Building a Culture of Observability Within your Organization

CNCF Webinar Series - May 7

Grant Schofield - Humio

Greetings.

Who am I?

HUMIO

https://humio.com

Overview

What is observability?

Observability's Past, Present, and Future

How can my organization make their systems observable?

What Observability isn’t….

New concept

Monitoring

A tool or product

Just your operations team’s concern

Observability is…

The intersection systems, applications, and **users**

An attribute of a system than encompasses many others such as: functionality, performance, testability, maintainability, monitorability

The ability to know how your system is working from the outside and being able to contextualize any events within

Observability is…(to me)

Having the right systems, and implementation, to know how your system is feeling about any request

Having the ability to know why the needle is oriented a particular way in the haystack

Having what I need, as an engineer, to debug a problem with minimal steps

Most importantly..

A cultural facet of your organization, like DevOps

Responsibility of everyone in the organization

Different for every organization

The Past

Control Theory

Control theory in control systems engineering is a subfield of mathematics that deals with the control of continuously operating dynamical systems in engineered processes and machines. The objective is to develop a control model for controlling such systems using a control action in an optimum manner without delay or overshoot and ensuring control stability.

Rudolf E. Kálmán

In control theory, observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs. The observability and controllability of a system are mathematical duals.

Kálmán's ground breaking work in the 50s and 60s that, lead to Kálmán filters, was used by Nasa for the Apollo and shuttle programs as well as a vast array of other applications.

Observing Systems of the Past

SaaS didn’t exist, built everything

Basic Open-source Software - Big Brother, RRDTool, MRTG, Cacti, NetSaint (Nagios)

Expensive (ugly) Enterprise tools - HP Openview, CA, etc

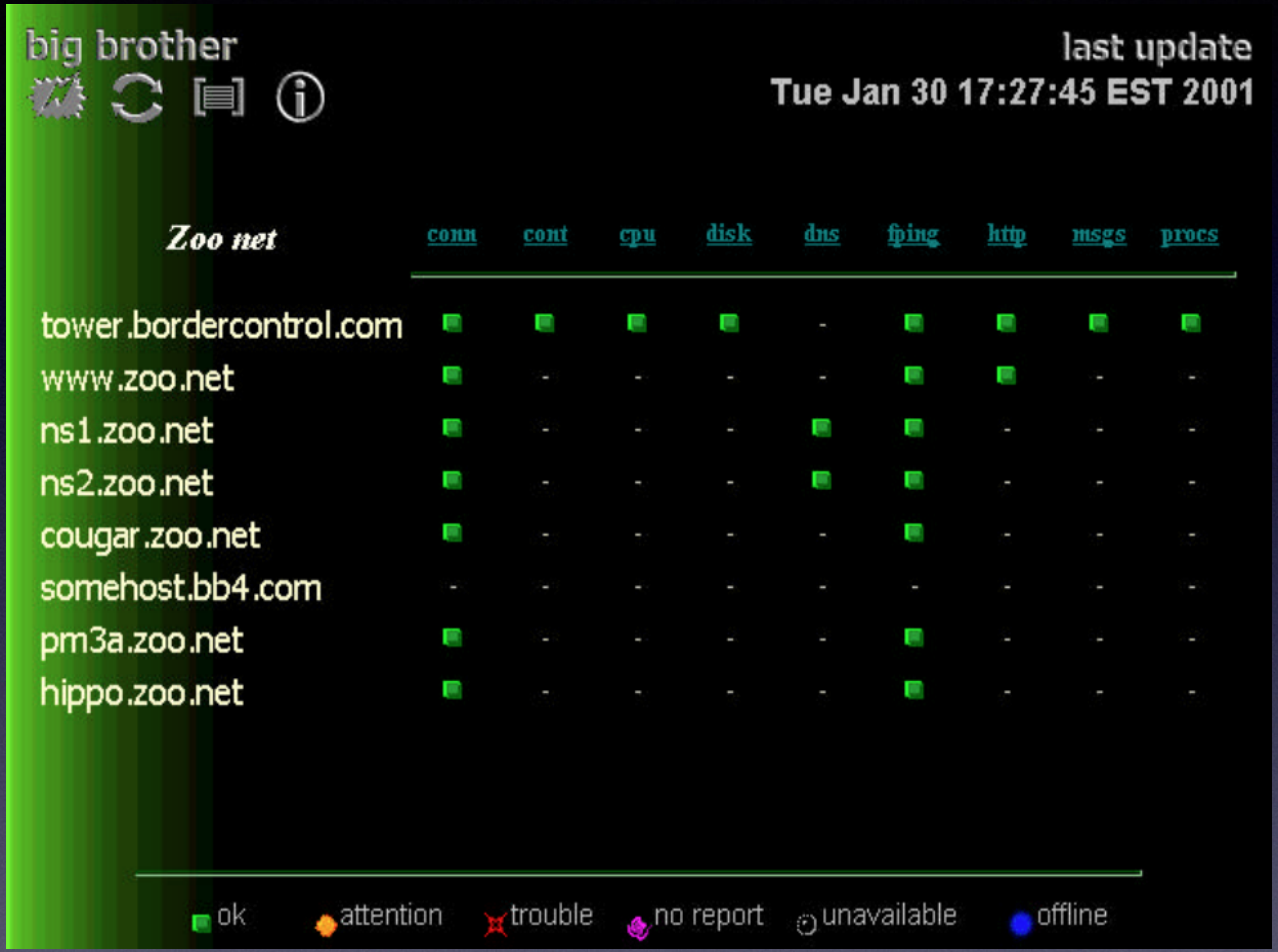
Script everything, or die trying

CREAM - *Cron Rules Everything Around Me*

World ran on rsync

What did we build?

Monitoring



big brother

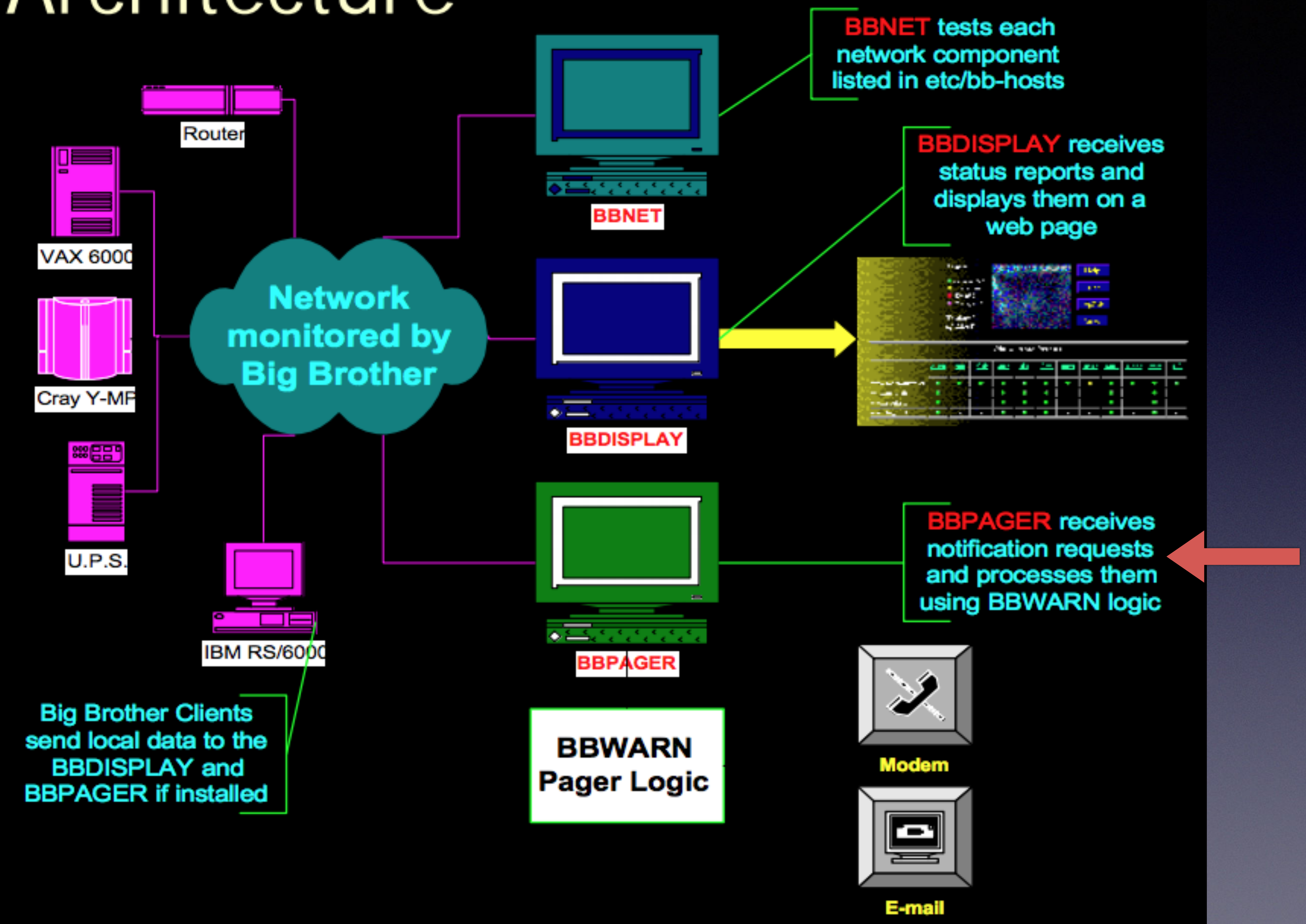
last update
Tue Jan 30 17:27:45 EST 2001

Zoo net	conn	cont	cpu	disk	dns	fping	http	msgs	procs
tower.bordercontrol.com	🟩	🟩	🟩	🟩	-	🟩	🟩	🟩	🟩
www.zoo.net	🟩	-	-	-	-	🟩	🟩	-	-
ns1.zoo.net	🟩	-	-	-	🟩	🟩	-	-	-
ns2.zoo.net	🟩	-	-	-	🟩	🟩	-	-	-
cougar.zoo.net	🟩	-	-	-	-	🟩	-	-	-
somehost.bb4.com	-	-	-	-	-	-	-	-	-
pm3a.zoo.net	🟩	-	-	-	-	🟩	-	-	-
hippo.zoo.net	🟩	-	-	-	-	🟩	-	-	-

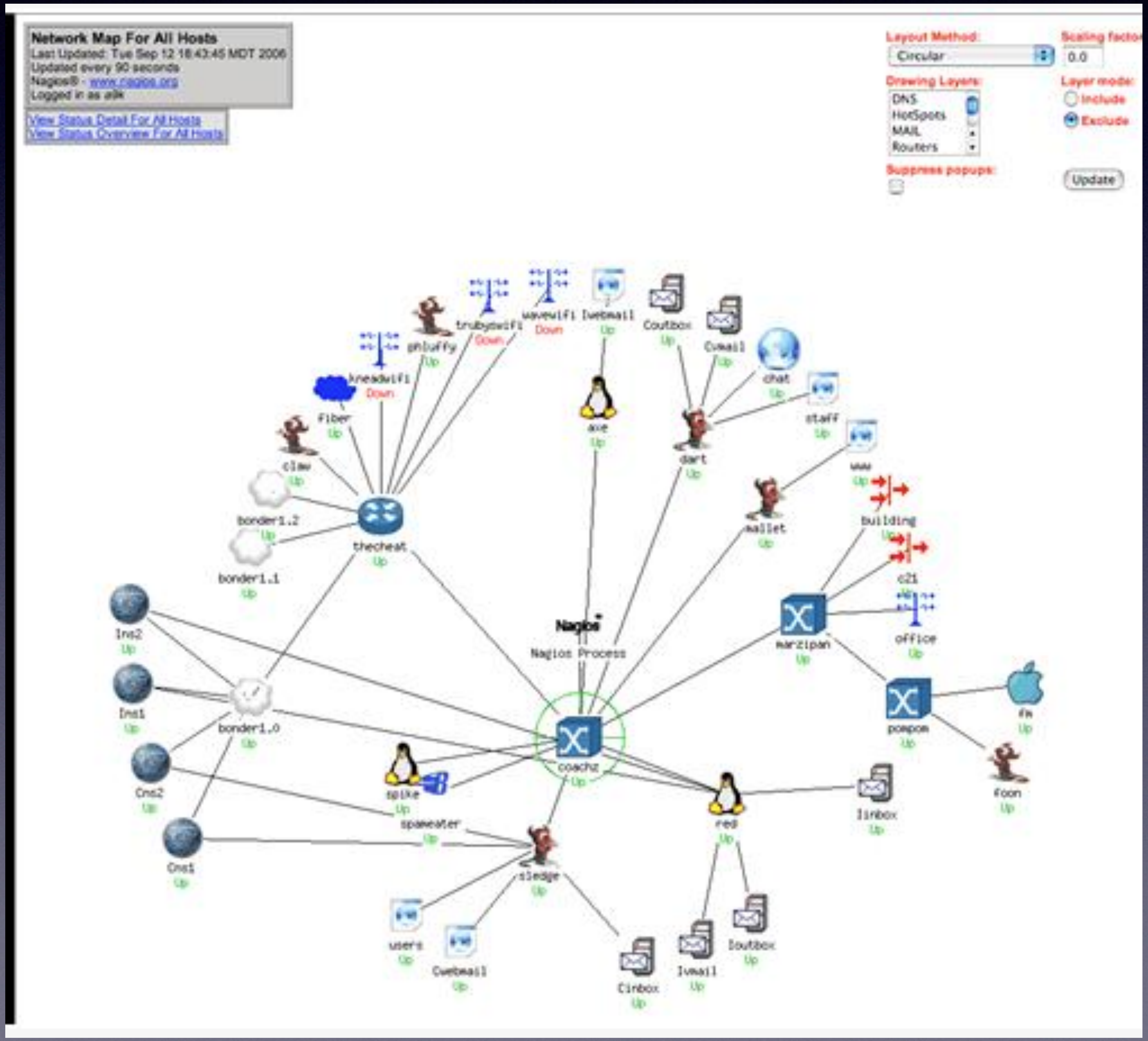
🟩 ok ● attention ✗ trouble ● no report ○ unavailable ● offline

Questionable Bash and Perl execed via Cron to feed Big Brother

Architecture


BBNET tests each network component listed in etc/bb-hosts
BBDISPLAY receives status reports and displays them on a web page
BBPAGER receives notification requests and processes them using BBWARN logic
Router
VAX 6000
Cray Y-MP
U.P.S.
IBM RS/6000
Network monitored by Big Brother
BBNET
BBDISPLAY
BBPAGER
BBWARN Pager Logic
Modem
E-mail
Big Brother Clients send local data to the BBDISPLAY and BBPAGER if installed

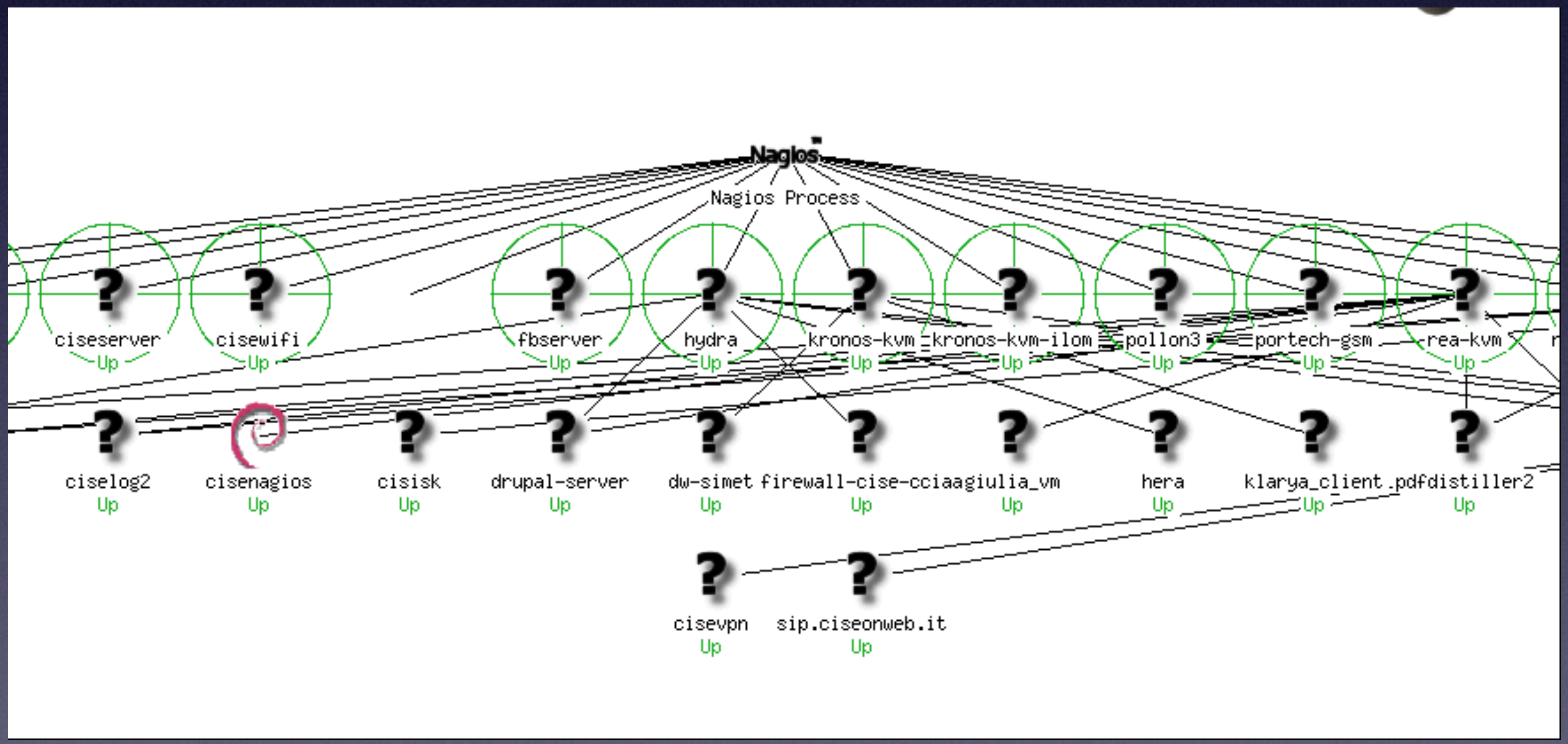
Do sysadmins have VRML colored dreams?



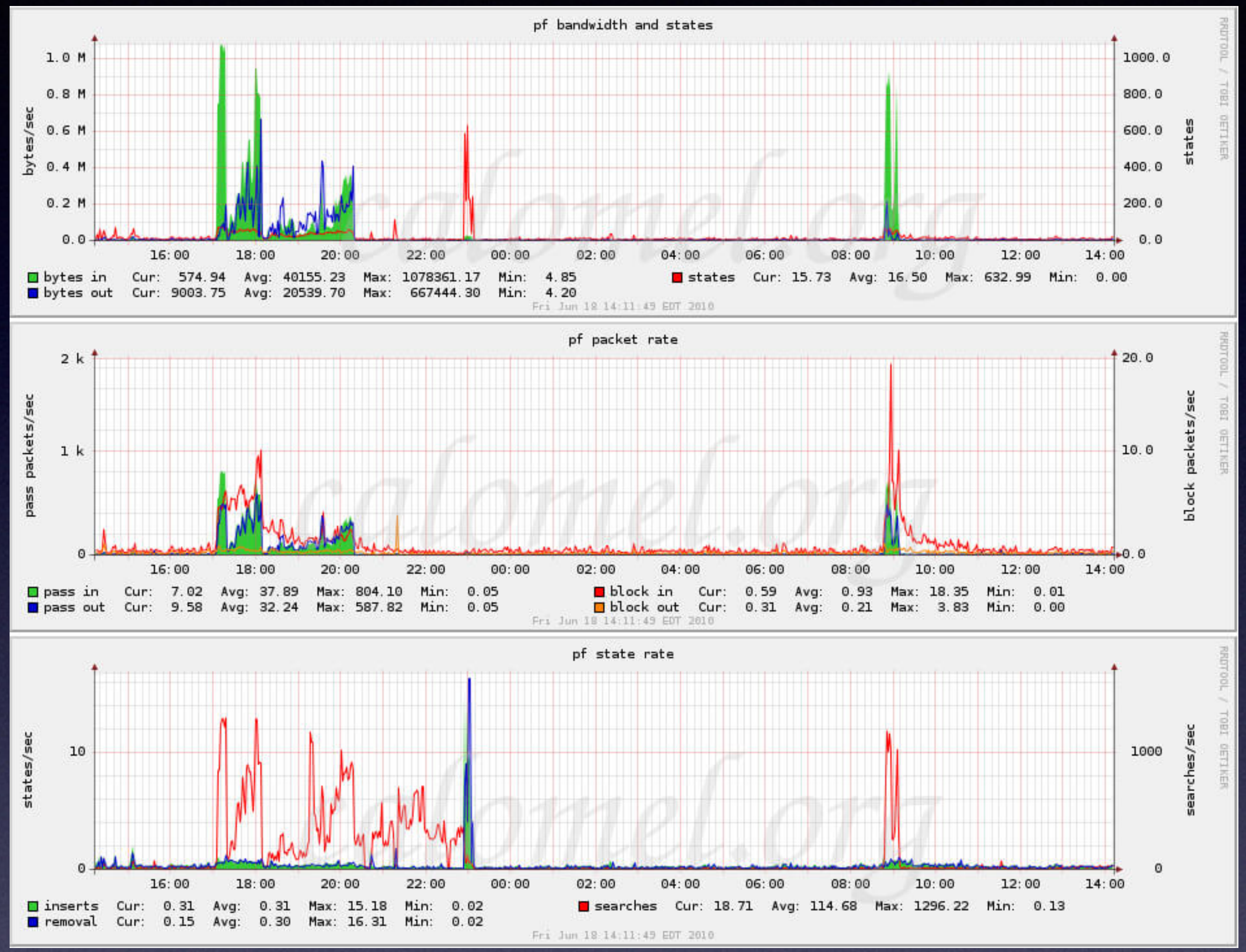
Network Map For All Hosts
Last Updated: Tue Sep 12 18:43:45 MDT 2006
Updated every 90 seconds
Nagios® - www.nagios.org
Logged in as atk

View Status Detail For All Hosts
View Status Overview For All Hosts

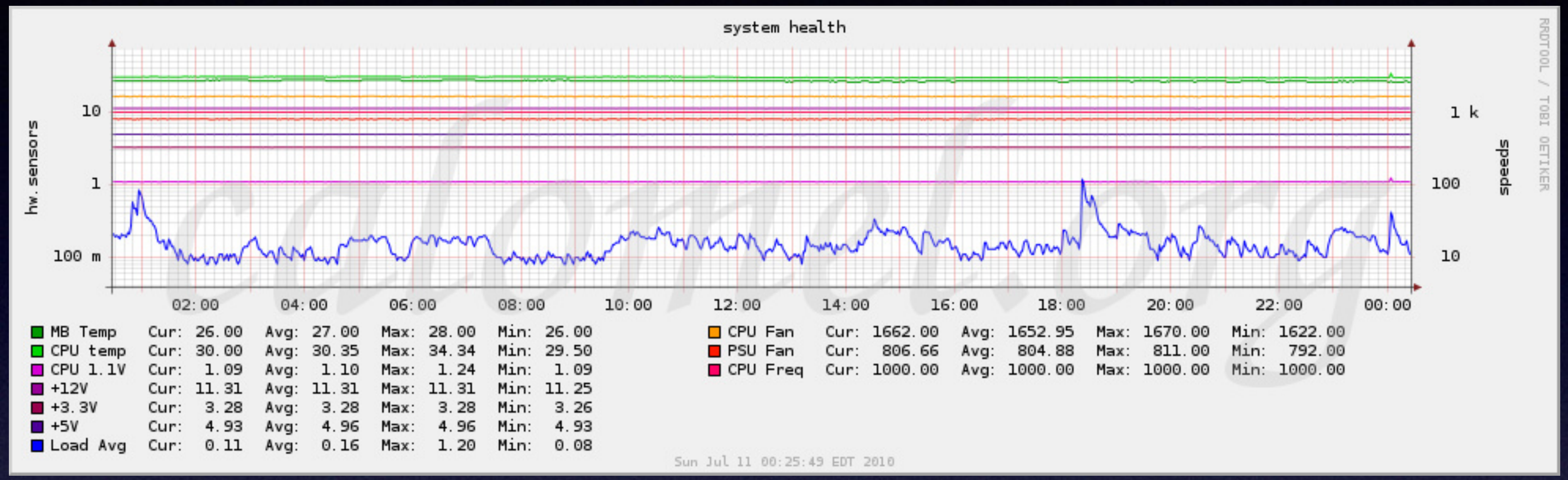
Layout Method: Circular
Scaling factor: 0.0
Drawing Layers: DNS, HotSpots, MAIL, Routers
Layer mode: Include [✓ Exclude]
Suppress popups: ☐



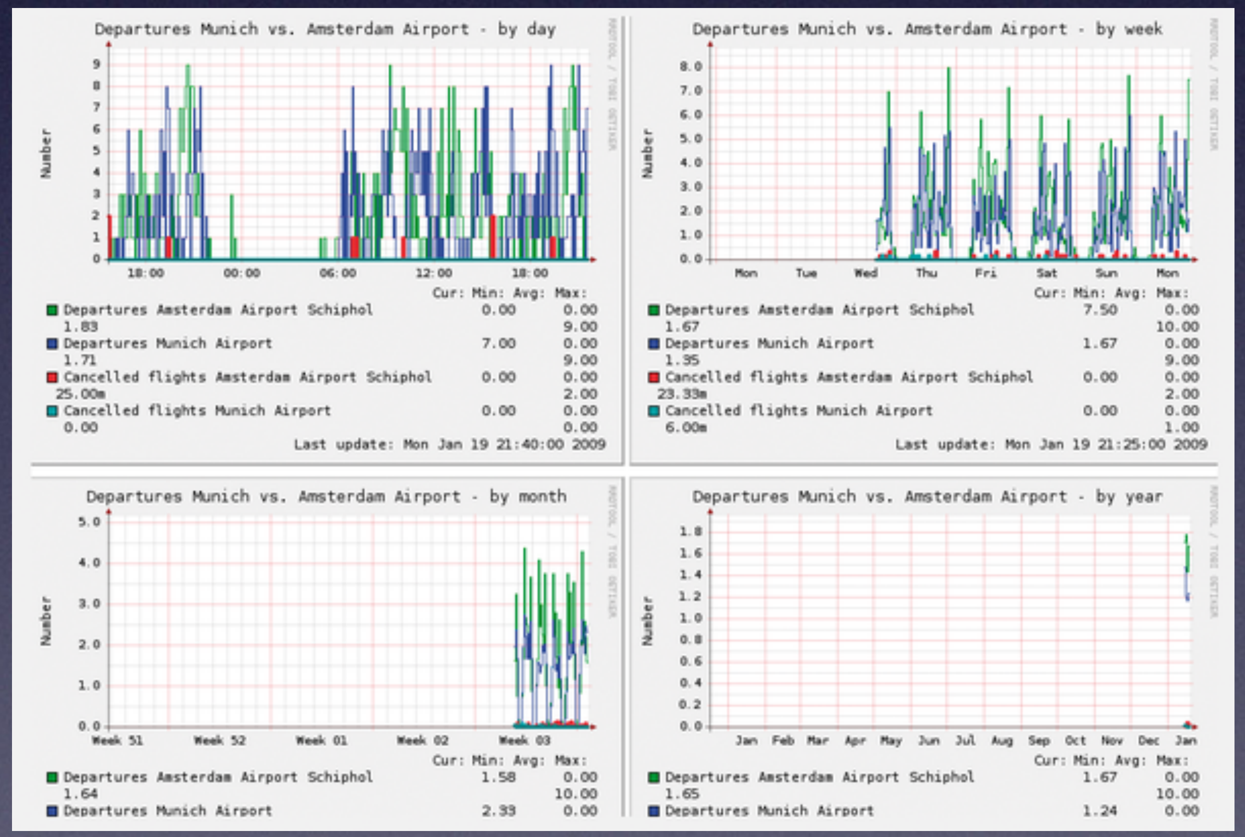
Metrics - Mostly RRDTool Based

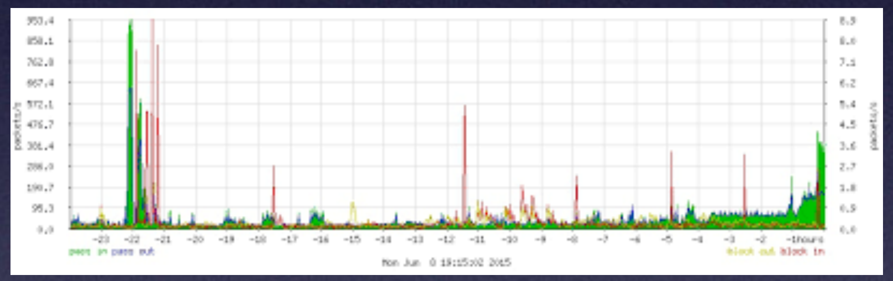


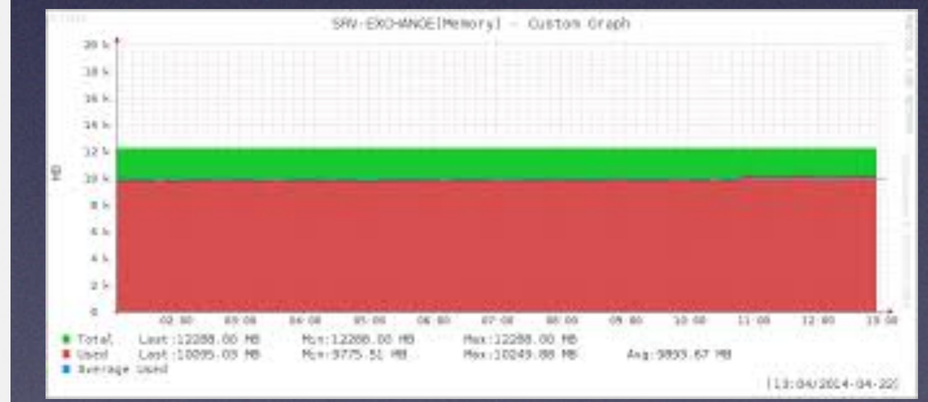
```
bytes in   Cur:   574.94  Avg: 40155.23  Max: 1078361.17  Min:  4.85     states   Cur: 15.73   Avg: 16.50   Max: 632.99   Min:  0.00
bytes out  Cur:  9003.75  Avg: 20539.70  Max:  667444.30  Min:  4.20

pass in    Cur:  7.02   Avg: 37.89   Max: 804.10   Min:  0.05      block in   Cur:  0.59   Avg:  0.93   Max: 18.35   Min:  0.01
pass out   Cur:  9.58   Avg: 32.24   Max: 587.82   Min:  0.05      block out  Cur:  0.31   Avg:  0.21   Max:  3.83   Min:  0.00

inserts    Cur:  0.31   Avg:  0.31   Max: 15.18   Min:  0.02       searches   Cur: 18.71   Avg: 114.68   Max: 1296.22   Min:  0.13
removal    Cur:  0.15   Avg:  0.30   Max: 16.31   Min:  0.02
```

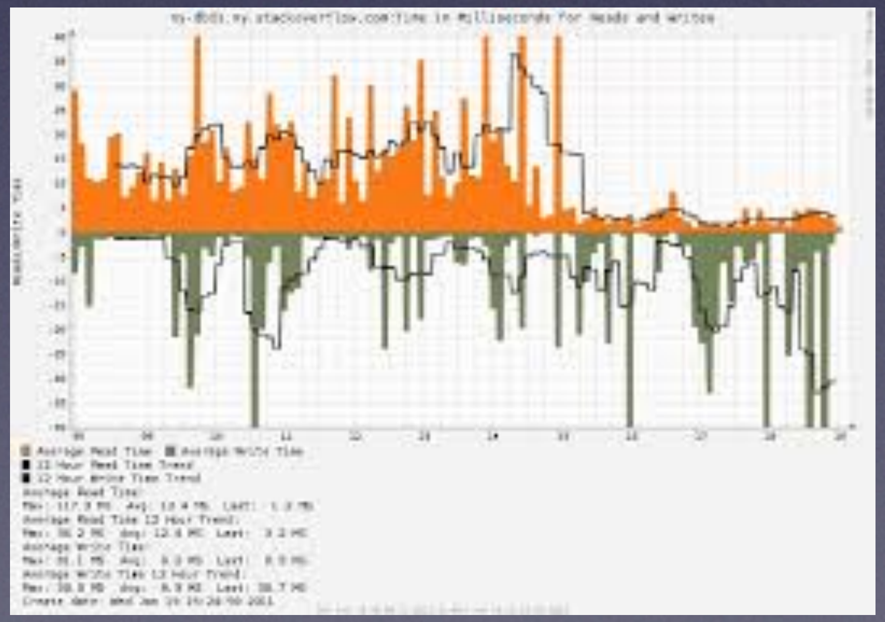


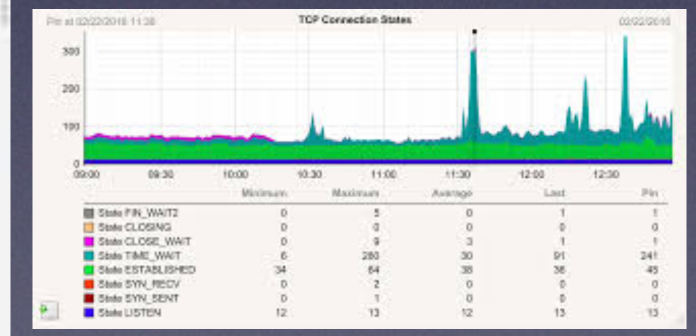
```
MB Temp    Cur:  26.00   Avg:  27.00   Max:  28.00   Min:  26.00      CPU Fan    Cur: 1662.00   Avg: 1652.95   Max: 1670.00   Min: 1622.00
CPU temp   Cur:  30.00   Avg:  30.35   Max:  34.34   Min:  29.50      PSU Fan    Cur:  806.66   Avg:  804.88   Max:  811.00   Min:  792.00
CPU 1.1V   Cur:   1.09   Avg:   1.10   Max:   1.24   Min:   1.09      CPU Freq   Cur: 1000.00   Avg: 1000.00   Max: 1000.00   Min: 1000.00
+12V       Cur:  11.31   Avg:  11.31   Max:  11.31   Min:  11.25
+3.3V      Cur:   3.28   Avg:   3.28   Max:   3.28   Min:   3.26
+5V        Cur:   4.93   Avg:   4.96   Max:   4.96   Min:   4.93
Load Avg   Cur:   0.11   Avg:   0.16   Max:   1.20   Min:   0.08
Sun Jul 11 00:25:49 EDT 2010
```



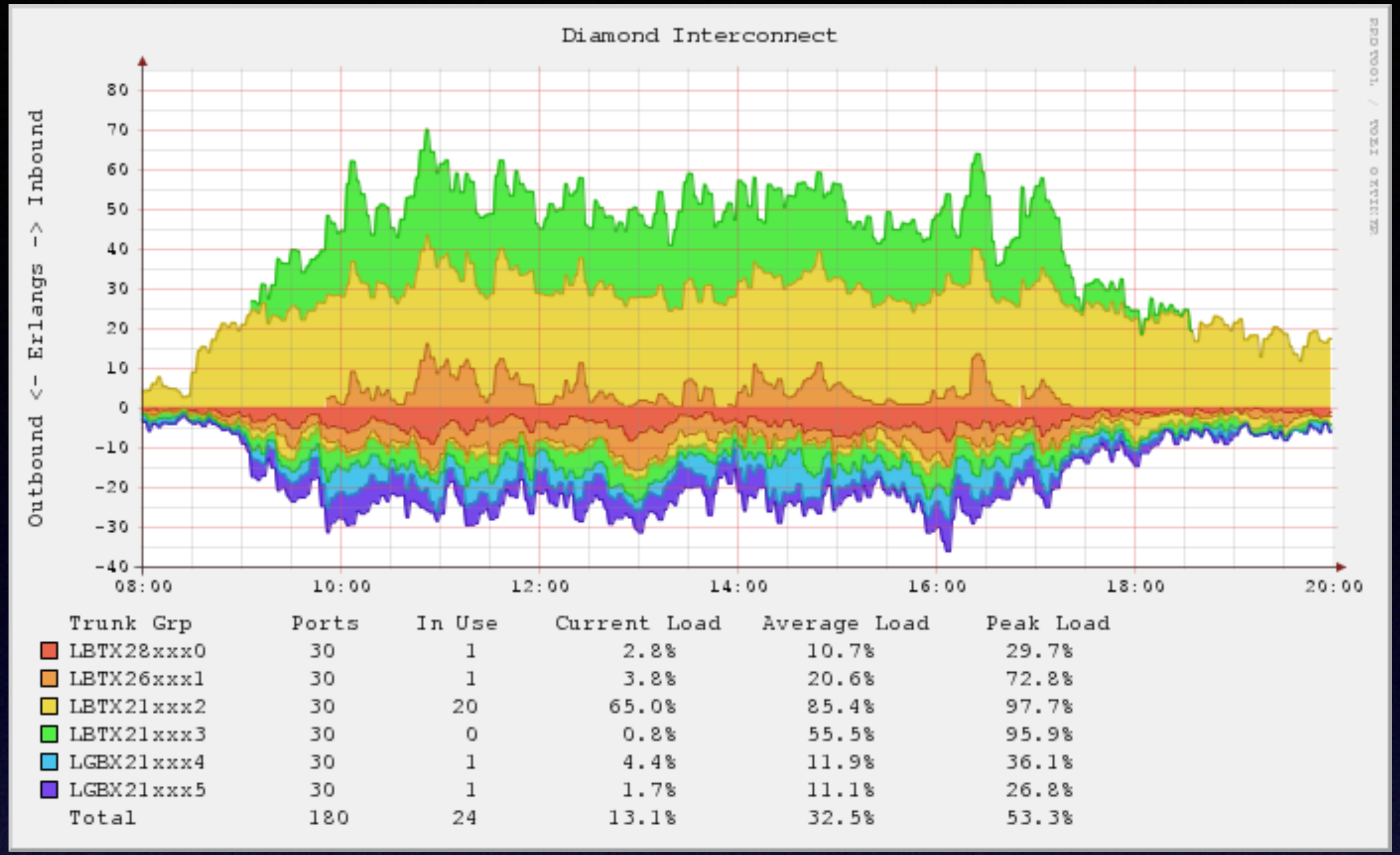




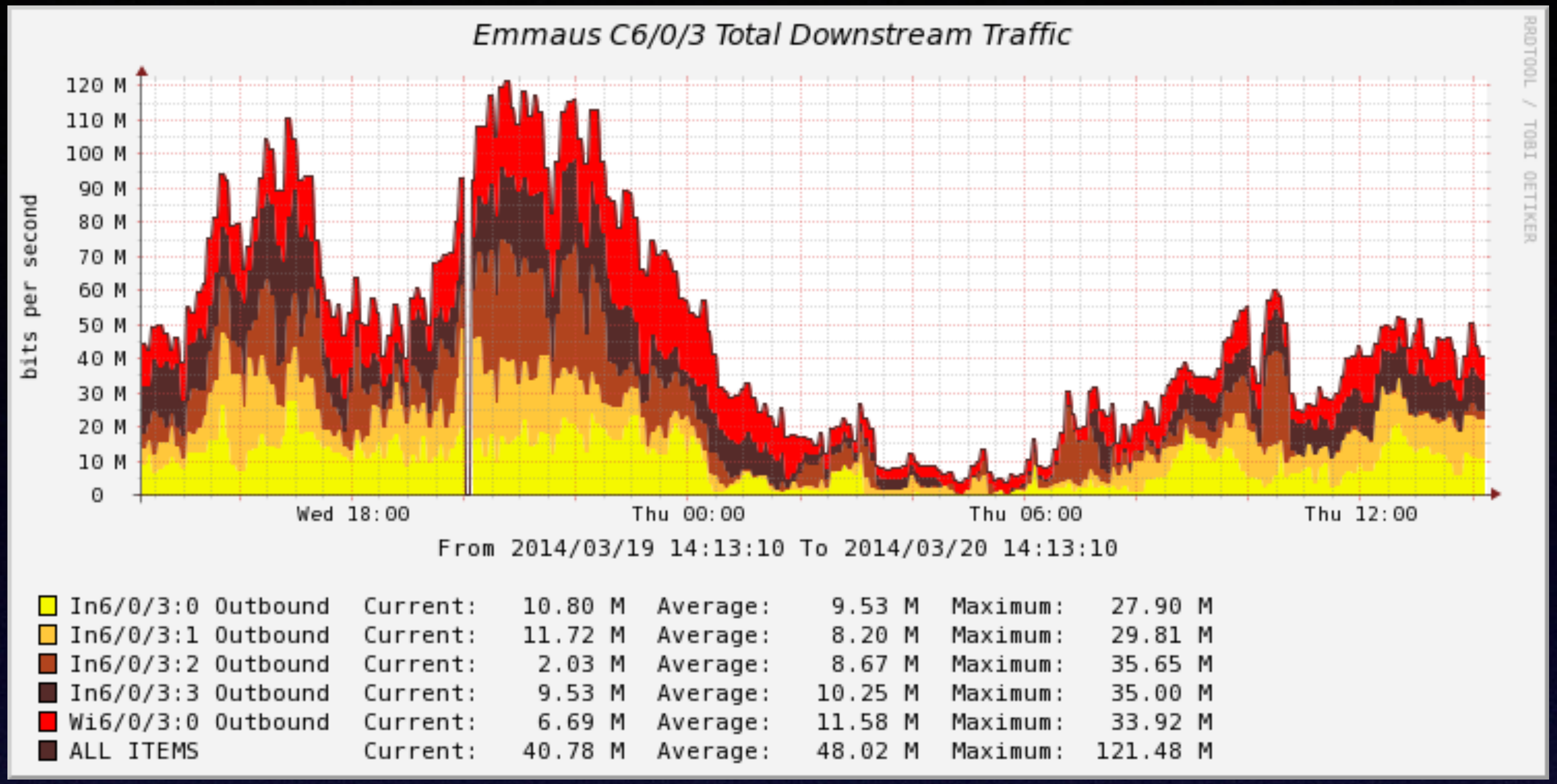




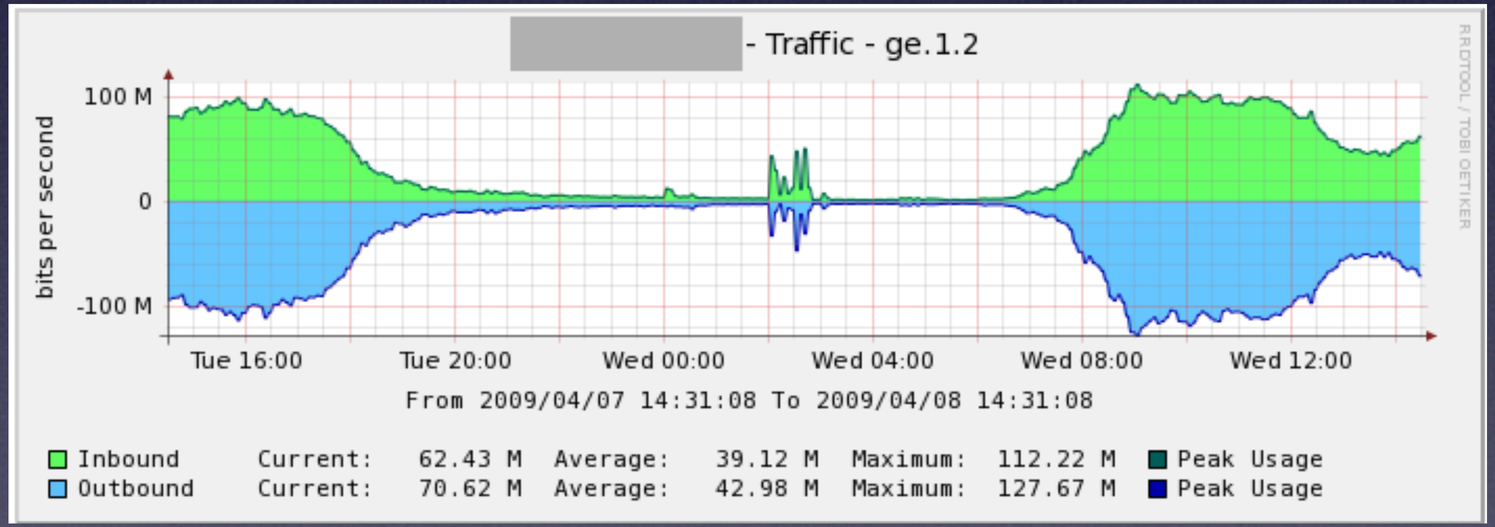
Rub more Perl in it



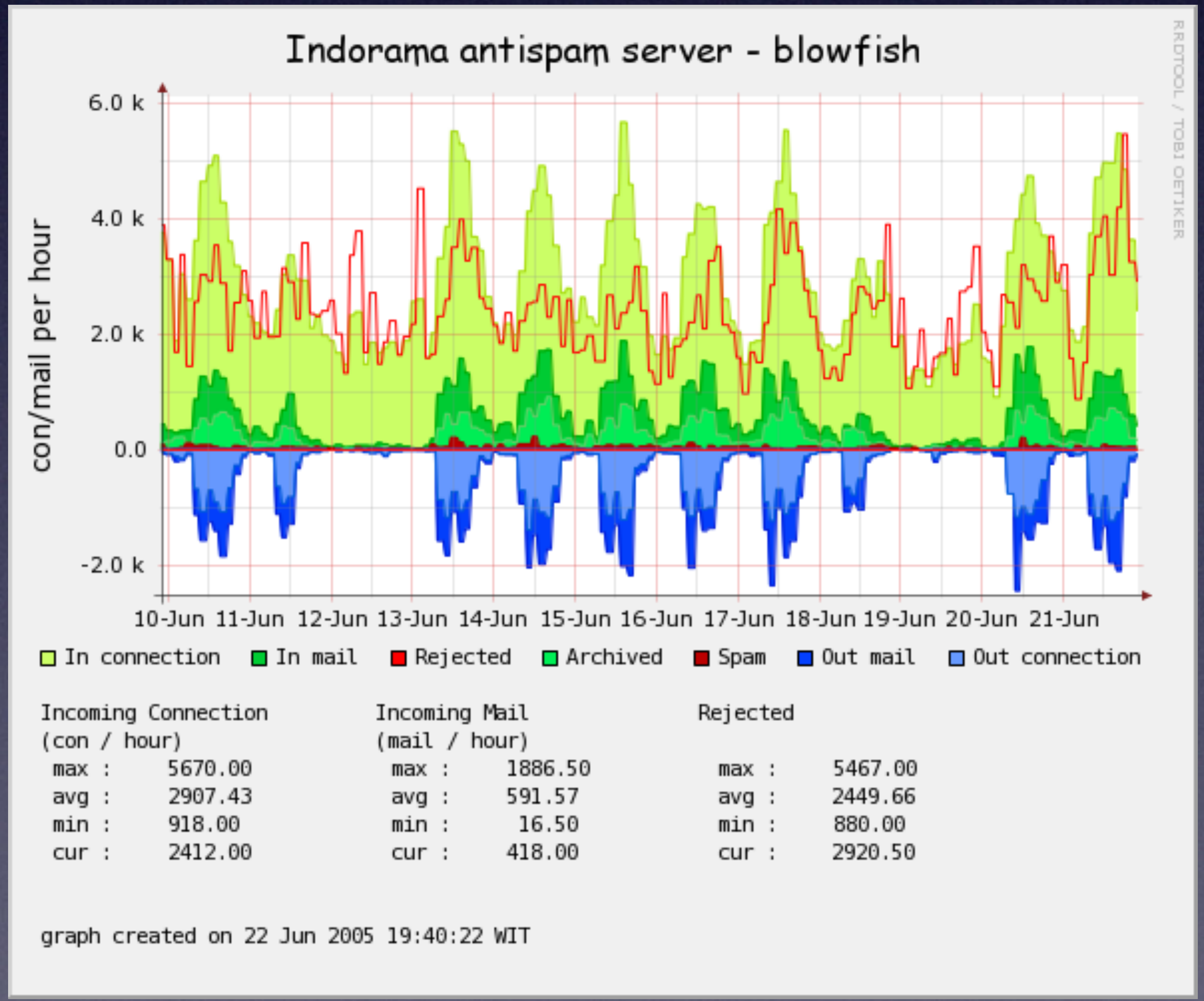
```
   Trunk Grp      Ports   In Use   Current Load   Average Load   Peak Load
🟥 LBTX28xxx0      30       1          2.8%          10.7%         29.7%
🟧 LBTX26xxx1      30       1          3.8%          20.6%         72.8%
🟨 LBTX21xxx2      30      20         65.0%          85.4%         97.7%
🟩 LBTX21xxx3      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%
```



```
                     From 2014/03/19 14:13:10 To 2014/03/20 14:13:10

🟨 In6/0/3:0 Outbound  Current:   10.80 M  Average:    9.53 M  Maximum:   27.90 M
🟧 In6/0/3:1 Outbound  Current:   11.72 M  Average:    8.20 M  Maximum:   29.81 M
🟫 In6/0/3:2 Outbound  Current:    2.03 M  Average:    8.67 M  Maximum:   35.65 M
🟫 In6/0/3:3 Outbound  Current:    9.53 M  Average:   10.25 M  Maximum:   35.00 M
🟥 Wi6/0/3:0 Outbound  Current:    6.69 M  Average:   11.58 M  Maximum:   33.92 M
🟫 ALL ITEMS           Current:   40.78 M  Average:   48.02 M  Maximum:  121.48 M
```



```
          From 2009/04/07 14:31:08 To 2009/04/08 14:31:08

🟩 Inbound    Current:   62.43 M  Average:   39.12 M  Maximum:  112.22 M  🟩 Peak Usage
🟦 Outbound   Current:   70.62 M  Average:   42.98 M  Maximum:  127.67 M  🟦 Peak Usage
```



```
🟩 In connection  🟩 In mail  🟥 Rejected  🟩 Archived  🟥 Spam  🟦 Out mail  🟦 Out connection

Incoming Connection      Incoming Mail          Rejected
(con / hour)             (mail / hour)
 max :     5670.00        max :     1886.50      max :     5467.00
 avg :     2907.43        avg :     591.57       avg :     2449.66
 min :     918.00         min :      16.50       min :     880.00
 cur :     2412.00        cur :     418.00       cur :     2920.50


graph created on 22 Jun 2005 19:40:22 WIT
```

Things improved with Cacti

Were our systems observable?

Kind of?

Intuition

Limiting Factors

Cost of storage

Cost of compute

Grepping lots of logs

No TSDBs or NoSQL - Filesystems and RDBMSs

No context from previous failures except gzipped collections of logs and MRTG graphs maybe screenshots

Limiting Factors

Your sysadmins problem

Not quite configuration management

Awful technologies (SNMP)

Mostly statically generated assets from logs

Tools were macro world view

Sysadmin Squad Goals of Yore

Not letting engineers access production

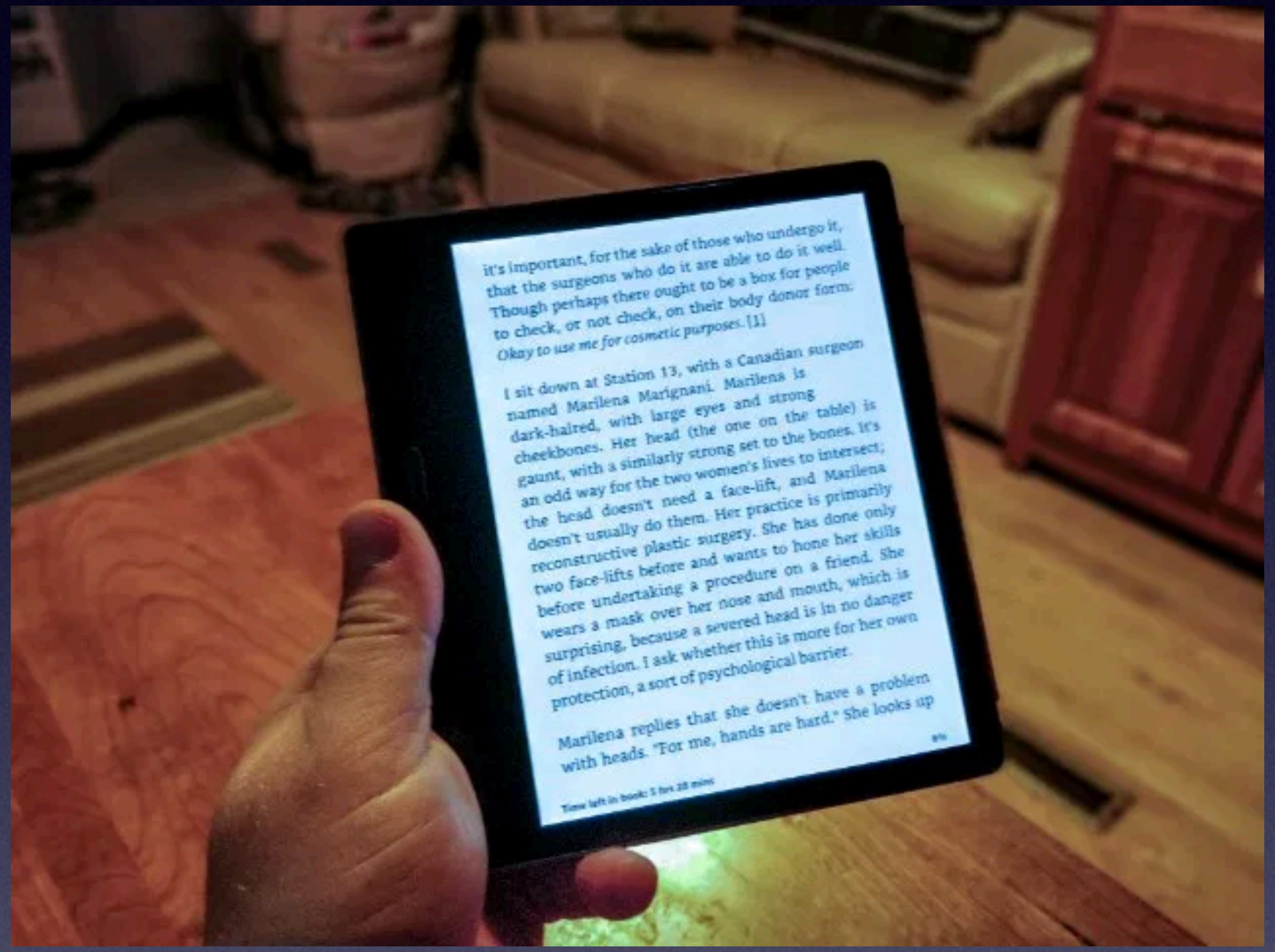
Having monitoring at all

Getting alerted reliably

Avoiding breaking things

Uptime, preferably **5 Nines**

Cross training, teaching intuition



it's important, for the sake of those who undergo it, that the surgeons who do it are able to do it well. Though perhaps there ought to be a box for people to check, or not check, on their body donor form: *Okay to use me for cosmetic purposes.* [1]

I sit down at Station 13, with a Canadian surgeon named Marilena Marignani. Marilena is dark-haired, with large eyes and strong cheekbones. Her head (the one on the table) is gaunt, with a similarly strong set to the bones. It's an odd way for the two women's lives to intersect; the head doesn't need a face-lift, and Marilena doesn't usually do them. Her practice is primarily reconstructive plastic surgery. She has done only two face-lifts before and wants to hone her skills before undertaking a procedure on a friend. She wears a mask over her nose and mouth, which is surprising, because a severed head is in no danger of infection. I ask whether this is more for her own protection, a sort of psychological barrier.

Marilena replies that she doesn't have a problem with heads. "For me, hands are hard." She looks up

Time left in book: 5 hrs 28 mins

The Present

SSD AWS Cloud Docker

Schedulers (Kubernetes) Grafana SaaS SRE

NoSQL Rest TSDBs APM

A lot.

Azure 15 Years of Moore's Law DevOps

Log Management Containers Containers GCP

Open-source SDN VMs

Microservices CNCF Distributed Systems

Key Factors Driving Observability's Adoption

Democratization of technology (DBs, Log aggregation, VMs)

Tradeoffs are much different today

Monitoring is about predicting failure, harder to predict today due to the complexity

Clouds are ephemeral

Microservices

Key Factors Driving Observability's Adoption

Multi-Region / Multi-cloud concerns

Complexity has increased exponentially

Monitoring and metrics systems are far more advanced, less brittle, many times built in

Many more disparate systems / APIs (internal and external)

Scale has increased dramatically for even simple applications

4 Pillars of Observability

Metrics

Logging

Distributed Tracing

UX (Alerting/ Visualization)

Metrics

Choose your own adventure, there are so many to choose from, beware of cardinality concerns (don’t tag metrics with UUIDs)

More predictable overhead than logs

Lower integration costs for applications than in the past, usually just a library or API call

You don’t always know what metrics you need

Logs

Contextualizes the lifecycle of a requests

The least standardized component of most environments

Generation is the easy part, storing is not

Finite amount

Sampling for the long term if necessary

Distributed Tracing

Newest component to modern infrastructure

Correlation IDs are a good place to start

UI not required, but more valuable if you do

Not APM

Difficult to retrofit unless you use a service mesh

Requires engineering wide adoption

Alerting / Monitoring / UX

Alert fatigue, make actionable alerts, not just your SREs

Alerts must include context

The solution is never, simply, to make a alert and dashboard for X thing

Duplication and one-off dashboards

Manage the entire stack with configuration management

Current Groundwork

Engineers want APMish things

The rise of SaaS Infrastructure Companies…… Scaling is Hard

Log aggregation (at scale) is easier: Humio, ELK, SaaS

Many more options than the past

Current Groundwork

Open and closed source options abound

OpenTracing

Systems capable of high throughput and ability to query data that has high cardinality

Integration of disparate systems simpler

How can I make my systems observable?

Start with Empathy for the User

The end goal is to make the best user experience possible

Users don’t care about your CPU Load

Users care about errors and latency

All of your observability goals should be focused on answering questions about end user experience

How?

Gather as much data as you can from any path a user takes and align the context from disparate systems around individual users / geographies (CDNs, Load Balancers, etc)

Instrument your clients focusing on key experiences in your applications as well as basic interactions such as DNS

How?

Adopt a DevOps approach in your organization with observability concerns

Standardize your engineering organizations requirements for observability, give engineers a carrot

Your goals will be congruent with other departments such as business and marketing

Don’t keep it a secret

New Approaches to Consider

AI / Machine Learning

Adopting stream processing infrastructure

Chaos Engineering

Testing in production

Sampling

Challenges

Percentiles are misleading at scale

Log amounts and sizes will increase

Scaling and spiky work loads

Complexity of tools when self managed

Many, especially edge, integrations are asynchronous

Challenges

Shoehorning into existing tools can lead to technical debt, substandard solutions

SaaS Cost at Scale

SaaS / Application Lock-in

Once SaaS cost is an issue, so is rolling your own

Not adopting standard approaches in your organization

Challenges

Intuition is still a powerful, necessary, tool

Uptime and MTTR are important, quality user experience encompasses both

Uptime is easier to come by than ever before

Best Practices

Not one size fits all, or even many, be mindful of your specific tradeoffs

Integrate systems, such as your CRM, Zendesk, etc for additional context

Balance cost and usage tradeoffs

Have your observability config live with your application code, deploy it with your CD system, bonus points for CI

Best Practices

Start small, one application end to end

Maximize context at every step of your request path

Be wary edge aggregation, you lose context

Canary deployments that are easy to see in your UX, handles real (your) user traffic

Developers on call

The Future

The next 10 Years

Cheaper, faster, better systems and clouds

Less complexity to the tooling

Less tool sprawl

Less SaaS

More real-time

Streaming native infrastructure, less HTTP APIs, log shipping

The next 10 Years

Observability Engineers and Orgs

Users are first class concerns for engineers

Sample on and off dynamically, by user?

Synchronous Edge Observability

Making everything observable, your team, your repos, etc will be commonplace

Building Your Observability Culture

It’s never over, there are always new questions

Promote the power of the ethos outside of engineering

Use context from your business to inform your focus on what is important to observe

Your observability will provide business value, push it up and out

Building Your Observability Culture

Foster a culture of accountability, you own your uptime and errors, as an organization

Empower engineers to make it easy to input and utilize the data, everyone instruments

Make it easy for first time users to take actionable measures

Culture of empathy for users, and each other

Brian Knox, Digital Ocean:

The goal of an Observability team is not to collect logs, metrics, or traces. It is to build a culture of engineering based on facts and feedback, and then spread that culture within the broader organization.

Acknowledgements

The Infrastructure Team at Vevo

Humio @meethumio (https://humio.com)

Cindy Sridharan, Apple, @copyconstruct (https://medium.com/@copyconstruct)

Charity Majors et. all, honeycomb.io @mipsytipsy

Visit us at Kubecon EU at Booth #SE54

Try Humio for Free
https://humio.com/getting-started
30 Day Free Trial - Free Cloud Tier

Thank you!

@schofield - grant@humio.com

Notes and Sources

https://www.humio.com/chaos-observability
https://distributed-systems-observability-ebook.humio.com/

https://medium.com/humio/data-driven-observabilty-logs-52e98e27a83b

https://distributed-systems-observability-ebook.humio.com/

https://www.honeycomb.io/resources/white-papers/

RRDTool https://calomel.org/rrdtool.html

Cacti - Many google image searches

https://en.wikipedia.org/wiki/Control_theory

https://www.vividcortex.com/blog/monitoring-isnt-observability