系统简介

在作业四中，我们使用了 Java Springboot 来完成分布式组成员服务系统（下面简称服务系统），在 **7** 台服务器上运行我们的服务系统程序，并使用 Vue3 来完成服务系统的可视化展示过程。（详细服务器列表与前后端接口请参见附录）

PS：演示视频在附件内，但是由于有两三个服务器用的阿里云的学生免费版，所以网络效果没有付费的好，导致它们本身发包收包率都很低

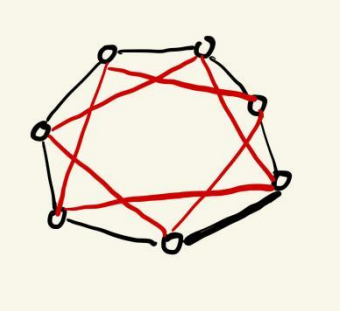
系统原理

1. 拓扑结构

在七个节点的分布式系统中，我们选择了一个在$N/2$的服务宕机的情况下，仍然能够稳定工作的拓扑网络。（当然此网络也可以运用与节点数量更多的分布式系统中）拓扑结构如下所示：

在$N > 4$ 的情况下，每个节点i先组成环，然后再与自身前一节点的一节点相连。

形式化表达为i节点与$(i-2+N)\%N$、$(i-1+N)\%N$、$(i+1)\%N$、$(i+2)\%N$相连



在这种拓扑结构中，每个组成员都会有 4 个联系人，这样会形成一个联系人列表。后续的 Gossip 协议和心跳检测协议都会在联系人列表中选择联系人进行消息的传递。

2. Gossip 协议

Gossip 协议有两种方式，一种是 Gossip Rumor Mongering，这种类型地协议会把接收到的更新内容（在本系统中为一条组成员信息）随机地选择节点传播出去；另一种是 Gossip Anti-Entrophy，这种类型的协议会把自身的所有数据进行传播，而不只是更新的数据。

在我们的系统中，这两种方法都有使用。以下是详细介绍。

Gossip Anti-Entropy 关注信息的准确性和最终的一致性。在 Gossip Anti-Entropy 方法中，节点会把自身所有的组成员列表随机发送给其余联系人节点。节点使用 Gossip Anti-Entropy 方法的前提是 1. 自身为 Introducer 时，向新加入的节点发送所有的组成员信息。2. 每个一段时间（1 秒）随机选择联系人来发送自己的所有的组成员信息，接收到信息的联系人如果有消息的更新，则后续使用 Gossip Rumor Mongering 方法对更新的消息进行传播。

在 **Gossip Rumor Mongering** 方法中，节点随机地选择其他几个联系人节点，并向它们发送信息，在此系统中，节点使用 Gossip Rumor Mongering 方法的前提是 1. 自己即将离开组 2.收到的信息中有比自身维护的信息更新的信息。

节点发送给接收信息的节点再随机地选择其他节点传播，这样信息就能迅速在整个网络

中扩散开来。

3. HeartBeat 心跳检测协议

3.1 心跳发送

定期发送：每个节点会定期（每隔 1s）向其他节点发送心跳信号。节点的选择是从当前节点的联系人中随机选择，每次心跳会随机选择 4 次并发送。

轻量级信号：心跳信息是轻量级的，只包含发送服务器的 ip 地址。

3.2 心跳接收和监测

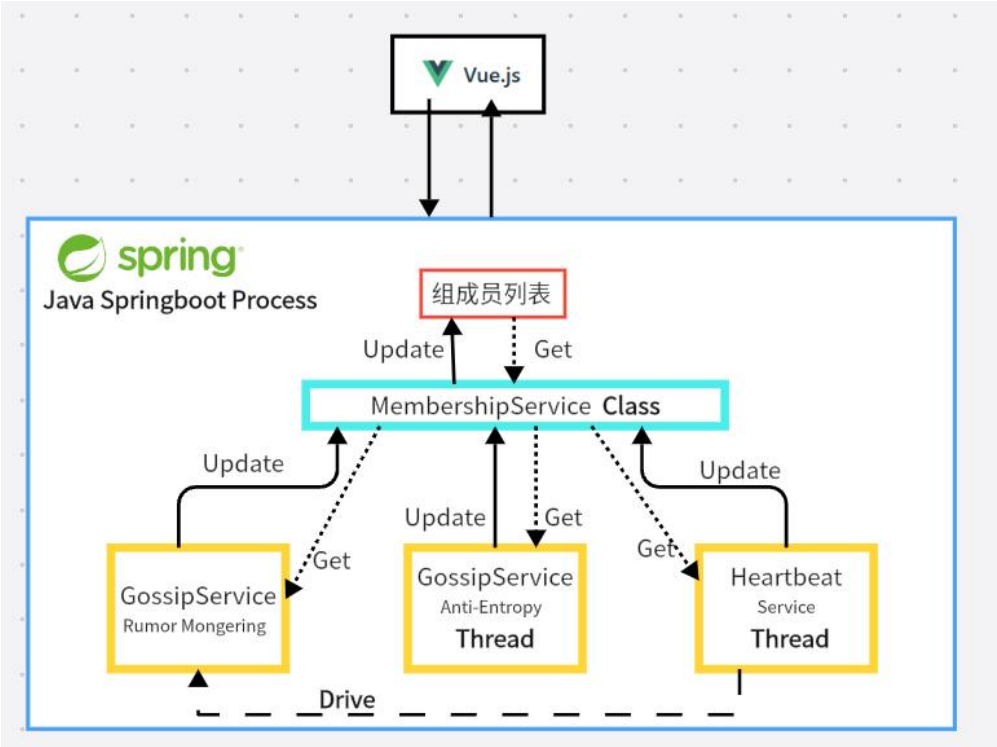
接收确认：其他节点接收到心跳后，即可确认该服务器仍然活跃。如果服务器在被确认为宕机后，又重新收到心跳，则认为是网络原因造成的消息不可达，这是把服务器更新为活跃。

超时检测：节点会监控从其他节点接收到的心跳。如果在预定的超时时间内没有收到某个节点的心跳，该节点会被认为以及宕机。

状态更新：一旦节点被判断为失效，系统会更新其状态，可能将其标记为离线或故障。这个信息可以通过 Gossip 协议传播给网络中的其他节点。

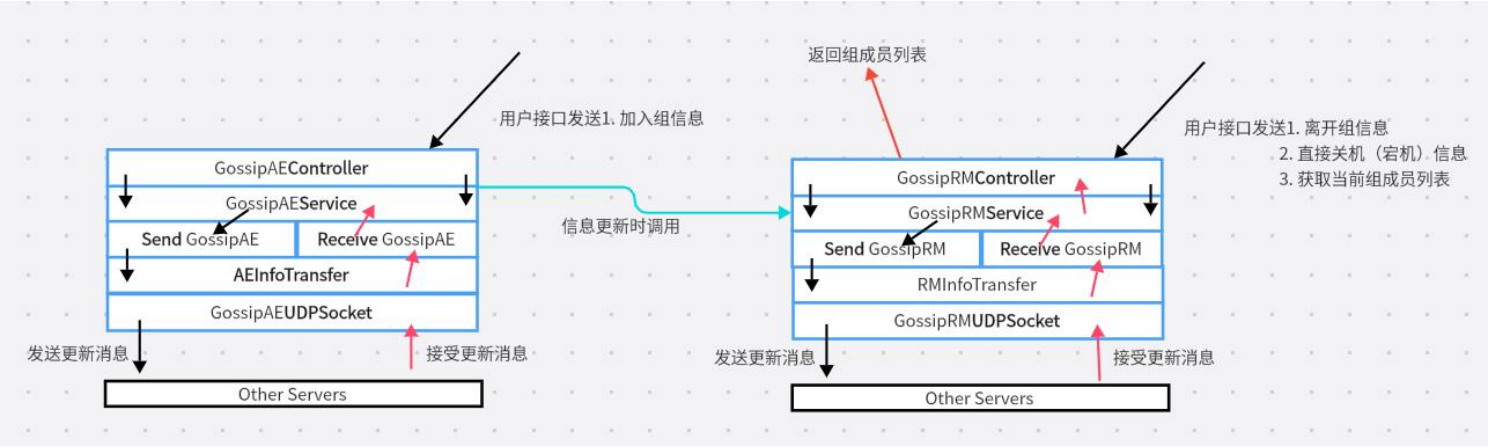
系统架构

在后端程序内，我们设计了两个主要的额外的主动发送消息的线程，然后通过建立一个线程安全的 CopyOnWriteList，让两个定时发送消息线程共同维护组成员列表。在服务系统中运行的连两个线程分别是心跳检测线程、Gossip Anti-Entropy 服务线程。这两个线程同时工作，然后驱动 Gossip Rumor Mongering 服务，协作更新组成员列表，将会保证分布式系统中的组成员列表能够一致地维护当前最新地各个服务器状态列表。



各部分设计

1. Gossip 设计



Gossip 协议是本系统最重要的内容。其中的 RM 与 AE 简写分别代表 Rumor Mongering 与 Anti-Entropy。RM 与 AE 的不同点请参见系统原理。我们在系统原理基础上分层将 Gossip 设计为了 5 个层次。

Controller 层负责接收用户发出的信号，在此次作业中为 join、leave,以及模拟的开机关机（宕机）。

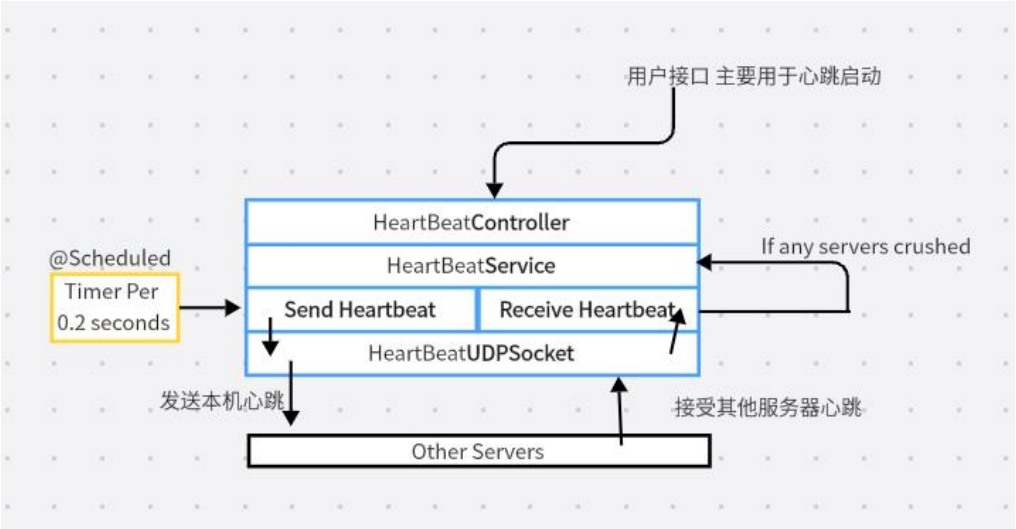
Service 层首先负责处理用户发出的信息，根据用户的请求有不同的操作；其次 Service 层会分析接收到的下层的数据，来判断是否需要更新自身的表单，如有更新，则使用 GossipRM 进行传播。（在图中为蓝色的线，无论是 AE、RM 还是 heartbeat 导致的更新，都将使用 RM 进行更新传播）。

Send 与 Receive 层负责封装消息，并启动传输。

InfoTransfer 层负责将消息封装为平台无关的模式，或者将接收到的信息转换成可使用的成员信息。这里我们选用的是平台无关模式为 json 格式。**消息格式请参见附录**

UDP 层负责数据的传输与接收。

2. **心跳检测设计**



心跳检测主要由一个定时的任务进行驱动。为了使系统更加统一，所以采取了与 Gossip 协议相同的架构。由于心跳检测本身使用了字符串，已经与平台无关，所以这里没有再进行冗余设计。其余具体功能可参见上方 Gossip 设计。

系统功能

1. **动态显示组成员列表：**在本系统中，组成员列表可以通过 Controller 中的 API 进行获取，同时，根据系统原理，组成员列表会在有变化时自动更新。
2. **组服务容错机制：**在本系统中，丢包率的模拟是在 UDP 层直接控制的，通过前端用户输入的丢包率来控制系统的丢包率大小。同时，为了确保系统容错，我们还使用了 AE 传播算法，在较长的时间间隔内向周围发送节点信息，以保持一致。
3. **日志查询：**在本系统中，日志存放在服务器本地，可以通过前端直接进行查询。
4. **可扩展性：**本系统已经在 3 台、7 台服务器的情况下对系统进行了测试。在更多服务器的情况下，只需要在服务器运行本系统程序，知道服务器 ip，即可将更多服务器纳入组。
5. **低延迟性：**在理想状况下，系统的延迟可以达到很低的水平，因为我们租用的是性能比较高的服务器，但是在最后提高性能测试时，我们发现在 UDP 端口处心跳信号产生了冲突。因为 UDP 本身不处理冲突，所以我们猜测在高心跳的情况下，数据冲突会造成心跳丢失，所以我们只能将心跳间隔保持在一个较高的水平。
6. **组成员列表的锁控制：**在本系统中，组成员列表是通过 CopyOnWriteList 控制的，所以组成员列表是线程安全的，并且我们设计的数据结构（4 连通）允许在多余 1 个、少于 N/2 服务器（3 个服务器宕机）发生故障的情况下仍然能够保证系统的整体性。
7. **组成员在多宕机时的性能情况：**在多宕机的情况下，我们发现消息传输的数量进行了增长，这是因为在多个服务器宕机的情况下，每个服务器都会产生一个自己的心跳判断，从而导致发送的 Gossip 消息变得更多。

附录

1. 服务器列表：

（已全天租用的服务器）

123.206.121.97

47.99.166.45

43.142.91.204

（随开随关的服务器，有可能 IP 会变化）

121.43.98.159

8.136.118.46

8.136.125.35

8.136.123.0

由于经费不是特别充足，如有需要，可以联系 2154177@tongji.edu.cn 打开

2. 消息格式：

1. GossipAEDTO 消息格式说明:

该消息格式是一个 Json 字符串，表示一个分布式系统中一组成员的信息，每个成员由 nodeIdentifier 标识。timestamp 表示信息收集的时间，source 表示消息的来源。例子如下：

```
{
    "timeStamp": "2024-01-01 12:00:00",
    "source": "127.1.1.1",
    "memberList": [
      {
        "nodeIdentifier": "127.0.0.1",
        "nodeStatus": "ACTIVE",
        "timeStamp": "2021-10-1 05:25:55"
      },
      {
        "nodeIdentifier": "127.0.0.2",
        "nodeStatus": "DEPARTED",
        "timeStamp": "2025-5-6 05:25:55"
      },
      {
        "nodeIdentifier": "127.0.0.3",
        "nodeStatus": "ACTIVE",
        "timeStamp": "2024-5-1 15:25:55"
      }
    ]
}
```

- timeStamp: 消息生成的时间戳。
- source: 生成消息的来源的 IP 地址或标识。
- memberList: 包含有关单个成员的信息的数组。
- nodeIdentifier: 成员节点的唯一标识符。
- nodeStatus: 成员节点的状态（例如“ACTIVE”或“DEPARTED”）。
- timeStamp: 表示成员节点的最后已知状态的时间戳。

2. GossipRMDTO 消息格式说明:

该消息格式是一个字符串，表示分布式系统中特定成员节点的更新信息，包括其 nodeIdentifier、nodeStatus、timestamp 和信息来源。

```
{
    "nodeIdentifier": "127.0.0.1",
    "nodeStatus": "ACTIVE",
    "timeStamp": "2023-10-01 05:54:54",
    "source": "127.0.0.2"
}
```

- nodeIdentifier: 成员节点的唯一标识符。
- nodeStatus: 成员节点的状态（例如“ACTIVE”或“DEPARTED”）。
- timeStamp: 表示成员节点的最后已知状态的时间戳。
- source: 提供此信息的来源的 IP 地址或标识。

3. 接口详情（IP 代表服务器 IP 地址）

主要功能接口

1.1 选择服务器作为 Introducer **(GET)**

http://**IP**:8001/final2/set-introducer?isIntroducer=true

用途：在选择界面时确认一个服务器作为 Introducer，来控制组成员的加入信息。

1.2 启动服务器**(POST　TEXT 形式 IP 地址)**

http://IPMember:8001/config/poweron

用途：将服务器进行开机，并且向参数中的 IP 地址（Introducer）发送请求加入的信息。

1.3 Introducer 接受组成员加入信息，组成员加入组**(GET)**

http://IPIntroducer:8001/final2/join-group?ip=IPMember

用途：在各个组成员后端中发送，Introducer 接收成功后会将 IPMemeber 加入组成员列表。

1.4 服务器启动心跳检测**(GET　参数：建立连接的服务器个数)**

http://IP:8001/start-heartbeat?serverNum=

用途：仅在首次启动时使用。在服务器第一次全部开机，并加入组之后，给每一个服务器发送心跳检测启动的命令。该命令只需要在服务器首次建立连接时使用，后续不需要在使用。

1.5 服务器离开组**(GET)**

http://IP:8001/final2/leave-group

用途：使得此 IP 服务器离开当前的服务组。

1.6 模拟服务器宕机**(GET)**

http://IP:8001/final2/crush

用途：模拟此 IP 服务器直接关机（即发生故障），直接退出组服务。

1.7 服务器丢包率设置**(GET)**

http://IP:8001/loss-rate

用途：手动设置服务器的丢包率大小。

辅助功能接口

1.8 查询此 IP 服务器维护的组成员列表**(GET)**

http:/IP:8001/final2/show-members

用途：获取服务器实时的组成员列表，用于观察和分析。

1.9 重启此 IP 服务系统**(POST)**

http://IP:8001/restart

用途：仅在确认重启后使用，此操作会丢失此 IP 服务器服务程序的所有信息，重新启动一个新的服务器服务程序。