NetFlow to guard the infrastructure

Yann Berthier

NANOG39 - Feb 4-7, 07

Outline I

1. An introductory view of NetFlow

2. The flow cache
   - Overview
   - Populating the cache
   - Keeping the cache under control
   - Querying the cache in real time
   - Exporting the content of the cache

3. Collect & analysis
   - Overview
   - Storage
   - Tools
   - nfdump / NfSen
   - direction & biflows
   - From PCAP to NetFlow
   - NetFlow and BGP
   - Other sources

Outline II

- Reductio ab aggregatio

4 What's next

- Perhaps not really next
- Resources

Outline

1. An introductory view of NetFlow
2. The flow cache
3. Collect & analysis
4. What's next

What is NetFlow ?

- A technology to gather informations on forwarded packets
  - In one or several caches
  - Whose content can be queried real time for troubleshouting
  - And exported to *collectors*
- A protocol to export those records to collectors
  - To use sophisticated tools for analysis
  - Near real-time analysis
  - For long-term analysis, trending, whatever
  - To store and retrieve if needed (forensics) / when asked
- Comes from one vendor, but industry + IETF standard (not quite, but ...)
  - Other *concurrent* technologies: CRANE, Diameter, LFAP, IPDR, sFlow, ...

What is a flow ?

A loose definition

A set of packets having common characteristics

Definition

A flow is a unidirectional set of packets that arrive at the router on the same subinterface, have the same source and destination IP addresses, Layer 4 protocol, TCP/UDP source and destination ports, and the same ToS (type of service) byte in the IP headers

Anatomy of a flow (v5)

- 7-tupple *key* fields
  - saddr, daddr, sport, dport, L3 proto, ToS, input ifIndex
- Additional fields
  - Byte count, packet count, start time, end time, output ifIndex, TCP flags, next hop, src AS, dst AS

Some characteristics

- L3-L4
- Flows are *unidirectionals* (ingress / egress)
- Flow-cache comes before ACL lookup
- Comes at a cost (memory, CPU) for the router
- Not the perfect solution - but not a lot of other candidates either

Rationale

- Old techno (96)
- A lot happening in the protocol and implementation front
- A lot of interest and projects evolving around NetFlow
- Questions keep being raised (How do i monitor ... What tool do you use for capture ... I want to have an overview of the traffic ...) where NetFlow is the answer - or at least part of it
- Infrastructure is a potential target
- What's needed is actual insight

NetFlow flavors

- Several versions of the export protocol
- Metering and export keep being worked on
- Depends on vendor, hardware, software, and combinations of those
- Even when and where supported, particularities to keep in mind

Versions

- v5 - De facto standard (supported by non-C vendors implementing NetFlow)
- v9 - template-based
  - Flexibility (for now means that the user has to be very flexible) - complexity
  - May have some compelling features - SCTP, IPv6, some L2 fields, additionnal L3 (ttl, ipid), BGP next-hop, ...
    - But probably not processed by your collector of choice anyway
  - (Fast) moving target
    - Some features are backported to v5 (e.g. transport independency)
- v10 aka IPFIX - IETF-blessed standard (in its way to, at least)
  - Based on NetFlow v9 ("forked" from)
- Other - v1 (obsolete), v7 (switches), v8 (router-based aggregation)

Hardware peculiarities

- Support is not equal on all devices
    - No support on the switches below the cat 45xx
- Peculiarities where supported
    - On L3 switches, no TCP Flags, many specificities depending upon type of sup engine, PFC, config, software, ...

Hardware peculiarities - on routers too

Big CAUTION on every Cisco doco concerning NetFlow on the 12k

Entering this command on a Cisco 12000 Series Internet Router causes packet forwarding to stop for a few seconds while NetFlow reloads the route processor and line card CEF tables. To avoid interruption of service to a live network, apply this command during a change window, or include it in the startup-config file to be executed during a router reboot

Less advertised caution

Despite ASIC support in Engine 2, 3 and 4+ Linecards 'Full NetFlow' still inflicts a heavy burden on memory and therefore sampled netFlow is preferred

Outline

1 An introductory view of NetFlow

2 The flow cache

- Overview
- Populating the cache
- Keeping the cache under control
- Querying the cache in real time
- Exporting the content of the cache

3 Collect & analysis

4 What's next

Principle

- A global memory cache, enabled for each interface
- number of entries - where it is kept
- For each packet entering an interface
    - Either an entry matching the key fields (saddr, daddr, sport, dport, L3 proto, ToS, input ifIndex) exists

  Byte and packet counts are summed, end time is updated, TCP flags are ORed
    - Or a new entry is created
- Expiration mechanism
- Export mechanism if so configured

Expiration mechanisms

When one of the following conditions is met

- Upon reception of a TCP *RST* or *FIN* packet
- After 15 seconds (default) for inactive flows
- After 30 minutes (default) for an active flow
- When the cache is full

Cisco, v5

Example

```
router(config)#ip flow-export version 5 [origin-as|peer-as]
router(config-if)#ip flow {ingress | egress}
```

v9 specifics

Example

```
#ip flow-export version 9 [origin-as|peer-as] bgp-nexthop
#ip flow-capture ip-id
#ip flow-capture mac-addresses
#ip flow-capture packet-length
#ip flow-capture ttl
#ip flow-capture vlan-id
```

NDE - NetFlow Data Export

- NDE on the MSFC exports statistics for flows routed in software
- NDE on the PFC exports statistics for flows routed in hardware
- Flow masks on the PFC - from *source-only* to *full-interface*

Example

```
Switch(config)#mls netflow
Switch(config)#mls flow ip full
! To report L2 bridged traffic - not all PFCs
Switch(config)#ip flow export layer2-switched
Switch(config)#mls nde sender version 5
Switch(config-if)#ip flow {ingress | egress}
Switch(config)#ip flow-export destination 192.0.2.1 2055
```

cflowd on Juniper

- Relies on Service Cards / Adaptive Service Interfaces
- Configuration of a sampling rule in the firewall statement
- Rate of said sampling is configured in the forwarding-options
  - Can be 1
- Input filter calling the firewall rule as defined above for each interface

cflowd on Juniper

Example

```
interfaces {
   sp-0/0/0 unit 0 family inet;
   fe-0/0/0 unit 0 family inet {
      filter input catch_all;
      address 10.88.17.126/28; }}
firewall {
   family inet filter catch_all term default then {
      sample; accept; }}
forwarding-options {
   sampling {
      input family inet {
         rate 1; max-packets-per-second 5000; }
      output { }}
```

FreeBSD

In-kernel ng_netflow(4) Netgraph node coupled with the ng_ipfw(4) ipfw Netgraph hook

Example

```
# ipfw add ngtee 10 ip from any to any
# ngctl mkpeer ipfw: netflow 10 iface0
# ngctl name ipfw:10 catchall
# ngctl msg catchall: setdlt { iface=0 dlt=12 }
# ngctl msg countall: settimeouts { inactive=3 active=300 }
```

FreeBSD cont'd.

Makes it easy to define flow export policies

Example

```
# ipfw add ngtee 5 tcp from any to me 22 in
# ngctl mkpeer ipfw: netflow 5 iface0
# ngctl name ipfw:5 ssh_in
# ngctl msg ssh_in: setdlt { iface=0 dlt=12 }
# ngctl msg ssh_in: settimeouts { inactive=3 active=300 }
```

- OOooohhHH
- Bad news is - AS fields are not populated - AAaahh...

- Tuning its size (32 to 256k depending on hw)
- Aging of entries
- Aggregation router side
- Sampling
- Input filtering
- Flow masks on Cat6.5k

Some performance considerations

- 64 bytes / entry in a cache
- Aggregation is more costly CPU-wise
- Done in hw for some platforms (Cat6.5k) - only export consumes CPU cycles
- Not fitted for more-than-reasonably busy routers
- Plan / test / monitor / prepare to rollback

Sampling or not

Depending on use / sf / hw

- Flow vs packet vs time - deterministic vs random
- Choice is not yours - check with your rep
- Effective (random sampled netflow) and scalable
- Granularity loss may not be an acceptable tradeoff for security
- Performance loss may not be an acceptable tradeoff for operations :D

Cisco

#show ip cache flow

```
SrcIf SrcIPaddress   DstIf DstIPaddress    Pr SrcP DstP Pkts
Vl90  10.11.10.50   Local  192.168.11.2    06 090D 0016  6
Vl97  192.168.19.214 Null  192.168.19.255  11 008A 008A  1
Vl16  192.168.17.165 Null  255.255.255.255 11 4001 006F  1
Vl10  192.168.15.65  Null  192.168.14.1    11 007B 007B  1
Vl10  192.168.15.166 Null  192.168.15.255  11 008A 008A  1
Vl10  192.168.15.84  Null  192.168.15.255  11 008A 008A  1
Vl10  192.168.15.75  Null  192.168.15.255  11 008A 008A  1
```

The top-talkers cache

- Possibility to create a dedicated cache for top-talkers
- Matching criteria, sorting order and number of entries are fixed at configuration
- Not the same as the *Dynamic Top Talkers* feature on the main cache

Example

```
router(config)#ip flow-top-talkers
router(config-flow-top-talkers)#top 30
router(config-flow-top-talkers)#sort-by bytes
```

The top-talkers cache, cont'd.

```
#show ip flow top-talkers
SrcIf   SrcIPaddress DstIf    DstIPaddress Pr SrcP DstP Bytes
Se1/6:0 10.18.16.77 Fa0/1    10.11.19.88  06 05FD 0455   34K
Se1/6:0 10.18.16.77 Fa0/1    10.11.19.65  06 05FD 045D   10K
Se1/6:0 10.18.16.77 Fa0/1    10.11.14.51  06 05FD 04BA  9508
Se1/6:0 10.18.16.77 Fa0/1    192.168.9.7  06 05FD 059E  7064
Fa0/1   10.11.19.88 Se1/6:0 10.18.16.77  06 0455 05FD  5380
```

The JunOS way

```
#show services accounting flow-detail | match bgp | trim 26
AAA.88.137.113 2146 Unknown BBB.88.137.126 bgp(179) 628 135136
00:30:00 585 95087
```

FreeBSD

#flowctl catchall show

```
SrcIf    SrcIPaddress DstIf DstIPaddress Pr SrcP DstP  Pkts
em0      10.19.11.2   lo0   10.18.5.89   1  0000 0000   51
(null)   10.18.5.89   em0   10.19.11.2   1  0000 0000   51
em0      10.21.5.19   lo0   10.18.5.89   1  0000 0000    8
(null)   10.18.5.89   em0   10.64.60.7   6  1466 9f8a    1
```

And the usual Unix tools | sort +*n* | head -n 10

Export mechanism

When expired, flows are packed together and sent to one or more collectors

- UDP based
- A header (24 bytes)
  - Version, number of *PDUs*, sequence number, ...
- 1-30 flow records (48 bytes each)
  - 1464-bytes packets
- Loss detection
  - SEQ number
- No provision for retransmission

On Cisco

Example

```
router(config)#ip flow-export destination 192.0.2.1 2055
router(config)#ip flow-export source loopback0
```

Exporting the content of the cache, Juniper

Example

```
forwarding-options {
   sampling {
      output {
 cflowd 192.168.25.1 {
    port 2055; version 5; autonomous-system-type origin; }
 interface sp-0/0/0 { source-address 192.168.25.2; };
```

On FreeBSD

Example

```
# ngctl mkpeer catchall: ksocket export inet/dgram/udp
# ngctl msg catchall:export connect inet/192.0.2.1:2055
```

Others

Soft meters / exporters

- PCAP to NetFlow: fprobe, softflowd, nProbe, ...
- Other sources: fw logs (PF, netfilter)

Improving the reliability of export

- Multiple (2) destinations where supported
- Multicast-aware collectors
  - Export destination can be a multicast address
  - Collectors join the multicast group
  - Out-of-the-box redundancy, scalability, easy testing of other collectors, collector diversity, ...
- Consider using other transport protocols - SCTP with backup ?
  - The way of the future as they say, IPFIX implementations *MUST* support SCTP
  - Though for now the list of collectors supporting SCTP is ... small (read: I don't know one, except NTOP being not a collector either)

Security considerations

- Unidirectional from *router to collector*
- UDP
- No crypto checksum
- No authentication of the exporter
- 48 bytes-long records for min 28-bytes UDP packets

You may want to protect the link between exporter and collector from a non invited participant

- ACLs, uRPF, TCP-Wrappers or alike

Outline

1 An introductory view of NetFlow

2 The flow cache

3 Collect & analysis

- Overview
- Storage
- Tools
- nfdump / NfSen
- direction & biflows
- From PCAP to NetFlow
- NetFlow and BGP
- Other sources
- Reductio ab aggregatio

4 What's next

Glories and duties of a collector

- Takes flows as input
  - May report loss
- Pre-processes
  - Deduplicates, long flows, filters, various aggregation schemes
- Stores
- Post-processes
  - Deduplicates, long flows, filters, various aggregation schemes
- Retrieval
  - Sorts, filters, aggregates,
- Graphs

Glories and duties of a collector

- Per record granularity
- Aggregation facilities and higher level views (while retaining granularity)
- Versatility
- Powerful CLI
- Graphs
- Can be splitted over different tools

Glories and duties of the analyst

- Look at graphs
- Look at TopNs
- Build filters to look at subsets of traffic
    - Control & management plane
    - Traffic targetting adresses in infrastructure subnets
    - Outgoing traffic from said addresses

More considerations

- IP-centric
  - Changing with v9: some L2, MPLS, ...
  - Not any-over-any friendly
- TCP flags are blended together
  - 'Isolated' flags are still worth looking at:
  - nfdump -R . -o long 'flags S and not flags ARPFU'
  - nfdump -R . -o long 'flags R and not flags SPFU'
- L7 streams are splitted upon 2+ records
- Source and destination does not match sockets' notion of client/server

Lost in collection

- Some figures: 3 STM1s, 2 borders, 1 (/1) sampling: 7 GB/day
- Mid term strategy: compression
- Long term: prune data and keep heavily aggregated summaries (topNs, ...)

Flat file vs database

- Db solution sounds sexy but is costly - per-record overhead
- B plan - keep granularity in "efficient" storages, and send aggregated and associated data in a db
- Probably no integration with used tools
- No normalized on-disk format

An overview of popular collectors

A lot in both the opensource and commercial world - varying a lot in scope and qualities (scoop)

- Arbor Peakflow SP, Lancope, Cisco MARS, Q1labs' QRadar, many more
- The venerable flow-tools, the yet more venerable CAIDA's cflowd, argus, SiLK, Stager, nfdump/NfSen, many more
- Others
  - aguri, glflow, panoptis, NTOP, stager, ...

Flow-tools / Flowscan

- Second historical collection-and-analysis package
- Many patches / associated scripts / ... / floating around
  - flow-tool to db, Flow Extract, ...
- Big user community
- Somewhat baroque (my opinion) CLI
- GUI - flowscan / cuflow / jkflow
- Not actively maintained (latest rel 0505) - no v9 support

nfdump/NfSen

- Clean modular design
- Powerful and efficient (easy) yet simple (way harder) CLI with PCAP-like syntax
- tightly coupled GUI with nice graphs
- Takes input from NetFlow v5, v7, v9 (not all fields though) and sFlow
- Actively maintained
- Some plugins / patches floating around (porttracker, nfsplit, Holt-Winter)
- Community is growing steadily
- Versatile - troubleshooting, perf, security, ... - real time, trending, long-term analyses & forensics
- Comes with a flow-tools 2 nfdump converter
  - All your flows are belong to us

And the winner is ...

Architecture

nfcapd / sfcapd

- NetFlow / sFlow capture daemon
- No pre-post processing of the records collected
- Reads data from the network and writes to disk
    - Fixed-time binary files (default 5 mn)
- May demux to other collector

nfdump

- Reads files and displays records after run-time processing

Architecture

Others - self-explicit

- nfreplay, nfexpire, nfprofile, ft2nfdump

NfSen

- Graphs based on rrdtools
- Front-end to nfdump queries
- Plugins

Installation

Straightforward

- http://sourceforge.net/projects/{nfdump,nfsen}
- Stable branch vs snapshots
  - En-route to NfSen 1.3 - major usability improvements
  - Whatever the version, keep nfdump and nfsen synchronized
- Upgrade path provided between releases/snapshots
- Nfdump dependencies: c compiler
- Nfsen dependencies: perl, php, rrdtools, php-extensions (SESSION, SOCKETS)
- Bumping of max. number of SVIPC semaphores may be needed (one per collector)

nfdump

- Reads nfcapd files
- Processes
  - Parses pcap-like filters (usuals + if, packets, bytes, pps, bps, bpp, as, duration)
  - Matches long flows
  - Aggregates (proto, prefix/length, ASN, port)
  - TopNs (record, ip, port, tos, as, if, proto)
  - Orders by flow, packet, bytes, pps, bps, bpp
  - Anonymizes
- Displays
  - Pre-defined formats / user-defined formats
  - Statistics (number of flows, per proto, packets, bytes, ...)

nfdump output

- Pre-defined formats: `-o "line|long|extended|pipe|raw|fmt"`
- `-o "fmt:%ts %td %sap %dap"`

-r nfcapd.200701202315 -c 3 -o "fmt:%td %sap %dap"

```
Duration       Src IP Addr:Port           Dst IP Addr:Port
   0.000    BBB.158.102.14:53          AAA.189.5.89:53
  11.000    CCC.249.66.142:52812       AAA.189.5.89:80
   1.000    DDD.218.231.90:4509        AAA.189.5.90:445
```

AS Matrix

```
nfdump -M  <source_list>  \
-R nfcapd.$start_tslot:nfcapd.$end_tslot \
-s record/bytes -A srcas,dstas -n 0 \
-o "fmt:%sas %das %byt"
```

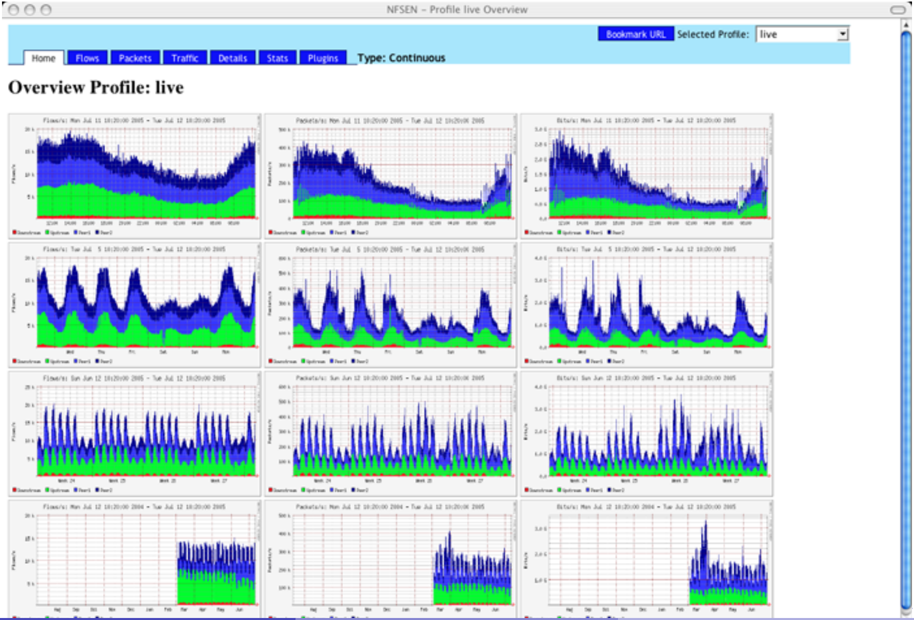
If user-defined is not enough - nfdump.c

```
Define your output format string.
Test the format using standard syntax -o "fmt:<your format>"
Create a #define statement for your output format,
similar than the standard output formats above.
Add another line into the printmap[] struct below
BEFORE the last NULL line for you format:
   { "formatname", format_special, FORMAT_definition, NULL },
  The first parameter is the name of your format as
  recognized on the command line as -o <formatname>
  The second parameter is always 'format_special' -
  the printing function.
  The third parameter is your format definition as defined
  in #define.
  The forth parameter is always NULL for user defined formats.
Recompile nfdump
```

$CONFDIR/nfsen.conf

- Various paths ($BASEDIR, $BINDIR, $LIBEXECDIR, $CONFDIR, $HTMLDIR, $VARDIR, $PROFILESTATDIR, $PROFILEDATADIR, $BACKEND_PLUGINDIR, $FRONTEND_PLUGINDIR, $PREFIX
- User of the {s,n}fcapd processes
- Data layout (see below)
- Disk filling % (warning message)
- Disk filling water marks for profiles
- Sources
    - $BINDIR/nfsen reconfig to add more

Main tab



NFSEN – Profile live Overview

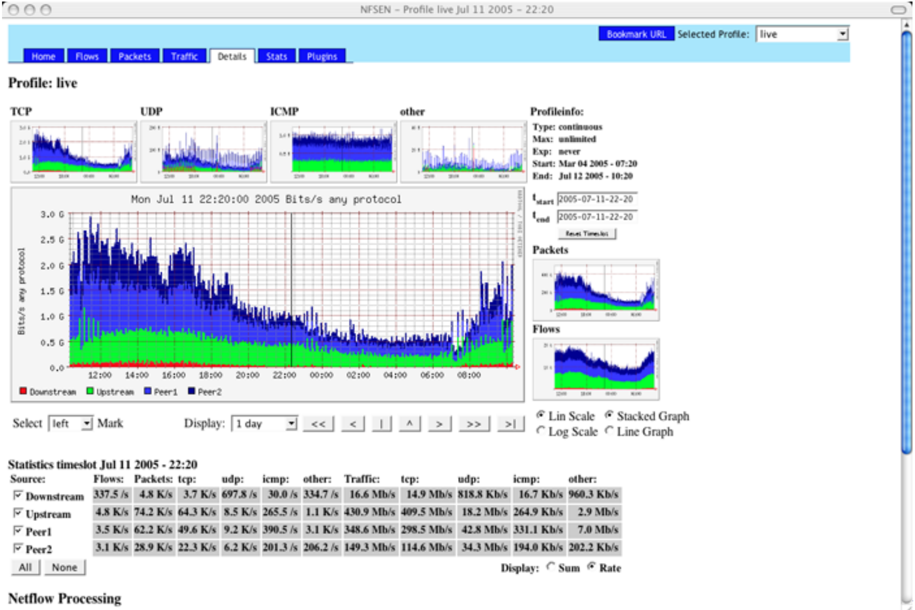
Bookmark URL
Selected Profile: live
Home
Flows
Packets
Traffic
Details
Stats
Plugins

Type: Continuous

Overview Profile: live

Flows/s: Mon Jul 11 10:20:00 2005 - Tue Jul 12 10:20:00 2005
Packets/s: Mon Jul 11 10:20:00 2005 - Tue Jul 12 10:20:00 2005
Bits/s: Mon Jul 11 10:20:00 2005 - Tue Jul 12 10:20:00 2005
Flows/s: Tue Jul 5 10:20:00 2005 - Tue Jul 12 10:20:00 2005
Packets/s: Tue Jul 5 10:20:00 2005 - Tue Jul 12 10:20:00 2005
Bits/s: Tue Jul 5 10:20:00 2005 - Tue Jul 12 10:20:00 2005
Flows/s: Sun Jun 12 10:20:00 2005 - Tue Jul 12 10:20:00 2005
Packets/s: Sun Jun 12 10:20:00 2005 - Tue Jul 12 10:20:00 2005
Bits/s: Sun Jun 12 10:20:00 2005 - Tue Jul 12 10:20:00 2005
Flows/s: Mon Jul 12 10:20:00 2004 - Tue Jul 12 10:20:00 2005
Packets/s: Mon Jul 12 10:20:00 2004 - Tue Jul 12 10:20:00 2005
Bits/s: Mon Jul 12 10:20:00 2004 - Tue Jul 12 10:20:00 2005
Downstream
Upstream
Peer1
Peer2

Navigation tab



Selected Profile: live

Profile: live

Profileinfo:

Type: continuous
Max: unlimited
Exp: never
Start: Mar 04 2005 - 07:20
End: Jul 12 2005 - 10:20

t_{start} 2005-07-11-22-20
t_{end} 2005-07-11-22-20

Select: left Mark   Display: 1 day

● Lin Scale ● Stacked Graph
○ Log Scale ○ Line Graph

Statistics timeslot Jul 11 2005 - 22:20

Source:	Flows:	Packets:	tcp:	udp:	icmp:	other:	Traffic:	tcp:	udp:	icmp:	other:
☑ Downstream	337.5 /s	4.8 K/s	3.7 K/s	697.8 /s	30.0 /s	334.7 /s	16.6 Mb/s	14.9 Mb/s	818.8 Kb/s	16.7 Kb/s	960.3 Kb/s
☑ Upstream	4.8 K/s	74.2 K/s	64.3 K/s	8.5 K/s	265.5 /s	1.1 K/s	430.9 Mb/s	409.5 Mb/s	18.2 Mb/s	264.9 Kb/s	2.9 Mb/s
☑ Peer1	3.5 K/s	62.2 K/s	49.6 K/s	9.2 K/s	390.5 /s	3.1 K/s	348.6 Mb/s	298.5 Mb/s	42.8 Mb/s	331.1 Kb/s	7.0 Mb/s
☑ Peer2	3.1 K/s	28.9 K/s	22.3 K/s	6.2 K/s	201.3 /s	206.2 /s	149.3 Mb/s	114.6 Mb/s	34.3 Mb/s	194.0 Kb/s	202.2 Kb/s

Display: ○ Sum ● Rate

Netflow Processing

Data organization

- Sources - aka exporters
- nfsplit
  - External contrib
  - Stands between the actual flows and nfcapd
  - Splits per interface
- Data lives in `$BASEDIR/profiles/<profile>/<source>`

```
# 0 default     no hierachy levels - flat layout - compatible
# 1 %Y/%m/%d    year/month/day
# 2 %Y/%m/%d/%H year/month/day/hour
# 3 %Y/%W/%u    year/week_of_year/day_of_week
# 4 %Y/%W/%u/%H year/week_of_year/day_of_week/hour
# 5 %Y/%j       year/day-of-year
# 6 %Y/%j/%H    year/day-of-year/hour
# 7 %Y-%m-%d    year-month-day
# 8 %Y-%m-%d/%H year-month-day/hour
```

Data organization, cont'd

- Profiles - 'live' post-install
  - Managed through web *and* CLI
  - Data is duplicated between profiles (should change)
  - Can be retroactive
  - Time-delimited or continuous
  - Expiration of old data - based on duration and/on size
  - Beware - thresholds are per profile, no global check
  - Beware - it is recommended to put the stats ($PROFILESTATDIR) and the data ($PROFILEDATADIR) on different volumes to prevent corruption in case of a disk full
- Channels - defined by filters

Managing profiles

- List all profiles

```
$BINDIR/nfsen -A
```

- Add a profile

```
$BINDIR/nfsen -a <profile> -c desc -B <starts> -E <ends>
```

- Modify a profile

```
$BINDIR/nfsen -m <profile>
```

- Delete a profile

```
$BINDIR/nfsen -d <profile>
```

Managing profiles

```
/data/nfsen/bin/nfsen -a slammer -B 2006-10-12-23-45 -S
other_in:other_out ’proto udp and port 1434’
```

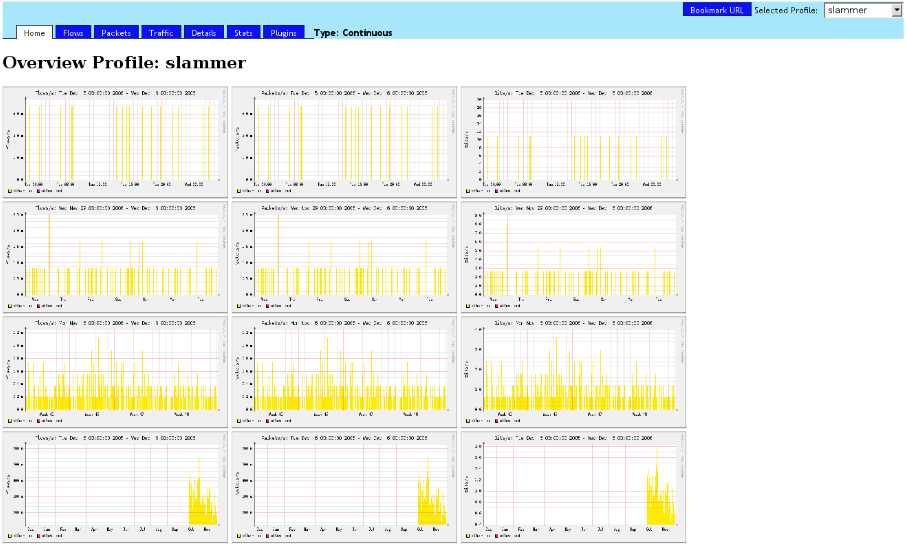
```
#
name    slammer
tstart  Thu Oct 12 23:45:00 2006
tend    Wed Dec  6 02:55:00 2006
updated Thu Oct 12 23:40:00 2006
filter  filter.txt
expire  0 hours
size    0
maxsize 0
sources other_in:other_out
type    continuous
locked  1
status  new
```

Managing profiles

/data/nfsen/bin/nfsen -l slammer

```
#
name    slammer
tstart  Thu Oct 12 23:45:00 2006
tend    Wed Dec  6 02:55:00 2006
updated Thu Oct 12 23:40:00 2006
filter  filter.txt
expire  0 hours
size    0
maxsize 0
sources other_in:other_out
type    continuous
locked  1
status  built 53.9%
```

Navigation tab



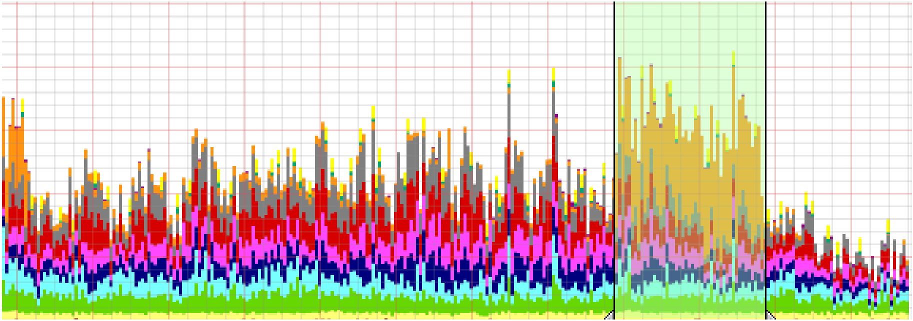
Bookmark URL
Selected Profile: slammer
Home
Flows
Packets
Traffic
Details
Stats
Plugins

Type: Continuous

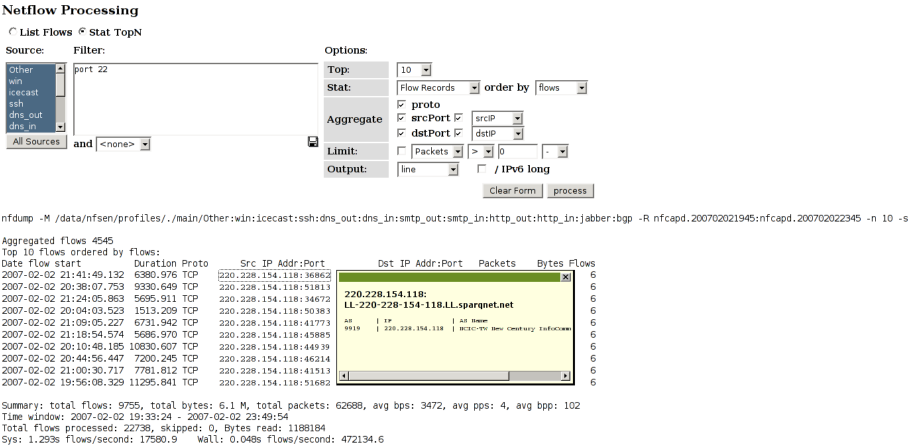
Overview Profile: slammer

Flows/s: Tue Dec 5 00:00:00 2006 - Wed Dec 6 00:00:00 2006
Packets/s: Tue Dec 5 00:00:00 2006 - Wed Dec 6 00:00:00 2006
Bits/s: Tue Dec 5 00:00:00 2006 - Wed Dec 6 00:00:00 2006
Flows/s: Wed Nov 29 00:00:00 2006 - Wed Dec 6 00:00:00 2006
Packets/s: Wed Nov 29 00:00:00 2006 - Wed Dec 6 00:00:00 2006
Bits/s: Wed Nov 29 00:00:00 2006 - Wed Dec 6 00:00:00 2006
other in
other out

Channels



Detail tab



Netflow Processing

○ List Flows ● Stat TopN

Source: Other, win, icecast, ssh, dns_out, dns_in — All Sources

Filter: port 22

and <none>

Options:

Top:	10
Stat:	Flow Records **order by** flows
Aggregate	☑ proto ☑ srcPort ☑ srcIP ☑ dstPort ☑ dstIP
Limit:	☐ Packets > 0 -
Output:	line ☐ / IPv6 long

Clear Form process

```
nfdump -M /data/nfsen/profiles/./main/Other:win:icecast:ssh:dns_out:dns_in:smtp_out:smtp_in:http_out:http_in:jabber:bgp -R nfcapd.200702021945:nfcapd.200702022345 -n 10 -s

Aggregated flows 4545
Top 10 flows ordered by flows:
Date flow start          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Packets    Bytes Flows
2007-02-02 21:41:49.132  6380.976 TCP    220.228.154.118:36862                                                6
2007-02-02 20:38:07.753  9330.649 TCP    220.228.154.118:51813                                                6
2007-02-02 21:24:05.863  5695.911 TCP    220.228.154.118:34672                                                6
2007-02-02 20:04:03.523  1513.209 TCP    220.228.154.118:50383                                                6
2007-02-02 21:09:05.227  6731.942 TCP    220.228.154.118:41773                                                6
2007-02-02 21:18:54.574  5686.970 TCP    220.228.154.118:45885                                                6
2007-02-02 20:10:48.185 10830.607 TCP    220.228.154.118:44939                                                6
2007-02-02 20:44:56.447  7200.245 TCP    220.228.154.118:46214                                                6
2007-02-02 21:00:30.717  7781.812 TCP    220.228.154.118:41513                                                6
2007-02-02 19:56:08.329 11295.841 TCP    220.228.154.118:51682                                                6

Summary: total flows: 9755, total bytes: 6.1 M, total packets: 62688, avg bps: 3472, avg pps: 4, avg bpp: 102
Time window: 2007-02-02 19:33:24 - 2007-02-02 23:49:54
Total flows processed: 22738, skipped: 0, Bytes read: 1188184
Sys: 1.293s flows/second: 17580.9    Wall: 0.048s flows/second: 472134.6
```

220.228.154.118:
LL-220-228-154-118.LL.sparqnet.net

```
AS        | IP                  | AS Name
9919      | 220.228.154.118     | NCIC-TW New Century InfoComm
```

AS resolution

$BASEDIR/libexec/Lookup.pm

```
my $whois_socket = IO::Socket::INET->new(
        PeerAddr  => ’whois.cyberabuse.org’,
        PeerPort  => 43,
        Proto     => ’tcp’,
        timeout   => 10 );
```

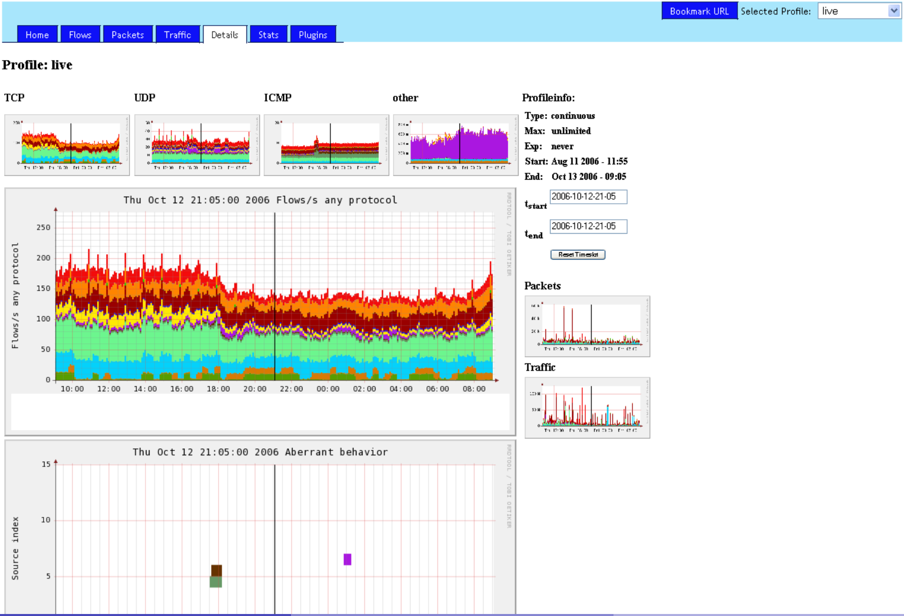
Plugins

- NfSen is "plugin" friendly
- Backend in perl and frontend in PHP
- Skeleton available in $BASEDIR/plugins/demoplugin.pm and $HTMLDIR/plugins/demoplugin.php
- No repository of plugins
  - Only publicly available plugin is PortTracker
  - Tracks ... TopN ports

Holt-Winter patches

- hw-patched rrdtools for NfSen
- External contrib
- Discussions pending regarding integration or not
- Can be installed in parallel to an existing NfSen installation
- Useful to spot traffic irregularities, especially among more verbose sources

Holt-Winter - main view



Bookmark URL | Selected Profile: live

Home | Flows | Packets | Traffic | Details | Stats | Plugins

Profile: live

TCP | UDP | ICMP | other

Profileinfo:

Type: continuous
Max: unlimited
Exp: never
Start: Aug 11 2006 - 11:55
End: Oct 13 2006 - 09:05

t_{start} 2006-10-12-21-05

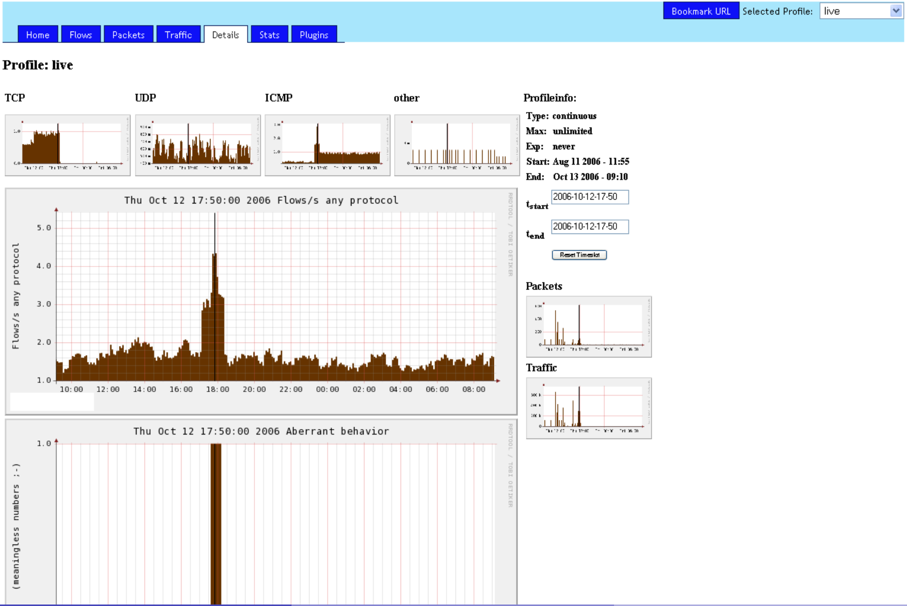
t_{end} 2006-10-12-21-05

Reset Timeslot

Packets

Traffic

Anomaly view



Profile: live

Profileinfo:
Type: continuous
Max: unlimited
Exp: never
Start: Aug 11 2006 - 11:55
End: Oct 13 2006 - 09:10

t_{start} 2006-10-12-17-50

t_{end} 2006-10-12-17-50

The example below is purely fictional

Example

```
+---------------------------------------+
| 1. The packet contains a specific     |
| crafted IP option.                    |
|---------------------------------------|
| AND                                   |
|---------------------------------------|
| 2. The packet is one of the following |
| protocols:                            |
|---------------------------------------|
|   * ICMP - Echo (Type 8) - 'ping'     |
|---------------------------------------|
|   * ICMP - Timestamp (Type 13)        |
|---------------------------------------|
|   * ICMP - Information Request (Type  |
|     15)                               |
```

Example

```
+---------------------------------------+
|   * ICMP - Address Mask Request (Type |
|     17)                               |
|---------------------------------------|
|   * PIMv2 - IP protocol 103           |
|---------------------------------------|
|   * PGM - IP protocol 113             |
|---------------------------------------|
|   * URD - TCP Port 465                |
|---------------------------------------|
| AND                                   |
|---------------------------------------|
| 3. The packet is sent to a physical   |
| or virtual IPv4 address configured on |
| the affected device.                  |
+---------------------------------------+
```

The ICMP case

- Source port == 0
- Code+type shoehorned into dst port - code is lower 8 bits, type higher 8 bits

tstamp request & tstamp reply

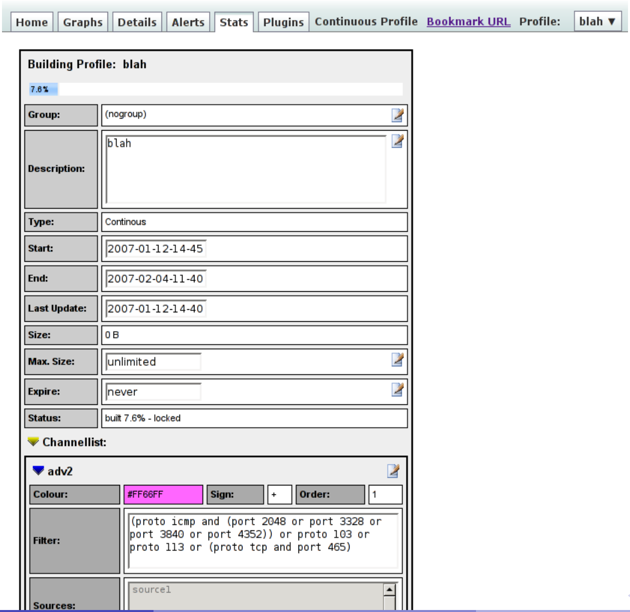
```
ICMP      192.168.2.11:0     ->     AAA.189.5.89:13.0
ICMP      AAA.189.5.89:0     ->     192.168.2.11:14.0
```

- No specific filters for ICMP though

Type 8, 13, 15, 17

proto icmp and (port 2048 or port 3328 or port 3840 or port 4352)

Adding a new profile and channel



Building Profile: blah

7.6%

Group:	(nogroup)
Description:	blah
Type:	Continous
Start:	2007-01-12-14-45
End:	2007-02-04-11-40
Last Update:	2007-01-12-14-40
Size:	0 B
Max. Size:	unlimited
Expire:	never
Status:	built 7.6% - locked

Channellist:

adv2

Colour:	#FF66FF	**Sign:**	+	**Order:**	1

Filter:

```
(proto icmp and (port 2048 or port 3328 or
port 3840 or port 4352)) or proto 103 or
proto 113 or (proto tcp and port 465)
```

Sources: source1

Analyst's temptation

- Looking at 'streams' rather than flows

Example

```
stime, client:sport, sflags, sbytes, spkts -> server:dport,
dflags, dbytes, dpkts, ...
```

- Makes it easier to track some irregularities / asymmetries
  - SYN / RST couples
  - Unusual proportions of s{bytes, pkts, ...} vs d{bytes, pkts, ...}
- Easier classification of L7 when knowing connect() and bind() sides
- Orthogonal problems

Direction

- Records convey no direction information
- No 3-way handshake granularity with TCP flags
- No sub-ms granularity for start times
- Besides, many reasons to have timestamps in the 'wrong' order
- No good heuristics beyond
  - Parse time-sorted
  - Swap saddr and daddr if

    `sport < 1025 && dport > 1024`
- Still too many incorrectly headed flows

Biflows

- Basically 3 places to aggregate 2 flows into 1 biflow
  - Router-side
  - Collect - one shot - increases complexity
  - post-process - run-time
- Some IPFIX drafts

Who'd want that ?

- Everybody trying to make sense of a (big, for some meaning of) packet capture
- In need of efficient tools
- Re-use of existing tools / methods

Argus

- `http://qosient.com/argus/`
- Collect and analysis of network data
  - PCAP (live or off-line)
  - NetFlow
- Stores flow-like data
  - Bidirectionnal records

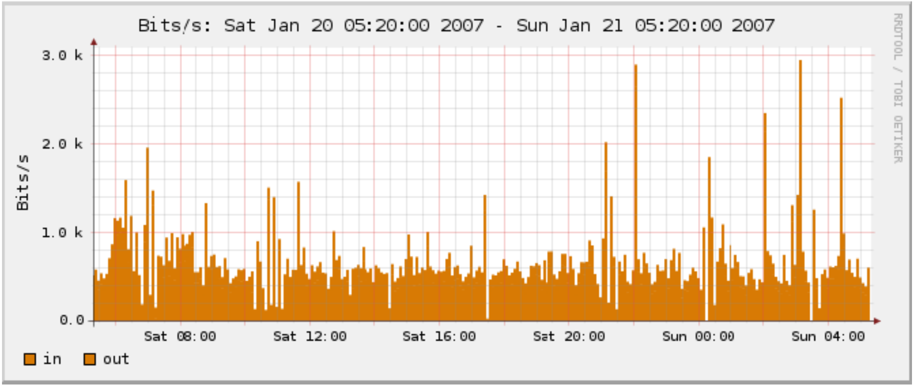
The other way around ? (from NetFlow to PCAP)

- Nobody would want that
- Lot of tools working on PCAP data where NetFlow records would be fine
  - Specialized reports, DoS, ...
- No normalized on-disk format - no incentive for tools' authors
- Disclaimer: very clumsy and inelegant
- `nfdump -> ASCII -> ipsumpdump -> PCAP` - OUCH
- `http://www.cs.ucla.edu/~kohler/ipsumdump/`

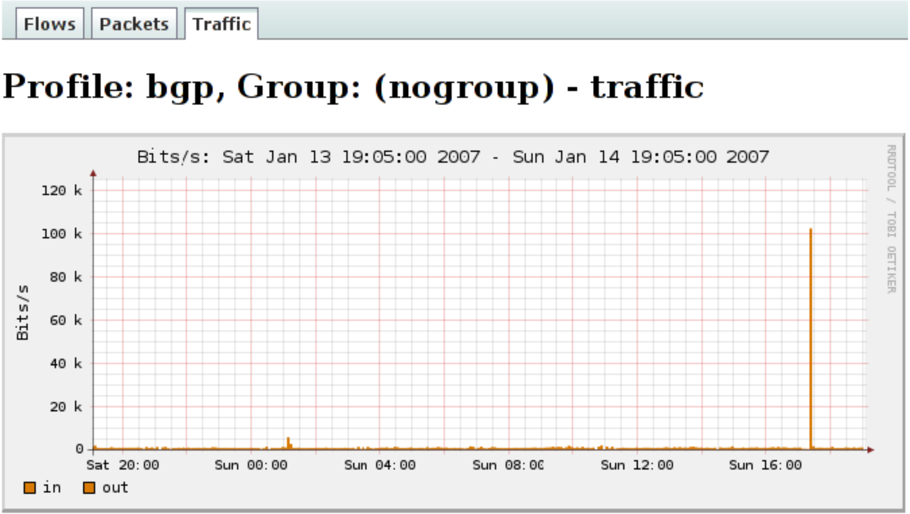
BGP

- Heartbeat-like traffic
    - Whose level of steroids in blood increases over time, but this is another story
- Pinpointing of BGP events (which scale ?)
- At least timeframe is provided - up to the ops to dig into BGP logs

BGP heartbeat

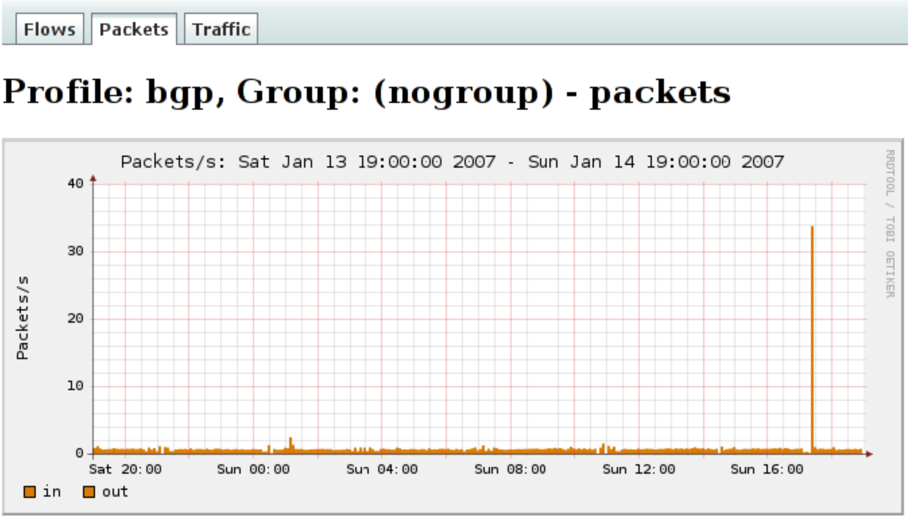


eBGP multihop instability



Profile: bgp, Group: (nogroup) - traffic

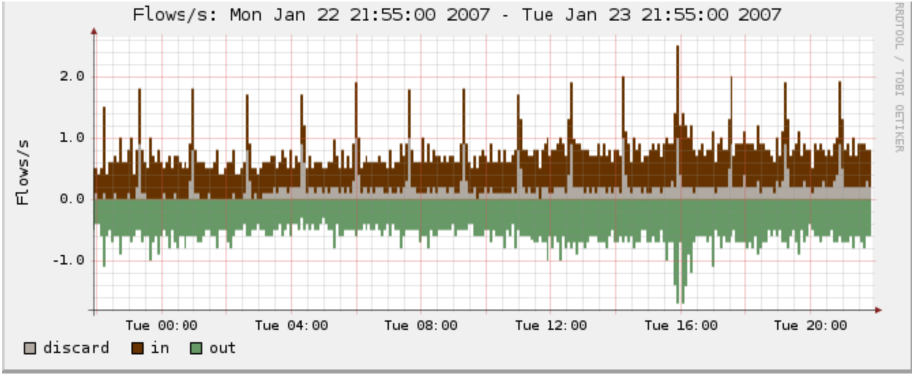
eBGP multihop instability

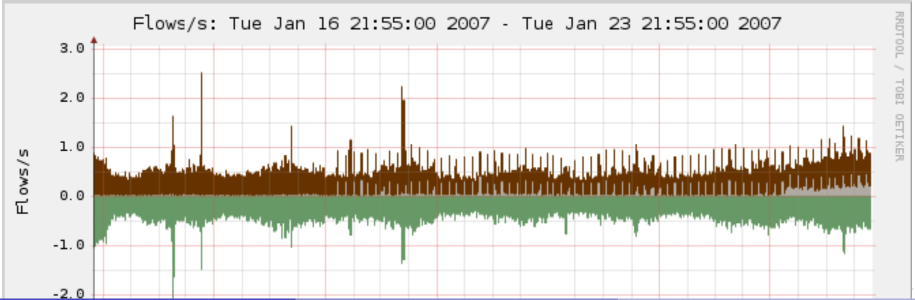


Profile: bgp, Group: (nogroup) - packets

Looking at the noise

- Sink Holes
- Non used address space
- Diverted traffic


Flows/s: Mon Jan 22 21:55:00 2007 - Tue Jan 23 21:55:00 2007
Flows/s
2.0
1.0
0.0
-1.0
Tue 00:00
Tue 04:00
Tue 08:00
Tue 12:00
Tue 16:00
Tue 20:00
discard
in
out
RRDTOOL / TOBI OETIKER


Flows/s: Tue Jan 16 21:55:00 2007 - Tue Jan 23 21:55:00 2007
Flows/s
3.0
2.0
1.0
0.0
-1.0
-2.0
RRDTOOL / TOBI OETIKER

Aguri

- http://www.csl.sony.co.jp/ kjc/software.html#aguri
- Automatic traffic aggregation - prefixes & ports
- PCAP-based

Aguri

aguri

```
[src address] 1049390337 (100.00%)
                AAA.64.60.71    1008893224 (96.14%)
                BBB.189.5.89    36286986 (3.46%)
[dst address] 1049390337 (100.00%)
                AAA.64.60.71    35493322 (3.38%)
                BBB.189.5.89    1012495103 (96.48%)
[ip:proto:srcport] 1049390337 (100.00%)
                4:6:22  1008772431 (96.13%)
                4:6:62008       35303106 (3.36%)
[ip:proto:dstport] 1049390337 (100.00%)
                4:6:22  35439904 (3.38%)
                4:6:62008       1008498420 (96.10%)
```

Outline

1. An introductory view of NetFlow
2. The flow cache
3. Collect & analysis
4. What's next
   - Perhaps not really next
   - Resources

NetFlow v9 and v10

- Disclaimer: I’m not affiliated with ...
- Disclaimer: check with your reps
- Templates ...
- Other transports
- User-defined caches and export policies
- Other vendors should follow

Dissemination of (net)flow specification rules

- Disclaimer
- draft-marques-idr-flow-spec-03
- "Successor" of uRPF + BGP null routes
- Former is a clever engineering trick to null route prefixes
- Latter formalizes the propagation of flow-like informations through BGP for further action
- Dst prefix, src prefix, proto, src & dst port, icmp type & code, TCP flags, packet length, ToS, fragment
- Basically a flow record

State of the flow-spec draft

- Implemented by one vendor
- Other is waiting for customer traction ?
- A NetFlow vendor from MI
- No (public) implementation on any Unix BGP daemon - too bad
  - Spot traffic to be acted upon from a BGP-speaking NetFlow analysis workstation
  - Use flow-spec support to inject and propagate said records

Some links

- http://www.cisco.com/go/netflow/
- http://www.cisco.com/en/US/products/ps6601/prod_presentation_list.html
- http://www.switch.ch/tf-tant/floma/software.html