DUNE FD DAQ
Inter-process Communication

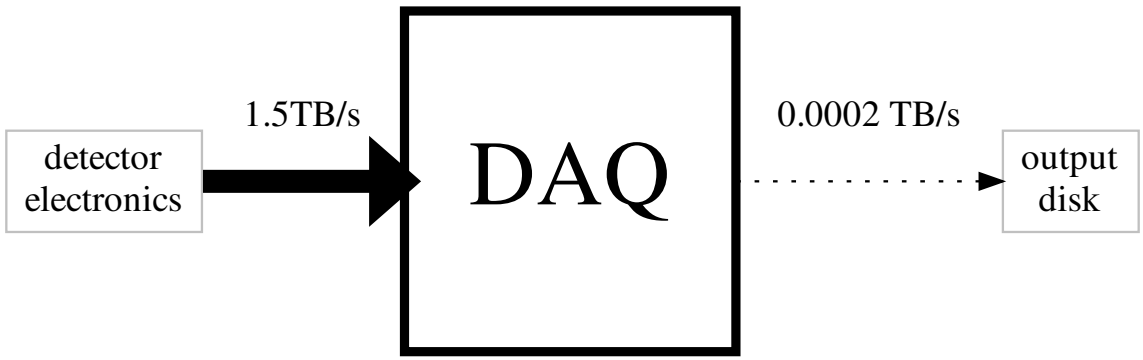
Brett Viren

Physics Department

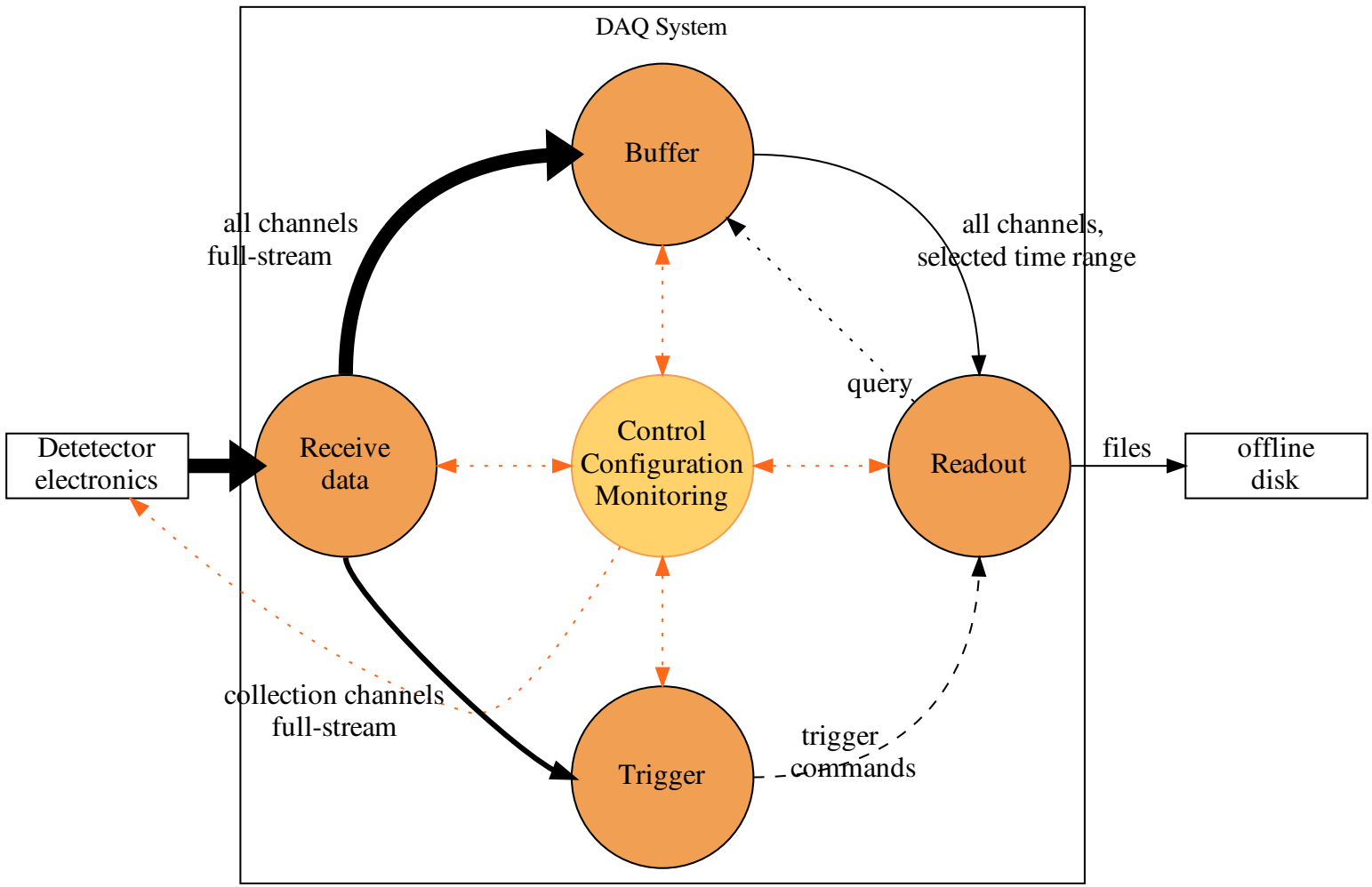
BROOKHAVEN
NATIONAL LABORATORY

DUNE DAQ CDR – Dec 2018

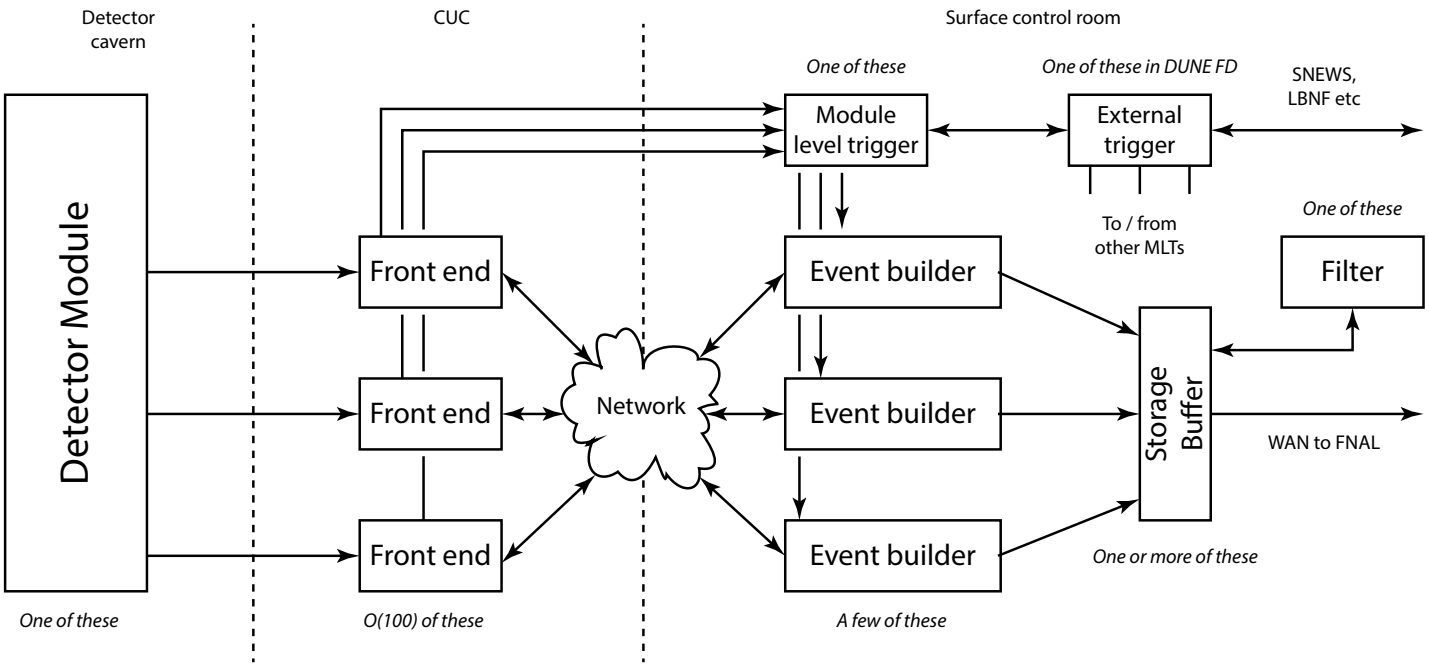
Zoom 1: Context



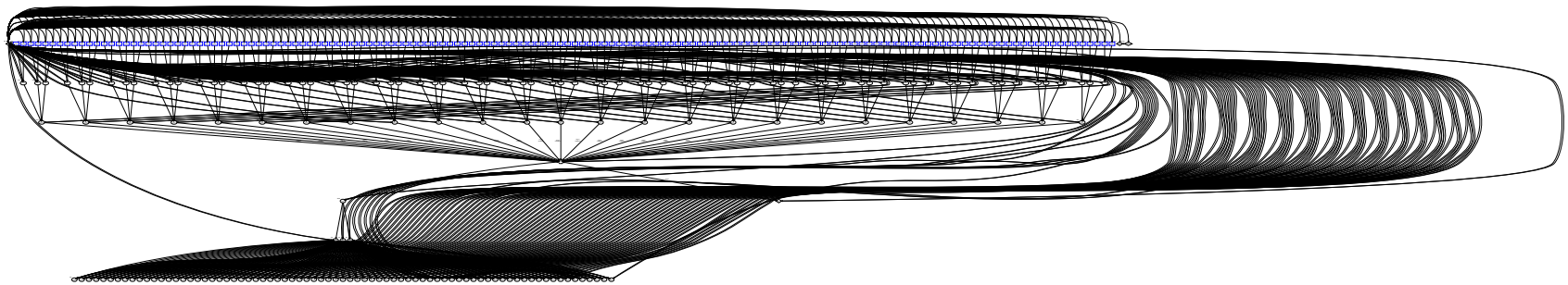
Zoom 2: Conceptual units



Zoom 3: Functional blocks



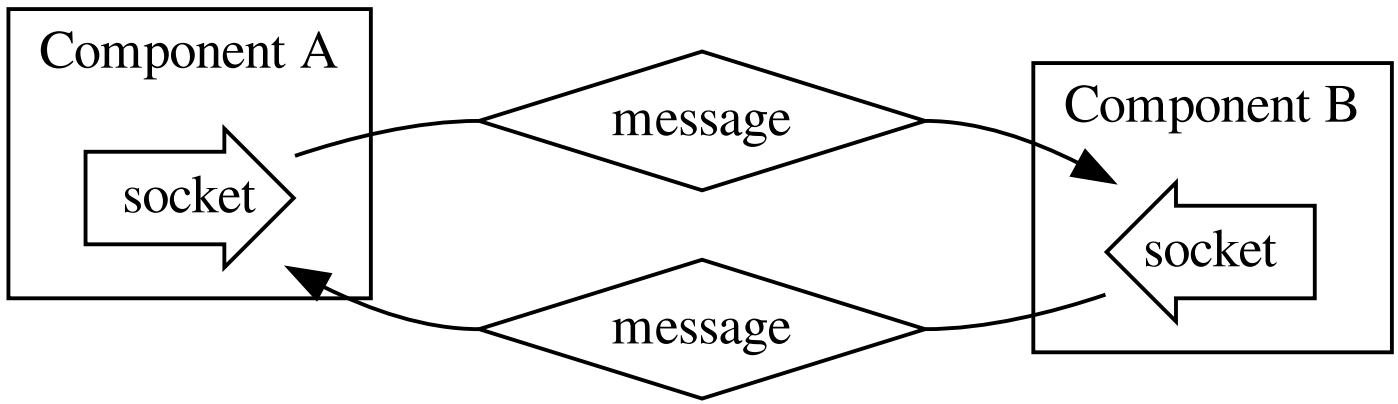
Zoom 4: Individual components



View of individual components and a **subset** of their communication connections.

Multiplicities: 1 SP module, 150 APAs, 75 front-end computers, 25 trigger processors, 1 module level trigger, external trigger, run control, configuration, monitor, dataflow orchestrator, a variable number of event builders, 75 front-end buffer interface services.

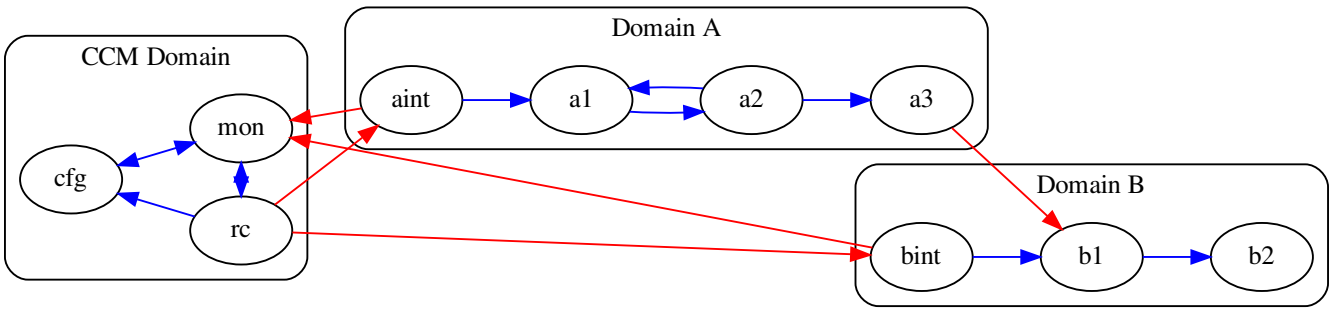
IPC: the arrows in the graph



Inter-process Communication (IPC) is the language spoken between DAQ components:

- Its words: message types, schema, formats.
- Its media: transports and distribution patterns.
- Its sentences and grammar: application-level protocols.

IPC Domains



Distinction:

Inter-domain A common "lingua franca" communication system between processes from different subsystems.

- Eg, Run Control sending commands to configure Event Builders or components sending log messages to Monitoring.

Intra-domain The IPC used for communications which is internal to a subsystem.

- Eg, Dataflow Orchestrator forwarding a trigger command to an Event Builder using artDAQ messaging.

Inter-domain IPC Requirements

DAQ will utilize a single **inter-domain** IPC (the "*lingua franca*" IPC) which shall

- Support **redundancy and fail-over** to avoid single points of failure.
- Support for component **discovery** and **presence** with a reaction time that is of order one second.
- Impose on data flows **negligible overheads** for latency, bandwidth and computation.
- Support DAQ features of **zero-downtime** and **self-healing**, among other.
- Provide required variety of **communication patterns** (eg pub/sub, client/server, pipelines, etc).
- **Scale** across thread, process and network transports.
- Provide language bindings for at least **C++ and Python**.

Intra-domain IPC Requirements

Intra-domain IPC options:

- Some domains **may** use *lingua franca* IPC directly, or
- A domain **may** use a unique IPC
  - Do not exclude good domain solutions based solely on differing IPC!
  - → Shall develop "*lingua franca*" **translation interfaces**, as needed.

Each **intra-domain** IPC has own requirements and challenges. Some examples:

Trigger	high input data rate (1.5 TB/s), high internal message rate (up to 300 kHz primitives per APA), synchronization requirements.
Run Control	scheduling reconfiguration message for for zero-downtime run changes, assuring all components are available and operational.
Monitoring	accepting log messages at different priorities from essentially all domains and components.

Interfaces

IPC is essentially 100% about interfacing.

- Numerous application-level protocols need to be understood.
- Shall formally define them and their message types, schema, format.

Some specific interfaces:

- Two-way exchange between DAQ and CISC (eg, HV status, DAQ PC health).
- Control/configure/monitor domain must interface with all other DAQ domains.
- Domain-to-*lingua franca* translation interfaces.

IPC System Design

- All messages carry a **data time** measuring, by a common clock, the sample time of associated detector data.
- Messages are versioned and typed and follow a **formal schema** and support meta–, structured– and packed–data.
- Components consist of one or more "**agents**".
- Each agent runs a thread loop, communicating with main thread via thread-safe queue. It may bind/connect its own operational sockets and it runs "payload" code providing some unique functionality.
- Components identify their agent socket addresses and associated **capabilities** via the IPC **discovery** mechanism.
- Larger DAQ structure is constructed by peers locating agent sockets by applying **rules** against discovered capabilities.
- Aggregation of agents into their components is driven by initial configuration. Large scale DAQ graph construction is thus **emergent**.
- Subsequent, **dynamic reconfiguration** may also occur.

Key Challenges for DAQ IPC

Variety of communication patterns and requirements.

- Triggering domain requires a high throughput (1.5 TB/s) pipelined layer with configured peers culminating in low-rate publishing to multiple, possibly unknown subscribers.
- Some components require auth/auth, others explicitly should not.
- All components must participate with control/configure/monitor domain.
- Some domains require non-trivial application-level protocols, others are simple.
- Dynamic discovery, automated aggregation, self-healing, zero-downtime, redundancy and fail-over require a varied set of IPC features.

Must factor out and reuse common communication patterns and their software implementations.

Validation

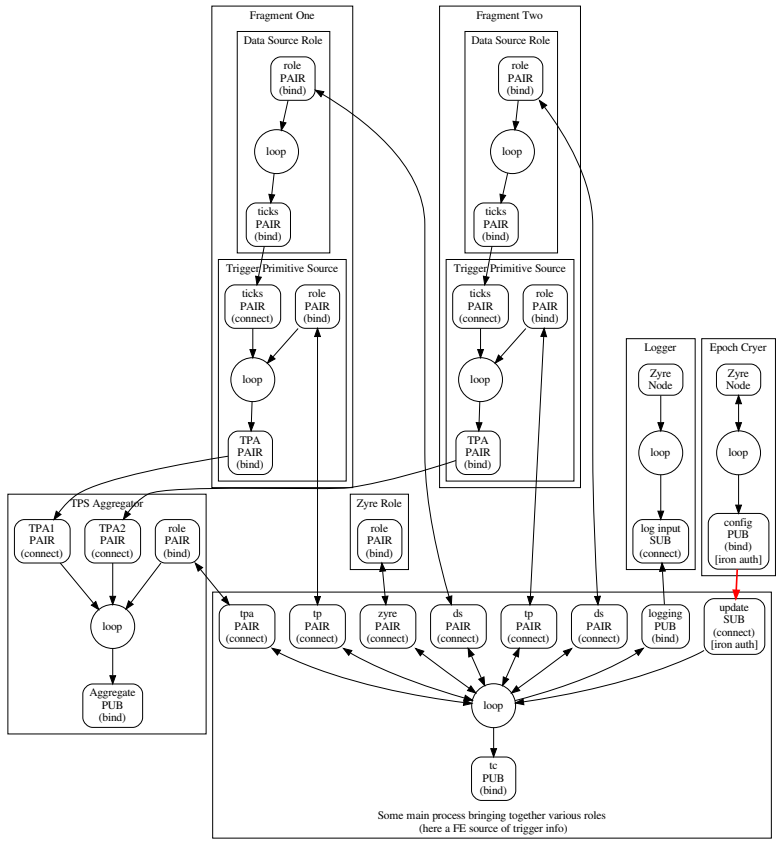
- NetIO provides pub/sub pattern and is in use in protoDUNE-SP/FELIX.
- artDAQ provides its own IPC and is well validated on many experiments.
- Actively evaluating **ZeroMQ** as basis for "*lingua franca*" IPC
  - Free Open Source Software, large active and supportive community, positively evaluated by CERN.
  - Supports pub/sub and many other patterns as well as discovery and presence, has C, C++, Python and many other language bindings.
  - Used as basis for and early prototype (link).

ZeroMQ-based IPC prototyping

Focus on trigger domain and interaction with RC, dummy payloads.

- Tests agent aggregation, discovery, logging, auth and configuration.
- Exercises thread/process/network transport transparency.
- Emphasis on abstracting commonalities, each payload is essentially a simple function.

Buzzword compliance: ZeroMQ/CZMQ/Zyre, Google Protocol Buffers, C++-17, SMS state machines, Elliptic Curve crypto, Waf build system, Jsonnet configuration, Jinja2 code generation, dynamic plugin system.



Concepts and technology are sound. Identified common code patterns automation techniques to manage complexity going forward.

Plans

- Continue prototyping ZeroMQ based solutions and develop initial message schema and protocols.
- Along with domain experts, develop and apply IPC technology selection criteria.
- Review early prototype protocols and iterate.
- Implement shared, base IPC application toolkit library.
- Integrate with existing domain implementations and assist with development of needed novel components.