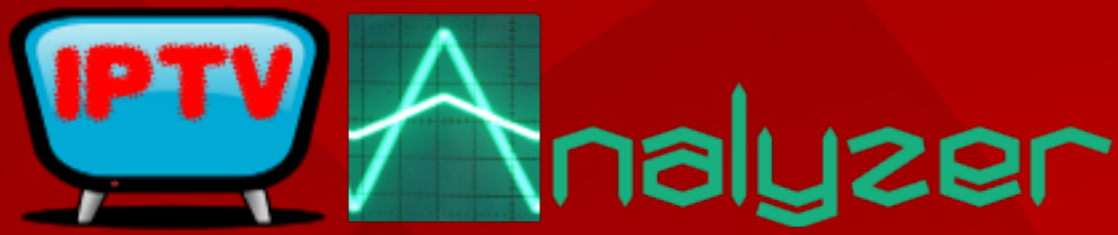
The IPTV-Analyzer

OpenSourceDays 2012

Jesper Dangaard Brouer
Senior Kernel Engineer, Red Hat
d.11/3-2012



Background / Disclaimer

- This is NOT a Red Hat product
  - Spare time hobby project
- Developed while at
  - ComX Networks A/S: Danish ISP
  - Release under GPLv2
- Troubleshooting tool
  - Developed while fixing
    - real-life IPTV quality issues

Who am I

- Name: Jesper Dangaard Brouer
  - Edu: Computer Science for Uni. Copenhagen
    - Focus on Network, Dist. sys and OS
  - Linux user since 1996, professional since 1998
    - Sysadm, Kernel Developer, Embedded
  - OpenSource projects, author of
      - ADSL-optimizer
      - CPAN IPTables::libiptc
      - IPTV-Analyzer
    - Patches accepted into
      - Linux kernel, iproute2, iptables, libpcap and Wireshark

What will you learn?

- This talk is about *network* challenges with IPTV
  - Part 1: The trouble I have seen
    - Measuring: You need eyes to see
  - Part 2: Saving the world
    - The IPTV-Analyzer
    - Future plans for an IPTV *shaper*
  - Help I need somebody
    - Need web-developer help**!**

Happily adding more channels

- What happened when more channels were added
  - High-Definition channels
  - More than 120 IPTV channels
  - Approx 600 Mbit/s multicast traffic.
- What happens to a realtime service
  - on a loaded Ethernet switch based network?

History: What was the problem?

- Customers were experiencing bad quality,
  - due to **packet drops**
  - (must drop on congested links, TCP/IP depend)
- Traffic had only reached 600Mbit/s
  - Drops were worse on loaded links.
  - But dedicated links also showed significant drops
    - 60 percent loaded link should not have drops

Finding cause of drops?

- Finding the cause of packet drops
  - Turned out to be a harder task than expected...
  - No clear drop pattern
  - ALL channels were seeing drops
  - Randomly across channels

What caused the drops?

- **Bursty traffic** from some of the IPTV streamers
  - Some of the "cheap" (120k DKK) streamers, bursting
  - Worst: Linux boxes with VLC and old 2.6.18 kernel
- The commercial streamers were harder to "fix"
  - Closed source
  - Slow and expensive support
    - Quote: “You need to buy the expensive model”
- Linux VLC machines easily fixed
  - by newer kernel with high resolution timer support

Hint slide: Enable Linux Highres

- Important to enable
  - high resolution timers and tickless OS
    - to avoid VLC multicast bursts
- Kernel config options
  - CONFIG_HIGH_RES_TIMERS=y
  - CONFIG_NO_HZ=y
- Watch the kernel log messages for:

```
“Switched to high resolution mode on CPU 0”
```

or

```
“Could not switch to high resolution mode on CPU 0”
```


History: Measure the problem

- First objective: Be able to measure the problem

You need eyes to see

  - Used "IP-probe" from BridgeTech, to detect/see drops.
    - Which is an IPTV analyzer appliance
      - internally runs Linux and has a special FPGA chip
      - Really nice tool, but very *expensive!*
    - Didn't show burstiness directly
      - Turned out to be important
      - Used Wireshark IO-graph

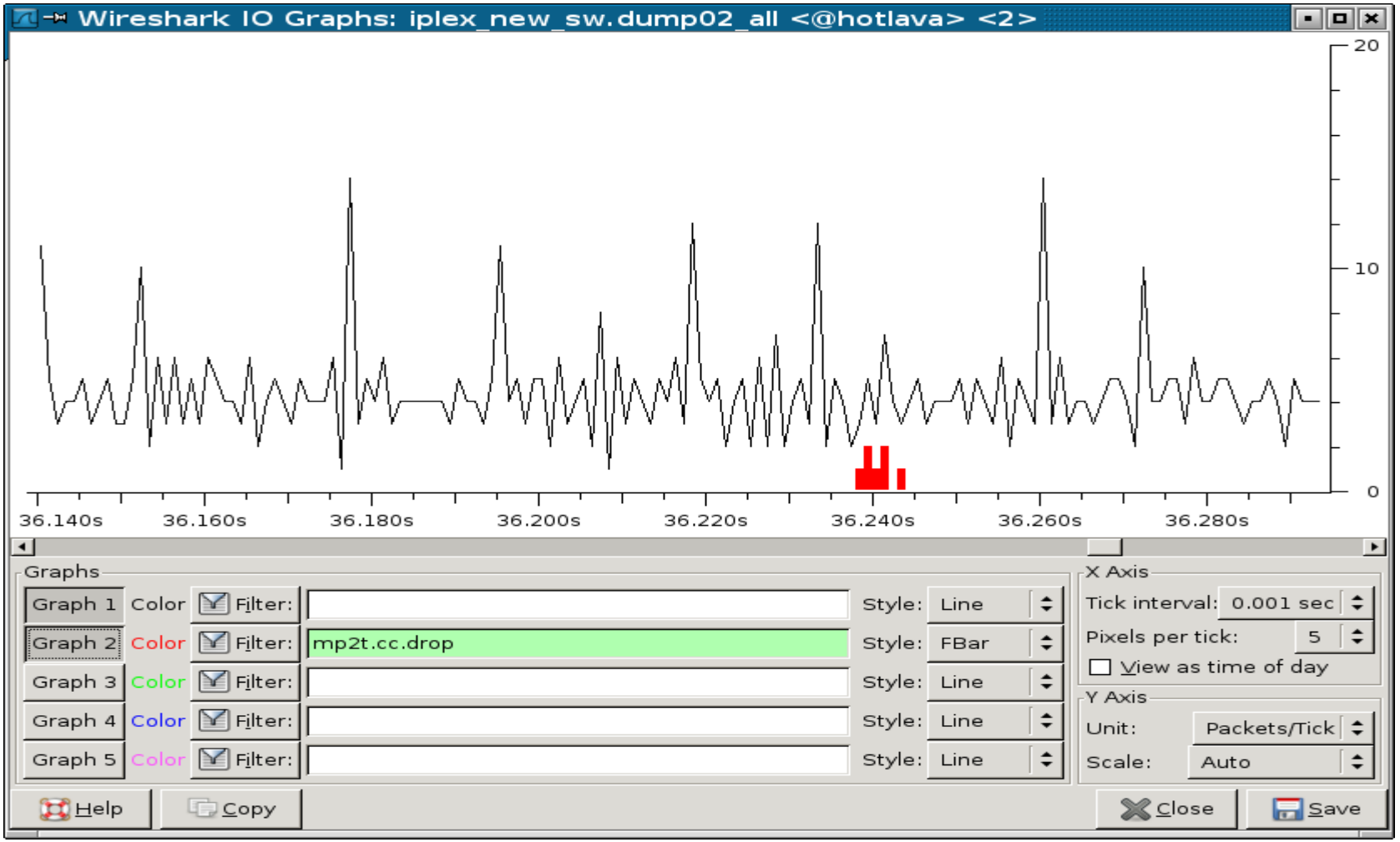
How did we locate the problem?

- BridgeTech probe tool, very helpful, showing the drops
  - but could **not show the burstiness** directly
- One of the Linux based streamers,
  - Alternating between 6 Mbit/s and 15 Mbit/s streams
  - **All** channels more drops during 15 Mbit/s streams
- Detailed analysis of traffic dumps
  - Wireshark was used to show burstiness
  - Wireshark IO-graph with resolution < 1 ms
  - Implemented drop detect my self in Wireshark

    (the power of open source)

Wireshark IO-graph

- Menu: Statistics → IO Graph → Tick Interval 0.001



Wireshark drop detection

- MPEG2 TS drop detection in Wireshark (mp2t)
  - Mainline v1.1.4 (svn r27381)
  - Improved in v1.3.4 (svn r30789)
    - Not fast enough for realtime/continuous usage
  - Dumping 600Mbit/s traffic
    - Standard tcpdump not fast enough
      - WARN cause “fake” packet drops (Func: “packet_rcv”)
    - Fix: Use tcpdump/pcap with Mem Mapped IO
      - Use **minimum libpcap version 1.0.0**
      - Function: “**t**packet_rcv” (See /proc/net/ptype)

Part 1: Summary

Watch out for bursts

- Packets bursts experiences
  - Surprisingly bursts do cause problems on 60% loaded links
  - Non-bursty streams are affected by bursty-streams
  - On Linux: Use Highres timers and no tick
  - Know you switch buffer sizes
    - don't place low-buffer switches in backbone
    - But **avoid bufferbloat**, by using AQM/QoS
    - - Create seperate queue for IPTV with larger buffer
    - - Create a smaller queue Internet traffic, allow drops

Part 2: Develop own IPTV-Analyzer

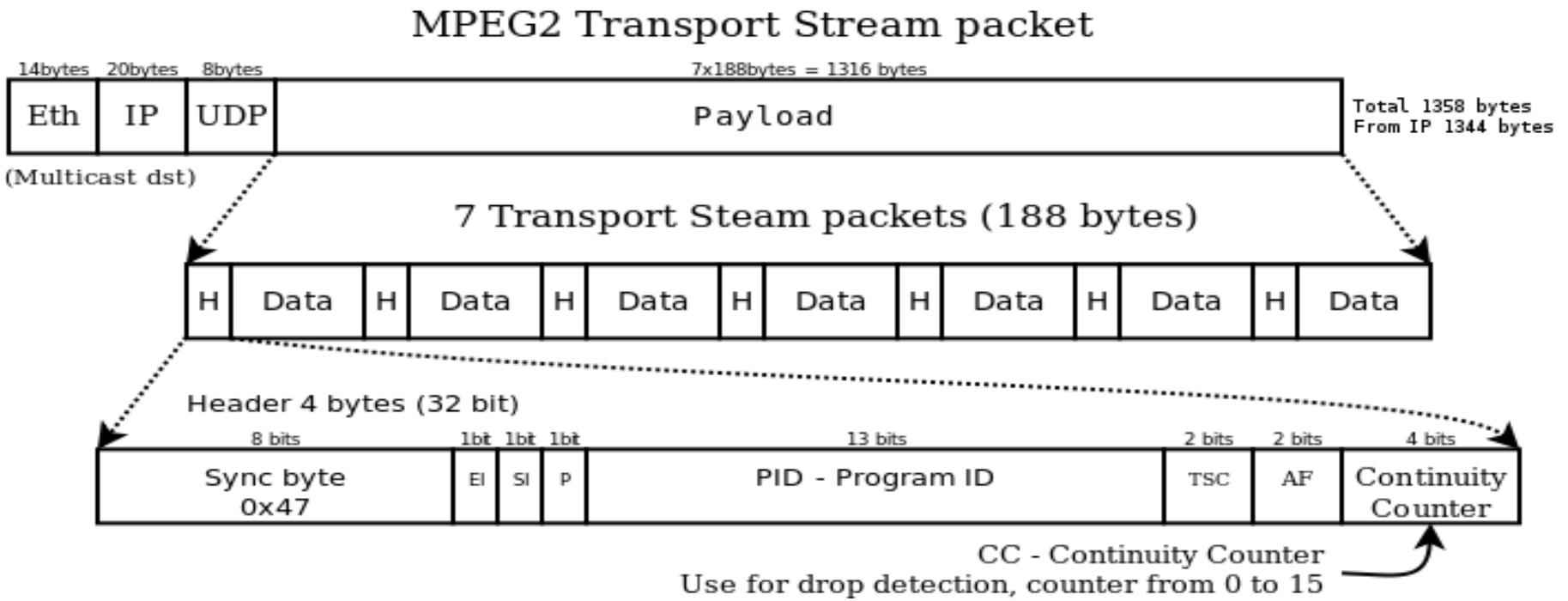
- Learned
  - Need to monitor the signal
  - Bursts can cause drops, on all channels
  - Bursts can occur per network segment
- Monitor each network segment
  - Approx measurement points: 12
  - Price per IPTV-probe approx 80.000 DKK
    - Total cost: 960.000 DKK
  - Develop our own IPTV-analyzer

Project plan: IPTV-Analyzer

Develop our own MPEG2-TS IPTV-Analyzer

- Milestone 1: Measure the drops – DONE
  - First objective: *measure* the problem
- Milestone 2: Measure the bursts
  - *Detect* potential issues, *before* drops occur
- Milestone 3: Smooth out the bursts
  - *Solve* the burst issue permanently

Deep Packet Inspection

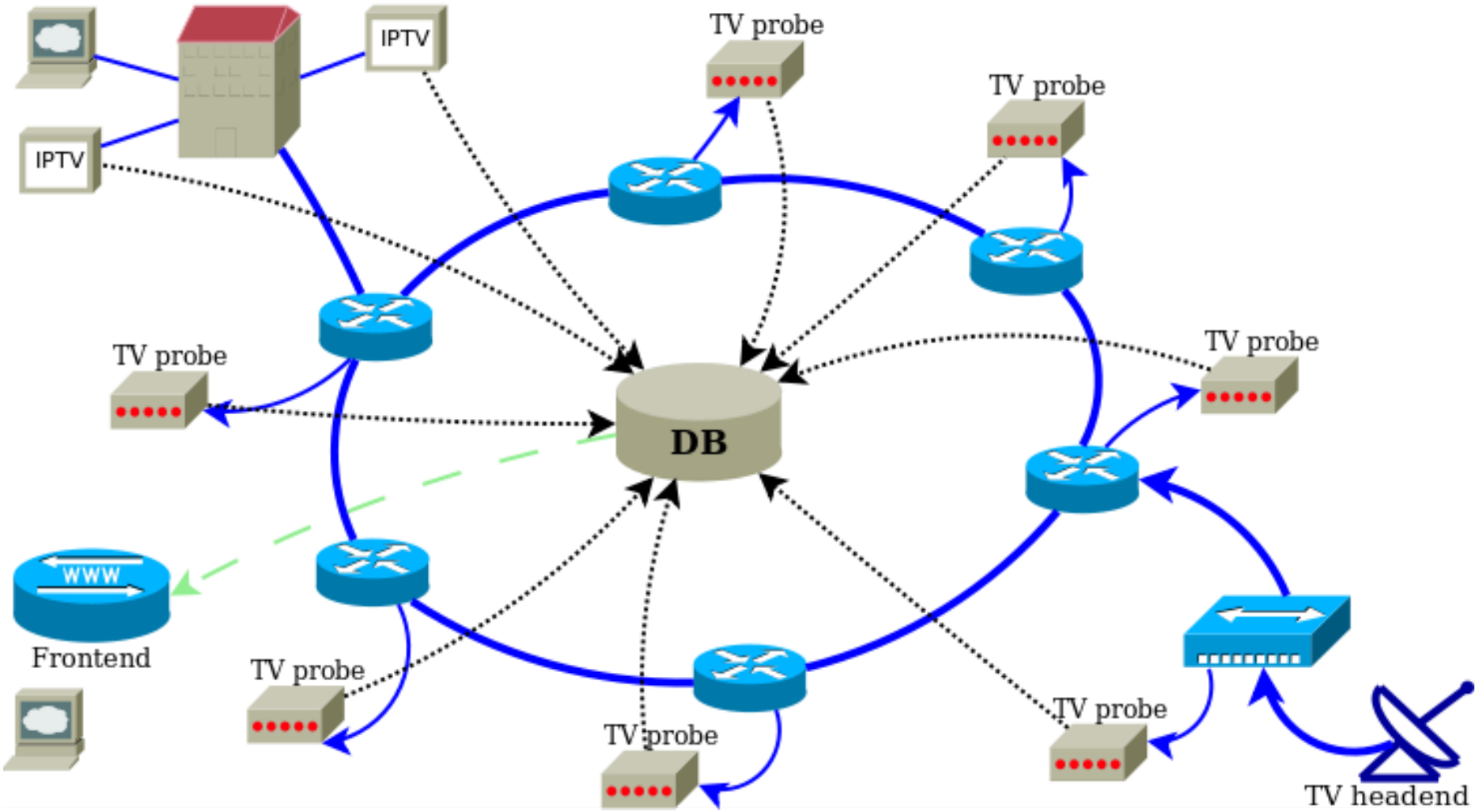


- Each IP-packet contains 7 Transport Stream Packets/Frames
- The Continuity Counter is very small values 0-15
  - Thus, an exact drop counter is hard as wrap around occurs quickly

Milestone 1: Measure drops

- Offline analysis
  - Contribute to Wirehark, mostly prototyping
- Need continuous monitoring (24x7x365)
  - Kernel module for detection (iptables)
  - Collector daemon (Perl)
  - Database design, storing events (MySQL)
  - Web based frontend (PHP)
  - Trick: use settop boxes as probes (bash)

IPTV Analyzer architecture



Kernel module: Efficient!

- Developed iptables/netfilter module (mpeg2ts)
  - tcpdump solution used 100% and caused false drops
  - For realtime/continuous drop monitoring
  - Only uses 2% CPU with 600Mbit/s traffic
    - on a low power ATOM 330 CPU (1.6Ghz)
- Implemented with RCU locking
  - for parallel processing on each CPU
    - (requires multi-queue NICs e.g. Intel 82576)
- Stats via /proc/ file

Collector

- Written as Perl daemon process
- Poll /proc every 10 sec
  - Compare with previous data → MySQL
  - Heartbeat (default 5 min) → MySQL
- SNMP-traps
  - Detect no-signal (in v0.9.2)
- Plan: Avoid polling
  - kernel report activity (via netlink)

Web-frontend: Probe overview

Probe overview

Please select a probe:

Choose a probe and adjust period

distance	probename	connected to switch	short location	location	description	id
1	tvprobe001a	albcs35	alb	alb-A Street8 Rack5	Main signal	2
2	tvprobe001a	albcr1	alb	alb-A Street4 Rack2	Main signal	1
2	tvprobe-dev2	albcr3	alb	ComX Albertslund Serverrum Z	Main signal	7
6	tvprobe004a	tgccs1	tgc	Tåstrup Global Connect	Signal before Arnakke	4
40	tvprobe003a	switch001	seas-nve	SEAS-NVE	Signal in SEAS/NVEs net	6
100	tvprobe002a	switch001	dansknet	Arnakke/DanskNet	Signal in Arnakke/DanskNet	3

From: 2010-02-22 20:00 2010-02-22 20:00:00

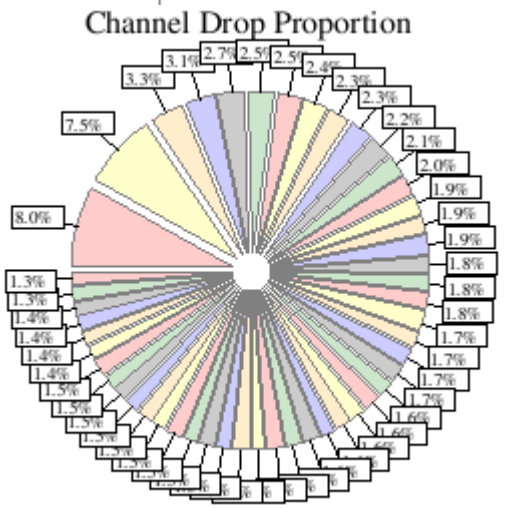
To: 2010-02-23 16:00 2010-02-23 16:00:00

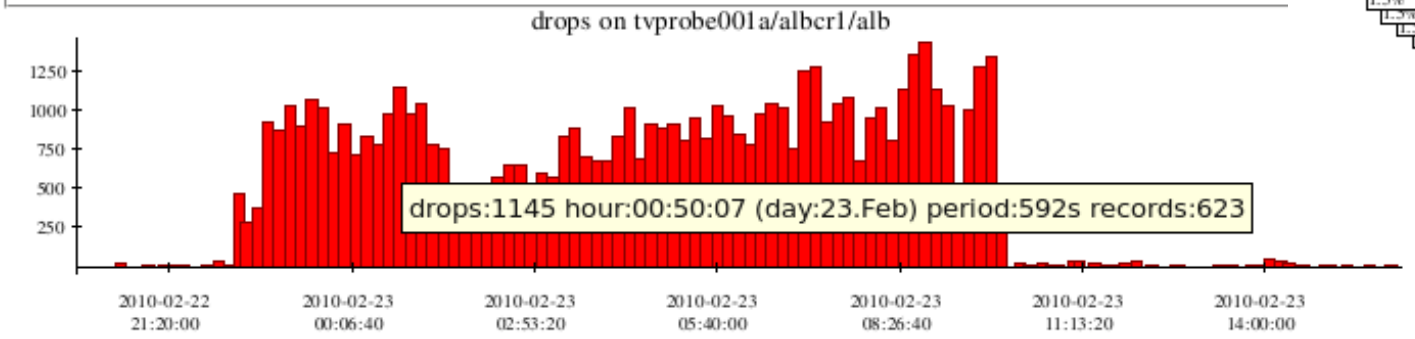
Period: 20 hours

Aggregation interval/period (bucket size) in sec: 600 (10 minutes)

Excessive level 5000 fix graph

Submit





Web-frontend: Channel view

Channel view

Multicast channel: 2[redacted].173.25

Choose a channel and adjust period

Choose a channel:

2[redacted].173.25 | Select

Selected probe: **tvprobe001a/albcr1/alb**

From: 2010-02-22 20:00:00 2010-02-22 20:00:00

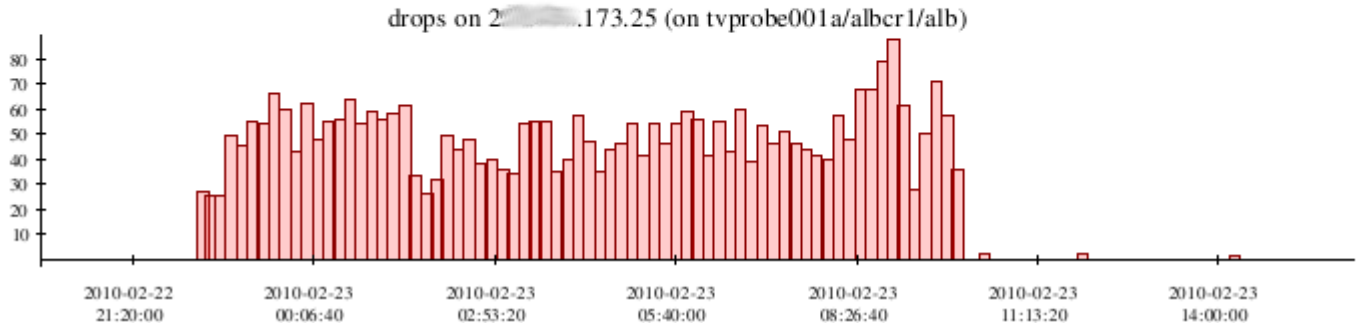
To: 2010-02-23 16:00:00 2010-02-23 16:00:00

Period: 20 hours (sec:72000)

Aggregation interval/period (bucket size) in sec: 600 (10 minutes)

Excessive level 5000 ☐ fix graph

sub-periods	probe (all)	drops	average sec between drops	measurement period	from	to	records
1	tvprobe001a/albcs35	5	0.00		2010-02-23 10:08	2010-02-23 10:08	1
1	tvprobe001a/albcr1	3509	16.25	15 hours, 50 minutes, 16 seconds	2010-02-22 22:24	2010-02-23 14:14	1790
1	tvprobe-dev2/albcr3	3509	16.25	15 hours, 50 minutes, 15 seconds	2010-02-22 22:24	2010-02-23 14:14	1803
1	tvprobe004a/tgccs1	221	316.26	19 hours, 24 minutes, 54 seconds	2010-02-22 20:12	2010-02-23 15:37	60
1	tvprobe003a/switch001	3510	16.25	15 hours, 50 minutes, 22 seconds	2010-02-22 22:24	2010-02-23 14:15	1807



Web-Frontend

- Current web-frontend is simple
  - Its not pretty-looking, but it works!

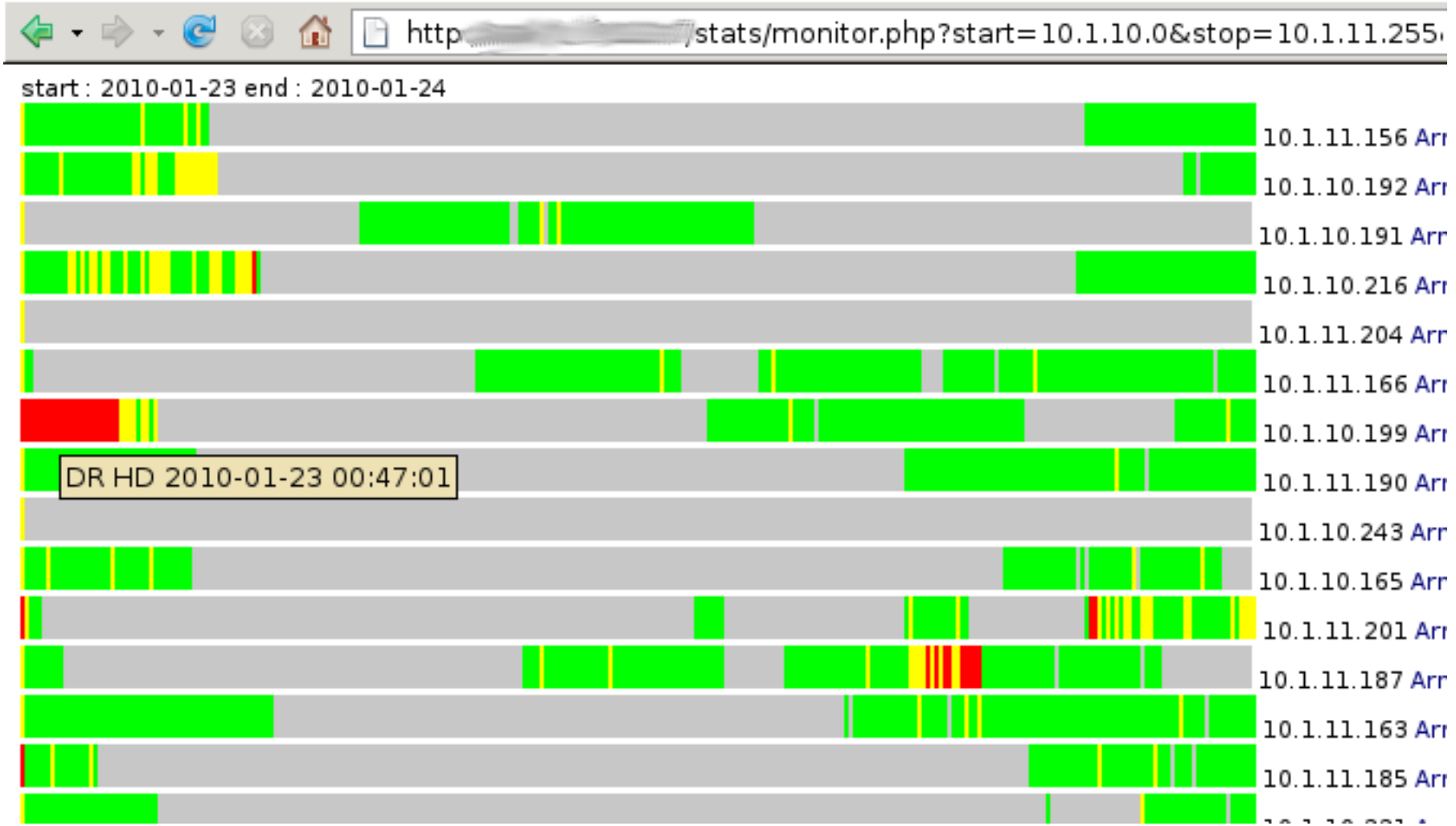
Help me PLEASE

- Web-developers needed on the project!
  - Want to rewrite frontend
    - Use AJAX and JSON instead of plain PHP
    - New graph module

Trick: Use settop box as probes

- See what the customer sees
  - As many probes as customers
  - But only one channel at a time
- Trick: The settop box runs Linux
  - local tool support asking for “sync errors”
  - install small bash script
    - periodically poll, and submit result back central
- Still need probes, in the network
  - need to identify the network segment

Settox boxes as probes



Open Source Status

- Wireshark
  - Drop patches accepted
  - PCR-clock patches, not accepted :-(
- IPTV-Analyzer gone 100% GPL
  - http://www.iptv-analyzer.org
  - Initial Public release (v0.9.0) 2011-05-09
  - 2011-05-24: v0.9.1 – no-signal detect
  - 2011-06-02: v0.9.2 – snmptrap no-signal events
  - git://github.com/netoptimizer/IPTV-Analyzer.git

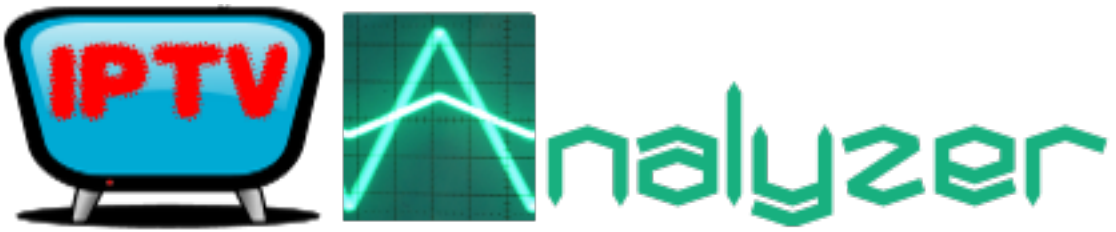
Future?

- Milestone 2: Bursts, *detect* issues, *before* drops occur
  - Currently down prioritized due new job...
- Milestone 3: Shaper, *solve* issue
  - Have an implementation idea...
  - Will it ever happen?
- Progress through cooperation?
  - Anybody want to cooperate?
  - Need some new “users” after ComX

The End

Goodbye

and thank you for your future patches ;-)



Want to try the IPTV-analyzer?

- **http://www.iptv-analyzer.org**

If using, please let me know, its a big motivation factor:

- netoptimizer@brouer.com

Switch buffer size is key!

- Problem comes from: The switch buffer size
  - Ethernet switches, are suppose to drops packets
  - QoS didn't help, all data was multicast
- From burst size
  - Directly calculate *required* switch buffer size
    - Fixed size packets of 1358 bytes on wire
    - Know your switch buffer sizes
    - don't place small-buffer switches in backbone
- BUT: Read about bufferbloat before increasing buffers

What is this talk NOT about

- Lot of challenges to implement multicast routing
  - Its not straight forward, BUT its not the focus of this talk.
- Shortly, pitfalls to watch out for:
  - It can be difficult to:
    - get different vendors multicast to work together
    - get even the same vendor
      - Watch out for firmware upgrades...
      - E.g: New firmware (on Foundry FWSX 424) drops packets, MC in HW.
  - Also Remember:
      - Apartment buildings Ethernet switches, proper config.
      - CPE equipment needs IGMP snooping, more Settop boxes