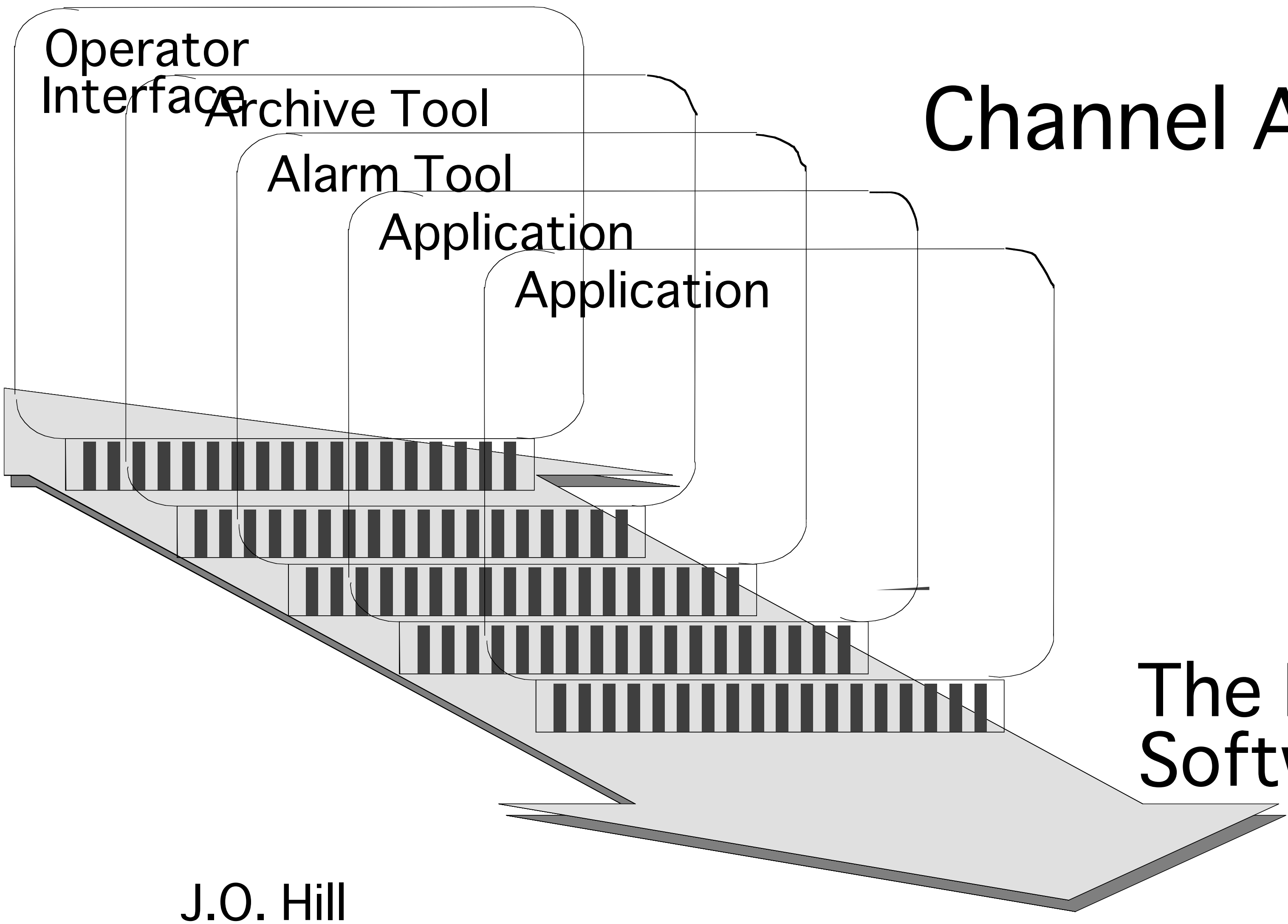
Channel Access:

The EPICS Software Bus



J.O. Hill

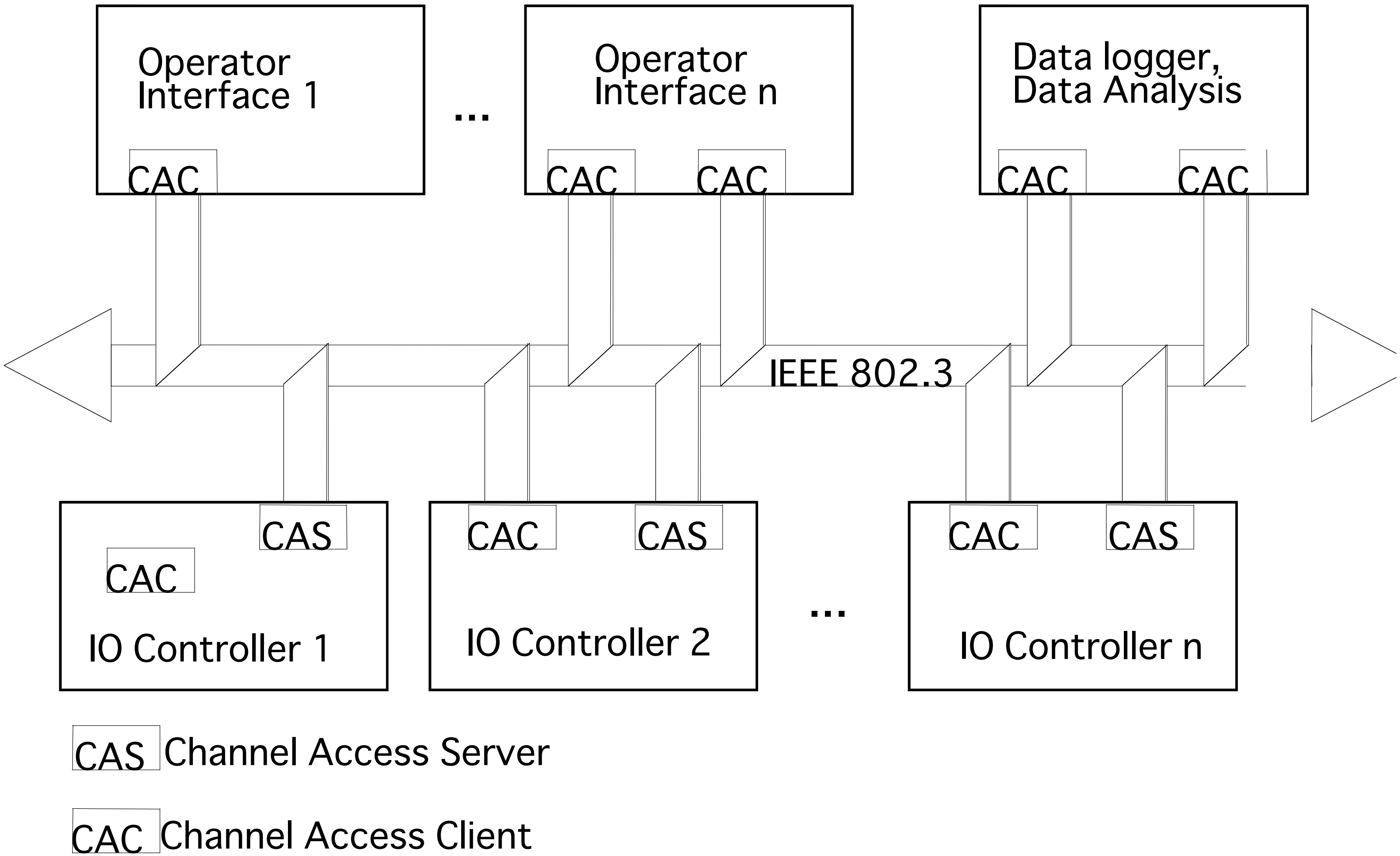
What is Channel Access (CA)

- *Standardized communication path to a field(s) within a record (process variable) in any IOC database(s).*
- *This field is a communication path to an IO channel or 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***
- o***perating system transparency***
- ***Network transparency (access to remote and local channels is identical)***
- C***PU architecture independence (silent data conversion)***
- ***Isolation from system software changes.***
- ***Efficiency (host IO channels)***
- ***Efficiency (network remote IO channels)***

Channel Access network architecture



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.*

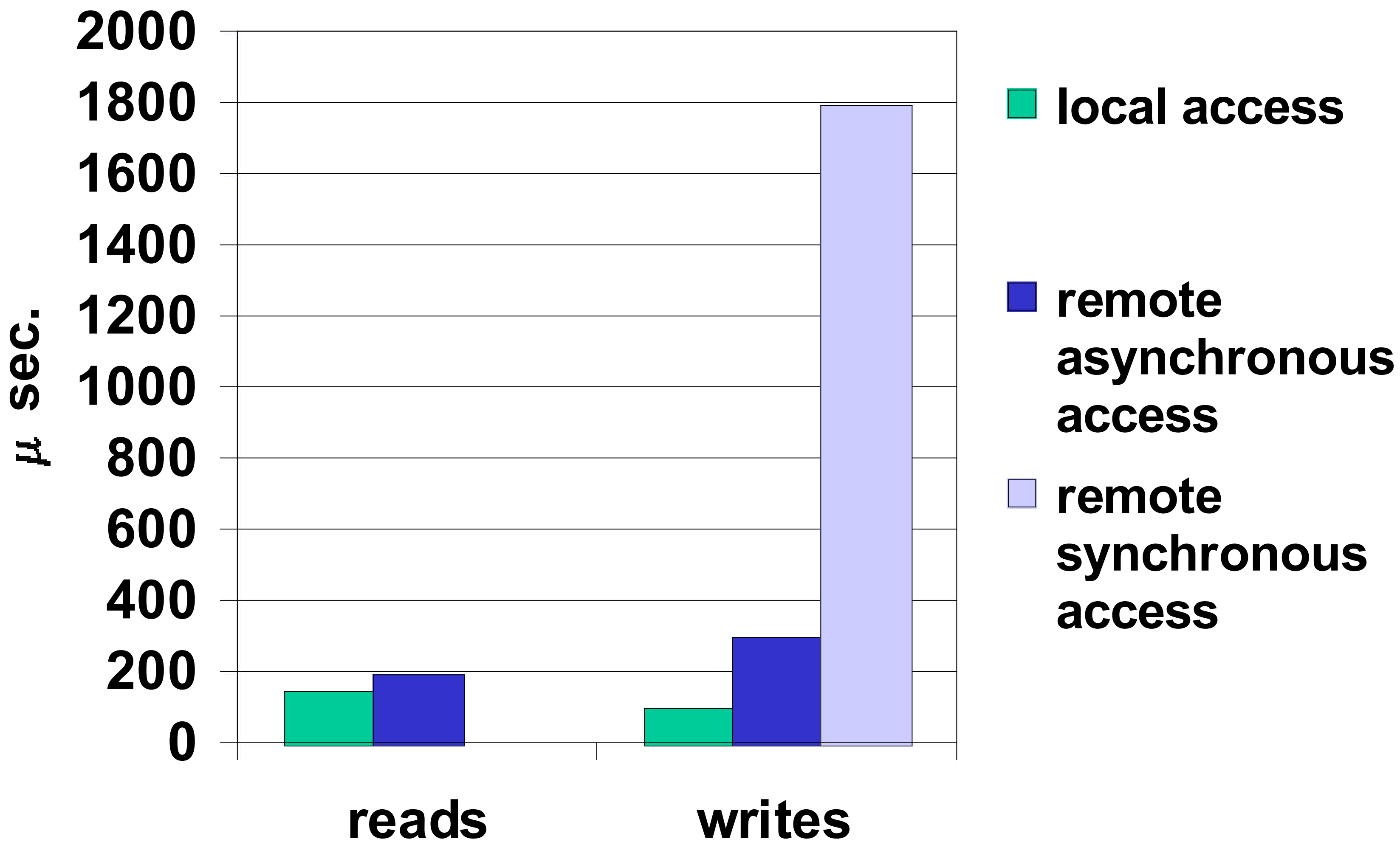
Asynchronous Nature of CA

- *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.*

Channel Access Performance for 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().*
- *A synchronous 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 in DCT.*

CA Function Status

- *All CA functions return standardized constants.*
- *Client installed exception handlers also receive status via these constants.*
- *CA status constants have names of the form "ECA_XXXX".*
- *Status constants contain info about severity.*
- *Status constants contain an index into a table of messages.*

CA Status Testing Examples

- *Checking severity and taking the default action :*
- *int status;*
  status = ca_xxx();
  SEVCHK(status,"this get fails only on Monday");
- *Checking the status yourself and printing the message:*
- *if(status != ECA_XXXX)*
  printf("the error- %s\n",ca_message(ECA_XXXX));

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>]*
ie
"rfhv01.LOPR"
or just
"rfhv01"
- *Record names assigned by project engineer in DCT 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 does 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.*
- *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.*

Channel Identifier

- ***Most CA functions with channel operands require a channel identifier.***
- ***CA provides a routine which opens a new channel given a channel name and returns an identifier.***
- ***CA resolves the channel's network address at runtime.***
  - CA looks for the channel on the host first.
  - If not found locally, CA performs name resolution by broadcasting the channel name on the network. The server on the IOC with the channel replies.
- ***Next you receive a connection event or ca_pend_io() unblocks.***

Establishing a Connection

- *Request a channel identifier followed by ca_pend_io().*
- *Request a channel identifier and specify a connection handler.*
- *Many CA functions return an error when a disconnected channel is specified.*

Difference between Remote and Local Channel Access

- *Status of local operations returned immediately*
- *Status of remote operations returned asynchronously to an exception handler upon failure*
- *To make codes run both remote and local handle error conditions in both places*

Channel Access OS variations

- *On VMS and vxWorks an event handler can preempt the main thread even if it is not in ca_pend_xxx().*
- *On UNIX there is always only one thread of control.*
- *On vxWorks additional threads of execution may join a CA client context by calling ca_import().*
  Note: CA clients should not be invoked directly from the shell. Spawn a separate task instead.

Software Releases

- *IOC core and CA client library EPICS major release number must match, or client will not find sever.*
- *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.*

Include Files

- ***cadef.h***
- ***chid, catype, MACROS,***
  argument passing to event routines
- ***caerr.h***
  - error constant definitions
    error message definitions
    severity extraction/testing MACROS
    message index extraction MACROS
- ***db_access.h***
  - CA standard data types defined

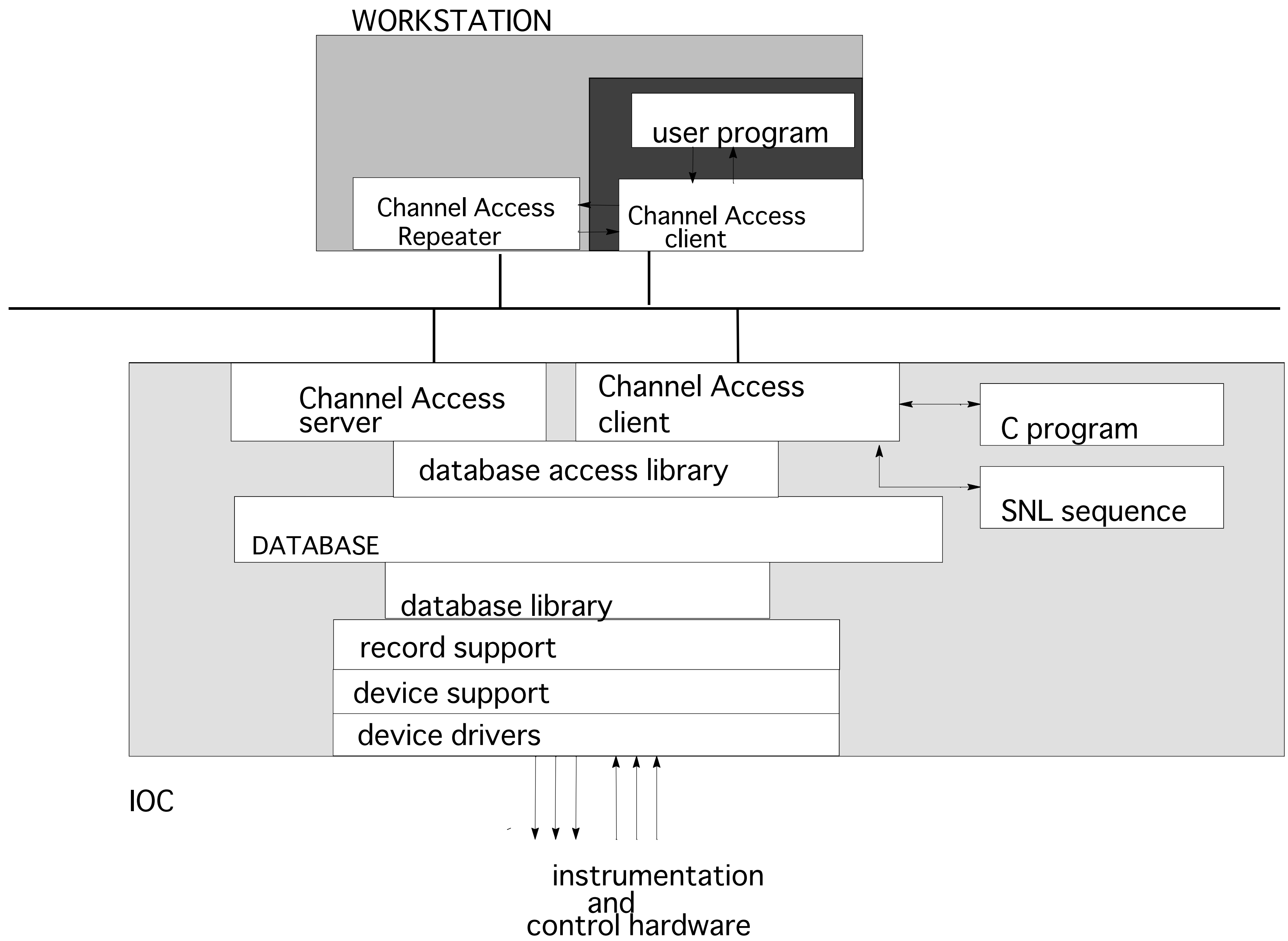
Libraries

- *UNIX:*
  - *libCom.a*
  - *libca.a*
- *vxWorks:*
  load ioc core software as usual then load your application code containing CA_xxxx() calls.
- *VMS:*
  See examples in

  <epics_root>/share/src/ca/BUILD_VMS.COM

CA Changes for R3.12

- *Get features of ca_build_and_connect() disabled. Prefered usage (callable interface) is now ca_search_and_connect().*
- *ca_sg_xxx() routines have been added for synchronizing sets of “puts” and “gets”.*
- *ca_put_callback() has been added for notification of “put” completion.*
- *CA supports database access control. Clients can receive notification when access changes.*


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

CONNECTIONS TO CHANNELS: opening a channel

- opening a channel produces a chid, which is the `handle' for subsequent interactions with the channel
- the chid is produced immediately, before any attempt actually to connect to the channel
- ca_pend_io() is used for searches which don't specify a connection handler

```
chid      pCh1, pCh2;
stat = ca_search("POWER:BASE", &pCh1);
SEVCHK(stat, "ca_search for pCh1");
stat = ca_search("POWER:LAG30", &pCh2);
SEVCHK(stat, "ca_search for pCh2");
stat = ca_pend_io(2.);
SEVCHK(stat, "pend for searches");
```

CONNECTIONS TO CHANNELS: closing a channel

- closing a channel wraps up all pending activity for it, including canceling its monitor (if any) and its connection handler (if any)
- Channel Access automatically closes channels at program termination, but explicit closes are good form

```
stat = ca_clear_channel(pCh1);
SEVCHK(stat, "ca_clear_channel for pCh1");
stat = ca_clear_channel(pCh2);
SEVCHK(stat, "ca_clear_channel for pCh2");
```