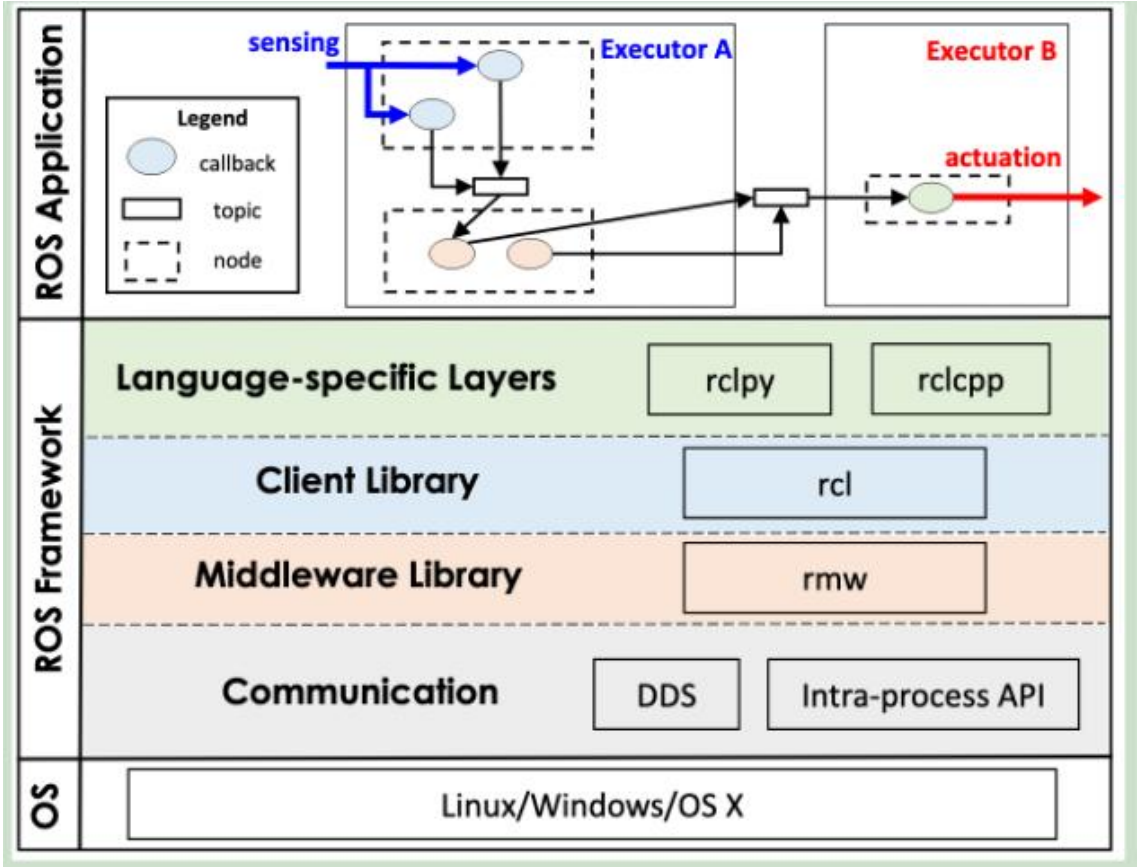
ROS 2

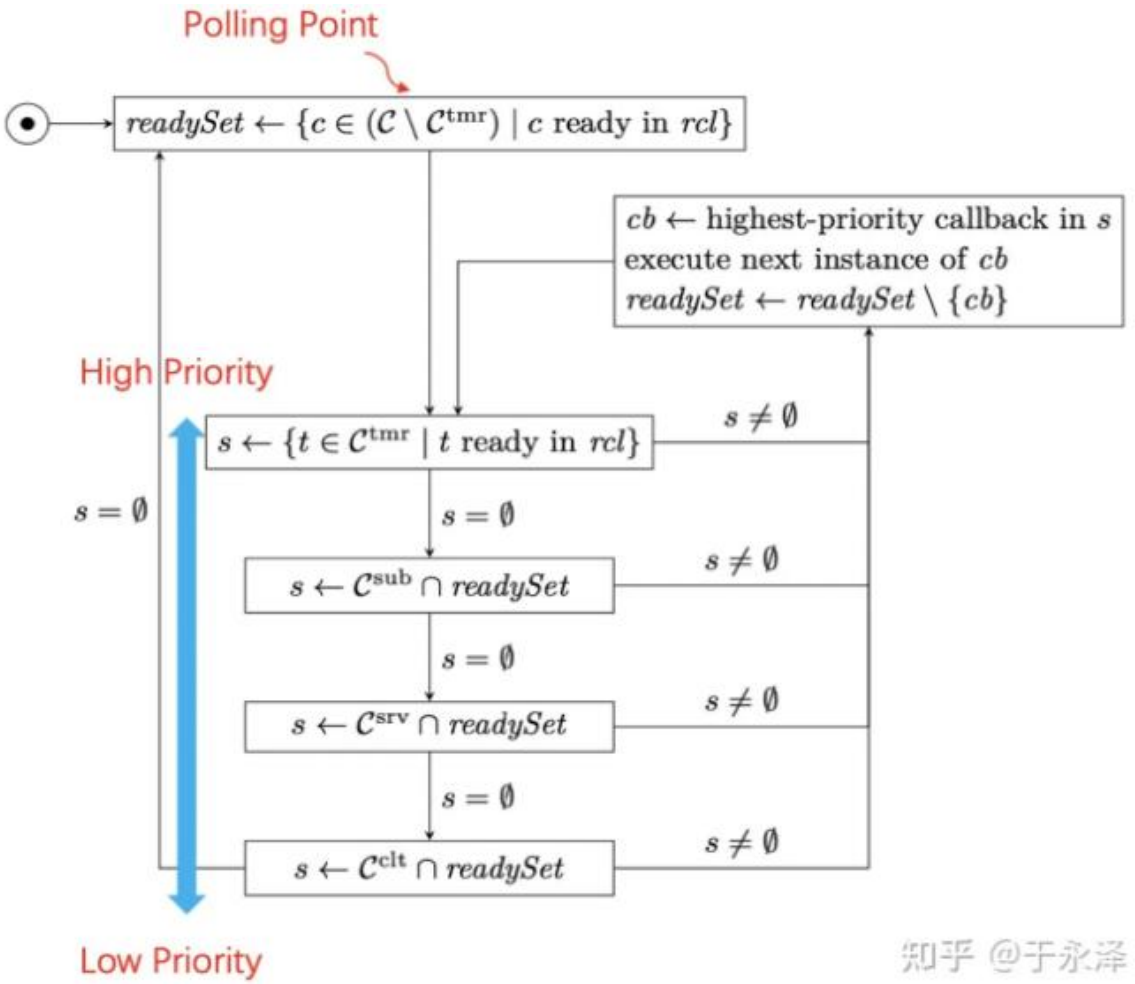
Architecture

- **Callback**: Minimum scheduling entity
  - Including timer, subscription, service, and client callbacks.
  - **Timer callbacks have a period (time trigger).**
  - **Others are regular callbacks (event trigger)**
  - With priority attribute: Timer>Subscription>Service>Client
- **Node**: publish–subscribe communication
  - Nodes publish messages on a topic, and nodes subscribed to the topic process each message by activating a callback.
  - **Can only running on the only executor**
  - When processing multiple callback functions, ROS executes them sequentially in the order in which messages are received.
  - Messages published by nodes to multiple topics are processed concurrently
- **Executor**: schedule callback execution
  - The executor itself occupies **a thread**
  - Nodes are assigned to different executors, and all callbacks within the node are scheduled by this executor.
  - Divided into single–threaded executors and multi–threaded executors
  - Single–threaded executor: handle all callbacks in a single thread
  - Multi–threaded executor: multiple threads handle callbacks



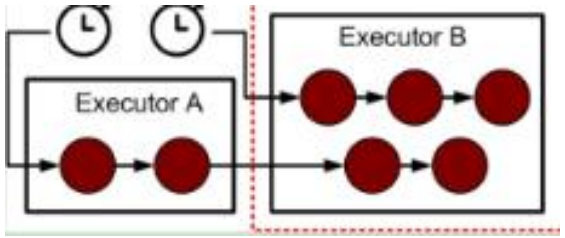
Scheduling

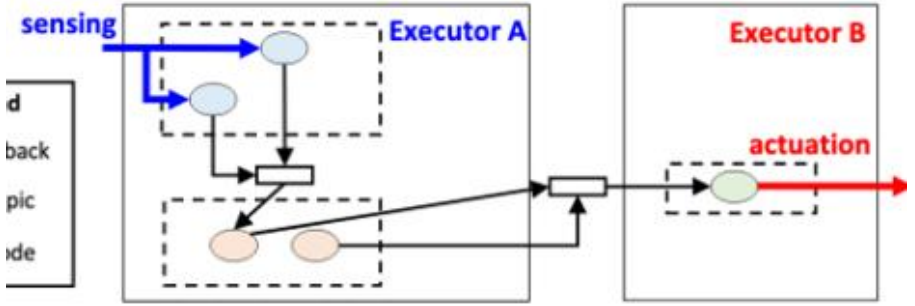
- **Polling point**: The executor reaches the polling point when it needs to select the next instance to execute but no instance is currently available.
- **Processing window**: between two polling points.
- **Single-threaded executor process**
  - Readyset is empty, reaching the polling point
  - All ready regular callbacks are added to readyset
  - The timer callback does not need to wait for the polling point, only needs to wait for the scheduled arrival
  - In a processing window, only the first instance of the callback is processed
  - The order in which instances are executed is determined by type priority:
  - Timers>Subscriptions>Services>Client
  - When the type priorities are the same, compare the callback registration time, and the earlier registration time has higher priority.
  - The first instance of all ready callbacks has been executed and readyset is empty.



Callback chain model

- The triggering relationship between callbacks constitutes the **callback chain**
- A callback chain can exist within a single executor, or span multiple executors.
  - Similar to a chain of tasks (callbacks) scheduled on a single or multiple cores (single-threaded or Multi-threaded executor), instances of tasks are similar to instances of callbacks
  - Callbacks only belong to one callback chain – single task chain model
  - Callbacks can belong to multiple callback chains – DAG model
- The callback chain usually starts with a timer callback, followed by regular callbacks.
- $C = \{C_{tm}, C_1, \ldots, C_n\}$, C_1^i, i-th instance of C_1 callback
- External events (e.g. sensors) are not constructs in ROS2 systems, but can be modeled as pseudo-callbacks
  - An external event interface is added, which can publish messages to ROS2 nodes and trigger corresponding callbacks to continue processing.
- Attributes:
  - Different instances of the same callback are executed in the order of activation. C_z^{i-1} must be completed before C_z^i can continue.
  - The executor selects a new instance only after the previous instance has completed. C_{z-1}^i is completed and can continue C_z^i





Message synchronization model

- Use the ApproximateTime strategy in ROS2 to synchronize messages from multiple topics with approximate timestamps
- Use a synchronization node to subscribe to messages from multiple other nodes, merge data from different topics at the synchronization node, and then output it to other required nodes after integration.
- Message sequence $M_i = \{m_i^1, m_i^2, \ldots\}$. m_i^k represents the k—th message generated by the i—th sensor. t(m_i^k) represents the timestamp (sampling time).
- Q_i represents the message queue of the i—th channel, and the corresponding message sequence is M_i. It will not block if it is infinite. The difference in timestamps of two consecutive messages in Q_i is at least T_i^B and at most T_i^w.
- ApproximateTime strategy:
  - The end of the message queue is the prediction point
  - When new news arrives, it is inserted into the end of the queue and the original prediction point is deleted.
  - Generate new prediction points based on new message timestamps
  - When messages arrive in all message queues, the one with the largest timestamp is selected as the pivot.
  - Messages that meet the time difference requirements are formed into a candidate set.
  - Select the output with the smallest time gap in the candidate set
  - The output message and the queue messages before it are cleared

Related work

- Time analysis based on callback chain
  - The impact of callback chain priority
    - PiCAS：New Design of Priority—Driven Chain—Aware Scheduling for ROS2
    - Response Time Analysis for Dynamic Priority Scheduling in ROS2
  - Time analysis of callback belonging to a single task chain
    - Response Time Analysis and Priority Assignment of Processing Chains on ROS2 Executors
  - Time analysis of callbacks belonging to multiple task chains
    - A ROS 2 Response—Time Analysis Exploiting Starvation Freedom and Execution—Time Variance
- Analysis based on message synchronization
  - Selection of output message set
    - SEAM：An Optimal Message Synchronizer in ROS with Well—Bounded Time Disparity
  - Queue overflow processing
    - Modeling and Property Analysis of the Message Synchronization Policy in ROS
  - End—to—end time of message synchronization
    - Worst—Case Time Disparity Analysis of Message Synchronization in ROS
- Analysis based on multi—threaded executor
    - RTeX：an Efficient and Timing—Predictable Multi—threaded Executor for ROS 2