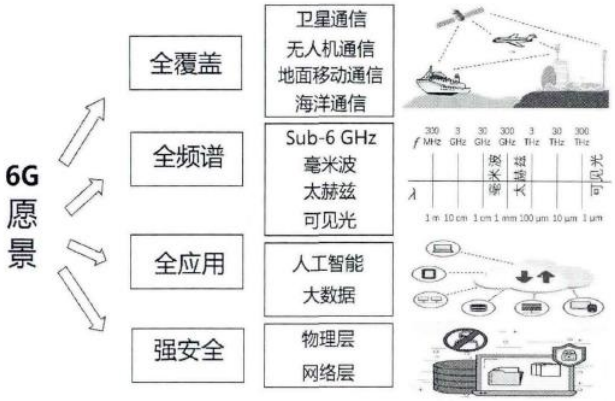
API 说明

1 无人机组网与通信

1.1 网络基础与 6G 网络简介

随着移动互联网时代的快速发展和信息全球化的深化拓展，移动通信技术正在迅猛地发展，深刻地影响和改变着人类的工作和生活方式。回顾移动通信技术的发展，第一代（The First Generation，1G）移动通信系统研发于 20 世纪 80 年代，使用模拟信号传输语音信息，提供通话服务，拉开了移动通信技术发展和演进的序幕。第二代（The Second Generation, 2G）移动通信于 20 世纪 90 年代诞生，完成了从模拟体制到数字体制的过渡。可以提供覆盖范围更广、更稳定的语音通话服务，同时开始扩展支持的业务维度。21 世纪初，第三代（The Third Generation, 3G）移动通信系统应运而生，得益于码分多址等关键技术的应用，可以实现 2Mbps 以上数据传输速率，并扩展了对移动多媒体数据业务的支持，实现了移动通信与互联网的结合。2012 年国际电信联盟正式发布了第四代（The Fourth Generation, 4G）移动通信标准，标志着 4G 移动通信系统进入商用部署。4G 移动通信以正交频分复用（Orthogonal Frequency Division Multiplexing, OFDM）和多输入多输（Multiple-Input Multiple-Output, MIMO）为关键技术，将数据传输速速率提高至 100Mbps 以上，在获得巨大商业成功的同时，成为了承载移动互联网数据业务的重要基础设施。在移动通信业务需求多元化增长的驱动下，第五代（The Fifth Generation,5G）移动通信标准 R15 和第一个演进标准 R16 已经分别于 2019 年和 2020 年正式确定，5G 通信系统在 2020 年全面进入商用阶段。5G 移动通信技术在支持增强型移动宽带（Enhance Mobile Broadband, e MBB）业务的基础上，将技术应用场景进一步扩展至高可靠低延时移动通信（Ultra-Reliable & Low Latency Communication, u RLLC ）以及大规模机器通信（Massive Machine Type Communication, m MTC)。在大规模多输入多输出（Massive MIMO)、毫米波（Millimeter wave, mm Wave）等关键技术的支持下 5G 移动通信技术实现了峰值速率、用户体验速率、频谱效率、移动性管理、时延、连接密度、网络能效、区域业务容量等性能的全方位提升，并逐渐渗透到垂直行业，提供大流量移动宽带通信业务，并被广泛应用于无人驾驶、工业控制、物联网等领域，为万物互联提供支撑。

随着 5G 移动通信的大规模商用，第六代（The Sixth Generation, 6G）移动通信技术研发也在各国正式展开，芬兰、日本、美国、韩国先后开启 6G 相关技术的研究，如图所示。我国于 2019 年 11 月启动了国家 6G 专项研究计划，成立了国家 6G 技术研发推进工作组和专家组，这标志着我国 6G 移动通信技术研发工作正式启动。6G 移动通信技术将致力于在 5G 的基础上继续深化移动互联，不断扩展万物互联的边界和范围，最终实现万物智联。从峰值传输速率的角度看，6G 将达到 Tbps 级的超高传输速率（巨流量)；从接入密度的角度看，6G 将达到 1000 万 / km^2~10000 万 / km^2 （巨连接)；从覆盖率的角度看，6G 将实现全球深度覆盖（广覆盖)；从智能化的角度看，6G 将由 5G 时代的初步智能最终实现高度智能（高智能)。为了满足 6G 移动通信系统的关键特性，提高移动通信用户的服务体验，6G 无线通信网络将采取新范式并采用新使能技术。新范式聚焦于全覆盖、全频谱、全应用、强安全四大趋势，如图所示。其中，全覆盖指 6G 将在 5G 陆地范围局部覆盖的基础上，通过卫星通信，无人机通信，海事通信等使能技术，构建空天地海一体化“泛在融合信息网络”，实现全球范围内的深度网络覆盖。



无人机通信是 6G 无线通信网络的重要组成部分，可以提高无人机任务执行效率，扩大无线网络覆盖范围。与地面通信场景相比，无人机通信场景具有独特信道特性，为无人机通信系统的设计带来了挑战。

1.2 无线信道建模

在无线通信领域，由于信道的变化导致接收信号的幅度发生随机变化的现象，这种现象称为信道衰落。信号在信道中一般通过电磁波传播，途中会产生衰落，衰落是一个针对功率的量，或者说是针对信号强度的量，当一个信号在传播过程中信号强度会不断减小。

我们要做的就是对信道衰落具体如何进行建模，这就是信道建模了，可以将信道衰落分为以下几类：

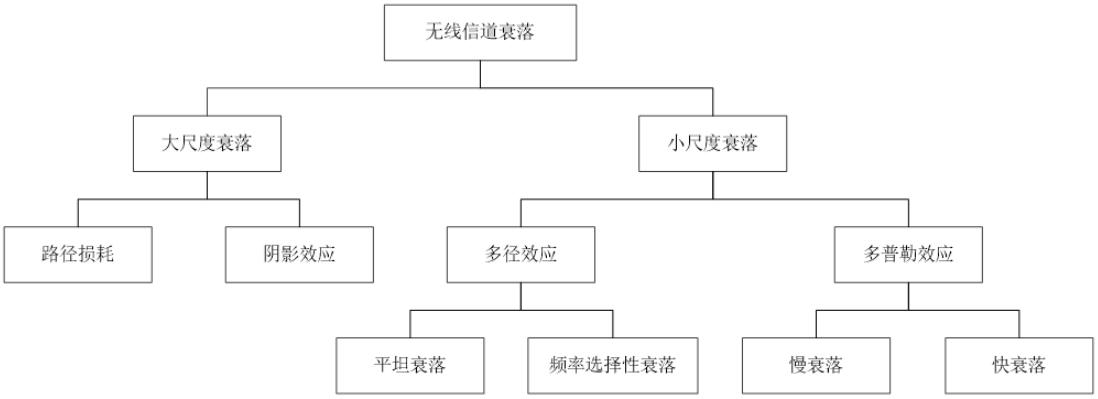


图 2 信道衰落的划分

大尺度衰落产生的原因为路径损耗、阴影效应。小尺度衰落产生的原因为多径效应以及多普勒效应。

所谓大尺度衰落，是指当移动设备移动通过的距离相对较长时（比如，小区大级别的距离）产生的衰落现象；所谓小尺度衰落是指移动台移动的距离比较短时，多条信号传播路径的相消或相张干涉会引起信号电平的快速波动。本节，我们只介绍大尺度衰落，小尺度衰落属于信号的微观层面，涉及到信号领域的分析，没有特定适合的建模，需要具体问题具体分析。

根据图 2，可以看到，大尺度衰落由路径损耗和阴影效应构成：

路径损耗指电波在空间中传播所产生的损耗，由传播中反射、绕射、散射产生，主要由移动设备收发天线的距离所决定，同时，也与电磁波传输路径的环境、电磁波频率、天

线等因素有关。对于距离而言，距离越远电磁波损耗越大，距离越近电磁波损耗越小。对于频率而言，频率越大损耗越大，频率越小损耗越小。对于环境因素而言，更为复杂，在崎岖的路径上路径损耗相对会更大，例如山地、丘陵，在平缓的路径上路径损耗相对会更小，例如草原、田地。路径损耗具有很好的可预测性，对于一种特定的环境，通常可以通过经验公式来估计，想要获得更精确的损耗则可通过理论分析、实地测量拟合得到。

阴影效应指在无线通信系统中，移动台在运动的情况下，由于大型建筑物和其他物体对电波的传输路径的阻挡而在传播接收区域上形成半盲区，从而形成电磁场阴影，这种随移动台位置的不断变化而引起的接收点场强中值的起伏变化叫做阴影效应。阴影效应一般服从对数正太分布，波动为大尺度波动，因此也具有可预测性。

首先，我们分析路径损耗，在分析其原理之前，我们需要了解电磁波的传播行为：可视传播以及不可视传播。可视传播指的是电磁波在信道中传播没有显著障碍物遮挡，不可视传播则指的是电磁波在信道中传播受到了一定障碍物遮挡。对电磁波传播行为的判断可以通过第一菲涅尔区来做参考。

早在 16 世纪，荷兰物理学惠更斯提出，一个波阵面的每个点（面源）可各看做是一个产生球面子波的次级球面波的中心波源，次级波源的波速与频率等于初级波的波速与频率；而且，以后任何时刻波阵面的位置是所有这种子波的包络面。这说明，介质中任何一处的波动状态是由介质各处的波动决定的。

而后，在 18 世纪，法国物理学家菲涅尔对惠更斯原理做了相应补充，用数学的形式给出了计算某点波动的计算方式，同时，提出了菲涅尔区的新概念。

菲涅尔原理认为，空间任意一点处的辐射场，是包围波源的任意封闭面上产生的所有次波，在该点所产生的场叠加的结果，包围波源的任意封闭面的每一点，都可以看作次波源，各自发出球面次波。

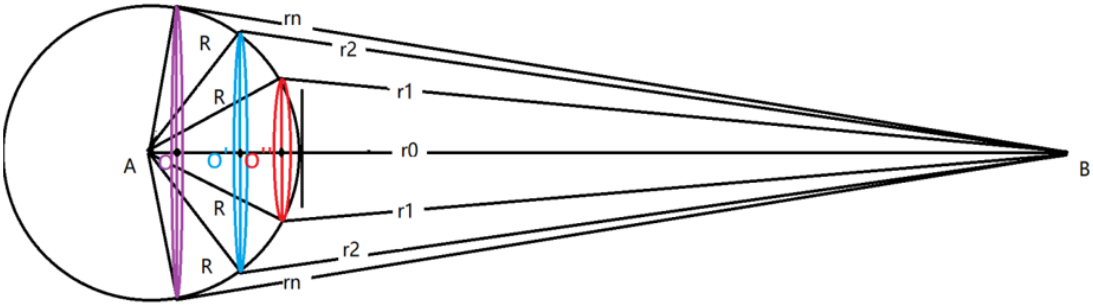


图 3 菲涅尔通道和半径

图 3 描述了菲涅尔区通道与半径。A、B 点分别为发送电磁波源和接收电磁波点，假定 A、B 直线路径上有 O''、O' 以及 O 三点。根据菲涅尔半波带法，将 A 点作为球心作半径为 R 的球，使得 A 经过球上的点到达 B 满足:

$$
\begin{aligned}
&r0+R=r0+R\\
&r1+R=r0+R+\frac{\lambda}{2}\\
&r2+R=r0+R+2\times\frac{\lambda}{2}\\
&\cdots\cdots\\
&rn+R=r0+R+n\times\frac{\lambda}{2}
\end{aligned}
$$

(9-1)

式中，λ为电磁波的波长。当n为 1 时，图中红色的圆即为$o"$点的第一菲涅尔区通道，其半径称为$o"$点第一菲涅尔区半径。同理，当n为 2 时，图中蓝色的圆即为o'点的第二菲涅尔区通道，其半径称为o'点的第二菲涅尔区半径。依此类推，n代表的紫色圆即为o点的第n菲涅尔区通道，其半径为O点的第n菲涅尔区半径。实际上，在通信领域更关心第一菲涅尔区半径，计算某点第一菲涅尔区半径公式为:

$$r = \sqrt{\frac{\lambda d_1 d_2}{d}} \quad (9\text{-}2)$$

$$d = d_1 + d_2 \quad (9\text{-}3)$$

式中，d为 A、B 两点的距离，d_1、d_2分别为 A 到该点的距离、B 到该点的距离，如图 3 中红色的圆代表的$o"$点第一菲涅尔区通道，其半径为$r = \sqrt{\frac{\lambda|Ao"||Bo"|}{|Ao"|+|Bo"|}}$

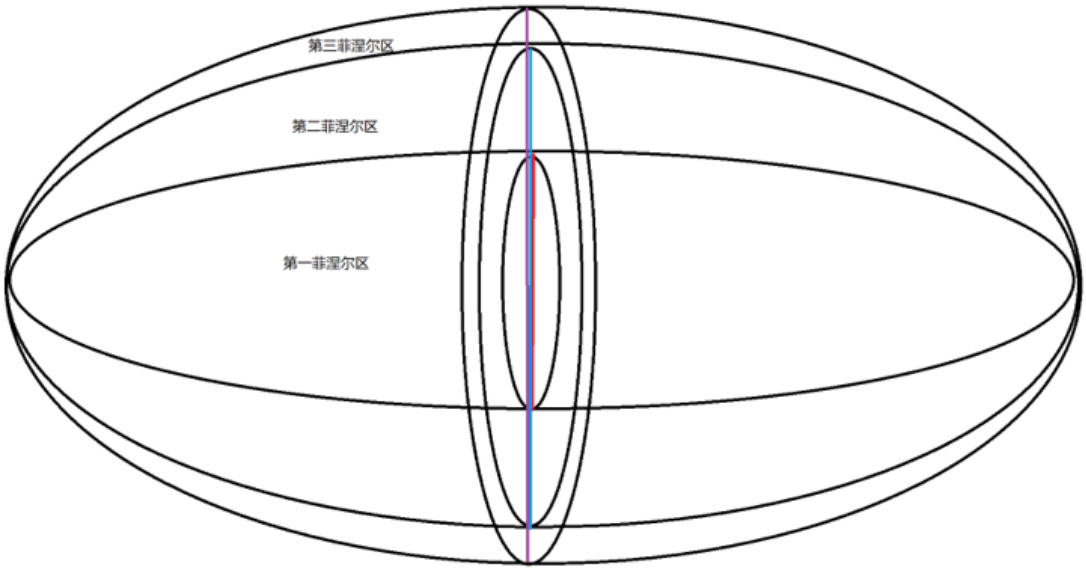


图 1 菲涅尔区的划分

图 4 描述了菲涅尔区的划分，从剖面上看，一个菲涅尔区由无数个菲涅尔通道组成，式（2-1）中做如下变换：

$$rn + R = r0 + R + n * \frac{\lambda}{2} = |AB| + n * \frac{\lambda}{2} \quad (9\text{-}4)$$

式中，$|AB|$为 AB 两点间的直线距离。当n确定时，式（9-4）中$|AB| + n * \frac{\lambda}{2}$为常数，等式右边恒为常数，使得等式左边也恒为常数，根据两焦点到任意一点的距离之和为常数的椭圆特点，满足该式中的所有点组成的轨迹可以看成以 A、B 为焦点的椭圆体。当n为 1 时，则椭圆体覆盖范围为第一菲涅尔区，依次类推，可以得到第n菲涅尔区。

通过菲涅尔原理，可以计算任意一点的场强大小，该点振幅大小为第一菲涅尔半波带产生的幅值的一半，说明第一菲涅尔区的影响是非常大的，在判断是否满足视距条件中，主要研究第一菲涅尔区。

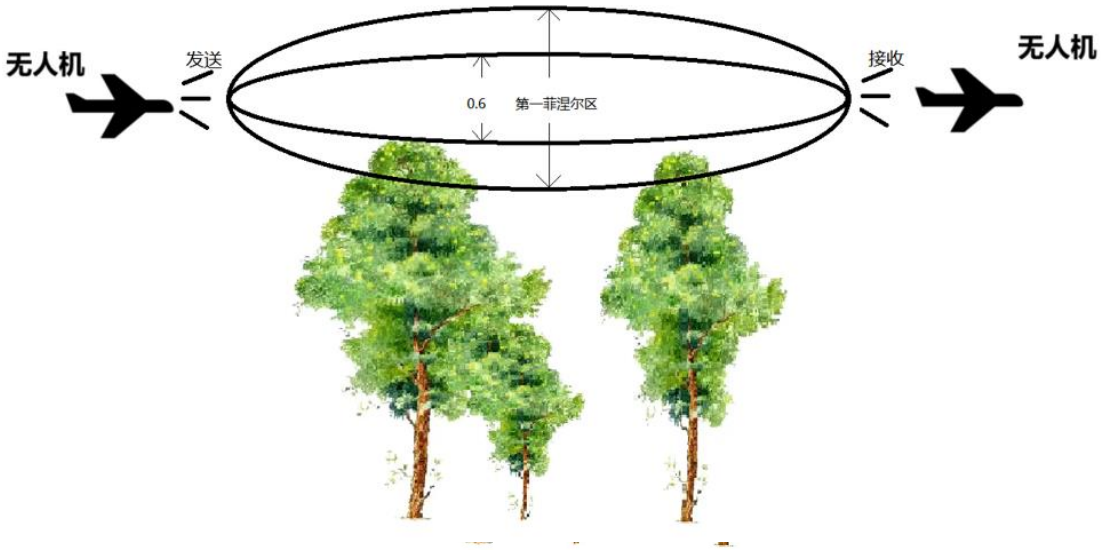


图 5 通信中遮挡菲涅尔区的示意图

图 5 演示了第一菲涅尔区（图中菲涅尔区）在通信中的影响，可视条件下，无线信号无遮挡地在发信端与接收端之间直线传播，这要求在第一菲涅尔区内没有对无线电波造成遮挡的物体，这种传播为视距传播（LOS Propagation）；如果条件不满足，则信号强度就会明显下降，当障碍物完全遮挡第一菲涅尔区时，这种传播称为非视距传播（NLOS Propagation）。一般来说，我们希望障碍物遮挡第一菲涅尔区域边界不超过其范围的 20%。

在判定传播行为可视的情况下，一般的，不管是任何情况的通信，我们可以用自由空间传播模型（Free space propagation Model）以及对数距离路径损耗模型（Log-distance Path Loss Model），这是适用性最广的模型。

自由空间传播模型用于预测接收机和发射机之间完全无阻挡的视距路径时接收信号的场强，属于大尺度路径损耗的无线电波传播的模型。其公式为：

$$P_r = P_t \frac{G_t G_r (\lambda)^2}{(4\pi)^2 d^2 L} \quad (9\text{-}5)$$

式中，P_t为发送功率，G_t为发送天线增益，G_r为接收天线增益，λ为波长（m），d为发送天线与接收天线的直线距离（m），L为系统损耗系数，包括系统硬件系统的总体衰减或损耗。

对式（9-6）中等式左右两边同时取对数转换为 db 后，可以得到以下表达式：

$$10logP_r = 10logP_t + 10log_{10}\left(\frac{G_t G_r (\lambda)^2}{(4\pi)^2 d^2 L}\right) \quad (9\text{-}6)$$

$$P_r[db] = P_t[db] - L_{Free}[db] \quad (9\text{-}7)$$

L_{Free}为自由空间传播模型的路径损耗：

$$L_{Free}[db] = -10log_{10}\left(\frac{G_t G_r (\lambda)^2}{(4\pi)^2 d^2 L}\right) = 10log_{10}\left(\frac{(4\pi)^2 d^2 L}{G_t G_r (\lambda)^2}\right) \quad (9\text{-}8)$$

通过L_{Free}，可以预测视距条件下的接收天线接收功率的情况。

对数距离路径损耗模型则是通过引入随环境而改变的路径损耗指数，来修正自由空间传播模型，其公式为：

$$P_r[db] = P_t[db] - L_{LogDistance}[db] \quad (9\text{-}9)$$

$$L_{LogDistance}[db] = L_{Free}(d_0)[db] + 10\gamma log_{10}\left(\frac{d}{d_0}\right) \quad (9\text{-}10)$$

式中，$L_{LogDistance}$为对数距离路径损耗模型的路径损耗，γ为路径损耗指数，d、d_0分别发送天线与接收天线的直线距离（m）和天线远场的参考距离（m）。由于天线近场存在散射现象，信道会较为复杂，一般式（9-10）只适用于发送距离 $d > d_0$ 的情形。d_0对于室内一般为 1m~10m，对于室外一般为 10m~100m 甚至更大。

那么，我们再来说说阴影效应的建模。

阴影效应产生是由于障碍物的遮挡而使接收区域产生阴影区，电磁波经过阴影区后会使信号下降。阴影效应产生的损耗量一般服从对数正太分布，用$X_\sigma[db]$表示，假定均值为$\bar{X}[db]$，方差为σ，其概率密度函数为：

$$P(X_\sigma) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(X_\sigma - \bar{X})^2}{2\sigma^2}\right) \quad (2\text{-}20)$$

所以，在有障碍物遮挡的情况中，还需要额外考虑X_σ的影响。

1.3 无人机常用应用层协议 (DDS、MAVlink)

DDS（Data Distribution Service for Real-Time Systems）是一种面向数据的中间件，它为实时系统提供了一种标准的方式来发布、发现和订阅分布式数据。

DDS 的核心概念是数据发布/订阅模型，其中发布者向主题（Topic）发布数据，而订阅者可以从这些主题接收数据。DDS 使用一种称为“服务发现协议”（SMP 或 SDP）的技术来发现网络上的其他 DDS 实体（如发布者、订阅者等）。这意味着当一个 DDS 节点启动时，它可以自动找到网络上可用的主题和服务。DDS 可以利用不同的传输层协议进行通信，包括 TCP/IP、UDP、共享内存等。这使得 DDS 能够适应各种网络环境和性能需求。DDS 提供了多种服务质量（QoS）策略，比如可靠性、持久性、顺序保证等。这些策略可以根据应用的具体需求调整，以优化数据传输。DDS 支持定义数据类型，并确保这些类型在不同平台上保持一致。这使得 DDS 能够处理复杂的异构系统。DDS 的设计目标之一就是提供低延迟、高吞吐量的实时通信。

DDS 作为无人机应用层协议有以下优点：实时性能：DDS 是专为实时系统设计的，能够满足无人机在飞行控制和其他关键任务中的低延迟要求。松耦合：DDS 允许发布者和订阅者之间松散地耦合，这意味着系统中的组件可以独立开发和更新，而不会影响其他部分。灵活性：DDS 支持动态网络拓扑，因此可以在飞行过程中添加或删除节点。这对于需要灵活调整任务配置的无人机来说非常重要。可扩展性：由于 DDS 是基于主题的数据发布/订阅模型，所以随着系统的复杂性和规模的增长，它可以轻松地处理更多的数据流和参与者。服务质量（QoS）策略：DDS 允许用户根据具体需求选择不同的 QoS 策略，比如可靠性、持久性、顺序保证等，以优化数据传输。标准化：DDS 是由 OMG 组织制定的标准，这有助于确保不同供应商之间的互操作性和兼容性。ROS 集成：DDS 与机器人操作系统（ROS）有很好的整合，特别是在 ROS 2 中，DDS 作为默认的通信框架被广泛采用。国防和工业应用：鉴于 DDS 最初是在军事领域开发的，并已成功应用于多个行业，它的可靠性和安全性对无人机应用来说是一个重要的优势。

DDS 的使用限制主要有以下方面。DDS 具有许多高级功能和配置选项，这使得它相对较为复杂。与一些简单的消息传递系统相比，DDS 可能需要更多的计算和内存资源来运行。为了支持实时性能，DDS 引入了一些额外的开销，比如缓存管理、网络协议处理等。这些开销可能会影响系统的整体效率。

MAVLink 协议是一种轻量级的通信协议，特别为小型无人航空系统（UAS）设计。它在无人机应用层中的作用是提供一种标准化的方式来交换飞行控制信息、传感器数据和其他与任务相关的数据。

MAVlink 作为无人机应用层协议有以下优点：紧凑的消息结构：消息体通常很小，适合于低带宽和高延迟的无线链路。跨平台支持：可以在各种硬件平台上运行，包括嵌入式设备和 Windows 系统。易于实现：有大量的开源库和工具可供使用，简化了开发过程。

MAVlink 主要有以下限制。缺乏服务质量保证：它不提供 QoS 机制来确保关键数据的可靠传输。缺乏实时性能：虽然可以用于实时系统，但不是专门设计为满足严格的实时要求。没有内置安全功能：需要外部机制来保护数据的安全性和完整性。

相比较而言，两种协议具有相似的功能，而具有不同的适用范围。主要区别为 DDS 是一种更为复杂也更加强大的实时数据分发协议。DDS 设计方向是为了解决大规模的分布式系统中的数据共享问题，因此更适合于需要强大数据管理和分发能力的大型、复杂的分布式系统。并且允许用户根据自身需求配置不同的 QoS 策略，定制数据传输。MAVlink 相比而言更适用于简单的无人机应用，尤其在资源受限的环境中。

1.4 无人机常用路由协议 (AODV、OLSR、DSR、DSDV)

AODV：Ad-hoc On-Demand Distance Vector Routing（AODV）是一种用于移动自组网（MANET）的路由协议。在这样的网络中，没有固定的基础设施或中心控制节点，每个节点都必须能够作为路由器转发数据包，以实现整个网络的通信。由于这种网络环境中的节点可以随意移动，因此网络拓扑可能会频繁变化。

其工作原理如下：当源节点需要发送数据到目的节点，并且在本地路由表中没有可用路由时，它会广播一个“路由请求”（RREQ）消息。RREQ 消息沿着网络传递，每个收到它的节点都会记录下该请求的来源并转发给其邻居。目的节点接收到 RREQ 后，回复一个“路由应答”（RREP）消息，其中包括到达源节点的最短路径信息。路由应答沿原路径返回，每个节点在收到应答后更新其路由表，以便将来可以通过该路径直接发送数据。

在无人机自组网中，AODV 协议的应用非常关键。由于无人机通常具有高度移动性，它们之间的通信网络必须能够快速适应不断变化的拓扑。此外，由于无人机可能执行远程任务或者处于偏远地区，网络带宽和能量效率都是重要的考虑因素。

AODV 的优势在于其按需特性，这意味着它仅在实际需要时才创建和维护路由，这有助于减少不必要的网络开销。然而，对于高动态性的场景，如多无人机协同飞行，AODV 可能需要进行优化以提高路由稳定性、降低延迟和改善整体性能。

OLSR：Optimized Link State Routing Protocol（OLSR）是一种用于移动自组网（MANET）的链路状态路由协议。与 AODV 这样的距离向量协议不同，OLSR 使用了全局网络拓扑信息来进行路由决策。

其工作原理如下：每个节点周期性地发送 Hello 消息给其邻居，以检测和更新邻居列表。基于邻居列表，每个节点选择一组 MPR 节点，并在下一次 Hello 消息中宣布这个选择。当收到一个 Hello 消息时，节点会更新其邻居和 MPR 集合。节点根据邻居和 MPR 信息生成一个 TC 消息，其中包含所有可达节点的信息。TC 消息通过 MPR 节点进行转发，从而在整个网络中传播拓扑信息。每个节点根据接收到的 TC 消息更新其路由表，以便为数据包找到最佳路径。

在无人机通信中，OLSR 可能比 AODV 更适合一些场景，因为它提供了更好的网络可视性和更快的收敛速度。然而，OLSR 的主要缺点是它需要更多的带宽和计算资源来维护和传播拓扑信息。

DSR：Dynamic Source Routing (DSR) 是一种用于移动自组网（MANET）的源路由协议。与 AODV 和 OLSR 这样的传统路由协议不同，DSR 不需要每个节点都维护整个网络的拓扑信息，而是让数据包携带完整的路径信息。

DSR 的工作原理如下：当源节点需要发送数据到目的节点时，它会在其路由缓存中查找是否有到达目的节点的有效路由。如果找到有效路由，则直接使用该路由发送数据包。如果没有找到有效路由，则源节点会广播一个“路由请求”（Route Request, RREQ）消息。RREQ 消息沿着网络传递，每个收到它的节点都会记录下该请求的来源并转发给其邻居。目的节点接收到 RREQ 后，回复一个“路由应答”（Route Reply, RREP）消息，其中包括到达源节点的最短路径信息。路由应答沿原路径返回，每个节点在收到应答后更新其路由缓存，以便将来可以通过该路径直接发送数据。

在无人机通信中，DSR 可能比其他路由协议更适合一些场景，因为它提供了更低的控制开销和更高的灵活性。然而，DSR 的主要缺点是它对网络中的环路非常敏感，需要采取额外的措施来避免环路。此外，由于每个数据包都需要携带完整的路径信息，这可能会导致数据包头部过大，从而影响传输效率。

DSDV：Destination-Sequenced Distance Vector (DSDV) 是一种用于移动自组网（MANET）的距离向量路由协议。与 AODV 和 OLSR 这样的按需路由协议不同，DSDV 使用周期性更新机制来维护网络中的路由表。

DSDV 的工作原理如下：每个节点定期发送“路由更新”（Route Update, RU）消息给其邻居，该消息包含了到其他所有已知节点的路由信息。每个收到 RU 消息的节点会更新其路由表，并将这些更新转发给它的邻居。路由表中的每条路由都有一个序列号，新接收到的路由如果序列号比当前存储的路由更大，则会被认为是更好的路由，从而被更新到路由表中。当源节点需要发送数据到目的节点时，它会在本地路由表中查找到达目的节点的最佳路径，并使用这个路径发送数据包。

在无人机通信中，DSDV 可能比其他路由协议更适合一些场景，因为它提供了更强的健壮性和更低的延迟。然而，DSDV 的主要缺点是它需要大量的带宽和计算资源来维护和传播全网范围的路由信息。此外，由于 DSDV 基于距离向量算法，它可能对网络中的链路波动和错误非常敏感。

2 无人机组网开发平台与仿真设计

2.1 粗粒度组网通信仿真接口

2.1.1 NetUavAPI.NetTransNode:__init__()

这个接口用以初始化一个新的通信节点。参数为 PX4MavCtrl.PX4MavCtrler()

2.1.2 startNetServ()

这个接口用以开启开启网络仿真器。将会开启 ListenMsgLoop 和 SendMsgLoop 两个循环。

参数：self,RecPort=-1,netSimPort=20030,netSimIP='224.0.0.10'

RecPort：监听端口。

netSimPort：发送端口。

netSimIP：发送 IP。

2.1.3 endHeartSer()

这个接口用以关闭网络仿真器。将会关闭 ListenMsgLoop 和 SendMsgLoop 两个循环。

2.1.4 NetUavAPI.HeartServer:__init__()

这个接口用以初始化心跳服务器。

参数：

self.ReceiveIP：接收 IP

self.ReceivePort：接收端口

self.SendIP：发送 IP

self.SendPort：发送端口

self.RecHeartThreadFlag：开启接收的指令

self.SendHeartThreadFlag：开启发送的指令

self.CheckStateThreadFlag：开启状态确认的指令

2.1.5 startHeartSer()

用以开启心跳检测服务器仿真。

将开启三个进程：ReceiveHeartSer，SendHeartSer，CheckConnectState。

ReceiveHeartSer：从绑定的 ReceiveIP，ReceivePort 接收数据。当 RecHeartThreadFlag 为 1

时，开启接收。

SendHeartSer：向指定的 SendIP，SendPort 发送信息。SendHeartThreadFlag 为 1 时，开启发送。

CheckConnectState：CheckStateThreadFlag 为 1 时，将会开启检测与其他飞机的通信延迟。

2.1.6 endHeartSer()

用以停止状态检测服务器。将停止 ReceiveHeartSer，SendHeartSer，CheckConnectState 三个进程。

2.2 粗粒度 UE 信号衰减仿真接口

2.2.1 CoarseNetworkSimulation:__init__()

接口用以初始化粗粒度仿真。

参数：numberOfNodes:int, IDstring='', maxRange:float=1.5, bandwidth:float=100.0

numberOfNodes：仿真节点数量

IDstring：节点 ID

maxRange：最大通信距离放大系数。

bandwidth：通信带宽。

2.2.2 runSimulation()

用以开启仿真。将开启 sendRoutingTable 和 receiveFromCopterSim 两个进程。通过调用 RedisUtils.RedisUtils()创建 Redis 通信。

2.2.3 sendRoutingTable()

用以将信号衰减数据发送至相应的 key。函数将调用 calculateRoutingTable()，calculate PacketLossRate()，GetPos()，calculatePathLose()，Dijkstra()函数。用以计算 UE 信号衰减信息以及自身位置信息。

calculateRoutingTable()：衰减率计算。

calculatePacketLossRate()：丢包率计算。

GetPos()：获取飞机自身位置信息。

calculatePathLose()：信号衰减信息计算。

Dijkstra()：维护节点间的最优通信。

2.2.4 CreateTarget()

用以创建 Redis 通信实例。并向 Redis 服务器发送一些信息。

2.3 粗粒度网络信号仿真接口

2.3.1 PublicUavData()

发布飞机自身信息。

```
UAV1 = {
  "timeDelay":TimeUnix,
  "CopterID":mav.CopterID, # 飞机仿真 ID
  "uavTimeStmp":mav.uavTimeStmp,  # double uavTimeStmp 时间戳
  "uavAngEular":mav.uavAngEular, # float uavAngEular[3] 欧拉角 弧度
  "uavVelNED":mav.uavVelNED, # float uavVelNED[3] 速度 米
  "uavPosGPSHome":mav.uavPosGPSHome, # double uavPosGPSHome[3] GPS 维度（度）、经度（度）、高度（米）
  "uavPosNED":mav.uavPosNED, # double uavPosNED[3] 本地位置 米 （相对起飞点）
  "uavGlobalPos":mav.uavGlobalPos} # double uavGlobalPos[3] 全局位置 （相对与所有飞机的地图中心）
```

2.3.2 sub_callback()

将收到的数据分别存储到本地的相应路径下，同时记录通信延迟。Channel 作为标识，data 数据结构在 PublicUavData()做了说明。

参数：

Channel: 订阅的频道。

Data: 频道收到的数据。

2.3.3 sub_data_multiple_channels()

用以订阅多个频道的信息。将订阅传入的频道，当收到的信息数据类型与期望的类型相同时，将调用回调函数对信息进行处理。将调用 sub_callback()

参数：

message_type：订阅的数据类型。

Channels：订阅的频道。

Callback：调用的回调函数。

2.4 MQTT 组网通信仿真接口

2.4.1 mqtt.client.Client:__init__()

这是 Mqtt 初始化客户端的接口。

参数：self, client_id="", clean_session=None, userdata=None, protocol=MQTTv311, transp

ort="tcp", reconnect_on_failure=True

client_id：这个参数作为当前节点连接到代理终端时使用的唯一客户端 ID 字符串。这个参数可以自己指定，也可以为空。如果为空，使用 MQTT v3.1.1 时，将会由代理终端为客户端生成 ID。如果使用 MQTT v3.1，则由自身随机生成。如果 id 为空，clean_session 必须为 True，否则会导致 Value Error 错误。

clean_session：这个参数是一个布尔值。为 True 时，当客户端与代理断开连接时，代理会清空客户端 ID 的所有信息。为 False 时，会保存当前客户端 ID 下的信息。

Userdata：这是用户自定义的数据。将会传递给回调函数 user_data_set()。

Protocol：参数允许显式设置此客户端应使用的 MQTT 版本。默认为 v3.1.1。

Transport：这个参数指定传输机制。有两个选择：websockets，tcp。默认是 tcp。

reconnect_on_failure：这个参数是一个布尔值。True 时，客户端在连接失败后会自动尝试重新连接。False 时，客户端不会重复尝试连接。默认为 True。

2.4.2 connect()

这个接口用以将创建的客户端与远程代理端连接。

参数：self, host, port=1883, keepalive=60, bind_address="", bind_port=0, clean_start=MQTT_CLEAN_START_FIRST_ONLY, properties=None

Host：这个参数代表远程代理的主机名或 IP 地址。

Port：是要连接到的服务器主机的网络端口。默认值为 1883。

Keepalive：这是客户端与代理通信的最大时间间隔。如果在时间内没有消息交换，客户端将发送 ping 消息给代理端。

bind_address：这个参数将用于确定客户端用于发送数据的本地 IP 地址。如果这个参数未被指定，系统将自行选择。

bind_port：这个参数用以确定客户端用以发送数据的本地端口。如果这个参数未被指定，系统将自行选择。

clean_start：这是 MQTT v5.0 中的一个连接参数，用于控制客户端和服务器之间的会话行为。True 时，客户端请求建立一个全新的会话，连接时，服务器将清空之前与该客户端 ID 的对话数据。False 时，将建立一个持久的会话，这代表代理将会保存之前与该客户端 ID 的对话数据。MQTT_CLEAN_START_FIRST_ONLY 时，这个参数只在客户端初次与代理连接时生效。

Properties：在 MQTT 连接包中发送的 MQTT v5.0 属性。

2.4.3 publish()

这个函数用以向代理端发送话题信息。

参数：self, topic, payload=None, qos=0, retain=False, properties=None

Topic：消息应发布的主题。

Payload：这是要实际发送的消息。

Qos：这是要选择的服务质量级别。0 时，客户端最多发送一次消息，传输速度最快，消息可能丢失，客户端不会确认代理端是否收到消息。1 时，客户端最少发送一次消息，增加了部分延迟，提供了更高的可靠性，客户端在确认代理段收到消息前会重新发送消息。2 时，客户端只发送一次消息，可靠性最高，客户端将与代理段进行复杂的交互以确保数据的一致。

Retain：这个参数将用以设定，是否将这条消息设定为该主题的“已知良好”保留消息。True 时，将会将消息定为保留消息。

Properties：（仅限 MQTT v5.0）要包含的 MQTT v5.0 属性。使用 Properties 类。

2.4.4 loop_start()

这是线程化客户端接口的一部分。只需调用一次此方法即可启动一个新线程来处理网络流量。这提供了一种替代方案，可以避免自己反复调用 loop() 方法。

这个函数不必传入参数。

2.4.5 disconnect()

这个函数用以断开客户端与代理端的连接。

参数：self, reasoncode=None, properties=None

Reasoncode：（仅限 MQTT v5.0）一个 ReasonCodes 实例，用于设置要随 disconnect 一起发送的 MQTT v5.0 原因代码。

properties：（仅限 MQTT v5.0）一个 Properties 实例，用于设置要包含的 MQTT v5.0 属性。如果未设置，则不发送任何属性。

2.4.6 tls_set()

这个函数用以配置网络加密和身份验证。启用 SSL/TLS 支持。函数必须在 connect()函数之前调用。

参数：self, ca_certs=None, certfile=None, keyfile=None, cert_reqs=None, tls_version=None, ciphers=None, keyfile_password=None

ca_certs：一个字符串路径，指向客户端应视为受信任的证书颁发机构证书文件。

certfile：指向 PEM 编码客户端证书的字符串。

keyfile：指向 PEM 编码客户端私钥的字符串。这个参数将和 certfile 用作基于 TLS 的身份验证的客户端信息。支持此功能取决于代理。

cert_reqs：允许更改客户端对代理施加的证书要求。默认值为 ssl.CERT_REQUIRED，这意味着代理必须提供证书。

tls_version：允许指定使用的 SSL/TLS 协议版本。默认情况下使用 TLS v1.2。

ciphers：是一个字符串，指定此连接允许使用的加密密码。

keyfile_password：如果 certfile 和 keyfile 的任何一个被加密并且需要密码解密，那么

可以使用 keyfile_password 参数传递密码。如果没有提供 keyfile_password，密码将在终端窗口中请求输入。

2.5 DDS 组网通信仿真接口

2.5.1 UavMessageWriter.Writer：__init__()

主要作用是设置和配置一个 DDS 发布者，包括创建域参与者、主题、发布者和数据写入器，并注册数据类型和支持的监听器。

参数：def __init__(self, domain, machine,UavMessageDataType,TopicType)

Domain：表示 DDS（Data Distribution Service）中的域 ID。

Machine：用于标识或者描述执行这个发布者的机器或设备。

UavMessageDataType：用于指定要发布的数据类型名称。

TopicType：表示要创建的主题的名称。

2.5.2 UavMessageWriter.Writer.write()

将给定的数据写入到 DDS（Data Distribution Service）系统中，以便其他订阅者可以接收到这些数据。

参数：def write(self,timeDelay,CopterID,uavTimeStmp,uavAngEular,uavVelNED,uavPosGPSHome,uavPosNED,uavGlobalPos)

timeDelay：延迟

CopterID：无人机 ID

uavTimeStmp：无人机时间戳

uavAngEular：无人机欧拉角

uavVelNED：无人机北东地坐标系下速度

uavPosGPSHome：初始化位置。维度，经度，高度

uavPosNED：无人机北东地坐标系中的位置

uavGlobalPos：无人机全球坐标系中的位置

2.5.3 UavMessageReader.Reader：__init__()

主要作用是设置和配置一个 DDS 订阅者，包括创建域参与者、主题、订阅者和数据读取器，并注册数据类型和支持的监听器。

参数：def __init__(self, domain, data_callback,UavMessageDataType,TopicType)

Domain：表示 DDS（Data Distribution Service）中的域 ID。

data_callback：这是一个函数对象参数，由用户提供。当订阅者接收到新的数据时，这个回调函数会被调用

UavMessageDataType：用于指定要订阅的数据类型名称。

TopicType：要订阅的主题的名称。

2.5.4 UavMessageReader.ReaderListener:__init__()

定义了一个名为 ReaderListener 的类，该类继承自 fastdds.DataReaderListener。DataReaderListener 是用于处理与数据读取器相关的事件和回调的类。

创建一个 ReaderListener 对象实例，它通常用于处理与数据读取器相关的事件和回调。这里的 data_callback 参数是一个用户提供的回调函数，当接收到新数据时会被调用。

参数：def __init__(self,data_callback,TopicType)

data_callback：订阅器触发时期望调用的回调函数

TopicType：订阅的话题名

2.6 Redis 组网通信仿真接口

2.6.1 RedisUtils:__init__()

作用是在创建该类的实例时设置和建立与 Redis 数据库的连接。

参数：

Host：设置 Redis 服务器的主机名或 IP 地址

Port：设置 Redis 服务器的端口号

Db：设置要连接的 Redis 数据库编号

Password：设置 Redis 服务器的密码

Rdb：创建一个 redis.StrictRedis 对象实例，并使用前面设置的主机名、端口号、数据库编号和密码连接到 Redis 服务器。这个对象 (self.rdb) 用于执行 Redis 命令

Pubsub：调用 self.rdb（即已连接的 Redis 实例）的 pubsub() 方法，返回一个 Pub/Sub（发布/订阅）对象实例 (self.pubsub)。这个对象用于处理 Redis 的发布/订阅功能

2.6.2 RedisUtils. sub_data()

作用是在指定的 Redis 频道上订阅数据，并在接收到匹配特定类型的消息时调用回调函数来处理这些消息。

参数：def sub_data(self,message_type ,channel, callback)

message_type：需要订阅的消息类型

channel：订阅的频道名

callback：回调函数

2.6.3 RedisUtils. pub_data()

在 Redis 中的指定频道上发布数据，并且发布的数据是以 JSON 格式编码的字符串。

参数：def pub_data(self,key, data)

Key：指定的发布频道

Data：需要发布的数据

2.6.4 RedisUtils. sub_data_multiple_channels()

作用是在指定的多个 Redis 频道上订阅数据，并在接收到匹配特定类型的消息时调用回调函数来处理这些消息。

参数：def sub_data_multiple_channels(self, message_type, channels, callback)

Channels：需要订阅的频道，数据类型应该是一个列表或者集合。

其余参数与 RedisUtils. sub_data()的参数意义相同。

2.6.5 RedisUtils.set_data()

用以在 redis 中创建或者更新对应的频道数据。

参数：def set_data(self,key, data)

Key：要创建或者更新的键值。

Data：需要写入的数据。

与 publish 不同的是，publish 的数据不会存储在 Redis 服务器中，set 的数据将会在 Redis 服务器中保存。

2.6.6 RedisUtils.get_data()

获取相应键值的值。

参数：def get_data(self,key)

Key：要获取的目标。

Set_data 与 get_data 作为相应的通信方式。与 pub，sub 不同，但都可实现通信功能。

2.7 NS3 细粒度组网通信仿真接口

2.7.1 ns3_transition()

作用是在 NS3 网络模拟环境中实现从一个无人机（src_copterID_Uav）向另一个无人机（dst_copterID_Uav）发送数据包（packet）的功能。

参数：void ns3_transition(uint16_t src_copterID_Uav, uint16_t dst_copterID_Uav, Ptr<Packet> packet)

src_copterID_Uav：发送数据的无人机 ID

dst_copterID_Uav：目标无人机的 ID

packet：需要发送的数据包

2.7.2 ReceiveUav()

功能是接收通过指定 Socket（sock）传来的数据包，并根据数据包内容进行进一步的处理和转发。

参数：void ReceiveUav(Ptr<Socket> sock)

Sock：需要接收数据的接口

2.3.3 参数配置

uint32_t routing_protocol = 1; //>>网络层使用的路由协议

```
switch (routing_protocol)
case 1: "OLSR";
case 2: "AODV";
case 3: "DSDV";
case 4: "AODVKMEANS";
case 5: "PARROT";
case 6: "GPSR";
case 7: "DSR";
default: exit(1);
```

int adhocNodes_N = 4; //>>自组网节点数量（默认为 4）

int BS_N = 1; //>>基站节点数量（默认为 1，即只需要 1 个该节点接收物理网络数据）

uint64_t frequency = 5180; //>>频率 Mhz

double txPowerBaseDbm = 20; //>>发送功率小 dbm

double txPowerEndDbm = 20; //>>发送功率大 dbm

double Start_time = 5; //>>仿真开始时间

double Total_time = 1000; //>>仿真结束时间

double init_energy = 1 * 11 * 3600; //>>最大初始能量

2.8 集群编队组网通信仿真接口

2.8.1 PX4MavCtrler: __init__()

参数：self, ID=1, ip='127.0.0.1',Com='udp',port=0

ID：仿真 ID

Ip：数据向外发送的 IP 地址

Com：与 Pixhawk 的连接模式

Port：端口号

函数用以初始化飞机，飞机的通信模式。

2.8.2 InitMavLoop()

参数：UDPMode=2

UDPMode：0 和 1 对应 UDP_Full 和 UDP_Simple Mode，2 和 3 对应 MAVLink_Full and MAVLink_Simple mode，4 对应 MAVLink_NoSend

isCom：串口通信的标志

isRealFly：网络直连模式的标志

isRedis：Redis 模式的标志

函数用以建立通信。默认模式是 MAVLink_Full

2.8.3 initOffboard()

这个函数不需要输入参数启动。

函数发送 Offboard 指令，使飞机进入 Offboard 模式，并且开始以 30Hz 的频率发送 Offboard 信息。

函数通过调用 sendMavOffboardAPI()函数进行信息发送。

2.8.4 sendMavOffboardAPI()

参数：type_mask=0,coordinate_frame=0,pos=[0,0,0],vel=[0,0,0],acc=[0,0,0],yaw=0,yawrate=0

type_mask：控制模式，例如本地位置控制，全球位置控制等

coordinate_frame：坐标系类型

pos：位置信息

vel：速度信息

acc：加速度信息

yaw：偏航角信息

yawrate：偏航角速率信息

函数根据 offMode 变量判定 offboard 模式，调用相应的函数发送信息。

2.8.5 SendPosNED()

参数：x=0,y=0,z=0,yaw=0

x：目标位置信息

y：目标位置信息

z：目标位置信息

yaw：偏航角

函数将把 offMode 变量赋值为 0，代表进入本地北东地坐标系位置控制。

2.8.6 endOffboard()

调用不需要参数，直接调用。

函数将会发送 PX4 退出 offboard 模式的指令，停止信息发送循环。

2.8.7 stopRun()

调用不需要参数，直接调用即可。

函数将会停止 mavlink 的信息监听循环。

2.8.8 initPointMassModel()

参数：intAlt=0,intState=[0,0,0]

intAlt：飞机初始高度

intState：飞机初始位置，初始偏航角

函数用以开启质点模型仿真。

2.8.9 PointMassModelLoop()

函数是质点模型的更新循环。将会由 initPointMassModel()函数调用。不需用户自启。

2.8.10 sendUE4PosNew()

参数：copterID=1,vehicleType=3,PosE=[0,0,0],AngEuler=[0,0,0],VelE=[0,0,0],PWMs=[0]*8,runnedTime=-1,windowID=-1

copterID：模拟飞机 ID

vehicleType：飞机样式

PosE：北东地坐标系位置

AngEuler：飞机姿态角

VelE：飞机速度

PWMs：转速

runnedTime：当前时间

windowID：窗口号

函数的作用是将传入的数据打包发送给 RflySim3D，用以创建一个新的 3D 模型或者更新模型的状态信息。

2.8.11 InitTrueDataLoop()

函数开启后将会初始化 UDP 真实数据监听循环。通过 30100 系列端口从 CopterSim 监听数据。

2.8.12 EndTrueDataLoop()

函数会关闭 UDP 真实数据监听循环。

2.8.13 endMavLoop()

函数和 stopRun()函数具有相同功能。将会关闭 20100 系列端口的数据监听。

2.8.14 SendMavCmdLong()

参数：command, param1=0, param2=0, param3=0, param4=0, param5=0, param6=0, param7=0

关于 command 和 param1~7 的定义，请参考：

https://mavlink.io/en/messages/common.html#COMMAND_LONG

https://mavlink.io/en/messages/common.html#MAV_CMD

函数的作用是，向 PX4 发送消息

2.8.15 sendMavOffboardCmd()

参数：type_mask,coordinate_frame, x, y, z, vx, vy, vz, afx, afy, afz, yaw, yaw_rate

type_mask：控制模式

coordinate_frame：坐标系信息

x：位置信息

y：位置信息

z：位置信息

vx：速度信息

vy：速度信息

vz：速度信息

afx：加速度信息

afy：加速度信息

afz：加速度信息

yaw：偏航角

yaw_rate：偏航角速率

函数用以向 PX4 发送 offboard 指令

2.8.16 sendUDPSimpData()

参数：ctrlMode,ctrls

ctrlMode：飞行模式

ctrls：控制参数。例如：如果飞行模式为起飞，则控制参数应该为[x, y, z, yaw]

函数将简易 UDP 消息发送到通信端口。

2.8.17 SendVelNED()

参数：vx=0,vy=0,vz=0,yawrate=0

vx：速度控制指令

vy：速度控制指令

vz：速度控制指令

yawrate：偏航角速率指令

函数用以将飞行模式切换为地球速度控制模式，切换 offboard 模式，发送消息的判别位，设定坐标系。

2.8.18 SendVelNEDNoYaw()

参数：vx,vy,vz

vx：速度控制指令

vy：速度控制指令

vz：速度控制指令

函数与 SendVelNED()具有相似功能。不同的是，函数不会设置偏航角速率指令，只发送速度控制指令。

2.8.19 SendVelFRD()

参数：vx=0,vy=0,vz=0,yawrate=0

vx：速度控制指令

vy：速度控制指令

vz：速度控制指令

yawrate：偏航角速率指令

函数将切换飞行模式为机体速度控制模式，offboard 模式为速度控制模式。切换工作坐标系为集体坐标系。坐标系正方向为前，右，下。

2.8.20 SendAttPX4()

参数：att=[0,0,0,0],thrust=0.5,CtrlFlag=0,AltFlg=0

att：根据 CtrlFlag 确定

CtrlFlag 0 : att 是三维向量，包含横滚角，俯仰角，偏航角，单位是角度

CtrlFlag 1 : att 是三维向量，包含横滚角，俯仰角，偏航角，单位是弧度

CtrlFlag 2 : att 是四维向量，包含四元数

CtrlFlag 3 : att 是三维向量，包含横滚角速率，俯仰角速率，偏航角速率，单位是 rad/s

CtrlFlag 4 : att 是三维向量，包含横滚角速率，俯仰角速率，偏航角速率，单位是 degree/s

thrust：根据 AltFlg 确定

AltFlg 0 ： 总推力，归在 0-1 范围内，（-1~1 适用于有反向推力的飞机）

AltFlg>0：期望高度

CtrlFlag：用于确定输入 att 定义的标志

AltFlg：用于确定输入 thrust 定义的标志

函数将切换飞机的飞行模式为姿态控制模式，切换 offboard 模式。将 FRD（前，右，下）坐标系下飞机目标姿态信息发送到 PX4

2.8.21 SendAccPX4()

参数：afx=0,afy=0,afz=0,yawValue=0,yawType=0,frameType=0

afx：加速度信息

afy：加速度信息

afz：加速度信息

yawValue：偏航信息

yawType：1，偏航角控制。2，偏航角速率控制

frameType：0，地北东地坐标系。1，机体 FRD 坐标系

函数将切换飞机的飞行模式为加速度控制模式，切换 offboard 模式。根据 frametype 切换坐标系。发送加速度控制信息到 PX4

2.8.22 SendVelNoYaw()

参数：vx,vy,vz

vx：速度控制指令

vy：速度控制指令

vz：速度控制指令

函数与 SendVelNEDNoYaw()功能相同，不同的是函数将当前坐标系设定为，机体 FRD 坐标系。

2.8.23 SendVelYawAlt()

参数：vel=10,yaw=6.28,alt=-100

vel：速度指令

yaw：偏航角指令

alt：高度指令

函数将设置飞行模式为速度高度偏航模式。切换 offboard 模式。切换坐标系为北东地坐标系。发送飞机的偏航信息，速度信息和高度信息给 PX4

2.8.24 SendPosGlobal()

参数：lat=0,lon=0,alt=0,yawValue=0,yawType=0

lat：纬度信息

lon：经度信息

alt：高度

yawvalue：偏航

yawType：0，无偏航。1，yawvalue 为偏航角。2，yawvalue 为偏航角速率

函数将切换飞行模式为地球位置控制模式，切换 offboard 模式，坐标系为北东地地球坐标系。发送飞机的位置控制信息和偏航信息给 PX4

2.8.25 SendPosNEDNoYaw()

参数：x=0,y=0,z=0,

x：目标位置信息

y：目标位置信息

z：目标位置信息

函数与 2.1.5 SendPosNED()功能类似。使用位置控制方式飞行。不同的是函数不会发送飞机的偏航信息。

2.8.26 SendPosFRD()

参数：x=0,y=0,z=0,yaw=0

x：目标位置信息

y：目标位置信息

z：目标位置信息

yaw：偏航信息

函数设定飞行模式为机体位置控制模式。设定 offboard 模式为本地北东地坐标系位置控制模式。设定工作坐标系为机体 FRD 坐标系，函数将位置控制指令和偏航角控制指令发送给 PX4.

2.8.27 SendPosFRDNoYaw()

参数：x=0,y=0,z=0

x：目标位置信息

y：目标位置信息

z：目标位置信息

函数的功能与 2.1.26 SendPosFRD()功能类似，不同的是函数不会发送飞机偏航角控制指令。

2.8.28 SendPosNEDExt()

参数：x=0,y=0,z=0,mode=3,isNED=True

x：目标位置信息

y：目标位置信息

z：目标位置信息

mode：0，滑翔模式。1，起飞模式。2，降落模式。3，旋翼机悬停模式，固定翼盘旋模式。4，固定翼飞机，无油门，无横滚/俯仰

isNED：True，本地北东地坐标系。False，机体坐标系北东地

函数用以发布飞机的位置控制。根据输入重新设定飞机的工作模式后发布位置控制指令。

2.8.29 enFixedWRWTO()

函数用以向在跑道的飞机发送允许起飞指令。函数不需参数调用。

2.8.30 SendCruiseSpeed()

参数：Speed=0

Speed：飞机的巡航速度。

函数用以改变飞机的巡航速度。

2.8.31 SendCruiseRadius()

参数：rad=0

rad：飞机的巡航半径。

函数用以修改飞机的巡航半径。

2.8.32 sendMavTakeOffLocal()

参数：xM=0,yM=0,zM=0,YawRad=0,PitchRad=0,AscendRate=2

xM：位置信息

yM：位置信息

zM：位置信息

YawRad：偏航角速率

PitchRad：俯仰角速率

AscendRate：升率

函数用以发送期望本地位置指令给飞机，使飞机起飞到期望位置。