علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۳، شماره ۱ (الف)، بهار ۱۳۸۴
صفحات ۱۱-۲۱
مقاله عادی

دسته بندی بسته های اینترنت بر اساس تطبیق پیشوندی با استفاده از تقاطع نگاشتهای بیتی

نیک محمد بلوچ زهی[۱] محمود فتحی[۲] صالح یوسفی[۲]

[۱]گروه مهندسی فنآوری اطلاعات، دانشگاه سیستان و بلوچستان ، زاهدان، ایران
[۲]دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، تهران، ایران

چکیده

با بالا رفتن سرعت خطوط ارتباطی و همچنین مطرح شدن کیفیت سرویسهای ارائه شده توسط شبکه، جدا کردن جریانهای متمایز در مسیریابهای اینترنت به عنوان یکی از راهکارهای بالا بردن سرعت و کارایی مسیریابها مورد توجه بوده است. در این فرآیند که دسته بندی بسته ها نامیده می شود، کلیّه بسته های متعلق به یک جریان مشخص، تحت تاثیر یک قانون خاص قرار گرفته و بطور یکسان توسط مسیریابها پردازش می شوند. به عبارت دیگر جریانها بر اساس یک یا چند مشخصه مانند آدرس مبداء، آدرس مقصد، پورت مبداء، پورت مقصد و شماره پروتکل از همدیگر متمایز می‌شوند و با هر کدام بر اساس قوانین تعریف شده برخورد می شود. انجام این عمل بر روی چندین فیلد سرآیند بسته عملی زمانبر و دارای بار حافظه ای بالایی می‌باشد. در این راستا روشهای مختلفی ارائه شده است که هر کدام با توجه به زمان دسته بندی، میزان حافظه مصرفی و نوع پیاده سازی(سخت افزاری- نرم افزاری)، برای کاربرد خاصی مناسب می باشند. اغلب این روشها نتوانسته اند مصالحه مناسبی بین معیارهای ذکر شده ارائه نمایند. در این مقاله روش جدیدی تحت عنوان PBBI[۱] ارائه می شود که با استفاده از خواص تفکیک کننده های واقعی که از مشاهدات تجربی حاصل شده اند و بیانگر این واقعیت هستند که تعداد پیشوندهای مجزا در هر بعد تفکیک کننده از تعداد قوانین آن بعد کمتر می‌باشد، میزان حافظه مصرفی و زمان دسته بندی مناسب‌تری نسبت به روشهای قبلی ارائه می دهد. ارزیابی کارایی انجام شده و همچنین نتایج حاصل ازشبیه سازی حاکی از کارایی مناسب الگوریتم ارائه شده است، به گونه ای که می‌توان گفت الگوریتم ارائه شده انتخاب مناسبی برای پیاده سازی تفکیک کننده های واقعی با تعداد قوانین زیاد می‌باشد.

کلمات کلیدی: دسته بندی بسته ها، ماتریس های بیتی و QOS

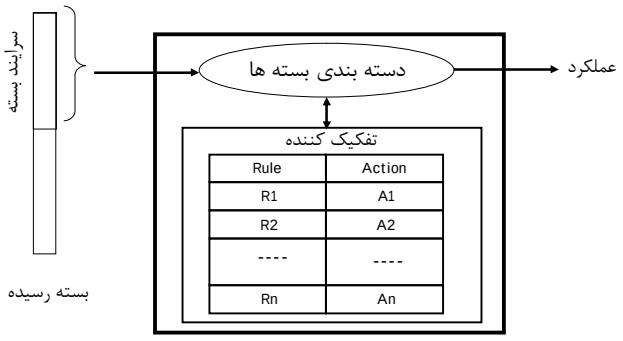
۱- مقدمه

امروزه با توجه به افزایش کاربران شبکه و ازدیاد حجم ترافیک موجود در خطوط ارتباطی و بوجود آمدن کاربردهای جدیدی مانند چند رسانه ای نیاز به شبکه هایی کاراتر افزایش یافته است. این مساله سبب شده است تا طراحان شبکه به دنبال راه حلی برای افزایش سرعت و کارایی شبکه باشند، که افزایش سرعت خطوط ارتباطی یکی از روشهای ارائه شده برای این منظور می باشد. به گونه ای که سرعت خطوط ارتباطی به بالاتر از "ترا بیت در ثانیه" رسیده است. برای رسیدن به کارایی مناسب در شبکه، مسیریابها و سوئیچها نیز باید بتوانند بسته ها را با همین سرعت مسیریابی وپردازش نمایند. علاوه بر مسیریابی و پردازش، مطرح شدن تضمین کیفیت سرویسها، عملیات خاصی بر روی بسته ها را می طلبد و در نتیجه مسیریابها به یک سری روش و مکانیزم جدید برای رزرو کردن منابع، تشکیل صف و زمانبندی عادلانه، جهت ارائه بهتر این سرویسها به کاربران نیاز دارند[۱].

با دسته بندی بسته ها و انجام پردازشهای مشابه بر روی بسته های مربوط به یک جریان، می توان زمان پردازش بسته ها را تا اندازه‌ای کاهش داد. برای رسیدن به سرعت و کیفیت بالاتر در ارائه سرویسها، مسیریابها بسته های دارای نیازهای کیفیت سرویس مشابه را در یک جریان قرار می دهند. عمل جداسازی بسته ها به جریانها بر اساس قوانینی که در مسیریابها وجود دارند انجام میشود، این قوانین در جدولی به نام تفکیک کننده[۲] به ترتیب اولویت قرار می‌گیرند[۲،۳].

نحوه دسته‌بندی به اینصورت است که با دریافت یک بسته، فیلدهای موجود در سرآیند بسته با قوانین موجود در تفکیک کننده مقایسه می شوند، سپس با اولویت

ترین قانون انتخاب می شود و براساس آن بسته در جریان متناظر قرار می‌گیرد. در اینجا هدف ارائه الگوریتمی است که بتواند برای هر بسته ورودی، با اولویت ترین قانون را از میان قانونهای موجود در تفکیک کننده انتخاب نماید. نمایی از چگونگی انجام این عمل در شکل ۱ آمده است.



شکل۱- دسته بندی بسته‌ها

اگر فرض کنیم که تفکیک کننده شامل N قانون می باشد($R_i, 0< i < N+1$) و هر قانون بر روی K بعد از سرآیند بسته تعریف شده است. بعد j ام قانون i بصورت $R_i[j]$ و فیلد jام سرآیند بسته بصورت $F[j]$ مورد ارجاع واقع می‌شوند. در این صورت ، گفته می شود بسته P با قانون R_i در انطباق است اگر به ازای کلیه فیلدهای بسته P،$F[j]$ با $R_i[j]$ در انطباق باشد. نوع انطباقی که بر روی هر کدام از فیلدها صورت می گیرد بستگی به نوع مقادیر آن فیلد دارد، با توجه به مقادیر فیلدها، آنها را به سه گروه تقسیم بندی می کنیم:

فیلدهای پیشوندی: هر پیشوند عبارت است از رشته ای از صفر و یکها که به یک * ختم می شوند و معمولا آدرسهای مبداء و مقصد بسته ها را با پیشوند نمایش می دهند.مثلا ممکن است آدرس مبداء بسته ای بصورت *111001 باشد.

فیلدهای محدوده ای: فیلدهایی مانند شماره پورتهای مبداء و مقصد در این دسته می گنجند.در این حالت مقدار هر فیلد شامل دو جزء می باشد که یکی نمایانگر ابتدای محدوده و دیگری نمایانگر پایان محدوده است.

فیلدهای دقیق: فیلدهایی مانند نوع پروتکل که دارای مقادیری محدود می باشند، در این دسته قرار می گیرند.

نوع انطباقی که بر روی فیلدها انجام می گیرد بستگی به نوع فیلد دارد، مثلا برای فیلدهای پیشوندی چنانچه مقدار فیلد بسته پیشوندی از مقدار تعیین شده در فیلد قانون باشد، می گوییم که بسته در این فیلد با قانون در انطباق می باشد، همچنین اگر مقدار فیلد محدوده ای بسته در بازهٔ تعیین شده توسط قانون بگنجد، بسته در این فیلد با قانون در انطباق می باشد، فیلدهای دقیق نیز فقط از نظر تساوی با همدیگر تطبیق داده می شوند.

بعنوان مثال اگر قوانین *R1(110*,010*,1-3,2-7,TCP)* و *R2(1110*,10*,2-3,1-3,UDP)* را داشته باشیم که در آنها دو فیلد اول پیشوندی و دو فیلد بعدی محدوده ای و فیلد آخری نیز از نوع دقیق می باشد و بسته ای مانند *P(11011,01000*,2,5,TCP)* به مسیریاب برسد، این بسته با قانون *R1* در انطباق خواهد بود.

می توان فیلدهای محدوده ای را به معادل پیشوندی آنها تبدیل نمود که هر فیلد محدوده ای می تواند به چندین پیشوند تبدیل شود. مثلا چنانچه محدودهٔ 4-10 به پیشوند تبدیل شود، شامل پیشوندهای *10 , *1 خواهد بود زیرا می توان بازهٔ 4-10 را به بازه های (*1)7-4 و (*10)8-10 تقسیم نمود.

روشهای زیادی برای دسته بندی بسته ها وجود دارند که پس از تقسیم بندی آنها به چهار گروه،هر گروه را جداگانه مورد بررسی قرار می دهیم.

الف)روشهای پایه‌ای

جستجـــوی خطی[۳] : در ایـن روش قـوانین بتـرتیب کاهش اولـویت در یک لیست پیوندی قرار می گیرند و بستهٔ ورودی به ترتیب با کلیهٔ بسته های لیست پیوندی مقایسه می شود.این الگوریتم از نظر میزان حافظه مناسب است زیرا هر قانون فقط یک بار در حافظه ذخیره می شود، اما قابلیت مقیاس پذیری ضعیفی دارد چون با افزایش تعداد قوانین، زمان دسته بندی بصورت خطی افزایش می یابد [۵،۴].

Trieهای سلسله مراتبی[۴] **:** یک Trie عبارتست از یک درخت دودویی که برای نمایش مجموعه ای از پیشوندها بکار می‌رود. در این روش هر بعد تفکیک کننده در یک سطح از درخت قرار می گیرد. این روش از نظر میزان حافظه مصرفی قابل قبول می باشد اما زمان جستجوی آن با افزایش تعداد قوانین افزایش می‌یابد [۵، ۳].

Trieهای هرس شده[۵] **:** این الگوریتم شبیه الگوریتم فوق می باشد با این تفاوت که با تکرار قوانین زمان جستجوی مورد نیاز را کاهش می دهد که با این عمل میزان حافظه مورد نیاز افزایش می یابد. در کل این الگوریتم برای تفکیک کننده های کوچک مناسب می باشد و در تفکیک کننده های امروزی که تعداد زیادی از قوانین را می توانیم داشته باشیم، کاربردی نخواهد داشت [۳].

ب) روشهای هندسی

Trieهای خوشه ای[۶]: در این ساختار مشکل حافظه مصرفی زیاد روش Trieهای هرس شده و مشکل زمان جستجوی زیاد روش Trieهای سلسله مراتبی حل شده است.به اینصورت که بجای تکرار قوانین از یک سری سوئیچ استفاده مـی شـود.بـا بکار بردن سوییچها در جاهایی که نیاز به تکـرار قـوانین داشـتیم(در روش هـرس شده)،از میزان حافظه مصرفی زیاد جلوگیری می شود.اما سـرعت جسـتجوی ایـن روش مشابه روش هرس می باشـد،زیرا در ایـن روش نیـز نیـازی بـه پیمایشـهای بازگشتی نداریم،بلکه فقط کافی است که سوییچ های ایجاد شـده را دنبـال کنـیم، زمان بهنگام سازی الگوریتم خیلی زیاد می‌باشد، بگونه ای که هنگام اضافه یا حذف کردن یک قانون بهتر است که ساختمان داده الگوریتم از اول ایجاد شود. بنـابراین این الگوریتم برای تفکیک کننده های ایستا در دو بعد مناسـب بـوده امـا براحتـی قابل تعمیم به حالت چند بعدی نمی باشد [۶].

Cross-Producting: الگوریتمTrieهای خوشه ای برای انطباق پیشوند دو بعدی مانند آدرسهای مبدأ و مقصد دارای کارایی خوبی می‌باشد. این روش نیاز به حافظه خطی دارد و زمانی معادل جستجوی دو پیشوند بر روی دو فیلد آدرس مقصد و مبدأ را می‌طلبد. با اینکه در خیلی از حالات ، بی‌نهایت مفید می‌باشد (مانند VPN)، اما درکاربردهای عملی مانند دوارهٔ آتش نیاز به تفکیک کننده های عمومی تری داریم.که در اینگونه کاربردها چون تفکیک کننده ها می توانند چند بعدی باشند،این ساختار نمی تواند خیلی مفید واقع شود.در این کاربردها نیاز به الگوریتم های بهتری داریم،یکی از این الگوریتم ها که می تواند برای چندین بعد مناسب باشد الگوریتم Cross-product می باشد.

ایده اصلی این روش به اینصورت می باشد: اگر تفکیک کننده را به یک سری ستون تکه‌تکه کنیم،بگونه ای که درI امین ستون همهٔ پیشوندهای مجزای فیلد I را ذخیره نماییم،در اینصورت تعداد مقادیر مجزای هر ستون در مقایسه با تعداد کل قوانین کم خواهد بود و یافتن یک مقدار از میان مجموعه مقادیر یک ستون سریعتر و راحت تر انجام می گیرد.با ترکیب مقادیر ستونها با همدیگر،می توانیم کلیه ترکیبات ممکن مقادیر فیلدهای مختلف را ایجاد نماییم که به هر کدام از این ترکیبات یک Cross-Product می گوییم[۶].

در این حالت ، برای یک بستهٔ داده شدهٔ P، ابتدا برای هر کدام از فیلدهای سرآیند بسته بصورت جداگانه بهترین انطباق پیشوند را می‌یابیم و سپس نتیجه همه یافته‌ها را با همدیگر ترکیب می‌نماییم[۶].

ج)روشهای مکاشفه‌ای

این گروه شامل روشهایی است که عمل دسته بندی را با توجه به بعضی از خواص تفکیک کننده هایی که در حال حاضـر در مسیـریابهای موجـود در شبکه مورد

استفاده قرار می‌گیرند (تفکیک کننده‌های واقعی) انجام میدهند. در این روشها ساختمان داده مورد نیاز الگوریتم با توجه به خواص این تفکیک کننده ها ساخته می‌شود[۷،۸].

دسته بندی با جریان بازگشتی[۷] : این روش بر مبنای نگاشت اطلاعات سرآیند بسته اطلاعاتی به تعداد بیت های کمتر در چندین فاز می باشد که این عمل را با توجه به خواص تفکیک کننده های واقعی انجام می دهد. این روش برای تعداد فیلدهای زیاد مناسب بوده و سرعت دستیابی به آن نیز بالا می باشد، اما دارای مقیاس پذیری ضعیفی است چون با افزودن یک فیلد به فیلدهای تفکیک کننده، ساختار آن به هم می ریزد و به علت دارا بودن پیاده سازی سخت افزاری، تغییرات در آن دشوار می باشد[۹].

برشهای سلسله مراتبی هوشمند[۸] : این الگوریتم بر خلاف الگوریتم قبلی دارای ساختاری پویا می باشد، بگونه ای که ساختمان داده خود را با ویژگیهای تفکیک کننده وفق می دهد و بر اساس آن ساخته می شود. این الگوریتم می تواند بین حافظه مصرفی و سرعت جستجو توازن برقرار کند. دسته بندی در این روش با پیمایش درخت و انجام جستجوی خطی در آخرین سطح انجام می گیرد. زمان پیش پردازش این الگوریتم بسیار بزرگ می باشد اما با تنظیم پارامترهای آن می توان بین زمان جستجو و میزان حافظه مصرفی توازن برقرار کرد [۱۰،۱۱].

فضای چند بعدی[۹] : این الگوریتم هر قانون K بعدی را به یک K تایی مرتب نگاشت می کند.بگونه ای که مجموعه قانونهای نگاشت شده به یک K تایی مرتب ، دارای طول ثابت و مشخصی بوده و در یک جدول درهم سازی[۱۰] قرار می گیرند. این الگوریتم در کل برای ابعاد زیاد مناسب بوده و زمان دسته بندی و زمان تغییرات نیز مناسب میباشد[۱۲].

د)روشهای سخت افزاری

Ternary CAM : حافظه CAM[۱۱] حافظه ای است که بصورت موازی و همزمان در یک پالس ساعت اولین برخورد عبارت مورد جستجو را برمی‌گرداند[۱۳] و چون در دسته بندی هدف این است که هر چه سریعتر به قانونی که با سرآیند بسته مطابقت دارد برسیم، پس اگر این حافظه مورد استفاده قرار گیرد، می تواند در سرعت جستجو موثر باشد. به این منظور جدول تفکیک کننده را داخل TCAM منتقل می کنیم که در اینصورت قوانین تفکیک کننده در سطرهای حافظه قرار خواهند گرفت و فیلدهای آن در ستونهای حافظه قرار می گیرند. برای انجام عمل جستجو، سرآیند بسته وارد ثبات آرگومان این حافظه می شود و درپالس بعدی اولین محل برخورد که همان شماره قانون می‌باشد، بعنوان خروجی از طریق ثبات Match مشخص می گردد[۱۴،۱۵]. اما مشکل عمده این روش این است که ابعادحافظه TCAM محدود می‌باشد و دیگر اینکه این حافظه ها دارای قیمت و توان مصرفی بالا می باشند. درنتیجه این روش فقط برای تفکیک کننده های با تعداد قوانین کم مناسب خواهد بود[۱۳،۱۶].

ماتریس بیتی[۱۲] : در این روش هر بعد به یک سری بازه مجزا تقسیم می شود، سپس برای هر بازه مجزا یک بردار بیتی در نظر گرفته می شود، اندازه این بردار بر حسب بیت برابر با تعداد قوانین تفکیک کننده می باشد. نحوه مقدار دهی این بردار به اینصورت است که اگر بازه مورد نظر در این بعد با قانونی همپوشانی داشته باشد، بیت متناظر با آن قانون مقدار یک و در غیر اینصورت بیت مورد نظر مقدار صفر را به خود می گیرد [۱۷].

برای انجام عمل دسته بندی این روش به اینصورت عمل می کند که به ازای هر فیلد سرآیند بسته، بازه مورد نظر در هر بعد را پیدا می کند، سپس بردارهای متناظر با این بازه ها را با همدیگر AND منطقی می‌شوند. در اینصورت اگر بیتی در بردار نتیجه برابر یک باشد، نمایانگر آن است که بسته رسیده با قانون متناظر با آن بیت در انطباق می باشد، اما با توجه به اینکه در تفکیک کننده ها قوانین به ترتیب اولویت مرتب شده اند، همیشه قانون متناظر با سمت چپ ترین یک در بردار نتیجه باید به بسته رسیده اعمال شود.

در این روش با توجه با اینکه تعداد بازه های ایجاد شده معمولا زیاد می‌باشد(دو برابر تعداد قوانین)، نیاز به حجم حافظه زیادی داریم بنابراین این روش فقط برای تفکیک کننده های در حجم کوچک مناسب می باشد[۱۷، ۷].

روش ماتریس بیتی دارای دو ویژگی مهم زیر می باشد:

فقط برای تفکیک کننده های در حد کوچک و متوسط مناسب بوده و بخوبی عمل می کند(به علت میزان حافظه مصرفی بالا برای تفکیک کننده های بزرگ مناسب نیست)[۵] و دیگر اینکه این روش دارای پیاده سازی نرم افزاری و سخت افزاری راحتی می باشد[۷،۸،۱۲].

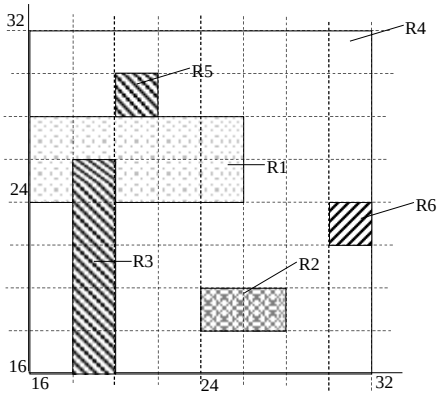
پس می توان گفت که این روش نسبت به روشهای ارائه شده در اغلب پارامترها بجز در میزان حافظه مصرفی، بهتر می‌باشد، در ارائه روشهای دسته بندی هدف ارائه روشی است که بتواند یک مصالحه بین پارامترهای دسته بندی برقرار نماید [۲،۷].

جهت درک بهترچگونگی انجام عمل دسته بندی توسط این روش، مثالی بر اساس تفکیک کننده شکل ۲می‌آوریم.

	Field1	Field2
R1	10*	110*
R2	110*	1001*
R3	1001*	10*
R4	1010*	1110*
R5	1111*	1011*
R6	1*	1*

شکل۲- تفکیک کننده نمونه

برای یافتن بازه های مجزای هر بعد، ابتدا باید کلیه پیشوندهای آن بعد را بر روی محور متناظر تصویر کنیم، سپس بازه های مجزا را با توجه به نقاط تلاقی قوانین از همدیگر تفکیک نماییم، تصویر بازه های مجزای هر بعد بر روی محورهای متناظر در شکل ۳ آورده شده است.



شکل۳- نمایش هندسی تفکیک کننده

در این روش به هر یک از بازه های ایجاد شده بر روی هر یک از محورها یک بردار N بیتی نسبت داده می شود، مثلا بردار نسبت داده شده به پنجمین بازه بر روی محور عمودی برابر با ۱۰۱۱۰۰ و بردار نسبت داده شده به سومین بازه روی محور افقی برابر ۱۰۰۱۱۰ می باشد.

در اینصورت این روش با دریافت بسته ای مانند P(101001,100101)، ابتدا بازه های متناظر را یافته و بردارهای نظیر را از حافظه می خواند، سپس AND

منطقی این بردارها را محاسبه می کند، در این مثال بازه متناظر با فیلد اول بازه سوم و برای فیلد دوم بازه شماره ۵ خواهد بود که بردارهای نظیر آنها باید از حافظه خوانده شوند.

فیلد اول => 100110
فیلد دوم => 101100
AND ----------
100100

مشاهده می شود که بسته رسیده با قانونهای اول و چهارم در انطباق است، از آنجا که قوانین به ترتیب اولویت مرتب شده‌اند،بنابراین قانون شماره یک باید به بسته رسیده اعمال شود.

با توجه به شکل ۳ می بینیم که تعداد بازه های ایجاد شده بر روی محور افقی برابر با ۹ و بر روی محور عمودی نیز برابر با ۹ می باشد(با در نظر گرفتن بازهٔ ۱۶-۰ که در شکل نشان داده نشده است)، پس در کل به ازای ۶ قانون ۱۸ بازهٔ مجزا خواهیم داشت، بنابراین باید ۱۸ بردار ۶ بیتی برای ذخیره تفکیک کننده ای با ۶ قانون در نظر بگیریم.

در جدول ۱ الگوریتمهای فوق از نظر زمان مورد نیاز برای دسته‌بندی و میزان حافظه مصرفی بر اساس سه پارامتر N (تعداد قوانین موجود در تفکیک کننده)، W (طول پیشوندها) ، K (تعداد ابعاد قانون‌ها) و $memWidth$ (طول کلمات حافظه) با همدیگر مقایسه شده‌اند. همانطور که دیده می‌شود، هر کدام از روش‌ها سعی در برقراری مصالحه مناسب بین پیچیدگی زمانی و پیچیدگی حافظه‌ای دارند ولی اغلب آنها در بهبود یکی از معیارها موفق تر بوده اند.

جدول ۱- مقایسه پیچیدگی زمانی و حافظه ای روشهای موجود

الگوریتم	زمان	حافظه
جستجوی خطی	N	N
Trieهای سلسله مراتبی	W^K	NKW
Trieهای هرس شده	KW	$N^K KW$
برشهای سلسله مراتبی هوشمند	K	N^K
Trieهای خوشه ای	$W^{(K-1)}$	NKW
Cross-Producting	KW	N^K
دسته بندی با جریان بازگشتی	K	N^K
جستجوی فضای چند بعدی	N	N
TCAM	1	N
روش ماتریس بیتی	$KW + KN/memwidth$	$K(2N+1)N$

روش ماتریس بیتی علیرغم اینکه از لحاظ پیچیدگی زمانی شرایط مناسبی دارد، دارای حافظه مصرفی بالایی می باشد. از این رو فقط برای تفکیک‌کننده‌های کوچک مناسب است. اما با توجه به اینکه حجم تفکیک کننده‌ها بطور روزافزون در حال رشد می‌باشد، این روش نمی‌تواند برای تفکیک کننده های بزرگی که در آینده با آنها سروکار خواهیم داشت،مورد استفاده قرار گیرد. در نتیجه چنانچه مقدارحافظه مصرفی زیاد این روش کاهش یابد، این روش گزینه مناسبی برای تفکیک کننده‌های بزرگ خواهد بود. در این راستا بعضی تلاش‌ها انجام شده است که بهبودهایی بر روی الگوریتم پایه ماتریس بیتی انجام شود تا بتوان آن را در تفکیک کننده‌هاب بزرگ واقعی مورد استفاده قرار داد. تا کنون کارهای محدودی برای بهبود روش ماتریس بیتی ارایه شده است. یکی از این روش‌ها ABV[۱۳] می باشد که با توجه به خواص تفکیک کننده های واقعی فقط زمان دسته بندی روش ماتریس بیتی را کاهش می دهد و حافظهٔ مصرفی این روش را افزایش می دهد [۷،۸]. این روش با توجه به این واقعیت که هر بسته رسیده به مسیریاب با تعداد محدودی از قوانین تفکیک کننده در انطباق خواهد بود، زمان دسته بندی بسته ها را تا حد زیادی افزایش می دهد که انجام این کار مستلزم بکارگیری ماتریسهای کمکی می باشد که سبب افزایش زمان دسته بندی الگوریتم می شوند. روش دیگری که برای بهبود روش ماتریس بیتی ارائه شده است روش AFBV[۱۴] می باشد که این روش نیز سرعت دسته بندی روش را بهبود بخشیده ولی میزان حافظهٔ مصرفی آنرا افزایش می دهد[۱۸].

در این مقاله روشی جدید تحت عنوان PBBI ارائه می دهیم که با توجه به خواص تفکیک کننده های واقعی که نشاندهنده واقعیت هستند که تعداد پیشوندهای مجزا در هر بعد تفکیک کننده از تعداد قوانین آن بعد کمتر می‌باشد، میزان حافظه مصرفی زیاد روش ماتریس بیتی را برای فیلدهای پیشوندی کاهش می‌دهد. همچنین روش پیشنهادی زمان دسته بندی روش ماتریس بیتی را نیز تا حد زیادی کاهش خواهد داد

۲- روش پیشنهادی (PBBI[۱۵])

با توجه به مطالب بالا می توان گفت که اگر میزان حافظه مصرفی روش ماتریس بیتی کاهش یابد، به گونه ای که میزان حافظه مصرفی روش به حد قابل قبولی برسد، می توان در تفکیک کننده های بزرگ نیز از آن استفاده کرد.

در این بخش روشی ارائه می شود که با توجه به خواص تفکیک کننده های واقعی حافظه مصرفی روش ماتریس بیتی را کاهش می دهد، خواصی از تفکیک کننده های واقعی که در ارائه این روش توسط مولفین مد نظر قرار گرفته اند، عبارتند از:

الف)تعداد بازه های مجزا بر روی هر بعد تفکیک کننده که با تصویر نمودن قوانین بر روی هر محور ایجاد می‌شوند، معمولا از تعداد قوانین تفکیک کننده بیشتر خواهد بود، به گونه ای که در بدترین حالت تعداد بازه های مجزا برابر با 2N+1 می باشد[۵].

ب)تعداد پیشوندهای مجزای(پیشوندهای غیر تکراری) هر بعد، از تعداد قوانین تفکیک کننده کمتر می باشد. زیرا مقادیر یک فیلد از تفکیک کننده در قانونهای مختلف تکرار می‌شوند. به گونه‌ای که اگر تعداد مقادیر مجزای هر بعد(مقادیر غیر تکراری) را T فرض کنیم،همواره داریم T<=N[۹].

در روش PBBI بر خلاف روش ماتریس بیتی می خواهیم بجای اینکه به هر بازهٔ مجزا یک بردار N بیتی نسبت دهیم، به هر پیشوند مجزا یک بردار N بیتی نسبت دهیم[۲۴]، که در این حالت چون تعداد پیشوندهای مجزا برابر T خواهد بود، T بردار N بیتی خواهیم داشت ، با توجه به خواص تفکیک کننده های واقعی می بینیم که انجام این عمل برای فیلدهایی از نوع پیشوندی مناسب می باشد، اما برای فیلدهایی مانند شماره پورتهای مبداء و مقصد که از نوع محدوده‌ای می‌باشند، مناسب نمی باشد.

اگر هر بازه یا محدوده بخواهد به کمک پیشوندها نمایش داده شود باید آنرا به پیشوندهای معادل تبدیل نمود. که با توجه به اینکه هر بازه می تواند به تعداد زیادی پیشوند تبدیل شود، این عمل دارای بار حافظه ای بالایی خواهد بود. تعداد پیشوندهایی که با تبدیل یک بازهٔ w بیتی(بازه ای که با w بیت قابل نمایش باشد.مثلا بازهٔ ۱۲-۴ یک بازهٔ ۴ بیتی است زیرا بزرگترین عدد این بازه یعنی ۱۲ با ۴ بیت قابل نمایش می باشد) به پیشوند بدست می آیند، از رابطهٔ زیر بدست می آید که در آن NumPrefix تعداد پیشوندهای تولید شده با توجه به w بیتی بودن بازه است[۵].

$$NumPrefix(w) = 2 * w - 2 \qquad (۱)$$

با توجه به اینکه طول فیلدهای محدوده ای در IPV.4 برابر ۱۶ بیت می باشد، پس طبق رابطهٔ فوق در بدترین حالت هر کدام از این فیلدها می توانند به ۳۰ پیشوند تبدیل شوند.که اگر برای ذخیرهٔ هر پیشوند مجزا یک بردار N بیتی در نظر گرفته شود،حافظه در نظر گرفته شده در بدترین حالت ($T = N$) به ازای ذخیرهٔ N بازه در حافظه طبق رابطه ۲ محاسبه خواهد شد.

$$\Pr efixM(N) = 30 * N * N \qquad (۲)$$

اما برای حالتی که عمل تبدیل به پیشوند را انجام ندهیم(بلکه قوانین را بر روی ابعاد مختلف تصویر نماییم)، چون هر بازه حداکثر به $2*N+1$ بازهٔ مجزا تبدیل می شود[۵] ، اگر برای ذخیرهٔ هر بازه N بیت نیاز داشته باشیم، میزان حافظه طبق رابطه ۳ بدست خواهد آمد:

$$RangeM(N) = N(2N+1) \qquad (۳)$$

با توجه به عبارات ۲ و۳ می بینیم که بکار بردن روش پیشنهادی برای فیلدهایی که از نوع محدوده ای باشند، مناسب نمی باشد، بنابراین از این روش برای فیلدهایی که از نوع پیشوندی باشند، مانند آدرس مبداء و مقصد استفاده می کنیم. نتایج حاصل از تفکیک کننده های واقعی نیز بیانگر همین واقعیت می باشد [۱۹]. به بیان دیگر روش PBBI دارای کارایی مناسبتری نسبت به روش ماتریس بیتی در مورد میدانهای پیشوندی میباشد ولی در مورد میدانهای محدوده‌ای استفاده از روش ماتریس بیتی توصیه میشود. شبیه‌سازی انجام شده در این مقاله نیز بر این اساس است.

الگوریتم PBBI دارای دو مرحله می باشد، این الگوریتم ابتدا در مرحله پیش پردازش، ساختمان داده های مورد نیاز برای ذخیره سازی و انجام عمل دسته بندی را ایجاد می نماید. سپس در مرحله بعدی از این ساختارها برای انجام عمل دسته بندی بسته ها استفاده می نماید، که مراحل انجام هر کدام از این دو بخش در شکل ۴ نشان داده شده است.

شکل۴- فلوچارت انجام مراحل الگوریتم PBBI
(الف) مرحله پیش پردازش (ب)مرحله دسته بندی

۲-۱ مرحله پیش پردازش الگوریتم

مرحله پیش پردازش شامل ایجاد ماتریس محدوده از روی تفکیک کننده اولیه، ایجاد Trieها وایجاد ماتریسهای پراکندگی به ازای هر بعد می‌باشد، در ابتدا هر کدام از اصطلاحات فوق را در زیر شرح می دهیم.

ماتریس محدوده: این ماتریس موقتی بوده و فقط در مرحله پیش پردازش مورد استفاده قرار میگیرد، در این ماتریس به ازای هر قانون یک سطر وجود دارد که در هر عنصر این ماتریس شماره منحصر بفردی ذخیره می شود که نشان دهندهٔ شمارهٔ نسبت داده شده به پیشوند متناظر می باشد. اما اندازهٔ هر عنصر بر حسب بیت با توجه به تعداد مقادیر مجزا می تواند در هر بعد متفاوت باشد، مثلاً چنانچه در یک بعد ۸ پیشوند متفاوت وجود داشته باشد، کافی که عناصر آن بعد را سه بیتی فرض نماییم.که منظور از نمایش تفکیک کننده در ماتریس محدوده با تعداد بیت کمتر نیز همین می باشد.

ماتریسهای پراکندگی: به ازای هر بعد تفکیک کننده یک ماتریس به نام ماتریس پراکندگی داریم که به ازای هر پیشوند مجزا تعیین می نماید که با چه قوانینی از تفکیک کننده در انطباق می باشد. این ماتریس نیز در مرحلهٔ پیش پردازش ایجاد می شود و اندازه آن نیز $T*N$ بیت می باشد که T تعداد پیشوندهای مجزا در بعد مورد نظر و N تعداد قوانین تفکیک کننده می باشد.

۲-۱-۱ ایجاد ماتریس محدوده از روی تفکیک کنندهٔ اصلی

این ماتریس دارای ابعاد $N * \hat{d}$ می باشد که در آن N تعداد قوانین و $\hat{d}$ ابعاد تفکیک کننده با تعداد بیت کمتر می‌باشد. این ماتریس در واقع یک نگاشت از تفکیک کننده اصلی می باشد که الگوریتم در این مرحله تعداد مقادیر مجزای هر بعد را یافته و به هر کدام یک شناسه منحصربفرد نسبت می دهد، این مقادیر در ماتریسی به نام ماتریس محدوده قرار می‌گیرند. شبه کد مورد نیاز برای ایجاد ماتریس محدوده از روی تفکیک کننده اصلی در شکل ۵ آمده است.

```
/* Create Range Matrix(R) From Classifier(C) */
For each dim d in 1:k do
   R(d,1) =1   /*   Set first row of matrix to 1  */
For each dim d in 1:k do
   MaxId=2
   For each prefix P in dim d do
    If p=m in dim d and m is before than P in Dim d then
      R(d,p)=R(d,m)
   Else
     R(d,p)=MaxId
     MaxId=MaxId+1.
```

شکل۵- شبه کد ایجاد ماتریس محدوده

ماتریس محدودهٔ ایجاد شده با توجه به با توجه به تفکیک کنندهٔ شکل ۲ و الگوریتم بالا طبق شکل ۶ خواهد بود.

	Field1	Field2
R1	1	1
R2	2	2
R3	3	3
R4	4	4
R5	5	5
R6	6	6

شکل۶- ماتریس محدوده

۲-۱-۲ ایجاد Trieها به ازای هر بعد

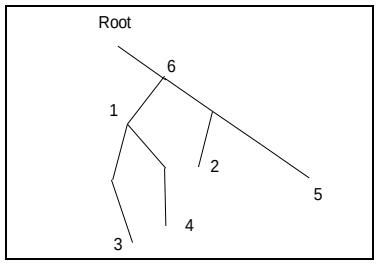
به ازای هر بعد تفکیک کننده یک Trie خواهیم داشت که با توجه به ماتریس محدوده و تفکیک کنندهٔ اصلی ایجاد می‌شود. روش ساخت Trie به اینصورت

می باشد که برای هر عبارت از ریشه شروع کرده و با هر مقدار صفر در پیشوند به فرزند چپ و با هر مقدار یک در پیشوند به فرزند راست حرکت می کنیم تا کلیهٔ پیشوندها به درخت اضافه شوند، شبه کد انجام این کار در شکل ۷ آمده است.

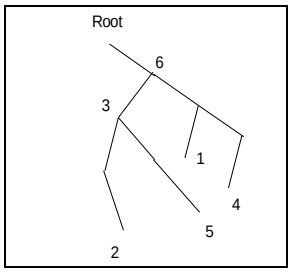
```
1.  For each dim d in 1:k do
2.  for Each Prefix P in Dim d do
3.    Add P to Trie(d)
4.     Add to last node of p prefix, R(d,p).
```

شکل۷- شبه کد مورد نیاز برای ایجاد Trieها

Trie های ایجاد شده با توجه به تفکیک کنندهٔ شکل ۲ و الگوریتم فوق در شکل۸ آمده است.



(الف)



(ب)

شکل۸- Trieهای ایجاد شده به ازای تفکیک کننده

(الف) Trie ایجاد شده با ازای بعد اول (ب)Trie ایجاد شده با ازای بعد دوم

۲-۱-۳ ایجاد ماتریسهای پراکندگی به ازای هر بعد

به ازای هر بعد از تفکیک کننده یک ماتریس پراکندگی داریم که اندازهٔ این ماتریس برابر با T*N بیت می‌باشد. بعبارتی در این ماتریس به ازای هر پیشوند مجزای بعد Kام تفکیک کننده،یک بردار N بیتی در نظر گرفته می‌شود.که هر بیت این بردارمتناظر با یک قانون می باشد. شبه کد مورد نیاز برای ایجاد ماتریسهای پراکندگی در شکل ۹ آمده است.

```
1.  Set D[T*N] as an Empty Matrix
2.  For Each dim d in 1:k  do
3.    For Each prefix p In Dim d Do
4.       Traverse  Trie(d)
5.        If m Exists in Trie(d) AND m=R(d,p) and P is a field of
    Rule I then
6.           D[p,I]=1
7.         Else
              D[p,I]=0.
```

شکل۹- شبه کد مورد نیاز برای ایجاد ماتریسهای پراکندگی

ماتریسهای پراکندگی ایجاد شده به ازای تفکیک کنندهٔ شکل ۲ در شکل ۱۰ آمده است.

0010*	1	0	0	0	0	0	1
0101*	0	1	0	0	0	0	0
1000*	0	0	1	0	0	0	0
1010*	0	0	0	1	0	0	0
1100*	0	0	0	0	1	1	0
11*	0	0	0	0	0	1	0

(الف)

0*	1	0	0	0	0	0	0
1001*	0	1	0	0	0	0	0
0110*	1	0	1	0	0	0	1
1010*	0	0	0	1	0	0	0
1100*	0	0	0	0	1	0	0

(ب)

شکل۱۰- ماتریسهای پراکندگی با توجه به تفکیک کننده

(الف)ماتریس پراکندگی بعد اول (ب) ماتریس پراکندگی بعد دوم

۲-۲ مرحلهٔ دسته بندی هر بستهٔ رسیده

هر گاه بسته ای مانند P به مسیریاب برسد،الگوریتم ابتدا به ازای هر فیلد بطور جداگانه و مستقل بزرگترین طول پیشوند را پیدا می کند، که در این مرحله شمارهٔ شناسهٔ پیشوندها بدست می آید.این عمل با استفاده از روش‌های پیدا کردن بزرگترین انطباق (LPM^i) انجام می گیرد که این برای جستجوی یک پیشوند بر روی درختهای پیشوندی که در این حالت درختها حتما باید یک بعدی باشند. این روش با انجام عمل جستجوی دودویی بر روی طول پیشوندها و استفاده از خواص تفکیک کننده های واقعی در ایجاد درخت جستجوی دودویی زمان جستجو برای یافتن یک پیشوند را از w به Log(w) کاهش می‌دهد.[۲۲]

این عدد همان شمارهٔ برداری می‌باشد که باید از ماتریس پراکندگی خوانده شود. سپس با توجه به شماره شناسه های بدست آمده، بردارهای موردنظر را از ماتریسهای پراکندگی خوانده و اشتراک آنها را بدست می‌آید. در این حالت اولین بیت دارای مقدار یک در بردار نتیجه، متناظر با قانونی است که باید به بستهٔ رسیده اعمال شود. جهت درک بهتر چگونگی انجام عمل دسته بندی توسط روش پیشنهادی و مرحله پیش پردازش،مثالی بر اساس تفکیک کننده شکل ۱ ارائه می شود. در مرحله اول، یعنی پیش پردازش از روی تفکیک کننده اصلی ماتریس محدوده را ایجاد می نماییم(شکل۶).

سپس با توجه به مرحله دوم بخش پیش پردازش، Trieهای هر بعد را رسم می کنیم،Trieهای ایجاد شده به ازای هر بعد تفکیک کننده در شکل۸ آمده است. همچنین با توجه به عملیات دو مرحله قبل، ماتریسهای پراکندگی هر بعد را محاسبه می کند. این ماتریسها در شکل ۱۰ آورده شده اند.

حال اگر بسته ای مانند P به مسیریاب برسد، ابتدا الگوریتم با توجه به ستون اول ماتریسهای پراکندگی، سطرهای متناظر را از ماتریسهای پراکندگی می خواند. که عمل یافتن سطر متناظر با توجه به سرآیند بسته می تواند توسط الگوریتمهای بزرگترین طول پیشوند متناظر انجام شود. سپس اشتراک سطرهای خوانده شده را محاسبه می کند. اولین بیت دارای مقداریک درماتریس نتیجه،متناظر با قانونی است که باید به این بسته اعمال شود.مثلا اگر بسته ای با سرآیند P(1100,1100) به مسیریاب برسد، با توجه به ستون اول ماتریسها، شماره سطرهایی که به ازای هر بعد باید خوانده شوند، عبارتند از:(5,5)

حال سطرهای متناظر را از ماتریسهای پراکندگی خوانده و اشتراک آنها را بدست می آوریم:

0 0 0 0 1 1 0

0 0 0 0 1 0 0

---------- AND

0 0 0 0 1 0 0

مشاهده می شود که بسته رسیده با قانون R5 در انطباق می باشد (بیت پنجم بردار نتیجه برابر یک است). و چون قوانین در تفکیک کننده ها همیشه به ترتیب اولویت مرتب می باشند. سمت چپ ترین یک در بردار نتیجه متناظر با شماره قانون مورد نظر می باشد.

حال میزان حافظه مصرفی روش PBBI و روش ماتریس بیتی را با توجه به تفکیک کننده شکل ۲ بدست می‌آوریم، همانگونه که در شکل ۳ مشاهده می شود، تعداد بازه‌های مجزای ایجاد شده برابر ۱۸ تا می باشد. درحالیکه تعداد پیشوندهای مجزا با توجه به ماتریس محدوده برابر ۱۲ تا می باشد. بنابراین روش پیشنهادی فقط به ۱۲ بردار ۶ بیتی و روش ماتریس بیتی به ۱۸ بردار ۶ بیتی نیاز دارد.

۳- ارزیابی الگوریتم PBBI

اگر تفکیک کننده شامل N قانون، هر قانون شامل K بعد، طول پیشوند برابر W و طول کلمات حافظه $Width$ بیت فرض شود، در اینصورت میزان حافظه مصرفی، تعداد عملیات مقایسه،تعداد دسترسی به حافظه وزمان پیش پردازش روش ارائه شده از قرار زیر می باشد:

چون ماتریس محدوده موقتی است پس نیازی به ذخیره ماتریس محدوده نداریم، بنابراین هیچ حافظه ای برای آن در نظر گرفته نمی شود و حافظه های مورد نیاز این روش، فقط حافظه مورد نیاز برای ذخیره Trieها و ماتریسهای پراکندگی می باشد. برای هر بعد یک Trie یک بعدی و یک ماتریس پراکندگی داریم که حافظه هر کدام، با فرض اینکه T پیشوند مجزا در هر بعد داشته باشیم طبق روابط ۴ و ۵ می باشد.

$$ST(T) = TK(W + Log(T)) \quad (۴)$$

$$SD(T) = TKN \quad (۵)$$

بنابراین میزان کل حافظه مصرفی الگوریتم برابر است با مجموع حافظه های مورد نیاز برای ذخیره Trieها و ماتریسهای پراکندگی که دراینصورت کل حافظه طبق رابطه ۶ خواهد بود.

$$S(T) = TK(N + W + Log(T)) \quad (۶)$$

در بدترین حالت چون T=N می باشد ،بنابراین میزان حافظه مصرفی روش پیشنهادی از قرار زیر خواهد بود.

در این حالت حافظه مورد نیاز برای Trieها برابر خواهد بود با:

$$ST(N) = NK(W + Log(N)) \quad (۷)$$

همچنین حافظه مورد نیاز برای ذخیره ماتریسهای پراکندگی طبق رابطه ۸ خواهد بود.

$$SD(N) = KN^2 \quad (۸)$$

بنابراین کل حافظه مصرفی از قرار زیر می باشد.

$$S(N) = NK(N + W + Log(N)) \quad (۹)$$

اما زمان جستجوی روش PBBI برابر است با تعداد عملیات مقایسه لازم جهت یافتن بزرگترین طول پیشوند در هر بعد، به علاوه تعداد عملیات دسترسی به حافظه که به ازای هر بعد باید انجام دهیم.

تعداد عملیات مقایسه برابر است با:

$$TC(T) = K * Log(W) \quad (۱۰)$$

همچنین تعداد عملیات دسترسی به حافظه طبق رابطه ۸ خواهد بود.

$$TMA(T) = K\frac{N}{Width} \quad (۱۱)$$

پیچیدگی زمان پیش پردازش الگوریتم ترکیبی از چند زمان مختلف خواهد بود، ابتدا باید از روی تفکیک کنندهٔ اولیه ماتریس محدوده را تشکیل دهیم که زمان این مرحله طبق شبه کد داده شده در شکل ۵ بصورت زیر خواهد بود.

$$TPR(N) = dN^2 \quad (۱۲)$$

زمان لازم برای ایجاد Trieها از روی پیشوندها و ماتریس محدودهٔ ایجاد شده در مرحلهٔ قبل نیز طبق رابطه ۱۳ محاسبه می‌شود.

$$TPTrie(N) = dwN \quad (۱۳)$$

همچنین زمان ایجاد ماتریسهای پراکندگی از روی تفکیک کننده، ماتریس محدوده وTrieها طبق رابطه ۱۴ بدست می آید.

$$TPD(N) = dN^2 \quad (۱۴)$$

می توان گفت که پیچیدگی زمان پیش پردازش دو روش در بدترین حالت تقریباً یکسان می باشد، اما با توجه به اینکه الگوریتم پیشنهادی براساس خواص تفکیک کننده های واقعی ارائه شده است، باید با توجه به این ویژگی این تفکیک‌کننده ها مورد بررسی قرار گیرد. در روش پیشنهادی مرحلهٔ ایجاد درختها چندان زمانبر نمی باشد، بلکه مرحلهٔ ایجاد ماتریسهای پراکندگی دارای زمان اجرای بیشتری می باشد که اگرT پیشوند مجزا داشته باشیم بایدT بردار N بیتی ایجاد نموده و آنها را مقدار دهی نماییم. همچنین مرحله زمانبر در روش ماتریس بیتی مقداردهی بردارهای N بیتی متناظر با بازه های ایجاد شده می‌باشد که در بدترین حالت $2N+1$ بازه مجزای ایجاد شده داریم. اما اگر فرض کنیم که تعداد بازه های مجزا برابر با L باشد، در تفکیک کننده های واقعی $T << L$ [۹]. در نتیجه می توان گفت که در روش ماتریس بیتی باید L بردار و در روش پیشنهادی باید T بردار را مقدار دهی نماییم. همچنین می توان زمان پیش پردازش را در هر کدام از دو روش متناظر با مقادیر L و T دانست.

پارامترهای مورد مقایسه دو روش در جدول ۲ آمده است که پارامترهای روش اول از مرجع[۱۷] گرفته شده است.

با توجه به جدول ۲ می بینیم که تعداد عمل دسترسی به حافظه دو روش یکسان می‌باشد ،اما تعداد عمل مقایسه دو روش جهت یافتن بردارهای متناظر برابر نیست و روش *PBBI* به تعداد عمل مقایسه کمتری نیاز دارد.

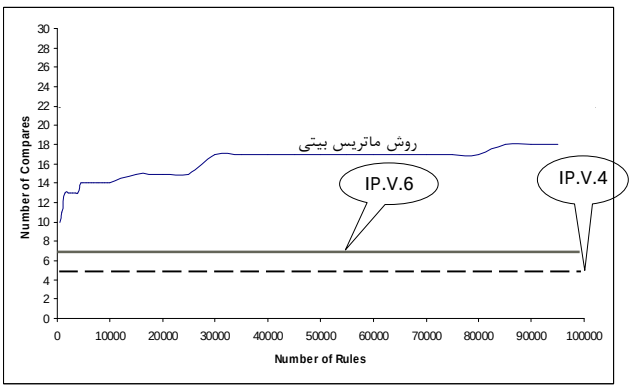
خاصیت مهم دیگر روش PBBI که یکی از مزایای عمده آن نیز می‌باشد، این است که در این روش تعداد عملیات مقایسه از تعداد قوانین تفکیک کننده مستقل می‌باشد.

جدول۲- مقایسهٔ پارامترهای کارایی روش پیشنهادی و روش ماتریس بیتی

روش	حافظهٔ مصرفی	تعداد عمل مقایسه	تعداد دسترسی به حافظه	زمان پیش پردازش
ماتریس بیتی	$K(2N+1)(N+2W)$	$K * Log_2^{2*N+1}$	$\frac{N}{Width}$	L
روش PBBI	$K\ T(N+W+Log(T))$	$K*Log(W)$	$\frac{N}{Width}$	T

این تعداد برای *IPV.4* فقط ۵ عمل مقایسه و برای *IPV.6* به ۷ عمل مقایسه می‌باشد. درحالیکه تعداد عملیات مورد نیاز روش ماتریس بیتی با افزایش تعداد قوانین افزایش می یابد[۱۷].

مقایسه دو روش از لحاظ تعداد اعمال مقایسه با استفاده از روابط موجود در جدول ۲ ، در نمودار شکل ۱۱ آورده شده است.

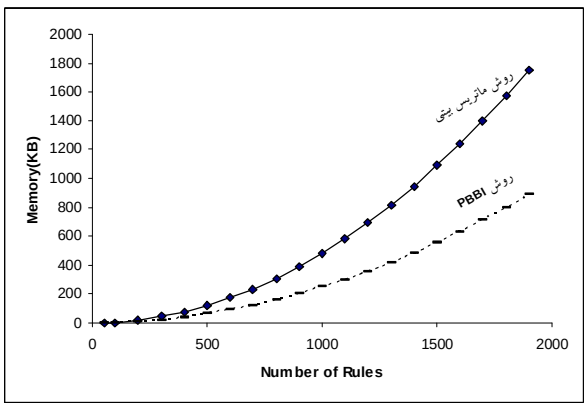


شکل۱۱- تعداد عملیات مقایسه روش ماتریس بیتی و روش PBBI

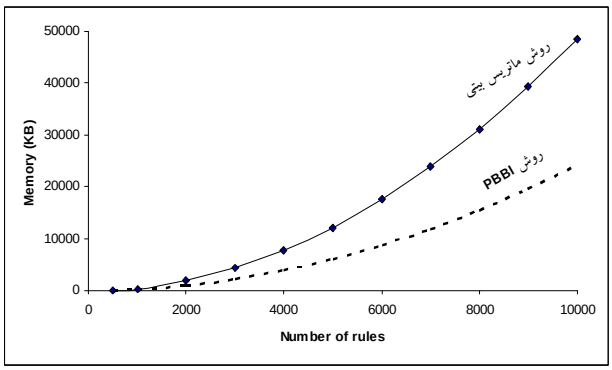
۴- شبیه سازی

در ادامه به منظور ارزیابی کارایی روش PBBI شبیه سازی انجام شده است. نحوه شبیه سازی به این صورت است که در ابتدا جدولهای تفکیک کننده در قالب فایلهای متنی تولید شده و سپس الگوریتم های ماتریس بیتی و PBBI که با زبان برنامه نویسی C++ و در محیط *Borland C++ Builder 6* پیاده‌سازی شده‌اند، به این جداول اعمال می‌گردند. جداولیکه برای شبیه‌سازی در این مقاله مورد استفاده قرار گرفتند، دو نوع می باشند: نوع اول که تفکیک کننده‌های واقعی می باشند و دارای کمتر از ۲۰۰۰ قانون می‌باشند، از آدرس اینترنتی مرجع [۲۰] گرفته شده اند. این تفکیک کننده‌ها در ادامه متن تحت عنوان تفکیک کننده‌های کوچک مورد ارجاع واقع شده‌اند. نوع دوم که در این مقاله تحت عنوان تفکیک کننده‌های بزرگ مورد ارجاع واقع می‌شوند، به صورت آزمایشگاهی و بر اساس خواص تفکیک کننده‌های واقعی که از مشاهدات تجربی مراجع [۲۰،۲۱] استخراج شده‌اند، تولید می‌شوند. این تفکیک کننده دارای کمتر از ۱۰۰۰۰ قانون می باشند. . نتایج حاصل از شبیه‌سازی دو روش به ازای تفکیک کننده‌های فوق حاکی از این است که روش PBBI در مورد هر تفکیک کننده، به ازای قوانین یکسان خروجی‌های یکسانی با خروجی‌های روش ماتریس بیتی تولید می‌نماید و به عبارت دیگر همانگونه که از سیر منطقی مراحل الگوریتم از قبل نیز قابل پیش بینی بود، الگوریتم PBBI دسته‌بندی بسته‌ها را به درستی انجام می‌دهد. برای محاسبهٔ میزان حافظهٔ مصرفی دو روش به ازای تفکیک کننده های موجود ، هر روش تفکیک کننده را در حافظه بار نموده و ساختارهای مورد نیاز جهت انجام عمل دسته بنـــدی را با توجـــه به تفکیک کننده ایجـــاد می نمایـد، سپس میزان حافظهٔ مورد نیاز ساختارها محاسبه می شود. از این عدد بعنوان میزان حافظهٔ مصرفی روش مورد نظر در شبیه سازی استفاده کرده ایم.

همانگونه که در شکل ۱۲ مشاهده می شود، میزان حافظهٔ مصرفی روش پیشنهادی نسبت به روش ماتریس بیتی کمتر می باشد که این نمودارها تایید کنندهٔ روابط بدست آمده در بخش ارزیابی الگوریتم می باشند.



(الف)

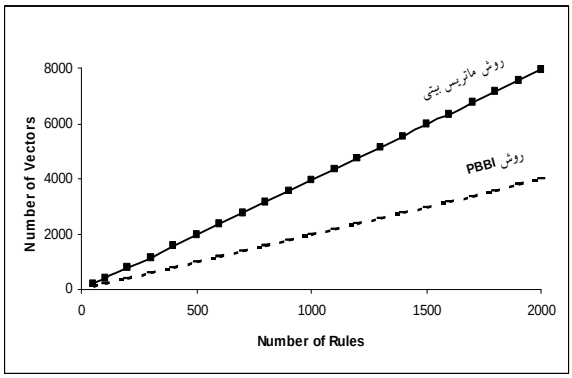


(ب)

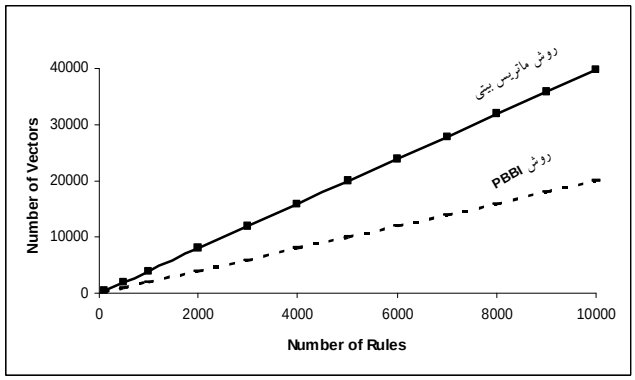
شکل۱۲- مقایسهٔحافظهٔ مصرفی

(الف) به ازای تفکیک کننده‌های کوچک (ب) به ازای تفکیک کننده‌های بزرگ

برای محاسبهٔ زمان پیش پردازش دو روش با توجه به اینکه در مرحلهٔ پیش پردازش زمانبرترین عمل، ایجاد ومقداردهی بردارهای ایجاد شده به ازای هر بعد می باشند. می توانیم فرض نماییم که زمان پیش پردازش متناظر با تعداد بردارها می باشد. از اینرو در مرحلهٔ شبیه سازی به ازای هر تفکیک کننده تعداد بردارهای ایجاد شده توسط هر روش را محاسبه نموده و از آن بعنوان پارامتری برای مقایسهٔ زمان پیش پردازش دو روش استفاده نموده‌ایم، که نمودارهای حاصل از این کار، در شکل۱۳ آمده است. مشاهده می شود که با افزایش تعداد قوانین تفکیک کننده ها، میزان بهبـــود زمـان پیش پـــردازش روش PBBI نسبت به روش ماتریس بیتی



(الف)



(ب)

شکل۱۳- مقایسه زمان پیش پردازش (الف)به ازای تفکیک کننده های کوچک (ب) به ازای تفکیک کننده های بزرگ

افزایش می یابد که این مطلب نیز بیانگر صحت روابط بدست آمده در بخش ارزیابی است.

میزان صرفه جویی در حافظه روش پیشنهادی نسبت به ماتریس بیتی قابل توجه می باشد،که با حذف قوانین افزونه در تفکیک کننده ها باز هم می توان میزان صرفه جویی را بهبود بخشید.[۵،۲۳]

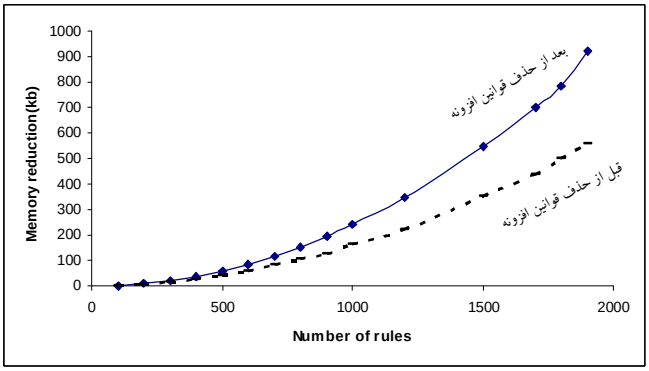
نمودارهای میزان صرفه جویی در حافظه روش پیشنهادی قبل از حذف قوانین افزونه و بعد از حذف این قوانین در شکل ۱۴ آمده است.

با مقایسه نمودارها می بینیم که با حذف قوانین افزونه می تـوان حافظـهٔ مصـرفی روش پیشنهادی را بهبود بخشید. بعنوان مثال در حالت قبلی برای تفکیک کننده ای با ۱۹۰۰ قانون میزان صرفه جویی در حافظه روش پیشنهادی برابر ۶۴۰ کیلـو بایت می‌باشد،درحالیکه با حذف قوانین افزونه میزان صرفه جـویی در حافظـه بـه ۹۷۰کیلو بایت می رسد.

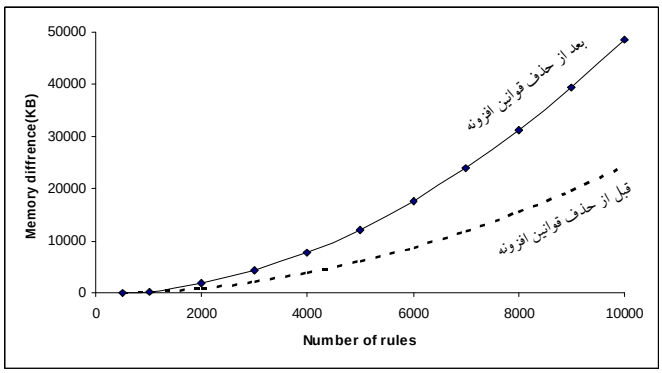
همانطور که از نتایج ارزیابی کارایی الگوریتم PBBI که به وسیله شبیه‌سازی هم تأیید شده است، استنباط میشود، این الگوریتم چه به لحاظ حافظه مصرفی و چه زمان دسته بندی کارایی مناسبتری از الگوریتم ماتریس بیتی دارد. ولی همانطور که از مقایسه جداول ۱و ۲ میتوان مشاهده کرد، این الگوریتم نسبت به بعضی از روشهای دیگر مانند برشهای سلسله مراتبی هوشمند و دسته بندی با جریان بازگشتی زمان دسته‌بندی بیشتری دارد که این امر نقطه ضعف روش پیشنهادی است.

۵- نتیجه گیری

با توجه به رشد شبکهٔ اینترنت و مطرح شدن کیفیت سرویسهای متفاوتی که شبکه ها باید به کاربران ارائه دهند، عمل دسته بندی کردن بسته ها به عنوان یکی از عملیات مهم و حساس شبکه، هر روز ابعاد تازه ای به خود می گیرد.



(الف)



(ب)

شکل۱۴- مقایسه میزان صرفه‌جویی در حافظه (الف)تفکیک کننده کوچک (ب) تفکیک کننده بزرگ

به گونه ای که می‌توان گفت برای اغلب سرویسهایی که شبکه ها ارائه می دهند، مسیریابها نیاز دارند که عمل دسته بندی را با سرعت بالا و میزان حافظه مصرفی کم انجام دهند. از اینرو ارائهٔ روشی که بتواند با سرعت مناسب و میزان حافظه مصرفی کم، عمل دسته بندی را انجام دهد و دارای پیاده سازی نرم افزاری و سخت افزاری راحتی باشد، ضروری به نظر می رسد. اغلب روشهای ارائه شده برای دسته بندی بسته ها سعی دارند تا حد امکان اکثر خصوصیات فوق را دارا باشند، اما هر روش توانسته است فقط بعضی از این خواص فوق را برآورده نماید.

روش ماتریس بیتی یکی از روشهایی است که اغلب خواص فوق را دارا بوده، اما دارای میزان حافظه مصرفی بالایی می‌باشد.که اگر میزان حافظه مصرفی آن را بتوان کاهش داد، می تواند از بقیه روشها موثرتر و مناسب تر باشد. در این مقاله روشی جدید تحت عنوان PBBI ، بر اساس خواص تفکیک کننده های واقعی ارائه شد که میزان حافظه و زمان دسته بندی روش ماتریس بیتی را تا حد زیادی کاهش داد.با توجه به نتایج حاصل از شبیه سازی می توان گفت که روش PBBI در ازای ۱۹۰۰ قانون ۶۴۰ کیلو بایت و در ازای ۱۰۰۰ قانون ۱۴ مگابایت در میزان حافظه مصرفی صرفه جویی می کندزمان دسته بندی روش PBBI نسبت به روش ماتریس بیتی نیز کمتر می باشد، بگونه ای که تعداد عملیات مقایسه مورد نیاز روش پیشنهادی برای انجام عمل دسته بندی، از تعداد قوانین تفکیک کننده مستقل بوده و فقط وابسته به طول پیشوندهای فیلدهای سرآیند بسته های رسیده می باشد. در این روش تعداد عملیات مقایسه برابر با ***Log(W)*** می باشد که در آن Wطول پیشوند است.بنابراین تعداد عملیات مقایسه روش پیشنهادی به ازای IP.V.4 برابر ۵ و برای IP.V.6 برابر ۷ خواهد بود. در حالیکه در روش ماتریس بیتی تعداد عملیات مقایسه با افزایش تعداد قوانین تفکیک کننده، افزایش می یابد،همچنین نتایج حاصل از شبیه سازی حاکی از زمان پیش پردازش مناسب الگوریتم PBBI نسبت به روش ماتریس بیتی است.روش PBBI با اینکه میزان حافظهٔ مصرفی کمتری نسبت به اغلب روشهای دسته بندی ارائه می دهد اما زمان

دسته بندی آن از بعضی روشهای مطرح دسته بندی بیشتر می باشد که این خود یکی از معایب روش پیشنهادی می باشد.

میزان حافظه مصرفی روش پیشنهادی با بقیهٔ روشهای مطرح دسته بندی بسته ها مانند برشهای سلسله مراتبی هوشمند و دسته بندی با جریان بازگشتی نیز قابل مقایسه می باشد،بعنوان مثال روش برشهای سلسله مراتبی هوشمند در ازای ۱۷۳۳ قانون به ۱/۴ مگابایت حافظه نیاز دارد،در حالیکه میزان حافظهٔ مصرفی روش پیشنهادی برابر با ۷۰۰ کیلوبایت می باشد، همچنین میزان حافظهٔ روش PBBI از حافظهٔ مصرفی روش دسته بندی با جریان بازگشتی بسیار پایین تر می باشد اما در عوض این روش نسبت به روشهای فوق دارای زمان دسته بندی ضعیفتری می باشد.

در نتیجه می توان گفت این روش درمقابل افزایش تعداد قوانین، دارای زمان دسته بندی، میزان حافظه مصرفی و زمان پیش پردازش بهتری نسبت به روش ماتریس بیتی بوده و در تفکیک کننده های بزرگ نیز بخوبی پاسخگو می‌باشد. به عبارت دیگر این روش در کلیه پارامترهای مورد نیاز روشهای دسته بندی، دارای مقیاس‌پذیری و کارایی بالایی است.

مراجع

[1] X. Sun," A Tutorial and Review About: IP Address Lookups and Packet Classification," School of Mathematics and Statistics Carleton University, Ottawa, Ontario CANADA K1S 5B6.

[2] A. Feldman and S. Muthukrishnan, "Tradeoffs for Packet Classification," *Proc. INFOCOM*, vol. 3, pp. 1193-1202, 2000.

[3] P. Gupta and N. McKeown, "Algorithms for Packet Classification," *IEEE Network Magazine*, 2001.

[4] T. Y.C. Woo, "A Modular Approach to Packet Classification: Algorithms and Results," Bell Laboratories, *Lucent technologies*, 2000.

[5] P. Gupta, *Algorithms for Lookups and Packet Classification* PhD dissertation, Computer Science Department, Stanford university, 2000.

[6] V. Srinivasan, "Fast and Scalable Layer four switching," *Proc. ACM Sigcomm*, pp. 203–14, *1998.*

[7] F. Baboescu and G. Varghese, "Aggregated bit vector search algorithms for packet filter lookups," *UCSD Tech. Report, cs2001-0673, 2001.*

[8] F. Baboescu and G. Varghese, "Scalable Packet Classification," *Proc. Of SIGCOM'01*, San Diego, California, USA, 2001.

[9] P. Gupta and N. McKeown, "Packet Classification on Multiple Fields," *Proc. Sigcomm, Comp. Commun.* Rev., vol. 29, no. 4, pp. 147–60, 1999.

[10] P. Gupta and N. McKeown, "Packet Classification Using Hierarchical Intelligent Cutting," *Proc. Hot Interconnects VII*, Stanford, 1999.

[11] F. Baboescu, S. Singh, G. Varghese and J. Wang," Packet Classification Using Multidimensional Cutting," *UCSD Technical Report CS2003-0736, 2003.*

[12] V. Srinivasan, G. Varghese, and S. Suri, "Packet Classification using Tuple Space Search," *Proceedings of ACM sigcomm'99*, vol. (August), 1999.

[13] D. Shah and P. Gupta, "Fast Update on Ternary-CAMs for Packet Lookups and Classification," *Proc. Hot Interconnects VIII*, Stanford Uni, 2000.

[14] M. Waldvogel, G. Varghese, J. Turner and B. Plattner, "Scalable High Speed IP Routing Table Lookups," *Proc. Of ACM SIGCOMM'97*, 1997.

[15] N. F. Huang et al., "A Fast IP Lookup Schema for Gigabit Switching Routers," *Proc. INFOCOMM'99*, 1999.

[16] M. A. Ruiz-Sanchex et al., "Survey and Taxonomy of IP Address Lookup Algorithms," *IEEE Network*, vol. 15, no. 2, pp. 8-23, 2001.

[17] T. Lakshman and D.Stidialis, "High Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching," *Proc.ACMSigcomm'98*, 1998.

[18] J. Li, H. Liu and K. Sollins, " AFBV: a Scalable Packet Classification Algorithm" , *ACM SIGCOMM Computer Communication Review,* vol. 32.

[19] W. T. Chen and J. L. Chaing, "A Two-Stage Packet Classification Algorithm," *Proc. Of ANIA'03, 2003.*

[20] Prefix database MaeEast, The Internet Performance Measurement and Analysis (IPMA) project, data vailable at http://www.merit.edu/ipma/routing_table/.

[21] S. Singh and F. Baboescu, "Packet classification repository,"[Online].Available: http://ial.ucsd.edu/classification

[22] M. Waldvogel, *Fast Longest Prefix Matching: Algorithms, Analysis, Application,* Ph.D. dissertation, Computer Science Department, Washington university, 2002.

[23] P. Gupta, "Routing Lookup and Packet Classification: Theory and Practice," Tutorial at Hot Interconnects VIII, Stanford, CA, [Online]. Available: http://klamath.stanford.edu/~pankaj/tallks/, 2000.

[24] م. فتحی، ن. م. بلوچ زهی، "ارائه روشی برای دسته بندی بسته ها بر اساس ماتریسهای بیتی"، **مجموعه مقالات نهمین کنفرانس انجمن کامپیوتر ایران**، ص ۱۰۳-۱۰۹ ، ۱۳۸۲.

[1] Prefix-Based Bit Map Intersection
[2] Classifier
[3] Linear search

[4] Hierarchical Tries
[5] Set Pruning Tries
[6] Grid of Tries
[7]Recursive Flow Classification
[8]Hierarchical Intelligent Cuttings
[9] Tuple space Search
[10] Hash Table
[11] Content Addressable memory
[12]Bit Vector(Bitmap Intersection)
[13]Aggregated Bit Vector
[14]Aggregating and Folding Bit Vector
[15] Prefix-Based Bitmap Intersection
[16] Longest Prefix Matching

نیک محمّد بلوچ زهی کارشناس ارشد مهندسی کامپیوتر در گرایش معماری سیستمهای کامپیوتری از دانشگاه علم و صنعت ایران می باشد، ایشان مدرک کارشناسی مهندسی کامپیوتر گرایش نرم افزار را از دانشگاه یزد درسال ۱۳۸۰ اخذ نمود است. وی هم اکنون عضو هیئت علمی گروه مهندسی فنآوری اطلاعات دانشگاه سیستان و بلوچستان است. زمینه‌های تحقیقاتی مورد علاقه نامبرده دسته بندی بسته‌ها و IP LOOKUP می‌باشند.

محمود فتحی درجه دکتری خود را در سال ۱۹۹۰ میلادی در زمینه طراحی پردازنده‌های RISC برای عملیات پردازش تصویر مورفولوژی از دانشگاه UMIST در انگلستان اخذ کرده است. وی از همین سال عضو هیئت علمی دانشکده کامپیوتر دانشگاه علم و صنعت ایران شد و هم اکنون با مرتبه دانشیاری مدیر گروه سخت‌افزار این دانشکده می‌باشد. دکتر فتحی تا کنون ۱۴۰ مقاله در کنفرانسها و مجلات داخلی و بین المللی به چاپ رسانده است. از نامبرده چهار کتاب در زمینه معماری کامپیوتر و شبکه‌های کامپیوتری به چاپ رسیده است. زمینه‌های مورد علاقه ایشان، پردازش تصویر، کیفیت سرویس در شبکه‌های کامپیوتری و طراحی سیستم‌های کامپیوتری با کارایی بالا می‌باشند.

صالح یوسفی فارغ التحصیل کارشناسی و کارشناسی ارشد مهندسی کامپیوتر از دانشگاه علم وصنعت ایران در گرایشهای سخت افزار و معماری سیستم‌های کامپیوتری به ترتیب در سالهای ۱۳۷۹ و ۱۳۸۱ می‌باشد. وی هم اکنون دانشجوی دکتری همان دانشکده است. زمینه‌های تحقیقاتی ایشان کیفیت سرویس و انتقال مالتی مدیا در شبکه های کامپیوتری سیمی و بی سیم می‌باشند.