سیستم‌های توزیع شده

تمرین سری چهارم

استاد: دکتر کمندی

پارسا محمدپور

پاییز ۱۴۰۳

پرسش یک

با تغییر و اصلاح الگوریتم ثبت دو مرحله‌ای[1]، الگوریتم جدیدی پیشنهاد دهید که وابستگی به یک هماهنگ‌کننده[2] 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[3] تصمیم‌گیری فرآیندها در حداکثر دو مرحله صورت می‌گیرد. روال کلی این الگوریتم در دو مرحله اجرا می‌شود:

در مرحله اول، تمامی فرآیندهابه جز فرآیندی که به عنوان هماهنگ‌کننده انتخاب شده‌است، 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 شده‌است، 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 درنظر 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 این مورد می‌توانیم **بیشتر از یک هماهنگ‌کننده** داشته باشیم. برای این منظور، مثلا فرآیندهای یک و دو را به عنوان هماهنگ‌کننده ها 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 باشن تصمیم یک را می‌گیرند؛ در غیراینصورت (چه مقدار صفر دریافت کرده باشند چه دراصلا 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 الگریتم باید گفت که مشابه با الگوریتم عادی، پیچیدگی زمانی آن همان **دو مرحله** باقی می‌ماند ولی از لحاظ پیچیدگی پیامی، تعداد پیام‌ها در حالتی که خطا نداشته باشیم (بیشترین مقدار تبادیلپیام وجود داشته باشد) **دو برابر** حالت قبل می‌شود. یعنی پیچیدگی پیامی برابر با **4(n-1)** می‌شود.

پرسش دو

یک تغییر برای الگوریتم ثبت سه مرحله‌ای[1] پیشنهاد دهید که در شرایط بدون شکست[2]، 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 اردر n [3] 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[4]، قابل‌قبول‌بودن[5] و خاتمه ضعیف[6] 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 همچین 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 نمی‌توانسه 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 هیچ‌جایی نمی‌توانسته تصمیم یک را برای فرآیندی ارسال کرده باشد. پس به تناقض می‌خوریم و دو فرآیند نمی‌توانند دو تصمیم متفاوت بگیرند. پس توافق برقرار است.

- **قابل‌قبول‌بودن**:

برای قابل‌قبول‌بودن هم مشخص است، اگر فرآیندی تصمیم صفر بگیرد، آن مقدار را برای هماهنگ‌کننده ارسال می‌کند و آن هم به محض دریافت آن مقدار تصمیم صفر را می‌گیرد و برای همه آن را ارسال می‌کنند و همه تصمیم صفر می‌گیرند. مخصوصا چون فرآیندها دچار خطا نمی‌شوند، پس مشکلی هم در ارسال و دریافت پیام و خطادارد شدن فرآیند نیست. پس همه چیز مطابق انتظار پیش می‌رود. اما اگر مقدار همه فرآیندها برابر با یک باشد، هیچکدام از آن‌ها در مرحله یک تصمیم نمی‌گیرند و فقط مقدار پیام خودشان را برای فرآیند هماهنگ‌کننده ارسال می‌کنند. در انتهای این مرحله هم با توجه به اینکه فرآیند هماهنگ‌کننده مقدار یک را از همه فرآیندها دریافت کرده، تصمیم یک را اتخاذ می‌کند و برای همه آن را ارسال می‌کند و همه تصمیم یک می‌گیرند. پس قابل‌قبول‌بودن برقرار است. نکته کلیدی این است که همه پیام‌ها اراسل می‌شوند و هیچ فرآیندی دچار خطا نمی‌شود و تا انتها به کار خود ادامه می‌دهد.

- **خاتمه**[1]:

در این الگوریتم، تحت اگر فرآیندی مقدار صفر را به عنوان مقدار اولیه داشته باشد، در همان ابتدا تصمیم می‌گیرد و تصمیم آن صفر است و خاتمه می‌یابد. اما اگر مقدار آن یک باشد، ابتدا تصمیم را برای فرآیند هماهنگ‌کننده ارسال می‌کند. سپس هماهنگ‌کننده پس از دریافت تمام پیام‌ها از تمام فرآیندها و نحوه تصمیم‌گیری تصمیم می‌گیرد و آن تصمیم را برای همه ارسال می‌کند و خودش هم بعد از ارسال تصمیمی که اتخاذ کرده بود را می‌گیرد و خاتمه می‌یابد. فرآیندهای دیگر نیز با توجه به دریافت تصمیم هماهنگ‌کننده ، تصمیم نهایی‌شان مشخص می‌شود و خاتمه می‌یابند. نکته کلیدی باز هم این است که چون نه خطای لینک داریک و نه خطای فرآیند، همه پیام‌ها ارسال می‌شود و مشکلی برای فرآیندی پیش نمی‌آید که مثلا تصمیم بگیرد ولی نتواند آن را برای کسی ارسال کند و یا ... . و برخلاف الگوریتم ثبت دو مرحله‌ای، در اینجا چون سیستم بدون خطاست، پس امکان این که فرآیند هماهنگ‌کننده تصمیم‌گیری نکند یا پیامی را نتواند برای کسی ارسال کند وجود ندارد، پس هر دو شرط خاتمه ضعیف[2] و خاتمه قوی[3] را داریم.

[1] Termination
[2] Weak Termination
[3] Strong Termination

پرسش سه

الگوریتم فلادست[1] 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 دو 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 یه 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 فلادست، تصمیم از پیش تایید شده[2] 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، قابل‌قبول‌بودن و Termination را دارد. در ادامه هر کدام از ا«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.

- **قابل‌قبول‌بودن**:

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 تصمیم گیری یا پیام هر فرآیند را دریافت کرده (که همان مقدار یکسان گفته شده است) یا آن مقدار را دریافت نکرده و آنن فرآیند دچار خطا شده است و اصلا مقدارش را برای فرآیند هماهنگ‌کننده ارسال نکرده استو 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 قابل‌قبول‌بودن برقرار است زیرا در مورد شرط قابل‌قبول‌بودن ما این را داریم که "اگر همه فرآیندها با مقدار یکاسن 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 قابل‌قبول‌بودن نیز برقرار است.

- **خاتمه**:

در این الگوریتم، درصورتی 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 کلا قبل از فرستادن پیام برای دیگر فرآیندها دچار خطا شود و این ممکن بود باعث این شود که کلا فرآیندها تصمیم‌گیری نکنند، اینجا هم دقیقا 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، حتما در انتهای کار، پیامی را از طرف فرآیند هماهنگ‌کننده دریافت می‌کنند (چون در این حالت درنظر گرفتیم که هماهنگ‌کننده دچار خطا نمی‌شود) پس پیامی دریافت می‌کنند و یا دچار خطا می‌شوند، که خب باز برای ما مهم نیستند چون شرط خاتمه برای فرآیندهای بدون خطا بود، یا دچار خطا نمی‌شوند که خب در این صورت طبق روال الگوریتم، تصمیمی که برای‌شان ارسال‌شده است را به عنوان تصمیم نهایی اتخاذ می‌کنند.

- **الگوریتم دوم:**

این الگوریتم به این صورت است که تعداد $f+1$ تا هماهنگ‌کننده درنظر 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 فلادست است. می‌شد البته این را هم به نوعی تغییر داد که تمام فرآیندها مقدارشان را برای تمام $f+1$ تا فرآیند هماهنگ‌کننده بفرستند و سپس آن $f+1$ تا فرآیند هماهمگ‌کننده، بیایند و بین خودشان الگوریتم فلادست را اجرا کنند و سپس در انتها همه این فرآیندهای هماهنگ‌کننده باقی‌مانده، تصمیم نهاییشان را برای تمام فرآیندها ارسال کنند و آن‌ها مطابق با روش سری قبلی (اگر اجتماع تمام مجموعه‌های ورودیتنها یک عضو داست، آن عضو تصمیم نهایی می‌شود؛ درغیر اینصورت تصمیم نهایی برابر با تصمیم از پیش تایین شده است) تصمیم‌گیری کنند. (البته در این روش نیازی نبود تمام مجموعه‌های ورودی را اجتماع بگیرند، می‌توانستند تنها از پیام یکی هم استفاده کنند. البته این کار که از پیام همه استفاده کنند، باز تا حدی ممکن است حالتی که یک فرآیند هماهنگ‌کننده دچار خطا شود و بازگردد و ادامه بدهد را هم پوشش بدهد.)