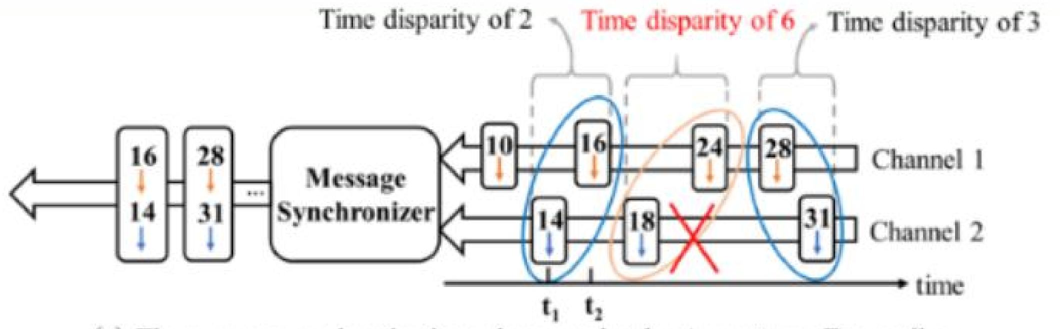
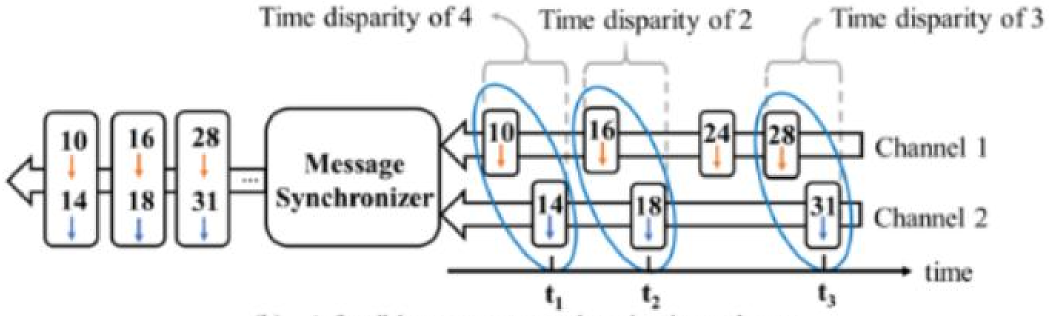
- SEAM — An Optimal Message Synchronizer in ROS with Well—Bounded Time Disparity (RTSS2023)
- The message synchronization mechanism in ROS 2 has been improved to support multi—sensor data fusion and ensure time consistency (simply put, the earliest message is synchronized within a certain range)
- The message in Ros2 is simply the data to be processed. In this article it can be either periodic or aperiodic.
- Traditional message synchronization ：
  - ExactTime：
    - Messages with exactly the same timestamps are selected for synchronization to form an output message set, while other messages that are not strictly synchronized are discarded.
    - Too strict and difficult to achieve
  - ApproximateTime,
    - Select the message output set by predicting arrival time （ Predict the time of the next message and calculate the time difference ）
    - Rely heavily on prediction, failure of prediction will lead to serious consequences for subsequent synchronization
- SEAM：
  - It does not rely on strict synchronization, does not rely on prediction, has a small amount of calculation, and is proven to be the optimal synchronization strategy.
    - For message sequences, the synchronization algorithm that generates more valid messages within a certain period of time is more optimal.
  - Only as part of ros2

- SEAM：
    - Message synchronization is specified within a range of time windows.
    - When new messages enter the synchronizer they are added to their respective queues （where their from）.
    - Until each queue has a message, the one with the largest timestamp in all the queue is selected as the basic message.
    - Find messages within the specified range (compared to the base message) and add them to the selection set.
    - In the candidate set, find the earliest time in each queue and add it to the output set.
    - If the output set includes messages from all queues, consolidate the messages and output them, and then remove all messages before the basic message in the candidate set.
    - If does not contain all queues, wait until satisfied

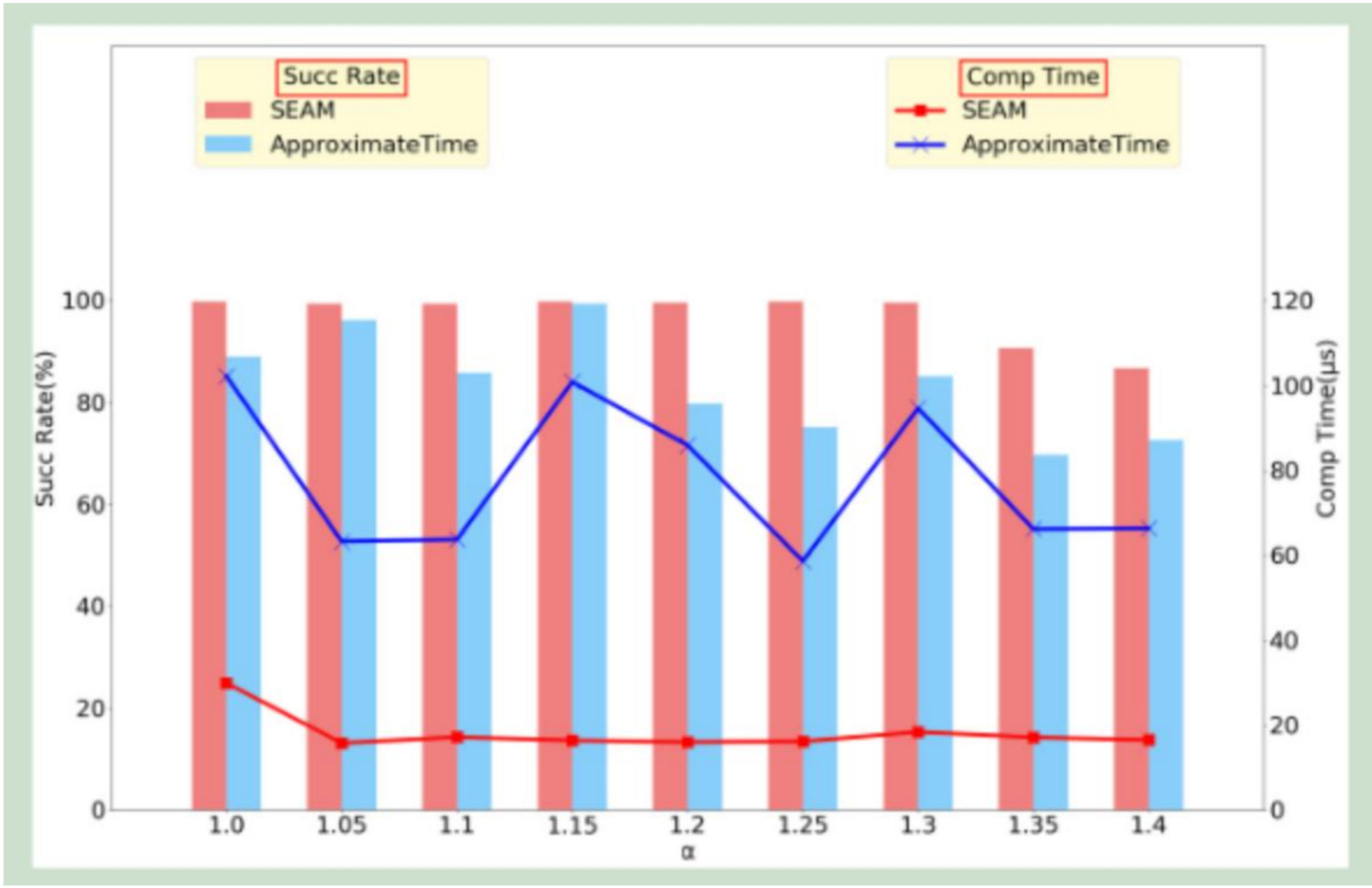


(a) The message synchronization scheme under the *ApproximateTime* policy



(b) A feasible message synchronization scheme

- SEAM:
  - **This article is significantly better than other methods for message synchronization. If improvements should be made, consider whether there are unsuitable scenarios.**
  - This paper summarizes the assumption that the delay experienced by a message varies randomly between 1 ms and 40 ms. **This may not be consistent with some actual industrial control scenarios**
  - As the jitter ratio α (ratio of maximum and minimum periods) increases, the success rates of both methods decrease. Because increased latency results in fewer messages being available within a fixed time interval.
  - **There is research on jitter in ROS, but it may be more focused on the system rather than the task chain.**



- Parallel Path Progression DAG Scheduling (ITC2023)
- Sustainable response time analysis for an arbitrary set of paths with a maximum number of processors
  - if there M processors, it is M—paths analysis
  - Find a set of paths that completely cover the DAG or an approximation of a bounded worst—case response time via a polynomial—time algorithm. And consider the changes with resources to ensure sustainable analysis and find the approximate optimal solution within the range according to the polynomial algorithm.
- Based on the above analysis, hierarchical scheduling is proposed:
- Gang Reservation Provisioning:
  - A group of processors are required to be available to complete a task together. The task first tries to find a set of processors that need to be all idle during execution. It is suitable for parallel work that requires a large number of processors to reduce switching overhead. But a waste of resources. .
- Ordinary Reservation Provisioning:
  - Distributed execution, each processor processes part of the task (ordinary DAG, task fragments do not need to be executed synchronously) makes better use of resources, but the implementation is complicated and considers how to allocate it among processors.

  - Similar resource analysis (non DAG) on multi—core processors has been done
  - LAG—based Analysis for Preemptive Global Scheduling with Dynamic Cache Allocation

- Real–Time Scheduling of Autonomous Driving System with Guaranteed Timing Correctness
- Provide an automated design–analysis–redesign process to improve the design efficiency of autonomous driving systems.
- An integrated framework is proposed to jointly analyze the schedulability of individual tasks and the end–to–end latency of task chains in multi–rate DAGs.
- Integer Linear Programming (ILP) techniques are designed to remove redundant workloads to increase the chance of meeting timing requirements.
- The proposed framework realizes the automated process of iteratively creating, analyzing, and improving AD system design.
    - That is, the analysis results of the previous iteration provide valuable guidance for redesigning the AD system in the next iteration.
    - During the iteration process, it is necessary to ensure that the system meets both network and physical time constraints, use ILP to reduce unnecessary workload, task set schedulability and end–to–end timing. Unsatisfied iterations will have additional constraints.

    - **The scope of the study is more towards the entire system rather than focusing on time or other constraints**

- Two articles related to DAG, considered from the perspective of path and probability respectively

- Bounding the Response Time of DAG Tasks Using Long
- The traditional Graham bound method is too conservative in some cases because it assumes that vertices on non—longest paths will execute in parallel with vertices on the longest path, thus interfering with each other.
- A new response time bounding method is proposed that considers the length of multiple longer paths. Workloads that must execute in parallel and cannot interfere with each other are more precisely identified (via virtual paths and constrained critical paths).
- Virtual Path
    - A collection of vertices executed in different time units. There is no direct sequential relationship between vertices. The length is the sum of the execution times of all vertices it contains
    - Since vertices on the virtual path will not be executed in parallel, those not on it are considered for parallel execution.
- Restricted Critical Path
    - A set of vertex sequences selected by a specific rule for a given execution sequence

- Minimizing Probabilistic End—to—end Latencies of Autonomous Driving
- Reduce the probabilistic end—to—end delays (PEELs) of task sequences through two stages.
  - PEELs take into account the uncertainty of task execution time
  - PEELs can give the worst—case delay for task completion at a given confidence level. For example, it can be expressed as "With 99% confidence, the execution delay of the task will not exceed a certain value."
- Two stages:
- Baseline schedule generation
  - Estimate the lower bound of the total unavailable idle time and generate a baseline schedule to avoid unavailable idle time as much as possible, thereby improving the feasibility of scheduling.
  - Non—available idle time: The processor has no tasks to execute at a certain time, and this time period cannot be used to start subsequent tasks in advance.
- Mission phase optimization
  - Search for optimal task phase combinations to further reduce PEELs for a specific task sequence

  - **There has been a lot of work on end—to—end analysis of DAG models**
  - **The jitter problem under the DAG model has not yet been considered.**