بسمه تعالی

سیستم‌های توزیع‌شده — نیمسال دوم ۹۷-۹۸
گزارش پروژه تحقیقاتی نظری — شماره دانشجویی: ۹۵۱۰۵۶۹۲، ۹۳۱۰۲۱۸۶

مروری اجمالی بر الگوریتم‌های اجماع بر پایه‌ی **Proof-of-Stake** در دفترکل‌های توزیع‌شده [۱]

۱ مقدمه

با معرفی پروتکل بیتکوین در سال ۲۰۰۸ و پیاده‌سازی موفق شبکه‌ی آن در سال‌های بعد، توجه زیادی به کاربردهای فناوری زیرساخت آن، یعنی بلاکچین [۲] شد. بلاکچین نوع خاصی از مفهوم کلی‌تر دفترکل توزیع‌شده است. یک دفترکل توزیع‌شده یک پایگاه‌داده‌ی توزیع شده است که گره‌های موجود در شبکه باید روی آخرین وضعیت [۳] آن به اجماع برسند. این پایگاه‌داده بسته به پروتکل می‌تواند داده‌ساختارهای متفاوتی داشته باشد.

به طور خاص در پروتکل بیت‌کوین این داده‌ساختار زنجیره‌ای از بلاک‌های به هم مرتبط است. تراکنش‌های جدید در قالب یک بلاک جدید در این پایگاه‌داده ذخیره شده و این بلاک به زنجیره اضافه می‌شود. هر بلاک حاوی hash قبلی خود در زنجیره است و اینگونه بلاک‌ها به هم زنجیر می‌شوند. ثبت هر بلاک در زنجیره به معنای اجماع اکثریت شبکه روی آن بلاک است. هر بلاک جدید حاصل فرآیندی است که به صورت توزیع شده و غیرمترکز بین گره‌های شبکه انجام می‌شود. بلاک‌های تولیدشده توسط گره‌های تولیدکننده‌ی آنها در شبکه - که ساختاری همتابه‌همتا دارد - منتشر شده و گره‌های دیگر یکی از بلاک‌های پیشنهادشده را به عنوان یک بلاک در زنجیره‌ی بلاک‌های خود ثبت می‌کنند. یک گره به طور معمول اولین بلاک معتبری که به دست او رسیده‌است را به زنجیره‌ی خود اضافه می‌کند، اما در مواقعی ممکن است دو بلاک به صورت همزمان به دست یک گره برسد. در این جا اصطلاحا یک انشعاب [۴] در زنجیره ایجاد می‌شود. در این مواقع یک گره همه‌ی انشعاب‌ها را ذخیره می‌کند تا اینکه در نهایت با استفاده از قانون انتخاب انشعاب [۵] یکی را به عنوان زنجیره‌ی اصلی انتخاب کند. این قانون در پروتکل بیت‌کوین قانون بلندترین زنجیره [۶] است. احتمال اینکه دو بخش عمده‌ی شبکه روی دو زنجیره‌ی متفاوت اما با طول یکسان بلاک تولید کنند با افزایش طول زنجیره‌ها به صورت نمایی کاهش می‌یابد [۱].

الگوریتمی که در بیت‌کوین برای پیشنهاد بلاک استفاده می‌شود Proof-Of-Work نام دارد. این الگوریتم ویژگی‌های خوبی دارد که شبکه را نهایتا به شکلی احتمالاتی به اجماع می‌رساند [۱] . اولین ویژگی جلوگیری از Sybil Attack است و دومین آن انتخاب رهبر برای پیشنهاد بلاک بعدی است. معیار انتخاب شدن حل یک مساله‌ی ریاضی است که راهی به جز Brute Force ندارد، پس گره‌ها بر اساس توان پردازشی‌ای که دارند می‌توانند به عنوان رهبر انتخاب شوند[۲] .

الگوریتم Proo-Of-Work معایبی نیز دارد که عبارت هستند از:

- مقدار زیادی انرژی هدر می‌رود.
- در برابر ASIC [۷] ها آسیب پذیر است و سیستم را به سمت متمرکز شدن می‌برد.
- Finality ندارد. [۳]

در زمان نگارش این متن بیت‌کوین تقریبا معادل کشور جمهوری چک انرژی مصرف می‌کند. با توجه به اینکه این انرژی صرف حل مساله‌ای می‌شود که هیچ آورده‌ای ندارد قابل قبول نیست[۴] . فناوری ASIC و تمرکزگرایی ماینرها باعث می‌شود هزینه‌ی بازگردانی زنجیره [۸] به مراتب کاهش یابد و امنیت سیستم به خطر افتد. داشتن Finality تضمین می‌کند که زنجیره هیچگاه بازگردانی نخواهد شد اما Proof-Of-Work این ویژگی را ندارد و این مشکل را به شکل احتمالاتی حل می‌کند، به این معنا که تضمین قطعی برای اینکه بلاک‌های قبلی بدون تغییر باقی بمانند وجود ندارد بلکه تنها احتمال تغییر بلاک با افزایش فرزندان آن کاهش می‌یابد.

---

[۱] distributed ledger
[۲] blockchain
[۳] state
[۴] fork
[۵] fork choice rule
[۶] longest chain
[۷] Application Specific Integrated Circuit
[۸] chain reversion

در همان سال‌های اول پیدایش بیت‌کوین به این مشکلات پی برده شد و پژوهشگران به دنبال الگوریتم جایگزینی بودند تا مشکلات Proof-Of-Work را نداشته باشد. تلاش‌های زیادی در این حوزه صورت گرفته است که ما به بررسی یکی از این الگوریتم‌ها یعنی Proof-Of-Stake می‌پردازیم.

تفاوت اصلی Proof-Of-Stake با Proof-of-Work در این است که یک گره به جای سرمایه‌گذاری روی قدرت پردازشی برای رقابت بیشتر با سایر گره‌ها، همان میزان سرمایه را به اصلاح گرو می‌گذارد که به آن stake می‌گویند. به این ترتیب گرهی که stake بیشتری دارد با احتمال بیشتری به عنوان رهبر برای ثبت بلاک بعدی انتخاب می‌شود. به همین خاطر نیازی به رقابت محاسباتی و مصرف انرژی زیاد وجود ندارد .

۲ تعریف مساله

مساله، اجماع در شبکه‌ی غیرمتمرکز همتا به همتا است. شبکه‌های غیرمتمرکز همتا به همتا در گذشته نیز وجود داشته‌اند مانند BitTorrent اما اجماع در چنین شبکه‌ای که فایل‌ها به اشتراک گذاشته می‌شوند را به خوبی نمی‌شود تعریف کرد. در شبکه‌ای غیرمتمرکز عمومی مانند بیت‌کوین که داده‌های مبادله‌شده در آن تراکنش‌های مالی است اجماع اهمیت ویژه‌ای می‌یابد و ما می‌خواهیم الگوریتم اجماعی ارائه دهیم که دو ویژگی Safety و Liveness را داشته باشد.

- Safety: امکان تصمیم‌گیری ناسازگار برای گره‌های شبکه وجود نداشته باشد.
- Liveness: گره‌ها نهایتا تصمیم‌گیری کنند. [۵]

مقاله‌ی Casper Correct By Construnction مدلسازی دقیقی از این ارائه می‌دهد که به شرح زیر است.

۱.۲ تعاریف

فرستندگان پیام در شبکه مجموعه‌ی گره‌های اجماع را تشکیل می‌دهند که به آن‌ها مجموعه‌ی تاییدکنندگان می‌گوییم و با نماد $\mathcal{V}$ نمایش می‌دهیم. بدون کاستن از فرض مساله و همچنین آماده‌سازی برای Proof-Of-Stake وزن هر گره تایید کننده با تابع زیر تعریف می‌شود.

$$\mathcal{W} : \mathcal{V} \to \mathbb{R}_+$$

t ، بازه‌ی گره‌های بیزانتین:

$$0 \leq t < \sum_{v \in \mathcal{V}} \mathcal{W}(v)$$

$\mathcal{C}$ ، مقادیر اجماع یک مجوعه‌ی چند عضوی است که برای اجماع دودویی $\mathcal{C} = \{0, 1\}$ و برای اجماع در بلاکچین مجموعه‌ی همه‌ی بلاکچین‌ها است.

$\mathcal{E}$ ، تابع تخمین:

$$\mathcal{E} : \Sigma \to \mathcal{P}(\mathcal{C}) \setminus \{\emptyset\}$$

$\mathcal{P}$ مجموعه‌ی توانی است. تابع تخمین یک مجوعه‌ی چند عضوی برمی‌گرداند و گره‌های تاییدکننده بین مقادیر اجماع انتخاب خواهند کرد.

۲.۲ تعاریف پروتکل

- Σ: مجموعه‌ای از پیام‌ها
- M: مجوعه‌ای از سه تایی‌های (consensus values, validator name, protocol state)

$$\Sigma \subset \mathcal{P}_{finite}(M) \quad (۱)$$
$$M \subset \mathcal{C} \times \mathcal{V} \times \Sigma \quad (۲)$$

تعاریف Estimate, Sender, Justification:

$$Estimate : M \to \mathcal{C} \tag{۳}$$
$$Sender : M \to \mathcal{V} \tag{۴}$$
$$Justification : M \to \Sigma \tag{۵}$$
$$Estimate((c, v, \sigma)) := c \tag{۶}$$
$$Sender((c, v, \sigma)) := v \tag{۷}$$
$$Justification((c, v, \sigma)) := \sigma \tag{۸}$$

تعریف انتقال حالت پروتکل $\rightarrow$:

$$\rightarrow : \Sigma \times \Sigma \to \{True, False\} \tag{۹}$$
$$\sigma_1 \to \sigma_2 :\Leftrightarrow \ \sigma_1 \subseteq \sigma_2 \tag{۱۰}$$

رفتار بیزانتین در شبکه را فرستادن پیام‌های متناقض تعریف می‌کنیم زیرا همین رفتار نیز برای نرسیدن به اجماع کافی است.
تعریف پیام متناقض:

$$\cdot \perp \cdot : M \times M \to \{True, False\} \tag{۱۱}$$
$$m_1 \perp m_2 :\Leftrightarrow \ Sender(m_1) = Sender(m_2) \wedge m_1 \neq m_2 \tag{۱۲}$$
$$\wedge\, m_1 \notin Justification(m_2) \wedge m_2 \notin Justification(m1) \tag{۱۳}$$

تعریف تایید کننده‌ی بیزانتین:

$$E : \Sigma \to \mathcal{P}(\mathcal{V}) \tag{۱۴}$$
$$E(\sigma) := \{v \in \mathcal{V} : \exists m_1 \in \sigma, \exists m_2 \in \sigma,\ m_1 \perp m_2 \ \wedge \ Sender(m_1) = v\} \tag{۱۵}$$

وزن گره‌های بیزانتین:

$$F : \Sigma \to \mathbb{R}_+ \tag{۱۶}$$
$$F(\sigma) := \sum_{v \in E(\sigma)} \mathcal{W}(v) \tag{۱۷}$$

تابع F یکنواخت است.

$$\sigma_1 \subseteq \sigma_2 \implies F(\sigma_1) \leq F(\sigma_2)$$

پروتکل آستانه‌ی تحمل t وزن بیزانتین را دارد.

$$\Sigma_t = \{\sigma \in \Sigma : F(\sigma) \leq t\}$$

حال Safety را تعریف می‌کنیم.

۳.۲ تعریف Safety

Casper CBC چارچوبی ارائه می‌دهد که می‌توان با آن Safety الگوریتم‌ها را بررسی کرد. اثبات این ویژگی برای پروتکل‌ها طولانی بوده و برای جزئیات بیشتر می‌توان به خود مقاله‌ی Casper CBC [۵] مراجعه کرد. به صورت کلی هر الگورتیمی که با استفاده از چارچوب ذکر شده بررسی شود و ویژگی زیر را داشته باشد Safe است.

$$F(\bigcup_{i=1}^{n}\sigma_i) \leq t \implies Consistent_C(\bigcup_{i=1}^{n} Decisions_{C,t}(\sigma_i))$$

۳ مرور و مقایسه کارهای پیشین

در این بخش ابتدا یک طرح کلی از نحوه‌ی کارکرد یک الگوریتم PoS ارائه کرده و تهدیدهای جدیدی که الگوریتم‌های PoS به وجود می‌آوردند را بیان می‌کنیم. سپس دو پروتکل شناخته‌شده را، که هر کدام شامل خانواده‌ای از الگوریتم‌هاست، بررسی و مقایسه می‌کنیم.

۱.۳ یک طرح ابتدایی از PoS

گفتیم که در یک الگوریتم PoS احتمال تولید بلاک بعدی توسط یک گره با میزان سرمایه یا اصطلاحا stake ی که دارد متناسب است. به این ترتیب می‌توان تصور کرد که چنین الگوریتمی جایگزین بسیار بهتری برای PoW از نظر میزان مصرف انرژی برق است.
یک نمونه‌ی کلی از یک الگوریتم PoS-based می‌تواند به شکل زیر باشد:

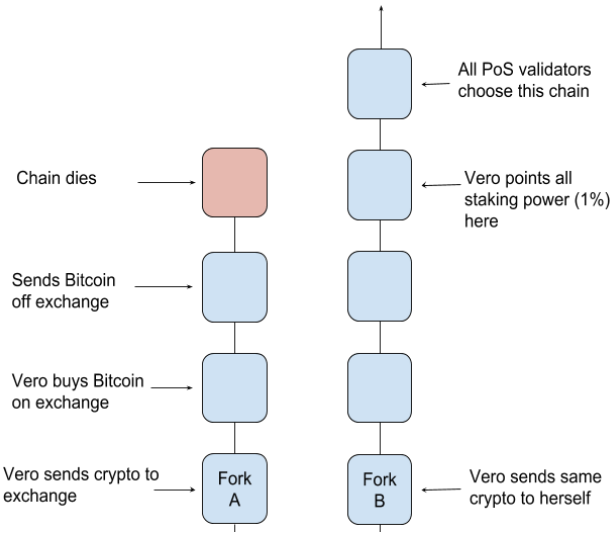
- هر گرهی با هر میزان سرمایه‌ی ناصفری می‌تواند در فرآیند تولید بلوک شرکت کند.
- گره‌های شرکت‌کننده مجبور نیستند سرمایه‌ی خود را به صورت یک security deposit گرو بگذارند.
- جریمه‌ای برای خرابکاری در شبکه وجود ندارد.

ویژگی‌های بالا یک الگوریتم ساده را توصیف می‌کنند. در مورد دو ویژگی آخر می‌توان توجیه کرد که هر چه یک گره سرمایه‌ی بیشتری در شبکه داشته باشد انگیزه‌اش برای خرابکاری کمتر می‌شود، زیرا ارزش سرمایه‌ی او رابطه‌ی مستقیمی با اعتبار و امنیت شبکه دارد. اما اگر انگیزه‌های افراد برای شرکت در شبکه را با دقت بیشتری بررسی کنیم متوجه خطرات احتمالی این الگوریتم ساده می‌شویم.

۲.۳ حملات احتمالی

هدف اصلی یک الگوریتم PoS کاهش مصرف انرژی است. کاهش مصرف انرژی با کاهش هزینه‌ی تولید یک بلاک تحقق می‌یابد اما همین کاهش هزینه، چالش‌های امنیتی جدیدی را معرفی می‌کند [۶].

۱.۲.۳ حمله‌ی Nothing at Stake



شکل ۱: یک سناریو نمونه برای حمله‌ی nothing at stake، منبع [۶]

فرض کنید علاوه بر ویژگی‌ها الگوریتم ساده‌ی توضیح‌داده شده در قسمت قبل، هنگامی که یک انشعاب رخ می‌دهد، زنجیره‌ای انتخاب خواهد شد که بیشترین رای را از نظر مجموع وزن stake ها بیاورد.

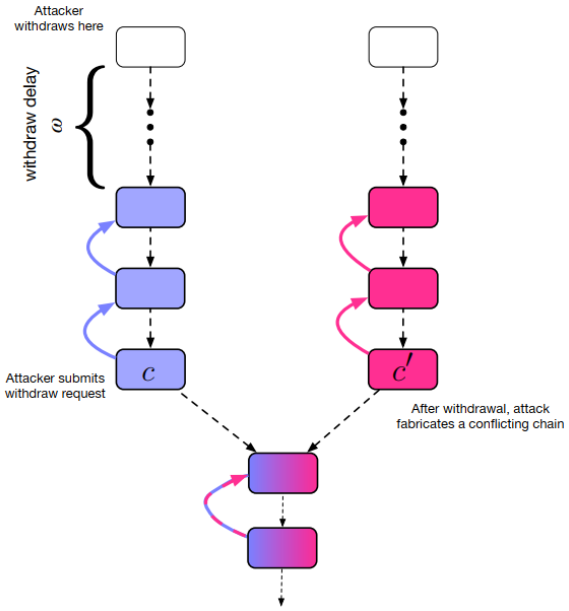
به دلیل راحت و سریع بودن تولید یک بلوک، یک گره بدون هیچ مشکلی می‌تواند تعداد زیادی بلوک را در زمان محدودی تولید کند. از این رو تمام گره‌ها این انگیزه را دارند که روی تمام انشعاب‌ها بلوک درست کنند. به این ترتیب در نهایت، هنگامی که یک انشعاب انتخاب شود، می‌توانند از جایزه بلاک‌ها یا کارمزد تراکنش‌هایی که در آن زنجیره ثبت کرده‌اند بهره‌مند شوند.

فرض ما این است که گره‌ها همواره بر اساس انگیزه‌های شخصی خود عمل می‌کنند. در این شرایط یک گره adversary با داشتن حتی ۱ درصد از مجموع stake ها می‌تواند به شبکه حمله کند. یک نمونه از این حمله که در شکل ۱ نشان داده شده است می‌تواند اینگونه باشد:

Vero در انشعاب A برای یک صرافی پول می‌فرستد و آن را تبدیل به بیتکوین می‌کند. با توجه به اینکه همه رای خود را روی همه‌ی انشعاب‌ها پخش می‌کنند. کافی‌است پس از مطمئن شدن از خارج کردن بیتکوین‌ها از صرافی تمام ۱ درصد stake خود را روی انشعاب B رای می‌دهد که در آن پول را به جای فرستادن برای صرافی برای حساب دیگری از خود فرستاده است. به این صورت چون قانون انتخاب انشعاب، بیشترین stake است، انشعاب B رای بیشتری خواهد آورد و انشعاب A حذف خواهد شد. به این صورت Vero یک double spending انجام می‌دهد.

در عمل این حمله تا حدی غیرواقعی به نظر می‌رسد و در دنیای واقعی انتظار می‌رود که حداقل بخشی از گره‌ها، نه بر انگیزه‌ی شخصی، بلکه بر اساس پروتکل عمل کنند. در هر صورت از نظر نظری امکان این حمله وجود دارد [۶].

۲.۲.۳ حمله‌ی Long Range



شکل ۲: یک سناریو نمونه برای حمله‌ی long range، منبع [۷]

فرض کنید الگوریتم ساده‌ی قسمت قبل را در راستای سخت‌تر شدن خرابکاری این گونه تغییر دهیم:

هر گره validator برای شرکت در فرآیند رای دادن باید پول خود را به صورت یک security deposit گرو بگذارد. بدون کم شدن از کلیت مسئله فرض کنید این قابلیت در پروتکل به صورت on-chain وجود دارد. در صورت خرابکاری و تشخیص دیگر validator ها به همراه آوردن اثبات برای تشخیص خود، پول گرو گذاشته‌شده توسط گره خرابکار می‌تواند به اصطلاح slash شده و به عنوان جریمه بین بقیه‌ی validator ها پخش شود. همچنین هر validator می‌تواند در هر زمان دلخواه درخواست خارج شدن از فرآیند را بدهد که از زمان درخواست تا آزاد شدن گرو میزانی تاخیر به منظور امنیت بیشتر (افزایش اطمینان از انجام نشدن double spending) وجود دارد.

همچنین فرض کنیم که الگوریتم ما یک الگوریتم BFT-based است و حداکثر کمتر از $1/3$ گره‌ها، خرابکار هستند. به همین دلیل قانون انتخاب انشعاب رای بیشتر از $2/3$ وزن stake هاست.

سناریو حمله که در شکل ۲ نشان داده‌شده است می‌تواند به این صورت باشد:

فرض کنید، مدت‌ها پیش در زمان t ، بیش از $2/3$ کل stake ها در اختیار گروه کوچکی از validator ها بوده باشد. این گروه کوچک به هر دلیلی می‌توانند با یکدیگر تبانی کرده و بخش بزرگی از زنجیره را با ایجاد یک انشعاب در یک بازه‌ی بزرگ (از زمان t تا به حال) تغییر دهند. برای اطمینان از جریمه نشدن، این گروه کوچک می‌تواند قبل از ایجاد انشعاب درخواست خارج کردن پول خود از گرو را داده و صبر کند تا این اتفاق روی زنجیره ثبت شده سپس انشعاب را ایجاد کند [۸].

۳.۳ Casper the Friendly Finality Gadget (FFG)

همانگونه که ذکر شده الگوریتم Proof-Of-Work فاقد ویژگی Finality است. الگوریتم Casper که توسط اتریوم ارائه شده است این ویژگی را به بلاک‌ها اضافه می‌کند. Casper به گونه‌ای طراحی شده است که می‌تواند به عنوان لایه‌ی بالایی یک الگوریتم پیشنهاد بلاک مانند Proof-Of-Work قرار گیرد. Casper دارای ویژگی‌های زیر است:

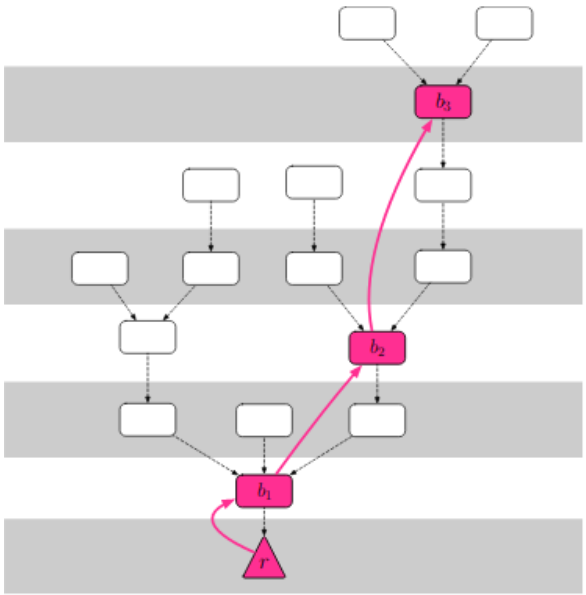
- قابلیت حساب‌داری: اگر تایید کننده‌ای قوانین را نقض کند می‌توان آن را تشخیص داد و جریمه کرد. جریمه کردن بدین شکل است که Stake او را بین بقیه‌ی اعضای مجموعه‌ی تایید کنندگان پخش می‌کنیم.
- مجموعه‌ی تایید کنندگان پویا: مجموعه‌ی گره‌های تایید کننده نیاز است که تغییر کند، تایید کننده‌های جدید باید بتوانند وارد شوند و تایید کننده‌های فعلی باید بتوانند خارج شوند.
- مقاوم در برابر Long Range Attacks
- قابلیت اضافه شدن به بلاکچین‌های Proof-Of-Work موجود با یک به‌روز رسانی نرم‌افزاری

۱.۳.۳ تعاریف

- Checkpoint: بلاک‌هایی با ارتفاع مضرب ۱۰۰
- رای: اجزای تشکیل‌دهنده‌ی رای به شرح زیر است:

  - s: the hash of any justified checkpoint (the ”source”)
  - t: any checkpoint hash is a descendent of s (the ”target”)
  - h(s): the height of checkpint s in the checkpoint tree
  - h(t): the height of checkpint t in the checkpoint tree
  - $\mathcal{S}$ signature of <s, t, h(s), h(t)> from the validator's private key

گره‌های تایید کننده به Checkpoint ها رای می‌دهند. هر کدام که بتواند رای دو سوم وزن stake ها را کسب کند به عنوان بلاک Justified مشخص می‌شود. checkpoint ی Finalize می‌شود اگر بلاک genesis باشد یا Justified باشد و لینکی از آن به checkpoint ی با ارتفاع یکی بیشتر از آن وجود داشته باشد.



شکل ۳: لینک‌هایی که زنجیره‌ی اصلی را مشخص می‌کنند، منبع [۷]

۲.۳.۳ قوانین لازم برای تامین **Safety**

هیچ گره تایید کننده‌ای نمی‌تواند دو رای متمایز با شرایط زیر دهد.

$$< v, s_1, t_1, h(s_1), h(t_1) > and < v, s_2, t_2, h(s_2), h(t_2) >$$

۱.

$$h(t_1) = h(t_2)$$

۲.

$$h(s_1) < h(s_2) < h(t_2) < h(t_1)$$

تایید کننده اگر به دو checkpont که در شاخه‌های مختلف هستند رای دهد و اگر به بازه‌ای رای بدهد که قبلا به آن رای داده است. دلیل وجود دومین قانون این است که اگر آخرین checkpoint در شبکه را تایید کننده شنیده پس لزومی ندارد که به قبل‌تر ها رای دهد و این رفتار به عنوان رفتار خرابکارانه شناخته می‌شود اگر هر کدام از قوانین فوق توسط تایید کننده‌ای نقض شود stake نقض‌کننده بین بقیه‌ی تایید کننده‌ها تقسیم خواهد شد.

۳.۳.۳ **Liveness**

Liveness الگوریتم Casper به Liveness الگوریتم زیرین وابسته است. اگر الگوریتم زیرین Liveness داشته باشد چون همیشه بلاک تولید می‌شود، پس می‌توانیم Finalized Checkpoint تولید کنیم.

اثبات : فرض کنید که a باشد Justified Checkpoint با بزرگترین ارتفاع و b باشد Target Checkpoint با بزرگترین ارتفاع برای هر تایید کننده‌ای که رای داده است.

هر Checkpoint ای مانند a' که از فرزندان a با ارتفاع $h(a') = h(b)+1$ است می‌تواند بدون نقض کردن قوانین Justified شود و سپس a' می‌تواند با اضافه کردن یک لینک از a' به فرزند مستقیمش Finalized شود.

۴.۳.۳ جلوگیری از **Long Range Attack**

یک انشعاب حاصل از Long Range Revision را در نظر بگیرید، اکثریت تایید کننده که در گذشته‌ی دور دو سوم وزن رای را داشتند بلاک‌هایی جدید را Finalize می‌کنند. در این مورد کافی است به سادگی آن بلاک‌ها توسط گره‌های دیگر نادیده گرفته شود زیرا که قبلا بلاک Finalize شده با همان ارتفاع دیده‌اند.

اثباتی غیررسمی بر کارکرد این روش در مقاله‌ی Casper آورده شده است.

۴.۳ **Ouroborous**

Ouroborous الگوریتم Proof-Of-Stake ی است که در Cardano مورد استفاده قرار گرفته است. با توجه به اینکه مقاله‌ی این الگوریتم بسیار طولانی است و نسخه‌های مختلفی از آن در طی زمان آمده است و ویژگی‌های مختلفی به آن اضافه کرده است همچون Scalability ما در اینجا صرفا ایده‌ی اصلی مقاله و ایده‌ی اصلی آن برای Safety، Liveness و Long Range Attack خواهیم آورد. برای جزئیات بیشتر با اثبات‌های دقیق ریاضی که حتی شامل تحلیل از منظر نظریه‌ی بازی نیز می‌شود می‌توانید به مقاله‌ی اصلی رجوع کنید.

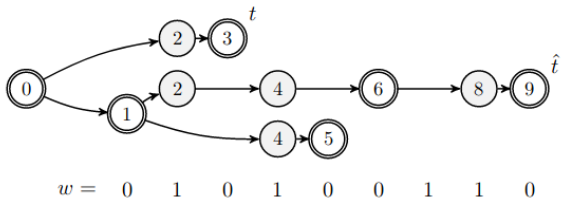
۱.۴.۳ ایده‌ی اصلی

الگوریتم زمان را به epoch های مساوی تقسیم می‌کند و هر epoch را به slot های مساوی تقسیم می‌کند. هر slot رهبر خود را دارد که مسئول ارائه‌ی بلاک است. رهبر هر slot اول هر epoch توسط یک Verifiable Random Function تعیین می‌شود. احتمال رهبر شدن برای یک slot با میزان پولی که stake کرده‌ایم متناسب است.

اجرایی از این الگوریتم را در نظر بگیرید. رشته‌ی ω را مطابق زیر تعریف می‌کنیم:

$$\omega_i = \begin{cases} 0 & \text{اگر slot دست گره درستکار باشد} \\ 1 & \text{اگر slot دست گره خرابکار باشد} \\ \perp & \text{اگر نتوان ادعایی برای slot داشت} \end{cases}$$

شکل که یک نمونه از اجرای الگوریتم است را در نظر بگیرید: در بعضی از slot ها که رهبر گره خرابکار بوده است یا اقدام به تولید چندین بلاک نموده است که باعث شده است fork در شبکه صورت گیرد یا کلا بلاکی ارائه نداده و باعث بوجود آمدن تاخیر در شبکه شده است. هدف ما این است که همه‌ی گره‌های درستکار شبکه به یک تاریخچه همگرا شوند. با انتخاب بلندترین زنجیره می‌توانیم نهایتا شبکه را به اجماع برسانیم. در ادامه به صورت شهودی چرایی درست بودن این انتخاب را بررسی می‌کنیم.



شکل ۴: اجرای الگوریتم Ouroboros [۹]

۲.۴.۳ راستی آزمایی

مسیر t در شکل را در نظر بگیرید. متغییرهای زیر را تعریف می‌کنیم:

- $gap(t)$: اختلاف با عمیق‌ترین بلاک درستکار
- $reverse(t)$: تعداد slot های خرابکار بعد از t
- $reverse(t) - gap(t)$:$reach(t)$

برای مثال برای t در گراف بالا داریم:

$$gap(t) = 4, reserve(t) = 3, reach(t) = -1$$

سپس برای Fork F خواهیم داشت:

$$reach(F) = \max reach(t)$$

$$margin(F) = \text{second best disjoint } reach(t)$$

برای رشته‌ی ω داریم:

$$\rho(\omega) = \max_F reach(F)$$
$$\mu(\omega) = \max_F margin(F)$$

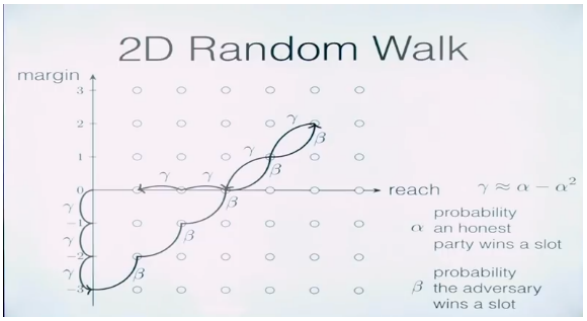
گره‌های خرابکار زمانی پیروز خواهند شد که

$$\rho(\omega) \geq 0$$

با استفاده از دو متغییری که تعریف کرده‌ایم یک Random Walk دو بعدی شکل می‌دهیم و نتیجه را بررسی می‌کنیم.

$$(\rho(\omega 1), \mu(\omega 1)) = (\rho(\omega) + 1, \mu(\omega) + 1)$$

$$(\rho(\omega 0), \mu(\omega 0)) = \begin{cases} (\rho(\omega) - 1, 0) & \rho(\omega) > \mu(\omega) = 0 \\ (0, \mu(\omega) - 1) & \rho(\omega) = 0 \\ (\rho(\omega) - 1, \mu(\omega) - 1) & \text{otherwise} \end{cases}$$



شکل ۵: 2D Random Walk [۹]

همانگونه که مشاهده می‌کنید β به منهای بی‌نهایت میل می‌کند و این یعنی گره‌های درستکار اگر در اکثریت باشند پیروز خواهند شد.

۳.۴.۳ جلوگیری از **Long Range Attack**

Ouroborous در این شرایط زنجیره‌ای را که چگال‌تر است را انتخاب می‌کند. چگال‌تر در اینجا به این معنی است که در slot های بیشتری بلاک درست تولید شده است.

۵.۳ مقایسه

پروتکل Ouroborous در مقایسه با پروتکل Casper دارای مدلسازی و اثبات‌های دقیق ریاضی بر اساس یک مجموعه از فرض‌های شفاف است که جایی برای تردید از نظر تئوری باقی نمی‌گذارد، در حالی که اطلاعات راجع به Casper صورت پراکنده در مجموعه‌ای از مقاله‌هایی به صورت draft، پست‌های وبلاگ یا فروم‌های تیم توسعه دهنده قابل دسترسی است و مقاله‌ای کاملا آکادمیک توسط تیم توسعه دهنده‌ی آن ارائه نشده است. با این حال تیم Ouroboros از نظر سابقه در مقایسه با تیم توسعه‌دهنده‌ی Casper هنوز در ابتدای راه است.

این دو پروتکل را می‌توان با توجه به موارد زیر با یکدیگر مقایسه کرد:

- میزان تحمل خرابکاری: فرض پروتکل Ouroboros این است که اکثر گره‌های شبکه honest هستند، از این رو در برابر حمله‌ی ۵۰ درصد مقاوم است. این میزان برای Casper برابر $1/3$ است.
- نهایی بودن تراکنش (finality): با توجه به وجود checkpoint ها در پروتکل Casper FFG می‌توان گفت که این پروتکل near-instant finality دارد (مثلا با فاصله‌ی هر ۱۰۰ بلوک). در مقابل پروتکل Ouroboros از این نظر شبیه پروتکل بیتکوین یک eventual consensus احتمالاتی دارد (برای مثال اگر فرض کنیم ۱۰ درصد مجموع stake ها در اختیار گره‌های خرابکار است احتمال برگشت خوردن یک تراکنش پس از نیم ساعت چیزی حدود ۱ بر روی ۱۰ تریلیون است).
- فرضیات در مورد شبکه: نسخه‌هایی از پروتکل Ouroboros علاوه بر شبکه‌ی سنکرون در یک شبکه‌ی نیمه سنکرون نیز کار می‌کنند اما در مورد پروتکل Casper در حال حاضر اطلاعات دقیقی در دست نیست.
- مقیاس‌پذیری: نسخه‌ای از پروتکل Casper از مفهومی به نام sharding برای مقیاس پذیری استفاده خواهد کرد. در این روش گره‌های شرکت کننده به دسته‌هایی تقسیم شده و هر کدام دسته‌ای تراکنش‌ها را ثبت می‌کنند (این یک شهود کلی است). به این ترتیب با نوعی موازی سازی این پروتکل مقیاس پذیر خواهد شد. مفهموم مشابهی نیز در نسخه‌ای از پروتکل Ouroboros به نام Ouroboros Hydra پیاده‌سازی خواهد شد.
- safety و liveness: پروتکل Ouroboros دارای هر دو ویژگی است. پروتکل Casper نیز safe است و liveness آن به liveness الگوریتم پیشنهاد بلاک زیرین وابسته است.

از عیب‌های پروتکل Casper نبود پشتوانه‌ی نظری formal است. از طرفی در الگوریتم انتخاب رهبر در پروتکل Ouroboros رهبر slot های بعدی یک epoch مشخص است. از این رو امکان حملات DoS به آن رهبر وجود دارد [۱۰].

۴ پیشنهاد برای کارهای آتی

در صورتی که بتوان الگوریتمی امن مبتنی بر PoS ارائه داد که در هر slot زمانی یک رهبر را بصورت گمنام انتخاب کند و زمانی هویت دیجیتال رهبر افشا شود که بلاک همان slot را پیشنهاد دهد، می‌توان از حملات DoS احتمالی جلوگیری کرد. این مسئله‌ی باز Secret Single-Leader Election (SSLE) نام دارد [۱۱].

مراجع

[1] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. (2008) https://bitcoin.org/bitcoin.pdf

[2] MM. Jahanara, A crash course on Proof-of-Stake (Part I)
`https://medium.com/coinmonks/a-crash-course-on-proof-of-stake-part-i-843e7a44c682`

[3] Proof-of-Stake-FAQ
`https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ`

[4] Bitcoin Energy Consumption Index
`https://digiconomist.net/bitcoin-energy-consumption`

[5] V. Zamfir, N. Rush, A. Asgaonkar, G. Piliouras. (2018)
Intoduction to "Minimal CBC Casper" Family of Consensus Protocols. https://github.com/cbc-casper/cbc-casper-paper/blob/master/cbc-casper-paper-draft.pdf

[6] J. Martinez, Understanding Proof of Stake: The Nothing at Stake Theory https://medium.com/coinmonks/understanding-proof-of-stake-the-nothing-at-stake-theory-1f0d71bc027

[7] V. Buterin, V. Griffith. (2019) Casper the Friendly Finality Gadget, https://arxiv.org/pdf/1710.09437.pdf

[8] A. Sharma, Understanding Proof of Stake through it's Flaws https://medium.com/@abhisharm/understanding-proof-of-stake-through-its-flaws-part-3-long-range-attacks-672a3d413501

[9] A. Kiayias, A. Russell, B. David, R. Oliynykov. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. Cryptology ePrint Archive, Report 2016/889. https://eprint.iacr.org/2016/889

[10] A. Kiayias, How does Casper compare to Ouroboros? https://iohk.io/blog/how-does-casper-compare-to-ouroboros/

[11] Protocol Labs, Secret Single-Leader Election (SSLE), research RPF https://github.com/protocol/research-RFPs/blob/master/RFPs/rfp-6-SSLE.md