



سیستم‌های توزیع شده

تمرین سری چهارم

استاد: دکتر کمندی

پارسا محمدپور

با تغییر و اصلاح الگوریتم ثبت دو مرحله‌ای^۱، الگوریتم جدیدی پیشنهاد دهید که وابستگی به یک هماهنگ‌کننده^۲ واحد را کاهش دهد. نحوه عملکرد الگوریتم خود را به صورت کامل شرح دهید. تاثیر این تغییر را بر پیچیدگی الگوریتم ارزیابی کنید.

برای حل این سوال ابتدا توضیح مختصری در رابطه با الگوریتم ثبت دو مرحله‌ای ارائه می‌دهیم.

الگوریتم ثبت دو مرحله‌ای:

این الگوریتم به این صورت است که فرآیند^۳ تصمیم‌گیری فرآیندها در حداکثر دو مرحله صورت می‌گیرد. روال کلی این الگوریتم در دو مرحله اجرا می‌شود:

در مرحله اول، تمامی فرآیندها به جز فرآیندی که به عنوان هماهنگ‌کننده انتخاب شده است، مقدار اولیه خودشان را برای فرآیند هماهنگ‌کننده ارسال می‌کنند و هر فرآیندی که مقدار اولیه آن برابر با یک است، بر روی مقدار یک تصمیم‌گیری می‌کند. فرآیندی که به عنوان هماهنگ‌کننده انتخاب شده است، این مجموعه مقادیر دریافتی را جمع می‌کند (و در یک بردار می‌ریزد) و اگر تمامی آن مقادیر دریافتی برابر با یک باشد، آن مقدار را به عنوان تصمیم نهایی اتخاذ می‌کند و سپس آن را به همه اعلام می‌کند و همه در این مرحله تصمیم یک را اتخاذ می‌کنند. اما اگر تمام مقادیر دریافتی برابر با یک نباشند، آن وقت فرآیند هماهنگ‌کننده مقدار صفر را به عنوان تصمیم نهایی در نظر می‌گیرد و سپس آن را به بقیه اعلام می‌کند. (حتی اگر فرآیندی هم در ارسال پیام در مرحله اول با شکست مواجه شود، یعنی قبل از ارسال پیام دچار مشکل شود و پیام را ارسال نکند، آن وقت باز هم تصمیم صفر را اتخاذ می‌کنیم).

در رابطه با پیچیدگی پیامی این الگوریتم هم باید بگوییم که در دو مرحله انجام می‌شود، پس پیچیدگی زمانی آن دو است. همچنین در رابطه با پیچیدگی پیامی هم باید بگوییم که پیچیدگی پیامی آن در حالتی که هیچ خطایی رخ ندهد، برابر با $2(n-1)$ می‌باشد.

حالا با توجه به گفته صورت سوال، اگر بخواهیم در این الگوریتم تغییری ایجاد کنیم تا وابستگی آن به یک هماهنگ‌کننده واحد کم شود، می‌توانیم این کار را انجام دهیم:

برای این مورد می‌توانیم بیشتر از یک هماهنگ‌کننده داشته باشیم. برای این منظور، مثلاً فرآیندهای یک و دو را به عنوان هماهنگ‌کننده‌ها انتخاب کنیم. سپس در مرحله اول تمام فرآیندها مقدار اولیه‌شان را برای هر دو فرآیند یک و دو ارسال می‌کنند سپس مشابه با الگوریتم عادی، اگر مقدارشان صفر است، تصمیم صفر را اتخاذ می‌کنند. سپس هر دو فرآیند یک و دو، مشابه الگوریتم عادی تصمیم نهایی را مشخص می‌کنند. (اگر از همه فرآیندها مقدار یک را دریافت کرده باشند تصمیم یک را می‌گیرند؛ در غیراینصورت (چه مقدار صفر دریافت کرده باشند چه دراصل هیچ مقداری دریافت نکرده باشند) تصمیم صفر را اتخاذ می‌کنند) سپس این تصمیم را به تمام فرآیندها انتقال می‌دهند. سپس، پس از پایان انتقال پیام‌ها، اگر همه فرآیندها پیام‌های دریافتی از هر دو فرآیند یک و دو یکسانی داشتند (و هنوز تصمیم نگرفته بودند) روی آن مقدار تصمیم می‌گیرند؛ در غیراینصورت بر روی مقدار صفر تصمیم می‌گیرند.

¹ Two-Phase Commit

² Coordinator

³ Process

در رابطه با پیچیدگی این الگوریتم باید گفت که مشابه با الگوریتم عادی، پیچیدگی زمانی آن همان **دو مرحله** باقی می ماند ولی از لحاظ پیچیدگی پیامی، تعداد پیام ها در حالتی که خطا نداشته باشیم (بیشترین مقدار تبادل پیام وجود داشته باشد) **دو برابر** حالت قبل می شود. یعنی پیچیدگی پیامی برابر با **$4(n-1)$** می شود.

یک تغییر برای الگوریتم ثبت سه مرحله‌ای^۱ پیشنهاد دهید که در شرایط بدون شکست^۲، فرآیندها بتوانند سریع تصمیم‌گیری کرده و متوقف شوند. الگوریتم شما باید دارای تعداد کمی دور و تعداد پیام از مرتبه n باشد. نحوه عملکرد الگوریتم خود را به صورت کامل شرح دهید. درستی این الگوریتم را اثبات کنید.

برای این کار می‌توانیم به اصلاحاتی به سراغ الگوریتم ثبت دو مرحله‌ای برویم. به طوریکه، هر فرآیندی که مقدار اولیه آن برابر با صفر است، بر روی صفر تصمیم بگیرد و آن مقدار را برای فرآیند هماهنگ‌کننده ارسال کند. سپس هماهنگ‌کننده به محض دریافت پیام صفر، بر روی صفر تصمیم‌گیری کند و آن را برای همه ارسال کند و بقیه به محض دریافت این پیام بر روی مقدار صفر تصمیم‌گیری کنند. (اگر فرآیندی که مقدار اولیه صفر را داشت همان هماهنگ‌کننده بود، باز هم تصمیم صفر را می‌گیرد و آن را برای همه ارسال می‌کند و در همان مرحله اول همه تصمیم صفر را اتخاذ می‌کنند.) در غیر اینصورت، اگر فرآیند هماهنگ‌کننده مقدار یک را از همه دریافت کرده بود (و همچنین مقدار خودش هم برابر با یک بود)، در مرحله بعدی آن مقدار را برای همه ارسال می‌کند و بر روی آن مقدار یک تصمیم می‌گیرد. در رابطه با پیچیدگی زمانی، باید بگوییم این الگوریتم در دو مرحله اجرا می‌شود. از لحاظ پیچیدگی پیامی هم، تعداد پیام‌های ردوبدل شده، از ارد n می‌باشد. چون در هر مرحله حداکثر n تا پیام ارسال می‌شود. (در حالتی که فرآیندی مقدار اولیه آن صفر باشد، پیام صفر را می‌فرستد و بر روی آن تصمیم‌گیری می‌کند و فرآیند هماهنگ‌کننده نیازی ندارد تا تصمیم نهایی که صفر است را دوباره برای آن ارسال کند. برای همین می‌شود این الگوریتم را طوری پیاده‌سازی کرد که کمتر از این تعداد پیام در یک سری حالات قابل انجام باشد، ولی حداکثر این تعداد پیام نیاز است.)

برای اثبات درستی این الگوریتم هم باید اثبات کنیم که هر سه مورد توافق^۴، قابل قبول بودن^۵ و خاتمه ضعیف^۶ رعایت می‌شوند. پس به ترتیب آن‌ها را اثبات می‌کنیم:

- توافق:

برای این مورد مشخص است. فرض می‌کنیم که یک فرآیند تصمیم یک را گرفته و دیگری تصمیم صفر را گرفته است سپس با توجه به اینکه در انتها به تناقض می‌رسیم، اثبات می‌شود که همچنین چیزی امکان ندارد و توافق برقرار است. برای اینکه این فرآیند تصمیم یک را بگیرد، باید از هماهنگ‌کننده مقدار یک را به عنوان تصمیم یک دریافت کرده باشد و همچنین فرآیند دیگر نیز برای اینکه تصمیم صفر را اتخاذ کند، یا باید مقدار اولیه خودش برابر با صفر بوده باشد یا مقدار صفر را از هماهنگ‌کننده شنیده باشد. که در هر دو صورت هم به تناقض می‌خوریم. چون اگر مقدار خودش برابر با صفر بوده باشد و تصمیم صفر را گرفته باشد، پس آن مقدار صفر را هم برای هماهنگ‌کننده ارسال کرده است و هماهنگ‌کننده چون پیام صفر را دریافت کرده است، پس نمی‌تواند تصمیمی جز صفر بگیرد طبق روند تصمیم‌گیری در الگوریتم پیشنهاد شده و نمی‌تواند به فرآیند دیگر که تصمیم یک را گرفته، مقدار یک را گفته باشد. اما اگر هم مقدار صفر را از هماهنگ‌کننده شنیده باشد، باز هم به صورت بدیهی به تناقض می‌خوریم چون هماهنگ‌کننده به یک فرآیند مقدار صفر را به عنوان تصمیم نهایی گفته است و به دیگری مقدار یک را به عنوان تصمیم نهایی گفته است. که باز هم به تناقض رسیدیم. چون اگر هماهنگ‌کننده مقداری برابر با صفر را دریافت کرده باشد و یا مقدار خودش برابر با صفر باشد تصمیم صفر را می‌گیرد و دیگر به کسی تصمیم یک را نمی‌گوید و اگر هم تصمیم یک را گرفته باشد، به این معنی است که از همه فرآیندها مقدار یک را

¹ Three-Phase Commit

² Failure-Free

³ $O(n)$

⁴ Agreement

⁵ Validity

⁶ Weak Termination

دریافت کرده است و مقدار خودش هم یک بوده است، پس هیچ جایی نمی توانسته تصمیم یک را برای فرآیندی ارسال کرده باشد. پس به تناقض می خوریم و دو فرآیند نمی توانند دو تصمیم متفاوت بگیرند. پس توافق برقرار است.

- قابل قبول بودن:

برای قابل قبول بودن هم مشخص است، اگر فرآیندی تصمیم صفر بگیرد، آن مقدار را برای هماهنگ کننده ارسال می کند و آن هم به محض دریافت آن مقدار تصمیم صفر را می گیرد و برای همه آن را ارسال می کنند و همه تصمیم صفر می گیرند. مخصوصا چون فرآیندها دچار خطا نمی شوند، پس مشکلی هم در ارسال و دریافت پیام و خطا دارد شدن فرآیند نیست. پس همه چیز مطابق انتظار پیش می رود. اما اگر مقدار همه فرآیندها برابر با یک باشد، هیچکدام از آن ها در مرحله یک تصمیم نمی گیرند و فقط مقدار پیام خودشان را برای فرآیند هماهنگ کننده ارسال می کنند. در انتهای این مرحله هم با توجه به اینکه فرآیند هماهنگ کننده مقدار یک را از همه فرآیندها دریافت کرده، تصمیم یک را اتخاذ می کند و برای همه آن را ارسال می کند و همه تصمیم یک می گیرند. پس قابل قبول بودن برقرار است. نکته کلیدی این است که همه پیام ها ارسال می شوند و هیچ فرآیندی دچار خطا نمی شود و تا انتها به کار خود ادامه می دهد.

- خاتمه^۱:

در این الگوریتم، تحت اگر فرآیندی مقدار صفر را به عنوان مقدار اولیه داشته باشد، در همان ابتدا تصمیم می گیرد و تصمیم آن صفر است و خاتمه می یابد. اما اگر مقدار آن یک باشد، ابتدا تصمیم را برای فرآیند هماهنگ کننده ارسال می کند. سپس هماهنگ کننده پس از دریافت تمام پیام ها از تمام فرآیندها و نحوه تصمیم گیری تصمیم می گیرد و آن تصمیم را برای همه ارسال می کند و خودش هم بعد از ارسال تصمیمی که اتخاذ کرده بود را می گیرد و خاتمه می یابد. فرآیندهای دیگر نیز با توجه به دریافت تصمیم هماهنگ کننده، تصمیم نهایی شان مشخص می شود و خاتمه می یابند. نکته کلیدی باز هم این است که چون نه خطای لینک داریک و نه خطای فرآیند، همه پیام ها ارسال می شود و مشکلی برای فرآیندی پیش نمی آید که مثلا تصمیم بگیرد ولی نتواند آن را برای کسی ارسال کند و یا ... و برخلاف الگوریتم ثبت دو مرحله ای، در اینجا چون سیستم بدون خطاست، پس امکان این که فرآیند هماهنگ کننده تصمیم گیری نکند یا پیامی را نتواند برای کسی ارسال کند وجود ندارد، پس هر دو شرط خاتمه ضعیف^۲ و خاتمه قوی^۳ را داریم.

¹ Termination

² Weak Termination

³ Strong Termination

الگوریتم فلدست^۱ و ثبت دو مرحله‌ای را در نظر بگیرید. از ترکیب این دو الگوریتم یک الگوریتم جدید برای تصمیم‌گیری ارائه دهید. توجه داشته باشید که الگوریتم باید شرایط مربوط به اجماع را داشته باشد. نحوه عملکرد الگوریتم خود را به صورت کامل شرح دهید و اثبات کنید که الگوریتم می‌تواند شرایط مربوط به اجماع را برآورده کند.

برای این سوال سه الگوریتم ارائه می‌دهیم. این الگوریتم‌ها به صورت زیر هستند:

• الگوریتم اول:

الگوریتم حاصل به این صورت است که در نظر می‌گیریم تمام هر فرآیند یک مقدار اولیه‌ای دارد (که این سری می‌تواند هر مقداری باشد و دیگر مصل مسئله ثبت دو مرحله‌ای منحصر به صفر و یک نیست) و همچنین یک فرآیند به عنوان هماهنگ‌کننده نیز داریم. در این الگوریتم، فرآیند هماهنگ‌کننده یک متغیر به نام W را در نظر می‌گیرد که یک مجموعه است و در ابتدای کار مقدار تصمیم خودش را در آن مجموعه قرار می‌دهد. سپس در مرحله اول، تمام فرآیندها مقدار اولیه‌شان را برای فرآیند هماهنگ‌کننده ارسال می‌کنند. سپس این فرآیند هماهنگ‌کننده مقدار دریافتی از سمت هر کدام از فرآیندها را در مجموعه خودش اضافه می‌کند. در انتهای این مرحله، وقتی هماهنگ‌کننده پیام تمام فرآیندها را پردازش کرد و کارش تمام شد، اگر تعداد اعضای مجموعه یکی بود (یعنی مقدار اولیه تمام فرآیندها و همچنین خود هماهنگ‌کننده یکسان بود)، روی آن مقدار تصمیم‌گیری صورت می‌گیرد و هماهنگ‌کننده آن مقدار را در مرحله بعدی برای همه ارسال می‌کند و در انتها هم خودش بر روی آن تصمیم می‌گیرد. در غیر این صورت (یعنی اگر تعداد اعضای مجموعه W در فرآیند هماهنگ‌کننده بیشتر از یکی بود) سپس مطابق با الگوریتم فلدست، تصمیم از پیش تایید شده^۲ اتخاذ می‌شود و فرآیند هماهنگ‌کننده مقدار تصمیم از پیش تایید شده را برای بقیه ارسال می‌کند و در انتها آن تصمیم را اتخاذ می‌کند.

برای اثبات اینکه این الگوریتم، شرایط مربوط به اجماع را برآورده می‌کند، باید اثبات کنیم که این الگوریتم توافق، قابل قبول بودن و Termination را دارد. در ادامه هر کدام از «ها» را یکی یکی اثبات می‌کنیم.

- توافق:

فرض می‌کنیم که دو تا فرآیند، بر روی دو مقدار مختلف تصمیم‌گیری کرده باشند. این یعنی اینکه باید فرآیند هماهنگ‌کننده برای آن‌ها، دو مقدار متفاوت را ارسال کرده باشد که این ممکن نیست؛ زیرا در الگوریتم تصمیم هماهنگ‌کننده بعد از دریافت تمام پیام‌ها اتخاذ می‌شد و در هیچ جایی هم تغییر نمی‌کرد، بنابراین ممکن نیست که هماهنگ‌کننده دو تا تصمیم متفاوت را برای دو فرآیند ارسال کرده باشد. پس به تناقض می‌خوریم. پس شرط توافق باید برقرار باشد.

- قابل قبول بودن:

در این الگوریتم، اگر تمام فرآیندها با مقدار تصمیم اولیه یکسان شروع به فعالیت کنند، چه دچار خطا بشوند چه دچار خطا نشوند، باز مقدار یکسانی را به هماهنگ‌کننده ارسال می‌کنند؛ یعنی هماهنگ‌کننده در هنگام تصمیم‌گیری یا پیام هر فرآیند را دریافت کرده (که همان مقدار یکسان گفته شده است) یا آن مقدار را دریافت نکرده و آن فرآیند دچار خطا شده است و اصلاً مقدارش را برای فرآیند هماهنگ‌کننده ارسال نکرده است و پس در هنگام تصمیم‌گیری، تنها مقدار موجود در مجموعه W موجود در هماهنگ‌کننده برابر با همان مقدار اولیه گفته شده است و طبق نحوه تصمیم‌گیری، آن مقدار به عنوان تصمیم اتخاذ می‌شود و به وسیله هماهنگ‌کننده برای همه فرآیندها ارسال می‌شود. پس تمام فرآیندها این مقدار را دریافت می‌کنند و روی آن تصمیم‌گیری انجام می‌دهند. پس تصمیم همه فرآیندها

¹ FloodSet

² Default

یکسان می‌شود. حالا اما اگر خود هماهنگ‌کننده قبل از ارسال همه یا تعدادی از پیام‌های تصمیم به بقیه فرآیندها دچار خطا شد چی؟ در این صورت باز هم شرایط برای قابل قبول بودن برقرار است زیرا در مورد شرط قابل قبول بودن ما این را داریم که "اگر همه فرآیندها با مقدار یکسان کار را شروع کنند، آن مقدار **تنها گزینه ممکن** برای تصمیم نهایی باید باشد" و طبق این تعریف، اگر فرآیندها تصمیم‌گیری نکنند مشکلی نیست، ولی اگر تصمیم‌گیری کنند، تنها تصمیم مورد قبول، تصمیم اولیه گفته شده است که در اینجا هم اثبات کردیم اگر تصمیم‌گیری توسط فرآیندها صورت بگیرد، بر روی همان مقدار است. پس قابل قبول بودن نیز برقرار است.

- خاتمه:

در این الگوریتم، در صورتی که فرآیند هماهنگ‌کننده همان فرآیندی باشد که دچار خطا می‌شود، ما به مشکل می‌خوریم و ممکن است که شرط خاتمه برقرار نشود. درست مانند حالتی که در ثبت دو مرحله‌ای داشتیم که ممکن بود فرآیند هماهنگ‌کننده درست بعد از تصمیم‌گیری و یا کلاً قبل از فرستادن پیام برای دیگر فرآیندها دچار خطا شود و این ممکن بود باعث این شود که کلاً فرآیندها تصمیم‌گیری نکنند، اینجا هم دقیقاً در همین حالت باعث می‌شود که فرآیندها نتوانند تصمیم‌گیری کنند و خاتمه برای آنها اتفاق نیفتد. اما اگر فرآیند هماهنگ‌کننده دچار خطا نشود، باعث می‌شود که ما شرط خاتمه را داشته باشیم. فرآیندهایی که خطادار نیستند، حتماً در انتهای کار، پیامی را از طرف فرآیند هماهنگ‌کننده دریافت می‌کنند (چون در این حالت در نظر گرفتیم که هماهنگ‌کننده دچار خطا نمی‌شود) پس پیامی دریافت می‌کنند و یا دچار خطا می‌شوند، که خوب باز برای ما مهم نیستند چون شرط خاتمه برای فرآیندهای بدون خطا بود، یا دچار خطا نمی‌شوند که خوب در این صورت طبق روال الگوریتم، تصمیمی که برای‌شان ارسال شده است را به عنوان تصمیم نهایی اتخاذ می‌کنند.

• الگوریتم دوم:

این الگوریتم به این صورت است که تعداد $f+1$ تا هماهنگ‌کننده در نظر می‌گیریم. هر فرآیند در مرحله اول، پیام خودش را به تمام این هماهنگ‌کننده‌ها ارسال می‌کند. در مرحله بعدی، تمام هماهنگ‌کننده‌ها با توجه به پیام‌های دریافتی تصمیم‌گیری می‌کنند (مشابه با تصمیم‌گیری در حالت قبلی که اگر مجموعه W شان فقط شامل یک عضو بود، آن را به عنوان تصمیم لحاظ می‌کنند وگرنه مقدار از پیش تایید شده را به عنوان تصمیم لحاظ می‌کنند) سپس در مرحله بعدی آن را برای تمام فرآیندها ارسال می‌کنند. سپس هر فرآیند اگر در مجموعه‌های دریافتی‌اش از تمام هماهنگ‌کننده‌هایی که پیامشان بهش رسیده است، شامل یک مقدار بود، آن را به عنوان تصمیم لحاظ می‌کنند در غیر این صورت، مقدار از پیش تایین شده را به عنوان تصمیم در نظر می‌گیرند. این الگوریتم بیشتر شبیه ترکیب الگوریتم ثبت دو مرحله‌ای و فلا دست است. می‌شد البته این را هم به نوعی تغییر داد که ترکیبی از الگوریتم صبت سه مرحله‌ای و فلا دست باشد که در الگوریتم بعدی به آن می‌پردازیم.

• الگوریتم سوم:

در این الگوریتم مطابق مطلب گفته شده در انتهای الگوریتم قبلی، الگوریتم سه مرحله‌ای را با فلا دست ترکیب می‌کنیم. بدین صورت که الگوریتم در سه فاز اجرا شود. این فازها به صورت زیر هستند:

- فاز اول:

برای این فاز بدین صورت عمل می‌کنیم که $f+1$ تا هماهنگ‌کننده در نظر می‌گیریم. تمام فرآیندها باید در این فاز که تنها شامل یک مرحله است، مقدار اولیه خودشان را برای تمام فرآیندهای هماهنگ‌کننده (که تعدادشان برابر با $f+1$ است) ارسال کنند. (حتی خود فرآیندهای هماهنگ‌کننده هم باید مقدار اولیه خودشان را برای دیگر فرآیندهای هماهنگ‌کننده ارسال کنند)

- فاز دوم:

در این فاز بین تمام $f+1$ تا فرآیند هماهنگ کننده که در مرحله قبل پیام‌هایی از جانب تمام فرآیندها دریافت کردند، الگوریتم فلا دست را اجرا می‌کنیم. (اجرای الگوریتم فلا دست در $f+1$ مرحله است که در هر مرحله آن، تمام فرآیندهای هماهنگ کننده تمام تصمیمات دریافتی‌شان را با یکدیگر به اشتراک می‌گذارند.) در این الگوریتم، تبادل پیام‌ها مطابق با الگوریتم فلا دست است که در آن هر فرآیند، یک مجموعه W را نگه می‌دارد که شامل تمام مقادیر دریافتی‌اش است. سپس در انتهای اجرای الگوریتم فلا دست برای این فرآیندهای هماهنگ کننده، این فرآیندها به سراغ تصمیم‌گیری می‌روند. نحوه تصمیم‌گیری آن‌ها نیز به صورت الگوریتم فلا دست است. یعنی اگر مجموعه W فرآیندها، مجموعه‌ای بود که فقط شامل یک عضو بود، آن عضو به عنوان تصمیم نهایی اتخاذ می‌شود؛ در غیر این صورت، تصمیم از پیش تایین شده را به عنوان تصمیم نهایی در نظر می‌گیرند.

- فاز سوم:

در این فاز تمام فرآیندهای هماهنگ کننده که هنوز دچار خطا نشدند، (از جایی که تعداد این فرآیندها برابر با $f+1$ است، پس قطعاً در این مرحله حداقل یک فرآیند هماهنگ کننده وجود دارد که تصمیم نهایی را در مرحله قبلی مشخص کرده و دچار خطا نشده است) تصمیم‌شان را برای تمام فرآیندهای دیگر (فرآیندهایی که هماهنگ کننده نیستند) ارسال می‌کنند. سپس فرآیندها با دریافت این مقدار، تصمیم‌گیری می‌کنند.

این الگوریتم نیز مشابه با الگوریتم اول، شرایط توافق، قابل قبول بودن و خاتمه را دارد. برای هر مورد اثبات کوتاهی ارائه می‌دهیم:

- توافق:

در این الگوریتم، چون در فاز دوم، الگوریتم فلا دست اجرا شده است، پس تمام هماهنگ کننده‌هایی که تا پایان آن مرحله دچار خطا شده‌اند، تصمیم یکسانی دارند یا به عبارتی مجموعه W آن‌ها یکسان است. (مطابق با الگوریتم فلا دست) سپس وقتی در مرحله آخر این تصمیم‌گیری را انجام می‌دادند (در الگوریتم فلا دست)، مقدار تصمیم‌شان یکسان می‌شود. پس وقتی در مرحله آخر آن را برای بقیه ارسال می‌کنند، فرآیندهای دیگر همه‌شان یک مقدار یکسان را از تمام هماهنگ کننده‌ها دریافت می‌کنند و تصمیم یکسانی می‌گیرند. پس شرط توافق به درستی رعایت می‌شود.

- قابل قبول بودن:

شرط قابل قبول بودن به آن صورت است که اگر همه با مقدار یکسانی اجرای الگوریتم را شروع کنند، در هر مرحله در هر فرآیند هماهنگ کننده، تنها همان مقدار یکسان را در مجموعه W خودش دارد و هر سری مقدار ورودی‌ای که دریافت می‌کند، همان مقدار است، پس در تمام مراحل، مجموعه W تنها شامل همان یک عضو باقی‌می‌ماند. پس در مرحله تصمیم‌گیری هم همان مقدار به عنوان تصمیم لحاظ می‌شود. پس اگر تمام فرآیندها با یک مقدار اولیه یکسان شروع به اجرای الگوریتم کنند، آن مقدار در انتها به عنوان تصمیم اتخاذ می‌شود. پس در این الگوریتم، شرط قابل قبول بودن، رعایت می‌شود.

- خاتمه:

در این الگوریتم، چون در انتهای فاز دوم که الگوریتم فلا دست تا انتها اجرا می‌شود و با توجه به اینکه تعداد هماهنگ کننده‌ها برابر با $f+1$ است (که f تعداد فرآیندهایی است که تا انتها ممکن است دچار خطا شوند) پس در بدترین حالت هم، باز حداقل یک فرآیند

هماهنگ‌کننده باقی می‌ماند که دچار خطا نشده باشد. پس در فاز آخر، حداقل یک فرآیند هماهنگ‌کننده باقی‌مانده است که مقدار تصمیم‌نهایی را برای بقیه ارسال کند. پس وقتی در مرحله آخر، تمام فرآیندها (تمام فرآیندهای باقی‌مانده) مقدار تصمیم را حداقل از یک فرآیند هماهنگ‌کننده دریافت می‌کنند و روی آن مقدار تصمیم می‌گیرند پس تمام فرآیندها در انتها تصمیم‌گیری می‌کنند (البته اگر دچار خطا نشوند) پس در این الگوریتم شرط خاتمه، رعایت می‌شود.

هر کدام از این سه تا الگوریتم گفته شده، مزایا و معایبی نسبت به یکدیگر دارند که باید با توجه به شرایط از یکی از آنها استفاده کرد. کثلاً در یکی زمان کمتری صرف اجرای الگوریتم می‌شود، یعنی در زمان سریع‌تری تصمیم‌گیری صورت می‌گیرد. (در تعداد مرحله کمتر) در یکی پیچیدگی پیامی کمتر است که باعث کاهش حجم فشار^۱ روی شبکه می‌شود. دیگری پیام بیشتر و زمان بیشتری دارد ولی با توجه به این دو مورد، شبکه قابل اطمینان‌تر^۲ است. یعنی احتمال اینکه تصمیم یک فرآیندی که دچار خطا می‌شود هم لحاظ شود، بیشتر می‌شود.

^۱ Load

^۲ More reliable