Running Reliable Services: Monitoring, Logging and Reporting

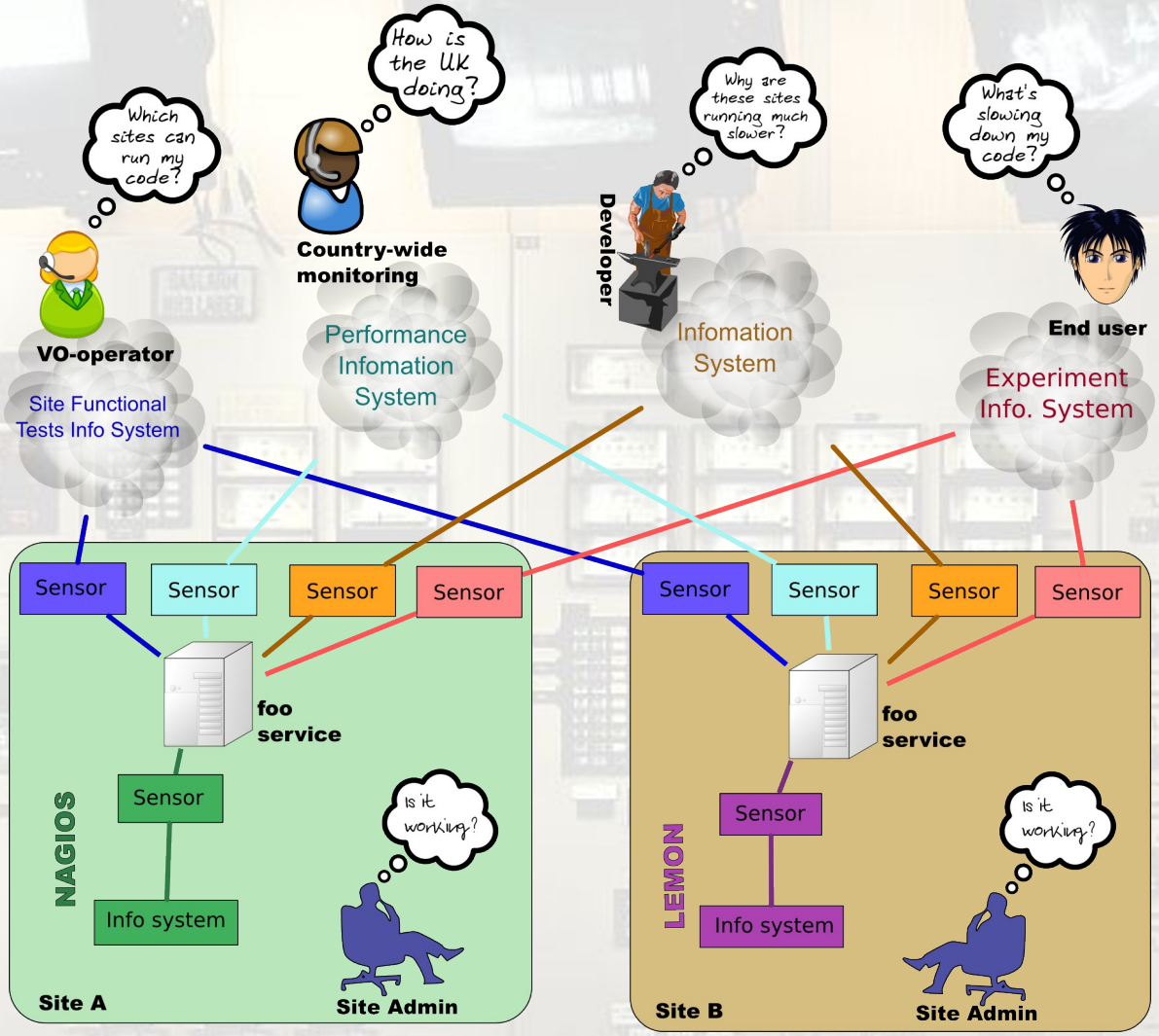
Paul Millar

UNIVERSITY of GLASGOW

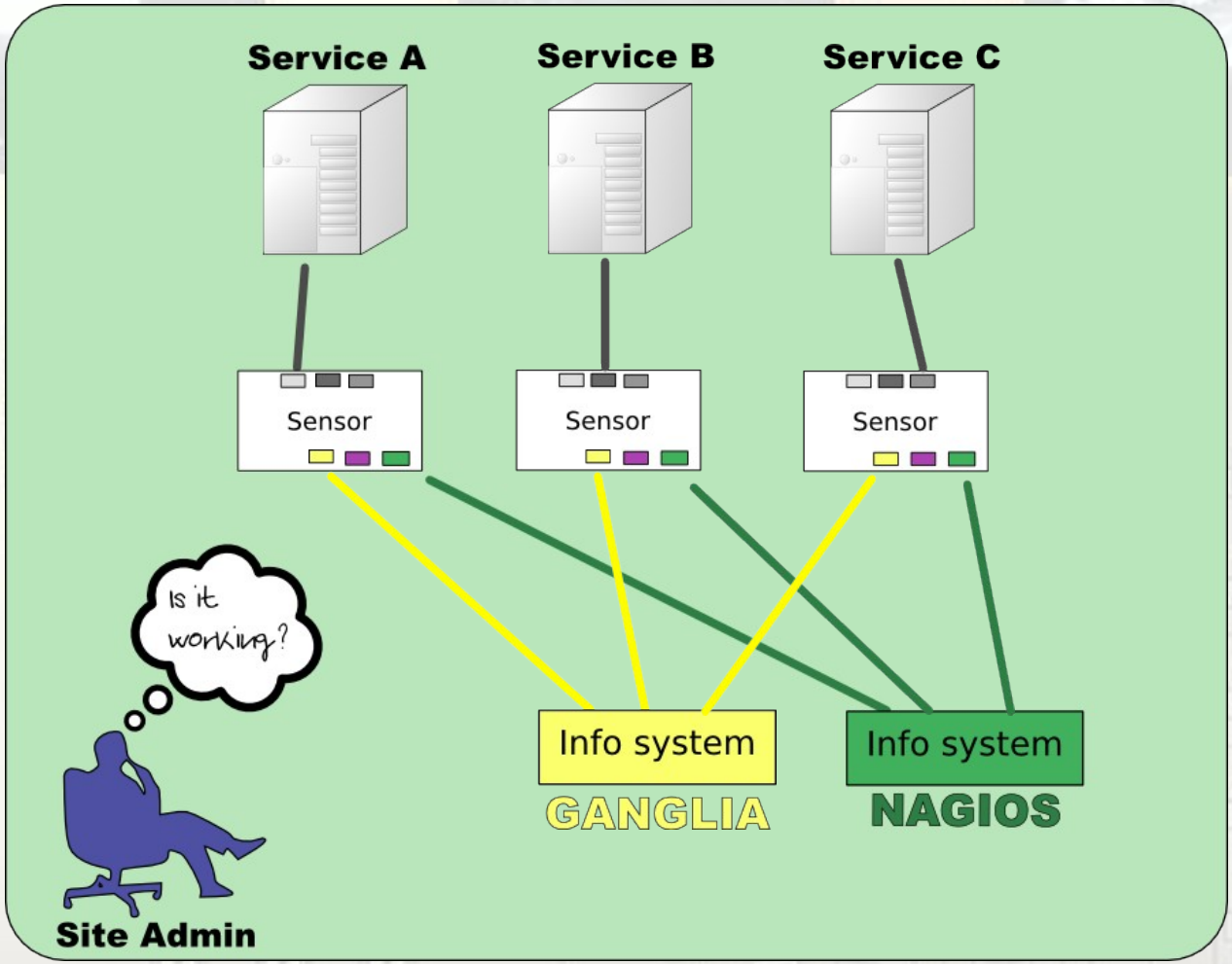
ScotGrid

GridPP
UK Computing for Particle Physics

Current situation



Ideal situation



MonAMI now

Monitoring:

MySQL,
Tomcat,
Apache,
process:
counting,
detailed info,
TCP connections,
NUT,
AMGA,
DPM,
dCache,
GridFTP.

Reporting:

Ganglia,
Nagios,
Ksysguard,
MonaLisa,
R-GMA,
File logs.

Looking at how to include this work within the WLCG Monitoring group.

http://monami.sourceforge.net/

Additional/Backup slides…

Robbie Burns,
Scotland's National Bard,
was born 248 years ago today.
Burns Suppers, held internationally,
involve poetry reading and singing,
various toasts (with whisky), and
eating "Haggis warm reeking, rich
wi' Champit Tatties,
Bashed Neeps."

Happy Burns day!

A brief history...

- Looking at "generic" metadata components.
- All metadata services needs to be a reliable.

  (which service doesn't?)
- Nothing is perfect: need to keep an eye on the components that make up a service.
- Needs to easily meshes with existing site fabric monitoring.
- Metadata service: metadata software, database, application container, network, free disk-space, ...

Types of monitoring

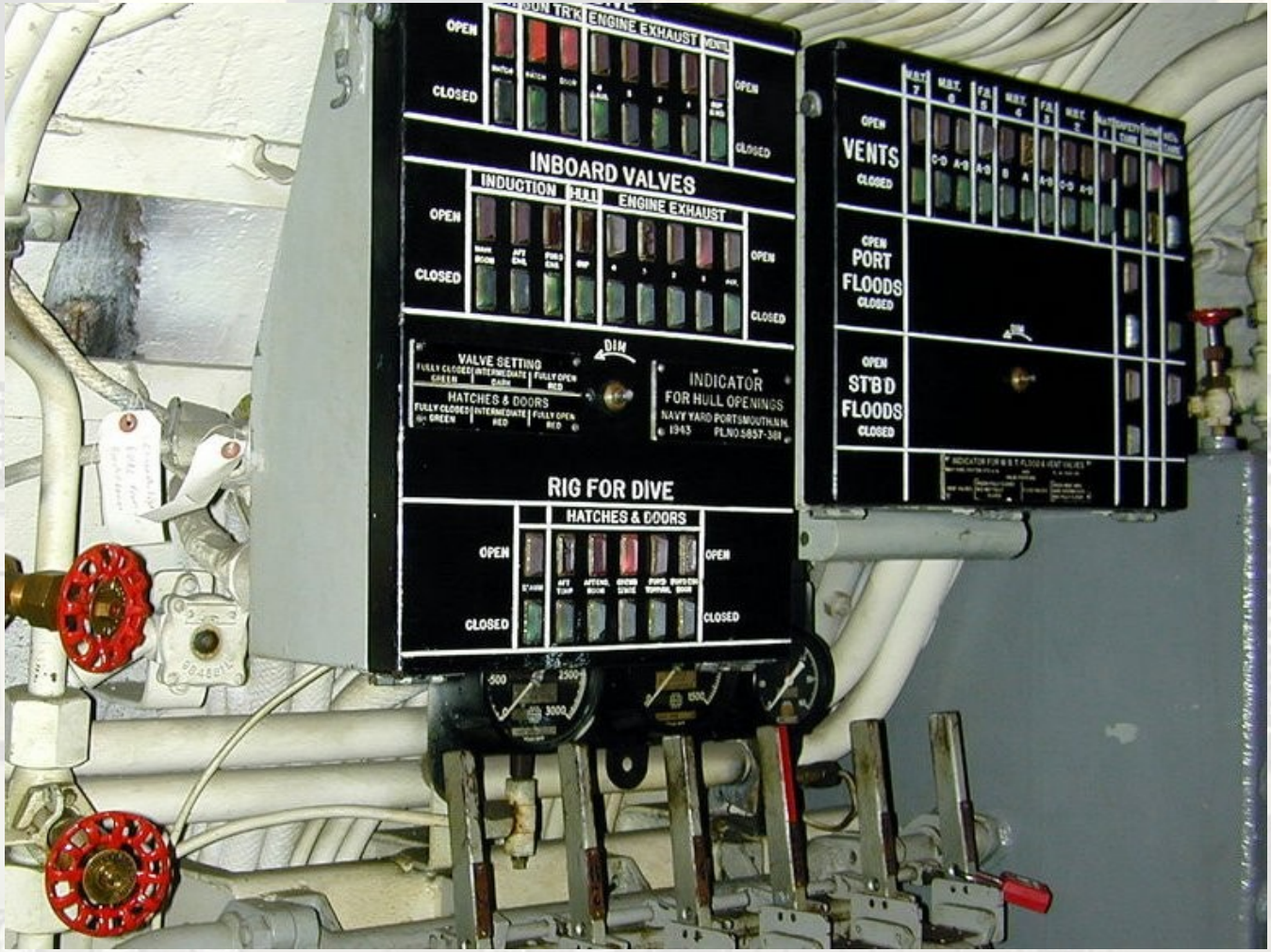
Overview

Alert

Graphical

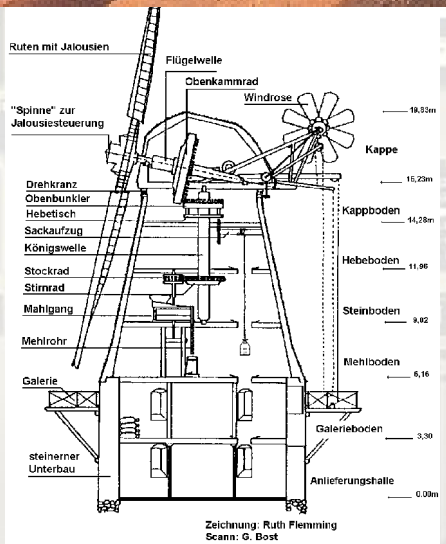
Logging

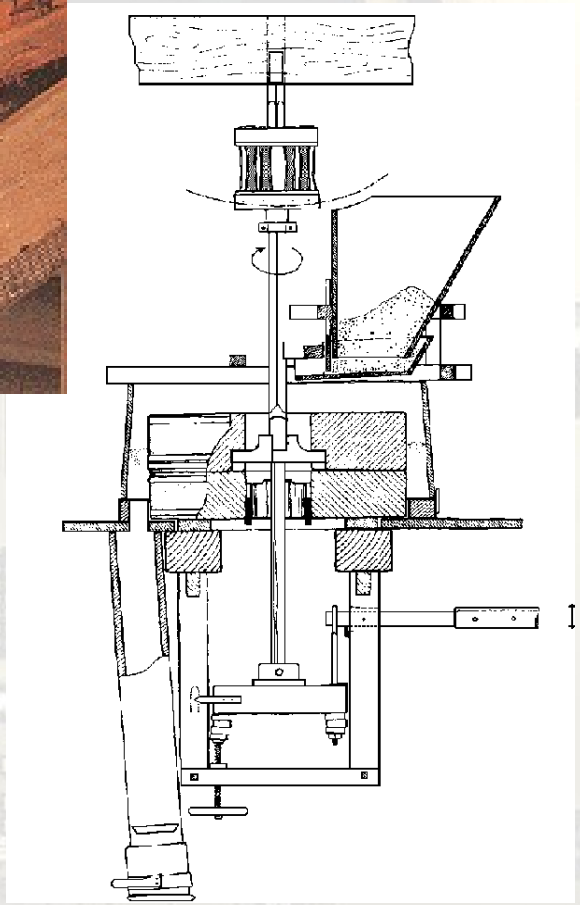
Detailed



Types of monitoring

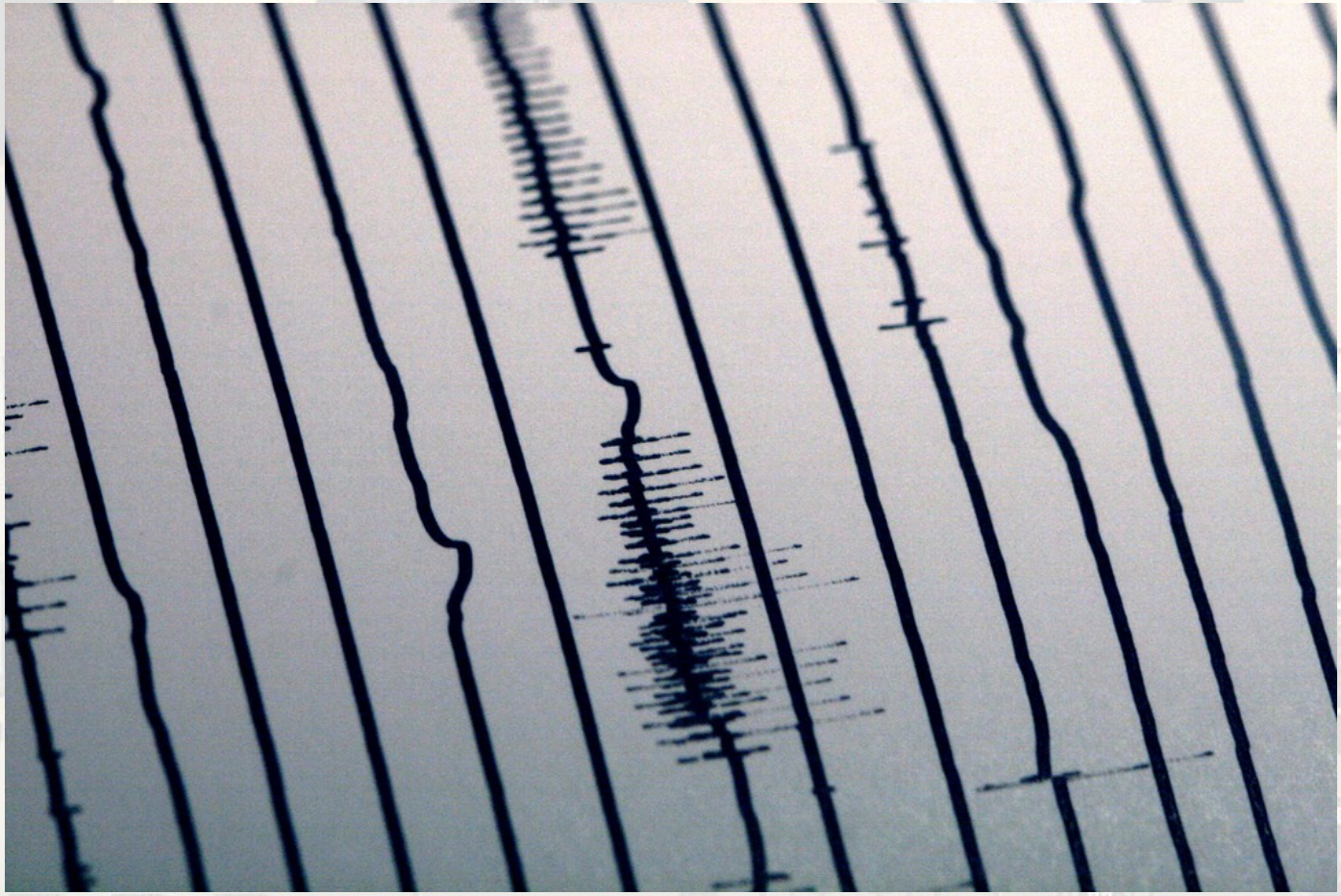
- Overview
- **Alert**
- Graphical
- Logging
- Detailed





Types of monitoring

- Overview
- Alert
- **Graphical**
- Logging
- Detailed



Types of monitoring

- Overview
- Alert
- Graphical
- **Logging**
- Detailed

Types of monitoring

Overview

Alert

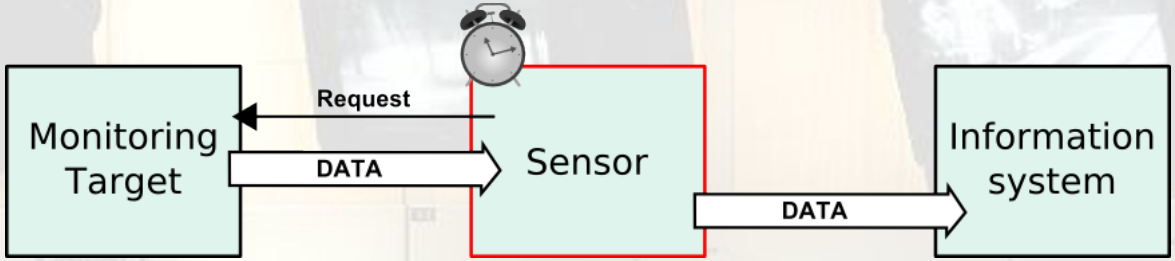
Graphical

Logging

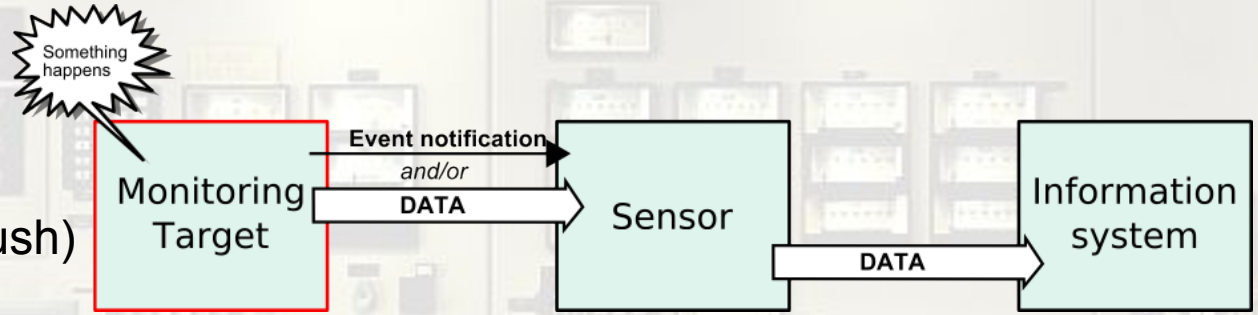
Detailed

Three basic interactions

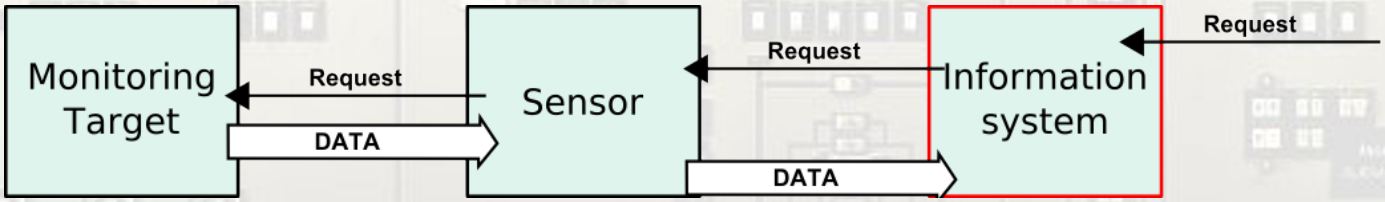
Polling
(synchronous push)



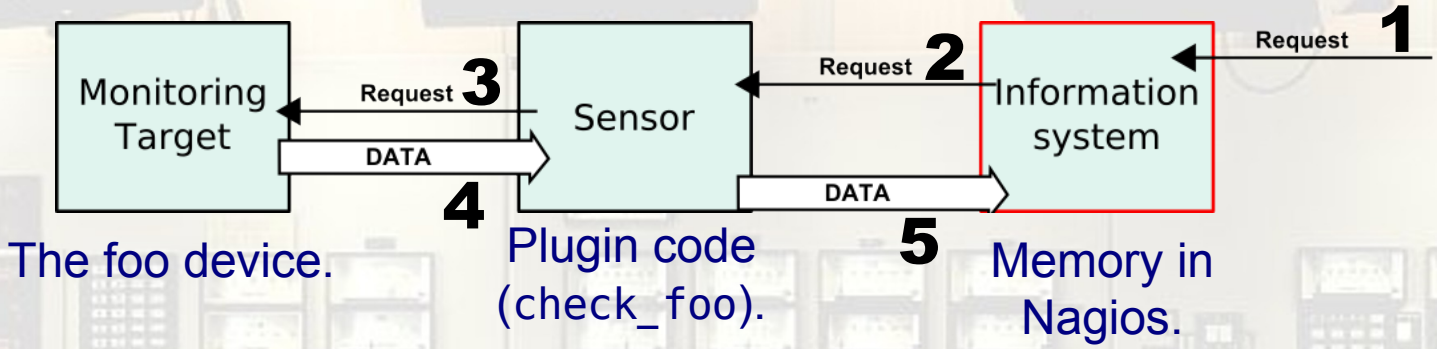
Events
(asynchronous push)



Detailed
Information Request
(asynchronous pull)

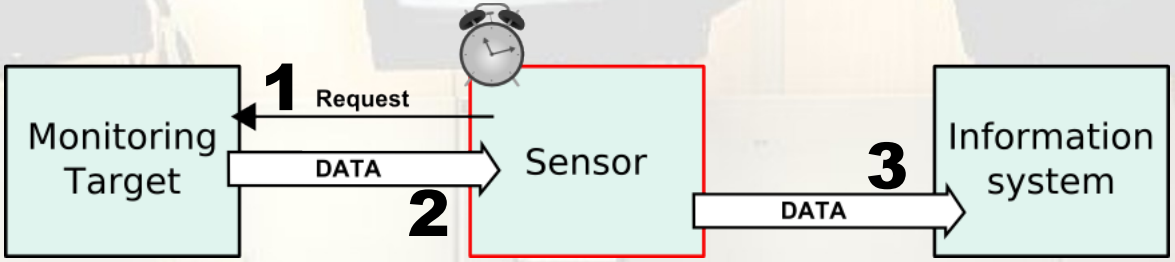


Case study: Nagios part 1 (Nagios plugins)



1. Nagios realises its time to check state of foo,
2. Nagios fork()s and exec()s, starting up the plugin,
3. Plugin requests foo current state,
4. It replies with current state,
5. Sensor sends back `OK`, `WARNING` or `CRITICAL`.

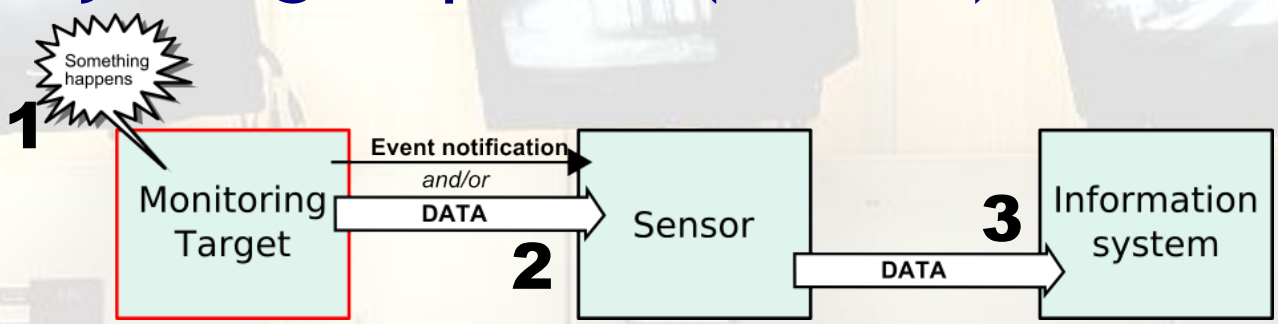
Case study: Nagios part 2 (state changes)



The foo plugin. foo monitoring within Nagios. Current state of monitored services

1. Nagios realises its time to check state of `foo`,
2. The `check_foo` plugin returns state `WARNING`,
3. This `WARNING` state is stored.

Case study: Nagios part 3 (alerts!)



1. State change! Service `foo` changes from `OK` to `WARNING`,
2. Notification handler picks up this change; configuration file indicates an email alert should be sent.
3. An email arrives in system-administrator's email INBOX

Seperating stakeholders

- MonAMI as a “monitoring service”.
- Different groups of people want to monitoring different things.
- Have seperate configuration files.
- Package a configuration file (or files) as an RPM, which depends on MonAMI.
- One can deploy as many of these monitoring RPMs as necessary.
- Site-admin can add additional monitoring.

Datatrees

- A tree of similar data (taken at the same time)
- Measurements have:
  - a name,
  - a value,
  - a type (e.g. integer, float, string),
  - (optionally) units,
  - additional metadata (static vs dynamic).
- Trees can be merged, subsets of trees selected

Anatomy of a plugin

- A plugin has:
  - a small structure describing what the plugin does and what are its capabilities,
  - functions monami_open and monami_close used to instantiate and destroy targets,
  - optionally monami_read and monami_write used to read or send data.

MonAMI

Stop people having to inventing their own sensors!

To provide data for different monitoring systems for different groups of people.

Not everyone wants to monitoring everything, so must be easy to configure!

http://monami.sourceforge.net/