IHMC Pub/Sub

RTPS/ROS2 communication protocol in realtime Java

Apache 2.0 Licensed

https://github.com/ihmcrobotics/ihmc-pub-sub
What does IHMC Pub/Sub do

- Lightweight wrapper around Fast-RTPS
  - No memory allocations for sending/receiving objects
  - Object orientated based on Fast-RTPS pub/sub interfaces
  - Could provide other back-ends (JRTPS?)
- Native Java IDL compiler
  - Every element is pre-allocated
  - No backing C/C++ code
  - Export data to YAML for long term storage
- RTPS Visualizer
  - Alpha quality
Sample kind: ALIVE
Data length: 36
Ownership Strength: -1
Source timestamp: 1505924768.62
Sample identity:
  Sequence number: 78
  GUID: GUID Prefix: 0100001ac1700000000000000 Entity ID: 00000103
  Related sample identity:
    Sequence number: -1.0
    GUID: GUID Prefix: 000000000000000000000000 Entity ID: 00000000
Data:
  Encapsulation: CDR_LE
  00000000 05 00 00 00 4A 61 76 61 00 00 00 0F 00 00 00 ....java ....
  00000010 48 65 6C 6C 6F 77 20 6E 64 20 27 00 00 00 Hello World 77....
  00000020 00 00 00
Connected | QoS policy valid
Why custom Java implementation

- Real-time safe
  - No allocations necessary to send/receive
  - Native Java objects
  - No need to compile messages to C++ and then generate a Java wrapper
- Allow abstract implementations of .idl files
  - Re-use existing data types
  - Vector, Point etc
  - Backwards compatible with our codebase
First results

- Variable logging using IHMC Pub/Sub
  - 10000 64 bit variables (LZ4 compression)
  - 1 Khz
  - RTPS Best Effort
- Do not use Fast-RTPS write directly from Realtime thread
  - Blocking call
  - Use separate thread with lockless data sync.
- Share large models over history
  - Be careful to tune heartbeat period for large topics
  - Use newest Fast-RTPS
Further work

- Support for ROS 2 messages
- Compilation step between .msg ↔ .idl necessary?
- Proper topic names and partitions for ROS 2
- Stabilize IHMC RTPS Visualizer