A Research Prototype of a Networked Smart Sensor System

John C. Eidson and Stan P. Woods
Measurement Systems Department
Instruments and Photonics Laboratory

HPL-95-91
August, 1995

Keywords

smart sensors,

distributed systems,

distributed measurements

electronic data sheets

A research prototype of a networked smart sensor system is described.

The contributions of this system study are in three areas:

1) A network interface which defines a common data model, a set of operations, and the ability to customize the node operation in a consistent manner .
2) A transducer interface which supports the 'electronic data sheet' for transducers with voltage and digital outputs.
3) A set of node behaviors which simplify the installation and configuration of a group of nodes to perform a measurement or control application.

This paper was presented at Sensors East in Boston, May 1995.

Copyright ©1995 Hewlett-Packard Company

TABLE OF CONTENTS

I: INTRODUCTION.....2
II: BACKGROUND.....2
III: OVERVIEW OF DISTRIBUTED MEASUREMENTS.....3
A: Transducer-related properties of distributed measurement nodes..... 3
B: Measurement-related properties of distributed measurement nodes..... 4
C: System or application-related properties of distributed measurement nodes....4
D: Communication protocol issues..... 4
E: Data management issues..... 5
F: Control protocol and real-time issues..... 6
IV: PROTOTYPE SYSTEM.....7
A: Design objectives and specifications..... 7
B: General description of an application using the prototype system..... 8
C: Smart node architecture..... 9
D: Operational aspects of smart nodes..... 11
V: INTERFACE DEFINITIONS.....12
A: Transducer interface..... 12
B: Network interface..... 12
VI: EXPERIENCE USING THE PROTOTYPE SYSTEM.....13
A: Printed circuit board manufacturing..... 13
B: Laboratory ambient condition monitoring..... 13
C: Miscellaneous systems..... 13
D: Observations..... 14
VII: TOPICS FOR FUTURE RESEARCH.....14
VIII: CONCLUSIONS.....15
Appendix A: Detailed description of system models. 15
A: Network interface..... 15
B: Behavioral models..... 17
C: Transducer interface..... 18
REFERENCES..... 20

I. INTRODUCTION

This paper discusses the use of distributed systems technology in applications involving measurement and control, a subject of much recent discussion, see references [1],[2],[3]. As part of this investigation we designed and constructed a networked system of smart sensors and actuators. The goal was to evaluate distributed smart sensor systems with respect to:

1. ease of system configuration
2. ease of system expansion and modification
3. ease of data management
4. ease of application development

V. BACKGROUND

Traditional test and measurement applications have been based on the use of centralized control and data management. The usual implementation involves a central controller that directs the actions of instruments, sensors, and actuators; polls for any results; and manages the resulting data.

Today, distributed measurements usually refers to systems comprising one or more controllers each with one-to-one connections to instruments, sensors, and actuators. Supervisory control and data acquisition, SCADA, systems are a typical example of such a distributed measuring system. Other distributed systems are essentially autonomous measuring devices such as data loggers or electric meters. In this case the data is brought manually to a central location or it may be polled over a communication link.

In both of these previous cases, the essential determination of the system's behavior resides in the central controller. Communication is one-to-one between the central controller and each instrument, sensor, or actuator.

In this prototype system, a more general notion of distributed measurements is used. Specifically the nodes, i.e., the instruments, actuators, and sensors, determine the overall behavior of the system, rather than depending on a central controller for control. In addition, nodes communicate directly with each other or with groups of other nodes without any restriction to one-to-one communication links. In the rest of this article, "distributed measurements" refers to this more general notion. "Distributed measurement systems" as used in this paper, also includes systems with actuators and more general purpose nodes such as displays.

It is unlikely that distributed measurement technology will be the best solution for all applications any more than traditional centralized technology is best for all applications. In general, a mixture of technologies will be the optimum for a given application. However, this prototype system and article deal almost exclusively with distributed measurement technology.

VI. OVER VIEW OF DISTRIBUTED MEASUREMENTS

Distributed measurement systems generally require more "intelligence" to be built into the nodes than a traditional centralized system designed for the same overall task.

It is useful to consider three aspects of distributed node intelligence: transducer-related, measurement-related, and system or application-related intelligence. These aspects might be termed "levels of smartness."

The following sections discuss some properties of measurements that need to be considered when designing nodes for a distributed system. In general, the more of these properties that are managed by the node, the more useful the node will be as a general component in distributed measurement systems.

For example, consider the transformation of the voltage output of a thermocouple into temperature. If this is not handled in the node containing the thermocouple, then every other node in the system that makes use of the thermocouple output must either make this transformation itself or obtain the result from a node that does make the transformation. Either of these choices provides less system design flexibility than doing the transformation in the node containing the sensor.

A. Transducer-related properties of distributed measurement nodes

Transducer-related properties include:

1. Physical variable: temperature, stress, etc.
2. Form of transducer input or output: voltage, change in resistance, digital signal
3. Calibration: relationship of transducer output to sensible value, e.g., converting the value of the voltage from a thermocouple to the measured temperature
4. Identity: transducer serial number, description, etc.
5. Limits of use: maximum and minimum values, acceptable operating environments, stability of calibrations, repeatability, etc.

In a distributed system, these items will be handled at the nodes containing the sensor or actuator. Current instruments generally manage these properties, while most available sensors and actuators do not. Many of the "smart sensors" available today manage only a few of these properties. Examples of such existing smart sensors are:

1. the Analog Devices ADXL50 accelerometer, which has built-in signal conditioning and self-test
2. the Smartec SMT160-30 temperature sensor with built-in signal conditioning and digital interface
3. the Omega Engineering PX763 pressure sensor with signal conditioning and an interface to the HART communications protocol

B. Measurement-related properties of distributed measurement nodes

Measurement-related properties include:

1. Measurement timing management: timed, polled, random, etc.
2. Local data management: store until requested, broadcast upon collection, etc.
3. Local computation: average, peak value, etc.
4. Identity: node identification, description, etc.
5. Location: coordinates or identifier of the measurement point

Instruments manage some but not all of these properties. Management is usually accomplished via external commands or from front panels. Currently available transducers generally do not deal with these properties.

For nodes to be generally useful in distributed systems, these sorts of properties need to be managed within the node.

C. System or application-related properties of distributed measurement nodes

System or application-related properties include:

1. Changing measurement properties in response to application-related messages, e.g., changing the sample rate in a collection of nodes
2. Defining the communication patterns among nodes, e.g., master/slave, client/server, peer to peer, etc.
3. Establishing and modifying communication patterns among nodes, e.g., modifying multicast membership
4. Managing the transport properties of communication among nodes: flow control, reliable delivery, etc.
5. Synchronizing the node clocks, if present
6. Conforming to system data and control models

Few current instruments, sensors, or actuators have the capability to manage these properties. There are a few new LAN-based instruments that have some capabilities in this area.

These properties are often the dominant factors in determining the usefulness of nodes in distributed measurement systems. Without proper support for managing these properties within the nodes, the design and implementation of a distributed system requires impractical levels of low level programming for each node.

D. Communication protocol issues

Multicast communication capability is desirable for distributed measurements. Here multicast means that MxN communication is supported, that is, multiple sources can communicate with multiple sinks.

In a distributed system, the logic that determines behavior resides in the nodes. In all but the most trivial measurements, this logic requires communication between

nodes in a variety of patterns. For example, data collected at a given node may be needed at several other nodes for display, archiving, or perhaps setting an actuator. Likewise, a given node, for example, a display node, may be managing data from a number of sensor nodes. This sort of communication is a good match to a multicast capability. Point-to-point communication is a special case of multicast and will be needed occasionally in distributed measurement systems.

Most existing protocols in use in test and measurement implementations present a point-to-point interface to the designer. However, it is worth noting that at the physical layer most are fundamentally multicast protocols since all nodes share a common multicast medium such as coax or RF. Notable exceptions are RS-232 and the 4-20ma loop which are dedicated physical links and can not support a multicast protocol. Point-to-point protocols are not adequate for distributed measurements as envisaged in this paper.

In the case of the IEEE-488 protocol, information transfer is restricted to a single talker at a time and is not as flexible as a true multicast protocol. The IEEE-488 protocol is not a good candidate protocol for distributed measurement systems.

LAN protocols often have a multicast capability for specialized needs. However, the normal application interface is point-to-point. The internet protocol, IP, layer in the protocol stack is usually used for point-to-point links. Most network applications (e.g., FTP, Telnet, RPC, NFS) use IP in this manner. There is provision for multicast in the IP layer. This multicast capability is often used for system-level protocols, e.g., NTP, and can be used for distributed measurement protocols as well.

One of the challenges in developing distributed measurement systems is finding appropriate multicast implementations.

E. Data management issues

In a distributed system the management of data must be more structured than in traditional systems. In centralized systems many data management tasks such as identifying the source and time of a measurement, are based on the properties of point-to-point communication links. In a distributed system using multicast, other techniques must be used for binding the various pieces of information in the system.

An example of such a technique is the data model for physical measurements. It is necessary to include the value or a reference to the value in the data sent from a distributed node for any item normally inferred from the properties of a point-to-point link in a traditional system, e.g., source node identity. The minimum elements of the relation representing a measurement include attributes for the value, units, time of measurement, the location of measurement, and usually some name. Depending on the type of system being built, additional elements such as accuracy or precision may be included. A more detailed description of the data models used in the prototype system is included in Appendix A of this paper.

The use of data models with the information binding implemented in the nodes, provides the best basis for realizing this project' s objectives, e.g., the "plug-and-play" behavior .

F. Control protocol and real-time issues

In a traditional centralized system, behavior is managed by the controller issuing detailed command messages to each of the remote nodes. Such systems, built using existing instruments and sensors, require many of these messages to be concerned with details of the internal operation of the node. These detail messages often dominate network traffic, [4].

In distributed measurement systems, the details of system behavior are determined by the nodes. The control protocol must therefore support each node internally managing the application details occurring at that node. In addition, the control protocol must support the transmission of synchronization messages between nodes to produce the correct overall system behavior .

Synchronization includes not only the timing of measurements but the overall progress of the application from one sequence of events to another .

In the distributed measurement systems we have built to date, the communication traffic consists mainly of these application synchronization messages and application measurement data.

There are three main real-time considerations in measurement systems:

1. the relative timing of measurements and real world phenomena, i.e., synchronization
2. the time needed to transport data
3. the time needed to process data

In traditional systems, synchronization is managed by the central controller , often in combination with hard-wired triggers between nodes. Any notion of real time is imposed by the central controller on the received data.

In distributed measurement systems, each node must explicitly deal with synchronization. Order , but not time specification, can be implemented via messages passed between nodes. If a true time specification is to be imposed, then nodes must have access to a clock. [1] A central time server is one possibility but introduces delays and probably excessive message traffic. For systems with more than a few nodes, a better choice is for each node to have a local clock participating in a synchronization protocol among the nodes.

1) Simple ordering of events can be accomplished by a message exchange. For example, to ensure that event A1 in node A executes before event B1 in node B (ordering), a message would be sent by node A after executing A1, to node B. Upon receipt, node B would execute B1. A major deficiency of this scheme is that only the order of the events is specified. The time difference between the two events (e.g., event B1 must follow event A1 by 50 msec.) cannot be specified because there is no control over the processing or delivery time of the messages.

To provide this time specification, it is necessary to base the execution of events on time rather than on the receipt of a message. By accessing a sufficiently accurate time reference, each node can be programmed to execute events at the appropriate times. The event times may be preprogrammed or based on times communicated in messages.

The real time requirement for transporting data over the network and processing this data within nodes is an issue of network and node design. In systems with many nodes, network bandwidth may be at a premium for many applications. Consequently , careful consideration of such things as the use of acknowledgments is necessary to avoid saturating the network. Like others [5], we have found using unacknowledged transport mechanisms with the appropriate end-to-end acknowledgment at the application level, minimizes network traffic and is appropriate for distributed systems.

IV. PROTOTYPE SYSTEM

A. Design objectives and specifications

The design objectives define the following specifications of system and node behavior and capability:

1. Connection of a node to the medium is all that is required for the node to become operational in the system (in a sensible way).
2. Deletion of a node for any reason (e.g., disconnection or power-down) does not cause any failure of system behavior, other than those effects due to missing the production or consumption of data and messages associated with the node.
3. All necessary protocols for operation of the system are distributed, i.e., there is no need for any form of central control during the operational phases of the system.
4. All nodes are capable of accepting an "application script" that tailors the inherent behavior of the node to meet the requirements of the specific overall system application.
5. Central control is permitted *but is not required* as a means of managing the modification of node behavior for specific applications. However, for this study the role of the central controller was limited to downloading "applications" and issuing commands to change the selection or parameters of one of the behavioral models within a node.
6. All nodes implement the same standard data, control, and behavioral models.
7. All nodes contain clocks that are synchronized with the clocks in other nodes of the system.
8. All nodes provide information sufficient to properly interpret communications with the node.
9. In this report, nodes that meet these design criteria are referred to as "smart" nodes.

B. General description of an application using the prototype system

As illustrated in Figure 1, a typical system contains a number of smart sensors and actuators. Laboratory prototype smart sensor nodes have been implemented to measure light intensity , relative humidity , temperature, carbon dioxide concentration, sound level, power line voltage, power line frequency , pH, fluid flow , acceleration, and atmospheric pressure. Laboratory prototype actuator nodes have been implemented to control a relay , control a D.C. motor , display a voltage, and display a byte of data with a group of LEDs.

In addition to the smart actuators and sensors, a number of laboratory prototype smart system nodes have been built to illustrate more complex functionality , while still being good citizens in the distributed measurement system. The display and archive nodes accept the output of all of the sensors and parse and display the data or save it to a file.

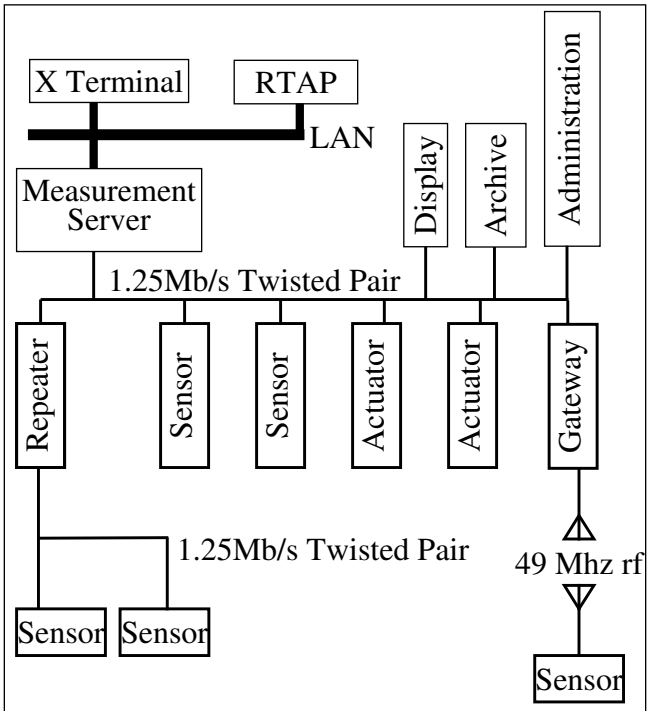

Figure 1. Prototype System Overview

A variety of communication media are used in the demonstration system. An Ethernet LAN is used in the traditional portion of the system. A1.25Mb/s twisted pair and a 49Mhz low-power RF link are used for communication in the distributed portion of the system. A repeater is shown to illustrate the segmentation of the network allowing more nodes than permitted with a single twisted pair .

The administration node downloads applications into smart nodes and sends commands to change the internal model selection and parameters.

The smart sensors, actuators, display , archive, and administration nodes constitute a pure distributed measurement system in the terms of this article. The remainder of the components illustrate how such a system might interface to traditional systems.

The measurement server node provides a remote-procedure-call, RPC, based interface to the LAN and provides visibility into the distributed measurement system. In this system the measurement server serves as a gateway by converting LAN packets to and from equivalent packets on the smart sensor network. One of the remaining research issues is the design of measurement servers and the way such servers present distributed measurement systems to the more traditional systems.

The measurement server communicates with RT AP, a Hewlett-Packard SCADA supervisory system. RT AP accepts the data from the smart sensors as delivered by the measurement server node. This data is archived in the RT AP database and is available for display , analysis, trending, etc. Access to RT AP is from any X-terminal on the LAN.

C. Smart node architecture

The architecture for the prototype smart nodes is illustrated in Figure 2.

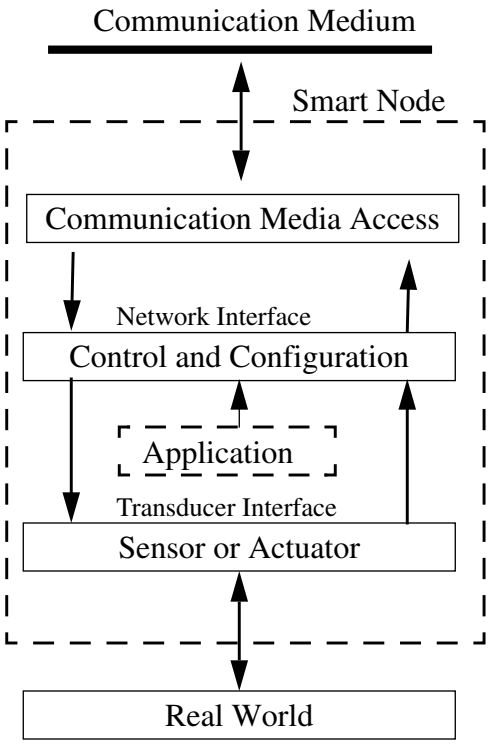

Figure 2. Smart Node Architecture

The Communication Media Access block handles the low-level protocol required to access the physical medium. Included in this block, is a media-dependent transceiver . This transceiver is the physical interface to the medium. Ideally , the interface between this block and the Control and Configuration block, i.e., the network interface, is media and protocol independent.

The Control and Configuration block includes the remainder of the protocol stack, i.e. the media and protocol independent portion, and the control and computation circuitry needed to implement the functions of the smart node. This Control and Configuration block is identical in all smart nodes.

The optional Application block consists of a ROM containing an application-specific script. Applications may also be loaded over the network. This block is located on a card that plugs into the smart node.

The Sensor or Actuator block contains the sensor or actuator transducer , any additional circuitry not contained in the control and configuration block, and an electronic data sheet containing specifications and calibration information unique to the specific sensor or actuator .

The two possible data paths within the smart node Control and Configuration block are illustrated in Figure 3.

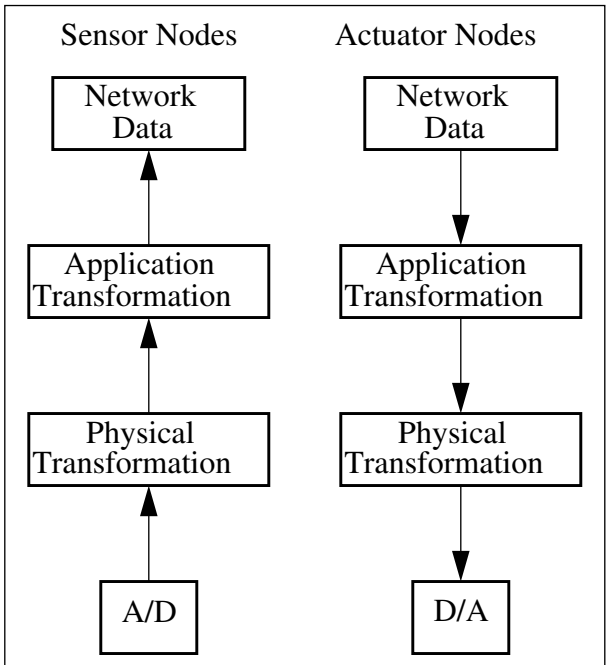

Figure 3. Data Path

The physical transformation converts between the digital representation in the International System of Units, SI, and the raw digitized representation provided by the transducer . This conversion is based on information from the “electronic data

sheet" contained in the transducer .

The application transformation is converts between the SI units representation and the application representation which appears at the network. In the absence of any provided application this will default to a SI unit representation at the network. This application transformation implements the portions of the application specific to the node. The transformation can be quite general. In addition to operations on the data, e.g. change of units, averaging, filters, limits, etc., the transformation can generate messages on the network.

In addition to the transformations on the data, the Control and Configuration block implements the behavior models defined for the smart nodes. These models specify the behavior of the node with respect to properties like sampling rate and data management.

D. Operational aspects of smart nodes

The internal operation of the smart nodes may be divided into two phases; start-up and normal-operation.

The start-up phase occurs after power -up or reset. During the start-up phase, the following sequence of events takes place within the node:

1. The transducer uploads the information contained in the "electronic data sheet".
2. Based on this information, the node configures itself as a sensor or as an actuator. In addition, it configures the physical transformation, as well as operating characteristics imposed by the transducer, for example, warm-up time and minimum sampling interval. Thus it is possible to completely change the nature of a node by substituting a different transducer. For example, a temperature transducer could replace a pressure transducer, or perhaps another temperature transducer with lower accuracy. These changes are reflected automatically in the transducer-related node behavior.
3. The contents of the application ROM upload. Based on this information, the node configures the application transformation.
4. The node monitors the network to detect the presence of other nodes. Based on the information received, the node configures the relevant properties, e.g., data management options.
5. At the end of the warm-up time the node begins normal operation.

Typical functions occurring during the normal-operation mode include:

1. For actuators, the node receives data via a multicast or point-to-point link. This data then moves along the data path illustrated in Figure 3 and normally results in an update of the actuator at the appropriate sampling time.
2. For sensors, the node triggers the sensor at the appropriate sampling time. The data then moves along the data path. Depending on the application transformation, this results in data or messages being placed on the network.

The application transformation can also result in more complex behavior such as reporting by exception.

3. For any node, the node monitors the network for messages that indicate a change in the operational behavior, e.g., multicast membership changes.

IV. INTERF ACE DEFINITIONS

The key to successful use of distributed technology lies in the interface and behavioral definitions. All members of a distributed system must adhere to the same set of definitions to permit graceful scaling, modification and operation of the system.

In smart nodes, the two key interfaces are the transducer interface and the network interface.

A. Transducer interface

This interface serves to define all aspects of the smart node that depend only on the transducer . There are four areas of concern: the physical form of the interface, identity specifications, operational specifications, and calibration specifications.

1. The physical form of the interface includes definitions for all signals, power levels, connectors, etc.
2. The identity specification must include a unique identifier that allows the smart node to determine whether the transducer has been changed. The prototype specification includes the manufacture's identification and an ASCII textual description of the transducer.
3. The transducer operational specifications of the prototype include the minimum sampling interval, transducer warm-up time, acquisition time, units, data representation, precision, and accuracy.
4. The calibration specification includes the method and parameters to enable the physical transformation to convert between SI units and the transducer signal. Also, the transducer contains the specification of the expiration of the calibration. To allow for recalibration, these items are stored in EEPROM.

This interface is the subject of a current NIST/IEEE standardization effort.

B. Network interface

This interface defines all aspects of the smart node that are visible to other nodes on the network. Data, connection, and behavioral models specify the network interface.

The use of common models allows the design of nodes that can operate on the data without further configuration, one of the objectives of this study .

The main feature of the data model is the network-level representation for measurement data. This representation includes value, units, time of measurement, and place of measurement. The data is normally communicated as a multicast message named "data" and is received by all nodes in the multicast group. Receiving nodes filter the incoming data based on their internal application information and on the

fields of the data message. For a more detailed discussion see Appendix A.

V. EXPERIENCE USING THE PROTOTYPE SYSTEM

The prototypes have been used in a number of applications which are described in the following sections.

A. Printed circuit board manufacturing

A system much like the one illustrated in Figure 1 was used to monitor conditions in the wet-chemical processing portion of a printed circuit board manufacturing operation within Hewlett-Packard. The system includes smart nodes to measure the temperature and pH of the solutions in various tanks and the flow rates and line pressure in the automatic spray equipment.

The measurement server transfers measurement results to an RT AP installation for process monitoring.

The interesting feature of this application is that periodic monitoring of the flow rates is inappropriate since the spray process is intermittent. An "application transformation" was produced which allowed the node to make measurements at the desired rate during spray time, but only place the data on the network when flows above a certain threshold are observed.

Data from RT AP is accessible via an X-terminal from anywhere within Hewlett-Packard. W e have also implemented a Mosaic interface to this data for demonstration purposes.

B. Laboratory ambient condition monitoring

A system similar to the printed circuit board system has been installed in several laboratory areas at Hewlett-Packard. This is a straightforward monitoring of atmospheric pressure, ambient temperature, humidity and concentration of carbon dioxide. RT AP is used for archival recording of the data for later correlation with defect data associated with various operations being conducted in these laboratories.

C. Miscellaneous systems

A variety of systems have been built to demonstrate various aspects of distributed measurement such as:

1. controlling a closed loop system
2. adding nodes to running systems
3. changing system application behavior based on events observed in one or more nodes and communicated using a multicast operation
4. partitioning of systems into multiple multicast groups

All of these systems used the same nodes. Only the application transformations were modified.

D. Observations

The following observations are based on our experience with the limited number and types of nodes in our current systems.

1. The maximum measured sampling rate possible for the current nodes is 25 samples per second. This figure can be expected to improve with a node processor faster than the Motorola MC143150 and Toshiba TMPN3150 chips used to implement the nodes.
2. The measured deviation of the sample period for a single node making periodic measurements at a rate below the maximum is on the order of 5 milliseconds.
3. The timing offset accuracy between nodes is on the order of 6 milliseconds based on comparing the time-stamps of observations of periodic measurements started at the same time.
4. System application writing and debugging is generally easier than that for a corresponding, centrally-controlled systems. This appears to be the result of:
   a. partitioning the application into node-specific operations which are implemented at the node
   b. using synchronization messages between the nodes
   c. using common data models which promote inter-changeability and inter-operability by eliminating excessive data manipulation
   d. using multicast communication patterns which provide more flexibility and allow for graceful addition of nodes
   e. modifying the system usually requires modifying the application specification only at the few directly involved nodes
5. System application writing and debugging would be easier if explicit support for application-events had been provided
6. Identifying, partitioning, and modelling transducer, measurement, and system or application-related issues allows node design to explicitly support these issues, greatly reducing the difficulty in constructing applications
7. Adopting SI units for all internal operations simplifies the design of the nodes. Application development is also simplified since only those nodes which must present data in human readable form need concern themselves with the variety of units representations for the same measure. Additional research is needed to design a robust specification technique for compound and dimensionless units [6].

VIII. TOPICS FOR FUTURE RESEARCH

As a result of our investigation, several topics were noted in which additional research is needed to allow more effective design and use of distributed system applications. These include:

1. Determining the set of useful models and support structures for implementing the application level of distributed systems. This will result in a more formal

definition of an interface between the “application” and “measurement” portions of the node architecture.

2. Designing a robust specification technique for compound and dimensionless units, and variable naming.
3. Designing an effective method for implementing unique identifications for transducers, nodes, etc.
4. Designing the environments for application development, simulation and debugging of distributed measurement systems.
5. Specifying a network interface, as defined in this paper, that is independent of the actual network transport and physical protocols used.
6. Ensuring low cost node design.

VII. CONCLUSIONS

This study has shown that the use of distributed system technology is useful and important in measurement, monitoring and control systems. With further development these techniques will enable straightforward construction of applications which are easily configured, expanded, modified, and maintained.

Appendix A: Detailed description of system models.

This appendix provides more detail into the top level definitions of the various models and interfaces of the smart nodes. There is considerably more detail to these models than is possible to present in this paper. However the level of definition given should illustrate how these models support the design objectives of the prototype system.

In the following sections, *italics* indicates that the item is an explanation intended to convey the intent of the underlying detail of the item, i.e., it is not a terminal symbol for the definition. A complete definition would extend to the bit structure of the representation.

A. Network interface

network-interface:
    network-data
    network-time
    application-script
    node-control
    network-connectivity
network-data:
    node-id
    sequence-number
    time-stamp
    variable-parameters

value

node-id:

A unique identifier referencing the source of the record

sequence-number:

The ordinal number of the set of network packets conveying this item of network-data. This may be a compound sequence number if network packet fragmentation is needed.

time-stamp:

A representation of the local node time. When related to data it indicates the time at which the data became or is to become valid.

variable-parameters:

variable-type
variable-name
variable-units
data-model

variable-type:

A type indicating the nature of the data, e.g. normal physical data, normal physical data from a node whose calibration is overdue, application level data,....

variable-name:

A key to either a standard enumerated variable names or an expression defining a user created name.

variable-units:

A key to either a standard enumerated SI unit or an expression defining a SI compound or user created unit

data-model:

A type indicating the data type used to represent the data value, e.g. IEEE-float, 3 IEEE-float vector , etc.

value:

The value of the data.

network-time:

node-id
time-stamp

application-script:

application-id
application-model
application-variable-map
application-specification

application-id:

A system unique identifier referencing this application

application-model:

The model used to represent the application. For the prototype: series, categorization, subroutine.

application-variable-map:

Defines the names, types, units and relationships between the variables at the input and output of the application transformation defined by the specification.

application-specification:

As appropriate to the application-model, defines either the relevant parameters or an executable subroutine.

node-control:

A definition of commands which may be used to change node state or to set parameters of various behavioral models, e.g. reset, stop-sampling, set-sample-rate.

network-connectivity:

A set of commands and data used to establish or modify the network connectivity .

B. Behavioral models

The selection and definition of the node behavioral models, along with the data models, visible either from the transducer or from the network determines the interchangeability and interoperability properties of the smart nodes in a distributed system. The following are two of the models used in the prototype to illustrate this point.

Operational state model:

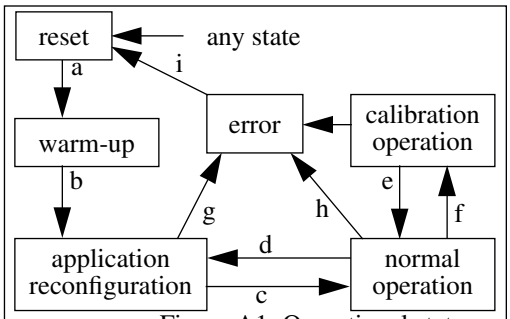

Figure A1: Operational state

The network state modifications possible from the network via “node-control” are:

reset (or power up)

d: download a new application

f: calibrate the node

e: accept new calibration

f: use previous calibration

i: reset

All other transitions are governed internally .

Sampling state model:

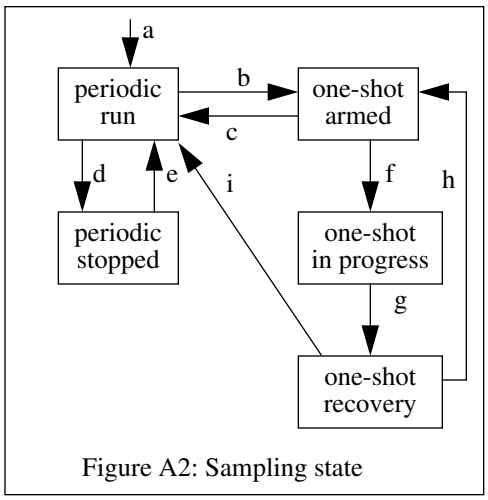

Figure A2: Sampling state

The network state modifications possible from the network via “node-control” are:

a: reset (or power up)
b: start-polled-mode
c: start-periodic-mode
d: start-measurement
f: start-measurement
h: arm-measurement
i: start-periodic-mode

All other transitions are governed internally .

C. Transducer interface

transducer -interface:
- operational-parameters
- transducer -description
- conversion-specification
- calibration-specification

operational-parameters:
- transducer -type
- transducer -id
- transducer -variable-name
- transducer -units
- range-model
- warm-up-time
- acquisition-time
- minimum-sampling-period
- physics-representation
- transducer -representation
- precision
- accuracy

transducer -type:

Indicates whether the transducer is a sensor or an actuator and the form of the physical interface. In the prototype the forms included: voltage signal, frequency signal and a byte stream.

transducer -id:

A unique identifier for the transducer .

transducer -variable-name:

A key to either a standard enumerated variable names or an expression defining a user created name.

transducer -units:

A key to either a standard enumerated SI unit or an expression defining a SI compound or user created unit

range-model:

Specifies the representation and values of the range of the physical variable accessible to this transducer .

warm-up-time:

The time required after power is applied before the transducer is operational.

acquisition-time:

The time required after the trigger signal applied to the transducer before the data is valid.

minimum-sampling-period:

The minimum time required between successive samples.

physics-representation:

The data representation used on the network side of the physical transformation. (see Figure 3)

transducer -representation:

The data representation used on the transducer side of the physical transformation.

precision:

The precision of the calibrated data.

accuracy:

The accuracy of the calibrated data.

transducer -description:

An ASCII description of the transducer .

conversion-specification:

conversion-model
default-conversion-values
calibrated-conversion-values

conversion-model:

Specifies the type of the conversion model used. In the prototype a series, a categorization and subroutine models are supported.

default-conversion-values:

Default value of the conversion values appropriate for the selected model.

calibrated-conversion-values:

Conversion values appropriate for the selected model.

calibration-specification:

calibration-type
calibration-description
calibration-script

calibration-type:

Specifies the method used to calibrate the transducer. The prototype supported: no calibration possible, self calibration, and an externally managed N point calibration.

calibration-description:

A human readable set of instructions on the calibration procedure.

calibration-script

An executable script defining the externally managed N point calibration.

REFERENCES

[1] James Pinto, ISA paper #94-598, "Networked, Programmable, Intelligent I/O, The 'Truly' Distributed Control Revolution", Proceedings of the Industrial Computing Conference, vol4, part 2, pgs. 141-147, October 23, 1994.

[2] Gary Tapperson, ISA paper #94-569, "Fieldbus: Migrating Control to Field Devices", Advances in Instrumentation and Control, vol. 49, part 3, pgs. 1239-1252, October 23, 1994

[3] James Pinto, ISA paper #94-514, "Fieldbus- A Neutral Instrumentation Vendor's Perspective", Advances in Instrumentation and Control, vol 49, part 2, pgs. 669-675, October 23, 1994.

[4] Stan Woods, and Keith Moore, "An Analysis of HP-IB Performance in Real Test Systems," Hewlett-Packard Laboratories Technical Report 93-20, August 1992

[5] J.H. Saltzer, D.P.Reed, and D. D. Clark, "End-To-End Arguments in System Design", ACM Transactions on Computer Systems, vol 2, No. 4, November 1984, pages 277-288.

[6] Stephanie Leichner, William Kent, Bruce Hamilton, "Dimensioned Data", Hewlett-Packard Laboratories private communication, February 1995.