Pure LabVIEW Implementation of EPICS Communication Protocol

Alexander Zhukov

Spallation Neutron Source, ORNL

NIWeek 2012

Oak Ridge National Laboratory

What is EPICS

- **Experimental Physics and Industrial Control System (EPICS)** is a set of Open Source software tools, libraries and applications developed collaboratively and used worldwide to create distributed soft real-time control systems for scientific instruments such as a particle accelerators, telescopes and other large scientific experiments.
- **Major collaborators**
  - ANL
  - LANL
  - ORNL (SNS)
  - SLAC (SSRL, LCLS)
  - JLAB (CEBAF)
  - DESY
  - BESSY
  - PSI (SLS)
  - KEK
- **Runs on variety of hardware/OS (Linux, VxWorks, Windows, Mac, RTEMS…)**

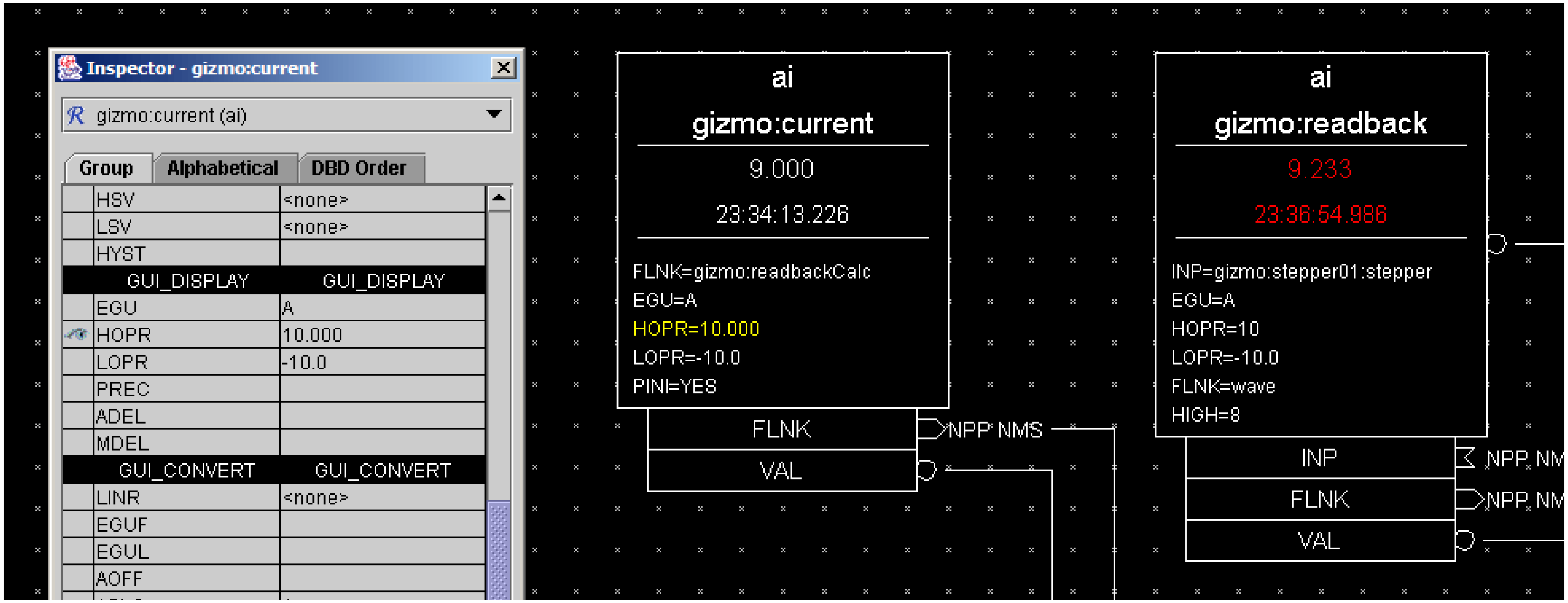
How EPICS works

- **A network based fully distributed client/server model**
- **Client and server use Channel Access (CA) protocol to communicate**
- **Everything spins around process variables (PV) – an entity similar to LabVIEW network shared variable**
  - A server (Input Output Controller – IOC) publishes data by updating PVs
  - PV corresponds to some value (measurement, setting, status etc)
  - Every PV has unique name
  - Client has ways to read PVs and update them over network
  - Client can subscribe to particular PV and monitor its value or state without polling it
- **Standard EPICS server also has records processing mechanism**
  - Control logic is programmed in records definition file
  - Allows perform routine tasks without explicitly programming in C
  - In addition to value every record has also status (OK, ALARMED, etc) and timestamp.

OAK RIDGE National Laboratory

Development process in EPICS

- **Create a driver talking to hardware (device support) - C code**
- **Put high level code in EPICS records programming**
- **Different tools exist to simplify record programming including graphical ones where developer connects blocks with lines (looks familiar!)**



How to interface LabVIEW and EPICS

- **We want to use LabVIEW for both low level and high level tasks, but still have connectivity with EPICS clients**
- **Several options exist**
  - Windows shared memory (SNS)
  - National Instruments cRIO implementation of shared memory: full IOC runs on VxWorks (LANL)
  - National Instruments CA server shared variable engine extension
  - Simple Channel Access (SCA) OS specific (LBNL)
  - CaLab Windows DLL (BESSY)
  - LabVIEW Native Channel Access for EPICS LANCE (Observatory Sciences
  - **Pure LabVIEW CA (SNS)**

Pure LabVIEW solution uses standard Network Connectivity VIs that are available on all platforms where LabVIEW is supported. The same code written in LabVIEW communicates with EPICS clients. No C code involved at all!

Oak Ridge National Laboratory

CA protocol

- Client wants to find out a value of a particular PV
- Client sends out UDP broadcast with PV name
- The server that has such PV replies
- Client checks if it already has TCP connection with this server
- If there is no such connection the client creates it
- If connection exists the client reuses existing connection
- After connection is established client exchanges messages with server over that connection.

OAK RIDGE
National Laboratory

LabVIEW implementation

- **Uses UDP/TCP VIs**
- **Object Oriented design**
  - References to objects implemented using queues (compatible with 8.5 version)
  - Every PV type corresponds to a LabVIEW class (with common functionality pushed to base class)
  - Main Server class can be extended adding functionality without digging into internals
  - Logging exists as a separate service allowing different storage engines (text files, HTTP posts)
- **No dynamic thread allocation, instead everything is served by specified number of Worker Threads that check out established connections from connection pool and perform message processing if needed**

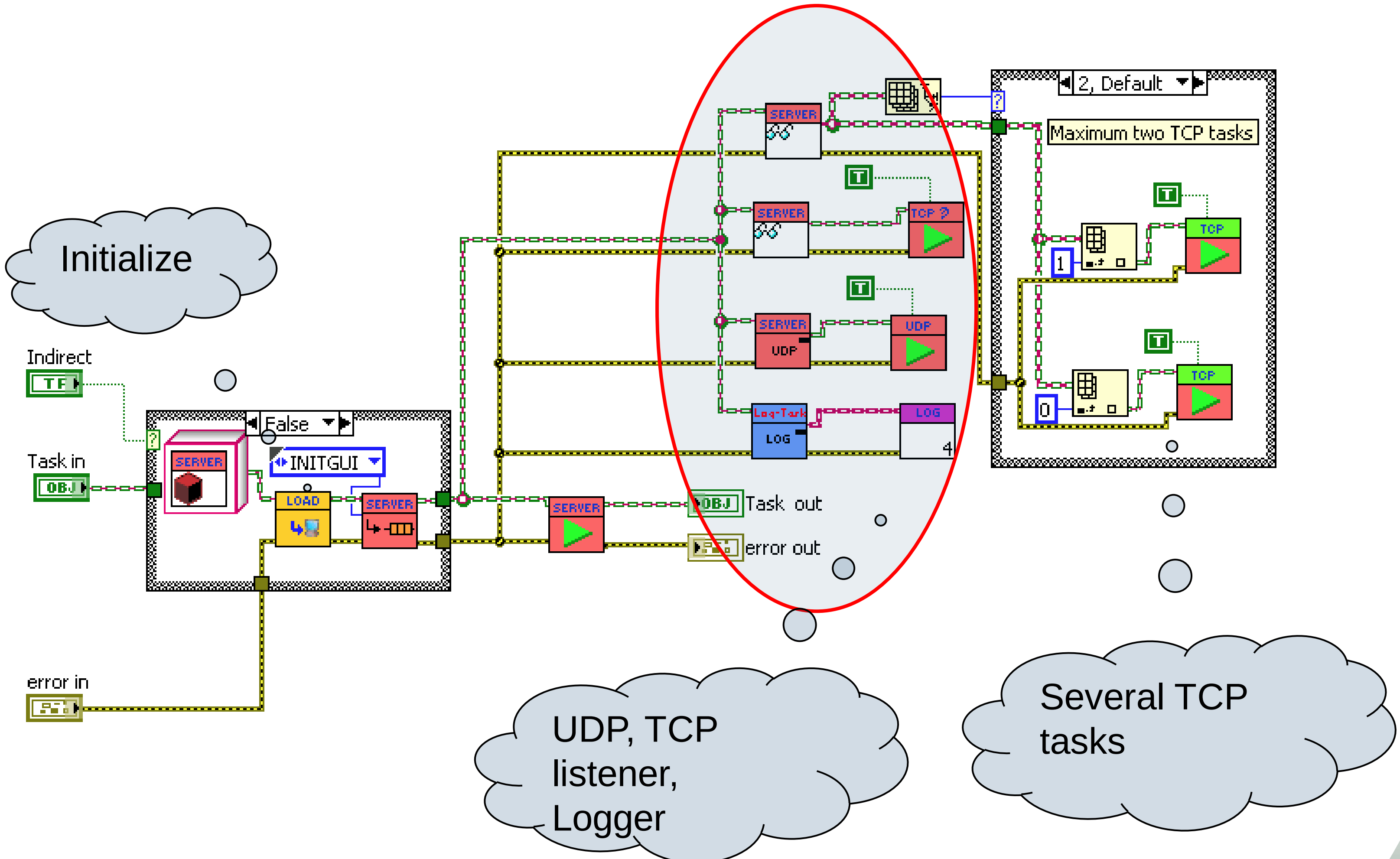
- **Naive implementation of standard Map (key-value pair)container in LabVIEW**
  - **All operations take O(N) in worst case**
  - **Heavy use of variant data type with casting**

Oak Ridge National Laboratory

Code internals: Task class

- **Task class implements basic functionality of a task**
  - Reading config file and initializing settings
  - Has a stub for Run method
  - Handles closing the task correctly
  - Shows/hides front panel (for debug purposes)
  - Task can receive events from other tasks (using Queue VIs)
- **There are several task types all inheriting from Task (all are state machines)**
  - UDP task (listens to UDP requests from clients and if PV exists replies)
  - TCP listener task (listens to TCP connections and if established puts the associated object into a pool)
  - TCP task, many instances can be active (takes connection from the pool, checks if it needs processing and returns it back to the pool)
  - Logger Task (logs different events to a file or web server)

Code internals: Launching all tasks

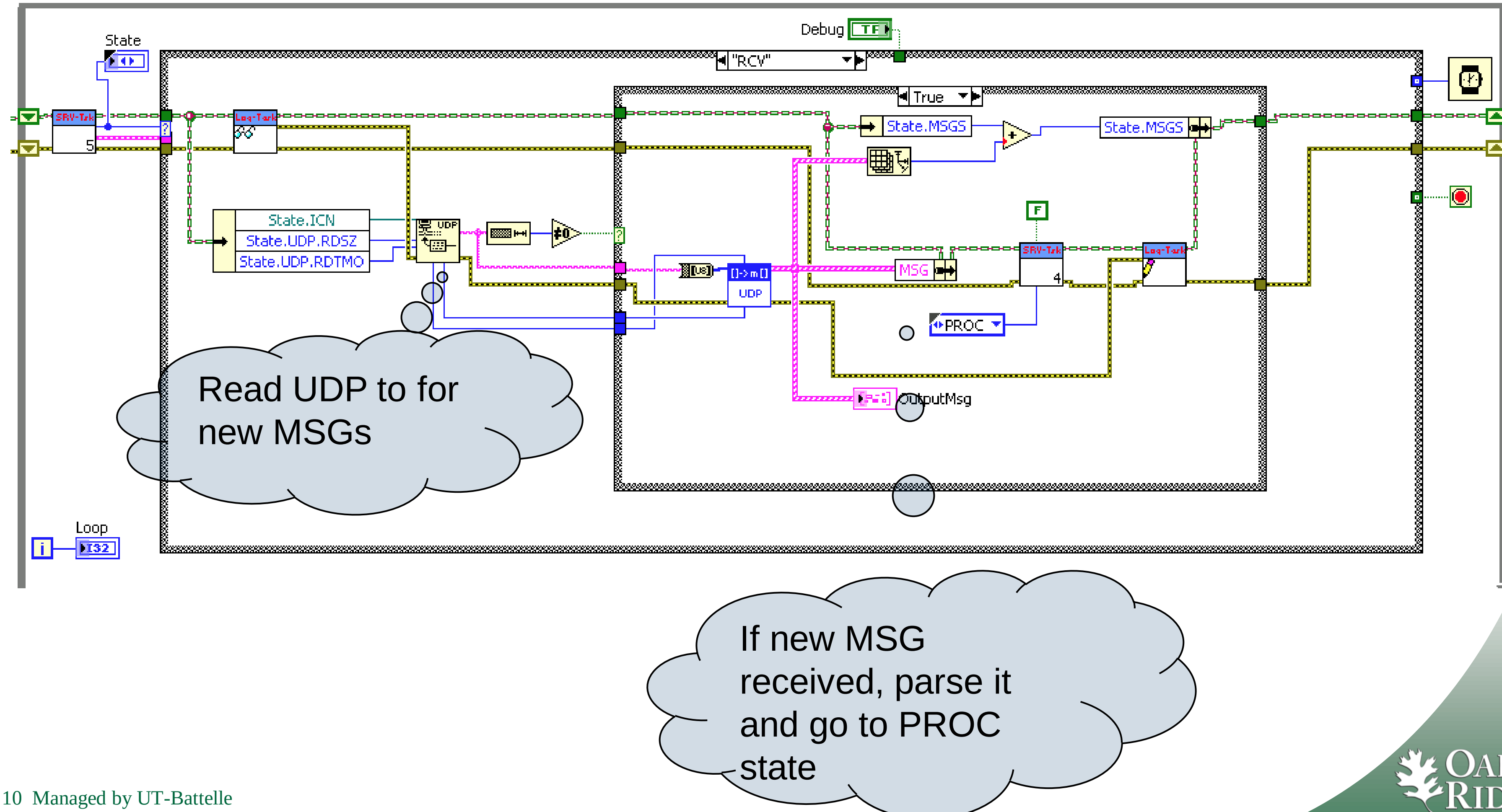


 Managed by UT-Battelle
for the Department of Energy

Oak Ridge National Laboratory

Code internals: UDP Task

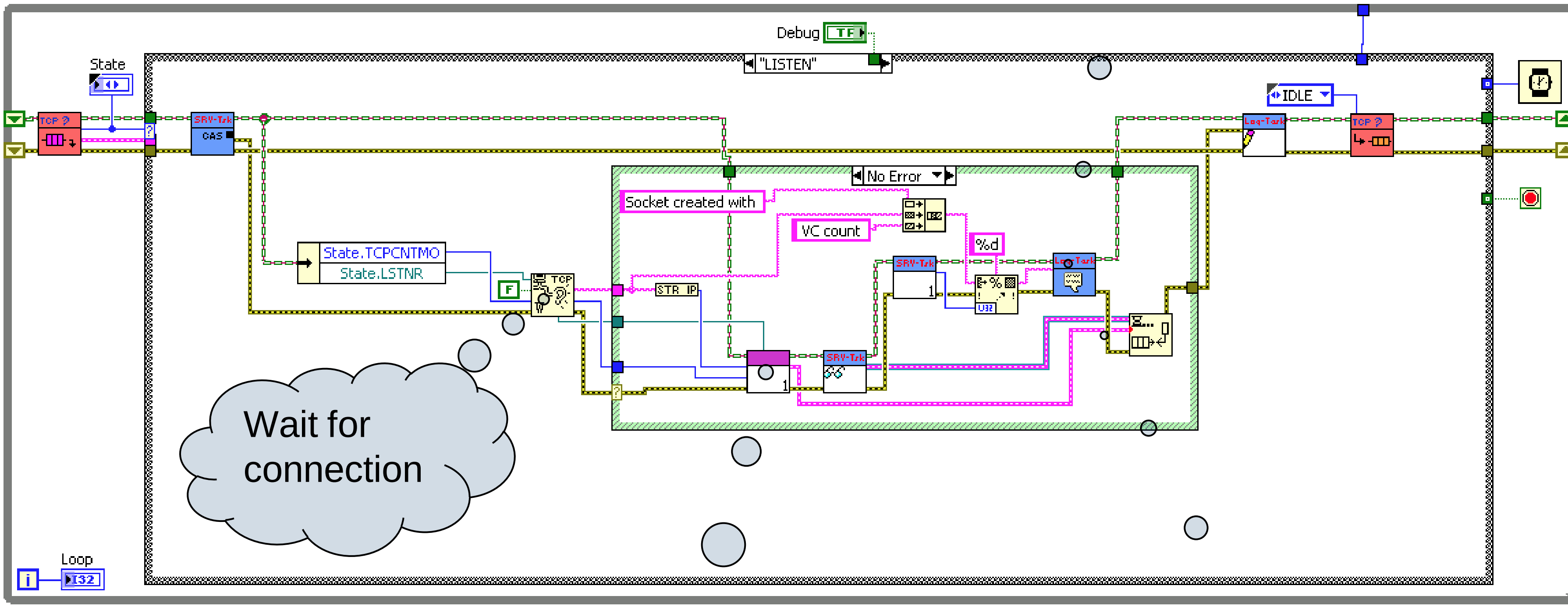
UDP Task listens to UDP port



Code internals: TCP Listener Task

TCP listener task waits for TCP connection

Log event of connection creation



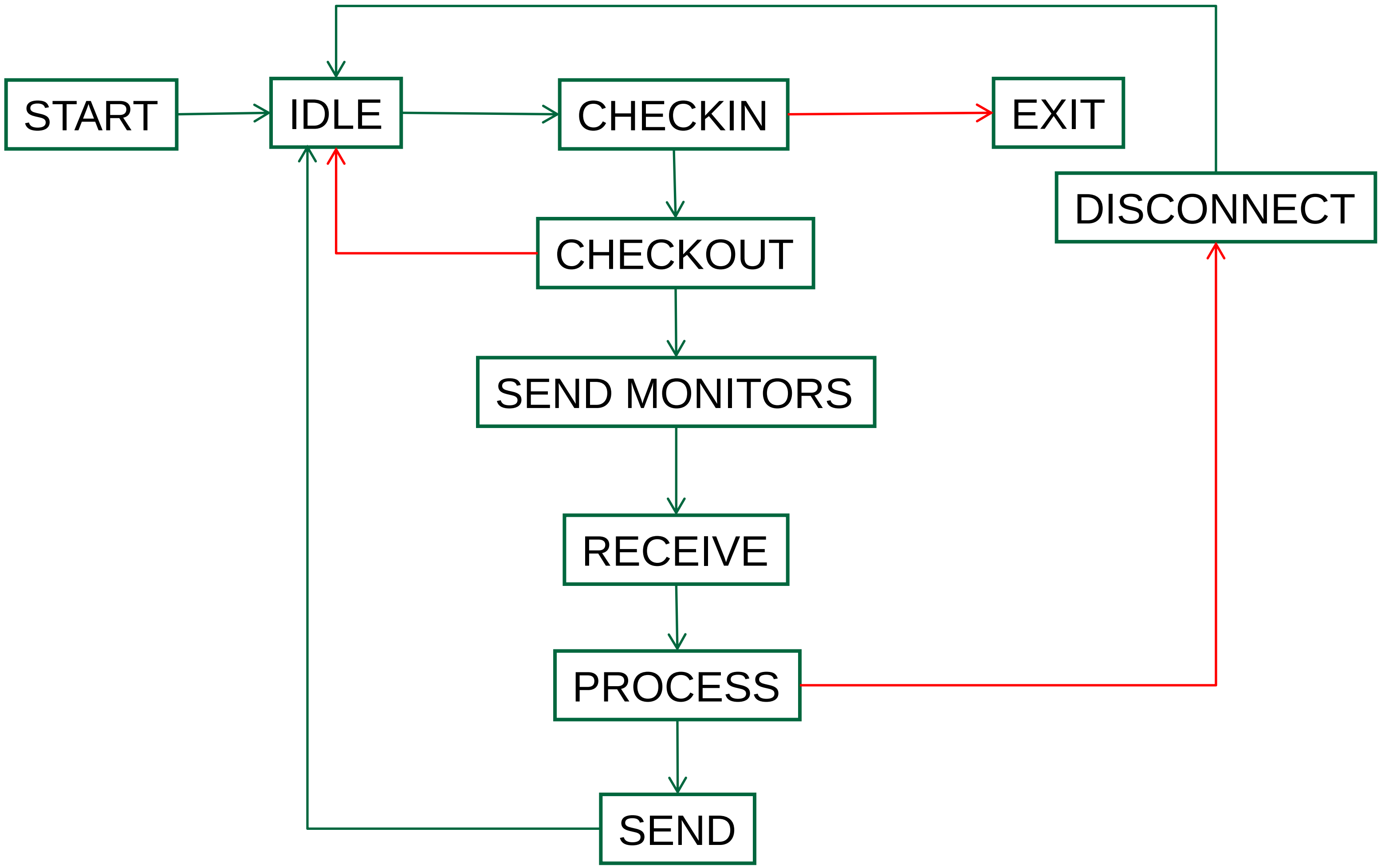
Wait for connection

If connection was established, create a new connection…

… and put it in the pool (queue)

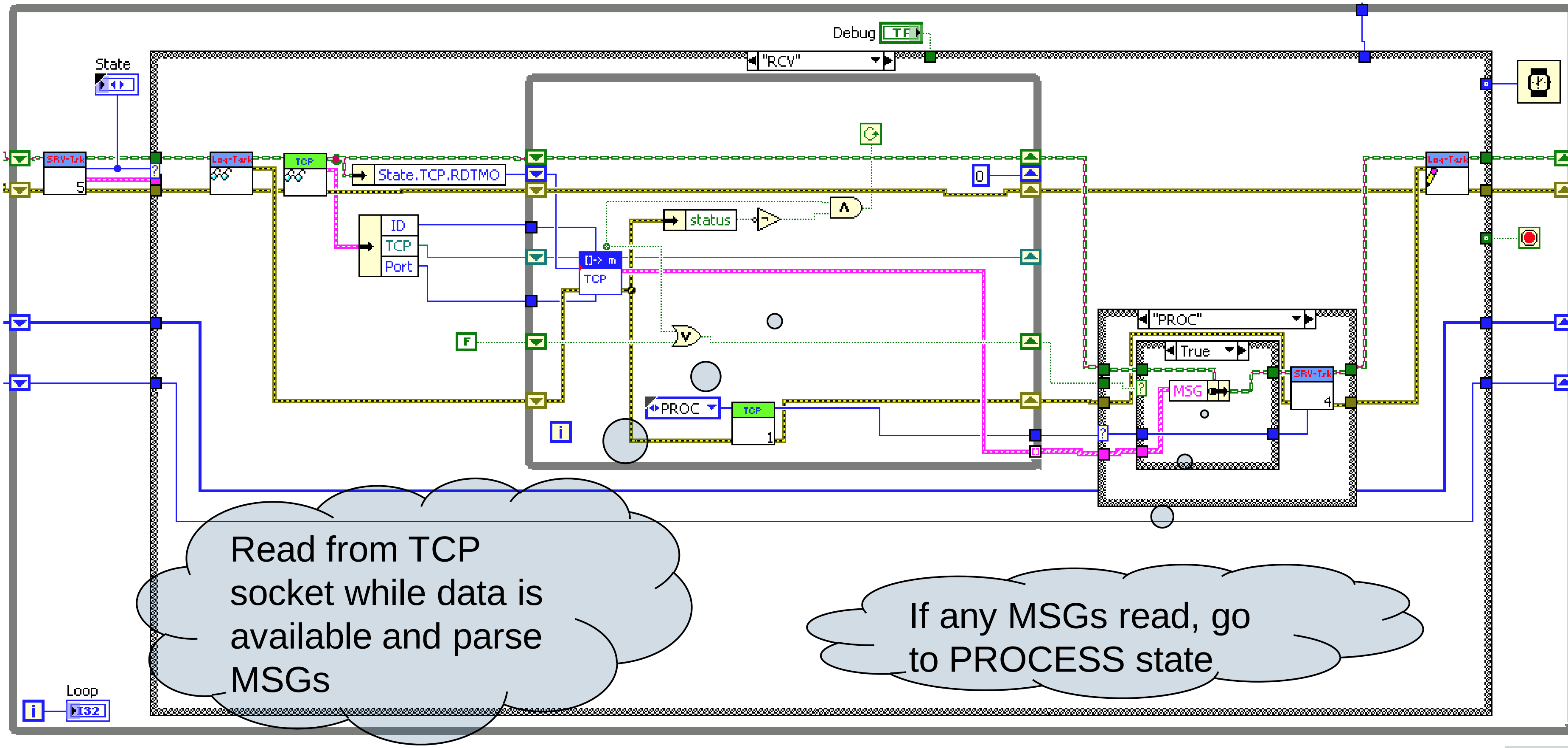
Code internals: TCP Task State Diagram

TCP Task processes connections. Several Tasks work with the same connection pool.

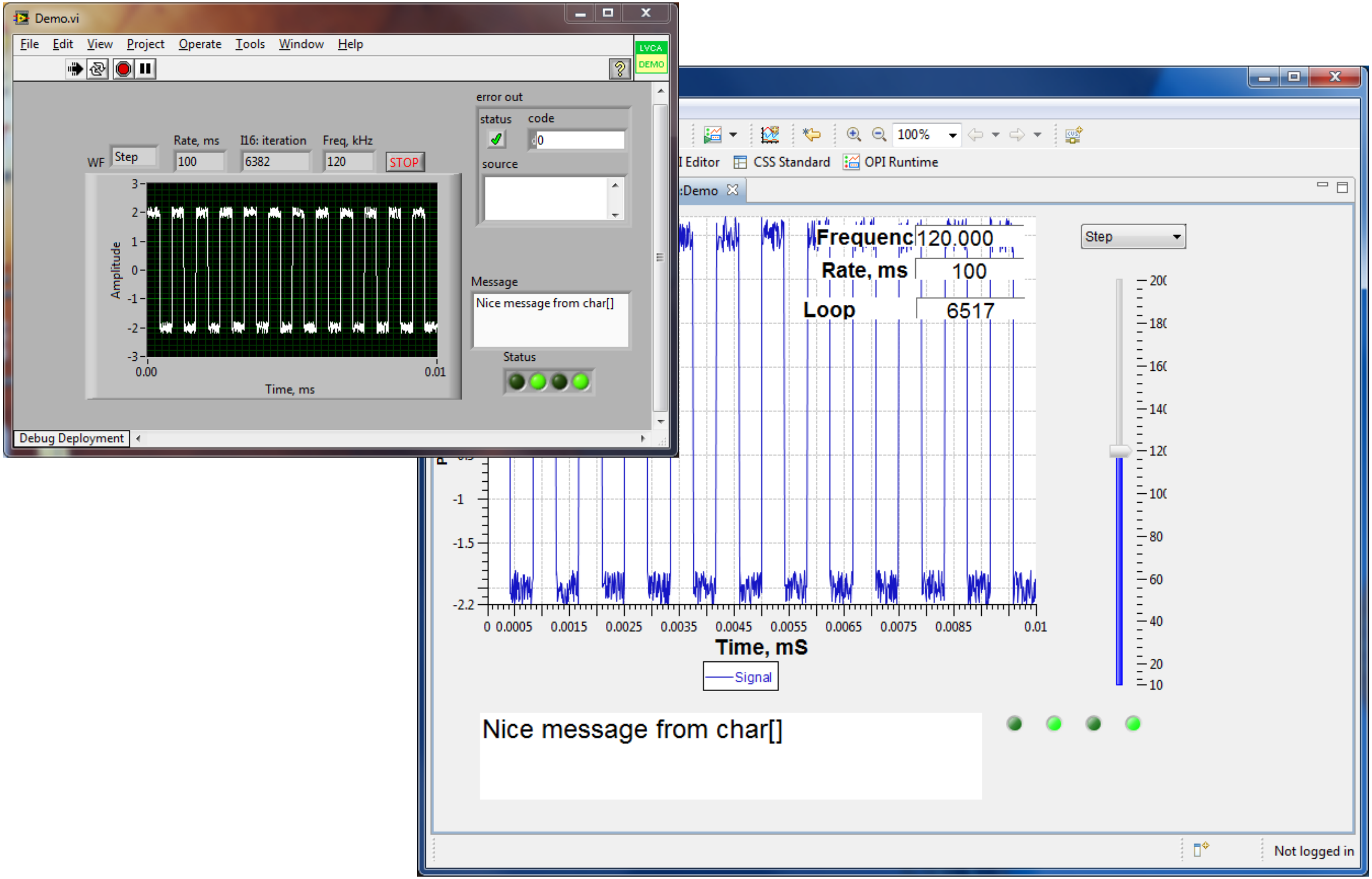


Code internals: TCP Task 1

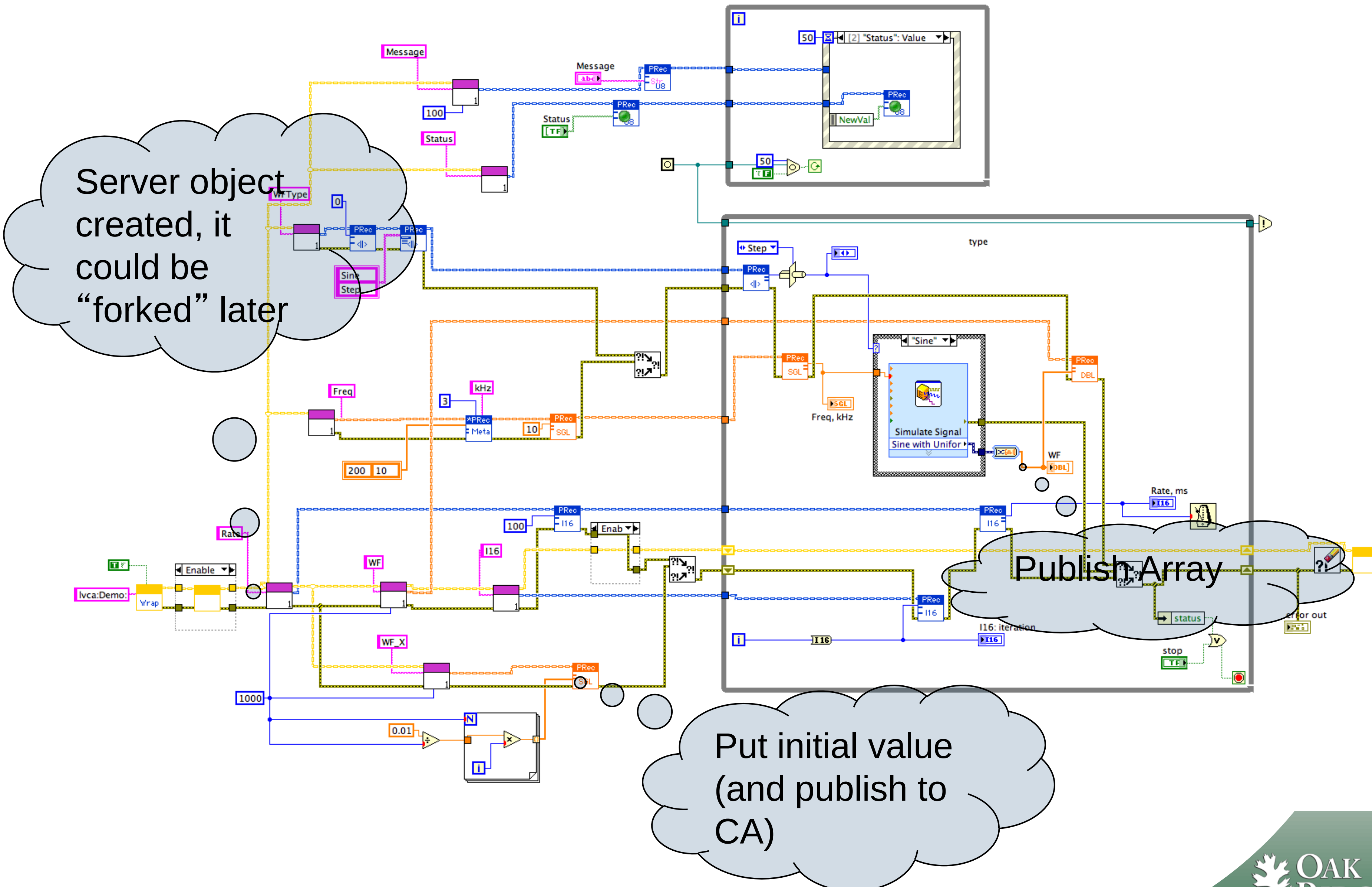
TCP Task processes connections



Demo server and GUI client



Demo program structure



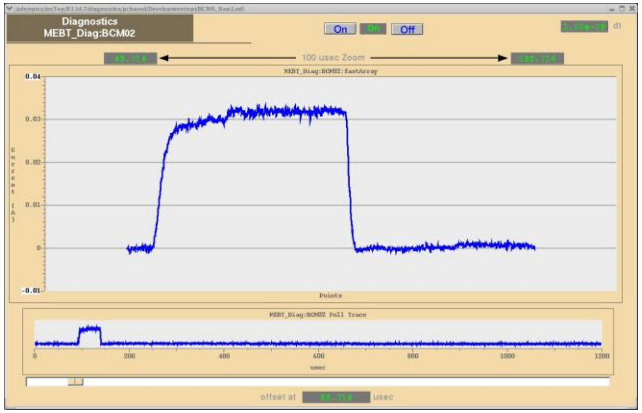
Code internals: problems and challenges

- **Multi-threaded environment**
  - Debugging is hard
  - Profiling is hard and sometime inconsistent
  - Queue operations are somewhat different (in terms of performance) on Windows and RT

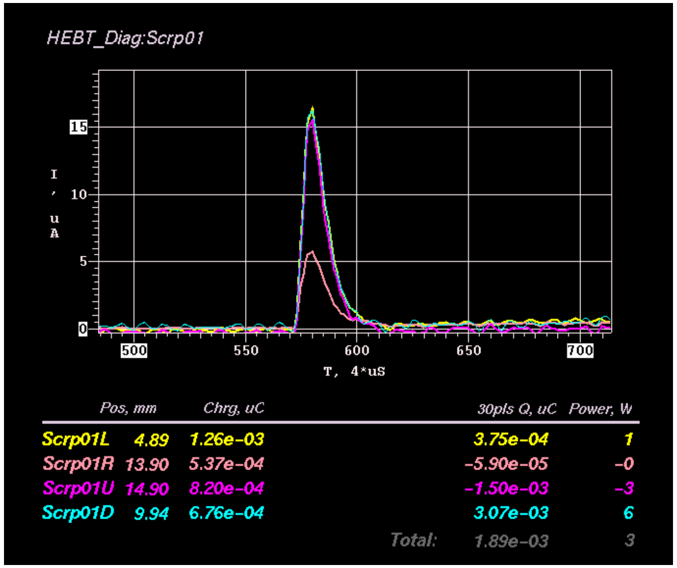
- **LabVIEW limitations**
  - OOP doesn’t have multiple inheritance (Java style interfaces)
  - No way to have custom C++ style template
  - No standard containers (Vectors, Maps)

Beam Diagnostics LabVIEW based instruments

- **Beam Instruments running on Windows at SNS**
  - Beam Position Monitors
  - Beam Current Monitors
  - Wire scanners
  - Laser profile monitors
  - Video monitors
  - Faraday Cups
  - Emittance scanners



- **Instruments running on cRIO platform**
  - Beam loss monitors
  - Beam current monitor
  - Collimator protection

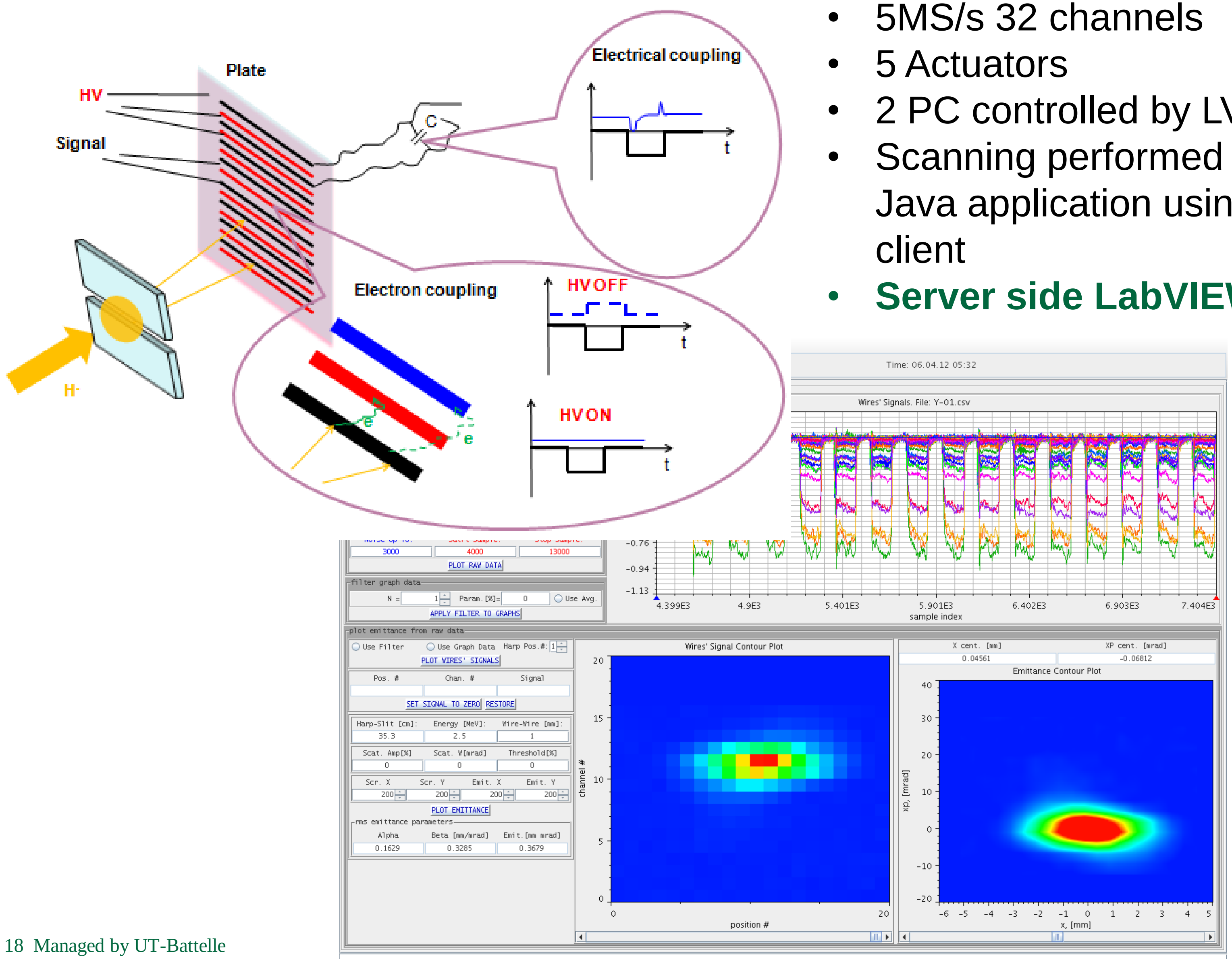


- **Typical data acquisition rate**
  - 1Hz for Windows based devices
  - 60Hz for cRIO

- **Publication rate almost always 1 Hz**

Real life example of data acquisition

- 200 MS/s 1 channel or
- 5MS/s 32 channels
- 5 Actuators
- 2 PC controlled by LV
- Scanning performed by external Java application using EPICS client
- **Server side LabVIEW only!!!**



Oak Ridge National Laboratory

Several instruments run this server implementation in production

- **PC based systems ~ 20 PCs total**
  - Laser profile scanners
  - Emittance scanners
  - Different attenuators
- **cRIO based systems 4 cRIO controllers total**
  - Collimator machine protection (with actual machine protection implemented in FPGA)

Performance considerations

- **SNS typical numbers for "big" EPICS server**
  - ~ 2k PVs
  - ~ 16 channels x 1k points per second
  - ~ 30 clients
  - Under 10 Mbit/s
- **Performance**
  - The biggest problem is setting up a test environment and test case
  - Number of PVs, clients connected, PV update rate, PV (array) size, CPU power form at least 5-dimensional parameter configuration space
  - one 1000 point WF at 300Hz/5 clients results in ~ 20% of CPU usage of iMAC with I7
  - Two 20k point updated at 1 Hz uses ~5% of CPU on an average 4 yrs old industrial PC

Oak Ridge National Laboratory

Summary

- Many LV EPICS interfaces exist
- There is no ideal one
- The need in full featured IOC is the key parameter for selection process
- Windows shared memory is still default way at SNS
- SNS pure LV version is cross-platform and seems to satisfy all needs, but is not finalized yet
  - Beta testers needed!
  - Client implementation is not ready yet
  - Performance tuning
  - Bad map implementation can become a bottleneck for servers hosting many (~5000 PVs) in busy networks
- The same programming technique can be used to implement any custom communication protocol
- LabVIEW can be used as general purpose programming tool and it is fast!

OAK RIDGE
National Laboratory

Questions

NATIONAL INSTRUMENTS LabVIEW

?

EPICS

OAK RIDGE National Laboratory