مجله محاسبات نرم
SOFT COMPUTING JOURNAL
تارنمای مجله: scj.kashanu.ac.ir
دانشگاه کاشان
University of Kashan

# تحلیل و مدل‌سازی سرورهای VoIP: رویکرد برنامه‌ریزی خطی✧

احمدرضا منتظرالقائم*، استادیار
گروه کامپیوتر، دانشکده مهندسی، دانشگاه صنعتی قوچان، خراسان رضوی، ایران.

**اطلاعات مقاله**

**تاریخچه مقاله:**
دریافت ۰۸ مرداد ماه ۱۳۹۸
پذیرش ۲۵ فروردین ماه ۱۳۹۹

**کلمات کلیدی:**
کنترل اضافه‌بار
شبکه‌های نرم‌افزار محور
برنامه‌ریزی خطی
پراکسی سرور Asterisk
توزیع حالت
مسئلهٔ بهینه‌سازی

**چکیده**

**پروتکلِ SIP در لایهٔ کاربرد برای آغاز، مدیریت و خاتمه جلساتِ چندرسانه‌ای توسط IETF استاندارد شده و به‌عنوان پروتکلِ اصلیِ سیگنالینگ هم در اینترنت و هم در شبکهٔ VoIP استفاده فراوانی یافته است. یکی از چالش‌های مطرح در این پروتکل، مسئلهٔ اضافه‌بار و عدم توزیع حالت مناسب است. وجود این چالش سبب می‌شود که طیف وسیع کاربران شبکهٔ نسل آینده با افت شدید کیفیت سرویس مواجه شوند. در این مقاله، مسئلهٔ توزیع حالت بین چندین گره را تعریف می‌کنیم چراکه حفظ حالت با مصرف قابل توجهی از منابع همراه است و منجر به اضافه‌بار می‌شود. در این مسئله، هدف افزایش گذردهی کلی تماس‌ها و دسترس‌پذیری سرورهاست. نخست چهارچوبی بر مبنای تکنولوژی شبکه‌های نرم‌افزارمحور ارائه و سپس این مسئله را به‌صورت یک مسئلهٔ بهینه‌سازی فرمول‌بندی و آن را به‌صورت یک ماژول بر روی کنترلر پیشنهادی، پیاده‌سازی و ارزیابی می‌کنیم. نتیجه، یک شبکهٔ مقیاس‌پذیرتر SIP است که به‌صورت پویا تعداد تقاضاهای SIP را که سرور برای آن‌ها حالتمند است، تعیین و در عین حال برای مابقی تقاضاها، نگهداری حالت را به سروری که پایین‌دست‌تر از خودش قرار دارد محول می‌کند. این طرح برخلاف سرورهای SIP موجود است؛ زیرا آن‌ها به‌صورت ایستا طوری پیکربندی می‌شوند که یا بدون حالت باشند یا حالتمند و لذا گذردهی غیربهینه‌ای را برای تماس نتیجه می‌دهند. ارزیابی عملکرد در دو سطح زیرساخت و کنترل انجام و نتایج ارائه می‌شود.**



## ۱. مقدمه

در این مقاله به توسعهٔ روش توزیع حالت و توزیع بارِ شبکهٔ SIP معرفی‌شده در مقالهٔ [۳۳] با بهره‌گیری از تکنولوژی شبکه‌های نرم‌افزارمحور می‌پردازیم. در این راستا ابتدا مقدمه‌ای در رابطه با شبکهٔ SIP خواهیم داشت. پروتکل شروع جلسه (SIP)[1] [۱] یک پروتکل لایهٔ کاربرد است که برای دستهٔ متنوعی از برنامه‌های کاربردی شامل VoIP[2]، IM[3] و Presence [۲] و امروزه 3GPP[4] [۳] برای برقراری و خاتمهٔ تماس استفاده می‌شود. علاوه بر آن، پیشنهادهایی مبنی بر استفاده از SIP به‌عنوان یک سازوکار سیگنالینگ برای هر نوع جلسه داده یا رسانه داده شده است [۴]. در SIP تقاضاهای ارتباط از طریق به کارگیری انبوهی از

✧ نوع مقاله: پژوهشی
* نویسنده مسئول
پست الکترونیک: ar.montazer@qiet.ac.ir (منتظرالقائم)

1. Session Initiation Protocol
2. Voice over IP
3. Instant Messaging
4. The 3rd Generation Partnership Project

**نحوه ارجاع به مقاله: منتظرالقائم، احمدرضا، «تحلیل و مدل‌سازی سرورهای VoIP: رویکرد برنامه‌ریزی خطی»، مجله محاسبات نرم، جلد ۷، شماره ۲، ص۲ـ۲۳، پاییز و زمستان ۱۳۹۷.**

سرورهای پراکسی ردوبدل می‌شوند. هریک از این سرورها مقداری از عملیات برقراری تماس را انجام می‌دهند. این کارکردها عبارت‌اند از کشف میزبان، مسیردهی، نگه داشتن حالت و تأیید مجوز. به‌تدریج که برنامه‌های کاربردی بیشتری از SIP استفاده می‌کنند، کارکردی که سرورهای SIP فراهم کرده‌اند، توسعه پیدا کرده است.

رویکرد سنتی در پشتیبانی از این کارکرد این است که تخصیص هر عملکرد به یک سرور جداگانه در انبوه برنامه‌های کاربردی باشد. در صورت تجاوز عملکردهایی که می‌بایست فراهم شوند، سرورهای مرکزی درخواست‌های ضروری را فراهم می‌سازند و مابقی سرورها فقط تقاضا را مسیردهی می‌کنند. اگر تعداد عملکردها از تعداد سرورها بیشتر باشد، برخی از سرورها چندین عملکرد را به عهده می‌گیرند. در هریک از این حالات، این تخصیص به‌صورت آماری تعیین می‌شود. این مقاله نشان می‌دهد که این امر یک گذردهی غیربهینه را نتیجه خواهد داد. ما به‌جای این روش، سازوکاری را برای توزیع پویای کارکردها بین سرورها پیشنهاد می‌کنیم. در این صورت، هر سرور فقط برای کسری از تقاضاهایی که از شبکهٔ سرورها عبور می‌کنند، یک کارکرد بخصوص را انجام می‌دهد. علاوه بر این، از تکنولوژی شبکه‌های نرم‌افزارمحور برای ارائهٔ یک چهارچوب یکپارچه و متمرکز برای مدیریت توزیع حالت در کل شبکه استفاده می‌کنیم. این تکنولوژی یک معماری جدید در شبکه است که دید یکپارچه نسبت به کل شبکه در اختیار می‌گذارد.

آزمایش‌های ما روی Asterisk [۵] که یک سرور SIP متن باز است، انجام شد. در این راستا به‌صورت تجربی منابع پردازشیِ صرف‌شده برای برقراری یک تماس VoIP را اندازه‌گیری کردیم. اندازه‌گیری‌های ما حاکی از این هستند که بسته به کارکرد فراهم‌شده، منابع صرف‌شده به‌طور قابل ملاحظه‌ای تغییر می‌کند؛ به‌خصوص ایجاد، ابقا و حذف حالتِ مربوط به تماس، یکی از قابل توجه‌ترین مصرف‌کنندگان منابع CPU[1] است. به سروری که حالت را نگه می‌دارد، سرور حالتمند[2] و به سروری که حالت را نگه نمی‌دارد، سرور بدون حالت[3] می‌گویند. این حالت توسط سرور به‌منظور نگه داشتن اطلاعات مجموعه‌ای از پیام‌ها مورد استفاده قرار می‌گیرد. حالت نگهداری‌شده به‌منظور تأمین دستهٔ متنوعی از کارکردها، شامل کاهش ارسال‌های مجدد غیرضروری و تأمین سرویس حسابداری استفاده می‌شود. ما همچنین میزان استفاده از CPU در سرور Asterisk را برای نگه داشتن حالت، در زمانی که بار تماس افزایش می‌یابد، اندازه گرفتیم. مجدداً اندازه‌گیری‌های ما نشان می‌دهند حداکثر باری که Asterisk می‌تواند به‌صورت حالتمند پشتیبانی کند، به مقدار قابل توجهی از آنچه در وضعیت بدون حالت می‌تواند پشتیبانی کند، کمتر است. پس ما توزیع حالت را به‌صورت یک مسئلهٔ بهینه‌سازی مدل کرده و نشان می‌دهیم که این موضوع دو نتیجه دارد:

اول اینکه پیکربندیِ ایستایِ مجموعه‌ای از سرورها، برای اینکه نسبت به تمامی تماس‌ها حالت دار یا فاقد حالت باشند (چیزی که امروزه انجام می‌شود)، به گذردهی غیربهینه منجر می‌شود؛ دوم اینکه می‌توان با توزیع حالت بین سرورها، گذردهی سیستم را به مقدار قابل توجهی بهبود بخشید. در این صورت هر سرور تنها برای بخشی از درخواست‌ها حالت را نگهداری می‌کند و برای بقیه، فاقد حالت باقی می‌ماند. اگر حالت هر تماس حداقل در یکی از سرورهای سیستم نگهداری شود، این سیستم به‌طور کامل برای مجموعه تقاضاهایی که از آن عبور می‌کنند، حالت‌دار خواهد بود. همین موضوع انگیزهٔ این مقاله برای پیاده‌سازی سرور SIPای است که توسط یک الگوریتم توزیع حالت سعی دارد برای هر پیکربندی از سرورها، به گذردهی بهینه دست یابد. این سرور به‌صورت پویا کسری از تقاضاها که برای آن‌ها حالت را نگه داشته است، مجدداً پیکربندی می‌کند؛ به‌طوری که سیستم در مجموع گذردهی تماس بیشتری را نتیجه خواهد داشت. در این مقاله نشان می‌دهیم که توزیع حالت به‌صورت پویا نسبت به توزیع حالت به‌صورت ایستا که در آن سرور به‌صورت استاتیک در وضعیت

1. Central Processing Unit

2. Stateful
3. Stateless

حالتمند یا بدون حالت قرار می‌گیرد، گذردهی به مقدار ۱۵٪ افزایش می‌یابد.

مشارکت‌های اصلی در این مقاله در زیر لیست شده‌اند:

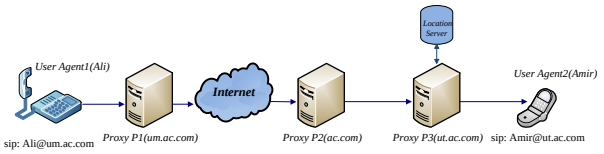
- بررسی «مصرف منابع CPU» در یک سرور SIP با جزئیات کامل؛
- معرفی «حفظ حالت» به‌عنوان یک منبع اصلی مصرف CPU؛
- طراحی یک چهارچوب پیشنهادی مبتنی بر تکنولوژی شبکه‌های نرم‌افزارمحور؛
- معرفی یک «فرمول بهینه‌سازی» برای «مسئلهٔ توزیع حالت»؛
- «طراحی، پیاده‌سازی و ارزیابی»؛ یک سرور SIP که به‌منظور بهبود گذردهی کلی در سیستم، حالت تماس را توزیع می‌کند.

هرچند این مقاله به‌طور خاص به مسئلهٔ توزیع حالت توجه دارد، برای حصول گذردهی نزدیک به بهینه، باید روش‌های جدیدی برای توزیع کارکردِ تقاضاهای تماس در انبوهی از پراکسی‌های SIP نیز مورد توجه قرار گیرند.

ادامهٔ این مقاله بدین صورت سازمان‌دهی شده است: در بخش ۲ به مرور پروتکل SIP و کارهای مرتبط می‌پردازیم. در بخش ۳ شبکه‌های نرم‌افزارمحور و تکنولوژی مدیریت یکپارچه شبکه را معرفی می‌کنیم. بخش ۴ ادبیات و پیشینهٔ تحقیق را بررسی می‌کند. در بخش ۵ ارزیابی مفصلی از سرور SIP ارائه می‌شود. بخش ۶ مسئلهٔ توزیع حالت را بررسی می‌کند. در بخش ۷ چهارچوب پیشنهادی مبتنی بر شبکه‌های نرم‌افزارمحور و فرمول‌بندی توزیع پویای حالت را ارائه می‌دهیم. در بخش ۸ به ارزیابی عملکرد آن می‌پردازیم. نتیجه‌گیری مقاله در بخش ۹ انجام شده است.

## ۲. مرور و پیش‌زمینهٔ پروتکل SIP

به‌منظور برقراری تماس‌های VoIP به‌وسیلهٔ پروتکل SIP، که شبیه شماره‌های تلفن است، از یک «شناسهٔ منبع یکنواخت[1]» SIP (SIP URI) برای برقراری تماس با یک کاربر استفاده می‌شود. برای مثال، SIP URI کاربر ALI به‌صورت sip: ALI@um.ac.com است. می‌توان چندین دستگاه تلفن را با یک URI ثبت کرد. پروتکل SIP به‌منظور مکان‌یابی صحیح دستگاه و همچنین ارسال تقاضای تماس، از انبوهی از برنامه‌های کاربردی شامل پراکسی سرورها بهره می‌برد. یک پیمایش معمولی برای برقراری تماس VoIP در شکل (۱) نشان داده شده است.




**شکل (۱): برقراری تماس در شبکهٔ VoIP به‌صورت سلسله‌مراتبی**

وقتی کاربر Ali (sip: Ali@um.ac.com) کاربر Amir با شناسه sip: Amir@ut.ac.com را می‌خواند، پیام تقاضای تماس آن به سروری که مسئول دامنهٔ *um.ac.com* است، یعنی پراکسی $P_1$ فرستاده می‌شود. سرور $P_1$ دربارهٔ چگونگی مسیردهی تماس به سرور مسئول برای دامنه Amir، یعنی سرور $P_2$ تصمیم می‌گیرد. سپس این تقاضا از طریق اینترنت به سرور $P_2$ ارسال می‌شود. سرور $P_2$ مسئول بالاترین سطح دامنه *ac.com* است و با تعیین دامنهٔ سطح پایین‌تر (محل Amir) تصمیم می‌گیرد که این تقاضای تماس را به کجا مسیردهی کند.

در این مثال، *sip: Amir@ut.ac.com* در دامنهٔ *ut.ac.com* واقع است؛ لذا تقاضای تماس به سرور $P_3$ که مسئول دامنهٔ این سطح فرعی است، مسیردهی می‌شود. سپس $P_3$ به‌منظور تعیین آدرس IP[2] کنونی تلفن مربوط به Amir با یک پایگاه داده موقعیت تماس می‌گیرد. پس از آن $P_3$ تقاضای برقراری تماس را به تلفن Amir (عامل کاربر $U_2$) مسیردهی می‌کند و این کاربر می‌تواند تماس را قبول یا رد کند. به‌محض اینکه موقعیت (آدرس IP) نقاط انتهایی به طریق گفته‌شده در بالا تعیین شود و تماس نیز مورد پذیرش قرار گیرد، رسانه به‌صورت مستقیم بین آن‌ها مسیردهی می‌شود و نیازی به گذر از سرورها ندارد.

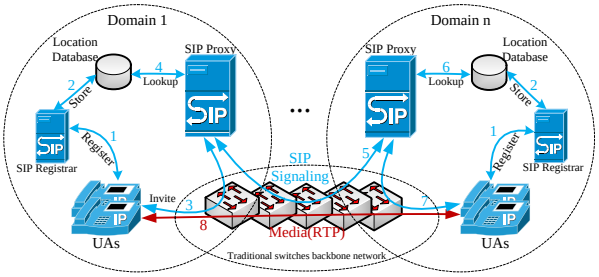
سرورهای SIP معمولاً به‌صورت سلسله‌مراتبی طی می‌شوند

1. Uniform Resource Identifier

2. Internet Protocol

[۶]؛ بدین معنی که در داخل یک دامنه، یک درخواست از طریق چندین سرور گذر می‌کند. همین موضوع مبنای یکی از فرضیات ماست: «چندین سرور در یک دامنه وجود دارد که می‌توان حالت را بین آن‌ها توزیع کرد.»

پیام‌هایی که برای برقراری تماس یا خاتمهٔ آن مبادله می‌شوند، در شکل (۲) نشان داده شده است. این شکل، نسخهٔ ساده‌شده‌ای از شکل (۱) است. همان طور که در شکل (۲) نشان داده شده، کل تماس از مجموعه‌ای از تراکنش‌ها تشکیل شده است. تراکنش تمامی پیام‌ها، از تقاضای شروع گرفته تا پاسخ نهایی آن را در بر می‌گیرد. دو عدد از این تراکنش‌ها در شکل (۲) نشان داده شده است؛ یکی تراکنش برقراری تماس و دیگری تراکنش خاتمهٔ تماس. تراکنش برقراری تماس با پیام INVITE آغاز می‌گردد و شامل تمامی پیام‌های مبادله‌شده تا پاسخ نهایی 200OK می‌شود. پیام‌های 1xx پاسخ‌هایی موقتی و میانی هستند و به‌منظور نشان دادن روند پیشرفت استفاده می‌شوند. یک مکالمه یا دیالوگ، تمامی تراکنش‌هایی را که بخشی از تماس (یا مکالمه) کلی هستند، در بر می‌گیرد. در شکل (۲)، این مکالمه دربردارندهٔ هم تراکنش برقراری تماس و هم تراکنش خاتمهٔ تماس هست.




**شکل (۲): تبادل پیام‌ها برای برقراری تماس در شبکهٔ VoIP**

سروری که در طول مدت‌زمان یک تراکنش، حالت را نگه می‌دارد، با عنوان سرور حالتمندِ تراکنش شناخته می‌شود. سرورهای $P_1$ و $P_3$ هر دو سرورهای حالتمندِ تراکنش هستند. در سوی مقابل، اگر یک سرور برای مدت‌زمان کل مکالمه حالت را نگه دارد، به آن حالتمندِ مکالمه گفته می‌شود. تنها سرور $P_1$ یک سرور حالتمند مکالمه است. این سرور دلیل گذر پیام BYE و پیام نهایی 200OK از خودش را تشریح می‌کند. با توجه به مزیت نگهداری حالت، سرورهای حالتمند تراکنش ارسال‌های مجدد را جذب می‌کنند، به تقاضاهای چندشاخه[1] می‌پردازند و تقاضاها و ثبت‌نام‌ها[2] را مسیردهی مجدد می‌کنند. سرورهای حالتمند وقتی استفاده می‌شوند که نیاز است، حالتِ تراکنش INVITE را به تراکنش‌های بعدی از قبیل REINVITE یا BYE متصل کند؛ این برای سرورهایی که اطلاعات حساب[3] را نگه می‌دارند، یا سرورهای کنفرانس که باید پارامترهای کنفرانس را در خود نگه دارند و برای کاربران جدید و هنگامی‌که به کنفرانس می‌پیوندند، مفید است. ما در این مقاله، «حالت» را به معنی «حالت تراکنش» به کار می‌بریم، مگر آنکه مورد دیگری برای آن تصریح شود. سرورهای غیرحالتمند هیچ حالتی را نگه نمی‌دارند. مزیت اصلی آن‌ها در توانایی آن‌ها در پردازش بسیار سریع تقاضاهاست. پرکاربردترین سرورهای پراکسی از جمله OpenSER و Asterisk می‌توانند هم حالتمند و هم غیرحالتمند باشند و می‌توان آن‌ها را به‌صورت ایستا طوری پیکربندی کرد که مطابق یکی از این دو وضعیت رفتار کنند.

این حالت با حالتی که توسط پروتکل‌های پایین‌دست از قبیل TCP یا IP نگه داشته می‌شود، متفاوت است؛ زیرا این حالت صرفاً یک حالت مبتنی بر برنامهٔ کاربردی است.

## ۳. شبکه‌های نرم‌افزارمحور و تکنولوژی مدیریت یکپارچهٔ شبکه

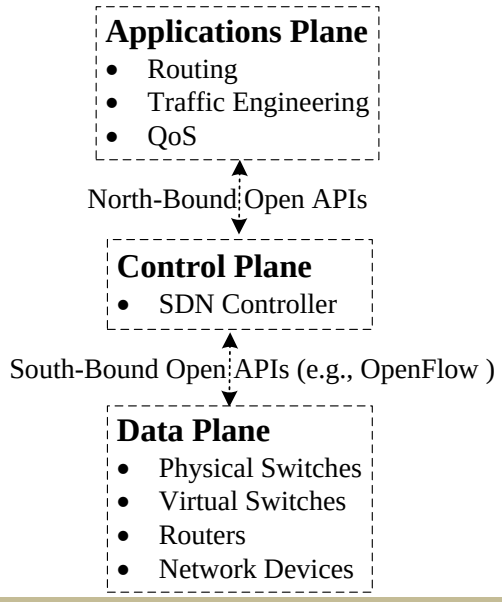
در این بخش به معرفی فناوری جدید به نام شبکه‌های نرم‌افزارمحور یا SDN[4] خواهیم پرداخت که در مدیریت منابع شبکه‌های گوناگون تأثیر شگرفی گذاشته‌اند.

پروتکل‌ها و کنترل‌های توزیع‌شده که داخل دستگاه‌های شبکه (از جمله سوئیچ‌ها، روترها و...) در حال اجرا هستند، فناوری‌های کلیدی هستند. با وجود استفادهٔ گسترده از آن‌ها، مدیریت شبکه‌های سنتی (از جمله شبکه‌های SIP) بسیار دشوار است؛ زیرا این شبکه‌ها به‌صورت عمودی پیاده‌سازی

1. Forking requests
2. Registrations
3. Accounting
4. Software-Defined Networking

شده‌اند؛ یعنی سطح کنترل که وظیفهٔ تصمیم‌گیری در خصوص کنترل ترافیک (مثلاً مسیریابی پیام‌ها) را بر عهده دارد و سطح داده که ترافیک را با توجه به تصمیمات اتخاذشده توسط سطح کنترل ارسال می‌کند، داخل هریک از دستگاه‌های شبکه تعبیه شده است. این امر باعث کاهش انعطاف‌پذیری و نوآوری می‌شود. همچنین اپراتورها مجبور به پیکربندی هریک از دستگاه‌ها با استفاده از فرمان‌های سطح پایین به‌صورت جداگانه هستند. شبکه‌های نرم‌افزارمحور یک مفهوم شبکه‌ای نوظهور هستند که با حذف ساختار عمودی، جداسازی سطح منطقی شبکه از دستگاه‌های شبکه و ترویج کنترل مرکزی شبکه، قصد رفع محدودیت‌های شبکه‌های فعلی را دارند. در SDN، عملکردهای شبکه را می‌توان به‌صورت نرم‌افزاری و فارغ از سخت‌افزار و مارک[1] خاصی توسعه و گسترش داد. با استفاده از SDN می‌توان یک دید کلی[2] از وضعیت شبکه، انعطاف‌پذیری بالا، مدیریت آسان و یکپارچه را در اختیار داشت که بهره‌وری شبکه را به‌طور چشمگیری افزایش و هزینه‌های لازم برای تهیهٔ تجهیزات گران‌قیمت و همچنین نیروی انسانی موردنیاز برای مدیریت شبکه را به‌شدت کاهش می‌دهد [۳۲]. شکل (۳) نمایی از معماری SDN را نمایش می‌دهد که از سه قسمت اصلی به نام صفحهٔ کنترل[3]، صفحهٔ داده[4] و صفحهٔ برنامه‌های کاربردی[5] تشکیل شده است.

صفحهٔ داده شامل دستگاه‌های شبکه از جمله دستگاه‌های مسیریابی، سوئیچ و... می‌باشد، به‌طوری که فاقد بخش کنترلی و یا نرم‌افزاری جهت تصمیم‌گیری‌های خودکار هستند. بخش هوشمند شبکه در کنترل‌کننده‌های نرم‌افزاری SDN قرار دارد که ساختار کلی شبکه را حفظ می‌کند (صفحهٔ کنترل). صفحهٔ برنامه‌های کاربردی نیز شامل مجموعه‌ای از برنامه‌های کاربردی مانند مسیریابی، فایروال، تعادل بار، نظارت و... است. پروتکلِ ارتباطی بین صفحات یکسری Open APIs استاندارد از جمله پروتکل OpenFlow می‌باشد.




**شکل (۳): معماری و مدل مرجع SDN**

## ۴. ادبیات و پیشینهٔ تحقیق

تحقیقات زیادی دربارهٔ عوامل کارایی پروکسی سرور SIP انجام شده است. در مقالهٔ [۷] به روش‌های کنترل اضافه‌بار در سرور پروکسی SIP پرداخته شده و برای اندازه‌گیری گذردهی از نرم‌افزار OPNET استفاده شده است. در مقالات [۸ و ۹]، SIP به‌صورت عملی همراه با پروتکل انتقال UDP و TCP پیاده‌سازی شده است که برای به‌دست‌آوردن نتایج بازدهی از OpenSER استفاده کرده است. SIPstone [۲] مجموعه محکی است که در آن معیارهای متنوعی برای ارزیابی توان سرورهای پروکسی، راهنما و ثبت‌کننده در پاسخ‌گویی به درخواست‌های SIP، پیشنهاد شده است. در [۱۰] یک محک دیگر برای اندازه‌گیری اثر سیستم‌عامل، پیکربندی سخت‌افزاری، پایگاه داده و لایهٔ انتقال انتخابی در کارایی SIP ارائه شده است. در [۱۱] روی چهار نوع پیاده‌سازی پروکسی که از لحاظ تخصیص حافظه متفاوت‌اند، آزمایش‌های عملی انجام گرفته است. نتایج این آزمایش‌ها نشان می‌دهد که پارامترهای مؤثر در کارایی پروکسی دو دسته‌اند: ۱. پارامترهای مرتبط با پروتکل (از قبیل طول پیام، متغیر بودن طول و نامرتب بودن سربارها)؛ ۲. پارامترهای مرتبط با نوع پیاده‌سازی سرور (مانند نحوهٔ تخصیص منابع سیستم‌عامل به تراکنش‌ها). همچنین در [۱۲] نیز تحقیقات

1. Brand
2. Global view
3. Control Plane
4. Data Plane
5. Applications Plane

مشابهی در مورد اثر سیستم‌عامل و نوع پیاده‌سازی پروکسی روی کارایی SIP انجام‌شده است. در [۱۳] با معرفی ابزاری تحت عنوان SIPperformer، تأثیر تأخیر پاسخ کاربر روی کارایی سرور SIP مورد تحلیل قرار گرفته است. در [۶] با قرار دادن تنها یک پروکسی بین عامل‌های کاربر، مسائلی چون قابل توجه بودن هزینهٔ امنیت در پیکربندی با تصدیق هویت، اثر حالتمند بودن یا نبودن پروکسی و نوع پروتکل لایهٔ انتقال روی کارایی پروکسی مورد مطالعه قرار گرفته است. دسته‌ای از پژوهش‌ها بر ارزیابی کارایی SIP تحت تکنولوژی‌های دسترسی مختلف متمرکز شده‌اند. در [۱۴] تأخیر سیگنالینگ SIP در برقراری جلسات IMS[1] برای کانال‌های مختلف WiMax با سرعت‌های متفاوت ارزیابی‌شده است. در [۱۶] تأثیر به‌کارگیری پروتکل‌های لایهٔ انتقال مختلف، بخصوص تأثیر مکانیزم کنترل پنجره در TCP، روی گذردهی و تأخیر برقراری تماس مورد ارزیابی قرار گرفته است. در [۱۷] نشان داده شده است که برخلاف تصور عام که علت استفادهٔ متداول از UDP در برابر TCP را سربار پردازشی کم آن قلمداد می‌کنند، ممکن است نامطلوب بودن کارایی در استفاده از TCP ناشی از نحوهٔ پیاده‌سازی پروکسی باشد. در [۲۳] یک متعادل‌کنندهٔ بار که به یک جدول هویت شامل بار سرورهای SIP مجهز است، سعی دارد تا اضافه‌بار کل را بین گروهی از سرورها تقسیم کند. در [۲۴] طراحی، پیاده‌سازی و ارزیابی عملکرد یک متعادل‌کنندهٔ بار برای سرورهای SIP مبتنی بر خوشه‌بندی ارائه شده است. این طرح روی سرورهای OpenSIPS مورد تجزیه و تحلیل قرار گرفته و سعی نموده پارامترهای کیفیت سرویس را اندازه‌گیری کند. در [۲۵] با استفاده از یک مدل برنامه‌ریزی خطی، مسئلهٔ کنترل پذیرش تعداد تماس‌ها با توجه به ظرفیت سرورهای SIP مورد بررسی قرار گرفته است. در [۲۶] تنظیمات مختلفِ موجود در سرورها جهت متعادل‌سازی بار SIP توضیح داده شده است. همچنین برای در دسترس بودن و مقیاس‌پذیری نیز نکاتی ارائه شده است. در [۲۷] یک تحلیل مقایسه‌ای از الگوریتم‌های توازن بار برای سرورهای SIP با توجه به زمان پاسخ‌گویی و عملکرد توان سرور ارائه شده است. در [۲۸] یک روش کنترل بار اضافی مبتنی بر نرخ برای شناسایی و کنترل اضافه‌بار در سرورهای SIP پیشنهاد شده است. مقالهٔ [۲۹] تکنیک جدید توازن بار را برای گردش پیام‌های پروتکل SIP ارائه می‌دهد تا درخواست‌هایی را که به یک خوشه از سرورهای SIP که بر اساس DHT (Distributed Hash Tables) ساخته می‌شوند، ارسال کند. مقالهٔ [۳۰] آزمایش‌های گسترده‌ای انجام داده و داده‌های جمع‌آوری‌شده را برای توصیف عملکرد سرورهای SIP به‌عنوان تابعی از تعداد هسته‌های CPU و میزان ورود تماس، تجزیه و تحلیل کرده است. بر اساس این تجزیه و تحلیل، نشان می‌دهد که چگونه می‌توان پارامترهای مختلف زمان‌بندی را به‌عنوان تابعی از تعداد هسته‌های CPU پیکربندی کرد تا به پیشرفت قابل توجهی در عملکرد سرور SIP رسید. در مقالهٔ [۳۱] یک روش با نام LBBSRT پیشنهاد شده است که با استفاده از یک کنترل‌کننده و تأخیر سرورها سعی در افزایش کیفیت سرویس تماس‌ها و جلوگیری از اضافه بار دارد.

به‌عنوان نتیجه‌گیری این بخش، شایان ذکر است که در کارهای پیشین تأثیر حالت و نگهداری آن بر منابع پروکسی SIP به‌صورت عملی مورد بررسی قرار نگرفته است. پس در این مقاله در نظر داریم به‌طور دقیق به تحلیل و بررسی پارامترهای میزان مصرف CPU، گذردهی و میانگین زمان برقراری تماس از پیکربندی‌های مختلف پروکسی سرور SIP در دو مد حالتمند و غیرحالتمند بپردازیم. در ادامهٔ این مقاله برای کارهای آتی، قصد بررسی اثر توزیع حالت بر مسائل امنیتی از قبیل حریم خصوصی و قابلیت اعتماد را داریم.

## ۵. ارزیابی میزان مصرف CPU سرور SIP

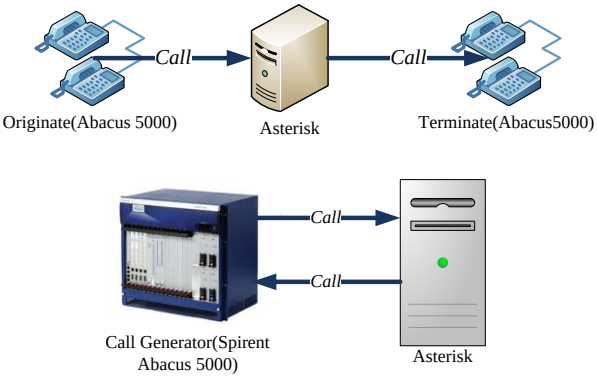
### ۵.۱. بررسی منابع سرور Asterisk در سناریوهای متفاوت

در این بخش، به ارزیابی Asterisk در بارهای تماس کم می‌پردازیم تا بدین ترتیب میزان مصرف CPU برای کارکردهای متنوعی را که فراهم می‌کند، به دست آوریم.

بستر آزمایش شامل مجموعه‌ای از کاربران (دستگاه Spirent Abacus 5000 [۲۱]) است که تقاضاهای خود را از طریق یک

1. IP Multimedia Subsystems

پراکسی Asterisk، به دسته‌ای دیگر از کاربران (دستگاه Spirent Abacus 5000 [۲۱]) ارسال می‌کنند. کارکرد این پراکسی با استفاده از نرم‌افزار OProfile [۲۲] رصد می‌شود. از دستگاه Spirent Call Generator هم به‌عنوان تماس‌گیرنده[1] و هم تماس‌پذیرنده[2] استفاده شده است. این مجموعه به ما این امکان را می‌دهد بدون اینکه Spirent به گلوگاهی تبدیل شود، نرخی در حدود ۱۰،۰۰۰ تماس بر ثانیه را تولید کنیم. همان گونه که در شکل (۴) مشخص است، می‌توان در این سناریو برای تقلیدِ هم تماس‌پذیرنده و هم تماس‌گیرنده از دستگاه تولید ترافیک Spirent استفاده کرد. در این آزمایش، Asterisk خود را برای تست بین دو طرف آن قرار دادیم.




**شکل (۴): جایگاه Asterisk و Spirent Abacus 5000 در سناریوی تست**

در آزمایش‌ها، سرور Asterisk به پنج وضعیت زیر پیکربندی شد. هر وضعیت بیانگر یک سرویس است که سرور برای تماس فراهم می‌کند. وضعیت‌های بعدی، سرویس‌های اضافی را فراهم می‌کنند (لذا منابع بیشتری را صرف می‌کنند). در ضمن، این سرور برای ترجمهٔ URI به آدرس IP نقطهٔ نهایی، در دیکشنری خود جست‌وجو[3] می‌کند. وضعیت‌های مختلف عبارت‌اند از:

۱. غیرحالتمند و بدون جست‌وجو[4]: در نتیجهٔ پرداختن به تماس، هیچ حالت مربوط به تماسی حفظ نمی‌شود. همچنین، این پیام حاوی اطلاعات کافی از جمله آدرس IP در SIP URI نقطهٔ نهایی است؛ لذا نیازی به جست‌وجو نیست.

۲. غیرحالتمند و با جست‌وجو[5]: در اینجا نیز مانند مورد قبل هیچ حالتی برای تماس حفظ نمی‌شود اما در پایگاه داده به‌منظور نگاشت URI به آدرس IP، تنها یک جست‌وجو انجام می‌شود.

۳. حالتمند تراکنش با جست‌وجو[6]: در این حالت، به‌منظور نگاشت URI پیام به یک آدرس IP، یک جست‌وجو انجام می‌شود. علاوه بر این، حالت فقط برای هر تراکنش به‌صورت جداگانه نگه داشته می‌شود.

۴. حالتمند مکالمه با جست‌وجو[7]: در اینجا نیز آدرس IP مورد جست‌وجو قرار می‌گیرد. علاوه بر این، حالت برای کل مدت تماس، که شامل چندین تراکنش می‌شود، نگه داشته می‌شود.

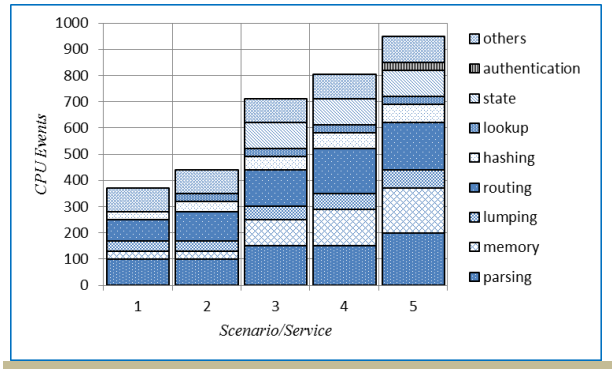
۵. حالتمند مکالمه با تأیید مجوز[8]: این وضعیت مانند قبلی است، علاوه بر اینکه پراکسی سرورها مجوزهای کاربران را نیز بررسی می‌کند.

با هر بار اجرا، سرور Asterisk در یکی از وضعیت‌های گفته‌شده در بالا پیکربندی می‌شود و دو کاربر دستگاه Spirent برای مدت ۱۰ دقیقه با نرخ ۱ تماس بر ثانیه اقدام به برقراری و قطع تماس از طریق این سرور می‌کنند. در طول این مدت، OProfile قسمت‌های گوناگون کارکرد Asterisk را پروفایل‌بندی می‌کند. نتایج حاصل از این اجراها در شکل (۵) نشان داده شده است.

محورهای عمودی در شکل (۵) نشان می‌دهد که با اجرای کارکردهای بیشتر توسط سرور، مقدار منابع بیشتری از CPU مصرف می‌شود. این نتیجه قابل انتظار بود اما اندازهٔ این افزایش قابل توجه است. در پایه‌ای‌ترین پیکربندی که در آن سرور غیرحالتمند است و هیچ جست‌وجویی انجام نمی‌دهد، میزان CPU مصرف‌شده تقریباً یک‌سوم حالتی است که در آن یک سرور به جست‌وجو می‌پردازد، حالت تراکنش را نگه می‌دارد و عملیات تأیید مجوز را انجام می‌دهد. با توجه به این

1. Originate
2. Terminate
3. Lookup
4. Stateless with No Lookup
5. Stateless with Lookup
6. Transaction Stateful with Lookup
7. Dialog Stateful with Lookup
8. Dialog Stateful with Authentication

موضوع، بدیهی است که مخارج برقراری یک تماس ساده بسته به پیچیدگی سرویس فراهم‌شده توسط سرور SIP بوده و به مقدار زیادی متغیر است.




**شکل (۵): هزینهٔ عملکردهای متفاوت سرور**

در مقایسه با ستون عدم جست‌وجو، مابقی موارد (از غیرحالتمند تا تأیید مجوز) بار پردازشی جست‌وجو دارند. این بار شامل جست‌وجو در یک پایگاه داده و یا یک حافظهٔ نهان[1] داخلی می‌شود. این بار پردازشی جست‌وجو به‌صورت یک نوار باریک در شکل (۴) منعکس شده است. به‌طور مشابه، برای نگهداری حالت و تأیید مجوز نیز شاهد رشد مصرف CPU هستیم. به‌علاوه، اکثر کارکردهای سرور همگام با افزایش سرویس افزایش می‌یابند؛ به‌خصوص شاهد افزایش قابل توجه هزینه‌های مربوط به تجزیهٔ[2] پیام، حافظه[3] و حالت در صورت افزایش سرویسِ ارائه‌شده هستیم. در بیشتر سرورهای SIP عمل تجزیهٔ پیام به‌کندی انجام می‌شود؛ یعنی تا جایی که لازم باشد پیام را تجزیه می‌کنند، البته به‌طوری که بتوان یا تقاضا را به مرحله بعد ارسال یا پاسخ مناسبی را ایجاد کرد. سرویس‌های سنگین‌تر نیاز دارند که بیشترِ پیام تجزیه شود. جست‌وجوها هزینه‌های تجزیه را به میزان قابل توجهی تغییر نمی‌دهند؛ زیرا URI تقاضای همواره باید تجزیه شود تا تعیین کند که آیا نیازی به انجام جست‌وجوی مسیر هست یا خیر. لذا هزینهٔ تجزیه در دو سناریوی اول تقریباً میزان منابع یکسانی را صرف می‌کنند. با این حال برای ایجاد و حفظ حالت، سرپیام[4]های بیشتری می‌بایست تجزیه شوند. دلیل آن این است که نگهداری حالت نیاز دارد که تقاضا به‌صورت منحصربه‌فرد مشخص شود. این کار با در نظر گرفتن مجموعه‌ای از فیلدها شامل From و To انجام می‌شود و لذا تمامی این سرپیام‌ها باید تجزیه شوند.

نگهداری حالت نیازمند تخصیص[5] و آزادسازی[6] حافظه است و این امر باعث افزایش حافظهٔ پردازشگر در سه سناریوی آخر می‌شود. این حالت مازاد باعث افزایش تجزیه و افزایش پردازش در حافظهٔ سرور می‌شود. با توجه به نمودار می‌توان مشاهده کرد که در نرخ تقاضای اندکِ ۱ تماس بر ثانیه، «حالتمند مکالمه بودن»[7] یا «حالتمند تراکنش بودن»[8] نسبت به غیرحالتمند بودن به‌ترتیب ۲ برابر و ۱/۷۵ برابر هزینه‌برتر است؛ لذا نگهداری حالت گزینهٔ خوبی برای توزیع است و در این مقاله نشان خواهیم داد که سرورهای حالتمند و غیرحالتمند در صورت افزایش نرخ تماس چگونه رفتار می‌کنند.

## ۲.۵. تأثیر حفظ حالت با افزایش نرخ تماس بر منابع

هدف از آزمایش این بخش، به‌دست‌آوردن چگونگی رفتار سرورهای حالتمند و غیرحالتمند در صورت افزایش بار می‌باشد. پراکسی سرور Asterisk برای اجرا در دو وضعیت زیر پیکربندی می‌شود:

۱. غیرحالتمند با جست‌وجو؛

۲. حالتمند تراکنش با جست‌وجو.

به‌منظور درک رفتار سرور، بار تا جایی که سرور به ۱۰۰٪ بهره‌برداری از CPU رسید، افزایش یافت. برای اطمینان از اینکه سرور فقط به‌دلیل بهره‌گیری[9] از CPU به اشباع رفته است (و نه حافظه)، Asterisk را با مقدار حافظهٔ کافی (۱۰۲۴ مگابایت) پیکربندی کردیم. همچنین از واسط‌های اترنت گیگابایت روی شبکهٔ آزمایشگاه IP-PBX[10] برای این تست

1. Cache
2. Parsing
3. Memory

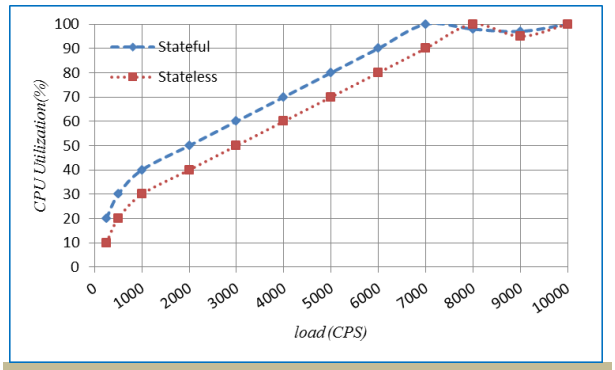
4. Header
5. Allocation
6. De allocation
7. Dialog stateful
8. Transaction stateful
9. Utilization
10. http://ippbx-lab.um.ac.ir/

استفاده نمودیم.

دستگاه Abacus 5000 ارسال بار را از ۲۰ تماس بر ثانیه شروع و به‌صورت پلکانی (۲۰ تماس بر ثانیه) افزایش می‌دهد. سرور Asterisk زمانی که مقدار بهره‌گیری از CPU، ۱۰۰٪ شده و نیز با افزایش بار گذردهی تماس افزایش نیابد، به اشباع می‌رود. همچنین در آستانهٔ نقطهٔ اشباع، افزایش زیادی در پیام‌های SIP 500 Server Busy دیده می‌شود و ارسال مجدد تقاضاهای تماس از سمت کاربران نیز افزایش می‌یابد. نتایج در شکل (۶) و (۷) نشان داده شده است.

### ۵،۲،۱. منابع حافظه‌ای

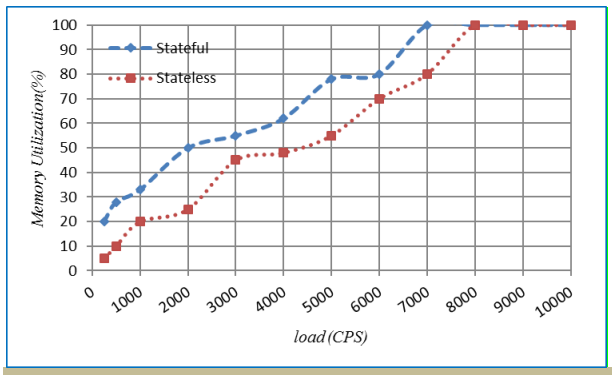
همان‌طور که در شکل زیر آورده شده است، مصرف CPU توسط سرور حالتمند با افزایش نرخ تماس با سرعت بالاتری نسبت به سرور غیرحالتمند افزایش می‌یابد. سرور حالتمند در حدود نرخ ۷۰۰۰ cps به اشباع می‌رود اما سرور غیرحالتمند در نرخ تقریبی ۷۰۰۰ cps وارد اشباع می‌شود. این اختلاف بین سرورهای حالتمند و غیرحالتمند مبنای به دست آوردن گذردهی بالاتر از طریق توزیع توابع حفظ حالت است.




**شکل (۶): افزایش مصرف CPU با زیادشدن نرخ تماس**

### ۲،۵،۲. منابع حافظه‌ای

همان‌طور که در شکل (۷) دیده می‌شود، میزان مصرف حافظهتوسط سرور حالتمند با افزایش نرخ تماس با سرعت بیشتری نسبت به سرور غیرحالتمند افزایش می‌یابد. سرور حالتمند در حدود نرخ ۷۰۰۰ cps به اشباع می‌رسد، اما سرور غیرحالتمند در نرخ تقریبی ۸۰۰۰ cps اشباع می‌شود. همان گونه که از شکل (۶) و (۷) مشخص است، الگوی منابع مصرفی (پردازنده و حافظه) مشابه هم است. در بخش بعد، یک الگوریتم توزیع حالت معرفی می‌کنیم. این الگوریتم درک بهتری دربارهٔ دلیل افزایش گذردهی توسط توزیع حالت می‌دهد.




**شکل (۷) : افزایش مصرف حافظه با زیادشدن نرخ تماس**

## ۶. توزیع حالت

### ۱،۶. پیکربندی‌های متفاوت سرورهای SIP

حالت ساده‌ای را در نظر بگیرید که در آن، یک تقاضا از طریق دو سرور سری ($S_1$ و $S_2$) در یک دامنه عبور می‌کنند. این همانند سناریوی تصویرشده در شکل (۲) است. پیکربندی‌های ممکن که می‌توان این سرورها را طبق آن تنظیم کرد عبارت‌اند از:

۱. هر دو حالتمند،

۲. یکی حالتمند و دیگری غیرحالتمند؛

۳. هر دو غیرحالتمند.

در حالت اول، هر سرور به لطف داشتن وضعیت حالتمند، برای هر تقاضایی که از آن عبور می‌کند، حالت را حفظ می‌کنند. در نتیجه، بیشترین تعداد تقاضایی که قادر به سرویس‌رسانی به آن‌هاست، برابر با «حد اشباع یک سرور حالتمند ($TH_{Stateful}$)» (در آزمایش‌های ما این مقدار تقریباً برابر ۷۰۰۰ تماس بر ثانیه به دست آمد) خواهد بود. از آنجا که هر دو سرور بار تقاضای یکسانی را می‌بینند، بیشترین گذردهی سیستم در حدود ۷۰۰۰ تماس بر ثانیه خواهد بود. همچنین هر دو سرور به‌طور ۱۰۰٪ از CPU خود بهره‌گیری می‌کنند. حالت دوم زمانی رخ می‌دهد که یکی از این دو سرور به‌صورت غیرحالتمند پیکربندی شود. این سیستم به کار خود تا گذردهیِ حدودا ۷۰۰۰ تماس بر ثانیه ادامه خواهد داد؛ زیرا توقف با سرور حالتمند است و بنابراین

گذردهی کلی را دیکته می‌کند. سرور حالتمند با ۱۰۰ درصد بهره‌گیری از CPU عمل می‌کند($S_1$) اما سرور غیرحالتمند به مقدار کمتری مورد بهره‌گیری قرار می‌گیرد ($S_2$). حالت سوم مربوط به زمانی است که هر دو سرور غیرحالتمند باشند. در این حالت، بیشترین گذردهی سیستم برابر با «حد اشباع یک سرور غیرحالتمند ($TH_{Stateless}$)» (در آزمایش‌های ما این مقدار تقریباً برابر ۸۰۰۰ تماس بر ثانیه به دست آمد) خواهد بود. این گذردهی به مقدار قابل توجهی بیشتر از گذردهی در دو حالت نخست است، اما در اینجا به هیچ وجه حالت در سیستم نگه داشته نمی‌شود؛ این سیستم برای تقاضاهای تماسی که نیاز به حفظ حالت برنامه دارند، قابل استفاده نیست.

بنابراین برای دو سرور سری، که هر سرور از قبل (به‌صورت ایستا) به‌صورت حالتمند یا غیرحالتمند پیکربندی می‌شود، بیشترین گذردهی برای تقاضاهای تماسی که نیاز به حفظ حالت برنامه دارند، برابر با حد اشباع یک سرور حالتمند است. با این حال همان‌طور که در حالت دوم مشاهده شد، $S_1$ سرور غالب است و $S_2$ کمتر مورد بهره‌گیری قرار می‌گیرد. اگر سرور $S_1$ نیمی از بار تقاضاهای خود را به دوش سرور $S_2$ بگذارد تا به‌صورت حالتمند به آن‌ها بپردازد (خودش برای این تقاضاها غیرحالتمند گردد)، آنگاه می‌توان مطمئن بود که در هر مرحله از هر دو سرور به‌طور برابر بهره‌گیری می‌شود. در نتیجه این سرورها قادر به پشتیبانی از تماس بیشتری خواهند بود.

در نتیجه، یک الگوریتم توزیع حالت می‌بایست نیازمندی‌های زیر را ارضا کند:

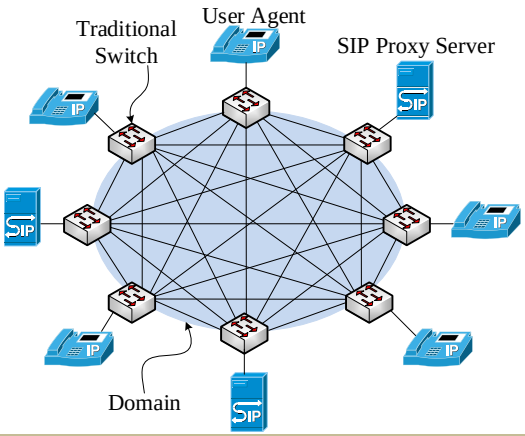
اولاً حالت مربوط به تماس باید از سوی گرهی در مسیر تقاضای تماس نگهداری شود. سایر گره‌ها این تماس را به‌صورت غیرحالتمند حواله می‌دهند و متحمل بار پردازشی مربوط به حفظ حالت نمی‌شوند؛ ثانیاً باید به‌صورت پویا تعیین کرد که در یک مسیر، کدام گره باید حالت تماس را نگه دارد، به‌طوری که مادامی که منابعی برای پرداختن به یک تماس ورودی به‌صورتی حالتمند موجود باشند، سیستم قادر باشد به آن تماس بپردازد.

به‌منظور تبیین ماهیت چنین الگوریتمی، این مسئله را به‌صورت یک مسئلهٔ بهینه‌سازی مدل می‌کنیم. . البته به‌دلیل داشتن یک دید کلی از وضعیت کل شبکه، از تکنولوژی نرم‌افزارمحور بهره برده و در ابتدا یک چهارچوب بدیع در این زمینه را پیشنهاد می‌دهیم.

## ۷. رویکرد پیشنهادی

### ۷.۱. چهارچوب پیشنهادی

در این بخش به معرفی چهارچوب پیشنهادی مبتنی بر شبکه‌های نرم‌افزارمحور می‌پردازیم. این چهارچوب در شکل (۸) آورده شده است. هدف ما حرکت به‌سمت شبکه‌های نرم‌افزارمحور در قالبِ این رویکرد است. شکل (۸)، یک دامنه از شبکهٔ SIP موجود را نشان می‌دهد که شبکه‌ای از سوئیچ‌های سنتی، زیرساخت ارتباطی بین عامل‌های کاربر و پروکسی سرورهای SIP را تشکیل داده‌اند و سطوح کنترل و داده از هم جدا نیستند. با افزایش تعداد عامل‌های کاربر امکان عدم توازنِ بار بین پروکسی‌های وجود دارد. در نتیجه با گذشت زمان برخی از پروکسی‌ها با اضافه‌بار مواجه شده، درحالی‌که برخی دیگر نسبتاً بیکار هستند. این موضوع باعث ایجاد تأخیر در برقراری تماس، از دست رفتن تماس‌ها و همچنین عدم استفادهٔ بهینه از منابع پروکسی‌ها می‌شود.




**شکل (۸): چهارچوب شبکه‌های سنتی SIP**

همان طور که در شکل (۹) مشاهده می‌شود، می‌توان برای داشتن یک دید کلی از شبکهٔ SIP، از چهارچوب پیشنهادی شکل (۹) استفاده کرد. همان طور که در این شکل مشاهده می‌شود، در سطح زیرساخت (لایه داده) پروکسی سرورهای SIP و دیگر تجهیزات SIP مانند تلفن ها وجود دارد به همراه

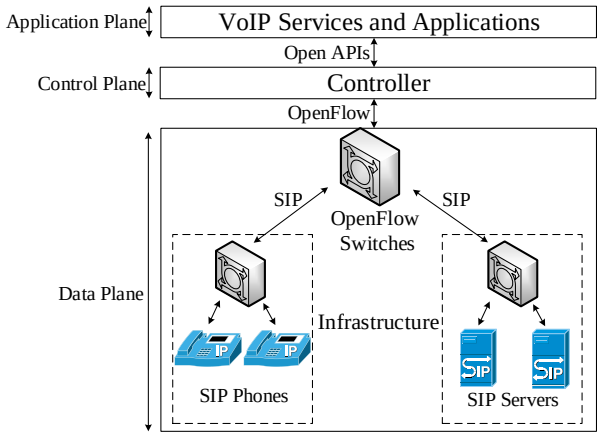
سوئیچ‌های OpenFlow. یک سوئیچ OpenFlow دارای جداول جریانی است که فیلدِ قاعدهٔ انطباق آن حاوی اطلاعات سرآیند لایهٔ ۲ است. به‌محض ورود بستهٔ جریانی به این سوئیچ، فیلدهای سرآیند بسته با فیلدهای قاعدهٔ انطباقِ موجود در جدول جریان مطابقت داده می‌شود، اگر تطابق پیدا کرد، سوئیچ اقدام مربوطه را روی بسته اعمال می‌کند. این اقدام می‌تواند شامل موارد زیر باشد:

۱. ارسال بسته‌ها به پورت خروجیِ خاصی

۲. بسته‌بندی[1] کردن بسته‌ها و ارسال به کنترل‌کننده

۳. حذف آن‌ها و...

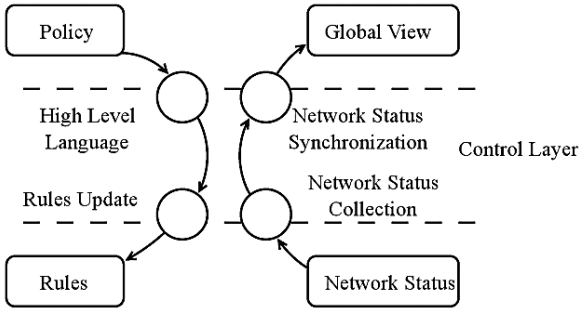
اگر بسته با هیچ قاعدهٔ انطباقی تطابق پیدا نکرد، بسته را در یک پیامِ Packet-In بسته‌بندی و برای کنترل‌کننده می‌فرستد. با دریافت پیامِ Packet-In توسط کنترل‌کننده، یک یا چند برنامهٔ کاربردی شروع به اجرا شده و سپس قاعدهٔ مناسب توسط پیام Flow-Mod در جدول جریان سوئیچ نصب می‌گردد، به‌نحوی‌که بسته‌های بعدیِ جریان مذکور بتوانند در سوئیچ پردازش شوند. علاوه بر این کنترل‌کننده می‌تواند بستهٔ خاصی را از طریق پیامِ Packet-Out به سطح داده و سوئیچ‌ها ارسال کند. این را نیز بایستی در نظر داشت که کل اطلاعات شبکه با استفاده از این سوئیچ‌ها و پروتکل LLDP در اختیار سطح کنترل برای تصمیم‌گیری قرار می‌گیرد؛ یعنی با داشتن کل اطلاعات سطح زیرساخت در سطح کنترل می‌توان یک دید یکپارچه نسبت به کل شبکه داشت و تصمیم‌گیری واحدی انجام داد.




**شکل (۹): چهارچوب پیشنهادی مبتنی بر شبکه‌های نرم‌افزارمحور**

1. Encapsulate

صفحهٔ کنترل، لایهٔ برنامه‌های کاربردی و لایهٔ زیرساخت را به هم متصل می‌کند. شکل (۱۰) طرح منطقیِ لایهٔ کنترل را نشان می‌دهد که شامل چهار مؤلفهٔ اصلی به نام‌های زبان سطح بالا[2]، فرایند به‌روزرسانی قانون[3]، فرایند جمع‌آوری وضعیت شبکه[4] و فرایند همگام‌سازی وضعیت شبکه[5] است.




**شکل (۱۰): طراحی منطقی لایه کنترل**

بنابراین مطابق شکل (۱۱) هر آنچه اطلاعات از لایهٔ زیرساخت برای تصمیم‌گیری مورد نیاز است، در اختیار کنترلر است. حال در لایهٔ برنامهٔ کاربردی بایستی تصمیمات مناسب با استفاده از یک مدل برنامه‌ریزی خطی گرفته شود. شکل (۱۱) جزئیات چهارچوب پیشنهادی را نشان می‌دهد. همان گونه که در این شکل مشاهده می‌شود، برای تصمیم‌گیری یکپارچه برای توزیع حالت در شبکه SIP در لایهٔ کنترلِ چهارچوب پیشنهادی، سه ماژول طراحی شده است. دو ماژول برای کنترل سوئیچ‌ها و پروکسی‌های SIP با توجه به خروجیِ ماژول بهینه‌ساز (optimizer). خروجی ماژول بهینه‌ساز تصمیم‌گیری برای نحوهٔ توزیع حالت است.

- ماژول networking resource controller

این ماژول مسئولِ جمع‌آوری اطلاعات شبکه مانند توپولوژی سوئیچ‌ها، لینک‌ها، ظرفیت در دسترس آن‌ها و همچنین توان[6] آن‌هاست.

- ماژول computational resource controller

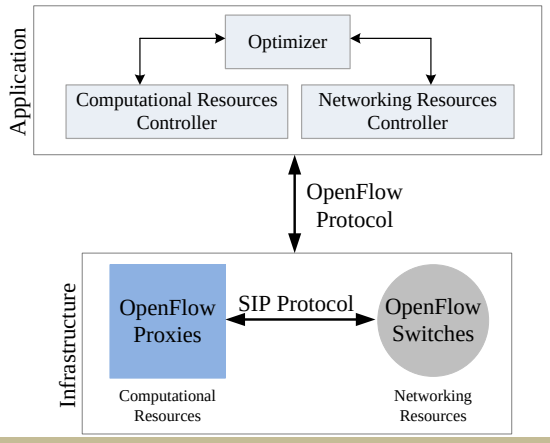
جمع‌آوری اطلاعاتِ توپولوژیِ پروکسی‌ها، ظرفیت در

2. High Level Language
3. Rules Update
4. Network Status Collection
5. Network Status Synchronization
6. Power

دسترس[1] آن‌ها و همچنین توان در دسترس آن‌ها بر عهدهٔ این ماژول است.

- ماژول optimizer

این ماژول شامل یک مدل ریاضیِ مسیریابی از نوع برنامه‌ریزی خطی[2] برای تخصیص بهینهٔ منابع پروکسی‌ها و سوئیچ‌هاست. ورودی این ماژول از ماژول‌های NRC و CRC است.




**شکل (۱۱): جزئیات چهارچوب پیشنهادی**

## ۲.۷. مسئلهٔ بهینه‌سازی توزیع حالت پیشنهادی

در این بخش به بررسی دقیق و با جزئیاتِ ماژول بهینه‌ساز یا optimizer می‌پردازیم. این ماژول با توجه به اطلاعات دریافتی از سطح زیرساخت شامل یک مدل برنامه‌ریزی خطی است که می‌تواند تصمیمات یکپارچه‌ای در مورد توزیع حالت بگیرد. در ادامه به تشریح این مدل برنامه‌ریزی خطیِ مربوط به ماژول بهینه‌ساز می‌پردازیم.

می‌توان پراکسی سرورهای SIP را در انواع توپولوژی شبکه نشان داد. در نتیجه می‌توان آن‌ها را به‌صورت مجموعه‌ای از نودها و به‌صورت یک گراف بیان کرد. تقاضاهای برقراری تماس از طریق یکی از گره‌های این گراف وارد سیستم می‌شوند، به‌وسیلهٔ لینک‌های داخل گراف از مجموعه‌ای از گره‌ها عبور می‌کنند و از یک گره پراکسی خارج می‌شوند. این گره آخر یا این تقاضا را به زیرساخت اینترنت و یا به عامل کاربر گیرندهٔ تماس حواله می‌کند. یک توپولوژی می‌تواند گره‌های ورودی بسیاری داشته باشد. این معادل است با داشتن یک گره منبع مجازیِ o که تماس‌ها را تولید و به گره‌های ورودی ارسال می‌کند. به‌طور مشابه، ممکن است چندین گره خروجی در سیستم وجود داشته باشد و این معادل است با یک گره گودالیِ z که تمامی گره‌های خروجی، تماس‌های خود را به آن مسیردهی می‌کنند. مزیت این روش این است که می‌توان شبکه را به‌صورت یک توپولوژی تک منبع، تک گودال در نظر گرفت و بدون از دست رفتن عمومیت، به یک فرمول بندی دقیق‌تری دست پیدا کرد.

در یک سیستم، سرور $i$ و یک تقاضای تماس مخصوص را که از گره $i$ عبور می‌کند، در نظر بگیرید. یک گره بالادست[3] وجود دارد که این تماس از آن به گره $i$ مسیردهی شده است. نام این گره را $u$ می‌گذاریم. این تماس بیانگر تمامی تماس‌های مسیردهی‌شده از $u$ به $i$ یا به‌عبارتی $C_{ui}$ است. برای مابقی این بحث، تمامی جملات ترافیک بیانگر نرخ تماس (بار) هستند. لذا $C_{ui}$ عبارت است از بار تماس از $u$ به $i$. $C_{ui}$ دارای دو مؤلفه است:

یکی تماس‌هایی که به ازای آن‌ها حالت از قبل حفظ شده است($C_{ui}^{BSF}$) و دیگری تماس‌هایی که به ازای آن‌ها قرار است حالت حفظ شود($C_{ui}^{\widetilde{SF}}$).

بنابراین داریم $C_{ui} = C_{ui}^{BSF} + C_{ui}^{\widetilde{SF}}$. حال مجموع شارش ورودی به گره $i$ یا $C_i$ عبارت است از مجموع تمام شارش‌های $C_{ui}$، ($C_i = \sum_{u \in US_i} C_{ui}$).

به‌طور مشابه، برای مجموع بار تماس‌هایی که به‌ازای آن‌ها حالت از قبل نگهداری شده و مجموع بار تماس‌هایی که به‌ازای آن‌ها قرار است حالت نگهداری شود، معادلاتی خواهیم داشت که به‌ترتیب عبارت‌اند از $C_i^{\widetilde{SF}} = \sum_{u \in US_i} C_{ui}^{\widetilde{SF}}$ و $C_i^{BSF} = \sum_{u \in US_i} C_{ui}^{BSF}$.

$US_i$ عبارت است از مجموعه تمام سرورهای بالادست ممکن برای گره $i$. در گره $i$، بار ورودی $C_i$ مجدداً بین تمامی مسیرهای پایین‌دست[4] ممکن توزیع خواهد شد. برای سرور

1. Available capacity
2. Linear programming
3. Upstream node
4. Downstream

پایین‌دستِ d، اگر بار تماس را $C_{id}$ بنامیم و $DS_i$ بیانگر تمامی سرورهای پایین‌دست برای گره i باشد، طبق قانون بقای شارش، $C_i = \sum_{d\in DS_i} C_{id}$ و $C_{id}$ شامل سه مؤلفه خواهد بود:

- ✓ $C_{id}^{FSF}$: شارش تماس از i به d که حالت آن از قبل توسط گرهی قبل از i نگه داشته شده است؛
- ✓ $C_{id}^{SF}$: شارش تماس از i به d که برای آن، گره i حالت را نگه داشته است؛
- ✓ $C_{id}^{\widetilde{SF}}$: شارش تماس از i به d که قرار است برای آن حالت نگه داشته شود.

«قیود شارش» معادلات زیر را الزام می‌کنند:

$$C_i^{BFS} = \sum_{d\in DS_i} C_{id}^{FSF}$$
$$C_i^{\widetilde{SF}} = \sum_{d\in DS_i} C_{id}^{SF} + \sum_{d\in DS_i} C_{id}^{\widetilde{SF}}$$

برای گره پایین‌دستِ d، مقدار شارشی که قبلاً از گره i حالت آن حفظ شده است برابر است با $C_{id}^{BSF} = C_{id}^{FSF} + C_{id}^{SF}$. برای ترافیک از گره بالادست u به i، به‌طور مشابه یک شکست و توزیع رخ داده است. لذا داریم $C_{ui}^{BSF} = C_{ui}^{FSF} + C_{ui}^{SF}$. این تضمین می‌کند که در هر گره، متغیرهای تصمیم $C_{id}^{FSF}$، $C_i^{SF}$ و $C_i^{\widetilde{SF}}$ خواهند بود. قیودی که در بالا ذکر شد تضمین می‌کنند که از هر دو طرف، به‌صورت انحصاری به حالت پرداخته شود. با وجود این، بایستی تضمین شود که حداقل در یکی از گره‌ها در مسیر حالت نگه داشته می‌شود. بنابراین تعداد تماس‌هایی که قرار است برای هر شارش از گره خروجی k به گره مجازی گودال، حالت آن‌ها نگه داشته شود باید برابر صفر باشد ($C_{kz}^{\widetilde{SF}} = 0$).

در بخش ۳ دیدیم که میزان بهره‌گیری از CPU برای پرداختن حالتمند به تقاضاها با پرداختن غیرحالتمند به تقاضاها متفاوت می‌باشد. تعداد تماس‌هایی که به‌صورت حالتمند و غیرحالمتند در گره i به آن‌ها پرداخته می‌شود، به‌ترتیب عبارت است از $\sum_{d\in DS_i} C_{id}^{SF}$ و $\sum_{d\in DS_i}\left(C_{id}^{\widetilde{SF}} + C_{id}^{FSF}\right)$. بنابراین اگر فرض شود که میزان بهره‌گیری از CPU ناشی از حواله کردن حالتمند تقاضا در گره i، که آن را با $U_i^{SF}$ نمایش می‌دهیم به تعداد تقاضاهایی که به‌صورت حالتمند به آن‌ها پرداخته می‌شود وابسته است، آنگاه خواهیم داشت $U_i^{SF} = f^{SF}\left(\sum_{d\in DS_i} C_{id}^{SF}\right)$.

به‌طور مشابه میزان بهره‌گیری از CPU در وضعیت غیرحالتمند برابر است با $U_i^{SL} = f^{SL}\left(\sum_{d\in DS\_i} C_{id}^{\widetilde{SF}} + C_{id}^{FSF}\right)$.

مجموع بهره‌گیری از CPU در گره i برابر مجموع این دو معادله است. این مجموع نباید از ۱۰۰٪ تجاوز کند و اگر نرمالیزه شوند، به معنای آن است که داریم $U_i^{SF} + U_i^{SL} \le 1$.

معادلات فوق، «قیود سیستم» را تشکیل می‌دهند. هدف، به‌دست‌آوردن یک توزیع حالت است، به‌طوری که گذردهی بیشینه شود. گذردهی اساساً عبارت است از شارش تماس از گره منبع مجازی o به تمامی گره‌های ورودی. در این نقطه می‌بایست حالت برای تمامی تماس‌ها نگه داشته شود، $C_{oa}^{FSF} = 0$، $C_{oa}^{SF} = 0$ به‌ازای گره ورودی a. سپس سعی می‌کنیم مجموع شارش‌ها از گره مجازی o به گره‌های ورودی یا به‌عبارت دیگر $\sum_{a\in \text{گره ورودی}} C_{oa}^{\widetilde{SF}}$ را بیشینه کنیم. مسئلهٔ کامل بهینه‌سازی خطی به فرم زیر است:

$$\text{Maximize} \sum_{a\in \text{entry node}} C_{oa}^{\widetilde{SF}} \quad \text{(I)}$$

Subject to

$$C_{oa}^{SF} = 0, C_{oa}^{FSF} = 0,\ \forall\, a \in \text{entry node}$$

$$\sum_{u\in US_i} (C_{ui}^{FSF} + C_{ui}^{SF}) = \sum_{d\in DS_i} C_{id}^{FSF} \quad \text{(II)}$$

$$\sum_{u\in US_i} C_{ui}^{\widetilde{SF}} = \sum_{d\in DS_i} C_{id}^{SF} + \sum_{d\in DS_i} C_{id}^{\widetilde{SF}} \quad \text{(III)}$$

$$C_{kz}^{\widetilde{SF}} = 0,\ \forall\, k \in \text{exit node}$$

$$f^{SF}\left(\sum_{d\in DS_i} C_{id}^{SF}\right) + f^{SL}\left(\sum_{d\in DS\_i} C_{id}^{\widetilde{SF}} + C_{id}^{FSF}\right) \le 1 \quad \text{(IV)}$$

حل این فرمول‌های بهینه‌سازی در تعیین حالت هر سرور به‌منظور بیشینه‌سازی گذردهی کمک خواهد کرد. البته باید به این نکته توجه کرد که برای سادگی محاسبات می‌توان فرمول‌بندی فوق را به‌صورت ذیل نیز بازنویسی کرد. این موضوع پیچیدگی را تا حدی کاهش خواهد داد؛ زیرا برای انجام محاسبات در ماژول بهینه‌ساز، کنترلر نیاز به منابع حافظه‌ای و پردازشی دارد و ممکن است خود آن به گلوگاه تبدیل شود. پس بهتر است عملیات محاسباتی حتی الامکان ساده‌تر گردد.

$$\text{Maximize} \sum_{a\in \text{entry node}} C_{oa}^{\widetilde{SF}} \quad \text{(I)}$$

Subject to

$$C_{oa}^{SF} = 0, C_{oa}^{FSF} = 0,\ \forall\ a \in \text{entry node}$$

$$\sum_{u\in US_i} (C_{ui}^{FSF} + C_{ui}^{SF}) - \sum_{d\in DS_i} C_{id}^{FSF} = 0 \quad \text{(II)}$$

$$\sum_{u\in US_i} C_{ui}^{\widetilde{SF}} - \sum_{d\in DS_i} C_{id}^{SF} - \sum_{d\in DS_i} C_{id}^{\widetilde{SF}} = 0 \quad \text{(III)}$$

$$C_{kz}^{\widetilde{SF}} = 0,\ \forall\ k \in \text{exit node}$$

$$f^{SF}\left(\sum_{d\in DS_i} C_{id}^{SF}\right) + f^{SL}\left(\sum_{d\in DS\_i} C_{id}^{\widetilde{SF}} + C_{id}^{FSF}\right) - 1 \le 0 \quad \text{(IV)}$$

در این فرمول‌بندی، مسائلی از قبیل «قیود مسیردهی» لحاظ نشده است. در این فرمول‌بندی فرض است که یک سرور قادر به تصمیم‌گیری و مسیریابی بین مسیرهای فراوان پایین‌دستِ ممکن می‌باشد (تقاضا را به کدام سمت بفرستد). در ضمن، تمامی مقصدهای تماس توسط هریک از این مسیرها قابل دستیابی هستند. با این حال در سناریوهای دنیای واقعی، تقاضای تماس از مسیری عبور خواهد کرد که توسط سازوکارهای مسیردهیِ شبکه تعیین می‌شوند. افزودن قیود مسیردهی کار ساده‌ای است. علاوه بر قیود تصریح‌شده در معادلهٔ (I)، باید قیودی که شارش ورودی در هر گره را به هریک از شارش‌های خروجی از آن گره ربط می‌دهند بیفزاییم؛ به‌عبارت دیگر می‌بایست برای گره $i$ قیودی به فرم $C_{id} = \wp_{id} * C_{id}, 0 \le \wp_{id} \le 1$ را تعریف کنیم. در این قید، $\wp_{id}$ کسری از شارش ورودی است که به مسیر پایین‌دست $d$ رفته است. علاوه بر این، باید با ملزم کردن اینکه جمع تمامی کسور برابر ۱ شود، ($\sum_{d\in DS_i} \wp_{id} = 1$)، بقای شارش را تضمین کنیم. در این صورت شارش خروجی $C_{id}$ تعدادی از کسور از پیش تعیین‌شده مجموع شارش ورودی $C_i$ است.

از شکل (۵) ملاحظه می‌شود که هم بهره‌گیری حالتمند و هم بهره‌گیری غیرحالتمند، روند خطی دارند. بنابراین می‌توان این توابع را به فرم $f^{SF}(x) = \frac{x}{TH\ Stateful}$ و $f^{SL}(x) = \frac{x}{TH\ Stateless}$ تخمین زد، که در آن‌ها $TH_{Stateful}$ و $TH_{Stateless}$ به‌ترتیب آستانه حالتمند و غیرحالتمند هستند. در آزمایش‌های ما $TH_{Stateful} \approx ۷۰۰۰$ و $TH_{Stateless} \approx ۸۰۰۰$ به دست آمد. در این حالت، فرمولاسیون بهینه‌سازی عبارت است از یک مسئلهٔ برنامه‌ریزی خطی[1] (LP) که می‌توان آن را به‌صورتی اثربخش حل کرد. در مابقی این مقاله فرض می‌شود که سیستم را می‌توان با یک مسئلهٔ برنامه‌ریزی خطی بیان کرد. حال می‌توان برای سناریوی دو سرورِ سری، در صورت داشتن مقادیر آستانه، از این فرمول‌بندی برای محاسبهٔ گذردهی بهینه استفاده کرد. پاسخ بهینه این است که هر سرور ۳۹۴۰ تماس بر ثانیه را به‌صورت حالتمند و مابقی (۳۹۴۰ تماس بر ثانیه) را به‌صورت غیرحالتمند نگه دارد. در این صورت مجموع گذردهی برابر ۷۸۸۰ تماس بر ثانیه خواهد بود که از گذردهیِ ایستا (تقریباً ۷۰۰۰ تماس بر ثانیه) بیشتر است. دلیل این افزایشِ گذردهی این است که از هر دو سرور به مقداری مساوی بهره‌گیری می‌شود؛ لذا گذردهی بالاتری به دست می‌آید. بنابراین حل این مسئله بهینه‌سازی، یک جواب بهینه را برای گذردهی نتیجه خواهد داد.

پاسخ مسئلهٔ برنامه‌ریزی خطی، مقادیر $C_{id}^{FSF}$، $C_{id}^{SF}$ و $C_{id}^{\widetilde{SF}}$ برای هر گره $i$ می‌باشد. با فرض خطی بودن توابع بهره‌گیری[2]، می‌توان قید (IV) را به‌صورت زیر بازنویسی کرد:

$$\left(\left\lceil\left(\sum_{d\in DS_i} C_{id}^{SF}\right)/\ TH_{Stateful}\right\rceil + \left[\left(\sum (C_{id}^{\widetilde{SF}} + C_{id}^{FSF})\right)/\ TH_{Stateless}\right]\right) \le 1 \quad \text{(V)}$$

همان طور که قبلاً ذکر شد، داریم $C_{id} = C_{id}^{FSF} + C_{id}^{SF} + C_{id}^{\widetilde{SF}}$. با جایگذاری $C_{id}^{FSF} + C_{id}^{\widetilde{SF}}$ در (V) و پس از مرتب‌سازی جملات داریم:

$$\sum_{d\in DS_i} C_{id}^{SF} \le \left(1 - \frac{\sum_{d\in DS_i} C_{id}}{TH_{Stateless}}\right) \Big/ \left(\frac{1}{TH_{Stateful}} - \frac{1}{TH_{Stateless}}\right) \quad \text{(VI)}$$

معادلهٔ (VI) نشان می‌دهد که مجموع حالاتی که باید نگه داشته شوند، تابعی از بار ورودی است. در نتیجه مادامی که شارش وروی از حد اشباع حالتمند ($TH_{Stateful}$) کمتر باشد، سرور می‌تواند تمامی تقاضاهایی را که هنوز حالتمند نیستند، به‌صورت حالتمند نگه دارد. به‌محض گذر شارش ورودی از $TH_{Stateful}$، سرور شروع به ترک حالت می‌کند. به‌محض گذر شارش ورودی از $TH_{Stateful}$، هریک از سرورها می‌بایست حفظ حالت را به سرورهای پایین‌دست بسپارند. این عملیات باید به‌نحوی باشد که مجموع حالات نگهداری‌شده در آن سرور از $TH_{Stateful}$ فراتر نرود. لذا سرورهای بالادست می‌توانند حالت را به سرورهای پایین‌دست بسپارند تا جایی که به

1. Linear Programming

2. Utilization constraint

گره‌های خروجی برسیم. به‌دلیل اینکه این گره‌ها هیچ مسیر پایین‌دستی که حالت را به آن‌ها بسپارند ندارند، با افزایش تعداد تماس‌ها به نگه داشتن حالت ادامه می‌دهند. این روند تا جایی که به بهره‌گیری بیشینه[1] نزدیک شوند، ادامه می‌یابد. در این نقطه آن‌ها یک پیام حاکی از «وضعیت اضافه‌بار»[2] را به سرورهای بالادستی مخابره می‌کنند. وقتی تمام مسیرهای پایین‌دست به اشباع رفت و خود سرور هم اشباع شد، این سرور یک پیام حاکی از وضعیت اضافه‌بار را به سرورهای بالادست مخابره می‌کند و در نهایت سیستم به‌کلی به اشباع می‌رود.

در هر سیستم، اندازه‌گیری‌ها را به‌صورت آنی نمی‌توان انجام داد. لذا باید بار ورودی و حالت نگهداری‌شده توسط آن سرور را به‌صورت دوره‌ای رصد کرد. فرض کنید بار ورودیِ رصدشده در سرور $i$ را با $O[t_i]$ و باری را که حالتمند است، با $O[t_i^{sf}]$ نشان دهیم. با توجه به معادلهٔ (VI)، می‌توان بر مبنای بار ورودی، مقدار حالتی که می‌توان توسط سرور حفظ نمود تا قید امکان‌پذیری را ارضا کرد محاسبه کرد. فرض کنید این مقدار را با $E[t_i^{SF}]$ نشان دهیم. چنانچه $E[t_i^{SF}] \geq O[t_i^{SF}]$ باشد، آنگاه سیستم حالت کمتری را نسبت به آنچه عملاً قادر به نگهداری است حفظ می‌کند و لذا نیازی به تغییر ندارد. اما اگر $E[t_i^{SF}] \leq O[t_i^{SF}]$ باشد، سیستم به واگذاری تعدادی از حالات نیاز دارد. در این صورت با انتخاب مسیرهای پایین‌دستی که می‌توان از طریق آن‌ها حالت را به یک سرور پایین‌دست محول کرد، حالت‌های لازم واگذار می‌شوند. این ما را قادر می‌سازد که شارش مازاد را روی یک مسیر به خصوص را که قرار است حالت در آن گره حفظ شود، جای داد.

برای لزوم حفظ یک حالت در یک نود خاص در یک مسیرِ شارشِ به‌خصوص از تقاضای تماس، چندین دلیل می‌تواند وجود داشته باشد: می‌تواند گره مذکور، گره خروجی برای آن تقاضاهای تماس باشد؛ و یا اینکه تمامی سرورهای پایین‌دست در آن مسیر به‌طور کامل تحت بهره‌گیری باشند؛ و یا اینکه فرض کنیم قیود مسیردهی نیز در جایگاه خود استوار هستند.

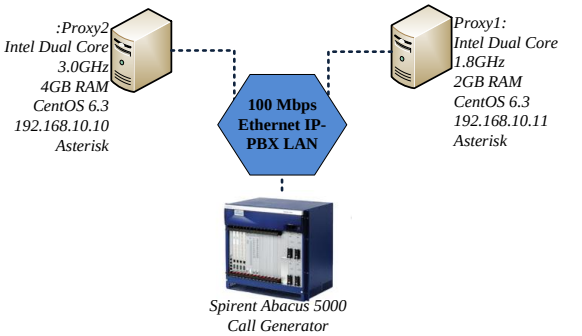
همان طور که قبلاً بحث شد، ممکن است واگذاری حالت بین تمامی مسیرها امکان‌پذیر نباشد. فرض کنید که $i$ یک گره خروجی است و همچنین $n$ مسیر پایین‌دست دارد. علاوه بر این فرض کنید که $k$ مسیر پایین‌دست هنوز اشباع نشده‌اند، در حالی که بقیهٔ آن‌ها ($n-k$ مسیر) اشباع شده‌اند (یعنی پیام‌های مبنی بر اضافه بار را به سرور $i$ ارسال کرده‌اند). در این صورت می‌توان بدون از دست دادن عمومیت، مسیرها را به‌گونه‌ای مرتب کرد که $i$ از ۱ تا $k$ اشباع نشده‌اند و از $k+1$ تا $n$ اشباع شده‌اند.

## ۸. پیکربندی و ارزیابی نتایج

### ۸.۱. ارزیابی سطح زیرساخت جهت توزیع حالت

برای ارزیابی از پیکربندی «دو سرور سری» بهره بردیم. این توپولوژیِ اساسی، زیرمجموعه‌ای از توپولوژی «چندین پروکسی سری» است که اولاً بلوک‌های ساختاری را برای توپولوژی‌های پیچیده‌تر فراهم می‌کند و همچنین ارزیابی آن‌ها می‌تواند بینشی را دربارهٔ توپولوژی شبکهٔ بزرگ‌تر به دست دهد. علاوه بر این بیانگر توپولوژی سرورها در دنیای واقعی نیز هست.

مطابق شکل (۱۲)، دسته‌ای از کاربرانِ توسط دستگاه Spirent درخواست‌های خود را به پراکسیِ اول می‌فرستند و این پراکسی این تقاضاها را به پراکسیِ دوم ارسال می‌کند. پراکسی دوم درخواست تماس را مجدداً به Spirent می‌فرستد. دو بار سناریو اجرا می‌شود: در اجرای اول، پراکسی‌ها به‌صورت ایستا پیکربندی می‌شوند و در اجرای دوم پراکسی‌ها الگوریتمِ توزیع پویا را اجرا می‌کنند.



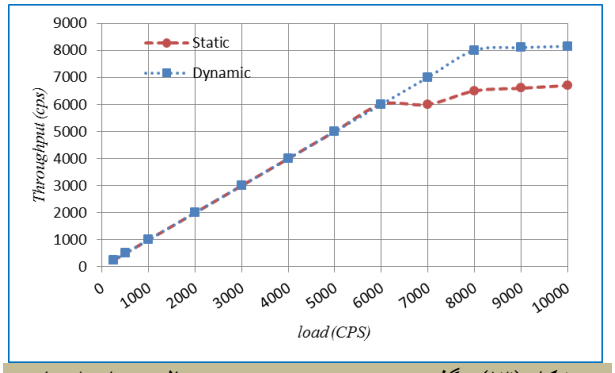

**شکل (۱۲): بستر آزمایش دو پروکسی**

نتایج مربوط به معیار ارزیابیِ گذردهی در شکل (۱۳) آورده

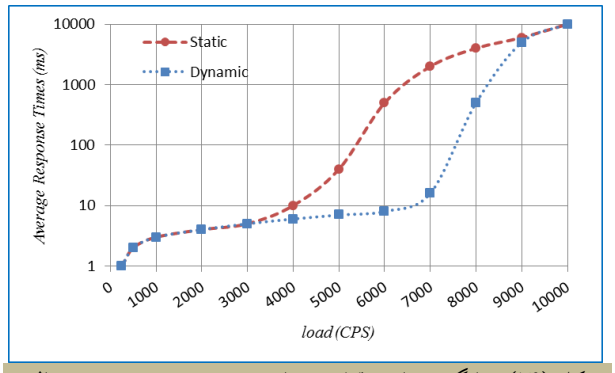
1. Maximum utilization
2. Overload

داده شده است. پیکربندی ایستا در حدود نرخ ۶۷۰۰ cps اشباع می‌شود و توزیع پویا تقریباً در نرخ ۸۱۵۰ cps اشباع می‌شود. یعنی کارایی را در حدود ۱۵٪ بهبود می‌بخشد. لذا با اینکه در بدو امر پیش‌بینی می‌شد که برای یک پیکربندی ایستای مشتمل بر دو سرور سری، بیشترین گذردهی برابر گذردهی بیشینهٔ یک سرور حالتمند است، اما اینطور نیست و گذردهی پایین‌تر است. در اصل با توزیع پویا می‌توان انتظار بهبود ۱۵ تا ۲۰ درصد در کارایی را داشته باشیم.

نتایج حاصل از اندازه‌گیری میانگین زمان‌های پاسخ در شکل (۱۴) نشان داده شده‌اند. می‌توان اثرات اشباع در افزایش زمان پاسخ، در هنگام استفاده از پیکربندی ایستا را در نرخ تقریبی ۶۵۰۰ تماس بر ثانیه به‌بعد مشاهده کرد. پیکربندی پویا قادر است زمان‌های پاسخ را تا نرخ بالاتری، پایین نگه دارد. به‌محض گذر از این حد نرخ، زمان‌های پاسخ به‌طور قابل توجهی بزرگ‌تر می‌شوند. دلیلش این است که تمامی تقاضاهای ازدست‌رفته می‌بایست در تمام مسیر مجدداً صادر شوند و این میانگین زمان پاسخ را افزایش می‌دهد.

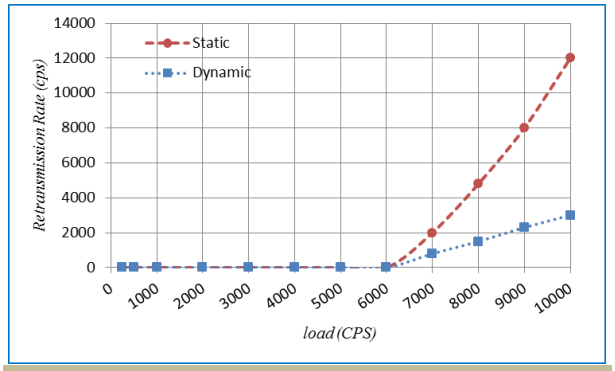



**شکل (۱۳) : گذردهی در دو وضعیت توزیع حالت پویا و ایستا**




**شکل (۱۴): میانگین زمان برقراری تماس در دو وضعیت توزیع حالت پویا و ایستا**

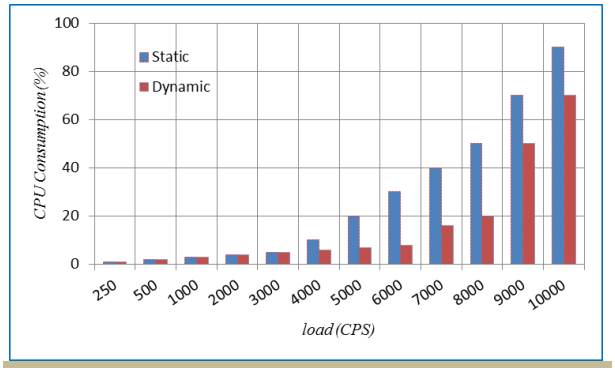
نتایج نرخ ارسال مجدد پیام‌ها نیز در شکل (۱۵) نشان داده شده است. همان گونه که می‌بینید، تا قبل از اینکه بار ورودی به ۶۰۰۰ تماس بر ثانیه برسد، نرخ ارسال مجدد در هر دو متد ایستا و پویا صفر است و به همین سبب گذردهی به‌صورت خطی صعود می‌کرد (شکل ۱۳). با این حال پس از عبورِ بار ورودی از ۶۰۰۰ تماس بر ثانیه تعداد پیام‌هایی که مجبور به ارسال مجدد می‌شوند شروع می‌شوند. سیر صعودی آن در توزیع استاتیک بسیار بیشتر از توزیع پویاست؛ زیرا در حالت توزیع پویا وضعیت پروکسی‌های SIP با توجه به شرایط شبکه و بار ورودی می‌تواند بروز گردد. همان گونه که ملاحظه می‌شود، برای مثال در بار ورودی ۱۰۰۰۰ تماس بر ثانیه، تعداد پیام‌هایی که به نتیجه نرسیده و دوباره ارسال شده‌اند، به‌ترتیب تقریبا ۱۸۷۶ و ۱۲۰۸۷ تماس بر ثانیه در متدهای پویا و ایستا بوده است. این نشان می‌دهد که در توزیع پویای حالت تا حد بسیار زیادی می‌توان از ارسال‌های مجدد زاید که باعث پر شدن صف پروکسی سرورهای SIP و در نهایت وقوع اضافه‌بار و اشباع منابع می‌شود، جلوگیری کرد. از آنجا که با وقوع اضافه‌بار در یک سرور SIP این اضافه‌بار به کل شبکه منتشر می‌شود، استفاده از توزیع پویای حالت می‌تواند در شبکه‌های بزرگ از انتشار اضافه‌بار در کل شبکه و وخامت اوضاع در طول زمان جلوگیری کند.



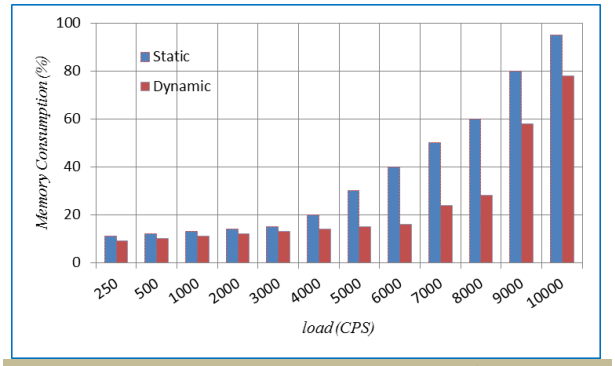

**شکل (۱۵): نرخ ارسال مجدد در دو وضعیت توزیع حالت پویا و ایستا**

در ادامه، میزان پردازنده و حافظهٔ مصرفی در دو حالت ایستا و پویا در شکل‌های (۱۶) و (۱۷) نشان داده شده است. همان گونه که ملاحظه می‌شود، میزان رشد مصرف منابع در حالت ایستا پس از بار ۶۰۰۰ تماس بر ثانیه سیر صعودی زیادی داشته است. البته توجه شود که مصرف منابع با افزایش بار

ورودی، رشد خطی می‌کند اما این رشد خطی حد آستانه‌ای دارد و پس از آن رشد نمایی می‌شود. این موضوع در بارهای ورودیِ بسیار بالا سبب اشباع منابع و افت سریع گذردهی می‌گردد. نکتهٔ قابل توجه دیگر این است که الگوی رفتاری مصرف منابع پردازنده و حافظه همانند یکدیگر است با این تفاوت که مصرف پردازنده معمولاً از مصرف حافظه کمتر است. همان طور که از شکل‌های (۱۶) و (۱۷) مشخص است تا قبل از بار ورودی ۶۰۰۰ تماس بر ثانیه، مصرف منابع در روش پویا نیز تقریباً ثابت و بسیار کم بوده است.

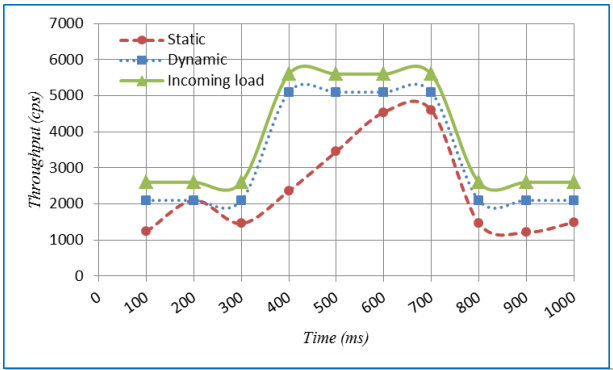



**شکل (۱۶): میانگین پردازندهٔ مصرفی در دو وضعیت توزیع حالت پویا و ایستا**



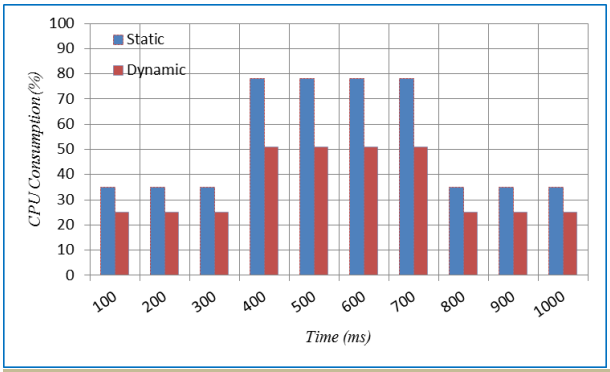

**شکل (۱۷): میانگین حافظهٔ مصرفی در دو وضعیت توزیع حالت پویا و ایستا**

در ادامه، به تست سناریویی می‌پردازیم که در آن، بار ورودی در طول زمان متغیر است. همان طور که در شکل (۱۸) نشان داده شده است، در ۳۰۰ ثانیهٔ اول بار ورودی ۲۵۰۰ تماس بر ثانیه می‌باشد. از ثانیهٔ ۳۰۰ تا ثانیهٔ ۷۰۰، بار ورودی ۵۵۰۰ تماس بر ثانیه بوده و در نهایت در ۳۰۰ ثانیهٔ آخر بار ورودی به ۲۵۰۰ تماس بر ثانیه کاهش پیدا می‌کند. همان گونه که مشاهده می‌شود در صورتی که از توزیع پویا در سیستم استفاده شود، گذردهی از الگوی بار ورودی تبعیت می‌کند، در حالی که در صورت استفاده از توزیع ایستا، گذردهی به‌سرعت نمی‌تواند نسبت به تغییرات بار ورودی عکس‌العمل نشان دهد و تقریباً گذردهی پایین‌تری را نتیجه می‌دهد.

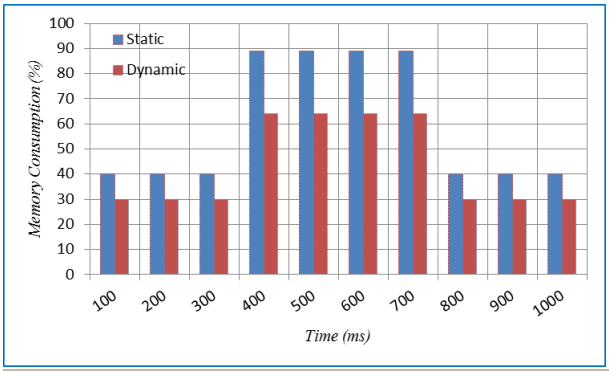



**شکل (۱۸): الگوی گذردهی در دو وضعیت توزیع حالت پویا و ایستا**

شکل (۱۹) و (۲۰) نیز الگوی میزان مصرف منابع را با توجه با تغییر بار ورودی نشان می‌دهد. همان گونه که در این شکل‌ها نیز مشاهده می‌شود، منابع سرورها با توجه به بار ورودی تغییر کرده و در پیک آن، یعنی بین فاصلهٔ زمانیِ ثانیه ۴۰۰ تا ۷۰۰، بیشترین استفاده را دارند.



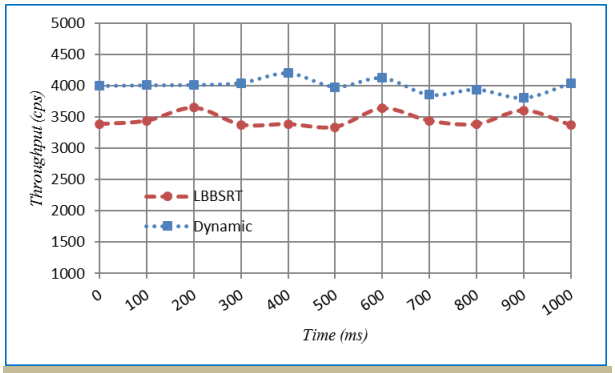

**شکل (۱۹): الگوی پردازندهٔ مصرفی در دو وضعیت توزیع حالت پویا و ایستا**



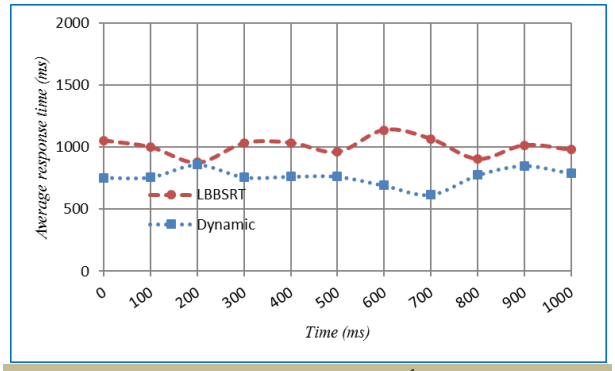

**شکل (۲۰): الگوی حافظهٔ مصرفی در دو وضعیت توزیع حالت پویا و ایستا**

در انتهای این بخش به مقایسهٔ روش توزیع حالت پویا با روش LBBSRT که در بخش ۱.۲ معرفی شد می‌پردازیم.

همان گونه که بیان شد، متد LBBSRT از یک کنترل‌کننده برای محاسبهٔ تأخیر سرورها استفاده می‌کند. بدین ترتیب برای ارسال بار از کمترین پروکسی SIP استفاده می‌کند. در این آزمایش، بار ثابتِ ۴۵۰۰ تماس بر ثانیه به مدت ۱۰۰۰ میلی‌ثانیه به توپولوژی تزریق شد. همان طور که در شکل‌های (۲۱) و (۲۲) مشاهده می‌شود، روش توزیع حالت پویا توانسته است به گذردهی بالاتر در تأخیر کمتر دست یابد. این موضوع به این دلیل است که توزیع حالت هوشمندانه‌تر کل سیستم را مورد ارزیابی قرار می‌دهد و حالت‌ها را متناسب با شرایط در لحظهٔ شبکه توزیع می‌کند. همان طور که ملاحظه می‌شود، میانگین گذردهی در روش پویا تقریباً ۴۰۰۰ تماس بر ثانیه اما در روش LBBSRT تقریباً ۳۵۰۰ تماس بر ثانیه است.

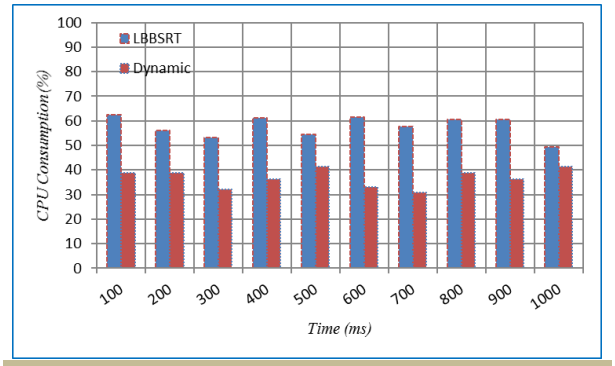



**شکل (۲۱): مقایسهٔ گذردهی برای روش توزیع حالت پویا و LBBSRT**

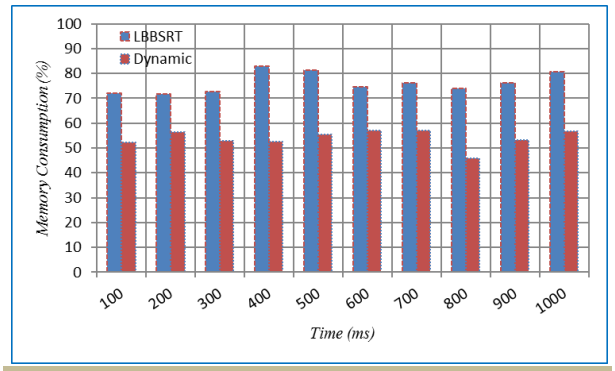



**شکل (۲۲): مقایسهٔ تأخیر برای روش توزیع حالت پویا و LBBSRT**

علت دستیابیِ روش توزیع حالت پویا به گذردهی بالاتر و تأخیر کمتر در مدیریت منابع است (شکل‌های ۱۷ و ۱۸). میانگین مصرف CPU و حافظه در روش پویا نسبت به روش LBBSRT کمتر است. میانگین مصرف CPU در متد پویا تقریباً ۴۰٪ و در روش LBBSRT تقریباً ۵۰٪ است. همچنین میانگین مصرف حافظه در روش پویا تقریباً ۵۰٪ و در روش LBBSRT تقریباً ۷۰٪ است؛ این نشان‌دهندهٔ آن است که روش توزیع حالت پویا توانسته است با مصرف منابع کمتر به گذردهی بالاتری دست یابد و به همین دلیل دیرتر از روش‌های مشابه مانند LBBSRT دچار اضافه‌بار خواهد شد.



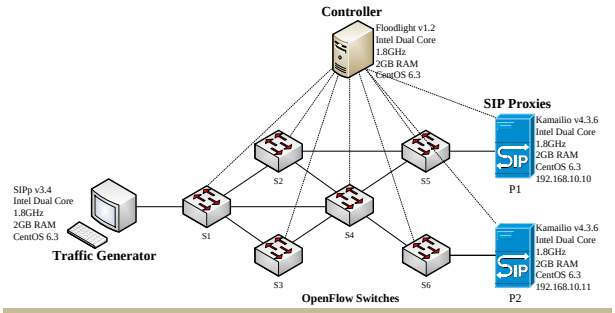

**شکل (۲۳): مقایسهٔ مصرف CPU برای روش توزیع حالت پویا و LBBSRT**




**شکل (۲۴): مقایسهٔ مصرف حافظه برای روش توزیع حالت پویا و LBBSRT**

## ۲.۸. ارزیابی سطح کنترل جهت توزیع بار

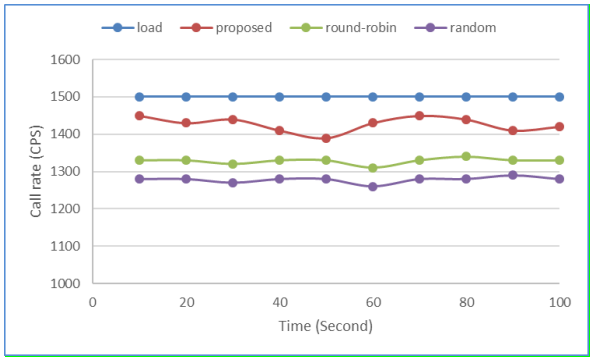
یکی از مزایای توزیع حالت بین پروکسی‌ها، توزیع بار بین آن‌هاست؛ یعنی توزیع حالت مناسب منجر به توزیع بار بین پروکسی سرورها می‌شود. در این بخش با ارزیابی کنترلر به بررسی این موضوع می‌پردازیم. برای ارزیابی عملکرد در این بخش از توپولوژی شکل (۲۵) استفاده می‌شود که شامل دو پروکسی SIP، ۶ سوئیچ OpenFlow و ۱ کنترل‌کننده است. برای پیاده‌سازی سوئیچ‌ها، Open vSwitch را بر روی ۶ ماشین‌مجازی نصب کردیم. جزئیات هریک از اجزا در این شکل نشان داده شده است. در ضمن، هر لینک شبکه دارای پهنای باند ۱۰ مگابیت بر ثانیه است.



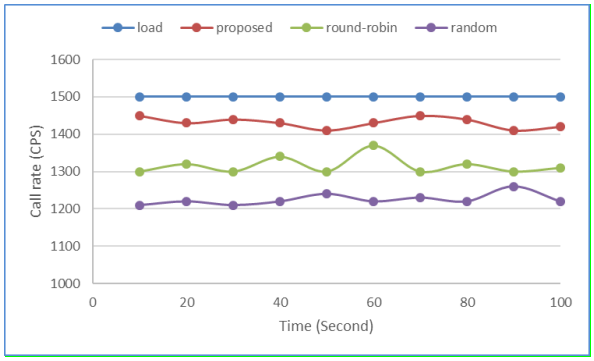

**شکل (۲۵): توپولوژی تست**

این آزمایش شامل دو سناریو با ترافیک پس‌زمینهٔ[1] متفاوت است. در سناریو ۱، ترافیک پس‌زمینهٔ هریک از پروکسی‌ها به‌طور مساوی ۵۰۰ بسته بر ثانیه است. در سناریو ۲ ترافیک پس‌زمینهٔ پروکسی ۱ (P1)، ۱۰۰۰ و پروکسی ۲ (P2)، ۵۰۰ بسته بر ثانیه است. همچنین به‌مدت ۱۰۰ ثانیه بار ارائه‌شدهٔ ثابت به میزانِ ۱۵۰۰ تماس بر ثانیه از طریق تولیدکنندهٔ ترافیک به سیستم تزریق می‌شود.

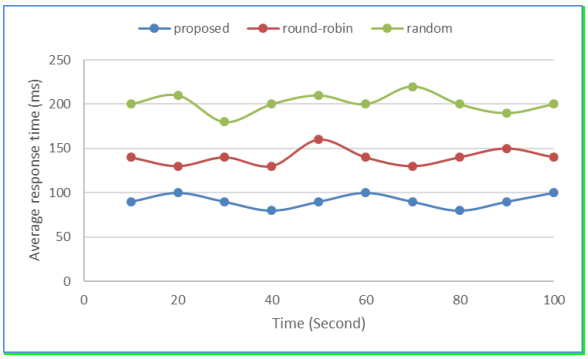
شکل (۲۶) نشان‌دهندهٔ کارایی سرورهای SIP است؛ به‌طوری که کنترل‌کننده برای انتخاب پروکسی سرور از سه متد متفاوتِ پیشنهادی، Round-robin و Random استفاده می‌کند. گذردهی پروکسی‌ها (تعداد تماس‌های سرویس داده‌شده توسط پروکسی‌ها در واحد زمان)، میانگین زمانِ پاسخ[2] آن‌ها (زمان بین ارسال INVITE از طرف User Agent تا زمان دریافت OK 200 از پروکسی SIP)، و منابع مصرفیِ پروکسی‌ها از جمله معیارهای ارزیابی هستند. هدف دستیابی به ماکزیمم گذردهی و کمترین زمان پاسخ بدون وقوع اضافه‌بار و با توجه به منابع است. متدِ پیشنهادی نسبت به متدهای Round-robin و Random در هر دو سناریو توانسته است به نتایج بهتری دست پیدا کند. به‌علاوه، نتایجِ متد پیشنهادی در هر دو سناریو مشابه یکدیگر است اما نتایجِ Raound-robin و Random در سناریو ۲ نسبت به سناریو ۱، بدتر شده است. دلیل آن این است که متفاوت بودنِ ترافیکِ پس‌زمینه پروکسی‌ها در سناریو ۲، باعث بدتر شدن توزیعِ بارِ کورکورانه این دو متد در طولِ زمان شده است. همان گونه که از مقایسهٔ شکل‌های (۲۶) مشاهده می‌شود، متد OpenSIP در هر دو سناریو توانسته است به گذردهی نزدیک به بارِ ارائه‌شده دست یابد، چراکه می‌تواند با استفاده از فیلد آمار، تخمین مناسبی از بارِ پروکسی‌ها بزند. به‌رغم مصرف منابعِ بیشتر در روش‌های Raound-robin و Random نسبت به متد پیشنهادی میانگین زمانِ پاسخ آن‌ها نیز بیشتر است. میزان مصرف منابعِ هر دو پروکسی درروش پیشنهادی تقریباً برابر است که نشان از توزیع آگاهانه و عادلانه بار توسط این روش دارد.




الف. گذردهی سرورهای SIP در سناریو ۱

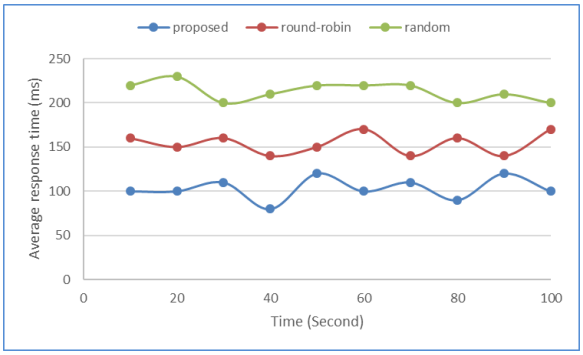



ب. گذردهی سرورهای SIP در سناریو ۲

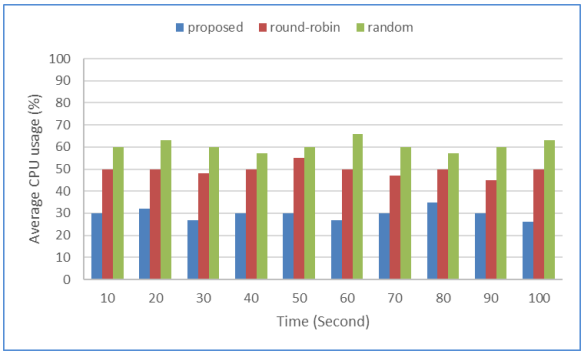



ج. میانگین زمان پاسخ سرورهای SIP در سناریو ۱

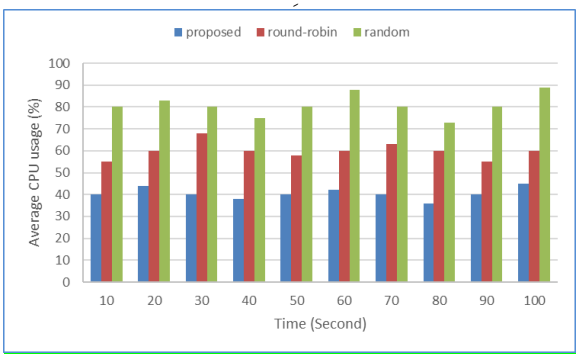
1. Background traffic
2. Call setup delay



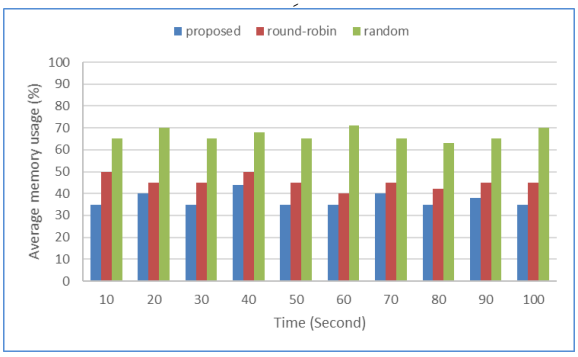

د. میانگین زمان پاسخ سرورهای SIP در سناریو ۲



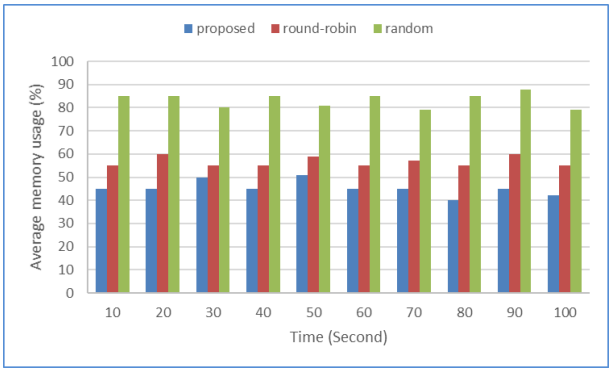

ک. میانگین پردازندهٔ مصرفیِ سرورهای SIP در سناریو ۱




گ. میانگین پردازندهٔ مصرفیِ سرورهای SIP در سناریو ۲




و. میانگین حافظهٔ مصرفیِ سرورهای SIP در سناریو ۱




هـ. میانگین حافظهٔ مصرفیِ سرورهای SIP در سناریو ۲

**شکل (۲۶): مقایسهٔ کاراییِ پروکسی سرورهای SIP در دو سناریو**

## ۹. نتیجه‌گیری

در این مقاله، به‌صورت تجربی و از طریق آزمایش، عملکرد یک سرور SIP تحت چندین سناریوی تماس ارزیابی کردیم. بر پایهٔ نتایج این مطالعهٔ عملکرد، مسئلهٔ مدیریت حالت را تعریف کردیم و برای رسیدن به یک پاسخ بهینه، یک مدل ریاضی را توسعه دادیم. در این راستا در ابتدا با استفاده از تکنولوژی شبکه‌های نرم‌افزارمحور یک چهارچوب متمرکز با یک کنترلر دارای سه ماژول طراحی کردیم. سپس به طراحی ماژول‌های مورد صحبت از جمله ماژول بهینه‌ساز پرداختیم. ما متد پیشنهادی خود را در مقایسه با الگوریتم‌های ایستایِ از پیش پیکربندی شده که امروزه وجود دارد ارزیابی کردیم و نشان دادیم که می‌توان به ۱۵ تا ۲۰٪ افزایش در بیشینهٔ گذردهی تماس دست یافت.

## منابع


[1] Widjaja I., Hilt V. and Schulzrinne H., "*Session Initiation Protocol (SIP) Overload Control*", 2008.

[2] Lindholm H., Vähäkangas T. and Raatikainen K., "*A Control Plane Benchmark for Telecommunications Signalling Applications*", In LinuxConf Europe, 2007.

[3] Gokhale S. S. and Lu J., "*Signaling performance of SIP based VoIP: A measurement-based approach*", presented at the Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE, 2005.

[4] Ohta M., "*Performance comparisons of transport protocols for session initiation protocol signaling*", presented at the Telecommunication Networking Workshop on QoS in Multiservice IP Networks, 2008.

[5] http://www.asterisk.org/

[6] Chi C., Wang D., Hao R. and Zhou W., "*Performance evaluation of SIP servers*", in: Communications and Networking in China, 2008. ChinaCom 2008. Third International Conference on, 2008.

[7] Hilt V. and Widjaja I., "*Controlling overload in networks of SIP servers*", in: Network Protocols, 2008.

[8] Shen C. and Schulzrinne H., "*On TCP-based SIP server overload control*", presented at the Principles, Systems and Applications of IP Telecommunications, 2010.

[9] Nahum E. M., Tracey J. and Wright C. P., "*Evaluating SIP server performance*", *SIGMETRICS Perform. Eval. Rev.*, vol. 35, pp. 349-350, 2007.

[10] Tian L., "*Study of SIP protocol through VoIP solution of "Asterisk"*", presented at the Global Mobile Congress (GMC), 2011.

[11] Cortes M., Ensor J. R. and Esteban J. O., "*On SIP performance*", Bell Labs Technical Journal, vol. 9, pp. 155-172, 2004.

[12] Wanke S., Scharf M., Kiesel S. and Wahl S., "*Measurement of the SIP parsing performance in the SIP express router*", Dependable and Adaptable Networks and Services, pp. 103-110, 2007.

[13] Montagna S. and Pignolo M., "*Performance evaluation of load control techniques in sip signaling servers*", presented at the Systems, 2008. ICONS 08. Third International Conference on, 2008.

[14] Homayouni M., Azhari S. V., Jahanbakhsh M., Akbari A., Mansoori A. and Amani N., "*Configuration of a sip signaling network: An experimental analysis*", in: Fifth International Joint Conference on INC, IMS and IDC, pp. 76-81, 2009.

[15] Gurbani V. K. and Jain R., "*Transport protocol considerations for session initiation protocol networks*", Bell Labs Technical Journal, vol. 9, pp. 83-97, 2004.

[16] Ram K. K., Fedeli I. C., Cox A. L. and Rixner S., "*Explaining the impact of network transport protocols on SIP proxy performance*", in: Performance Analysis of Systems and software, 2008. ISPASS 2008.

[17] Wang Y., "*SIP overload control: a backpressure-based approach*", *ACM SIGCOMM Computer Communication Review*, vol. 40, pp. 399-400, 2010.

[18] Hong Y., Huang C. and Yan J., "*Impact of Retransmission Mechanism on SIP Overload: Stability Condition and Overload Control*", Journal of Networks, vol. 7, pp. 52-62, 2012.

[19] Homayouni M., "*Controlling Overload in SIP Proxies: An Adaptive Window Based Approach Using No Explicit Feedback*", in: Global Telecommunications Conference (GLOBECOM 2010), Dec. 2010.

[20] Pesch D., Pous M. I. and Foster G., "*Performance evaluation of SIP-based multimedia services in UMTS*", *Computer Networks*, vol. 49, pp. 385-403, 2005.

[21] http://www.spirent.com/

[22] http://oprofile.sourceforge.net/news/

[23] Bharrat S. J., Asveren T. and Hart J., "Load balancing among voip server groups", U.S. Patent Application no. 12/771,618.

[24] Aggarwal S., Kaushal S. and Kumar H., "Load Balancing and Clustering Scheme for Real-Time VoIP Applications", Advances in Computer Communication and Computational Sciences. Springer, Singapore, pp. 451-461, 2019.

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

[26] Chauhan, Amita, Chauhan A., Mahajan N., Kumar H. and Kaushal S., "Framework for SIP-Based VoIP System with High Availability and Failover Capabilities: A Qualitative and Quantitative Analysis", ICT Systems and Sustainability. Springer, Singapore, pp. 273-286, 2020.

[27] Abdullah A., Basha S. M. and Sattar S. A., "A comparative study on load balancing algorithms for sip servers", Information Systems Design and Intelligent Applications. Springer, New Delhi, pp. 79-88, 2016.

[28] Abdullah A., Basha S. M. and Sattar S. A., "A novel rate based overload control method for sip servers", Emerging Research in Computing, Information, Communication and Applications. Springer, New Delhi, pp. 143-152, 2016.

[29] Mounika, G. and Murali G., "A communication aware load balancing technique for cluster file systems based on distributed hash tables (DHTs)", 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS). IEEE, 2017.

[30] Krishnamurthy R. and Rouskas G. N., "Performance evaluation of multi-core, multi-threaded SIP proxy servers (SPS)", *2016 IEEE International Conference on Communications (ICC)*. IEEE, pp. 1-6, 2016.

[31] Zhong H., Fang Y. and Cui J., "LBBSRT: An efficient SDN load balancing scheme based on server response

time", Future Generation Computer Systems, vol. 68, March 2017, pp. 183-190, 2017.

[32] Xia, Wenfeng, et al. "A survey on software-defined networking." *IEEE Communications Surveys & Tutorials* 17.1, 27-51, 2015.

[33] A. Montazerolghaem, M. H. Y. Moghaddam and S. Effati, "Implementing state distribution model in asterisk server," *7'th International Symposium on Telecommunications (IST'2014)*, Tehran, 2014, pp. 608-612, doi: 10.1109/ISTEL.2014.7000777.