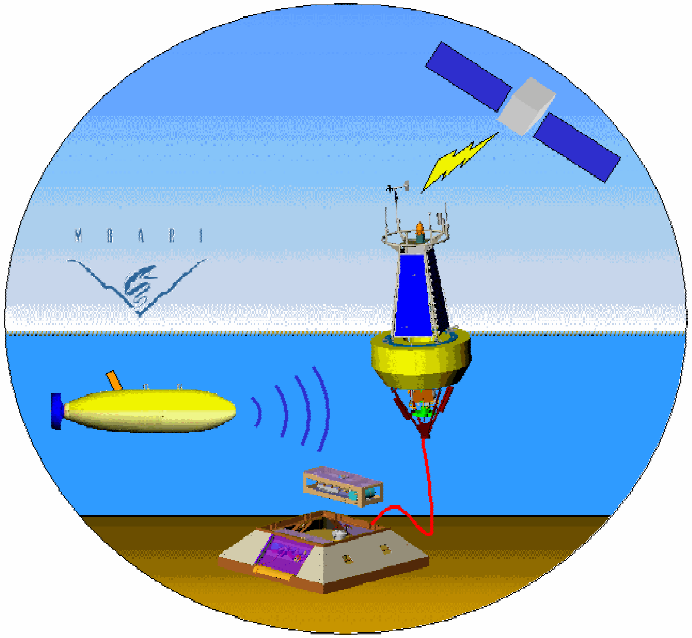
Some MOOS software features

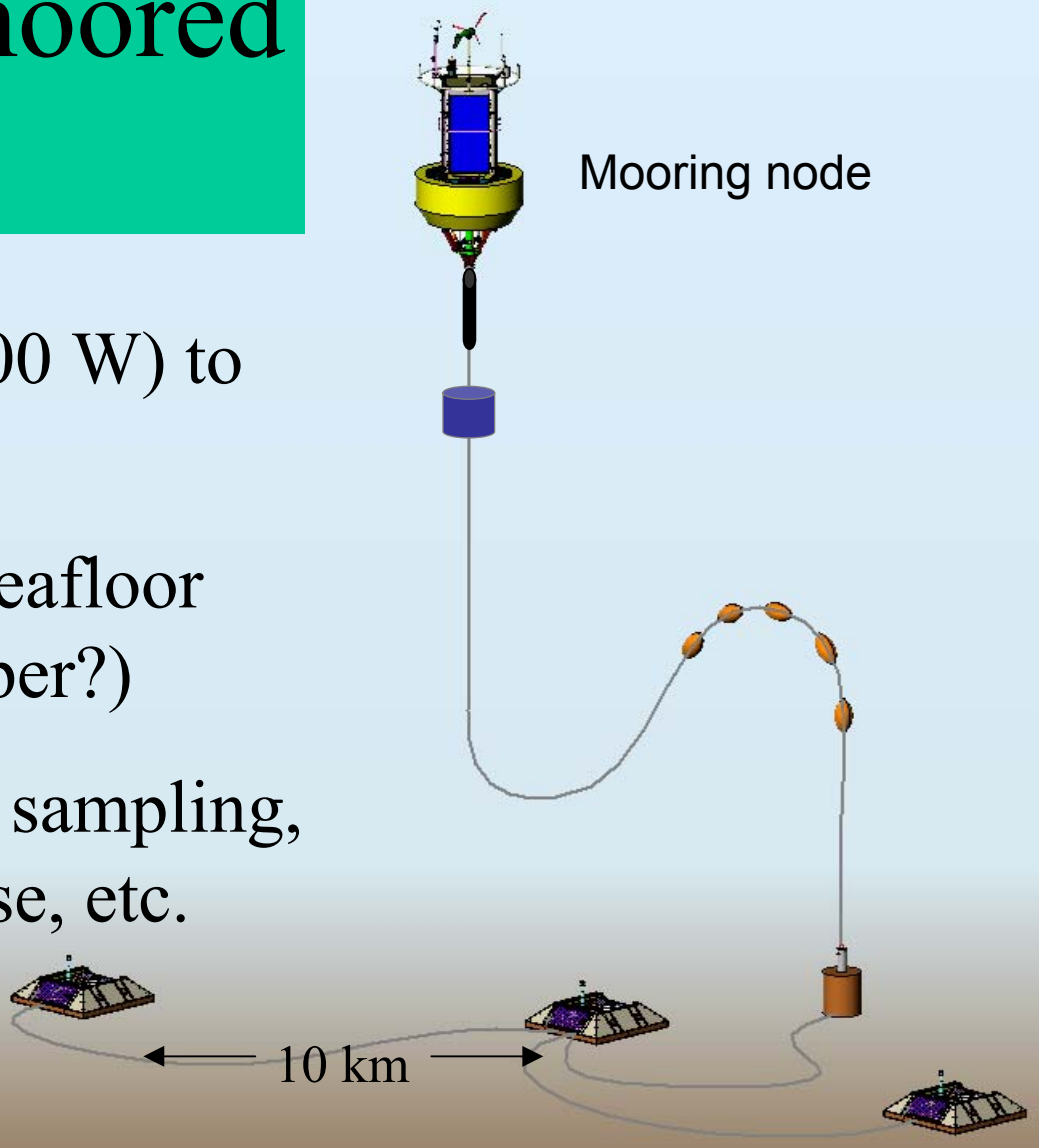


SIAM Team

January 29, 2003

MOOS moored network

- Power (20-100 W) to seafloor
- Network to seafloor (fiber or copper?)
- Autonomous sampling, event response, etc.



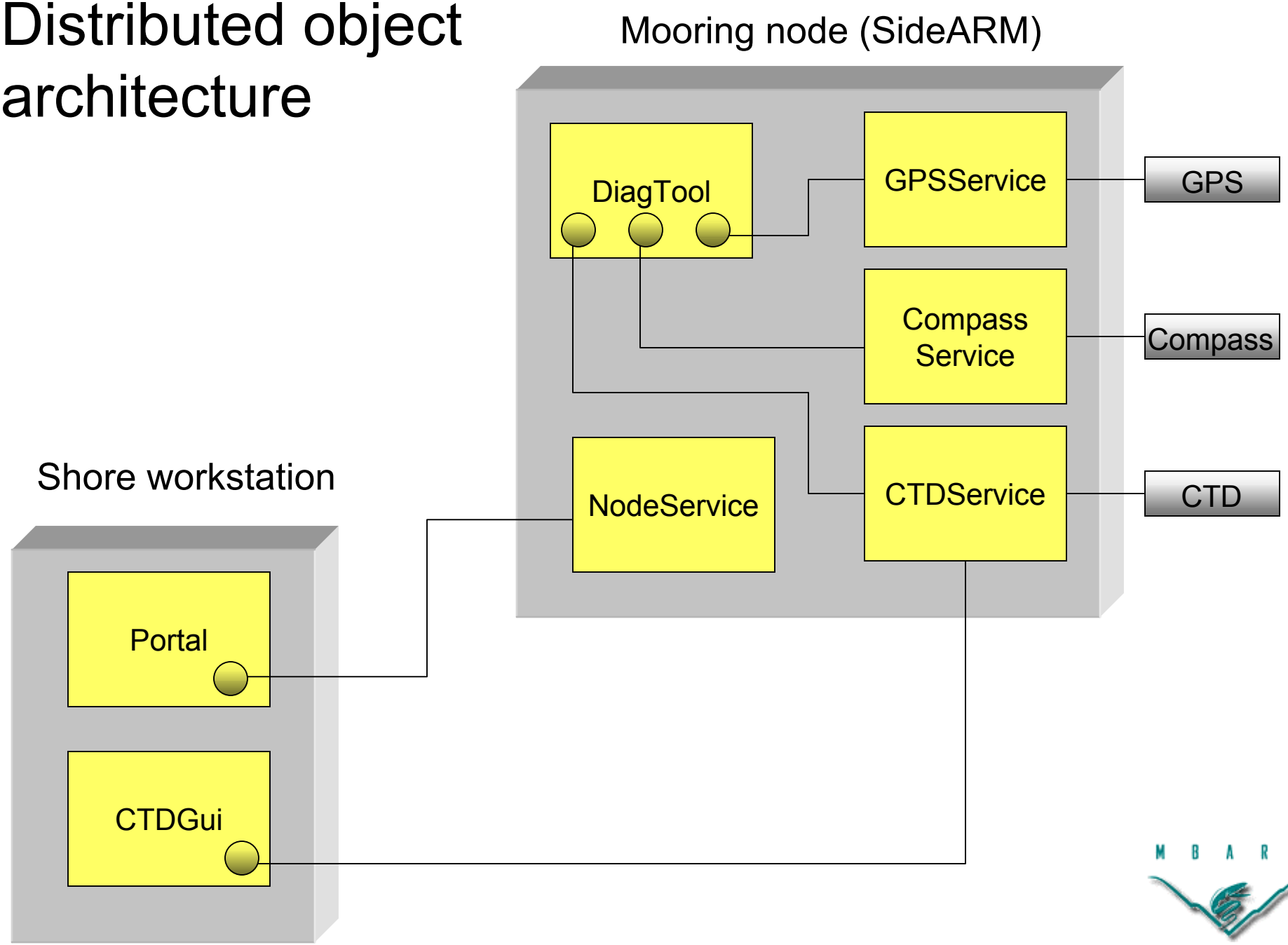
MBARI

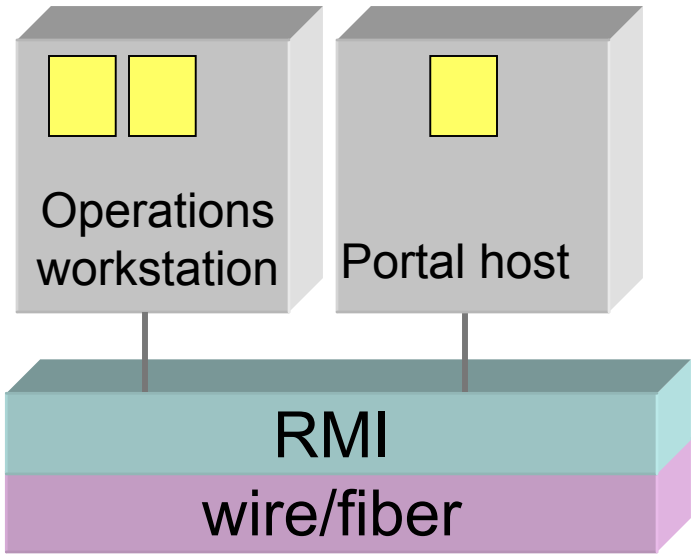
Distributed objects

- Software components interact across network
- Client components interact with *services*

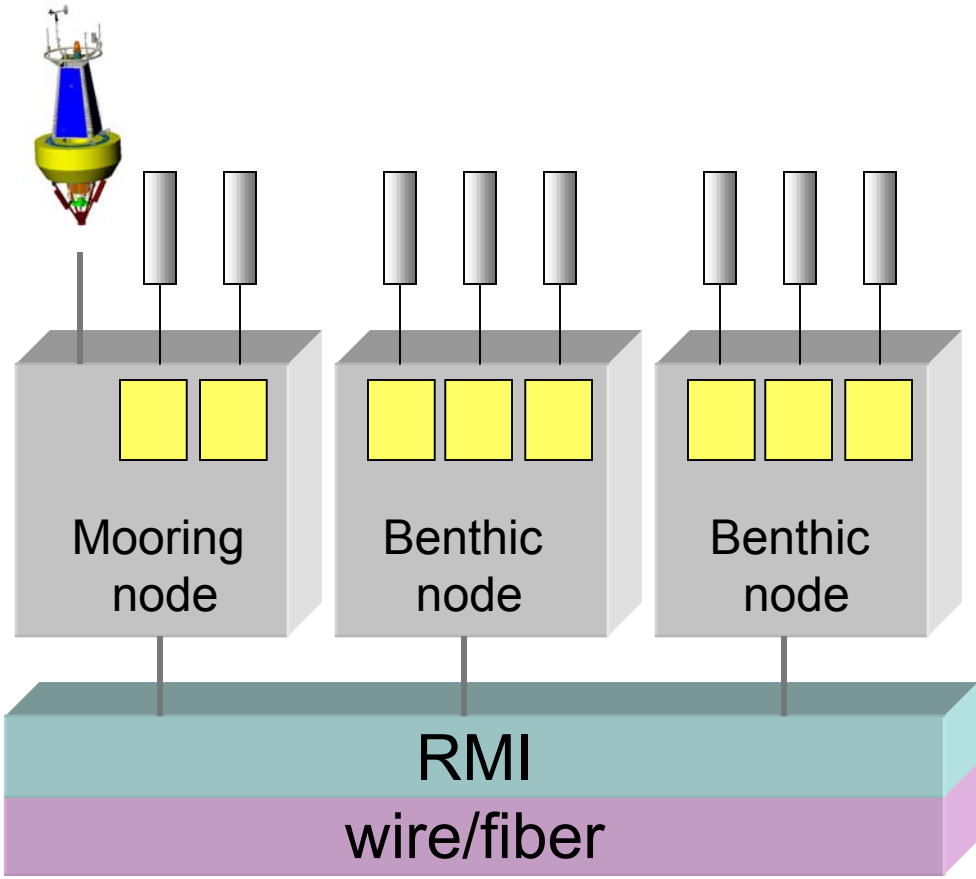
- We will use *Java RMI-IIOP* for object distribution
  - TCP/IP based protocol

Distributed object architecture





Shore network



Moored network

MBARI

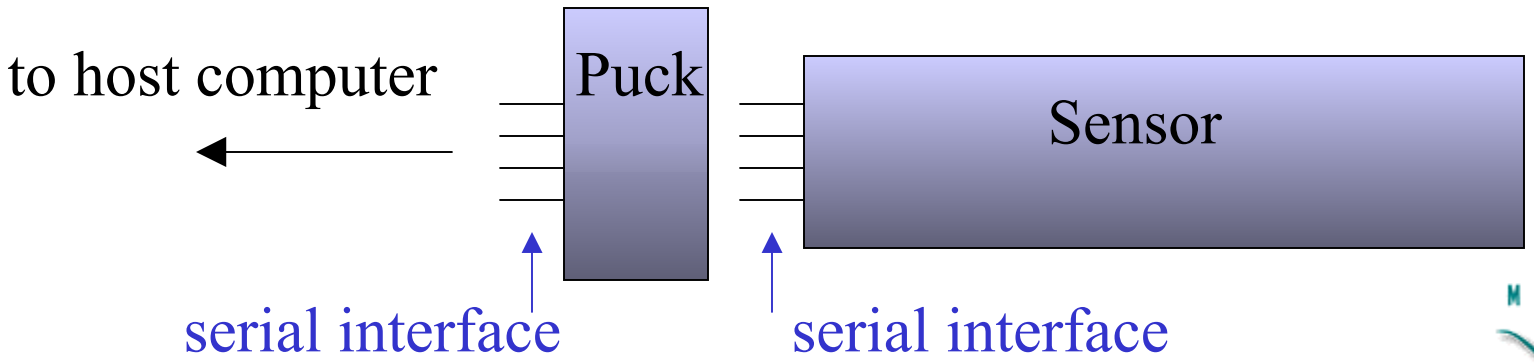
Plug and work

- Physically plug device into node's serial port

- Device and node cooperate to *automatically* install device service and calibration files, modify node configuration, and run the service

Sensor puck concept

- A puck is closely coupled to specific sensor, always travels with its sensor
- Puck contains sensor service code, metadata
- Host retrieves sensor information from puck when the sensor/puck is plugged in



MBARI

Straw procedures

- Operational “use cases” for system
- Stimulate discussion of use cases, functional requirements

System integration/configuration

- Kent Headley

Key Differences Between OASIS and MOOS Configuration

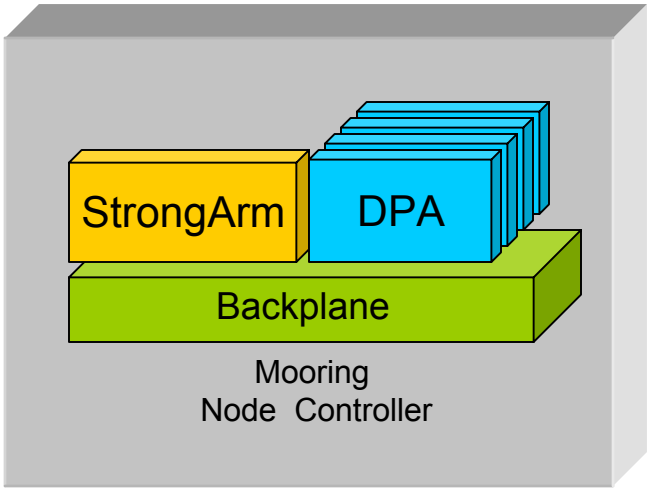
- MOOS is more complicated...
  - New capabilities create new administrative tasks
  - Integration with other platforms and systems introduces new complexity
  - New components and interfaces have a learning curve (we need feedback to reduce the slope)
  - Driver developers need to be aware of more complicated system issues
  - And Yet...

Key Differences Between OASIS and MOOS Configuration

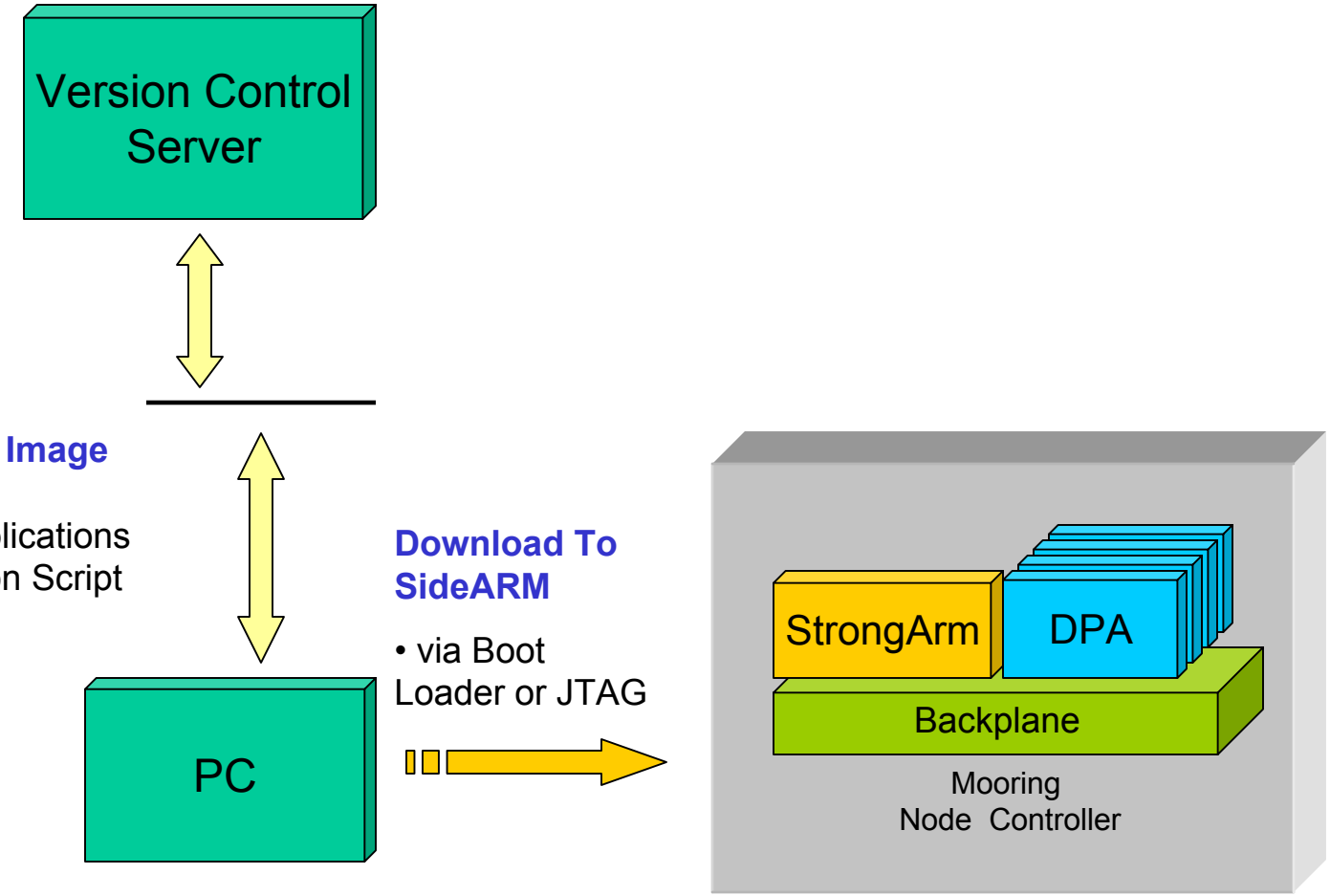
- MOOS is simpler...
  - Hardware design provides improved visibility and control
  - MOOS is resource-rich: less tweaking to make things work right
  - MOOS will provide greater fault tolerance and better diagnostic tools (with your help)
  - Plug and Work makes it easier to manage configuration
  - Administration tools use familiar, standard interafaces: Linux, Java
  - Your input will shape the interfaces you will use most often for operating and maintaining the system

MOOS Configuration
Single Node Hardware Configuration

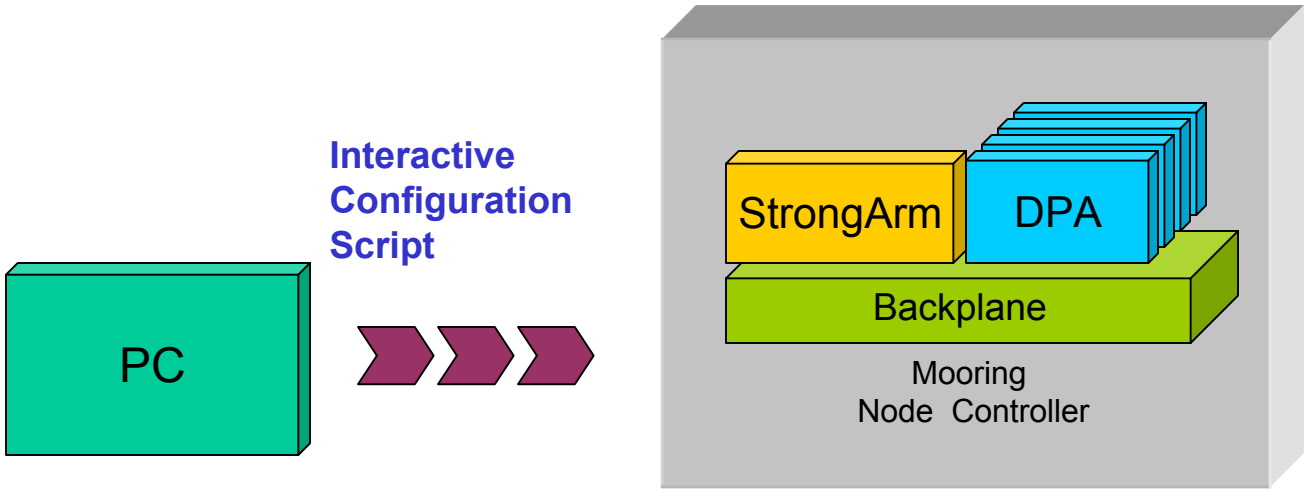
StrongArm and DPA Firmware Loaded



MOOS Configuration Loading Software



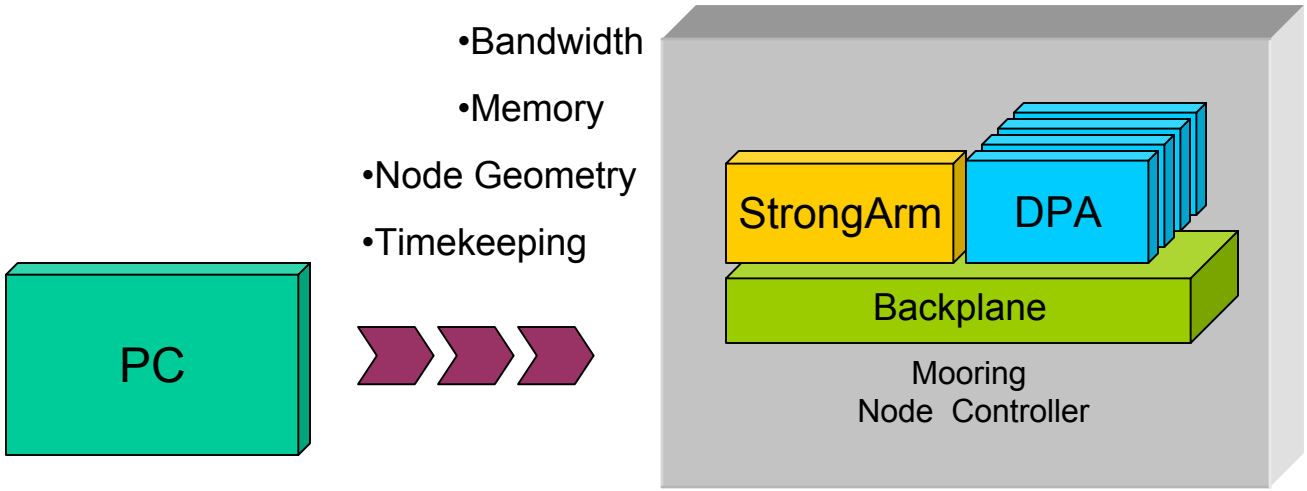
MOOS Configuration Node Setup



MOOS Configuration Node Setup

Interactive Set Up

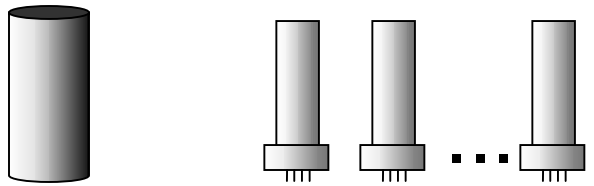
- Node ID
- Communications
- Network Settings
- Sampling Parameters
- Resource Management
  - Power
  - Storage
  - Bandwidth
  - Memory
- Node Geometry
- Timekeeping

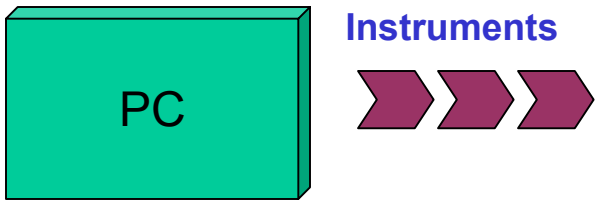


MOOS Configuration
Add Instrumentation

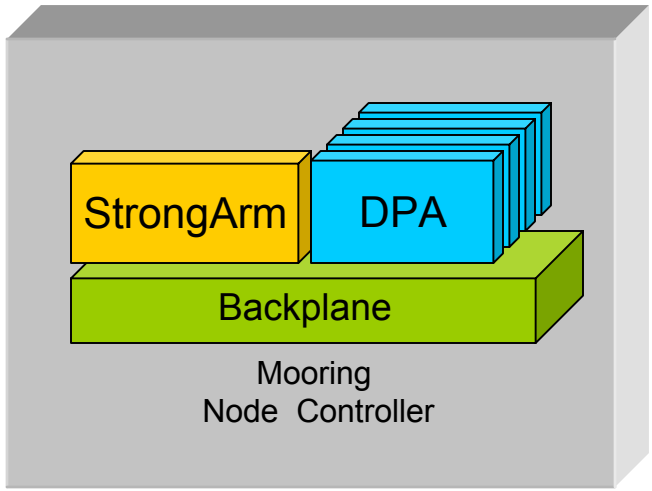
Telecom
- Globalstar
- Radio

Instruments and Pucks
- Pucks Pre-configured





Run Script to Probe for Instruments

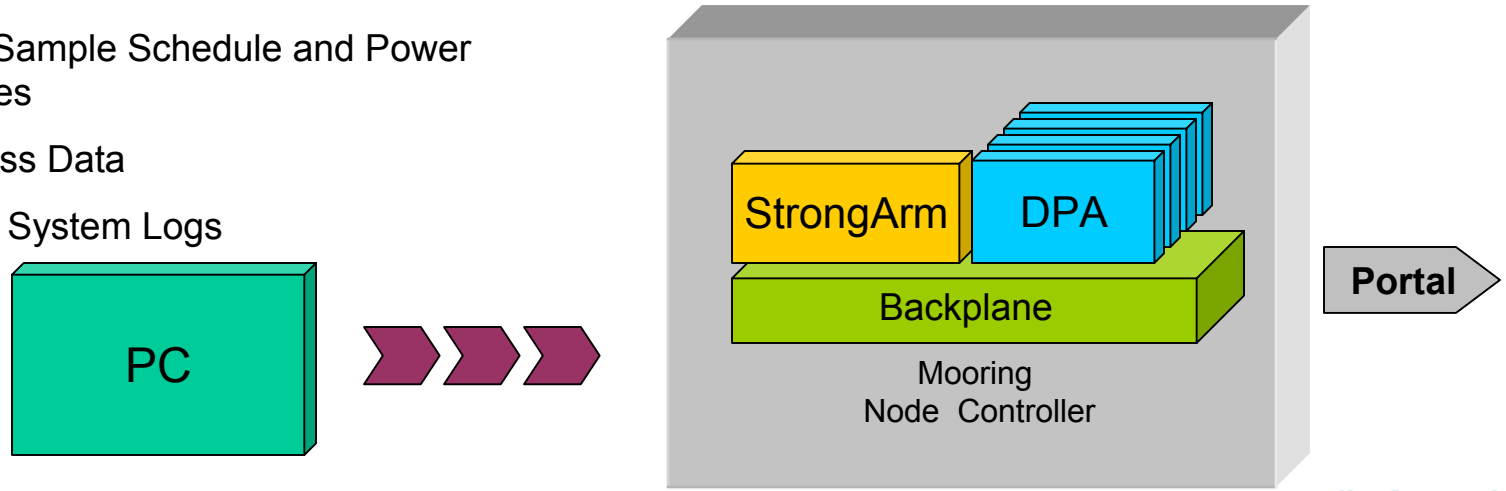


MOOS Configuration Integration And Test

Command Line Tools Provided to Verify Operation

- Enable/Disable Drivers
- Driver Status
- Console to Instrument
- Run Driver (sync, decoupled from schedule adjustment)
- Send driver comms and data to terminal
- Edit Sample Schedule and Power Policies
- Access Data
- View System Logs

Test Portal, SSDS Provide Access to Test Data for QC



MBARI

MOOS Configuration Installation

Procedures

- Initiate graceful shutdown via script
- Remove System Power
- Move and mount instruments and controller
- Apply System Power

MOOS Configuration Deployment

Procedures

- Monitor system while enroute via RF or cabled serial console. Disable drivers as needed
- Use RF console to monitor system during deployment (e.g., while attaching T-String pods, etc.)
- Make inter-node connections
- Restart any stopped drivers

MOOS Configuration Maintenance

Procedures

- Shutdown affected drivers while enroute via RF serial console
- While on site, use RF or serial cabled console to monitor system while performing maintenance procedures
- Instrument swapping procedures covered separately
- Once all instruments are ready, probe for new instruments (if new or different) or restart drivers

MOOS Configuration
What's missing from this picture?

• What scenarios (use cases)
would you most like to develop further?

•What have we left out?

•How would you change/add/remove
from what has been proposed?

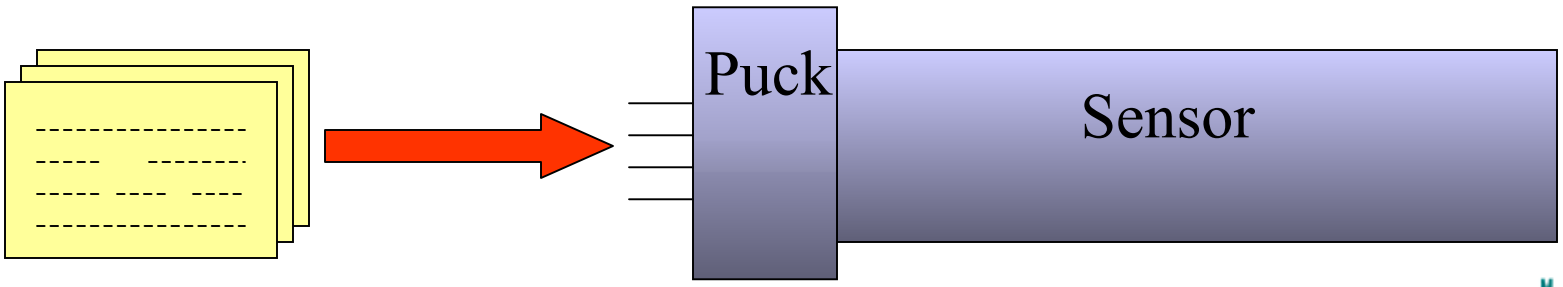
•How would this system be to maintain if
there were 50 nodes? 100?

•What suggestions do you have for the user
interface?

Sensor configuration

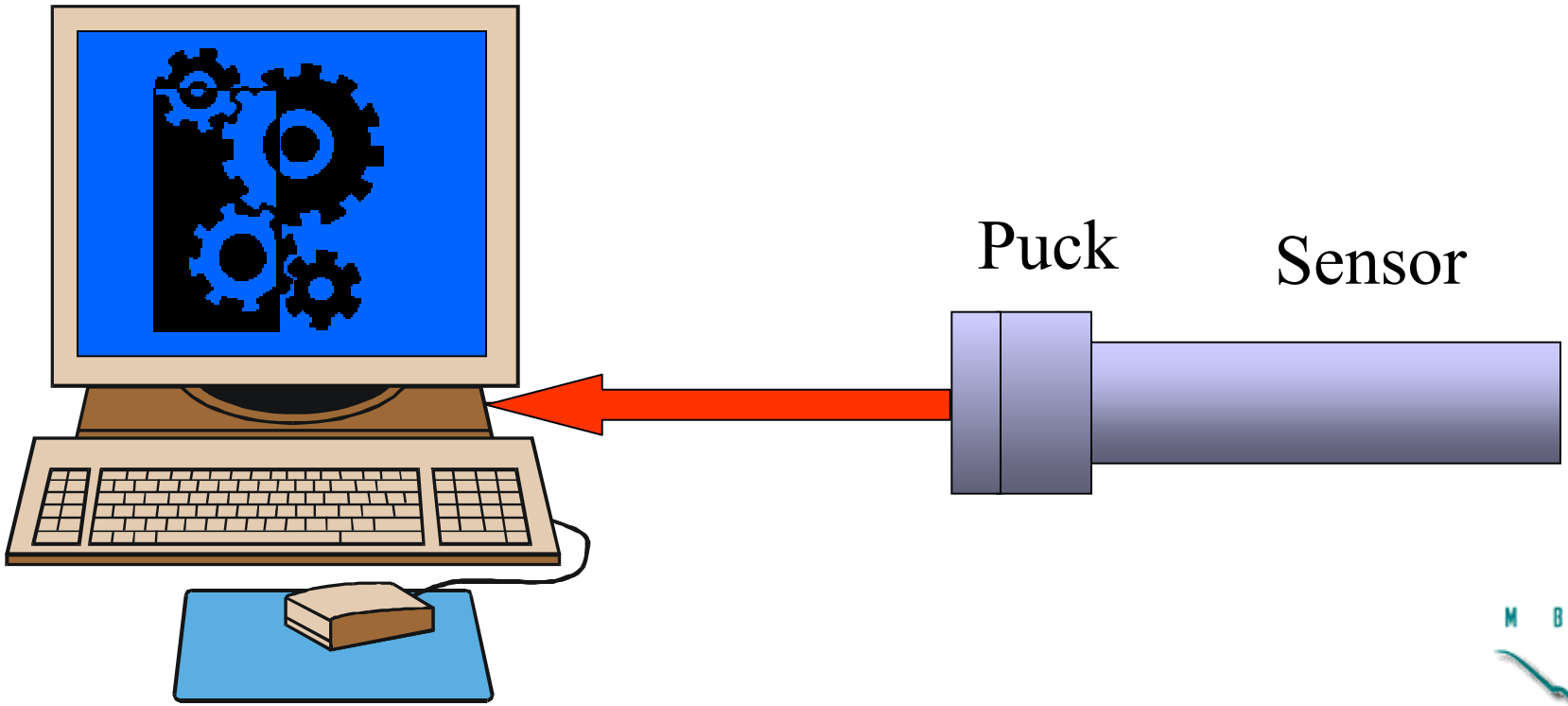
Sensor Configuration

- Attach a puck to the sensor
- Fill out a metadata template
- Create the puck image using the appropriate driver
- Write the puck image to the puck

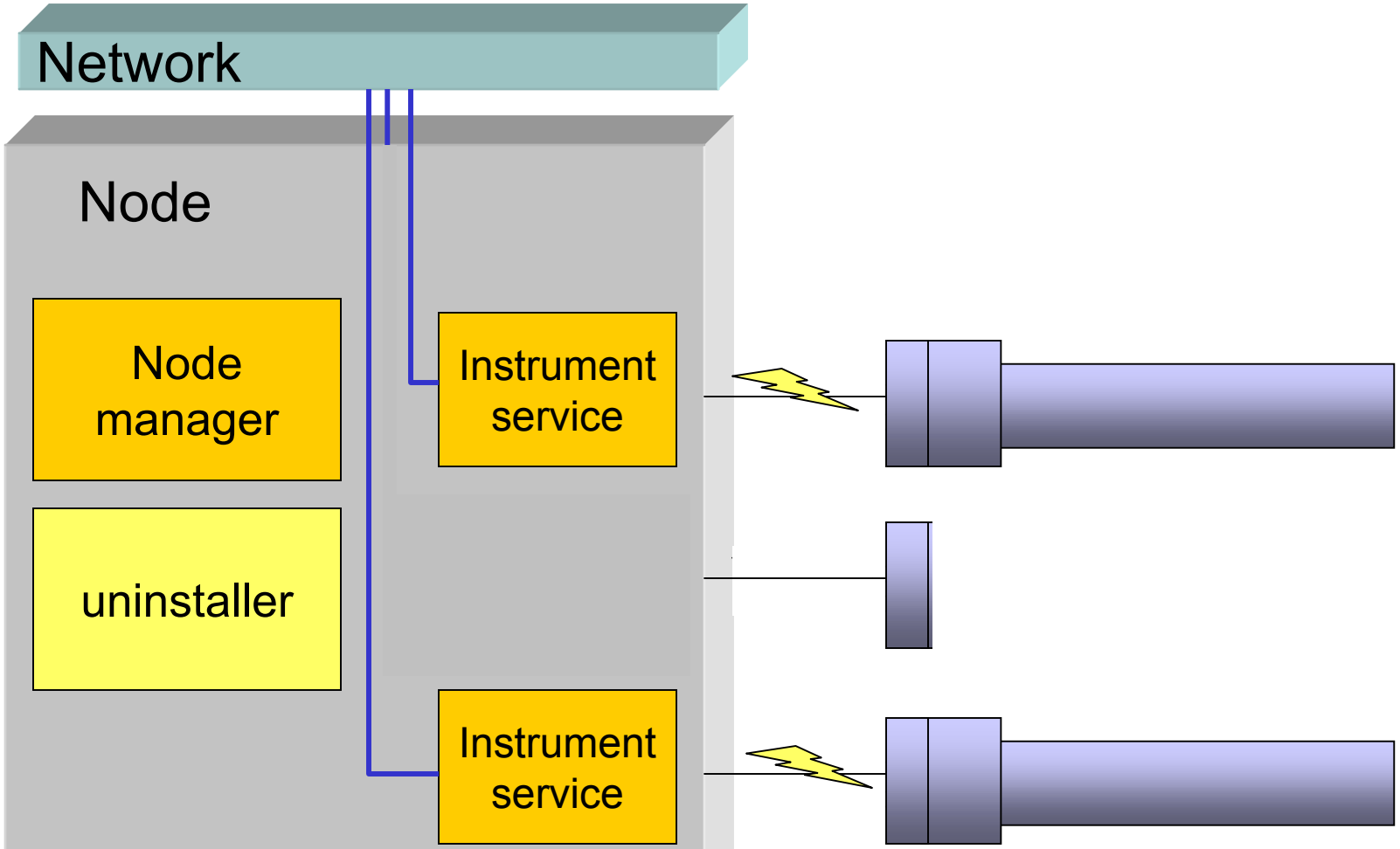


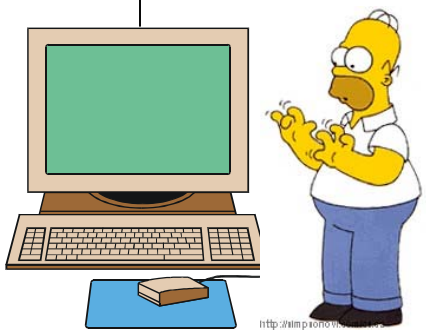
Sensor configuration

- Optionally the driver can be tested on the configuration system



Instrument swap use case



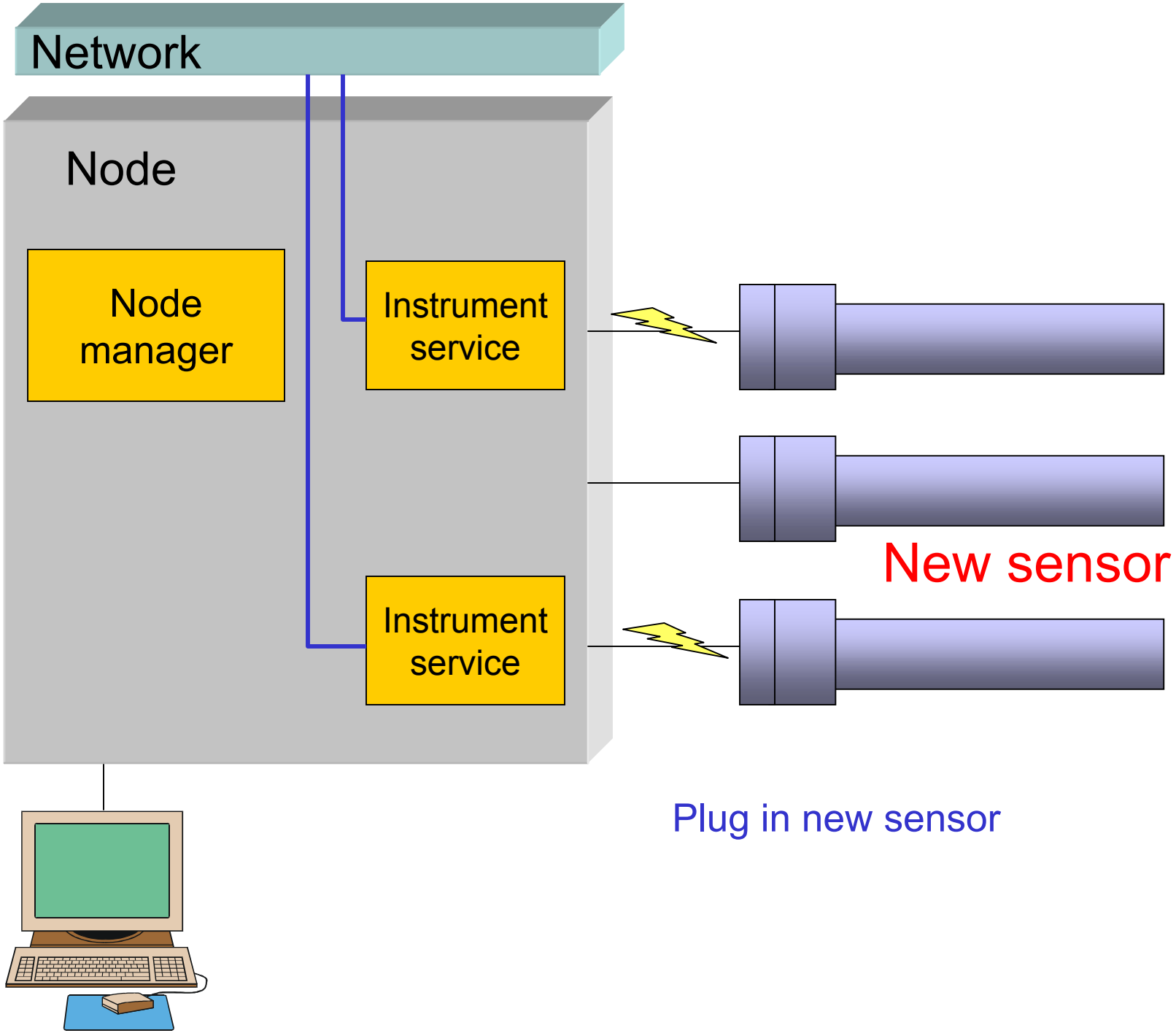


Operator runs *uninstaller* on port:

Uninstall disconnects port power, shuts down service

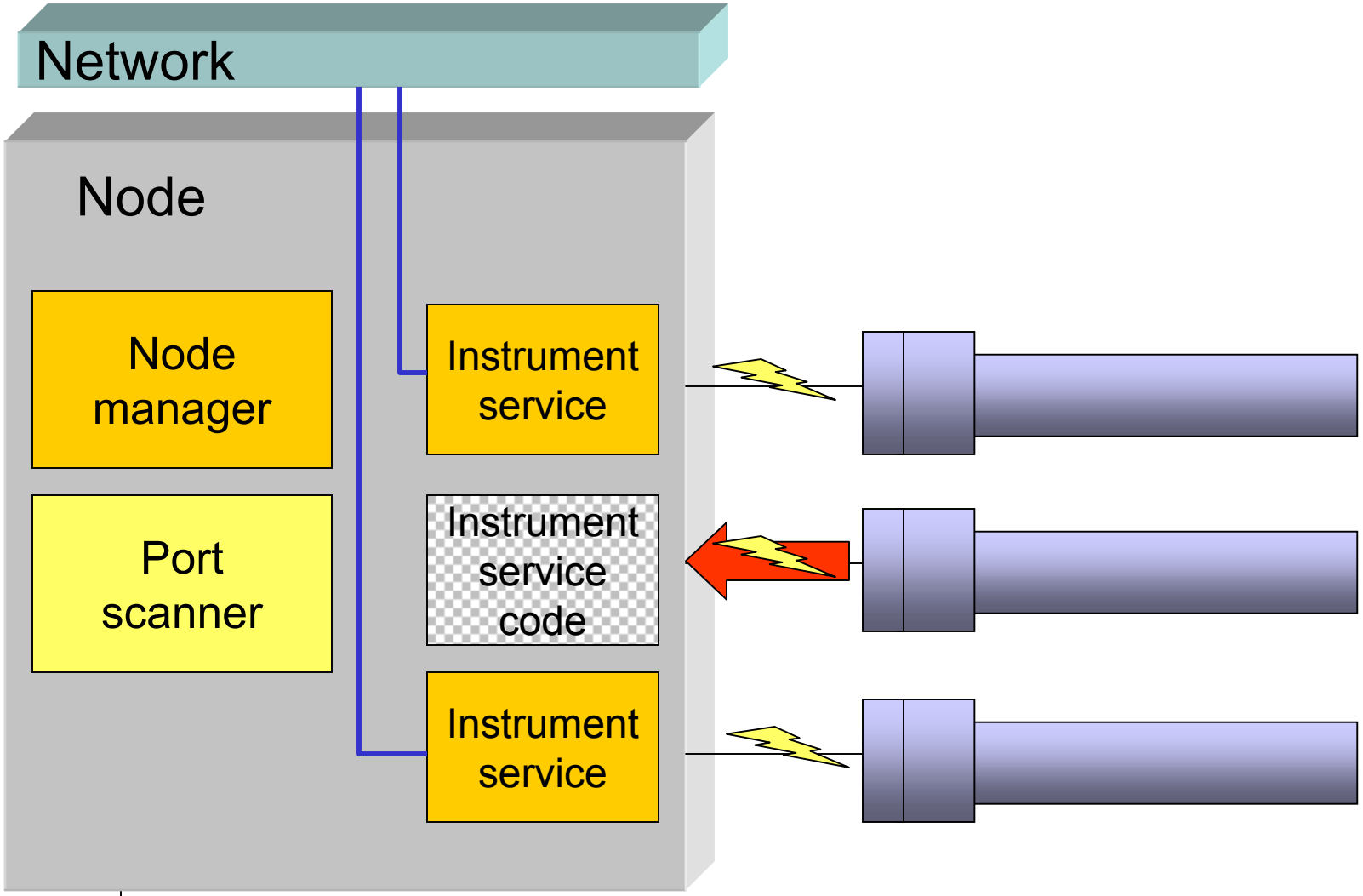
Safe to unplug instrument

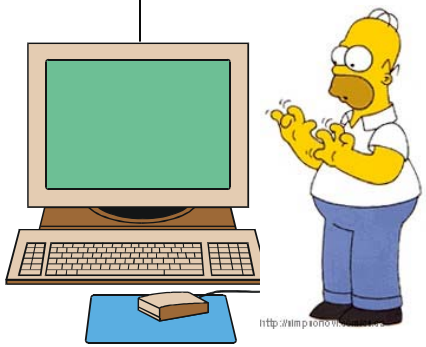
MBARI



Plug in new sensor

MBARI



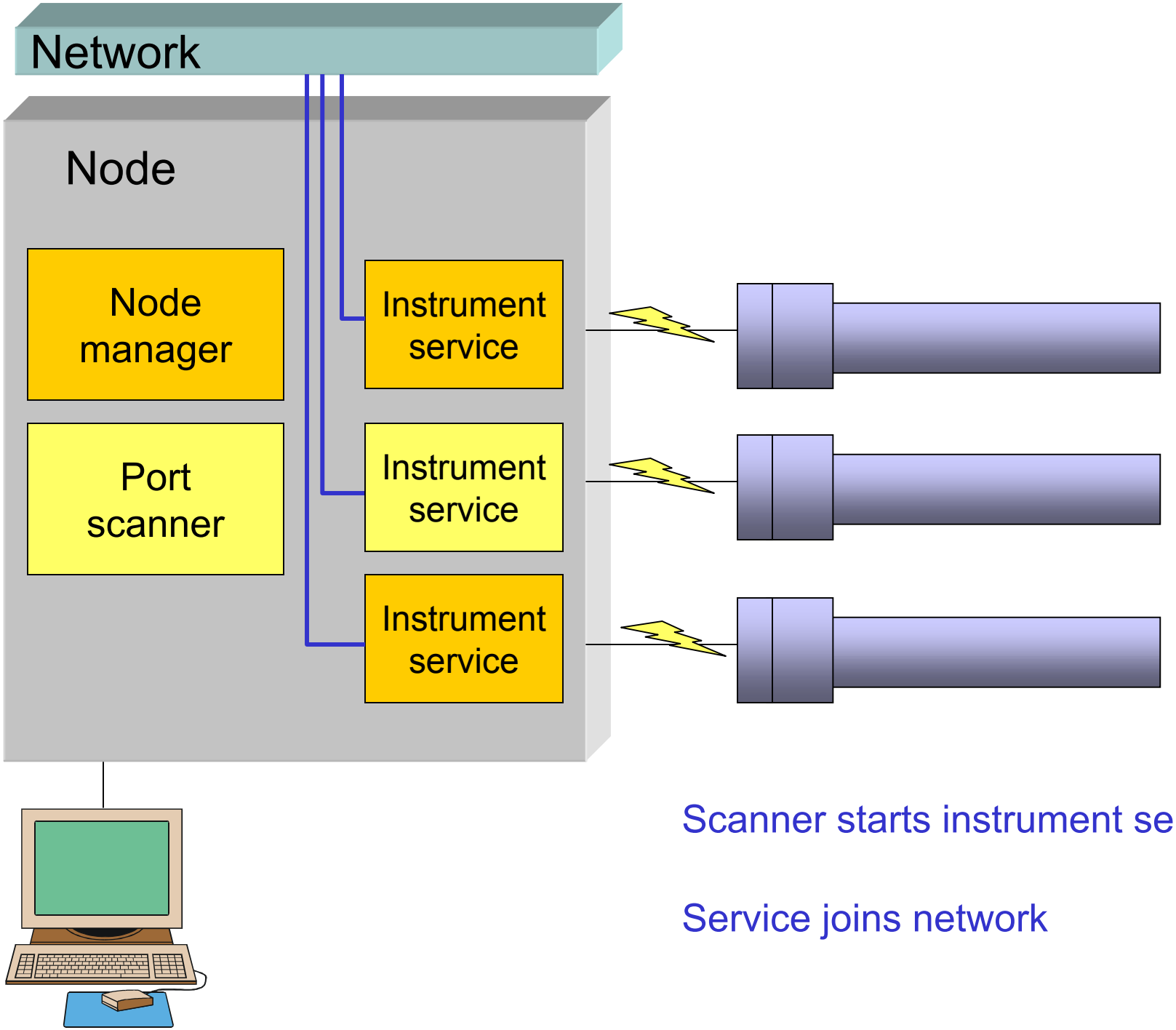


Operator runs *portscanner*:

Scanner applies power to port

Scanner extracts information (incl. service code) from puck

MBARI



Scanner starts instrument service

Service joins network

MBARI