Architectur Figure
of the
DataPoint

Author: M. Siegl, G. Damianitsch
File: Architectur Figure
Modified: 2015-02-10
Version: 1.0

Abstract

This document describes the architectur of the DataPoint in a Software Defined Gateway (SDG). To understand this document it is required to understand the basic idea of a SDG.

Details on how a GBot has to access the DataPoint are not within the scope of this document. To recognize those information see the API-documentation of the software and the source code of the GBots within the technical showcase.

Content

1 Operational Information Flow

In the document *General Description of the SW-Interface inside the Gateway* the requirements are described, which the DataPoint has to fullfill for a Gbot. The GBot expect a well formed and prepared DataPoint Value (DP Value).

The DataPoint Value has to represent a Physical Value, detected by a Sensor. Therefore some processing and calculation is necessary. In Figure 1 the principle of information flow is shown.

The Physical Value (e.g. Temperature) is measured by the Sensor. The Sensor is connected on a field-bus system (e.g. RS422, CAN or other).

The Driver handles the protocol on the field-bus and acquires the Measured Value from the Sensor. The Sensor provides some Bits, the Measured Value, which represents the size of the current Physical Value. In most cases the Measured Value is a linear function between 0% to 100% of the measurement range.

Next the Measured Value has to be formated to a common format, which is independent of the Sensors specific behavior. This task is performed by the Gatherer. The Gatherer must be configured in conjunction to the Sensors manner. The Sensor Value is well formated for postprocessing in a generic way – it is of type *double*.

Next the Sensor Value runs through an Adapter Chain be become the Buffer Value. The Adapter Chain is a chain of methods (similar to pipes in Unix). The adapters could have memory inside (remembers previous values). The Buffer Value is the uniform representation of the Physical Value.

If the caluclation by the Adapter Chain results to a changed Buffer Value, these new Buffer Value is distributed through the Daemon Endpoint to the Client Endpoint up to the DataPoint.

The DataPoint provides the Buffer Value as one value of the DataPoint Values to the GBot. Several methods exists for transfering the DataPoint Values into the GBot.

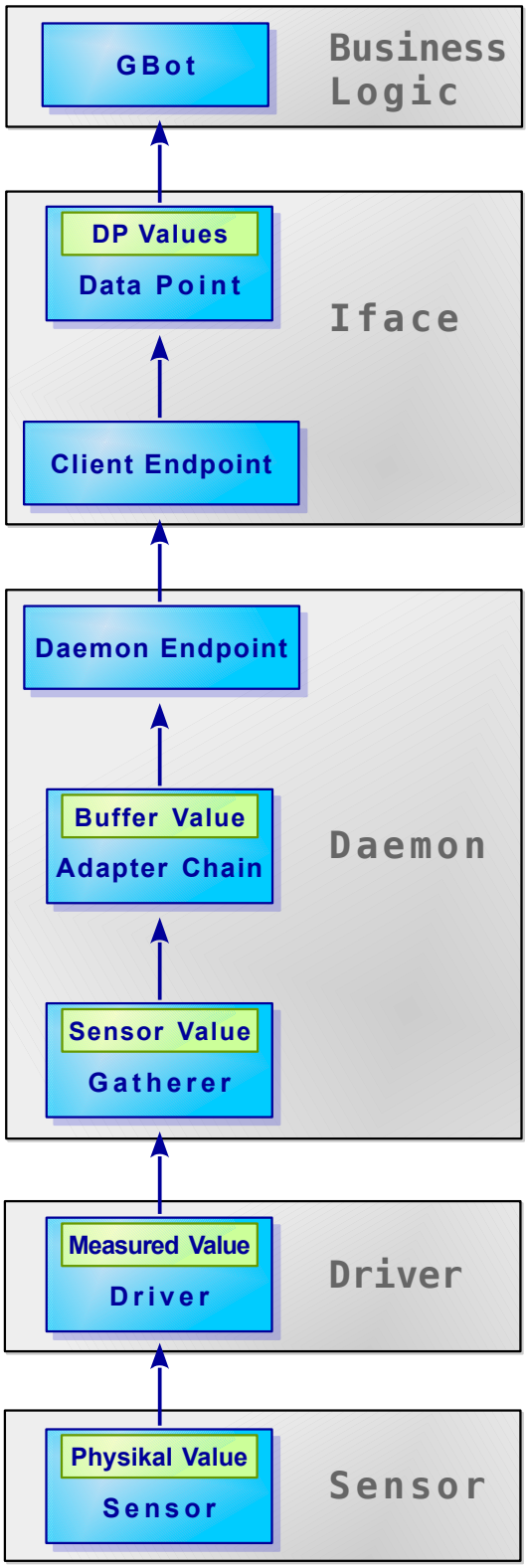

Figure 1: Operational Information Flow

2 Scaling up

The structure of the software is designed for scaling up (see figure 2). On one hand, the system can handle more then one GBot; on one or more then one JVM. On the other hand, the system is able to handle some Sensors and assigned Gatherers. Additionally each Sensor Value can feed more than one Adapter Chain (see figure 3).

2.1 Multiple GBots

The information flow beween the Daemon and the GBots goes through the Daemon Endpoint and Client Endpoint. On a Gateway only one single Daemon runs within a JVM – the JVM-Daemon. Several more JVMs could run on the Gateway – JVM-GBot-1 and JVM-GBot-2, house one or more GBots – GBot-A to Gbot-D.

The Client Endpoint is a singleton within its JVM. Each DataPoint connects to the Client Endpoint. The Client Endpoint connects to the Daemon Endpoint. The connection presetting is based on tcp port 3449 of localhost. This defaults can be overridden with system properties, but there are no provisions implemented for synchronization of timestamps from multiple hosts.

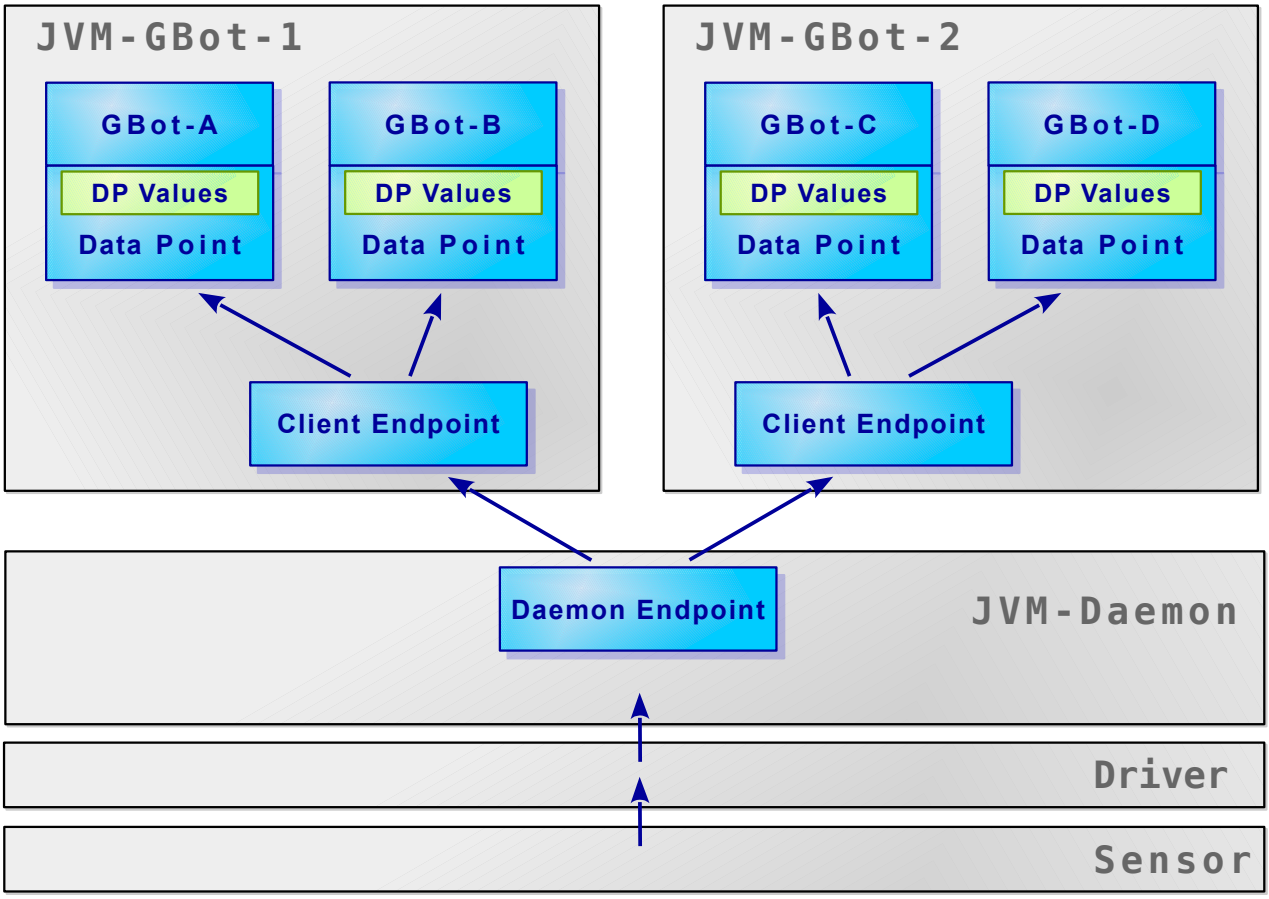

Figure 2: Multiple GBots and Multiple JVMs for GBots

It is possible, that one GBot serves more then one DataPoint (not shown in figure 2). Regardless this possibility, a single DataPoint is able to provide multiple DataPoint Values to the GBot.

2.2 Multiple Adapter Chains

Each Sensor is direct associated to a gatherer. The Gatherer provides the Sensor Value an Adapter Chain. At startup the system for each Gatherer an empty Adapter Chain will be created (see figure 3 – the blue Adapter Chains). So the GBots are able to get the Sensor Value in unmodified manner.

In addition, it is possible, but not necessary, to create extra Adapter Chains, feed by an existing Gatherer (the orange Adapter Chains). This extra Adapter Chains can be configured to adapt the Sensor Value to be more convenience to the GBot. The Buffer Value differs now from the Sensor Value.

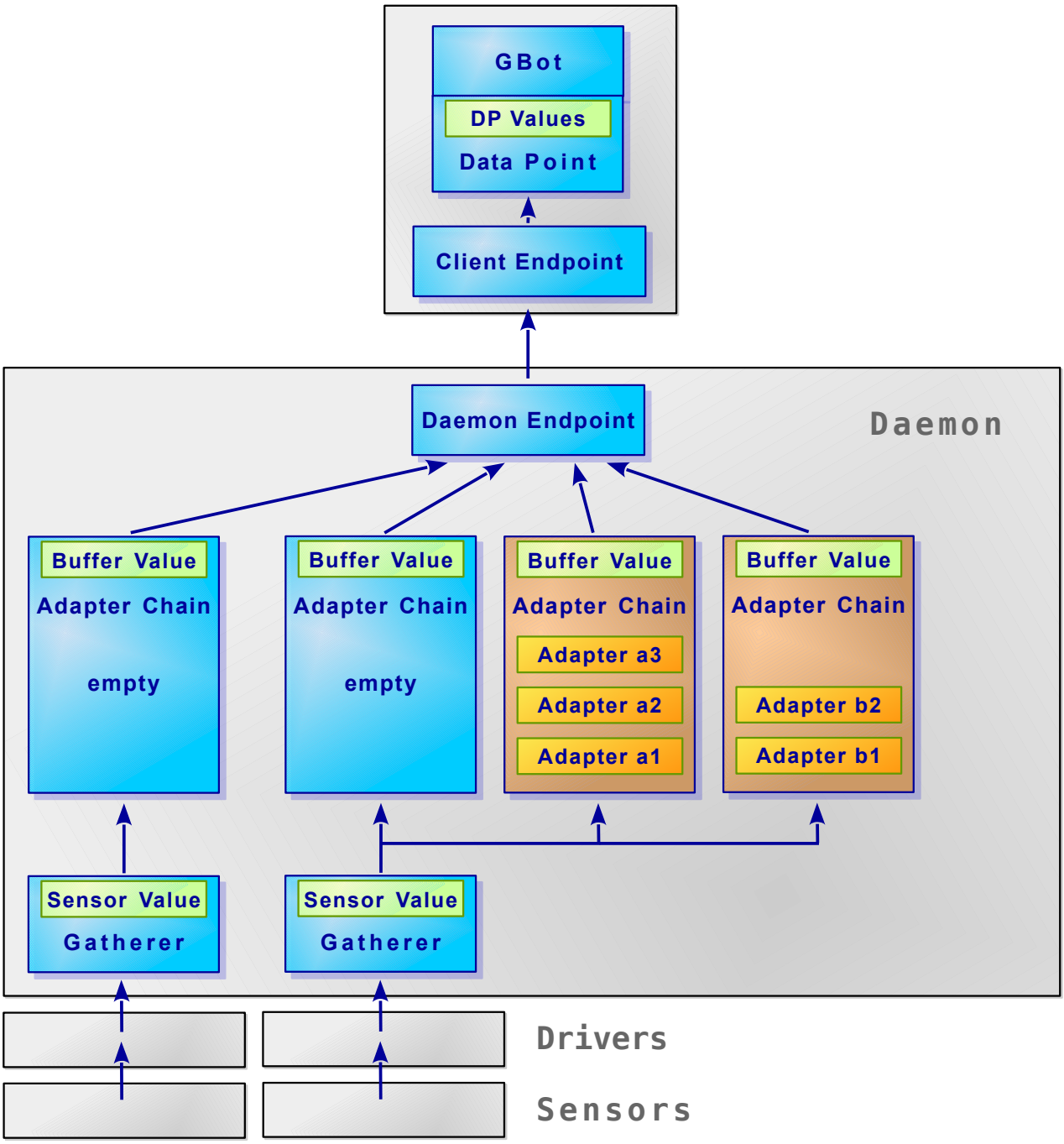

Figure 3: Multiple Buffer Values

The description of the currently available Adapters can be seen in the API-documentation. The number of Adapters within one chain is not limited. Multiple usage of one Adapter is allowed.

3 Configuration

The system has to be configurated. On one hand the individual behavior of the sensor must be configured during physical setup of the system. On other hand the extra adapter chains must be created and configured by the GBots according to there needs.

3.1 Sensor spezific Configuration

The physical setup of the system includes the following tasks:

- Connection of the sensor to the field bus, configure the sensor appropriate, so it is a member of the field bus (e.g. Modbus device ID; e.g. I2C address)
- Configuration of the Driver for a propper communication to the sensor. Die Driver must poll the Sensor to get accurate Measured Values.
- Configuration of the Gatherer to normalize the Measured Value to get a Sensor Value which is independent of the Sensor type and the field bus it is connected.
- Also it is possible to give some extra information to the Gatherer – Meta Info. This Meta Info can be used by GBots to find and select a desired Gatherer. For details about this mechanism see the API-documentation.

This physikal setup has to be performed by stuff, who install the Gateway and the Sensors in front.

3.2 Adapter Chain Configuration

When the Sensor specific configuration is done, the Buffers are available to be accessed by GBots. Sometimes GBots do not need Sensor Values purely, but conditioned. So the GBots can define and configure extra Adapter Chains. This configuration should be done during runtime of the GBot.

The elements within an Adapter Chain can be reconfigured during runtime; it is not necessary to restart any component in the system.