مقاله پژوهشی

یک روش ترافیک‌آگاه برای دسته‌بندی بسته‌ها با هدف کاهش تعداد دفعات دسترسی به حافظه

سعید اسدروز، محمد نصیری، مهدی عباسی و حاتم عبدلی

چکیده: دسته‌بندی بسته‌ها نقش بسزایی در بهبود عملکرد تجهیزات شبکه‌ای از جمله مسیریاب‌ها، دیواره‌های آتش و سیستم‌های تشخیص نفوذ ایفا می‌کند. الگوریتم‌های دسته‌بندی بسته عموماً مبتنی بر ساختار داده‌ای ایستا هستند که الگوی رفتاری ترافیک ورودی را در بهینه‌سازی ساختار جستجو در نظر نمی‌گیرند. در این پژوهش، ویژگی‌های آماری ترافیک ورودی در نظر گرفته شده و از ساختمان داده‌های کمکی ترافیک‌آگاه در کنار ساختارهای اصلی استفاده شده است. از آنجا که حجم غالب ترافیک اینترنت، مربوط به جریان‌های بلندمدت است، برای مدت‌زمانی نه چندان کوتاه، اکثر مطابقت‌های قوانین در زیردرخت‌های مشخصی از درخت جستجو قرار دارند. برای بهره‌گیری از این ویژگی، در این پژوهش از ساختار داده درخت AVL برای نگهداری قوانین دسته‌بند و از حدهای بالا و پایین مجموعه قوانین به عنوان گره‌های درخت جستجو استفاده شده است. ارزیابی‌ها نشان می‌دهد که با افزایش چولگی بسته‌های آزمون، تعداد دفعات دسترسی به حافظه الگوریتم دسته‌بندی ترافیک‌آگاه نسبت به الگوریتم دسته‌بندی پایه کاهش قابل توجهی دارد. بر اساس ارزیابی‌ها، دسته‌بندی بسته ترافیک‌آگاه با استفاده از قوانین پرتکرار می‌تواند میانگین کل تعداد دفعات دسترسی به حافظه و در نتیجه زمان جستجو را بیش از ۴۰ درصد کاهش دهد.

کلیدواژه: **ترافیک‌آگاه، درخت AVL، دسته‌بندی بسته‌ها، محلی‌بودن مراجعات.**

۱- مقدمه

امروزه با توجه به ازدیاد حجم ترافیک موجود در شبکه و افزایش تعداد کاربران اینترنت و نیز مطرح‌شدن سرویس‌های اینترنتی مختلف، نیاز به شبکه‌هایی با کارایی بالاتر افزایش یافته است. برای رسیدن به کارایی مناسب در شبکه، ارائه راهکارهایی برای افزایش سرعت تصمیم‌گیری در مورد بسته‌های دریافتی در ابزارهای پردازشگر شبکه مانند مسیریاب‌ها[1] و دیواره‌های آتش[2] امری ضروری است. به این منظور، ابزارهای پردازشگر شبکه از الگوریتم‌های دسته‌بندی بسته‌ها[3] برای تفکیک ترافیک اینترنتی ورودی و ارائه سرویس‌های مختلف استفاده می‌کنند.

دسته‌بندی بسته‌ها تعیین می‌کند که هر بسته به چه جریانی تعلق دارد و مبتنی بر یک یا چند فیلد سرآیند بسته است. فیلدهای سرآیند بسته می‌تواند شامل آدرس IP مبدأ[4]، آدرس IP مقصد[5]، نوع پروتکل[6]، شماره درگاه مبدأ[7] و شماره درگاه مقصد[8] باشد [۱]. در فرایند دسته‌بندی با دریافت هر بسته، جدول قوانین برای یافتن مناسب‌ترین قانون منطبق با بسته دریافتی جستجو می‌شود. هر قانون علاوه بر محدوده‌های مجاز برای هر فیلد سرآیند، عملی[9] را که باید روی بسته‌های منطبق‌شده اعمال شود، مشخص می‌کند. در دسته‌بندی بسته‌ها، اقدام متناظر با قانون انطباق‌یافته با بسته ورودی به آن اعمال می‌شود. از آنجا که ممکن است یک بسته با مجموعه‌ای از قوانین منطبق شود، اقدام مرتبط با بااولویت‌ترین قانون برای آن بسته اعمال می‌شود [۱].

الگوریتم‌های دسته‌بندی ایستا فقط از ویژگی‌های مجموعه قوانین دسته‌بند برای کاهش پیچیدگی الگوریتم جستجو استفاده می‌کنند و الگوی ترافیک ورودی را برای بهینه‌سازی ساختار داده جستجو در نظر نمی‌گیرند. الگوریتم‌های دسته‌بندی پویای بسته یا ترافیک‌آگاه، ساختار داده داخلی درخت را به صورت پویا با الگوی ترافیک ورودی وفق می‌دهند. این الگوریتم‌ها با در نظر گرفتن ویژگی‌های ترافیک ورودی از جمله محلی‌بودن مراجعات[10] به بهینه‌سازی ساختار داده داخلی خود می‌پردازند. بهینه‌سازی ساختار داده داخلی به گونه‌ای است که موجب بهبود کارایی الگوریتم دسته‌بندی در حالت متوسط[11] می‌شود.

ترافیک اینترنتی دارای ویژگی‌هایی است که می‌توان از آنها برای بهبود عملکرد الگوریتم‌های دسته‌بندی استفاده کرد. مطالعات قبلی [۲] و [۳] نشان می‌دهند که بعضی از پارامترهای جریان‌ها (مانند نرخ ارسال و اندازه[12]) به هم مرتبط[13] می‌باشند و با این که اکثریت جریان‌ها را جریان‌های کوچک تشکیل می‌دهند ولی حجم غالب ترافیک اینترنت (به بایت) متعلق به تعداد کمی جریان‌های بزرگ است. محلی‌بودن موقتی مراجعات بیان می‌کند که در آینده نزدیک به جریان‌های اخیراً مشاهده‌شده، ارجاع صورت خواهد گرفت. این خاصیت به شکل بارزی در ترافیک

این مقاله در تاریخ ۱۳ بهمن ماه ۱۳۹۸ دریافت و در تاریخ ۳۰ تیر ماه ۱۳۹۹ بازنگری شد.

سعید اسدروز، دانشکده مهندسی، دانشگاه بوعلی سینا، همدان، ایران، (email: s.asadrooz92@basu.ac.ir).
محمد نصیری، دانشکده مهندسی، دانشگاه بوعلی سینا، همدان، ایران، (email: m.nassiri@basu.ac.ir).
مهدی عباسی، دانشکده مهندسی، دانشگاه بوعلی سینا، همدان، ایران، (email: abbasi@basu.ac.ir).
حاتم عبدلی (نویسنده مسئول)، دانشکده مهندسی، دانشگاه بوعلی سینا، همدان، ایران، (email: abdoli@basu.ac.ir).

1. Router
2. Firewall
3. Packet Classification
4. Source IP
5. Destination IP
6. Protocol
7. Source Port
8. Destination Port
9. Action
10. Locality of Reference
11. Average Case
12. Size
13. Correlate

اینترنت، دیده شده است [۴].

در درختان جستجو، این ویژگی ترافیک اینترنت (تعداد زیادی جریان کوچک در مقابل تعداد کمی جریان بزرگ) باعث نامتوازن‌بودن الگوی پیمایش درخت‌ها شده است. به بیان دیگر، معمولاً اکثر جستجوها در زیردرختان مشخصی پایان می‌یابد. بنابراین تعداد کمی از برگ‌ها دارای تعداد برخورد زیادی می‌باشند، یعنی مسیرهای کمی به صورت فراوان جستجو می‌شوند. همچنین نشان داده شده که ۹۰٪ ترافیک در کمتر از ۱٪ مسیرهای جستجو قرار گرفته است [۵]. چند تا از مهم‌ترین ویژگی‌های ترافیک اینترنت را می‌توان به صورت زیر بیان کرد [۶] و [۷]:

- **چولگی[1] اندازه جریان:** اندازه جریان به معنای تعداد بسته‌هایی است که آن جریان را تشکیل می‌دهند. حدود ۲۰ درصد جریان‌ها دارای ۱۰ بسته یا بیشتر می‌باشند که حدود ۷۰ درصد از کل ترافیک را حمل می‌کنند، یعنی اکثر ترافیک اینترنت از تعداد کمی جریان‌های سنگین‌وزن تشکیل یافته است. پس برای کاهش زمان جستجو و ارسال کل بسته‌های ترافیک، مطلوب است که تعداد مقایسه‌ها برای جریان‌های سنگین‌وزن کاهش یابد.
- **چولگی مدت جریان:** مدت جریان به عنوان زمان سپری‌شده بین رسیدن اولین و آخرین بسته تعریف می‌شود. حدود ۲۰ درصد جریان‌ها پنج ثانیه یا بیشتر طول می‌کشند و حدود ۶۰ درصد ترافیک را حمل می‌کنند، یعنی اکثر ترافیک اینترنت به وسیله تعداد کمی از جریان‌های طولانی‌مدت تشکیل شده است [۶] و [۷].

به طور کلی همه بسته‌های یک جریان با قانون یکسانی منطبق می‌شوند. بنابراین این مشاهدات نشان می‌دهد که تعداد کمی از قوانین برای دسته‌بندی حجم غالب ترافیک اینترنت در مدت زمان قابل توجهی استفاده می‌شوند. به این ویژگی، محلی‌بودن انطباق جریان‌ها در دسته‌بندی بسته‌ها گفته می‌شود. همچنین گزارش شده که حدود ۹۰٪ بسته‌ها فقط با ۲۵٪ قوانین منطبق می‌شوند [۸].

ویژگی‌های بیان‌شده نشان می‌دهند که در نظر گرفتن الگوی ترافیک ورودی و انجام بهینه‌سازی درخت جستجوی مبتنی بر فراوانی بسته‌ها می‌تواند برای بهبود کارایی دسته‌بندی و کاهش زمان جستجوی الگوریتم مفید باشد.

ساختار مقاله در ادامه به این صورت است که در بخش دوم کارهای انجام‌شده و مرتبط با دسته‌بندی ترافیک‌آگاه بررسی می‌شود. در بخش سوم روش پیشنهادی برای دسته‌بندی بسته‌ها با استفاده از درخت AVL[2] آورده شده و بخش چهارم شامل جزئیات دو سناریوی پیاده‌سازی شده می‌باشد. در بخش پنجم ارزیابی‌های مربوط به روش پیشنهادی آورده شده و بخش ششم به نتیجه‌گیری و کارهای آتی اختصاص دارد.

۲- کارهای انجام‌شده

الگوریتم‌های دسته‌بندی بسته‌ها به دو نوع کلی الگوریتم‌های مبتنی بر سخت‌افزار و الگوریتم‌های مبتنی بر نرم‌افزار تفکیک می‌شوند. الگوریتم‌های سخت‌افزاری مانند الگوریتم‌های مبتنی بر TCAM [۹] و [۱۰] بر اساس سخت‌افزار ساخته می‌شوند. روش‌های سخت‌افزاری با وجود داشتن سرعت دسته‌بندی بالا، از انعطاف‌پذیری کافی برای پشتیبانی از تعداد زیاد قوانین و فیلدهای دسته‌بندی برخوردار نیستند. راه حل‌های مبتنی بر نرم‌افزار طیف وسیعی از پژوهش‌های انجام‌شده را در بر می‌گیرند که با وجود داشتن سرعت پردازش قابل قبول، از نظر انطاف‌پذیری، توان مصرفی و هزینه، غالباً از شرایط بهتری نسبت به روش‌های سخت‌افزاری برخوردارند [۱۱] تا [۱۹].

در بین الگوریتم‌های نرم‌افزاری برای دسته‌بندی بسته‌ها، الگوریتم‌های درخت تصمیم‌گیری به روشی هندسی و با تجزیه بازگشتی فضای قوانین و ایجاد درخت تصمیم مسأله را حل می‌کنند [۱۱]، [۱۳] و [۱۶]. در این الگوریتم‌ها از رهیافت‌هایی استفاده می‌شود تا بین سرعت جستجو و حافظه مصرفی یک مبادله و توازن ایجاد شود. هرچند در الگوریتم‌های فعلی مبتنی بر درخت تصمیم، از ویژگی‌های پویای ترافیک شبکه در الگوریتم استفاده نشده است. برخی از الگوریتم‌های مبتنی بر درخت تصمیم‌گیری، بر اساس برش در سطح بیت قوانین جدول دسته‌بندی طراحی شده‌اند که توانسته‌اند بهبود قابل توجهی از نظر حافظه مصرفی ایجاد کنند [۱۱] و [۱۲].

Bergamini و همکاران، شبکه‌ای از میانبرها را برای تقویت درخت تصمیم‌گیری استفاده کرده‌اند [۵]. در این روش برای افزایش سرعت جستجو در مسیرهای مکرراً پیموده‌شده، از وفق‌دادن درخت جستجو با الگوی ترافیک ورودی استفاده شده است. این روش از سابقه آماری گره‌های درخت جستجو استفاده کرده و به صورت دوره‌ای موقعیت میانبرها را تنظیم می‌کند. در این روش، سعی شده که با استفاده از میانبرها، عمق درخت جستجو کم شود و در نتیجه تعداد دسترسی به حافظه کاهش یابد. در این مقاله، تعیین موقعیت قرارگیری میانبرها برای افزایش بهره و چگونگی به روز رسانی موقعیت‌ها به صورت حل‌نشده باقی مانده و مورد بحث قرار نگرفته است.

Hamed و همکاران روش رد اولیه[3] برای بیشینه[4]کردن رد اولیه جریان‌های مخرب جهت محافظت در برابر حملات DoS ارائه داده‌اند [۲۰]. همچنین از درخت باینری الفبایی با استفاده از الگوی ترافیک ورودی، برای کاهش زمان انطباق بسته‌ها استفاده شده است. یکی از ویژگی‌های ترافیکی مهم، چولگی انطباق ترافیک است که از این ویژگی برای ساخت درخت جستجوی الفبایی با پیچیدگی زمانی $O(n \log n)$ و پیچیدگی فضایی $O(dn)$ استفاده می‌شود، به طوری که n نماد تعداد قوانین و d نماد تعداد فیلد قوانین است. این روش دارای این محدودیت است که سربار به روز رسانی درخت جستجو می‌تواند به طور قابل توجهی زیاد باشد.

Acharya روش TFO[5] را برای وفق‌دادن کارایی دیواره‌های آتش با تغییرات پویای مشخصات ترافیک شبکه ارائه داده و تمرکز آنها بر روی دیواره‌های آتش مبتنی بر لیست[6] است [۲۱]. در این دیواره‌های آتش، قوانین به صورت یک لیست اولویت می‌باشند و اولویت هر قانون به رتبه و موقعیت آن در لیست اشاره دارد. قوانینی که اخیراً انطباق یافته‌اند رتبه بالاتری نسبت به بقیه دارند. اولین انطباق، مشخص‌کننده اقدام تعیین‌شده به وسیله دیواره آتش برای آن بسته ورودی است. این روش دارای این محدودیت است که قوانین با فراوانی انطباق بالا به صورت آفلاین ایجاد می‌شوند. همچنین دارای پیچیدگی زمانی $2n$ و پیچیدگی فضایی n است به طوری که n نماد تعداد قوانین است.

روشی به نام DRO[7] برای بهینه‌سازی دیواره آتش ارائه شده است [۷]

1. Skewness
2. Georgy Adelson-Velsky and Evgenii Landis
3. Early Rejection
4. Maximum
5. Traffic Aware Firewall Optimization
6. List-Based Firewall
7. Dynamic Rule-Ordering Optimization

که این روش از مشخصات ترافیک اینترنت برای محاسبه پویای آمارها و بهینه‌کردن ترتیب قوانین دسته‌بندی بسته‌ها استفاده می‌کند. روش ارائه‌شده، از چولگی انطباق قوانین برای بهترکردن کارایی دیواره آتش استفاده می‌کند. به هر قانون یک وزن داده می‌شود به طوری که متناسب با فراوانی انطباق و تازگی انطباق آن قانون می‌باشد. با توجه به وزن قوانین، یک درخت بیشینه هرم[1] از قوانین با پیچیدگی فضایی $O(n)$ و پیچیدگی زمانی $O(n^2)$ ایجاد می‌گردد. این درخت به صورت دوره‌ای، بازسازی شده و قوانین دارای وزن بیشتر در بالای درخت قرار می‌گیرند. با نزدیک‌شدن به ریشه درخت، قوانین با تعداد دسترسی‌های حافظه کمتری قابل دستیابی می‌باشند و در نتیجه سرعت جستجو افزایش پیدا خواهد کرد. این روش برای سیاست‌هایی با تعداد قوانین همپوشان زیاد مناسب نیست.

El-Atawy و همکاران فضای ترافیک را قطعه‌بندی کرده و برای قطعه‌های به وجود آمده، یک درخت هافمن[2] ایجاد می‌کنند که روش‌های STS[3] و SLS[4] را برای بهینه‌کردن ساختار درخت جستجو ارائه داده‌اند [۶]. این روش‌ها از ویژگی‌های ترافیک شبکه استفاده می‌کنند تا با استفاده از روش‌های تطبیقی، به کارایی بالایی دست یابند. پیچیدگی زمانی و فضایی هر دو روش به ترتیب برابر با $O(ns)$ و s است به طوری که n نماد تعداد قوانین و s نماد تعداد قطعه‌ها است.

Acharya و همکاران روشی را تحت عنوان OPTWALL ارائه داده‌اند که چارچوبی برای بهینه‌سازی سلسه‌مراتبی تطبیقی دیواره آتش[5] است [۲۲]. ویژگی‌های اصلی این روش، طراحی سلسله‌مراتبی، شیوه‌های تقسیم[6]، مکانیسم وفق‌دهی آنلاین ترافیک و طرحی برای مقابله با حملات حملات DoS است. در این تحقیق از یک راهکار بهینه‌سازی سلسه‌مراتبی برای ساختن زیرمجموعه سیاست‌های وزن- متعادل[7] استفاده استفاده شده است. چالش اصلی این کار، نگهداری جامعیت معنایی مجموعه سیاست‌ها در هر سطح سلسه‌مراتب است.

Shaikot و همکاران دسته‌بند ترافیک‌آگاه npf را ارائه داده‌اند [۲۳]. دسته‌بندی npf، ساختار داده داخلی را به صورت پویا و مبتنی بر الگوی ترافیک ورودی، دوباره بازسازی می‌کند. این روش از محلی‌بودن مراجعات بسته‌ها استفاده می‌کند تا زمان جستجو برای قوانین پربازدید کاهش یابد. در این روش برای هر کدام از فیلدهای آدرس مبدأ، آدرس مقصد، درگاه مبدأ و درگاه مقصد یک درخت قانون مبتنی بر بازه (IBT)[8] ایجاد می‌شود. برای ترافیک‌آگاه کردن npf، در گره‌های درخت از ویژگی محبوبیت[9] استفاده می‌شود. اگر هر کدام از بازه‌های آدرس‌ها یا شماره درگاه‌ها به طور فراوان در دسته‌بندی استفاده شود، آن گاه گره مربوط به آن محدوده یا بازه به سمت ریشه درخت انتقال داده می‌شود.

Shaikot و همکاران ادامه کار قبلی خود [۲۳] را ارائه داده‌اند که برای پنج فیلد آدرس مبدأ، آدرس مقصد، درگاه مبدأ، درگاه مقصد و پروتکل یک IBT ایجاد می‌کنند [۲۴]. نحوه بازسازی پویای ساختار جستجو مبتنی بر الگوی ترافیک ورودی با استفاده از چرخش[10] تشریح شده است. ارزیابی‌ها نشان می‌دهد که باید بین کاهش میانگین زمان جستجو در دسته‌بند و سربار ناشی از بازسازی پویای درخت با الگوی ترافیک، مصالحه‌ای صورت گیرد.

روشی برای تحلیل ترافیک شبکه و ترتیب‌دادن قوانین دیواره آتش بر اساس ویژگی‌های محلی و موقت ترافیک به صورت پویا توسط Sairam و همکاران ارائه شده است [۲۵]. این روش شامل دو مرحله رد اولیه و بهینه‌سازی انطباق بسته‌ها است. ابتدا قوانین قطعه‌بندی می‌شوند و سپس تبدیل به عبارت دودویی[11] شده و از روی آنها یک درخت هافمن ایجاد می‌شود. در مرحله رد اولیه، از درخت LCP[12] استفاده می‌شود که می‌تواند در زمان $O(mn)$ و با حافظه‌ای از مرتبه $O(n)$ ساخته شود. n نماد تعداد قوانین لیست سیاه و m برابر با ۳۲ نماد تعداد بیت‌های آدرس IP است.

Dong و همکاران یک روش رهیافتی را بر اساس آنتروپی اطلاعات پیشنهاد داده‌اند تا یک درخت تصمیم‌گیری را با یک حافظه حداقلی ایجاد کنند [۱۶]. هرچند این روش‌ها به دلیل جستجوی مستقلی که بر روی فیلدهای دسته‌بندی انجام می‌دهند، بیشتر برای مجموعه قوانین با تعداد محدود قابل استفاده است [۱۸].

نویسندگان در پژوهشی دیگر [۱۷]، یک الگوریتم دسته‌بندی را با به کارگیری یک درخت توسعه‌یافته (X-tree) و تجمیع آن با یک ساختمان داده احتمالی به نام فیلتر Cuckoo ارائه کرده‌اند. بنا بر نتایج این تحقیق، این روش دارای سرعت جستجو و کارایی خوبی است. با توجه به این که X-tree در این الگوریتم به حافظه زیادی احتیاج دارد، فیلتر احتمالی Cuckoo با هدف کاهش نیازمندی‌های حافظه با این روش تلفیق شده است که البته نقطه ضعف کاهش دقت را به دنبال دارد [۱۷].

الگوریتم دیگری به نام CuckooFlow با استفاده از فیلتر Cuckoo و بر اساس ویژگی محلیت ترافیک شبکه و استفاده از حافظه نهان، برای افزایش سرعت دسته‌بندی شبکه‌های مبتنی بر نرم‌افزار (SDN) ارائه شده است [۱۹]. با این حال تعداد قوانین استفاده‌شده در ارزیابی مشخص نیست و الگوریتم ارائه‌شده دارای پیچیدگی زیادی است. زمان جستجو یا تعداد دسترسی به حافظه هم در این کار گزارش نشده است.

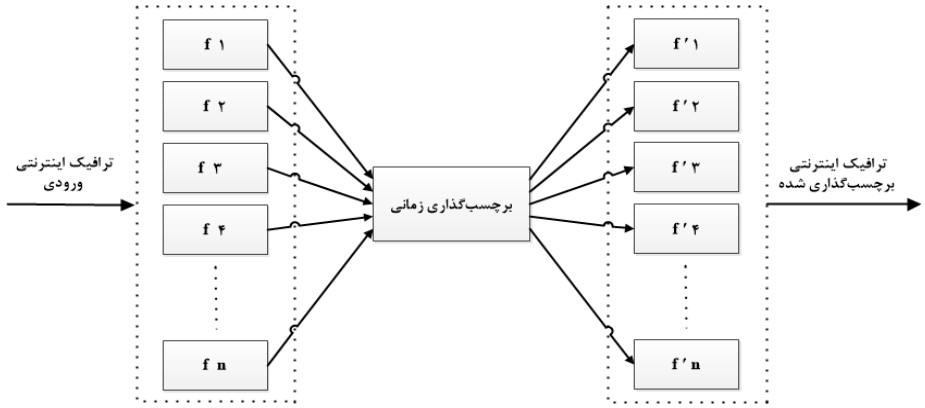
۳- روش پیشنهادی

پس از بررسی کارهای پیشین و با در نظر گرفتن الگوی ترافیک ورودی و انجام بهینه‌سازی درخت جستجوی مبتنی بر فراوانی بسته‌ها، روشی برای بهبود کارایی دسته‌بندی بسته‌ها ارائه شده که شامل سه بخش است و در ادامه شرح داده خواهد شد.

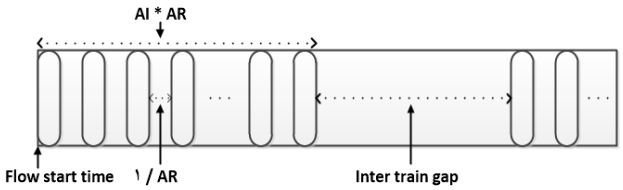
۳-۱ برچسب‌گذاری زمانی و ایجاد جریان در بسته‌های تولیدشده توسط ClassBench

با استفاده از پارامتر محلی‌بودن[13] ابزار ClassBench [۲۶]، می‌توان جریان‌هایی با تعداد بسته‌های مختلف ایجاد کرد. این پارامتر را می‌توان بین ۰ تا ۱ تنظیم کرد. هرچه مقدار این پارامتر به عدد ۱ نزدیک‌تر باشد، جریان‌های حجیم‌تری تولید خواهند شد. در بسته‌های تولیدشده توسط

1. Max Heap
2. Huffman
3. Segment-Based Tree Search
4. Segment-Based List Search
5. Adaptive Hierarchical Firewall Optimization Framework
6. Splitting Techniques
7. Load-Balanced Policy Subset
8. Interval-Based Rule Tree
9. Popularity
10. Rotation
11. Boolean
12. Longest Common Prefix
13. Locality



شکل ۱: روند کلی برچسب‌گذاری زمانی بسته‌ها.



شکل ۲: بسته‌های یک جریان داده‌ای.

ابزار ClassBench، بسته‌های هر جریان بدون لحاظ زمان ورود[1] و به صورت پشت سر هم قرار داده می‌شوند. مثلاً با پارامتر محلی‌بودن برابر با ۱، اگر برای یک جریان ۲۰۰۰ بسته تولید شود آن گاه این ۲۰۰۰ بسته متعلق به آن جریان به صورت متوالی در بسته‌های تولیدشده قرار خواهند گرفت. با این حال، در محیط واقعی اینچنین نیست و در هر لحظه، بسته‌های جریان‌های مختلف به ابزارهای پردازشگر شبکه خواهند رسید. در اینجا سعی بر آن شده تا به بسته‌های تولیدشده توسط ابزار ClassBench، برچسب‌های زمانی مجازی اضافه شود. با این کار، بسته‌های تولیدشده به جریان‌های واقعی شباهت بیشتری پیدا خواهند کرد و می‌تواند برای آزمودن و تحلیل کارایی الگوریتم‌های ترافیک‌آگاه استفاده شود.

در شکل ۱، روند کلی برچسب‌گذاری بسته‌ها نشان داده شده است. ابتدا جریان‌های مختلف ترافیک اینترنتی ورودی از هم متمایز می‌شوند و سپس عملیات برچسب‌گذاری زمانی روی بسته‌های متعلق به جریان‌های مختلف انجام می‌شود. در این شکل، جریان‌های ورودی با نماد f و جریان‌های برچسب‌گذاری شده با نماد f' نشان داده شده است.

تولید برچسب زمانی و اختصاص آنها به بسته‌ها، برای هر جریان به صورت جداگانه انجام می‌شود. در این فرایند به جای استفاده از زمان واقعی از یک رویکرد ساعت مجازی مبتنی بر بسته استفاده شده است [۲۷] تا [۲۹].

برای هر جریان از متغیرهای FST[2]، AR[3]، AI[4] و $Inter\_train\_gap$ استفاده می‌شود. از متغیر FST برای نشان‌دادن زمان ورود اولین بسته جریان مربوط استفاده می‌شود. برای هر جریان، طبق توزیع یکنواخت، مقداری بین صفر و FST_high به آن اختصاص داده می‌شود [۲۷] و [۲۸]. از متغیر AR برای نگهداری میانگین نرخ ارسال هر جریان داده‌ای استفاده می‌شود. در ارزیابی‌های انجام‌شده برای هر جریان، طبق توزیع یکنواخت، مقداری بین ۱۰ و ۱۵ اختصاص داده می‌شود. به بیان دیگر به هر جریان داده‌ای، یک ساعت مجازی اختصاص داده می‌شود که با رسیدن هر بسته، به اندازه $1/AR$ افزایش می‌یابد. هر بار، مقدار این ساعت مجازی به عنوان برچسب زمانی به بسته‌ها اضافه می‌شود.

هر جریان داده‌ای ممکن است در چندین دوره زمانی تکرار شود. متغیر AI طول این دوره‌های زمانی را نگهداری می‌کند و هر بار طبق توزیع هندسی، مقداری بین ۵ تا ۳۵ به آن اختصاص می‌یابد.

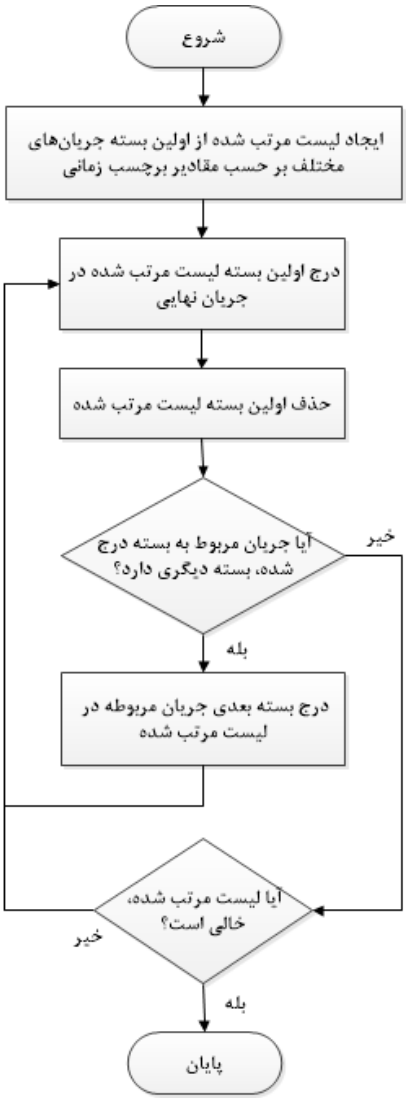
ساعت مجازی برای هر جریان داده‌ای، بعد از تعداد $AI \times AR$ بسته، به اندازه متغیر $Inter\_train\_gap$ افزایش می‌یابد. هر بار طبق توزیع نمایی، مقداری بین ۱ و ۱۰ به متغیر $Inter\_train\_gap$ اختصاص می‌یابد. اغلب از توزیع نمایی برای مدل‌سازی زمان بین رخدادهای مستقل که با نرخ میانگین ثابت رخ می‌دهند استفاده می‌شود. در شکل ۲، یک جریان به همراه بسته‌های آن نشان داده شده و همچنین متغیرهای مربوط و مفهوم آنها نیز در شکل آمده است.

در این مرحله، بسته‌های جریان‌های مختلف به ترتیب صعودی برچسب زمانی، در یک جریان مجتمع‌شده قرار می‌گیرند. فرایند مجتمع‌سازی به این صورت است که ابتدا لیست مرتب‌شده‌ای از اولین بسته جریان‌های مختلف ایجاد می‌شود. سپس اولین بسته لیست مرتب‌شده یعنی بسته با کوچک‌ترین مقدار برچسب زمانی از لیست حذف و در جریان مجتمع‌شده درج می‌شود. آن گاه بسته بعدی آن جریان در لیست مرتب‌شده درج می‌شود. سپس دوباره بسته با کوچک‌ترین مقدار برچسب زمانی در جریان مجتمع‌شده درج می‌شود. این روند ادامه می‌یابد تا زمانی که همه بسته‌های جریان‌های مختلف در جریان مجتمع‌شده درج شده باشند. شکل ۳ روندنمای مربوط به چگونگی مجتمع‌سازی جریان‌های مختلف را نشان می‌دهد.

تجهیزات پردازشگر شبکه، تعداد بسته‌های ورودی محدودی را در یک لحظه زمانی می‌توانند به خروجی بفرستند. اگر در یک لحظه زمانی، تعداد بسته‌های بیشتری نسبت به ظرفیت موجود وارد شوند، آن گاه بسته‌های مازاد بر ظرفیت را باید بر اساس اولویت میانگیری[5] کرد. بسته‌های ورودی ورودی میانگیرشده، در لحظه‌های زمانی بعدی که تعداد بسته کمتری نسبت به ظرفیت خروجی وارد شود به ترتیب اولویت به خروجی ارسال می‌شوند.

در این پژوهش، بسته با فیلد شماره بسته پایین‌تر، دارای اولویت بالاتری است.

1. Arrival Time
2. Flow Start Time
3. Average Transmission Rate
4. Average Interval
5. Buffer



شکل ۳: روندنمای مجتمع‌سازی در الگوریتم پیشنهادی.

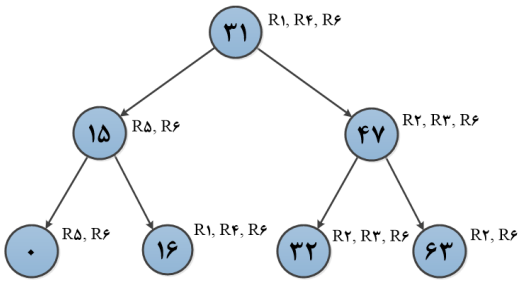
۳-۲ الگوریتم دسته‌بندی پایه

الگوریتم دسته‌بندی پایه، الگوریتم دسته‌بندی اولیه‌ای است که قوانین دسته‌بند را با استفاده از درخت AVL نگهداری می‌کند. در این الگوریتم از حدهای پایین و بالای قوانین به عنوان گره‌های درخت جستجو استفاده می‌شود. در این پژوهش، ابتدا این الگوریتم جدید دسته‌بندی استفاده‌کننده از درخت AVL ارائه شده است. سپس روش ترافیک‌آگاه دسته‌بندی بسته مبتنی بر این الگوریتم دسته‌بندی پایه برای کاهش تعداد دسترسی به حافظه و تسریع عملیات دسته‌بندی بسته‌ها ارائه شده است.

۳-۲-۱ درخت AVL

درخت AVL، یک نوع درخت جستجوی دودویی[1] خودمتوازن‌کننده[2] است و دارای این ویژگی است که اختلاف ارتفاع زیردرخت چپ و راست هر گره حداکثر یک است. بنابراین برای حفظ این ویژگی، ممکن است طی عملیات درج و حذف با انجام چرخش‌هایی[3] یک یا چند بار متوازن‌سازی صورت گیرد. این ویژگی تضمین می‌کند که ارتفاع درخت AVL هرگز از $1/44 \log n$ بیشتر نخواهد شد [۳۰]. اگر n تعداد گره‌های درخت باشد، آن گاه عملیات درج، حذف و جستجو در یک درخت جستجوی دودویی در بدترین حالت با پیچیدگی $O(n)$ انجام خواهد شد، در حالی که درخت جستجوی AVL این عملیات را در بدترین و بهترین حالت با پیچیدگی $O(\log n)$ انجام می‌دهد.

1. Binary Search Tree
2. Self Adjusting
3. Rotations



شکل ۴: درخت جستجوی AVL ایجادشده با حدهای پایین و بالای قوانین جدول ۱.

جدول ۱: نمونه‌ای از آدرس‌های پیشوندی مبدأ و حدود پایین و بالا [۳۱].

قوانین	آدرس پیشوندی مبدأ	حد پایین	حد بالا	دهدهی حد پایین	دهدهی حد بالا
R۱	۰۱*	۰۱۰۰۰۰	۰۱۱۱۱۱	۱۶	۳۱
R۲	۱*	۱۰۰۰۰۰	۱۱۱۱۱۱	۳۲	۶۳
R۳	۱۰*	۱۰۰۰۰۰	۱۰۱۱۱۱	۳۲	۴۷
R۴	۰۱*	۰۱۰۰۰۰	۰۱۱۱۱۱	۱۶	۳۱
R۵	۰۰*	۰۰۰۰۰۰	۰۰۱۱۱۱	۰	۱۵
R۶	*	۰۰۰۰۰۰	۱۱۱۱۱۱	۰	۶۳

۳-۲-۲ ایجاد درخت جستجوی AVL

در این پژوهش، از حدهای پایین و بالای مجموعه قوانین برای مقادیر گره‌های درخت جستجوی AVL استفاده می‌شود. حدهای پایین و بالای هر قانون محاسبه و به اعداد دهدهی تبدیل می‌شوند [۳۱]. جدول ۱، نمونه‌ای از آدرس‌های پیشوندی مبدأ و حدهای پایین و بالای آنها را برای شش قانون نشان می‌دهد. در این جدول برای سادگی از آدرس‌های مبدأ شش‌بیتی استفاده شده است. در گام بعدی، اعداد دهدهی به دست آمده برای حدهای پایین و بالای هر قانون به ترتیب در درخت جستجوی AVL درج خواهند شد.

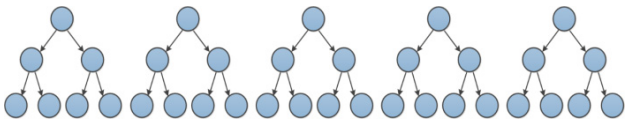
ابتدا اعداد دهدهی به دست آمده از حدهای پایین و بالای مجموعه قوانین در درخت جستجوی AVL درج می‌شود. سپس قوانین مربوط به هر گره درخت جستجو به آن اختصاص داده می‌شود. به این منظور، برای هر گره به طور جداگانه روی همه قوانین جستجوی خطی انجام می‌شود. نتیجه جستجو برای هر گره شامل قوانینی است که مقدار آن گره در محدوده حد پایین و بالای آنها است. سپس به هر گره، قوانین مربوط اختصاص می‌یابد. برای مثال در شکل ۴، درخت جستجوی AVL ایجادشده با حدهای پایین و بالای قوانین جدول ۱ آورده شده است.

در هر گره برای نگهداری قوانین اختصاص‌یافته، از ساختار داده مجموعه بیتی[4] یا آرایه بیتی[5] استفاده می‌شود. اندازه مجموعه بیتی مربوط مربوط به هر گره، به اندازه تعداد کل قوانین است، یعنی به ازای هر قانون از یک بیت استفاده می‌شود. به همین جهت، استفاده از مجموعه بیتی برای نگهداری قوانین مربوط به هر گره، موجب کاهش چشم‌گیری در مصرف حافظه خواهد شد.

4. Bitset
5. Bit Array

R۱	R۲	R۳	R۴	R۵	R۶
۱	۰	۰	۱	۰	۱

شکل ۵: داده ساختار مجموعه‌بیتی مربوط به گره ۳۱.



شکل ۶: درخت‌های پنج‌گانه جستجوی AVL.

برای هر گره، پس از انجام جستجوی خطی روی همه قوانین و یافتن قوانین مربوط، فقط خانه‌های متناظر مجموعه بیتی با آن قوانین با عدد یک و بقیه خانه‌های مجموعه بیتی با صفر مقداردهی می‌شوند. شکل ۵، مجموعه بیتی مربوط به قوانین اختصاص‌یافته به گره ۳۱ درخت جستجوی AVL شکل ۴ را نشان می‌دهد.

اجزای ساختار داده هر گره درخت جستجوی AVL برای انجام دسته‌بندی بسته‌ها عبارتند از:

- متغیر *Data*: برای نگهداری مقدار هر گره
- اشاره‌گر *Left_child*: برای نگهداری آدرس فرزند چپ
- اشاره‌گر *Right_child*: برای نگهداری آدرس فرزند راست
- مجموعه بیتی *Rule_bitset*: برای نگهداری لیست قوانین اختصاص‌یافته به هر گره

در این پژوهش برای هر کدام از فیلدهای پنج‌گانه آدرس مبدأ، آدرس مقصد، درگاه مبدأ، درگاه مقصد و پروتکل مربوط به قوانین، یک درخت جستجوی AVL جداگانه ایجاد می‌شود (شکل ۶).

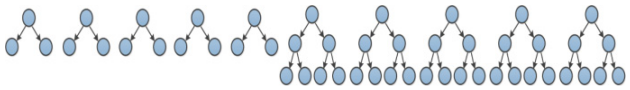
۳-۲-۳ جستجوی بسته ورودی روی درخت AVL

در ابتدا فایل مربوط به بسته‌های تولیدشده توسط ابزار ClassBench خوانده شده و سپس برای هر بسته، عمل جستجو به صورت موازی روی درخت‌های پنج‌گانه قوانین انجام می‌شود. نتیجه جستجو روی هر کدام از درخت‌های پنج‌گانه به صورت یک مجموعه بیتی برگردانده می‌شود. مجموعه بیتی برگردانده‌شده متعلق به گره انطباق‌یافته با بسته است. در صورت عدم تطابق بسته با هیچ کدام از گره‌ها، یک مجموعه بیتی تهی برگردانده می‌شود. در آخر، از نتایج درخت‌های پنج‌گانه اشتراک گرفته می‌شود تا مجموعه بیتی نتیجه نهایی به دست آید. برای اشتراک‌گیری، مجموعه بیتی‌های برگردانده‌شده به عنوان نتیجه جستجوی درخت‌های پنج‌گانه با هم AND می‌شوند. اقدام مربوط به بااولویت‌ترین قانون به عنوان نتیجه دسته‌بندی بسته ورودی در نظر گرفته می‌شود. بااولویت‌ترین قانون، قانون متناظر با اولین بیت یک در مجموعه بیتی نتیجه نهایی است.

۳-۳ الگوریتم دسته‌بندی ترافیک‌آگاه

در این روش، در کنار درخت‌های پنج‌گانه قوانین اصلی، از پنج درخت AVL دیگر نیز برای نگهداری قوانین پرتکرار استفاده شده است (شکل ۷). اگر بسته ورودی زودتر توسط درخت‌های پنج‌گانه قوانین پرتکرار انطباق یابد آن گاه جستجو در درخت‌های پنج‌گانه قوانین اصلی متوقف خواهد شد.

به عنوان یکی از ویژگی‌های ترافیک اینترنتی، بیان شد که تعداد کمی از قوانین برای دسته‌بندی حجم غالب ترافیک اینترنت در مدت زمان قابل توجهی استفاده می‌شوند. در اینجا برای استفاده از این ویژگی، با



شکل ۷: درخت‌های بزرگ سمت راست، درخت‌های قوانین اصلی و درخت‌های قوانین پرتکرار در سمت چپ.

اضافه‌کردن یک متغیر *Match_counter* به ساختار هر گره درخت جستجو، تعداد دفعات انطباق هر گره با بسته‌ها شمارش می‌شود. با ورود هر بسته اینترنتی و انجام عمل دسته‌بندی، مقدار متغیر *Match_counter* مربوط به گره انطباق‌یافته با آن بسته، یک واحد افزایش می‌یابد. هدف از این کار، یافتن قوانین با فراوانی انطباق بالا (قوانین پرتکرار) برای درج در درخت‌های قوانین پرتکرار است. درخت‌های قوانین پرتکرار به دلیل داشتن تعداد گره‌های کمتر نسبت به درختان قوانین اصلی، جستجو و انطباق بسته ورودی را با تعداد دسترسی به حافظه کمتری انجام می‌دهند.

٤- پیاده‌سازی و ارزیابی

برای پیاده‌سازی سناریوهای مورد نظر، از زبان برنامه‌نویسی C++ و سیستمی با پردازنده intel core i۵ M۴۳۰ ۲٫۲۷GHz و حافظه با دستیابی مستقیم GB ۴ استفاده شده است.

مجموعه قانون و بسته‌هایی که برای ارزیابی الگوریتم پیشنهادی لازم است با استفاده از ابزار ClassBench ایجاد شده است. با استفاده از ابزار ClassBench می‌توان فایل مجموعه قوانین[1] و فایل بسته‌ها[2] را ایجاد کرد. باید به ازای هر قانون موجود در فایل مجموعه قوانین، یک جریان از بسته‌های منطبق‌شده با آن، ایجاد شود. در بسته‌های تولیدشده توسط ابزار ClassBench، به ازای هر بسته، شماره قانون[3] منطبق‌شده با بسته قرار می‌گیرد. با استفاده از این فیلد شماره قانون، هر بسته به جریان مشخصی تعلق می‌گیرد.

سه نوع کلی از مجموعه قانون‌ها توسط ابزار ClassBench ایجاد می‌شود که عبارتند از لیست کنترل دسترسی[4] (ACL)، دیواره آتش (FW) و زنجیره IP[5] (IPC). در این پژوهش از مجموعه قانون نوع ACL۲ با اندازه‌های مختلف در ارزیابی‌ها استفاده شده است. هر کدام از مجموعه قوانین تولیدشده با توجه به نوع و اندازه آنها نام‌گذاری می‌شوند. برای مثال، ACL۲_۱K به مجموعه قوانین لیست کنترل دسترسی با اندازه حدود ۱۰۰۰ قانون اشاره دارد. در این پژوهش، در ارزیابی‌ها از مجموعه قانون نوع ACL۲ با اندازه ۱۰۰۰ و ۸۰۰۰ و بسته‌های با تعداد ۸۰۰۰، ۳۲۰۰۰ و ۱۲۸۰۰۰ استفاده شده است. نماد ACL۲_۱K و ACL۲_۸K به ترتیب برای نشان‌دادن مجموعه قوانین با اندازه‌های ۱۰۰۰ و ۸۰۰۰ است.

در اینجا برای بررسی تأثیر محلی‌بودن مراجعات بر دسته‌بندی بسته‌ها، با استفاده از پارامتر *locality* ابزار ClassBench، بسته‌هایی با میزان چولگی متفاوت ۰، ۰٫۲۵، ۰٫۵، ۰٫۷۵ و ۱ تولید شده است. هرچه مقدار چولگی بیشتر باشد به همان نسبت، تعداد تکرار بسته‌های جریان‌های مختلف بیشتر خواهد بود.

جدول ۲ تعداد گره درخت‌های پنج‌گانه قوانین اصلی ایجادشده بر اساس مجموعه قوانین ACL۲، FW۲، IPC۲ و Random را نشان

1. Rule Set
2. Packet Header
3. Rule Number
4. Access Control List
5. IP Chain

جدول ۲: تعداد گره درخت‌های پنج‌گانه قوانین اصلی برای مجموعه قوانین با نوع و اندازه‌های مختلف.

نوع قانون	اندازه مجموعه قانون	نوع درخت				
		SRC_IP	DEST_IP	SRC_Port	DEST_Port	Protocol
ACL۲	۵۰۰	۳۸۷	۴۵۵	۲	۲۶	۵
	۱۰۰۰	۴۱۲	۷۱۸	۲	۲۸	۵
	۸۰۰۰	۶۹۷۹	۷۵۵۳	۲	۲۸	۵
FW۲	۵۰۰	۵۴۵	۱۶۱	۱۰	۲	۵
	۱۰۰۰	۱۳۵۸	۴۶۵	۱۰	۲	۵
	۸۰۰۰	۱۲۲۸۱	۳۹۳۴	۱۰	۲	۵
IPC۲	۵۰۰	۱۲۱	۵۰	۴	۴	۴
	۱۰۰۰	۲۳۴	۷۲	۴	۴	۴
	۸۰۰۰	۵۸۰۰	۱۰۳۹۵	۴	۴	۴
Random	۵۰۰	۸۳۸	۸۱۳	۹۹۳	۹۹۲	۲۱۹
	۱۰۰۰	۱۵۸۳	۱۵۷۲	۱۹۶۵	۱۹۷۳	۲۵۳
	۸۰۰۰	۱۱۰۳۱	۱۰۸۷۸	۱۴۱۵۹	۱۴۲۲۳	۲۵۶

می‌دهد. در این جدول نشان داده شده که تعداد گره درخت‌های SRC_Port، DEST_Port و Protocol برای مجموعه قوانین نوع ACL۲، FW۲ و IPC۲ خیلی کم است در حالی که تعداد گره‌ها برای نوع Random زیاد می‌باشد.

٤-١ جزئیات پیاده سازی

در این بخش جزئیات بیشتری از پیاده‌سازی الگوریتم ترافیک‌آگاه و سناریوی به روز رسانی آنلاین و پارامترهای مهمی که در آنها استفاده شده است بیان می‌شود.

٤-١-١ پارامترهای ترافیک‌آگاه

در این پژوهش از پارامترهای زیر نیز برای نگهداری آستانه‌های مورد نیاز دسته‌بندی ترافیک‌آگاه استفاده شده است:

پارامتر $Insert\_th$: کمترین تعداد انطباق مورد نیاز را جهت درج یک گره جدید در درخت‌های پنج‌گانه قوانین پرتکرار نگهداری می‌کند.

پارامتر $Delete\_th$: کمترین تعداد انطباق یک گره پرتکرار را جهت جلوگیری از کاندید جایگزینی‌شدن آن مشخص می‌کند.

پارامتر $Rechance\_th$: حداکثر تعداد دفعات اعطای شانس مجدد به یک گره با قوانین بدون انطباق را مشخص می‌کند. در صورت بزرگ‌ترشدن مقدار متغیر $Rechance$ گره از این پارامتر، آن گره از درخت قوانین پرتکرار حذف خواهد شد.

پارامتر $Cache\_size\_max$: حداکثر تعداد گره‌های یک درخت قوانین پرتکرار را نگهداری می‌کند.

پارامتر $Time\_interval$: در پایان هر دوره زمانی به روز رسانی‌های درخت‌های قوانین پرتکرار انجام می‌شود. مقدار این پارامتر مشخص می‌کند که هر دوره زمانی چه قدر طول می‌کشد.

٤-١-٢ سناریوی به روز رسانی آنلاین

در این سناریو نیز از بسته‌های برچسب‌گذاری شده استفاده می‌شود. عملکرد این سناریو به صورت زیر است:

- در پایان هر جستجو، با انطباق بسته ورودی با یک گره، مقدار متغیر $Match\_counter$ آن یک واحد افزایش می‌یابد و سپس مقدار آن با پارامتر $Insert\_th$ مقایسه می‌شود. در صورت بزرگ‌تربودن مقدار متغیر $Match\_counter$ از پارامتر $Insert\_th$، آن گره در درخت قوانین پرتکرار درج خواهد شد.
- به دلیل درج آنلاین گره‌های پرتکرار، نیازی به پیمایش درخت‌های قوانین اصلی در پایان هر دوره زمانی برای یافتن گره‌های پرتکرار نیست. تنها در پایان هر دوره زمانی، عملیات پیمایش درخت‌های قوانین پرتکرار جهت مقداردهی متغیرهای هر گره و در صورت لزوم حذف آنها انجام می‌شود.

٤-٢ جزئیات پیاده‌سازی

در این قسمت به معرفی معیارهای ارزیابی کارایی الگوریتم دسته‌بندی خواهیم پرداخت.

٤-٢-١ زمان جستجو

برای ارزیابی زمان جستجوی الگوریتم، تعداد دفعات دسترسی به حافظه برای خواندن یک گره درخت جستجو در نظر گرفته شده است. هرچه تعداد کل دسترسی‌ها به حافظه کمتر باشد دسته‌بندی بسته‌ها سریع‌تر خواهد بود. در ارزیابی‌ها، تعداد دسترسی‌ها به حافظه به دو قسمت زیر تقسیم شده‌اند:

- تعداد دسترسی‌ها به حافظه مربوط به عملیات دسته‌بندی
- تعداد دسترسی‌ها به حافظه مربوط به عملیات به روز رسانی درخت‌های قوانین شامل درج و حذف یک گره و پیمایش درخت جستجو برای انجام به روز رسانی‌ها

در نمودارهای ارزیابی، نماد *Memory Access* برای نشان‌دادن تعداد دسترسی‌ها به حافظه، نماد *Skewness* برای مقادیر چولگی مختلف و نماد *Number of Packets* برای نشان‌دادن تعداد بسته‌ها است.

٤-٢-٢ نرخ برخورد

قوانین با فراوانی انطباق بالا در درخت‌های قوانین پرتکرار ذخیره می‌شوند. به خاطر تعداد کم این قوانین و ارتفاع کم درخت‌های قوانین پرتکرار، هرچه تعداد بسته‌های بیشتری با استفاده از این قوانین دسته‌بندی شوند تعداد کل دسترسی‌ها به حافظه کمتر خواهد بود. نرخ برخورد بیشتر به معنای دسته‌بندی تعداد بسته‌های بیشتری با قوانین پرتکرار و در نتیجه دسته‌بندی سریع‌تر است. نرخ برخورد از (۱) به دست می‌آید

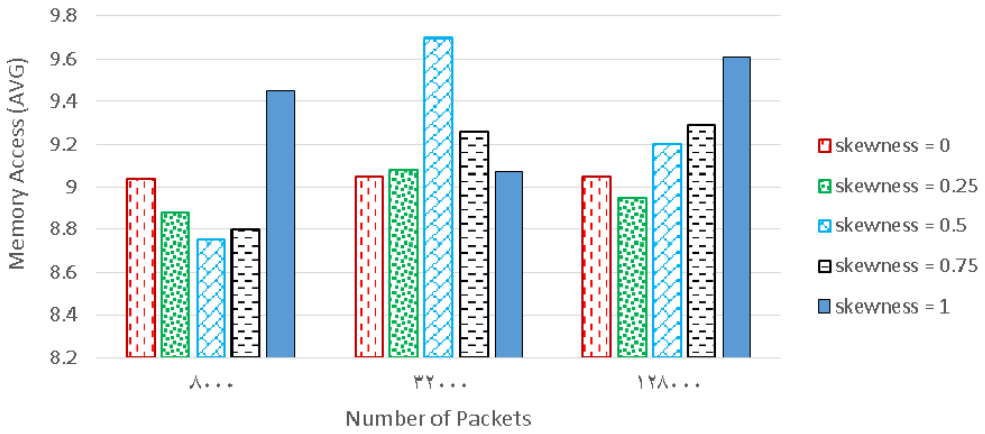
$$Hit\ Rate = \frac{Number\ of\ Matched\ Packets}{Number\ of\ Packets} \times ۱۰۰ \qquad (۱)$$

در این رابطه، نماد *Number of Packets* نشان‌دهنده تعداد کل بسته‌ها و نماد *Number of Matched Packets* نشان‌دهنده تعداد بسته‌هایی است که به وسیله درخت‌های قوانین پرتکرار دسته‌بندی شده‌اند. در جداول و نمودارها از نماد *Hit Rate* برای نشان‌دادن مقادیر مربوط به این معیار ارزیابی کارایی استفاده شده است.

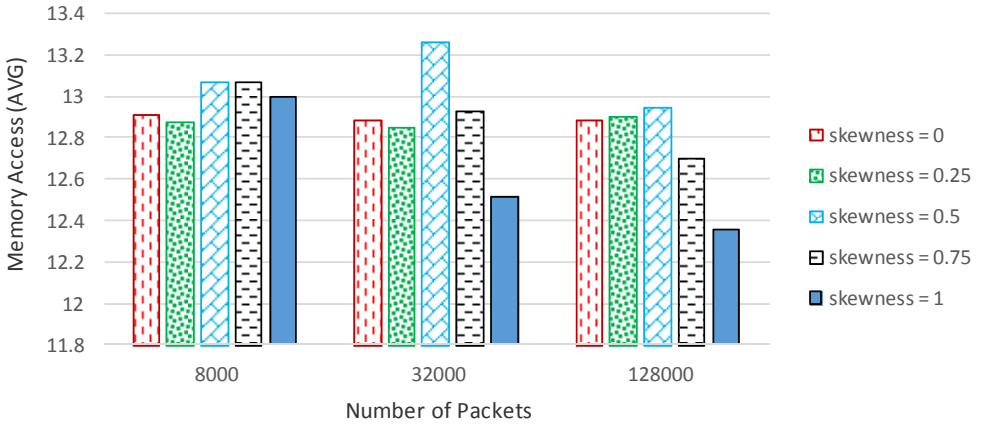
٤-٢-٣ نرخ بهره

این معیار ارزیابی نشان‌دهنده میزان کاهش تعداد کل دسترسی به حافظه برای دسته‌بندی بسته‌ها با الگوریتم ترافیک‌آگاه نسبت به دسته‌بندی بسته‌ها با الگوریتم پایه است. در این پژوهش از رابطه زیر برای به دست آوردن نرخ بهره استفاده می‌شود

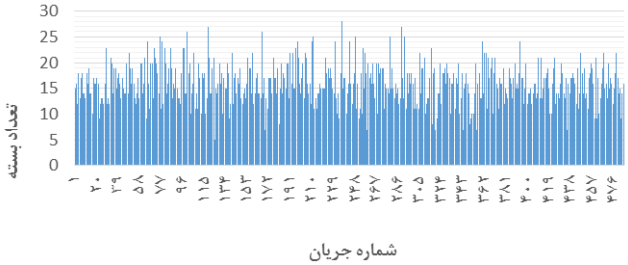
$$Gain\ Rate = \frac{BaseMemoryAccesses - TAMemoryAccesses}{Base\ Memory\ Accesses} \times ۱۰۰ \qquad (۲)$$



شکل ۸: میانگین تعداد دسترسی به حافظه به ازای هر بسته برای مجموعه قانون با اندازه ۱۰۰۰.



شکل ۹: میانگین تعداد دسترسی به حافظه به ازای هر بسته برای مجموعه قانون با اندازه ۸۰۰۰.



شکل ۱۰: تعداد بسته جریان‌های تولیدشده به تعداد ۸۰۰۰ به ازای چولگی صفر.

نماد *Base Memory Accesses* نشان‌دهنده تعداد کل دسترسی به حافظه الگوریتم دسته‌بندی پایه و نماد *TAMemoryAccesses* نشان‌دهنده تعداد کل دسترسی به حافظه الگوریتم دسته‌بندی ترافیک‌آگاه است. در ارزیابی‌ها از نماد *Gain Rate* برای نشان‌دادن مقادیر مربوط به معیار ارزیابی نرخ بهره استفاده شده است.

٤-۲-٤ میزان حافظه مصرفی

هرچه تعداد گره‌های درخت‌های قوانین پرتکرار یا پارامتر *Cache_size_max* بیشتر باشد تعداد دسترسی کمتری را خواهیم داشت. از یک مقداری به بعد افزایش پارامتر *Cache_size_max* تأثیری نخواهد داشت و در واقع باید مصالحه‌ای بین مقدار حافظه مصرفی و تعداد دسترسی به حافظه صورت گیرد.

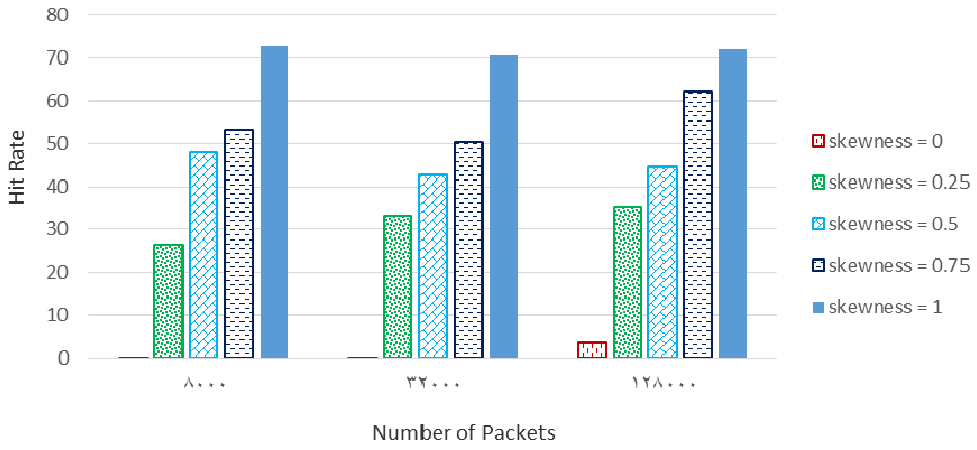
٤-۳ نتایج ارزیابی دسته‌بندی پایه

در شکل ۸ و ۹ مشاهده می‌شود که هرچه اندازه قوانین بزرگ‌تر شود میانگین تعداد دسترسی به حافظه بیشتر می‌شود. در هر یک از این نمودارها مشاهده می‌شود که با چولگی‌های مختلف بسته‌های آزمون، وابسته به محل قرارگیری قوانین با فراوانی انطباق بالا در درخت جستجو تعداد دسترسی به حافظه کاهش یا افزایش می‌یابد. یعنی اگر قوانین پرتکرار در سطح‌های نزدیک به گره ریشه درخت جستجو قرار داشته باشند سبب کاهش و در غیر این صورت باعث افزایش تعداد دسترسی‌ها به حافظه می‌شود. این نمودارها نشان می‌دهد که الگوریتم دسته‌بندی پایه، ویژگی محلی‌بودن مراجعات را در جهت کاهش تعداد دسترسی‌ها به حافظه در نظر نمی‌گیرد.

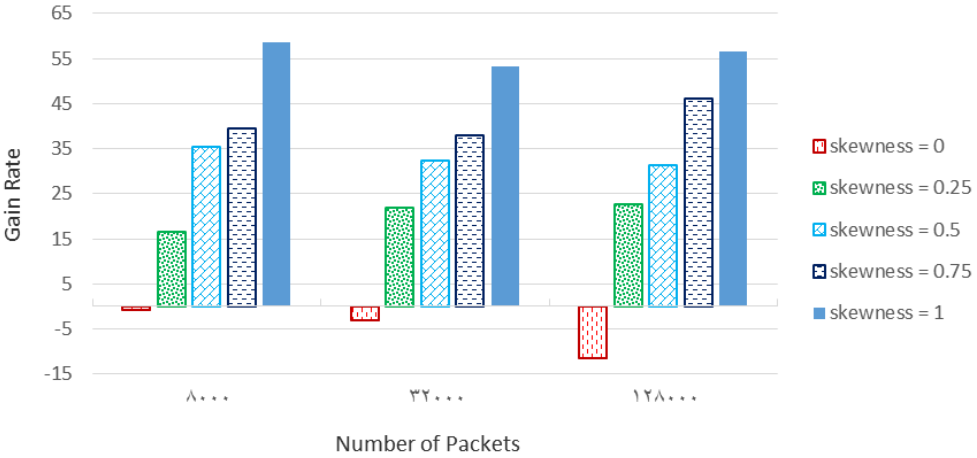
٤-٤ نتایج ارزیابی الگوریتم دسته‌بندی ترافیک‌آگاه

شکل‌های ۱۰ تا ۱۲، نتایج ارزیابی الگوریتم دسته‌بندی ترافیک‌آگاه (سناریوی به روز رسانی آنلاین) برای مجموعه قوانین مختلف و بسته‌های با تعداد مختلف را نشان می‌دهد. شکل ۱۰ بیانگر تعداد بسته جریان‌های تولیدشده به تعداد ۸۰۰۰ و به ازای چولگی صفر است. در شکل ۱۱ و ۱۲ به ترتیب نرخ برخورد و نرخ بهره برای مجموعه قوانین با اندازه ۸۰۰۰ آورده شده است. مشاهده می‌شود که با افزایش چولگی به دلیل افزایش تکرار بسته‌های جریان‌های مختلف، نرخ برخورد و نرخ بهره افزایش چشم‌گیری می‌یابد به طوری که نرخ بهره می‌تواند تا بیش از ۴۰ درصد افزایش داشته باشد. همچنین هرچه تعداد بسته‌های آزمون بیشتر باشد نرخ برخورد به ازای چولگی‌های مختلف نیز افزایش بیشتری می‌یابد.

برای بسته‌های آزمون بزرگ‌تر، نرخ برخورد به ازای چولگی‌های پایین افزایش چشم‌گیری می‌یابد و به نرخ برخورد چولگی‌های بالاتر نزدیک‌تر می‌شود. افزایش نرخ برخورد باعث افزایش نرخ بهره می‌شود، یعنی هرچه بسته‌های بیشتری با درخت‌های قوانین پرتکرار انطباق یابد تعداد کل



شکل ۱۱: نرخ برخورد درخت‌های قوانین پرتکرار در دسته‌بندی با تعداد بسته‌های مختلف به ازای چولگی‌های متفاوت با مجموعه ACL۲_۸K.



شکل ۱۲: نرخ بهره الگوریتم دسته‌بندی ترافیک‌آگاه با تعداد بسته‌های مختلف به ازای چولگی‌های متفاوت با مجموعه قانون نوع ACL۲_۸K.

دسترسی به حافظه کاهش بیشتری می‌یابد.

در شکل ۱۲ مشاهده می‌شود که نرخ بهره برای بسته‌های با چولگی صفر، منفی شده است. مطابق شکل ۱۰ این به آن علت است که تعداد بسته‌های اکثر جریان‌ها بین ۱۰ تا ۲۵ است. یعنی چون مقدار پارامتر *TH_insert* برابر با ۱۵ است قوانین مربوط به اکثر جریان‌ها به درخت قوانین پرتکرار اضافه شده و سپس دوباره حذف می‌شوند که عدم پایداری را در پی دارد. نرخ بهره‌های منفی مربوط به دیگر قوانین نیز به همین دلیل به وجود آمده است.

با افزایش اندازه مجموعه قوانین مورد ارزیابی، همچنان نرخ برخورد و نرخ بهره با افزایش چولگی بسته‌های آزمون با اندازه‌های مختلف افزایش چشم‌گیری می‌یابد، یعنی الگوریتم دسته‌بندی ترافیک‌آگاه ارائه‌شده برای مجموعه قوانین با اندازه‌های مختلف کارا است.

در جدول ۳، نتایج ارزیابی الگوریتم پیشنهادی دسته‌بندی ترافیک‌آگاه و سناریوی به روز رسانی آنلاین برای مجموعه قوانین نوع ACL۲_8k و بسته‌های با تعداد ۸۰۰۰، ۳۲۰۰۰ و ۱۲۸۰۰۰ به ازای کجی‌های ۰، ۰/۲۵، ۰/۵، ۰/۷۵ و ۱ آورده شده است. در این جدول، کمینه، میانگین و بیشینه تعداد دسترسی به حافظه عملیات دسته‌بندی، میانگین تعداد دسترسی به حافظه عملیات به روز رسانی، مجموع میانگین تعداد دسترسی به حافظه عملیات دسته‌بندی و عملیات به روز رسانی، مجموع میانگین تعداد دسترسی به حافظه عملیات دسته‌بندی و عملیات به روز رسانی با استفاده از حافظه TCAM برای ذخیره درخت‌های قوانین پرتکرار، میانگین تعداد دسترسی به حافظه الگوریتم دسته‌بندی پایه، نرخ برخورد درخت‌های قوانین پرتکرار و نرخ بهره تعداد دسترسی به حافظه الگوریتم دسته‌بندی ترافیک‌آگاه نسبت به الگوریتم دسته‌بندی پایه آورده شده است. مشاهده می‌شود که با افزایش کجی بسته‌های آزمون، میانگین تعداد دسترسی به حافظه عملیات دسته‌بندی و میانگین تعداد دسترسی به حافظه عملیات به روز رسانی کاهش می‌یابد.

جدول ۴ هم تعداد دسترسی به حافظه در عملیات دسته‌بندی بسته‌های الگوریتم ترافیک‌آگاه پیشنهادی را با کار مشابه و ترافیک‌آگاه دیگر که شامل ۲ الگوریتم "TA-DT" و "TA-CBDT" [۸] است نمایش می‌دهد. در این جدول *Avg* بیانگر متوسط تعداد دسترسی و *Max* بیانگر بیشترین (بدترین حالت) تعداد دسترسی به حافظه در اجرای عملیات الگوریتم است. دقت شود که تعداد قوانین مجموعه داده‌ای در روش پیشنهادی ۵۰۰ قانون، یعنی ۵ برابر تعداد قوانین جدول ارزیابی در [۸] است که با ۱۰۰ قانون ارزیابی انجام شده است. با این وجود، تعداد دسترسی به حافظه در روش پیشنهادی، هم در حالت متوسط و هم در بدترین حالت، برای مقادیر چولگی کمتراز ۰/۵، نزدیک به مقادیر مرجع [۸] است. برای مقادیر چولگی بزرگتر یا مساوی با ۰/۵، تعداد دسترسی (زمان جستجو) در الگوریتم پیشنهادی کمتر است. بنابراین با وجود تعداد قوانین بیشتر (مجموعه داده‌ای بزرگ‌تر)، کارایی بهتر روش پیشنهادی مشخص است.

٤-٥ محاسبه پیچیدگی‌ها

در این قسمت پیچیدگی تعداد دسترسی به حافظه و پیچیدگی حافظه مصرفی الگوریتم‌های دسته‌بندی پایه و الگوریتم دسته‌بندی ترافیک‌آگاه به بایت محاسبه شده و نمادهای استفاده‌شده برای محاسبه پیچیدگی‌ها در جدول ۵ آمده است.

جدول ۳: تعداد دسترسی به حافظه و میانگین نرخ برخورد برای مجموعه قانون ACL۲_۸K .

تعداد بسته در ارزیابی	Skewness	تعداد دسترسی به حافظه							Hit Rate	Gain Rate
		Classify			Avg Update	Sum	Avg TCAM	Avg Base		
		Min	Avg	Max						
۸۰۰۰	۰	۲	۱۲٫۸۷	۱۵	۰٫۲۱	۱۳٫۰۹	۱۳٫۰۳	۱۲٫۹۱	۰٫۴۶	-۰٫۷۵
	۰٫۲۵	۱	۱۰٫۳۰	۱۵	۰٫۴۳	۱۰٫۷۳	۱۰٫۱۶	۱۲٫۸۸	۲۶٫۴۳	۱۶٫۶۱
	۰٫۵	۱	۸٫۰۷	۱۵	۰٫۳۶	۸٫۴۴	۷٫۴۹	۱۳٫۰۷	۴۸٫۱۵	۳۵٫۴۳
	۰٫۷۵	۲	۷٫۵۶	۱۵	۰٫۳۳	۷٫۹	۶٫۹۱	۱۳٫۰۷	۵۳٫۳۲	۳۹٫۵۰
	۱	۱	۵٫۱۷	۱۵	۰٫۲	۵٫۳۸	۴٫۴۶	۱۳	۷۲٫۷۷	۵۸٫۶۱
۳۲۰۰۰	۰	۲	۱۲٫۸۵	۱۶	۰٫۴۳	۱۳٫۲۸	۱۳٫۲	۱۲٫۸۹	۰٫۳۷	-۳٫۰۴
	۰٫۲۵	۱	۹٫۵۹	۱۵	۰٫۵۳	۱۰٫۱۲	۹٫۳۶	۱۲٫۸۵	۳۳٫۰۲	۲۱٫۹۲
	۰٫۵	۱	۸٫۶۷	۱۵	۰٫۲۸	۸٫۹۶	۸٫۳۶	۱۳٫۲۶	۴۲٫۸۱	۳۲٫۴۴
	۰٫۷۵	۱	۷٫۶۲	۱۵	۰٫۳۷	۸	۷٫۱۶	۱۲٫۹۳	۵۰٫۴۸	۳۸٫۰۹
	۱	۲	۵٫۵۷	۱۵	۰٫۲۸	۵٫۸۵	۴٫۷۴	۱۲٫۵۱	۷۰٫۵۸	۵۳٫۱۸
۱۲۸۰۰۰	۰	۲	۱۲٫۵۷	۱۶	۱٫۷۹	۱۴٫۳۷	۱۳٫۹۶	۱۲٫۸۹	۳٫۷	-۱۱٫۴۸
	۰٫۲۵	۱	۹٫۳۸	۱۶	۰٫۶	۹٫۹۹	۹٫۱۹	۱۲٫۹۰	۳۵٫۲۱	۲۲٫۵۶
	۰٫۵	۱	۸٫۴۱	۱۶	۰٫۴۸	۸٫۹	۸	۱۲٫۹۵	۴۴٫۵۹	۳۱٫۲۶
	۰٫۷۵	۱	۶٫۴۹	۱۶	۰٫۳۶	۶٫۸۵	۵٫۸۱	۱۲٫۷	۶۲٫۱۵	۴۶٫۰۳
	۱	۱	۵٫۱۵	۱۵	۰٫۲۲	۵٫۳۷	۴٫۵۷	۱۲٫۳۶	۷۱٫۹۴	۵۶٫۵۲

جدول ۴: مقایسه تعداد دسترسی به حافظه در الگوریتم پیشنهادی با الگوریتم ارائه‌شده در [۸].

روش پیشنهادی			کار ارائه‌شده در [۸]		
Skewness	تعداد دسترسی به حافظه برای مجموعه قانون ACL۲_۵۰۰		تعداد دسترسی به حافظه		
	Avg	Max	TA-DT		TA-CBDT
			Avg	Max	Avg
۰	۵٫۱۰	۱۱	۴٫۸۰	۱۱٫۵۸	۴٫۸۴
۰٫۲۵	۴٫۹۸	۱۱			
۰٫۵	۴٫۶۴	۱۱			
۰٫۷۵	۴٫۰۱	۱۱			
۱	۳٫۶۵	۱۱			

جدول ۵: نمادهای استفاده‌شده برای محاسبه پیچیدگی‌ها.

نماد	توضیح نماد
n_1	بیشینه تعداد گره درخت‌های پنج‌گانه قوانین اصلی
n_2	بیشینه تعداد گره درخت‌های پنج‌گانه قوانین پرتکرار
n_3	مجموع تعداد گره‌های درخت‌های پنج‌گانه قوانین اصلی
n_4	مجموع تعداد گره‌های درخت‌های پنج‌گانه قوانین پرتکرار
Hit rate	نرخ برخورد بسته‌های ورودی با درخت‌های قوانین پرتکرار
r	تعداد قوانین دسته‌بند

٤-٥-١ پیچیدگی تعداد دسترسی به حافظه

درخت جستجوی AVL مورد استفاده، یک نوع درخت جستجوی باینری متوازن است. بنابراین حداکثر ارتفاع آن $\log n$ است به طوری که نماد n، نشان‌دهنده تعداد گره‌های درخت جستجوی AVL است. در اینجا، تعداد دسترسی به حافظه برای خواندن هر گره درخت جستجو را یک در نظر می‌گیریم.

در الگوریتم دسته‌بندی پایه، هر بسته ورودی در بهترین حالت با یک بار دسترسی به حافظه (انطباق با ریشه درخت) دسته‌بندی می‌شود.

در بدترین حالت، بسته ورودی، درخت جستجو را تا گره برگ پیمایش می‌کند و تعداد دسترسی به حافظه به اندازه ارتفاع درخت است. رابطه (۳) پیچیدگی تعداد دسترسی به حافظه الگوریتم دسته‌بندی پایه را در بدترین حالت نشان می‌دهد

$$MemoryAccess = \mathrm{O}(\log_{۲}^{n_1}) \quad (۳)$$

در الگوریتم دسته‌بندی ترافیک‌آگاه، علاوه بر درخت‌های قوانین اصلی از درخت‌های قوانین پرتکرار نیز استفاده می‌شود. هر بسته ورودی به طور هم‌زمان در درخت‌های قوانین اصلی و درخت‌های قوانین پرتکرار جستجو می‌شود. اگر بسته ورودی با گره از درخت‌های پرتکرار انطباق یابد، آن گاه جستجو در درخت‌های قوانین اصلی متوقف می‌شود. در غیر این صورت، جستجوی بسته ورودی در درخت‌های قوانین اصلی ادامه می‌یابد. در (۴) پیچیدگی تعداد دسترسی به حافظه سناریوی به روز رسانی آنلاین الگوریتم دسته‌بندی ترافیک‌آگاه در بدترین حالت نشان داده شده است

$$MAOnline = (۱ - HitRate) \times \mathrm{O}(\log_{۲}^{n_1}) + (HitRate \times \mathrm{O}(\log_{۲}^{n_2})) + \beta \quad (۴)$$

در (۴)، نماد *Hit Rate* به تعداد تکرار بسته‌های ورودی یا چولگی بسته‌ها و پارامترهای ترافیک‌آگاه *Rechance_th*، *Insert_th*، *Delete_th*، *Time_interval* و *cache_size_max* وابسته است. هرچه چولگی بسته‌ها بیشتر باشد *Hit rate* نیز بیشتر خواهد شد. نماد β، بیانگر تعداد دسترسی به حافظه عملیات به روز رسانی آنلاین درخت‌های قوانین

پرتکرار به ازای هر بسته است. تعداد دسترسی به حافظه عملیات به روز رسانی وابسته به مقادیر پارامترهای *Time_interval*، *Insert_th*، *Delete_th*، *Cache_size_max* و *Rechance_th* هستند.

٤-٥-٢ پیچیدگی حافظه مصرفی

در (۵)، پیچیدگی حافظه مصرفی الگوریتم دسته‌بندی پایه آورده شده است

$$MemoryStorage = n_3 \times (12 + \frac{r}{8}) \qquad (5)$$

متغیر *Data*، اشاره‌گر *Left_child* و اشاره‌گر *Right_child* هر کدام چهار بایت حافظه مصرف می‌کنند. عدد ۱۲، مجموع حافظه این سه متغیر است. هر گره شامل یک مجموعه بیتی *Rule_bitset* با اندازه تعداد کل قوانین یعنی r است. هر خانه این مجموعه بیتی یک بیت فضا اشغال می‌کند و در نتیجه فضای مصرفی توسط مجموعه بیتی *Rule_bitset* با "$r/8$" نشان داده شده است.

در الگوریتم دسته‌بندی ترافیک‌آگاه، اجزای ساختار داده گره‌های درخت‌های پنج‌گانه قوانین (اصلی و پرتکرار) عبارت هستند از متغیر *Data*، اشاره‌گر *Left_child*، اشاره‌گر *Right_child*، مجموعه بیتی *Rule_bitset*، متغیر *Match_counter* و متغیر *last_match_time*. ساختار داده گره‌های درخت‌های پنج‌گانه قوانین اصلی، علاوه بر اجزای گفته‌شده در فوق، شامل متغیرهای *cached*، *Rechance* و *Replacement_candidate* است.

رابطه (۶) بیانگر پیچیدگی حافظه مصرفی الگوریتم دسته‌بندی ترافیک‌آگاه برای درخت‌های پنج‌گانه قوانین اصلی است

$$MemoryStorage = n_3 \times (25 + \frac{r}{8}) \qquad (6)$$

متغیر *Data*، اشاره‌گر *Left_child*، اشاره‌گر *Right_child* و متغیر *Match_counter* هر کدام چهار بایت، متغیر *Last_time_stamp* هشت بایت و متغیر *Cached* یک بایت فضا اشغال می‌کنند. عدد ۲۵، بیانگر مجموع حافظه این شش متغیر است.

رابطه (۷) پیچیدگی حافظه مصرفی الگوریتم دسته‌بندی ترافیک‌آگاه برای درخت‌های پنج‌گانه قوانین پرتکرار است

$$MemoryStorage = n_4 \times (29 + \frac{r}{8}) \qquad (7)$$

در این رابطه علاوه بر متغیرهای فوق، متغیر چهاربایتی *Rechance* و متغیر یک‌بایتی *Replacement_candidate* اضافه شده و عدد ۲۹ بیانگر مجموع حافظه این هشت متغیر است.

در (۸)، پیچیدگی حافظه مصرفی الگوریتم دسته‌بندی ترافیک‌آگاه آورده شده است. با توجه به این که حافظه مصرفی کل الگوریتم ترافیک‌آگاه برابر با مجموع حافظه مربوط به درخت‌های پنج‌گانه قوانین اصلی و حافظه مربوط به درخت‌های قوانین پرتکرار است، (۸) از حاصل‌جمع (۶) و (۷) به دست می‌آید

$$MemoryStorage = n_3 \times (25 + \frac{r}{8}) + n_4 \times (29 + \frac{r}{8}) \qquad (8)$$

برای مجموعه قانون ACL۲ با اندازه ۵۰۰، الگوریتم دسته‌بندی پایه به ۶۵ کیلوبایت حافظه برای نگهداری درخت‌های پنج‌گانه قوانین نیاز دارد. در حالی که الگوریتم دسته‌بندی ترافیک‌آگاه به ۷۶ کیلوبایت حافظه برای نگهداری درخت‌های قوانین اصلی و ۷ کیلو بایت حافظه برای نگهداری درخت‌های قوانین پرتکرار نیاز دارد. مشاهده می‌شود که الگوریتم ترافیک‌آگاه حافظه بیشتری را مصرف می‌کند. شکل ۱۱ نشان می‌دهد که در سناریوی به روز رسانی آنلاین الگوریتم دسته‌بندی ترافیک‌آگاه تعداد کل دسترسی به حافظه تا ۵۵ درصد کاهش یافته است. یعنی در ازای حافظه مصرفی بیشتر، تعداد کل دسترسی به حافظه به طور قابل توجهی کم شده است.

برای پارامتر *Cache_size_max* در دسته‌بندی ترافیک‌آگاه از مقدار ۱۵ استفاده شده است، یعنی حداکثر تعداد گره درخت‌های قوانین پرتکرار می‌تواند ۱۵ باشد. با افزایش این پارامتر، تعداد دسترسی به حافظه کاهش و حافظه مصرفی افزایش می‌یابد. بنابراین بین تعداد دسترسی به حافظه و حافظه مصرفی باید مصالحه‌ای برقرار شود.

۵- نتیجه‌گیری

در این پژوهش، ویژگی‌های آماری ترافیک ورودی در نظر گرفته شده و برای کاهش تعداد دفعات دسترسی به حافظه و افزایش سرعت جستجو، از داده‌ساختارهای کمکی در کنار داده‌ساختارهای اصلی استفاده شده است. در الگوریتم پیشنهادی، از ساختار درخت خودمتوازن AVL برای نگهداری قوانین استفاده شده است. الگوریتم دسته‌بندی پایه ارائه‌شده برای دسته‌بندی هر بسته ورودی، از جستجوی موازی درخت‌های پنج‌گانه قوانین استفاده می‌کند در حالی که الگوریتم دسته‌بندی ترافیک‌آگاه، از قوانین پرتکرار و اثر محلی‌بودن مراجعات بسته‌ها در ساختن درخت‌های قوانین کمکی بهره می‌برد. ارزیابی‌ها نشان می‌دهند که با افزایش چولگی بسته‌های آزمون، نرخ برخورد درخت‌های قوانین پرتکرار نیز افزایش می‌یابد. همچنین نشان داده شد که با افزایش چولگی‌های بسته‌های آزمون، نرخ بهره الگوریتم دسته‌بندی ترافیک‌آگاه ارائه‌شده نیز افزایش می‌یابد، یعنی افزایش محلی‌بودن مراجعات بسته‌های ترافیک ورودی به قوانین دسته‌بند سبب کاهش بیش از ۴۰ درصدی تعداد کل دسترسی به حافظه الگوریتم دسته‌بندی ترافیک‌آگاه نسبت به الگوریتم دسته‌بندی پایه می‌گردد.

در کارهای آتی می‌توان در پایان هر دوره زمانی، ترافیک‌های منطبق‌شده با قانون پیش‌فرض متناظر با ترافیک مخرب را شناسایی و سپس آنها را در درخت قوانین بسته‌های مخرب درج کرد. با این روش، بسته‌های مخرب در همان ابتدای دسته‌بندی توسط درخت قوانین بسته‌های مخرب شناسایی می‌شوند و از دسته‌بندی آنها توسط درخت‌های قوانین اصلی جلوگیری می‌شود. همچنین می‌توان مقداردهی پویا و آنلاین پارامترهای ترافیک‌آگاه را با توجه به الگوی ترافیک ورودی انجام داد که می‌تواند بر اساس نرخ برخورد و نرخ بهره در پایان هر دوره زمانی انجام گیرد.

مراجع

[1] P. Gupta and N. McKeown, "Algorithms for packet classification," *IEEE Network,* vol. 15, no. 2, pp. 24-32, Mar. 2001.

[2] H. Oudah, B. Ghita, and T. Bakhshi, "A novel features set for internet traffic classification using burstiness," in *Proc. Int. Conf. ICISSP*, pp. 397-404, Prague, Czech Republic, 23-25 Feb..

[3] L. Ding, J. Liu, T. Qin, and H. Li, "Internet traffic classification based on expanding vector of flow," *COMPUT. NETW*, vol. 129, no. 1, pp. 178-192, Dec. 2017.

[4] H. Kim, *et al.*, "Internet traffic classification demystified: myths, caveats, and the best practices," in *Proc. Int. Conf. ACM CoNEXT*, Article 11, pp. 1-12, Dec. 2008.

[5] A. Bergamini and L. Kencl, "Network of shortcuts: an adaptive data structure for tree-based search methods," in *Proc. Int. Conf. Networking,* pp. 523-535, May 2005.

[6] A. El-Atawy, T. Samak, E. Al-Shaer, and H. Li, "Using online traffic statistical matching for optimizing packet filtering performance," in *Proc. IEEE Int. Conf.*, on Computer Communications, pp. 866-874, Anchorage, AK, USA, 6-12 May 2007.
[7] H. Hamed and E. Al-Shaer, "Dynamic rule-ordering optimization for high-speed firewall filtering," in *Proc. Int. Conf. ASIACCS,* pp. 332-342, Taipei Taiwan, 21 Mar.2006.
[8] E. Cohen and C. Lund, "Packet classification in large ISPs: design and evaluation of decision tree classifiers," in *Proc. Int. Conf. ACM SIGMETRICS*, pp. 73-84, Jun. 2005.
[9] W. Yu, S. Sivakumar, and D. Pao, "Pseudo-TCAM: SRAM-based architecture for packet classification in one memory access," *IEEE Networking Letters*, vol. 1, no. 2, pp. 89-92, Feb. 2019.
[10] S. S. Vegesna, A. C. Nara, and N. M. Sk, "A novel rule mapping on TCAM for power efficient packet classification," *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, vol. 24, no. 5, pp. 1-23, Jun. 2019.
[11] Z. Liu, S. Sun, H. Zhu, J. Gao, and J. Li, "BitCuts: a fast packet classification algorithm using bit-level cutting," *Computer Communications*, vol. 109, no. 1, pp. 38-52, Sept. 2017.
[12] M. Abbasi, S. V. Fazel, and M. Rafiee, "MBitCuts: optimal bit-level cutting in geometric space packet classification," *The J. of Supercomputing*, vol. 76, no. 4, pp. 3105-3128, Apr. 2020.
[13] W. Li, X. Li, H. Li, and G. Xie, "Cutsplit: a decision-tree combining cutting and splitting for scalable packet classification," in *Prof. IEEE Conf. on Computer Communications*, *INFOCOM'18,* pp. 2645-2653, Honolulu, USA, 16-19 Apr. 2018.
[14] C. L. Hsieh, N. Weng, and W. Wei, "Scalable many-field packet classification for traffic steering in SDN switches," *IEEE Trans. on Network and Service Management*, vol. 16, no. 1, pp. 348-361, Sept. 2018.
[15] W. Li, D. Li, Y. Bai, W. Le, and H. Li, "Memory-efficient recursive scheme for multi-field packet classification," *IET Communications*, vol. 13, no. 9, pp. 1319-1325, Feb. 2019.
[16] X. Dong, M. Qian, and R. Jiang, "Packet classification based on the decision tree with information entropy," *The J. of Supercomputing*, vol. 76, no. 6, pp. 4117-4131, Jun. 2020.
[17] A. A. Abdulhassan and M. Ahmadi, "Cuckoo filter-based many-field packet classification using X-tree," *The J. of Supercomputing*, vol. 75, no. 9, pp. 5667-5687, Sept. 2019.
[18] S. Greenberg, T. Sheps, D. A. Leon, and Y. Ben-Shimol, "Packet classification using GPU and one-level entropy-based hashing," *IEEE Access*, vol. 8, [18] pp. 80610-80623, Apr. 2020.
[19] B. Xiong, Z. Hu, Y. Luo, and J. Wang, "Cuckooflow: achieving fast packet classification for virtual openflow switching by exploiting network traffic locality," in *Proc. IEEE Intl Conf. on Parallel & Distributed Processing*, pp. 1123-1130, Xiamen, China, 16-18 Dec. 2019.
[20] A. El-Atawy, H. Hamed, and E. Al-Shaer, *Adaptive Statistical Optimization Techniques for Firewall Packet Filtering*, DePaul Univ., Chicago, IL, Tech. Rep. CTI-TR-05-019, 2005.
[21] S. Acharya, J. Wang, Z. Ge, T. F. Znati, and A. Greenberg, "Traffic-aware firewall optimization strategies," in *Proc. IEEE Int. Conf. ICC'06*, vol. 5, pp. 2225-2230, Istanbul, Turkey, 11-15 Jun. 2006.
[22] S. Acharya, *et al.*, "OPTWALL: a hierarchical traffic-aware firewall," in *Proc. Int. Conf. NDSS*, 11 pp., San Diego, CA, USA, 28 Feb. 2007.
[23] S. H. Shaikot and M. S. Kim, "NPF: A simple, traffic-adaptive packet classifier using on-line reorganization of rule trees," in *Proc. IEEE Int. Conf. LCN*, pp. 899-906, Zurich, Switzerland, 20-23 Oct. 2009.
[24] S. H. Shaikot and M. S. Kim, "Lightweight traffic-aware packet classification for continuous operation," in *Proc. IEEE Int. Conf. Int. Symp. on Applications and the Internet*, pp. 59-67, Seoul, Korea (South), 19-23 Jul. 2010.
[25] A. S. Sairam, R. Kumar, and P. Biswas, "Implementation of an adaptive traffic-aware firewall," in *Proc. Int. Conf. SINCONF*, pp. 385-391, , Glasgow, Scotland UK, 9 Sept. 2014.
[26] D. E. Taylor and J. S. Turner, "Classbench: a packet classification benchmark," *IEEE/ACM Trans. on Networking*, vol. 15, no. 3, pp. 499-511, Jun. 2007.
[27] L. Zhang, "Virtual clock: a new traffic control algorithm for packet switching networks," in *Proc. Int. Conf. SIGCOMM*, pp. 19-29, Philadelphia, PA, USA, 1 Aug. 1990.
[28] R. Jain and S. Routhier, "Packet trains--measurements and a new model for computer network traffic," *IEEE J. SEL AREA COMM.*, vol. 4, no. 6, pp. 986-995, Sept. 1986.
[29] N. Alborz and L. Trajkovic, "Implementation of virtualclock scheduling algorithm in OPNET," in *Proc. Int. Conf. OPNETWORK*, 7 pp., Aug. 2001.
[30] D. E. Knuth, The Art of Computer Programming 3: Sorting and Searching, AddisonWesley, MA, 1973.
[31] T. Srinivasan, M. Nivedita, and V. Mahadevan, "Efficient packet classification using splay tree models," in *Proc. Int. Conf. IJCSNS*, vol. 6, pp. 28-35, Seoul, Korea (South), May 2006.

سعید اسدروز مدرک کارشناسی ارشد خود را در رشته مهندسی کامپیوتر گرایش شبکه‌های کامپیوتری از دانشگاه بوعلی سینا در سال ۱۳۹۴ اخذ نمود. علاقمندی پژوهشی وی به بررسی و بهبود روش‌های دسته‌بندی بسته‌ها در اینترنت و نیز پیاده‌سازی و ارزیابی آنهاست.

محمد نصیری مدارک کارشناسی و کارشناسی ارشد خود را در رشته مهندسی نرم‌افزار از دانشگاه‌های علم و صنعت ایران و صنعتی شریف به ترتیب در سالهای ۱۳۷۹ و ۱۳۸۱ دریافت کرد. سپس مدرک دکتری خود را در گرایش شبکه‌های کامپیوتری در سال ۲۰۰۸ از پلی‌تکنیک گرونوبل–فرانسه اخذ نمود. وی هم‌اکنون به عنوان دانشیار مهندسی کامپیوتر در دانشگاه بوعلی سینا مشغول به انجام فعالیت‌های آموزشی و پژوهشی است. علاقمندی‌های اصلی وی پژوهش در حوزه‌های مختلف شبکه از جمله بهبود پروتکل‌های کنترل دسترسی و مسیریابی در فناوری‌های نوین شبکه، طراحی و ارزیابی پروتکل‌های ارتباطی در اینترنت اشیاء، دسته‌بندی بسته‌ها و جریان‌های ترافیکی است.

مهدی عباسی در سال‌های ۱۳۸۰ و ۱۳۸۵ به ترتیب مدرک کارشناسی مهندسی کامپیوتر و مدرک کارشناسی ارشد معماری سیستم‌های کامپیوتری خود را از دانشگاه صنعتی شریف دریافت نمود. او در سال ۱۳۹۱ موفق به اخذ درجه دکترا در مهندسی کامپیوتر از دانشگاه اصفهان گردید. دکتر عباسی از سال ۱۳۹۱ در گروه مهندسی کامپیوتر دانشگاه بوعلی سینا در همدان مشغول به فعالیت گردید. ایشان در حال حاضر با مرتبه علمی دانشیار در زمینه‌های علمی متنوعی مانند ایده‌های نو در معماری کامپیوتر، اینترنت اشیا، بهینه سازی، سیستم‌های تعبیه شده توزیع شده و پردازش سیگنال مشغول تدریس و تحقیق می‌باشد.

حاتم عبدلی تحصیلات خود را در مقاطع کارشناسی مهندسی کامپیوتر و کارشناسی ارشد معماری سیستم‌های کامپیوتری در سالهای ۱۳۷۸ و ۱۳۸۲ از دانشگاه صنعتی اصفهان به پایان رسانده است و پس از آن به عنوان مربی در دانشگاه بوعلی سینا مشغول به کار بوده است. نامبرده در سال ۱۳۹۶ موفق به اخذ درجه دکترا در مهندسی کامپیوتر از دانشگاه اصفهان گردید و هم اکنون استادیار گروه مهندسی کامپیوتر دانشگاه بوعلی سینا می‌باشد. زمینه‌های تحقیقاتی مورد علاقه ایشان شامل موضوعاتی مانند حساب کامپیوتری، معماری کامپیوتر، رایانش تقریبی، و سیستم‌های نهفته و اینترنت اشیاء می‌باشد.