Web-Based DAQ Systems: Connecting the User and Front-End Electronics

Thomas Lenzi*
Université Libre de Bruxelles (ULB)

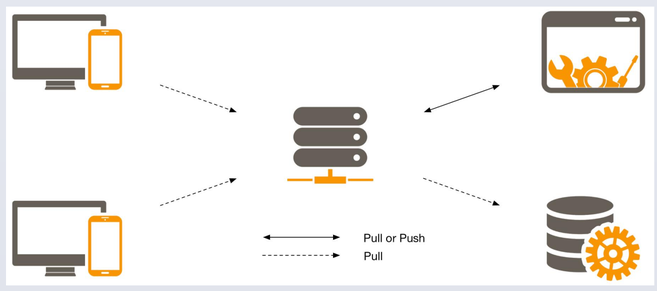
Introduction

The backbone of every DAQ system is the integration of its components into a single coherent architecture. The latter must provide a seamless user experience abstracting every aspects of the design and be optimized for efficient data readout and transfer. While companies provide all-in-one systems that are well integrated and include intuitive user interfaces, they are rarely suitable for custom applications that need to interface with components from different vendors. Custom interfaces need to be developed to control the system and transfer data between the subsystems.

The architecture of a DAQ system is mainly defined by the way data is distributed between the components: it is either pulled or pushed. Data is pulled from one component to another when a transfer solely occurs upon request from the receiver, and is pushed when the decision to forward data is taken by the transmitter. As each component either pushes or pulls data, a bottleneck appears in the system due to limitations in the bandwidth, storage capacity, etc. This in turn can cause data losses or system failures if not handle properly. Classical approaches to DAQ system design make use of a central node which handles the system and coordinates the communication between the components. However, new technologies have enabled the development of new topologies which blur the lines, integrating components which fulfil multiple roles.

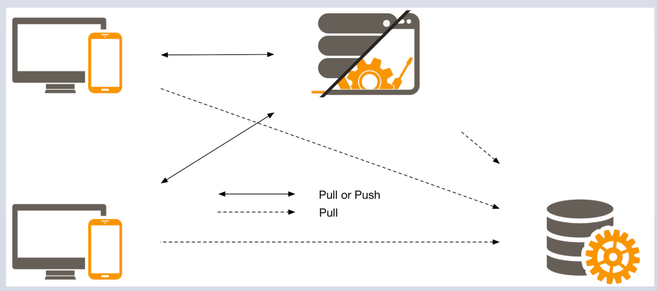
Star Network Topology

The architecture of the DAQ used in most systems relies on the presence of the server which acts as the central node connecting the clients, the database, and the electronics. All transactions are required to transit through it, slowing down the performance of the whole infrastructure when scaling up. Furthermore, the client needs to constantly pull data from the server and the database to keep informed of the changes happeing in the system or those submitted by other clients.



Mesh Network Topology

The system described in this poster moves away from a star network in which the server is the central piece and tries to form a mesh network where every component can directly access all other nodes. By taking advantage of the increase in computational power of microelectronics, it is possible to merge the functionalities of the server and the electronics in a single embedded system. Furthermore, developments in web technologies have enabled easy and reliable two-way communication between the client and the server and between the client and the database. The WebSockets technology provides response times up to ten times faster than classical solutions used to pull the server. The obtained system thus allows for faster inter-node communication which releases the traffic on given data paths and allows for more bandwidth and activity.



Running a Web Server on a MicroBlaze

A Microblaze processor is embedded on an FPGA and acts as web server to deliver web applications to clients which connect to the IP address of the board. Once the web application has been downloaded by the client, real-time communication is established over WebSockets to transmit requests between parties.

When an Internet browser pulls a web page from a server, it sends an Hypertext Transfer Protocol (HTTP) request containing the type of request and the path of the page. After analysis of the HTTP request, the server performs the desired operation and returns a status code, information headers, and optionally data.

HTTP request

```
GET /index.html HTTP/1.1
Host: www.web-daq.com
```

HTTP response

```
HTTP/1.1 200 OK
Date: Thu, 28 July 2016 10:36:02 GMT
Content-Type: text/plain; charset=UTF-8
Content-Encoding: UTF-8
Content-Length: 29
This is the web application !
```

WebSockets and TCPSockets

To allow the server to push data and enable real-time communication, the WebSockets standard was introduced in the HTML5 release. WebSockets are sockets that can be used inside the Internet browser to communicate with a server following a given data format. When a connection is established, it remains open on both sides reducing the overhead on each data packet and decreasing the latency.

Upon connection from a client to the server, a dedicated HTTP request is sent informing the latter that the WebSockets protocol is being used. The request contains an upgrade header which informs the server of the change of communication protocol, along with a random key. The response from the server informs the client that the switch was made and returns a hash which prevents multiple connections from a same client. The server stores the information of the client in a list of active users until the latter disconnects or the connection times out. After this handshaking procedure, data can be sent between parties without restriction.

Handshake request

```
GET /websocket HTTP/1.1
Host: www.web-daq.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key:
x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 13
Origin: www.web-daq.com
```

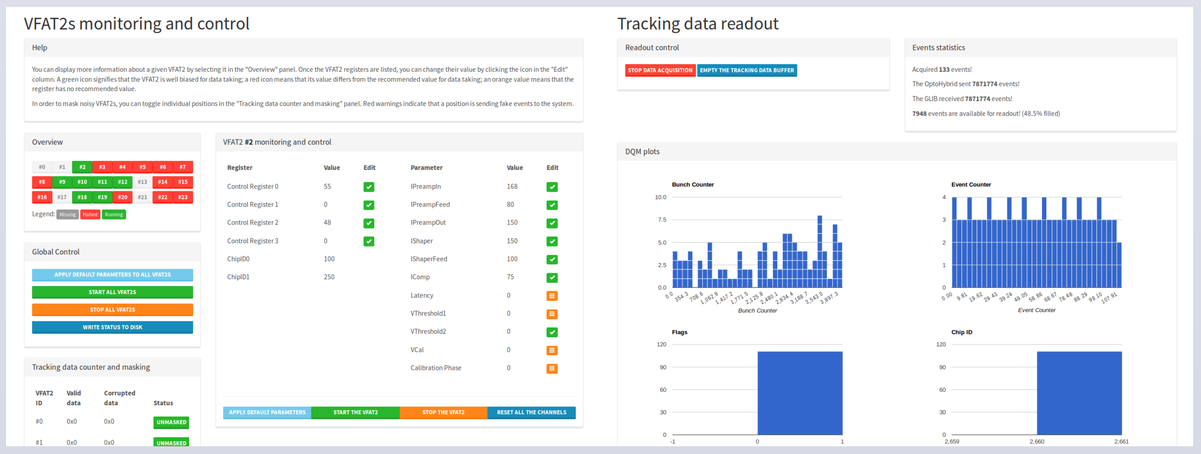
Handshake response

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept:
HSmrc0sM1YUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

Although WebSockets provide an implementation of sockets in the web browser, they also define a format for the data that is being transmitted. This limits the field of application to the connection between a WebSockets client and server. A more generic approach uses the TCPSockets, a raw implementation of sockets in the web browser which is still in experimental phase. This technology enables the connection between any systems that support TCP sockets, including databases. A direct communication between the client and the database can thus be established in order to retrieve stored data, effectively by passing the need for an intermediary server.

User Interface Design

To handle data in a dynamic and convenient way, the web pages use Bootstrap and AngularJS. Bootstrap is a design framework which provides simple and readable themes to layout the pages. The content of the page on the other hand is managed by AngularJS which is a front-end JS framework. It helps rendering dynamic content in function of the data received from the server. Actions can be triggered or disabled according to the state of the system, data can be updated live, etc. It renders the user experience more fluid and enjoyable by providing a usable set of actions and an efficient navigation through the control pages.



Conclusions

Using recent developments in web technologies, we were able to create an innovative data acquisition system architecture which provides higher flexibility and better utilizes the resources of the nodes in the network. Compared to classic designs which use pull/push techniques to transfer data, the new implementation offers event driven interactions, making use of the bandwidth only when necessary.

References

Node.js - https://nodejs.org
AngularJS - https://angularjs.org
Socket.IO - http://socket.io
Bootstrap - http://getbootstrap.com/

*The author is supported by the F.R.S.-F.N.R.S.

RESEARCH POSTER PRESENTATION DESIGN © 2012
www.PosterPresentations.com