Titan scheduler

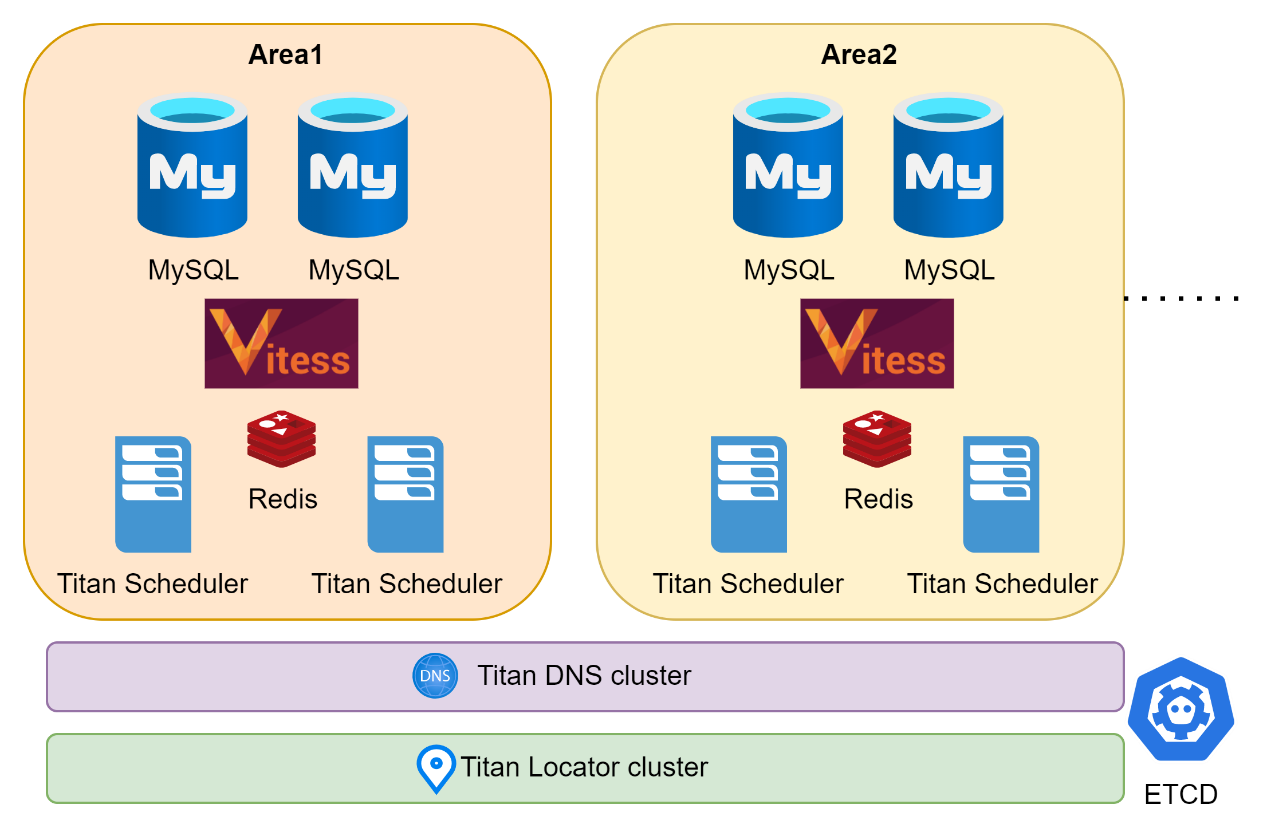
# 概述

作为titan网络的中枢，scheduler主要的职责如下：

* 管理网络中的节点，titan网络中的节点都需要连接某个scheduler，以加入titan网络。
* 管理网络中的assets，scheduler提供接口，管理者通过这些接口，操作网络中的assets对象，例如：新建、删除、查询或修改assets对象。
* 响应用户拉取assets数据的请求，为用户分配拥有该assets的资源节点，以便用户可以从这些节点拉取assets数据。
* 评估节点的贡献，生成所有节点的贡献记录存档carfile，并把存档carfile的CID发布到FVM合约上。Scheduler按照设定的周期对网络中的节点，进行存储和带宽抽查，记录抽查结果；节点为用户提供下载服务后，scheduler根据节点和用户提交的有效的工作量证明，记录节点工作量；抽查结果以及节点工作量都是scheduler评估节点贡献的依据。
* 根据节点的贡献对节点发送奖励。Scheduler根据titan网络设定的奖励周期和奖励规则，按照节点的贡献，定期地为节点发送奖励，生成奖励结果存档carfile，存档carfile的CID发布到FVM合约上。

# 部署

Titan网络的部署示意图：



Titan网络是按照area部署的。每一个area，都有一个titan area网络。这些titan area网络组成整个titan网络。

一般地，一个area对应一个city。Titan area网络中的所有节点，均运行在该city的城域以太网，或者该city的城域网内。因此，节点之间传递数据是非常快的，因为他们通讯不需要跨越国家骨干网。

如图中所示，每个area，部署多个titan scheduler服务器，推荐至少部署两个，以避免单个titan scheduler失效时导致该area的titan网络不再可用。

如图中所示，每个area，部署一套Vitess分布式MySQL数据库系统，多个titan scheduler 实例共用同一套Vitess数据库系统，因此这些scheduler服务器实例看到的数据是一致的。

如图中所示，每个area，可考虑部署一套Redis cluster，用于加速数据的读取。

图中的Titan locator，负责发现所有的titan scheduler服务器，并在收到用户请求时，把请求路由到距离该用户最近的scheduler服务器。每个titan scheduler启动完成后，都会向titan 网络的ETCD服务器注册，因此titan locator能够通过ETCD服务发现所有的scheduler服务器以及这些scheduler所服务的titan area网络。

图中的Titan DNS，负责把节点网络名字映射到节点的IP地址。每一个titan网络中的节点，都会获得一个网络名字，客户端需要titan DNS服务把节点的网络名字解析为节点的IP地址，才能访问节点。

# 数据库

## MySQL

Titan 网络有许多数据，例如节点信息、Assets信息等等，需要持久化的存储，我们选择MySQL作为持久化数据库。

## Vitess: 分布式MySQL

考虑到数据规模，我们需要分布式的MySQL数据库；考虑到数据的可靠性，我们需要Master-Slave架构，避免单点故障导致数据丢失。我们选择Vitess作为MySQL的分布式解决方案，Vitess是用于部署、扩展和管理大型MySQL实例集群的数据库解决方案，Vitess集MySQL数据库的很多重要特性和NoSQL数据库的可扩展性于一体。其内建拆分分片功能使我们能够对MySQL数据库集群无限水平扩展，同时无需为应用添加分片逻辑。

Vitess具有以下优点：

* 性能

Vitess自动重写对数据库性能有损害的查询。它还使用缓存机制来调解查询，并防止重复查询同时到达数据库。

* 管理

Vitess可以支持自动处理主故障转移和备份等功能。它使用分布式元数据服务来跟踪和管理服务器，使应用程序无需关心数据库拓扑变化。

* 连接池

Vitess避免了MySQL连接的高内存开销。 Vitess服务器轻松地一次处理数千个连接。

* 分片管理

MySQL本身并不提供拆分分片功能，但是业务数据量增大到一定程度，我们是需要增加集群的。Vitess提供在线拆分功能，只需要很少的时间就完成新集群的切换，无需在应用程序中添加任何拆分逻辑。

* 工作流

Vitess会跟踪有关集群配置的所有元数据，以便集群拓扑始终是最新的，对不同的客户端保持一致。

## 内存数据库

Titan 网络有一些数据需要频繁的读取，例如，carfile的CID，以及拥有carfile的edge节点列表等等，这些数据需要在每一次用户从titan网络下载数据被读取，如果每次都从MySQL读取这些数据，请求量大的时候，从MySQL读取就可能成为瓶颈。

因此，我们可以把这些频繁读取的数据，放到内存数据库，例如Redis中，加快读取速度。同时，通过LRU原则，把很久没被读取的数据从Redis逐出，避免Redis的内存占用过高。

# 核心组件

## 节点管理

所有节点都需要通过scheduler加入titan 网络。节点启动后与scheduler建立通讯连接，并完成节点登录流程。

Scheduler通过调用节点的JSON-RPC API接口，管理节点，以及节点上的资源：存储、带宽、计算等等。

另一边，节点通过调用scheduler的JSON-RPC API接口，向scheduler报告节点的状态信息、运行事件等等。

### 节点类型

Scheduler管理三种类型的节点，分别是validator节点，candidate节点，以及edge节点。其中，validator节点和candidate节点属于L1节点，edge节点属于L2节点。

三种节点和scheduler的层次关系如下图所示：



#### Candidate节点

Candidate节点是L1节点，和edge节点比较起来，它数量少，但是要求的具体的资源高很多，例如很大的存储和带宽，以及很好的计算能力。

L1节点需要负责执行titan网络中的大部分计算需求，例如运行可信容器。这需要L1节点具有较强的CPU和较大的内存。

相比较L1节点，L2节点对计算能力的要求较低，L2节点更多是处理assets数据的存取加速，因此倾向于带宽和存储资源。

#### Validator节点

Validator节点是被选举为validator的candidate节点。Validator节点与candidate节点的差异是，前者需要参与titan网络的抽查流程，负责对被指派的节点进行抽查，并把抽查结果上报scheduler。

#### Edge节点

Edge节点是titan 网络的L2节点，也是titan网络最外围的节点。

Edge节点利用其带宽和存储资源，把assets数据从源拉取到节点自身存储中。当用户从titan网络请求assets数据时，titan网络会根据用户的位置信息，分配与之网络距离最短的edge节点，用户从这些edge节点拉取assets数据。

所谓网络距离最短，是指用户和edge节点位于同一个城域网（或，城域以太网）中，他们之间的数据流不需要跨域国家骨干网/海底光缆等等，因此网络带宽大、延时低，数据传输速度就快。

### 节点注册

Edge节点，candidate节点要加入titan网络，需要先完成节点的注册。

节点注册流程包括如下的工作：

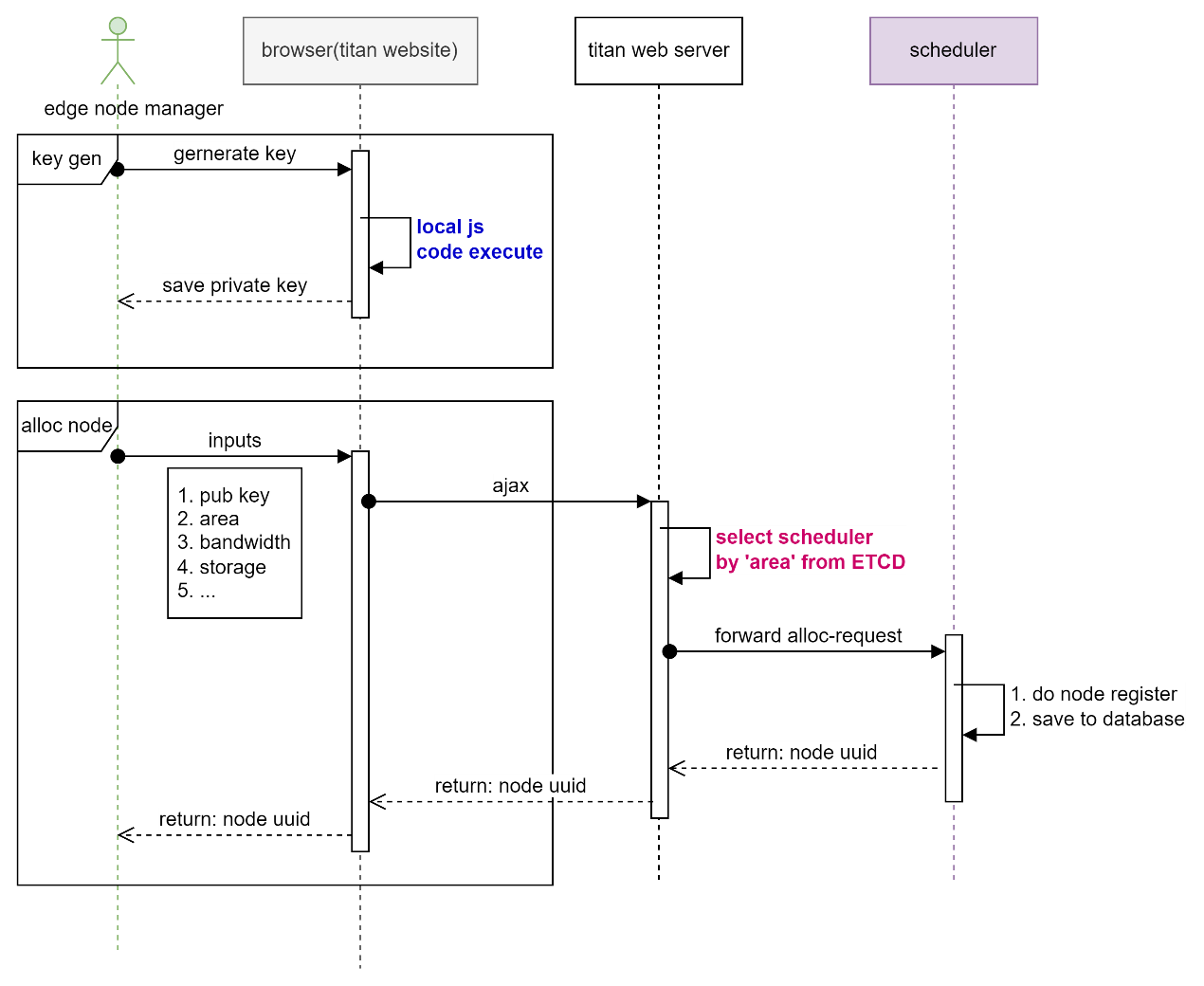
* 本地生成public/private key pair
* 选择节点的area(city)
* 选择贡献的资源额度

节点管理者保存private key到本地存储。

然后完成上述信息填写后，点击节点注册按钮，web程序会把public key和其他信息，根据area(city)发送给titan 网络中对应area的titan scheduler，由后者来完成节点注册。

完成注册后，节点管理者需要把private key、节点的UUID等信息导入到节点的启动配置中，这样才能让节点启动后通过这些信息和scheduler交互。

节点注册流程如下图所示：



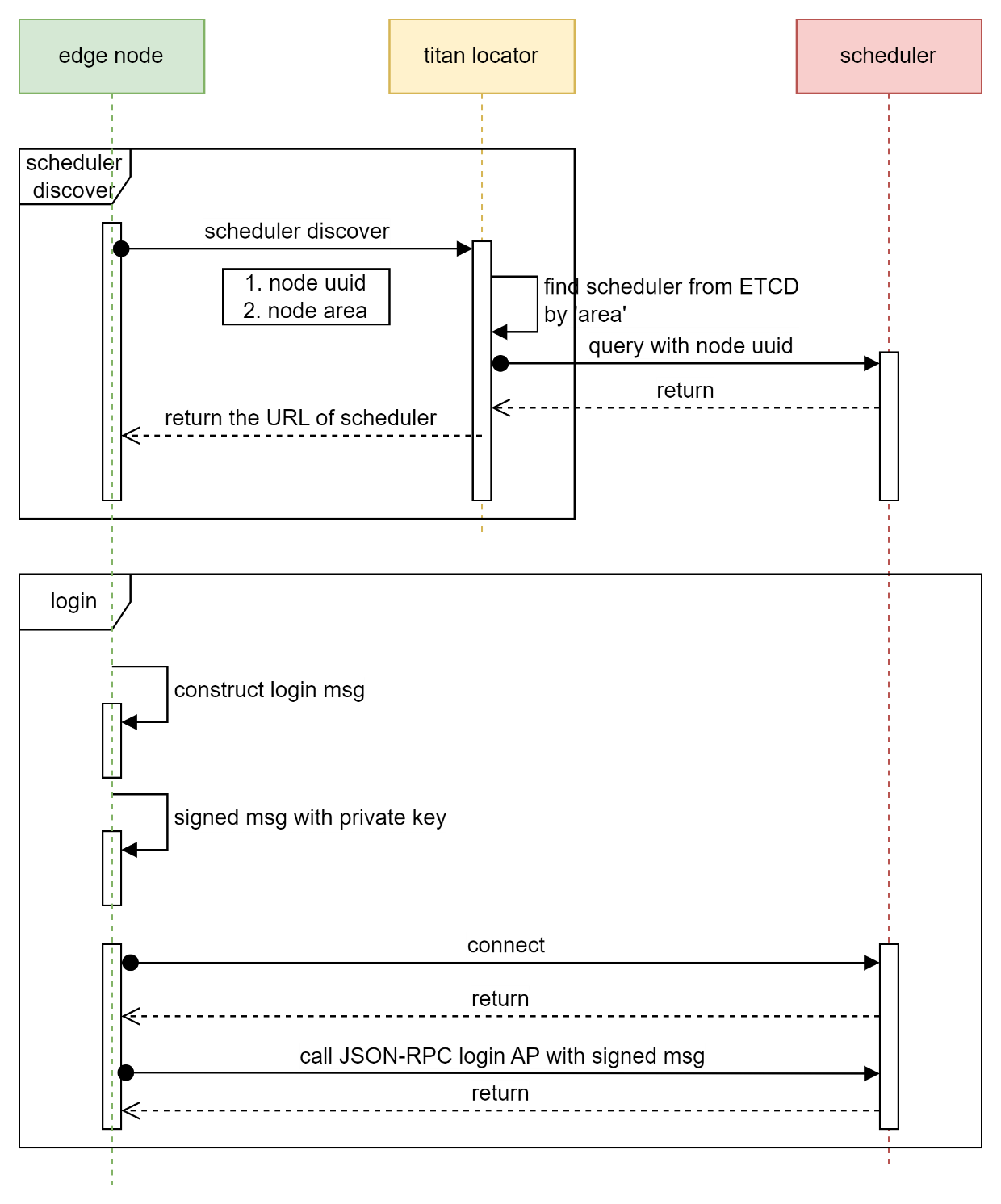
### 节点登录

节点需要登录到scheduler，才能加入titan网络。

完成节点注册后，节点就可以通过注册后获得节点ID，利用private key 签名，生成登录请求消息，并登录到titan网络。

登录时，节点首先连接titan locator，获得节点所在的titan scheduler的IP地址信息，之后和scheduler建立连接，发送登录消息完成登录。

登录流程如下图所示：



### 节点可靠性评分

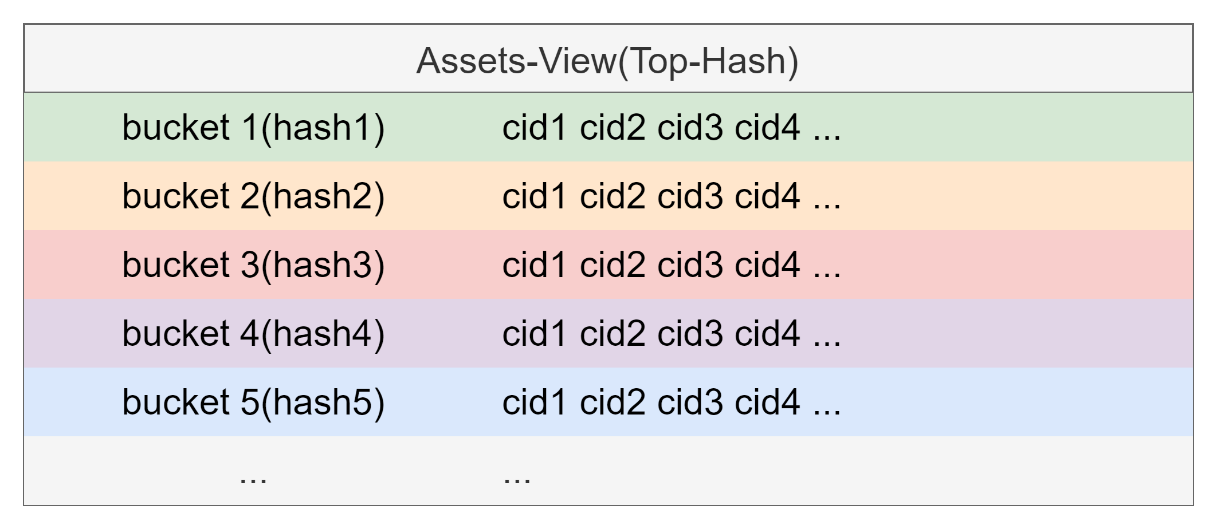
Scheduler对每一个节点，赋予一个称之为可靠性评分的属性。评分依据是节点的在线情况：节点的在线时间越长，评分越高；节点的离线越是频繁，评分越低。

Scheduler在选择用于存储asset数据的edge节点时，会优先选择可靠性评分高的edge节点。

### 数据视图

Scheduler 用数据视图的方式来记录每个节点上的Assets信息。节点自身也以数据视图的方式，记录自身所有的Assets信息。Scheduler定期地和节点对齐数据视图，确保Assets信息是一致的。如果Assets信息不一致，例如，scheduler认为某节点上存在某个asset的数据，但节点并没有这个asset的数据，那么当scheduler把该节点分配给请求该asset资源的用户时，用户从该节点拉取asset的数据就会失败。

数据视图的示意图：



数据视图把节点所有的Assets，划分到多个buckets中。每个bucket分到若干个Assets，bucket记录这些Assets的CID。每个bucket的hash，是由bucket中所有的Assets的CID串接后再次hash得到。所有的bucket的hash串接，再hash就得到整个数据视图的hash。

Scheduler和节点同步数据视图时，首先比较Top-hash，当Top-hash不同时，才比较每个bucket的hash，试图找出hash不同的bucket，当确定了不一致的bucket后，再进一步比较bucket内的CID，直到找到不一致的CID。通过把Assets划分到多个buckets，我们可以快速找到两个数据视图的不一致的地方。

Scheduler和节点会维护各自的数据视图，例如新增Asset，或者删除Asset时，数据视图中的bucket都会发生变化，需要重新计算bucket的hash，和Top-hash。

### 数据同步

Scheduler在数据库中记录每一个节点上的Asset数据的hash信息，并定期地与节点同步，以确保节点的数据视图正确。

同步是由scheduler发起，目前titan scheduler的设定是每天同步一次，这个同步周期是可配置的。

为了加速同步过程，scheduler和节点把Asset数据的hash，划分到若干buckets中， 每个bucket的hash由bucket中所有Asset数据的hash组合得到，而top hash则由所有bucket的hash组合得到。

同步时，首先比较top hash，如果一致，表明数据视图一致，同步过程结束。如果top hash不一致，则逐个bucket hash比较，找出hash不一致的bucket，再从该bucket中找到不一致的Asset数据。

节点需要修正不一致的Asset数据。以保持数据视图和scheduler的一致。

## Assets管理

Titan网络的核心目标是：利用网路内大量的节点的带宽、存储、计算资源，把用户需求的数据cache到与用户网络距离最短的节点上，加速数据的存取，提升用户体验。

每一份数据，在titan网络中是称为asset。

Assets管理是titan scheduler的核心系统，负责asset全生命周期的过程管理。

### Asset

Asset是titan网络中的基本数据单元。Titan网络管理许多asset，titan scheduler的数据库记录每个asset的信息，例如asset的状态，哪些节点存有asset的副本等等。

每个节点的存储存在多个不同asset的副本。

用户通过请求titan网络中的asset，拉取asset的数据。

Titan网络管理者可通过scheduler提供的API管理asset。

下面列举三个Assets管理的核心API。

#### 创建Asset

通过该接口新建一个Asset。

其中，需要指定的参数：

* Asset的源source
* Asset的CID
* Asset的可靠性指标（replica数量）
* Asset的下载带宽需求

Scheduler会检查该CID所指定的asset是否已经存在，如果已经存在，则scheduler会返回asset的基本信息，例如asset当前状态、建立时间、replica数量等等。

Asset的源source，如果是HTTP源，scheduler会使用HEAD命令检查asset在source的状态，如果发生错误，则scheduler不能创建asset对象。

如果source是IPFS源，scheduler不会检查asset在source的状态，这是因为IPFS源没有类似于HTTP HEAD的API接口，以便用于快速检查数据状态。

#### 查询Asset

通过该接口，查询asset的基本信息，例如asset当前状态、建立时间、replica数量等等。

#### 删除Asset

通过该接口，从titan网络中删除asset。

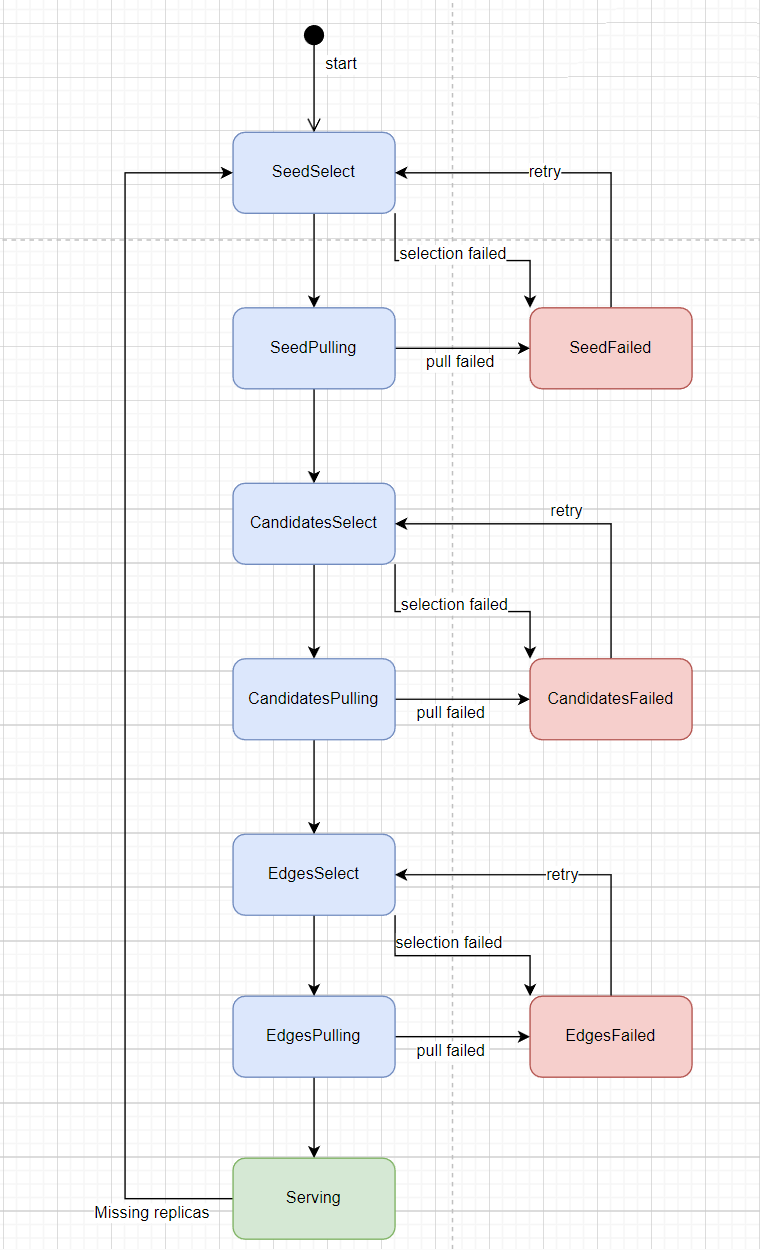
Asset的删除，包括：

* Scheduler请求节点删除asset数据，释放存储资源
* Scheduler从数据库删除asset相关的信息
* Scheduler记录asset删除事件
* 由于节点删除了asset数据，因此节点需要更新local数据视图
* Scheduler同时更新节点的数据视图，与节点的local数据视图一致

### Asset的状态机

Asset 建立之后，scheduler会启动asset的生命周期管理流程。

每个asset都有自己的状态机，状态的转换如下图所示：



Asset的状态：

* Seed-Select: 是asset对象的初始状态，scheduler选择N的 L1节点，N是可配置的，默认是1。这N的L1节点，作为种子节点，从source拉取asset数据。之所以称为种子节点，是因为此时titan 网络没有这个asset的数据，需从source拉取，种子节点拉取完毕，之后titan 网络就具有这个asset的数据。
* Seeds-pulling：被选中的种子L1节点，开始从source拉取asset的数据。这个拉取过程可能耗时较长，取决于source服务器的下行带宽，和数据的大小。
* Candidates-select: 拉取asset数据成功完毕后，titan网络就拥有asset的数据了。Scheduler需要把这个asset数据分发给更多的L1节点，以避免因单个L1节点失效，需再次从source服务器拉取asset数据。因此，scheduler需要选择更多的candidates节点，以便接下来把asset数据分发给这些L1节点。
* Candidates-pulling: 被选中的candidates节点，从种子L1节点拉取asset数据。注意是从种子L1节点，而不是从source那拉取asset数据。也就是说，此时的asset数据，是源自titan网络自身，而不是外部的source。
* Edges-select: 在L1节点分发asset数据完毕后，scheduler接着把asset数据分发到若干edge节点上。Scheduler需要选择edge节点，选择多少个edge节点，以及选择哪个edge节点，是动态变化的。Asset的可靠性要求越高，则scheduler选择的edge节点数量越多；同时edge节点的可靠性评分越高，被选中的可能越大。Scheduler这个阶段的目标是：选择足够的edge节点，这些被选择的节点的总可靠性，需满足asset的可靠性需求。另一个scheduler需要考虑的因素是asset的下载带宽需求，这些被选择的edge节点，总的带宽能力，需满足asset的下载带宽需求。
* Edges-pulling: 被选中的edge节点，从L1节点拉取asset数据。Scheduler分批次地让edge节点从L1节点拉取，即是，如果有N的L1节点，则每批次是N的edge节点，这样做的目的是让部分edge节点尽快拉取完成。如果所有edge节点同时去L1节点拉取，则每个edge节点得到的L1节点的带宽变小。
* Serving: 当asset数据分发完毕后，asset的状态进入服务状态，此时用户就可以从asset的edge节点拉取asset数据了。事实上，由于Edges-pulling阶段，部分edge节点先完成asset从L1节点的拉取，所以Edges-pulling阶段没结束时，用户就已经可以从edge节点拉取asset数据了。

### Asset有效性

Scheduler会持续监控asset的状态。

如果asset的edge节点中，有一些edge节点失效而导致asset的可靠性降低，或者，导致asset的下载带宽不足，那么scheduler会让asset状态从Serving状态切换到Edges-select状态，并执行Edges-select阶段的流程，以便选择更多的edge节点来存储该asset，从而恢复asset的可靠性和下载带宽。

## 抽查系统

Scheduler中的抽查系统，主要职责是根据设定的周期，定期地对titan网络中的节点，进行存储和带宽资源的验证，包括：

* 根据设定的选举周期，定期选举产生validators
* 每次抽查开始时，为被抽查的节点随机指派validator
* 抽查结果验证并记录，这些记录会作为评估节点贡献的依据

抽查目的主要是两个：

* 存储检验：检验节点是否正确存储某个IPFS block或者普通文件range，scheduler在每一轮抽查时从Filecoin主网获取随机数Seed，因此每一轮、每个节点被抽查的block/range index是随机且不同的，这就使得节点不得不保存所有block/range。
* 带宽检验：检验节点的上行带宽资源，这个过程是伴随存储检验发生的，因为存储检验时，节点需要向validator发送block/range，因此validator就能计算出节点的发送速度，进而评估节点的上行带宽

### Validator

在抽查过程中，需要数据发送和数据的Hash(CID) 计算，考虑到titan网络节点众多，因此需要大量的带宽和计算。作为抽查过程的焦点，validator节点需要具备较大的带宽和计算能力，能够高效地完成titan网络的抽查流程。

如果某个validator失效，可能导致edge节点没有validator进行抽查；而validator的带宽资源不稳定，可能导致抽查的结果出错。

因此，validator不仅仅资源需求更大，其资源的稳定性要求也更高。

另一方面来说，作为validator，节点能够获得更多的奖励。

每一轮抽查，我们从validator中选择若干个（N）有效的validator（离线的validator视为失效），N是可配置的。默认是选择所有在线的validators。

然后，我们为每个validator分配edge节点：根据validator的资源情况，给他随机分配M的edge节点，这些edge节点就是validator本轮需要抽查的edge节点。

在分配edge节点时，主要的考虑是：所有分配给某个validator的edge节点的上行带宽总和，不大于validator下行带宽的70%，这样validator的下行带宽就留有足够余地，避免因为validator下行带宽波动，导致抽查结果出现偏差。

由于validator和edge节点的随机配对，每一轮都会重新配对，因此某个validator在每一轮需要抽查的edge节点并不相同，这样的安排能避免validator和edge节点串通作弊。

### Validator选举

Validator节点是从candidate节点中随机选举产生的。

Titan网络设定一个选举周期P，每个周期开始时，从所有的candidate节点中选择一个比率的节点作为validator节点。这个比率R是可变的，依据titan网络的发展进行适当的调整。

例如，titan网络发展前期，candidate节点较少，R可能是100%；等到candidate节点数量多了，R就会降低。

Validator节点需要经过随机选举产生，是希望借此提高抽查作弊的难度。

### 配对

每一轮抽查开始时，scheduler都会对validator和edge节点随机配对。这样做的目的：一是避免validator和edge节点串通作弊，二是避免同一个validator对edge节点抽查而由于validator的原因导致抽查结果偏差。

我们定义validator的下行带宽窗口v-window：每个v-window具有相同的下行带宽B，B是可以配置的。越大下行带宽的validator，具有v-window越多。

配对的过程复杂度是O(N+M)，其中N是所有v-window数量，M是被抽查的节点（edge节点和candidate节点）数量。

### 抽查执行

抽查开始时，scheduler调用edge节点的API，告知edge节点本轮抽查的参数。参数包括随机数Seed，validator的IP地址信息等等。

其中，随机数Seed是scheduler从Filecoin主网获取的分布式随机数，每一轮抽查scheduler都会重新获取。

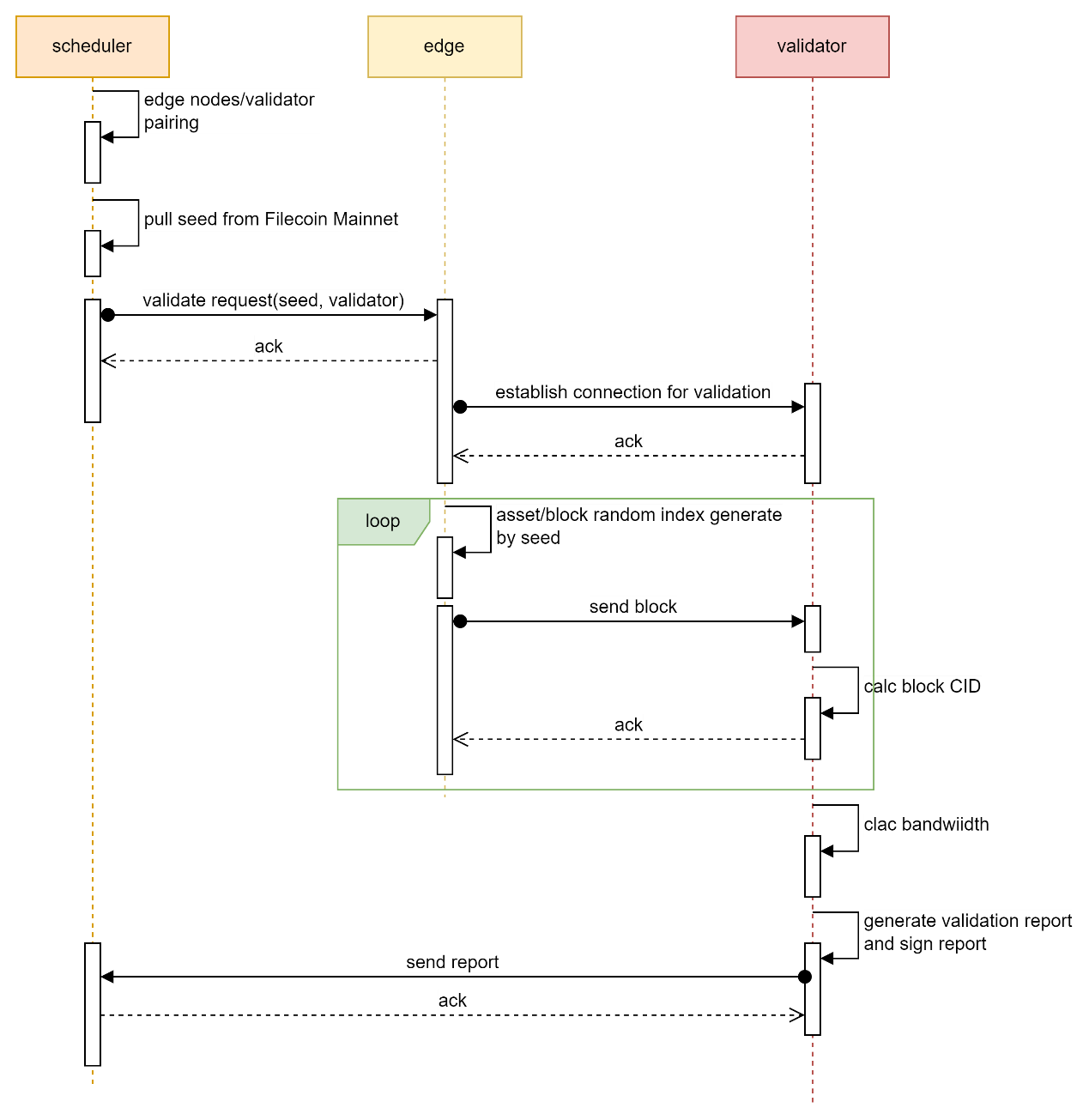
Edge节点根据validator的IP地址，和validator建立用于抽查的通讯连接。

通讯连接建立之后，edge节点用随机数Seed生成随机数，并从随机数获得若干数字作为索引，选择本地数据视图中的bucket以及bucket中的asset对象（carfile），最后选择asset对象（carfile）中的block，并把block发送给validator。

Validator接收到block数据后，计算出block的CID，如果edge节点发送多个block，则逐个接收并计算每一个block的CID。

Edge节点发送完毕，validator获得所有edge节点发送的block的cid，并计算出edge节点的平均发送速度，然后把这些信息作为该edge节点的抽查结果，签名并发送给scheduler。

流程如下图：



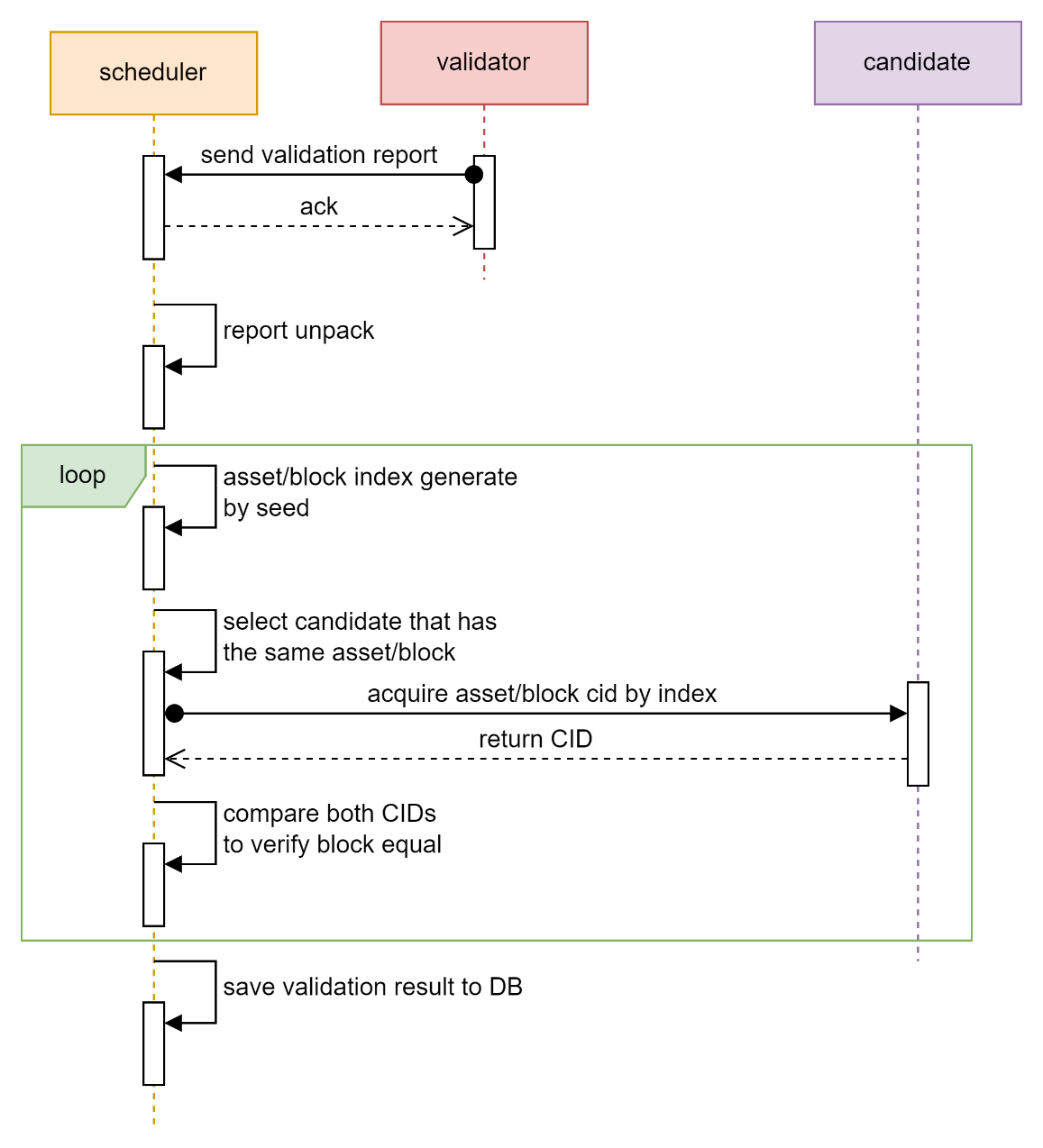
### 抽查结果处理

Scheduler收到validator发送的抽查结果后，使用validator的pub key验证抽查结果，以确保抽查结果来自于validator。

然后，scheduler会对抽查结果中包含的CID，逐个检查。Scheduler服务器自身并不保存block的数据，但scheduler知道哪些L1节点拥有这些数据，因此scheduler会请求拥有相同block数据的L1节点发回该block的CID，并与抽查结果中的CID比较。如果两个CID一致，则表明edge节点存储的block是正确的，反之则表明edge节点的block数据错误。

抽查结果除了包含一个或者多个block的CID，还包含edge节点发送这些block数据的平均速度。Scheduler记录这个平均速度，作为这个时间段edge节点的带宽贡献。

对抽查结果的处理流程如下：



### 带宽抽查豁免

由于抽查时，不仅检验存储的正确性，也检验了带宽，通过让edge节点持续发送若干block，validator能测量出edge节点的上行带宽。

但，如果此时，edge节点正在给用户发送asset数据，而带宽抽查挤占了上行带宽资源，势必会影响edge节点给用户发送数据的速度；另外，由于edge节点正在给用户发送数据，那么抽查测量得到的带宽就不是edge节点真实的上行带宽。

因此，当edge节点正在给用户发送数据时，edge节点接收到scheduler的抽查指令，edge节点和validator建立抽查通讯连接时，通过参数告知validator，edge节点需要带宽抽查豁免，仅进行存储抽查。

在本轮抽查结束后，scheduler会检查edge节点的带宽抽查豁免是否有效：scheduler检查该时间段内，是否把edge节点分配给某个用户，如果是，则认为豁免申请有效。反之，则认为豁免申请无效。

豁免申请有效时，虽然edge节点没有进行带宽抽查，scheduler会把最近若干次该edge节点的带宽测量结果作为参考，并给与edge节点相应的带宽奖励。

如果豁免被认为是无效的，那么scheduler会视edge节点为缺席带宽抽查，edge节点无法获得带宽奖励。

## 用户请求系统

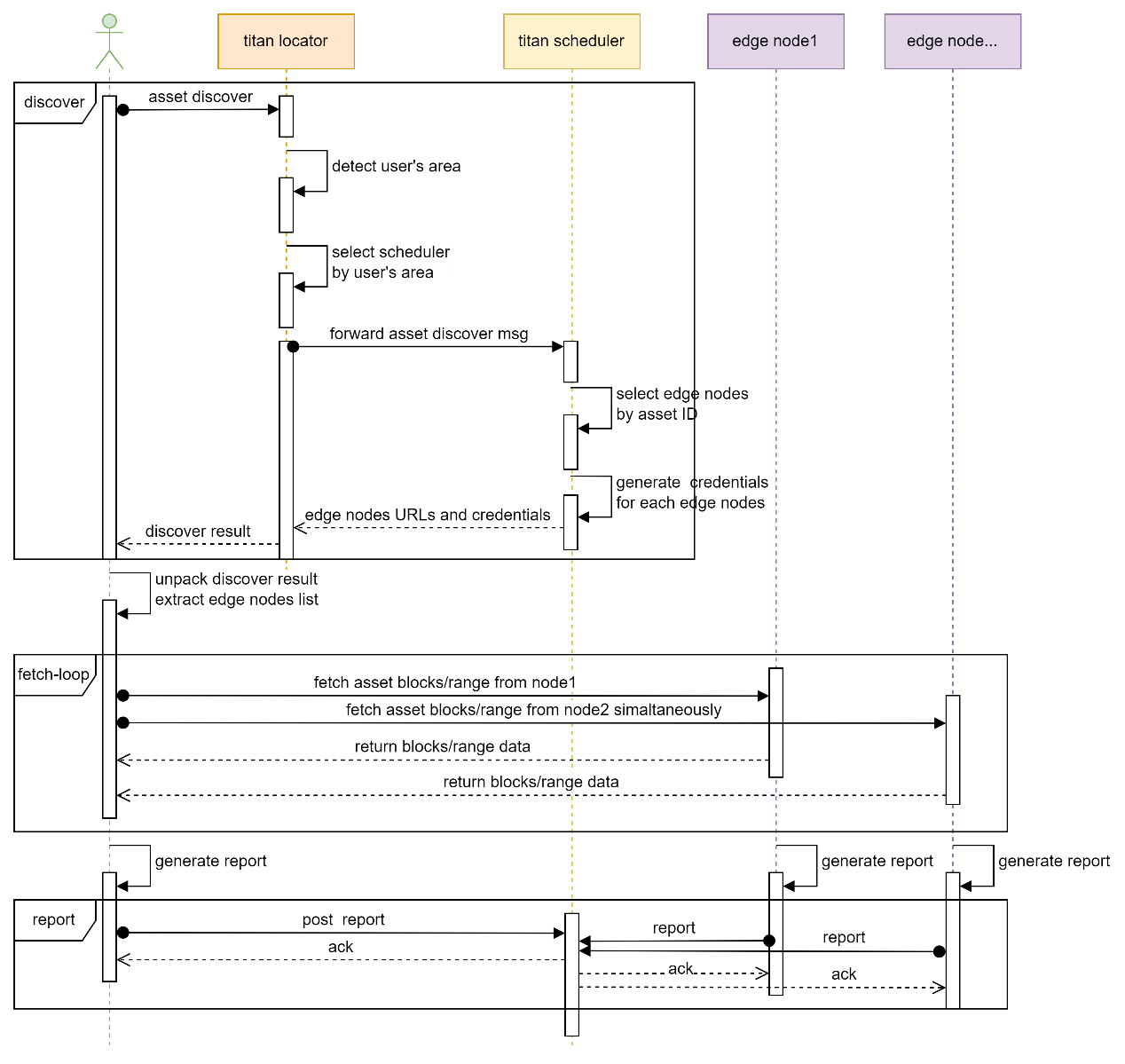
Titan网络的目标是加速数据存取速度，提升用户体验。

Titan网络的方法是，利用多个area(city)的titan子网的节点资源，把asset数据经由最短的距离发送给用户。

在用户请求获取asset数据时，titan网络需要根据用户的IP地址等位置信息，为用户分配同一个area(city)的edge节点，之后用户就可以从这些edge节点拉取asset数据。

由于提供数据的edge节点和用户处于同一个area，也即是同一个城域网，数据传输不需要经过繁忙的国家骨干网或海底光缆，因此传输带宽大、延时低，所以数据获取速度快，提升用户的数据获取体验。

用户从titan网络获取asset流程如下：



概要来说，用户从titan网络获取asset数据包括三个步骤：

* Asset发现：用户通过请求titan locator，locator根据用户的IP地址等，得到用户所在的area，然后把用户请求转发给该area的titan scheduler，后者给用户返回若干具有该asset数据的节点的URL列表，以及节点的访问凭证列表。
* Asset拉取：用户拿到edge节点列表后，就可以通过edge节点拉取资源数据，用户可以并发拉取过程：也即是同时从多个edge节点拉取资源数据的不同部分，以加快数据拉取速度。
* 报告：拉取完成后，用户向scheduler提交asset数据拉取报告。

### Asset访问凭证

Asset访问凭证有两个作用，第一是资源的下载授权，如果没有凭证，是不能够从edge节点下载资源的；第二是edge节点需要基于凭证生成工作量证明，表明edge节点提供了凭证所指定的资源的下载服务。

Asset访问凭证是titan scheduler服务器使用edge节点的public key加密，并使用scheduler的private key签名的数据块。由于只有对应的edge节点才能解密，因此这个凭证仅对特定edge节点有效。Scheduler利用自身的私钥签名，确保凭证是scheduler签发的。

凭证的数据块，包含且不限于如下内容：

* 凭证的唯一ID
* 凭证的carfile
* 凭证的下载速度限制
* Client ID，用户端的唯一ID
* 凭证的有效期等等

Scheduler返回edge节点列表给用户时，scheduler会为每一个edge节点生成一个访问凭证。

用户需要利用凭证，才能从edge节点拉取数据，edge节点会检查凭证的有效性，如果凭证无效，edge节点拒绝用户拉取请求。

## Workload管理系统

Workload管理，主要目的是接收titan网络中的节点的workload报告，并验证报告的有效性。根据有效的workload报告，记录节点的workload到数据库。这些有效的workload记录会作为评估节点对titan网络的贡献的依据。

用户从每个edge节点拉取数据完成后，向scheduler发送报告。

Edge节点定时地提交的workload报告，包含且不限于如下内容：

* 凭证的唯一ID
* 用户ID
* 下载速度
* 下载数据量
* 下载开始时间
* 下载结束时间等等

一个最终有效的workload报告，当且仅当edge节点和用户都提交了相同的且有效的workload证明，才被承认。也就是说，仅仅是edge节点，或者用户，提交了workload报告，是不会被视为有效的workload，因为也不会被作为评估edge节点的贡献的依据。

## FVM智能合约

Titan 网络目前有两个地方使用FVM的合约。

一是把定期生成的关于贡献记录（工作量证明）的carfile的CID保存到合约中。因此，合约为titan网络提供了回溯，任何人都能够通过这些CID获得对应的carfile，通过人工或利用工具，分析和检验carfile内的贡献记录的正确性。

二是利用FVM DATA-DAO合约，实现titan网络数据备份到Filecoin storage provider。

### 节点贡献记录FVM合约

Titan网络在FVM上部署一个叫 “节点贡献记录”的合约，该合约主要作用是，不断地保存新增的CID，这些CID是carfile CID，对应一个一个的carfile，每一个carfile的内容是关于titan网络中节点的贡献记录。

Scheduler会定期打包节点的贡献记录，包括抽查记录，下载工作量证明，生成carfile并保存到Filecoin DC deal（DC Sectors）。同时，scheduler会把生成的carfile的cid，提交到titan网络的FVM合约：贡献合约上，任何人都可以通过合约的接口，获取某时间段的贡献记录carfile的cid，然后通过cid从Filecoin SP获得carfile，并审查carfile中的节点贡献记录。

### 冷备份FVM DATA-DAO合约

Titan 网络中的数据，主要是作为hot cache数据使用。为了提高数据的可靠性，titan网络定期把数据备份到Filecoin Storage Provider。Titan 网络利用FVM的DATA-DAO合约，发布备份数据的存储订单，Filecoin 的Storage Provider通过合约接收并完成storage订单。Storage Provider完成storage订单后，会获得相应的奖励。

当titan 网络需要恢复数据时，titan网络需要从完成存储订单的Storage Provider retrieve数据，因此Storage Provider需要提供足够的带宽，以便titan网络能够尽快retrieve。Titan网络通过FVM DATA-DAO合约发布retrieve订单，Filecoin Storage Provider通过合约接收并完成retrieve订单。Storage Provider完成retrieve订单后，会获得相应的奖励。