemetry data may get imprecise (ie. due to sampling)
- Use interface stats as gold standard
- Mold telemetry data .. to match interface stats:
  - Builds on Traffic Matrix estimation methods:
    - Tutorial: Best Practices for Determining the Traffic Matrix in IP Networks, NANOG 43
  - Adds telemetry data to linear system to solve
  - Solve system such that there is strict conformance with link stat values, with other measurements matched as best possible

Briefly on scalability

- A single collector might not fit it all:
  - Memory: can't store all BGP full routing tables
  - CPU: can't cope with the pace of telemetry export
- Divide-et-impera approach is valid:
  - Assign PEs (both telemetry and BGP) to collectors
  - If natively supported DB:
    - Assign collectors to DB nodes
    - Cluster the DB
  - If not-natively supported DB:
    - Assign collectors to message brokers
    - Cluster the messaging infrastructure