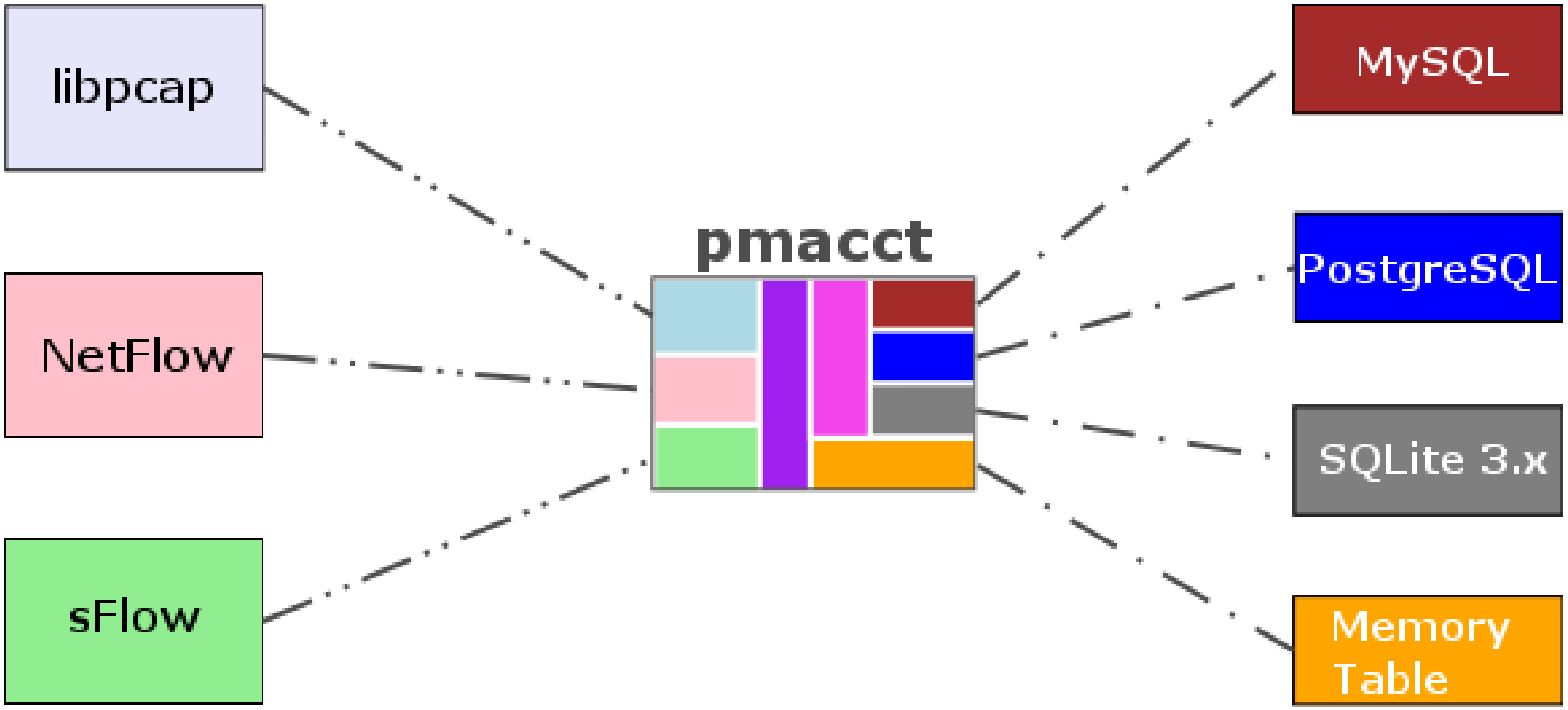
“pmacct, a new player in the network management arena”

http://www.pmacct.net

Paolo LUCENTE, CNR-Italy

Istanbul, 25 April 2006

What is pmacct ?



pmacct: why, when and how .. (I)

- The project came out of operational needs, 3 years ago (beginnings of 2003)
- At the time it was easy to get data either:
  - “static”, ie. fixed view of your network traffic data. Full stop.
  - logged on the disk in a range of proprietary format; then APIs to get in touch with them.
  - nicely arranged on the console screen or web browser of choice.

pmacct: why, when and how .. (II)

Though, we were still missing:

- A way to get data from our network, being also able to choose how to report them and supporting multiple collection methods.
- A straight way to feed network data to external applications in order to build figures, graphs, plots, sums, etc.
- A straight way to powerfulness and flexibility offered by the SQL data language.

pmacct is a PASSIVE network monitoring tool

Passive network monitoring is basically an observation point; it enables us to understand:

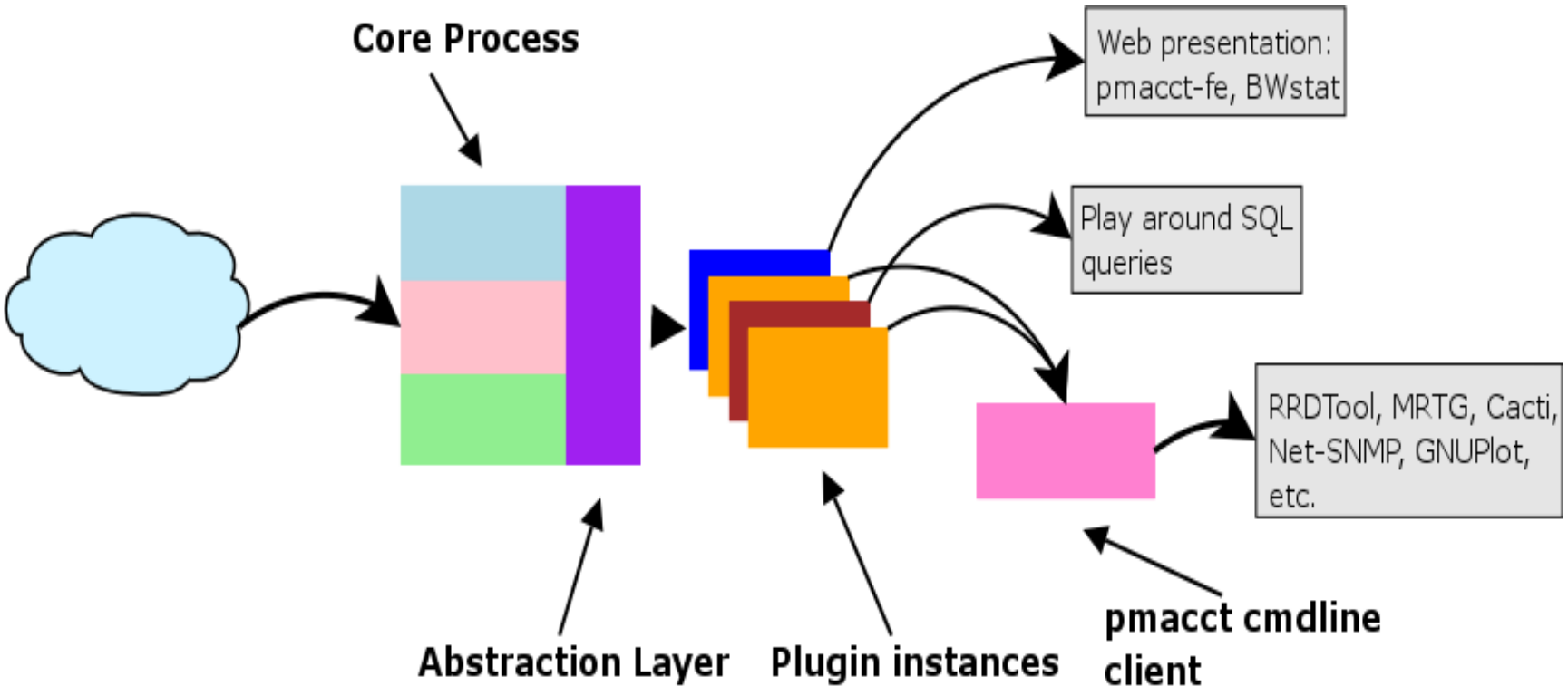
- ✓ who is using the network.
- ✓ which applications/services are most used.
- ✓ how much bandwidth is in use over the time.
- ✓ are we generating DoS / target of a worm ?
- ✓ how our BGP peerings behave.
- ✓ what is that sudden hill in the last traffic graph ?

ACTIVE network monitoring tools

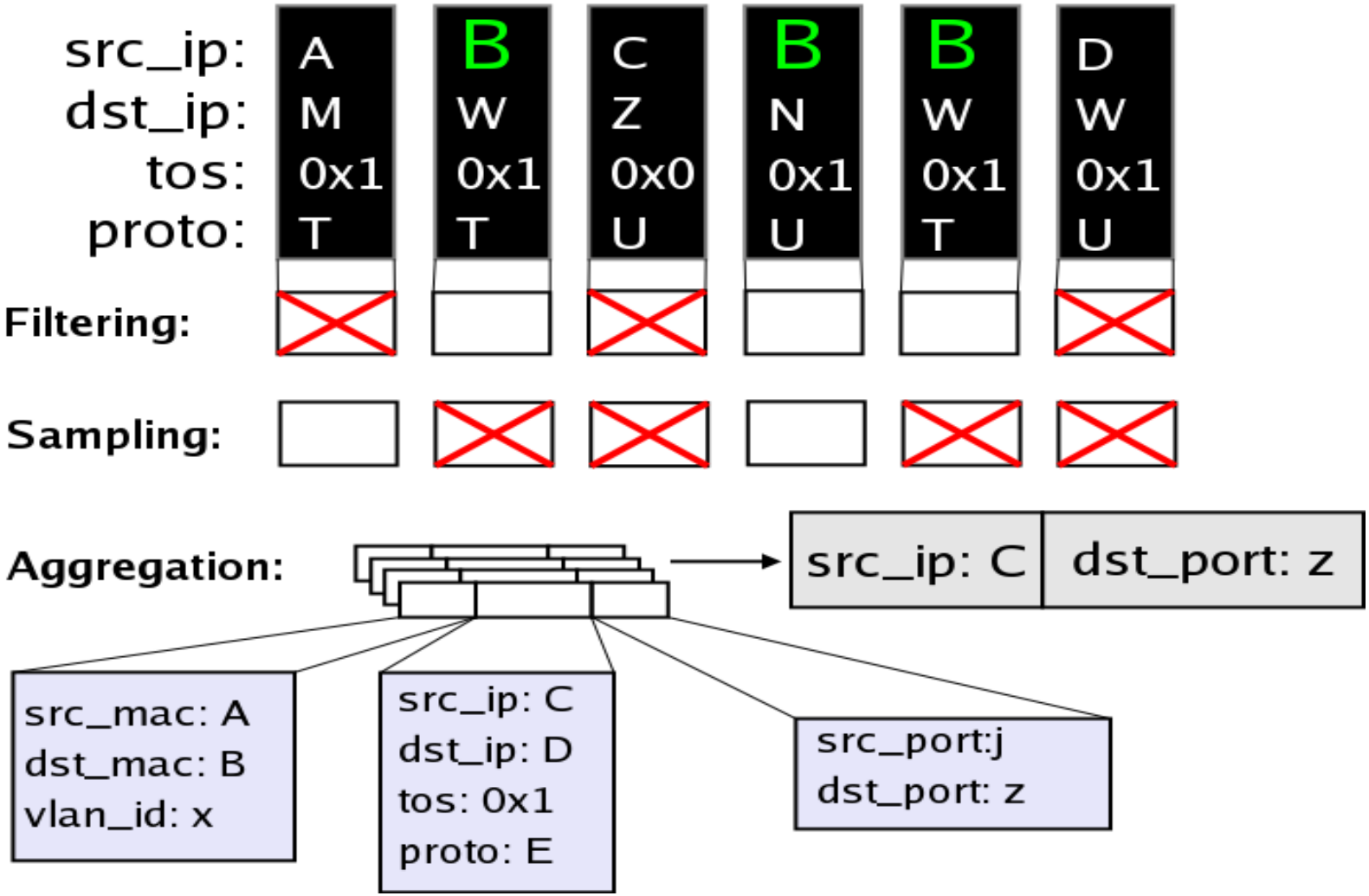
.. by contrast, they are probes injected in the network; and they enable us to understand different things:

- ✓ How many packets get lost ?
- ✓ Do all the probes have the same trip ?
- ✓ How much it takes to deliver the probe ?
- ✓ Hey, let's check that our premium IP offering works as expected under heavy traffic loads

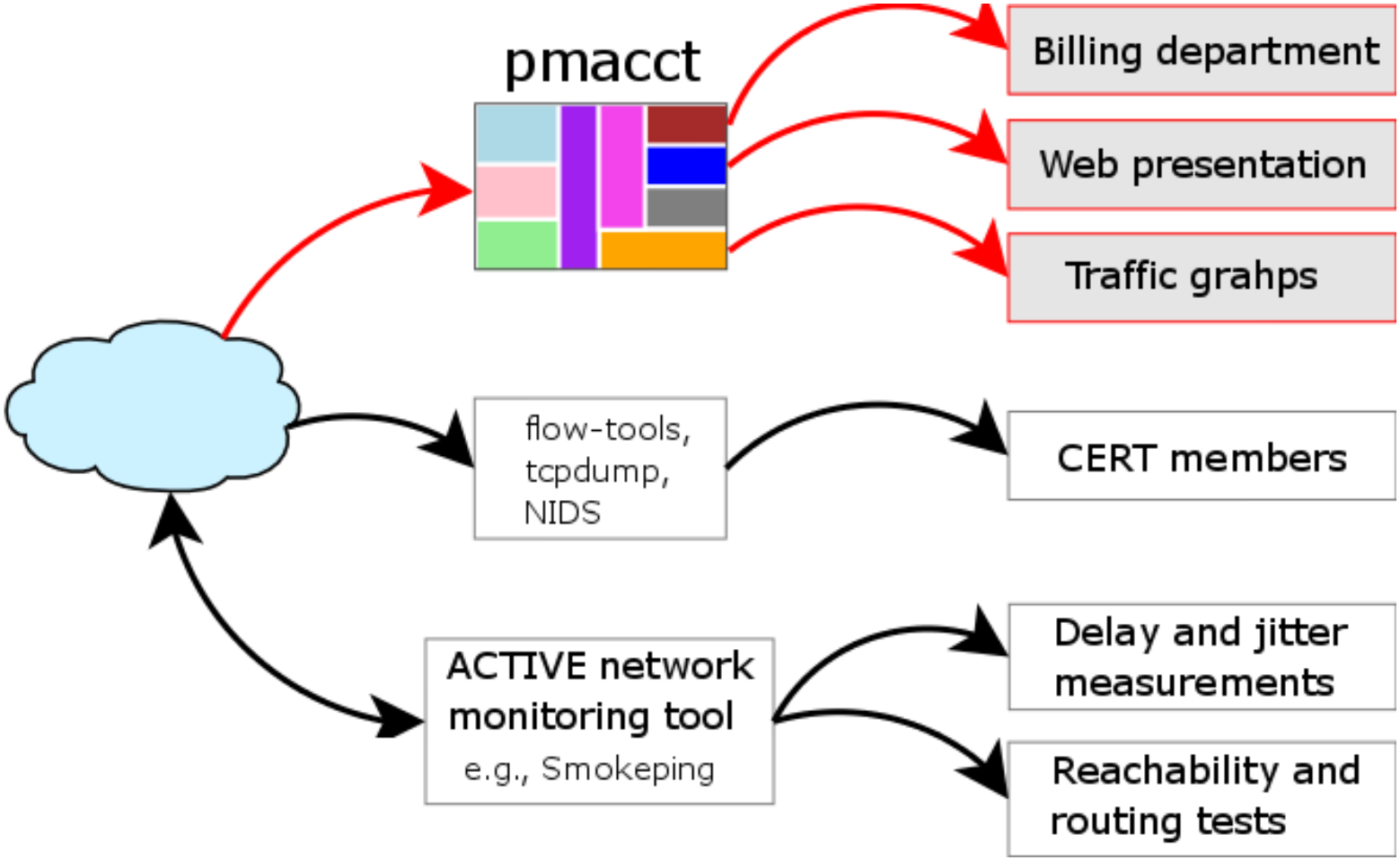
pmacct, the modular architecture: one collector, multiple views



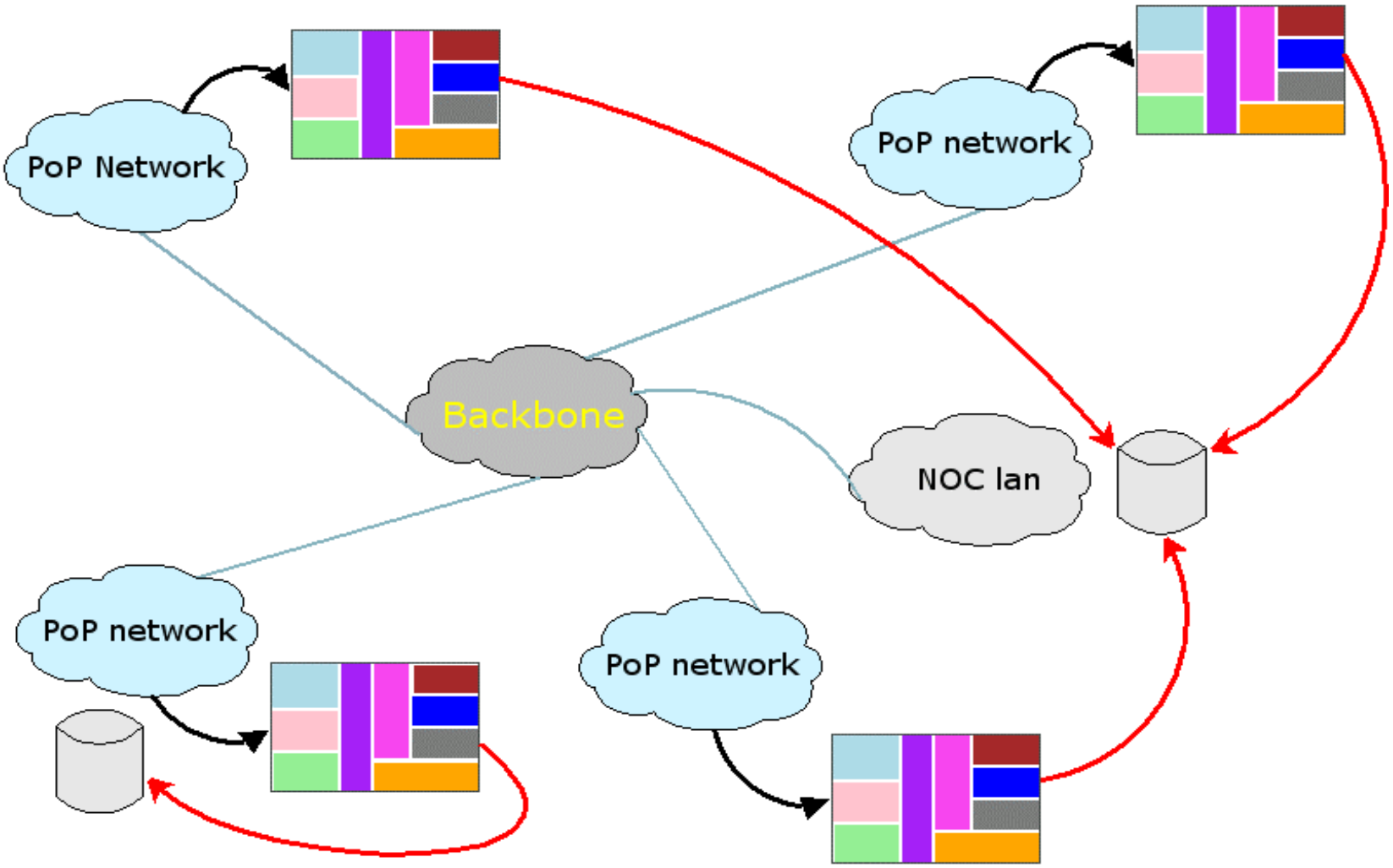
pmacct: reporting traffic data from broadband networks (I)



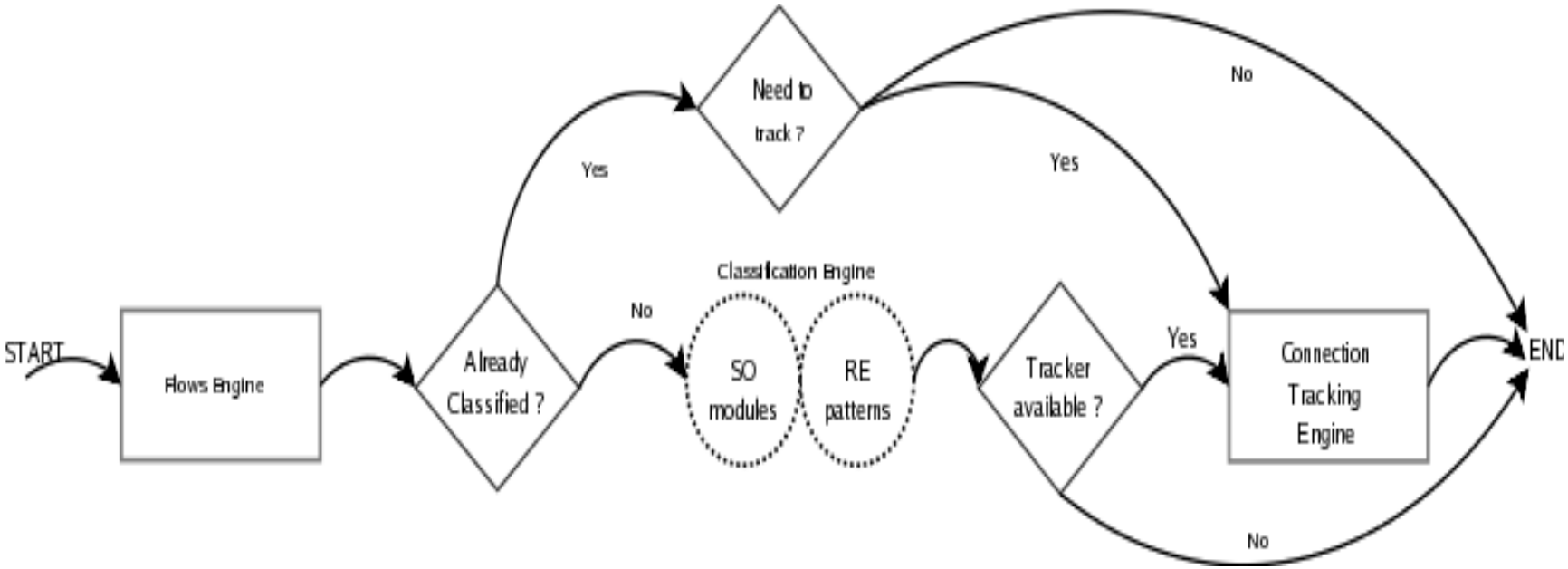
pmacct: reporting traffic data from broadband networks (II)



pmacct: an outlook of the distributed architecture



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.

“pmacct, a new player in the network management arena”

http://www.pmacct.net

Part II

Examples and results

The newbie hat: In+Out (sum) traffic per host (I)

```
shell> cat pmacctd-imt.conf
!
! pmacctd configuration example
!
interface: eth0
plugins: memory
!
aggregate: sum_host, flows
networks_file: networks.lst
```

The newbie hat: In+Out (sum) traffic per host (II)

```
shell> ./pmacct -s
SRC IP          PACKETS     FLOWS       BYTES
150.145.84.4    2           2           152
150.145.82.19   7594        38          6584356
150.145.87.15   1           1           128
150.145.90.255  2           2           466
150.145.80.51   127224      8819        23678985
150.145.81.18   2           2           460
150.145.87.159  83          11          8758
150.145.80.0    22          1           1144
150.145.87.108  1           1           247
150.145.84.156  34          9           2856
150.145.81.255  33          7           6662
150.145.82.10   1423        30          1091800
150.145.87.6    16787       3361        929034
[ … continues … ]
```

The newbie hat: In+Out (sum) traffic per host (III)

a) The –M : getting a specific entry wrapped by a formatted output

shell> ./pmacct -c src_host -M 150.145.80.101

SRC IP	PACKETS	FLOWS	BYTES
150.145.80.101	287522	2616	273081046

b) The –N : getting the counters. Introducing the -r reset flag. The quick way to glue pmacct to external tools

shell> ./pmacct -c src_host -N 150.145.80.101 -r

334701089

shell> ./pmacct -c src_host -N 150.145.80.101

2790707

Building network traffic graphs (I)

```
interface: eth0
plugins: memory[out], memory[in]
!
aggregate[out]: src_net
aggregate_filter[out]: vlan and src net 150.145.80.0/20
imt_path[out]: /tmp/pmacct_out.pipe
!
aggregate[in]: dst_net
aggregate_filter[in]: vlan and dst net 150.145.80.0/20
imt_path[in]: /tmp/pmacct_in.pipe
```

Building network traffic graphs (II)

```
shell> cat mrtg-example.sh
#!/bin/sh

unset OUT
unset IN

OUT=`pmacct -c src_host –p /tmp/pmacct_out.pipe -N 150.145.80.0 -r`
IN=`pmacct -c dst_host –p /tmp/pmacct_in.pipe -N 150.145.80.0 -r`

echo $OUT
echo $IN
echo 0
echo 0
```

Building network traffic graphs (III)

```
shell> cat mrtg.conf
[ … ]
# Target specific definitions
Target[pp]: `/usr/local/pmacct/scripts/mrtg-example.sh`
SetEnv[pp]: MRTG_INT_IP="150.145.80.0" MRTG_INT_DESCR="Server LAN"
MaxBytes[pp]: 1250000
LegendI[pp]:
Title[pp]: Server LAN
PageTop[pp]: <H1>Server LAN</H1>
<TABLE>
   <TR><TD>System:</TD>  <TD>Server LAN</TD></TR>
   <TR><TD>Maintainer:</TD> <TD>CNR-BA Staff</TD></TR>
   <TR><TD>Ip:</TD>  <TD>150.145.80.0</TD></TR>
</TABLE>
[ … ]
```

Network traffic data, the SQL way (I)

```
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
```

Network traffic data, the SQL way (II)

```
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)
```

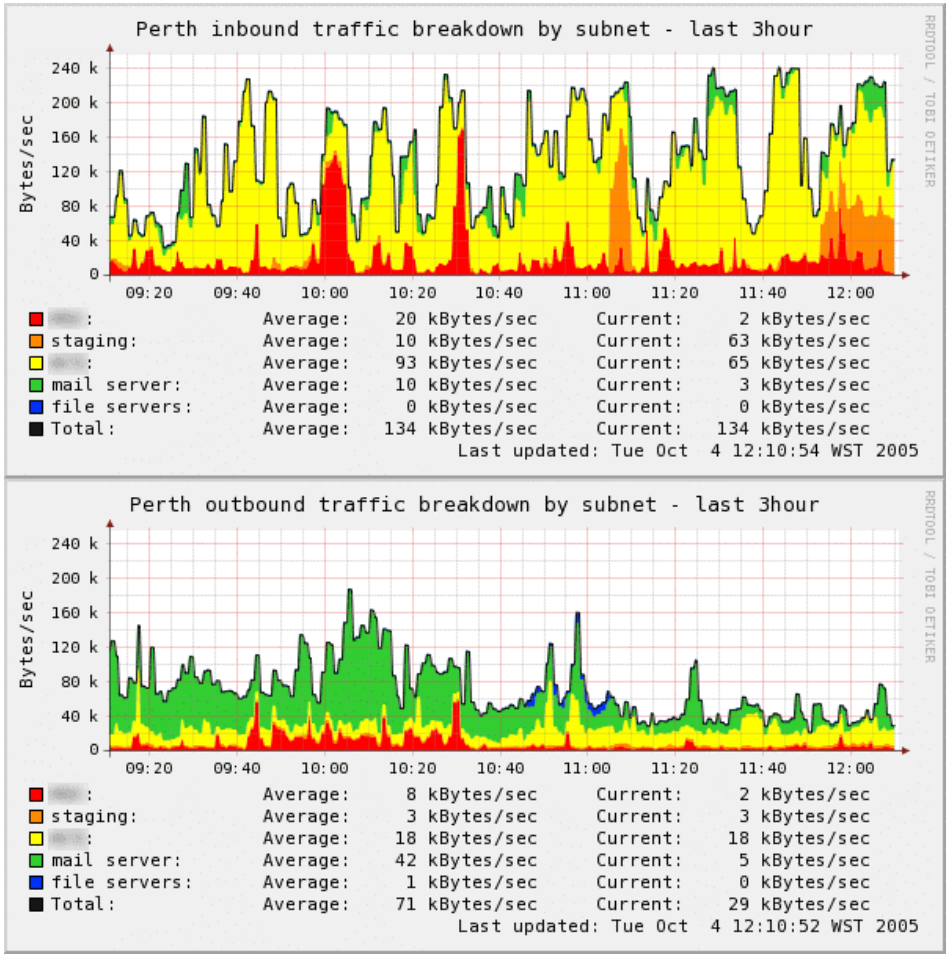
Network traffic data, the SQL way: what about “top N” ?

```
shell> psql –U pmacct –c “SELECT port_dst, ip_proto, packets, bytes \
      FROM dst_ports_db \
      WHERE dst_src = '150.145.80.101' AND \
      stamp_inserted = '2006-01-09 12:00:00‘ \
      ORDER BY bytes DESC
      LIMIT 10;”

 port_dst |  ip_proto |  packets |     bytes
----------+-----------+----------+----------------
      119 |         6 | 1084915 | 1594897858
       25 |         6 |   385883 |   374188510
       80 |         6 |    24632 |    26649410
      110 |         6 |    14595 |    15556361
       22 |         6 |    10775 |    13201890
      443 |         6 |     2943 |     1929708
      143 |         6 |      911 |     1111241
       53 |         1 |      607 |      879218
      995 |         6 |     9399 |      541329
       20 |         6 |      140 |      188855
(10 rows)
```

Network traffic data, the SQL way: classification and “top N” !

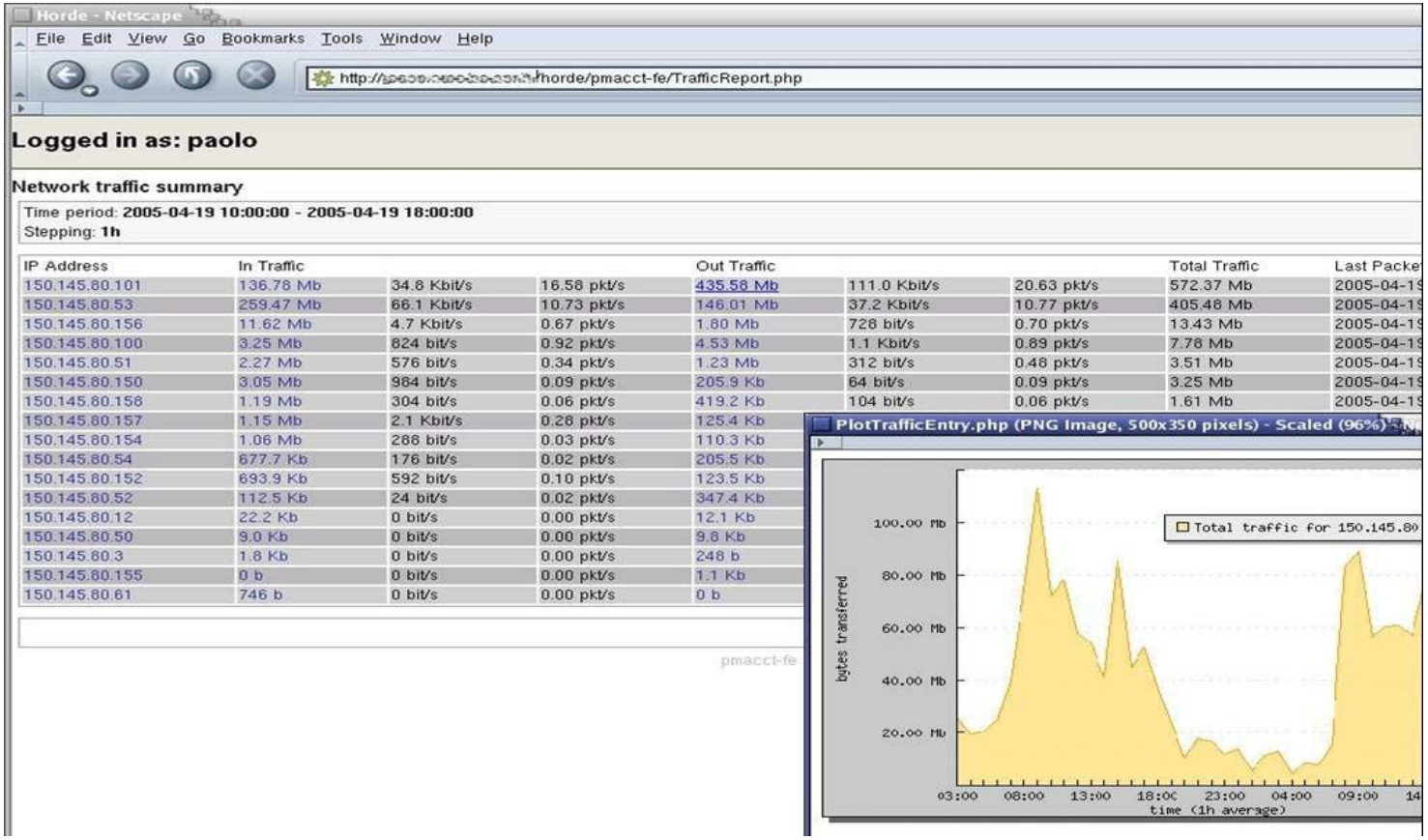
```
shell> psql -U pmacct -c "SELECT class_id, packets, bytes, flows \
      FROM acct_v5 \
      ORDER BY bytes DESC \
      LIMIT 10;"

    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)
```

pmacct: results (I)
by Martin Pot, from RRDtool gallery

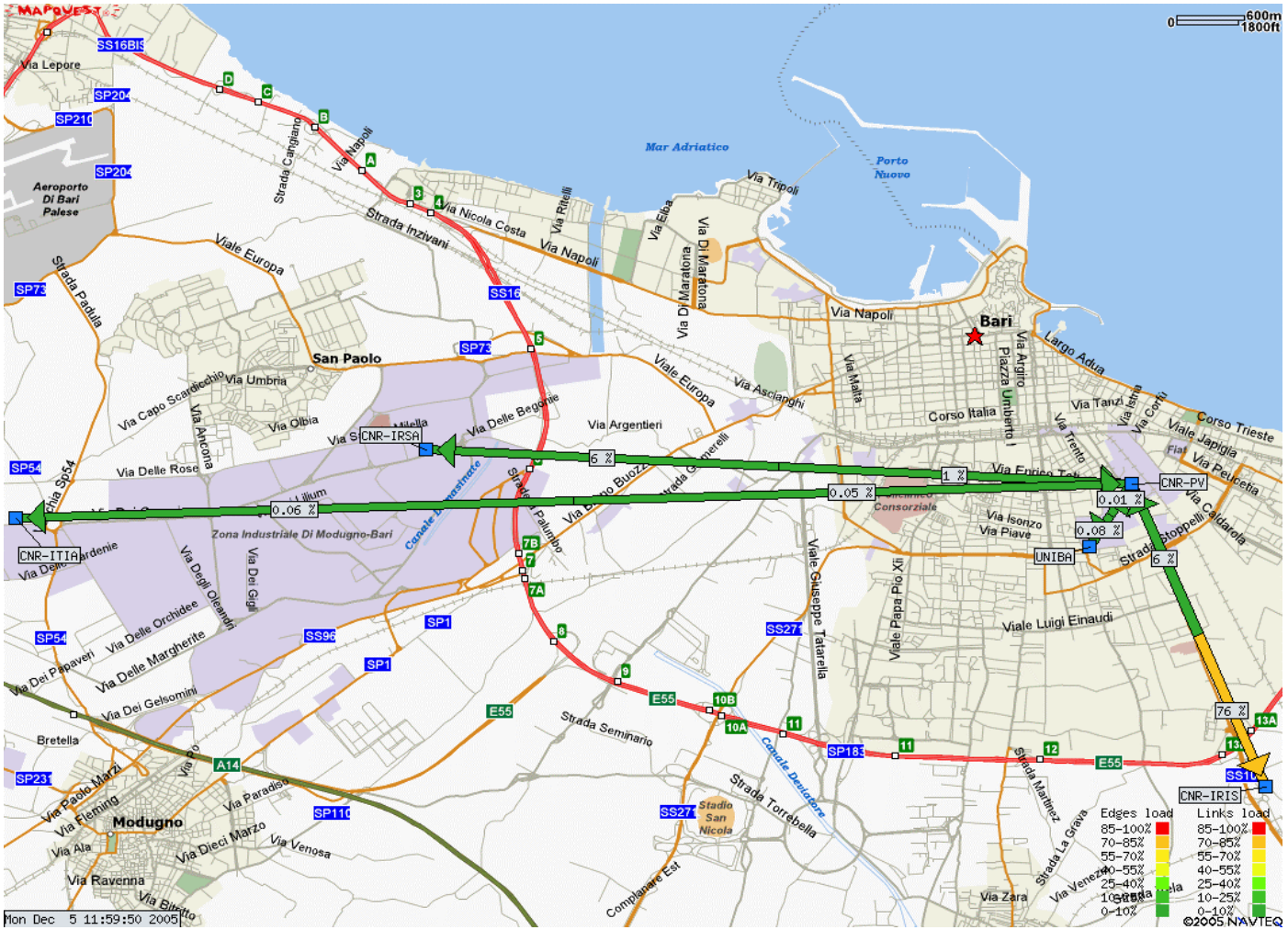


pmacct: results (II)
pmacct-fe screenshot (A)

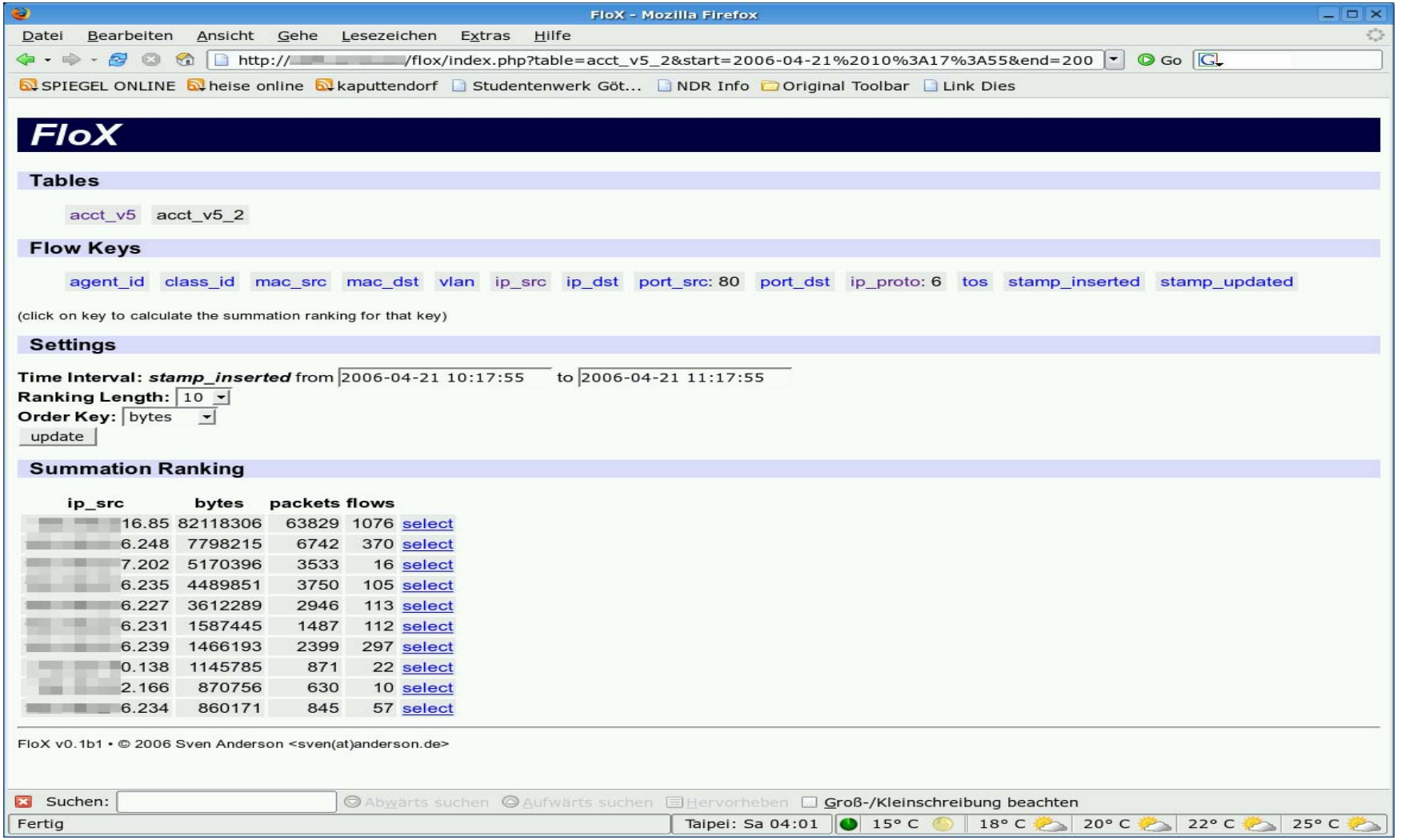
pmacct: results (II)
pmacct-fe screenshot (B)



pmacct: results (III)
network weather maps with GWEN



A preview of FloX, the flow explorer by Sven Anderson



Thank you for your attention !

http://www.pmacct.net

Paolo LUCENTE, paolo@pmacct.net