Meeting of the Architecture Working Group with Giovanna Lehmann

September 20th

Items discussed:

- Ros context and interfaces with neighboring systems
- Present ROS architecture
- Open issues

Summary of the open issues

1. **ROS data multiplexing**. It is not clear if the ROS should provide partial event building. The old DAQ-1 architecture foresaw that the ROS would respond to Event Building requests sending ROS fragments, which are build combining several ROB fragments coming from different ROLs. Between the Dataflow scenarios that are proposed at the moment there are ones that foresee that all the individual ROB event fragments (~1.6K) should be sent directly to the DataCollection network in case of Event Building requests. As the selection of a candidate Dataflow scenario is now urgent, the AWG is supposed to act as a driving force to identify parameters that should be measured to validate the different ones.
2. **Interface with DataCollction**. The interface point between ROS and DataCollection is ambiguously defined.
   - ROS High Level applications (IOManager) link a software library (mantained by DC), which provides different implementations of the DC message passing API, based on different underlying protocols (TCP/UDP/Raw Ethernet). Thus in this case the actual interface point is the DC message passing API.
   - In some architecture scenarios, ROS buffer modules (RobIn), which are custom devices, could be directly exposed to DC messages. In this case the RobIn will have to implement the logic for receiving and interpreting DC messages directly in the firmware (i.e. no library could be used). Thus the actual interface point in this case would be the DC message format

   An alternative scenario has been evocated during the discussion: ROS applications could expose a Buffer Management interface to DC, that will have to provide a light application that would run on the same machines as the ROS applications and would take care of data retrieval using the ROS Buffer Management API. Such a scenario could provide a cleaner division of responsibilities between the components.
3. **Security of CLEAR messages**. The ROS mandate is to keep incoming ROD fragments inside its internal buffers, until an explicit CLEAR message is received from DataCollection. The ROS buffers could get filled (thus issuing a XOFF!) if a large fraction of CLEAR messages would be lost. DC claims that using a secure protocol (like TCP) to transport CLEAR messages would compromise the global DataFlow network performances. Different alternative mechanisms for avoiding ROS Buffer overflows have been proposed but no decision has ever been taken.

4. **Data sampling criteria**. This question its clearly related to the global discussions about data monitoring in the Dataflow. At the moment it is not clear if at all the ROS should respond to data monitoring requests. In case it should, there are again different scenarios which have a much different impact on the ROS architecture.
    - The ROS could only sample requested data (i.e. ROD fragments that have already been requested by ROI or EB requests). The implementation of such a scenario is straightforward.
    - The ROS could receive sampling requests for ANY kind of Fragments coming from the RODs. Implementing such a scenario would most likely require modifications on the firmware of the RobIn.
5. Should the ROS units be “resources” or “segments” relatively to DAQ partitioning? The implications of such a choice on the ROS architecture has not been addressed extensively.
6. **Error recovery**. The definition and implementation of possible automatic error handling functionalities in the area of the ROS could again be strongly dependent on the final architecture choices for the system. If all the custom ROS buffers (RobIn) will be directly exposed to Data Collection, it is not clear who would be responsible of handling ROS errors and to possibly start automatic error recovery procedures.