نویسنده: ایوب کریمی

ایمیل نویسنده: ayubkarimi587@gmail.com

منتشر شده توسط

کوین ایران

کوین ایران بزرگترین پایگاه خبری فارسی زبان در حوزه فناوری بلاکچین، ارزهای رمزنگاری شده و پلتفرم های مرتبط با بلاکچین است. این وب سایت در سال ۱۳۹۲ توسط بابک جلیلوند و آرش محبوب، با هدف اطلاع رسانی، آموزش و مشاوره به جامعه فارسی زبان علاقه مند به رمز ارزها راه اندازی شد و اولین مقاله آن در ۱۴ دی ماه همان سال، با عنوان «بیت کوین چیست؟» منتشر گردید . هدف کوین ایران از معرفی این فناوری ایجاد محیطی پژوهشی و آموزشی در راستای استفاده صحیح از این فناوری جهت ارائه تسهیلات و رفاه به جوامع فارسی زبان است.

www.coiniran.com

متاکوین ها

ایده پشت یک متاکوین، داشتن پروتکلی است که در بالای بیت کوین مستقر است و از تراکنش های بیت کوین برای ذخیره تراکنش های متاکوینی استفاده می‌کند، اما عملکرد انتقال فازی متفاوتی دارد و آن APPLY می‌باشد. از آنجا که پروتکل متاکوین نمی‌تواند از تراکنش های متاکوینی غیر معتبر جلوگیری کند و نمی‌تواند از ظهور آنها در بلاکچین بیت کوین ممانعت به عمل آورد، یک قاعده اضافه گردیده و اگر APPLY (S,TX) خطای برگشت دهد، پروتکل در حالت اصلی خود یعنی APPLY (S,TX)=S قرار می‌گیرد.

این یک مکانیزم آسان را برای ایجاد پروتکل رمزارز اختیاری، فراهم می‌کند که به طور بالقوه ویژگی های پیشرفته ای دارد که نمی‌تواند داخل خود بیت کوین پیاده شود؛ اما بدین شیوه با هزینه ی توسعه خیلی پایین ایجاد می‌شود زیرا پیچیدگی های استخراج و شبکه بندی از قبل توسط پروتکل بیت کوین اعمال و اجرا شده اند.

بنابراین به طور عمومی، دو رویکرد در مورد ساختن یک پروتکل اجماع وجود دارد؛ یکی ساختن یک شبکه مستقل و دیگری ساختن پروتکلی در بالای بیت کوین. رویکرد اول، در حالی که به طور معقولی در مورد اپلیکیشن هایی مانند Namecoin موفق بوده است، برای پیاده سازی و اعمال مشکل می‌باشد.

هر پیاده سازی شخصی نیازمند این است که خودش یک بلاکچین مستقل را راه اندازی کند، علاوه بر آن ساختن و تست همه تراکنش فازی ضروری و کد شبکه بندی را نیز عهده دار می‌باشد. به علاوه مجموعه اپلیکیشن هایی برای فناوری اجماع غیر متمرکز پیش بینی می‌شود و این یک توزیع قانونی قدرت را به دنبال خواهد داشت که در آنجا اکثریت عظیمی از اپلیکیشن ها بسیار کوچکتر از آن می‌باشند که بتوانند بلاکچین خود را ضمانت کنند.

همچنین توجه داشته باشید که در آنجا طبقه های عظیمی از اپلیکیشن های غیر متمرکز موجود هستند، مخصوصا سازمان های خودمختار و غیر متمرکزی که نیاز است با همدیگر تعامل داشته باشند. از طرف دیگر، این رویکرد مبتنی بر بیت کوین، این عیب را دارد که ویژگی های تایید پرداخت تسهیل شده بیت کوین را به ارث نمی‌برد.

SPV برای بیت کوین کار می‌کند زیرا می‌تواند از عمق بلاکچین به عنوان یک پروکسی (نایب) برای اعتبار استفاده کند. در جاهایی که سوابق یک تراکنش به اندازه کافی دور می‌شوند، در این حالت گفته می‌شود آنها به طور قانونی بخشی از فاز مربوطه می‌باشند.

از طرف دیگر پروتکل های متا که بر مبنای بلاکچین هستند، نمی‌توانند بلاکچین را مجبور کنند که تراکنش هایی را که در فضای پروتکل خودشان معتبر نیستند، مشمول گرداند. بنابراین پیاده سازی پروتکل SPV به شیوه کاملا ایمن، لازم است که همه مسیر را به طرف عقب جستجو کند تا به اندازه بلاکچین بیت کوین برسد و تعیین کند که آیا تراکنش های خاصی معتبرند یا نه.

در حال حاضر همه پیاده سازی های جزئی متا پروتکل های بر مبنای بیت کوین متکی به سرور قابل اعتمادی هستند که داده هایی را فراهم کند که به طور قابل دفاعی می‌توان گفت که بسیار به استاندارد نزدیک است، مخصوصا با توجه به آنکه یکی از مقاصد ابتدایی رمزارز ها این است که نیاز به اعتماد را از میان بردارد.

اسکریپت

پروتکل بیت کوین حتی بدون هیچ گونه الحاقی، نسخه ضعیفی از قرارداد های هوشمند را تسهیل می‌کند. UTXO در بیت کوین فقط با یک کلید عمومی مالکیت پیدا نمی‌کند، بلکه همچنین اسکریپت پیچیده تری در زبان برنامه نویسی stack-based ساده ابراز می‌شود.

در این الگو، تراکنشی که آن UTXO را خرج می‌کند، باید داده هایی را فراهم کند که اسکریپت را قانع کند. براستی حتی مکانیزم مالکیت کلید عمومی اصلی، از طریق یک اسکریپت پیاده می‌شود. این اسکریپت یک امضای منحنی بیضوی را به عنوان ورودی اعمال می‌کند که آن را در برابر تراکنش و آدرسی که مالک UTXO است، تایید می‌کند و اگر تایید موفق باشد، 1 و در غیر این صورت 0 ارجاع داده می‌شود.

اسکریپت های پیچیده تر دیگری برای موارد استفاده اضافی متعددی موجود هستند. به عنوان مثال فرد می‌تواند اسکریپتی بسازد که نیازمند امضا هایی شامل دو از سه کلید خصوصی می‌باشد تا اعتبارسازی انجام شود. (multisig) این چیدمان برای اکانت های سازمانی، اکانت های پس انداز ایمن و موقعیت های شخص ثالث بازرگان، مفید است.

اسکریپت ها همچنین می‌توانند برای پرداخت بخشش هایی که برای راه حل های محاسباتی داده می‌شوند، استفاده شوند و فرد می‌تواند حتی اسکریپتی بسازد که چیزی مانند این که UTXO بیت کوین متعلق به چه کسی است، نشان داده شود. اگر بتوان یک مدرک SPV فراهم کرد که فرد یک تراکنش Dogecoin را فرستاده است، در اساس این امر مجوز یک مبادله متقابل غیر متمرکز رمزارز را می‌دهد.

اما زبان اسکریپت، آنچنان که در بیت کوین پیاده شده، محدودیت های مهمی هم دارد:

عدم کفایت کامل تورینگ

می‌توان گفت در حالی که زیر مجموعه عظیمی از محاسبه موجود است که زبان اسکریپت بیت کوین آن را پشتیبانی می‌کند، اما نمی‌توان گفت که همه چیز را پشتیبانی می‌کند و طبقه اصلی که مفقود می‌شود، حلقه ها هستند. این برای اجتناب از وجود حلقه های بی شمار در حین تایید تراکنش انجام می‌شود.

از لحاظ تئوری این مانع برای برنامه نویسان اسکریپت برطرف کردنی است زیرا هر حلقه می‌تواند با تکرار ساده کد زیربنایی برای چندین بار، شبیه سازی شود ولی در این کار اما و اگری وجود دارد و آن این که آنها تبدیل به اسکریپت هایی می‌شوند که خیلی از لحاظ بافتی ناکارامد می‌باشند.

به عنوان مثال پیاده سازی یک الگوریتم امضای منحنی بیضوی دیگر، احتمالا نیازمند 256 نوبت ضرب کردن تکراری می‌شود که همگی به طور انفرادی در کد قرار می‌گیرند.

پنهان شدن بها

محال است که یک اسکریپت UTXO، کنترل مطلوبی را بر روی مقدار برداشتی ایجاد کند. به عنوان مثال یک مورد استفاده مهم از یک قرارداد oracle، قرارداد hedging می‌باشد که در آن A و B با هزار دلار ارزش بیت کوین کنار می‌آیند و اسکریپت بعد از سی روز، معادل هزار دلار بیت کوین را به A و بقیه را به B می‌فرستد.

این امر نیازمند یک پایگاه داده خواهد بود که ارزش بیت کوین را به دلار تعیین کند، اما حتی در آن صورت بهبود عظیمی بر حسب اعتماد و ضرورت های زیربنایی به نسبت راه حل های کاملا مرکزی که الان موجود است، صورت می‌گیرد. اما از انجا که UTXO مکانیزم همه یا هیچ دارد، تنها راه برای رسیدن به این امر از طریق هک ناکارامد داشتن تعداد زیادی UTXO یا واحد های پولی متنوع است (برای مثال یک UTXO که 2^k است و هر k تا 30 می-تواند باشد). پایگاه داده باید تایید کند که کدام UTXO باید به A و کدام باید به B فرستاده شود.

فقدان مرحله

UTXO می‌تواند خرج شده و یا خرج نشده باشد؛ موقعیتی برای قرارداد ها و یا اسکریپت های چند مرحله ای وجود ندارد که هر مرحله داخلی دیگر، فراسوی آن را نگه دارند. این امر ساخت قرارداد های انتخاب های چند مرحله ای، پیشنهاد های مبادله غیر متمرکز و یا پروتکل های تعهد رمزنگاری دو مرحله ای را سخت می‌کند (برای بخشش های محاسباتی ایمن لازم است) .

این همچنین به این معنی است که UTXO می‌تواند فقط برای ساخت قرارداد های ساده یکبار برای همیشه به کار برده شود و برای قرارداد های پیچیده که رکورد را ثبت می‌کنند مانند سازمان های غیر مرکزی، مورد استفاده قرار نمی‌گیرد و این سبب می‌شود که پیاده سازی متاپروتکل ها سخت باشد. ترکیب حالت باینری با پنهان سازی بها، به معنی اپلیکیشن مهم دیگری می‌باشد که در آن محدودیت برداشت غیر ممکن است.

پوشیدگی بلاکچین

UTXO برای داده های بلاکچین مانند نانس و هش قبلی، پنهان است. این امر شدیدا برای اپلیکیشن های قمار و چندین طبقه دیگر محدودیت ایجاد می‌کند که این کار با محروم کردن زبان اسکریپت یک منبع باارزش بالقوه از قابلیت تصادفی بودن انجام می‌پذیرد. بنابراین سه رویکرد در زمینه ساختن اپلیکیشن های پیشرفته در بالای رمزارز دیده می‌شود. ساختن یک بلاکچین جدید، استفاده از اسکریپت در بالای بیت کوین و ساختن یک متاپروتکل در بالای بیت کوین، سه رویکرد مذکور هستند.

ساختن یک بلاکچین جدید باعث آزادی نامحدودی در ایجاد مجموعه ویژگی های خاصی می‌شود اما به قیمت صرف زمان برای توسعه و تلاش برای خود-راه اندازی تمام می‌شود. استفاده از اسکریپت برای پیاده سازی و استاندارد سازی آسان است اما از لحاظ قابلیت ها و متاپروتکل ها بسیار محدود است، بنابراین با وجود سهولت، از عیب هایی در مقیاس پذیری رنج می‌برد.

با استفاده از اتریوم، قصد داریم که چارچوب تعمیم یافته ای را بسازیم که ویژگی های هر سه الگوی فوق الذکر را به طور همزمان ایجاد کند.

اتریوم

هدف اتریوم این است که مفاهیم اسکریپت، آلتکوین ها و متاپروتکل های روی زنجیره را ادغام کند و بهبود بخشد و سبب شود که توسعه دهندگان بتوانند اپلیکیشن های اجماع- محور اختیاری را ایجاد کنند که این اپلیکیشن ها مقیاس پذیری و استاندارد سازی داشته باشند و مجموعه کاملی از ویژگی ها را داشته باشند و آسانی توسعه و قابلیت همکاری اجزا با هم را به طور همزمان ارائه دهد.

اتریوم این کار را با ساختن آنچه که در اساس یک لایه فوندانسیون انتزاعی نهایی است، انجام می‌دهد. این لایه یک بلاکچین با زبان برنامه نویسی کامل تورینگ است که در داخل آن تعبیه شده است و به هر فردی اجازه می‌دهد که قرارداد های هوشمند و اپلیکیشن های غیر مرکزی را ایجاد کند که در آنجا افراد می‌توانند قوانین اختیاری خود برای مالکیت، فرمت تراکنش و عملکرد های انتقال فازی، ایجاد کنند.

نسخه ساده ای از Namecoin می‌تواند در دو سطر کد، نوشته شود و پروتکل های دیگر مانند واحد پول ها و سیستم های اعتباری می‌توانند با کمتر از بیست سطر کد نوشته شوند. قرارداد های هوشمند که باکس هایی رمزنگاری شده هستند و محتوی ارزش هستند و فقط زمانی باز می‌شوند که شرایط معینی حاکم باشد، همچنین می‌توانند در بالای پلتفرم اتریوم ساخته شوند و قدرت آن بسیار عظیم تر از آنچه است که توسط اسکریپت بیت

کوین ارائه می‌شد، زیرا قدرت های اکمال تورینگ، آگاهی ارزشی، آگاهی بلاکچینی و قابلیت فازی به آن افزوده شده است.

اکانت های اتریوم

در اتریوم، قابلیت فازی متشکل از چیز هایی است که اکانت نامیده می‌شوند و هر اکانت یک آدرس 20 بایتی داشته و انتقال های فازی دارد که در واقع انتقال ارزش و اطلاعات بین اکانت ها می‌باشد. یک اکانت اتریوم محتوی چهار زمینه است:

1- نانس که شمارنده ای است که برای حصول اطمینان از این که هر تراکنش فقط یک بار پردازش می‌گردد، استفاده می‌شود.

2- موجودی حال حاضر اتر در اکانت

3- کد قرارداد اکانت اگر موجود باشد.

4- ذخیره اکانت که در حالت اصلی و اولیه خالی است.

اتر سوخت داخلی و اصلی شبکه اتریوم است و برای پرداخت هزینه تراکنش ها به کار می‌رود. به طور کلی دو نوع اکانت در این سیستم وجود دارد. نوع اول اکانت های با مالکیت خارجی است که با کلید های خصوصی کنترل می-شوند و نوع دوم اکانت های قراردادی که با کد قراردادشان کنترل می‌شوند.

یک اکانت با مالکیت خارجی، کد ندارد و فرد می‌تواند پیام هایی را از چنین اکانتی با ایجاد و امضای یک تراکنش، بفرستد. در یک اکانت قراردادی هر زمان که چنین حسابی پیامی را دریافت می‌کند، کدش فعال می‌شود و سبب می‌گردد که آن در ذخیره داخلی خوانده و نوشته شود و در عوض پیام های دیگری را بفرستد و یا قرارداد هایی را ایجاد کند.

پیام ها و تراکنش ها

پیام ها در اتریوم چیزی شبیه به تراکنش ها در بیت کوین هستند، اما سه تفاوت اساسی دارند. اول، یک پیام اتریومی می‌تواند هم توسط موجودیت خارجی و هم توسط یک قرارداد ایجاد شود، در حالی که یک تراکنش بیت کوینی فقط می‌تواند توسط موجودیت خارجی ایجاد شود.

دوم این که برای پیام های اتریومی یک انتخاب صریح موجود است که داده ها را مشمول کند و سرانجام دریافت کننده یک پیام اتریومی، اگر یک حساب قراردادی باشد، این اختیار را دارد که جواب دهد و این بدین معنی است که پیام های اتریومی همچنین مفهوم عملکرد ها را در احاطه خود قرار می‌دهند.

اصطلاح تراکنش در اتریوم، برای اشاره به بسته داده های امضا شده به کار می‌رود که پیامی را ذخیره می‌کند تا از یک حساب با مالکیت خارجی ارسال شود. تراکنش ها شامل دریافت کننده پیام، یک امضا که فرستنده را شناسایی می‌کند، مقدار اتر و داده هایی که باید فرستاده شود، می‌باشد و علاوه بر اینها شامل دو بها که STARTGAS و GASPRICE نامیده می‌شوند نیز هستند.

به منظور جلوگیری از گسترش نمایی و داشتن حلقه های بی نهایت در یک کد، هر تراکنش لازم است که محدوده ای را تعیین کند برای این که بداند چند مرحله محاسباتی از اجرای کد را می‌تواند گسترش دهد که این شامل هم پیام اصلی باشد و هم هر پیام اضافی دیگری که در خلال اجرا تولید می‌شود.

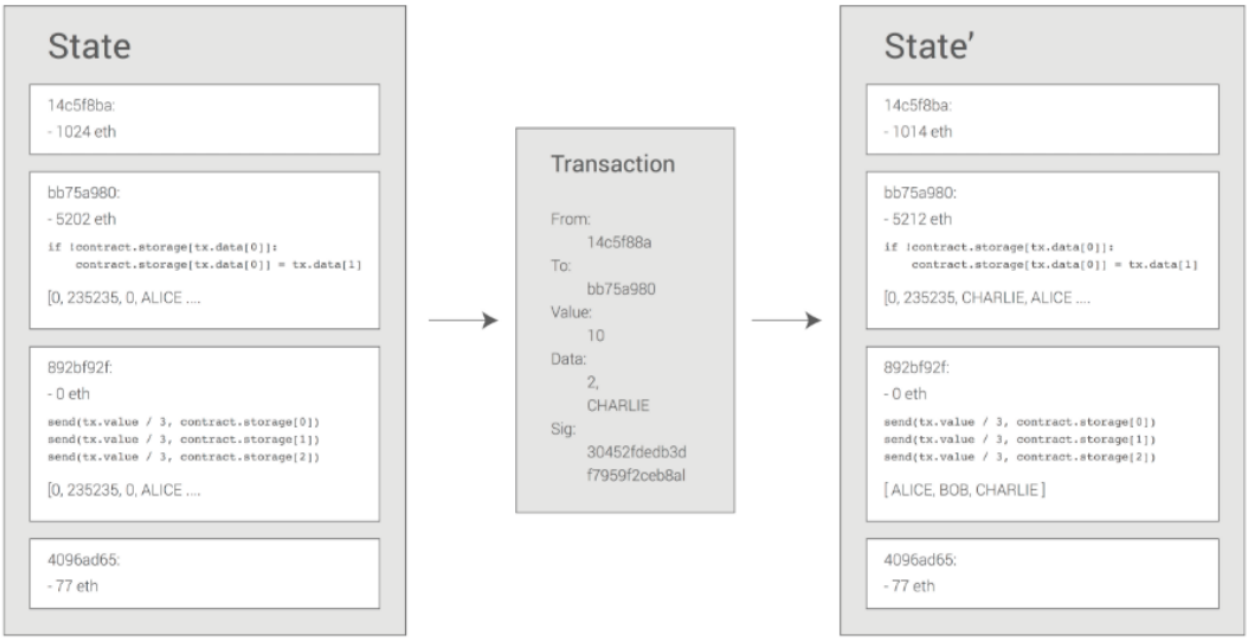
STARTGAS این محدوده ذکر شده است و GASPRICE هزینه ای است که باید به استخراج گر برای هر مرحله محاسباتی پرداخت گردد. اگر gas برای اجرای تراکنش تمام شود، همه ی تغییرات فازی برگشت داده می‌شوند به جز پرداخت هزینه ها و اگر اجرای تراکنشی متوقف شود و این در حالی است که تعدادی gas باقی مانده باشد، در آن صورت بخش باقی مانده هزینه ها به ارسال کننده پس داده می‌شود.

همچنین یک نوع تراکنش جداگانه هم وجود دارد و یک نوع پیام مطابق با آن، که برای ایجاد یک قرارداد به کار می‌روند و آدرس قرارداد بر اساس هش نانس اکانت و داده های تراکنش محاسبه می‌گردد.

یک پیامد مهم مکانیزم پیام، خصوصیت "شهروند درجه یک بودن" اتریوم است و آن این ایده است که قرارداد ها قدرت یکسانی با حساب های خارجی دارند، از جمله توانایی ارسال پیام و ایجاد قرارداد های دیگر. این سبب می-شود که قرارداد ها به طور همزمان چندین نقش مختلف را ایفا کنند، برای مثال یکی از آنها می‌تواند داشتن یک عضو از یک سازمان غیر مرکزی (یک قرارداد) و یک حساب شخص ثالث باشد (قرارداد دیگر).

مثال مذکور بین یک فرد ناایمن که امضاهای Lamport ضد کوانتم سفارشی را به کار می‌برد (قرارداد سوم) و موجودیتی است که در امضا مساعدت می‌کند و خودش از اکانتی با پنج کلید برای امنیت استفاده می‌کند (قرارداد چهارم). قدرت پلتفرم اتریوم این است که سازمان غیر متمرکز و قرارداد شخص ثالث، لازم نیست نگران این باشند که هر طرف قرارداد چه نوع اکانتی دارد.

عملکرد انتقال فازی اتریوم



عملکرد انتقال فازی اتریوم 'APPLY(S,TX) -> S می‌تواند به صورت زیر تعریف شود:

1- بررسی کنید که آیا تراکنش به خوبی شکل گرفته (یعنی تعداد بهای درستی دارد)، امضا معتبر است و نانس مطابقت دارد با نانس اکانت فرستنده؛ اگر چنین نباشد باید خطای برگشت داده شود.

2- هزینه تراکنش را به صورت STARTGAS * GASPRICE محاسبه کنید و از امضا آدرس فرستنده را تعیین کنید. هزینه را از موجودی اکانت فرستنده کسر کنید و نانس فرستنده را افزایش دهید. اگر موجودی کافی برای خرج کردن موجود نیست، خطا اعلام کنید.

3- GAS = STARTGAS را آغاز کنید و برای هر بایت مقدار معینی gas را آزاد کنید تا هزینه بایت ها در تراکنش پرداخت شود.

4- بهای تراکنش را از اکانت فرستنده به اکانت دریافت کننده انتقال دهید. اگر اکانت دریافت کننده هنوزموجود نیست، آن را ایجاد نمایید. اگر اکانت دریافت کننده یک قرارداد است، کد قرارداد را اجرا کنید تا زمانی که تکمیل می‌شود و یا تا هنگامی که gas تمام می‌شود.

5- اگر انتقال بها به دلیل این که فرستنده پول کافی ندارد، با شکست مواجه شد و یا در اجرای کد gas تمام شد، همه تغییرات مرحله ای به جز پرداخت هزینه ها را برگردانید و این هزینه ها را به اکانت استخراج گر اضافه نمایید.

6- در غیر این صورت، هزینه ها را برای همه gas باقی مانده به اکانت فرستنده باز گردانید و هزینه هایی که سبب مصرف gas شده را به اکانت استخراج گر ارسال نمایید.

برای مثال فرض کنید که کد قرارداد به شرح زیر است:

:[[if !contract.storage[msg.data[0

[contract.storage[msg.data[0]] = msg.data[1

توجه داشته باشید که در واقعیت کد قرارداد به صورت کد EVM سطح پایین نوشته شده است. این مثال به صورت Serpent نوشته شده که زبان سطح بالای ما می‌باشد و برای شفافیت، می‌توان آن را به کد EVM ترجمه کرد. فرض کنید که ذخیره سازی قرارداد در ابتدا خالی است و یک تراکنش به ارزش 10 اتر، 2000 gas و 0.001 قیمت gas فرستاده می‌شود و دو زمینه داده نیز موجود هستند: [2, '3']['CHARLIE']

فرآیند لازم برای عملکرد انتقال فازی در این مورد به صورت زیر است:

1. بررسی کنید که تراکنش معتبر بوده و به خوبی شکل گرفته باشد.

2. بررسی کنید که ارسال کننده تراکنش حداقل 2000*0.001=2 ether داشته باشد. اگر چنین بود، 2 اتر را از اکانت فرستنده کم کنید.

3. gas=2000 را شروع کنید، با این فرض که تراکنش 170 بایت طول دارد و هزینه بایت 5 است، 850 را از آن کم کنید و 1150 gas بر جای می‌ماند.

4. ده اتر دیگر را از حساب ارسال کننده کم کنید و آن را به اکانت قرارداد اضافه کنید.

5. کد را اجرا کنید. در این مورد ساده است و آن بررسی می‌کند که آیا ذخیره سازی قرارداد در شاخص 2 به کار رفته است یا نه. توجه داشته باشید که اگر به کار نرفته، ذخیره سازی را در شاخص 2 و به بهای CHARLIE تنظیم کنید. فرض کنید که این 187 gas می‌خواهد، بنابراین مقدار gas باقی مانده به صورت 963=187 -1150 می‌باشد.

6. 963*0.001=0.963 اتر را دوباره به اکانت فرستنده اضافه کنید و فاز نتیجه را اعلام کنید.

اگر قراردادی در طرف دریافت کننده تراکنش موجود نمی بود، هزینه کل تراکنش به صورت ساده مساوی با GasPrice فراهم شده ضربدر طول تراکنش بر حسب بایت می‌شد و داده هایی که در کنار تراکنش فرستاده می-شدند، بی ارتباط می بود. به علاوه توجه داشته باشید که پیام هایی که قرارداد محرک آنها بوده است، می‌توانند یک محدوده gas را به محاسبه ای که تولید می‌کنند، اختصاص دهند.

اگر زیر مجموعه محاسبات gas را تمام کند، برگشت فقط تا نقطه اعلام پیام صورت می‌گیرد. بنابراین، دقیقا مانند تراکنش ها، قرارداد ها نیز می‌توانند منابع محاسباتی محدودشان را با تنظیم محدوده های سخت گیرانه در زیر مجموعه محاسباتی که تولید می‌کنند، ایمن نمایند.

اجرای کد

در قرارداد های اتریوم کد به زبان بایت کد stack-based و در سطح پایین نوشته می‌شود که به آن کد ماشین مجازی اتریوم و یا به اختصار کد EVM گفته می‌شود. کد متشکل از یک مجموعه بایت می‌باشد که هر بایت ارائه دهنده یک عملکرد است. به طور کلی، اجرای کد حلقه نامحدودی است که متشکل از انجام تکراری عملکرد در شمارش گر برنامه رایج (که از صفر شروع می‌شود) می‌باشد و سپس شمارش گر برنامه تا یک افزایش می‌یابد تا این که انتهای کد برسد و یا این که خطایی و یا دستورالعمل STOP و یا RETURN ظاهر شود.

این عملکرد ها به سه نوع فضا برای ذخیره سازی داده دسترسی دارند:

- اولی stack که محتوی به ترتیب عکس ورود است و در آن ارزش های 32 بایتی می‌تواند راه اندازی و وارد شود.

- حافظه که یک رسته بایتی قابل توسعه نامحدود است.

- ذخیره سازی دراز مدت قرارداد، یک انبارش کلید یا بها در جایی که کلید ها و بها ها هر دو 32 بایت هستند. بر خلاف دو مورد قبل که بعد از انتهای محاسبات دوباره تنظیم می‌شوند، ذخیره سازی برای دراز مدت ادامه پیدا می-کند.

این کد همچنین می‌تواند به بها، ارسال کننده، داده های پیام ورودی و علاوه بر آن داده های پیشرو بلوک دسترسی داشته باشد و آن کد همچنین می‌تواند یک رسته بایت از داده ها را به عنوان خروجی انعکاس دهد.

مدل اجرایی رسمی کد EVM، به طور حیرت برانگیزی ساده است. در حالی که ماشین مجازی اتریوم در حال عملکرد است، فاز محاسباتی کامل آن می‌تواند توسط مجموعه ای از داده ها تعریف شود. (block_state, (memory, stack, pc, gas,transaction, message, code که block-state فاز جهانی است و شامل همه اکانت ها، موجودی ها و ذخیره سازی می‌شود. در هر دور اجرا، دستورالعمل های رایج با به کار بردن کد pc-th byte پیدا می‌شود و هر دستورالعمل بر حسب این که چگونه مجموعه داده ها را تحت تاثیر قرار می‌دهد، تعریف خاص خود را دارد.

به عنوان مثال ADD دو گزینه را از stack خارج می‌کند و مجموعا آنها را راه اندازی می‌کند، gas را به یک تقلیل می‌دهد و pc را به یک افزایش می‌دهد و SSTORE دو آیتم برجسته را از stack می‌راند و آیتم دوم را در ذخیره سازی قرارداد بر طبق شاخصی که توسط آیتم اول تعیین شده، درج می‌کند.

به علاوه در این حالت gas تا 200 تقلیل پیدا می‌کند و pc تا یک افزایش می‌یابد. اگرچه روش های زیادی برای بهینه کردن اتریوم از طریق ترجمه به موقع وجود دارد اما یک پیاده سازی اصلی اتریوم می‌تواند در چند صد سطر کد انجام شود.

بلاکچین و استخراج

بلاکچین اتریوم تا حدود زیادی مشابه بلاکچین بیت کوین است، اگرچه تفاوت هایی هم دارد. تفاوت اصلی بین بیت کوین و اتریوم در زمینه معماری بلاکچین این است که بلوک های اتریوم بر خلاف بیت کوین شامل یک کپی از لیست تراکنش و جدید ترین فاز می‌باشد.

به جز آن، دو ارزش دیگر یعنی رقم بلوک و سختی هم در بلوک ذخیره می‌شود. الگوریتم معتبر سازی بلوک در اتریوم به شرح زیر است:

1- بررسی کنید که آیا بلوک قبلی ارجاع داده شده موجود است و آیا معتبر است.

2- بررسی کنید که رکورد زمانی بلوک بزرگتر از بلوک مرجع پیشین بوده و کمتر از 15 دقیقه ادامه داشته باشد.

3- بررسی کنید که شماره بلوک، سختی، ریشه تراکنش، ریشه uncle و محدوده gas (مفاهیم متنوع سطح پایین خاص اتریوم) معتبر باشند.

4- بررسی کنید که گواه کار در بلوک معتبر باشد.

5- اجازه بدهید S[0]، STATE-ROOT بلوک قبلی باشد.

6- اجازه بدهید TX لیست تراکنش بلوک باشد با n تراکنش. برای همه به صورت [n-1, setS[i+1...0 =APPLY(S[i],TX[i]). اگر هر اپلیکیشنی خطا اعلام کند و یا کل gas در بلوک مصرف شود، به صورتی که این نقطه از GASLIMIT تجاوز کند، خطا اعلام می‌شود.

7- اجازه بدهید S_FINAL به صورت S[n] باشد اما پاداش بلوک را که به استخراج گر پرداخت می‌شود، اضافه کنید.

8- بررسی کنید که آیا S-FINAL مشابه ROOT-STATE است، اگر چنین باشد بلوک معتبر است و در غیر این صورت معتبر نیست.

این رویکرد در نگاه اول ممکن است شدیدا ناکارا به نظر برسد زیرا لازم است که کل مرحله را با هر بلوک ذخیره کند، اما در حقیقت این کارایی قابل مقایسه با بیت کوین است. دلیل این است که این فاز در یک ساختار درختی ذخیره می‌شود و بعد از هر بلوک فقط بخش کوچکی از درخت لازم است تغییر کند.

بنابراین به طور عمومی بین دو بلوک مجاور، بخش عظیمی از درخت باید مشابه باشد و بنابراین داده ها می‌توانند برای یک بار ذخیره شوند و با استفاده از اشاره گر ها (یعنی هش های زیرمجوعه درخت ها) دوبار ارجاع داده شوند. نوع خاصی درخت که درخت Patricia نامیده می‌شود، برای دست یابی به این مقصود استفاده می‌گردد که این درخت شکل تعدیل یافته ای از مفهوم درخت Merkle است و به گره ها اجازه می‌دهد که به طور موثری نه تنها تغییر یابند بلکه درج و حذف گردند.

علاوه بر این، به دلیل این که همه اطلاعات فازی، بخشی از آخرین بلوک است، لازم نیست که کل تاریخچه بلاکچین ذخیره گردد. این استراتژی اگر می‌توانست برای بیت کوین اعمال شود، 5 تا 20 برابر صرفه جویی در فضا را برای آن به ارمغان می‌آورد.

اپلیکیشن ها

در کل، سه نوع اپلیکیشن بر بالای اتریوم موجود هستند. طبقه اول اپلیکیشن های مالی اند که برای کاربران، شیوه های قدرتمند تری از مدیریت و ورود به قرارداد ها را با استفاده از پول خود فراهم می‌آورند. این شامل زیرمجموعه پول ها، ماخوذات مالی، قرارداد های پوشش ریسک، کیف پول های پس انداز، وصیت نامه ها و در نهایت حتی تعدادی از قرارداد های استخدامی تمام عیار می‌شود.

طبقه دوم، اپلیکیشن های شبه مالی هستند که پول دخیل است اما همچنین جنبه غیر پولی عظیمی هم در مورد آن مطرح است. یک مثال خوب برای این مطلب، بخشش های بالاجباری است که برای راه حل های مسائل محاسباتی داده می‌شود. سرانجام طبقه سوم اپلیکیشن هایی مانند رای گیری آنلاین و مدیریت غیر مرکزی می-باشند که اصلا مالی نیستند.

سیستم های توکن

سیستم های توکن موجود بر روی بلاکچین، اپلیکیشن های زیادی دارند، از زیر مجموعه پول ها گرفته که شامل دارایی هایی مانند دلار آمریکا، طلا و یا سهام شرکتی می‌شوند، تا چیز هایی مانند توکن های فردی که ارائه دهنده ی اموال هوشمند هستند، کوپن های غیر قابل جعل ایمن و حتی سیستم های توکنی که هیچ رابطه ای با ارزش های مرسوم ندارند و به عنوان سیستم های امتیازی برای ایجاد انگیزه به کار می‌روند.

پیاده سازی سیستم های امتیازی در اتریوم به طور حیرت انگیزی ساده است. نکته کلیدی برای درک این است که کل یک واحد پول و یا یک سیستم توکن، در اساس یک پایگاه داده با یک عملکرد است و آن عملکرد این است که مقدار X واحد را از A کم کن و مقدار X واحد را به B بده، با این شرط که اولا فرد X حداقل X واحد قبل از تراکنش داشته باشد و دوما تراکنش توسط A تایید شود. کل آنچه که باعث پیاده سازی یک سیستم توکن می‌شود، پیاده سازی این منطق در قرارداد می‌باشد.

کد اصلی برای پیاده سازی یک سیستم توکن در Serpent مانند زیر به نظر می‌رسد:

```
from = msg.sender
[to = msg.data[0
[value = msg.data[1

:if contract.storage[from] >= value
    contract.storage[from] = contract.storage[from]  value
    contract.storage[to] = contract.storage[to] + value
```

این در اساس یک پیاده سازی دقیق عملکرد انتقال فازی در " سیستم بانکی" است که قبلا در این نوشتار مورد بحث قرار گرفت. چند سطر دیگر کد لازم است تا افزوده گردد و قدم اول برای توزیع واحد پول ها و چندین مورد جانبی دیگر فراهم گردد. در حالت ایده آل یک عملکرد باید به قرارداد های دیگر افزوده گردد تا بتوانند برای موجودی یک آدرس تفحص کنند.

اما آن کل مطلبی است که نسبت به این موضوع موجود است. در تئوری، سیستم های توکن مبتنی بر اتریوم که مانند زیر مجموعه پول عمل می‌کنند، پتانسیل این را دارند که شامل ویژگی مهم دیگری نیز شوند که متاپول های مبتنی بر بیت کوین و واقع در روی زنجیره، فاقد آن هستند و آن ویژگی مهم توانایی پرداخت هزینه های تراکنش مستقیما توسط همان واحد پول خاص می‌باشد.

این امر به شیوه ای پیاده می‌شود که قرارداد، یک موجودی اتری داشته باشد که با آن، اتری را که برای پرداخت هزینه تراکنش ها به ارسال کننده به کار رفته، پس دهد و این موجودی دوباره توسط جمع آوری واحد پول های داخلی پر شود که این امر هزینه بر است و نیازمند فروش مجدد در یک مزایده ی دائما در حال جریان می‌باشد.

بنابراین لازم است که کاربران، اکانت های خود را با اتر فعال کنند، اما زمانی که اتر موجود باشد می‌تواند دوباره مورد استعمال قرار بگیرد، زیرا قرارداد هر بار آن را پس می‌دهد.

ماخوذات مالی و پول های دارای ارزش ثابت

ماخوذات مالی رایج ترین اپلیکیشن یک قرارداد هوشمند هستند و از ساده ترین ها نیز برای پیاده سازی به صورت کدی می‌باشند. چالش اصلی در پیاده سازی قرارداد های مالی این است که اکثریت آنها نیازمند مرجع برای یک تیکر قیمت خارجی هستند. برای مثال یک اپلیکیشن خیلی مورد علاقه، قرارداد هوشمندی است که در برابر ریسک نوسانات اتر (یا هر واحد پول دیگر) پوشش امنیتی اعمال می‌کند؛ اما انجام این امر نیازمند این است که قرارداد بداند که ارزش اتر به دلار چقدر است.

آسان ترین شیوه برای انجام این امر از طریق یک قرارداد "داده های آپدیت پذیر" است که توسط طرف خاصی مانند NASDAQ پشتیبانی شود و طوری طراحی شده باشد که آن طرف بتواند هر وقت که لازم بود، آن را آپدیت کند و یک میانجی را ایجاد کند که به قرارداد های دیگر اجازه دهد که برای آن قرارداد پیام بفرستند و جوابی دریافت کنند که قیمت را برای آنها فراهم کند.

با احتساب آن عنصر مهم، قرارداد پوشش ریسک مانند زیر به نظر می‌رسد:

1- منتظر طرف A باشید تا هزار اتر را وارد کند.

2- منتظر طرف B باشید تا هزار اتر را وارد کند.

3- ارزش هزار اتر را به دلار ثبت کنید، محاسبه را با مراجعه به قرارداد داده های آپدیت پذیر انجام دهید، فرض کنیم X دلار هم به صورت ذخیره موجود است.

4- بعد از سی روز، اجازه دهید A یا B قرارداد را به منظور فرستادن X دلار به ارزش اتر به A و بقیه آن به B، مورد بررسی و پرس و جو قرار دهد (محاسبه دوباره با تفحص از قرارداد داده های آپدیت پذیر انجام می‌شود تا قیمت جدید فراهم گردد).

چنین قراردادی پتانسیل برجسته ای در تجارت رمزارز ها دارد. یکی از مشکلات اصلی رمزارز ها این حقیقت است که آنها بی ثبات و دارای نوسان هستند، اگرچه بسیاری از کاربران و تجار ممکن است امنیت و راحتی معامله با دارایی های رمزنگاری شده را خواستار باشند اما تعداد بسیاری از آنها نمی‌خواهند با چشم انداز از دست دادن 23 درصد از دارایی هایشان تنها در یک روز مواجه باشند.

تا کنون، رایج ترین راه حلی که پیشنهاد داده شده، دارایی های دارای حمایت صادر کننده بوده است. این ایده به این صورت است که صادر کننده یک پول زیر مجموعه را ایجاد می‌کند که در آن، آنها حق صدور و لغو واحد ها را دارند و یک واحد پول را برای هر کسی که به آنها یک واحد مشخص از دارایی های زیر بنایی مانند طلا و دلار (به صورت آفلاین) می‌دهد، فراهم کنند.

صادر کننده سپس تعهد می‌نماید که یک واحد از دارایی های زیربنایی را برای هر کسی که یک واحد از دارایی های رمزارز را پس می‌فرستد، فراهم کند. این مکانیزم به هر دارایی غیر رمزارز اجازه می‌دهد که تعالی یابد و به دارایی رمزارز تبدیل شود، به شرط این که صادر کننده قابل اعتماد باشد.

اما در عمل صادر کنندگان همیشه قابل اعتماد نیستند و در بعضی موارد زیربنای بانکی بسیار ضعیف است و یا برای ارائه چنین سرویس هایی، خیلی محتاط است. ماخوذات مالی گزینه دیگری را فراهم می‌کنند. در اینجا به جای وجود یک صادر کننده که هزینه ها را فراهم کند تا از یک دارایی پشتیبانی به عمل آید، یک بازار غیر متمرکز از دارندگان، شرط می‌بندند که قیمت یک دارایی مرجع رمزنگاری شده بالا می‌رود و این امر در این جا نقش ایفا می-کند.

بر خلاف صادرکنندگان، نگه دارندگان در زمینه پرداخت سر وقت معامله با شکست مواجه نمی‌شوند زیرا قرارداد پوشش ریسک، پول های آنها را در حساب شخص ثالث نگه می‌دارد. توجه داشته باشید که این رویکرد کاملا غیر متمرکز نیست زیرا هنوز هم یک منبع مورد اعتماد برای فراهم ساختن تیکر قیمت، مورد نیاز است.

اگرچه با وجود این موارد، این امر هنوز بهبود عظیمی در زمینه تقلیل نیازمندی های زیربنایی است (به جز در مواردی که فرد صادر کننده است، در دیگر موارد صادر کردن آپدیت و تغذیه قیمت نیازمند هیچ مجوزی نیست و می‌توان آن را در زمره آزادی های بیان قرار داد) و پتانسیل تقلب را کاهش می‌دهد.

سیستم های اعتبار و هویت

رمزارز جایگزین پیشگام (Namecoin)، تلاش کرد تا از یک بلاکچینِ شبیه بیت کوین، برای فراهم کردن یک سیستم ثبت نام استفاده کند که در آنجا کاربران می‌توانستند نام هایشان را در یک پایگاه داده عمومی، همراه با دیگر داده ها ثبت کنند. استفاده اصلی یک سیستم DNS همچنان که قبل ذکر شد، طرح بندی نام های دامین مانند bitcoin.org (یا در مورد Namecoin، bitcoin.bit) به یک آدرس آی پی می‌باشد.

استفاده های دیگری منجمله تایید مجوز ایمیلی و سیستم های اعتباری پیشرفته تر بالقوه نیز موجود هستند. در پایین یک قرارداد اصلی برای فراهم نمودن یک سیستم ثبت نام شبیه Namecoin در اتریوم دیده می‌شود:

```
if !contract.storage[tx.data[0]]:
    contract.storage[tx.data[0]] = tx.data[1]
```

این قرارداد بسیار ساده است و کل آن پایگاه داده ای در داخل شبکه اتریوم است که می‌تواند به آن اضافه شود، اما نمی‌توان آن را اصلاح کرد و یا برداشت. هر کسی با مقداری هزینه می‌تواند یک نام را ثبت کند که برای همیشه باقی می‌ماند. یک قرارداد ثبت نام پیچیده تر یک شرط عملکردی خواهد داشت که به قرارداد های دیگر اجازه

بررسی آن را می‌دهد و همچنین یک مکانیزم برای مالک (ثبت کننده اول) نام خواهد داشت که می‌تواند داده ها را تغییر دهد یا مالکیت را منتقل کند. فرد حتی می‌تواند عملکرد اعتبار و شبکه اعتماد را در بالا اضافه کند.

انبارش فایل غیر مرکزی

در چند سال گذشته، تعدادی شرکت انبارش فایل اینترنتی محبوب ظاهر شده اند که برجسته ترین آنها Dropbox بوده، که این شرکت ها به کاربران اجازه می‌دهند که یک پشتیبان از هارد درایو را آپلود کنند و این سرویس پشتیبان را برای آنها ذخیره می‌کند و به کاربر اجازه می‌دهد که در ازای یک هزینه ماهانه به آن دسترسی داشته باشد.

اما در این نقطه از زمان، بازار انبارش فایل نسبتا ناکارامد است. یک نگاه سریع به راه حل های متنوع موجود نشان می‌دهد که مخصوصا در حالت ناخوشایند سطح 20 تا 200 گیگ که در آن هیچ تخفیفی صورت نمی‌گیرد و هزینه ها پایین نمی‌آید، قیمت های ماهانه برای هزینه های انبارش فایل مرسوم، به صورتی است که فرد بیش از هزینه کل هارد درایو در تنها یک ماه پرداخت می‌کند.

قرارداد های اتریوم می‌تواند اجازه توسعه یک اکوسیستم انبارش فایل غیر متمرکز را بدهد، که در آنجا کاربران می-توانند مقادیر کمی پول با اجاره دادن هارد خود به دست آورند و این فضای استفاده نشده می‌تواند برای پایین آوردن قیمت انبارش فایل مورد استفاده قرار گیرد.

بخش زیربنایی کلیدی چنین ابزاری می‌تواند چیزی باشد که ما آن را " قرارداد Dropbox غیر متمرکز" می‌نامیم. شیوه کارکرد این قرارداد به این صورت است که در درجه اول فرد داده های مورد علاقه خود را به صورت بلوک هایی منشعب می‌کند و هر بلوک را برای امنیت و حریم شخصی رمز دار کرده و به واسطه آن یک درخت Merkle ساخته می‌شود.

سپس فرد قراردادی را با این قانون ایجاد می‌کند که در هر N بلوک، قرارداد یک شاخص تصادفی را در درخت Merkle انتخاب کند (استفاده از هش بلوک قبلی که از کد قرارداد قابل دسترسی باشد، به عنوان منبع تصادفی بودن) و مقدار X اتر را به اولین موجودیت بدهد تا تراکنش با یک پرداخت تسهیل شده تامین گردد که این پرداخت مدرک شبه تایید شده از مالکیت بلوک در شاخص خاصی در درخت می‌باشد.

وقتی کاربری می‌خواهد که فایل آنها را دوباره دانلود کند، آنها می‌توانند از یک پروتکل کانال ریزپرداخت استفاده نمایند تا فایل را بهبود بخشند (مانند پرداخت 1 szabo برای هر 32 کیلوبایت). موثر ترین رویکرد از لحاظ هزینه این است که پرداخت کننده تراکنش را تا انتها پخش نکند بلکه به جای آن تراکنش را با یک تراکنش نسبتا سودآور تر که بعد از هر 32 بایت دارای همان نانس می‌باشد، جایگزین کند.

از ویژگی مهم این پروتکل این که اگرچه ممکن است به نظر آید فرد در حال اعتماد به گره های تصادفی زیادی است تا تصمیم نگیرند که فایل را به فراموشی بسپارند، اما فرد می‌تواند با شکافتن فایل به قطعات زیادی از طریق تسهیم سری و نظارت بر قرارداد ها ببیند که آیا هر قطعه هنوز در مالکیت گره خاصی است یا نه و با این کار ریسک را تا نزدیک صفر کم کند. اگر یک قرارداد هنوز هم پول پرداخت می‌کند، این امر یک مدرک رمزنگاری شده را فراهم می‌کند، دلیلی بر اینکه فردی هنوز آنجا در حال ذخیره فایل است.

سازمان های خودمختار غیر متمرکز

مفهوم عمومی یک سازمان غیر متمرکز، موجودیتی مجازی است که مجموعه ی معینی از اعضا یا سهامداران دارد که شاید با 67 درصد اکثریت، حق این را داشته باشد که پول های این موجودیت را خرج کند و کد آن را اصلاح کند. اعضا به طور جمعی تصمیم می‌گیرند بر این که چگونه سازمان باید پول هایش را تخصیص دهد.

روش های تخصیص وجوه یک DAO می‌تواند دامنه ای از بخشش ها، حقوق ها تا حتی مکانیزم های عجیب تری مانند یک واحد پول درونی برای پاداش دادن به کارها، داشته باشد. این امر در اساس همان تله گذاری قانونی یک شرکت سنتی یا یک سازمان غیر انتفاعی را تکرار می‌کند اما تنها از فناوری بلاکچین رمزنگاری شده برای اجرا استفاده می‌کند.

تا به حال اکثر گفتگو ها در زمینه DAO حول و حوش مدل سرمایه گرایی یک سازمان خودمختار غیر متمرکز DAC بوده است که دارای سهامداران دریافت کننده سود و یا سهام قابل معامله بوده اند. گزینه دیگری که شاید بتوان آن را به عنوان جامعه خودمختار غیر متمرکز توصیف کرد، همه اعضا در تصمیم گیری سهمی مساوی دارند و لازم است که 67 درصد از اعضای موجود با اضافه کردن و یا حذف یک عضو موافقت کنند.

این ضرورت که هر فرد تنها یک عضویت دارد، ممکن است لازم باشد که به طور جمعی توسط گروه اعمال شود. یک طرح کلی برای بیان چگونگی کد دادن به DO در ادامه شرح داده شده است. ساده ترین طرح در واقع بخشی از یک کد خود اصلاح گر است که تغییر می‌یابد به شرطی که دو سوم اعضا با تغییر موافق باشند.

اگرچه این کد از لحاظ تئوری غیر قابل تغییر است اما فرد در واقع می‌تواند به این مقصود دست یابد و با داشتن بخش هایی از کد در قرارداد های جداگانه یک تغییر پذیری واقعی ایجاد کند و آدرس قراردادی را که در انبارش قابل اصلاح ذخیره شده، داشته باشد.

در یک پیاده سازی ساده قرارداد DAO، سه نوع تراکنش موجود خواهد بود که توسط داده های فراهم شده در تراکنش از هم مجزا می‌شوند:

[0,i,k,v] برای ثبت پیشنهادی با شاخص i تا آدرس را در شاخص k انبارش به ارزش v تغییر دهد.

[0,i] برای ثبت یک رای در طرفداری از پیشنهاد i.

[2,i] برای نهایی ساختن پیشنهاد i به شرطی که آرای کافی جمع آوری شده باشد.

سپس قرارداد می‌تواند برای هر یک از این موارد شروطی داشته باشد. یک سابقه از همه تغییرات انبارش باز به همراه لیستی از افرادی که به آنها رای داده اند، حفظ می‌شود. لیستی از همه اعضا موجود خواهد بود. وقتی هر تغییر انبارش به دو سوم اعضایی که برای آن رای داده اند، برسد، یک تراکنش نهایی تغییر را اجرا می‌کند.

یک ساختار پیچیده تر توانایی رای دادن در داخل آن تعبیه شده و ویژگی هایی مانند ارسال تراکنش، افزودن و حذف اعضا را دارد و حتی ممکن است برای یک نیابت رای گیری به شیوه دموکراسی روان فراهم شود (یعنی هر کس می‌تواند فردی را برای رای دادن تعیین کند و این تعیین انتقالی است، بنابراین اگر A شخص B را تعیین کند و B نیز شخص C را تعیین کند، پس C رای A را تعیین می‌کند.)

این طراحی به DO اجازه می‌دهد که به طور طبیعی به عنوان یک جامعه غیر متمرکز رشد کند و به افراد اختیار می‌دهد که در نهایت وظیفه فیلتر این که چه فردی عضو کارشناسان است یا نه را نیابت کنند. اگرچه بر خلاف سیستم رایج، کارشناسان در طول زمان به این حیطه وارد می‌شوند و از آن خارج می‌گردند زیرا اعضای هر جامعه صف بندی خود را تغییر می‌دهند.

یک مدل جایگزین دیگر برای یک سازمان غیر متمرکز این است که هر اکانت می‌تواند هیچ یا تعدادی سهم داشته باشد و دو سوم سهام برای تصمیم گیری لازم است. یک ساختار کامل شامل عملکرد مدیریت دارایی است که این عملکرد توانایی پیشنهاد دادن برای خرید یا فروش سهام می‌باشد و همچنین توانایی پذیرفتن پیشنهاد ها (ترجیحا با یک مکانیزم مطابق با سفارش در داخل قرارداد).

فرآیند نیابت همچنین سبب انجام سبک دموکراسی روان می‌شود و مفهوم هیات مدیران را عمومی کرده و تعمیم می‌دهد. در آینده مکانیزم های پیشرفته تری برای مدیریت سازمانی ممکن است پیاده شود. در این نقطه است که یک سازمان غیر متمرکز (DO) می‌تواند به عنوان یک سازمان خودمختار غیر متمرکز توصیف شود. (DAO)

تفاوت بین این دو سازمان مبهم است، اما خط تقسیم عمومی این است که آیا مدیریت عموما از طریق یک فرآیند شبه سیاسی انجام می‌شود یا یک فرآیند خودکار. یک امتحان شهودی خوب برای این امر، معیار "عدم وجود زبان رایج" است. آیا یک سازمان می‌تواند هنوز عملکرد داشته باشد در حالی که دو عضو آن با یک زبان صحبت نکنند؟

واضح است که در این حالت یک سازمان سنتی به سبک سهامداری با شکست مواجه می‌شود، اما چیزی مانند پروتکل بیت کوین احتمال بیشتری دارد که موفق شود. مدیریت Robin Hanson، مکانیزمی برای مدیریت سازمانی

از طریق بازار های پیش بینی بوده و مثال خوبی است برای آنچه مدیریت خودمختار صحیح ممکن است به نظر برسد.

توجه کنید که نباید تصور کرد که همه DAO ها برتر از DO ها هستند، اتوماسیون در واقع الگویی است که احتمالا منفعت های عظیمی در جاهایی خاص دارد و در جاهای دیگر هم ممکن است عملی نباشد، و بسیاری سازمان های شبه DAO نیز موجود هستند.