تهیه کننده

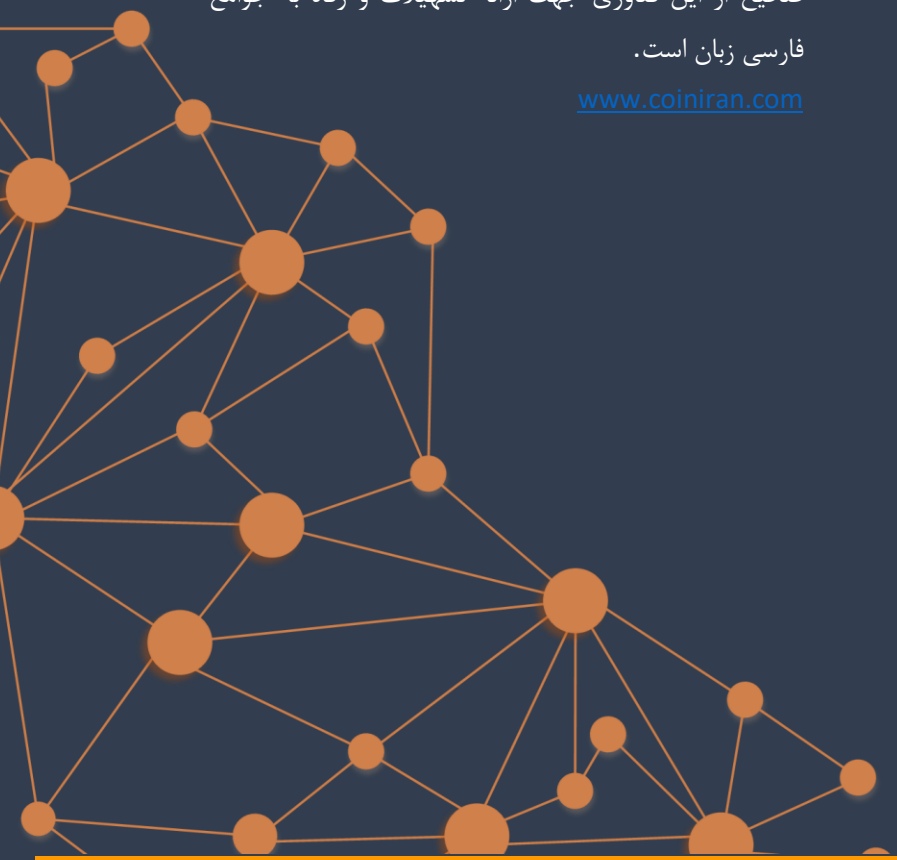
امین محقق

منتشر شده توسط

کوین ایران

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

www.coiniran.com



معرفی پلتفرم Swirlds

توسعه و اجرای نرم افزارهای غیر متمرکز بر روی پایگاه های داده توزیع شده

مقدمه

پایگاه داده های توزیعی اغلب باید ماشین های حالات تکثیر شده ای باشند که تحمل خطای بیزانس را دارند (تحمل خطای بیزانس، یک مشکل در یک سیستم است وقتی که مولفه ای اگر درست کار نکند، بقیه متوجه نمی شوند که کدام بخش در حال خطاست یا صادقانه رفتار نمی کند.

وضعیت را می توان به این شکل توضیح داد: فرض کنید ۹ جنرال با ارتش خودشان در محل های مختلف مستقر هستند و شهری مثل بیزانس را محاصره کرد اند. حالا می خواهند تصمیم بگیرند یا حمله کنند یا عقب نشینی کنند. هر کدام به دیگری پیام می فرستد و رای خود را می دهد. چهار ژنرال رای به عقب نشینی و چهار ژنرال دیگر رای به حمله می دهد. اما اگر ژنرال نهم، به چهار ژنرال اولی رای دهد که عقب نشینی کنید و به چهار ژنرال طرفدار حمله، رای به حمله بدهد، پس یک سری از ژنرال ها تصور می کنند که عقب نشینی رای اکثریت است و یک سری دیگر تصور می کنند که رای اکثریت بر حمله است. بدین ترتیب، سیستم طوری کار می کند که اگر یکی خطا کند یا رای متفاوتی صادر کند، بقیه شبکه متوجه نخواهند شد).

برخی از نویسندگان از اصطلاح «بیزانس» به معنای ضعیف کلمه استفاده می کنند، مثلا فرض را بر این می گیرند که حمله کنندگان به شبکه با یکدیگر دست به یکی نمی کنند، یا اینکه ارتباطات خیلی همزمان انجام نمی شود(و بنابراین تبانی به صورت همزمان رخ نمی دهد). در این جزوه، کلمه بیزانس به معنای دقیق کلمه یا تعریف اصلی آن استفاده می شود: فرض بر این می گیریم که درست تا زیر یک سوم اعضا می توانند مهاجم باشند، می توانند دست به یکی کنند، و می توانند هر پیامی را بین اعضای شبکه پاک کنند یا به تاخیر اندازند.

حمله کنندگان می توانند شبکه را با تاخیر یا پاک کردن هر پیامی، کنترل کنند، اگرچه هر بار، اگر یک عضو صادق، مکررا پیام ها را به عضو دیگر بفرستد، حمله کنندگان نهایتا باید یک پیام را اجازه بدهند که منتقل شود. فرض بر این است که امضاهای امن دیجیتالی وجود دارد و بنابراین حمله کنندگان نمی‌توانند پیام ها را بدون اینکه قابل تشخیص باشد، تغییر دهند. فرض بر این است که توابع هش امنی وجود دارد و از این لحاظ نمی توان تبانی کرد. این جزوه قصد دارد الگوریتم توافقی هش گراف **Swirlds** را پیشنهاد و توصیه کند و تحمل خطای بیزانس را با تعریف دقیق اش به اثبات می‌رساند.

براساس قضیه **FLP** هیچ سیستم بیزانسی قطعی کاملا نمی تواند ناهمزمان باشد و تاخیر در پیام ها نامحدود باشد، و همچنان توافق و اجماع تضمین شده باشد. اما برای سیستم های غیرقطعی می توان با احتمال یک به توافق و اجماع رسید. الگوریتم توافقی هش گراف کاملا ناهمزمان، غیرقطعی است و به توافق بیزانس با احتمال یک می رسد.

برخی از سیستم ها مانند **Paxos** یا **Raft** از یک رهبر یا لیدر استفاده می کنند که می تواند در مقابل تاخیرهای زیاد آسیب پذیر باشد، اگر که یک مهاجم به این لیدر یک حمله **denial-of-service attack** انجام دهد (معمولا با درخواست های زیاد، امکان دسترسی به سرویس را از فرد می گیرند). بسیاری از سیستم ها را حتی می توان با یک کلاینت مهاجم به تاخیر انداخت. در واقع، این جزوه پیشنهاد می‌کند چنین سیستم هایی با این آسیب پذیری را بهتر است **Byzantine fault survivable** توصیف کرد تا **Byzantine fault tolerance**. توافق هش گراف از یک لیدر استفاده نمی کند و در مقابل حملات **DoS** علیه بخش کوچکی از اعضا مقاوم است.

سیستم های دیگر، مانند بیت کوین، بر بلاک چین با گواه اثبات کار مبتنی هستند. این تمام مشکلات فوق را برطرف می کند. اما چنین سیستم هایی نمی توانند بیزانسی باشند، چون یک عضو هرگز مطمئن نیست که آیا یک توافق حاصل شده است یا نه. آنها فقط حدی از اطمینان دارند که طی زمان ادامه می‌یابد. اگر دو بلاک همزمان ماین شوند، پس زنجیره فورک خواهد کرد تا وقتی کامیونیتی به این توافق برسد که روی کدام شاخه کار را ادامه دهند. اگر بلاک ها به آهستگی اضافه شوند، پس کامیونیتی همیشه روی شاخه طولانی تر بلاک ها را اضافه می کند و سرانجام شاخه دیگر از رشد بازمی ماند و می‌توان آن را هرس کرد زیرا متوقف شده است. این باعث ناکارآمدی می شود از این جهت که برخی از بلاک ها با اینکه به خوبی ماین شده اند اما به هر حال به دور انداخته می شوند. همچنین بدین معناست که لازم است سرعت استخراج بلاک ها را کاهش داد تا کامیونیتی بتواند شاخه های اضافی را سریع تر از ایجاد شاخه های جدید، هرس کند. هدف گواه اثبات کار همین است. با مجبور کردن ماینرها به حل مسائل محاسباتی دشوار برای ماین یک بلاک، تضمین می شود که کل شبکه به اندازه کافی (و به طور متوسط) تاخیر داشته باشد.

الگوریتم توافقی هشگراف، معادل یک زنجیره بلاک است که زنجیره در آن پیوسته در حال شاخه دادن است، و در آن هیچ بلاکی متوقف و دور انداخته نمی شود و هر ماینری اجازه دارد که هر ثانیه هر بلاک جدیدی که می خواهد را بدون نیاز به گواه اثبات و با کارایی صد در صد ماین کند.

همچنین بلاک چین های مبتنی بر گواه اثبات کار نیاز دارند که نیروی برق برای محاسبات اضافی را حیف و میل کنند و ریگ های استخراج گران قیمت باید خریداری شود. یک سیستم مبتنی بر گواه زمان انقراض (**proof-of-expired-time**) با استفاده از چیپ های سخت افزاری قابل اعتماد که برای مدت طولانی تاخیر ایجاد می کنند و گویی محاسبات گواه اثبات کار انجام شده است، می تواند از هرز رفتن نیروی برق جلوگیری کند (اگرچه شاید از هزینه های ریگ استخراج جلوگیری نکند). با وجود این، لازم است که تمامی مشارکین به شرکتی که این چیپ ها را تولید می کند، اعتماد کنند. این اعتماد به تولید کننده چیپ در برخی مواقع وجود دارد، اما نه در برخی از مواقع دیگر، مانند وقتی که **FreeBSD** تغییر کرد تا برای امنیت اعداد تصادفی صرفا روی سخت افزار **RDRAND** متکی نباشد، زیرا (به قول آنها) «ما نمی توانیم دیگر به آنها اعتماد کنیم.»

سیستم های توافقی بیزانس برای توافق بیزانس توسعه یافته اند تا از مشکلات فوق ممانعت به عمل آید. این سیستم ها معمولا پیام هایی را برای رای گیری بین اعضا رد و بدل می کنند. برای اینکه **n** عضو در مورد یک مسئله بله یا خیر، تصمیم بگیرند، برخی از سیستم ها لازم است تا از پیام های **O(n)** استفاده کنند تا به سراسر شبکه ارسال شود. سیستم های دیگر لازم است تا از پیام های **O(n2)** یا حتی **O(n3)** برای هر تصمیم باینری استفاده کنند. یک الگوریتم برای یک تصمیم بله یا نه منفرد را می‌توان به تصمیم گیری روی کل نظم مجموعه ای از تراکنش ها بسط داد و این می تواند موجب افزایش ترافیک رای ها باشد. هش گراف هیچ رایی به شبکه نمی فرستد چون تمامی رای گیری ها مجازی است.

مفاهیم کلیدی

الگوریتم توافقی هش گراف مبتنی بر مفاهیم کلیدی زیر است:

تراکنش ها: هر عضوی می تواند یک تراکنش امضا شده در هر زمانی ایجاد کند. تمامی اعضا یک کپی از آن بدست می آورند و کامیونیتی به توافق بیزانس روی نظم این تراکنش ها می رسد.

انصاف (Fairness) برای یک گروه کوچک از مهاجمین باید غیرممکن است که بتوانند به طور غیرمنصفانه ای روی نظم تراکنش ها که مطابق توافق انتخاب شده، تاثیر بگذارند.

سخن پراکنی– (Gossip) اطلاعات توسط هر عضوی مکررا پخش شده و به صورت تصادفی به اعضای دیگر فرستاده می شود و هر آنچه می دانند را به آنها می گویند.

هش گراف– یک ساختار داده که ثبت می کند که چه کسی به چه کسانی و با چه نظم و ترتیبی خبررسانده است.

گوسیپ درباره گوسیپ (از این بعد عبارت انگلیسی را می آوریم): هش گراف با پروتوکول گوسیپ، پخش می شود. اطلاعات پراکنده شده تاریخ خود گوسیپ هستند، پس این گوسیپ یا سخن پراکنی است درباره خود گوسیپ. این پهنای باند بسیار اندکی به جز خود گوسیپ درباره تراکنش ها استفاده می کند.

رای گیری مجازی: هر عضوی یک کپی از هش گراف دارد، پس آلیس می تواند محاسبه کند که باب چه رایی را برای او فرستاده است، اگر که یک پروتوکول توافق بیزانس سنتی برای ارسال آرا استفاده شود، پس باب نیازی نیست واقعا بداند که آلیس چه رایی داده است. هر عضوی می تواند روی هر تعداد از تصمیمات به توافق بیزانس برسد بدون اینکه یک رای منفرد حتی فرستاده شده باشد. هش گراف به تنهایی کافی است. پس هیچ پهنای باندی به جز گوسیپ هش گراف استفاده نمی شود.

شاهدین معروف (famous witnesses): کامیونیتی پروتوکل های توافقی بیزانس را با استفاده از $O(n \log n)$ برای مسائل متفاوت بله یا نه به شکل «آیا رویداد x پیش از رویداد y رخ داده است؟»، اجرا و فهرستی از n تراکنش را به نظم در آورد. یک رویکرد سریعتر این است که تنها تعداد معدودی از رویداد ها انتخاب شود که شاهد خوانده می شوند. یک شاهد معروف خوانده می شود اگر هش گراف نشان دهد که اغلب اعضا این شاهد را بلافاصله بعد از ایجادش می بینند. پس کافی است که پروتوکول توافقی بیزانس را برای شاهدین اجرا کنیم و با تصمیم در مورد هر شاهد با سئوال "آیا این شاهد مشهور است؟". وقتی توافق بیزانس روی مجموعه دقیقی از شاهدین معروف حاصل شد، پس گرفتن یک هش گراف از نظم کلی تمامی رویداد آسان می شود.

مشاهده قوی (Strongly seeing): با توجه هر دو x و y در هش گراف، می توان بلافاصله محاسبه کرد که آیا x می تواند y را قویا مشاهده کند، پاسخ مثبت خواهد بود اگر آنها به هم با مسیرهای مستقیم متعددی از طریق تعداد اعضای کافی به هم وصل شده باشند. این مفهوم یک کلید لما (Lemma Key) ایجاد می کند. اینکه اگر آلیس و باب هردو بتوانند رای مجازی کارول را روی یک سئوال مشخص محاسبه کنند، پس آلیس و باب یک پاسخ یکسان می گیرند. این لما (Lemma)، بنیادی را برای بقیه اثبات های ریاضی توافق بیزانس با احتمال یک ایجاد می کند.

گوسیپ درباره گوسیپ

توافق هش گراف از یک پروتوکول گوسیپ استفاده می کند. یعنی اینکه یک عضو مانند آلیس عضو دیگری را به صورت تصادفی مثلا باب انتخاب می کند؛ بعد آلیس به باب تمامی اطلاعاتی که تاکنون می‌داند را بازگو می کند. آلیس سپس به طور تصادفی به اعضای دیگر کمپین نیز اطلاعاتش را می‌رساند. باب نیز همین کار را مکررا انجام می دهد و تمامی اعضا نیز به همین ترتیب. بدین ترتیب، اگر یک عضو منفرد از اطلاعات جدیدی آگاهی یابد، به طور تصاعدی در سراسر کامیونیتی پخش می شود تا اینکه همه از این اطلاعات جدید آگاهی می یابند.

تاریخ هر پروتوکول گوسیپ را می توان با یک گراف مانند شکل ۱ به تصویر کشید.

THE SWIRLDS HASHGRAPH CONSENSUS ALGORITHM - SWIRLDS-TR-2016-01 5

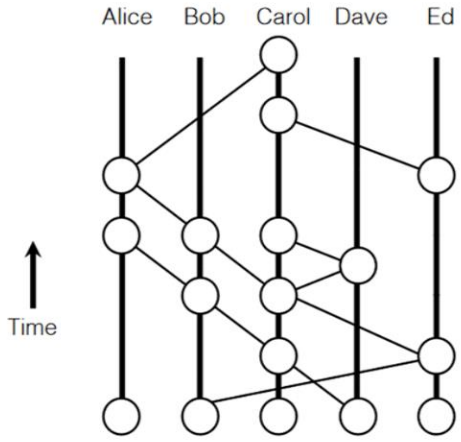


FIGURE 1. Gossip history as a directed graph. The history of any gossip protocol can be represented by a graph where each member is one column of vertices. When Alice receives gossip from Bob, telling her everything he knows, that gossip event is represented by a vertex in the Alice column, with two edges going downward to the immediately-preceding gossip events by Alice and Bob.

هر راس در ستون آلیس نماینده یک رویداد گوسیپ است. برای مثال، رویداد فوقانی در ستون آلیس نشانگر این است که باب یک گوسیپ را به آلیس فرستاده و در آن همه اطلاعاتی که می دانسته را ارسال کرده است. راس مربوط به باب دو خط رو به پایین دارد، که نشان دهنده گوسیپ هایی است که قبلا از سوی دیگران به باب فرستاده شده است. زمان در گراف رو به بالاست، پس رئوس پایینی تر نشان دهنده رویدادهایی هستند که زودتر در تاریخچه پیش آمده است. در یک پروتوکول گوسیپ معمولی، چنین دیاگرامی صرفا برای بحث روی پروتوکول استفاده می شود، هیچ گراف واقعی مانند این، هیچ جا در حافظه ذخیره نمی شود.

در توافق هش گراف، یک گراف یک ساختار داده واقعی است. مثال ۲ تصویر کننده این ساختار داده است. هر رویداد (یا راس) در حافظه به صورت مجموعه ای از بایت ها ذخیره می شود که توسط خالق آن امضا شده است. برای مثال، یک رویداد توسط آلیس (با رنگ قرمز) این را ثبت می کند که باب یک گوسیپ به وی ارسال کرده که در آن هرچه می دانسته را به وی گفته است. این رویداد توسط آلیس ایجاد شده و به امضای او می رسد و این شامل هش های دو رویداد دیگری است: یکی آخرین رویداد خود آلیس و دومی آخرین رویداد باب پیش از این همزمانی گوسیپ. رویداد قرمز همچنین می تواند شامل یک **payload** از هر تراکنشی باشد که آلیس در آن لحظه انتخاب می کند و شاید یک **timestamp** باشد که زمان و تاریخی است که آلیس ان را ایجاد کرده است. رویدادهای پیشین این رویداد (با رنگ خاکستری) در آن شامل نمی شوند اما در مجموعه هش های کریپتوگرافیک تعیین کننده هستند. ساختارهای داده با گراف های هش برای اهداف دیگر استفاده می شود مثلا در**Git** که رئوس، نسخه ای یک درخت فایل هستند و خطوط نماینده تغییرات. اما **Git** ثبت نمی کند که چگونه

پروتوکول های گوسیپ برای انتقال طیف وسیعی از اطلاعات مورد استفاده فراوان هستند. می توان از آنها در پراکنش هویت کاربران، تراکنش ها یا بلاک های بلاک چین استفاده کرد یا هر اطلاعات دیگری که لازم است توزیع شود. اما چه می شود اگر از این پروتوکول برای گوسیپ کردن درباره خود گوسیپ ها استفاده شود؟ چه می شود اگر اعضا خود هش گراف را به یکدیگر منتقل کنند؟ وقتی باب به آلیس سخنی را ارسال می کند، باب به وی تمامی رویدادهایی را که می داند و آلیس نمی داند، به اطلاعش می رساند.

گوسیپ کردن یک هش گراف به مشارکین شبکه اطلاعات بسیار زیادی می دهد. اگر یک تراکنش جدید در بخش پی لود (**payload**) یک رویداد باشد، به سرعت به تمامی اعضا پخش خواهد شد تا اینکه همه از آن باخبر شوند. آلیس از این تراکنش مطلع خواهد شد و او خواهد دانست که دقیقا چه زمانی باب از این تراکنش مطلع شده است. و آلیس خواهد دانست که دقیقا چه زمانی کارول از این که باب از این تراکنش آگاه شده، مطلع شده است. زنجیره عمیقی از چنین انعکاس هایی وقتی میسر می شود که تمامی اعضا یک کپی از هش گراف را داشته باشند. همانطور که هش گراف به سمت بالا رشد می‌کند، اعضای متفاوت ممکن است از رویدادهای جدید بالای هش گراف، زیرمجموعه های اندکی متفاوت داشته باشند، اما به سرعت و زود از همه رویدادها خبر دار خواهند شد. به علاوه، اگر آلیس و باب هردو یک رویداد مشخص را داشته باشند، پس تضمین شده است که آنها هردو همه رویدادهای پیشین آن را نیز دارند. همه اینها الگوریتم قدرتمندی را از جمله برای تحمل خطای بیزانس مهیا می‌کند.

برای این ارتباطات بسیار اندکی نیز لازم است. اگر یک کامیونیتی صرفا تراکنش های امضا شده را گوسیپ می کرد، پس میزان مشخصی از پهنای باند لازم بود. اما اگر به جای آن، یک هش گراف را گوسیپ کند، و اگر تراکنش های کافی وجود داشته باشد که یک رویداد معمولی شامل حداقل یک تراکنش باشد، پس میزان رد و بدل اطلاعات اندک خواهد بود. به جای اینکه آلیس یک تراکنش را امضا کند، وی رویدادی را امضا می کند که شامل این تراکنش است. به هر حال، وی فقط یک امضا ارسال می کند. در هر دو حالت، وی باید تراکنش خودش را ارسال کند. تنها مقدار اطلاعات اضافی ارسال این است که وی باید دو هش را ارسال کند. در شکل دوم، آلیس به کارول رویداد قرمز رنگ را ارسال نخواهد کرد تا زمانی که کارول از قبل تمامی رویدادهای پیشینش را داشته باشد. (یا از آلیس گرفته باشد یا همزمان از کس دیگری). پس آلیس نیاز نیست که دو هش از دو رویداد والد آبی رنگ ارسال کند. کافی است به کارول بگوید که این رویداد یک والدش با آلیس است و والد دیگر از باب می آید. با فشرده سازی مناسب، این را می‌توان با چند بایت ارسال کرد و فقط چنددرصدی به سایز پیام اضافه می شود.

الگوریتم توافقی

کافی نیست که اطمینان حاصل کنیم که هر عضو هر رویدادی را می داند. همچنین لازم است که روی نظم خطی رویدادها نیز توافق حاصل شود و بنابراین روی تراکنش ثبت شده داخل رویدادها به اجماع رسید. در اغلب پروتوکول تلرانس خطای بیزانس که فاقد لیدر هستند، اعضا باید رای یکدیگر را ببینند. پس برای اینکه n عضو موافق یک سئوال ساده بله/ خیر باشند، لازم است که $O(n2)$ پیام رای در شبکه ارسال شود، و هر عضوی به هر عضو دیگری رای اش را می گوید. برخی از این پروتوکول ها نیازمند رسید هایی روی آرای ارسالی هستند و بنابراین به $O(n3)$ تبدیل می کند. و همچنین نیازمند مراتب متعدد رای گیری هستند که باعث افزایش پیام های ارسالی می شود.

توافق هش گراف نیاز به این ندارد که رایی ارسال شود. هر عضو یک کپی از هش گراف را دارد. اگر آلیس و باب هر دو یک هش گراف یکسان داشته باشد، پس آنها می توانند نظم کلی رویدادها را مطابق با تابع قطعی هش گراف محاسبه کنند و هر دو یک جواب می گیرند. بنابراین، توافق بدست می آید حتی بدون ارسال پیام رای.

البته آلیس و باب ممکن است دقیقا هش گراف یکسانی را در هر لحظه نداشته باشند. هش گراف آنها معمولا در مورد رویدادهای قدیمی تر یکسان است. اما برای رویدادهایی که اخیرا ایجاد شده، هرکدام ممکن است رویدادهایی را داشته باشد که دیگری هنوز ندیده است. به علاوه، گاهی ممکن است یک رویداد جدید به کامیونیتی ارسال کند و در مکان پایین تری در هش گراف قرار گیرد. الگوریتم توافقی هش گراف با سیستمی از پس این مسائل برمی آید که بهتر اسم آن را رای گیری مجازی گذاشت.

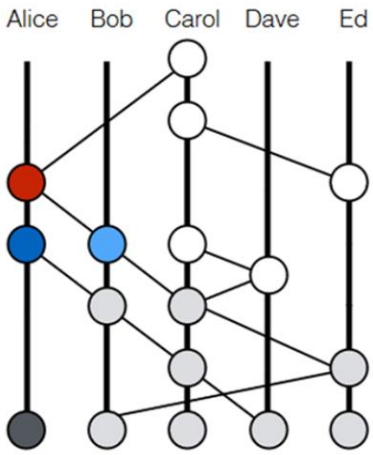


FIGURE 2. The hashgraph data structure. Alice creates an *event* (red) recording the occurrence of Bob doing a gossip sync to her and telling her everything he knows. The event contains a hash of two parent events (blue): the self-parent (dark blue) by the same creator Alice, and the other-parent (light blue) by Bob. It also contains a payload of any new transactions that Alice chooses to create at that moment, and a digital signature by Alice. The other ancestor events (gray) are not stored in the red event, but they are determined by all the hashes. The other self-ancestors (dark gray) are those reachable by sequences of self-parent links, and the others (light gray) are not.

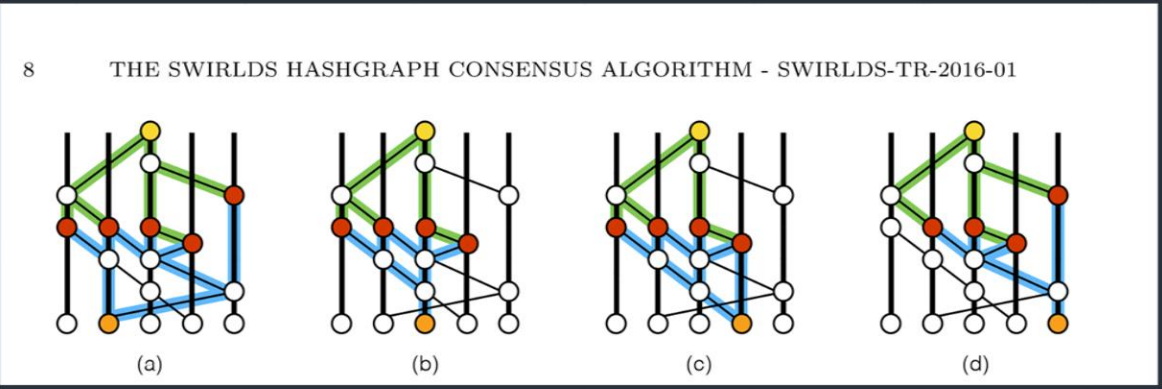
فرض کنید که آلیس یک هش گراف A دارد و باب هش گراف B را. این هش گرافها در هر لحظه ممکن است اندکی متفاوت باشند، اما همیشه با هم سازگار هستند. سازگاری یعنی اینکه اگر این دو هش گراف هر دو رویداد X را داشته باشند، پس آنها همیشه دقیقا مجموعه یکسانی از رویدادهای پیشین (نیاکان) این رویداد را خواهند داشت و دقیقا هر دو مجموعه یکسانی از خطوط بین این رویدادهای پیشین را نیز خواهند داشت. اگر آلیس از رویداد X باخبر است و باب خیر، و هردو صادق بوده و فعالانه مشارکت داشته باشند، پس ما می توانیم انتظار داشته باشیم که باب نیز به سرعت از رویداد X از طریق پروتوکول گوسیپ خبردار خواهد شد. الگوریتم توافقی فرض را بر این می گیرد که بالاخره باب رویداد X را خواهد دید اما فرضی از این ندارد که این چقدر سریع رخ خواهد داد. این پروتوکول کاملا ناهمزمان است و درباره مدت زمان یا سرعت گوسیپ و میزانی پیشروندگی آن پیش فرضی ندارد. آلیس نظم کلی رویدادها را در هش گراف A با محاسبه مجموعه ای از انتخاب ها (elections) محاسبه خواهد کرد. در هر انتخاب، برخی از رویدادها در هش گراف A تصور می‌شوند که یک رای داده اند و در مورد برخی دیگر رویدادها، تصور می شود که رای دریافت کرده اند. آلیس انتخاب های متعددی را محاسبه خواهد کرد و یک رویداد مشخص در برخی از انتخاب ها ممکن است مشارکت داده شود و در برخی از انتخاب ها خیر و بنابراین آرای متفاوتی در انتخابات متفاوت ریخته خواهد شد. در واقعیت کس دیگری مثل باب در این انتخابات مشارکتی ندارد. این صرفا یک محاسبه ای است که آلیس به صورت لوکال انجام می دهد، وی محاسبه می کند که باب چه رای می-توانسته به وی فرستاده باشد، اگر باب واقعی واقعا آرا در اینرنت به وی می فرستاد.

این رای گیری مجازی چند مزیت دارد. علاوه بر حل مسئله پهنای باند، اطمینان حاصل می شود که اعضا همیشه آرای خود را مطابق قواعد محاسبه می کنند. اگر آلیس صادق باشد، وی آرای مجازی را برای باب مجازی که صادق است محاسبه می کند. حتی اگر باب متقلب باشد، وی نمی تواند با رای نادرست باب مجازی، آلیس را فریب دهد.

باب می تواند به روش های متفاوتی سعی کند تقلب کند. فرض کنید باب رویداد **x** را با یک هش والد شخصی به رویداد پیشین اش یعنی **z** ایجاد کرده است. بعد باب یک رویداد **y** را هم ایجاد می کند و هش والد شخصی **z** را می دهد به جای اینکه هش والد شخصی **x** را بدهد. این یعنی که رویدادها در هش گراف باب دیگر یک زنجیره نخواهند بود و اکنون یک درخت ایجاد شده و باب فورک کرده است. اگر باب رویداد **x** را به آلیس و رویداد **y** را به کارول گوسیپ کند، پس برای مدت زمانی، آلیس و کارول ممکن است از فورک خبردارنباشند. و آلیس ممکن است یک رای مجازی برای **x** محاسبه کند که متفاوت است با رای مجازی کارول برای رویداد **y**.

الگوریتم توافقی هش گراف مانع چنین حمله ای می شود. آن این کار را براساس مفهومی مانند دیدن یکدیگر و مفهوم دیدن قوی یکدیگر **strongly seeing** انجام می دهد. اینها مبتنی هستند بر تعاریف از بر تعاریف **ancestor** و **self-ancestor** و هر رویدادی در نظر گرفته می شود که هم یک **ancestor** است و هم **self-ancestor** خودش.

اگر باب دو رویداد **x** و **y** ایجاد کند، هیچکدام آنها **self-ancestor** دیگری نیست و بنابراین باب با فورک کردن تقلب کرده است. اگر رویداد **w**، رویداد **x** را به عنوان **ancestor** خود داشته باشد، اما **y** را به عنوان **ancestor** خود نداشته باشد، پس رویداد **w** می تواند رویداد **x** را ببیند. اما اگر هم **x** و هم **y** رویداد **w** باشند، پس **w** هیچکدام از آنها را نمی‌بیند و نه هیچ رویداد دیگری توسط خالق آن. به عبارت دیگر، رویداد **w** می تواند رویداد **x** را ببیند اگر **x** برای آن شناخته شده باشد و هیچ فورکی توسط خالقی که می شناسد ایجاد نشده باشد.



اگر n عضو وجود دارد، پس یک رویداد w بنا به تعریف قویا (strongly seeing) رویداد x را می بیند اگر بتواند 2n/3 رویداد از اعضای مختلف ببیند که هرکدام انها می توانند رویداد x را ببینند. این مفهوم در شکل ۳ به تصویر کشیده شده است. چهار کپی از هش گراف یکسان نشان داده شده که هرکدام یک رویداد مختلف در پایین هش گراف با رنگ نارنجی مشخص است. در هش گراف d، رویداد زردرنگ در بالا می تواند چهار رویداد قرمز رنگ را از اعضای مختلف ببیند و هرکدام از انها می توانند رویداد نارنجی رنگ را در پایین مشاهده کند. این امر برای سه هش گراف دیگر نیز صادق است، فقط اینکه در هش گراف a، پنج رویداد قرمز رنگ وجود دارد. اما فقط چهار رویداد برای قویا دیده شدن لازم است، چون این مثال n=5 است و کوچکترین عدد بزرگتر از 2n/3، عدد چهار است.

این مفهوم یک پروتوکول را برای دستیابی به تولرانس خطای بیزانس در اختیار قرار می دهد بدون هیچ رای گیری واقعی و فقط از طریق رای گیری مجازی.

در رای گیری مجازی، وقتی رویداد x به برخی از سئوالات بله/خیر جواب می دهد، این رای صرفا به عنوان یک تابع از نیاکان یا ancestor های x محاسبه می شود. این رای تنها وقتی در نظر گرفته می‌شود که از رویداد x به اخلافش (رویدادهای فرزند) ارسال شده باشد مانند رویداد w، با این شرط که w بتواند رویداد x را قویا ببیند. در بخش پنجم اثبات می شود که اگر x و y، شاخه های مختلف یک فورک غیرمجاز باشند، پس w می تواند یکی از x و y را قویا ببیند اما نه هر دو را. به علاوه، اگر هش گرافهای A و B سازگار باشند، پس برای یک رویداد ممکن نیست که بتواند x را قویا در A ببیند و رویداد دیگر y را در هش گراف Bقویا ببیند. این گزاره (لما) سنگ بنای اثبات بیزانس است. تضمین می کند که حتی اگر یک مهاجم سعی در فورک کردن داشته باشد، نمی تواند باعث شود که اعضای متفاوت روی نظم و ترتیب متفاوت رویدادها تصمیم بگیرند.

برخی از الگوریتم های توافق بیزانس لازم دارند که اعضا رسیدهایی را نیز برای رایی که دریافت می‌کنند، ارسال کنند تا مانع شوند که آلیس رای های ناسازگاری را به باب و کارول بفرستد. بین این رسید ها و مفهوم مشاهده قوی مشابهت هایی وجود دارد.

با توجه به این تعاریف، پروتوکول توافقی هش گراف را می توان با الگوریتم هایی در شکل های ۴ ۵ ۶ و ۷ بدست داد.

الگوریتم اصلی در شکل چهار نشان می دهد که ارتباطات بسیار ساده است: آلیس به طور تصادفی یک عضو دیگر مانند باب را انتخاب می کند و به او تمامی رویدادهایی که می داند را ارسال می کند. سپس باب یک رویداد جدید ایجاد کرده و این فکت که گوسیپی انجام شده را ثبت می کند.

این پروتوکول ساده گوسیپ برای صحت و تولرانس خطای بیزانس کفایت می کند. اما برای بهبود کارایی آن می تواند به شیوه های مختلف توسعه یابد. برای مثال، آلیس و باب باید به یکدیگر بگویند که چه رویدادهای دیگری را از قبل می دانند، پس آلیس به باب تمامی رویدادهایی را که می داند و باب نمی داند را ارسال می کند. در این پروتوکول لازم است که آلیس رویدادها را با یک نظم توپولوژیکال بفرستد، پس باب همیشه یک والد رویداد را قبل از دریافت آن رویداد در اختیار دارد. این پروتوکول باید حتی بگوید که بعد از همزمان شدن آلیس با باب، باب نیز بلافاصله با آلیس همزمان می شود. سینک ها یا همزمانی های متعدد در یک لحظه رخ می دهد، پس آلیس باید با اعضای متعدد در یک لحظه همزمان باشد. این کارها و دیگر کارهای بهینه ساز را می توان استفاده کرد اما درهمین حد ساده کفایت می کند.

مفید است که یک روند برای هر رویداد به عنوان یک تابعی از نیاکانش داشته باشیم. در **divideRounds** در شکل پنج، برای هر رویداد شناخته شده یک عدد صحیح روند اختصاص داده می شود و تابعی است از اعداد روند نیاکانش. هش گراف ها در شکل ۳ نشان می دهند که چگونه این کار انجام می شود. اگر تمامی رویداد ها در ردیف پایین روند **r** باشند، پس بقیه رویدادها در این شکل ها نیز روند **r** خواهند بود، به جز برای رویداد زرد رنگ که روند **r+1** خواهد بود. رویداد زردرنگ روند بیشتری دارد چون قادر است که رویدادهای روند **r** را قویا مشاهده کند. اولین رویداد در تاریخچه، با عدد روند **r=1** مشخص می شود، پس تمامی روندهای بعدی به این ترتیب تعیین می‌شوند. هر رویدادی نهایتا یک عدد روند دارد که ایجاد کرده و یک روندی دارد که دریافت کرده است. این را روند یا **round number** می خوانیم.

برای هر عضو مشخصی، اولین رویدادی که در هر روند ایجاد می کنند یک شاهد **witness** خوانده می‌شود. تنها رویدادهای شاهد هستند که آرای مجازی را ارسال و دریافت می کنند. این در پروسژور **decideFame** در شکل شش مشخص شده است. این پروسژور جایی است که توافق بیزانس رخ می‌دهد. برای هر شاهد، تعیین می کند که آیا مشهور هست یا خیر. یک شاهد مشهور است اگر بسیاری از شاهدین در روند بعدی بتوانند آن را ببینند و مشهور نیست اگر نتوانند آن را ببینند. پروتوکول توافقی بیزانس انتخاباتی را برای هر شاهد برگزار می کند تا تعیین کند که آیا مشهور است یا نه. برای یک شاهد **x** در روند **r**، هر شاهد در روند **r+1** رای خواهد داد که آیا آن مشهور است یعنی می‌تواند آن را بتواند ببیند. اگر بیشتر از **2n/3** شاهد آن را ببیند، پس ان مشهور خواهد بود. اگر رای بیش از این باشد، پس این کار برای هر روندهای بعدی که ممکن باشد تکرار خواهد شد. برای دفاع در مقابل مهاجمینی که می توانند اینترنت را کنترل کنند، روندهای دوره ای یا **coin rounds** وجود خواهد داشت که شاهدین می توانند به صورت گمنام رای دهند.

این یعنی که حتی اگر یک مهاجم بتواند تمامی پیام ها را روی اینترنت کنترل کند، هنوز شانسی است که کامیونیتی بتواند به طور تصادفی به آستانه $2n/3$برسد.

در شکل ششم، الگوریتم به کار ادامه می دهد اگر خط $if\ d=1$ تغییر کند به $if\ d=2$. در این الگوریتم جدید، هر انتخاب یک روند بعدتر آغاز خواهد شد. در هر روند، ابتدا با $d=1$ همه انتخابات اجرا خواهد شد. اگر شهرت هر شاهد در این روند تعیین شود، پس $2n/3$ یا اعضای کمتری شاهدین مشهور را در این روند ایجاد می شوند، پس انتخابات برای فقط این روند با $d=2$اجرا خواهد شد. با این الگوریتم پیوندی یا هیبریدی، تمامی قضایا در این جزوه درست خواهد بود از جمله گواه تلرانس خطای بیزانس. برای روندهایی که انتخابات جدید را برگزار می کنند، زمان برای توافق شاید ۲۰ درصد افزایش خواهد داشت. اما این در عمل به ندرت اتفاق می افتد و وقتی هم انجام شد، باید تعداد اعضای مشهور را افزایش دهد تا انصاف یا $fairness$ تضمین شود.

وقتی توافق روی این حاصل شد که هر شاهد در یک روند مشخص مشهور است یا خیر، به آسانی می‌توان از ان برای تعیین یک $timestamp$ توافقی و یک نظم کلی توافقی روی رویدادهای قدیمی تر استفاده کرد. این کار با پروسژور $finalOrder$ در شکل هفت انجام می شود.

ابتدا، روندهای دریافتی محاسبه می شود. رویداد x یک روند دریافتی rرا دارد اگر آن اولین روندی باشد که تمامی شاهدین معروف از اخلاف یا فرزندان آن هستند و شهرت از شاهد برای روندهای کمتر یا معادل با r تعیین می‌شود.

سپس زمان دریافتی یا **received time** محاسبه می شود. فرض کنید رویداد **x** یک روند دریافتی **r** دارد و آلیس یک شاهد مشهور یونیک مانند **y** را در روند **r** ایجاد کرده است. این الگوریتم **z** را به عنوان اولین **self-ancestor** **y** را پیدا می کند. فرض کنید **t** مهر زمانی باشد که آلیس که در هنگام ایجاد رویداد **z** ثبت کرده است. زمان دریافتی برای **x**، میانه یا **median** تمامی این مهرهای زمانی خواهد بود یعین برای تمامی ایجاد کنندگان شاهدین مشهور در روند **r**.

سپس نظم توافقی محاسبه می شود. تمامی رویدادها براساس روند دریافتی شان منظم می شوند. اگر دو رویداد روند دریافتی یکسانی داشته باشد، پس بر اساس زمانی دریافتی اش منظم می شوند.