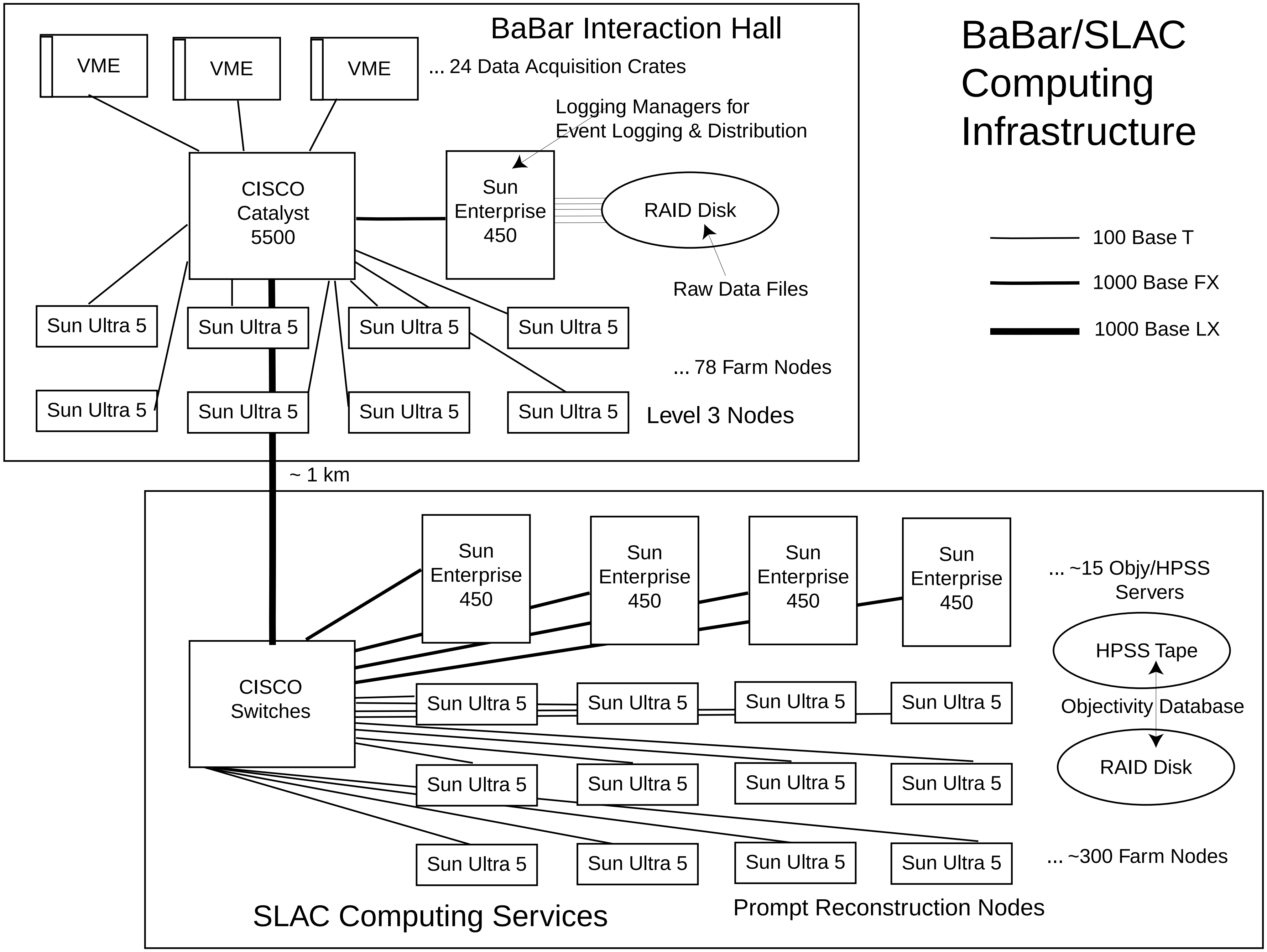
Event Logging and Distribution for BaBar Online System

Sridhara Dasu
University of Wisconsin-Madison
T. Glanzman, T. Pavel
Stanford Linear Accelerator Center

- **BaBar Computer Farm and Tasks Overview**
- **Logging Manager**
  - Event logging after Level-3 trigger
  - Event distribution for Prompt Reconstruction
- **Software Technologies**
  - Object Oriented Design
  - TCP/IP Socket Communication
  - CORBA Controls/Monitoring
- **Performance**
  - Event Logging
  - Event Distribution
- **Technology Evaluation**

BaBar Farm Overview

BaBar/SLAC Computing Infrastructure



Online Tasks Overview

- **Online Dataflow** (Event Dataflow Aspects)
  - Acquisition from frontend
  - Event building (<event size> = 32 kB, <event rate> < 2 kHz)
- **Online Event Processing**
  - Level-3
    - Reduce dataflow rate from 2 kHz to 100 Hz
    - Portions of reconstruction code run
  - Event data quality assurance
  - Framework for Calibration
    - Electronics, pulser and source
  - Log raw data after level-3 to intermediate store
- **Online Prompt Reconstruction**
  - Distribute events from "disk buffer" to reconstruction processes
  - Support "full" reconstruction of good events (<reco event size> = ~200 kB, <input rate> = 100 Hz)
    - "Physics" based calibration
    - Event data based alignment
    - Event type (physics) tagging
  - Ensure reconstruction stores raw+reco data to the persistent store (Objectivity Object Database)

This Talk

Logging Manager

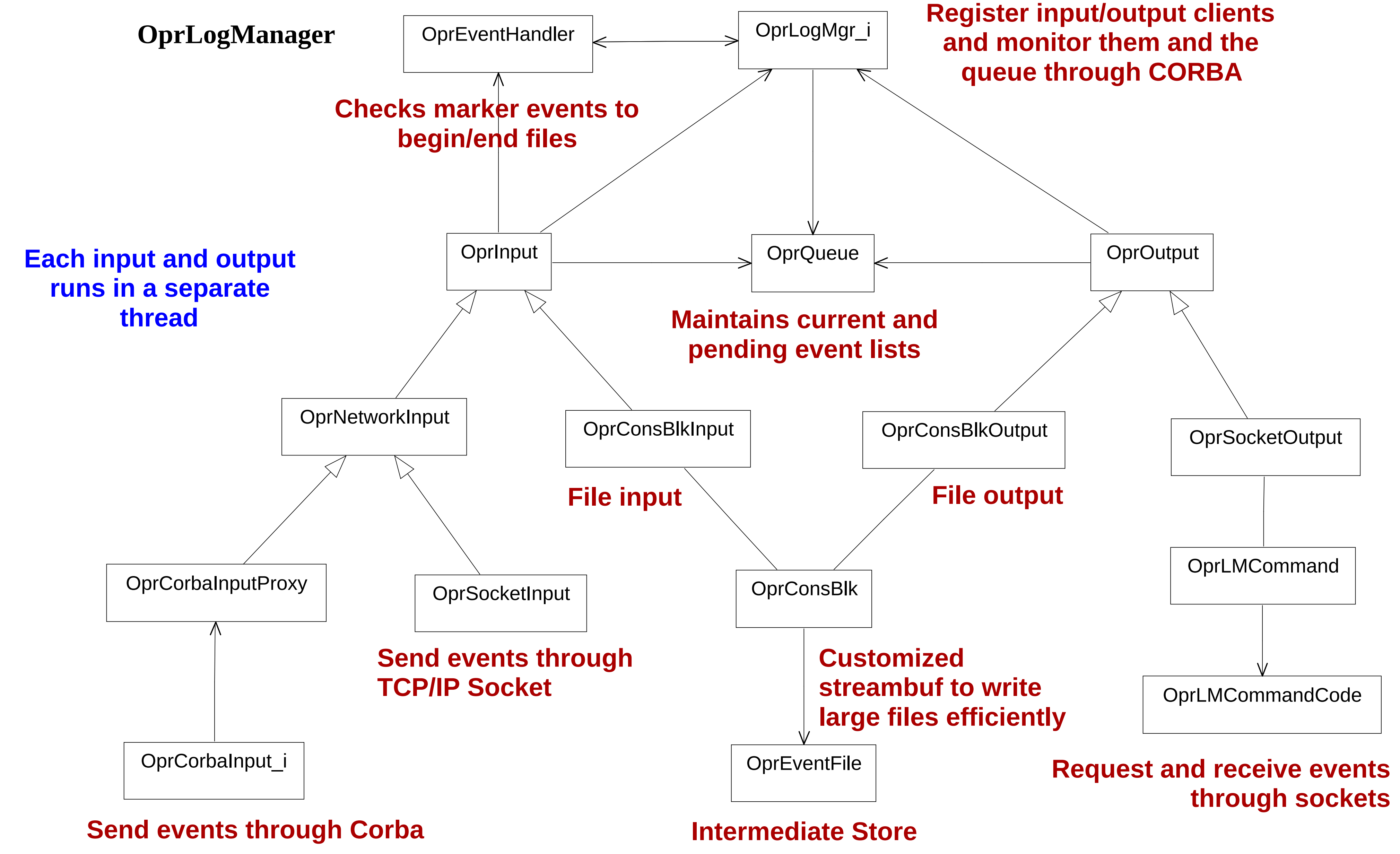
- **Event Logging and Distribution**
  - Tasks are similar
    - Event logging: Several outputs with one output
    - Event distribution: One input with several outputs
    - A several input - several output program can serve both tasks
  - Input and Output types
    - Network IO
      - TCP sockets (moderate performance - reliable connections)
      - CORBA (lower performance - but more convenient for object sharing)
    - File IO
- **BaBar choice: Build “Logging Manager” that tackles both tasks**
  - Use TCP for data transport
  - Use CORBA for monitor and controls
  - Use OO design for building this dual use program
  - Use multi-threading for high performance

Technology selection

- Hardware
  - Event bandwidth: 30-50 kB * 100 Hz = 3-5 MB/s
  - Server (Collects data for logging and distribute it: 6-10 MB/s)
    - Gigabit ethernet card
    - RAID disk with 2 x Ultra wide SCSI controllers
    - 4 processor machine (Sun Enterprise 450)
  - Clients (Level 3 / Online processing / Prompt reconstruction)
    - Processing times not fully optimized yet
    - ~200 Ultra SPARC 333 MHz CPUs
    - 100BaseT
- Software
  - UML for Object oriented design
    - Rational Rose, but not code generation
  - Programming environment
    - Usual unix tools in Solaris environment
    - Rogue Wave STL
  - TCP/IP socket communication for data
    - ACE object oriented wrappers (Doug Schmidt et al., CS, Washington Univ.)
  - Monitor/Control
    - CORBA - TAO implementation based on ACE (Doug Schmidt et al.)

High quality free software

Logging Manager

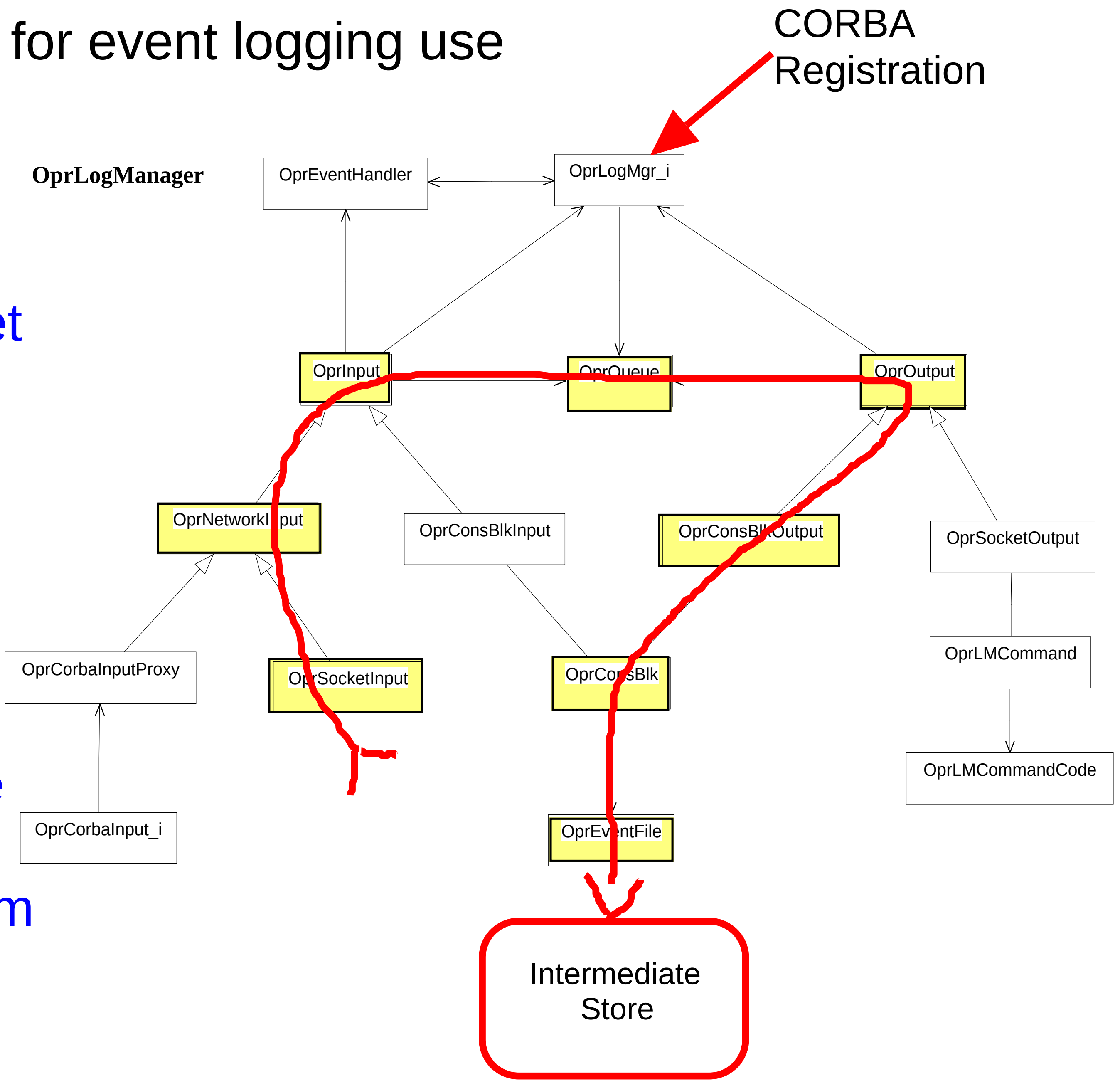


OprLogManager
OprEventHandler
OprLogMgr_i
Register input/output clients and monitor them and the queue through CORBA
Checks marker events to begin/end files
OprInput
OprQueue
OprOutput
Each input and output runs in a separate thread
Maintains current and pending event lists
OprNetworkInput
OprConsBlkInput
OprConsBlkOutput
OprSocketOutput
File input
File output
OprCorbaInputProxy
OprSocketInput
OprConsBlk
OprLMCommand
Send events through TCP/IP Socket
Customized streambuf to write large files efficiently
OprLMCommandCode
OprCorbaInput_i
OprEventFile
Request and receive events through sockets
Send events through Corba
Intermediate Store

Socket Input

for event logging use

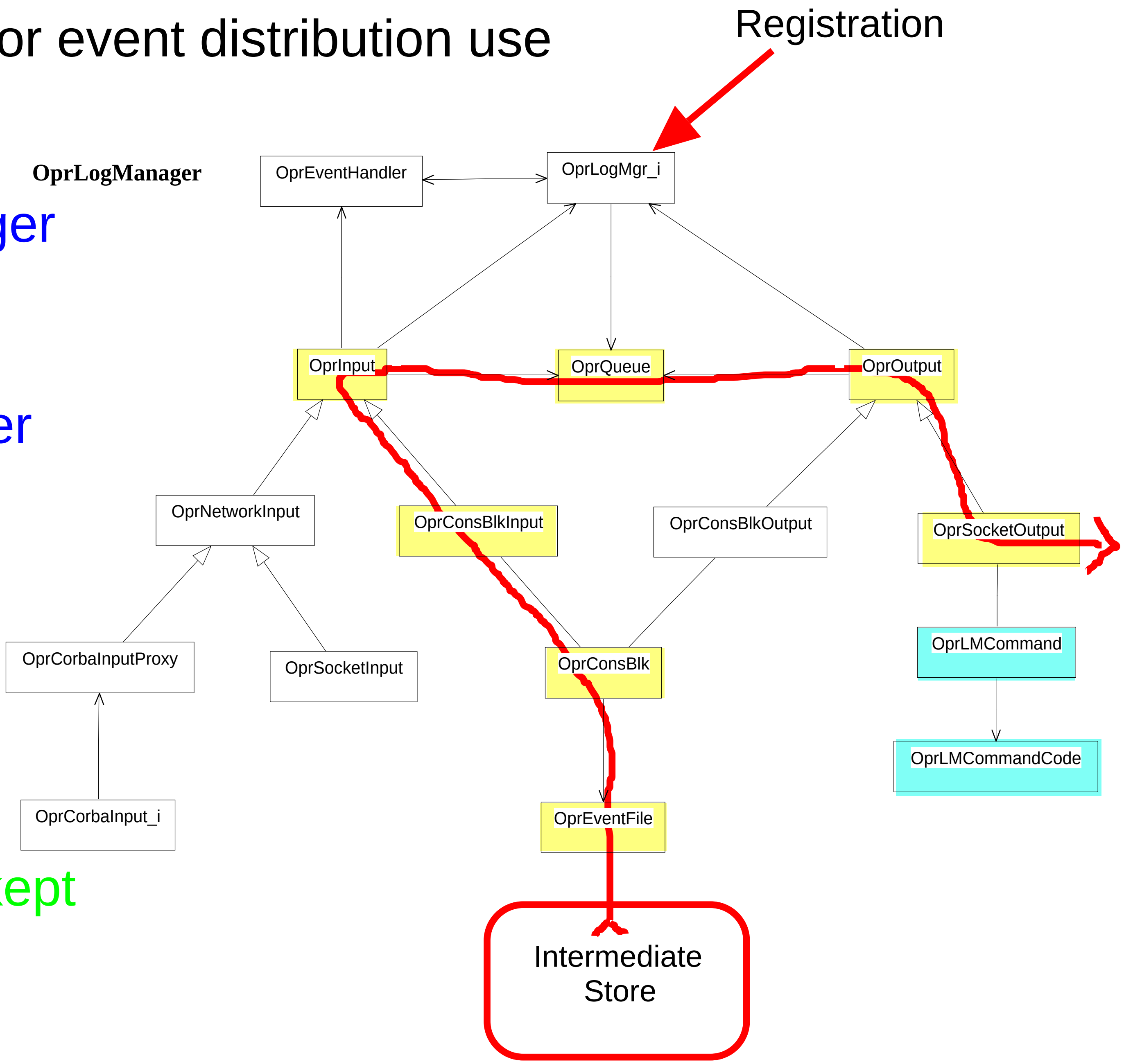
- **Level-3 processes**
  - Use CORBA registerSocketInput
  - Obtains EventSocket
  - Uses socket communication with Logging Manager
- **A run control process**
  - Uses CORBA registerFileOutput
  - Initiates Open/Close files through special events in data stream (BeginRun/EndRun)



Socket Output

for event distribution use

- **OprManager**
  - Start Logging Manager
  - Register file input
  - Get resources from Global Farm Manager
  - Create OprDaemon processes
- **OprDaemon**
  - Register for socket output
  - Get events
    - Socket protocol kept to bare minimum



Prompt Reconstruction Daemon

- **Multi-threaded implementation**
- **CORBA client of Logging Manager**
    - Registers itself for output
- **Socket client of LM**
    - Establishes socket communication with LM
    - Gets events from LM
    - Acknowledges “done” status when event is reconstructed
- **Manages PR framework**
    - Sets up shared memory
    - Spawns out a “PR framework” process and hands it shm key
    - Monitors PR framework progress
        - Event done status communication through Unix pipe
    - Recreates framework on crashes
- **CORBA server**
    - Provides its status
    - Provides PR Framework status

PR Framework

- **Single threaded "offline" program**
  - Supports running reconstruction software in online world
  - Enables calculation of calibration constants and alignment data
  - Writes PR Framework status to shared memory for monitoring
- **Event handling**
  - Reads events from the shared memory
  - Drives the offline reconstruction modules
  - When reconstructed data is logged to the Objectivity persistent store, communicates to the PR Daemon that the event is done
- **Marker handling**
  - Logging Manager drives framework states using marker events
    - Sets up histogramming for begin ConsBlk marker
    - Computes statistics ... for end ConsBlk marker
    - Ensures histogram storing for store marker
    - A "finalize" node does calibration and alignment calculations

Status

- **Logging Manager**
  - Design & first production release (Jan 1999)
    - 0.5 Man year effort
  - In production use for both event logging & distribution
    - Is robust - event logging LM runs for weeks
  - Maintenance and changes
    - Changes to either task do not impact the other
      - OO design helped in a big way
      - Core code did not change although we changed event and marker distribution strategy many times.
- **PR Daemon**
  - Quick design implemented in 2 months. Works!
  - Negligible maintenance load.
- **PR Framework**
  - Full BaBar reconstruction program - many developers - complex interaction with Objectivity
  - Customized PR interaction well isolated - works!

Performance

Use Case	Server	Clients	Typical use (Requirement)	Stess test (Measurement)
Event Logging	Sun Enterprise 450 1000BaseFX	32 Sun Ultra-5 100BaseT	100 Hz 30-50 kB 3-5 MB/s	1 kHz 35 kB 35 MB/s
CORBA Event Logging	Sun Enterprise 450 1000BaseFX	Sun Sparcs 4 100BaseT 3 10BaseT	-	100 Hz 50 kB 5 MB/s
PR Event Distribution	Sun Enterprise 1000BaseFX	100-200 Sun Ultra-5 100BaseT	100 Hz 30-50 kB 3-5 MB/s	400 Hz 35 kB 14 MB/s

- **Event Logging**
  - Typical use: 100 Hz logging of 30-50 kB events after Level-3 from 32 nodes
  - Stress test: Open Level-3
- **Event Distribution**
  - Typical use: Distribution to many nodes running full reconstruction and write to objectivity. Large transaction time for objectivity results in many pending events resulting in large memory buffer for logging manager.
    - Current use: 100 nodes, 180s mean transaction time
    - Future use: 200 nodes (140 Hz steady-state operation verified)
  - Stress test: Distribution to 200 nodes with dummy framework

Technology assessment/Summary

- **Object oriented design**
  - Enabled flexible design for use in two tasks
  - Enabled quick implementation
  - Is enabling easy maintenance and feature changes
- **Solaris platform**
  - Good performance - multi-thread scheduling and network
- **STL**
  - Very useful - must use where one can
- **ACE/TAO**
  - CORBA good for complex communication
  - ACE wrappers help write OO TCP/IP socket code with little effort
  - ACE wrappers include multi-threading support
  - Cross platform - although not used works on Linux, OSF, NT ...
  - Good performance - Great price ($0)
- **Did we make a good choice?**
  - YES!