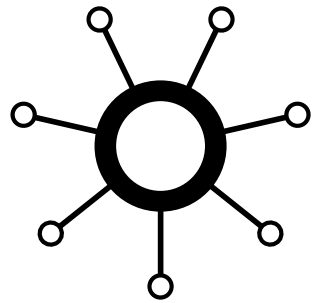
Monitoring mit **collect**d und Nagios

Lass dir sagen, was deine Rechner machen

Sebastian „tokkee“ Harl <sh@teamix.net>

team(ix) GmbH / **collect**d core team

Nagios Workshop 2011
23. Mai 2011, Hannover



Über team(ix)

- gegründet 2001
- Spezialisierung auf solide Infrastruktur
- Ursprünge: Open-Source und Netzwerke
  - Debian
  - Nagios
  - Schulungen
  - u.v.m.
- Heute auch:
  - NetApp
  - VMWare
  - Riverbed (WAN-Beschleunigung)
  - Juniper
  - N-IX (Nürnberger Internet-eXchange)

Warum **collect**d und Nagios?

- collectd ist auf das effiziente Sammeln und Speichern von Performance-Daten ausgelegt
- Nagios kann davon profitieren und sich auf das eigentliche Monitoring konzentrieren

Warum **collect**d und Nagios?

- collectd ist auf das effiziente Sammeln und Speichern von Performance-Daten ausgelegt
- Nagios kann davon profitieren und sich auf das eigentliche Monitoring konzentrieren

- **WIP!**
- **Diskussion, Kommentare und Rants erwünscht!** ☺

Was ist **collect**d?

- **collect**d sammelt Leistungsdaten von Rechnern
- Leistungsdaten sind zum Beispiel:
  - CPU-Auslastung
  - Speichernutzung
  - Netzwerkverkehr
- Daten werden erhoben, verarbeitet und gespeichert
- Häufig: Darstellung als Graphen
- $\rightarrow$ Performance-Analyse, Kapazitätsplanung
- Nicht verwechseln mit *Monitoring!*
- Homepage: `http://collectd.org/`

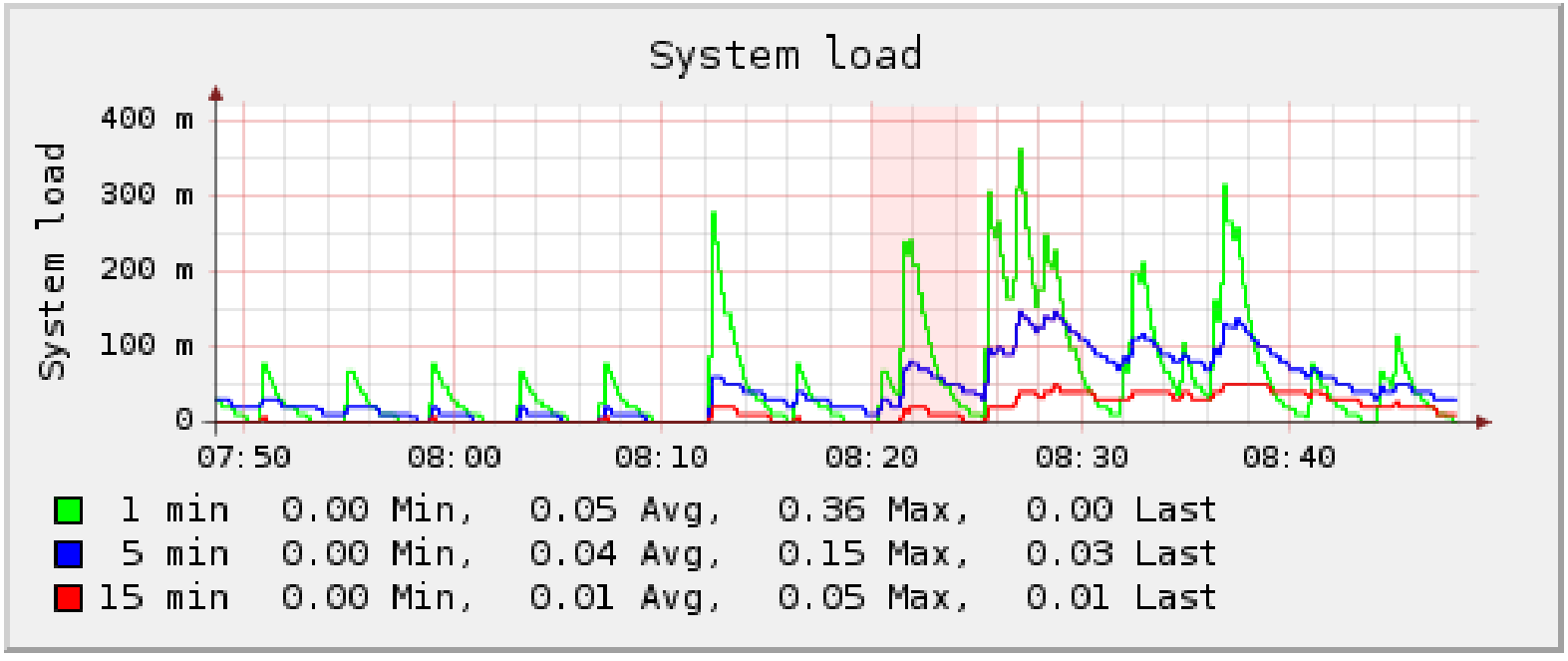
Wichtige Eigenschaften

- Daemon
- Freie Software (größtenteils GPLv2)
- Portierbar (Linux, *BSD, Solaris, . . .)
- Skalierbar (OpenWrt, . . . , Cluster / Cloud)
- Effizient (Default-Auflösung: 10 Sekunden)
- Modular (über 100 Plugins in Version 5.0)

Wichtige Eigenschaften

- Daemon
- Freie Software (größtenteils GPLv2)
- Portierbar (Linux, *BSD, Solaris, . . .)
- Skalierbar (OpenWrt, . . . , Cluster / Cloud)
- Effizient (Default-Auflösung: 10 Sekunden)
- Modular (über 100 Plugins in Version 5.0)

10-Sekunden-Auflösung



Wichtige Eigenschaften

- Daemon
- Freie Software (größtenteils GPLv2)
- Portierbar (Linux, *BSD, Solaris, . . .)
- Skalierbar (OpenWrt, . . . , Cluster / Cloud)
- Effizient (Default-Auflösung: 10 Sekunden)
- Modular (über 100 Plugins in Version 5.0)

Wichtige Eigenschaften

- Daemon
- Freie Software (größtenteils GPLv2)
- Portierbar (Linux, *BSD, Solaris, . . .)
- Skalierbar (OpenWrt, . . . , Cluster / Cloud)
- Effizient (Default-Auflösung: 10 Sekunden)
- Modular (über 100 Plugins in Version 5.0)

Verfügbare Plugins (Auswahl)

apache	amqp	apcups	ascent	battery
bind	conntrack	contextswitch	cpu	cpufreq
csv	curl	curl_json	dbi	df
disk	dns	email	entropy	exec
filecount	fscache	GenericJMX	gmond	hddtemp
interface	ipmi	iptables	ipvs	irq
java	libvirt	load	logfile	madwifi
match_regex	mbmon	memcachec	memcached	memory
Monitorus	multimeter	mysql	netapp	netlink
network	nfs	nginx	notify_email	ntpd
nut	olsrd	onewire	openvpn	OpenVZ
oracle	perl	ping	postgresql	powerdns
processes	protocols	python	routeros	rrdcached
rrdtool	sensors	serial	snmp	swap
syslog	table	tail	tape	target_scale
tcpconns	teamspeak2	ted	thermal	tokyotyrant
unixsock	uptime	users	uuid	vmem
vserver	wireless	write_http	xmms	zfs_arc

- Daemon läuft auf jedem Client (Ausnahme: SNMP o.ä.)
- üblicherweise: ein oder mehrere zentrale Server, die Werte von Clients empfangen (Push-Modell)
- First steps: `install; select plugins; start daemon; enjoy` ;-)

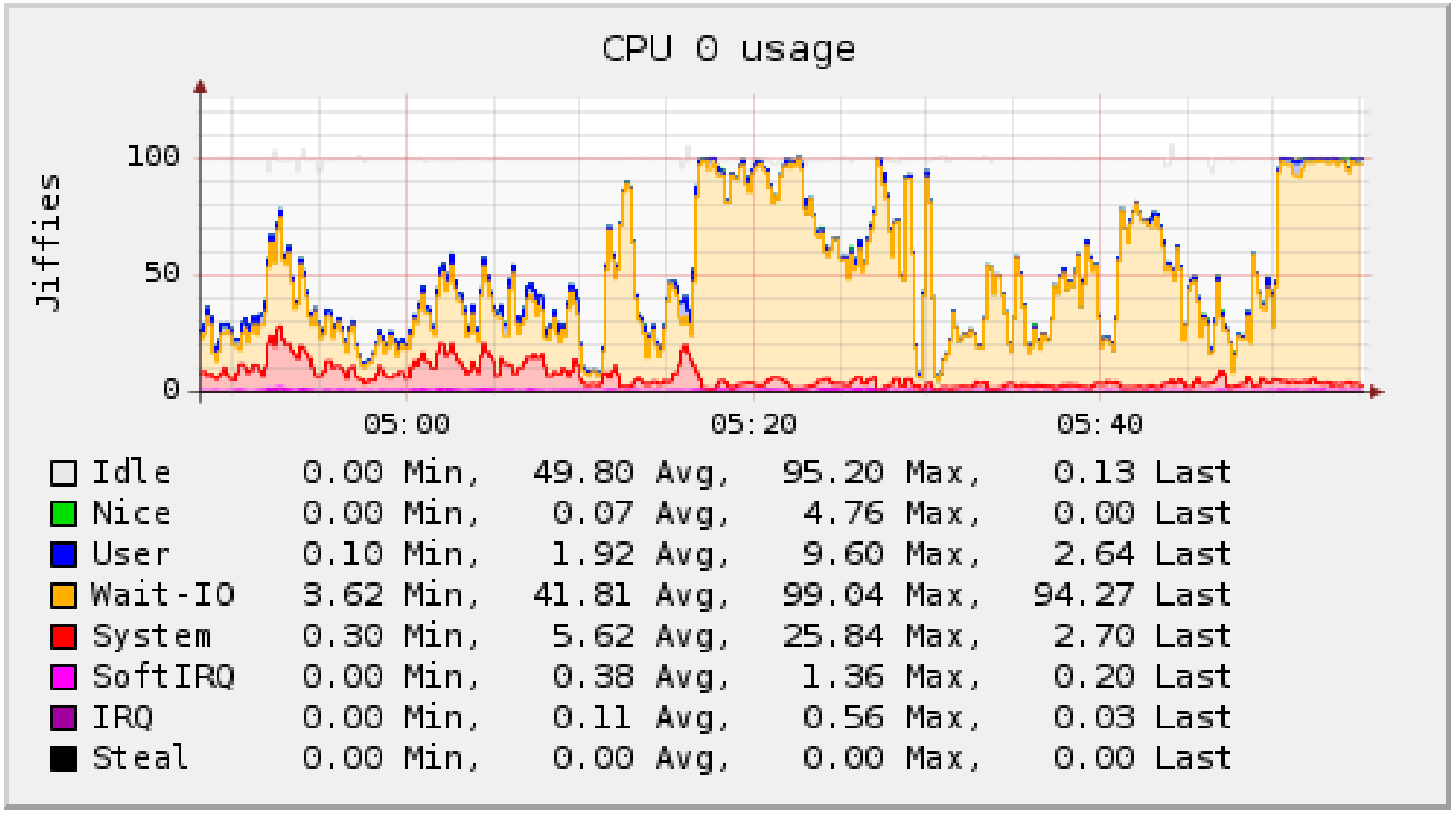
Synopsis

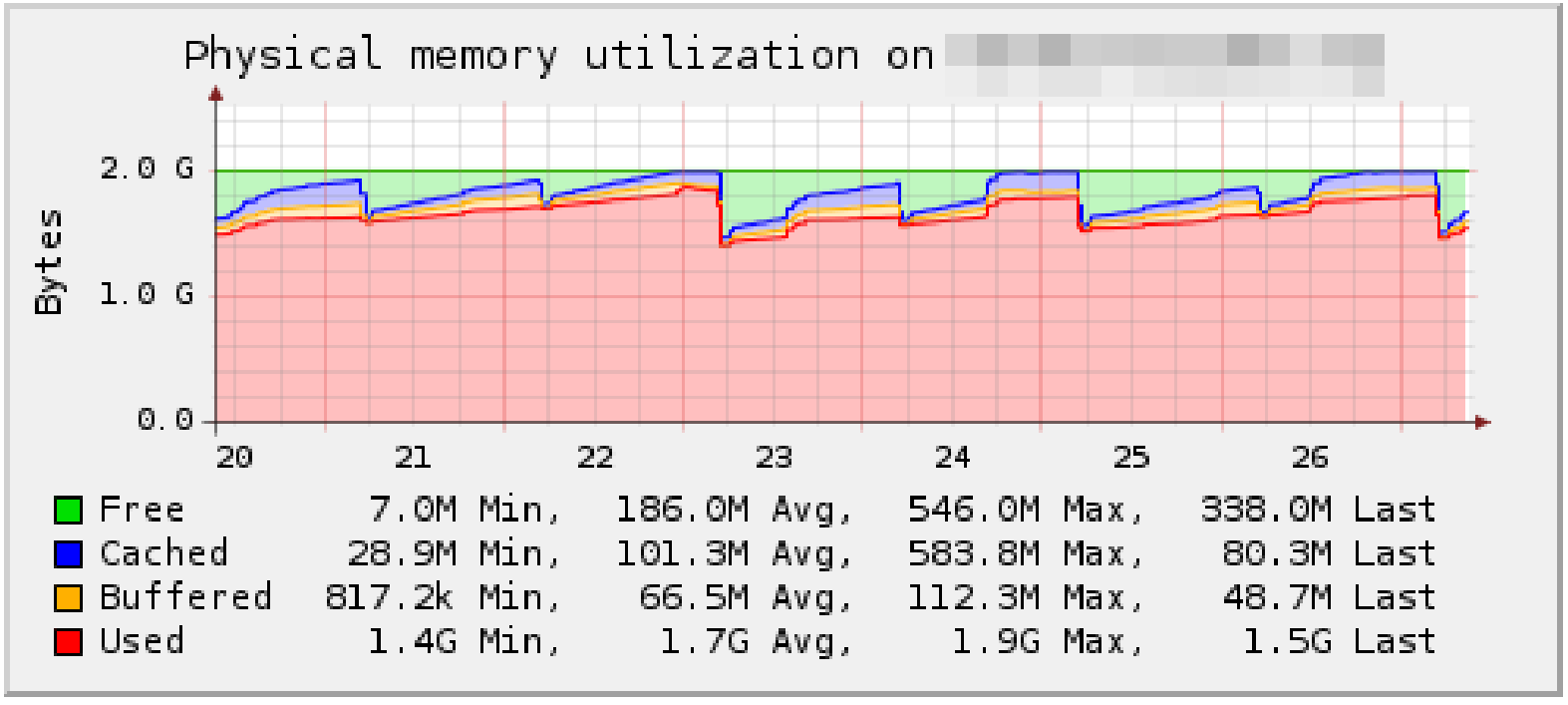
```
LoadPlugin "cpu"
LoadPlugin "memory"
LoadPlugin "interface"
```

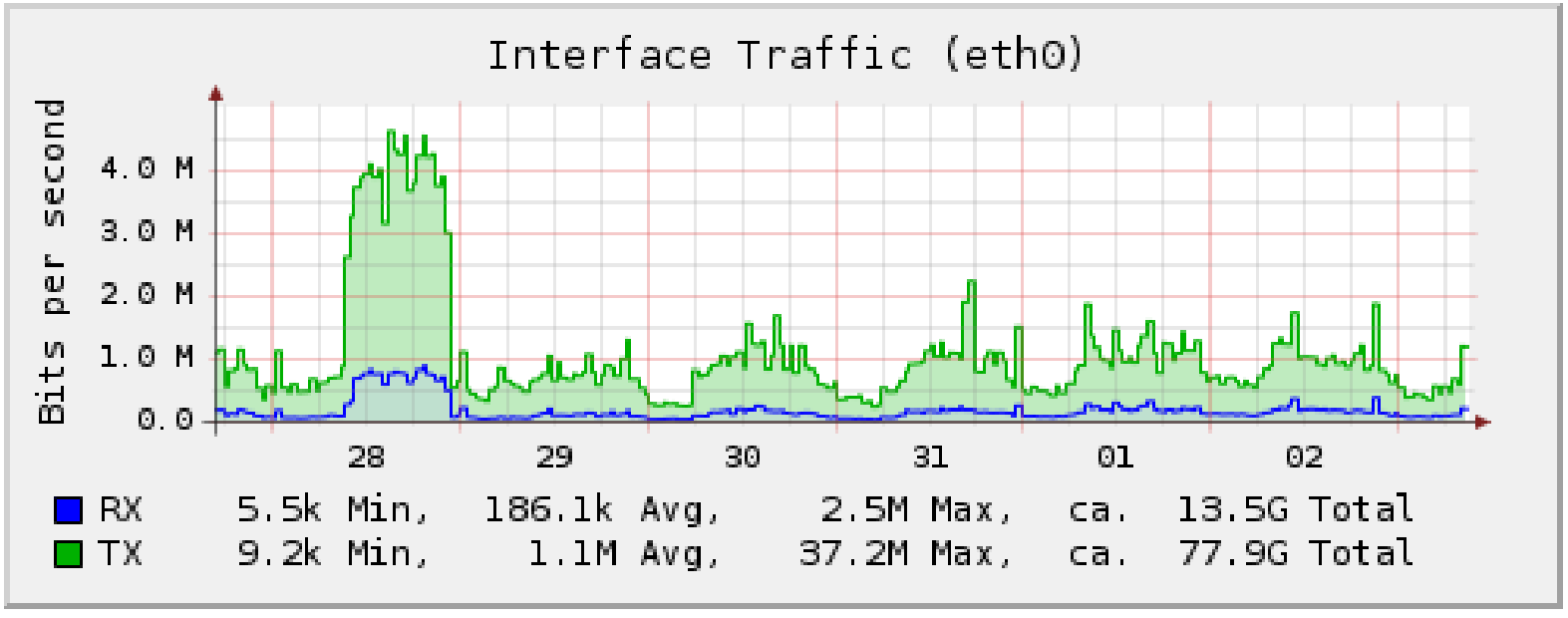
Synopsis

```
LoadPlugin "cpu"
LoadPlugin "memory"
LoadPlugin "interface"

<Plugin interface>
  Interface lo
  Interface sit0
  IgnoreSelected true
</Plugin>
```


CPU 0 usage
Jiffies
100
50
0
05:00
05:20
05:40
Idle 0.00 Min, 49.80 Avg, 95.20 Max, 0.13 Last
Nice 0.00 Min, 0.07 Avg, 4.76 Max, 0.00 Last
User 0.10 Min, 1.92 Avg, 9.60 Max, 2.64 Last
Wait-IO 3.62 Min, 41.81 Avg, 99.04 Max, 94.27 Last
System 0.30 Min, 5.62 Avg, 25.84 Max, 2.70 Last
SoftIRQ 0.00 Min, 0.38 Avg, 1.36 Max, 0.20 Last
IRQ 0.00 Min, 0.11 Avg, 0.56 Max, 0.03 Last
Steal 0.00 Min, 0.00 Avg, 0.00 Max, 0.00 Last


Physical memory utilization on
Bytes
2.0 G
1.0 G
0.0
20
21
22
23
24
25
26
Free 7.0M Min, 186.0M Avg, 546.0M Max, 338.0M Last
Cached 28.9M Min, 101.3M Avg, 583.8M Max, 80.3M Last
Buffered 817.2k Min, 66.5M Avg, 112.3M Max, 48.7M Last
Used 1.4G Min, 1.7G Avg, 1.9G Max, 1.5G Last


Interface Traffic (eth0)
Bits per second
4.0 M
3.0 M
2.0 M
1.0 M
0.0
28
29
30
31
01
02
RX 5.5k Min, 186.1k Avg, 2.5M Max, ca. 13.5G Total
TX 9.2k Min, 1.1M Avg, 37.2M Max, ca. 77.9G Total

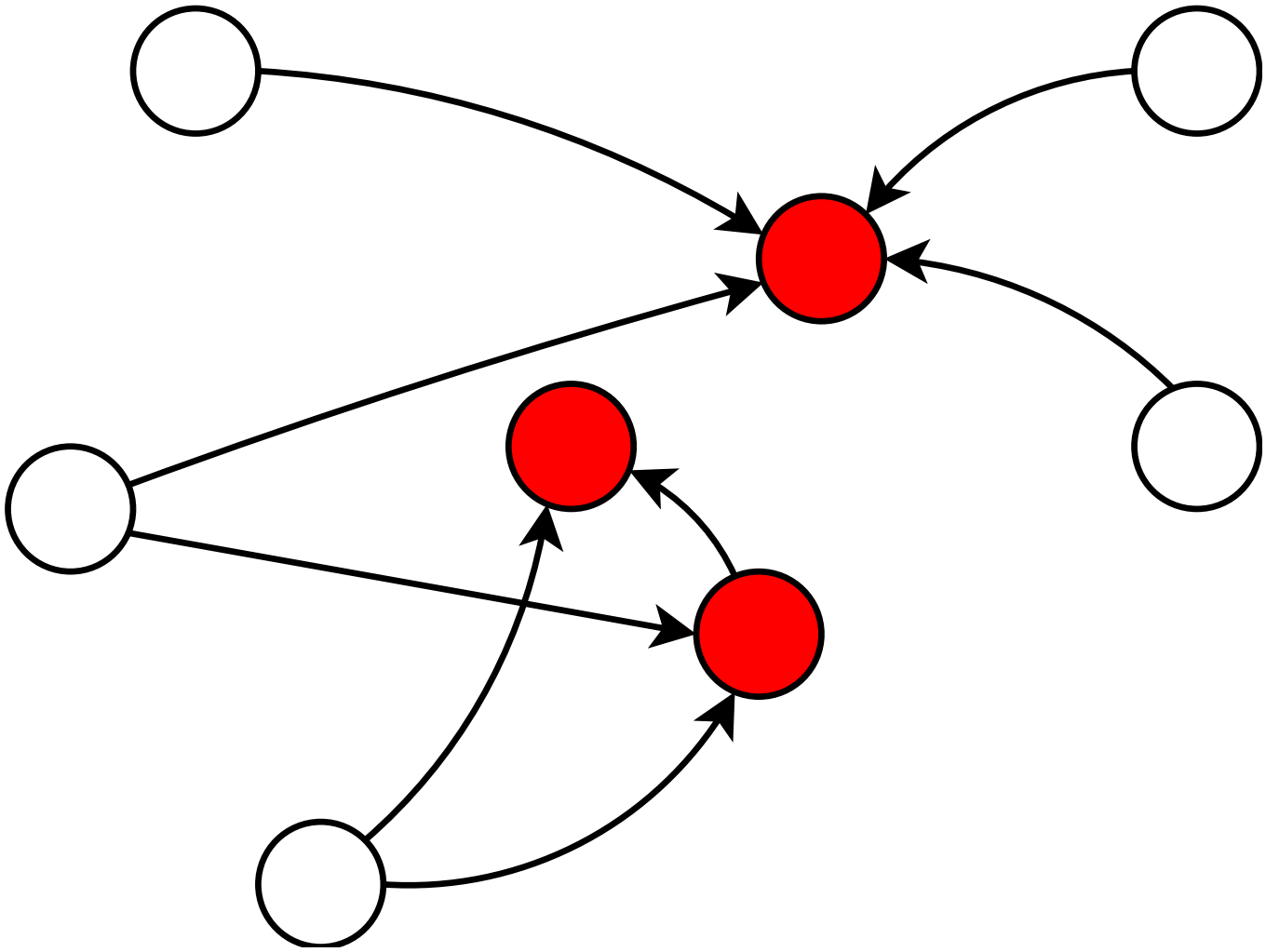
Betriebsarten

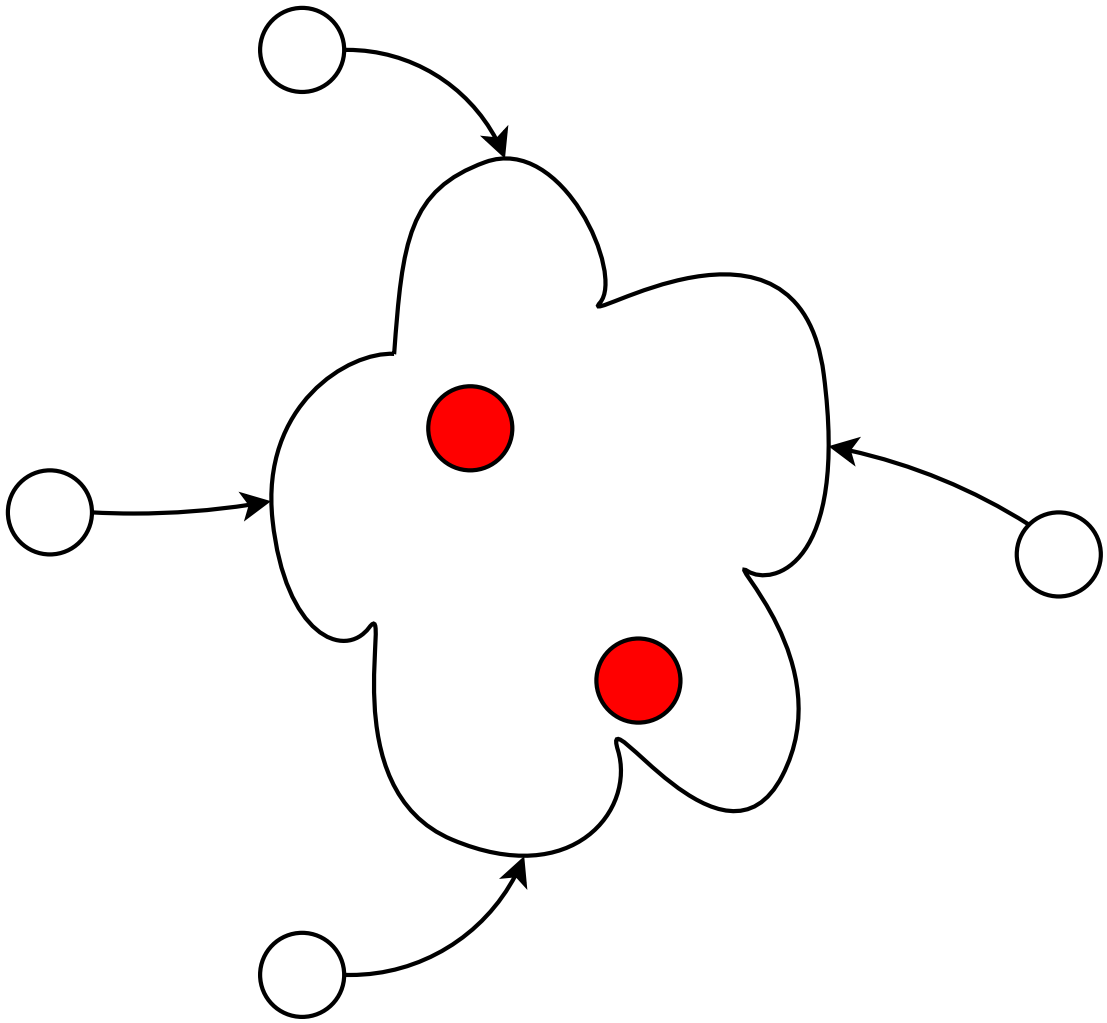
- Daten versenden *(„Client“)*
- Daten empfangen *(„Server“)*
- Weiterleiten *(„Proxy“)*
- Unicast *(„Punkt-zu-Punkt“)*
- Multicast *(„Punkt-zu-Gruppe“)*
- IPv4 und IPv6

Ein Daemon für alles

Rolle des Daemon hängt von der Konfiguration ab.

Netzwerk-Plugin: Unicast





Synopsis: Client

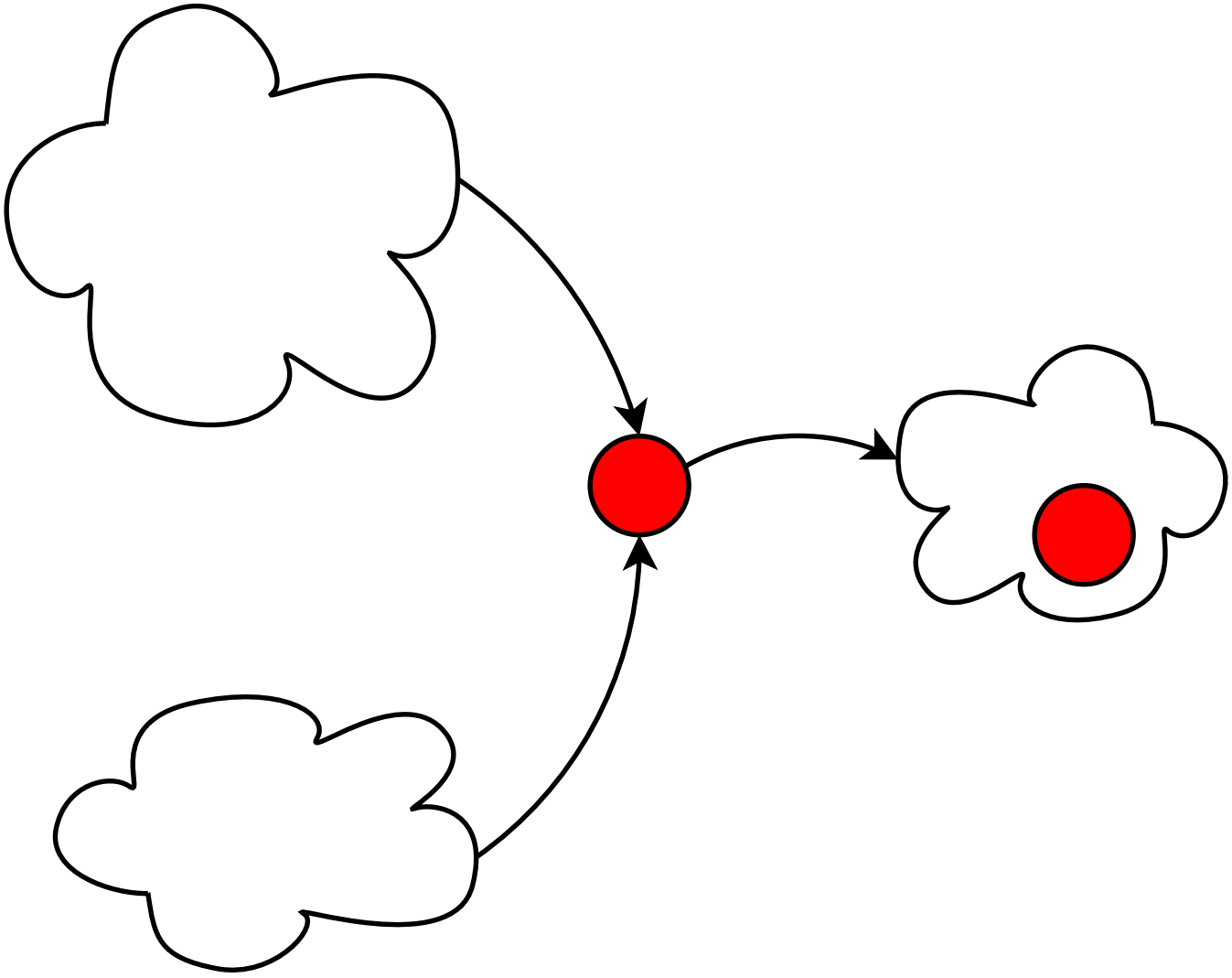
```
LoadPlugin "network"

<Plugin "network">
  Server "collectd0.musterfirma.de"
  Server "collectd1.musterfirma.de"
  Server "ff18::efc0:4a42"
</Plugin>
```

Synopsis: Server

```
LoadPlugin "network"

<Plugin "network">
  Listen "collectd0.musterfirma.de"
  Listen "ff18::efc0:4a42"
</Plugin>
```



Synopsis: Proxy

```
LoadPlugin "network"

<Plugin "network">
  Listen "collectgw.extern.musterfirma.de"
  Server "collectd1.intern.musterfirma.de"
  Forward true
</Plugin>
```

- Schreibt Daten effizient in RRD-Dateien → Caching
- Funktionalität nun in RRDtool als RRD Caching Daemon verfügbar

Synopsis

```
LoadPlugin "rrdtool"

<Plugin "rrdtool">
  DataDir "/var/lib/collectd/rrd"
</Plugin>
```

Konfiguration

```
<Plugin "rrdtool">
  DataDir "/var/lib/collectd/rrd"

  CacheTimeout 3600  # 1 hour
  CacheFlush 86400   # 1 day

  WritesPerSecond 30
</Plugin>
```

- `FLUSH` ermöglicht dennoch die graphische Darstellung von aktuellen Daten

UNIXSOCK Plugin

- öffnet einen UNIX-Domain-Socket zur Interaktion mit dem Daemon
- Text- und Zeilen-basiertes Protokoll
  (z. B. `PUTVAL`, `FLUSH`, `LISTVAL`, `GETVAL`)
- ermöglicht, u.a., die Abfrage von Werten
- `collectd-nagios`
- `collectdctl` (ab Version 5.0)
- `cussh.pl`: *„collectd UNIX socket shell“*

UNIXSOCK Plugin: Beispiel


```
-> | GETVAL "FQDN/load/load"
<- | 3 Values found
<- | shortterm=4.000000e-02
<- | midterm=6.000000e-02
<- | longterm=7.000000e-02
```

Namensschema

- jeder Datensatz hat einen eindeutigen Identifier
  - Hostname
  - Plugin Name
  - Plugin Instanz (optional)
  - Typ
  - Typ Instanz (optional)
- `hostname/plugin[-instanz]/typ[-instanz]`

- Beispiel: `localhost/cpu-0/cpu-idle`

Verfügbare Datensätze

- `cd /var/lib/collectd; ls **/*.rrd`
- `collectdctl listval`

```
cpu-0/cpu-idle
cpu-0/cpu-interrupt
cpu-0/cpu-nice
cpu-0/cpu-softirq
cpu-0/cpu-steal
cpu-0/cpu-system
cpu-0/cpu-user
cpu-0/cpu-wait
interface-eth0/if_errors
interface-eth0/if_octets
interface-eth0/if_packets
load/load
memory/memory-buffered
memory/memory-cached
memory/memory-free
memory/memory-used
users/users
...
```

collectd-nagios

- Nagios-Plugin
- verbindet sich zum UNIX Socket von collectd
- fragt einen (WIP: mehrere) Datensatz ab
- überprüft übergebene Schwellwerte
- kehrt mit Nagios-kompatiblen Rückgabewert zurück

collectd-nagios

```
% collectd-nagios -h
Usage: collectd-nagios <-s socket> <-n value_spec> <-H hostname> [options]

Valid options are:
  -s <socket>    Path to collectd's UNIX-socket.
  -n <v_spec>    Value specification to get from collectd.
                 Format: 'plugin-instance/type-instance'
  -d <ds>        Select the DS to examine. May be repeated
                 to examine multiple
                 DSes. By default all DSes are used.
  -g <consol>    Method to use to consolidate several DSes.
                 See below for a list of valid arguments.
  -H <host>      Hostname to query the values for.
  -c <range>     Critical range
  -w <range>     Warning range
```

collectd-nagios: Beispiel

```
# adduser nagios collectd
# su - nagios

% collectd-nagios -s /var/run/collectd-unixsock \
                  -H FQDN -n load/load
OKAY: 0 critical, 0 warning, 3 okay |
  shortterm=0.120000;;;;
  midterm=0.130000;;;;
  longterm=0.180000;;;;
```

(Umbrüche nur auf den Folien)

collectd-nagios: Beispiel

```
% collectd-nagios -s /var/run/collectd-unixsock \
                  -H FQDN -n df/df-root \
                  -d free -d used -g percentage
OKAY: 31.289281 percent |
  free=46196220000.000000;;;;
  used=101446100000.000000;;;;
```

(Umbrüche nur auf den Folien)

Integration in Nagios: Commands


```
define command {
  command_name check_collectd
  command_line /.../collectd-nagios
               -s /.../collectd-unixsock
               -H $HOSTNAME$ -n $ARG1$
               -d $ARG2$
               -w $ARG3$ -c $ARG4$
}
```

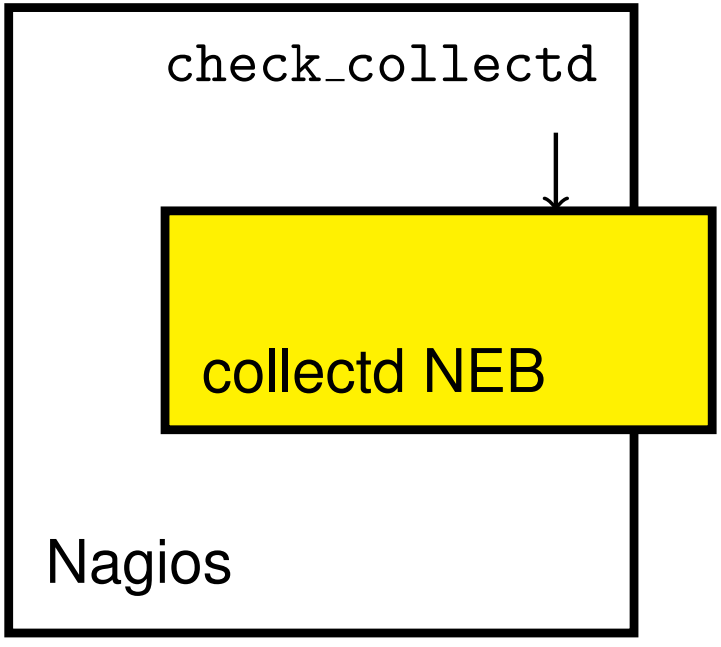
Integration in Nagios: Commands

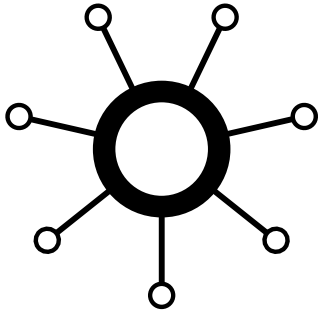

```
define command {
  command_name check_collectd_percentage
  command_line /.../collectd-nagios
               -s /.../collectd-unixsock
               -H $HOSTNAME$ -n $ARG1$
               -g percentage
               -d $ARG2$ -d $ARG3$
               -w $ARG4$ -c $ARG5$
}
```

Integration in Nagios: Services

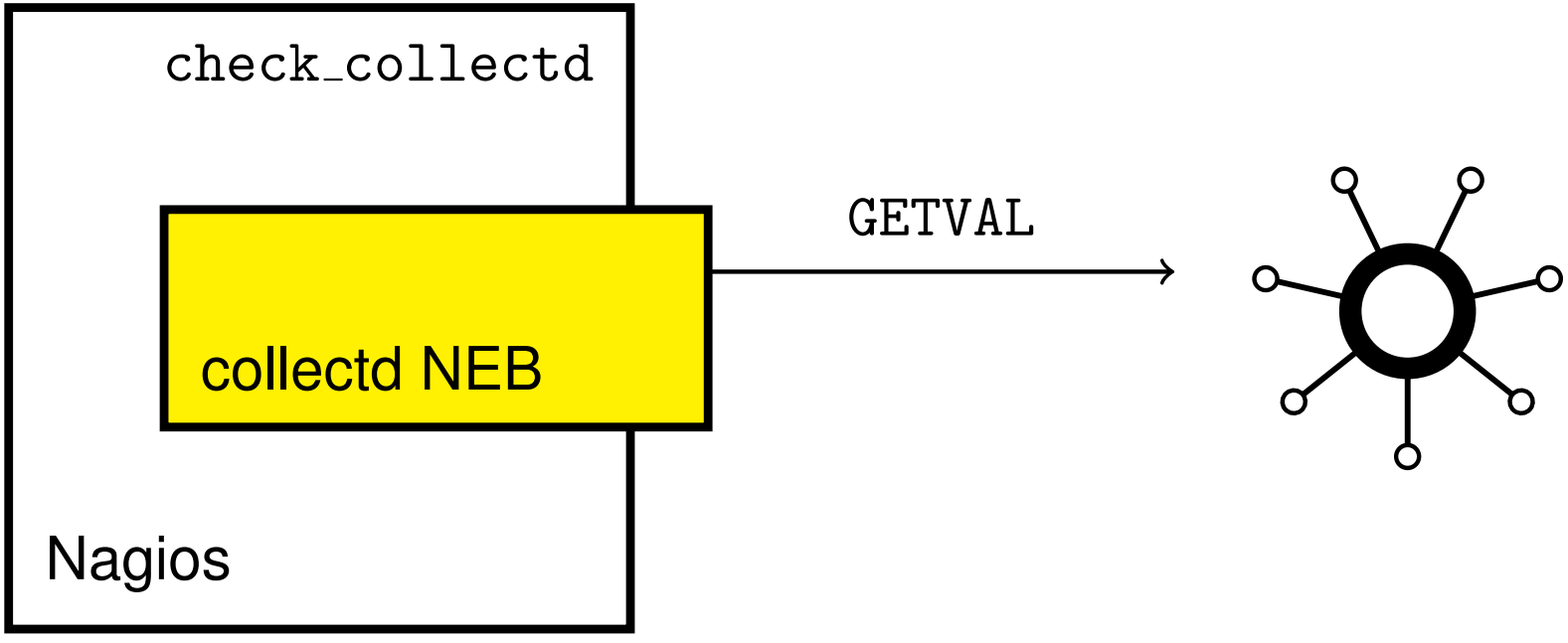

```
check_collectd_percentage!df/df-root!free!used!20:!10:
check_collectd!load/load!midterm!5!10
check_collectd!users/users!users!50!70
check_collectd!postgresql-webapp/users-loggedin!users!...
check_collectd!tail-postfix/gauge-connections_in!value!...
```

Zukunftsmusik: aktiver Modus

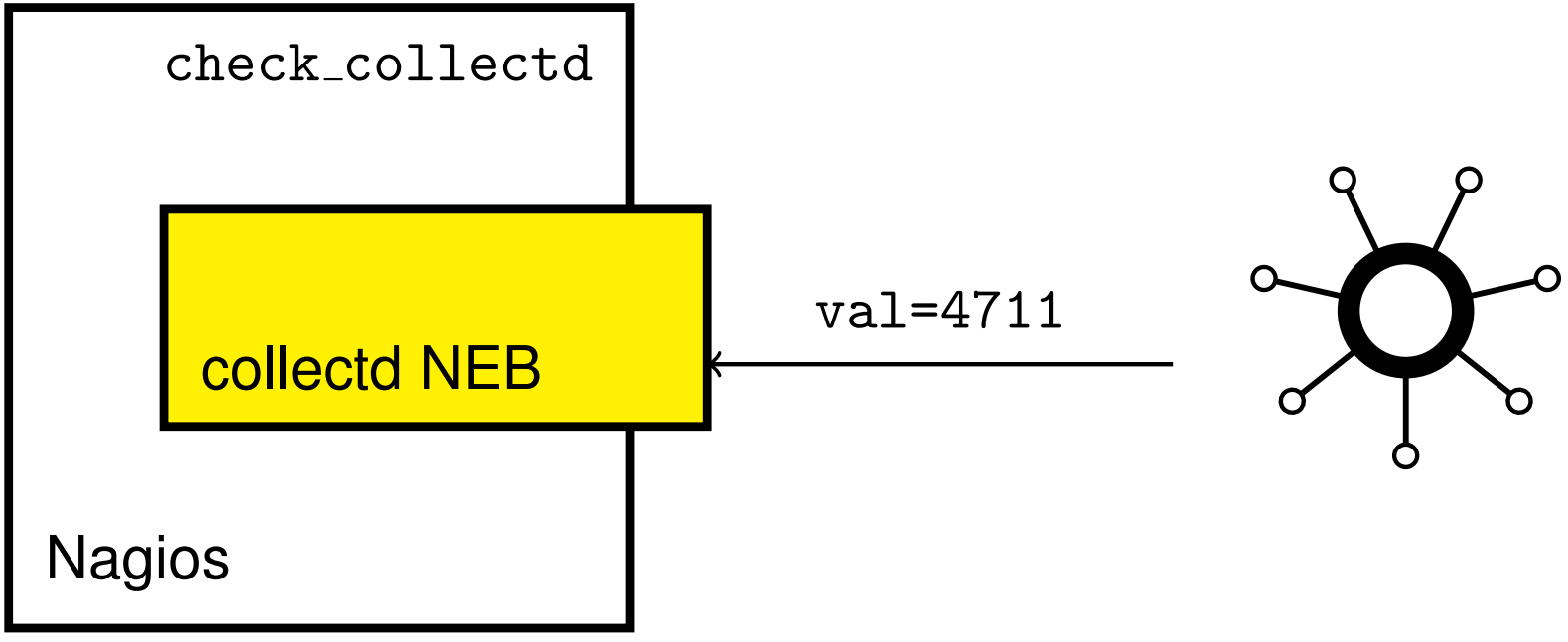




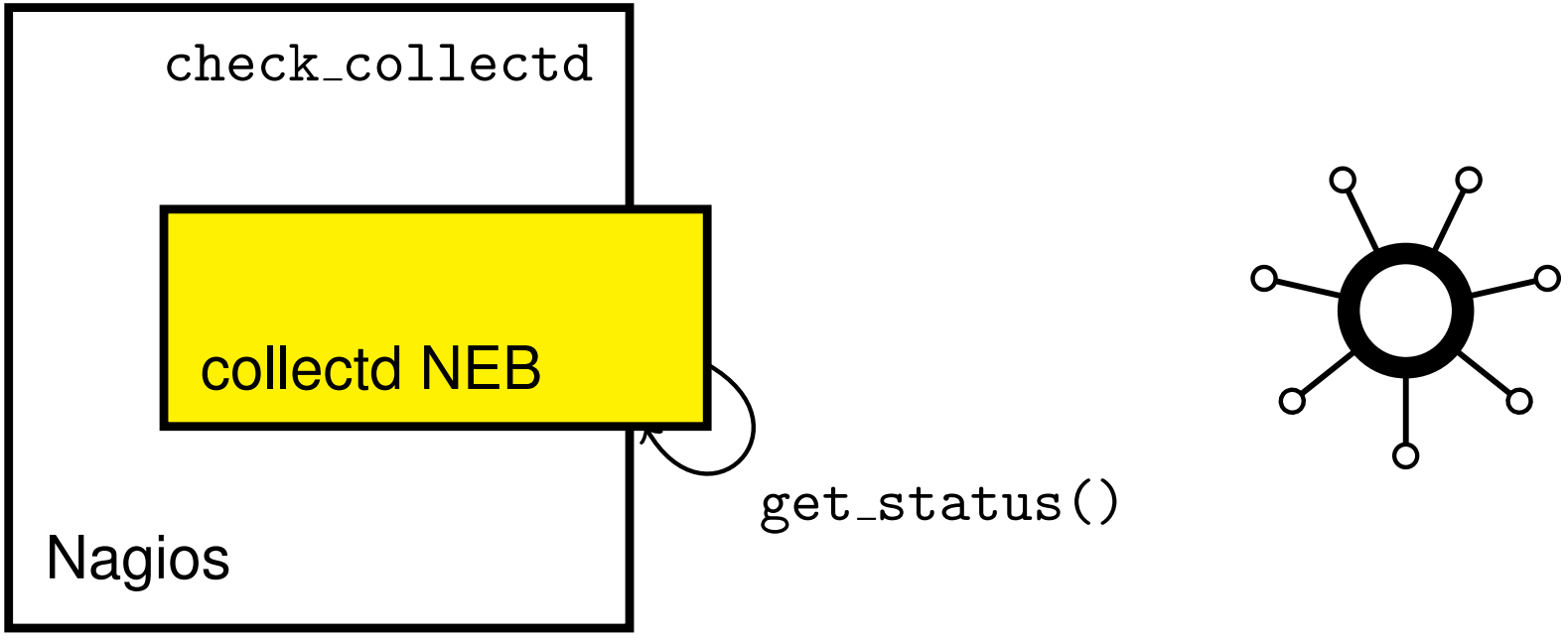
Zukunftsmusik: aktiver Modus



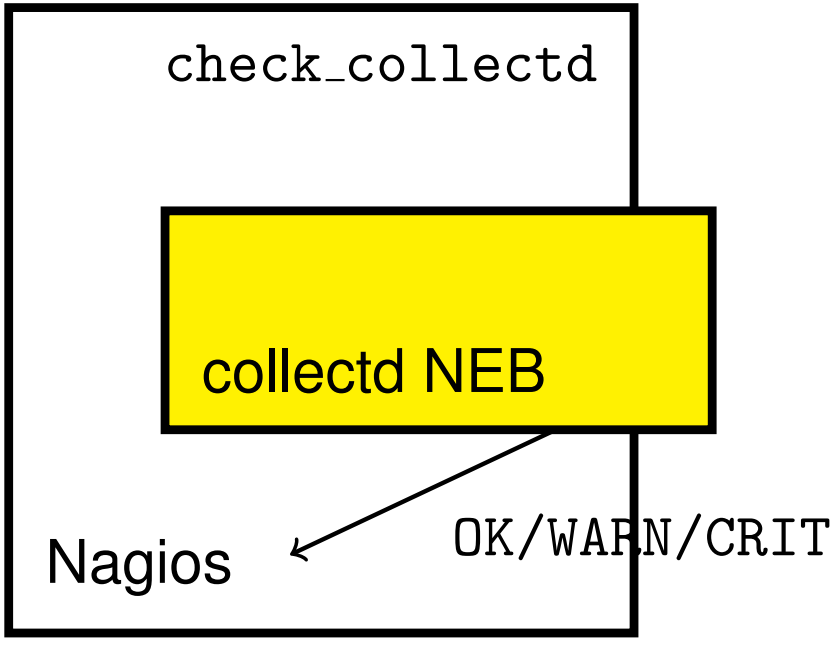
Zukunftsmusik: aktiver Modus

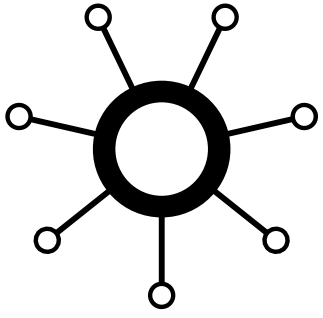


Zukunftsmusik: aktiver Modus



Zukunftsmusik: aktiver Modus

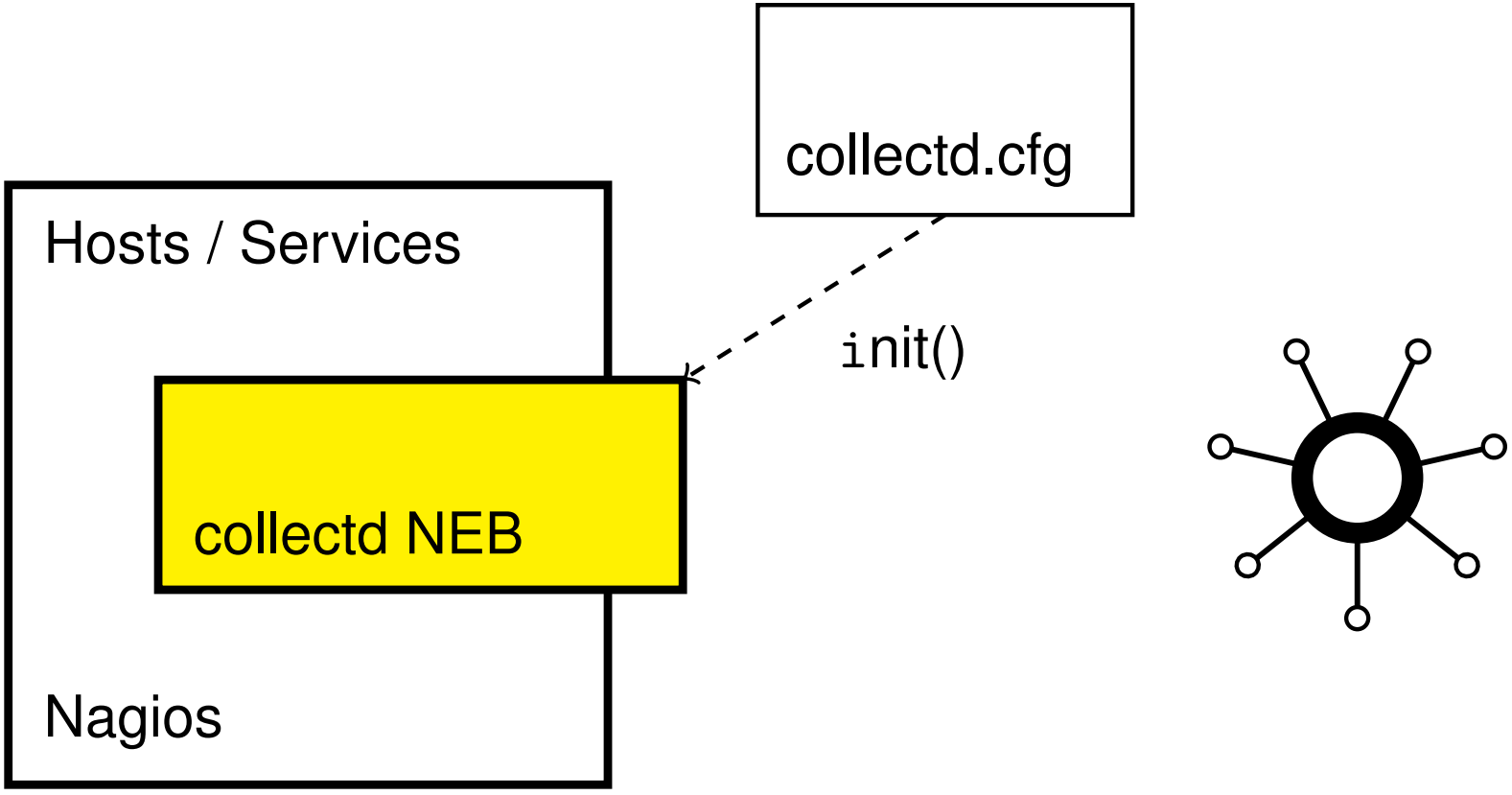




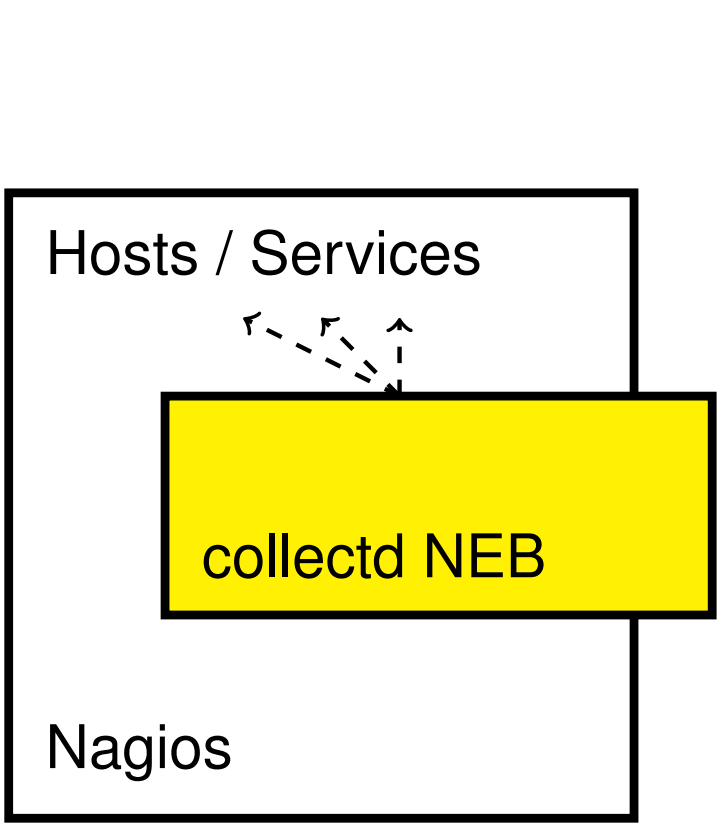
Zukunftsmusik: aktiver Modus

- „aktiver“ Event Broker: „embedded collectd-nagios“
- Erkennen von collectd Checks (konf. command name, custom variable, o.ä.)
- Abfrage von collectd über UNIX Socket
- Auswerten der Daten
- active check result an Nagios übergeben

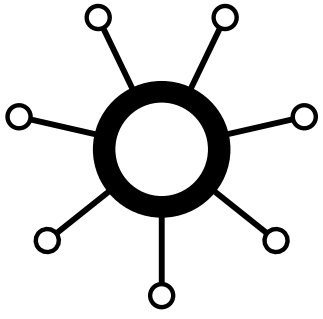
Zukunftsmusik: passiver Modus



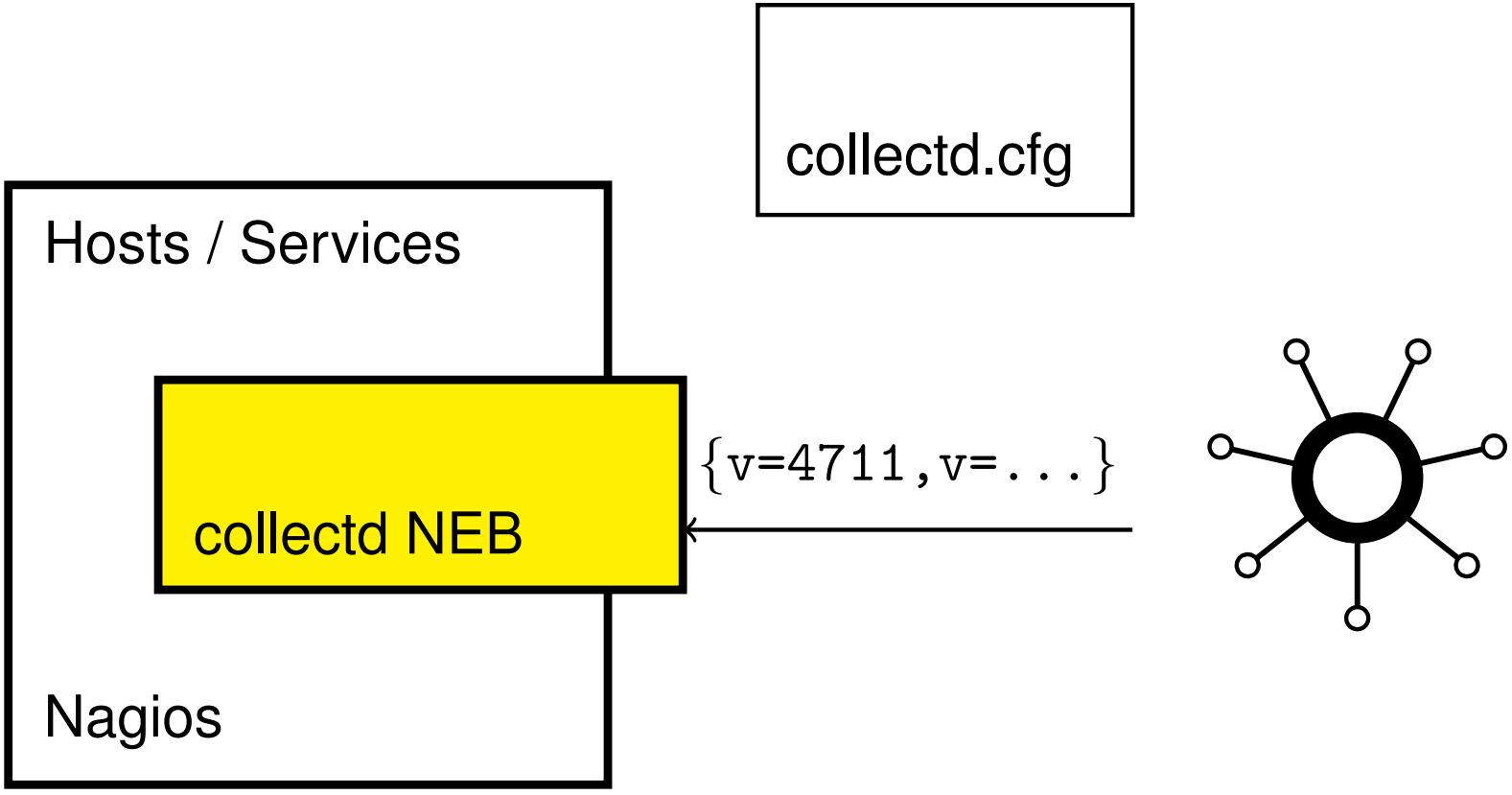
Zukunftsmusik: passiver Modus



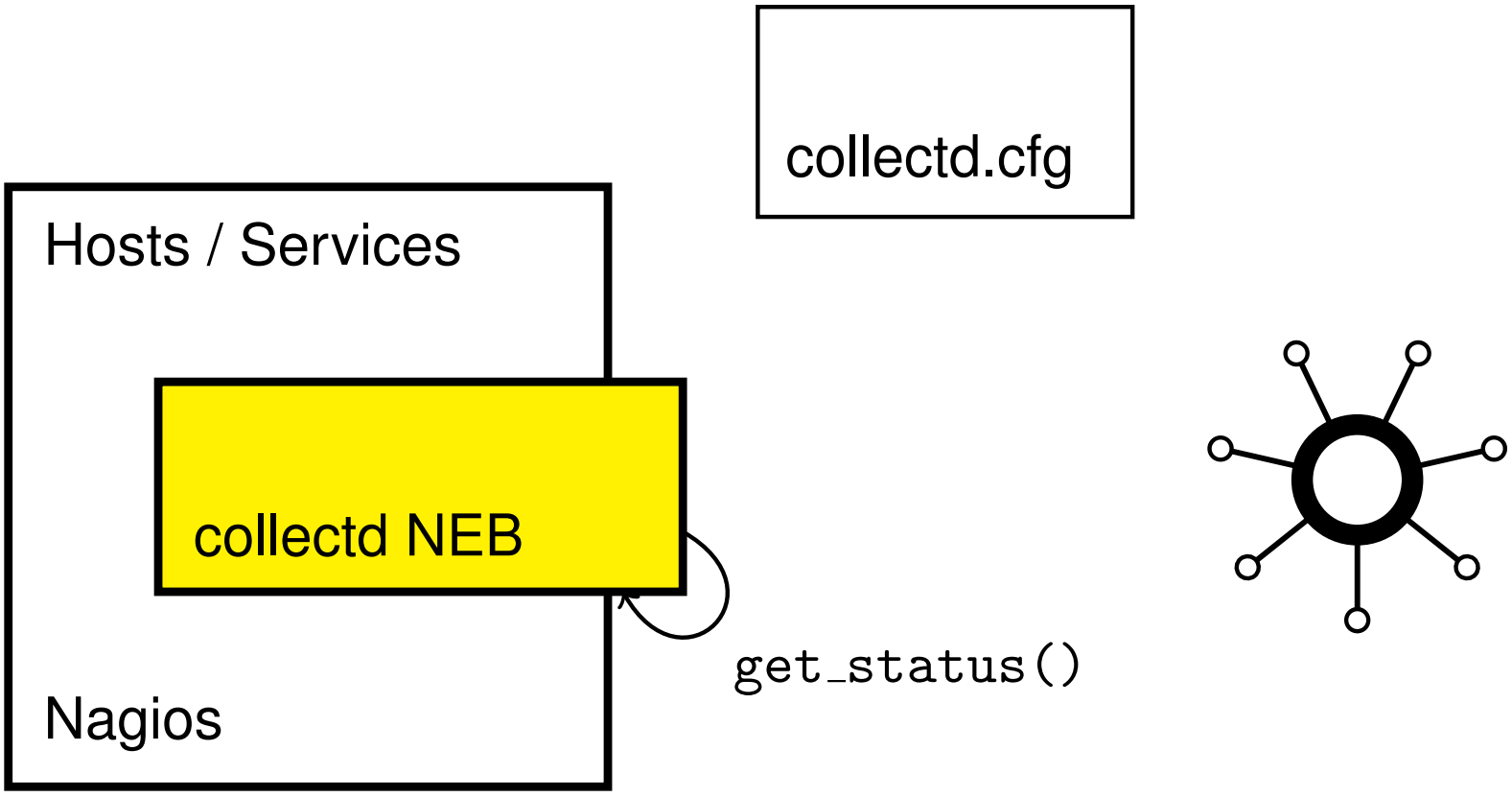
collectd.cfg



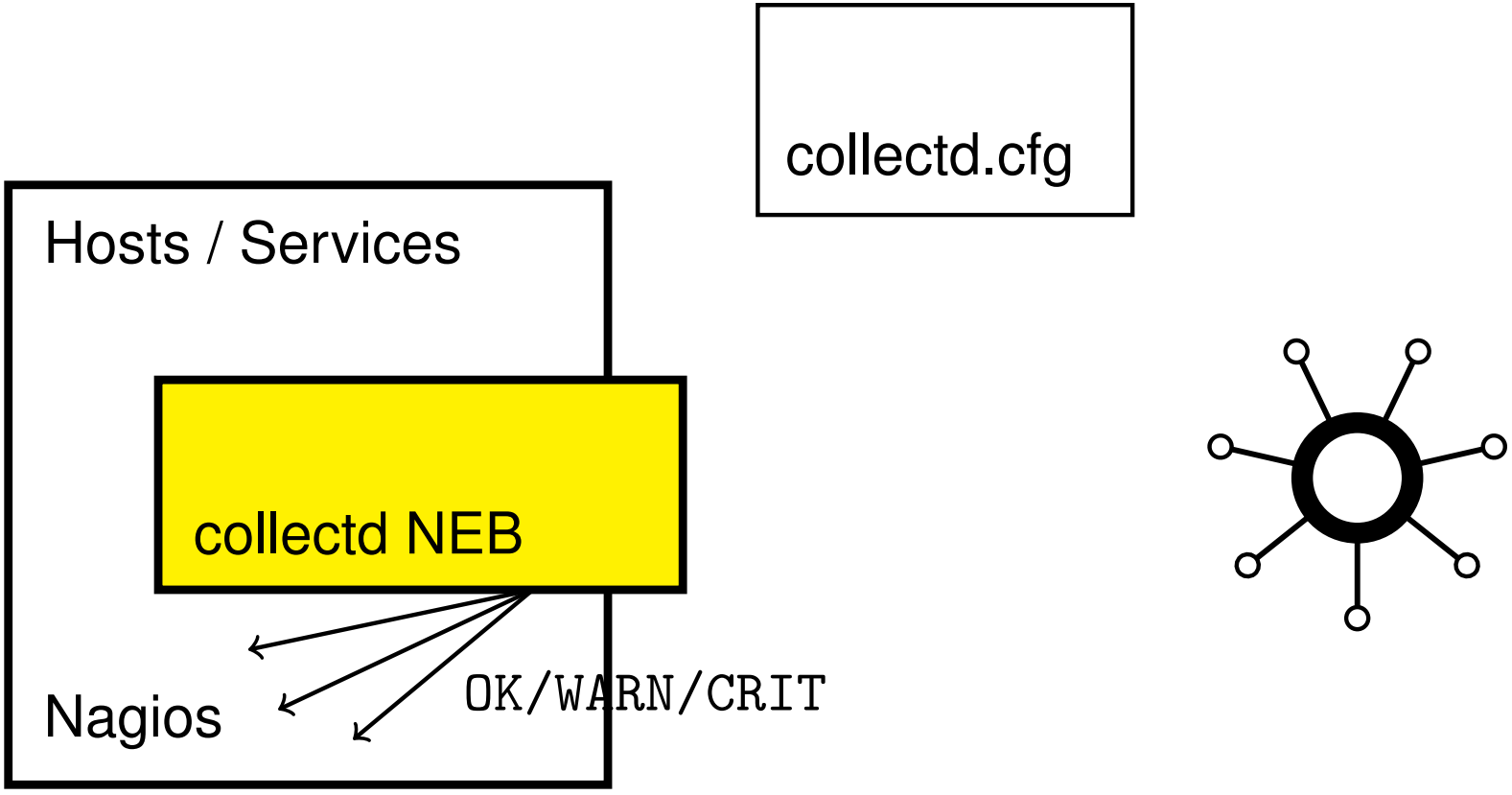
Zukunftsmusik: passiver Modus



Zukunftsmusik: passiver Modus



Zukunftsmusik: passiver Modus

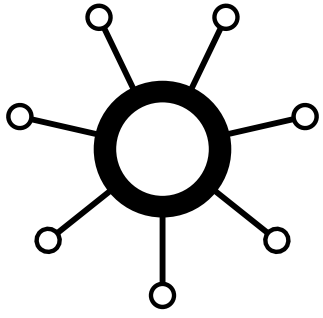


Zukunftsmusik: passiver Modus

- „passiver“ Event Broker: collectd Netzwerk Client
- Konfiguration von gewünschten Werten und Schwellwerten in eigener Konfigurationsdatei (`collectd.cfg`)
- Erzeugung von dazugehörigen Hosts und Services
- Server für collectds Netzwerk-Plugin
- Auswerten der gewünschten Werte
- active/passive check results an Nagios übergeben

Vielen Dank für die Aufmerksamkeit!

Gibt es Fragen?



Kontakt:
Sebastian „tokkee“ Harl
team(ix) GmbH, Nürnberg
<sh@teamix.net>

<collectd@verplant.org> — irc.freenode.net/#collectd — http://identi.ca/collectd
http://tokkee.org/events.html

Artikel zu **collect**d:

`http://linuxtechnicalreview.de/Vorschau/(show)`
`/Themen/Monitoring/Performance-Analyse-mit-Collectd`

Allgemeines

- Fragt Netzwerk-Zubehör via SNMP ab
- *Generisch:* Nicht für ein gestimmtes Gerät geschrieben
- Mehrere Geräte werden parallel abgefragt

Konfiguration

- „Data“-Blöcke
- „Host“-Blöcke

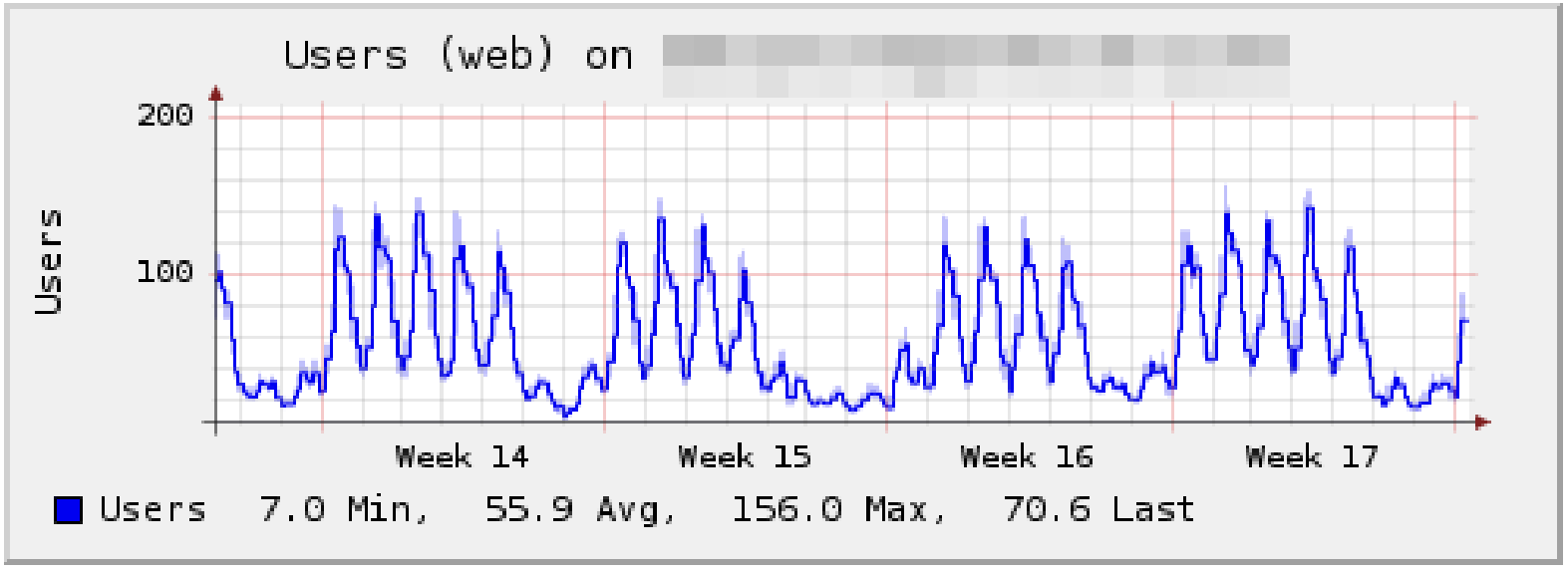
Synopsis: Data-Block

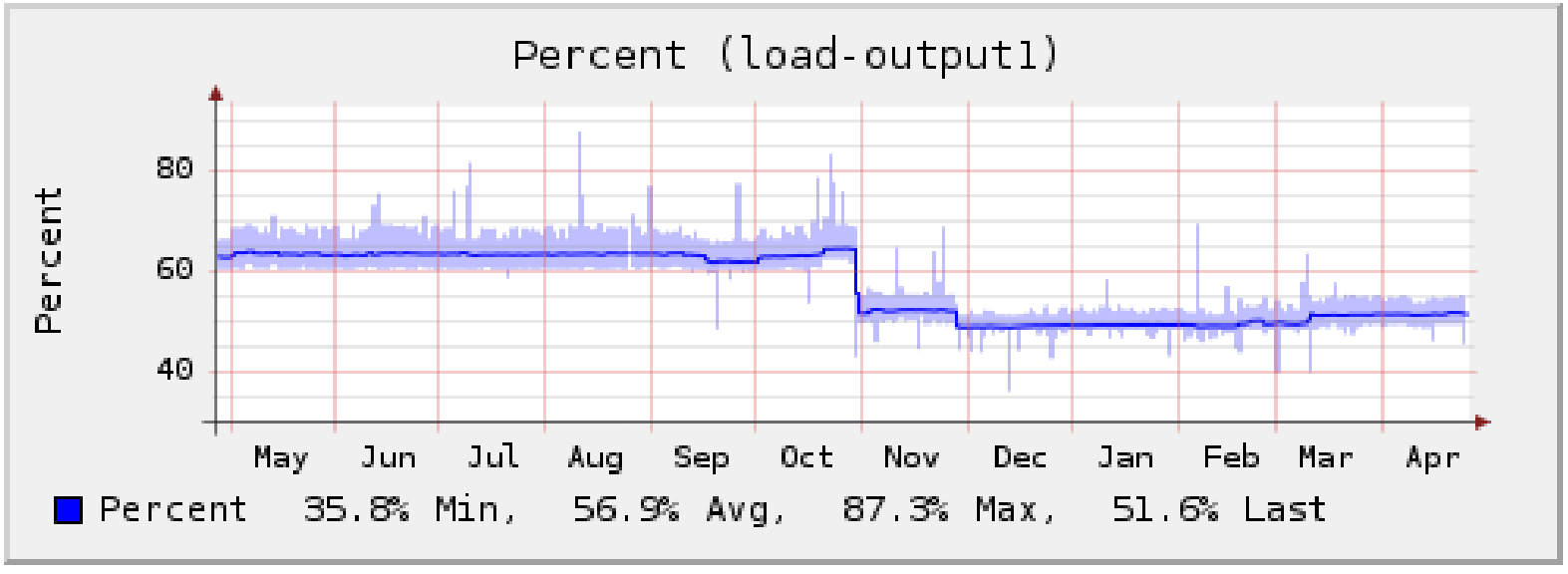
```
<Plugin "snmp">
  <Data "ifmib_if_octets64">
    Type "if_octets"
    Table true
    Instance "IF-MIB::ifName"
    Values "IF-MIB::ifHCInOctets" \
           "IF-MIB::ifHCOutOctets"
  </Data>
</Plugin>
```

Synopsis: Host-Block

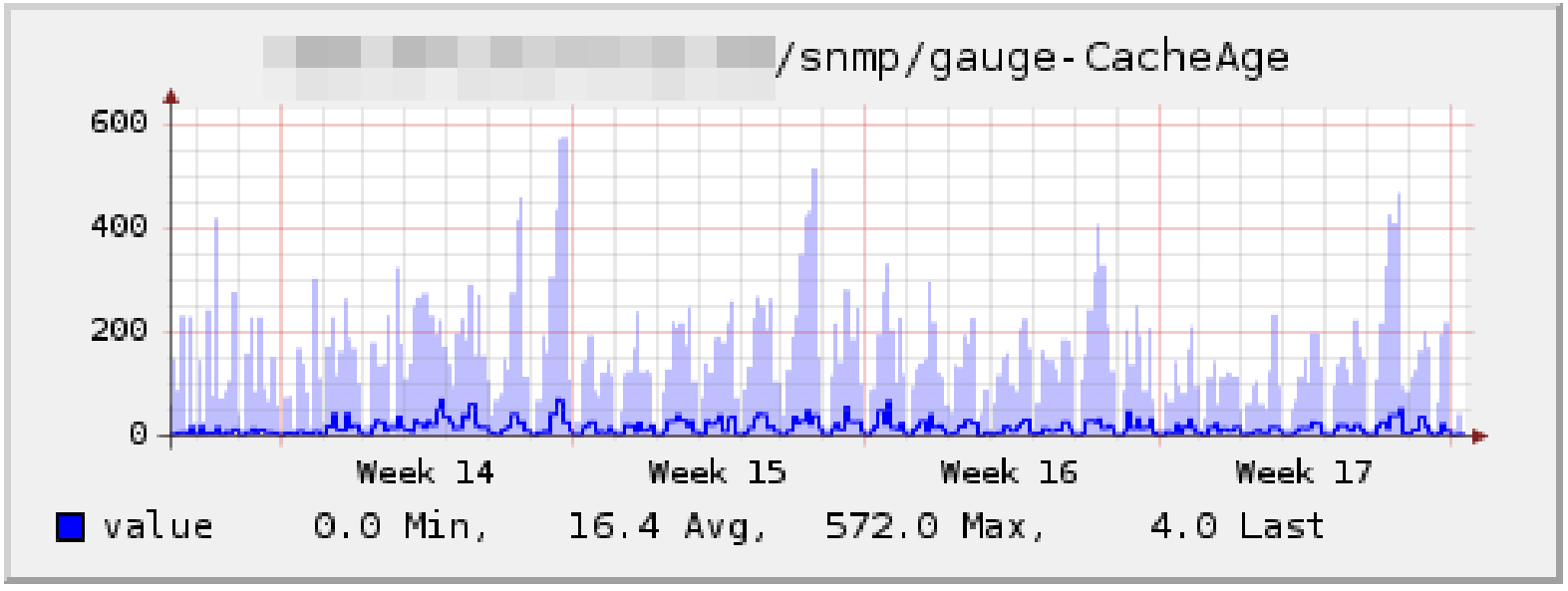
```
<Plugin "snmp">
  <Host "switch0.intern.musterfirma.de">
    Address "10.0.42.2"
    Version 1
    Community "public"
    Collect "ifmib_if_octets64"
    Interval 60
  </Host>
</Plugin>
```

SNMP-Plugin: Users




Percent (load-output1)
Percent
80
60
40
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
Percent 35.8% Min, 56.9% Avg, 87.3% Max, 51.6% Last

SNMP-Plugin: Cache-Alter



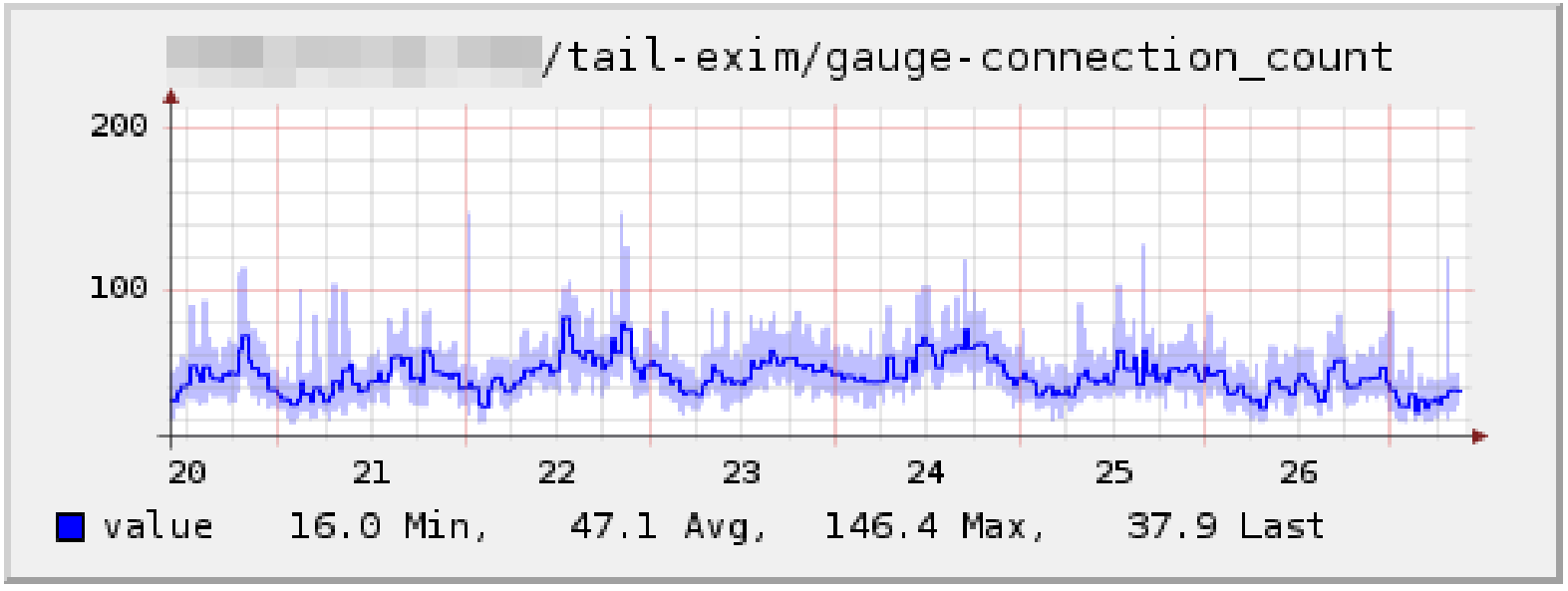
Allgemeines

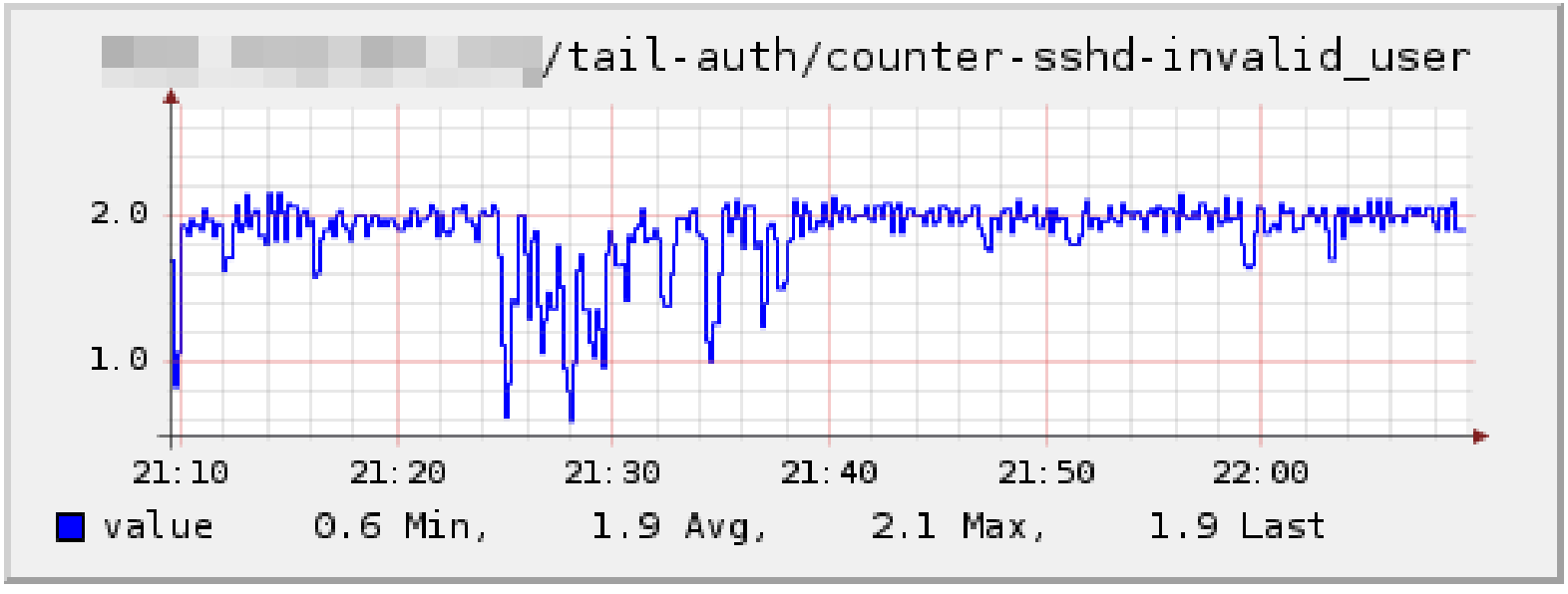
- Verfolgt Log-Dateien
- Extrahiert Werte oder zählt Ereignisse
- Selektion der Zeilen / Werte mit regulären Ausdrücken
- Verwendbar für MTAs, Web-Server, . . .

Konfiguration

```
<Plugin "tail">
  <File "/var/log/exim4/mainlog">
    Instance "exim"
    <Match>
      Regex "S=([1-9][0-9]*)"
      DSType "CounterAdd"
      Type "ipt_bytes"
      Instance "total"
    </Match>
  </File>
</Plugin>
```

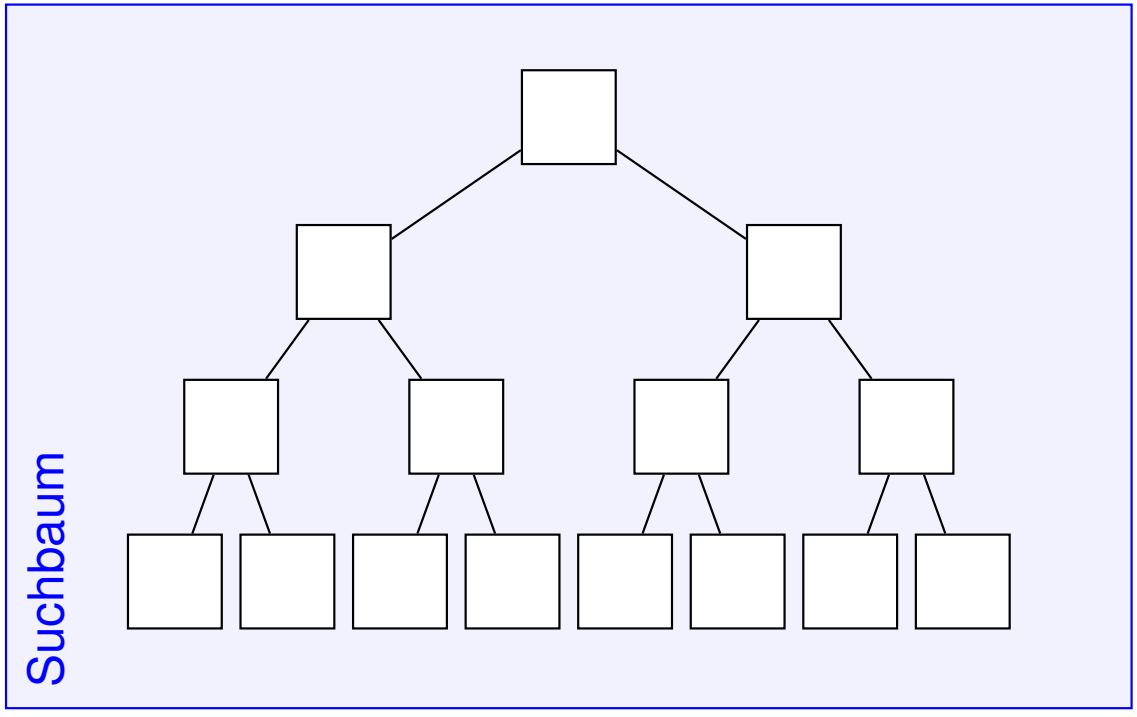
tail-Plugin: Verbindungen von Exim



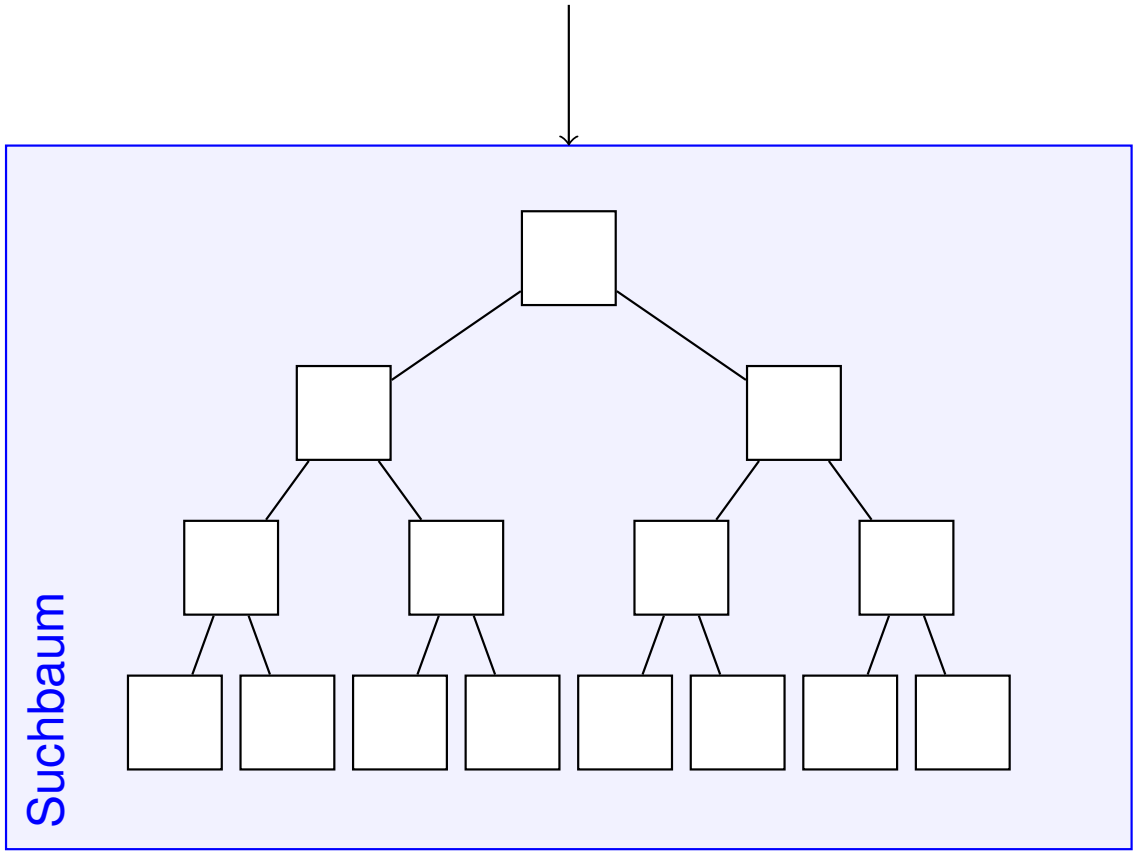

/tail-auth/counter-sshd-invalid_user
2.0
1.0
21:10
21:20
21:30
21:40
21:50
22:00
value 0.6 Min, 1.9 Avg, 2.1 Max, 1.9 Last

Allgemeines

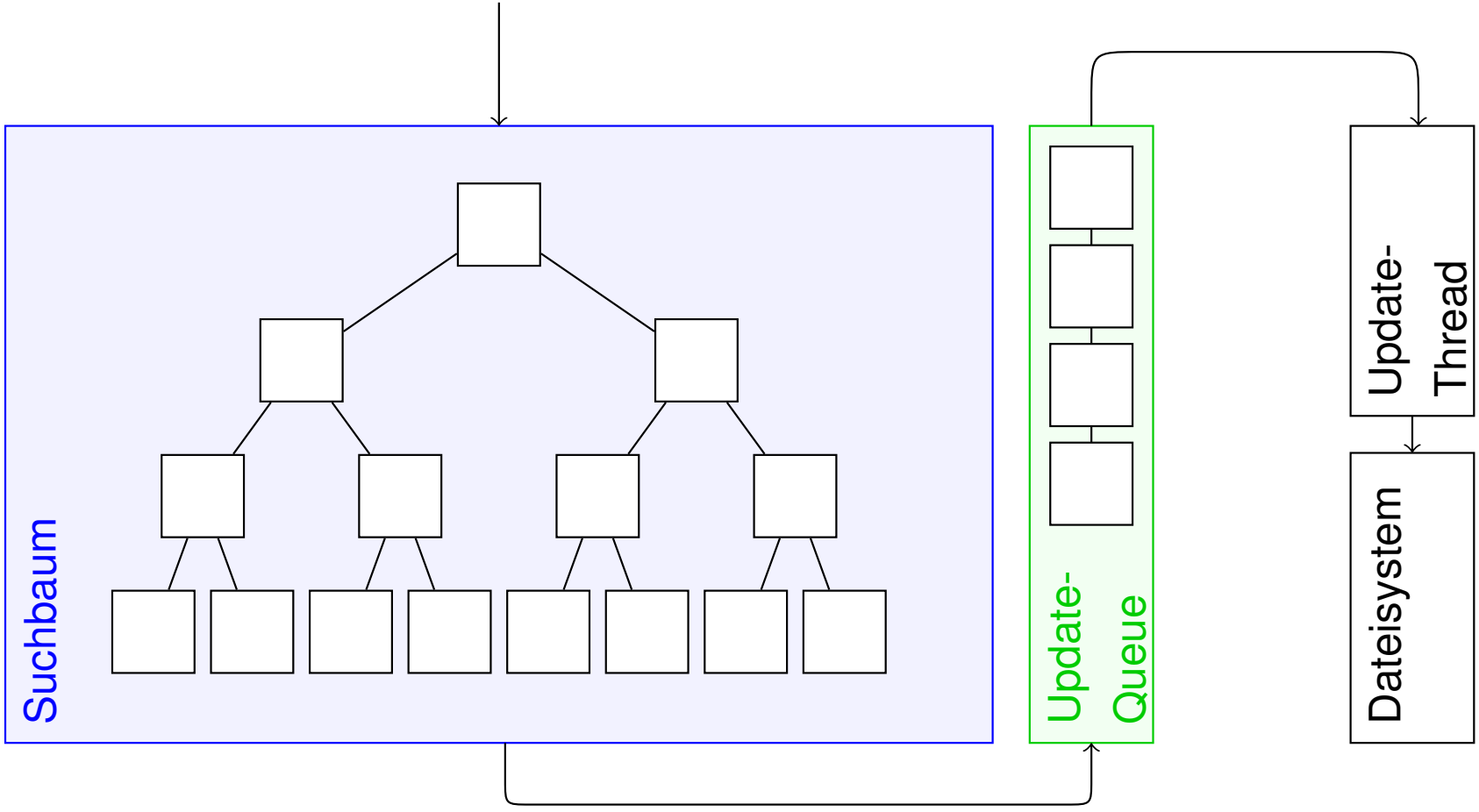
- Update-Prinzip des RRDtool-Plugins
- Eigenständiger Daemon
- Integration in RRDtool 1.4
- Weitere Funktionen, z. B. Journaling
- Vorteil: Neustart von **collect**d ohne Cache-Verlust



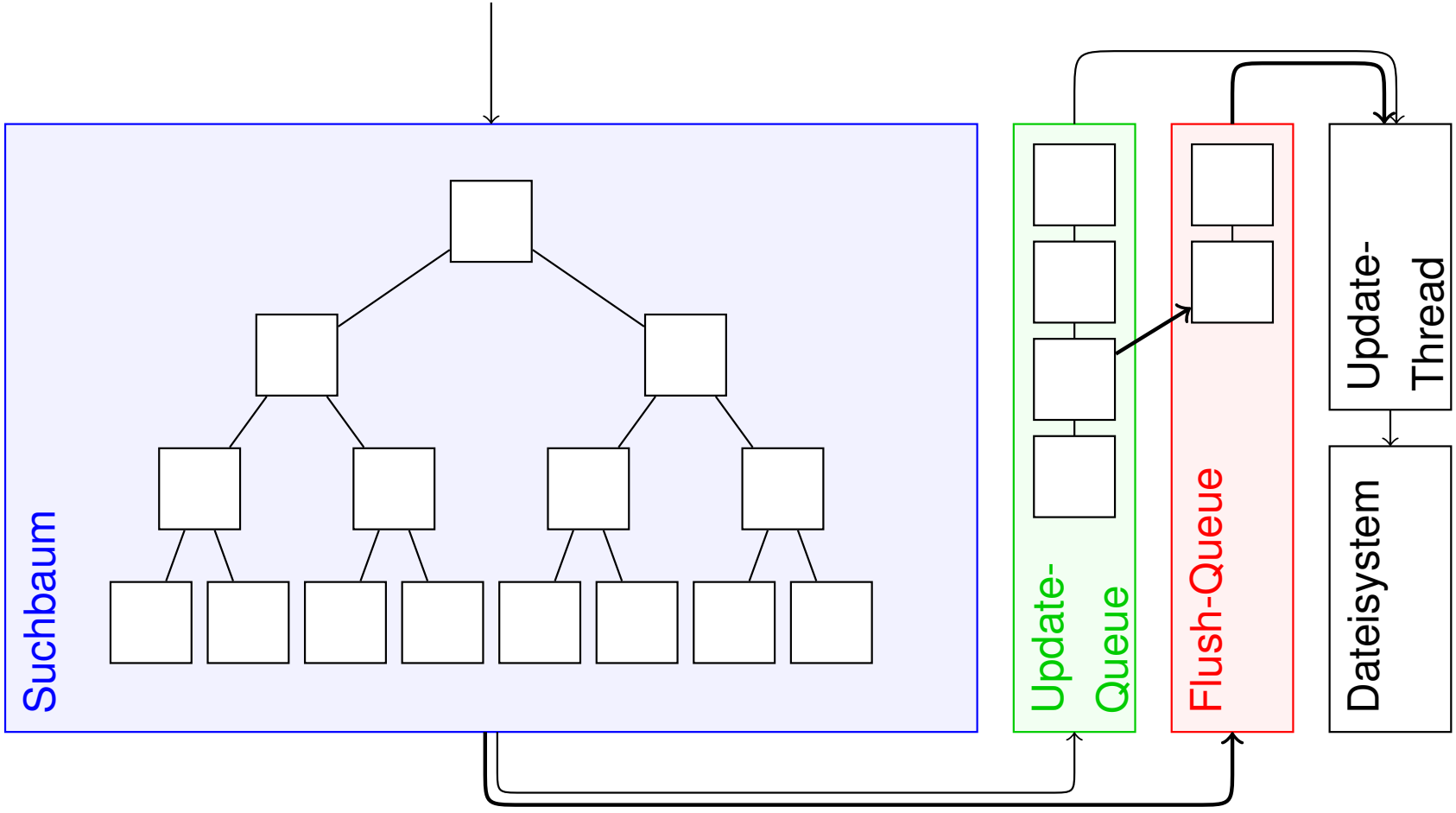
Graphik © Florian „octo“ Forster



Graphik © Florian „octo“ Forster



Graphik © Florian „octo“ Forster



Graphik © Florian „octo“ Forster

Allgemeines

- Integriert einen Perl-Interpreter
  (vergleichbar zu Apaches `mod_perl`)
- Instanziierung und Syntax-Analyse nur einmal
- Exportiert die API
  ($\rightarrow$ nicht nur Lese-Plugins möglich)

Perl-Plugin: Beispiel


```
package Collectd::Plugin::Magic;
use Collectd qw( :all );
sub magic_read
{
  my $vl = { plugin => 'magic',
             values => [Magic->getCurrentLevel ()] };
  plugin_dispatch_values ('magic_level', $vl);
}
plugin_register (TYPE_READ, 'magic', 'magic_read');
```

Allgemeines

- Führt Programme aus
- Liest von deren Standard-Ausgabe
- Können über längere Perioden laufen
  (vgl. `init`)

exec-Plugin: Beispiel


```
#!/bin/sh
INTVL=${COLLECTD_INTERVAL:-10}
CHOST="${COLLECTD_HOSTNAME:-localhost}"
IDENT="$CHOST/magic/magic_level"
while sleep $INTVL
do
   VALUE=‘magic --level‘
   echo "PUTVAL \"$IDENT\" interval=$INTVL N:$VALUE"
done
```

Allgemeines

- Integriert eine „Java Virtual Maschine“ (JVM)
- Exportiert die API
  ($\rightarrow$ nicht nur Lese-Plugins möglich)
- Prinzipielle Ähnlichkeit zum Perl-Plugin

Java-Plugin: Beispiel


```
import org.collectd.api.Collectd;
import org.collectd.api.CollectdReadInterface;
public class MagicPlugin
    implements CollectdReadInterface
{
  public int read ();     /* Callback-Funktion */
  public MagicPlugin (); /* Konstruktor */
}
```

Java-Plugin: Beispiel


```
public int read ()
{
  ValueList vl = new ValueList ();
  vl.setHost ("testhost");
  vl.setPlugin ("magic");
  vl.setType ("magic_level");
  vl.addValue (Magic.getCurrentLevel ());
  return (Collectd.dispatchValues (vl));
}
```

Java-Plugin: Beispiel


```
public MagicPlugin ()
{
  /* Callback-Funktion anmelden */
  Collectd.registerRead ("MagicPlugin", this);
}
```

Erweiterungen: Zusammenfassung

- **collect**d API nutzen
  C, Perl, Python und Java möglich

Erweiterungen: Zusammenfassung

- **collect**d API nutzen
  C, Perl, Python und Java möglich
- Externe Programme erweitern
  unixsock-Plugin ermöglicht Kommunikation

Erweiterungen: Zusammenfassung

- **collect**d API nutzen
  C, Perl, Python und Java möglich
- Externe Programme erweitern
  unixsock-Plugin ermöglicht Kommunikation
- Eigenes Programm / Skript schreiben
  $\rightarrow$ exec-Plugin

Über den Tellerrand: Zubehör

- `snmp-probe-host.px`
  Erzeugt semi-automatisch `<Host />`-Blöcke für das SNMP-Plugin
- jcollectd
  Java-Implementierung des Netzwerk-Protokolls *(→ JMX)*
- kcollectd
  KDE-Programm zur Near-Realtime-Anzeige von Graphen
- Perl, Ruby, Python Module und C-Bibliothek für die Kommunikation mit dem unixsock-Plugin verfügbar