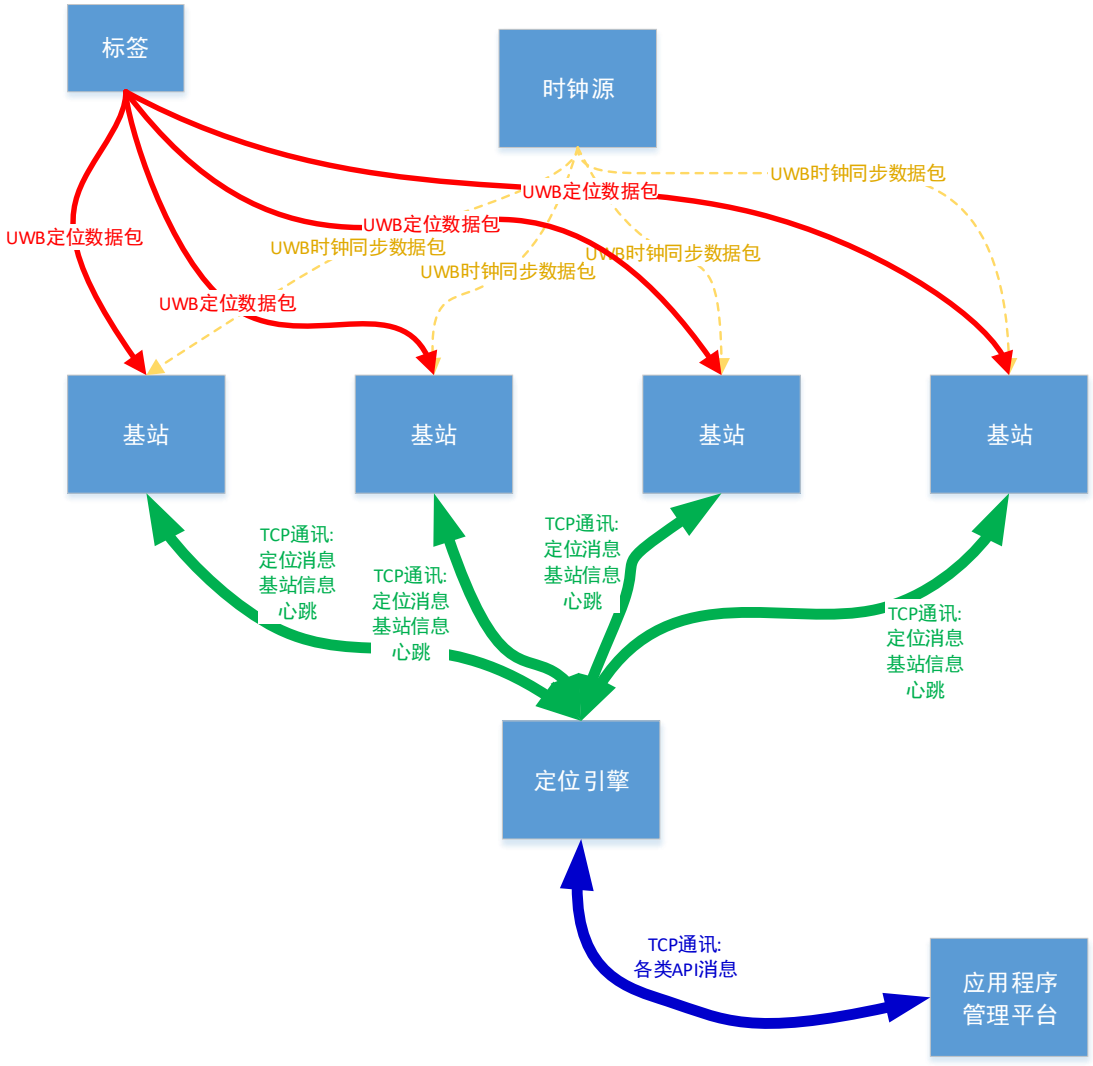
JLRTLS 数据流向与处理流程

1. 数据流向

在 JLRTLE 中涉及到的数据有几种：定位标签发给基站的 UWB 无线数据包、基站与定位引擎之间的 TCP 数据包、定位引擎与应用程序之间的 TCP 数据包，以及时钟源发给基站的 UWB 无线时钟同步数据包。



2. 处理流程

2.1. UWB 时钟同步

时钟源定期广播“**UWB 时钟同步包**”。时钟同步包的发送间隔可以在时钟源的配置中进行配置，通常设置为 200ms。如果间隔时间太长，可能会因为基站间的个体差异导致基站的时钟差异太大，从而使计算出的坐标误差变大；如果间隔时间太短，会导致时钟同步包占用频道的时钟太长，影响到定位数据包的通讯。

定位基站收到时钟源发出的时钟同步包后，会与时钟源进行时钟同步。时钟源和基站内部的时钟信号的来源是石英晶振，因为制造上原因，每一个晶振的频率会有细微的差异。对于普通的应用来说，这些细微的差异不会有多大影响，可以忽略。但是对于精确定位来说，这些差异是致命的，我们必须确保两点：各个基站都使用相同的时间体系、各个基站的时间差异不能太大。所以我们需要对定位基站的时钟定期与时钟源进行同步。

一个定位系统中会有多个定位区域，每个定位区域需要有一个时钟源，定位区域与时钟源之间是**一对一关系**。大部分的基站都服务于一个区域，它只需要与这个区域的时钟源同步就可以了。有些基站在区域的边界上，可能会被多个区域共用，对于这些基站，它们需要**与多个区域的时钟源同步**。

每个基站最多可以与 5 个时钟源同步，如果在基站配置中没有指定同步的时钟源，它会与最近收到的 5 个不同来源的时钟同步数据包的发送者同步。如果在这个基站的附近有超过 5 个时钟源，可能会使它与这些时钟源轮流同步，这会使它没有稳定的同步对象，导致无法取得准确的时间戳。对于定位区域比较多的系统，应该在基站配置中指定要同步的时钟!

2.2. UWB 定位数据包的发送与接收

定位标签定期发出 UWB 无线定位数据包。在标签附近的基站收到这个数据包后，会记录下收到这个数据包的时间戳，然后把这个数据包的内容以及收到的时间，通过 TCP 网络发送给定位引擎。

UWB 定位数据包中，包含有两个重要的信息：(1)、**标签的 ID**，这是一个 EUI64 地址，它由 64 个二进制位构成。标签的 Id 在整个系统中是唯一的，在出厂时固化在标签硬件中。(2)、**数据包序号 Seq64**，这是一个 64 位整数，即 Int64。标签不断发出的数据包中，这个序号是递增的。

2.3. 基站与定位引擎的交互

基站的主要任务是接收标签发出的 UWB 定位数据包，然后把收到的 UWB 定位数据重新组装后，通过 TCP 连接发送给定位引擎。

标签是以广播的方式在空中发送 UWB 定位数据包，在覆盖范围内的基站会收到这些 UWB 定位数据包。距离太远的基站收不到，我们会认为这个标签超出了基站的覆盖范围。

基站与定位引擎之间的数据交互主要以 TCP 连接的方式进行。定位引擎会在 TCP 端口 1200 上侦听。如果基站没有指定 RTLE 地址，基站会以 UDP 方式在局域网内广播发送“RTLE 发现包”，如果在局域网中存在定位引擎，定位引擎会以 UDP 方式广播“RTLE 发现应答包”，通知基站自己的 IP 地址。基站收到“RTLE 发现应答包”后，会自动向这个数据包中指定的 IP 地址发起 TCP 连接。

基站收到 UWB 定位数据包后，会把接收到该 UWB 定位数据包的**时间戳**、**基站的 ID**、对应的**时钟源的 ID**，以及该定位数据包的内容(**标签 Id**、**数据包序号**)等，组装成一个“测距消息”，把这个测距消息发送给定位引擎。

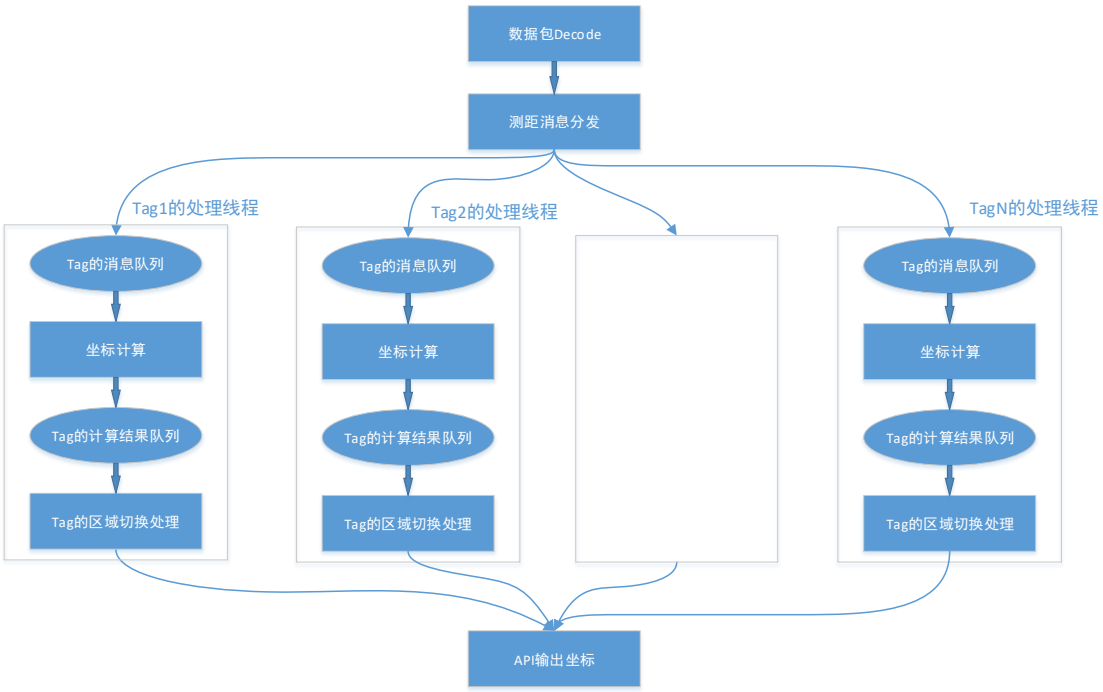
注意：
因为基站可能会与多个时钟源同步，对于某一个标签发出的某一个序号的定位数据包，这个基站会每一个同步中的时钟源生成对应的“测距消息”。例如这个基站与 4 个时钟源同步，则会生成 4 条“测距消息”，分别带有 4 个不同的时钟源 Id、对应的 4 个时间戳。

定位引擎收到基站发送的任何消息后，都会回复一个 NET_MESSAGE_ALIVE 消息，这个回复主要供基站确认 TCP 连接工作正常。

其实，基站并不太关心具体的某一条消息是否到达定位引擎，它只在宏观层面关心能不能正常把数据发送到定位引擎。定位引擎回复的心跳消息，其实只是让基站知道 TCP 连接工作正常。如果基站长时间(10 秒)没有收到定位引擎发来的心跳消息，基站认为 TCP 连接工作不正常，可能会是因为某些原因使连接看上去像是正常，但是实际上不正常。它会主动断开与定位引擎的 TCP 连接，再重新连接定位引擎。
基站一旦与定位引擎建立 TCP 连接，基站至少保证 2 秒会发送一个数据包给定位引擎。如果有正常的业务数据包(例如测距消息)，则发送它；如果没有正常的业务数据，最后一次数据发送过了 2 秒，基站会发送一个心跳消息给定位引擎。

3. 定位引擎的内部处理流程

定位引擎的任务是计算标签的坐标。



我们的关注点是测距消息。

- 定位引擎收到测距消息后，先进行 decode，生成一个 Java 内部的测距消息对象。
- 测距消息会被分发到每个标签对应的消息队列。系统中的每一块标签都会有一个线程负责处理与其相关的事务。对于新标签，系统会为其创建一个新的线程。
- 标签处理线程会从该标签的消息队列中取出测距消息，进行坐标计算。坐标计算是以区域为单位进行的，每个区域单独计算，计算的条件是：**某区域下的基站发来的该标签的某 seq 的消息的数量达到区域配置中要求的消息数量**。例如系统中存在区域 A(时钟源为 CSA)、B(时钟为 CSB), 区域 A 下有基站 A1/A2/A3/A4, 区域 B 下有基站 B1/B2/B3/B4, 标签为 T1，区域的计算条件都配置为最小 4 条消息。假设现在标签 T1 发出一个 seq 为 123 的定位数据包。那么，当标签 T1 的消息队列中存在 T1-A1-CSA-SEQ123、T1-A2-CSA-SEQ123、T1-A3-CSA-SEQ123、T1-A4-CSA-SEQ123 这 4 条消息时，A 区域的坐标计算条件满足。以为上的计算条件，如果有一条不满足，都不会进行坐标计算。
- 坐标计算出来会，这些坐标会放到一个**计算结果队列**中。
- 有一个线程会对计算结果队列进行分析，当某一个 SEQ 的最后一个坐标计算出来的时间与当前时间的差超过某个值的时候，对该 SEQ 进行区域切换处理。这个差值的目的是等一些有消息晚到的区域有机会计算出坐标。因为有可能会有多个区域都能正常接收到标签的信号，都能对标签的坐标进行计算。例如，有 3 个区域都能收到标签的坐标，系统收到的消息有早有晚，总之，我们可能会收到 3 组消息，计算出 3 个坐标。坐标计算出来后，延时 20ms，通常在局域网内，第 2 个、第 3 个坐标不会晚那么久到达。
- 当区域切换的条件满足后，对该 SEQ 的坐标进行投票。投票模块会计算各种权重因子，为几个坐标计算出一个分值，得分高的坐标胜出，被选中。
- 被选中的坐标输出到 API。