GNU Radio 4.0: Packet<T> → Tensor<T> → DataSet<T> - 2nd Round

handling of chunked data + meta-information (↔ transient recording, data-multiplexing, non-RF signals e.g. matrix/image/tensors…)

(1) *#6182* expressing signal measurement errors/confidence intervals

(2) #---- synchronising data (streams) from different flow-graphs and nodes

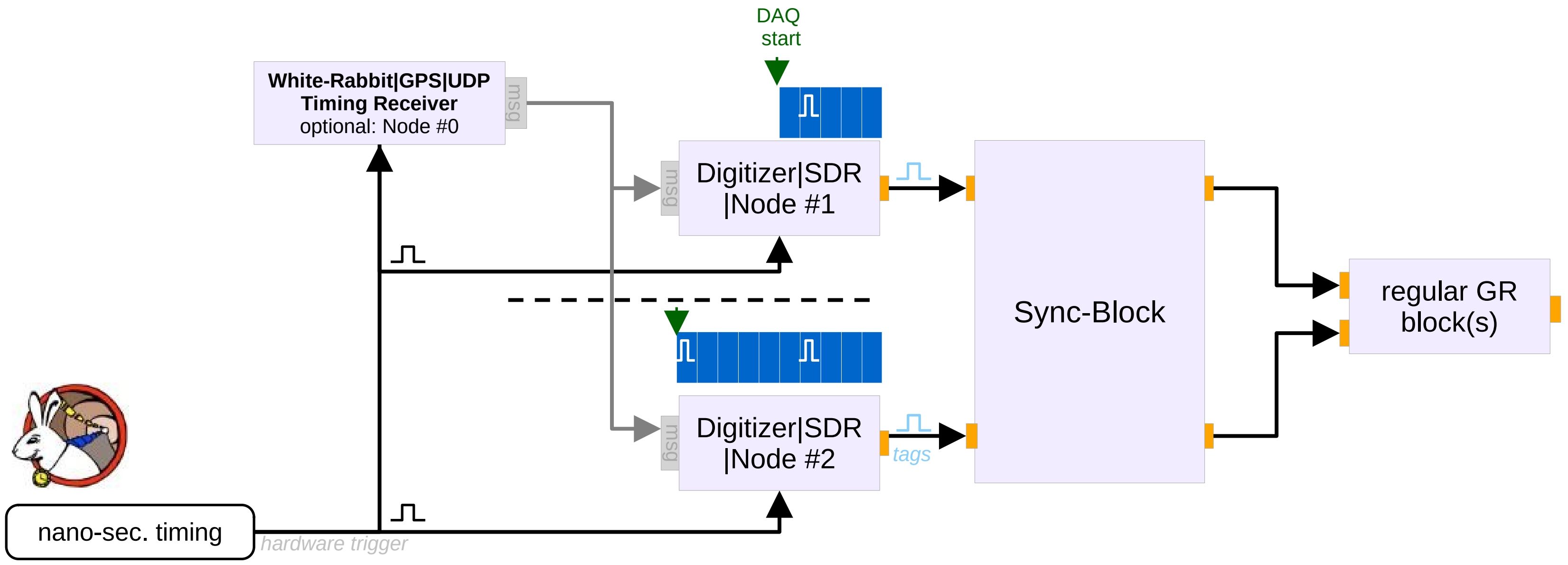
(3) #---- flow-graph post-processing of chunked data → new 'meta-data + `std::mdspan<>`'-like data type

A. Krimm, Ralph J. Steinhagen, GNU Radio Meeting – online, 2023-03-23
with input from Josh Mormon & John Sallay

Finland France Germany India Poland Romania Russia Slovenia Sweden UK

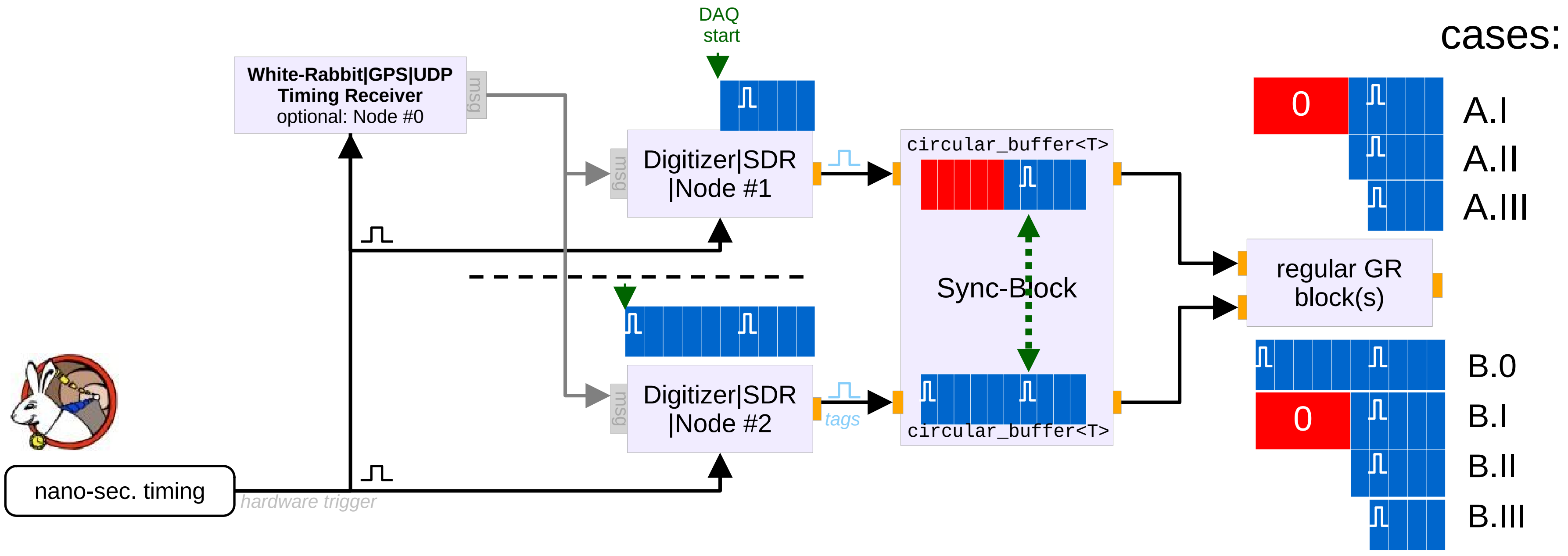
(2) # - - - - synch. data (streams) from different flow-graphs & nodes

- MIMO signals – if possible – are usually synchronised via each RX channel being on the same DAQ system
- not always possible: limited #channel per device (↔costs), largely spacially distributed DAQs (e.g. FAIR: 4.5 km)
- real-world problem: (re-)synchronise physically/spacially distributed sources within the same flow-graph
- failure cases to consider: ‘reconnecting/restarting SDRs/nodes’, ‘no data’ & time-outs, … clock-drifts, transmission delays, ...



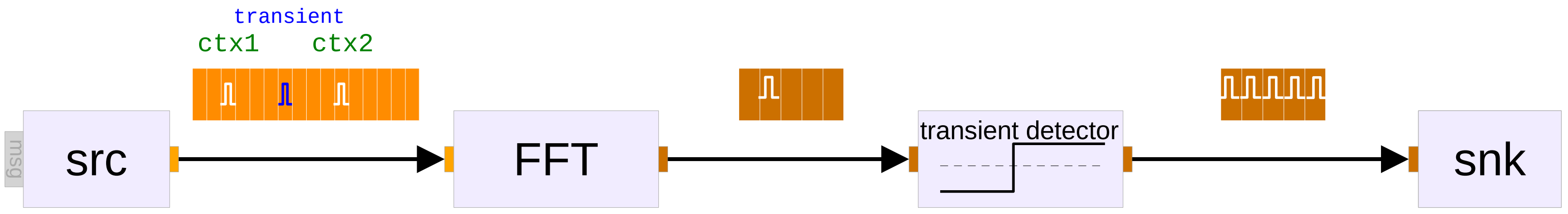
(2) #- - - - synch. data (streams) from different flow-graphs & nodes

- MIMO signals – if possible – are usually synchronised via each RX channel being on the same DAQ system
- not always possible: limited #channel per device (↔costs), largely spacially distributed DAQs (e.g. FAIR: 4.5 km)
- real-world problem: (re-)synchronise physically/spacially distributed sources within the same flow-graph
- failure cases to consider: ‘reconnecting/restarting SDRs/nodes’, ‘no data’ & time-outs, … clock-drifts, transmission delays, ...

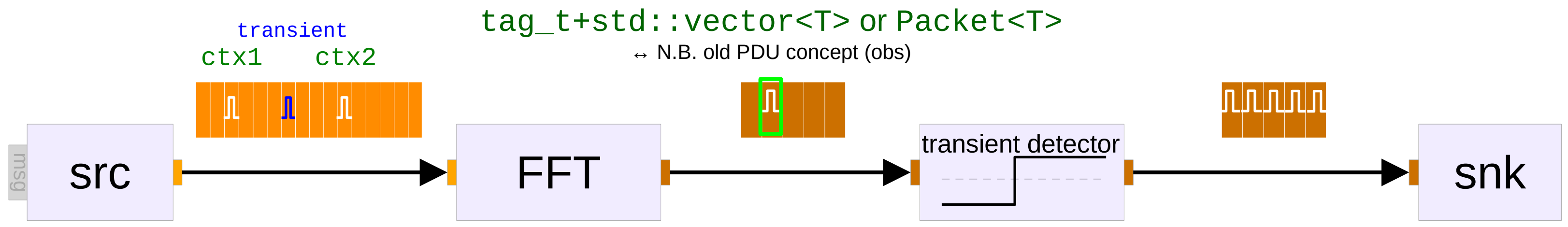


solved through standardised ‘tag_t’s;
TRIGGER_NAME, TRIGGER_TIME,TRIGGER_OFFSET

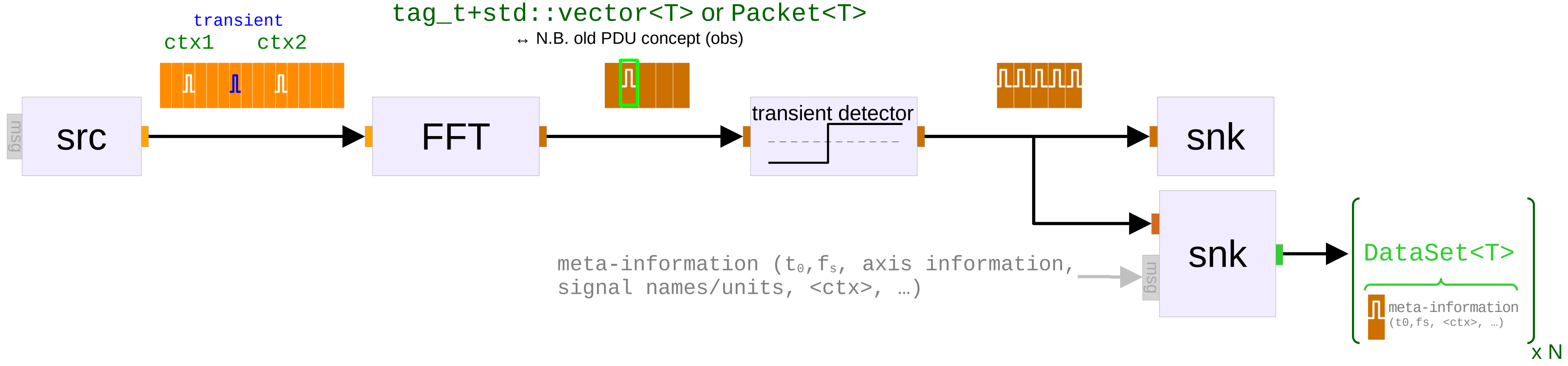
(3) #- - - - flow-graph post-processing of chunked data → new DataSet<T>



(3) #- - - - flow-graph post-processing of chunked data → new DataSet<T>

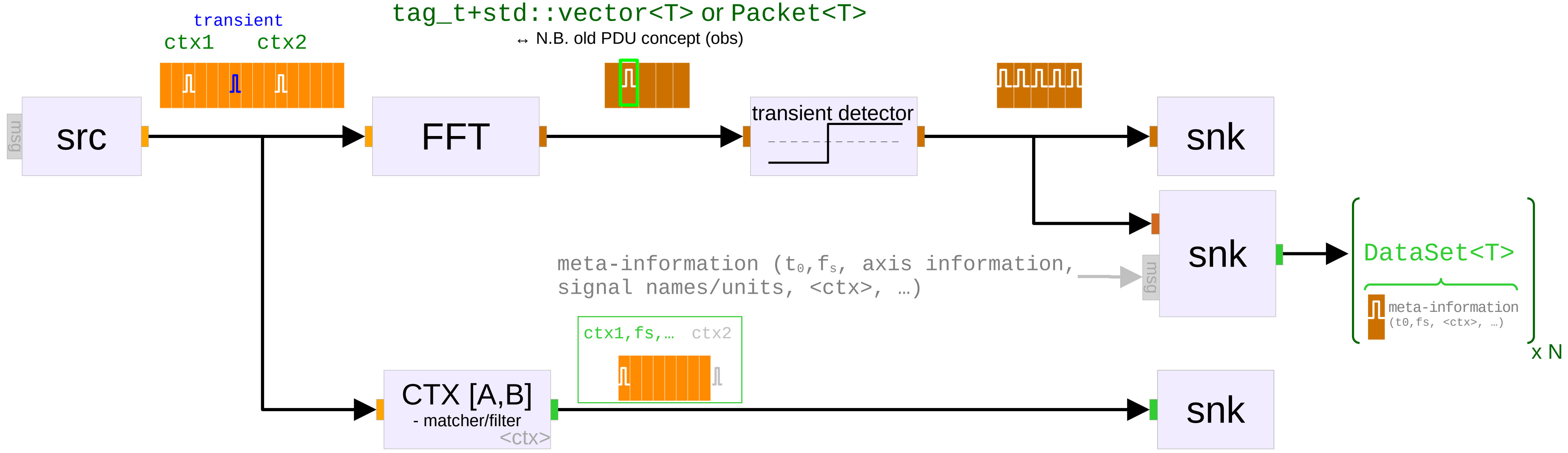

transient
ctx1
ctx2
tag_t+std::vector<T> or Packet<T>
↔ N.B. old PDU concept (obs)
msg
src
FFT
transient detector
snk

(3) #- - - - flow-graph post-processing of chunked data → new DataSet<T>

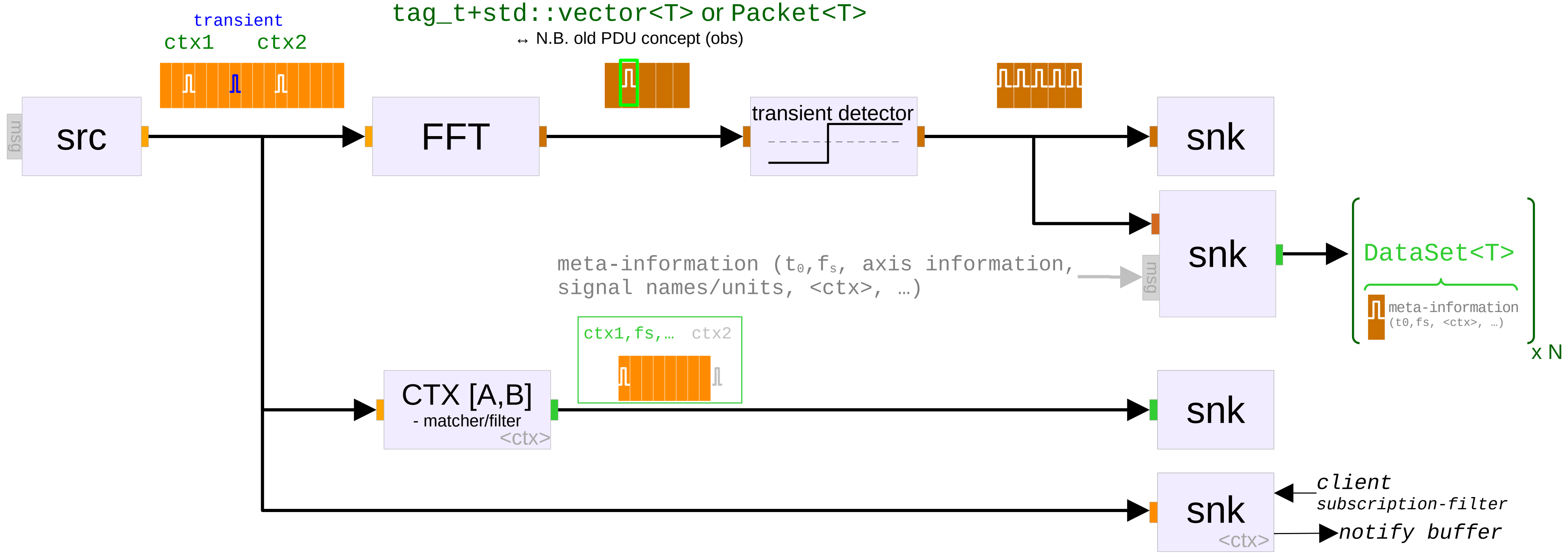


transient
ctx1
ctx2
tag_t+std::vector<T> or Packet<T>
↔ N.B. old PDU concept (obs)
msg
src
FFT
transient detector
snk
snk
msg
meta-information (t₀,fₛ, axis information, signal names/units, <ctx>, …)
DataSet<T>
meta-information (t0,fs, <ctx>, …)
x N

(3) #---- flow-graph post-processing of chunked data → new DataSet<T>

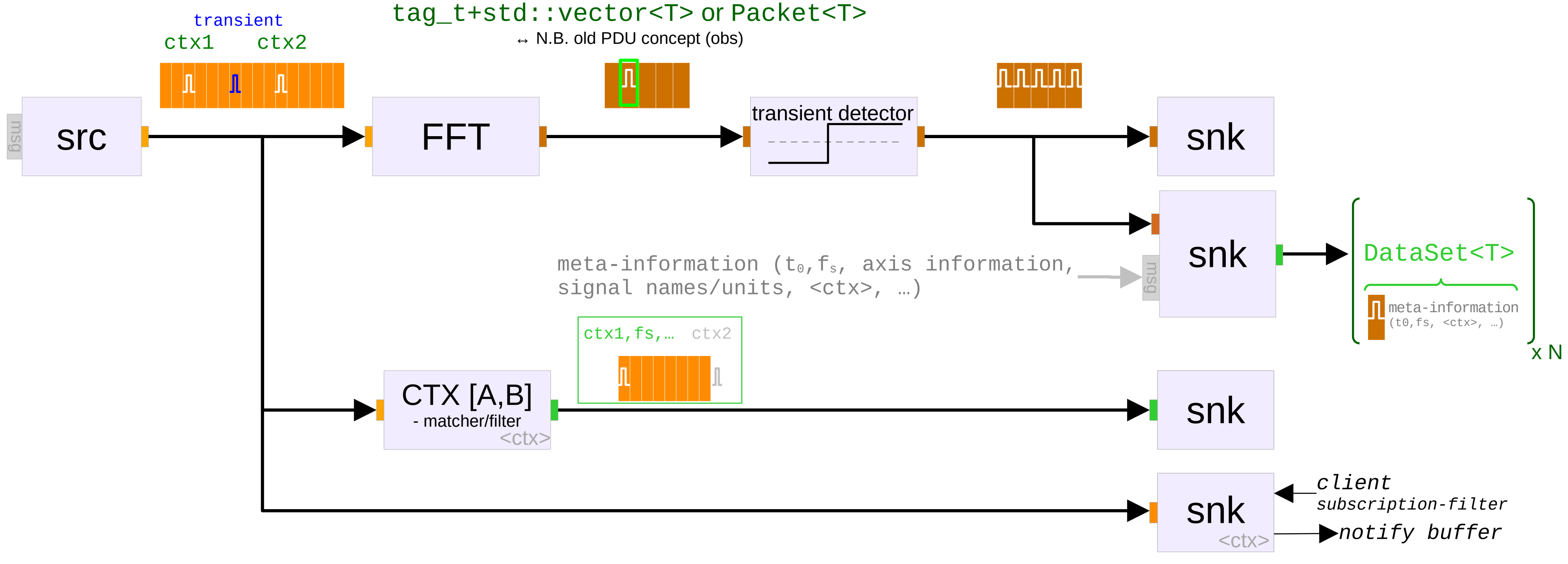


(3) #- - - - flow-graph post-processing of chunked data → new DataSet<T>

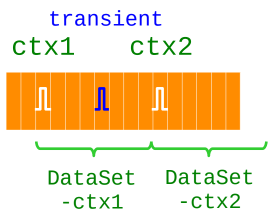


transient
ctx1
ctx2
tag_t+std::vector<T> or Packet<T>
↔ N.B. old PDU concept (obs)
msg
src
FFT
transient detector
snk
snk
snk
snk
meta-information (t₀,fₛ, axis information, signal names/units, <ctx>, …)
msg
DataSet<T>
meta-information (t0,fs, <ctx>, …)
x N
ctx1,fs,…
ctx2
CTX [A,B]
- matcher/filter
<ctx>
<ctx>
client subscription-filter
notify buffer

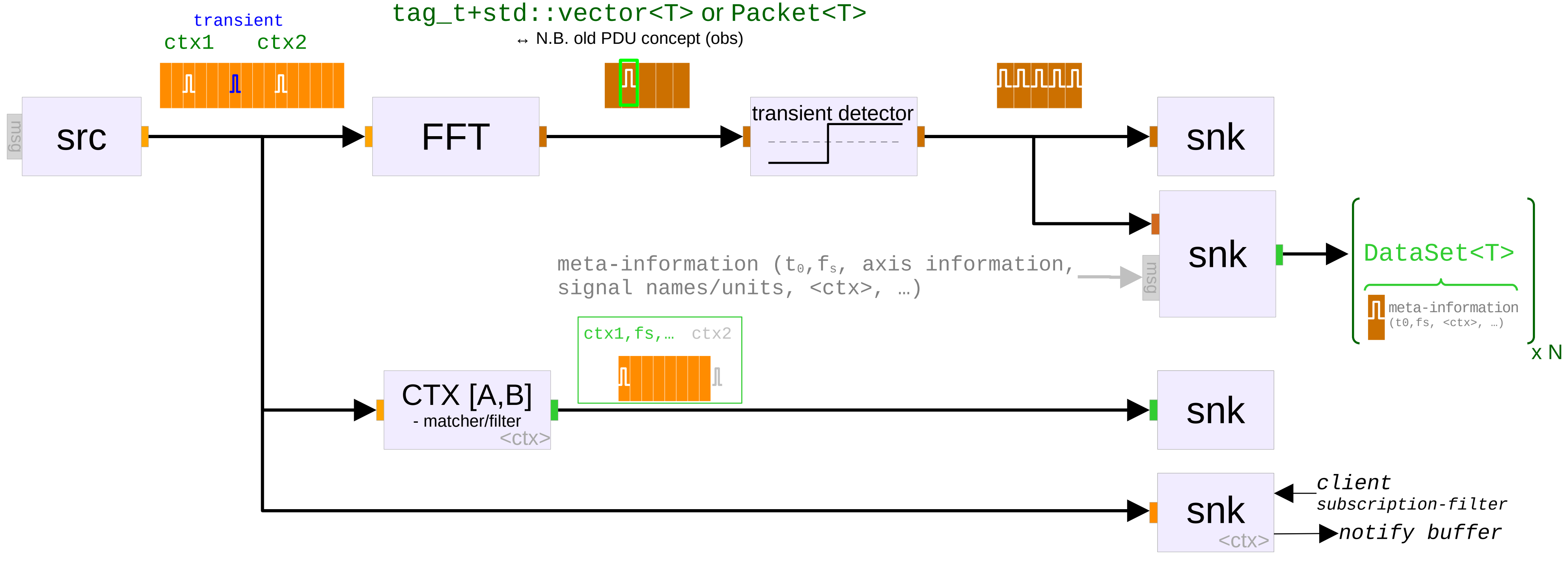
(3) #- - - - flow-graph post-processing of chunked data → new DataSet<T>



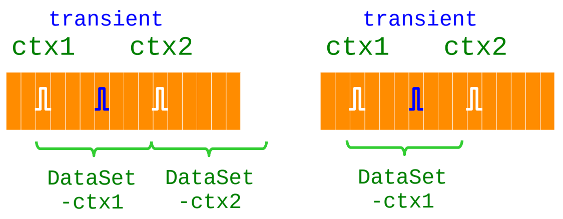
filter modes: ‘multiplexed’



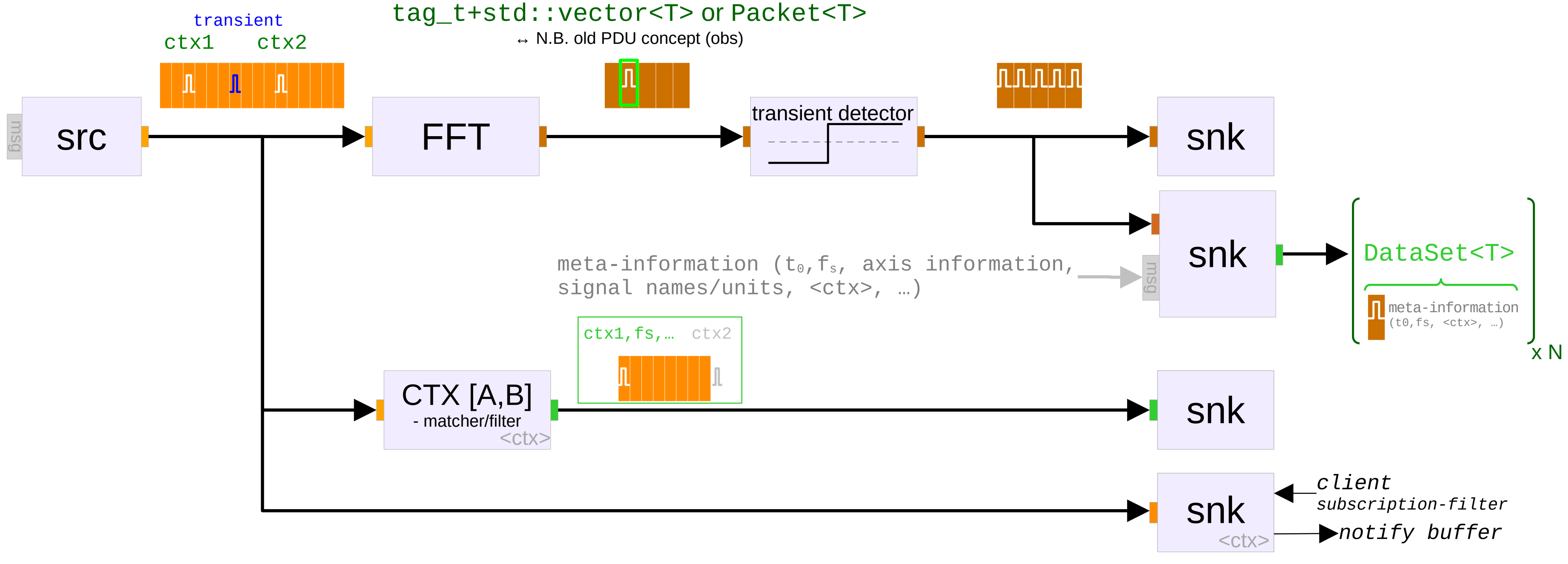
(3) #- - - - flow-graph post-processing of chunked data → new DataSet<T>


transient
ctx1 ctx2
tag_t+std::vector<T> or Packet<T>
↔ N.B. old PDU concept (obs)
msg
src
FFT
transient detector
snk
snk
msg
DataSet<T>
meta-information (t0,fs, <ctx>, …)
x N
meta-information (t₀,fₛ, axis information, signal names/units, <ctx>, …)
ctx1,fs,… ctx2
CTX [A,B]
- matcher/filter
<ctx>
snk
snk
<ctx>
client subscription-filter
notify buffer

filter modes: 'multiplexed' 'multiplexed'
on ctx1-only



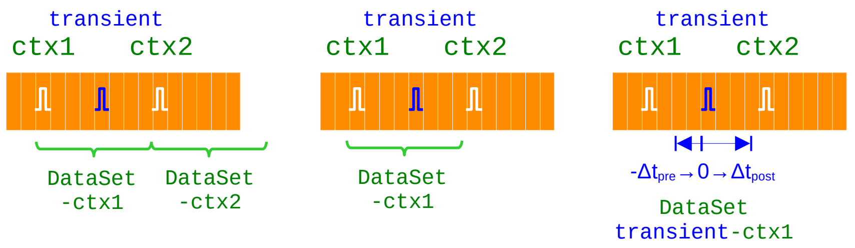
(3) #- - - - flow-graph post-processing of chunked data → new DataSet<T>



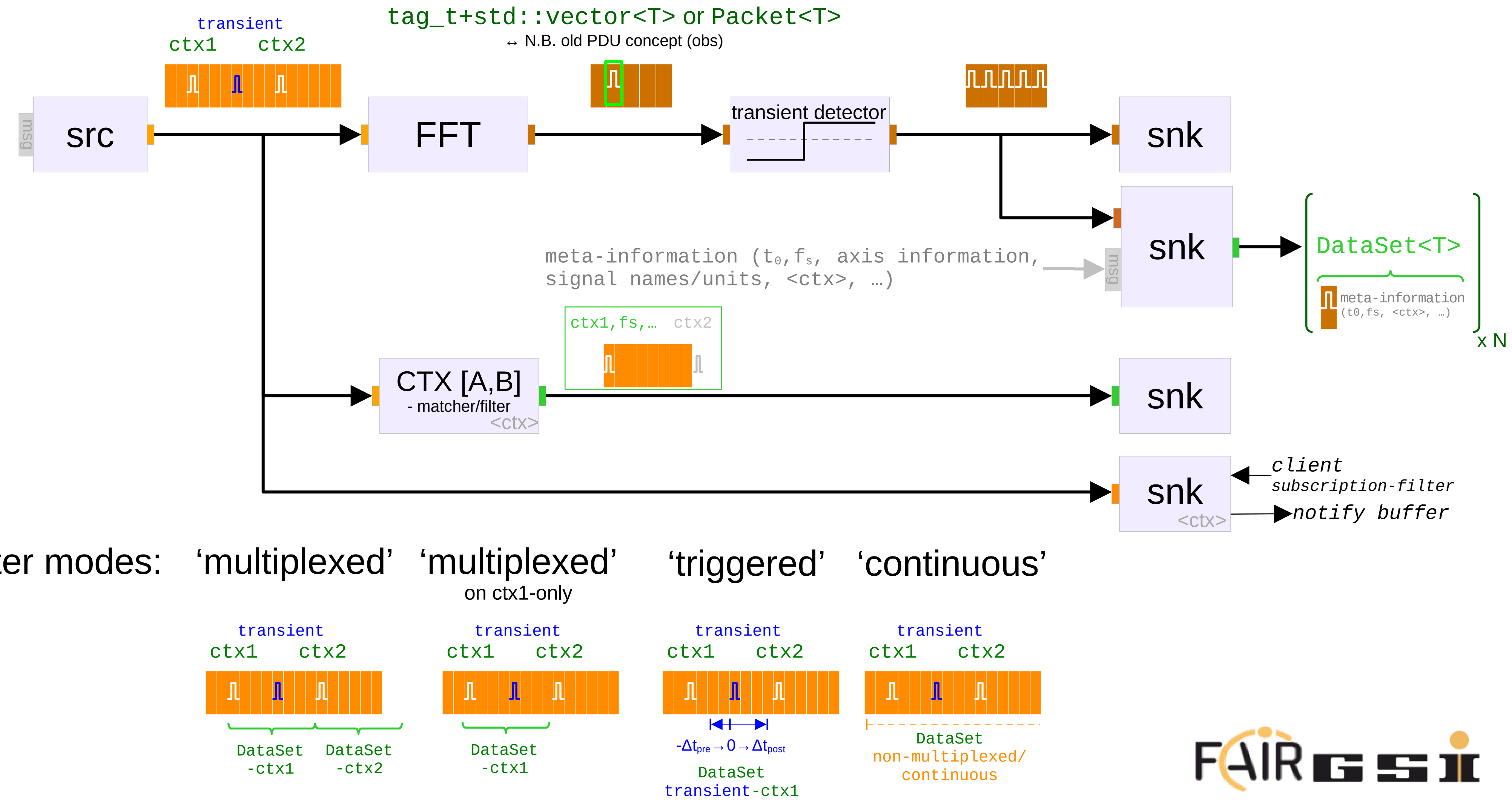
transient
ctx1
ctx2
tag_t+std::vector<T> or Packet<T>
↔ N.B. old PDU concept (obs)
msg
src
FFT
transient detector
snk
snk
msg
meta-information (t₀,fₛ, axis information, signal names/units, <ctx>, …)
DataSet<T>
meta-information (t0,fs, <ctx>, …)
x N
ctx1,fs,…
ctx2
CTX [A,B]
- matcher/filter
<ctx>
snk
snk
<ctx>
client subscription-filter
notify buffer

filter modes: 'multiplexed' 'multiplexed' 'triggered'

on ctx1-only



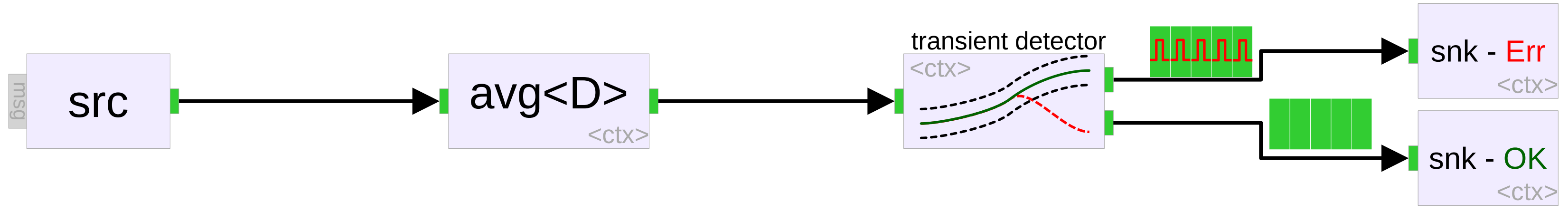
(3) #- - - - flow-graph post-processing of chunked data → new DataSet<T>



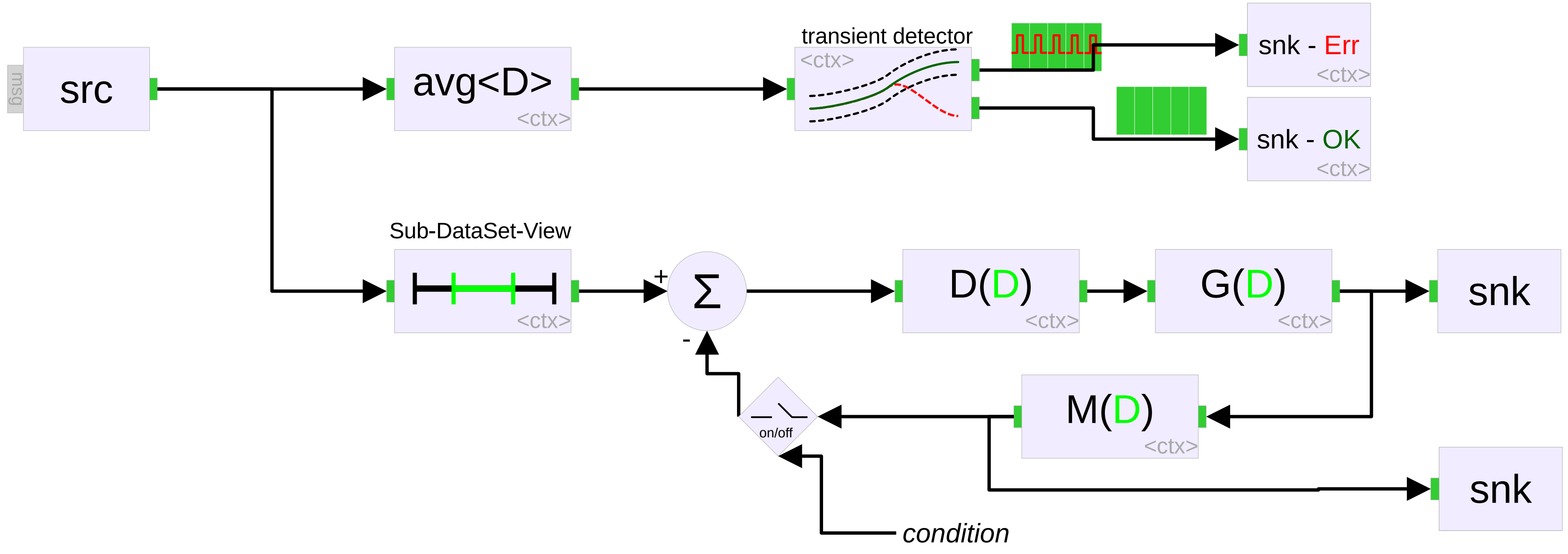
transient
ctx1 ctx2
tag_t+std::vector<T> or Packet<T>
↔ N.B. old PDU concept (obs)
msg
src
FFT
transient detector
snk
snk
snk
snk
meta-information (t₀,fₛ, axis information, signal names/units, <ctx>, …)
msg
DataSet<T>
meta-information (t0,fs, <ctx>, …)
x N
ctx1,fs,… ctx2
CTX [A,B]
- matcher/filter
<ctx>
client subscription-filter
notify buffer
<ctx>

filter modes: 'multiplexed' 'multiplexed' on ctx1-only 'triggered' 'continuous'

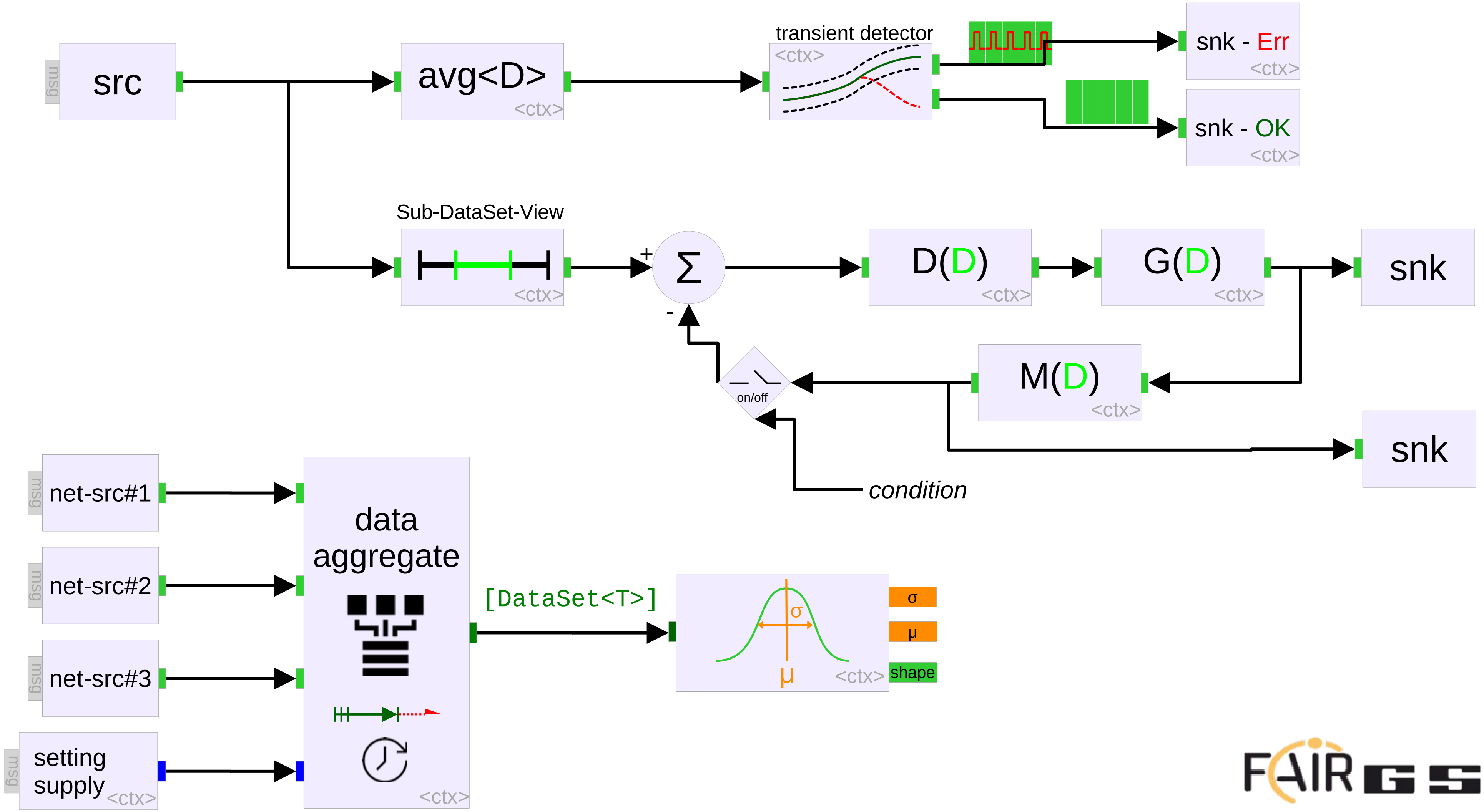
(3) #- - - - flow-graph post-processing of chunked data → new DataSet<T>



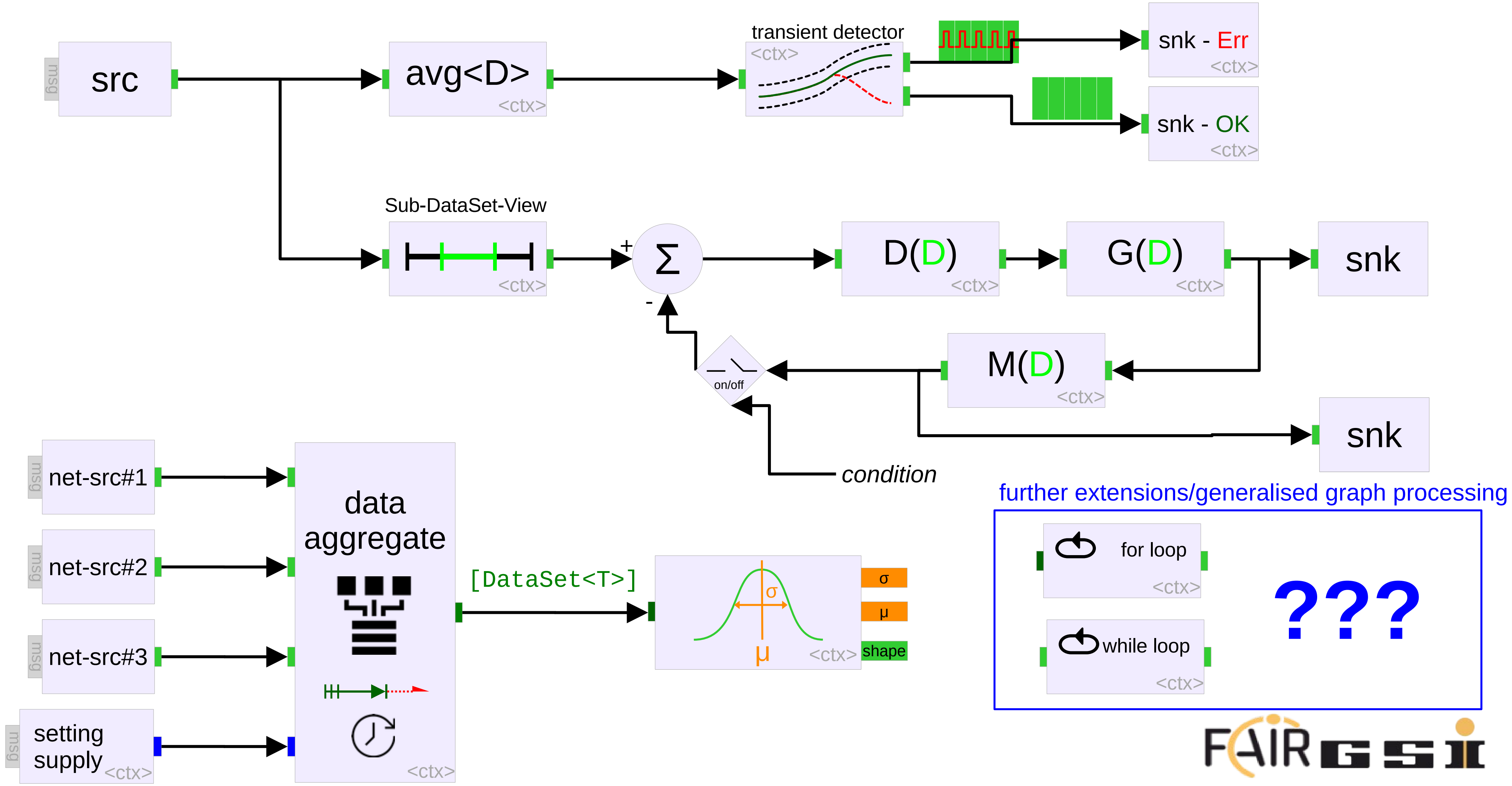
(3) #- - - - flow-graph post-processing of chunked data → new DataSet<T>



(3) # - - - - flow-graph post-processing of chunked data → new DataSet<T>



(3) #- - - - flow-graph post-processing of chunked data → new DataSet<T>



FAIR GSI

(3) #- - - - three new types → Packet<T> → Tensor<T> → DataSet<T>

```cpp
template<typename T>
struct DataSet {



    std::int64_t                                        timestamp = 0; // UTC timestamp [ns]
    // axis layout:
    std::vector<std::string>                            axis_names; // e.g. time, frequency, …
    std::vector<std::string>                            axis_units; // axis base SI-unit
    std::vector<std::vector<T>>                         axis_values; // explicit axis values

    // signal data layout:
    std::vector<std::int32_t>                           extents; // extents[dim0_size, dim1_size, …]
    std::variant<layout_right, layout_left, std::string>  // row-major, column-major, “special”
                                                        layout;
    // signal data storage:
    std::vector<std::string>                            signal_names;  // size = extents[0]
    std::vector<std::string>                            signal_units;  // size = extents[0]
    std::vector<T>                                      signal_values; // size = \PI_i extents[i]
    std::vector<T>                                      signal_errors; // size = \PI_i extents[i]
    std::vector<std::vector<T>>                         signal_ranges; // [[min_0, max_0], [min_1, …]

    // meta data
    std::vector<std::map<std::string, pmt::pmtv>>  meta_information;
    std::vector<std::map<std::int64_t, pmt::pmtv>> timing_events; // ↔ gr::tag_t
    // [..] constructors, accessors, ...
};
```

(3) #- - - - three new types → Packet<T> → Tensor<T> → DataSet<T>

```
template<typename T>
struct DataSet {
 Packet
 ▌
 ▌  std::int64_t                                   timestamp = 0; // UTC timestamp [ns]
    // axis layout:
    std::vector<std::string>                       axis_names; // e.g. time, frequency, …
    std::vector<std::string>                       axis_units; // axis base SI-unit
    std::vector<std::vector<T>>                    axis_values; // explicit axis values

    // signal data layout:
    std::vector<std::int32_t>                      extents; // extents[dim0_size, dim1_size, …]
    std::variant<layout_right, layout_left, std::string>  // row-major, column-major, “special”
                                                   layout;
    // signal data storage:
    std::vector<std::string>                       signal_names;  // size = extents[0]
    std::vector<std::string>                       signal_units;  // size = extents[0]
 ▌  std::vector<T>                                 signal_values; // size = \PI_i extents[i]
    std::vector<T>                                 signal_errors; // size = \PI_i extents[i]
    std::vector<std::vector<T>>                    signal_ranges; // [[min_0, max_0], [min_1, …]

    // meta data
 ▌  std::vector<std::map<std::string, pmt::pmtv>>  meta_information;
    std::vector<std::map<std::int64_t, pmt::pmtv>> timing_events; // ↔ gr::tag_t
    // [..] constructors, accessors, ...
};
```

(3) #- - - - three new types → Packet<T> → Tensor<T> → DataSet<T>

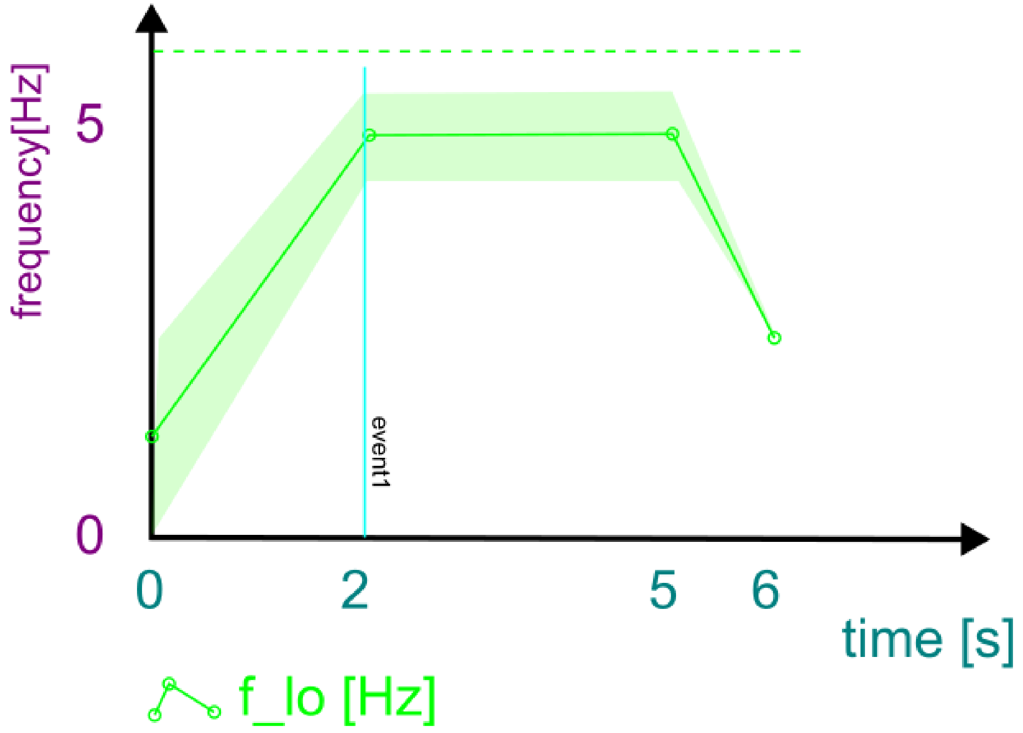
```
template<typename T>
struct DataSet {
  Packet
   Tensor
    std::int64_t                                   timestamp = 0; // UTC timestamp [ns]
    // axis layout:
    std::vector<std::string>                       axis_names; // e.g. time, frequency, …
    std::vector<std::string>                       axis_units; // axis base SI-unit
    std::vector<std::vector<T>>                    axis_values; // explicit axis values

    // signal data layout:
    std::vector<std::int32_t>                      extents; // extents[dim0_size, dim1_size, …]
    std::variant<layout_right, layout_left, std::string>  // row-major, column-major, “special”
                                                   layout;
    // signal data storage:
    std::vector<std::string>                       signal_names;  // size = extents[0]
    std::vector<std::string>                       signal_units;  // size = extents[0]
    std::vector<T>                                 signal_values; // size = \PI_i extents[i]
    std::vector<T>                                 signal_errors; // size = \PI_i extents[i]
    std::vector<std::vector<T>>                    signal_ranges; // [[min_0, max_0], [min_1, …]

    // meta data
    std::vector<std::map<std::string, pmt::pmtv>>  meta_information;
    std::vector<std::map<std::int64_t, pmt::pmtv>> timing_events; // ↔ gr::tag_t
    // [..] constructors, accessors, ...
};
```

(3) #- - - - three new types → Packet<T> → Tensor<T> → DataSet<T>

```cpp
template<typename T>
struct DataSet { // – numeric/measurement based data (e.g. generation of graphs/plotting)
 Packet
  Tensor
    std::int64_t                                   timestamp = 0; // UTC timestamp [ns]
    // axis layout:
    std::vector<std::string>                       axis_names; // e.g. time, frequency, …
    std::vector<std::string>                       axis_units; // axis base SI-unit
    std::vector<std::vector<T>>                    axis_values; // explicit axis values

    // signal data layout:
    std::vector<std::int32_t>                      extents; // extents[dim0_size, dim1_size, …]
    std::variant<layout_right, layout_left, std::string>  // row-major, column-major, “special”
                                                   layout;

    // signal data storage:
    std::vector<std::string>                       signal_names;  // size = extents[0]
    std::vector<std::string>                       signal_units;  // size = extents[0]
    std::vector<T>                                 signal_values; // size = \PI_i extents[i]
    std::vector<T>                                 signal_errors; // size = \PI_i extents[i]
    std::vector<std::vector<T>>                    signal_ranges; // [[min_0, max_0], [min_1, …]

    // meta data
    std::vector<std::map<std::string, pmt::pmtv>>  meta_information;
    std::vector<std::map<std::int64_t, pmt::pmtv>> timing_events; // ↔ gr::tag_t
    // [..] constructors, accessors, ...
};
```

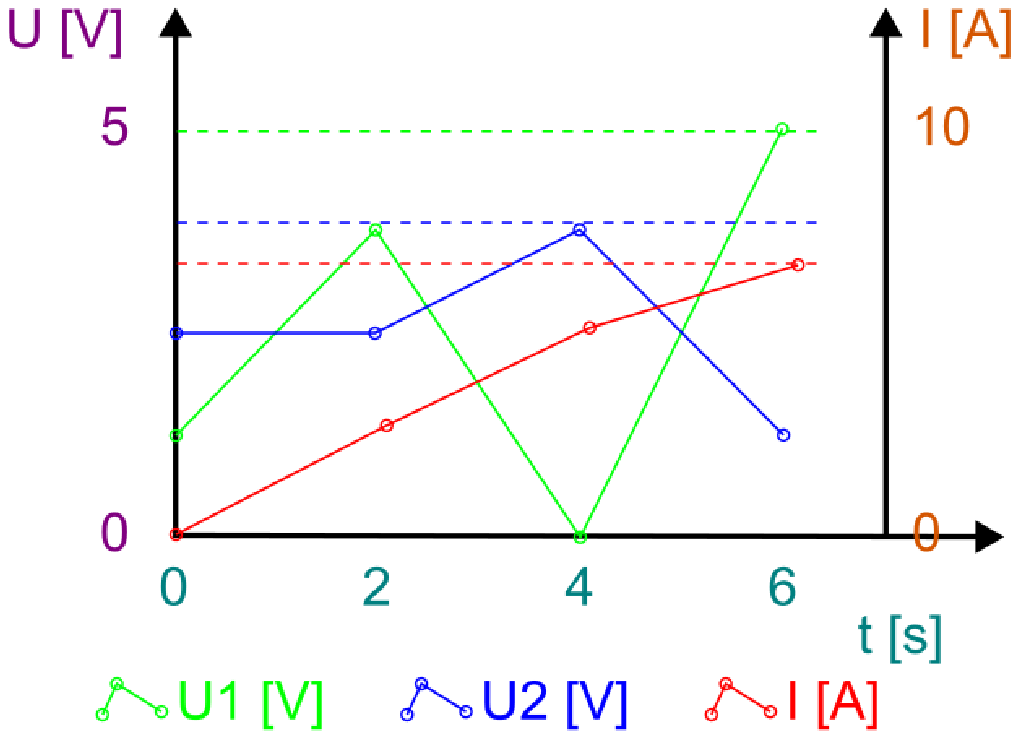
(3) #- - - - DataSet<T> – Example: 1-dim function

```
{
timestamp: 123456789; // [ns]
axis_names: ["time", "frequency"];
axis_units: ["s", "Hz"];
axis_values: [[0, 2, 5, 6], [0, 5]];
extents: [1, 4];
layout: layout_right;
signal_names: ["f_lo"];
signal_units: ["Hz"];
signal_values: [1, 5, 5, 2];
signal_errors: [1, 0.5, 0.5, 0];
signal_ranges: [[0 , 6]];
signal_status: [{"locked": true}];
timing_events: [{123456791: <pmtv>}];
}
```



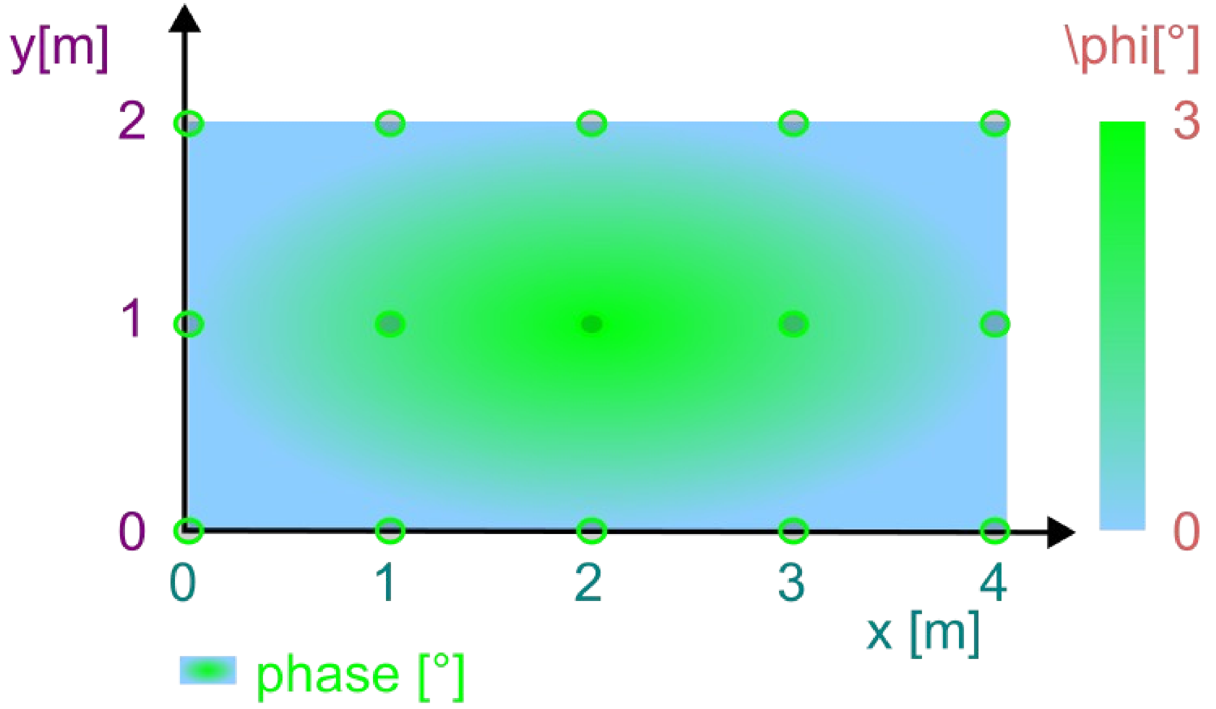
(3) #- - - - DataSet<T> – Example: N=3 x 1-dim function

```
{
timestamp: 123456789; // [ns]
axis_names: ["t", "U", "I"];
axis_units: ["s", "V", "A"];
axis_values: [[0, 6],[0, 5], [0, 10]];
extents: [3, 4];
layout: layout_right;
signal_names: ["U1", "U2", "I"];
signal_units: ["V", "V", "A"];
signal_values: [
  1,3,0,5,
  2,2,3,1,
  0,2,4,5];
signal_errors: [];
signal_ranges: [[0,5],[0,4],[0,5]];
signal_status: [];
timing_events: [];
}
```



(3) #- - - - DataSet<T> – Example: Image/Matrix/Tensor

```
{
timestamp: 123456789; // [ns]
axis_names: ["x", "y", "\phi"];
axis_units: ["m", "m", "°"];
axis_values: [[0, 4], [0, 2]];
extents: [1, 5, 3];
layout: layout_right;
signal_names: ["T"];
signal_units: ["°"];
signal_values: [
  0, 0, 0, 0, 0,
  0, 2, 3, 2, 0,
  0, 0, 0, 0, 0];
signal_errors: [];
signal_ranges: [0,3];
signal_status: [];
timing_events: [];
}
```



(3) #---- Packet<T> → Tensor<T> → DataSet<T> – Summary

- new buffer/ports greatly simplify, unify, and do not need to distinguish between: streams, messages, primitives, PDUs, or (a-)synchronuous communications, → opens new options:
  - gracefully retire old PDUs and other similar types → now: std::vector<T> + tag_t or `Packet<T>`
  - optional choice on ports: single-producer (↔“streaming ports”) vs. multi-producer (→“message ports”)
  - individual block::work() implementer’s choice:
    - synchronous processing → matching number of samples on input ports
    - asynchronous processing → process samples/packets as they arrive

- No ‘one size fits all’ solution need: `Packet<T>`, `Tensor<T>`, and `DataSet<T>`
  - target 80/20 solution, type-safe, and intuitive usage by new users/novice developers
  - single work function can handle all three use-cases
    N.B. use composition and ‘`if constexpr (requires {….})`` rather than inheritance
  - initial strategy: no in-tree blocks other than:
    - generic math operations: add, subtract, multiply, concatenate, FFT, selector, …
    - merge generic OOT blocks merged as needed

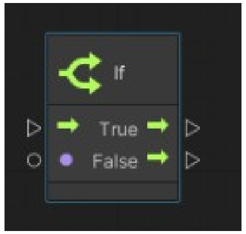
- should differentiate between strict port-types and ‘`pmt::pmtv`’ types
  (N.B. `sizeof(pmtv) == max(sizeof(T)…)`)

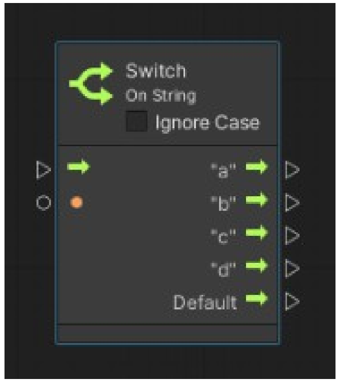
- `do we want support arbitrary user-defined types? e.g. via`
  - `common serialisation protocol (e.g. YAS) or std::map<std::string, pmt::pmtv>`

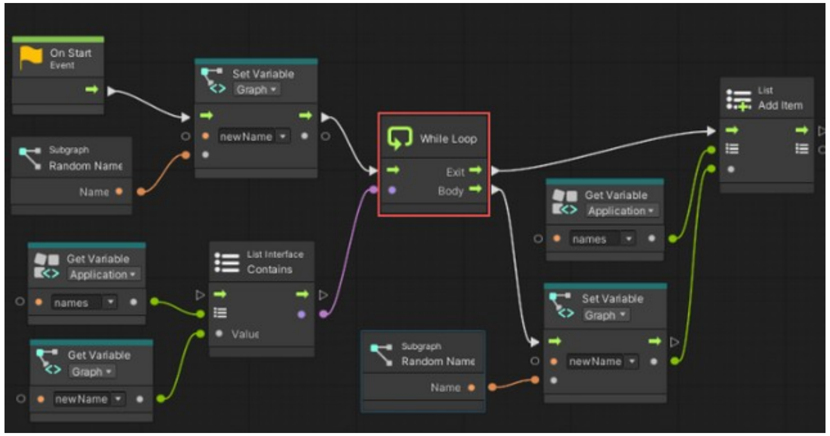
Future Vision/Extension: Inspiration from Unity Control Node …

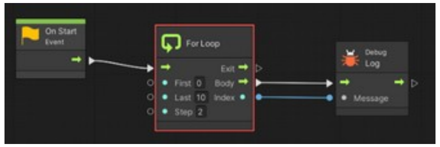
Basic Scripting of more complex signal flow/processing mechanisms

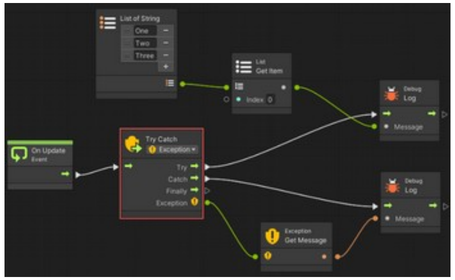
- https://docs.unity3d.com/Packages/com.unity.visualscripting@1.8/manual/vs-control.html

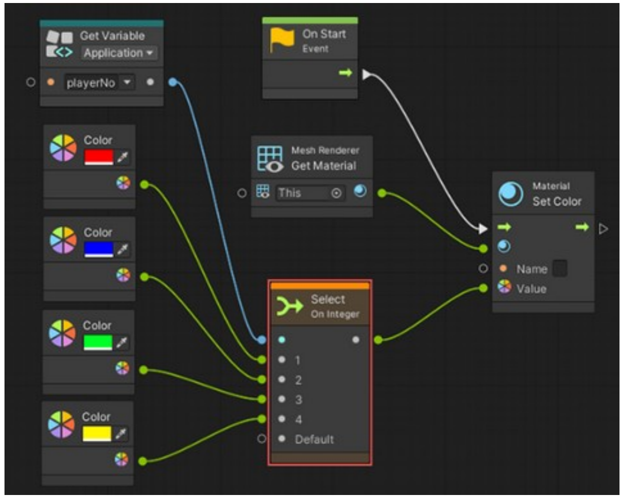

If
True
False

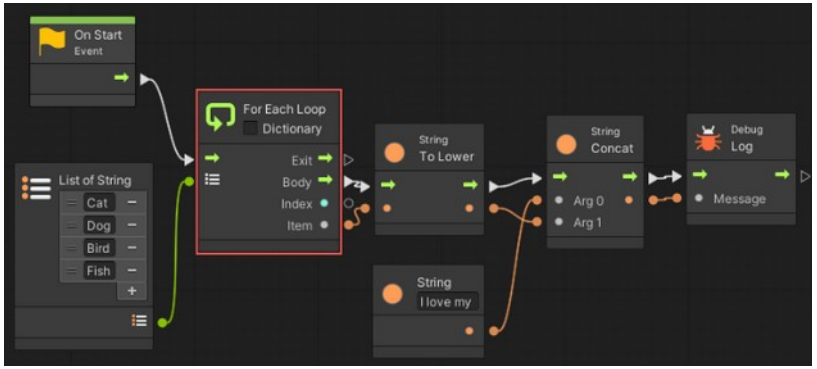

Switch
On String
Ignore Case
"a"
"b"
"c"
"d"
Default

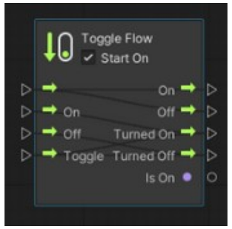

On Start
Event
Set Variable
Graph
newName
While Loop
Exit
Body
List
Add Item
Subgraph
Random Name
Name
Get Variable
Application
names
List Interface
Contains
Value
Get Variable
Graph
newName
Get Variable
Application
names
Set Variable
Graph
newName
Subgraph
Random Name
Name

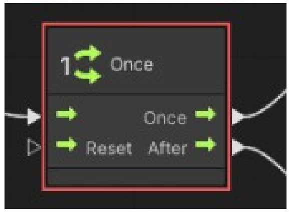

On Start
Event
For Loop
Exit
First 0
Body
Last 10
Index
Step 2
Debug
Log
Message

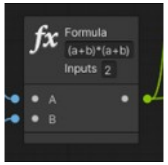

List of String
One
Two
Three
List
Get Item
Index 0
Debug
Log
Message
On Update
Event
Try Catch
Exception
Try
Catch
Finally
Exception
Debug
Log
Message
Exception
Get Message


Get Variable
Application
playerNo
On Start
Event
Color
Mesh Renderer
Get Material
This
Material
Set Color
Name
Value
Color
Color
Select
On Integer
1
2
3
4
Default
Color


On Start
Event
For Each Loop
Dictionary
Exit
Body
Index
Item
List of String
Cat
Dog
Bird
Fish
String
To Lower
String
Concat
Arg 0
Arg 1
Debug
Log
Message
String
I love my


Toggle Flow
Start On
On
On
Off
Off
Turned On
Toggle
Turned Off
Is On


Once
Once
Reset
After


Formula
(a+b)*(a+b)
Inputs 2
A
B