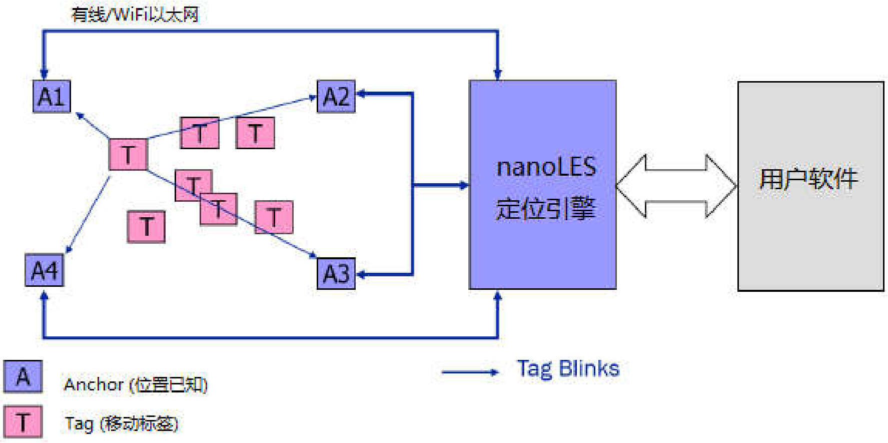
# Chirp UWB 系统 Socket 接口 V1.1

LES 软件采用 TXT 文本通讯、BIN 二进制通讯方式，与客户自主设计的外部软件进行通讯。下文针对本引擎软件所采用的 Socket 通讯协议进行描述。

## 系统架构

在一个典型的系统构架中，定位基站 A1~A4 均连接在以太网上，定位标签所发出的定位数据包将被 A1~A4 收到，由此产生能够参与后续计算的定位原始数据(Raw Data)。这些 Raw Data 将通过 UDP 通讯协议，被发送到定位引擎软件，进行位置计算。



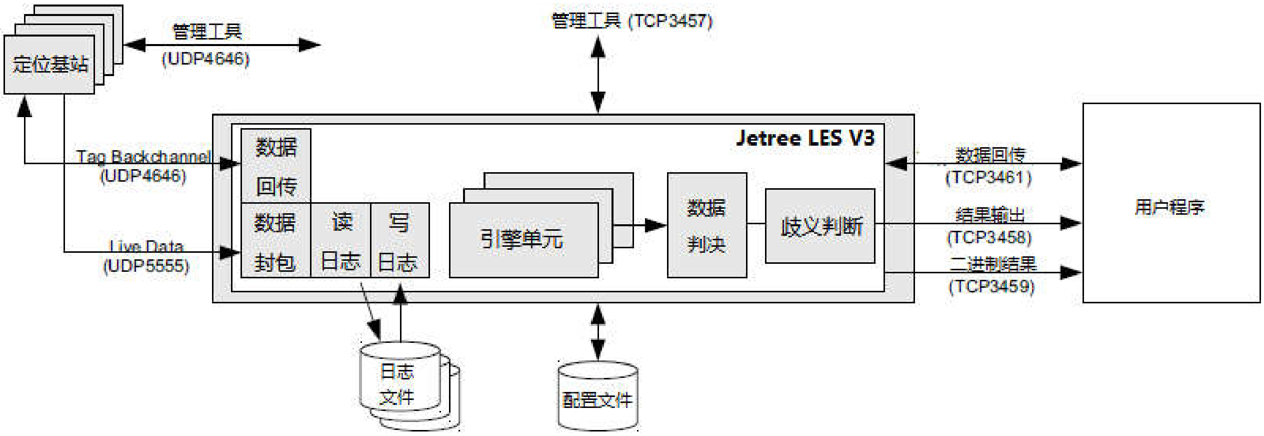
一般情况下，定位引擎需要管理，包括预先配置参数以及获取系统运行状态：

A) 引擎算法参数

B) 基站参数及坐标系

C) 控制并获取引擎运行状态

通讯接口示意图：



以下内容假设：

1) 运行引擎软件 LES 的计算机 IP 地址为 192.168.1.30，
   基站地址设置为 192.168.1.170, 192.168.1.171，基站数据投送目标地址（Server）均设置为 192.168.1.30:5555

2) 以 VC++开发软件举例，**发送数据方式为：**
   m_MClientSocket.Send(Buff,StrLen);　　其中 Buff 为即将发送的数据缓冲，StrLen 为发送数据长度；
   **接收数据方式为：**
   m_MClientCmdComfirmed = CheckCmdConfirmed();　其中 m_MClientCmdComfirmed 为状态标志位；

3) 这里的描述不管用户是否使用诸如多线程、其它高级语言、其它高级设计框架；请客户考虑软件的稳定性；

## 参数管理接口

管理软件的作用为管理定位基站设备参数、引擎参数，并控制系统运行、停止；运行流程为：

| | | | | | |
|---|---|---|---|---|---|
| 启动并创建Socket | → | 获取引擎状态 | → | 设置系列端口与参数 | → |
| 设置基站IP与坐标 | → | 运行引擎 | → | 关闭引擎(如有需要) | → |

(1) 连接引擎并获取状态：默认为127.0.0.1:3457 或者192.168.1.30:3457

MClientSocket.Connect(“127.0.0.1”, 3457);
Buff = “get status\r\n”
m_MClientSocket.Send(Buff, StrLen);

请确认返回值是否为：R:stop 此状态为停止运行状态,
如果返回为：R:run，则处于运行状态，需要先停止引擎后再配置:
*Buff = “stop\r\n”*
*m_MClientSocket.Send(Buff, StrLen);*
*返回: R:0*

(2) 一般情况下，引擎参数通过.ini文件进行默认配置。也可以在线设置一系列端口，包括输出定位结果端口uiPort、管理工具端口clientPort,等等; 然后配置引擎运行参数；

A) 配置定位结果输出端口: Buff = “set option uiPort "3456"\r\n”
B) 配置管理工具连接端口: Buff = “set option clientPort "3457"\r\n”
C) 配置接收基站信息端口: Buff = “set option anchorPort "5555"\r\n”
D) 设置引擎参数：几维定位？ 选择范围: 2/3
Buff = “set option nDimensions "2"\r\n”
E) 设置最小定位基站数： 选择范围: 2/3/4
Buff = “set option minContributingAnchors "2"\r\n”
F) 设置位置滤波器算法是否启用: 选择范围 false/true
Buff = “set option posFilterEnabled "true"\r\n”
G) 设置OFFSET补偿算法是否启用: 选择范围 false/true
Buff = “set option offsetCompensationEnabled "false"\r\n”
H) 设置位置滤波器算法是否启用: 选择范围 false/true
Buff = “set option useMpComp "true"\r\n”

还有一系列的参数一般情况下不需要配置，但是需要用户的软件在必要时候支持配置；

(3) 设置基站坐标信息

A) 清除原来的坐标信息，依据客户的软件架构来确定，比如，也可以设计为查询Anchor，列表中没有再添加，而不是先清除再重新写入；

Buff = “clear anchor all\r\n” m_MClientSocket.Send(Buff, StrLen);

B) 设置基站的坐标、IP地址、站名等；其中关于基站的ID号，引擎采用短号方式，即只考虑基站最后两个字节；如对MAC地址为：180B52000D53，取最后两个字节0x0D53，转换成10进制3411；

Buff = “set anchor 3411 "Anchor000D53" "192.168.1.170" 0.8 0.3 2.0\r\n”
m_MClientSocket.Send(Buff, StrLen);

再设置MAC地址为180B5200117C基站的信息，其中0.8 0.3 2.0以及10.8 0.3 2.0为两台基站的坐标；

Buff = “set anchor 4476 "Anchor00117C" "192.168.1.171" 10.8 0.3 2.0\r\n”

C) 清除所有tag缓存信息；

```
Buff = “clear tag all\r\n”            m_MClientSocket.Send(Buff, StrLen);
```

(4) 运行引擎

```
Buff = “start\r\n”                    m_MClientSocket.Send(Buff, StrLen);
```

(5) 停止引擎

```
Buff = “stop\r\n”                     m_MClientSocket.Send(Buff, StrLen);
```

## 位置数据接口

根据上面配制，客户的可视化软件”Visual Platform”采用 uiPort 端口号为 3458，命令接口步骤如下：

1. Visual Platform 新建 TCP Client 方式连接 LES 引擎端口 3458 后，LES 首先发送以下识别字符，共 39 字节；

```
nanoLES,SLMF,1.0,1.0,Jetree Rev 8663\r\n
```

2. Visual Platform 发送指令获取 Anchors 坐标，共 10 个字节，坐标数据可以用于标示地图上的设备位置，也可以忽略；

```
getanchors
```

3. LES 首先对 Visual Platform 回复定义信息，共计 871 个字节，以下数据每行后加入\r\n。

```
FieldDefinition,Name=Tag_id,Type=HexBinary
FieldDefinition,Name=Tag_Id_Format,Type=HexBinary
FieldDefinition,Name=X,Type=Double
FieldDefinition,Name=Y,Type=Double
FieldDefinition,Name=Z,Type=Double
FieldDefinition,Name=Battery,Type=HexBinary
FieldDefinition,Name=Timestamp,Type=DateTime
FieldDefinition,Name=AnchorName,Type=String
FieldDefinition,Name=IpAddressV4,Type=String
FieldDefinition,Name=BlinkId,Type=Integer
FieldDefinition,Name=QualityIndicator,Type=Integer
FieldDefinition,Name=Payload,Type=HexBinary


MessageDefinition,Source= nanoLES,Format=A,Tag_Id,Tag_Id_Format,X,Y,Z,Battery,Timestamp,AnchorName,IpAddressV4
MessageDefinition,Source= nanoLES,Format=T,Tag_Id,Tag_Id_Format,X,Y,Z,Battery,Timestamp,BlinkId,QualityIndicator
MessageDefinition,Source= nanoLES,Format=TP,Tag_Id,Tag_Id_Format,X,Y,Z,Battery,Timestamp,BlinkId,QualityIndicator,Payload
```

4. LES 发送 Anchor 的坐标数据到 Visual Platform，例如下述 3 台 Anchor 数据：

```
ack 0x0A    本行共 4 字节
nanoLES,A,00000d53,00,0.80,0.30,2.00,64,2015-01-13T14:02:10,Anchor000D53,192.168.1.170
nanoLES,A,0000117c,00,10.80,0.30,2.00,64,2015-01-13T14:02:10,Anchor00117C,192.168.1.171
```

5. Visual Platform 源源不断接收到 LES 提供的 ASIC 格式数据如下：

```
nanoLES,T,032a2559,00,-0.17,1.92,0.00,inf,2020-12-28T02:47:23.310,79,1.44218,1,new-section,-101.625
nanoLES,T,032a2559,00,-0.17,1.92,0.00,inf,2020-12-28T02:47:23.310,7A,1.44218,1,new-section,-101.625
nanoLES,T,032a2559,00,-0.17,1.92,0.00,inf,2020-12-28T02:47:23.310,7B,1.44218,1,new-section,-101.625
nanoLES,T,032a2559,00,-0.17,1.92,0.00,inf,2020-12-28T02:47:23.310,7C,1.44218,1,new-section,-101.625
```

若采用另一种自定义模式数据，则格式为：

```
102981727,1320256754.312,4.820563,1.692251,0.000000,rQWtfQAA
102981727,1320256754.562,4.876924,1.926067,0.000000,rgWufQAA
102981727,1320256754.812,4.930466,2.034810,0.000000,rwWvfQAA
102981727,1320256755.062,4.982050,2.136542,0.000000,sAWwfQAA
```

6. Visual Platform 所接收到的连续数据格式如下：

```
nanoLES,SLMF,1.0,1.0,Jetree Rev 8663
FieldDefinition,Name=Tag_id,Type=HexBinary
FieldDefinition,Name=Tag_Id_Format,Type=HexBinary
FieldDefinition,Name=X,Type=Double
FieldDefinition,Name=Y,Type=Double
FieldDefinition,Name=Z,Type=Double
FieldDefinition,Name=Battery,Type=HexBinary
FieldDefinition,Name=Timestamp,Type=DateTime
FieldDefinition,Name=AnchorName,Type=String
FieldDefinition,Name=IpAddressV4,Type=String
FieldDefinition,Name=BlinkId,Type=Integer
FieldDefinition,Name=QualityIndicator,Type=Integer
FieldDefinition,Name=Payload,Type=HexBinary
MessageDefinition,Source= nanoLES,Format=A,Tag_Id,Tag_Id_Format,X,Y,Z,Battery,Timestamp,AnchorName,IpAddressV4
MessageDefinition,Source= nanoLES,Format=T,Tag_Id,Tag_Id_Format,X,Y,Z,Battery,Timestamp,BlinkId,QualityIndicator
MessageDefinition,Source= nanoLES,Format=TP,Tag_Id,Tag_Id_Format,X,Y,Z,Battery,Timestamp,BlinkId,QualityIndicator,Payload
nanoLES,T,032a2559,00,-0.17,1.92,0.00,inf,2020-12-28T02:47:23.310,79,1.44218,1,new-section,-101.625
nanoLES,T,032a2559,00,-0.17,1.92,0.00,inf,2020-12-28T02:47:23.310,7A,1.44218,1,new-section,-101.625
nanoLES,T,032a2559,00,-0.17,1.92,0.00,inf,2020-12-28T02:47:23.310,7B,1.44218,1,new-section,-101.625
nanoLES,T,032a2559,00,-0.17,1.92,0.00,inf,2020-12-28T02:47:23.310,7C,1.44218,1,new-section,-101.625
```

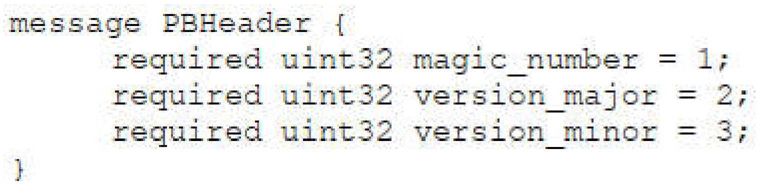
上述协议中，位置数据的解析格式如下：

nanoLES,T,032a2559,00,-0.17,1.92,0.00,inf,2020-12-28T02:47:23.310,79,1.44218,1,new-section,-101.625

| | | | |
|---|---|---|---|
| 1. nanoLES ： | 协议头 | 8. 79： | 序列号 |
| 2. T： | 数据类型 | 9. 1.44218： | 忽略 |
| 3. 032a2559： | ID，4 字节 | 10. 1： | 是否有效 |
| 4. 00： | 电量 | 11. new-section： | 单元名称 |
| 5. -0.17,1.92,0.00： | 坐标 X，坐标 Y，坐标 Z | 12. -101.625： | 信号均值 |
| 6. Inf： | 忽略 | | |
| 7. 2020-12-28T02:47:23.310： | 时间 | | |

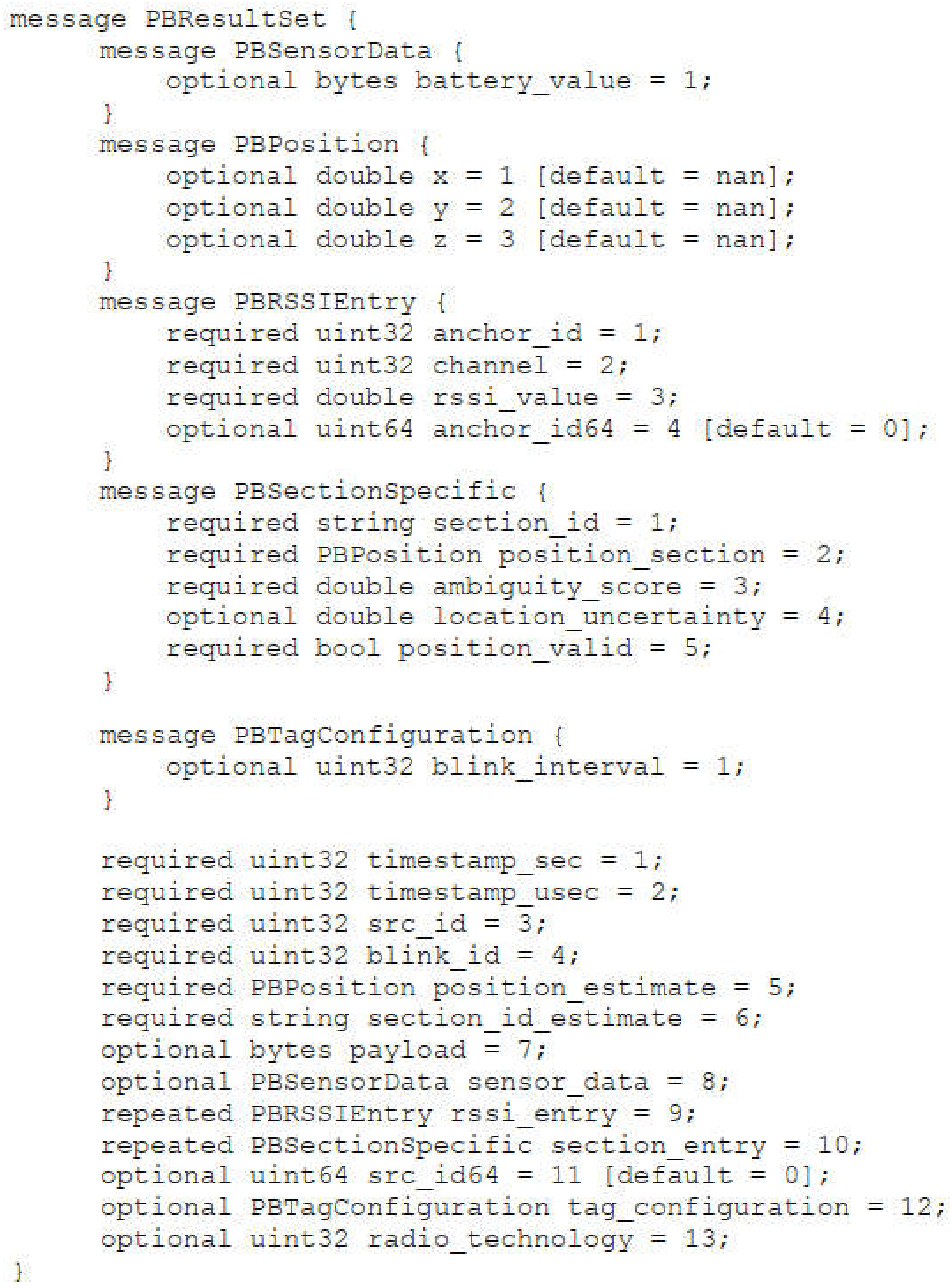
## 二进制协议（详见 Google Protocol 协议部分）

1. 采用 Google Protocol 协议， 当用户 Visual Platform 连接到引擎时，采用如下协议：



```
message PBHeader {
        required uint32 magic_number = 1;
        required uint32 version_major = 2;
        required uint32 version_minor = 3;
}
```

2. 当引擎持续向用户 Visual Platform 发送数据时，采用如下协议：



```
message PBResultSet {
        message PBSensorData {
            optional bytes battery_value = 1;
        }
        message PBPosition {
            optional double x = 1 [default = nan];
            optional double y = 2 [default = nan];
            optional double z = 3 [default = nan];
        }
        message PBRSSIEntry {
            required uint32 anchor_id = 1;
            required uint32 channel = 2;
            required double rssi_value = 3;
            optional uint64 anchor_id64 = 4 [default = 0];
        }
        message PBSectionSpecific {
            required string section_id = 1;
            required PBPosition position_section = 2;
            required double ambiguity_score = 3;
            optional double location_uncertainty = 4;
            required bool position_valid = 5;
        }

        message PBTagConfiguration {
            optional uint32 blink_interval = 1;
        }

        required uint32 timestamp_sec = 1;
        required uint32 timestamp_usec = 2;
        required uint32 src_id = 3;
        required uint32 blink_id = 4;
        required PBPosition position_estimate = 5;
        required string section_id_estimate = 6;
        optional bytes payload = 7;
        optional PBSensorData sensor_data = 8;
        repeated PBRSSIEntry rssi_entry = 9;
        repeated PBSectionSpecific section_entry = 10;
        optional uint64 src_id64 = 11 [default = 0];
        optional PBTagConfiguration tag_configuration = 12;
        optional uint32 radio_technology = 13;
}
```