# 概述

Edge节点是运行titan edge 程序的设备。

作为titan网络的边缘设备，edge节点负责titan网络内容分发过程的最后一公里，把内容以最短网络距离传输给用户。

Edge节点运行于titan的某个area网络实例，titan网络是由多个titan area网络构成。

Edge节点一般是家庭设备，因此titan网络可以拥有众多的edge节点，利用这些edge节点存储和带宽资源，构建titan网络庞大的资源池。

每一个edge节点都会根据titan scheduler的指令，pull各种asset数据到自己存储中，并响应用户的请求，发送这些数据给用户。

Edge节点依据他贡献的存储和带宽资源，获取相应的收益。

# Edge节点的意义

例如，中国深圳的用户需要拉取位于美国的source上的数据，这些数据需要通过海底光缆从美国传输到中国，然后经过中国的国家骨干网，最后才到达用户的设备。在高峰期，海底光缆和国家骨干网繁忙导致丢包率高，延时大，用户下载体验很差。

Edge节点的目标是通过把数据保存到本地的存储，用户从靠近的edge节点拉取数据，避开骨干网和海底光缆，使得下载速度稳定，提升用户体验。

Titan 网络会根据用户的IP地址信息，估算出用户所在的城市，为用户分配同城市/区域的edge节点。因此用户从edge节点拉取数据，数据仅需要经过城域网（城域IP网/城域以太网），城域网网络带宽大，延时低，用户的下载速度得到保证。

中国大多数家庭的ADSL，都是FTTH，光纤到家，家庭带宽是很大的，上行带宽普遍到达20Mbps~50Mbps，这些上行带宽利用率很低，因为家庭用户普遍是使用下行带宽。因此，edge节点运行在家庭网络，能够充分利用空闲的上行带宽。

在低峰期，edge节点利用空闲的下行带宽拉取asset数据，在高峰期，利用空闲的上行带宽分发asset数据，使得数据尽可能地在城域网内流通，不需要跨域骨干网和海底光缆，降低骨干网和海底光缆的负荷。

# Edge节点的资源需求

Edge节点主要的资源是存储和带宽，存储资源用于存储asset数据，带宽资源用于给用户拉取asset数据。

我们期望每个edge节点的物理存储空间能够达到2TB以上。

带宽方面，edge节点的下行带宽主要用于从source拉取asset数据到本地存储，edge节点的上行带宽主要是用于把asset数据发送给用户。 家庭宽带网络的下行带宽一般都比上行带宽大很多，因此edge节点的带宽主要是上行带宽，我们期望每个edge节点的上行带宽能够达到20Mbps以上。

# Edge节点的缺点和应对

Edge节点一般是运行在家庭环境的设备。家庭环境意味着，存在网络不稳定、供电不稳定等等会导致设备离线的因素。

例如，中国的家庭网络一般是ADSL，通过拨号上网，获得的IP地址是临时的，可能每一天变化一次IP地址。现在由于移动设备大量增加导致IP地址不足够，家庭网络获得的IP地址甚至不是公网IP，而是ISP的内网IP地址，通过ISP的NAT设备访问公网。

因此，edge节点离线的可能性较高，我们通过把asset数据复制到多个edge节点，避免单个节点失效时，用户没法下载asset数据。

Titan scheduler根据edge节点的在线和离线情况，估算他的预期可靠性。Asset数据的可靠性，可以依据所有拥有该asset 数据副本的edge节点的可靠性，进行评估。如果asset数据可靠性低于某个阈值，titan scheduler会选择更多的edge节点持有该asset数据副本。

# Edge节点的贡献和收益

Edge节点把asset数据从source拉取到自己的存储设备，之后，附近用户需要同一数据时，直接从edge节点拉取，一来减轻了source的带宽负担，二来降低用户获取数据的耗时，提升用户体验。

因此，edge节点为titan网络贡献了存储，和带宽。Titan网络对edge节点的奖励也分为两类，一是基于存储的奖励，edge节点贡献的存储越大，奖励越多；另一类是基于带宽的奖励，edge节点贡献的带宽越大，奖励越多。

根据titan网络的发展规划，奖励模型分为两个阶段。第一个阶段对应titan网络发展早期阶段，这个阶段，titan网络中的asset数据还不够多，因此edge节点的存储的利用率不高，另外也不会有太多用户从网络中的edge节点拉取数据，因此edge节点的带宽利用率也不高。此时相当于titan网络租用了edge节点的存储和带宽，只是还不能充分利用这些资源，因此需要一个抽查过程来确保edge节点的存储和带宽的有效性。只要edge节点的资源通过了抽查验证，titan网络就会为edge节点发出奖励。

第二个阶段对应titan网络发展的成熟期，此时网络中有较多的asset数据，以及有较多用户从网络的edge节点拉取数据。Edge节点存储贡献主要依据真正的存储的asset数据的多少，当然，物理存储越大，能够存储的asset数据越多，奖励也就越多。而带宽的贡献，只要是依据用户从edge节点拉取数据量，以及速率，如果没有用户从edge节点拉取数据，那么节点的带宽贡献就会很小。

奖励模型在titan网络发展过程中会继续优化，以便更合理反应edge节点的贡献。

# Edge节点加入网络

Edge节点需要加入titan 网络，其存储和带宽资源才能被titan网络利用。例如，edge节点加入网络之后，titan scheduler才能会选择它作为asset数据的存储，才能让用户从它那拉取asset数据。

Edge节点加入网络的流程，分两步，首先请求titan locator服务器，获得edge节点的scheduler服务器访问地址；然后edge节点连接scheduler服务器，并发送使用private key签名的登录请求。

Titan网络会根据节点注册时选择的area(city)，确定节点应该由哪个scheduler管理，并把节点的注册信息写入到scheduler数据库。Edge节点询问titan locator服务器时，后者根据edge节点的ID就能确定节点所属的titan scheduler。

Edge节点向titan scheduler发送请求时，后者根据节点的ID从数据库中加载节点注册时登记的信息，如果ID不存在，则scheduler拒绝节点登录。Scheduler会使用节点注册时提供的public key验证登录请求，验证失败时，scheduler会拒绝节点登录。

## 节点注册

节点管理者首先需要一个titan账号，这个账号是一个有效的Filecoin钱包地址，可以通过titan website申请，或者通过lotus客户端申请得到。

使用账号登录titan website之后，选择节点注册入口，就会进入节点注册页面。这个页面主要是需要用户做一些选择，例如节点计划运行在哪个area(city)，节点计划贡献的资源额度，包括存储额度和带宽额度等等。然后注册页面运行本地的JS脚本，生成节点的ID和密钥。其中，private key会让用户选择保存，public key以及节点ID会连同用户的选择信息，发送给titan web服务器，后者交给titan scheduler，由scheduler完成注册流程。

一旦注册流程完成，节点就会被绑定到账号上。

之后，节点管理者需要把private key和节点ID导入到节点的程序中，节点程序利用这些信息，登录titan网络。

## Keepalive

Edge节点需要定期地发送保活数据包到给titan scheduler，保持节点在scheduler的在线状态。如果edge节点不是在线状态，titan scheduler不会对节点抽查，也不会指派用户从这个节点下载数据，因此edge节点无法获得收益。

# Edge节点的资源抽查

Titan网络的资源是由多个edge节点的资源组成，edge节点的资源的有效性，决定了titan网络的有效性。因此titan网络定期地对edge节点的资源进行抽查，并把抽查结果跟edge节点的收益挂钩，通过这种方式鼓励edge节点保持其资源的稳定可靠，从而提升titan网络的可靠可用。

抽查主要是为了检查edge节点的asset数据的存储，以及检查edge节点的网络带宽。抽查策略随着titan网络的发展，会有变化，例如，当titan网络的带宽资源绝大部分用于用户下载时，抽查就不能再检查edge节点的带宽了，因为那样会占用edge节点的上行带宽，影响了用户下载体验。

抽查是由scheduler规划和发起的，按照设定，每天对edge节点进行若干次抽查。首先，scheduler会选择一些L1节点作为验证者，这里的L1节点是相对于edge节点而言，edge节点在titan网络中被称为L2节点。每一轮抽查开始时，scheduler会把被抽查的edge节点信息发送给某个验证者节点，然后验证者节点就会对edge节点进行抽查。

对edge节点抽查的过程是，如下图所示：

Titan scheduler每一轮抽查开始时，从Filecoin主链获得分布式随机数，结合每个edge节点的ID，生成该节点的抽查随机数，这个随机数决定了需要抽查edge节点的哪个asset，以及asset中的哪个block。Titan scheduler把这些信息发送给验证者，验证者对edge节点抽查时发给edge节点。Edge节点根据这些信息，读取对应的asset中的block，发回给验证者。

验证者收到edge节点发送的block后，计算出block的CID，计算出edge节点发送block的网络速度等等，把这些作为抽查结果，发送给titan scheduler。

Titan scheduler收到抽查结果后，需要对比block的CID，是否和数据库上的block CID一致，如果不一致则表明edge节点数据有问题。

Titan scheduler每一轮都会把所有edge节点的抽查结果打包，并生成零知识证明，把这个证明提交到FVM合约上，以供回溯。

# Edge节点Asset系统

Edge节点核心任务是，把asset数据从指定的source拉取到edge节点的存储系统中，然后通过HTTP 服务器给用户提供下载服务。因此，对asset的管理，包括asset的数据存储，asset对象查询，asset数据视图一致性等，是edge节点的核心功能。

## Edge节点存储管理

每一个asset，都需要相应的存储，用于保存asset的数据。Edge节点除了需要用于asset对象的存储，也需要用于节点管理的存储，edge节点自身产生的数据，例如用于记录asset对象CID的key/value数据库，节点运行日志等等。考虑到存储设备故障率，我们建议这两类存储是分属不同的物理存储设备。举例来说，存储asset数据的物理存储设备，它的读写更频繁，因此更容易出现故障，如果用于节点管理的数据也位于它之上，导致节点无法正常运行。

Edge节点的Storage manger组件用于管理节点的存储。例如，节点管理者可以挂载多个存储路径，并设置每个路径可用存储额度，和设置存储的用途。例如用于存储节点管理数据，或者用于存储asset对象数据。Storage manger通过定时stat存储路径，检测存储路径是否在线。存储路径离线后，其上的asset数据无法使用，edge节点会向titan scheduler上报asset数据失效，scheduler会避免使用asset数据失效的edge节点。

## Edge节点Asset对象管理

Edge节点的Assets manager组件，负责管理节点上的所有asset对象。对于已经完成数据拉取的asset对象，Assets manager通过把这些asset对象划分到不同的buckets，然后通过buckets hash，定期地与titan scheduler同步，确保edge节点的asset数据视图和scheduler的asset数据视图是一致的。在响应scheduler创建新的asset对象请求时，Assets manager通过Storage manager组件申请用于新asset对象的存储空间，并负责从scheduler指定的source拉取asset数据到新分配的存储上，完成asset请求。

用户通过edge节点的HTTP 服务器下载asset数据时，HTTP服务器需要通过Assets manager获得asset数据的读取接口。如果用户只是下载asset数据的某个block，则还需要asset对象的index数据，而Assets manager 负责index的加载和管理，例如当index不再需要时，能够卸载index，减少内存占用。

## Edge节点pull asset

Titan scheduler调用这个接口，请求edge节点拉取一个asset到本地存储。如果没有错误，edge节点将从titan scheduler指定的source拉取该asset(carfile)的所有blocks，并使用BFS（深度优先遍历）的方式把这些blocks写入到一个car v2 格式的carfile文件，写入是使用”go-car” golang library完成的，go-car也会构建carfile的index并写入到carfile文件。

需要carfile的index，是因为edge节点的下载服务，支持用户通过block的ID请求单个block内容，因此我们需要index来找到block位于carfile中的偏移，才能快速读出block的内容。

Edge节点收到pull asset请求之后，首先检查asset是否已经被pull到本地存储，如果是则返回错误，告知scheduler该asset已经被pull到本地。

Edge节点检查当前是否有asset pull任务正在进行，如果是，则需要让新的asset pull请求排队，因为edge节点只允许一个正在进行的asset pull任务。

Edge节点把正在进行的以及排队的asset pull请求都做了持久记录，因此即使edge节点程序重启，它也能从持久记录恢复这些asset pull请求。

Edge节点通过HTTP协议从源拉取asset数据。

对于carfile，edge节点是逐个block请求source，这主要是考虑到网络错误发生时，损失的数据是若干blocks，已经下载的blocks不需要重新下载。后面我们也可能支持以range的方式下载carfile，每一个range下载完成后，通过range的hash检验数据是否正确，网络错误发生时，我们损失的是若干个ranges，已经下载的ranges不需要重新下载。Range的方式比block的方式粒度更大，减少HTTP请求次数，而且range的方式不仅可以下载carfile，还可以下载其他格式的文件，例如原始的视频文件。

Edge节点asset pull请求之后，scheduler会周期地查询asset pull请求的进度。如果edge节点崩溃，或者网络出现问题，scheduler查询失败若干次后，就会认为asset pull请求已经失败，scheduler将会重新选择其他edge节点进行asset pull。

## Edge节点释放asset

Titan scheduler请求edge节点释放不再需要的asset，edge节点删除本地的carfile文件和index文件。

Edge节点还需要从carfile管理器中删除asset记录，更新carfile所属的bucket的hash，以便之后快速完成与scheduler的状态同步。

## 查询Edge节点的asset状态

Titan scheduler会调用这个接口获取asset的信息，edge节点返回asset的文件大小，文件创建日期等基础信息。如果asset不存在，则返回错误信息。

## 检查Edge节点的carfile完整性

Titan scheduler会调用这个接口，让edge节点重新扫描carfile内含的blocks，检查CID和block内容是否一致。

这个过程需要扫描整个carfile，因此可能很耗时，edge节点会控制该接口的并发数量，超过并发数量就会返回“设备忙碌”错误。

## 与调度服务器的状态同步

Edge节点的存储持有asset数据，它需要定期跟titan scheduler对齐，以确保asset数据一致。如果asset数据不一致，edge节点需要从titan网络的其他拥有该数据的节点重新获取asset数据。

Titan scheduler的数据库记录每一个节点保存的所有的carfile hash列表。这些hash被划分到多个bucket中，以提高状态同步的处理速度。

同步过程是由titan scheduler发起。Titan scheduler和edge节点把所有carfile划分到若干个buckets，同步时主要是比较bucket的hash：

bucket\_hash = HASH (all carfile hash)

top\_hash = HASH (all bucket\_hash)

Titan scheduler首先发送top\_hash给edge节点。Edge节点也通过同样的方法计算本地的top\_hash，如果两个top\_hash相同，则表明数据是一致的，状态同步完成。

如果top\_hash不同，则表明存在不一致的数据，首先需要找出不一致的bucket：titan scheduler把所有bucket\_hash发送给edge 节点，后者逐个bucket比较，找出不一致的bucket。

确定了不一致的bucket之后，edge节点请求titan scheduler发回差异的bucket中的carfile hash 列表，找出不一致的carfile，最后修正这些差异的carfile。

# Edge节点的下载服务

Edge节点是通过HTTP服务器的方式对外提供下载服务。

考虑到edge节点通常是运行在家庭环境，因此NAT穿透是常态。NAT穿透需要服务器和客户端协作才能完成，因此我们也提供了SDK，集成了NAT穿透服务，因此用户可以通过使用我们的SDK，从edge节点拉取资源，后续章节会详述NAT穿透过程。

## 下载协议的选择

Edge节点的HTTP服务器，同时支持基于TCP的HTTP，和基于QUIC(UDP)的HTTP。

基于TCP的HTTP，优点是客户端支持广泛，但缺点是NAT穿透困难。

基于QUIC的HTTP，优点是QUIC基于UDP，NAT穿透容易，同时QUIC也能改善TCP拥塞情形下的带宽利用率，缺点是客户端支持没有基于TCP的HTTP广泛，浏览器来说，Chrome以及同样的基于Chromium内核的浏览器都支持QUIC的HTTP。

Edge节点通常运行于家庭网络环境，也就是其通常位于NAT设备之后，因此需要NAT穿透，所以基于QUIC的HTTP下载协议是重要的。

## HTTP 服务器

Edge节点的HTTP服务器实现，参考了IPFS HTTP Gateway的定义：<https://github.com/ipfs/specs/blob/main/http-gateways/PATH_GATEWAY.md>。

Trusted模式

Edge节点把carfile内含的原始文件提取出来，发回给用户。Edge节点为了加快响应，对最近访问的（LRU）carfile原始文件asset，这样就不需要重复提取过程。

Edge节点确保发送的内容的正确，譬如carfile的内容或block的内容计算得到的CID必须和carfile或block的CID一致，这个计算过程edge节点是在asset这些内容时就已经进行过的，因此不需要在用户请求时重复进行。

用户信任edge节点发回的数据。

Trustless模式

Edge节点不会提取carfile内含的原始文件，而是直接把carfile或者block发回给用户。用户收到carfile或block后，需要自己验证所接收的内容是否正确，譬如根据内容计算出的CID需要一致。而edge节点则不会进行计算。

另一方面，由于这些内容都是edge节点所管理的asset，在pull这些内容的时候，edge节点会对内容进行计算和验证，确保其计算出的CID是一致的。

因此，从edge节点的角度来看，trust和trustless模式的差异，就只有edge节点是提取carfile内的原始数据（解码）送给用户，还是直接发送carfile的数据给用户。

IPNS

我们使用CID来表示block/carfile的唯一标识，CID是基于block/carfile内容的哈希，如果block/carfile内容更新，CID就会变化，因此也必须同步更新CID。这就带来一个问题：资源更新时，其标识也被迫更新，客户端就没法根据一个确定的标识去获取更新后的内容。

因此，我们需要IPNS（InterPlanetary Name System），每一个IPNS对应一个CID记录，该CID记录可以更新。

内容更新时，我们同时更新该IPNS对应CID记录，那么这个IPNS就对应到了新的CID记录。这就类似于我们DNS域名系统，我们有一个域名例如example.com，它对应的IP地址记录，可以更新。

Edge节点后续版本需要支持IPNS，以便能够满足用户使用同一个标识获得更新后的内容。

DNSLink

IPNS是基于资源拥有者的公钥的哈希，例如/ipns/hash{resource-owner-pub-key}，因此可读性不好。除了IPNS，还可以使用DNSLink，利用DNS域名系统的TXT记录，来达到IPNS的效果，而且DNSLink的名字可读性更强，例如\_dnslink.docs.ipfs.tech，通过DNS TXT记录，可以被解析为：

*/ipfs/bafybeieenxnjdjm7vbr5zdwemaun4sw4iy7h4imlvvl433q6gzjg6awdpq*

对DNSLink的支持，我们计划在后续版本实现到Titan DNS服务上，由Titan DNS服务维护TXT记录，并响应用户的DNS查询。

Caching 控制

Edge节点的asset管理是受titan网络的scheduler服务器控制的，例如只有scheduler可以增加或者删除edge节点的asset内容。

Edge节点不对用户提供cache的控制机制，因此edge节点会忽略用户的HTTP Headers的 cache相关设置。

Subdomain Gateway

现在的浏览器，都有一个跨域安全模式。模式的介绍，模式的意义。

我们需要满足这个模式，就需要subdomain的方式。

### HTTP接口

#### GET接口

GET /ipfs/{cid}[/{path}][?{params}]

参数说明

cid：指定资源的CID，例如carfile或者block的CID

path: 可选的资源path，用于指定CID为根的UnixFS目录下的路径

params：可选的query parameters，用于指定服务器返回结果时选项

#### HEAD接口

HEAD /ipfs/{cid}[/{path}][?{params}]

跟GET接口类似，但是edge节点不发送carfile/block的内容，用于支持用户查询CID对应 的内容的信息。

例如，查询CID对应的内容是否存在；查询CID对应的内容的长度；查询CID对应的内容是什么类型（图片等等）。

#### query parameters

filename

当用户的请求指定了filename，例如filename=abc.jpg时，edge节点会在HTTP Response的头部设置Content-Disposition参数，并根据文件名字的后缀设置Content-Type参数。

download

当用户的请求指定了download，例如download=true时，edge节点会在HTTP Response的头部设置Content-Disposition: attachment[;filename=...]，告知用户侧不用渲染该响应，而应该弹出Save as对话框，让用户保存文件。

format

用户通过format参数指定他期望的内容的格式。效果等价于在HTTP Request的header中设置Accept参数。

对应的关系如下：

|  |  |
| --- | --- |
| format=raw | Accept:application/vnd.ipld.raw |
| format=car | Accept:application/vnd.ipld.car |
| format=tar | Accept:application/x-tar |
| format=dag-json | Accept:application/vnd.ipld.dag-json |
| format=dag-cbor | Accept:application/vnd.ipld.dag-cbor |
| format=json | Accept:application/json |
| format=cbor | Accept:application/cbor |

### HTTP Request Header

Accept

用于指明用户能够接收的内容格式

application/vnd.ipld.raw：原始的block内容，edge节点不需要解析block的内容，直接发送给用户即可

application/vnd.ipld.car：原始的carfile内容，edge节点不需要解析carfile的内容，直接发送给用户即可

application/x-tar:如果CID所指的内容是UnixFS文件夹，则把文件夹打包成tar文件，发回给用户。如果CID所指的内容不是UnixFS文件夹，则返回400，表示Bad Request。

application/vnd.ipld.dag-json: Edge节点把CID所指的内容通过IPLD Data Model转换为DAG-JSON format格式，如果内容本身就是DAG-JSON format格式则无需转换，发送给用户。如果转换过程出错，则返回500给用户，表示内部错误。

application/vnd.ipld.dag-cbor: Edge节点把CID所指的内容通过IPLD Data Model转换为DAG-CBOR format格式，如果内容本身就是DAG-CBOR format格式则无需转换，发送给用户。如果转换过程出错，则返回500给用户，表示内部错误。

application/json:类似于application/vnd.ipld.dag-json，unless the CID's codec already is json (0x0200)，edge节点直接把原始的内容（已经是json格式）发回给用户。

application/cbor: 类似于application/vnd.ipld.dag-cbor，unless the CID's codec already is cbor (0x51) ，edge节点直接把原始的内容（已经是cbor格式）发回给用户。

Range

当且仅当Accept参数是application/vnd.ipld.raw或者application/vnd.ipld.car时，range用于指定原始carfile或者block的某一区段。当Accept参数不是application/vnd.ipld.raw或者application/vnd.ipld.car时，edge节点会返回400错误，表示不支持这样的请求。

### HTTP Response Header

Status

200：表示请求成功

206：当用户按range请求内容时，206回复表示请求成功

400：表示无效的请求

429：表示edge节点当前繁忙，稍后再试

500：表示edge节点内部错误

Response Headers

Etag：用于用户侧标识接收到的内容，Etag的值，必须位于双引号内。edge节点使用CID填充Etag，例如”bafy…foo”。如果用户提供了filename参数，则Etag的值由CID加filename的后缀组成；如果用户提供了format参数，则Etag的值由CID加format组成，例如”bafy…foo.raw”， ”bafy…foo.car”等等。

Cache-Control：edge节点用“public, max-age=29030400, immutable”填充。

Content-Type: edge节点根据用户请求时提供的format，如果是carfile类型，则必须带上carfile的版本号，例如Content-Type: application/vnd.ipld.car; version=1；当用户请求时没有指定format时，edge节点会通过CID的codec，或者内容的metadata等信息填充该字段。

Content-Disposition:当用户请求时设置download=true参数，或者指定format为carfile/raw时，edge节点会填写该字段。分两种，Content-Disposition: inline，表示用户端直接地展示内容；Content-Disposition: attachment，表示用户侧弹出“另存为”对话框，让用户保存文件。

Content-Length: edge节点会填充该字段，用于指明内容的长度。

Content-Range: 当且仅当用户请求时提供range参数，edge节点才会填充该字段。

Accept-Ranges: edge节点在回复用户HEAD请求时，如果内容是否可以按range获取的，填充该字段，告知用户端可以按range获取内容。

X-Content-Type-Options: 当内容格式是carfile/raw时，edge节点会用X-Content-Type-Options: nosniff填充，告知用户端不需要根据内容探测content type。

### HTTP Response Payload

一般地edge节点发送给用户的payload格式，由用户请求时指定的format决定。

如在Accept一节所说明的一样，例如，当用户请求时指定的format为：

application/vnd.ipld.raw时：发送原始的carfile/block给用户。

application/vnd.ipld.car时：发送carfile给用户，注意此时carfile内的block顺序是任意的，例如深度优先或者广度优先的方式排列。

tar时：把UnixFS文件夹打包成tar文件发送给用户。

当用户不指定时，edge节点尝试根据CID的codec来确定payload的格式。

### 浏览器环境下的跨域访问限制

浏览器要求：

相同协议，例如均是HTTPS；

相同端口，例如均是443；

相同的域名

例如一个https的443端口的页面，请求http的内容，或者请求非443端口的内容，浏览器都认为是跨域。

浏览器把请求分类为两类，一是简单请求；二是非简单请求；对于简单请求，跨域是直接允许的，但是简单请求的响应内容格式只能是指定的两三种，例如txt，json，form-data这种，不能是是octet-stream等。

对于非简单请求，浏览器就需要询问原网页的服务器，通过发送OPTION HTTP 请求询问，只有服务器允许跨域，浏览器才允许跨域访问发生。

有几个显然的原因：其一，edge节点的端口一般都不是443/80端口，运营商对于443/80端口可能基于法律法规的限制只对备案后的IP放行；其二，edge节点对应的域名，是titan网络的域名，跟原始网页的域名肯定不同；这两个原因使得对edge节点的访问必然是跨域访问。因此，如果需要在浏览器环境访问edge节点的资源，就需要原始网页服务器允许对edge节点的跨域访问。

Titan网络为每一个edge节点分配二级/三级域名。Titan网络的DNS服务负责对这些域名进行IP解析。

## NAT穿透

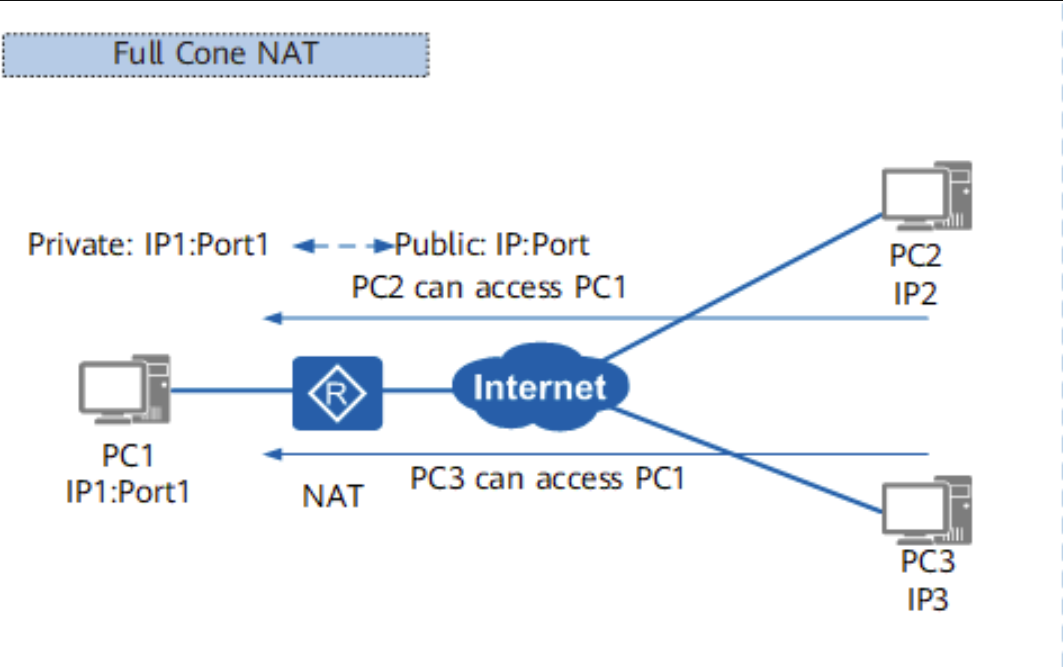
### NAT类型介绍

我们根据STUN的定义，确定以下4种类型的NAT设备：

Full-cone NAT（全锥形NAT）

这类NAT是最容易穿透的。

从相同的内网IP和端口往外发送都会被映射到相同的外网IP和端口，例如所有的内网192.168.1.100:3000往外发送都会被映射到公网66.249.66.82:4000。而任何外网的IP和端口，给公网66.249.66.82:4000发送数据包都会被NAT设备转发给内网192.168.1.100:3000。



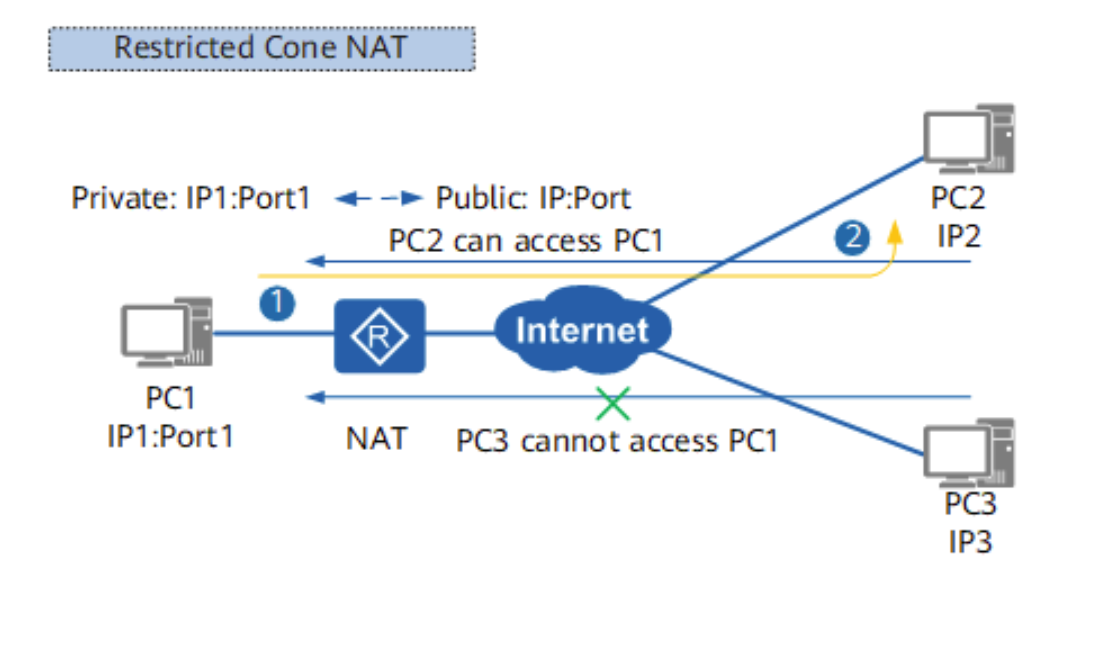
由于这类NAT没有限制，外网所有的IP和端口都可以给映射后的内网IP和端口发包，从网络安全角度来说，存在风险，因此较少NAT设备采用这种NAT方式。

Restricted-cone NAT（受限的锥形NAT）

这类NAT需要穿透动作。

从相同的内网IP和端口往外发送都会被映射到相同的外网IP和端口，例如所有的内网192.168.1.100:3000往外发送都会被映射到公网66.249.66.82:4000，这点和Full-cone一样，但是仅当被记录的外网的IP，给公网66.249.66.82:4000发送数据包才会被NAT设备转发给内网192.168.1.100:3000。

简要的来说，就是当且仅当内网IP对某个外网IP发送过数据包，这个外网IP才能对内网IP发送数据包，如下图所示，IP3想要给IP1发送数据包，则IP1必须首先给IP3发送数据包，以便NAT设备有IP3的记录：

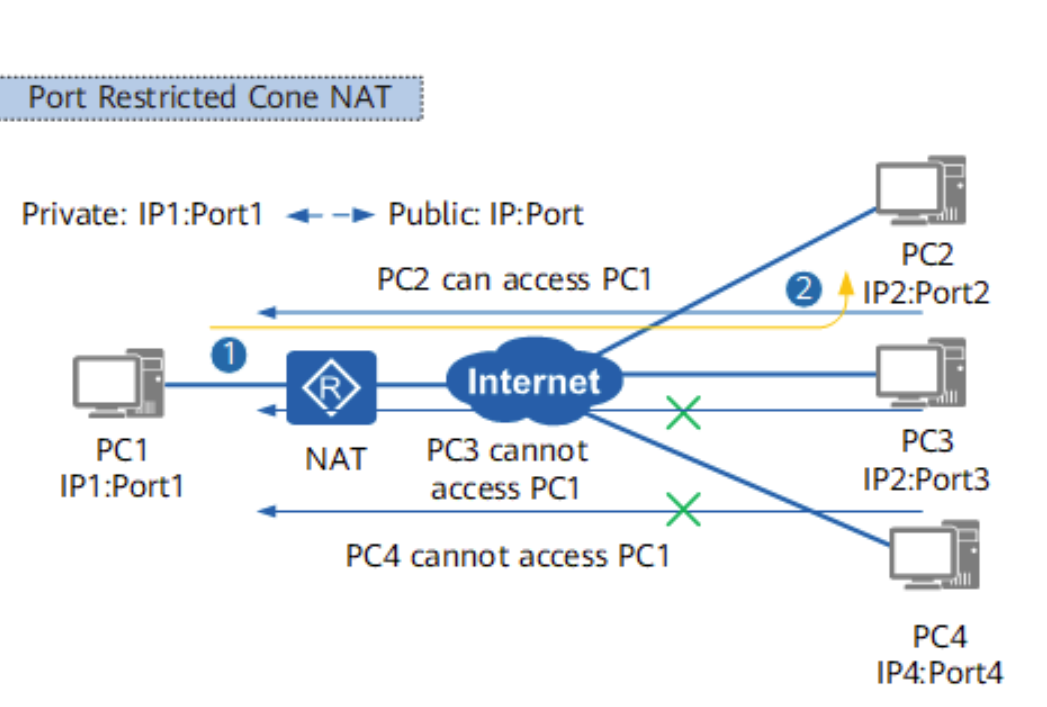


Port-restricted cone NAT（端口受限的锥形NAT）

这类NAT需要穿透动作。

从相同的内网IP和端口往外发送都会被映射到相同的外网IP和端口，例如所有的内网192.168.1.100:3000往外发送都会被映射到公网66.249.66.82:4000。但是仅当被记录的外网的IP和端口，给公网66.249.66.82:4000发送数据包才会被NAT设备转发给内网192.168.1.100:3000。

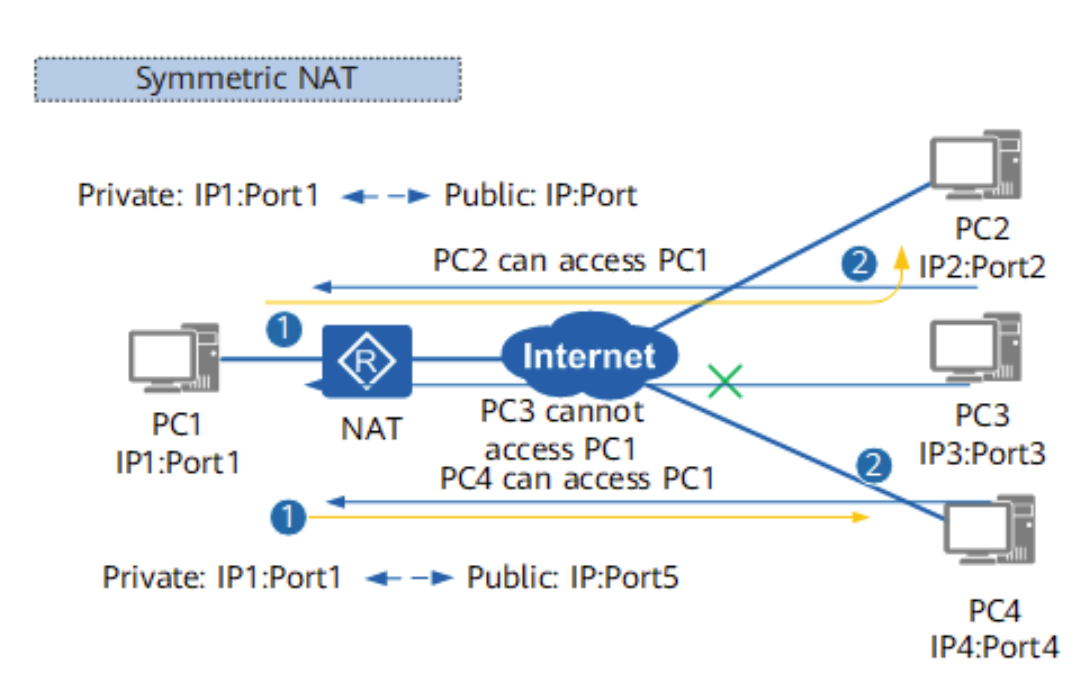
也就是说：和Restricted-cone NAT相比较，多了一个端口约束，如下图所示，IP2:Port3发给IP:Port的包将会被NAT设备丢弃，因为NAT设备记录里面仅有IP2:Port2记录，并没有Port3的记录：



Symmetric NAT（对称型NAT）

这类NAT难以穿透。

因为从相同的内网IP和端口向同一个外网目标IP和端口发送，被映射到相同的外网IP和端口，也就是说，这个映射基于 <内网IP端口；目标外网IP和端口> 的，不同的目标外网IP和端口，属于不同的映射。如下图所示，PC1的IP1:Port1对IP2:Port2和对IP4:Port4发包，NAT设备建立了两个不同的映射，分别映射到端口Port和端口Port5，也就是PC2机器看到的是PC1的映射IP:Port，而PC4看到的是PC1的映射IP:Port5。PC3则无法和PC1通讯，因为PC1并没有主动给PC3发送数据包：



### 我们的NAT类型识别

基于UDP协议，边缘节点分别对我们的两个服务器发送UDP数据包，两个服务器获得边缘节点的公网IP和端口，进行比较，如果他们看到的边缘节点的公网IP和端口是一致的，则边缘节点的NAT设备是锥形的；如果他们看到的边缘节点的公网IP和端口不一致，则边缘节点的NAT设备是对称型的。

如果边缘节点的NAT设备是锥形的，我们继续以下流程：

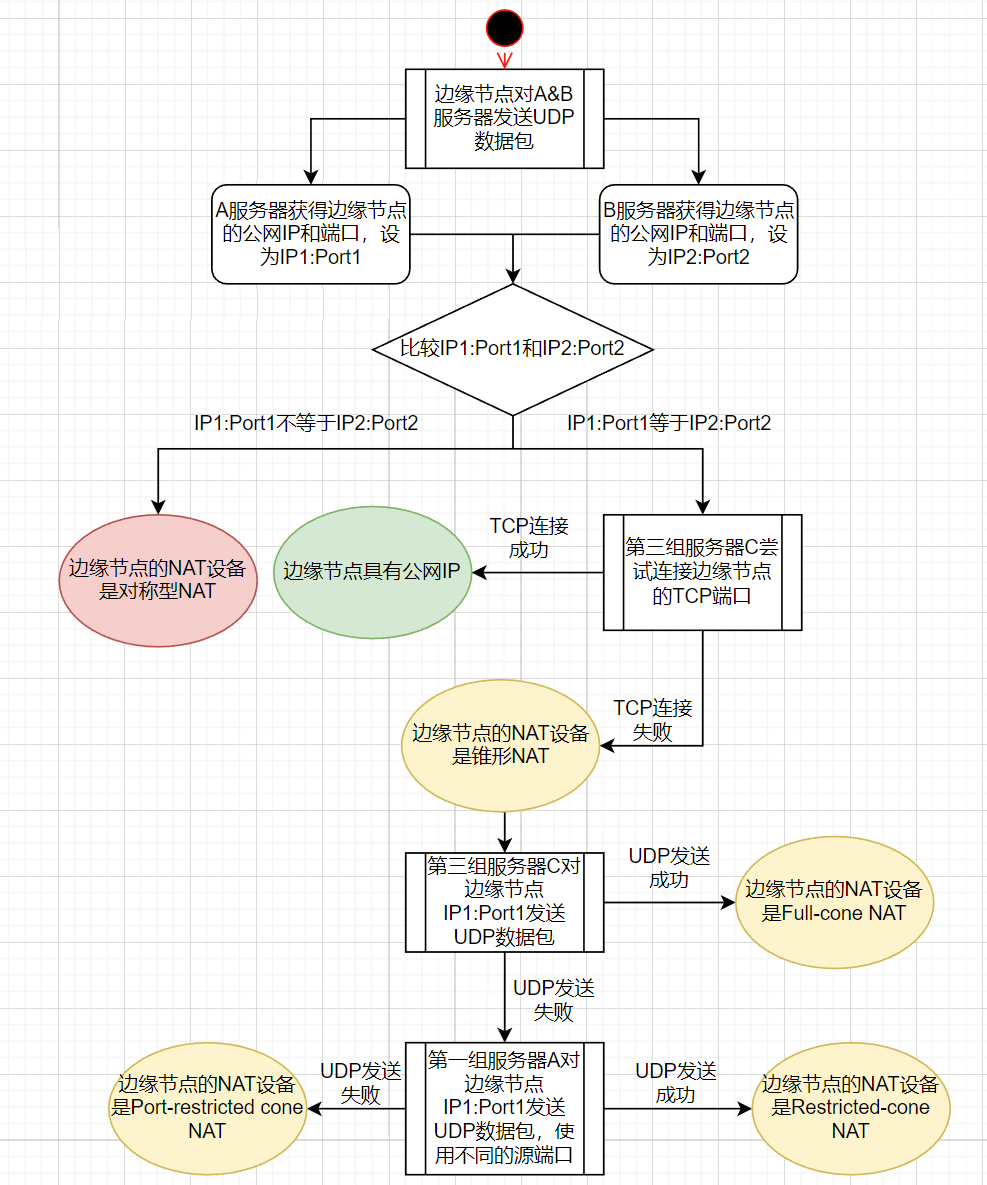
我们利用第三组服务器对边缘节点的TCP端口进行TCP连接，如果连接成功建立，表明边缘节点属于公网IP。

我们利用第三组服务器对边缘节点的公网IP和端口发送UDP数据包，如果发包成功，则边缘节点的NAT设备是Full-cone类型的。

我们让第一组服务器以不同的端口给边缘节点的公网IP和端口发送UDP数据包，如果成功，表明边缘节点的NAT设备是Restricted-cone类型的。

如上两个测试都失败时，我们认为边缘节点的NAT设备是Port-restricted cone类型的。

用下图来总结识别流程：



### 我们的穿透实现

Edge节点和用户一般都是运行在家庭网络环境，他们一般需要经过两层NAT设备，如下图所示：

（edge节点）……（edge节点的家庭NAT设备）……（edge节点的运营商NAT设备）……（公网）……（用户的运营商NAT设备）……（用户的家庭NAT设备）……（用户）

为了提高NAT穿透成功率，我们建议在（edge节点的家庭NAT设备）进行固定映射，包括SNAT（source NAT）和DNAT（destination NAT）设置映射，消除（edge节点的家庭NAT设备）的NAT，这样edge节点就只需要穿越一层NAT设备：

（edge节点）……（edge节点的运营商NAT设备）……（公网）……（用户的运营商NAT设备）……（用户的家庭NAT设备）……（用户）

用户端使用我们的SDK，确认他的NAT类型。用户请求资源所在的edge节点链接信息时，scheduler会提供edge节点的NAT类型。

然后用户端使用我们的SDK利用QUIC数据包，和edge节点建立链接，流程如下：

1. Edge节点的NAT类型是公网IP，用户端直接连接edge节点
2. Edge节点的NAT类型是Full cone NAT，用户端直接连接edge节点
3. 用户是公网IP，用户请求scheduler告知edge节点，edge节点链接用户
4. 用户是Full cone NAT，用户请求scheduler告知edge节点，edge节点链接用户
5. 任意一方是Restricted cone NAT，用户请求scheduler告知edge节点进行相互发送数据包，成功穿透并建立链接
6. 双方都是Port restricted cone NAT，用户请求scheduler告知edge节点进行相互发送数据包，成功穿透并建立链接
7. 一方是Symmetric NAT，另一方是Symmetric NAT 或者Port restricted cone NAT，用户请求scheduler告知edge节点进行相互发送数据包并猜测对方可能的端口，尝试若干次，猜对就能建立链接；否则放弃建立链接

## 下载流程

用户如何获得edge节点的访问方式

我们的SDK在浏览器环境获取carfile流程

1. 用户使用我们的SDK拉取carfile
2. SDK初始化时借助scheduler确定自身NAT类型
3. SDK向locator请求carfile的可用edge节点
4. Locator根据用户端的IP，选择若干个scheduler，并询问他们carfile的edge节点
5. 拥有该carfile的scheduler返回可用的edge节点（以及凭证）列表给用户端
6. SDK根据自身的NAT类型，以及edge节点的NAT类型，决定是否需要NAT穿透才能建立connection
7. SDK的connection builder利用edge节点列表，构建connections
8. SDK选择connections，并发拉取carfile

要从edge节点拉取内容，用户需要提交服务凭证。Edge节点验证凭证的有效，才会提供下载服务。

服务凭证由scheduler生成，每个scheduler有独立的密钥对，因此生成的凭证也不同。

用户使用服务凭证获得下载授权，edge节点使用凭证获得工作量证明。

## 下载工作量证明

### 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签名，确保凭证是scheduler签发的。

凭证的数据块包括以下内容：

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

### Asset访问凭证的生成

用户发起资源请求，locator依据就近原则把请求转发给某个titan scheduler。

Titan scheduler检查用户是否满足资源的访问限制，如果不满足则给用户发送错误信息。

Titan scheduler查找资源所在的edge节点，资源一般都具有多个副本，分别存储于多个edge节点。

Titan scheduler选择若干edge节点，并为每一个edge节点生成asset访问凭证。

### 工作量证明

用户端提交的工作量证明，使用scheduler的public key加密，包含如下内容：

凭证的唯一ID

Edge节点ID

下载速度

下载数据量

下载开始时间

下载结束时间等等

Edge节点提交的工作量证明，使用scheduler的public key加密，并使用自己的private key签名，包括如下内容：

访问凭证的唯一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

一个最终有效的工作量证明，当且仅当edge节点和用户都提交了相同的且有效的工作量证明才被承认，也就是说，仅仅是edge节点，或者用户，提交了有效工作量证明，是不会被计入edge节点的贡献。

### 关于作弊的考虑

#### 存储作弊

Titan 网络每天都会抽查edge节点的blocks，每次抽查随机选择若干blocks，edge节点需要完整地发送这些blocks给titan网络的L1节点，L1节点根据接收到blocks计算hash并把hash发送给titan scheduler，由scheduler确认这些blocks数据是否正确。这个机制要求edge节点必须正确地存储所有指派给他的blocks。换而言之，edge节点难以存储作弊。

#### 带宽作弊

前文所述，titan网络前期我们是通过带宽抽查来计算edge节点的带宽贡献，带宽抽查是在存储抽查的基础上完成的，也就是edge节点完成存储抽查后，L1节点就可以计算出edge节点发送blocks的带宽。除非edge节点能够控制对他进行抽查的L1节点，让这个L1节点给出一个虚假的带宽，但是L1节点是多个，而抽查edge节点的L1节点是随机指派的，因此edge节点 难以进行带宽作弊。

但是当edge节点的带宽越来越多地用于响应用户的资源下载请求后，我们需要通过用户的下载报告，来计算edge节点的带宽贡献。如果某个用户端，是恶意的edge节点控制的，那么这个用户端，就可以报告一个虚假的带宽，使得该恶意的edge节点能够带宽作弊。

我们目前采用的方法是，通过提高作弊的难度来抑制。由于资源位于多个edge节点，被edge节点控制的用户端，他每次请求资源时，scheduler指派给他的边缘节点列表是随机的，并不一定包含该恶意的edge节点。因此恶意的edge节点，需要多次尝试，才能获得作弊机会。后面我们会结合更多方法，例如使用TEE(Trusted Execution Environment)等方法，进一步提升带宽作弊难度。

# Edge节点updater

Updater是一个独立运行的daemon程序，用于更新edge节点程序。

我们的edge节点，有一部分是盒子设备，不方便人工更新程序，因此需要一个updater来完成节点程序的更新。

Edge节点的管理者通过登录titan网站，在他的节点管理页面，点击“更新”按钮后，titan scheduler会给节点上的updater程序发送请求，后者把新程序下载到备用区，然后通过修改Linux 软连接，使得节点程序符号指向备用区的新的节点程序，重启节点程序完成更新。