Open questions about NetworkManager and its usage in dunedaq

...

Marco Roda

Intro

- The directions I received for the development of the NetworkManager are fragmented and not comprehensive
  - Apologise in advance if some questions sound silly, but I prefer this rather than guessing each other thoughts
- These slides are a collection of areas that I don't fully understand
  - Hopefully they will be useful to clarify the desired direction and help develop the necessary understanding
- I'll try to focus on concrete examples
  - Some bigger picture questions are still open

“Configuration style guide”

“Configuration style guides”

- This section contains questions about the desired usage of the new interface and the relation with the old queue interface
  - I understand the answer can change depending if we look at the short term or long term
    - Please let’s try to specify this when it’s necessary

Non equivalent interfaces

- NetworkManager and Queues have non equivalent interfaces
  - NetworkManager is geared around a callback mechanism
  - Just replacing queues in favour of NetworkManager will not work without changing completely the logic of some modules
    - Yet possibly almost everything could be re-coded
    - I would like to understand better the final goal here

- When to use NetworkManager and when to use a standard queue?
  - Are queues supposed to be deprecated at some point?

Queue roles in multithreading

- SPSC and MPMC queues have a standard role in multithreaded design
  - They are the typical solution when data with different frequencies have to be handled by a single thread

- There will be cases where the NetworkManager will need to be coupled with a queue
  - Do we like to keep using the old appfwk queues?
  - Shall we use a more direct interface - like using follyqueues directly?
  - Are these queues to be used inside a module or still structure the modules so that queues are still used inter-modules?

Mixing between network usage and other configuration

- Can the following be a good summary?
  - for messaging and data exchange between applications NetworkManager will be preferred
  - internally an application should keep going with queues

Performances and dependency on conf commands

- Right now there is the implicit assumption that the same operation performed within the same run will have the same effect
  - If I send a data requests to a certain GeoID, they will reach the same destination
  - Will this be true in the NetworkManager era?

- How performant do we expect the Singleton to be?
  - Can it handle lot of requests from the same application?

Usage examples

Evolution of the TRB's DataRequest sending map

- Before the TRB had a map from a GeoID to the dedicated queue
  - Retrieved at configuration
  - how does this model scale?
    - What is the string we will pass to the NetworkManager?

- Options
  - String with an address to be received via configuration
    - configuration specifies that address for each GEOID
    - What sets the configuration?

  - String with an unique ID to be received via configuration
    - configuration specifies that unique ID for each GEOID
    - NetworkManager will know the uniqueID
    - This does not really solve the complexity of the configuration, I think

  - Something like GeoID.ToString()
    - No configuration on the TRB needed anymore!
    - How does the Network Manager will know where to send?

TRB receiving model