Next Generation Epics Interface To Abstract Data

Jeff Hill, Ralph Lange

In Pursuit of High Level Applications

- EPICS: a toolkit approach to *process control*
- EPICS: a toolkit approach to *physics modeling and control*?
  - Physics toolkits integrated with EPICS not routinely shared
  - They view EPICS as only a source and destination for data
- Can we better foster development of advanced toolkits above and beyond process control?
  - Are limitations imposed by EPICS communication software interfaces?

Fundamentals We Don't Intend to Change

- Publish-and-subscribe communication strategies
- Message-batching capabilities
- Asynchronous callback synchronized with external events
- Interfaces encourage proper toolkit structure
  - Robust response to loss of communication
  - Avoidance of application programmer introduced deadlocks
  - Portability between workstations and embedded systems

Fostering Integration With High Level Applications

- Vigorous open source collaboration requires well-defined software interfaces breaking large effort into moderate sized modular components
- EPICS communication software interfaces lack capabilities encouraging layering of software modules above and beyond the requirements of distributed process control.

Fixed Process Variable Attribute Set

Name	Display limits
Class	Control Limits
Data type	Alarm Limits
Vector dimension	Alarm condition
Value	Alarm Acknowledge Transient
Time Stamp	Alarm acknowledge severity
Units	Number of decimal digits
Multi-state label names	

Fixed Process Variable Subscription Event Set

Change of state (default dead band)
Change of state (archiving dead band)
Alarm condition change of state

Example Fixed Set Deficiency — Data Acquisition System

- Spool physics events off to disk
- Synchronize acquisition of property set with trigger event
  - Event and property sets not extensible by plug-ad-play applications
  - Advanced toolkits need to define new complex data types and event types

Example Fixed Set Deficiency — Star Tracking System

- Two parameters set new telescope position
  - EPICS can set only one process variable at a time
  - Risk of a less than optimal path to destination
- Ad-hoc methods must be contrived
  - Write position process variables and then write move command process variable
    - Error prone and thread unsafe interface

Example Fixed Set Deficiency — Star Tracking System

- Advanced toolkits need to install new complex data types initially unknown to core components
- This allows modern software communication paradigms such as message passing and command completion synchronization

Interfacing With Proprietary Data — Current Practice

- Many self-describing data file formats and associated programming interfaces are available
- Two methods are commonly used by *communications* systems

Interfacing With Proprietary Data — RPC systems such as CORBA

- Compiler reads file describing data structures and function call interfaces
  - produces header files for target language
  - produces object code stubs for transferring data on and off the wire
- Approach is very efficient at run time

Interfacing With Proprietary Data — RPC systems such as CORBA

- Cant extract arbitrary subset of elements from complex data type
  - Purpose of fields in data structure unknown
    - No route between arbitrary data structures in different programs
  - Impacts flexibility of publish and subscribe systems such as EPICS
- Difficulties when multi-dimensional array bounds change at run time

Interfacing With Proprietary Data — GDD and CDEV

- C++ class encapsulates proprietary data
  - Data stored internally as a union, or a linked list of unions if the data is compound
  - Each entry in the data is assigned a property type such as “units”, “limits”, or “time-stamp”
- Extraction of arbitrary property subset
- Insertion of new properties at any time

Interfacing With Proprietary Data — GDD and CDEV

- Large storage and execution overhead
  - Data description stored with *every* data instance
  - GDD has modal, complex support library
- Users find approach daunting
  - Frequent conversion between native storage formats and system's imposed data container
  - GDD efficiently indexes data by property identifier — but only in certain modes

Interfacing With Proprietary Data — Another Approach

- Toolkits exporting data derive from C++ abstract base class interface
- Programs aware of interface can examine or modify the data
- Support library provides functions for comparing, converting, copying between dissimilar data sets
- Toolkits export data in native format
- Data description can be fixed at compile time

Interfacing With Proprietary Data — Another Approach

- Compared with GDD and CDEV
  - Less complex support library
  - Data format not translated when crossing system interfaces
  - Lower storage and execution overhead
    - Data description may be stored separately from each data instance

Interfacing With Proprietary Data — Another Approach

- Compared with RPC systems
  - No compiler that generates stubs moving data on and off the wire
  - RPC stubs are more efficient
  - Similar per-instance storage overhead
  - RPC systems do not have facilities to extract property subset

Conclusions

- EPICS includes a comprehensive set of process control communications primitives
- But we aspire to cultivate advanced integration of high-level modular toolkits

Conclusions

- The fundamental endpoint for EPICS communication is the process variable with a fixed set of properties and subscription update events
- Advanced toolkits need to define new complex data types and new subscription update events initially unknown to system components

Conclusions

- New C++ based interface to arbitrary complex data eliminates existing barriers
- Important distinctions are revealed when comparing with existing practice
  - Subset of properties can be extracted from arbitrary data
  - Interface does not impose a storage format
  - Structure of the data can be efficiently determined at compile time