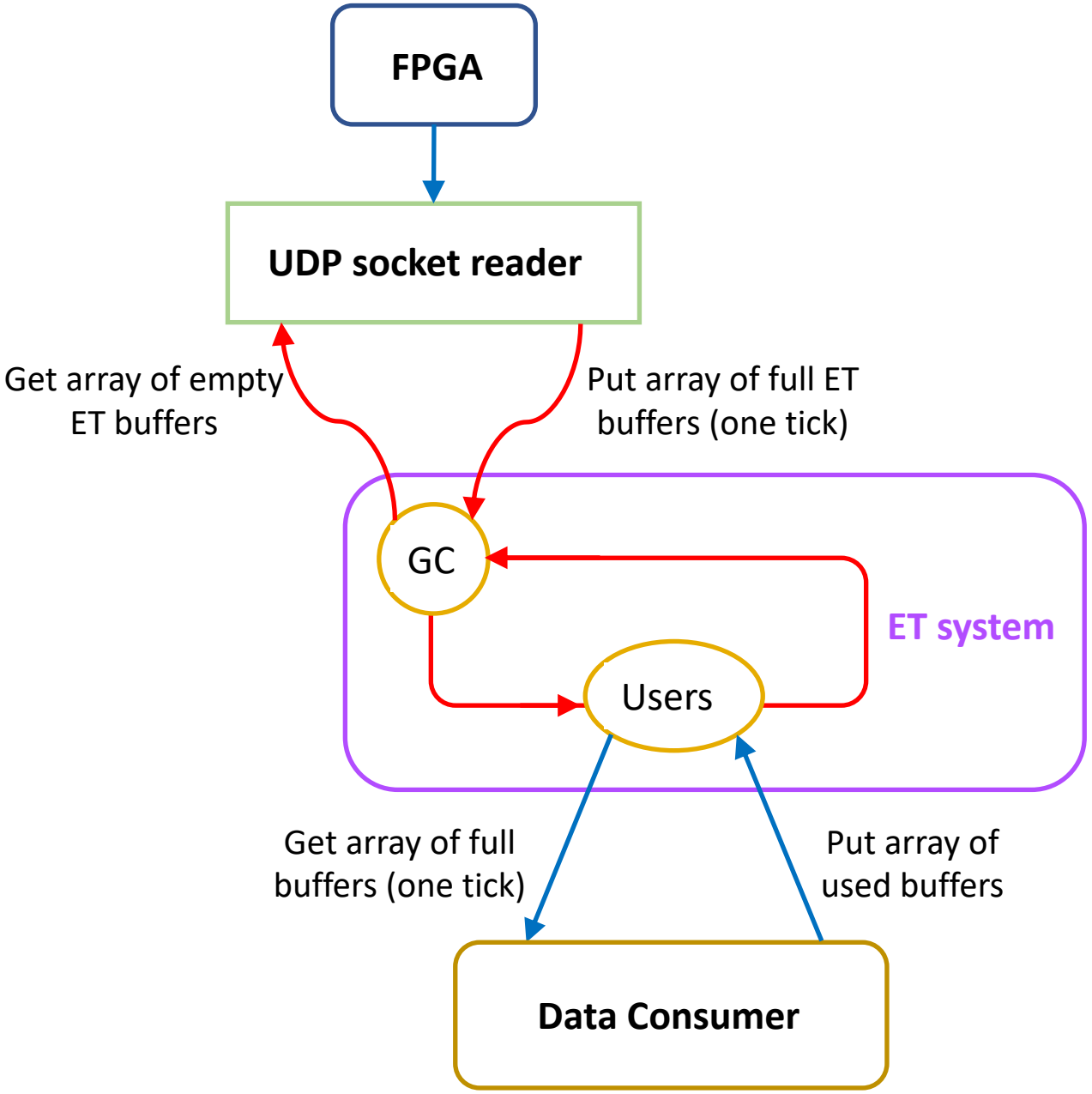


The Event Transport or ET system, based on shared memory, can be used as a data storage FIFO, decoupling the reading thread from data users.

Since we're reading UDP packets, only 1 reading thread can be used. In this scheme, it does not have to allocate memory but can use pre-allocated ET system buffers. These buffers are reused.

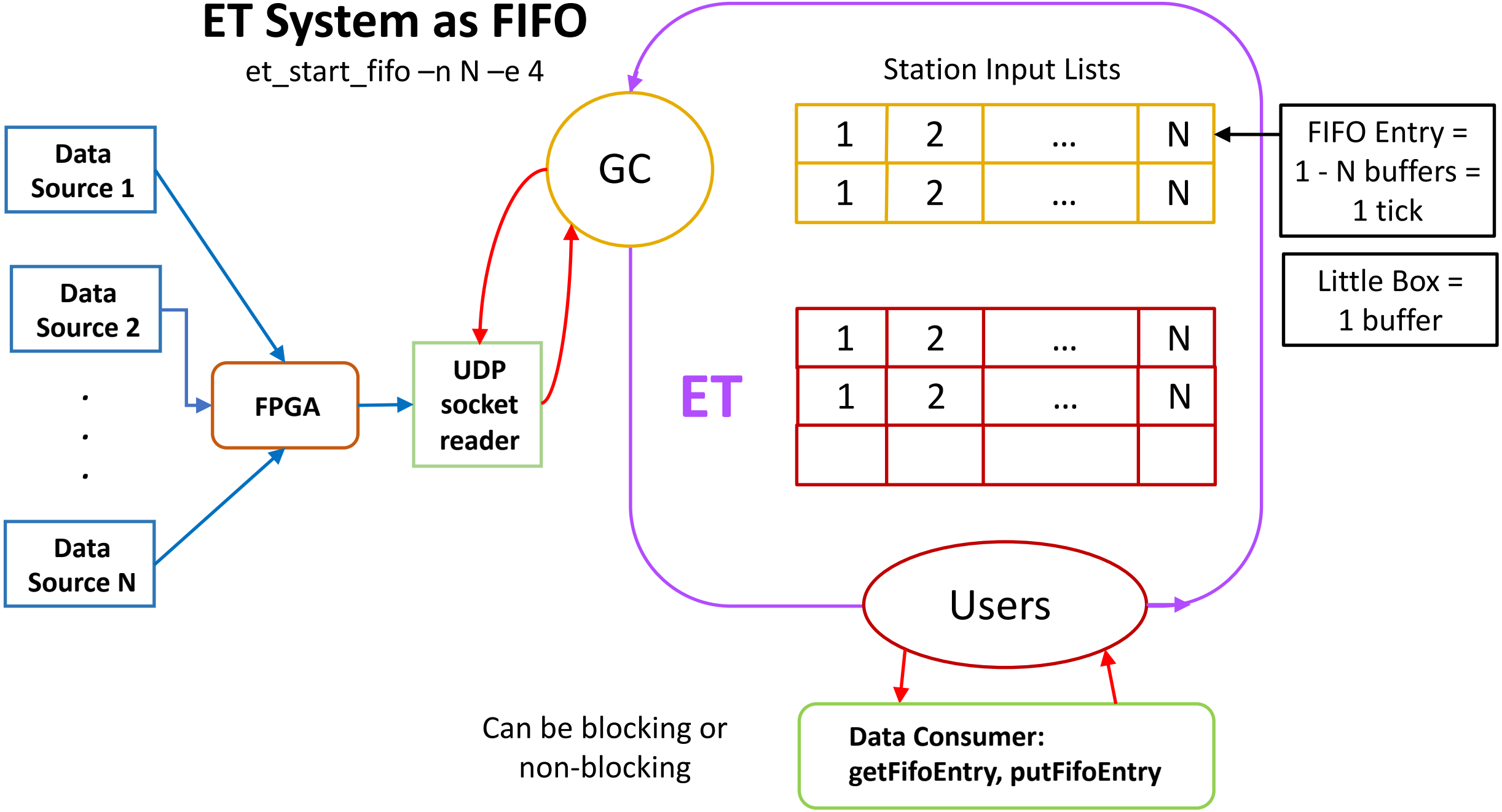
When initially configured, the ET system needs to know how many data sources max will be used.

If there are no (or slow) consumers, the reader may be unable to obtain more buffers and will dump the incoming data.

ET code can be C, C++ or Java.

ET System as FIFO

et_start_fifo –n N –e 4



There is one reading thread which disentangles incoming packets. It can handle:

- ✓ data from multiple sources
- ✓ out-of-order packets
- ✓ overlapping ticks

The reading thread gets a single fifo entry – which is an array of buffers. These buffers exist in shared memory. This fifo entry corresponds to 1 tick.

Thread places reconstructed data from one data source into a single buffer. If there are multiple sources, each source will have its own buffer in a single fifo entry.

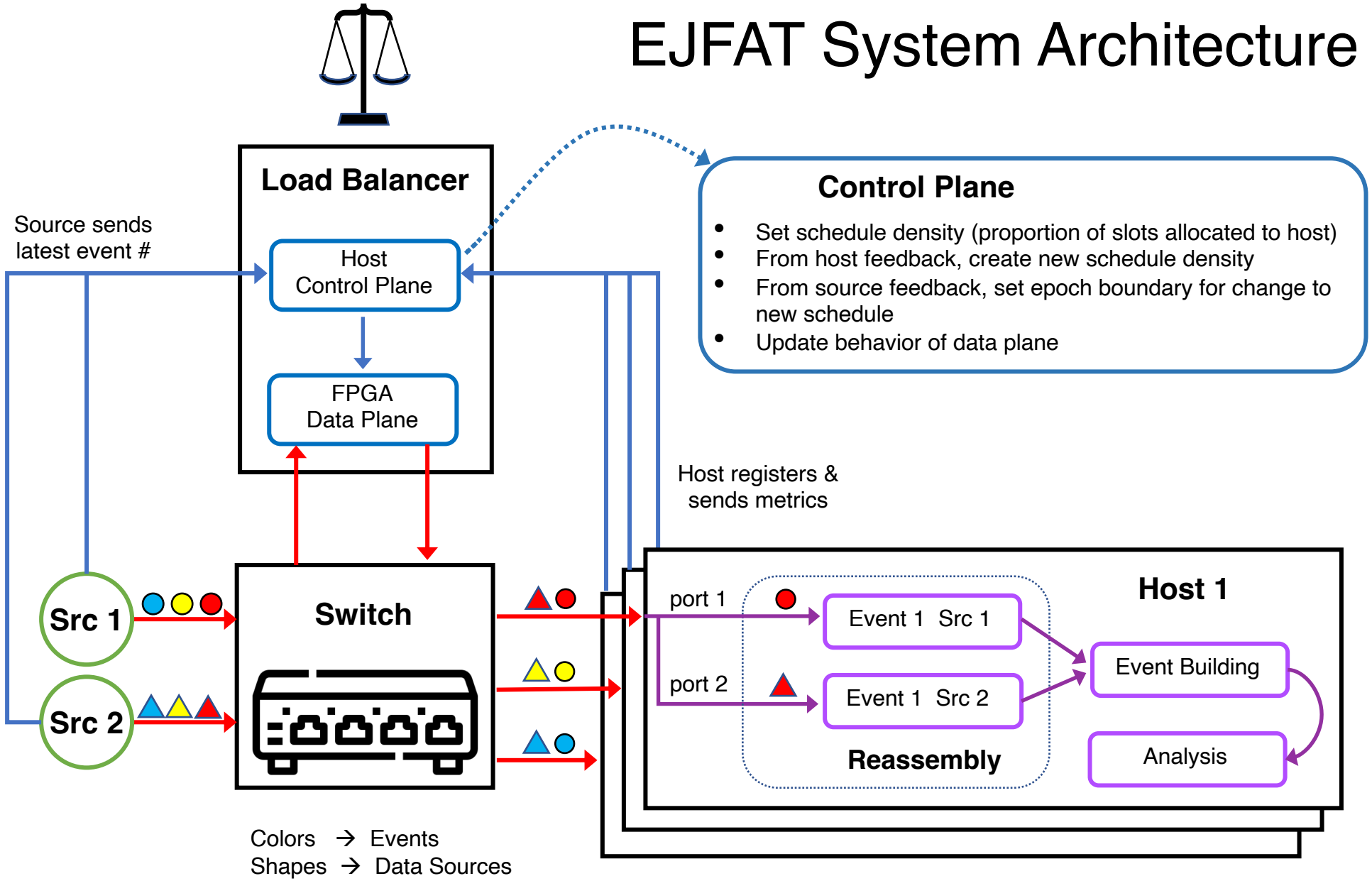
When the data from each source is reconstructed, then the entire fifo entry is returned to the ET system and made available for back-end users, through the “Users” station.

The fifo is by default non-blocking. So if the data users are slow, all incoming packets are read, placed into buffers, then dumped.

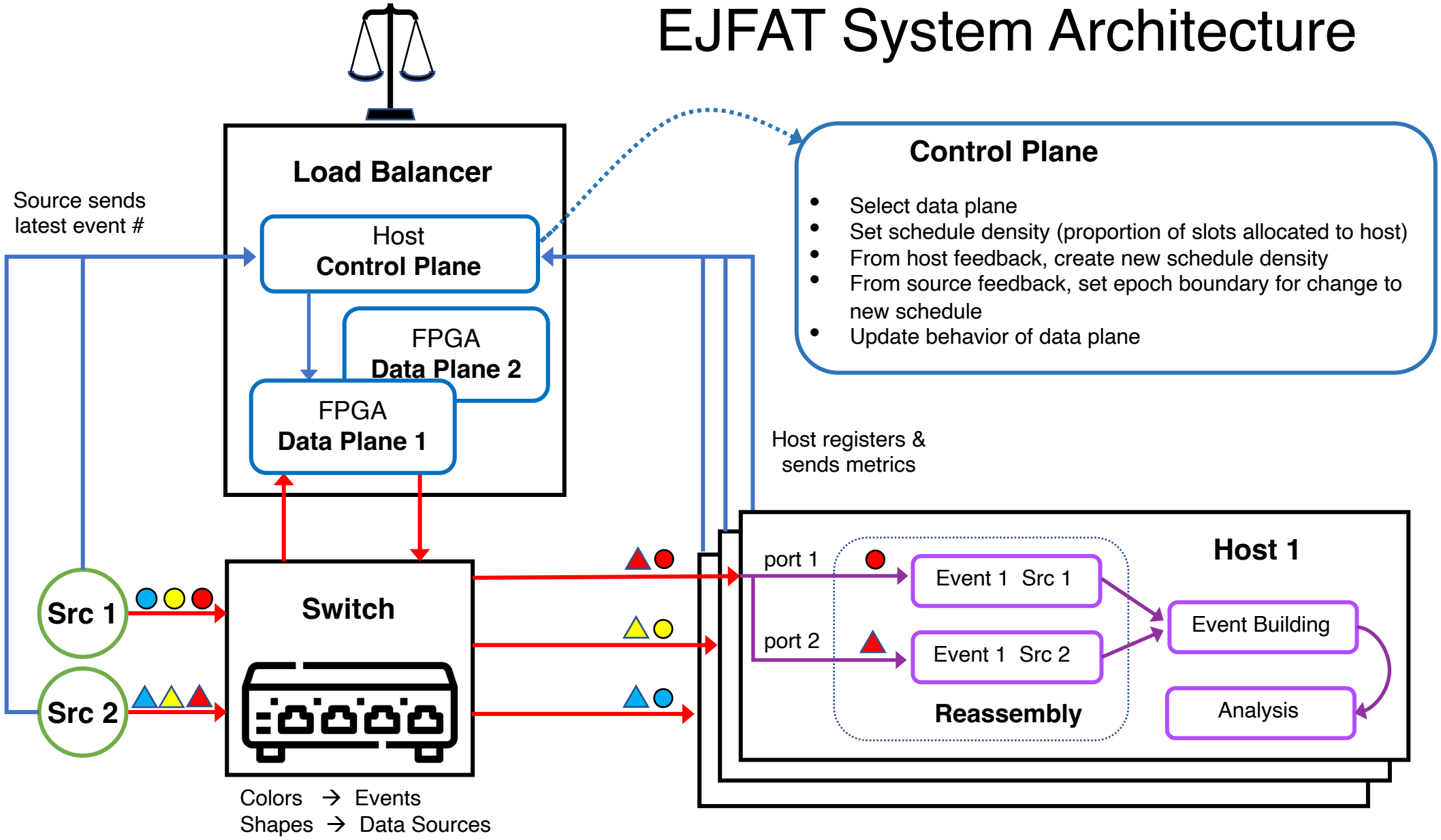
Data users can get metadata from each buffer. Among other things, the metadata tells the user if a buffer has data and what the data source id is.

In the previous diagram, “GC” just stands for GrandCentral station and contains a list of available fifo entries/buffers for placing new data in. The “Users” station contains a list of available fifo entries/buffers with valid data.

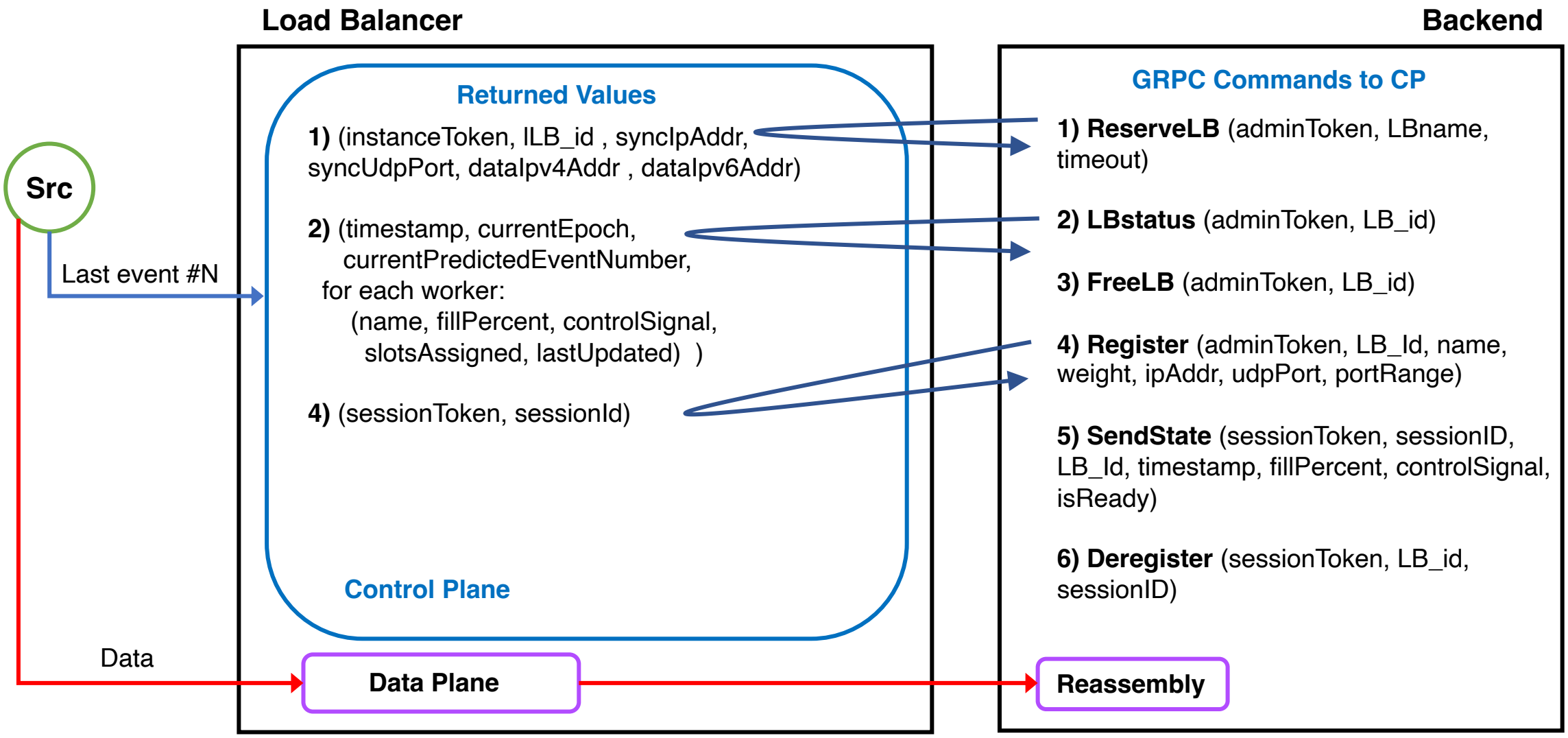
EJFAT System Architecture



EJFAT System Architecture



EJFAT CP Communications



EJFAT Event Reassembly