## 新要求在最下面[新要求（2023.2）：](#_新要求（2023.2）：)

### 构建一个编码模拟的网络架构

**（旧）要求:**

1、整体架构(图结构，不同层有连接，同层也有连接（路由直接交换信息，广播的时候用），所以不能用树结构):

网络架构一共有N层，第1层是IOT层, 2到(N-1) 层为Router层，N为Cloud层，每一层有m个结点(设备)

PS:数据产生后，就通过最短路径向上路由. (最短路由,数据结构上有)

2、IOT层:结点数可修改，每个结点都可以模拟产生数据，每个结点产生的数据量、产生速度可修改，数据传送到上一层路由(一个 IOT结点产生的数据只能传给一个Router层的结点，一个Router层的结点可以处理多个IOT层结点的数据)，产生数据是异步的即产生数据后不用等待数据处理完再产生，而是模拟一直产生数据。

3、Router 层:结点数可修改,每层上的结点都可以处理数据(模拟就是减少一部 分数据量，同层处理能力相同，越往上处理能力越强)，处理数据的能力可以修改，剩余的数据传到上层来处理。

4、Cloud 层:处理剩余的数据。

5、数据:每个数据有数据ID、类别ID,每层进行处理后,数据量就会减少.

**实现的效果:**

1.如果IoT devices产生数据量多,就会有些数据在上层处理.越多,就有可能到cloud处理了，越少,就会在边缘层处理完了.

2、目前先不用管数据到哪里，先进行模拟数据往上传，数据被处理就行。

3、网络图最后把哪个数据从哪一层的哪个设备到下一层的设备的数据传递过程展示出来(最好用图示展示)，记录每一层的结点上处理了多少数据、处理了什么类别的数据。

## 数据传输所需要的算法

（MDAR）算法解释

1. 从IOT层产生的数据向上传送时，让同类数据走同一条路由。具体实现过程：

在开始的时候, 所有devices设备类数据到达cloud的路由路数是. cloud向外说, 其到达cloud的路由跳数为0, 距离cloud一跳的devices听到后, 将自己到达cloud的跳数设为1, 然后, 同样向外广播, 这样就形成了下图(a)所示和初始路由. 这时如果devices 5产生的一个类的数据, 这时数据沿最短路由路由到cloud后, 此路由路径上的所有devices都将到达cloud的跳数设为0. 这样将跳数为0, 再向外广播, 其它devices就更新自己的路由表. 这时此路由路径上到达cloud的类数据的跳数为0, 因而, 其附近折devices有类的数据时, 就会走这条路路由. 如 下图(c)所示. 这样就能够促使同类的数据汇集到一起路由。

（PS：最短路由的生成按照构建最优路径路由表，采用链路状态路由算法进行构造，动态更新路由表，找到最优路由表。）

(a) (b) (c)



1. 在数据走同一条路由时，让每层的结点负载均衡，具体实现为：

每层结点处理数据的时候可能会存在：部分节点data处理非常多，而有些节点处理的非常小。这样，造成有时data在节点停留很长时间，而同时，部分节点没有数据来处理。这样负载不均衡。

我的处理方法是：如果某个节点负载非常重，则向下层游发送消息，告诉其负载太重了，让数据提前绕路传送，别给我了。然后，向上层发消息，问有没有负载轻的，我负载太重了，你们如果有负载轻的，告诉我一下，我就有些data不处理，直接发原始数据，让你们帮我处理，这样就能够做负载均衡。

1. （ODAS）相同数据进行融合，具体实现为：

（ODAS）可以实现这样的效果：如果数据之间是有相关性的话，就能够使得aggregation之后的数据量变的很小。如果数据之间相关性弱，则aggregation之后的数据量没减少多少。

## 输出结果

1. **数据相关解释：**

**原生数据：只在IOT层产生。**

**新数据=服务数据：就是最底层上传到边缘之后，经过处理了就是服务数据**

**旧数据：是被负载均衡转发的服务数据。**

1. 如何被处理（处理的依据和结果）：处理就相当于减少一部分原生数据量（减少的是“结点所具有的处理数据的能力”）同时产生“服务数据（新数据）”（“服务数据”产生的数量是：减少的原生数据\*处理率）。

B、什么时候返回：原生数据被全部变成服务数据（新数据）返回按照返回时候的速度。

C、如何返回：服务数据（新数据）按照最短路由算法返回到**产生数据的那个结点**

D、数据汇聚：如果（ODAS）开关开着的话，是让同类数据走相同路由，原生数据的数据量会减少（减少的数量=相似数据的相似度范围\*0.5\*原生数据总量），**比如有数据a和数据b，每个数据分配了一个相似数据的相似度（相似度范围里的随机一个值），然后他们两都是第1类数据（属于相同数据的范畴了），当他们两都传到第二层的某个结点上了，他们两再往上上传的时候就会把相同部分进行整合，需要往上上传的数据量就减少了，减少的多少就是a、b他们各自的原生数据量\*相似数据的相似度\*0.5**

1. **接口修改方便，可以直观进行修改，单独放到一起可以统一设置，加上中文备注：**

A.数据产生速度设置

B.数据上传的速度设置（就是数据从一个结点到另一个结点的速度，一个结点和另一个结点是有距离的，**比如距离是30，速度如果是3的话，从一个结点到另一个结点时间就是10了**）

C.数据处理率（大于0小于1的数），

D.结点处理数据的能力（设置一个范围，到时候随机给结点分配，高层分配的一定比底层的大，层数越高处理能力越强）

E.产生的总的原生数据量设置

F.MDAR算法开关（开的话原生数据就按照【相同数据走相同路由】算法走，关掉的话就原生数据产生后按照最短路径走）

G.ODAS开关（开的话同类原生数据还会聚合减少，关的话正常进行处理减少数据量，同类也不会汇聚数据，**这个就是说是不是要启用那个汇聚的效果，只要是为了进行对比，不开启相同的数据就不会再减少数据量了**）

H.（如果ODAS是关的话就不用考虑这两个变量了，但需要设置这两个变量） 数据的种类数（**这个是这个网络要产生多少种数据，如果没有汇聚效果其实种类多少都不影响结果，所以如果是关着那就不用管了**）

I.（如果ODAS是关的话就不用考虑这两个变量了，但需要设置这两个变量） 相似数据的相似度范围（大于0小于1的数，**这个前面说他的作用了，到时候每个数据会从这个范围内获取一个值**）

1. **结果输出（ps：这些都是所有结点的总和）**
2. 各层一共处理了多少原生数据（通过ODAS汇聚减少的不算在这个里面）
3. 各层一共传输了多少数据量（所有结点的：上层传过来的原生数据+服务数据+别的层 负载均衡 下放来的数据量（旧数据））

**（负载均衡的数据实际上也就是旧数据，但是需要记录一下一共给别的层传了多少，记录一个值）**

H.别的层负载均衡下放来的数据量一共有多少（旧数据有多少）

G.各层一共汇聚减少了多少原生数据量（ODAS）

C.各层一共计算数据花了多少时间（原生数据量=0的时候花了多长时间）

I.数据传输一共花了多长时间（数据光上传的时间）

G.数据返回一共花了多长时间

D.数据被处理完一共花了多长时间（数据处理时间+数据上传时间+数据返回时间（返回时间是数据按照最短路由算法回来所话费的时间））

E.各层一共数据传输的总距离（从产生到返回**产生数据的那个结点的距离**）

**关于计算时间和距离：这个总时间可以一直往上累加，不用去记录每一条，比如有个num，最开始是0.然后有多少加上去多少，不用记录下每一条是多少**

PS：结点之间传输展示不是很重要。重要的是要将处理后的结果输出出来。具体需要的输出结果，最好中文说明解释下代码怎么实现**数据聚合和负载均衡那块**

# 新要求（2023.2）：

说明：上面这个网络架构是适用于一般情况下的网络架构，当出现突发流量时会出现节点缓冲区爆满（虽然负载均衡可以缓解，但是在短时间内迁移热数据不是一个很好的解决策略），导致网络堵塞。所以我们需要在面对“突发流量”时，这个网络架构将同时采用第二种网络的策略。

在进行第一种策略的同时，增加第二种数据处理策略（两个算法组成）：

**实现效果：**

**一、Burst data integration routing（Bdir）算法：**  
1、网络流量监控：Bdir算法通过监控数据包到达率来判断网络是否出现突发流量，当到达率超过阈值时，认为出现了突发流量。

2、数据包分流：当检测到突发流量时，Bdir算法会对数据包进行分流，将部分数据包从原节点转移至Assistant节点进行处理。数据包的分流依据包括该层节点的数据包大小和处理时间因素。

3、Assistant节点选择：Bdir算法会评估网络中所有可用Assistant节点的处理能力，并选择适合处理转移数据包的Assistant节点。

4、数据包转移：被选择的数据包会被转移到相应的Assistant节点进行处理。为了确保Assistant节点有足够的处理资源来处理数据包，Bdir算法会使用拥塞控制机制来限制转移速率。

5、数据包处理：Assistant节点会对转移的数据包进行处理，处理效果是将多个同类型小的数据包合成大的数据包（处理大的数据包相对于小的数据包来说，具有更好的性能和效率），并将处理结果返回给原节点，处理完成的机制也是数据包达到一定大小或时间长短。

**二、Assistant共同加速（Assistant Efficient Routing）AER算法**

6、性能监控：Bdir算法不断监控网络中数据包到达率，通过之前设计的负载均衡算法调整转移速率和Assistant节点数量，以保持高性能并防止拥塞问题的发生。

7、某一个Assistant节点对于所收到的相同数据包的量还是有限，或者面对庞大的数据量并不是那么容易进行结合，所以有一个Assistant共同加速AER算法：利用多个Assistant节点来进行共同打包，原理是：数据量达到Assistant节点加速阈值（这个阈值是根据该层数据量进行动态设置的），其他Assistant节点便动态生成，共同处理同一个Assistant节点上的数据，各个Assistant节点进行连接，将数据进行动态分配。

**增加接口：接口修改方便，可以直观进行修改，单独放到一起可以统一设置，加上中文备注：**

A、Bdir接口开关

B、AER接口开关（可以单独使用Bdir）

C、突发流量阈值

D、数据分流数据包大小

E、数据分流时间阈值

F、Assistant节点的数量

G、小数据包组装完成时大小

H、最长组装时间

I、共同加速阈值基础值

**输出结果：**

**在原来输出结果的基础上新增：**

1. 一共打包的数据量
2. 打包所花费的时间
3. 所调用的Assistant节点的数量