项目名称：KubeMesh 高可用架构设计

项目导师：王杰章

申请人： 达益鑫

日期： 2022年5月22日

邮箱：2374087322@qq.com

本项目地址：

```
https://github.com/NKDYX/KubeEdgeMeshLearning
```

一．项目背景

✓ 1．项目任务的基本需求

> KubeEdge 基于 Kubernetes 构建，将云原生容器化应用程序编排能力延伸到了边缘。但是，在边缘计算 场景下，网络拓扑较为复杂，不同区域中的边缘节点往往网络不互通，并且应用之间流量的互通是业务的首要 需求，而 EdgeMesh 正是对此提供了一套解决方案。
>
> —官网项目描述

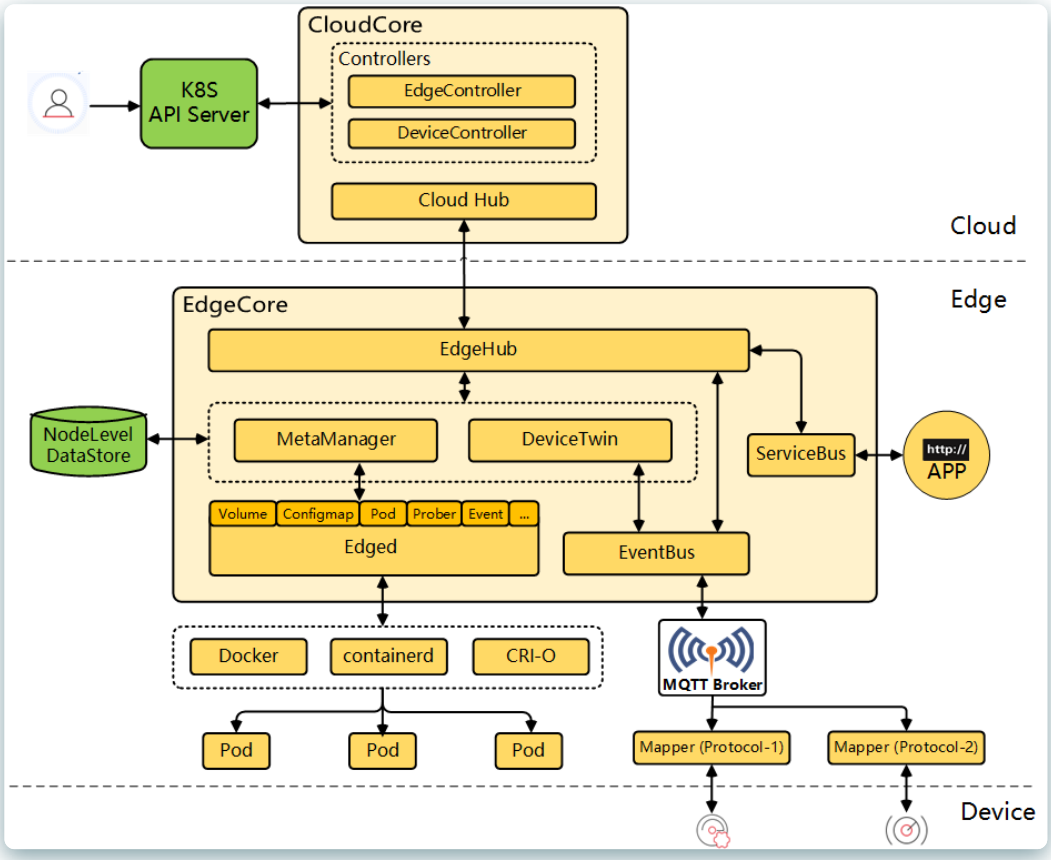
以下是Github Issue 当中所给的任务描述

> 1. 实现 EdgeMesh 的 aoturelay 和 autonat 机制
>
>   核心要求是：将EdgeMesh-Server 的功能合并到EdgeMesh-Agent当中，主要目的是能够尽可能地实现节点的分布式网络治理和通信，让Agent之间能够不通过中枢直接通信
>
> 2. 在EdgeMesh内实现多播dns的机制
>
>   确保即使外网无法连接，非协调节点也能正常工作；实现类似于节点容灾自治的功能。

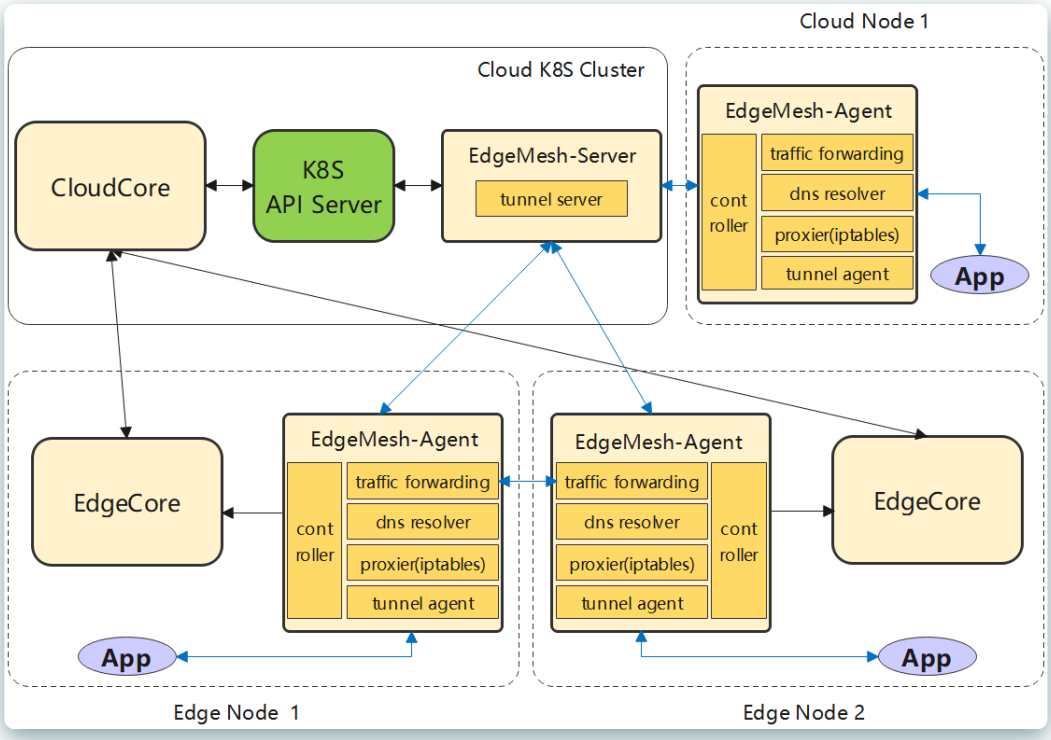
✓ 2. 项目相关技术细节

- 对于 KubeEdge 架构的理解和认识：
  - 在云端：CloudCoreService
    - CloudHub：一个web socket服务器，负责在云端观察变化，缓存并发送消息到EdgeHub。
    - EdgeController：一个扩展的 kubernetes 控制器，它管理边缘节点和 pod 元数据，以便可以将数据定位到特定的边缘节点。
    - DeviceController：一个扩展的 kubernetes 控制器，用于管理设备，以便设备元数据/状态数据可以在边缘和云之间同步。
  - 在边缘端：EdgeCoreService
    - EdgeHub：一个 Web 套接字客户端，负责与边缘计算的云服务交互（如 KubeEdge 架构中的边缘控制器）。 这包括将云端资源更新同步到边缘，以及向云端报告边缘端主机和设备状态的变化。
    - Edged：在边缘节点上运行并管理容器化应用程序的代理。
    - EventBus：一个 MQTT 客户端，用于与 MQTT 服务器（mosquitto）交互，为其他组件提供发布和订阅功能。
    - ServiceBus：与 HTTP 服务器（REST）交互的 HTTP 客户端，为云组件提供 HTTP 客户端功能，以访问在边缘运行的 HTTP 服务器。
    - DeviceTwin：负责存储设备状态并将设备状态同步到云端。 它还为应用程序提供查询接口。
    - MetaManager：edged 和 edgehub 之间的消息处理器。 它还负责在轻量级数据库（SQLite）中存储/检索元数据。

  在测试集群当中使用keadm部署，主要是对两个特应用进程做管理和启动，以下是架构图



- 对于EdgeMesh 架构的理解和认识：
  - EdgeMesh Server
    - Tunnel-Server：与 EdgeMesh-Agent 建立连接，协助打洞以及为 EdgeMesh-Agent 提供中继能力
  - EdgeMesh Agent
    - Proxier：负责配置内核的iptables规则，拦截对EdgeMesh进程的请求
    - DNS Resolver ：内置DNS解析器，将节点内的DNS请求解析成服务集群IP
    - Traffic Forwarding：基于Go-Chassis框架的流量转发模块，负责应用之间的流量转发
    - Controller：通过 KubeEdge 边缘端的 Local APIServer 能力获取元数据（如 Service、Endpoints、Pod 等）
    - Tunnel-Agent：基于LibP2P，使用中继和打孔提供跨子网通信的能力



✓ 3． 项目需要解决的问题

核心问题就是让 Agent 实现分布式网络，或者说将Server的功能并入到每一个节点的代理当中，让节点之间能够直接建立通讯，尽量摆脱中心化的网络架构和通信机制。

问题核心是如何能够将中心机能暂时边缘化，让节点 Agent 仅在特定或者说有限时间与中心节点创建双向连接，获得重要的集群信息

二． 解决方案Ⅰ：可行性分析（个人的介绍）

✓ 1． 技术背景知识

A． 项目经历：

在南开大学与天津市共同开发的云边端边缘智能系统的搭建当中主要负责了基于K8S建立的容器编排系统的搭建和再开发。

主要研究方向和内容涉及到边缘网络通信和架构设计

B. 个人相关知识储备

- kubernetes相关

  在实验室的项目服务器当中搭建完成多节点K8S基础架构，使用kubeadm作为核心构建道具，网络插件选用了calicaco，针对某些需要智能机制的服务做了网络Mesh的功能转发，主要是联邦学习的算法需求。

- Golang开发相关

  使用 Golang 开发过 K8S 当中的中心 Server 和边缘 Agent 监控插件，主要使用的是Websocket ，让Agent以推送的方式将的监控信息(Cadviser 以及 CRD 的监控 api 返回 json)传回 Server 存储并展示。

  对于 Mesh 的 Sever-Agent 架构有较多的实践经验，入手和深入探究有较多的基础。

  附件当中是本科毕设论文**基于容器编排架构Kubernetes的监控系统设计**

- 分布式网络架构相关

  EdgeGallary，SuperEdge以及KuboardK8S三类架构都有一些实践经验和深入的安装探索了解

- Linux 网络协议栈的部分知识

  以ucore和Linux作为实验学习对象，研究过Linux内核网络通信原理

C. 相关调研和实验

- 个人虚拟机实现两台机子的Server和Agent的搭建

  查看了CloudService 以及MeshSever的运行日志和运行nginx之后的信息，大体了解了如何实现信息和数据的传递，同时配置MeshServer和MeshAgent，但在一些地方还需要更多深入的测试和了解。

- 源代码初步阅读：

  因为安装的需要和看懂日志文件的查阅，目前最主要看了这几部分的代码，但仍有许多问题和想法想要进一步和老师交流。

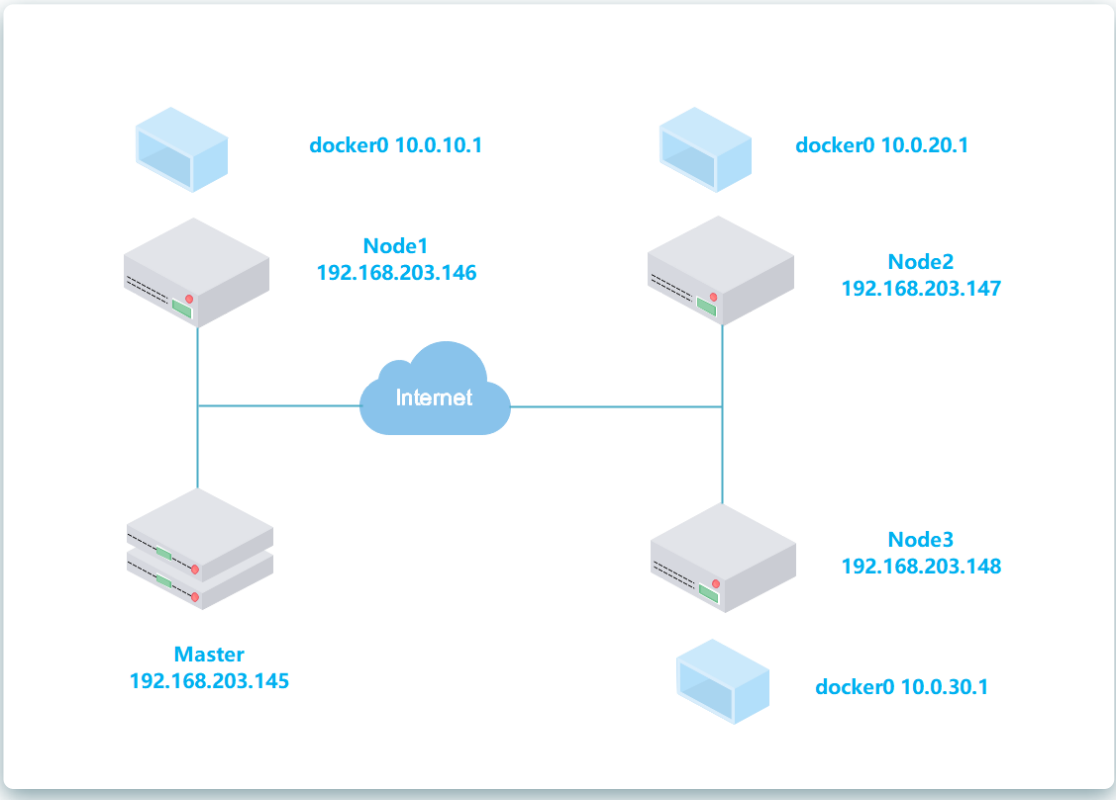
```
│   └─pkg
│       ├─chassis
│       ├─dns
│       ├─gateway
│       ├─proxy
│       └─tunnel
│           ├─config
│           ├─controller
│           └─proxy
│               └─pb
├─build
│   ├─agent
│   │   └─kubernetes
│   │       ├─edgemesh-agent
│   │       └─edgemesh-gateway
│   ├─crds
│   │   └─istio
│   ├─helm
│   │   ├─edgemesh
│   │   │   ├─charts
│   │   │   │   ├─agent
│   │   │   │   │   └─templates
│   │   │   │   └─server
│   │   │   │       └─templates
│   │   │   ├─crds
│   │   │   │   └─istio
│   │   │   └─templates
│   │   └─gateway
│   │       └─templates
│   └─server
│       └─edgemesh
```

✓ 2. 问题与研究重点分析

在 Kubernetes 的设计中，网络功能主要涉及容器到容器之间的直接通信，抽象到Pod到Pod之间的通信，Pod到Service之间的通信以及集群内部与外部组件之间的通信；这些功能是基于Linux网络机制来实现的虚拟化隔离技术。

就这点出发，将集群的网络通讯简化为一个较为简单的模型来做分析和理解。可以从以下两个图来分析集群IP地址的虚拟化转化流程和访问逻辑，实质上是做了一个类小型数据中心的三层交换机结构，或者说做了多级的路由表映射，将原本的网络结构变成了可控的虚拟网路结构，但不论如何实际到基建设施当中的数据都必须要指明实际网络当中的地址。

图一：两个节点之间的通信



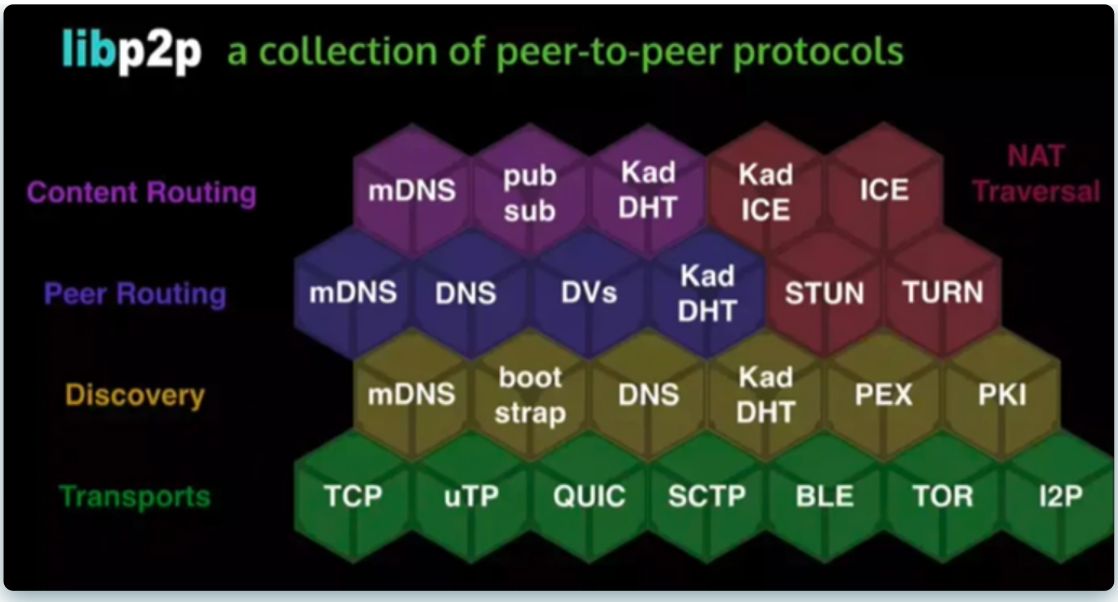
图二：一个控制节点三个工作节点的网络通信结构

由于这样的网络设计，虚拟地址就意味着可能无法保证其对应的真实地址记录是准确的，在Kubernetes集群当中，就可能会导致Pod IP漂移现象。为了防止PodIP对应不正确的问题，Kubernetes提出了service的概念，每个service都会对应到一组后端的一个应用实例上，通过提供恒定不变的Cluster IP来解决这个问题，同时也提供了proxy组件，基于控制面提供的一些信息维护Cluster IP到Pod IP的转换规则。举例来说就是只要用户访问www.baidu.com，集群就能够将其解析为当前能够提供百度浏览器服务的服务器地址，不论是访问数据库或者是实际发送探针信息。所以当client需要访问该服务时，一般只需要访问这个不变的Cluster IP即可。这个流量会经过本机的网络栈，被替换成了mysql后端实例的Pod IP，通过容器网络请求发送。

从指南当中可以初步了解EdgeMesh的工程逻辑是将边缘节点之间的通信分做了局域网内和跨局域网。如果是局域网内的通信，就直接通信因为地址是可以直接连接的没有经过上层的地址转化；如果是跨网的通信，EdgeMesh利用P2P打洞技术，成功时Agent就直接建立直连通道，否则通过Server中继转发。在这个机制的基础上要将Server的功能并入Agent并做到足够稳定的分布式连接，需要重点关注的是功能性上的问题：

- 一是Server的核心功能是对数据库的访问，对整个集群地址分配情况的掌握功能，因为他在Master节点或者说CloudServiceCore附近，能够实际获得注册节点以及服务的网络地址。如果无法直接获得连接对象Agent所在位置，在现有情况下就必须要到Server端中继转发；否则访问和建立连接的有可能就是一个不存在的地方。
- 二是Server的协调同步功能，比如在监控场景中：各个节点将自己的信息传给中心节点以不断同步当前集群的状态，实现负载均衡以及Pod的调度等等，如果要实现集群的理想情况分布式通信，最好当然每个节点都知道其他节点的状态以及地址，且发生变化时每一个节点都能够实现同步，但这样势必会导致巨大的资源消耗。
- 三是Server的封装管理功能，分布式网络理想的情况就是大家都在同一个数据平面来做通信，地址和服务都是对应的；但是在高负载且调用链条复杂的情况之下，就会存在容灾能力不足以及资源使用不均的问题。

libP2P是一个分布式P2P通讯高度模块化的工具包，从网络协议到应用层的通信工具，都有对应的模块能够使用，可以从下图看出他能够提供的服务模块



总而言之，无论如何都需要在边缘节点的代理当中记录相关的路由才能够保障网络通讯的正常进行，但是由于Kubernetes自己的网络设计，它将不同容器并入到同一个网络资源之下，这样一来对于应用开发者来说，所看到的容器和服务资源都是在一台机子上完成的；但也是这样不得不在节点之间设立Center来做通讯，而各个节点只有暂时的网络通信的地址，这样的设计是不能够满足对于分布式网络需求的；毕竟资源路由不是地址路由，这样的封装也必然导致了中心化的信息索取问题，使用穿透技术以及分布式网络管理的算法，这都是能够优化问题的思路。

关键是网络地址问题，集群内的地址以及对外的资源地址获取、存储、更新问题。

✓ 3. 已有的相关技术和解决方案

A. 业界相关

以下主要是P2P实现网络通讯以及内网穿透打洞等相关技术的一些学习结果，主要有中继、逆向连接、打洞（hole punching）等。

P2P技术属于覆盖层网络(Overlay Network)的范畴，在网络当中每个节点既可以从其他节点得到服务，也可以向其他节点提供服务，目前主流有三类结构：DHT结构 、树形结构 、网状结构。相关的一些研究和定义可以从这两篇综述当中来进一步探索：**[1]Peer to Peer Networks Management Survey** 以及 **[2]A survey of peer-to-peer content distribution technologies** 。在应用方面中兴也有对应的文件可以参考 **P2P技术原理及应用** 。

对于内网穿透技术相关，主要从**内网穿透工具的原理与开发实战** 开始学习和探索，基本了解了基础的内网穿透种类和实现的工具。

B. 科研界

主要是边缘节点自治以及直接通信的相关研究，尤其是 selfdriving-network 以及 区块链相关的研究。

对于selfdriving-network方面来说，工程上能够用到的应该是文章当中的某些思想，尤其是结构设计流程逻辑上，但是要实现的话可能就是思考的太多于复杂。相关的工作是：**[1] ISP-aware P2P 网络中协同缓存的效率研究** 提及了协同缓存的理念，这对于 Agent 局域缓存或者说共享缓存内容有一定的借鉴性； **[2] 学习路由非结构化 P2P 网络中的查询：在查询解析约束下实现吞吐量最优** 主要是在边缘 Agent 相互通信减少性能消耗来做借鉴的，不过其中的P2P网络非结构化的说法在Mesh当中是不太适合的，但这个算法可能在 Agent 的负载控制中有一些效果.我对于Mesh的结构理解是包括了一部分自组织网络的认识，实际上Mesh目前的结构就很符合自组织网络边缘架构的部分需求：能够在脱离中心节点的调控之下实现一定程度的自动化。所以也找了一些相关的工作：**[1] 自组织网络的系统思维办法** 和 **[2] 自驱动网络中的辅助时延和带宽敏感应用** 尤其是后一篇讲到的想法：“从缓冲区和 ping 延迟的角度检测应用程序的恶化，并在不需要任何显式信令的情况下应用补救措施来提高应用程序的性能。”

三. 解决方案II：方法设计（知识应用）

✓ 1. 可行性分析

首先需要将目标进一步的细化，就当前情况来说，EdgeMesh已经能够做到：对在线情况同一网络直接建立连接，不同网络打洞直连或server中继转发建立连接；对离线场景，控制面数据通过 KubeEdge 边云通道下发，实现了轻量的DNS服务器，在一定范围内请求闭环；而且开发工具主要是libp2p的模块，这部分封装了相当的多的网络协议工具，结合Linux的一些编程技术能够做到一定程度上的网络可控。

我目前的想法是对这些机制做进一步的优化，以下是几个可行的方案思考。

- 对在线场景，将创建中间连接Tunnel Server能力卸载到几个边缘节点，做多层的扩散和下发，这样节点就能够从其他边缘节点中获取直连对象的地址，或者是直接就能找到通信对象。拿网络的算法来做类比，就是当前主机在小范围局域网内做了一次广播就获得了通讯的路由路径，而并不需要走多层结构去根服务器找对方的实际地址。

- 对离线场景，增加Agent的本地缓存机制，或者是与周边其他Online节点连接的机制，这样对离线节点来说能够做到边缘局域网的自治和容灾，同时也能够使用分布式不同路径访问，将访问负载和网络断联的风险压力承载到网络结构上。

✓ 2．方法设计

这里讨论的方案是基于对项目和课题现阶段调研学习结果的思考和实验来提出的，可能后期阶段在导师更多地修正和本人更多的实验测试之后会采用更优质地方案。

A．对 Agent 增加部分Server功能：

当集群建立完成后，依据当前网络结构和资源状态选择部分 Agent 与 Server 同步 etcd 当中存储的节点信息，甚至是直接创建 Node-Server 作为一个虚拟的边缘中心服务，并将这几个节点标记为Local Master，同时注册周边所有Agent的地址，广播一次信息。当其他节点需要建立直连通道的时候就可以往着相邻可达网络中的这类节点而不用去Mesh Sever。同时这几个Agent之间可以构建直连网络层，自己维护一个独立的网络地址映射，这样只要每隔一段时间同Server同步一次，或者是设置特定的Trigger来触发同步更新的操作，就能够让这个虚拟的局域边缘达到相对的分布式管理。

B．对 Agent 增加缓存和定时容灾机制：

当无法连接MeshServer 或者是存在访问流量堵塞、资源消耗严重的情况，Agent使用本地缓存或者是直连周边节点服务，提供一个虚拟的向Server连接，而不是直接报错或者停止容器服务。当然这个还需要设置一个超时重连的机制，同时在Server和Agent都需要保证这个服务和地址在断联一段时间后直接重启。

首先是 Cache 机制的设置，参看之前提到的协同缓存的想法。我想为可以为 EdgeMesh-Agent 做一个功能性上的分类，在不影响他的轻量特性和功能的前提之下，给予他一个 Controller 能够生成和管理特定的文件，这样子就可以依据这样的标签将边缘节点按照 Agent 分做计算网络和存储网络，当面对断联的情况就能使用Cache机制暂时保证这部分局域网络的通信资源需求，同时找寻其他能够连接上 Master 的节点重新创建连接，而这些通讯应该是都是使用P2P技术来做的直接沟通。

其次是为了整个集群的稳定性和服务提供的鲁棒性，需要在一段容忍窗口期过后 Server 发起连接同时 Agent 也要确认自己是否还有可能连上，如果不能够满足两个条件的话就让主节点重新再建立新的资源对象并释放这部分边缘的资源，如果之后能够再连上可能还会存在缓存合并以及一些资源同步的问题。

四． 时间规划：阶段任务划分

✓ 1． 个人可用于项目开发的时间规划

这个暑期都会在学校，同时由于我所做的研究方向主要在分布式容器集群的优化管理和边缘网络的建设，这部分的工作与我的研究内容相辅相成，是会有充裕的时间来做相关内容的开发，主要的开发时间集中在6月中到8月初，可占用时间能够延长到9月，覆盖整个暑假的时间，是能够完成这项工程任务的。

✓ 2． 详细时间规划

5月10日 到 6月6日：项目及特性熟悉期

在本地电脑完成简易的KubeEdge，EdgeMesh搭建，同时尝试多个测试样例形式的容器及服务生成，通过调试手段和查看日志的方式来实际学习系统的运作方式和执行逻辑。

根据实验室师兄IFPS系统相关的一些研究，重点再libp2p上来做实验性质的Agent开发实验

6月6日--6月30日：项目开发预热期

深入学习和测试EdgeMesh的代码工程逻辑，熟悉KubeEdge的代码细节

学习和使用Libp2p工具包的使用

对分布式架构和内网穿透技术的各项研究做调研，同时和导师保持至少每周一次组会式讨论 ，更新本周的实验内容和结果，探讨系统架构的开发实验结果，推进开发流程

7月1日到9月30日： 项目开发期

一阶段： 基础功能开发研讨实验期

时间：7月1日到8月初，大致一个月的时间

目标：与导师研发出较为符合期望的分布式Agent架构和功能，代码上实现基础的功能和内容。

二阶段：功能基础实现和改进期

时间：8月初到8月中后期【大致计划在8月15日】，大致半个月的时间。

目标：开发完成基础课题要求的功能，能够在一般条件下正常运行并返回期望的结果，性能上不超过一定的容忍区间。

三阶段：中期功能检验和修改期

时间：8月中旬【大致在16日或者是20日】，大致花费不到一周的时间。

目标：交予老师和项目组代码，主要方式是自己的github代码仓库的形式，确认功能的实现和基础的鲁棒性。

四阶段：功能检验和查bug，系统调优期

时间：8月中旬到项目结束。

目标：依据修改意见对系统代码做修改，依据自己测试以及github的issue反馈修改bug，依据系统性能的调优标准优化程序运行逻辑。

五． 相关参考资料以及个人仓库索引：

- 个人**github**：将会详细记录本次课题从调研学习到具体功能实现的各项细节和问题，同步核心的Mesh Agent代码
- 项目主要参考文件：
  - **https://edgemesh.netlify.app/zh/advanced/architecture.html#%E6%A6%82%E8%A7%88**
  - **https://kubeedge.io/zh/docs/**
  - **https://blog.csdn.net/weixin_41033724/article/details/122954940**