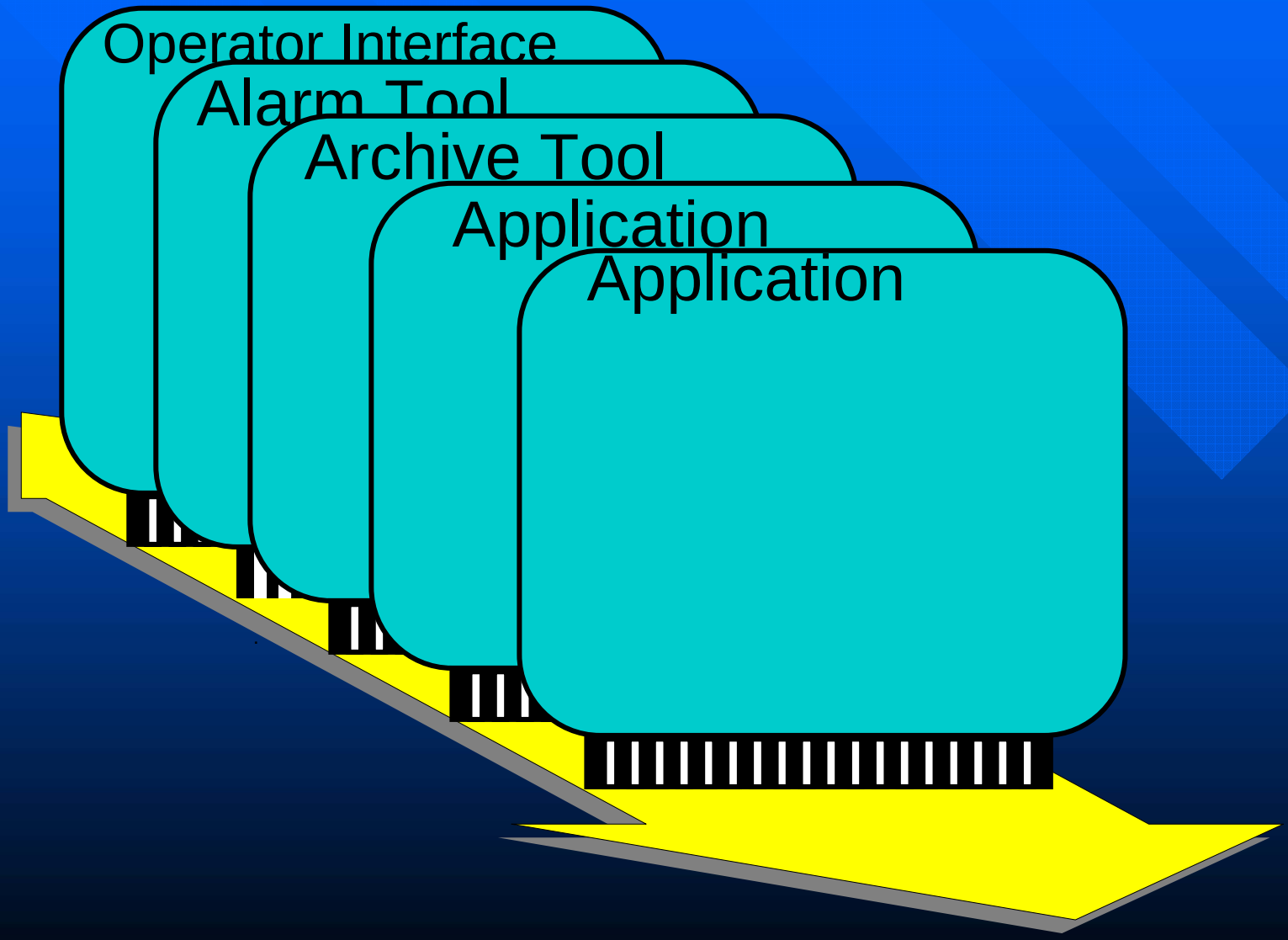
# Channel Access Concepts

Jeff Hill

# Channel Access:
# The EPICS Software Bus



# What is Channel Access (CA)

- A standardized communication path to a process variables
  - To communicate with an IO channel or software
  - Frequently a process variable is a field within a record within a database within an IOC
- An integrating layer for software modules in the control system
- A callable interface
  - Libraries of functions (subroutines)
- A network protocol

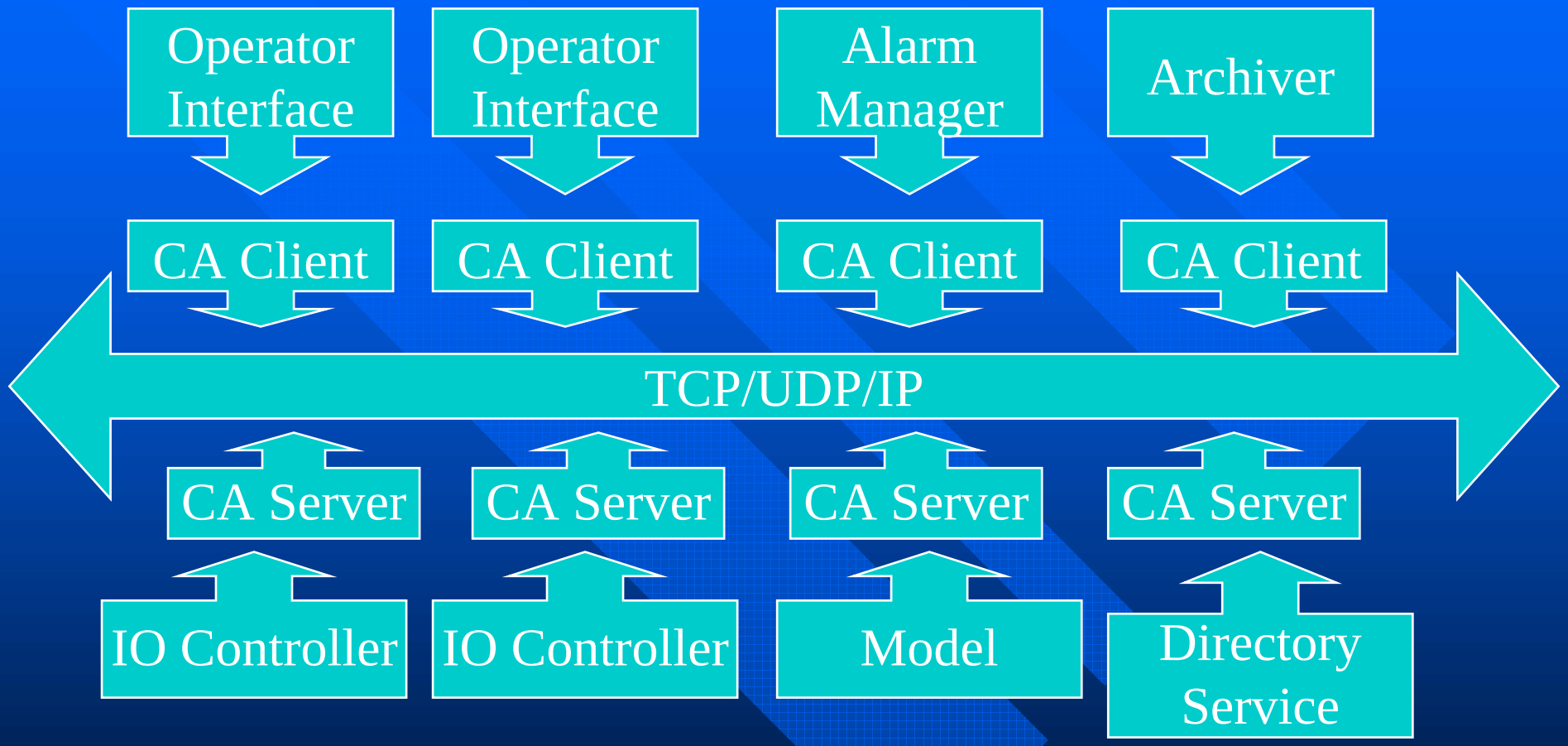
# Why Use Channel Access

- Callable interface designed for
  - Easy use by casual applications
  - More complex uses required by system software components for
    - » operator interaction, alarm management, data archival …
- Operating System and CPU Architecture Independence
  - Silent data conversion
  - Portable interfaces for asynchronous interactions
- Network Transparency
  - Identical access to remote and local channels
- Isolation from system software changes
- Efficiency
  - Host local IO channels
  - Network remote IO channels

# Client Server

- Clients
  - Instantiated when process variables must be viewed or manipulated
  - Connect full duplex byte-serial virtual circuits to servers
  - Initiate requests
    - » Reads (gets), writes (puts), subscriptions (monitors) etc
  - Receive and process corresponding responses
- Servers
  - Instantiated where process variables need to be published
  - Receive requests, process them, and send responses

# Client Server Distributed Architecture



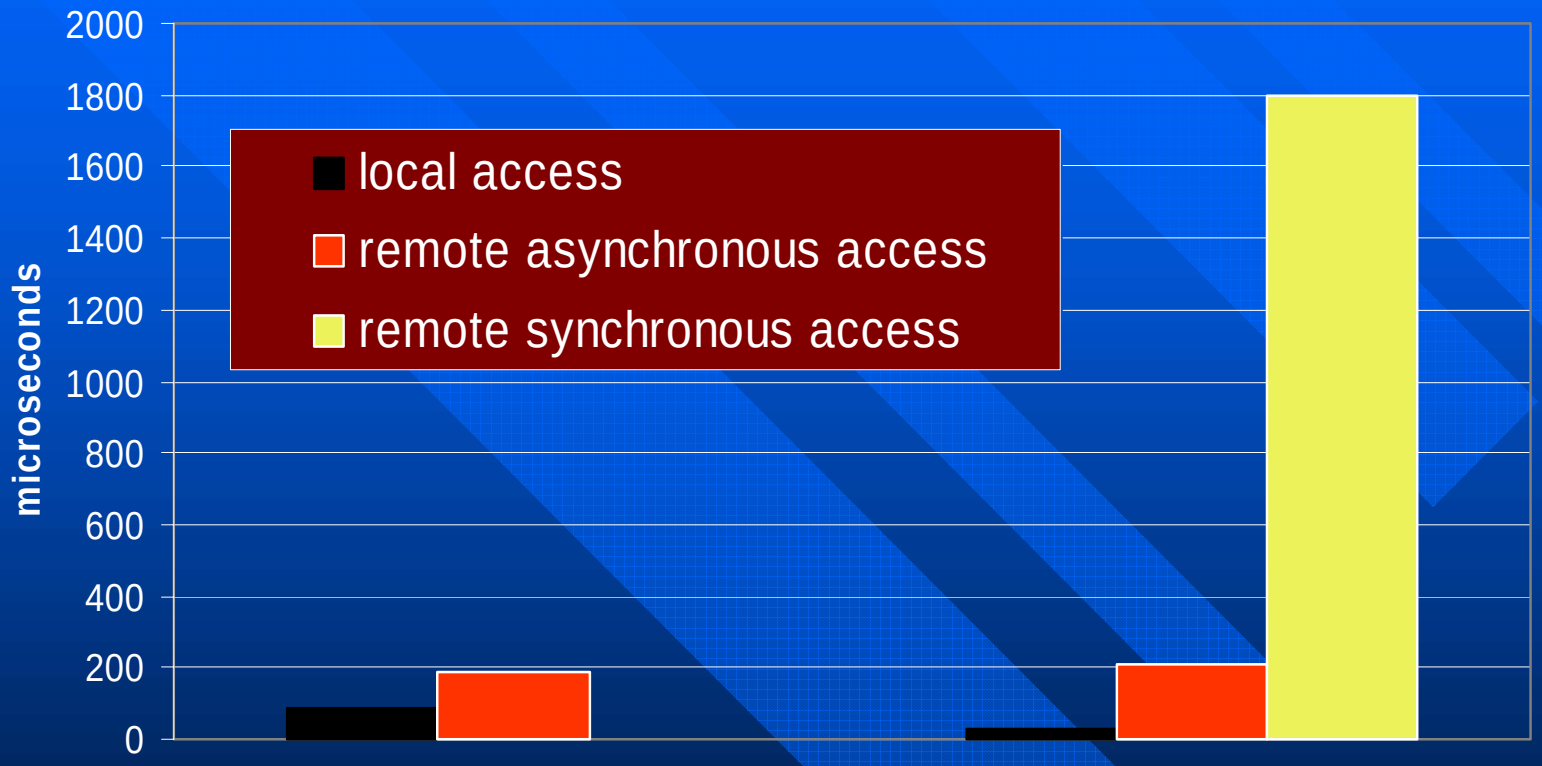
# Asynchronous Nature of CA

- CA does *not t*ransmit a request and then wait for its response to arrive within each library call
- Requests are queued from within library calls
  - Requests are transmitted only when
    - » You ask, or the queue reaches maximum capacity
- Requests are serialized
  - The client library, virtual circuit, and server library all work together guaranteeing that requests are executed in the server preserving the order in which they were received in the client library
- Data fetched is generally not immediately available

# Why is CA Asynchronous?

- Combined operations permit more efficient use of shared hardware resources
  - buses, networks, interrupts, etc
    - » Reduction of collisions, improved packet packing, …
- Combined operations can be more efficient when passing through software layers
  - Operating systems, libraries
    - » Reduction of context switches, mutual exclusion locks, …
- Parallel processing
  - Sometimes it is useful to perform labor on a local processor while several operations are completing on remote processors
- Process control is asynchronous by nature

# Performance




Round Trip Synchronization: A Bottle Neck

# Methods of Synchronizing

- Callback based response notification
- Some simple programs prefer for the library to directly update their variables
  - Data fetches of this type are guaranteed to be complete after successful completion status is returned from completion blocking function

# Events

- In a process control system, application programs 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

# Subscriptions

- Clients subscribe for updates
  - Clients specify triggering event – currently one of
    - » Value change (default dead band)
    - » Value change (archive dead band)
    - » Alarm change
  - Clients specify the content of the data capsule that is to be sent with each update
- Monitoring the server's state this way can reduce network load
  - Request message soliciting update need not be sent
  - Updates need only be sent on change

# Event Delivery

- Events are serialized
  - A per-client queue isolates high priority event posting agent from lower priority server agent
    - » When client event consumption is slower than event production
      - Quota limits maximum queue size
        - Last event on queue is replaced
  - Events are delivered in the order of occurrence
- CA client applications using subscriptions
  - Use callback based notification
  - Tend to be state change scheduled – with a short code stub for each state change that is being monitored

# Event Rate Management

- The rate at which updates are sent can be managed by the project engineer
  - By limiting event production rates
    - » By adjusting the scan rates of function blocks (records)
    - » By adjusting hardware interrupt rates
  - By specifying the magnitude of a significant change for analog channels
    - » Adjusting the dead bands in the function blocks (records)

# Failure Notification

- Requested operation might fail in the server
  - Long after the library call that initiated it returned
    - » Consequence of asynchronous client interface
- Client Application Notification
  - When possible, the callback function that the client application supplied with the request is used for notification
  - Otherwise, a global exception callback handler is used
- Operations that fail in the server are nearly always caused by by poorly formed requests – by programming errors

# Default Global Exception Handler

- Prints a diagnostic message
- Aborts the client program if the failure condition is severe

# Channel Names

- CA requires only that channels have a name
- Process variable names in the IOC's function block database
  - <record name>[.<field name>]
    - » rfhv01.LOPR"
    - » "rfhv01"
      - If the field name is omitted, the field "VAL" is assumed
        - This field contains a control or read back value
  - Field names and purposes are record type specific
    - » Details in "Record Reference Manual"
  - Record names are assigned by project engineer

# Data Types

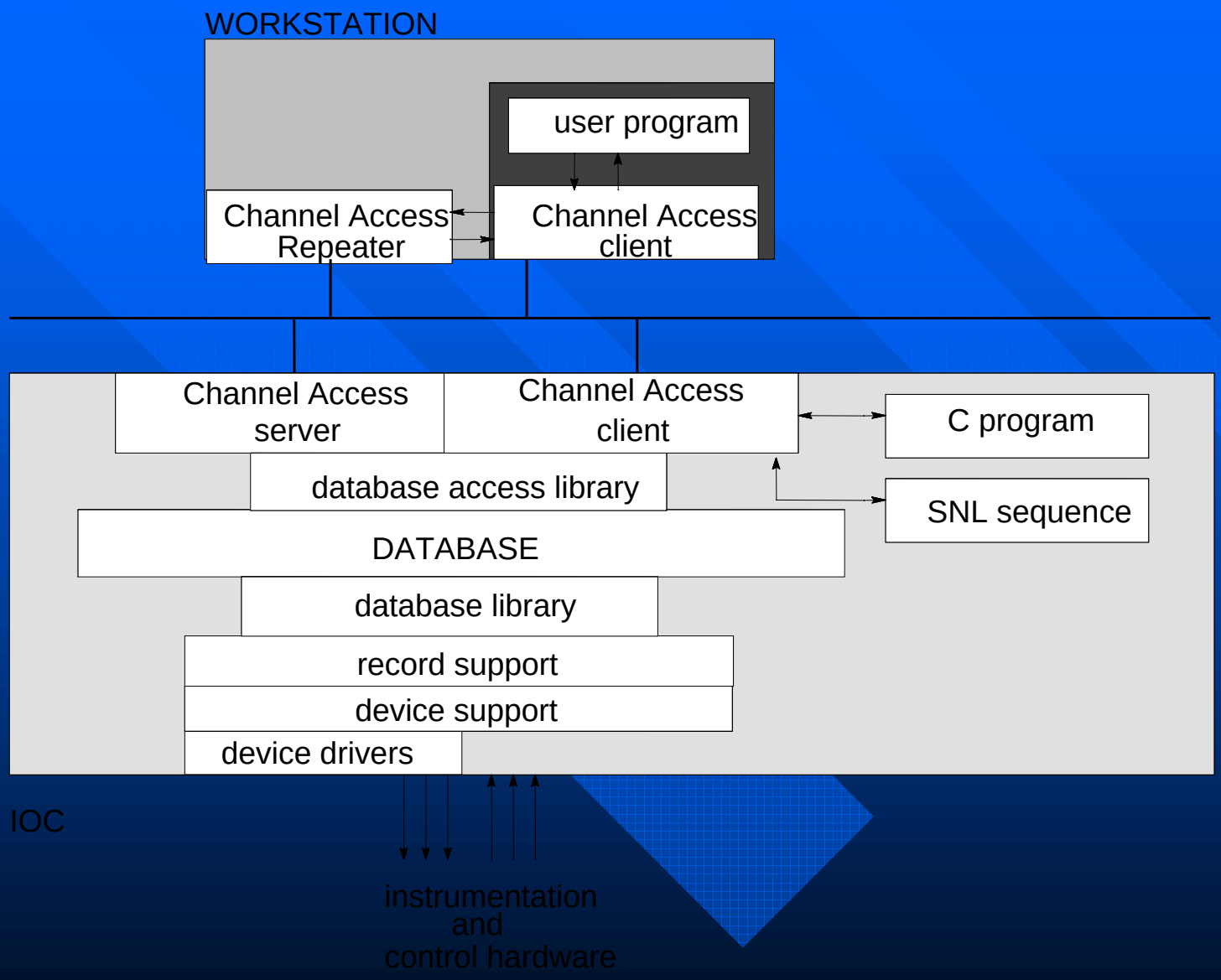
- All channels have “native” data storage type in the server
  - Currently, all native data storage types are “atomic”.
  - The atomic primitive data types include:
    - » integer, floating point, string, enumerated etc.
- When transferring a new value to/from a channel the client program specifies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

# Compound Data Types

- Channel value is combined with additional meta data (properties) of the channel
  - Value + Alarm Status + Timestamp + Drive Limits
- Involve the database record as a whole
- Used with gets and subscriptions, but very rarely with puts
- Data types are described in
  - In the “CA Reference Manual”
  - In db_access.h

# Connection Management

- Network Connections are inherently transient
- A channel's connection state is either not found, connected, or disconnected
- Connection requires a server for the channel and a valid network path to the server
- The client library is responsible for
  - Initially connecting the channel
  - Reconnecting the channel should there be a loss of service
- A callback function to be used for connection state change notification can be specified when a channel is created


WORKSTATION
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