طراحی روشی برای افزودن امکان تشخیص خطا در رأی‌گیرنده‌های نرم‌افزاری برای سیستم‌های نهفته بحرانی-ایمن

محمدرضا رضائی[1]، یاسر صداقت[2]
آزمایشگاه سیستم‌های نهفته توزیع‌شده اتکاپذیر(DDEmS)
دانشکده مهندسی کامپیوتر، دانشگاه فردوسی مشهد، مشهد، ایران
mohammadreza.rezaee@stu.um.ac.ir[1], y_sedaghat@um.ac.ir[2]

چکیده- سیستم‌های بحرانی-ایمن نظیر سیستم کنترل پرواز هواپیما و یا سیستم کنترل قطارهای سریع‌السیر، سیستم‌هایی هستند که رخداد خرابی در آنها می‌تواند منجر به فاجعه جانی، مالی یا زیست‌محیطی شود. لذا جلوگیری از رخداد هرگونه خرابی در این سیستم‌ها یک نیاز اساسی محسوب می‌گردد. یکی از راه‌های موجود برای عدم رخداد خرابی در این سیستم‌ها استفاده از تکنیک‌های پوشش اشکال می‌باشد. برای پوشش اشکالات موجود در یک سیستم روش‌های چندنسخه‌ای نظیر NVP و NMR به صورت گسترده‌ای مورد استفاده قرار می‌گیرند. در این روش‌ها، تعداد N ماژول بر روی داده‌های یکسان عملیات مشابهی را انجام می‌دهند و هر یک نتیجه‌ای را تولید می‌نمایند. سپس نیاز است تا یک واحد رأی‌گیرنده بتواند از میان نتایج تولید شده یک نتیجه‌ی صحیح نهایی را استخراج نماید. از آنجایی که ورودی‌های یک رأی‌گیرنده ممکن است اعداد غیردقیقی باشند که رأی‌گیرنده آنها را به صورت غیرهمزمان دریافت می‌کند، این نیاز وجود دارد تا رأی‌گیرنده‌ای طراحی گردد که بتواند از بین ورودی‌های صحیح و غلط دریافت شده، بهترین جواب را در کمترین زمان ممکن تولید کند. از آنجا که خرابی واحد رأی‌گیرنده می‌تواند منجر به خرابی سیستم شود، بنابراین باید به ایمنی و دسترس‌پذیری آن نیز توجه ویژه‌ای داشت. در روش پیشنهادی با استفاده از تکنیک‌های مختلف سعی بر آن شده است تا امکان تشخیص خطا به رأی‌گیرنده‌هایی که این امکان را ندارند افزوده شود که نتیجه آن افزایش ایمنی رأی‌گیرنده بوده است.

کلمات کلیدی: پوشش اشکال، برنامه‌نویسی چندنسخه‌ای، رأی‌گیرنده‌ی نرم‌افزاری، سیستم‌های بحرانی-ایمن

1- مقدمه

سیستم‌های بحرانی-ایمن[1] سیستم‌هایی هستند که رخداد خرابی[2] در آنها می‌تواند منجر به رخداد فاجعه‌ی جانی، مالی یا زیست محیطی شود [1]. این سیستم‌ها نیازمند اتکاپذیری[3] هستند و یک سیستم اتکاپذیر خود نیازمند قابلیت اطمینان[4] بالا می‌باشد. قابلیت اطمینان تابعی از زمان است که به صورت احتمال اینکه سیستم در بازه‌ی زمانی $[t_0 , t_1]$ وظایف خود را به درستی انجام دهد با این شرط که در زمان t_0 درست کار کرده باشد، تعریف می‌شود [2]. در صورتی که سیستم نتواند وظایف محوله خودش را به درستی انجام دهد در آن سیستم خرابی رخ داده است [3]. برای اجتناب[5] از خرابی نباید خطایی[6] در سیستم رخ‌دهد و برای اینکه خطا در سیستم رخ ندهد باید اثر اشکالات[7] موجود در سیستم پوشانده[8] شود. استفاده از افزونگی[9] یکی از روشهای پوشش اشکال است که شامل NVP[10] و NMR[11] می‌شود. برنامه‌های زیادی وجود دارند که در صورت رخداد خرابی در آنها احتمال رخداد فاجعه در آنها زیاد است. فشارهای اقتصادی، اجتماعی و روانی قابل توجهی برای کاهش احتمال

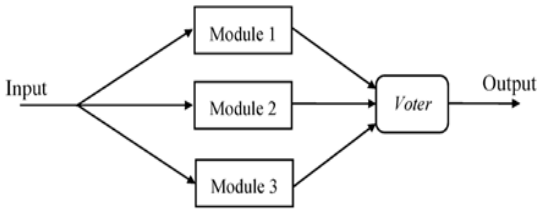
[1] Safety-Critical Systems
[2] Failure
[3] Dependability
[4] Reliability
[5] Avoidance
[6] Errors
[7] Faults
[8] Mask
[9] Redundancy
[10] N-Version Programming
[11] N-Modular Redundant

رفتارهای پرخطر سیستم‌ها، مخصوصاً در بخش‌های صنعتی زیر وجود دارد:

- ✓ حمل و نقل (هواپیما، خطوط راه‌آهن و خودروها)
- ✓ فرآیندهای کنترلی (در مکان‌هایی که مواد سمی، آتش‌زا و یا منفجره مورد استفاده قرار می‌گیرند)
- ✓ ایستگاه‌های انرژی اتمی و برنامه‌های نظامی
- ✓ سیستم‌های پایگاه داده‌ای و محاسباتی

در این برنامه‌ها، هدف، افزایش اتکاپذیری سیستم (شامل قابلیت اطمینان، ایمنی[12] و دسترس‌پذیری[13]) می‌باشد.

پوشش اشکال، یکی از راهکارهای اولیه برای تثبیت و یا بهبود رفتار عادی سیستم در محیط‌های پر خطر می‌باشد [4]، [5]. ساده ترین حالت NMR که در شکل 1 نمایش داده‌شده است، TMR[14] می‌باشد که در آن سه ماژول مشابه، عملیات یکسانی را بر روی داده‌های یکسان انجام می‌دهند و نتیجه را به رأی‌گیرنده منتقل می‌کنند [6]. رأی‌گیرنده با استفاده از نتایج بدست آمده، نتیجه‌ی واحدی را به عنوان خروجی مشخص می‌نماید.



شکل 1 سیستم TMR

رأی‌گیرنده‌ها را می‌توان به دو صورت سخت‌افزاری و نرم‌افزاری پیاده‌سازی کرد. هرکدام دارای معایب و مزایایی می‌باشند. ماهیت و ساختار سیستم تحمل‌پذیر اشکال مشخص می‌کند که کدام یک از رأی‌گیرنده‌های نرم‌افزاری یا سخت‌افزاری مورد استفاده قرار گیرند. رأی‌گیری در سطح حسگرها نیازمند رأی‌گیرنده‌های سخت‌افزاری است، در حالی که داوری در مورد خروجی‌های ماژول‌های نرم‌افزاری مختلف و افزونه، به احتمال زیاد نیازمند رأی‌گیرنده‌های نرم‌افزاری می‌باشد.

اگر در یک سیستم، فقط از یک رأی‌گیرنده استفاده کنیم، تک نقطه‌ی خرابی[15] برای کل سیستم به‌وجود می‌آید. راه‌حل‌هایی با استفاده از رأی‌گیرنده‌های توزیع شده، قابلیت اطمینان بیشتری را به وجود می‌آورند، ولی پیچیدگی سیستم را به دلیل نیاز به لینک‌های ارتباطی بین ماژول‌ها و رأی‌گیرنده‌های توزیع‌شده و همچنین سربار تعاملات، بیشتر می‌کنند [7]. برای انجام این راه حل مثلا در یک TMR از سه رأی‌گیرنده استفاده می‌شود که در مجموع سه خروجی تولید می‌کنند، در حالی که فقط به یک خروجی نیاز می‌باشد. برای این منظور در طول عملیات، یکی از رأی‌گیرنده‌ها را به عنوان رأی‌گیرنده‌ی اولیه تعریف کرده تا خروجی آن، مستقیما خروجی کل سیستم را مشخص کند. رأی‌گیرنده‌های باقیمانده و یدکی[16]، می‌توانند کاملا خاموش باشند یا در اصطلاح در حالت آماده به‌کار سرد[17] قرار گیرند. می‌توانند همانند ماژول‌های فعال اجرا شوند تا بتوان در صورت نیاز با کمترین تاخیر آن‌ها را وارد سیستم کرد که به این حالت نیز آماده به‌کار داغ[18] گفته می‌شود. رأی‌گیرنده‌های یدکی می‌توانند در حالتی بین دو حالت قبل نیز کار کند که به این حالت آماده به‌کار گرم[19] گفته می‌شود [8].

2- پیشینه

رأی‌گیرنده‌ها را می‌توان به شیوه‌های مختلفی طبقه‌بندی کرد. در ادامه این بخش انواع رأی‌گیرنده‌های انتخاب‌کننده‌ی نتایج[20]، ادغام کننده‌ی نتایج[21]، رأی‌گیرنده‌های مخلوط و پیش‌نگر را مورد بررسی قرار می‌گیرد.

رأی‌گیرنده‌ی همرای[22] زمانی اقدام به تولید خروجی می‌نماید که تمامی ورودی‌هایش موافق و قابل قبول باشند. این تکنیک خیلی سخت‌گیرانه عمل می‌کند و در برنامه‌هایی که رسیدن به توافق روی تمامی ورودی‌ها به شدت حیاتی است، به کار می‌رود. این رأی‌گیرنده هیچ گونه اشکالی را پوشش نمی‌دهد.

رأی‌گیرنده‌ی اکثریت، خروجی را از میان n ورودی‌اش تولید می‌کند، به شرطی که حداقل $\lceil \frac{n+1}{2} \rceil$ ورودی‌ها با یکدیگر

[12] Safety
[13] Availability
[14] Triple Modular Redundant System

[15] Single Point of Failure
[16] Spare
[17] Cold Standby
[18] Hot Standby
[19] Warm Standby
[20] Result Selection Voters
[21] Result Amalgamation Voters
[22] Unanimity Voter

موافق باشند. **رأی‌گیرنده‌ی جمع‌گرا** نسخه‌ی ساده‌تر از رأی‌گیرنده‌ی اکثریت می‌باشد و رای‌دهندگی m از n را پیاده‌سازی می‌کند (برای مثال رأی‌گیری 2 از 5 یا 3 از 7). رای‌دهندگان جمع‌گرا به طور معمول دارای تعداد فرد ماژول میانی می‌باشند، بنابراین در رأی‌گیرنده حالت برابری میان نتایج به‌وجود نمی‌آید. با سه متغیر، رای‌دهندگان اکثریت و جمع‌گرا مشابه یکدیگر عمل می‌کنند. با این حال رأی‌گیرنده‌ی جمع‌گرا در موافقت‌ها، هیچ‌گاه بدتر از رأی‌گیرنده‌ی اکثریت عمل نمی‌کند [9]. هرگاه مقداری دارای اکثریت باشد، دارای جمع‌گرایی نیز هست و بنابراین هرگاه یک استراتژی اکثریت موفقیت‌آمیز باشد، استراتژی جمع‌گرایی نیز موفق خواهد بود.

رأی‌گیرنده‌ی میانه، مقدار میانه‌ی ورودی‌هایش را انتخاب می‌نماید. فلسفه‌ی اصلی پشت این رویکرد، علاوه بر انتخاب سریع، این است که رأی‌گیرنده قادر به کنترل حالتی است که چندین پاسخ صحیح وجود دارد و احتمال این‌که مقدار انتخاب‌شده توسط رأی‌گیرنده، صحیح باشد و یا حداقل در بازه‌ی قابل قبول قرار داشته باشد، زیاد است. لازم به ذکر است که رای‌دهندگان اکثریت و جمع‌گرا نمی‌توانند چندین خروجی صحیح را مدیریت کنند. برای درک بیشتر، در نظر بگیرید برای یک ورودی خاص، ماژول‌ها می‌بایست معادله‌ی $x^2 + 7x + 10 = 0$ را حل کنند، در حالی که $x=2$ یا $x=5$ پاسخ‌های صحیح می‌باشند. حال فرض کنید که رأی‌گیرنده مجموعه‌ی (1 2 5) را از ماژول‌های افزونه دریافت می‌کند. رأی‌گیرنده‌ی اکثریت برای این مجموعه نمی‌تواند خروجی تولید کند و یک کد خطا ایجاد می‌نماید، در حالی که رأی‌گیرنده‌ی میانه مقدار 2 را به عنوان خروجی انتخاب می‌کند که مقداری صحیح می‌باشد [6].

رأی‌گیرنده‌ی میانه ذاتاً قابلیت کشف اشکال ندارد که باعث می‌شود برای برنامه‌های بحرانی- ایمن ناکارآمد باشد، مگر این‌که با مکانیزم‌های کشف اشکال ترکیب شود. در حالت کلی یک رأی‌گیرنده‌ی میانه می‌تواند نتیجه‌ی صحیح را در صورت وجود حداکثر $\lfloor \frac{n-1}{2} \rfloor$ ورودی اشکال‌دار تولید نماید. به سادگی می‌توان نشان داد که خروجی رأی‌گیرنده‌ی میانه هیچ‌گاه بدترین مقدار نمی‌باشد. بدترین مقدار، مقداری است که خطا در آن بیشترین اندازه را دارد.

رأی‌گیرنده‌ی میانگین، مقدار میانگین ورودی‌هایش را به عنوان خروجی تولید می‌کند. این رأی‌گیرنده معمولا به عنوان رأی‌گیرنده‌ی خروجی یک سیستم TMR توزیع‌شده استفاده می‌شود. در چنین سیستمی، رأی‌گیری، تک نقطه‌ی خرابی ایجاد نمی‌کند، چراکه هر ماژول رأی‌گیرنده‌ی خاص خود را دارد و میانگین مقادیر تولیدشده توسط هر ماژول، به خروجی داده می‌شود.

رأی‌گیرنده‌ی میانگین وزن‌دار، مقدار متوسط وزنی نتایج ماژول‌ها را محاسبه می‌نماید. وزن‌ها می‌توانند از قبل مشخص شوند و یا به‌صورت پویا تنظیم شوند. مقادیر از پیش تعیین‌شده می‌توانند بر اساس تخمین‌هایی از قابلیت اطمینان ماژول‌ها باشند و یا براساس احتمال خرابی ماژول‌های افزونه تعیین شوند [10]. تنظیمات پویا نیز می‌توانند بر اساس اطلاعات شناخته‌شده مرتبط با ماژول‌ها و یا فاصله‌ی بین ورودی‌ها باشند [11]. سپس وزن‌های محاسبه شده w_i توسط یک الگوریتم میانگین وزنی $x_o = \frac{\sum w_i x_i}{\sum w_i}$، برای محاسبه‌ی خروجی رأی‌گیرنده، مورد استفاده قرار می‌گیرند. در این رابطه x_i بیان‌گر ورودی‌های رأی‌گیرنده و x_o بیان‌گر خروجی‌های رأی‌گیرنده می‌باشند. واضح است که کارایی نسبی نسخه‌های مختلف رای‌دهندگان میانگین وزنی، تابعی از وزن‌های مورد استفاده می‌باشد. این رای‌دهندگان نیز ذاتاً توانایی کشف اشکال را ندارند.

عیب رای‌دهندگان اکثریت و جمع‌گرا این بود که ممکن است در صورت توافق روی ورودی‌های نادرست یکسان، خروجی نادرستی را ارائه دهند. به عبارت دیگر، این‌گونه رای‌دهندگان قادر به تمیز دادن میان ورودی‌های مورد توافق درست و نادرست نمی‌باشند. هنگامی که فضای خروجی ماژول‌ها کوچک است، امکان دارد نسخه‌های متعدد نرم‌افزاری، خروجی یکسان و نادرستی را تولید کنند. به عنوان مثال، در فضای خروجی باینری، همه‌ی نتایج خروجی‌های نادرست یکسانند، در نتیجه ممکن است رای‌دهندگان جمع‌گرا و اکثریت روی خروجی‌های نادرست به نتیجه برسند و عواقب فاجعه بار ایجاد نمایند.

گروهی از رای‌دهندگان، متفاوت با رای‌دهندگان عمومی، از اطلاعات اضافی مانند سطوح اعتماد ماژول‌ها، اطلاعات تشخیص on-line ماژول‌ها یا اطلاعات احتمالی مختلف برای بهبود کارایی رأی‌گیری استفاده می‌کنند. به این‌گونه رای‌دهندگان، رأی‌گیرندهای مخلوط گفته می‌شود. اگر چنین اطلاعاتی وجود داشته باشد، این رای‌دهندگان می‌توانند خروجی‌های دقیق‌تری نسبت به الگوریتم‌های رأی‌گیری عمومی تولید نمایند.

برنامه‌های کنترلی توکار عموماً سیستم‌هایی چرخه‌ای می‌باشند که بین نتایج هر چرخه با چرخه‌ی بعدیشان روابطی

وجود دارد. دانش روابط بین نتایج متوالی در **رای‌دهندگان پیش‌نگر**[۲۳] برای تولید نتایج در هنگام عدم توافق مورد استفاده قرار می‌گیرد. هنگامی که میان ورودی‌ها ناسازگاری تشخیص داده می‌شود، تاریخچه‌ی نتایج قبلی رأی‌گیرنده برای تولید نتیجه‌ی مورد انتظار استفاده می‌شود. نتیجه‌ی مورد انتظار با تک‌تک ورودی‌ها مقایسه می‌شود تا انتخاب صورت گیرد. هر ورودی که فاصله‌ی آن با نتیجه‌ی پیش‌بینی‌شده کمتر از یک مقدار آستانه‌ی پیش‌نگری[۲۴] باشد، می‌تواند به عنوان خروجی رأی‌گیرنده مشخص شود. مقدار آستانه‌ی پیش‌نگری یکی از خصوصیات برنامه می‌باشد. متدهای پیش‌نگری مختلفی برای پیاده‌سازی این‌گونه رای‌دهندگان مورد استفاده قرار گرفته‌اند. رأی‌گیرنده‌ی پیش‌نگر خطی[۲۵] از دو نتیجه‌ی قبلی رأی‌گیرنده برای پیش‌بینی استفاده می‌کند. در رأی‌گیرنده‌ی پیش‌نگر- تصحیح کننده[۲۶] یا پیش‌نگر مرتبه اول[۲۷] پیش‌بینی دقیق‌تری با صرف هزینه‌ی زمان اجرای بیشتر تهیه شده است. این رای‌دهندگان از الگوریتم پیش‌نگر- تصحیح کننده‌ی Milne استفاده می‌کنند[12]. این الگوریتم سه نتیجه‌ی قبلی رأی‌گیرنده را در نظر می‌گیرد تا مقدار مورد انتظار را تولید نماید. رأی‌گیرنده‌ی پیش‌نگر سه‌دامنه‌ای[۲۸] نمونه‌ی توسعه یافته‌ی رأی‌گیرنده‌ی پیش‌نگر- تصحیح کننده می‌باشد که در آن رکوردهای اشکال[۲۹] نیز مورد استفاده قرار می‌گیرد. رکورد اشکال در مواقع ناسازگاری با جلوگیری از انتخاب نتایج ماژول‌های غیر قابل اعتماد استفاده می‌شود. رکورد اشکال هر ماژول، هر بار که نتیجه‌ی آن در حالت اکثریت شرکت نمی‌کند، یک واحد افزایش می‌یابد. اگر هیچ سازگاری برقرار نباشد، رکورد بدون اشکال[۳۰] افزایش می‌یابد. رکوردهای اشکال در مواقع ناسازگاری، زمانی که ورودی در محدوده‌ی آستانه‌ی پیش‌نگری قرار داشته باشد، مورد استفاده قرار می‌گیرند. اگر نتیجه‌ای توسط ماژولی با مقدار پایین‌تر از میانگین رکورد اشکال تولید شده باشد، فرض می‌شود که صحیح است، در غیر این صورت هیچ نتیجه‌ای انتخاب نمی‌شود. در بخش بعدی روش طراحی شده معرفی گردیده است که توسط آن می‌توان به روش‌هایی که امکان تشخیص خطا ندارند این امکان را افزود.

3- روش پیشنهادی

همانطور که پیش از این گفته شد رأی‌گیرنده‌هایی وجود دارند که در آنها امکان تشخیص خطا وجود ندارد (برای مثال رأی‌گیرنده میانه و میانگین). این رأی‌گیرنده‌ها در تمامی چرخه‌های رأی‌گیری بدون توجه به ورودی‌هایی که رأی‌گیرنده دریافت می‌کند اقدام به تولید ورودی می‌کنند بنابراین احتمال انتخاب نتیجه غلط توسط این دسته از رأی‌گیرنده‌ها بالا خواهد بود. اگر بتوانیم روشی طراحی کنیم که توسط آن برخی از خطاهای رخداده در این رأی‌گیرنده‌ها را کشف کنیم می‌توان تعداد جواب‌های صحیح و غلط آنها را بهبود بخشید. رأی‌گیرنده‌هایی که امکان تشخیص خطا دارند (مانند رأی‌گیرنده اکثریت) در برخی از موارد ممکن است پاسخی به صورت "No-Result" تولید کنند. این پاسخ بدان معنا است که ورودی‌های رأی‌گیرنده به صورتی بوده‌اند که رأی‌گیرنده نتوانسته است پاسخ مناسبی توسط آنها تولید نماید(ورودی‌ها در توافق نبوده‌اند). برای اینکه بتوان مشخص کرد که رأی‌گیرنده در چه زمان‌هایی پاسخ تولید نکند، از میزان اختلاف ورودی ها با یکدیگر استفاده می‌کنیم.

منطق رأی‌گیرنده میانه بر این اساس است که اگر اکثریت صحیح باشد، میانه هم حتماً در اکثریت قرارمی‌گیرد (بر طبق فرض تعداد ورودی‌های رأی‌گیرنده فرد هستند) ولی اگر اکثریت اشتباه باشد، رأی‌گیرنده میانه هم حتماً پاسخ غلط تولید می‌کند. از طرف دیگر در زمانی که اکثریت وجود ندارد رأی‌گیرنده میانه ممکن است پاسخ صحیح یا غلط تولید کند در حالی که رأی‌گیرنده اکثریت "No-Result" تولید می‌کند. اکثریت در رأی‌گیری از میزان اختلاف ورودی‌ها محاسبه می‌شود یعنی اگر میزان اختلاف دو ورودی از یک میزان مشخص (آستانه رأی‌گیری[۳۱]) افزایش یابد، گفته می‌شود این دو ورودی در اکثریت نیستند. حال اگر بتوانیم رابطه‌ای میان میزان اختلاف ورودی‌ها و صحیح یا غلط بودن پاسخ نهایی در رأی‌گیرنده میانه بیابیم، می‌توان حالاتی را یافت که تولید "No-Result" در آن به‌صرفه باشد.

[23] Predictive Voters
[24] Prediction-threshold
[25] Linear-predictor Voter
[26] Predictor-corrector Voter
[27] First Order Predictor
[28] Three-domain Predictor voter
[29] fault Record
[30] No-fault Record
[31] Voter Threshold

اگر به ازای هر ورودی (m_i) میزان اختلاف آن ورودی با میانه را D_i در بگیریم، برای ورودی‌های کوچکتر از میانه، D_i مقداری منفی، برای خود میانه صفر و برای ورودی‌های بزرگتر از میانه مقداری مثبت خواهد بود. از آنجایی که تعداد ورودی‌های قبل و بعد از میانه با هم مساوی است (تعداد ورودی‌ها طبق فرض عددی فرد است)، اگر تمام ورودی‌ها به یکدیگر نزدیک باشند میزان مجموع اختلاف‌ها (SOD[32]) عددی نزدیک به صفر می‌شود و اگر میزان پراکندگی در ورودی‌ها زیاد باشد میزان SOD می‌تواند عددی بزرگ یا کوچک باشد. برای مثال در یک سیستم TMR اگر هر دو ورودی غیر از میانه به آن نزدیک باشند میزان SOD نزدیک به صفر خواهدبود. اگر فقط یکی از این دو ورودی به میانه نزدیک باشد SOD تقریباً برابر با میزان اختلاف ورودی دیگر با میانه است و اگر هیچ‌کدام از دو ورودی به میانه نزدیک نباشد SOD ممکن است مقادیر مختلفی را بپذیرد. از طرف دیگر اگر قدرمطلق اختلاف ورودی‌ها با میانه (ASOD[33]) را در نظر بگیریم این میزان نیز می‌تواند برای تمایز شرایط مختلف کمک کننده باشد. مطالب فوق به صورت فرمال به شکل زیر تعریف می‌گردد

a. Sort the voter inputs ascending, at each voting cycle. As a result, for any voter input i (x_i) where $i < n$ then $x_i < x_{i+1}$.
b. For each input define D_i as $D_i = x_{n/2} - x_i$
c. Define SOD as $SOD = \sum_{i=1}^{n} D_i$
d. Define ASOD as $ASOD = \sum_{i=1}^{n} |D_i|$

(where n is the number of voter inputs)

اگر نتایج ممکن را بر اساس دو پارامتر SOD و ASOD و بر اساس کوچک یا بزرگ بودن آنها طبقه بندی کنیم چهار حالت مختلف محتمل است:

1. SOD کوچک و ASOD کوچک باشد
2. SOD کوچک و ASOD بزرگ باشد
3. SOD بزرگ و ASOD کوچک باشد
4. SOD بزرگ و ASOD بزرگ باشد

حالت 1 نشان دهنده این خواهدبود که ورودی‌ها به یکدیگر نزدیک هستند و میتوان میانه را به عنوان یک جواب مناسب انتخاب کرد. حالت 2 زمانی اتفاق خواهد افتاد که اختلاف دو ورودی دیگر نسبت به میانه زیاد است اما این اختلاف برای هردو یکسان می‌باشد و شاید بتوانیم از روی میزان این اختلاف به یک جواب مناسب دست‌یابیم. رخداد حالت 3 اصولاً از لحاظ ریاضی غیرممکن است. حالت شماره 4 نیز زمانی رخ خواهدداد که میزان اختلاف ورودی‌ها زیاد بوده و تولید پاسخ امری غیر منطقی است.

کوچک یا بزرگ بودن دو پارامتر SOD و ASOD بر اساس مقایسه آنها با یک مقدار ثابت صورت خواهد پذیرفت. این مقدار ثابت را می‌توان به عنوان ضریبی از آستانه رأی‌گیری در نظرگرفت که برای بدست آوردن بهترین ضریب، رأی‌گیرنده باید در شرایط مختلفی مورد آزمون قرار بگیرد. اگر شرایط مختلفی که برای یک رأی‌گیرنده در یک سیستم TMR پیش می‌آید را بررسی کرده و نحوه پاسخ رأی‌گیرنده میانه و رأی‌گیرنده طراحی شده را با یکدیگر مقایسه نمائیم حالات مختلف زیر رخ خواهدداد:

1. هر سه نسخه، ورودی صحیح تولید کنند. در این حالت هر دو رأی‌گیرنده میانه و طراحی شده پاسخ صحیح تولید می‌کنند.
2. یکی از نسخه‌ها ورودی اشتباه تولید کرده و دو نسخه دیگر ورودی صحیح تولید نمایند. در این حالت نیز هر دو رأی‌گیرنده پاسخ صحیح تولید می‌نمایند.
3. ورودی دو نسخه از سه نسخه اشتباه باشد. در این حالت اگر بطور تصادفی پاسخ صحیح در بین دو پاسخ غلط قرار بگیرد هر دو رأی‌گیرنده پاسخ صحیح تولید می‌کنند. در غیر این صورت پاسخ رأی‌گیرنده میانه غلط است و رأی‌گیرنده طراحی شده بنا به فاصله ورودی‌ها از یکدیگر ممکن است پاسخ غلط یا "No-Result" تولید کند.
4. هر سه نسخه، ورودی اشتباه تولید کنند. در این حالت رأی‌گیرنده میانه حتماً پاسخ غلط تولید می‌کند و رأی‌گیرنده طراحی شده بنا به فاصله ورودی‌ها از یکدیگر ممکن است پاسخ غلط یا "No-Result" تولید کند.

در ادامه آزمایش‌های مختلفی برای شبیه سازی شرایط مختلف انجام شده‌است که نحوه انجام و نتایج آن در بخش بعدی آمده است.

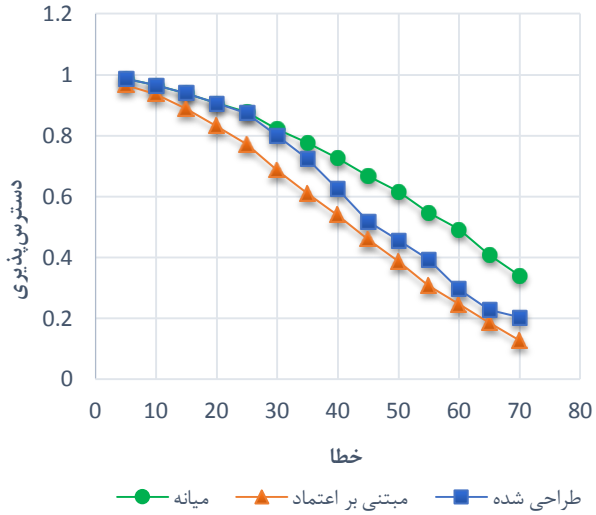
[32] Sum Of Diversions
[33] Absolute Sum Of Diversions

4- نتایج آزمایش

برای ارزیابی روش پیشنهادی آن را با رأی‌گیرنده میانه مقایسه می‌کنیم زیرا رأی‌گیرنده طراحی شده به نحوی بهبوددهنده رأی‌گیرنده میانه است. از طرف دیگر رأی‌گیرنده طراحی شده روشی است برای افزودن تشخیص خطا در رأی‌گیرنده‌های غیرتوافقی پس می‌توان آن را با یک روش دارای تشخیص خطا مثل رأی‌گیرنده اکثریت مبتنی بر اعتماد[۳۴] [13] نیز مقایسه نمود تا میزان بهبود روش مشخص گردد. برای انجام این عمل مانند مقالات [6]،[14]،[15] رأی‌گیرنده را در یک سیستم TMR فرضی قرار داده و بعد از تولید ورودی برای رأی‌گیرنده آزمایش‌های مورد نظر شبیه‌سازی خواهدشد. در این روش با تزریق اشکال به ورودی‌های تولید شده می‌توان رفتار انواع اشکالات با احتمال‌های مختلف را طراحی نمود. هر نسخه تولیدکننده ورودی توسط فرمول $100 + 100\ sin(t)$ یک ورودی تولید خواهد نمود که در این فرمول 100 پاسخ صحیح در نظرگرفته شده‌است و با تغییر میزان t می‌توان به این ورودی با احتمال‌های گوناگون اشکال تزریق نمود. برای مثال اگر t در بازه [-1،+1] انتخاب شود، حداکثر میزان $100\ sin(t)$ حدوداً برابر 1/75 خواهدشد و پاسخ تولید شده توسط این فرمول در بازه [101/75 ، 98/25] قرار خواهدگرفت. اگر آستانه رأی‌گیری برابر 0/5 در نظر گرفته شود احتمال تولید ورودی غلط در این بازه از اعداد حدوداً برابر 70٪ خواهد بود.

در هر آزمایش تعداد 10000 دوره رأی‌گیری وجود دارد و هر نسخه برای هر کدام از این دوره‌ها توسط فرمول قبلی یک ورودی تولید می‌کند که با این تعداد تکرار آزمایش می‌توان با احتمال 95٪ اطمینان داشت که نتیجه آزمایش حداکثر 1٪ خطا دارد. بعد از انجام آزمایش تعداد پاسخ‌های صحیح (n_c)، تعداد پاسخ‌های غلط (n_i) و تعداد "No-Result" ها (n_n) شمارش شده و برای استفاده در معیارهای ارزیابی مورد استفاده قرارمی‌گیرند. معیارهای ارزیابی در کارهای پیشین دسترس‌پذیری و ایمنی بوده است که دسترس‌پذیری به صورت آمادگی برای انجام وظایف به صورت صحیح و ایمنی به صورت عدم تولید پاسخ فاجعه‌آمیز تعریف می‌شوند. تعاریف انجام‌شده برای دو معیار فوق به صورت $S = 1 - \frac{n_i}{n}$ و $A = \frac{n_c}{n}$ فرمول خواهندشد[16].

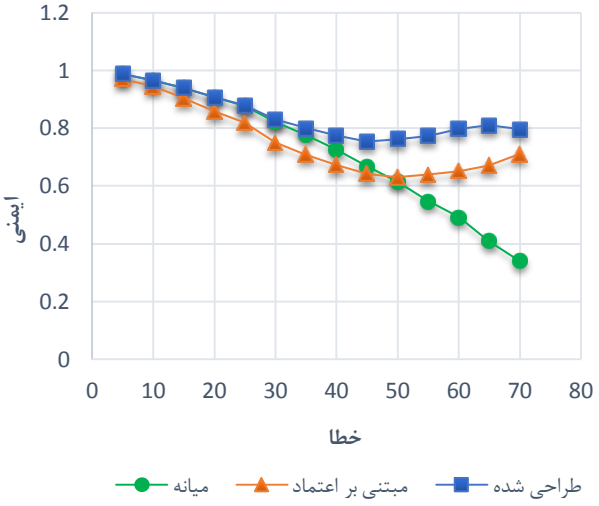
در آزمایش‌های انجام‌شده در یک نسخه به میزان 20٪ بیشتر از دو نسخه دیگر امکان رخداد خطا وجود دارد که در نمودارها عدد نمایش داده شده میزان احتمال خطا در دو نسخه دارای خطای کمتر است. شکل 2 نمودار دسترس‌پذیری و شکل 3 نمودار ایمنی برای رأی‌گیرنده‌های مورد نظر را نشان می‌دهند. در مجموع آزمایش‌های انجام‌شده و در مقایسه با رأی‌گیرنده میانه میزان دسترس‌پذیری به طور میانگین در حدود 1٪ کاهش دارد که به علت تشخیص بعضی از پاسخ‌های صحیح به عنوان غلط و تولید "No-Result" است. از طرف دیگر میزان ایمنی حدوداً 4٪ افزایش یافته که نشان دهنده میزان تشخیص خطای قابل قبولی است.



همانطور که در بخش قبلی نیز اشاره شد در این آزمایش‌ها باید مقادیری برای مقیاس کوچک یا بزرگ بودن دو پارامتر SOD و ASOD یافته شود. این مقیاس به عنوان ضریبی از آستانه رأی‌گیری در نظر گرفته شده‌است که آزمایش‌ها نشان دادند که برای کوچک بودن، میزان 2 برابر آستانه رأی‌گیری و برای بزرگ بودن میزان 6 برابر آستانه رأی‌گیری مقیاس‌های مناسبی هستند و در صورتی که تنها در این مواقع پاسخ تولید شود نتیجه بهینه بدست خواهد آمد.

[34] Confidence-Based Voter

شکل 2 دسترس‌پذیری - خطا



شکل 3 ایمنی - خطا

همانطور که در شکل 2 مشاهده می‌شود دسترس‌پذیری هر سه رأی‌گیرنده با افزایش احتمال رخداد خطا افزایش می‌یابد که به علت کاسته شدن از تعداد پاسخ‌های صحیح است. در شکل 3 برای ایمنی دو رأی‌گیرنده‌ای که امکان تشخیص خطا دارند این حالت رخ نخواهد داد، زیرا با افزایش احتمال خطا تعداد پاسخ‌های غلط افزوده نمی‌شود بلکه این تعداد "No-Result" ها است که افزایش می‌یابد.

5- نتیجه‌گیری

در این مقاله هدف یافتن روشی برای افزودن امکان تشخیص خطا در رأی‌گیرنده‌های نرم‌افزاری بوده‌است. تشخیص خطا در این سیستم‌ها امری بسیار مهم است زیرا کاربرد این رأی‌گیرنده‌ها در سیستم‌های بحرانی-ایمن، نظیر هواپیماهای مسافربری و سفینه‌های فضایی است که رخداد خرابی در آنها منجر به بروز فاجعه خواهدشد. با افزودن طرح این مقاله به رأی‌گیرنده میانه که امکان تشخیص خطا در آن وجود ندارد مشاهده شد که ایمنی در این رأی‌گیرنده به میزان حدوداً 4٪ افزایش یافته است. اگرچه این میزان در حالت کلی تغییر زیادی به نظر نمی‌رسد اما همانطور که گفته شد به علت ماهیت محیط کاری رأی‌گیرنده‌ها، تفاوت ایجاد شده بسیار مهم است.

6- منابع

[1] J. C. Knight, "Safety critical systems: challenges and directions," in *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on*, 2002, pp. 547-550.

[2] I. Koren and C. M. Krishna, *Fault-tolerant systems*: Morgan Kaufmann, 2010.

[3] H. Kopetz, Real-time systems: design principles for distributed embedded applications: Springer, 2011.

[4] B. W. Johnson, Design and Analysis of Fault-Tolerant Digital Systems. New York: Addison-Wesley, 1989.

[5] J.-C. Laprie, "Dependable computing and fault-tolerance," *Digest of Papers FTCS-15*, pp. 2-11, 1985.

[6] G. Latif-Shabgahi, J. M. Bass, and S. Bennett, "A taxonomy for software voting algorithms used in safety-critical systems," *IEEE Transactions on, Reliability*, vol. 53, pp. 319-328.2004 ,

[7] V. De Florio, G. Deconinck, and R. Lauwereins, "The EFTOS voting farm: a software tool for fault masking in message passing parallel environments," in *Euromicro Conference, 1998. Proceedings. 24th*, 1998, pp. 379-386.

[8] B. Parhami, "A taxonomy of voting schemes for data fusion and dependable computation," *Reliability Engineering & System Safety*, vol. 52, pp. 139-151, 1996.

[9] D. M. Blough and G. F. Sullivan, "A comparison of voting strategies for fault-tolerant distributed systems," in *Reliable Distributed Systems, 1990. Proceedings., Ninth Symposium on*, 1990, pp. 136-145.

[10] Z. Tong and R. Y. Kain, "Vote assignments in weighted voting mechanisms," *Seventh Symposium on, Reliable Distributed Systems, 1988. Proceedings.*, 1988, pp. 138-143.

[11] R. B. Broen, "New Voters for Redundant Systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 97, pp. 41-45, 1975.

[12] F. S. Acton, Numerical methods that usually work: MAA, 1990.

[13] M. Rezaee, Y. Sedaghat, and M. K. Farmad, "A Confidence-based Software Voter for Safety-Critical Systems," *2014 IEEE 12th International Conference on, Dependable, Autonomic and Secure Computing (DASC)* , 2014, pp. 196-201.

[14] G. Latif-Shabgahi and S. Bennett, "Adaptive majority voter: a novel voting algorithm for real-time fault-tolerant control systems," in

EUROMICRO Conference, 1999. Proceedings. 25th, 1999, pp. 113-120.

[15] G. Latif-Shabgahi, S. Bennett, and J. M. Bass, "Smoothing voter: a novel voting algorithm for handling multiple errors in fault-tolerant control systems," *Microprocessors and Microsystems*, vol. 27, pp. 303-313, 2003.

[16] G. Latif-Shabgahi, M. Tokhi, and M. Taghvaei, "Voting with dynamic threshold values for real-time fault tolerant control systems," in *Pre-prints of the 16th IFAC World Congress*, 2005.