# RRDtool Tips & Tricks

Tobias Oetiker <tobi@oetiker.ch>
OETIKER+PARTNER AG

# Recipe for Success

Resolve the problems
before anyone else finds them.

Tobi Oetiker <tobi@oetiker.ch>

# ... and talk about it

Being able is only half the story,
the others must know too!

# For RRDtool this means

- Figure the problem
- Find the relevant data
- Collect the data
- Understand the results
- Draw a pretty graph
- Show it to people

# Definition of an RRD file

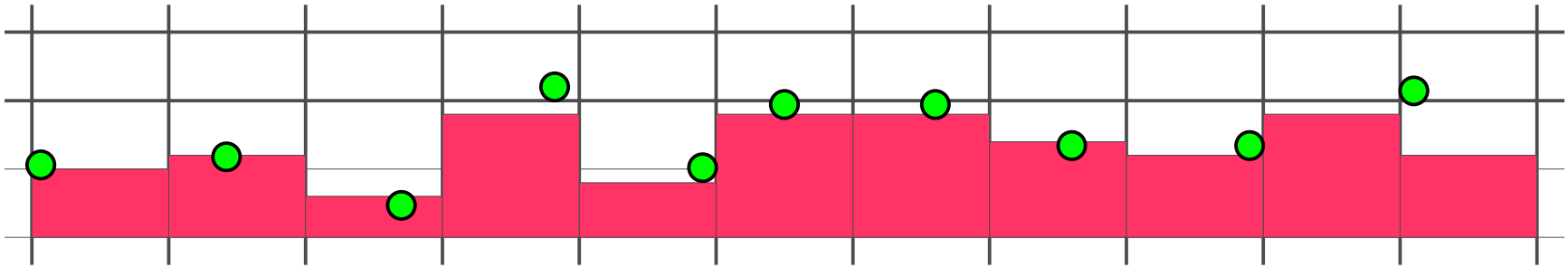
- Header
- Data sources – DS
- Round Robin Archives - RRA

# What is a Data Source

- Anything with numbers
- Log files
- SNMP counters
- /proc entries
- Output from an external program

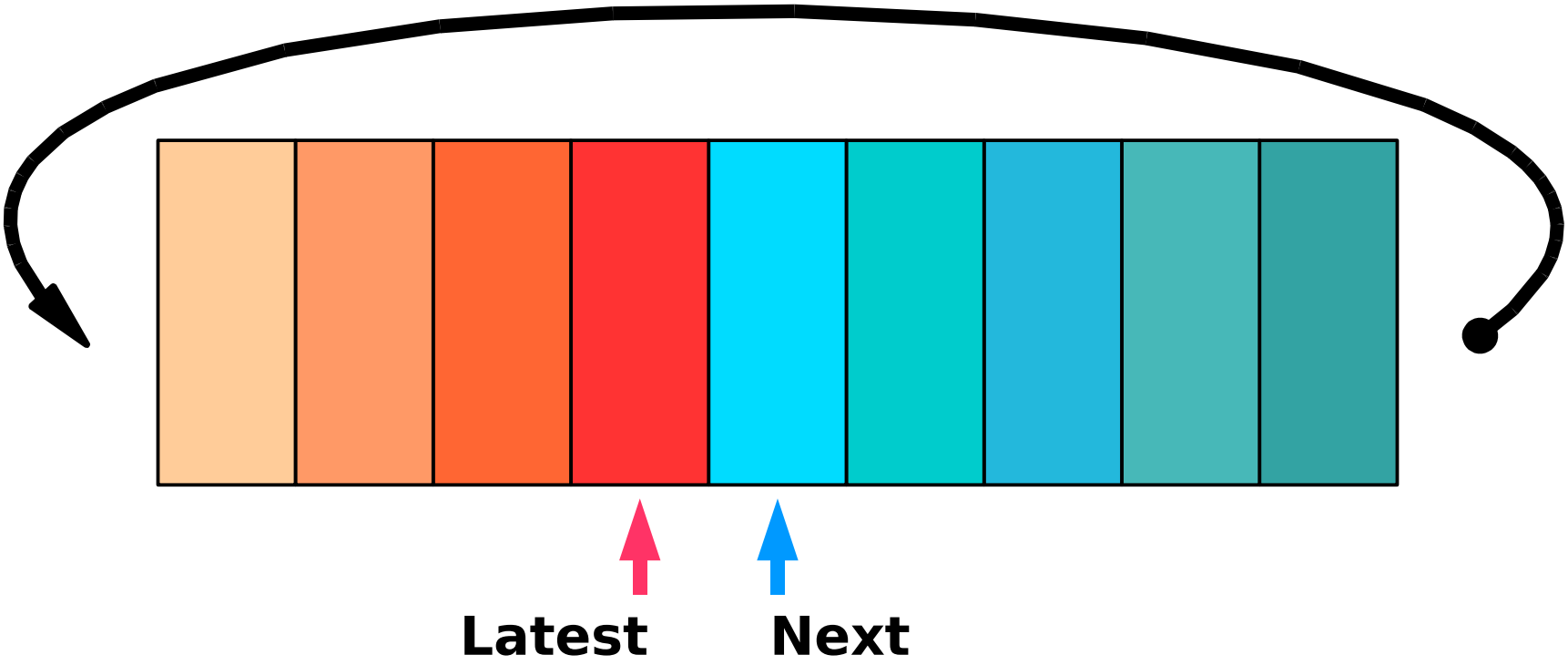
# Remember this on DS

- RRDtool handles the UNKNOWN
- Everything is a rate.
  “real data” is not kept!
- Pick the right sampling interval.
  (Double the frequency)



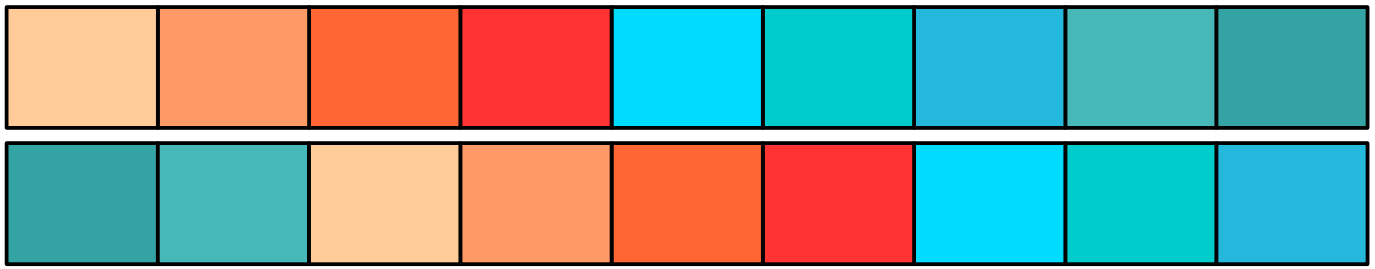
# The Round Robin Principle

- fixed number of storage slots



# Multiple Archives

- Keep data ready at the right resolutions
  - 5 Minute AVERAGE for 1 day
  - 1 hour MAX for a month




- Fit it to the questions you expect.

# RRD pitfalls I

- I send an update every 5 minutes, but no data shows up in my RRD files.

Where would you look ?

# RRD pitfalls II

- I used update to enter value X, but when I use "`rrdtool fetch`" to verify, I see only value Y.

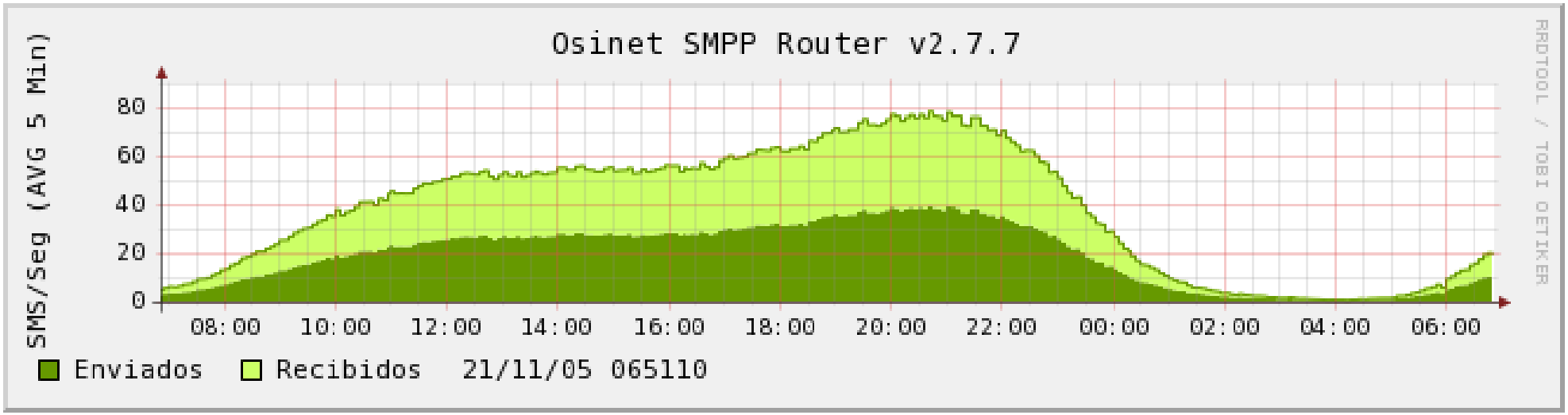
What happened to value X?

# RRD pitfalls III

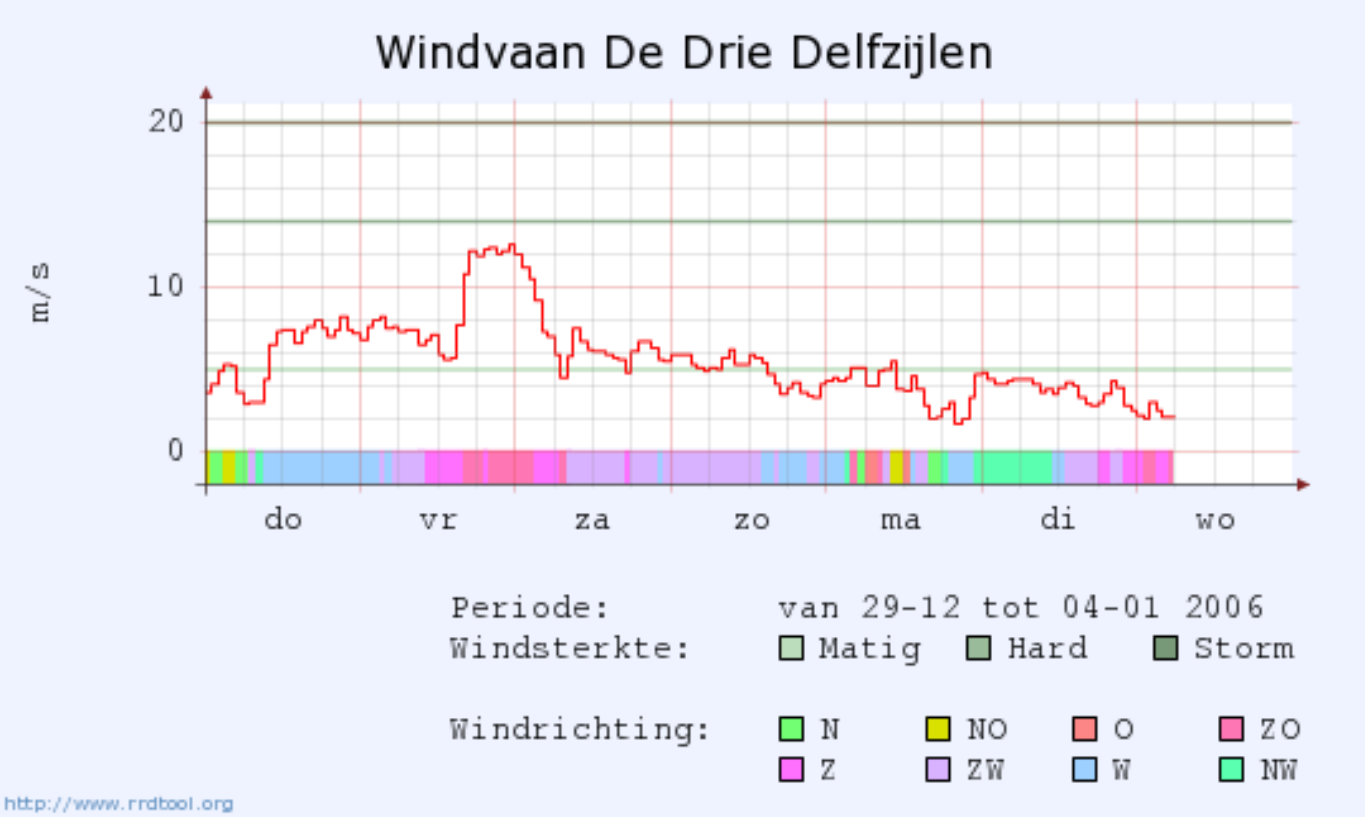
- I created an RRD file with `--start 100000`. Running the first update takes incredibly long.

  What is rrdtool doing ?

# Presenting the Results I



# Presenting the Results II




Tobi Oetiker <tobi@oetiker.ch>

# RRDtool Graph work flow

- Get the data
- Run RPN calculations
- Put it on the graph

```
rrdtool graph my.png \
  DEF:in=first.rrd:speed:AVERAGE \
  CDEF:in8=in,8,* \
  LINE:in8#ff0000:'Bit Speed'
```

# Getting the Data

- man rrdgraph_data

```
DEF:x=file.rrd:speed:MAX...
DEF...:step=3600
DEF...:start=end-1h
DEF...:end=11\:00
DEF...:reduce=AVERAGE
```

# Calculating with RPN

- RPN = Reverse Polish Notation
- This is what HP Calculators used to do
- 1 [enter] 2 [enter] [+]
- no operator precedence
- simple programming language
- 1 [enter] 2 [enter] [keep larger]

# RRDtool Graph RPN Basic

- man rrdgraph_rpn

- CDEF:bits=octets,8,*
- CDEF:avg=in,out,+,2,/
- CDEF:bigger=x,y,MAX
- CDEF:lim=a,0,100,LIMIT

Tobi Oetiker <tobi@oetiker.ch>

# RRDtool Graph RPN Adv

- man rrdgraph_rpn

- CDEF:bigger=x,y,LT,x,y,IF
- CDEF:unzero=x,UN,0,x,IF
- CDEF:avg2=v1,v2,v3,3,AVG
- CDEF:wind=x,1800,TREND
- CDEF:time=x,POP,TIME

# CDEF Pitfalls

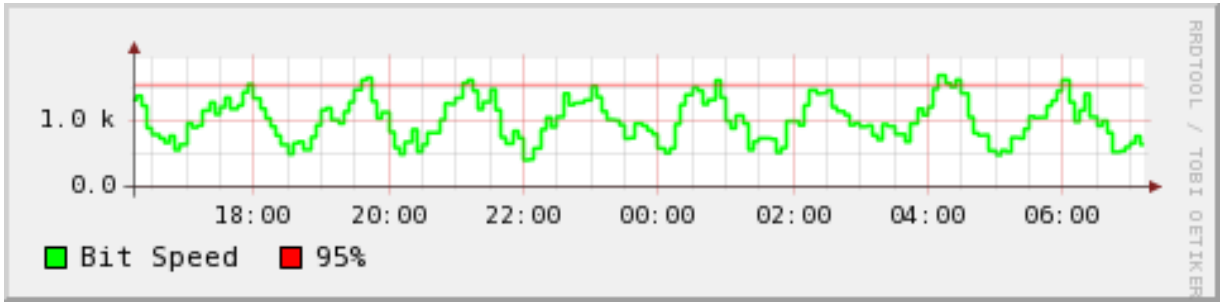
- there must be a DEF/CDEF variable in every CDEF the expression.
- CDEF:x=1,2,+  is INVALID!
- Trick:
  CDEF:x=y,POP,1,2,+

# VDEF Expressions

- VDEF:var=data,95,PERCENT
- gives a single value!
- looks like CDEF but it’s NOT
- result is usable in CDEF

# Drawing a “simple” graph

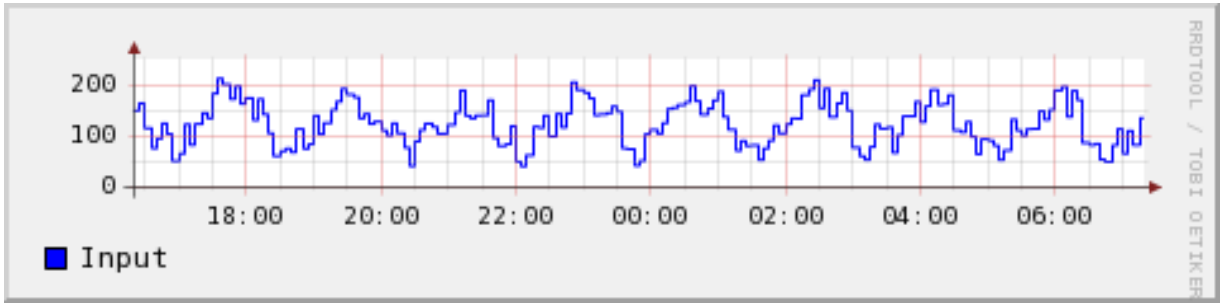
```
rrdtool graph my.png \
 DEF:in=first.rrd:speed:AVERAGE \
 CDEF:bits=in,8,* \
 VDEF:ninefive=bits,95,PERCENT \
 LINE2:in8#00ff00:’Bit Speed’ \
 LINE0.5:ninefive#ff0000:’95%’
```



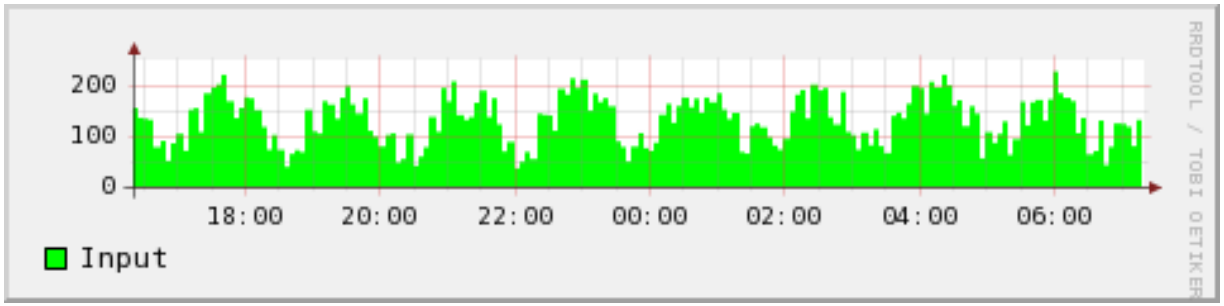

Tobi Oetiker <tobi@oetiker.ch>

# Drawing Elements

- `LINE:input#0000ff:Input`

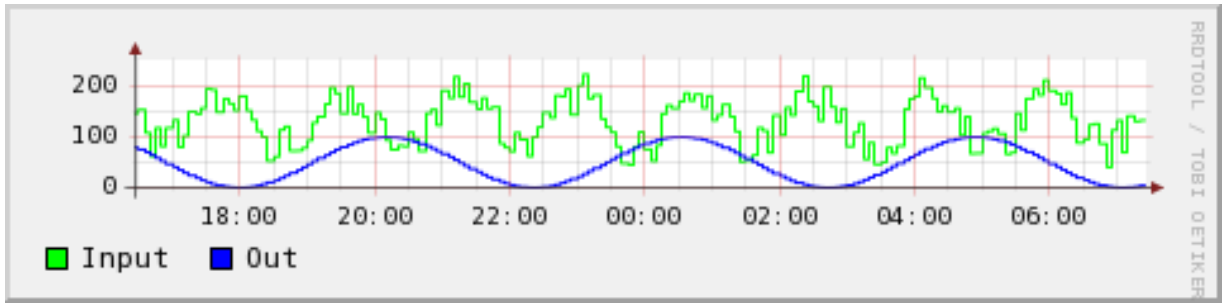



- `AREA:input#00ff00:Input`

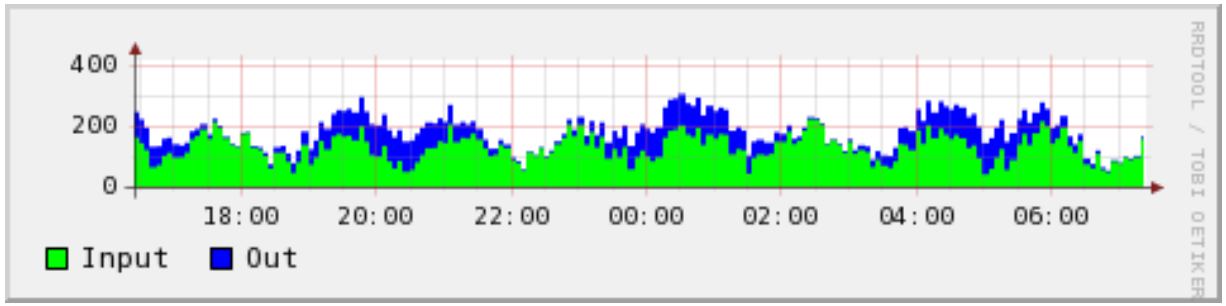


# Graphing several Elements

- LINE:... LINE:....




- AREA:... AREA:...:**STACK**



# Talking through Graphs

- What is your goal ?
- Who is your audience ?
- What is the context ?

# Graphs with a Message

- We need more bandwidth.
- It is too hot in the server room.
- You are over your bandwidth allocation.
- The compute cluster is overloaded.
- All services are running ‘green’

# What would convince you?

Tobi Oetiker <tobi@oetiker.ch>

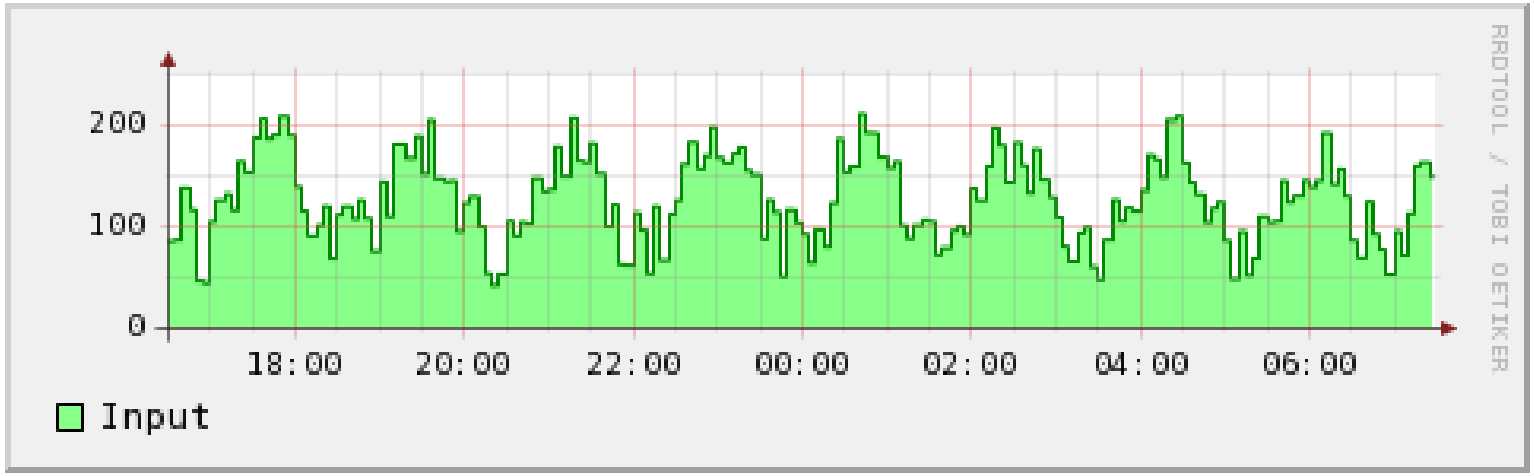
# Design Principles

- Use the color scheme of your org.
- Mimic other designs.
- Fit into the context.
- The design supports the message!

If your users remember the design.
You have missed your goal!

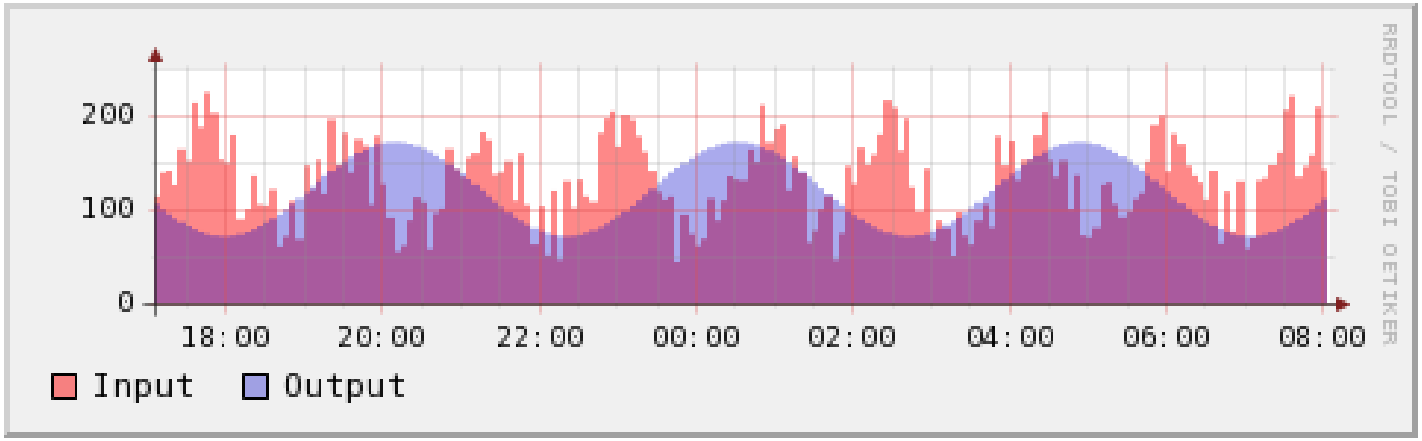
# Effect: Outlining

```
rrdtool graph my.png \
 DEF:in=first.rrd:speed:AVERAGE \
 AREA:in#8f8:’Bit Speed’ \
 LINE:in#080
```



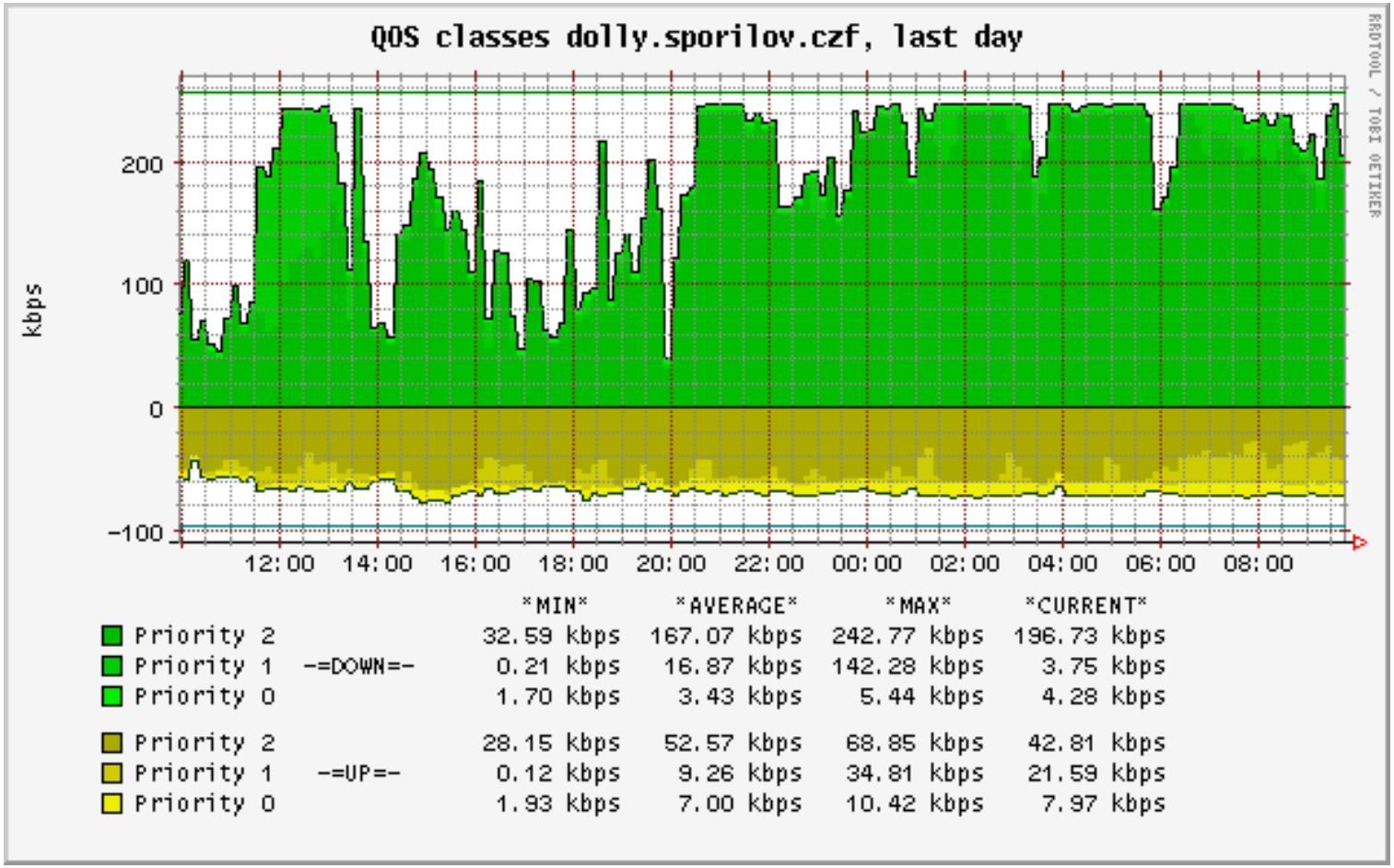
# Effect: Transparency

```
rrdtool graph my.png \
 DEF:in=first.rrd:in:AVERAGE \
 DEF:out=first.rrd:out:AVERAGE \
 AREA:in#f007:Input \
 AREA:in#0f05:Output
```



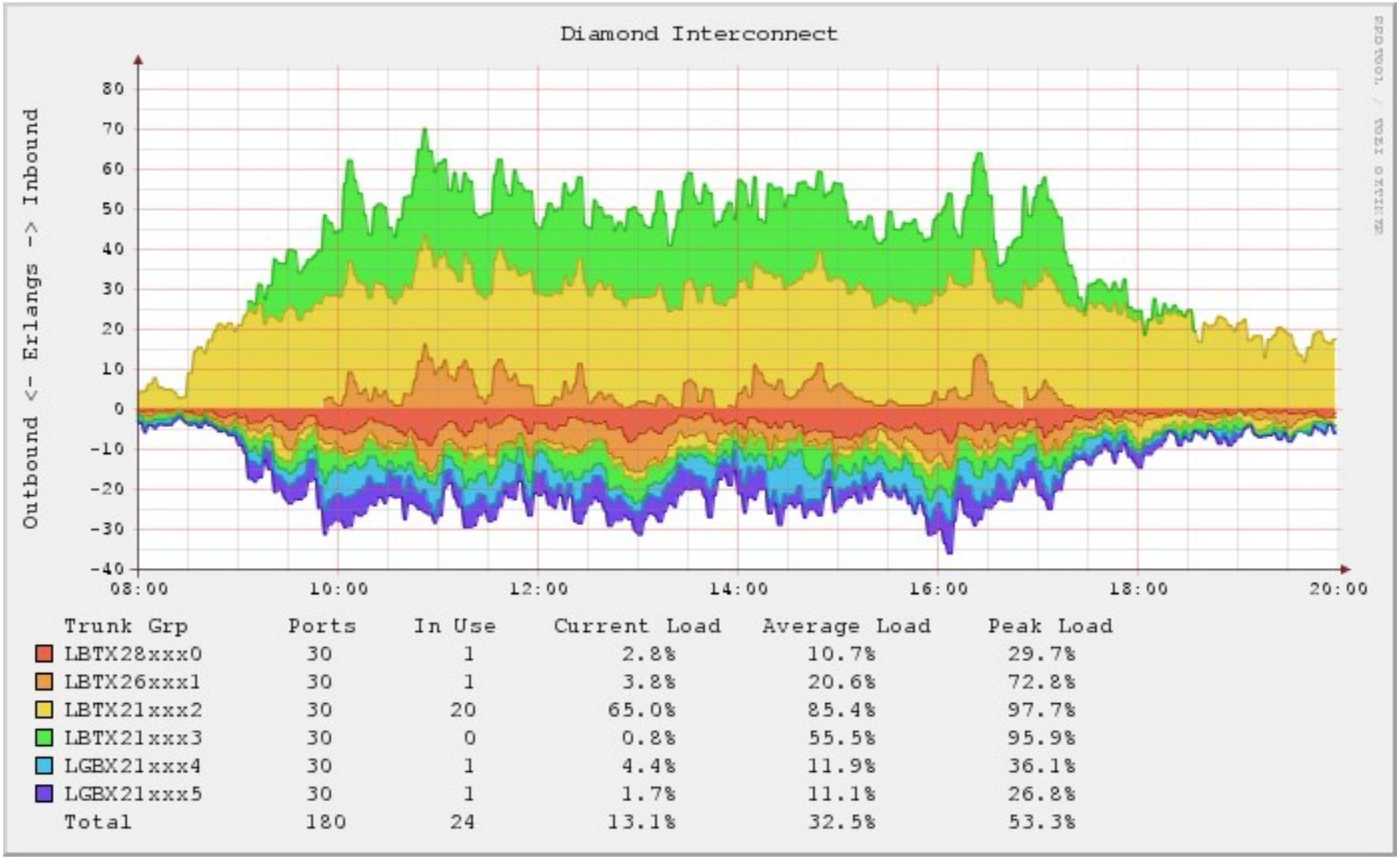

Tobi Oetiker <tobi@oetiker.ch>

# Gallery I



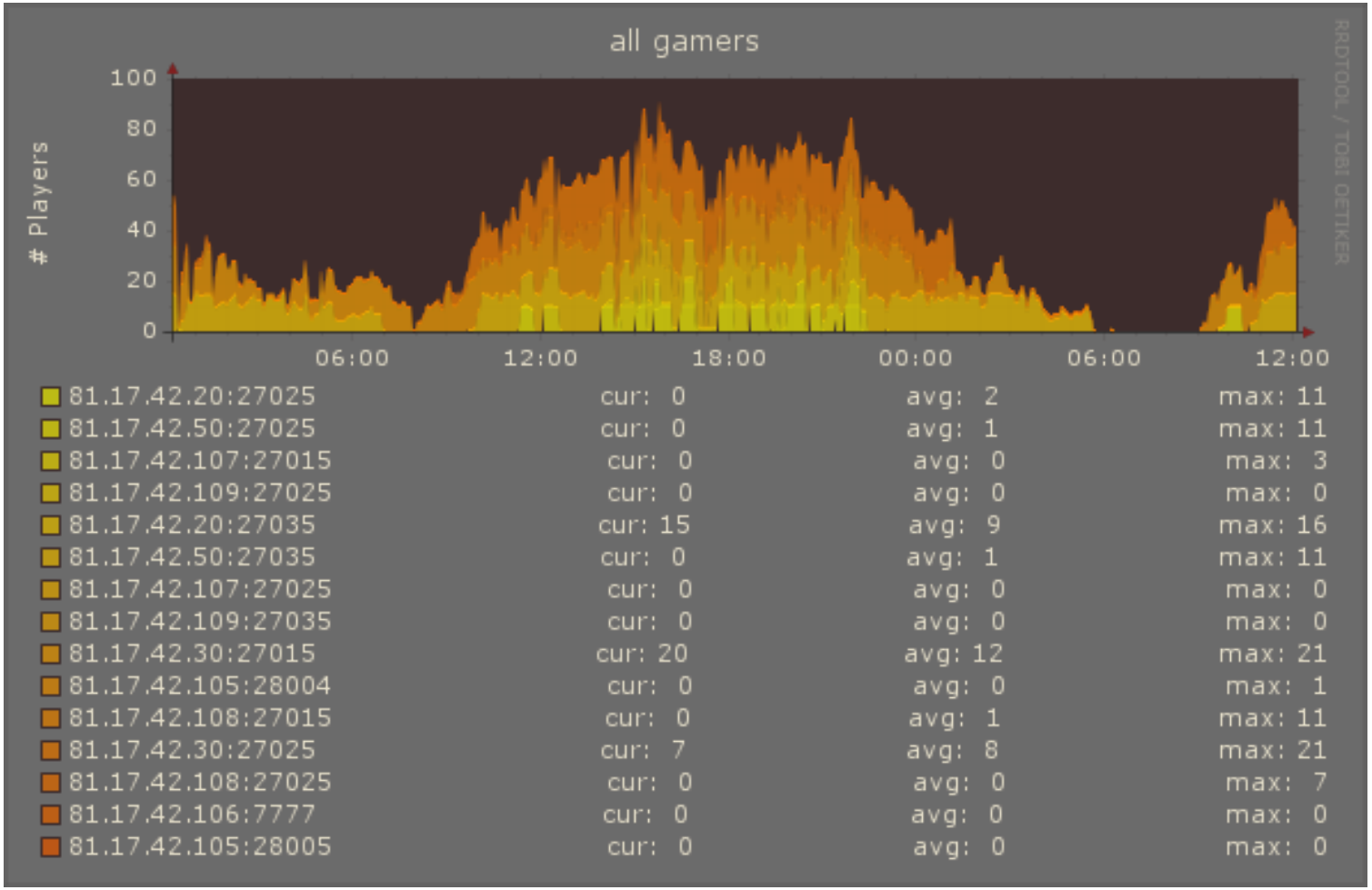

Tobi Oetiker <tobi@oetiker.ch>

# Gallery II




| Trunk Grp | Ports | In Use | Current Load | Average Load | Peak Load |
|---|---|---|---|---|---|
| LBTX28xxx0 | 30 | 1 | 2.8% | 10.7% | 29.7% |
| LBTX26xxx1 | 30 | 1 | 3.8% | 20.6% | 72.8% |
| LBTX21xxx2 | 30 | 20 | 65.0% | 85.4% | 97.7% |
| LBTX21xxx3 | 30 | 0 | 0.8% | 55.5% | 95.9% |
| LGBX21xxx4 | 30 | 1 | 4.4% | 11.9% | 36.1% |
| LGBX21xxx5 | 30 | 1 | 1.7% | 11.1% | 26.8% |
| Total | 180 | 24 | 13.1% | 32.5% | 53.3% |

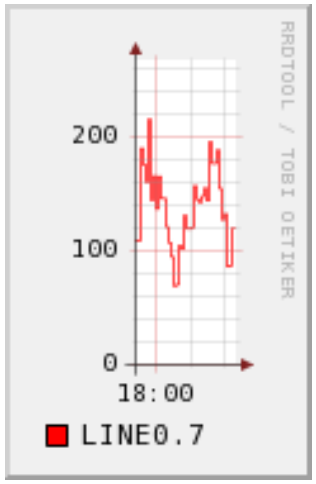
# Gallery III

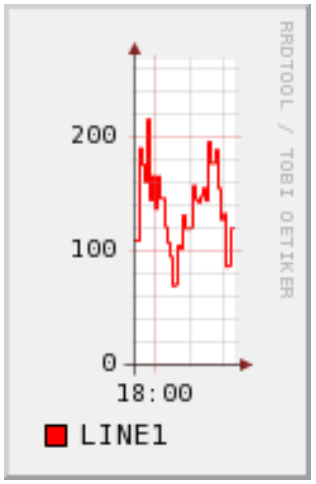


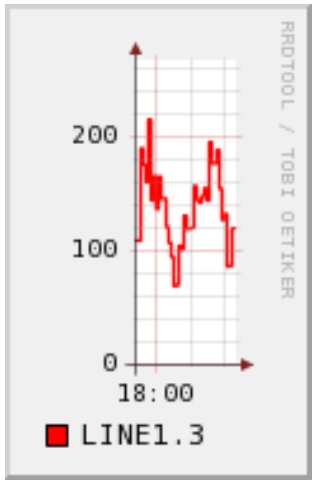

Tobi Oetiker <tobi@oetiker.ch>

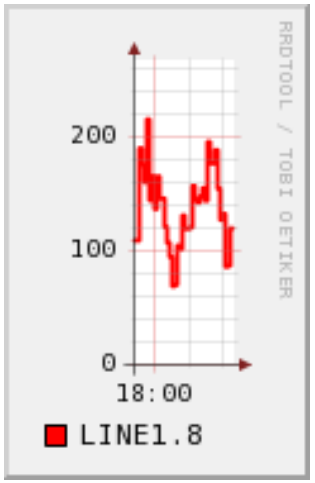
# Graphing Pitfalls

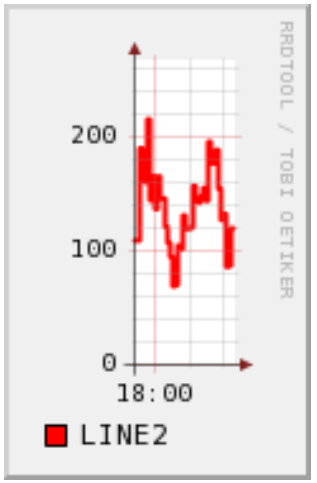
- Too much/little Design
- Too much/little Content
- Soft Lines




Tobi Oetiker <tobi@oetiker.ch>

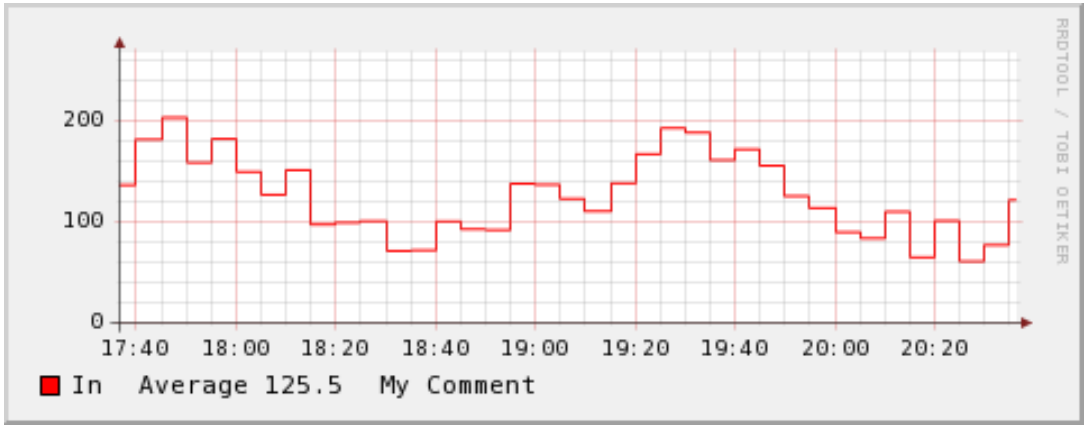
# Adding Number to Graphs

- VDEF results can be printed
- Into the graph with GPRINT
- To the caller with PRINT
- Use sprintf formatting %lf

```
%lf      0.666666666667
%3.2lf   0.67
```

# GPRINT and COMMENT

```
rrdtool graph my.png \
  DEF:in=first.rrd:speed:AVERAGE \
  LINE:in#f00 VDEF:avg=a,AVERAGE \
  GPRINT:avg:"Average %5.1lf" \
  COMMENT:"My Comment"
```




Tobi Oetiker <tobi@oetiker.ch>

# Whats new in 1.3 ?

Tobi Oetiker <tobi@oetiker.ch>

# MMAP IO

- Most IO switched to MMAP
- From 12k to 20k "in core" updates per second on Core Duo.
- 6 million Updates in 5 Minutes.
- ... if there were no Disks

by Bernhard Fischer

# fadvise / madvise

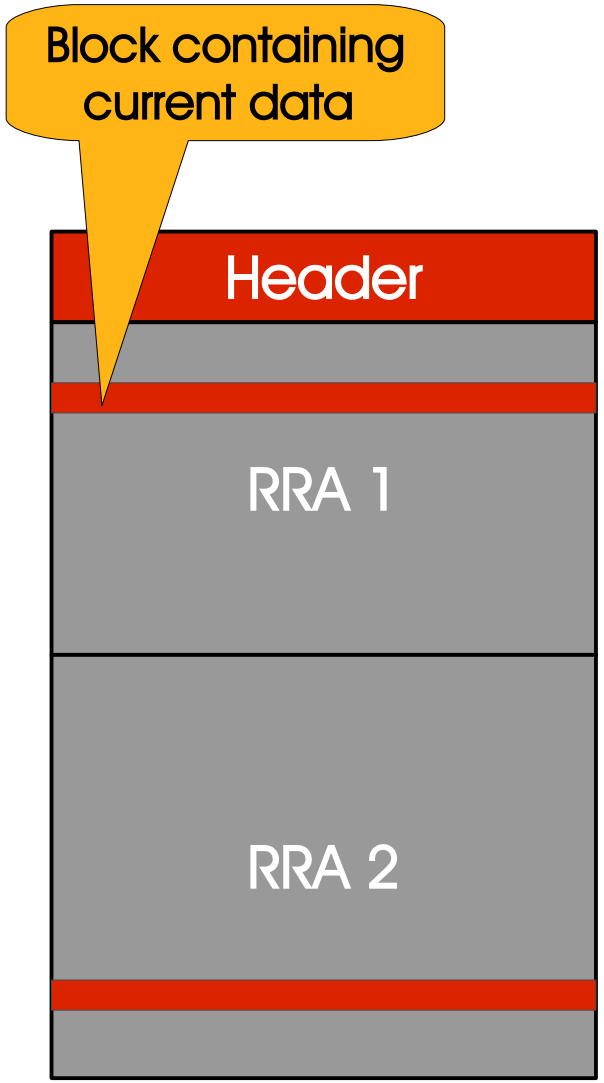
- We know more about caching RRDs than the OS!
- Use fadvise and madvise
- Stop read ahead with: POSIX_FADV_RANDOM, MADV_RANDOM
- At least on Linux: moving target.

# Cache preservation

- Use fdatasync, FADV_DONTNEED to drop data we do not need.
- Keep more sensible data in cache.
- Writing a double (8 byte) in a 512 byte block requires block read/write unless in cache.
- More cached = more speed.

# "Hot blocks"

- All hot blocks in-core saves lots of reads.
- Limited by available cache.
- RRDtool helps by evicting "useless" data.
- Only tested on Linux.



# Performance Checklist

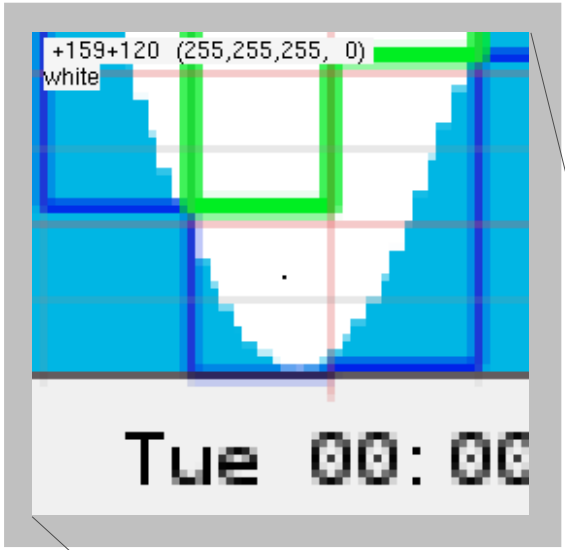
- Cache for all hot-blocks (64bit is good): 512 (Header), 512 * RRAs
- Ultimately disk bound.
- On a Thinkpad T60 (Core Duo, 1 GB): 20k in core updates per s -> mem-lim 300 updates to disk per s -> 90k/5m
- see also /proc/sys/vm/dirty*

# We love Cairo

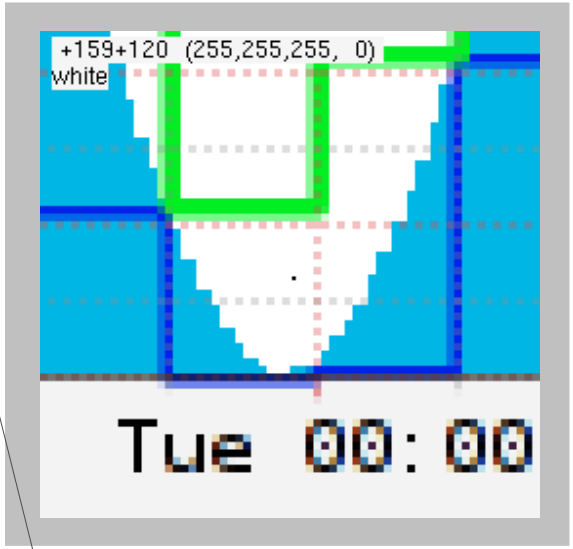
- libart replaced by cairo and pango (12 years ago MRTG wrote ppm files)
- instant gratification:
  - png, pdf, eps, svg
  - fontconfig
  - inline formatting
  - configurable anti aliasing

# Optical tuning – grid fitting

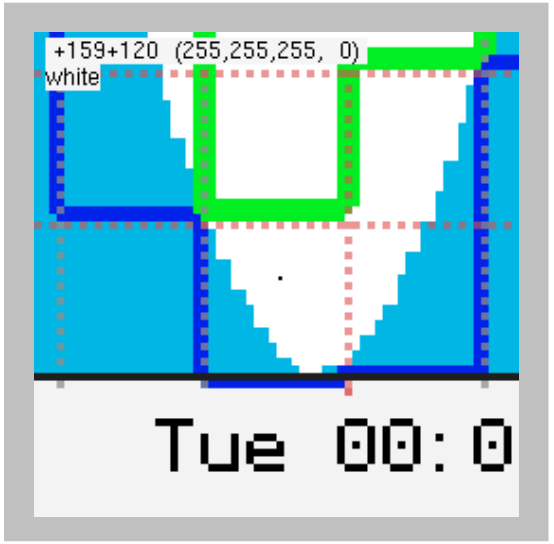
**1.2**

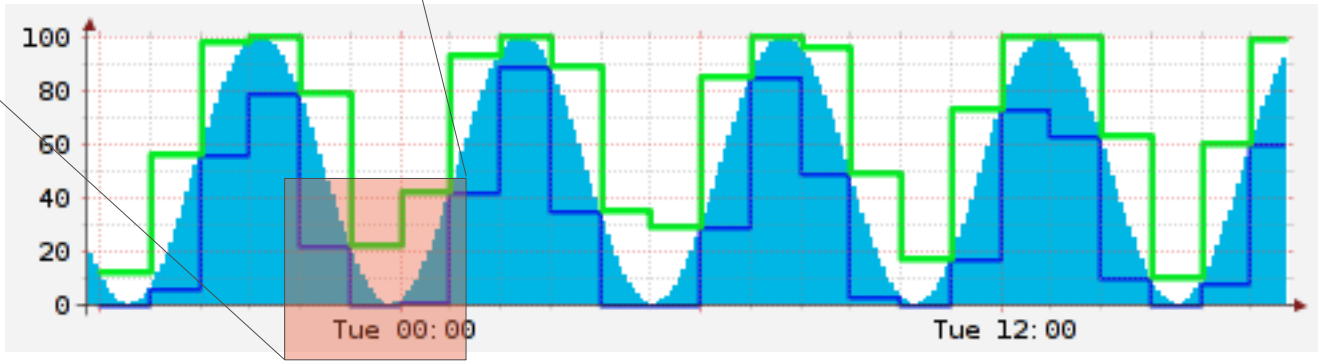



**1.3**




**1.3 anti-anti-alias**




Tobi Oetiker <tobi@oetiker.ch>

# Inline text formatting

- All text rendered with pango markup text
- Keys for font-size, font-family, color, underline, strike-through, raise, style.
- Shortcuts for Bold (b), Big (big), Italic (i), Strikethrough (s), Subscript (sub), Superscript (sup), Small (small), Monospace (tt), Underline (u)

# Multiplicative Holt Winters

- Support for a moving base line: MHPREDICT
- Seasonal variation is a coefficient multiplied by the baseline.
- New format version (004) when used.

by Evan Miller of IMVU

# Plans for 1.4

- Portable data format
  - IEEE 754 is really almost the same everywhere.
  - only NAN representation varies
  - data alignment is arch/os dependent.
  - portable format is a few "defines" away.

# Summary

- feature complete beta out now.
- memory mapped IO all over.
- fadvise and madvise for cache optimization.
- graphing switched to cairo/pango.
- Holt-Winters support for moving baseline.