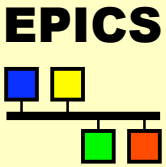
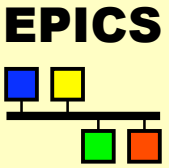
NSRL
EPICS
Seminar



Channel Access Concepts

Kazuro Furukawa, KEK, (2000-2004)
<kazuro.furukawa @ kek.jp>
(Bob Dalesio, LANL, USPAS1999)
(Andrew Johnson, APS, USPAS2003)

Why Channel Access?



- We don't have

  Network transparent computing in general (yet)
  - Plenty of administration tasks for sharing data
  - No standard way for remote processing
    - RPC, COM, CORBA, RMI, .NET, etc.

- In EPICS, Channel Access provides

  Efficient Software Bus on Network

  Performance, Rapid development, Maintenance, Administration

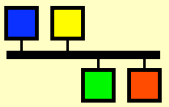
  Tuned for each operating system

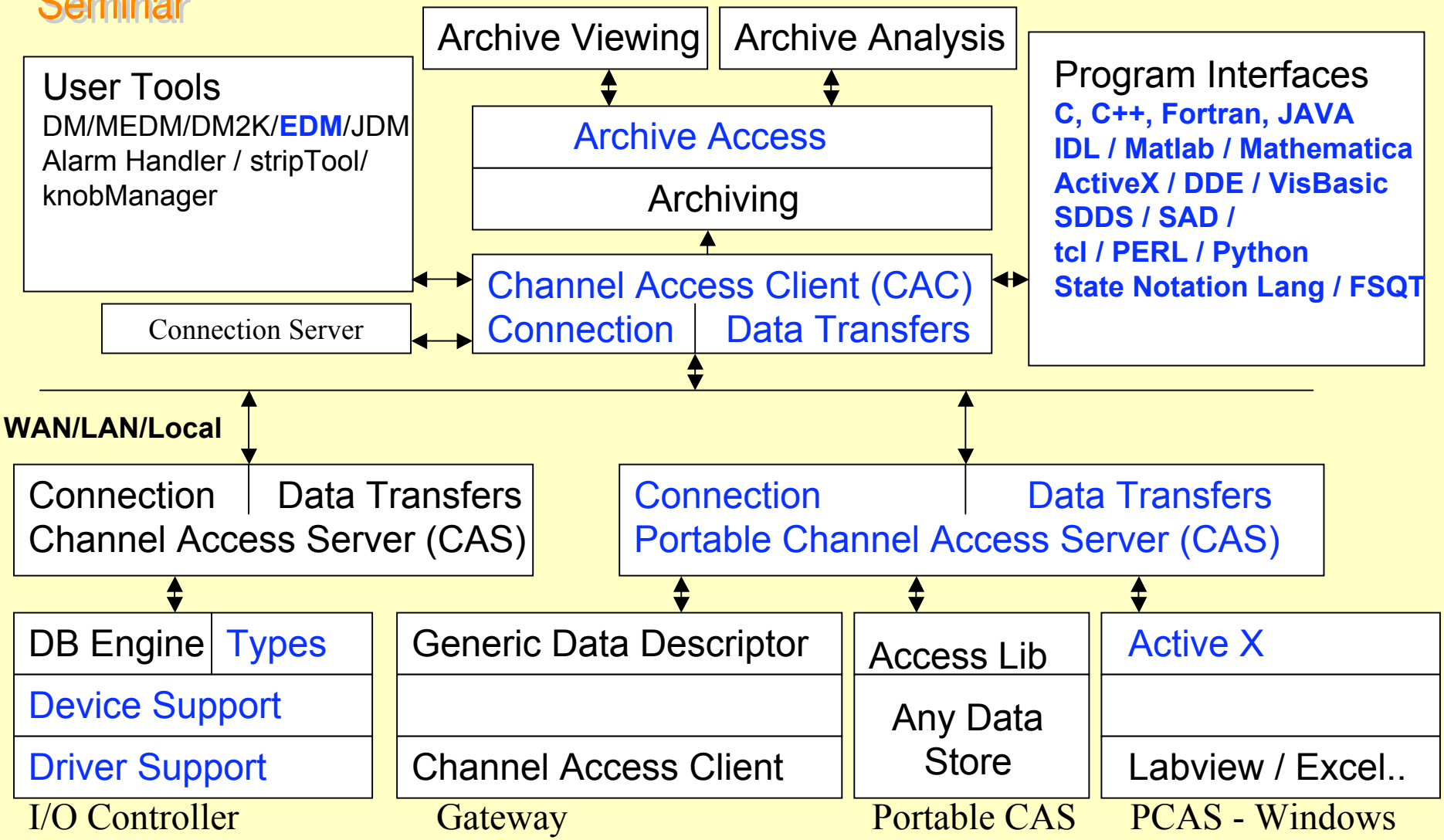
  (But no internal documentation yet)

NSRL EPICS Provides Interfaces at All Levels

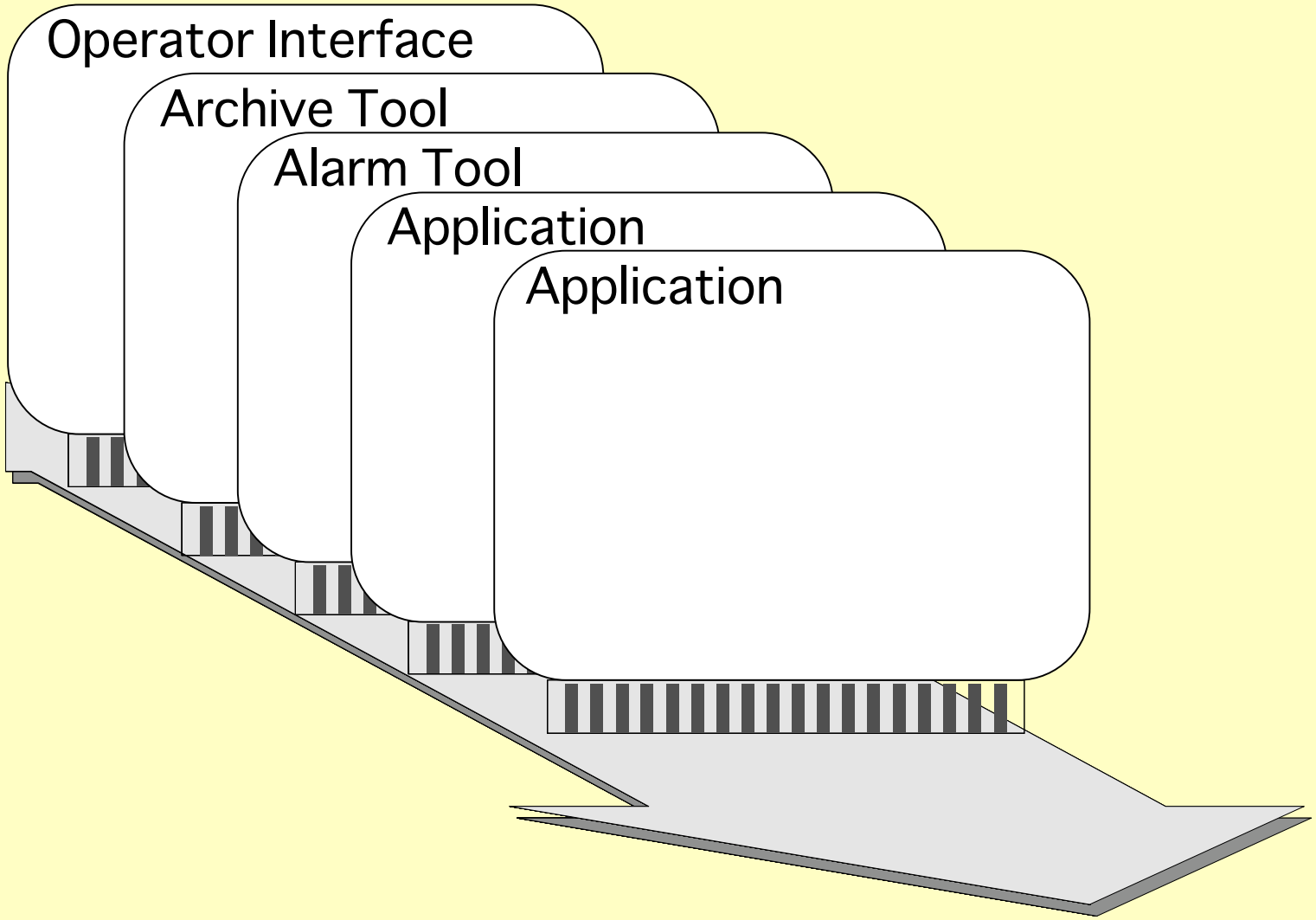
NSRL
EPICS
Seminar

EPICS

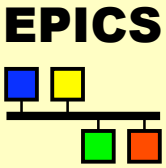




Channel Access : The EPICS Software Bus



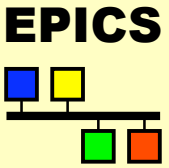
Control Processing Functionalities



- On-demand Synchronous
- On-demand Asynchronous
- Periodical
- On-Change
  - (with value deadband)
  - (with rate limit)
- On-Event (Trigger)
- Cache
- List of Control Points
- Static Information and History using the Same API

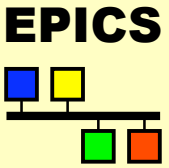
Existent Control Systems Support a Part of Those

What is Channel Access (CA)



- Standardized communication path to a named data in a Channel Access server, usually a field within a record (process variable) in any IOC database.
- This data is a communication path to/from a hardware IO channel or other software.
- Integrates software modules into the control system.
- A callable interface (library of subroutines).

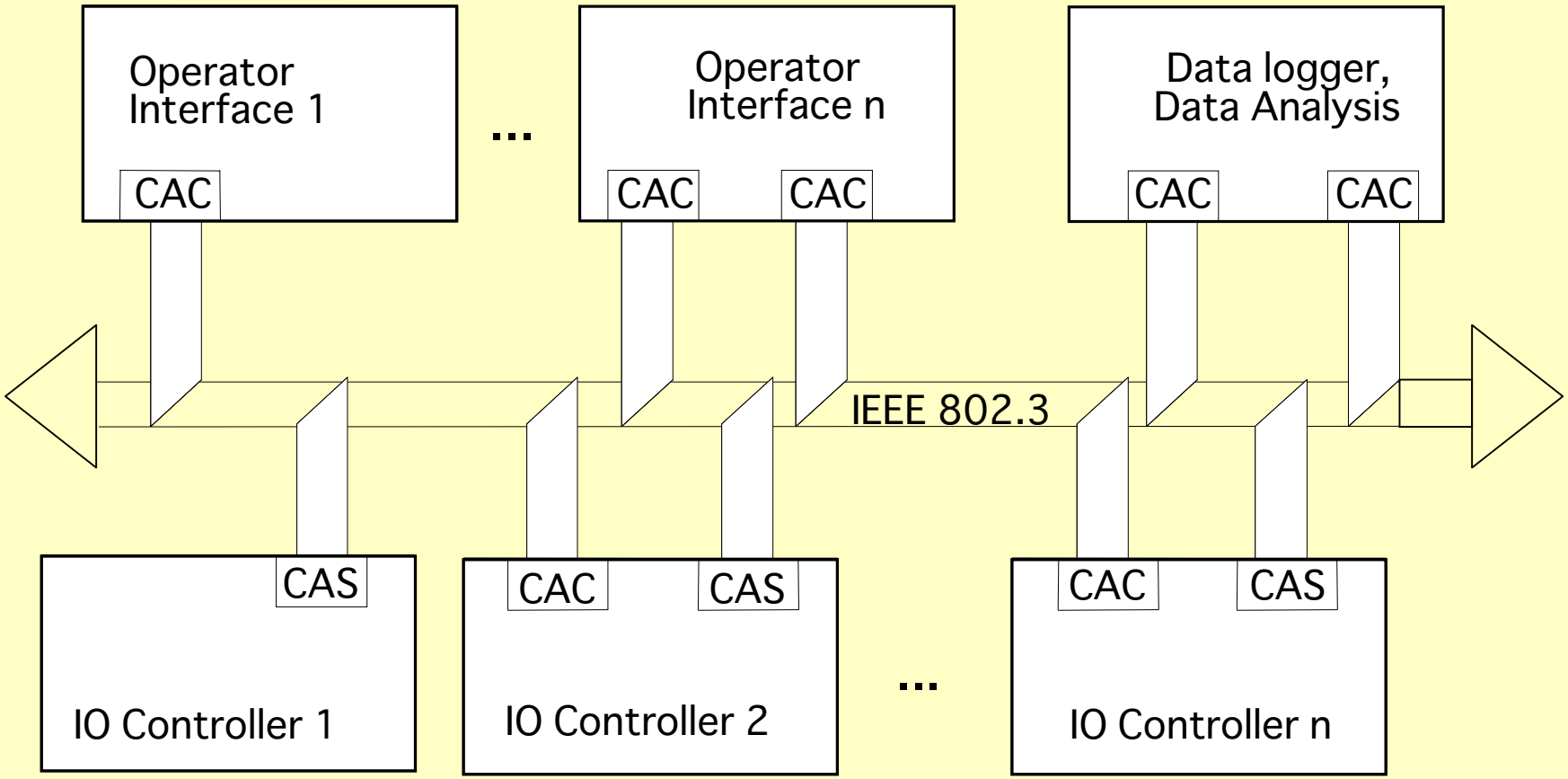
Why Use Channel Access



- Callable interface designed for easy use by casual applications
- Callable interface also designed for use by system software components such as the operator interface, sequencer, and the archiver
- Operating system transparency
- Network transparency
  (access to remote and local channels is identical)
- CPU architecture independence (silent data conversion)
- Isolation from system software changes.
- Efficiency (host IO channels)
- Efficiency (network remote IO channels)

Channel Access network architecture

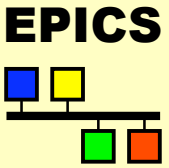
EPICS



CAS Channel Access Server

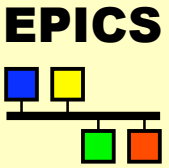
CAC Channel Access Client

Client Server Model



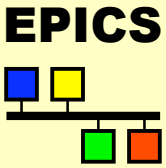
- CA is a network service.
- Clients use the callable interface (link to the CA library).
- Server replicated in each IOC (part of iocCore).
- Clients make requests to the servers across the network.
- CA defines a network protocol.
- Local operations are initiated and performed by the host processor.
- Remote operations are initiated but not performed on the host processor.

Asynchronous Nature of CA



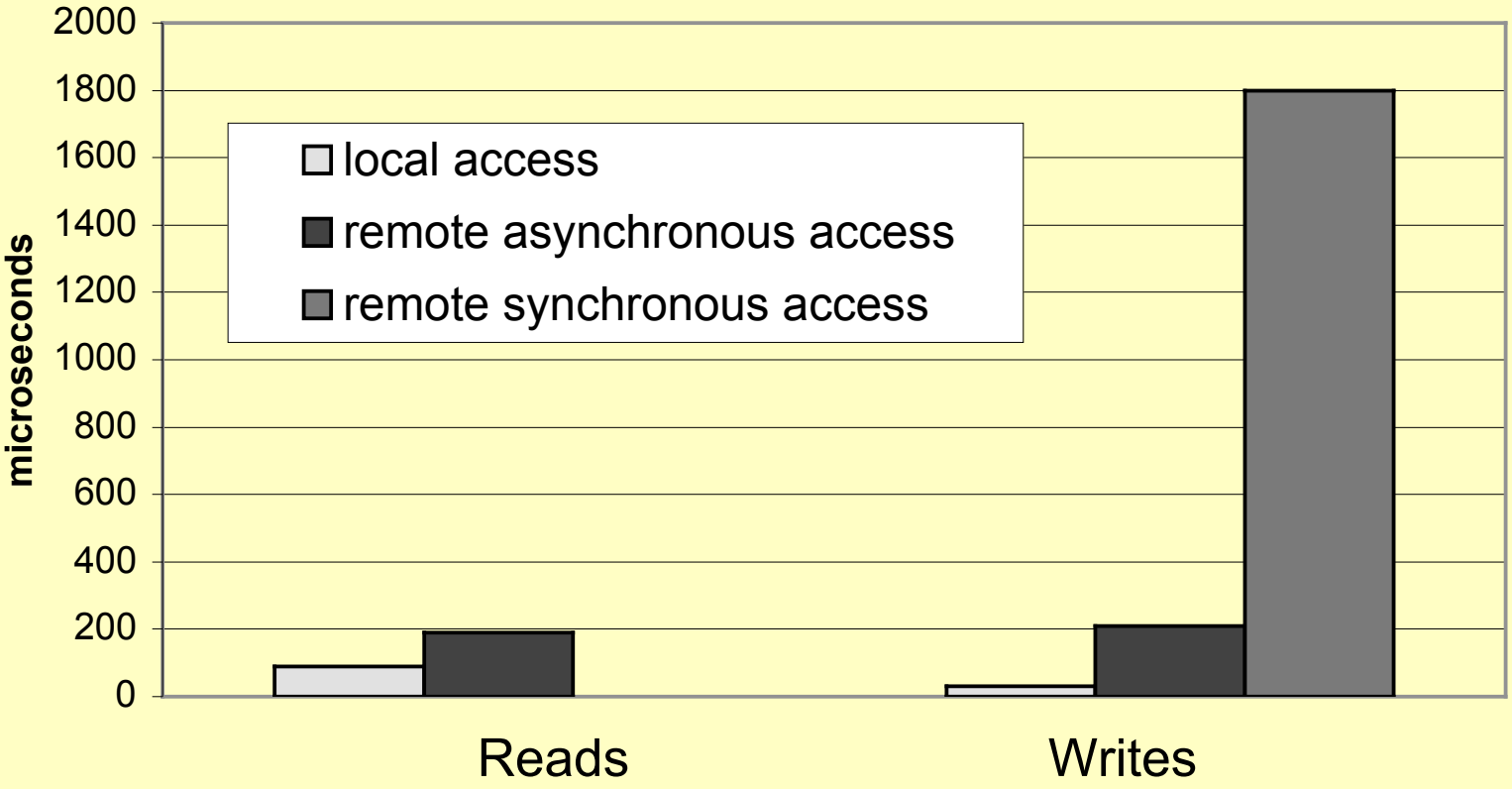
- CA does not wait to gain access to the network prior to returning from each library call.
- Remote operation requests are buffered and sent when the buffer fills or when you ask.
- Data fetched from a remote machine is generally not immediately available.
- With few exceptions values written into your variables by CA should not be referenced until you have synchronized.
- All operations guaranteed to be executed in the order requested.

Why is CA Asynchronous?



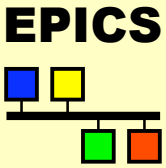
- Combined operations are more efficient when sharing a common resource such as a bus.
- Combined operations can be more efficient when passing through system software layers.
- Sometimes it is useful to perform labor on a local processor while several operations are completing on remote processors.
- The plant is often asynchronous.

CA Performance



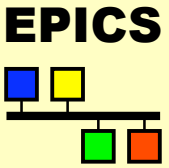
Simple Unconverted I/O Channel Reads and Writes

Methods of Synchronizing



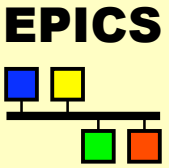
- No data fetches are left outstanding after completing a call to ca_pend_io().
  - Asynchronous replacement for ca_get():
    ca_get()
    ca_pend_io()
- Use a monitor.
- Use fetch with callback.
- Use a synch protocol with a remote program.

Event Propagation



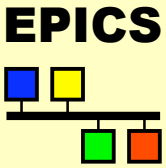
- In any process control system, an application program must be prepared to respond to any one of a number of asynchronous events.
- Events include hardware or software state changes (limit switches, flow indicators, out of range analog channels, software exceptions, etc.)

CA Software Interface to Events



- An event is a significant change in either the value of a field or the condition of the record as a whole.
- Events are placed in a queue and handled in the order that they occurred.
- A channel can be monitored by specifying a handler to be run each time an event occurs.
- CA client applications using events tend to be tree structured.
- Updating the client's local value for a channel this way can save on network traffic since a message soliciting the update need not be sent.

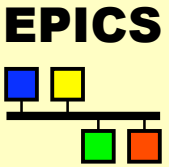
Event Rate Management (Analog Channels)



- The rate at which updates are sent over the network should be minimized by the project engineer within reasonable constraints. This rate is managed by adjusting the channel's deadband and scan rate.

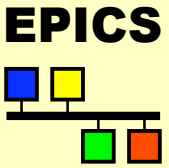
- (R3.15 may provide facility to limit monitor rate (?))

CA Exceptions



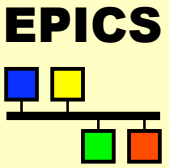
- Since the CA client library does not wait to gain access to the network prior to returning from each call an operation can fail in the server after the library call that initiated it returns.
- Status of these unsuccessful operations are returned from the server to the client's exception handler.
- The default exception handler prints a message for each unsuccessful operation and aborts the client if the condition is severe.
- Operations which fail in the server are nearly always caused by programming errors.

Channel Naming Convention



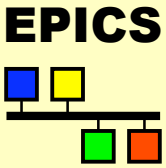
- CA requires that channels have a name.
- The IOC database requires names of the form:
  - <record name>[.<field name>]
    - rfhv01.LOPR”
    - “rfhv01”
- Record names are assigned by project engineer following project naming convention.
- Record field names and purposes are record type specific
- A list of the field names available for each record can be obtained from the database documentation (EPICS Record Reference Manual.)
- If the field name is omitted, the field .VAL is assumed. This field contains a control or read back value.

Native Data Types



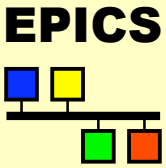
- All channels have a “native” data storage type in the IOC.
- All native data storage types are “atomic”.
- Atomic data types include:
  - integer, floating point, string, enumerated etc.
- When transferring a new value to/from a channel the client program specifies the data format to supply/receive it in. This is often referred to as the external data type.
- The external data type can be different from the native type if conversion is possible.

Compound Data Types



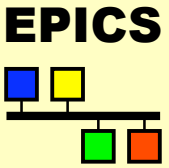
- Compound data types contain a channel value combined with additional status or configuration information.
  - Value + Alarm Status + Timestamp + Drive Limits
- Compound types involve the database record as a whole.
- Compound types can currently only be used with gets and monitors.
- Data types are described in db_access.h. (DBR_XXXX)

Connection Management



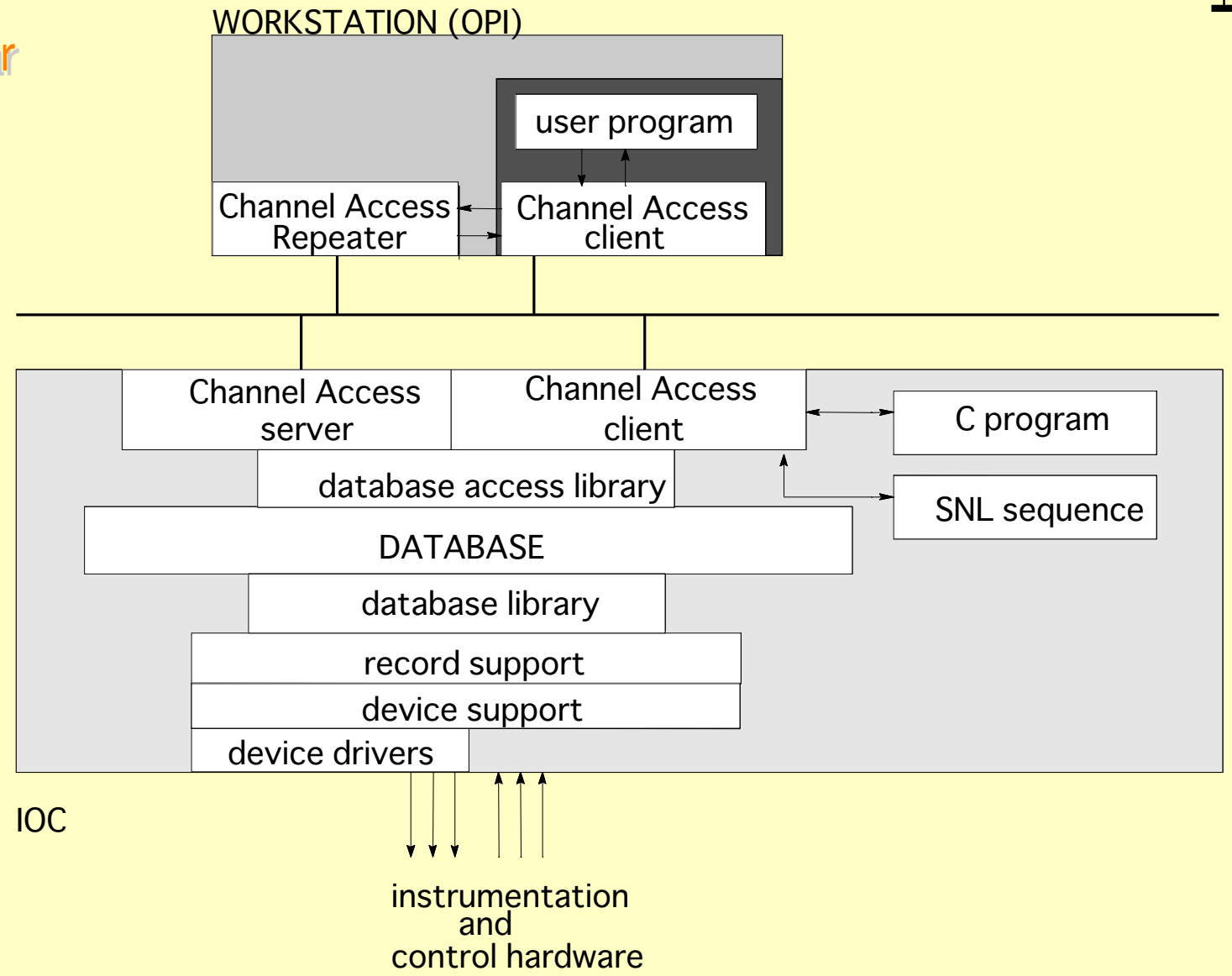
- Network Connections are inherently transient.
- A channel's connection state is either not found, connected, or disconnected.
- CA allows you to specify a handler to be run when a channel's connection state changes.
- Connection requires a server for the channel and a valid network path to the server.
- CA automatically restores connection upon notification from the server.

Software Releases



- IOC core and CA client library EPICS major release number must match, or client will not find server.
- This is due to potential CA protocol redesign between major releases.
- CA protocol is upwardly compatible within a major release.
- When new features are added to the client, older versions of the server won't support them.

EPICS


WORKSTATION (OPI)
user program
Channel Access Repeater
Channel Access client
Channel Access server
Channel Access client
C program
database access library
SNL sequence
DATABASE
database library
record support
device support
device drivers
IOC
instrumentation and control hardware