pmacct: introducing BGP natively into a NetFlow/sFlow collector

Paolo Lucente

the pmacct project | AS286

<paolo at pmacct dot net>

http://www.pmacct.net/

UKNOF #14 meeting, London, 11th Sep 2009

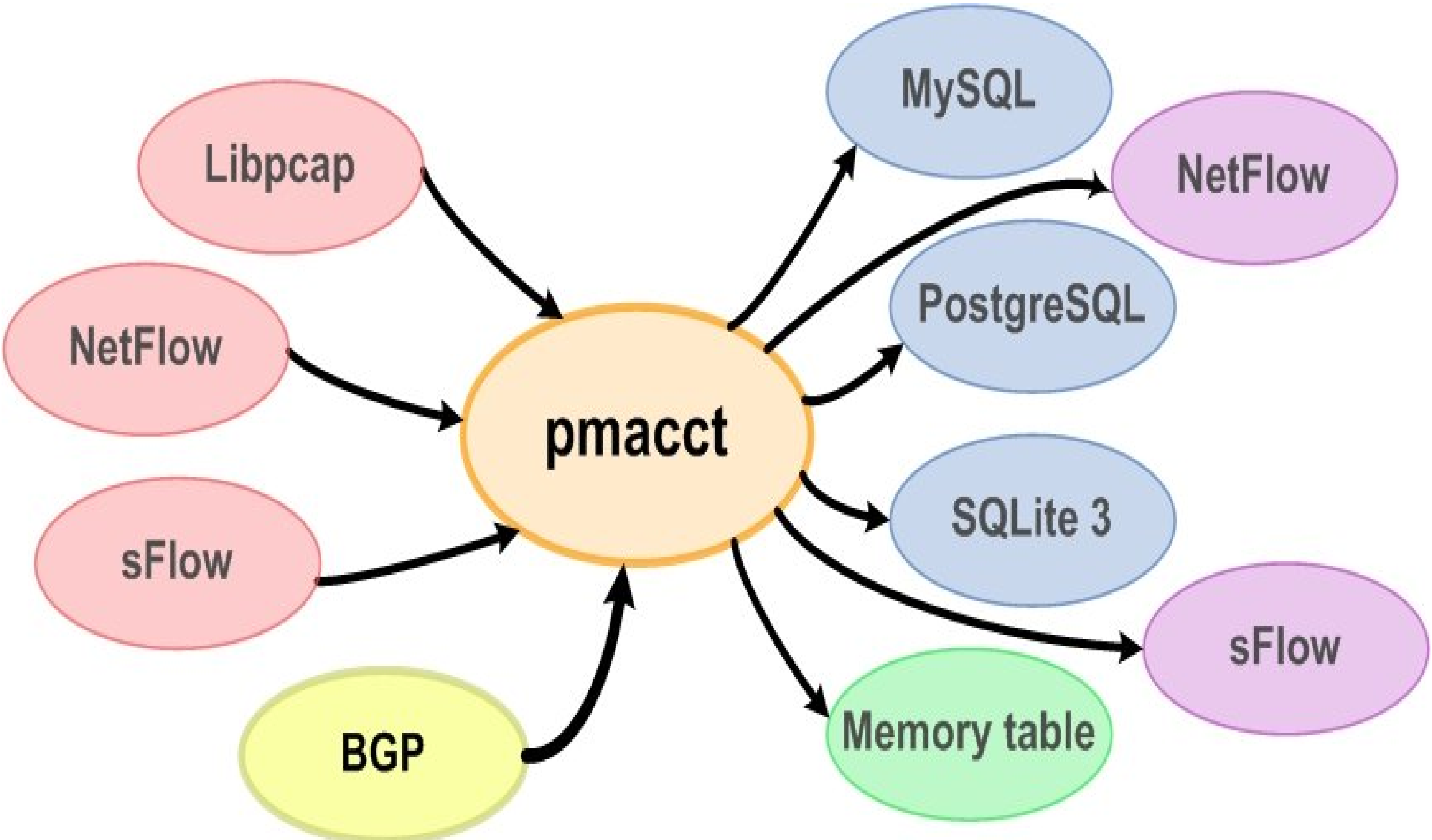
pmacct: introducing BGP natively into a NetFlow/sFlow collector

Agenda

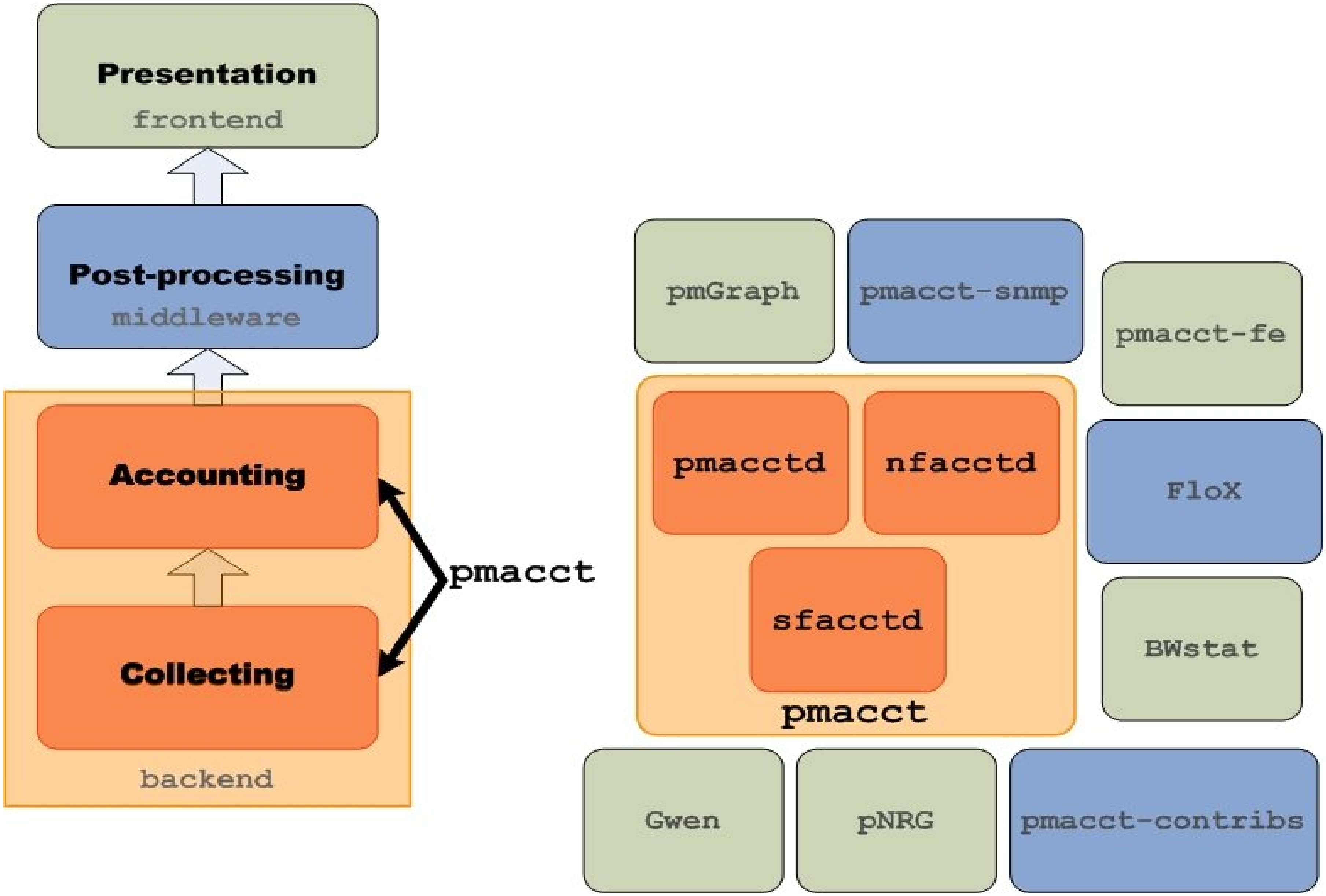
I. **Introduction**

II. Recent developments

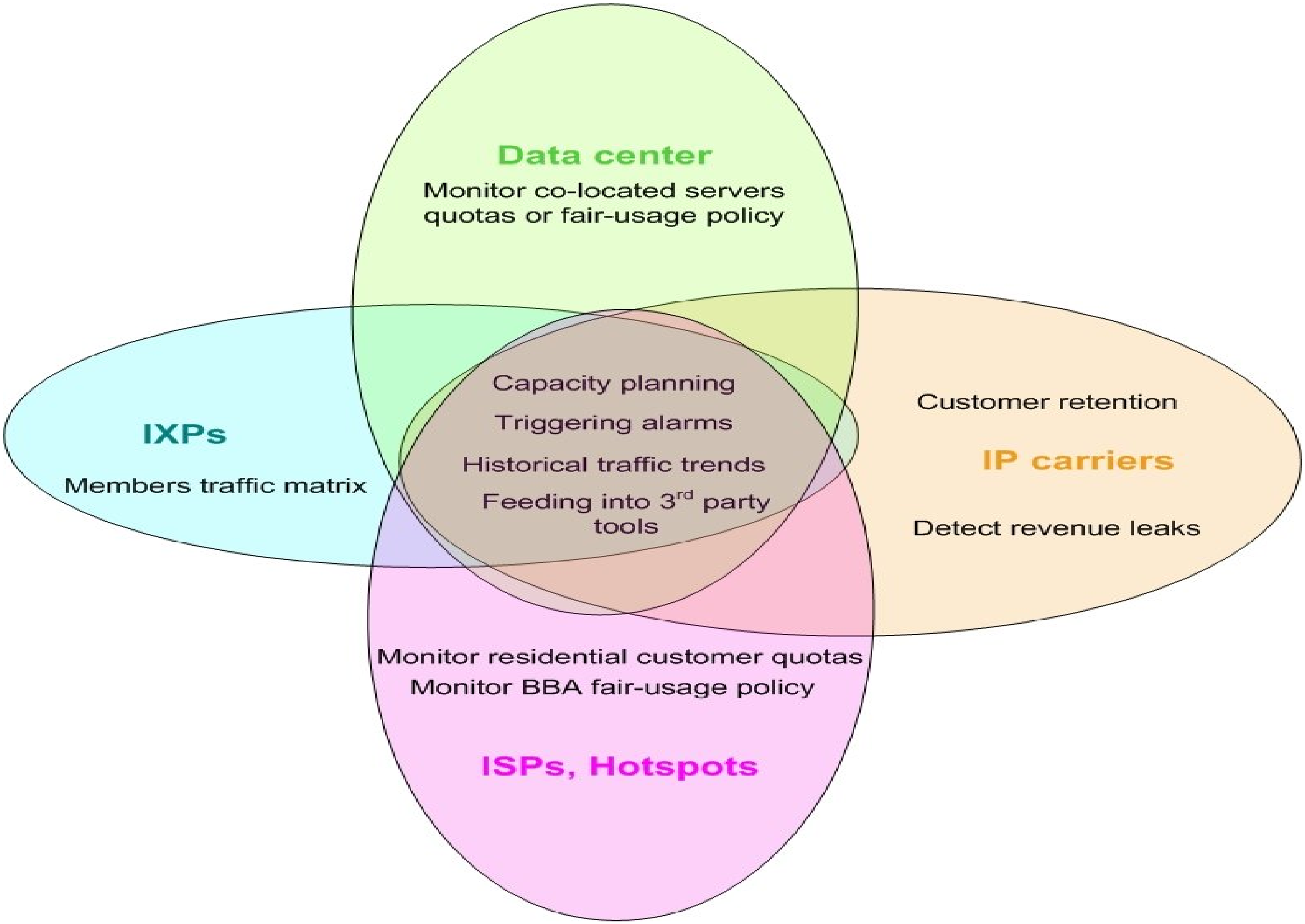
pmacct is open-source, free, GPL'ed software



pmacct: where is it sitting?



pmacct: typical usage scenarios



pmacct: introducing BGP natively into a NetFlow/sFlow collector

Agenda

I. Introduction

II. **Recent developments**

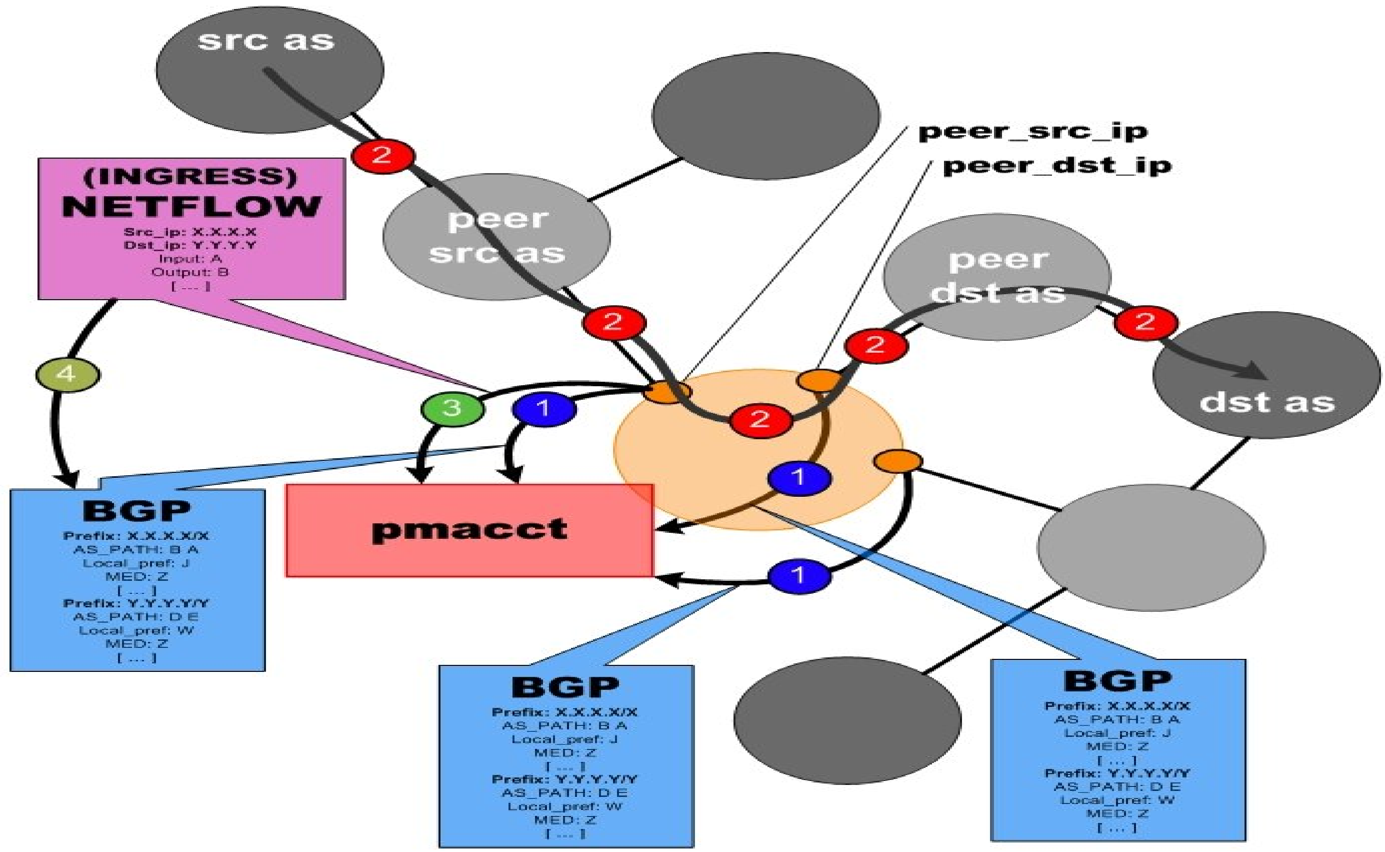
The BGP peer who came from NetFlow (and sFlow)

- pmacct introduces a Quagga-based BGP daemon
  - Implemented as a parallel thread within the collector
  - Doesn’t send UPDATEs and WITHDRAWs whatsoever
  - Behaves as a passive BGP neighbor
  - Maintains per-peer BGP RIBs
  - Supports 32-bit ASNs
  - Supports both IPv4 and IPv6
- Joins NetFlow (or sFlow) and BGP data basing on:
  - NetFlow source address == BGP source address

The BGP peer who came from NetFlow (and sFlow) (cont.d)

- Relevant implementation details:
  - Bases on trust: peers are not defined but a max number of peers who can connect is defined instead
  - Ensures iBGP peering by presenting itself as part of the AS of the neighbor, as read in the OPEN message
  - Enables the following traffic aggregation primitives: AS path, Local Preference, MED, Peer ASNs (freely mixed with origin ASNs), Communities, Peer IPs
- Caveats:
  - BGP multi-path is not supported

The BGP peer who came from NetFlow (and sFlow) illustrated



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

Theory applied, SQL example (1/3)

```
shell> cat pmacct-create-db_bgp_v1.mysql
[ … ]
create table acct_bgp (
Tag             agent_id 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,
BGP             peer_ip_src CHAR(15) NOT NULL,
Fields          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,
Counters        packets INT UNSIGNED NOT NULL,
                bytes BIGINT UNSIGNED NOT NULL,
Time            stamp_inserted DATETIME NOT NULL,
                stamp_updated DATETIME,
                PRIMARY KEY (…)
);

shell> mysql -u root -p < pmacct-create-db_bgp_v1.mysql
```

```
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
[ … ]
```

Theory applied, SQL example (2/3)

- See traffic delivered to a specific BGP peer

```
mysql> SELECT  peer_as_dst, bytes, stamp_inserted \
       FROM acct_bgp \
       WHERE peer_as_dst = <peer | customer | IP transit> \
       GROUP BY peer_as_dst
```

- Same as above but also per location if peering at multiple places

```
mysql> SELECT  peer_as_dst, peer_ip_dst, bytes, stamp_inserted \
       FROM acct_bgp \
       WHERE peer_as_dst = <peer | customer | IP transit> \
       GROUP BY peer_as_dst, peer_ip_dst
```

- Simply replace “dst” with “src” in the above examples to see incoming traffic

Theory applied, SQL example (3/3)

- See outgoing traffic breakdown to all BGP peers of the same kind (ie. peers, customers, transit)

```
mysql> SELECT  peer_as_dst, bytes, stamp_inserted \
       FROM acct_bgp \
       WHERE local_pref = <<peer | customer | IP transit> pref> \
       GROUP BY peer_as_dst
```

Why integrating BGP at the collector?

- One might argue validity of the work: “*hey, this all could have been done at the router!*”.
- Maintaining per-peer BGP RIBs at the collector has several advantages; some implemented:
  - Follow default route, having the BGP RIB of the default gateway of a PE with partial BGP view or default-only
  - Decouple NetFlow from BGP by mapping NetFlow agents to BGP peers
  - Give better chances to the source peer-AS (later)

Miscellaneous features implemented

- AS Path radius:
  - AS PATHs can get long. This can easily get counter productive (ie. waste space)
  - Cuts AS Path down to the specified number of AS hops. *Assumption*: people might be more interested into what happens around them.
- Communities pattern filter
  - IP prefixes can have many communities attached
  - Only a small subset might be relevant to the accounting infrastructure
  - Hence filtering capabilities are made available

The source peer AS issue

- Traffic is traditionally routed to destination
- Problem #1: limited information about where traffic enters the AS domain. Applying this to traditional NetFlow it means: either origin ASNs OR peer ASNs
- Problem #2: asymmetric routing. Applying this to traditional NetFlow it means: FIB lookup on the source prefix; result: source peer AS is where traffic should have entered the AS domain, *if it was symmetrical*.

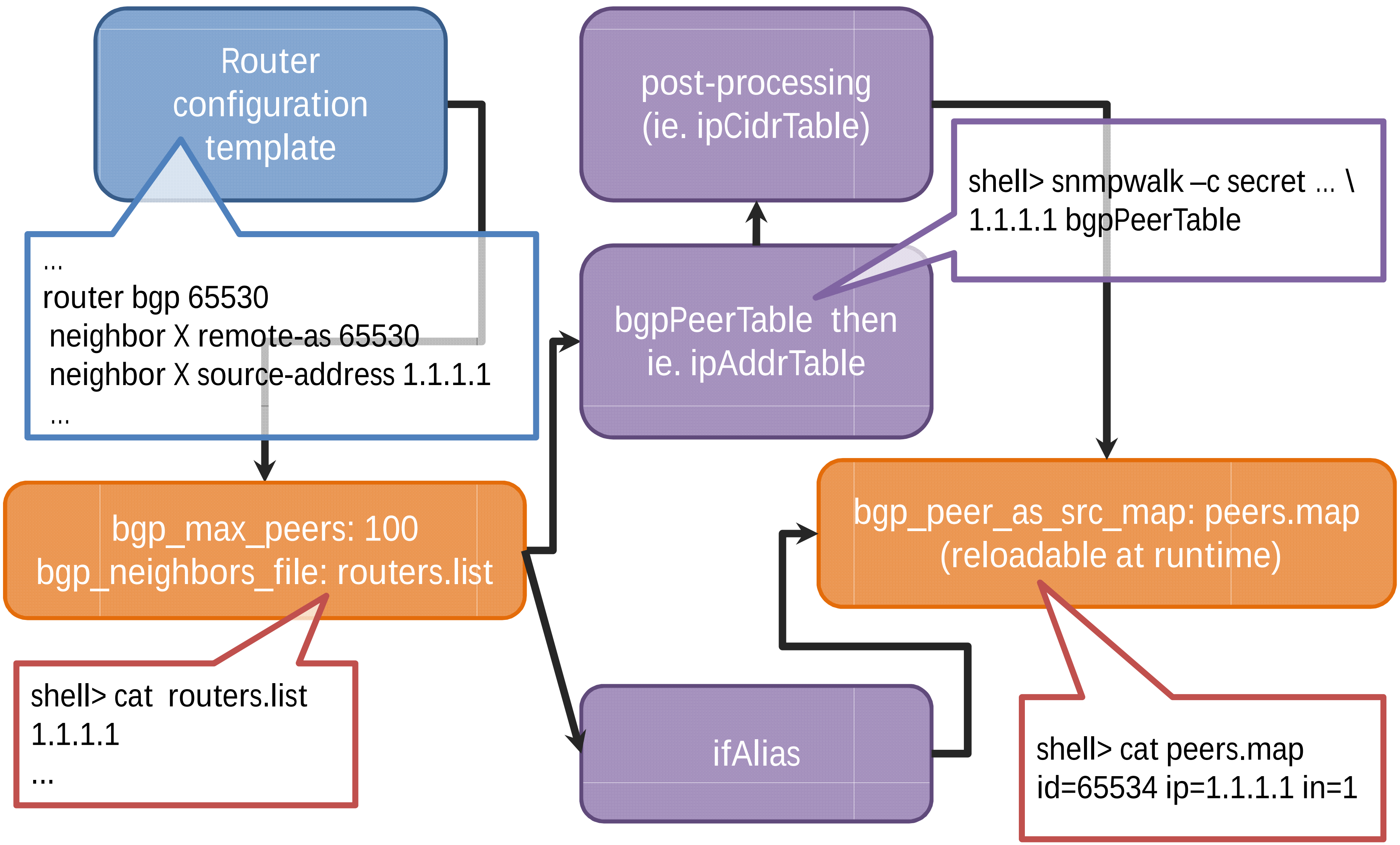
The source peer AS issue mitigated

- Having per-peer BGP RIBs on-board paves the way to capture both origin AND peer ASNs
- Source peer AS is statically mapped against:
  - A whole port, ie. input port field.
  - Source MAC address. Should be the way to go and depends on NetFlow v9 or sFlow.
  - BGP next-hop lookup against the source prefix:
    - Only choice if running NetFlow v5 in scenarios where multiple peers are off a single port (ie. IXP)
    - Accuracy is not really predictable
    - Combine with input port to limit false positives

Auto-discovery of source peer ASNs

- Mix of: SNMP, “bgp_neighbors_file” feature and pmacct maps reloadable at runtime:
  - Fetch all eBGP peers from bgpPeerTable
  - Having local IP addresses fetch ifIndex from ipAddrTable
  - Detect multiple eBGP peers (ie. IXP scenarios) off the same (sub-)interface so that src_mac or bgp_nexthop can be appended. For example, check netmask length
  - Detect eBGP multi-hops by checking ifIndex against Loopback interfaces. If true, resolve in ipCidrTable
- Establish a small set of rules within your group, ie. when setting interface descriptions (ifAlias)

Auto-discovery of source peer ASNs



The case of entities on the provider IP address space

- For such entities:
  - AS PATH would be empty;
  - origin and peer ASNs would be NULL.
- Problem: how to recognize them?
- Include IP prefix information in the DB structure: cumbersome and wouldn’t scale.
- Split approach:
  - Source: map ifIndexes or MAC addresses to peer AS
  - Destination: rely on BGP communities

The case of entities on the provider IP address space (cont.d)

– Ad-hoc feature (bgp_stdcomm_pattern_to_asn):
  - Prerequisite: entities on the provider IP address space get tagged with a standard BGP community. You are in full control of the granularity of the assignment.
  - pmacct config: bgp_stdcomm_pattern_to_asn: XXX:Y..
  - Say, an entity is tagged with community XXX:YYY; XXX value is mapped to the peer AS; YYY value is mapped to the origin AS.
  - Works transparently for both sources and destinations; in future can make use of extended BGP communities.

Miscellaneous implementation details

- A BGP RIB of ~275-290K entries accounts for some 35-40MB of memory. Multiply this by the number of peers giving the full table.
- Attributes table is shared amongst all the RIBs; it typically takes ~10MB of memory.
- BGP RIB implements a binary tree; attributes table implements hashing.
- RIB entries have pointers to attributes; once an attribute is not referenced by any RIB entry, it is removed.

pmacct: introducing BGP natively into a NetFlow/sFlow collector

Thanks for your attention!

Questions?

Paolo Lucente

the pmacct project | AS286

<paolo at pmacct dot net>

http://www.pmacct.net/

UKNOF #14 meeting, London, 11th Sep 2009