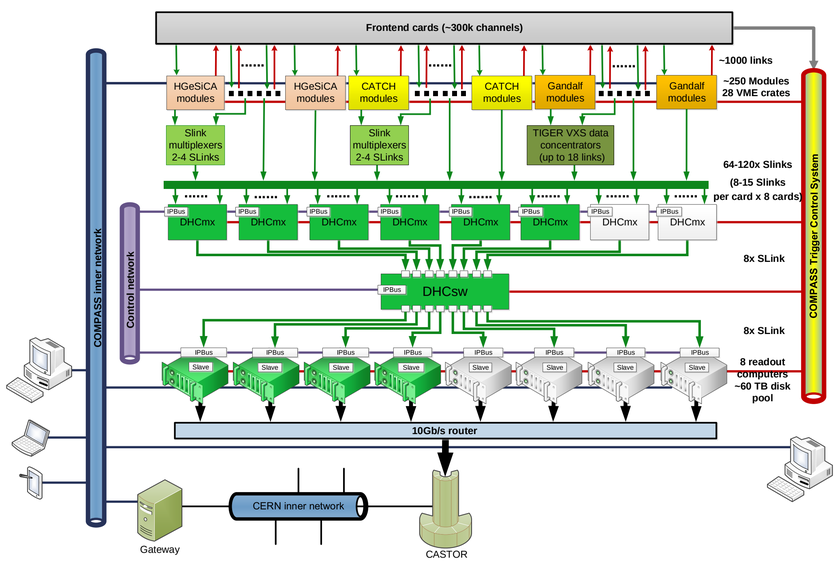
The Online Monitoring API for the DIALOG Library of the COMPASS Experiment

O. Subrt[ab], V. Jary[b], J. Novy[b], J. Tomsa[b], M. Virius[b], M. Bodlak[c], A. Kveton[c], S. Huber[d], I. Konorov[d], D. Levit[d], D. Steffen[d], V. Frolov[e]

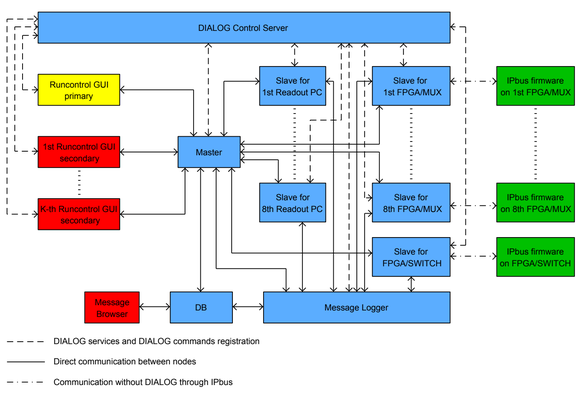
(a) CERN, Geneva, Switzerland, (b) Czech Technical University in Prague, Czech Republic, (c) Charles University, Prague, Czech Republic, (d) Technical University of Munich, Germany, (e) Joint Institute for Nuclear Research, Dubna, Russia

iFDAQ Architecture – Hardware Structure

- Hardware based E.B.
- Data concentrated by 6 (up to 8) DAQ modules with multiplexer firmware
- Distribution to 4 (up to 8) readout computers by DAQ module switch firmware
- Full events received by servers
- Consistency check at many layers
- Events checked and transferred to DATE data format



iFDAQ Architecture – Software Structure



- Runcontrol GUI is a graphical user interface
- Master is a main control process
- Slave-readout readouts and verifies data
- Slave-control monitors and controls the FPGA cards
- MessageLogger stores informative and error messages into the database
- MessageBrowser provides an intuitive access to messages stored in the database

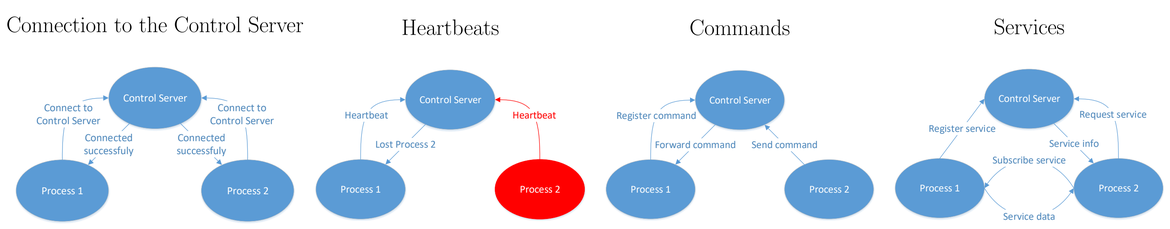
DIALOG Library – Overview

- New communication library for inter-process communication
- It is implemented in Qt framework
- Dialog means conversation, talk or speech (**D** – distributed, **I** – inter-process, **A** – asynchronous, **L** – library, **O** – open, **G** – general)
- Communication based on the publish/subscribe method
- It uses TCP/IP protocol and sockets for message transmission
- It saturates the 10 Gbps network bandwidth already with messages of size 1 kB
- It was deployed and fully incorporated to all processes in May 2016 (before Run 2016)

DIALOG Library – Design Requirements

- **Efficient communication mechanism** – asynchronous behavior, sending and receiving as soon as possible
- **Uniformity** – all processes use the same communication mechanism
- **Transparency** – any running process should be able to communicate with any other process
- **Reliability and robustness** – system recovery in a self-recoverable manner from error situations

DIALOG Library – Scenarios



DIALOG Library – Online Monitoring API

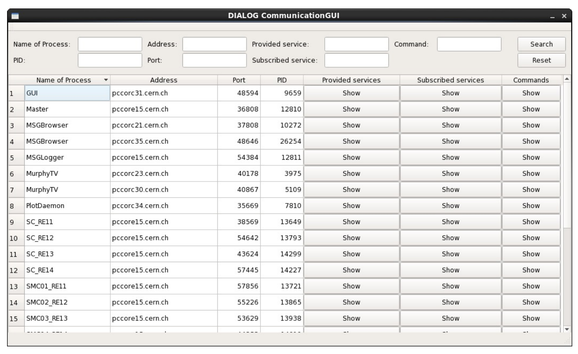
The DIALOG library distinguishes three types of process. The process type determines the purpose of a process and how the DIALOG library should deal with it.

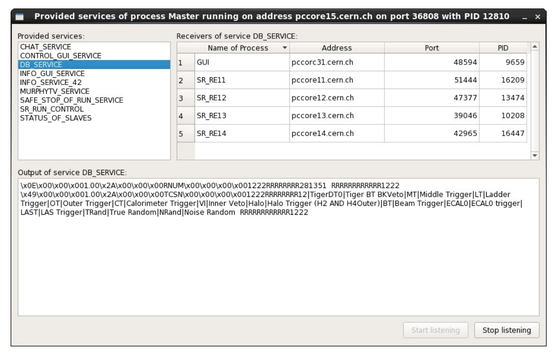
- **ControlServer** – The Control Server keeps an up-to-date list of all processes, services and commands in the system. It receives registration messages from providers and request messages from subscribers. All processes send heartbeats at regular intervals so that the Control Server can be assured that they are working properly.
- **Custom** – It covers all processes they should communicate through the DIALOG library.
- **Monitoring** – Processes being responsible for the monitoring of the DIALOG library. Each of them represents a monitoring tool with a unique purpose. This general concept offers an easy to use API for a developer to implement any monitoring tool.

Firstly, a monitoring tool is connecting to the Control Server as well as anyone else. The Control Server recognizes the *Monitoring* type of a process and deals with it like with a monitoring tool during its whole life cycle. The Control Server sends monitoring info to a monitoring tool in XML format after successful connection of the monitoring tool to the Control Server. This monitoring info contains list of all connected processes of type *Custom* to the Control Server. The monitoring info in XML format also consists of all services being provided and subscribed by processes and all commands being registered in the Control Server. If something changes (e.g. lost process, providing of a new service, connection of a new process, etc.), the Control Server will recognize it and re-send a new monitoring info immediately to all monitoring tools currently connected.

DIALOG GUI

The DIALOG GUI is one example among many of the monitoring tools using the online monitoring API for the DIALOG library. The behavior of complex distributed applications can be very difficult to understand without the help of a dedicated tool for online monitoring. The DIALOG GUI allows the visualization of the processes involved in the distributed communication system. Moreover, a user can start to listen to the selected process and see the whole process communication.





DIALOG POST Daemon

The DIALOG POST Daemon serves as a middleman between desktop application and web application. The DIALOG POST Daemon's purpose is to catch all communication being sent among all processes with process type *Custom* and transmit all data in JSON format using HTTP and its POST method. The POST request is received by a web application for a communication measurement. Results from the communication measurement tool can help with understanding of bottlenecks and better load balancing of processes on machines.

DIALOG WebSockets Daemon

WebSockets is an advanced technology that makes it possible to open an interactive communication session between the user's browser and a server. With this API, an application can send messages to a server and receive event-driven responses without having to poll the server for a reply. WebSocket provides full-duplex communication channels over a single TCP connection. The idea is to develop the DIALOG WebSockets Daemon as a middleman between desktop communication system and a web application. Runcontrol GUI for monitoring and controlling of the iFDAQ as a web application is planned for year 2019.

Supported by

MINISTRY OF EDUCATION, YOUTH AND SPORTS
Bundesministerium für Bildung und Forschung
Excellence Cluster Universe
MLL

Place/Time

CHEP 2018 Sofia, Bulgaria
July 9 – 13, 2018
Sofia