یک نگرش چندعاملهٔ گسترده برای افزایش تحمل‌پذیری خطا

مریم سادات میریان حسین‌آبادی
mmirian@ut.ac.ir

مجید نیلی احمدآبادی
mnili@ut.ac.ir

قطب علمی کنترل و پردازش هوشمند و آزمایشگاه هوش مصنوعی و رباتیک، گروه مهندسی برق و کامپیوتر
دانشکده فنی- دانشگاه تهران

چکیده : ایجاد تحمل‌پذیری خطا در سیستم‌های گسترده، امری دست‌یافتنی و در عین حال دشوار است. یک سیستم چندعامله به عنوان نمونهٔ مهمی از سیستم‌های گسترده با بهره‌گیری از تکنیک‌های رایج تحمل‌پذیری خطا می‌تواند قابلیت کنارآمدن با انواع متفاوتی از خطاها را پیدا کند. اما آنچه در این مقاله مورد تاکید است استفاده از ذات چندعاملهٔ سیستم‌ها و بهره‌گیری از تکنیک‌هایی است که برخاسته از ویژگی‌های خاص سیستم‌های چندعامله است. در این طرح در صورت بروز مشکلی برای یک یا تعدادی از عامل‌ها، عامل‌های هم‌تیمی او با اعمال تغییراتی متناسب در وظایف خود سعی در کمک به او می‌نمایند و وظیفه وی را نیز با همکاری یکدیگر به دوش می‌گیرند و تا زمانی که مشکل آن عامل برطرف نشده به این روش از کاهش کارایی سیستم جلوگیری می‌کنند. این همکاری با یک تصمیم‌گیری گسترده و در طی چند فاز مختلف صورت می‌گیرد تادر صورت وجود چندین عامل نیازمند کمک، در نهایت عامل‌هایی با اولویت بالاتر برای کمک انتخاب شوند که بیش از بقیه در کارایی کلی نیم موثرند.

کلمات کلیدی : سیستم چندعامله، تحمل‌پذیری خطا، درخواست کمک، تصمیم‌گیری گسترده.[1]

۱. مقدمه

یک سیستم گسترده را می‌توان به منزلهٔ تیمی از عامل‌های همکار[2] دانست که هر یک وظیفهٔ خاصی به عهده دارند و از تعامل آنها با یکدیگر یک عملکرد واحد حاصل می‌شود. این شیوهٔ نگرش موجب می‌شود که بتوان از ایدهٔ سیستم‌های چندعامله برای توسعهٔ یک سیستم گستردهٔ تحمل‌پذیر خطا استفاده کرد.

یکی از تکنیک‌های رایج برای ایجاد تحمل‌پذیری خطا در یک سیستم، استفاده از افزونگی[3] است. خاصیت چندعاملی و عمل‌نمودن به صورت مستقل از یکدیگر خود نقطهٔ امیدی است بر این مطلب که در صورت وقوع یک خطا، این مشکل به کل سیستم توسعه نمی‌یابد و از گسترش آن به نقاط دیگر جلوگیری می‌شود. علاوه بر افزونگی صرف، در سیستم‌های چندعامله، تکنیک‌های پیشرفته‌تری هم برای ایجاد تحمل‌پذیری خطا وجود دارد و آن استفاده از خود عامل‌های موجود در سیستم برای رفع کاستی‌های ناشی از خطای عامل هم‌تیمی آنهاست.به بیان دیگر به جای جایگزین نمودن یک عامل خراب با یک عامل دیگر از خارج، از قابلیت‌های در حال حاضر اضافهٔ دیگر عاملهای موجود در طرح استفاده کنیم و نقش عامل خراب را نیز به نحوی بر دوش همکارانش قرار دهیم تا شرایط بحرانی موجود با کمترین تاثیر نامطلوب بر کارایی مرتفع گردد.در اصل علت استفاده از یک محیط چندعامله برای بیان ایده‌های این مقاله آن است که در چنین محیطی عناصر متفاوتی برای تقبل نقش یک عامل خراب در سیستم وجود دارد و تنها استفادهٔ مناسب از منابع موجود در سیستم بر عهدهٔ روتین تحمل‌پذیری خطاست.

در این مقاله نیز بهره‌گیری از خود عامل‌های درگیر در حل مساله به منظور رفع خطا در سیستم مورد تاکید است بدون اینکه از عامل واسطه‌ای[4] به منظور ایجاد هماهنگی در تنظیم وظایف جدید استفاده شود.به بیان بسیار خلاصه، یک عامل در صورت مواجهه با مشکلی که وی را از ادامهٔ همکاری بازدارد، یک پیام درخواست کمک صادر می‌کند و سایر هم‌تیمی‌های خود را از این واقعه مطلع می‌کند.دیگر عامل‌ها با دریافت درخواست وی بررسی می‌کنند که آیاامکان کمک به وی را دارندیا خیر. این بررسی از طریق یک فاز تصمیم‌گیری صورت می‌گیرد. در صورت تمایل به کمک و در صورتی که تعدادی عامل، خراب و نیازمند کمک باشند، عامل‌های کمک‌رسان بررسی می‌کنند که به کدام عامل‌ها و با چه ترتیبی کمک نمایند.

[1] Distributed Decision-Making
[2] Cooperative Agents
[3] Redundancy
[4] Middle Agents

در این مقاله پس از بررسی کارهای مشابه انجام شده در این زمینه که غالبا مرتبط با تشخیص خطا[1] هستند و نه رفع خطا[2]، روش پیشنهادی این مقاله را تشریح می‌کنیم و در ادامه به بیان نحوهٔ تحقق بستر آزمون[3] می‌پردازیم. در ادامه معیار ارزیابی کارایی شرح داده می‌شود و بعد از توضیحی در مورد سناریوی آزمون و نتایج شبیه‌سازی، نتیجه‌گیری و کارهای آینده مطرح می‌گردد.

۲. مروری بر کارهای مشابه

در مورد ایجاد تحمل‌پذیری خطا در یک سیستم چندعامله، دو دیدگاه متفاوت وجود دارد : اولین نظریه بر این اصل استوار است که چون سیستم‌های چندعامله ساختار ماجولار دارند، ذاتاً تحمل‌پذیر خطا هستند و اگر خطایی در یک ماجول بروز نماید می‌تواند از کل سیستم ایزوله شده و در سیستم انتشار نیابد. نظریه دوم بیان می‌کند که سیستم‌های چندعامله ذاتاً غیرامن هستند و چون کنترل در آنها توزیع شده بوده و تا حد زیادی غیرقطعی هستند، نمی‌توان یک رفتار را به ویژه در شرایط خطا تضمین نمود[۳].

در بسیاری از تحقیقات انجام شده در حوزه‌های MAS[4] و DAI[5] عموماً فرض می‌شود که عامل‌ها بدون خرابی هستند و تئوری‌ها و معماری‌های ارائه‌شده از بحث در مورد عامل‌های دچار اشکال اجتناب می‌کنند. در ادامه چند نمونه از تحقیقاتی که تحمل‌پذیری‌خرابی در سیستم‌های چندعاملهٔ نرم‌افزاری را به صورت محدود و در شرایط خاص فراهم آورده‌اند ذکر شده و سپس مرتبط ترین کارها به زمینهٔ این پژوهش را با شرح بیشتری بیان می‌کنیم.

در [۱] محیطی برای سیستم‌های توزیع‌شدهٔ همکار پیشنهاد شده‌است. هر یک از این سیستم‌ها به عنوان یک عامل خودمختار[6] در نظر گرفته می‌شود و مجموعه عامل‌ها برای حل مسائل کاربردی واقعی که طبیعتی گسترده دارند با هم همکاری دارند. در این محیط، تحمل‌پذیری خرابی با جایگزینی عامل‌های خراب انجام می‌شود و پس از جایگزینی عامل در برنامهٔ گروه تجدیدنظر صورت می‌گیرد.

در[۲] یک سری سرویس‌های مستقل از دامنه[7] برای رفع خطاهایی مانند حالات استثنایی[8] که ممکن است در محیط رخ بدهد، با اعمال نمودن تغییراتی به پروتکل هماهنگ‌سازی[9] معروف تیم‌های چندعامله یعنی Contract Net معرفی شده است. در[۳] استفاده از عامل‌های محافظ[10] به عنوان روشی برای ایجاد تحمل‌پذیری خطا در سیستم‌های چندعامله پیشنهاد شده‌است. عامل‌های محافظ، به منظور حفظ یک سری ویژگی‌ها در سیستم و نیز جلوگیری از ورود سیستم به وضعیت‌های ناخواسته در سیستم قرار می‌گیرند. عامل‌های محافظ یک ساختار کنترلی برای سیستم‌های چندعامله پدید می‌آورند. علت استفاده از عامل‌های محافظ آن است که آزادی عامل‌ها را در جایی که لازم است به منظور حفظ جامعیت سیستم محدود نمائیم ولی این کنترل آزادی باید ساده و قابل تغییر باشد.در[٤] استفاده از کارتیمی برای ساختن معماری مقاومی که بتواند بر خرابی‌های احتمالی واسط‌ها، فائق آید، پیشنهاد شده است.در این روش با استفاده از عملکرد تیمی همواره به صورت تضمین‌شده‌ای تعدادی واسط فعال در سیستم چندعامله داریم که عامل‌های دیگر با آنها در ارتباطند.

کاری نسبتا متفاوت در[۵]، یک مکانیزم ساختاردهی مجدد به وسیلهٔ روش‌هایی که در رشد جنین موجودات زنده استفاده می‌گردد را بیان می‌کند. در این سیستم نشان داده شده است که استفاده از ویژگی‌های سطح پایین و با سرعت بالا که در رفع خطای سیستم بدن جنین در طی فرایند تکاملی شکل می‌گیرند، روشی مناسب برای کاربردهای کنترلی بلادرنگ است. مفهوم Embryonics که از ترکیب واژهای جنین‌شناسی[11] و الکترونیک استخراج شده است، از ایدهٔرشد جنین در ارگانیزم‌های چند مولکولی گرفته شده است. ایدهٔ اصلی این روش در نحوهٔ پیاده‌سازی ساختاردهی مجدد است که قابلیت هر سلول یا گروهی از سلول‌ها را توسط همسایه‌های آنها مشخص می‌کند و بر اساس این روش، عملکرد هر سلول در صورتی که در موقعیت دیگری در بدن جنین قرار بگیرد، تفاوت خواهد نمود. این مساله به این شکل امکان‌پذیر می‌گردد که هر سلول یک کپی کامل از DNA که ارگانیزم را توصیف می‌کند دارا باشد.

اکنون به تحقیقاتی که بیش از بقیه به جهت گیری این مقاله نزدیک هستند، می‌پردازیم. در یک سیستم چندعامله، ایدهٔ متفاوت برای تحمل‌پذیری خطا این است که وقتی عاملی، خطایی را در یک سیستم چندعامله مشخص نمود، سیستم به گونه‌ای تغییر ساختار یابد که یا خطا را برطرف نماید یا به نحوی با آن کنار بیاید.منظور از تغییر ساختار، تغییر عامل‌های همکار یا تغییر در تعداد عامل‌های سیستم گسترده می‌باشد. رفتارتحمل‌پذیرخرابی‌در ربات متحرک به معنای آشکارسازی و تشخیص خرابی‌ها به صورت خودکار و توانایی ادامهٔ فعالیت پس از بروز خرابی می‌باشد. [٦] به آشکارسازی وتشخیص خرابی می‌پردازد و هدف آن توسعهٔ روشی است که رباتهای متحرک را در آشکارسازی، تشخیص و بر طرف کردن خرابی‌ها یاری رساند. ALLIANCE یک معماری کنترلی توزیع شده تحمل‌پذیرخرابی برای رباتهای همکار غیرمتجانس[12] است که با انتخاب اعمال به صورت تطبیقی، کنترل همکاری بین ربات‌ها را انجام می‌دهد.

[1] Fault Detection
[2] Fault Clearing
[3] Testbed
[4] Multi Agent Systems
[5] Distributed Artificial Intelligence
[6] Autonomus Agents
[7] Domain Independent Services
[8] Exceptions
[9] Coordination
[10] Sentinel Agents
[11] Embryology
[12] Heterogenous

انتخاب عمل به صورت تطبیقی با توجه به علایق ربات‌ها وتحلیل کارآیی سایر ربات‌ها در زمان‌های گذشته و حال انجام می‌شود.

در [۷] و [۸] ویژگی‌های عمومی وظایف در مأموریت یک گروه از ربات‌های همکار بررسی شده، که نتیجهٔ آن توصیهٔ روش کمک‌رسانی ربات‌ها به یکدیگر برای افزایش تحمل‌پذیری خرابی می‌باشد.در این روش با توسعهٔ معماری ALLIANCE قابلیت کمک‌رسانی به آن اضافه شده است.

در [۹] کمک‌رسانی در یک سیستم چندعامله به منظور افزایش تحمل‌پذیری خطا شرح داده شده است که در آن صرف ضریب بحرانی[1] بودن وظیفهٔ یک عامل محتاج به کمک را، ملاک تصمیم‌گیری عامل کمک‌رسان قرار داده و از پرداختن به پارامترهای دیگر در تصمیم گیری پرهیز نموده است. [۱۰] چند استراتژی برای ریسک کردن در کمک‌رسانی را معرفی نموده‌است. منظور از ریسک آن است که یک عامل بدون اینکه به زمان رسید وظیفه خود توجه کند، اقدام به کمک‌رسانی می‌کند و در صورت رسیدن وظیفهٔ جدیدی برای خود عامل، به علت اولویت بالاتر کارخودش، کار کمک‌رسانی با وقفه مواجه می‌شود. در [۱۱] مسالهٔ ناشکیبایی[2] عامل در کمک‌رسانی پرداخته شده است و یک عامل، پیش از تصمیم‌گیری نهایی برای کمک که بر اساس استراتژی‌های پایه‌ای برای زمانبندی صورت می‌گیرد، میزان میل به کمک‌رسانی را در خود بررسی می‌کند . در صورت رسیدن میل به آستانه، عامل اقدام به بررسی فاز بعدی تصمیم‌گیری برای کمک می‌کند و بر مبنای نتیجهٔ این مرحله که طی آن، عامل‌های مناسب کمک‌رسانی را از میان عامل‌های محتاج به کمک برگزیده است، ترتیب کمک‌رسانی نهایی را نیز تعیین می‌کند. در [۱۱] ضریب بحرانی بودن وظیفهٔ عامل‌ها در تعیین میزان ناشکیبایی دخالتی نداشته و در اصل میزان اهمیت وظیفهٔ عامل‌ها درست به اندازهٔ یکدیگر است. درحالیکه در [۱۲] تعریف میزان ناشکیبایی توسعه یافته‌است و با در نظر گرفتن ضریب بحران‌های متفاوت برای وظیفهٔ عامل‌ها، فرمول محاسبهٔ کارایی متفاوتی هم ارائه شده‌است. این مقاله به عنوان ادامه‌ای از کارهای انجام شده، سیاست‌های زمانبندی ترکیبی[3] در کمک‌رسانی را معرفی می‌کند وبا توجه به آزمایش‌های متعدد انجام شده، نشان می‌دهد که توجه به تمایل عامل به کمک‌رسانی و استفاده از سیاست‌های ترکیبی، کارایی سیستم را به طور قابل توجهی بهبود می‌بخشد.

[1] Criticality Coefficient
[2] Impatience
[3] Hybrid Scheduling Policies

۳. روش پیشنهادی

در این بخش فلسفهٔ ارائه و شرح کامل این روش به عنوان یک روش ایجاد تحمل‌پذیری خطا دریک سیستم چندعامله مطرح می‌شود.

محیط و فرضیات

به منظور معرفی بهتر ایده‌ها، یک سیستم چند عاملهٔ نوعی مشابه یک سیستم کنترل گسترده[4] برگزیده شده است. در این سیستم تعدادی عامل وجود دارند که هر یک وظیفهٔ خاص خود را دارند و داده‌های مورد نیاز آنها توسط یک تولید کنندهٔ داده، تولید شده و توسط یک باس مشترک در اختیارشان قرار می‌گیرد. در اصل این داده‌ها حکم داده‌های ورودی از طریق سنسورهای موجود در محیط کنترل گسترده را دارند و هر عامل با پردازشی که روی دادهٔ ورودی خود انجام می‌دهد، یک فرمان کنترلی برای محرک خارج از محیط ارسال می‌کند. این فرمان در حقیقت، خروجی هر یک از عامل‌هاست که به صورت مستقل از بقیه تولید شده و روی باس خروجی قرار می‌گیرد. در این سیستم یک شبیه‌ساز خطا نیز وجود دارد که به صورت تصادفی موجب بروز خطاهایی با طولهایی متفاوت برای هر یک از عامل‌ها می‌شود.

در این سیستم به مسالهٔ تشخیص خطا پرداخته نشده‌است و فرض بر این است که هر عامل قادر است از رخداد خطای خود آگاهی یابد و این مساله را با ارسال درخواست کمک به بقیه نیز اطلاع دهد. بنابراین یک باس مشترک به منظور انتقال درخواست‌های کمک وجود دارد. این درخواست کمک باید حتی‌الامکان کوتاه بوده و حاوی مهمترین اطلاعات مورد نیاز برای کمک‌رسانی به دیگر عامل‌ها باشد. علاوه بر این فرض شده است که یک عامل قادر است وظیفهٔ هم‌تیمی‌های خود را نیز در مواقع لزوم انجام بدهد. فرض دیگری که در سیستم لحاظ شده آن است که ضریب بحران وظیفهٔ هر عامل با عامل‌های دیگر متفاوت است و بنابراین فرمول کارایی سیستم که در بخش۵ به آن اشاره شده است، با توجه به ضریب بحران وظیفهٔ عامل‌ها محاسبه می‌شود.

هر عامل باید از متوسط/ ماکزیمم زمان اجرای وظیفهٔ دیگر عامل‌ها آگاهی داشته باشد. این اطلاع، عامل را یاری می‌کند تا در مورد امکان‌سنجی کمک مورد تقاضا تصمیم بگیرد. علاوه بر این، زمان واقعی انجام کار هر عامل ثابت نیست و این خود بر پویایی مساله می‌افزاید. به منظور جلوگیری از ارسال کمک‌های تکراری به یک عامل که موجب به هدر رفتن امکانات و توان سیستم می‌شود، عامل‌های کمک‌رسان پیش از کمک، وضعیت یک پرچم مشترک اختصاصی را بررسی می‌کنند. اگر وضعیت این پرچم پایین باشد به این معناست که تا این لحظه به این عامل کمکی نشده است و بنابراین هر کمکی که

[4] Distributed Control System

ارسال شود مطلوب خواهد بود. اما در صورتی که این پرچم بالا باشد، عامل کمک‌رسان متوجه می‌شود که در مورد دادهٔ فعلی به این عامل کمک شده است و در صورت ارسال کمک، این کمک تکراری خواهد بود. با ارسال دادهٔ جدیدی برای یک عامل، پرچم دوباره به حالت پایین می‌آید تا امکان کمک‌رسانی را فراهم نماید.

معرفی روش

همانطور که قبلا نیز بیان شد، هدف این تحقیق ارائهٔ روشهایی است که در آنها خود عامل‌ها به صورت گسترده و با تغییر نقش، اقدام به کمک‌رسانی به عامل‌های خراب می‌کنند تا از کاهش کارایی جلوگیری شده و سیستم با تحمل‌پذیری خرابی بیشتری قادر به ادامهٔ کار باشد.

در این روش، کمک‌رسانی هنگامی آغاز می‌شود که یک عامل با مشکلی مواجه شود. در این شرایط، دیگر عامل‌ها از طریق پاس کمک متوجه می‌شوند که کدام عامل نیازمند کمک است و دادهٔ مورد نیاز برای انجام وظیفهٔ این عامل را از پاس مشترک داده برمی‌دارند. به محض اینکه عامل کمک‌رسان وضیفهٔ خود را تکمیل نمود، شروع به پردازش تقاضاهای کمک می‌کند. در ابتدا بررسی می‌کند که کمک به کدام یک از عامل‌های خراب مطابق با میل و در جهت منفعت وی می‌باشد. در این فاز از تصمیم‌گیری، پارامترهای مختلفی برای اتخاذ یک تصمیم جامع مورد بررسی قرار می‌گیرد که جزئیات آن کمی جلوتر مطرح شده است. در انتهای این فاز و بر اساس تصمیمات اتخاذ شده، تعداد محدودتری از عامل‌های کمک‌رسان، خود را آمادهٔ کمک‌رسانی احساس می‌کنند. اگر بیش از یک عامل نیازمند کمک باشند، عامل‌های کمک‌رسان بر اساس یک استراتژی کمک‌رسانی توزیع شده که در مرحلهٔ طراحی تعیین شده است، ترتیب کمک‌رسانی را تعیین می‌کنند. این سیاستها به دو دستهٔ عمدهٔ ساده و ترکیبی[1] تقسیم می‌شوند که در بخش۳-٤ آورده شده‌اند.

رقابت عامل‌ها و انتخاب کمک‌رسانها

هنگامی که یک درخواست کمک می‌رسد، بسیار محتمل است که عامل‌های کمک‌رسان متعددی در سیستم آمادهٔ کمک‌رسانی باشند. بنابراین یک رقابت بر سر مسالهٔ کمک‌رسانی انجام می‌شود. این رقابت در واقع یک فاز تصمیم‌گیری است که میزان تمایل یک عامل کمک‌رسان را به انجام وظیفهٔ مورد تقاضا محاسبه می‌کند. در صورت رسیدن به آستانهٔ تمایل، کاندیداهایی برای کمک‌رسانی به مجموعه‌ای از عامل‌های خراب انتخاب می‌شوند. در محاسبهٔ میزان تمایل این پارامترها دخیل هستند : RT_i زمان باقیمانده برای عامل کمک‌رسان در صورتی که کمک مورد تقاضا را انجام بدهد، C_i ضریب بحران وظیفهٔ عامل کمک‌رسان، C_j ضریب بحران وظیفهٔ عامل خراب، HNo_{ij} تعداد کمک‌هایی که تا بحال توسط عامل کمک‌رسان مورد نظر به عامل خراب مذکور انجام شده است و FNo تعداد عامل‌های خراب در سیستم در لحظهٔ پاسخگویی به تقاضای کمک.

توجه به زمان باقیمانده نشان می‌دهد، در صورتی که عامل کمک‌رسان وقت آزادی بسیار بیشتر از آنچه مورد نیاز است دارد، کمک خواستن و ایجاد مشغولیت برای وی موجب می‌شود که اگر درخواست کمکی با زمان مورد نیاز طولانی‌تر برسد، وهیچ یک از عامل‌ها وقت کافی نداشته باشند، عامل کمک‌رسانی یافت نشود و سیستم دچار مشکل گردد. ضریب بحران وظیفهٔ هر دو عامل کمک‌رسان و خراب به منظور انتساب اولویت بالاتر به عامل با اهمیت‌تر برای کمک در نظر گرفته شده است. با در نظر گرفتن تعداد کمک‌هایی که تا بحال توسط یک عامل به عامل محتاج به کمک صورت گرفته است، می‌توان اثر مجازی برای یادگیری در سیستم شبیه‌سازی نمود، به این معنا که با وجود اینکه در سیستم فعلی، یادگیری به معنای واقعی وجود ندارد، در صورتی که افزایش تعداد کمک‌های تابحال انجام شده، موجب کاهش زمان انجام وظیفه‌هایی بشود که در آینده به یک عامل محول می‌شوند، قادر خواهیم بود که اثر مجازی برای یادگیری شبیه‌سازی کنیم و بنابراین عامل‌های باتجربه‌تر با میل بیشتری اقدام به کمک می‌نمایند.

با در نظر گرفتن تعداد عامل‌های خراب در سیستم در یک لحظه، عامل کمک‌رسان می‌تواند یک شناخت نسبی از وضعیت عامل‌های هم‌تیمی خود بدست آورد و متوجه شود که میزان نیاز تیم به کمک وی تا چه اندازه جدی است.

روش‌های مختلفی برای ترکیب عوامل مختلف در محاسبهٔ تمایل به کمک عامل و نیز محاسبهٔ مقادیر بهینهٔ مورد استفاده در فرمول تمایل وجود دارد که یکی از آنها در این تحقیق به این شکل معرفی می‌شود:

$$Willingness2Help_{ij} = \frac{C_j.FNo.HNo_{ij}}{C_i.RT_i}$$

در این تحقیق، یک آستانهٔ تمایل تطبیقی برگزیده شده است که با توجه به بیشینه و کمینهٔ مقادیر دخیل در فرمول محاسبه می گردد.

انتخاب مجموعه ای از عامل‌های نیازمند برای کمک‌رسانی

اکنون پس از فاز رقابت، هر یک از عامل‌های مایل به کمک، مجموعه ای از عامل‌های نیازمند به کمک را کاندیدا نموده‌اند. لازم به یادآوری است که این مجموعه عامل‌ها لزوما تمام عامل‌های نیازمند به کمک نیستند، بلکه عامل‌هایی هستند که تمایل به کمک به آنها، در یک یا تعدادی از عامل‌ها فراتر از آستانهٔ تمایل بوده است.. برای اینکه ترتیب کمک‌رسانی به این مجموعه از عامل‌ها توسط هریک از کمک‌رسانها تعیین شود، از یکی از سیاستهایی که در

[1] Hybrid

اینجا شرح داده شده‌اند، استفاده می‌شود. این سیاستها به دو دستهٔ ساده و ترکیبی تقسیم می‌شوند.

سیاستهای ساده

استراتژی‌های کمک‌رسانی ساده‌ای که مربوط به ترتیب تخصیص زمان پردازشی عامل به وظایف مورد تقاضا هستند، عبارتند از:

Best Fit (BF): انتخاب عاملی با طولانی‌ترین وظیفه، به شرطی که قابل انجام در زمان باقیماندهٔ عامل کمک‌رسان باشد. در این سیاست، عامل کمک‌رسان زمان لازم برای اجرای وظیفهٔ مورد نیاز عامل خراب را از زمان باقیمانده‌ای که می‌تواند صرف کمک‌رسانی نماید کم می‌کند و سپس این تفاضل را به صورت صعودی مرتب می‌کند. بدین شکل یک لیست حاصل می‌شود که اگر به ترتیب از ابتدای لیست سرویس دهی آغاز شود، سیاست BF پیاده شده است. با این سیاست، عامل کمک‌رسان طولانی‌ترین وظیفهٔ قابل انجام را برمی‌گزیند با این فرض که دیگر عامل‌هایی که زمان آزاد کمتری دارند، با احتمال بیشتری خواهند توانست دیگر کارهای کوتاهتر را انجام بدهند.

Shortest Job First (SJF): انتخاب عاملی با کوتاهترین زمان انجام وظیفه، به شرطی که قابل انجام در زمان باقیماندهٔ عامل کمک‌رسان باشد. این سیاست معمولا توسط عامل‌های خودخواه انتخاب می‌شود چرا که در این سیاست عامل، کارهای کوتاهتر را انجام می‌دهد و کارهای طولانی‌تر را رها می‌کند تا عامل‌های هم‌تیمی به آنها سرویس‌دهی نمایند.

First Come First Served (FCFS) : انتخاب اولین عاملی که تقاضای کمک نموده است، به شرطی که قابل انجام در زمان باقیماندهٔ عامل کمک‌رسان باشد. در این سیاست به تقاضاهای کمک به ترتیب، بر اساس زمان رسیدشان سرویس‌دهی می‌شود.

Earliest Deadline First (EDF) : انتخاب اولین عاملی که مهلت زمانی‌اش به سرآمده است، به شرطی که قابل انجام در زمان باقیماندهٔ عامل کمک‌رسان باشد.

Most Critical First (MCF) : انتخاب عاملی با مهم‌ترین وظیفه، به شرطی که قابل انجام در زمان باقیماندهٔ عامل کمک‌رسان باشد. در این روش، توجه به ضریب بحران یک عامل موجب می‌شود که کار عامل‌های بحرانی‌تر پیش از بقیه انجام گیرد و این موجب نجات تیم از خرابی کلی می‌شود.

معرفی سیاستهای ترکیبی

یک سیاست زمانبندی ترکیبی در اصل، انجام دو سیاست ساده به صورت متوالی است. برای مثال اگر سیاست MCF را در نظر بگیریم، لیست حاصل از اعمال این سیاست به تنهایی، لیستی است از عامل‌هایی که درخواست کمک صادر کرده اند، مرتب شده بر اساس ضریب بحران وظیفهٔ هر کدام. در این لیست، در صورتی که دو یا چند عامل دارای ضریب بحران برابر باشند، می‌توان از پارامترهای دیگری چون طول اجرای وظیفهٔ مورد تقاضا، مهلت زمانی وظیفهٔ هر عامل و زمان دریافت تقاضای کمک استفاده نموده، لیستی جدید بدست آورد که عامل‌های با ضریب بحران یکسان بر اساس این اطلاع اضافه‌تر، مرتب شده باشند. این سیاست‌ها را که روی نتایج اولیهٔ MCF عمل می‌کنند، به ترتیب BF_over_MCF ، SJF_over_MCF ، EDF_over_MCF و FCFS_over_MCF نامیده‌ایم.به همین ترتیب، سیاست‌های ترکیبی حاصل از انجام دوبدوی سیاست‌های سادهٔ بخش قبل که قابل انجام و بامعنی بوده اند در جدول ١ آورده شده‌اند:

جدول ١. سیاست‌های ترکیبی زمانبندی برای کمک‌رسانی

-	-	BF_over_MCF
MCF_over_SJF	EDF_over_BF	SJF_over_MCF
FCFS_over_SJF	MCF_over_BF	EDF_over_MCF
EDF_over_SJF	FCFS_over_BF	FCFS_over_MCF

در بخش٦ نتایج حاصل از شبیه‌سازی آزمایش‌های انجام شده با سیاست‌های ساده و ترکیبی نشان داده شده است و در نهایت مقایسه ای از کارایی این دو دسته بیان گردیده است.

انجام نهایی عملیات کمک

اکنون عامل کمک‌رسان بر اساس سیاستی که در پیش‌گرفته‌است، یک لیست مرتب شده از وظایف برای انجام‌دادن دارد و می‌تواند از ابتدای لیست سرویس‌دهی را آغاز نماید تا اینکه زمان باقیمانده‌اش به پایان برسد. وقتی زمان باقیمانده به پایان برسد، m عامل (نه لزوما کلیهٔ عامل‌های نیازمند کمک و نه همهٔ عامل‌هایی که عامل کمک‌رسان تمایل کمک به آنها را داشته است)، کمک‌رسانی شده‌اند. عامل‌هایی که به آنها کمک نشده‌است، ممکن است در همین فاز توسط دیگر عامل‌ها کمک‌رسانی شوند. البته ممکن است، همین عامل کمک‌رسان که فعلا قادر به کمک نبوده در گام‌های بعدی به این عامل‌ها کمک نماید.

٤. تحقق بستر آزمون

از آنجا که در یک سیستم گسترده، کلیهٔ اجزاء باید به صورت موازی با یکدیگر فعال بوده، توالی در زمان برای اجرا نداشته باشند و نیز به علت تمایل به بررسی ایده‌ها در یک محیط واقعی سخت افزاری، از شبیه‌سازی توسط زبان توصیف سخت افزاری VHDL استفاده شد. با توصیف عامل‌ها در سطح رفتاری در این زبان، قادر خواهیم بود که از ویژگی‌های قدرتمند آن مانند

همگام سازی[1]، سلسله مراتب طراحی و کنترل کامل زمانی در تمامی سطوح بهره بگیریم.

۵. معیار ارزیابی کارایی

تعداد وظیفه‌های با موفقیت انجام شدهٔ هر عامل، با توجه به ضریب بحران هر عامل به عنوان معیاری برای کارایی در نظر گرفته شده است. این معیار در فرمول زیر نشان داده شده است:

$$Performance = \frac{\sum_{i=1}^{Agents_No} \sum_{j=1}^{Tasks} n_{ij} C_{ij}}{\sum_{i=1}^{TotalTaskNo} C_i}$$

که در آن $Agents_No$ تعداد عامل‌های سیستم، n_{ij} تعداد وظیفه‌های $Agent_i$ با ضریب بحران C_{ij} است که C_{ij} خود درجه اهمیت $Task_j$ مربوط به $Agent_i$ است.

پارامتر $TotalTaskNo$ نشان دهندهٔ کل تعداد وظایف سیستم است. در این سیستم، وظیفه‌ها تنها به یک دلیل ممکن است از دست بروند و آن این است که یک عامل دچار خرابی شده باشد و بقیهٔ عامل‌ها هم یا به دلیل عدم تمایل و یا به دلیل محدودیت‌های زمانی خود، قادر به کمک به وی نباشند. کمک به یک عامل و از دست رفتن وظیفه‌های خود عامل کمک‌رسان، نمی‌تواند به عنوان یک دلیل مطرح باشد چرا که عامل‌ها پس از بررسی محدودیت‌های زمانی خود و امکان‌سنجی کمک، اقدام به کمک‌رسانی می‌نمایند.

٦. سناریوی آزمون و نتایج شبیه‌سازی

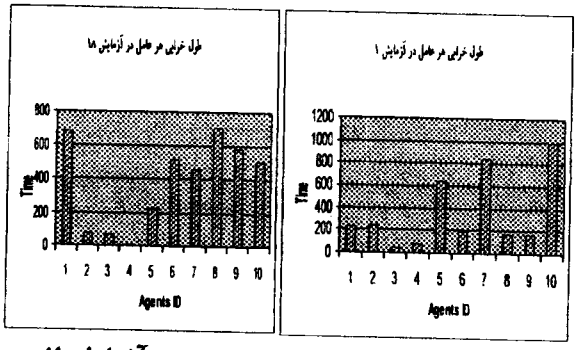
در این بخش سناریوهای آزمون، چگونگی شرایط آزمایش‌ها و نتایج حاصل از شبیه‌سازی‌ها ارائه می‌شود.هر یک از استراتژی‌های شرح داده شده در بخش۳-۲، با انجام ۳۰ آزمایش با الگوهای تصادفی برای خطا شبیه‌سازی شده‌اند.

به عنوان مثال، شکل۱ ، دو مورد از الگوهای تصادفی را که برای خرابی عامل‌ها در نظر گرفته‌ایم، نشان می‌دهد.هر یک از این آزمایش‌ها تا ٤٢٠٠ واحد زمانی ادامه می‌یابند. نرخ ورود داده‌ها برای هر عامل ۱/۲۰ بوده و ضریب بحران وظیفهٔ عامل‌ها در طول این آزمایش‌ها درجدول۲ نشان داده شده است. جدول ۳ به منظور مقایسهٔ کارایی استراتژی‌های زمانبندی برای کمک‌رسانی نشان داده شده است. در این جدول متوسط کارایی هر استراتژی در طی این ۳۰ آزمایش نشان داده شده است.

[1] Concurrency

جدول ۲- ضریب بحران وظیفهٔ عامل‌ها

ID	Criticality
Agent1	5
Agent2	5
Agent3	5
Agent4	5
Agent5	5
Agent6	8
Agent7	8
Agent8	8
Agent9	10
Agent10	10



شکل ۱ . طول خرابی عامل‌ها در دو آزمایش از سی آزمایش نمونه

جدول ۳- مقایسهٔ کارایی استراتژی‌های زمانبندی

سیاست‌های زمانبندی	متوسط کارایی
SJF_over_MCF	۹۸،۵
EDF_over_MCF	۹۸،۱
BF_over_MCF	۹۷،۵
FCFS_over_MCF	۹۷،۲
MCF	۹۷
EDF	۹۷
EDF_over_BF	۹۵
MCF_over_BF	۹٤،٤
FCFS_over_BF	۹٤
BF	۹٤
SJF	۸۸،۵
EDF_over_SJF	۸۸
MCF_over_SJF	۸۷،٤
FCFS_over_SJF	۸۷،۱
FCFS	۸۲

با توجه به اینکه متوسط کارایی سیستم در طی این ۳۰ آزمایش و بدون هرگونه عملیات کمک‌رسانی در حدود ٤٥٪ است، مشاهده می‌شود که با استفاده از این سیاست‌ها، به طور متوسط حدود٤٠٪ بهبود در کارایی حاصل می‌شود و در بهترین حالت با استفاده از سیاست SJF_over_MCF کارایی در حدود ٥٠٪ افزایش یافته است.

۷. نتیجه‌گیری و کارهای آینده

آزمایش‌های انجام شده نشان می‌دهد که در نظر گرفتن یک مکانیزم تصمیم‌گیری جامع، برای امکان‌سنجی کمک در شرایط خرابی می‌تواند تاثیرات قابل ملاحظه‌ای روی کارایی سیستم داشته باشد و همچنین درنظرگرفتن سیاست‌های صحیحی که اولویت را در کمک‌رسانی معین کنند، بسیار اهمیت دارد.

کمک‌رسانی صرفا بر اساس درجه اهمیت یک عامل، همانطور که در [۹] انجام شده است، در مقایسه با ریسک برای کمک که در [۱۰] انجام شده است از کارایی خوبی برخوردار نیست. البته با در نظر گرفتن تمایل به کمک، حتی با وجود استفاده از سیاست‌های سادهٔ زمانبندی که در[۱۱] و [۱۲] هم انجام شده‌است، پیشرفت قابل‌ملاحظه‌ای در صحت نسبی تصمیم‌گیری‌ها حاصل می‌شود که شاهد این مدعا افزایش کارایی سیستم است. استفاده از سیاست‌های ترکیبی که در این مقاله معرفی شد، نشان‌دهندهٔ بهبود بیشتری در کارایی است. از جمله قدم‌های بعدی این تحقیق، تغییر پویای روش تصمیم‌گیری برای کمک‌رسانی است. مورد دیگر اعمال تغییراتی در معماری است به نحوی که در صورت وجود عامل‌های سری در طرح، این روش همچنان پاسخگوی نیازهای تحمل‌پذیری خطا باشد.

مراجع

[1]Deen, S.M., "A fault-tolerant cooperative distributed system", Proceedings Ninth International Workshop on Database and Expert Systems Applications, 1998, pp. 508-513

[2]Chrysanthos Dellarocas and Mark Klein, "An Experimental Evaluation of Domain-Independent Fault Handling Services in Open Multi-Agent Systems", ICMAS2000

[3]Staffan Hugg, "A Sentinel Approach to Fault Handling in Multi-Agent Systems", Department of Computer Science, University of Karlskrona/Ronneby, Sweden.

[4]Sanjeev Kumar , Philip R. Cohen,"Towards a Fault-Tolerant Multi-Agent System Architecture", Oregon Graduate Institute, Autonomus Agent 2000

[5]Daryl Bradley, Cesar Ortega-Sanchez, Andy Tyrrell, "Embryonics + Immunotronics: A Bio-Inspired Approach to Fault Tolerance", Department of Electronics, University of York, Heslington, York.

[6]L. E. Parker, "ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots," in Proc. 1994 IEEE/RSJ/GI Int. Conf. Intell. Robot. Syst. (IROS '94), Munich, Germany, Sept. 1994, pp. 776—783

[7] Foad Ghaderi, Majid Nili, "Fault-Tolerance in Cooperative Robots Using others' Helps", 7th Iranian Computer Conference, pp 220-227, Feb. 2002.

[8]Foad Ghaderi, Majid Nili Ahmadabadi, "Distributed Cooperative Fault Tolerance In A Team Of Object Lifting Robots", Accepted in IROS2002, Oct. 2002.

[9]Maryam S. Mirian, Majid Nili Ahmadabadi, Zainalabedin Navabi, "A New Task Redistribution Method for Fault Clearing in Multi-Agent Systems", Accepted in IEEE SMC2002, Tunisia, Oct. 2002

[10] Maryam S. Mirian, Majid Nili Ahmadabadi, Zainalabedin Navabi, "Evaluating Task Redistribution Methods for Fault Clearing in Multi-Agent Systems", ERSA'02, pp219-225, Las Vegas, USA, May 2002

[11] Maryam S. Mirian, Majid Nili Ahmadabadi, Zainalabedin Navabi, "Agent's Role Reconfiguration Based on Decision-Making for Distributed Fault Recovery in Multi Agent Systems", AIM'02, Workshop of EURASIA-ICT 2002, pp 289-293, Shiraz, Iran, Oct. 2002.

[12] Maryam S. Mirian, Majid Nili Ahmadabadi, Zainalabedin Navabi, "A Decision-Making based Approach for Fault-Handling in Multi-Agent Systems", Accepted in ICONIP'02, Singapore, Nov. 2002.