دانشگاه صنعتی شریف
دانشکده‌ی مهندسی برق

گزارش شماره دو پایان نامه ارشد برای طرح گرنت همراه اول
مهندسی برق

عنوان:

# طراحی الگوریتم آنلاین برای مدیریت هزینه در شبکه کانال های پرداخت

نگارش:

**مهسا باستان خواه**

استاد راهنما:

**دکتر محمد علی مداح علی**

آبان ۱۴۰۱


بسم الله الرحمن الرحیم

# چکیده

امروزه استفاده از شبکه کانال های پرداخت[۱] مبتنی بر بلاکچین به عنوان یکی از عملی ترین راه حل های مشکل عدم مقیاس پذیری بلاکچین بسیار مورد توجه قرار گرفته است. کاربران با استفاده از شبکه کانال های پرداخت برای تراکنش های روزمره،در عین اینکه از تمام تضمین های امنیتی و محرمانگی بلاکچین بهره مند میشوند، میتوانند از پرداخت کارمزد های سرسام آور بلاکچین خودداری کنند.

نحوه کار کانال پرداخت به صورت خلاصه به شرح زیر است. دو نفر برای ایجاد یک کانال پرداخت باید با فرستادن تراکنش مخصوصی به بلاکچین، مقداری سپرده برای کانال خود ذخیره کنند. بعد از ایجاد کانال دو نفر میتوانند تا سقف سپرده خود تراکنش برون بلاکچینی [۲] برای هم بفرستند و چون این تراکنش ها به صورت محلی[۳] و بدون مراجعه به بلاکچین انجام میشوند بسیار سریع هستند و کارمزد آن ها ناچیز است. در انتها وقتی طرفین تصمیم به بستن کانال خود میگیرند با ارسال تراکنش دیگری به بلاکچین میتوانند سپرده خود را آزاد کنند. بدین ترتیب با تنها دو تراکنش درون بلاکچینی[۴] یکی برای ایجاد کانال و دیگری برای بستن کانال، امکان ارسال صدها تراکنش برون بلاکچینی فراهم میشود.

یکی از مهم ترین محدودیت های کانال پرداخت این است که افراد امکان اضافه کردن سپرده به کانال را فقط و فقط در هنگام ایجاد کانال دارند و اگر بعدا تصمیم به افزایش سپرده خود بگیرند باید کانال را بسته و کانال جدیدی ایجاد کنند که امری هزینه بر است. بنابراین کاربران تمایل دارند مقدار سپرده کافی در کانال از همان ابتدا قرار دهند اما از طرف دیگر نباید بیش از اندازه هم در کانال پول بگذارند زیرا امکان استفاده از این پول را تا بستن کانال نخواهند داشت. در نتیجه کاربران هنگام ایجاد کانال با یک مسأله تصمیم گیری آنلاین روبرو هستند. اما در عمل مساله از این هم پیچیده تر است زیرا میلیون ها کاربر با کانال پرداخت های دو به دویی که تشکیل میدهند، شبکه عظیمی از کانال های پرداخت [۵] را تشکیل میدهند. در این شبکه هر دو نودی که که یک کانال پرداخت مشترک دارند میتوانند بی واسطه برای هم تراکنش بفرستند اما نود هایی که کانال مستقیم با هم ندارند باید با استفاده از سایر نود های شبکه به عنوان واسطه، تراکنش خود را در شبکه مسیریابی و ارسال کنند. به طور مثال شبکه ای به صورت A-B-C را در نظر بگیرید که در آن نود های A و C کانال پرداخت مشترک ندارند اما هر دو با

[۱] payment channel
[۲] بدون مراجعه به بلاکچین off-chain transaction
[۳] local
[۴] on-chain
[۵] payment channel networks

B کانال مشترک دارند؛ در این شبکه A میتواند برای B پول بفرستد و B همان پول را به C ارسال کند و تراکنش A به C با یک واسطه انجام خواهد شد. بنابراین نود های موجود در شبکه میتوانند در دو نقش کاربر(فرستنده یا گیرنده) یا سرویس دهنده(واسطه) ایفای نقش کنند. نود های واسطه در ازای انتقال تراکنش های کاربران کارمزد دریافت میکنند پس تمایل دارند که تا حد امکان تراکنش های بیشتری را مسیریابی کنند؛ اما از طرفی اگر واسطه ها حریصانه تمام تراکنش های کاربران را مسیریابی کنند، کانال هایشان خالی از پول میشود. مثلا در مثال بالا فرض کنید A قصد دارد تعداد تراکنش زیادی برای C بفرستد، اگر B تمام این تراکنش ها را مسیریابی کند، هیچ پولی در کانال B-C برای او باقی نخواهد بود و در عوض Bمقدار زیادی پول در کانال A-B مقدار زیادی پول خواهد داشت. در چنین شرایطی میگوییم کانال B نامتعادل شده است. نامتعادل شدن کانال امر مطلوبی نیست زیرا مانع انتقال تراکنش های بعدی در جهت خالی شده از پول میشود. پس نود های واسطه شبکه کانال های پرداخت در هر لحظه با یک تصمیم گیری آنلاین روبرو هستند؛ اینکه کدام یک از تراکنش های کاربران را انتقال دهند. بنابراین در مجموع میبینیم که نود های شبکه کانال های پرداخت چه هنگام ایجاد کانال و چه بعدا د زمان ارسال تراکنش های برون بلاکچینی باید مدام تصمیمات آنلاینی در خصوص مدیریت کانال خود بگیرند.

نود های شبکه کانال های پرداخت نیاز به الگوریتمی برای مدیریت کانال خود دارند. این الگوریتم باید آنی باشد به این معنی که الگوریتم برای اتخاذ تصمیمات زمان زیادی ندارد. طراحی یک الگوریتم بهینه آنلاین که درباره مدیریت سپرده های نود ها تصمیم گیری میکند نه تنها در این حوزه مورد نیاز است بلکه میتواند در حوزه های دیگر همچون شبکه های مخابراتی برای حل مسأله admission control هم سود بخش باشد. در این پایان نامه الگوریتم آنلاینی برای مدیریت یک تک کانال پرداخت طراحی میکنیم. الگوریتم ما یک الگوریتم آنلاین است به این معنی که هیچ فرض خاصی روی توزیع تراکنش های آینده ندارند و تنها با اطلاعات گذشته و لحظه حال تصمیمی اتخاذ میکند. در این پایان کران بالای هزینه الگوریتممان را برای بدترین دنباله تراکنش [۶] ممکن اثات میکنیم؛ و در نهایت با پیاده سازی نشان میدهیم که الگوریتم ما در عمل بسیار بهتر از تضمین تئوری اثبات شده عمل میکند و همچنین دو heuristic با الهام از الگوریتم اصلی طراحی میکنیم که در عمل هزینه را تا نصف هزینه الگوریتم اصلی کاهش میدهد.

**کلیدواژه‌ها:** بلاکچین، شبکه کانال های پرداخت، الگوریتم آنلاین، admission control

---

[۶] worst-case analysis

# فهرست مطالب


**۱ مقدمه** ۷

۱-۱ اهمیت موضوع . . . . . ۱۰

۱-۲ دست آورد های تحقیق . . . . . ۱۱

۱-۳ ساختار پایان‌نامه . . . . . ۱۲

**۲ مفاهیم اولیه** ۱۴

۲-۱ تراکنش ها در بیتکوین . . . . . ۱۴

۲-۲ تراکنش های کانال پرداخت . . . . . ۱۶

۲-۲-۱ ایجاد کانال . . . . . ۱۶

۲-۲-۲ استفاده از کانال . . . . . ۱۷

۲-۳ تراکنش های با واسطه . . . . . ۲۰

۲-۴ Lightning Network . . . . . ۲۱

۲-۵ الگوریتم های آنلاین . . . . . ۲۲

**۳ مدلسازی مسئله** ۲۴

۳-۱ کانال پرداخت . . . . . ۲۴

۳-۲ تراکنش ها . . . . . ۲۵

۳-۳ شارژ کردن کانال روی بلاکچین . . . . . ۲۷
۳-۴ متعادل کردن کانال روی شبکه کانال های پرداخت . . . . . ۲۸
۳-۵ تعریف الگوریتم بهینه آفلاین و الگوریتم آنلاین . . . . . ۲۸
۳-۶ مثالی از مسئله کانال های دو طرفه . . . . . ۳۱

**۴ کارهای پیشین** **۳۴**
۴-۱ اهمیت متعادل نگه داشتن کانال های شبکه پرداخت . . . . . ۳۴
۴-۲ متعادل کردن کانال برون بلاکچینی . . . . . ۳۵
۴-۳ استفاده از الگوریتم های آنلاین برای حل مسائل شبکه کانال های پرداخت . . . . . ۳۵
۴-۴ ارتباط مسئله ما با مسائل مشابه در شبکه های مخابراتی . . . . . ۳۶

**۵ نتایج تئوری** **۳۷**
۵-۱ دو زیر الگوریتم پر کاربرد . . . . . ۳۸
۵-۲ توصیف الگوریتم مساله کانال دو طرفه . . . . . ۴۱
۵-۳ اثبات ضریب رقابتی مساله کانال دو طرفه . . . . . ۴۵

**۶ پیاده سازی** **۵۳**
۶-۱ پیدا کردن الگوریتم OFF به کمک `dynamic programming` . . . . . ۵۳
۶-۲ مقایسه هزینه الگوریتم ON و OFF . . . . . ۵۶
۶-۳ بررسی پارامتر های توابع هزینه برای `Lightning Network` . . . . . ۶۱

**۷ نتیجه‌گیری** **۶۳**

# فصل ۱

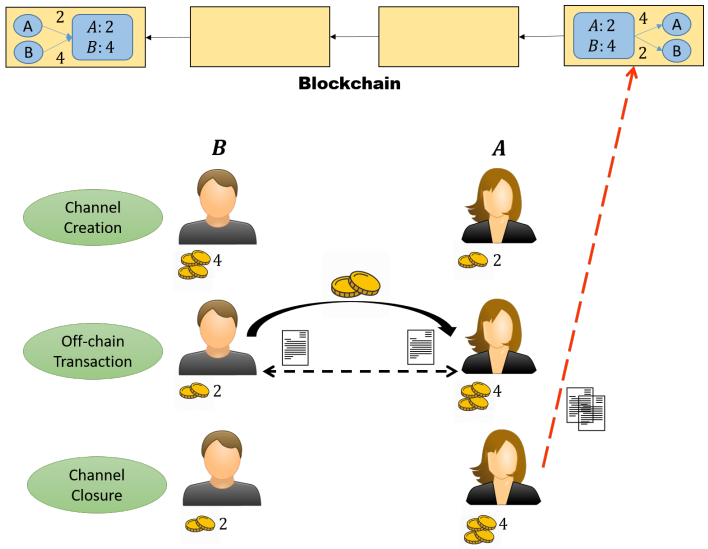
# مقدمه

بلاکچین هایی همچون بیتکوین به دلیل ماهیت توزیع شده والگوریتم اجماع پیچیده و وقت گیری که دارند با مشکل عدم مقیاس پذیری روبرو هستند. عدم مقیاس پذیری به این معنی است که سیستم نمیتواند تعداد بسیار زیاد تراکنش را پردازش کند. به طور مثال بلاکچین بیتکوین تنها میتواند ۷ تراکنش در ثانیه را پردازش کند در حالیکه رقبای متمرکز بلاکچین همچون visa بیش از هزاران تراکنش را در هر ثانیه پردازش میکنند. به علاوه، حتی وقتی تراکنش ها وارد بلاکچین میشوند تایید شدن آن ها معمولا حداقل چند دقیقه به طول می انجامد، به طور مثال در بلاکچین بیتکوین نزدیک یک ساعت طول میکشد تا یک تراکنش تأیید نهایی شود. یکی از مورد استقبال ترین راه حل هایی که برای حل مشکل مقیاس ناپذیری و کندی بلاکچین ارائه شده است استفاده از شبکه کانال های پرداخت[۱] است. شبکه کانال های پرداخت اولین بار با پیاده سازی Lightning Network روی بلاکچین بیتکوین معرفی شد. [۱] بعد ها شبکه کانال های پرداخت Raiden هم با الهام از Lightning Network بر بلاکچین اتریوم[۲] توسعه داده شد. [۲] کاربران میتوانند با ارسال یک تراکنش ایجاد کانال [۳] روی بلاکچین، یک کانال پرداخت ایجاد کنند. با این تراکنش در واقع طرفین کانال پرداخت، مقداری پول را در این کانال پرداخت به سپرده میگذارند. پس از ایجاد کانال، طرفین میتوانند بدون مراجعه به بلاکچین و با رد و بدل کردن تعدادی امضای دیجیتال برای هم تراکنش محلی فوری[۴] با کارمزد بسیار اندک و بفرستند. مبادله امضاهای دیجیتال برای حفظ امنیت مالی طرفین الزامی است.

---

[۱] payment channel network
[۲] Ethereum
[۳] channel creation
[۴] instant

شکل ۱-۱ نحوه کار یک کانال پرداخت را نشان میدهد. ابتدا کاربر A ۲ واحد پول و کاربر B ۴ واحد پول سپرده میکند و یک کانال پرداخت میسازند. تراکنش ایجاد کانال روی بلاکچین قرار میگیرد. پس از ایجاد کانال امکان ارسال تراکنش برون بلاکچینی فراهم میشود. B میخواهد برای A دو واحد پول واریز کند پس A و B امضاهای دیجیتال رد و بدل میکنند و دو واحد پول به صورت برون بلاکچینی به موجودی A اضافه میشود. پس از مدتی A تصمیم میگیرد کانال را ببندد؛ از امضاهای رد و بدل شده پیشین استفاده میکند تا یک تراکنش بستن کانال ایجاد کند. پس از اجرای این تراکنش هر کدام از A و B سه واحد پول میگیرند. توجه کنید که در مرحله دوم A و B میتوانند به تعداد نامحدود تراکنش برون بلاکچینی ایجاد کنند. پس با دو تراکنش درون بلاکینی، امکان ایجاد تعداد نامحدود ترامنش برون بلاکچینی ارزان و سریع فراهم شد. اما باید توجه کرد که مجموع موجودی A و B که به آن ظرفیت کانال [۵] می گویند همواره عدد ثابت ۶ است و قابل افزایش یا کاهش نیست.




شکل ۱-۱: نحوه شکل گیری، استفاده و بستن یک کانال پرداخت

از اتصال کاربران مختلف با کانال های پرداخت یک شبکه از کانال های پرداخت ایجاد میشود که میتواند افرادی که کانال پرداخت مستقیم به هم ندارند را هم با یک یا تعدادی واسطه به هم متصل کند. مثلا ۳ کاربر A-B-C را در نظر بگیرید که A-B و B-C کانال پرداخت دارند. در این صورت A و C اگرچه کانال پرداخت مشترک ندارند اما میتوانند از B به عنوان واسطه استفاده کنند و برای هم تراکنش برون بلاکچینی ارسال کنند؛ بدین صورت که A مقدار پول مورد نظر را برای B می فرستد و B همان مقدار پول را برای C میفرستد. این دو تراکنش ها atomic هستند که به این معنی است که یا هر دو آنها انجام

[۵] capacity

میشوند و یا هر دو برگشت میخورند. معمولا فرد واسطه یعنی B مقداری کارمزد از A میگیرد اما این کارمزد در برابر کارمزد های تراکنش های درون بلاکچینی بسیار ناچیز است و صرفا نقش ایجاد انگیزه برای واسطه ها را دارد. البته گاهی نود های واسطه ممکن است برخی تراکنش ها را به دلایلی رد کنند. مثلا ممکن است اندازه تراکنش بیشتر از موجودی آن نود واسطه در کانال باشد یا اینکه موجودی نود واسطه را در حد غیر قابل قبولی کاهش دهد و یا اینکه میزان کارمزد آن مطلوب نود واسطه نباشد.

یکی از مشکلات بسیار مهم شبکه کانال های پرداخت این است که بعد از ایجاد کانال هیچ راهی برای افزودن سپرده به کانال وجود ندارد. مثلا در مثال بالا در کانال A-B فرض کنید با شروع از سپرده اولیه ۴ – ۲ ،A ۲ تراکنش هر کدام به ارزش ۱ بیتکوین برای B میفرستد؛ پس از انجام این دو تراکنش موجودی آنها در کانال به ترتیب ۶ – ۰ خواهد بود. پس از این تا زمانی که B تراکنشی برای A نفرستد، A نمیتواند تراکنشی بفرستد زیرا موجودی اش صفر است. به کانال پرداختی که در آن موجودی یک نفر صفر(یا بسیار کم است) کانال نامتعادل [۶] میگوییم. کانال های نامتعادل برای کاربران به خصوص برای نود های واسطه اصلا مطلوب نیستند زیرا امکان ایجاد تراکنش از یک سمت کانال را به کل از بین میبرند. در این پایان نامه گاهی نود های واسطه را سرویس دهنده [۷] می نامیم. نود های سرویس دهنده نود هایی هستند که با هدف درآمد سازی به شبکه کانال های پرداخت ملحق میشوند و با ذخیره کردن مقدار چشم گیری سپرده، تعداد زیادی کانال با کاربران زیادی ایجاد میکنند تا تراکنش های آنها را مسیریابی کنند و در ازای آن کارمزد بگیرند. داشتن کانال های نامتعادل توانایی سرویس دهنده ها را در انتقال تراکنش ها از یک جهت کاهش میدهد و برای کسب و کار آنها مشکل ایجاد میکند. Lightning network دو راه حل را برای حل مشکل کانال های نامتعادل پیشنهاد میدهد:

۱. شارژ کردن درون بلاکچینی: در این روش طرفین کانال نامتعادل آن کانال را میبندند و کانال جدیدی با سپرده بیشتر باز میکنند. این عمل باعث ایجاد دو تراکنش درون بلاکچینی میشود. یک تراکنش برای بستن کانال قدیمی و یک تراکنش برای ایجاد کانال جدید.

۲. متعادل کردن برون بلاکچینی: این روش بدون مراجعه به بلاکچین و صرفا با تعدادی تراکنش برون بلاکچینی توزیع سپرده ها را در کانال نامتعادل تغییر میدهد و به نسبت روش قبل ارزان تر است. در بخش ؟؟ به طور مفصل این روش را توضیح میدهیم.

از آنجاییکه هر دو روش بالا هزینه بر هستند و محدودیت هایی را اعمال میکنند، تصمیم گیری بر

[۶] depleted channel
[۷] service provider

سر اینکه چه زمانی کدام یک از آنها انجام گیرد تصمیم سختی است. همچنین توجه کنید که به عنوان یک کاربر یا یک نود واسطه، معمولا نود اطلاعات دقیقی از تراکنش های آینده ندارد و در نتیجه نود ها باید سیاست تصمیم گیری ای را اتخاذ کنند که بر اساس تاریخچه و بدون فرضی روی تراکنش های آینده، تصمیم گیری میکند.

هدف این پایان نامه این است که سیاست آنلاینی طراحی کند یک تک کانال پرداخت را در کلی ترین حالت ممکن در نظر میگیرد و به سوالات زیر که برای بیشینه کردن سود و کمینه کردن هزینه طرفین کانال مطرح میشود پاسخ میدهد:

۱. چه زمانی ایجاد یک کانال پرداخت نسبت به ارسال درون بلاکچینی تراکنش به مقرون به صرفه است؟

۲. اگر تصمیم به ایجاد کانال پرداخت شد، طرفین چه مقدار سپرده باید در آن قرار دهند؟

۳. اگر طرفین میخواهند نقش واسطه را ایفا کنند چه تراکنش هایی را باید بپذیرند و چه تراکنش هایی را نپذیرند؟

۴. اگر کانال پرداخت نامتعادل شد، طرفین باید کدام یک از راه های مقابله با کانال نامتعادل را اتخاذ کنند و سپرده کانال را چقدر باید تغییر دهند؟

## ۱-۱ اهمیت موضوع

هدف از طراحی شبکه کانال های پرداخت ایجاد بستری ارزان و سریع برای انجام تراکنش های کوچک و روزانه [۸] است. بهره بری کاربران از شبکه کانال های پرداخت تا حد زیادی به نحوه مدیریت کانال توسط آنها و سرویس دهنده ها بستگی دارد. مدیریت نادرست کانال ها توسط کاربران میتواند منجر به نامتعادل شدن کانال های آن ها شود و معمولا هزینه اصلاح یک کانال نامتعادل بسیار زیاد است. همچنین مدیریت نادرست کانال ها توسط سرویس دهنده هاهم به ضرر خود سرویس دهنده ها و هم به ضرر کاربران است. اگر سرویس دهنده ها نتوانند کانال های خود را درست مدیریت کنند، سود آنها کاهش می یابد و انگیزه ای برای ارائه خدمات نخواهند داشت که با توجه به اهمیت حیاتی سرویس دهنده

[۸] micro payment

ها برای شبکه، این امر بسیار مضر است. با بررسی آخرین داده های موجود از Lightning Network [۳] میتوان دید که در سال ۲۰۲۱ حدود ۶۳۰۰ در شبکه وجود دارد که بیش از ۵۰ درصد آن ها تنها از ۱۰ سرویس دهنده خدمات میگیرند. یعنی اگر ۱۰ سرویس دهنده اصلی Lightning Network عملکرد مناسبی نداشته باشند، نیمی از شبکه مختل خواهد شد! در واقع بدون وجودسرویس دهنده ها، امکان ارسال تراکنش های با واسطه از بین میرود و همه کاربران مجبورند کانال های دو به دو با هم ایجاد کنند.

در نتیجه ارائه الگوریتمی که این مسئله مدیریت کانال را در یک مدل واقع بینانه، با کمترین فروض محدود کننده و به صورت بهینه حل کند، بسیار ارزشمند است.

## ۱-۲ دست آورد های تحقیق

در این پایان نامه برای حل مسأله مدیریت آنلاین کانال های پرداخت، ابتدا از حل تئوری یک نسخه بسیار ساده شده و غیرواقع گرایانه مسأله شروع میکنیم و سپس در دو گام مدل را پیچیده تر واقع گرایانه تر میکنیم طوری که مسأله نهایی تا حد خوبی بیشتر پیچیدگی های کانال های پرداخت در دنیای واقعی را در بر دارد. این دو زیر مسأله به شرح زیر هستند:

۱. **زیر مسأله ۱ (کانال یکطرفه همیشه پذیرنده[۹])** کانال پرداختی ساده و غیرواقع نگرانه ای با دو کاربر A و B را در نظر بگیرید که در آن همیشه فقط A برای B پول میفرستد، یعنی کانال یکطرفه است. همچنین فرض کنید که باید تمام تراکنش ها حتما انجام شود و کاربران امکان رد کردن تراکنش ها را ندارند(اگر A یک کاربر عادی باشد رد تراکنش به این معنی است که A تراکنشش را خارج از کانال پرداخت و از طرق دیگر انجام میدهد و اگر A یک سرویس دهنده باشد رد کردن تراکنش به این معنی است که A تصمیم میگیرد از کارمزد این تراکنش صرف نظر کند و این تراکنش را مسیریابی نکند). همچنین برای ساده سازی فرض کنید که اگر پول A در کانال پرداخت تمام شد، باید کانال را ببندد و کانال جدید باز کند یا به عبارت دیگر تنها راه متعادل کردن کانال، شارژ کردن درون بلاکچینی است و متعادل کردن برون بلاکچینی برای ساده سازی مجاز نیست. این مدل، اولین و ساده ترین مدلی است که بررسی میکنیم و برای آن الگوریتمی

[۹] Unidirectional stream without rejection

آنلاین با نسبت رقابتی [۱۰] برابر ۲ ارائه میدهیم و اثبات میکنیم که این بهترین نسبت رقابتی ای ست که یک الگوریتم آنلاین میتواند به آن دست یابد.

۲. **زیر مسأله ۲ (کانال یکطرفه مجاز به رد تراکنش[۱۱])** در این زیر مسأله همانند مدل قبلی جهت تراکنش ها همیشه یکطرفه است اما این بار دارندگان کانال میتوانند تصمیم بگیرند کدام تراکنش ها را انتقال دهند و کدام ها را رد کنند. مشابه مدل قبل متعادل کردن برون بلاکچینی مجاز نیست. برای این مدل الگوریتم آنلاین با نسبت رقابتی $\frac{\sqrt{۵}-۱}{۲} + ۲$ ارائه میدهیم و اثبات میکنیم که این الگوریتم بهینه است.

۳. **مسأله اصلی (کانال دوطرفه [۱۲])** در کلی ترین حالت مسأله تراکنش ها در هر دو جهت وجود دارند و صاحبان کانال نه تنها میتوانند تراکنش ها را به دلخواه بپذیرند یا رد کنند بلکه میتوانند از هر دو روش شارژ کردن درون بلاکچینی و متعادل کردن برون بلاکچینی برای متعادل کردن کانال خود استفاده کنند. برای این مدل الگوریتم آنلاین با نسبت رقابتی $۲\log C + ۷$ طراحی میکنیم. ( $C$ یک عدد ثابت است که بستگی به ویژگی های گراف شبکه کانال های پرداخت دارد و مثلا در `Lightning Network` حدودا برابر ۴ است ). همچنین به عنوان کران پایین نشان میدهیم که هیچ الگوریتم آنلاینی با نسبت رقابتی $o(\sqrt{\log C})$ وجود ندارد.

الگوریتم ها و اثبات های تئوری زیرمسأله ۱ و ۲ به عنوان بلوک های سازنده برای حل مسأله اصلی مورد استفاده قرار میگیرد.

## ۱-۳ ساختار پایان‌نامه

این پایان‌نامه شامل پنج فصل است. فصل دوم دربرگیرنده‌ی تعاریف اولیه‌ی مرتبط با پایان‌نامه است. در فصل سوم مسئله‌ی دورهای ناهمگن و کارهای مرتبطی که در این زمینه انجام شده به تفصیل بیان می‌گردد. در فصل چهارم نتایج جدیدی که در این پایان‌نامه به دست آمده ارائه می‌گردد. در این فصل، مسئله‌ی درخت‌های ناهمگن در چهار شکل مختلف مورد بررسی قرار می‌گیرد. سپس نگاهی کوتاه به

[۱۰] `competitive ratio` معیاری است که هزینه یک الگوریتم آنلاین را با هزینه الگوریتم بهینه آفلاین که از پیش به تمام تراکنش های آینده دسترسی دارد، مقایسه میکند. در قسمت ۲-۵ به طور مفصل این معیار و نحوه محاسبه آن را توضیح میدهیم.

[۱۱] Unidirectional stream with rejection

[۱۲] Bidirectional stream

| مسئله | نسبت رقابتی | کران پایین |
|---|---|---|
| Unidirectional stream without rejection | 2 | 2 |
| Unidirectional stream with rejection | $2+\frac{\sqrt{5}-1}{2}$ | $2+\frac{\sqrt{5}-1}{2}$ |
| Bidirectional stream | $7+2\log C$ | $\theta(\sqrt{\log C})$ |

جدول ۱-۱: خلاصه نتایج تئوری این پایان نامه. ستون اول نام (زیر)مسأله، ستون دوم نسبت رقابتی و ستون

مسئله‌ی مسیرهای ناهمگن خواهیم داشت. در انتها با تغییر تابع هدف، به حل مسئله‌ی کمینه کردن حداکثر اندازه‌ی درخت‌ها می‌پردازیم. فصل پنجم به نتیجه‌گیری و پیش‌نهادهایی برای کارهای آتی خواهد پرداخت.

# فصل ۲

# مفاهیم اولیه

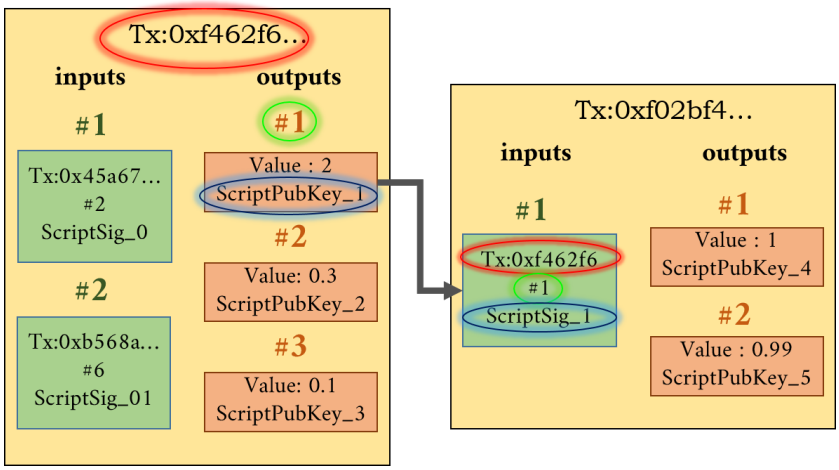
در این فصل مفاهیم اولیه لازم برای فهم مسأله و نتایج پایان نامه را مطرح میکنیم. ابتدا نحوه پردازش تراکنش ها در بلاکچین بیتکوین را توضیح میدهیم و سپس توضیح میدهیم که تراکنش های یک کانال پرداخت چه تفاوتی با تراکنش های عادی درون بلاکچینی دارند و چگونه میتوان تراکنش برون بلاکچینی امن داشت. در این بخش هنگام توضیح جزئیات پروتکل ها بلاکچین بیتکوین و شبکه کانال های پرداخت آن یعنی Lightning network را به عنوان معیار در نظر میگیریم زیرا اولا امروزه Lightning network با داشتن بیش از ۱۷۰۰۰ نود، پرکاربر ترین شبکه کانال های پرداخت موجود است [۴] و ثانیا مفاهیم پایه ای تراکنش های درون بلاکچینی و برون بلاکچینی کمابیش برای تمام بلاکچین ها یکسان است و تنها تفاوت در پروتکل های پیاده سازی شده است، پس تفاوت چندانی ندارد که کدام بلاکچین را به عنوان معیار قرار دهیم.

## ۲-۱ تراکنش ها در بیتکوین

بیتکوین یک بلاکچین UTXO-based است، در این سیستم هر تراکنش یک یا تعدادی ورودی و یک یا تعدادی خروجی دارد. در هر تراکنش مجموع بیتکوین ورودی ها برابر است با مجموع بیتکوین خروجی ها و کارمزد تراکنش. شکل ۲-۱ را ببینید. هر تراکنش یک هش [۱] یکتا دارد که شناسه تراکنش محسوب میشود. هر کدام از ورودی های یک تراکنش یکی از خروجی های یک تراکنش قدیمی تر را خرج میکند.

[۱] hash

هر خروجی دو داده در بر دارد : ۱)مقدار پول موجود در آن ۲) یک کد قفل کننده (ScriptPubKey) که کلید عمومی [۲] و سایر مشخصات کسی که میتواند خروجی را خرج کند مشخص میکند. مثلا در شکل ۲-۱ به خروجی شماره ۱ تراکنش سمت چپ دقت کنید. این خروجی دو بیتکوین دارد و کد قفل کننده ScriptPubKey_1، کلید عمومی کسی که میتواند این پول را خرج کند مشخص میکند. این خروجی توسط ورودی شماره ۱ تراکنش سمت راست خرج میشود. هر ورودی سه داده را در بر دارد ۱) هش تراکنشی که میخواهد یکی از خروجی های آن را خرج کند (در این مثال هش تراکنش سمت چپ که با قرمز رنگ مشخص شده است) ۲) شماره خروجی مورد نظر( در این مثال، عدد ۱ که با سبز رنگ مشخص شده است ) ۳) یک کد باز کننده قفل که شامل امضای صاحب پول و سایر اثبات های مورد نیاز خروجی است (در این مثال، ScriptSig_1 که با رنگ سرمه ای مشخص شده است) . همچنین دقت کنید که در تراکنش سمت راست ورودی ۲ بیتکوین دارد و مجموع خروجی ها ۱/۹۹ بیتکوین است؛ ۰/۰۱ بیتکوین هم به عنوان کارمزد شبکه در نظر گرفته شده است.




شکل ۲-۱: ساختار یک تراکنش در بیتکوین

توجه کنید که کد قفل کننده و باز کننده قفل باید سازگار باشند مثلا دو عبارت زیر سازگار هستند:

**ScriptPubKey:** locked with <PubKey_A>
**ScriptSig:** Signature of A

کد قفل کننده میتواند شروط بیشتر و پیچیده تری هم برای خرج کننده خروجی ایجاب کند. مثلا به کد قفل کننده و بازکننده زیر توجه کنید که به امضای هر دو کاربر A و B نیازدارد. این خروجی مانند یک حساب دو کاربره عمل میکند زیرا برای خرج کردن پول آن تایید هر دو کاربر A و B لازم است.

[۲] public key

**ScriptPubKey:** locked with <PubKey_A> and <PubKey_B>
**ScriptSig:** Signature of A and signature of B

**قفل زمانی** تراکنش های بیتکوین میتوانند شامل جزئیات دیگری مانند قفل زمانی[۳] هم باشند. قفل زمانی به این معنی است که یک تراکنش را پیش از زمان مقرر نمیتوان به بلاکچین ارسال کرد. به طور مثال اگر شما تراکنشی با قفل زمانی December 31 بسازید و آن را پیش از این تاریخ به بلاکچین ارسال کنید، ماینر ها این تراکنش را ثبت نمیکنند و تا روز December 31 صبر کرده و بعد آن را ثبت میکنند.

**خروجی های خرج نشده** [۴] به خروجی هایی که تاکنون خرج نشده اند Unspent Transaction Output (UTXO) میگویند. ماینر[۵] ها در شبکه بیتکوین دو وظیفه اصلی دارند:

۱. اطمینان حاصل کنند که هیچ خروجی ای بیش از یکبار خرج نمیشود .

۲. بررسی کنند که هر کد باز کننده به درستی کد قفل کننده متناظر را باز میکند.

## ۲-۲ تراکنش های کانال پرداخت

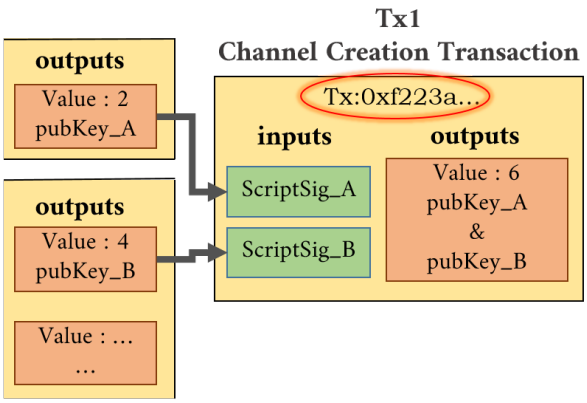
در این بخش نوع ساخت و استفاده از کانال پرداخت های بر مبنای زمان[۶] ها را توضیح میدهیم. کانال پرداخت های رایج در Lightning Network معمولا از نوع punishment based payment channel هستند که جزئیات پیچیده تری نسبت به کانال های بر مبنای زمان دارند. با این وجود چون اصول اولیه و کلیات هر دو این پروتکل ها مشابه هم است، فهم اصول کانال های بر مبنای زمان کافی است. برای خواندن درباره تفاوت این دو پروتکل ایجاد کانال میتوانید به منبع [؟] مراجعه کنید.

### ۲-۲-۱ ایجاد کانال

همانطور که پیش از این گفته شد، دو کاربر برای ایجاد کانال پرداخت باید یک تراکنش درون بلاکچینی "ایجاد کانال" بسازند. شکل ۲-۲ تراکنش Tx1 را نشان میدهد که نمونه ای از یک تراکنش ایجاد کانال

[۳] time lock
[۴] UTXO
[۵] miner
[۶] time based payment channels

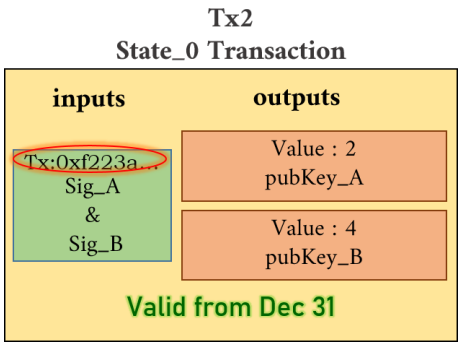
است. A یکی از UTXO هایش به ارزش ۲ بیتکوین و B یکی از UTXO هایش به ارزش ۴ بیتکوین را در کانال سپرده میکنند. خروجی Tx1 یک UTXO دو کاربره با موجودی ۶ است.




شکل ۲-۲: مثالی از یک تراکنش ایجاد کانال پرداخت. این تراکنش درون بلاکچینی است یعنی روی بلاکچین بیتکوین فرستاده میشود.

## ۲-۲-۲ استفاده از کانال

همزمان یا اندکی پیش از امضای تراکنش درون بلاکچینی Tx1، A و B مشترکا یک تراکنش برون بلاکچینی (Tx2) هم ساخته و هر دو آن امضا میکنند اما آن را روی بلاکچین نمیفرستند، این تراکنش تنها در حافظه محلی A و B ذخیره میشود.

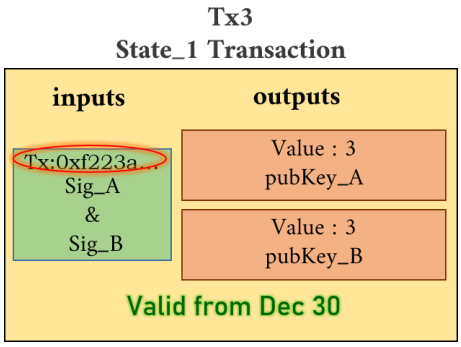



شکل ۲-۳: تراکنش حالت صفر کانال پرداخت. این تراکنش برون بلاکچینی است یعنی توسط A و B مشترکا ساخته و امضا شده و ذخیره می شود ولی تا زمان بسته شدن کانال روی بلاکچین قرار نمیگیرد.

Tx2 تک خروجی تراکنش Tx1 را خرج میکند و ، پول موجود در کانال را به همان نسبت اولیه ۴ – ۲ بین A و B تقسیم میکند. تراکنش Tx2 تنها پس از زمان مقرر قفل زمانی(در این مثال December 31) بر بلاکچین ثبت میشود. Tx2 در واقع توزیع پول در کانال را در لحظه ایجاد آن یا لحظه صفر نشان

میدهد به همین دلیل به آن تراکنش لحظه صفر میگوییم. در صورتی که هیچ تراکنشی بین A و B انجام نگیرد و پس از مدتی یکی از A یا B تصمیم بگیرد کانال را ببندد، آن فرد میتواند تراکنش Tx2 را روی بلاکچین قرار دهد. چون Tx2 پیش از این توسط هر دو A و B امضا شده است پس تراکنش معتبری است و بعد از تاریخ قفل زمانی آن A و B هر کدام میتوانند سهم خود را از کانال پرداخت بگیرند و کانال بسته میشود. اما در عمل A و B کانال پرداخت ساخته اند تا از آن استفاده کنند نه اینکه آن را بلافاصله ببندند پس تراکنش Tx2 عملا هیچگاه استفاده نمیشود. این تراکنش تنها و تنها ساخته میشود تا به هر دو طرف تضمین دهد که اگر طرف دیگر پاسخگو نبود، پول آن ها در کانال قفل نمیماند و هر کدام میتوانند سهم خود را از کانال دریافت کنند.

اکنون با یک مثال توضیح میدهیم که در عمل چگونه از کانال پرداخت ایجاد شده در شکل ۲-۲ استفاده میشود. فرض کنید B میخواهد برای A ۱ بیتکوین در کانال پول بریزد. B تراکنش برون بلاکچینی Tx3 نمایش داده شده در شکل ۲-۴ را میسازد و امضا میکند و برای A میفرستد. A هم Tx3 را امضا کرده و ذخیره میکند. Tx3 توزیع پول موجود در کانال را به ۳ – ۳ تغییر میدهد. هرگاه هر کدام از A یا B بخواهند این کانال را ببندند کافی است Tx3 را روی بلاکچین بفرستد و در تاریخ December 30 سهم خود از کانال را پس بگیرند.

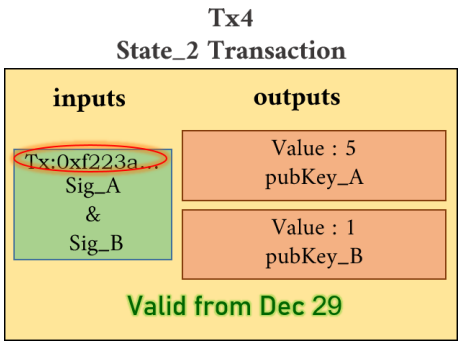



شکل ۲-۴

اما یک مشکل وجود دارد؛ چگونه تضمین دهیم که هیچ کدام از طرفین تراکنش قدیمی تر Tx2 را روی بلاکچین قرار نمیدهند تا کانال را با یک توزیع پول قدیمی ببندند؟ دقت کنید که هر دو Tx2 و Tx3 تک خروجی تراکنش Tx1 را خرج میکنند بنابراین فقط یکی از آن ها قابل اجرا روی بلاکچین است. محدودیت زمانی اعمال شده روی خروجی تراکنش های Tx2 و Tx3 تضمین کننده این است که تراکنشی که دیرتر تولید شده است، Tx3، زودتر اجرا خواهد شد؛ با یک مثال این موضوع را توضیح میدهیم. فرض کنید A میخواهد سر B کلاه بگذارد و بدون هیچ اطلاع قبلی تراکنش Tx2 را روی بلاکچین

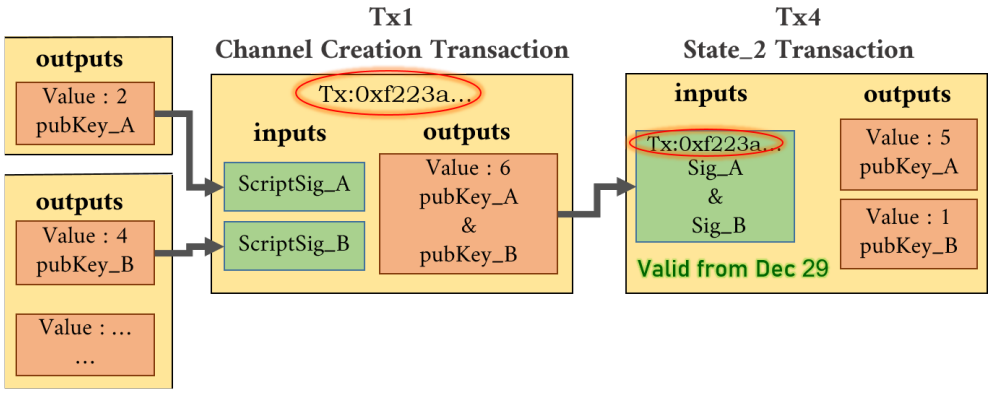
قرار میدهد؛ B با دیدن این عمل، تراکنش Tx3 را به بلاکچین میفرستد. قفل زمانی Tx3 روز December 30 باز میشود پس این تراکنش یک روز زودتر از Tx2 قابل اجرا است. تراکنش Tx3 در روز December 30 انجام میشود و بعد از آن، تراکنش Tx2 دیگر قابل اجرا نخواهد بود زیرا ورودی آن قبلا توسط Tx3 مصرف شده است.

منطق مشابهی در تمام مدت استفاده از کانال استفاده میشود مثلا اگر روز بعد B بخواهد به A ۲ بیتکوین بدهد، باید تراکنش Tx4 را تولید کند و امضا کرده و برای او بفرستد.




شکل ۲-۵

در نهایت وقتی طرفین تصمیم بگیرند کانال پرداخت را ببندند، آخرین تراکنش را روی بلاکچین میفرستند یعنی در نهایت فقط یکی از تراکنش های Tx2 Tx3 Tx4 اجرا میشود. همانطور که در بالا توضیح دادیم، قفل زمانی کاهنده تراکنش ها تضمین میکند که همیشه جدید ترین تراکنش اجرا شود.




شکل ۲-۶: بستن کانال پرداخت

شکل ۲-۶ نشان میدهد که چطور تراکنش Tx4 با خرج کردن تک خروجی Tx1، کانال پرداخت را میبندد. مجددا دقت کنید که تراکنش Tx4 در ابتدا برون بلاکچینی بود ولی در نهایت چون آخرین تراکنش بود روی بلاکچین قرار گرفت و درون بلاکچینی شد اما سایر تراکنش ها یعنی Tx2 و Tx3 برون

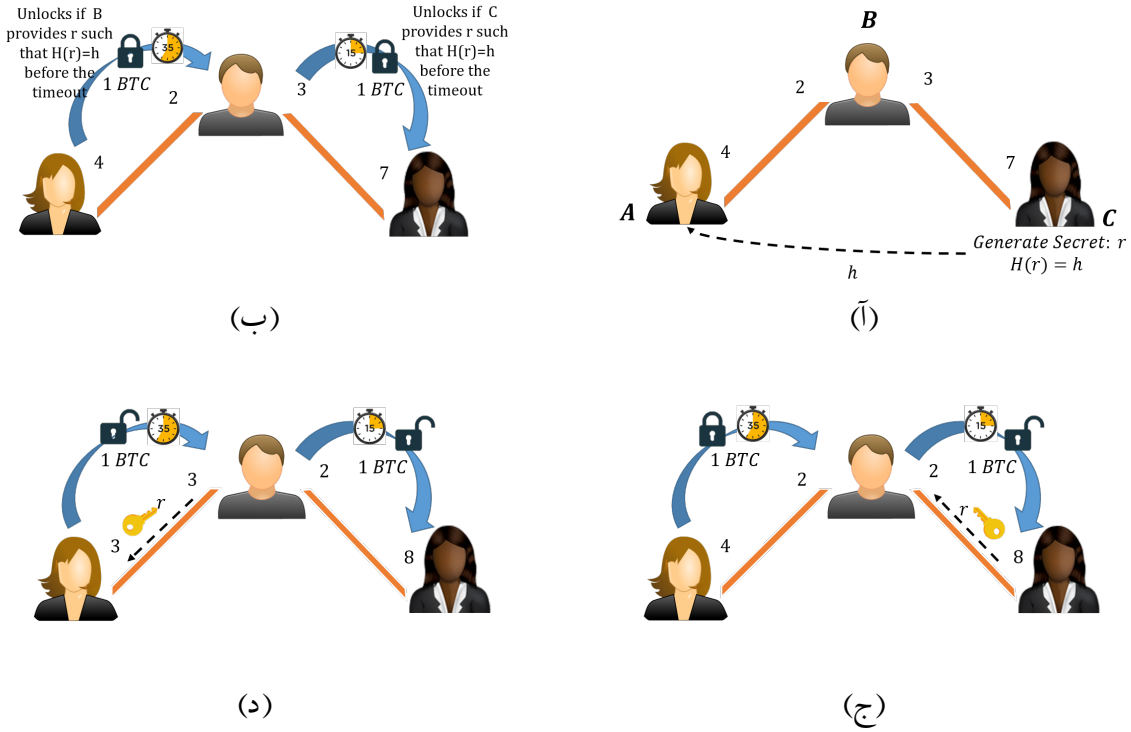
بلاکچینی هستند و پیش از بستن کانال میتوان تا زمانی که موجودی طرفین اجازه میدهد بیشماره تراکنش برون بلاکچینی داشت.

## ۲-۳ تراکنش های با واسطه

همانطور که در قسمت قبل توضیح دادیم، با ایجاد یک کانال پرداخت دو کاربر A و B میتوانند تنها با دو تراکنش درون بلاکچینی تعداد دلخواهی تراکنش برون بلاکچینی برای هم بفرستند اما همچنان محدودیت این روش این است که کاربران دو به دو باید کانال پرداخت ایجاد کنند. برای حل این مشکل توسعه دهندگان Network Lightning پروتکلی طراحی کرده اند که این امکان را به کاربران میدهد که با یک یا چندین واسطه تراکنش برای هم بفرستند [۱]. تراکنش با واسطه را با یک مثال توضیح میدهیم. به شکل ۲-۷ توجه کنید، کاربر A و B با هم و کاربر B و C با هم کانال دارند و کاربر A میخواهد با استفاده از B به عنوان واسطه برای C ۱ بیتکوین بفرستد. در شکل ۲-۷آ که مرحله اول را نشان میدهد میتوانید موجودی هر فرد در کانال ها پیش از انجام تراکنش را ببینید. در این مرحله C یک مقدار تصادفی و محرمانه $r$ را انتخاب کرده و آن را از یک تابع چکیده ساز[۷]، که در اینجا با $H(.)$ نمایش داده شده است عبور میدهد تا $h$ حاصل شود و بعد $h$ را برای A ارسال میکند. در اینجا لازم است توضیح مختصری درباره توابع چکیده ساز بدهیم. این توابع یک ورودی از طول دلخواه را گرفته و به خروجی به طول معین تبدیل میکنند. در حالیکه محاسبه این توابع از نظر محاسباتی ساده است اما محاسبه وارون آن ها دشوار است بنابراین با داشتن $h$ محاسبه $r$ ممکن نیست.

در مرحله دو که در شکل ۲-۷ب نمایش داده شده، A یک تراکنش ۱ بیتکوینی برای B میفرستد اما شرط اینکه این تراکنش اجرا شود این است که B بتواند پیش از زمان معین timeout $r$ را طوری پیدا کند که $H(r) = h$ سپس B تراکنش مشابهی را منتها با timeout کوتاه تر تولید کرده و برای C میفرستد. تنها کسی که $r$ را میداند C است پس پیش از به پایان رسیدن C timeout، $r$ را برای B فرستاده (شکل ۲-۷ج) و قفل تراکنش باز میشود و ۱ بیتکوین به صورت برون بلاکچینی از B به C واریز میشود. سپس چون timeout تراکنش A-B بیشتر از timeout تراکنش B-C است B زمان کافی خواهد داشت که $r$ را برای A بفرستد و ۱ بیتکوین خود را از او بگیرد (شکل ۲-۷د) و به این ترتیب B همان پولی را که در کانال با C از دست میدهد در کانال با A بدست می آورد. اگر زمان timeout هر کدام از تراکنش ها

[۷] Hash function




(ب) (آ)

(د) (ج)

شکل ۲-۷: مراحل انجام یک تراکنش با واسطه در شبکه کانال های پرداخت

بگذرد آن تراکنش کلا برگشت میخورد پس اگر C آفلاین شود و نتواند $r$ را تا پیش از timeout نفرستد، تراکنش B-C برگشت میخورد و بالطبع چون B هم به هیچ وجه نمیتواند $r$ را پیدا کند، تراکنش A-B هم برگشت میخورد به همین دلیل میگوییم این تراکنش ها atomic هستند یعنی یا هر دو با هم انجام میشوند یا هر دو برگشت میخورند.

## ۲-۴ Lightning Network

Lightning Network که شبکه کانال های پرداخت پیاده سازی شده روی بیتکوین است امروزه بیش از ۱۷۰۰۰ کاربر دارد [۴]. این شبکه از به هم پیوستن کاربران توسط کانال های پرداخت به روشی که در بالا توضیح داده شد تشکیل شده است. این شبکه توپولوژی متمرکزی دارد به این معنی که تعداد بسیار کمی نود نقش واسطه را در بیشتر تراکنش ها بازی میکنند. به طور مثال با بررسی داده های بهار سال ۲۰۲۱ [۳] به این نتیجه رسیدیم که ۱۰ نود پر ارتباط شبکه به بیش از ۵۰ درصد شبکه سرویس میدهند. نحوه کار این سرویس دهنده ها به این صورت است که کاربران برای ورود به شبکه کانال های

پرداخت تنها با یکی از این سرویس دهنده ها کانال پرداخت ایجاد میکنند و بعد تمام تراکنش هایشان را با واسطه این سرویس دهنده ارسال میکنند. معمولا این سرویس دهنده با یکدیگر هم کانال های پرداخت متعددی دارند و در نتیجه میتوانند با تعداد واسطه اندک تراکنش های کاربران را به انجام برسانند. این سرویس دهنده ها در ازای جابجایی تراکنش ها کارمزد دریافت میکنند.

## ۲-۵ الگوریتم های آنلاین

[۸] در این قسمت به طور مختصر توضیح میدهیم که منظور از الگوریتم آنلاین چیست و چرا ضرورت دارد که برای حل مساله تصمیم گیری مدیریتی شبکه کانال های پرداخت از الگوریتم آنلاین استفاده کنیم. الگوریتم های آنلاین الگوریتم هایی هستند که فرض خاصی روی توزیع ورودی های مساله ندارند و برای موقعیت هایی قابل استفاده هستند که محیط به قدری سریع تغییر میکند که اصولا فرض روی توزیع متغیر ها فرض قابل قبولی نیست و یا اینکه محیط به اصطلاح متخاصم [۹] است یعنی محیط دنباله ورودی را انتخاب میکند و میتواند ورودی هایی را انتخاب کند که الگوریتم شما در بدترین حالت ممکن خودش عمل کند. پس در این روش تحلیل الگوریتم را طوری طراحی میکنیم که در صورت بدترین ورودی ممکن هم بتوان تضمین کرد الگوریتم خیلی بد عمل نمیکند. مساله و الگوریتم آنلاین را با مثال معروف کرایه اسکی [۱۰] توضیح میدهیم [۵]. فرض کنید شما میخواهید اسکی بازی کنید و تجهیزات ندارید و هزینه اجاره کردن اسکی برای مدت زمان $t$ ماه $t$ تومان است. از طرفی اگر تجهیزات اسکی را بخرید باید ۱ تومان بپردازید اما دیگر نیاز به پرداخت کرایه ندارید. حالا شما در سر این دو راهی هستید که بعد از چه مدت زمانی ($z$) تجهیزات را بخرید. مسلما اگر شما بدانید که تا چه مدت قصد دارید اسکی بازی کنید تصمیم گیری راحت است زیرا اگر بدانید که قصد دارید بیش از یکماه اسکی بازی کنید به صرفه تر است که تجهیزات را همان ابتدا بخرید و در غیر این صورت تجهیزات را کرایه کنید. اما مشکل این است که شما از قبل نمیدانید که چقدر میخواهید اسکی بازی کنید و این زمان ($u$) به صورت متخاصمانه توسط محیط تعیین میشود. هزینه الگوریتمی که در زمان $z$ تجهیزات را میخرد به صورت

[۸] online algorithms
[۹] adversarial
[۱۰] Ski-Rental

زیر تعیین میشود:

$$Cost_z(u) = \begin{cases} u & u \leqslant z \\ z+۱ & u > z \end{cases} \quad (۱-۲)$$

الگوریتم بهینه آفلاین OFF الگوریتمی است که مقدار $u$ را میداند. الگوریتم بهینه در این مثال برای $u > ۱$ تجهیزات را میخرد و در غیر این صورت تجهیزات را اجاره میکند.

$$Cost_{\text{Cost}_{\text{OFF}}}(u) = \begin{cases} u & u \leqslant ۱ \\ ۱ & u > ۱ \end{cases} \quad (۲-۲)$$

**تعریف ۲-۱** *الگوریتم آنلاین* ON *را* `c-competitive` *گوییم اگر به ازای هر ورودی* $I$ *همواره داشته باشیم:*

$$\text{Cost}_{\text{ON}}(I) \leqslant c \cdot \text{Cost}_{\text{OFF}}(I)$$

برای مثال کرایه اسکی، الگوریتم آنلاینی را در نظر بگیرید که تا پایان ماه اول تجهیزات را کرایه میکند و در پایان ماه اول تجهیزات را میخرد. اگر $u \leqslant ۱$ باشد هزینه این الگوریتم برابر هزینه ON است اما اگر $u > ۱$ باشد، هزینه این الگوریتم ۲ است در حالیکه هزینه الگوریتم بهینه ۱ است پس ضریب رقابتی این الگوریتم آنلاین ۲ است یعنی برای بدترین ورودی ممکن هم هزینه این الگوریتم ۲ برابر هزینه بهینه است.

اکنون تمام مقدمات مورد نیاز را مطرح کرده ایم و میتوانیم مستقیما وارد مدل بندی ریاضی مسئله مان شویم.

# فصل ۳

# مدلسازی مسئله

در این قسمت مدلسازی مساله به همراه مدل هزینه برای کانال دو طرفه را بررسی میکنیم. یعنی کانالی که در آن هر دو طرف کانال میتوانند برای هم تراکنش بفرستند. در زیر مساله های اول و دوم مدل هزینه ای تفاوت اندکی با مدل مطرح شده در این بخش دارد که در بخش ؟؟ این تفاوت را توضیح میدهیم.

## ۳-۱ کانال پرداخت

ما شبکه کانال های پرداخت را با یک گراف غیر جهت دار $G = (V, E)$ نمایش میدهیم که در آن راس ها کاربران و یال ها کانال های پرداخت هستند. به طور مثال کانال پرداخت بین دو کاربر $\ell, r \in V$ یک یال در گراف است $(\ell, r) \in E$. ما در این پایان نامه فقط روی یک زوج کاربر $\ell, r$ و کانال بینشان تمرکز میکنیم. پولی که $\ell$ و $r$ در این کانال دارند به ترتیب با $b(\ell)$ و $b(r)$ [۱] نمایش داده میشود و مجموع پولی که در کانال وجود دارد $b(\ell) + b(r)$ را ظرفیت کانال مینامیم. ظرفیت کانال از زمان باز شدن تا بسته نشدن کانال ثابت باقی میماند اما سهم هر کس از ظرفیت کانال یعنی همان $b(\ell)$ و $b(r)$ با ارسال تراکنش تغییر میکند.

[۱] balance

## ۳-۲ تراکنش ها

در این پایان نامه قصد داریم مدیریت تراکنش های برون بلاکچینی کانال $(\ell, r)$ را مدیریت کنیم بنابراین همیشه منظور از تراکنش تراکنش برون بلاکچینی است مگر اینکه صریحتا خلاف آن را ذکر کنیم. دنباله تراکنش $X_t = (x_1, x_2, \ldots, x_t), x_i \in \mathbb{R}$ را در نظر بگیرید که تمام تراکنش های رسیده شده به کانال $(\ell, r)$ از لحظه ۱ تا $t$ را نشان میدهد. تراکنش ها به ترتیب زمان سر میرسند. هر تراکنش با یک عدد حقیقی مثبت و یک جهت (یا معادلا یک علامت) نمایش داده میشود که عدد حقیقی بیانگر اندازه تراکنش است (مثلا ۱ بیتکوین، ۰/۰۵ بیتکوین و ...) و جهت نشان دهنده این است که تراکنش از $\ell$ به $r$ است یا از $r$ به $\ell$. مثلا $\overrightarrow{x}$ (یا $x > 0$) نمایانگر تراکنشی است که از $\ell$ به $r$ میرود. در این پایان نامه از اول حروف `left` و `right` یعنی $\ell$ و $r$ استفاده میکنیم تا جهت راست به چپ یا چپ به راست تراکنش ها هم معنی بصری شهودی تری داشته باشد.

وقتی میگوییم تراکنشی در جهت $\ell$ به $r$ است دو حالت وجود دارد:

۱. یا خود $\ell$ میخواهد برای $r$ پول بفرستد و این تراکنش را ایجاد کرده.

۲. یا یکی از همسایه های $\ell$ میخواهد پولی را به $r$ یا یکی از همسایه های $r$ برساند و در این تراکنش $(\ell, r)$ نقش واسطه را دارد.

در هر دو این حالت ها ما تراکنش را به صورت(یا معادلا $x > 0$) $\overrightarrow{x}$ نمایش میدهیم و برعکس تراکنش از $r$ به $\ell$ را با $\overleftarrow{x}$ (یا معادلا $x < 0$) نمایش میدهیم.

وقتی یک تراکنش جدید فرا میرسد $\ell, r$ میتوانند تصمیم بگیرند آن را انجام دهند یا خیر. همانطور که در بخش ۲-۲ توضیح دادیم، امضا و تایید هر دو عضو یک کانال برای انجام شدن یک تراکنش لازم است پس $\ell$ و $r$ باید هر دو تایید کنند تا تراکنش انجام شود. دنباله $X_t = (x_1, x_2, \ldots, x_t)$ دنباله ی تمام تراکنش های درخواست شده به $(\ell, r)$ را نمایش میدهد فارغ از اینکه این تراکنش ها توسط $r$ و $\ell$ پذیرفته میشوند یا رد میشوند. $r$ و $\ell$ تنها زمانی میتوانند تراکنش $\overrightarrow{x}$ را بپذیرند که $b(\ell) \geqslant x$ باشد و بعد از پذیرش تراکنش توزیع پول ها در کانال به صورت زیر تغییر میکند:

$$b(\ell) \leftarrow b(\ell) - x, b(r) \leftarrow b(r) + x$$

و برای تراکنش $\overleftarrow{x}$ همه چیز دقیقا برعکس است یعنی برای پذیرش این تراکنش باید $b(r) \geqslant x$ باشد و

پس از پذیرش این تراکنش، توزیع پول ها به صورت زیر تغییر میکند:

$$b(\ell) \leftarrow b(\ell) + x, b(r) \leftarrow b(r) - x$$

$r$ و $\ell$ ممکن است به دلایل متعدد تصمیم بگیرند تراکنشی را رد کنند مثلا ممکن است پول کافی برای انجام تراکنش نداشته باشند و یا اینکه تراکنش به حدی بزرگ باشد که در صورت انجام آن موجودی یک طرف کانال صفر یا بسیار کم شود و طرفین تصمیم بگیرند که پذیرش این تراکنش با توجه به موجودی فعلی در کانال معقول نیست. رد کردن تراکنش دو معنی میتواند اینجا داشته باشد. مثلا تراکنش $\overrightarrow{x}$ را در نظر بگیرید. همانطور که پیش از این گفتیم یا این تراکنش را خود $\ell$ ایجاد کرده یا تراکنش یکی از همسایه های $\ell$ است که از او خواسته این تراکنش را انتقال دهد.

۱. اگر تراکنش توسط خود $\ell$ ایجاد شده باشد ولی $r$ و $\ell$ آن را انجام ندهند معنای آن در دنیای واقعی این است که $\ell$ باید پولی برای $r$ بفرستد ولی به دلیل عدم توافق، نمیتواند این پول را به صورت مجانی در کانال پرداخت اش با $r$ بفرستد و نیاز دارد این تراکنش را از طریق یک پلتفرم دیگر مثلا بانک یا تراکنش های درون بلاکچینی روی بلاکچین بفرستد، پس این امر هزینه ای را متحمل $\ell$ خواهد کرد چون انجام تراکنش روی هر پلتفرمی مثل بانک یا بلاکچین یک کارمزد دارد.

۲. اگر تراکنش برای یکی از همسایگان $\ell$ باشد و $r$ و $\ell$ تصمیم بگیرند آن را انتقال ندهند، از کارمزد آن تراکنش دارند صرف نظر میکنند چون همانطور که قبلا گفتیم واسطه ها در شبکه کانال های پرداخت برای انتقال تراکنش ها کارمزد میگیرند. این چشم پوشی از کارمزد را هم میتوان با یک هزینه مدل کرد.

پس در هر دو حالت بالا عدم انجام تراکنش در کانال برای $\ell$ و $r$ هزینه ای در بردارد. برای رد کردن تراکنش به اندازه $x$ هزینه خطی $Rx + f_۲$ را در نظر میگیریم. $R, f_۲ \in \mathbb{R}^+$ اعداد ثابت هستند که در در قسمت ؟؟ توضیح میدهیم که چگونه باید انتخاب شوند.

**نکته مهم:** دقت کنید که در مدل هزینه ای مطرح شده در این قسمت که مدل هزینه ای کانال دو طرفه است فرض را بر این میگیریم که طرفین کانال با هم همکاری دارند و میخواهند مجموع هزینه هایشان را به کمک هم کمینه کنند.

## ۳-۳ شارژ کردن کانال روی بلاکچین

[۲] گاهی انجام تراکنش های متعدد از یک جهت کانال باعث میشود تمام پول کانال دست یک نفر باشد و دیگری پولش صفر شود. مثلا اگر پول $\ell$ صفر شده باشد تا زمانی که تراکنشی از سمت $r$ نیاید، $\ell$ نمیتواند هیچ تراکنشی بفرستد و این امر مطلوب نیست به همین دلیل گاهی طرفین تصمیم میگیرند این کانالی که به اصطلاح نامتعادل [۳] شده است را روی بلاکچین ببندند و کانال جدیدی باز کنند. این کار را شارژ کردن درون بلاکچینی مینامیم. همانطور که میدانیم برای بستن یک کانال پرداخت باید $\ell$ و $r$ یک تراکنش روی بلاکچین بفرستند همچنین برای باز کردن کانال پرداخت جدید هم باید یک تراکنش دیگر بسازند. پس شارژ کردن درون بلاکچینی هزینه ای را متحمل $\ell$ و $r$ میکند که ما این هزینه را با $f_1 \in \mathbb{R}^+$ نشان میدهیم. اما شارژ کردن درون بلاکچینی یک هزینه ضمنی دیگر هم دارد. وقتی دو کاربر $\ell$ و $r$ یک کانال پرداخت ایجاد میکنند و در مجموع مقدار $F$ بیتکوین در آن ذخیره میکنند، این بیتکوین ها تا پیش از بسته شدن کانال از درسترس $\ell$ و $r$ خارج میشوند. اگر این پول در کانال قفل نشده بود $\ell$ و $r$ میتوانستند آن را سرمایه گذاری کنند و با نرخ بهره ای سود به دست آورند اما حالا که در کانال ذخیره شده این سود را از دست میدهند. این فرصت سرمایه گذاری را که با باز کردن کانال جدید از بین میرود را به صورت یک هزینه در نظر میگیریم و آن را با $\beta F$ نشان میدهیم. $F$ پول قفل شده در کانال است و $\beta$ عددی بین ۰ و ۱ است که میزان سوددهی سرمایه گذاری در جهان خارج را نشان میدهد. هر چه $\beta$ بزرگ تر باشد، ذخیره کردن پول در کانال پرداخت به جای سرمایه گذاری کردن ضرر بیشتری برای $\ell$ و $r$ دارد و بالطبع هزینه بیشتری هم میپردازند.

پس در مجموع هزینه شارژ کردن درون بلاکچینی و باز کردن یک کانال جدید با ظرفیت $F$ برابر $\beta F + f_1$ است که در آن $f_1 \in \mathbb{R}^+$ نماینده کارمزد بستن یک کانال و باز کردن کانال جدید بر بلاکچین است. $F$ مجموع پولی است که $\ell$ و $r$ در کانال قرار میدهند و $\beta \in [0, 1]$ نماینده میزان سوددهی یک واحد پول با سرمایه گذاری در دنیای خارج از شبکه کانال های پرداخت است.

[۲] on-chain recharging
[۳] depleted channel

## ۳-۴ متعادل کردن کانال روی شبکه کانال های پرداخت

[۴] همانطور که پیش از این هم مطرح شد هزینه شارژ کردن کانال روی بلاکچین بسیار بالا است بنابراین در عمل بسیار کم از این روش استفاده میشود. در واقعیت معروف ترین نسخه های کاربری Lightning Network بدون استفاده از بلاکچین و بر روی خود شبکه کانال های پرداخت کانال خود را متعادل میکنند. اما چگونه؟ با یک مثال توضیح میدهیم. به حلقه سمت چپ شکل ۳-۴ توجه کنید. میبینید که $\ell$ در کانال $(\ell, r)$ پولی ندارد اما در کانال با $h$ مقدار ۱۵ واحد پول دارد. $\ell$ میخواهد بدون اینکه مجبور شود کانالش با $r$ را ببندد، به موجودیش در کانال $(\ell, r)$ بیفزاید؛ $\ell$ یک تراکنش حلقه ای ایجاد میکند که با واسطه $h$ برای خودش ۵ واحد پول از کانال $(\ell, h)$ به کانال $(\ell, r)$ بریزد. فلش های بنفش رنگ مسیر تراکنش را نشان میدهند. در واقع فرستنده و گیرنده این تراکنش خود $\ell$ است.


شکل ۳-۱: مثالی از متعادل کردن کانال روی شبکه کانال های پرداخت

پس متعادل کردن کانال روی شبکه کانال های پرداخت چیزی نیست جز تراکنشی در یک دور. در این تراکنش با واسطه هم مانند هر تراکنش با واسطه دیگری $h$ که واسطه است از $\ell, r$ کارمزد دریافت میکند. در قسمت ۳-۲ توضیح دادیم که کارمزدی که یک واسطه برای جابجایی یک تراکنش به ارزش $x$ میگیرد یک تابع خطی از $x$ به صورت $Rx + f_۲$ است. پس متعادل کردن روی شبکه کانال های پرداخت در حلقه ای که $C$ واسطه دارد $C(Rx + f_۲)$ است. مثلا در شکل فقط یک واسطه وجود دارد، $h$، پس $C = ۱$ است.

## ۳-۵ تعریف الگوریتم بهینه آفلاین و الگوریتم آنلاین

الگوریتم بهینه آفلاین که آن را با OFF نمایش میدهیم، یک دنباله تراکنش $X_t = (x_۱, x_۲, \ldots, x_t), x_i \in \mathbb{R}^+$ به عنوان ورودی میگیرد و به عنوان خروجی سه دنباله $Reb_t$ و $Y_t$ و $Rech_t$ را تحویل میدهد.

[۴] off-chain rebalancing

$$Rech_t = \{rech_۱, rech_۲, \ldots, rech_t\}, rech_i \in \{\mathbb{R}^{\geqslant ۰}\} \times \{\mathbb{R}^{\geqslant ۰}\}$$

$$Reb_t = \{reb_۱, reb_۲, \ldots, reb_t\}, reb_i \in \mathbb{R}$$

$$Y_t = \{y_۱, y_۲, \ldots, y_t\}, y_i \in \{\texttt{Accept}, \texttt{Reject}\}$$

دنباله $Rech_t$ نشان دهنده این است که در چه زمان هایی الگوریتم OFF شارژ کردن درون بلاکچینی انجام داده است. هر عضو دنباله $Rech_t$ یک زوج مرتب است $rech_i = (rech_i^\ell, rech_i^r)$ که نشان دهنده میزان پول اضافه شده به $\ell$ و $r$ هستند.

دو متغیر جدید $A_\ell(X_i)$ و $A_r(X_i)$ را تعریف میکنیم که به ترتیب نشان دهنده متغیر های $b(\ell), b(r)$ پس از پردازش تراکنش $i$ توسط الگوریتم OFF هستند. در لحظه $i$ پس از انجام شارژ کردن درون بلاکچینی، موجودی طرفین کانال به صورت زیر تغییر میکند:

$$A_\ell(X_i) = A_\ell(X_{i-۱}) + rech_i^\ell$$

$$A_r(X_i) = A_r(X_{i-۱}) + rech_i^r$$

همچنین هزینه الگوریتم OFF به صورت زیر تغییر میکند:

(۱-۳)

$$\text{Cost}_{\text{OFF}}(X_i) = \begin{cases} \text{Cost}_{\text{OFF}}(X_{i-۱}) & \text{If } rech_i^\ell = ۰ \text{ and } rech_i^r = ۰ \\ \text{Cost}_{\text{OFF}}(X_{i-۱}) + \beta(rech_i^\ell + rech_i^r) + f_۱ & \text{If } rech_i^\ell > ۰ \text{ or } rech_i^r > ۰ \end{cases}$$

متغیر $reb_i$ نشان دهنده اندازه و جهت متعادل کردن برون بلاکچینی است. عبارت زیر تغییر موجودی کاربران پس از متعادل کردن برون بلاکچینی در لحظه $i$ را نشان میدهد.

(۲-۳)

$$\text{Cost}_{\text{OFF}}(X_i) = \begin{cases} A_\ell(X_i) = A_\ell(X_i) - reb_i, A_r(X_i) = A_r(X_i) + reb_i & \text{If } ۰ < reb_i \leqslant A_\ell(X_i) \\ A_r(X_i) = A_r(X_i) - |reb_i|, A_\ell(X_i) = A_\ell(X_i) + |reb_i| & \text{If } -A_r(X_i) \leqslant reb_i < ۰ \end{cases}$$

دقت کنید که اگر در سمت چپ نیاز به پول باشد OFF از راست به چپ ($reb_i < ۰$) پول منتقل میکند و برعکس اگر در سمت راست نیاز به پول باشد OFF از چپ به راست ($reb_i > ۰$) پول منتقل میکند و

هیچ گاه از هر دو جهت همزمان پول منتقل نمیکند زیرا این کار فقط باعث افزایش هزینه نالازم میشود. هزینه OFF پس از انجام متعادل کردن برون بلاکچینی به صورت زیر تغییر میکند:

$$\text{Cost}_{\text{OFF}}(X_i) = \begin{cases} \text{Cost}_{\text{OFF}}(X_{i-1}) & \text{If } reb_i = 0 \\ \text{Cost}_{\text{OFF}}(X_{i-1}) + C(R|reb_i| + f_2) & \text{If } reb_i \neq 0 \end{cases} \quad (3\text{-}3)$$

در نهایت متغیر $y_i$ نشان دهنده این است که آیا الگوریتم OFF تراکنش $x_i$ را پذیرفته یا خیر. اعضای دنباله $Y_i$ باید در شرط زیر صدق کنند :

$$\begin{cases} \text{If } x_i > 0 \text{ then } y_i = \texttt{Accept} \quad \text{only If } A_\ell(X_{i-1}) > x_i \\ \text{If } x_i < 0 \text{ then } y_i = \texttt{Accept} \quad \text{only If } A_r(X_{i-1}) > |x_i| \end{cases} \quad (4\text{-}3)$$

شرط بالا به این معنی است که الگوریتم فقط زمانی میتواند تراکنشی را بپذیرد که پول کافی برای آن داشته باشد. هزینه OFF پس از پذیرش یا رد هر تراکنش به صورت زیر تغییر میکند:

$$\text{Cost}_{\text{OFF}}(X_i) = \begin{cases} \text{Cost}_{\text{OFF}}(X_{i-1}) & \text{If } y_i = \texttt{Accept} \\ \text{Cost}_{\text{OFF}}(X_{i-1}) + R|x_i| + f_2 & \text{If } y_i = \texttt{Reject} \end{cases} \quad (5\text{-}3)$$

دقت کنید که ممکن است در لحظه $i$، OFF یک کدام یا هر دو روش شارژ کردن و متعادل کردن را انجام دهد. در این صورت ترتیب آپدیت شدن موجودی کانال یعنی $A_\ell(X_i)$ و $A_r(X_i)$ به ترتیبی که در بالا گفته شد انجام میشود، یعنی اول شارژ کردن درون بلاکچینی موجودی کانال را آپدیت میکند، بعد متعادل کردن برون بلاکچینی و در نهایت پذیرش تراکنش، همچنین هزینه OFF به اندازه مجموع هزینه همه گام های انجام شده افزایش میابد.

نکته مهم دیگری که باید مورد توجه قرار گیرد این است که الگوریتم OFF الگوریتمی با مینیمم هزینه است. یعنی اگر مجموعه $\mathcal{ALG}$ مجموعه کل الگوریتم هایی باشد که با گرفتن دنباله ورودی $X_t$، سه دنباله معتبر خروجی $Reb_t$ و $Y_t$ و $Rech_t$ را تحویل میدهند الگوریتم OFF کمترین هزینه را بین تمام الگوریتم های

$\mathcal{ALG}$ دارد. یعنی :

$$\text{Cost}_{\text{OFF}}(X_i) = \min_{alg \in \mathcal{ALG}} \text{Cost}_{alg}(X_i) \quad \forall X_i$$

همانطور که در بخش ۲-۵ توضیح دادیم الگوریتم آفلاین به کل دنباله ورودی دسترسی دارد و بعد تصمیم گیری میکند. اما الگوریتم آنلاین که آن را با ON نمایش میدهیم، اعضای دنباله تراکنش ها را یک به یک میبیند و بعد از دیدن $x_i$ پیش از دیدن سایر تراکنش ها باید درباره $reb_i$ و $y_i$ و $rech_i$ تصمیم گیری کند. هزینه الگوریتم آنلاین $\text{Cost}_{\text{ON}}(X_i)$ برای پردازش دنباله تراکنش $X_i$ هم با هر تصمیم مشابه الگوریتم آفلاین آپدیت میشود. بدیهی است که همانطور که در مثال Ski-Rental دیدیم هزینه الگوریتم آنلاین معمولا بیشتر از هزینه الگوریتم آفلاین است و ماکسیمم نسبت این دو هزینه را ضریب رقابتی نامیده و با $c$ نمایش میدهیم:

$$\text{Cost}_{\text{ON}}(X_i) \leqslant c \cdot \text{Cost}_{\text{OFF}}(X_i) \quad \forall X_i$$

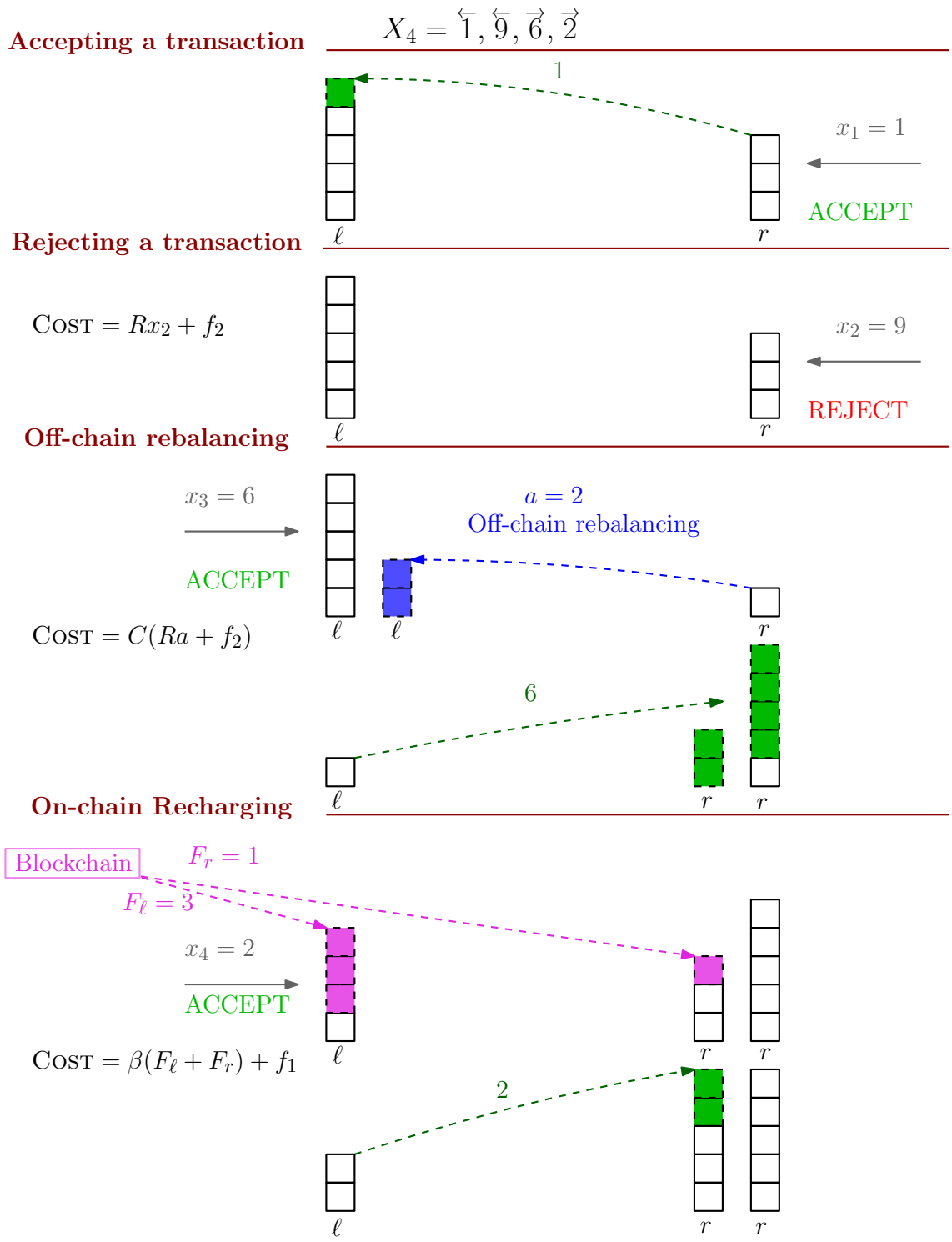
## ۳-۶ مثالی از مسئله کانال های دو طرفه

در این قسمت خلاصه ای از گزینه های موجود برای مدیریت کانال پرداخت و هزینه آن ها ارائه میدهیم که مطالب این فصل را خلاصه و جمع بندی میکند. $\ell, r$ یک کانال پرداخت دارند و در هر لحظه با تراکنشی به اندازه دلخواه و در جهت $\ell$ به $r$ یا $r$ به $\ell$ روبرو هستند. $\ell, r$ باید مشترکا تصمیم بگیرند که کدام تراکنش ها را بپذیرند و کدام ها را رد کنند. رد کردن هر تراکنش به اندازه $x$ هزینه ای برابر $Rx + f_۲$ دارد. برای پذیرش تراکنش به اندازه $x$، فرستنده تراکنش باید حداقل به اندازه $x$ در کانال پول داشته باشد. به شکل ۳-۲ توجه کنید که در آن یک دنباله تراکنش های $\overrightarrow{۲}$ , $\overrightarrow{۶}$ , $\overleftarrow{۹}$ , $\overleftarrow{۱}$ به کانال پرداخت $(\ell, r)$ میرسد. پول اولیه هر دو کاربر $\ell$ و $r$ در کانال ۴ است. با پذیرش تراکنش اول ۱ واحد پول از $r$ به $\ell$ منتقل میشود که با سبز رنگ نمایش داده شده است. تراکنش دوم $\overleftarrow{۹}$ است و در این مثال $\ell$ و $r$ تصمیم میگیرند آن را رد کنند و هزینه $۹R + f_۱$ را متحمل شوند در نتیجه پول $\ell$ و $r$ پس از این تراکنش تغییری نمیکند. تراکنش سوم $\overrightarrow{۶}$ است و $\ell$ و $r$ میخواهند آن را بپذیرند اما مشکل این است که $\ell$ پول کافی در کانال برای پذیرش آن ندارد. پس $\ell$ و $r$ با همکاری هم کانال خود را ابتدا متعادل میکنند تا $\ell$ پول کافی داشته باشد. همانطور که در بخش ۳-۴ توضیح دادیم، در روش متعادل کردن کانال، $\ell$ برای خودش از یکی از کانال های دیگرش به کانال $\ell$ و $r$ پول میفرستد. در مثال شکل ۳-۲ $\ell$ ۲ واحد پول میریز. پس با انجام دادن "متعادل کردن کانال روی شبکه کانال های پرداخت" ابتدا ۲ واحد پول از $r$ به $\ell$ منتقل میشود (۲ بلوک آبی رنگ) و موجودی جدید $\ell$ ۷ و موجودی جدید $r$ ۱ میشود. در این پایان

نامه برای سادگی فرض میکنیم که متعادل کردن کانال سریع انجام میشود پس $\ell$ میتواند پس از متعادل کردن کانال سریعا تراکنش $x_۳$ را بپذیرد. با پذیرفتن این تراکنش ۶ واحد پول (بلوک های سبز رنگ) از $\ell$ به $r$ منتقل میشود. در مجموع هزینه ای که $\ell$ و $r$ برای انجام این تراکنش متحمل شدند همان هزینه متعادل کردن کانال بود که برابر $C(۲R + f_۱)$ است و $C$ تعداد واسطه ها در حلقه متعادل کردن کانال است. در نهایت تراکنش $\overrightarrow{۲}$ میرسد که مجددا $\ell$ و $r$ قصد پذیرش آن را دارند ولی چون $\ell$ پول کافی برای پذیرش آن را ندارند مجبور اند مقداری پول تزریق کنند. در این مثال از روش شارژ کردن کانال روی بلاکچین استفاده میکنند یا به عبارتی این کانال را میبندند و کانال جدیدی باز میکنند که در آن موجودی $\ell$ ۳ واحد بیشتر است و موجودی $r$ ۱ واحد بیشتر از حالت فعلی است. این پول های اضافه شده با بلوک های صورتی رنگ نمایش داده شده اند. هزینه شارژ کردن کانال روی بلاکچین همانطور که در قسمت ۳-۳ توضیح داده شد، برابر است با:

$$\beta(F + f_1), F = F_\ell + F_r$$

پس از شارژ کردن کانال موجودی $\ell$ برای پذیرش تراکنش $\overrightarrow{۲}$ کافی میشود.




شکل ۳-۲: خلاصه کانال پرداخت دو طرفه

# فصل ۴

# کارهای پیشین

## ۴-۱ اهمیت متعادل نگه داشتن کانال های شبکه پرداخت

به صورت سنتی مهم ترین الگوریتم های مسیریابی برای تراکنش ها در Lightning Network همچون Flare [؟]، SilentWhispers [؟] و SpeedyMurmurs [؟]. بیشتر روی افزایش نرخ تراکنش بر ثانیه تمرکز میکنند تا متعادل نگه داشتن کانال ها. اما اخیرا چندین پژوهش جدید در راستای بهبود تعادل کانال ها و اصلاح کانال های نامتعادل انجام شده است. مثلا [؟] برای اولین بار یک استراتژی امن برای متعادل کردن کانال های Lightning Network پیشنهاد داد؛ [؟] ارسال تراکنش ها در چندین مسیر را پیشنهاد میدهد که در حین اینکه نرخ ارسال تراکنش بالا میرود، به متعادل نگه داشتن کانال ها هم کمک میکند. [؟] توابع کارمزد را به نحوی تغییر میدهد که به کاربران انگیزه دهد که کانال ها را بیشتر در مسیری که به متعادل ماندن کانال کمک میکند استفاده کنند؛ و [؟] با پیش بینی تراکنش ها در آینده، سعی میکند برای میزان پولی که باید در لحظه ساخت به کانال اضافه شود تخمین مناسبی بزند. اما در این پایان نامه ما مساله متعادل نگه داشتن کانال ها را از نگاه دیگری بررسی میکنیم. ما هزینه های پردازش تراکنش ها را در نظر میگیریم و الگوریتمی طراحی میکنیم که این هزینه را کمینه کند. از آنجاییکه کانال های نامتعادل معمولا مجبور به رد کردن تعداد زیادی تراکنش میشوند، هزینه این کانال ها بالاست و در نتیجه الگوریتمی که ما طراحی میکنیم به صورت غیرمستقیم از کانال های نامتعادل و یکسو شده اجتناب میکند. تفاوت دیگری که این پایان نامه با سایر پروژه های موجود در این حوزه دارد این است که ما هیچ فرض خاصی روی توزیع دنباله تراکنش های آینده نمیکنیم.

## ۴-۲ متعادل کردن کانال برون بلاکچینی

[۱] متعادل کردن برون بلاکچینی به عنوان جایگزینی ارزان (به جای بستن کانال و باز کردن کانال جدید) برای اضافه کردن پول به یک سمت کانال مورد بررسی تحقیقات گذشته قرار گرفته است. برای نسخه های مختلف Lightning Network همچون c-lightning [۲] و lnd [۳] پلاگین هایی [۴] پیاده سازی شده است که از متعادل کردن برون بلاکچینی پشتیبانی میکند. [؟] یک روش ابتکاری ارائه میدهد که به طور اتوماتیک زمان هایی که متعادل کردن برون بلاکچینی باید انجام شود را تشخیص میدهد. ما هم در این پایان نامه به بررسی زمان هایی که متعادل کردن برون بلاکچینی انجام شود میپردازیم اما این تصمیم را در کنار و وابسته به سایر تصمیمات مدیریتی کانال همچون پذیرفتن یا رد کردن تراکنش ها میگیریم. اخیرا برخی از مقاله ها مانند [؟] و [؟] مساله متعادل کردن را در کل کانال های Lightning Network به صورت یک مساله جهانی[۵] در نظر میگیرند و این مساله را به فرم یک مساله LP در می آورند و جواب بهینه را بدین صورت پیدا میکنند. این دسته از مقالات به طور مستقیم به روشی که ما در این پایان نامه در پی گرفته ایم مرتبط نیستند زیرا ما در اینجا بر یک تک کانال متمرکز میشویم نه بر کل شبکه کانال های پرداخت.

## ۴-۳ استفاده از الگوریتم های آنلاین برای حل مسائل شبکه کانال های پرداخت

[؟] و [؟] نزدیک ترین کارها به این پایان نامه هستند زیرا آن ها هم از الگوریتم های آنلاین (آنالیز بدترین حالت) برای تصمیم گیری درباره پذیرش تراکنش ها در شبکه کانال های پرداخت استفاده میکنند. [؟] مسئله پذیرش یا عدم پذیرش تراکنش ها در یک تک کانال را در نظر میگیرد اما بر خلاف ما، امکان شارژ کردن درون بلاکچینی یا متعادل کردن برون بلاکچینی را به کاربران نمیدهد. [؟] اثبات میکند که برای مدلی که در نظر گرفته الگوریتم آنلاینی با ضریب رقابتی محدود نمیتوان پیدا کرد.

[؟] مسئله زمان بندی شارژ کردن درون بلاکچینی را در نظر میگیرد اما اجازه ی رد کردن تراکنش

[۱] rebalancing Off-chain
[۲] https://github.com/lightningd/plugins/tree/master/rebalance
[۳] https://github.com/bitromortac/lndmanage
[۴] plugin
[۵] global

یا انجام متعادل کردن برون بلاکچینی را به کاربران نمیدهد.

## ۴-۴ ارتباط مسئله ما با مسائل مشابه در شبکه های مخابراتی

مسائل کنترل پذیرش [۶] به طور مثال online call admission سالهاست که در شبکه های مخابراتی مورد بررسی قرار گرفته است [؟، ؟]. اما مسائلی که در شبکه های مخابراتی بررسی میشود دو تفاوت عمده با مسئله کنترل پذیرش در شبکه کانال های پرداخت دارد:

۱) در شبکه های مخابراتی، ظرفیت یک لینک در یک جهت مستقل از جریان اطلاعاتی که از سمت دیگر لینک وارد میشود است در حالیکه در مسئله شبکه کانال های پرداخت، آمدن تراکنش ها از سمت دیگر کانال باعث افزایش موجودی سمت پذیرنده میشود و پذیرنده میتواند بعدا از این پول برای پذیرش تراکنش ها استفاده کند.

۲) انتقال اطلاعات در یک لینک باعث کاهش ظرفیت لینک در آینده نمیشود و فقط از ظرفیت موجود لحظه ای لینک میکاهد، در صورتی که در شبکه کانال های پرداخت پذیرفتن یک تراکنش از یک سمت به سمت دیگر کانال باعث جابجایی همیشگی پول میشود.

این دو تفاوت عمده باعث میشود که مسئله ی کنترل پذیرش در شبکه کانال های پرداخت از نظر الگوریتمی مسئله ای کاملا متفاوت با پذیرش در شبکه های مخابراتی باشد.

[۶] Control Admission

# فصل ۵

# نتایج تئوری

در این فصل ابتدا دو زیرالگوریتم پرکاربرد که از آنها در طراحی الگوریتم اصلی مان استفاده میکنیم را معرفی میکنیم و سپس به سراغ توصیف و پیدا کردن ضریب رقابتی الگوریتم آنلاینمان برای حل مساله کانال دو طرفه میرویم.

در تمام نتایج تئوری این پایان نامه فرض میکنیم که $\beta$ که در تابع هزینه شارژ کردن درون بلاکچینی کانال نقش داشت برابر ۱ است. این فرض به هیچ وجه محدود کننده نیست زیرا هدف ما در این پایان نامه پیدا کردن ضریب رقابتی است که نسبت بین هزینه ON و OFF است. برای هر دنباله تراکنش $X_t$ میتوان نوشت:

$$c = \frac{\text{Cost}_{\text{ON}}(X_t)}{\text{Cost}_{\text{OFF}}(X_t)} = \frac{\frac{\text{Cost}_{\text{ON}}{}^{(X_t)}}{\beta}}{\frac{\text{Cost}_{\text{OFF}}{}^{(X_t)}}{\beta}}$$

پس میتوان هر دو تابع هزینه را تقسیم بر $\beta$ کرد بدون اینکه ضریب رقابتی تغییر کند. در نتیجه تابع هزینه شارژ کردن درون بلاکچینی به صورت $F + \frac{f_1}{\beta}$ در می آید. سایر توابع هزینه هم به همین صورت تقسیم بر $\beta$ میشود. در این صورت باید یک نسخه جدید از تمام پارامتر های توابع هزینه مسئله ارائه دهیم:

$$f_1' := \frac{f_1}{\beta}, f_2' := \frac{f_2}{\beta}, R' := \frac{R}{\beta}$$

. از آنجا که ضریب رقابتی الگوریتم اصلی ما وابسته به هیچ کدام از پارامتر های بالا نیست، برای سادگی در تمام این پایان نامه از همان نام های $f_1, f_2, R$ استفاده میکنیم.

## ۵-۱ دو زیر الگوریتم پر کاربرد

**دنبال کردن موجودی الگوریتم OFF** الگوریتم بهینه آفلاین OFF را در بخش ۵-۳ معرفی کردیم. پیدا کردن OFF در حالت کلی مساله کانال دو طرفه NP-hard است اما مجموع پولی که الگوریتم OFF پس از پردازش دنباله تراکنش $X_t$ در کانال دارد یعنی $A(X_t) := A_\ell(X_i) + A_r(X_i)$ را میتوان به صورت تقریبی با ضریب ثابت از مقدار واقعی محاسبه کرد، میتوانید به مقاله [؟] برای اطلاعات بیشتر درباره چگونگی محاسبه این تقریب مراجعه کنید.

در این پایان نامه، الگوریتم آنلاینی که طراحی میکنیم مبنی بر استفاده از تقریب $A(X_t)$ در هر لحظه است. یعنی به طور خاص فرض میکنیم اوراکلی[۱] وجود دارد که در هر لحظه $t$ میتوان تابع $\text{Funds}(X_t)$ اراکل را فراخوانی کرد و خروجی $A(X_t)$ را گرفت.

به طور خاص الگوریتم انلاینی که در این پایان نامه طراحی میکنیم پس از دیدن تراکنش $x_i$، پیش از هر تصمیمی ابتدا تابع $A(X_i) \leftarrow \textsc{Funds}(\{x_1, x_2, \ldots, x_i\})$ فراخوانی میکند که خروجی $A(X_i)$ را میدهد و بعد با داشتن این اطلاعات درباره اینکه آیا باید با انجام دادن شارژ کردن درون بلاکچینی به کانال پول اضافه کند یا نه تصمیم میگیرد. این نحوه استفاده از اطلاعات تقریبی درباره الگوریتم OFF در طراحی الگوریتم ON روش بسیار مرسومی است و بسیاری از کارهای پیشین هم روشی مشابه همین روش را پیش گرفته اند [؟].

**تصمیم گیری درباره شارژ کردن کانال روی بلاکچین با تقلید از الگوریتم OFF** الگوریتم ۱ چگونگی تصمیم گیری درباره شارژ کردن کانال توسط ON با داشتن دسترسی به تابع $\text{Funds}(X_t)$ را نشان میدهد. این الگوریتم متغیری به نام $F_{tracker}$ را ذخیره میکند. هر بار که تراکنش جدید فرا میرسد ON $F_{tracker}$ را با خروجی تابع $\text{Funds}(X_t)$ یعنی $A(X_t)$ مقایسه میکند. اگر $F_{tracker}$ کوچک تر بود آن گاه ON پول موجود در کانال را افزایش میدهد طوری که مجموع پول کانال به $\gamma(\text{Funds}(X_t) + \delta)$ برسد. (درباره اینکه چه بخشی از این پول را به $\ell$ و چه بخشی را به $r$ اختصاص میدهد در بخش های بعدی صحبت میکنیم. ) همچنین الگوریتم مقدار $F_{tracker}$ را به مقدار جدید $\text{Funds}(X_t) + \delta$ آپدیت میکند و دوباره همین روند را ادامه میدهد. این الگوریتم را $(\delta, \gamma)$-recharging مینامیم و در طراحی الگوریتم اصلی از آن استفاده میکنیم.

[۱] Oracle

**Algorithm 1:** $(\gamma, \delta)$-recharging

**Initialise:** $F_{tracker}, X \leftarrow 0, \emptyset$

**for** *transaction* $x$ *in order of arrival* **do**
  concatenate $x$ to $X$
  $F'_{tracker} \leftarrow \textsc{Funds}(X)$
  **if** $F'_{tracker} > F_{tracker}$ **then**
    $F_{tracker} \leftarrow F'_{tracker} + \delta$
    recharge to $\gamma F_{tracker}$
**end**

دقت کنید که $\delta, \gamma$ پارامتر های این الگوریتم هستند که توسط ON تعیین میشوند. هرچه $\delta, \gamma$ بزرگ تر باشند، الگوریتم ON به ازای هر افزایش در $A(X_t)$ بیشتر به کانال پول اضافه میکند. همچنین $\delta$ تعیین میکند که عمل شارژ کردن کانال چقدر زود به زود انجام شود. اگر $\delta$ برابر ۰ باشد آن گاه با کوچک ترین تغییر در $A(X_t)$، شاخه `if` الگوریتم ۱ فعال میشود و عمل شارژ کردن انجام میگیرد (که طبعا هزینه هم در پی دارد). از طرفی اگر $\delta$ خیلی بزرگ باشد، در اولین شارژ کردن، ON مقدار زیادی پول به کانال اضافه میکند ولی مدت زمان خیلی زیادی طول میکشد تا $A(X_t)$ به $F_{tracker}$ برسد و دوباره شاخه `if` فعال شود.

یک حالت خاص الگوریتم $(\delta, \gamma)$-`recharging` زمانی رخ میدهد که $\delta = ۰$ و $\gamma = ۱$. در این صورت الگوریتم دقیقا میزان $A(X_t)$ را دنبال میکند و $F_{tracker}$ برابر مجموع پول موجود در کانال است. هرگاه پول موجود در کانال از $A(X_t)$ کمتر شود آنگاه الگوریتم موجودیش را افزایش میدهد تا آن را به $A(X_t)$ برساند. جدول مثالی از ۵-۱ میزان تغییر $F_{tracker}$ و پول ON را با تغییر $A(X_i)$ نشان میدهد.

**لم ۵-۱** *الگوریتم ۱ تضمین میکند که همیشه ON حداقل $\gamma$ برابر OFF پول دارد و همچنین مجموع هزینه شارژ کردن درون بلاکچینی ON تا لحظه $t$ حداکثر برابر*

| ON for channel the in locked Amount | $F_{tracker}$ | $A(X_i)$ |
|---|---|---|
| 0 | 0 | 0 |
| $\gamma(\delta+\varepsilon)$ | $\delta+\varepsilon$ | $\varepsilon$ |
| $\gamma(\delta+\varepsilon)$ | $\delta+\varepsilon$ | $\delta$ |
| $2\gamma(\delta+\varepsilon)$ | $2(\delta+\varepsilon)$ | $\delta+2\varepsilon$ |

Table 5-1: An exampe of $(\gamma,\delta)$- recharging $(\delta>\varepsilon>0)$

$\gamma(A_t+\delta)+f_1\cdot\lceil\frac{A_t}{\delta}\rceil$ *است.*

*اثبات.* اگر $i$ یکی از لحظات شارژ کردن الگوریتم ۱ باشد، موجودی کانال ON در این لحظه برابر است با $\gamma(F_{tracker})$ و $F_{tracker}=A(X_i)+\delta$ از طرفی به محض اینکه در لحظه $j>i$، $F_{tracker}<A(X_j)$ رخ دهد دوباره عمل شارژ شدن انجام میگیرد. پس در تمام لحظات $i\leqslant k<j$ ، پول موجود در کانال بیشتر از $\gamma$ برابر $A(X_k)$ است پس بخش اول قضیه درست است.

برای به دست آوردن هزینه ای که ON برای شارژ کردن کانال تا لحظه $t$ متحمل میشود، اولا دقت کنید در این لحظه ON ماکسیمم $\gamma(A(X_t)+\delta)$ پول در کانال دارد (با فرض اینکه دقیقا در بدترین حالت در لحظه $t$ عمل شارژ کردن را انجام داده). پس $F_{total}(t)$ که مجموع پولی است که ON با شارژ کردن درون بلاکچینی تا لحظه $t$ به کانال خود اضافه کرده در رابطه زیر صدق میکند:

$$F_{total}(t)\leqslant\gamma(A(X_t)+\delta)$$

حالا باید ببینیم حداکثر چندبار عمل شارژ کردن انجام شده تا بتوانیم مجموع هزینه ثابتی (یعنی $f_1$ )که ON برای شارژ کردن ها پرداخته را محاسبه کنیم. اگر برای هر $i<t$ داشته باشیم $A(X_{i+1})=A(X_i)+\delta$ آن گاه برای همه تراکنش ها شاخه `if` الگوریتم ۱ فعال میشود و ON در این صورت باید به ازای هر تراکنش شارژ کردن را انجام دهد و $f_1$ را بپردازد که در این صورت بیشترین هزینه ثابت را متحمل میشود. . این اتفاق حداکثر $\lceil\frac{A(X_t)}{\delta}\rceil$ تعداد بار رخ میدهد و در نتیجه هزینه ثابت ON حداکثر $f_1\lceil\frac{A(X_t)}{\delta}\rceil$ خواهد بود. در نتیجه هزینه کلی ON کمتر یا برابر خواهد بود با :

$$F_{total}(t)+f_1\lceil\frac{A(X_t)}{\delta}\rceil\leqslant\gamma(A(X_t)+\delta)+f_1\lceil\frac{A(X_t)}{\delta}\rceil$$

□

**لم ۵-۲** *اگر* $A(X_t) > ۰$ *آن گاه برای هزینه الگوریتم* OFF *داریم:* $\mathrm{Cost}_{\mathrm{OFF}}(X_t) \geqslant A(X_t) + f_۱$.

*اثبات.* اولا نشان میدهیم که هزینه OFF با گذر زمان یا تغییر نمیکند یا بیشتر میشود یعنی دنباله $\mathrm{Cost}_{\mathrm{OFF}}(X_t)$ یک دنباله صعودی در زمان است، به طور خاص نشان میدهیم اگر اضافه شدن $x_{t+۱}$ به دنباله $X_t$ باعث شود که $\mathrm{Cost}_{\mathrm{OFF}}(X_{t+۱}) < \mathrm{Cost}_{\mathrm{OFF}}(X_t)$ این امر با بهینه بودن الگوریتم OFF تناقض دارد: دقت کنید که $\mathrm{Cost}_{\mathrm{OFF}}(X_{t+۱})$ شامل هزینه تراکنش های $x_۱, \ldots, x_t$ به اضافه هزینه $x_{t+۱}$ است. هزینه پردازش $x_{t+۱}$ همواره بزرگ تر مساوی صفر است (صفر در حالت داشتن موجودی کافی و پذیرش، در غیر این صورت اکیدا بزرگ تر از صفر). پس نزولی بودن $\mathrm{Cost}_{\mathrm{OFF}}(X_t)$ نشان دهنده این است که الگوریتم در لحظه $t+۱$ هزینه اکیدا کمتری برای پردازش تراکنش های $x_۱, \ldots, x_t$ میپردازد که این با بهینه بودن OFF در تمام لحظات تناقض دارد. با داشتن این نکته در ذهن حالا میتوان تحلیل کرد که اگر $A(X_t) > ۰$ یعنی OFF حداقل یکبار کانال را در لحظه ی $t. < t$ به اندازه $A(X_t)$ شارژ کرده پس داریم:

$$\mathrm{Cost}_{\mathrm{OFF}}(X_t) \geqslant \mathrm{Cost}_{\mathrm{OFF}}(X_{t.}) \geqslant A(X_t) + f_۱$$

□

## ۵-۲ توصیف الگوریتم مساله کانال دو طرفه

در این قسمت کلی ترین حالت مساله را در نظر میگیریم که در آن دو کاربر $\ell, r$ با همکاری هم میخواهند مجموع هزینه های کانالشان را کمینه کنند. به دنبال الگوریتم آنلاین ONهستیم که در هر لحظه $i$ با مشاهده تراکنش $x_i$ و دانستن موجودی کانال در این لحظه ابتدا تابع $\mathrm{Funds}(X_i)$ فراخوانی میکند و سپس $y_i, rech_i, reb_i$ را به عنوان خروجی تحویل میدهد. اول از همه دقت کنید که در این بخش فرض میکنیم که $R = ۰$ است یعنی هزینه رد کردن تراکنش یک هزینه ثابت $f_۲$ و هزینه متعادل کردن برون بلاکچینی هم برابر مقدار ثابت $Cf_۲$ است. در بخش ؟؟ میبینیم که با توجه به مقدار بسیار کوچک $R$ در شبکه های واقعی، صفر فرض کردن این مقدار از واقع بینانه بودن مدل چندان نمیکاهد اما در عوض به دستیابی به نتایج تئوری کمک چشمگیری میکند.

الگوریتم ۴ الگوریتم اصلی ما را نشان میدهد. این الگوریتم از ۳ زیر الگوریتم استفاده میکند، اولا در دل خود الگوریتم ۴ الگوریتم $(\delta, \gamma)$-recharging با مقادیر $\delta = f_۱, \gamma = ۴ + ۲\lceil \log C \rceil$ اجرا

میشود. ثانیا در الگوریتم ۴ دو تابع DECIDE(.) و HANDLEFUNDS(.) فراخوانی میشود که توصیف این توابع را به ترتیب در الگوریتم های ۲ و ۳ میتوانید مشاهده کنید.

**Algorithm 2:** Decision on transaction

DECIDE$(F_{tracker}, x, B^{sdr}, B^{rcv})$

$Status \leftarrow$ `Accept`

**if** $\frac{F_{tracker}}{2^i} < x \leqslant \frac{F_{tracker}}{2^{i-1}}$ *and* $x \leqslant B_i^{sdr}$ **then**

Accept $x$

$X \leftarrow \min(F_{tracker}, B_i^{sdr} - x + B_o^{sdr})$

$B_o^{sdr} \leftarrow \max(0, B_i^{sdr} - x + B_o^{sdr} - F_{tracker})$

$B_i^{sdr} \leftarrow X$

**else if** $x_i \leqslant \frac{F_{tracker}}{C}$ *and* $x \leqslant B_s^{sdr}$ **then**

Accept $x$

$X \leftarrow \min(2F_{tracker}, B_s^{sdr} - x + B_o^{sdr})$

$B_o^{sdr} \leftarrow \max(0, B_s^{sdr} - x + B_o^{sdr} - 2F_{tracker})$

$B_s^{sdr} \leftarrow X$

**else if** $x_i \leqslant \frac{F_{tracker}}{C}$ *and* $x > B_s^{sdr}$ **then**

Do off-chain rebalancing to fill $B_s$ and pay $f_2 C$.

$B_o^{rcv} \leftarrow B_o^{rcv} - (2F_{tracker} - B_s^{sdr})$.

$B_s^{rcv} \leftarrow B_s^{rcv} - x$.

Accept $x$

$B_s^{sdr} \leftarrow 2F_{tracker}$.

**else**

Reject x

$Status \leftarrow$ `Reject`

**return** $(B^{sdr}, B^{rcv}, Status)$

الگوریتم ۴ را خط به خط توضیح میدهیم. در خط ۴، الگوریتم ON ، یک تراکنش جدید دریافت کرده و آن را به دنباله تراکنش هایی که تاکنون دیده است اضافه میکند. سپس تابع Funds$(X)$ را فراخوانی میکند و با منطقی دقیقا مشابه با الگوریتم ۱ ، اگر Funds$(X) < F_{tracker}$ آن گاه $F_{tracker}$

**Algorithm 3:** Handling funds coming from the other side

HANDLEFUNDS$(F_{tracker}, x, B)$
  $X \leftarrow \min(2F_{tracker}, B_s + x)$
  $x \leftarrow \max(x + B_s - 2F_{tracker}, 0)$
  $B_s \leftarrow X$
  **for** $i \in [\lceil \log C \rceil]$ *in decreasing order* **do**
    **if** $x > 0$ **then**
      $X \leftarrow \min(F_{tracker}, B_i + x)$
      $x \leftarrow \max(x + B_i - F_{tracker}, 0)$
      $B_i \leftarrow X$
  **end**
  $B_o \leftarrow B_o + x$ **return** $(B)$
**return**

را آپدیت میکند: $F_{tracker} \leftarrow \text{Funds}(X) + f_1$ و موجودی کانال را به اندازه ای افزایش میدهد که در کانال در مجموع $F_{tracker}(۲\lceil \log C \rceil + ۴)$ پول باشد. در اینجا لازم است در خط ۴ الگوریتم توقفی داشته باشیم و توضیح دهیم که ON چگونه پول جدید را بین $\ell$ و $r$ تقسیم میکند. ON در هر سمت کانال $۲ + \lceil \log C \rceil$ سبد بودجه در نظر میگیرد. در این پایان نامه از کلمه سبد بودجه به معنی یک ظرفی با ظرفیت تعیین شده (متغیر با زمان) استفاده میکنیم که در آن میتوان تا سقف ظرفیتش پول ریخت و این پول برای پذیرفتن تراکنش هایی گه در یک رنج خاص هستند استفاده میشود. سبد بودجه های سمت راست کانال را با $B_s^r, B_1^r, B_2^r, \ldots, B_{\lceil \log C \rceil}^r, B_o^r$ و مشابها سبد بودجه های سمت چپ کانال را با $B_s^\ell, B_1^\ell, B_2^\ell, \ldots, B_{\lceil \log C \rceil}^\ell, B_o^\ell$ نمایش میدهیم. ظرفیت سبد بودجه های $B_s^\ell, B_s^r$ برابر $۲F_{tracker}$ است، ظرفیت سبد بودجه های $B_1^\ell, B_2^\ell, \ldots, B_{\lceil \log C \rceil}^\ell$ و $B_1^r, B_2^r, \ldots, B_{\lceil \log C \rceil}^r$ برابر $F_{tracker}$ است و سبد بودجه های $B_o^\ell, B_o^r$ ظرفیت مشخصی ندارند و میتوانند به هر اندازه پر شوند.

در خط ۴ الگوریتم ۴ وقتی عمل شارژ کردن انجام میشود همه سبد بودجه های سمت راست و چپ به جز $B_o^\ell, B_o^r$ به اندازه ظرفیتشان پر میشوند.

اگر بخواهیم پیش از دقیق شدن در سایر گام های الگوریتم یک شهود کلی نسبت به دلیل وجودی سبد بودجه ها بدهیم میتوان گفت که سبد بودجه های $B_s^\ell, B_s^r$ بودجه پذیرش تراکنش هایی با اندازه کوچکتر

**Algorithm 4:** Main algorithm

**Initialise:** left side buckets $B^{\ell}$

**Initialise:** right side buckets $B^{r}$

**Initialise:** tracker $F_{tracker}, X \leftarrow 0, \emptyset$

**for** *transaction* $x$ *in order of arrival* **do**

  concatenate $x$ to $X$

  $F'_{tracker} \leftarrow \textsc{Funds}(X)$

  **if** $F'_{tracker} > F_{tracker}$ **then**

    $F_{tracker} \leftarrow F'_{tracker} + f_1$

    recharge to $2(2 + \lceil \log C \rceil) F_{tracker}$

  $sdr, rcv \leftarrow \ell, r$

  **if** $x$ *is from right to left* **then**

    $sdr, rcv \leftarrow r, \ell$

  $B^{sdr}, B^{rcv}, Status \leftarrow \textsc{Decide}(F_{tracker}, x, B^{sdr}, B^{rcv})$

  **if** $Status ==$ `Accept` **then**

    $B^{rcv} \leftarrow \textsc{HandleFunds}(F_{tracker}, x, B^{rcv})$

**end**

$B_i^r, B_i^\ell, i \in \{۱, ۲, \ldots, \lceil \log C \rceil - ۱\}$ در سمت راست و چپ هستند و سبد بودجه های مساوی $\frac{F_{tracker}}{C}$ برای پذیرش تراکنش هایی با اندازه $\frac{F_{tracker}}{۲^{i-۱}} < x \leqslant \frac{F_{tracker}}{۲^i}$ استفاده میشوند. همچنین سبد بودجه های

$B^{\ell}_{\lceil \log C \rceil}, B^{r}_{\lceil \log C \rceil}$ برای پذیرش تراکنش های بازه $\frac{F_{tracker}}{C} < x \leqslant \frac{F_{tracker}}{۲^{\lceil \log C \rceil - ۱}}$ استفاده میشوند. سبد بودجه های $B^{\ell}_o, B^{r}_o$ هم پول های اضافه (سرریز) را در صورت وجود نگه میدارند و بعدا اگر هر کدام از سبد بودجه های دیگر خالی شدند، به آنها پول انتقال میدهند. دقت کنید که سبد بودجه ها یک مفهوم کاملا انتزاعی هستند به این معنی که صرفا ON آنها را در نظر میگیرد تا بتواند بودجه اش را با برنامه ریزی قبلی به تراکنش ها اختصاص دهد. جابجایی پول بین سبد بودجه های یک سمت کانال هیچ هزینه ای ندارد اما جابجایی پول از سبد بودجه های یک سمت کانال به سمت دیگر، تنها با اسال تراکنش به آن سمت یا با متعادل کردن برون بلاکچینی امکان پذیر است که البته در ادامه هنگام توصیف الگوریتم های ۲ و ۳ توضیحات دقیق تری درباره سبد بودجه ها و نحوه استفاده و پر کردن آنها میدهیم.

در خطوط ۴ تا ۴، بسته به جهت تراکنش فرستنده و گیرنده مشخص میشود و بعد در خط ۴ تابع $\text{Decide}(F_{tracker}, x, B^{sdr}, B^{rcv})$ فراخوانی میشود. $B^{sdr}, B^{rcv}$ به ترتیب مجموعه سبد بودجه های فرستنده و گیرنده را نمایش میدهند. از این به بعد به توصیف تابع $\text{Decide}(.)$ میپردازیم. هدف این تابع این است که با گرفتن وضعیت سبد بودجه ها و اندازه تراکنش تصمیم بگیرد که تراکنش را بپذیرد یا خیر و بسته به پذیرش یا عدم پذیرش موجودی سبد بودجه ها را هم بروزرسانی کند. بسته به اندازه تراکنش دو حالت وجود دارد. یا تراکنش در بازه مربوط به یکی از سبد بودجه های $B_i^{sdr} \quad i \in [۱, \lceil \log C \rceil]$ قرار دارد که در این صورت تراکنش تنها در حالتی پذیرفته میشود که سبد بودجه $B_i^{sdr}$ پول کافی برای پذیرش آن داشته باشد. خطوط ؟؟ تا ؟؟ الگوریتم ۲ مربوط به این حالت است. پس از پذیرش تراکنش موجودی $B_i^{sdr}$ به اندازه $x$ کم میشود ولی بلافاصله سبد بودجه $B_o^{sdr}$ بررسی میشود تا اگر پول سرریزی در آن موجود است این پول به $B_i^{sdr}$ منتقل شود و آن را تا سقف ظرفیتش یعنی $F_{tracker}$ پر کند.

خط های ؟؟ تا ؟؟ الگوریتم ۲ به بررسی تراکنش هایی میپردازد که اندازه آنها کمتر از $\frac{F_{tracker}}{C}$ است و موجودی سبد بودجه $B_s^{sdr}$ برای پذیرش $x$ کافیست. در این صورت تراکنش پذیرفته میشود، پول از $B_s^{sdr}$ کم میشود و بلافاصله بررسی میشود تا اگر پول اضافه ای در $B_o^{sdr}$ موجود بود، $B_s^{sdr}$ را تا سرحد ظرفیتش یعنی $۲F_{tracker}$ پر کند. خط های ؟؟ تا ؟؟ الگوریتم ۲ حالتی را بررسی میکند که اندازه تراکنش کوچک تر از $\frac{F_{tracker}}{C}$ است و موجودی $B_s^{sdr}$ برای پذیرش کافی نیست. در این صورت بر خلاف تراکنش های بزرگ تر که رد میشند، ON عمل متعادل کردن درون بلاکچینی را انجام میدهد تا پول کافی در سمت فرستنده داشته باشد و بتواند تراکنش را بپذیرد. در اثبات ضریب رقابتی الگوریتم خواهیم دید که چرا ON تراکنش های کوچک را بی قید و شرط میپذیرد. در واقع شهود کلی این است که ؟؟

## ۵-۳ اثبات ضریب رقابتی مساله کانال دو طرفه

دو شارژ کردن درون بلاکچینی متوالی الگوریتم ON را به طور دلخواه در نظر بگیرید. زمان شارژ کردن اول را با $t_۱$ و زمان شارژ کردن بعدی را با $t_۲$ نشان میدهیم. در بازه زمانی $(t_۱, t_۲)$ مقدار $F_{tracker}$ و در نتیجه ظرفیت سبد بودجه ها ثابت است و با توجه به خط ۴ الگوریتم ۴ میدانیم که مجموع کل پولی که OFF در کانال دارد از $F_{tracker}$ کم تر است.

هزینه الگوریتم های ON و OFF از چندین منبع مختلف ناشی میشود. ۱) شارژ کردن درون بلاکچینی، آن را با $\text{Cost}^{۱}_{\text{OFF}}(X), \text{Cost}^{۱}_{\text{ON}}(X)$ نمایش میدهیم. ۲) رد کردن تراکنش ها، این بخش از هزینه را با $\text{Cost}^{۲}_{\text{OFF}}(X), \text{Cost}^{۲}_{\text{ON}}(X)$ نمایش میدهیم. ۳) متعادل کردن برون بلاکچینی، این بخش هزینه را با

$$\text{Cost}^{۳}_{\text{OFF}}(X), \text{Cost}^{۳}_{\text{ON}}(X)$$

نمایش میدهیم.

**لم ۵-۳** *با اجرای الگوریتم ۴ برای هر لحظه $t$ که در آن $A(X_t) > ۰$ داریم :*

$$\text{Cost}^{۱}_{\text{ON}}(X) \leqslant (۵ + ۲\lceil \log C \rceil)\text{Cost}^{۱}_{\text{OFF}}(X) \qquad (۱-۵)$$

*اثبات.* با توجه به لم ۵-۱ میتوان نوشت:

$$\text{Cost}^{۱}_{\text{ON}}(X_t) \leqslant \gamma(A_t + \delta) + f_{۱} \cdot \lceil \frac{A_t}{\delta} \rceil \leqslant \gamma(A_t + \delta) + f_{۱} \cdot (\frac{A_t}{\delta} + ۱) \qquad (۲-۵)$$

با جایگذاری $\delta = f_{۱}, \gamma = ۴ + ۲\lceil \log C \rceil$ داریم:

$$\text{Cost}^{۱}_{\text{ON}}(X_t) \leqslant (۴+۲\lceil \log C \rceil)\cdot(A_t+f_{۱})+(A_t+f_{۱}) = (۵+۲\lceil \log C \rceil)\cdot(A_t+f_{۱}) \qquad (۳-۵)$$

با استفاده از نتیجه لم ۵-۲ میتوان نوشت:

$$\text{Cost}^{۱}_{\text{ON}}(X_t) \leqslant (۵ + ۲\lceil \log C \rceil) \cdot (A_t + f_{۱}) \leqslant (۵ + ۲\lceil \log C \rceil)\text{Cost}^{۱}_{\text{OFF}}(X_t) \qquad (۴-۵)$$

□

**ادعای ۵-۴** *اگر فرض کنیم که در بازه $t \in [t_{۱}, t_{۲})$ بین هر دو شارژ کردن متوالی الگوریتم ON ، موجودی کانال OFF یعنی $A(X_t)$ برابر متغیر $F_{tracker}$ در الگوریتم ۴ است، این فرض باعث کاهش هزینه محاسبه شده برای OFF و در نتیجه افزایش ضریب رقابتی میشود.*

*اثبات.* همانطور که پیش از این گفتیم در بازه بین هر دو شارژ کردن الگوریتم ON ،موجودی کانال OFF کمتر مساوی $F_{tracker}$ است. از این پس برای سادگی فرض میکنیم $\forall t \in [t_{۱}, t_{۲})$ $A(X_t) = F_{tracker}$ یعنی فرض میکنیم موجودی OFF از همان ابتدای بازه ماکسیمم مقدار خود را دارد تا انتهای بازه. این کار تنها به نفع الگوریتم OFF است و در محاسبه ضریب رقابتی مشکلی ایجاد نمیکند (فقط ضریب رقابتی را بیشتر میکند نه کمتر زیرا بیشترین توانایی ممکن را برای OFF در نظر گرفته ایم). □

**قضیه‌ی ۵-۵** *اگر $t_1$ و $t_2$ دو زمان متوالی شارژ کردن کانال توسط الگوریتم* ON *(توصیف شده در الگوریتم ۴ ) باشد آن گاه:*

$$\begin{aligned}(\text{Cost}^2_{\text{ON}}(X_{t_2}) + \text{Cost}^3_{\text{ON}}(X_{t_2})) - (\text{Cost}^2_{\text{ON}}(X_{t_1}) \\ + \text{Cost}^3_{\text{ON}}(X_{t_1})) \leqslant 2 \cdot \Big((\text{Cost}^2_{\text{OFF}}(X_{t_2}) \\ + \text{Cost}^3_{\text{OFF}}(X_{t_2})) - (\text{Cost}^2_{\text{OFF}}(X_{t_1}) + \text{Cost}^3_{\text{OFF}}(X_{t_1}))\Big)\end{aligned} \tag{۵-۵}$$

*یعنی هزینه ای که* ON *صرف رد کردن تراکنش ها یا متعادل کردن کانال در بازه $[t_1, t_2]$ میکند، حداکثر دو برابر هزینه ای است که* OFF *صرف این کار میکند.*

*اثبات.* زمان بین $t_1$ تا $t_2$ را به چندین epoch راست و epoch چپ تقسیم میکنیم. زمانی که ON در سمت چپ کانال کمتر از $(2 + \lceil \log C \rceil)F_{tracker}$ پول دارد در یک epoch چپ هستیم (یعنی حداقل یکی از سبد بودجه های سمت چپ کمتر از ظرفیتش پول دارد) و برعکس زمانی که ON در سمت راست کانال کمتر از $(2 + \lceil \log C \rceil)F_{tracker}$ پول دارد در یک epoch راست هستیم (یعنی حداقل یکی از سبد بودجه های سمت راست کمتر از ظرفیتش پول دارد). epoch های راست و چپ با هم تداخل ندارند و کافیست یکی از epoch ها را، مثلا یکی از epoch های سمت چپ را در نظر بگیریم و نشان دهیم که هزینه های ON در این epoch در سمت چپ کانال حداکثر دو برابر هزینه های OFF است. دقت کنید که در یک epoch چپ، در سمت راست کانال همه سبد بودجه ها پر هستند پس هر تراکنشی که $x \leqslant F_{tracker}$ در سمت راست بیاید پذیرفته میشود و هزینه ای برای ON در بر ندارد و فقط لازم است هزینه های ON در سمت چپ کانال را بررسی کنیم. همچنین دقت کنید که OFF مجموعا در هر دو سمت کانال حداکثر به اندازه $F_{tracker}$ پول دارد پس تراکنش های بزرگ تر از این مقدار را رد میکند و ON هم طبق الگوریتم آن ها را رد میکند پس میتوان این تراکنش ها را در نظر نگرفت.

پس میخواهیم نشان دهیم در هر epoch دلخواه در به طور مثال در سمت چپ (برای سمت راست هم اثبات مشابه است) هزینه ای که ON برای رد کردن تراکنش ها و متعادل کردن برون بلاکچینی انجام میدهد حداکثر دو برابر هزینه ای است که OFF برای این امور میپردازد. یک epoch سمت چپ زمانی آغاز میشود که تمام سبد بودجه های چپ پر باشند و تراکنشی از سمت چپ بیاید و از یکی از سبد بودجه های چپ مقداری پول برداشته شود و موجودی $B_o^\ell$ برای پر کردن این سبد بودجه کافی نباشد.

هزینه هر دو الگوریتم OFF و ON را روی تراکنش هایی در رنج های مختلف (مربوط به سبد

بودجه های مختلف) مقایسه میکنیم.

**گام اول:** اول تراکنش هایی که کوچک تر مساوی $\frac{F_{tracker}}{C}$ هستند و بودجه آنها از سبد $B_s^\ell$ تامین میشود را بررسی میکنیم. در ابتدای یک epoch چپ $B_s^\ell$ به اندازه $۲F_{tracker}$ پول دارد و هر تراکنشی در بازه $x \leqslant \frac{F_{tracker}}{C}$ را میپذیرد تا زمانی که پولش تمام شود و آن گاه برای تراکنش بعدی ای که در این بازه بیاید عمل متعادل کردن برون بلاکچینی انجام میدهد، هزینه $Cf_۲$ را برای این کار میپردازد و $B_s^\ell$ را به اندازه ای پر میکند که پس از پذیرفتن تراکنش با اندازه $x$ موجودی باقی مانده در $B_s^\ell$ برابر $۲F_{tracker}$ شود.

نکته حاشیه ای : دقت کنید که انجام متعادل کردن برون بلاکچینی به اندازه ذکر شده برای ON همیشه امکان پذیر است زیرا اینکه $B_s^\ell = a < x$ است نشان دهنده این است که $B_o^r \geqslant ۲F_{tracker} - a$ پس ON میتواند به اندازه $۲F_{tracker} - a$ از $B_o^r$ بردارد و به اندازه $x$ از $B_s^r$ بردارد و بعد از پذیرش تراکنش چون مقدار $x$ از چپ به راست منتقل میشود دوباره $B_s^r$ پر خواهد شد.

حالا میخواهیم هزینه OFF برای تراکنش های این بازه را بررسی کنیم. اولا یک نکته ی بسیار مهم را دقت کنید، در سمت راست کانال ON تمام تراکنش ها را دارد میپذیرد و پول این تراکنش ها در سمت چپ کانال طبق الگوریتم ۳ وارد با الویت ترین سبد بودجه یعنی $B_s^\ell$ میشود بنابراین میزان پولی که ON از سمت کانال دریافت میکند و برای پذیرش تراکنش های سبد بودجه $B_s^\ell$ استفاده میکند بیشتر مساوی پولی است که OFF از سمت راست کانال دریافت میکند.

وقتی ON متعادل کردن برون بلاکچینی انجام میدهد مجموع اندازه تراکنش هایی که در بازه $x \leqslant \frac{F_{tracker}}{C}$ آمده است تا قبل از این لحظه حداقل برابر $۲F_{tracker}$ بوده است. حتی با فرض اینکه OFF در ابتدای epoch همه پولش که حداکثر $F_{tracker}$ است در سمت چپ بوده، حداکثر تعدادی تراکنش که مجموع آنها $F_{tracker}$ بوده را توانسته بپذیرد و بقیه تراکنش ها را یا رد کرده است یا با انجام دادن متعادل کردن برون بلاکچینی آن ها یا بخشی از آن ها را پذیرفته است. اگر متعادل کردن برون بلاکچینی انجام داده باشد که در این صورت هزینه OFF هم حداقل $Cf_۲$ بوده و در نتیجه هزینه ON برای این تراکنش ها کمتر مساوی هزینه OFF است. اما اگر OFF متعادل کردن برون بلاکچینی انجام نداده پس حتما تعدادی تراکنش که هر کدام کوچکتر مساوی $\frac{F_{tracker}}{C}$ هستند و جمعشان حداقل $F_{tracker}$ بوده را رد کرده است. تعداد این تراکنش ها حداقل $C = \frac{\frac{F_{tracker}}{F_{tracker}}}{C}$ عدد بوده در نتیجه OFF برای رد کردن آنها حداقل $Cf_۲$ پرداخته. پس در هر شرایطی هزینه OFF بیشتر مساوی هزینه ON بوده است. با منطق مشابه میتوان نتیجه گرفت که از ابتدا تا انتهای epoch ، هزینه OFF برای

پردازش تراکنش های مربوط به سبد بودجه $B_s^\ell$ بیشتر مساوی هزینه ON برای پردازش این تراکنش هاست.

**گام دوم:** حالا هزینه ON و OFF روی تراکنش هایی را بررسی میکنیم که در محدوده $[\frac{F_{tracker}}{C}, \frac{F_{tracker}}{۲^{\lceil \log C \rceil - ۱}}]$ هستند. زمان رسیدن اولین تراکنش این بازه به سمت چپ کانال را $t'$ و زمان رسیدن آخرین تراکنش این بازه را $t''$ مینامیم. موجودی $B_{\lceil \log C \rceil}^\ell$ در ابتدای epoch برابر با $F_{tracker}$ است و ON این پول را برای پذیرش تراکنش هایی که بازه مربوطه هستند خرج میکند تا زمانی که این سبد بودجه خالی شود، پس از آن ON تا زمانی که پول کافی در $B_{\lceil \log C \rceil}^\ell$ برای پذیرش تراکنش های این بازه نداشته باشد، آن ها را رد میکند تا زمانی که از سمت راست پول برسد و به موجودی $B_{\lceil \log C \rceil}^\ell$ اضافه کند. نکته مهم این است که ON هیچگاه برای پذیرش تراکنش هایی که در این بازه هستند متعادل کردن برون بلاکچینی انجام نمیدهد.

فرض ساده کننده این گام: فرض کنید که در تمام زمان بین $[t', t'']$، $B_s^\ell$ کاملا پر است و خالی هم نمیشود (یعنی تلویحا در این مدت تراکنشی برای سبد بودجه $B_s^\ell$ از سمت چپ نمی آید).

با داشتن این فرض ساده کننده میتوانیم سه نتیجه ی خیلی مهم بگیریم:

۱) پولی که ON برای پذیرفتن تراکنش های بازه $[\frac{F_{tracker}}{C}, \frac{F_{tracker}}{۲^{\lceil \log C \rceil - ۱}}]$ در لحظه $t'$ اختصاص داده است حداقل برابر کل پولی است که OFF در این لحظه در سمت چپ کانال دارد. زیرا موجودی $B_{\lceil \log C \rceil}^\ell$ برابر $F_{tracker}$ است که بیشتر از کل پولی است که OFF در هر دو طرف کانال دارد.

۲) پولی که با پذیرفتن تراکنش های سمت راست توسط ON به سبد بودجه $B_{\lceil \log C \rceil}^\ell$ اضافه میشود بیشتر از پولی است که OFF با پذیرفتن تراکنش های سمت راست کانال در سمت چپ بدست می آورد. دلیل این ادعا این است که چون ON تمام تراکنش های سمت راست را میپذیرد و چون $B_s^\ell$ در تمام این مدت کاملا پر است پس تمام پولی که از سمت راست می آید به $B_{\lceil \log C \rceil}^\ell$ اضافه میشود. (برای یادآوری به الگوریتم۳ مراجعه کنید.)

۳) OFF در بازه زمانی $[t', t'']$ به هیچ وجه تعادل کردن برون بلاکچینی انجام نمیدهد. دلیل این امر این است که فرض کردیم در این مدت همه تراکنش ها بزرگ تر از $\frac{F_{tracker}}{C}$ هستند. در واقع OFF هیچ گاه برای پذیرش تراکنش هایی که همه آن ها مقدار بزرگ از $\frac{F_{tracker}}{C}$ دارند متعادل کردن برون بلاکچینی انجام نمیدهد زیرا هزینه متعادل کردن برون بلاکچینی $Cf_۲$ است و با این کار OFF میتواند حداکثر $F_{tracker}$ پول را از یک سمت به سمت دیگر ببرد و در نتیجه کمتر از $C$ تراکنش را در این بازه قبول کند که این نتیجه میدهد که بهتر است که به جای متعادل کردن برون بلاکچینی OFF همه

این تراکنش ها را رد کند و هزینه کمتری متحمل $Cf_۲$ شود. در نتیجه میتوان گفت که OFF هم مشابه ON اگر پولش تمام شود تراکنش های این بازه را رد میکند.

نتیجه نهایی ۳ نکته بالا این است که در مجموع در بازه $[t', t'']$ ON بیشتر مساوی OFF پول برای پذیرش تراکنش های مربوط به سبد بودجه $B^{\ell}_{\lceil \log C \rceil}$ دارد. حالا به این نکته دقت کنید که تراکنش هایی که مربوط به این سبد بودجه هستند در بازه $\frac{F_{tracker}}{C} < x \leqslant \frac{F_{tracker}}{۲^{\lceil \log C \rceil - ۱}}$ قرار دارند، یعنی کوچک ترین تراکنش این بازه نصف بزرگ ترین تراکنش آن است. بدترین دنباله تراکنش ممکن که نسبت تعداد تراکنش های رد شده توسط ON به تعداد تراکنش های رد شده توسط OFF را ماکسیمم میکند از جنس دنباله زیر است:

$$\{\underbrace{\frac{F_{tracker}}{۲^{\lceil \log C \rceil - ۱}}, \frac{F_{tracker}}{۲^{\lceil \log C \rceil - ۱}} \cdots,}_{۲^{\lceil \log C \rceil - ۱}} \underbrace{\frac{F_{tracker}}{C}, \frac{F_{tracker}}{C}, \ldots}_{C}\} \qquad (۶-۵)$$

که با دیدن آن ON $۲^{\lceil \log C \rceil - ۱}$ تراکنش اول را میپذیرد و پولش تمام میشود و مجبور میشود $C$ تراکنش بعدی را رد کند اما OFF دقیقا برعکس عمل میکند. پس برای این دسته از تراکنش ها خواهیم داشت:

$$\frac{\text{Cost}^{۲}_{\text{ON}}(X_{t_۲}) - \text{Cost}^{۲}_{\text{ON}}(X_{t_۱})}{\text{Cost}^{۲}_{\text{OFF}}(X_{t_۲}) - \text{Cost}^{۲}_{\text{OFF}}(X_{t_۱})} \leqslant \frac{C}{۲^{\lceil \log C \rceil - ۱}} \leqslant ۲ \qquad (۷-۵)$$

پس در گام دوم نشان دادیم با داشتن فرض ساده کننده ای که در بالا ذکر شد هزینه رد کردن تراکنش ها برای ON حداکثر دو برابر OFF است و همچنین نشان دادیم با داشتن فرض مذکور OFF در این بازه متعادل کردن برون بلاکچینی انجام نمیدهد پس رابطه ۵-۵ به طور خاص برای تراکنش های سبد بودجه $B^{\ell}_{\lceil \log C \rceil}$ برقرار است.

**گام سوم:** حالا میخواهیم در این گام نشان دهیم چگونه با حذف کردن فرض ساده کننده ی گام ۲ همچنان به نتیجه مطلوب برسیم. به صورت کلی در بازه $[t', t'']$ ممکن است تراکنش هایی برای سبد بودجه $B^{\ell}_{s}$ به سمت چپ کانال برسد و با پذیرفته شدن این تراکنش ها $B^{\ell}_{s}$ سرخالی شود و در نتیجه اگر تراکنشی از سمت راست آمد، $B^{\ell}_{s}$ را با الویت بیشتری نسبت به $B^{\ell}_{\lceil \log C \rceil}$ پر کند.
حذف کردن فرض ساده کننده، حل کردن چالش اول) فرض کنید در لحظه ای در بازه $[t', t'']$، $B^{\ell}_{s}$ به اندازه

$a < F_{tracker}$ سرخالی است، پس اگر تراکنشی از سمت راست بیاید تا سقف $a$ مقدار از آن به $B^{\ell}_{s}$ اضافه میشود. اینکه $B^{\ell}_{s}$ سرخالی است به این معنی است که قبلا تراکنش یا تراکنش هایی کوچکتر

مساوی $\frac{F_{tracker}}{C}$ با مجموع مقدار $a$ به سمت چپ کانال ارسال شده است، اگر OFF این تراکنش ها را پذیرفته باشد پس دربازه $[t', t'']$ به اندازه $a$ از بودجه OFF برای پذیرفتن تراکنش های سبد $B^{\ell}_{\lceil \log C \rceil}$ پول کم شده است و در بهترین حالت OFF به اندازه $F_{tracker} - a$ بودجه برای پذیرش تراکنش های این سبد دارد. با آمدن تراکنش هایی از راست به چپ OFF ممکن است بتواند این مقدار $a$ را برای پذیرفتن تراکنش های بعدی استفاده کند و در این صورت بودجه اش برای پذیرفتن تراکنش های $B^{\ell}_{\lceil \log C \rceil}$ را به $F_{tracker}$ افزایش دهد. اما دقت کنید که الگوریتم ON برای $B^{\ell}_{\lceil \log C \rceil}$ از همان ابتدا مقدار بودجه $F_{tracker}$ اختصاص داده بود پس در مجموع بودجه ای که ON برای پذیرفتن تراکنش های سبد بودجه $B^{\ell}_{\lceil \log C \rceil}$ اختصاص میدهد بیشتر مساوی OFF است.

حالا حالتی را بررسی میکنیم که OFF این تراکنش های کوچکتر مساوی $\frac{F_{tracker}}{C}$ با مجموع مقدار $a$ را رد کرده است. در این صورت بودجه اولیه OFF برای پذیرش تراکنش های سبد بودجه $B^{\ell}_{\lceil \log C \rceil}$ میتواند همانند ON برابر $F_{tracker}$ باشد و بعدا با آمدن تراکنش های راست به چپ ممکن است بودجه OFF تا $F_{tracker} + a$ هم افزایش پیدا کند در حالیکه بودجه ON همان $F_{tracker}$ باقی می ماند و این ممکن است ON را ناچار کندکه تراکنش هایی از سبد $B^{\ell}_{\lceil \log C \rceil}$ با مجموع هزینه $a$ را رد کند در حالیکه OFF آنها را میپذیرد. بگذارید هزینه هایی که ON و OFF برای رد کردن تراکنش های مذبور میپردازند را مقایسه کنیم: OFF تعدادی تراکنش با مجموع مقدار $a$ را از سبد $B^{\ell}_{s}$ حذف کرده است پس هزینه رد کردنش حداقل برابر $f_{\text{۲}} \frac{aC}{F_{tracker}}$ در حالیکه هزینه رد کردن تعدادی تراکنش با مجموع $a$ توسط ON از سبد $B^{\ell}_{\lceil \log C \rceil}$ از این مقدار اکیدا کم تر است (زیرا تراکنش های این سبد بزرگ تر هستند). در نتیجه با احتساب هزینه های رد کردن، بودجه زیاد تر OFF کمکی به کمتر کردن هزینه هایش نمیکند و میتوان بودجه اضافی OFF به مقدار $a$ را در نظر نگرفت (زیرا فقط باعث افزایش هزینه اش در سمت چپ میشود) و فرض کرد هر دو ON و OFF بودجه یکسان $F_{tracker}$ را دارند پس از اثبات گام ۲ میتوان استفاده کرد.

به طور کلی میتوان نتیجه گرفت زمانی که سبد بودجه $B^{\ell}_{j}$ سرخالی است، استفاده از پول تراکنش های راست به چپ برای پذیرش تراکنش هایی که مربوط به سبد های $B^{\ell}_{k},\ k < j$ هستند، تنها باعث افزایش هزینه OFF در سمت چپ کانال میشود.

حذف کردن فرض ساده کننده، قدم دوم) اگر تراکنش های کوچک تر از $\frac{F_{tracker}}{C}$ در بازه زمانی $[t', t'']$ در سمت چپ کانال داشته باشیم، آن گاه انجام دادن متعادل کردن برون بلاکچینی ممکن است برای OFF سودبخش باشد و با انجام این کار بتواند تعدادی تراکنش از دسته $T_s$ بپذیرد که آن ها را با $\zeta_s$

نشان میدهیم و تعدادی تراکنش از دسته $T_{\lceil \log C \rceil}$ که آن ها را با $\zeta_{\lceil \log C \rceil}$ نشان میدهیم. به زبان دیگر OFF میتواند با متعادل کردن برون بلاکچینی بودجه اش برای پذیرش تراکنش های $T_{\lceil \log C \rceil}$ را زیاد کند در حالیکه ON هیچ گاه برای این دسته از تراکنش ها متعادل کردن برون بلاکچینی انجام نمیدهد. باید بررسی کنیم که در این صورت ON آیا این تراکنش ها را میپذیرد یا خیر و اگر نمیپذیرد آیا این امر باعث افزایش هزینه اش میشود یا خیر. نشان میدهیم که انجام متعادل کردن برون بلاکچینی برای پذیرش برخی از تراکنش های $T_{\lceil \log C \rceil}$ همیشه هزینه OFF را به اندازه $Cf_۲$ زیاد میکند اما هزینه ON را اکیدا کمتر از این مقدار اضافه میکند. اگر $B_s^\ell \geqslant F_{tracker}$ آن گاه ON میتواند همه تراکنش های $\zeta_s$ را بدون هیچ هزینه ای بپذیرد و حتی اگر مجبور به رد کردن تراکنش های $\zeta_{\lceil \log C \rceil}$ هم شود باز هم چون این تراکنش ها بزرگ تر $\frac{F_{tracker}}{C}$ هستند و مجموعشان کمتر از $F_{tracker}$ است هزینه رد کردنشان کمتر از $Cf_۲$ خواهد شد. اما اگر $B_s^\ell < \sum_{x \in \zeta_s} x < F_{tracker}$ آن گاه ON متعادل کردن برون بلاکچینی انجام میدهد تا تراکنش های $\zeta_s$ را بپذیرد و هزینه $Cf_۲$ را میپردازد، به علاوه برای رد کردن تراکنش های $\zeta_{\lceil \log C \rceil}$ هم مقداری کمتر از $Cf_۲$ میپردازد. اما نکته ای که باید دقت کرد این است که چون $B_s^\ell < F_{tracker}$ پس پیش از این OFF حداقل یکبار به اندازه $Cf_۲$ هزینه داده است و در مجموع برای پردازش تمام تراکنش های $T_s$ تا این لحظه و تراکنش های $\zeta_{\lceil \log C \rceil}$ حداقل $۲Cf_۲$ هزینه کرده است در حالیکه هزینه ای که ON برای پردازش تراکنش های $T_s$ تا این لحظه و تراکنش $\zeta_{\lceil \log C \rceil}$ پرداخت کرده است کمتر مساوی $۲Cf_۲$ است.

□

# فصل ۶

# پیاده سازی

دستاورد های این فصل به شرح زیر است:
۱) با استفاده از dynamic programming الگوریتمی برای پیدا کردن جواب بهینه آفلاین OFF برای مساله کانال های دو طرفه ارائه میدهیم. توجه کنید که جواب بهینه آفلاین برای این مساله NP-Hard است [؟] و در نتیجه پیاده سازی آن برای دنباله های تراکنش طولانی بهینه نیست، با این وجود در این بخش این الگوریتم را برای دنباله تراکنش های به نسبت کوتاه پیاده سازی میکنیم تا بتوانیم هزینه آن را در عمل با هزینه الگوریتم آنلاینی که در فصل ۵ طراحی کرده ایم مقایسه کنیم.
۲) الگوریتم آنلاین مساله کانال های دو طرفه که ضریب رقابتی آن را در فصل ۵ اثبات کردیم پیاده سازی میکنیم و ضریب رقابتی آن را در عمل پیدا کرده و با نتایج تئوری مقایسه میکنیم.
۳) پارامتر های توابع هزینه مدلمان را برای Lightning Network که یک نمونه ی عملی شده و پرکاربر شبکه کانال های پرداخت است پیدا میکنیم.

## ۶-۱ پیدا کردن الگوریتم OFF به کمک dynamic programming

برای دنباله تراکنش $X_i$ تابع $\text{Cost}_i^{rej+reb}(F_\ell, F_r)$ را تعریف میکنیم. این تابع نمایانگر مینیمم هزینه ممکن رد کردن تراکنش (rej) و متعادل کردن برون بلاکچینی (reb) الگوریتمی است که دنباله تراکنش $X_i$ را پردازش میکند به طوریکه پس از پردازش $x_i$ موجودی سمت چپ کانال برابر با $b(\ell) = F_\ell \geqslant ۰$ و موجودی سمت راست کانال برابر $b(r) = F_r \geqslant ۰$ باشد. با داشتن $\text{Cost}_{i-۱}^{rej+reb}(.,.)$ برای مقادیر

مختلف $F_\ell$ و $F_r$ میتوان $\text{Cost}_i^{rej+reb}(F_\ell, F_r)$ را با بررسی اکشن های مختلف الگوریتم برای تراکنش $i$ ام محاسبه کرد. بدون از دست دادن عمومیت حالتی را که تراکنش $x_i$ از چپ به راست است را بررسی میکنیم. برای تراکنش از راست به چپ هم دقیقا مشابه همین حالت بندی ها را میتوان انجام داد.

۱) اگر الگوریتم تراکنش $i$ ام را رد کند آن گاه موجودی طرفین کانال در دو لحظه $i-۱$ و $i$ مشابه هم است و میتوان نوشت:

$$\text{Cost}_i^{rej+reb}(F_\ell, F_r|rej) \coloneqq \text{Cost}_{i-۱}^{rej+reb}(F_\ell, F_r) + Rx_i + f_۲ \qquad (۱-۶)$$

۲) اگر الگوریتم تراکنش $i$ را بپذیرد بدون اینکه پیش از آن متعادل کردن برون بلاکچینی انجام دهد، آنگاه پردازش تراکنش هزینه ای ندارد اما موجودی طرفین کانال پس از پذیرش تراکنش تغییر میکند.

$$\text{Cost}_i^{rej+reb}(F_\ell, F_r|acc) \coloneqq \begin{cases} \text{Cost}_{i-۱}^{rej+reb}(F_\ell + x_i, F_r - x_i) & \text{If } F_r > x_i \\ \infty & \text{O.W.} \end{cases} \qquad (۲-۶)$$

۳) اگر الگوریتم ابتدا متعادل کردن برون بلاکچینی انجام دهد و بعد تراکنش $i$ را بپذیرد میتوان هزینه الگوریتم را بر حسب مقدار جابجایی پول در حین متعادل کردن به صورت زیر نوشت:

(۳-۶)

$$\text{Cost}_i^{rej+reb}(F_\ell, F_r|reb) \coloneqq \begin{cases} \min_{a \leqslant F_\ell + x_i} \text{Cost}_{i-۱}^{rej+reb}(a, F_r + F_\ell - a) + C \cdot R(F_\ell + x_i - a) + Cf_۲ & \text{If } F_r > \\ \infty & \text{O.W} \end{cases}$$

دقت کنید که الگوریتم بهینه متعادل کردن برون بلاکچینی را تا آخرین لحظه ای که برای پردازش تراکنشی به پول بیشتر نیاز دارد به تعویق می اندازد، پس میتوان فرض کرد اگر الگوریتم تنها زمانی متعادل کردن برون بلاکچینی از راست به چپ انجام میدهد که میخواهد تراکنشی از چپ به راست را بپذیرد. الگوریتم بهینه برای تصمیم گیری درباره هر تراکنش، هزینه اکشن های ذکر شده را مقایسه میکند و بهترین را بر میگزیند یعنی:

(۴-۶)

$$\text{Cost}_i^{rej+reb}(F_\ell, F_r) = \min\{\text{Cost}_i^{rej+reb}(F_\ell, F_r|rej), \text{Cost}_i^{rej+reb}(F_\ell, F_r|acc), \text{Cost}_i^{rej+reb}(F_\ell, F_r|reb)\}$$

از طرفی دقت کنید که در تابع $\text{Cost}_i^{rej+reb}(.,.)$ هزینه شارژ کردن درون بلاکچینی را در نظر نگرفتیم، اما OFF این هزینه را هم در نظر میگیرد. پس OFF از بین همه $F_\ell$ و $F_r$ های ممکن آنهایی را برمی گزیند که مجموع همه هزینه ها را کمینه میکنند. به علاوه OFF هزینه حالتی که همه تراکنش ها را هم

رد کند و $F_\ell$ و $F_r$ هر دو صفر باشند را هم در بین گزینه هایش در نظر میگیرد چون ممکن است برای برخی از دنباله های تراکنش ها اصولا شارژ کردن کانال به هیچ وجه توجیه پذیر نباشد.

$$\text{Cost}_{\text{OFF}}(X_i) = \min\{\min_{F_\ell, F_r \geqslant 0} \text{Cost}_t^{rej+reb}(F_\ell, F_r) + F_\ell + F_r + f_1, \sum_{i\in[1,t]} (R|x_i| + f_2)\} \quad (۵-۶)$$

دقت کنید که OFF چون از پیش به کل دنباله تراکنش دسترسی دارد، عمل شارژ کردن را حداکثر یکبار همان ابتدا انجام میدهد و کانال را دقیقا به میزانی که تا انتها نیاز خواهد داشت شارژ میکند تا فقط یکبار هزینه $f_1$ را بپردازد.

چندین نکته را باید در خصوص معادله ۶-۵ در نظر گرفت:

۱) برای اینکه بتوان مینیمم سازی معادله ۶-۵ را در عمل محاسبه کرد، باید مقادیر تراکنش ها گسسته باشد تا در نتیجه بتوان $F_\ell$ و $F_r$ را هم گسسته در نظر گرفت و برای کمینه کردن تابع هزینه روی یک فضای گسسته متناهی جستجو انجام داد. به همین منظور در این بخش برای سادگی فرض میکنیم که تراکنش ها اعداد طبیعی هستند، دقت کنید که بسته به رنج واقعی تراکنش ها میتوان تمام ضرایب مساله را به نحوی نرمالایز کرد که بعد از نرمالایز کردن اندازه تراکنش ها اعداد طبیعی باشد؛ پس این فرض به هیچ وجه محدود کننده نیست.

۲) میتوان روی آرگومان های تابع مینیمایز معادله ۶-۵ کران هایی پیدا کرد که تعداد سرچ های لازم را کاهش دهد. اگر $F_r^*$ و $F_\ell^*$ آرگومان های تابع کمینه کردن معادله ۶-۵ باشند، برای دنباله تراکنش مشخص $X_t = \{x_1, x_2, \ldots, x_t\}$ میتوان نوشت:

$$F_\ell^* + F_r^* \leqslant \sum_{i\in[1,t]} |x_i| \quad (۶-۶)$$

در معادله ۶-۶ $|x_i|$ نمایانگر اندازه تراکنش فارغ از جهت آن است. پیام اصلی معادله ۶-۶ این است که OFF حتی اگر بخواهد همه تراکنش ها را بپذیرد و هیچ گاه متعادل کردن برون بلاکچینی انجام ندهد و به تراکنش هایی که از سمت مقابل می آیند هم برای پذیرش تراکنش های یک سمت وابسته نباشد، در لحظه ۱ در سمت چپ کانال به اندازه مجموع تمام تراکنش های از چپ به راست و در سمت راست کانال به اندازه مجموع تمام تراکنش های از راست به چپ پول میگذارد، گذاشتن پولی بیشتر از این در کانال فقط باعث افزایش هزینه بی جهت میشود.

**قضیه‌ی ۶-۱** *اگر برای دنباله $X_t$ که اندازه تراکنش های آن اعداد طبیعی هستند تعریف کنیم:*

$$S := \sum_{i\in[1,t]} |x_i| \quad (۷-۶)$$

*آن گاه پیچیدگی محاسباتی پیدا کردن* OFF *به روش* `dynamic programming` *برای این دنباله* $\mathcal{O}(S^{۳}t)$ *است.*

*اثبات.* با شروع از $i = ۱$ و افزودن یک به یک به $i$ ،میتوان جدولی از کل مقادیر $\text{Cost}_i^{rej+reb}(F_\ell, F_r)$ برای تمام

$F_\ell, F_r \in [۱, S]^{۲}$ و تمام زمان های $i \leqslant t$ ساخت و هزینه محاسباتی این عمل $\mathcal{O}(S^{۲}t)$ است.

طبق معادله ۶-۵، OFF باید برای $S$ مقدار مختلف $F_\ell + F_r$ مراحل زیر را طی کند:
OFF باید برای هر $i \leqslant t$ هربار تابع $\text{Cost}_i^{rej+reb}(.,.)$ را با حداکثر $S^{۲}$ آرگومان مختلف فراخوانی کند و مقادیر آن ها را با هم مقایسه کند. در هر گام زمانی هزینه محاسباتی OFF حداکثر $S^{۲}$ است و در مجموع $t$ گام وجود دارد و OFF باید این کار را برای $S$ مقدار مختلف $F_\ell + F_r$ تکرار کند پس در مجموع هزینه نهایی OFF $\mathcal{O}(S^{۳}t)$ است. □

در انتها لازم به ذکر است که استفاده از `dynamic programming` برای پیدا کردن OFF یک مزیت اصلی دارد و آن این است که در حین پیدا کردن جواب بهنیه برای هر دنباله از تراکنش ها، جواب بهینه زیردنباله های آن هم به دست می آید و در نتیجه در پیاده سازی ها میتوانیم به راحتی اوراکلی را که الگوریتم ۴ به آن نیاز دارد فراهم کنیم.

## ۶-۲ مقایسه هزینه الگوریتم ON و OFF

الگوریتم OFF را به روش توصیف شده در قسمت ۶-۱ پیاده سازی کرده ایم و الگوریتم ON را هم طبق الگوریتم ۴ و با کمک از OFF به عنوان اوراکل، پیاده سازی کرده ایم. برای مقایسه هزینه این الگوریتم ها، ۵۰ دنباله تراکنش هر کدام از طول ۵۰ را به طور تصادفی تولید کرده ایم. تراکنش ها را به صورت مستقل از توزیع گاوسی با میانگین ۰ و انحراف معیار ۳ نمونه برداری کرده و سپس آن ها را به نزدیک ترین عدد صحیح گرد کرده ایم زیرا همانطور که در قسمت ۶-۱ توضیح دادیم برای پیدا کردن OFF به روش `dynamic programming`، اندازه تراکنش ها باید گسسته باشد. آزمایش را به ازای مقادیر مختلف پارامتر های تابع هزینه که در زیر آمده است تکرار کرده ایم:

$$R = ۰, f_۱ = ۳, f_۲ \in \{۰/۵, ۲\}, C \in \{۲, ۸\}$$

برای مقایسه ON و OFF ، ۵ متغیر زیر را در هر آزمایش محاسبه کرده و در نهایت روی کل ۵۰ دنباله میانگین گرفته ایم:

۱. هزینه کل (مجموع هزینه های رد کردن + هزینه های شارژ کردن درون بلاکچینی + هزینه های متعادل کردن برون بلاکچینی)

۲. مجموع کل پول کانال در انتهای پردازش تراکنش ها که در جدول با $A(X)$ نمایش داده شده است.

۳. نسبت تعداد تراکنش های پذیرفته شده به کل

۴. میزان کل پول جابجا شده در کانال با متعادل کردن برون بلاکچینی

۵. تعداد کل شارژ کردن های برون بلاکچینی انجام شده

جدول ۶-۱ خلاصه ای از نتایج این آزمایش را نشان میدهد. همانطور که در جدول مشاهده میشود، نسبت هزینه ON به OFF در عمل بسیار کمتر از کران بالای اثبات شده در فصل ۵ است.

**طراحی روش های ابتکاری برای بهبود هزینه ON** [۱] جدول ۶-۱ نشان میدهد که میزان پولی که ON به کانال اضافه میکند بسیار بسیار بیشتر از OFF است و علاوه بر آن مشاهده میکنیم که با افزایش $C$ در یک رنج متوسط، هزینه OFF تغییر چندانی نمیکند اما هزینه ON بسیار سریع رشد میکند. البته این نتیجه قابل انتظار است زیرا ON الگوریتم $(4 + 2\lceil \log C \rceil, f_1)$-recharging را دنبال میکند پس با افزایش $C$، پولی که در هر شارژ کردن به کانال اضافه میکند بیشتر میشود اما متاسفانه بخش زیادی از این پول عملا مورد استفاده قرار نمیگیرد زیرا همانطور که در جدول میبینید نرخ پذیرش ON نه تنها از OFF بیشتر نیست بلکه کم تر هم هست. دلیل آن هم این است که ON از سبد بودجه استفاده میکند تا بودجه تراکنش هایی در رنج های مختلف را جدا از هم قرار دهد. این امر سبب میشود که حتی اگر تراکنش های زیادی هم در محدوده یک سبد بودجه خاص قرار نگیرند، باز هم ON برای آن تراکنش ها همیشه مقداری پول ذخیره کند و هیچ گاه از این پول برای پذیرش تراکنش های در رنج های دیگر استفاده نکند؛ حتی اگر تعداد آن تراکنش ها بسیار زیاد باشد. دلیل این خاصیت های ON این است که این الگوریتم برای بدترین دنباله های موجود طراحی شده پس الگوریتم بدبینی است که با زیادی شارژ

[۱] Heuristics

کردن کانال و با اطمینان حاصل کردن از اینکه برای هر رنجی از تراکنش مقدار از پیش تعیین شده ای بودجه دارد، میخواد تضمین کند که هزینه اش هیچگاه از حد خاصی بالاتر نمیرود. اما میتوان ON را تغییرات اندکی داد به نحوی که کم تر محتاطانه عمل کند و عملکرد آن را برای دنباله تراکنش های خوش رفتار (مثلا نمونه گیری شده از توزیع گاوسی) بهبود داد. برای این منظور دو روش ابتکاری با الهام از ON طراحی میکنیم که کمتر از ON کانال را شارژ میکنند و کم تر بدبینانه عمل میکنند. اولین روش ابتکاری مان را ON-I مینامیم. ON-I زیر الگوریتم $(\lceil \log C \rceil, f_1)$-recharging را دنبال میکند و بر خلاف ON سبد بودجه های مختلف ندارد. ON-I تمام تراکنش های کمتر مساوی $F_{tracker}$ را میپذیرد و هرگاه پول کافی برای تراکنشی نداشت به روش زیر عمل میکند: اگر اندازه تراکنش کمتر مساوی $\frac{F_{tracker}}{C}$ بود، متعادل کردن کانال انجام میدهد طوری که نصف کل پول در یک سمت کانال و نصف دیگر در سمت دیگر قرار گیرد و تراکنش را میپذیرد؛ اما اگر اندازه تراکنش از $\frac{F_{tracker}}{C}$ بزرگ تر بود آن گاه ON-I آن را رد میکند.

همانطور که در جدول ۶-۱ میبینید هزینه ON-I بسیار کمتر از ON و بسیار نزدیک به OFF است ولی در عوض نرخ پذیرش آن کم تر است. دلیل آن هم بدیهتا به خاطر ضریب کمتر شارژ کردن کانال $(\lceil \log C \rceil)$ و همچنین به این دلیل است که ON-Iسبد بودجه های مجزا برای تراکنش های مختلف ندارد و ممکن است تعداد زیادی از تراکنش های بزرگ $(x > \frac{F_{tracker}}{C})$ را رد کند.

با مراجعه به جدول ۶-۱ یک مشکل مشترک بین ON و ON-I میبینیم، هر دو این الگوریتم ها برای حالت $f_2 = 2$ کانال را به طور متوسط بیشتر از ۵ بار شارژ میکنند و چون هر بار شارژ کردن هزینه $f_1$ را دارد این عمل باعث افزایش بسیار زیاد هزینه شده. دلیل این تعداد شارژ کردن زیاد هم این است که هر دو این الگوریتم ها به محضی که $F_{tracker} < A(X_t)$ میشود عمل شارژ کردن را انجام میدهند.

روش ابتکاری ON-II را معرفی میکنیم که دقیقا عین ON-I عمل میکند جز اینکه کانال را هنگامی شارژ میکند که $F_{tracker} < \alpha A(X_t)$ برای یک $\alpha$ دلخواه بزرگ تر از ۱. دقت کنید که اگر قرار دهیم $\alpha = 1$ آن گاه ON-II دقیقا عین ON-I خواهد بود. برای آزمایش های نشان داده شده در جدول ۶-۱ از $\alpha = 2$ استفاده کرده ایم که همانطور که میتوان دید، این امر باعث شده که تعداد شارژ کردن های ON-II نسبت به ON-I برای حالت $f_2 = 2$ تا بیشتر از نصف کاهش یابد و همچنین هزینه کلی ON-II از هر دو ON و ON-I کم تر است.

نکته ای که برای انتخاب $\alpha$ باید مورد توجه قرار گیرد این است که اگر $f_2 >> f_1$ آن گاه شارژ کردن کانال به مراتب پر هزینه تر از رد کردن تراکنش هاست پس باید ترجیحا $\alpha$ بزرگ تر انتخاب شود اما اگر

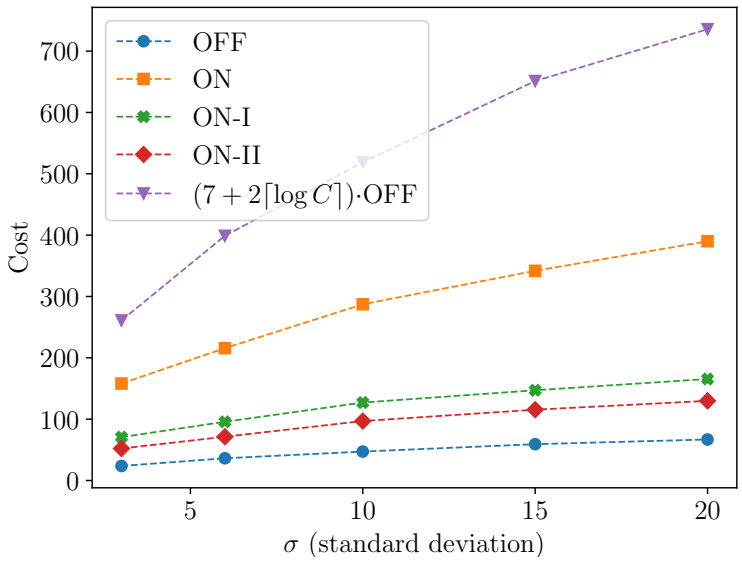
$f_۱ \sim f_۲$ آن گاه بهتر است $\alpha$ های نزدیک به ۱ انتخاب شود زیرا با یکبار شارژ کردن کانال میتوان تعداد زیادی تراکنش را پذیرفت و از پرداختن $f_۲$ برای تعداد زیادی تراکنش جلوگیری کرد.

| Param | | OFF | | | | | ON | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C | $f_2$ | Cost | $A(X)$ | Accept rate | Off-chain rebalancing | Rechargings | Cost | $A(X)$ | Accept rate | Off-chain rebalancing | Rechargings |
| 2 | 0.5 | **15.02** | 6.4 | 0.78 | 0.8 | 1 | **63.3** | 44.26 | 0.50 | 0.9 | 2.18 |
| 8 | 0.5 | **15.21** | 6.38 | 0.77 | 0 | 1 | **87.79** | 69.06 | 0.50 | 0 | 2.04 |
| 2 | 2 | **23.6** | 14.26 | 0.95 | 5.36 | 1 | **127.02** | 100.2 | 0.91 | 0.38 | 5.86 |
| 8 | 2 | **24.5** | 13.9 | 0.92 | 0 | 1 | **184.32** | 156.6 | 0.9 | 0 | 5.84 |
| Param | | ON-I | | | | | ON-II | | | | |
| C | $f_2$ | Cost | $A(X)$ | Accept rate | Off-chain rebalancing | Rechargings | Cost | $A(X)$ | Accept rate | Off-chain rebalancing | Rechargings |
| 2 | 0.5 | **30.74** | 6.86 | 0.44 | 11.18 | 2.18 | **26.52** | 5.56 | 0.41 | 10.22 | 1.2 |
| 8 | 0.5 | **39.98** | 19.86 | 0.48 | 0 | 2.04 | **32.94** | 15.9 | 0.46 | 0 | 1.18 |
| 2 | 2 | **66.16** | 14.42 | 0.84 | 25.48 | 5.86 | **60.38** | 11.3 | 0.78 | 27.16 | 2.2 |
| 8 | 2 | **81.35** | 42.72 | 0.89 | 1.5 | 5.84 | **58.38** | 33.3 | 0.83 | 1.56 | 2.16 |

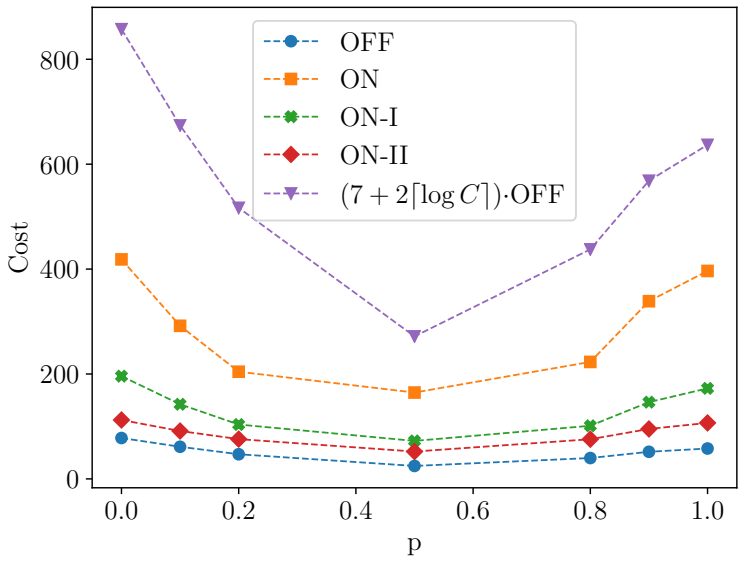
جدول ۶-۱: مقایسه هزینه ها و نحوه کارکرد الگوریتم آنلاین و الگوریتم بهینه آفلاین.

**مشاهده تغییر هزینه الگوریتم ها با تغییر توزیع تراکنش ها** میتوان حدس زد که دنباله تراکنشی که در بخش قبل برای آزمایش هایمان استفاده کردیم دنباله خوش رفتاری است و باعث ایجاد هزینه های بسیار بالا نمیشود. دلیل این امر این است که اولا جهت تراکنش ها تقارن داشت پس انتظار داشتیم تراکنش هایی که از راست به چپ می آیند تقریبا برابر تراکنش هایی باشند که از چپ به راست می آیند و با اندکی شارژ کردن کانال بتوان تراکنش های زیادی را پذیرفت. ثانیا انحراف معیار توزیع گاوسی ای که تراکنش ها را از آن نمونه برداری کرده بودیم ۳ بود که عدد نسبتا کمی است و این باعث میشد تراکنش ها در رنج اندازه هم و نسبتا کوچک باشند. در این بخش انحراف معیار توزیع گاوسی را از ۳ تا ۲۰ افزایش میدهیم و نمودار تغییر هزینه الگوریتم های مختلف را در تصویر ۶-۱ نمایش میدهیم. همانطور که در شکل مشخص است هزینه تمام الگوریتم ها با افزایش انحراف معیار افزایش میابد و این

امر دو دلیل دارد. اولا با زیاد شدن انحراف معیار اندازه تراکنش ها دور تر از هم میشود و احتمال اینکه تراکنش هایی با اندازه مشابه از دو سمت بیاید و باعث خنثی کردن پول لازم شود کم تر میشود، ثانیا با افزایش انحراف معیار تراکنش های بزرگ تر بیشتری ارسال میشود که یا نرخ رد کردن را بالا میبرد یا باعث شارژ بیشتر کانال میشود.




شکل ۶-۱: تغییر هزینه الگوریتم ها با افزایش انحراف معیار توزیع تراکنش ها




شکل ۶-۲: تغییر هزینه الگوریتم ها با تغییر قرینگی دنباله ها

نکته ای که در شکل ۶-۱ باید به آن دقت کرد این است که هزینه الگوریتم ON همواره از کران بالای تئوری آن فاصله بسیار زیادی دارد و این تاییدی بر درستی ضریب رقابتی است.

برای از بین بردن قرینگی دنباله تراکنش ها این بار ۵۰ دنباله هر کدام شامل ۵۰ تراکنش را تحت شرایط زیر تولید میکنیم: اندازه هر تراکنش از یک توزیع folded normal نمونه گیری شده که توزیع گاوسی متناظر آن میانگین صفر و انحراف معیار ۳ دارد. سپس جهت تراکنش ها توسط یک توزیع برنولی با پارامتر $p$ تعیین شده است. ۱ شدن متغیر برنولی نشان دهنده این است که تراکنش از چپ به راست است و در غیر این صورت از راست به چپ است. میانگین هزینه تمام الگوریتم ها به اضافه کران بالای تئوری هزینه الگوریتم ON را در تصویر ۶-۲ رسم کرده ایم. همانطور که انتظار داشتیم با نزدیک تر شدن $p$ به $\frac{1}{2}$ که به معنی متقارن تر شدن دنباله است، هزینه همه الگوریتم ها کمتر میشود. همچنین مجددا میتوان مشاهده کرد که هزینه ON از کران بالای تئوری آن فاصله ی قابل توجهی دارد.

در هر دو تصویر ۶-۱ و ۶-۲ پارامتر های توابع هزینه به شرح زیر هستند:

$$f_1 = 3, f_2 = 2, R = 0, C = 4, \alpha = 2$$

## ۶-۳ بررسی پارامتر های توابع هزینه برای Lightning Network

با بررسی تعداد زیادی از کانال های Lightning Network متوجه شدیم که $f_1$ که همان هزینه ثابت ایجاد کانال است حدود ۱۰۰۰ ساتوشی [۲] است و بسیار بیشتر از $f_2$ است که برای تعداد بسیار زیادی از کانال ها حدود ۱ ساتوشی است [۳، ۴]. با توجه به اینکه $f_1 >> f_2$ میتوان نتیجه گرفت که الگوریتم ON در مورد Lightning Network به صورت زیر عمل میکند: تا زمانی که تعداد تراکنش های دنباله تراکنش از حد مشخصی کم تر است، ON کانالی ایجاد نمیکند که در دنیای واقعی به این معنی است که الگوریتم به کاربران توصیه میکند که تراکنش هایشان را از طرق دیگر انجام دهند، هرچند این امر هزینه بر است اما هزینه آن همچنان کمتر از این است که باز کردن کانال به صرفه باشد. زمانی که دنباله تراکنش از حدی بزرگ تر شد یا در دنیای واقعی تعداد تراکنش های دو کاربر از حدی بالاتر رفت، الگوریتم توصیه به باز کردن کانال میکند.

[۲] 1 Satoshi is 0.00000001 BTC.

**پیدا کردن $C$ به طور تجربی در Lightning Network** از آنجاییکه $C$ نقش اساسی ای در ضریب رقابتی الگوریتم ON دارد، بر آن شدیم تا مقدار آن در Lightning Network را به طور تجربی پیدا کنیم. برای این کار از یکی از بروز ترین اسنپ شات[۳] های این شبکه که مربوط به سپتامبر سال ۲۰۲۱ است استفاده کردیم [۳]. در این اسنپ شات در مجموع ۱۱۷,۸۹۴ کانال وجود دارد که بعد از یکجا در نظر گرفتن کانال های کاربرانی که بیش از یک کانال دارند ۶۳,۸۲۰ کانال باقی ماند. برای هر یک از این کانال ها طول کوتاه ترین دور در گراف را که شامل کانال است، پیدا کردیم و آن را در جدول ۶-۲ منعکس کرده ایم. حدود ۹/۵ درصد کانال ها در هیچ دوری حضور ندارند و در نتیجه برای این کاربران اصولا متعادل کردن برون بلاکچینی امکان پذیر نیست. اما تعداد بسیار زیادی از کانال های شبکه عضو دوری به طول ۴ یا ۵ هستند.

| **Cycle length** | $\leqslant 4$ | 5 | 6 | 7 | N.A. |
|---|---|---|---|---|---|
| **Frequency** | 49,424(77.44%) | 7,758(12.16%) | 469(0.73%) | 12(0.02%) | 6,157(9.65%) |

جدول ۶-۲: طول کوتاه ترین دوری که هر کانال شبکه Lightning Network عضوی از آن است.

اما نکته بسیار مهمی که باید به آن توجه کرد این است که موجودی طرفین کانال در Lightning Network یک داده ی خصوصی است و نمیتوان به آن دست یافت. هرچند مجموع پول طرفین کانال را میتوان با مراجعه به بلاکچین پیدا کرد اما اینکه این پول چگونه به دو طرف تخصیص یافته یک داده ی خصوصی است. در نتیجه ما اطلاعاتی از موجودی کاربران در دور هایی که در Lightning Network پیدا کرده ایم نداریم و برخی از این دور ها ممکن است فاقد پول کافی برای انجام متعادل کردن برون بلاکچینی باشند.

[۳] snapshot

# فصل ۷

# نتیجه‌گیری

در این فصل، ضمن جمع‌بندی نتایج جدید ارائه‌شده در پایان‌نامه، مسائل باز باقی‌مانده و همچنین پیش‌نهادهایی برای ادامه‌ی کار ارائه می‌شوند.

# Bibliography

[1] J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf, 2015.

[2] Raiden network. https://raiden.network/, 2017.

[3] C. Decker. Lightning network research; topology, datasets. https://github.com/lnresearch/topology. Accessed: 2022-04-01.

[4] Lightning network search and analysis engine.

[5] T. L. J. S. Prof. C. Stamm, K.-T. Foerster. Online algorithms tutorial. https://disco.ethz.ch/courses/hs11/des/lectures/des_chapter5.pdf.