DAQ/FLES Status

Focus on new developments

Dirk Hutter
hutter@compeng.uni-frankfurt.de

Jan de Cuveland
cuveland@compeng.uni-frankfurt.de

FIAS Frankfurt Institute for Advanced Studies
Goethe-Universität Frankfurt am Main, Germany

CBM

CBM Collaboration Meeting
2025-10-21

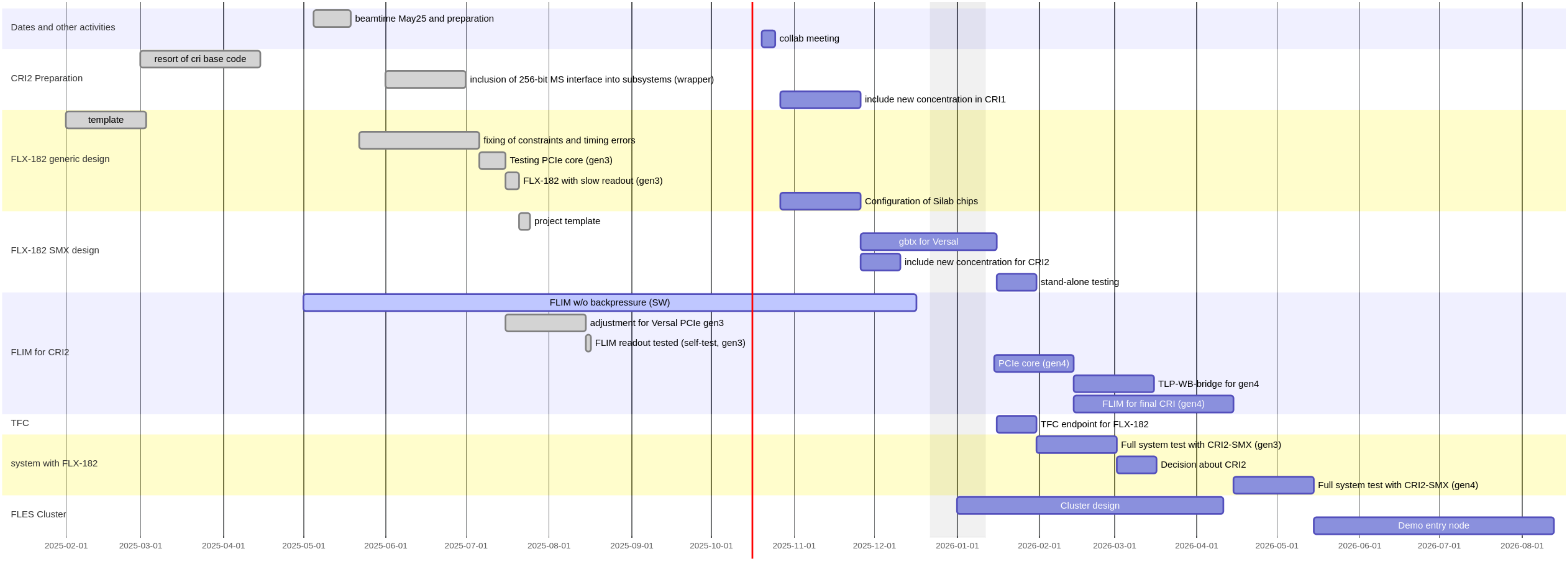
Recent DAQ activities

- Main focus: towards **CRI2 @ SIS100**
- Multiple FLX-182 test setups established
- CRI2 setup at CERN (ALICE DAQ lab)
  - Installed in Feb 2025, decommissioned in Sep 2025
  - Fitted with one FLX-182 borrows from ATLAS
  - Possible future cooperation with ALICE on testing the FLX-155 at CERN
- CRI2 setup at GSI (devel13)
  - Installed Jun 2025, replacement for bricked card Sep 2025
  - BNL team replaced card team and helps to understand what went wrong

FLX-182 setup for CBM at ALICE / CERN

FLX-182 setup at GSI

CRI2 development and testing

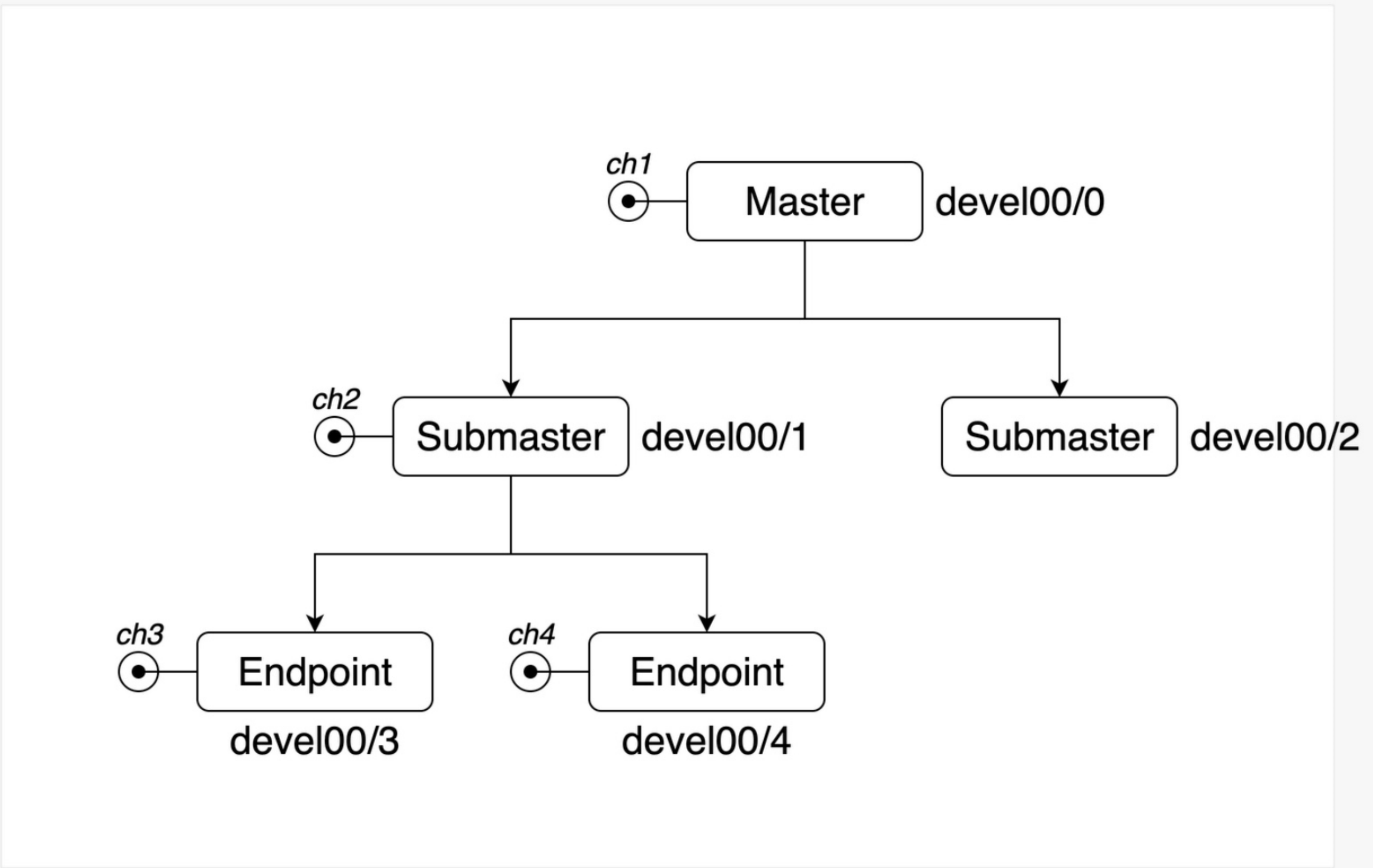


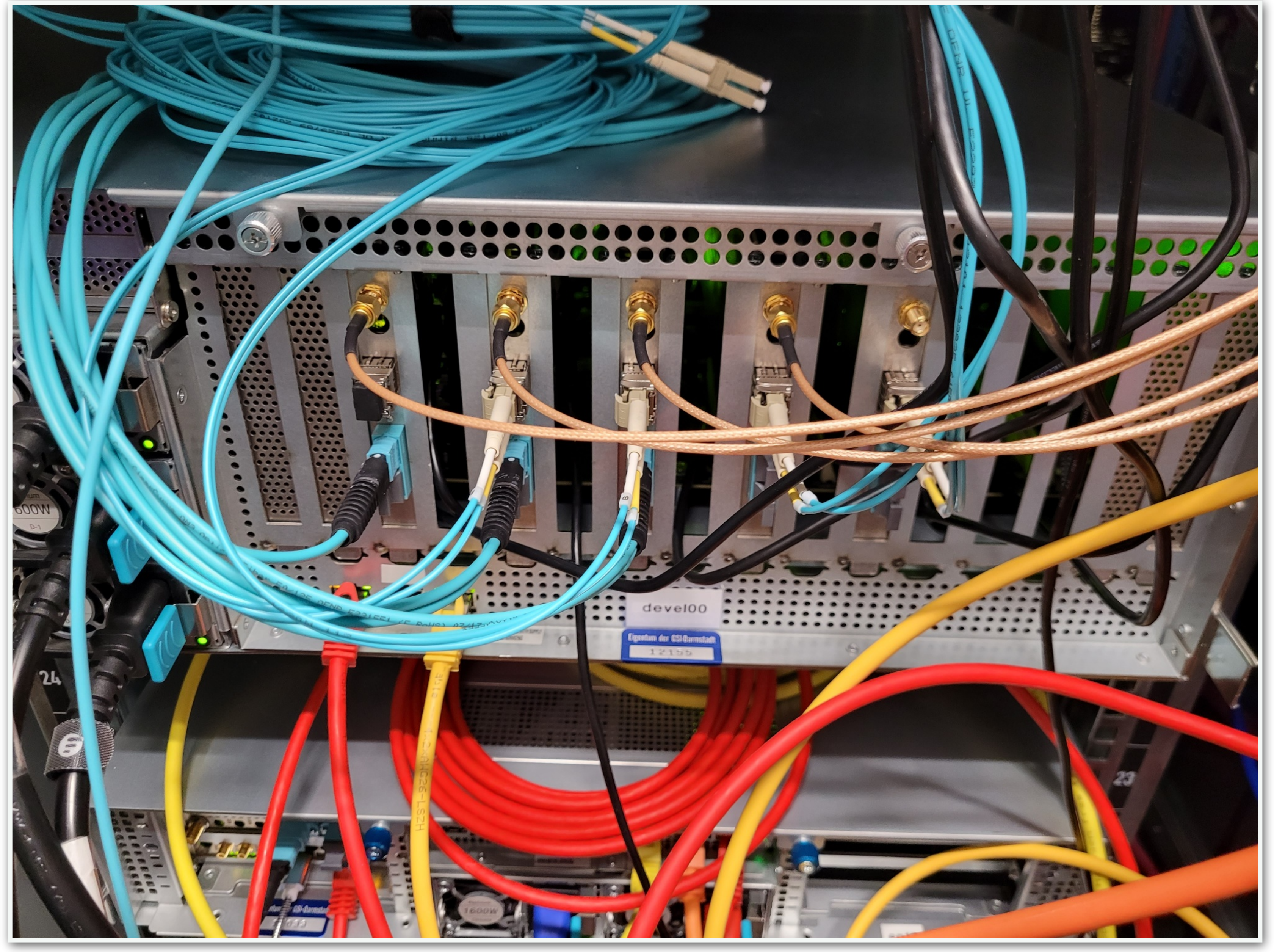
- Activity in all areas
  - Basic design with **PCIe**, **slow-control** and intermediate version of **FLIM** available and in testing
  - Next step: continue testing, **GBTx** for Versal, **TFC** endpoint for FLX-182

- Goal: PRR in Q1/2026 (including decision about FLX-182 as CRI2 for CBM)

Other activities towards SIS100

- TFC2 SIS100 development setup
  - Flexible setup with 5 CRI cards in devel00 (6 needed for SIS100)
  - SIS100 config: 1 Master + 4 Submasters
  - Sandbox config: 1 Master + 2 Submasters + 2 Endpoints
- TFC2-demonstrator interfaced to CRI2 setup
  - TFC connection devel00 <=> devel13
- Development and testing of DCA multi-threading capability
- Upgrade of CBM TOF DAQ setup in PI Heidelberg
- Negotiations with BNL on CRI2 production options





Recent FLES developments

- FLES input interface (Uni F/FIAS)
  - Previously improved FLIM design ported to new CRI hardware
  - **Local data aggregation** on entry nodes implemented
- Timeslice forwarding ongoing (ZIB)
  - Proof-of-Concept implementation of a central manager and dynamic nodes
  - Using Libfabric, IB and TCP providers
- FLES online data management software Flesnet (Uni F/FIAS)
  - **Complete rewrite** of timeslice building software stack

Development of FLES timeslice building

- Flesnet software stack – foundation of CBM readout
  - Start of Flesnet development: 2011
  - Full chain is operational and actively used in CBM test setups
  - Established concepts and data structures
- In productive operation at mCBM since 2018
  - Hundreds of terabytes successfully recorded
- Meets the fundamental requirements for operation at SIS100

Motivation for rewrite:

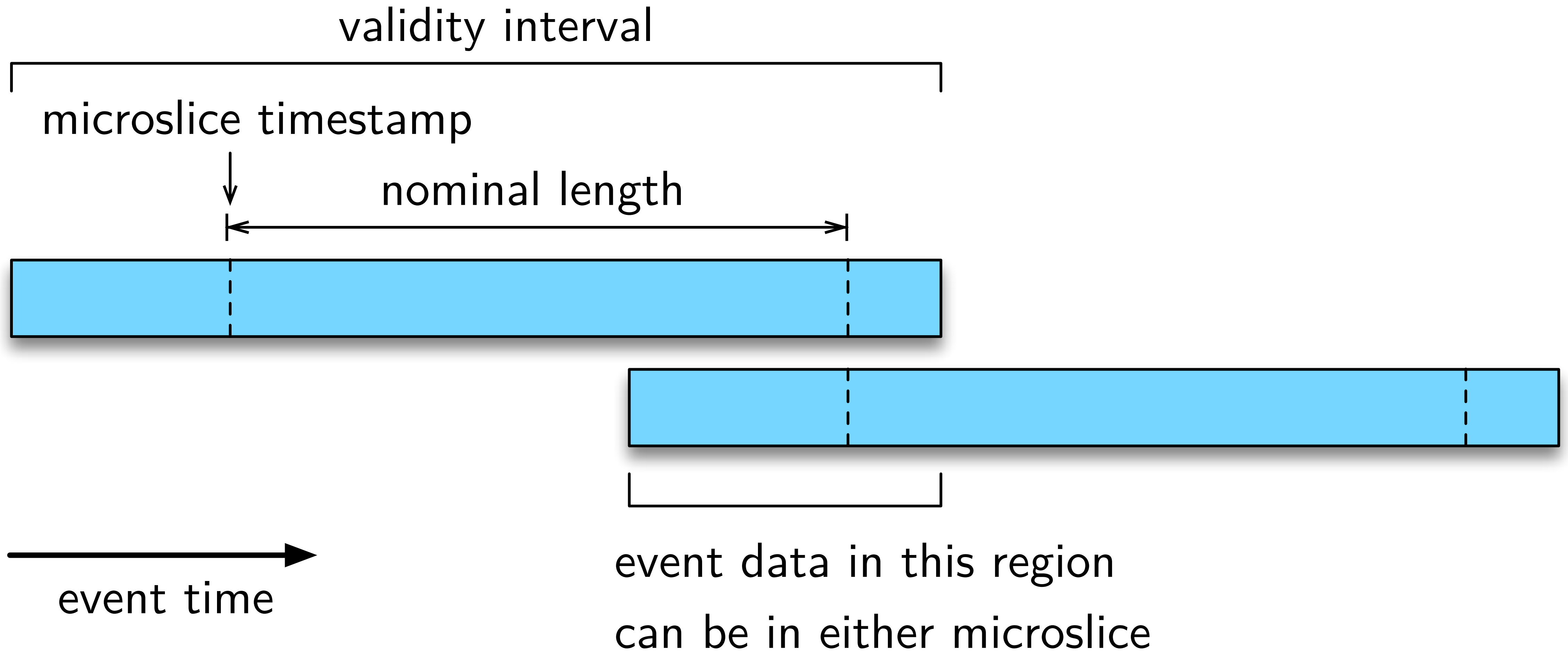
- **Monolithic run concept**, does not leverage the full potential of a free streaming readout system
- **Not very resilient** against external malfunctions (e.g., interface violations by FEE components)
- **Requirements have evolved** since the initial design of Flesnet
- **Additional experience** from years of mCBM and ALICE operation

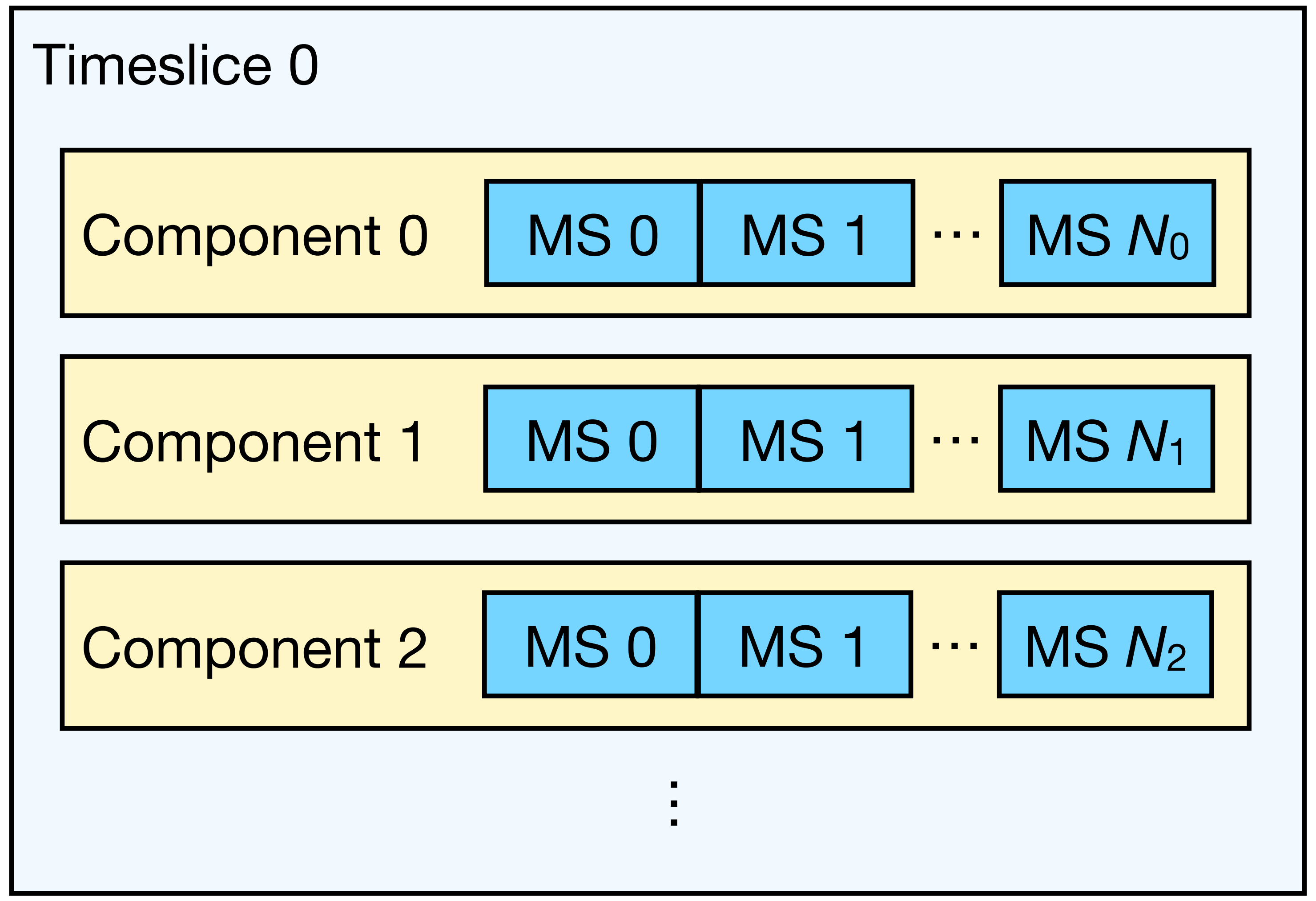
The new FLES data distribution system

- Evolved concepts
  - From data-driven operation to **wall-time-driven operation**
  - From handshake with closed back pressure path to **opportunistic operation with timeouts**
  - From static timeslice assembly to **dynamically calculated components with flexible overlap**
  - From ring buffers to **dynamic buffer management** on receiver
  - From distributed decisions to **central manager**
  - From static to **dynamic data path configuration** (including sources, sinks, and distribution)

- Some advantages
  - Resilience against malformed or missing data from detectors
  - Support for arbitrary and individual microslice sizes
  - Overlap in both directions (before and after), independent of microslice size
  - Resilience against overload (component data sizes, buffers, network)
  - No head-of-line blocking in timeslice buffer when timeslices are locked by consumer
  - Improved memory efficiency in case of different component data sizes
  - Support for centrally aggregated monitoring
  - Support for dynamic scaling of build nodes, resilience against failing nodes

Reminder: microslice and timeslice concept





- The CRI splits detector data streams into short, context-free time intervals and encapsulates them into microslices
  - Each component defines the maximum time deviation of physical event data in a microslice with respect to the microslice reference timestamp
  - Microslice guarantee: All included measurements have an event time in the validity interval
- Timeslice building combines data from all sources to processing intervals called timeslices
  - A timeslice is the collection of all microslices with a validity interval intersecting the timeslice core interval
  - Subsequent timeslices overlap to handle data at boundaries

New feature: local data aggregation

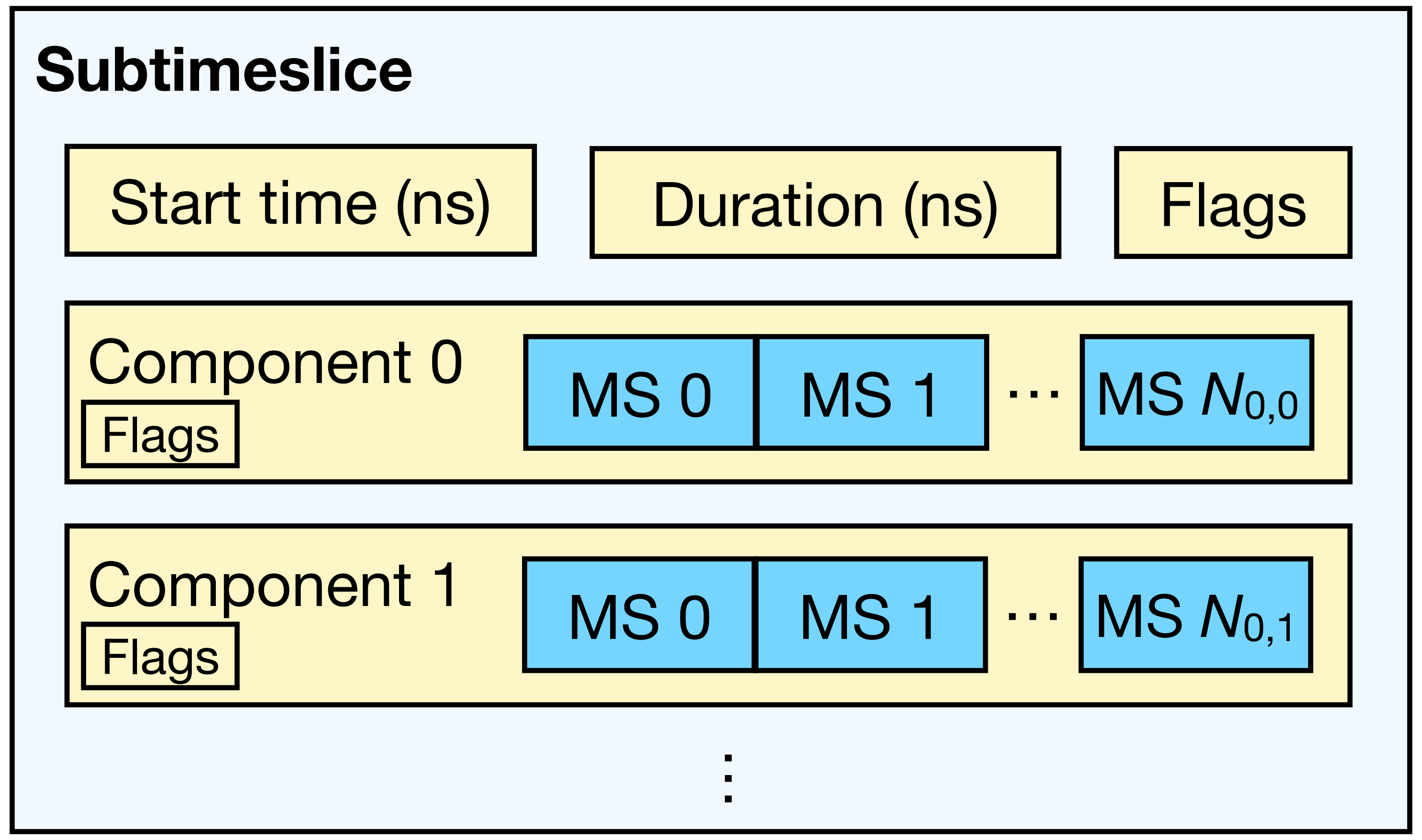
- **Requirements have evolved** since the initial design of Flesnet
  - **Increased density**: fewer nodes, higher data rates, faster network technology

	Initial parameters	Current parameters
Node usage	One set of nodes for both timeslice building and online computing	Dedicated FLES-IN nodes for timeslice building only
FLES-IN nodes	> 250	~ 30–60
FPGA PCI cards per node	~ 1	~ 3–7
Microslice streams per card	~ 1	~ 6
Microslice streams per node	**~ 1**	**~ 20–40**

- Adapt by implementing **local data aggregation** on each node

Local data aggregation

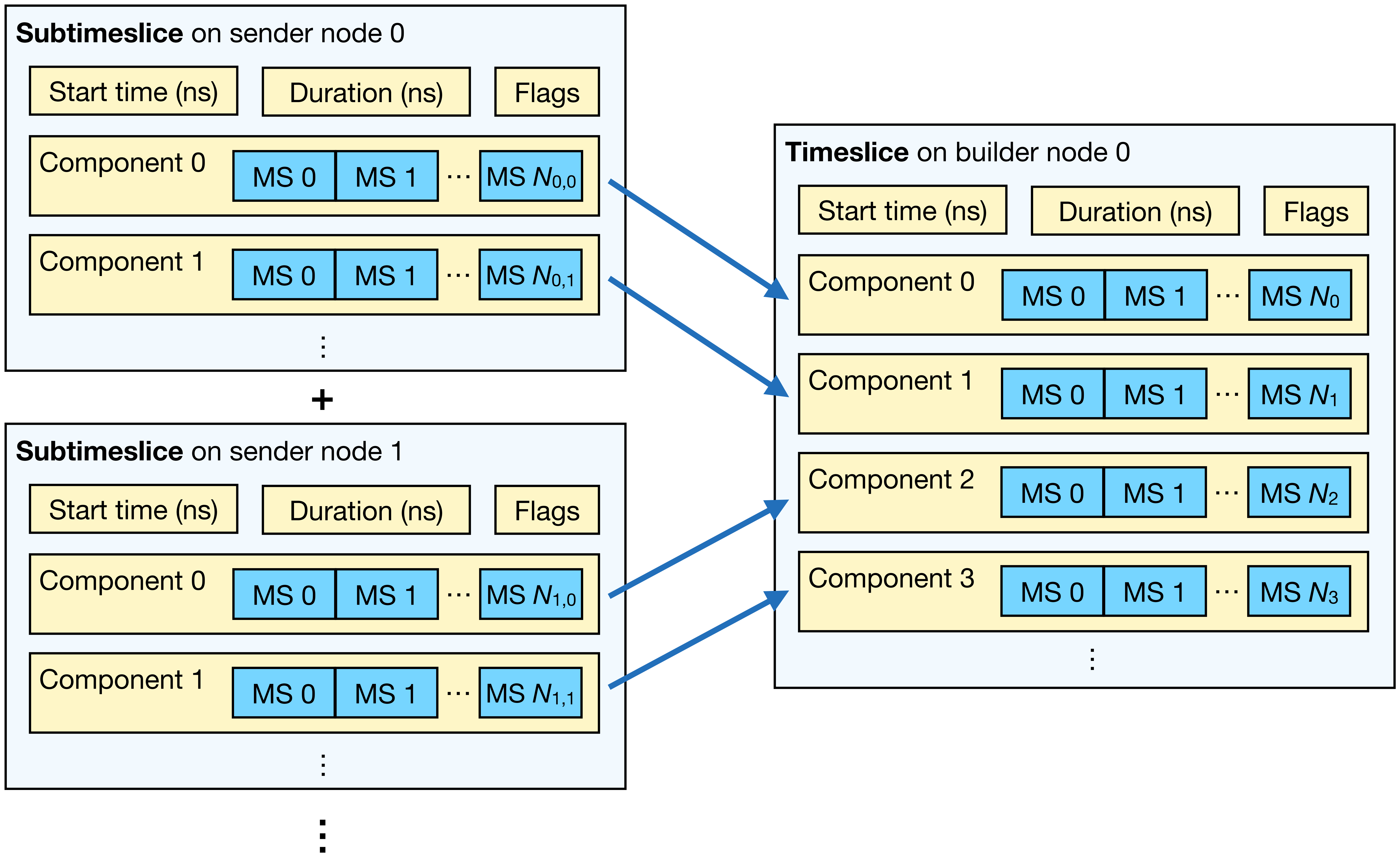
- New data structure: **subtimeslice**
  - Like a timeslice, but with only a subset of components
  - Aggregate per node in two dimensions: in time and across components
  - Aggregate status flags on component and subtimeslice level
  - Introduce (sub)timeslice building related status flags



- Provide **microslices** → Provide **subtimeslices**
  - Enables **improved resilience** against detector malfunction
  - Allows local **timeouts** and changing **link states**
  - Ideal prerequisite for **modularized system startup**

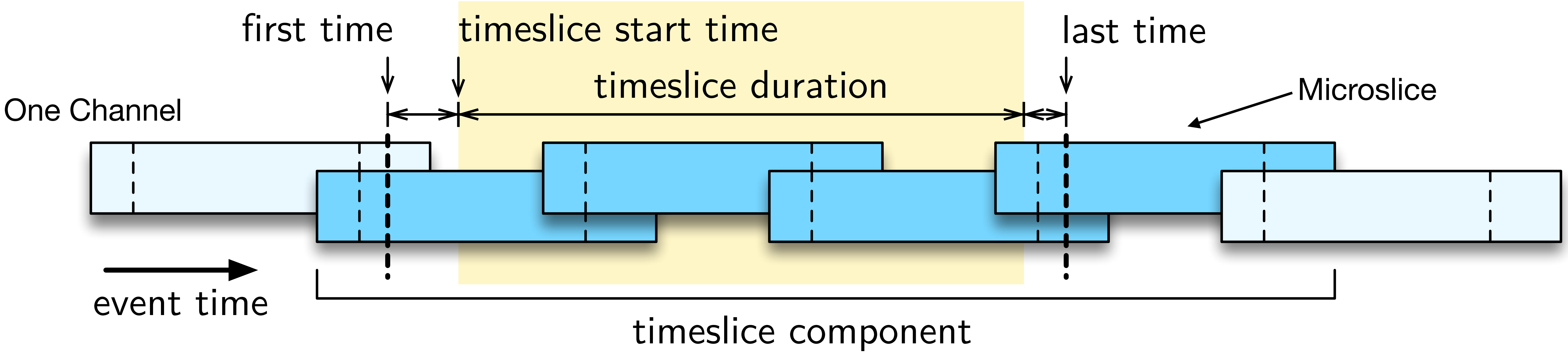
- 1 cri_server per **card** → 1 subtimeslice server per **node**
  - Only need to manage 1 server process
  - Simplify local synchronization across cards

Timeslice building from subtimeslices



- Build timeslices from **timeslice components** → build timeslices from **subtimeslices**
  - Significantly decreased networking **overhead** and buffer fragmentation
  - Less complex **scheduling** of timeslice building over the network
- Timeslice
  - Concatenate components
  - Combine Flags
  - Start time, duration must match

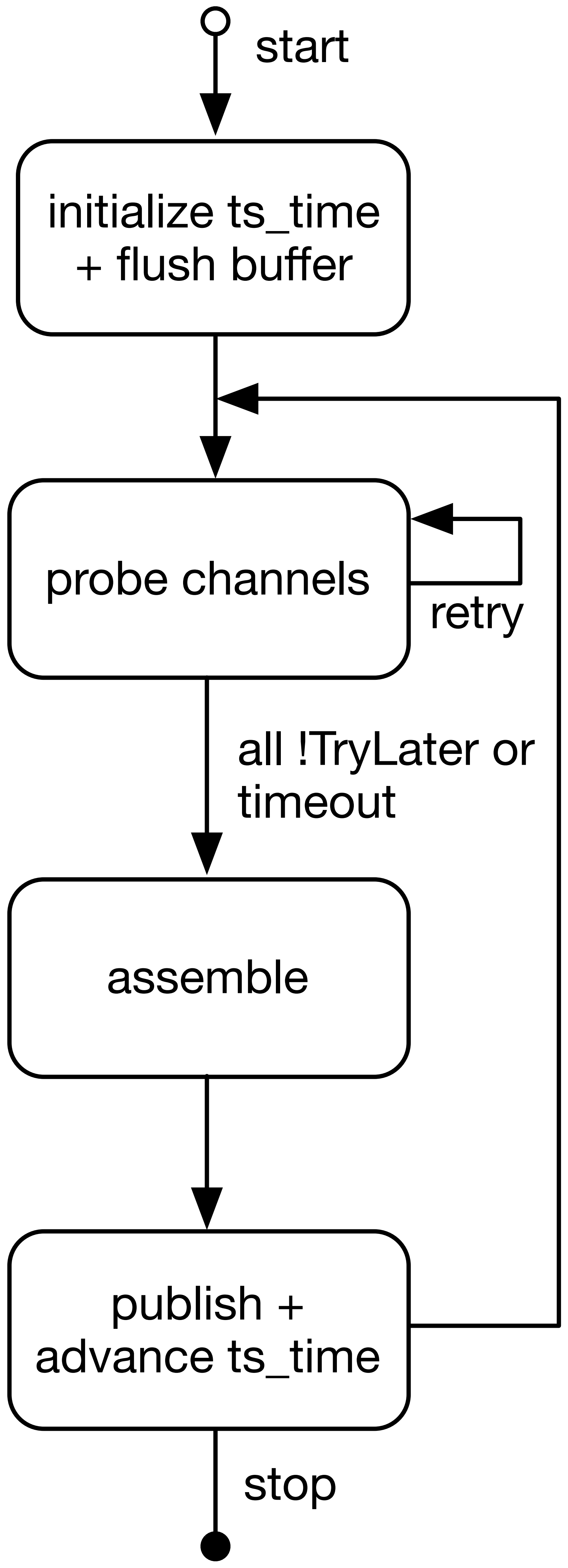
New feature: dynamic timeslice component building



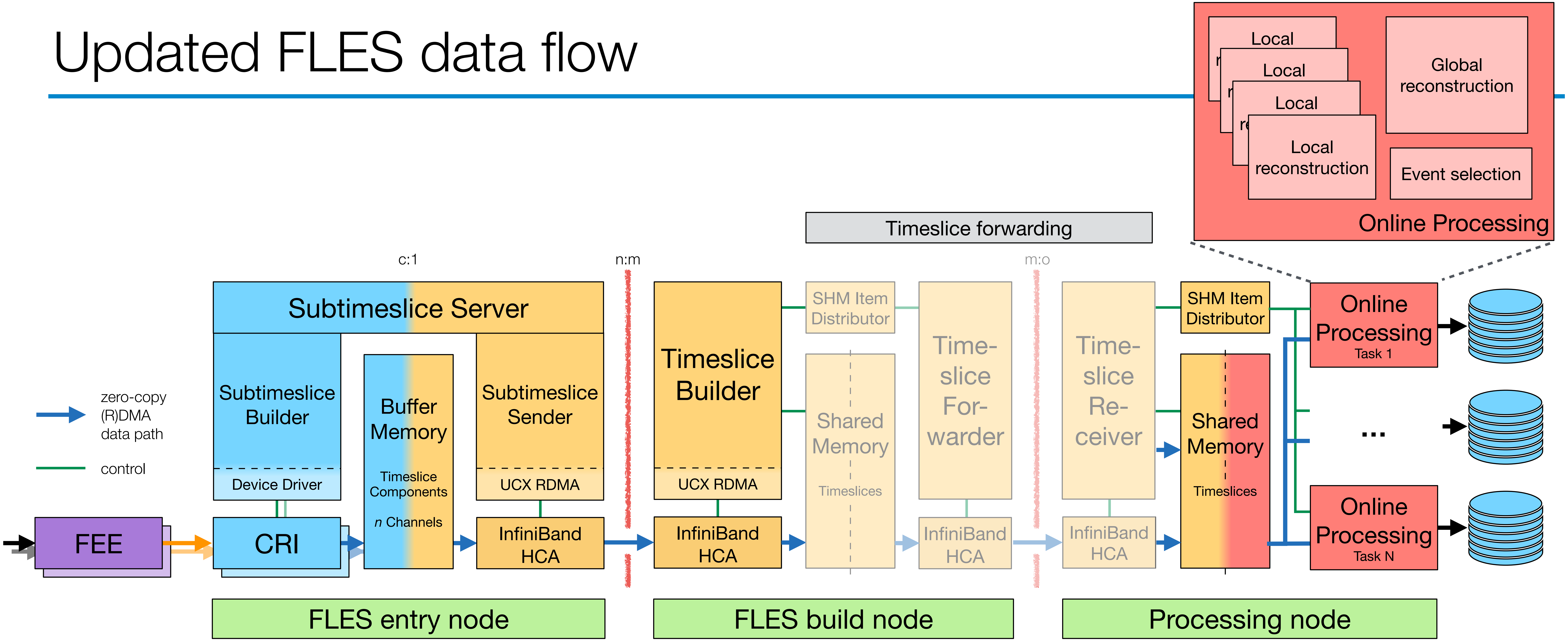
- Calculate the interval [first time, last time) based on the timeslice start time and duration and microslice validity intervals
- **Dynamically** assemble each component based on **binary search** for the first and last microslice
  - First microslice := first microslice in buffer with (time <= first time)
  - Last microslice := last microslice in buffer with (time < last time)
- Advantages
  - **Robust against missing microslices**, only assumes incrementing microslice times
  - Does not require microslice generation to be synchronized over all CBM
  - Automatically (re)syncronize channels at start and after failures

Subtimeslice building algorithm

- Try to collect timeslice components from each channel
  - Opportunistic operation with short timeouts (e.g., 1 ms)
- Periodically probe (non-ready) channels for the current TSC
  - **Fast probing** based on first and last microslice in the buffer:
  - (time of first microslice in buffer) > first time
    → we can never provide that component, **failed**
  - (time of last microslice in buffer) < last_time or (buffer is empty)
    → we cannot provide that component yet, **try later**
- If all channels are ready/failed or the timeout is reached, assemble a subtimeslice by searching the microslices from all ready channel
  - **Ignore non-ready channels**, mark subtimeslices as incomplete if needed
  - Note that the timeout is absolute (wall time > ts end time + timeout), so timeout delays don't stack
- Acknowledge of microslices also based on time
  - Binary search to find the buffer index

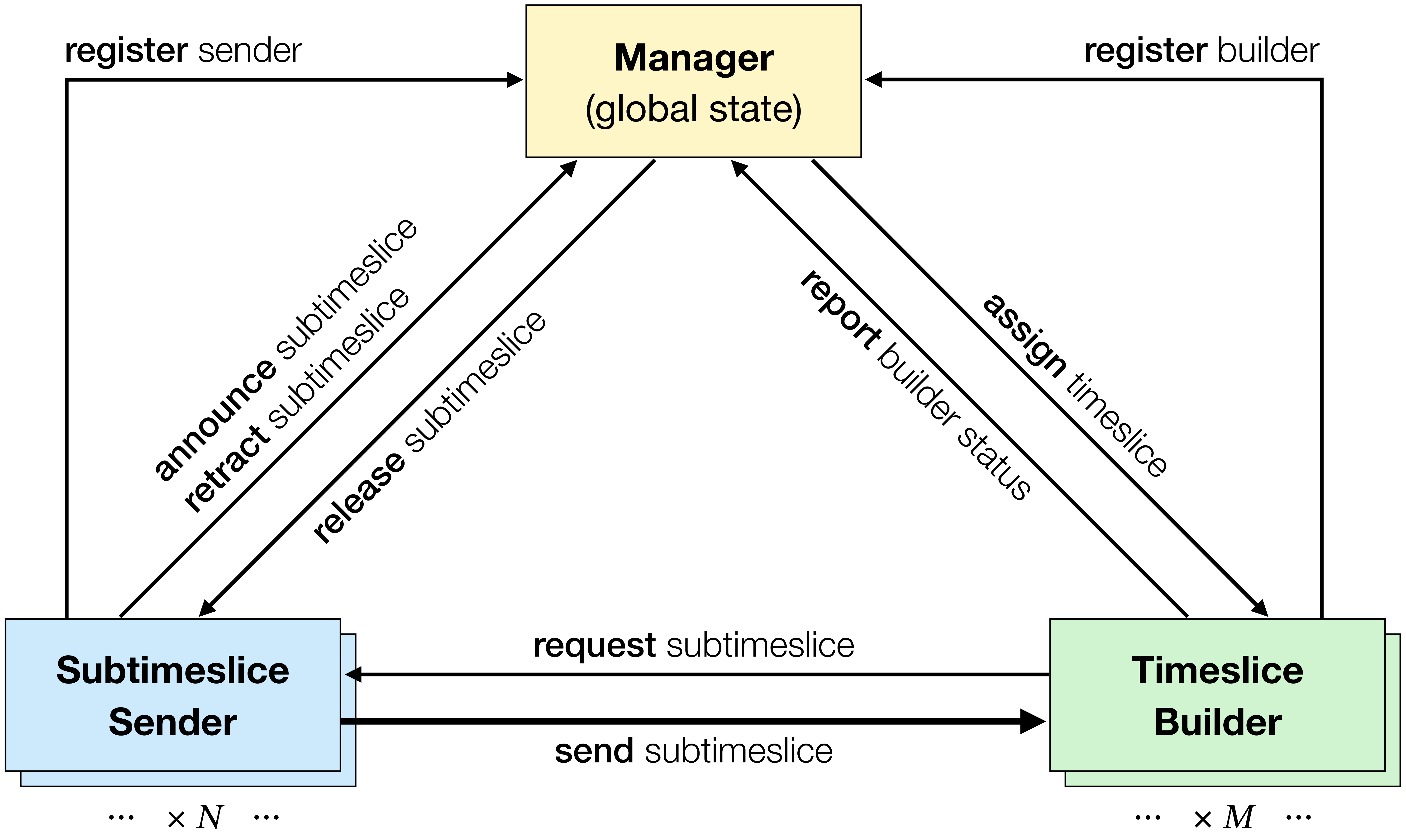


Updated FLES data flow



- **Local aggregation** to subtimeslice on entry node
- **Timeslice building** on build nodes
- (Optional) **forwarding** to processing nodes
- Concept includes a **central server** for load balancing and fault tolerance

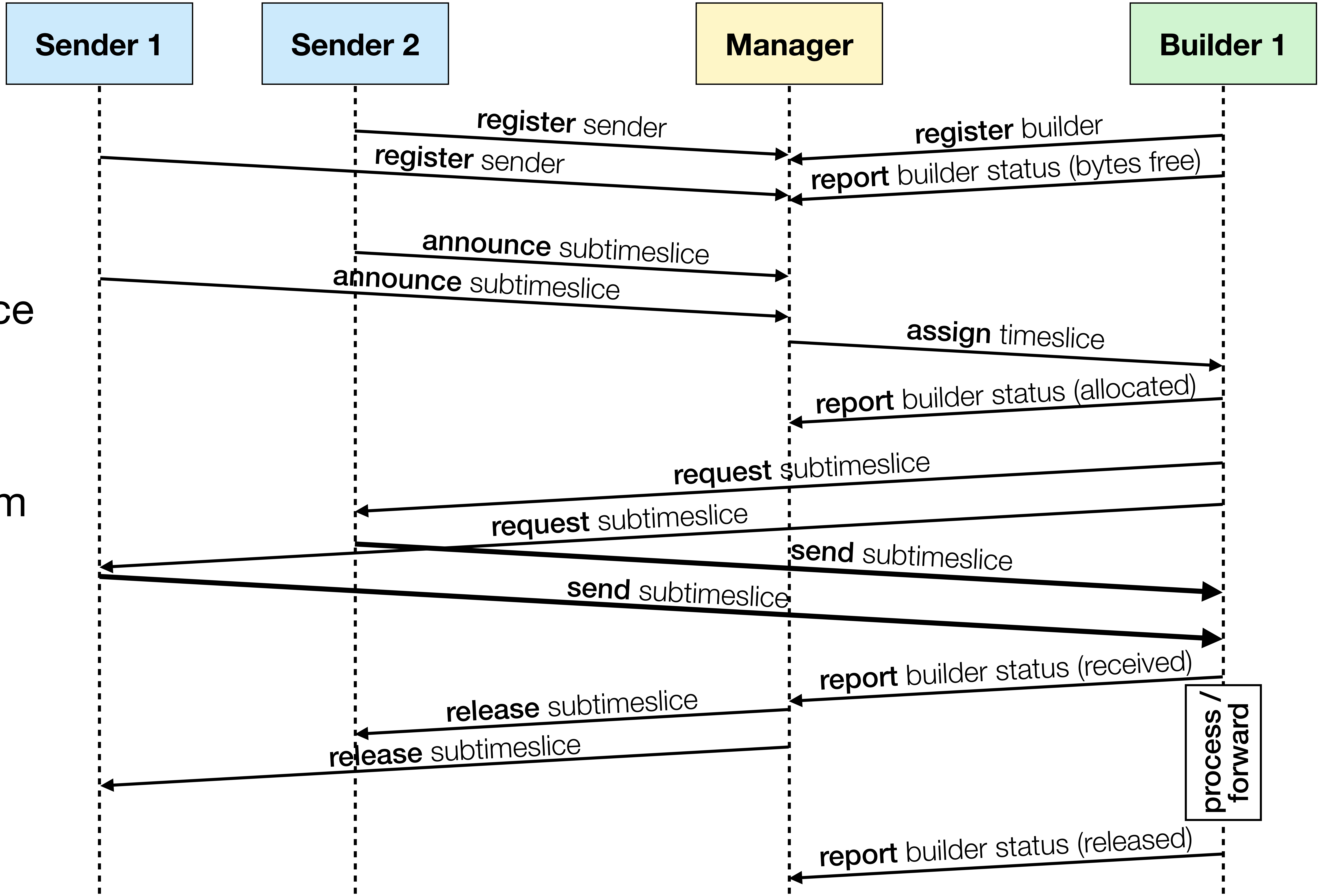
New timeslice building: components



- Central managing service
  - Keeps track of buffer states, assigns timeslices to builders, manages timeouts, releases data
  - Dynamic (opportunistic) data path configuration, live scalability
- Communication between components using UCX
  - "an open-source, production-grade communication framework for data-centric and high-performance applications"
- Flexible networking
  - Use InfiniBand RDMA by default
  - Also replaces ZeroMQ transport for lab setups

New timeslice building: example sequence, ideal case

- After initial registration, **senders** independently **announce** built subtimeslices
- **Manager assigns** timeslice to one of the builders
- **Builder requests** and **receives** data directly from senders
- **Manager** finally **releases** the data
- CBM: ~50 sender/builder nodes



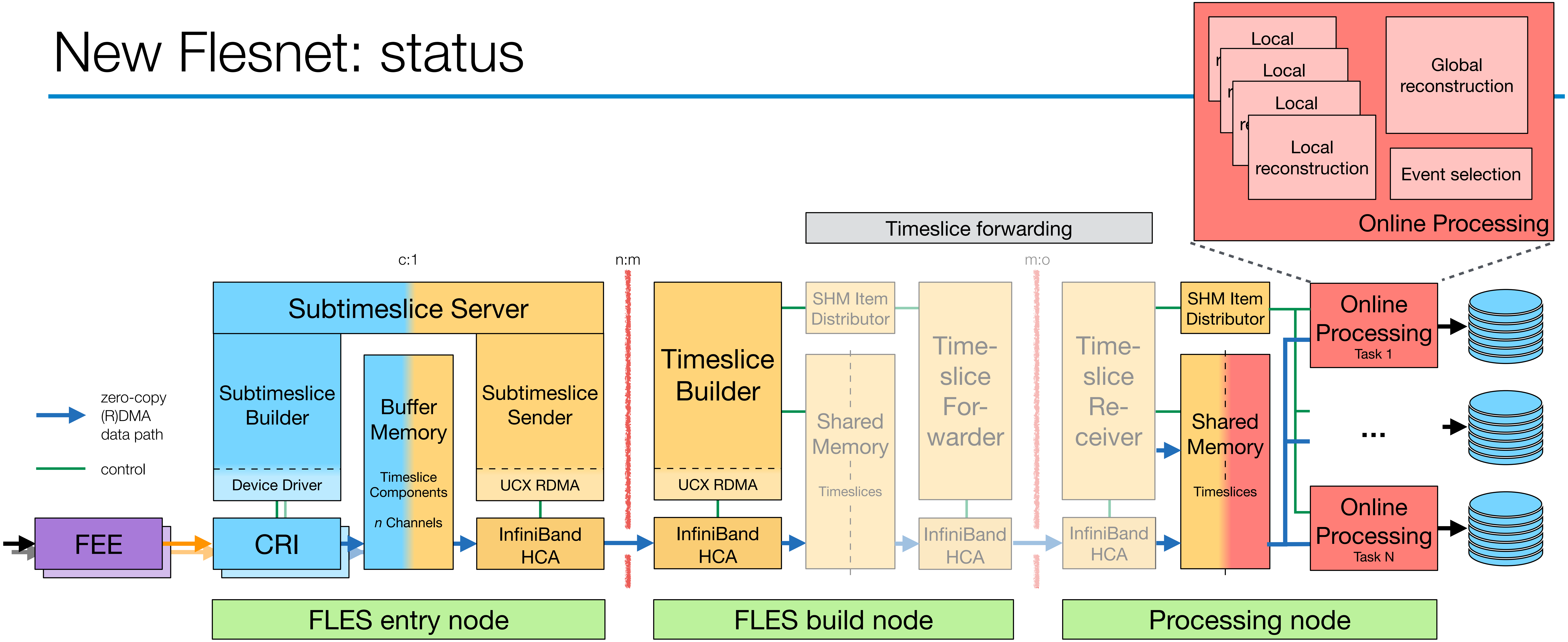
New timeslice building: handing non-ideal cases gracefully

- Goals
  - Primary goal: retain system stability
  - Secondary goal: minimize dead time (= data loss)
- Failure handling
  - Failing sender: after short timeout (example: 20 ms), build with remaining contributions (mark as incomplete)
  - Failing builder: skip when assigning timeslices
  - Failing manager: restart manager during run (reset global state by flushing buffers)
- All components can be stopped and restarted during an active data taking
- Overload handling
  - No back pressure from build buffers to senders (discard immediately if build buffers full)
  - Senders can retract subtimeslices to prevent head-of-line blocking in their ring buffers

New Flesnet: implications for online analysis software

- Flesnet rewrite is (almost) transparent for consumers
  - TimesliceAutoSource class updated
- Only minor changes to established Timeslice API
  - Adding access to new features (timeslice duration, flags)
- As before: need to select data based on time
  - All measurements with **event time in core interval** are included (timeslice guarantee), but **additional measurements** will be there and must be ignored
  - 1. Perform all reconstruction steps on all available data
  - 2. After event building, discard all events with reconstructed physical time outside of core interval
- Need reliable data quality monitoring, reporting to ECS
  - Running data taking is no indication of good data quality / completeness
  - Flesnet can only provide measures based on timeslice metadata

New Flesnet: status



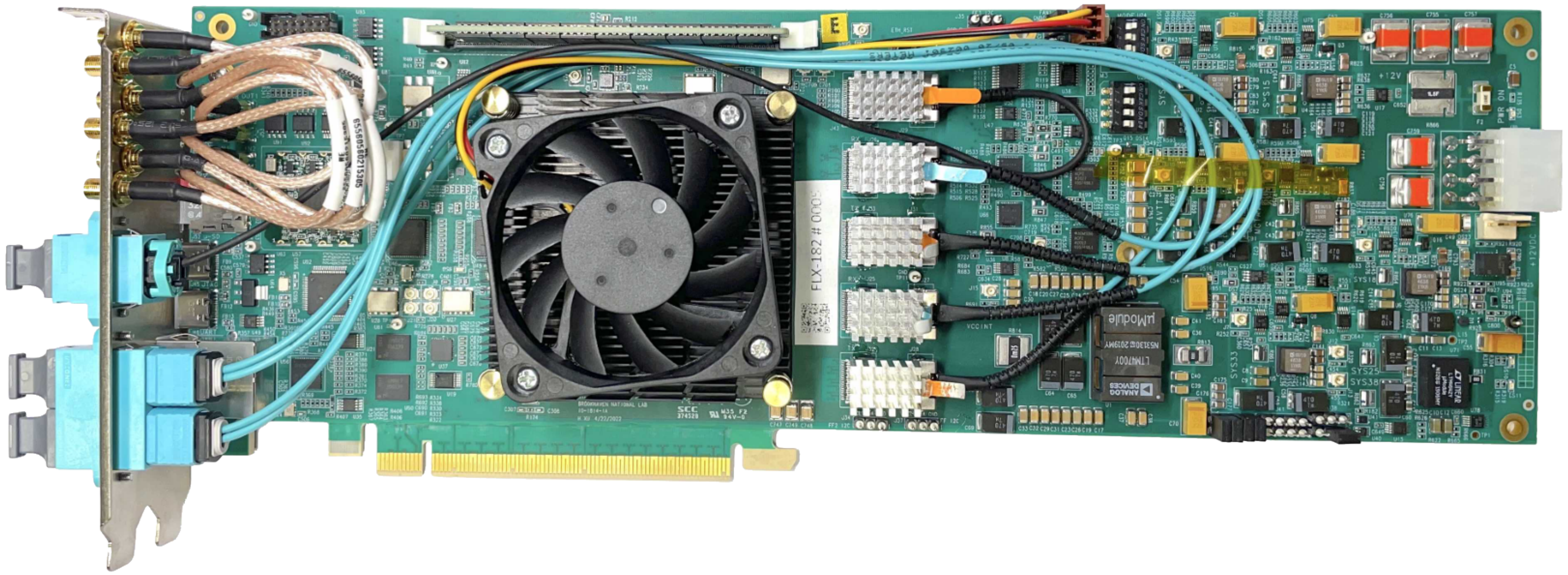
- Initial version of new data chain operational in development setup

- Latest state in Git branch: https://github.com/cbm-fles/flesnet/tree/dev_tscbuilder

Bonus Slides

Not to be shown by default

CRI2 procurement



- Baseline: FLX-182 card as CRI2
  - Testing and verification ongoing
- PRR is foreseen in Q1/2026
- Production options:
  - Option A: Strategic Partnership Project Agreement (SPP) – production at BNL
  - Option B: International Cooperative Research and Development Agreement (iCRADA) – production run by CBM, with support from BNL

Time stamp considerations (1) – single measurement

- Each measurement has:
  - A **time stamp** (known during readout)
  - The time of the associated **physics event** (assigned after reconstruction)
- These **time stamps differ** for several reasons
  - Limited precision in TFC time distribution
  - Limited precision in per-subsystem time distribution to FEE
  - Limited intrinsic detector time resolution
  - Physics and detector effects (e.g., particle time of flight, drift velocity)
- Handled by **microslice** concept
  - Specify time **uncertainty interval** of measurements (per FLES input)
  - Generate timeslices with sufficient **overlap**

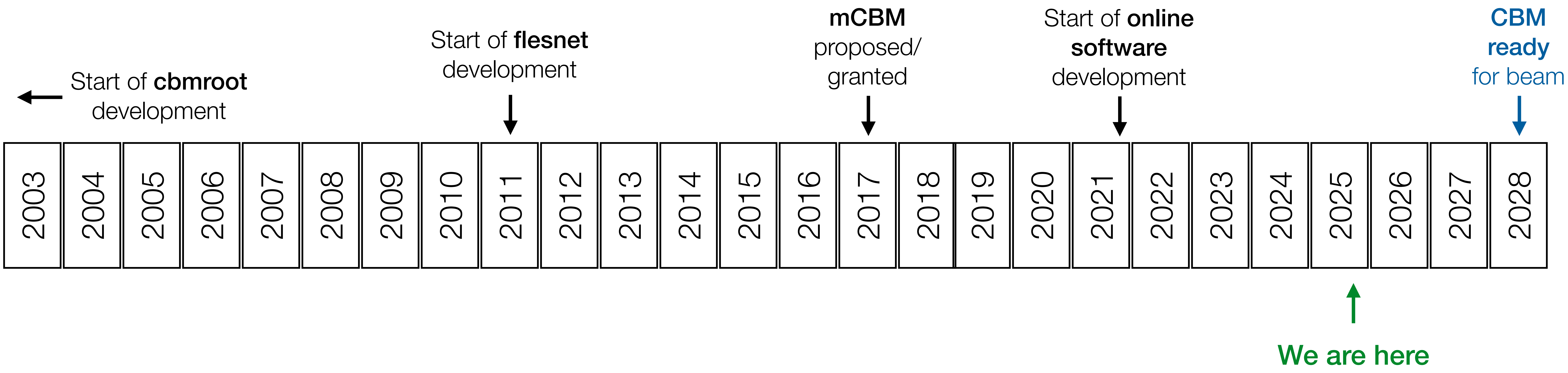
Time stamp considerations (2) – stream of measurements

- Frontend message streams are **not** automatically merged **in chronological order** by detector readout electronics
  - Streams cannot easily be cut into intervals
- However, the maximum time deviation is usually known (e.g. unsorted only within one epoch, FEE drain time)
- We can handle this **limited time deviation** via same overlap concept
- Reduces requirements on microslice building in hardware
- CRI design implementation example:
  - Start new microslice when first measurement in corresponding time interval is encountered
  - Put any subsequent measurements into the new microslice, even if timestamp is lower again
  - Generally specify larger interval of possible corresponding event time for all microslices

Example values for parameters

Parameter	Possible value
total data rate	500 GB/s
timeslice duration	20 ms
typ. timeslice size	10 GB
typ. microslice duration	200 µs
typ. microslice size	100 kB
# components (channels)	1000
# entry / build nodes	50
entry node buffer memory per channel	2 GB
subtimeslice build timeout	1 ms
subtimeslice announce timeout	20 ms

Timeline



Start of **cbmroot** development

Start of **flesnet** development

mCBM proposed/ granted

Start of **online software** development

CBM ready for beam

2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028

We are here