Managing Sensor Network Configuration and Metadata in Ocean Observatories Using Instrument Pucks

Kent L. HEADLEY, Dan DAVIS, Duane EDGINGTON, Lance McBRIDE , Thomas C. O'REILLY, Michael RISI

Monterey Bay Aquarium Research Institute

KEYWORDS: Sensor Networks, Metadata, Puck, Automatic Configuration

ABSTRACT

Cabled observatories, such as MARS or the regional scale cabled observatory system planned for in the NSF Ocean Observatories Initiative (OOI), consist of many deployed instruments that communicate with human operators and shore-side data repositories. In addition, these deployed devices may actually communicate with one another, facilitating capabilities such as autonomous event response. These potentially complex interactions between multiple entities - human and machine - require that knowledge of the system configuration be available to participants. Users of instrument data require information - metadata - about the sensor that generated the data. Software that coordinates and controls instruments requires access to the software interfaces of those devices. "Manual configuration" has been used on small-scale systems, but in a network consisting of hundreds or thousands of instruments, the configuration challenge becomes critical. We propose to address the problem through automation of the configuration process, which will be achieved at several levels.

Automated configuration will simplify the system operator's task of building and maintaining the observatory network. We describe a small, low-powered information storage device that we call a "instrument puck". When plugged into a suitable computer (lab workstation, deployed observing node), information can be written to or read from the puck. While an instrument is being prepared for initial integration into the observatory, a technician "loads" a puck with information necessary to configure the instrument within the observatory, and then physically attaches the puck to its instrument. Thereafter the attached puck always travels with its instrument, no matter where it is being installed in the observing network.

The information loaded into the puck encompasses whatever is necessary to enable automatic configuration and system integration of the instrument when it is plugged into the observatory network, and any other information required by observatory policies. This information may include structured descriptions of the instrument's sensor and data characteristics (metadata). The information can also include actual software code that is retrieved from the puck and executed by an observatory node when the device is plugged in; this code could implement distributed instrument control and data retrieval interfaces, allowing network-wide access to the instrument functionality. We believe the puck concept to be a powerful one; a given instrument puck is configured just once, enabling automatic configuration of its instrument no matter where it is installed on the network thereafter.

We also describe mechanisms by which an instrument and its puck can be "discovered" by the observatory network when the devices are plugged in. Several approaches are explored, with varying degrees of automation. We evaluate these approaches with special consideration to electrical and safety aspects of the undersea environment. Information and results from our prototyping efforts will also be presented.

1.Introduction

Cabled observatories, such as MARS or the regional scale cabled observatory system planned for in the NSF Ocean Observatories Initiative (OOI), [7] will be comprised of many instruments deployed on a network. Software and hardware elements will enable bi-directional communication between instruments and other observatory subsystems, such as shore-based data repositories and human operators. Moreover, instruments may actually communicate with each other, facilitating capabilities such as autonomous event response. These potentially complex interactions between multiple networked entities – human and machine – can be facilitated by installing software and other information onto the system at the time the physical device is plugged in. We refer to this installation process as *configuration*. Software and information which is installed during configuration includes the following:

Interface software – enables software components to communicate with the instrument across a network. For example, a graphical user interface or data acquisition application communicates with the instrument via its software interface. The interface software executes on a processor that may be physically distant from the device itself.

Instrument driver software – implements the operations requested by calls to the interface. The driver software usually executes on a processor that is physically very close to the device itself.

Instrument metadata – identifies and describes an instrument in machine-readable or human-readable format. Metadata may include device manufacturer, serial number, calibration coefficients, etc. Raw instrument data has little scientific value in the absence of its metadata.

In small-scale observing systems, instruments are usually configured "manually"; when the instrument is physically plugged into the network, a human operator also executes the necessary computer commands to install software and metadata. However, manual configuration does not scale well to a large ocean observatory. In a networked system, the operator might need to configure more than one machine. Manual configuration is typically a tedious, repetitive, and hence error-prone process in the best of circumstances (the marine environment can at times be very challenging to work in from the physiological standpoint, which results in more opportunity for error). In an ocean observatory consisting of hundreds or thousands of instruments, the configuration problem becomes critical.

We are developing automated configuration mechanisms that we believe will greatly simplify the tasks of building and maintaining an ocean observatory. Our design objective is what

we refer to as *plug and work* functionality. When an instrument is physically plugged into the system, its configuration information should be automatically installed onto the system, with very minimal manual intervention required.

In particular, we are developing a low-cost, low-power device called an *instrument puck* (Figure 1). A puck is attached ahead of time to any instrument that is to be deployed in the observatory. The puck's function is to store software and metadata for its instrument, and to make that information available to the system when the instrument is plugged into the observatory. The puck has two communication ports (Figure 2). One port attaches permanently to an instrument's communication port, and the puck-instrument pair always travels together. The puck's other port provides an interface to a *host computer*, which can be either a workstation or a deployed observatory node processor. Before deployment, the puck and its attached instrument are plugged into a workstation and a technician "loads" the puck with appropriate software and metadata. When an instrument is deployed, its puck is plugged into an observatory node, and the node retrieves the instrument software and metadata from the puck. The node then switches the puck into *pass through mode* and communicates directly with the instrument. We believe the puck concept to be a powerful one: a given instrument puck can be configured just once, enabling automatic configuration of its instrument no matter where it is installed on the network thereafter.

The current version of the puck is designed for instruments that communicate with an observatory host computer via serial RS-232 or RS-485, as these are the most common interfaces for oceanographic instruments today. There is no reason why pucks could not be designed for other physical interfaces as well.

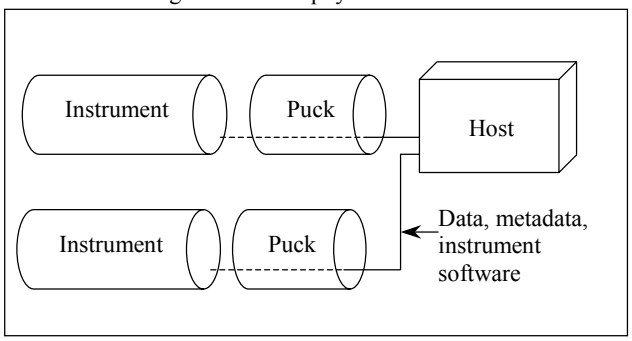

FIGURE 1
Instrument Puck Concept

2.Instrument Puck Architecture

The instrument puck hardware consists of a microcontroller, a flash memory device, a signal relay to switch the serial communications to the attached instrument, a serial transceiver, and an isolated power supply (Figure 2). The microcontroller used for the puck is the Texas Instruments MSP430. The MSP430 has a built-in UART necessary for communicating with the host computer and an SPI port that is used to interface with the flash memory device. The MSP430 also has general purpose I/O necessary for such things as switching the signal relay and putting the serial transceiver into different modes of operation. The MSP430 is an extremely low power device, thereby adding only trivially to the power requirements of the instrument when the puck is attached.

The puck's flash memory device is a 2 MB Atmel DataFlash®. The puck's flash memory can be read or rewritten by the host computer via the puck's serial interface. All memory accesses, whether reading or writing, must take place while the puck is in "puck mode". The memory is used to store instrument software (in the form of Java™ classes) and metadata. Both the classes and the metadata are stored in a single binary file known as a Java™ Archive (JAR). Since the JAR is a single binary image it can easily be stored in the puck's memory device. JAR files incorporate compression, allowing the puck memory to be used more efficiently as well as decreasing the amount of time required by the host computer to read the information from the puck. Currently our instrument software sizes range from 7 KB to 12 KB when compressed leaving ample room for metadata and any other information we choose to include in the JAR.

The communications switch is handled with a signal relay although a solid state switch is also being investigated. Serial communications can be switched directly to the attached instrument once the host computer has retrieved the metadata and instrument software. When serial communications have been switched from the puck's microcontroller to the attached serial instrument, the puck is in what we refer to as *pass through mode*. No level shifting or processing of the serial stream occurs once the puck is put into *pass through mode*; the host computer is ostensibly attached directly to the instrument.

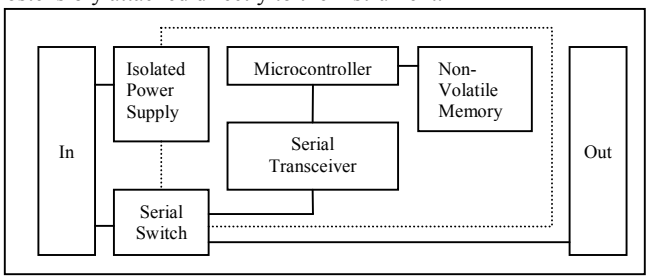

FIGURE 2
Instrument Puck Block Diagram

A multi-protocol serial transceiver was selected for the puck allowing it to be used in an RS-232, RS-422, or RS-485 environment. The MAX3160 serial transceiver can be switched into RS-232 or RS-485/RS-422 mode by the MSP430 allowing the puck to be attached to a variety of serial instruments. The MAX3160 can be put into a low power shutdown mode when the puck is put into *pass through mode*.

Power for the puck is supplied via an isolated DC to DC converter. The isolation is necessary to ensure good communications with the instrument. Assuming the puck is used with a high current serial device, especially over some distance, the "ground" at the source and the "ground" at the device will be at different potentials due to the high current and the supply wire resistance. An isolated ground for communications eliminates any problem related to this effect.

Instrument installation and discovery is achieved as follows:

- The instrument and attached puck are plugged into the platform
- When the port is powered up by the host, the puck enters *puck mode* and waits for queries from the host computer
- When the host computer queries the port it discovers a puck and proceeds to upload the instrument software and metadata from the puck's flash memory
- After the instrument software and metadata have been uploaded to the host computer the puck is put into *pass through mode*
- The instrument software is executed on the host computer and communicates directly with both the serial instrument attached to the puck and the observing network

It is worth noting that the puck does not execute instrument software directly and need not understand the contents of its flash memory. The more capable host computer executes the instrument

software and the metadata are delivered to a data archiving system.

3.Application To an Ocean Observatory

3-1.Instrument Puck in the MOOS/SIAM Architecture

The instrument puck is a prominent feature in the architecture of observatory systems being developed at MBARI[1]. The MBARI Ocean Observing System (MOOS) project is aimed at developing large-scale ocean observatory facilities. Key features of MOOS include its great degree of connectivity between its many platforms (ROVs, AUVs, benthic instrumentation nodes, and moorings) and the highly integrated approach to data collection and archiving.[1, 3]

The MOOS project is divided into several major components; among them are Software Infrastructure and Applications for MOOS (SIAM), MOOS Mooring Controller (MMC, a networked instrument controller), and Shore Side Data System (SSDS, a highly integrated data archiving system).

SIAM comprises the software "glue" that binds the MOOS system components together. SIAM components are responsible for many critical system functions, including

- Basic data collection
- Delivery of data to the Shore Side Data System (SSDS, another of the seven main MOOS subsystems)
- Management of metadata
- Providing application and programmatic interfaces to enable access to the MOOS system by human users and software clients across the network

SIAM is also involved in low-level system functions such as

- Power management
- Communications
- Time keeping

In addition, SIAM facilitates high-level functions involving data integration and synthesis:

- Event detection
- Adaptive sampling.

The SIAM architecture is intended to make it very scalable with respect to available network bandwidth and available power. It is designed to operate on low-power, narrow bandwidth networks, e.g., solar powered moorings or battery-powered drifters with packet radio/satellite telemetry. SIAM nodes can also take advantage of high-power, high-speed platforms (e.g., cabled observatories).

3-2.SIAM Components

SIAM software is written in Java™, making it possible to develop and use code on different platforms. SIAM is logically divided into two major software subsystems, Deployed and Operations. The Deployed system consists of the software components installed in the remote location, for example in a mooring controller or contained in an instrument puck, and in general is responsible for data collection. The Operations software components, mostly on land-based or shipboard computers, provide connectivity to the Deployed system, delivering data to SSDS and providing interfaces to science and operations users and software agents.

Currently, the Deployed subsystem is being hosted on the MOOS Mooring Controller (MMC). The MMC consists of a SideARM (a StrongARM/Linux processor board) and up to six Dual Port Adapters (DPA), each of which provides communication and power switching for two instruments. A third board, the Backplane, connects the SideARM and DPAs, providing the power and data buses connecting them. Each MMC node, capable of controlling 12 instruments, has RS-485 and Ethernet connections, allowing them to be networked.

The Deployed subsystem software a Node Application, composed of a number of software "managers" that take care of the low-level system tasks described earlier. The shore-side Operations software includes a Portal and a collection of applications to access and maintain the system. The Portal has a number of important functions, bridging the Deployed and Operations subsystems. All communication with the Deployed subsystem takes place via the Portal, which can manage potentially limited bandwidth, arbitrate between various software agents requesting access to the nodes, and providing store-and-forward control services.

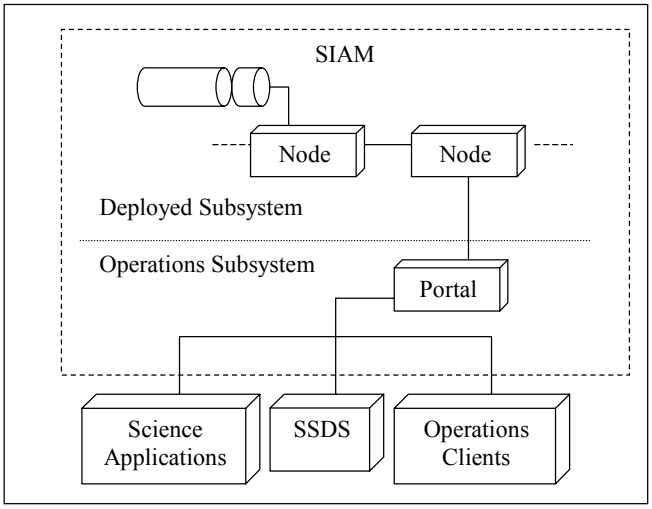

FIGURE 3
SIAM Block Diagram

3-3.Interactions Between Instrument Pucks and SIAM

The instrument puck and the SIAM Node Application address the complex issues of dynamic system configuration and metadata management in the field. Instruments connected to the MOOS are mated to an instrument puck containing the instrument metadata. This metadata includes a unique MOOS system ID, default data acquisition schedule, serial number, manufacturer, calibration coefficients, and a description of the data produced by the instrument. The puck also contains the interface and driver software that provides services related to the instrument (data acquisition services, interfaces, network access).

When an instrument and its puck are plugged into an MMC port, the Node Application reads the contents of the puck, retrieves and installs the service software, and sends the instrument's metadata to the SSDS. This self-configuration process exemplifies *plug and work* capability: when instruments are plugged into the MMC, they are automatically configured by the system, which also dynamically detects and propagates changes to configuration and other metadata throughout the system.

The instrument's metadata are sent to the SSDS, signaling the arrival of the instrument's data stream and establishing its context. As changes to this context occur, they are reflected in the SSDS as new metadata are injected into the data stream by the Node

Application or by the instrument services. If appropriate, the metadata contained in the instrument puck may also be altered (though the metadata contained in the puck is for the most part immutable, or at least changes very infrequently).

Thus, the puck provides the critical binding of metadata to the instrument: the instrument and it's metadata are never separated, avoiding problems that arise in systems in which metadata is centrally managed. In such systems, it is possible for the metadata to become unsynchronized with the instrument, so that it no longer reflects the true state of the instrument or the context in which it is operating. A system in which metadata must be managed manually is prone to subtle errors and does not scale well on large systems with complex configurations. The MOOS/SIAM/puck architecture, in contrast, reduces the incidence of such problems by coupling metadata with its source and providing mechanisms for detecting and propagating changes to metadata.

To implement *plug and work* instrument swapping, it is necessary to manually shutdown the instrument service associated with instrument. Once an instrument has been installed on the port, the Node Application must be manually instructed to scan the port to discover the instrument. These manual steps are necessary for several reasons: Firstly, power must be removed before potentially exposing the instrument's connector to seawater. Secondly, in order to maintain a hardware interface compatible with common instruments, it was undesirable to use common signal pins to support automatic discovery operations. Finally, to keep the number of conductors (and hence enclosure penetrations) to a minimum and maintain compatibility with existing cables, it was not desirable to add additional conductors for automatic discovery. Other approaches (magnetic switches, in-band signaling) were considered and rejected on the basis of their complexity and potentially poor reliability.

3-4.Results And Conclusions

A proof-of-concept puck design was demonstrated in 2001. This design carried a small instrument service and GUI, which was downloaded across a network and run by a preliminary version of the Node Application on a PC, since SIAM hardware components were not available at that time.

In August of 2002, the first release of the MMC hardware was deployed on an existing mooring in the Monterey Bay [1,3]. A prototype version of the SIAM software was used to collect data from a CTD, a power meter and several other sensors. The data were returned each hour to a Portal on shore using a radio modem. This initial deployment of SIAM did not include power management, the ability to participate in a network of other nodes, or *plug and work* capability.

In December 2002, the MMC tested the MOOS Test Mooring (MTM). The MTM deployment was made to test the MOOS mooring structure, power system and novel cable design during winter weather conditions[1, 3]. The MTM release of SIAM featured several new instrument services and a satellite telemetry link, which delivered data to the prototype SSDS via the Portal. The MTM deployment also used a single controller with a fixed configuration.

A version of *plug and work* capability, using JAR files on a file system in lieu of a puck, was recently demonstrated. Each virtual puck (JAR file) contained an instrument service, a sampling schedule and instrument metadata.

The demonstration consisted of a node running several instrument services running on an MMC node. Though all of the configurations were loaded by using a configuration file, one of the configuration entries referenced a symbolic link to a virtual puck (JAR). The demonstration proceeded as follows:

- One of the instrument services was shut down
- The instrument was removed and replaced with a different instrument
- The symbolic link was changed to point to a new virtual puck
- The Node Application was instructed to scan the instrument port, causing it to load the new puck contents
- The Node Application places instrument metadata indicating the change of configuration into the data stream
- The Scheduler loads the schedule provided by the instrument service and initiates data acquisition

Puck hardware is currently being implemented; in the near future, we expect to complete a puck hardware prototype and a revision of the SIAM software that will include *plug and work* support using the prototype puck.

ACKNOWLEDGMENTS

We gratefully acknowledge the MMC hardware team for their technical input on the puck architecture. We would also like to thank the MBARI Support Engineering, Observatory Support and Technical Support Groups for providing technical support, and to Doug Au and Keith Raybould for their management support.

REFERENCES

[1] M Chaffey and E Mellinger, "Communications and power to the seafloor: MBARI's ocean observing system mooring concept," in *Proceedings of the Oceans 2001 MTS/IEEE Conference*, in press, 2001

[2] T O'Reilly, D Edgington, D Davis, R Henthorn, M McCann, T Meese, W Radochonsky, M Risi, B Roman, R Schramm, " 'Smart Network' Infrastructure for the MBARI Ocean Observing System," in *Proceedings of the Oceans 2001 MTS/IEEE Conference*, in press, 2001

[3] T Meese, D Edgington, W Radochonsky, S Jensen, K Headley, "A new mooring controller platform: an evolution of the OASIS instrument controller toward a distributed ocean observing system," in *Proceedings of the Oceans 2001 MTS/IEEE Conference*, in press, 2001

[4] Sun Java RMI website, java.sun.com/products.jdk/rmi

[5] Jini website, www.jini.org

[6] Universal Plug and Play website, www.upnp.org

[7] OOI website, http://www.geo-prose.com/projects/ooi.html