An IEEE 1451.1 Summary

Rick Schneeman, Computer Scientist

rschneeman@nist.gov

US Department of Commerce
National Institute of Standards and Technology (NIST)
Gaithersburg, Maryland 20899 USA

Introduction

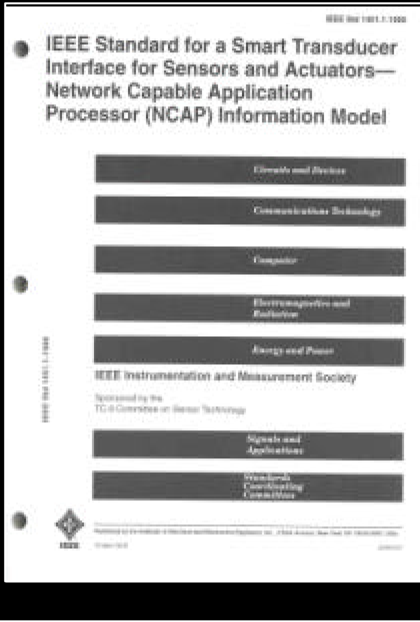
- Who we are: NIST mission is to help increase US industry competitiveness through advanced research, standards, and technology collaboration
- Member of the Sensor Development and Application Group (SDAG) within the Manufacturing Engineering Laboratory (MEL) at NIST
- Member of the Working Group on the IEEE Standard for a Smart Transducer Interface for Sensors and Actuators — Network Capable Application Processor (NCAP) Information Model, or IEEE 1451.1 (“dot1”)

IEEE 1451 Overview/Goals

- Provide standardized communication interfaces for smart transducers, both sensors and actuators. In the form of a standard hardware and software definition/specification.
- Simplify the connectivity and maintenance of transducers to device networks through such mechanisms as common Transducer Electronic Data Sheet (TEDS) and standardized Application Programming Interfaces (API)
- Allow plug-and-play with 1451 compatible transducers among different devices using multiple control networks
- Give sensor manufacturers, system integrators, and end-users the ability to support multiple networks and transducer families in a cost effective way

Part 1: IEEE 1451.1 Overview/Goals

- *"The specifications provide a comprehensive data model for the factory floor, and a simple application framework to build interoperable distributed applications*…" Dr. Jay Warrior, Agilent Technologies, Chair IEEE 1451.1 WG
- In general, IEEE 1451.1 accomplishes this by providing:
  - Transducer application portability (software reuse)
  - Plug-and-play software capabilities (components)
  - Network independence (network abstraction layer)



IEEE Std 1451.1-1999

IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Network Capable Application Processor (NCAP) Information Model

Circuits and Devices

Communications Technology

Computer

Electromagnetics and Radiation

Energy and Power

IEEE Instrumentation and Measurement Society

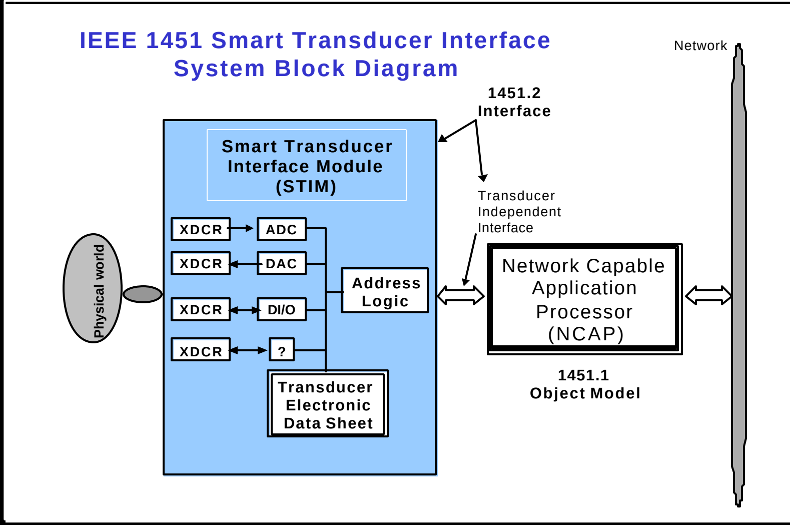
Sponsored by the
TC-9 Committee on Sensor Technology

Signals and Applications

Standards Coordinating Committees

IEEE Std 1451.1-1999

- The standard specifies these capabilities by defining software interfaces for:
  - Application functions in the NCAP that interact with the network that are independent of any network
  - Application functions in the NCAP that interact with the transducers that are independent of any specific transducer driver interface

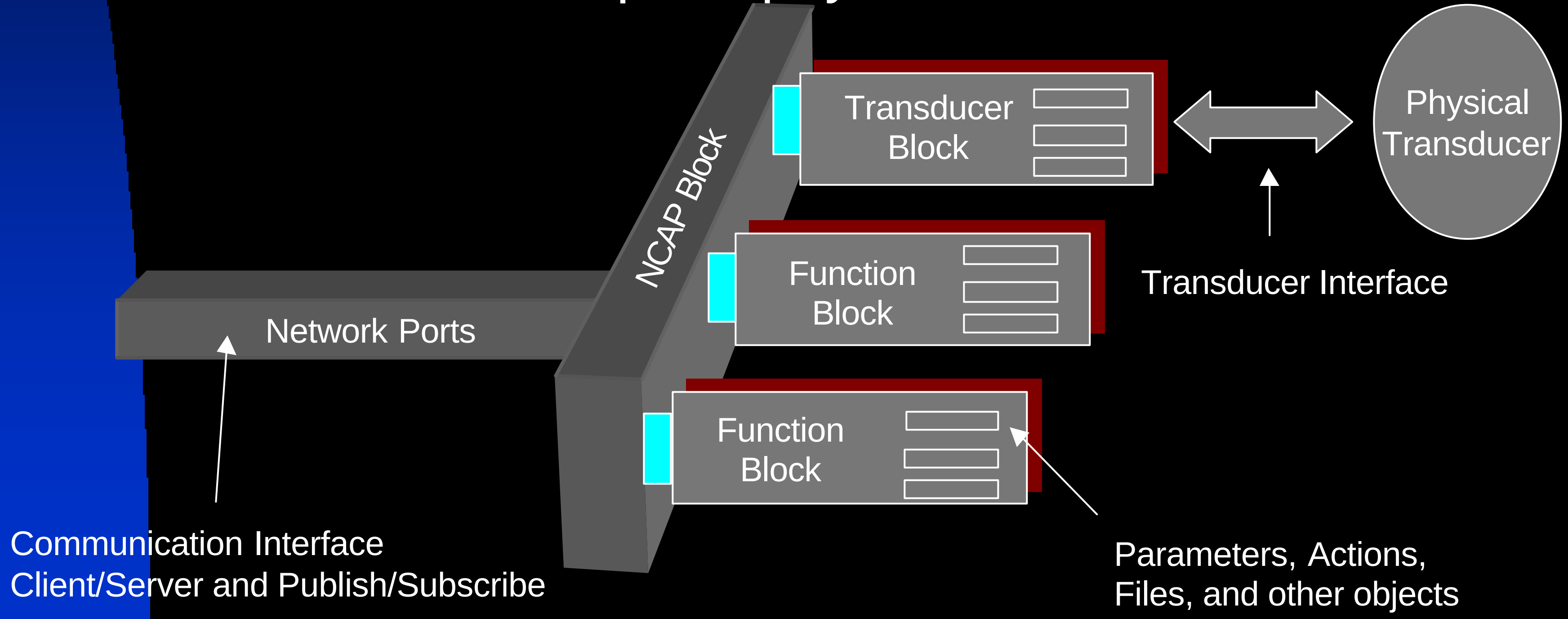


IEEE 1451.1 Overview/Goals (Cont.)

- IEEE 1451.1 software architecture is defined using three different models or views of the transducer device environment:
  - An Object Model, defines transducer device specific abstract objects – or, classes with attributes, methods, and state behavior
  - A Data Model, defines information encoding rules for transmitting information across both local and remote object interfaces
  - A Network Communication Model, supports a client/server and publish/subscribe paradigm for communicating information between NCAPs

Conceptual View of an IEEE 1451.1 NCAP

- Uses a “backplane” or “card cage” concept
- NCAP centralizes and “glues” all the system and communications facilities together
- Network communication viewed through the NCAP as ports
- Function block application code is “plugged” in as needed
- Transducer blocks map the physical transducer to the NCAP



Communication Interface
Client/Server and Publish/Subscribe

Parameters, Actions, Files, and other objects

IEEE 1451.1 Communication Model

- Provides two styles of inter-NCAP communication
- Client/Server: A tightly coupled, point-to-point model for one-to-one communication scenarios – typically used for configuration, attribute accessors, and operation invocations
- Publish/Subscribe: A loosely coupled, model for many-to-many and one-to-many communication scenarios – typically used for broadcasting or multicasting measurement data and configuration management (i.e., node or NCAP discovery) information

Implementing IEEE 1451.1

- An IEEE 1451.1 C++ Reference Implementation provides a concrete representation of the abstract Smart Transducer Information Model (IEEE Std 1451.1-1999, Dated 18 April 2000). The NIST implementation is called “1451.1 Lite”, as it is a subset of the complete specification.
- A subset of the IEEE 1451.1 implementation has also been developed in Java to provide an architecture neutral NCAP configuration tool.
- The C++ implementation uses the open-source Adaptive Communication Environment (ACE) from the Washington University at St. Louis.

IEEE 1451.1 Benefits

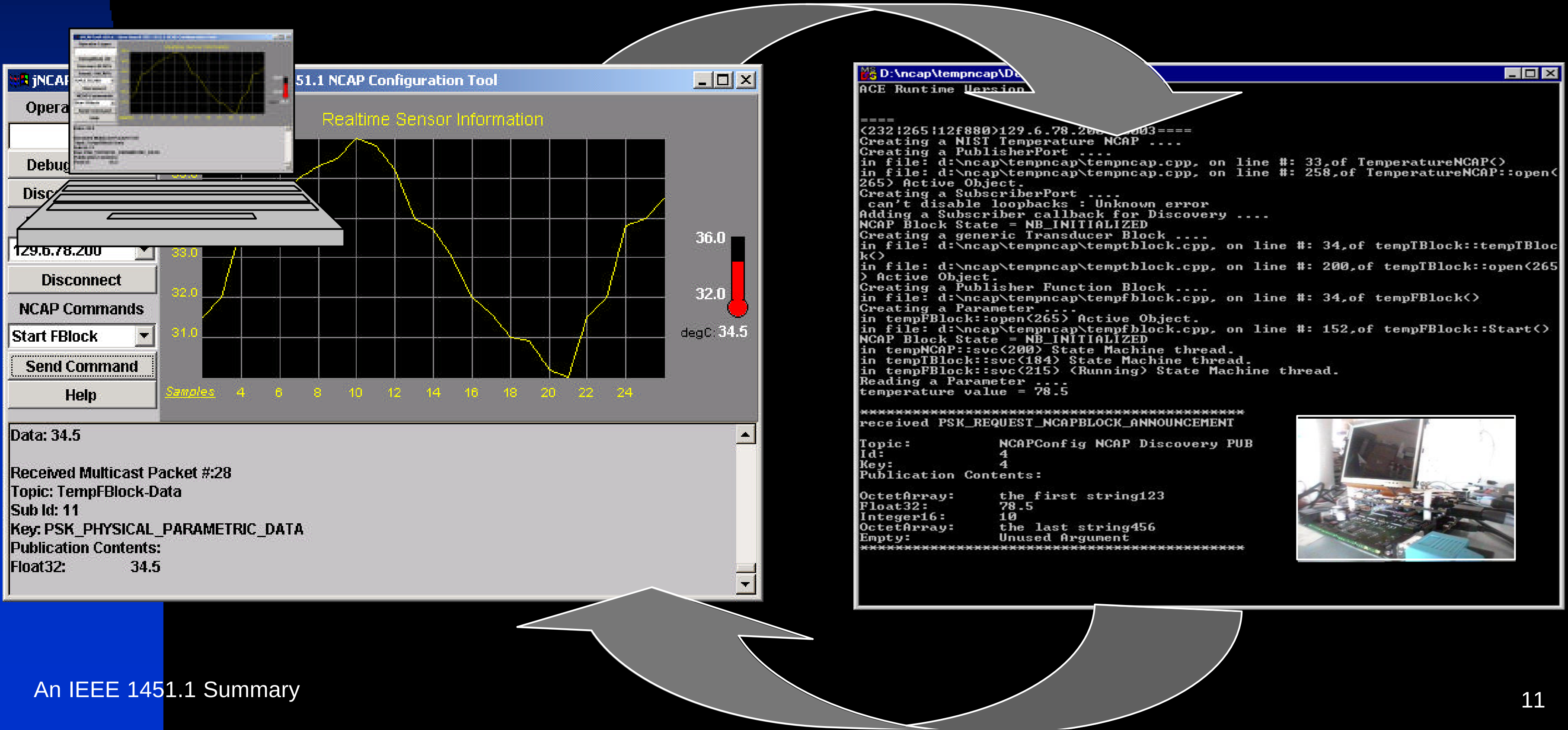
- Using P1451.1 provides:
  - an extensible object-oriented model for smart transducer application development and deployment
  - application portability achieved through agreed upon application programming interfaces (API)
  - network neutral interface allows the same application to be plug-and-play across multiple network technologies
  - leverages existing networking technology, does not re-implement any control network software or protocols
  - a common software interface to transducer hardware i/o

Looking at an IEEE 1451.1 Application

- A minimal IEEE 1451.1 application consist of a few classes:
  - An NCAP Block (consolidates system and communication housekeeping)
  - A Transducer Block (provides the software connection to the transducer device)
  - A Function Block (provides the transducer application algorithm (i.e., obtain and multicast temperature data every second)
  - Parameters (contains the network accessible variables that hold and update the data)
  - Ports (network communication objects for publishing and subscribing to information or interacting with other NCAPs using client/server

Executing an IEEE 1451.1 Application

- An embedded Temperature NCAP Application is running from a remote location on the NIST Intranet
  - As part of the system configuration, a NIST developed Java tool on a Notebook issues a discovery multicast, finds the NCAP, and starts the remote NCAP’s Function Block
  - The remote NCAP Function Block responds by publishing temperature data every second as the Java tool records the information



```
Data: 34.5

Received Multicast Packet #:28
Topic: TempFBlock-Data
Sub Id: 11
Key: PSK_PHYSICAL_PARAMETRIC_DATA
Publication Contents:
Float32:          34.5
```

D:\ncap\tempncap\D

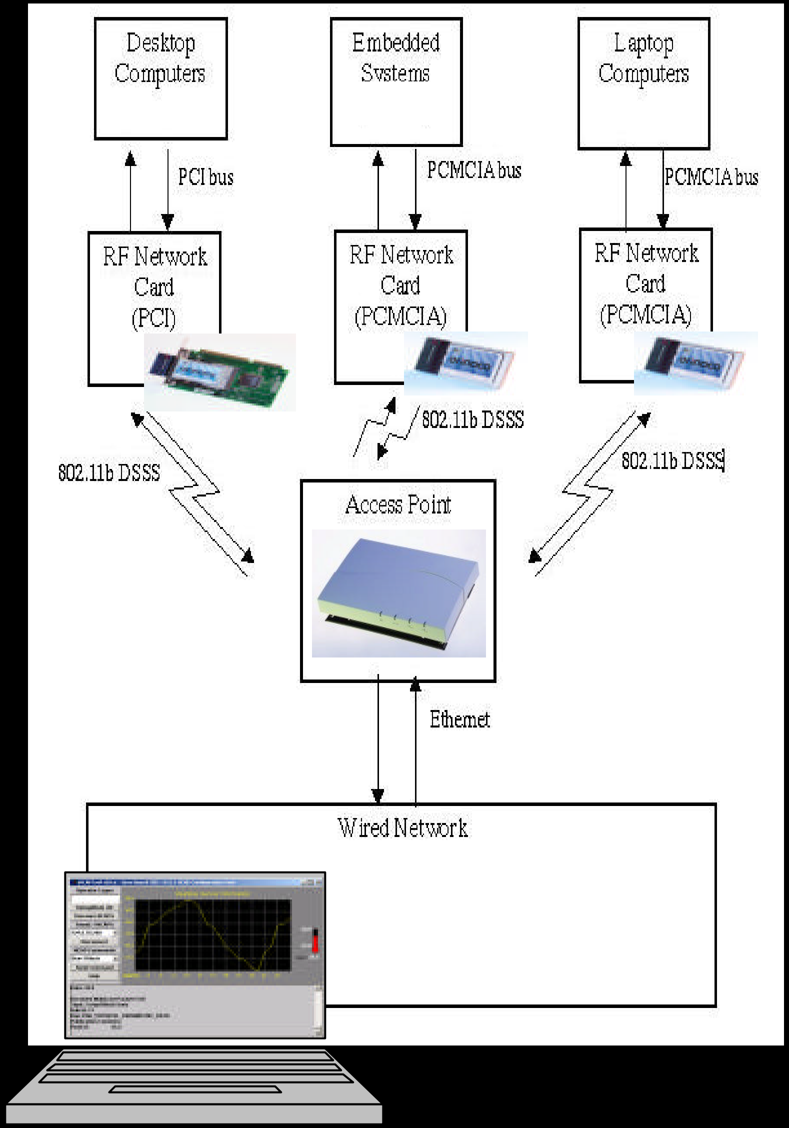
```
ACE Runtime Version


====
<232|265|12f880>129.6.78.20   003====
Creating a NIST Temperature NCAP ....
Creating a PublisherPort ....
in file: d:\ncap\tempncap\tempncap.cpp, on line #: 33,of TemperatureNCAP()
in file: d:\ncap\tempncap\tempncap.cpp, on line #: 258,of TemperatureNCAP::open(
265) Active Object.
Creating a SubscriberPort ....
 can't disable loopbacks : Unknown error
Adding a Subscriber callback for Discovery ....
NCAP Block State = NB_INITIALIZED
Creating a generic Transducer Block ....
in file: d:\ncap\tempncap\temptblock.cpp, on line #: 34,of tempTBlock::tempTBloc
k()
in file: d:\ncap\tempncap\temptblock.cpp, on line #: 200,of tempTBlock::open(265
) Active Object.
Creating a Publisher Function Block ....
in file: d:\ncap\tempncap\tempfblock.cpp, on line #: 34,of tempFBlock()
Creating a Parameter ....
in tempFBlock::open(265) Active Object.
in file: d:\ncap\tempncap\tempfblock.cpp, on line #: 152,of tempFBlock::Start()
NCAP Block State = NB_INITIALIZED
in tempNCAP::svc(200) State Machine thread.
in tempTBlock::svc(184) State Machine thread.
in tempFBlock::svc(215) (Running) State Machine thread.
Reading a Parameter ....
temperature value = 78.5

*********************************************
received PSK_REQUEST_NCAPBLOCK_ANNOUNCEMENT

Topic:          NCAPConfig NCAP Discovery PUB
Id:             4
Key:            4
Publication Contents:

OctetArray:     the first string123
Float32:        78.5
Integer16:      10
OctetArray:     the last string456
Empty:          Unused Argument
*********************************************
```

How NIST uses IEEE 1451.1 in a Wireless Environment

- The NIST C++ IEEE 1451.1 reference implementation uses TCP/IP as its underlying control network.
- From TCP/IP, IP multicast and TCP unicast features are used to implement publish/subscribe and client/server, respectively
- ACE is used to abstract the networking code from the application; therefore it is highly adaptive to various protocols
- Wired 802.3 Ethernet has been used primarily for testing. No changes were needed in ACE to support this protocol.
- Wireless 802.11b (11Mbps) Ethernet has also been used for testing. Again, no changes were made to ACE as the TCP/IP protocol is compatible with both 802.3 and 802.11b physical mediums.

How NIST uses IEEE 1451.1 in a Wireless Environment

- Testing scenarios included using a wired subnet connected to a wireless extension of the subnet
- Wireless extension uses an Agere (formerly Lucent) Orinoco AP-1000 dual card “access point”
- Range extender antennas are also connected to the access point and each PC-CARD
- Each node on the wireless side executes an IEEE 1451.1 NCAP application
- Java Configuration tool executes beyond the wireless net on the wired subnet



Summary

- IEEE 1451.1 is a large and comprehensive standard that addresses the needs of the smart transducer industry for providing portability and network independent access.
- NIST has begun implementing a good deal of the standard with emphasis on getting the software communication and infrastructure in place in order to start using the code.
- Choosing and implementing the standard with a solid object-oriented framework such as ACE provides a robust environment for real-time network communication.
- Migrating the implementation to other middleware such as CORBA for heavier weight uses will be reasonable to do
- Several projects at NIST will use the implementation for supporting manufacturing related activities

Summary (cont)

- Continued testing in the wireless space is required to gauge the effectiveness of the implementation.
- Bluetooth trials are forthcoming; however, the lack of multicast support will severely impact the applications – continued research here is a must
- Other lightweight middleware packages are going to be isolated – xml and soap, etc; however, these protocols do not support asynchronous messaging or publish subscribe in efficient ways
- Slimmer implementations of the IEEE 1451.1 will need to be experimented with for use with the smaller micro platforms.

For more information….

- ACE can be found at: *www.cs.wustl.edu/~schmidt/ACE.html*
- 1451.1-1999 IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Network Capable Application Processor Information Model 2000: ISBN 0-7381-1768-4
- The NIST IEEE 1451 Web Site provides information about 1451, publications, and demonstrations at: *ieee1451.nist.gov*