# Simulations distribuées, multi-robots : pourquoi et comment garantir la répétabilité des résultats ?

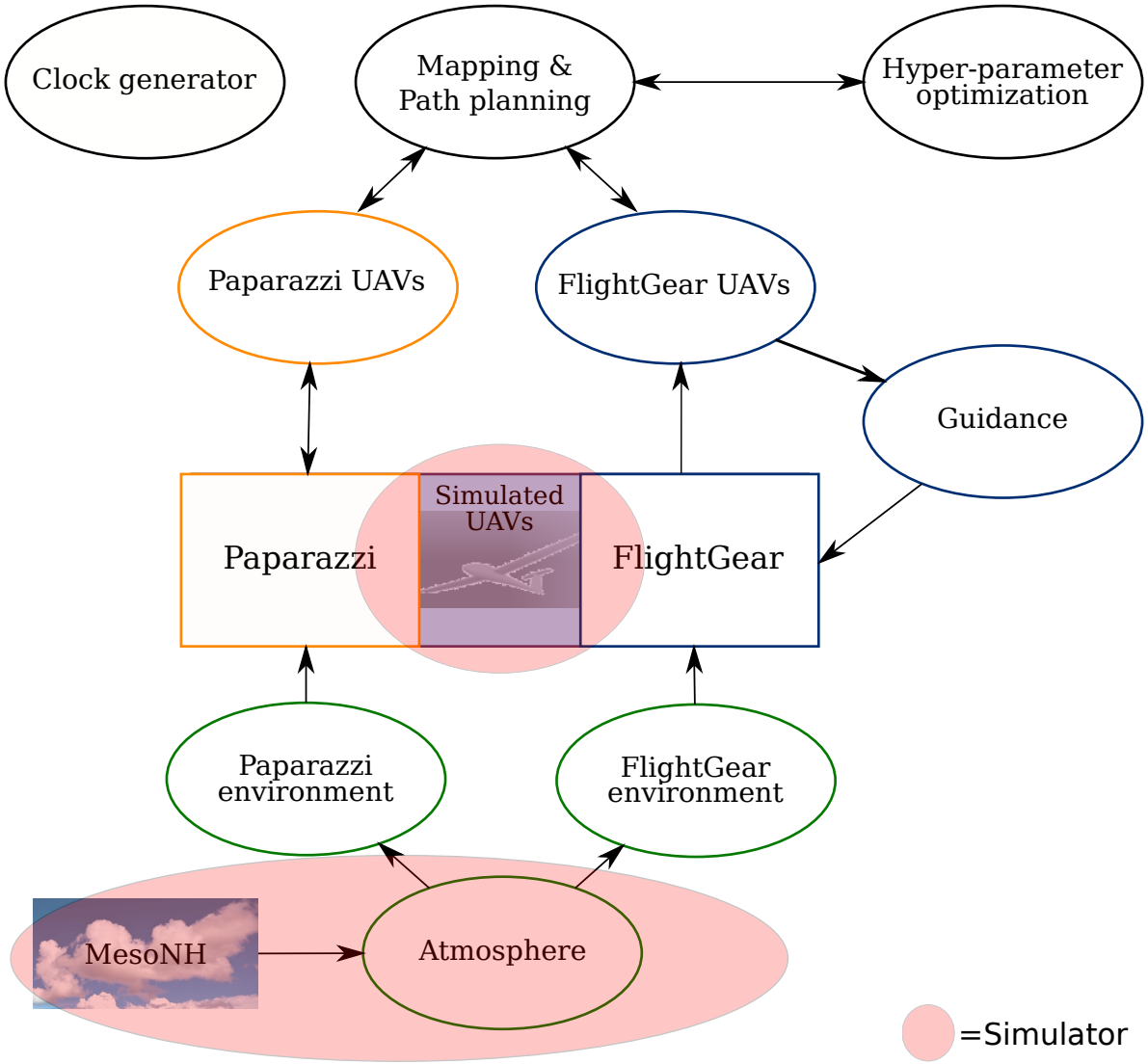
## Distributed, multi-robot simulations: why and how to ensure repeatibility of results ?

Christophe Reymann

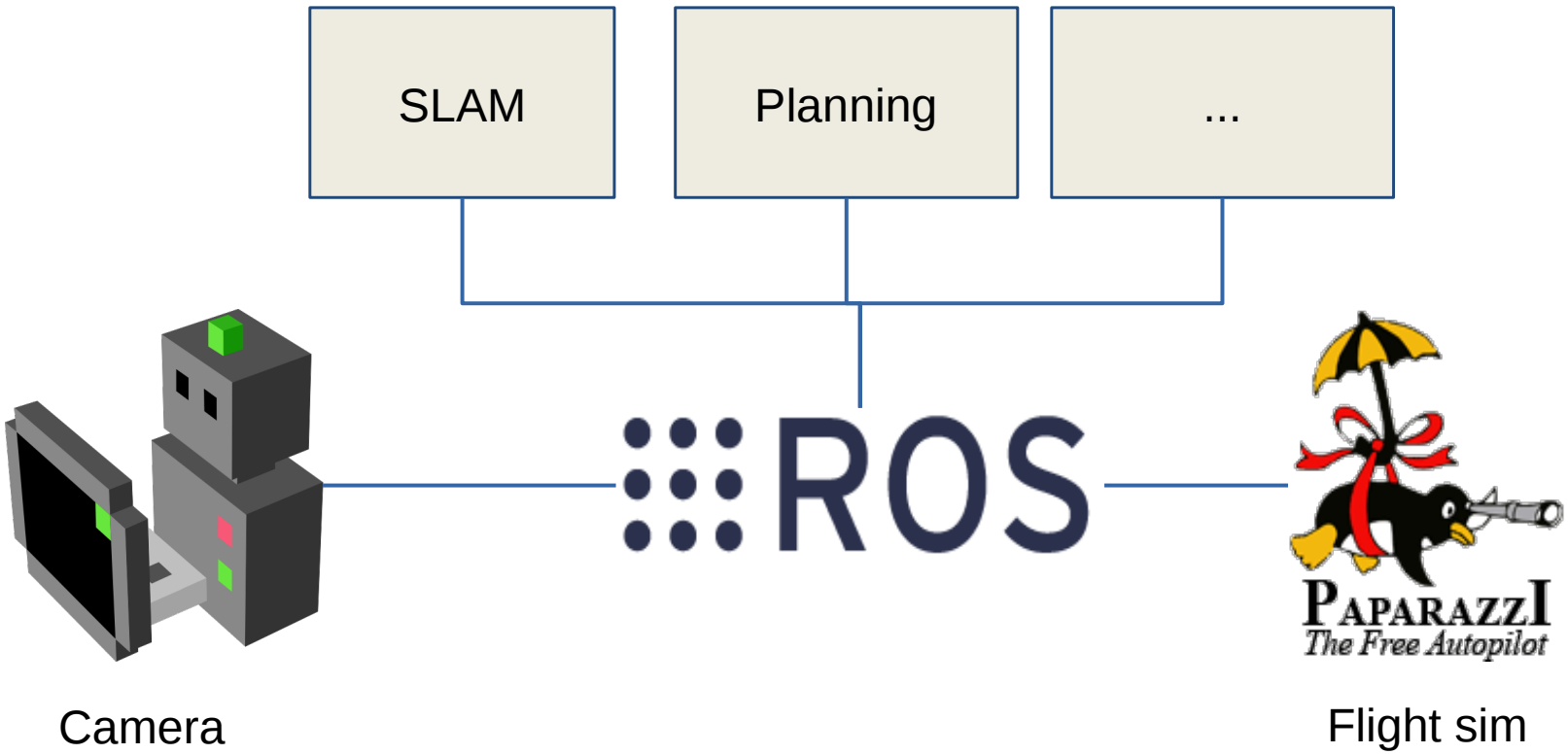
LAAS-CNRS

19-20 oct 2017

# Skyscanner


Clock generator
Mapping &
Path planning
Hyper-parameter
optimization
Paparazzi UAVs
FlightGear UAVs
Guidance
Simulated
UAVs
Paparazzi
FlightGear
Paparazzi
environment
FlightGear
environment
MesoNH
Atmosphere
=Simulator

# Precidrone


SLAM
Planning
...
ROS
Paparazzi
The Free Autopilot
Camera
Flight sim

# What do I want from the simulation framework?

- Compose distributed, modular simulations, interconnect simulators
- Switch and mix betw. simulated and real components
- Synchronize simulations, control time advancement
- Have repeatable simulation results
- Run realtime simulations
- Easy to integrate

# Overview

Case studies

Why do we need repeatability?

Why not HLA ?

Proposal: the DSAAM library

Discussion

# What is repeatability?

The same initial conditions should always produce the same simulation results.

Replicability $\neq$ Reproductibility

*Replicability is not Reproducibility: Nor is it Good Science*,

C. Drummond, 2009

- Reproductibility implies introduction of variability
- Replicability can't be used to prove scientific results
- Only useful to detect scientific fraud (debatable)

# What is repeatability's purpose for the scientist ?

We are not (only) scientists, but also (mostly) software developpers.

Bug finding and checking:

- Removes unwanted variability
- Replicate bugs
- Regression testing

Ease of development and use:

- Removes unwanted variability
- Prototyping with non-realtime algorithms
- Launching of simulations in batches

Verifiable results ?

# Is my simulation repeatable ?

Probably not, if you are using a robotic simulator.

# What deters repeatability ?

System variability:

- Networking delays
- Dropped messages (full buffers)
- System power & load
- Scheduling policies
- ...

Interaction simulation $\leftrightarrow$ tested algorithm(s):

- Out of order message processing
- Non-deterministic algorithm
- ...

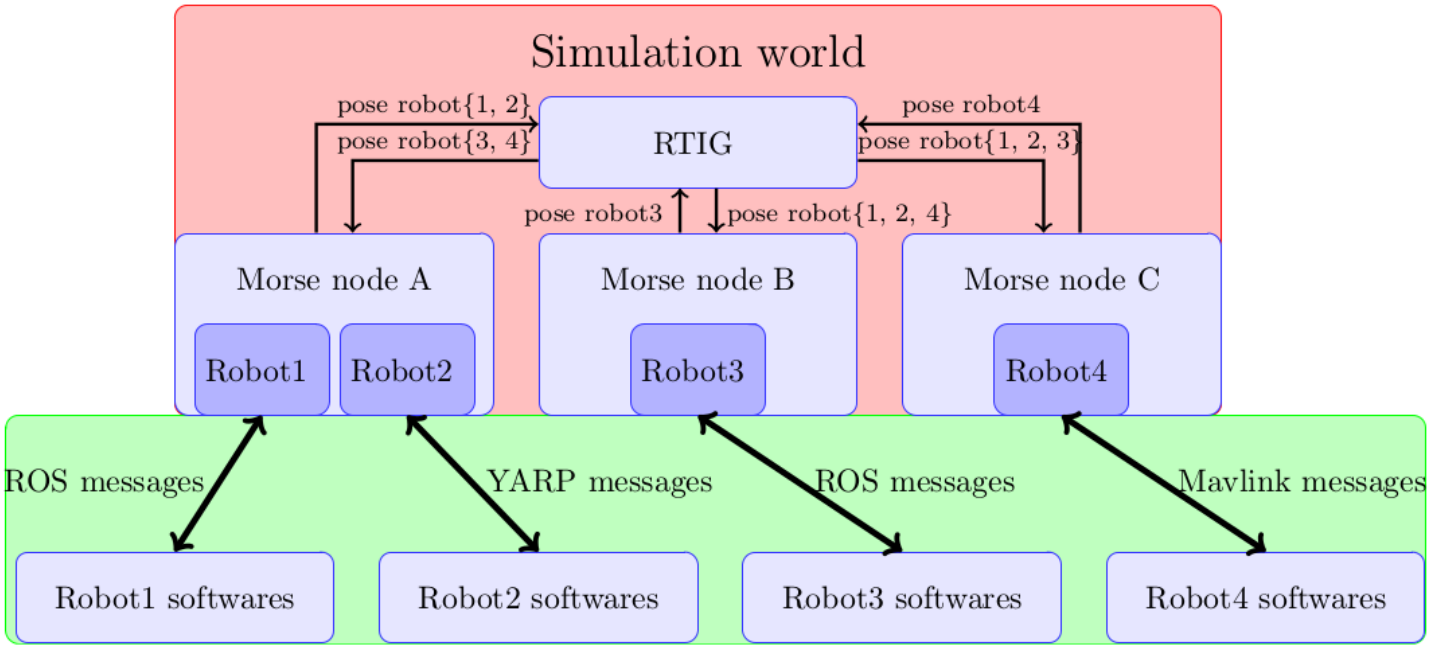
# Brief intro to HLA

Industry/Military standard

- Interconnect simulators (exchange state)
- Manage simulation
- Time management
- Best effort (realtime) mode

... what about DIS ?

# HLA integration in MORSE



# Time management in HLA

- Sim. wants to advance time to T
- Sim. sends request
- RTI delivers all messages up to T
- Sim. request is granted
- Sim. advances time to T

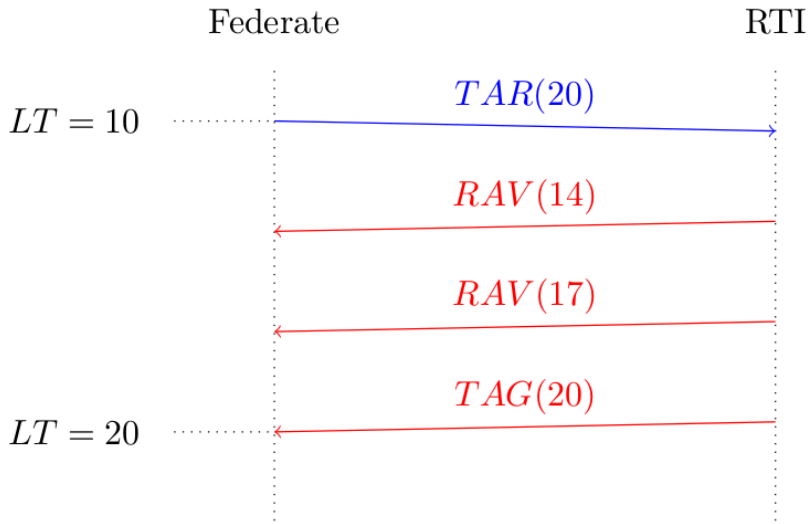



Fig. 2. Typical execution sequence

# Advantages and drawbacks

Pros:

- It just works™
- Provides time management
- It is an industry standard
- Centralized architecture

Cons:

- Military standard
- Quite complex to use, not many training resources
- Dependency on the RTI: no API/wire compatibility
- Adds another middleware for the roboticist
- No tools, no open-source ecosystem
- Centralized architecture

DSAAM = Decentralized Synchronisation Architecture for Asynchronous Middleware (name may change)

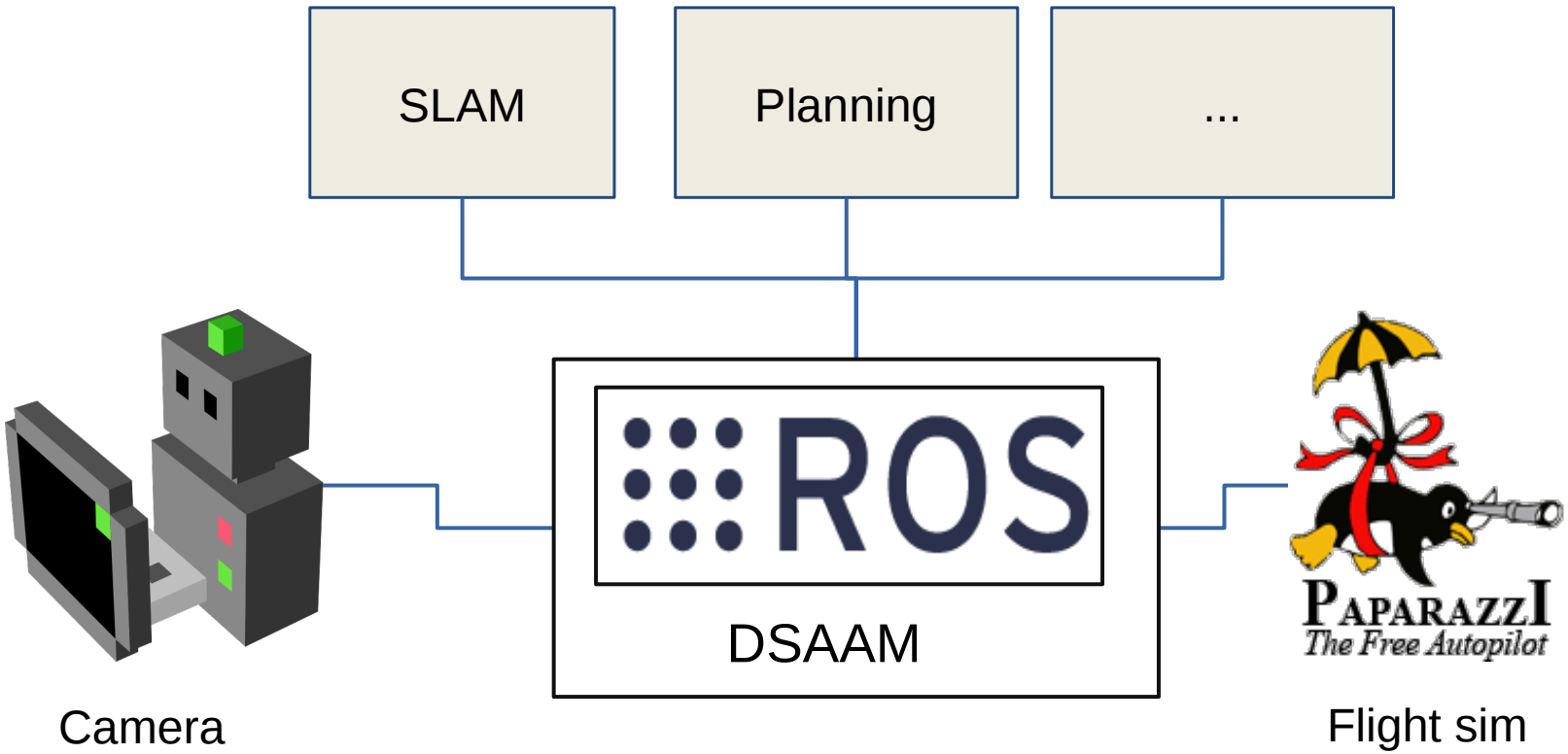
- (Only) provides time management (so far)
- Between com. middleware and app.
- May work with any kind of message passing communication middleware
- As least intrusive as possible

Minimal requirement: Messages should include timestamp

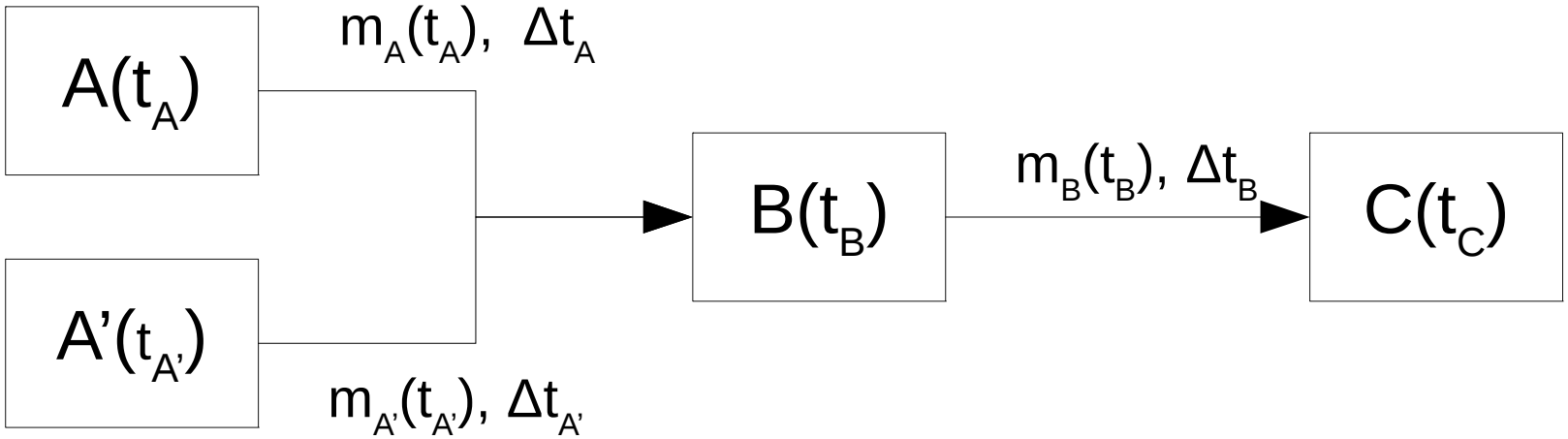
Current status:

- POC with raw threads and ROS
- native Python & C++ implementation

# Precidrone


SLAM
Planning
...
ROS
DSAAM
Camera
Paparazzi
The Free Autopilot
Flight sim

# Introduction



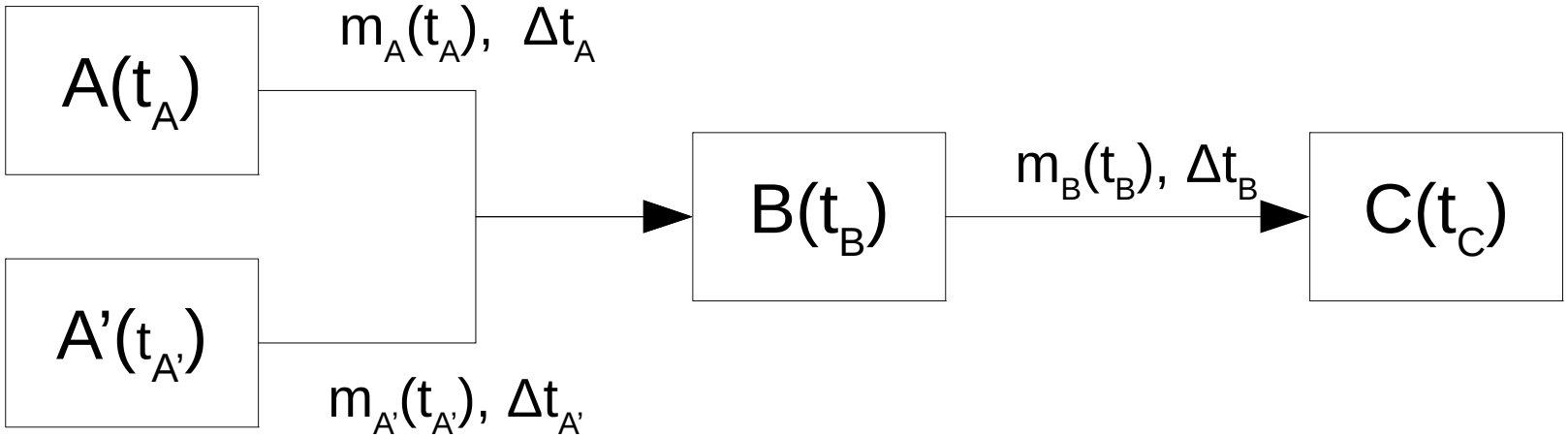

Node:

- has input and output message flows
- and it's own clock

Message flow:

- Message exchange between nodes
- Message are timestamped
- Has a message rate (evt. dynamic)

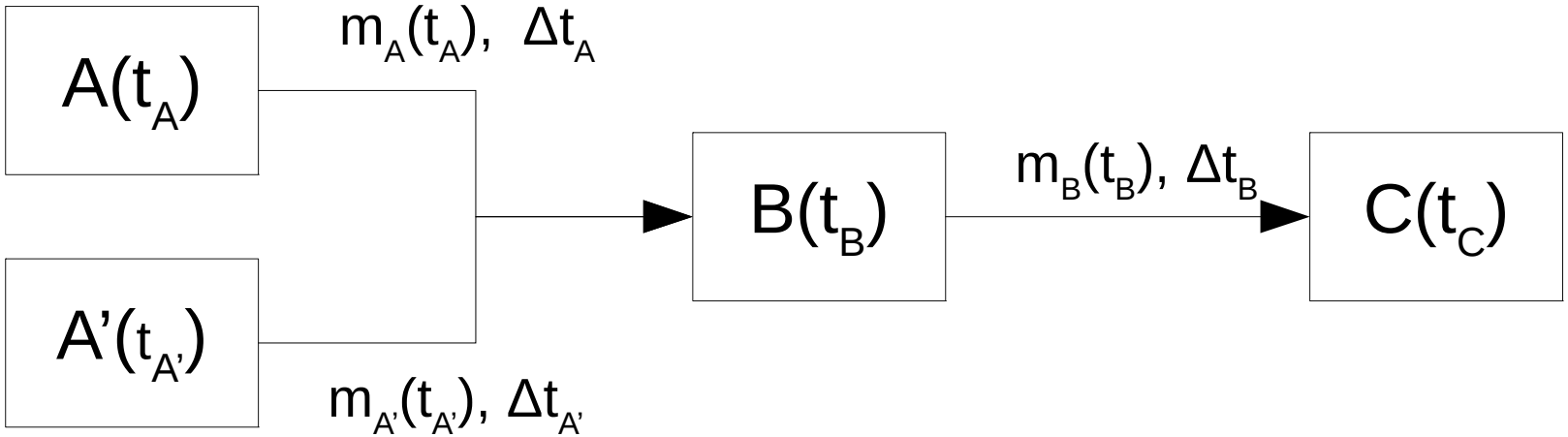
# Time management (1)




Node can only advance time if all messages have been processed up to that time:

$$m_B(t), \Delta_{t_B} \Rightarrow \text{ next incoming message for B at } t' >= t \quad (1)$$

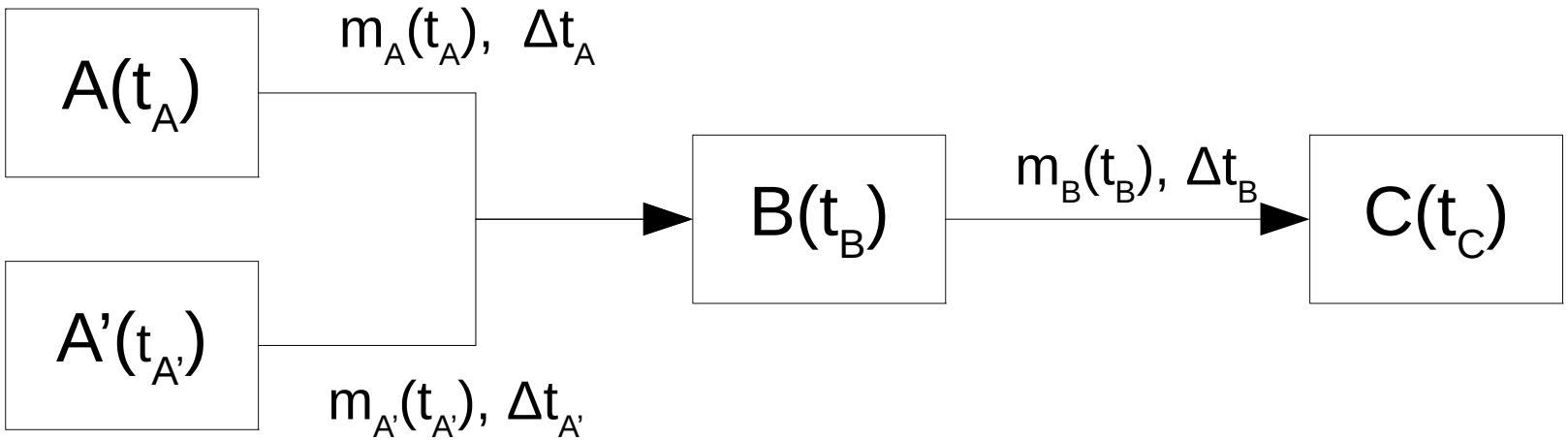
# Time management (2)




Next published message timestamp can be deduced from the previous one:

$$m_B(t), \Delta_{t_B} \Rightarrow \text{ next outgoing message } m_B(t + \Delta_{t_B}) \qquad (2)$$

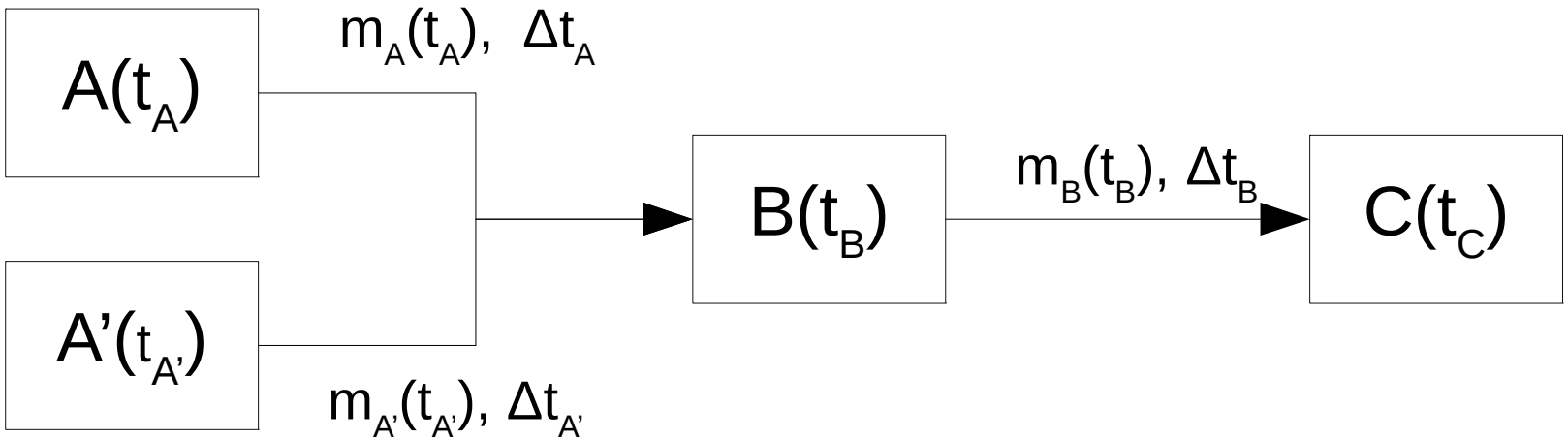
# Time management (3)




Messages are processes chronologically:

$$t < t' \Rightarrow m(t) \text{ processed before } m'(t') \text{ by the node} \qquad (3)$$

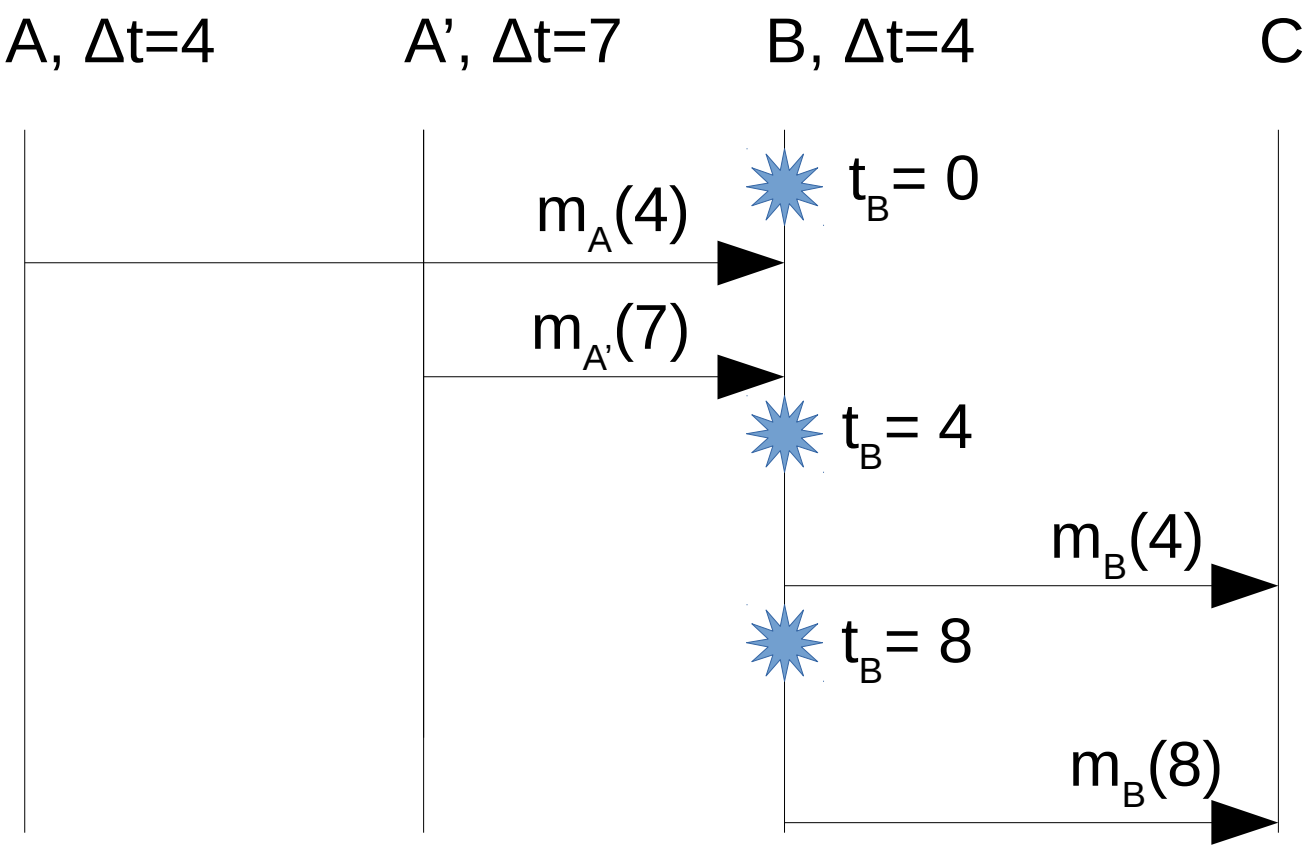
# Time management (4)



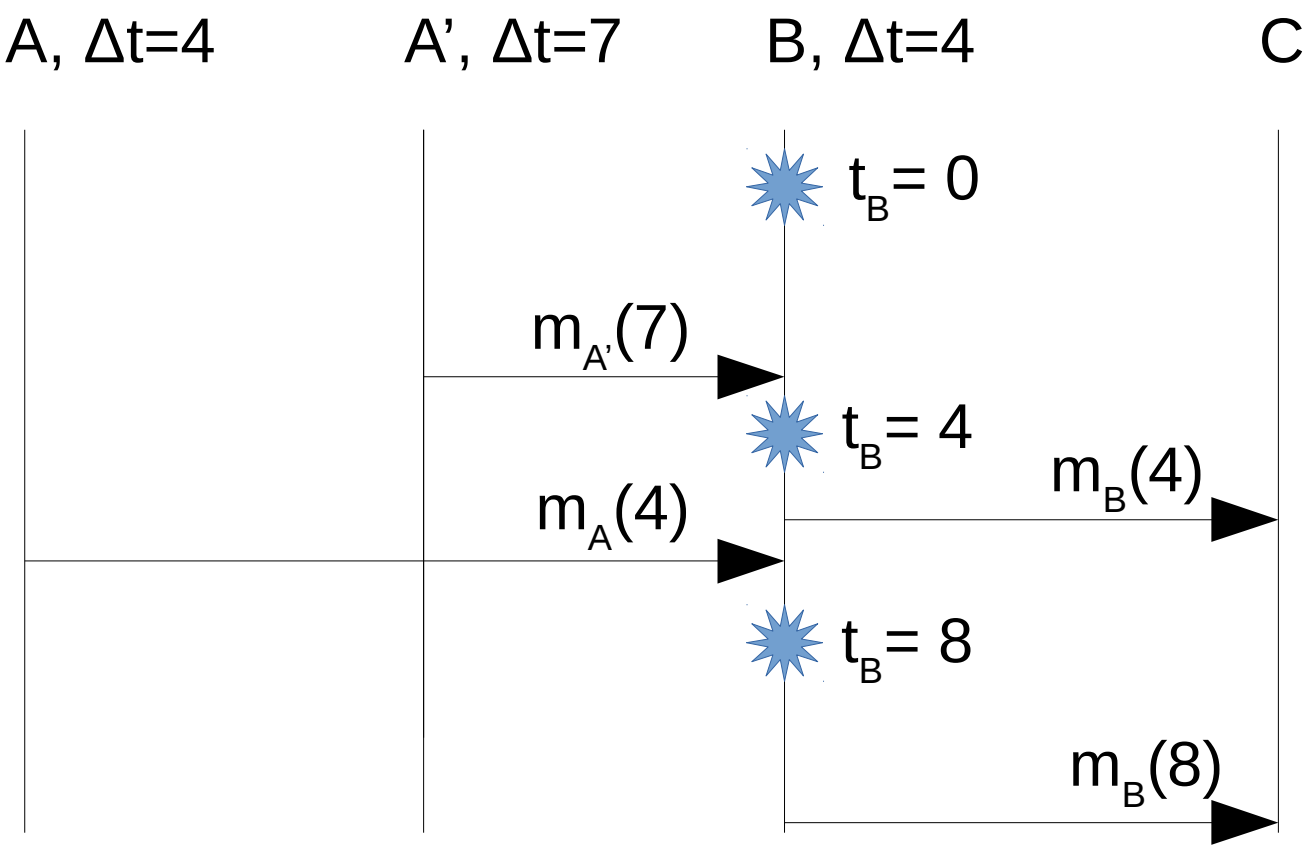

No message shall be lost:

$$\text{B emits message} \Rightarrow \text{ receivers queues are not full} \qquad (4)$$

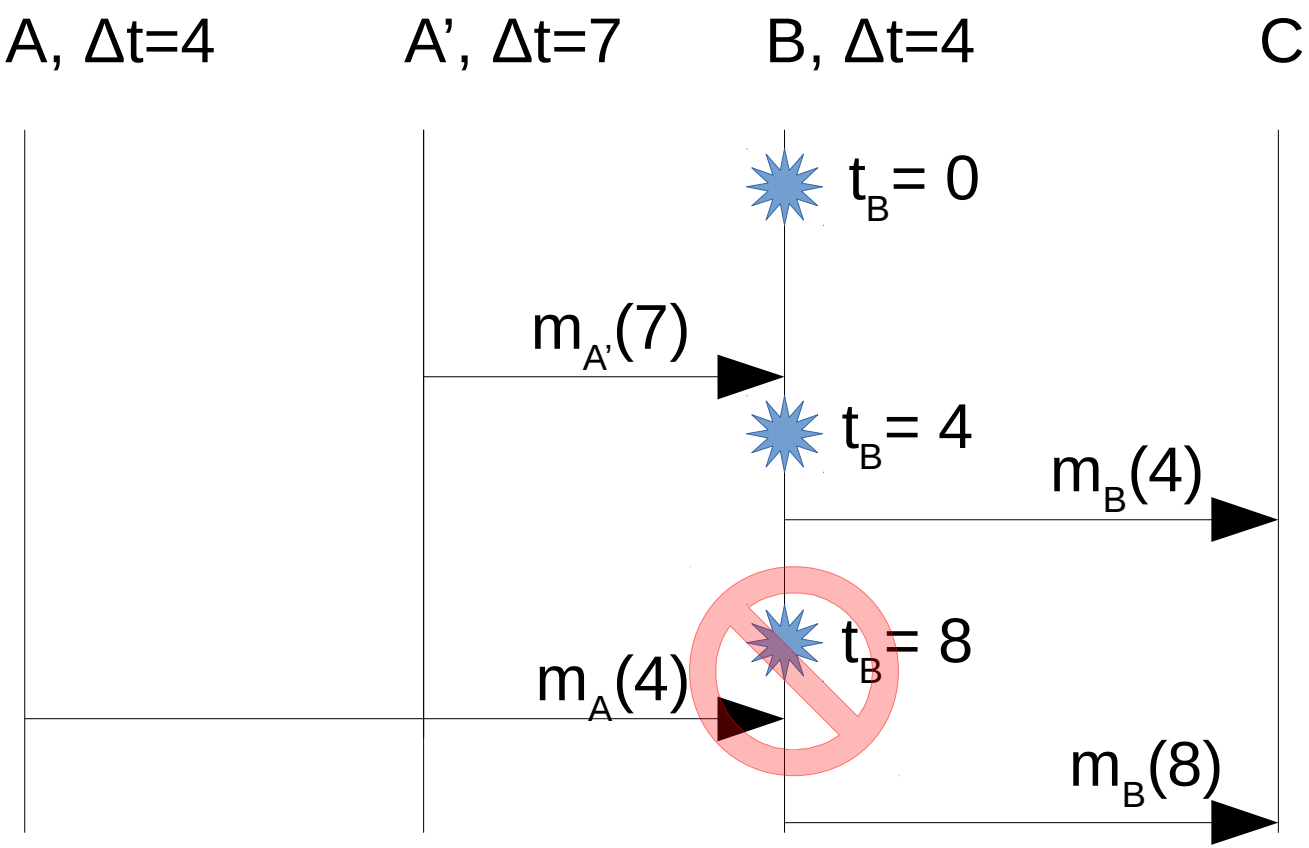
# Example (1)



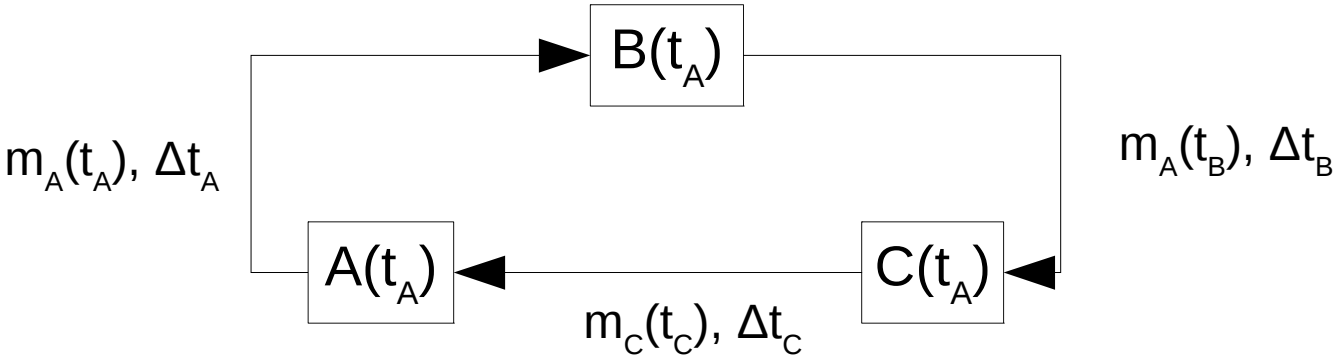
# Example (2)

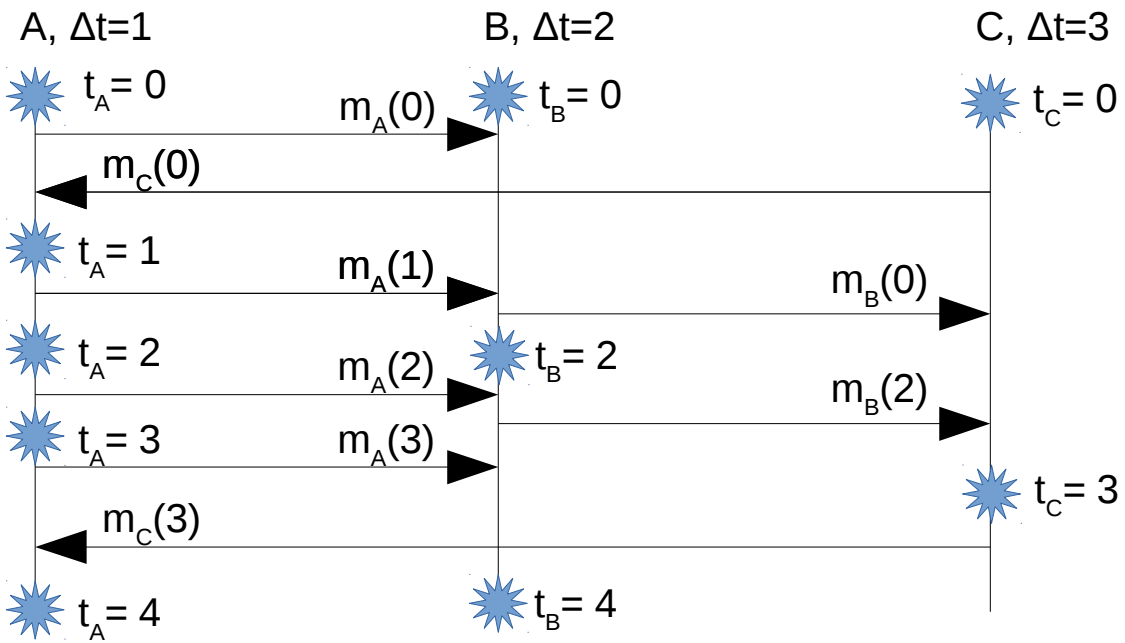


# Example (3)

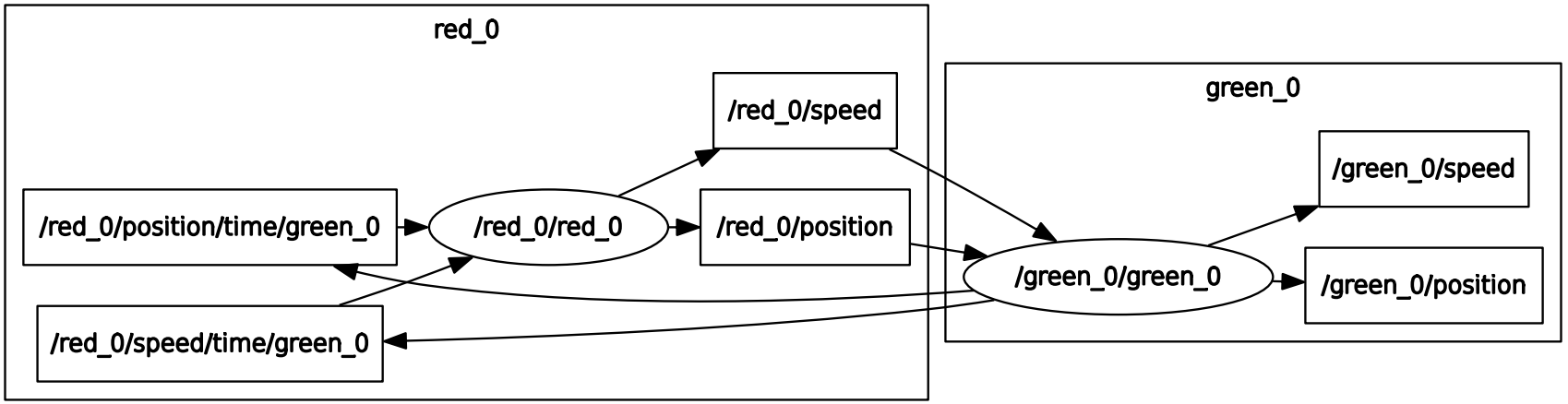


# Example with cycles



# ROS example


red_0
/red_0/speed
/red_0/position/time/green_0
/red_0/red_0
/red_0/position
/red_0/speed/time/green_0
green_0
/green_0/speed
/green_0/green_0
/green_0/position

# Dummy example : ROS API in Python (1)

```
from geometry_msgs.msg import PointStamped
from dsaam.ros import RosNode, Time

def process_pos(name, message, next_message_time):
    #process message

node = RosNode('B', start_time=Time(0), dt=Time(4),
               queue_size=3)

node.setup_publisher('/B/position', PointStamped,
                     dt=Time(4),
                     subscribers=['C'])

node.setup_subscriber('/A/position', PointStamped,
                      callback=process_pos,
                      dt=Time(2))
```

# Dummy example : ROS API in Python (2)

```
(...)

node.init_ros()

while true:
    t = node.next()
    if t > node.t + node.dt:
        m, t = simulation_step()
        node.send('/C/position', m, t)
```

# Summary: what's good

- Decentralized time management
- Low overhead, no additional copy

- Can be implemented with any p2p message passing middleware
- Possibility of dynamic message rate

- Easy to integrate
- (Almost) transparent for apps. using "raw" middleware
- Benefits from the existing middleware ecosystem

# Summary: what's bad

Difficult to fix:

- Cannot be used with mixed middlewares
- Not compatible with event based simulation (HLA is)

Could be improved or added:

- Message ACK overhead
- No simulation management
- Static flows, no dynamic reconfiguration (cf prev. point)
- No realtime/best effort mode

# Questions

For the audience:

- Is repeatability a concern for you?
- Do you find the proposed solution interesting?
- If not, why?
  - Any game stoppers?
  - Missing features?
- ... vs HLA?