M1 network Project Whitepaper

Beta 0.3

author: loew

Contents

■ Abstract

Meta 1 network (M1 network/M1N) is a decentralized cloud rendering platform for GameFi, virtual live streaming and panoramic social scenes. It naturally supports the data design of the blockchain where every chain and every user can quickly realize and experience the high-definition and full-realistic visualization metaverse service. It will be the infrastructure of metaverse in Web3.

The development of the metaverse until today is inseparable from the decentralized thought and its core economic system. In the future, every chain may derive its own metaverse ecology and civilization. M1N project is committed to assisting every chain in achieving that. We also provide every user pass-through service in metaverse through the edge cloud rendering container. Players can switch services among different metaverses in M1N by using their own tokens.

M1N takes you into the metaverse era of high-definition and full-realistic visualization.

■ Challenges

Since the development of the blockchain based metaverse, most of games are rendered based on WebGL instead of open metaverse services. The vast majority of developers currently still develop games based on engines such as Unity or UE. Compared with the traditional development of game service, there are a series of challenges resulting in the slower development of metaverse at present:

1. Insufficient services for the metaverse cannot quickly form an ecosystem.

The ecological formation of an edge issuance to this chain is a very slow process. Because the essence of application construction: migration of developers, is based on this chain. This seriously hinders the ecological formation of the chain. And the lack of application ecology will prevent more common users from entering the corresponding metaverse ecology quickly, thus forming Rashomon.

2. Traditional game developers cannot quickly enter the blockchain space.

There are currently millions of GameFi players and developers, yet there are merely thousands of GameFi. The reason for traditional game developers not being able to quickly convert to GameFi developers is that there exist many problems in design, operation, rendering…when developing GameFi. Those problems prevent game developers from developing more GameFi.

3. The economic model draws more attention than the game experience.

At present, GameFi developers pays more attention to the construction of its economic model rather than rendering and interaction, which are still relatively primitive. Most of PC browser based GameFi are rendered through WebGL. There always runs too many computer calculations or over-long loading time when users are playing the game. The game device also affects the compatibility of some games. For example, after FLASH was discontinued, WebGL was adopted more often. But browsers have memory limits for JS. The memory limit for JS in Chrome X86 is 1.4GB. When running large games, players would gain poor computer experience, which results in a high rate of user churn.

4. Insufficient infrastructure leads to poor gaming experience.

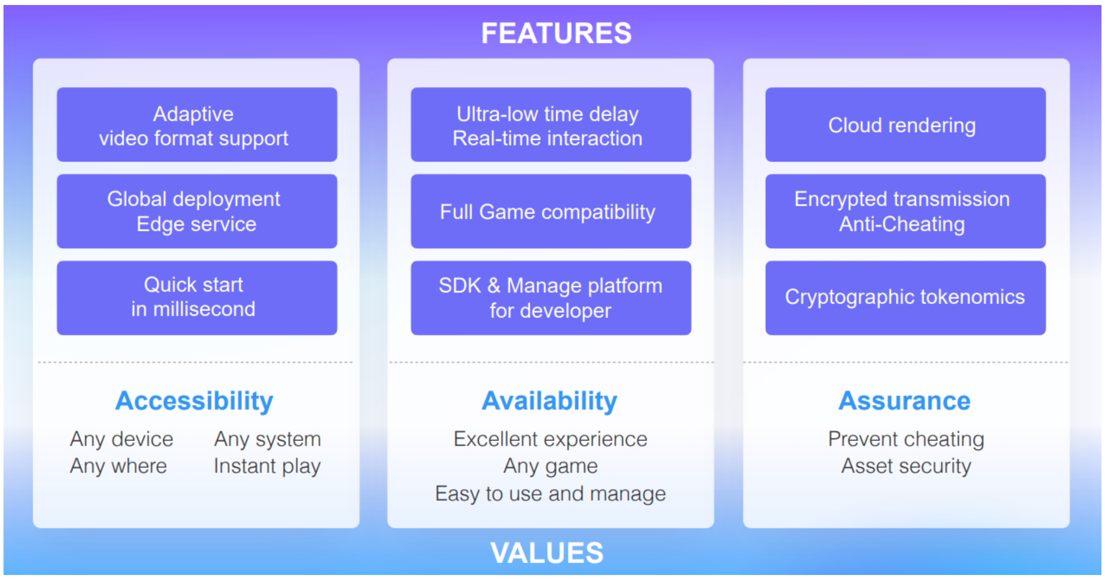
The development and operation of the game is complicated, especially for the current metaverse expansion game, whose balance and experience are strictly controlled. If it is merely based on Chrome or game blockchains, it will not develop into an immersive metaverse game. Currently, GameFi’s basic infrastructure cannot meet the requirements of virtualization and rendering for the future metaverse games. It would be difficult for players to gain better experience.

Let’s take the ray tracing effect as an example. When there exists the effect, we can get a higher sense of proximity from the game. Everything our human eyes can see is reflective or luminous, even the gleaming dark universe. We especially need ray tracing for the reality of an object. Accordingly, we need to adopt DirectX 11 or a higher version, or the Nvidia RTX or SDK package, etc. Only using WebGL may prevent players from gaining better experience. In the future, GameFi is in need of huge amounts of rendering resources to improve its experience.

■ Solution

M1N solve the problem of rapid construction of the chain ecology by designing a container. Through the container, both PC game developers and mobile game developers can quickly enter into the process of metaverse game development, and combine their own economic models with services to provide players with the best experience. We also provide rapid construction of the initial metaverse ecology for a chain. All developers can deploy their services in different chains, forming a basic ecology for each chain. We deploy a batch of containers managed through a decentralized peer-to-peer (P2P) network and each of them is a metaverse service operating environment where the necessary computing resources, storage resources and rendering resources are supplied. Metaverse developers develop their own services just like how they develop traditional games and then deploy the services in our containers. The services are running in browsers, mobile terminals, TV sets, and VR glasses and the metaverse users don’t need to install applications. When players start the game, we will select the nearest container node to provide service for players to experience it through interactive video.

■ Features



✧ High Efficiency

M1N Project helps any blockchain establish its own metaverse ecology, enabling metaverse players to enter according metaverse civilization quickly. M1N Project provides metaverse players with an interactive video channel that can be connected to metaverse through edge cloud rendering, and realizing the rapid development of the blockchain and metaverse ecology.

✧ No Installation

M1N Project provides containers for metaverse developers to deploy services. The interaction between the player and the service is realized through an interactive video stream. Metaverse players can enjoy services without installing any client. M1N Project is installation-free for any player in any metaverse.

✧ Low Latency

M1N Project manages metaverse containers through a huge network. When a player starts to play in the metaverse, we will choose the nearest service node to provide the metaverse rendering service. The metaverse player’s operation and interaction with the metaverse can be controlled within 30ms. We provide low-latency metaverse services.

✧ Superior Quality

M1N Project mainly realizes metaverse experience and interaction through interactive video streams. We can provide users with 720P, 1080P and 4K resolutions, and can provide different experience versions of 30FPS, 60FPS, 90FPS and 120FPS.

✧ Decentralization

M1N Project adopts decentralized management. The container of M1N Project adopts P2P network to manage the edge scheduling of users and metaverse. Users contribute their own computing power to serve many other metaverse users, combining with P2P scheduling capabilities, the decentralized management of containers is achieved. Meta 1 Token (M1 Token) records transactions between containers and users based on blockchain technology.

✧ High Fusion

M1N Project provides metaverse developers and users with services that can develop and connect to the metaverse. We provides basic visualization and interactive services for every metaverse, allowing developers to develop their own metaverse ecology easily and rapidly and allowing users to try all metaverse services without some special device.

■ Compared Render Farm

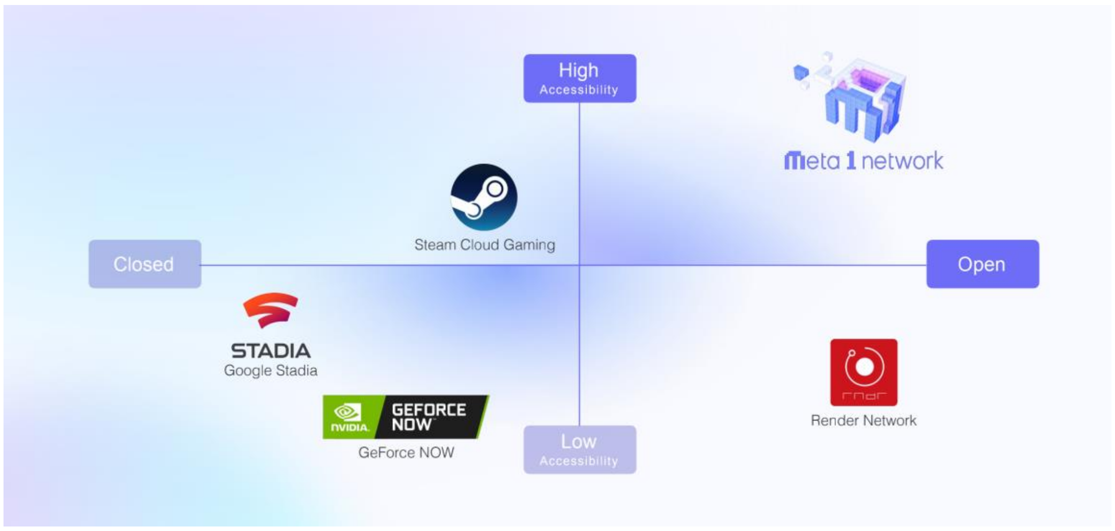
✧ What is Render Farm

Render farm is actually a popular name. In fact, we should call it “distributed parallel set computing system”, which is a parallel computing system built by using ready-made CPU or GPU, Ethernet and operating system. It is usually used for offline rendering in CG or traditional animation industries to achieve the visualization of model or action. It decomposes tasks and renders pictures frame by frame. When there is need of lots of computing power, multi-machine parallel computing is usually adopted and then the tasks are combined as a whole. The render farm is characterized by good rendering image quality and low timeliness. But it has certain limitation: it can only change image quality with time rather than interacting in the rendering process to achieve real-time rendering.

✧ What is Real-time Rendering

Real-time rendering (lightweight rendering) is that the computer renders the data into a picture for each frame, and then presents it on the screen. The data of each frame is constantly changing, so the picture of each frame is constantly moving. It can manipulate and interact in real time and process 3D images at a very high speed, achieving realistic effects at the same time. Real-time rendering pays more attention to real time and interactivity. Usually, the scene needs to be optimized, so as to improve the calculation speed and shorten the latency. M1N is a real-time rendering container which can make the picture in metaverse service visible and interactive to users in real time through the interactive video stream.

■ Competitive Product Analysis



✧ RNDR

As a blockchain based offline GPU render farm, RNDR mainly provides rendering services for CG content. This project is a distributed offline parallel computing system. Users output video from the platform by submitting CG rendering requirements, which is not an interactive live video. RNDR is aimed at offline rendering market but M1N is aimed at real-time interactive rendering market.

✧ Google

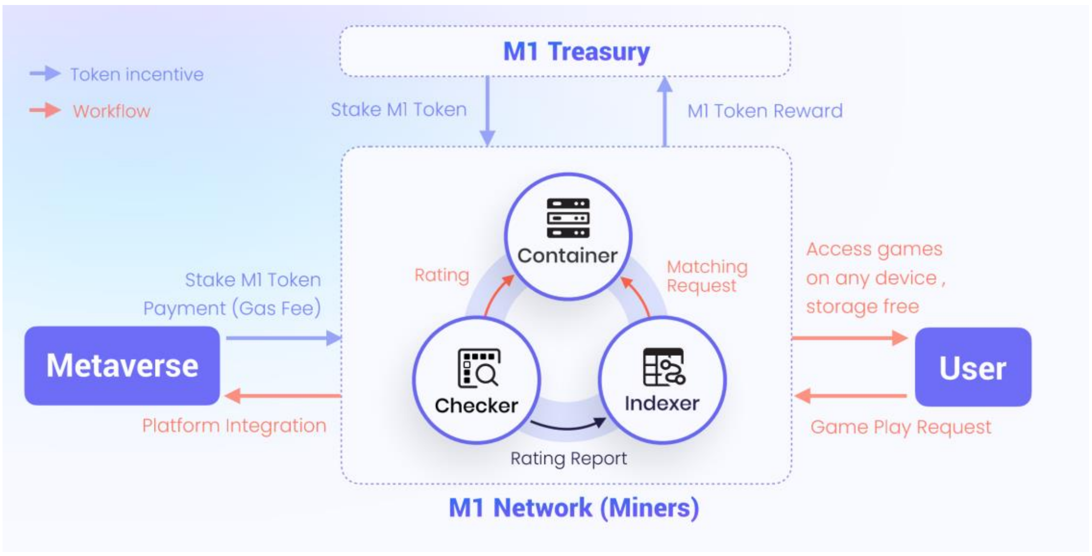
Google currently launches a centralized service for cloud gaming and it is a pure cloud gaming store which can introduce games to provide players with cloud gaming services. The main difference is that M1N not only provides players with metaverse services rendering, but also provides developers with basic real-time interactive rendering services that completely connecting publishers and players. More importantly, Google's cloud gaming platform is centralized but M1N is completely decentralized. M1N is the most authentic metaverse real-time rendering service platform in Web3.

■ M1 network Project Technology

◆ Architecture Design

M1N Project takes C/C++ as the main body to develop the container and puts the metaverse service on the server to run. It transfers the rendered metaverse audio and video images to the terminal in the form of interactive video streaming, and the metaverse client no longer needs to be installed in terminal such as TVs, mobile phones, PCs, and tablets. We don't need to worry about how the metaverse adapts to different software and hardware platforms and if the terminal performance can meet its requirements or not, etc.

◆ Application Workflow



M1N Project is an infrastructure project that connects users with metaverse. It is committed to bringing more and more users a better metaverse service experience through low-cost methods, and accelerating the development of metaverse talents and infrastructure. First of all, every metaverse service developer can publish services to M1N, and accelerate their own metaverse service rendering and distribution by purchasing a certain container. M1N will reward metaverse service developers to give them greater motivation to provide better services. Container providers contribute their own idle resources to provide better rendering resources for edge metaverse players, and they will also obtain income. When a metaverse player finds a service he wants to experience, he can experience it by paying a certain fee and can gain revenue from the service.

◆ Technical Difficulties

Real time: The overall latency of the metaverse service includes service logic operation time, audio and video rendering time, encoding latency, network transmission latency, client-side decoding latency, and the latency of client sending control information to the server. The real-time interaction of metaverse must reach the level that can meet players' requirements, which poses great technical challenges. It is also affected by the performance of the hardware and the network to some degree. It will be difficult to achieve real time without enough bandwidth.

Virtualization: Virtualization is very mature on the server-side. We have virtual machine technology and various container technologies, although it is not that mature on the desktop. Ordinary virtual desktops do not support GPU virtualization, yet metaverse service rely heavily on GPU rendering. M1N cannot be achieved without GPU virtualization, making virtualization a big technical bottleneck.

Decentralization: M1N's operation and maintenance management are different from traditional servers, because the server hardware that we use is different. M1N adopts fully decentralized container management, and the hardware load is very high. This poses new challenges to container management and these problems must be solved technically.

◆ Platform Selection

Among all the platforms that metaverse can run on, Windows is more suitable. Although the Linux platform is open, it has no metaverse service support. Other console game platforms are basically closed technologies. The Android platform is also very suitable for metaverse. The concept of running an Android metaverse service on the server and then transferring it to the Android device looks strange, but actually metaverse service providers like the concept very much. Many devices don’t allow the installation of third-party applications and the monitoring is stricter, so we use the cloud to bypass this limitation. It is very helpful for closed hardware products. There are two main routes for the virtualization technology of Windows games. One is the virtual machine solution. The main problem is that the GPU virtualization technology is immature. It may require some professional graphics card support. The cost is very high and the performance loss is very large. Every game running an according Guest OS is a waste of memory. The plan was rejected by us. Lack of available container-level technology on Windows leave us no choice but using API Hook to manually implement virtualization. We call it the Sandbox solution. The Sandbox solution hooks over all the system APIs used by the game, making the game thinks that it is running on a normal OS. But actually the OS has already taken over by us. The advantage is the performance loss is very small and there is basically no additional loss. What makes it painful is to adapt to each API and each game. Those game are usually not open-source and game developers may not cooperate with you to modify the code, so some hack techniques are used to adapt to each game. M1N will deploy and run metaverse services that are from Windows, Linux, Android and other terminals.

◆ Audio & Video Technology

The video stream uses H.264/H.265/VP9 encoding and mainly 720P/1080P@30fps, 1080P@60fps. Audio coding uses AAC. Since the standard encapsulation format does not contain control flow and cannot transmit user operation data, we have defined an encapsulation format, simply encapsulating the raw streams of H.264 and AAC and other formats and sending them to the client. M1N has made special optimizations for audio and video.

First, avoid using B frames to reduce the latency; make larger GOP settings to reduce the proportion of I frames, ensuring that the bit rate consumed by each frame is within a maximum controllable range; make zero latency setting to ensure that every time a frame of data is input, the encoder will immediately output the encoded data of this frame, avoiding the encoder to buffer the frame data; adopt bitrate control. Using a fixed bit rate algorithm is not suitable, because there always exist still pictures in the game for a period of time. The bit rate is very low at that time. The variable frame encoder will allocate a large number of bits to encode, causing extremely large data of this frame and then additional lagged network data transmission. Thus, we adopted an adaptive algorithm to ensure that the overall bit rate is within the maximum range and meanwhile the bit rate consumed by each frame is within a maximum controllable range, ensuring that the data transmission latency of each frame is controllable. For some hardware, turn off the buffer and directly output the picture, and use software decoding to support zero latency output. The performance of Android devices is uneven and the performance of early low-end chips cannot meet real-time decoding. GPUs are used for extra acceleration. In network transmission, use TCP to make optimizations at the network layer for latency reduction and combine with the current RTC protocol.

◆ Game Developer & Publisher

The release of M1N based metaverse service is relatively simple. Metaverse developers only need to access the SDK and release the metaverse service according to the M1N operating documents. The basic process is as follows:

- The metaverse service team can independently choose the commonly used 3D engines on the market to develop their own metaverse services, such as Unity or UE.
- After the main part of the metaverse service is developed, it can be connected to the corresponding wallet and payment channel based on the signaling channel provided by M1N.
- Package the metaverse service into an apk on the mobile terminal or an exe on the desktop.
- Publish the metaverse service installation package to the M1N metaverse center through the M1N publishing platform.
- M1N generates a web page or h5 address for preview or formal operation interface.
- The metaverse service team can distribute their own M1N metaverse entry and enter.
- Metaverse players can open the address to log in or pay through the wallet, and experience their own metaverse life.
- All player operations are sent to the metaverse service in the container through signaling, cloud container, and cloud container proxy.
- The cloud container generates a real-time video stream to realize the real-time interaction between the player and metaverse service container.

◆ Command Channel

As a container-level design, M1N Project can use data channel capabilities to establish communication between the client and cloud applications. The cloud application creates a UDP server and monitors a UDP port (localhost 127.0.0.1, the recommended range is 10000-20000), and starts to wait to receive UDP packets. The metaverse rendering client calls the metaverse cloud rendering SDK interface to create a transparent transmission channel, and the target port parameter in the SDK interface is the port monitored by the cloud. The cloud rendering client first sends a custom data packet, and the cloud application UDP will receive the request and parse out the local proxy port. The obtained local proxy port sends a custom data packet which will be returned to the client application through the created data channel.

➢ UDP server example：

Sample code:

```
int main() {
        int udp_socket_fd = socket(AF_INET, SOCK_DGRAM, 0);
        if (udp_socket_fd == -1) {
                printf("socket failed!\n");
                return -1;
        }
        struct sockaddr_in bind_addr = { 0 };
        bind_addr.sin_family = AF_INET;
bind_addr.sin_port = htons(xxxx);
bind_addr.sin_addr.s_addr = inet_addr("0.0.0.0");
        int ret = bind(udp_socket_fd, (struct sockaddr *)&bind_addr, sizeof(bind_addr));
        if (ret < 0) {
                perror("bind fail:");
                close(udp_socket_fd);
                return -1;
        }
struct sockaddr_in    upstream_addr = { 0 };
int len = sizeof(upstream_addr);
        char buf[1024] = { 0 };
        while (true) {
        ret = recvfrom(udp_socket_fd, buf, sizeof(buf), 0, (struct sockaddr *)
& upstream_addr, &len);
                if (ret == -1) {
                        break;
                }
              const char* response = "response";
                sendto(udp_socket_fd,   response,   strlen(response),   0,   (struct   sockaddr   *)   &
upstream_addr, sizeof(upstream_addr));
        }
        return 0;
}
```

➢ Front-end example:

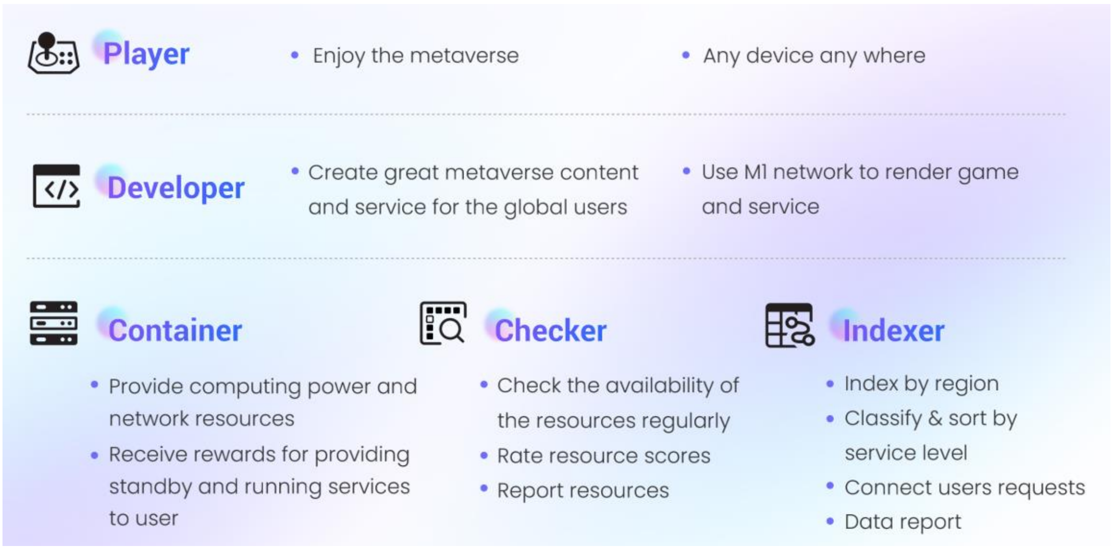
Smaple code:

```
(async _ => {
        const onMessage = msg => {
                console.log("recvice data:", msg);
        };
        const result = await new Promise((resolve, reject) => {
                const timer = setInterval(async _ => {
                        const ret = await M1NSDK.createCustomDataChannel({
```

```
                    destPort: xxxx, onMessage
                });
                if (ret.code == 0) {
                    resolve(ret);
                    clearInterval(timer);
                }
            }, 2000);
        });
        if (result.code == 0) {
            result.sendMessage('test');
        }
        result.sendMessage(`${custom_data}`);
})();
```

■ M1 Token Design

◆ Participant Role



Player
- Enjoy the metaverse
- Any device any where

Developer
- Create great metaverse content and service for the global users
- Use M1 network to render game and service

Container
- Provide computing power and network resources
- Receive rewards for providing standby and running services to user

Checker
- Check the availability of the resources regularly
- Rate resource scores
- Report resources

Indexer
- Index by region
- Classify & sort by service level
- Connect users requests
- Data report

M1N Project is an open and decentralized metaverse infrastructure platform. The main participants are:

Container: M1N Container provide computing power and network resources. It receive rewards for providing standby and running services to users.

Indexer: M1N Indexer is an indexer that mainly provides container information for M1N Container, and realizes the communication of containers between Game Player and Game Publisher.

Checker: M1N Checker verify the validity of the container to ensure the reasonable compliance of the container.

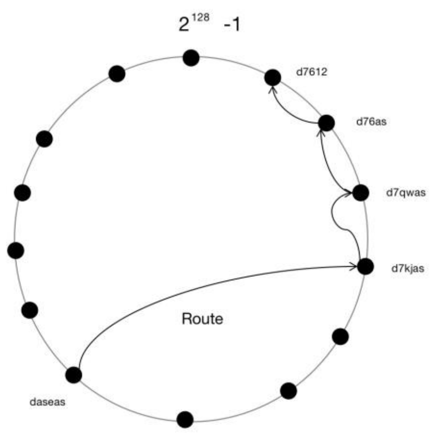
Game Developer: M1N Game Developer is the main metaverse service contributor in M1N. It develops its own metaverse service based on various existing game engines and deploys the service on M1N. It provides the service to metaverse players and obtains corresponding income. It can also purchase M1N Container to provide better metaverse experience for players.

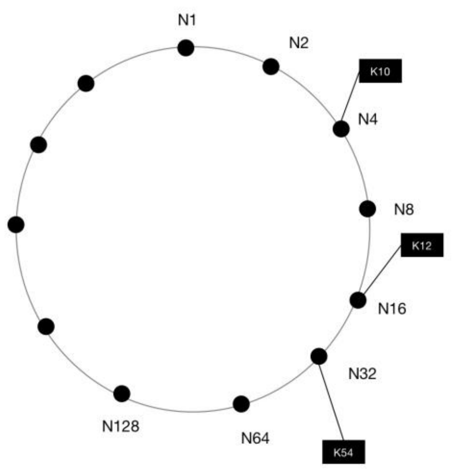
Game Player: M1N Game Player is the main user group of M1N. After metaverse service developers release their services through M1N Container, Game Player can experience the service through M1N and obtain its own revenue. M1N Game Player will also be the main consumer and acquire more rendering resources by purchasing container.

Let’s take a simple process as an example. First, the container contributor deploys its own idle resources to the M1N Container, meaning that the Container is waiting for the access list. At this time, Checker initiates a calculation instance run by the Container and the combination is returned to Checker. After the Checker has passed the verification, the Container enters the standby state, indicating that the Container can provide services to the outside world. The information will be sent to the Indexer data through the Checker. Indexer will index a distributed hash table (DHT) and be updated. When a metaverse service publisher or developer needs a container, it will obtain DHT routing table and find corresponding Container and deploy corresponding game in the Container. When a game player enters, he can directly pull up the game to play the game.

◆ Dispatch Algorithm

The Container of M1N adopts the P2P method based on decentralized structured topology DHT. The DHT is actually a huge hash table maintained by a large number of dynamic nodes in a wide area. It is divided into discrete blocks. Each node is assigned to a hash block of its own, and becomes the manager of this hash block. Decentralization and atomic self-organization have become two important goals of design. Through the cryptographic hash function, the name or keyword of an object is mapped to a 128-bit or 160-bit hash value. All nodes in a DHT based system are mapped to a space, for example, a hash function mapping a bit name to a hash value.





The DHT structure can adapt to the dynamic joining/exiting of nodes, and can show good scalability, robustness, uniformity of node ID distribution, and self-organization capabilities. The overlay network adopting deterministic topology enable DHT to find the destination node as long as it exists in the network and to guarantee its accuracy. The most classic cases are Tapestry, Chord, CAN, and Pastry.

The biggest problem of the DHT structure is that its maintenance mechanism is more complicated. Especially the network fluctuation (Churn) caused by frequent joining/exiting of nodes will greatly increase the maintenance cost of DHT. Another problem is that DHT only supports exact keyword matching queries, and cannot support complex queries such as content/semantics. The management of P2P in M1N is based on the accounting method of the blockchain itself and combined with the characteristics of DHT to develop DHT on blockchain. We use the record information of blockchain as a routing table of DHT. When there are enough routing tables, the efficiency of single-point query is guaranteed and complex content and semantic queries can be achieved through blockchain's current analysis tools. Combining with the location information, the blockchain Node and the Peer Route Table's location information are bound so as to realize the quick query for the closest container location and to realize container's positioning and scheduling.

◆ Rendering Algorithm

M1N not only manages the rendering node that based on blockchain, but also deals with the process of the single container rendering interactive video. When the physical link between the container and the game player is close to the logical network link, we can render the user's image quality in the form of interactive video without considering if the game application is installed locally or not, and output 720P or 1080P or 4K resolution to balance user resources and user experience. The maximum real-time interaction capability is achieved by constructing the video rendering downlink channel and the user interaction uplink channel. M1N Project supports the rendering of 360-degree VR content: H264/265 and VP8/9 coding and decoding for ordinary video, AAC or other coding and decoding format for audio, ERP or other coding and decoding for VR content.

M1N Project adopts three-level rendering technology: the first level is engine-level rendering technology; the second level is container-level rendering technology; the third level is video-level rendering technology. They correspond to the running state, output state and experience state of the game respectively. Engine-level rendering technology mainly provides rendering image quality at the data-driven stage. Container-level rendering technology provides image quality effects from the video codec level. Video-level rendering technology provides experience from the transmission and terminal decoding levels.

Engine-level rendering technology:

M1N Project mainly solves the rendering interaction problems of Unity, UE and other commonly used game engines. Compared with popular GameFi, M1N provides more game engine choices. M1N Project implements OpenGL, Direct3D, GDI and other low-level rendering engines at the bottom, and makes a lot of optimization in light reflection. We assume that three points are created in an empty three-dimensional space. A plane has at least three points. The object is composed of different planes. In graphics, triangles are generally utilized. Assuming that in a three-dimensional space, if you want to see a triangular surface in a two-dimensional display, you need to render the triangle from the three-dimensional space to the two-dimensional space. In the real world, triangles scatter light in all directions. For a browser-level WebGL, it is impossible to have the resources to calculate all rays of light in different directions. The computer-rendered image only calculates the light scattered to the direction of the human eye, which is also called the camera direction. By adding a camera in the 3D space and placing a screen network at the front perspective point, each frame is a pixel of the rendered image, drawing a ray that intersects the camera perspective point. If these rays intersect on the screen, the screen can observe this triangle, then mark this triangle border. We only render the pixels in the overlapping part between the border and the pixels without processing the other parts, and we get a triangular image. This method of projecting to the pixel grid is one of the simple rendering methods. Through this example we can understand that points and planes are the core elements that affect the resources required for rendering. M1N Project adopts AI algorithms to greatly optimize the point-plane relationship in the rendering process to reduce the plane without affecting the rendering effect. We put efforts in making the light splitting more suitable for the relationship between the human eye and the screen to achieve visual artifacts and the rendering effect of the final output video. Considering that most of current game resources are based on the point-plane infrastructure as the game asset, we develop the M1N Augmented Reality technology on the basis of reducing the plane and increasing the points in order to achieve the ultimate engine-level effect improvement.

Container-level rendering technology:

M1N Project mainly solves the terminal and optimal real-time encoding technology. At present, the commonly used video coding includes H265/H264/VP8/VP9 and so on. Since the characteristics of different terminals are different, we use Android as an example. Hardware decoding is to call the special module code of GPU to reduce CPU operation, which has lower requirements on CPU and other hardware. Software decoding requires CPU operations, which increases the burden on the CPU and power consumption. Hardware decoding is to hand over part of the video data that originally was all processed by the CPU to the GPU. Since the parallel computing power of the GPU is much higher than that of the CPU, the load on the CPU can be greatly reduced and then some other programs can be run at the same time. For Android devices, the most commonly used SOCs are Qualcomm, HiSilicon and MediaTek. Most of these SOCs integrate a lot of functions, such as CPU, GUP, DSP, ISP, video decoding, audio decoding, etc. Thus, when we mention SOC, we are not only refers to the CPU.

Advantage of Hardware Decoding: It is more power-saving, suitable for long time mobile video plays and live streaming. It is better to use hardware decoding with limited battery of mobile phone. It can also reduce CPU usage by giving CPU to other threads, which is conducive to the fluency of the phone.

Advantages of Soft Decoding: It has better adaptability. Software decoding does not need to consider about the hardware decoding support of the device but mainly occupies the operation of CPU. It is very performance-consuming and power-consuming. It is better to use software decoding when the device has sufficient power or when the hardware decoding support of the device is insufficient.

We adopt adaptive codec solutions, matching the characteristic parameters of the terminal with the following container and automatically scheduling to the most optimized encoding container to achieve the best terminal experience, ensuring the rationality and experience of video encoding.

Video-level rendering technology:

M1N Project mainly solves the problem of anti-weak network. Latency is a concern for everyone and our research found that players have different requirements in latency for different types of games. The requirement for FPS game latency is stricter. If it exceeds 100ms, players may not accept it. The requirement for RPG game latency is within 500ms, and the requirement for real-time strategy game latency is within 1000ms (one second), which is still acceptable. In terms of transmission, we added WebRTC in addition to the original RTSP. WebRTC provides low-latency and P2P communication, which is more suitable for the application scenario of container. Our test found that in a weak network environment with about 3% packet loss, WebRTC will have better image quality and lower latency. WebRTC is open-source and supported by mainstream browsers. It is also cross-platform without requiring plug-ins. Native applications can be accessed as long as they implement WebRTC, which is also the benefit of using the WebRTC.

In conclusion, M1N rendering algorithm adopts a three-level construction from the engine level to the video level and then to the transmission level, combining with the edge scheduling algorithm to jointly control the latency of M1N below 50ms, bringing better experience for game players.

◆ Token Design

M1 Token adopts exponential decay method. We can define the exponential decay formula as

$$-\frac{\mathrm{d}N}{N} = \frac{\mathrm{d}t}{\tau}$$

This [formula] is a number with a time dimension, and its meaning will be explained later. In fact, we can interpret this formula from another angle by changing it a little bit:

$$-\frac{\mathrm{d}N}{\mathrm{d}t} = \frac{N}{\tau}$$

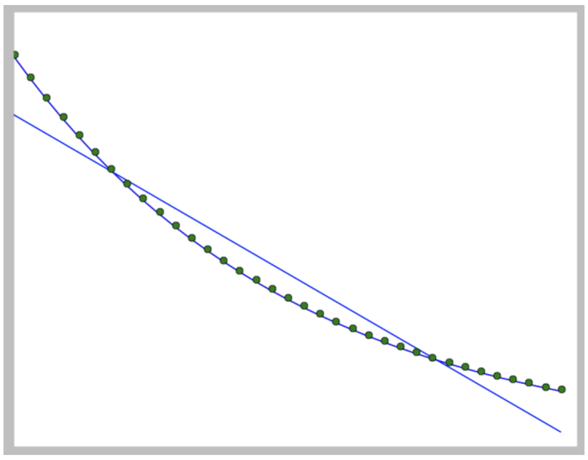
We can find that the left side of the equal sign is the rate of decrease in the number of tokens, and the negative sign means decrease. The right side of the equal sign is [formula] times the current token number. In other words, the rate of token decay is proportional to the number of tokens. We can get the function between the number of atoms [formula] and time [formula].

$$N(t) = N_0 e^{-\frac{t}{\tau}}$$

The number of tokens decreases with time in a negative exponential manner of [formula], and we obtain [formula] from the formula through definition. In this way, the meaning of [formula] is very obvious, which is that the initial number of tokens is 100000000. We use $T_{1/2}$ to indicate the half-life is 4 years, then we can deduce:

$$\frac{N_0}{2} = N(T_{1/2}) = N_0 e^{-\frac{T_{1/2}}{\tau}}$$

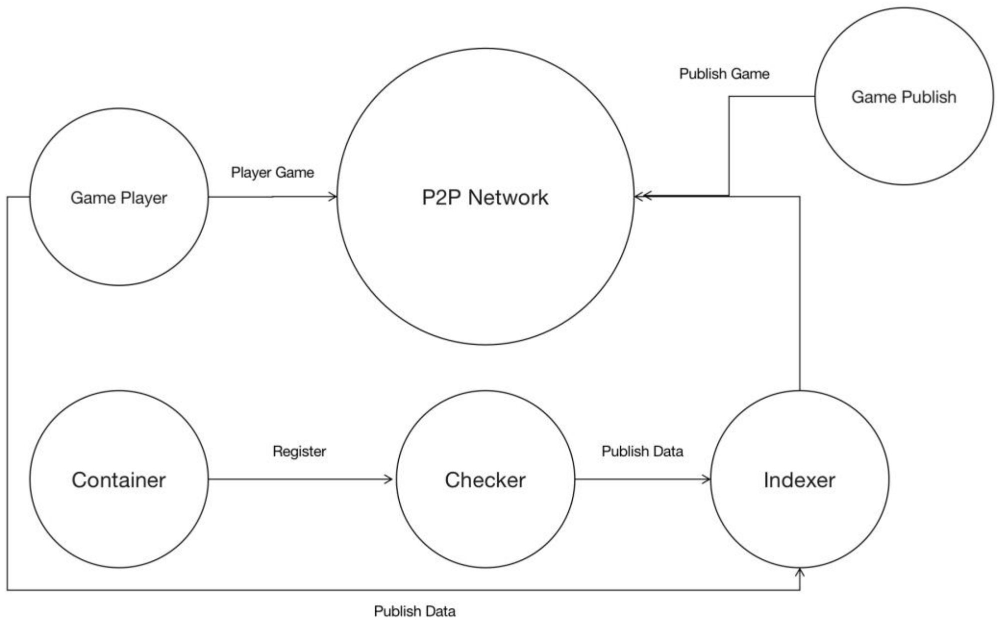
The specific curve diagram is as follows:


（Release Curve）

Through this release curve, we can add more participation to the M1N Project in the early stage, and ensure M1 Token’s value and sustainability by reducing 1/2 of the release within a 4 year’s half-life.

◆ Consensus Algorithm

M1N Project adopts improved proof of stake (PoS), which is a consensus algorithm that is more environmentally friendly and greener than proof of work (PoW). PoS mechanism changes the competing computing power in the PoW mechanism to system equity. The greater the equity, the greater the probability of becoming the next bookkeeper. PoS doesn’t require mining and consuming of a lot of electricity and energy. It is more decentralized than PoW type cryptocurrencies such as Bitcoin. PoS requires the purchase of 51% of the currency rather than costly 51% computing power attack of the PoW algorithm, and requires no attack. The cryptocurrency of the PoS mechanism adds new currencies at a certain annual interest rate to avoid inflation or austerity, effectively maintaining the basic stability.



M1N Project will be based on a hybrid model of variable computing power + Indexer + Checker. Take our basic workflow as an example. After the user deploys the M1N Container, it is verified through Checker and the routing information is stored through Indexer. The entire process manager, Container, starts to join the network, and then Checker will check the standby container every 15 minutes. If the Checker is not checking after the Container enters the service state, Game Player will synchronize the Checker information to the Indexer. Game Publish can selectively purchase Containers based on standby Container on P2P Network.

■ Find Out More

➢ Official website

www.m1nk.io