مقاله پژوهشی

طراحی یک کنترل‌کننده هوشمند اضافه بار جهت استفاده در شبکه‌های نسل آینده

مهدی خزائی

چکیده: SIP به عنوان پروتکل سیگنالینگ برای زیرسیستم‌های مبتنی بر IP (IMS) در نظر گرفته شده و از طرفی IMS به عنوان پلتفرم شبکه‌های نسل آینده معرفی گردیده است. SIP برخلاف ویژگی‌های مثبتی مانند مبتنی بر متن، مبتنی بر IP، مستقل از داده انتقالی، پشتیبانی از جابه‌جایی و انتها به انتها بودن، فاقد مکانیزم مناسبی در مواجه با اضافه بار می‌باشد. از این رو، این چالش باعث خواهد شد که کاربران گسترده شبکه‌های نسل آینده با افت شدید کیفیت در خدمات مواجه شوند. IMS توزیع‌شده، یک شبکه پیچیده محسوب می‌گردد که متشکل از زیرسیستم‌هایی است که با یکدیگر در فعل و انفعال می‌باشند. در نتیجه، سیستم‌های چندعامله می‌توانند ابزار مناسبی برای حل مشکل اضافه بار در این شبکه باشند. به این منظور، هر سرور IMS به عنوان یک عامل هوشمند در نظر گرفته می‌شود که با حفظ خودمختاری، قابلیت یادگیری و مذاکره با بقیه عامل‌ها را داراست تا اضافه بار توسط ارتباطات و دانش جابه‌جاشده در بین عامل‌ها رفع گردد. در این مقاله، به واسطه سیستم‌های چندعامله و خواص آنها، روش گام به گام مبتنی بر حذف ارائه گردیده که نتایج شبیه‌سازی و مقایسه با روش معروف ارائه‌شده قبلی، بهبود کارایی را نشان می‌دهد.

***کلیدواژه:* IMS، SIP، کنترل اضافه بار، سیستم‌های چندعامله.**

۱- مقدمه

سوئیچینگ بسته‌ای به فراهم‌کنندگان خدمات این امکان را داده که کاربردهای چندرسانه‌ای را به مشترکان خود ارائه دهند. ولی رویکرد آینده صنعت مخابرات به سوی شبکه‌های نسل آینده می‌باشد که در آن، انواع شبکه‌های دسترسی ثابت و سیار بر اساس IP مجتمع شده و یک بستر استاندارد و یکپارچه را برای خدمات چندرسانه‌ای ارائه می‌کنند. در شبکه‌های نسل آینده قابلیت ارائه سرویس‌های چندرسانه‌ای مبتنی بر IMS[1] به عنوان پلتفرم استاندارد وجود داشته که در آن تمام خدمات چندرسانه‌ای مانند ارسال پیام فوری، ارسال ویدئو و ارسال صوت (VOIP) به صورت یک‌جا و مجتمع بدون وابستگی سخت‌افزاری و نرم‌افزاری قابل پیاده‌سازی است. از این رو، پروتکل SIP توسط ۳GGP به عنوان معماری پایه IMS به تصویب رسیده است. IMS از نسل سوم شبکه‌های موبایل به بعد توسط اکثر اپراتورها مورد استفاده قرار گرفته و بیشتر تلفن‌های سلولی و وسایل بی‌سیم، پروتکل SIP را به عنوان ایجادکننده نشست‌های چندرسانه‌ای پشتیبانی می‌کنند. پروتکل SIP قابلیت برقراری و مدیریت چندین نشست به صورت هم‌زمان را دارا است [۱].

SIP علی‌رغم مدیریت نشست‌ها در خیلی از کاربردهای چندرسانه‌ای، از مشکل اضافه بار رنج می‌برد و زمانی که بر روی پروتکل لایه انتقال غیر مطمئن مانند UDP اجرا شود، جهت مقابله با گم‌شدن بسته‌ها از مکانیزم ارسال مجدد استفاده می‌کند. این کار با نگه‌داشتن زمان‌سنج‌های مختلف به ازای هر درخواست ارسالی انجام شده که در صورت عدم دریافت پاسخ مربوط در مدت زمان مشخص، درخواست‌ها مجدداً ارسال می‌شوند. این مکانیزم در شرایط اضافه بار سبب وخیم‌ترشدن اوضاع می‌گردد، زیرا تمام ظرفیت CPU صرف پردازش بسته‌های تکراری و زمان‌سنج‌ها شده و عملاً گذردهی به سمت صفر می‌رود. اما امروزه با رشد حجم حافظه‌ها و کاهش قیمت آنها و از طرفی با اتمام ماشین‌های حالت تراکنش‌ها توسط CPU معمولاً کمبود حافظه اتفاق نخواهد افتاد [۲].

زمانی که SIP با اضافه بار مواجه می‌شود، مکانیزم کنترل اضافه بار آن فعال گردیده و پیام‌های درخواست ایجاد نشست جدید را با ارسال پاسخ ۵۰۳ رد می‌کند. در این روش، ابتدا باید پیام‌ها تجزیه شوند تا مشخص گردد که درخواستی جدید است یا خیر و سپس پاسخ رد برای درخواست‌های جدید ایجاد شود. ظرفیتی که سرور صرف تجزیه پیام‌ها و تولید پاسخ رد می‌کند، عملاً دور ریخته شده و پردازش مفیدی صورت نمی‌گیرد که علاوه بر اثر منفی بر اضافه بار، سرور باید تمام توان خود را صرف ردکردن درخواست‌ها نماید. از این رو طراحی یک کنترل‌کننده اضافه بار برای IMS اجتناب‌ناپذیر است. کنترل‌کننده اضافه بار باید بتواند اطلاعات مورد نیاز جهت تصمیم‌گیری در مورد اضافه بار را جمع‌آوری و بر اساس این اطلاعات، عمل مناسب جهت مواجهه با اضافه بار را تعیین کند و سپس آن را به شبکه اعمال نماید. بسته به نوع اضافه بار و محل قرارگرفتن کنترل‌کننده، روش‌ها و سیاست‌های مختلفی جهت مقابله با اضافه بار ارائه شده است [۳].

به دلیل پیچیدگی و غیر مستدل بودن مسئله اضافه بار در IMS، اکثر روش‌های ارائه‌شده جهت طراحی کنترل‌کننده از روش‌های تقریبی ریاضی یا روش‌های فراابتکاری استفاده می‌کنند که گاهی دارای خطای زیادی بوده و در برخی شرایط، جواب‌های قابل قبولی ارائه نمی‌دهند. از این رو در این مقاله روش جدیدی مبتنی بر یادگیری ماشین بدون ناظر جهت طراحی کنترل‌کننده اضافه بار ارائه می‌شود. در این راستا، سرورهای IMS به عنوان عامل‌های هوشمند یادگیرنده در نظر گرفته شده که قابلیت مذاکره با بقیه عامل‌های شبکه را دارا می‌باشند. این عامل‌ها با تعامل با محیط پویا و از طریق سعی و خطای بدون ناظر، میزان بار قابل تحمل را یاد گرفته و از طریق مذاکره با بقیه عامل‌ها، موجب کنترل اضافه بار در IMS می‌گردند.

پیش‌زمینه‌ها به همراه کارهای مرتبط جهت ارائه روش پیشنهادی در

این مقاله در تاریخ ۱ شهریور ماه ۱۴۰۰ دریافت و در تاریخ ۱۷ بهمن ماه ۱۴۰۰ بازنگری شد.

مهدی خزائی (نویسنده مسئول)، گروه مهندسی کامپیوتر، دانشگاه صنعتی کرمانشاه، کرمانشاه، ایران، (email: m.khazaei@kut.ac.ir).

1. IP Multimedia Subsystem

جدول ۱: علایم و پارامترهای استفاده‌شده در مقاله.

پارامتر	توصیف
S	حالت فعلی عامل
r	پاداش دریافتی توسط عامل
a	عمل انتخابی توسط عامل
α	نرخ یادگیری عامل
γ	فاکتور تخفیف یادگیری
T_w	آستانه هشدار
T_c	آستانه اضطرار
S_j^i	تمدید j ام مربوط به ناحیه i ام
$States$	تعداد حالات موجود
G_i^{new}	گذردهی مفید عامل i ام قبل از عمل انتخاب‌شده
G_i^{old}	گذردهی مفید عامل i ام بعد از عمل انتخاب‌شده
$Load_i$	تعداد نشست‌های درون صف عامل i ام
$CPU_{Occupancy}^i$	درصد مشغول‌بودن CPU عامل i ام
$CPU_{Capacity}^i$	ظرفیت CPU عامل i ام
R	تعداد نشست‌های خارج از ناحیه امن عامل i ام
D	تعداد نشست‌های قابل پردازش عامل i ام در ناحیه امن
I	شماره دور مذاکره
β	ضریب کاهشی مذاکره

بخش دوم آورده شده است. در بخش سوم روش پیشنهادی ارائه گردیده و ارزیابی کارایی و تحلیل نتایج در بخش چهارم آمده است. نتیجه‌گیری و کارهای آینده نیز در بخش پنجم قرار دارند.

۲- پیش‌زمینه‌های مرتبط

از آنجا که هدف طراحی کنترل‌کننده اضافه بار IMS مبتنی بر عوامل هوشمند می‌باشد، لذا نیاز است مفاهیم و کارهای مرتبط به طور مختصر در ادامه توضیح داده شود. همچنین جدول ۱، علایم و پارامترهای استفاده‌شده در مقاله به همراه توصیف آنها را نشان می‌دهد.

۲-۱ یادگیری عامل‌ها

امروزه استفاده از سیستم‌های چندعامله برای حل مسایل پیچیده مورد توجه فراوان قرار گرفته است. یک سیستم چندعامله از چندین عامل تشکیل گردیده که سعی می‌کند مسایلی را که حل آنها برای یک سیستم متمرکز و یکپارچه، مشکل و گاهی غیر ممکن است به کمک یکدیگر حل نمایند. در این راستا می‌توان یک سیستم پیچیده را به مجموعه‌ای از کارهای کوچک‌تر تقسیم کرد تا هر عامل یک قسمت از کار را انجام دهد و از همکاری بین آنها به نتیجه مطلوب رسید.

عامل یک موجود نرم‌افزاری یا سخت‌افزاری تعریف می‌شود که در محیط واقع شده و برای رسیدن به اهدافش به صورت خودمختار فعالیت می‌کند. عامل از طریق حسگرهایش محیط را درک می‌نماید و از طریق محرک‌هایش بر محیط اثر می‌گذارد. هر چیز اطراف عامل به جز خودش را محیط عامل می‌گویند. قرارگرفتن عامل در یک محیط به این معنا است که عامل می‌تواند بخشی از محیط را مشاهده و یا در آن تغییری ایجاد کند. از طرفی، ارتباط در بسیاری از سیستم‌های چندعامله یک مفهوم مهم به شمار می‌رود. بدون وجود ارتباط، عامل‌ها باید فقط به قضاوت‌های مبتنی بر مشاهدات خود تکیه کنند، در صورتی که وجود ارتباط باعث شده که عامل‌ها قادر به گرفتن تصمیمات هماهنگ‌تری باشند. رسیدن به توافق در تصمیم‌گیری بدون دخالت دیگران، در حوزه همکاری بین

۱ مقداردهی اولیه به جدول Q
۲ دریافت حالت فعلی محیط (s)
تکرار حلقه زیر تا رسیدن به شرط جذب:
۳-۱ انتخاب عمل a
۳ ۳-۲ دریافت پاداش از محیط r
۳-۳ دریافت حالت جدید s'
۳-۴ تغییر مقدار جدول Q با توجه به (۱)

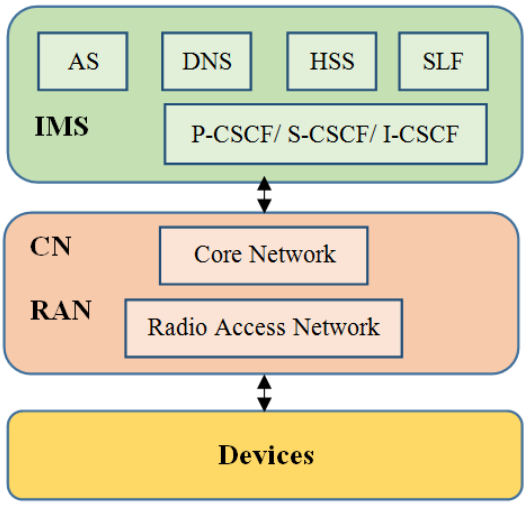
شکل ۱: الگوریتم روش یادگیری Q.

عامل‌هایی نمود می‌یابد که در آن، هدف عامل‌ها حصول یک سود دوجانبه در زمینه‌های مورد علاقه مشترک می‌باشد که مذاکره، روش غالبی برای دستیابی به توافق توسط عامل‌ها است. پروتکل مذاکره، مجموعه‌ای از قوانین است که تمام عامل‌ها آن را شناخته‌اند و در میان آنها حاکم است. در فرایند مذاکره، توافق زمانی حاصل می‌شود که بین پیشنهادهای مطرح‌شده اشتراکی وجود داشته باشد. عامل‌ها نسبت به نتایج مذاکره، اولویت شخصی دارند و همچنین می‌توانند نسبت به استفاده از پیشنهادها محدودیت‌هایی قایل شوند [۴] و [۵].

در مبحث یادگیری عامل‌ها، مسایلی هستند که منابع برای حل آنها ناقص یا کم می‌باشد و باعث شده که امکان استفاده از الگوریتم‌های یادگیری باناظر وجود نداشته باشد. همچنین گاهی اطلاع دقیقی از آنچه که باید یاد گرفته شود وجود ندارد که در این شرایط، یادگیری بدون ناظر قابل پیاده‌سازی خواهد بود. یادگیری تقویتی، یادگیری بدون ناظری است که عامل تلاش می‌کند تا تعامل خود با محیط پویا را از طریق سعی و خطا بهینه نماید. یادگیری تقویتی در واقع چگونگی نگاشت موقعیت‌های مختلف به اعمال، جهت حصول بهترین نتیجه با بیشترین پاداش است. در خیلی از موارد، اعمال نه تنها روی پاداش همان مرحله بلکه روی پاداش مراحل بعد هم تأثیرگذار است. دو خصوصیت "سعی و خطا" و "پاداش با تأخیر" مهم‌ترین خصوصیات یادگیری تقویتی می‌باشند. یادگیری تقویتی از این رو مورد توجه است که راهی برای آموزش عامل‌ها برای انجام یک عمل از طریق پاداش و تنبیه است، بدون این که لازم باشد نحوه انجام عمل را برای عامل مشخص نمود [۶]. یادگیری تقویتی استاندارد یا یادگیری Q، یک روش مستقل از مدل بوده که در آن عامل هیچ دسترسی به مدل انتقال ندارد. یادگیری Q بر اساس فرایند تصمیم مارکوف، با داشتن تابع پاداش و دریافت پاداش با تأخیر می‌تواند خط مشی بهینه را یاد بگیرد و عامل (عمل،حالت) $Q^*(s,a)$ را توسط تعامل پیوسته با محیط و به صورت سعی و خطا تخمین می‌زند. مراحل یادگیری Q مطابق الگوریتم شکل ۱ و معادله زیر است

$$Q(s,a) = \alpha \times (r + \max_{a'} Q(s',a')) + (1-\alpha) \times Q(s,a) \quad (1)$$

الگوریتم هر بار از یک حالت اولیه شروع می‌کند و با انجام یک سری اعمال و دریافت پاداش به حالت هدف می‌رسد. در حالت هدف، حالت عامل با انجام هر عملی تغییر نکرده و هیچ پاداشی از محیط دریافت نمی‌کند که به این حالت، حالت جذب گفته می‌شود. انتخاب عمل می‌تواند بر اساس کاوش و یا بهره‌برداری صورت گیرد. انتخاب عمل به صورت کاوش یعنی بدون توجه به مقادیر جدول Q، عملی به طور تصادفی انتخاب می‌شود. با این کار ممکن است اعمال بهینه‌ای که تا کنون انتخاب نشده‌اند کشف و به جدول اضافه شوند. ولی در انتخاب عمل به فرم بهره‌برداری، بهترین عمل مطابق جدول Q ساخته‌شده انتخاب خواهد گردید. در (۱)، α نرخ یادگیری عامل است و مقداری بین صفر و یک می‌گیرد. مقدار یک سبب شده که عامل فقط جدیدترین اطلاعات را



شکل ۲: معماری ساده هسته IMS.

در نظر گرفته و مقدار صفر باعث می‌شود که عامل هیچ یادگیری نداشته باشد. پارامتر γ نیز که مقداری بین صفر و یک دارد، فاکتور تخفیف نامیده می‌شود و برای مشخص‌کردن پاداش‌های آینده است. مقدار صفر یعنی عامل فقط پاداش فعلی را در نظر می‌گیرد ولی مقادیر نزدیک به یک سبب شده که عامل برای رسیدن به پاداش بیشتر، زمان بیشتری انتظار بکشد. اگر تمامی زوج‌های (عمل، حالت) به صورت مکرر تجربه شوند و نرخ یادگیری در طول زمان کاهش یابد، یادگیری Q با احتمال یک به مقدار بهینه $Q^*(s,a)$ همگرا می‌شود. از کاربردهای یادگیری تقویتی به عنوان نمونه می‌توان به تداوم خدمات در IMS، سیستم‌های حمل و نقل هوشمند و کنترل ترافیک شهری، مدیریت شبکه‌های حسگر توزیع‌شده بی‌سیم و سیستم‌های نرم‌افزاری پیچیده اشاره کرد [۷] تا [۱۲].

IMS ۲-۲

برخلاف کاربردهای سنتی، هدف IMS ادغام انواع مختلف خدمات و کاربردهای چندرسانه‌ای و همگرایی بین شبکه‌های سیمی، بی‌سیم و سیار است. IMS یک شبکه سوییچینگ بسته‌ای است که بر روی بستر پروتکل IP گسترش یافته و معماری سه‌لایه‌ای دارد که شامل لایه کاربرد، لایه کنترل و لایه کاربر می‌باشد. وظیفه لایه کاربرد این است که به کاربران، اجازه دسترسی به برنامه‌های چندرسانه‌ای و نشست‌های فراهم‌شده توسط سرورهای کاربردی را بدهد که این دسترسی از طریق پایانه‌های ثابت یا سیار امکان‌پذیر می‌باشد. از طریق این لایه، استفاده از چندین سرویس در یک نشست یا چندین نشست هم‌زمان توسط کاربران امکان‌پذیر خواهد بود. وظیفه لایه کنترل، مدیریت و کنترل نشست‌ها و اطلاعات کاربران است. لایه کاربر، مسئول فراهم‌کردن ارتباط با کاربران بوده و کاربران با استفاده از شبکه‌های مبتنی بر IP به شبکه IMS متصل می‌شوند. علاوه بر یکپارچه‌سازی، کنترل نشست و توسعه سرویس‌ها، تضمین کیفیت سرویس و امکان محاسبه هزینه‌ها تحت معیار واحد، از دیگر ویژگی‌های IMS می‌باشد. معماری هسته IMS در شکل ۲ به نمایش درآمده است [۱۳].

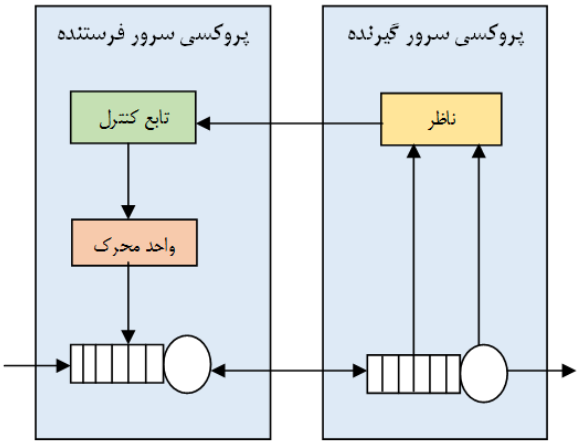
همان طور که در شکل مشخص شده است، دو پایگاه داده HSS[1] و SLF[2] در معماری IMS وجود دارند. HSS منبع اصلی ذخیره‌سازی اطلاعات مشترکان و سرویس‌های مربوط به آنها می‌باشد. وقتی چندین HSS توسط مدیر شبکه استفاده می‌شود، پایگاه داده SLF برای یافتن آدرس HSSای که اطلاعات مشترک مورد نظر را نگهداری می‌کند، استفاده می‌گردد. ASها سرویس‌های ارزش افزوده چندرسانه‌ای در IMS را ارائه می‌دهند. توابع کنترل نشست تماس (CSCF)[3]، پروکسی سرورهای SIPی هستند که هر یک وظیفه خاص خود را دارند که وجه مشترک آنها نقشی است که در هنگام فرایند ثبت نام، ایجاد نشست و مسیریابی SIP ایفا می‌کنند که در ادامه مختصر به عملکرد هر یک پرداخته می‌شود:

- P-CSCF[4] یک سرور حالتمند SIP است که اولین نقطه تماس کاربران با شبکه IMS می‌باشد. همه ترافیک کاربران به این سرور منتقل شده و همچنین ترافیک شبکه از طریق این سرور به کاربران منتقل می‌گردد.
- S-CSCF[5] یک سرور حالتمند SIP است که همیشه در شبکه خانگی مشترک قرار دارد. این سرور نقطه مرکزی IMS محسوب شده که وظایفی از قبیل مدیریت فرایند ثبت نام، تصمیم‌گیری مسیریابی، نگهداری حالت نشست‌ها و ذخیره اطلاعات سرویس‌ها را بر عهده دارد. تمام بسته‌های سیگنالینگ SIP مرتبط با ترمینال‌های IMS از S-CSCF می‌گذرند تا با پردازش محتویات بسته‌ها، اقدامات بعدی مشخص گردد.
- I-CSCF[6] یک سرور بدون حالت SIP است که نقطه تماس با یک اپراتور بوده که بتوان از طریق آن با مشترکان درون آن اپراتور اتصال ایجاد کرد. I-CSCF در هنگام ثبت نام با دریافت اطلاعات از HSS، یک سرور S-CSCF را بر اساس سیاست‌های تعریف‌شده، تخصیص می‌دهد. از دیگر وظایف این سرور، به دست آوردن گام بعدی در مسیریابی از طریق HSS و همچنین هدایت درخواست‌های رسیده به S-CSCF و یا AS تخصیص داده شده است.

فرایندهای اصلی سیگنالینگ در IMS شامل ثبت نام و حذف آن، شروع و پایان نشست، به هم زدن نشست و راهنمایی مجدد نشست می‌باشد. کاربر پس از دسترسی به شبکه IP و در حین دریافت آدرس IP، آدرس P-CSCF مناسب را دریافت می‌کند. تمامی کاربران قبل از آغاز هر نشست باید در IMS ثبت نام کنند. پس از ثبت نام موفق کاربر، P-CSCF کاربر با توجه به بسته پاسخ رسیده می‌داند که به هر کاربر کدام S-CSCF تخصیص یافته و همچنین هر S-CSCF نیز می‌داند که برای دسترسی به هر کاربر باید با کدام P-CSCF ارتباط برقرار نماید که از این اطلاعات برای برقراری یک نشست استفاده می‌شود.

زمانی که کاربر A می‌خواهد یک نشست با کاربر B برقرار کند، یک بسته درخواست INVITE را به P-CSCFی که IMS در موقع اتصال به وی تخصیص داده است، ارسال می‌کند. P-CSCS پس از پردازش‌های لازم، بسته را به S-CSCFی که در فرایند ثبت نام به کاربر تخصیص داده شده است، ارسال می‌نماید. S-CSCF پس از پردازش بسته، بر اساس شناسه کاربری B، I-CSCF مربوط در حوزه B را یافته و بسته را به آن تحویل می‌دهد. I-CSCF با تماس با HSS، سرور S-CSCF تخصیص داده شده به B را یافته و بسته را به آن تحویل می‌دهد. S-CSCF نیز با اطلاعاتی که در زمان ثبت نام به دست آورده است، بسته INVITE را به P-CSCF کاربر B تحویل داده و از آنجا نیز بسته

1. Home Subscriber Server
2. Subscription Locator Function
3. Call Session Control Functions
4. Proxy CSCF
5. Serving CSCF
6. Interrogating CSCF



شکل ۳: معماری یک کنترل‌کننده اضافه بار.

به B ارسال می‌گردد. پس از دریافت INVITE توسط B، این کاربر بسته پاسخ را تولید می‌نماید و آن را به سوی A از همان مسیری که INVITE رسیده است، ارسال می‌کند. پس از تبادل تعدادی بسته به صورت رفت و برگشت، یک نشست بین A و B بر اساس پروتکل SIP در CSCFها تشکیل خواهد شد [۱].

SIP یک پروتکل لایه کاربرد مبتنی بر متن است که جهت ایجاد، نگهداری، تغییر و خاتمه نشست‌های چندرسانه‌ای استفاده و توسط IETF در RFC ۳۲۶۱ استاندارد گردیده است. منظور از تراکنش SIP، یک درخواست و تمامی پاسخ‌های مرتبط با آن است که بین دو عنصر مجاور SIP رد و بدل می‌شوند. پروکسی سرورها بسته به شرایط و نیاز شبکه، به صورت حالتمند و یا بدون حالت پیکربندی می‌شود. پروکسی سرور حالتمند، اطلاعات حالت هر تراکنش را برای انجام برخی عملیات‌ها، مانند ارسال مجدد بسته‌های سیگنالینگ نگهداری می‌کند. اما در پروکسی سرور بدون حالت، هیچ تراکنشی روی پروکسی سرور ایجاد نشده و پروکسی سرور تنها وظیفه دریافت پیام و مسیریابی آن را بر عهده دارد [۱۴].

از آنجا که در اکثر مواقع، مکانیزم کنترل اضافه بار تعبیه‌شده در SIP نمی‌تواند با حجم بار اضافی مقابله کند، لذا نیاز به کنترل‌کننده اضافه بار در IMS اجتناب‌ناپذیر است [۱۵]. کنترل‌کننده اضافه بار از سه جزء اصلی تشکیل می‌شود. واحد ناظر، مسئول نظارت و جمع‌آوری اطلاعات از پارامترهایی است که توسط طراح تعیین گردیده است. این واحد اطلاعات جمع‌آوری‌شده را در اختیار تابع کنترل قرار می‌دهد. تابع کنترل بر اساس یک الگوریتم تعریف‌شده توسط طراح، سیاست و چگونگی میزان دریافت بار را تعیین می‌کند و این سیاست را در اختیار واحد محرک قرار می‌دهد. واحد محرک بر اساس سیاست‌های رسیده، بار اضافه را رد می‌کند. شکل ۳ یک کنترل‌کننده اضافه بار را نشان داده که واحد ناظر در سرور گیرنده و تابع کنترل و محرک در سرور فرستنده قرار دارند [۱۶].

اگر کنترل‌کننده در یک سرور قرار داشته باشد روش، کنترل داخلی نامیده شده و در غیر این صورت جهت مقابله با اضافه بار سرورها با هم همکاری می‌کنند و کنترل خارجی نامیده می‌شود. کنترل خارجی به دو دسته گام به گام و انتها به انتها تقسیم می‌گردد. در کنترل اضافه بار به روش گام به گام، سرور تحت اضافه بار می‌تواند به صورت مستقیم به سرور بالادستی خود اطلاع دهد که چه مقدار درخواست را برایش ارسال نماید که دچار اضافه بار نشود. از آنجا که روش ارائه‌شده در این تحقیق یک روش گام به گام برای سرورهای قطاری متوالی در IMS است، در ادامه چند مورد از کارهای شاخص مرتبط با تحقیق مورد بررسی قرار خواهند گرفت [۱۷].

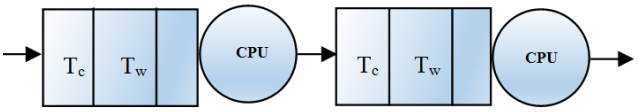
در [۱۸]، میزان اشغال پردازنده در سرور بالا و پایین‌دستی حساب شده و با ارسال این مقدار از طریق بازخورد در بازه‌های زمانی، نرخ ورود درخواست نشست به سرور پایین‌دست کنترل می‌شود. در [۱۶] از دو روش صریح اطلاع‌رسانی به سرور بالادست استفاده می‌گردد. در روش اول، وقتی مدت لازم جهت پردازش پیام‌های درون صف از حد آستانه بیشتر شود، درخواست‌ها با پیام ۵۰۳ رد می‌شوند و مقدار تأخیر را برای سرور بالادست می‌فرستد و سرور بالادست در این مدت تعیین‌شده از ارسال درخواست به سرور دچار اضافه بار خودداری می‌کند. در روش دوم این مرجع، بر اساس میزان درصد اشغال CPU، نرخ دریافت درخواست‌ها را محاسبه و برای سرورهای بالادست ارسال می‌کند. سرورهای بالادست نیز به اندازه مشخص‌شده برای سرور پایین‌دست بسته ارسال می‌کنند. در این مقاله همچنین یک روش ضمنی مبتنی بر پنجره ارائه شده که اندازه پنجره در سرور بالادستی با توجه به پیام‌های دریافتی و منقضی‌شدن تایمرها تعیین می‌گردد. در [۱۹]، یک روش ضمنی مبتنی بر پنجره بر اساس تعداد تصدیق‌های دریافتی آورده شده که اگر در سرور بالادست، نرخ تعداد پیام‌های تصدیق دریافتی نسبت به درخواست‌های ارسالی از حد آستانه بیشتر شود، طول پنجره را افزایش داده و اگر کمتر باشد، طول پنجره را نصف می‌کند و در صورتی که پیام ۵۰۳ را دریافت نماید، اندازه پنجره صفر می‌شود تا درخواستی ارسال نگردد. در [۲۰]، یک روش مدیریت جریان مطرح شده که در آن سرور ارسال‌کننده، CPU سرور تحت

اضافه بار را رصد کرده و سپس با طبقه‌بندی بسته‌ها و حذف هوشمندانه بسته‌های ارسال مجددشده و کنترل تماس‌های فعال، از مواجهه سرور پایین‌دست با اضافه بار جلوگیری به عمل می‌آورد. در [۲۱]، یک روش صریح ارائه شده که بر اساس نرخ ارسال عمل می‌کند. مکانیزم اضافه بار در سرور بالادستی بر اساس پیام‌های ۵۰۳ دریافتی سعی می‌کند که اضافه بار در سرور پایین‌دست را پیش‌بینی و بر اساس آن نرخ ارسال را تنظیم نماید. در [۱۶]، ۳ روش مبتنی بر پنجره پیشنهاد شده که در آنها سرور تحت اضافه بار به طور پویا و مستمر، ظرفیت پاسخگویی کنونی خود را تخمین می‌زند و تحت عنوان تعداد پنجره‌های موجود، نرخ بار قابل تحمل را به سرورهای بالادست اطلاع می‌دهد. در [۲۲]، یک روش ضمنی مبتنی بر پنجره ارائه شده که از تأخیر ایجاد نشست به عنوان معیار تعیین اندازه پنجره به شیوه مشابه کنترل تصادم در TCP استفاده می‌کند. در [۲۳]، یک روش ضمنی ارائه گردیده که از مکانیزم سطل سوراخ‌دار جهت ارسال درخواست‌های نشست به سرور پایین‌دستی استفاده می‌کند. سرور بالادست نرخ ارسال را به روش تدریجی بر اساس دریافت و یا عدم دریافت پیام‌های ۵۰۳ تنظیم می‌کند. مرجع [۲۴] با مدل‌کردن فعل و انفعالات بین سرور پایین‌دستی و بالادستی، یک سیستم کنترل بازخوردی متناسب با انتگرال را در قالب دو الگوریتم ارائه کرده تا با تنظیم نرخ ارسال‌های مجدد، اضافه بار را کاهش دهد. در [۲۵] و [۲۶] از مفهوم شبکه‌های نرم‌افزاری و توابع مجازی‌ساز جهت مقابله با اضافه بار در SIP استفاده شده است.

۳- طراحی کنترل‌کننده بار

در بسیاری از الگوریتم‌های کنترل بار، پارامترهایی وجود دارند که کارایی الگوریتم به طور چشم‌گیری به مقادیر این پارامترها وابسته است. مثلاً در الگوریتم‌های ارائه‌شده مبتنی بر اشغال CPU [۳]، یک مقدار آستانه برای میزان مشغول‌بودن CPU در نظر گرفته شده که هر گاه درصد اشغال CPU از این آستانه بیشتر شود، الگوریتم کنترل اضافه بار

عمل می‌کند و یا در روش مبتنی بر پنجره [۲۲]، یک مقدار آستانه تأخیر



شکل ۴: دو پروکسی سرور حالتمند قطاری IMS.

```
BEGIN

  if (Numer_Invites >= Tc)
     While (Number_Invites <=Tc)
        Reject_Invites_503_Messages();

  else if (Numer_Invites>=Tw)
          Start_Negotiations_upstream();

          else
             Do_Nothings();

END
```

شکل ۵: الگوریتم کنترل بار.

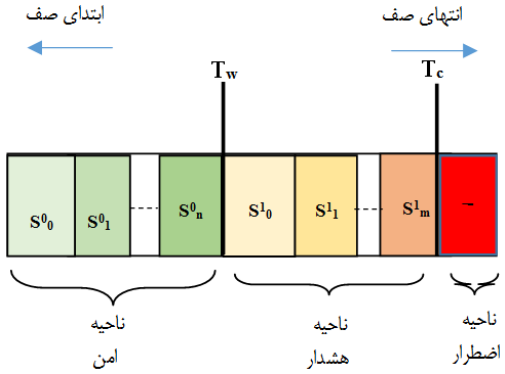
در سرور بالادست تعیین شده که بر اساس مقایسه تأخیر پاسخ‌های رسیده با این آستانه، اندازه پنجره در سرور بالادست تعیین می‌شود و کارایی الگوریتم ارائه‌شده کاملاً به مقدار این آستانه وابسته است.

بهترین و دقیق‌ترین روش برای محاسبه مقادیر پارامترهای ذکرشده، استفاده از روابط ریاضی است. ولی اکثراً به دلیل پیچیدگی و یا غیر مستدل بودن، نمی‌توان با استفاده از روابط ریاضی مقادیر مطلوب را به دست آورد. راه حل دیگر برای تعیین مقادیر پارامترها، روش سعی و خطا می‌باشد. یعنی با استفاده از شبیه‌سازی و قراردادن مقادیر مختلف بررسی گردد کدام یک، جواب بهتری تولید می‌کند. مشکل این روش آن است که مقادیر به دست آمده فقط برای آن حالت شبکه مناسب بوده و اگر شرایط شبکه تغییر نماید، دیگر جواب‌های مطلوبی تولید نمی‌شود و مجدداً باید محاسبات انجام پذیرد. در چنین شرایطی می‌توان از روش جدیدی مبتنی بر یادگیری استفاده کرد.

در IMS سه سرور SIP متوالی (S-CSCF, I-CSCF, S-CSCF) همان طور که اشاره شد، وجود دارد که سرور I-CSCS بدون حالت بوده و در اضافه بار نقش ندارد، اما دو سرور قطاری S-CSCF حالتمند هستند و در تشکیل و مدیریت نشست‌ها نقش دارند که پروکسی سرور نامیده می‌شوند. از آنجا که فاصله دو سرور یک گام است کنترل‌کننده، روش گام به گام را جهت کنترل بار استفاده خواهد کرد که در شکل ۴ نشان داده شده است. همان طور که در شکل مشاهده می‌شود، برای صف هر پروکسی سرور دو آستانه هشدار (T_w) و اضطرار (T_c) تعریف شده که تصمیم‌گیری و واکنش کنترل‌کننده مطابق الگوریتم شکل ۵ به مقادیر این دو آستانه وابسته است که در هر دو عامل اجرا می‌شود.

عملکرد الگوریتم به صورت خلاصه به این صورت است که اگر تعداد درخواست‌های تشکیل نشست درون صف بیشتر از T_c باشد، درخواست‌ها به صورت محلی توسط پیام ۵۰۳ رد می‌شوند. اگر تعداد بین دو آستانه باشد، فرایند مذاکره با سرور بالادستی فعال می‌گردد، ولی اگر تعداد درخواست‌ها کمتر از T_w باشد هیچ واکنشی صورت نمی‌گیرد.

در کنترل‌کننده طراحی‌شده، واحد ناظر، تابع کنترل و واحد محرک در خود سرور قرار داشته که می‌توان آنها را به عنوان یک واحد مجزا نیز در نظر گرفت. واحد ناظر به اطلاعات طول صف و نرخ درخواست‌های ورودی و خروجی صف هر سرور احتیاج دارد که این اطلاعات را از ماشین حالات تعریف‌شده در سرورها توسط پروتکل SIP (مطابق RFC ۳۲۶۱)



شکل ۶: تخصیص حالات به صف عامل.

به دست آورده و نیاز به سرکشی مداوم و سربار مازاد نیست. تابع کنترل، الگوریتم یادگیری را پیاده‌سازی کرده و واحد محرک، فرایند مذاکره با سرور بالادستی را انجام می‌دهد [۲۷] و [۲۸].

در این راستا، هر پروکسی سرور به عنوان یک عامل هوشمند با قابلیت یادگیری و مذاکره تعریف می‌شود. این عامل‌ها با دریافت بسته‌ها و نظارت بر منابع خود، دانشی را از محیط کسب می‌کنند و بر اساس یادگیری Q، مقادیر T_w و T_c صف خود را می‌آموزند. از لازمه‌های طراحی یک الگوریتم یادگیری Q، تعریف مناسب فضای حالت، فضای اعمال و تابع پاداش است. در روش یادگیری استفاده‌شده سعی گردیده که فضای حالات و اعمال تا حد امکان کوچک تعریف شوند تا همگرایی الگوریتم تسریع گردد.

فضای حالت

جهت تعریف فضای حالات مطابق شکل ۶، ۳ ناحیه امن، هشدار و اضطرار و ۲ آستانه هشدار (T_w) و اضطرار (T_c) در صف پروکسی سرور تعریف می‌شود. جهت تعیین مقادیر T_w و T_c به ناحیه امن و هشدار، تعدادی حالت تخصیص می‌یابد. حالت اول $(i=0)$ و تمدیدهای آن در ناحیه امن صف و حالت دوم $(i=1)$ و تمدیدهای آن در ناحیه هشدار صف قرار می‌گیرند. S_0^i حالت شروع هر ناحیه و فاصله تا ابتدای آن ناحیه را مشخص می‌کند و S_j^i تمدید شماره j ام برای ناحیه i ام است. برای ناحیه امن n تمدید و برای ناحیه هشدار m تمدید در نظر گرفته شده است. تعداد کل حالات در نظر گرفته شده از (۲) به دست می‌آید

$$|States| = \sum_{i=0}^{n} S_i^0 + \sum_{i=0}^{m} S_i^1 \qquad (۲)$$

در طول فرایند یادگیری، هر یک از این فضاهای حالت معرف طول مشخصی از صف برای تعیین مقدار T_w و T_c است. به عنوان مثال، اگر مقدار S_0^0 برابر ۷ باشد و هر یک از حالات تمدید برابر ۲ تعریف شده باشد، مقدار T_w برابر ۷ است و در صورت رفتن به اولین تمدید مقدار آن ۹ می‌شود.

فضای عمل

در روش یادگیری پیشنهادشده برای هر عامل ۶ عمل تعریف می‌گردد که در موقعیت‌های مناسب روی حالت فعلی اعمال می‌شوند. این ۶ عمل عبارت هستند از افزایش، کاهش و ثبات در ناحیه امن و همچنین افزایش، کاهش و ثبات در ناحیه هشدار. افزایش به این معنا است که از حالت فعلی به حالت بعدی انتقال انجام گیرد. کاهش یعنی از حالت فعلی به حالت قبلی منتقل شود و ثبات یعنی در اثر این عمل حالت تغییر نمی‌کند. اعمال از طریق کاوش انتخاب می‌شوند.

```
BEGIN
I=1;
DownStreamServer_Calculate_R_To_UpstreamServer();
while (R!=0) {
   if (UpstreamServer != SafeRegion) {
      if (UpstreamServer == P-CSCF)
         Reject_R_Invites_locally();
      else {
         Calculate_R();
         Send_R_To_UpstreamServer();
      }
   }
   else {
      Calculate_D();
      Send_D_To_DownStreamServer();
      DownStreamServer_Reject_(R-D)_Invites_locally();
      if (DownstreamServer == SafeRagin) {
         R=0;
         DownStreamServer_R_To_UpstreamServer();
      }
      else {
         Update_R();
         DownStreamServer_R_To_UpstreamServer();
      }
   }
   I=I+1;
}
END
```

شکل ۷: پروتکل فرایند مذاکره جهت کاهش بار.

جدول Q

این جدول در طول فرایند یادگیری برای ذخیره و به روز رسانی مقادیر Q بر اساس (۱) مورد استفاده قرار می‌گیرد. اگر این جدول به صورت ماتریسی دوبعدی در نظر گرفته شود، اندیس سطرها، حالات و اندیس ستون‌ها، اعمال را مشخص می‌کنند. اندازه ماتریس جدول Q برابر $states \times ۶$ است که مقدار $states$ همان مقدار (۲) می‌باشد. مقدار اولیه جدول Q صفر در نظر گرفته می‌شود.

تابع پاداش

عمل هدایت عامل در فضای حالات و اعمال به منظور دستیابی به یک سیاست بهینه توسط تابع پاداش صورت می‌گیرد. در واقع تابع پاداش است که مشخص می‌کند عملی که یک عامل با قرارگرفتن در یک حالت انجام می‌دهد، چقدر او را به هدف نزدیک‌تر می‌کند. در روش یادگیری ارائه‌شده از مفهوم گذردهی مفید توسط عامل به عنوان تابع پاداش استفاده می‌شود. میزان پاداشی که یک عامل برای انجام عملی دریافت می‌کند، متناسب است با میزان افزایشی که در گذردهی مفید خود نسبت به حالت قبل می‌دهد. اگر عامل در اثر این عمل باعث شود که گذردهی مفید در عامل بالادست نیز افزایش یابد، پاداش بیشتری دریافت می‌کند ولی اگر باعث کاهش گذردهی مفید در عامل بالادست شود، پاداش کمتری نصیب او می‌شود. تابع پاداش برای عامل iام به صورت (۳) تعریف می‌شود

$$Reward_i = (G_i^{new} - G_i^{old}) - (G_j^{old} - G_j^{new}) \qquad (۳)$$

که G_i^{old} و G_i^{new} به ترتیب گذردهی مفید عامل iام قبل و بعد از عمل انتخاب‌شده روی حالت فعلی است. با توجه به تعریف تابع پاداش، مشخص است که مقادیر این تابع بعد از اجرای عمل انتخاب‌شده قابل محاسبه می‌باشد، لذا مقادیر جدول Q با یک گام تأخیر به روز می‌شوند که تأثیری در فرایند یادگیری ندارد [۲۹].

تابع تغییر حالت

یکی دیگر از الزامات طراحی الگوریتم یادگیری Q، تعریف نحوه تغییر حالت با انجام عملی مشخص روی حالت فعلی است. در روش یادگیری استفاده‌شده، حالات با توجه به عمل انتخابی به صورت زیر تغییر می‌کنند:

۱) اگر عمل انتخابی افزایش در ناحیه امن و یا هشدار باشد و تعداد تمدیدهای انجام‌شده از حداکثر تعداد تمدیدهای در نظر گرفته شده کمتر باشد، حالت بعدی تمدید خواهد بود که در (۴) آورده شده است. ولی اگر عامل در آخرین تمدید مجاز باشد، حالت بعدی همان حالت فعلی است

$$Transfer = (S_j^i, a) = S_{j+۱}^i \quad ; \ i = ۰ \text{ or } ۱ \ , \ S_j^i \neq S_n^i \text{ and } S_j^i \neq S_m^i \qquad (۴)$$

۲) اگر عمل انتخابی کاهش در ناحیه امن و یا هشدار باشد، حالت بعدی تمدید قبلی و در صورت قرارداشتن در اولین تمدید، حالت شروع $S_۰^i$ خواهد بود. اگر حالت فعلی، حالت شروع باشد، حالت بعد همان حالت شروع است که (۵) این انتقال را نشان می‌دهد

$$Transfer = (S_j^i, a) = S_{j-۱}^i \quad ; \ i = ۰ \text{ or } ۱ \text{ and } S_j^i \neq S_۰^i \qquad (۵)$$

۳) اگر عمل انتخابی ثبات در ناحیه امن و یا هشدار باشد، حالت بعد همان حالت فعلی خواهد بود که در (۶) آمده است

$$Transfer = (S_j^i, a) = S_j^i \quad ; \ i = ۰ \text{ or } ۱ \qquad (۶)$$

به دست آوردن مقادیر T_w و T_c

بعد از مرحله یادگیری، مقادیر مناسب T_w و T_c هر عامل باید با توجه به آنچه که او فراگرفته استخراج شود. به این منظور از اطلاعات نهایی جدول Q عامل استفاده می‌شود، به این صورت که برای هر ناحیه از حالت شروع $S_۰^i$ آغاز کرده و با توجه به مقادیر جدول و انتخاب عمل مربوط و توجه به بیشترین مقدار در هر سطر، مجموعه حالات تمدید برای آن ناحیه محاسبه می‌شود.

مذاکره جهت اعمال سیاست کنترل‌کننده

در فرایند مذاکره، یک مجموعه از عامل‌ها به همراه مجموعه‌ای از متغیرها شرکت دارند که هر متغیر به مجموعه‌ای از عامل‌ها وابسته است. عامل‌ها بر سر مجموعه‌ای از امکان‌ها (مقادیر) مذاکره می‌کنند و برای رسیدن به توافق، امکان‌ها از طریق مذاکره به متغیرها نسبت داده می‌شوند. در کنترل‌کننده، مجموعه عامل‌های شرکت‌کننده در مذاکره، عامل‌های شکل ۴ در نظر گرفته می‌شود. متغیرها، اندازه بار ارسالی به عامل پایین‌دستی است و امکان‌ها، مقادیر پیشنهادی توسط عامل‌ها برای به دست آوردن میزان بار ارسالی یا دریافتی می‌باشد. عامل‌ها بر اساس یک استراتژی تعریف‌شده که در پروتکل شکل ۷ آمده است، در مذاکره شرکت می‌کنند.

یک دور پیشنهاد، شامل پیشنهاد آغازکننده و پاسخ بقیه عامل‌ها به آغازکننده می‌باشد. در اینجا آغازکننده، عاملی است که با عبور بار از T_w وارد ناحیه هشدار شده و پاسخ‌دهنده عامل‌های بالادست هستند. خلاصه عملکرد پروتکل شکل ۷ به صورت زیر است:

عامل پایین‌دست j از عامل بالادست i می‌خواهد که در بازه زمانی Δt، تعداد درخواست‌های تشکیل نشست ارسالی خود را به اندازه R کاهش دهد و R از (۷) به دست می‌آید. $Load$ تعداد نشست‌های صف عامل می‌باشد

جدول ۲: مقادیر پارامترهای فرایند یادگیری در OCL.

	پارامتر	مقدار
	حالت اول ناحیه امن	۱۰
	طول هر تمدید ناحیه امن	۱
	تعداد تمدیدهای ناحیه امن	۱۵
	حالت اول ناحیه هشدار	$۷+T_w$
پارامترهای الگوریتم	طول هر تمدید ناحیه هشدار	۱
یادگیری Q در OCL	تعداد تمدیدهای ناحیه هشدار	۱۵
	پارامتر α در (۱)	۰/۷
	نرخ یادگیری	۰/۹
	فاکتور تخفیف (γ)	۰/۴
	نرخ اکتشاف	۰/۷

$$R=\frac{load_j-Tw_j}{\Delta t}\times(1-CPU^j_{Occupancy})\times CPU^j_{Capacity} \qquad (۷)$$

اگر عامل i در ناحیه امن نباشد، از عامل بالادست خود که یک P-CSCS است می‌خواهد تا از بین درخواست کابران به اندازه $R_{ij}+R_{pi}$ درخواست را به صورت محلی رد نماید. در غیر این صورت اگر عامل i در ناحیه امن باشد، طبق (۸) محاسبه می‌کند که در مدت Δt با توجه به ظرفیت CPU خود چه میزان نشست را می‌تواند پردازش کند که از ناحیه امن خارج نشود، سپس مقدار D را به اطلاع عامل j می‌رساند

$$D=\frac{TW_i-load_i}{\Delta t}\times(1-CPU^i_{Occupancy})\times CPU^i_{Capacity} \qquad (۸)$$

عامل j با دریافت پاسخ عامل i به اندازه $R_{ij}-D_{ij}$ درخواست را به صورت محلی از درخواست‌های انتهای صف خود حذف می‌نماید. اگر وارد ناحیه امن شد، مقدار R_{ij} برابر صفر را برای عامل i ارسال کرده و مذاکره خاتمه می‌یابد. در غیر این صورت طبق (۹) میزان کاهش بار از سوی عامل i را به روز کرده و برای آن ارسال می‌کند

$$R=(\sum_{k=1}^{I}R^k_{ij}-D^k_{ij})\times\beta\times I \qquad (۹)$$

که β ضریب کاهشی و I شماره دور مذاکره است. روند فوق تا اتمام مذاکره تکرار می‌شود.

٤- ارزیابی کارایی

جهت ایجاد بستر شبیه‌سازی، پروتکل SIP بر اساس RFCهای ۳۲۶۱ و ۶۰۲۶ در NS-۲ پیاده‌سازی گردیده و پروتکل UDP به عنوان پروتکل لایه انتقال در نظر گرفته شده است. ظرفیت پردازشی سرورها معادل ۳۰۰ نشست در ثانیه (SPS) می‌باشد. کاربرها می‌توانند هم‌زمان چندین درخواست نشست را ارسال و یا دریافت کنند. عامل‌های کاربر، ظرفیت نامحدود دارند. در عامل سرور اولویت پردازش با پیام‌های مذاکره است، زیرا عدم رسیدگی به موقع به این پیام‌ها باعث رخداد اضافه بار در شبکه می‌گردد. از آنجا که پیام‌های کنترلی، متنی هستند و تراکنشی را در سرور ایجاد نمی‌کنند، لذا در تشدید اضافه بار نقشی نخواهند داشت. طول صف هر پروکسی سرور، محدود در نظر گرفته شده که اگر هنگام ورود درخواستی صف پر باشد، درخواست از بین برود. گذردهی مفید، تأخیر برقراری نشست، تعداد ارسال‌های مجدد، پایداری و عکس‌العمل سریع، معیارهای اصلی جهت ارزیابی کارایی کنترل‌کننده اضافه بار با قابلیت یادگیری[1] (OCL) هستند.

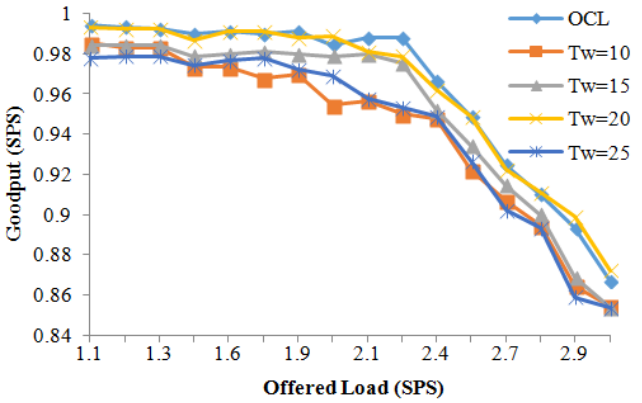
منظور از گذردهی مفید، تعداد نشست‌های موفقی است که عامل سرور قادر می‌باشد که در واحد زمان به آنها رسیدگی کند. یک نشست زمانی موفق است که پاسخ متناظر درخواست تشکیل نشست در کمتر از ۱۰ ثانیه دریافت شود. تأخیر برقراری نشست، مدت زمانی است که طول می‌کشد تا یک نشست ایجاد گردد. زمان پاسخگویی به نشست‌ها در گیرنده به صورت نمایی و با میانگین ۳۰ ثانیه در نظر گرفته شده است. تعداد نشست‌های ارسال مجددشده با تعداد نشست‌های ردشده نسبت مستقیم دارد که هرچه نشست‌های کمتری رد شوند، یعنی درخواست کمتری ارسال مجدد شده و بالعکس. پایداری یعنی کنترل‌کننده اضافه بار باید به گونه‌ای طراحی شود که سبب نوسان گذردهی روی پروکسی سرورها نشود و همچنین از صفرشدن گذردهی مفید پیشگیری کند. OCL باید قادر باشد که نسبت به وقوع ناگهانی اضافه بار به سرعت واکنش نشان دهد، به طوری که یک جهش ناگهانی در میزان ترافیک، سرور را با اضافه بار و از کار افتادگی مواجه نکند. به طور مشابه، اگر ترافیک تحمیلی به سرور به طور ناگهان کاهش یابد، تمامی کنترل‌های اعمال‌گردیده روی نشست‌های دریافتی باید به سرعت برداشته شده و وضعیت به حالت عادی برگردد. بار ورودی به صورت توزیع پواسن به روش قبول و رد به شبکه وارد شده و معیارهای ارزیابی کارایی با فاصله اطمینان ۹۵٪ استخراج شده‌اند [۳۰] و [۳۱]. در نمودارها بار ورودی بر ظرفیت شبکه تقسیم و نرمال شده و پارامترهای یادگیری Q استفاده‌شده در OCL، مطابق جدول ۲ هستند. برای به دست آوردن مقادیر T_w و T_c، حالات مختلف ورودی در نظر گرفته شده که با اجراهای متوالی، مقادیر این دو آستانه همگرا و در OCL به کار گرفته می‌شوند.

٤-۱ بررسی صحت عملکرد OCL

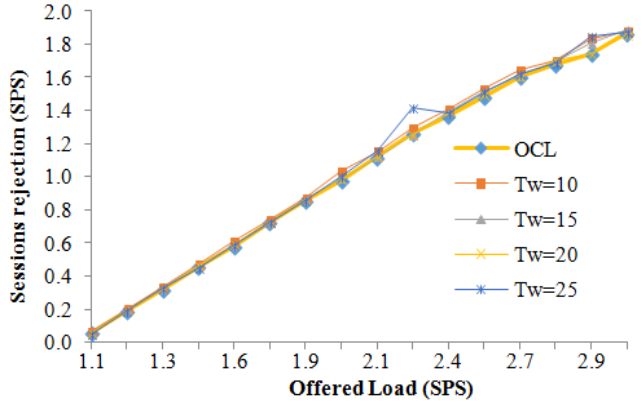
جهت بررسی عملکرد OCL نسبت به کنترل‌کننده بدون قابلیت یادگیری (OC)، میانگین و واریانس گذردهی مفید، تأخیر برقراری نشست‌ها و تعداد نشست‌های ارسال مجددشده در جدول ۳ آمده است. در OC مقادیر بهینه T_c برابر ۴۰ و T_w برابر ۱۰، ۱۵، ۲۰ و ۲۵ با سعی و خطا به دست آمده است. لذا روشی عملکرد بهتری دارد که میانگین بهتر و واریانس کمتری داشته باشد. برای گذردهی مفید، مقدار $T_w=۲۰$ عملکرد بهتری دارد. مقدار میانگین و واریانس $T_w=۱۵$ برای تأخیر برقراری نشست‌ها بهتر عمل کرده و $T_w=۲۰$ ارسال مجددها را بهینه‌تر می‌نماید. از آنجا که تأخیر برقراری نشست‌ها برای $T_w=۱۵$ و $T_w=۲۰$ تفاوت چندانی با هم ندارند، مقدار $T_w=۲۰$ و $T_c=۴۰$ برای مقایسه کارایی OCL با OC انتخاب می‌شود.

همان طور که در شکل‌های ۸ تا ۱۰ نیز مشخص است، کارایی روش OCL تقریباً برابر با OC است و حتی از لحاظ میانگین، عملکرد بهتری دارد. در این شکل‌ها زمانی که تعداد نشست‌های ورودی کمتر از ظرفیت شبکه باشند، مقادیر ترسیم نمی‌شوند زیرا در این حالت مقادیر T_w و T_c نقشی در عملکرد عادی شبکه ندارند. در شکل ۱۱ گذردهی مفید و در شکل ۱۲ تأخیر برقراری نشست جهت بررسی پایداری و واکنش سریع OCL نمایش داده شده است. در شکل ۱۱ بعد از تغییر ناگهانی تعداد نشست‌های ورودی، گذردهی OCL تقریباً ثابت می‌ماند در حالی که برای OC مقداری کم شده و مجدداً اضافه می‌گردد، زیرا در OC برای همه سرورها $T_w=۲۰$ و $T_c=۴۰$ در نظر گرفته شده است. با یکسان در نظر

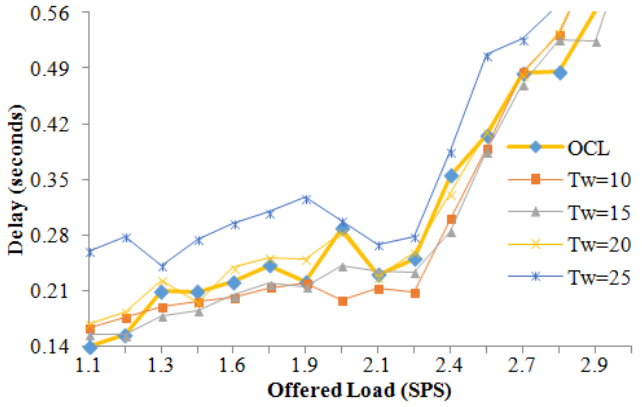
1. Overload Controller with Learning



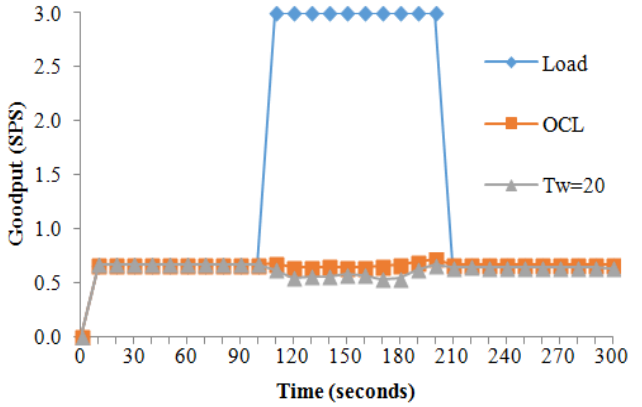
شکل ۸: گذردهی مفید OCL در مقایسه OC با T_w و T_c بهینه.



شکل ۱۰: نرخ ارسال مجدد OCL در مقایسه OC با T_w و T_c بهینه.



شکل ۹: تأخیر OCL در مقایسه OC با T_w و T_c بهینه.



شکل ۱۱: گذردهی مفید OCL تحت تغییر ناگهانی بار.

جدول ۳: مقایسه دو روش OCL و OC.

		OCL	$T_w=۱۰$	$T_w=۱۵$	$T_w=۲۰$	$T_w=۲۵$
گذردهی مفید	میانگین	۸۶۶٫۹۴	۸۴۸٫۶۹	۸۵۶٫۷۵	۸۶۶٫۱۹	۸۴۹٫۳۸
	واریانس	۱۴۱۱	۱۴۴۵	۱۵۸۹	۱۲۷۱	۱۵۳۹
تأخیر برقراری نشست	میانگین	۰٫۳۱۸۵	۰٫۳۱۲۴	۰٫۳۰۴۳	۰٫۳۳۳۳	۰٫۳۸۵۱
	واریانس	۰٫۰۲۳	۰٫۰۳۰۵	۰٫۰۲۴۷	۰٫۰۲۵۷	۰٫۰۲۲۹
تعداد ارسال مجددها	میانگین	۹۱۶٫۵۶	۹۴۲٫۴۴	۹۲۶٫۴۴	۹۱۸٫۳۸	۹۳۸٫۳۸
	واریانس	۲۶۸۵۴۱	۲۷۰۴۹۱	۲۷۵۱۱۰	۲۶۴۴۹۸	۲۸۵۳۸۸

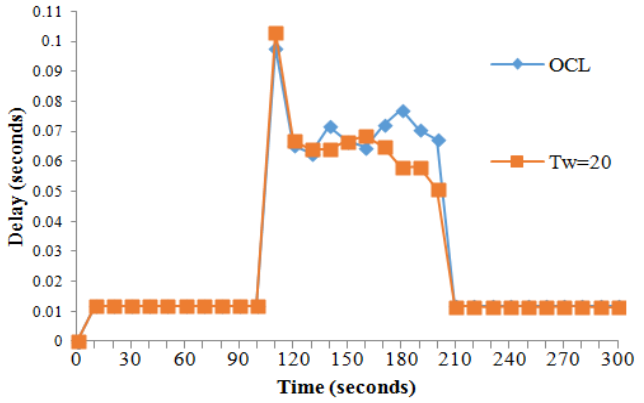
گرفتن این مقادیر برای تمام سرورها، ممکن است در کل، گذردهی تقریباً خوبی حاصل گردد ولی این احتمال وجود دارد که به دلیل بهینه‌نبودن این مقادیر برای تک‌تک سرورها، سرور برای لحظاتی با افزایش بار مواجه شود. از این رو بعد از تغییر ناگهانی، ابتدا گذردهی مقداری کاهش یافته تا اضافه بار رفع شود و سپس کم‌کم افزایش می‌یابد. در شکل ۱۲ بعد از افزایش ناگهانی بار، تأخیر در هر دو روش افزایش یافته تا بار اضافی وارد شبکه نشود و با ثابت‌شدن بار ورودی، تأخیر کاهش یافته و تقریباً برای هر دو مکانیزم ثابت می‌ماند. با این که تأخیر OC کمی بهتر می‌باشد، ولی همان طور که در شکل مشخص است با OCL تفاوتی چندانی ندارد. بعد از کاهش بار ورودی در ثانیه ۲۰۰، گذردهی و تأخیر هر دو روش به حالت قبل از تغییر ناگهانی برمی‌گردد. در نتیجه، OCL یک روش پایدار هنگام تغییرات بار ورودی است.

۴-۲ ارزیابی کارایی OCL

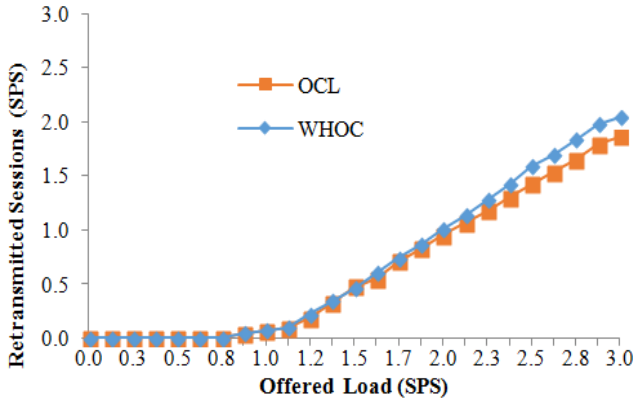
پس از بررسی مزیت OCL، کارایی آن با روش جدید کنترل اضافه بار هولونیک مبتنی بر پنجره (WHOC)[1] مقایسه می‌گردد [۲۷]. گذردهی مفید و تأخیر برقراری نشست‌ها در شکل‌های ۱۳ و ۱۴ نمایش داده شده است. همان طور که مشخص می‌باشد، وقتی که میزان بار ورودی کمتر از ظرفیت شبکه باشد، گذردهی مفید مکانیزم‌ها مشابه است زیرا هیچ اضافه باری در شبکه رخ نمی‌دهد. در WHOC وقتی تعدادی از نشست‌ها در صف سروری جمع می‌شوند، اندازه پنجره سرورهای لبه، مقداری کاهش یافته تا این تعداد نشست موجود در صف پردازش شوند. از این رو در ظرفیت شبکه، گذردهی مفید مقداری کمتر از روش OCL است. اما در OCL، هر زمان تشخیص داده شود که یکی از عامل‌ها از ناحیه امن خارج شده است، سریعاً واکنش نشان داده و با جلوگیری از ورود بار اضافی از مبدأ، تلاش می‌کند که در شبکه اضافه بار رخ ندهد که تأخیر به دست آمده در شکل ۱۴ نیز همین نتیجه را تأیید می‌کند.

شکل ۱۵ نرخ ارسال مجددها را نشان می‌دهد. همان گونه که قبلاً هم بیان شد، هرچه نرخ ارسال مجددها بیشتر باشد یعنی منابع بیشتری از شبکه صرف پردازش درخواست‌ها شده که در نهایت نتیجه‌ای به همراه

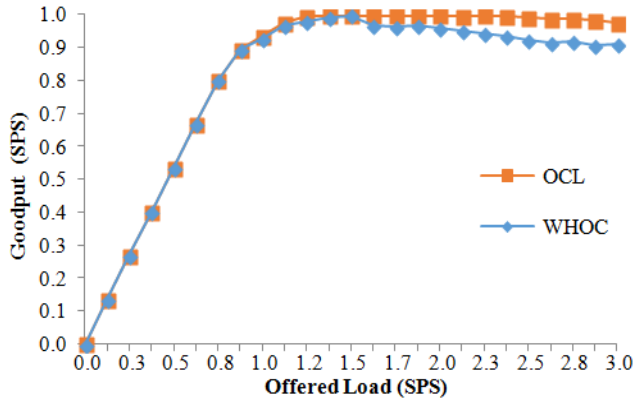
1. Window-Based Holonic Overload Control



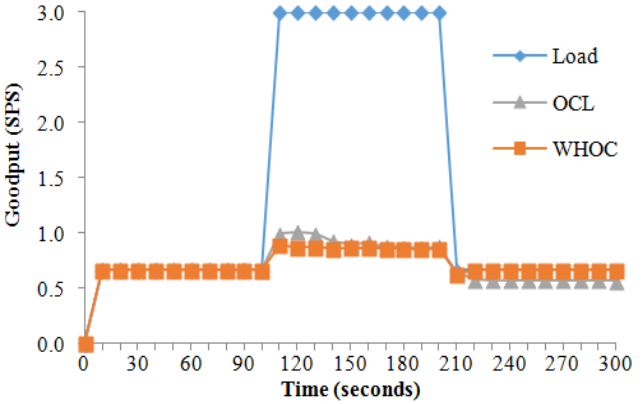
شکل ۱۲: تأخیر برقراری نشست‌ها در OCL تحت تغییر ناگهانی بار.



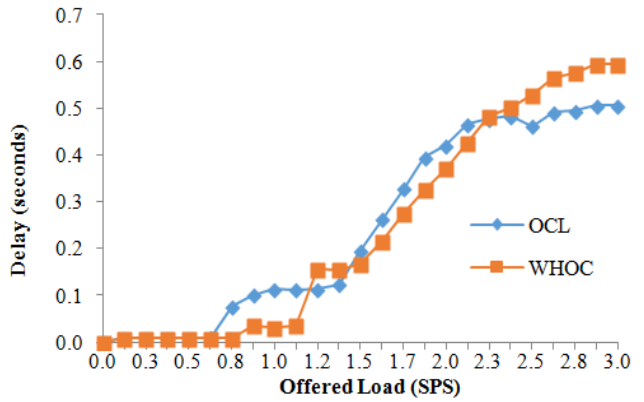
شکل ۱۵: نرخ ارسال مجددها برای مکانیزم‌های کنترل اضافه بار.



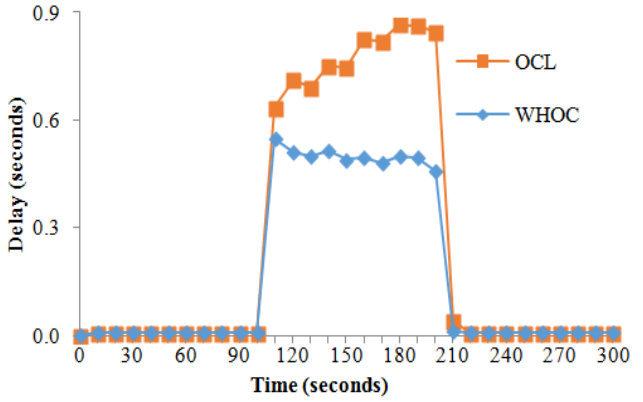
شکل ۱۳: گذردهی مفید برای مکانیزم‌های کنترل اضافه بار.



شکل ۱۶: گذردهی مفید روش‌های مورد بررسی تحت تغییر ناگهانی بار.



شکل ۱۴: تأخیر برقراری نشست‌ها برای مکانیزم‌های کنترل اضافه بار.

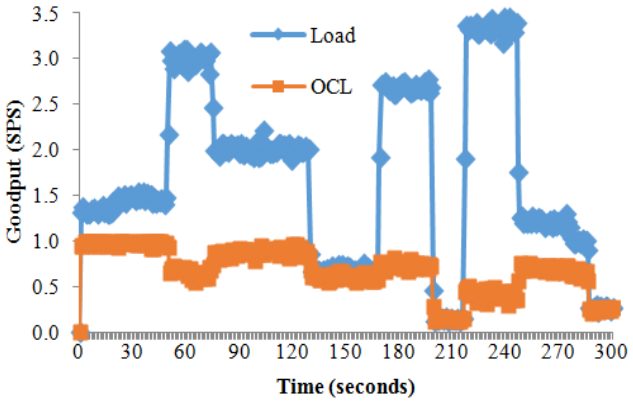


شکل ۱۷: تأخیر روش‌های مورد بررسی تحت تغییر ناگهانی بار.

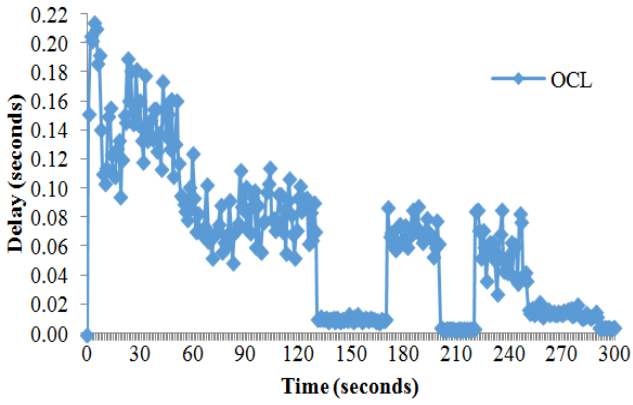
ندارند. WHOC نسبت به OCL حدود ۸٪ نشست‌ها را بیشتر رد می‌کند زیرا در این روش، نشست‌ها فقط به اندازه پنجره‌های محاسبه‌شده وارد شبکه گردیده و بقیه نشست‌ها قطعاً رد می‌شوند. اما در OCL هیچ فرضی در مورد استفاده از ظرفیت پردازنده نشده است، پس تا زمانی که تعداد نشست‌های ورودی کوچک‌تر یا مساوی ظرفیت شبکه باشند، بار ورودی طوری به شبکه وارد می‌شود که هیچ نشستی رد نمی‌گردد که این امر با پایش مداوم کنترل‌کننده و نگهداری حداکثری عامل‌ها در ناحیه امن محقق خواهد شد.

گذردهی مفید و تأخیر برقراری نشست‌ها جهت آزمایش پایداری روش OCL در شکل‌های ۱۶ و ۱۷ نشان داده شده است. همان طور که در شکل ۱۶ مشخص می‌باشد، WHOC نسبت به روش OCL نوسانات ابتدایی در لحظه تغییر بار ندارد، زیرا اندازه پنجره‌ها محدود بوده و هیچ گاه از ظرفیت پردازش پردازنده بیشتر نمی‌شود. اما در روش OCL تا زمان شروع فرایند مذاکره و رسیدن عامل‌ها به توافق، مقداری نوسان ابتدایی وجود داشته که با توافق صورت‌گرفته رفع خواهد شد. با این حال OCL واکنش سریع و پایداری نسبت به تغییر ناگهانی بار خواهد داشت. در لحظه ۲۰۰ ثانیه، پس از بازگشت مجدد تعداد نشست‌ها به حالت ابتدایی، OCL سریع به گذردهی قبلی برگشته و محدودیت‌های کنترلی سریعاً برداشته می‌شوند. تأخیر مشخص‌شده در شکل ۱۷ نیز تأکیدی بر پایداری OCL است.

جهت بررسی عملکرد روش OCL در یک محیط تقریباً واقعی، گذردهی مفید و تأخیر برقراری نشست‌ها برای بار ورودی متغیر توسط توزیع پواسن در شکل‌های ۱۸ و ۱۹ نمایش داده شده است. با توجه به شکل ۱۸ می‌توان دانست که OCL به خوبی تغییرات تعداد نشست‌های ورودی را دنبال می‌کند و خود را با آن تطبیق می‌دهد، بدون این که شبکه دچار از کار افتادگی شود. در شکل ۱۹ نیز تأخیر، کنترل و با تغییرات بار ورودی کم و زیاد گردیده و سیستم دچار ناپایداری نمی‌شود. این عملکرد تأکیدی بر پایداری و واکنش سریع روش OCL است.



شکل ۱۸: گذردهی مفید OCL تحت نشست‌های ورودی متغیر.



شکل ۱۹: تأخیر نشست‌های OCL تحت نشست‌های ورودی متغیر.

۵- نتیجه‌گیری و کارهای آینده

در آینده نزدیک، IMS به همراه پروتکل SIP به مهم‌ترین بستر جهت کاربردهای چندرسانه‌ای تبدیل خواهد شد. سرورهای IMS در مواجهه با اضافه بار، دچار افت گذردهی و از کار افتادگی می‌شوند و به علاوه اضافه بار در کل شبکه گسترش می‌یابد. از این رو مبحث کنترل اضافه بار در IMS، سیستمی پیچیده محسوب شده که سیستم‌های چندعامله جایگزین خوبی به جای روش‌های حل کلاسیک جهت حل این چالش می‌باشند. در سیستم‌های چندعامله می‌توان یک کار بزرگ را به مجموعه‌ای از کارهای کوچک‌تر تقسیم کرد تا هر عامل یک قسمت از کار را انجام دهد. در این پژوهش، سرورهای IMS به عنوان عامل‌هایی با قابلیت یادگیری و مذاکره در نظر گرفته شده‌اند که با استفاده از یادگیری Q، یک روش کنترل اضافه بار گام به گام مبتنی بر حذف، پیاده‌سازی شده که با پیچیدگی خطی به حذف مشکل اضافه بار کمک خواهند کرد.

در روش ارائه‌شده، فرایند یادگیری توسط هر عامل به صورت مستقل انجام می‌گیرد. گرچه این نوع یادگیری برای به دست آوردن پارامترهای مربوط به هر عامل مناسب است، ولی برای نشان‌دادن واکنش‌های بهینه توسط تمام عامل‌ها بهتر است که یادگیری در کل شبکه انجام گیرد. از طرفی در IMS موجوداتی مانند HSS و DNS وجود دارند که مبتنی بر SIP نیستند و می‌توانند در اضافه بار نقش داشته باشند که به عنوان کارهای آینده می‌توان این موجودیت‌ها را نیز وارد مسئله کرد. از طرفی، امروزه با حرکت پردازش به سوی محیط‌های ابری با توابع مجازی‌ساز شبکه، قیمت ساختار و بستر IMS کاهش یافته و می‌توان آن را به سرعت به صورت مقیاس‌پذیر توسعه داد. از این رو به عنوان کار آینده می‌توان روش ارائه‌شده را در محیط ابری با استفاده از NFV پیاده‌سازی کرد.

مراجع

[1] P. Agrawal, Y. Jui-Hung, C. Jyh-Cheng, and Z. Tao, "IP multimedia subsystems in 3GPP and 3GPP2: overview and scalability issues," *Communications Magazine, IEEE*, vol. 46, no. 1, pp. 138-145, Jan. 2008.

[2] C. Shen, H. Schulzrinne, and E. Nahum, "Session Initiation Protocol (SIP) Server Overload Control: Design and Evaluation," in Principles, Systems and Applications of IP Telecommunications. Services and Security for Next Generation Networks, S. Henning, S. Radu, and N. Saverio, Eds.: Springer-Verlag, pp. 149-173, 2008.

[3] V. Hilt and I. Widjaja, "Controlling overload in networks of SIP servers," in *Proc. IEEE Int. Conf. on Network Protocols*, pp. 83-93, Orlando, FL, USA, 19-22 Oct. 2008.

[4] J. Davin, P. Riley, and M. Veloso, "CommLang: Communication for Coachable Agents," Robot Soccer World Cup VIII, vol. 3276, D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, Eds. (Lecture Notes in Computer Science: Springer Berlin Heidelberg, pp. 46-59, 2005.

[5] M. Wooldridge, An Introduction to MultiAgent Systems, Wiley, 2009.

[۶] م. عبدوس، *ارائه یک مدل یادگیری تقویتی با نظارت چندسطحی در سیستم‌های چندعامله هولونی*، پایان‌نامه دکتری در مهندسی کامپیوتر (هوش مصنوعی و رباتیک)، مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، ۱۳۹۲.

[7] H. C. Hsieh and J. L. Chen, "Distributed multi-agent scheme support for service continuity in IMS-4G-Cloud networks," *Computers & Electrical Engineering*, vol. 42, pp. 49-59, Feb. 2015.

[8] M. Abdoos, N. Mozayani, and A. C. Bazzan, "Hierarchical control of traffic signals using Q-learning with tile coding," *Applied Intelligence*, vol. 40, no. 2, pp. 201-213, Mar. 2014.

[9] M. Abdoos, N. Mozayani, and A. L. C. Bazzan, "Holonic multi-agent system for traffic signals control," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5-6, pp. 1575-1587, May/Jun. 2013.

[10] Y. Yao, V. Hilaire, A. Koukam, and W. Cai, "A holonic model in wireless sensor networks," in *Proc. Int. Conf. on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 491-495, Harbin, China, 15-17 Aug. 2008.

[11] S. M. Hosseini and N. Mozayeni, "An intelligent method for resource management in wireless networks," in *Proc. 5th Conf. on Information and Knowledge Technology, IKT'13*, pp. 371-376, Shiraz, Iran, 28-30 May 2013.

[12] M. Cossentino, N. Gaud, V. Hilaire, S. Galland, and A. Koukam, "ASPECS: an agent-oriented software process for engineering complex systems," *Autonomous Agents and Multi-Agent Systems*, vol. 20, no. 2, pp. 260-304, 2010.

[13] M. Poikselkä, The IMS: IP Multimedia Concepts and Services, J. Wiley & Sons, 2006.

[14] V. K. Gurbani and R. Jain, "Transport protocol considerations for session initiation protocol networks," *Bell Labs Technical J.*, vol. 9, no. 1, pp. 83-97, 2004.

[15] M. Ohta, "Overload control in a SIP signaling network," *International J. of Electrical and Electronics Engineering*, vol. 3, no. 2, pp. 87-92, 2009.

[16] E. N. V. Hilt, C. Shen, and A. Abdelal, "Design considerations for session initiation protocol (SIP) overload control," Internet Engineering Task Force (IETF), Request for Comments (RFC) 6357, 2011.

[17] J. Liao, J. Wang, T. Li, J. Wang, J. Wang, and X. Zhu, "A distributed end-to-end overload control mechanism for networks of SIP servers," *Comput. Netw.*, vol. 56, no. 12, pp. 2847-2868, 12 Aug. 2012.

[18] A. Akbar, S. M. Basha, and S. A. Sattar, "A cooperative overload control method for SIP servers," in *Proc. Int. Conf. on Communications and Signal Processing, ICCSP'15*, pp. 1296-1300, Melmaruvathur, India, 2-4 Apr. 2015.

[19] H. Dong-Yeop, P. Ji Hong, Y. Seung-Wha, and K. Ki-Hyung, "A window-based overload control considering the number of confirmation massages for SIP server," in *Proc.4th Int. Conf. on Ubiquitous and Future Networks, ICUFN'12*, pp. 180-185, Phuket, Thailand, 4-6 Jul. 2012.

[20] S. Jing, T. Ruixiong, H. Jinfeng, and Y. Bo, "Rate-based SIP flow management for SLA satisfaction," in *Proc. IFIP/IEEE Int. Symp. on Integrated Network Management*, pp. 125-128, Long Island, NY, USA, 1-5 Jun. 2009.

[21] R. G. Garroppo, S. Giordano, S. Niccolini, and S. Spagna, "A prediction-based overload control algorithm for SIP servers," *IEEE*

Trans. on Network and Service Management, vol. 8, no. 1, pp. 39-51, Mar. 2011.

[22] S. V. Azhari, M. Homayouni, H. Nemati, J. Enayatizadeh, and A. Akbari, "Overload control in SIP networks using no explicit feedback: a window based approach," *Computer Communications*, vol. 35, no. 12, pp. 1472-1483, 1 Jul. 2012.

[23] A. Abdelal and W. Matragi, "Signal-based overload control for SIP servers," in *Proc. 7th IEEE Consumer Communications and Networking Conf.*, 7 pp., Las Vegas, NV, USA, 9-12 Jan. 2010.

[24] Y. Hong, C. Huang, and J. Yan, "Applying control theoretic approach to mitigate SIP overload," *Telecommunication Systems*, vol. 54, no. 4, pp. 387-404, Dec. 2013.

[25] A. Montazerolghaem, M. H. Yaghmaee, A. Leon-Garcia, M. Naghibzadeh, and F. Tashtarian, "A load-balanced call admission controller for IMS cloud computing," *IEEE Trans. on Network and Service Management*, vol. 13, no. 4, pp. 806-822, Dec. 2016.

[26] A. Montazerolghaem, M. H. Y. Moghaddam, and A. Leon-Garcia, "OpenSIP: toward software-defined SIP networking," *IEEE Trans. on Network and Service Management*, vol. 15, no. 1, pp. 184-199, 2018.

[27] M. Khazaei and N. Mozayani, "A dynamic distributed overload control mechanism in SIP networks with holonic multi-agent systems," *Telecommunication Systems*, vol. 63, no. 3, pp. 437-455, 2016.

[28] M. Khazaei and N. Mozayani, "Overload management with regard to fairness in session initiation protocol networks by holonic multiagent systems," *International J. of Network Management*, vol. 27, no. 3, Article ID: e1969, May/Jun. 2017.

[۲۹] ا. اسماعیلی، *ارائه روشی به منظور کنترل ترافیک در تقاطع شهری با استفاده از سیستم‌های چندعاملی هولونی*، پایان‌نامه کارشناسی ارشد مهندسی کامپیوتر (هوش مصنوعی و رباتیک)، مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، ۱۳۸۹.

[30] J. Liao, J. Wang, T. Li, J. Wang, J. Wang, and X. Zhu, "A distributed end-to-end overload control mechanism for networks of SIP servers," *Computer Networks*, vol. 56, no. 12, pp. 2847-2868, Aug. 2012.

[31] J. Wang, J. Liao, T. Li, J. Wang, J. Wang, and Q. Qi, "Probe-based end-to-end overload control for networks of SIP servers," *J. of Network and Computer Applications*, vol. 41, pp. 114-125, May 2014.

مهدی خزائی تحصیلات خود در مقطع کارشناسی مهندسی کامپیوتر گرایش سخت افزار را در سال ۱۳۸۳ و کارشناسی ارشد و دکتری مهندسی کامپیوتر گرایش معماری سیستم-های کامپیوتری را به‌ترتیب در سال‌های ۱۳۸۶ و ۱۳۹۶ در دانشگاه علم و صنعت ایران به پایان رسانده است و هم‌اکنون استادیار دانشکده فناوری اطلاعات دانشگاه صنعتی کرمانشاه می‌باشد. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: شبکه‌های حسگر بی‌سیم، اینترنت اشیاء، یادگیری ماشین و شبکه‌های چند رسانه‌ای.