RRDtool advanced Topics

Tobias Oetiker

OETIKER+PARTNER AG

OpenNMS User Conference 2013

A different kind of Database

creating a simple rrd

```
#!/bin/sh
PATH=/scratch/rrd4/bin:$PATH
R=rrdtool
$R create first.rrd \
   --step=300 \
   --start=1199999699 \
   DS:temperature:GAUGE:600:-40:100 \
   RRA:AVERAGE:0.4:1:5 \
   RRA:AVERAGE:0.4:3:2 \
   RRA:MIN:0.4:3:2 \
   RRA:MAX:0.4:3:2
```

One Datasource, 4 Round Robin Archives

feeding data

```
1
2  #!/bin/sh
3  R=rrdtool
4  u(){
5   $R update first.rrd $1
6  }
7
8  u 1199999700:00
9  u 1200000000:10
```

Feed in some data. One or several updates at once.

inside the database I

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rrd SYSTEM
                    "http://oss.oetiker.ch/rrdtool/rrdtool.dtd">
<rrd> <version> 0003 </version>
    <step> 300 </step> <!-- Seconds -->
    <lastupdate> 1200000900 </lastupdate>
    <!-- 2008-01-10 22:35:00 CET -->

    <ds>
        <name> temperature </name>
        <type> GAUGE </type>
        <minimal_heartbeat> 600 </minimal_heartbeat>
        <min> -4.0000000000e+01 </min>
        <max> 1.0000000000e+02 </max>

        <!-- PDP Status -->
        <last_ds> 40 </last_ds>
        <value> 0.0000000000e+00 </value>
        <unknown_sec> 0 </unknown_sec>
    </ds>
```

inside the database II

```
24      <!-- RRA:AVERAGE:0.4:1:5 -->
25      <rra>
26          <cf> AVERAGE </cf>
27          <pdp_per_row> 1 </pdp_per_row> <!-- 300 seconds -->
28
29          <params>
30          <xff> 4.0000000000e-01 </xff>
31          </params>
32          <cdp_prep>
33              <ds>
34              <primary_value> 4.0000000000e+01 </primary_value>
35              <secondary_value> 0.0000000000e+00 </secondary_value>
36              <value> NaN </value>
37              <unknown_datapoints> 0 </unknown_datapoints>
38              </ds>
39          </cdp_prep>
40          <database>
41              <row><v> NaN </v></row>
42              <row><v> 1.0000000000e+01 </v></row>
43              <row><v> 2.0000000000e+01 </v></row>
44              <row><v> 3.0000000000e+01 </v></row>
45              <row><v> 4.0000000000e+01 </v></row>
46          </database>
47      </rra>
```

inside the database III

```
48
49      <!-- RRA:AVERAGE:0.4:3:2  -->
50      <rra>
51          <cf> AVERAGE </cf>
52          <pdp_per_row> 3 </pdp_per_row> <!-- 900 seconds -->
53
54          <params>
55          <xff> 4.0000000000e-01 </xff>
56          </params>
57          <cdp_prep>
58              <ds>
59              <primary_value> 2.0000000000e+01 </primary_value>
60              <secondary_value> 3.0000000000e+01 </secondary_value>
61              <value> 4.0000000000e+01 </value>
62              <unknown_datapoints> 0 </unknown_datapoints>
63              </ds>
64          </cdp_prep>
65          <database>
66              <row><v> NaN </v></row>
67              <row><v> 2.0000000000e+01 </v></row>
68          </database>
69      </rra>
70
71
```

inside the database IV

```
72
73      <!--  RRA:MIN:0.4:3:2 -->
74      <rra>
75          <cf> MIN </cf>
76          <pdp_per_row> 3 </pdp_per_row> <!-- 900 seconds -->
77
78          <params>
79          <xff> 4.0000000000e-01 </xff>
80          </params>
81          <cdp_prep>
82              <ds>
83              <primary_value> 1.0000000000e+01 </primary_value>
84              <secondary_value> 3.0000000000e+01 </secondary_value>
85              <value> 3.0000000000e+01 </value>
86              <unknown_datapoints> 0 </unknown_datapoints>
87              </ds>
88          </cdp_prep>
89          <database>
90              <row><v> NaN </v></row>
91              <row><v> 1.0000000000e+01 </v></row>
92          </database>
93      </rra>
94
95
```

inside the database V

```
96        <!--  RRA:MAX:0.4:3:2 -->
97        <rra>
98            <cf> MAX </cf>
99            <pdp_per_row> 3 </pdp_per_row> <!-- 900 seconds -->
100
101           <params>
102           <xff> 4.0000000000e-01 </xff>
103           </params>
104           <cdp_prep>
105               <ds>
106               <primary_value> 3.0000000000e+01 </primary_value>
107               <secondary_value> 3.0000000000e+01 </secondary_value>
108               <value> 4.0000000000e+01 </value>
109               <unknown_datapoints> 0 </unknown_datapoints>
110               </ds>
111           </cdp_prep>
112           <database>
113               <row><v> NaN </v></row>
114               <row><v> 3.0000000000e+01 </v></row>
115           </database>
116       </rra>
117
118   </rrd>
```

rrd features

- optimized for time-series data
- fixed size rotating data store
- constant on-disk size
- no maintenance
- on the fly consolidation

rrd features

- optimized for time-series data
- fixed size rotating data store
- constant on-disk size
- no maintenance
- on the fly consolidation

rrd features

- optimized for time-series data
- fixed size rotating data store
- constant on-disk size
- no maintenance
- on the fly consolidation

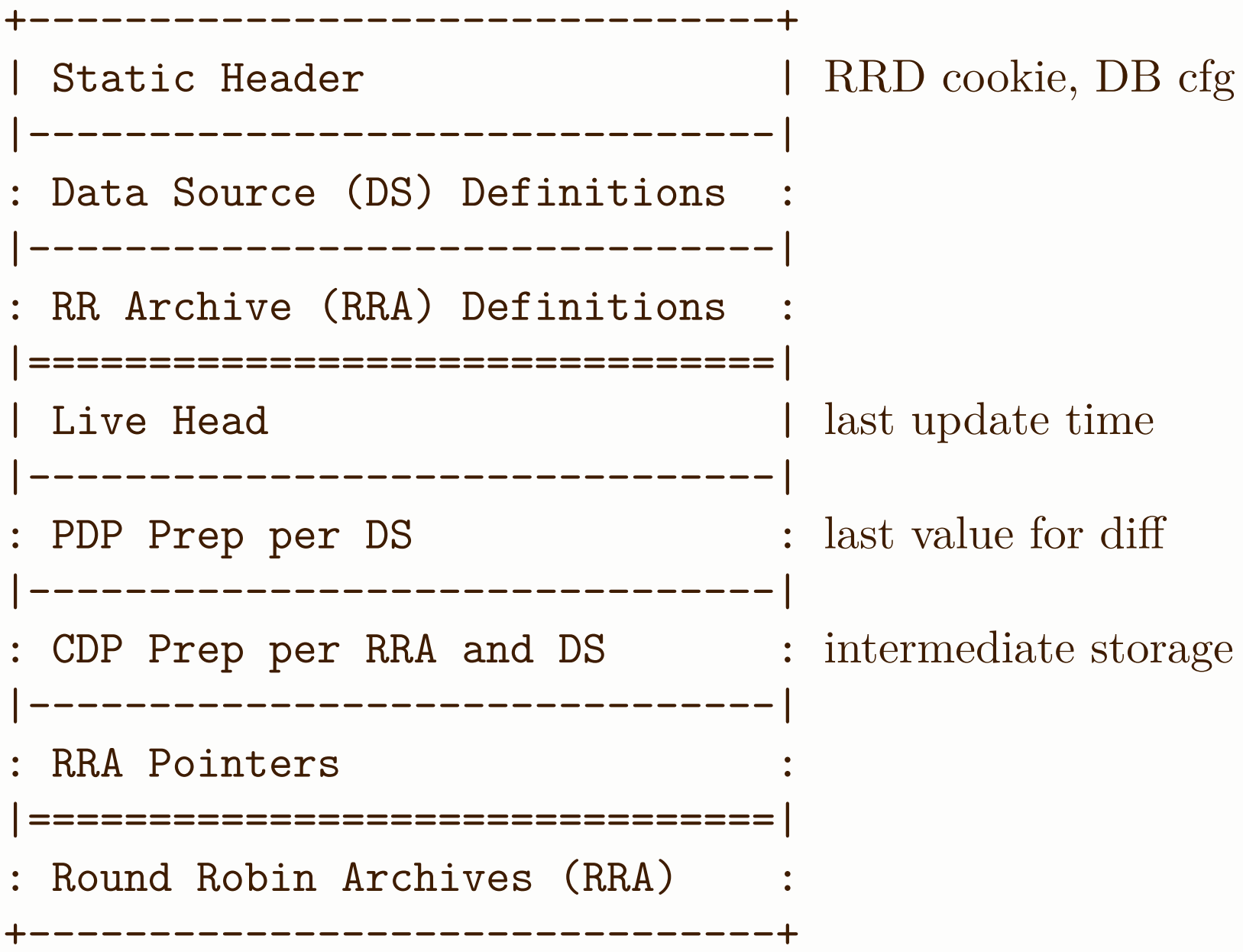
rrd features

- optimized for time-series data
- fixed size rotating data store
- constant on-disk size
- no maintenance
- on the fly consolidation

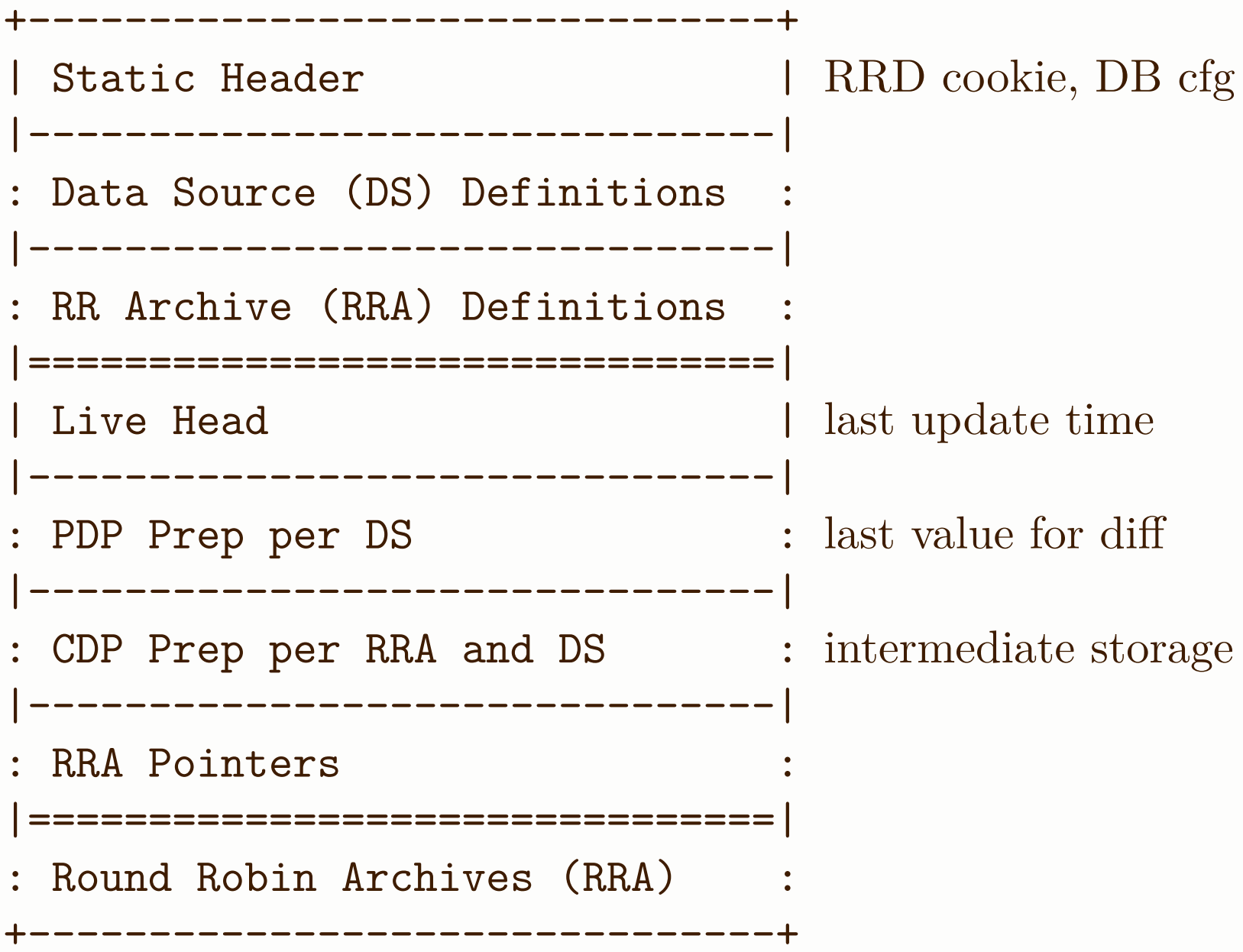
rrd features

- optimized for time-series data
- fixed size rotating data store
- constant on-disk size
- no maintenance
- on the fly consolidation

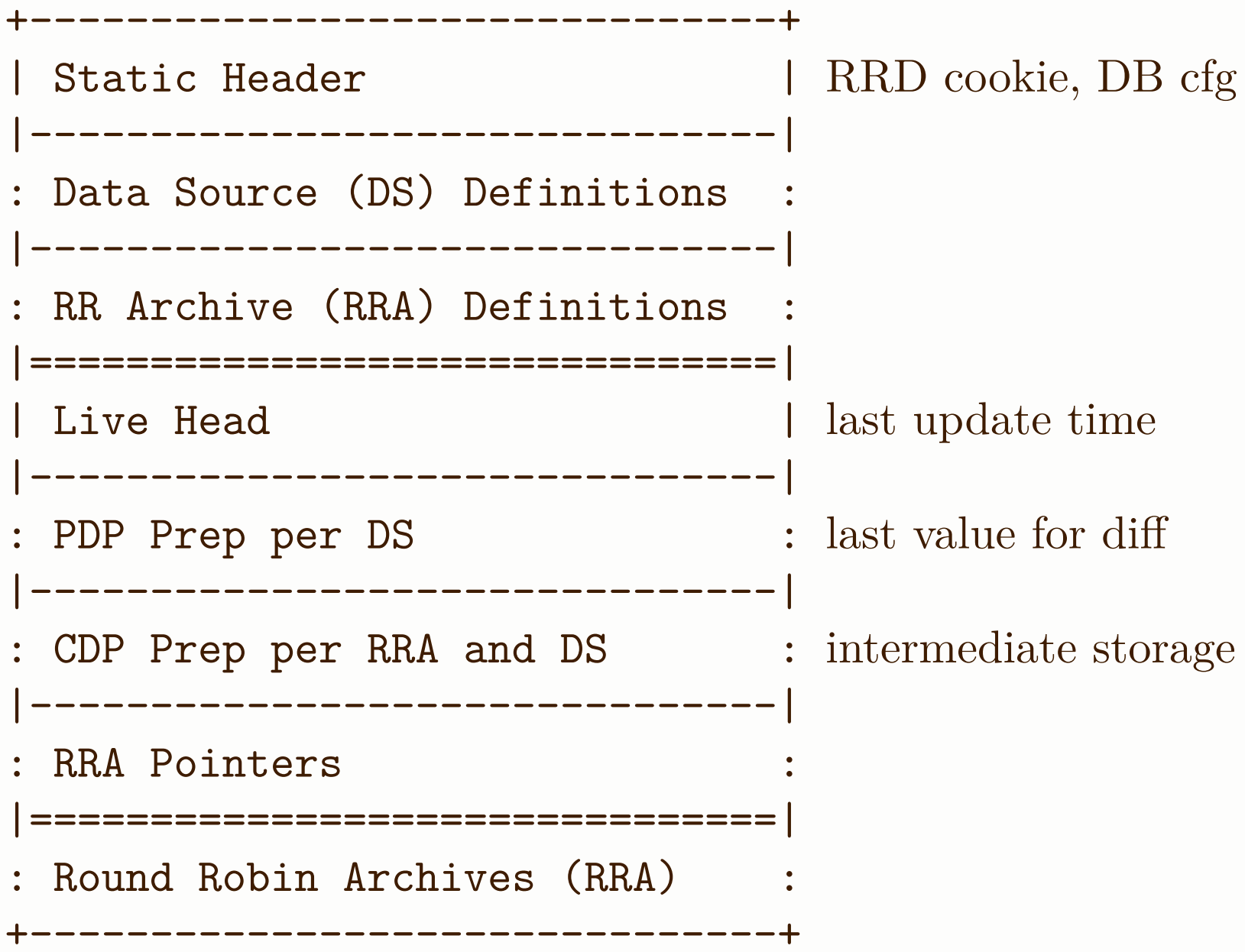
on-disk structure



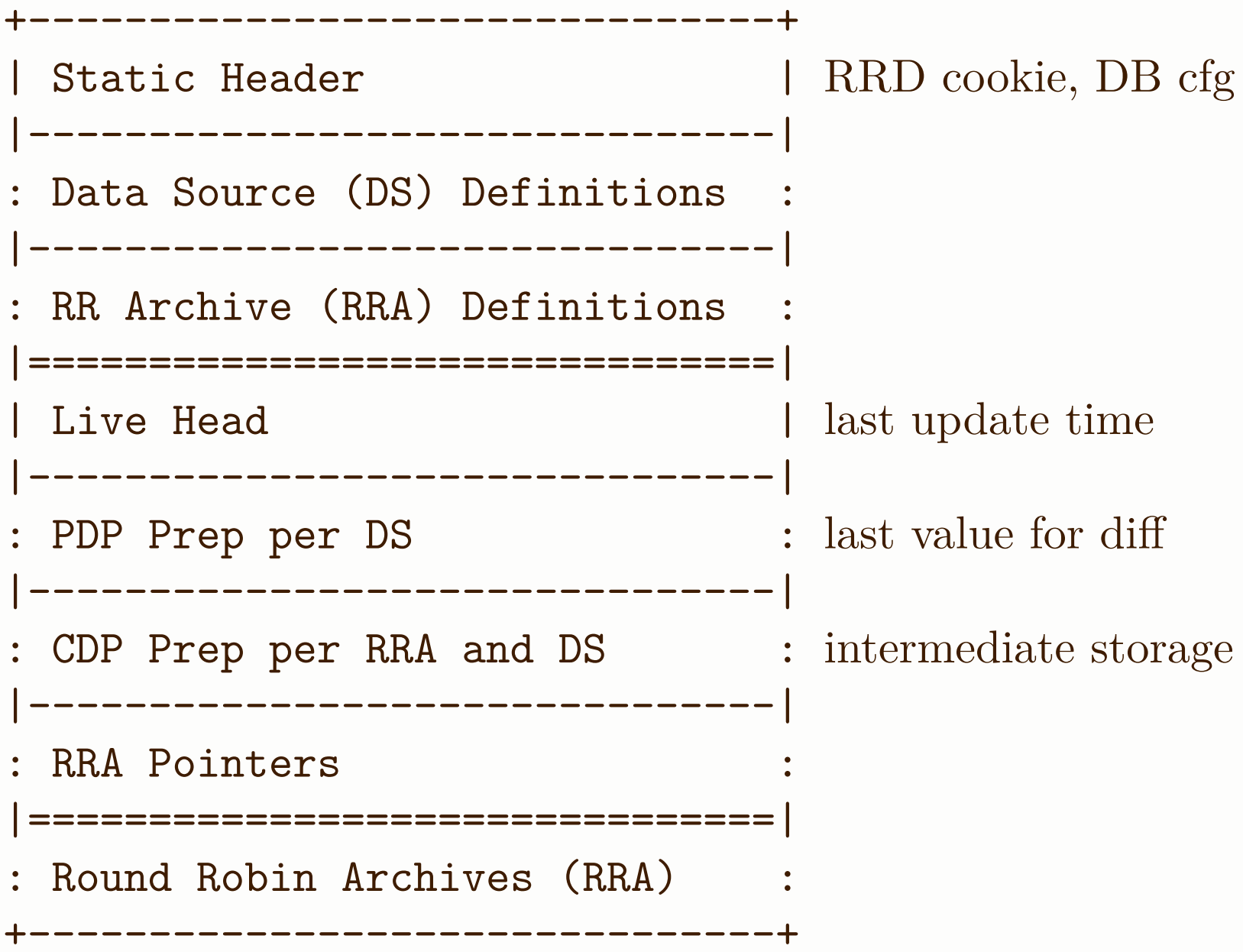
on-disk structure



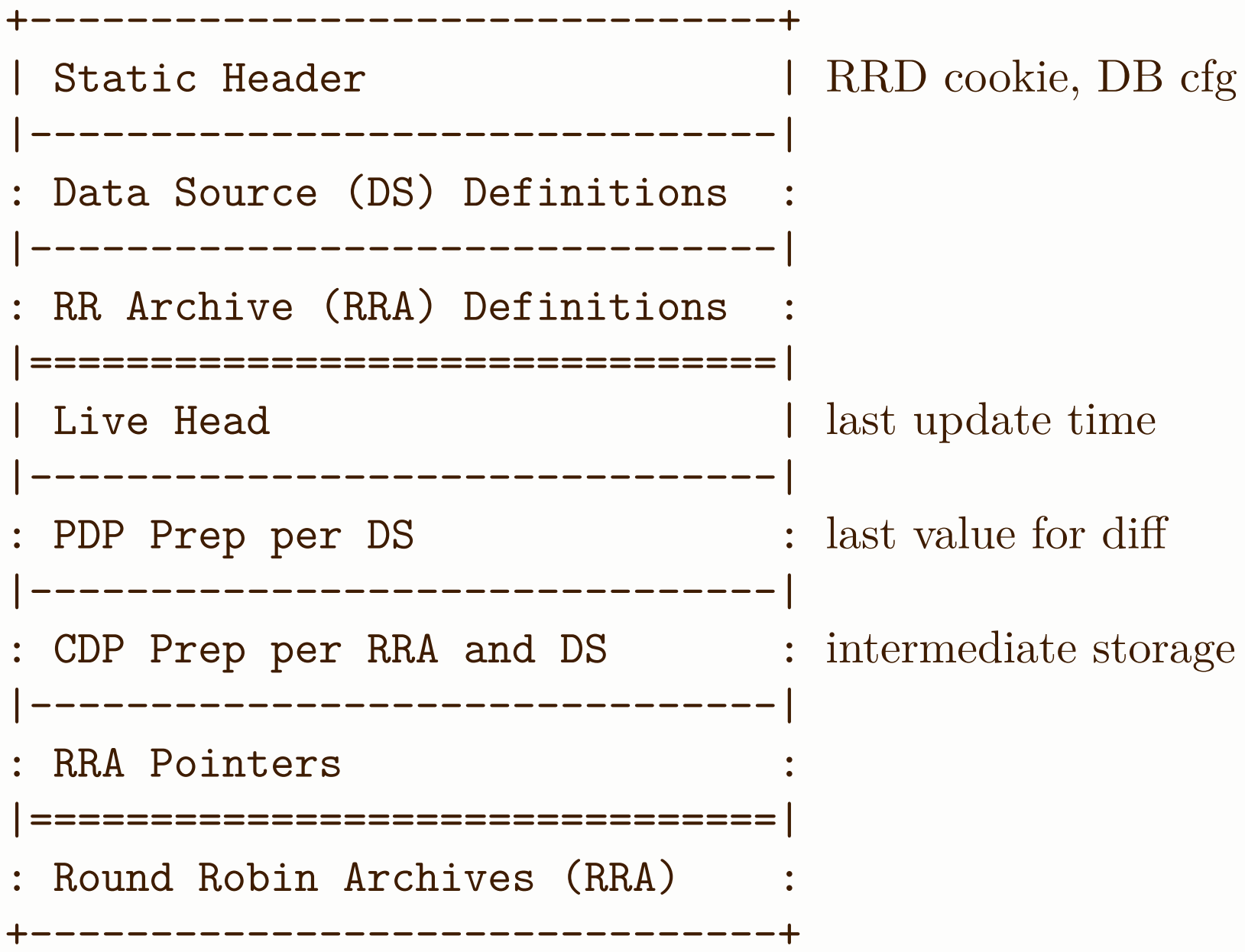
on-disk structure



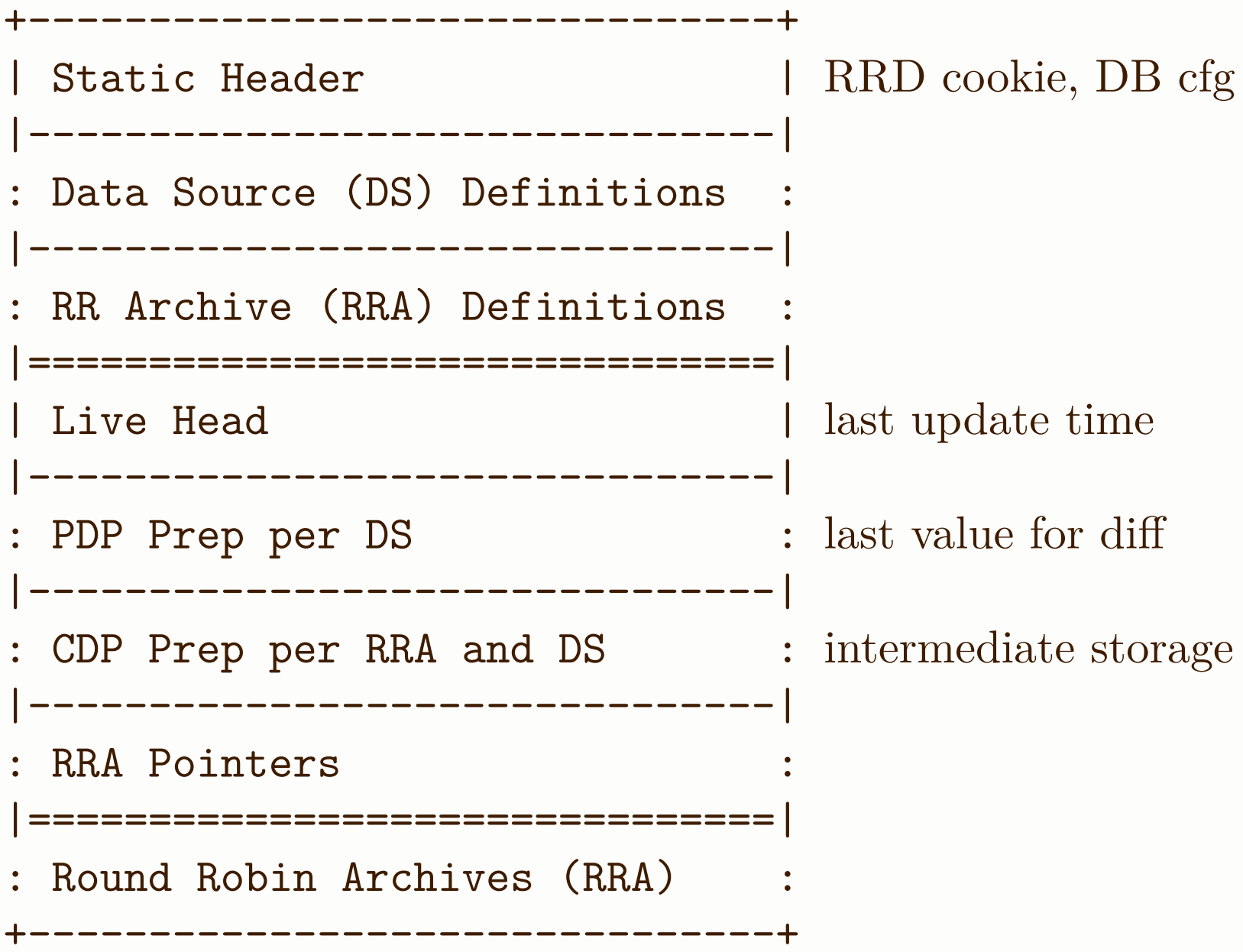
on-disk structure



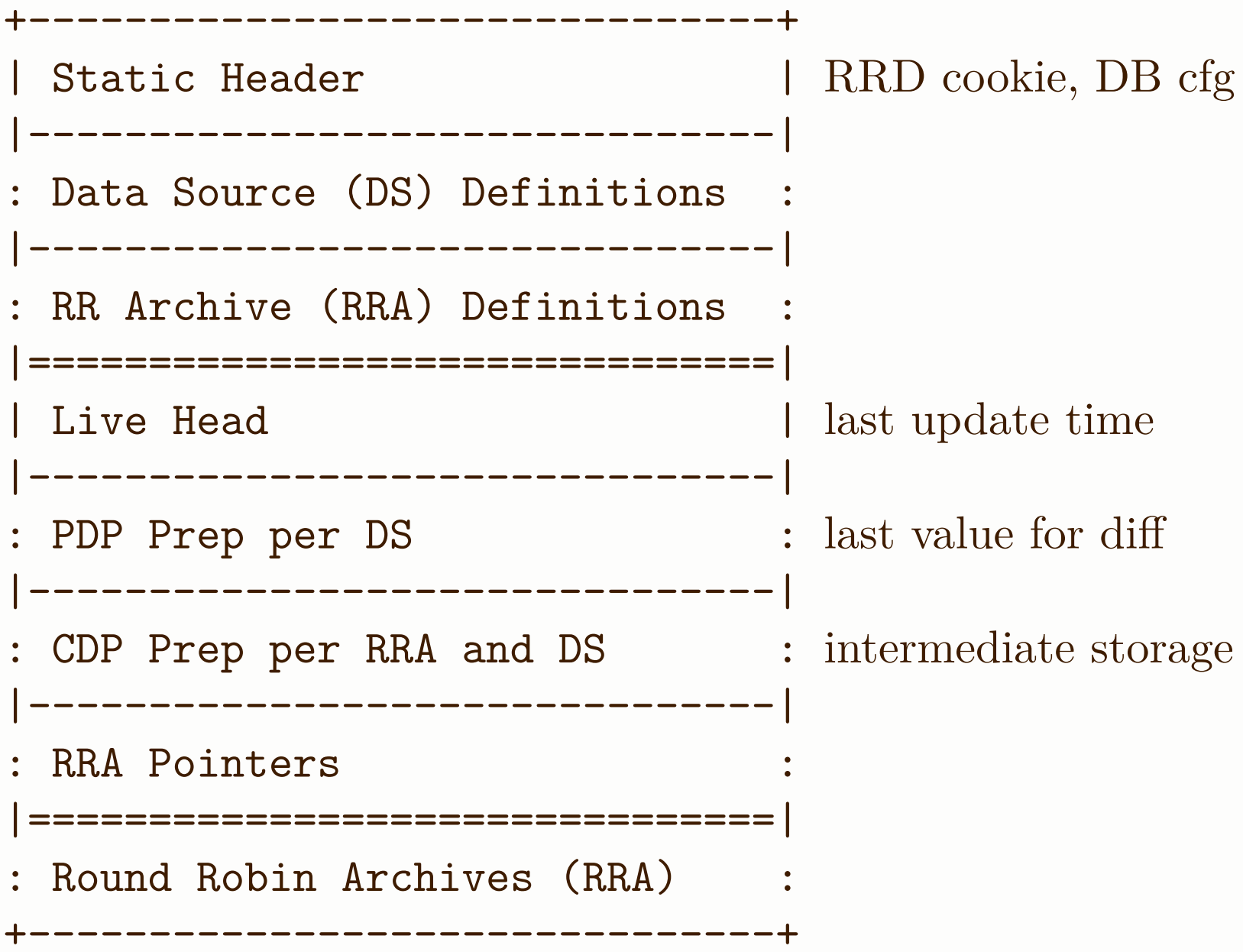
on-disk structure



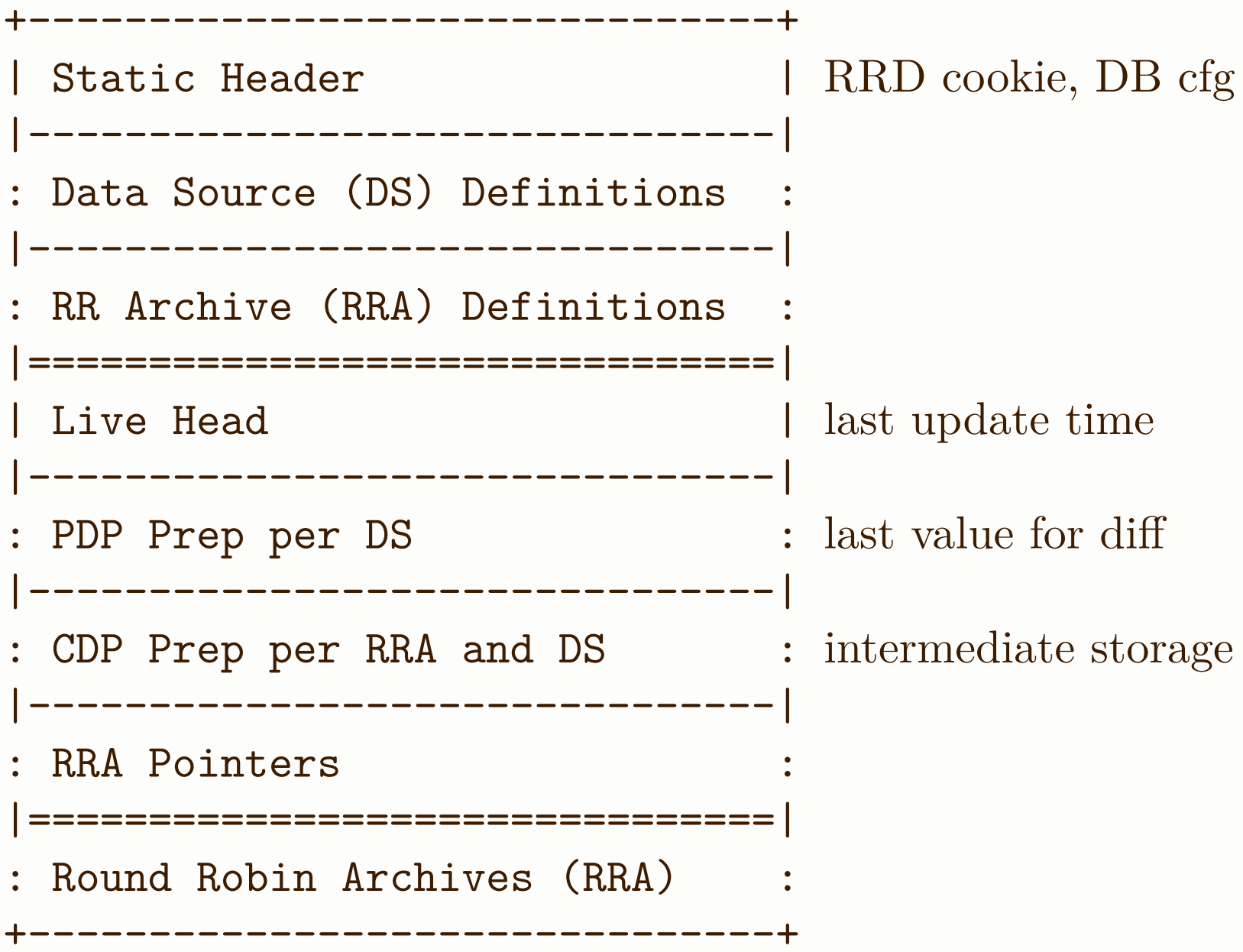
on-disk structure



on-disk structure



on-disk structure



irregular data arrival intervals

```
#!/bin/sh
PATH=/scratch/rrd4/bin:$PATH
R=rrdtool
$R create real.rrd \
   --step=300 \
   --start=1199999699 \
   DS:distance:COUNTER:600:-40:100 \
   RRA:AVERAGE:0.4:1:5

u(){
 $R update real.rrd $1
}

u 1200000000:0
u 1200000150:15
u 1200000310:31
u 1200000640:64
u 1200000910:91
```

database after the irregular updates

```
$R fetch real.rrd -s 1200000000 -e 1200000899 AVERAGE

                           distance

1200000300: 1.0000000000e-01
1200000600: 1.0000000000e-01
1200000900: 1.0000000000e-01
```

- rrdtool re-binning at work
- major difference to a normal db
- all bins contain 1.0
- the time is the 'end-time' of the bin.

database after the irregular updates

```
$R fetch real.rrd -s 1200000000 -e 1200000899 AVERAGE

                           distance

1200000300: 1.0000000000e-01
1200000600: 1.0000000000e-01
1200000900: 1.0000000000e-01
```

- rrdtool re-binning at work
- major difference to a normal db
- all bins contain 1.0
- the time is the 'end-time' of the bin.

database after the irregular updates

```
$R fetch real.rrd -s 1200000000 -e 1200000899 AVERAGE

                           distance

1200000300: 1.0000000000e-01
1200000600: 1.0000000000e-01
1200000900: 1.0000000000e-01
```

- rrdtool re-binning at work
- major difference to a normal db
- all bins contain 1.0
- the time is the 'end-time' of the bin.

database after the irregular updates

```
$R fetch real.rrd -s 1200000000 -e 1200000899 AVERAGE

                           distance

1200000300: 1.0000000000e-01
1200000600: 1.0000000000e-01
1200000900: 1.0000000000e-01
```

- rrdtool re-binning at work
- major difference to a normal db
- all bins contain 1.0
- the time is the 'end-time' of the bin.

optimizing your rrds

- update of multi DS RRD is cheep
- single update interval per RRD
- RRD modification is expensive
- RRD size and update performance are independent
- RRA complexity affects update performance

optimizing your rrds

- update of multi DS RRD is cheep
- single update interval per RRD
- RRD modification is expensive
- RRD size and update performance are independent
- RRA complexity affects update performance

optimizing your rrds

- update of multi DS RRD is cheep
- single update interval per RRD
- RRD modification is expensive
- RRD size and update performance are independent
- RRA complexity affects update performance

optimizing your rrds

- update of multi DS RRD is cheep
- single update interval per RRD
- RRD modification is expensive
- RRD size and update performance are independent
- RRA complexity affects update performance

optimizing your rrds

- update of multi DS RRD is cheep
- single update interval per RRD
- RRD modification is expensive
- RRD size and update performance are independent
- RRA complexity affects update performance

fetching data

fetch is for reading data from an rrd

```
1       RRA:AVERAGE:0.5:1:2 \
2       RRA:AVERAGE:0.5:2:3
```

- one RRA with two 300s entries
- one RRA with three 600s entries

fetching data

fetch is for reading data from an rrd

```
1       RRA:AVERAGE:0.5:1:2 \
2       RRA:AVERAGE:0.5:2:3
```

- one RRA with two 300s entries
- one RRA with three 600s entries

playing catch with fetch

first pull 300 seconds

```
> rrdtool fetch x.rrd -r 300 \
  -s 1200000600 -e 1200000900 AVERAGE

1200000900: 4.0000000000e+01
1200001200: 5.0000000000e+01
```

then pull 900 seconds

```
> rrdtool fetch x.rrd -r300 \
  -s 1200000000 -e 1200000900 AVERAGE

1200000600: 2.5000000000e+01
1200001200: 4.5000000000e+01
```

fetch recap

- looking for complete coverage
- resolution is only a suggestion
- time stamp in output marks the END of the period
- end-time differences caused problems
- since 1.3, only the start-time is relevant for coverage
- outside the rra everything is NaN

fetch recap

- looking for complete coverage
- resolution is only a suggestion
- time stamp in output marks the END of the period
- end-time differences caused problems
- since 1.3, only the start-time is relevant for coverage
- outside the rra everything is NaN

fetch recap

- looking for complete coverage
- resolution is only a suggestion
- time stamp in output marks the END of the period
- end-time differences caused problems
- since 1.3, only the start-time is relevant for coverage
- outside the rra everything is NaN

fetch recap

- looking for complete coverage
- resolution is only a suggestion
- time stamp in output marks the END of the period
- end-time differences caused problems
- since 1.3, only the start-time is relevant for coverage
- outside the rra everything is NaN

fetch recap

- looking for complete coverage
- resolution is only a suggestion
- time stamp in output marks the END of the period
- end-time differences caused problems
- since 1.3, only the start-time is relevant for coverage
- outside the rra everything is NaN

fetch recap

- looking for complete coverage
- resolution is only a suggestion
- time stamp in output marks the END of the period
- end-time differences caused problems
- since 1.3, only the start-time is relevant for coverage
- outside the rra everything is NaN

Graphing

rrdgraph syntax 101

for graph command syntax, there are two basic rules:

1. `--options` start with a double dash
2. graphing instructions start with a letter

```
rrdtool graph output
   DEF:var=rrd:DS:AVARAGE
   LINE:var#hex-rgb-color:Comment
```

`DEF` and `LINE` are *graphing instructions.*

rrdgraph syntax 101

for graph command syntax, there are two basic rules:

1. `--options` start with a double dash
2. graphing instructions start with a letter

```
rrdtool graph output
   DEF:var=rrd:DS:AVARAGE
   LINE:var#hex-rgb-color:Comment
```

`DEF` and `LINE` are *graphing instructions.*

rrdgraph syntax 101

for graph command syntax, there are two basic rules:

1. `--options` start with a double dash
2. graphing instructions start with a letter

```
rrdtool graph output
   DEF:var=rrd:DS:AVARAGE
   LINE:var#hex-rgb-color:Comment
```

`DEF` and `LINE` are *graphing instructions.*

rrdgraph syntax 101

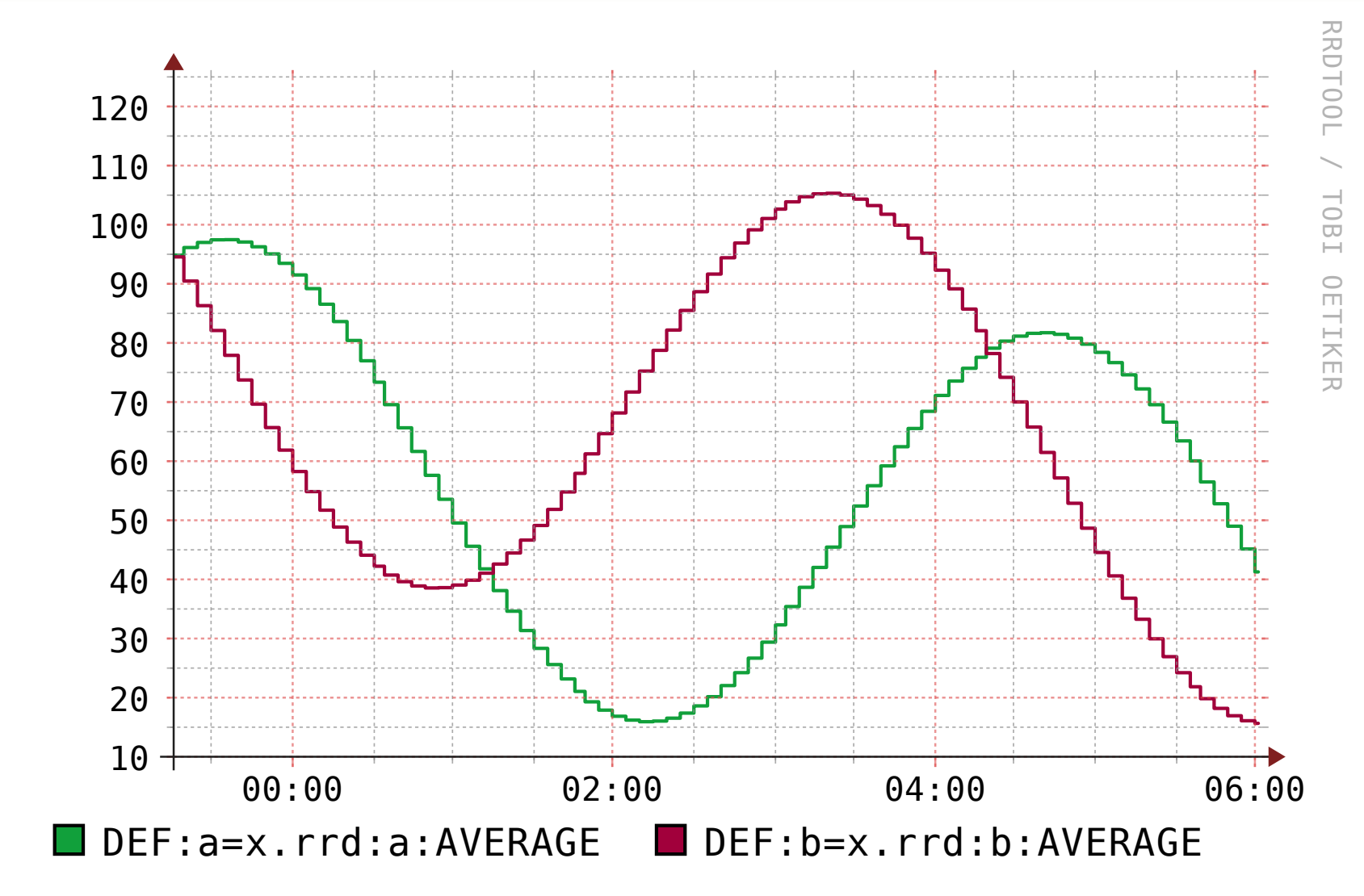
for graph command syntax, there are two basic rules:

1. `--options` start with a double dash
2. graphing instructions start with a letter

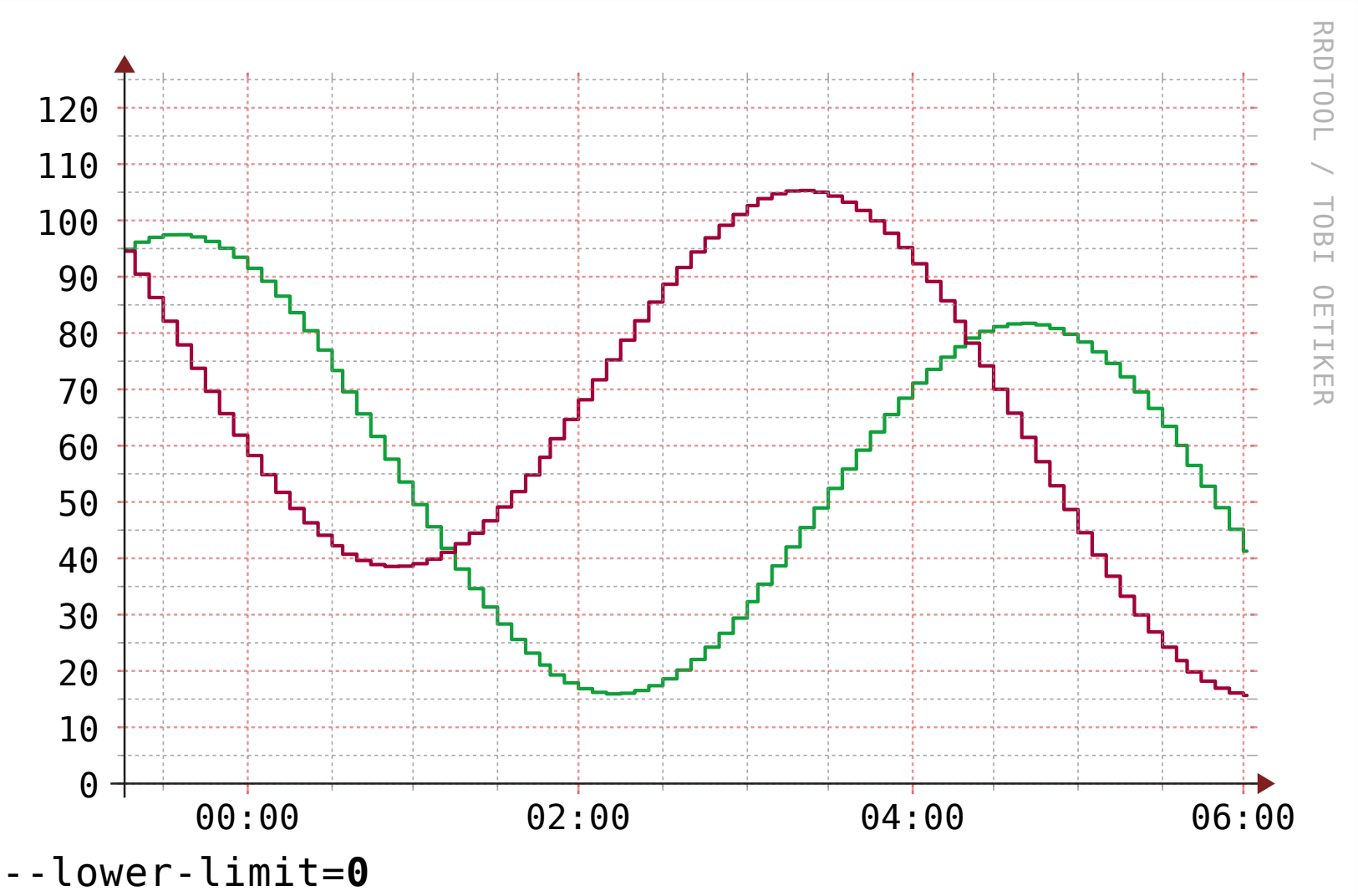
```
rrdtool graph output
   DEF:var=rrd:DS:AVARAGE
   LINE:var#hex-rgb-color:Comment
```

`DEF` and `LINE` are *graphing instructions.*

normal line

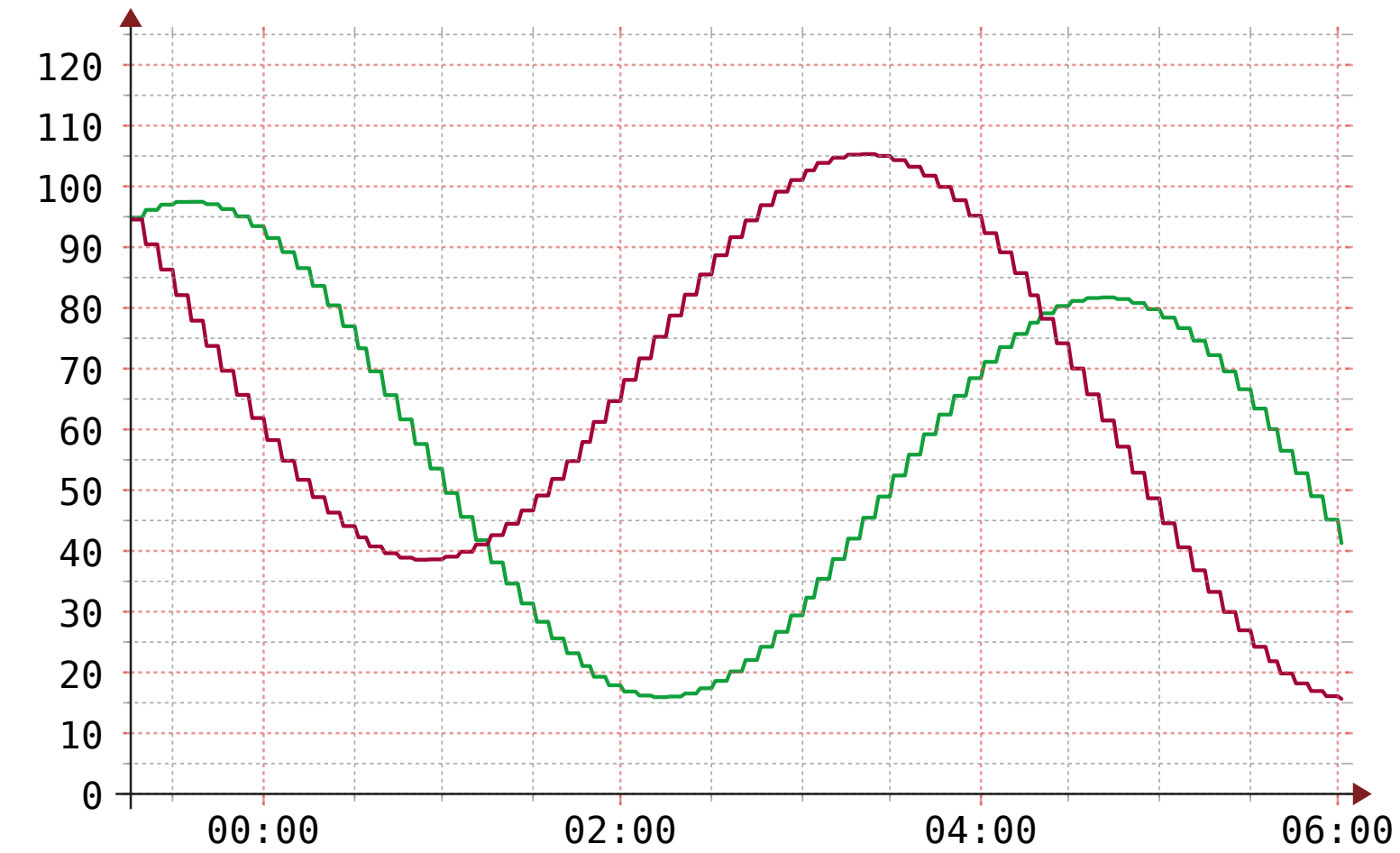

120
110
100
90
80
70
60
50
40
30
20
10
00:00
02:00
04:00
06:00
DEF:a=x.rrd:a:AVERAGE
DEF:b=x.rrd:b:AVERAGE
RRDTOOL / TOBI OETIKER

lower limit



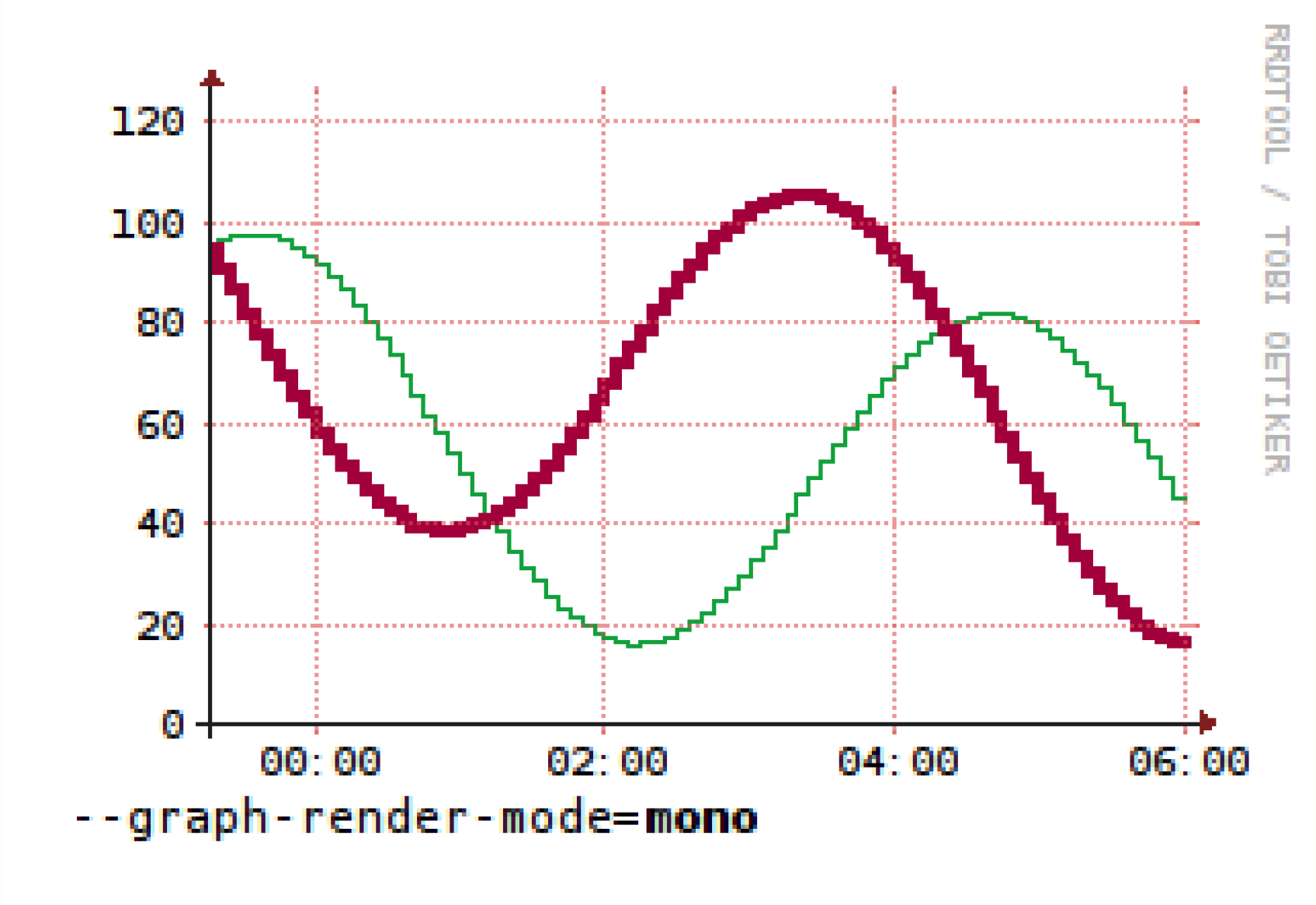
`--lower-limit=0`

slope mode

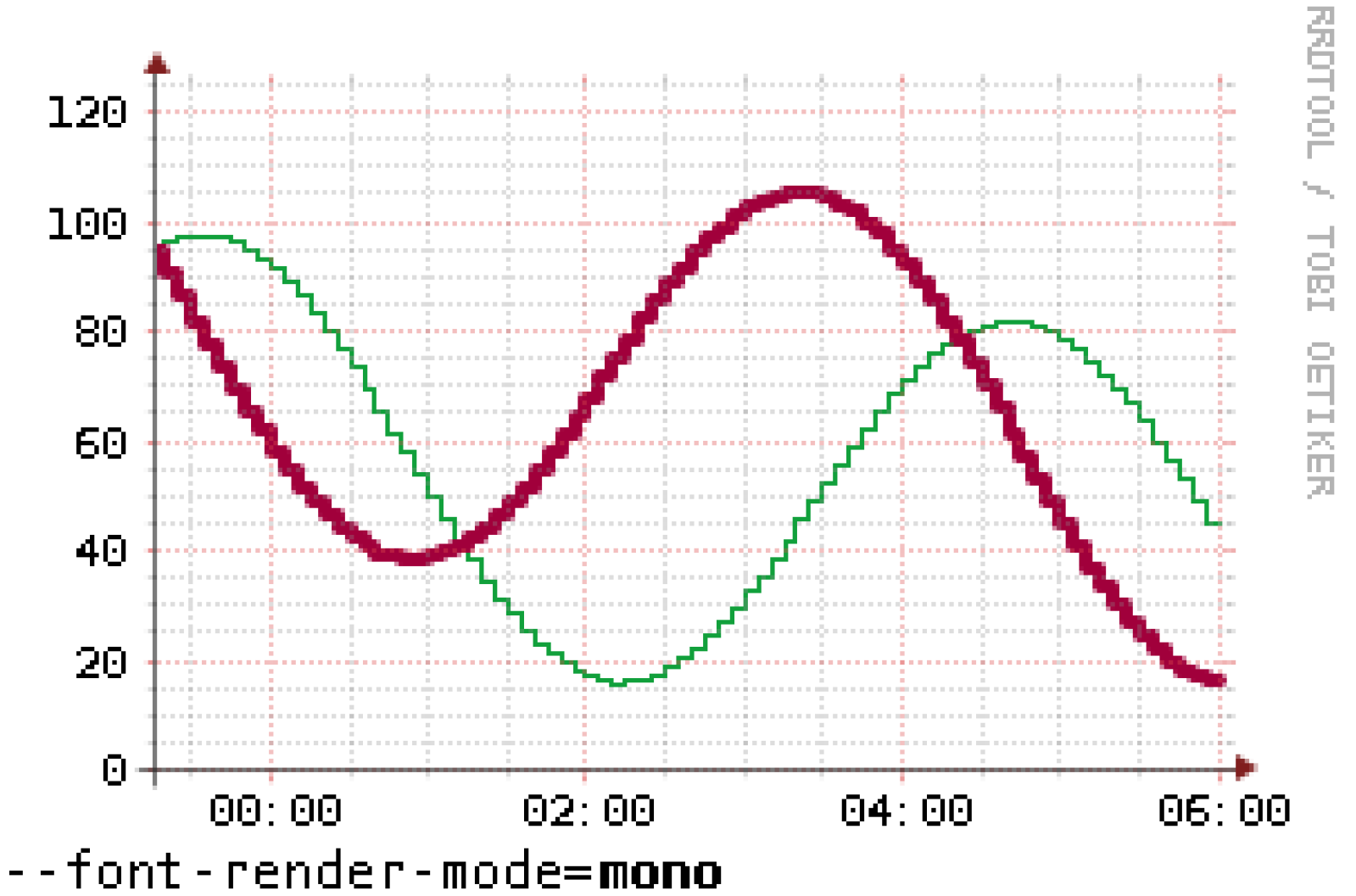


--slope-mode

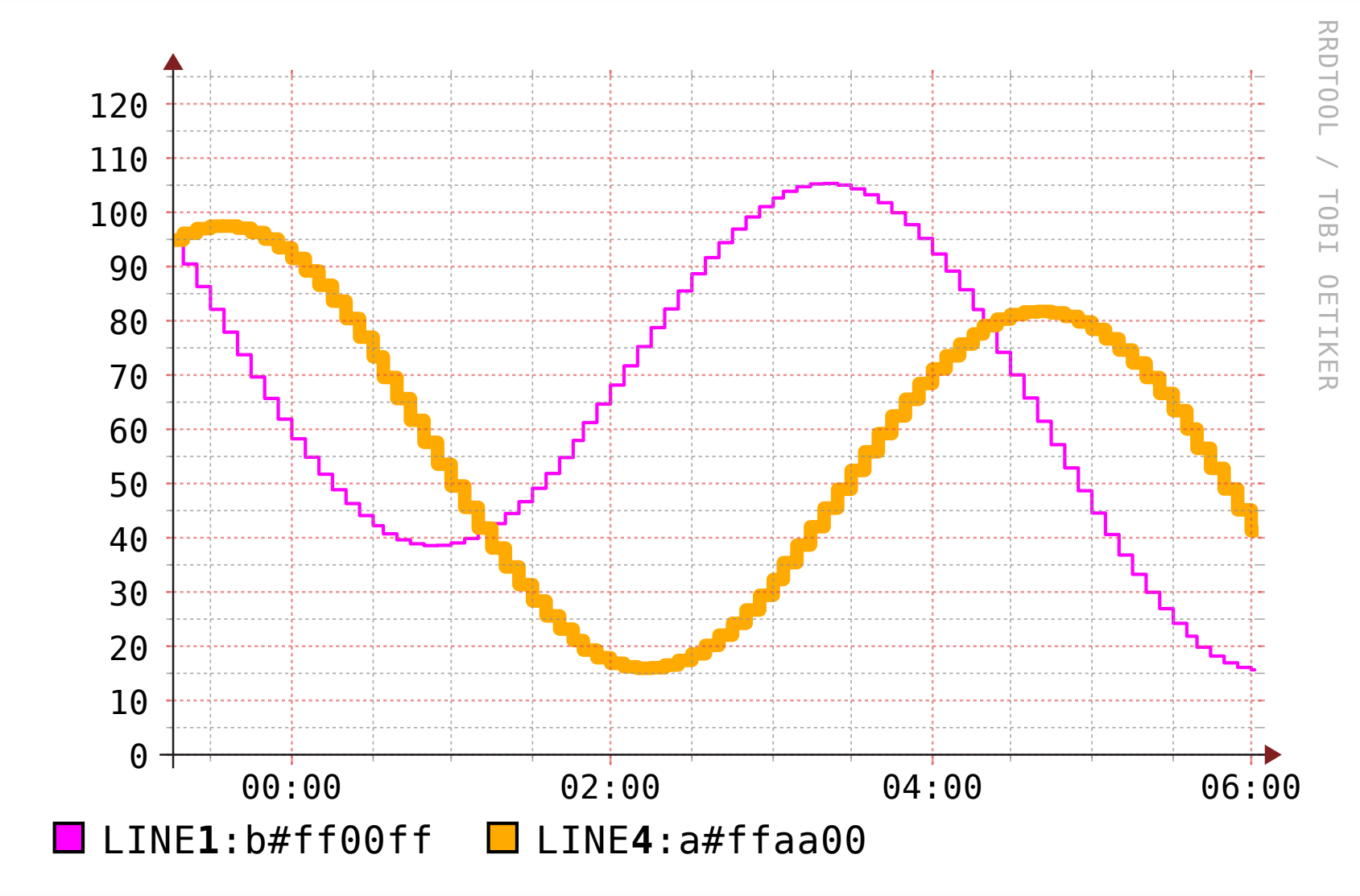
anti-anti-aliasing: graph



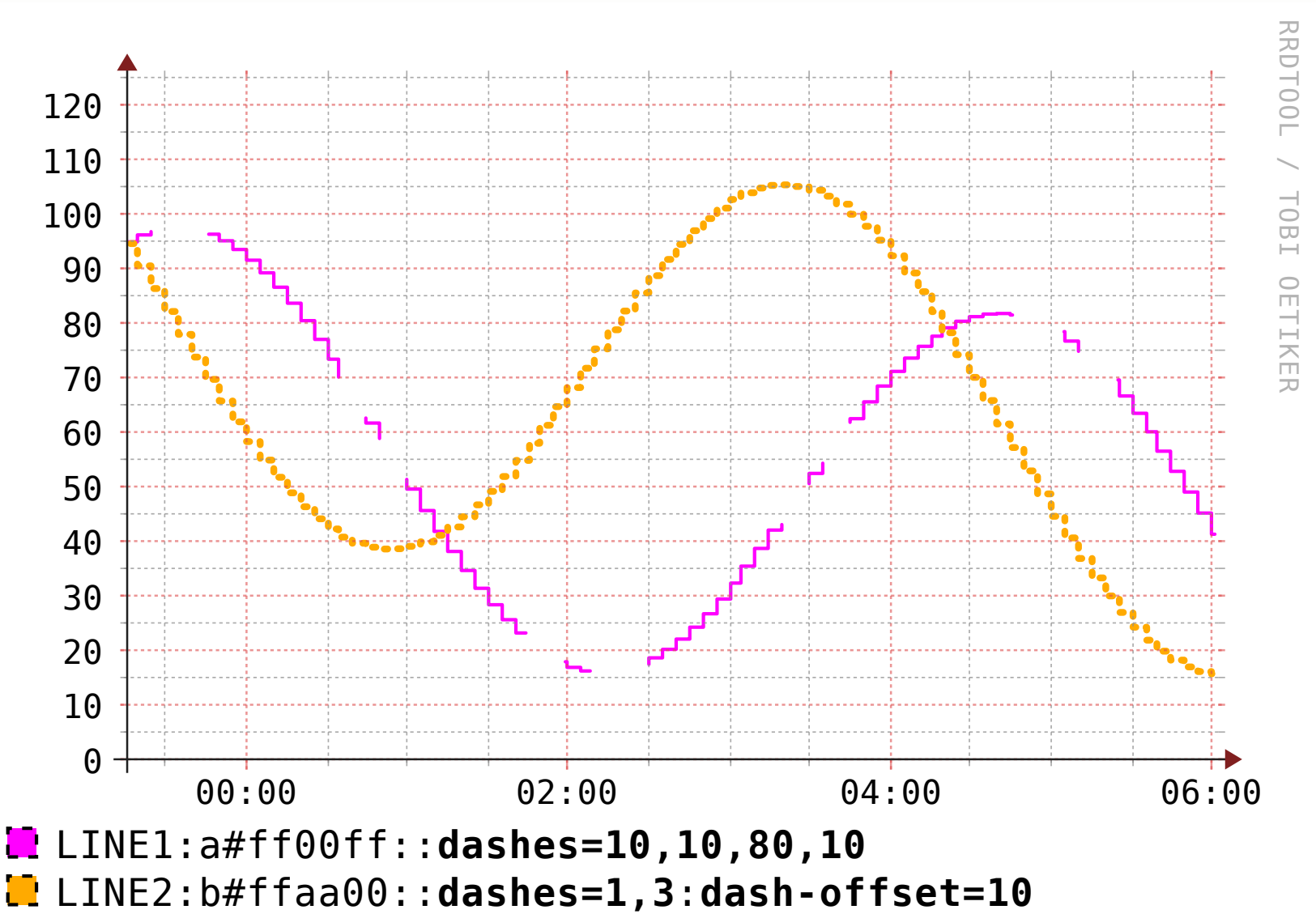
anti-anti-aliasing: font



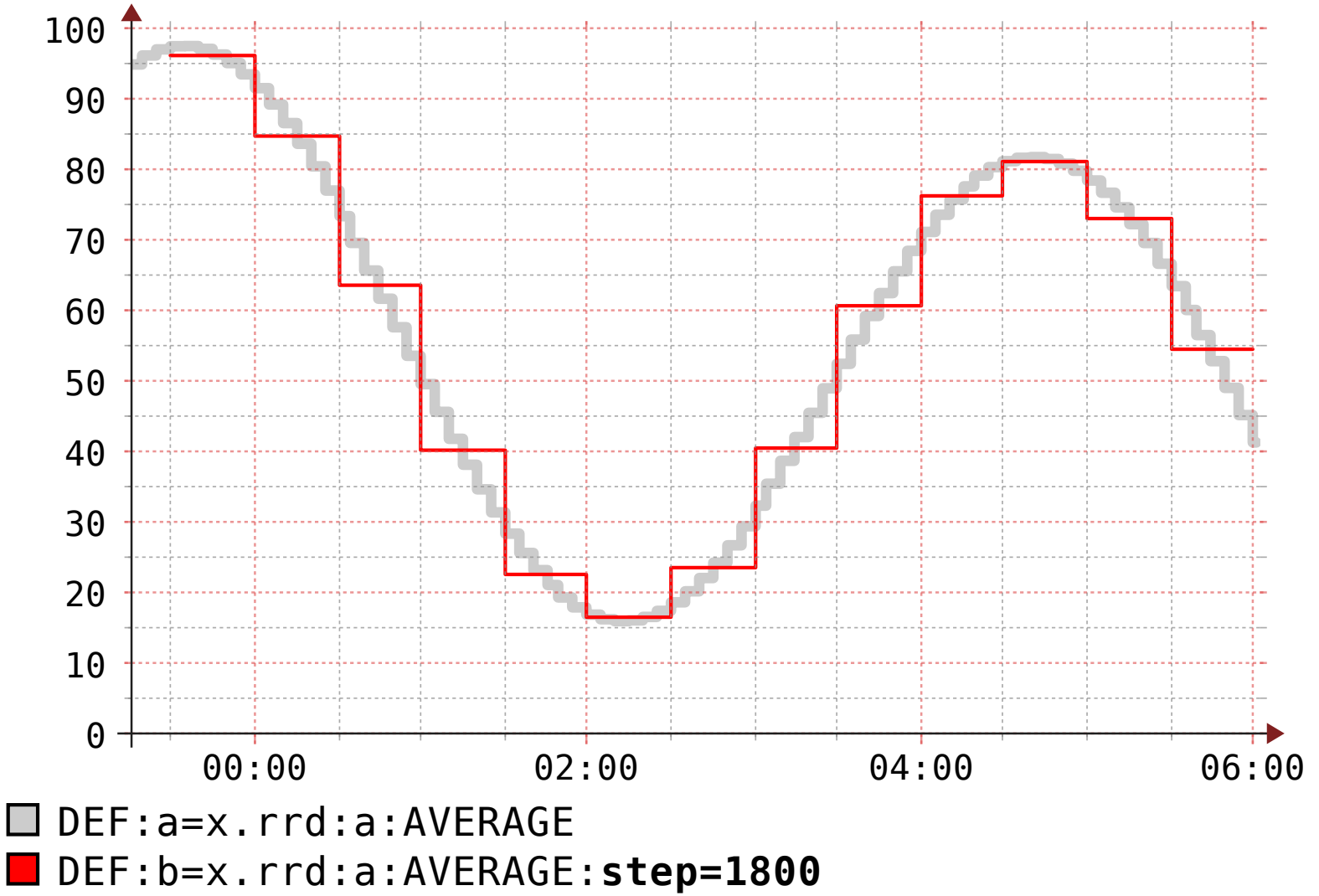
line width


120
110
100
90
80
70
60
50
40
30
20
10
0
00:00
02:00
04:00
06:00
LINE1:b#ff00ff
LINE4:a#ffaa00
RRDTOOL / TOBI OETIKER

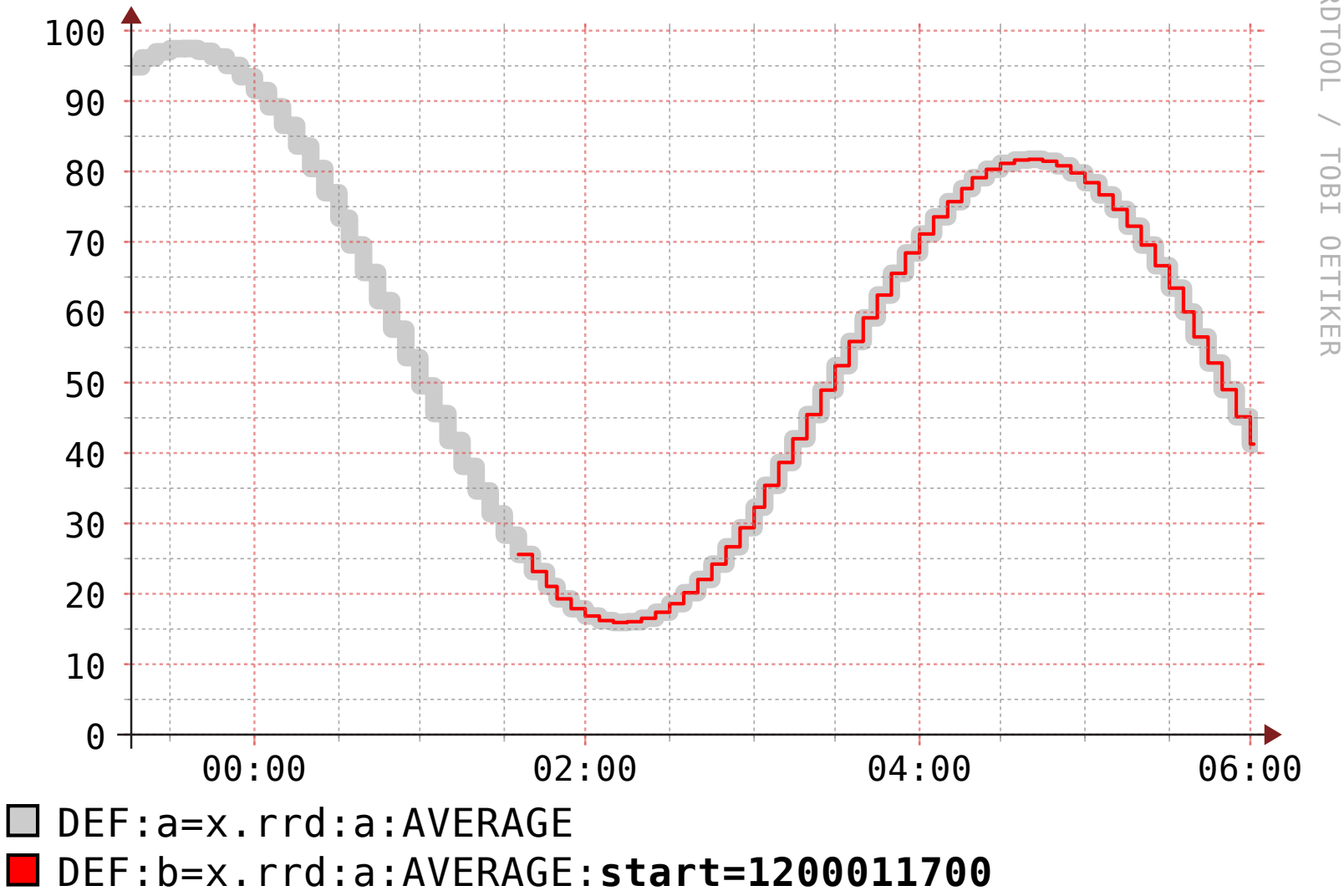
dashed line


RRDTOOL / TOBI OETIKER
120
110
100
90
80
70
60
50
40
30
20
10
0
00:00
02:00
04:00
06:00
LINE1:a#ff00ff::dashes=10,10,80,10
LINE2:b#ffaa00::dashes=1,3:dash-offset=10

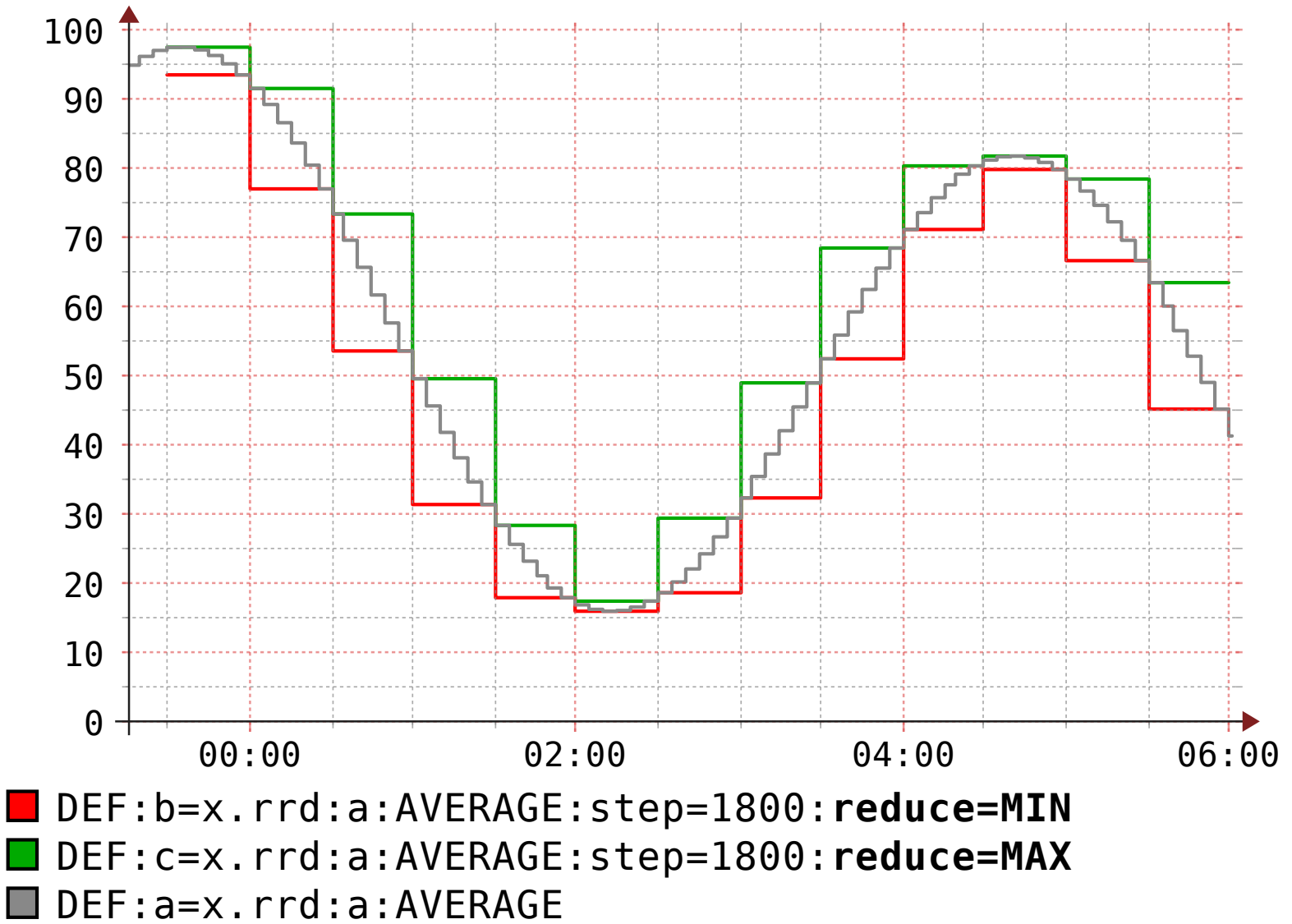
DEF with :step



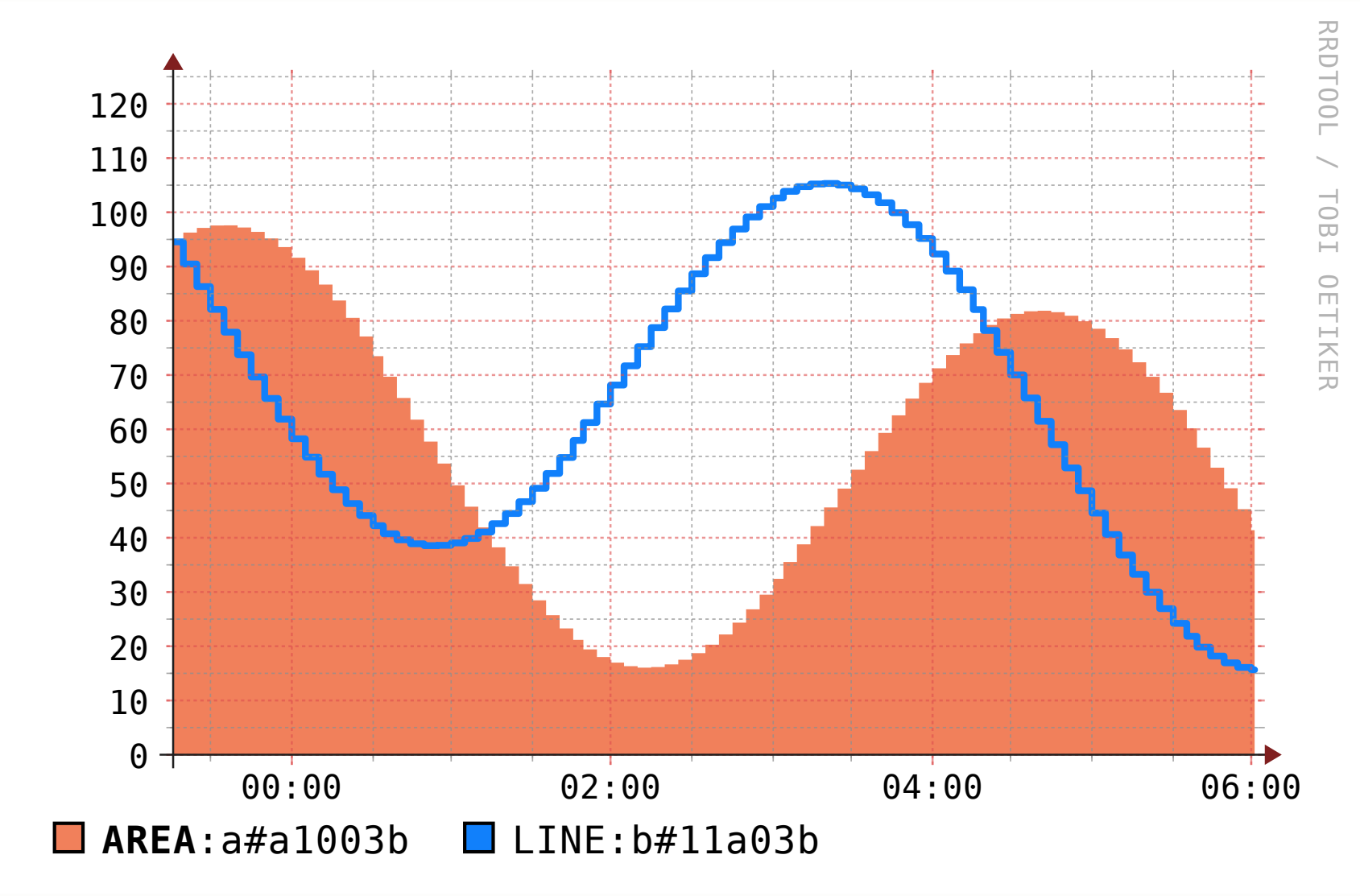
DEF with :start



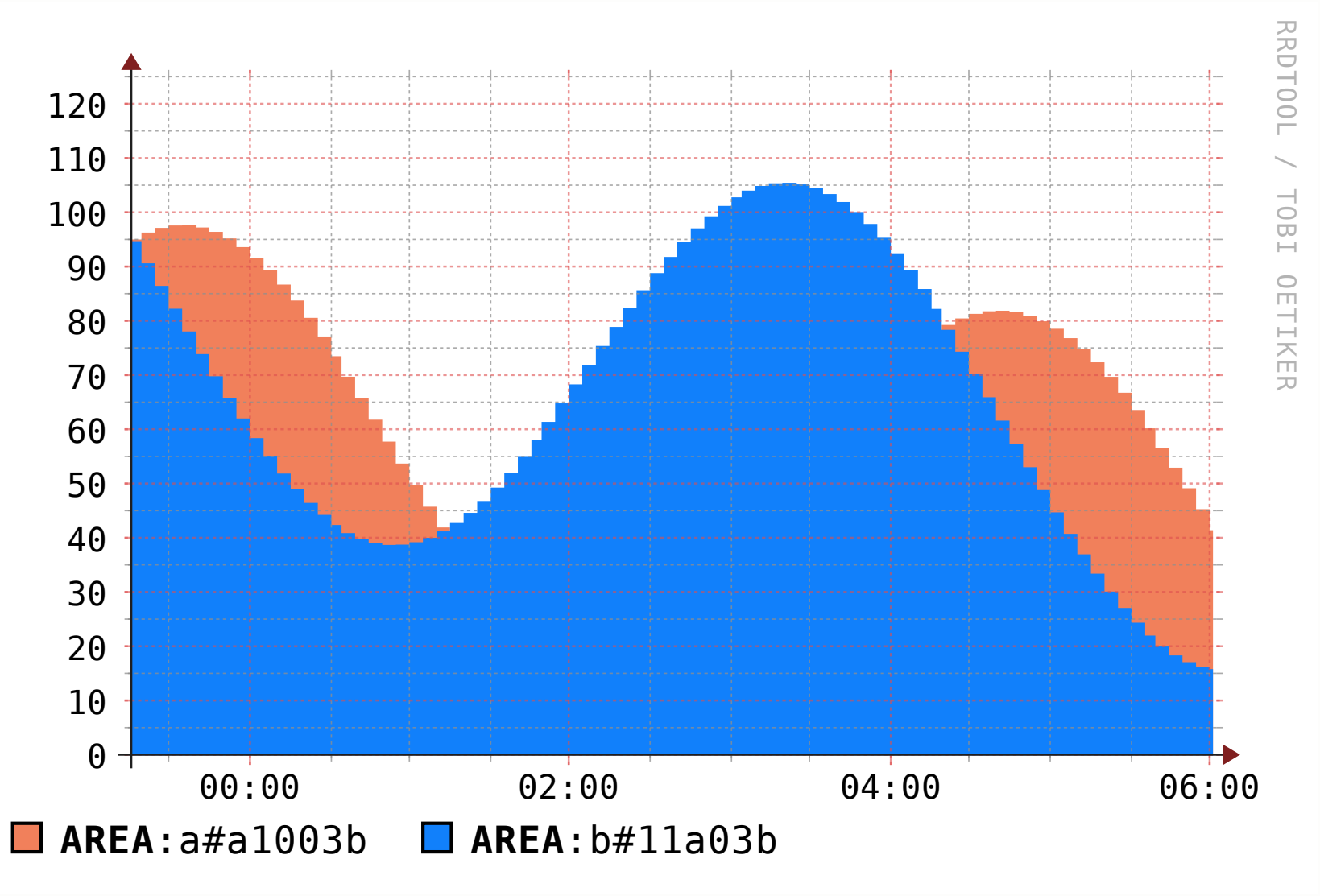
DEF with :reduce



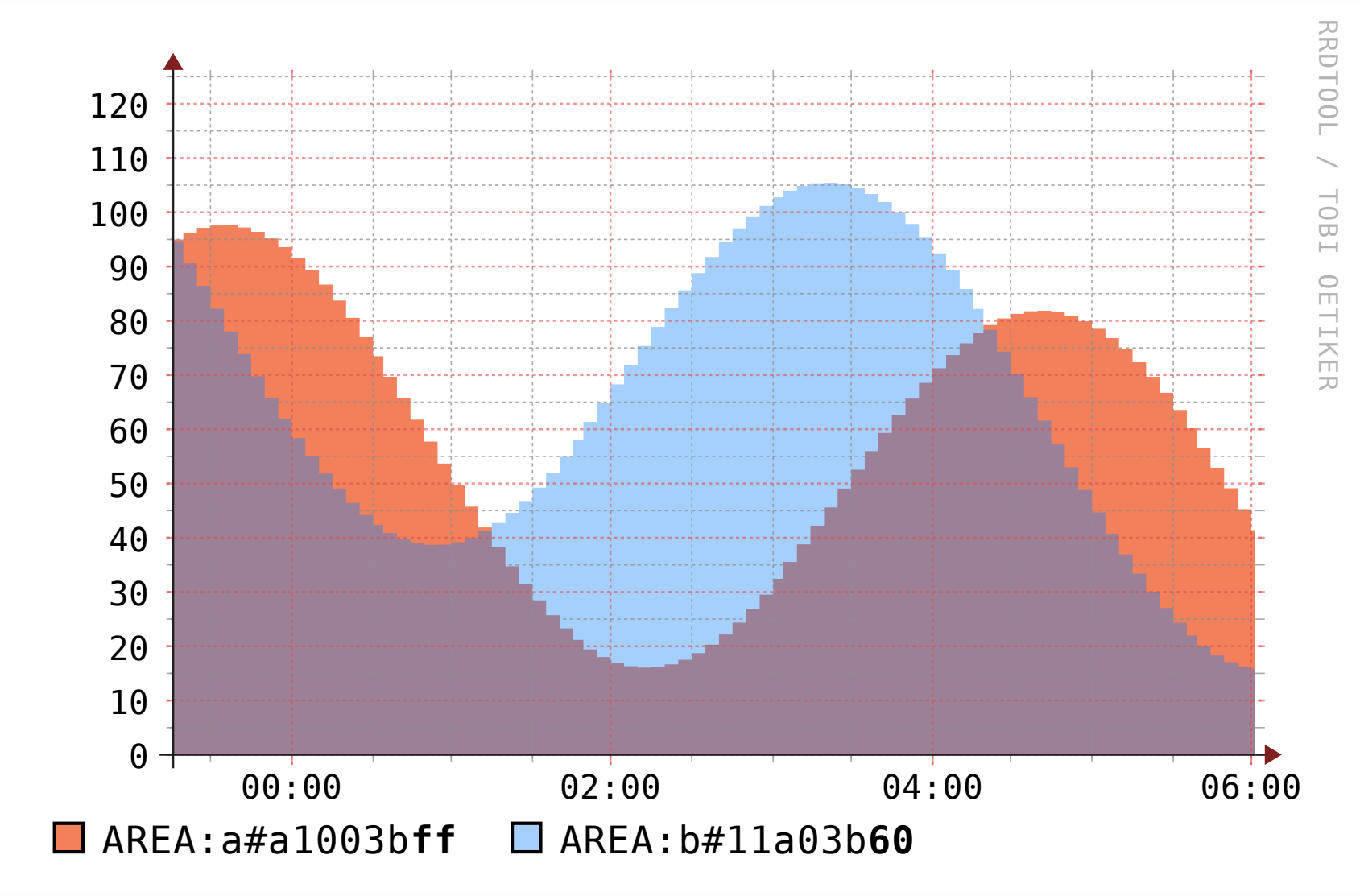
AREA simple


120
110
100
90
80
70
60
50
40
30
20
10
0
00:00
02:00
04:00
06:00
AREA:a#a1003b
LINE:b#11a03b
RRDTOOL / TOBI OETIKER

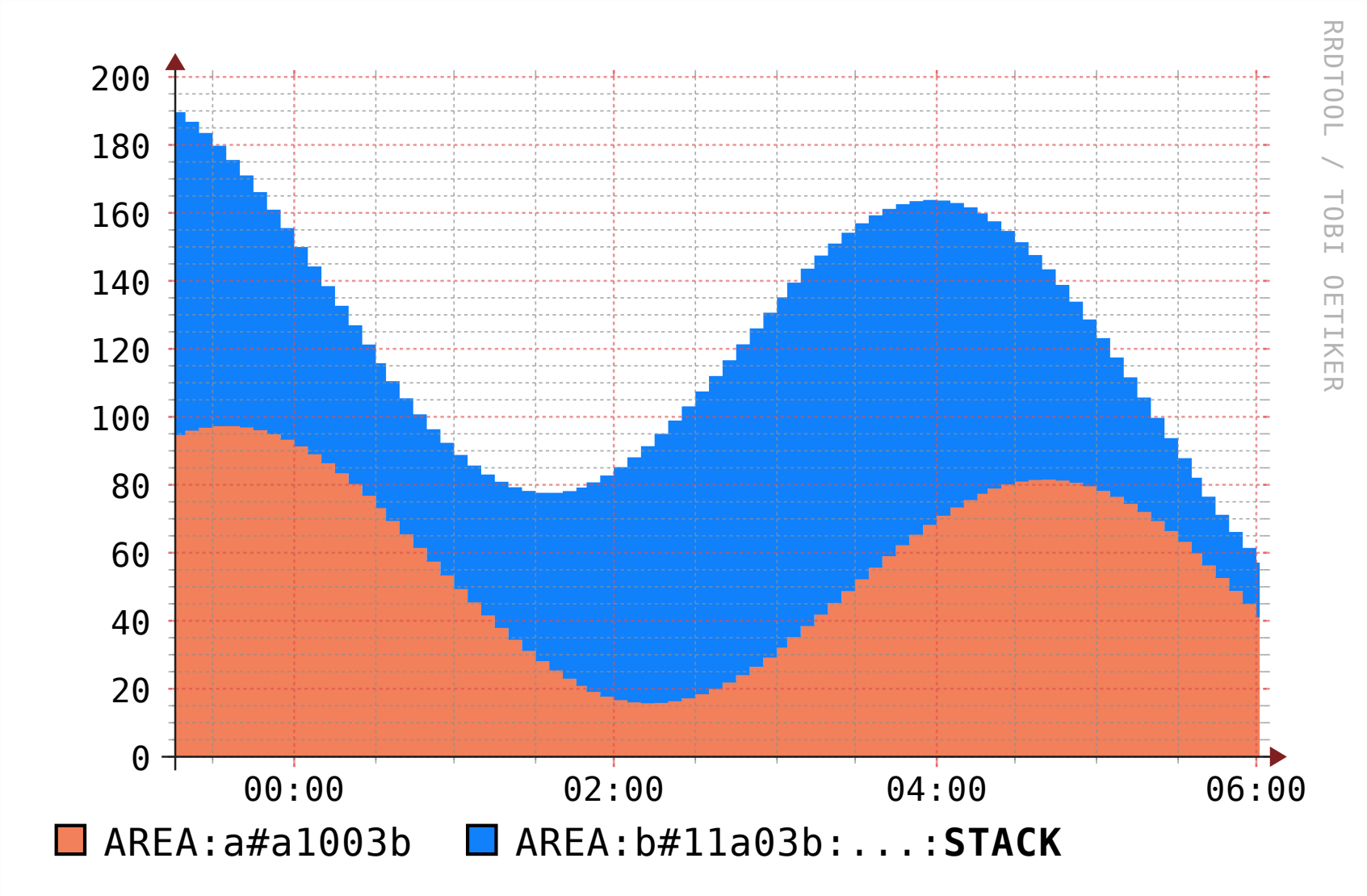
two AREAs


120
110
100
90
80
70
60
50
40
30
20
10
0
00:00
02:00
04:00
06:00
AREA:a#a1003b
AREA:b#11a03b
RRDTOOL / TOBI OETIKER

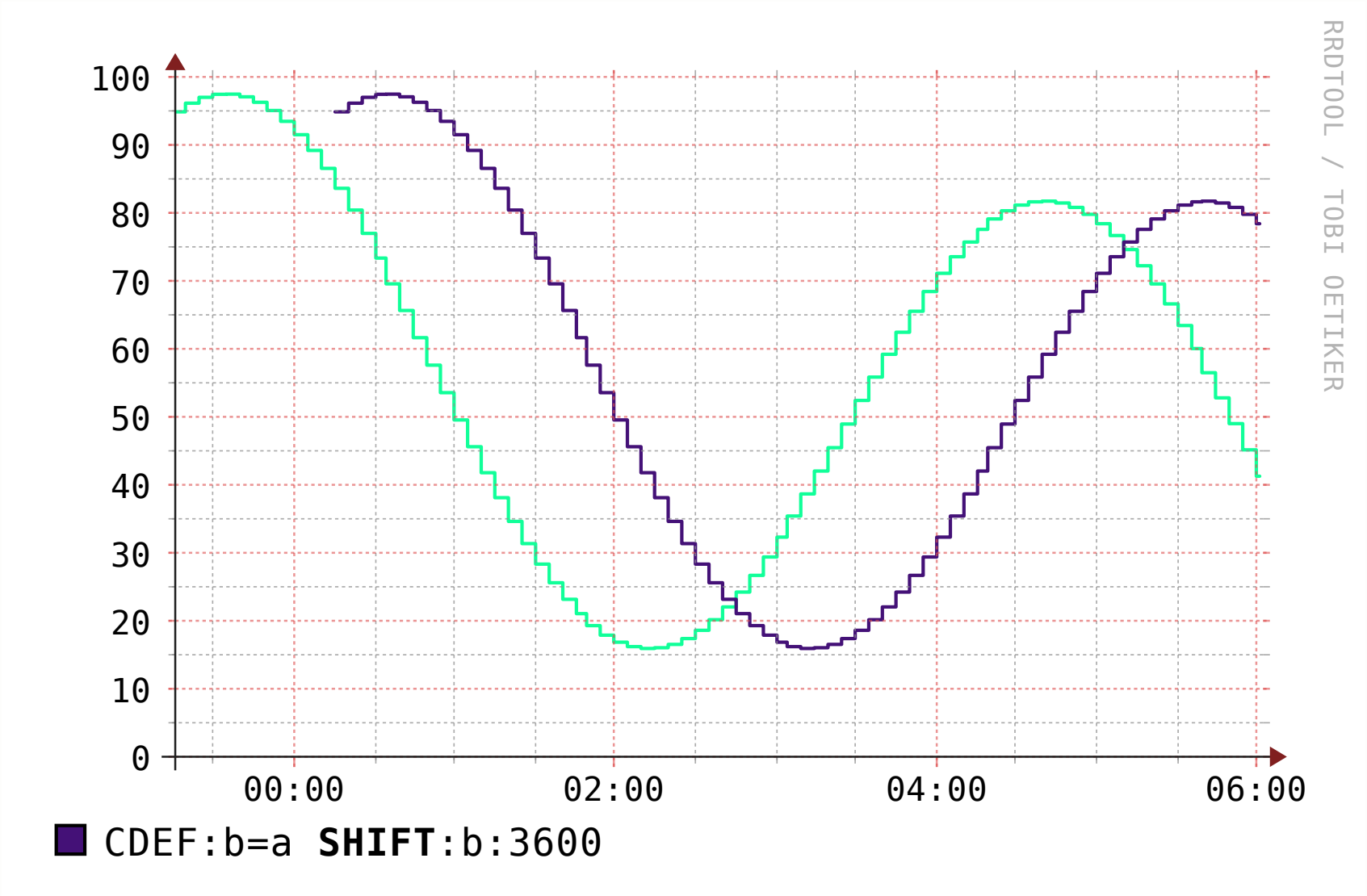
transparent AREA



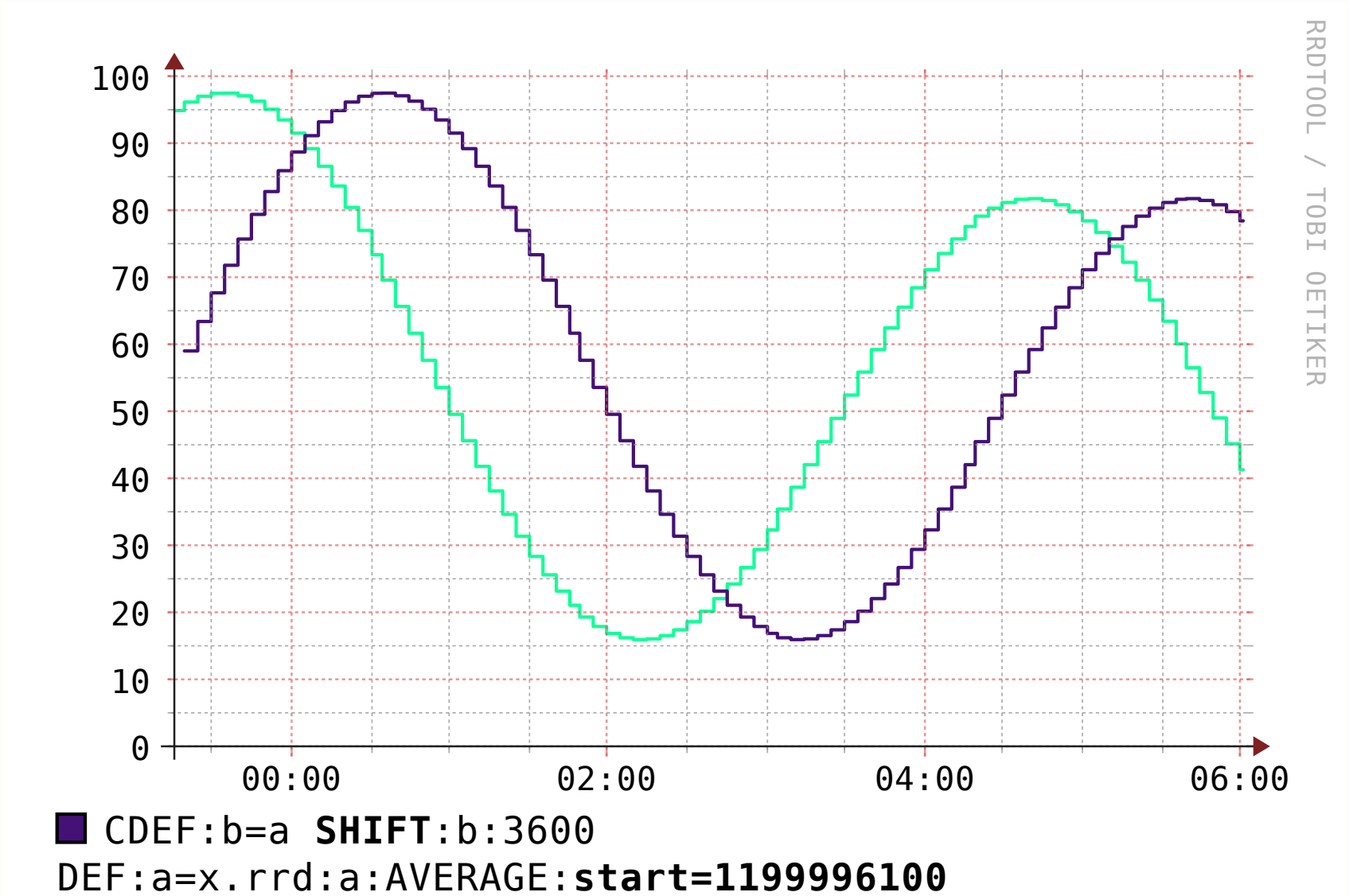
stacked AREA



time shift


100
90
80
70
60
50
40
30
20
10
0
00:00
02:00
04:00
06:00
RRDTOOL / TOBI OETIKER
CDEF:b=a SHIFT:b:3600

shifting with extra data



Revers Polish Notation (RPN) Math

RPN basics: Step 0

$$15 + 23 = 38$$

```
1: NAN
2: NAN
3: NAN
```

RPN basics: Step 1

$$\mathbf{15} + 23 = 38$$

```
[15]          1: 15
              2: NAN
              3: NAN
```

RPN basics: Step 2

$15 + \mathbf{23} = 38$

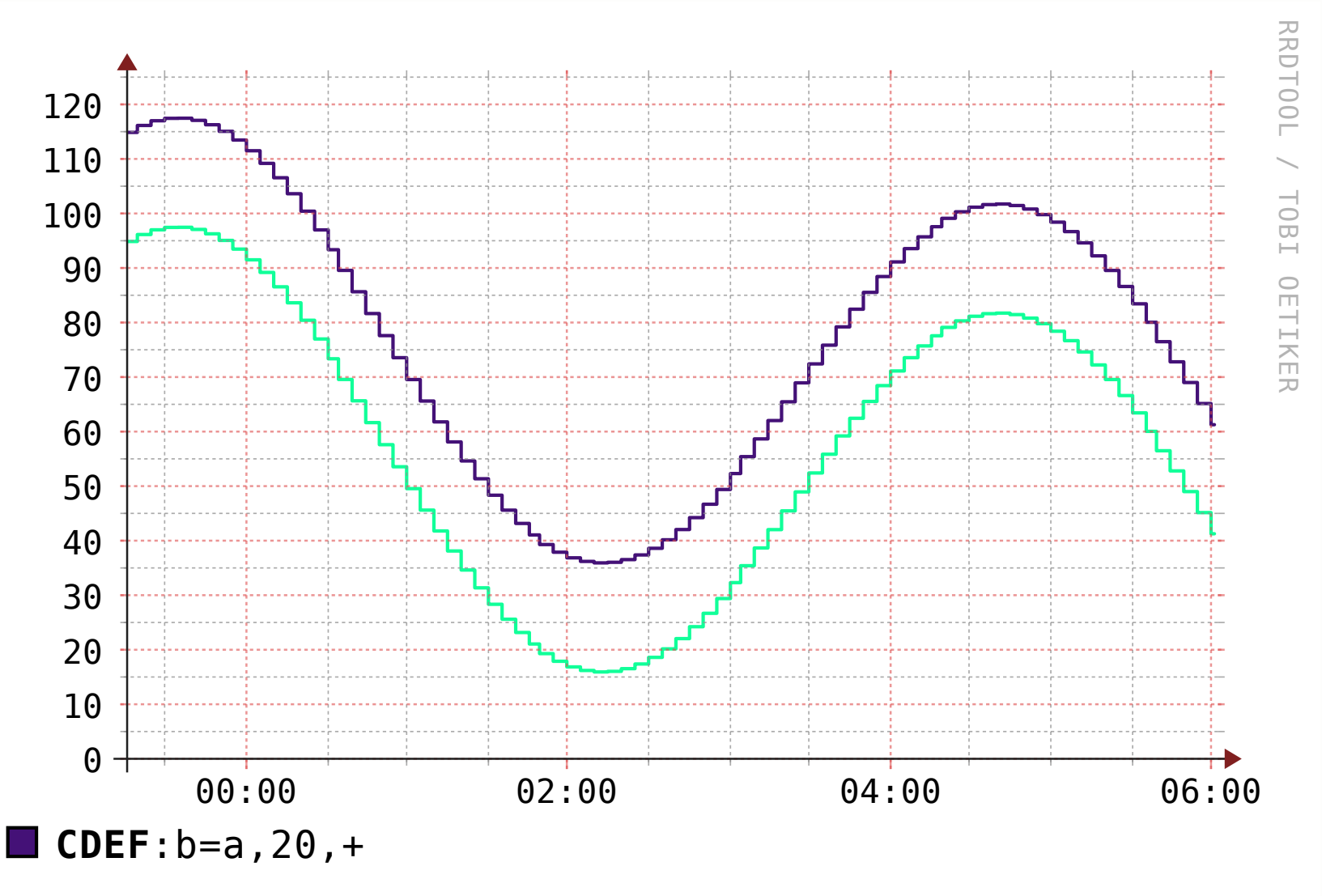
```
[23]          1: 23
              2: 15
              3: NAN
```

RPN basics: Step 3

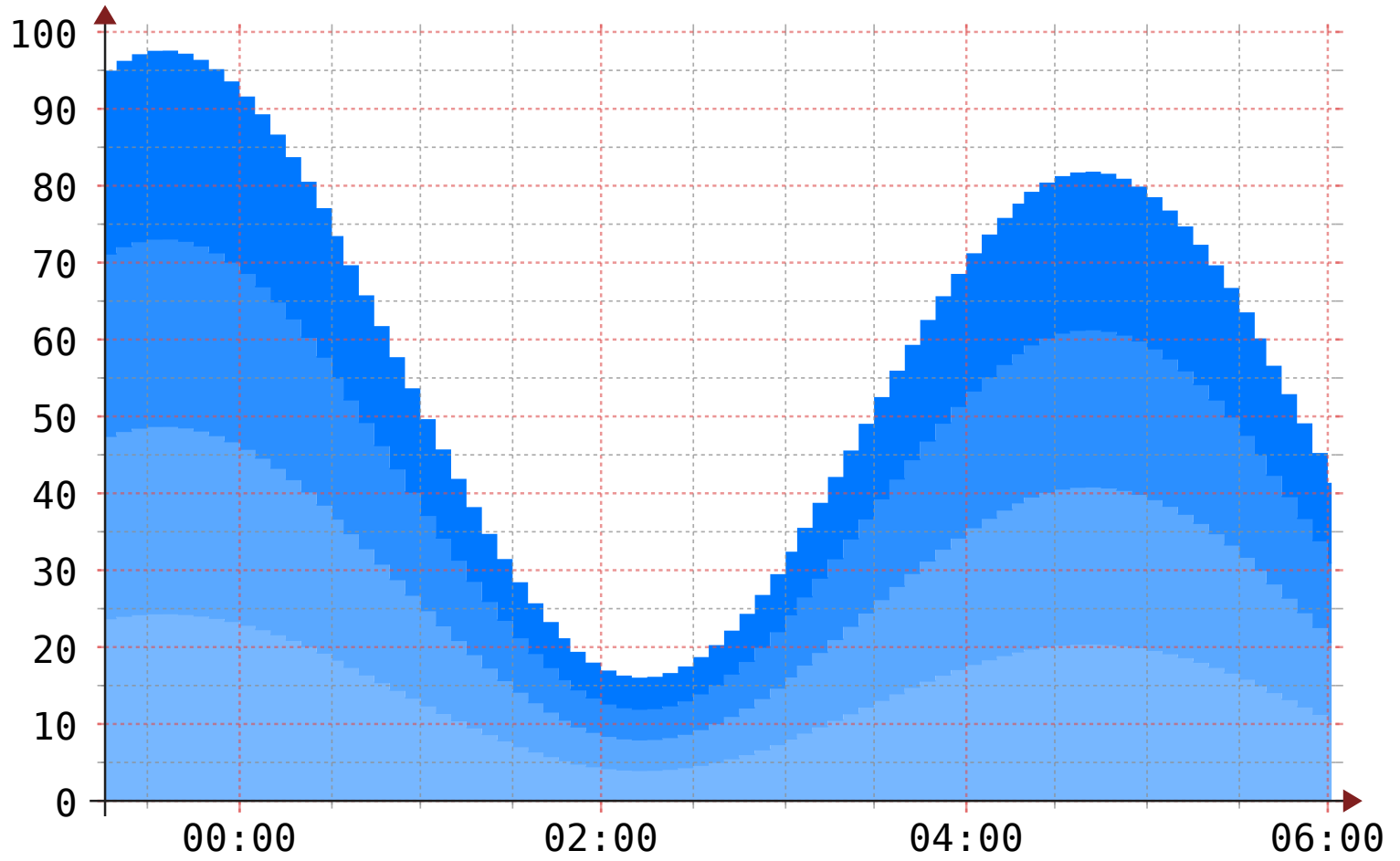
$15+23 = 38$

```
[+]          1: 38
             2: NAN
             3: NAN
```

math in the graph (+)

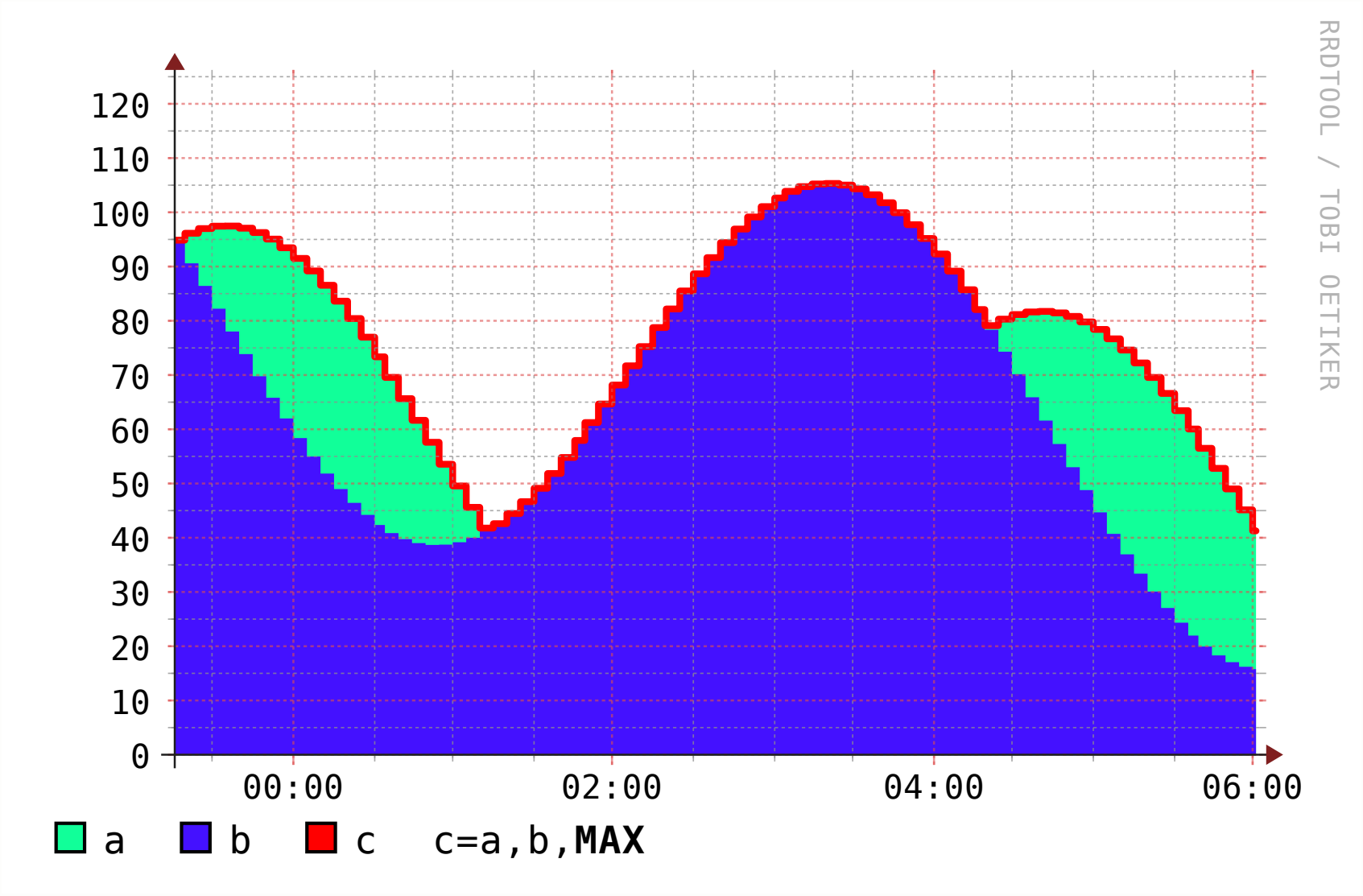


simple gradient

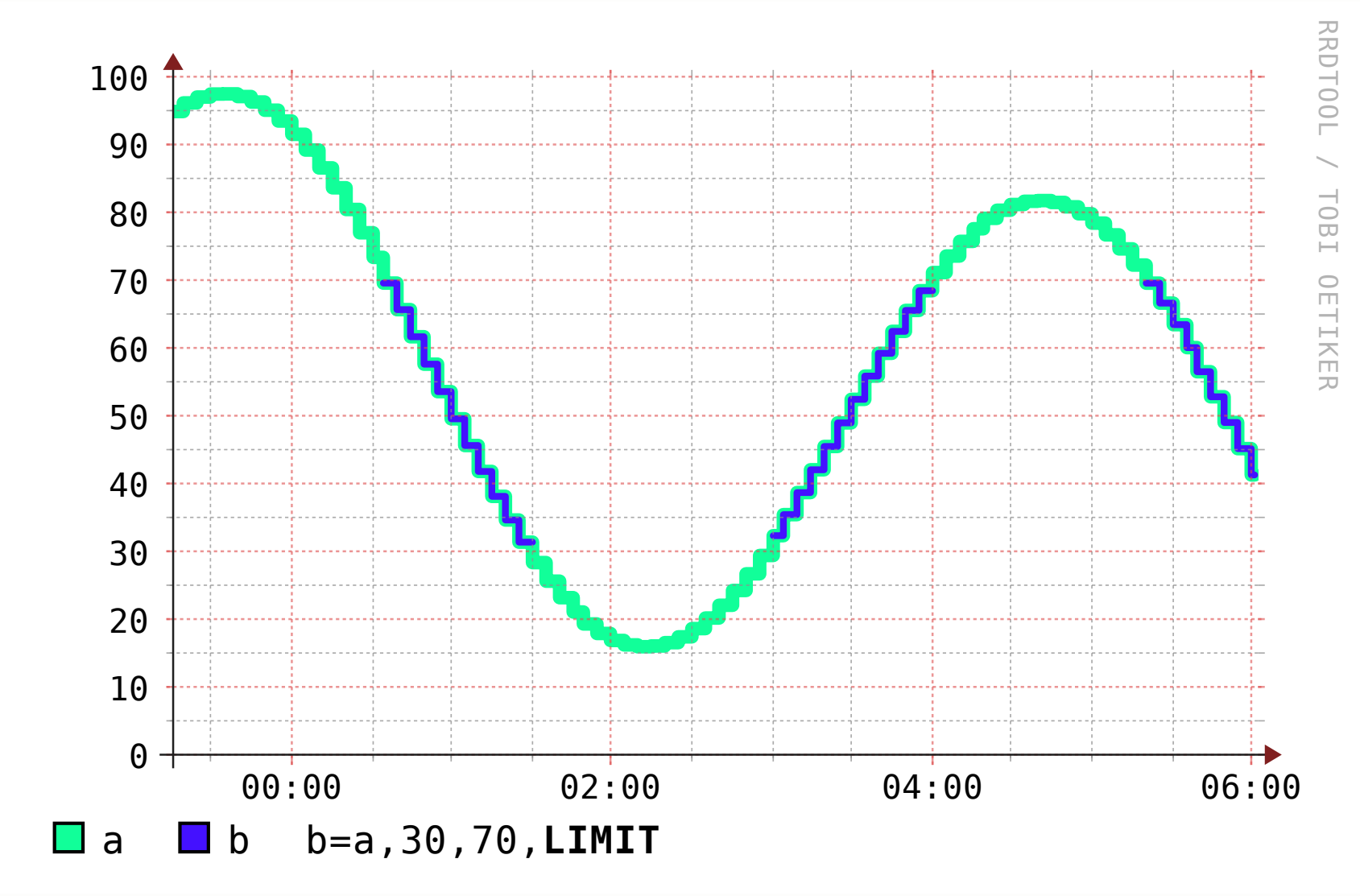



```
CDEF:c=a,4,/
AREA:c#77b7ff
AREA:c#5aa8ff::STACK
AREA:c#2b8fff::STACK
AREA:c#0078ff::STACK
```

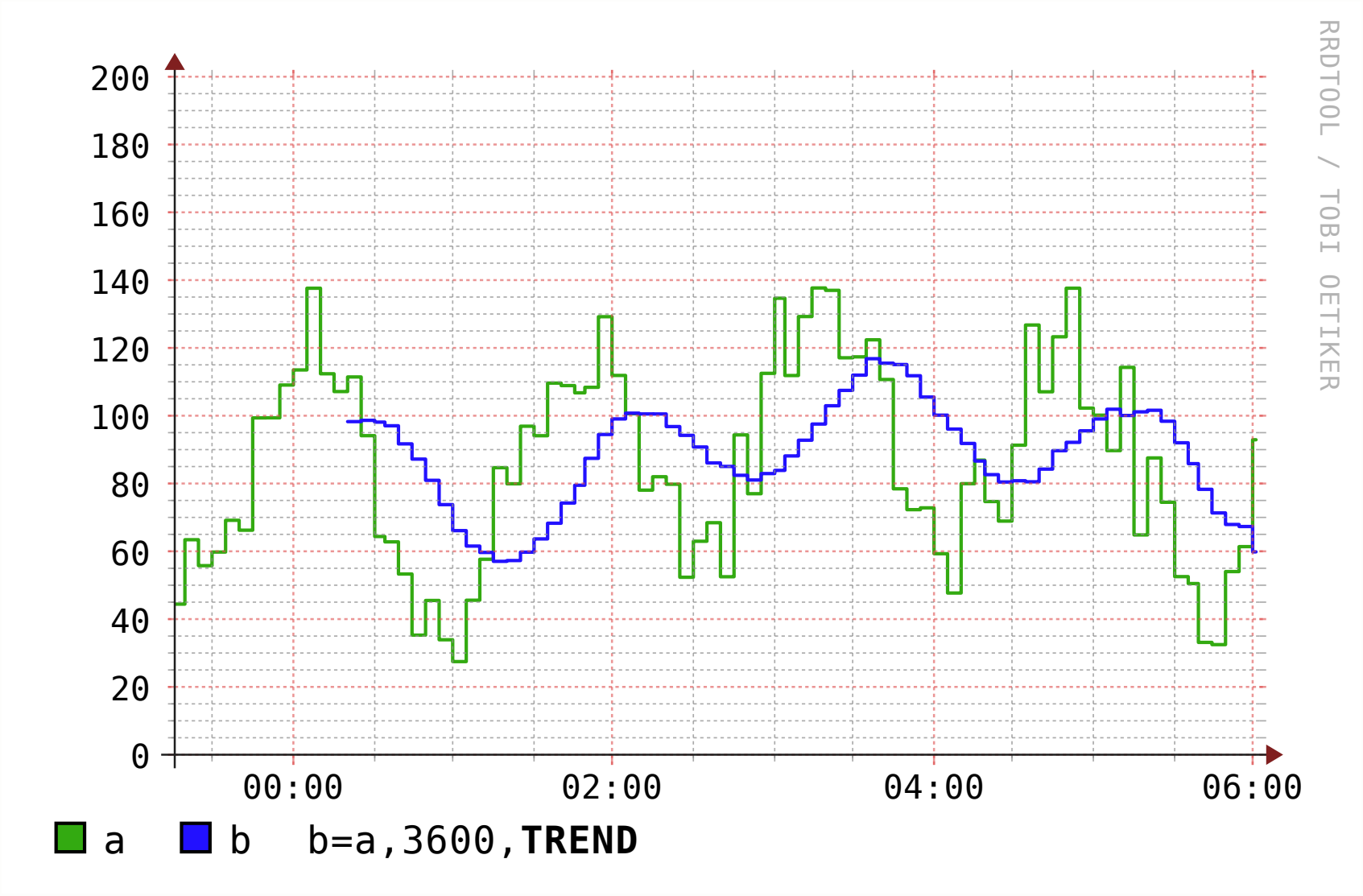
the MAX function



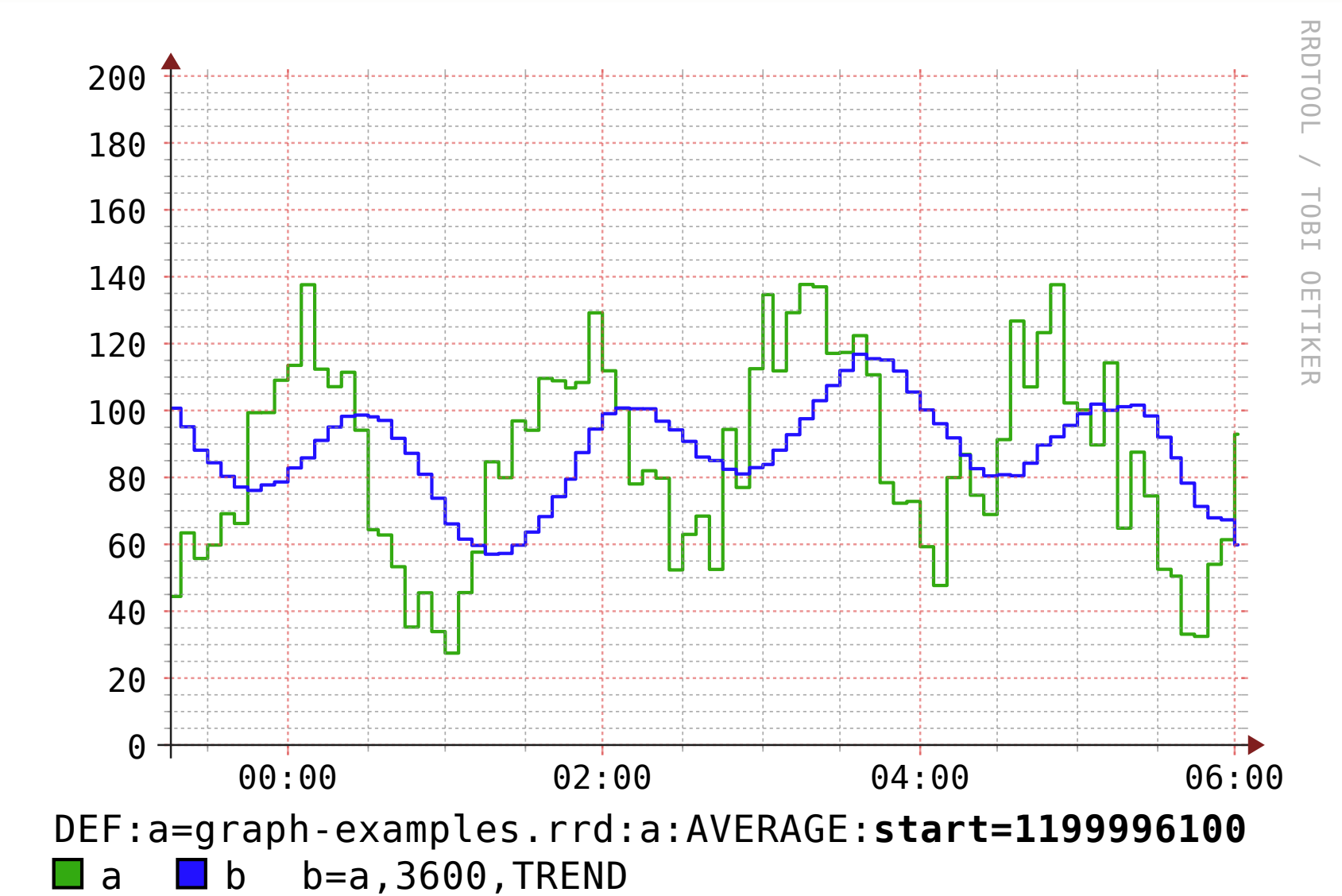
the LIMIT function



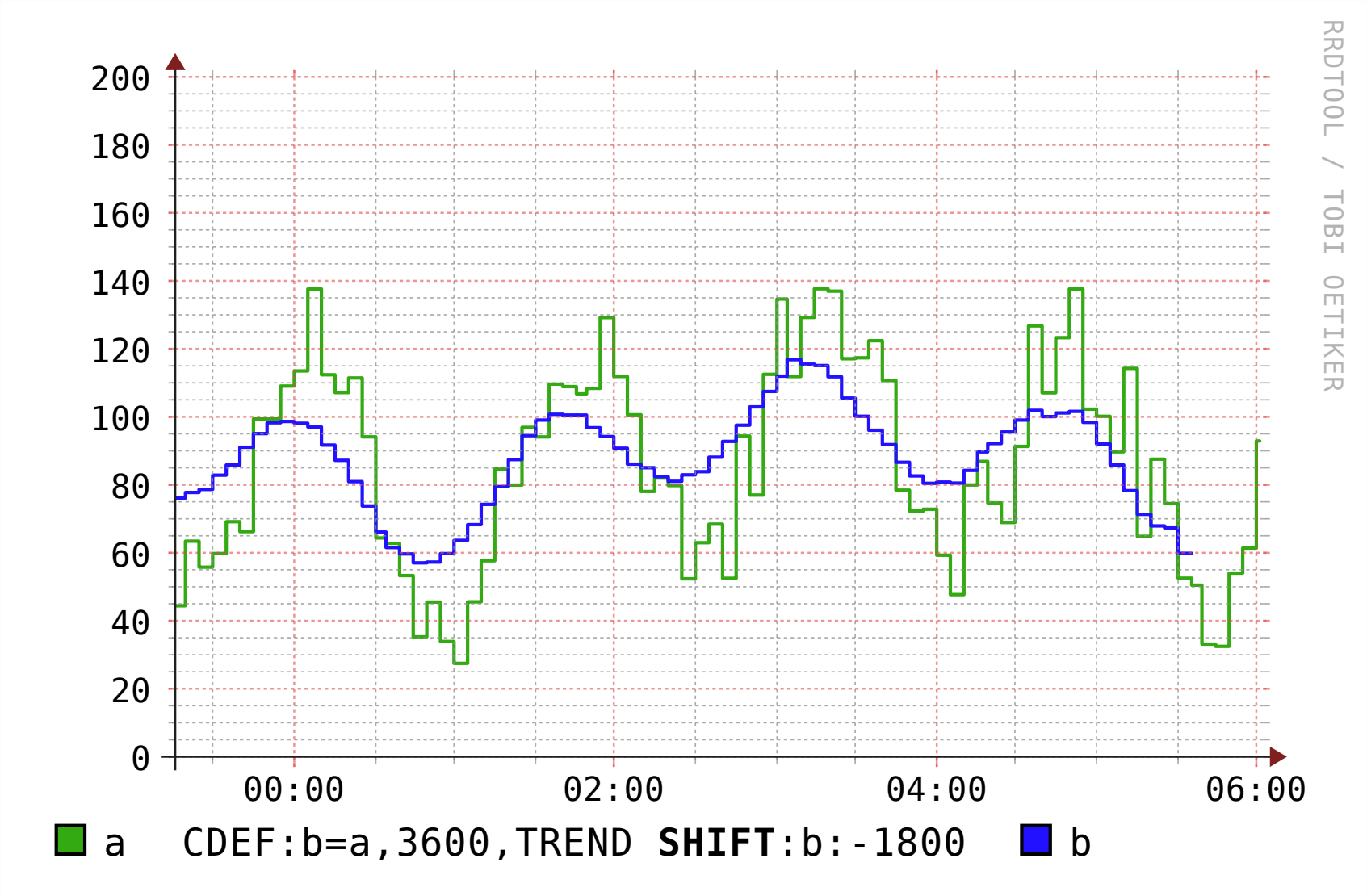
the TREND function



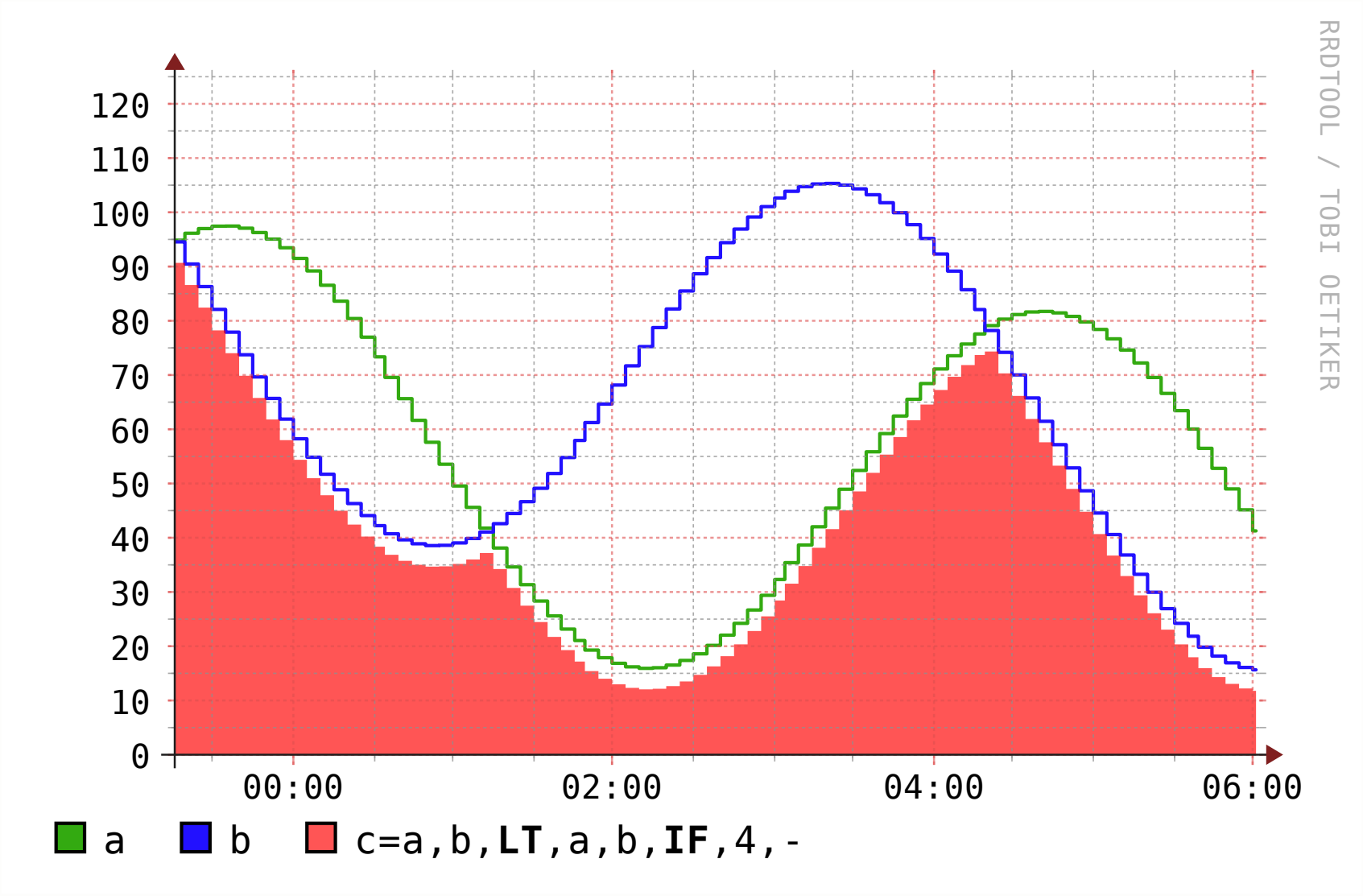
the TREND with early start



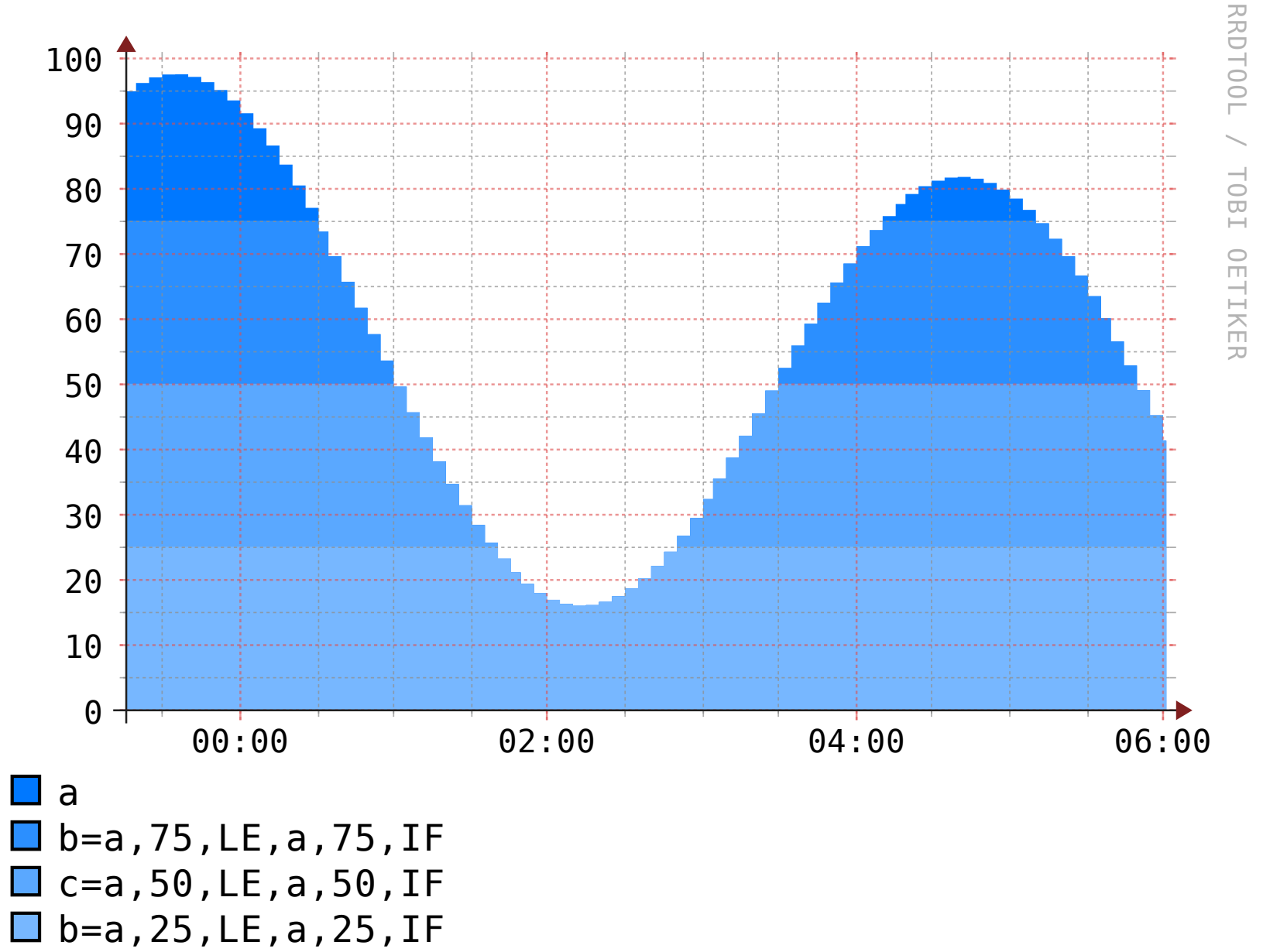
the TREND and SHIFT



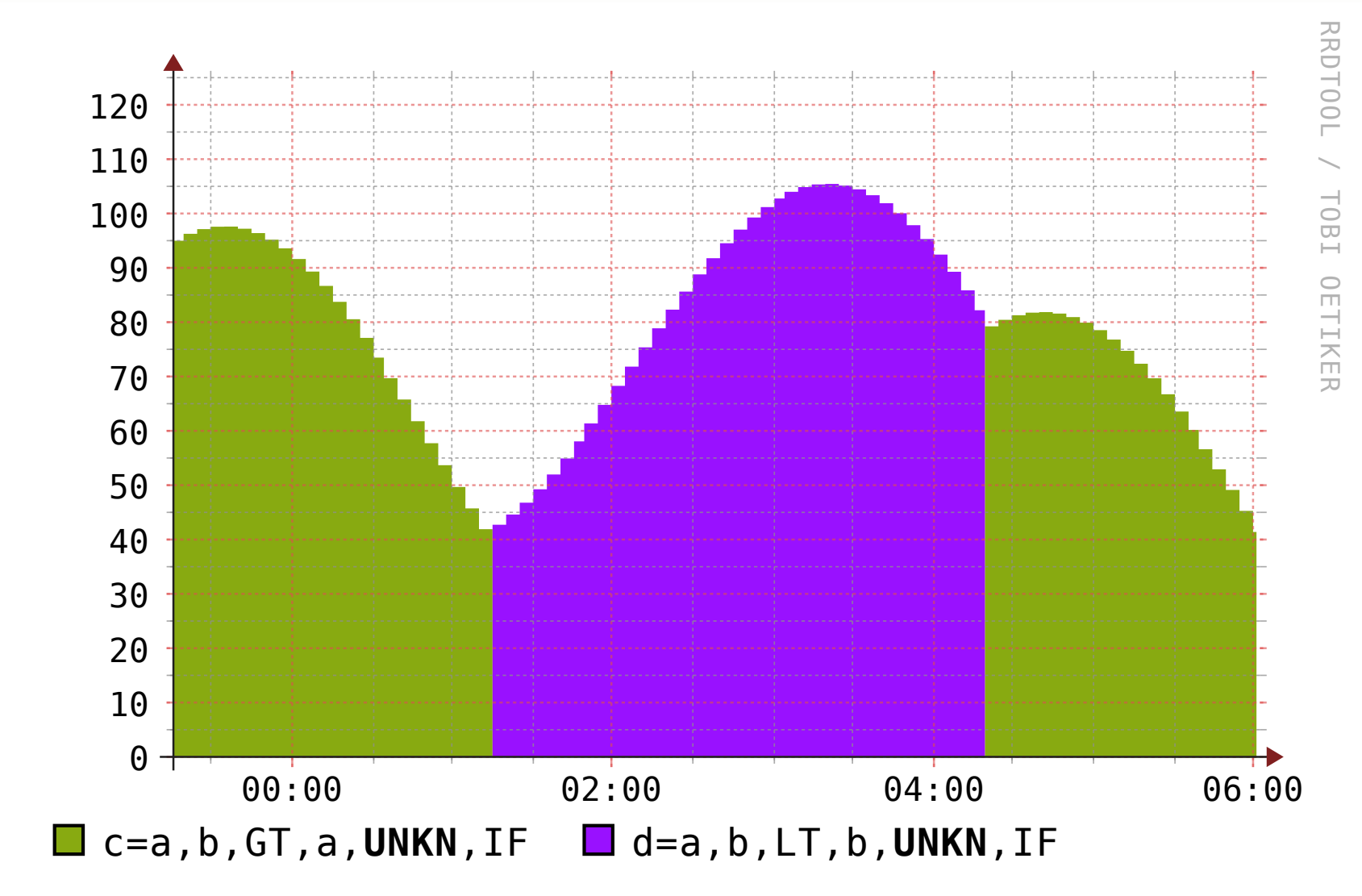
the IF function



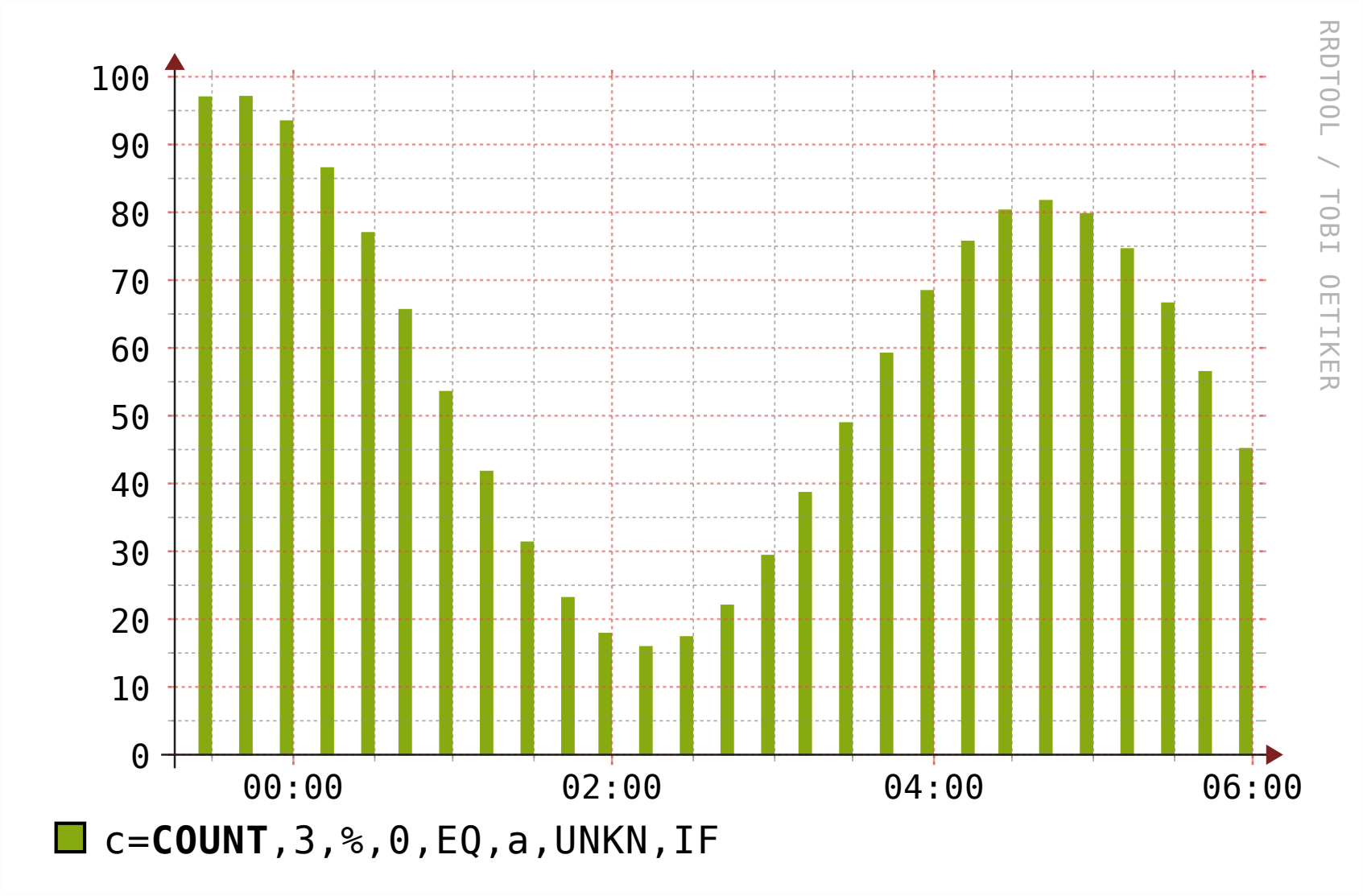
horizontal gradient



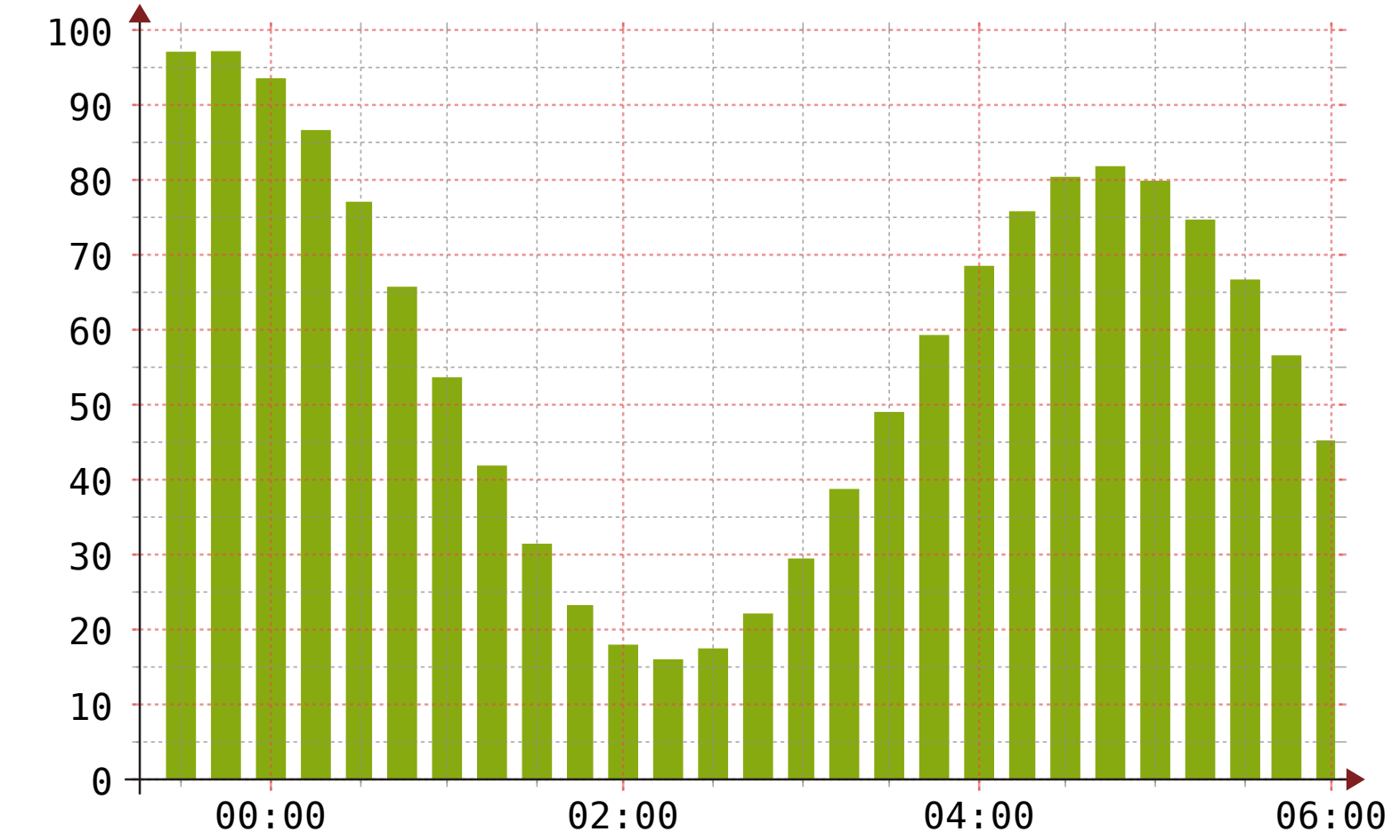
about invisibility



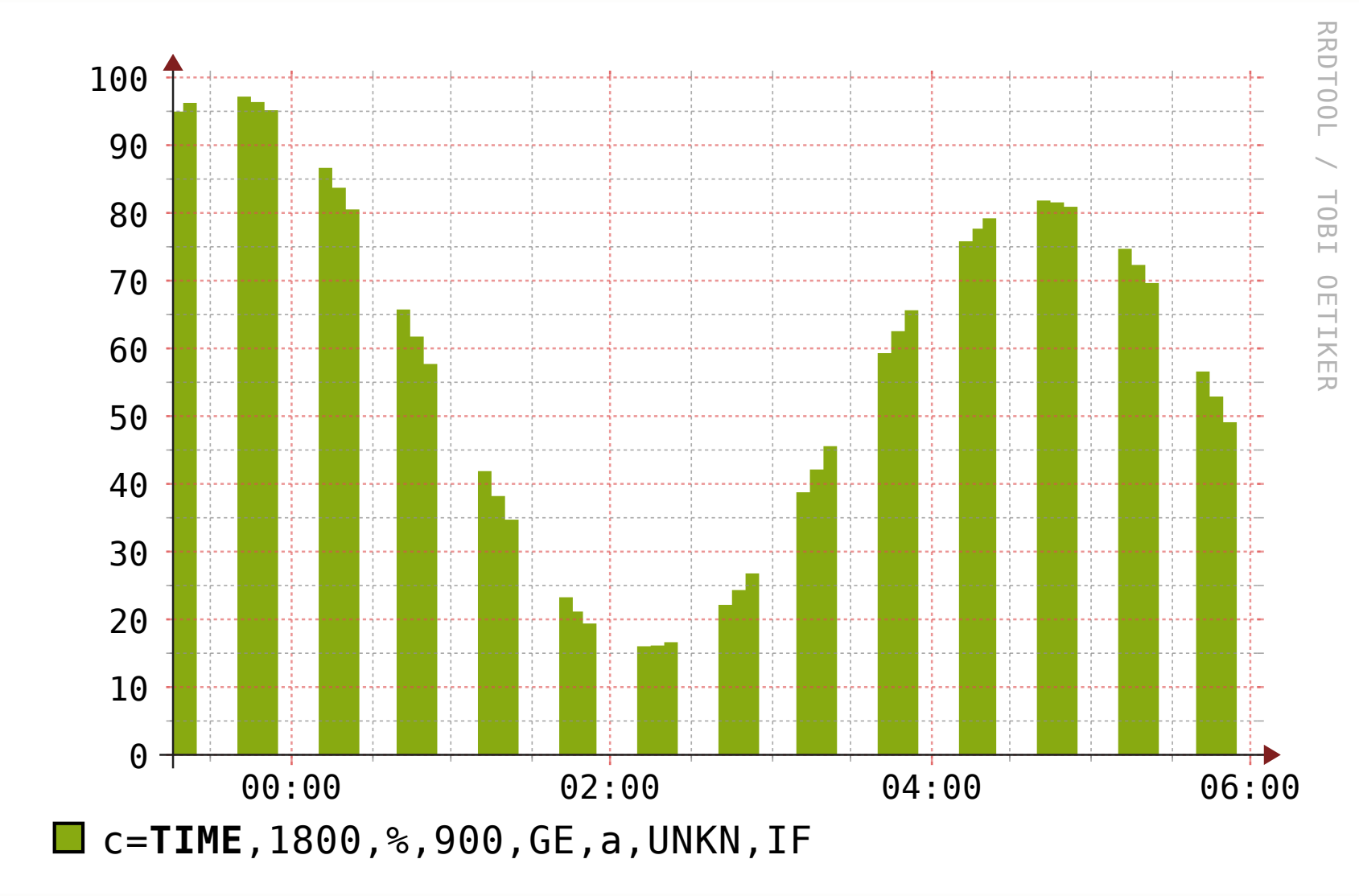
positional drawing count



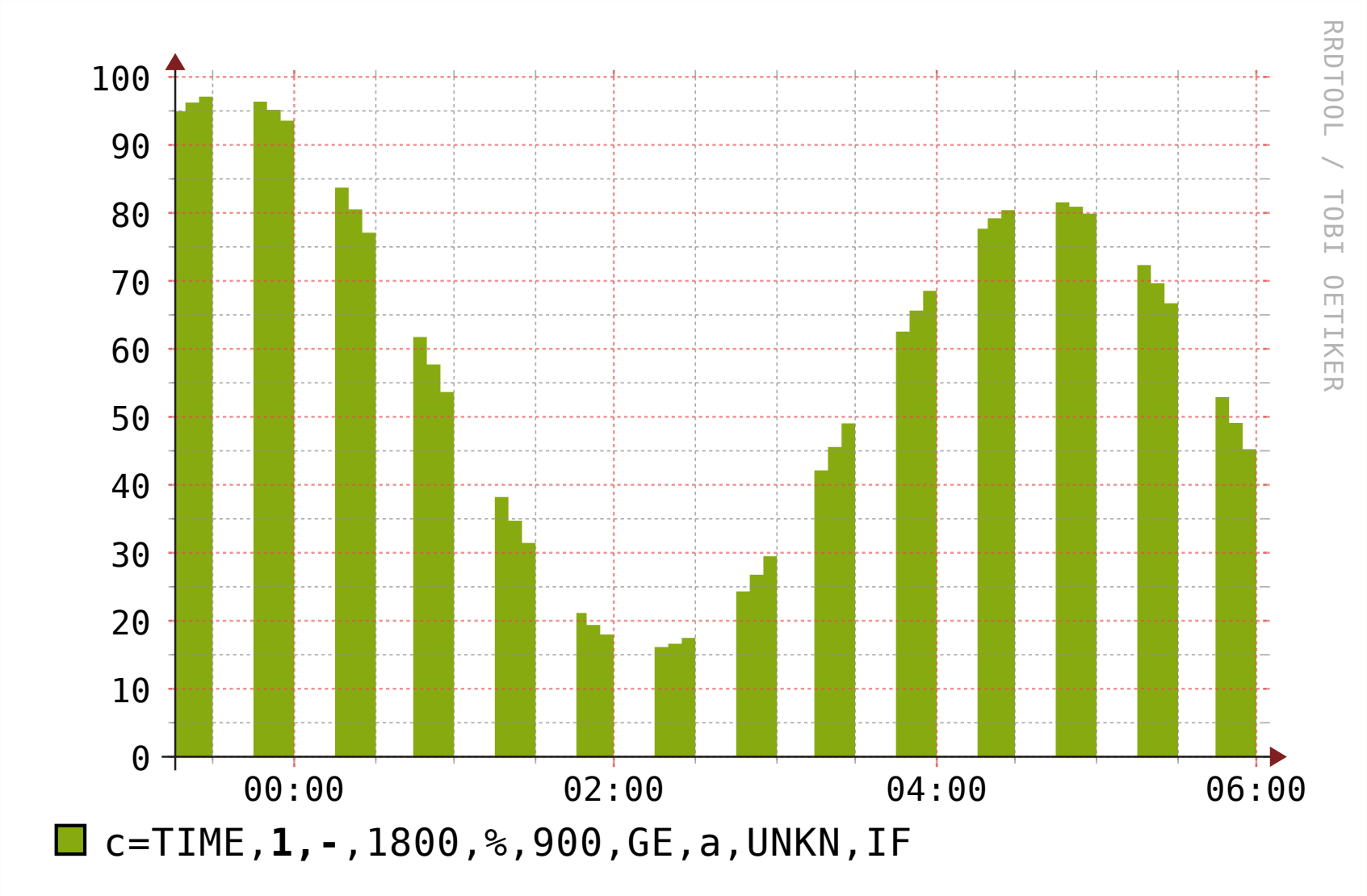
access the previous value



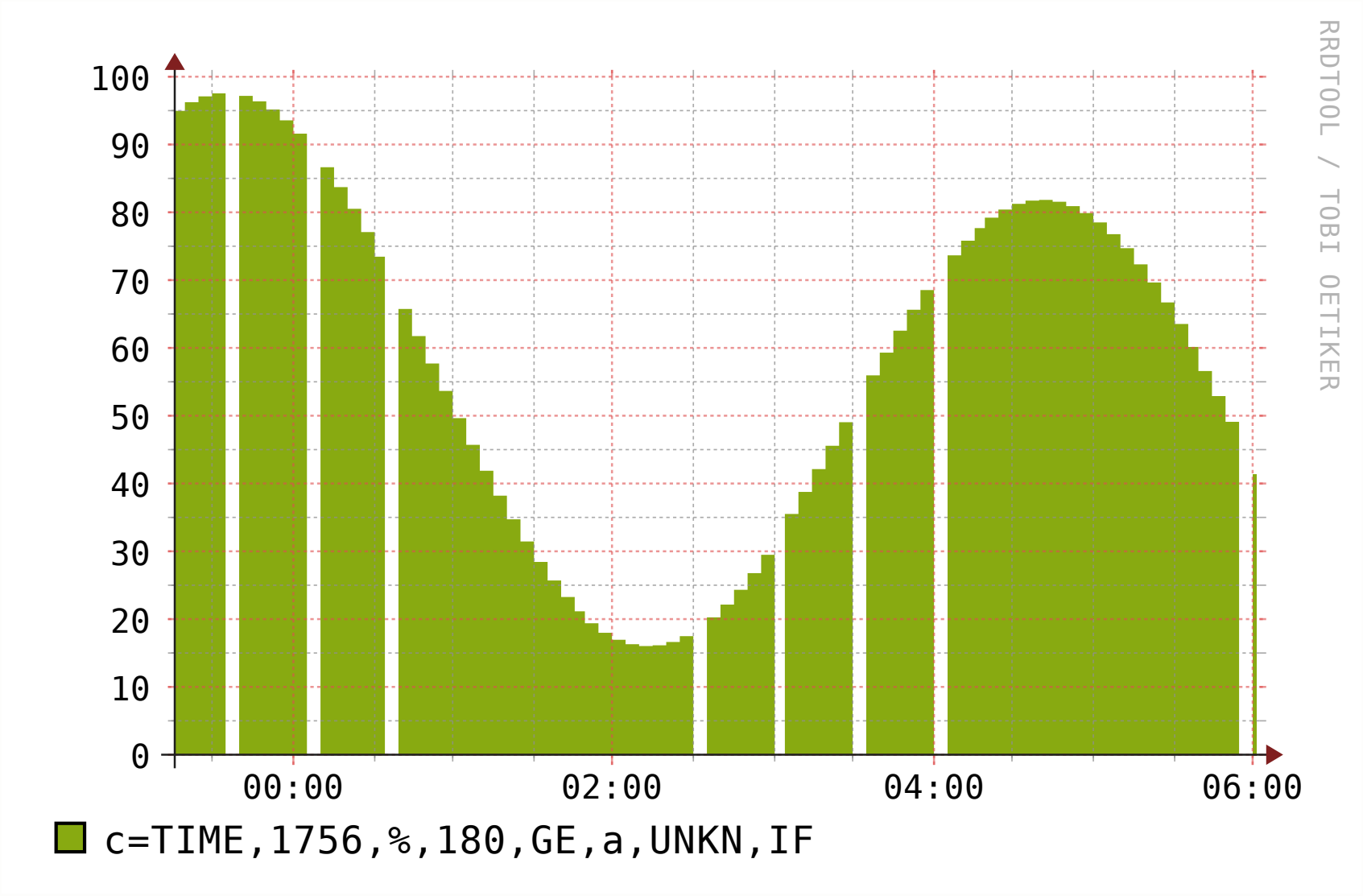
positional drawing time



positional drawing time-shifting



time and resolution issues



CDEF internals

- data may come in different resolutions
- all items in a CDEF must have the same resolution
- resolution is expanded to greatest common devisor (gcd)
- example: gcd(6,9) = 3, gcd(1,6) = 1

trick: an rrd with one a second step.

```
rrdtool create one.rrd --step=1
   DS:one:GAUGE:2:U:U
   RRA:AVERAGE:0.5:1:1
```

CDEF internals

- data may come in different resolutions
- all items in a CDEF must have the same resolution
- resolution is expanded to greatest common devisor (gcd)
- example: gcd(6,9) = 3, gcd(1,6) = 1

trick: an rrd with one a second step.

```
rrdtool create one.rrd --step=1
   DS:one:GAUGE:2:U:U
   RRA:AVERAGE:0.5:1:1
```

CDEF internals

- data may come in different resolutions
- all items in a CDEF must have the same resolution
- resolution is expanded to greatest common devisor (gcd)
- example: gcd(6,9) = 3, gcd(1,6) = 1

trick: an rrd with one a second step.

```
rrdtool create one.rrd --step=1
   DS:one:GAUGE:2:U:U
   RRA:AVERAGE:0.5:1:1
```

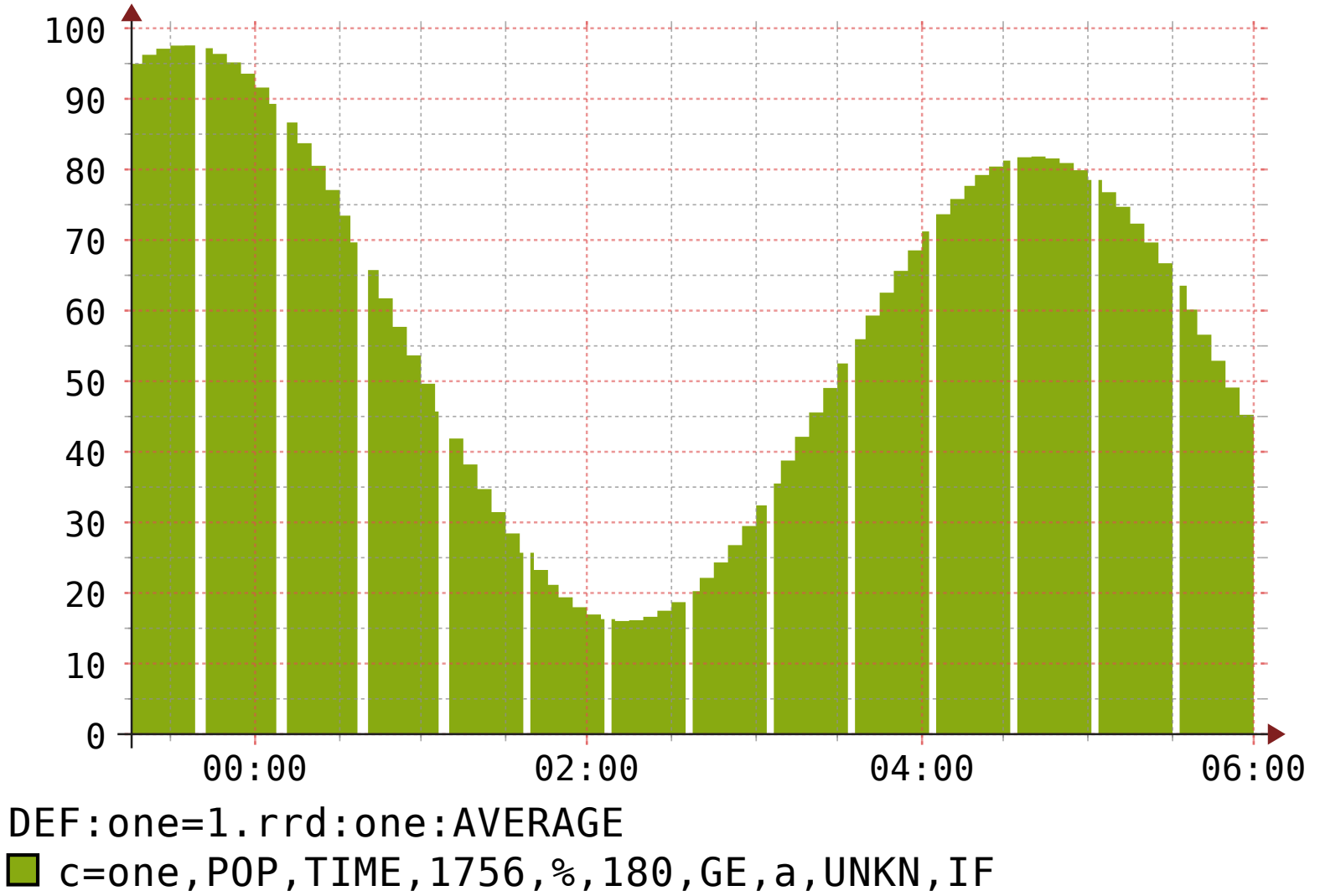
CDEF internals

- data may come in different resolutions
- all items in a CDEF must have the same resolution
- resolution is expanded to greatest common devisor (gcd)
- example: gcd(6,9) = 3, gcd(1,6) = 1

trick: an rrd with one a second step.

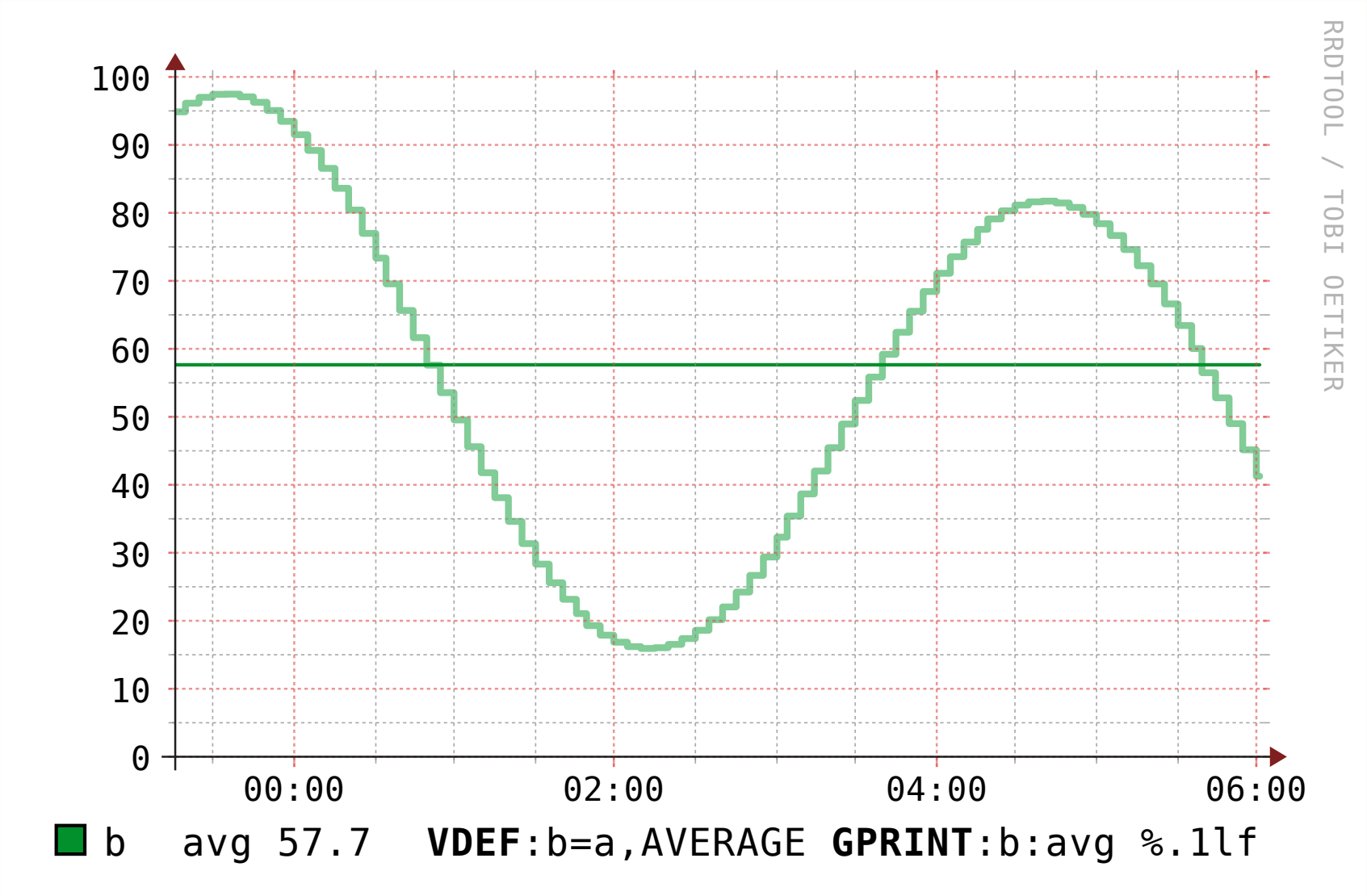
```
rrdtool create one.rrd --step=1
   DS:one:GAUGE:2:U:U
   RRA:AVERAGE:0.5:1:1
```

step=1 trick: high resolution cdef

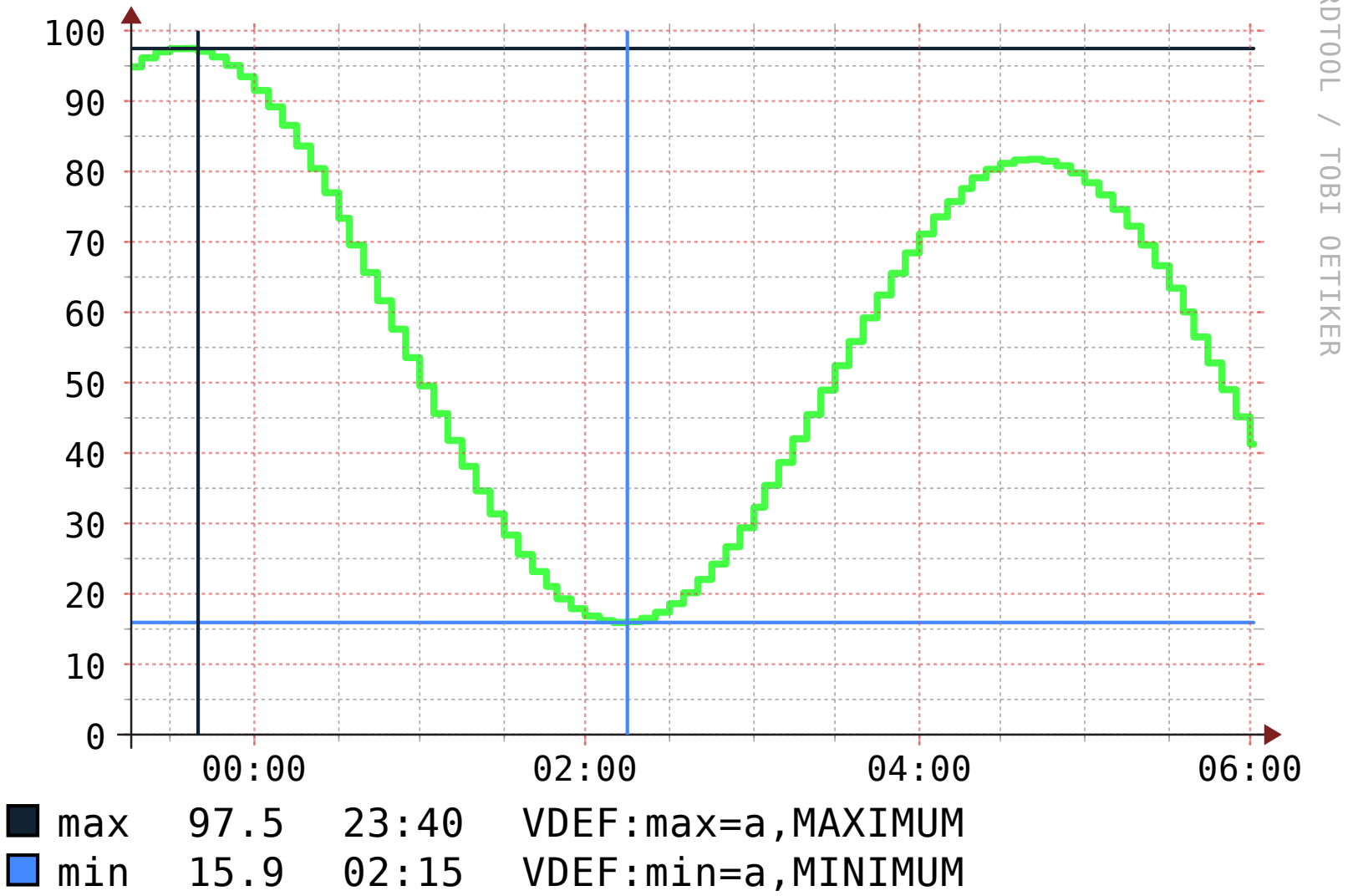


Consolidation functions

finding the average



calculating min and max

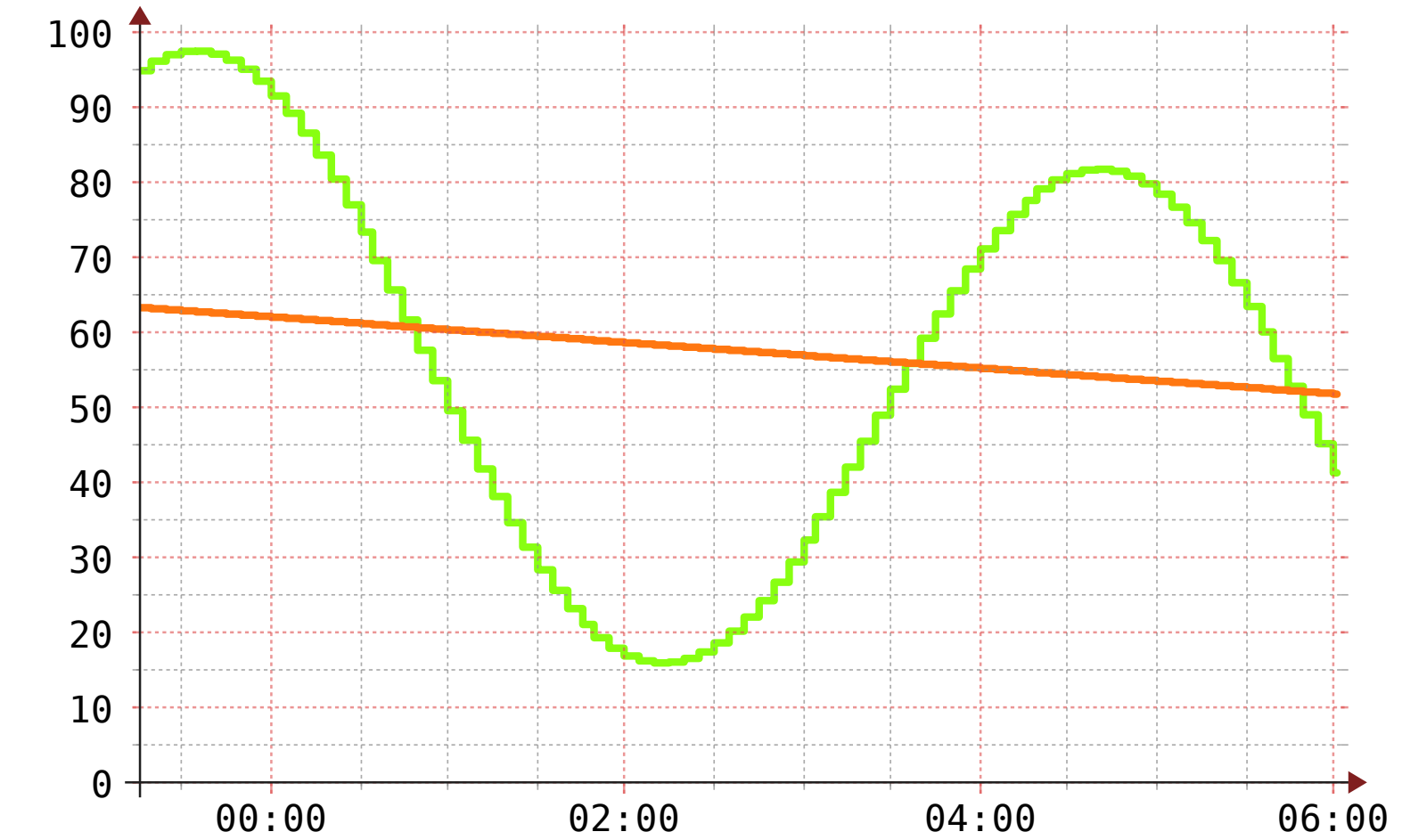


min max code example

```
LINE:a#456:a
VDEF:max=a,MAXIMUM
LINE:max#123
VRULE:max#123:maximum
GPRINT:max:%.1lf
GPRINT:max:%H\:%M:strftime
```

A VDEF result has a value and a time assigned.

Least Squares Line (y=x*m+b)



Holt Winters Aberrant Behaviour Detection

about alert generation

- when something unexpected happens send an alert
- fixed thresholds are too wide a net
- moving averages weigh all data equal
- holt winters can predict the future
- and no one considers himself clever enough to use it

about alert generation

- when something unexpected happens send an alert
- fixed thresholds are too wide a net
- moving averages weigh all data equal
- holt winters can predict the future
- and no one considers himself clever enough to use it

about alert generation

- when something unexpected happens send an alert
- fixed thresholds are too wide a net
- moving averages weigh all data equal
- holt winters can predict the future
- and no one considers himself clever enough to use it

about alert generation

- when something unexpected happens send an alert
- fixed thresholds are too wide a net
- moving averages weigh all data equal
- holt winters can predict the future
- and no one considers himself clever enough to use it

about alert generation

- when something unexpected happens send an alert
- fixed thresholds are too wide a net
- moving averages weigh all data equal
- holt winters can predict the future
- and no one considers himself clever enough to use it

rrd - holt winters assumptions

- data is periodic in nature
- data has continuity
- data continuity is periodic
- recent data is more important

rrd - holt winters assumptions

- data is periodic in nature
- data has continuity
- data continuity is periodic
- recent data is more important

rrd - holt winters assumptions

- data is periodic in nature
- data has continuity
- data continuity is periodic
- recent data is more important

rrd - holt winters assumptions

- data is periodic in nature
- data has continuity
- data continuity is periodic
- recent data is more important

holt winters aberrant behavior

- prediction of future value and confidence band
- confidence band is like a standard deviation
- real value compared to predicted value +/- confidence band

holt winters aberrant behavior

- prediction of future value and confidence band
- confidence band is like a standard deviation
- real value compared to predicted value +/- confidence band

holt winters aberrant behavior

- prediction of future value and confidence band
- confidence band is like a standard deviation
- real value compared to predicted value +/- confidence band

holt winters configuration

- HWPREDICT for starters
- tweaking required
- know the knobs to turn
- use real data to test
- FAILURES very short
- `rrdtool tune` and `resize`

holt winters configuration

- HWPREDICT for starters
- tweaking required
- know the knobs to turn
- use real data to test
- FAILURES very short
- `rrdtool tune` and `resize`

holt winters configuration

- HWPREDICT for starters
- tweaking required
- know the knobs to turn
- use real data to test
- FAILURES very short
- `rrdtool tune` and `resize`

holt winters configuration

- HWPREDICT for starters
- tweaking required
- know the knobs to turn
- use real data to test
- FAILURES very short
- `rrdtool tune` and `resize`

holt winters configuration

- HWPREDICT for starters
- tweaking required
- know the knobs to turn
- use real data to test
- FAILURES very short
- `rrdtool tune` and `resize`

holt winters configuration

- HWPREDICT for starters
- tweaking required
- know the knobs to turn
- use real data to test
- FAILURES very short
- `rrdtool tune` and `resize`

holt winters parameters

`RRA:HWPREDICT:`*rows*`:`*alpha*`:`*beta*`:`*period*

alpha: adaption rate of the baseline (1 fast, 0 slow)

beta: adaption rate of the slope (1 fast, 0 slow)

period: how many steps in a period (use 1 to disable)

gamma: seasonal adaption rate of the baseline (alpha by default)

dev_gamma: seasonal adaption rate of the confidence band (gamma by default)

the gamma and confidence band are tunable with `rrdtool tune`

the rrdtool holt winters formula

```
a - baseline (RRA CDP Parameter)
b - slope (RRA CDP Parameter)
c - seasonal (SEASONAL RRA)
d - deviation (DEVSEASONAL RRA)
pred - predicted value
real - real value

pred{next} = a{now} + b{now} + c{next_prev_period}

a{now} = alpha * (real{now} - c{now_prev_period})
         + (1-alpha) * ( a{prev} + b{prev})
b{now} = beta  * (a{now} - a{prev})
         + (1-beta) * b_prev
c{now} = gamma * (real{now} - a{now})
         + (1-gamma) * c{now_prev_period}
d{now} = dev_gamma * abs(real{now} - pred{now})
         + (1-dev_gamma) * d{now_prev_period}
```

the rrdtool holt winters formula

```
a - baseline (RRA CDP Parameter)
b - slope (RRA CDP Parameter)
c - seasonal (SEASONAL RRA)
d - deviation (DEVSEASONAL RRA)
pred - predicted value
real - real value

pred{next} = a{now} + b{now} + c{next_prev_period}

a{now} = alpha * (real{now} - c{now_prev_period})
         + (1-alpha) * ( a{prev} + b{prev})
b{now} = beta  * (a{now} - a{prev})
         + (1-beta) * b_prev
c{now} = gamma * (real{now} - a{now})
         + (1-gamma) * c{now_prev_period}
d{now} = dev_gamma * abs(real{now} - pred{now})
         + (1-dev_gamma) * d{now_prev_period}
```

the rrdtool holt winters formula

```
a - baseline (RRA CDP Parameter)
b - slope (RRA CDP Parameter)
c - seasonal (SEASONAL RRA)
d - deviation (DEVSEASONAL RRA)
pred - predicted value
real - real value

pred{next} = a{now} + b{now} + c{next_prev_period}

a{now} = alpha * (real{now} - c{now_prev_period})
         + (1-alpha) * ( a{prev} + b{prev})
b{now} = beta  * (a{now} - a{prev})
         + (1-beta) * b_prev
c{now} = gamma * (real{now} - a{now})
         + (1-gamma) * c{now_prev_period}
d{now} = dev_gamma * abs(real{now} - pred{now})
         + (1-dev_gamma) * d{now_prev_period}
```

the rrdtool holt winters formula

```
a - baseline (RRA CDP Parameter)
b - slope (RRA CDP Parameter)
c - seasonal (SEASONAL RRA)
d - deviation (DEVSEASONAL RRA)
pred - predicted value
real - real value

pred{next} = a{now} + b{now} + c{next_prev_period}

a{now} = alpha * (real{now} - c{now_prev_period})
         + (1-alpha) * ( a{prev} + b{prev})
b{now} = beta  * (a{now} - a{prev})
         + (1-beta) * b_prev
c{now} = gamma * (real{now} - a{now})
         + (1-gamma) * c{now_prev_period}
d{now} = dev_gamma * abs(real{now} - pred{now})
         + (1-dev_gamma) * d{now_prev_period}
```

the rrdtool holt winters formula

```
a - baseline (RRA CDP Parameter)
b - slope (RRA CDP Parameter)
c - seasonal (SEASONAL RRA)
d - deviation (DEVSEASONAL RRA)
pred - predicted value
real - real value

pred{next} = a{now} + b{now} + c{next_prev_period}

a{now} = alpha * (real{now} - c{now_prev_period})
         + (1-alpha) * ( a{prev} + b{prev})
b{now} = beta  * (a{now} - a{prev})
         + (1-beta) * b_prev
c{now} = gamma * (real{now} - a{now})
         + (1-gamma) * c{now_prev_period}
d{now} = dev_gamma * abs(real{now} - pred{now})
         + (1-dev_gamma) * d{now_prev_period}
```

the rrdtool holt winters formula

```
a - baseline (RRA CDP Parameter)
b - slope (RRA CDP Parameter)
c - seasonal (SEASONAL RRA)
d - deviation (DEVSEASONAL RRA)
pred - predicted value
real - real value

pred{next} = a{now} + b{now} + c{next_prev_period}

a{now} = alpha * (real{now} - c{now_prev_period})
         + (1-alpha) * ( a{prev} + b{prev})
b{now} = beta  * (a{now} - a{prev})
         + (1-beta) * b_prev
c{now} = gamma * (real{now} - a{now})
         + (1-gamma) * c{now_prev_period}
d{now} = dev_gamma * abs(real{now} - pred{now})
         + (1-dev_gamma) * d{now_prev_period}
```

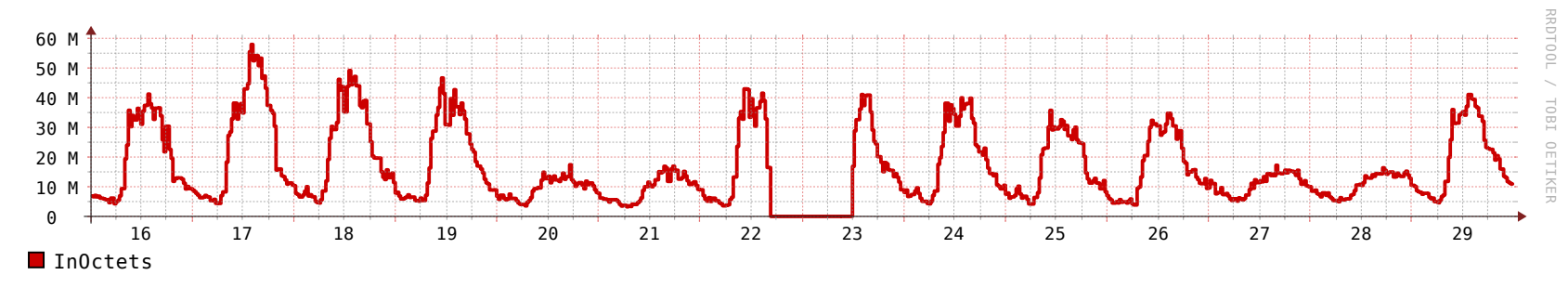
the rrdtool holt winters formula

```
a - baseline (RRA CDP Parameter)
b - slope (RRA CDP Parameter)
c - seasonal (SEASONAL RRA)
d - deviation (DEVSEASONAL RRA)
pred - predicted value
real - real value

pred{next} = a{now} + b{now} + c{next_prev_period}

a{now} = alpha * (real{now} - c{now_prev_period})
         + (1-alpha) * ( a{prev} + b{prev})
b{now} = beta  * (a{now} - a{prev})
         + (1-beta) * b_prev
c{now} = gamma * (real{now} - a{now})
         + (1-gamma) * c{now_prev_period}
d{now} = dev_gamma * abs(real{now} - pred{now})
         + (1-dev_gamma) * d{now_prev_period}
```

hw demo: the test data

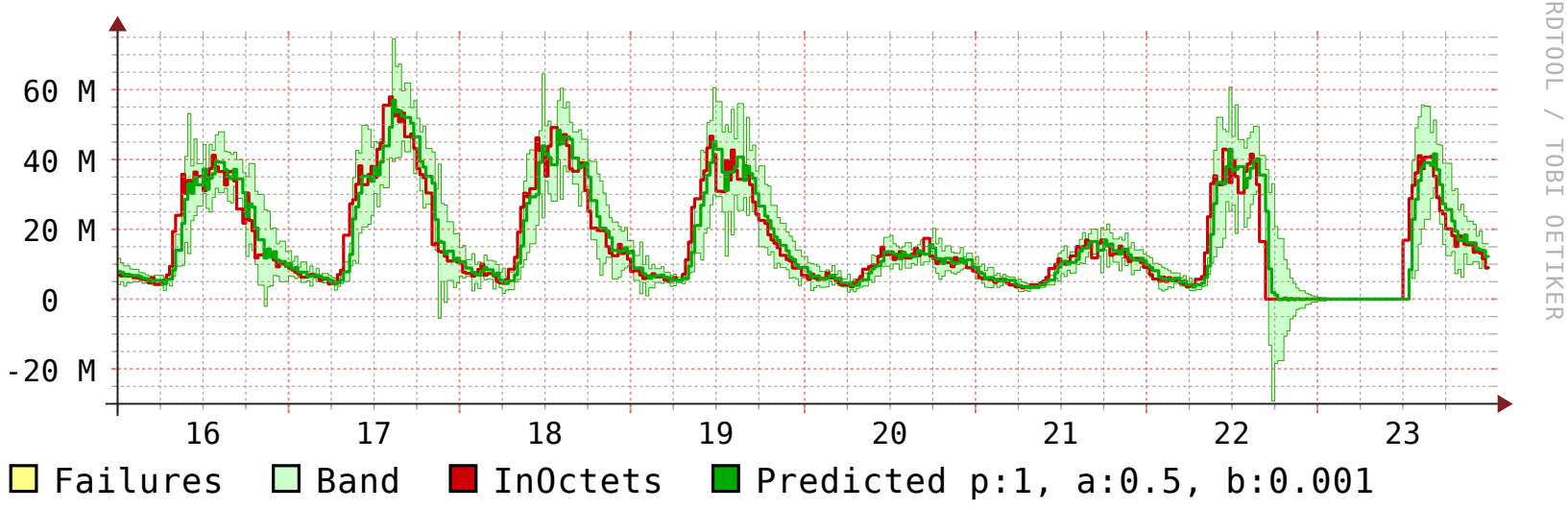


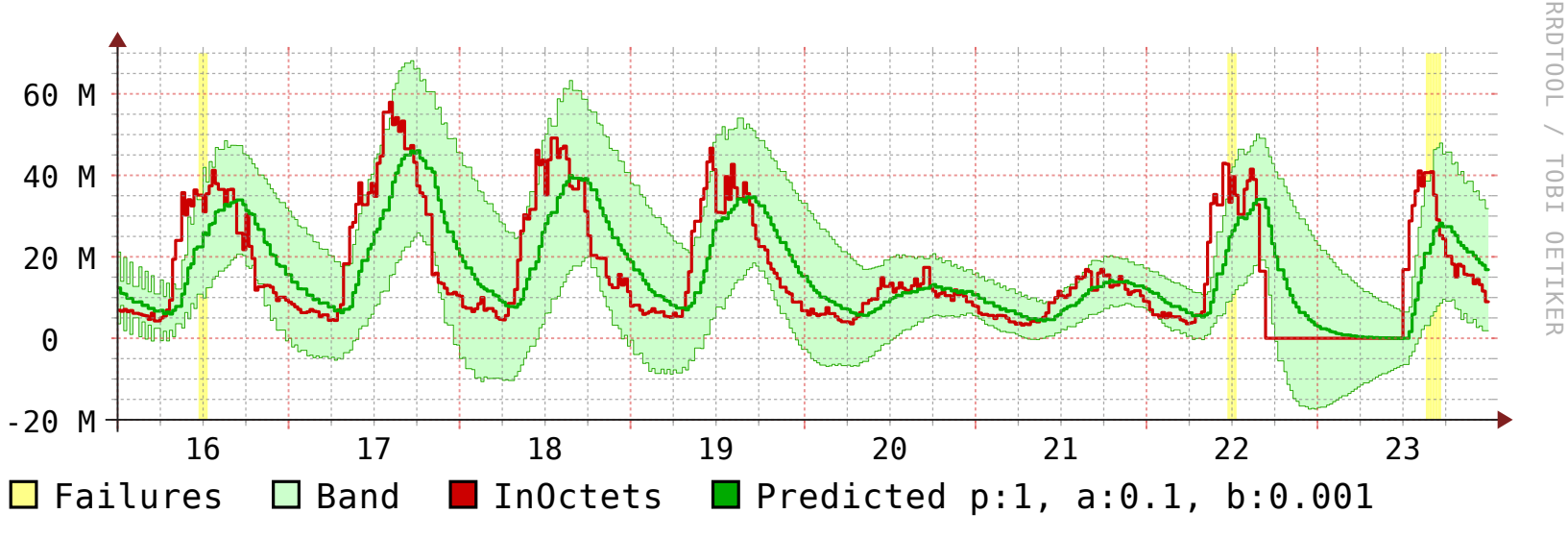
traffic at a peering point

drawing a hw graph

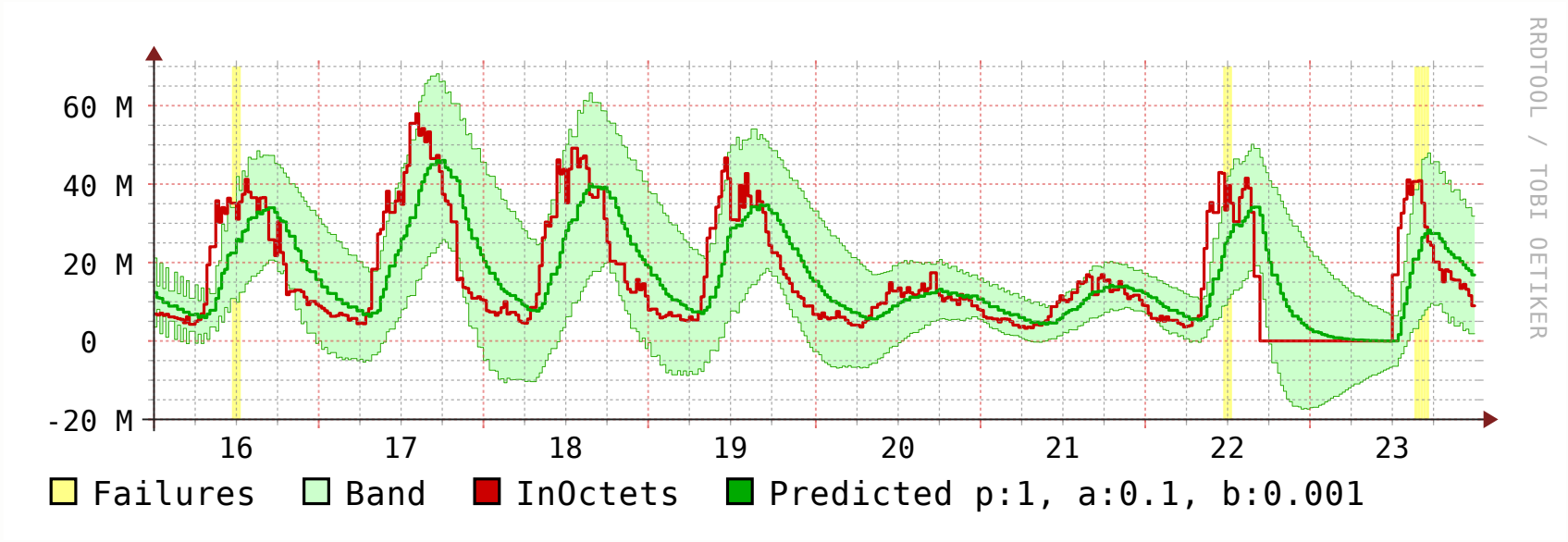
```
1  DEF:in=hw.rrd:in:AVERAGE
2  DEF:pred=hw.rrd:in:HWPREDICT
3  DEF:conf=hw.rrd:in:DEVPREDICT
4  DEF:fail=hw.rrd:in:FAILURES
5  TICK:fail#ff8:1:Failures
6  CDEF:lowconf=pred,conf,2,*,-
7  LINE1:lowconf
8  CDEF:confwidth=conf,4,*
9  AREA:confwidth#cfc:Band:STACK
10 LINE0.1:0#3a1::STACK
11 LINE0.1:lowconf#3a1
12 LINE1:in#c00:InOctets
13 LINE1:pred#0a0:Prediction
```

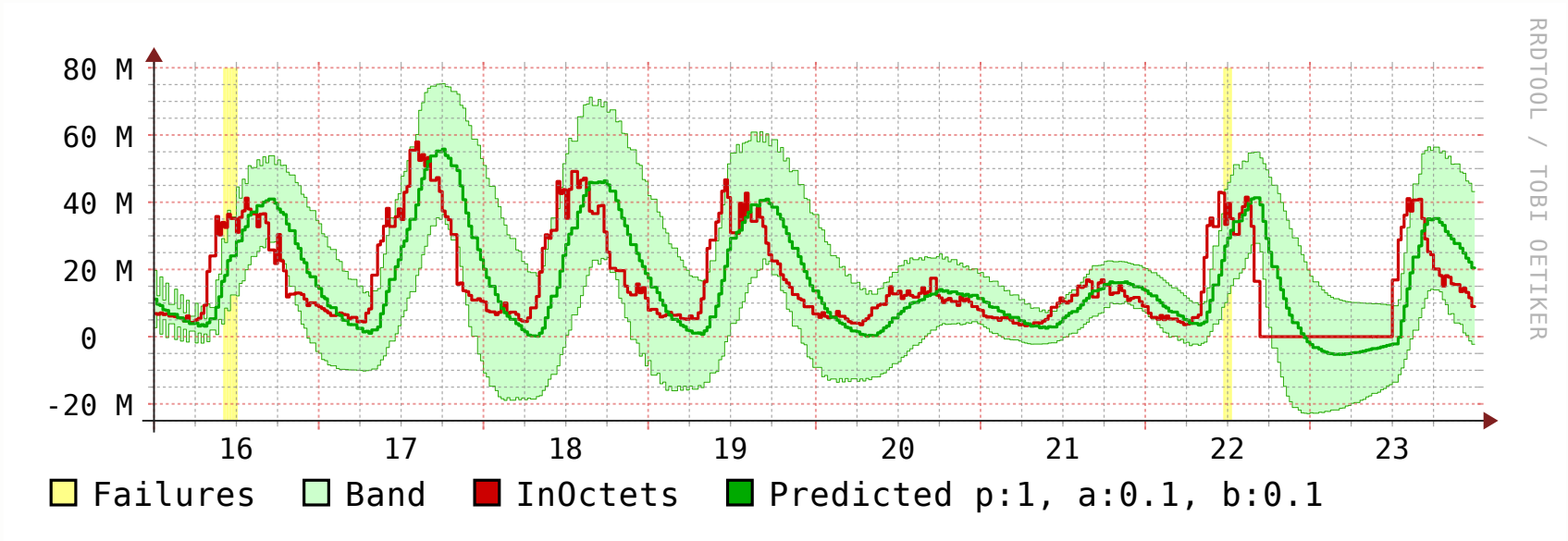
hw demo: alpha



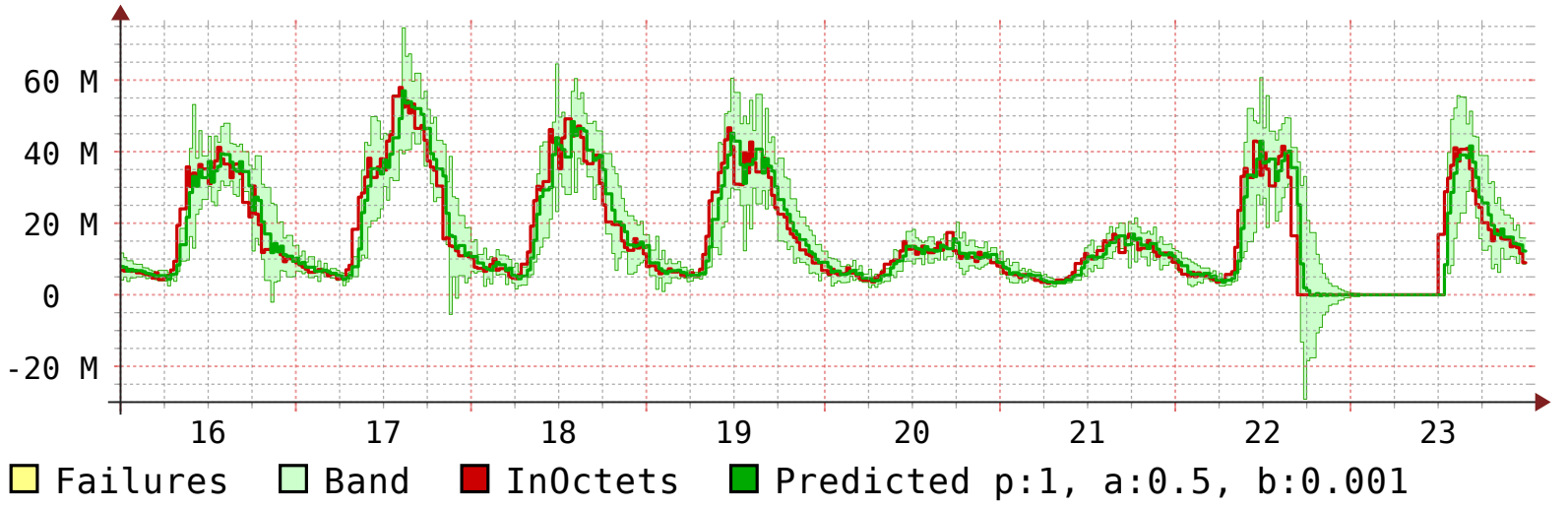


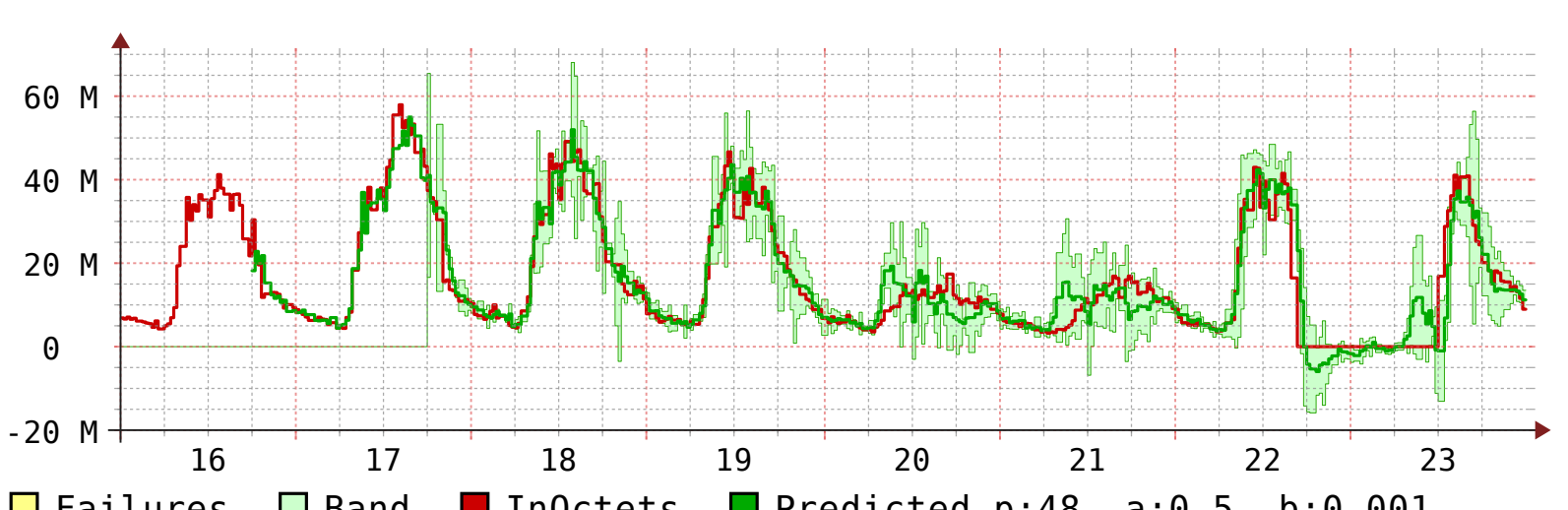
hw demo: beta


60 M
40 M
20 M
0
-20 M
16
17
18
19
20
21
22
23
Failures
Band
InOctets
Predicted p:1, a:0.1, b:0.001
RRDTOOL / TOBI OETIKER

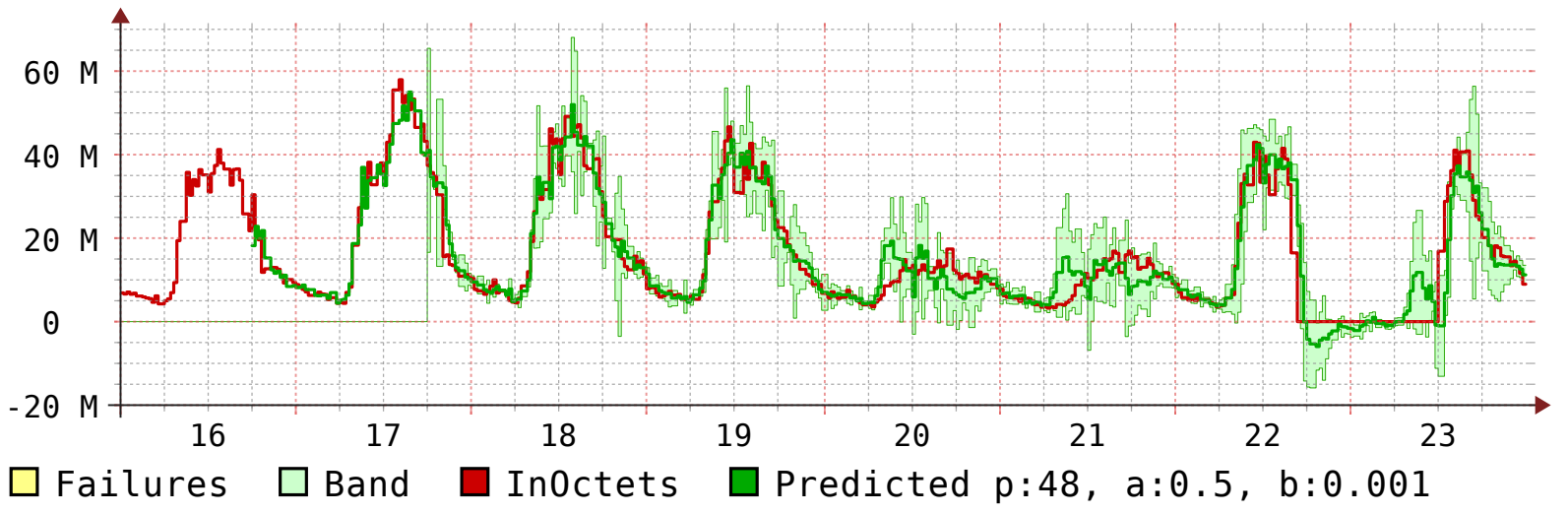

80 M
60 M
40 M
20 M
0
-20 M
16
17
18
19
20
21
22
23
Failures
Band
InOctets
Predicted p:1, a:0.1, b:0.1
RRDTOOL / TOBI OETIKER

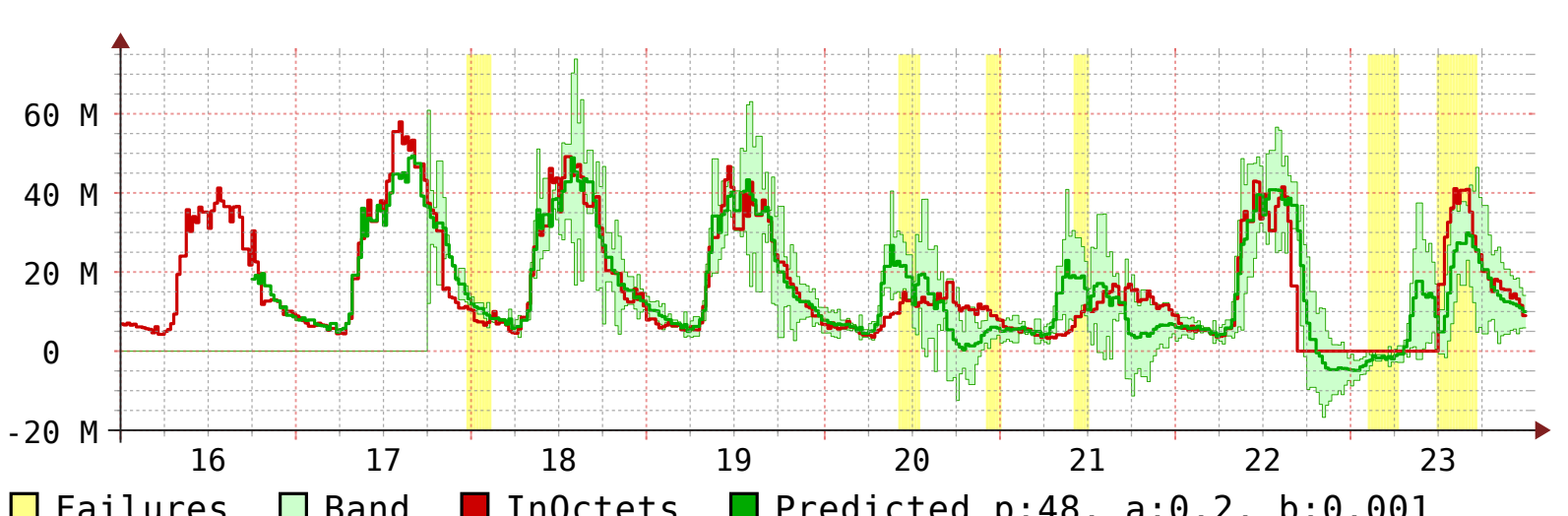
hw demo: period



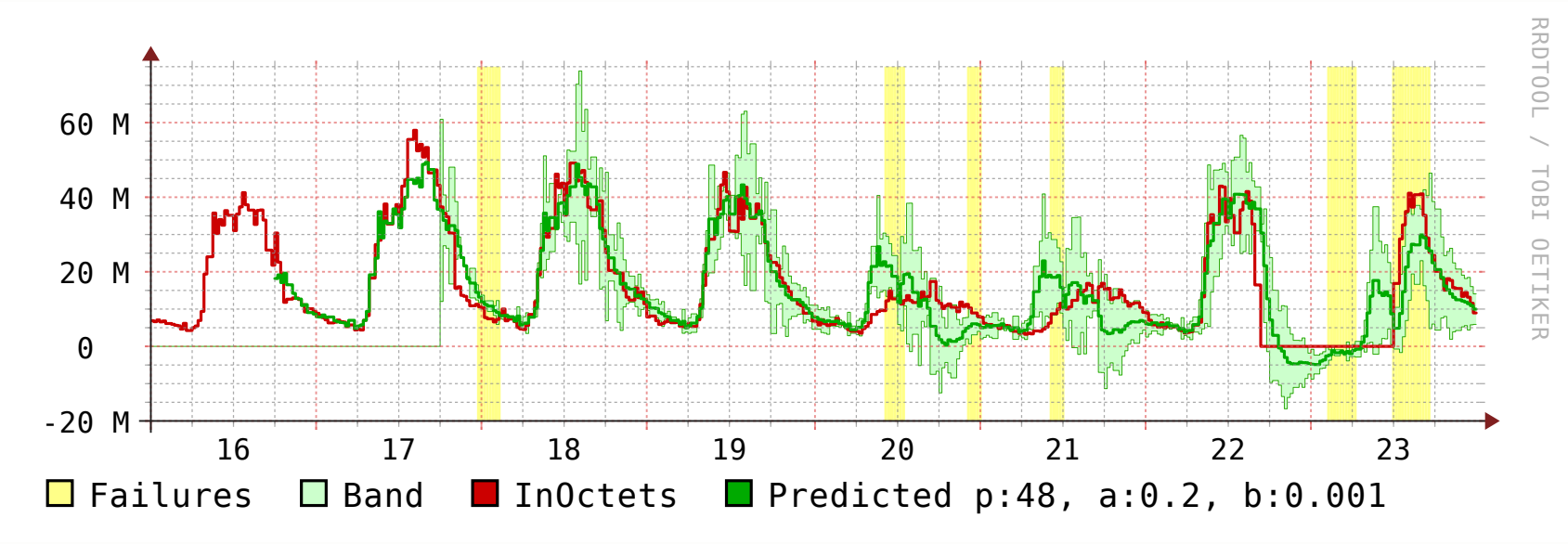


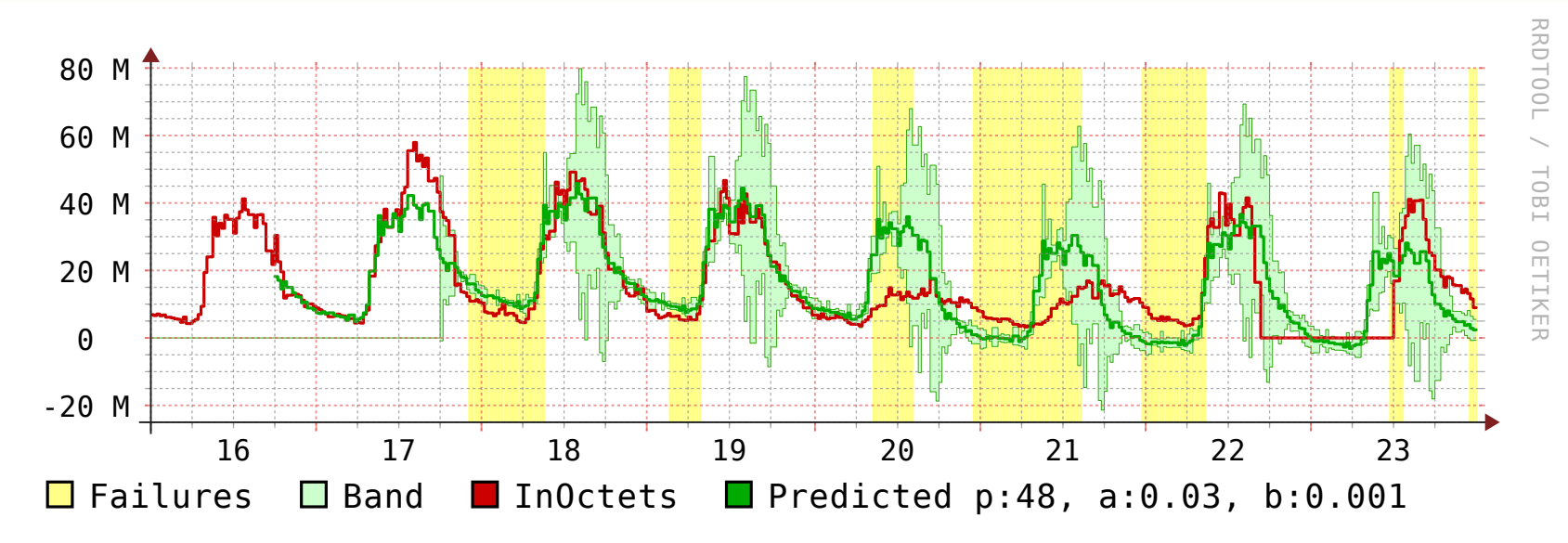
hw demo: tuning



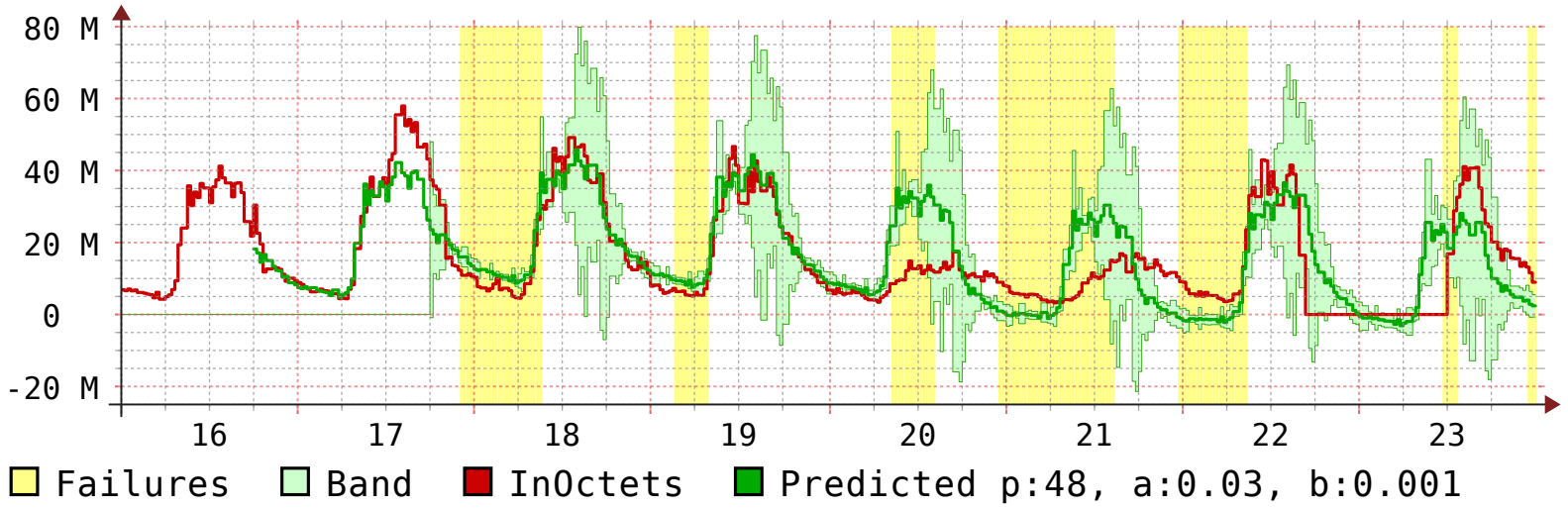


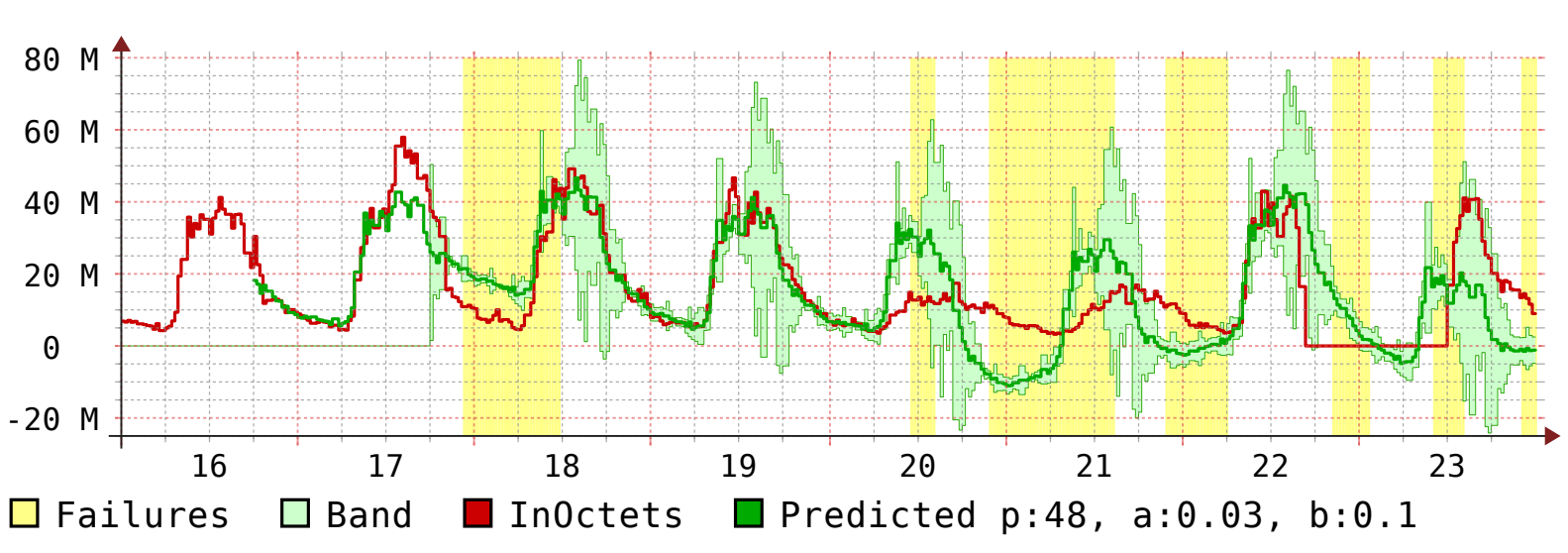
hw demo: tuning II





hw demo: tuning III





The *v Interfaces

graphv script

```
#!/usr/bin/perl -w
use strict;
use lib qw( /scratch/rrd4/lib/perl );
use RRDs;
my $out = RRDs::graphv(
      '-', '--start' => '00:00 20080916',
      '--end' => 'start+8d',
      '--lower-limit' => 0,
      '--imgformat' => 'PDF',
      'DEF:a=hw-demo.rrd:in:AVERAGE',
      'LINE1:a#c00:InOctets');
my $ERROR = RRDs::error;
die "ERROR: $ERROR\n" if $ERROR;
map {
     print $_.' = '.substr($out->{$_},0,8)."\n"
} sort keys %$out;
```

graphv output

```
graph_height = 100
graph_left = 51
graph_top = 22
graph_width = 400
image = %PDF-1.4
image_height = 163
image_width = 481
value_max = 60000000
value_min = 0
```

v-interfaces

- rrdtool info
- rrdtool update**v**
- rrdtool graph**v**

v-interfaces

- rrdtool info
- rrdtool update**v**
- rrdtool graph**v**

v-interfaces

- rrdtool info
- rrdtool update**v**
- rrdtool graph**v**

RRD Caching Daemon

rrdcached — pushing rrd performance

- i/o comes in 4k chunks
- normal update is $\sim$ 100 bytes
- grouping updates = performance for free
- data in memory
- journaling disaster recovery
- simple integration using enviroment variables
- communication via unix socket or ip
- limited security for remote operation
- no updatev support yet
- available with 1.4

created by Florian Forster and Kevin Brintnall

rrdcached — pushing rrd performance

- i/o comes in 4k chunks
- normal update is $\sim$ 100 bytes
- grouping updates = performance for free
- data in memory
- journaling disaster recovery
- simple integration using enviroment variables
- communication via unix socket or ip
- limited security for remote operation
- no updatev support yet
- available with 1.4

created by Florian Forster and Kevin Brintnall

rrdcached — pushing rrd performance

- i/o comes in 4k chunks
- normal update is $\sim$ 100 bytes
- grouping updates = performance for free
- data in memory
- journaling disaster recovery
- simple integration using enviroment variables
- communication via unix socket or ip
- limited security for remote operation
- no updatev support yet
- available with 1.4

created by Florian Forster and Kevin Brintnall

rrdcached — pushing rrd performance

- i/o comes in 4k chunks
- normal update is $\sim$ 100 bytes
- grouping updates = performance for free
- data in memory
- journaling disaster recovery
- simple integration using enviroment variables
- communication via unix socket or ip
- limited security for remote operation
- no updatev support yet
- available with 1.4

created by Florian Forster and Kevin Brintnall

rrdcached — pushing rrd performance

- i/o comes in 4k chunks
- normal update is $\sim$ 100 bytes
- grouping updates = performance for free
- data in memory
- journaling disaster recovery
- simple integration using enviroment variables
- communication via unix socket or ip
- limited security for remote operation
- no updatev support yet
- available with 1.4

created by Florian Forster and Kevin Brintnall

rrdcached — pushing rrd performance

- i/o comes in 4k chunks
- normal update is $\sim$ 100 bytes
- grouping updates = performance for free
- data in memory
- journaling disaster recovery
- simple integration using enviroment variables
- communication via unix socket or ip
- limited security for remote operation
- no updatev support yet
- available with 1.4

created by Florian Forster and Kevin Brintnall

rrdcached — pushing rrd performance

- i/o comes in 4k chunks
- normal update is $\sim$ 100 bytes
- grouping updates = performance for free
- data in memory
- journaling disaster recovery
- simple integration using enviroment variables
- communication via unix socket or ip
- limited security for remote operation
- no updatev support yet
- available with 1.4

created by Florian Forster and Kevin Brintnall

rrdcached — pushing rrd performance

- i/o comes in 4k chunks
- normal update is $\sim$ 100 bytes
- grouping updates = performance for free
- data in memory
- journaling disaster recovery
- simple integration using enviroment variables
- communication via unix socket or ip
- limited security for remote operation
- no updatev support yet
- available with 1.4

created by Florian Forster and Kevin Brintnall

rrdcached — pushing rrd performance

- i/o comes in 4k chunks
- normal update is $\sim$ 100 bytes
- grouping updates = performance for free
- data in memory
- journaling disaster recovery
- simple integration using enviroment variables
- communication via unix socket or ip
- limited security for remote operation
- no updatev support yet
- available with 1.4

created by Florian Forster and Kevin Brintnall

rrdcached — pushing rrd performance

- i/o comes in 4k chunks
- normal update is $\sim$ 100 bytes
- grouping updates = performance for free
- data in memory
- journaling disaster recovery
- simple integration using enviroment variables
- communication via unix socket or ip
- limited security for remote operation
- no updatev support yet
- available with 1.4

created by Florian Forster and Kevin Brintnall

Future

Future RRDtool Features

- full remote support for rrdtool operations
- getopt to popt migration for thread-safety
- portable data format
- in-memory updates for cached to support updatev
- rrd internal journal for i/o optimization
- separation of database and charting features
- json interface and javascript charting frontend

Or anything else someone is willing
to provide time or money for.

Future RRDtool Features

- full remote support for rrdtool operations
- getopt to popt migration for thread-safety
- portable data format
- in-memory updates for cached to support updatev
- rrd internal journal for i/o optimization
- separation of database and charting features
- json interface and javascript charting frontend

Or anything else someone is willing
to provide time or money for.

Future RRDtool Features

- full remote support for rrdtool operations
- getopt to popt migration for thread-safety
- portable data format
- in-memory updates for cached to support updatev
- rrd internal journal for i/o optimization
- separation of database and charting features
- json interface and javascript charting frontend

Or anything else someone is willing
to provide time or money for.

Future RRDtool Features

- full remote support for rrdtool operations
- getopt to popt migration for thread-safety
- portable data format
- in-memory updates for cached to support updatev
- rrd internal journal for i/o optimization
- separation of database and charting features
- json interface and javascript charting frontend

Or anything else someone is willing
to provide time or money for.

Future RRDtool Features

- full remote support for rrdtool operations
- getopt to popt migration for thread-safety
- portable data format
- in-memory updates for cached to support updatev
- rrd internal journal for i/o optimization
- separation of database and charting features
- json interface and javascript charting frontend

Or anything else someone is willing
to provide time or money for.

Future RRDtool Features

- full remote support for rrdtool operations
- getopt to popt migration for thread-safety
- portable data format
- in-memory updates for cached to support updatev
- rrd internal journal for i/o optimization
- separation of database and charting features
- json interface and javascript charting frontend

Or anything else someone is willing
to provide time or money for.

Future RRDtool Features

- full remote support for rrdtool operations
- getopt to popt migration for thread-safety
- portable data format
- in-memory updates for cached to support updatev
- rrd internal journal for i/o optimization
- separation of database and charting features
- json interface and javascript charting frontend

Or anything else someone is willing
to provide time or money for.

Future RRDtool Features

- full remote support for rrdtool operations
- getopt to popt migration for thread-safety
- portable data format
- in-memory updates for cached to support updatev
- rrd internal journal for i/o optimization
- separation of database and charting features
- json interface and javascript charting frontend

Or anything else someone is willing
to provide time or money for.

Future RRDtool Features

- full remote support for rrdtool operations
- getopt to popt migration for thread-safety
- portable data format
- in-memory updates for cached to support updatev
- rrd internal journal for i/o optimization
- separation of database and charting features
- json interface and javascript charting frontend

Or anything else someone is willing
to provide time or money for.

Examples

The size of an rrd - code

```
1  #!/usr/bin/perl
2  sub rrd_sizer {
3    my ($ds_cnt,$rra_sz,$rra_cnt) = @_;
4    system 'rrdtool', 'create', 'sizer.rrd',
5        map({ "DS:d${_}:GAUGE:600:U:U" } 1..$ds_cnt),
6        map({ "RRA:AVERAGE:0.5:1:$rra_sz" } 1..$rra_cnt);
7    my $size = -s 'sizer.rrd';
8    printf "DSs: %1d  RRA Row: %1d  RRAs: %1d == %3d byte\n",
9        $ds_cnt,$rra_sz,$rra_cnt,$size;
10   return $size;
11   }
12 #                          DSs RRAs RRA Rows
13 my $base    = rrd_sizer  1,    1,    1;
14 my $ds      = rrd_sizer  2,    1,    1;
15 my $rra_sz  = rrd_sizer  1,    2,    1;
16 my $rra_cnt = rrd_sizer  1,    1,    2;
17 printf "+1 DS:       %3d byte\n",($ds - $base);
18 printf "+1 RRA Row:  %3d byte\n",($rra_sz - $base);
19 printf "+1 RRA:      %3d byte\n",($rra_cnt - $base);
```

the size of an rrd - result

```
DSs: 1  RRA Row: 1  RRAs: 1 == 552 byte
DSs: 2  RRA Row: 1  RRAs: 1 == 872 byte
DSs: 1  RRA Row: 2  RRAs: 1 == 560 byte
DSs: 1  RRA Row: 1  RRAs: 2 == 752 byte
+1 DS:       320 byte
+1 RRA Row:    8 byte
+1 RRA:      200 byte
```

- overhead is minimal
- 8 byte per double
- . . . per datasource
- . . . per RRA
- . . . per RRA row

the size of an rrd - result

```
DSs: 1  RRA Row: 1  RRAs: 1 == 552 byte
DSs: 2  RRA Row: 1  RRAs: 1 == 872 byte
DSs: 1  RRA Row: 2  RRAs: 1 == 560 byte
DSs: 1  RRA Row: 1  RRAs: 2 == 752 byte
+1 DS:       320 byte
+1 RRA Row:    8 byte
+1 RRA:      200 byte
```

- overhead is minimal
- 8 byte per double
- . . . per datasource
- . . . per RRA
- . . . per RRA row

the size of an rrd - result

```
DSs: 1  RRA Row: 1  RRAs: 1 == 552 byte
DSs: 2  RRA Row: 1  RRAs: 1 == 872 byte
DSs: 1  RRA Row: 2  RRAs: 1 == 560 byte
DSs: 1  RRA Row: 1  RRAs: 2 == 752 byte
+1 DS:       320 byte
+1 RRA Row:    8 byte
+1 RRA:      200 byte
```

- overhead is minimal
- 8 byte per double
- . . . per datasource
- . . . per RRA
- . . . per RRA row

the size of an rrd - result

```
DSs: 1  RRA Row: 1  RRAs: 1 == 552 byte
DSs: 2  RRA Row: 1  RRAs: 1 == 872 byte
DSs: 1  RRA Row: 2  RRAs: 1 == 560 byte
DSs: 1  RRA Row: 1  RRAs: 2 == 752 byte
+1 DS:       320 byte
+1 RRA Row:    8 byte
+1 RRA:      200 byte
```

- overhead is minimal
- 8 byte per double
- . . . per datasource
- . . . per RRA
- . . . per RRA row

the size of an rrd - result

```
DSs: 1  RRA Row: 1  RRAs: 1 == 552 byte
DSs: 2  RRA Row: 1  RRAs: 1 == 872 byte
DSs: 1  RRA Row: 2  RRAs: 1 == 560 byte
DSs: 1  RRA Row: 1  RRAs: 2 == 752 byte
+1 DS:       320 byte
+1 RRA Row:    8 byte
+1 RRA:      200 byte
```

- overhead is minimal
- 8 byte per double
- . . . per datasource
- . . . per RRA
- . . . per RRA row

Real Live Example

Traffic Stats for oss.oetiker.ch - Mozilla Firefox

File Edit View History Bookmarks Tools Help

SB Gedafe Google Dict http://oss.oe float i

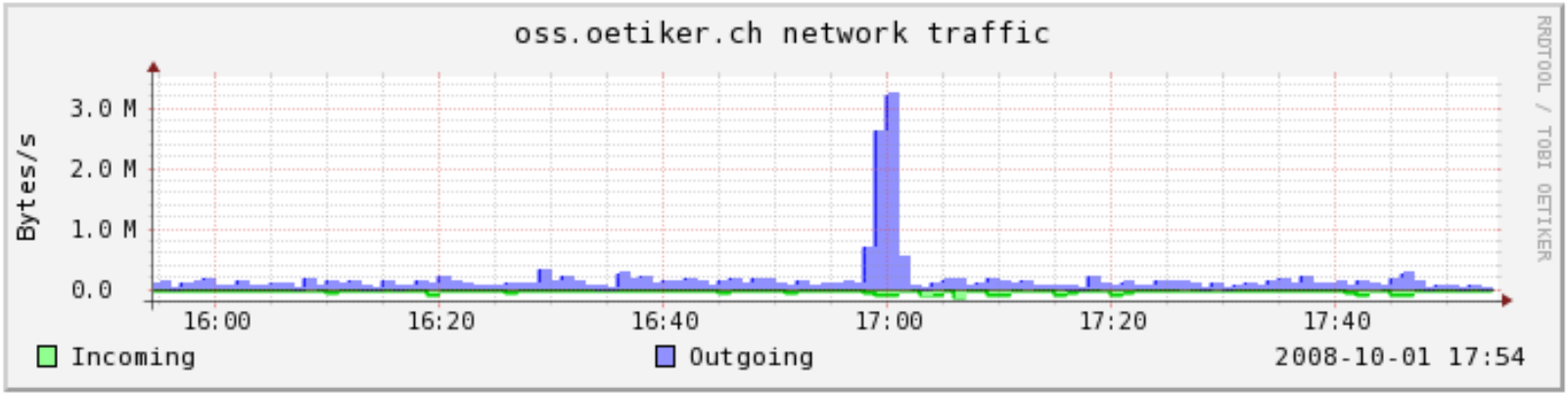
oogle josephse... Notes on... lisa2000... InstaMap... Faceboo... Traffi...

Traffic Stats for oss.oetiker.ch

The Bytes

Period	Incoming	Outgoing	Total
2008-10-01	1.25 G	5.52 G	6.77 G
2008-10	1.21 G	5.36 G	6.56 G
2008	500.08 G	2356.76 G	2856.84 G

Current



Done

data acquisition I

```
#!/bin/sh
# use from cron
# * * * * *  /path/to/ifbyteget.sh eth0

PATH=/bin:/usr/bin
export PATH

cd /home/oposs/public_html/stats

if [ ! -f $1.rrd ]; then

rrdtool create $1.rrd \
   --step=60 \
   DS:in:DERIVE:70:0:100000000 \
   DS:out:DERIVE:70:0:100000000 \
   RRA:AVERAGE:0.5:1:1500 \
   RRA:AVERAGE:0.5:60:10000 \
   RRA:MIN:0.5:60:10000 \
   RRA:MAX:0.5:60:10000 \
   RRA:AVERAGE:0.5:1440:1000 \
```

data acquisition II

```
21      RRA:MIN:0.5:1440:1000 \
22      RRA:MAX:0.5:1440:1000
23  fi
24
25  rrdtool update $1.rrd \
26      N:`grep $1: /proc/net/dev \
27          | sed 's/.*://' | awk '{print $1":"$9}'`
```

rrdcgi: scripting for the poor I

```
#!/usr/bin/env rrdcgi
<html>
<head>
<title>Traffic Stats for oss.oetiker.ch</title>
</head>
<body>
<h1>Traffic Stats for oss.oetiker.ch</h1>

<h2>The Bytes</h2>
<table border="1" cellspacing="0" cellpadding="2">
<tr><td>Period</td>
    <td>Incoming</td>
    <td>Outgoing</td>
    <td>Total</td></tr>

<!--
<RRD::GRAPH -
        --start="midnight"
        --end="start+24h"
        --imginfo=" "
```

rrdcgi: scripting for the poor II

```
21          DEF:in=lan.rrd:in:AVERAGE:step=1800
22          DEF:out=lan.rrd:out:AVERAGE:step=1800
23          VDEF:is=in,TOTAL
24          PRINT:is:"%0.2lf %s"
25          VDEF:os=out,TOTAL
26          PRINT:os:"%0.2lf %S"
27          CDEF:sum=in,out,+
28          VDEF:ss=sum,TOTAL
29          PRINT:ss:"%0.2lf %S"
30  >
31  -->
32
33  <tr><td><RRD::TIME::NOW %Y-%m-%d></td>
34      <td align="right"><RRD::PRINT 0></td>
35      <td align="right"><RRD::PRINT 1></td>
36      <td align="right"><RRD::PRINT 2></td></tr>
37
38  <!--
39  <RRD::GRAPH -
40          --start="<RRD::TIME::NOW %Y%m01>"
41          --end="now"
```

rrdcgi: scripting for the poor III

```
42          --imginfo=" "
43          DEF:in=lan.rrd:in:AVERAGE:step=1800
44          DEF:out=lan.rrd:out:AVERAGE:step=1800
45          VDEF:is=in,TOTAL
46          PRINT:is:"%0.2lf %s"
47          VDEF:os=out,TOTAL
48          PRINT:os:"%0.2lf %S"
49          CDEF:sum=in,out,+
50          VDEF:ss=sum,TOTAL
51          PRINT:ss:"%0.2lf %S"
52  >
53  -->
54
55  <tr><td><RRD::TIME::NOW %Y-%m></td>
56      <td align="right"><RRD::PRINT 0></td>
57      <td align="right"><RRD::PRINT 1></td>
58      <td align="right"><RRD::PRINT 2></td></tr>
59
60  <!--
61  <RRD::GRAPH -
62          --start="<RRD::TIME::NOW %Y0101>"
```

rrdcgi: scripting for the poor IV

```
63          --end="now"
64          --imginfo=" "
65          DEF:in=lan.rrd:in:AVERAGE:step=1800
66          DEF:out=lan.rrd:out:AVERAGE:step=1800
67          VDEF:is=in,TOTAL
68          PRINT:is:"%0.2lf %s"
69          VDEF:os=out,TOTAL
70          PRINT:os:"%0.2lf %S"
71          CDEF:sum=in,out,+
72          VDEF:ss=sum,TOTAL
73          PRINT:ss:"%0.2lf %S"
74  >
75  -->
76
77  <tr><td><RRD::TIME::NOW %Y></td>
78      <td align="right"><RRD::PRINT 0></td>
79      <td align="right"><RRD::PRINT 1></td>
80      <td align="right"><RRD::PRINT 2></td></tr>
81  </table>
82
83  <h2>Current</h2>
```

rrdcgi: scripting for the poor V

```
84
85  <RRD::SETVAR start -2h>
86  <RRD::SETVAR end now>
87  <RRD::INCLUDE graph.inc>
88
89  <h2>Day</h2>
90
91  <RRD::SETVAR start -24h>
92  <RRD::SETVAR end now>
93  <RRD::INCLUDE graph.inc>
94
95  <h2>7 Days</h2>
96
97  <RRD::SETVAR start -7d>
98  <RRD::SETVAR end now>
99  <RRD::INCLUDE graph.inc>
100
101 <h2>Month</h2>
102
103 <RRD::SETVAR start -30d>
104 <RRD::SETVAR end now>
```

rrdcgi: scripting for the poor VI

```
105  <RRD::INCLUDE graph.inc>
106
107  <h2>This Year</h2>
108
109  <RRD::SETVAR start "Jan1">
110  <RRD::SETVAR end   "Dec31">
111  <RRD::INCLUDE graph.inc>
112
113  <h2>Last Year</h2>
114
115  <RRD::SETVAR start "Jan1-365d">
116  <RRD::SETVAR end   "Dec31-365d">
117  <RRD::INCLUDE graph.inc>
118
119  </body>
120  </html>
```

rrdcgi: include file function I

```
<p>
<RRD::GRAPH lan<RRD::GETVAR start>.png
        --title="oss.oetiker.ch network traffic"
        --vertical-label=Bytes/s
        --start="<RRD::GETVAR start>"
        --end="<RRD::GETVAR end>"
        --width=600
        --height=100
        DEF:in=lan.rrd:in:AVERAGE
        CDEF:nin=in,-1,*
        LINE1.5:nin#00d000
        AREA:nin#90ff90:Incoming
        DEF:out=lan.rrd:out:AVERAGE
        LINE1.5:out#2020ff
        AREA:out#9090ff:Outgoing
        LINE0.5:0#000
        COMMENT:"<RRD::TIME::NOW '%Y-%m-%d %H\:%M'>\j"
>
</p>
```

?

Tobi Oetiker <tobi@oetiker.ch>