การคัดแยกแพ็กเก็ตแบบ 2 มิติ โดยการจัดกลุ่มพรีฟิก

Two-Dimensional Packet Classification Based on Prefix Grouping

ทวีศักดิ์ กิจกาญจนารัตน์*

ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยธรรมศาสตร์ ศูนย์รังสิต ตำบลคลองหนึ่ง อำเภอคลองหลวง จังหวัดปทุมธานี 12120

วรา วังฉาย และ พันธุ์ปิติ เปี่ยมสง่า

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์

แขวงลาดยาว เขตจตุจักร กรุงเทพมหานคร

โรเบอร์โต โรแฮสเซสซา

ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ สถาบันเทคโนโลยีแห่งนิวเจอร์ซีย์

นิวเจอร์ซีย์ 07102 สหรัฐอเมริกา

Taweesak Kijkanjanarat*

Department of Electrical and Computer Engineering, Faculty of Engineering,

Thammasat University, Rangsit Centre, Khlong Nueng, Khlong Luang, Pathum Thani 12120

Wara Wangchai and Punpiti Piamsa-nga

Department of Computer Engineering, Faculty of Engineering, Kasetsart University,

Ladyao, Chatuchak, Bangkok 10900

Roberto Rojas-Cessa

Department of Electrical and Computer Engineering, New Jersey Institute of Technology,

New Jersey 07102, U.S.A.

บทคัดย่อ

งานวิจัยนี้นำเสนอโครงสร้างข้อมูลและขั้นตอนวิธีในการคัดแยกแพ็กเก็ตแบบ 2 มิติ โดยการจัด filter ออกเป็นกลุ่มตามค่า prefix ซึ่งทำให้โครงสร้างข้อมูลสามารถรองรับ filter set ที่มีขนาดใหญ่และรองรับ filter set ได้ทุกประเภท ขั้นตอนวิธีในการคัดแยกแพ็กเก็ตได้ถูกออกแบบให้การคัดแยกแต่ละมิติทำคู่ขนานไปพร้อม ๆ กัน จึงทำให้การคัดแยกแพ็กเก็ตสามารถทำได้อย่างรวดเร็ว จากผลการทดสอบทั้งในแง่ของการใช้หน่วยความจำของโครงสร้างข้อมูล และการวัดจำนวนครั้งที่ติดต่อหน่วยความจำเพื่ออ่านข้อมูลขณะทำการคัดแยกแพ็กเก็ต โดยใช้ benchmark ที่มีชื่อว่า Classbench ซึ่งเป็นเครื่องมือในการสร้าง filter set จำลอง โดยเปรียบเทียบผลกับงานวิจัย

***ผู้รับผิดชอบบทความ :** taweesak@engr.tu.ac.th

อื่น ๆ พบว่าโครงสร้างข้อมูลที่ filter set ขนาด 50,000 filters ใช้หน่วยความจำสูงสุดเพียงแค่ 5 MB และมีจำนวนครั้งที่ติดต่อหน่วยความจำสูงสุดเพียง 26 ครั้ง ต่อการคัดแยก 1 แพ็กเก็ต โดยขนาดหน่วยความจำของโครงสร้างข้อมูลและจำนวนครั้งที่ติดต่อหน่วยความจำในการคัดแยกแพ็กเก็ตเพิ่มขึ้นเพียงเล็กน้อยเมื่อจำนวน filter ของ filter set เพิ่มขึ้น จึงทำให้งานวิจัยนี้มีสมบัติในเรื่องของ scalability นอกจากนี้ เนื่องจากขั้นตอนวิธีในการแมทช์ค่า filter ไม่ซับซ้อน จึงทำให้งานวิจัยนี้สามารถนำไปประยุกต์เพื่อออกแบบและติดตั้งลงในอุปกรณ์ฮาร์ดแวร์ได้ด้วยเทคโนโลยีที่มีอยู่ในปัจจุบันได้

คำสำคัญ : การคัดแยกแพ็กเก็ต; การจัดกลุ่มพรีฟิก; การแมทช์ค่าฟิลเตอร์; การแมทช์พรีฟิก

Abstract

Nowadays, packet classification is still an essential methodology to serve the needs of Internet applications. Several researches have overcome speed of packet classification but none of them has been scalable for over tens of thousands filter sets. This paper proposes a two-dimensional packet classification scheme. Based on grouping prefix fields of filters, the data structure takes very small memory requirements. The packet classification rate is also fast since each dimension is classified in a parallel manner. The scheme performance is measured by using filter sets generated from the well-known benchmark called ClassBench. Based on experimental results, the memory taken for the data structures is about 5 MB and it takes at most 26 memory accesses when classifying a packet. Also, the scheme can apply to all types of filter sets. In summary, it is found that the size of the filter set does not have any significant impact on the scheme performance. The proposed scheme tremendously improves performance of packet classification in terms of scalability and can be easily implemented with today's technology.

Keywords: packet classification; prefix grouping, filter match, prefix match

1. บทนำ

การใช้งานอินเทอร์เน็ตในปัจจุบัน พบว่ามีความแตกต่างกันในกลุ่มของผู้ใช้บริการในแง่ของความคาดหวังจากบริการที่จะได้รับ และความสามารถในการจ่ายค่าบริการอินเทอร์เน็ต ผู้ใช้บริการบางท่านยินดีที่จะจ่ายค่าบริการในราคาสูง เพื่อแลกกับการบริการที่มีคุณภาพเพื่อใช้ application ประเภท video streaming ซึ่งต้องการ bandwidth ระดับสูง ขณะที่ผู้ใช้บริการบางท่านต้องการจ่ายค่าใช้บริการในราคาที่ต่ำกว่า เนื่องจากต้องการใช้เพียงแค่บริการพื้นฐานทั่วไป เช่น การรับส่ง e-mail ด้วยเหตุนี้ การให้บริการของอินเทอร์เน็ตจึงจำเป็นต้องมีความแตกต่างกัน เพื่อให้สามารถตอบสนองต่อความต้องการของผู้ใช้บริการที่หลากหลายได้ internet service provider (ISP) จึงมีหน้าที่ตอบสนองต่อการร้องขอการบริการที่แตกต่างกันของผู้ใช้บริการโดยที่การบริการต่าง ๆ นั้นจะถูกจำแนกด้วยหลายปัจจัยตามความเหมาะสมของผู้ใช้บริการ ได้แก่ การจำกัด

bandwidth การทำ authorization เพื่อกำหนดกลุ่มผู้ใช้บริการ การจำแนกเส้นทางการส่งต่อข้อมูลในเครือข่าย (traffic isolation) เป็นต้น เพื่อให้การบริการของอินเทอร์เน็ตมีความแตกต่างกัน อุปกรณ์ค้นหาเส้นทางในเครือข่ายหรือ router จะต้องมีความสามารถในการจำแนกข้อมูลหรือแพ็กเก็ต (IP packet) ของผู้ใช้บริการ และสามารถกระทำ action ต่อแพ็กเก็ตที่วิ่งเข้ามาที่ routerได้ตามความเหมาะสม

โดยทั่วไป อินเทอร์เน็ตมีการให้บริการที่เรียบง่ายที่เรียกว่า best-effort service กล่าวคือ ในการส่งต่อแพ็กเก็ตจาก router หนึ่งไปยัง router ถัดไป แต่ละ router จะอาศัยข้อมูลเพียงแค่ destination IP address (DA) ซึ่งอยู่ที่เฮดเดอร์ของแพ็กเก็ต โดยพิจารณาว่าสำหรับ DA นี้จะต้องส่งต่อแพ็กเก็ตไปยัง router ถัดไปตัวใด โดยพิจารณาแมทช์ค่า DA กับ prefix ต่าง ๆ ที่เก็บใน forwarding table ที่อยู่ที่ router โดย router จะส่งต่อแพ็กเก็ตไปให้กับ router ถัดไป (next hop) ที่ถูกระบุโดย prefix ที่แมทช์กับ DA แบบ longest match

การให้บริการแบบ best-effort นี้ ไม่เหมาะสมกับสถานการณ์ปัจจุบันที่ผู้ใช้บริการต้องการบริการที่แตกต่างกัน เนื่องจากทุกแพ็กเก็ตถูกกระทำอย่างเท่าเทียมกันโดยไม่สนใจว่าต้นทางเป็นใคร และเป็นแพ็กเก็ตที่เกิดจากการใช้งานของ application ใด ด้วยเหตุนี้จึงได้มีการปรับเปลี่ยนกระบวนการตรวจสอบแพ็กเก็ตที่ router เพื่อให้สามารถตอบสนองต่อผู้ใช้บริการที่มีความสำคัญแตกต่างกัน ซึ่งเรียกกระบวนการนี้ว่าการคัดแยกแพ็กเก็ต (packet classification) การคัดแยกแพ็กเก็ตจึงไม่ได้อาศัยข้อมูลเพียงแค่ DA เท่านั้น แต่จะใช้ฟิลด์อื่น ๆ ของเฮดเดอร์ของแพ็กเก็ตด้วย ได้แก่ source IP address (SA), source port number (SP), destination port number (DP), protocol flag (prot) เป็นต้น โดยทั่วไปการคัดแยกแพ็กเก็ตจะมีความหลากหลายในการพิจารณาฟิลด์ที่ใช้ ซึ่งขึ้นอยู่กับเป้าหมายของการบริการ แต่ส่วนใหญ่จะใช้ 5 ฟิลด์ ที่กล่าวมา ซึ่งเรียกว่าการคัดแยกแพ็กเก็ตแบบ 5 มิติ (five-dimensional packet classification)

งานวิจัยนี้นำเสนอโครงสร้างข้อมูลและขั้นตอนวิธีในการคัดแยกแพ็กเก็ตแบบ 2 มิติ (two-dimensional packet classification) กล่าวคือ ในการคัดแยกแพ็กเก็ตจะพิจารณาข้อมูลเฉพาะในส่วนของ source IP address และ destination IP address ถึงแม้ว่าปัญหาการคัดแยกแพ็กเก็ตแบบ 2 ฟิลด์ นั้นดูเหมือนว่าจะเป็นเพียงแค่ส่วนหนึ่งของการคัดแยกแพ็กเก็ตแบบ 5 ฟิลด์ แต่ปัญหานี้ก็ยังถือว่ามีความสำคัญที่สามารถประยุกต์ใช้งานจริงได้ในหลายกรณี เช่น (1) application ประเภท VPN โดยเฉพาะ MPLS VPN ซึ่งต้องการค่า source IP address และ destination IP address ในการสร้างระบบเครือข่าย VPN (2) IP multicast ซึ่งเป็นโพรโทคอลการส่งข้อมูลเฉพาะกลุ่มในเครือข่าย โดยจะใช้ single source IP address และ multicast address group ในการรับส่ง multicast message ระหว่างกลุ่มเครือข่าย [1] ซึ่งการส่งนี้จะกระทำเพียงครั้งเดียว ทำให้ประหยัด bandwidth เครือข่าย IP multicast ได้ถูกนำไปใช้ในงานประเภท internet television และ streaming m media (3) IP firewall ซึ่งกำหนดค่า range ของ source IP address และ destination IP address ที่อนุญาตหรือไม่อนุญาตในการเข้าถึงเครือข่ายหนึ่ง ๆ [2] เป็นต้น งานวิจัยนี้นำเสนอโครงสร้างข้อมูลและขั้นตอนวิธีในการคัดแยกแพ็กเก็ตแบบ 2 มิติ โดยโครงสร้างข้อมูลเกิดจากการนำค่า prefix ในแต่ละฟิลด์ของ filter ใน filter table คือ source prefix และ destination prefix มาจัดกลุ่ม จากนั้นนำข้อมูลที่จัดกลุ่มได้จากแต่ละฟิลด์มาทำการ cross-product

เพื่อให้โครงสร้างข้อมูลใช้หน่วยความจำน้อยที่สุด โดยกระบวนการคัดแยกแพ็กเก็ต จะทำการคัดแยกแต่ละฟิลด์ในลักษณะที่ทำคู่ขนานไปพร้อม ๆ กัน (parallel) ซึ่งจากการทดสอบพบว่าโครงสร้างข้อมูลใช้หน่วยความจำสูงสุดเพียงแค่ 5 MB ต่อ filter set ขนาด 50,000 filters ซึ่งน้อยมาก นอกจากนี้โครงสร้างข้อมูลยังมีส่วนช่วยเพิ่มประสิทธิภาพในการคัดแยกแพ็กเก็ต โดยการคัดแยกแพ็กเก็ตแต่ละครั้งมีจำนวนครั้งโดยเฉลี่ยที่ติดต่อหน่วยความจำเพียงแค่ 10 ถึง 26 ครั้ง โดยประมาณ ซึ่งมีความรวดเร็วมาก ทำให้สามารถรองรับการบริการที่แตกต่างกันได้อย่างทันท่วงที ส่งผลให้ลดปัญหาคอขวดที่เกิดขึ้นต่อ router ได้เป็นอย่างดี

เค้าโครงงานวิจัยนี้ประกอบด้วย (1) ปัญหาการคัดแยกแพ็กเก็ต (2) งานวิจัยที่เกี่ยวข้อง ซึ่งจะกล่าวถึงภาพรวมของงานวิจัยการคัดแยกแพ็กเก็ต โดยแบ่งออกเป็นกลุ่มอย่างชัดเจน (3) โครงสร้างข้อมูลและขั้นตอนวิธีของงานวิจัยนี้ (4) ผลการทดสอบโครงสร้างข้อมูลและขั้นตอนวิธี โดยแสดงผลเปรียบเทียบกับงานวิจัยอื่น ๆ และ (5) สรุปงานวิจัย

2. ปัญหาการคัดแยกแพ็กเก็ต

หัวข้อนี้จะกล่าวถึงหลักในการคัดแยกแพ็กเก็ตและการประเมินประสิทธิภาพของการคัดแยกแพ็กเก็ต ซึ่งมีรายละเอียดดังนี้

2.1 หลักในการคัดแยกแพ็กเก็ต

กำหนดให้มี filter set ที่ประกอบด้วย filter จำนวน n filters คือ F_1, F_2, F_3, ..., F_n โดยแต่ละ filter F_i ประกอบด้วย K ฟิลด์ เช่น ใน IPv4 ฟิลด์ต่าง ๆ ประกอบด้วยค่า source prefix (Src prefix), destination prefix (Dest prefix), source port number (SP), destination port number (DP) และ protocol flag (Prot) โดยวิธีการแมทช์ค่าของแต่ละฟิลด์จะแตกต่างกันขึ้นอยู่กับรูปแบบของฟิลด์นั้น ๆ ซึ่งมีอยู่ด้วยกัน 3 ลักษณะ คือ

(1) prefix match เป็นวิธีที่ใช้กับการแมทช์ฟิลด์ที่เป็น source/destination prefix โดยค่า source/destination IP address ของแพ็กเก็ต P จะแมทช์กับค่า source/destination prefix ของ F_i ถ้าแต่ละบิตที่มีนัยสำคัญ (significant bit) ของ source/destination prefix มีค่าเท่ากับแต่ละบิตในตำแหน่งเดียวกันของ source/destination IP address ของ P

(2) range match เป็นวิธีที่ใช้กับการแมทช์ฟิลด์ที่เป็น source/destination port number โดยฟิลด์ของ F_i ที่เป็น source/destination port number จะอยู่ในรูปแบบที่เป็นช่วงของตัวเลข $[Min_{F_i}, Max_{F_i}]$ ค่า source/destination port number ของแพ็กเก็ต P จะแมทช์กับค่า source/destination port number ของ F_i ถ้า source/destination port number ของ P เป็นค่าหนึ่งในช่วง $[Min_{F_i}, Max_{F_i}]$

(3) exact match เป็นวิธีที่ใช้กับการแมทช์ฟิลด์ที่เป็น protocol flag โดย protocol flag ของแพ็กเก็ต P จะแมทช์กับค่า protocol flag ของ F_i ถ้า protocol flag ของ P มีค่าเท่ากับ protocol flag ของ F_i

แพ็กเก็ต P ใด ๆ จะถือว่าแมทช์กับ F_i ก็ต่อเมื่อทุกค่าของเฮดเดอร์ฟิลด์ของ P แมทช์กับทุกฟิลด์ของ F_i ตามวิธีการแมทช์ของแต่ละฟิลด์ เมื่อแพ็กเก็ต P แมทช์กับ F_i แล้ว P จะถูกกระทำโดย router ตามค่า action ที่ถูกกำหนดไว้ใน F_i เช่น drop แพ็กเก็ต, forward แพ็กเก็ตไปยัง router ถัดไปโดยใช้ high-speed link เป็นต้น อย่างไรก็ตาม P หนึ่ง ๆ สามารถแมทช์กับ F_i ได้หลายค่า ในการตัดสินว่า filter ใดแมทช์กับ P ได้ดีที่สุด จะกำหนดให้แต่ละ filter มีค่าความสำคัญ (priority) โดย filter ที่แมทช์กับ P ได้ดีที่สุด คือ filter ที่มีค่าความสำคัญสูงสุด

ตัวอย่างของการแมทช์สามารถอธิบายได้ดังนี้ สมมติว่าแพ็กเก็ต P วิ่งมาที่ router R ใด ๆ ที่มี filter set ตามตารางที่ 1 โดยที่ * หมายถึงค่า don't care และ P มีโครงสร้างของเฮดเดอร์ฟิลด์ {source IP address, destination IP address, source port number, destination port number, protocol flag} = {00101101, 01111010, 64, 1024, TCP} แพ็กเก็ต P จะแมทช์ (1) source IP address แบบ prefix match โดย source IP address ของ P แมทช์กับ source prefix ของ F_4, F_5, F_7, F_8 (2) destination IP address แบบ prefix match โดย destination IP address ของ P แมทช์กับ destination prefix ของ F_0, F_5, F_6,F_7, F_{10} (3) source port number แบบ range match โดย source port number ของ P แมทช์กับช่วงของค่า port number ของ F_0, F_2, F_4, F_5, F_6, F_7, F_8 (4) destination port number แบบ range match โดย destination port number ของ P แมทช์กับช่วงของค่า port number ของทุก filter ตั้งแต่ F_0 ถึง F_{10} และ (5) protocol flag แบบ exact match โดย protocol flag ของ P ซึ่งก็คือ TCP แมทช์กับ protocol flag ของ F_1, F_2, F_3, F_4, F_5, F_7 F_8, F_9 เมื่อรวมผลลัพธ์ของการแมทช์ด้วยวิธี and operation จะทำให้ได้ Filter สุดท้ายที่แมทช์กับ P คือ F_5 และ F_7 ในงานวิจัยนี้ จะกำหนดให้ filter เลขน้อยมีค่าความสำคัญสูงกว่า filter เลขมาก ดังนั้นจะได้ว่า flter ที่แมทช์กับ P ได้ดีที่สุด คือ F_5 เนื่องจาก F_5 มีค่าความสำคัญที่สูงกว่า F_7 ดังนั้น P จะถูก router R กระทำตาม action ที่เป็นค่า Act_5

ตารางที่ 1 ตัวอย่าง filter set ที่ router R

#	Src prefix	Dest prefix	SP	DP	Prot	Action
F0	001101*	*	[64-64]	[0-65535]	UDP	Act_0
F1	10110*	1100*	[1024-1024]	[1024-1024]	TCP	Act_1
F2	10110*	1100*	[64-1222]	[1024-1024]	TCP	Act_2
F3	0100*	1*	[1024-1024]	[0-1024]	*	Act_3
F4	*	11001*	[0-65535]	[0-1024]	TCP	Act_4
F5	001*	0111*	[64-64]	[0-65535]	*	Act_5
F6	01011*	011110*	[64-1222]	[1024-1024]	UDP	Act_6
F7	001*	0111*	[64-1222]	[0-65535]	*	Act_7
F8	*	1100*	[64-1222]	[0-65535]	TCP	Act_8
F9	10110*	1100*	[1048-4096]	[256-1024]	TCP	Act_9
F10	001101*	*	[1024-1024]	[1024-1024]	ICMP	Act_{10}

2.2 การประเมินประสิทธิภาพของการคัดแยกแพ็กเก็ต

การประเมินประสิทธิภาพของการคัดแยกแพ็กเก็ตจะมีเกณฑ์ที่สำคัญในการพิจารณาอยู่ด้วยกัน 2 ประการ คือ ขนาดของหน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูลที่ใช้แทน filter set และความรวดเร็วในการคัดแยกแพ็กเก็ตของขั้นตอนวิธีที่ใช้ในการคัดแยกแพ็กเก็ต

2.2.1 ขนาดของหน่วยความจำ (memory requirement) ที่ใช้สำหรับโครงสร้างข้อมูล

งานวิจัยของการคัดแยกแพ็กเก็ตจะทำการวัดขนาดของหน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูลที่ใช้แทน filter set โดยถ้าโครงสร้างข้อมูลใช้ขนาดของหน่วยความจำน้อย แสดงว่าการคัดแยกแพ็กเก็ตนั้นมีประสิทธิภาพ เนื่องจากสามารถสร้างโครงสร้างข้อมูลนี้ลงในหน่วยความจำที่มีขนาดที่เทคโนโลยีปัจจุบันสามารถรองรับได้

2.2.2 ความรวดเร็วในการคัดแยกแพ็กเก็ต (packet classification rate) ของขั้นตอนวิธี

จุดประสงค์ที่สำคัญของขั้นตอนวิธีการคัดแยกแพ็กเก็ต คือ ความสามารถในการคัดแยกแพ็กเก็ตที่จะต้องทำให้เร็วที่สุด เพื่อหลีกเลี่ยงปัญหาคอขวดที่เกิดขึ้นกับ router การวัดความเร็วของการคัดแยกแพ็กเก็ตของขั้นตอนวิธีจะวัดจำนวนครั้งที่ติดต่อกับหน่วยความจำ (the number of memory accesses) ขณะทำการค้นหา filter ที่แมทช์กับแพ็กเก็ตในโครงสร้างข้อมูล โดยคิดเทียบต่อ 1 แพ็กเก็ต ในการวัดดังกล่าว ถ้าจำนวนครั้งที่ติดต่อหน่วยความจำมีค่าน้อย เวลาที่ใช้ในการคัดแยกแพ็กเก็ตจะมีค่าน้อย ซึ่งส่งผลให้การคัดแยกแพ็กเก็ตของขั้นตอนวิธีมีประสิทธิภาพสูง

3. งานวิจัยที่เกี่ยวข้อง

นับตั้งแต่อดีตจนถึงปัจจุบันได้มีการสำรวจงานวิจัยที่เกี่ยวกับการคัดแยกแพ็กเก็ตแบบ 2 มิติ [3-5] ซึ่งงานวิจัยเหล่านี้มีความหลากหลายทั้งในแง่ของโครงสร้างข้อมูลและขั้นตอนวิธี โดยสามารถจำแนกการคัดแยกแพ็กเก็ตแบบ 2 มิติ เป็นกลุ่ม ๆ ได้ดังนี้

3.1 งานวิจัยทางด้านฮาร์ดแวร์ (hardware-based solution)

กลุ่มของงานวิจัยทางด้านฮาร์ดแวร์จะมีการใช้หน่วยความจำพิเศษที่เรียกว่า ternary content addressable memories (TCAMs) [6-8] ซึ่งความสามารถของ TCAMs จะมีความรวดเร็วมากในการคัดแยกแพ็กเก็ต เพราะสมบัติพื้นฐานที่ว่าสามารถกำหนด filter set ลงไปได้โดยตรงในหน่วยความจำนั่นคือ กระบวนการค้นหาจะทำได้ใน *O(1)* อย่างไรก็ตาม TCAMs ก็มีข้อเสียที่ไม่เหมาะสมต่อการนำมาใช้ในงานคัดแยกแพ็กเก็ต ดังต่อไปนี้ (1) TCAMs เป็นฮาร์ดแวร์ที่ใช้พลังงานจำนวนมากเมื่อเทียบกับหน่วยความจำทั่ว ๆ ไป เพราะข้อมูลหนึ่ง ๆ ที่บันทึกใน TCAMs จะต้องใช้ transistor จำนวนมาก (2) เนื่องจาก TCAMs ใช้ transistor จำนวนมาก จึงทำให้มีราคาสูง (3) TCAMs ไม่สามารถนำข้อมูลประเภท range เช่น ค่า port number มาใส่ในหน่วยความจำได้โดยตรง ซึ่งจำเป็นจะต้องมีการกระบวนการในการแปลงค่า range เป็นค่า prefix ก่อน จากนั้นจึงนำมาใส่ในหน่วยความจำ ซึ่งเป็นวิธีที่มีความซับซ้อน และสิ้นเปลืองพื้นที่หน่วยความจำ จากข้อจำกัดดังกล่าวของ TCAMs ทำให้ TCAMs ยังไม่เป็นที่นิยมที่จะนำมาใช้งานอย่างจริงจัง ด้วยเหตุผลนี้เองที่ทำให้งานวิจัยการคัดแยกแพ็กเก็ตทางด้านซอฟต์แวร์ยังคงมีความสำคัญ

3.2 งานวิจัยทางด้านซอฟต์แวร์ (software-based solution)

กลุ่มของงานวิจัยทางด้านซอฟต์แวร์จะเป็นการพัฒนาโครงสร้างข้อมูลและขั้นตอนวิธีภายใต้เทคโนโลยีของหน่วยความจำพื้นฐานประเภท SRAM ซึ่งสามารถจำแนกเป็นกลุ่มย่อย ๆ ตามโครงสร้างข้อมูลและขั้นตอนวิธีที่ออกแบบได้ดังต่อไปนี้

3.2.1 ขั้นตอนวิธีแบบเชิงเส้น (linear search algorithm) งานวิจัยรูปแบบนี้ [7,9] จะใช้โครงสร้างข้อมูลที่ไม่ซับซ้อน ซึ่งก็คืออาร์เรย์ (array) ของ index ที่บอกถึงกลุ่มของ filter ที่ค่าฟิลด์อยู่ในช่วงขอบเขตเดียวกัน โดยที่แต่ละช่องของอาร์เรย์จะมี pointer ชี้ไปยังโครงสร้างข้อมูลแบบ tree ที่จัดเก็บ

filter ที่ตกอยู่ในกลุ่มนี้ สำหรับกระบวนการคัดแยกแพ็กเก็ตจะกระทำแบบ linear search ในอาร์เรย์และแบบ binary search ใน tree ซึ่งจะเห็นได้ว่า ในกระบวนการคัดแยกแพ็กเก็ต ถึงแม้จะมีการนำ binary search ซึ่งมี worst-case time *O(log n)* มาใช้ในการคัดแยกแพ็กเก็ตใน tree แต่กระบวนการคัดแยกแพ็กเก็ตโดยรวมจะมีความช้ามาก อันเนื่องมาจากกระบวนการ linear search ซึ่งมี worst-case time เป็น *O(n)* ทำให้ยังไม่เหมาะสมต่อการนำมาใช้งานจริง

3.2.2 ขั้นตอนวิธีแบบ trie (trie-based algorithm) รูปแบบของขั้นตอนวิธีแบบ trie [10-12] จะนำ prefix ของ filter มาสร้างเป็นโครงสร้างแบบ trie ซึ่งมีความคล้ายคลึงกับ tree โดยกระบวนการคัดแยกแพ็กเก็ตจะกระทำภายใต้ trie โดยการแมทช์ค่า filter จะค้นหาจากโหนด root จนถึงโหนด leaf ของ trie ข้อเสียของวิธีนี้ คือ เมื่อ filter set มีขนาดใหญ่ขึ้น ขนาดของ trie ก็จะกว้าง (breadth) และสูง (height) ตามไปด้วย ส่งผลให้กระบวนการคัดแยกแพ็กเก็ตภายใต้โครงสร้างข้อมูลนี้ช้า อีกทั้งหน่วยความจำที่ใช้ก็จะมากขึ้นแบบ exponential จึงมีข้อจำกัดในการนำมาใช้งานจริง เพื่อรองรับ filter set ที่มีขนาดใหญ่

งานวิจัยล่าสุดที่ใช้โครงสร้างแบบ trie ได้แก่ binary search on levels with replication control (BSOL-RC) [13] ซึ่งอาศัยการควบคุม filter ที่มีความซ้ำซ้อน (replication) สูงโดยจัด filter เหล่านี้ลงในโครงสร้างแบบ decision tree ที่ถูกออกแบบขึ้นมาโดยเฉพาะ จึงทำให้โครงสร้างหลักไม่มีโหนดที่มี filter ที่ซ้ำซ้อน ดังนั้นขนาดของหน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูลทั้งหมดจะมีค่าน้อยกว่างานวิจัยอื่น ๆ ที่ใช้โครงสร้างแบบ trie อย่างไรก็ตาม จากผลของงานวิจัยนี้ พบว่าการค้นหา filter ที่แมทช์กับแพ็กเก็ต จำนวนครั้งสูงสุดที่ใช้ติดต่อกับหน่วยความจำต่อการคัดแยก 1 แพ็กเก็ต มีค่าสูงถึง 37 ครั้ง ซึ่งถือว่าเป็นค่าที่สูงเมื่อเทียบกับงานวิจัยอื่น ๆ

3.2.3 ขั้นตอนวิธีแบบ heuristic (heuristic-based algorithm) งานวิจัยในกลุ่มนี้ได้แก่ recursive flow classification (RFC) [1] ซึ่งโครงสร้างข้อมูลเกิดจากการแบ่งฟิลด์ต่าง ๆ ของ filter ออกเป็นกลุ่ม แต่ละกลุ่มจะถูก mapping ซึ่งกันและกัน ก่อให้เกิดโครงสร้างข้อมูลใหม่เป็นอาร์เรย์ข้อมูล จนสุดท้ายจะได้ตารางข้อมูลที่เกิดจากการ mapping ของตารางข้อมูลก่อนหน้าทั้งหมด สำหรับกระบวนการคัดแยกแพ็กเก็ตจะกระทำภายใต้ชุดข้อมูลที่ถูกแบ่งตามแต่ละเฟส โดยใช้กระบวนการ map index ของตารางข้อมูลโดยตรงกับเฮดเดอร์ฟิลด์ของแพ็กเก็ต อย่างไรก็ตาม ด้วยโครงสร้างข้อมูลที่ซับซ้อนและในชุดของกลุ่มข้อมูลที่ถูกแบ่งมีการใช้หน่วยความจำเป็นจำนวนมากอันเนื่องมาจากอาร์เรย์ที่ใช้เก็บข้อมูลมีขนาดใหญ่ ทำให้ RFC ไม่สามารถรองรับ filter set ที่มีขนาดใหญ่ได้ อย่างไรก็ตาม ได้มีงานวิจัย hierarchical space mapping (HSM) [14] ซึ่งแก้ไขข้อเสียของ RFC โดยวิธีการ คือ จะมีโครงสร้างอาร์เรย์หรือ mapping table ในเฟสที่สองแค่ 2 โครงสร้างเท่านั้น ซึ่งช่วยลดขนาดของหน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูล สำหรับกระบวนการคัดแยกแพ็กเก็ตจะต่างกับ RFC คือ ใช้วิธี binary search ที่ตารางข้อมูลเพื่อหาค่า index ของตารางข้อมูลในเฟสถัดไป ซึ่งวิธีนี้เป็นผลเสียต่อ HSM เนื่องจากกระบวนการ binary search มี worst-case time เป็น *O(log n)* ในขณะที่การ map index ของ RFC มี worst-case time เป็น *O(1)*

งานวิจัยล่าสุดที่พัฒนาต่อยอดจาก RFC ได้แก่ multi-iteration RFC [15] โดยงานวิจัยนี้นำเสนอโครงสร้าง decision tree เพื่อกระจาย filter ไปยังโหนด leaf และนำ filter ที่อยู่ที่แต่ละโหนด leaf

มาสร้างเป็นโครงสร้าง RFC โดยโครงสร้าง RFC ของแต่ละโหนด leaf ที่ได้จะมีขนาดเล็ก อย่างไรก็ตาม เนื่องจากวิธีการนี้จะต้องมีตารางเพื่อเก็บข้อมูลการ mapping ระหว่างค่า index ที่เชื่อมโยงโครงสร้าง RFC กับ decision tree จึงทำให้ต้องใช้หน่วยความจำเพิ่มเติมเพื่อรองรับตารางนี้

งานวิจัย HyperCuts [16] และ HyperSplit [17] เป็นงานวิจัยที่มีความซับซ้อนเช่นเดียวกับ RFC โดยแนวคิดของงานวิจัยนี้อาศัยความเหลื่อมล้ำของค่าฟิลด์ (field overlap) ของ filterโดยมีกระบวนการในการจัดการข้อมูลเหล่านี้ก่อนแล้วจึงนำมาสร้างเป็นโครงสร้างข้อมูลแบบ multi-way Tree สำหรับการคัดแยกแพ็กเก็ตจะกระทำภายใต้ tree นี้ โดยการ Map แพ็กเก็ตตามค่าฟิลด์ในแต่ละโหนด จนสุดท้ายการเดินทางภายใน tree สิ้นสุดที่โหนด leaf งานวิจัยนี้มีข้อเสีย คือ ค่าความสูง (height) ของ tree มีค่ามาก ทำให้กระบวนการค้นหาช้า อีกทั้งกระบวนการสร้างโครงสร้างข้อมูลจะมีการจองหน่วยความจำเป็นจำนวนมาก ทำให้ไม่สามารถรองรับ filter set ที่มีขนาดใหญ่ได้

4. โครงสร้างข้อมูลและขั้นตอนวิธีของงานวิจัยนี้

จากงานวิจัยทั้งหมดที่กล่าวมาข้างต้น แม้จะมีความสามารถในการคัดแยกแพ็กเก็ต แต่ก็ยังมีข้อจำกัดอยู่หลายประการ โดยเฉพาะความสามารถในการรองรับ filter set ที่มีขนาดใหญ่ (กล่าวคือ ขาดสมบัติในเรื่องของ scalability) ซึ่งปัจจัยนี้เป็นสิ่งสำคัญในการพัฒนาโครงสร้างข้อมูลและขั้นตอนวิธีสำหรับการคัดแยกแพ็กเก็ตของงานวิจัยนี้ งานวิจัยนี้ได้เสนอการคัดแยกแพ็กเก็ตแบบ 2 มิติ โดยอาศัยการจัดกลุ่ม prefix ในแต่ละฟิลด์ของ filter เพื่อจำกัดการใช้หน่วยความจำของโครงสร้างข้อมูล ซึ่งจากการวิจัยพบว่าสามารถรองรับ filter Set ที่มีขนาดใหญ่ได้ และด้วยกระบวนการ cross-product ภายใต้การจัดกลุ่มข้อมูล ทำให้ขั้นตอนวิธีสามารถค้นหาข้อมูลได้อย่างรวดเร็วอีกด้วย เนื่องจากโครงสร้างข้อมูลและขั้นตอนวิธีที่นำเสนอในงานวิจัยนี้ไม่มีความซับซ้อน ทำให้ในอนาคตสามารถนำงานวิจัยนี้ไปประยุกต์เพื่อออกแบบและติดตั้งลงในอุปกรณ์ฮาร์ดแวร์ได้อย่างง่ายดาย ซึ่งแสดงให้เห็นว่าโครงสร้างข้อมูลและขั้นตอนวิธีที่เสนอในงานวิจัยนี้สามารถนำมาประยุกต์ใช้งานได้ด้วยเทคโนโลยีปัจจุบัน

แนวคิดของงานวิจัยนี้ได้แรงบันดาลใจเริ่มต้นมาจากการสังเกตผลการศึกษาของงานวิจัย DCFL [18] ซึ่งได้ให้ข้อสังเกตว่าค่าในแต่ละฟิลด์ของ filter ใน filter set จะมีค่าที่ซ้ำ ๆ กันเป็นจำนวนมาก จึงเกิดแนวคิดให้กับผู้วิจัยว่าถ้าในแต่ละฟิลด์ มีการจัดกลุ่มค่า prefix ค่า port และค่า protocol flag จำนวนกลุ่มที่ได้จะมีค่าน้อยมากเมื่อเทียบกับจำนวน filter ทั้งหมดของ filter set ซึ่งแสดงให้เห็นว่าแท้จริงแล้วจำนวน filter ทั้งหมดใน filter set มีค่าที่ unique น้อยมาก ดังนั้นในการคัดแยกแพ็กเก็ตจึงน่าจะมีการจัด filter ออกเป็นกลุ่ม ๆ ตามค่าในแต่ละฟิลด์ก่อน โดยขณะทำการคัดแยกแพ็กเก็ตให้คัดแยกตามจำนวนกลุ่มแทนการคัดแยกตามจำนวน filter ซึ่งน่าจะทำให้การคัดแยกแพ็กเก็ตรวดเร็วกว่า เพื่อตรวจสอบสมมติฐานนี้ งานวิจัยนี้จึงได้ลองทำการวิเคราะห์ filter set ที่นำมาจาก benchmark ที่ชื่อว่า ClassBench [19] ซึ่งแบ่ง filter ออกเป็น 3 ประเภทหลัก คือ access control list (ACL), firewall (FW) และ IP chain (IPC) โดยได้ผลการวิเคราะห์ดังแสดงในตารางที่ 2

จากตารางที่ 2 จะเห็นได้ว่าจำนวน filter ในแต่ละ filter set จะมีค่า unique น้อยมาก เช่น ACL5 มีจำนวน 4,562 filters เมื่อลองจัดกลุ่ม prefix แยกกันระหว่าง source และ destination prefix

พบว่าได้ source prefix จำนวน 1,179 กลุ่ม (ข้อมูลลดลงเหลือ 25.80 %) และ destination prefix จำนวน 306 กลุ่ม (ข้อมูลลดลงเหลือ 6.70 %) นอกจากนี้ยังพบว่าจำนวนของค่า unique prefix ที่แมทช์กับแพ็กเก็ตหนึ่ง ๆ ในแต่ละฟิลด์ มีค่าไม่เกิน 2 ซึ่งผลที่ได้เป็นไปตามสมมติฐานที่ผู้วิจัยตั้งไว้

ตารางที่ 2 ผลการวิเคราะห์ค่า unique ของ source/destination prefix ของ filter ต่าง ๆ ใน filter set

Filter set	#Entry	Source prefix	Destination prefix	จำนวนสูงสุดของค่า unique ที่แมทช์แต่ละแพ็กเก็ต
ACL1	748	280	109	2
ACL3	2410	546	480	3
ACL5	4562	1179	306	2
FW1	286	87	43	3
FW4	270	113	38	3
IPC1	1724	637	380	4
IPC2	192	55	56	2

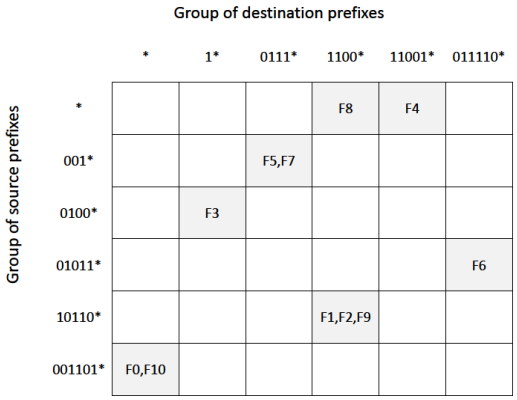
จากสมมติฐานนี้ทำให้ผู้วิจัยได้พัฒนาโครงสร้างข้อมูลและขั้นตอนวิธีของการคัดแยกแพกเก็ตแบบ 2 มิติ โดยแทนที่จะนำ filter ทั้งหมดใน filter set มาสร้างโครงสร้างข้อมูล ผู้วิจัยได้ทำการจัดกลุ่ม prefix ของ filter แยกกันในแต่ละมิติก่อน จากนั้นจึงนำผลที่ได้มาสร้างโครงสร้างข้อมูล ซึ่งทำให้ลดขนาดของหน่วยความจำที่ใช้และส่งผลถึงความสามารถในการคัดแยกแพ็กเก็ตที่รวดเร็วอันเนื่องมาจากโครงสร้างข้อมูลที่มีขนาดเล็ก สำหรับรายละเอียดของสิ่งที่เสนอในงานวิจัยจะถูกแยกออกเป็น 3 ส่วน ดังนี้ คือ (1) กระบวนการจัดกลุ่ม filter set เพื่อนำมาใช้ในการสร้างโครงสร้างข้อมูล (2) รายละเอียดโครงสร้างข้อมูล (3) ขั้นตอนวิธีในการคัดแยกแพ็กเก็ตตามค่า source/destination IP address ของแพ็กเก็ต

4.1 กระบวนการจัดกลุ่ม filter set

จากการจัดกลุ่ม filter ต่าง ๆ ในตารางที่ 1 ตามค่า prefix โดยแยกกันระหว่างส่วนของ source และ destination prefix ได้ผลดังแสดงในตารางที่ 3 เมื่อนำ filter เหล่านี้มาจัดลงในโครงสร้างข้อมูลแบบ cross-product จะได้ผลดังรูปที่ 1 โดยค่าภายใน cross-product จะเก็บหมายเลข filter (filter number) ที่อ้างถึง filter ที่มีค่า source prefix ตามแนว row และ destination prefix ตามแนว column โดยการค้นหา filter ที่แมทช์ จะเริ่มจากการแมทช์กลุ่มของ prefix ในส่วนของ source (ในด้าน row) และส่วนของ destination (ในด้าน column) ในลักษณะที่ทำคู่ขนานไปพร้อม ๆ กัน ซึ่งการแมทช์กลุ่มของ prefix สามารถใช้ขั้นตอนวิธีที่มีประสิทธิภาพ เช่น helix [20] หรือขั้นตอนวิธีอื่น ๆ ในการแมทช์ค่า prefix เมื่อทราบ row และ column ที่แมทช์แล้ว จะสามารถหา filter ที่แมทช์ได้จากตำแหน่งที่ cross กัน โดยถ้าการแมทช์ให้ผลลัพธ์มากกว่า 1 filter ผลลัพธ์สุดท้ายของการแมทช์ จะได้ filter ที่มีหมายเลขน้อยที่สุด ซึ่งเป็น filter ที่มีค่าความสำคัญสูงสุด

ตารางที่ 3 การจัดกลุ่มค่า source และ destination prefix ของ filter ในตารางที่ 1

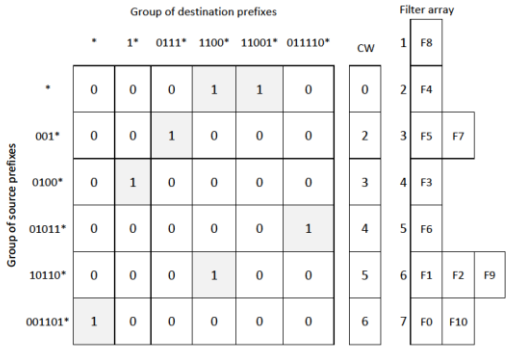
Source		Destination	
Prefixes	Filters	Prefixes	Filters
*	F4, F8	*	F0, F10
001*	F5, F7	1*	F3
0100*	F3	0111*	F5, F7
01011*	F6	1100*	F1, F2, F8, F9
10110*	F1, F2, F9	11001*	F4
001101*	F0, F10	011110*	F6



รูปที่ 1 โครงสร้างข้อมูลแบบ cross-product

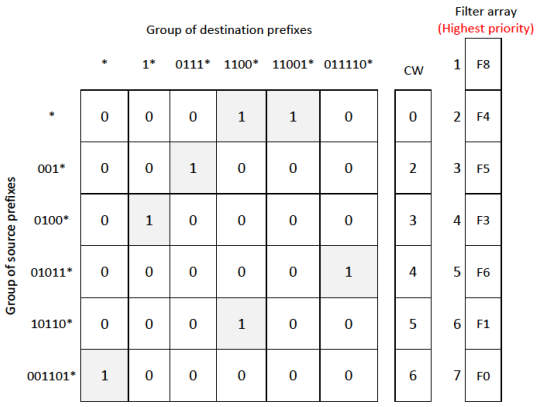
4.2 โครงสร้างข้อมูล

แม้โครงสร้างข้อมูลในรูปที่ 1 จะเป็นโครงสร้างข้อมูลที่ไม่ซับซ้อนและมีความรวดเร็วในการคัดแยกแพ็กเก็ต แต่โครงสร้างข้อมูลนี้มีข้อเสียที่สำคัญ คือ เมื่อ filter set มีขนาดใหญ่ เช่น กรณีของ FW เมื่อทำการจัดกลุ่ม filter ตามค่า source และ destination prefix และจัด filter ลงในโครงสร้างแบบ cross-product จะพบว่าจำนวน filter ในแต่ละ entry มีเป็นจำนวนมาก ส่งผลให้ entry หนึ่ง ๆ ของ cross-product ใช้หน่วยความจำค่อนข้างสูง เพื่อรองรับกับจำนวน filter ที่มีมาก ซึ่งทำให้ขนาดโครงสร้างข้อมูลที่ใช้ในการคัดแยกแพ็กเก็ตมากขึ้นตามไปด้วย ดังนั้นผู้วิจัยจึงได้ปรับโครงสร้างข้อมูลใหม่ดังแสดงในรูปที่ 2 โดยแยกค่า filter ออกมาเก็บใน array ที่เรียกว่า filter array และเปลี่ยนโครงสร้าง cross-product ให้เป็นอาร์เรย์ 2 มิติ ของบิต โดยถ้า entry ใดมีค่าบิตเป็น 1 หมายถึง entry นั้นมี filter อยู่ แต่ถ้า entry ใดมีค่าบิตเป็น 0 แสดงว่าไม่มี filter อยู่ใน entry นั้น แต่ละ entry ที่มีค่าบิตเป็น 1 จะถูก map ไปยังแต่ละตำแหน่ง (แต่ละค่าของ index) ใน filter array ซึ่งเก็บ filter ที่สอดคล้องกับ cross-product นั้น ซึ่งการหาค่า index ของ filter array ที่เก็บ filter ที่สอดคล้องกับ cross-product หนึ่ง ๆ สามารถหาได้จากการพิจารณาว่า entry ที่มีค่าบิตเป็น 1 ของ cross-product นั้น เป็นบิต 1 ลำดับที่เท่าใดของแถว จากนั้นให้นำมาค่านี้มารวมกับค่าที่เก็บใน code word (CW) ที่อยู่ที่แถวดังกล่าว ตัวอย่าง เช่น บิต 1 ใน entry ของ cross-product (*, 11001*) เป็นบิต 1 ลำดับที่ 2 ของแถวแรก เมื่อนำ 2 มารวมกับค่า CW ของแถวแรกซึ่งเท่ากับ 0 จะได้ 2 + 0 = 2 ซึ่งก็คือค่า index ของ filter array ที่เก็บ *F4* ซึ่งเป็น filter ที่สอดคล้องกับ (*, 11001*)



รูปที่ 2 โครงสร้างข้อมูลแบบ cross-product ที่ปรับใหม่ โดยปรับโครงสร้างให้เป็นอาร์เรย์ 2 มิติของบิต

ในทางปฏิบัติ การแมทช์ค่า filter จะพิจารณา filter ที่แมทช์กับแพ็กเก็ตหนึ่ง ๆ ได้ดีที่สุดเพียงแค่ filter เดียว โดย router จะกระทำ action ตามที่ระบุใน filter นั้น ในงานวิจัยนี้ผู้วิจัยจะใช้ policy การคัดแยกแพ็กเก็ตที่อิงตามค่าความสำคัญ (priority) ของ filter โดยกำหนดให้ filter ที่มีหมายเลขน้อยมีค่าความสำคัญสูง filter ที่มีหมายเลขมากมีค่าความ สำคัญต่ำ filter ที่แมทช์กับแพ็กเก็ตได้ดีที่สุด คือ filter ที่มีหมายเลขน้อยที่สุดในบรรดา filter ที่แมทช์ทั้งหมด จากการกำหนด policy เช่นนี้ทำให้สามารถปรับ เปลี่ยนโครงสร้างข้อมูลในรูปที่ 2 ได้ดังในรูปที่ 3 โดยให้ filter array เก็บเฉพาะ filter ที่มีค่าความสำคัญสูงสุดเท่านั้น ซึ่งช่วยให้ขนาดของหน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูลลดลง



รูปที่ 3 โครงสร้างข้อมูลแบบ cross-product ที่เก็บเฉพาะ filter ที่มีค่าความสำคัญสูงสุด

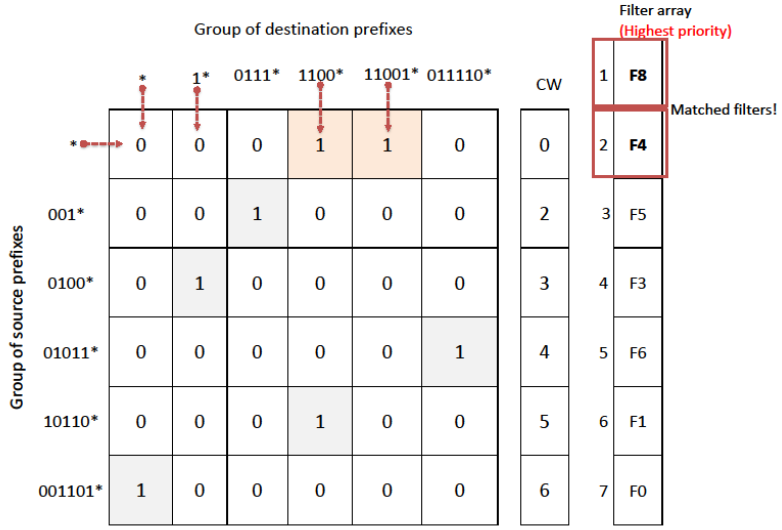
4.3 ขั้นตอนวิธีในการคัดแยกแพ็กเก็ต (IP classification algorithm)

เทคโนโลยีในปัจจุบันมีความก้าวหน้ามาก สามารถประมวลผลงานต่าง ๆ ภายในระยะเวลาอันสั้น โดยใช้วิธีประมวลผลแบบทำคู่ขนานกัน ซึ่งสามารถอธิบายได้อย่างง่าย ๆ คือ ในการประมวลผลข้อมูล จะทำการแบ่งงานให้ CPU แต่ละตัวประมวลผลไปพร้อม ๆ กัน (multi-processing) และนำผลลัพธ์จากการประมวลผลของ CPU แต่ละตัวมารวมกัน ซึ่งการประมวลผลดังกล่าว มีความสำคัญต่องานที่มีข้อมูลขนาดใหญ่ เพราะจะได้ผลลัพธ์ที่รวดเร็วกว่าวิธีประมวลผลแบบทำตามลำดับ (sequential computing)

งานวิจัยนี้ ได้ออกแบบให้การคัดแยกแพ็กเก็ตแต่ละมิติ (แต่ละฟิลด์) กระทำแบบคู่ขนานกัน กล่าวคือ การค้นหากลุ่มของ prefix ที่แมทช์ในส่วนของ source prefix และส่วนของ destination prefix จะกระทำแบบคู่ขนานไปพร้อม ๆ กัน ซึ่งสามารถอธิบายได้ด้วยตัวอย่างต่อไปนี้ สมมติว่าแพ็กเก็ต *P* ซึ่งมีเฮดเดอร์ = {source IP address, destination IP address} = {10101101, 11001010} วิ่งเข้ามาที่ router *R* การคัดแยก *P* ตามโครงสร้างข้อมูลในรูปที่ 3 จะเริ่มจากการค้นหากลุ่มของ prefix ที่แมทช์กับค่า {10101101, 11001010} ของ *P* แบบคู่ขนานไปพร้อม ๆ กัน ซึ่งจะพบว่าในมิติของ source prefix ค่าที่แมทช์จะมี * เพียงค่าเดียว ขณะที่ในมิติของ destination prefix ค่าที่แมทช์จะมี *, 1*, 1100* และ 11001* เมื่อนำค่าที่แมทช์ของแต่ละมิติมา cross กัน จะได้ว่าค่าที่แมทช์ทั้ง 2 มิติ คือ ค่า cross-product ที่ entry มีค่าบิตเท่ากับ 1 ซึ่งได้แก่ ค่า cross-product (*, 1100*) และ (*, 11001*) ดังแสดงในรูปที่ 4

บิต 1 ใน entry ของ (*, 1100*) เป็นบิต 1 ลำดับที่ 1 ของแถวแรก เมื่อนำ 1 มารวมกับค่า CW ของแถวแรกซึ่งเท่ากับ 0 จะได้ 1 + 0 = 1 ซึ่งก็คือค่า index ของ filter array ที่เก็บ *F8* ซึ่งเป็น filter ที่สอดคล้องกับ (*, 1100*) บิต 1 ใน entry ของ (*, 11001*) เป็นบิต 1 ลำดับที่ 2 ของแถวแรก เมื่อนำ 2 มารวมกับค่า CW ของแถวแรกซึ่งเท่ากับ 0 จะได้ 2 + 0 = 2 ซึ่งก็คือค่า index ของ filter array ที่เก็บ *F4*

ซึ่งเป็น filter ที่สอดคล้องกับ (*, 11001*) ซึ่งจะได้ว่า filter ที่แมทช์กับ *P* คือ *F8* และ *F4* แต่เนื่องจาก *F4* มีค่าความสำคัญที่สูงกว่า *F8* ดังนั้น *F4* จึงเป็น filter ที่แมทช์ *P* ได้ดีที่สุด ดังนั้น *P* จะถูก Router *R* กระทำตาม Action ที่เป็นค่า Act_4



รูปที่ 4 ตัวอย่างการคัดแยกแพ็กเก็ตที่นำเสนอในงานวิจัย

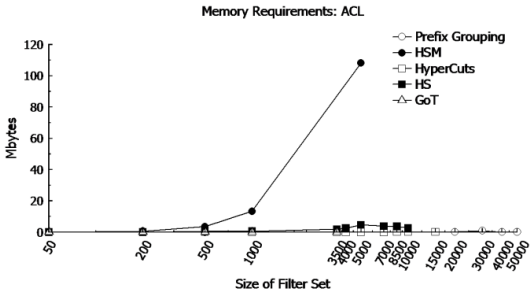
5. ผลการทดสอบโครงสร้างข้อมูลและขั้นตอนวิธี

ตามที่กล่าวไว้ในหัวข้อ 2.2 ในการประเมินประสิทธิภาพของการคัดแยกแพ็กเก็ต จะมีเกณฑ์ที่ใช้ในการพิจารณาอยู่ด้วยกัน 2 ประการ คือ ขนาดของหน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูล และจำนวนครั้งที่ใช้ในการอ่านหน่วยความจำขณะที่ขั้นตอนวิธีทำการค้นหา filter ที่แมทช์กับแพ็กเก็ต ซึ่งในการประเมินประสิทธิภาพการคัดแยกแพ็กเก็ตของงานวิจัยนี้ ผู้วิจัยได้ทำการทดสอบโดยเปรียบเทียบผลกับงานวิจัย Grid of Trie (GoT) [10] HyperCuts [16] HyperSplit (HS) [17] และ HSM [14] โดยใช้ filter set ที่ได้จาก ClassBench ซึ่งแบ่งออกเป็น 3 ประเภท คือ ACL, FW และ IPC โดยมีขนาดที่แตกต่างกันตั้งแต่ 50 ถึง 50,000 filters ในส่วนของการทดสอบกระบวนการคัดแยกแพ็กเก็ต ผู้วิจัยได้ใช้ชุดข้อมูลแพ็กเก็ต (packet trace) ที่ ClassBench สร้างขึ้นเพื่อให้ใช้สำหรับทดสอบการคัดแยกแพ็กเก็ตโดยตรง

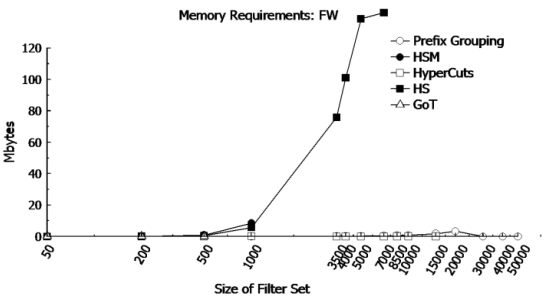
5.1 ขนาดของหน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูล (memory requirement)

เนื่องจากงานวิจัยนี้ (prefix grouping) มีการจัด filter ออกเป็นกลุ่มตามค่าของ prefix เพื่อลดจำนวนของข้อมูลที่จะนำมาจัดลงในโครงสร้างข้อมูล และมีการปรับโครงสร้างข้อมูลในส่วนของ cross-product ให้เป็นอาร์เรย์ของบิต จึงทำให้หน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูลมีค่าน้อยมาก โดยเมื่อเทียบกับหน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูลของงานวิจัยอื่น ๆ ที่ใช้ filter set เดียวกัน จะพบว่าหน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูลของงานวิจัยอื่น ๆ จะมีค่าเพิ่มขึ้นอย่างมีนัยสำคัญเมื่อขนาดของ

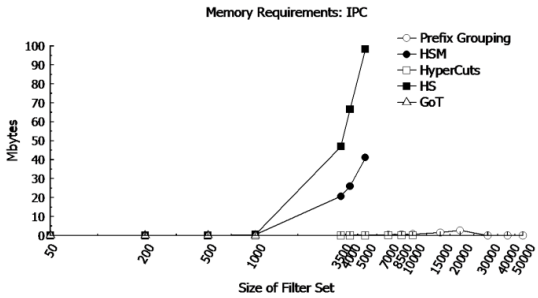
filter set เพิ่มขึ้น (ดังแสดงรูปที่ 5 ถึง 7) โดยเฉพาะ HSM และ HS ซึ่งมีการเพิ่มขึ้นของหน่วยความจำที่ใช้แบบ exponential ดังจะเห็นได้จากรูปที่ 7 ที่ filter set ขนาด 5,000 filters พบว่า HS ใช้หน่วยความจำสูงถึง 100 MB ส่วนงานวิจัย GoT และ HyperCuts แม้จะใช้หน่วยความจำไม่มากเมื่อเทียบกับ HSM และ HS แต่จากการทดสอบของผู้วิจัยพบว่า ทั้งสองงานวิจัยนี้ ไม่สามารถสร้างโครงสร้างข้อมูลเพื่อรองรับ filter set ที่มีขนาดใหญ่ได้ ด้วยเหตุนี้ จึงไม่มีข้อมูลผลการทดสอบของทั้งสองงานวิจัยนี้เมื่อ filter set มีขนาดใหญ่ โดยจากตัวอย่างในรูปที่ 8 จะพบว่า GoT สามารถสร้างโครงสร้างข้อมูลเพื่อรองรับ filter set ที่มีขนาดเพียงแค่ 3,500 filter เท่านั้น ขณะที่ในรูปที่ 7 ก็จะพบเช่นเดียวกันว่า HyperCuts สามารถสร้างโครงสร้างข้อมูลเพื่อรองรับ filter set สูงสุดได้เพียง 8,500 filters ขณะที่ทำการทดสอบผล ผู้วิจัยยังพบอีกว่า filter set บางประเภทมีผลต่อหน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูลของงานวิจัยบางงาน เช่น filter set ประเภท FW มีผลต่องานวิจัย HS ที่ไม่สามารถสร้าง filter set ที่มีขนาดมากกว่า 7,000 filters (ดังแสดงในรูปที่ 6) หรือ filter set ประเภท IPC มีผลต่องานวิจัย HSM ที่ไม่สามารถสร้าง filter set ที่มีขนาดมากกว่า 5,000 filters (ดังแสดงในรูปที่ 7) เป็นต้น ซึ่งสามารถอธิบายถึงสาเหตุได้ว่าเนื่องจาก filter แต่ละประเภทจะมีการกระจายตัวของ prefix length (prefix length distribution) ของ prefix ของ filter แตกต่างกัน เมื่อนำค่า prefix เหล่านี้มาจัดเก็บลงในโครงสร้างข้อมูล จะมีผลทำให้โครงสร้างข้อมูลมีความซับซ้อนมากน้อยต่างกัน [17] ซึ่งเมื่อเปรียบเทียบประสิทธิภาพกับงานวิจัยนี้ จะพบว่าทั้งจำนวน filter และประเภทของ filter ไม่มีผลใด ๆ ต่อโครงสร้างข้อมูลที่ออกแบบ กล่าวคือ โครงสร้างข้อมูลของงานวิจัยนี้สามารถรองรับ filter ได้ทุกประเภทที่มีการใช้งานจริง และรองรับกับ filter set ที่มีขนาดใหญ่ได้ โดยใช้หน่วยความจำสูงสุดไม่เกิน 5 MB เท่านั้นซึ่งถือว่าน้อยมาก โดยจากรูปที่ 5 ถ้านำ HSM ออกจากการพิจารณา เพื่อขยายภาพของการใช้หน่วยความจำของงานวิจัยอื่น ๆ ที่เหลือ จะได้ผลดังแสดงในรูปที่ 8 ซึ่งจะเห็นว่างานวิจัยอื่น ๆ ที่เหลือมีการเพิ่มขึ้นของหน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูลแบบ Linear ตามขนาดของ filter set ที่เพิ่มขึ้น งานวิจัยนี้เป็นเพียงงานวิจัยเดียวที่การใช้หน่วยความจำสำหรับโครงสร้างข้อมูลไม่ได้เพิ่มตามขนาดของ filter set ทำให้เป็นงานวิจัยเดียวที่โครงสร้างข้อมูลสามารถรองรับ filter set ที่มีขนาดถึง 50,000 filters ได้โดยใช้หน่วยความจำประมาณ 5 MB เท่านั้น และมีขีดความสามารถในการรองรับการใช้งานที่ filter set ขนาดมากกว่า 50,000 filters ได้อีก จึงมีสมบัติในเรื่องของ scalability



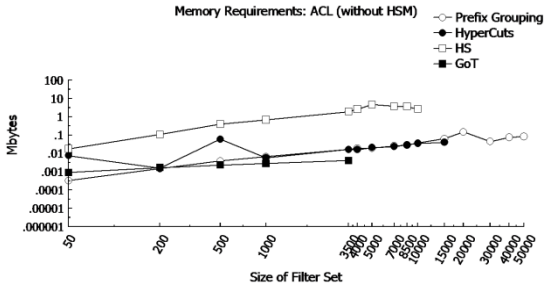
รูปที่ 5 หน่วยความจำที่ใช้สำหรับ filter ชนิด ACL



รูปที่ 6 หน่วยความจำที่ใช้สำหรับ filter ชนิด FW



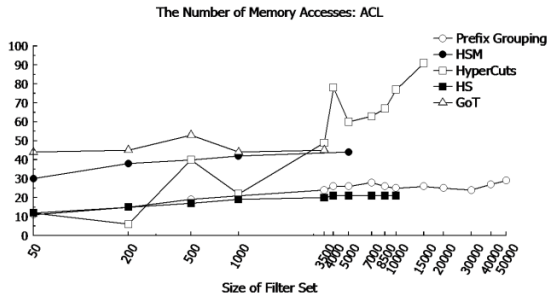
รูปที่ 7 หน่วยความจำที่ใช้สำหรับ filter ชนิด IPC



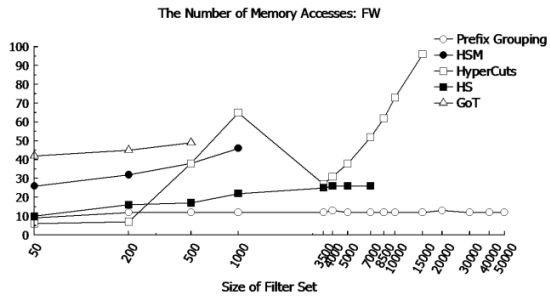
รูปที่ 8 หน่วยความจำที่ใช้สำหรับ filter ชนิด ACL (ไม่รวม HSM)

5.2 จำนวนครั้งที่ใช้ในการอ่านหน่วยความจำ (the number of memory accesses) ขณะทำการคัดแยกแพ็กเก็ต

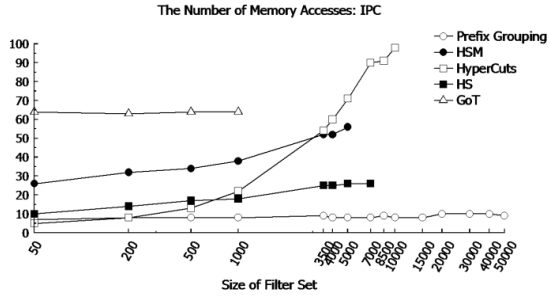
การนับจำนวนครั้งที่ใช้ในการอ่านหน่วยความจำขณะที่ขั้นตอนวิธีทำการค้นหา filter ที่แมทช์กับแพ็กเก็ต โดยคิดเทียบต่อ 1 แพ็กเก็ต ซึ่งได้ผลการทดสอบดังแสดงในรูปที่ 9 ถึง 11



รูปที่ 9 จำนวนครั้งในการอ่านหน่วยความจำสำหรับ filter ชนิด ACL



รูปที่ 10 จำนวนครั้งในการอ่านหน่วยความจำสำหรับ filter ชนิด FW



รูปที่ 11 จำนวนครั้งในการอ่านหน่วยความจำสำหรับ filter ชนิด IPC

จากผลการทดสอบพบว่าการคัดแยกแพ็กเก็ตแต่ละครั้ง งานวิจัยนี้ (prefix grouping) ใช้จำนวนครั้งในการอ่านหน่วยความจำน้อยมาก (โดยเฉลี่ยจะใช้เพียงแค่ 10 ถึง 26 ครั้ง โดยประมาณต่อการคัดแยก 1 แพ็กเก็ต) โดยเมื่อขนาดของ filter set เพิ่มขึ้น จำนวนครั้งในการอ่านหน่วยความจำมีค่าเพิ่มขึ้นเพียงเล็กน้อย (แทบจะคงที่) ทั้งนี้เป็นเพราะ ในการค้นหากลุ่มของ prefix ที่แมทช์ในส่วนของ source prefix และ destination prefix ได้ใช้วิธีการที่ทำแบบคู่ขนานไปพร้อม ๆ กัน และในช่วงของการคำนวณเพื่อหาค่า index ของ filter array จากโครงสร้าง cross-product เนื่องจากโครงสร้างนี้เก็บข้อมูลในรูปของบิต การอ่านหน่วยความจำในแต่ละครั้งสามารถอ่านข้อมูลในปริมาณที่มาก ทำให้การติดต่อกับโครงสร้างนี้ใช้จำนวนครั้งในการอ่านหน่วยความจำน้อย โดยเมื่อ

เปรียบเทียบผลกับงานวิจัย HyperCuts จะพบว่า เมื่อขนาดของ filter set เพิ่มขึ้น จำนวนครั้งในการอ่านหน่วยความจำจะมีค่าเพิ่มขึ้นอย่างเห็นได้ชัด ขณะที่งานวิจัย HSM และ HS จำนวนครั้งในการอ่านหน่วยความจำก็มีค่าเพิ่มขึ้นเช่นเดียวกัน แต่ไม่มากเท่ากับงานวิจัย HyperCuts โดยลักษณะการเพิ่มขึ้นจะเป็นแบบ linear ส่วนงานวิจัย GoT จะมีจำนวนครั้งในการอ่านหน่วยความจำสูงสุดไม่เกิน 64 ครั้ง (โดยเฉลี่ยอยู่ประมาณ 40 ถึง 64 ครั้ง) ซึ่งสอดคล้องกับลักษณะของโครงสร้างข้อมูลที่ใช้ในงานวิจัย ซึ่งเป็นแบบ trie 2 มิติ โดยความสูงของ trie จะมีค่าไม่เกิน 64

จากค่าของจำนวนครั้งที่ใช้ในการอ่านหน่วยความจำต่อการคัดแยก 1 แพ็กเก็ต สามารถนำมาหาอัตราการคัดแยกแพ็กเก็ต (packet classification rate) ได้ดังตัวอย่างต่อไปนี้ ถ้าการอ่านหน่วยความจำแบบ SRAM แต่ละครั้ง ใช้เวลา 10 ns (nanoseconds) [21] จากผลการทดสอบในรูปที่ 9 พบว่าที่ filter Set ขนาด 50,000 filters งานวิจัยนี้ใช้จำนวนครั้งในการอ่านหน่วยความจำ 20 ครั้ง ต่อการคัดแยก 1 แพ็กเก็ต ดังนั้นเวลาที่ใช้การคัดแยก 1 แพ็กเก็ตคิดเป็น 200 ns หรือกล่าวได้ว่าอัตราการคัดแยกแพ็กเก็ตคิดเป็น 1/200 ns = 5 ล้าน แพ็กเก็ตต่อ 1 วินาที ซึ่งถือว่าเร็วมาก

6. สรุปงานวิจัย

การพัฒนาโครงสร้างข้อมูลและขั้นตอนวิธีในการคัดแยกแพ็กเก็ตโดยใช้หน่วยความจำพื้นฐานแบบ SRAM ยังคงได้รับความสนใจในการศึกษาวิจัย เมื่อเทียบกับการใช้หน่วยความจำแบบ TCAMs อันเนื่องมาจากข้อจำกัดเกี่ยวกับขั้นตอนวิธีคัดแยกแพ็กเก็ตและข้อจำกัดทางด้านพลังงานและค่าใช้จ่ายในการใช้งาน TCAMs บทความนี้ได้นำเสนอการออกแบบโครงสร้างข้อมูลและขั้นตอนวิธีเพื่อใช้ในการคัดแยกแพ็กเก็ตแบบ 2 มิติ กล่าวคือ ในการคัดแยกแพ็กเก็ตจะพิจารณาข้อมูลเฉพาะในส่วนของ source IP address และ destination IP address ของแพ็กเก็ต ถึงแม้ว่าการคัดแยกแพ็กเก็ตแบบ 2 มิติ นี้ดูเหมือนว่าจะเป็นเพียงแค่ส่วนหนึ่งของการคัดแยกแพ็กเก็ตแบบ 5 มิติ แต่โดยตัวของมันเองก็ถือว่าเป็นปัญหาที่มีความสำคัญต่องานในระบบเครือข่ายอย่างมาก เนื่องจากสามารถประยุกต์กับการใช้งานจริงได้ในหลายกรณี ซึ่งการพัฒนาการคัดแยกแพ็กเก็ตแบบ 2 มิติ นี้ยังคงได้รับความสำคัญและมีการพัฒนาเรื่อยมา งานวิจัยนี้ได้นำเสนอโครงสร้างข้อมูลและขั้นตอนวิธีในการคัดแยกแพ็กเก็ตแบบ 2 มิติ โดยอาศัยการจัด filter ออกเป็นกลุ่มตามค่า prefix เพื่อลดจำนวนข้อมูลที่จะนำมาจัดลงในโครงสร้างข้อมูล และมีการปรับโครงสร้างข้อมูลในส่วนของ cross-product ให้เป็นอาร์เรย์ของบิต จึงทำให้หน่วยความจำที่ใช้สำหรับโครงสร้างข้อมูลมีค่าน้อยมาก สามารถรองรับ filter set ที่มีขนาดใหญ่ได้ (กล่าวคือ มีสมบัติในเรื่องของ scalability) นอกจากนี้ยังสามารถรองรับกับ filter set ได้ทุกประเภทอีกด้วย ในส่วนของขั้นตอนวิธีที่ใช้ในการคัดแยกแพ็กเก็ต เนื่องจากการค้นหากลุ่มของ prefix ที่แมทช์ในส่วนของ source prefix และ destination prefix ได้ถูกออกแบบให้ทำคู่ขนานไปพร้อม ๆ กัน และโครงสร้างในส่วนของ cross-product ได้ถูกออกแบบให้เก็บข้อมูลในรูปของบิต จึงทำให้การอ่านหน่วยความจำแต่ละครั้งสามารถอ่านข้อมูลในปริมาณที่มาก ทำให้การติดต่อกับโครงสร้างนี้ใช้จำนวนครั้งในการอ่านหน่วยความจำน้อย จากการทดสอบประสิทธิภาพของการคัดแยกแพ็กเก็ตพบว่าสามารถทำได้อย่างรวดเร็วและมีประสิทธิภาพโดยจำนวนครั้งที่ใช้ในการอ่านหน่วยความจำมีค่าเพิ่มขึ้นเพียงเล็กน้อย (แทบจะคงที่) เมื่อขนาดของ filter set เพิ่มขึ้น นอกจากนี้เนื่องจากการ

ค้นหา filter ที่แมทช์กับแพ็กเก็ตอาศัยเพียงการหาค่า index ของ filter array จากโครงสร้างของ cross-product ซึ่งการคำนวนค่า index สามารถทำได้อย่างง่ายดาย จึงทำให้การคัดแยกแพ็กเก็ตของงานวิจัยนี้สามารถนำไปประยุกต์เพื่อออกแบบและติดตั้งลงในอุปกรณ์ฮาร์ดแวร์ด้วยเทคโนโลยีที่มีอยู่ในปัจจุบันได้

7. เอกสารอ้างอิง

[1] Gupta, P. and McKeown, N., 1999, Packet classification on multiple fields, SIGCOMM Comput. Commun. Rev. 29: 147-160.

[2] Wencheng, L. and Sahni, S., 2006, Packet classification using pipelined two-dimensional multibit tries, pp. 808-813, Computers and Communications 2006 (ISCC' 06) Proceedings.

[3] Gupta, P. and McKeown, N., 2001, Algorithms for packet classification, Network, IEEE 15: 24-32.

[4] Dixit, M., Barbadekar, B.V. and Barbadekar, A.B., 2009, Packet classification algorithms, pp. 1407-1412, Industrial Electronics 2009 (ISIE 2009).

[5] Taylor, D.E., 2005, Survey and taxonomy of packet classification techniques, ACM Comput. Surv. 37: 238-275.

[6] Montoye, 1994, Don't Care in a Content Addressable Memory Cell, United States Patent 5,319,590, HaL Computer Systems, Inc., June 1994.

[7] Woo, T.Y.C., 2000, A modular approach to packet classification: Algorithms and results, IEEE Infocom.

[8] Kempke, M., 1998, Ternary CAM Memory Architecture and Methodology, United States Patent 5,841,874, Motorola, Inc., November, 1998.

[9] Gupta, P. and McKeown, N., 1999, Packet classification using hierarchical intelligent cuttings, pp. 34-41, Hot Interconnects VII.

[10] Srinivasan, V., Varghese, G., Suri, S. and Waldvogel, M., 1998, Fast and scalable layer four switching, SIGCOMM Comput. Commun. Rev. 28: 191-202.

[11] Baboescu, F., Sumeet, S. and Varghese, G., 2003, Packet classification for core routers: Is there an alternative to CAMs ?, INFOCOM 1: 53-63.

[12] Baboescu, F. and Varghese, G., 2005, Scalable packet classification, Networking 13: 2-14.

[13] Cheng, Y. and Wang, P., 2013, Packet classification using dynamically generated decision trees, Computers IEEE Transactions on.

[14] Xu, B., Jiang, D. and Li, J., 2005, HSM: A fast packet classification algorithm, pp. 987-992, Advanced Information Networking and Applications 2005 (AINA 2005).

[15] Chun-Hui, T., Hung-Mao, C. and Pi-Chung, W., 2013, Packet classification using multi-iteration RFC, pp. 748-753, Computer Software and Applications Conference Workshops (COMPSACW).

[16] Singh, S., Baboescu, F., Varghese, G. and Wang, J., 2003, Packet classification using

multidimensional cutting, Proceedings of the 2003 Conference on Applications, Technologies, Architectures and Protocols for Computer Communications, Karlsruhe, Germany.

[17] Qi, Y., Xu, L., Yang, B., Xue, Y. and Li, J., 2009, Packet classification algorithms: From theory to practice, pp. 648- 656, INFOCOM 2009, IEEE.

[18] Taylor, D.E. and Turner, J.S., 2005, Scalable packet classification using distributed crossproducing of field labels, INFOCOM 2005, . 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE 1: 269-280.

[19] Taylor, D.E. and Turner, J.S., 2005, ClassBench: A packet classification benchmark, INFOCOM 2005, 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE 3: 2068-2079.

[20] Rojas-Cessa, R., Kijkanjanarat, T., Thirapittayatakul, N. and Apipattanamontre, P., 2010, Practical and scalable method to perform IP lookup in one memory access time, Technical Report (THMNJ.TR201006), Dept. of Elect. and Comp. Eng., New Jersey Institute of Technology, Newark, NJ.

[21] Recap: Virtual Memory and Cache, Available Source: http://software.intel.com/en-us/articles/recap-virtual-memory-and-cache, Fabuary 20, 2013.