Working With Flow Data in an Academic Environment in the DDoSVax Project at ETH Zuerich

Arno Wagner

wagner@tik.ee.ethz.ch

Communication Systems Laboratory

Swiss Federal Institute of Technology Zurich (ETH Zurich)

Outline

1. Academic users
2. Context: The DDoSVax project
3. Data collection and processing infrastructure
4. Software / Tools
5. Technical lessons learned
6. Other lessons learned

Note: Also see my FloCon 2004 slides at `http://www.tik.ee.ethz.ch/~ddosvax/` or Google("ddosvax")

Academic Users

- PhD Researchers
- Students doing Semester-, (Diploma-) and Master-Theses
- (Almost) no forensic work

Users will write their own tools
$\Rightarrow$ Support is needed to make them productive fast:

- Software: Libraries, example tools, templates
- Initial explanations
- Advice and some supervision

The DDoSVax Project

`http://www.tik.ee.ethz.ch/~ddosvax/`

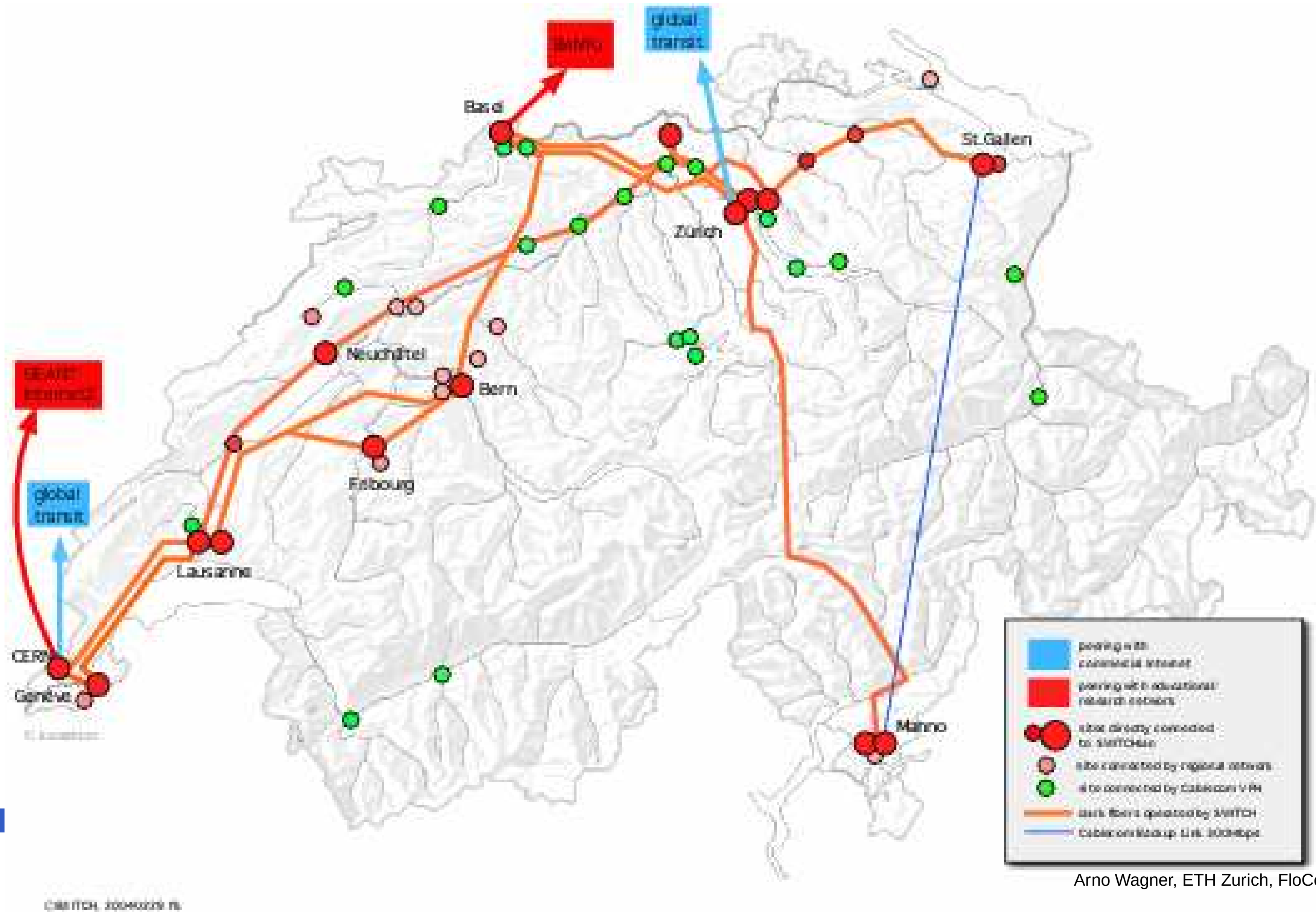
- Collaboration between SWITCH (www.switch.ch, AS559) and ETH Zurich (www.ethz.ch)
- Aim (long-term): Near real-time analysis and countermeasures for DDoS-Attacks and Internet Worms
- Start: Begin of 2003
- Funded by SWITCH and the Swiss National Science Foundation

DDoSVax Data Source: SWITCH

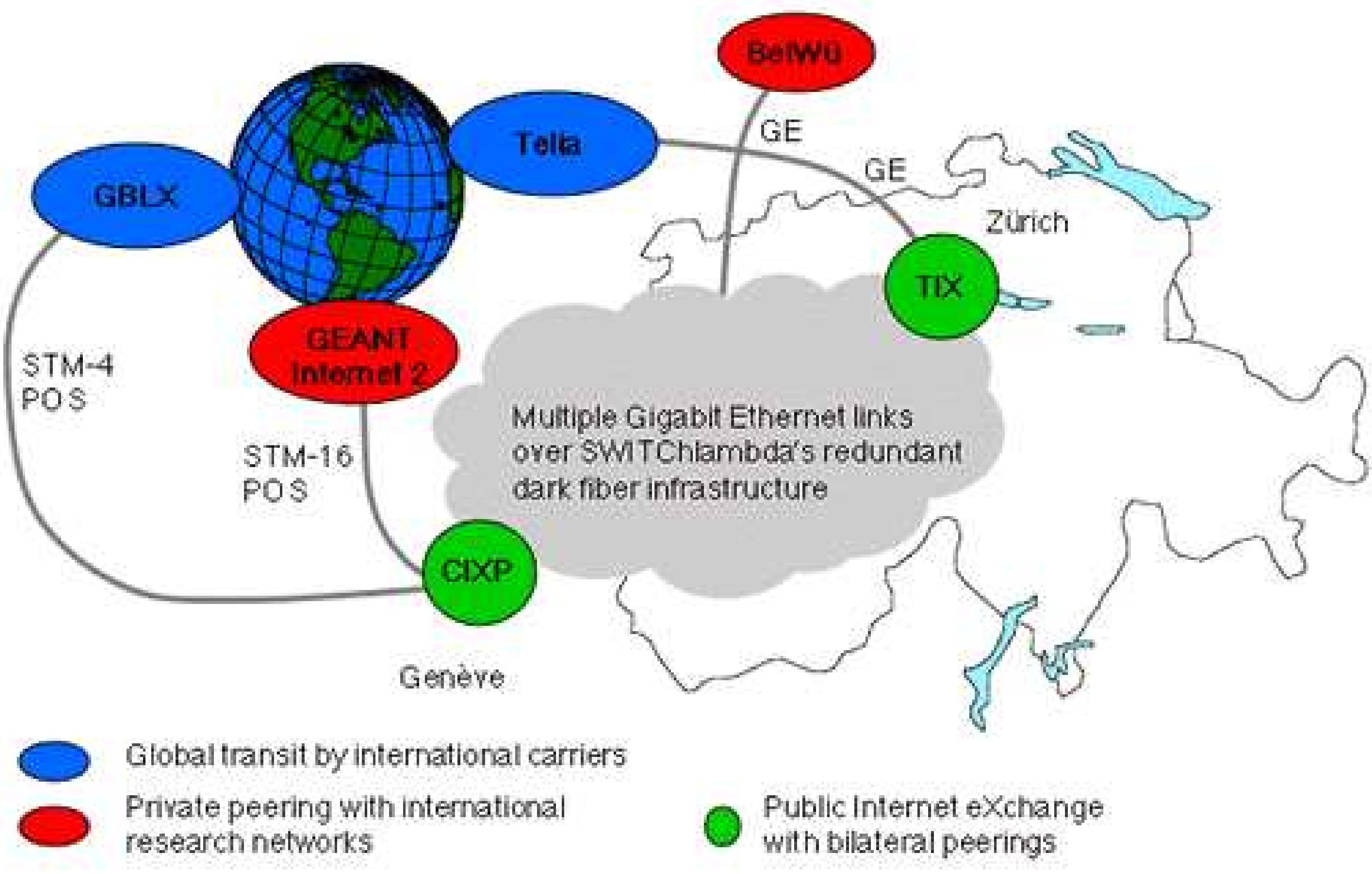
The Swiss Academic And Research Network

- .ch Registrar
- Links most Swiss Universities
- Connected to CERN
- Carried around 5% of all Swiss Internet traffic in 2003
- Around 60.000.000 flows/hour
- Around 300GB traffic/hour

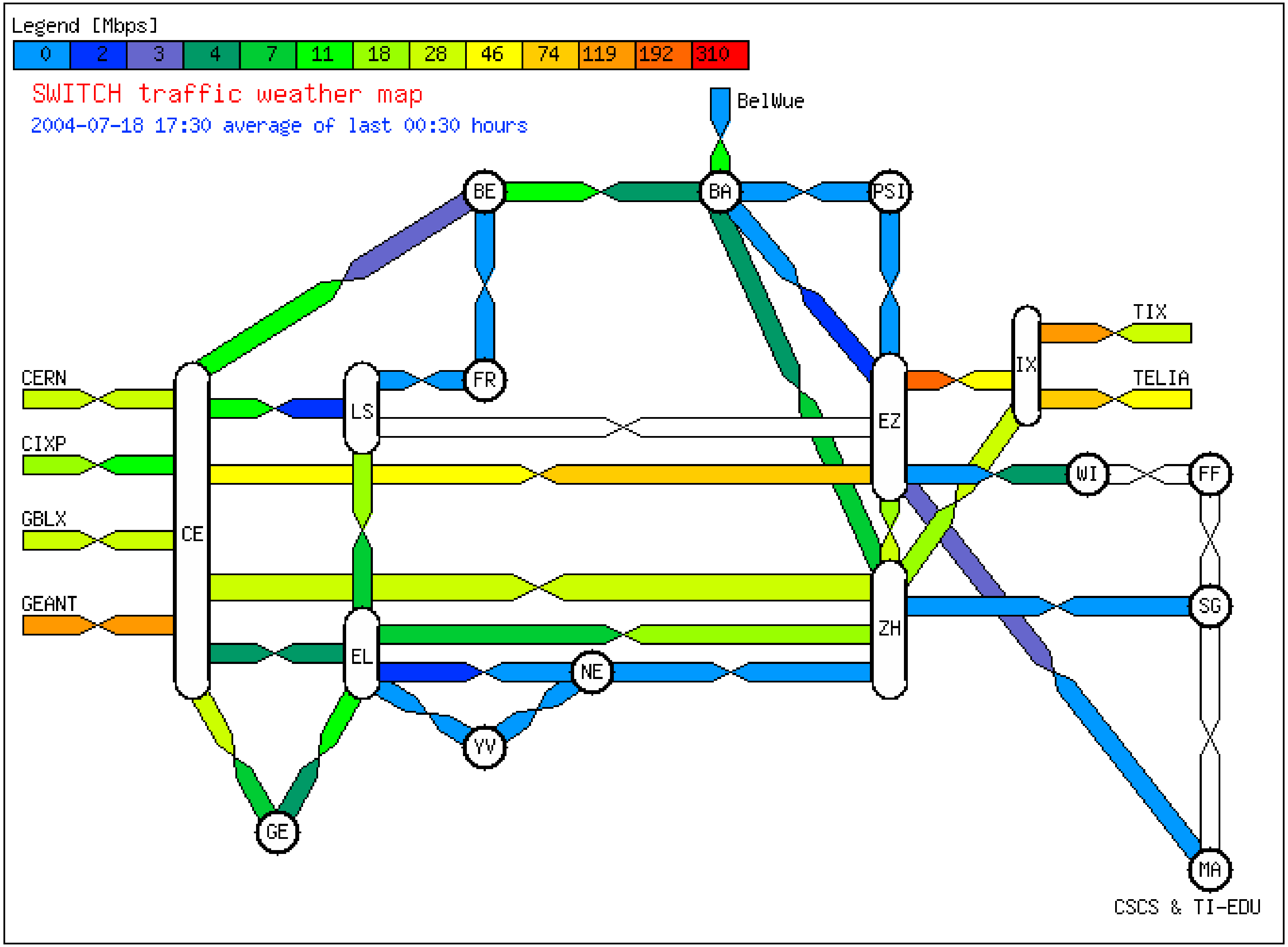
The SWITCH Network



SWITCH Peerings



SWITCH Traffic Map



NetFlow Data Usage at SWITCH

- Accounting
- Network load monitoring
- SWITCH-CERT, forensics
- DDoSVax (with ETH Zurich)

Transport: Over the normal network

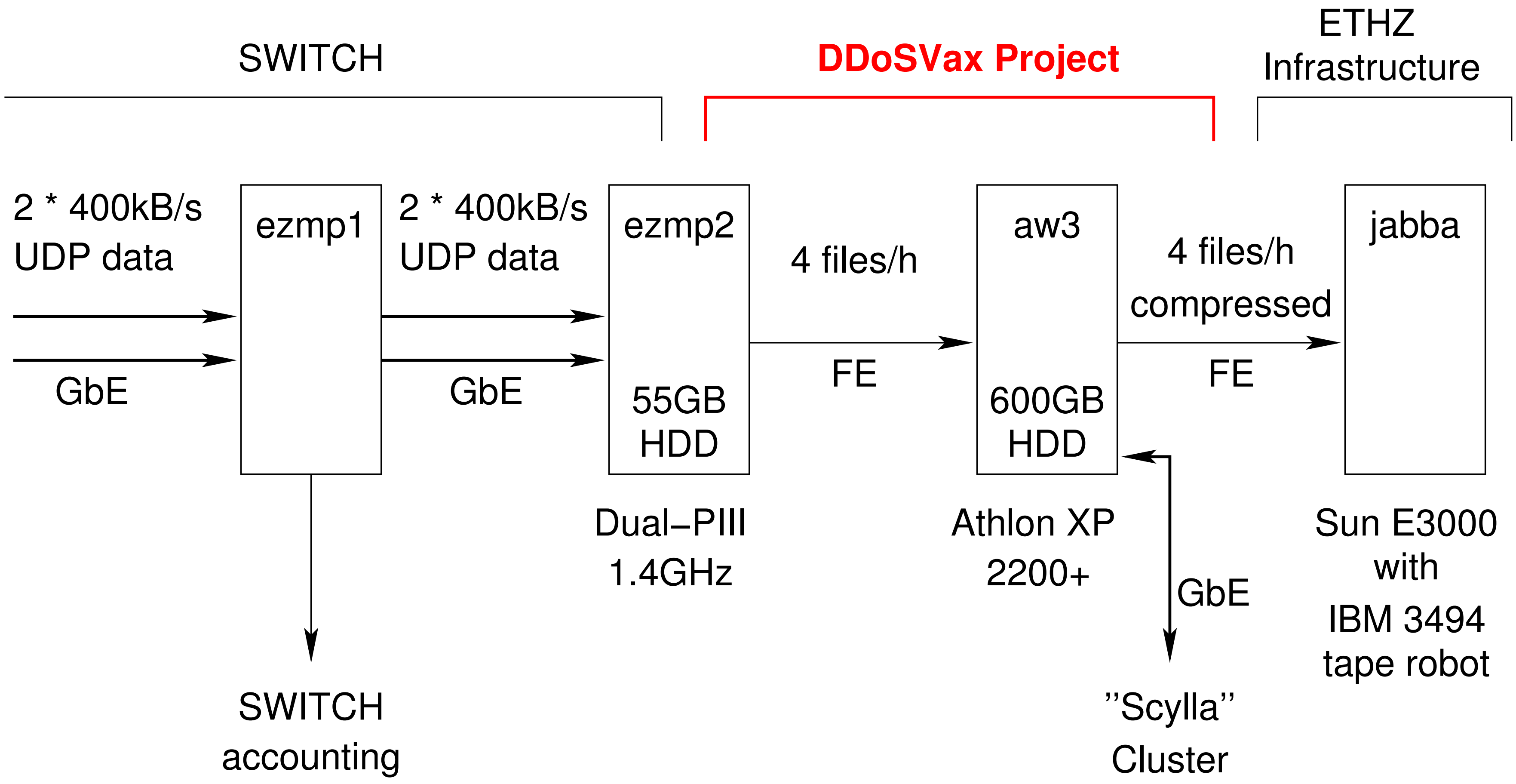
Collaboration Experience

- DDoSVax inspired SWITCH to crate their own short-term NetFlow archive for forensics
- Quite friendly and competent exchange with the (small, open minded) SWITCH technical and security staff.
- SWITCH may want to use our archive in the future as well
- Main issue with SWITCH: Privacy concerns

Network Dynamics

- No topological changes with regard to flow collection so far.
- Collection quality got better due to better hardware (routers).
- IP space (AS559) was a bit enlarged in the last year.

Collection Data Flow



NetFlow Capturing

- One Perl-script per stream
- Data in one hour files

Critical: (Linux) socket buffers:

- Default: 64kB/128kB max.
- Maximal possible: 16MB
- We use 2MB (app-configured)
- 32 bit Linux: May scale up to 5MB/s per stream

Capturing Redundancy

- Worker / Supervisor (both demons)
- Super-Supervisor (cron job)
  For restart on reboot or supervisor crash
- Space for 10-15 hours of data on collector

No hardware redundancy

Long-Term Storage

Unsampled flow-data since March 2003
Bzip2 compressed raw NetFlow V5 in one-hour files

- We need most data-fields and precise timestamps
- We don't know what to throw away
- We have the archive space
- Causes us to be CPU bound (usually)
  $\Rightarrow$ Makes software writing a lot easier!

Computing Infrastructure

The ”Scylla” Cluster
Servers:

- aw3: Athlon XP 2200+, 600GB RAID5, GbE
  does flow compression and transfer
- aw4: Dual Athlon MP 2800+, 3TB RAID5, GbE
- aw5: Athlon XP 2800+, 400GB RAID5, GbE

Nodes:

- 22 * Athlon XP 2800+, 1GB RAM, 200GB HDD, GbE

Total cost (est.): 35 000 USD + 3 MM

Software

- Basic NetFlow libraries (parsing, time handling, transparent decompression, . . .)
- Small tools (conversion to text, statistics, packet flow replay, . . .)
- Iterator templates: Provide means to step through one or more raw data files one a record-by-record basis
- Support libraries: Containers, IP table, PRNG, etc.

All in c (gcc), commandline only. Most written by me. Partially specific to SWITCH data.

Lessons Learned (Technical)

Software:

- KISS is certainly valid.
- Unix-tool philosophy works well.
- Human-readable formats and Perl or Python are very useful for prototyping and understanding.
- Add information headers (commandline, etc.) to output formats (also binary)!
- Take care on monitoring the capturing system.
- Keep a measurement log!

Lessons Learned (Technical)

Hardware/OS:

- Needed much more processing power and disks storage than anticipated
  $\Rightarrow$ Plan for infrastructure growth!
- Get good quality hardware.

Lessons Learned (Technical)

Capturing and storage: Bit-errors do happen!
We use **bzip2 -1** on 1 hour files (about 3:1)

- Observed: 4 bit errors in compressed data/year
- 1 year ~ 5TB compressed $\Rightarrow 1$ error $/\ 1.2 * 10^{12}$ Bytes
- bzip2 -1 $\Rightarrow$ loss of about 100kB per error
  Unproblematic to cut defect part
  Note: gzip, lzop, ... will loose *all* data after the error
- Source of errors: RAM, busses, (CPU), (disk), (Network)

Lessons Learned (Technical)

Processing: Bit Errors do happen!

- Scylla-Cluster used OpenMosix $\Rightarrow$ Process migration and load balancing
- Observed problem: Frequent data corruption.
- Source: A single weak bit in 44 RAM modules
  Diag-time with memtest86: > 3 days!
  Process migration made it vastly more difficult to find!
- No problems with disks, CPUs, network, tapes.
- Some problems with a 66MHz PCI-X bus on a server.

Lessons Learned (Users)

Students need to understand what they are doing.

- Human-readable and scriptable output helps a lot!
- Clean sample code is essential.
- Tell students what technical skills are expected *clearly* before they commit to a thesis.
- Make sure students code cleanly and that they understand algorithmic aspects.

Thank You!