**بسم الله الرحمن الرحیم**

# موضوع : مکانیزمی برای کش کردن درخواست ها در الگوریتم Blockene


**چکیده**

بلاک چین ، سیستمی برای ثبت و ضبط داده‌ها است. این داده‌ها می‌توانند برای نمونه تراکنش‌های بانکی باشند یا اسناد مالکیت، قرارها، پیام‌های شخصی یا دیگر اطلاعات. ویژگی بلاک چین این است که کار ذخیرهٔ این داده‌ها بدون وجود یک مدیر و صاحب‌اختیار مرکزی ، امکان‌پذیر میکند و نمی‌توان با خراب کردن یک نقطهٔ مرکزی داده‌های ذخیره‌شده را تحریف یا نابود کرد. در گوشی هوشمند به دلیل کمبود منابع استفاده از این فناوری چالش بزرگی را ایجاد کرده به تازگی الگوریتمی به نام (بلاک کنه) Blockene معرفی شده که استفاده از منابع را در گروه های عضو کاهش میدهد این روش با داشتن سربار کم قادر است توانایی بالایی از معاملات را برای تعداد زیادی از شرکت کنندگان فراهم آورد ولی یکی از چالش های این روش در دست نبودن گوشی موبایل است یعنی در زمان خاصی که موقع ارسال اطلاعات به سرور است و یا موقع ارسال تراکنش ، گوشی به هر دلیل (مثل نبودن اینترنت ) نتواند اطلاعات را ارسال کند ما در این مقاله مکانیزمی را برای کش کردن درخواست ها ارائه می دهیم که کارایی این الگوریتم را به مقدار زیادی بالا می برد در نهایت ما این روش را با شبیه سازی اثبات میکنیم.


1. مقدمه

بلاک‌چین یا زنجیره بلوکی، پایگاه داده‌ای متشکل از فهرست سوابق تراکنش‌هایی است که همواره رشد می‌کند و بر تعداد آن افزوده می‌شود. به این سوابق بلاک (Block) می‌گویند که از طریق رمزنگاری (Cryptography) به یکدیگر متصل می‌شوند. بلاک‌ها در زنجیره‌ای از نودهایی (Node) با شبکه‌ی همتابه‌همتا قرار می‌گیرند و به مخزن ذخیره‌سازی که تشکیل می‌شود، دفتر

کل دیجیتال (Digital Ledger) می‌گویند. به عبارتی دیگر، بلاک چین به‌عنوان نوعی دفتر کلی توزیع‌شده (Distributed Ledger) شناخته می‌شود که تاریخچه‌ی دارایی دیجیتال را با استفاده از غیر متمرکز سازی و هش رمزنگاری (Cryptographic Hash) غیر قابل تغییر و شفاف می‌کند.
همه بلاک‌چین ها امروز به گره های عضو نیاز دارند تا سرورهای قدرتمندی را با منابع قابل توجه شبکه ، ذخیره سازی و محاسبه اجرا کنند.
بنابراین Blockchains امروز محدود به استفاده از موارد است که در آن اعضا انگیزه قوی برای شرکت داشته باشد و از این رو بتوانند هزینه های منابع بالا را تامین کنند.

Blockchains مستلزم آن است که اکثریت (معمولا دو سوم)از اعضا صادق باشن این ویژگی که تضمین آن ، در هنگام شرکت تعداد زیادی از اعضا آسان تر است.

ما در این مقاله به بهبود عملکردی الگوریتم Blockene میپردازیم این الگوریتم با سبک بودن و مقیاس پذیری ، پذیرش گسترده توسط میلیون ها کاربر را امکان پذیر می کند
موفقیت کلیدی در Blockene این است که به جای آنکه اعضا برای اجرای سرورهای قدرتمند درخواست بدهند آن ها را قادر می سازد به عنوان شهروندان درجه یک در اجماع و اجرا شرکت کنند ، حتی در حالی که از دستگاه های سبک تر از تلفن های هوشمند استفاده می کنید و هزینه ها را به ترتیب بزرگتر کاهش می دهد.

مشکلی که در این الگوریتم است در موضوع ارسال درخواست ها به سرور است که به هر دلیلی (نبود اینترنت) ارسال درخواست با مشکل مواجه شود.

ما یک مکانیزمی را معرفی میکنیم که به کش کردن درخواست ها منجر می شود و در نهایت اکثر درخواست ها به سرور میرسد

2. آثار مرتبط

اولین مورد بلاک چین کنسرسیوم است (به عنوان مثال HyperLedger ) که با محدود کردن تعداد بلاک چین ها به تعداد کمی از گره ها ، می توانند الگوریتم های اجماع بیزانس سنتی (ویژگی است که امکان تحمل مجموعه‌ای از خطاهای خاص را می‌دهد) را به جای اجماع مبتنی بر اثبات کار محاسبه کند.

معماری دوم بلاک چین های اثبات سهام است که قدرت رای گیری گره عضو را با مقدار پولی که گره عضو در بلاکچین دارد گره می زند.

در حالی که دو معماری فوق ، یعنی بلاک چین های کنسرسیوم و بلاک چین های اثبات سهام ، هزینه محاسبه خام گره های عضو را برآورده می کنند ، اما هنوز برای تلفن های هوشمند بسیار گران هستند.

به طور خاص ، آنها برای شبکه و منابع ذخیره سازی سنگین هستند ، زیرا آنها نیاز دارند که گره های عضو همیشه با وضعیت "فعلی" بلاکچین به روز باشند

بعلاوه ، چنین بلوک چین ، فضای ذخیره سازی ترابایت را در گره های عضو مصرف می کند ، زیرا هر گره عضو یک نسخه محلی از بلاک چین را ذخیره می کند.

حتی بلاک چین هایی که تلفن های هوشمند را هدف قرار می دهند همان فلسفه به روز بودن گره های عضو را دارند و بنابراین هزینه های شبکه و فضای ذخیره سازی را متحمل می شوند.

برخی از بلاک چین ها هزینه ذخیره سازی را ، با خرد کردن(Sharding) آدرس می دهند

OmniLedger یک بلاک چین است که اخیرا معرفی شده که به شرکت کنندگان اجازه می دهد فقط یک تکه (shard) از بلاک چین را ذخیره کند این روش نوع متفاوتی از Byzcoin برای رسیدن به اجماع سریع است.

Rapid Chain همچنین از Sharding برای کاهش هزینه ذخیره سازی استفاده می کند

هر دوی این کارها فقط برای چند هزار شرکت‌کننده در نظر گرفته می‌شوند و همچنین نیاز دارند که شرکت کنندگان یک بخش بزرگ (۱۳ یا ۱۶) کل blockchain را ذخیره کنند. گره های سبک اما ناتوان !

در مقابل ، Blockene ، به استفاده از منابع سبک برای اعضای کلاس اول که در اجماع و اعتبار سنجی شرکت می کنند ، دست می یابد.

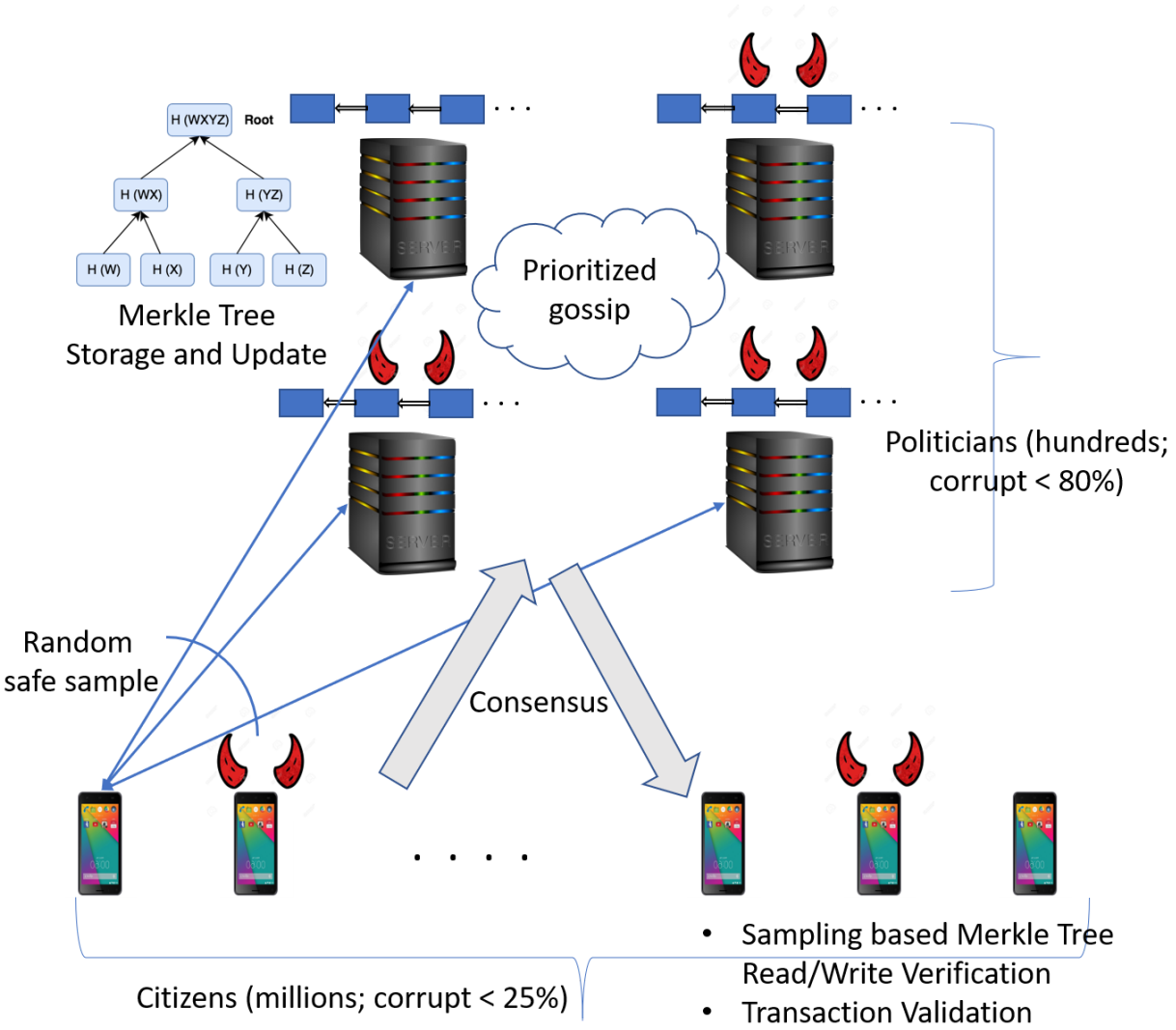
بعلاوه ، برخلاف Ethereum که به اکثریت صادقانه گره های سنگین مرتبط است (فقط گره های سنگین می توانند رای دهند) ،Blockene تا 80٪ از گره های "سنگین" (به عنوان مثال ، سیاستمداران) را خراب کار می داند.

اعضا در Blockene فقط به یک تلفن هوشمند و انتقال داده ناچیز (کمتر از 60 مگابایت در روز) و محاسبه ناچیز نیاز دارند.

این کار با قادر ساختن گره‌های عضو برای کار با حداقل وضعیت مورد نیاز برای انجام یک بلوک خاص، و انجام کار تنها چند بار در روز انجام شود یعنی با تلاش برای همیشه به روز بودن ، به این هدف دست می یابد.

3. طرح بندی مسئله و روش ارائه شده

Blockene از یک معماری جدید و دو لایه با اعتماد نامتقارن استفاده می کند. این معماری در شکل 1 نشان داده شده است.




شکل 1 : معماری الگوریتم Blackene

دو نوع گره در Blackene وجود دارد: شهروندان(کاربران) و سیاستمداران (سرور ها).

شهروندان دارای منابع محدود هستند (به عنوان مثال، بر روی گوشی های هوشمند اجرا می شوند)، تعداد زیادی (میلیون ها) نفر هستند و تنها نهادهای دارای قدرت رای گیری در سیستم هستند (مشارکت در اجماع)

سیاستمداران ، قدرتمند هستند و تعدادشان  کم است (صدها نفر)، با این حال آنها قدرت رای گیری ندارند سیاستمداران تنها تصمیماتی را که توسط شهروندان انجام می دهند ، را اجرا می کنند و نمی توانند به تنهایی تصمیم بگیرند.

از آنجا که شهروندان در اجماع شرکت می کنند ، حداقل دو سوم شهروندان باید صادق باشند ، در حالی که دیگران می توانند بدخواهانه و تبانی کنند و Blockene اجازه ایجاد فساد گسترده در سطح میلیون ها شهروند را نمیدهد و یا سخت میکند با این حال، سیاستمداران از اعتماد بسیار پایین برخوردار هستند.

Blockene فقط به 20٪ از سیاستمداران نیاز دارد تا صادق باشند ، 80٪ باقیمانده سیاستمداران می توانند بدخواه باشند و با یک سوم شهروندان مخرب تبانی کنند .

با این حال مشکل در دست نبودن گره شهروند هم وجود دارد که ممکن است زمانی پیش بیاید که اینترنت و دیتای گوشی تلفن همراه در دست نباشد در این قسمت ما مکانیزمی را برای کش کردن ارائه میدهیم که برای هر درخواست چک میکنیم که اگر به اینترنت متصل باشد عمل ارسال دیتا انجام شود ولی اگر به هر دلیلی از جمله نبودن اینترنت درخواست ارسال نشد تمام درخواست ها در لیستی در همان شهروند (موبایل) ذخیره می شوند و در زمانی که نوبت به این شهروند رسید اگر به اینترنت متصل بود علاوه بر

ارسال درخواست فعلی به درخواست های قبلی هم رجوع می کند و آن درخواست ها را که ممکن است حاوی تراکنش هم باشد ارسال میکند .

4. طراحی شبیه سازی

ما برای طراحی این شبیه سازی از زبان جاوا استفاده کردیم و ابتدا الگوریتم اصلی را پیدا کردیم و خروجی که در مقاله اصلی موجود بود بدست آوردیم بعد سعی کردیم مکانیزم خود را اعمال کنیم که در قسمت نتایج این موضوع را نشان دادیم

در ابتدا بعد ساخت یک کلاس کانفیگ (Config.java) و تنظیم پیش فرض های شبیه سازی ، موجودیت های موجود در این شبیه سازی مثل سیاستمداران و شهروندان را به شبیه سازی اضافه کردیم (در تابع setup) و در حین اضافه کردن مشخص کردیم که چه تعداد خرابکار هستند این موجودیت ها هر کدام یک کلاس مجزا هستند .

در ادامه این موجودیت ها را در کلاس Connection بهم متصل کردیم این متصل کردن به شکل تصادفی است و این تصادفی بودن به این شکل است که هر شهروند به یک سیاست مدار متصل میشود و البته هر سیاستمدار میتواند چند شهروند داشته باشد .

در ادامه تابع run را اجرا کردیم که در هر ثانیه کار میکند و البته زمان در شبیه ساز ما ثانیه است در هر ثانیه شهروندان سعی میکنند درخواستی به سرور بفرستند به این شکل که تابع communicationInSec را فرا می خوانند که در کلاس Citizen تعریف شده و تابع در ابتدا اینترنت را چک میکند که این تابع به

شکلی تصادفی و البته وابسته به تعداد شهروندان و سیاستمداران و یک عدد احتمالی برای نبودن اینترنت کار میکند و خروجی این تابع به شکل بولین به ما میگوید که به اینترنت متصل است یا خیر.

اگر به اینترنت متصل باشیم و اگر مکانیزم کشینگ فعال باشد در ابتدا با فراخوانی تابع checkPreviousRequestInCach برسی میکند که آیا درخواستی از قبل کش شده یا خیر اگر درخواستی کش شده باشد ان درخواست ها را اول ارسال میکند .

در ادامه بعد برسی کشینگ ، به ارسال درخواست فعلی میرسیم که به شکلی تصادفی برسی میشود که آیا در این دور باید تراکنشی انجام شود یا خیر البته این تابع تصادفی هم وابسته به تعداد سیاستمداران و شهروندان و یک عدد احتمالی برای تراکنش است .

اگر باید تراکنش انجام دهد اول این تراکنش را به سیاست مدار ارسال میکند و سیاستمدار هم جواب درخواست را میدهد این جواب ممکن است حاوی خطا یا خرابکاری سیاست مدار باشد که نتیجه این درخواست ها در لیست از نتایج این تابع (communicationInSec) برگشت داده میشود .

و اگر به هر دلیلی شهروند به اینترنت دسترسی نداشت ، درخواست ذخیره میشود البته در شبیه سازی از یک متغیر برای شمارش تعداد درخواست ها و ذخیره سازی استفاده شده نتیجه خروجی منفی درخواست در لیستی از نتایج این تابع (communicationInSec) برگشت داده میشود.

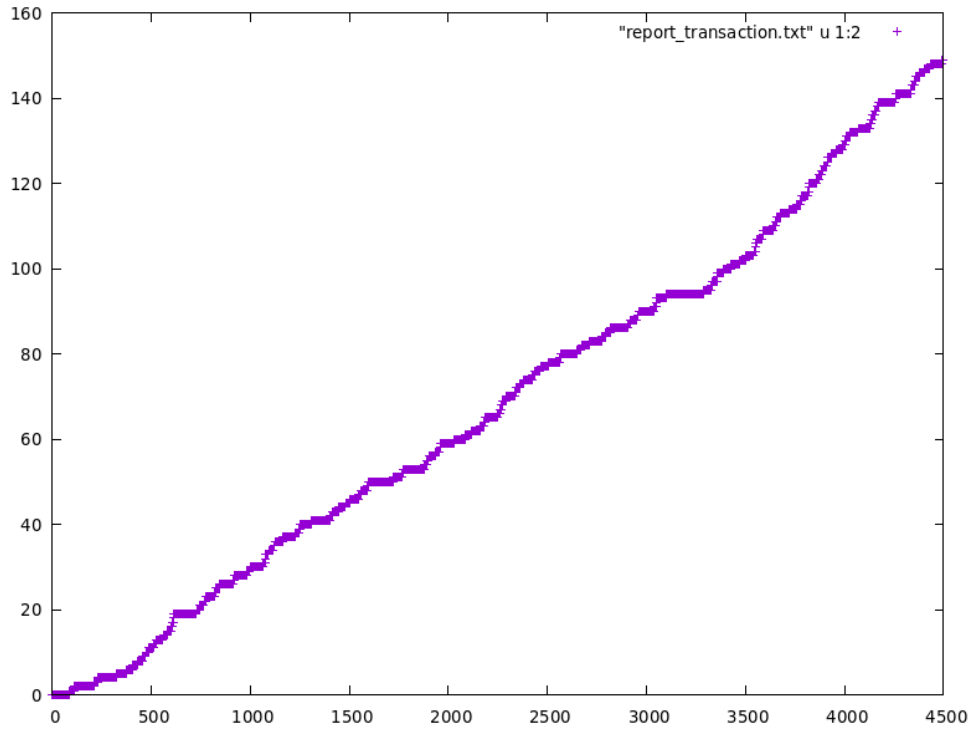
دلیل درست کردن خروجی به شکل لیست به این دلیل است که در درخواست ها کش شده ممکن است چندین درخواست باشد که نتایج به شکل

لیست از خروجی درخواست ها از تابع (communicationInSec) خارج میشود.

بعد خروج نتایج از تابع (communicationInSec) نتایج را تحلیل میکنیم و تعداد تراکنش ها و تعداد تلاش ها برای ارسال درخواست در ثانیه را ذخیره و خروجی را در نمودار به کمک کلاس GraphPanel به نمایش در میاوریم البته خروجی فایل هم از تحلیل به شکل txt هم برای تراکنش و هم برای تعداد درخواست ها ایجاد میکنیم با این خروجی امکان تحلیل بیشتر با برنامه های نمایش گراف وجود دارد برای مثال یک فایل bash در لینوکس برای اجرا خروجی در پکیج gnuplot ایجاد کردیم که با اجرای آن خروجی بهتری از شبیه سازی در دسترس است.

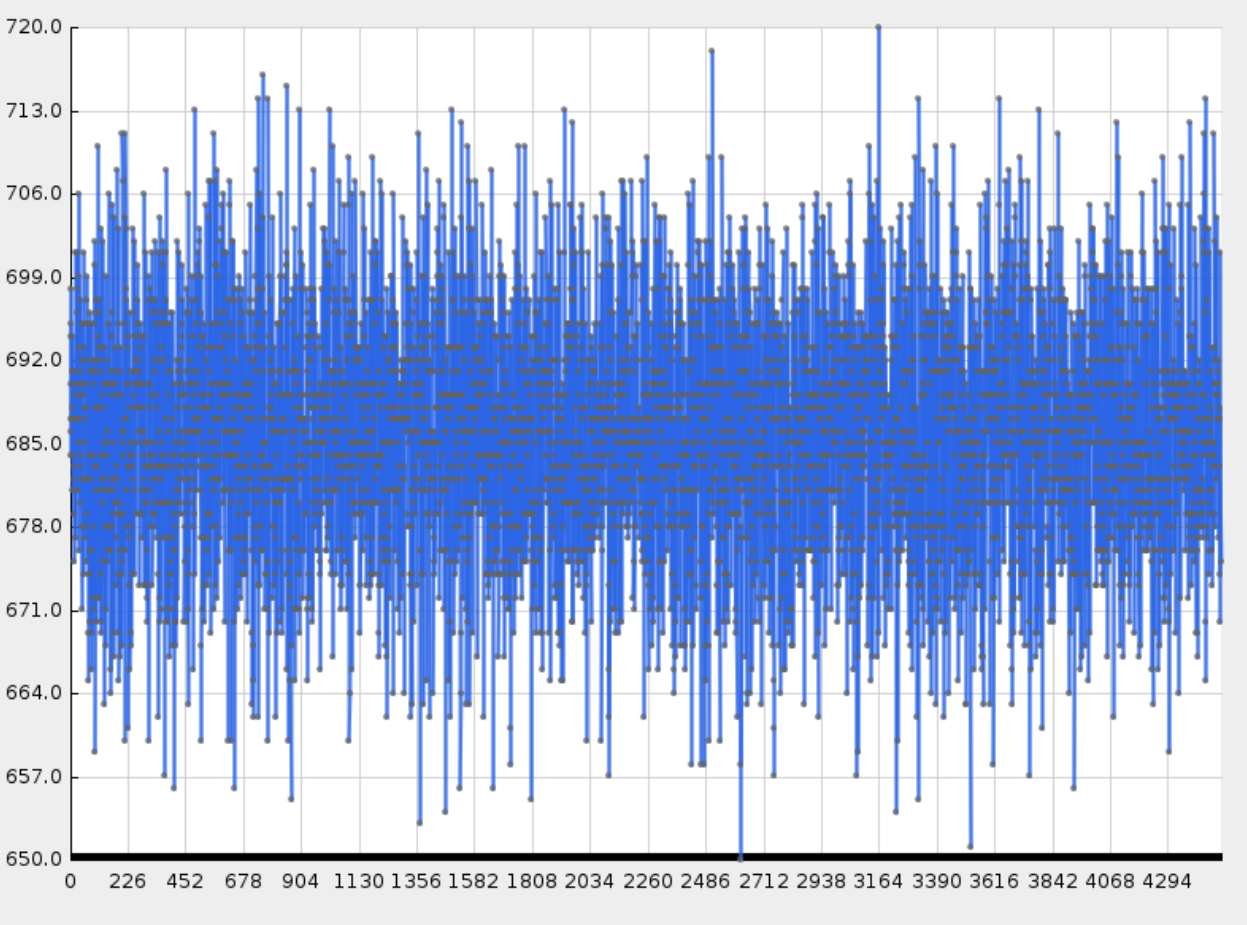
5. نتایج و ارزیابی عملکرد

ما این شبیه سازی را برای حدود ۸۰۰ کاربر شهروند و ۲۰۰ کاربر سیاستمدار در ۴۵۰۰ ثانیه به دو شکل با حالت کشینگ و بدون کشینگ با ۸۰٪ سیاستمدار خرابکار و ۲۵٪ شهروند خرابکار انجام دادیم.



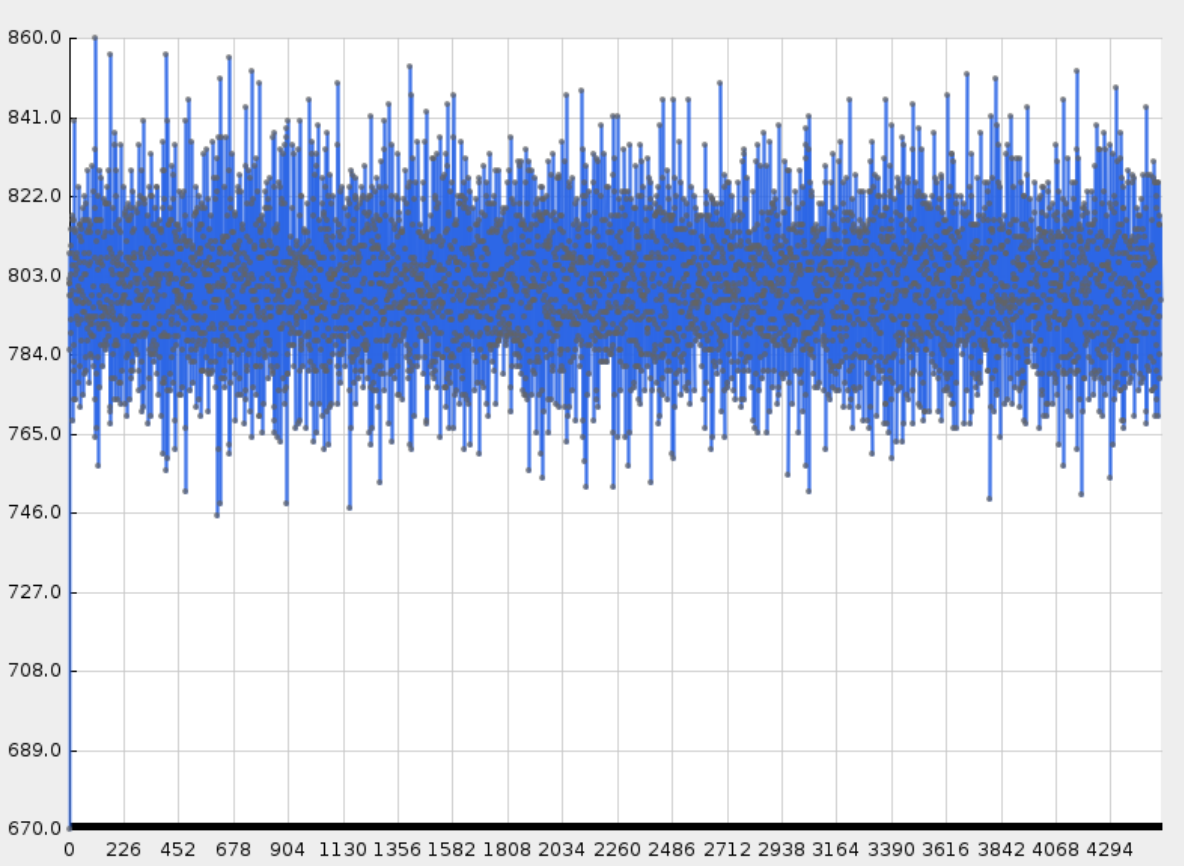

تصویر ۲: تعداد تراکنش های انجام شده در ۴۵۰۰ ثانیه

همانطور که در تصویر ۲ قابل مشاهده است با در نظر گرفتن درصد خرابکاری در هر ۳۵۰۰ ثانیه حدودا ۱۰۰ تراکنش انجام می شود که البته در هر اجرا به دلیل رندوم بودن و تصادفی بودن شبیه سازی امکان تغییر حدودی این درصد ها وجود دارد. این نتیجه همان نتیجه است که در مقاله اصلی Blocene موجود است . در ادامه نتایج اعمال مکانیزم کشینگ خود را نشان می دهیم.




تصویر ۳ : تعداد تلاش ها برای ارسال درخواست های شهروندان به سیاستمدار بدون حالت کشینگ

در حالت بدون کشینگ و طبق آنچه در خروجی تصویر ۳ قابل مشاهده است تعداد تلاش ها موفق برای ارسال درخواست در حدود ۶۸۵ درخواست در هر ثانیه است و تعداد کل درخواست ها در این شبیه سازی ۳۰۸۵۸۵۳ درخواست است




تصویر ۴ : تعداد تلاش ها برای ارسال درخواست های شهروندان به سیاستمدار با حالت کشینگ

و در حالت کشینگ و انچه در تصویر ۴ قابل نمایش است حدود ۸۰۳ درخواست در هر ثانیه است و تعداد تراکنش های کل در این حالت ۳۵۹۹۸۶۷ درخواست است و دلیل رند نشدن کل تراکنش ها هم در این است که بعد ۴۵۰۰ ثانیه یک سری درخواست ها کش شده ممکن است در شهروندان به شکل کشینگ موجود باشد که بعد ۴۵۰۰ ثانیه ممکن است اجرا شوند.

آنچه در این شبیه سازی قابل برداشت است با حالت کشینگ تقریبا همه درخواست ها قابل ارسال هستند و این موضوع در کارایی الگوریتم تاثیر گذار است .

## 6. نتیجه گیری

ما در این مقاله مکانیزمی برای بهبود الگوریتم Blocene در کش کردن درخواستها ارائه دادیم و نشان دادیم که با این روش تقریبا تمام درخواست قابل ارسال است .

این مکانیزم در بهبود الگوریتم Blocene خیلی تاثیر گذار است و امکان بهره وری الگوریتم را بالا میبرد.

## 7. منابع


[1] Satija, Sambhav, et al. "Blockene: A High-throughput Blockchain Over Mobile Devices." 14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20). 2020.

[2] Androulaki, Elli, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." Proceedings of the thirteenth EuroSys conference. 2018.

[3] Kiayias, Aggelos, et al. "Ouroboros: A provably secure proof-of-stake blockchain protocol." Annual International Cryptology Conference. Springer, Cham, 2017.

[4] Kogias, Eleftherios Kokoris, et al. "Enhancing bitcoin security and performance with strong consistency via collective signing." 25th {usenix} security symposium ({usenix} security 16). 2016.

[5] Micali, Silvio, Michael Rabin, and Salil Vadhan. "Verifiable random functions." 40th annual symposium on foundations of computer science (cat. No. 99CB37039). IEEE, 1999.

[6] Pass, Rafael, and Elaine Shi. "Hybrid consensus: Efficient consensus in the permissionless model." 31st International Symposium on Distributed Computing (DISC 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.