pmacct and network analytics at AS2914

Paolo Lucente

NTT Communications | pmacct
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)?

- 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

PTT Forum 11, Sao Paulo – Dec 2017
NTT Com’s Global IP Network AS2914
pmacct is open-source, free, GPL’ed software

libpcap
NetFlow
IPFIX
sFlow
Streaming Telemetry
MySQL
PgSQL
SQLite
MongoDB
BerkeleyDB
RabbitMQ
Kafka
memory tables
NetFlow
IPFIX
BMP
BGP
IGP
GeoIP
http://www.pmacct.net/
pmacct: a few simple use-cases

NetFlow
IPFIX
->
flat-files

sFlow
->
tee

BMP
->
Kafka

libpcap
->
IPFIX
pmacct: a slightly more complex use-case
The use-case for message brokers

PMACCT → Kafka → RabbitMQ → Elasticsearch → Cassandra → Prometheus → InfluxDB → OpenTSDB → Grafana → Kibana → Superset
Use cases by industry

ISPs, Hotspots, Data-center
- Monitor customer quotas or fair-usage policy
- Peering

IXPs
- Infer member relations
- Provide members traffic stats

Mobile operators
- Verify roaming charges
- Inspect subscribers behaviour

IP Carriers, CDNs
- Detect revenue leaks
- Customer retention
- Peering

SDN
- Query of traffic stats on custom spatial and temporal bounds

Capacity planning
- Triggering alarms
- Historical traffic trends
- Feeding into 3rd party tools
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
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, i.e.:
    - input interface at ingress router
    - source MAC address
- Perform data reduction at the PE (i.e. sampling)
Getting telemetry to the collector (cont.d)

- Multiple flow collectors can be in use, i.e. 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 pmaacct
2. Traffic flows
3. NetFlow records are sent to pmaacct
4. pmaacct 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
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 (...)
);
Post-processing and reporting

- Traffic delivered to a BGP peer, per location:
  
  ```sql
  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:
  
  ```sql
  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:
  
  ```sql
  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
pmacacct 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 (i.e. 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