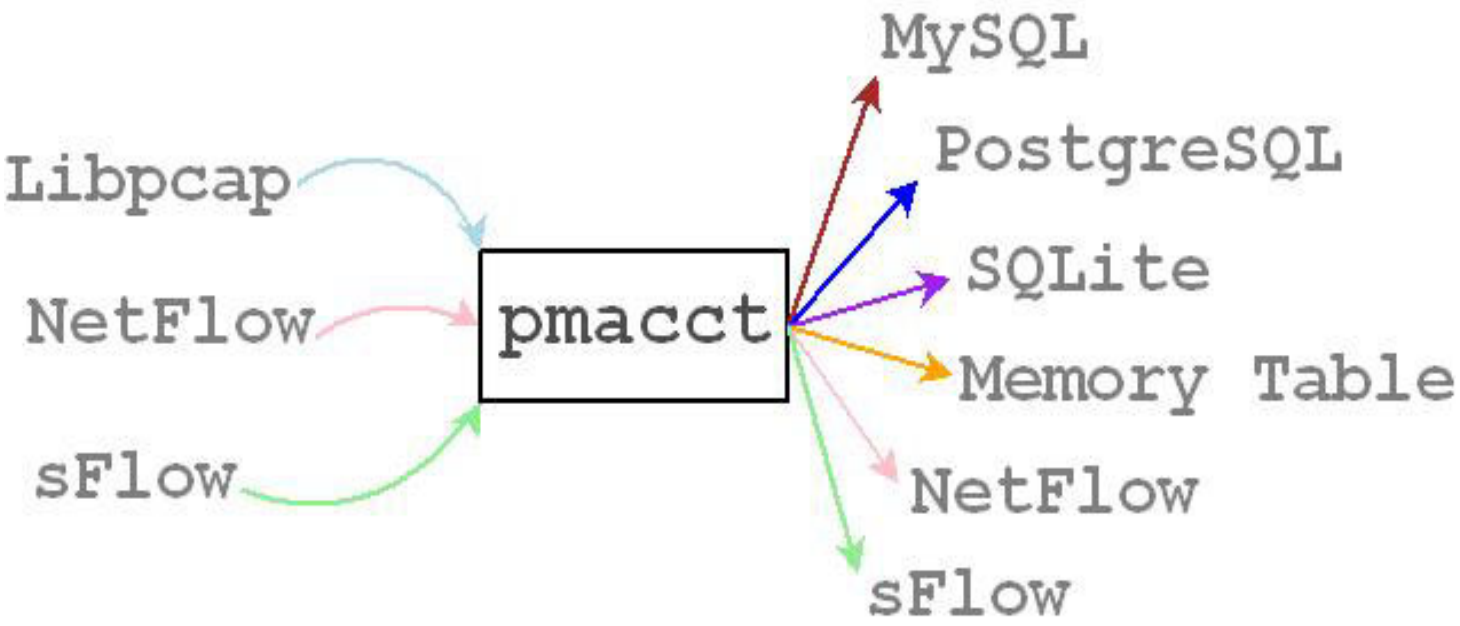
“IP accounting reloaded, the **pmacct** project: NetFlow, sFlow, SQL, RRD, stream classification and much more …”

Paolo Lucente *<paolo at pmacct dot net>*

“IP accounting reloaded, the **pmacct** project: NetFlow, sFlow, SQL, RRD, stream classification and much more …”

Part I, Introduction

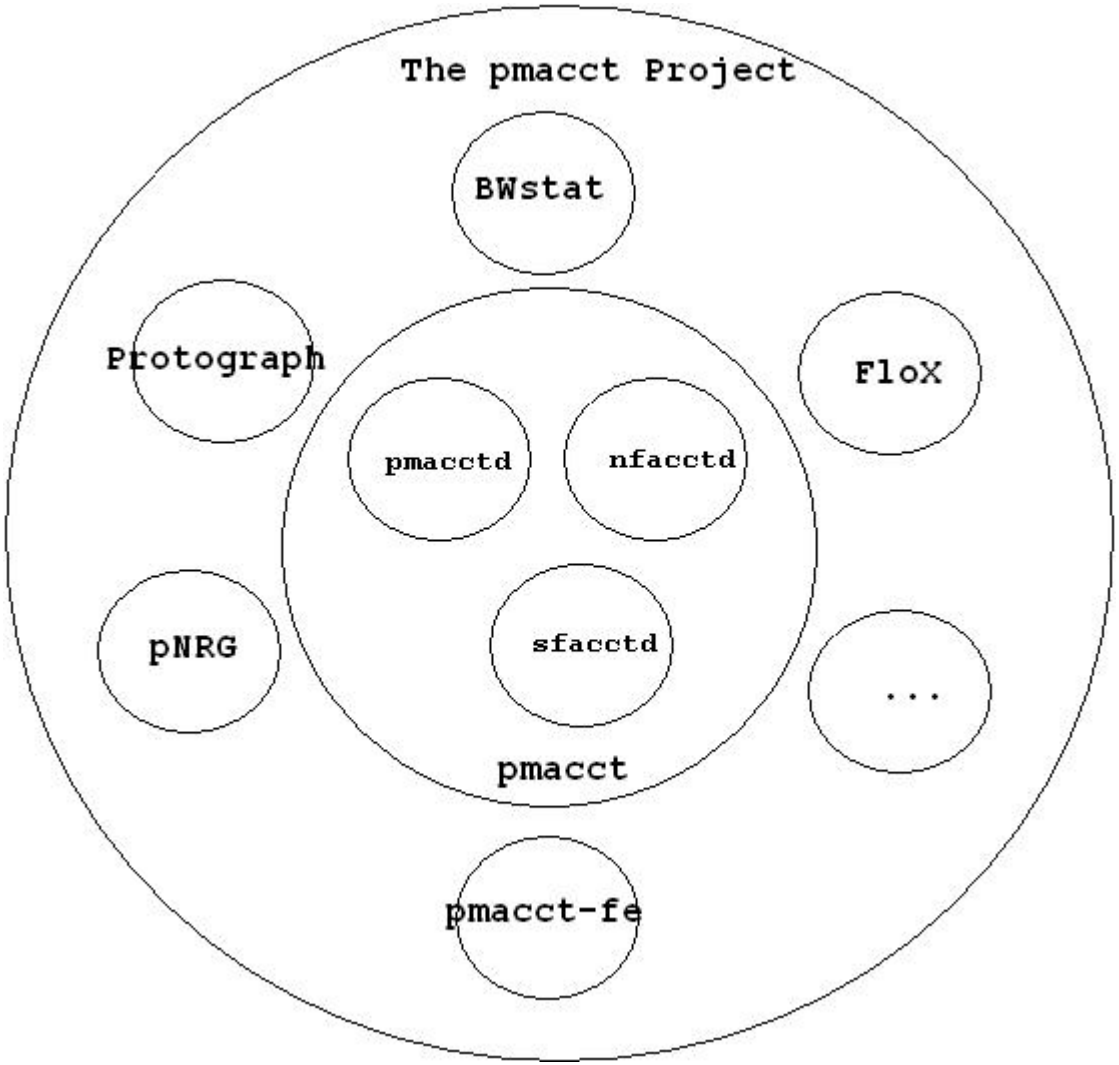
pmacct, what is this?



pmacct, an overview

- **Runs on Linux, BSDs and Solaris**
- **Support for both IPv4 and IPv6**
- **Collects data through libpcap, NetFlow v1/v5/v7/v8/v9 and sFlow v2/v4/v5**
- **Saves data directly to Memory Tables and RDBMS (MySQL, PostgreSQL and SQLite)**
- **Exports data to remote collectors through NetFlow v5/v9 and sFlow v5**
- **Flexible architecture to tag, filter, redirect, aggregate and split captured data**
- **Classification of traffic streams**
- **Support for sampling and renormalization**
- **Pluggable architecture to ease integration of new capturing environments and data backends**
- **Careful SQL support: data pre-processing, triggers, recovery methods, dynamic table naming**
- **Supported by 3rd-party tools specifically aimed to presenting and graphing network data**

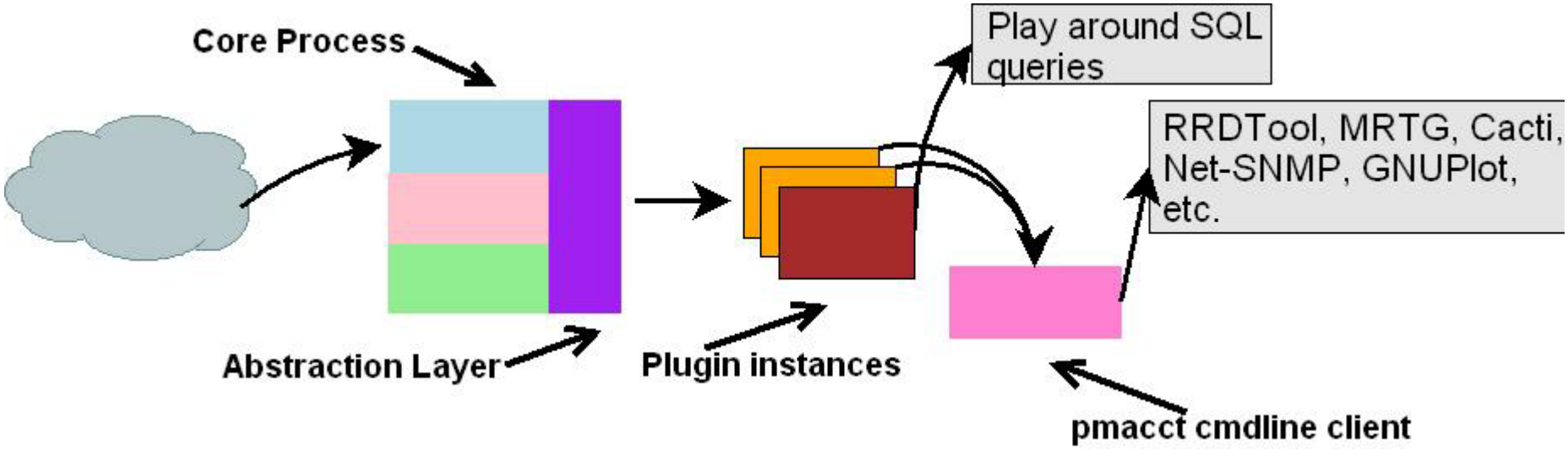
The **pmacct** project



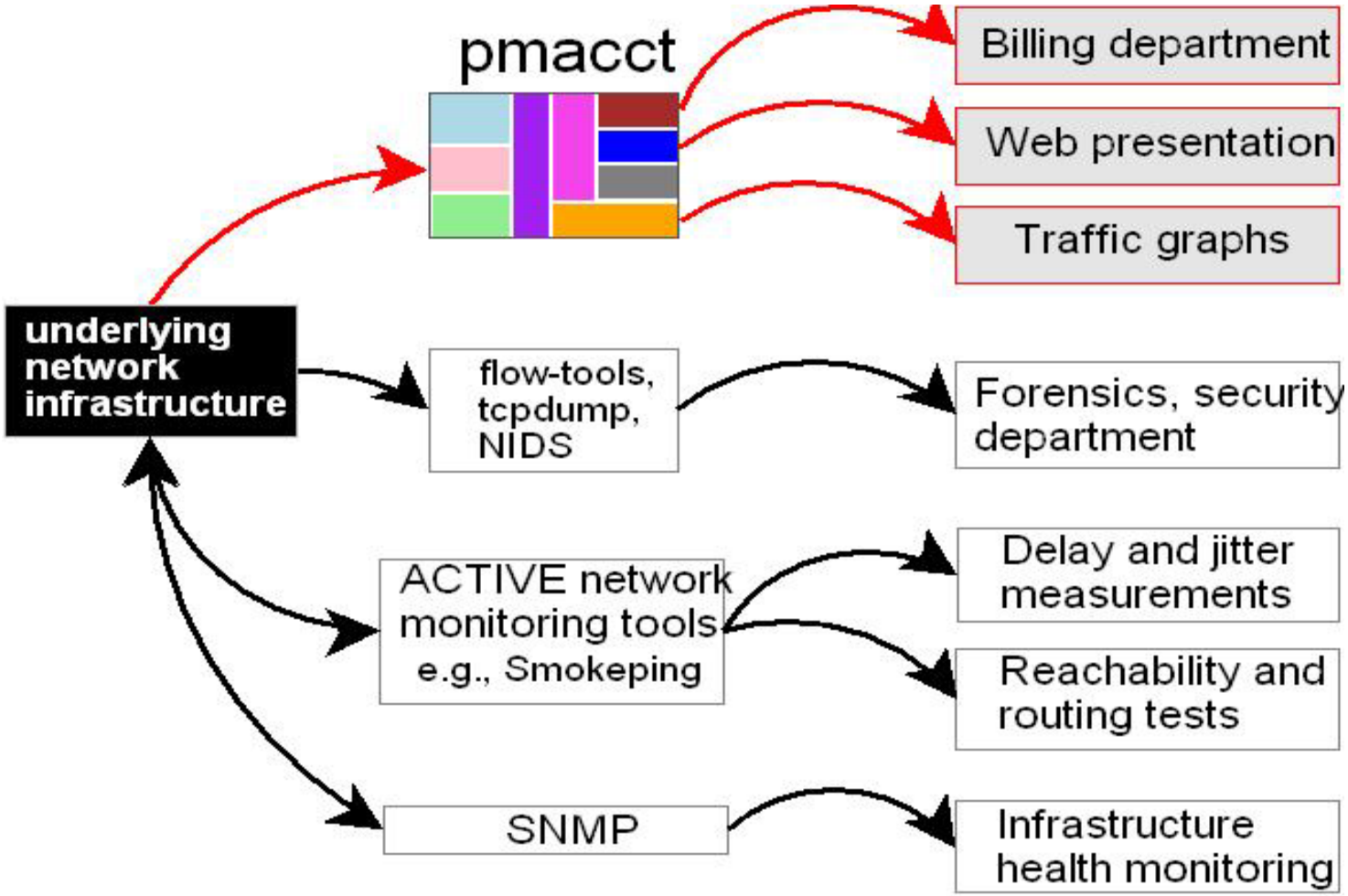
pmacct, sample configuration

```
daemonize: true
interface: eth0
plugins: memory[a], memory[b], mysql[c]
networks_file: /tmp/networks.lst
!
aggregate[a]: src_host
imt_path[a]: /tmp/acct_out.pipe
aggregate_filter[a]: src net 10.0.0.0/16
!
aggregate[b]: dst_host
imt_path[b]: /tmp/acct_in.pipe
aggregate_filter[b]: dst net 10.0.0.0/16
!
aggregate[c]: sum_host
sql_db[c]: pmacct
sql_table[c]: acct_v1
sql_refresh_time[c]: 90
sql_history[c]: 10m
```

pmacct, the modular architecture



A possible taxonomy



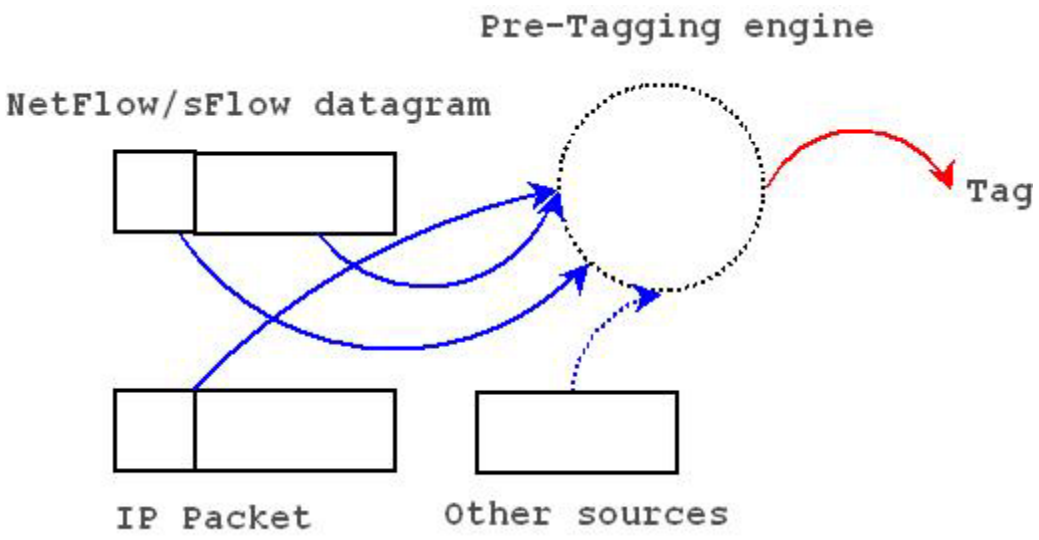
“IP accounting reloaded, the **pmacct** project: NetFlow, sFlow, SQL, RRD, stream classification and much more …”

Part II, Recent news

Some recent news

- **Stream classification through Regular Expressions, Shared Objects. A thread-aware version has been also developed**
- **Tagging infrastructure has been improved through the introduction of new 'label', 'jeq', 'return' and 'stack' directives**
- **sFlow v5 and NetFlow v5/v9 export capabilities have been introduced**
- **Various SQL enhancements: sql_max_writers, sql_aggressive_classification, sql_use_copy, sql_history_since_epoch**

Pre-Tagging: the idea



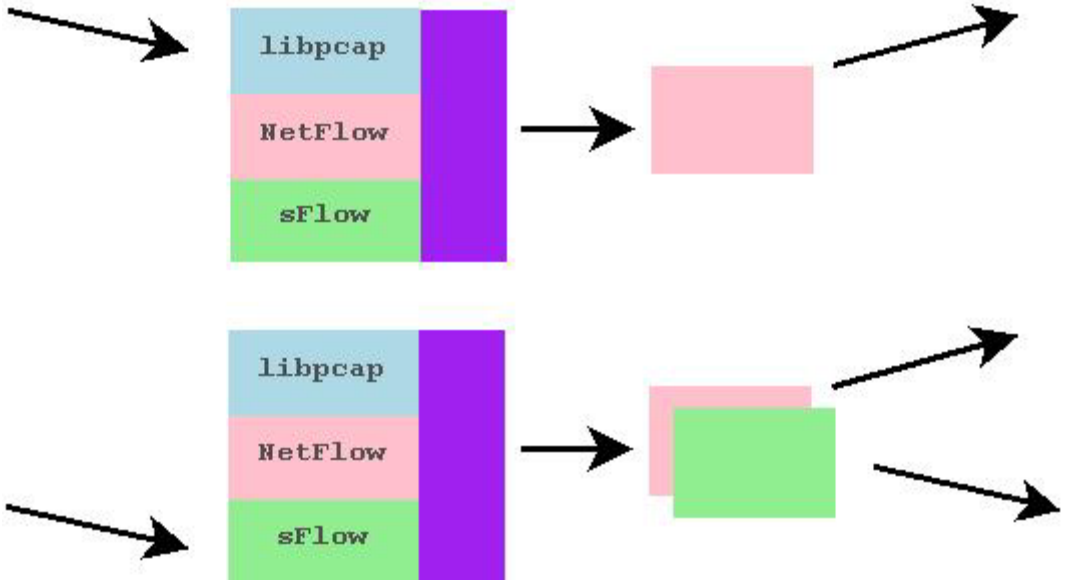
Pre-Tagging: how it works

- **Multiplexes informations into a single "tag" field and consists of a set of rules**
- **The first matching rule wins, ie. like firewall rules**
- **The "tag" can be filtered out on a per-plugin basis**
- **Supported keys include: input and output interfaces, source and destination AS numbers, BGP next-hop, libpcap-style filters, AgentID/Engine ID-Type, *AS path***
- **Newly introduced keys ("label", "jeq" and "return") are aimed to change the default rule-flow, which may result extremely useful**

Pre-Tagging: examples

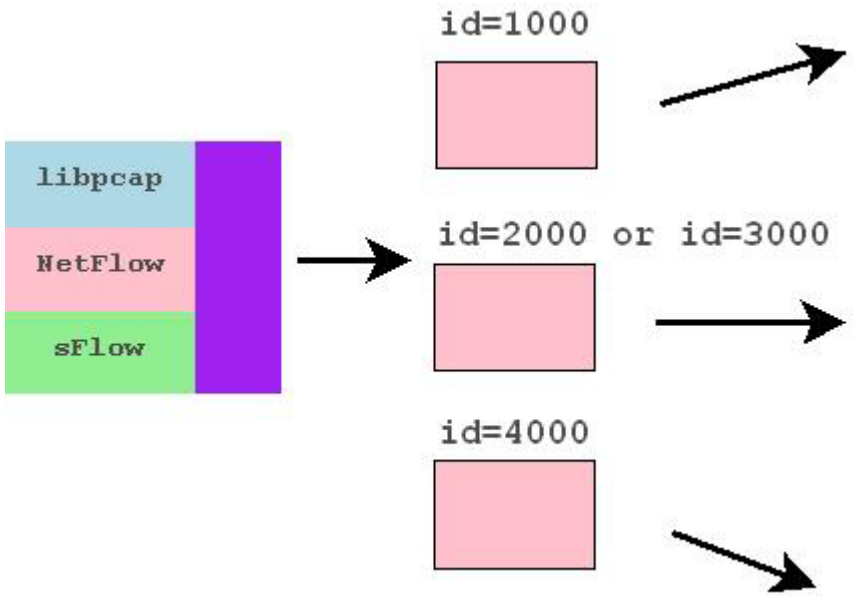
```
id=3000 filter='net 10.0.0.0/8‘

id=1000 ip=192.168.1.1 in=7
id=2000 ip=192.168.1.1 in=8 out=16
id=3000 ip=192.168.1.1 engine_type=1 engine_id=0
id=4000 ip=192.168.1.1 bgp_nexthop=172.16.10.1

id=8000 ip=192.168.1.1 in=10 jeq=eval_out return=true
id=8001 ip=192.168.1.1 in=20 jeq=eval_out return=true
id=8002 ip=192.168.1.1 in=30 jeq=eval_out return=true
id=9000 ip=192.168.1.1 out=1 label=eval_out
id=9001 ip=192.168.1.1 out=2
id=9002 ip=192.168.1.1 out=3
```

Exporting NetFlow and sFlow …

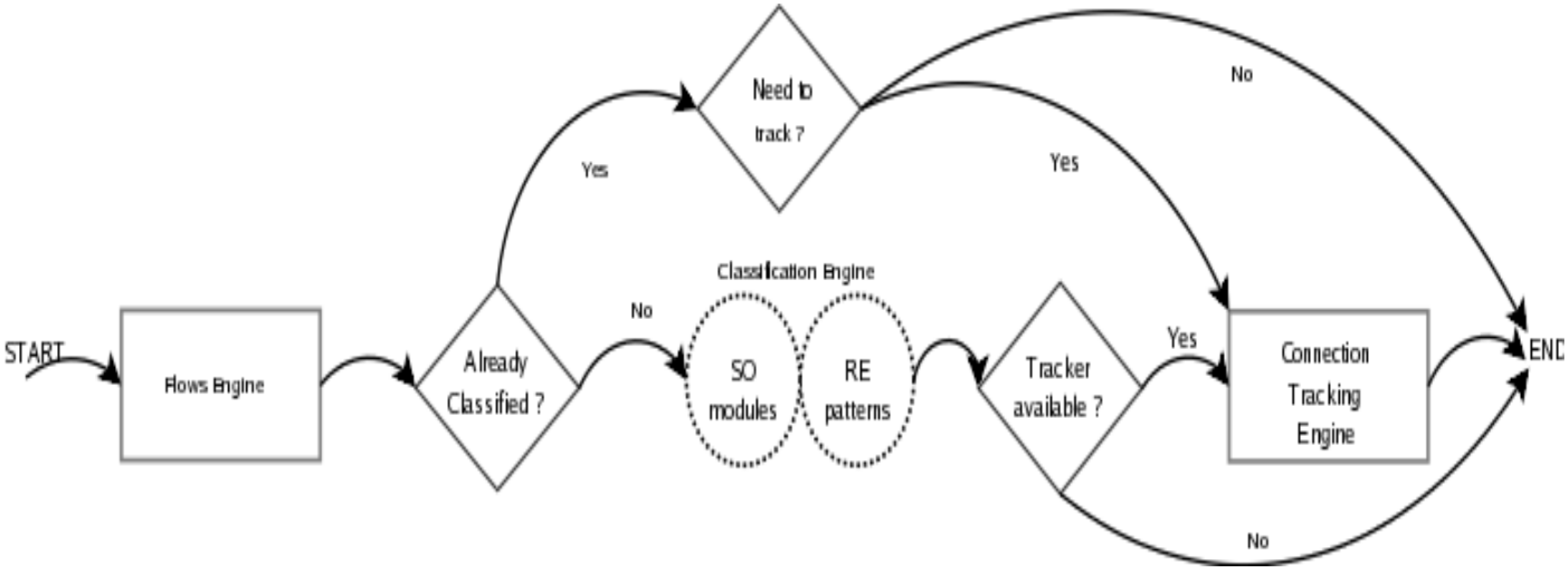


… and now involving tags

```
id=1000 ip=192.168.1.1 engine_type=100 engine_id=1
id=2000 ip=192.168.1.1 filter='net 10.0.0.0/8'
id=3000 ip=192.168.1.1 filter='ether host 00:01:02:03:04:05:06'
id=4000 ip=192.168.1.1 bgp_nexthop=172.16.10.1
```



pmacct: about classification



pmacct: classification, RE

An example of Regular Expressions applied to classification (this is from the L7-filter project repository):

```
http/(0\.9|1\.0|1\.1) [1-5][0-9][0-9] [\x09-\x0d -~]*(connection:|content-type:|content-length:|date:)|post [\x09-\x0d -~]* http/[01]\.[019]
```

pmacct: classification, SO

```
u_int32_t classifier(struct pkt_classifier_data *data, int caplen, void **context, void **rev_context, void **extra)
{
	struct rtp_context *ctx = NULL;
	rtp_hdr_t *hdr = (rtp_hdr_t *) data->payload_ptr;
	u_int16_t init;
	u_int8_t version, pt;

	init = ntohs(hdr->init);

	version = init >> 14;
	pt = init & 0x7f;

	if ( version == 2 && (pt < 35 || pt >= 96) ) { /* Possibly, we are facing a RTP stream */
		if (!(*context)) {  /* We don't have enough data about the stream */
			ctx = malloc(sizeof(struct rtp_context));
			if (ctx) {
				ctx->seq = ntohs(hdr->seq);
				*context = ctx;
			}
			return 0;
		}
		else {
			ctx = (struct rtp_context *) *context;
			if (ntohs(hdr->seq) == ctx->seq+1) return 1;
		}
	}
	return 0;
}
```

pmacct: classification, RE vs. SO

- ✓ Regular Expressions (RE) classifiers are proficient against the packet payload, easy to develop and suitable for text-based protocols.
- ✓ Shared Object (SO) classifiers are powerful (ie. because of contexts), not limited to just catch patterns (ie. Machine Learning tecniques) and deal smoothly with binary-encoded protocols. BUT require extensive and careful development.

An example: classification and “top N”

```
shell> psql -U pmacct -c "SELECT class_id, packets, bytes, flows \
      FROM acct_v5 \
      ORDER BY bytes DESC \
      LIMIT 10;"

    class_id     |  packets      |    bytes           |  flows
-----------------+---------------+--------------------+-------------
  nntp           | 533424546 | 534913922183 |      13480
  http           | 567179034 | 409970727835 | 22581928
  smtp           | 336913736 | 116445824169 | 17286471
  ssh            | 139908289 | 108291107166 |   1110903
  edonkey        | 167213900 | 107343376842 |   4501937
  ftp            | 197626712 |   97059417721 |    139749
  pop3           |   86367951 |   60221933775 |   1462006
  ssl            |   62489714 |   34784217799 |   2602435
  bittorrent     |   52031296 |   31068910458 |    414216
  rtsp           |   20099589 |    9595494054 |      3959
(10 rows)
```

"IP accounting reloaded, the **pmacct** project: NetFlow, sFlow, SQL, RRD, stream classification and much more …"

Part III, Examples

The SQL way

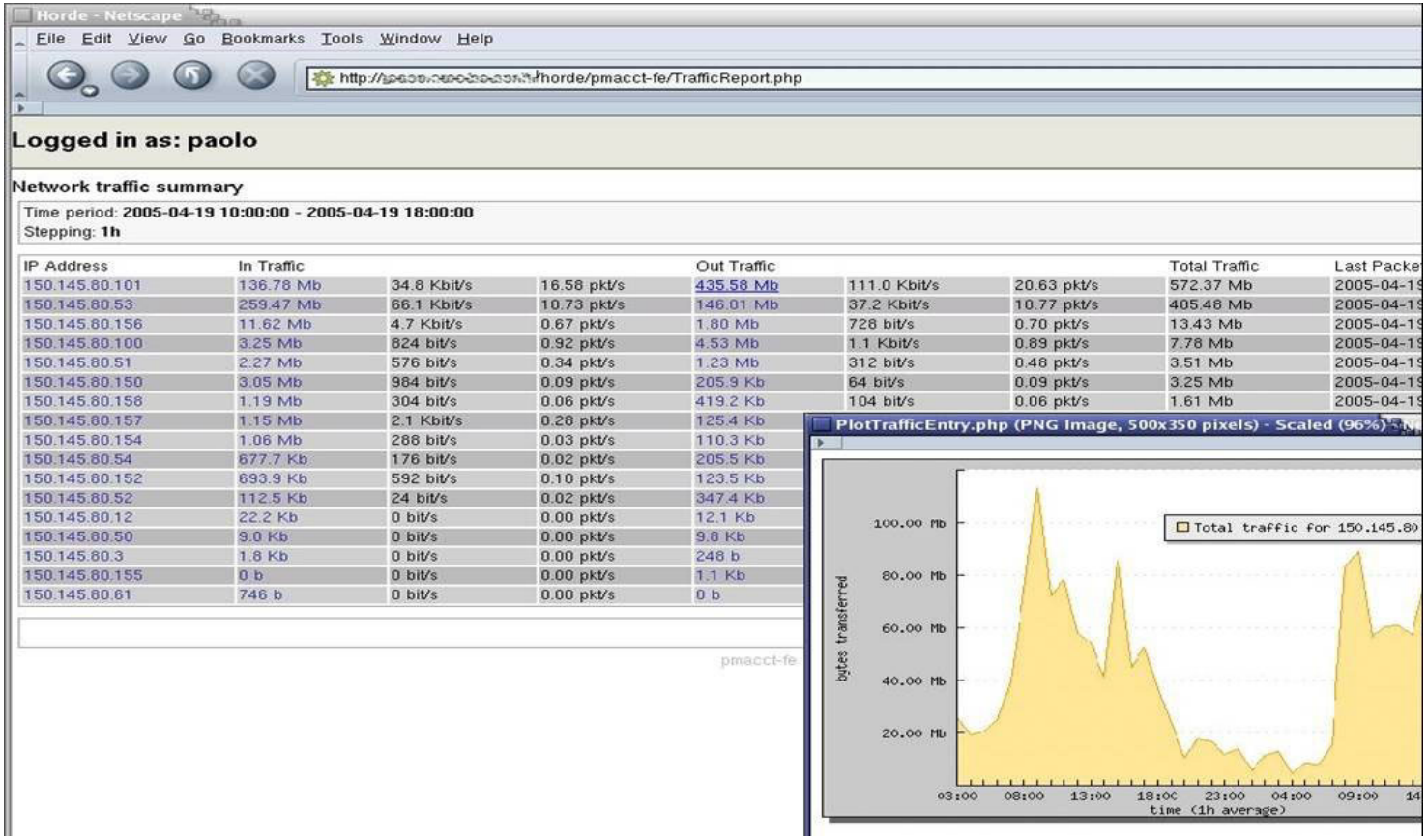
```
interface: eth0
plugins: pgsql[out], pgsql[in]
!
aggregate[out]: src_host
aggregate_filter[out]: vlan and src net 150.145.80.0/20
sql_table[out]: acct_out
!
aggregate[in]: dst_host
aggregate_filter[in]: vlan and dst net 150.145.80.0/20
sql_table[in]: acct_in
!
sql_refresh_time: 60
sql_history: 1h
sql_history_roundoff: h
sql_preprocess: minb=60000
```

The SQL way(cont.d)

```
shell> psql -U pmacct -c "SELECT * FROM acct_out \
      WHERE ip_src = '150.145.80.101‘ \
      ORDER BY stamp_inserted DESC \
      LIMIT 10;"

     ip_src          |  packets |  bytes       |   stamp_inserted       |    stamp_updated
---------------------+----------+------------+--------------------------+-------------------------
 150.145.80.101 |  355394 | 29925806 | 2006-01-08 16:00:00 | 2006-01-08 16:48:02
 150.145.80.101 |  556245 | 46096570 | 2006-01-08 15:00:00 | 2006-01-08 16:00:02
 150.145.80.101 |   26364 | 12618610 | 2006-01-08 14:00:00 | 2006-01-08 15:00:02
 150.145.80.101 |  196319 | 16508068 | 2006-01-08 13:00:00 | 2006-01-08 14:00:01
 150.145.80.101 |  341143 | 40921593 | 2006-01-08 12:00:00 | 2006-01-08 13:00:02
 150.145.80.101 |  208050 | 30011464 | 2006-01-08 11:00:00 | 2006-01-08 12:00:01
 150.145.80.101 |  196337 | 15404272 | 2006-01-08 10:00:00 | 2006-01-08 11:01:02
 150.145.80.101 |  205970 | 16656939 | 2006-01-08 09:00:00 | 2006-01-08 10:00:03
 150.145.80.101 |  376094 | 22589504 | 2006-01-08 08:00:00 | 2006-01-08 09:00:02
 150.145.80.101 |   14779 |   6913855 | 2006-01-08 07:00:00 | 2006-01-08 08:01:01
(10 rows)
```

pmacct-fe: intro

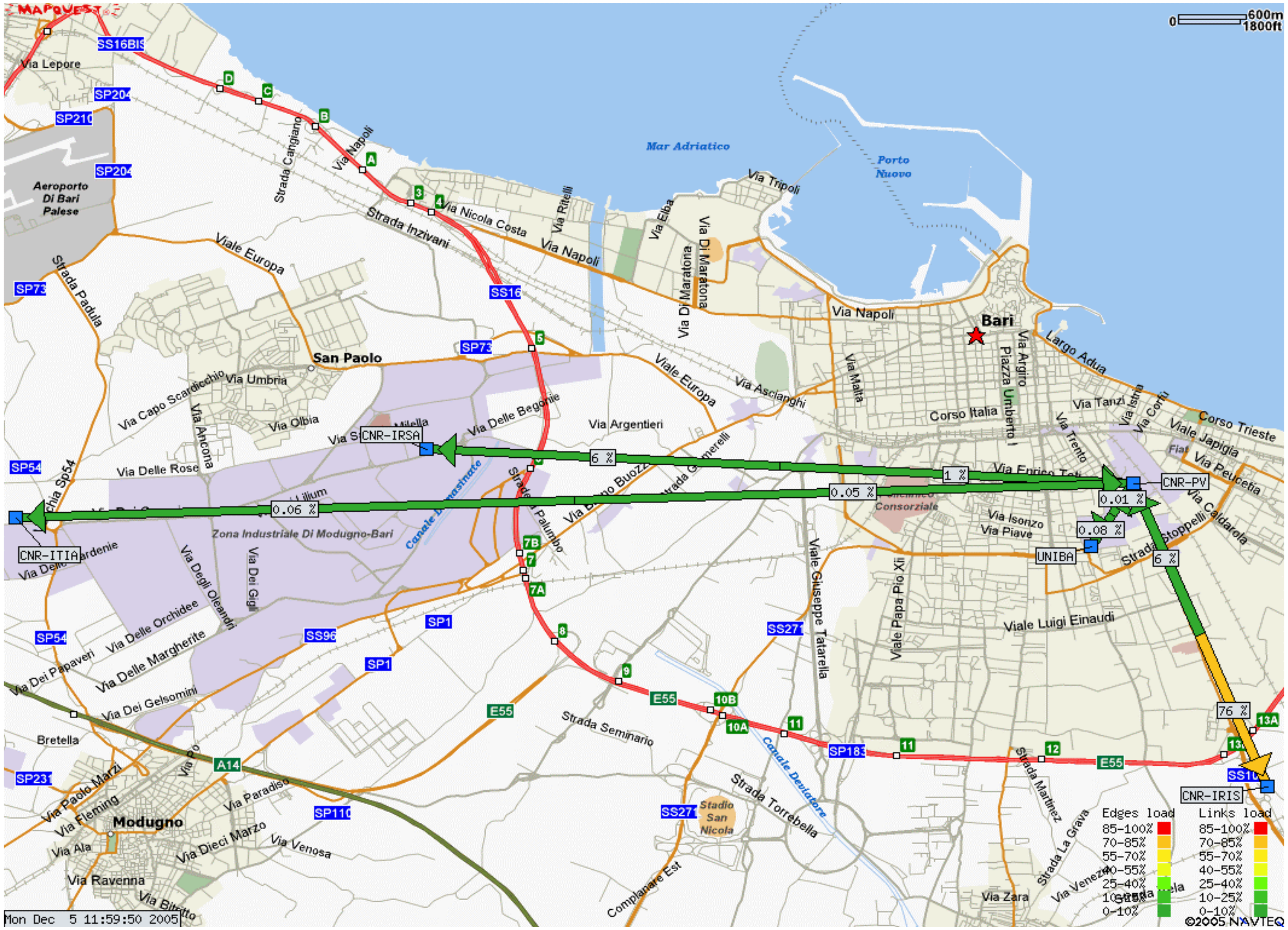
pmacct-fe: selection



GWEN: underlying pmacct configuration

```
!
! pmacctd
!
interface: eth0
plugins: memory[in], memory[out]
networks_file: /etc/pmacct/networks.list
aggregate[out]: src_net
aggregate[in]: dst_net
aggregate_filter[out]: src net 10.0.0.0/8
aggregate_filter[in]: dst net 10.0.0.0/8
imt_path[out]: /tmp/pmacct_out.pipe
imt_path[in]: /tmp/pmacct_in.pipe
```

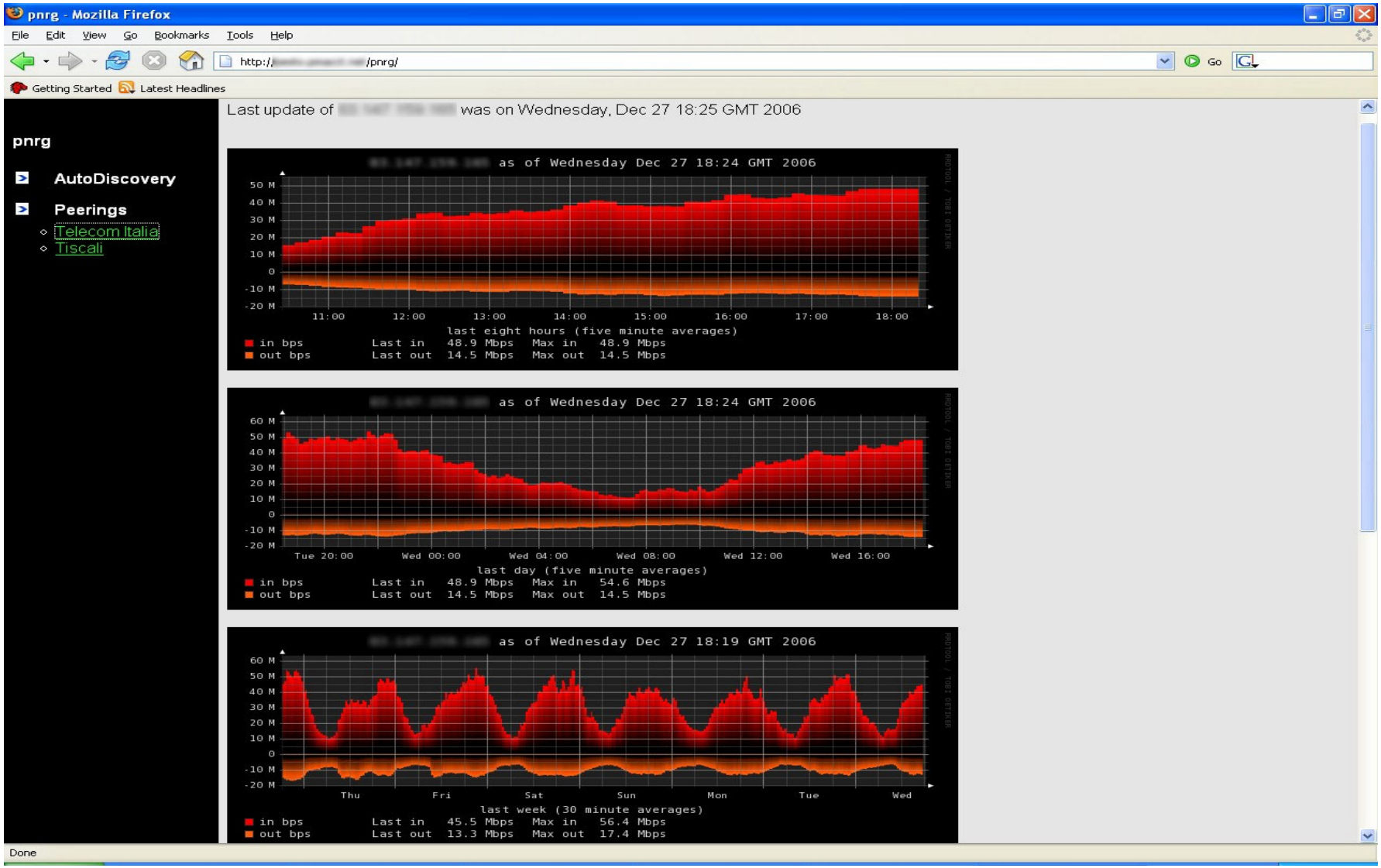
GWEN: network weather maps



PNRG: underlying pmacct configuration

```
!
! pmacctd
!
interface: eth0
plugins: memory[in], memory[out]
aggregate[out]: src_host
aggregate[in]: dst_host
aggregate_filter[out]: src net 10.0.0.0/8
aggregate_filter[in]: dst net 10.0.0.0/8
imt_path[out]: /tmp/pmacct_out.pipe
imt_path[in]: /tmp/pmacct_in.pipe
```

PNRG: RRD graphs applying SNMP auto-discovery concepts



Thanks for your attention !

http://www.pmacct.net

Paolo LUCENTE, <*paolo at pmacct dot net*>