The Assimilation Monitoring Project

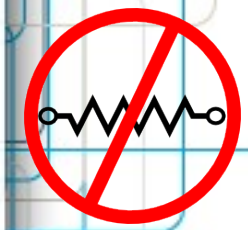
or

How to assimilate a million servers
and not get indigestion

#AssimMon @OSSAlanR

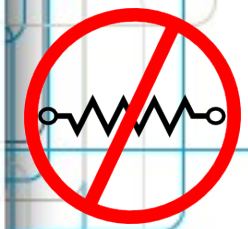
http://assimmon.org/

Alan Robertson <alanr@unix.sh>
Project Founder



Project background

- Sub-project of the Linux-HA project
- Personal-time open source project
- Currently license: slightly modified LGPL, small amount GPL
- Currently around 25K lines of code
- A work-in-progress

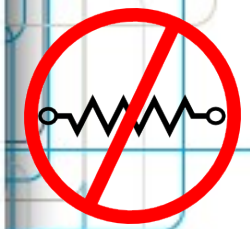


Project Scope

Discovery-Driven Exception Monitoring of Systems and Services

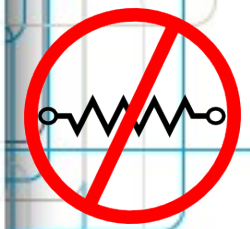
- **EXTREME** monitoring scalability
  >> 10K systems without breathing hard

- Integrated Continuous Stealth Discovery™: systems, switches, services, and dependencies – *without setting off network alarms*



Problems Addressed

- Scaling monitoring up nearly indefinitely
- Discovering systems, services, switches and dependencies without setting of network security alarms
- Distinguishing server from switch failures
- Using dependency information to highlight root causes of failures



Architectural Overview

Collective Monitoring Authority (CMA)

- One authority per monitored environment

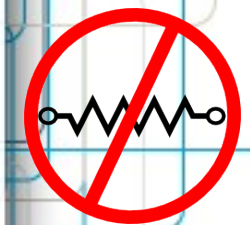
Nanoprobes

- One nanoprobe per running OS image

Data Storage

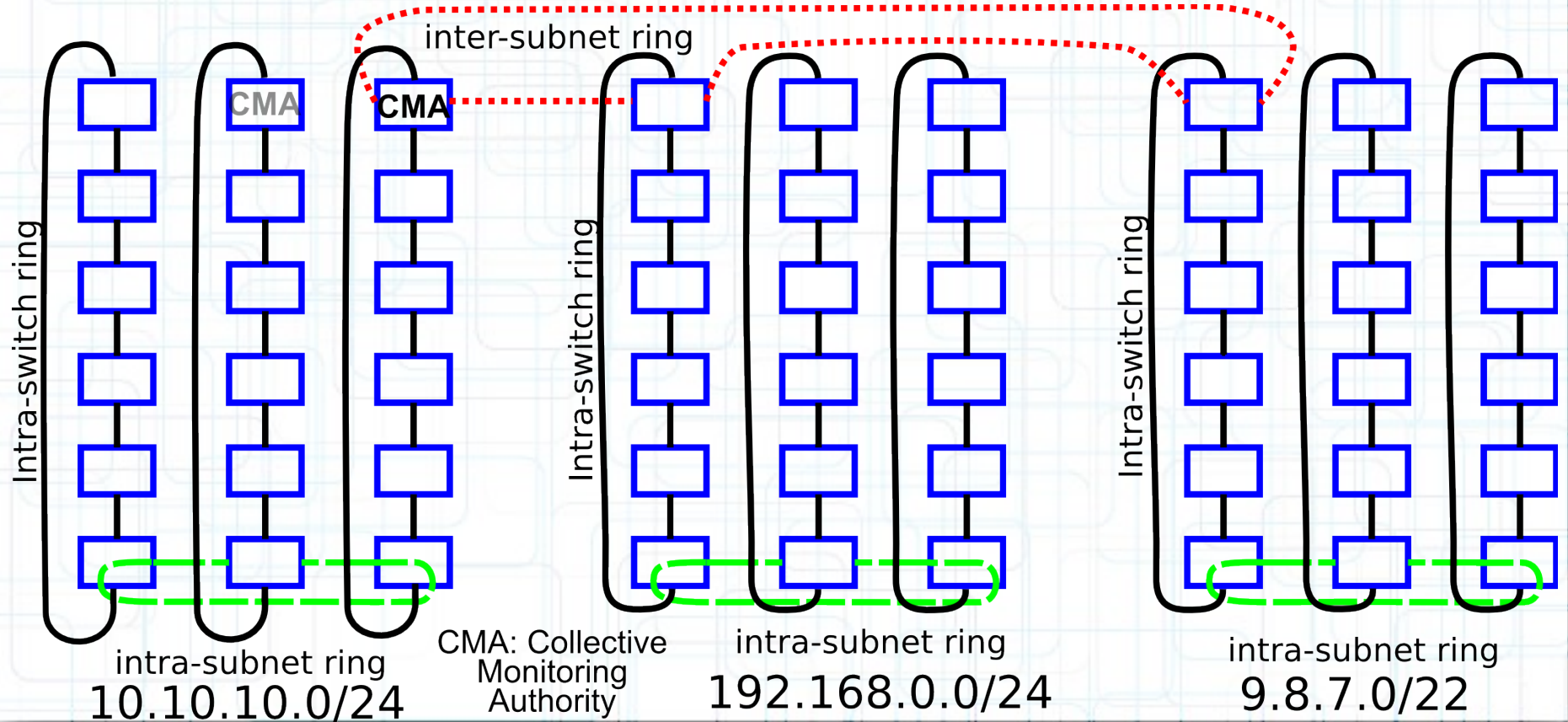
- Central Neo4j graph database

- Discovery, Server monitoring need *no configuration at all*
- General Rule: “No News Is Good News”



Massive Scalability – *or* "I see dead servers in *O*(1) time"

- **Adding systems does not increase the monitoring work on any system**
- Each server monitors 2 or 4 neighbors
- Each server monitors its own *services*
- Ring repair and alerting is *O*(n) – but a very small amount of work
  - Ring repair for a million nodes is less than 10K packets *per day*



Continuous Integrated Stealth Discovery

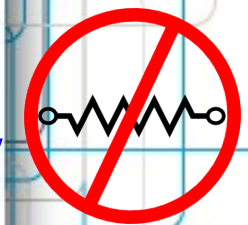
Continuous - Ongoing, incremental

Integrated - Monitoring does discovery; stored in same database

Stealth - No network privileges needed - *no port scans or pings*

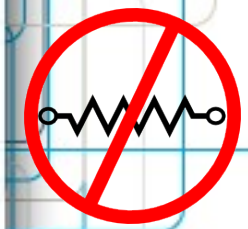
Discovery - Systems, switches, clients, services and *dependencies*

➔ **Up-to-date picture of pieces & how they work w/o "network security jail" :-D**



Service Monitoring

- Service monitoring by Linux-HA LRM daemon
  - LRM == Local Resource Manager
- Well-proven code
  - No-news is good news
  - Several classes of resource agents
- Implements Open Cluster Framework standard
- Each system monitors own services
- Also able to start, stop & migrate resources



Monitoring Pros and Cons

Pros

Scalable to ≈ any size

Easy to understand

Uniform work distribution

No single point of failure

Distinguishes switch vs host failure

Geographically sensitive

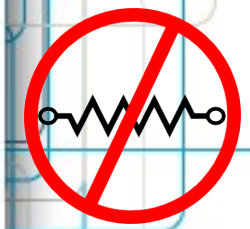
Lightweight on network

Cons

Requires active agents

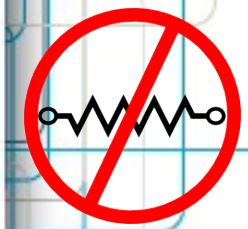
More complex than ping script

Potential slowness at power-on



Basic CMA Functions (python)

- Configure & direct nanoprobes
- Listen to nanoprobe alerts & discovery
- Updates rings: nanoprobe join/leave
- Updates database
- Issues alerts, suggests root causes
- Provides information for **UI**s (coming!)



Nanoprobe Functions ('C')

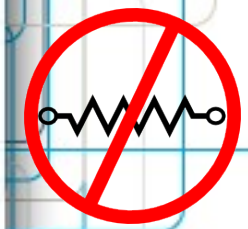
Announce self to CMA

- Reserved multicast address

Do what CMA says

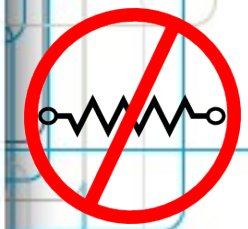
- receive configuration information
  - CMA addresses, ports, defaults
- send/expect heartbeats
- perform discovery actions
- perform monitoring actions
  - monitor services, thresholds, etc.

No persistent state



Why a graph database? (Neo4j)

- Dependency & Discovery information: graph
- Circular monitoring structures: graph
- Root cause queries ⇨ graph traversals – notoriously slow in relational databases
- Visualization of relationships
- Avoid converting relational→graph & back
- Schema-less design: good for constantly changing heterogeneous environment



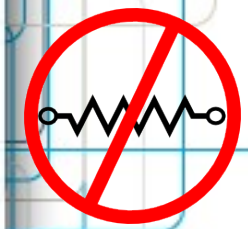
How does discovery work?

Nanoprobe scripts perform discovery

- Each discovers one kind of information
- Can take arguments (in environment)
- Output **JSON**

CMA stores Discovery Information

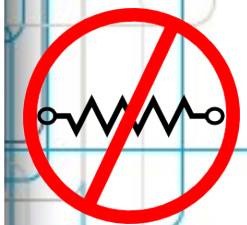
- JSON stored in Neo4j database
- Discovery types can have custom CMA plugins => graph nodes and relationships



sshd *Service* JSON Snippet (from netstat and /proc)

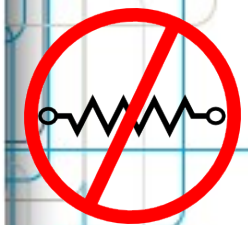
```
"sshd": {
  "exe":                    "/usr/sbin/sshd",
  "cmdline":                [ "/usr/sbin/sshd", "-D" ],
  "uid":                    "root",
  "gid":                    "root",
  "cwd":                    "/",
  "listenaddrs": {
    "0.0.0.0:22": {
      "proto":              "tcp",
      "addr":               "0.0.0.0",
      "port":               22
    },
```

and so on...

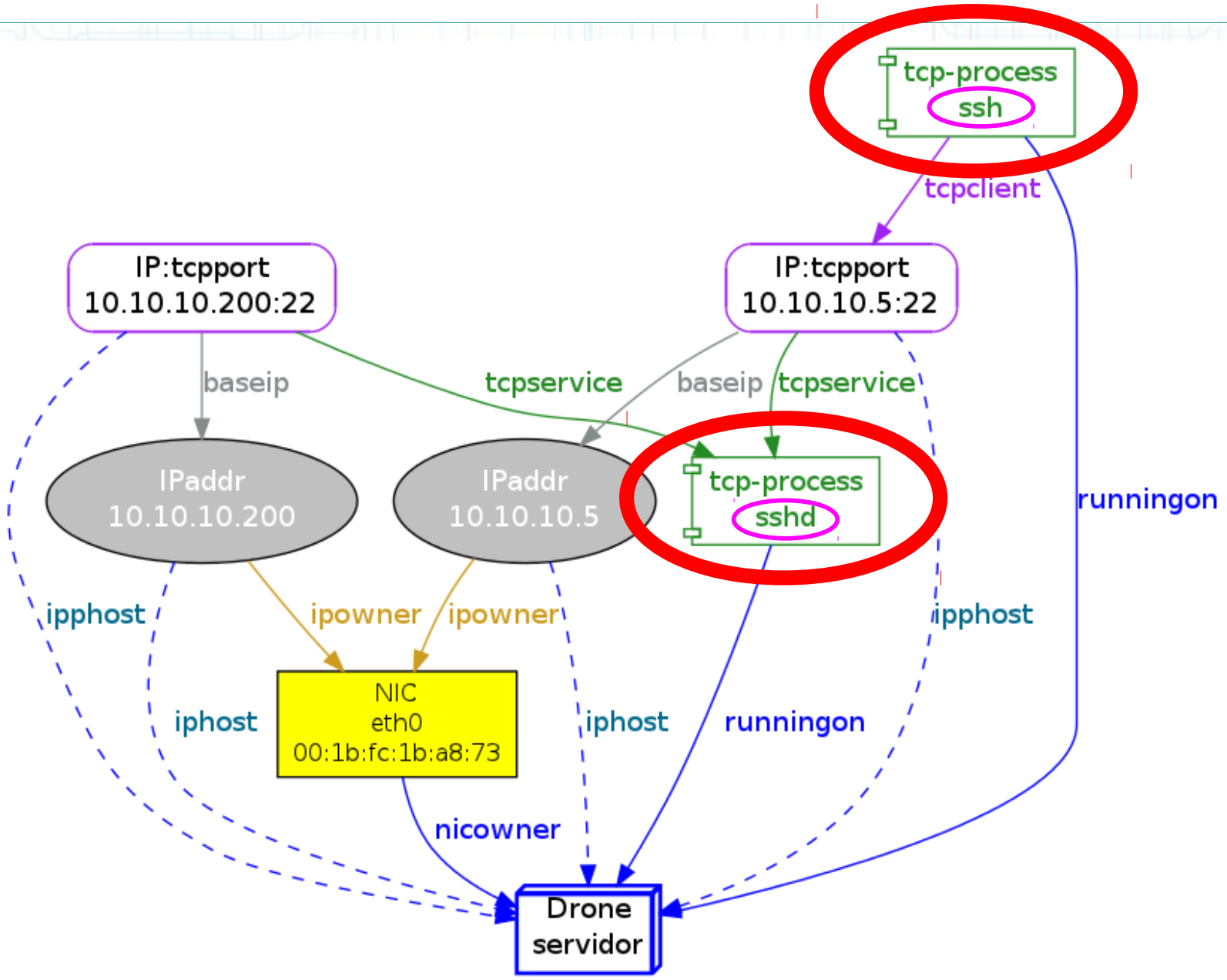


ssh *Client* JSON Snippet (from netstat and /proc)

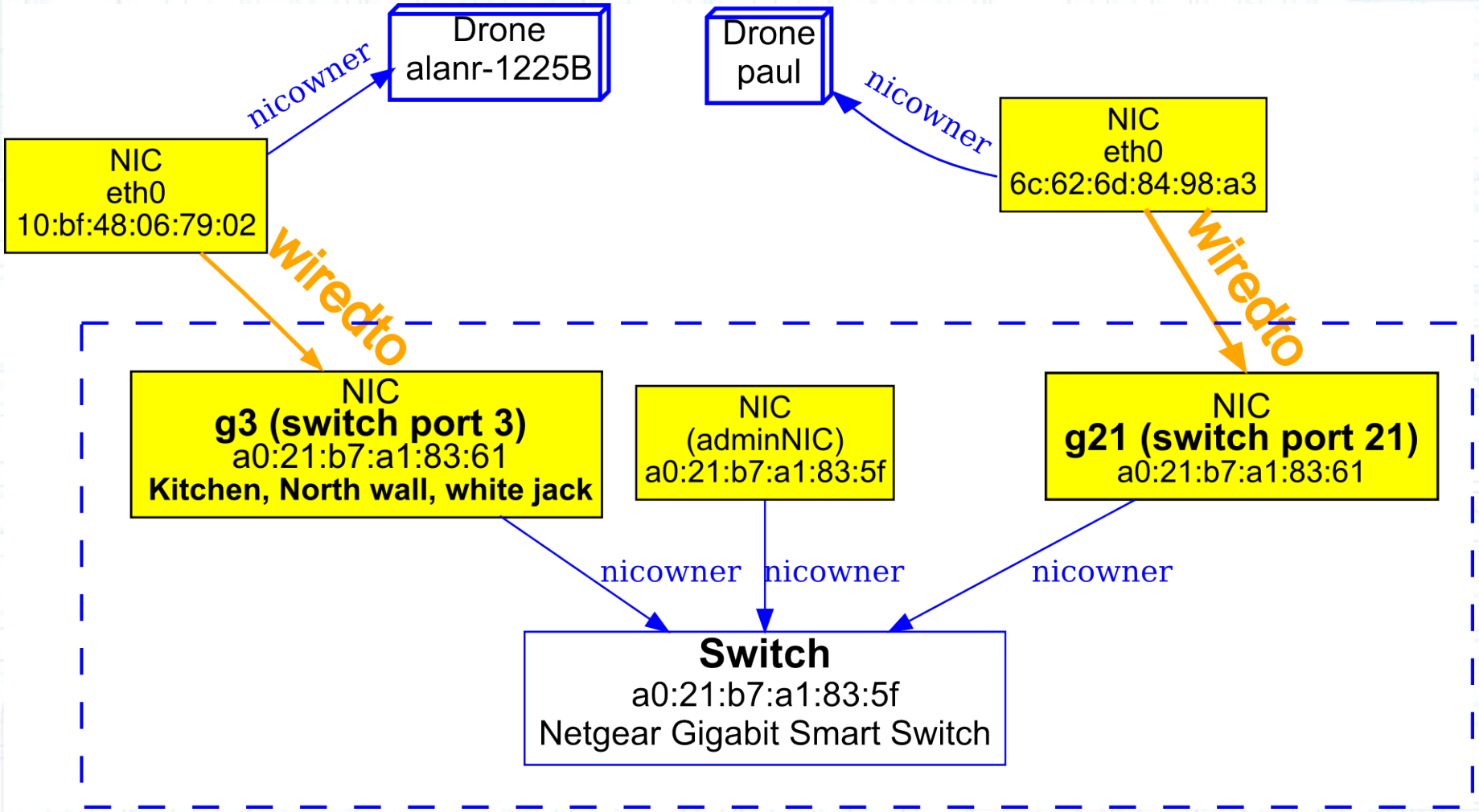
```
"ssh": {
  "exe":                  "/usr/sbin/ssh",
  "cmdline":              [ "ssh", "servidor" ],
  "uid":                  "alanr",
  "gid":                  "alanr",
  "cwd":                  "/home/alanr/monitor/src",
  "clientaddrs": {
    "10.10.10.5:22": {
      "proto":            "tcp",
      "addr":             "10.10.10.5",
      "port":             22
    },          and so on...
```



ssh -> sshd dependency graph



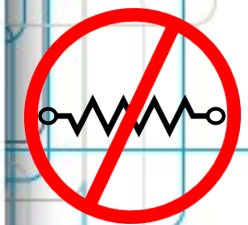
Switch Discovery Data from LLDP (or CDP)



Binary LLDP (CDP) Data is transformed to JSON by CRM

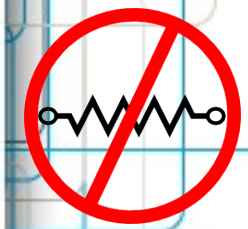
Current State

- Can build and play with: heartbeat, discovery
- Good unit test infrastructure
- Nanoprobe code:
  - Written and works well
  - Lacking:
    - Integration w/LRM
    - Real digital signatures, encryption, compression
- CMA code works, *much* more to go
- Several discovery methods written



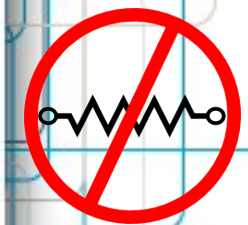
Future Plans

- First Release Planned End of 2012
- Integrate with LRM for Service Monitoring
- Dynamic (aka cloud) Infrastructure specialization
- Much more discovery
- Alerting
- Reporting
- Create / Audit / Be ITIL CMDB
- Add Statistical Monitoring



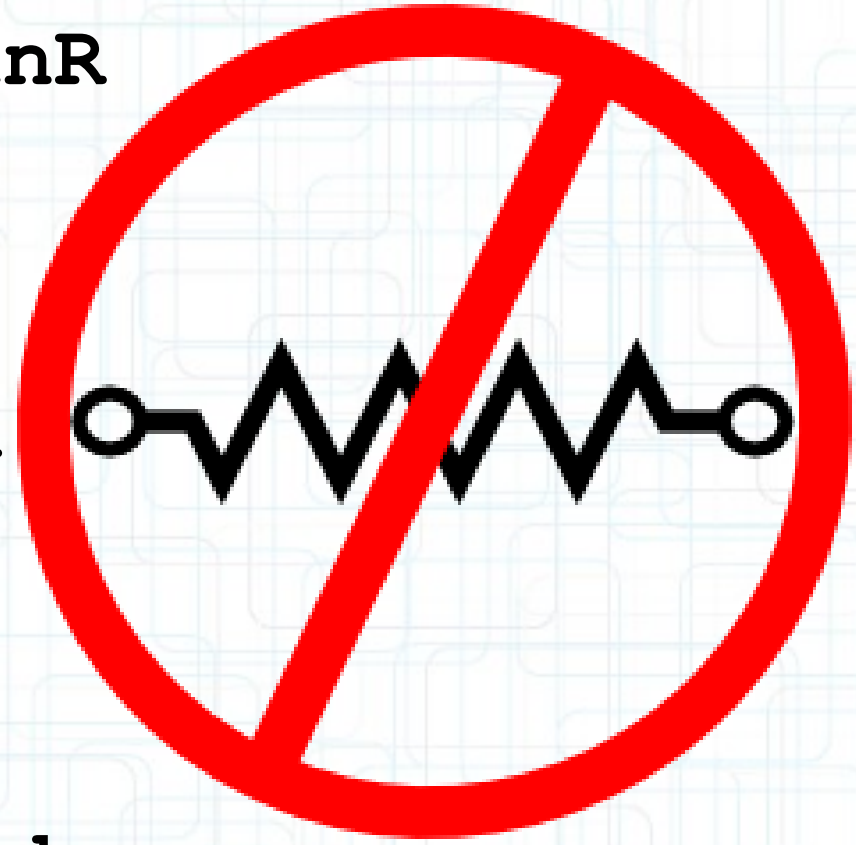
Get Involved!

- Powerful Ideas and Infrastucture
- Fun, ground-breaking project
- Incredible Opportunities
- Needs for *every* kind of skill
  - Awesome User Interfaces (UI/UX)
  - Test Code (simulate 10^6 servers!)
  - Packaging, Continuous Integration
  - Python, C, script coding
  - Evangelism, community building
  - Documentation
  - Feedback: Testing, Ideas, Plans



Resistance Is Futile!

#AssimMon @OSSAlanR



Project Web Site

http://assimmon.org

Blog

techthoughts.typepad.com

lists.community.tummy.com/cgi-bin/mailman/admin/assimilation