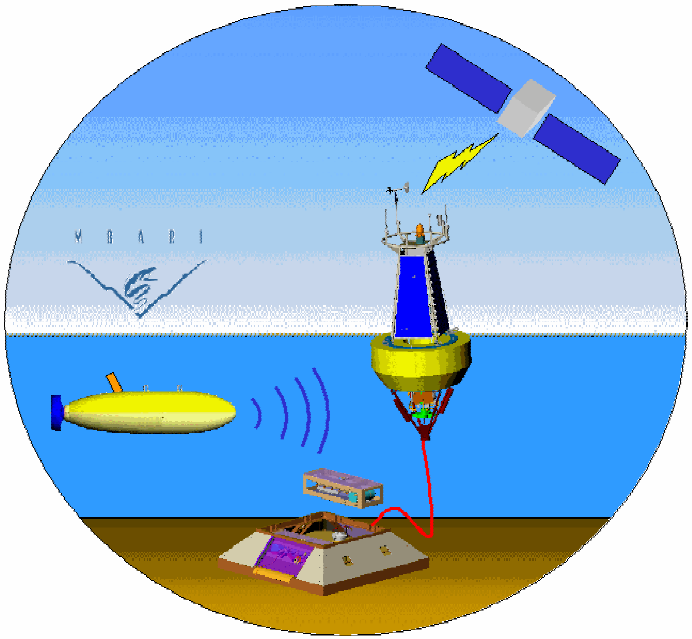
Instrument pucks



Michael Risi

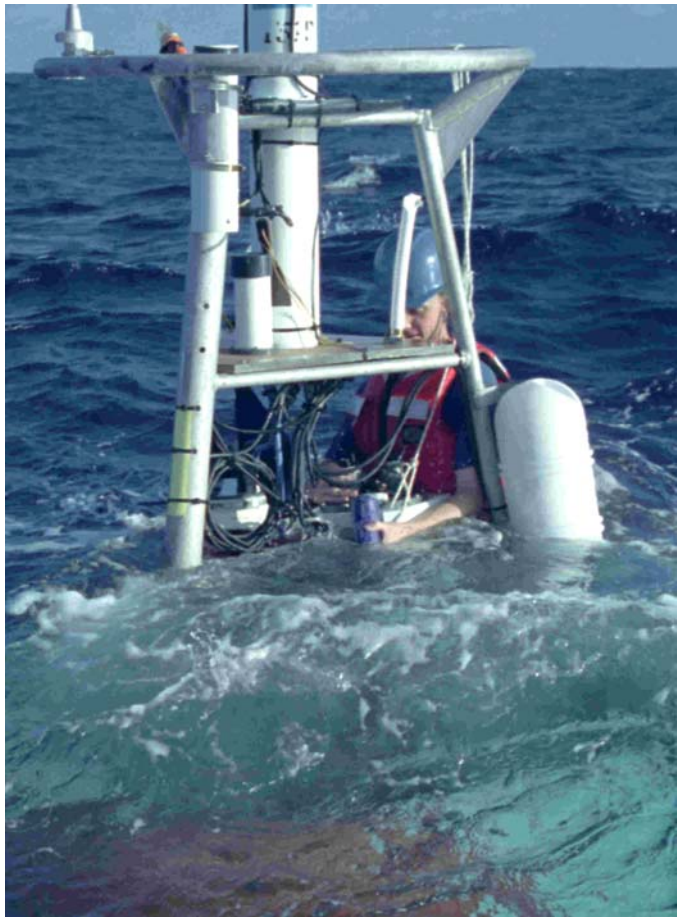
SIAM design review

November 17, 2003

Instrument pucks

- Pucks and *Plug-and-Work*
- The MBARI puck prototype
- Puck software interface
- Pucks in practice (A Puck's Tale)
- Embedding a puck within an instrument

The configuration problem



Pete Strutton, 1998, Equatorial Pacific

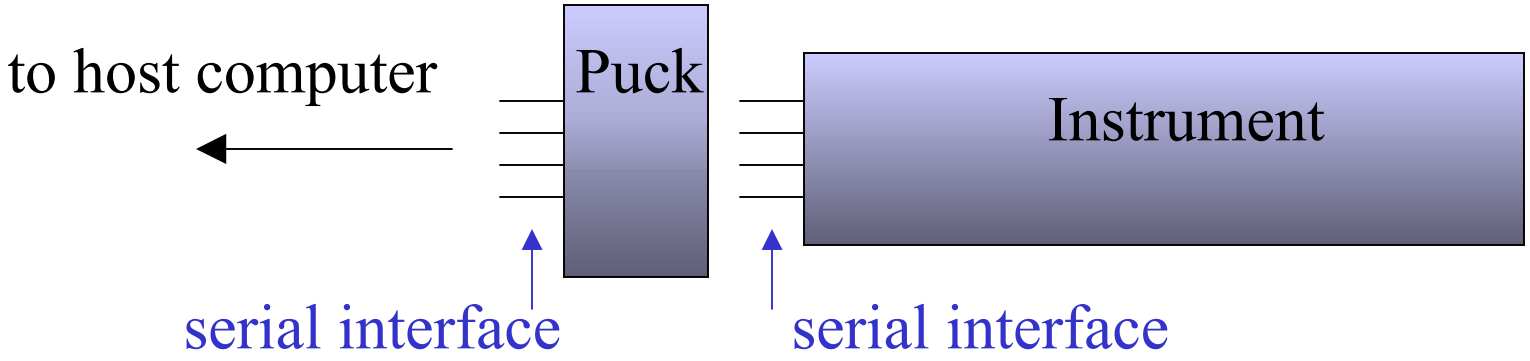
- Many steps required to install a device:
  - Plug device into host port
  - Install device software, configuration files, metadata
  - Modify host's configuration file (port #, baud rate, etc)
  - Associate metadata with new data stream
- Manual process is time-consuming, tedious, and error-prone
- Does not scale well

MBARI

The Plug-and-Work Solution...

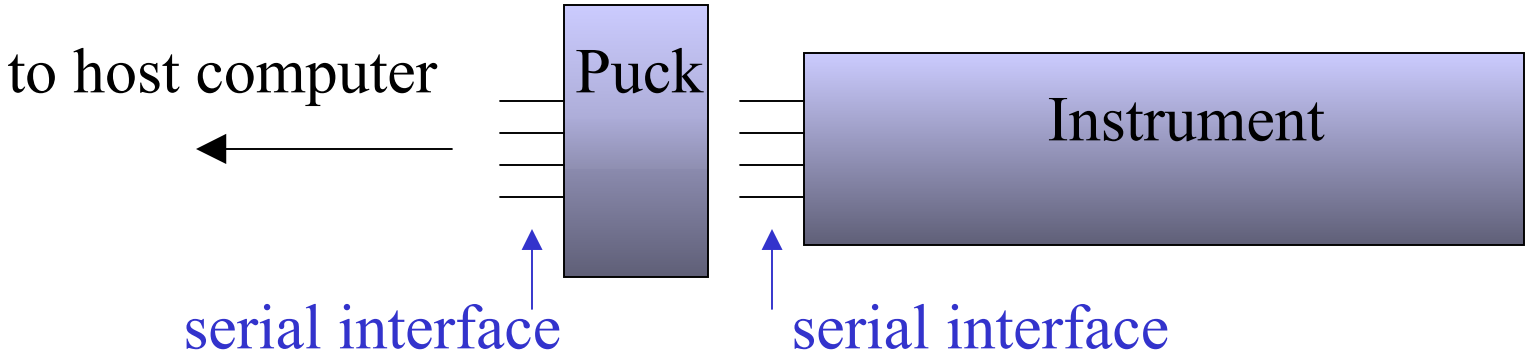
- Physically plug device into host serial port
- On user command, instrument service code and metadata are *automatically* retrieved from… “the instrument” itself!
- Metadata are *automatically* updated and inserted into the archiving data stream on startup and when state changes occur

Instrument puck concept



- Simple, small storage device
- Puck is closely coupled to a specific instrument, always travels with its instrument
- Stores instrument-related information (e.g. unique ID, instrument service code, other metadata)

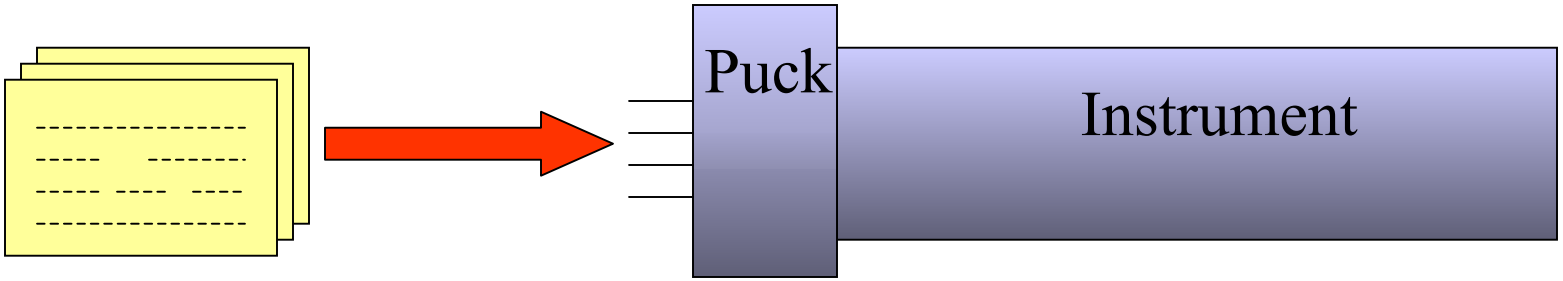
Instrument puck concept

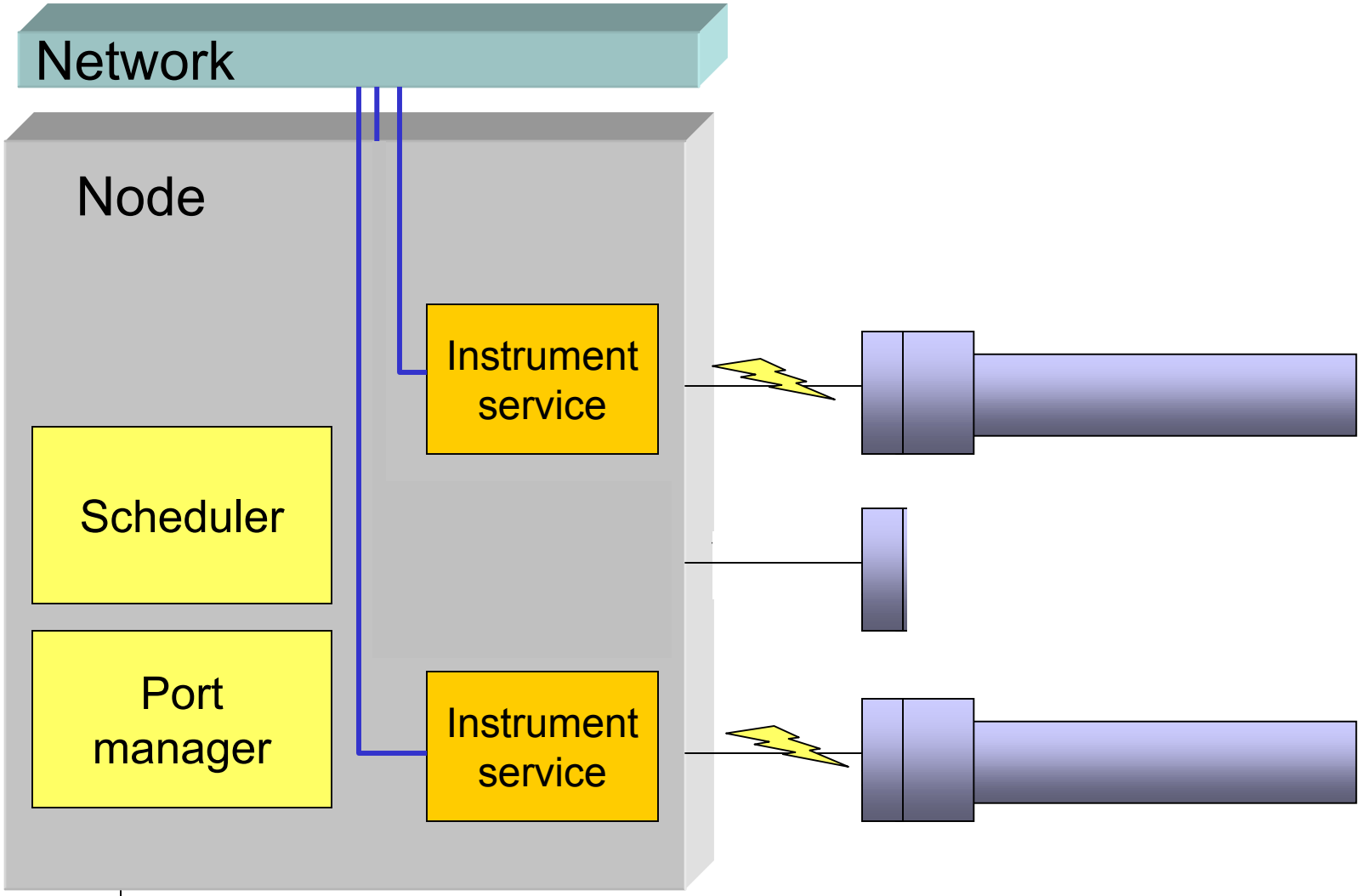


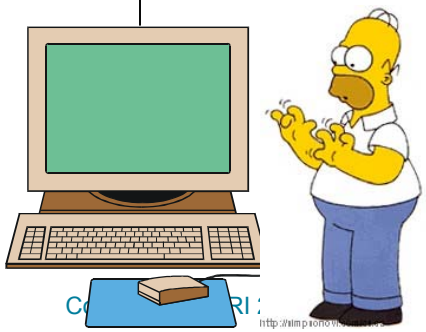
- Instrument information is retrieved by host when puck and instrument are plugged into host
- Host uses information appropriately
  - e.g. puck could store instrument driver code which is executed on host (*plug-and-work*)

Pre-deployment puck configuration for MOOS

- Attach puck to instrument
- Fill out metadata template
- Select appropriate instrument driver code
- Write metadata and driver bytes to puck





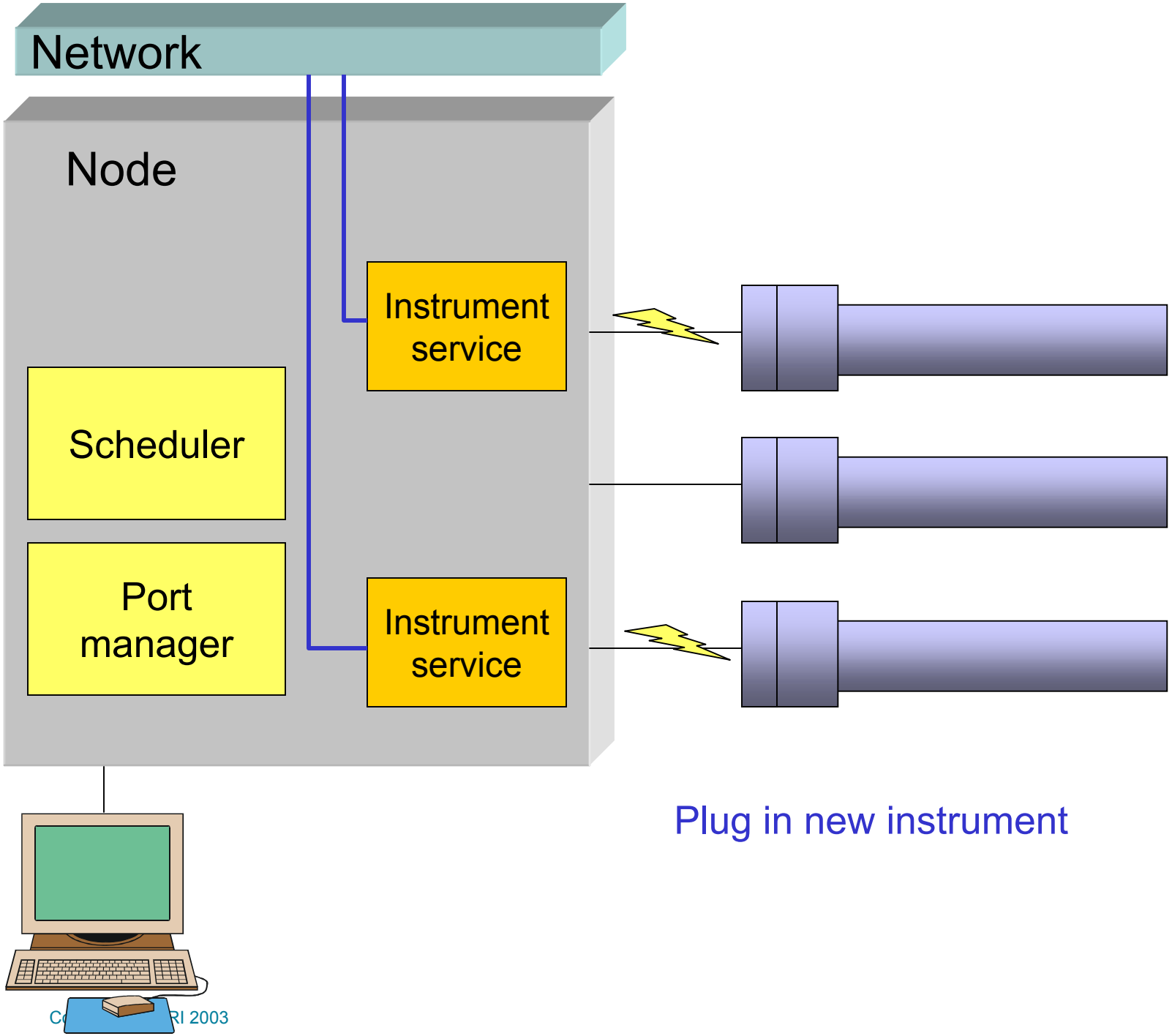


Operator runs *shutdownPort*:

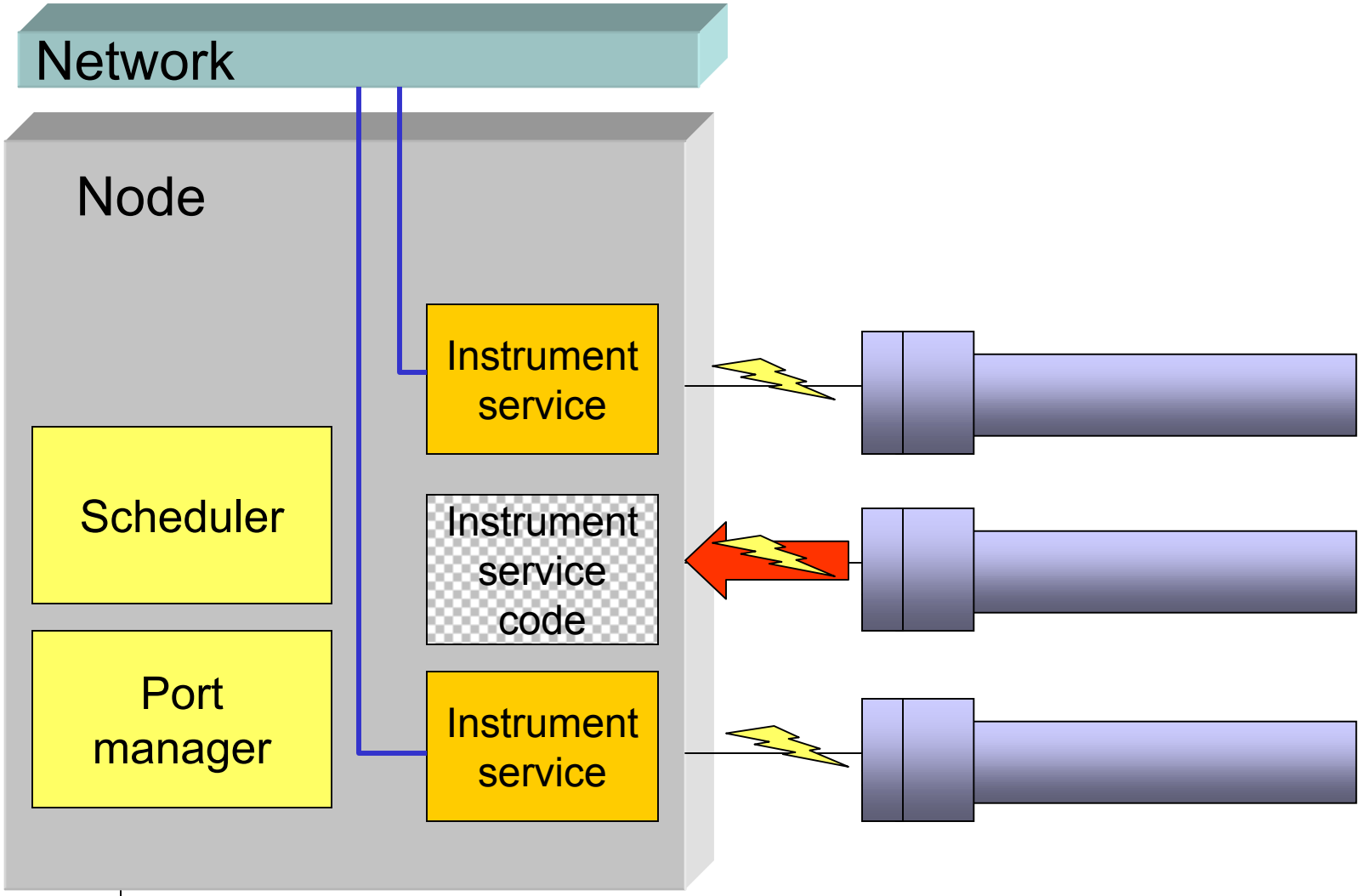
Port manager disconnects port power, shuts down service

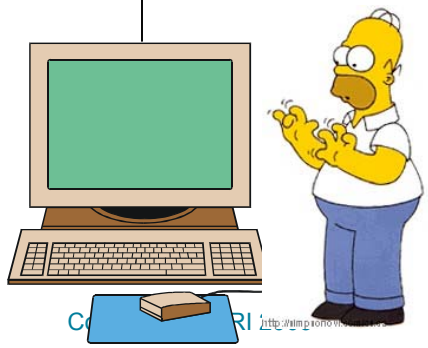
Safe to unplug instrument

MBARI



Plug in new instrument



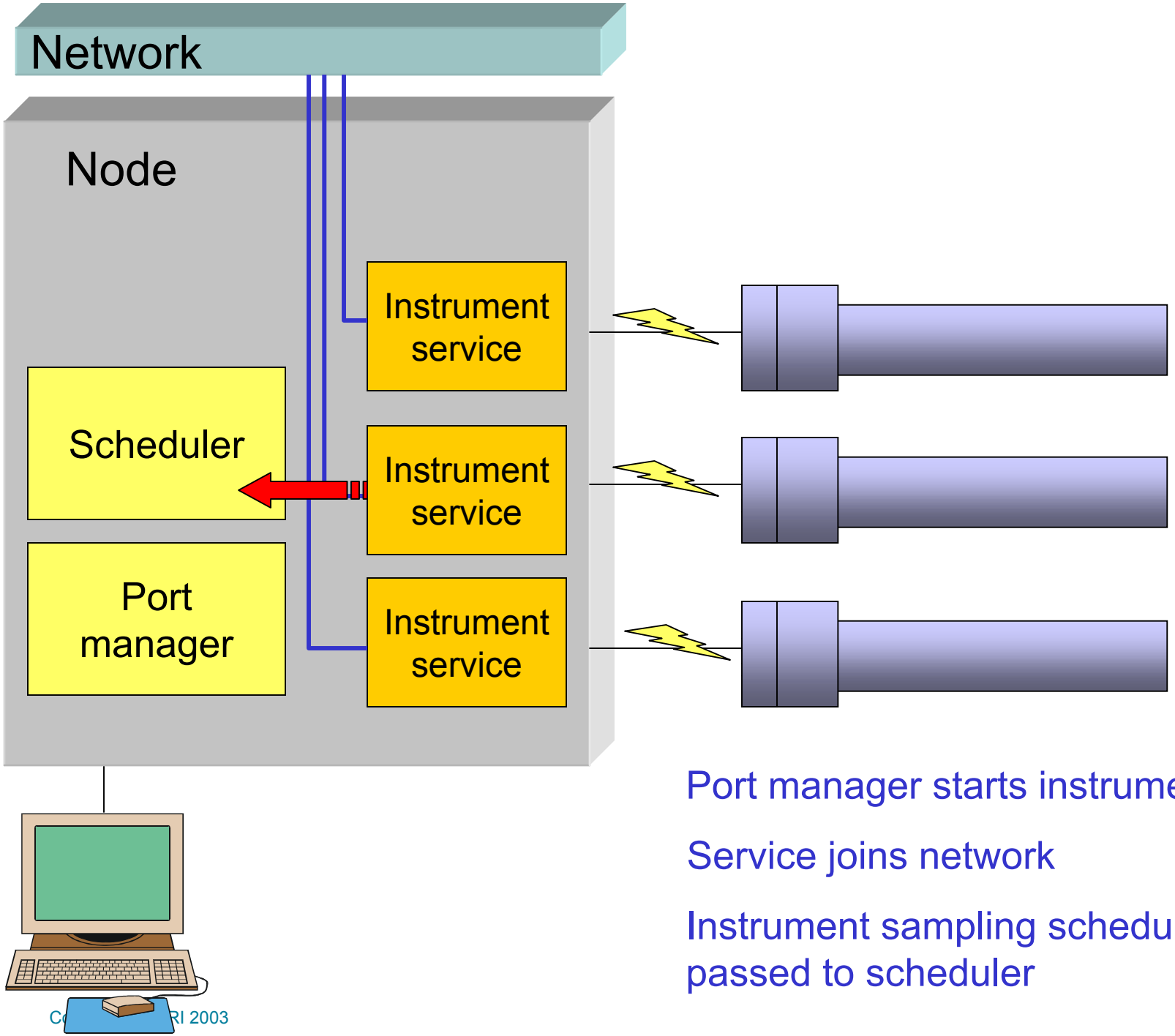


Operator runs *scanPort*

Port manager applies power to port

Port manager extracts service code, properties, etc from puck

MBARI



Port manager starts instrument service

Service joins network

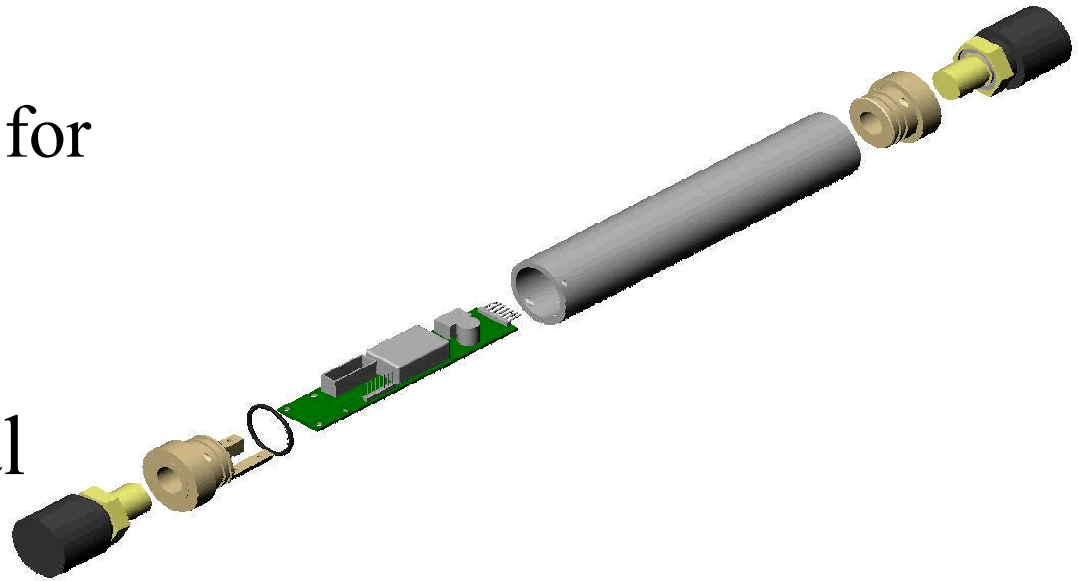
Instrument sampling schedule passed to scheduler

MBARI puck prototype

- Standalone puck that can be attached to any serial instrument
- Isolated communications and puck power
- One Megabyte persistent memory store
- Implements the puck software interface

What's inside this puck?

- 7.3728 MHz MSP430 microprocessor
  - 48 KByte flash for puck firmware
  - 2 KByte RAM
- MAX3160 serial transceiver
- 1 MByte SPI Flash storage



MBARI

MBARI puck prototype features

- Why Isolated communications and power?
  - Allows full compliance with MOOS Mooring electrical interface
  - Puck concept does not require this isolation
  - A smaller cheaper non-isolated implementation is planned for 2004

MBARI puck prototype features

- The MAX3160 serial transceiver
  - A low-power serial transceiver that supports RS-232 and RS-485
  - RS-485 firmware not implemented on MBARI puck prototype (2004?)
  - Puck concept does not require RS-485

MBARI puck prototype features

- How much memory does the puck require?
  - The MOOS/SIAM project selected one Megabyte as a starting point
  - In practice much less than this is currently being used

Puck software interface

- Discovering a puck connected to a platform
- The puck software interface
- What is *sensor mode*?

Discovery of puck

- Host applies power to port and prompts puck at 9600 8N1
- If puck present, host will extract puck contents
- If puck not present, host will power down port and isolate from system

Puck command summary

- RM - Read from the pucks memory
- WM - Write to the pucks memory
- FL - End a writing session
- ER - Erase the pucks memory
- GA - Get the internal address pointer
- SA - Set the internal address pointer
- SZ - Return pucks memory size
- SM - Put the puck into sensor mode
- SB - Set the baud rate of the puck
- VB - See if another baud rate is supported
- VR - Get the pucks version string

Sensor Mode

- When the MBARI puck enters *sensor mode*
  - communications lines are switched to the attached instrument
  - puck microprocessor goes into low-power sleep
- In an embedded puck the *sensor mode* command may not serve any real function

Pucks in practice

- What goes into a MOOS puck
- How is it stored on the puck
- How is it retrieved from the puck

MOOS puck memory contents

- Puck header
  - Service name
  - Unique ID
  - Optional instrument parameters
- Puck jar file payload
  - service.properties
  - Java byte code

MOOS puck memory map

0x0000 0x0003	Organization (4 bytes)
0x0004 0x0007	Type\Version (4 bytes)
0x0008 0x0017	UUID (16 bytes)
0x0018 0x0023	payload start address (4 bytes) payload end address (4 bytes) payload checksum crc32 (4 bytes)
0x0024	sensor data sheet -driver name (256 bytes) -ISI ID (8 bytes) -max current required (4 bytes) -etc. The format will be determined by the version and newer version wills always be backward compatible with older versions.
start address	driver and meta-data in jar file

Puck jar file

- Allows multiple files to be stored in a single binary image
- Java support for creating, reading, and writing jar files
- Incorporates compression to make more efficient use of memory and decrease payload retrieval time

Typical puck jar file contents

- Contents of the Metsys.jar file

```
service.properties
moos/devices/metsys/Metsys.class
moos/devices/metsys/Metsys_Stub.class
moos/devices/metsys/Metsys_Skel.class
```

Instrument Jar File Sizes

Instrument Service	Jar file (bytes)
KVH Compass	11959
Garmin GPS	17166
Gashound Subsystem	13642
Environmental Processor	11888
MBARI Metsys	16067
Satlantic ISUS	16559
Seabird CTD	12983
Seabird Temperature String	17431
WetLabs ECO Flourometer	14361
WHOI Asimet	16604
RDI Workhorse ADCP	16162

MOOS puck host utilities

- writePuck - store payload on puck
- readPuck - retrieve payload from puck
- catPuck - quickly identify puck's contents by scanning puck header only

Embedding the puck

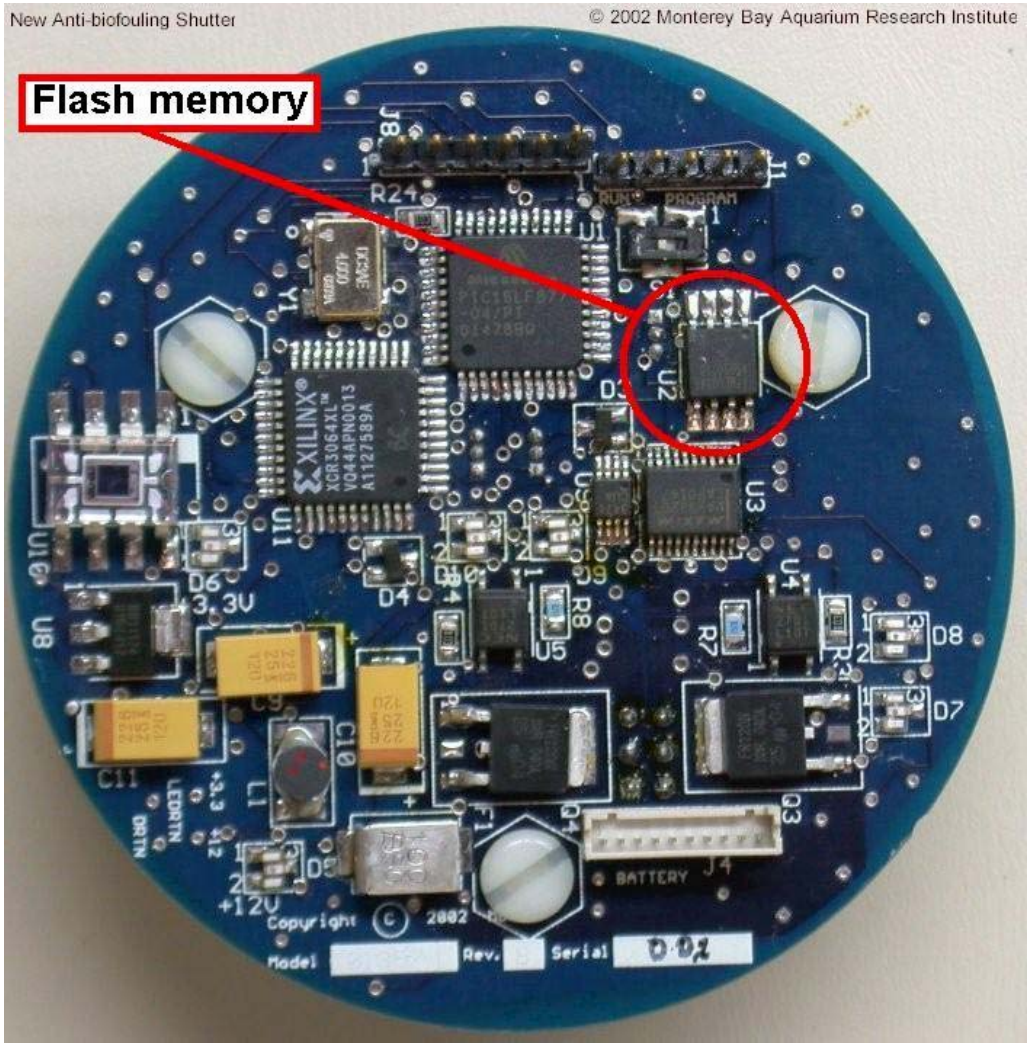
- An embedded puck example: *serial shutter*
- How to embed a puck into an existing instrument

MBARI Serial Shutter

- Anti-biofouling shutter for optical instruments
- Controlled by host via serial interface
- Only additional hardware required was single flash memory chip
- Puck software interface is simple; easy to implement on embedded processors

The MBARI Serial Shutter

MBARI Serial Shutter



What about existing instruments?

- An existing instrument can be made puck compatible if its...
  - serial port can support 9600 8N1 serial settings on power-up
  - processor is capable of implementing the puck software interface
  - storage is sufficient for puck payload

Discussion