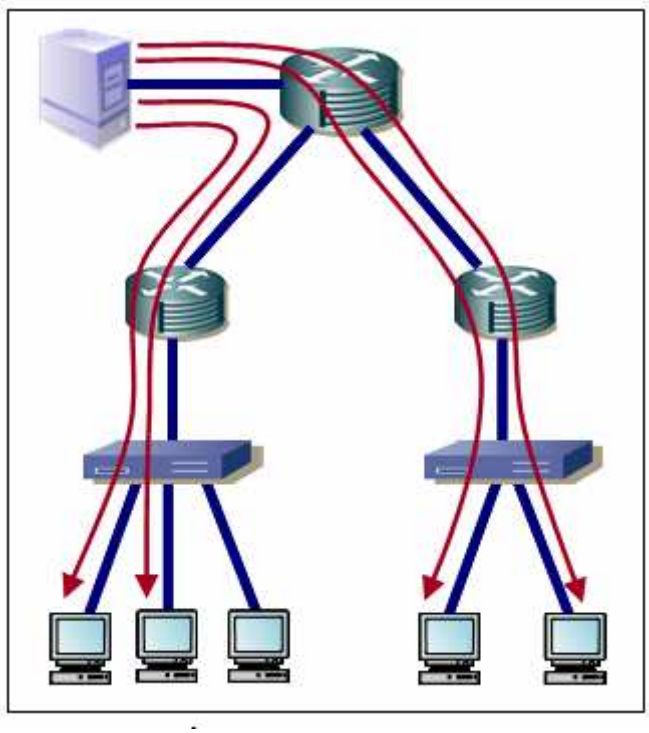
อินเทอร์เน็ต ตัวอุปกรณ์ NAT ก็จะแปลงไอพีจาก 192.168.2.12 ไปเป็น 203.44.135.9 ซึ่งถ้ามองจากเครือข่ายภายในแล้วจะเห็นว่า เครื่องในเครือข่ายภายในสามารถติดต่อออกไปยังเครือข่ายภายนอกได้โดยตรง ในขณะที่เครื่องจากภายนอกจะไม่สามารถติดต่อเข้ามาได้ถ้าเครื่องจากเครือข่ายภายในไม่ได้เป็นฝ่ายเริ่มต้นติดต่อก่อน และข้อมูลที่ส่งออกไปยังเครือข่ายภายนอกนั้นจะเป็นข้อมูลที่มีไอพีแอดเดรสต้นกำเนิดเป็นไอพีแอดเดรสสำหรับเครือข่ายภายนอก (203.44.135.9) ของอุปกรณ์ NAT [13]

2.1.2 ยูนิคาสต์ (Unicast)

การส่งข้อมูลแบบยูนิคาสต์เป็นการส่งแบบเจาะจงผู้รับเพียงเครื่องเดียว ในกรณีที่ต้องการส่งข้อมูลให้กับผู้รับหลายเครื่อง การส่งแบบยูนิคาสต์จะสร้างข้อมูลซ้ำตามจำนวนผู้รับ ดังภาพที่ 2.2 (แสดงการส่งข้อมูลแบบยูนิคาสต์) ซึ่งแสดงการส่งข้อมูลให้ผู้รับจำนวน 4 เครื่อง ผู้ส่งต้องสร้างชุดข้อมูลจำนวน 4 ชุดแล้วจึงส่งออกไป ถ้าเป็นการส่งข้อมูลมัลติมีเดียจะทำให้สิ้นเปลืองแบนด์วิดธ์เป็นอย่างมาก

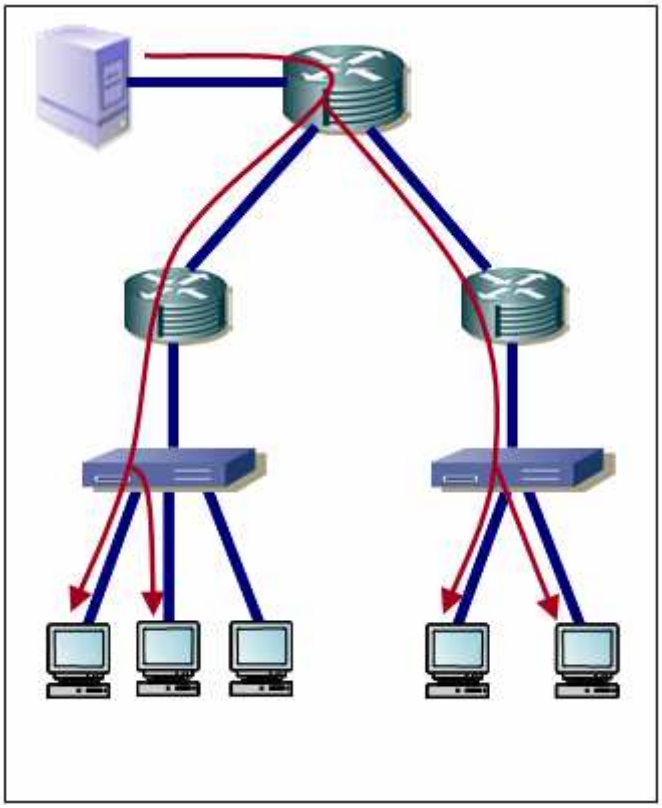
ภาพที่ 2.2 แสดงการส่งข้อมูลแบบยูนิคาสต์



2.1.3 มัลติคาสต์ (Multicast)

การส่งข้อมูลแบบมัลติคาสต์เป็นการแก้ไขปัญหาสิ้นเปลืองแบนด์วิดธ์ของการส่งข้อมูลแบบยูนิคาสต์ ในกรณีที่ต้องการส่งข้อมูลชุดเดียวไปยังผู้รับปลายทางหลายๆ เครื่อง ดังภาพที่ 2.3 (แสดงการส่งข้อมูลแบบมัลติคาสต์) การส่งข้อมูลแบบมัลติคาสต์จึงช่วยประหยัดแบนด์วิดธ์ เนื่องจากข้อมูลที่ถูกส่งออกไปมีเพียงชุดเดียว แต่ส่งไปถึงผู้รับได้หลายเครื่อง การนำส่งข้อมูลแบบมัลติคาสต์นี้จึงนำมาใช้ในการส่งข้อมูลที่เป็นมัลติมีเดีย ส่งทั้งภาพและเสียงไปพร้อมๆ กัน รวมถึงการประชุมทางไกลแบบแสดงภาพและเสียงด้วย

ภาพที่ 2.3 แสดงการส่งข้อมูลแบบมัลติคาสต์

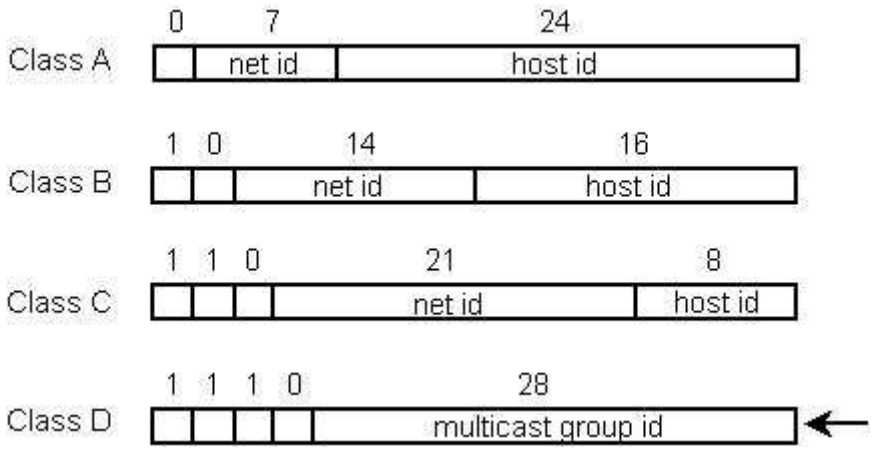


การส่งข้อมูลแบบมัลติคาสต์จะใช้มัลติคาสต์โปรโตคอล ซึ่งเป็นโปรโตคอลที่มีการสร้างไอพีอีกชุดหนึ่งเพื่อกำหนดแอดเดรสของผู้รับใหม่ โฮสต์ที่ต้องการรับข้อมูลจะร้องขอข้อมูลโดยไอพีของโฮสต์ที่ร้องขอจะถูกเก็บลงเป็น Distribution Tree ของระบบมัลติคาสต์ เมื่อข้อมูลมัลติคาสต์ถูกส่งออกไปในเครือข่าย เราเตอร์จะค้นหาว่าโฮสต์ที่รับข้อมูลอยู่ที่ใด จึงทำสำเนาข้อมูลนั้นแล้วส่งไปยังเครือข่ายนั้นๆ ต่อไปจนถึงผู้รับ เมื่อใดที่โฮสต์ไม่ต้องการข้อมูลมัลติคาสต์แล้ว ก็จะส่ง

คำร้องขอออกจากกลุ่ม กลไกในการเข้าร่วมกลุ่มหรือออกจากกลุ่มจะเกิดขึ้นตลอดเวลาในเครือข่าย

หน่วยงานควบคุมการกำหนดของไอพีแอดเดรสได้กำหนดให้ใช้ Class D สำหรับไอพีมัลติคาสต์ หมายความว่ากลุ่มของไอพีมัลติคาสต์แอดเดรสทั้งหมดจะอยู่ในช่วง 224.0.0.0 ถึง 239.255.255.255 โดยลักษณะไอพีแอดเดรสของ Class D นั้น จะไม่มีการแบ่ง net id และ host id ซึ่งแตกต่างจากไอพีแอดเดรสของ class A-C ดังภาพที่ 2.4 (แสดงไอพีแอดเดรสของ Class A-D)

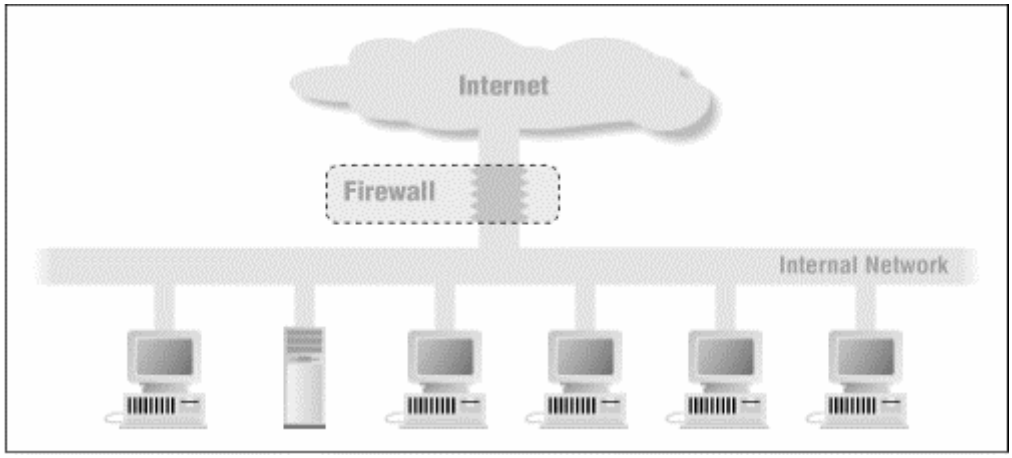
ภาพที่ 2.4 แสดงไอพีแอดเดรสของ Class A-D



2.1.4 ไฟร์วอลล์ (Firewall)

ไฟร์วอลล์ ในความหมายทางด้านคอมพิวเตอร์นั้น หมายถึง ระบบป้องกันอันตรายจากอินเทอร์เน็ตหรือเครือข่ายภายนอก เนื่องจากปัจจุบันอินเทอร์เน็ตมีบทบาทสำคัญต่อการดำเนินกิจกรรมต่างๆ เป็นอย่างมาก ไม่ว่าจะเป็นด้านการติดต่อสื่อสาร ธุรกิจ การศึกษา หรือว่าเพื่อความบันเทิง องค์กรต่างๆ ทั้งภาครัฐและเอกชน ต่างก็นำเอาเครือข่ายของตนเชื่อมต่อเข้ากับอินเทอร์เน็ตเพื่อที่จะได้รับประโยชน์เหล่านี้ ซึ่งการนำเอาเครือข่ายไปเชื่อมต่อกับอินเทอร์เน็ตนั้นทำให้ทุกคนที่ใช้อินเทอร์เน็ตสามารถเข้ามายังเครือข่ายนั้นๆ ได้ ปัญหาที่ตามมาก็คือความเสี่ยงในเรื่องความปลอดภัยของเครือข่ายที่อาจถูกเจาะระบบ หรือขโมยข้อมูลได้

ภาพที่ 2.5 แสดงไฟร์วอลล์กั้นระหว่างอินเทอร์เน็ตกับเครือข่ายภายใน [13]

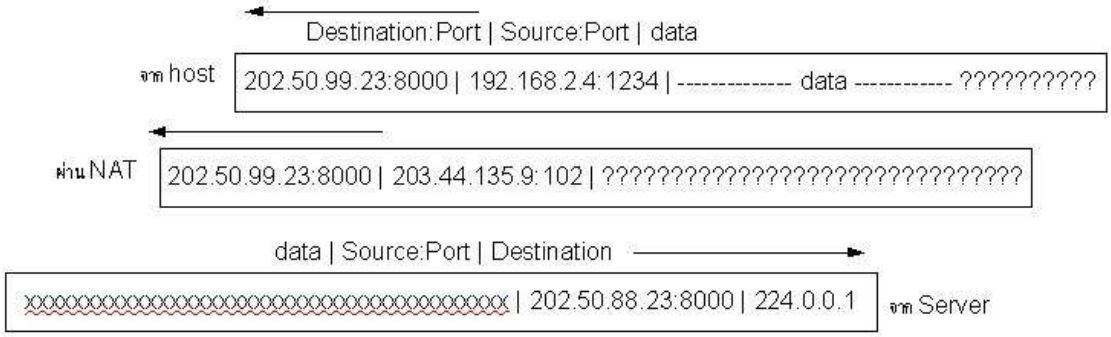


ไฟร์วอลล์เป็นคอมโพเน็นต์หรือกลุ่มของคอมโพเน็นต์ทำหน้าที่ในการควบคุมการเข้าถึงระหว่างเครือข่ายภายนอกหรือเครือข่ายที่คาดว่าไม่ปลอดภัย กับเครือข่ายภายในหรือเครือข่ายที่เราต้องการจะป้องกัน ดังภาพที่ 2.5 (แสดงไฟร์วอลล์กั้นระหว่างอินเทอร์เน็ตกับเครือข่ายภายใน) โดยที่คอมโพเน็นต์นั้นอาจจะเป็นเราเตอร์ เครื่องคอมพิวเตอร์ หรือเครือข่ายประกอบกัน ขึ้นอยู่กับวิธีการหรือสถาปัตยกรรมของไฟร์วอลล์ที่ใช้ [13] ซึ่งไฟร์วอลล์ที่ใช้ในระบบปฏิบัติการ ลินุกซ์ส่วนใหญ่คือ iptables

2.1.5 ปัญหาและข้อจำกัดของการส่งข้อมูลแบบมัลติคาสต์ผ่าน NAT

การส่งข้อมูลแบบมัลติคาสต์เป็นการแก้ไขปัญหาการสิ้นเปลืองแบนด์วิดธ์ของการส่งแบบยูนิคาสต์ แต่การนำ NAT เข้ามาใช้ ทำให้แพ็กเก็ตของมัลติคาสต์ไม่สามารถส่งผ่าน NAT เข้าไปได้ ดังภาพที่ 2.6 (แสดงส่วนหัวของแพ็กเก็ตที่ส่ง เข้า/ออก ผ่าน NAT) เนื่องจากข้อมูลแอดเดรสปลายทางในแพ็กเก็ตที่ส่งมาจากมัลติคาสต์เป็นมัลติคาสต์แอดเดรส ซึ่งเป็นแอดเดรสที่ NAT Server ไม่สามารถค้นหาโฮสต์ปลายทางได้ NAT จึงทิ้งแพ็กเก็ต

ภาพที่ 2.6 แสดงส่วนหัวของแพ็กเก็ตที่ส่ง เข้า/ออก ผ่าน NAT



จากปัญหาดังกล่าวจึงเป็นสาเหตุให้โฮสต์ที่อยู่หลัง NAT นั้นไม่สามารถรับข้อมูลแบบมัลติคาสต์ได้ ดังนั้นถ้าสามารถส่งข้อมูลแบบมัลติคาสต์ผ่าน NAT ได้ก็จะช่วยให้เกิดความสะดวกในองค์กร ทรัพยากรระบบและเวลาอีกด้วย

2.2 งานวิจัยที่เกี่ยวข้อง

ที่ผ่านมาได้มีการศึกษาปัญหาที่เครื่องลูกข่ายหลัง NAT server ไม่สามารถติดต่อสื่อสารกับเครื่องลูกข่ายหลัง NAT server อื่นได้ ปัจจุบันได้มีวิธีการต่างๆ ที่จะแก้ไขปัญหานี้ได้แก่ การใช้ Proxy server เป็นการออกแบบโดยใช้โปรโตคอลเริ่มเซสชัน (SIP) เพื่อเริ่มการติดต่อโปรโตคอลที่ใช้ในการตรวจหาชนิดของ NAT server และสามารถท่องไปตามเครื่องลูกข่ายแต่ละเครื่องหลัง NAT server (STUN) นอกจากนี้ยังมีการใช้ TURN เพื่อขยายความสามารถของ STUN ให้เชื่อมต่อเครื่องลูกข่ายที่อยู่หลัง NAT server ที่แตกต่างกันได้ หรือด้วยวิธีการแก้ไขที่ NAT server เอง เป็นต้น

ปัจจุบันอุปกรณ์ใหม่ๆ สามารถรองรับการทำงานของมัลติคาสต์ได้แล้ว แต่ยังไม่สามารถนำมาทดแทน NAT server ได้ เนื่องจากเครื่องที่ใช้เป็น NAT server ส่วนใหญ่เป็นเครื่องคอมพิวเตอร์ ซึ่งไม่ได้ทำหน้าที่เป็นแค่ NAT เท่านั้น แต่ยังรวมไปถึงไฟร์วอลล์, DHCP, DNS และใช้สำหรับการแชร์ไฟล์กันภายในด้วย ดังนั้นการที่จะเปลี่ยนไปใช้เราเตอร์มาทำงานแทน NAT นั้นจึงเป็นเรื่องที่ยาก อีกทั้งเราเตอร์แต่ละยี่ห้อก็มีการทำงานเฉพาะตามมาตรฐานของผู้ผลิต การที่ผู้ใช้จะแก้ไขหรือปรับแต่งให้เป็นไปตามที่ต้องการนั้นทำได้ยาก

ดังนั้นผู้วิจัยจึงเลือกใช้วิธีการในการแก้ไขโดยปรับแต่งที่เครื่อง NAT server เนื่องจากสามารถรองรับการทำงานของโปรแกรมต่างๆ ได้ทันที โดยที่ไม่ต้องแก้ไขโปรแกรมใหม่ แต่การ

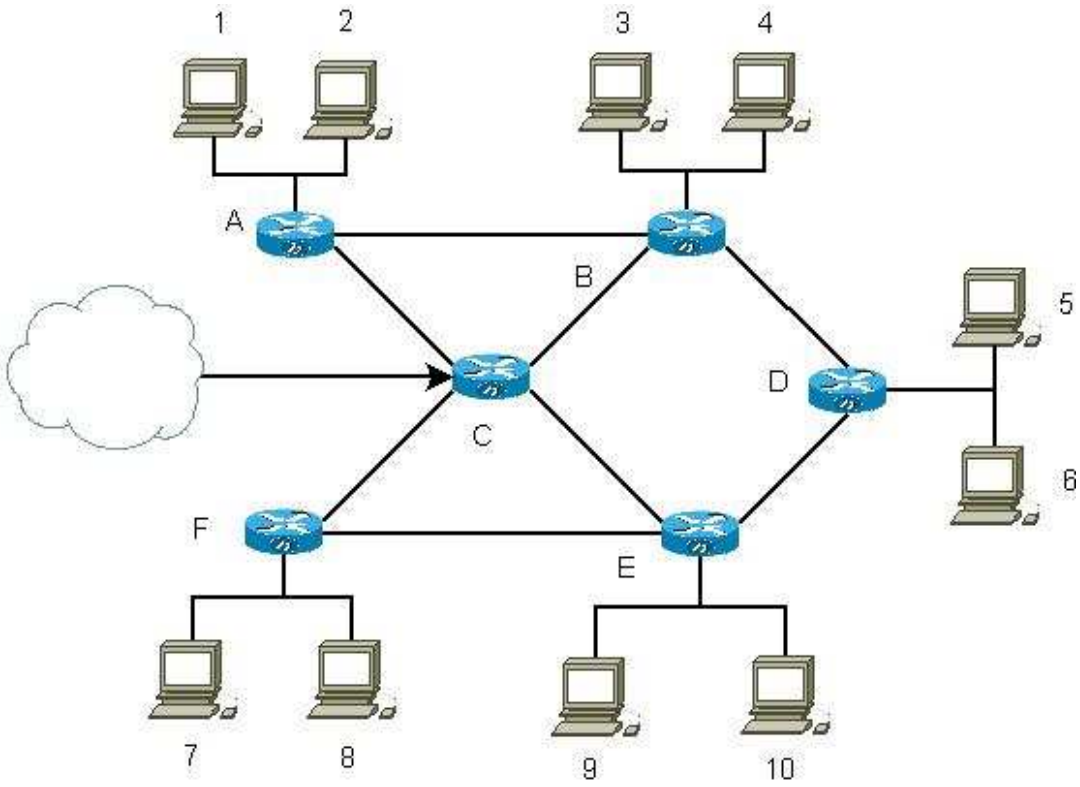
แก้ไขที่ NAT server นั้นก็มีความซับซ้อนมากด้วยเช่นกัน หากแก้ไขไม่ครอบคลุมอาจทำให้มีข้อบกพร่องจนถูกโจมตีจากภายนอกได้

2.2.1 Multicast Distribution Trees (MDT) [12]

MDT เป็นการเลือกเส้นทางของมัลติคาสต์เราเตอร์ในการส่งแพ็กเก็ตระหว่างเราเตอร์ด้วยกันเอง เพื่อให้มั่นใจว่าเราเตอร์แต่ละตัวได้รับมัลติคาสต์แพ็กเก็ตตามที่ต้องการ

จากภาพ 2.7 (ผังเครือข่ายระหว่าง router กับ host) โฮสต์เครื่องที่ 2, 4, 8 และ 10 เป็นสมาชิกกลุ่มมัลติคาสต์ ดังนั้น มัลติคาสต์เราเตอร์ที่ต่อโดยตรงกับโฮสต์เหล่านั้น ได้แก่ เราเตอร์ A, B, C, E และ F เท่านั้นที่ต้องการมัลติคาสต์แพ็กเก็ตจริงๆ

ภาพที่ 2.7 ผังเครือข่ายระหว่าง router กับ host



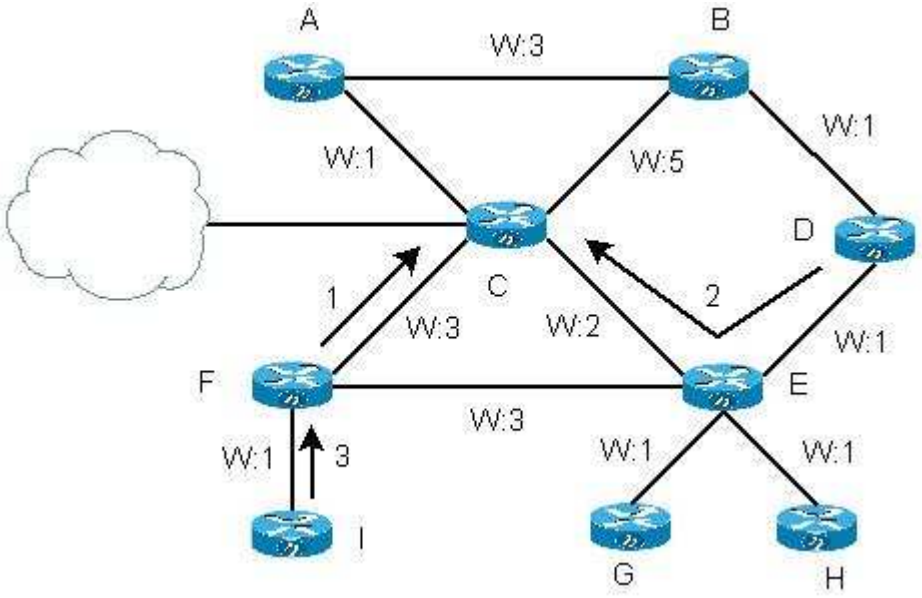
จุดประสงค์ที่สำคัญในการเลือกเส้นทางมัลติคาสต์ คือ การหาแขนงหรือกิ่งก้าน (tree) ที่เชื่อมต่อกับขอบเราเตอร์ทั้งหมดที่ติดกับโฮสต์ที่เป็นสมาชิกของกลุ่มมัลติคาสต์ โดยมัลติคาสต์แพ็กเก็ตจะเดินทางจากผู้ส่งไปยังทุกโฮสต์ที่อยู่ใน tree ตามเส้นทาง tree ที่สร้างไว้ ซึ่ง tree นั้นจะ

ประกอบด้วยเราเตอร์ที่ไม่ได้ติดต่อกับโฮสต์ที่เป็นสมาชิกของกลุ่มมัลติคาสต์โดยตรงด้วย (ตัวอย่างเช่น จากรูป ได้แก่ เราเตอร์ D) โดยในทางปฏิบัติมี 2 วิธีที่ใช้ในการสร้างเส้นทางของ Tree คือ Group - Shared Tree และ Source – Based Tree โดย Group – Shared จะใช้เส้นทางในการกระจายแพ็กเก็ตเพียงเส้นทางเดียวสำหรับผู้ส่งทุกคนในกลุ่ม ขณะที่ Source – Based จะสร้างทางขึ้นมาสำหรับผู้ส่งแต่ละคนเป็นการเฉพาะ

Multicast Routing Using a Group – Shared Tree

เนื่องจากวิธีนี้ไม่ว่าผู้ส่งจะเป็นใครก็ตามจะต้องใช้ Tree เดียวกันในการส่งแพ็กเก็ตถึงผู้รับ ดังนั้นจึงต้องหา Tree ที่เชื่อมต่อกับขอบเราเตอร์ (เราเตอร์ที่ต่อกับโฮสต์ที่เป็นสมาชิกของกลุ่มมัลติคาสต์โดยตรง) ทุกตัว ซึ่งวิธีที่นิยมใช้ในการเลือกเส้นทางสำหรับสร้าง Tree คือ Center – Based approach โดยการกำหนด Center node หรือ Rendezvous point หรือ Core ขึ้นมา เริ่มด้วยขอบเราเตอร์ส่ง join message มาที่ center node โดยที่ join message ที่ส่งมานั้นจะถูกส่งต่อไปตามเราเตอร์ต่างๆ ด้วยวิธียูนิคาสต์จนกว่าจะถึง center node หรือเราเตอร์ที่เป็นส่วนหนึ่งของ tree และสำหรับเส้นทางที่ join message ใช้เดินทางมายัง center node ก็จะกลายเป็นกิ่งหนึ่งของ tree นั้นไป (เชื่อมระหว่างขอบเราเตอร์ที่เป็นผู้ส่ง join message นั้นมากับ center node) ซึ่งเส้นทางใหม่นี้จะถูกต่อเข้ากับกิ่งหนึ่งของ tree นั้นที่มีอยู่ก่อนแล้ว ตัวอย่างเช่น จากภาพ 2.8 (Multicast Routing ด้วยวิธี Shared Tree) ให้เราเตอร์ C เป็น center node ของ tree และโหนด F เป็นสมาชิกของกลุ่มมัลติคาสต์ ซึ่งส่ง join message ให้ C ดังนั้น เส้นเชื่อมต่อ CF จึงกลายเป็นกิ่งแรกของ tree จากนั้น โหนด D เข้าร่วม tree โดยการส่ง join message ให้ C ซึ่งจะต้องส่งผ่านโหนด E ดังนั้นเส้นทาง CED จึงต่อเข้ากับ tree เดิมกลายเป็นกิ่งใหม่อีกกิ่งหนึ่งของ tree สุดท้ายโหนด I ส่ง join message มาให้ C โดยส่งผ่านมาทางโหนด F แต่เนื่องจากโหนด F เป็นส่วนหนึ่งของ tree อยู่แล้ว ดังนั้นเมื่อ join message เดินทางมาถึงโหนด F เส้นเชื่อมต่อ CF จึงต่อเข้ากับ tree โดยทันที (โดยเป็นการต่อกิ่ง CF ออกไปเป็น CFI)

ภาพที่ 2.8 Multicast Routing ด้วยวิธี Shared Tree



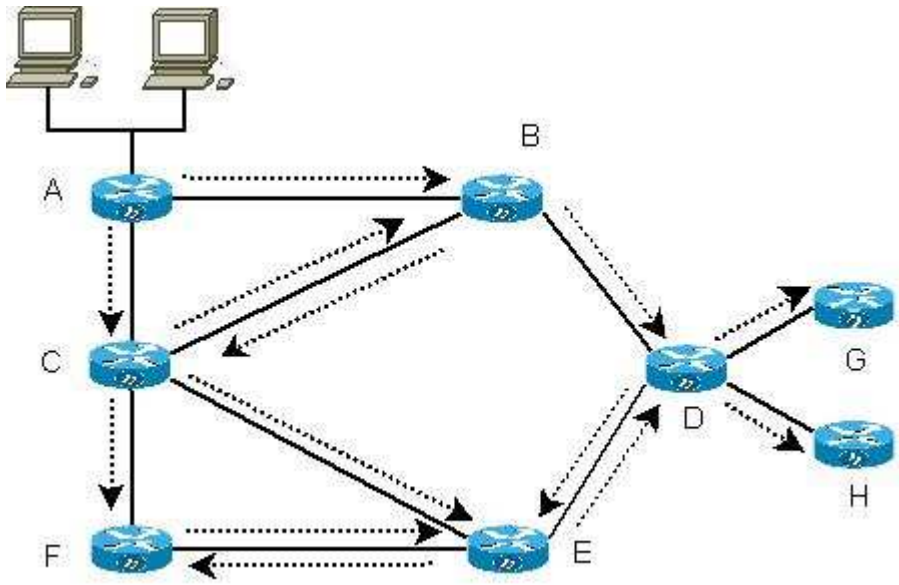
Group – Shared Tree มีข้อได้เปรียบในเรื่องของพื้นที่ที่ใช้เก็บสถานะในแต่ละเราเตอร์น้อย แต่สำหรับข้อเสียของวิธีนี้คือ ในบางโอกาสเส้นทางระหว่างผู้ส่งกับผู้รับอาจไม่ใช่เส้นทางที่เหมาะสมที่สุด ซึ่งอาจทำให้เกิดความล่าช้าในการขนส่งแพ็กเก็ตและสำหรับผู้ออกแบบเครือข่ายควรต้องคำนึงถึงการเลือก center node ด้วย

2.2.2 Multicast Routing Using a Source-Based Tree (SBT) [12]

ขณะที่วิธี Group-Shared Tree สร้าง tree เพียงเส้นทางเดียวไม่ว่าจะใช้ส่งแพ็กเก็ตที่มาจากผู้ส่งคนใดก็ตาม แต่วิธี Source - Based Tree จะสร้าง tree สำหรับผู้ส่งแต่ละคนในกลุ่มมัลติคาสต์ในทางปฏิบัติจะใช้วิธี Reverse Path Forwarding (RPF) Algorithm ในการสร้าง tree เมื่อเราเตอร์ได้รับมัลติคาสต์แพ็กเก็ตซึ่งมีที่อยู่ผู้ส่งระบุอยู่ด้วยนั้นเราเตอร์จะส่งต่อแพ็กเก็ตออกไปยังทุกเส้นเชื่อมที่ต่อกับตน (ยกเว้น เส้นเชื่อมด้านที่ได้รับแพ็กเก็ตเข้ามา) แต่เราเตอร์จะยอมรับและส่งต่อแพ็กเก็ตนั้นก็ต่อเมื่อเราเตอร์ได้รับแพ็กเก็ตนั้นจากเส้นเชื่อมที่มีเส้นทางที่สั้นที่สุดวัดจากตัวเองกลับไปหาผู้ส่งเท่านั้น (shortest path back to the sender) ถ้าเราเตอร์ ได้รับแพ็กเก็ตจากเส้นเชื่อมอื่น หรือได้รับแพ็กเก็ตซ้ำ เราเตอร์ก็จะไม่สนใจแพ็กเก็ตนั้น และไม่ส่งต่อด้วย วิธีนี้เราเตอร์ไม่ต้องการรู้ว่าเส้นทางที่สั้นที่สุดของตนเองจนถึงต้นกำเนิดแพ็กเก็ตคือเส้นทางใด แต่จะรู้เพียงแค่เส้นทางที่สั้นที่สุดของตนเองกับผู้ส่งที่อยู่ติดกันเท่านั้น จากภาพ 2.9 (Multicast Routing

ด้วยวิธี Based Tree) เราเตอร์ A ส่งต่อ source-S packet ไปให้เราเตอร์ C และ B จากนั้น เราเตอร์ B จะส่งต่อแพ็กเก็ตที่ได้รับจาก A (เพราะ A เป็นเส้นทางที่สั้นที่สุด เมื่อวัดจาก B ไปยังผู้ส่ง) ไปยังเราเตอร์ C และ D หลังจากนั้น B จะไม่สนใจ source-S packets ที่ได้รับจากเราเตอร์อื่นๆ อีก เมื่อพิจารณาที่เราเตอร์ C ซึ่งได้รับ source-S packet โดยตรงจาก A และ B แต่เนื่องจาก B ไม่ใช่เส้นเชื่อมที่สั้นที่สุด เมื่อวัดจาก C ไปยังผู้ส่ง C จึงไม่สนใจ (ทิ้ง) แพ็กเก็ตที่ได้รับจาก B และในขณะเดียวกัน C จะรับแพ็กเก็ตที่ได้รับโดยตรงจาก A และส่งต่อแพ็กเก็ตนั้นไปยังเราเตอร์ B, E และ F

ภาพที่ 2.9 Multicast Routing ด้วยวิธี Based Tree



RPF เป็นวิธีที่ดี แต่ถ้าพิจารณาที่เราเตอร์ D ซึ่งจะต้องส่งต่อแพ็กเก็ตไปให้เราเตอร์ G และ H ถึงแม้ว่าเราเตอร์ G และ H จะไม่ติดต่อกับโฮสต์ที่เป็นสมาชิกของกลุ่มมัลติคาสต์ก็ตาม เนื่องจากตัวอย่างนี้ เราเตอร์ที่ถัดไปจาก D มีเพียงแค่เราเตอร์ G และ H จึงไม่เกิดปัญหามากนัก แต่ถ้าสมมติว่าเส้นเชื่อมที่ต่อจาก D มีจำนวนมาก ซึ่งต่างก็ต้องได้รับมัลติคาสต์แพ็กเก็ตที่ไม่ต้องการจากเราเตอร์ D ดังนั้นจึงต้องมีวิธีการที่ใช้แก้ปัญหาการได้รับมัลติคาสต์แพ็กเก็ตที่ไม่ต้องการ ภายใต้วิธี RPF ซึ่งเรียกว่า prunning นั่นคือ เมื่อมัลติคาสต์เราเตอร์ได้รับมัลติคาสต์แพ็กเก็ตในขณะที่ตนเองไม่ได้เชื่อมต่อกับโฮสต์ที่เป็นสมาชิกของกลุ่มมัลติคาสต์นั้น เราเตอร์จะส่ง prune message กลับไปยังเราเตอร์ต้นทางที่ส่งแพ็กเก็ตมาให้ และเมื่อเราเตอร์ได้รับ prune

message จาก downstream router มันก็จะส่งต่อ prune message ไปยัง upstream router ของมันเองด้วย

Source – Based Tree มีข้อได้เปรียบตรงที่สามารถสร้างเส้นทางที่เหมาะสมระหว่างผู้ส่งกับผู้รับ ซึ่งสิ่งนี้จะช่วยการันตีว่าจะใช้เวลาในการส่งข้อมูลน้อย แต่ในการหาเส้นทางที่เหมาะสมที่สุดก็มีต้นทุนสูงเพราะว่าเราเตอร์จะต้องเก็บรักษาข้อมูลสำหรับผู้ส่งแต่ละราย ซึ่งถ้าเครือข่ายมีจำนวนผู้ส่งเป็นพันๆ เครื่อง และยังมีจำนวนกลุ่มเป็นพันๆ กลุ่ม จะยิ่งช่วยเร่งให้เกิดปัญหาทรัพยากรของเราเตอร์ไม่เพียงพอ ดังนั้นในการออกแบบเครือข่าย ผู้ออกแบบเครือข่ายจะต้องพิจารณาถึงเรื่องของความสิ้นเปลืองของหน่วยความจำที่เกิดจากขนาดของตารางเลือกเส้นทางมัลติคาสต์ด้วย

สมาชิกของกลุ่มมัลติคาสต์สามารถเข้าหรือออกจากกลุ่มได้ตลอดเวลา ดังนั้น Distribution tree จึงต้องถูกปรับปรุงอยู่ตลอดเวลา เมื่อผู้รับทุกคนในกิ่งๆ หนึ่งของ tree ยกเลิกการเป็นสมาชิกกลุ่ม เราเตอร์จะตัดกิ่งนั้นออกจาก distribution tree และหยุดส่งต่อข้อมูลไปยังกิ่งนั้น แต่ถ้าผู้รับคนใดคนหนึ่งเพียงหนึ่งคนในกิ่งนั้นกลับมาเป็นสมาชิกของกลุ่มอีก เราเตอร์ก็จะปรับปรุง distribution tree อีกครั้ง และเริ่มส่งต่อข้อมูล

2.2.3 Protocol-Independent Multicast [12]

Protocol-Independent Multicast (PIM) routing protocol เป็นอีกหนึ่ง Internet multicast routing protocol ที่ใช้กันอย่างกว้างขวางในอินเทอร์เน็ตถูกพัฒนาขึ้นมาโดยกลุ่มงาน IETF เพื่อสร้างมาตรฐานโปรโตคอลเลือกเส้นทางมัลติคาสต์ (Multicast routing protocol) สำหรับอินเทอร์เน็ต โดยที่โปรโตคอลไม่ขึ้นอยู่กับโปรโตคอลเลือกเส้นทางแบบยูนิคาสต์ใดๆ PIM มีรูปแบบการทำงานสองประเภทคือ

- แบบหนาแน่น (dense mode) [1] เป็นการทำงานในสภาพที่มีสมาชิกลุ่มอยู่รวมกันอย่างหนานแน่นและมีแบนด์วิดธ์เหลือเป็นจำนวนมาก

- แบบกระจาย (sparse mode) [3] เป็นการทำงานในสภาพที่สมาชิกกลุ่มอยู่กระจายกันในอินเทอร์เน็ตและแบนด์วิดธ์ไม่จำเป็นต้องมีเหลือมาก แบบกระจายนี้อาจมีสมาชิกในกลุ่มน้อยหรือมากก็ได้

2.2.3.1 PIM Dense Mode มีหลักการทำงานคล้ายกับ DVMRP คือใช้วิธีบรอดคาสต์ย้อนกลับและตัดตอนเส้นทางที่ไม่มีสมาชิกออกไป ความแตกต่างหลักของ PIM Dense Mode กับ DVMRP คือ PIM Dense Mode อาศัยตารางเส้นทางที่เกิดจากโปรโตคอลยูนิคาสต์ใดๆ ที่ใช้งาน

อยู่ในขณะนั้นเพื่อปรับเปลี่ยนเส้นทางมัลติคาสต์ โดยที่ตัวเองไม่ได้ใช้หรือขึ้นกับโปรโตคอลเฉพาะตัวใด ต่างจาก DVMRP ที่ผนวกการทำงานของ RIP เป็นส่วนหนึ่งของโปรโตคอล หรือ MOSPF ที่นำ OSPF มาใช้ในตัวเอง

2.2.3.2 PIM Sparse Mode เหมาะสำหรับการใช้ในสภาพแวดล้อมที่มีการส่งข้อมูลจำนวนมากเป็นช่วงเวลาแต่ว่ามีผู้รับปลายทางจำนวนน้อย ตัวอย่างโปรแกรมประยุกต์ที่เหมาะกับ PIM Sparse Mode คือ การประชุมทางไกลผ่านเครือข่าย WAN ซึ่งการใช้บรอดคาสต์ทำให้สิ้นเปลืองแบนด์วิดธ์มาก PIM Sparse Mode ทำงานในรูปแบบที่แตกต่างกันจาก PIM Dense Mode สองลักษณะดังนี้

-เราเตอร์ที่เชื่อมระหว่างกันโดยใช้ PIM Sparse Mode ต้องส่งข้อความผ่าน IGMP ว่าต้องการรับข้อมูลมัลติคาสต์ มิฉะนั้นจะไม่มีการปล่อยข้อมูลไปยังเราเตอร์นั้น เมื่อเทียบกับ PIM Dense Mode แล้วเราเตอร์จะปล่อยข้อมูลมัลติคาสต์ออกไปตั้งแต่แรกเริ่มจนกว่าจะได้รับข้อความตัดตอน (prune message) ค่าโดยปริยายของ PIM Dense Mode จึงเป็นการปล่อยข้อมูลมัลติคาสต์ ในขณะที่ PIM Sparse Mode จะกันข้อมูล multicast จนกว่าจะมีการแจ้งขอโดยตรง

- แต่ละกลุ่มมัลติคาสต์จะมีเราเตอร์หนึ่งตัวทำหน้าที่เป็น "จุดนัดพบ" (rendezvous point) ระหว่างผู้ส่งกับผู้รับ เมื่อผู้ส่งต้องการส่งข้อมูลจะต้องส่งไปยังจุดนัดพบก่อนเสมอ และผู้รับก็ต้องเข้าร่วมกลุ่มโดยลงทะเบียนที่จุดนัดพบเช่นกัน

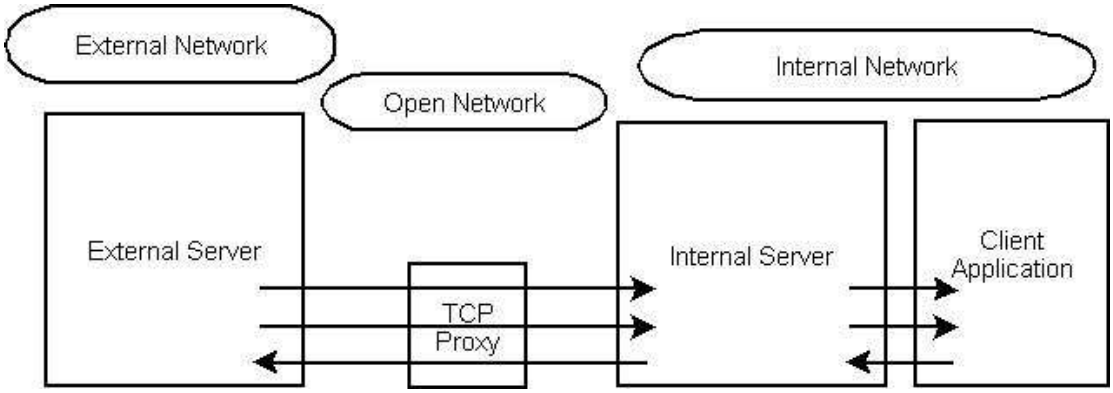
ลักษณะการทำงานของ PIM Sparse Mode คือ ข้อมูลมัลติคาสต์จะส่งผ่านจุดนัดพบประจำกลุ่มก่อนเสมอ หลังจากนั้นหากมีเส้นทางใหม่ที่เหมาะสมกว่าก็จะใช้เส้นทางนั้นแทน

ในปัจจุบันมีเราเตอร์บางชนิดสนับสนุน PIM นอกจากนี้ DVMRP ไม่ได้ออกแบบมาเพื่อใช้กับการเลือกเส้นทางแบบยูนิคาสต์ ดังนั้นเราเตอร์ที่ต้องเลือกเส้นทางทั้งยูนิคาสต์และมัลติคาสต์ต้องมีกระบวนการเลือกเส้นทางแยกออกจากกัน

2.2.4 UDP Proxy Solution - UDP Tunneling

แนวความคิดที่แบ่ง Proxy Server ออกเป็น 2 ส่วน เป็นแนวความคิดของ L. Oria [8] ที่ได้ศึกษาปัญหาระหว่างมัลติคาสต์กับไฟร์วอลล์ ซึ่งเป็นการแก้ปัญหาทางหนึ่งที่ให้ระบบสามารถอนุญาตหรือปฏิเสธการรับมัลติคาสต์แพ็กเก็ตผ่านไฟร์วอลล์ได้ โดยที่ Proxy server ด้านหนึ่งต่อกับเซิร์ฟเวอร์ภายใน ส่วนอีกด้านต่อกับเซิร์ฟเวอร์ภายนอก เพื่อที่จะทำการส่งมัลติคาสต์แพ็กเก็ตจากภายในไปสู่ภายนอกด้วยวิธี UDP Tunnel เป็นการเปิดเพื่อเชื่อมต่อระหว่างเซิร์ฟเวอร์ภายในและเซิร์ฟเวอร์ภายนอก ดังภาพที่ 2.10 (UDP Proxy with UDP Tunneling over TCP)

ภาพที่ 2.10 UDP Proxy with UDP Tunneling over TCP

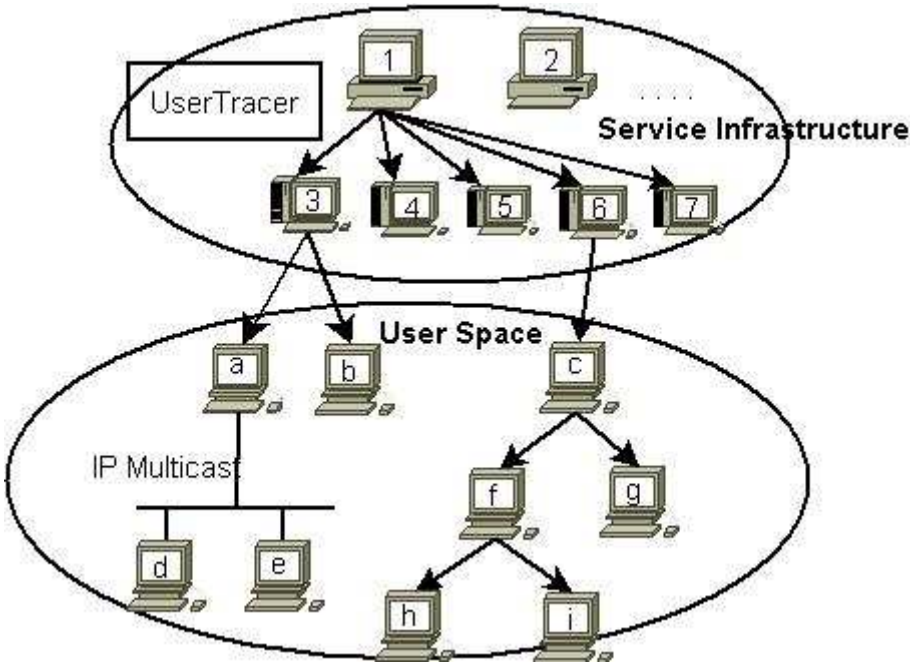


Tunnel จะเป็นการใช้วิธีพื้นฐานที่มีอยู่บน TCP Proxy ที่ตั้งอยู่บนโอเพ่นอีเทอร์เน็ต ซึ่งไคลเอ็นต์จะติดต่อกับเซิร์ฟเวอร์ภายใน และแลกเปลี่ยนข้อความที่ระบุโปรโตคอลสำหรับติดต่อระหว่าง proxy server และไคลเอ็นต์ ทางด้านโปรแกรมของ proxy client จะเปิด TCP Control ให้ติดต่อกับ proxy server ภายใน เมื่อโปรแกรมของ proxy client ต้องการเปิดการติดต่อกับกลุ่มของหมายเลขไอพีและพอร์ต ก็จะส่งข้อความพิเศษไปให้กับ proxy server ภายในโดยผ่าน TCP control connection กับไอพี และพอร์ตของโฮสต์ที่ต้องการเข้าร่วม เมื่อเซิร์ฟเวอร์ภายในเปิด UDP Tunnel ระหว่างตัวเซิร์ฟเวอร์ภายในกับเซิร์ฟเวอร์ภายนอกโดยการเปิด TCP Connection เฉพาะกับเซสชัน เมื่อเซิร์ฟเวอร์ภายนอกเป็นสมาชิกกับมัลติคาสต์เซสชันอย่างสมบูรณ์แล้ว เซิร์ฟเวอร์ภายนอกรับแอดเดรสของแพ็กเก็ต ที่เซสชันส่งมาแบบเข้ารหัส และส่งแพ็กเก็ต ให้กับเซิร์ฟเวอร์ภายใน เมื่อเซิร์ฟเวอร์ภายในได้รับแล้วก็จะถอดรหัสแพ็กเก็ตออกและส่งไปให้กับไคลเอ็นต์ผ่าน UDP Connection ที่เปิดไว้ก่อน และมีการเปิดโปรโตคอลสื่อสารอยู่แล้วระหว่างไคลเอ็นต์และ UDP proxy เมื่อเซิร์ฟเวอร์ภายในรับข้อมูลจากไคลเอ็นต์ผ่านโปรโตคอลการสื่อสารจะถูกเข้ารหัสและส่งให้กับเซิร์ฟเวอร์ภายนอก เมื่อเซิร์ฟเวอร์ภายนอกได้รับก็จะถอดรหัสแล้วส่งข้อมูลไปสู่อินเทอร์เน็ต

วิธีของ Loico นั้นทำให้ความสามารถของมัลติคาสต์หมดไป เนื่องจากการทำ tunneling นั้น มัลติคาสต์แพ็กเก็ตจะถูกเข้ารหัสใหม่และส่งออกจาก proxy server ในรูปของยูนิคาสต์แทน เป็นลักษณะเดียวกับงานวิจัยของ T. C. Wan, C. H. Leong, R. C. W. Thum [10] ที่ใช้รูปของแบบยูนิคาสต์ แต่การวิจัยชิ้นนี้จะแก้ปัญหาที่ NAT Server

2.2.5 HyMoNet

ภาพที่ 2.11 สถาปัตยกรรมของ HyMoNet



งานวิจัยของ B. Chang, Y. Shi และ N. Zhang [2] ได้ศึกษาถึงเรื่องการติดต่อมัลติคาสต์แพ็กเก็ตระหว่างไคลเอ็นต์ 2 ไคลเอ็นต์ ที่อยู่ภายใต้ NAT server ทั้งคู่ ภาพที่ 2.11 (สถาปัตยกรรมของ HyMoNet) แสดงโครงสร้างของ HyMoNet ซึ่งประกอบด้วย 2 ส่วนได้แก่ Service Infrastructure และ User Space

1. Service Infrastructure ถูกติดตั้งโดยผู้ให้บริการและมี UserTracer server และยังมีชุดของ stream data servers ที่ระบุโดย 1,2,3,....

- UserTracer ทำงานเป็นโหนดกลางของระบบและตอบสนองต่อการควบคุมของ HyMoNet node เป็นกระบวนการเข้าร่วมหรือออกจากกลุ่มของแต่ละผู้ใช้และปรับปรุงสถานะของแต่ละผู้ใช้ UserTracer ช่วยให้ HyMoNet node สามารถหาโหนดพ่อแม่ และยังช่วยให้โหนดกู้คืนสำหรับในกรณีที่โหนดพ่อแม่ออกจากระบบ UserTracer สามารถทำงานร่วมกับโปรแกรมอื่นซึ่งช่วยให้โหนด 2 โหนด สามารถติดต่อกันได้ ซึ่งวิธีดังกล่าวจะสร้างระบบ UserTracer แบบระบบกระจายให้เป็นไปตามแผนการกระจายงานและช่วยให้ระบบมีความเสถียรมากขึ้น

- Stream data Server เป็นเครื่องจักรที่มีประสิทธิภาพสูงและถูกติดตั้งใน backbone network ซึ่งมีหน้าที่คล้ายกับ content delivery network โดยที่มีโทโพโลจีของ stream data server เป็นไปตามการติดตั้งของโทโพโลจีจริงของเครือข่ายและความต้องการของผู้ใช้ จากภาพที่ 2.11 (สถาปัตยกรรมของ HyMoNet) โหนด 1 เป็น data source และโหนด 2 เป็น redundancy ซึ่งจะทำงานหลังจากที่โหนด 1 มีปัญหา โหนด 3 ถึง 7 ทำหน้าที่เป็นตัวแทนอย่างรวดเร็ว ซึ่งจะคอยรับข้อมูลจากโหนด 1 และส่งข้อมูลไปให้โหนดลูกของโหนด 1

2. User Space เป็นส่วนของโหนดในระบบทั้งหมดซึ่งเข้ามาเพื่อรับสายข้อมูล การติดต่อทั้งหมดจะเชื่อมต่อกันในลักษณะไดนามิก โครงสร้างของ user space จะเป็นแบบต้นไม้หรือเรียกว่า กลุ่มของมัลติคาสต์ โดยโหนดทั้งหมดจะรับข้อมูลจากโหนดพ่อแม่ ซึ่งโหนดที่อยู่ด้านบนสุดของ user space จะรับข้อมูลโดยตรงจากส่วนที่อยู่ด้านล่างสุดของ service infrastructure

โหนด c, f, g, h และ i เป็นต้นไม้ซึ่งติดต่อกันผ่าน UDP unicast ส่วนโหนด a, d และ e เป็นกลุ่มของมัลติคาสต์ และโหนด a, b และ c รับข้อมูลจาก service infrastructure

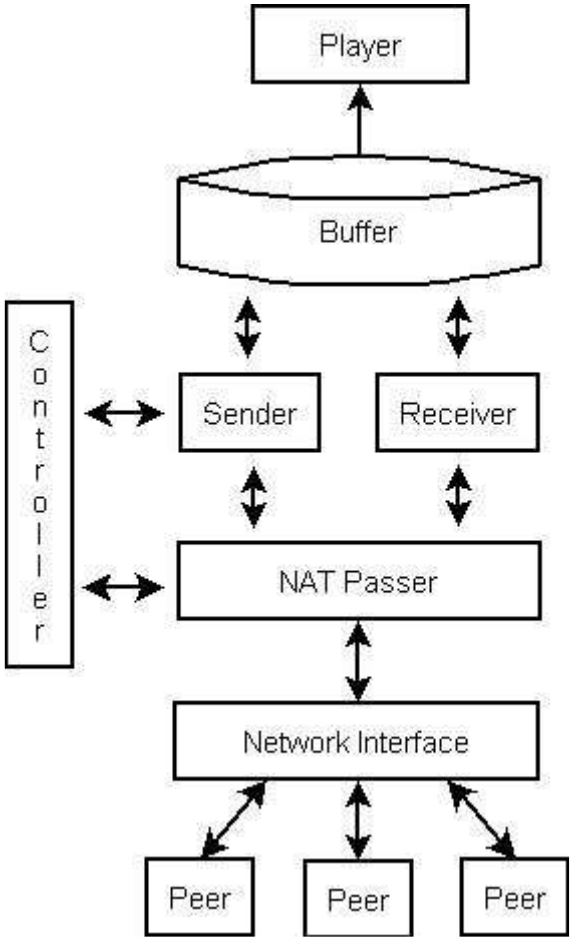
สถาปัตยกรรมของ HyMoNet node แสดงไว้ในภาพที่ 2.12 (สถาปัตยกรรมของ HyMoNet node) โดยแบ่งเป็นเกณฑ์ได้ 3 แบบได้แก่

1. NAT Passer ซึ่งทำงานร่วมกับ UserTracer และช่วยให้โหนดผ่าน NAT ได้โดยเชื่อมต่อโดยตรงกับโหนดอื่น

2. Sender-Buffer-Receiver ทำหน้าที่เกี่ยวกับการส่งและรับข้อมูลของโหนดและรวบรวมสายข้อมูลให้กับ player

3. Controller ทำหน้าที่ดูแลข้อมูลผู้ใช้ของโหนดคอยเชื่อมต่อและปรับปรุงการจับคู่กับโหนดอื่น

ภาพที่ 2.12 สถาปัตยกรรมของ HyMoNet node



เมื่อพิจารณาจากโฮสต์ A และ B ต้องการที่จะติดต่อกันในระบบ peer-to-peer ถ้าไม่มี NAT ระหว่างโฮสต์ทั้ง 2 หรือถ้ามีโฮสต์ใดโฮสต์หนึ่งอยู่ภายหลัง NAT แล้ว การเชื่อมต่อก็จะทำได้ง่ายระหว่างโฮสต์ A และ B แต่ผู้วิจัยต้องการศึกษาว่าทำอย่างไรที่จะให้โฮสต์ A และ B เชื่อมต่อกันได้โดยตรงในขณะที่อยู่ภายหลัง NAT ทั้งโฮสต์ A และ B

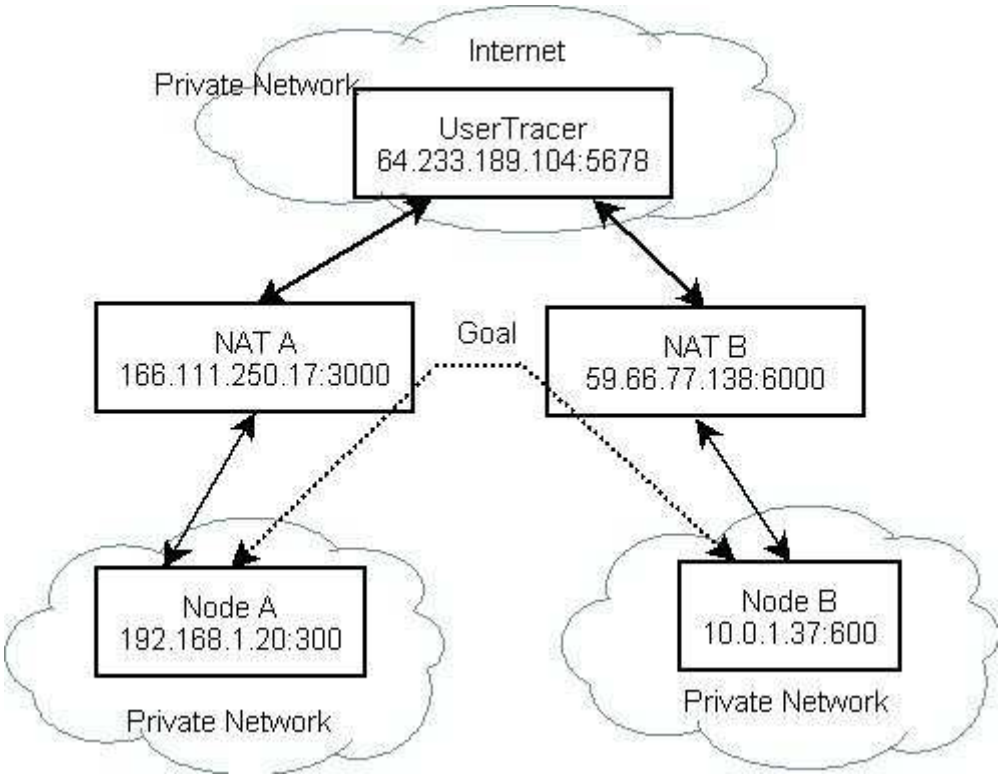
ในโปรโตคอล STUN ได้เสนอวิธี UDP hole punching ซึ่งอนุญาตให้มีการติดต่อกันได้โดยตรงระว่างโฮสต์ที่อยู่ภายหลัง NAT เปรียบเทียบกับ NAT Pass Strategy ที่ได้มีการใช้แล้วกับระบบจริงและยังสามารถแก้ไขบางวิธีของ STUN ให้ตรงตามความต้องการของอุปกรณ์เครือข่ายที่ใช้อยู่จริง

เมื่อโหนด A อยู่ภายใต้ NAT ส่งแพ็กเก็ตไปที่เครือข่ายอินเทอร์เน็ต NAT จะทำการจับคู่แอดเดรสเริ่มต้นและพอร์ตของแพ็กเก็ต กับแอดเดรสและพอร์ตภายนอกทั่วไป เพื่อที่จะผ่าน NAT โดย global address pair ของแต่ละ NAT นั้นจะทำการเรียนรู้โดยการช่วยเหลือของ

โปรแกรมของผู้ที่ร่วมมือกันที่จะพยายามให้เกิดการติดต่อกันได้โดยตรงผ่าน NAT เมื่อทั้งโฮสต์ A และ B ที่อยู่ภายหลัง NAT ได้ส่งแพ็กเก็ต UDP ที่ถูกต้องไปที่ peer อื่น ถือเป็นการสร้างความจำเป็นของ NAT port mapping และการเชื่อมต่อเซสชัน

ใน HyMoNet นั้น UserTracer ทำงานกับโปรแกรมช่วยเพื่อให้เชื่อมต่อกันได้โดยตรง แต่ละ NAT passer ของผู้ใช้มี module ที่ทำงานร่วมกับ UserTracer และ session ของ NAT Passer module

ภาพที่ 2.13 Environment in which the NAT Pass strategy works



เมื่อโหนด A และ B อยู่หลัง NAT gateway NAT A และ NAT B ตามลำดับ เพื่อที่จะติดต่อไปยังแต่ละที่ได้โดยตรง โหนด A และโหนด B จะส่ง แพ็กเก็ต ไปที่ UserTracer เพื่อที่จะให้ UserTracer ทราบว่าไอพีแอดเดรสที่ผ่านการจับคู่จาก NAT ว่าเป็นค่าอะไร และเมื่อ UserTracer แจ้งให้โหนด A ทราบว่า NAT B ส่งค่าอะไรมาและแจ้งให้โหนด B ทราบว่า NAT A ส่งค่าอะไรมา หลังจากนั้นแล้วโฮสต์ A และโฮสต์ B จะเชื่อมต่อกันระหว่าง NAT A และ NAT B โดยไม่ต้องกลับไปใช้ UserTracer อีก

จากภาพที่ 2.13 (Environment in which the NAT Pass strategy works) จะมีการทำงานดังนี้

1. โหนด A และโหนด B ทำการส่งแพ็กเก็ต ไปที่ UserTracer

2. เซิร์ฟเวอร์จะทำการแจ้งโดยส่งไอพี 59.66.77.138:6000 ให้กับโหนด A และส่งไอพี 166.111.250.17:3000 ให้โหนด B

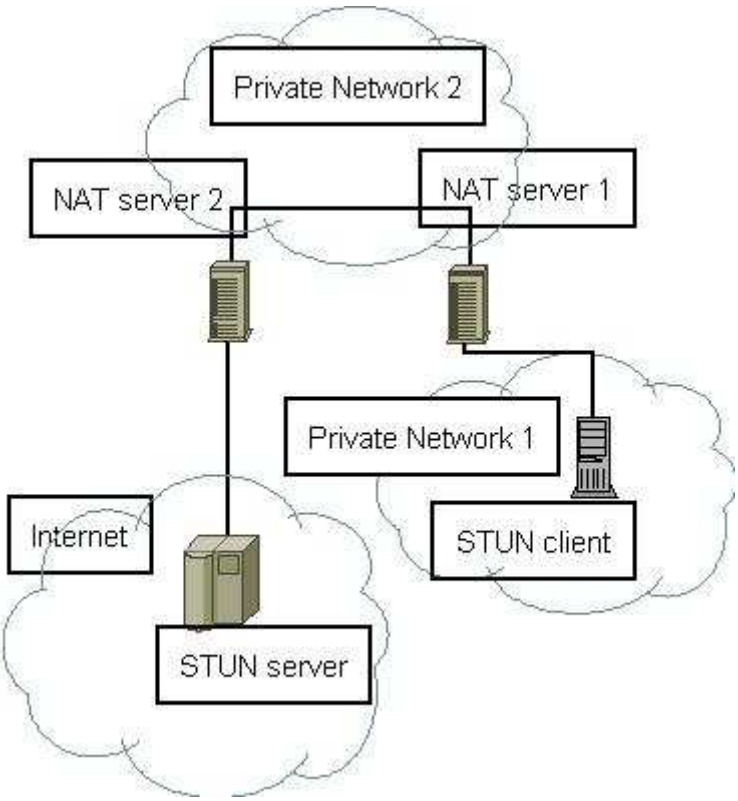
3. โหนด A ส่งแพ็กเก็ต ไปหาโหนด B ที่ไอพี 59.66.77.138:6000 แต่ว่าแพ็กเก็ตจะไม่สามารถส่งไปหาโหนด B เพราะว่า NAT B จะทำการทิ้งแพ็กเก็ต เนื่องจากโหนด B ไม่เคยส่งแพ็กเก็ตออกไปหาไอพี 166.111.250.17:3000 เลย

4. โหนด B ติดต่อไปยังโหนด A ด้วยไอพี 166.111.250.17:3000 การเชื่อมต่อก็จะสมบูรณ์ เนื่องจากโหนด A เคยส่งไปหาไอพี 59.66.77.138:6000 แล้ว

งานวิจัยเรื่อง HyMoNet แสดงให้เห็นถึงการเชื่อมต่อโดยมี UserTracer เป็นตัวกลางในการเชื่อมต่อให้ไคลเอ็นต์ที่อยู่หลัง NAT server ทั้ง 2 ไคลเอ็นต์นั้นให้สามารถรับส่งข้อมูลได้ที่เป็นมัลติคาสต์แพ็กเก็ตได้ซึ่งเป็นลักษณะเดียวกับงานวิจัยของ B. Ford, P. Srisuresh, D. Kegel [4] ถ้าผู้เขียนโปรแกรมดึงเอาความสามารถของระบบ HyMoNet สำหรับโปรแกรม live media streaming ต่างๆ ก็จะสามารถแก้ปัญหาระหว่าง NAT server กับ มัลติคาสต์ได้ แต่ว่าจุดประสงค์ของงานวิจัยนี้คือการแก้ปัญหาระหว่าง NAT server กับมัลติคาสต์ โดยที่โปรแกรมต่างๆ ที่เขียนมาเพื่อใช้งานมัลติคาสต์ยังคงสามารถทำงานได้ภายใต้การทำงานเดิม

2.2.6 Simple Traversal of UDP through NATs (STUN)

ภาพที่ 2.14 STUN Configurations



จาก RFC 3489 ได้เสนอโปรโตคอล STUN [6] ดังภาพที่ 2.14 (STUN Configuration) เป็นลักษณะที่ STUN client จะติดต่อจาก private network 1 ติดต่อกับ private network 2 ผ่าน NAT server 1. private network 2 ติดต่อไปยัง internet ผ่าน NAT server 2. STUN server อยู่บน internet

STUN คือรูปแบบไคลเอ็นต์-เซิร์ฟเวอร์ทั่วไป ที่ไคลเอ็นต์ส่งคำร้องขอไปที่เซิร์ฟเวอร์ และเซิร์ฟเวอร์ส่งผลกลับให้กับไคลเอ็นต์ ซึ่งคำร้องก็จะมี 2 แบบ คือ Binding Request ส่งผ่าน UDP และ Shared Secret Request ส่งบน TLS ผ่าน TCP ทำหน้าที่ดังนี้

- Binding Request (BR) สำหรับเพื่อจุดประสงค์ในการตัดสินการ binding โดย NAT client ส่ง BR ไปยังเซิร์ฟเวอร์ผ่าน UDP ซึ่งเซิร์ฟเวอร์ตรวจ ไอพีแอดเดรสเริ่มต้น/พอร์ต จากคำร้องขอและทำสำเนาเข้าไปในคำตอบกลับที่ส่งกลับไปให้ไคลเอ็นต์ บางพารามิเตอร์ในคำร้องขอ

จะอนุญาตให้ไคลเอ็นต์ ขอคำตอบกลับจากที่ส่งมาจากที่อื่นหรือเซิร์ฟเวอร์ส่งคำตอบกลับจากแอดเดรส/พอร์ต ที่ต่างออกไปคุณสมบัติจัดหา message integrity และ authentication

- Shared Secret Request (SSR) ร้องขอ server ให้ส่ง username/password ชั่วคราวซึ่ง user/password ใช้สำหรับ subsequent Binding Request

วิธีที่ให้ STUN ที่ค้นหา NAT และเรียนรู้ที่จะใช้ binding allocate

STUN client เป็นแบบที่ฝังในแอพพลิเคชันซึ่งจำเป็นต้องได้รับ ไอพีสาธารณะ/พอร์ต ถึงจะสามารถรับข้อมูลได้ ตัวอย่าง STUN client อาจจำเป็นที่จะรับ ไอพีแอดเดรส/พอร์ต ที่ได้จาก real time transport protocol (RTP) เมื่อแอพพลิเคชันเริ่มทำงาน STUN client ในแอพพลิเคชันจะส่ง STUN SSR ไปที่เซิร์ฟเวอร์เพื่อขอรับ username/password และส่ง BR ออกไป STUN server สามารถค้นหาผ่าน DNS SRV record และ STUN client จะได้รับการยอมรับ โดยไคลเอ็นต์จะถูก configure กับโดเมนที่ใช้ค้นหา STUN server โดยทั่วไปแล้วโดเมนจะจัดหาบริการของโปรแกรมที่ใช้แต่ไคลเอ็นต์สามารถตัดสินใจเลือกแอดเดรสหรือชื่อโดเมนของ STUN server ผ่านทางอื่นได้

STUN BR ถูกใช้ค้นหาชนิดของ NAT และค้นหา ไอพีสาธารณะ/พอร์ต ที่จับคู่ผ่าน NAT server โดยใช้ BR ส่งไปที่ STUN server ผ่าน UDP เมื่อเซิร์ฟเวอร์ได้รับ BR แล้วเซิร์ฟเวอร์จะทะลุผ่าน NAT ระหว่าง STUN client และ STUN server ผ่าน UDP ได้ ผลก็คือแอดเดรสเริ่มต้นของการร้องขอที่ได้รับมาจากเซิร์ฟเวอร์จะสร้างคู่แอดเดรส โดย NAT กับ STUN server จะสำเนาไอพีแอดเดรสเริ่มต้น/พอร์ต ใน STUN Binding Response และส่งกลับไปที่ โอพีแอดเดรสเริ่มต้น/พอร์ต ของ STUN request สำหรับชนิดของ NAT ทั้งหมด สามารถรู้ได้โดยการตอบสนองที่จะมาถึง STUN client

เมื่อ STUN client รับ STUN Binding Response แล้ว STUN client จะเปรียบเทียบกับไอพีแอดเดรส/พอร์ต ในแพ็กเก็ตกับไอพีแอดเดรสสาธารณะ/พอร์ตของ STUN client ซึ่งจะ bound เมื่อส่งคำร้องขอไปแล้ว ถ้า ไอพีแอดเดรส/พอร์ต ไม่เหมือนกัน STUN client จะอยู่ภายใต้ NAT server

ในกรณีของ full-cone NAT นั้น ไอพีแอดเดรส/พอร์ต ที่อยู่ในส่วนของ STUN response จะใช้ได้ทุกๆ โฮสต์บนอินเทอร์เน็ตที่ส่งแพ็กเก็ตไปที่แอพพลิเคชันนั้น โดย STUN request และแอพพลิเคชันจำเป็นที่จะต้องฟังอย่างเดียวบนไอพีแอดเดรส/พอร์ตจาก STUN request ที่ส่งไปแล้ว ทุกๆ แพ็กเก็ต ที่ส่งจากโฮสต์บนอินเทอร์เน็ตไปที่ public address/port

โฮสต์จะไม่ทราบว่าอยู่หลัง full-cone NAT อย่างแน่นอน ซึ่งแท้จริงแล้ว STUN client จะไม่ทราบเลยว่าอยู่หลัง NAT การที่ไคลเอ็นต์จะทราบนั้นจะต้องใช้ STUN BR เป็นวิธีการที่ยืดหยุ่นและใช้ได้ทั่วไปไคลเอ็นต์ควรจะส่ง STUN BR ครั้งที่ 2 ครั้งนี้ควรจะมีไอพีแอดเดรสที่ต่างออกไป แต่มาจากไอพีแอดเดรสเริ่มต้น/พอร์ตเดิม แต่ถ้าไอพีแอดเดรสเริ่มต้น/พอร์ตที่ตอบกลับมานั้นแตกต่างจากครั้งแรก ไคลเอ็นต์จะทราบได้ว่าอยู่หลัง symmetric NAT ถ้าไคลเอ็นต์ อยู่หลัง full-cone NAT แล้ว ไคลเอ็นต์สามารถที่จะส่ง STUN BR กับ flag ที่บอกว่า STUN server ส่ง response จาก ไอพีแอดเดรส/พอร์ต ที่ต่างออกไปจาก request ที่รับมา กล่าวคือถ้า client BR ไปที่ ไอพีแอดเดรส/พอร์ต A/B ใช้ ไอพีแอดเดรสเริ่มต้น/พอร์ต X/Y STUN server ควรจะส่ง Binding Response ไปที่ X/Y ที่ใช้ ไอพีแอดเดรสเริ่มต้น/พอร์ต C/D ถ้า ไคลเอ็นต์รับ response แล้วจะทราบได้ว่าอยู่หลัง full-cone NAT

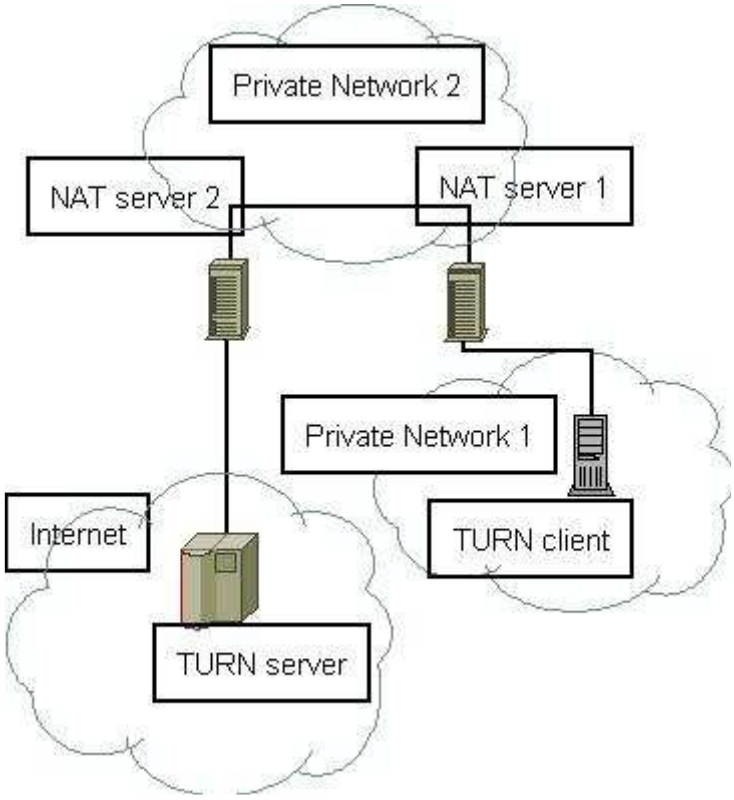
STUN อนุญาตให้ไคลเอ็นต์ร้องขอเซิร์ฟเวอร์ให้ส่ง Binding Response จากไอพีแอดเดรส/พอร์ตเดียวกันของคำร้องที่ได้รับมา แต่พอร์ตที่ได้รับมาไม่เหมือนเดิม เราจะใช้เพื่อตรวจสอบว่าไคลเอ็นต์นั้นอยู่หลัง port restricted cone NAT หรืออยู่หลัง restricted cone NAT

จากการสังเกตจะทราบว่าการปรับแต่งไม่ได้เป็น Permissible configuration อย่างเดียว และ STUN server สามารถตั้งอยู่ได้ทุกที่รวมถึงอยู่รวมกับไคลเอ็นต์อื่น STUN server ต้องการเพียงอย่างเดียวคือไคลเอ็นต์ต้องหาพบและถ้าไคลเอ็นต์พยายามรับ Publicly Routable Address แสดงว่าเซิร์ฟเวอร์อยู่บนอินเทอร์เน็ต

2.2.7 Traversal Using Relay NATs (TURN)

Internet-draft ของ MIDCOM ที่เป็นรูปแบบที่เพิ่มเติมมาจากโปรโตคอล STUN [7] โดยที่รูปแบบการติดตั้ง TURN แสดงได้ดังภาพที่ 2.15 (TURN Configuration) โดยที่ TURN Client อยู่ที่ Private Network 1 และเครือข่ายเชื่อมกับ Private Network 2 ผ่าน NAT server 1 ส่วน Private Network 2 ติดต่อกับอินเทอร์เน็ตผ่าน NAT server 2 บนอินเทอร์เน็ตมี TURN Server

ภาพที่ 2.15 TURN Configurations



TURN เป็นรูปแบบของโปรโตคอลไคลเอ็นต์-เซิร์ฟเวอร์ โดย TURN จะใช้ syntax และ option พื้นฐานทั่วไปเหมือน STUN เพื่อความสะดวกในการร่วมกันพัฒนาของทั้ง STUN และ TURN โดย TURN จะระบุ request message เรียกว่า allocate ซึ่งคือการร้องขอ TURN server ให้ allocate public ip address และพอร์ต โดย TURN server นั้นสามารถทำงานได้ทั้ง TCP และ UDP และการ allocate นั้นอนุญาตให้ไคลเอ็นต์ร้องขอแอดเดรสและพอร์ตผ่านทาง TCP และ UDP ได้ด้วย

เริ่มจาก TURN client ค้นหาแอดเดรสของ TURN server ซึ่งเมื่อ TURN client หา TURN server พบก็จะส่ง TURN allocate request ไปให้ TURN จะจัดหาวิธีการสำหรับ mutual authentication และการตรวจสอบ Integrity สำหรับทั้งการร้องขอและการตอบรับ บนพื้นฐานการ share secret

อย่างไรก็ดี TURN Server จะไม่ relay ทุกแพ็กเก็ตจาก PA ไปยัง BA จนกระทั่งไคลเอ็นต์ส่งแพ็กเก็ต ผ่าน TURN Server นั้นก็คือไคลเอ็นต์ส่งคำสั่ง TURN ซึ่งรวมอยู่ใน แพ็กเก็ตข้อมูลและไอพีแอดเดรสปลายทางและพอร์ต โดย TURN server รับคำสั่งและส่งต่อแพ็กเก็ตไป

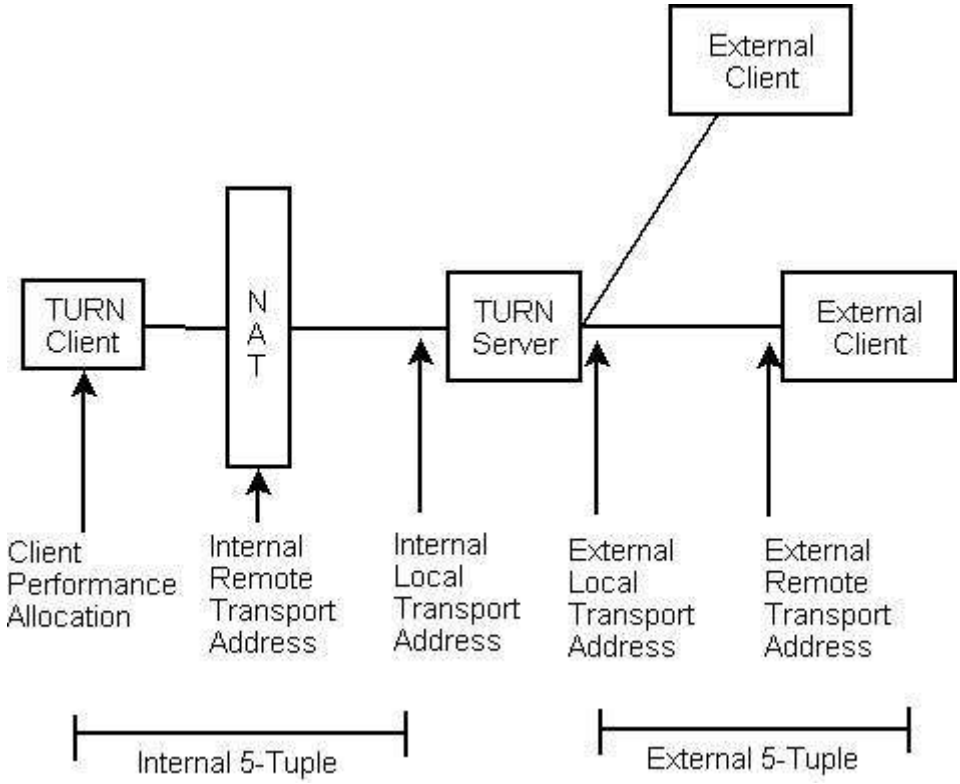
ตามไอพีแอดเดรสและพอร์ต โดยเพิ่ม permission สำหรับไอพีแอดเดรส ดังนั้น inbound packet จากแอดเดรสและพอร์ตนั้นได้รับการยอมรับในลักษณะของ TCP ทำให้ TURN server จะเปิดการเชื่อมต่อแบบ TCP ไปที่เป้าหมายเมื่อพร้อมทำงาน

TURN server รับแพ็กเก็ตโดยการเชื่อมต่อแบบ UDP หรือ TCP โดยส่งคำร้องขอไปที่ไคลเอ็นต์ด้วยการเข้ารหัสข้อมูลของข้อความ ซึ่งเป็นวิธีการที่ไม่มีประสิทธิภาพ ผลก็คือ เมื่อไคลเอ็นต์สิ้นสุดการเชื่อมต่อกับไคลเอ็นต์อื่นที่อยู่ข้างนอกแล้ว ไคลเอ็นต์ต้องสามารถที่จะติดตั้ง active destination request โดยที่ request แจ้งให้เซิร์ฟเวอร์ทราบ ทั้งนี้ปลายของแพ็กเก็ตที่ส่งไปไม่ได้เข้ารหัส ถ้าไคลเอ็นต์ส่งแพ็กเก็ตไป TURN server ซึ่งไม่ได้เป็น TURN packet ซึ่งส่งไปที่ปลายทางเหมือนแพ็กเก็ตจากไคลเอ็นต์ภายนอกที่ TURN server ส่งต่อไปที่ไคลเอ็นต์โดยปราศจากการเข้ารหัสข้อมูล Indication message

Active Destination ครั้งที่ Set แล้วไม่สามารถเปลี่ยนแปลงประสิทธิภาพของการเชื่อมต่อแบบ TCP ได้จากไคลเอ็นต์ไปที่ TURN Server แล้วเปลี่ยนไปเป็นเจ้าของของแอพพลิเคชันเพื่อที่จะ set active destination request

การกระทำทั้งหมดนี้ TURN Server จะปรับปรุงการ binding ระหว่าง internal 5-tuple และ 1 หรือมากกว่า External 5-tuples ซึ่งได้แสดงให้ดูในภาพที่ 2.16 (Internal/External 5-Tuple)

ภาพที่ 2.16 Internal/External 5-Tuple



Internal 5-tuple แทนการเชื่อมต่อระหว่าง TURN Server และ TURN Client ซึ่งเป็นการเชื่อมต่อจริงในวิธีของ TCP และในวิธีของ UDP นั้น คือการรวมกันของไอพีแอดเดรสและพอร์ตจาก TURN Client ที่ส่ง Allocate request มากับไอพีแอดเดรสและพอร์ตที่ Allocate Request นั้นได้ส่งไปแล้ว external local transport address คือไอพีแอดเดรสและพอร์ตออกจาก TURN Client (The allocate transport address)

External 5-tuple คือ การรวมกันของ External local transport address กับไอพีแอดเดรสและพอร์ตของไคลเอ็นต์ภายนอก ซึ่ง TURN Client ได้ทำการติดต่อผ่าน TURN Server เริ่มต้นไม่ได้เป็นในแต่ละ External 5-tuples เมื่อ TURN Client ไม่มีการติดต่อกับโฮสต์อื่นๆ ที่แพ็กเก็ตได้รับหรือส่งจาก allocate transport address ที่ External 5-tuples นั้นถูกสร้างขึ้น

สำหรับ TCP TURN Server ไม่จำเป็นที่ต้องพิจารณาข้อมูลที่ได้รับ แต่เป็นเพียงการส่งต่อข้อมูลทั้งหมดระหว่าง socket pair ซึ่งมีความสัมพันธ์กัน ในวิธีของ UDP นั้น TURN Server จะหา magic cookies ใน 128 byte แรกของแต่ละ UDP Packet ถ้าพบจะระบุในแพ็กเก็ตเรียกว่า TURN control packet เพื่อใช้สำหรับรักษาการเชื่อมต่อและการลดลงของ Binding ในวิธี

ของ TCP หากฝ่ายใดฝ่ายหนึ่งปิดการติดต่อ TURN Server ก็จะปิดการเชื่อมต่อทั้งหมด และสำหรับทั้ง TCP และ UDP เมื่อ TURN Server หมดเวลาการเชื่อมต่อในกรณีที่ไม่ได้รับข้อมูลหลังจากที่หมดช่วงเวลาแล้ว ช่วงของการส่งไปยังไคลเอ็นต์ใน TURN จะตอบสนองที่ allocate request

TURN Server จะรับ UDP Packet และการเชื่อมต่อ TCP จากไคลเอ็นต์ภายนอกเท่านั้น TURN Server อนุญาตให้ไคลเอ็นต์ได้สิทธิ์ในการส่งแพ็กเก็ตนี้ ที่ Specific transport address

2.2.8 Hardware

CISCO ได้พัฒนาวิธีการใหม่ๆ ให้กับฮาร์ดแวร์ โดยเพิ่มความสามารถให้กับฮาร์ดแวร์เพื่อนำมาใช้งานทดแทนเราเตอร์ ซึ่งเป็นอุปกรณ์ที่รู้จักมัลติคาสต์แอดเดรสมีวิธีการหาเส้นทางในหลายลักษณะซึ่งตั้งค่าได้เองตามคู่มือของแต่ละอุปกรณ์ อีกทั้งยังทำงานได้รวดเร็วกว่าซอร์ฟแวร์ เนื่องจากทำงานได้โดยตรงกับฮาร์ดแวร์

สรุป

จากเอกสารและงานวิจัยที่เกี่ยวข้องทำให้ผู้วิจัยมีแนวคิดในการแก้ปัญหาการส่งข้อมูลมัลติคาสต์แอดเดรสผ่าน NAT server โดยเลือกวิธีแก้ไขปัญหาที่ NAT server เนื่องจาก

1. การแก้ปัญหาด้วยวิธีนี้ทำให้โปรแกรมที่ใช้งานอยู่ยังสามารถทำงานได้ โดยไม่ต้องเข้าไปแก้ไขโปรแกรมใหม่

2. การแก้ปัญหาด้วยการแก้ไขโปรแกรมที่ NAT server นั้น ยังไม่มีนักวิจัยเข้ามาศึกษา และแก้ปัญหาเรื่องนี้โดยตรง เนื่องจากเป็นวิธีการที่ค่อนข้างซับซ้อน เพราะจะต้องเข้าใจการเขียนโปรแกรม โดยนำโปรแกรมที่มีไว้ให้แล้วในซอร์สโค้ดของระบบปฏิบัติการลินุกซ์ในส่วนของ module network ซึ่งการเข้าไปทำความเข้าใจเพื่อให้สามารถเขียนโปรแกรมได้นั้นทำได้ช้ามาก และจะเปลี่ยนแปลงเสมอเมื่อมีการออกเวอร์ชันใหม่