مقایسه کارایی الگوریتم‌های H-trie و AQT در دسته‌بندی بسته‌ها

مهدی عباسی[۱]

[۱] استادیار گروه مهندسی کامپیوتر، دانشکده مهندسی، دانشگاه بوعلی‌سینا، همدان، abbasi@basu.ac.ir

چکیده

از وظایف اصلی پردازنده‌های شبکه‌ای دسته‌بندی بسته‌ها می‌باشد. مهمترین مسئله در این زمینه، استفاده از الگوریتمی است که بتواند بسته‌ها را با سرعتی در حد سرعت انتقال اطلاعات در شبکه دسته‌بندی کند. این الگوریتم باید در مصرف حافظه هم به صورت بهینه عمل کند. الگوریتم‌های دسته‌بندی به دو رده‌ی کلی نرم‌افزاری و سخت‌افزاری تقسیم می‌شوند. روش‌های نرم‌افزاری علیرغم قابلیت توسعه‌پذیری و سفارشی‌سازی دارای سرعت پایین‌تری نسبت به روش‌های سخت‌افزاری هستند. در میان الگوریتم‌های دسته‌بندی بسته‌ها، الگوریتم‌های جستجوی درختی تعادل مناسبی میان سرعت دسته‌بندی و حافظه مورد نیاز برقرار کرده‌اند. با این وجود انتخاب الگوریتم مناسب از دیدگاه حافظه مصرفی یا سرعت دسته-بندی نیازمند ارزیابی و مقایسه کارایی با استفاده از تستهای استاندارد است. در این مقاله، دو الگوریتم درختی H-trieو AQT پیاده‌سازی شده و کارایی آنها از دیدگاه زمان دسته‌بندی بسته‌ها و میزان حافظه مصرفی مورد مقایسه قرار گرفته است. نتایج نشان می‌دهد، هرچند الگوریتم AQT از لحاظ سرعت دسته‌بندی بسته‌ها بهتر از H-trie است حافظه بیشتری نیز برای پیاده سازی به ویژه در دسته‌بندهای بزرگتر لازم دارد.

واژه‌های کلیدی

دسته‌بندی بسته‌ها، الگوریتم‌های درختی، مقایسه کارایی، **AQT**، **H-trie**.

۱- مقدمه

برای استفاده از حداکثر کارایی شبکه، مسیریاب‌ها و سوئیچ‌ها باید بتوانند بسته‌ها را با سرعت شبکه پردازش نمایند[۱]. در نتیجه مسیریاب‌ها به مکانیزم جدیدی به نام دسته‌بندی بسته مجهز شده‌اند. فرآیند طبقه‌بندی بسته در شبکه به جریان‌های مختلف، در مسیریاب‌های اینترنت و سوئیچ‌ها را دسته‌بندی بسته‌ها می‌نامند[۲-۶]. الگوریتم‌های دسته‌بندی بسته، در تمام تجهیزات شبکه که بر روی بسته‌ها پردازش‌های خاص را با سرعت بالاتری انجام می‌دهند، مورد نیازند. این نیاز، به ویژه در تجهیزات پردازشگر مورد استفاده در شاهراه‌های اینترنت همچون مسیریاب‌ها اهمیت ویژه‌ای می‌یابد.

عمل جداسازی بسته‌ها به جریان‌ها بر اساس فیلترهایی که در مسیریاب‌ها وجود دارد، انجام می‌شود. این فیلترها درجدولی به نام دسته‌بند، به ترتیب اولویت قرار می‌گیرند[۲-۶]. نحوه دسته‌بندی به این صورت است که با دریافت یک بسته، فیلدهای موجود در سرآیند بسته با فیلترهای موجود در دسته‌بند مقایسه می‌شوند. سپس، از میان فیلترهای منطبق، فیلتر با اولویت بیشتر انتخاب می‌شود. براساس فیلتر انتخابی، بسته در جریان متناظر با فیلتر قرار می‌گیرد. به عبارت دیگر، جریان‌ها براساس مشخصه‌هایی مانند آدرس مبدا، آدرس مقصد، شماره پورت مبدا، شماره پورت مقصد و نوع پروتکل از همدیگر متمایز می‌شوند و با هر کدام براساس فیلترهای تعریف شده برخورد می‌شود[۳-۶].

الگوریتم‌های دسته‌بندی بسته‌ها در حالت کلی به روشهای سخت‌افزاری و نرم‌افزاری پیاده‌سازی می‌شوند. روش‌های سخت‌افزاری با وجود داشتن سرعت بالا دارای مشکلاتی نیز می‌باشند. از جمله این مشکلات می‌توان به قیمت بالا، مصرف توان بالا، غیر قابل گسترش بودن و ناکارآمدی در استفاده بهینه از فضای ذخیره‌سازی اشاره کرد. با توجه به مشکلات گفته شده در دسته‌بندهای سخت‌افزاری و به دلیل قابلیت توسعه‌پذیری و سفارشی‌سازی دسته‌بندهای نرم‌افزاری، این گونه از دسته‌بندها مورد توجه قرار گرفته‌اند. تعداد زیادی از الگوریتم‌های دسته‌بند نرم‌افزاری در سال‌های اخیر ارائه شده‌است. Taylor این الگوریتم‌ها را در چهار کلاس جستجوی خطی[۱]، تجزیه[۲]، درخت تصمیم‌گیری[۳] و فضای چندتایی[۴] طبقه‌بندی کرده است[۶]. روش‌های ارائه شده، هر یک با توجه به زمان دسته بندی و میزان حافظه مصرفی، برای کاربرد خاصی مناسب می‌باشد. اغلب این روش‌ها نتوانسته‌اند مصالحه مناسبی بین زمان و میزان حافظه مصرفی ایجاد

نمایند[۶]. الگوریتم‌های درختی به خوبی توانسته‌اند که بین مصرف حافظه و سرعت جستجو یک تعادل مناسب برقرار کنند. با این وجود، انتخاب الگوریتم دسته‌بندی درختی مناسب با توجه به نیازمندیهای حافظه و سرعت دسته‌بندی یک چالش اساسی است[۲, ۷-۱۰].

در ادامه مقاله، در بخش‌های دوم و سوم، ابتدا نحوه دسته‌بندی بسته‌ها توسط دو الگوریتم AQT و H-trie بیان می‌شود. در بخش چهارم، پس از معرفی معیارهای ارزیابی کارایی الگوریتم‌های دسته‌بندی بسته-ها، نحوه پیاده‌سازی و ارزیابی آنها توضیح داده شده و نتایج اجرای آنها تحلیل و با هم مقایسه می‌شود. بخش نهایی مقاله به نتیجه گیری از مقاله و ارائه پیشنهادها اختصاص داده شده است.

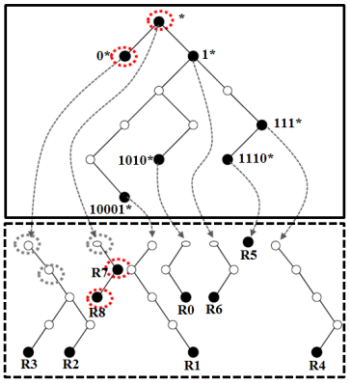
۲. الگوریتم درختی سلسله مراتبی (H-trie)

یکی از الگوریتم‌های مبتنی بر درخت تصمیم، روش درخت سلسله مراتبی است[۴-۶]. در الگوریتم درختی سلسله مراتبی، برای دسته-بندی بسته‌ها از آدرس IP مبدأ و آدرس IP مقصد برای ساختن درخت تصمیم‌گیری استفاده می‌شود. بعنوان یک مثال، از درخت تشکیل یافته شکل (۱) براساس ۹ فیلتر موجود در جدول (۱) می‌توان استفاده نمود. نحوه تشکیل درخت بدین صورت می‌باشد که ابتدا آدرس IP مبدأ فیلترها به صورت بیت به بیت خوانده می‌شود؛ سپس، با مشاهده هر بیت "۰" یا "۱" به ترتیب سمت چپ یا راست درخت مورد پیمایش قرار می‌گیرد. وقتی که علامت * [۵] دیده شد یعنی تکمیل درخت براساس آدرس IP مبدأ آن قانون به پایان رسیده است. بنابراین، با استفاده از یک اشاره‌گر برای ساختن ادامه درخت از آدرس IP مقصد استفاده می‌شود. نحوه ساختن درخت توسط آدرس IP مقصد دقیقا شبیه به نحوه ساختن درخت توسط آدرس IP مبدأ است[۳].

جدول ۱: مجموعه قوانین نمونه برای درخت H-trie

Rule	Src. IP	Dest. IP	Src. Port	Dest. Port	Prtl
Rule 0	1010*	01*	0, 65535	25	17
Rule 1	10001*	0111*	53	443	4
Rule 2	0*	1110*	53	1024,65535	6
Rule 3	0*	1100*	53	25	17
Rule 4	111*	1110*	53	443	4
Rule 5	1110*	*	0, 65535	2788	4
Rule 6	1*	10*	53	5632	6
Rule 7	*	1*	53	25	17
Rule 8	*	10*	0.65535	2788	6

دسته‌بندی بسته بدین صورت است که با دریافت یک بسته ورودی ابتدا آدرسهای IP مبداو مقصد، آدرس درگاه مبدا و مقصد و پروتکل از بسته استخراج می شود. مثلاً(6 ,2788 ,10000,542 ,00000) ابتدا پیمایش از ریشه درخت براساس بیتهای فیلد آدرس مبدا شروع می‌شود. در مسیر پیمایش اگر گره‌ای اشاره‌گر به درخت آدرس Ip مقصد داشت، درون یک صف قرار می‌گیرد؛ با اتمام پیمایش درخت آدرس Ip مبدا، درخت‌های متناظر با گره‌های موجود در صف، براساس



شکل ۱: ساختار درخت سلسله مراتبی جدول ۱

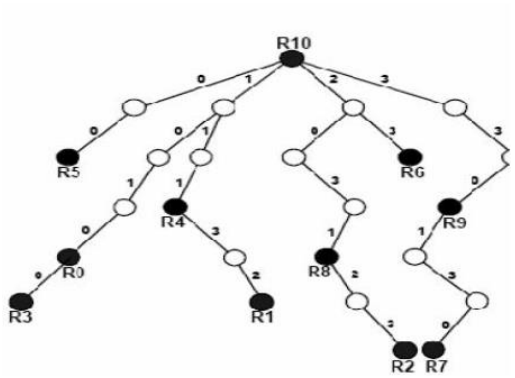
بیتهای فیلد آدرس مقصد بسته پیمایش می‌شوند. در مسیر پیمایش تمام فیلترهای موجود در مسیر ذخیره می‌شوند و سپس فیلترهای ذخیره شده به منظور انتخاب بهترین فیلتر انطباقی به صورت خطی جستجو می‌شوند. در این مثال فیلتر انطباق یافته R8 می‌باشد. پیچیدگی زمانی الگوریتم درخت سلسه مراتبیبرابر با $O(W^d)$ است که W حداکثر طول پیشوند هر فیلد و d تعداد فیلدهای مورد بررسی می‌باشند. همچنین پیچیدگی ذخیره‌سازی درخت$O(NdW)$ است که N تعداد فیلترهای دسته‌بندی‌کننده می‌باشد[۳].

۳. الگوریتم درخت چهارتایی مبتنی بر ناحیه (AQT)

الگوریتم AQT، یک درخت بر اساس آدرس‌های مبدا و مقصد تشکیل می‌دهد. در این درخت هر گره می‌تواند چهار فرزند داشته باشد[۴-۶]. به عنوان مثال، برای قوانین جدول۲، درخت AQT متناظر در شکل۲ نشان داده شده است.

جدول ۲: مجموعه قوانین نمونه برای درخت AQT

Rule no.	Src prefix	Dst prefix	Src port (start, end)	Dst port (start, end)	Prtl
R0	0000*	1010*	53, 53	443, 443	17
R1	000111*	11110*	53, 53	25, 25	6
R2	101011*	001101*	53, 53	25,25	17
R3	00000*	10100*	67, 67	5632	6
R4	000*	1110*	1024	1024	6
R5	0011*	00*	53, 53	25, 25	4
R6	110*	01*	0, 65535	5632	6
R7	110010*	110110*	0, 65535	5632	6
R8	1010*	00110*	53, 53	25, 25	6
R9	11010*	110*	0, 15576	2783	4
R10	*	110*	*	*	*



شکل۲: درخت AQT متناظر با قوانین جدول۲

در جدول۲، قانون‌ها از فیلدهایی تشکیل شده‌اند که در شش ستون نمایش داده شده است. این فیلدها، به ترتیب از چپ به راست عبارتند از: شماره قانون، آدرس مبدا، آدرس مقصد، آدرس پورت مبدا، آدرس پورت مقصد و پروتکل. لازم به ذکر است با توجه به اینکه نحوه تعریف پورت به صورت محدوده می‌باشد، عدد اول در ستون‌های چهار و پنج از نشان دهنده ابتدای دامنه و عدد دوم نشان دهنده انتهای دامنه است.

درخت AQT متناظر با قوانین جدول ۲ در شکل ۲ نمایش داده شده است. از آدرس‌های مبدا و مقصد برای تشکیل درخت استفاده می‌نماییم. با توجه به اینکه درخت چهارتایی هر گره می‌تواند چهار فرزند داشته باشد برای یافتن زیر درخت مناسب برای درج قانون‌ها همانند به شرح زیر عمل می‌کنیم. از آدرس آی پی مبدا و مقصدا دو بیت متناظر را برداشته و شماره دودویی حاصل را به عنوان شماره شخه‌ای از درخت که قانون باید در آن درج شود استفاده می‌کنیم. این روند را به ازاء سایر بیتهای پیشوند آدرس IP مبدا و مقصد ادامه می دهیم تا قانون در گره مناسب درج شود[۴-۶].

۴. پیاده سازی و ارزیابی

برای ارزیابی دو الگوریتم AQT و H-Trie دو کد به زبان C# طراحی و پیاده سازی شد. کدهای مذکور روی سیستمی با پردازنده Intel 2.30 GHz و حافظه 4GB اجرا شد.

از آنجا که برای اجرای دو الگوریتم مورد نظر نیاز به بسته‌هایی با سرآیندهای ساختگی* بود از ابزار Classbench در این تحقیق استفاده شد. این ابزار علاوه بر تولید بسته‌هایی با سرآیندهای ساختگی، قوانینی متناظر با توزیع و پراکندگی بسته‌ها مذکور تولید می‌کند[۹].

*Synthetic

مسلما، مقایسه کارایی الگوریتم‌های دسته‌بندی بسته‌ها نیاز به معیارهای استانداردی دارد. چنین معیارهایی از سوی محققان معرفی و در تحقیقات مرتبط استفاده شده است. در ادامه به معرفی این معیارها و اهمیت آنها پرداخته می‌شود.

۴-۱. تعداد دسترسی ها به حافظه

تعداد مراجعاتی که در الگوریتم دسته‌بندی درختی، برای پیمایش درخت به حافظه انجام می‌شود نشان دهنده زمان دسته‌بندی بسته ها است. از انجا که در اجرای الگوریتم‌ها، حافظه به عنوان گلوگاهی در سرعت اجرای دستورات حافظه‌ای به شمار می‌رود، الگوریتم‌هایی که برای دسته‌بندی یک بسته تعداد دسترسی بیشتری به حافظه داشته-باشند سرعت کمتری خواهند داشت[۲-۵, ۸-۹]. جدول ۳، تعداد کل دسترسی های دو الگوریتم مورد مقایسه را به حفظه سیستم جهت دسته‌بندی تعداد مشخصی از بسته‌ها طبق مجموعه قوانین مشخصی با تعداد یکسان نشان می‌دهد. طبق این جدول، در همه موارد تعداد مراجعات الگوریتم H-Trie به حافظه بیشتر از الگوریتم AQT است. در نتیجه، انتظار می رود سرعت دسته‌بندی الگوریتم AQT بیشتر از الگوریتم H-trie باشد. این افزایش سرعت با افزایش تعداد قوانین دسته‌بند که به افزایش عمق درخت سلسله مراتبی می‌انجامد بیشتر مشهود است. به عنوان مثال، در مورد در حالی که برای دسته‌بندی ۳۰ بسته با توجه به ۱۰ قانون در دسته‌بند، تعداد دسترسیهای الگوریتم AQT به حافظه، تقریبا پنج برابر کمتر از H-trie است، این نسبت با افزایش تعداد بسته ها و تعداد قوانین به ترتیب به ۲۰۲ بسته و ۳۰ قانون به حدود ۱۶ برابر می‌رسد.

این نتایج نشان می‌دهد که کارایی زمانی الگوریتم AQT به ویژه در دسته‌بندهایی با تعداد قوانین بیشتر بسیار بهتر از H-trie است.

جدول۳: زمان دسته‌بندی بسته‌ها توسط AQT و H-trie

تعداد قانون‌ها	تعداد بسته‌ها	AQT	H-Trie
۱۰	۳۰	۳۹	۱۸۶
۲۰	۶۰	۷۲	۶۷۳
۳۰	۲۰۲	۳۷۰	۶۱۶۹

۴-۲. حافظه موردنیاز

یکی دیگر از معیارهای مهم در انتخاب الگوریتم بمناسب برای دسته-بندی بسته‌ها میزان حافظه مصرفی الگوریتم برای ذخیره ساختار داده است. هر الگوریتم دسته‌بندی بسته، میزان فضای حافظه متفاوتی را لازم دارد. از این دیدگاه، الگوریتمی مناسبتر است که فضای حافظه مورنیاز آن کم بوده و همچنین، با افزایش تعداد قوانین دسته‌بند افزایش قابل توجهی نداشته باشد[۷].

برای بررسی و مقایسه این معیار، حافظه مورد نیاز در زمان اجرای دو الگوریتم متناظر با مجموعه قوانینی متفاوت، اندازه گرفته و در جدول شماره ۴ ثبت شده است.

جدول۴: حافظه مورنیاز AQT و H-trie

تعداد قانون‌ها	تعداد بسته‌ها	H-trie Memory (byte)	AQT Memory (byte)
۱۰	۳۰	۱۷۶۰	۴۸۰۹
۲۰	۶۰	۳۵۲۰	۷۲۶۷
۳۰	۲۰۲	۵۲۸۰	۱۳۲۱۶

بر اساس اطلاعات موجود در جدول شماره ۴، الگوریتم AQT به علت تعداد اشاره‌گرهای بیشتر نسبت به الگوریتم H-trei حافظه بیشتری را مصرف می‌کند. اختلاف میزان حافظه موردنیاز دو الگوریتم با افزایش تعداد قوانین دسته‌بند افزایش قابل توجهی می یابد. میزان این اختلاف با افزایش تعداد قوانین دسته‌بند به صورت غیرخطی افزایش می‌یابد. بنابراین، با توجه به نتایج زمانی می‌توان نتیجه گرفت که هرچند الگوریتم AQT سرعت بیشتری نسبت به H-trie در دسته‌بندی به کمک مجموعه قوانین زیاد دارد اما حافظه مصرفی مشکل حاد این الگوریتم در چنین کاربردهایی است.

۵. نتیجه‌گیری

دسته‌بندی بسته‌ها از پردازش‌های پایه در پردازنده‌های شبکه‌ای است. مهم‌ترین مساله در این زمینه، استفاده از الگوریتمی است که بتواند بسته‌ها با سرعتی در حد سرعت شبکه دسته‌بندی کند. این الگوریتم‌ها باید در مصرف حافظه هم به صورت بهینه عمل کنند. روش‌های موجود نتوانسته‌اند مصالحه مناسبی بین زمان و میزان حافظه مصرفی ایجاد نمایند. تمرکز این مقاله بر روی الگوریتم‌های جستجوی درختی است که از جمله الگوریتم‌های مطرح برای دسته‌بندی بسته‌ها است.

در ارزیابی‌های انجام شده از مجموعه قوانین و بسته‌های آزمایشی تولید شده توسط ابزار Classbench استفاده شد. ارزیابی‌های انجام شده بر روی دو الگوریتم درختی AQT و H-Trie به خوبی نشان می‌دهد که الگوریتم H-Trie میزان حافظه مصرفی کمتر ولی زمان طبقه‌بندی بسته‌ها طولانی تر است. در مقابل، الگوریتم AQT میزان حافظه مصرفی بیشتری برای دسته‌بندی سریعتر بسته‌ها لازم دارد.

ایده‌ای که می‌توان پیشنهاد داد آن است که با استفاده از تکنیکهای فشرده سازی ساختمان داده، فضای حافظه موردنیاز برای AQT را کاهش دهیم تا در دسته‌بندی بسته‌ها بر اساس مجموعه قوانین زیاد دچار مشکل حافظه نگردد ایده دیگر استاده از ساختارداده‌های حافظه-دار در دسته‌بند است. در چنین ساختارهایی قوانین پر ارجاع با تعداد دسترسی کمتری به حافظه در اختیار دسته‌بند قرار می‌گیرند[۱-۲, ۵, ۸-۹, ۱۱-۲۱].

مراجع

[۱] *D. E. Comer, Network Systems Design Using Network Processors: Intel 2XXX Version: Prentice-Hall, Inc., 2005.*

[۲] *A. Feldman and S. Muthukrishnan, "Tradeoffs for packet classification," in INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 2000, pp. 1193-1202.*

[۳] *P. Gupta and N. McKeown, "Packet classification on multiple fields," ACM SIGCOMM Computer Communication Review, vol. 29, pp. 147-160, 1999.*

[۴] *P. Gupta and N. McKeown, "Algorithms for packet classification," Network, IEEE, vol. 15, pp. 24-32, 2001.*

[۵] *C. U. L. f. R. i. Statistics, et al., IP Address Lookups and Packet Classification: A Tutorial and Review: Laboratory for Research in Statistics and Probability= Laboratoire de recherche en statistique et probabilités, Carleton University, 2002.*

[۶] *D. E. Taylor, "Survey and taxonomy of packet classification techniques," ACM Computing Surveys (CSUR), vol. 37, pp. 238-275, 2005.*

[۷] *V. Puš and J. Kořenek, "Reducing memory in high-speed packet classification," in Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International, 2012, pp. 437-442.*

[۸] *S. H. Shaikot and M. S. Kim, "Lightweight Traffic-Aware Packet Classification for Continuous Operation," in Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on, 2010, pp. 59-67.*

[۹] *T. Srinivasan, et al., "Efficient packet classification using splay tree models,"*

IJCSNS International Journal of Computer Science and Network Security, vol. 6, pp. 28-35, 2006.

[۱۰] *P.-C. Wang, et al., "A fast packet classification by using enhanced tuple pruning," in Protocols for High Speed Networks, 2002, pp. 180-191.*

[۱۱] *A. Bergamini and L. Kencl, "Network of shortcuts: An adaptive data structure for tree-based search methods," in NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems, ed: Springer, 2005, pp. 523-535.*

[۱۲] *M. A. Weiss and S. Hartman, Data structures and problem solving using Java vol. 204: Addison-Wesley Reading, 1998.*

[۱۳] *R. Jain, "Characteristics of destination address locality in computer networks: a comparison of caching schemes," Computer networks and ISDN systems, vol. 18, pp. 243-254, 1990.*

[۱۴] *H. Hamed and E. Al-Shaer, "Dynamic rule-ordering optimization for high-speed firewall filtering," in Proceedings of the 2006 ACM Symposium on Information, computer and communications security, 2006, pp. 332-342.*

[۱۵] *J. Bell and G. Gupta, "An evaluation of self - adjusting binary search tree techniques," Software: Practice and Experience, vol. 23, pp. 369-382, 1993.*

[۱۶] *S. Mythrei and R. Dharmaraj, "Packet Classification Based On Standard Access Control List".*

[۱۷] *D. D. Sleator and R. E. Tarjan, "Self-adjusting binary search trees," Journal of the ACM (JACM), vol. 32, pp. 652-686, 1985.*

[۱۸] *R. T. Cheruku, "Splay Tree," December 13, 2011.*

[۱۹] *Z. Trabelsi and S. Zeidan, "Splay trees based early packet rejection mechanism against DoS traffic targeting firewall default security rule," in Information Forensics and Security (WIFS), 2011 IEEE International Workshop on, 2011, pp. 1-6.*

[۲۰] *S. Acharya, et al., "Traffic-aware firewall optimization strategies," in Communications, 2006. ICC'06. IEEE International Conference on, 2006, pp. 2225-2230.*

[۲۱] *A. El-Atawy, et al., "Using online traffic statistical matching for optimizing packet filtering performance," in INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, 2007, pp. 866-874.*

زیرنویس‌ها

[1] Linear Search
[2] Decomposition
[3] Decision Tree
[4] Tuple Space
[5] Wildcard