cosylab
CONTROL SYSTEM LABORATORY

FECOS

Matej Šekoranja *matej.sekoranja@cosylab.com*

Miha Vitorovič *miha.vitorovic@cosylab.com*

Rok Štefanič *rok.stefanic@cosylab.com*

What is FECOS?

- **F**ront**E**nd **Co**ntrol **S**ystem

- A LabVIEW object oriented framework running on all FECs
  - Uniform and transparent network communication
  - Standard services like error reporting and logging
  - Access to timing system
  - *Etc...*

- Each FEC will be able to run multiple applications – FECOS components

Progress overview

- STM based publish/subscribe messaging
- Design for real-time actions

Real-time requirements

- Single and multipoint ADC
  - Performing data acquisition with precise time-stamp
- Close loop motion control
  - Precise timing and triggering required
- Local timing generation

- Watchdog supervision of real-time tasks

Real-time design

- Instructions and best practices for all required use cases
  - Most use cases can and should be solved in hardware
- Real-time tasks only available in *Op* state
  - Runs in parallel to normal component *Op* method
- Integration of real-time tasks with watchdog
  - Monitoring the health of real-time task
  - Handle watchdog requests in the least disruptive way
  - Long actions need to be able to abort on request from framework

Real-time tasks in FECOS

- Real-time tasks only available in *Op* state
- Runs in parallel to normal component *Op* method
- Fast-reacting actions
- On-demand real-time actions

Fast-reacting actions

- Started when component enters the Op state by framework
- Waiting for external trigger – hardware event
- Can start the action instantly on trigger

On-demand real-time actions

- Started by FECOS trigger – user command or component event
- Start-up time delay not important
- Performs real-time actions when running
- Exists when finished

Watchdog

- Each component monitors its own real-time action
  - Reports to device watchdog
- Supervision of the actions through real-time FIFO
- Component running multiple real-time methods in parallel not supported
  - Multiple loops inside one method not a problem
- Real-time action that runs indefinitely may be required to exit

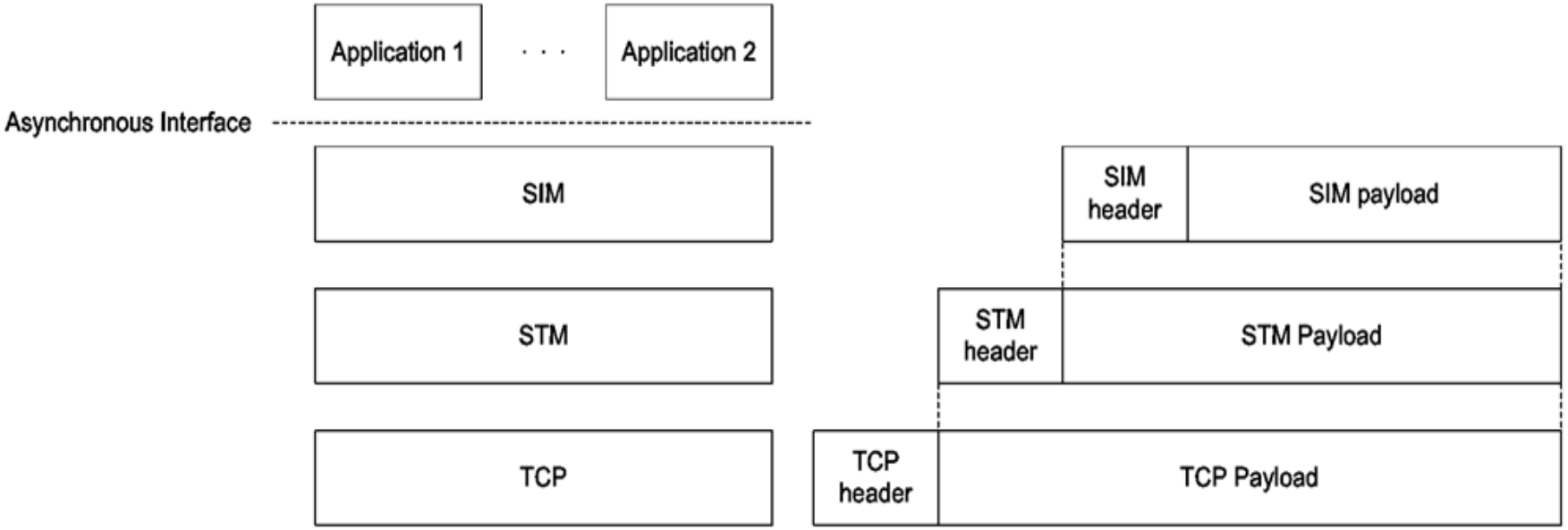
STM based protocol

- What is STM?
  - NI LabView Simple TCP Messaging
    - Very thin layer above TCP
- Why STM?
  - Support for LV-RT (library provided by NI)
    - We don't want to have "LV-RT DIM problems" again

Data Size (32 bits)	Meta Data ID (16 bits)	Data

SIM

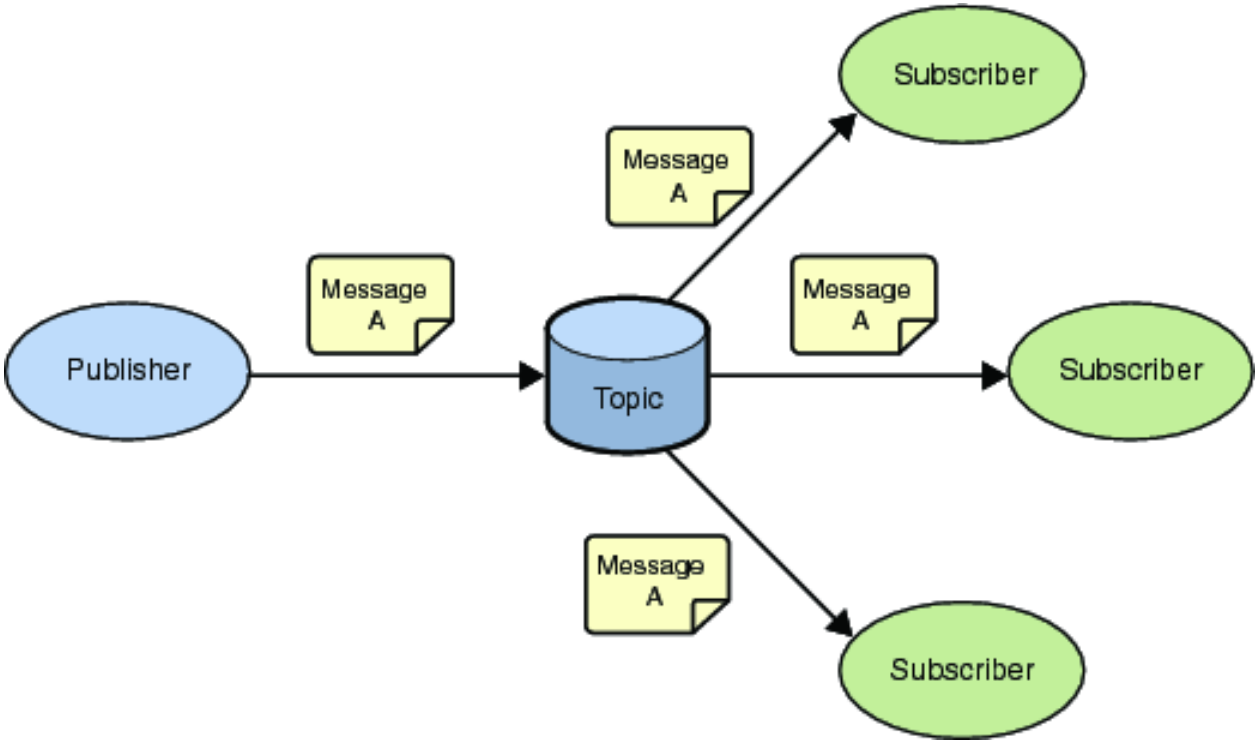
- STM is too “raw”
- SIM (SImple Messaging) is layer above STM



- Defines general header that removes need of layers above to define another header – focus only on payload
- Will be used by MTS, VAA… to be used as “the” MA communication protocol

MAPS

- Replacement for DIM
- MAPS (MedAustron Publish-Subscribe) is a messaging protocol based on SIM
- Light-weight client implementation (easy to implement, even in LabView)
- MAPS clients connect to MAPS server



MAPS (cntd.)

- "Tag" based
  - Every published message is given a set of tags
  - Every subscription is defined by set of of tags
  - Subscriber gets a message only if message tag-set is super-set of subscription tag-set, e.g.

    - ["current"] - will get all the currents
    - ["current", "VA1"] - will get all the currents of virtual acc. 1

    - [] - give me all the messages
    - ["private1208"] - something private

- Automatic (transparent) reconnection/re-subscription of clients to a server

Roadmap

- XML parsing
- Dynamic component instantiation
- TINE integration
- Real-time capabilities
- Logging and error handling
  - *Remote*
- Entering/Leaving state methods
- Reduction of data points
- Scratch Pad
- Basic device support
- Improving development experience