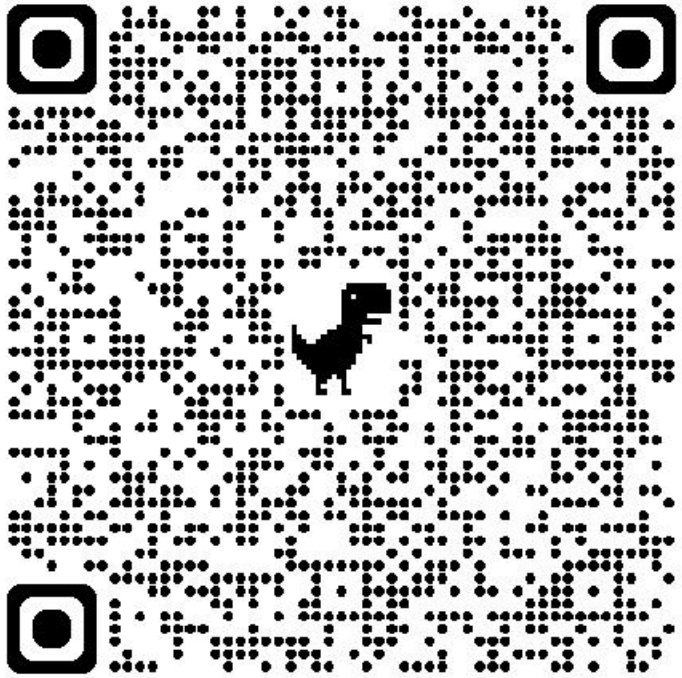


Apollo星火计划自动驾驶技术培训报名

apollo | 开发者社区 | 星火计划 SPARK PROJECT

CyberRT

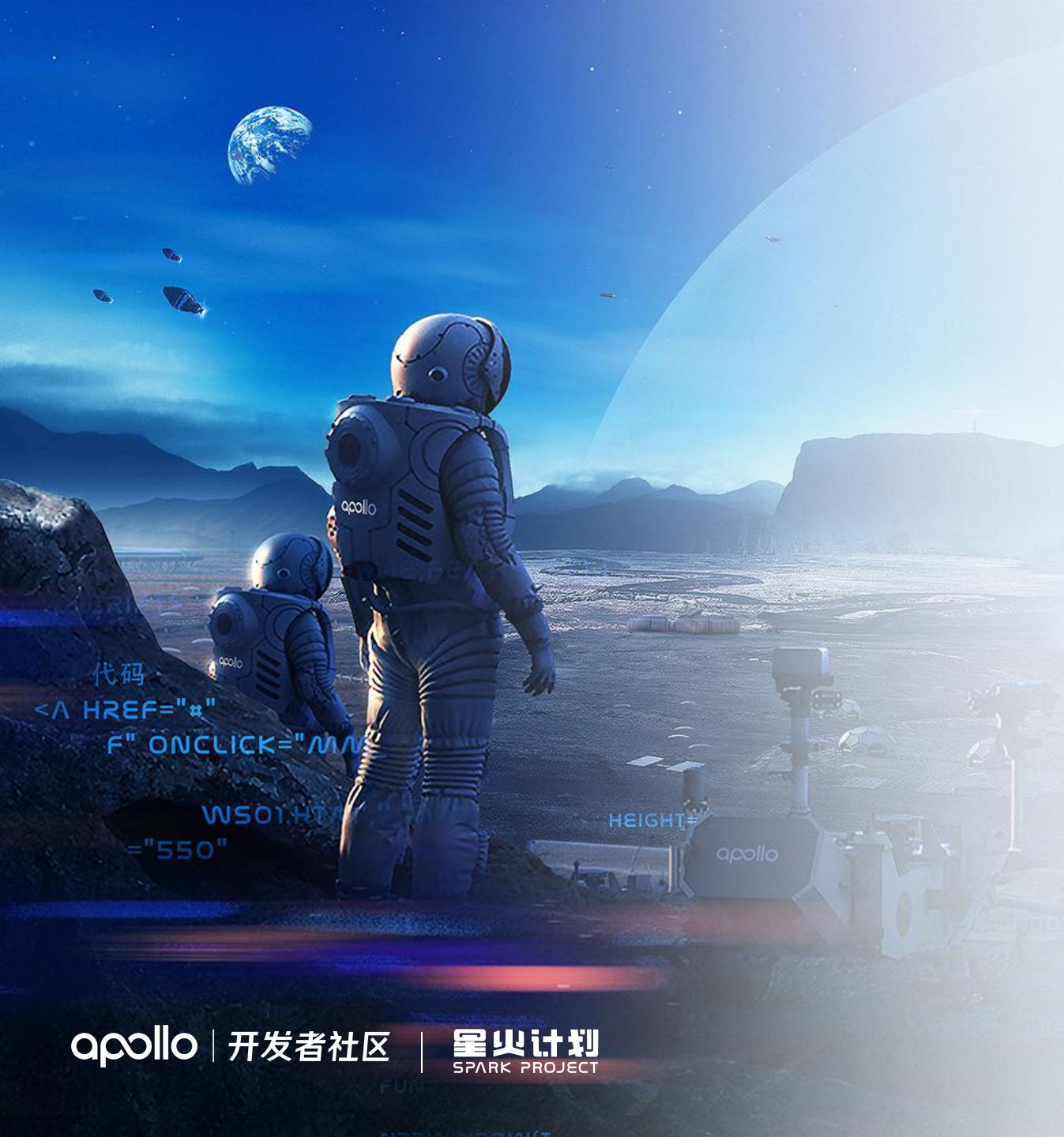
讲师：王翼

王翼 (yiak.wy@gmail.com)

内容概要，学习目标

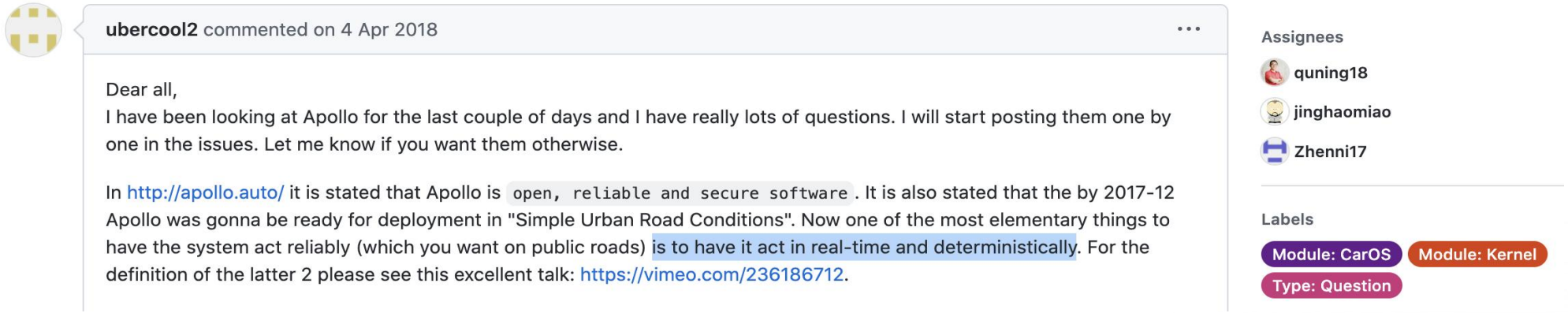
1. 了解 CyberRT 在 Apollo生态的位置，以及的主要作用

2. 了解 CyberRT 基本概念和用法



CyberRT诞生背景

实时操作系统的引入Cyber Issue in Github #3707 ：



ubercool2 commented on 4 Apr 2018

Dear all,
I have been looking at Apollo for the last couple of days and I have really lots of questions. I will start posting them one by one in the issues. Let me know if you want them otherwise.

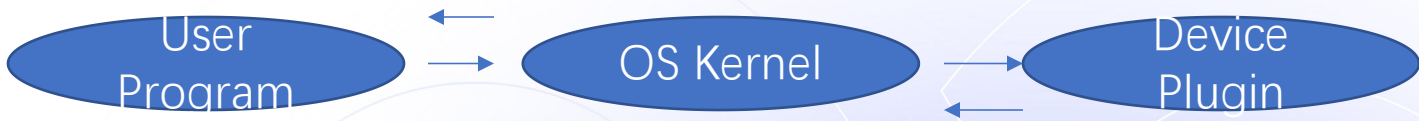
In http://apollo.auto/ it is stated that Apollo is `open, reliable and secure software`. It is also stated that the by 2017-12 Apollo was gonna be ready for deployment in "Simple Urban Road Conditions". Now one of the most elementary things to have the system act reliably (which you want on public roads) is to have it act in real-time and deterministically. For the definition of the latter 2 please see this excellent talk: https://vimeo.com/236186712.

Assignees
quning18
jinghaomiao
Zhenni17

Labels
Module: CarOS
Module: Kernel
Type: Question

开发者希望系统可以在实时操作系统上运行：

- 静态分配（static allocation） - not too much new/malloc ! Memory pool & static 更好 。
- 非阻塞调用
- 进程内通信 - 代替网络协议栈通信



Apollo 3.5版本以前构建在通用分时多户操作系统(Time-Sharing OS) Linux上，基于ROS1的通信模型。

CyberRT正是为了解决调度、通信和任务管理实时响应要求而诞生。

CyberRT诞生背景

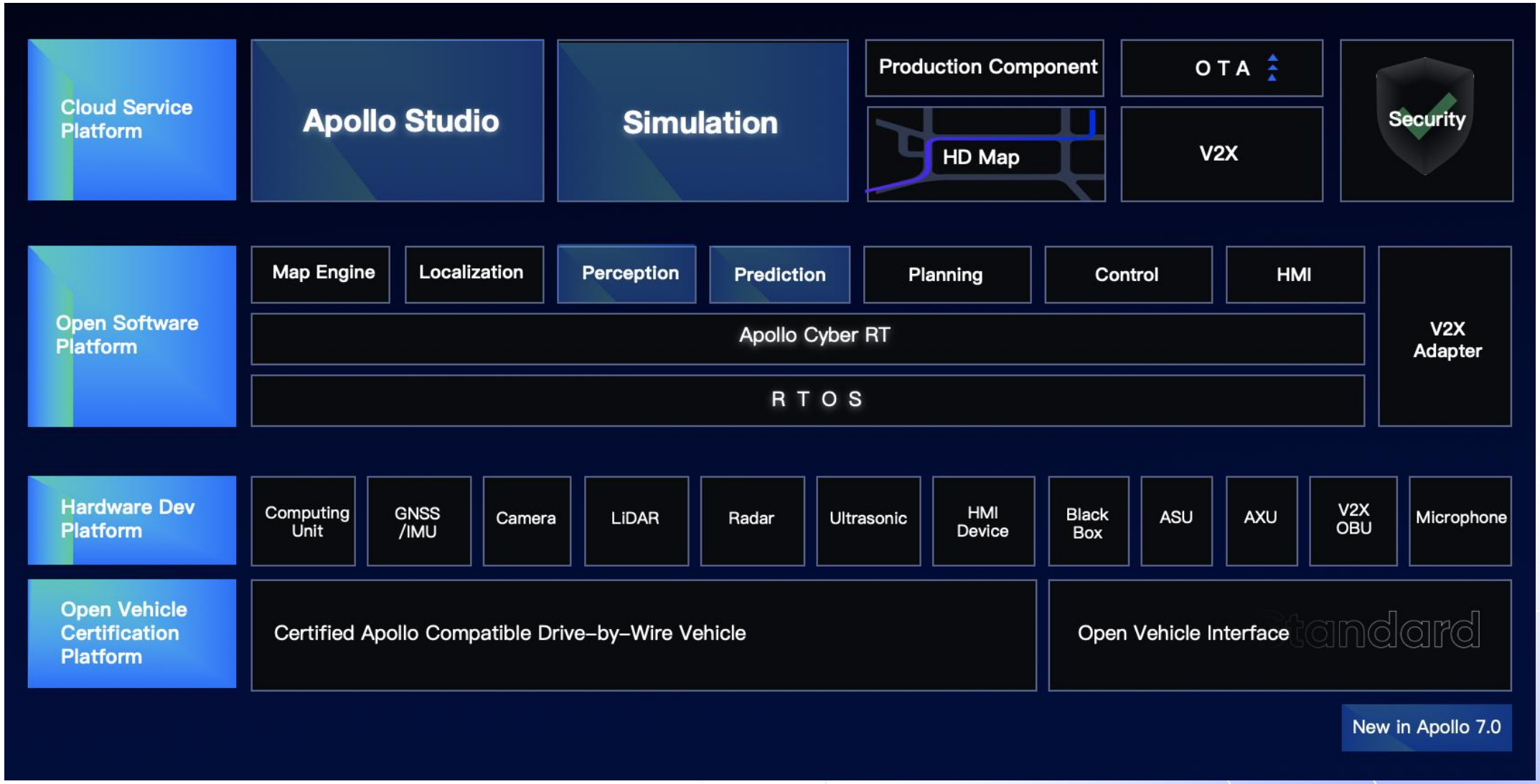


Fig 1 : from apolloauto 7.0 arch

CyberRT诞生背景

- History
  - Apollo 3.5 : Real Time Framework
- Feature
  1. 满足自动驾驶的任务需求和实时性
     - 任务调度（DAG）
     - 协程和内存管理（RTOS/TSOS 扩展）
     - 数据融合（Multi-in, Multi-out）

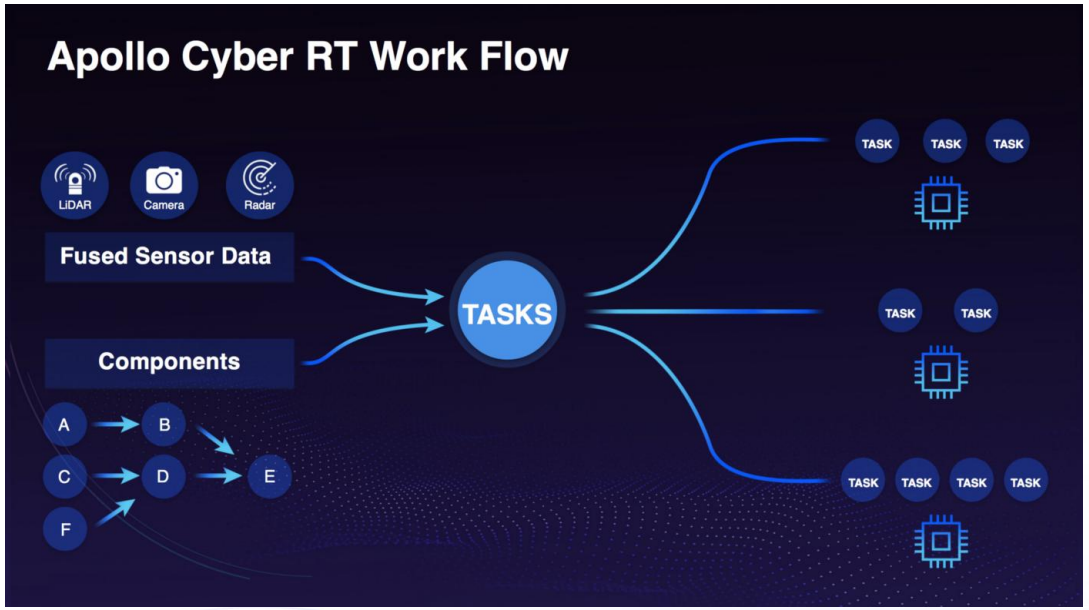


Fig 2: Cyber workflow [1]

  2. 继承Fast RTPS（eProsima Fast RTPS）：
     - 去中心化通讯RTPS 通讯：QoS, Participant / Node / Reader / Writer / Service / Client
     - 同主机通讯(进程内/见)： 传输层利用共享内存SHM，传递消息指针，实现符合 RTPS 模型
     - 设备间通讯媒介：Protobuf

CyberRT诞生背景

3. Real Time OS & CyberRT

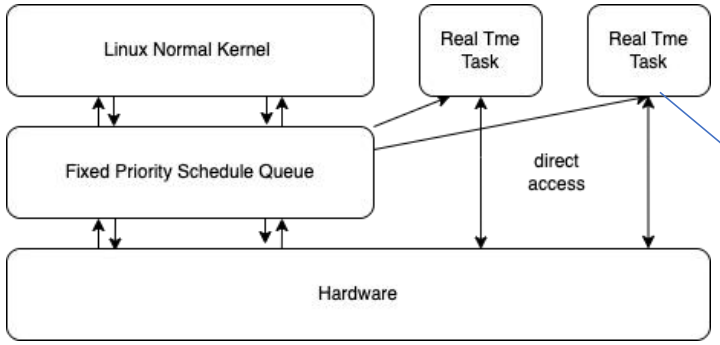


Fig 3 : RT-Linux

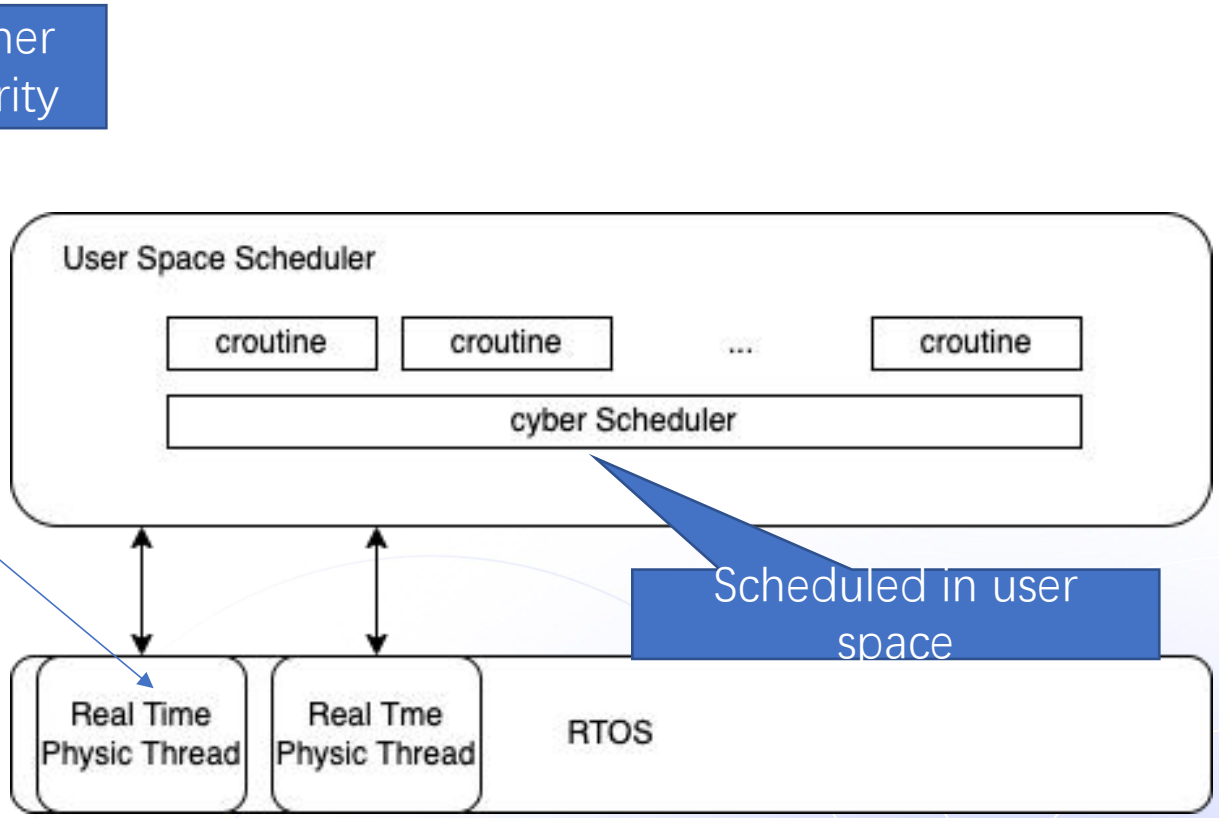
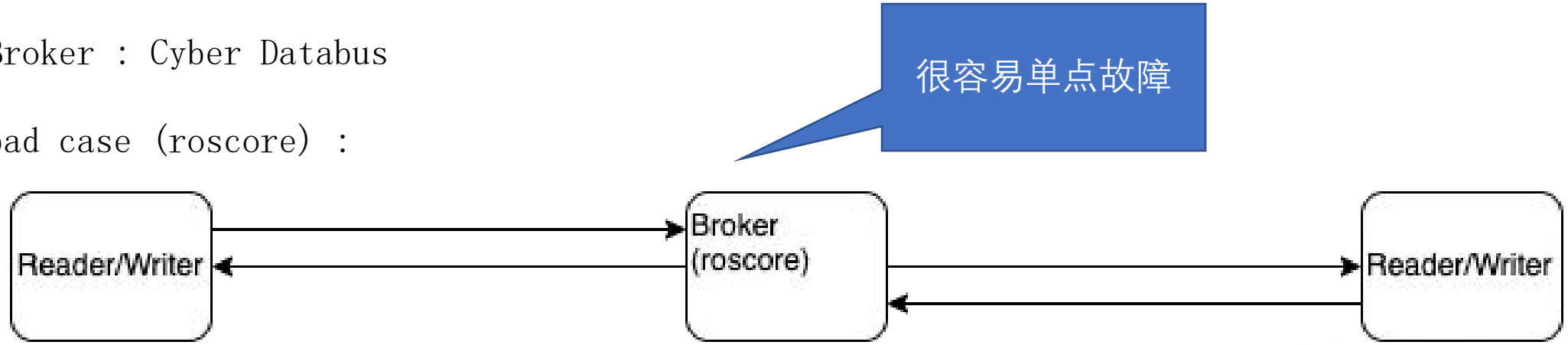


Fig 4 : Cyber in user space

CyberRT诞生背景

4. No Broker : Cyber Databus

A bad case (roscore) :

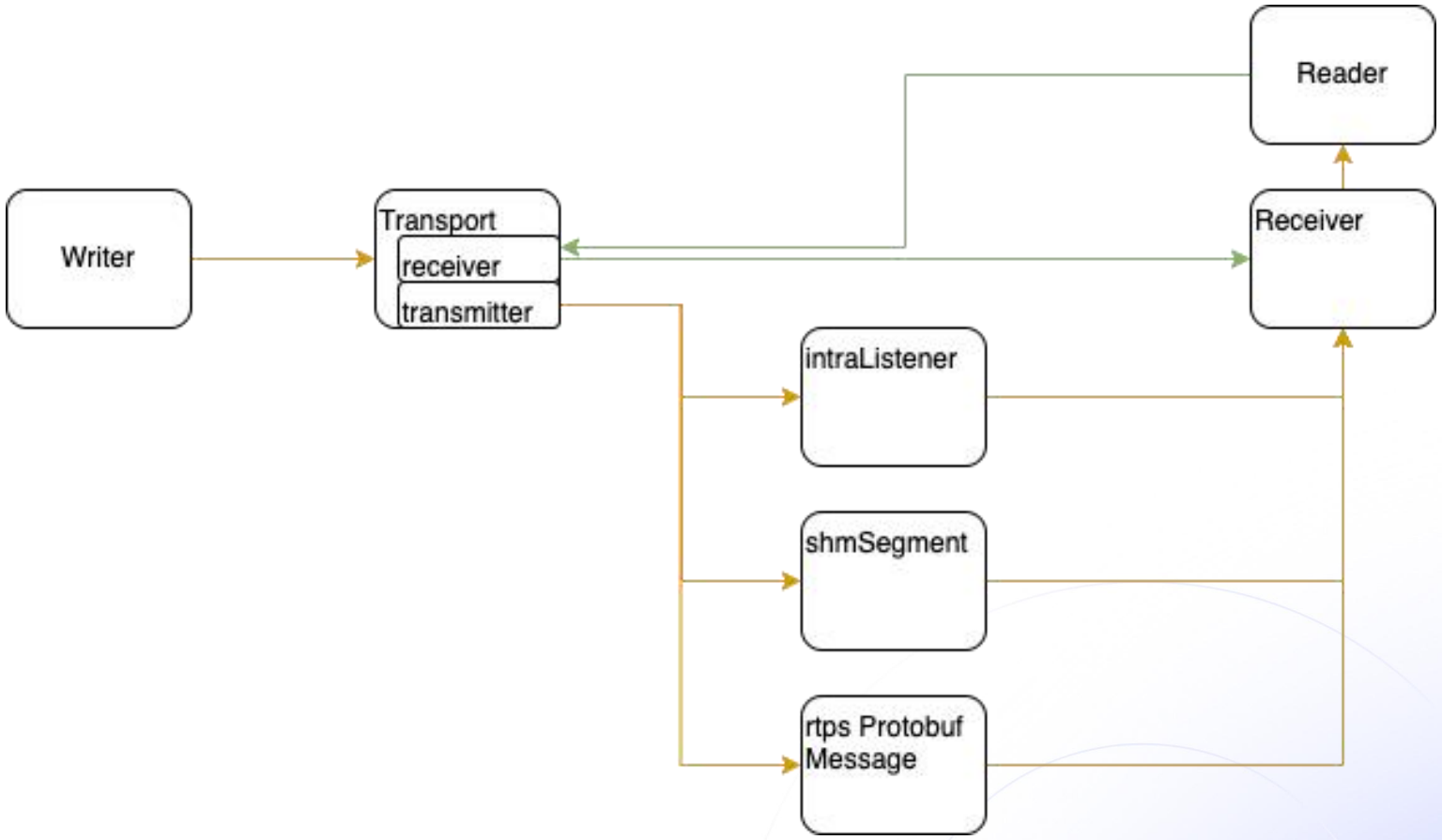


ROS1 : DCPS model implementation

希望Reader 和 Writer 不通过中间人，以databus的方式直接进行消息沟通

Cyber 引入Transport 来完成这个工作极大的提升了系统稳定性

CyberRT诞生背景



Cyber Reader/Writer communication implementation

CyberRT诞生背景

5. 实时性由QoS和实时线程保证

* 更快的通讯 ： 进程内点对点回调 (IntraTransimitter)， 进程间用ShmMemory

* 灵活出让Cpu时间 ： 协程 (用户态线程)

* 7.0 版本 ： 更大覆盖的无锁队列实现

本章小结

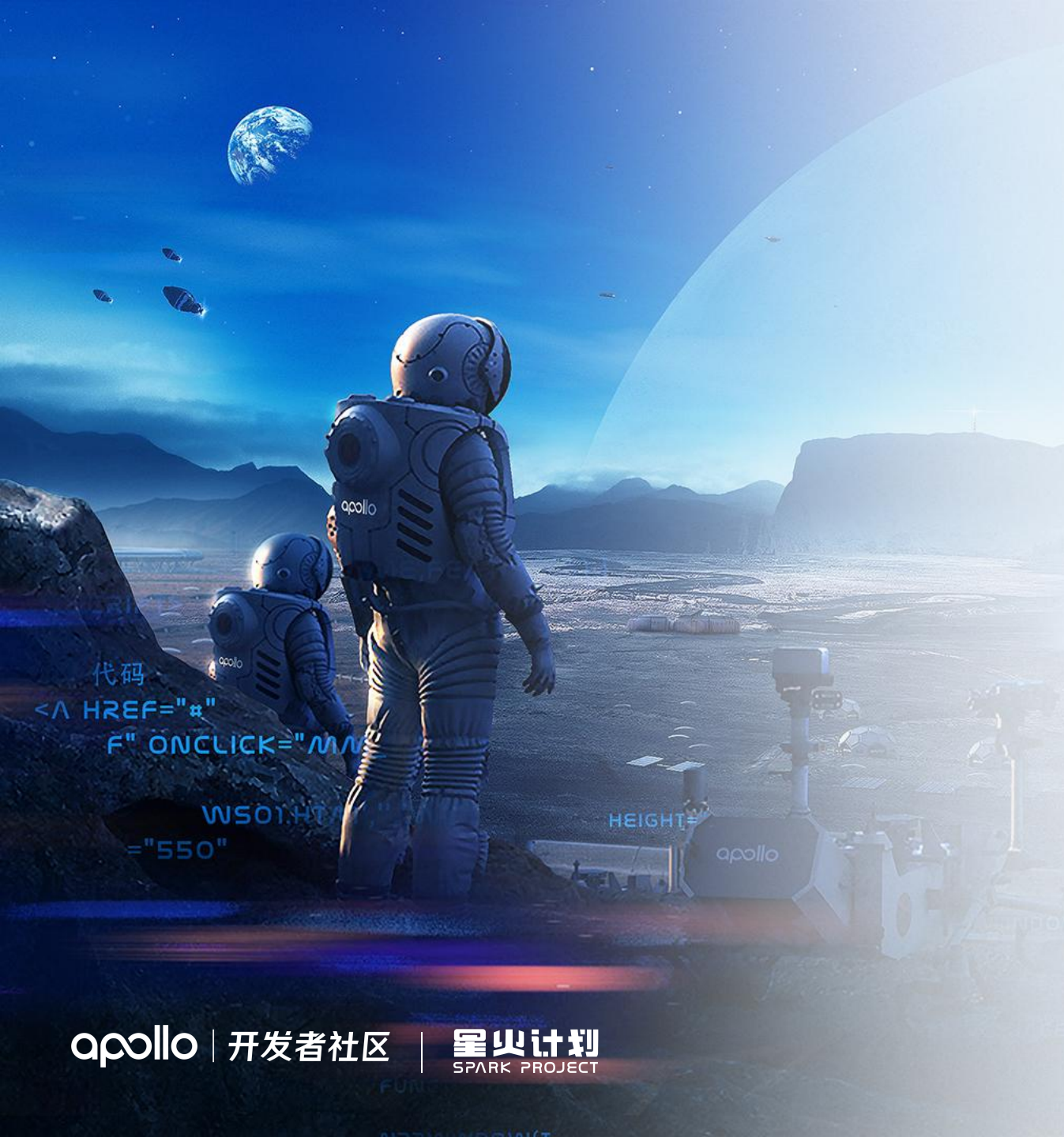
CyberRT 是为了解决时时任务而诞生的软件系统

* 建立了高效的去中心的数据通信通道
* 通过QoS，实时线程，以及用户态化保证实时性

CyberRT 针对自动驾驶任务特点进行了计算架构的优化

* 基于有向无环的任务调度
* 数据融合

CyberRT在Apollo扮演承上启下的角色，是整个数据流通的关键



CyberRT 在 apollo 框架的作用

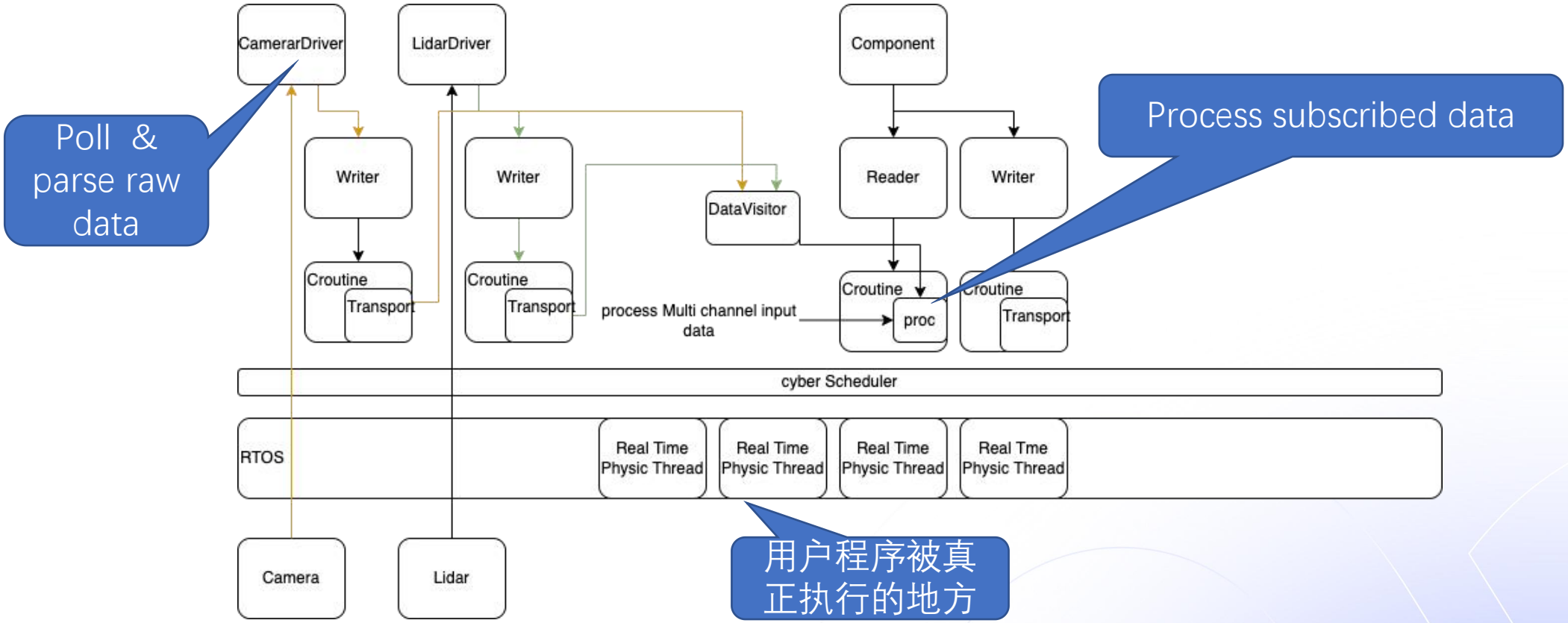


Fig 5 : Hower cyber works in apollo

CyberRT 在 apollo 框架的作用

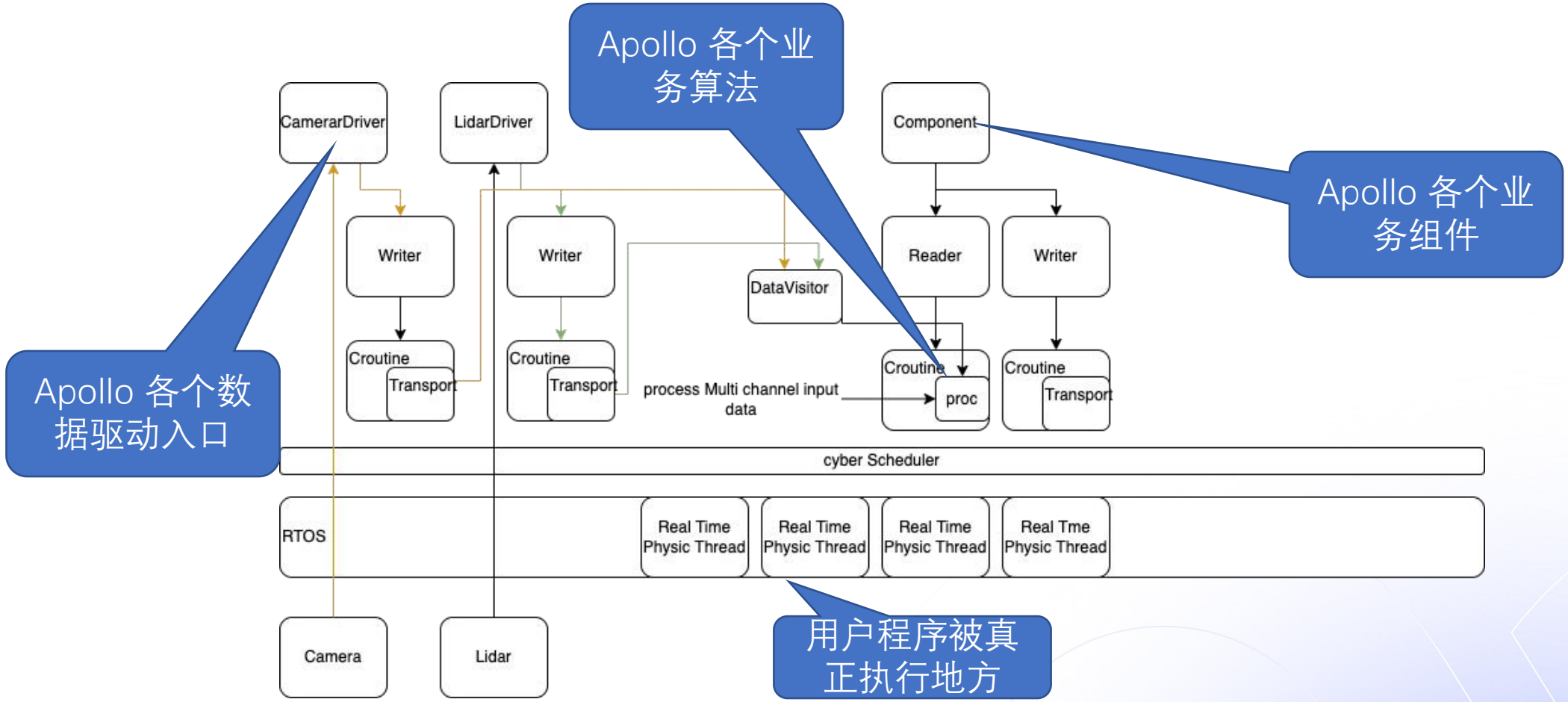
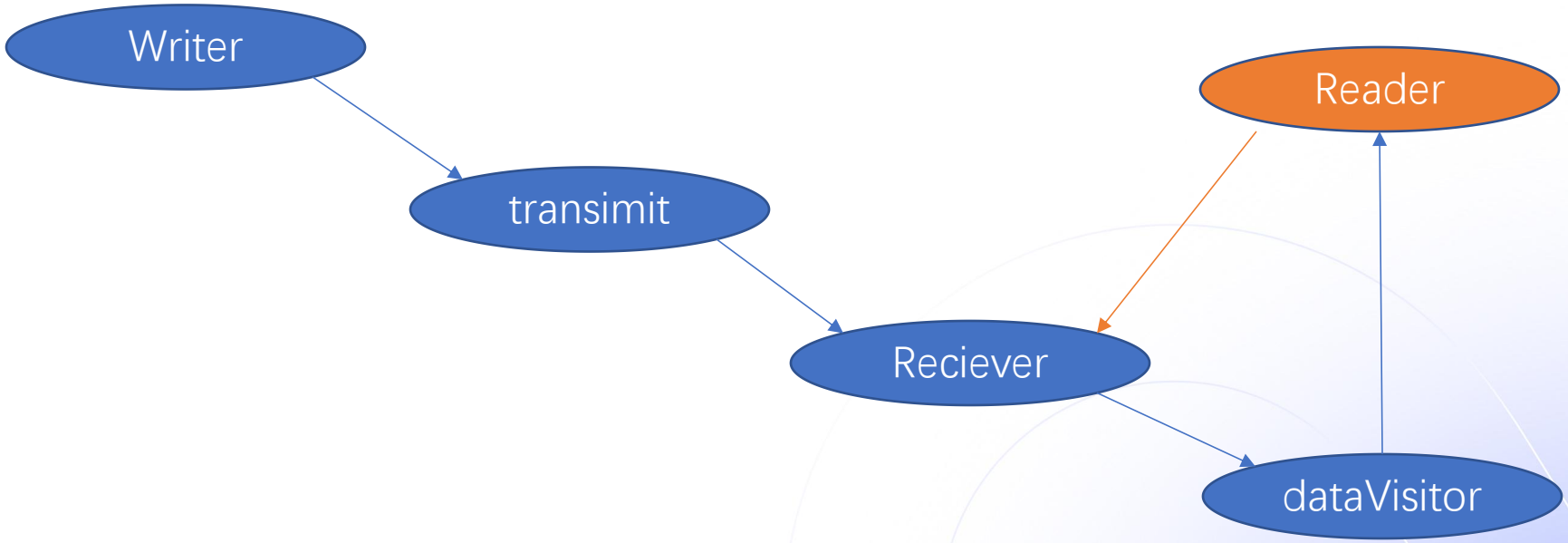
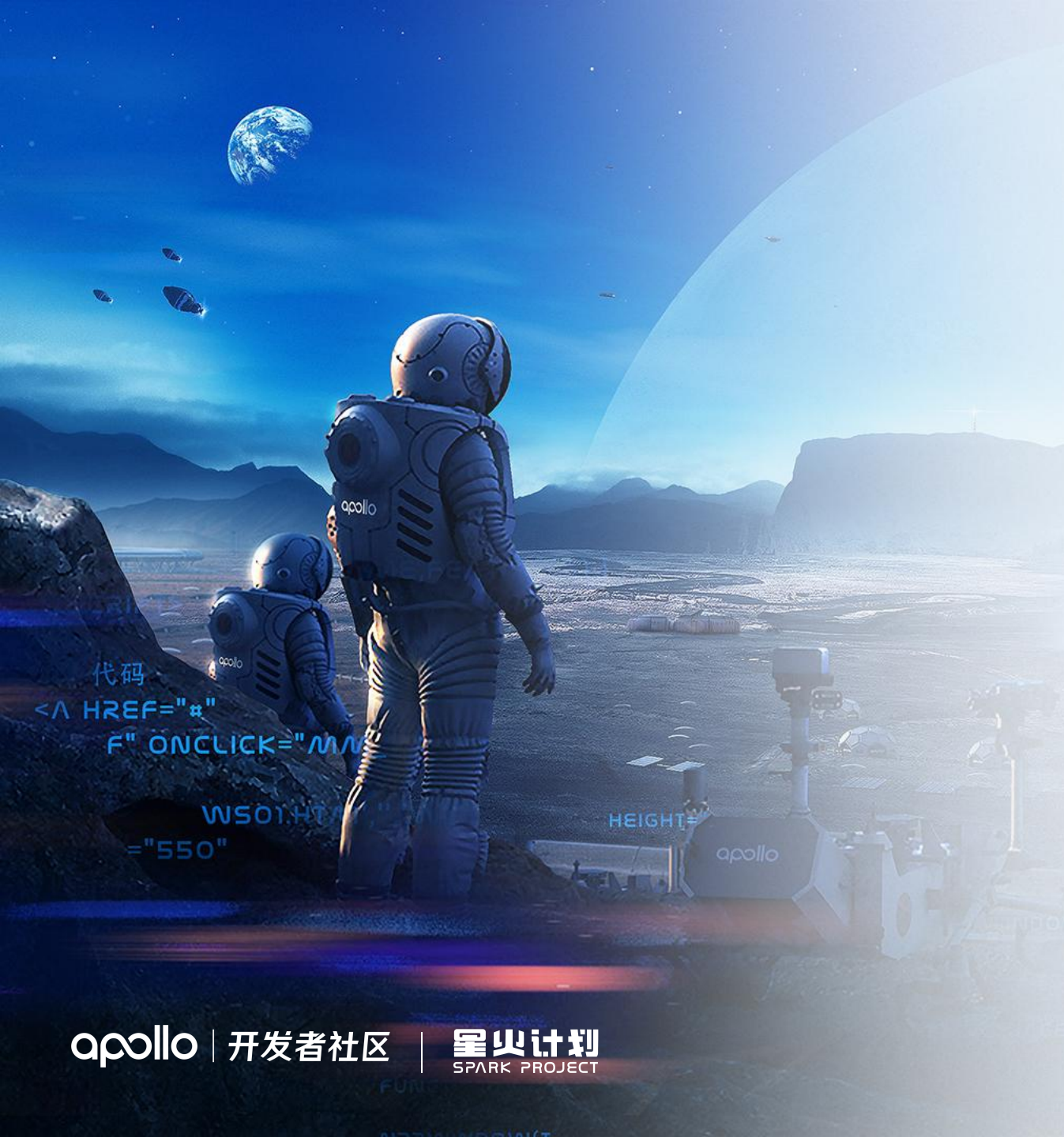


Fig 5 : Hower cyber works in apollo

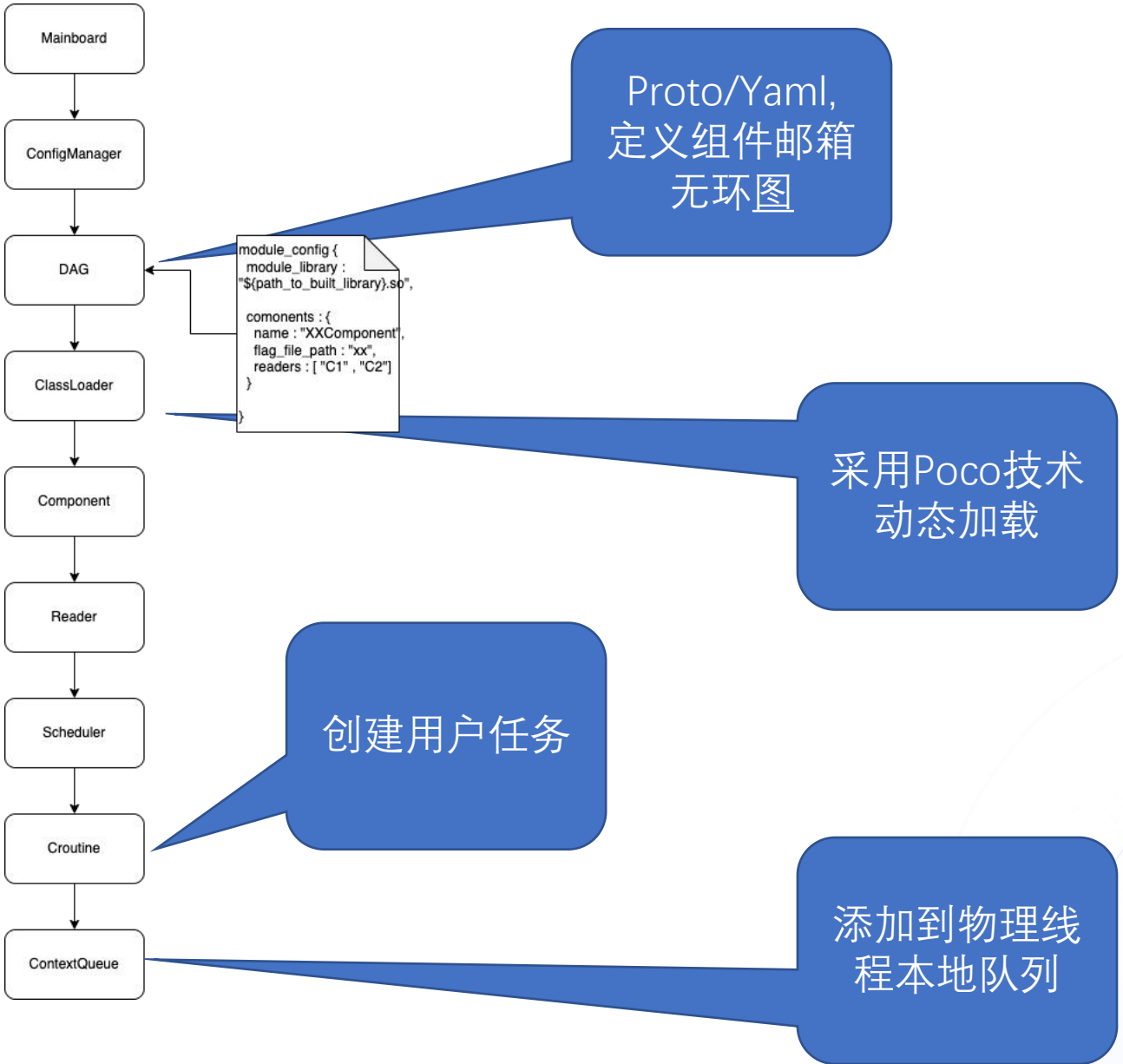
本章小结

CyberRT建立了的数据流通路径





CyberRT组件，通信机制



通过Component添加数据订阅/发布

- 默认会创建Reader，并定义QoS
- Writer 由用户创建
- 消息通过Datavisitor来同步，实现数据融合
- 注册服务

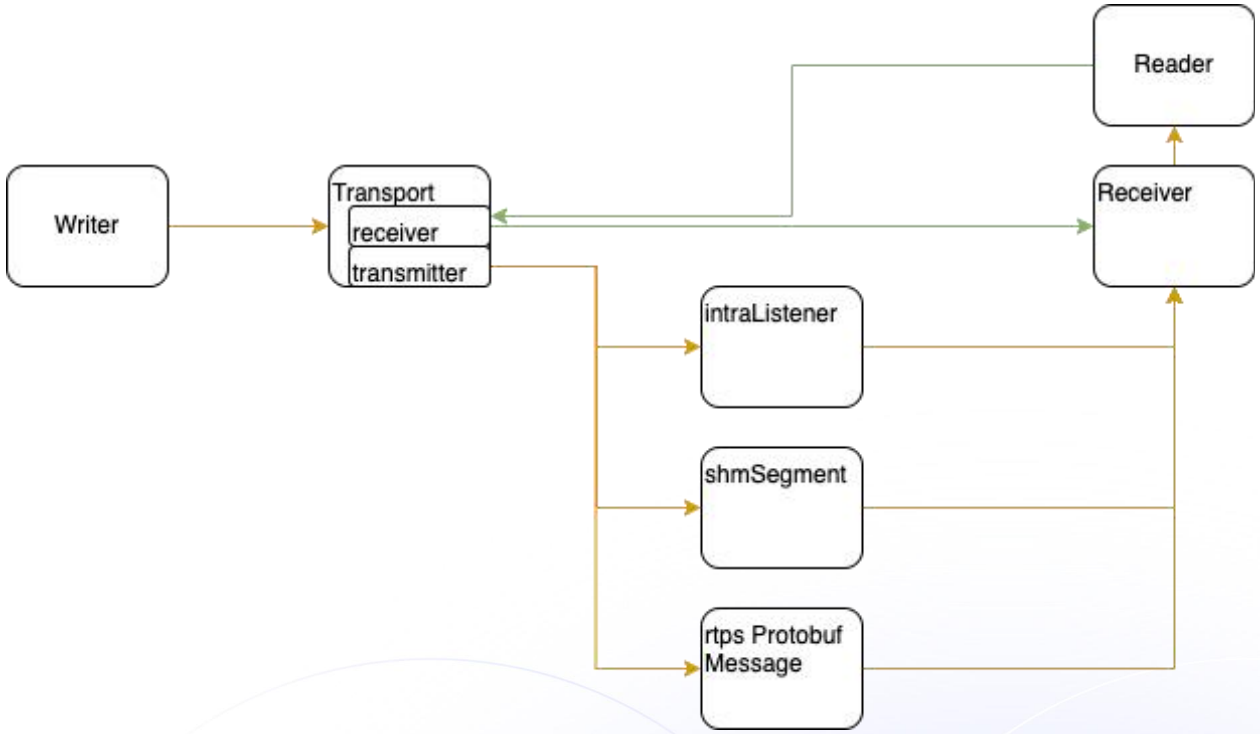
CyberRT组件，通信机制

Reader / Writer 通过Transport来通信：

1. 同进程内部，通过回调机制（数据消息裸指针）

2. 同主机，进程间，通过Share memory (ShmMemory)，并通过事件完成数据块共享（数据消息共享内存块指针）

3. 跨主机/域，采用UDP/TCP (fast-RTPS)进行消息传递



Cyber Reader/Writer communication implementation

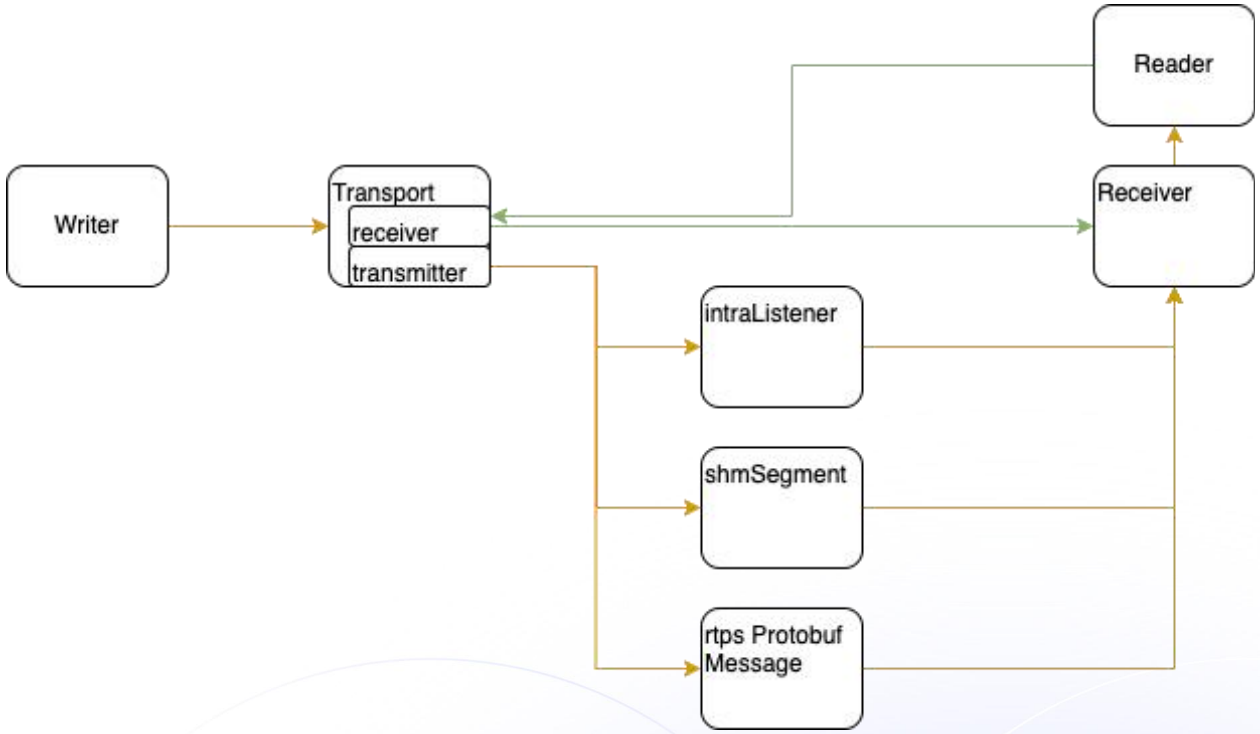
CyberRT组件，通信机制

Reader / Writer 通过Transport来通信：

1. 同进程内部，通过回调机制（数据消息裸指针）

2. 同主机，进程间，通过Share memory（ShmMemory），并通过事件完成数据块共享（数据消息共享内存块指针）

3. 跨主机/域，采用UDP/TCP（fast-RTPS）进行消息传递



Cyber Reader/Writer communication implementation

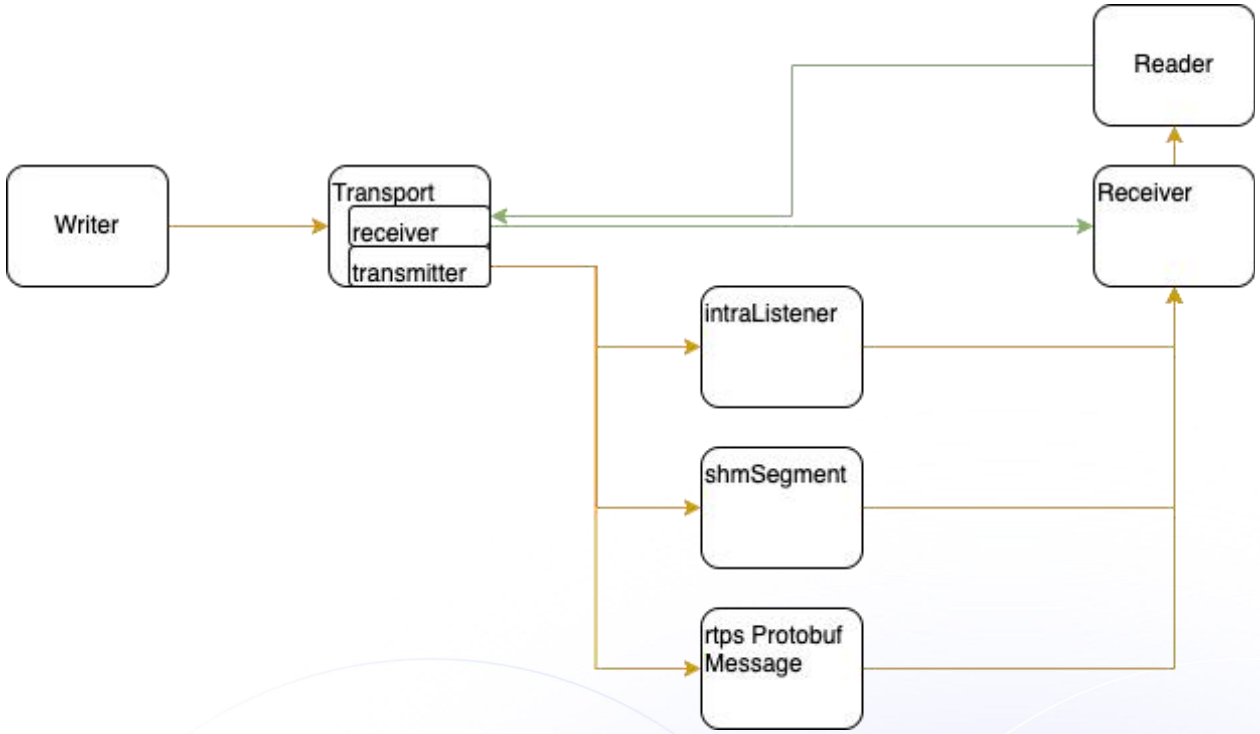
CyberRT组件，通信机制

Reader / Writer 通过Transport来通信：

1. 同进程内部，通过回调机制（数据消息裸指针）

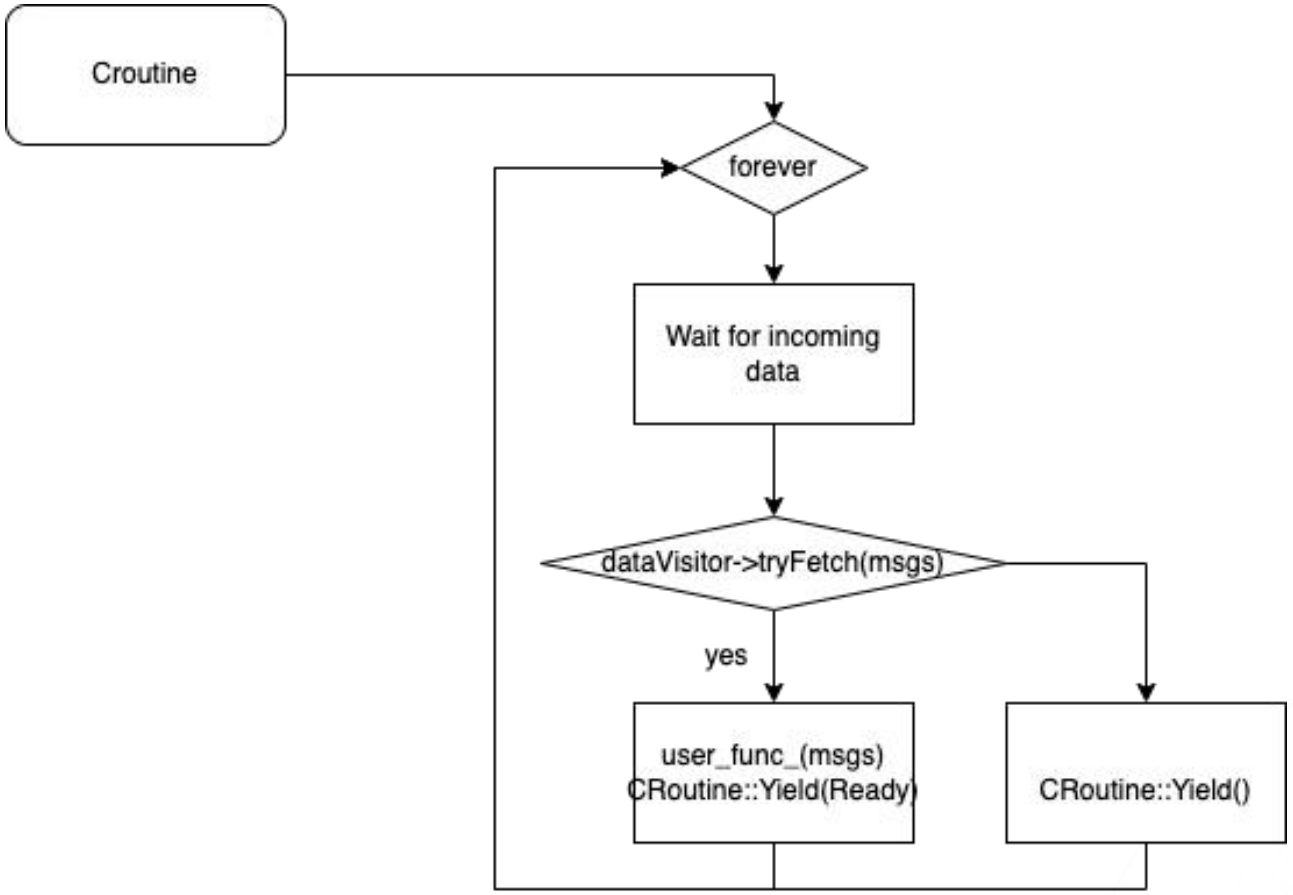
2. 同主机，进程间，通过Share memory (ShmMemory)，并通过事件完成数据块共享（数据消息共享内存块指针）

3. 跨主机/域，采用UDP/TCP (fast-RTPS)进行消息传递



Cyber Reader/Writer communication implementation

CyberRT组件，通信机制



Datavisitor

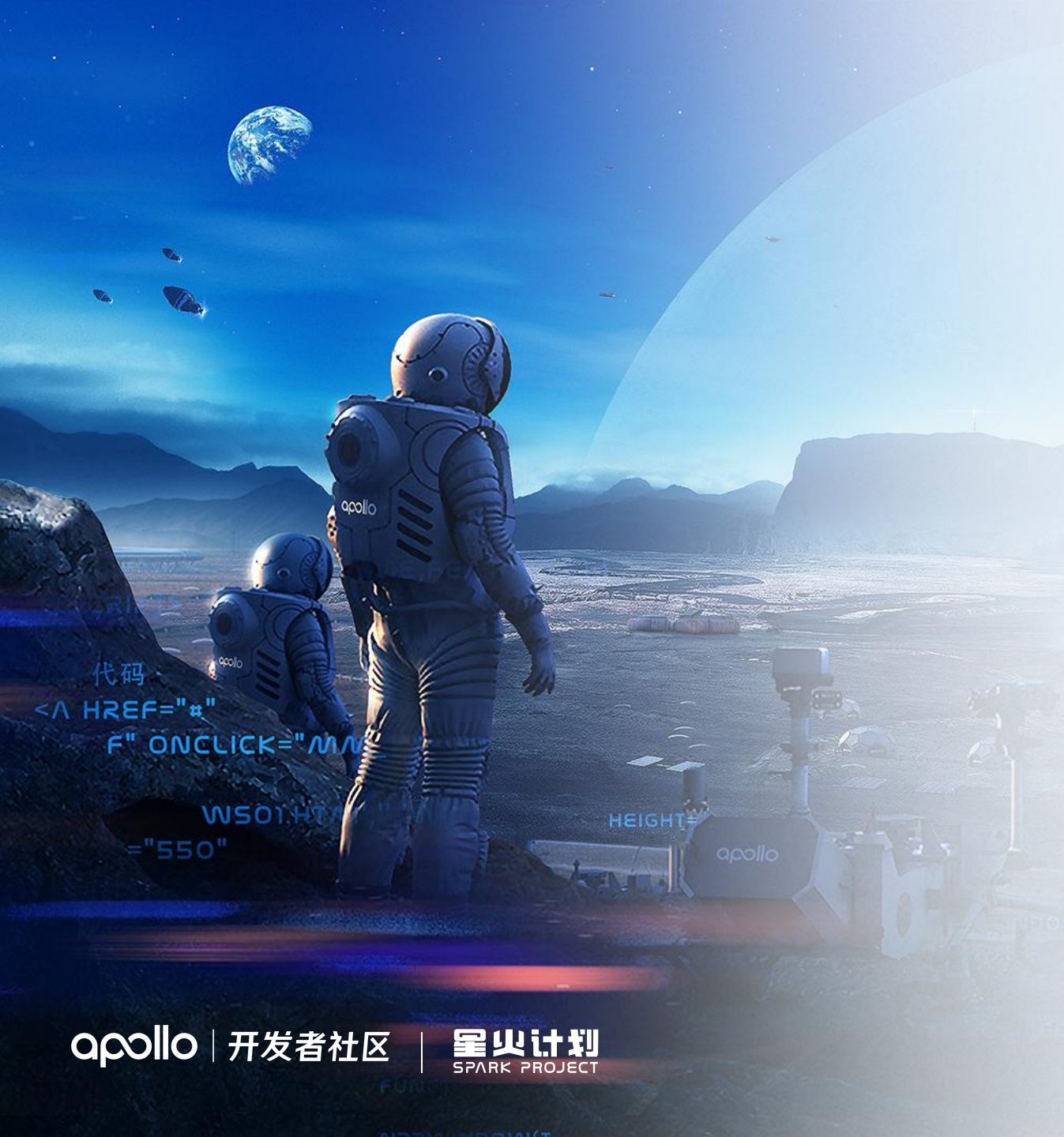
定义了数据buffer_m_i，和缓存，超过频控要求，即丢弃；

可以融合多路消息，对多路输入节点，只有当同一时刻附近，数据都Ready，时候才会触发用户回调，实现数据融合；

本章小节

Cyber 通过 mainboard 读取 dag文件创建Copomonent 中的Writer 和 Reader对象，分别进行数据发布和订阅，然后并将两者加入到通信拓扑图中，无中心节点依赖

数据通过Tansmitter从Writer流通到Reader，换存在Datavisitor中通过数据融合传递给业务函数（Proc）接口，实现数据流通和处理



实践案例

☐ 实验内容：

发布消息/订阅消息并进行简单的处理

☐ 实验目的：

了解自动驾驶研发流程

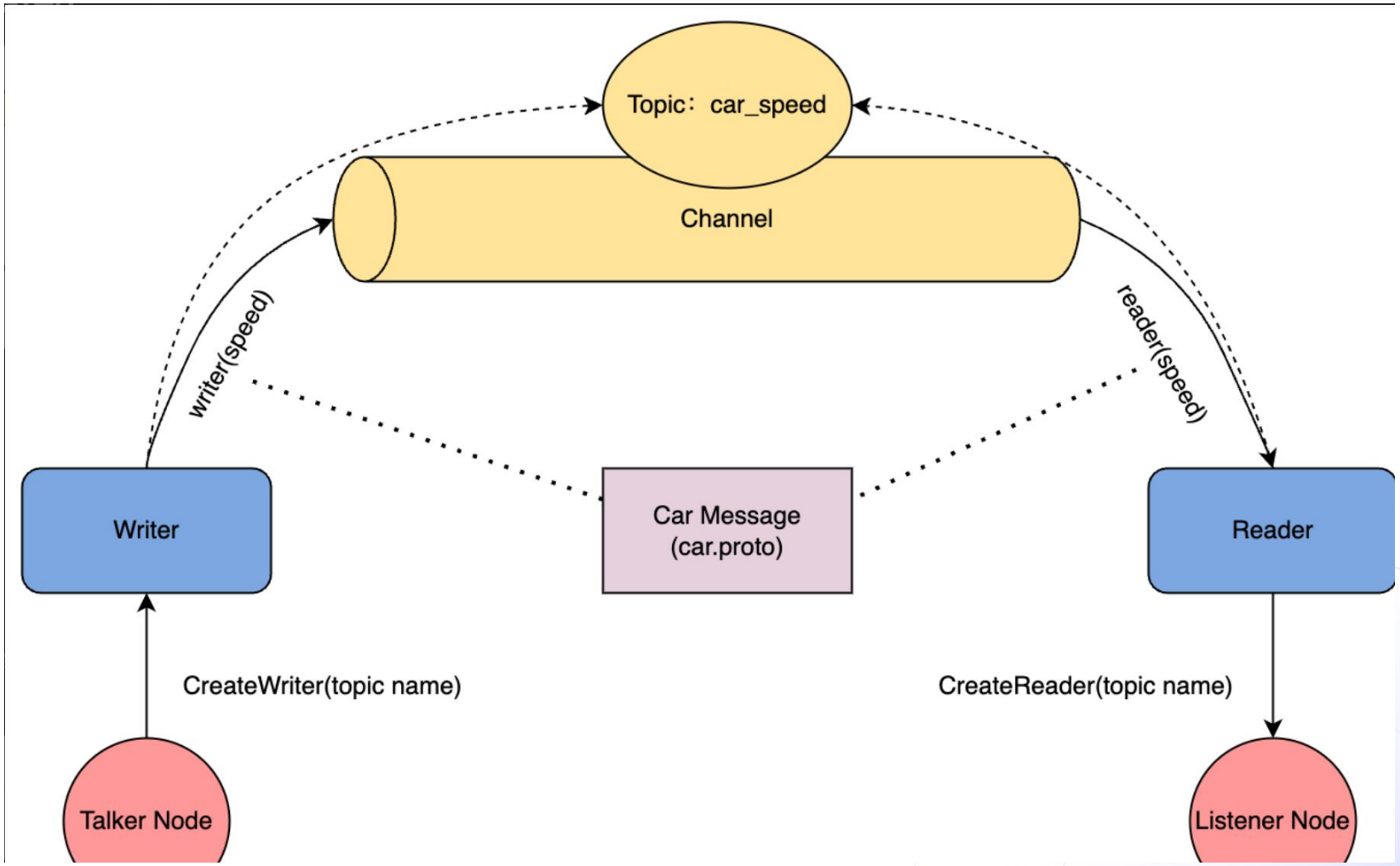
熟练掌握使用 Cyber 进行分布式应用开发

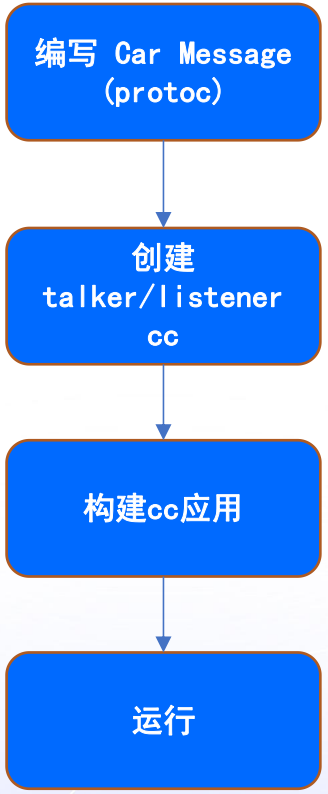
☐ 实验配置：

云端：Apollo Studio 云实验室　python 数据发布订阅实验项目

本地：Apollo 7.0 / Multi data distribution control (MDDC)

实验流程


Topic：car_speed
Channel
writer(speed)
reader(speed)
Writer
Car Message
(car.proto)
Reader
CreateWriter(topic name)
CreateReader(topic name)
Talker Node
Listener Node

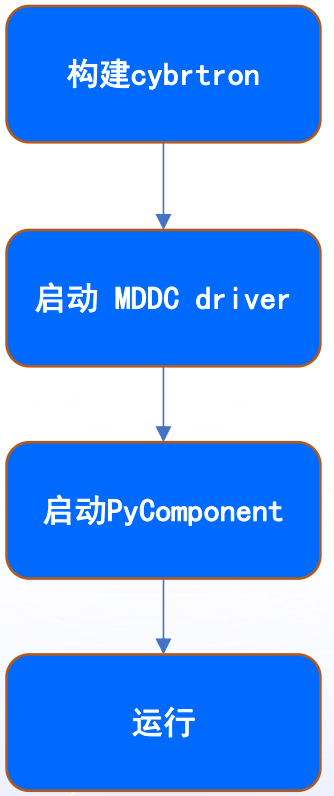

编写 Car Message
(protoc)
创建
talker/listener
cc
构建cc应用
运行

Py实验流程

Multi Data Distribution Control (MDDC) 实验

实验地址： https://github.com/yiakwy-mapping-team/cybertron

* zlib need to updated mannually (rules_proto)
* need use anaconda to downgrade Python 3.10 to Python 3.6 (**)
* move GCC so to anaconda python



本章小结

通过Cybertron我们学习和掌握了如何进行节点通讯，和任务串联

☛ 思考：有很多任务调度系统，比如Airbnb的 Airflow，如果Cybertron不在RTOS上跑，能否用来跑任务调度，相比纯python和纯java系统有什么收益？参考Dreamview和wasm，是否可以在Web上做一个任务调度看板？

参考文献

1. https://medium.com/@apollo.baidu/apollo-cyber-rt-the-runtime-framework-youve-been-waiting-for-70cfed04eade

问答环节

扫码报名，与更多开发者一起学习成长

Apollo开发者社区

Apollo开发者社区官网：
https://studio.apollo.auto/community

Apollo GitHub： https://github.com/ApolloAuto