สำนักหอสมุดกลาง_พระจอมเกล้าลาดกระบัง

เครือข่ายทดลองโพรโตคอล IPv6

IPv6 Testbed

T046167

นายวัชร ตั้งพานิชยานนท์

นายวีรวัฒน์ ศรีตะลาลัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เลขหมู่.....
เลขทะเบียน 46167
วัน, เดือน, ปี 20 มี.ค. 2546

.b.....
.i.....

เครือข่ายทดลองโพรโตคอล IPv6

IPv6 TESTBED

โดย

นายวัชร ตั้งพานิชยานนท์

นายวีรวัฒน์ ศรีตะลาลัย

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

อาจารย์อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เครือข่ายทดลองโพรโตคอล IPv6

นายวัชร ตั้งพานิชยานนท์
นายวีรวัฒน์ ศรีตะลาลัย
อ.ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อ.อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ในโลกปัจจุบันมีการใช้งานอินเทอร์เน็ตกันอย่างแพร่หลายในทุกๆ ด้านไม่ว่าจะเป็นเพื่อความบันเทิงหรือทางธุรกิจ และมีแนวโน้มที่จะมีผู้ใช้บริการอินเทอร์เน็ตมากขึ้นเรื่อยๆ ในทุกปี ดังนั้นจึงมีความต้องการไอพีแอดเดรสเพิ่มเป็นจำนวนมาก กอรปกับความต้องการทางด้านประสิทธิภาพสูงขึ้นเพื่อสนับสนุนงานหลายๆ ด้าน

โครงงานนี้จัดทำขึ้นเพื่อเตรียมความพร้อมในการปรับเปลี่ยนจากโพรโตคอล IPv4 ไปสู่โพรโตคอล IPv6 โดยการสร้างแบบจำลองเครือข่ายรูปแบบต่างๆ พร้อมกับวิธีการปรับเปลี่ยน ศึกษาปัญหาที่เกิดจากการปรับเปลี่ยน และความเข้ากันระหว่างโพรโตคอลรุ่นเก่ากับรุ่นใหม่

ความจำเป็นของโครงการนี้คือเพื่อเตรียมตัวเพื่อจัดการปัญหาที่อาจจะเกิดขึ้น เนื่องจากการปรับเปลี่ยนเป็นสิ่งที่ยากจะหลีกเลี่ยงในอนาคต

IPv6 Testbed

Mr. Vachara Tangpanichayanont

Mr. Weerawat Seetalalai

Mr. Thana Hongsuwan Advisor

Mr. Akkradech Watcharapupong Advisor

ABSTARCT

Nowadays, Internet is widespreadly used by every field, not only in entertainment but also in business field. Its trend is on the increase on the yearly basis, therefore, there is a big demand of IP addresses, couples with demand for efficiency in order to support various jobs.

This project is made for prepairing the change from IPv4 to IPv6 by creating diverse scenarios together with methods of adaptation; and for solving the problems caused by this change. Also, this project is made for studying the interoperation between the old and new protocol.

In seeking the solutions to the future problems, this project is necessitated because it is very hard to avoid the change from IPv4 to IPv6.

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

สารบัญตาราง

หน้าที่

สารบัญภาพ

หน้าที่

บทที่ 1
บทนำ

IPv4 พัฒนาในปี ค.ศ. 1975 และได้ให้แอดเดรสประมาณ 4.2 พันล้านแอดเดรส แม้ว่าดูเหมือนมีแอดเดรสให้ใช้อย่างเพียงพอ แต่ในความเป็นจริงทุกเครื่องคอมพิวเตอร์ ทุกอินเทอร์เฟซบนเครื่องคอมพิวเตอร์ต้องการแอดเดรสที่เป็นของตนเอง ดังนั้นในปัจจุบันเรากำลังอยู่ในสภาวะขาดแคลนแอดเดรส

ดังนั้น IPv6 (Internet Protocol version 6) จึงพัฒนาเพื่อสนับสนุนจำนวนผู้ใช้ที่เพิ่มมากขึ้นในแต่ละปี และออกแบบมาเพื่อปรับปรุงประสิทธิภาพบางอย่างของ IPv4 ในความเป็นจริงปัญหาของ IPv4 ไม่ใช่เรื่องจำนวนแอดเดรส แต่เป็นเรื่องการแบ่งกลุ่มของแอดเดรสทำให้แอดเดรสจำนวนมากสูญเสียไปอย่างไร้ประโยชน์

IPv6 เสนอขึ้นในปี ค.ศ. 1994 RFC 1752, The Recommendation for The Next Generation Protocol และได้ถูกอนุมัติโดย Internet Engineering Steering Group ให้เป็น Proposed Standard ในปีเดียวกัน ปี ค.ศ. 1998 IPv6 Core Protocol ก็ได้ถูกพัฒนาขึ้นมาโดย IETF

1.1 จุดประสงค์ของโครงงาน

- ศึกษาและทำความเข้าใจเกี่ยวกับโพรโตคอล IPv6
- ศึกษาความเป็นไปได้ของเครือข่ายรูปแบบต่างๆ ในการปรับเปลี่ยนจากโพรโตคอล IPv4 มาเป็นโพรโตคอล IPv6

1.2 ขอบเขตของโครงงาน

- สร้างรูปแบบจำลองเครือข่ายแบบต่างๆ และขั้นตอนการปรับเปลี่ยนจากโพรโตคอล IPv4 ไปเป็นโพรโตคอล IPv6
- วิเคราะห์ถึงปัญหาที่จะเกิดขึ้นจากการปรับเปลี่ยนไปสู่ IPv6

1.3 งานในภาคการศึกษาที่ 1

- ศึกษาทำความเข้าใจในเรื่องต่างๆ ของโพรโตคอล IPv6 จาก RFCs , Internet Drafts , Proposed Standard , Standard Track และจากหนังสือที่เกี่ยวข้อง
- ศึกษาโพรโตคอลที่ทำงานร่วมกับโพรโตคอล IPv6 เช่น Neighbor Discovery Protocol , ICMPv6
- เปรียบเทียบและวิเคราะห์ประสิทธิภาพระหว่างโพรโตคอล IPv4 กับโพรโตคอล IPv6 ในแง่มุมต่างๆ
- ศึกษาวิธีการทำงานร่วมกันระหว่าง IPv4 และ IPv6

1.4 งานในภาคการศึกษาที่ 2

- สร้างรูปแบบเครือข่ายต่างๆ (Scenario) และหาวิธีในการปรับเปลี่ยนรูปแบบเหล่านั้นให้สามารถสนับสนุนการทำงานทั้งโพรโตคอล IPv4 และโพรโตคอล IPv6
- ทดลองการเชื่อมต่อกับเครือข่ายภายใน 6bone เพื่อติดต่อกับเครือข่ายที่ใช้เฉพาะ IPv6

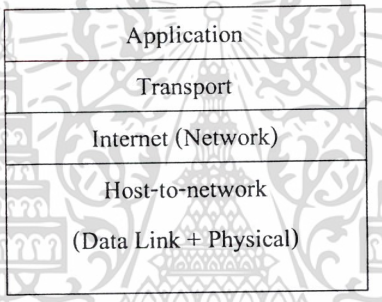
1.5 ปัญหาและอุปสรรค

- ระบบปฏิบัติการบางระบบอาจจะไม่มี Patch ที่ใช้ในการปรับเปลี่ยนไปเป็นโพรโตคอล IPv6
- โพรโตคอลแสตกบางตัวยังไม่สามารถใช้ความสามารถของโพรโตคอล IPv6 ทั้งหมด และบางตัวเป็นรุ่นที่ใช้สำหรับการทดสอบเท่านั้น

บทที่ 2
ทีซีพี/ไอพี และ โพรโตคอลที่เกี่ยวข้อง

ทีซีพี/ไอพี เป็นโพรโตคอลมาตรฐานที่มีการใช้งานกันอย่างแพร่หลายในทุกๆระดับ ทั้งเครือข่ายเฉพาะขนาดเล็กตลอดจนเครือข่ายในบริเวณกว้าง ทีซีพี/ไอพีผ่านการออกแบบให้เป็นอิสระจากคอมพิวเตอร์ทั้งฮาร์ดแวร์และระบบปฏิบัติการ กลไกภายในมีความเชื่อถือได้สูงและทำงานได้แม้ในภาวะที่ผิดปกติ รวมทั้งยังสามารถเลือกเส้นทางได้ในการส่งข้อมูลได้แม้บางเส้นทางเกิดการชำรุด

ระบบการสื่อสารข้อมูลในเครือข่ายคอมพิวเตอร์ประกอบไปด้วยฮาร์ดแวร์และซอฟต์แวร์จึงมีการแบ่งแยกออกเป็นส่วนๆเพื่อช่วยลดความซับซ้อน ชื่อ ทีซีพี/ไอพี มีที่มาจากโพรโตคอลสองโพรโตคอลคือ TCP – Transmission Control Protocol และ IP – Internet Protocol แต่ ทีซีพี/ไอพี ไม่ได้มีส่วนประกอบเพียงแค่สองโพรโตคอลเท่านั้นแต่แบ่งออกเป็นระดับต่างๆดังรูปที่ 2.1



รูปที่ 2.1 แบบอ้างอิง ทีซีพี/ไอพี

Host-to-network ในระดับชั้นนี้จะมีการทำงานใน 2 ระดับคือ Physical และData Link

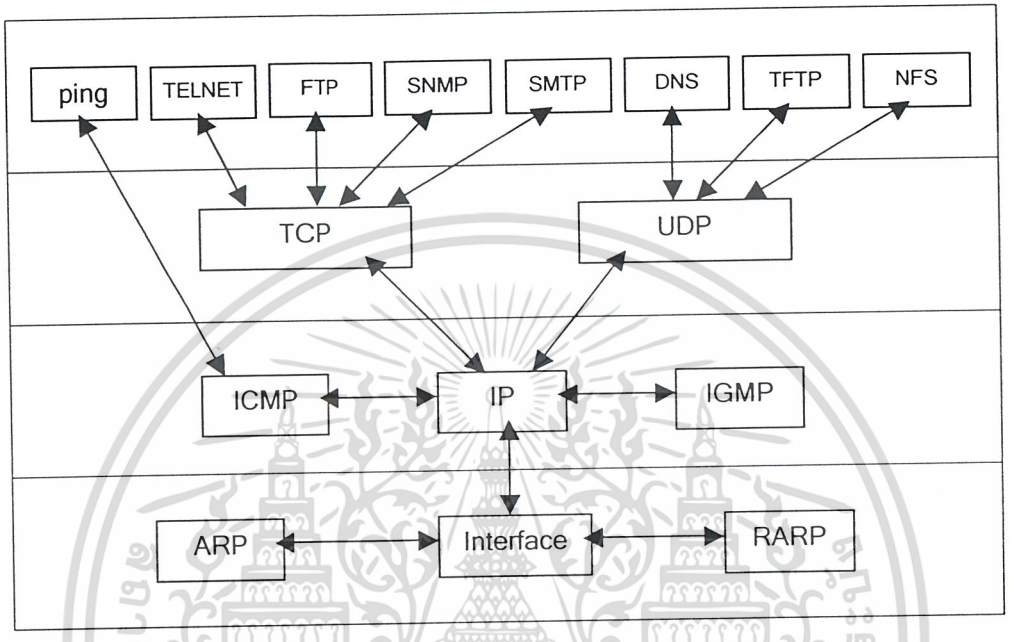
- Physical ชั้นของการกำหนดคุณสมบัติฮาร์ดแวร์ เช่นคุณสมบัติทางกลอย่างหัวต่อ ชนิดสายสื่อสาร คุณสมบัติทางไฟฟ้าอย่างลักษณะสัญญาณอัตราเร็ว เป็นการกำหนดวิธีการส่งข้อมูลในระดับบิต ตัวอย่างเช่น RS232 และ X.21
- Data Link ในระดับชั้นนี้จะทำงานด้านการเชื่อมโยงเข้ากับสายสื่อสาร ตัวอย่างมาตรฐานในระดับชั้นนี้เช่นอีเทอร์เน็ตและโทเคนริง

Internet ทำหน้าที่เลือกเส้นทางเพื่อส่งข้อมูลระหว่างสถานีต้นทางและปลายทาง ตัวอย่างเช่นโพรโตคอล ไอพี

Transport ทำหน้าที่จัดเตรียมการส่งข้อมูลระหว่างสถานีต้นทางและปลายทางโดยการสถาปนาการเชื่อมต่อและรักษาการเชื่อมต่อตลอดจนยกเลิกการเชื่อมต่อเมื่อสิ้นสุดกระบวนการ และอาจยังต้องทำหน้าที่รักษาความถูกต้องของข้อมูล มีโพรโตคอลในระดับชั้นนี้อยู่สองโพรโตคอลคือ ทีซีพี และ ยูดีพี

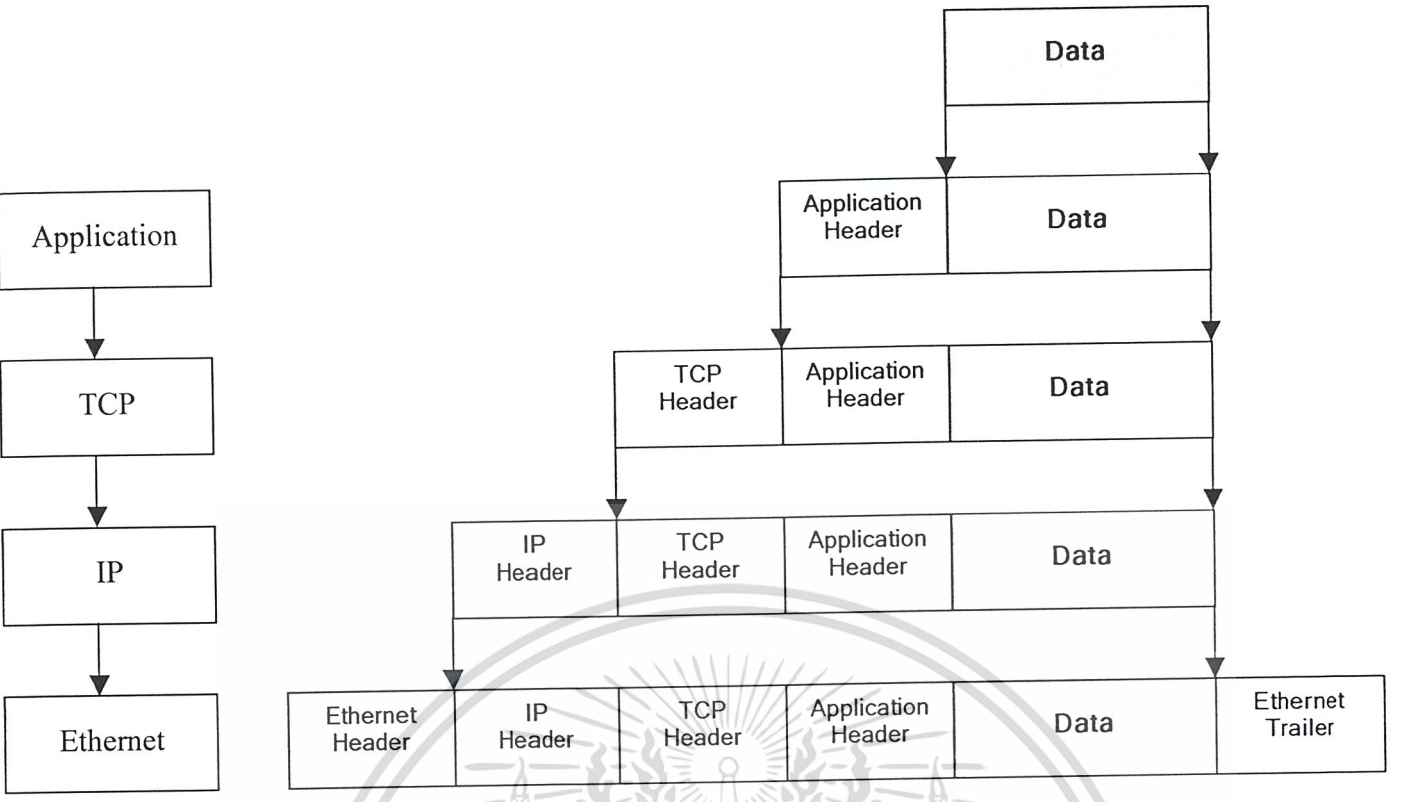
Application เป็นระดับชั้นการทำงานของโปรแกรมประยุกต์ตัวอย่างโพรโตคอลเช่น FTP SMTP TELNET

การทำงานหนึ่งๆนั้นไม่ได้ใช้โปรดตคอลพร้อมกันทั้งหมด แต่ใช้งานสัมพันธ์กันไปในแต่ละระดับชั้น การซ้อนทับของโพรโตคอลจากชั้นบนลงไปชั้นล่างเรียกว่าโพรโตคอลแสตกดังรูปที่ 2.2



รูปที่ 2.2 โพรโตคอลแสตกของ ทีซีพี/ไอพี

ไอพีเป็นโพรโตคอลที่อยู่ในระดับชั้น Network เป็นแกนสำคัญในโพรโตคอลแสตกเนื่องจากทั้งยูดีพีและทีซีพีต่างก็ต้องใช้โพรโตคอลไอพีเพื่อเลือกเส้นทางในการส่งแพ็กเก็ต ข้อมูลจะถูกส่งจากโพรโตคอลชั้นบนสุดลงมาโพรโตคอลชั้นล่างสุดในสถานีส่งและถูกเปลี่ยนเป็นสัญญาณทางไฟฟ้าส่งผ่านสื่อตัวกลางไปยังสถานีปลายทาง ที่สถานีปลายทางก็จะรับสัญญาณแล้วส่งผ่านไปยังโพรโตคอลระดับบน เมื่อข้อมูลผ่านไปในแต่ละระดับชั้น โพรโตคอลในชั้นนั้นจะผนวกข้อมูลที่ใช้ในการทำงานของโพรโตคอลระดับนั้นเรียกว่าส่วนหัวของโพรโตคอล (Protocol Header) เข้าไปกับส่วนข้อมูลและเรียกกระบวนการนี้ว่าเอ็นแคปซูเลชั่น(Encapsulation) ข้อมูลจะถูกส่งไปยังส่วนล่างกว่าโดยโพรโตคอลชั้นล่างจะมองว่าข้อมูลที่ส่งมาทั้งหมดนั้นเป็นข้อมูลทั้งหมดโดยไม่ทราบว่าส่วนใดเป็นข้อมูลจริงหรือส่วนหัวของโพรโตคอล แสดงการทำงานดังรูปที่ 2.3



รูปที่ 2.3 เอ็นแคปซูเลชั่น (Encapsulate) ข้อมูลตามโพรโตคอลแสตก

และเมื่อแพ็กเก็ตข้อมูลมาถึงสถานีปลายทาง ส่วนหัวของโพรโตคอลก็จะถูกถอดออกในแต่ละชั้น และจะเหลือแต่ส่วนข้อมูลเมื่อถึงชั้นบนสุด กระบวนการนี้เรียกว่าการถอดแพ็กเก็ต (Decapsulation)

2.1 เครือข่ายเฉพาะที่

IEEE ได้กำหนดเครือข่ายเฉพาะที่หรือแลน (Lan) ออกมาเป็นมาตรฐานที่นิยมใช้กันแพร่หลายดังต่อไปนี้

1) IEEE 802.3 Ethernet

เป็นเครือข่ายที่มีความเร็วในการส่งข้อมูล 10 เมกะไบต์ต่อวินาที สถานีในเครือข่ายจะรูปแบบการเชื่อมต่อ (topology) เป็นดาว (star) หรือบัส (bus) โดย IEEE ได้กำหนดไว้หลายประเภทตามชนิดสายสัญญาณเช่น

- 10Base5 มีการเชื่อมต่อแบบบัสใช้สายโคแอกเชียล (Coaxial)แบบหนาความยาวของสายในเซ็กเมนต์ไม่เกิน 500 เมตร
- 10Base2 มีการเชื่อมต่อแบบบัสใช้สายโคแอกเชียลแบบบาง ความยาวของสายในเซ็กเมนต์หนึ่งไม่เกิน 185 เมตร
- 10BaseT มีการเชื่อมต่อแบบดาวและใช้ฮับเป็นศูนย์กลาง ใช้สายยูทีพี (UTP-Unsheild Twisted Pair) ความยาวไม่เกิน 100 เมตร

เลข 10 หมายถึงความเร็วในการส่งข้อมูล 10 เมกะไบต์ต่อวินาที คำว่า Base หมายถึงสัญญาณชนิด เบสแบน (Baseband) และเลขที่อยู่ท้ายหมายถึงความยาวของสายแต่หากเป็นตัวอักษรจะหมายถึงชนิดของสายเช่น F – Fiber optics

ส่วนมาตรฐานความเร็ว 100/Sec ที่นิยมใช้กันในปัจจุบันเช่น 100BaseTX และ 100BaseFX และยังมีอีเทอร์เน็ต ความเร็ว 1 กิกะบิตซึ่งก็เริ่มมีความนิยมใช้แพร่หลายกันมากขึ้น เช่น 1000BaseLX และ 1000BaseSX

อีเทอร์เน็ตใช้โพรโตคอล Carrier Sense Multiple Access with Collision Detection – CSMA/CD เป็นตัวกำหนดการทำงานโดยช่วงเวลาหนึ่งจะมีเพียงสถานีเดียวที่ครอบครองสายสัญญาณเพื่อส่งข้อมูล โดยสถานีที่จะทำการส่งข้อมูลก็จะตรวจเช็คว่าขณะนั้นสายสัญญาณมีผู้ใดครอบครองหรือไม่ ถ้าว่างก็จะส่งข้อมูลได้ทันทีแต่ถ้าไม่ว่างก็จะคอยจนกว่าจะว่าง ขณะที่ส่งข้อมูลก็จะตรวจสอบสายสัญญาณไปพร้อมกันด้วยเพื่อตรวจว่าในระยะเวลาที่ใกล้เคียงกันนั้นมีสถานีอื่นตรวจสอบได้ว่าสายว่างและส่งข้อมูลออกมาหรือไม่ หากเกิดเหตุการณ์นี้ก็จะเกิดการชนกัน (Collision) ของข้อมูล สถานีก็จะหยุดส่งและสุ่มหาเวลาที่เริ่มส่งใหม่ เพราะฉะนั้นในเครือข่ายที่มีสถานีงานมากก็จะเกิดการแย่งชิงสายสัญญาณกันและเกิดการชนกันเกือบตลอดเวลา อีเทอร์เน็ตจึงไม่เหมาะกับการใช้งานแบบเวลาจริงเพราะไม่สามารถกำหนดเวลาได้ว่าจะได้ใช้งานเมื่อใด

2) IEEE 802.4 Token Bus

มีการเชื่อมต่อแบบบัสเหมือนกับอีเทอร์เน็ต แต่มีการกำหนด Token เพื่อทำหน้าที่ให้จังหวะกับสถานีต่างเข้าครอบครองสายสัญญาณ โดย Token จะถูกส่งต่อไปยังสถานีต่างๆเป็นวงรอบ สถานีที่ได้รับโทเคน เท่านั้นจึงมีสิทธิใช้สายสัญญาณเพื่อส่งข้อมูล สายที่ใช้มักเป็นสายโคแอกเชียลและมีอัตรราเร็ว 1 5 และ 10 เมกะไบต์ต่อวินาที

3) IEEE 802.5 Token Ring

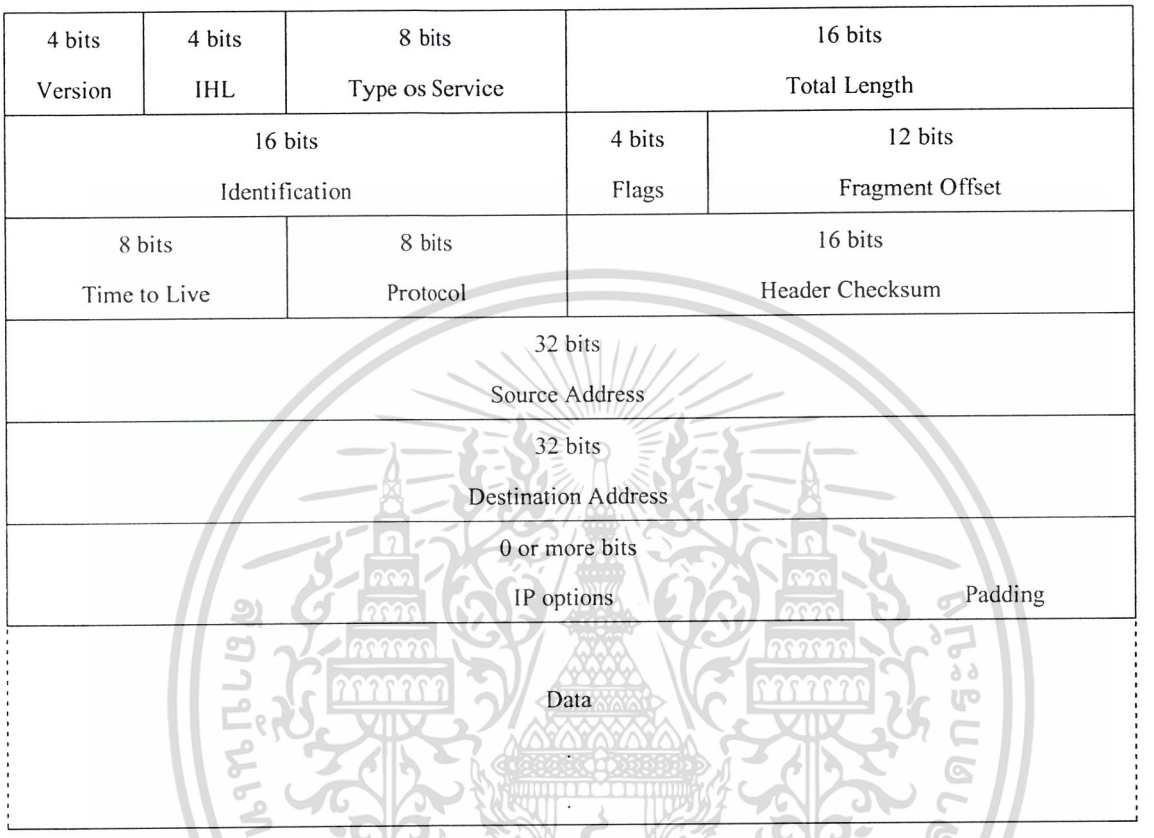
เป็นเครือข่ายที่เชื่อมต่อแบบวงแหวนด้วยสายคู่ตีเกลียวหรือเส้นใยนำแสง มีอัตราการส่งข้อมูลเป็น 4 และ 16 เมกะไบต์ต่อวินาที การทำงานจะมีเฟรมพิเศษเรียกว่าโทเคนอิสระ (Free Token) วิ่งวนรอบเครือข่าย สถานีที่จะส่งข้อมูลต้องรอให้เฟรมนี้วิ่งมาถึงและบรรจุข้อมูลลงไป เมื่อสถานีรับได้รับแล้วก็จะสำเนาข้อมูลไว้แล้วปล่อยกลับมาสถานีส่ง สถานีส่งก็จะปล่อยเฟรมว่างคืนสู่ระบบต่อไป การทำงานแบบนี้สามารถประเมินเวลาในการทำงานได้

2.2 โพรโตคอลไอพีรุ่น 4

ไอพีเป็นโพรโตคอลแกนของทีซีพี/ไอพี ข้อมูลที่ถูกส่งออกไปจะอยู่ในรูปแบบที่เรียกว่า ดาต้าแกรม (Datagram) ประกอบไปด้วยส่วนหัวและส่วนข้อมูล หน้าที่หลักของไอพีคือจัดขนาดของข้อมูลให้พอเหมาะและเลือกเส้นทางที่เหมาะสมเพื่อจัดส่งดาต้าแกรม ไอพีจัดว่ามีรูปแบบการทำงานเป็นแบบไม่สามารถเชื่อถือได้ (Unreliable) คือไม่รับประกันว่าข้อมูลจะถูกส่งไปยังปลายทางสำเร็จ และไม่มีการ

เชื่อมต่อ (Connectionless) คือไม่มีการสถาปนาการเชื่อมต่อกำหนดเส้นทางระหว่างต้นทางและปลายทาง ไม่มีการเก็บสถานะใดๆของดาต้าแกรมที่ส่งออกไป ดาต้าแกรมอาจจะไปถึงที่หมายโดยไม่เรียงลำดับและถูกปล่อยให้เป็นหน้าที่ของโพรโตคอลในระดับชั้นที่สูงกว่า

ไอพี ดาต้าแกรมจะมีรูปแบบเลขฐานสองเป็น Big endian มีรูปแบบดังรูปที่ 2.4



รูปที่ 2.4 รูปแบบดาต้าแกรมของ Internet Protocol (Version 4)

- Version - แสดงรุ่นของโพรโตคอล (ปัจจุบันเป็น 4)
- Internet Header Length (IHL) - บอกความยาวเฉพาะส่วนหัวของดาต้าแกรมโดยนับตั้งแต่ส่วน Version ไปจนถึงบิตสุดท้ายก่อนส่วนข้อมูล หน่วยนับจะมีหน่วยเป็นจำนวนเท่าของ 4 ไบต์
- Type of Service (TOS) - กำหนดรูปแบบการให้บริการ
- Total Length - บอกความยาวทั้งหมดของดาต้าแกรมรวมทั้งส่วนหัวและส่วนข้อมูล มีหน่วยนับเป็นไบต์ ไอพี ดาต้าแกรมจึงมีขนาดสูงสุดได้ $2^{16} - 1$ หรือเท่ากับ 65535 ไบต์
- Identification - เป็นหมายเลขประจำชิ้นดาต้าแกรม ใช้ในกรณีที่ต้องมีการแบ่งย่อยข้อมูล (Fragment) เพื่อบอกว่าข้อมูลนี้เป็นชุดเดียวกัน

- Flags - ใช้ควบคุมรูปแบบสำหรับการแฟรกเมนต์ (Fragment) โดยบิตแรกไม่มีการใช้งาน บิตที่สองบอกว่าอนุญาติให้มีการแฟรกเมนต์หรือไม่ และบิตที่สามหากมีค่าเป็น 1 หมายถึงเป็นคาต้าแกรมอันสุดท้ายที่แฟรกเมนต์ของชุดนั้น
- Fragment Offset - เราเตอร์จะทำการแฟรกเมนต์ข้อมูลออกเป็นส่วนๆ แต่ละส่วนจะมีขนาดเป็นจำนวนเท่าของ 8 ไบต์ ข้อมูลในส่วนนี้จะบอกว่าเป็นตำแหน่งที่เท่าไหร่ของข้อมูลทั้งหมดเช่นหากมีค่าเป็น 80 หมายถึงว่าข้อมูลไบต์แรกของคาต้าแกรมนั้นเป็นข้อมูลไบต์ที่ 80 x 8 = 640 ของข้อมูลทั้งหมดในชุดนั้น
- Time to Live (TTL) - กำหนดจำนวนเราเตอร์ที่คาต้าแกรมจะเดินทางผ่านได้ ทุกๆ ครั้งที่คาต้าแกรมเดินทางผ่านเราเตอร์ ค่าในส่วนนี้จะถูกลดลงครั้งละหนึ่ง และเมื่อลดลงจนเหลือศูนย์แล้ว เราเตอร์จะทิ้งคาต้าแกรมนั้นไปและรายงานความผิดพลาดกลับไปยังสถานีต้นทางด้วยไอซีเอ็มพี (ICMP) มีเพื่อป้องกันการวนรอบโดยไม่สิ้นสุดของคาต้าแกรม
- Protocol - ใช้บอกโพรโตคอลในระดับบนที่เอ็นแคปซูเลชั่นข้อมูลนี้มา เพื่อที่สถานีปลายทางจะได้ส่งข้อมูลให้กับโพรโตคอลระดับบนได้ถูกต้อง
- Header Checksum - ใช้ตรวจสอบความผิดพลาดของส่วนหัวโดยไม่รวมส่วนข้อมูล
- Source Address - กำหนดแอดเดรสของสถานีต้นทาง
- Destination Address - กำหนดแอดเดรสของสถานีปลายทาง
- Option - ในส่วนนี้จะมีขนาดไม่คงที่ ใช้กำหนดการทำงานเพิ่มเติมสำหรับคาต้าแกรมเช่น ด้านความปลอดภัย
- Padding - ใช้ต่อท้ายออปชั่นเพื่อให้ผลรวมทั้งสองส่วนนี้มีขนาดเป็นจำนวนเท่าของ 4 ไบต์
- Data - ข้อมูลที่ถูกส่งมาจากโพรโตคอลระดับบน

2.3 ยูดีพี (UDP - User Datagram Protocol)

ยูดีพี เป็นโพรโตคอลในระดับชั้น Transport หน้าที่ของโพรโตคอลในชั้นนี้คือการจัดเตรียมการส่งถ่ายข้อมูลระหว่างสถานีต้นทางและปลายทางตามชนิดของโปรแกรมประยุกต์ ทีซีพี/ไอพี มีโพรโตคอลที่ทำงานในระดับชั้นนี้อยู่ 2 โพรโตคอลคือยูดีพีและทีซีพี

ยูดีพี มีลักษณะการทำงานเป็นไม่มีการเชื่อมต่อ (Connectionless) คือไม่มีการสถาปนาการเชื่อมต่อระหว่างสถานีต้นทางและปลายทาง ยูดีพีจะทำการส่งข้อมูลโดยไม่มีการตรวจสอบว่าสถานีปลายทางพร้อมที่จะรับข้อมูลหรือไม่ ในกรณีที่เกิดปัญหากับคาต้าแกรมเช่น เกิดการสูญหาย คาต้าแกรมซ้ำหรือไปถึงที่หมายผิดลำดับ ยูดีพีจะไม่มีหน้าที่ที่จะจัดการกับความผิดพลาดนี้ โดยจะถือว่าเป็นหน้าที่ของโปรแกรมประยุกต์ที่จะตรวจสอบและจัดการแก้ไขเอง

ข้อมูลที่ถูกส่งโดย ไอพี จะระบุค่าในส่วน Protocol ว่าต้องส่งข้อมูลนั้นให้กับยูดีพีหรือทีซีพี และจากนั้น ยูดีพีก็จะส่งข้อมูลนั้นต่อให้กับโพรโตคอลในระดับชั้นโปรแกรมประยุกต์(Application layer) โดยจะมีค่าที่บ่งบอกว่าจะส่งให้กับโพรโตคอลใดเรียกว่าพอรต์ (Por)t และการจับคู่ของไอพีแอดเดรสและพอรต์ เรียกว่า ซ็อกเก็ต (Socket) ทีซีพี และ ยูดีพี จะใช้ซ็อกเก็ตเป็นตัวแยกแยะโปรเซสต้นทางและปลายทางที่ติดต่อกัน รูปแบบของ ยูดีพีดาต้าแกรมเป็นดังรูปที่ 2.5

Source Port (16 bit)	Destination Port (16 bit)
Length (16 bit)	Checksum (16 bit)
Data	

รูปที่ 2.5 รูปแบบดาต้าแกรมของ ยูดีพี

- Source Port - ขนาด 16 บิต เป็นหมายเลขพอร์ตของสถานีต้นทาง
- Destination Port - ขนาด 16 บิต เป็นหมายเลขพอร์ตของสถานีปลายทาง
- Length - ขนาด 16 บิต บอกความยาวของดาต้าแกรมรวมทั้งส่วนหัวและส่วนข้อมูล
- Checksum - ขนาด 16 บิต ผลรวมตรวจสอบ โดยคำนวณจากทั้งส่วนหัวและข้อมูล

การคำนวณผลรวมตรวจสอบใน ยูดีพี นั้นจะคำนวณโดยนำข้อมูลเพิ่มเติมบางส่วนมาคำนวณด้วยจะเรียกส่วนนี้ว่าส่วนหัวเทียม (Pseudo-Header) ยูดีพีจะใช้ส่วนนี้คำนวณ Checksum เท่านั้นและจะไม่ส่งออกไป รูปแบบของส่วนหัวเทียมเป็นดังรูปที่ 2.6

1	16	32
Source IP address		
Destination IP address		
0 (Zero)	Protocol	UDP Length

รูปที่ 2.6 Pseudo-Header สำหรับ ยูดีพี

- Source Address - ขนาด 32 บิต เป็นไอพีแอดเดรสต้นทาง
- Destination Address - ขนาด 32 บิต เป็นไอพีแอดเดรสปลายทาง
- Zero - ขนาด 8 บิต มีค่าเป็น 0 ทุกบิต
- Protocol - ขนาด 8 บิต ชนิดของโพรโตคอลสำหรับ ยูดีพี มีค่าเป็น 17
- Length - ขนาด 16 บิต ขนาดส่วนหัวรวมกับข้อมูล (ไม่รวมส่วนหัวเทียม)

บริการ	พอร์ต/โพรโตคอล	การทำงาน
echo	7/UDP	สะท้อนคาต้าแกรมกลับ
daytime	13/UDP	รายงานวันเวลาเครื่อง
TFTP	69/UDP	ถ่ายโอนแฟ้มแบบง่าย
SNMP	161/UDP	บริการบริหารจัดการเครือข่าย
Bootps	67/UDP	พอร์ตเซิร์ฟเวอร์สำหรับบีโมตบูต
Bootpc	68/UDP	พอร์ตไคลเอนต์สำหรับบีโมตบูต

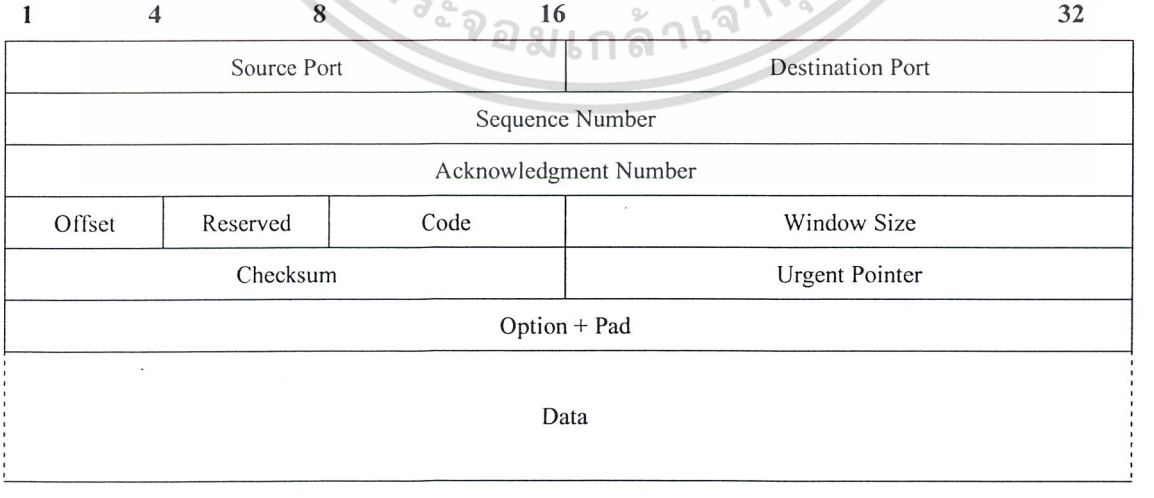
ตารางที่ 2.1 ตัวอย่างพอร์ต ยูดีพี

สำหรับเนื้อหาเพิ่มเติมกี่ยวกับ ยูดีพี สามารถหาเพิ่มเติมได้ใน RFC 768

2.4 ทีซีพี (TCP - Transmission Control Protocol)

ทีซีพีเป็นโพรโตคอลที่มีลักษณะการให้บริการแบบต้องมีการเชื่อมต่อ และรับประกันความเชื่อถือในการส่งข้อมูล โดยการตรวจสอบเซกเมนต์ที่ผิดปกติและมีการจัดส่งเซกเมนต์นั้นซ้ำใหม่ รวมทั้งจัดลำดับให้ถูกต้องก่อนส่งข้อมูลนั้นให้กับโพรโตคอลในระดับชั้นโปรแกรมประยุกต์ ข้อมูลในส่วนนี้เรียกว่า เซกเมนต์ (Segment)

สถานีต้นทางจะต้องมีการเชื่อมต่อกับปลายทางก่อนจะส่งข้อมูล เพื่อที่จะแน่ใจได้ว่าปลายทางพร้อมที่จะรับข้อมูล และจะทำการปิดการเชื่อมต่อเมื่อการส่งข้อมูลเสร็จสิ้น ทีซีพี มีการทำงานแบบไคลเอ็นต์ – เซิร์ฟเวอร์คือจะมีฝ่ายที่ให้บริการและร้องขอบริการ การส่งข้อมูลระหว่างทั้งสองฝ่ายนั้นจะเป็นแบบ Full-duplex รูปแบบส่วนหัวของ ทีซีพี เป็นดังรูปที่ 2.7



รูปที่ 2.7 รูปแบบส่วนหัวของ ทีซีพี

- Source Port - ขนาด 16 บิต เป็นหมายเลขพอร์ตสถานีต้นทาง
- Destination Port - ขนาด 16 บิต เป็นหมายเลขพอร์ตสถานีปลายทาง
- Sequence Number - ขนาด 32 บิต ทุกครั้งที่มีการสถาปนาการเชื่อมต่อ ทีซีพีจะเลือกเลขขึ้นมาค่าหนึ่งเพื่อกำหนดแทนไบต์แรกของข้อมูลที่จะส่ง ค่านี้เรียกว่า Sequence Number ซึ่งไม่จำเป็นจะต้องเป็น 1 และข้อมูลในเซกเมนต์ถัดไปก็จะมีค่า Sequence Number สัมพันธ์กับค่าที่เลือกมาตอนแรก
- Acknowledgement Number - ขนาด 32 บิต เป็นค่าเลขตอบรับข้อมูลที่ได้รับ เช่นเมื่อได้รับเซกเมนต์ที่มีค่า Sequence Number เป็น 200 และมีขนาดข้อมูล 100 ไบต์ ก็จะตอบรับกลับไปโดยมีค่าเลขตอบรับเป็น 200 + 100 + 1 = 301 หมายความว่าได้รับข้อมูลถึงไบต์ที่ 300 แล้วและร้องขอข้อมูลไบต์ที่ 301 ต่อ
- Offset - ขนาด 4 บิต ใช้บอกตำแหน่งของไบต์ที่เป็นส่วนข้อมูล ซึ่งเท่ากับเป็นการบอกขนาดของส่วนหัว มีขนาดเป็นจำนวนเท่าของ 4 ไบต์
- Reserved - ขนาด 4 บิต สำรองไว้ในอนาคต
- Code - แบ่งออกเป็นส่วนย่อย 6 ส่วน ส่วนละ 1 บิตเรียงตามลำดับดังนี้
  - Urgent - หากมีค่าเป็น 1 หมายความว่า Urgent pointer บรรจุตำแหน่งข้อมูลที่ต้องดำเนินการเร่งด่วน
  - Acknowledgment - หากมีค่าเป็น 1 หมายความว่าเป็นเซ็กเมนต์ตอบรับ
  - Push - หากมีค่าเป็น 1 หมายความว่าต้องรีบส่งข้อมูลไปยังโพรโตคอลประยุกต์ทันที
  - Reset - หากมีค่าเป็น 1 หมายความว่าให้ยกเลิกการเชื่อมต่อนี้
  - Synchronoze - หากมีค่าเป็น 1 หมายความว่าขอเริ่มต้นสถาปนาการเชื่อมต่อ
  - Finish - หากมีค่าเป็น 1 หมายความว่าขอจบการเชื่อมต่อ
- Window Size - ขนาด 16 บิต ใช้แจ้งขนาดบัฟเฟอร์ที่มีอยู่ มีหน่วยเป็นไบต์ ฝ่ายส่งต้องไม่ส่งข้อมูลเกินขนาดนี้
- Checksum - ขนาด 16 บิต ผลรวมตรวจสอบทั้งส่วนหัวและข้อมูล มีการใช้ส่วนหัวเทียมเหมือนกับ ยูดีพี ด้วย
- Urgent Pointer - ขนาด 16 บิต ชี้ตำแหน่งไบต์ข้อมูลที่ต้องดำเนินการเร่งด่วน
- Options - ขนาดไม่คงที่ ใช้กำหนดงานเพิ่มเติมอาจไม่มีก็ได้
- Pad - ขนาด 0 ถึง 24 บิต ใช้เพิ่มเข้าไปต่อท้าย Option เพื่อให้ขนาดรวมเป็นจำนวนเท่าของ 4 ไบต์

ทีซีพี ให้บริการด้านความเชื่อถือการลำเลียงเซกเมนต์ข้อมูลที่สำคัญคือ

1. การตรวจจับและแก้ไขข้อผิดพลาด

เซกเมนต์ที่ได้รับอาจจะเกิดความผิดพลาดได้เนื่องจากสายสื่อสารหรือโพรโตคอลระดับล่าง ทีซีพี จัดการปัญหานี้โดยใช้ผลรวมตรวจสอบ หากค่าผลรวมตรวจสอบไม่ถูกต้องจะดำเนินการส่งเซกเมนต์นั้นใหม่ หรือในกรณีที่เซกเมนต์อาจจะเดินทางไปไม่ถึงปลายทาง ทีซีพีจะใช้การตอบรับกลับเพื่อบอกว่าได้รับเซกเมนต์นั้นแล้ว โดยจะมีการตั้งเวลาเพื่อรอการตอบรับจากฝ่ายรับ หากไม่มีการตอบรับก็จะส่งซ้ำ

2. การจัดลำดับเซกเมนต์
   ในกรณีที่เซกเมนต์มาถึงโดยไม่เรียงลำดับ ทีซีพีทางฝั่งฝ่ายรับจะมีหน้าที่เรียงลำดับเซกเมนต์ให้ถูกต้อง รวมทั้งยังกำจัดเซกเมนต์ที่ซ้ำซ้อนด้วย
3. การควบคุมกระแสข้อมูล
   ทีซีพี ใช้เทคนิคที่เรียกว่าหน้าต่างเลื่อน (Sliding Window) ในการควบคุมให้ฝ่ายส่งจัดเตรียมข้อมูลในปริมาณที่ฝ่ายรับจะสามารถรับได้จริงโดยไม่เกินขนาดของบัฟเฟอร์ที่เหลืออยู่

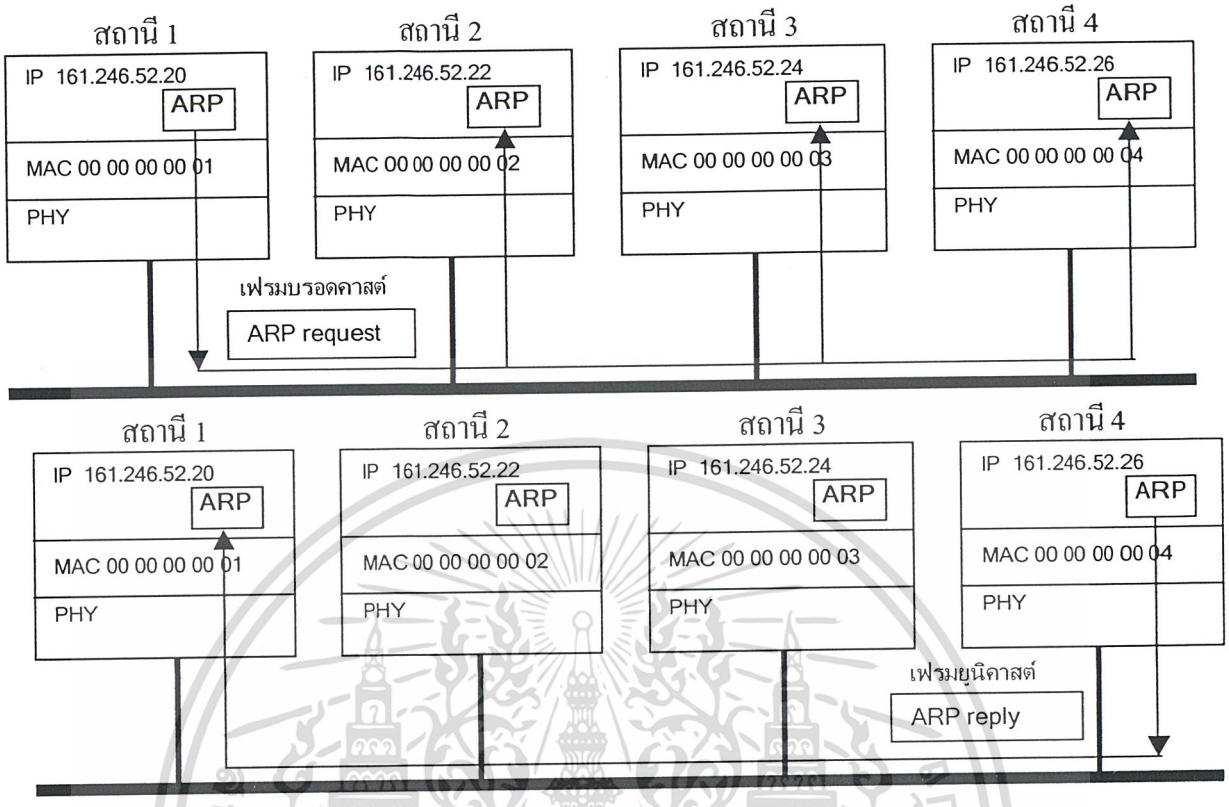
สำหรับเนื้อหาเพิ่มเติมเกี่ยวกับ ทีซีพี สามารถหาอ่านเพิ่มเติมได้จาก RFC 793

2.5 โพรโตคอลอื่นๆที่เกี่ยวข้อง

1) เออาร์พี (ARP) และอาร์เออาร์พี (RARP)

การสื่อสารในระดับล่างระหว่างสถานีภายในเครือข่ายจะใช้ฮาร์ดแวร์แอดเดรสที่กำหนดไว้ในการ์ดอินเทอร์เฟซเพื่อสื่อสารกัน ดังนั้นสถานีต้นทางจะต้องทราบว่าที่สถานีปลายทางนั้นมีฮาร์ดแวร์แอดเดรสอะไร จึงได้มีการกำหนดโพรโตคอลที่ใช้ทำหน้าที่นี้ไว้ใน RFC 826 เรียกว่า Address Resolution Protocol – ARP โดยการสร้างตารางที่เก็บคู่ของไอพีแอดเดรสและฮาร์ดแวร์แอดเดรส เรียกว่าตารางเออาร์พี หรือเออาร์พีแคช (ARP Cache) โพรโตคอลเออาร์พี ได้ถูกออกแบบมาให้เพื่อใช้ในเครือข่ายที่มีลักษณะการทำงานแบบบรอดคาสต์ จึงเหมาะกับเครือข่าย IEEE 802.3/4/5 และยังมีอีกโพรโตคอลหนึ่งซึ่งทำงานตรงข้ามกับเออาร์พี เรียกว่า Reverse Address Resolution Protocol – RARP โดยจะทำหน้าที่ค้นหาไอพีแอดเดรสจากฮาร์ดแวร์แอดเดรส

หลักการทำงานเริ่มจากสถานีต้นทางที่ต้องการส่งข้อมูลไปยังสถานีปลายทางจะค้นหาฮาร์ดแวร์แอดเดรสจากเออาร์พีแคชหากไม่พบก็จะสร้างดาต้าแกรมร้องขอ (ARP request) โดยใส่ไอพีแอดเดรสที่ต้องการทราบฮาร์ดแวร์แอดเดรสลงไปและบรอดคาสต์ออกไป สถานีทุกสถานีในเครือข่ายนั้นจะได้รับดาต้าแกรมนี้และจะตรวจสอบว่ามีไอพีแอดเดรส ตรงกับของตนเองหรือไม่หากตรงก็จะส่งดาต้าแกรมตอบรับ (ARP reply) กลับไปพร้อมกับฮาร์ดแวร์แอดเดรสของตัวเอง สถานีต้นทางก็จะนำค่า ไอพีแอดเดรสนี้ไปใช้ และ เก็บค่าลงตาราง สถานีปลายทางก็จะเก็บค่า ไอพีแอดเดรส ของสถานีต้นทางลงตารางเช่นเดียวกัน การทำงานแสดงดังรูปที่ 2.8



รูปที่ 2.8 การทำงานของ ARP

คู่ของ ไอพีแอดเดรส และฮาร์ดแวร์แอดเดรสจะคงอยู่ใน cache อยู่ช่วงระยะเวลาหนึ่ง แตกต่างกันตามระบบปฏิบัติ มีประโยชน์ในกรณีที่มีการเปลี่ยนแปลงสภาพแวดล้อมของระบบเช่นอาจจะมีการเปลี่ยนแปลงการ์ดเครือข่ายใหม่โดยที่ ไอพีแอดเดรส ยังคงเป็นค่าเดิม สำหรับรายละเอียดเพิ่มเติมเกี่ยวกับ ARP และ RARP หาอ่านได้ที่ RFC 826 และ RFC 903

2) ไอซีเอ็มพี (ICMP)

ไอพี เป็นโพรโตคอลที่ไม่มีกลไกในการรับประกันในการนำส่งข้อมูล หากเกิดความผิดพลาดขึ้น ไอพี จะใช้ Internet Control Message Protocol (ICMP) รายงานความผิดพลาดนั้นกลับไปยังสถานีต้นทาง ไอซีเอ็มพี ไม่ได้ช่วยรับประกันความถูกต้องของในการส่ง ไอพี ดาต้าแกรม แต่ช่วยให้สามารถตรวจสอบความผิดพลาดได้ง่ายขึ้น

ไอซีเอ็มพีจะถูกส่งไปยังสถานีต้นทางโดย ไอพี ดาต้าแกรม หาก ไอพี ดาต้าแกรมที่บรรจุไอซีเอ็มพีนี้เกิดการสูญหายในระหว่างทาง จะไม่มีการแจ้งกลับไปด้วยไอซีเอ็มพีอีกเพื่อป้องกันการแจ้งความผิดพลาดแบบบวนซ้ำ และห้ามใช้ไอซีเอ็มพีกับความผิดพลาดต่อไปนี้

- ไม่ใช้กับดาต้าแกรมที่เป็นการบรอดคาสต์ มัลติคาสต์
- ไม่ใช้กับส่วนอื่นของการแฟรกเมนต์ยกเว้นส่วนแรก

- ไม่ใช้กับการทำลูปแบ็ค

ไอซีเอ็มพีจะมีข้อมูลในส่วน type เพื่อแจ้งลักษณะความผิดพลาดและจะมีค่า code เพื่อแยกย่อยลงไปในรายละเอียด ตัวอย่างประเภทของความผิดพลาดแสดงดังตารางที่ 2.2

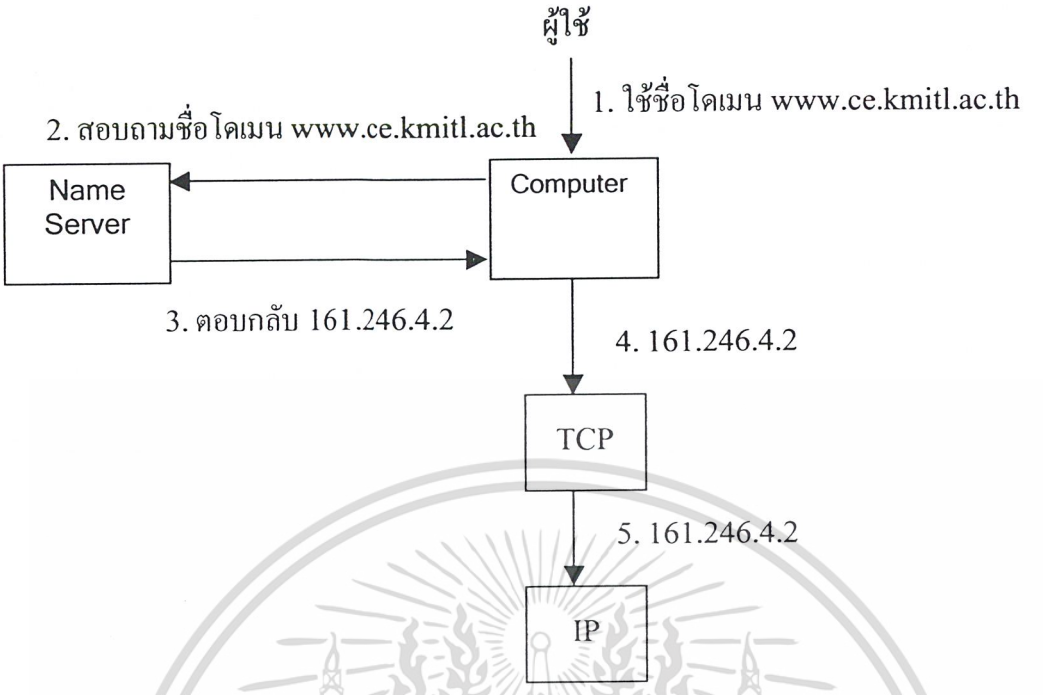
type	ความหมาย
0	Echo reply
3	Destination unreachable
4	Source quench
5	Redirect
8	Echo request
9	Router advertisement
10	Router solicitation

ตารางที่ 2.2 ตัวอย่างแสดงประเภทความผิดพลาดของ ICMP

สำหรับรายละเอียดเพิ่มเติมเกี่ยวกับ ICMP หาอ่านได้ที่ RFC 792

3) ดีเอ็นเอส (DNS)

Domain Name System (DNS) หรือ ระบบชื่อโดเมนเป็นระบบที่ใช้ในการตั้งชื่อทรัพยากรเครือข่าย ใช้งานกันทั่วไปในการตั้งชื่อให้กับโฮสต์แทน ไอพีแอดเดรสเพื่อสะดวกในการใช้งาน ดีเอ็นเอสจะเก็บข้อมูลในลักษณะกระจาย โดยไม่มีหน่วยงานใดเก็บข้อมูลครอบคลุมทั้งอินเทอร์เน็ต ในแต่ละเครือข่ายจะมีดีเอ็นเอสเซิร์ฟเวอร์เก็บรักษาข้อมูลเรียกว่าเนมเซอร์เวอร์ (Name Server) และสามารถจัดการกับข้อมูลนั้นได้อย่างอิสระ ขั้นตอนการทำงานนั้นแสดงดังรูปที่ 2.9



รูปที่ 2.9 ขั้นตอนการใช้งาน DNS

กระบวนการทำงานของดีเอ็นเอสประกอบด้วย Resolver ซึ่งเป็นโปรแกรมในเครื่องของผู้ขอบริการ ทำหน้าที่รับคำสั่งจากโปรแกรมประยุกต์และนำผลลัพธ์ที่ได้กลับคืนโปรแกรมประยุกต์ โดยรีโซวเวอร์ (Resolver) มีอยู่ 2 ประเภทคือ สตับรีโซวเวอร์ (Stub Resolver) จะรวมอยู่ในโปรแกรมประยุกต์เลย และฟูลรีโซวเวอร์ (Full Resolver) จะแยกตัวออกมาเป็นโปรเซสต่างหาก ทางด้านเนมเซอร์เวอร์นั้นจะต้องให้บริการสอบถามทั่วทั้งระบบอินเทอร์เน็ต หากรีโซวเวอร์ทำการร้องขอข้อมูลที่ไม่มีอยู่ในระบบตัวเนมเซอร์เวอร์จะต้องดำเนินการค้นหามาให้ได้โดยหากเป็นการขอบริการแบบเวียนซ้ำ (Recursive) แล้ว ตัวเนมเซอร์เวอร์จะทำหน้าที่สอบถามกับเนมเซอร์เวอร์อื่นเพื่อให้ได้มาซึ่งคำตอบให้กับผู้ขอบริการ แต่หากเป็นการร้องขอแบบทำซ้ำ (Iterative) ตัวเนมเซอร์เวอร์จะเพียงส่งแอดเดรสของเนมเซอร์เวอรอื่นกลับมาเพื่อให้ รีโซวเวอร์ของผู้ขอบริการไปสอบถามกับ เนมเซอร์เวอร์ตัวอื่นเอง

นอกจากการบริการสอบถามไอพีแอดเดรสจากโดเมนเนมแล้ว ดีเอ็นเอสยังรับผิดชอบการสอบถามโดเมนเนมจากไอพีแอดเดรสด้วย ดีเอ็นเอสใช้บริการทั้งยูดีพีและทีซีพี สำหรับเนื้อหาเพิ่มเติมเกี่ยวกับดีเอ็นเอส สามารถหาเพิ่มเติมได้จาก RFC 974 , RFC 1034 และ RFC 1035

บทที่ 3
ทำไมจึงต้องเป็น IPv6

การเกิดขึ้นของ IPv6 นั้นไม่ได้หมายความว่าเทคโนโลยีทั้งหมดนั้นจะเหนือกว่า IPv4 เราอาจจะคาดหวังให้มีการเปลี่ยนแปลงไปสู่ IPv6 ให้เร็วที่สุด แต่ก็ยังคงเป็นหัวข้อโต้แย้งกันอยู่ เพราะว่ามีความจำเป็นที่จะต้องทำให้สามารถใช้งานร่วมกับ IPv6 ด้วย และที่สำคัญจะต้องไม่เป็นการบังคับให้ระบบอินเทอร์เน็ตทั้งหมดต้องทำการปรับเปลี่ยน

3.1 เหตุผลทางธุรกิจ

การเจริญเติบโตของอินเตอร์เน็ตเป็นไปอย่างสูง ธุรกิจหลายๆประเภทได้นำเสนอสินค้าและบริการของตนผ่านอินเตอร์เน็ต IPv6 จึงมีความสำคัญสำหรับธุรกิจต่างๆ เครือข่ายขนาดใหญ่ และระบบอินเตอร์เน็ต เนื่องจากในองค์กรธุรกิจมีความจำเป็นที่จะต้องใช้การติดต่อแบบอินเตอร์แอคทีฟ และยังต้องการใช้งานโปรแกรมประยุกต์ผ่านเครือข่ายที่ต้องการแบนวิดท์ (bandwidth) สูง IPv6 จึงถือเป็นทางรอดของเครือข่ายองค์กรธุรกิจ

เหตุผลหลักในการปรับเปลี่ยนสู่ IPv6

- ✓ การขาดแคลนไอพีแอดเดรส
- ✓ การเติบโตของตารางการเลือกเส้นทางของแบ็คโบนเราเตอร์ (Backbone Router) ที่ใหญ่เกินไป
- ✓ ความต้องการใช้งานความสามารถเพิ่มเติมเช่น ด้านความปลอดภัย และประสิทธิภาพของการเลือกเส้นทาง

IPv6 ได้ถูกประกาศออกมาเป็นมาตรฐานและถูกเผยแพร่ออกมา ทำให้องค์กรต่างๆทั่วโลกสามารถนำไปทดสอบการทำงานได้ ในสภาพการทำงานที่ต่างๆกันไปทั้งการทำงานบน Unix และ Linux IPv6ได้ถูกออกแบบมาให้มีประสิทธิภาพสูง มีความยืดหยุ่น รวมทั้งได้แก้ไขข้อผิดพลาดที่เกิดขึ้นจาก IPv4 IPv6 ได้นำเสนอคุณลักษณะที่ดีหลายประการตัวอย่างเช่น การมีที่อยู่อย่างมากมายมหาศาล มีการจัดรูปแบบของแพ็กเก็ตใหม่ มีการใช้การติดตั้งแบบอัตโนมัติ (Autoconfiguration) ช่วยลดภาระการทำงานของคนได้ในการกำหนดที่อยู่และพารามิเตอร์ต่างๆเช่น MTU

3.1.1 การจัดการแอดเดรสและตารางหาเส้นทาง (Addressing and Routing)

IPv6 ช่วยแก้ปัญหาที่เกิดขึ้นระหว่างเครือข่ายหลักในระดับโลก IPv6 จะให้มีผู้ออกแบบระบบแบ็คโบน เพื่อสร้างลำดับชั้นการสืบทอดการควบคุมเส้นทาง โดยผู้ออกแบบนั้นก็คือองค์กรที่มีเครือข่ายขนาดและผู้บริการเผยแพร่อินเตอร์เน็ต(Internet Service Provider – ISP) มาทำงานร่วมกันโดยระบบที่ออกแบบจะขึ้นอยู่กับการควบคุมดูแลการสืบทอดที่อยู่โดยจะออกแบบตามระบบโทรศัพท์ให้มีค่า Prefix

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

นำหน้า หากไม่มีการ สืบทอดการควบคุมดูแลเป็นลำดับแล้ว เราเตอร์ในแบ็คโบน (backbone) จะต้องเก็บข้อมูลเส้นทางการเข้าถึงในทุกๆเครือข่ายบนโลกซึ่งมีจำนวนมหาศาลมาก ซึ่งเทคนิคนี้เริ่มมีใช้ใน IPv4 เรียกว่า Classes InterDomain Routing – CIDR

ที่ใช้บิตมาร์ค (bit mask)ในการแบ่งส่วนต่างของแอดเดรสออกเป็นเลขเครือข่าย ซับเน็ต และ โฮสต์ CIDR อนุญาติให้แบ็คโบนเราเตอร์เก็บเพียงแค่ค่าเดียวก็สามารถเข้าถึงเครือข่ายในระดับต่ำกว่าได้หลายเครือข่าย แต่ CIDR ไม่รับรองประสิทธิภาพและความยืดหยุ่น เพื่อที่จะหลีกเลี่ยงการกำหนดค่าแยกไปในแต่ละเส้นทาง เครือข่ายในระดับที่ต่ำกว่าจะต้องทำการรวม Prefix เพื่อที่จะนำเสนอในตารางเลือกเส้นทางของเราเตอร์ในระดับที่สูงกว่า

3.1.2 กำจัดกรณีพิเศษ (Eliminating special case)

ปัญหาส่วนใหญ่ที่เกิดขึ้นในแบ็คโบนทุกวันนี้คือ ความแตกต่างของการใช้งานในระดับองค์กรธุรกิจกับการใช้งานส่วนบุคคล เมื่อองค์กรไม่สามารถทำการรวมเส้นทางได้อย่างมีประสิทธิภาพ ทำให้เกิดภาระแก่แบ็คโบนเราเตอร์ ถ้าองค์กรไม่สามารถนำเสนอที่อยู่เดียวบนอินเทอร์เน็ตได้ ก็อาจจะถูกบังคับให้ตัดออกจากระบบและไม่สามารถเข้าสู่อินเทอร์เน็ตได้

สำหรับผู้ใช้งานส่วนตัวที่ไม่มีที่อยู่ที่เป็นของตัวเอง จะต้องการเกตเวย์และอาจจะเป็น Network Address Translators – NATs เพื่อที่จะจัดการกับการเชื่อมต่อสู่โลกภายนอก NATs ช่วยให้องค์กรสามารถมีโครงสร้าง address เป็นของตนเองได้โดยไม่ต้องกังวลว่าจะเกิดการขัดแย้งกับภายนอก NATs เป็นอุปกรณ์ที่ติดตั้งอยู่ระหว่างองค์กรกับอินเทอร์เน็ต เปลี่ยน address ส่วนตัวที่ใช้ภายในองค์กรให้เป็นแอดเดรสที่ใช้กันทั่วโลกซึ่งมีจำนวนน้อยกว่าดังรูปที่ 3.1

รูปที่ 3.1 Network Address Translator – NAT

NAT อาจจะมีความเหมาะสมกับบางองค์กร โดยทั่วไปแล้วจะใช้กับองค์กรที่ไม่สามารถเชื่อมต่อกับอินเทอร์เน็ตได้ทั้งหมด NAT ใช้การแทนที่ในส่วน address ในทุกๆแพ็กเก็ตที่เข้าและออกจากองค์กร ซึ่งจะทำให้เกิดปัญหาคอขวด NAT อาจจะเหมาะที่จะใช้ในเครือข่ายขนาดเล็ก แต่เมื่อเครือข่ายขององค์กรมีขนาดใหญ่ขึ้น ประสิทธิภาพของ NAT จะต้องเพิ่มขึ้นในรูปแบบขนาน (parallel) ปัญหาคอขวดจะเพิ่มมากขึ้นเนื่องจากความยากในการรวมและจังหวะในการทำงานร่วมกันของอุปกรณ์ NAT หลายๆตัวในองค์กรเดียว

46167

NAT จะเกิดปัญหาเมื่อโปรแกรมประยุกต์รวมไอพีแอดเดรสไว้ในส่วนข้อมูลที่ทำการขนส่งในระดับที่สูงกว่าระดับชั้นเครือข่ายซึ่งมีหลายโปรแกรมเช่น โปรแกรมที่เกี่ยวกับการ FTP , Mobile IP และ Windows Internet Name Service (WINS) ถ้าให้ NAT ทำการจัดการแพ็กเก็ตในระดับโปรแกรมประยุกต์แล้ว ก็จะมีบางส่วนที่จัดการไม่ได้ทำให้โปรแกรมเกิดการล้มเหลว NAT ยังสามารถทำการหยุดการทำงานของดีเอ็นเอสเพราะว่า มันทำงานในระดับที่สูงกว่าระดับชั้นเครือข่าย และยังป้องกันการใช้ระบบรักษาความปลอดภัยในระดับไอพี ระหว่างจุดสิ้นสุดของการส่งข้อมูล ในปัจจุบัน NAT ถูกใช้ในสถานการณ์บางอย่างในองค์กรขนาดเล็ก แต่ในระยะยาวแล้วจะส่งผลเสียต่อระบบอินเทอร์เน็ต

3.1.3 ลดภาระในการดูแลระบบ

งานหลักๆของการดูแลระบบในปัจจุบันคือคือการกำหนดค่าพารามิเตอร์ต่างๆที่เกี่ยวกับเครือข่ายให้กับคอมพิวเตอร์และโหนดต่างๆ ตัวอย่างเช่น ไอพีแอดเดรส ค่าดีเอ็นเอสเซอร์เวอร์ และ ค่า configuration ต่างๆที่ต่างกันในแต่ละโหนด โดยทั่วๆไปแล้วกระบวนการนี้จะทำด้วยคน แต่ก็มีความพยายามที่จะใช้ Dynamic Host Configuration Protocol – DHCP แต่ก็ทำให้การดูแลระบบยุ่งยากขึ้น

ข้อเสียใน IPv4 คือในกรณีที่องค์กรต้องการที่จะเปลี่ยนแปลงค่าของอุปกรณ์ ตัวอย่างเช่น กำหนดไอพีแอดเดรสใหม่เมื่อองค์กรเปลี่ยน ISP ซึ่งอาจจะต้องเปลี่ยนแปลงค่าไอพีแอดเดรสใหม่ทั้งหมดเพื่อให้ตรงกับ Prefix ของ ISP ใหม่หรืออุปกรณ์ Network Address Translation – NATs การเปลี่ยนแปลง Prefix มีผลต่อกระบวนการหาเส้นทาง (routing) ของเครือข่ายในองค์กรและโหนดที่เชื่อมต่อ มีอยู่ 2 ทางที่อาจจะเกิดความผิดพลาดได้

1. ส่วน Prefix อาจจะยาวเกินกว่าที่ระบบจะสามารถเพิ่มจำนวนของโหนดที่สามารถเชื่อมต่อได้
2. เส้นทางที่เครือข่ายเชื่อมต่อเข้าด้วยกันหรือเชื่อมต่อกับภายนอกอาจจะเปลี่ยนแปลง

เหตุการณ์อันใดอันหนึ่งอาจจะเกิดขึ้นได้ทำให้จำเป็นจะต้องมีการกำหนดค่าต่างๆใหม่ จะเป็นการดีมากๆที่จะสามารถทำการปรับเปลี่ยนได้โดยไม่ต้องใช้เวลามาก

ปัญหาเหล่านี้ยังสามารถเกิดขึ้นได้กับเครือข่ายขององค์กรขนาดเล็กได้ด้วย ในกรณีที่ผู้ใช้ตามบ้านจะต้องการติดต่อไปยังที่ทำงานโดยผ่านอินเทอร์เน็ต เครือข่ายขนาดเล็กอาจจะหายไปจากตารางเลือกเส้นทางของ แบ็คโบน เราเตอร์ ถ้าไม่ทำการปรับเปลี่ยนลำดับการควบคุมแอดเดรส ขณะที่เครือข่ายขนาดใหญ่ที่ไม่ทำการปรับเปลี่ยนอาจจะส่งผลเสียต่อระบบอินเทอร์เน็ตได้ ใน IPv4 ผู้ใช้ตามบ้านไม่สามารถที่จะกำหนดแอดเดรสได้อย่างอิสระ หรือในกรณีอื่นๆ ในที่ทำงานขนาดเล็กอาจจะต้องใช้แอดเดรสร่วมกัน ทำให้เกิดปัญหาเป็นอย่างมากเมื่อมีอุปกรณ์ต่างๆเชื่อมต่อกับอินเทอร์เน็ตเพิ่มขึ้น

3.1.4 ด้านความปลอดภัย

การเข้ารหัส การพิสูจน์ตัวจริง และการรักษาความถูกต้องของข้อมูล เป็นสิ่งจำเป็นสำหรับองค์กรธุรกิจ ด้วยเหตุนี้ IPv6 จึงถูกกำหนดให้มีส่วนเพิ่มเติมส่วนหัวที่ช่วยในเรื่องความปลอดภัย ส่วนเพิ่มเติมส่วนหัว Authentication ช่วยให้ผู้รับรู้ได้ว่าแพ็กเก็ตที่ได้รับนั้นมาจากผู้ส่งตรงตามที่ระบุไว้ใน Source

address จริงรึเปล่า ช่วยป้องกันการทำการหลอกลวงที่อยู่ (IP spoofing) ซึ่งเป็นเทคนิคที่ทำให้เกิดการขโมยข้อมูลที่มีค่าทางธุรกิจ ใน IPv4 เป็นไปไม่ได้ที่เซอร์เวอร์จะตรวจสอบว่าแพ็กเก็ตถูกส่งมาจากโหนดใด บางองค์กรพยายามป้องกันโดยการติดตั้ง Firewall แต่ก็ทำให้เกิดปัญหาใหม่ๆ รวมไปถึงเกิดคอขวดขึ้นลดประสิทธิภาพ

IPv6 ใช้กระบวนการที่เป็นมาตรฐานในการพิสูจน์ตัวจริงของแพ็กเก็ตที่ได้รับในระดับชั้นเครือข่าย (Network layer) เพื่อให้เครือข่ายที่แตกต่างกันสามารถทำการพิสูจน์ตัวจริงที่ทำงานร่วมกันได้ การทำ IPv6 implementation จึงกำหนดให้ต้องสนับสนุน MD5 และ SHA-1 แต่ในข้อกำหนด (Specification) นั้นไม่ขึ้นอยู่กับอัลกอริทึมจึงสามารถใช้อย่างอื่นได้

นอกจากการทำ spoofing แล้วปัญหาหลักทางด้านความปลอดภัยยังเกิดจากการทำ Packet sniffers ซึ่งทำให้เกิดการขโมยหมายเลขบัญชีเงินฝากและบัตรเครดิต พาสเวิร์ด ข้อมูลความลับทางการค้าหรือข้อมูลที่เป็นส่วนตัวอื่นๆ ใน IPv6 จะทำการแก้ปัญหานี้โดยใช้ส่วนเพิ่มเติมส่วนหัวที่ช่วยในการเข้ารหัสสำหรับ end-to-end ในระดับชั้นเครือข่าย ในส่วนเพิ่มเติมส่วนหัวนี้จะระบุคีย์ที่จะใช้และข้อมูลอื่นๆ ที่เข้าใจตรงกัน สำหรับในส่วนเพิ่มเติม IPv4 นั้นก็ได้มีการกำหนดไว้แต่ไม่ได้มีการใช้กันมากนัก

3.1.5 อุปกรณ์เคลื่อนที่

ในระยะเวลา 20 ปีต่อจากนี้การใช้งานอินเทอร์เน็ตจะมีความแตกต่างจากปัจจุบันอย่างคาดไม่ถึง การใช้งานในช่วงที่ผ่านมาอาจจะมีการใช้ เครื่องคอมพิวเตอร์เคลื่อนที่ , PDAs , โมเด็มผ่านระบบดาวเทียม และ World Wide Web ในอนาคตเราอาจจะได้ใช้คอมพิวเตอร์ที่ติดตั้งภายในรถยนต์ที่สามารบอกได้ว่าขณะนี้เราอยู่ที่ไหนและจะใช้เส้นทางใดเพื่อหลีกเลี่ยงปัญหาการตจราจรติดขัด อาจจะสามารถแก้ไขตารางนัดหมายเพื่อบอกให้คนในครอบครัวรู้ว่าตอนนี้เราอยู่ที่ไหน และอาจจะทำการควบคุมระบบต่างๆภายในบ้านอย่างระบบทำความร้อนหรือแสงสว่าง และในรถคันนั้นก็อาจจะสามารถตรวจสอบระบบความเสียหายภายในที่สามารถสั่งการไปยังอู่ซ่อมรถให้เตรียมชิ้นส่วนไว้ เมื่อเราไปถึงอู่ก็สามารทำการเปลี่ยนได้ทันที ไม่ว่าเราจะอยู่ที่ไหนหรือทำอะไร อุปกรณ์ที่เราใช้ทุกๆอย่างจะสามารถเชื่อมต่อกับอินเทอร์เน็ตได้ และ IPv6 จะสามารถตอบสนองต่อสิ่งนี้ได้

การจัดการกับเครื่องคอมพิวเตอร์เคลื่อนที่โดย IPv4 นั้นทำได้ยากด้วยเหตุผลหลายประการ

- เครื่องคอมพิวเตอร์เคลื่อนที่จำเป็นจะต้องใช้แอดเดรสที่ไม่เจาะจงในแต่ละครั้งที่เชื่อมต่อกับอินเทอร์เน็ต ซึ่งเป็นการยากสำหรับแอดเดรสใน IPv4
- เครื่องคอมพิวเตอร์เคลื่อนที่ต้องการการพิสูจน์ตัวจริงที่เชื่อถือได้ซึ่งความสามารถนี้ IPv4 ไม่สนับสนุน
- ใน IPv4 จะเป็นการยากที่จะตัดสินว่าโหนดเคลื่อนที่เหล่านั้นเชื่อมต่อในอยู่ในเครือข่ายเดียวกันหรือไม่
- ไม่น่าเป็นไปได้สำหรับ IPv4 ที่จะบอกให้คู่สื่อสารทราบในกรณีที่มีการย้ายสถานที่

ปัญหาเหล่านี้สามารถแก้ไขได้โดยใช้ IPv6 การใช้งานอุปกรณ์เคลื่อนที่นี้ได้ถูกคำนึงถึง

เป็นอย่างมากตั้งแต่เริ่มการออกแบบ IPv6 การออกแบบคุณลักษณะต่างๆที่เพิ่มขึ้นมาเช่น Autoconfiguration , Routing header , Encapsulation , Securitiy และ Anycast address ล้วนเพื่อช่วยสนับสนุนการใช้งานอุปกรณ์เคลื่อนที่ทั้งนั้น แท้ที่จริงแล้วบางส่วนของดาวเทียมในยุโรปได้ทำการปรับเปลี่ยนให้เป็น IPv6 แล้ว การใช้งานอุปกรณเคลื่อนที่สามารถใช้ Flow label เพื่อจัดการให้มีคุณภาพที่ดีขึ้นได้

IPv6 สามารถมีจำนวนที่อยู่ได้มากมายมหาศาล มีการแบ่งระดับชั้น (hierachy) ของแอดเดรสออกเป็นหลายระดับโดยใช้รูปแบบ CIDR กระบวนการเลือกเส้นโดยใช้ตารางการหาเส้นทาง นั้นสามารถกำหนดค่าได้ง่ายและทำให้ขนาดของตารางการหาเส้นทาง ไม่ใหญ่โตมาก ข้อดีมากๆของ IPv6 ก็คือช่วยตอบสนองความต้องการมีที่อยู่เป็นของตัวเองได้ โดย ISP สามารถมีที่อยู่เพื่อแจกจ่ายให้กับธุรกิจขนาดเล็กและผู้ใช้งานส่วนตัวทั่วๆไปได้อย่างเพียงพอ

3.1.6 การกำหนดแอดเดรสแบบอัตโนมัติ (Address Autoconfiguration)

เมื่อเริ่มต้นการทำงาน โหนด IPv6 จะสร้าง local IPv6 address ขึ้นมาเป็นของตัวเองโดยใช้การติดตั้งอัตโนมัติแบบไร้สถานะ (Steteless Autoconfiguration) โดยไม่ร้องขอความช่วยเหลือจากเซอร์เวอร์ กระบวนการนี้ช่วยให้โหนดสามารถติดต่อกับเราเตอร์ที่มันเชื่อมต่ออยู่ได้ โดยโหนดจะทำการรวม 48 หรือ 64 บิต MAC address (ซึ่งถูกกำหนดโดยบริษัทผู้ผลิต) เข้ากับ Prefix เครือข่ายซึ่งรับรู้ได้จาก เราเตอร์ ที่เชื่อมต่ออยู่ ทำให้ผู้ใช้เครื่อง (End-user) ไม่จำเป็นต้องร้องขอความช่วยเหลือจากผู้ที่มีความรู้ กระบวนการนี้ทำให้ค่าใช้จ่ายในการดูแลระบบต่ำลง ในเครือข่าย IPv4 มีการใช้ Dynamic Host Configuration Protocol – DHCP เพื่อลดการทำงานของคนลง DHCP เป็นเครื่องมือช่วยในการติดตั้งแบบเต็มสถานะ (Stateful) เพราะว่ามันจะทำการเก็บรักษาตารางของแอดเดรสที่จะทำการกำหนดให้กับโหนดที่เข้ามาเชื่อมต่อใหม่ มีการสร้าง DHCP ในเวอร์ชั่นใหม่เพื่อนำมาใช้งานกับ IPv6 เรียกว่า DHCPv6 (draft-ietf-dhc-dhcpv6ext-11.txt) ซึ่งมีการใช้ DHCP server เพื่อทำการ มัลติแคสต์ ไปสู่ในแต่ละโหนด

การทำการติดตั้งแบบอัตโนมัตินี้มีประโยชน์สำหรับองค์กรธุรกิจเป็นอย่างมาก เมื่อองค์กรทำการเปลี่ยน ISP ทำให้ต้องทำการเปลี่ยนPrefix การติดตั้งอัตโนมัติจะทำให้โฮสต์สามารถทำการปรับเปลี่ยนค่าต่างๆได้โดยอัตโนมัติรวมทั้งยังช่วยให้คอมพิวเตอร์เคลื่อนที่สามารถได้แอดเดรสได้อย่างอัตโนมัติไม่ว่าจะทำการเชื่อมต่อเครือข่ายที่ใดก็ตาม

3.1.7 รูปแบบส่วนหัวของ IPv6

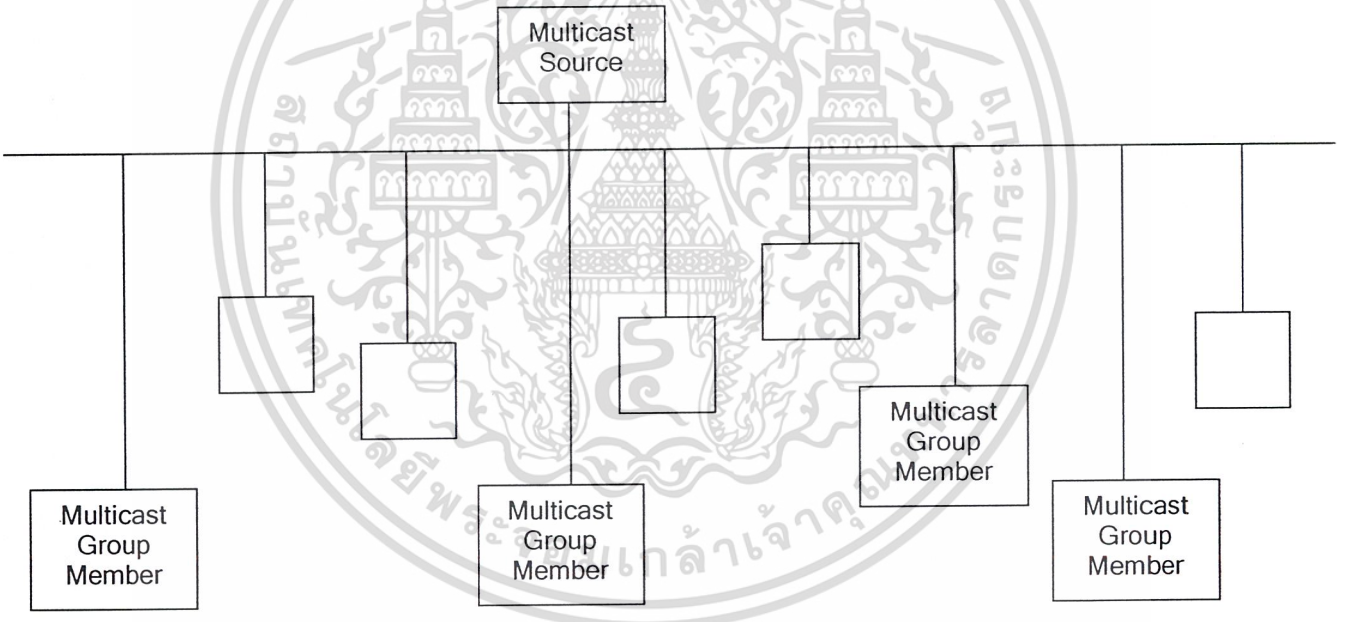
IPv6 ได้ทำการตัดทิ้งข้อมูลในส่วนหัวบางส่วนจาก IPv4 การออกแบบโครงสร้างอย่างง่ายทำให้ช่วยลดแบนวิดท์ได้ ถึงแม้ว่าขนาดของแอดเดรสจะมีขนาดถึง 128 บิต (16 ไบต์) จะมีขนาดเป็น 4 เท่าของ IPv4 แต่ขนาดของส่วนหัวทั้งหมดมีขนาดเพิ่มเป็น 2 เท่าของ IPv4 เท่านั้น

IPv6 ได้ออกแบบให้ส่วนที่ใช้การทำงานพิเศษต่างๆถูกประมวลผลเป็นลักษณะสายข้อมูล (Streamline) โดยจะกำหนดให้การทำงานพิเศษให้เป็น “ส่วนเพิ่มเติมส่วนหัว” และจัดวางให้อยู่ระหว่าง

ส่วนหัวของ IPv6 กับส่วนหัวของข้อมูลในชั้น Transport-layer ส่วนเพิ่มเติมส่วนหัวโดยส่วนมากแล้วจะไม่ถูกประมวลผลโดยโหนดที่เป็นทางผ่านจากต้นทางไปสู่ปลายทาง ซึ่งตรงข้ามกับ IPv4 เป็นผลให้การสูญเสียประสิทธิภาพจากการประมวลผลส่วนเพิ่มเติมในทุกๆโหนดตัวกลางใน IPv4 ถูกกำจัดไป ส่วนหัวของ IPv6 จึงมีขนาดที่แปรผันได้และมีความยืดหยุ่นเนื่องจากสามารถกำหนดส่วนเพิ่มเติมใหม่ๆขึ้นมาได้

3.1.8 มัลติแคสต์

เครือข่ายในยุคใหม่มีความต้องการส่งสายข้อมูลต่อเนื่อง (Stream) ในรูปแบบภาพต่อเนื่องและเสียง ทั้งเพื่อประโยชน์ทางด้านข่าวหรือการเงิน ข้อมูลที่มีลักษณะต่อเนื่องเป็นช่วงจะถูกส่งไปยังกลุ่มที่มีความสัมพันธ์กันแต่อาจจะมีที่อยู่กระจัดกระจายห่างกัน โดยปกติแล้ว เซอร์เวอร์ จะส่งข้อมูลที่ต่อเนื่องเป็นช่วงเวลาไปยังสมาชิกที่ลงชื่อไว้ ความสามารถของ มัลติแคสต์ สามารถส่งข้อมูลไปยังสมาชิกแต่ละคนได้โดยใช้เส้นทางอย่างมีประสิทธิภาพ มีการทำซ้ำข้อมูลเท่าที่จำเป็นดังรูปที่ 3.2



รูปที่ 3.2 การทำงานของ Multicast

จากรูปที่ 3.2 แพ็กเก็ตเพียงอันเดียวจากผู้ส่งจะได้รับโดยสมาชิกกลุ่ม เมื่อมีเครือข่ายแยกย่อยออกไป แพ็กเก็ตจะถูกเผยแพร่ออกไปยังสมาชิกกลุ่มเป็นรูปแบบต้นไม้ (tree)

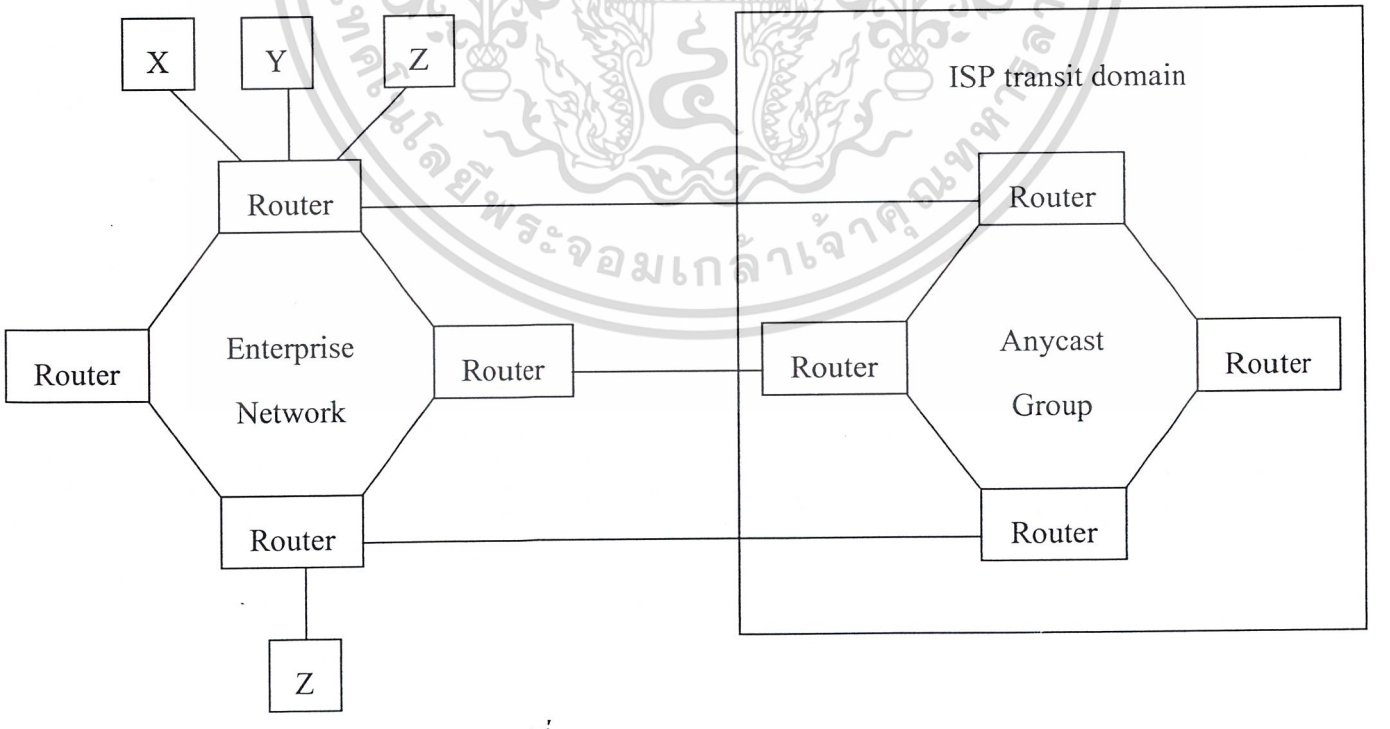
เราเตอร์จะใช้มัลติแคสต์โพรโตคอลเช่น DVMRP (Distance Vector Multicast Routing Protocol) และ PIM (Protocol Independent Multicast) หรือ MOSPF (Multicast Open Shortest Path First) เพื่อสร้างแบบจำลองต้นไม้การแพร่กระจายแพ็กเก็ตเชื่อมต่อไปยังสมาชิกทุกคน สมาชิกใหม่จะ

กลายมาเป็นสมาชิกได้โดยการส่งข้อความ Join ไปยังเราเตอร์ที่อยู่ใกล้ๆ แบบจำลองต้นไม้ก็จะสร้างเส้นทางใหม่ขึ้น ด้วยวิธีนี้ เซอร์เวอร์ ก็จะสามารถส่งแพ็กเก็ตเพียงแพ็กเก็ตเดียวไปยังสมาชิกทุกคนได้โดยมีการทำซ้ำแพ็กเก็ตน้อยที่สุด ทำให้ช่วยลดการทำงานของทั้ง เซอร์เวอร์ และทรัพยากรของเครือข่ายมัลติแคสต์ถูกสร้างขึ้นมาใช้ใน IPv4 แต่IPv6 ทำการขยายความสามารถของมัลติแคสต์โดยการการกำหนดที่อยู่สำหรับมัลติแคสต์ให้มีจำนวนเพิ่มมากขึ้น โหนดและเราเตอร์ที่เป็น IPv6 จำเป็นจะต้องทำการสนับสนุนมัลติแคสต์ ในความเป็นจริงแล้ว IPv6 ไม่ได้มีบรอดแคสต์แอ็ดเดรสโดยตรง เสมือนเป็นมัลติแคสต์ ที่ผันแปร scope เท่านั้น

3.1.9 เอนี่แคสต์

เป็นการทำงานที่ถูกเพิ่มขึ้นมาใน IPv6 ตามแนวคิดแล้วเอนี่แคสต์นั้นอยู่ระหว่างยูนิแคสต์และมัลติแคสต์ กลุ่มของโหนดใดๆอาจจะถูกจัดให้เป็นกลุ่มเอนี่แคสต์ แพ็กเก็ตใดๆที่มีที่อยู่ปลายทางเป็นเอนี่แคสต์ จะถูกส่งไปยังโหนดใดโหนดหนึ่งที่อยู่ในกลุ่มนั้น โดยปกติแล้วจะเป็นโหนดที่มีส่วนเชื่อมต่อที่ใกล้ที่สุด โหนดที่อยู่ในกลุ่มของเอนี่แคสต์จะถูกปรับเป็นพิเศษให้จำค่าเอนี่แคสต์แอ็ดเดรสซึ่งก็อยู่ในส่วนของยูนิแคสต์แอ็ดเดรส

เอนี่แคสต์เป็นบริการใหม่และยังไม่มีโปรแกรมประยุกต์ที่พัฒนามาสนับสนุน โดยการใช้เอนี่แคสต์ องค์กรสามารถส่งไปยังเราเตอร์ตัวหนึ่งที่เป็น แบ็คโบน ของ ISP ดังรูปที่ 3 ถ้า เราเตอร์ ทั้งหมดมีเอนี่แคสต์แอ็ดเดรสเดียวกัน การส่งข้อมูลจากองค์กรก็มีเส้นทางที่เชื่อมต่อไปยังอินเทอร์เน็ตหลายเส้นทาง และหากเราเตอร์ตัวใดหยุดทำงาน เราเตอร์ตัวอื่นก็สามารถรับทำหน้าที่ข้อมูลต่อได้โดยอัตโนมัติ



รูปที่ 3.3 Anycast addressing

จากรูปที่ 3.3 สมมติว่าโหนด X , Y และ Z ใน Enterprise network ส่งข้อมูลไปยังเอนี่แคสต์แอ็ดเดรสที่บริการโดย แบ็คโบน เราเตอร์ในกลุ่มเอนี่แคสต์ของ ISP Transit Domain เราเตอร์รอบนอกของ enterprise จะส่งข้อมูลไปยัง ISP และเราเตอร์ตัวใดตัวหนึ่งใน ISP ก็จะได้รับข้อมูลนั้น

Anycast ถูกสร้างมาเพื่อจุดประสงค์ในการให้บริการที่ต่อเนื่องในกรณีที่เกิดเหตุขัดข้องกับโหนดที่ให้บริการ ตัวอย่างเช่นการกำหนดให้ ดีเอ็นเอส เซอร์เวอร์ หลายๆตัวในองค์กรมีเอนี่แคสต์แอ็ดเดรสเดียวกัน

3.1.10 คุณภาพการให้บริการ

ใน IPv4 มีส่วนสำหรับการเรียกใช้บริการที่แตกต่างกัน ส่วน IPv6 ก็มีส่วน"ระดับชนิดการจราจร (traffic)" ที่ทำหน้าที่คล้ายกัน เพื่อสนับสนุนการร้องขอการจัดการที่แตกต่างกัน ทั้ง IPv4 และ IPv6 ต่างก็สนับสนุน RSVP โพรโตคอลเพื่อการทำงานที่ซับซ้อนแตกต่างกัน รูปแบบส่วนหัวของ IPv6 มีส่วนที่ระบุชนิดของการส่งข้อมูลขนาด 20 บิต เพื่อช่วยในการจัดการ เป็นการสนับสนุนผู้ที่จะทำการสร้างเครือข่ายที่มีการทำงานแบบ Quality-of-Service (QoS) ผลิตภัณฑ์ที่มีการทำงานแบบนี้ยังคงอยู่ในระหว่างการออกแบบ แต่ IPv6 ก็ได้สร้างวางพื้นฐานเพื่อรองรับการทำงานนี้เอาไว้แล้ว ข้อดีอีกอย่างสำหรับการทำ QoS ใน IPv6 ก็คือส่วน flow label นี้ถูกกำหนดให้อยู่ในส่วนหัวของ IPv6 ทำให้สามารถแยกการจัดการเฉพาะเพื่อประโยชน์ในการเลือกเส้นทางได้ยิ่งไปกว่านั้น flow label สามารถใช้แยก flow ได้ในกรณีที่ข้อมูลถูกเข้ารหัสอยู่ (หมายเลขพอร์ตถูกซ่อนอยู่)

3.1.11 การเปลี่ยนไปสู่ IPv6

การเปลี่ยนจาก IPv4 ไปสู่ IPv6 นั้นสามารถเลือกได้จากหลายๆทาง บางคนเสนอให้มีการเปลี่ยนไปใช้ IPv6 ให้เร็วที่สุด แต่บางคนอาจจะชอบมากกว่าที่จะรอให้มีการใช้งาน IPv4 จนกระทั่งเกิดการขาดแคลนแอดเดรสเสียก่อน หรือมีอีกทางหนึ่งคือให้มีการใช้งานร่วมกันระหว่าง IPv6 และ IPv4 เพราะฉะนั้นผู้ออกแบบโพรโตคอลของ IETF จะต้องแน่ใจได้ว่าโฮสต์และเราเตอร์สามารถเปลี่ยนเป็น IPv6 ได้อย่างไม่มีปัญหา การเปลี่ยนจะต้องไม่เป็นการกีดกันโหนด IPv4 และจะต้องไม่ทำให้เกิดปัญหากับการใช้งานของผู้ใช้งานทั่วโลกด้วย การเปลี่ยนนั้นมีความยืดหยุ่นสำหรับผู้ดูแลระบบ เพราะสามารถจะทำการเปลี่ยนที่โหนดปลายทางก่อนหรือโหนดตัวกลางก่อนก็ได้ โหนดที่เริ่มทำการเปลี่ยนแปลงแล้วจำเป็จต้องสามารถทำงานร่วมกับ IPv4 ได้ต่อไปเป็นช่วงเวลาหนึ่ง (อาจจะกินเวลาหลายปีหรือไม่สามารถบอกได้ว่าถึงเมื่อไหร่) จึงมีการออกแบบ dual-stack สำหรับโฮสต์และเราเตอร์รวมทั้งการทำการมุดอุโมงค์ (Tunneling) IPv6 โดยผ่าน IPv4

3.1.12 ระบบดีเอ็นเอสใน IPv6

Domain Name Service (DNS) เป็นสิ่งที่ผู้ดูแลระบบจะต้องทำการจัดการก่อนที่จะทำการติดตั้ง dual-stack โฮสต์ ผู้ออกแบบใน IETF ได้มีการกำหนด “DNS Extension to Support IP Version 6” (RFC 1886) ในข้อกำหนดนี้ได้มีการประกาศชนิดเรคคอร์ด AAAA ขึ้นมาใช้จับคู่ระหว่างโดเมนเนม

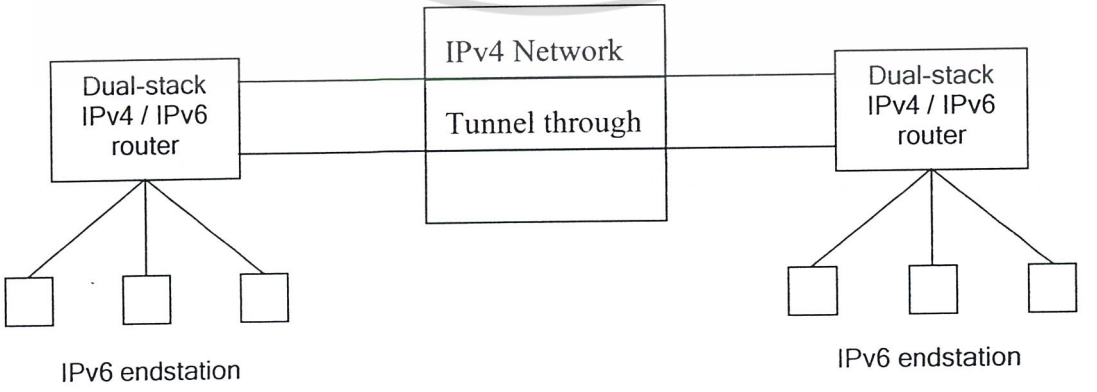
และไอพีแอดเดรส และ Domain name lookups แบบ 128 บิตก็ถูกประกาศด้วย เมื่อ dual-stack โฮสต์ส่งข้อความถามดีเอ็นเอส ถ้าคำตอบออกมาเป็น 32 บิต IPv4 ก็จะถูกใช้ แต่ถ้าคำตอบออกมาเป็น 128 บิต IPv6 ก็จะถูกใช้ หาก ดีเอ็นเอส ยังไม่ได้สนับสนุน IPv6 โอสต์สามารถสร้าง local name table ขึ้นมาใช้เองก่อนได้

3.1.13 การปรับเปลี่ยนโปรแกรมโปรแกรมประยุกต์สำหรับ IPv6

โปรแกรมประยุกต์ที่ไม่ได้มีการเรียกใช้งานฟังก์ชันที่ติดต่อกับเครือข่าย (ตัวอย่างเช่น ซ็อกเก็ต, ดีเอ็นเอส API หรือ มีการใช้ไอพีแอดเดรส) ไม่จำเป็นต้องมีการปรับเปลี่ยนใดๆสำหรับการทำงานใน dual-stack โฮสต์ สำหรับโปรแกรมที่มีการเรียกใช้ API ที่ติดต่อกับเครือข่ายเช่น โปรแกรมที่ใช้ ดีเอ็นเอส หรือ Socket จะต้องถูกเพิ่มความสามารถเพื่อจัดการกับแอดเดรสแบบ IPv6 และหากต้องการใช้ความสามารถอื่นๆเช่น Security , QoS ก็จะต้องทำการแก้ไขเพิ่มขึ้น

3.1.14 การจัดการเส้นทางในเครือข่าย IPv6/IPv4

เราเตอร์ที่ทำงานได้ทั้ง IPv6 และ IPv4 จะถูกติดตั้งเพิ่มมากขึ้น ในขณะเดียวกันเราเตอร์ที่ทำงานได้แต่ IPv4 ก็ยังคงอยู่มาก Multi-protocol extensions สำหรับ IPv6 ถูกประกาศโดย IETF มีการใช้งานกันอย่างมากใน 6bone ตั้งแต่ปี 1997 ถูกพัฒนาโดยบริษัทผู้ผลิตเราเตอร์และ gated daemon และยังมีโพรโตคอลสำหรับ IPv6 อื่นๆที่มีการใช้กันมากเช่น OSPF และ RIP ด้วย ผู้ดูแลระบบอาจจะตัดสินใจแยกเครือข่าย IPv6 แยกออกจากเครือข่าย IPv4 เพื่อแยกการดูแลออกจากกัน หรืออีกทางเลือกหนึ่งอาจจะรวมทั้ง 2 ระบบไว้ใน domain , area และซับเน็ตเดียวกันก็ได้ โดยเริ่มต้นแล้ว IPv6 โฮสต์อาจจะทำการเชื่อมต่อกับภายนอกโดยผ่าน IPv4 เราเตอร์ โดยใช้เทคนิคที่เรียกว่า Tunneling ทำการส่ง IPv6 แพ็กเก็ตใน IPv4 แพ็กเก็ต ช่วยให้สามารถใช้ประโยชน์จากเครือข่าย IPv4 ที่มีอยู่แล้วโดยไม่ต้องทำการเปลี่ยนแปลงใดๆ ดังรูปที่ 3.4



รูปที่ 3.4 IPv6 over IPv4 Tunneling

dual-stack เราเตอร์หรือโฮสต์จะทำการใส่ส่วนหัวของ IPv4 ข้างหน้า IPv6 แพ็กเก็ต (Encapsulates) เราเตอร์ IPv4 จะทำการส่งต่อโดยไม่รู้ว่ามี IPv6 แพ็กเก็ตซ่อนอยู่ ทางฝั่งผู้รับ dual-stack เราเตอร์หรือโฮสต์ก็จะทำการ Decapsulates แพ็กเก็ต IPv6 ออกมา

3.1.15 ออโตเมติกทันเนิล

Automatic Tunneling ใช้ IPv4 compatible address โดยการใช้ 0 เติมหน้าแอดเดรสแบบ IPv4 ให้มีขนาด 128 บิต โดยวิธีนี้อุปกรณที่เป็นปากทางเข้าอุโมงค์สามารถเปลี่ยน IPv6 ให้เป็น IPv4 ได้โดยง่าย แม้จะไม่สามารถใช้ประโยชน์จากจำนวนแอดเดรสที่เพิ่มขึ้นมาใน IPv6 ได้ แต่สามารถใช้ flow label , Authentication , Encryption , Multicast , Anycast ได้ และมีข้อดีคือสามารถปรับเปลี่ยนไปสู่ IPv6 ได้โดยไม่ต้องทำการแก้ไขโครงสร้าง address และ subnet เดิม และวิธีนี้จะไม่จำเป็นเมื่อ แบ็คโบน เราเตอร์ทั้งหมดสนับสนุน IPv6 เรียบร้อยแล้ว

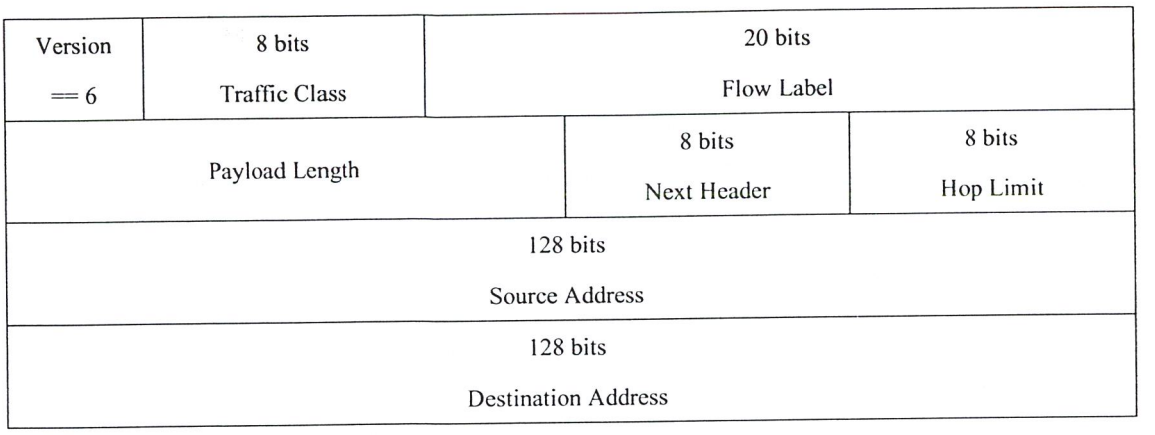
3.2 เหตุผลทางเทคนิค

3.2.1 เปรียบเทียบส่วนหัวของ IPv6 กับส่วนหัวของ IPv4

ทั้งสองแบบมีข้อมูลที่ระบุ version , source address และ destination address แต่รูปแบบส่วนหัวของ IPv6 จะดูเรียบง่ายกว่าเพื่อช่วยเพิ่มประสิทธิภาพการประมวลผลโดยโหนดระหว่างทาง ดังรูปที่ 3.5 และรูปที่ 3.6

<table>
<tr><td>Version
== 4</td><td>4 bits
IHL</td><td>8 bits
Type os Service</td><td colspan="2">16 bits
Total Length</td></tr>
<tr><td colspan="3">16 bits
Identification</td><td>4 bits
Flags</td><td>12 bits
Fragment Offset</td></tr>
<tr><td colspan="2">8 bits
Time to Live</td><td>8 bits
Protocol</td><td colspan="2">16 bits
Header Checksum</td></tr>
<tr><td colspan="5">32 bits
Source Address</td></tr>
<tr><td colspan="5">32 bits
Destination Address</td></tr>
<tr><td colspan="5">0 or more bits
IP options</td></tr>
</table>

รูปที่ 3.5 รูปแบบส่วนหัวของ IPv4



รูปที่ 3.6 รูปแบบส่วนหัวของ IPv6

ส่วนหัวของ IPv4 จะมีขนาดแปรผันได้ในขณะที่ IPv6 จะมีขนาดคงที่ 40 ไบต์ช่วยให้ผู้ออกแบบซอฟแวร์สำหรับเราเตอร์สามารถออกแบบเพื่อให้เกิดประสิทธิภาพสูงสุดได้ ประสิทธิภาพของระบบจะเพิ่มขึ้นเนื่องจากมีจำนวนข้อมูลในส่วนหัวลดลง ส่วน Header Length ถูกตัดทิ้งเนื่องจากมีขนาดคงที่ ส่วน Total Length ใน IPv4 ถูกเปลี่ยนให้เป็น Payload Length โดยไม่นวมขนาดของส่วนหัวคือ 40 ไบต์ และมีขนาดได้สูงสุดเป็น 64 กิโลไบต์ แพ็กเก็ตที่มีขนาดใหญ่กว่านี้เรียกว่า Jumbogram สามารถส่งได้โดยการกำหนดให้ Payload Length มีค่าเป็น 0 และมีการเพิ่มส่วนเพิ่มเติมส่วนหัวเข้าไป ส่วน Time-to-live (TTL) ถูกเปลี่ยนชื่อเรียกเป็น Hop Limit ซึ่งช่วยบรรยายความหมายได้ดีกว่า ส่วนนี้ใช้เพื่อหยุดลูปโดยจะลดลงครั้งละหนึ่งเสมอเมื่อผ่านฮอปและจะถูกตั้งค่ามาโดยโหนดผู้ส่ง เมื่อค่านี้กลายเป็น 0 แพ็กเก็ตนั้นจะถูกทิ้งไป มีค่าสูงสุดได้ถึง 255 ซึ่งเพียงพอสำหรับเครือข่ายขนาดใหญ่ มีหลายส่วนที่ถูกตัดทิ้งไปคือ Fragment offset , Identification , Flags , Checksum ส่วน Type of Service ใน IPv4 ถูกแทนที่โดย Traffic Class และ Flow Label ส่วนที่เกี่ยวกับการแฟรกเมนต์ คือ Offset , Identification และ Flags ถูกย้ายไปในส่วน Option และสุดท้ายส่วน Checksum นั้นถูกยกเลิกไปเนื่องจากเกิดความซ้ำซ้อนกับการตรวจสอบโดยโพรโตคอลในระดับชั้นอื่นๆ และเป็นการเพิ่มภาระให้กับเราเตอร์

3.2.2 ส่วนเพิ่มเติมส่วนหัว

ในส่วนหัวของ IPv4 จะมีส่วนออปชั่นที่ช่วยในเรื่อง Security , Source routing และการทำงานอื่นๆ แต่ข้อมูลในส่วนนี้มักไม่ถูกใช้เนื่องจากเราเตอร์โดยส่วนมากแล้วจะให้ความสำคัญกับแพ็กเก็ตที่มีส่วนออปชั่นน้อย ดังนั้น IPv6 จึงได้นำส่วนออปชั่นไปไว้ในส่วนที่เรียกว่าส่วนเพิ่มเติมส่วนหัว โดยจะอยู่หลังส่วนหัวและอยู่ก่อนส่วนหัวของโพรโตคอลในระดับชั้น Transport ในส่วน Payload และจะจัดการเกี่ยวกับด้าน Security , Fragmentation , Source Routing และการทำงานพิเศษอื่นๆ โดยไม่มีการจำกัดจำนวนของส่วนเพิ่มเติมส่วนหัว รูปที่ 3.7 แสดงส่วนเพิ่มเติมที่จัดการเกี่ยวกับ Fragmentation และ Encryption

IPv6 Header	Fragmentation Header	Encryption Header	Transport, etc

รูปที่ 3.7 ส่วนเพิ่มเติมส่วนหัวของ IPv6

ส่วน Protocol ถูกแทนที่โดยส่วน Next Header ทุกๆส่วนหัวรวมทั้งส่วนเพิ่มเติมจะต้องมีข้อมูลนี้ เพื่อใช้บอกชนิดของส่วนที่จะตามมา ซึ่งอาจจะเป็นส่วนหัวของ TCP หรือ UDP หรือว่าอาจจะเป็นส่วนเพิ่มเติมส่วนหัวอื่นๆ IETF ได้ทำการกำหนดส่วนเพิ่มเติมส่วนบางส่วนแล้วและได้แนะแนวทางในการเรียงลำดับเอาไว้ดังนี้

- Primary IPv6 Header
- Hop-by-Hop options header
- Destination options header 1
- Source routing header
- Fragmentation header
- Authentication header
- IPv6 Encryption header
- Destination options header 2

ตามด้วยส่วนหัวของโพรโตคอลในระดับที่สูงกว่าหรือ payload ในแต่ละส่วนเพิ่มเติมจะเกิดขึ้นเพียงครั้งเดียวในแต่ละแพ็กเก็ตยกเว้น Destination header

3.2.3 โครงสร้างแอดเดรสของ IPv6

เรื่องที่เป็นข้อโต้แย้งกันมากในการเปรียบเทียบระหว่าง IPv6 และ IPv4 ก็คือขนาดในส่วนแอดเดรสของทั้งสองโพรโตคอล แต่ส่วนที่มีความสำคัญเท่าเทียมกันก็คือการจัดโครงสร้างลำดับชั้นการควบคุมแอดเดรส ใน IPv4 นั้นได้แบ่งออกเป็นคลาส A , B และ C ที่ทำการแบ่งส่วนที่เป็นเครือข่ายและโฮสต์ แต่ไม่ได้มีการกำหนดให้แอดเดรส ในระดับที่สูงกว่าควบคุมแอดเดรส ในระดับที่ต่ำกว่าหลายๆส่วนๆ เหมือนกับระบบโทรศัพท์ที่แบ่งออกเป็น area code ทำให้เกิดการสูญเสียที่อยู่โดยไม่จำเป็น มีการพยายามที่จะแก้ไขโดยใช้ CIDR เพื่อยืดอายุของ IPv4 แต่ก็ยังไม่มีการอิมพิวเมนต์ ในระบบอินเทอร์เน็ตในเครือข่ายใหญ่ๆ ทำให้ประโยชน์ในการลดขนาดตาราง และ การใช้แอดเดรส ได้เต็มประสิทธิภาพที่เกิดจาก CIDR ยังไม่ถูกใช้อย่างเต็มที่ ทำให้ IPv4 ยังคงมีการสูญเสียแอดเดรส และยังคงมีการส่งข้อมูลที่ไม่มีประสิทธิภาพและตารางการเลือกเส้นทางที่ใหญ่อยู่

3.2.4 ลำดับการควบคุมแอ็ดเดรสใน IPv6

จากประสบการณการใช้งาน IPv4 ทำให้ผู้ออกแบบ IPv6 ได้คำนึงถึงการออกแบบโครงสร้างแอดเดรส ให้มีการแบ่งสืบทอดการปกครองออกเป็นส่วนๆสำหรับเส้นทางการส่งข้อมูลทั่วโลก ที่จุดสูงสุดของลำดับชั้นจะเป็นจะเป็นระดับประเทศ (Top Level Aggregators - TLA) และ TLAs จะแบ่งกลุ่ม

ของแอดเดรส ให้กับ Next Level Aggregator (NLA) ซึ่งก็คือ ผู้ที่ทำหน้าที่แจกจ่ายและองค์กรที่มีเครือข่ายขนาดใหญ่ จากนั้นก็จะจัดต่อให้กับสมาชิก การเลือกเส้นทางจะมีประสิทธิภาพเพราะ NLAs ที่อยู่ภายใต้ TLA เดียวกันจะมีPrefixเดียวกัน และจะต่อเนื่องไปถึงสมาชิกด้วย

3 bits	13 bits	8 bits	24 bits	16 bits	64 bits
FP	TLA ID	RES	NLA ID	SLA ID	Interface ID

รูปที่ 3.8 การแบ่ง address โดยยึดหลักการเป็นส่วนประกอบ (Aggregate)

จากรูปที่ 3.8 3 บิตแรกจะระบุชนิดของแอดเดรส(Unicast , Multicast ฯลฯ) 13 บิต ต่อไปจะระบุ TLA ซึ่งกระจายไปทั่วโลก 8 บิต ถัดมาสำรองไว้ใช้ในอนาคต และต่อมาก็เป็น NLA และส่วนต่อมาก็จะเป็น Site Level Aggregator (SLA) สุดท้ายก็เป็น Interface ID

ส่วน แบ็คโบน เราเตอร์จะต้องเก็บเส้นทางมากกว่า 40000 เส้นทาง แม้ว่าขนาดของอินเทอร์เน็ตจะขยายขึ้นเรื่องๆ แต่ด้วยวิธีการสืบทอดการปกครองแล้ว ทำให้เราเตอร์สามารถจำกัดขนาดตารางการเลือกเส้นทางได้ ผู้แจกจ่ายสามารถกำหนดแอดเดรสโดยไม่ต้องขออนุญาติจากส่วนกลาง

บทที่ 4

การจัดการแอดเดรสของโพรโตคอล IPv6

(IPv6 Addressing)

IPv6 ใช้การกำหนดแอ็ดเดรสขนาด 128 บิต ทำให้สามารถอ้างอิงตำแหน่งได้มากกว่า IPv4 หลายเท่า โดยมีการแบ่งแยกประเภทของตำแหน่งออกเป็น 3 ประเภท

Unicast addresses – เป็นการกำหนดแอ็ดเดรสสำหรับอินเทอร์เฟซเดียวสามารถแบ่งแยกย่อยออกได้เป็น provider-based addresseses , site-local-use addresses , link-local-use addresses เมื่อแพ็กเก็ตถูกระบุที่อยู่เป็น unicast address โพรโตคอลจะทำการส่งต่อแพ็กเก็ตไปยังอินเทอร์เฟซที่ระบุตามตำแหน่งนั้น

Anycast addresse -- เป็นการกำหนดประเภทแอ็ดเดรสที่ถูกระบุเพิ่มขึ้นมาใหม่จาก IPv4 แอ็ดเดรสในประเภทนี้จะถูกกำหนดให้กับอินเทอร์เฟซมากกว่าหนึ่ง เมื่อแพ็กเก็ตถูกระบุแอ็ดเดรสปลายทางเป็น anycast address โพรโตคอลจะทำการส่งต่อแพ็กเก็ตนั้นไปยังอินเทอร์เฟซที่อยู่ใกล้ที่สุดที่ระบุแอ็ดเดรสเป็นตำแหน่งนั้น

Multicast Address – เป็นรูปแบบการกำหนดแอ็ดเดรสที่สามารถกำหนดกลุ่มได้ถึง 2^{120} กลุ่มมัลติแคสต์แอ็ดเดรส แพ็กเก็ตจะถูกส่งไปยังทุกๆอินเทอร์เฟซที่อยู่ในกลุ่มของมัลติแคสต์แอ็ดเดรส

การกำหนดแอ็ดเดรสทุกประเภทจะถูกกำหนดให้กับอินเทอร์เฟซ ทุกๆอินเทอร์เฟซจะมีสภาพเป็นโหนดหนึ่ง ยูนิแคสต์แอ็ดเดรส หนึ่งแอ็ดเดรส จะถูกกำหนดให้กับหนึ่งอินเทอร์เฟซ แต่หนึ่งอินเทอร์เฟซอาจจะมีแอ็ดเดรส ได้หลายประเภทและประเภทละหลายแอ็ดเดรส โดยจะมีข้อยกเว้นคือ

1. ยูนิแคสต์ หนึ่งแอ็ดเดรสสามารถถูกกำหนดให้กับหลายอินเทอร์เฟซได้ถ้ามีการอิมพลีเมนต์ให้มีเพียงอินเทอร์เฟซเดียวที่ติดต่อกับ internet layer ซึ่งมีประโยชน์ในการทำ load-sharing
2. เราเตอร์สามารถเชื่อมต่อแบบจุดต่อจุด (point-to-point) กับโหนดที่ไม่ได้กำหนดที่อยู่ได้ถ้าโหนดนั้นไม่ได้เป็นต้นทางหรือปลายทางของ IPv6 ดาต้าแกรม

4.1 การเขียน IPv6 ในรูปแบบข้อความ

มีการเขียนได้ 3 แบบ

1. แบบมาตรฐาน คือ การเขียนในรูปแบบเลขฐานสิบหก (RFC 1884) มีรูปแบบ

X:X:X:X:X:X:X:X โดย X เป็น เลขฐานสิบหก 16 บิต ตัวอย่างเช่น

FCAB:B897:4578:0000:3310:6550:7110:8250

หากมีช่วงที่ขึ้นต้นด้วย 0 สามารถละเว้นได้เช่น

FCAD:0:0:5:26:999:1000:1

2. แบบย่อ เป็นการเขียนแบบย่อโดยใช้เครื่องหมายโคลอนติดกัน 2 ตัวเป็น :: แทนช่วงที่มีแต่เลขศูนย์ทั้งช่วงติดต่อกันหลายช่วง แต่มีข้อแม้คือสามารถใช้ได้ครั้งเดียว มีตัวอย่างการใช้ดังนี้

แบบเต็ม	แบบย่อ
1080:0:0:0:8:800:200C:417A	1080::8:800:200C:417A
FF01:0:0:0:0:0:0:43	FF01::43
0:0:0:0:0:0:0:1	::1
0:0:0:0:0:0:0:0	::
0:0:0:0:1:9:0:0	::1:9:0:0 or 0:0:0:0:1:9::

ตารางที่ 4.1 ตัวอย่างการเขียนในรูปแบบย่อ

3. แบบผสม เป็นการใช้รูปแบบของ IPv4 และ IPv6 ผสมกัน มีรูปแบบเป็น X:X:X:X:X:X.D.D.D.D โดย X เป็นการเขียนเลขฐานสิบหกใน 6 ส่วนแรก(ส่วนละ 16 บิต) และ D แสดงการเขียนในรูปแบบเลขฐานสิบใน 2 ส่วนหลังโดยเขียนแยกเป็น 4 ส่วนส่วนละ 8 บิต ตัวอย่างเช่น
::A1F6:3415 สามารถเขียนได้เป็น 0:0:0:0:0:0:161.246.52.21

ในกรณีที่มีการใช้ address prefixes สามารถเขียนได้โดยใช้เครื่องหมาย '/' ต่อท้ายตามด้วยเลขฐานสิบที่เป็นขนาดของพรีฟิกซ์ โดยหากมีหารเขียนแบบย่อแล้วลงท้ายด้วย :: สามารถละ :: โดยเขียนพรีฟิกซ์ต่อท้ายได้เลย ตัวอย่างเช่น
12AB:0:0:CD30:0:0:0:0 มีพรีฟิกซ์ 60 บิตเป็น 12AB00000000CD3 สามารถเขียนได้ดังนี้
12AB:0000:0000:CD30:0000:0000:0000:0000/60
12AB::CD30:0:0:0:0/60
12AB:0:0:CD30::/60
12AB:0:0:CD30/60

4.2 การแบ่งส่วนแอ็ดเดรสของ IPv6

บิตที่มีนัยสำคัญสูงจะบ่งบอกประเภทของแอ็ดเดรส เรียกว่า Format prefixes(FP) ดังตารางต่อไปนี้

Allocation	Prefix	อัตราส่วนต่อแอ็ดเดรสทั้งหมด
Reserved	0000 0000	1/256
Unassigned	0000 0001	1/256
Reserved for NSAP Allocation	0000 001	1/128

Unassigned	0000 010	1/128
Unassigned	0000 011	1/128
Unassigned	0000 1	1/32
Unassigned	0001	1/16
Aggregatable Global Unicast Address	001	1/8
Unassigned	010	1/8
Unassigned	011	1/8
Unassigned	100	1/8
Unassigned	101	1/8
Unassigned	110	1/8
Unassigned	1110	1/16
Unassigned	1111 0	1/32
Unassigned	1111 10	1/64
Unassigned	1111 110	1/128
Unassigned	1111 1110 0	1/512
Link-Local-Use Addresses	1111 1110 10	1/1024
Site-Local-Use Addresses	1111 1110 11	1/1024
Multicast Addresses	1111 1111	1/256

ตารางที่ 4.2 แสดง Format prefix

มีการกำหนดการใช้งานแอ็ดเดรสไปแล้วประมาณ 15% แอ็ดเดรสที่ขึ้นต้นด้วย FF จะเป็นมัลติแคสต์

4.3 ยูนิแคสต์แอ็ดเดรส

IPv6 Unicast Address ถูกกำหนดให้มาสค์แบบต่อเนื่อง ในลักษณะแบบ Classless Interdomain Routing (CIDR) มีการใช้งาน 3 ประเภ

1. Provider-based unicast addresses ISP เป็นผู้จัดสรรค์แอ็ดเดรสนี้ให้กับองค์กรโดยไม่ซ้ำกันทั่วโลก
2. Site-local-use address กำหนดให้ใช้กับอุปกรณ์ภายใน Intranet โดยองค์กรต่างๆสามารถเข้าเชื่อมต่อกับ Internet ได้ในภายหลังโดยไม่กระทบต่อโครงสร้างเครือข่าย
3. Link-local-use address ใช้ในการเชื่อมต่อระหว่างลิ้งค์ต่อลิ้งค์เป็นส่วนตัว เช่นเครื่องคอมพิวเตอร์แบบพกพาเชื่อมต่อผ่านทางสายโทรศัพท์หรือคลื่นวิทยุ

Unicast Address Format

โหนด IPv6 อาจจะไม่รู้ถึงโครงสร้างของแอ็ดเดรส ดังนั้นจึงพิจารณาเป็นยูนิแคสต์แอ็ดเดรสที่ไม่มีโครงสร้างภายในเป็น Node address 128 บิตดังรูปที่ 4.1

128 bits (16 octets)
Node address

รูปที่ 4.1 รูปแบบ Node address

โหนดบางโหนดอาจจะรู้ subnet prefixes ที่มันเชื่อมต่ออยู่ ในกรณีนี้โหนดนั้นจะแบ่งแอ็ดเดรสออกเป็น subnet prefixes n บิตและ Interface ID 128-n บิตดังรูปที่ 4.2

n bits	128 – n bits
Subnet prefix	Interface ID

รูปที่ 4.2 รูปแบบที่แบ่งออกเป็น Subnet prefix และ Interface ID

รูปแบบของยูนิแคสต์บางแบบมีการใช้งานมากกว่าแบบอื่น มีรูปแบบอย่างง่ายที่ใช้ประโยชน์ในการทำ Autoconfiguration โดยใช้ identifier ที่ไม่ซ้ำกันทั่วโลก ตัวอย่างเช่น โหนดสามารถรับรู้ Subnet ID จากการประกาศของ Router ที่เชื่อมต่ออยู่ และโหนดนั้นสามารถสร้าง IPv6 address ของตัวเองได้โดยการใช้ IEEE MAC address ของตัวมันเองเป็น Interface ID หรืออาจใช้ link-layer address ชนิดอื่นเช่น E.164 เป็น Interface ID ได้
มีโครงสร้างดังรูปที่ 4.3

n bits	80-n bits	48 bits
Subscriber prefix	Subnet ID	Interface ID

รูปที่ 4.3 รูปแบบที่แบ่งออกเป็น Subnet prefix , Interface ID และมี Subscriber prefix

ในบางครั้ง องค์กรใดๆอาจจำเป็นต้องเพิ่มลำดับชั้นในโครงสร้างภายใน ก็ทำได้โดยแบ่ง Subnet ID ออกเป็น Area ID และ Subnet ID ดังรูปที่ 4.4

S bits	n bits	m bits	128 -s- n-m bits
Subscriber prefix	Area ID	Subnet prefix	Interface ID

รูปที่ 4.4 รูปแบบที่มีการแบ่งย่อยเป็น Area ID

The Unspecified Address (RFC 1884)

Address 0:0:0:0:0:0:0:0 ถูกกำหนดให้เป็น Unspecified Address โดยสามารถเขียนในรูปแบบ 0::0 ได้ ถูกใช้งานในกรณีที่อินเทอร์เฟซยังไม่ได้รับการกำหนด Address ให้ตัวอย่างเช่น IPv6 ดาต้าแกรมที่ถูกส่งจากโฮสต์ที่พึ่งเริ่มต้นการทำงานและยังไม่รู้ address ของตัวเองจะมี Source address เป็น Unspecified address โดย Unspecified address จะไม่ถูกใช้เป็น Destination address ทั้งใน IPv6 ดาต้าแกรมและ IPv6 Routing Header

The Loopback Address

ถูกกำหนดให้มีแอ็ดเดรส เป็น 0:0:0:0:0:0:0:1 และสามารถเขียนได้เป็น 0::1 โหนดใดๆจะใช้แอ็ดเดรสนี้ในการส่ง IPv6 ดาต้าแกรมสู่ตัวมันเอง โดย

- ห้ามใช้แอ็ดเดรสนี้เป็น Source หรือ Destination ในการส่งดาต้าแกรมออกไปยังอินเทอร์เฟซใดๆ
- ห้ามกำหนดให้เป็นแอ็ดเดรสของโหนดใดๆ

IPv6 Address ที่เกิดจาก IPv4 Address

การใช้งาน IPv6 ร่วมกับ IPv4 มีอยู่ 2 รูปแบบ

รูปแบบแรก ให้ Host หรือ Router ทำการ Tunnel IPv6 แพ็กเก็ต ผ่านระบบเครือข่าย IPv4 โหนดใดๆที่ใช้ระบบนี้จะทำการกำหนดยูนิแคสต์แอ็ดเดรสที่มี IPv4 แอ็ดเดรสเป็น 32 บิตล่าง โดยเรียกว่า IPv4-compatible IPv6 address และมีพรีฟิกซ์เป็น 0 จำนวน 96 บิตดังรูป 4.5

80 bits	16 bits	32 bits
0000.....0000	0000	IPv4 address

รูปที่ 4.5 รูปแบบของ IPv4-compatible address

รูปแบบที่สอง เรียกว่า IPv4-mapped IPv6 ใช้เป็นแอ็ดเดรสของโหนด IPv4 ที่ไม่สนับสนุน IPv6 เพื่อติดต่อกับ IPv6 โดยแอ็ดเดรสนี้จะไม่ถูกกำหนดให้กับ IPv6 โหนดใดๆ มีรูปแบบดังนี้

80 bits	16 bits	32 bits
0000.....0000	FFFF	IPv4 address

รูปที่ 4.6 รูปแบบของ IPv4--mapped IPv6

Network Service Access Point Address (NSAP)

กำหนดให้ขึ้นต้นด้วย 0000001 มีลักษณะคล้ายกับ CIDR ใน RFC 1888 ได้กำหนด NSAP-IPv6 Mapping ไว้

7 bits	121 bits
0000001	To be defined

รูปที่ 4.7 รูปแบบของ Network Service Access Point Address (NSAP)

Internetwork Packet Exchange Addresses (IPX)

กำหนดให้ขึ้นต้นด้วย 0000010 ดังรูป

7 bits	121 bits
0000010	To be defined

รูปที่ 4.8 รูปแบบของ Internetwork Packet Exchange Addresses (IPX)

Provider-Based Global Unicast Addresses

ขึ้นต้นด้วย 010 มีโครงสร้างคล้ายกับ IPv4 แบบ CIDR มีการคาดกันว่ารูปแบบนี้จะมีการใช้กันมากในโหนด IPv6 ที่เชื่อมต่อเข้ากับอินเทอร์เน็ตประกอบด้วย Format Prefix , Registry ID , Provider ID , Subscriber ID และ Intra-Subscriber ดังรูป

3 bits	n bits	m bits	o bits	125-n-m-o bits
010	Registry ID	Provider ID	Subscriber ID	Intra-Subscriber

64 bits

รูปที่ 4.9 รูปแบบของ Provider-Based Global Unicast Addresses

1) Format Prefix

มีค่าเป็น 010 ตามตาราง ระบุประเภทของแอ็ดเดรสตามตาราง Format Prefix (FP)

2) Registry ID

ใช้ในการแบ่งแยกแอ็ดเดรสออกเป็นพื้นที่ทางภูมิศาสตร์

3) Provider ID

ระบุหมายเลขของผู้ที่ทำการแจกจ่ายแอ็ดเดรส (Provider) ในแต่ละพื้นที่

4) Provider ID

ใช้ในการแบ่งแยกออกเป็นพื้นที่ย่อยๆในเขตรับผิดชอบของแต่ละ Provider โดยเป็นหน้าที่ของ Provider ที่ทำการกำหนดโครงสร้างในส่วนนี้เอง อาจจะแบ่งแยกออกเป็นกลุ่มย่อยๆ ตาม topology

5) Intra-Subscriber

แล้วแต่ Provider จะทำการกำหนดในส่วนนี้ โดยอาจจะแบ่งแยกออกเป็น SubnetID และ Interface ID และใช้ Autoconfiguration

Local-Use Addresses

RFC 1884 แบ่งออกเป็น 2 ประเภทคือ

Link-local และ Site-local โดย Link-local ใช้ในการเชื่อมต่อแบบ single มีจุดประสงค์เพื่อใช้ในการ Autoconfiguration , Neighbor Discovery หรือในกรณีที่ไม่มี Router โดย Router จะไม่ทำการส่งต่อแพ็กเก็ตที่มี Source address เป็น Link-local ออกไปนอกเครือข่าย มีรูปแบบดังรูป

10 bits	n bits	118-n bits
1111111010	0	Interface ID

รูปที่ 4.10 รูปแบบของ Link-local

สำหรับ Site-local ใช้ในกรณีที่เครือข่ายไม่ได้ทำการเชื่อมต่อกับ Internet โดยเครือข่ายนั้นไม่ต้องการ Address prefix และเมื่อต้องการเชื่อมต่อกับ Internet ในภายหลังก็ทำได้โดยการแทนที่ Site-local Prefix กับ Subscriber prefix Router จะไม่ทำการส่งต่อแพ็กเก็ตที่มี Source address เป็น Site-local ออกไปนอกเครือข่าย มีรูปแบบดังรูป

10 bits	n bits	M bits	118-n bits
1111111011	0	Subnet ID	Interface ID

รูปที่ 4.11 รูปแบบของ Site-local

4.4 เอนี่แคสต์แอ็ดเดรส

สามารถถูกกำหนดให้ได้มากกว่าหนึ่งอินเทอร์เฟซ เมื่อแพ็กเก็ตถูกส่งไปยังเอนี่แคสต์แอ็ดเดรส แพ็กเก็ตนั้นจะถูกส่งไปยังอินเทอร์เฟซที่ใกล้ที่สุด โดยเอนี่แคสต์แอ็ดเดรสจะอยู่ในส่วนของยูนิแคสต์แอ็ดเดรส และจะเป็นรูปแบบใดก็ได้ จึงไม่สามารถแยกได้ระหว่างยูนิแคสต์และเอนี่แคสต์ ในทางปฏิบัติก็คือการกำหนดยูนิแคสต์แอ็ดเดรสเดียวกันให้กับโหนดหลายๆ โหนดนั่นเอง
แต่ต้องมีการระบุเพิ่มเติมให้แต่ละโหนดทราบว่าเป็นเอนี่แคสต์แอ็ดเดรสและมีข้อห้ามอยู่ 2 ประการคือ

- ห้ามใช้เอนี่แคสต์แอ็ดเดรส เป็น Source address ของ IPv6 แพ็กเก็ต
- ห้ามกำหนดเอนี่แคสต์ให้กับโฮสต์ใดๆ ให้กำหนดได้กับ Router เท่านั้น

Subnet Router Anycast Address จะถูกกำหนดขึ้นมีรูปแบบดังรูป

n bits	128-n bits
Subnet prefix	00000000000

รูปที่ 4.12 รูปแบบของ Subnet Router เอนี่แคสต์แอ็ดเดรส

แพ็กเก็ตจะถูกส่งไปยังเราเตอร์ตัวใดตัวหนึ่งที่อยู่ใน Subnet นั้น โดย Router ทุกตัวจะต้องสนับสนุนกระบวนการนี้

4.5 มัลติแคสต์แอ็ดเดรส

มีรูปแบบดังต่อไปนี้

8 bits	4 bits	4 bits	112 bits
11111111	Flgs	Scop	Group ID

0	0	0	T

รูปที่ 4.13 รูปแบบของมัลติแคสต์แอ็ดเดรส

ทุกๆอินเทอร์เฟซที่เป็นมัลติแคสต์จะได้รับแพ็กเก็ตที่ระบุปลายทางเป็นมัลติแคสต์นั้นๆ
มัลติแคสต์แอ็ดเดรส สามารถกำหนดให้กับระบบ หรือ เครือข่ายใดๆ และห้ามใช้เป็น Source Address ใน IPv6 คาต้าแกรม หรือ Routing header ขึ้นต้นด้วย FF แล้วตามด้วย Flgs โดย 3 บิตแรกของ Flgs ยังไม่มีการใช้งานและถูกกำหนดให้เป็น 0 และบิต T ถ้ามีค่าเป็น 0 หมายความว่า มัลติแคสต์แอ็ดเดรส นั้นเป็นแบบถาวรและใช้ได้ทั่วโลก แต่ถ้าเป็น 1 ก็เป็นการใช้ชั่วคราว

ถัดจาก Flgs ก็จะเป็น Scop จำกัด Scope ดังตารางต่อไปนี้

Value	Description
0	Reserved
1	Node-local scope
2	Link-local scope
3	(unassigned)
4	(unassigned)
5	Site-local scope
6	(unassigned)
7	(unassigned)
8	Organization-scope
9	(unassigned)
A	(unassigned)
B	(unassigned)
C	(unassigned)
D	(unassigned)
E	Global scope
F	Reserved

ตารางที่ 4.3 ความหมายต่างของส่วน Scop

Group ID จะเป็นตัวบอกกลุ่มแต่ละกลุ่มโดยอาจจะเป็นได้ทั้งแบบถาวรและไม่ถาวร

DNS ที่สนับสนุน IPv6 จะสร้างเรคคอร์ดใหม่สำหรับ IPv6 โดยมีชนิดเป็น A6 ทำให้แยกความแตกต่างกับ IPv4 ซึ่งมีชนิดเป็น A ได้

บทที่ 5

ข้อกำหนดคุณลักษณะของ IPv6

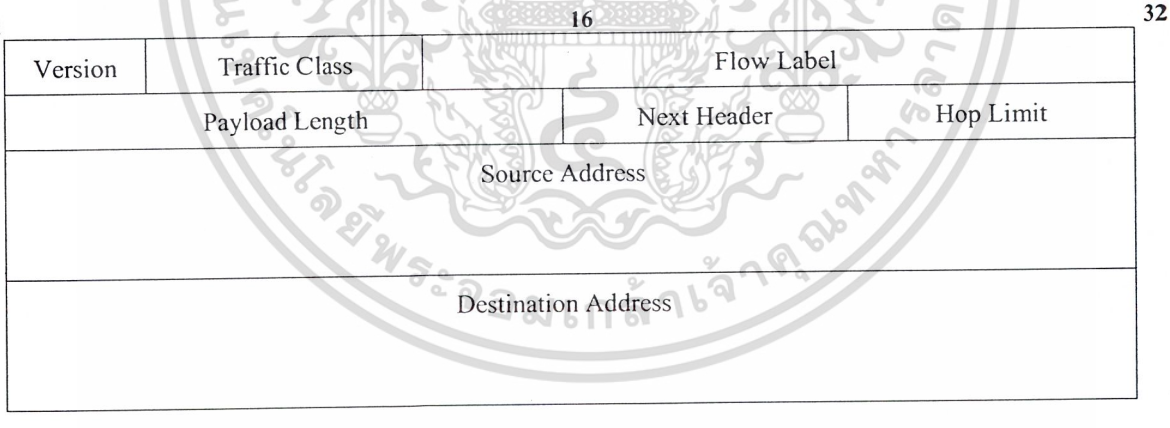
(IPv6 Specification)

ข้อมูลที่ส่งบนเครือข่ายจะแบ่งออกเป็นส่วนๆ เรียกว่าแพ็กเก็ตหรือดาต้าแกรม ในแพ็กเก็ตหรือดาต้าแกรมจะประกอบไปด้วยข้อมูลต่างๆ เช่น เวอร์ชันของโพรโตคอล ส่วนตรวจสอบความผิดพลาดและความถูกต้องของข้อมูล (Error Detection and Correction) ข้อมูลเกียวกับโหนดที่ส่งและโหนดที่รับ ในแต่ละแพ็กเก็ตหรือดาต้าแกรมจะประกอบไปด้วยสองส่วนคือ ส่วนหัวของมันเองและส่วนข้อมูล

ส่วนหัวนั้นจะบรรจุข้อมูลที่แตกต่างกันไปตามโพรโตคอลที่ใช้ และส่วนข้อมูลนั้นสามารถเป็นได้ทั้งข้อมูลแบบ ตัวหนังสือ หรือ ข้อมูลแบบไบนารี่

5.1 โครงสร้างข้อมูลส่วนหัวของ IPv6 (IPv6 Header)

ส่วนหัวของ IPv6 มีขนาดเพียงสองเท่าของส่วนหัวของ IPv4 คือมีขนาด 40 ไบต์ แม้ว่า IPv6 จะมีขนาดแอดเดรสมากกว่าถึงสี่เท่า (IPv4 มีแอดเดรสขนาด 32 บิต IPv6 มีแอดเดรสขนาด 128 บิต) ทั้งนี้เนื่องจากในส่วนหัวของ IPv6 ได้มีการตัดข้อมูลบางส่วนออกไปเพื่อเหตุผลทางด้านประสิทธิภาพ (ทำให้เวลาในการประมวลผลของแต่ละแพ็กเก็ตน้อยเท่าที่จะเป็นไปได้ และจำกัดการใช้แบนด์วิทธ์) เนื้อหาของบทนี้ส่วนใหญ่มาจาก RFC-2460



รูปที่ 5.1 แสดงโครงสร้างข้อมูลส่วนหัวของ IPv6

Version : มีขนาด 4 บิต ใช้บอกหมายเลขรุ่น (ใช้ 0x06)

Traffic Class : มีขนาด 8 บิต ดูรายละเอียดในหัวข้อทราฟฟิกคลาส (Traffic Class)

Flow Label : มีขนาด 20 บิต ดูรายละเอียดในหัวข้อโฟลเลเบิล (Flow Label)

Payload Length : มีขนาด 16 บิต ใช้บอกขนาดของข้อมูลทั้งหมดที่ต่อจากส่วนหัวของ IPv6 เนื่องจากมีขนาด 16 บิต หมายความว่าข้อมูลจริงอาจจะมีขนาดยาวได้ถึง 64 กิโลไบต์ ถ้าค่านี้

แสดงค่าเป็นศูนย์และค่า Next Header มีค่าเป็นศูนย์ แสดงถึงมีการใช้ฮ็อปบายฮ็อปออปชั่น (Hop-by-Hop Options) หมายถึงแพ็กเก็ตนี้มีขนาดความยาวจริงของแพ็กเก็ตได้จากจัมโบแกรมเพโหลดออปชั่น (Jumbo Payload Option – RFC-2675)

Next Header : มีขนาด 8 บิต ใช้ระบุถึงข้อมูลที่ต่อจากส่วนหัวนี้ เช่น 51 จะหมายถึง ส่วนหัวของ การพิสูจน์ตัวจริง (Authentication) ใช้ค่าเดียวกันกับ ค่า Protocol ใน IPv4 (RFC-1700) ดังตัวอย่างในตารางที่ 5.1

Decimal	Protocol
0	Reserved
1	ICMP
2	IGMP
6	TCP
17	UDP
43	SIP-SR (Source Route)
44	SIP-FRAG (Fragment)
50	SIPP -ESP(SIPP Encapsulating Security Payload)
51	SIPP-AH (SIPP Authetication Header)
58	ICMPv6
59	No Next Header
60	Destination Options

ตารางที่ 5.1 แสดงตัวอย่างที่ระบุใน RFC-1700

Hop Limit : มีขนาด 8 บิต ใช้ระบุจำนวนของฮ็อป (Hop) ที่แพ็กเก็ตสามารถส่งผ่านจากต้นทางถึงปลายทางได้ โดยจะลดลงครั้งละหนึ่งในแต่ละจุดที่ส่งผ่านแพ็กเก็ตนี้ แพ็กเก็ตจะถูกทิ้งไปถ้าค่านี้แสดงค่าเป็นศูนย์

Source Address : มีขนาด 128 บิตแสดงแอดเดรสของผู้ส่งข้อมูล

Destination Address : มีขนาด 128 บิตแสดงแอ็กเดรสของผู้รับข้อมูล (อาจไม่ใช่ผู้รับจริง ถ้ามีส่วนหัวของการหาเส้นทางปรากฏในแพ็กเก็ต)

ทราฟฟิกคลาส (Traffic Class)

ค่าของทราฟฟิกคลาสใน IPv6 ใช้เพื่อให้เราเตอร์หรือโหนดระบุถึงความแตกต่างหรือความสำคัญของแพ็กเก็ต โดยใช้เหมือนกับค่า Type of Service ใน IPv4 เพื่อจัดการบริการแบบต่างๆ ให้กับแพ็กเก็ต (การใช้ค่านี้ยังอยู่ในระหว่างการทดลอง) โดยค่าบิตต่างๆ และความหมายของทราฟฟิกคลาสจะกำหนดในภายหลังเมื่อมีข้อตกลงที่เป็นที่ยอมรับ

โฟลเลเบิล (Flow Label)

ค่าของโฟลเลเบิลในส่วนหัวของ IPv6 ใช้โดยโหนดต้นทางเพื่อเป็นตัวระบุว่าแพ็กเก็ตนั้นๆ ต้องการการจัดการหรือการบริการแบบไหน เช่น การบริการแบบเวลาจริง (Real-time Service) เมื่อเราเตอร์รับแพ็กเก็ตเข้ามา ก็ตรวจสอบค่าของโฟลเลเบิลและจัดการบริการที่เหมาะสมให้กับแพ็กเก็ต การใช้โฟลเลเบิลนั้นยังอยู่ในระหว่างการทดลอง โหนดใดๆที่ไม่สนับสนุนความสามารถนี้ต้องระบุค่าโฟลเลเบิลเป็นศูนย์เมื่อทำการส่ง แพ็กเก็ตออกไป และไม่ต้องสนใจค่านี้เมื่อรับแพ็กเก็ตมา

แพ็กเก็ตทุกแพ็กเก็ตที่อยู่ในโฟลเดียวกันหมายถึงต้องมีค่าโฟลเลเบิล,แอดเดรสต้นทางและแอดเดรสปลายทางตรงกัน ดังนั้นจึงสามารถมีค่าโฟลเลเบิลได้หลายค่าในเส้นทางหนึ่งๆ ส่วนระยะเวลาสูงสุดของโฟลและการติดตั้งโฟลเลเบิลนั้นจะถูกกำหนดในส่วนรายละเอียดของโพรโตคอลอื่น (เช่น Resource Reservation Protocol)

เมื่อโหนดหยุดการทำงานและเริ่มทำงานใหม่ต้องระวังที่จะไม่ใช้โฟลเลเบิลค่าเดิมที่พึ่งใช้ไปและยังไม่ระยะหมดเวลาของโฟลโดยอาจทำได้โดยการบันทึกค่าโฟล

โฟลเลเบิลอาจใช้โดย ISP เพื่อให้บริการแก่ผู้ใช้ที่ต้องการการบริการแบบพิเศษเช่น ขนาดแบนด์วิทธ์ที่กว้างกว่าและค่าหน่วง (Delay) ที่ต่ำกว่าปกติ แต่ข้อเสียของการใช้โฟลเลเบิล คือต้องเสียเนื้อที่ของหน่วยความจำบางส่วนของเราเตอร์ระหว่างทางเพื่อเก็บค่าโฟลเลเบิล

5.2 ส่วนเพิ่มเติมส่วนหัวของ IPv6 (IPv6 Extension Headers)

ใน IPv6 ข้อมูลเพิ่มเติมต่างๆ ในชั้นอินเทอร์เน็ตแยกออกไปจากส่วนหัวของ IPv6 เพื่อประสิทธิภาพในการส่งข้อมูล และการเพิ่มออปชั่นต่างๆที่อาจเกิดขึ้นใหม่ ส่วนเพิ่มเติมวางอยู่ระหว่างส่วนหัวของ IPv6 และส่วนหัวของโพรโตคอลในชั้นที่สูงกว่า โดยส่วนเพิ่มเติมแต่ละแบบนั้นจะถูกระบุโดยค่า Next Header ดังรูปที่ 5.2

IPv6 Header Next Header = TCP	TCP Header + Data

IPv6 Header Next Header = Routing	Routing Header Next Header = TCP	TCP Header + Data

IPv6 Header Next Header = Routing	Routing Header Next Header = Fragment	Fragment Header Next Header = TCP	Fragment of TCP Header + Data

รูปที่ 5.2 แสดงการใช้ค่า Next Header เพื่อระบุส่วนเพิ่มเติมส่วนหัว

โดยปกติแล้วส่วนเพิ่มเติมส่วนหัวนี้ไม่ประมวลผลหรือตรวจสอบโดยโหนดต่างๆที่แพ็กเก็ตนั้นผ่านจนเมื่อถึงโหนดปลายทางจึงเริ่มประมวลผล และในการประมวลผลต้องทำตามส่วนเพิ่มเติมส่วนหัวที่ปรากฏอยู่ในแพ็กเก็ตอย่างเป็นลำดับตามที่ส่วนเพิ่มเติมส่วนหัวนั้นปรากฏ โหนดปลายทางจะต้องไม่เลือกส่วนเพิ่มเติมส่วนหัวที่ปรากฏอยู่ข้างหลังมาประมวลผลก่อนส่วนเพิ่มเติมส่วนหัวที่อยู่ข้างหน้า

จะมีข้อยกเว้นถ้าแพ็กเก็ตนั้นมีการใช้ฮ็อปบายฮ็อปออปชั่น (Hop-by-Hop Options) ส่วนเพิ่มเติมส่วนหัวในแพ็กเก็ตนั้นจะต้องถูกประมวลผลโดยทุกๆโหนดที่แพ็กเก็ตนั้นผ่านไป โดยส่วนหัวของฮ็อปบายฮ็อปออปชั่นจะต้องอยู่ติดกับส่วนหัวของ IPv6 และค่าของ Next Header ในส่วนหัวของ IPv6 แสดงค่าเป็นศูนย์

ถ้าโหนดที่ทำการประมวลผลแพ็กเก็ตไม่รู้จักค่า Next Header ที่แสดงในแพ็กเก็ต โหนดนั้นก็ควรจะส่งข้อความ ICMP Parameter Problem ไปยังแหล่งที่มาของแพ็กเก็ต ด้วยค่า Code เป็น 1 (“unregonized Next Header type encountered”) และตัวชี้ของไอซีเอ็มพีชี้ไปยังค่าที่มีปัญหา การกระทำนี้ควรจะกระทำด้วยถ้าเกิดค่า Next Header มีค่าเป็นศูนย์ในส่วนหัวใดๆ นอกจากส่วนหัวของ IPv6 รายละเอียดของไอซีเอ็มพีดูรายละเอียดในบท ICMPv6

ในอิมพิวเมนต์ IPv6 แบบสมบูรณ์นั้นจะต้องรู้จักกับส่วนเพิ่มเติมส่วนหัวเหล่านี้

ฮ็อปบายฮ็อปออปชั่น (Hop-by-Hop Options)

เราติง (Routing)

แฟรกเมนต์ (Fragment)

ออปชั่นปลายทาง (Destionation Options)

การพิสูจน์ตัวจริง (Authentication)

การเข้ารหัส (Encapsulating Security Payload)

รายละเอียดของส่วนเพิ่มเติมส่วนหัวแบบการพิสูจน์ตัวจริงและการเข้ารหัสจะกล่าวถึงในบทถัดไป

การเรียงลำดับในส่วนเพิ่มเติมส่วนหัว

เมื่อมีการใช้ส่วนเพิ่มเติมส่วนหัวมากกว่าหนึ่งแบบในแพ็กเก็ตจะต้องมีการเรียงลำดับกัน โดยแนะนำให้มีลำดับดังนี้

IPv6 Header

Hop-by-Hop Options Header

Destination Options Header (1)

Routing Header

Fragment Header

Authetication Header

Encapsulating Security Payload

Destination Options Header (3)

Upper-Layer Protocol

(1) ออปชั่นสำหรับโหนดแรกที่แพ็กเก็ตเดินทางมาถึง และออปชั่นสำหรับโหนดถัดๆไปที่แพ็กเก็ตเดินทางไปถึงที่กำหนดโดยส่วนหัวของการหาเส้นทาง

(2) ออปชั่นสำหรับประมวลผลโดยโหนดปลายทางเท่านั้น

ถ้าส่วนหัวของชั้นที่สูงกว่า (Upper-Layer) เป็นส่วนหัวของ IPv6 อีกอันหนึ่ง มีความเป็นไปได้ว่าส่วนหัวของ IPv6 ที่ตามมานั้นอาจจะตามด้วยส่วนเพิ่มเติมส่วนหัวของมันเองก็ได้

IPv6 โหนดใดๆ ต้องพยายามทำการประมวลผลส่วนเพิ่มเติมส่วนหัวตามลำดับที่มันปรากฏ ยกเว้นส่วนหัวของฮ็อปบายฮ็อปออปชั่น (Hop-by-Hop Options) ที่ต้องตามหลังของส่วนหัวของ IPv6 เท่านั้น

ออปชั่น (Option)

ส่วนหัวของฮ็อปบายฮ็อปออปชั่น และส่วนหัวของออปชั่นปลายทางจะประกอบไปด้วย Type-Length-Value (TLV) รวมกันเป็นออปชั่น โดยมีรูปแบบดังนี้

Option Type	Opt Data Len	Option Data

รูปที่ 5.3 แสดงรูปแบบของออปชั่น

Option Type : ขนาด 8 บิต ใช้บอกชนิดของออปชั่น

Opt Data Len : ขนาด 8 บิต ใช้บอกความยาวของ Option Data

Option Data : เป็นข้อมูลของออปชั่น มีความยาวแบบแปรผันได้

สองบิตบนสุดของ Option Type จะไว้ใช้เพื่อเป็นตัวบอกโหนดที่ทำการประมวลผลออปชั่นนี้ว่าต้องทำอย่างไรถ้าไม่สามารทำการประมวลผลออปชั่นนี้ได้ เช่น ไม่รู้จักออปชั่นนี้ มีค่าดังนี้

00 : ข้ามออปชั่นนี้ไปแล้วทำการประมวลผลต่อไป

01 : ทิ้งแพ็กเก็ตนี้ไป

10 : ทิ้งแพ็กเก็ตนี้ไปโดยไม่ถือว่าแอดเดรสปลายทางจะเป็นแบบมัลติแคสต์ แล้วส่งข้อความแบบ ICMP Parameter Problem และตั้งค่า Code เป็น 2 ไปยังโหนดต้นทาง

11 : ทิ้งแพ็กเก็ตนี้ไปก็ต่อเมื่อแอดเดรสปลายทางไม่ใช่แบบมัลติแคสต์ แล้วส่งข้อความแบบ ICMP Parameter Problem และตั้งค่า Code เป็น 2 ไปยังโหนดต้นทาง

บิตบนสุดบิตที่สามของ Option Type มีไว้เพื่อบอกว่าค่าของ Option Data ในออปชั่นนั้นสามารถเปลี่ยนเส้นทางเพื่อไปสู่เป้าหมายสุดท้ายได้หรือไม่ เมื่อมีส่วนหัวของการพิสูจน์ตัวจริงปรากฏอยู่ในแพ็กเก็ต สำหรับ ออปชั่นใดๆ ที่สามารถเปลี่ยนเส้นทางได้ ข้อมูลทั้งหมดที่อยู่ใน Option Data ต้องถูกจัดการเหมือนว่าข้อมูลเหล่านั้นมีค่าเป็นศูนย์ทั้งหมดเมื่อตรวจสอบความถูกต้องของแพ็กเก็ต ตัวอย่างของออปชั่น เช่น Jumbo Payload Option

แพดดิ้งออปชั่น (Padding Option)

แพดดิ้งออปชั่นมีไว้ใช้เพื่อความต้องการการจัดเรียงออปชั่น (ออปชั่นทุกออปชั่นต้องจัดเรียงให้อยู่ในขอบของแพ็กเก็ต ยกเว้น Pad1 และ PadN ออปชั่น) และเพื่อเพิ่มข้อมูลเข้าไปในส่วนหัวเพื่อให้ส่วนหัวนี้มีค่าเป็นจำนวนเท่าของแปด แพดดิ้งออปชั่นมีอยู่สองแบบ (IPv6 โหนดทุกโหนดต้องสามารถประมวลผล Padding ออปชั่นได้)

0

รูปที่ 5.4 แสดงโครงสร้างของ Pad1 ออปชั่น(Pad1 ออปชั่นเป็นออปชั่นแบบพิเศษจะไม่มีค่า Opt Data Len และค่า Option Data)

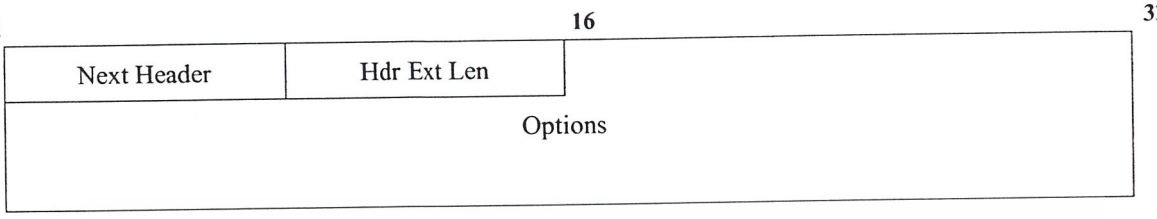
Pad1 ออปชั่นใช้สำหรับแทรกข้อมูลจำนวนแปดบิตเข้าไปในส่วนของออปชั่น ถ้าต้องการแทรกข้อมูลมากกว่าแปดบิตต้องใช้ออปชั่นแบบ PadN

1	Opt Data Len	Option Data

รูปที่ 5.5 แสดงโครงสร้างของ PadN ออปชั่น

5.3 ส่วนหัวของฮ็อปบายฮ็อปออปชั่น (Hop-by-Hop Options Header)

ส่วนหัวของฮ็อปบายฮ็อปออปชั่นมีไว้ใช้เพื่อบรรจุออปชั่นเพิ่มเติมเพื่อให้โหนดทุกๆ โหนดที่แพ็กเก็ตนี้ถูกส่งผ่านตั้งแต่ต้นทางถึงปลายทาง ส่วนหัวของฮ็อปบายฮ็อปออปชั่นจะถูกระบุด้วยค่า Next Header ที่อยู่ในส่วนหัวของ IPv6 ด้วยค่า 0 มีโครงสร้างดังนี้



รูปที่ 5.6 แสดงโครงสร้างส่วนหัวของฮ็อปบายฮ็อปออปชั่น

Next Header : ขนาด 8 บิต ใช้เหมือนค่า Next Header ในส่วนหัวของ IPv6

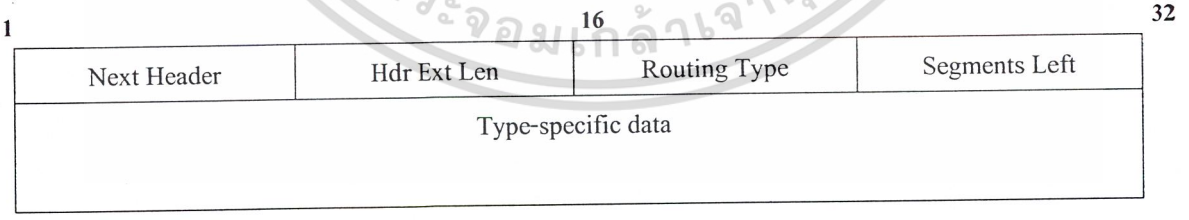
Hdr Ext Len : ขนาด 8 บิต ใช้บอกขนาดส่วนหัวของฮ็อปบายฮ็อปออปชั่น ไม่รวม 8 บิตแรก

Options : ขนาดแปรผันได้ใช้บรรจุออปชั่นแบบต่างๆ ที่ต้องการให้โหนดทุกโหนดทำการประมวลผล

จัมโบเพโหลดออปชั่นสามารถถูกบรรจุอยู่ในค่า Options ในส่วนหัวของฮ็อปบายฮ็อปออปชั่น

5.4 ส่วนหัวของเราติง (Routing Header)

ส่วนหัวของเราติงใช้โดยโหนดต้นทางเพื่อระบุแอดเดรสของโหนดระหว่างทางที่แพ็กเก็ตต้องถูกส่งผ่านก่อนที่จะถึงโหนดปลายทาง การใช้งานจะคล้ายกับการใช้งาน Loose Source and Record Route Option ใน IPv4 ส่วนหัวของการหาเส้นทางจะถูกระบุด้วยค่า Next Header ที่อยู่ในส่วนหัวก่อนหน้า ด้วยค่า 43 และมีโครงสร้างดังนี้



รูปที่ 5.7 แสดงโครงสร้างส่วนหัวของเราติง

Next Header : ขนาด 8 บิต ใช้เหมือนค่า Next Header ในส่วนหัวของ IPv6

Hdr Ext Len : ขนาด 8 บิต ใช้บอกขนาดส่วนหัวของ Routing Options ไม่รวม 8 บิตแรก

Routing Type : ขนาด 8 บิต ใช้บอกรูปแบบที่ใช้ในส่วนหัวนี้ ตัวอย่างจะกล่าวถึงในส่วนต่อไป

Segments Left : ขนาด 8 บิต ใช้บอกจำนวนของโหนดที่ยังเหลืออยู่ (โหนดที่ยังต้องผ่านก่อนไปถึงโหนดปลายทาง)

Type-specific data : ขนาดแปรผันได้ รูปแบบจะถูกกำหนดใน Routing Type

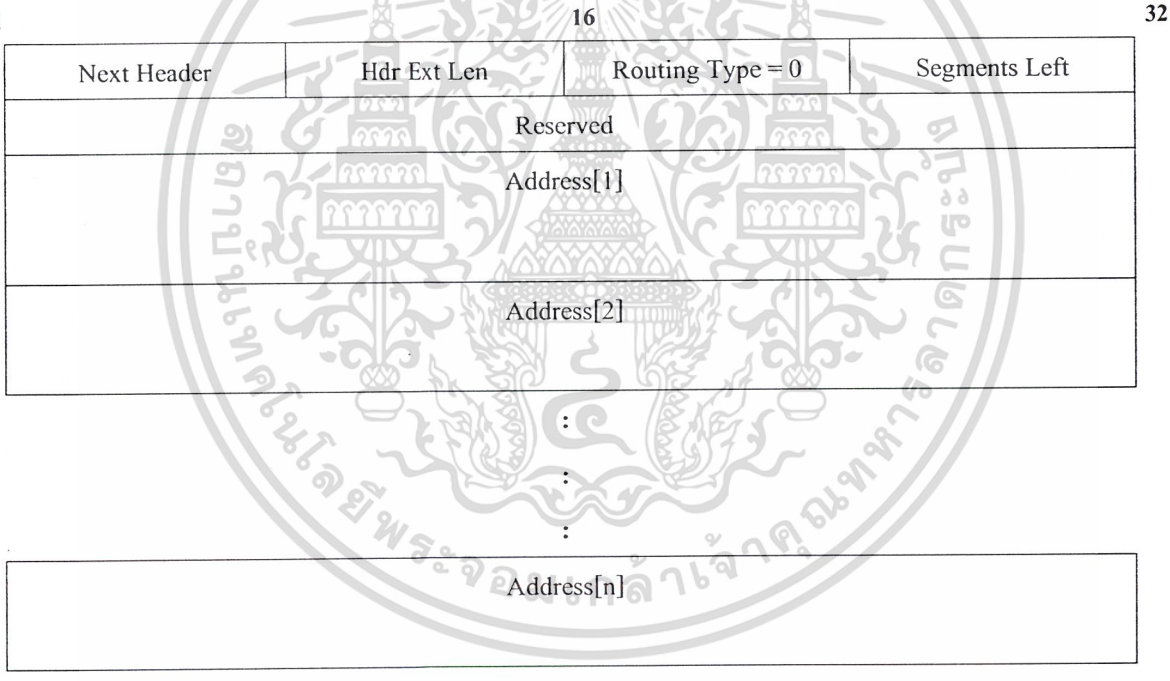
ถ้าในระหว่างการประมวลผลโหนดเจอส่วนหัวของเราติงที่ค่า Routing Type ที่ไม่สามารถประมวลผลได้ โหนดจะต้องทำตามขั้นตอนดังนี้

ถ้าค่า Segments Left มีค่าเป็นศูนย์โหนดต้องทิ้งส่วนหัวของเราติงแล้วประมวลผลส่วนหัวถัดไป

ถ้าค่า Segments Left มีค่าไม่เท่ากับศูนย์โหนดต้องทิ้งแพ็กเก็ตแล้วส่งข้อความ ICMP Parameter Problem และตั้งค่า Code เท่ากับ 0 ไปยังโหนดต้นทาง

ถ้าขณะประมวลผลส่วนหัวของเราติง โหนดระหว่างทางตรวจได้ว่าต้องส่งแพ็กเก็ตต่อไปยังเครือข่ายอื่นที่มีค่า MTU (Maximum Tranfer Unit) น้อยกว่าขนาดของแพ็กเก็ต โหนดต้องทิ้งแพ็กเก็ตนี้แล้วส่งข้อความ ICMP Packet Too Big ไปยังโหนดต้นทาง

ตัวอย่างของการใช้ค่า Routing Type ที่มีค่าเป็นศูนย์



รูปที่ 5.8 แสดงโครงสร้างส่วนหัวของ Routing ที่มีค่า Routing Type เป็นศูนย์

Reserved : ขนาด 32 บิต ใช้สำรองไว้เพื่อการใช้ในอนาคต จะถูกตั้งค่าเป็นศูนย์ทั้งหมดโดยโหนดที่ส่งและโหนดปลายทางไม่ต้องประมวลผลค่านี้

Address[1..n] : ค่าแอดเดรสจาก 1 ถึง n แอดเดรสแบบมัลติแคสต์ต้องไม่ปรากฏในส่วนนี้และต้องไม่ปรากฏในส่วนหัวของ IPv6 (Source และ Destiantion Address) ที่มีการใช้ส่วนเพิ่มเติมเราติงที่มีค่า Routing Type เป็นศูนย์

ส่วนหัวของเราติงจะไม่ถูกประมวลผลจนกว่าจะถึงโหนดปลายทางที่ระบุโดยค่า Destination Address ที่อยู่ในส่วนหัวของ IPv6 เมื่อแพ็กเก็ตเดินทางไปถึงโหนดปลายทาง แพ็กเก็ตที่มีค่า Routing Type เป็นศูนย์จะถูกประมวลผลดังตัวอย่างต่อไปนี้

สมมติว่าต้องการส่งแพ็กเก็ตทางจากโหนดต้นทางคือ S ไปยังโหนดปลายทางคือ D โดยผ่านโหนด I1,I2 และ I3 ตามลำดับ

ขณะแพ็กเก็ตเดินทางจาก S ไป I1

ค่า Source Address = S , Destination Address = I1 (ส่วนหัวของ IPv6)

ค่า Hdr Ext Len = 6 , Segments Left = 3

Address[1] = I2

Address[2] = I3

Address[3] = D (ส่วนหัวของ Routing)

ขณะแพ็กเก็ตเดินทางจาก I1 ไป I2

ค่า Source Address = S , Destination Address = I2 (ส่วนหัวของ IPv6)

ค่า Hdr Ext Len = 6 , Segments Left = 2

Address[1] = I1

Address[2] = I3

Address[3] = D (ส่วนหัวของ Routing)

ขณะแพ็กเก็ตเดินทางจาก I2 ไป I3

ค่า Source Address = S , Destination Address = I3 (ส่วนหัวของ IPv6)

ค่า Hdr Ext Len = 6 , Segments Left = 1

Address[1] = I1

Address[2] = I2

Address[3] = D (ส่วนหัวของ Routing)

ขณะแพ็กเก็ตเดินทางจาก I3 ไป D

ค่า Source Address = S , Destination Address = D (ส่วนหัวของ IPv6)

ค่า Hdr Ext Len = 6 , Segments Left = 0

Address[1] = I1

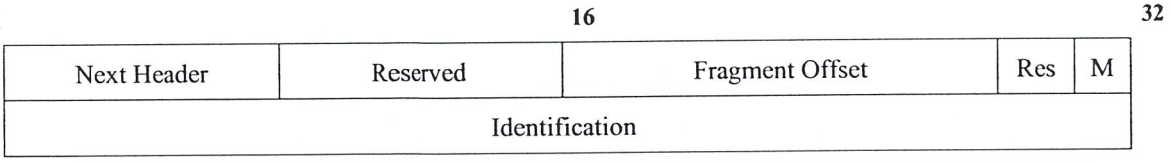
Address[2] = I2

Address[3] = I3 (ส่วนหัวของ Routing)

เมื่อถึงโหนดสุดท้ายแล้วค่า Segments Left จะเป็นศูนย์

5.5 ส่วนหัวของแฟรกเมนต์ (Fragment Header)

ส่วนหัวของแฟรกเมนต์จะใช้เมื่อโหนดต้นทางส่งแพ็กเก็ตที่มีความยาวมากกว่าค่า MTU ไปยังโหนดปลายทาง การแฟรกเมนต์ใน IPv6 นั้นไม่เหมือนกับใน IPv4 เพราะใน IPv6 การแฟรกเมนต์เกิดจากโหนดต้นทางเท่านั้น ส่วนหัวของแฟรกเมนต์จะถูกระบุด้วยค่า Next Header ที่อยู่ในส่วนหัวก่อนหน้าด้วยค่า 44



รูปที่ 5.9 แสดงโครงสร้างส่วนหัวของแฟรกเมนต์

Next Header : ขนาด 8 บิต ใช้เหมือนค่า Next Header ในส่วนหัวของ IPv6

Reserved : ขนาด 8 บิต ใช้สำรองไว้เพื่อการใช้ในอนาคต จะถูกตั้งค่าเป็นศูนย์ทั้งหมดโดยโหนดที่ส่งและโหนดปลายทางไม่ต้องประมวลผลค่านี้

Fragment Offset : ขนาด 13 บิต ใช้เพื่อระบุว่าแพ็กเก็ตนี้เป็นของข้อมูลส่วนใด

Res : ขนาด 2 บิต ใช้สำรองไว้เพื่อใช้ในอนาคต

M : ขนาด 1 บิต ใช้ระบุว่ายังมีแพ็กเก็ตที่ตามหรือไม่หรือเป็นแพ็กเก็ตสุดท้าย

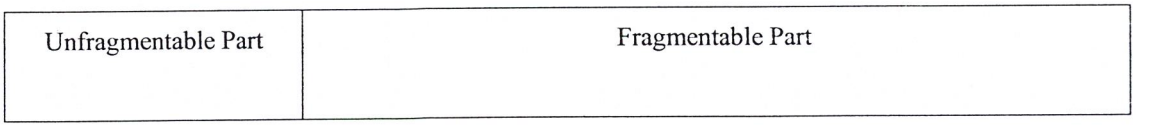
1 = ยังมีแพ็กเก็ตตามมา

0 = เป็นแพ็กเก็ตสุดท้าย

Identification : ขนาด 32 บิต รายละเอียดอยู่จะในส่วนถัดไป

สำหรับแพ็กเก็ตทุกๆ แพ็กเก็ตที่ต้องถูกแฟรกเมนต์ โหนดต้นทางจะกำหนดค่า Identification ค่านี้ต้องแตกต่างกับค่าที่กำหนดให้กับแพ็กเก็ตที่ถูกแฟรกเมนต์ก่อนหน้านี้ที่มีค่า Source Address และค่า Destination Address ตรงกัน (เพื่อให้โหนดปลายทางสามารถระบุได้อย่างถูกต้องว่าแพ็กเก็ตนี้เป็นของข้อมูลใด) ถ้ามีส่วนหัวของเราติงปรากฏอยู่ในแพ็กเก็ตค่า Destination Address ที่ใช้คิดคือค่าแอดเดรสที่อยู่ปลายทางจริงๆ

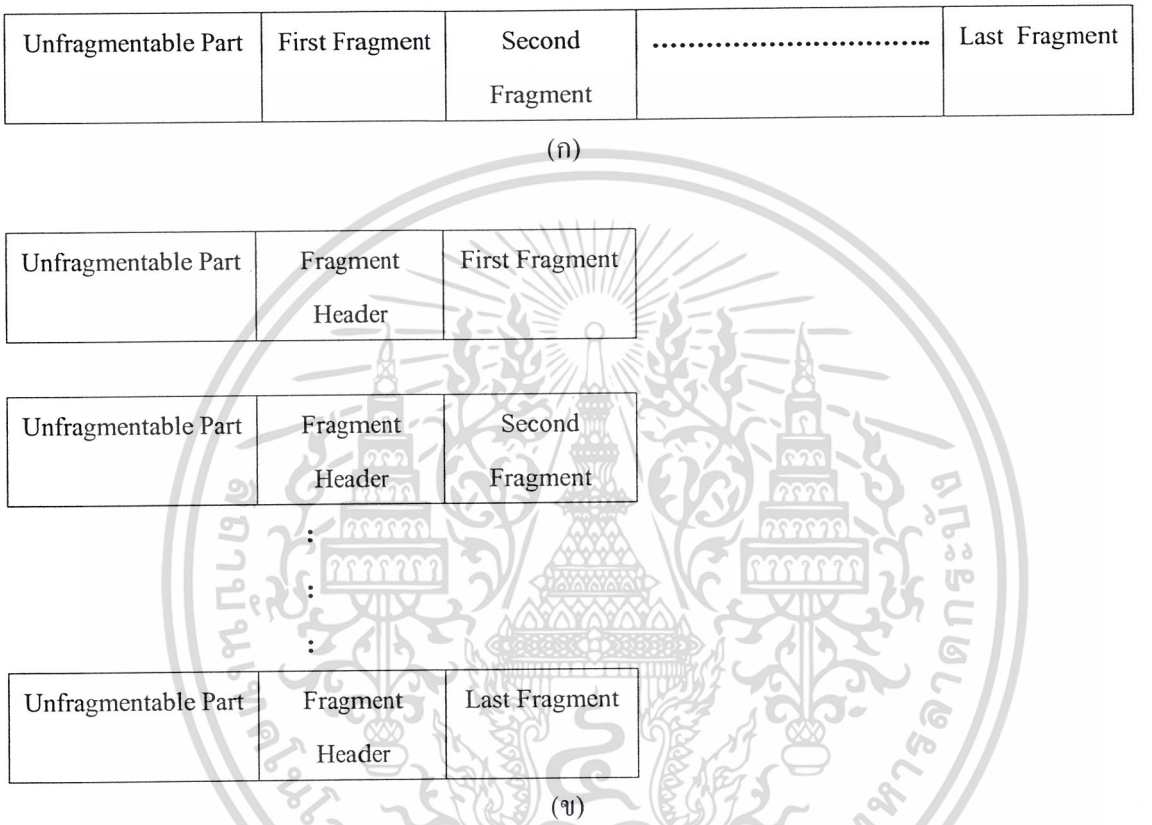
ในแพ็กเก็ตตั้งต้นขนาดใหญ่ที่ยังไม่ได้แฟรกเมนต์จะประกอบไปด้วยสองส่วน ดังนี้



รูปที่ 5.10 แสดงแพ็กเก็ตตั้งต้น

ส่วนที่ไม่สามารถแบ่งย่อยได้ (Unfragmentable Part) ประกอบไปด้วยส่วนหัวของ IPv6 และส่วนเพิ่มเติมส่วนหัวของ IPv6 ที่ต้องถูกประมวลผลโดยโหนดระหว่างทาง นั้นคือตั้งแต่ส่วนหัวของ IPv6 จนถึงส่วนหัวของเราติง (ถ้ามี) รวมทั้งส่วนหัวของฮ็อปบายฮ็อปออปชั่นด้วย

ส่วนที่สามารถแบ่งย่อยได้ (Fragmentable Part) ประกอบไปด้วยส่วนที่เหลือของแพ็กเก็ตดั้งต้นทั้งหมด นั้นคือส่วนเพิ่มเติมส่วนหัวที่จะถูกประมวลผลเมื่อถึงโหนดสุดท้ายและรวมทั้งส่วนข้อมูลของโพรโตคอลในระดับที่สูงกว่า



รูปที่ 5.11 (ก) แสดงการแบ่งแพ็กเก็ตดั้งต้นออกเป็นส่วนย่อยๆ (ข) แสดงการนำส่วนย่อยมาประกอบเป็นแพ็กเก็ตย่อยก่อนส่งออกไปจากโหนด

แต่ละแพ็กเก็ตย่อยประกอบไปด้วยส่วนต่างๆ ดังนี้

- ส่วนที่ไม่สามารถแบ่งย่อยได้จากแพ็กเก็ตดั้งต้นที่มีการคำนวณค่า Payload Length ใหม่และเปลี่ยนค่า Next Header ที่อยู่ในส่วนหัวหลังสุดแสดงค่าเป็น 44 (แสดงว่าส่วนหัวถัดไปเป็นส่วนหัวของแฟรกเมนต์)
- ส่วนหัวของแฟรกเมนต์ ประกอบไปด้วย ค่า Next Header ที่ระบุถึงส่วนหัวแรกสุดที่อยู่ในส่วนที่แบ่งย่อยได้ในแพ็กเก็ตดั้งต้น และค่าอื่นๆ ที่ได้กล่าวมาแล้ว

- ส่วนที่ถูกแบ่งย่อย ค่าความยาวของส่วนที่ถูกแบ่งย่อยต้องเลือกให้เหมาะสมเพื่อให้ขนาดความยาวแพ็กเก็ตมีค่าเหมาะสมกับค่า MTU

เมื่อแพ็กเก็ตย่อยไปถึงโหนดปลายทางแพ็กเก็ตจะถูกรวมกันเข้าอีกครั้ง (แพ็กเก็ตจะถูกรวมเมื่อมีค่า Source Address ค่า Destination Address และค่า Identification เหมือนกัน) ด้วยกฎดังนี้

ส่วนที่ไม่สามารถแบ่งย่อยได้จากแพ็กเก็ตย่อยแพ็กเก็ตแรก (ไม่รวมส่วนหัวของแฟรกเมนต์) โดยค่า Next Header ลำดับสุดท้ายที่อยู่ในส่วนแบ่งย่อยไม่ได้จะถูกเปลี่ยนโดยรับมาจากค่า Next Header ที่อยู่ในส่วนหัวของแฟรกเมนต์แรกสุด ส่วนค่า Payload Length สามารถคำนวณได้จากสูตรดังนี้

PL.orig = PL.first - FL.first - 8 + (8 * FO.last) + FL.last

โดยที่

PL.orig = ค่า Payload Length ของแพ็กเก็ตดั้งต้น

PL.first = ค่า Payload Length ของแพ็กเก็ตที่ถูกแบ่งย่อยแพ็กเก็ตแรก

FL.first = ความยาวของส่วนย่อยที่ตามมาหลังส่วนหัวของแฟรกเมนต์ของแพ็กเก็ตแรก

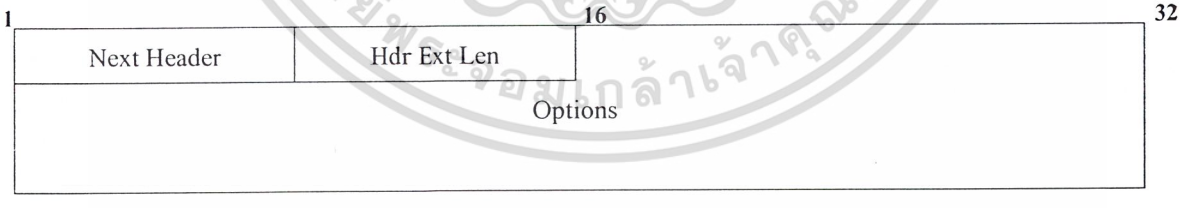
FO.last = ค่า Fragment Offset ที่อยู่ในส่วนหัวของแฟรกเมนต์ของแพ็กเก็ตสุดท้าย

FL.last = ความยาวของส่วนย่อยที่ตามมาหลังส่วนหัวของแฟรกเมนต์ของแพ็กเก็ตสุดท้าย

ส่วนหัวของแฟรกเมนต์จะไม่ปรากฏในแพ็กเก็ตที่สุดท้ายถูกรวมเข้าด้วยกัน

5.6 ส่วนหัวของออปชั่นปลายทาง (Destination Options Header)

ส่วนหัวของออปชั่นปลายทางใช้เมื่อมีข้อมูลที่จำเป็นต้องถูกตรวจสอบโดยโหนดปลายทาง (หรือโหนดระหว่างทางที่ถูกกำหนดโดยส่วนหัวของเราติง) ส่วนหัวของออปชั่นปลายทางจะถูกระบุด้วยค่า Next Header ที่อยู่ในส่วนหัวก่อนหน้า ด้วยค่า 60 โครงสร้างจะแสดงดังรูป (มีรูปแบบเหมือนโครงสร้างส่วนหัวของฮ็อปบาย ฮ็อปออปชั่น)



รูปที่ 5.12 แสดงโครงสร้างส่วนหัวของออปชั่นปลายทาง

Next Header : ขนาด 8 บิต ใช้เหมือนค่า Next Header ในส่วนหัวของ IPv6

Hdr Ext Len : ขนาด 8 บิต ใช้บอกขนาดส่วนหัวของออปชั่นปลายทางไม่รวม 8 บิตแรก

Options : ขนาดแปรผันได้ใช้บรรจุออปชั่นแบบต่างๆ ที่ต้องการให้โหนดทุกโหนดประมวลผล

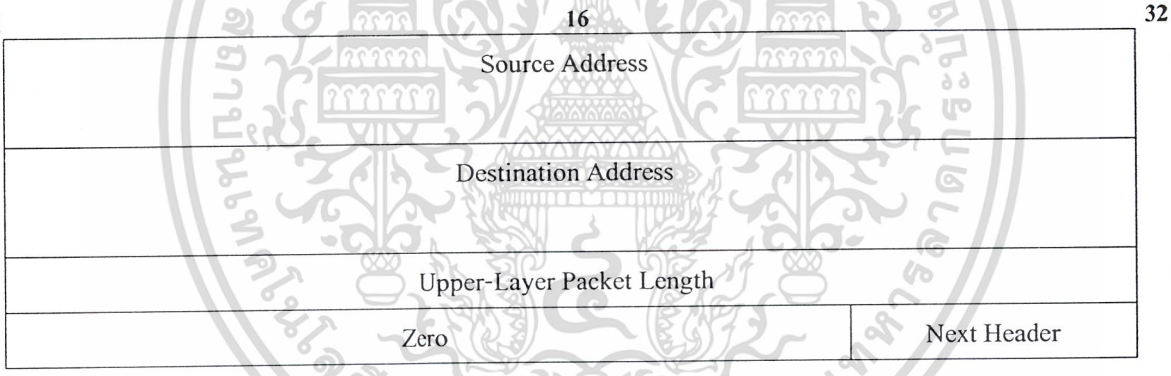
ขนาดของแพ็กเก็ต

IPv6 ระบุว่าในทุกเครือข่ายในอินเตอร์เน็ตมีค่า MTU คือ 1280 ไบต์หรือมากกว่า สำหรับเครือข่ายที่ไม่สามารถส่งข้อมูลขนาด 1280 ไบต์ในหนึ่งแพ็กเก็ต การแฟรกเมนต์ต้องถูกแบ่งในชั้นที่ต่ำกว่า IPv6 (ในชั้น Link Layer) แต่เพื่อการทำให้เหมาะเมื่อมีการใช้เอ็นแคปซูเลชั่น (Encapsulaiton) ค่า MTU ควรจะถูกตั้งไว้เป็น 1500 ไบต์ ในการส่งแพ็กเก็ตที่มีขนาดมากกว่าค่า MTU โหนดนั้นต้องแบ่งแพ็กเก็ต ออกเป็นแพ็กเก็ตย่อย และโหนดต้องสามารถรับแพ็กเก็ตที่ถูกแฟรกเมนต์ที่รวมกันแล้วมีขนาดมากกว่า 1500 ไบต์ได้

IPv6 โหนดที่ส่งแพ็กเก็ตไปหา IPv4 โหนดอาจได้รับข้อความ ICMP Packet Too Big เมื่อค่า MTU ใน ฮ็อปต่อไปมีค่าต่ำกว่า 1280 ในกรณีนี้ IPv6 โหนดไม่จำเป็นต้องลดขนาดของแพ็กเก็ตลงในการส่งครั้งถัดไปแต่ให้เพิ่มส่วนหัวของแฟรกเมนต์เข้าไปแทน

โพรโตคอลในชั้นที่สูงกว่า (Upper-Layer Protocol)

โพรโตคอลในชั้นที่สูงกว่าชั้นอินเตอร์เน็ต ต้องปรับปรุงเช็คซัมโดยต้องมีการใช้ส่วนหัวเทียม (Pseudo Header) ที่รวมแอดเดรสขนาด 128 บิตของ IPv6 แทนที่แอดเดรสขนาด 32 บิตของ IPv4 โดยส่วนหัวเทียมแสดงได้ดังรูป



รูปที่ 5.13 แสดงส่วนหัวเทียม

ถ้าแพ็กเก็ตมีส่วนหัวของเราติงปรากฏอยู่ ค่า Destination Address คือค่าแอดเดรสสุดท้ายที่แพ็กเก็ตนั้นเดินทางไปถึง

Next Header ของส่วนหัวเทียมระบุถึงค่าของโพรโตคอลในชั้นที่สูงกว่า (เช่น 6 สำหรับ TCP, 17 สำหรับ UDP)

Upper-Layer Packet Length คือขนาดความยาวของส่วนหัวของโพรโตคอลในชั้นที่สูงกว่าและข้อมูลของโพรโตคอลในชั้นที่สูงกว่า สำหรับบางโพรโตคอลจะข้อมูลความยาวอยู่แล้ว เช่น ยูดีพี จะมีค่า Length ในส่วนหัวของมันเอง ดังนั้นค่า Upper-Layer Packet Length คือค่าข้อมูลนั้นเอง แต่สำหรับบางโพรโตคอลที่ไม่ ปรากฏข้อมูลส่วนนี้ ค่า Upper-Layer Packet Length จะเป็นค่า Payload Length ของ

IPv6 ลบกับขนาดของส่วนเพิ่มเติมส่วนหัวที่ปรากฏอยู่ระหว่างส่วนหัวของ IPv6 และโพรโตคอลในชั้นที่สูงกว่า

สำหรับ ICMPv6 จะใช้ค่าส่วนหัวเทียมข้างต้นลงไปในการคำนวณเช็คซัมด้วย จะแตกต่างกับไอซีเอ็มพีของ IPv4 ที่ไม่ได้รวมส่วนหัวเทียมลงไปในการคำนวณเช็คซัมการเปลี่ยนแปลงนี้เนื่องมาจากเหตุผลเพื่อป้องกันไอซีเอ็มพีจากการส่งผิดเนื่องจากการผิดพลาดของข้อมูลในส่วนหัวของ IPv6 (ในส่วนหัวของ IPv4 มีค่าเช็คซัมอยู่แล้ว)

บทที่ 6

ส่วนหัวของการพิสูจน์ตัวจริงและการเข้ารหัส

(Authentication Header and Encapsulating Security Payload Header)

6.1 ส่วนหัวของการพิสูจน์ตัวจริง (Authentication Header - AH)

AH ให้การสนับสนุน Data Integrity และการพิสูจน์ตัวจริงของไอพีแพ็กเก็ต คุณสมบัติของ Data Integrity คือการป้องกันการแก้ไขข้อมูลในระหว่างการส่งแพ็กเก็ต คุณสมบัติของการพิสูจน์ตัวจริงคือความสามารถให้ผู้ใช้หรือโหนดปลายทางสามารถตรวจสอบผู้ส่งได้ AH ยังสนับสนุนการป้องกันการจู่โจมแบบการปลอมแอดเดรส (Address Spoofing)

AH ให้การป้องกันข้อมูลในส่วนหัวของ IP และข้อมูลในชั้นที่สูงกว่า แต่ค่าต่างๆในส่วนหัวของไอพีที่มีการเปลี่ยนแปลงค่าในระหว่างการส่งข้อมูล AH ไม่สามารถให้การป้องกัน และ AH ไม่ได้ให้การป้องกันจากการดักจับเพื่ออ่านข้อมูลเพราะไม่ได้ทำการเข้ารหัสไว้

AH ใช้ข้อมูลทั้งหมดของแพ็กเก็ตมาใช้ในการคำนวณเพื่อตรวจสอบแพ็กเก็ต ยกเว้นส่วนที่มีการเปลี่ยนแปลงระหว่างการส่งข้อมูล เช่น Hop Limit

วิธีที่ใช้ใน AH คือ Hashing for Message Authentication Codes (HMAC) กับ MD5 หรือ SHA-1

โครงสร้างของ AH มีดังรูป

1	16	32
Next Header	Payload Length	Reserved
Security Parameters Index		
Sequence Number Field		
Authentication Data		

รูปที่ 6.1 แสดงโครงสร้างของ AH

Security Parameters Index (SPI) : ขนาด 32 บิต ใช้บอก Security Association (SA) ของแพ็กเก็ตนี้ ถ้าไม่มีการใช้ SA ค่านี้ต้องตั้งเป็นศูนย์ทั้งหมด

Security Number Field : ขนาด 32 บิต ใช้สำหรับป้องกันรีเพลย์แอ็คแท็ก (Replay-Attack) สำหรับ SA ที่ระบุ ผู้ส่งและผู้รับจะกำหนดเป็นศูนย์เมื่อเริ่มกำหนด SA และแพ็กเก็ตแรกที่ส่งโดย SA นั้นจะมีค่าเป็นหนึ่ง โดยจะเพิ่มค่าขึ้นเรื่อยๆ ทีละหนึ่ง

Authentication Data : มีขนาดแบบแปรผันได้ ประกอบไปด้วยข้อมูลที่ไว้สำหรับตรวจสอบความถูกต้องของแพ็กเก็ตนี้

ทรานสปอรต์และทันเนิลโหมดของ AH (AH Transport and Tunnel Mode)

สำหรับทรานสปอรต์ AH จะถูกจัดการแบบปลายสู่ปลาย (end-to-end) นั้นคือจะไม่ถูกตรวจสอบโดยโหนดระหว่างทาง ดังนั้น AH จะปรากฏหลังจากฮ็อปบายฮ็อปออปชั่น , ส่วนหัวของเราติง , ส่วนหัวของแฟรกเมนต์ และจะปรากฏก่อนส่วนหัวของออปชั่นปลายทางดังรูป

IPv6 Header	Hop-by-Hop/Routing/Fragment Header	AH	Destination Header	Data

รูปที่ 6.2 แสดงทรานสปอรต์โหมดสำหรับ AH

สำหรับทันเนิลโหมดแพ็กเก็ตทั้งหมดจะถูกครอบโดย IPv6 Header อีกชั้น และ AH จะปรากฏระหว่าง IPv6 Header ชั้นในและ IPv6 Header ชั้นนอก โดย IPv6 Header ชั้นในจะบรรจุ Source และ Destination Address ระหว่างโหนดที่ส่งและรับจริง ส่วนหัวของ IPv6 ชั้นนอกจะบรรจุ Source และ Destination Address อาจจะบรรจุแอดเดรสที่แตกต่างกันออกไปก็ได้ เช่นแอดเดรสของ Firewall ดังรูป

New IPv6 Header	Extenstion Headers	AH	Original IPv6 Header	Extenstion Headers	Data

รูปที่ 6.3 แสดง Tunnel Mode สำหรับ AH

สำหรับทันเนิลโหมดไอพี แพ็กเก็ตดั้งเดิมทั้งหมดจะถูกป้องกันโดย AH และไอพีแพ็กเก็ตชั้นนอกจะถูกป้องกันโดย AH ในบางส่วนเท่านั้น

การคำนวณของข้อมูลของการพิสูจน์ตัวจริง (Authentication Data)

โพรโตคอล AH ใช้ข้อมูลของส่วนหัวในการคำนวณเพื่อหา Authentication Data โดยลำดับในการคำนวณมีดังนี้

- SA ถูกเลือกตาม Security Policy
- ข้อมูลต่างๆที่เปลี่ยนระหว่างการส่งข้อมูลคิดเป็นศูนย์ทั้งหมด รวมทั้งค่า Authentication Data
- ค่า Authentication Data ถูกคำนวณตามวิธีการถูกระบุก่อนการติดต่อ (MD5 , SHA-1)
- ค่า Authentication Data ถูกเพิ่มเข้าไปในข้อความ

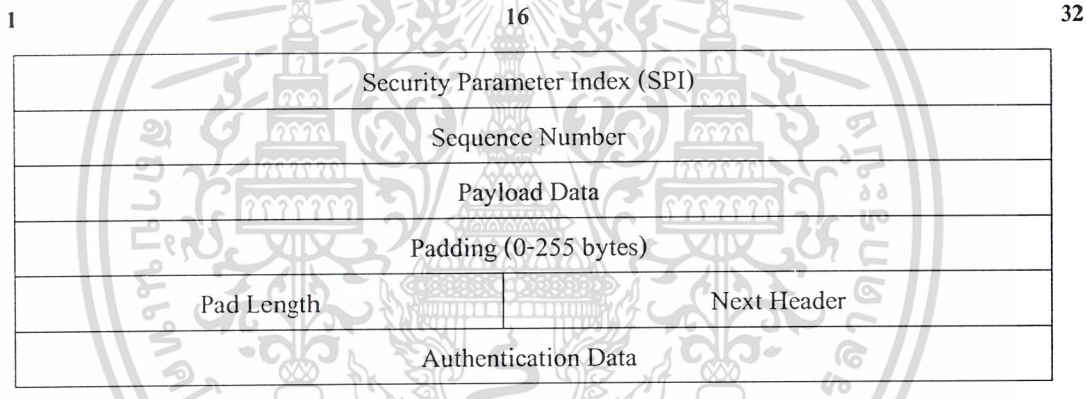
6.2 ส่วนหัวของการเข้ารหัส (Encapsulating Security Payload - ESP)

ESP จะให้ความปลอดภัยทางด้าน Data Integrity และการเป็นความลับ และยังสามารถให้การสนับสนุนการพิสูจน์ตัวจริงได้โดยขึ้นอยู่กับวิธีที่เลือกใช้

โพรโตคอล ESP ต้องการอย่างน้อยที่สุดคือทุกๆ อิมพิวเมนต์ต้องให้การสนับสนุนการใช้ DES ใน Cipher Block Chaining (CBC) Mode เพื่อความสามารถในการติดต่อร่วมกันได้ทั่วทั้งอินเทอร์เน็ต และยังมีอีกหลายวิธีที่สนันสนุนโดยขึ้นอยู่กับอิมพิวเมนต์ดังนี้

- Three-key triple DES
- RC5
- IDEA
- Three-key triple IDEA
- CAST
- Blowfish

โครงสร้างของ ESP มีดังรูป



รูปที่ 6.4 แสดงโครงสร้างของ ESP

Security Parameters Index : ขนาด 32 บิต ใช้บอก Security Asscociation (SA) ของแพ็กเก็ตนี้ ถ้าไม่มีการใช้ SA ค่านี้ต้องตั้งเป็นศูนย์ทั้งหมด

Security Number Field : ขนาด 32 บิต ใช้สำหรับป้องรีเพลย์แอ็คแท็กสำหรับ SA ที่ระบุ ผู้ส่งและผู้รับจะกำหนดเป็นศูนย์เมื่อเริ่มกำหนด SA แพ็กเก็ตแรกที่ส่งโดย SA นั้นจะมีค่าเป็นหนึ่งโดยจะเพิ่มค่าขึ้นเรื่อยๆ ทีละหนึ่ง

Payload Data : มีขนาดแบบแปรผันได้ คือข้อมูลหรือ IPv6 แพ็กเก็ตที่ถูกป้องกันโดยการเข้ารหัส

Padding : มีขนาดตั้งแต่ 0-255 ไบต์ ใช้สำหรับเติมข้อมูลใน Payload Data ให้เท่ากับจำนวนเท่าของขนาดของบล็อกที่ต้องการ โดยวิธีการเข้ารหัสหรือใช้สำหรับการจัดเรียงให้ข้อมูลเรียกกันในแบบ 32 บิต

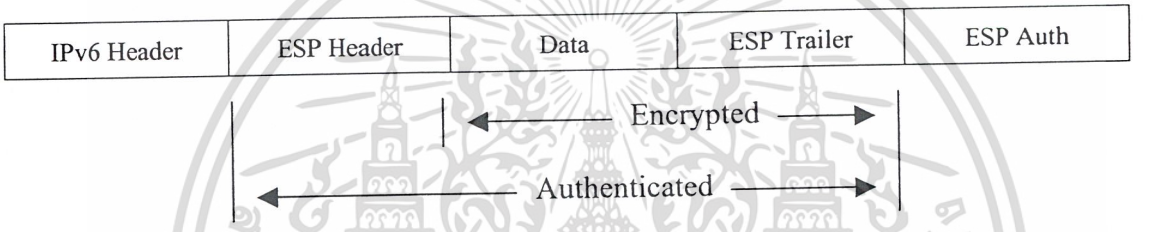
Pad Length : ใช้บอกขนาดของแพดดิ้ง

Next Header : ใช้ระบุข้อมูลที่บรรจุอยู่ใน Payload Data โดยระบุส่วนหัวแรกใน Payload Data

Authentication Data : มีขนาดแบบแปรผันได้ใช้บรรจุข้อมูลที่ใช้สำหรับการคำนวณของ ESP

ทรานสปอรต์และทันเนิลโหมดของ ESP (ESP Transport and Tunnel Modes)

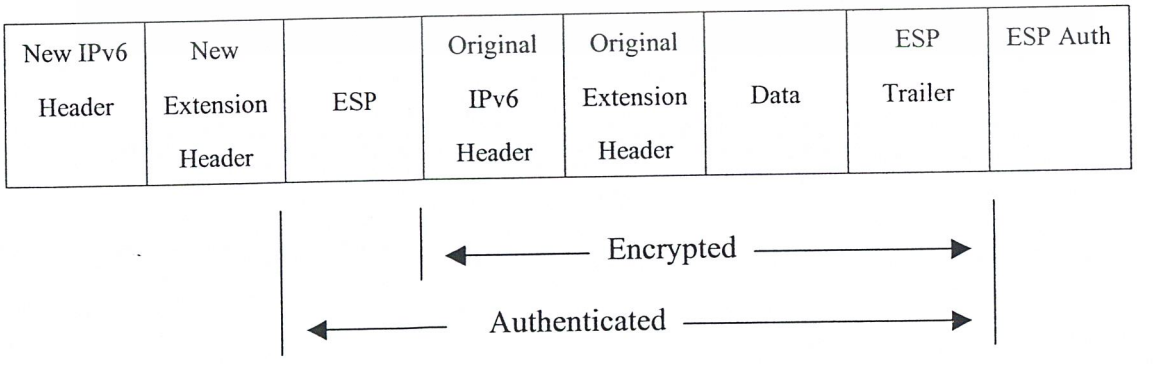
สำหรับทันเนิลโหมด ESP ให้การป้องกันสำหรับข้อมูลในโพรโตคอลที่อยู่ชั้นสูงกว่า (Upper-Layer Protocol) แต่ไม่ให้การป้องกันข้อมูลในส่วนหัวของ IPv6 เฉพาะ host-to-host Implementation สามารถใช้ ทรานสปอรต์โหมดในโหมดนี้ ESP จะ ปรากฏหลังจากส่วนหัวของ IPv6 และก่อนโพรโตคอลที่อยู่ชั้นสูงกว่า หรือก่อนส่วนเพิ่มเติมส่วนอื่น ดังรูป



รูปที่ 6.5 แสดงทรานสปอรต์โหมดของ ESP

สำหรับทันเนิลโหมด ใช้สำหรับการเข้ารหัสแพ็กเก็ตเดิมทั้งหมด วิธีนี้สามารถป้องกันการตรวจสอบแพ็กเก็ตระหว่างการส่งแพ็กเก็ตได้ (Traffic Analysis)

ในขณะที่ทรานสปอรต์โหมดเหมาะสำหรับการป้องกันระหว่างโฮสต์ที่สนันสนุน ESP แต่ทันเนิลโหมด มีประโยชน์สำหรับการป้องกันข้อมูลระหว่างเครือข่ายโดยจะติดตั้งวิธีนี้ระหว่าง Securitty Gateway ทำให้ภาระของโฮสต์หรือเครือข่ายภายในในการประมวลผลการเข้ารหัสลดลง และยังทำให้มีจำนวนคีย์ (Key) ลดลง รูปแบบของทันเนิลโหมดแสดงได้ดังรูป



รูปที่ 6.6 แสดง Tunnel Mode ของ ESP

พิจารณาจากกรณีที่ว่าโฮสต์ภายนอกต้องการติดต่อกับโฮสต์ภายในที่ถูกป้องกันโดย Firewall จะมีขั้นตอนดังนี้

1. สร้างแพ็กเก็ตที่มี Destination Address เป็นค่าแอดเดรสของโฮสต์ภายใน เข้ารหัสข้อมูลตามวิธีที่กำหนดไว้ เพิ่มส่วนหัวของ ESP แล้วสร้างส่วนหัวของ IPv6 อีกอันหนึ่งที่มี Destination Address เป็นแอดเดรสของ Firewall
2. แพ็กเก็ตใหม่ที่ได้มาถูกส่งไปยัง Firewall โดยเราเตอร์ระหว่างทางจะตรวจสอบเฉพาะส่วนหัวของ IPv6 และส่วนเพิ่มเติมส่วนหัวอันใหม่เท่านั้น
3. Firewall ทำการตรวจสอบข้อมูลที่ได้มา และถอดรหัสตามวิธีที่กำหนด แล้วส่งแพ็กเก็ตที่ถูกถอดรหัสแล้วส่งไปยังโฮสต์ภายใน

บทที่ 7

ไอซีเอ็มพีสำหรับ IPv6

(Internet Control Message Protocol for IPv6 - ICMPv6)

IPv6 ได้นำไอซีเอ็มพีสำหรับ IPv4 ตามที่ได้กำหนดไว้ใน RFC 792 มาทำการปรับปรุงเปลี่ยนแปลงในหลายๆส่วนโดยเรียกว่า ไอซีเอ็มพีสำหรับ IPv6 ในบทนี้จะทำการกล่าวถึงรูปแบบของไอซีเอ็มพีสำหรับ IPv6 โดยเนื้อหาได้อ้างอิงจาก RFC 2463

ไอซีเอ็มพีสำหรับ IPv6 ถูกใช้โดยโหนด IPv6 เพื่อรายงานความผิดพลาดที่เกิดขึ้นในการส่งแพ็กเก็ตและเพื่อช่วยทำงานอื่นๆในระดับชั้น Internet Layer ตัวอย่างเช่น ping ไอซีเอ็มพีสำหรับ IPv6 ถือเป็นส่วนส่วนหนึ่งของ IPv6 และจำเป็นอย่างยิ่งที่จะต้อง Implement ในทุกๆ IPv6 โหนด

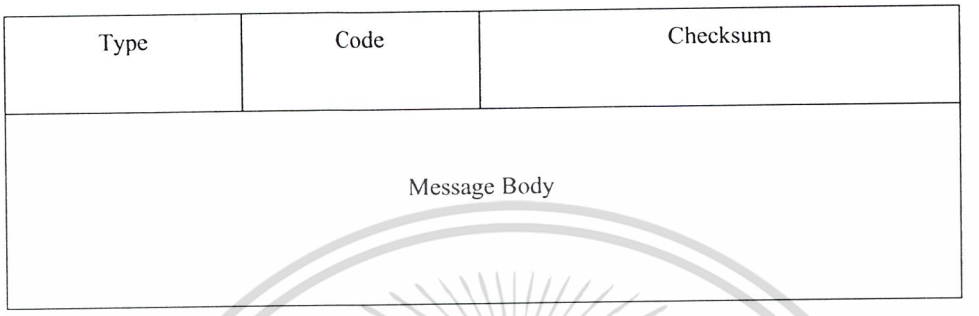
7.1 รูปแบบโดยทั่วไป (Message General Format)

ข้อความของไอซีเอ็มพีสำหรับ IPv6 ถูกแบ่งออกเป็น 2 กลุ่มคือ ข้อความที่รายงานความผิดพลาด(Error Message) และข้อความที่บ่งบอกข้อมูลข่าวสาร(Information Message) โดยส่วน Type ของ Error Message จะมีค่าบิตที่มีนัยสำคัญสูงสุดเป็น 0 เพราะฉะนั้นค่า Type ของ Error Message จึงมีค่าตั้งแต่ 0 – 127 และ Information Message จะมีค่าตั้งแต่ 128 – 255 ตัวอย่างค่า Type เป็นดังตารางที่ 7.1

ICMPv6 Error Messages	
1	Destination Unreachable
2	Packet Too Big
3	Time Exceeded
4	Parameter Problem
ICMPv6 Information Messages	
128	Echo Request
129	Echo Reply
130	Group Membership Query
131	Group Membership Report
132	Group Membership Termination
133	Router Solicitation
134	Router Advertisement
135	Neighbor Solicitation
136	Neighbor Advertisement

ตารางที่ 7.1 ตารางแสดงตัวอย่างค่า Type ของ ไอซีเอ็มพีสำหรับ IPv6

ไอซีเอ็มพีสำหรับ IPv6 จะอยู่ต่อจากส่วนหัวของ IPv6 โดยอาจจะมีส่วนเพิ่มเติมส่วนหัวอื่นๆด้วยหรือไม่ก็ได้ โดยค่า Next Header ที่บ่งบอกว่าส่วนต่อไปเป็น ไอซีเอ็มพีสำหรับ IPv6 คือ 58 (ต่างจากค่าของ ICMP for IPv4 ซึ่งมีค่าเป็น 1) ข้อความต่างๆของ ไอซีเอ็มพีสำหรับ IPv6 จะมีรูปแบบหลักดังรูปที่ 7.1



รูปที่ 7.1 รูปแบบหลักของ ไอซีเอ็มพีสำหรับ IPv6

- ค่าในส่วน Type จะบอกชนิดของข้อความและบอกรูปแบบของข้อมูลในส่วน Message Body ด้วย
- ค่าในส่วน Code จะขึ้นกับค่า Type ใช้เป็นตัวบอกข่าวสารในระดับที่ละเอียดขึ้น
- ค่า Checksum ใช้ตรวจสอบความผิดพลาดที่อาจเกิดขึ้นในการส่ง

โหนดที่ส่งพอซีเอ็มพีสำหรับ IPv6 จะต้องกำหนดทั้ง Source และ Destination Address ในส่วนหัวของ IPv6 ก่อนที่จะทำการคำนวณ Checksum ถ้าโหนดนั้นๆมียูนิแคสต์แอ็ดเดรสมากกว่าหนึ่ง มีข้อกำหนดในการเลือก Source Address ดังนี้

1) ถ้าข้อความไอซีเอ็มพีสำหรับ IPv6 ที่ส่งกลับไปเป็นการตอบข้อความที่ส่งมาให้โหนดนั้น โดยที่ข้อความที่ส่งมาให้นั้นมี Destination Address เป็นยูนิแคสต์แอ็ดเดรสอันดับหนึ่งของโหนดนั้นแล้ว Source Address จะต้องเป็น Destination Address จากข้อความที่ส่งมาที่โหนดนั้น
2) ถ้าข้อความไอซีเอ็มพีสำหรับ IPv6 ที่ส่งกลับไปเป็นการตอบข้อความที่ส่งมาให้โหนดนั้น โดยที่ข้อความที่ส่งมาให้นั้นมี Destination Address เป็นมัลติแคสต์แอ็ดเดรส หรือเอนี่แคสต์แอ็ดเดรสกลุ่มที่โหนดนั้นเป็นสมาชิกอยู่แล้ว Source Address จะต้องเป็นยูนิแคสต์แอ็ดเดรสจากส่วนเชื่อมต่ออันที่ได้รับข้อความที่ส่งมา
3) ถ้าข้อความ ไอซีเอ็มพีสำหรับ IPv6 ที่ส่งกลับไปเป็นการตอบข้อความที่ไม่ได้มี Destination Address อยู่ที่โหนดนั้นแล้ว Source Address จะต้องเป็นยูนิแคสต์แอ็ดเด

รส ของโหนดอันที่จะมีส่วนช่วยในการหาข้อผิดพลาดให้มากที่สุด ตัวอย่างเช่น ถ้าไอซีเอ็มพีสำหรับ IPv6 ที่จะส่งออกไปเป็นการตอบแพ็กเก็ตที่ต้องการให้โหนดนั้นส่งต่อแต่ไม่สามารถส่งต่อได้ Source Address ควรจะเป็นยูนิแคสต์แอ็ดเดรสของส่วนเชื่อมต่อที่อันที่ทำให้การส่งต่อล้มเหลว

4) หากเป็นนอกเหนือจากนี้ Source Address จะต้องเป็นยูนิแคสต์แอ็ดเดรสของส่วนเชื่อมต่อที่จะทำหน้าที่ส่งไอซีเอ็มพีสำหรับ IPv6 กลับไป โดยจะหาได้จากตารางการหาเส้นทาง

ในส่วน Checksum จะมีขนาด 16 บิตแบบ One's Complement ของผลรวมของข้อความไอซีเอ็มพีสำหรับ IPv6 โดยเริ่มตั้งแต่ส่วน Type และยังรวมไปถึงส่วนหัวเทียม (Pseudo-header) ตามที่กำหนดไว้ใน RFC 1883 โดยส่วน Next Header ในส่วนหัวเทียมมีค่าเป็น 58 ซึ่งต่างจากการคำนวณ Checksum ในไอซีเอ็มพีสำหรับ IPv4

ข้อกำหนดที่สำคัญอย่างยิ่งในการอิมพิวเมนต์ IPv6 มีดังต่อไปนี้

1) ถ้าได้รับ ICMPv6 Error message ที่ไม่รู้จักจะต้องส่งต่อให้กับโพรโตคอลในระดับที่สูงกว่า

2) ถ้าได้รับ ICMPv6 Information message ที่ไม่รู้จักจะต้องทิ้งข้อความนั้นโดยไม่ต้องส่ง ไอซีเอ็มพีสำหรับ IPv6 อีก

3) ทุกๆ ICMPv6 Error message จะต้องถูกจำกัดขนาดไม่ให้เกินขนาด MTU ที่น้อยที่สุดที่ IPv6 กำหนดไว้

4) ในกรณีที่โพรโตคอลในระดับชั้นอินเทอร์เน็ต ต้องการให้ส่งต่อ ICMPv6 Error message ให้กับโพรโตคอลในระดับชั้นที่สูงกว่า ชนิดของโพรโตคอลในระดับชั้นที่สูงกว่าจะรู้ได้จากส่วน Message Body ถ้าแพ็กเก็ตIPv6ที่ได้รับมีจำนวนส่วนเพิ่มเติมส่วนหัวมากอย่างผิดปกติแล้ว มีความเป็นไปได้ที่ชนิดของโพรโตคอลในระดับชั้นที่สูงกว่าอาจจะไม่อยู่ในส่วน ไอซีเอ็มพีสำหรับ IPv6 เนื่องจากมีขนาดเกิน MTU ในกรณีนี้ส่วน Error message จะถูกตัดทิ้งหลังจากที่ทำการประมวลผลชั้นไอพีแล้ว

5) ICMPv6 Error message จะต้องไม่ถูกส่งออกไปเพราะผลจากการได้รับ

5.1) ICMPv6 Error message

5.2) แพ็กเก็ตที่มี Destination address เป็นมัลติแคสต์แอ็ดเดรสแต่มีข้อยกเว้นเมื่อ เป็น Packet Too Big Message หรือ Parameter Problem Message ที่มี Code เป็น 2

5.3) แพ็กเก็ตเป็นแบบ Link-Layer Multicast หรือ Link-Layer Broadcast โดยมีข้อยกเว้นเหมือน 5.2

5.4) แพ็กเก็ตที่มี Source address ไม่ปกติเช่น Unspecified address , Multicast address หรือ Anycast address ที่โหนดนั้นรู้จัก

6) โหนดที่ส่งไอซีเอ็มพีสำหรับ IPv6 จะต้องจำกัด ICMPv6 Error message ที่ส่งเพื่อไม่ให้แบนวิดท์สูญเปล่า โดยอาจจะใช้วิธีจำกัดจำนวนไอซีเอ็มพีสำหรับ IPv6 ที่จะส่งในช่วงเวลาหนึ่งๆ หรือใช้วิธีจำกัดอัตราส่วนต่อแบนวิดท์สำหรับส่งไอซีเอ็มพีสำหรับ IPv6 โดยทั้งสองแบบต้องมีการตั้งค่าที่เหมาะสม

7.2 ข้อความบ่งบอกความผิดพลาดของ ICMPv6 (ICMPv6 Error Message)

7.2.1 ข้อความบ่งบอกไม่สามารถส่งแพ็กเก็ตถึงที่หมายได้ (Destination Unreachable Message)

Type	Code	Checksum
Unused		
ไม่เกิน Minimum MTU(576 Octets)		

รูปที่ 7.2 รูปแบบของ ไอซีเอ็มพีสำหรับ IPv6 ชนิด Destination Unreachable Message

Destination Address - นำมาจาก Source Address ของแพ็กเก็ตที่เกิดปัญหา

Type - 1

Code - 0 No route to destination
1 Communication with destination adminstratively prohibited
2 (not assigned)
3 Address unreachable
4 Port unreachable

Unused - ไม่มีการใช้งานในทุกๆ Code และต้องเซ็ตเป็น 0 โดยฝ่ายส่งและฝ่ายรับไม่ต้องสนใจ

ข้อความชนิดนี้ควรจะถูกสร้างโดยเราเตอร์หรือโพรโตคอลในระดับชั้นอินเทอร์เน็ต ใช้ในกรณีที่ไม่สามารถส่งแพ็กเก็ตให้ถึงปลายทางได้โดยสาเหตุใดๆที่ไม่ได้เกิดจากความคับคั่ง (Congestion) (จะต้องไม่สร้างไอซีเอ็มพีสำหรับ IPv6 เพราะความคับคั่งโดยเด็ดขาด)

หากเกิดความผิดพลาดเนื่องมาจากไม่สามารถค้นหาเส้นทางส่งต่อจากตารางการหาเส้นทางได้ในส่วน Code จะมีค่าเป็น 0 (จะเกิดได้ในกรณีที่ไม่ได้กำหนด default route ไว้) และโหนดที่ได้รับข้อความชนิดนี้จะต้องแจ้งต่อให้กับโพรโตคอลในระดับชั้นที่สูงกว่า

7.2.2 ข้อความบงบอกขนาดแพ็กเก็ตใหญ่เกินไป (Packet Too Big Message)

Type	Code	Checksum
MTU		
ไม่เกิน Minimum MTU(576 Octets)		

รูปที่ 7.3 รูปแบบของ ไอซีเอ็มพีสำหรับ IPv6 ชนิด ination Address - นำมาจาก Source Address ของแพ็กเก็ตที่เกิดปัญหา

Type - 1

Code - กำหนดให้เป็น 0

MTU - ขนาดสูงสุดที่สามารถส่งต่อไปยังโหนดถัดไปได้

ข้อความชนิดนี้จะต้องถูกส่งโดยเราเตอร์เมื่อไม่สามารถส่งต่อแพ็กเก็ตเนื่องมาจากขนาดของแพ็กเก็ตมีขนาดใหญ่กว่าการเชื่อมต่อส่วนที่จะส่งออกไปโดยจะรายงานค่า MTU กลับไป ข้อความชนิดนี้เป็นข้อยกเว้นสำหรับ Multicast Address โดยสามารถรายงาน Message ชนิดนี้กลับไปได้

7.2.3 ข้อความบ่งบอกเวลาเกินที่กำหนด (Time Exceeded Message)

Type	Code	Checksum
Unused		
ไม่เกิน Minimum MTU(576 Octets)		

รูปที่ 7.4 รูปแบบของ ไอซีเอ็มพีสำหรับ IPv6 ชนิด Time Exceeded Message

Destination Address - นำมาจาก Source Address ของแพ็กเก็ตที่เกิดปัญหา

Type - 3

Code - 0 Hop limit exceeded in transit

1 Fragment reassembly time exceeded

Unused - ไม่มีการใช้งานในทุก Code และต้องเซ็ตเป็น 0 โดยฝ่ายส่งและฝ่ายรับไม่ต้องสนใจ

เมื่อเราเตอร์ได้รับแพ็กเก็ตที่มีค่า Hop Limit เป็น 0 หรือเราเตอร์ลดค่าจนเหลือ 0 แล้วจะทำการทิ้งแพ็กเก็ตนั้นแล้วส่ง ข้อความชนิดนี้โดยมี Code เป็น 0 กลับไป โหนดที่ได้รับแพ็กเก็ตชนิดนี้จะต้องส่งต่อให้กับโพรโตคอลในระดับชั้นที่สูงกว่า

7.2.4 ข้อความบ่งบอกความผิดพลาดในพารามิเตอร์ (Parameter Problem Message)

Type	Code	Checksum
Pointer		
ไม่เกิน Minimum MTU(576 Octets)		

รูปที่ 7.5 รูปแบบของ ไอซีเอ็มพีสำหรับ IPv6 ชนิด Parameter Problem Message

Destination Address - นำมาจาก Source Address ของแพ็กเก็ตที่เกิดปัญหา

Type - 4

Code - 0 Erroneous header field encountered

1 Unrecognized Next Header type encountered

2 Unrecognized IPv6 Option encountered

Pointer - ระบุตำแหน่งที่ตรวจพบข้อผิดพลาด จะชี้ไปตำแหน่งที่อยู่นอกส่วนไอซีเอ็มพีสำหรับ IPv6 ถ้าตำแหน่งที่เกิดข้อผิดพลาดมากกว่าขนาดที่ชี้ได้

เมื่อโหนด IPv6 ตรวจพบปัญหาที่ไม่สามารถจะดำเนินการต่อจนจบกระบวนการได้ จะต้องทิ้งแพ็กเก็ตนั้นและควรที่จะส่ง ข้อความชนิดนี้กลับไปโดยระบุชนิดของปัญหาและตำแหน่งที่ตรวจพบด้วย และโหนดที่ได้รับข้อความชนิดนี้จะต้องส่งต่อให้กับโพรโตคอลในระดับที่สูงกว่าดำเนินการ

7.3 ข้อความบ่งบอกข่าวสารของ ICMPv6 (ICMPv6 Information Message)

7.3.1 ข้อความแสดงการร้องขอ (Echo Request Message)

Type	Code	Checksum

Identifier	Sequence Number
Data	

รูปที่ 7.6 รูปแบบของ ไอซีเอ็มพีสำหรับ IPv6 ชนิด Echo Request Message

Destination Address - แอดเดรสใดๆ

Type - 128

Code - 0

Identifier - ตัวระบุที่จะใช้จับคู่กับ Echo Replies อาจมีค่าเป็น 0

Sequence Number - จำนวนลำดับที่ใช้จับคู่กับ Echo Replies อาจมีค่าเป็น 0

Data - อาจมีหรือไม่มีก็ได้ ถ้ามีต้องเป็นจำนวนที่หารแปดลงตัว

ทุกๆโหนดจำเป็นที่จะต้องอิมพิวเมนต์ ICMPv6 Echo responder ที่สามารถรับ Echo Requests และสามารถตอบ Echo Replies ได้ และควรที่จะมีการอิมพิวเมนต์ ในระดับชั้นโปรแกรมประยุกต์ เพื่อทำงานตรงนี้ด้วย (หากมีก็จะส่งข้อความนี้ต่อให้)

7.3.2 ข้อความแสดงการตอบรับ (Echo Reply Message)

Type	Code	Checksum
Identifier		Sequence Number
Data		

รูปที่ 7.7 รูปแบบของไอซีเอ็มพีสำหรับ IPv6 ชนิด Echo Request Message

Destination Address - มาจาก Source Address ของ Echo Request แพ็กเก็ต

Type - 129

Code - 0

Identifier - ตัวระบุที่จะใช้จับคู่กับ Echo Requests อาจมีค่าเป็น 0

Sequence Number - จำนวนลำดับที่ใช้จับคู่กับ Echo Requests อาจมีค่าเป็น 0

Data - ข้อมูลที่มาจาก Echo Request Message

Source Address ของ Echo Request Message จะต้องเหมือนกับ Destination Address ของ Echo Reply Message และ Echo Reply ควรที่จะถูกส่งตอบกลับ Echo Request ที่มี Destination เป็น มัลติแคสต์แอ็แดรส โดยมี Source Address เป็นแบบยูนิแคสต์แอ็แดรส

7.4 พิจารณาด้านความปลอดภัย (Security)

การติดต่อโดยใช้ ไอซีเอ็มพีสำหรับ IPv6 สามารถทำการพิสูจน์ตัวจริง(Authentication) โดยใช้ IP Authentication Header ได้ (RFC 2402) โดยฝ่ายรับจะต้องทำการพิสูจน์ตัวจริงและทิ้งแพ็กเก็ตที่ไม่ผ่านการพิสูจน์ตัวจริง

ผู้ดูแลระบบอาจจะทำการปรับแต่งให้โหนดละทิ้งไอซีเอ็มพีสำหรับ IPv6 ที่ไม่ได้ทำการพิสูจน์ตัวจริงหรือทำการเข้ารหัส (Encapsulating Security Payload – RFC 2406) ได้

ไอซีเอ็มพีสำหรับ IPv6 อาจจะเป็นเป้าหมายในการโจมตีจากผู้บุกรุกได้ โดยอาจมีลักษณะดังต่อไปนี้

1. อาจมีการหลอกผู้รับไอซีเอ็มพีสำหรับ IPv6 ถูกส่งมาจากผู้ส่งที่ไม่ถูกต้อง ทางแก้คือใช้การพิสูจน์ตัวจริง
2. อาจมีการทำให้การตอบกลับส่งไปยังผู้รับที่ไม่ถูกต้องได้
3. ไอซีเอ็มพีสำหรับ IPv6 อาจจะถูกเปลี่ยนข้อมูล
4. ไอซีเอ็มพีสำหรับ IPv6 อาจจะถูกใช้เพื่อโจมตีการให้บริการโดยการส่งแพ็กเก็ตที่ผิดพลาดซ้ำๆ ทำให้ ทางแก้โดยจำกัดแพ็กเก็ตไอซีเอ็มพีสำหรับ IPv6 ที่จะส่งออกไป

บทที่ 8
โพรโตคอลการค้นพบโหนดที่อยู่ใกล้
(Neighbor Discovery Protocol)

Neighbor Discovey Protocol ประกอบขึ้นจากโพรโตคอลหลายๆ โพรโตคอลของ IPv4 เช่น Address Resolution Protocol (ARP, RFC-826) , ICMP Router Discovery Protocol (RDISC,RFC-1256) , ICMP Redirection Message (RFC-792) ข้อมูลในบทนี้ส่วนใหญ่มาจาก RFC-2461

โฮสต์และเราเตอร์ต่างๆ ใช้ Neighbor Discovery Protocol ในการหาลิ้งเลเยอร์แอดเดรสและยังใช้โพรโตคอลนี้ในการหาเราเตอร์ที่ติดกันที่ยอมส่งแพ็กเก็ตผ่านออกไป และใช้สำหรับการตรวจสอบว่าโหนดที่อยู่ในเครือข่ายเดียวกันมีโหนดไหนบ้างที่ยังสามารถติดต่อได้ และเมื่อเราเตอร์ที่ติดต่อไม่สามารถส่งแพ็กเก็ตได้อีกต่อไปโหนดก็จะใช้โพรโตคอลนี้ในการหาเราเตอร์ตัวอื่นที่ยอมรับการส่งแพ็กเก็ตแทน

Neighbor Discovery Protocol จะมีการใช้แอดเดรสแบบมัลติแคสต์ในชั้นลิ้งค์เลเยอร์สำหรับการให้บริการบางอย่าง ดังนั้นในลิ้งค์รูปแบบต่างๆ ต้องมีการกำหนดมาตรฐานของมันเอง เช่น IPv6 Over FDDI , IPv6 Over ATM

Neighbor Discovery Protocol มีการใช้แอดเดรสหลายแบบดังนี้

- แอดเดรสสำหรับโหนดทุกโหนดที่อยู่ในลิ้งค์ดียวกัน (All-nodes multicast address) ใช้ FFO2::1
- แอดเดรสสำหรับเราเตอร์ทุกตัวที่อยู่ในลิ้งค์ดียวกัน (All-routers multicast address) ใช้ FFO2::2
- แอดเดรสแบบมัลติแคสต์ที่ใช้ภายในลิ้งค์ (Solicited-node multicast address) สามารถสร้างได้โดยการนำ 32 บิตล่างของแอดเดรสเป้าหมายไปต่อกับค่าพรีฟิกซ์ (Prefix) ขนาด 96 บิต FF02:0:0:0:0:1
- แอดเดรสที่ใช้อยู่ภายในลิ้งค์เดียวกัน (Link-local address)
- แอดเดรสแบบระบุค่าไม่ได้ (Unspecified address) สำหรับบอกว่าโหนดนั้นๆ ไม่รู้แอดเดรสของตัวเอง อาจใช้เป็นค่า Source Address เมื่อโหนดนั้นๆ ยังไม่รู้แอดเดรสของตัวเอง ใช้ 0:0:0:0:0:0:0:0

8.1 ภาพรวมของโพรโตคอล

Neighbor Discovery Protocol ช่วยในการแก้ปัญหาที่เกี่ยวกับการติดต่อระหว่างโหนดที่อยู่ในลิ้งค์เดียวกัน โดยมีวิธีการต่างๆเพื่อช่วยแก้ปัญหาดังนี้

Router Discovery : เพื่อแก้ปัญหาว่าโฮสต์จะค้นพบและติดต่อกับเราเตอร์ที่อยู่ในลิ้งค์เดียวกันอย่างไร

Prefix Discovery : เพื่อแก้ปัญหาว่าโฮสต์จะค้นพบค่าพรีฟิกซ์ได้อย่างไร (Address Prefix) โหนดจะใช้ค่าพรีฟิกซ์เพื่อค้นหาว่าเป้าหมายที่จะส่งข้อมูลไปนั้นต้องส่งผ่านเราเตอร์หรือไม่

Parameter Discovery : เพื่อแก้ปัญหาว่าโหนดจะสามารถรับรู้ค่าตัวแปรต่างๆ เช่น ค่า MTU ค่า Hop Limit เพื่อใช้ในสร้างแพ็กเก็ตออกไปได้อย่างไร

Address Autoconfiguration : เพื่อแก้ปัญหาว่าโหนดจะติดตั้งค่าแอดเดรสให้กับอินเตอร์เฟซของตนโดยอัตโนมัติได้อย่างไร

Address Resolution : เพื่อแก้ปัญหาว่าโหนดจะหาค่าลิ้งเลเยอร์แอดเดรส ของโหนดเป้าหมายที่อยู่ในลิ้งค์เดียวกันได้อย่างไร

Next-hop Determination : เพื่อแก้ปัญหาว่าโหนดจะแปลงค่าแอดเดรสของเครื่องเป้าหมายไปยังแอดเดรสของเครื่องที่จะต้องทำการส่งแพ็กเก็ตนั้นผ่านไป (อาจเป็นเราเตอร์หรือเครื่องเป้าหมายก็ได้)

Neighbor Unreachability Detection : เพื่อแก้ปัญหาว่าโหนดจะรู้ได้อย่างไรว่าโหนดที่อยู่ในลิ้งค์เดียวกันโหนดไหนสามารถติดต่อได้ โหนดไหนไม่สามารถติดต่อได้

Duplicate Address Detection : เพื่อแก้ปัญหาว่าโหนดจะรู้ได้อย่างไรว่าแอดเดรสที่กำลังจะใช้นั้นไม่ได้ถูกใช้โดยโหนดอื่น

Redirection : เพื่อแก้ปัญหาว่าเราเตอร์จะทำอย่างไรเพื่อบอกให้โหนดที่จะส่งแพ็กเก็ตมาว่าจะเป็นทางเลือกที่ดีกว่าถ้าส่งแพ็กเก็ตนั้นไปยังเราเตอร์ตัวอื่น

Neighbor Discovery Protocol ได้กำหนดแพ็กเก็ตไอซีเอ็มพีมาห้าแบบดังนี้

Router Solicitation : ใช้เมื่อโฮสต์เริ่มใช้งาน โฮสต์จะส่งไอซีเอ็มพีแบบนี้เพื่อให้เราเตอร์ส่งไอซีเอ็มพี แบบ Router Advertisement ตอบกลับมาทันทีโดยไม่ต้องคอยให้ครบช่วงระยะเวลา (เราเตอร์จะส่งไอซีเอ็มพีแบบ Router Advertisement ออกมาทุกๆช่วงเวลาที่กำหนด)

Router Advertisement : เราเตอร์จะใช้ไอซีเอ็มพีแบบนี้เพื่อประกาศค่าต่างๆ เช่น ค่าพรีฟิกซ์ ค่า Hop Limit และใช้เพื่อยืนยันว่าตนเองยังทำงานอยู่ออกไปทุกๆช่วงเวลาที่กำหนด หรือใช้เพื่อตอบกลับเมื่อได้รับไอซีเอ็มพี แบบ Router Solicitaion

Neighbor Solicitation : ส่งโดยโหนดเพื่อหาค่าลิ้งเลเยอร์แอดเดรสของโหนดในลิ้งค์เดียวกันหรือใช้ตรวจสอบว่าโหนดในลิ้งค์เดียวกันยังสามารถติดต่อได้หรือไม่ และยังใช้สำหรับตรวจหาว่ามีการใช้แอดเดรสซ้ำกันหรือไม่

Redirect : ใช้โดยเราเตอร์เพื่อบอกโหนดว่าควรส่งแพ็กเก็ตไปยังเราเตอร์ตัวที่เหมาะสมสำหรับเป้าหมายหนึ่งๆ

สำหรับการทำงานของ Neighbor Discovery Protocol เราเตอร์แต่ละตัวจะประกาศข้อความ Router Advertisement ทุกๆ ช่วงเวลาเพื่อบอกถึงการมีอยู่ เมื่อโฮสต์ได้รับข้อความนี้จากเราเตอร์แต่ทุกตัว

ในลิ้งค์เดียวกันก็จะสร้างรายชื่อของเร้าเตอร์หลัก (Default Router) ที่ใช้ส่งแพ็กเก็ตออกไป เราเตอร์จะสร้างข้อความนี้ถี่เพียงพอที่จะให้โฮสต์รับรู้ถึงการมีอยู่ของมัน Router Advertisement ยังอนุญาตให้เราเตอร์บอกโฮสต์ว่าจะจัดการติดตั้งแอดเดรสโดยอัตโนมัติอย่างไร

โหนดจะสามารถจัดการหาค่าลิ้งค์เลเยอร์แอดเดรสของของโหนดที่ต้องการได้โดยการประกาศข้อความ Neighbor Solicitation เพื่อสอบถาม เมื่อโหนดเป้าหมายได้รับข้อความนี้ก็จะตอบค่าลิ้งค์เลเยอร์แอดเดรสของตนกลับมาด้วยข้อความ Neighbor Advertisement ไปยังโหนดที่สอบถาม

8.2 การเปรียบเทียบกับ IPv4

Neighbor Discovery Protocol เกิดจากหลายๆ โพรโตคอลของ IPv4 มารวมกันดังที่กล่าวมาแล้ว โดยได้ปรับปรุงความสามารถต่างๆ ดังนี้

- Router Discovery เป็นส่วนหนึ่งของชุดโพรโตคอลพื้นฐาน โฮสต์จึงไม่มีความจำเป็นที่จะต้องคอยสอดส่องข้อมูลจากโพรโตคอลของการหาเส้นทางอื่นๆ
- Router Advertisement ได้บรรจุข้อมูลของลิ้งเลเยอร์แอดเดรสไว้อยู่แล้ว จึงไม่จำเป็นต้องมีการแลกเปลี่ยนแพ็กเก็ตเพิ่มเติมเพื่อหาลิ้งเลเยอร์แอดเดรส
- Router Advertisement ได้บรรจุข้อมูลของค่าพรีฟิกซ์ จึงไม่มีความจำเป็นที่ต้องหาวิธีการอื่นในการกำหนดหรือติดตั้งค่าเน็ตมาสก์ (netmask)
- เราเตอร์สามารถประกาศค่า MTU สำหรับโฮสต์ในลิ้งค์เดียวกัน เพื่อให้แน่ใจว่าในลิ้งค์เดียวกันโหนดทุกโหนดใช้ค่า MTU เดียวกันหมดสำหรับลิ้งค์ที่ปราศจากการกำหนดค่า MTU ที่ดี
- Redirect บรรจุลิ้งเลเยอร์แอดเดรสของโหนดที่ต้องส่งแพ็กเก็ตนั้นไปหาใหม่อยู่แล้ว จึงไม่จำเป็นต้องมีส่งค่าแอดเดรสนั้นแยกออกไปต่างหาก
- โดยปกติแล้วค่าพรีฟิกซ์จะมีอยู่หลายแบบ โฮสต์สามารถเรียนรู้ค่าเหล่านี้ทั้งหมดจาก Router Advertisement แต่อย่างไรก็ดีเราเตอร์สามารถปรับเพื่อให้ค่าพรีฟิกซ์บางแบบไม่ต้องถูกส่งออกไปในการประกาศของ Router Advertisement ในกรณีนี้โฮสต์จะรับรู้ว่าเป้าหมายที่จะติดต่อนั้นไม่ได้อยู่ในลิ้งค์เดียวกัน ทำให้ต้องส่งแพ็กเก็ตไปยังเราเตอร์ เราเตอร์ก็สามารถส่งแพ็กเก็ตนั้นต่อไปแทน
- Neighbor Unreachability Detection คือส่วนสำคัญส่วนหนึ่งเพื่อเพิ่มความมั่นคงให้กับการส่งแพ็กเก็ตเมื่อแพ็กเก็ตส่งไปยังเราเตอร์ที่ไม่สามารถทำงานได้ หรือส่งไปยังโหนดที่เปลี่ยน ลิ้งเลเยอร์แอดเดรส เช่น โหนดเคลื่อนที่สามารถเคลื่อนออกไปนอกลิ้งค์โดยไม่ต้องสูญเสียการติดต่ออันเนื่องมาจากข้อมูลเก่าที่อยู่ในหน่วยความจำของเออาร์พี (ARP)
- ไม่เหมือนใน IPv4 Router Discovery ในไอซีเอ็มพี แบบ Router Advertisement ไม่มีการบรรจุค่า Preference ค่านี้ไม่มีความจำเป็นอีกต่อไป เพราะ Neighbor

Unreachability Detection จะคอยตรวจเราเตอร์ที่ไม่สามารถใช้งานได้และเปลี่ยนไปยังเราเตอร์ที่ใช้งานได้แทน

- การใช้ Link local address (ดูในบทการจัดการแอดเดรส) เพื่อระบุถึงเราเตอร์ทำให้เป็นไปได้สำหรับโฮสต์ใดๆ ว่ายังสามารถติดต่อกับกลุ่มของเราเตอร์เดิมได้ในกรณีที่มีการปรับเปลี่ยนค่าแอดเดรสทั้งเครือข่าย เพื่อใช้ค่าพรีฟิกซ์แบบใหม่
- การใช้ค่า Hop Limit เท่ากับ 255 เป็นวิธีเพื่อป้องกันโหนดภายนอกลิ้งค์ส่งข้อความใดๆ ของ Neighbor Discovery Protocol เข้ามาภายในเครือข่ายได้ทั้งแบบตั้งใจหรือเป็นอุบัติเหตุ ใน IPv4 โหนดภายนอกลิ้งค์สามารถส่งข้อความแบบ ICMP Redirect และ Router Advertisement เข้ามาภายในลิ้งค์ได้
- การให้ Address Resolution อยู่ในชั้นของไอซีเอ็มพีทำให้โพรโตคอลไม่ขึ้นอยู่กับรูปแบบเครือข่าย (ATM , Ethernet) เหมือนโพรโตคอลเออาร์พีและทำให้ใช้ความสามารถของส่วนเพิ่มเติมส่วนหัวแบบการพิสูจน์ตัวจริงและการเข้ารหัสของ IPv6 ได้

8.3 ข้อความเราเตอร์ร้องขอ (Router Solicitaion Message)

โฮสต์จะส่งข้อความนี้เพื่อให้เราเตอร์ส่งข้อความ Router Advertisement ตอบกลับมาทันที โดยสาเหตุที่โฮสต์จะส่งข้อความนี้มามีดังเช่น

- โฮสต์เปิดใช้อินเตอร์เฟซที่เวลาเริ่มทำงาน
- อินเตอร์เฟซเริ่มใช้งานอีกครั้งหลังจากการใช้งานล้มเหลวหรือจากการปิดการใช้งานอินเตอร์เฟซ
- การปิดการใช้งานเราเตอร์ทำให้เราเตอร์ตัวเดิมไม่สามารถส่งแพ็กเก็ตออกไป
- โฮสต์เริ่มใช้งานในลิ้งค์นี้เป็นครั้งแรก

Router Solicitation มีรูปแบบโครงสร้างดังรูป

1 16 32

Type	Code	Checksum
Reserved		
Options		

รูปที่ 8.1 แสดงโครงสร้างของ Router Solicitation

ข้อมูลในส่วนของไอพี

Source Address : ค่าแอดเดรสของอินเตอร์เฟซที่ส่งแพ็กเก็ตนี้ออกมา ใช้แอดเดรสแบบระบุค่าไม่ได้ถ้าไม่ทราบแอดเดรสของตนเอง

Destination Address : โดยปกติแล้วใช้แอดเดรสสำหรับเราเตอร์ทุกตัวที่อยู่ในลิ้งค์เดียวกัน

Hop Limit : 255

ข้อมูลในส่วนของไอซีเอ็มพี

Type : 133

Code : 0

Checksum : ดูในบทไอซีเอ็มพีสำหรับ IPv6

Reserved : ไม่มีการใช้ค่านี้ ให้โหนดที่ส่งแพ็กเก็ตนี้ตั้งเป็นศูนย์ทั้งหมดและไม่ต้องสนใจค่านี้สำหรับโหนดที่รับแพ็กเก็ต

ค่า Options ที่สามารถใช้ได้มีดังนี้

Source link-layer address : เป็นลิ้งเลเยอร์แอดเดรสของผู้ส่ง

8.4 ข้อความเราเตอร์ประกาศ (Router Advertisement Message)

เราเตอร์จะประกาศข้อความแบบนี้เมื่อมีการร้องขอด้วย Router Solicitation หรือจะประกาศทุกๆระยะเวลาที่กำหนด เพื่อบอกว่าเราเตอร์ยังทำงานอยู่และใช้ประกาศค่าข้อมูลต่างๆ ทำให้ผู้ดูแลระบบสามารถจัดการกับค่าต่างๆ เพียงแค่ที่เราเตอร์แล้วเราเตอร์จะประกาศค่าเหล่านั้นออกไปเองโดยอัตโนมัติ เช่น ค่า MTU ค่า Hop Limit โฮสต์จะใช้ข้อมูลนี้ในการสร้างรายชื่อของเราเตอร์ที่สามารถทำงานได้

Router Advertisement มีรูปแบบโครงสร้างดังนี้

1 16 32

Type	Code			Checksum
Cur Hop Limit	M	O	Reserved	Router Lifetime
Reachable Time				
Retrans Timer				
Options				

รูปที่ 8.2 แสดงรูปแบบโครงสร้างของ Router Advertisement

ข้อมูลในส่วนของไอพี

Source Address : ค่าแอดเดรสที่ใช้อยู่ภายในลิ้งค์เดียวกัน (Link-local address) ของเราเตอร์ที่ส่งข้อความนี้ไป

Destination Address : โดยปกติแล้วใช้แอดเดรสของโหนดที่ประกาศ Router Solicitation หรือใช้แอดเดรสสำหรับโหนดทุกโหนดในลิ้งค์เดียวกัน (All-nodes multicast address)

Hop Limit : 255

ข้อมูลในส่วนของไอซีเอ็มพี

Type : 134

Code : 0

Checksum : ดูในบทไอซีเอ็มพีสำหรับ IPv6

Cur Hop Limit : ขนาด 8 บิต แสดงค่าที่ส่งให้ค่า Hop Limit ที่อยู่ในส่วนหัวของไอพี

M (Managed address configuration) : ขนาด 1 บิต เมื่อมีค่าเป็นหนึ่งโฮสต์จะใช้ Stateful Protocol (DHCP) สำหรับรับค่าแอดเดรสมาใช้ (โฮสต์รับค่าแอดเดรสมาจากเซริฟเวอร์)

O (Other stateful configuration) : ขนาด 1 บิต เมื่อมีค่าเป็นหนึ่งโฮสต์จะใช้ Stateful Protocol สำหรับรับค่าหรือข้อมูลอื่นๆ ที่ไม่ใช่แอดเดรสมาใช้

Reserved : ขนาด 6 บิต โดยให้ผู้ส่งตั้งเป็นศูนย์ทั้งหมดและไม่ต้องสนใจค่านี้สำหรับโหนดที่รับแพ็กเก็ต

Router Lifetime : ขนาด 16 บิต ใช้เพื่อบอกว่าเราเตอร์ตัวนี้สามารถใช้งานเป็นเราเตอร์หลัก (Default Router) ได้เป็นเวลาเท่าใด โดยมีหน่วยเป็นวินาที เวลามากที่สุดคือ 18.2 ชั่วโมง ถ้าค่านี้เป็น 0 หมายความว่าเราเตอร์ตัวนี้ไม่ต้องการให้ใช้เป็นเราเตอร์หลัก และโหนดที่รับค่านี้มาก็ไม่ควรตั้งเราเตอร์ตัวนี้เป็นเราเตอร์หลัก

Reachable Time : ขนาด 32 บิต มีหน่วยเป็นมิลลิวินาที ใช้เพื่อบอกให้โหนดสมมติว่าเราเตอร์ตัวนี้ใช้งานได้เป็นเวลาเท่าใดหลังจากได้รับการยืนยันมา ใช้โดย Neighbor Unreachability Detection ค่าเป็นศูนย์หมายความว่าไม่สามารถระบุค่าได้

Restrans Timer : ขนาด 32 บิต มีหน่วยเป็นมิลลิวินาที ใช้เพื่อบอกระยะเวลาที่จะส่งข้อความ Neighbor Solicitation อีกครั้ง ค่าเป็นศูนย์หมายความว่าไม่สามารถระบุค่าได้

ค่า Options ที่สามารถใช้ได้มีดังนี้

Source link-layer address : เป็นลิ้งเลเยอร์แอดเดรสของผู้ส่งข้อความนี้

MTU : ใช้เพื่อบอกค่า MTU

Prefix Information : ใช้สำหรับบอกค่าพรีฟิกซ์ที่มีใช้อยู่ในลิ้งค์ และใช้สำหรับช่วยในการกำหนดค่าแอดเดรส เราเตอร์ควรประกาศค่าพรีฟิกซ์ที่มีใช้อยู่ทั้งหมดในลิ้งค์เพื่อให้โฮสต์ที่มีหลายอินเตอร์เฟซเลือกได้ถูกอย่างต้องว่าต้องส่งแพ็กเก็ตออกไปทางอินเตอร์เฟซไหน

8.5 ข้อความโหนดใกล้เคียงร้องขอ (Neighbor Solicitation Message)

โหนดจะใช้ข้อความแบบ Negihbor Solicitation เพื่อร้องขอลิ้งเลเยอร์แอดเดรส ของโหนดเป้าหมายที่จะส่งแพ็กเก็ตไปให้ และในเวลาเดียวกันก็จะให้ลิ้งเลเยอร์แอดเดรส ของตัวเองแก่โหนดเป้าหมาย โหนดสามารถใช้ข้อความนี้เพื่อตรวจสอบว่าโหนดเป้าหมายนั้นยังสามารถใช้งานได้หรือไม่

โหนดสามารถใช้ข้อความนี้เพื่อหาลิ้งเลเยอร์แอดเดรส ได้โดยการส่งแอดเดรสแบบมัลติแคสต์ (Solicited-node multicast address) ถามถึงลิ้งเลเยอร์แอดเดรส ไปยังโหนดในลิ้งค์เดียวกัน เมื่อโหนดเป้าหมายได้รับข้อความนี้ก็จะตอบกลับมาด้วยข้อความแบบ Neighbor Advertisement

เรายังสามารถใช้ข้อความนี้ในการตรวจสอบว่ามีการใช้แอดเดรสซ้ำกันหรือไม่โดยวิธีการใช้จะอยู่ใน RFC-1971 " IPv6 Stateless Address Autoconfiguration "

โหนดจะใช้แอดเดรสแบบมัลติแคสต์เมื่อถามถึงลิ้งเลเยอร์แอดเดรส และจะใช้แอดเดรสแบบยูนิแคสต์เมื่อต้องการตรวจสอบการใช้แอดเดรสซ้ำกัน

Neighbor Solicitation มีโครงสร้างดังนี้

1 16 32

Type	Code	Checksum
Reserved		
Target Address		
Options..		

รูปที่ 8.3 แสดงโครงสร้างของ Neighbor Solicitation

ข้อมูลในส่วนของไอพี

Source Address : แอดเดรสที่ของโหนดที่ส่งข้อความนี้หรือแอดเดรสแบบระบุค่าไม่ได้

Destination Address : แอดเดรสแบบมัลติแคสต์ (Solicited-node multicast address) หรือแอดเดรสของโหนดเป้าหมาย

Hop Limit : 255

ข้อมูลในส่วนของไอซีเอ็มพี

Type : 135

Code : 0

Checksum : ดูในบทไอซีเอ็มพีสำหรับ IPv6

Reserved : ไม่มีการใช้ค่านี้ ให้ผู้ส่งตั้งเป็นศูนย์ทั้งหมดและไม่ต้องสนใจค่านี้สำหรับโหนดที่รับแพ็กเก็ต

Target Address : แอดเดรสของโหนดเป้าหมายต้องไม่เป็นแอดเดรสแบบมัลติแคสต์

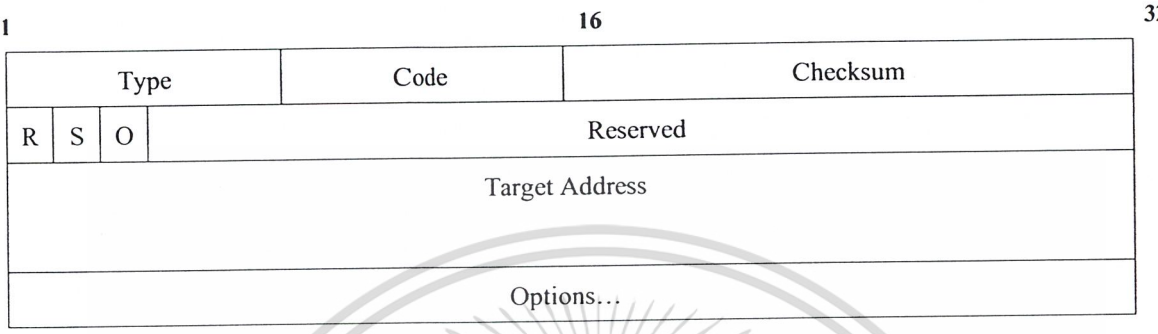
ค่า Options ที่สามารถใช้ได้มีดังนี้

Source link-layer address : เป็นลิ้งเลเยอร์แอดเดรสของผู้ส่งข้อความนี้

8.6 ข้อความโหนดใกล้เคียงประกาศ (Neighbor Advertisement Message)

โหนดจะส่งข้อความแบบนี้เพื่อตอบกลับข้อความ Neighbor Solicitation และโหนดยังสามารถส่งข้อความ Neighbor Advertisement เมื่อมีการเปลี่ยนลิ้งเลเยอร์แอดเดรส ของตนเองเพื่อการกระจายข้อมูลอย่างรวดเร็ว

Neighbor Advertisement มีโครงสร้างดังนี้



รูปที่ 8.4 โครงสร้างของ Neighbor Advertisement

ข้อมูลในส่วนของไอพี

Source Address : แอดเดรสที่ของโหนดที่ส่งข้อความนี้

Destination Address : สำหรับตอบกลับเมื่อได้รับข้อความ Neighbor Solicitation ใช้แอดเดรสของโหนดที่ส่งข้อความนั้น แต่ถ้าโหนดนั้นใช้แอดเดรสแบบระบุค่าไม่ได้ให้ใช้แอดเดรสสำหรับทุกโหนดที่อยู่ในลิ้งค์เดียวกัน (All-nodes multicast address) และสำหรับการส่งจากโหนดเองโดยไม่ได้รับข้อความ Neighbor Solicitation ก็จะใช้แอดเดรสสำหรับทุกโหนดในลิ้งค์เดียวกัน

Hop Limit : 255

ข้อมูลในส่วนของไอซีเอ็มพี

Type : 136

Code : 0

Checksum : ดูในบท ไอซีเอ็มพีสำหรับ IPv6

R (Router Flag) : เมื่อมีค่าเป็นหนึ่งหมายความผู้ส่งข้อความคือเราเตอร์ โดยค่า R นี้ใช้สำหรับ Neighbor Unreachability Detection เพื่อตรวจเราเตอร์ที่เปลี่ยนเป็นโฮสต์

S (Solicited Flag) : เมื่อมีค่าเป็นหนึ่งหมายความว่าข้อความนี้เกิดจากการตอบรับ Neighbor Solicitation โดยค่า S นี้ใช้สำหรับ Neighbor Unreachability Detection ค่านี้ต้องเป็นศูนย์เมื่อเป็นการประกาศแบบมัลติแคสต์หรือการประกาศโดยไม่ได้รับข้อความ Neighbor Solicitation

O (Override Flag): เมื่อมีค่าเป็นหนึ่งหมายความว่าโหนดที่รับข้อความนี้ควรจะปรับปรุงหน่วยความจำที่เก็บลิ้งเลเยอร์แอดเดรสของตน

Reserved : ไม่มีการใช้ค่านี้ ให้โหนดที่ส่งแพ็กเก็ตตั้งเป็นศูนย์ทั้งหมดและไม่ต้องสนใจค่านี้สำหรับโหนดที่รับแพ็กเก็ต

Target Address : สำหรับการตอบกลับจากข้อความ Neighbor Solicitation ให้ใช้แอดเดรสที่ส่งข้อความนั้นมา และสำหรับการส่งจากโหนดเองค่าแอดเดรสเป็นของโหนดที่เปลี่ยนลิ้งเลเยอร์แอดเดรส ค่า Target Address ต้องไม่เป็นแอดเดรสแบบมัลติแคสต์

ค่า Options ที่สามารถใช้ได้มีดังนี้

Target link-layer address : ค่าลิ้งเลเยอร์แอดเดรสของผู้ส่งข้อความนี้

8.7 ข้อความเปลี่ยนเส้นทาง (Redirect Message)

เราเตอร์จะใช้ข้อความนี้เพื่อบอกโฮสต์ถึงทางเลือกที่ดีกว่าที่จะส่งแพ็กเก็ตไปทางโหนดอื่น และเมื่อค่า Target Address เท่ากับค่า Destinaiton Address (ในส่วนไอซีเอ็มพี) ก็จะหมายความว่าจริงๆ แล้วเป้าหมายนั้นอยู่ภายในลิ้งค์เดียวกัน เมื่อโฮสต์ได้รับข้อความนี้โอสก์ควรปรับปรุงตารางการส่งแพ็กเก็ตของตนเอง

Redirect Message มีโครงสร้างดังนี้

1 16 32

Type	Code	Checksum
Reserved		
Target Address		
Destination Address		
Options		

รูปที่ 8.5 แสดงโครงสร้างของ Redirect Message

ข้อมูลในส่วนของไอพี

Source Address : ต้องเป็นแอดเดรสที่ใช้อยู่ภายในลิ้งค์ของโหนดที่ส่งข้อความนี้

Destination Address : แอดเดรสของโหนดที่ส่งแพ็กเก็ตที่ทำให้เกิดข้อความนี้

Hop Limit : 255

ข้อมูลในส่วนของไอซีเอ็มพี

Type : 137

Code : 0

Checksum : ดูในบทไอซีเอ็มพีสำหรับ IPv6

Reserved : ไม่มีการใช้ค่านี้ ให้โหนดที่ส่งแพ็กเก็ตตั้งเป็นศูนย์ทั้งหมดและไม่ต้องสนใจค่านี้สำหรับเราเตอร์ที่รับแพ็กเก็ต

Target Address : แอดเดรสที่ให้โฮสต์ส่งแพ็กเก็ตไปแทน

Destination Address : แอ็ดแดรซของเป้าหมายสุดท้าย

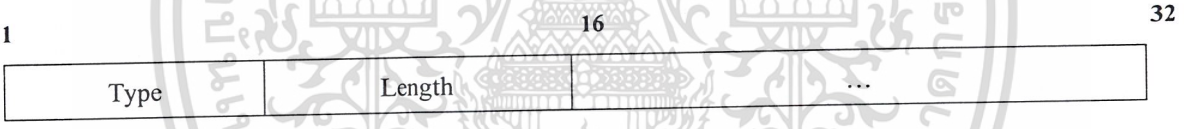
ค่า Options ที่สามารถใช้ได้มีดังนี้

Target link-layer address : เป็นลิ้งเลเยอร์แอดเดรสของเป้าหมายที่ให้ส่งแพ็กเก็ตไปแทน

Redirected Header : ข้อมูลของแพ็กเก็ตที่ทำให้เกิดข้อความนี้มากเท่าที่เป็นไปได้โดยไม่ให้เกิน 1280 ไบต์

8.8 โครงสร้างของออปชั่น (Option Formats)

โครงสร้างของออปชั่นมีดังนี้



รูปที่ 8.6 แสดงโครงสร้างของออปชั่น

Type : ขนาด 8 บิต มีค่าความหมายดังตารางนี้

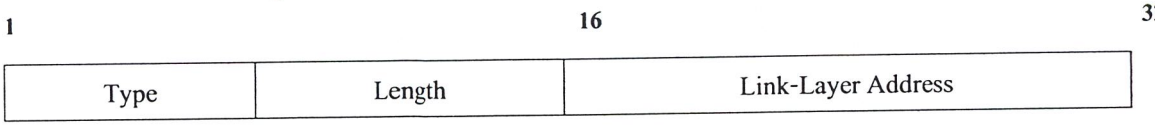
ชื่อของออปชั่น	ค่า
Source Link-Layer Address	1
Target Link-Layer Address	2
Prefix Information	3
Redirected Header	4
MTU	5

ตารางที่ 8.1 แสดงความหมายของค่า Type

Length : ขนาด 8 บิต แสดงความยาวของออปชั่น (รวมค่า Type และค่า Length)

8.8.1 ออปชั่นแบบลิ้งเลเยอร์แอดเดรสของโหนดต้นทางหรือโหนดเป้าหมาย (Source/Target Link-Layer Address)

มีโครงสร้างดังรูป



รูปที่ 8.7 แสดงโครงสร้างของออปชั่นแบบ Source/Target Link-Layer Address

Type : 1 สำหรับ Source Link-Layer Address

2 สำหรับ Target Link-Layer Address

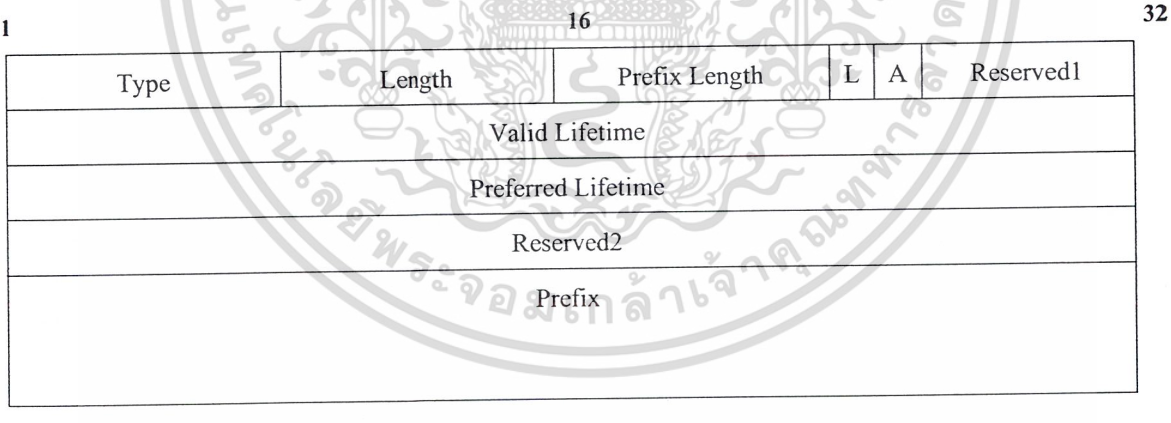
Link-Layer Address : ค่าแอดเดรสของชั้นที่ต่ำกว่าชั้นอินเตอร์เน็ต เช่น IEEE 802 Address

ค่า Source Link-Layer Address คือค่าลิ้งเลเยอร์แอดเดรสของผู้ส่งแพ็กเก็ต ใช้สำหรับ Neighbor Solicitation , Router Solicitation และ Router Advertisement

ค่า Target Link-Layre Address คือค่าลิ้งเลเยอร์แอดเดรสของโหนดที่ให้ส่งแพ็กเก็ตไปหา ใช้สำหรับ Neighbor Advertisement และ Redirect Message

8.8.2 ข้อมูลพรีฟิกซ์ (Prefix Information)

มีโครงสร้างดังรูป



รูปที่ 8.8 แสดงโครงสร้างของออปชั่นแบบ Prefix Information

Type : 3

Prefix Length : ขนาด 8 บิต แสดงความยาวของค่าพรีฟิกซ์มีค่าตั้งแต่ 0-128

L (Link Flag) : เมื่อมีค่าเป็นหนึ่งหมายความว่า ค่าออปชั่นนี้สามารถนำมาใช้เพื่อบอกว่าค่าพรีฟิกซ์นี้สำหรับโหนดภายในลิ้งค์

A (Autonomous Address-Configuration Flag) : เมื่อมีค่าเป็นหนึ่งหมายความว่าค่าพรีฟิกซ์นี้สามารถนำมาใช้สำหรับการติดตั้งแอดเดรสแบบอัตโนมัติได้

Reserved1,2 : ไม่มีการใช้ค่านี้ ให้โหนดที่สร้างแพ็กเก็ตตั้งเป็นศูนย์ทั้งหมดและไม่ต้องสนใจค่านี้สำหรับผู้รับแพ็กเก็ต

Valid Lifetime : ขนาด 32 บิต มีหน่วยเป็นวินาที บอกถึงเวลาที่สามารถอ้างได้ค่าพรีฟิกซ์นี้ถูกต้อง ค่านี้ใช้ในการตัดสินใจว่าโหนดที่ใช้ค่าพรีฟิกซ์นี้อยู่ในลิ้งค์หรือไม่ และค่านี้ยังใช้โดยโพรโตคอลสำหรับการติดตั้งแอดเดรสโดยอัตโนมัติ

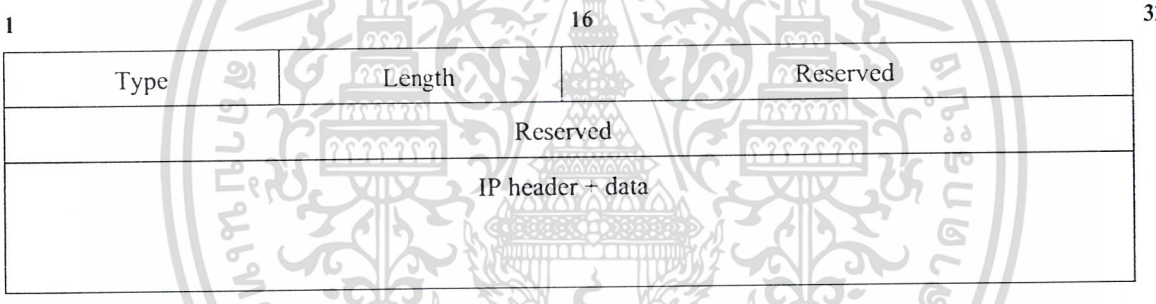
Preferred Lifetime: ขนาด 32 บิต มีหน่วยเป็นวินาที ใช้บอกเวลาที่แอดเดรสที่เกิดจากการใช้ค่าพรีฟิกซ์นี้ สามารถใช้งานได้ ถ้ามีค่าเป็นหนึ่งทั้งหมดหมายถึงสามารถใช้งานได้ตลอดไป

Prefix : ค่าพรีฟิกซ์บิตที่อยู่หลังจากค่า Prefix Length ให้ตั้งเป็นศูนย์ทั้งหมด

ออปชั่นนี้จะปรากฏในข้อความแบบ Router Advertisement

8.8.3 ส่วนหัวของการเปลี่ยนเส้นทาง (Redirected Header)

มีโครงสร้าง ดังรูป



รูปที่ 8.9 แสดงโครงสร้างของออปชั่นแบบ Redirected Header

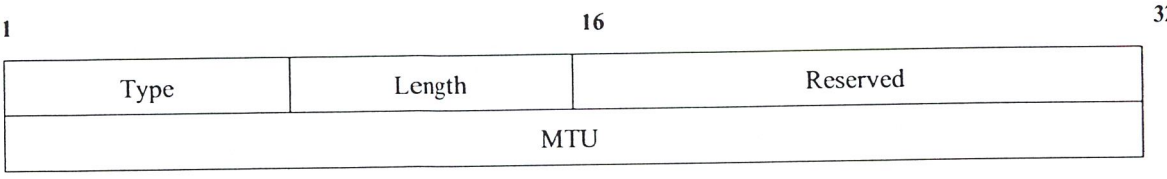
Type : 4

Reserved : ไม่มีการใช้ค่านี้ ให้โหนดที่สร้างแพ็กเก็ตนี้ตั้งเป็นศูนย์ทั้งหมดและไม่ต้องสนใจค่านี้สำหรับผู้รับแพ็กเก็ต

IP Header + data : ประกอบไปด้วยส่วนหัวของไอพี และข้อมูลที่ถูกตัดตอนเพื่อไม่ให้มีขนาดเกิน 1280 ไบต์

8.8.4 ออปชั่นบอกค่าขนาดข้อมูลใหญ่สุด (MTU)

มีโครงสร้าง ดังรูป



รูปที่ 8.10 แสดงโครงสร้างของออปชั่นแบบ MTU

Type : 5

Reserved : ไม่มีการใช้ค่านี้ ให้ผู้ส่งตั้งเป็นศูนย์ทั้งหมดและไม่ต้องสนใจค่านี้สำหรับผู้รับแพ็กเก็ต

MTU : ขนาด 32 บิต แสดงค่า MTU ที่ใช้ในลิ้งค์

ออปชั่นนี้จะถูกใช้ในข้อความแบบ Router Advertisement เพื่อให้แน่ใจว่าทุกโหนดภายในลิ้งค์ใช้ค่า MTU เดียวกัน

บทที่ 9

ส่วนเพิ่มเติมดีเอ็นเอสสำหรับ IPv6

(DNS Extensions to Support IPv6 Address Aggregation and Renumbering)

ดีเอ็นเอส สำหรับ IPv6 ได้มีการปรับปรุงที่สำคัญดังนี้

- เรคคอร์ดแบบใหม่คือ "A6" ใช้สำหรับการเปลี่ยนจากชื่อเป็นแอดเดรส
- การสอบถามเปลี่ยนจากที่สนุบสนุน IPv4 อย่างเดียวเป็นสนับสนุนทั้ง IPv4 และ IPv6
- โดเมนใหม่คือ "IP6.ARPA" ใช้สำหรับเปลี่ยนจากแอดเดรสเป็นชื่อ

การเปลี่ยนแปลงนี้ถูกออกแบบมาให้ทำงานได้ร่วมกับโปรแกรมที่มีอยู่ก่อนแล้ว เรคคอร์ดสำหรับ IPv6 ก่อนที่ได้กำหนดก่อนหน้านี้คือ AAAA จะถูกแทนที่ด้วยเรคคอร์ดแบบ A6 เพราะเรคคอร์ดแบบ A6 ได้ถูกออกแบบมาให้สนับสนุนความสามารถในการเปลี่ยนแปลงแอดเดรส (Network Renumbering) และความสามารถในการมีหลายบ้าน (Multihoming)

9.1 การแปลงจากชื่อไปเป็นแอดเดรส

แอดเดรสแบบ IPv6 ถูกเก็บไว้ในหนึ่งเรคคอร์ดแบบ A6 หรือมากกว่า เรคคอร์ดแบบ A6 อาจเก็บ แอดเดรสแบบ IPv6 แอดเดรสเดียวหรือเก็บแบบช่วงที่ติดกัน ข้อมูลของค่าพรีฟิกซ์จะประกอบไปด้วย ความยาวของพรีฟิกซ์ และชื่อ ดูได้จากหัวข้อโครงสร้างของเรคคอร์ดแบบ A6

แอปพลิเคชั่นต่างๆที่ต้องการดูค่าแอดเดรสแบบ IPv6 จะก่อให้ดีเอ็นเอสรีโซว์เวอร์เข้าถึงเรคคอร์ดแบบ A6 หลายเรคคอร์ด ทำให้อาจจะได้ค่าแอดเดรสแบบ IPv6 หลายแอดเดรสส่งกลับมา

9.2 การแปลงจากแอดเดรสไปเป็นชื่อ

วิธีใหม่สำหรับการแปลงแอดเดรสเป็นชื่อจะใช้ DNS Label แบบใหม่เรียกว่า "Bit-String Label"

ตัวอย่างเช่น ต้องการหาชื่อของแอดเดรส

3ffe:7c0:40:9:a00:20ff:fe81:2b32 จะใช้ QCLASS=IN , QTYPE=PTR อาจได้กลับมาดังนี้

(x หมายความว่าเป็นเลขฐาน 16)

\[x3FFE07C0004000090A0020FFFE812B32/128].IP6.ARPA.

\[x0A0020FFFE812B32/64].\[x0009/16].\[x3FFE07C00040/48].IP6.ARPA.

ตัวอย่างอย่างละเอียดดูได้จากหัวข้อถัดไป

มีการกำหนดเรคคอร์แบบใหม่อีกเรคคอร์ดคือ "DNAME" คล้ายกับเรคคอร์ดแบบ "CNAME" ใช้สำหรับกำหนดชื่อทั้งเครือข่ายย่อยมากกว่าโหนดใดโหนดหนึ่งเพียงโหนดเดียว เช่น

QNAME a.b.c.d.e.f อาจจะได้เรคคอร์ดแบบ DNAME ดังนี้

d.e.f DNAME w.xy.

ทำให้เปลี่ยนเป็น a.b.c.w.xy.

9.3 โครงสร้างของเรคคอร์ดแบบ A6

ส่วน RDATA ของเรคคอร์ดแบบ A6 มีรูปแบบโครงสร้างดังนี้

Prefix Length	Address Suffix	Prefix name

รูปที่ 9.1 แสดงโครงสร้างเรคคอร์ดแบบ A6

Prefix Length : ขนาด 1 ไบต์ คือค่าความยาวของพรีฟิกซ์ มีค่าตั้งแต่ 1-128 บิต

Address Suffix : มีขนาดตั้งแต่ 0-16 ไบต์ แสดงค่าส่วนหลังของแอดเดรสที่มีขนาด 128 ลบด้วยค่า Prefix Length

Prefix Name : ขนาด 0- 255 ไบต์ แสดงชื่อของโดเมน

ในการได้รับแอดเดรสแบบ IPv6 ที่เป็นของชื่อนั้นดีเอ็นเอส ตัวลูกจะได้รับเรคคอร์ดแบบ A6 ที่ต่อกันปีนลูกโซ่ โดยจะเริ่มจากเรคคอร์ดที่เป็นของชื่อนั้น และตามด้วยจะได้เรคคอร์ดที่เป็นของชื่อของพรีฟิกซ์ นั้นไปเรื่อยๆ จนกระทั้งสุดท้ายจะเป็นเรคคอร์ดที่มีค่า Prefix Length เป็น 0

แอดเดรสแบบ IPv6 จะได้มาจากการประกอบข้อมูลในเรคคอร์ดลูกโซ่เข้าด้วยกัน โดยนำทุกบิตของเรคคอร์ดก่อนหน้านี้ที่ครอบคลุมถึงตำแหน่งที่ดูได้จากค่า Prefix Length มาต่อกันไปเรื่อยๆ

9.4 ตัวอย่างการแปลงจากชื่อไปเป็นแอดเดรส

สมมติว่าไซต์ X คือไซต์แบบมีหลายบ้าน คือจากผู้ให้บริการ A และผู้ให้บริการ B ผู้ให้บริการ A ก็มีหลายบ้านคือจากผู้ให้บริการ C และ D ผู้ให้บริการ B ก็ได้รับการบริการจากผู้ให้บริการ E

C,D,E มีผู้ดูแล TLA (Top-Level Aggregator) เดียวกันคือ ALPHA-TLA.ORG มีค่า TLA เท่ากับ 2345 ผู้ดูแล TLA ได้กำหนด NLA (Next-Level Aggregator) ให้ C,D,E ตามลำดับดังนี้

2345:00C0::/28 , 2345:00D0::/28 , 2345:000E::/32

C ได้กำหนดค่า NLA ให้ A คือ 2345:00C1:CA00::/40

D ได้กำหนดค่า NLA ให้ A คือ 2345:00D2:DA00::/40

E ได้กำหนดค่า NLA ให้ B คือ 2345:000E:EB00::/40

ผู้ให้บริการ A ได้กำหนดค่า Subscriber Identification ให้ X คือ 11
ผู้ให้บริการ B ได้กำหนดค่า Subscriber Identification ให้ X คือ 22

ดังนั้นไซต์ X จะมีค่าพรีฟิกซ์ ดังนี้
2345:00C1:CA11::/48 จาก A สำหรับเส้นทางไปสู่ C
2345:00D2:DA11::/48 จาก A สำหรับเส้นทางไปสู่ D
2345:000E:EB22::/48 จาก B สำหรับเส้นทางไปสู่ E

สมมติว่าโหนด N จากซับเน็ต 1 ของไซต์ X ใช้ค่า Interface Identifier เท่ากับ
1234:5678:9ABC:DEF0

ดังนั้นในตัวอย่างนี้โหนด N จะมีค่าแอดเดรสอยู่ 3 ค่า ดังนี้
2345:00C1:CA11:0001:1234:5678:9ABC:DEF0
2345:00D2:DA11:0001:1234:5678:9ABC:DEF0
2345:000E:EB22:0001:1234:5678:9ABC:DEF0

กำหนดให้ X,A,B,C,D,E แสดงใน ดีเอ็นเอส ได้ตามลำดับดังนี้ X.EXAMPLE , A.NET , B.NET , C.NET , D.NET , E.NET และแต่ละโดเมนมีโดเมนย่อยคือ IP6 โหนด N จะถูกระบุได้โดยโดเมน N.X.EXAMPLE เรคคอร์ดข้างล่างจะปรากฏในดีเอ็นเอส ของ X

N	A6 64 ::1234:5678:9ABC:DEF0	SUBNET-1.IP6	**– r1**
SUBNET-1.IP6	A6 48 0:0:0:1::	IP6	**– r2**
IP6	A6 48 0::0	SUBSCRIBER-X.IP6.A.NET.	**– r3**
IP6	A6 48 0::0	SUBSCRIBER-X.IP6.B.NET.	

และที่อื่นจะปรากฏ

SUBSCRIBER-X.IP6.A.NET.	A6 40 0:0:0011::	A.NET.IP6.C.NET.	**– r4**
SUBSCRIBER-X.IP6.A.NET.	A6 40 0:0:0011::	A.NET.IP6.D.NET.	
SUBSCRIBER-X.IP6.B.NET.	A6 40 0:0:0022::	B-NET.IP6.E.NET.	
A.NET.IP6.C.NET.	A6 28 0:0001:CA00::	C.NET.ALPHA-TLA.ORG.	**– r5**
A.NET.IP6.D.NET.	A6 28 0:0002:DA00::	D.NET.ALPHA-TLA.ORG.	
B-NET.IP6.E.NET.	A6 32 0:0:EB00::	E.NET.ALPHA-TLA.ORG.	

C.NET.ALPHA-TLA.ORG. A6 0 2345:00C0:: -- r6

D.NET.ALPHA-TLA.ORG. A6 0 2345:00D0::

E.NET.ALPHA-TLA.ORG. A6 0 2345:000E::

ดังนั้นโหนด N จะมีแอดเดรสที่ได้จากการประกอบข้อมูลในเรคคอร์ดดังนี้

จาก r1 จะได้ 0000:0000:0000:0000:**1234:5678:9ABC:DEF0**

จาก r2 จะได้ 0000:0000:0000:**0001:1234:5678:9ABC:DEF0** นำ 64 (128-64) บิตหลังทุกบิต มาจาก r1

จาก r3 จะได้ 0000:0000:0000:**0001:1234:5678:9ABC:DEF0** มีค่าเท่าเดิมเพราะนำบิตหลัง 80 (128-48) บิตมาจาก r2

จาก r4 จะได้ 0000:0000:00**11:0001:1234:5678:9ABC:DEF0** นำ 88 (128-40) บิตหลังมาจาก r3

จาก r5 จะได้ 0000:0**001:CA11:0001:1234:5678:9ABC:DEF0** นำ 100 (128-28) บิตหลังมาจาก r4

จาก r6 จะได้ **2345:00C1:CA11:0001:1234:5678:9ABC:DEF0** เป็นแอดเดรสที่สมบูรณ์เพราะค่า Prefix Length เท่ากับ 0

จากตัวอย่างข้างต้นจะเป็นการใช้ความสามารถในการเปลี่ยนแปลงแอดเดรส เช่นไซต์ X ต้องการเปลี่ยนจากผู้ให้บริการ A ไปเป็นผู้ให้บริการอื่นก็สามารถทำได้โดยง่าย แต่จะมีปัญหาเมื่อดีเอ็นเอส ของผู้ให้บริการมีปัญหา (เช่น ไม่สามารถใช้บริการได้) แต่ถ้าทำตามแบบปกติคือ

N A6 0 2345:00C1:CA11:0001:1234:5678:9ABC:DEF0

A6 0 2345:00D2:DA11:0001:1234:5678:9ABC:DEF0

A6 0 2345:000E:EB22:0001:1234:5678:9ABC:DEF0

ก็จะมีปัญหาเมื่อต้องการเปลี่ยนแปลงผู้ให้บริการ

ตัวอย่างการแปลงจากชื่อไปเป็นแอดเดรส

จากตัวอย่างข้างต้น IPv6.ARPA จะเพิ่มเรคคอร์ดสำหรับ ALPHA-TLA.ORG ดังนี้

\[x234500/24] DNAME IP6.ALPHA-TLA.ORG.

ในระดับ TLA สำหรับผู้ให้บริการ C,D,E ก็จะเพิ่มเรคคอร์ด

\[xC/4].IP6.ALPHA-TLA.ORG. DNAME IP6.C.NET.

\[xD/4].IP6.ALPHA-TLA.ORG. DNAME IP6.D.NET.

\[x0E/8].IP6.ALPHA-TLA.ORG. DNAME IP6.E.NET.

ในผู้ให้บริการ A ก็จะเพิ่มเรคคอร์ด

\[x1CA/12].IP6.C.NET. DNAME IP6.A.NET.

\[x2DA/12].IP6.D.NET. DNAME IP6.A.NET.

\[xEB/8].IP6.E.NET. DNAME IP6.B.NET.

\[x11/8].IP6.A.NET. DNAME IP6.X.EXAMPLE.

\[x22/8].IP6.B.NET. DNAME IP6.X.EXAMPLE.

ในไซต์ X ก็จะเพิ่ม

\[x0001/16] DNAME SUBNET-1

\[x123456789ABCDEF0].SUBNET-1 PTR N.X.EXAMPLE.

ดังนั้นถ้าดีอ นเอสรีโซว์เวอร์ ต้องการหาชื่อจากแอดเดรส

2345:00C1:CA11:0001:1234:5678:9ABC:DEF0

ถาม Server สำหรับ IPv6.ARPA

QNAME=\[x234500C1CA110001123456789ABCDEF0/128].IP6.ARPA.

จะได้คำตอบ

\[x234500/24].IP6.ARPA. DNAME IP6.ALPHA-TLA.ORG.

ถาม Server สำหรับ IPv6.ALPHA-TLA.ORG

QNAME=\[xC1CA110001123456789ABCDEF0/104].IP6.ALPHA-TLA.ORG.

จะได้คำตอบ

\[xC/4].IP6.ALPHA-TLA.ORG. DNAME IP6.C.NET.

ถาม Server สำหรับ IP6.C.NET

QNAME=\[x1CA110001123456789ABCDEF0/100].IP6.C.NET.

จะได้คำตอบ

\[x1CA/12].IP6.C.NET. DNAME IP6.A.NET.

ถาม Server สำหรับ IP6.A.NET

QNAME=\[x110001123456789ABCDEF0/88].IP6.A.NET.

จะได้คำตอบ

\[x11/8].IP6.A.NET. DNAME IP6.X.EXAMPLE.

\[x1CA/12].IP6.C.NET. DNAME IP6.A.NET.

\[x2DA/12].IP6.D.NET. DNAME IP6.A.NET.

\[xEB/8].IP6.E.NET. DNAME IP6.B.NET.

\[x11/8].IP6.A.NET. DNAME IP6.X.EXAMPLE.

\[x22/8].IP6.B.NET. DNAME IP6.X.EXAMPLE.

ในไซต์ X ก็จะเพิ่ม

\[x0001/16] DNAME SUBNET-1

\[x123456789ABCDEF0].SUBNET-1 PTR N.X.EXAMPLE.

ดังนั้นถ้าดือ นเอสรีโซว์เวอร์ ต้องการหาชื่อจากแอดเดรส

2345:00C1:CA11:0001:1234:5678:9ABC:DEF0

ถาม Server สำหรับ IPv6.ARPA

QNAME=\[x234500C1CA110001123456789ABCDEF0/128].IP6.ARPA.

จะได้คำตอบ

\[x234500/24].IP6.ARPA. DNAME IP6.ALPHA-TLA.ORG.

ถาม Server สำหรับ IPv6.ALPHA-TLA.ORG

QNAME=\[xC1CA110001123456789ABCDEF0/104].IP6.ALPHA-TLA.ORG.

จะได้คำตอบ

\[xC/4].IP6.ALPHA-TLA.ORG. DNAME IP6.C.NET.

ถาม Server สำหรับ IP6.C.NET

QNAME=\[x1CA110001123456789ABCDEF0/100].IP6.C.NET.

จะได้คำตอบ

\[x1CA/12].IP6.C.NET. DNAME IP6.A.NET.

ถาม Server สำหรับ IP6.A.NET

QNAME=\[x110001123456789ABCDEF0/88].IP6.A.NET.

จะได้คำตอบ

\[x11/8].IP6.A.NET. DNAME IP6.X.EXAMPLE.

\[x1CA/12].IP6.C.NET. DNAME IP6.A.NET.

\[x2DA/12].IP6.D.NET. DNAME IP6.A.NET.

\[xEB/8].IP6.E.NET. DNAME IP6.B.NET.

\[x11/8].IP6.A.NET. DNAME IP6.X.EXAMPLE.

\[x22/8].IP6.B.NET. DNAME IP6.X.EXAMPLE.

ในไซต์ X ก็จะเพิ่ม

\[x0001/16] DNAME SUBNET-1

\[x123456789ABCDEF0].SUBNET-1 PTR N.X.EXAMPLE.

ดังนั้นถ้าคือ นเอสรีโซว์เวอร์ ต้องการหาชื่อจากแอดเดรส

2345:00C1:CA11:0001:1234:5678:9ABC:DEF0

ถาม Server สำหรับ IPv6.ARPA

QNAME=\[x234500C1CA110001123456789ABCDEF0/128].IP6.ARPA.

จะได้คำตอบ

\[x234500/24].IP6.ARPA. DNAME IP6.ALPHA-TLA.ORG.

ถาม Server สำหรับ IPv6.ALPHA-TLA.ORG

QNAME=\[xC1CA110001123456789ABCDEF0/104].IP6.ALPHA-TLA.ORG.

จะได้คำตอบ

\[xC/4].IP6.ALPHA-TLA.ORG. DNAME IP6.C.NET.

ถาม Server สำหรับ IP6.C.NET

QNAME=\[x1CA110001123456789ABCDEF0/100].IP6.C.NET.

จะได้คำตอบ

\[x1CA/12].IP6.C.NET. DNAME IP6.A.NET.

ถาม Server สำหรับ IP6.A.NET

QNAME=\[x110001123456789ABCDEF0/88].IP6.A.NET.

จะได้คำตอบ

\[x11/8].IP6.A.NET. DNAME IP6.X.EXAMPLE.

ถาม Server สำหรับ IP6.X.EXAMPLE

QNAME=\[x0001123456789ABCDEF0/80].IP6.X.EXAMPLE.

จะได้คำตอบ

\[x0001/16].IP6.X.EXAMPLE. DNAME SUBNET-1.IP6.X.EXAMPLE.

\[x123456789ABCDEF0/64].SUBNET-1.X.EXAMPLE. **PTR** N.X.EXAMPLE.

เรคคอร์ดแบบ DNAME ที่ได้มานั้นสามารถนำไปเก็บในหน่วยความจำเพื่อความสะดวกในการถาม แอดเดรสที่ใกล้เคียงกันครั้งต่อไป

บทที่ 10
การปรับเปลี่ยนสู่ IPv6
(Transition Mechanism)

ความสำเร็จของการเปลี่ยนจาก IPv4 มาเป็น IPv6 นั้นขึ้นอยู่กับความสามารถในการเข้ากันได้ของ IPv6 ในโครงสร้างเครือข่ายขนาดใหญ่ของ IPv4 สำหรับโหนด IPv6 ต่างๆ ที่ต้องการติดต่อกับ IPv4 และต้องการใช้โครงสร้างเครือข่ายของ IPv4 ต้องมีกรรมวิธี ดังนี้

- Dual IP layer วิธีการนี้จะสนับสนุนทั้ง IPv4 และ IPv6 แบบสมบูรณ์ทั้งในโฮสต์และเราเตอร์
- IPv6-Over-IPv4 Tunneling คือการใส่ส่วนหัวของ IPv6 ด้วยส่วนหัวของ IPv4 เพื่อให้สามารถส่งแพ็กเก็ตผ่านไปในโครงสร้างของ IPv4 ได้ โดยวิธีการทันเนิลมีอยู่สองแบบ คือ แบบคอนฟิกทันเนิล (Configured Tunneling) และแบบออโตเมติกทันเนิล (Automatic Tunneling)

แอดเดรสที่ใช้จะมีอยู่สองแบบ ดังนี้

IPv4-compatible Address : คือแอดเดรสที่จัดให้กับโหนดที่ใช้งานได้ทั้ง IPv4 และ IPv6 โดยจะมีการใช้ค่าพรีฟิกซ์ขนาด 96 บิตคือ 0:0:0:0:0:0 และต่อท้ายด้วยแอดเดรสแบบ IPv4 อีก 32 บิต แอดเดรสแบบนี้จะถูกใช้โดยวิธี ออโตเมติกทันเนิลดังรูปที่

IPv6-native Address : คือแอดเดรสใดๆ ที่มีค่าพรีฟิกซ์นอกเหนือจาก 0:0:0:0:0:0

รูปที่ 10.1 แสดง IPv4-compatible Address

10.1 ดูอัลไอพีเลเยอร์ (Dual IP Layer)

วิธีที่ง่ายที่สุดของโหนดแบบ IPv6 ที่จะสามารถติดต่อกับโหนดแบบ IPv4 คือการติดตั้ง IPv4 Implementation แบบสมบูรณ์ลงไปในโหนดแบบ IPv6 คล้ายกับการใช้ IPv4 ควบคู่กับโพรโตคอลตัวอื่น เช่น IPX , NetBIOS , AppleTalk , DECnet โดย IPv6 ก็ถือเป็นโพรโตคอลเพิ่มเติมทำให้ได้โหนดที่มีความสามารถในการติดต่อทั้ง IPv6 และ IPv4

โหนดที่มีความสามารถในการติดต่อทั้ง IPv6 และ IPv4 จะถูกเรียกว่า โหนดแบบ IPv6/IPv4 โหนดนี้จะมีความสามารถในการส่งและรับทั้งแพ็กเก็ตแบบ IPv6 และ IPv4

ในการติดต่อกับโหนดแบบ IPv4 โหนดนี้ก็จะใช้แอดเดรสแบบ IPv4 และในการติดต่อกับโหนดแบบ IPv6 โหนดนี้ก็จะใช้แอดเดรสแบบ IPv6

Dual IP Layer อาจจะถูกใช้ในจุดเชื่อมต่อของโหนดที่ใช้วิธี IPv6-over-IPv4 Tunneling หรือไม่ก็ได้ และโหนดแบบ IPv6/IPv4 ที่สนับสนุนการทำทันเนิลก็อาจจะสนับสนุนคอนฟิกทันเนิลอย่างเดียวหรือสนับสนุนทั้งคอนฟิกทันเนิลและออโตเมติกทันเนิลก็ได้ ดังนั้นโหนดแบบ IPv6/IPv4 จะมีทั้งหมดสามแบบคือ

- โหนดแบบ IPv6/IPv4 ที่ไม่สนับสนุนการทำทันเนิล
- โหนดแบบ IPv6/IPv4 ที่สนับสนุนเฉพาะคอนฟิกทันเนิล
- โหนดแบบ IPv6/IPv4 ที่สนับสนุนทั้งคอนฟิกทันเนิลและออโตเมติกทันเนิล

10.2 การติดตั้งแอดเดรสให้โหนดแบบ IPv6/IP4

เพราะว่าโหนดแบบนี้สนับสนุนทั้ง IPv6 และ IPv4 ดังนั้นโหนดนี้อาจได้รับการติดตั้งแอดเดรสทั้ง แอดเดรสแบบ IPv6 และ IPv4 โดยแอดเดรสทั้งสองแบบอาจจะสัมพันธ์กันหรือไม่ก็ได้

โหนดที่ใช้วิธีออโตเมติกทันเนิล จะใช้แอดเดรสแบบ IPv4-compatible Address จึงเหมือนว่าแอดเดรสแบบเดียวใช้ได้ทั้ง IPv6 และ IPv4 โดยทั้ง 128 บิตใช้สำหรับเป็นแอดเดรสแบบ IPv6 และ 32 บิตหลังใช้สำหรับ IPv4

โหนดแบบ IPv6/IP4 อาจได้รับแอดเดรสมาจากวิธี Stateless Autoconfiguration หรือ Stateful Autoconfiguration วิธีการแบบนี้อาจทำให้ทั้ง IPv4-compatible Address และIPv6-native Address

โหนดแบบ IPv6/IP4 อาจใช้วิธีการของ IPv4 เพื่อรับแอดเดรสแบบ IPv4 และสำหรับโหนดที่สนับสนุน ออโตเมติกทันเนิล ก็อาจใช้แอดเดรสที่ได้รับมานี้มาสร้างเป็น IPv4-compatible Address โดยการเติม 0:0:0:0:0:0 ต่อไปด้านหน้า วิธีการนี้จะเป็นประโยชน์เมื่อในเครือข่ายนั้นเราเตอร์และเครื่องเซิรฟ์เวอร์ที่ทำการติดตั้ง แอดเดรสยังไม่ถูกนำมาใช้งาน

10.3 สเตทฟูลและสเตทเลสออโต้คอนฟิก (Stateful and Stateless Autoconfiguration)

IPv6 กำหนดวิธีการให้แอดเดรสแบบอัตโนมัติไว้ 2 แบบ คือ Stateful และ Stateless Autoconfiguration

ใน Stateless Autoconfiguration ไม่ต้องการการติดตั้งหรือแก้ไขใดๆจากโฮสต์ แต่ต้องการการติดตั้งหรือแก้ไขบางส่วนจากเราเตอร์ และไม่ต้องมีเซิรฟ์เวอร์เพิ่มเติม ในวิธีการนี้จะยอมให้โฮสต์ติดตั้งแอดเดรสของตนเองโดยอาศัยข้อมูลจากตัวเองและข้อมูลที่ได้จากการประกาศของเราเตอร์ (Router Advertisement) เราเตอร์จะประกาศค่าพรีฟิกซ์ เพื่อใช้บอกซับเน็ต ในขณะที่โฮสต์ก็จะกำหนดค่า Interface Identifier ที่มีเพียงหนึ่งเดียวในซับเน็ต ค่าแอดเดรสก็จะได้จากการรวมกันของค่าพรีฟิกซ์ และค่า Interface Identifier แต่ถ้าไม่มีเราเตอร์ปรากฏอยู่โฮสต์ก็ยังสามารถสร้าง Link-Local Address แต่แอดเดรสแบบนี้ใช้ติดต่อได้เฉพาะโหนดที่อยู่ในลิ้งค์เดียวกันเท่านั้น รายละเอียดของแอดเดรสแบบต่างๆ ดูใน บทการจัดการแอดเดรส

ใน Stateful Autoconfiguration โฮสต์สามารถรับค่าแอดเดรสและข้อมูลอื่นๆได้จากเครื่องเซิรฟ์เวอร์เครื่องเซิรฟ์เวอร์นี้จะมีฐานข้อมูลของตนเพื่อใช้ในการแจกจ่ายค่าแอดเดรส

การใช้วิธีแบบ Stateless Autoconfiguration จะใช้เมื่อในเครือข่ายนั้นไม่เจาะจงหรือเข้มงวดเรื่องการกำหนดแอดเดรสตราบใดที่แอดเดรสนั้นๆ มีเพียงหนึ่งเดียวและสามารถหาเส้นทางได้ แต่วิธี Stateful Autuconfiguration จะใช้เมื่อเครือข่ายนั้นต้องการควบคุมการใช้แอดเดรสอย่างเข้มงวด แต่ทั้ง Stateful และ Stateless Autoconfiguration สามารถใช้ร่วมกันได้ กล่าวคือค่าแอดเดรสสามารถรับจาก Stateless Autoconfiguration แต่ข้อมูลอื่นๆ สามารถรับได้จาก Stateful Autoconfiguration ในเครือข่ายใดๆ จะรู้ว่าจะใช้การกำหนดแอดเดรสแบบใด จะรู้ได้จากข้อความ Router Advertisement

ในการกำหนดแอดเดรสทั้งสองแบบนั้น แอดเดรสที่ได้อาจจะไม่ได้ถาวรถ้ามีการกำหนดระยะเวลาการใช้แอดเดรสมาด้วย และเมื่อหมดเวลาค่าแอดเดรสนั้นอาจจะถูกกำหนดให้กับอินเทอร์เฟซอื่นและโฮสต์อาจจะได้รับค่าแอดเดรสอื่นมาแทน ในการจัดการเรื่องการหมดเวลานี้จึงแบ่งการใช้แอดเดรสเป็น 2 ช่วงเวลาคือ “Preferred” คือสามารถใช้การติดต่อทั่วๆไปได้ และ “Deprecated” คือแอดเดรสนี้ไม่สามารถใช้ในการเปิดการติดต่อใหม่ได้ แต่ยังสามารถใช้ได้กับการติดต่อที่ใช้แอดเดรสนี้มาก่อนแล้วและยากที่จะเปลี่ยนค่าแอดเดรสโดยปราศจากการรบกวนการบริการ

เพื่อความแน่ใจว่าแอดเดรสที่จะใช้นั้นมีเพียงหนึ่งเดียว โหนดต้องใช้การตรวจสอบแอดเดรสที่ใช้ซ้ำกัน (Duplicate Address Detection) ก่อนที่จะเริ่มใช้แอดเดรสนั้น โดยการส่งข้อความแบบ Neighbor Solicitation ไปโดยบรรจุค่าแอดเดรสที่ได้รับมาใน Target Address และถ้าค่าแอดเดรสนั้นมีการใช้อยู่ก่อนแล้ว โหนดที่ใช้แอดเดรสนี้มาก่อนก็จะส่งข้อความแบบ Neighbor Advertisement ไปยังโหนดที่ทำการตรวจสอบการใช้แอดเดรสซ้ำกัน เพื่อบอกให้ทราบว่าแอดเดรสถูกใช้อยู่ก่อนแล้ว

เมื่อโหนดได้รับค่าแอดเดรสแล้ว ในขั้นตอนต่อไปคือการกำหนดค่าต่างๆ โดยจะได้รับจากข้อความแบบ Router Advertisement

ดีเอ็นเอส

เรคคอร์ดแบบใหม่ที่ใช้สำหรับแอดเดรสแบบ IPv6 คือ เรคคอร์ดแบบ “A6” ที่สนับสนุนเรคคอร์ดสำหรับแอดเดรสแบบ IPv6 ก่อนหน้านี้คือ “AAAA” ดังนั้นสำหรับโหนดแบบ IPv6/IPv4 ที่ต้องการติดต่อโดยตรงทั้ง IPv4 และ IPv6 โหนดต้องสามารถจัดการกับเรคคอร์ดแบบ A เพื่อติดต่อกับ IPv4 และเรคคอร์ดแบบ A6 และ AAAA เพื่อใช้ติดต่อกับ IPv6

เมื่อดีเอ็นเอสถูกถามถึงแอดเดรสแบบ A6/AAAA ดีเอ็นเอสก็ควรจะส่ง แอดเดรสแบบ IPv6 ตอบกลับไป และเมื่อถูกถามแอดเดรสแบบ A ก็ควรตอบกลับด้วยแอดเดรสแบบ IPv4 เมื่อมีการตอบแอดเดรสกลับมาทั้งแบบ IPv6 และ IPv4 ก็ควรเลือกที่จะใช้ IPv6 ก่อน หรือขึ้นอยู่กับการกำหนดของโหนดแบบ IPv6/IPv4

10.4 IPv6 โอเวอร์ IPv4 ทันเนิล (IPv6-over-IPv4 Tunneling)

การทำทันเนิลใช้เพื่อการติดต่อสื่อสารของ IPv6 โดยผ่านโครงสร้างของ IPv4 โดยวิธีการนี้จะถูกใช้มากที่สุดในช่วงเวลาที่กำลังปรับเปลี่ยนจาก IPv4 ไปเป็น IPv6

โหนดแบบ IPv6/IPv4 สามารถใช้วิธีการทันเนิลโดยการใส่แพ็กเก็ตแบบ IPv6 เข้าไปในแพ็กเก็ตแบบ IPv4 (Encapsulation) โดยการทำทันเนิลสามารถทำได้ในหลายช่วงด้วยกันคือ

- เราเตอร์ถึงเราเตอร์ (Router-to-Router)
- โฮสต์ถึงเราเตอร์ (Host-to-Router)
- โฮสต์ถึงโฮสต์ (Host-to-Host)
- เราเตอร์ถึงโฮสต์ (Router-to-Host)

ในการทำทันเนิลสองแบบแรก แพ็กเก็ตจะถูกส่งไปยังเราเตอร์ เราเตอร์จะเป็นผู้ทำการถอดแพ็กเก็ตนั้นออก (Decapsulation) แล้วส่งแพ็กเก็ตนั้นต่อไปยังโหนดเป้าหมาย การส่งไปยังเราเตอร์ทำให้จุดหมายของ การทำทันเนิลแตกต่างไปจากจุดหมายที่แท้จริงที่แพ็กเก็ตต้องเดินทางไปถึง นั้นคือแพ็กเก็ตแบบ IPv6 ที่ถูกทำ การทำทันเนิลจะไม่ได้ให้แอดเดรสแบบ IPv4 ของเราเตอร์ที่ถอดแพ็กเก็ต ดังนั้นแอดเดรสของเราเตอร์จะต้องรู้ได้จากข้อมูลที่ได้ถูกกำหนดไว้ก่อนหน้าแล้วบนโหนดที่จะการทำทันเนิล วิธีการนี้ก็คือ "คอนฟิกทันเนิล"

ในการทำทันเนิลสองแบบหลัง แพ็กเก็ตจะถูกทำทันเนิลจนถึงโหนดปลายทาง เป้าหมายของโหนดที่ทำการถอดแพ็กเก็ตคือเป้าหมายเดียวกันกับที่บอกไว้ในแพ็กเก็ตตั้งต้น ดังนั้นแอดเดรสที่จะใช้ในการทำทันเนิล สามารถรับรู้ได้จากค่า Destination Address ในแพ็กเก็ตตั้งต้น ถ้าแอดเดรสเป็นแบบ IPv4-compatible Address แอดเดรสที่ใช้สำหรับ การทำทันเนิล คือค่า 32 บิตหลัง วิธีการนี้ค่าแอดเดรสที่ใช้การทำทันเนิลไม่ต้องถูกกำหนดไว้ก่อน ดังนั้นจึงเรียกว่า "ออโตเมติกทันเนิล"

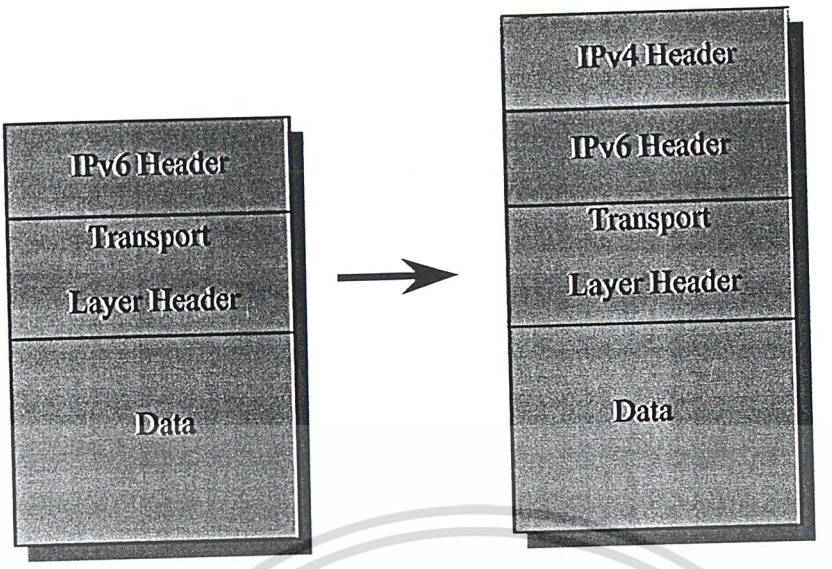
วิธีการทำทันเนิลทั้งสองแบบนี้คือคอนฟิกทันเนิล และออโตเมติกทันเนิล แตกต่างกันที่วิธีการรับค่าแอดเดรสที่จะใช้ในการทำทันเนิล แต่ขั้นตอนที่เหลือจะเหมือนกัน ดังนี้

- โหนดที่ทำทันเนิล สร้างส่วนหัวของ IPv4 ครอบใส่แพ็กเก็ตตั้งต้น แล้วส่งแพ็กเก็ตออกไป
- โหนดที่ถอดแพ็กเก็ต นำส่วนหัวของ IPv4 ออก แล้วปรับปรุงข้อมูลส่วนหัวของ IPv6 เช่นค่า Hop Limit แล้วส่งต่อไปยังโหนดปลายทางในกรณีที่โหนดที่คลายแพ็กเก็ตไม่ใช้โหนดปลายทาง

ในส่วนต่อไปจะอธิบายถึงรายละเอียดของการทำทันเนิลที่ใช้ทั้งสองแบบ

10.5 วิธีการทำทันเนิลโดยทั่วไป (Common Tunneling Mechanisms)

การครอบแพ็กเก็ต (Encapsulation) ของแพ็กเก็ตแสดงได้ดังรูป



รูปที่ 10.2 แสดงการครอบแพ็กเก็ต

นอกเหนือจากการครอบด้วยส่วนหัวของ IPv6 โหนดที่ทำการครอบแพ็กเก็ตต้องจัดการเรื่องต่างๆ ดังนี้

- จัดการเรื่องเมื่อไรต้องทำการแฟรกเมนต์และเมื่อไรต้องส่งข้อความแบบ ICMP Packet Too Big ไปยังโหนดต้นทาง
- จัดการเรื่องการส่งไอซีเอ็มพีสำหรับ IPv4 จากโหนดระหว่างทางที่ดำเนินการทำทันเนิลไปยังโหนดต้นทางด้วยไอซีเอ็มพีสำหรับ IPv6

10.6 การทำทันเนิล MTU และแฟรกเมนต์(Tunnel MTU and Fragmentation)

โหนดที่ครอบแพ็กเก็ตสามารถใช้ค่า MTU เท่ากับ 65515 (65535-20) ไบต์ (20 คือส่วนหัวของ IPv4) โหนดนี้จะส่งข้อความแบบ ICMP Packet Too Big ไปยังโหนดต้นทางเมื่อแพ็กเก็ตมีขนาดเกินค่า MTU แต่วิธีการนี้ไม่มีประสิทธิภาพด้วยเหตุผลสองประการ คือ

1) จะเกิดการแฟรกเมนต์เกินความจำเป็น
2) การแฟรกเมนต์ใดๆที่เกิดในอุโมงค์ต้องถูกรวบรวมกลับเข้าด้วยกันโดยโหนดที่ถอดแพ็กเก็ต สำหรับโหนดที่เป็นเราเตอร์จะต้องการหน่วยความจำเพิ่มเติมส่วนหนึ่งเพื่อทำการรวมแพ็กเก็ตก่อนที่จะส่งแพ็กเก็ตนั้นต่อไป

การแฟรกเมนต์สามารถทำให้ลดลงได้โดยโหนดที่ครอบแพ็กเก็ตสามารถตรวจสอบค่า MTU ของอุโมงค์ที่จะส่งแพ็กเก็ตไปโดยใช้ IPv4 Path MTU Discovery Protocol แล้วเก็บค่า MTU ไว้ ดังนั้น IPv6 Layer สามารถดูอุโมงค์ที่ใช้ส่งข้อมูลเป็นลิ้งเลเยอร์โดยมีค่า MTU ที่ได้จากผลลัพธ์ที่เก็บไว้

อย่างไรก็ตามวิธีนี้ก็ยังไม่สามารถใช้ได้ในกรณีที่ค่า MTU ในอุโมงค์มีขนาดเล็กที่ทำให้ค่า MTU ของแพ็กเก็ตตั้งต้น (แพ็กเก็ต IPv6) มีขนาดน้อยกว่า 1280 ไบต์ (ค่า MTU ใน IPv6 ต้องมีค่าไม่น้อยกว่า 1280 ไบต์) ดังนั้นในกรณีนี้ของ IPv6 ก็จำเป็นต้องใช้ค่า MTU เท่ากับ 576 ไบต์เพื่อสร้างแพ็กเก็ตส่งเข้า

ไปในอุโมงค์ แล้วเมื่อถูกครอบแพ็กเก็ตด้วยส่วนหัวของ IPv4 ก็จะทำให้ขนาดแพ็กเก็ตมีใหญ่กว่าที่โครงสร้างภายในอุโมงค์สามารถรองรับได้ ทำให้ต้องใช้การแฟรกเมนต์เข้าช่วย

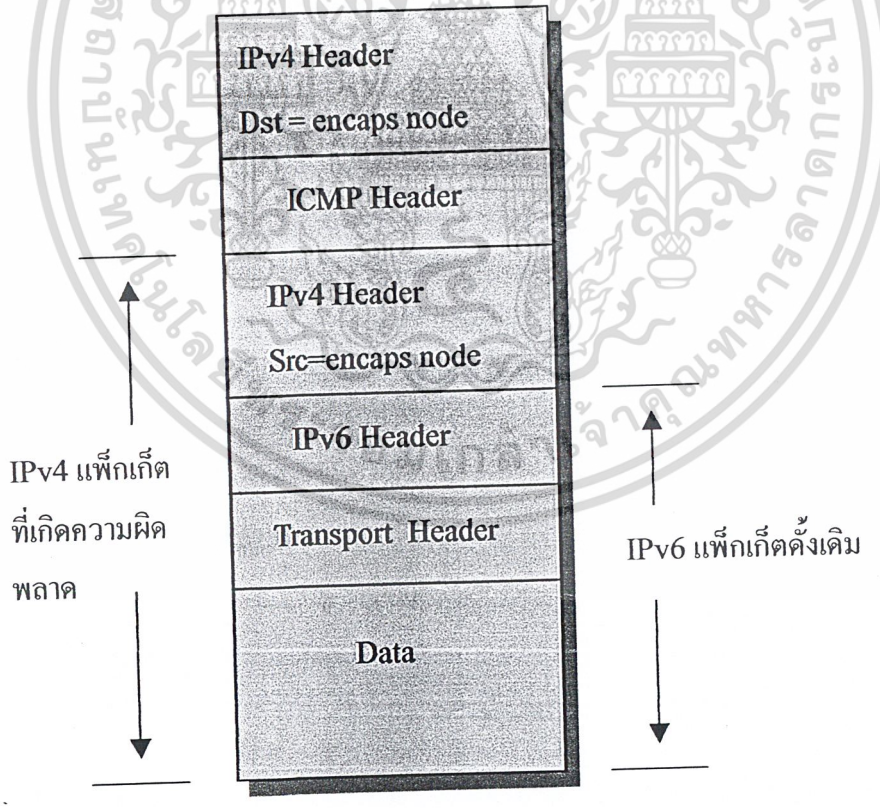
Hop Limit

IPv6-over-IPv4 Tunnel จะถือว่าอยู่ในโมเดลแบบฮอปเดียว นั้นคือค่า Hop Limit จะลดลงไป 1 เมื่อแพ็กเก็ตส่งผ่านอุโมงค์ออกมา เพื่อให้อุโมงค์นั้นมองไม่เห็นโดยผู้ใช้ และทำให้ตรวจจับไม่ได้โดยโปรแกรมตรวจสอบเครือข่าย เช่น Traceroute

IPv4 ICMP Errors

โหนดที่ครอบแพ็กเก็ตอาจได้รับข้อความแบบไอซีเอ็มพีสำหรับ IPv4 จากเราเตอร์ภายในอุโมงค์ ข้อความเหล่านี้จะถูกส่งไปยังโหนดที่ทำการครอบแพ็กเก็ต การจัดการกับข้อความแบบนี้ขึ้นอยู่กับจำนวนข้อมูลที่ได้ใส่เข้ามาในข้อความ

เราเตอร์เก่าแบบ IPv4 จำนวนมากส่งข้อมูลกลับมาเพียง 8 ไบต์ซึ่งไม่เพียงพอที่จะรวมค่า Source Address และค่า Destination Address ของ IPv6 กลับมา ถ้าแพ็กเก็ตที่ส่งกลับมามีข้อมูลให้เพียงพอโหนดที่ทำการครอบแพ็กเก็ตอาจสร้างไอซีเอ็มพีสำหรับ IPv6 เพื่อส่งกลับไปยังโหนดตั้งต้น



รูปที่ 10.3 แสดงข้อความแบบ IPv4 ICMP เพื่อส่งกลับมายังโหนดที่ครอบแพ็กเก็ต

ส่วนหัวของ IPv4 ที่ใช้ครอบแพ็กเก็ต

ค่าต่างๆ ของส่วนหัวของ IPv4 จะถูกตั้งค่าดังนี้

Version : 4

IP Header Length (in 32-bits words): 5 (จะไม่มีออปชั่นของ IPv4 เพิ่มเติม)

Type of Service : 0

Total Length : ขนาดของส่วนหัวของ IPv6 บวกกับข้อมูลใน IPv6 บวกกับส่วนหัวของ IPv4

Identification : กำหนดให้มีค่าไม่ซ้ำกันในการส่งแพ็กเก็ตแต่ละครั้ง

Flags : ถ้าจำเป็นต้องแฟรกเมนต์ ให้ตั้งค่า MF

Fragment Offset : ถ้าไม่มีการแฟรกเมนต์ไม่ต้องตั้งค่าในส่วนนี้

Time to Live : ตั้งตามที่ได้กำหนดไว้ในอิมพิวเมนต์

Protocol : 41 หมายถึงแพ็กเก็ต IPv6 ในส่วนข้อมูลของ IPv4

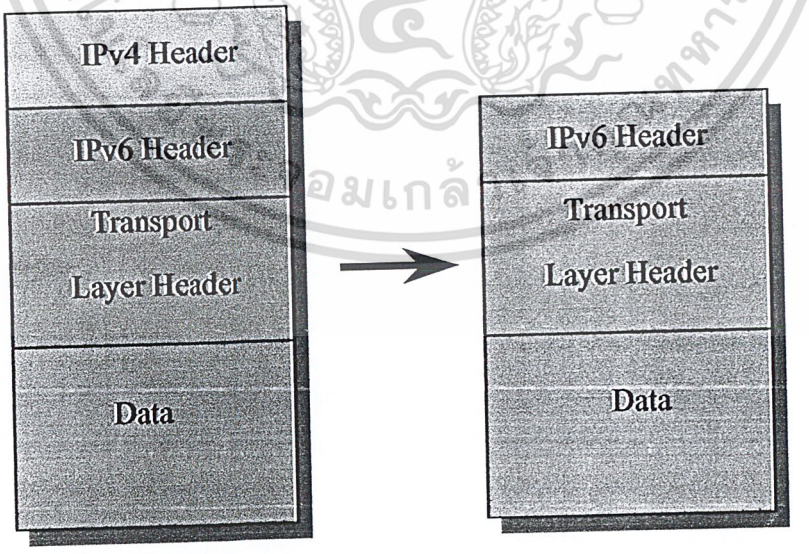
Header Checksum : คือค่าเช็คซัมของส่วนหัวของ IPv4

Source Address : คือค่าแอดเดรสของอินเทอร์เฟซที่ส่งแพ็กเก็ตนี้ออกไปในโหนดที่ทำการครอบแพ็กเก็ต

Destination Address : คือค่าแอดเดรสของโหนดที่อุโมงค์สิ้นสุด

การถอดแพ็กเก็ต IPv6 ที่อยู่ใน IPv4 (Decapsulating IPv6-in-IPv4 Packets)

เมื่อโหนดแบบ IPv6/IPv4 ได้รับแพ็กเก็ต IPv4 ที่มีค่า Protocol เท่ากับ 41 โหนดจะนำส่วนหัวของ IPv4 ออกมา ดังรูป



รูปที่ 10.4 แสดงการถอดแพ็กเก็ต

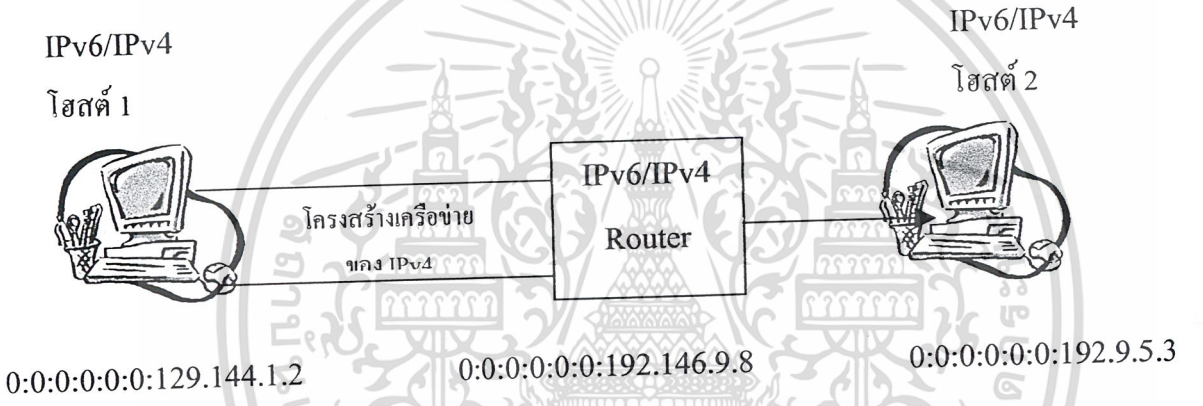
โหนดจะทิ้งส่วนหัวของ IPv4 และเมื่อแพ็กเก็ตที่ถูกถอดส่วนหัวของ IPv4 ออกมานั้นต้องถูกส่งต่อค่า Hop Limit ก็จะถูกลดลงไปหนึ่ง ก่อนที่จะถูกถอดส่วนหัวของ IPv4 ไปนั้นโหนดต้องทำการรวมรวมแพ็กเก็ตให้ครบก่อน ถ้ามีการแฟรกเมนต์เกิดขึ้น

10.7 คอนฟิกทันเนิล

ในวิธีการนี้โหนดที่ทำการครอบแพ็กเก็ตต้องมีข้อมูล (ค่าแอดเดรส) สำหรับโหนดที่อยู่ปลายทางของอุโมงค์ (โหนดที่ทำการถอดส่วนหัวของ IPv4 ออก) การตัดสินใจว่าแพ็กเก็ตไหนต้องทำทันเนิลนั้นส่วนใหญ่จะถูกตัดสินจากข้อมูลของเราติง

โหนดที่ใช้โครงสร้างเครือข่ายของ IPv4 อาจใช้วิธีคอนฟิกทันเนิล เพื่อส่งแพ็กเก็ตไปแบ็คโบนของ IPv6 ถ้าทราบค่าแอดเดรสของเราเตอร์ขอบของ IPv6 แบ็คโบน (Backbone)

ตัวอย่างของ คอนฟิกทันเนิล แสดงได้ดังรูป



รูปที่ 10.5 แสดงตัวอย่างของ คอนฟิกทันเนิล

จากตัวอย่างโฮสต์ 1 ต้องการติดต่อกับโฮสต์ 2 โดยจะถูกส่งผ่านโครงสร้างเครือข่ายของ IPv4 ทำให้ต้องสร้างอุโมงค์แล้วไปยังเราเตอร์ โดยค่าแอดเดรสที่ใช้ทั้งส่วนหัวของ IPv4 และ IPv6 แสดงได้ดังตาราง

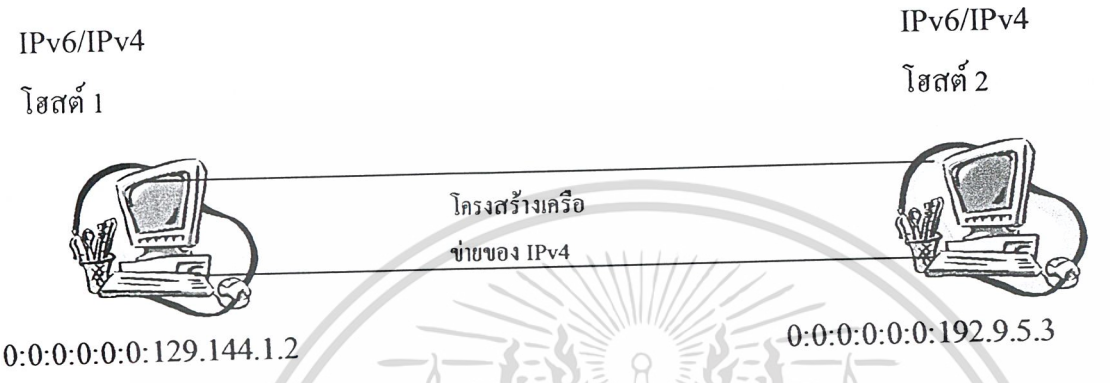
	Address	Header
Source	129.144.1.2	IPv4 Header
Destination	192.146.9.8	
Source	0:0:0:0:0:0:129.144.1.2	IPv6 Header
Destination	0:0:0:0:0:0:192.9.5.3	

ตารางที่ 10.1 แสดงค่าแอดเดรสที่ใช้ในตัวอย่างในรูป 10.5

10.8 ออโตเมติกทันเนิล

ในวิธีการนี้โหนดแบบ IPv6/IPv4 จำเป็นต้องตรวจดูว่าแพ็กเก็ตไหนสามารถใช้ออโตเมติกทันเนิล ได้ โดยดูได้จากค่าพรีฟิกซ์ เท่ากับ 0:0:0:0:0:0 แพ็กเก็ตไหนที่ไม่ได้ใช้ค่าพรีฟิกซ์นี้จะไม่สามารถทำออโตเมติกทันเนิลได้

ตัวอย่างของออโตเมติกทันเนิลแสดงได้ดังรูป



รูปที่ 10.6 แสดงตัวอย่างของออโตเมติกทันเนิล

จากตัวอย่างโฮสต์ 1 ต้องการส่งแพ็กเก็ตไปยังโฮสต์ 2 โดยจะถูกส่งผ่านโครงสร้างเครือข่ายของ IPv4 และค่าแอดเดรสของโฮสต์ 2 เป็นแบบ IPv4-compatible Address โดยค่าแอดเดรสที่ใช้ทั้งส่วนหัวของ IPv4 และ IPv6 แสดงได้ดังตาราง

	Address	Header
Source	129.144.1.2	IPv4 Header
Destination	192.9.5.3	
Source	0:0:0:0:0:0:129.144.1.2	IPv6 Header
Destination	0:0:0:0:0:0:192.9.5.3	

ตารางที่ 10.2 แสดงค่าแอดเดรสที่ใช้ในตัวอย่างในรูป 10.6

สรุป การทำทันเนิล ดูได้จากตาราง

Tunneling Type	Encapsulaing Node	Decapsulating Node	IPv4 Address
Automatic	Source Host	Destination Host	32 บิตล่างของแอดเดรสโหนดเป้าหมาย
Automatic	Router	Destination Host	32 บิตล่างของแอดเดรสโหนดเป้าหมาย
Configured	Source Host	Router	32 บิตล่างของแอดเดรสของเราเตอร์
Configured	Router	Router	32 บิตล่างของแอดเดรสของเราเตอร์

ตารางที่ 10.3 แสดงสรุปของ การทำทันเนิล

10.9 วิธีการส่งแพ็กเก็ต (Sending Algorithm)

การส่งแพ็กเก็ตจากโหนดแบบ IPv6/IPv4 ต้องมีการตัดสินใจว่าเมื่อไรจะส่งแพ็กเก็ตแบบ IPv4 เมื่อไรจะส่งแพ็กเก็ตแบบ IPv6 และเมื่อไรจะใช้คอนฟิกทันเนิลหรือออโตเมติกทันเนิล โดยสามารถทำได้ดังนี้

- ส่งแพ็กเก็ตแบบ IPv4 ไปยังโหนดเป้าหมายที่เป็น IPv4
- ส่งแพ็กเก็ตแบบ IPv6 ไปยังโหนดเป้าหมายที่เป็น IPv6 ที่อยู่ในลิ้งค์เดียวกัน
- ใช้ ออโตเมติกทันเนิล ไปยังโหนดเป้าหมายที่ใช้ IPv4-compatible Address ที่ไม่ได้อยู่ในลิ้งค์เดียวกัน
- ส่งแพ็กเก็ตแบบ IPv6 ไปยังโหนดเป้าหมายที่เป็น IPv6 ที่ไม่ได้อยู่ในลิ้งค์เดียวกัน เมื่อมีเราเตอร์ปรากฏอยู่ในลิ้งค์
- ใช้ คอนฟิกทันเนิล ไปยังโหนดเป้าหมายที่ใช้ IPv6-native Address

โดยสรุปวิธีการส่งดูได้จากตาราง

แอดเดรสของโหนดเป้าหมาย	เป้าหมายอยู่ในลิ้งค์หรือไม่	มี IPv4 เราเตอร์อยู่ในลิ้งค์หรือไม่	มี IPv6 เราเตอร์อยู่ในลิ้งค์หรือไม่	รูปแบบแพ็กเก็ตที่จะถูกส่งออกไป	IPv6 Destination Address	IPv4 Destination Address	Data Link Destination Address
IPv4	Yes	N/A	N/A	IPv4	N/A	E4	EL
IPv4	No	Yes	N/A	IPv4	N/A	E4	RL
IPv4	No	No	N/A	UNRCH	N/A	N/A	N/A
IPv4-compat	Yes	N/A	N/A	IPv6	E6	N/A	EL
IPv4-compat	No	Yes	N/A	IPv6/4	E6	E4	RL
IPv4-compat	No	No	Yes	IPv6	E6	N/A	RL
IPv4-compat	No	No	No	UNRCH	N/A	N/A	N/A
IPv6-Only	Yes	N/A	N/A	IPv6	E6	N/A	EL
IPv6-Only	No	N/A	Yes	IPv6	E6	N/A	RL
IPv6-Only	No	Yes	No	IPv6/4	E6	T4	RL
IPv6-Only	No	No	No	UNRCH	N/A	N/A	N/A

ตารางที่ 10.4 วิธีการส่งแพ็กเก็ต

ค่าในตารางมีความหมายดังนี้

N/A : ไม่มีการใช้งาน

E6 : ค่าแอดเดรสแบบ IPv6 ของโหนดเป้าหมาย

E4 : ค่าแอดเดรสแบบ IPv4 ของโหนดเป้าหมาย

EL : ดาต้าลิ้งค์แอดเดรสของโหนดเป้าหมาย

T4 : ค่าแอดเดรสแบบ IPv4 ของโหนดปลายทางของอุโมงค์

R6 : ค่าแอดเดรสแบบ IPv6 ของเราเตอร์

R4 : ค่าแอดเดรสแบบ IPv4 ของเราเตอร์

RL : ดาต้าลิ้งค์แอดเดรสของเราเตอร์

IPv4 : แพ็กเก็ตแบบ IPv4

IPv6 : แพ็กเก็ตแบบ IPv6

IPv6/4 : แพ็กเก็ตแบบ IPv6 ถูกครอบด้วย IPv4

UNRCH: ไม่สามารถไปถึงเป้าหมายได้

10.10 การติดต่อระหว่างโหนด IPv6 ภายในโดเมนเดียวกันโดยปราศจากอุโมงค์ (Transmission of IPv6 over IPv4 Domains without Explicit Tunnels)

วิธีการนี้เป็นหนึ่งในวิธีการที่ใช้ในช่วงการเปลี่ยนแปลงจาก IPv4 ไป IPv6 โดยวิธีการนี้จะใช้สำหรับโหนดแบบ IPv6 ที่ต้องการติดต่อกันภายในลิ้งค์ของ IPv4 โดยไม่จำเป็นต้องใช้แอดเดรสแบบ IPv4-compatible Address และการใช้คอนฟิกทันเนิล วิธีการนี้เรียกว่า "IPv6 over IPv4" หรือ "6over4"

6over4 จะใช้แอดเดรสแบบ Link-local Address (ค่าพรีฟิกซ์คือ FE80::/64) โดยค่า Interface ID ขนาด 64 บิตคือค่าแอดเดรสแบบ IPv4 ขนาด 32 บิตแล้วต่อด้านหน้าด้วย 0 อีก 32 บิตโดยบิต Universal/Local เท่ากับศูนย์หมายความว่าแอดเดรสนี้ไม่มีค่าเป็นหนึ่งเดียว แสดงได้ดังรูป

FE80:0000:0000:0000	0000:0000	IPv4 Address

รูปที่ 10.7 แสดงแอดเดรสที่ใช้ในวิธี 6over4

10.11 การติดต่อระหว่างโหนด IPv6 โดยผ่านกลุ่มของเครือข่าย IPv4 โดยปราศจากอุโมงค์ที่ชัดแจ้ง (Connection of IPv6 Domains via IPv4 Clounds without Explicit Tunnels)

วิธีการนี้เป็นวิธีการเสริมสำหรับเครือข่ายที่มีแอดเดรสแบบ IPv4 อย่างน้อยหนึ่งแอดเดรส เพื่อให้เครือข่ายแบบ IPv6 ที่ติดต่อกับแวน (WAN) ที่ไม่มีโครงสร้างที่สนับสนุน IPv6 ให้สามารถติดต่อกับเครือข่าย IPv6 อื่นได้โดยมีการปรับแต่งให้น้อยที่สุด โดยจะคิดเหมือนว่าแวน นั้นเป็นลิ้งเลเยอร์เครือข่ายใดๆ ที่ใช้วิธีนี้ไม่จำเป็นต้อง IPv4-compatible Address หรือคอนฟิกทันเนิล เรียกวิธีการนี้ว่า "6to4" และส่วนใหญ่จะถูกติดตั้งไว้ที่เราเตอร์ขอบ

วิธีการนี้จะใช้ค่า FP (Format Prefix) คือ 001 และได้ถูกกำหนด TLA (Top Level Aggregator) ขนาด 13 บิตคือ 0x0002 นั้นคือสามารถแสดงได้ในรูปแบบ Prefix ได้ดังนี้ 2002::/16

และค่า NLA (Next Level Aggregator) ของเครือข่ายใดๆ ที่ใช้วิธีการนี้ก็คือแอดเดรสแบบ IPv4 ของเครือข่าย โดยแสดงได้ดังรูป

FP=001 (3บิต)	TLA =0x0002 (13 บิต)	แอดเดรสแบบ IPv4 (32 บิต)	SLA ID (16 บิต)	Interface ID (64 บิต)

รูปที่ 10.8 แสดงการใช้แอดเดรสของ วิธี 6to4

IPv6 แพ็กเก็ตจากเครือข่าย 6to4 จะถูกครอบด้วย IPv4 แพ็กเก็ตเมื่อแพ็กเก็ตเดินทางออกจากเครือข่ายผ่านเครือข่าย IPv4 และค่า Source และ Destinaiton Address ในส่วนหัวของ IPv4 คือค่าแอดเดรสแบบ IPv4 ที่ได้รับมาจากแอดเดรสแบบ IPv6 ค่า Protocol ส่วนหัวของ IPv4 แสดงค่าเท่ากับ 41

10.12 ตัวอย่างการใช้วิธี 6to4

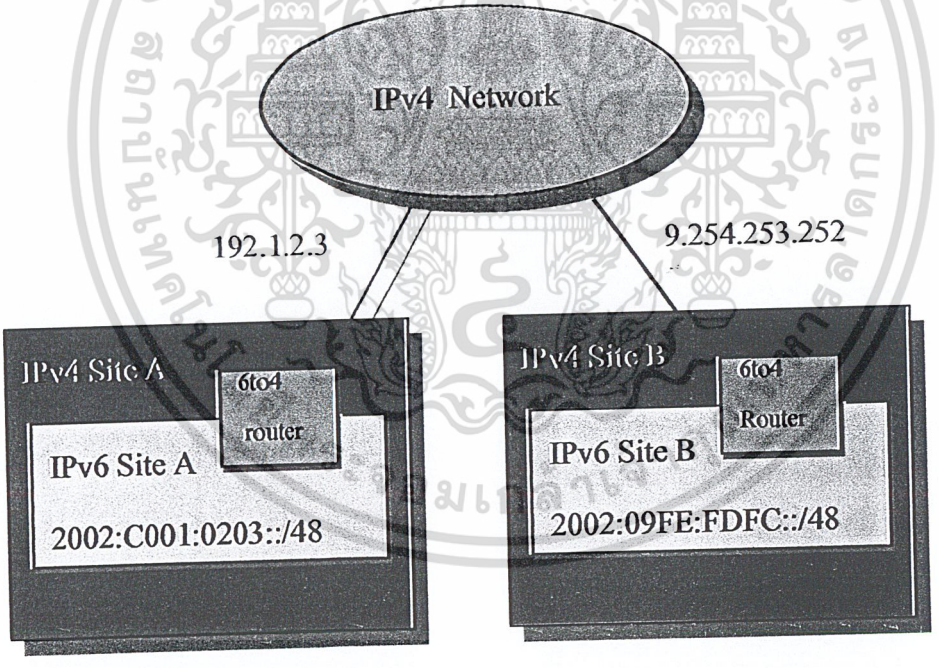
10.12.1 ตัวอย่างที่ 1 ในกรณีที่ทุกเครือข่ายทำงานเหมือนกัน

เป็นตัวอย่างการใช้วิธี 6to4 ที่ง่ายที่สุด โดยแต่ละเครือข่ายต้องมีการเชื่อมต่อกับเครือข่าย IPv4 อย่างน้อยหนึ่งทาง ความต้องการมีเพียงแค่แต่ละเครือข่ายแบบ IPv6 สามารถส่งแพ็กเก็ตแบบ IPv4 ที่มีค่า Protocol เท่ากับ 41 ไปยังเครือข่ายอื่น ตามนิยามเมื่อแต่ละเครือข่ายมีการใช้แอดเดรสที่มีค่าพรีฟิกซ์ ที่ได้กล่าวมาแล้วข้างต้น ดีเอ็นเอส ของแต่ละเครือข่ายจะสร้างบันทึกสำหรับแอดเดรสเหล่านั้น

สมมติว่า ไซต์ A มีแอดเดรสแบบ IPv4 คือ 192.1.2.3 แล้ว ดีเอ็นเอส จะสร้างบันทึกที่มีค่าพรีฟิกซ์ คือ {FP=001,TLA=0x0002,NLS=192.1.2.3}/48

ไซต์ B มีแอดเดรสแบบ IPv4 คือ 9.254.253.252 แล้วดีเอ็นเอส จะสร้างบันทึกที่มีค่าพรีฟิกซ์ คือ {FP=001,TLA=0x0002,NLS=9.254.253.252}/48

เมื่อโฮสต์ของไซต์ B สอบถามแอดเดรสของโฮสต์ของไซต์ A โฮสต์ของไซต์ B จะได้แอดเดรสที่มีค่า Prefix {FP=001,TLA=0x0002,NLS=192.1.2.3}/48 ตอบกลับมา เมื่อได้แอดเดรสตอบกลับมาแล้ว โฮสต์ของไซต์ B ก็จะสร้างแพ็กเก็ตแบบ IPv6 ที่ครอบด้วยส่วนหัวของ IPv4 แล้วส่งออกไปตามปกติ ดังรูป



รูปที่ 10.9 แสดงการติดต่อระหว่างโฮสต์ไซต์ A และโฮสต์ของไซต์ B

ภายในเครือข่ายแบบ 6to4 แพ็กเก็ตที่มีค่าพรีฟิกซ์ 2002::/16 จะถูกส่งไปยังเราเตอร์ขอบแบบ 6to4 สำหรับเราเตอร์มีอย่างเดียวที่ต้องเปลี่ยนแปลงคือเมื่อมีแพ็กเก็ตที่มีค่า Destination Addrss มีค่าพรีฟิกซ์เท่ากับ 2002::/16 เราเตอร์ต้องสร้างแพ็กเก็ตแบบ IPv4 ครอบโดยจะรับค่าแอดเดรสแบบ IPv4 มาจากแอดเดรสแบบ IPv6

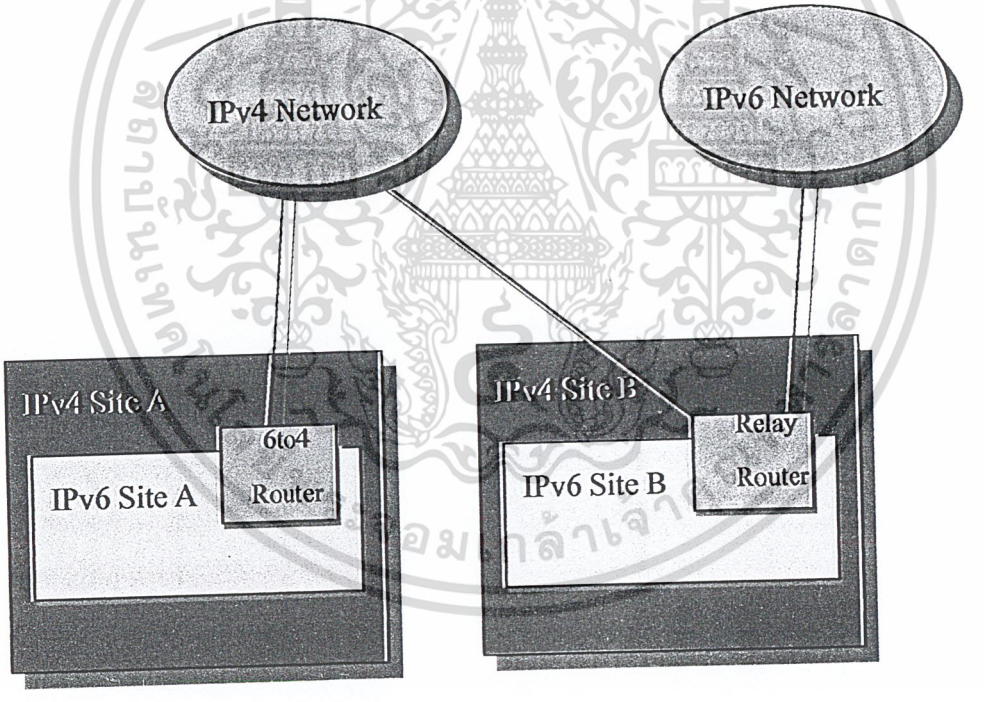
10.12.2 ตัวอย่างที่ 2 ในกรณีที่มีบางเครือข่ายติดต่อกับโครงสร้าง IPv6

รูปแบบนี้จะเกิดในช่วงการเปลี่ยนจาก IPv4 เป็น IPv6 คือ บางไซต์แบบ IPv6 มีการติดต่อกับเครือข่าย IPv4 เพียงอย่างเดียว ในขณะที่บางส่วนเป็นกลุ่มของไซต์แบบ IPv6 ธรรมดา (เครือข่ายใน 6bone) ที่มีการเชื่อมต่อกัน ไซต์ทั้งสองแบบต้องการติดต่อซึ่งกันและกัน

เราเตอร์ที่มีความสามารถทั้งการติดต่อแบบ 6to4 และแบบ IPv6 ธรรมดาในหัวข้อนี้จะเรียกว่า รีเลย์เราเตอร์ (Relay Router) เพื่อแบ่งแยกระหว่างเราเตอร์ที่สนับสนุนเฉพาะวิธี 6to4 และเราเตอร์ที่สนับสนุนเฉพาะแบบ IPv6 ธรรมดา

ในรูปแบบนี้จำเป็นต้องมีเราเตอร์อย่างน้อยหนึ่งตัวที่ทำหน้าที่เป็นรีเลย์เราเตอร์ระหว่างกลุ่มของไซต์แบบ 6to4 และกลุ่มไซต์แบบ IPv6 ธรรมดา โดยรีเลย์เราเตอร์ก็ไม่ต่างกับเราเตอร์ทั่วๆไปเพียงแค่มีอย่างน้อยอินเทอร์เฟซหนึ่งสนับสนุนแบบ 6to4 และอย่างน้อยอีกอินเทอร์เฟซหนึ่งเป็นแบบ IPv6 ธรรมดา

จากรูป สมมติว่าโฮสต์ของไซต์ A ต้องการติดต่อกับกลุ่มของไซต์แบบ IPv6 ธรรมดาต้องติดต่อผ่านรีเลย์เราเตอร์ของไซต์ B (โดยรีเลย์เราเตอร์อาจเป็นของ ISP ที่ให้บริการการเชื่อมต่อกับกลุ่มของไซต์แบบ IPv6 หรือเป็นของไซต์ที่เปิดให้บริการ)



รูปที่ 10.10 แสดงการติดต่อจากไซต์ A ไปยังกลุ่มของไซต์แบบ IPv6 โดยผ่านรีเลย์เราเตอร์ของไซต์ B

ดังนั้นในการเปลี่ยนจาก IPv4 ไปยัง IPv6 โดยผ่านวิธี 6to4 อาจทำได้โดย

1. เริ่มใช้ IPv6 ภายในเครือข่ายโดยอาจใช้วิธี IPv6 แบบธรรมดา แบบ 6over4 หรือแบบการใช้อุโมงค์
2. ปรับแต่งเราเตอร์ขอบให้เป็นสนับสนุนวิธี 6to4

3. ระบุรีเลย์เราเตอร์ที่เต็มใจส่งผ่านแพ็กเก็ตไปสู่กลุ่มของ IPv6
4. เมื่อมีการติดต่อกับกลุ่มของ IPv6 ได้โดยตรงแล้ว อาจหยุดการใช้งาน 6to4

ถ้ามีการใช้ NAT (Network Address Translator) อยู่ในเครือข่าย วิธีการข้างต้นนี้ (6to4) ก็ยังสามารถใช้ได้เพราะแอดเดรสแบบ IPv4 ที่ใช้ก็ยังเป็นแอดเดรสที่มีเพียงหนึ่งเดียวทำให้เกิดแอดเดรสแบบ IPv6 เพียงหนึ่งเดียวเพียงแต่แอดเดรสแบบ IPv6 ที่ใช้กันในเครือข่ายจะต่างกับแอดเดรสที่ส่งออกไปนอกเครือข่าย

บทที่ 11

การทดลอง

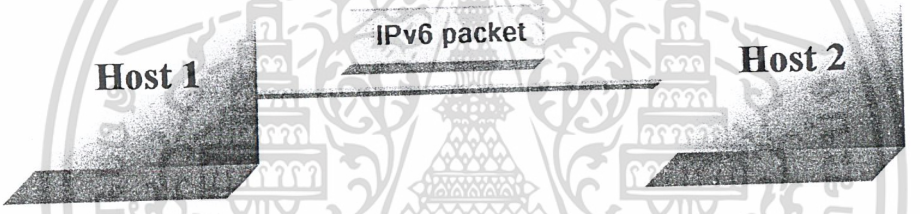
11.1 การทดลองที่ 1 : การติดต่อภายในเครือข่ายเดียวกัน

จุดประสงค์

ปรับเปลี่ยนโฮสต์ใดๆ ที่ใช้ IPv4 อย่างเดียวให้สามารถใช้ได้ทั้ง IPv4 และ IPv6 และทดสอบการติดต่อกันภายในเครือข่ายเดียวกัน

สภาพแวดล้อม

โฮสต์เดี่ยวเชื่อมต่อกันด้วย Ethernet ภายในซับเน็ตเดียวกัน



รูปที่ 11.1.1 สภาพแวดล้อมการเชื่อมต่อสำหรับทุกโฮสต์ในการทดลองที่ 1

การส่งผ่านแพ็กเก็ตระหว่างกันไม่จำเป็นต้องอาศัยการทำทันเนิล สามารถส่งแพ็กเก็ตของ IPv6 โดยตรงถึงกันได้เลยโดยใช้แอ็ดเดรสแบบลิ้งค์-โลคัลแอ็ดเดรสในการติดต่อกัน

การทดลอง

11.1.1 การทดลองที่ 1.1 ทดสอบการเชื่อมต่อระหว่างโหนด IPv6 ในระบบปฏิบัติที่ต่างกันภายในเครือข่ายเดียวกัน ว่าสามารถเชื่อมต่อกันได้หรือไม่

สมมติฐาน การทำงานของ IPv6 โพรโตคอลจะไม่ขึ้นกับระบบปฏิบัติการจึงสามารถติดต่อและทำงานร่วมกันได้

ตัวแปรต้น ระบบปฏิบัติการ Window 2000 , Linux Redhat 7.2 , Windows XP

ตัวแปรตาม ผลการติดต่อ

สภาพแวดล้อม

โฮสต์ 1

- ระบบปฏิบัติการ Windows XP
- IPv4 แอ็ดเดรส 161.246.5.111
- หมายเลข อินเทอร์เฟซ 00-E0-7D-03-29-A1

โฮสต์ 2

- ระบบปฏิบัติการ Windows 2000
- IPv4 แอ็ดเดรส 161.246.5.113
- หมายเลข อินเทอร์เฟซ 00-0E-29-58-FA-31

โฮสต์ 3

- ระบบปฏิบัติการ Linux Redhat 7.2
- IPv4 แอ็ดเดรส 161.246.5.19
- หมายเลข อินเทอร์เฟซ 00:04:E2:0D:FF:4F

วิธีทดลอง

1. ติดตั้ง IPv6 ให้กับโฮสต์ทดลองที่ 1 ที่ติดตั้งระบบปฏิบัติการ Window XP โดยใช้คำสั่ง

   ipv6 install

   โฮสต์ 1 จะมี IPv6 แอ็ดเดรสเป็น FE80::2E0:7DFF:FE03:29A1

   ติดตั้ง IPv6 ให้กับโฮสต์ทดลองที่ 2 ที่ติดตั้งระบบปฏิบัติการ Window 2000

   โฮสต์ 2 จะมี IPv6 แอ็ดเดรสเป็น FE80::20E:29FF:FE58:FA31

   ติดตั้ง IPv6 ให้กับโฮสต์ 3 ที่ติดตั้งระบบปฏิบัติการ Linux Redhat 7.2 โดยเพิ่มคำสั่งดังต่อไปนี้

   ในไฟล์ /etc/sysconfig/network

   NETWOKING_IPV6=yes

   ในไฟล์ /sys/sysconfig/network-scripts/ifcfg-eth0

IPV6INIT="yes"

ใช้คำสั่งต่อไปนี้

route —A inet6 add ::/0 dev eth0

โฮสต์ 3 จะมี IPv6 แอ็ดเดรสเป็น FE80::A1F6:513

2. เมื่อโอสต์ทั้ง 3 พร้อมใช้งาน IPv6 แล้วจะทำการทดสอบโดยการ ping ติดต่อระหว่างกัน โดยการทำงานนั้นเนื่องจากเป็นการติดต่อภายใต้เครือข่ายเดียวกันแพ็กเก็ตจะถูกส่งออกมาจากโฮสต์โดยไม่ต้องผ่านการเอ็นแค็ปและแอ็ดเดรสที่ใช้จะเป็นแบบลิ้งค์-โลคัลแอ็ดเดรส และถูกส่งต่อถึงกันโดยผ่านอีเทอร์เน็ต
3. ได้ผลสรุปการทดลองดังตาราง

ระบบปฏิบัติการโฮสต์ต้นทาง	ระบบปฏิบัติการโฮสต์ปลายทาง	ผลการติดต่อ
Window XP	Window 2000	สามารถติดต่อได้
Window XP	Linux Redhat 7.2	สามารถติดต่อได้
Window 2000	Window XP	สามารถติดต่อได้
Window 2000	Linux Redhat 7.2	สามารถติดต่อได้
Linux Redhat 7.2	Window XP	สามารถติดต่อได้
Linux Redhat 7.2	Window 2000	สามารถติดต่อได้

ตารางที่ 11.1.1 ผลการติดต่อระหว่างโฮสต์ที่ใช้ระบบปฏิบัติการต่างกันภายในเครือข่ายเดียวกัน

จากตารางแสดงว่าสามารถใช้งาน IPv6 ติดต่อกันภายในเครือข่ายได้แม้จะใช้ระบบปฏิบัติการที่ต่างกัน

ข้อสรุป

ความสามารถในการติดต่อของ IPv6 โพรโตคอลเสมือนกับความสามารถในการติดต่อของ IPv4 โพรโตคอล นั่นคือจะไม่ขึ้นกับระบบปฏิบัติการ

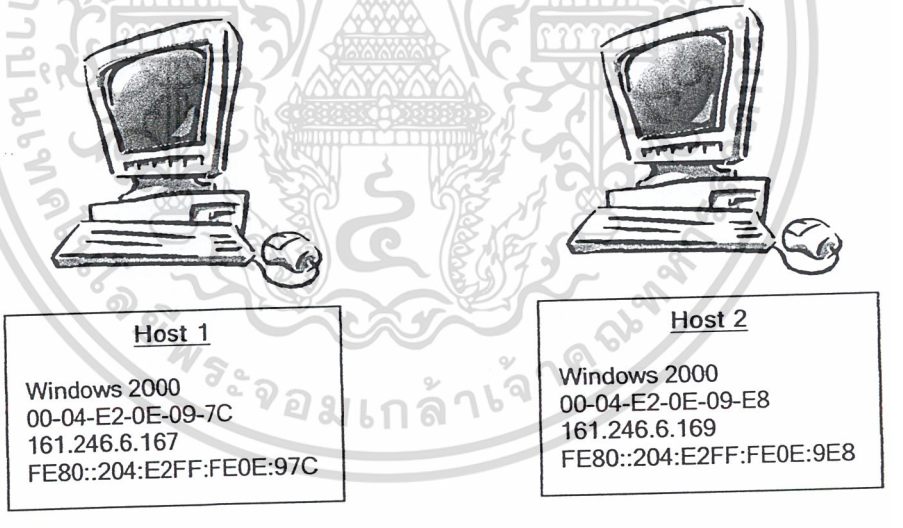
11.1.2 การทดลองที่ 1.2 ทดสอบความสามารถทางด้านความปลอดภัยระหว่าง 2 โหนดภายในเครือข่ายเดียวกัน

คอนฟิกให้มีการใช้ส่วนหัวทางด้านความปลอดภัย

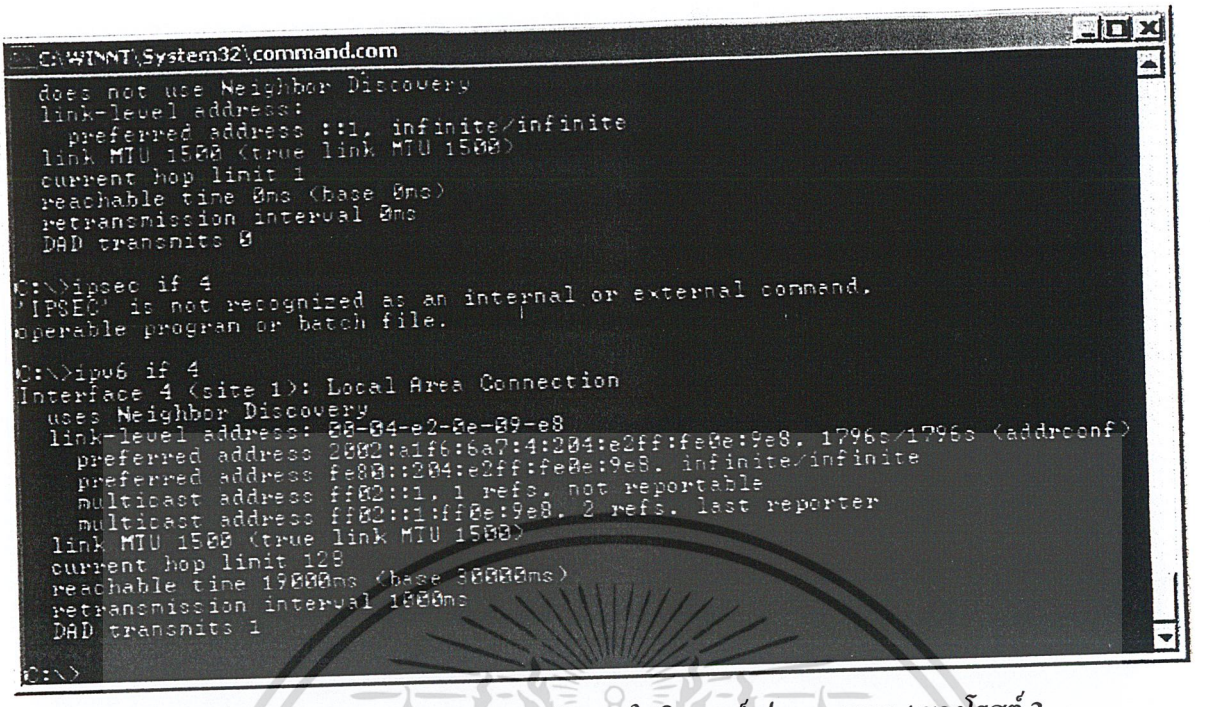
สมมติฐาน สามารถติดต่อและใช้ระบบรักษาความปลอดภัยร่วมกันได้

วิธีทดลอง

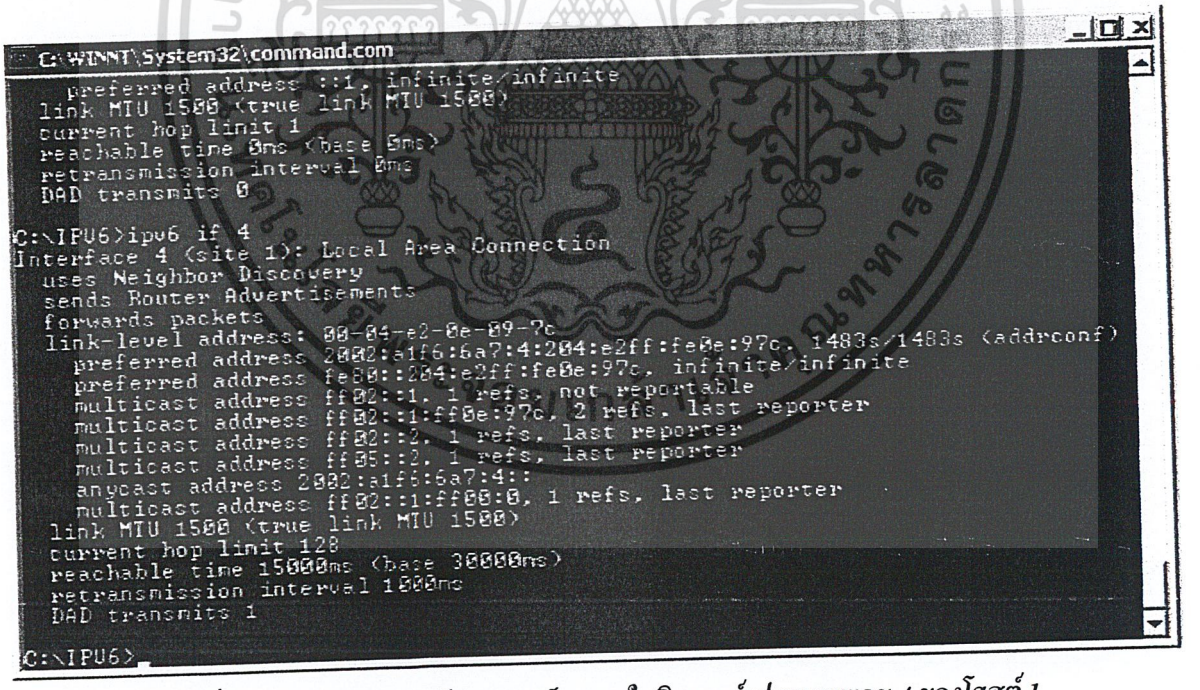
1. ติดตั้ง Microsoft IPv6 dual-stack ให้กับโฮสต์ที่อยู่ภายในเครือข่ายเดียวกัน โดยใช้ระบบปฏิบัติการ Windows 2000 และเชื่อมต่อกันด้วยอีเทอร์เน็ต โดยโฮสต์ 1 มี IPv4 แอ็ดเดรสเป็น 161.246.6.167 และมีหมายเลขของการ์ดอินเทอร์เฟซเป็น 00-04-E2-0E-09-7C จะได้ลิ้งค์โลคัลแอ็ดเดรสสำหรับโฮสต์ 1 เป็น FE80::204:E2FF:FE0E:97C ที่อินเทอร์เฟซหมายเลข 4 สำหรับใช้ติดต่อภายในเครือข่าย และโฮสต์ 2 มี IPv4 แอ็ดเดรสเป็น 161.246.6.169 และมีหมายเลขของการ์ดอินเทอร์เฟซเป็น 00-04-E2-0E-09-E8 จะได้ลิ้งค์โลคัลแอ็ดเดรสสำหรับโฮสต์ 1 เป็น FE80::204:E2FF:FE0E:9E8 ที่อินเทอร์เฟซหมายเลข 4 แสดงดังรูป



รูปที่ 11.1.2 แสดงสภาพแวดล้อมที่ใช้ทดลองด้านความปลอดภัยระหว่างโฮสต์ภายในเครือข่ายเดียวกัน



รูปที่ 11.1.3 แสดงรายละเอียดของแอ็ดเดรสในอินเทอร์เฟซหมายเลข 4 ของโฮสต์ 2



รูปที่ 11.1.4 แสดงรายละเอียดของแอ็ดเดรสในอินเทอร์เฟซหมายเลข 4 ของโฮสต์ 1

2. สร้าง IPSec Security Association (SA) ระหว่างทั้งสองโฮสต์ เพื่อทำการพิสูจน์ตัวจริง (Authentiation) โดยใช้อัลกอริทึ่ม Message Digest 5 (MD5) ร่วมกับ HMAC โดยที่โฮสต์ 1 ทำการสร้างไฟล์ Policy (.SPD) และ Association (.SAD) โดยใช้คำสั่ง

ipsec6 c test

จะได้ไฟล์สำหรับกำหนด Policy ชื่อ test.spd และไฟล์ Association ชื่อ test.sad ใช้โปรแกรมเท็กซ์อีดิเตอร์เปิดไฟล์ test.spd ขึ้นมาและทำการเพิ่มแถวเข้าไปดังตาราง

ชื่อฟิลด์	ค่าตัวอย่าง
Policy	2
RemoteIPAddr	FE80::204:E2FF:FE0E:9E8
LocalIPAddr	- *
RemotePort	- *
Protocol	- *
LocalPort	- *
IPSecProtocol	AH
IPSecMode	TRANSPORT
RemoteGWIPAddr	*
SABundleIndex	NONE
Direction	BIDIRECT
Action	APPLY
InterfaceIndex	0

ตารางที่ 11.1.2 แสดง Policy บนโฮสต์ 1

จากนั้นเปิดไฟล์ test.sad ขึ้นมาและทำการเพิ่ม Association เข้าไปดังตาราง

ชื่อฟิลด์	ค่าตัวอย่าง
SAEntry	2
SPI	3001
SADestIPAddr	FE80::204:E2FF:FE0E:9E8
DestIPAddr	POLICY
SrcIPAddr	POLICY

Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	OUTBOUND
SecPolicyIndex	2

ตารางที่ 11.1.3 แสดง Association ของทราฟฟิคจากโฮสต์ 1 ไป โฮสต์ 2 บน

ชื่อฟิลด์	ค่าตัวอย่าง
SAEntry	1
SPI	3000
SADestIPAddr	FE80::204:E2FF:FE0E:97C
DestIPAddr	POLICY
SrcIPAddr	POLICY
Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	INBOUND
SecPolicyIndex	2

ตารางที่ 11.1.4 แสดง Association ของทราฟฟิคจากโฮสต์ 2 ไป โฮสต์ 1 บน

จากนั้น สร้างไฟล์ที่บรรจุคีย์จากการคอนฟิกกำหนดให้ใช้ไฟล์ชื่อ test.key โดยใช้เท็กซ์อีดิเตอร์สร้างใส่ข้อความลงไป กำหนดให้เป็น

"This is a key"

3. ที่โฮสต์ 2 พิมพ์คำสั่ง

ipsec6 c test

และทำการเพิ่ม Policy และ Association เข้าไปในไฟล์ test.spd และ test.sad ดังนี้

ชื่อฟิลด์	ค่าตัวอย่าง
Policy	2
RemoteIPAddr	FE80::204:E2FF:FE0E:97C
LocalIPAddr	- *
RemotePort	- *
Protocol	- *
LocalPort	- *
IPSecProtocol	AH
IPSecMode	TRANSPORT
RemoteGWIPAddr	*
SABundleIndex	NONE
Direction	BIDIRECT
Action	APPLY
InterfaceIndex	0

ตารางที่ 11.1.5 แสดง Policy บนโฮสต์ 2

ชื่อฟิลด์	ค่าตัวอย่าง
SAEntry	2
SPI	3001
SADestIPAddr	FE80::204:E2FF:FE0E:9E8
DestIPAddr	POLICY
SrcIPAddr	POLICY
Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	INBOUND
SecPolicyIndex	2

ตารางที่ 11.1.6 แสดง Association ของทราฟฟิคจากโฮสต์ 1 ไป โฮสต์ 2 บน

SAD File Field Name	Example Value
SAEntry	1
SPI	3000
SADestIPAddr	FE80::204:E2FF:FE0E:97C
DestIPAddr	POLICY
SrcIPAddr	POLICY
Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	OUTBOUND
SecPolicyIndex	2

ตารางที่ 11.1.7 แสดง Association ของทราฟฟิคจากโฮสต์ 2 ไป โฮสต์ 1 บน

สร้างไฟล์ชื่อ test.key ใช้บรรจุคีย์ดังนี้

"This is a key"

4. ทำการเพิ่ม Policy และ Association เข้าสู่ระบบโดยที่โฮสต์ 1 พิมพ์คำสั่งว่า

   ipsec6 a test

   และที่โฮสต์ 2 พิมพ์คำสั่งว่า

   ipsec6 a test

   เมื่อทำการตรวจดูโดยใช้คำสั่ง ***ipsec6 sp*** และ ***ipsec6 sa*** จะพบ Policy และ Association ที่สร้างมา

5. ทดสอบโดยการ Ping จากโฮสต์ 1 ไปยังโฮสต์ 2 โดยใช้คำสั่ง

   ping6 FE80::204:E2FF:FE0E:9E8 —n 10

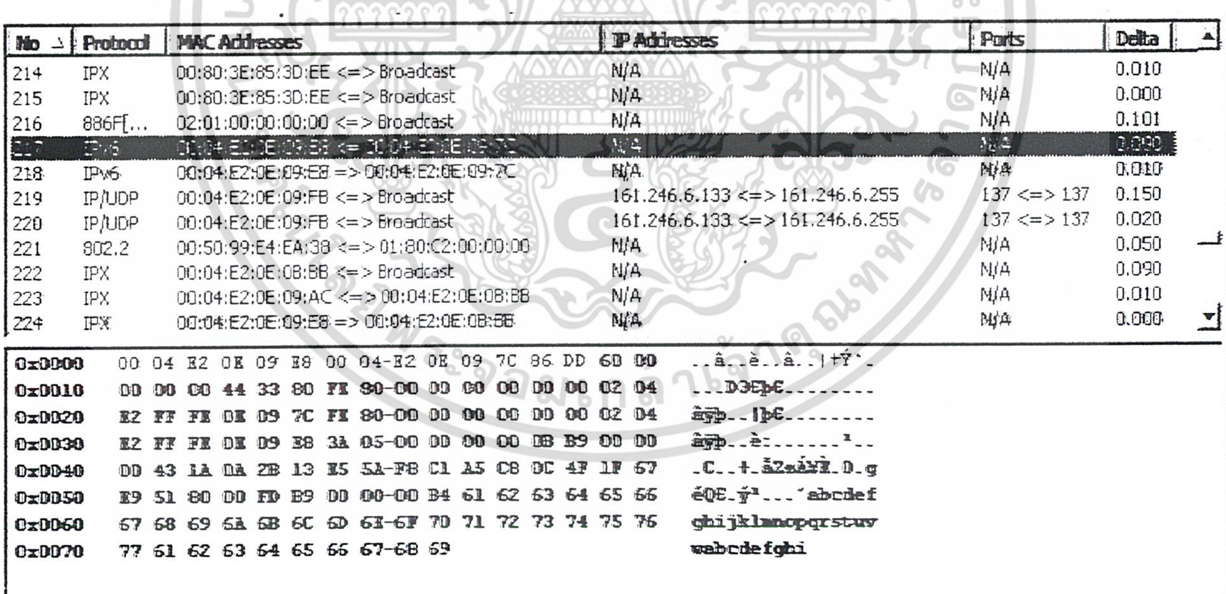
   และตรวจสอบเส้นทางการส่งแพ็กเก็ตโดยใช้คำสั่ง

   tracert6 FE80::204:E2FF:FE0E:9E8

รูปที่ 11.1.5 แสดงการทดสอบโดยการ ping และ tracert

จากภาพที่ 11.1.5 จะเห็นว่าแพ็กเก็ตได้ถูกส่งไปยังโฮสต์ที่ 2 โดยตรงเลย

6. ที่โฮสต์ 2 ใช้โปรแกรมตรวจจับแพ็กเก็ตที่ส่งมาจากโฮสต์ 1



รูปที่ 11.1.6 แสดงแพ็กเก็ตที่เกิดจากการ ping มาจากโฮสต์ 1

จากรูปจะเห็นว่า

- แพ็กเก็ตได้ถูกส่งตรงมาจากโฮสต์ 1 โดยไม่ได้ผ่านการทันเนิลโดย IPv4
- สังเกตค่าในส่วน Next header ในส่วนหัวของ IPv6 แสดงในภาพเป็นไบต์ที่ 0x0014 มีค่าเป็น 33H ซึ่งเป็นส่วนหัวแบบ Authentication

- ส่วน AH จะเริ่มต้นที่ไบต์ที่ 0x0036 ในภาพเป็น 3AH ซึ่งเป็นการบอกว่าส่วนที่อยู่ต่อจากส่วน AH นี้เป็น ICMPv6 เมื่อตรวจสอบค่าในฟิลด์ Security Parameter Index ซึ่งในภาพเป็นไบต์ที่ 0x003C และ 0x003D มีค่าเป็น BB9H เท่ากับ 3001 ในเลขฐานสอง ซึ่งเป็นค่าทราฟฟิกจาก Host 1 ไปยัง Host 2 ที่ได้ทำการกำหนดไว้
- ในส่วน ICMPv6 ไบต์ที่ 0x0052 และ 0x0053 ซึ่งเป็นส่วน Type และ Code มีค่าเป็น 128 และ 0 ตามลำดับ แสดงว่าเป็นแพ็กเก็ตแบบ Echo Request Message ของ ICMPv6

7. เมื่อโฮสต์ 2 ได้รับ Echo Request Message ก็จะตอบกลับไปแสดงดังรูป

No.	Protocol	MAC Addresses	IP Addresses	Ports	Delta
270	IP/UDP	00:04:E2:0E:09:FB <=> Broadcast	161.246.6.133 <=> 161.246.6.255	137 <=> 137	0.070
271	IP/UDP	00:04:E2:0E:09:FB <=> Broadcast	161.246.6.133 <=> 161.246.6.255	137 <=> 137	0.020
272	IPX	00:04:E2:0E:09:FB <=> Broadcast	N/A	N/A	0.080
273	886F[...	02:01:00:00:00:00 <=> Broadcast	N/A	N/A	0.140
274	IPv6	00:04:E2:0E:09:E8 <=> 00:04:E2:0E:09:7C	N/A	N/A	0.131
275	IPv6	00:04:E2:0E:09:E8 <=> 00:04:E2:0E:09:7C	N/A	N/A	0.010
276	802.2	00:50:99:E4:EA:38 <=> 01:80:C2:00:00:00	N/A	N/A	0.180
277	886F[...	02:01:00:00:00:00 <=> Broadcast	N/A	N/A	0.180
278	IP/UDP	00:04:E2:0E:09:FB <=> Broadcast	161.246.6.133 <=> 161.246.6.255	137 <=> 137	0.010
279	IP/UDP	00:04:E2:0E:09:FB <=> Broadcast	161.246.6.133 <=> 161.246.6.255	137 <=> 137	0.020
280	IPv6	00:04:E2:0E:09:E8 => 00:04:E2:0E:09:7C	N/A	N/A	0.060

```
0x0000  00 04 E2 0E 09 7C 00 04-E2 0E 09 E8 86 DD 60 00   ..â..|..â..èt¥`.
0x0010  00 00 00 44 3A 80 FE 80-00 00 00 00 00 00 02 04   ...D:€þ€........
0x0020  E2 FF FE 0E 09 E8 FE 80-00 00 00 00 00 00 02 04   âÿþ..èþ€........
0x0030  E2 FF FE 0E 09 7C 3A D5-00 00 00 00 BB B8 00 00   âÿþ..|:Õ....»¸..
0x0040  00 05 53 C8 EB 79 03 EC-E5 60 15 07 01 DA 29 1C   ..SÈëy.ìå`...Ú).
0x0050  6E 7D B1 00 FC B7 00 00-00 B6 61 62 63 64 65 66   n}±.ü·...¶abcdef
0x0060  67 68 69 6A 6B 6C 6D 6E-6F 70 71 72 73 74 75 76   ghijklmnopqrstuv
0x0070  77 61 62 63 64 65 66 67-68 69                     wabcdefghi
```

รูปที่ 11.1.7 แสดงแพ็กเก็ตที่เกิดจากตอบกลับการ ping เข้าสู่โฮสต์ 1

จากรูปค่าในส่วนที่เป็น Type และ Code ของ ICMPv6 มีค่าเป็น 129 และ 0 ตามลำดับ แสดงว่าเป็นข้อความแบบ Echo Reply Message

ข้อสรุป

เมื่อคอนฟิกโหนดสองโหนดให้ใช้ความสามารถทางด้านความปลอดภัยของ IPv6 แพ็กเก็ตที่จะถูกส่งระหว่างโหนดสองโหนดนั้นจะถูกครอบคลุมด้วยความสามารถด้านความปลอดภัยของ IPv6 โดยการทดลองนี้ ไม่เป็นที่นิยมใช้เพราะเป็นการติดต่อภายในเครือข่ายเดียวกัน และข้อเสียของการใช้ความสามารถด้านความปลอดภัยของ IPv6 ก็คือ ไม่ว่าจะเป็นแพ็กเก็ตที่ต้องการความ ปลอดภัยสูงหรือไม่ ก็จะถูกบังคับให้ใช้ออปชั่นนี้หมด อาจทำให้ประสิทธิภาพลดลง

11.1.3 การทดลองที่ 1.3 ทดสอบการใช้งาน IPv4 หลังจากการติดตั้งแสตก

ทดสอบการติดต่อกับโหนดที่เป็น IPv4 ภายในเครือข่ายเดียวกันว่าสามารถติดต่อกันได้หรือไม่

สมมติฐาน หลังจากการติดตั้งดูอัล-แสตกจะสามารถใช้งาน IPv4 ได้ตามปกติ

สภาพแวดล้อม

โอสต์ 1

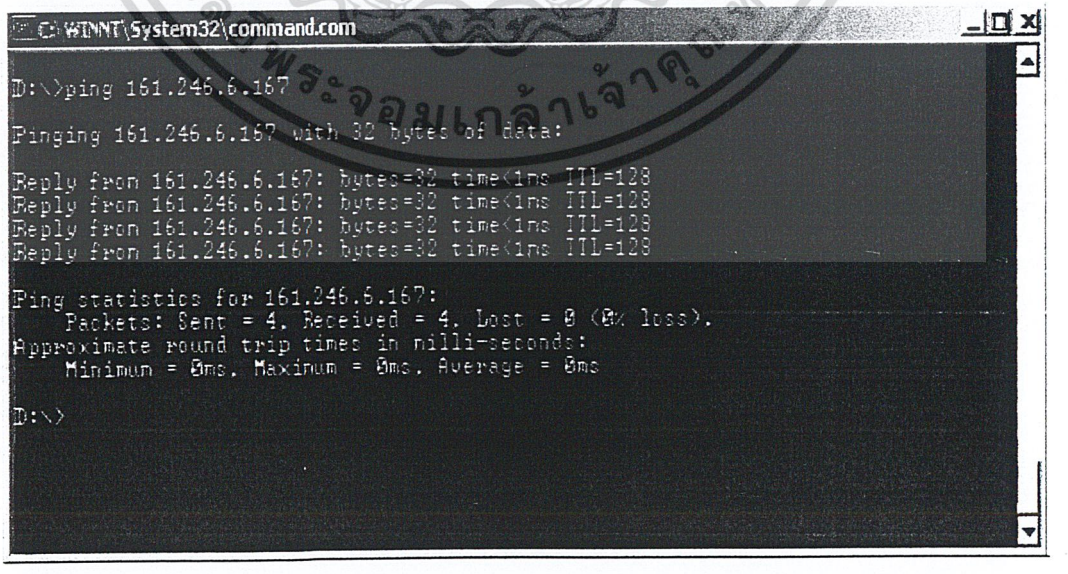
- ระบบปฏิบัติการ Windows2000
- หมายเลขอินเทอร์เฟซ 00-04-E2-0E-09-7C
- IPv4 แอ็ดเดรส 161.246.6.167
- IPv6 ลิ้งค์-โลคัล แอ็ดเดรส FE80::204:E2FF:FE0E:97C

โฮสต์ 2

- ระบบปฏิบัติการ Windows2000
- IPv4 แอ็ดเดรส 161.246.6.169

วิธีทดลอง

1. ติดตั้ง Microsoft dual-stack สำหรับ Windows 2000 ให้กับโอสต์ 1
2. เมื่อใช้คำสั่ง ipv6 if จะพบว่าที่อินเทอร์เฟซหมายเลข 4 มีลิ้งค์-โลคัลแอ็ดเดรสเป็น FE80::204:E2FF:FE0E:97C ซึ่งเป็นแอ็ดเดรสที่ได้จากหมายเลขประจำการ์ดเครือข่ายของเครื่องคือ 00-04-E2-0E-09-7C ขณะนี้โฮสต์ 1 พร้อมจะใช้งาน IPv6 ติดต่อกับเครื่องภายในเครือข่ายเดียวกันได้
3. ทดสอบความสามารถในการทำงานร่วมกับ IPv4 โดยการติดต่อกับโฮสต์ 2 ซึ่งเป็นโฮสต์ที่ทำงานภายใต้ IPv4 เท่านั้นและอยู่ในเครือข่ายเดียวกับโฮสต์ 1 ทดสอบโดยใช้การ ping ติดต่อกันระหว่างจากโอสต์ 1 ไปยัง โอสต์ 2 ได้ผลดังภาพ



รูปที่ 11.1.8 ผลการติดต่อจากโฮสต์ดูอัลแสตกไปยังโอสต์ IPv4 เท่านั้นที่อยู่ภายในเครือข่ายเดียว

เมื่อทำการ ping ติดต่อจากโฮสต์ 2 กลับไปยังโอสต์ 1 ก็ได้ผลในลักษณะเดียวกัน

4. จากภาพแสดงว่าโฮสต์ที่เป็นดูอัล-แสตกสามารถติดต่อกับโอสต์ที่ทำงานเฉพาะ IPv4 ที่อยู่ภายในเครือข่ายเดียวกันได้

ข้อสรุป

หลังจากการติดตั้งดูอัล-แสตก จะสามารถทำให้ใช้งานได้ทั้ง IPv6 และ IPv4 สำหรับเครือข่ายใดๆที่ต้องการให้ใช้ IPv6 โพรโตคอลได้และยังต้องการติดต่อกับ IPv4 โพรโตคอล จะสามารถทำได้โดยการติดตั้งดูอัล-แสตก

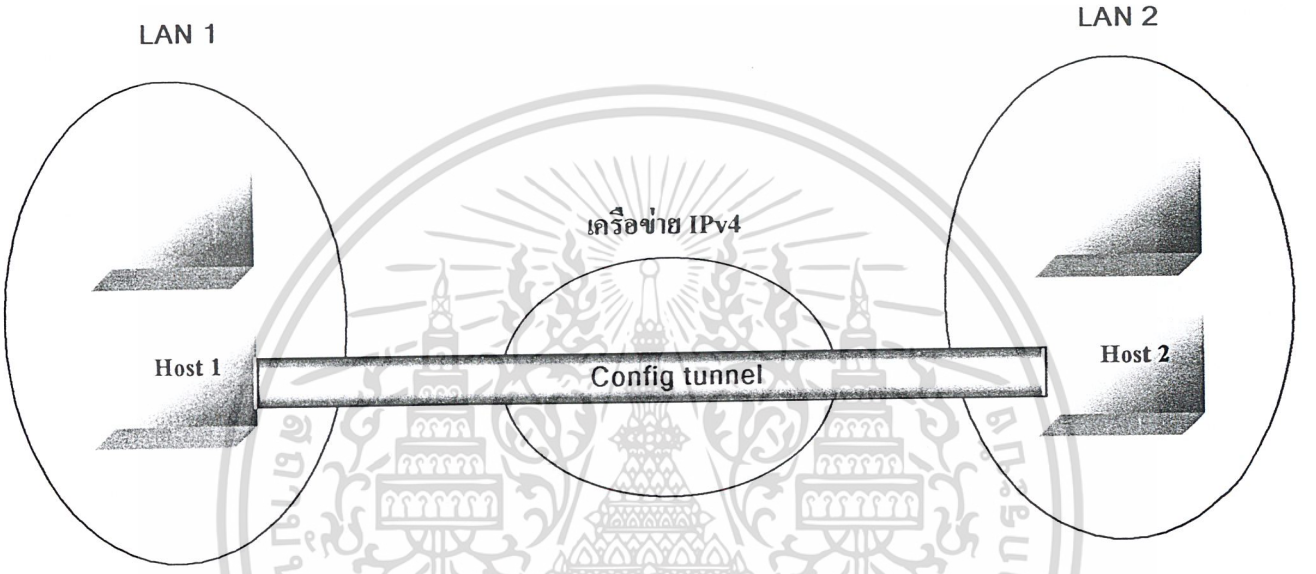
11.2 การทดลองที่ 2 : การติดต่อระหว่างเครือข่าย

จุดประสงค์

ทดลองการใช้งาน IPv6 ติดต่อกันระหว่างโหนดที่อยู่ต่างเครือข่ายกัน

สภาพแวดล้อม

โฮสต์ 2 โฮสต์ใดๆที่อยู่ต่างเครือข่ายกันเชื่อมต่อกันผ่านเราเตอร์



รูปที่ 11.2.1 ลักษณะการเชื่อมต่อของโฮสต์ในการทดลองที่ 2

การติดต่อกันทำได้โดยการคอนฟิกให้มีการทำทันเนิลโดยตรงจากโฮสต์ต้นทางไปยังปลายทางโดยตรง

การทดลอง

11.2.1 การทดลองที่ 2.1 ทดสอบการติดต่อ IPv6 ในระบบปฏิบัติการที่ต่างกันระหว่างโฮสต์ที่อยู่ต่างเครือข่ายกันกัน

ทดสอบการเชื่อมต่อระหว่างโหนด IPv6 ในระบบปฏิบัติที่ต่างกันและโอสต์ทั้งสองอยู่ต่างเครือข่ายกัน ว่าสามารถเชื่อมต่อกันได้หรือไม่

สมมติฐาน ความสามารถในการติดต่อและการทำงานของ IPv6 จะไม่ขึ้นอยู่กับระบบปฏิบัติการและไม่ว่าจะอยู่ในเครือข่ายเดียวกันหรือไม่ ทำให้สามารถติดต่อและทำงานร่วมกันได้

ตัวแปรต้น ระบบปฏิบัติการ Window 2000 , Linux Redhat 7.2 , Windows XP

ตัวแปรตาม ผลการติดต่อ

สภาพแวดล้อม

โฮสต์	ระบบปฏิบัติการ	IPv4 Address
1	Windows 2000	161.246.6.167
2	Windows XP	161.246.5.111
3	Linux Redhat 7.2	203.107.243.152

ตารางที่ 11.2.1 สภาพแวดล้อมสำหรับการทดลองที่ 2.1

วิธีทดลอง

1. ติดตั้ง IPv6 ให้กับโฮสต์ทดลองที่ 1 ที่ติดตั้งระบบปฏิบัติการ Window 2000

   โฮสต์ 1 จะมี IPv6 แอ็ดเดรสเป็น 2002:A1F6:6A7:3:0:3:A1F6:6A7

   ติดตั้ง IPv6 ให้กับโฮสต์ทดลองที่ 2 ที่ติดตั้งระบบปฏิบัติการ Window XP โดยใช้คำสั่ง

   ipv6 install

   โฮสต์ 2 จะมี IPv6 แอ็ดเดรสเป็น 2002:A1F6:56F:2:0:2:A1F6:56F

   ติดตั้ง IPv6 ให้กับโฮสต์ 3 ที่ติดตั้งระบบปฏิบัติการ Linux Redhat 7.2 ทำได้ดังนี้

   ในไฟล์ /etc/sysconfig/network เพิ่มคำสั่ง

   NETWOKING_IPV6=yes

   ในไฟล์ /sys/sysconfig/network-scripts/ifcfg-sit1 เพิ่มคำสั่ง

IPV6INIT="yes"

IPV6TUNNELIPV4="161.246.6.167"

IPV6ADDR="2002:CB6B:F398::CB6B:F398/127"

ในไฟล์ /sys/sysconfig/network-scripts/ifcfg-sit2 เพิ่มคำสั่ง

IPV6INIT="yes"

IPV6TUNNELIPV4="161.246.5.111"

IPV6ADDR="2002:CB6B:F398::CB6B:F398/127"

โฮสต์ 3 จะมี IPv6 แอ็ดเดรสเป็น 2002:CB6B:F398::CB6B:F398

2. ที่โฮสต์ 1 คอนฟิกให้เอ็นแคป์แพ็กเก็ตโดยตรงสู่โฮสต์ปลายทางโดยการใช้คำสั่ง

ipv6 rtu 2002:A1F6:56F/48 3/::161.246.5.111

ipv6 rtu 2002:CB6B:F398/48 3/::203.107.243.152

ที่โฮสต์ 2 ใช้คำสั่ง

ipv6 rtu 2002:A1F6:6A7/48 2/::161.246.6.167

ipv6 rtu 2002:CB6B:F398/48 2/::203.107.243.152

ที่โฮสต์ 3 ใช้คำสั่ง

/etc/sysconfig/network-scripts/ifup sit1

/etc/sysconfig/network-scripts/ifup sit2

route –A inet6 add 2002:A1F6:6A7::/48 dev sit 1

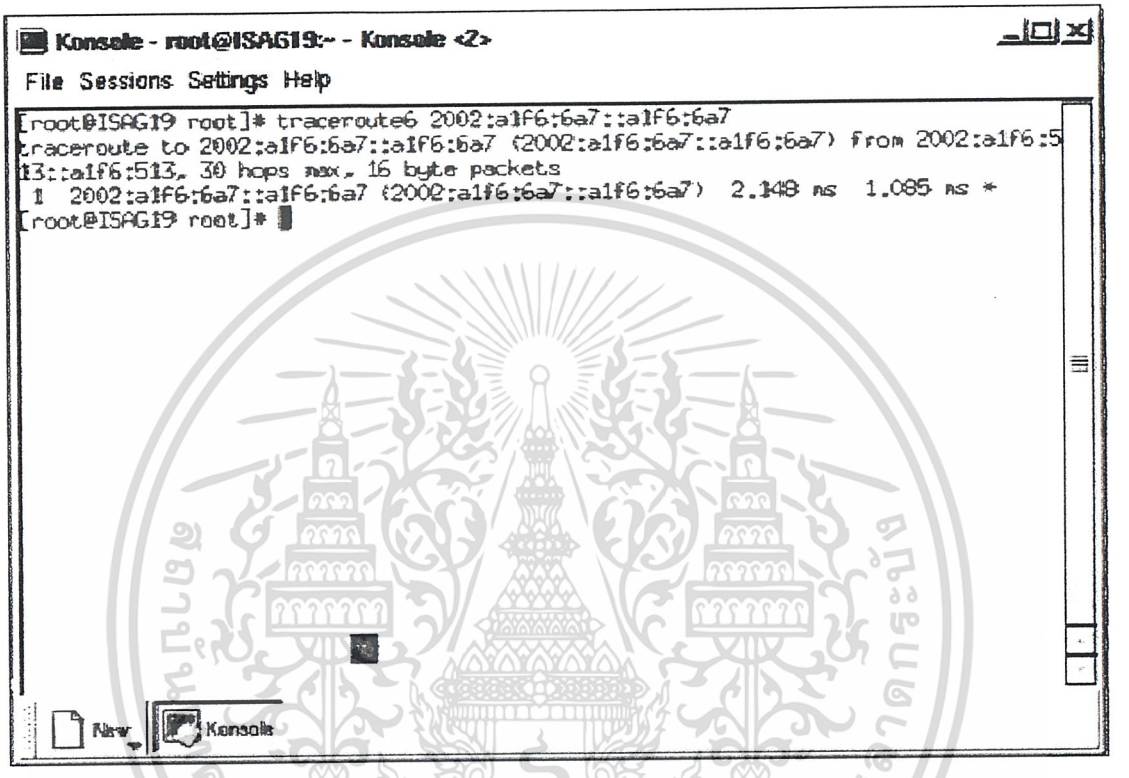
route –A inet6 add 2002:A1F6:56F::/48 dev sit 2

3. เมื่อทดสอบโดยการ ping ติดต่อกันทั้งหมดได้ผลดังตาราง

ระบบปฏิบัติการโฮสต์ต้นทาง	ระบบปฏิบัติการโฮสต์ปลายทาง	ผลการติดต่อ
Windows 2000	Windows XP	สามารถติดต่อได้
Windows 2000	Linux Redhat 7.2	สามารถติดต่อได้
Windows XP	Windows 2000	สามารถติดต่อได้
Windows XP	Linux Redhat 7.2	สามารถติดต่อได้
Linux Redhat 7.2	Windows XP	สามารถติดต่อได้

Linux Redhat 7.2	Windows 2000	สามารถติดต่อได้

ตารางที่ 11.2.2 ผลการทดสอบโดยการ ping ติดต่อกันระหว่างโฮสต์ที่ติดตั้งระบบปฏิบัติการต่างๆในเครือข่ายที่ต่างกัน



รูปที่ 11.2.2 ผลการติดต่อโดยการ ping จากโฮสต์ Linux มายัง Windows 2000 ที่อยู่ต่างเครือข่ายกันโดยผ่านการเอ็นแค็ปถึงกันโดยตรง

ข้อสรุป

ความสามารถในการติดต่อของ IPv6 จะเหมือนกับ IPv4 นั่นคือจะไม่ขึ้นอยู่กับว่าจะอยู่ในเครือข่ายเดียวกันหรือไม่ หรือจะใช้ระบบปฏิบัติการเดียวกันหรือไม่ ก็ยังจะสามารถทำงานร่วมกันได้

11.2.2 การทดลองที่ 2.2 ทดสอบความสามารถทางด้านความปลอดภัยระหว่าง 2 โหนดที่อยู่ต่างเครือข่ายกัน

คอนฟิกให้มีการใช้ส่วนหัวทางด้านความปลอดภัยระหว่างโหนดที่อยู่ต่างเครือข่ายกัน

สมมติฐาน สามารถใช้ความปลอดภัยระหว่างสองโหนดใดๆได้ แม้ว่าสองโหนดนั้นจะไม่อยู่ภายในเครือข่ายเดียวกัน

สภาพแวดล้อม

โฮสต์ 1

- ระบบปฏิบัติการ Windows 2000
- IPv4 แอ็ดเดรส 161.246.6.167
- IPv6 แอ็ดเดรส 2002:A1F6:6A7:3:0:3:A1F6:6A7

โฮสต์ 2

- ระบบปฏิบัติการ Windows 2000
- IPv4 แอ็ดเดรส 161.246.5.113
- IPv6 แอ็ดเดรส 2002:A1F6:571:3:0:3:A1F6:571

วิธีทดลอง

1. ติดตั้ง IPv6 dual-stack ให้กับโฮสต์ 1 และโฮสต์ 2
2. คอนฟิกให้แต่ละตัวทำหน้าที่เป็น 6to4 เราเตอร์เอง และส่งต่อแพ็กเก็ตโดยการทำคอนฟิกทันเนิลโดยตรงถึงกัน โดยที่โฮสต์ 1 พิมพ์คำสั่ง

   ipv6 rtu ::/0 3/:161.246.5.113 pub

   และที่โอสต์ 2 พิมพ์คำสั่ง

   ipv6 rtu ::/0 3/:161.246.6.167 pub

3. สร้าง IPSec Security Association (SA) ระหว่างทั้งสองโฮสต์ เพื่อทำการพิสูจน์ตัวจริง (Authentiation) โดยใช้อัลกอริทึ่ม Message Digest 5 (MD5) ร่วมกับ HMAC โดยที่โฮสต์ 1 ทำการสร้างไฟล์ Policy (.SPD) และ Association (.SAD) โดยใช้คำสั่ง

   ipsec6 c test

   จะได้ไฟล์สำหรับกำหนด Policy ชื่อ test.spd และไฟล์ Association ชื่อ test.sad ใช้โปรแกรมเท็กซ์อีดิเตอร์เปิดไฟล์ test.spd ขึ้นมาและทำการเพิ่มแถวเข้าไปดังตาราง

ชื่อฟิลด์	ค่าตัวอย่าง
Policy	2
RemoteIPAddr	2002:A1F6:571:3:0:3:A1F6:571
LocalIPAddr	- *

RemotePort	- *
Protocol	- *
LocalPort	- *
IPSecProtocol	AH
IPSecMode	TRANSPORT
RemoteGWIPAddr	*
SABundleIndex	NONE
Direction	BIDIRECT
Action	APPLY
InterfaceIndex	0

ตารางที่ 11.2.3 แสดง Policy บนโฮสต์ 1

จากนั้นเปิดไฟล์ test.sad ขึ้นมาและทำการเพิ่ม Association เข้าไปดังตาราง

ชื่อฟิลด์	ค่าตัวอย่าง
SAEntry	2
SPI	3001
SADestIPAddr	2002:A1F6:571:3:0:3:A1F6:571
DestIPAddr	POLICY
SrcIPAddr	POLICY
Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	OUTBOUND
SecPolicyIndex	2

ตารางที่ 11.2.4 แสดง Association ของทราฟฟิคจากโฮสต์ 2 ไป โฮสต์ 1 บนโฮสต์ 1

ชื่อฟิลด์	ค่าตัวอย่าง
SAEntry	1
SPI	3000
SADestIPAddr	2002:A1F6:6A7:3:0:3:A1F6:6A7
DestIPAddr	POLICY
SrcIPAddr	POLICY
Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	INBOUND
SecPolicyIndex	2

ตารางที่ 11.2.5 แสดง Association ของทราฟฟิคจากโฮสต์ 1 ไป โฮสต์ 2 บนโฮสต์ 1

จากนั้น สร้างไฟล์ที่บรรจุคีย์จากการคอนฟิกกำหนดให้ใช้ไฟล์ชื่อ test.key โดยใช้เท็กซ์อีดิเตอร์สร้างใส่ข้อความลงไป กำหนดให้เป็น

"This is a key"

4. ที่โฮสต์ 2 พิมพ์คำสั่ง

ipsec6 c test

และทำการเพิ่ม Policy และ Association เข้าไปในไฟล์ test.spd และ test.sad ดังนี้

ชื่อฟิลด์	ค่าตัวอย่าง
Policy	2
RemoteIPAddr	2002:A1F6:6A7:3:0:3:A1F6:6A7
LocalIPAddr	- *
RemotePort	- *
Protocol	- *
LocalPort	- *
IPSecProtocol	AH

IPSecMode	TRANSPORT
RemoteGWIPAddr	*
SABundleIndex	NONE
Direction	BIDIRECT
Action	APPLY
InterfaceIndex	0

ตารางที่ 11.2.6 แสดง Policy บนโฮสต์ 2

ชื่อฟิลด์	ค่าตัวอย่าง
SAEntry	2
SPI	3001
SADestIPAddr	2002:A1F6:571:3:0:3:A1F6:571
DestIPAddr	POLICY
SrcIPAddr	POLICY
Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	INBOUND
SecPolicyIndex	2

ตารางที่ 11.2.7 แสดง Association ของทราฟฟิคจากโฮสต์ 1 ไป โฮสต์ 2 บนโฮสต์ 2

ชื่อฟิลด์	ค่าตัวอย่าง
SAEntry	1
SPI	3000
SADestIPAddr	2002:A1F6:6A7:3:0:3:A1F6:6A7
DestIPAddr	POLICY

SrcIPAddr	POLICY
Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	OUTBOUND
SecPolicyIndex	2

ตารางที่ 11.2.8 แสดง Association ของทราฟฟิคจากโฮสต์ 2 ไปโฮสต์ 1 บนโฮสต์ 2

สร้างไฟล์ชื่อ test.key ใช้บรรจุคีย์ดังนี้

"This is a key"

5. ทำการเพิ่ม Policy และ Association เข้าสู่ระบบโดยที่โฮสต์ 1 พิมพ์คำสั่งว่า

   ipsec6 a test

   และที่โฮสต์ 2 พิมพ์คำสั่งว่า

   ipsec6 a test

   เมื่อทำการตรวจดูโดยใช้คำสั่ง ***ipsec6 sp*** และ ***ipsec6 sa*** จะพบ Policy และ Association ที่สร้างมา

6. ทดสอบโดยการ Ping จากโฮสต์ 1 ไปยังโฮสต์ 2 โดยใช้คำสั่ง

   ping6 2002:A1F6:571:3:0:3:A1F6:571

   และตรวจสอบเส้นทางการส่งแพ็กเก็ตโดยใช้คำสั่ง

   tracert6 2002:A1F6:571:3:0:3:A1F6:571

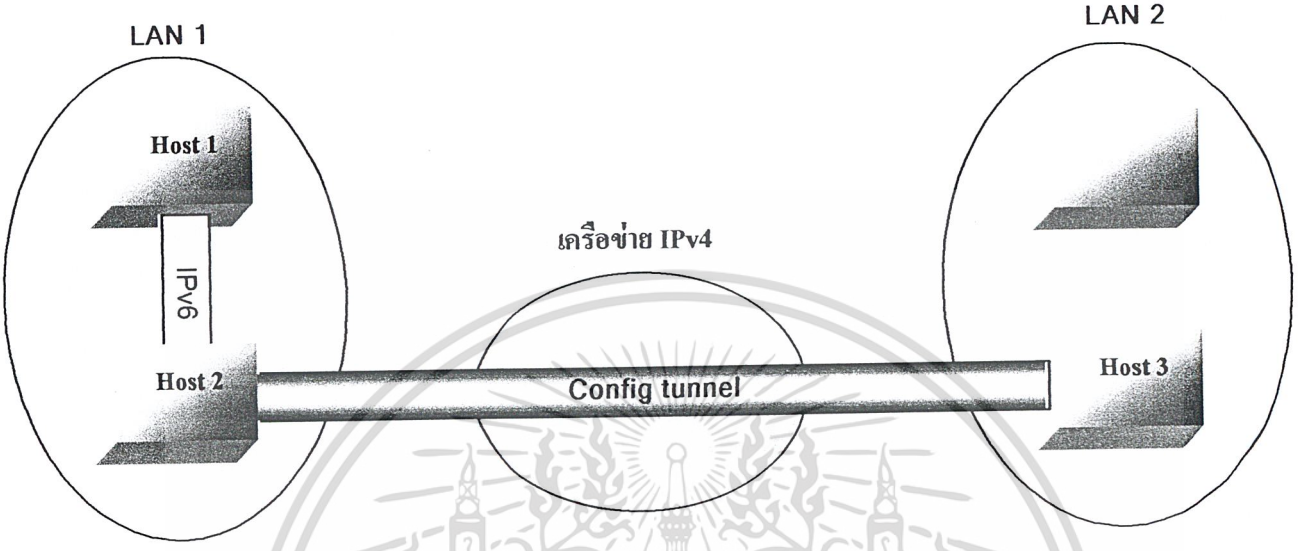
   เมื่อใช้โปรแกรมตรวจจับแพ็กเก็ตมาวิเคราะห์ดูพบว่ามีการใช้ส่วนหัวของการพิสูจน์ตัวจริง โดยค่าในส่วน Next header ของส่วนหัวของ IPv6 เป็น 33H ซึ่งเป็นส่วนเพิ่มเติมส่วนหัวชนิด Authentication เหมือนกับการทดลองที่ 1.2 แสดงว่าสามารถใช้ส่วนหัวด้านความปลอดภัยสำหรับการส่งแพ็กเก็ตข้อมูลระหว่างโหนดใดๆที่อยู่ต่างเครือข่ายได้

ข้อสรุป

เมื่อปรับแต่งคอนฟิกให้ใช้ความสามารถด้านความปลอดภัยของ IPv6 ให้กับโหนดสองโหนดใดๆ แพ็กเก็ตที่จะส่งไปมาระหว่างสองโหนดนั้นจะถูกครอบคลุมด้วยความปลอดภัยของ Ipv6 ทั้งหมด การทดลองนี้เหมาะสำหรับบุคคลสองคนที่ต้องการติดต่อกันแต่ไม่ได้อยู่ในเครือข่ายเดียวกันสามารถติดต่อกันได้โดยบุคคลอื่นไม่สามารถรับรู้ถึงข้อมูลที่ถูกส่งระหว่างสองคนนั้น

11.2.3 การทดลองที่ 2.3 ทดสอบการติดต่อ IPv6 โดยกำหนดให้โฮสต์อื่นเป็นฝ่ายเอ็นแค็ป

ทดสอบโดยกำหนดให้มีเครื่องหนึ่งทำหน้าที่เป็นตัวเอ็นแค็ปแพ็กเก็ต IPv6 ทั้งหมดภายในเครือข่ายออกสู่ภายนอก



รูปที่ 11.2.3 แสดงการเชื่อมต่อสำหรับทดลองที่ 2.3

สมมติฐาน สามารถติดตั้งโหนดใดๆให้เป็น IPv6 เกตเวย์ได้และโหนดใดๆที่อยู่ในเครือข่ายสามารถใช้เกตเวย์เป็นทางออกไปสู่เครือข่ายอื่นได้

สภาพแวดล้อม

โฮสต์ 1

- ระบบปฏิบัติการ Windows 2000
- IPv4 แอ็ดเดรส 161.246.6.169
- หมายเลข อินเทอร์เฟซ 00-04-E2-0E-09-E8

โฮสต์ 2

- ระบบปฏิบัติการ Windows 2000
- IPv4 แอ็ดเดรส 161.246.6.167
- หมายเลข อินเทอร์เฟซ 00-04-E2-0E-09-7C

โฮสต์ 3

- ระบบปฏิบัติการ Windows 2000
- IPv4 แอ็ดเดรส 161.246.5.113

วิธีทดลอง

1. ติดตั้ง Microsoft IPv6 dual-stack ให้กับโฮสต์ทดลองที่ติดตั้งระบบปฏิบัติการ Windows 2000 ทั้ง 3 โฮสต์ โดยกำหนดให้โฮสต์ 2 จะทำหน้าที่เป็นเราเตอร์บริการทำหน้าที่ส่งต่อแพ็กเก็ตจากทุกเครื่องภายในเครือข่ายส่งต่อให้ถึงปลายทาง มีหมายเลขอินเทอร์เฟซเป็น 00-04-E2-0E-09-7C และ แอ็ดเดรส IPv4 เป็น 161.246.6.167 ส่วนโฮสต์หมายเลข 1 เป็นโฮสต์ที่อยู่ภายในเครือข่ายเดียวกับโฮสต์ 2 เป็นผู้ขอใช้บริการจากโฮสต์ 2 มีหมายเลขอินเทอร์เฟซเป็น 00-04-E2-0E-09-E8 และมีแอ็ดเดรส IPv4 เป็น 161.246.6.169 และที่โฮสต์ 3 กำหนดให้เป็นโฮสต์ปลายทางที่อยู่ต่างเครือข่ายกัน มีแอ็ดเดรส IPv4 เป็น 161.246.5.113 และมี IPv6 แอ็ดเดรสเป็น 2002:A1F6:571:3:0:3:A1F6:571
2. คอนฟิกโฮสต์ 2 ให้ทำหน้าที่เป็นเกตุเวย์บริการแก่โฮสต์ตัวอื่นภายในเครือข่าย โดยการพิมพ์คำสั่ง

   6to4cfg -r

   คำสั่งนี้จะทำให้โฮสต์ 2 ทำหน้าที่เป็นเราเตอร์และกระจายค่าไซต์พรีฟิกซ์ FE80:A1F6:6A7:4 และ 2002:A1F6:6A7:3 ซึ่งได้มาจากค่า IPv4 แอ็ดเดรสของโฮสต์ 2 ไปให้กับโฮสต์อื่นๆภายในเครือข่ายเดียวกัน และยังทำหน้าที่ Forward แพ็กเก็ตออกนอกเครือข่ายให้
3. ในขณะนี้ที่โฮสต์ 1 จะมี IPv6 แอ็ดเดรสเพิ่มขึ้นมา 2 แอ็ดเดรสซึ่งเกิดจากการประกาศค่าไซต์พรีฟิกซ์จากโฮสต์ 2 คือ FE80:A1F6:6A7:4:204:E2FF:FE0E:9E8 ที่อินเทอร์เฟซหมายเลข 4 และ 2002:A1F6:6A7:3:0:3:A1F6:6A9 ที่อินเทอร์เฟซหมายเลข 3
4. ที่โฮสต์ 2 คอนฟิกให้ส่งต่อแพ็กเก็ตไปยังปลายทาง ในการทดลองนี้คือโฮสต์ 3 โดยการพิมพ์คำสั่ง

   ipv6 rtu 2002:A1F6:571/48 3/::161.246.5.113 pub

   โฮสต์ 2 จะทำการส่งต่อแพ็กเก็ตที่มีพรีฟิกซ์เป็น 2002:A1F6:571/48 ไปยังโฮสต์ที่มี IPv4 แอ็ดเดรสเป็น 161.246.5.113 โดยการทำคอนฟิกทันเนิล
5. คอนฟิกให้โฮสต์ 1 หรืออาจจะเป็นโฮสต์ใดๆในเครือข่ายที่ต้องการให้ส่งต่อแพ็กเก็ตออกนอกเครือข่ายผ่านโฮสต์บริการ (โฮสต์ 2) ทำการส่งต่อแพ็กเก็ตมายังโฮสต์บริการโดยการพิมพ์คำสั่ง

   ipv6 rtu ::/0 4/FE80:A1F6:6A7:4:204:E2FF:FE0E:97C pub

   แพ็กเก็ต IPv6 ทั้งหมดจะถูกส่งต่อจากโฮสต์นี้ไปยังโฮสต์บริการโดยไม่ผ่านการเอ็นแค็ป จากการคอนฟิกทั้งหมดสามารถสร้างเส้นทางการเดินของแพ็กเก็ตจากต้นทางไปยังปลายทางได้ในการทดลองนี้คือจากโฮสต์ 1 ไปยังโฮสต์ 3

6. สามารถทดสอบการทำงานได้โดยการ ping ที่โฮสต์ 1 โดยใช้คำสั่ง

   ping6 2002:A1F6:571:3:0:3: A1F6:571

7. สำหรับการทำงานทั้งหมดนั้นจะเริ่มจากเมื่อโฮสต์ใดๆภายในเครือข่ายต้องการจะติดต่อกับโฮสต์ภายนอกเครือข่ายก็จะส่ง IPv6 แพ็กเก็ตที่มีแอ็ดเดรสปลายทางเป็นโฮสต์ปลายทางนอกเครือข่ายไปยังโฮสต์บริการ โฮสต์บริการก็จะตรวจสอบแอ็ดเดรสปลายทางและทำการเอ็นแค็ปโดยแพ็กเก็ต IPv4 โดยมีแอ็ดเดรสปลายทางของส่วนหัวของ IPv4 เป็นแอ็ดเดรส IPv4 ของโฮสต์ปลายทาง โดยตรวจสอบจากตารางการเลือกเส้นทาง

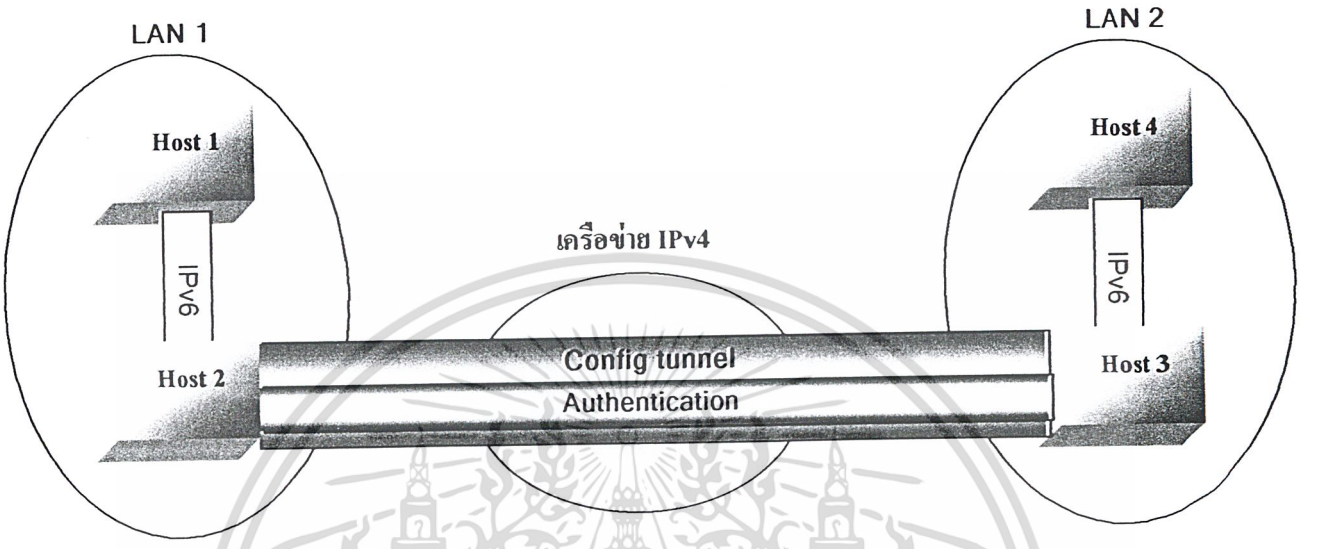
ข้อสรุป

เมื่อทำการปรับแต่งโหนดใดโหนดหนึ่งให้เป็น IPv6 เกตเวย์และปรับแต่งตารางเราติงที่เหมาะสม โหนดอื่นๆที่อยู่ในเครือข่ายเมื่อทำการปรับแต่งให้รู้จักเกตเวย์โหนดเหล่านี้จะสามารถติดต่อไปเครือข่ายอื่นได้โดยผ่านเกตเวย์ ทำให้โหนดอื่นๆที่อยู่ในเครือข่ายไม่ต้องมีความสามารถด้านเอ็นแค็ปและดีแค็ป เพียงแค่มีเครื่องที่มีความสามรถเอ็นแค็ปและดีแค็ปเพียงเครื่องเดียว เครื่องทุกเครื่องในเครือข่ายก็จะสามารถส่งข้อมูลผ่านออกไปได้

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

11.2.4 การทดลองที่ 2.4 ทดสอบการใช้ส่วนหัวด้านความปลอดภัยกับโหนดที่ถูกกำหนดให้เป็นโฮสต์บริการออกนอกเครือข่าย

ทดสอบโดยกำหนดให้แต่ละเครือข่ายมีโฮสต์ที่ทำหน้าที่เป็นตัวเอ็นแค็ปแพ็กเก็ต IPv6 ทั้งหมดภายในเครือข่ายออกสู่ภายนอก และใช้ความสามารถด้านความปลอดภัยกับโหนดนี้



รูปที่ 11.2.4 แสดงการเชื่อมต่อสำหรับการทดลองที่ 2.4

สมมติฐาน สามารถติดตั้ง IPv6 เกตเวย์ที่มีความปลอดภัยระหว่างสองเครือข่ายใดๆ ได้

สภาพแวดล้อม

โฮสต์	ระบบปฏิบัติการ	หมายเลขอินเทอร์เฟซ	IPv4
1	Windows 2000	00-04-E2-0E-09-E8	161.246.6.169
2	Windows 2000	00-04-E2-0E-09-7C	161.246.6.167
3	Windows 2000	00-E0-29-58-FA-31	161.246.5.113
4	Windows 2000	00-E0-7D-03-29-A1	161.246.6.111

ตารางที่ 11.2.9 สภาพแวดล้อมสำหรับการทดลองที่ 2.4

วิธีทดลอง

1. ติดตั้งดูอัลแสตกให้กับโฮสต์ทั้ง 4 เครื่อง
2. คอนฟิกโฮสต์ 2 และ โฮสต์ 3 ทำหน้าที่เป็นเกตเวย์ทางออกนอกเครือข่ายและใช้ความสามารถ IPSec โดยคอนฟิกได้โดยใช้คำสั่ง

   6to4cfg -r

ที่โฮสต์ 2 และโฮสต์ 3 เพื่อให้ทั้ง 2 เครื่องทำงานเป็นเราเตอร์หรือเกตเวย์ของแต่ละเครือข่ายย่อยโดย

โฮสต์ 2 จะกระจายพรีฟิกซ์ FE80:A1F6:6A7:4::/64 และ 2002:A1F6:6A7:3::/64 ให้กับอินเทอร์เฟซหมายเลข 4 และ 3 ตามลำดับ (อินเทอร์เฟซ 4 สำหรับลิ้งค์-โลคัลแอ็ดเดรส และ อินเทอร์เฟซ 3 สำหรับการทำทันเนิล)

โฮสต์ 3 จะกระจายพรีฟิกซ์ FE80:A1F6:571:4::/64 และ 2002:A1F6:571:3::/64 ให้กับอินเทอร์เฟซหมายเลข 4 และ 3 ตามลำดับ

โฮสต์ 1 และ โฮสต์ 4 จะนำค่าพรีฟิกซ์ที่ได้มาสร้างเป็น IPv6 แอ็ดเดรสได้ดังนี้

โฮสต์ 1 จะได้แอ็ดเดรส FE80:A1F6:6A7:4:204:E2FF:FE0E:9E8 ที่อินเทอร์เฟซหมายเลข 4 และ แอ็ดเดรส 2002:A1F6:6A7:3:0:3:A1F6:6A9 ที่อินเทอร์เฟซหมายเลข 3

โฮสต์ 4 จะได้แอ็ดเดรส FE80:A1F6:571:4:2E0:7DFF:FE03:29A1 ที่อินเทอร์เฟซหมายเลข 4 และ แอ็ดเดรส 2002:A1F6:571:3:0:3:A1F6:56F ที่อินเทอร์เฟซหมายเลข 3

3. กำหนดตารางเราติ้งให้กับโฮสต์ 1 และ 4 ให้ใช้อินเทอร์เฟซ 4 เพื่อติดต่อกับโฮสต์ 2 และ 3 โดยที่โฮสต์ 1 ใช้คำสั่ง

ipv6 rtu ::/0 4/FE80:A1F6:6A7:4:204:E2FF:FE0E:97C pub

และที่โฮสต์ 4 ใช้คำสั่ง

ipv6 rtu ::/0 4/FE80:A1F6:571:4:2E0:29FF:FE58:FA31

4. กำหนดตารางเราติ้งให้กับเราเตอร์โดยที่โฮสต์ 2 พิมพ์คำสั่ง

ipv6 rtu 2002:A1F6:571/48 3/::161.246.5.113 pub

และที่โฮสต์ 3 พิมพ์คำสั่ง

ipv6 rtu 2002:A1F6:6A7/48 3/::161.246.5.113 pub

5. กำหนดความสามารถด้านความปลอดภัยระหว่างโฮสต์ 2 และ โฮสต์ 3 ตามการทดลองที่ 2.2

6. ทดสอบการทำงานโดยใช้คำสั่งที่โฮสต์ 1

ping6 2002:A1F6:571:3:0:3:A1F6:56F

เมื่อตรวจสอบแพ็กเก็ตระหว่างโฮสต์ 2 และ โฮสต์ 3 จะพบว่ามีการใช้ส่วนเพิ่มเติมส่วนหัวด้านความปลอดภัย

ข้อสรุป

เมื่อทำการปรับแต่ง IPv6 เกตเวย์ให้ใช้ความสามารถด้านความปลอดภัยระหว่างสองเครือข่ายใดๆ แพ็กเก็ตที่จะส่งระหว่างสองเครือข่ายนี้จะถูกครอบคลุมด้วยความสามารถด้านความปลอดภัยของ IPv6 การทดลองนี้เหมาะสำหรับ เครือข่ายสองเครือข่ายที่ต้องการติดต่อกันโดยต้องการความปลอดภัย โดยสามารถปรับแต่งระบบความปลอดภัยได้ที่เครื่องเพียงเครื่องเดียวนั่นคือเครื่อง IPv6 เกตเวย์ ข้อมูลที่จะถูกส่งระหว่างโฮสต์ที่อยู่ภายในเครือข่ายสองเครือข่ายนี้จะถูกครอบคลุมด้วยความปลอดภัย แต่จะมีข้อเสียคือไม่ว่าข้อมูลนั้นจะต้องการให้เป็นความลับหรือไม่ก็จะถูกทำการเข้ารหัสที่เครื่อง IPv6 เกตเวย์ทั้งหมด ดังนั้นถ้ามีข้อมูลจำนวนมากที่ไม่จำเป็นต้องทำให้เป็นความลับ จะทำให้ประสิทธิภาพของระบบโดยรวมลดลง เครื่องเกตเวย์จึงควรจะเป็นเครื่องที่มีประสิทธิภาพสูง

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

11.2.5 การทดลองที่ 2.5 ทดสอบการแฟรกเมนต์ใน IPv6

กำหนดให้ข้อมูลที่ส่งมีขนาดมากกว่าขนาด MTU เพื่อทดสอบการแฟรกเมนต์

สมมติฐาน เมื่อส่งแพ็กเก็ตที่มีขนาดใหญ่กว่าขนาด MTU ที่เครือข่ายที่ต้องการส่งแพ็กเก็ตไปยอมรับได้ เครื่องที่ส่งแพ็กเก็ตจะใช้ความสามารถของการแฟรกเมนต์

สภาพแวดล้อม

โฮสต์ 1

- ระบบปฏิบัติการ Windows 2000
- IPv4 แอ็ดเดรส 161.246.6.167
- IPv6 แอ็ดเดรส 2002:A1F6:6A7::A1F6:6A7

โฮสต์ 2

- ระบบปฏิบัติการ Linux Redhat 7.2
- IPv4 แอ็ดเดรส 161.246.5.19
- IPv6 แอ็ดเดรส 2002:A1F6:513::A1F6:513

วิธีทดลอง

1. ติดตั้ง Microsoft Dual-Stack ให้กับโฮสต์ 1 เหมือนกับการทดลองที่ผ่านมา
2. ทำการคอนฟิกให้เอ็นแค็ปแพ็กเก็ต IPv6 ไปยังโฮสต์ 2 โดยตรงโดยใช้คำสั่ง

   ipv6 rtu ::/0 3/:161.246.5.19 pub
3. คอนฟิกลีนุกซ์

   ในไฟล์ /etc/sysconfig/network เพิ่มคำสั่ง

   NETWOKING_IPV6=yes

   IPV6_MTU=1500

   ในไฟล์ /sys/sysconfig/network-scripts/ifcfg-sit1 เพิ่มคำสั่ง

   IPV6INIT="yes"

   IPV6TUNNELIPV4="161.246.6.167"

   IPV6ADDR="2002:A1F6:513::A1F6:513/127"

   ใช้คำสั่งต่อไปนี้ เพื่อเพิ่มตารางเราติง

   /etc/sysconfig/network-scripts/ifup sit1

   route –A inet6 add ::/0 dev sit 1

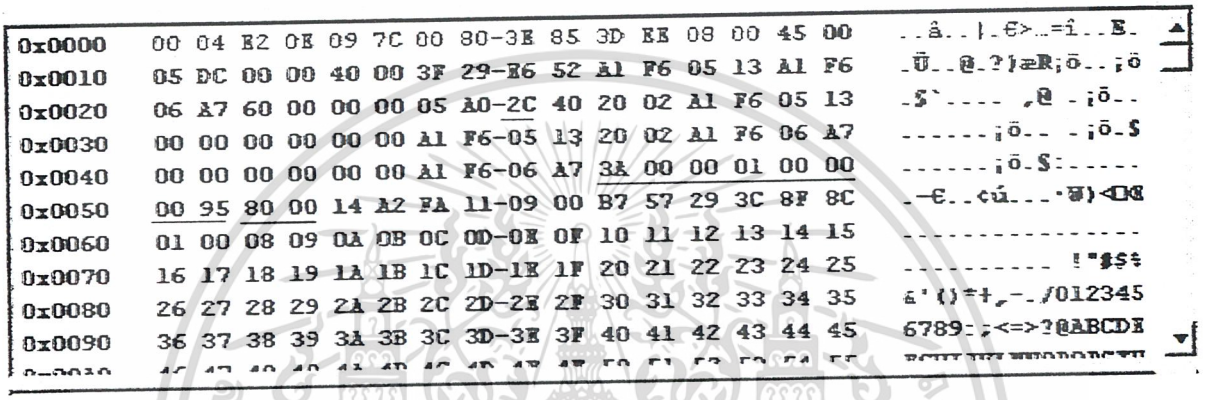
4. ทดสอบโดยการ ping จากโฮสต์ 2 มายังโฮสต์ 1 โดยกำหนดขนาดของข้อมูลเป็น 3008 ไบต์ โดยใช้คำสั่ง

   ping6 –s 3008 2002:A1F6:6A7::A1F6:6A7

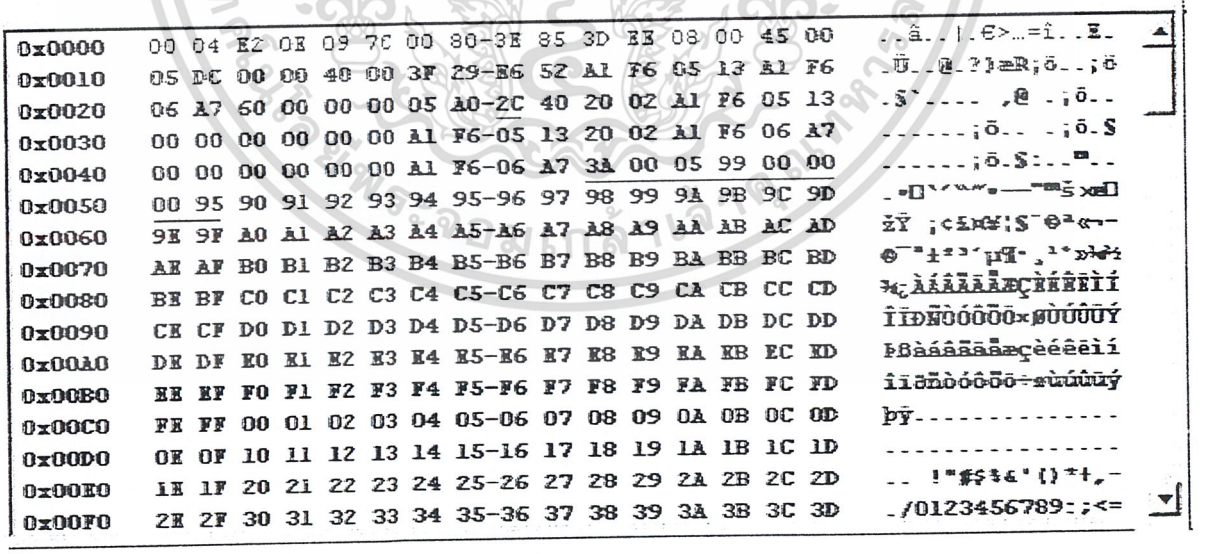
   ใช้โปรแกรมตรวจจับแพ็กเก็ตที่โฮสต์ 1 เพื่อนำมาตรวจสอบ

5. ได้แพ็กเก็ตแสดงดังภาพ

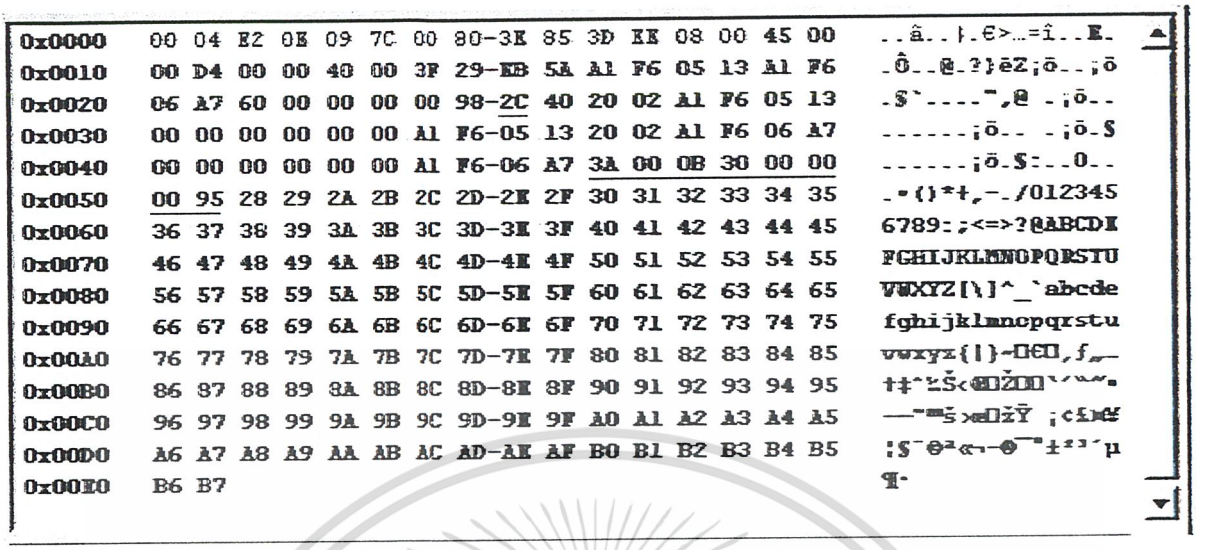
แพ็กเก็ตที่ 1



แพ็กเก็ตที่ 2



แพ็กเก็ตที่ 3



รูปที่ 11.2.5 ผลการจับแพ็กเก็ตที่มีการใช้ส่วนเพิ่มเติมส่วนหัวของการแฟร็กเมนต์

จากภาพ

- ค่าในส่วน Next Header ของส่วนหัวของ IPv6 ในทุกๆแพ็กเก็ต (แสดงในภาพที่แอ็ดเดรส 0x0028) มีค่าเป็น 2CH แสดงว่าส่วนที่อยู่ถัดจากส่วนหัวของ IPv6 ทั้งสามแพ็กเก็ตนั้นเป็นส่วนเพิ่มเติมส่วนหัวแบบแฟรกเมนต์ โดยส่วนเพิ่มเติมส่วนหัวแบบแฟรกเมนต์ในแต่ละแพ็กเก็ตจะมีขนาด 8 ไบต์แสดงในภาพตั้งแต่แอ็ดเดรส 0x004A – 0x0051
- ค่าในส่วน Next Header ของส่วนเพิ่มเติมส่วนหัวแบบแฟรกเมนต์ของแต่ละแพ็กเก็ต (แสดงในภาพที่แอ็ดเดรส 0x004A) จะมีค่าเป็น 3AH ซึ่งเป็นการบอกว่าส่วนที่อยู่ต่อจากส่วนนี้เป็นส่วนของ ICMPv6 และเฉพาะในแพ็กเก็ตที่ 1 จะเป็นส่วนเริ่มต้นของ ICMPv6 โดยมีค่า Type เป็น 80H (แอ็ดเดรส 0x0052) แสดงว่าเป็นแพ็กเก็ตร้องขอการตอบรับ
- เมื่อดูขนาดของแพ็กเก็ต

  แพ็กเก็ตที่ 1 และ 2 มีขนาด 1500 ไบต์ (ไม่รวมส่วนของอีเทอร์เน็ต 13 ไบต์) แบ่งเป็น

ส่วนหัวของ IPv4	20	ไบต์
ส่วนหัวของ IPv6	40	ไบต์
ส่วนเพิ่มเติมส่วนหัวแบบแฟรกเมนต์	8	ไบต์
ส่วนข้อมูล	1432	ไบต์

  แพ็กเก็ตที่ 3 มีขนาด 219 ไบต์ (ไม่รวมส่วนของอีเทอร์เน็ต 13 ไบต์) แบ่งเป็น

ส่วนหัวของ IPv4	20	ไบต์
ส่วนหัวของ IPv6	40	ไบต์
ส่วนเพิ่มเติมส่วนหัวแบบแฟรกเมนต์	8	ไบต์

ส่วนข้อมูล 144 ไบต์

รวมข้อมูลทั้งหมดที่ได้รับ 1432 + 1432 + 144 = 3008 ไบต์ เท่ากับข้อมูลที่ส่งมา

- ค่าในส่วน Identification ของทั้งสามแพ็กเก็ต มีค่าเป็น 0095H (แสดงในภาพที่แอ็ดเดรส 0x0050 และ 0x0051) เป็นการบอกว่าข้อมูลทั้งสามแพ็กเก็ตนี้เป็นข้อมูลชุดเดียวกัน
- สังเกตค่าในส่วน Fragment Offset (แอ็ดเดรส 0x005C จนถึงบิตที่ 8 – 4 ของแอ็ดเดรส 0x005D) ซึ่งเป็นการบอกว่าข้อมูลของแพ็กเก็ตนี้เป็นส่วนใดของข้อมูลทั้งหมด และค่าแฟล็ก M (บิตที่ 1 ของแอ็ดเดรส 0x005D) หากมีค่าเป็น 0 จะเป็นการบอกว่าแพ็กเก็ตนี้เป็นแพ็กเก็ตสุดท้าย

แพ็กเก็ต	ขนาดข้อมูล (ไบต์)	Fragment Offset	M
1	1432	0	1
2	1432	179 เท่ากับไบต์ที่ 1432	1
3	144	358 เท่ากับไบต์ที่ 2864	0

ตารางที่ 11.2.10 แสดงรายละเอียดข้อมูลในแต่ละแพ็กเก็ต

เป็นการพิสูจน์ได้ว่าเมื่อขนาดของข้อมูลที่ทำการส่งมีขนาดมากกว่า MTU ของการเชื่อมต่อก็สามารถทำการแฟร็กเมนต์ที่โฮสต์ผู้ส่งได้เพื่อช่วยลดภาระของเราเตอร์ทางผ่านได้

ข้อสรุป

ความสามารถด้านแฟรกเมนต์ช่วยให้สามารถส่งแพ็กเก็ตภายในเครือข่ายที่รองรับค่า MTU สูงไปยังเครือข่านที่รองรับค่า MTU ที่ต่ำกว่าได้โดยการแบ่งเป็นแพ็กเก็ตย่อยๆ ความสามารถด้านแฟรกเมนต์ของ IPv6 จะช่วยลดภาระการทำงานของเราเตอร์ เพราะโฮสต์ที่ทำการส่งแพ็กเก็ตจะเป็นผู้ทำการแบ่งแพ็กเก็ตเป็นแพ็กเก็ตย่อยๆ ส่วนโฮสต์ปลายทางจะเป็นโฮสต์ที่รวมแพ็กเก็ตซึ่งส่วนนี้จะเหมือนกับการแฟรกเมนต์ใน IPv4

11.2.6 การทดลองที่ 2.6 ทดสอบการใช้งาน IPv4 หลังจากการติดตั้งแสตก

ทดสอบการติดต่อกับโหนดที่เป็น IPv4 ที่อยู่ต่างเครือข่ายกันว่าสามารถติดต่อกันได้หรือไม่

สมมติฐาน เมื่อทำการติดตั้งดูอัล-แสตกแล้วก็ยังสามารถใช้งาน IPv4 ได้เหมือนเดิม

ตัวแปรต้น การใช้ IP เวอร์ชั่นที่ต่างกัน

ตัวแปรตาม ผลการติดต่อ

สภาพแวดล้อม

โฮสต์ 1

- ระบบปฏิบัติการ Windows XP
- IPv4 แอ็ดเดรส 161.246.5.111

โฮสต์ 2

- ระบบปฏิบัติการ Windows2000
- IPv4 แอ็ดเดรส 161.246.6.167

วิธีทดลอง

1. ตั้ง IPv6 dual-stack ให้กับเครื่องที่ใช้งาน Window XP

   ipv6 install

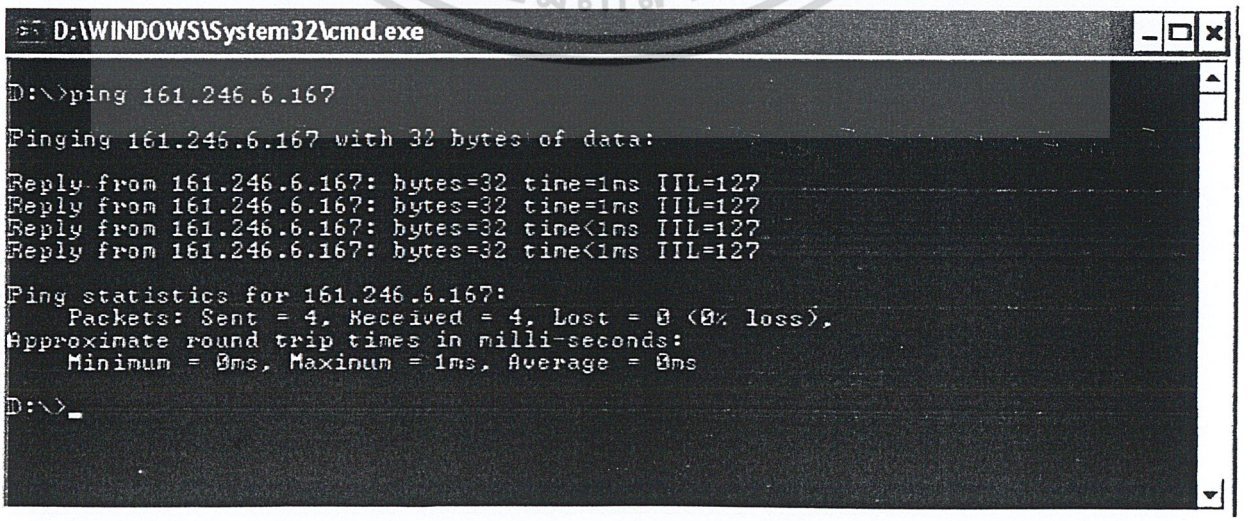
2. ขณะนี้โฮสต์ 1 สามารถใช้งาน IPv6 ได้ ทดสอบการทำงานร่วมกับ IPv4 โดยทำการ ping ไปยังโฮสต์ 2 ซึ่งเป็นโฮสต์ที่สนับสนุนเฉพาะ IPv4 เท่านั้นและอยู่ต่างเครือข่ายกันโดยใช้คำสั่ง

   ping 161.246.6.167

   และที่โฮสต์ 2 ก็ทำการติดต่อไปยังโฮสต์ 1 โดยใช้คำสั่ง

   ping 161.246.5.111

   ได้ผลการทดลองดังภาพ



รูปที่ 11.2.6 ผลการติดต่อกับโฮสต์ IPv4 หลังจากติดตั้ง IPv6 แล้ว

3. จากภาพแสดงว่าโฮสต์ที่เป็นดูอัล-แสตกสามารถติดต่อกับโอสต์ที่ทำงานเฉพาะ IPv4 ที่อยู่ต่างเครือข่ายกันได้

ข้อสรุป

เมื่อติดตั้งดูอัล-แสตกแล้วก็ยังสามารถใช้งาน IPv4 ได้เหมือนเดิมแม้จะเป็นการติดต่อข้ามเครือข่าย เพราะเป็นอิมพลีเมนต์ที่ออกแบบมาให้สนับสนุนทั้ง IPv6 และ IPv4

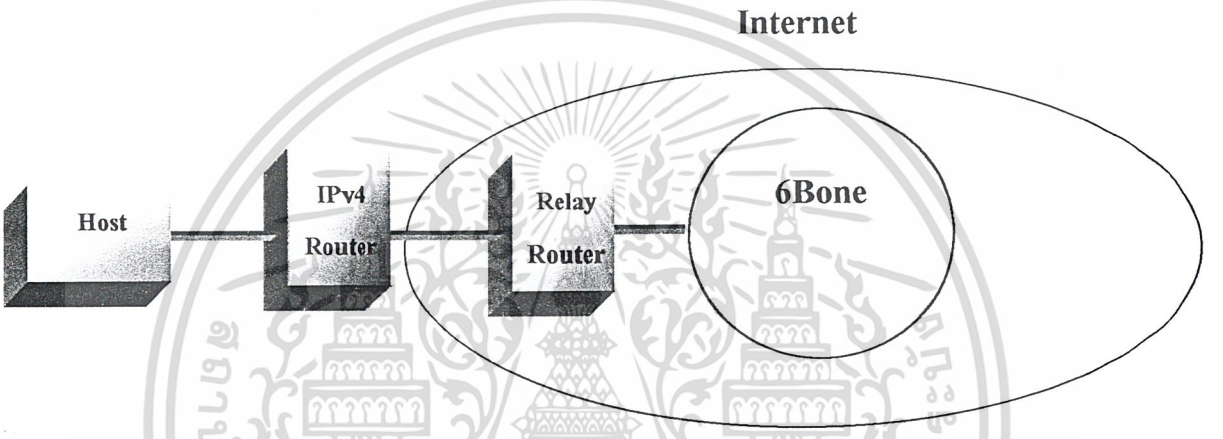
11.3 การทดลองที่ 3 : ติดต่อเครือข่ายเชื่อมกับ 6bone

จุดประสงค์

ทดลองการใช้งาน IPv6 ติดต่อกับโหนดภายในเครือข่าย 6Bone

สภาพแวดล้อม

ทดลองการติดต่อกับโหนดภายในเครือข่ายของ 6Bone โดยผ่านทางรีเลย์เราเตอร์ที่ทำหน้าที่เป็นปากทางเข้า



รูปที่ 11.3.1 แสดงการเชื่อมต่อสำหรับการทดลองที่ 3

การทดลอง

11.3.1 การทดลองที่ 3.1 ทดสอบการเชื่อมต่อกับ 6Bone ในระบบปฏิบัติที่ต่างกันภายในเครือข่ายต่างกัน ว่าสามารถเชื่อมต่อกันได้หรือไม่

สมมติฐาน สามารถติดต่อและทำงานได้แม้ว่าจะติดต่อกันผ่านทาง 6Bone

ตัวแปรต้น ระบบปฏิบัติการ Window 2000 , Linux Redhat 7.2 , Windows XP

ตัวแปรตาม ผลการติดต่อ

สภาพแวดล้อม

โฮสต์ 1

- ระบบปฏิบัติการ Windows XP
- IPv4 แอ็ดเดรส 161.246.5.111

โฮสต์ 2

- ระบบปฏิบัติการ Windows 2000
- IPv4 แอ็ดเดรส 161.246.6.167

โฮสต์ 3

- ระบบปฏิบัติการ Linux Redhat 7.2
- IPv4 แอ็ดเดรส 161.246.5.19

วิธีทดลอง

1. ติดตั้ง IPv6 ให้กับโฮสต์ทดลองที่ 1 ที่ติดตั้งระบบปฏิบัติการ Window XP และเชื่อมต่อเข้ากับรีเลย์เราเตอร์ของ Hurricane โดยพิมพ์คำสั่ง

   ipv6 adu 2/3FFE:1200:3028:FF01::46D9

   ipv6 rtu ::/0 3/::64.71.128.26 pub

   โฮสต์ 1 จะมี IPv6 แอ็ดเดรสเป็น 3FFE:1200:3028:FF01::46D9
2. ติดตั้ง IPv6 ให้กับโฮสต์ทดลองที่ 2 ที่ติดตั้งระบบปฏิบัติการ Window 2000 และเชื่อมต่อเข้ากับรีเลย์เราเตอร์ของ Freenet (การเชื่อมต่อเข้ากับ Freenet สามารถใช้เครื่องมือของ Freenet ที่ดาวน์โหลดได้จากเว็บของ Freenet) โดยพิมพ์คำสั่ง

   tspc –rf tspc.conf

   ipv6 rtu ::/0 2/:206.123.31.114 pub life 1800

   โฮสต์ 2 จะมี IPv6 แอ็ดเดรสเป็น 2002:A1F6:6A7:3:0:3:A1F6:6A7

   ติดตั้ง IPv6 ให้กับโฮสต์ 3 ที่ติดตั้งระบบปฏิบัติการ Linux Redhat 7.2 โดยจะใช้การเชื่อมต่อเข้ากับ Hurricane Electric โดยทำได้ดังนี้

   ในไฟล์ /etc/sysconfig/network

NETWOKING_IPV6=yes

ในไฟล์ /sys/sysconfig/network-scripts/ifcfg-sit1

IPV6INIT="yes"

IPV6TUNNELIPV4="64.71.128.26"

IPV6ADDR="3FFE:1200:3028:FF01::4D69/127"

ใช้คำสั่งต่อไปนี้

/etc/sysconfig/network-scripts/ifup sit1

route –A inet6 add ::/0 dev sit 1

3. เมื่อติดตั้ง IPv6 ให้กับโฮสต์ทั้ง 3 เรียบร้อยแล้วทำการทดสอบโดยการ ping6 ติดต่อระหว่างกัน โดยการทำงานนั้นโฮสต์ทั้ง 3 จะทำการเอ็นแค็ป IPv6 แพ็กเก็ตเพื่อส่งต่อให้กับรีเลย์เราเตอร์ของแต่ละโฮสต์ แพ็กเก็ตนั้นจะถูกส่งผ่านเข้า 6Bone และส่งมายังโฮสต์ปลายทางโดยการเอ็นแค็ปเช่นกัน
4. ได้ผลสรุปดังตาราง

ระบบปฏิบัติการโฮสต์ต้นทาง	ระบบปฏิบัติการโฮสต์ปลายทาง	ผลการติดต่อ
Window XP	Window 2000	สามารถติดต่อได้
Window XP	Linux Redhat 7.2	สามารถติดต่อได้
Window 2000	Window XP	สามารถติดต่อได้
Window 2000	Linux Redhat 7.2	สามารถติดต่อได้
Linux Redhat 7.2	Window XP	สามารถติดต่อได้
Linux Redhat 7.2	Window 2000	สามารถติดต่อได้

ตารางที่ 11.3.1 ผลการติดต่อระหว่างโฮสต์ที่ใช้ระบบปฏิบัติการต่างกันโดยผ่าน 6Bone

จากตารางแสดงว่าแม้จะใช้โฮสต์ที่ทำงานภายใต้ระบบปฏิบัติการที่ต่างกัน มีการอิมพลีเมนต์ IPv6 แสตกที่แตกต่างกัน และเข้าเชื่อมต่อกับ 6Bone โดยใช้รีเลย์เราเตอร์ต่างกันก็สามารถเชื่อมต่อกันได้

ข้อสรุป

ความสามารถในการติดต่อของ IPv6 จะไม่ขึ้นอยู่กับว่าต้องติดต่อผ่านกันทาง 6 หรือไม่ หรือจะใช้ระบบปฏิบัติการเดียวกันหรือไม่ ก็ยังจะสามารถทำงานร่วมกันได้

11.3.2 การทดลองที่ 3.2 ทดลองการทำ 6to4 ผ่านรีเลย์เราเตอร์โดยกำหนดให้โฮสต์ผู้ส่งทำหน้าที่เป็น 6to4 เราเตอร์

ทดลองให้โหนดที่ทำหน้าที่ส่งทำหน้าที่เป็น 6to4 เราเตอร์เอง

สมมติฐาน สามารถติดต่อกับโหนดที่อยู่ภายใน 6Bone ได้โดยผ่านรีเลย์เราเตอร์

วิธีทดลอง

1. ติดตั้ง IPv6 ให้กับโฮสต์ที่ใช้ระบบปฏิบัติการ Windows XP โดยพิมพ์คำสั่ง

   ipv6 install

   เมื่อใช้คำสั่ง ***ipv6 if*** จะพบว่ามีการกำหนดอินเทอร์เฟซและแอ็ดเดรสให้

2. เชื่อมต่อกับ 6Bone โดยใช้รีเลย์เราเตอร์ของ Hurricane Electric โดยก่อนใช้ต้องไปลงทะเบียนในเว็บไซต์ก่อน โดยลงทะเบียนได้ Acount มาเป็น Babypongkemon เมื่อลงทะเบียนแล้ว ทำการกำหนดเส้นทางการส่งแพ็กเก็ตโดยกำหนดให้ทุกแพ็กเก็ต ส่งไปยัง IPv6 แอ็ดเดรส ::64.71.128.26 ซึ่งเป็นรีเลย์เราเตอร์ของ Hurricane โดยพิมพ์คำสั่ง

   ipv6 rtu ::/0 2/::64.71.128.26 pub

   จากนั้นทำการเพิ่มแอ็ดเดรสที่ได้มาจากลงทะเบียนให้กับอินเทอร์เฟซหมายเลข 2 โดยพิมพ์คำสั่ง

   ipv6 adu 2/3FFE:1200:3028:FF01::46D9

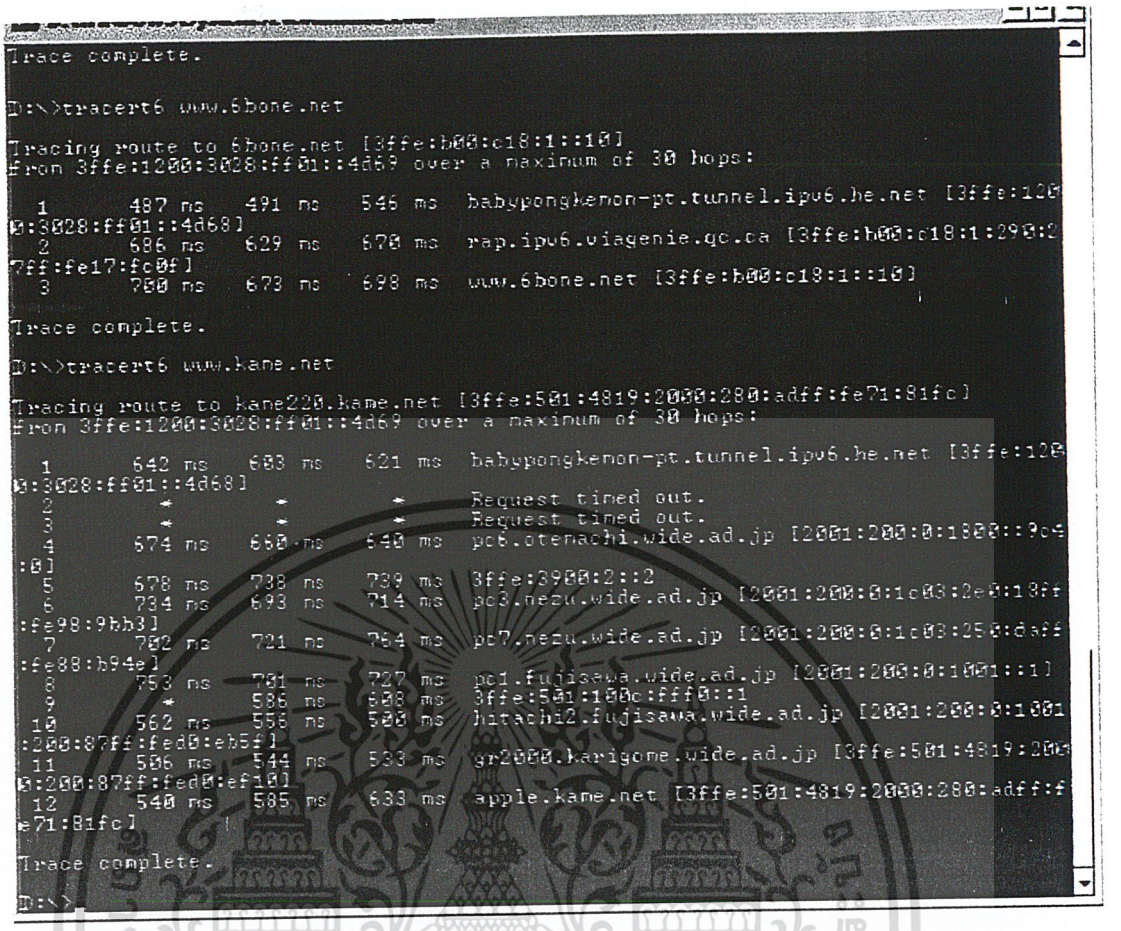
   เมื่อมีการส่ง IPv6 แพ็กเก็ตก็จะทำการทันเนิลตามวิธี 6to4 และส่งไปยังรีเลย์เราเตอร์และเข้าสู่ 6Bone ต่อไป

3. ทดสอบการทำงานโดยคำสั่ง tracert ไปที่ www.6Bone.net และ www.kame.net โดยพิมพ์คำสั่ง

   tracert6 www.6bone.net

   tracert6 www.kame.net

   จะได้ผลลัพธ์ดังรูป



รูปที่ 11.3.2 แสดงผลลัพธ์จากคำสั่ง tracert ไปยัง IPv6 ไซต์

4. จากภาพจะเห็นว่าแอ็ดเดรสที่ได้รับกลับมาจาก DNS นั้นเป็นแบบ IPv6 และเส้นทางการเดินทางของแพ็กเก็ตไปยังไซต์ทั้งสองนั้นจะต้องผ่าน babypongkemon-pt.tunnel.ipv6.he.net ซึ่งเป็นรีเลย์เราเตอร์ก่อนเป็นอันดับแรกจึงจะเดินทางผ่าน 6Bone ไปยัง IPv6 ไซต์ต่างๆ

ข้อสรุป

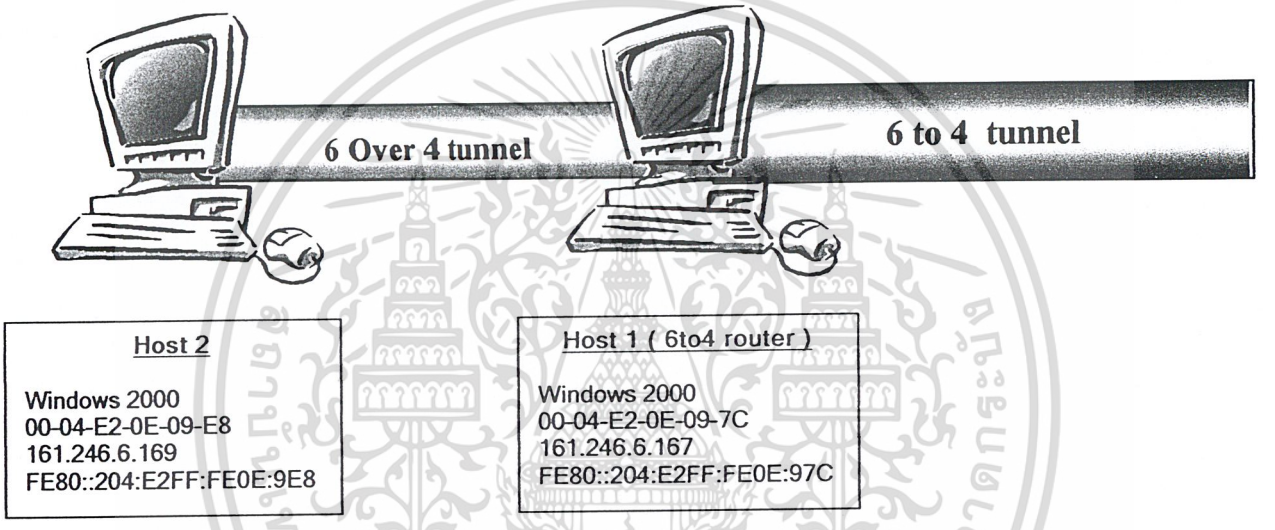
สามารถปรับแต่งให้โหนดใดๆให้เป็นส่วนหนึ่งของ 6Bone ได้โดยผ่านทางรีเลย์เราเตอร์ นั่นคือเมื่อโหนดต้องการส่งแพ็กเก็ตไปยังโหนดที่อยู่ใน 6Bone ก็จะสร้างแพ็กเก็ต IPv6 ภายใน IPv4 ส่งไปยังรีเลย์เราเตอร์ และรีเลย์เราเตอร์กธจะทำการฟอร์เวิดแพ็กเก็ตเข้าสู่ 6Bone การทดลองนี่เหมาะสำหรับโอสต์เดี่ยวๆหรือเป็นผู้ใช้ตามบ้านที่ต้องการติดต่อกับเครือข่าย IPv6 ก็สามารถทำได้โดยการลงทะเบียนเพื่อขอใช้รีเลย์เราเตอร์ เพียงเท่านี้ก็จะสามารถเข้าสู่ 6Bone ได้

11.3.3 การทดลองที่ 3.3 ทดลองการทำ 6to4 ผ่านรีเลย์เราเตอร์โดยกำหนดให้มีโฮสต์ที่ทำหน้าที่เป็น 6to4 เราเตอร์บริการ

ทดลองให้โหนดมีโหนดทำหน้าที่ให้บริการเป็น 6to4 เราเตอร์ให้กับโหนด IPv6 ตัวอื่นภายในเครือข่าย

สมมติฐาน สามารถกำหนดให้โหนดใดๆในเครือข่ายทำหน้าที่เป็น 6to4 เราเตอร์เพื่อติดต่อกับโหนด IPv6 ที่อยู่ใน 6Bone ได้

สภาพแวดล้อม



รูปที่ 11. 3.3 การทดลองกำหนดการเชื่อมต่อกับ 6Bone โดยผ่านโฮสต์บริการที่เป็น 6to4 เราเตอร์

วิธีทดลอง

1. ติดตั้ง Microsoft IPv6 dual-stack ให้กับโฮสต์ทดลองที่ติดตั้งระบบปฏิบัติการ Window 2000 ทั้งโฮสต์ 1 และ โฮสต์2 โดยกำหนดให้โฮสต์ 1 จะทำหน้าที่เป็น 6to4 เราเตอร์บริการ มีหมายเลขอินเทอร์เฟซเป็น 00-04-E2-0E-09-7C และ แอ็ดเดรส IPv4 เป็น 161.246.6.167 ส่วนโฮสต์หมายเลข 2 มีหมายเลขอินเทอร์เฟซเป็น 00-04-E2-0E-09-E8 และมีแอ็ดเดรส IPv4 เป็น 161.246.6.169
2. ติดตั้ง freenet client ให้กับโฮสต์ 1 เพื่อทำการเชื่อมต่อกับ 6Bone โดยพิมพ์คำสั่ง

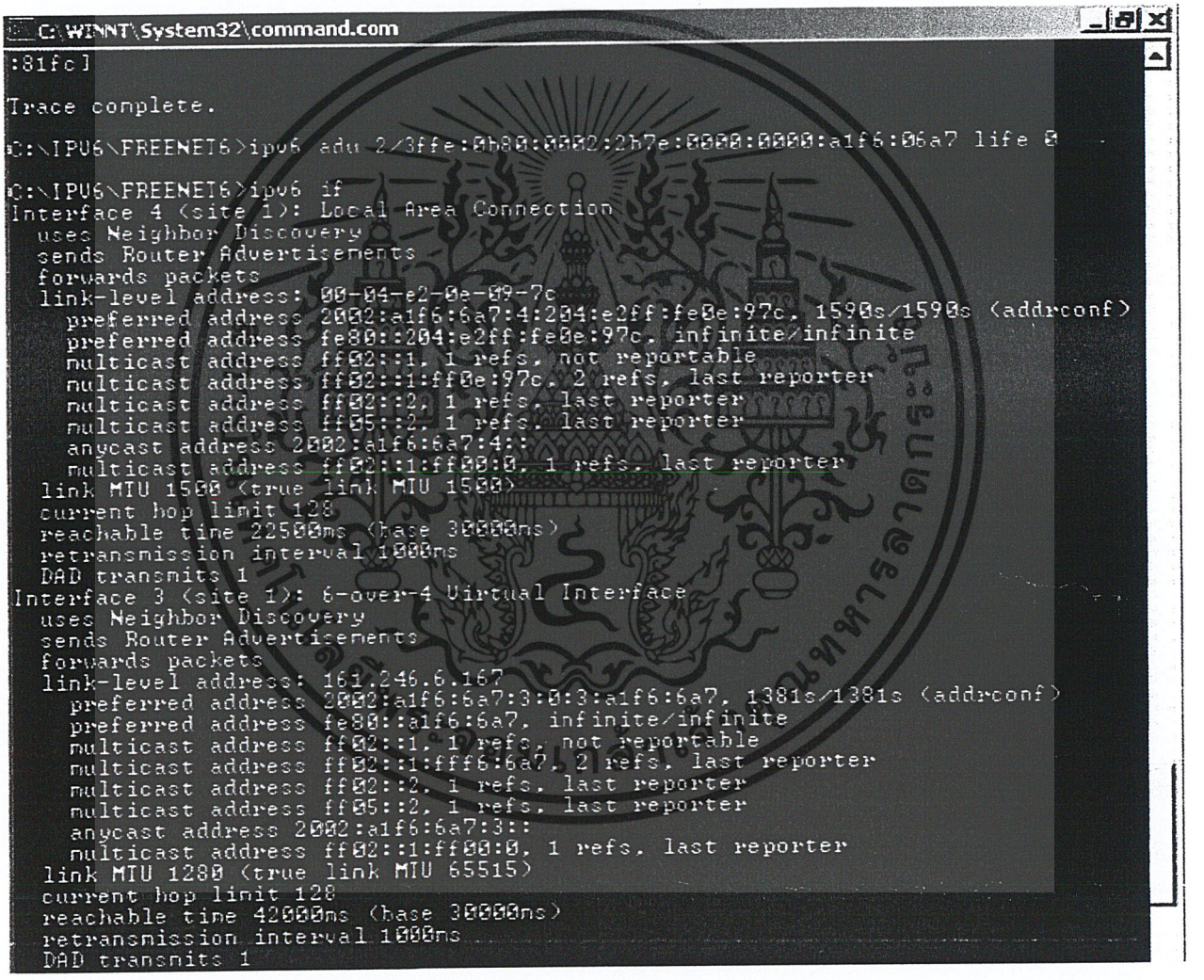
   tspc –rf tspc.conf

   คำสั่งนี้ใช้ในการลงทะเบียนกับ freenet เพื่อให้รีเลย์เราเตอร์ของ freenet ยอมรับแพ็กเก็ตที่ถูกส่งมาจากโฮสต์ 1

3. คอนฟิกโฮสต์ 1 ให้ทำหน้าที่เป็นเกตุเวย์บริการแก่โฮสต์ตัวอื่นภายในเครือข่าย โดยการพิมพ์คำสั่ง

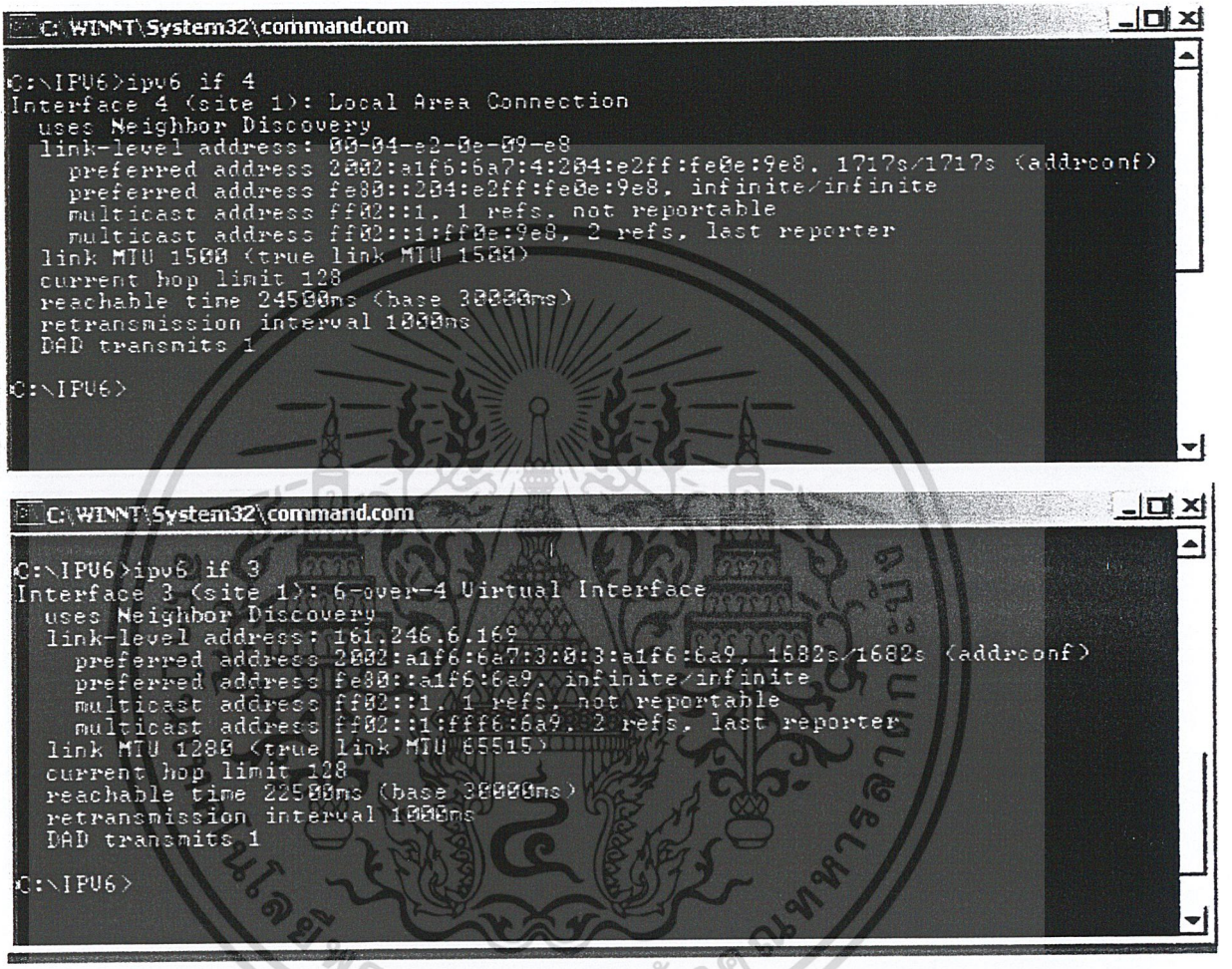
6to4cfg -r

คำสั่งนี้จะทำให้โฮสต์ 1 ทำหน้าที่เป็นเราเตอร์และกระจายค่าไซต์พรีฟิกซ์ 2002:A1F6:6A7:4 และ 2002:A1F6:6A7:3 ซึ่งได้มาจากค่า IPv4 แอ็ดเดรสของโฮสต์ 1 ไปให้กับโฮสต์อื่นๆภายในเครือข่ายเดียวกัน และยังทำหน้าที่ Forward แพ็กเก็ตให้แสดงดังรูป



รูปที่ 11.3.4 แสดงการกำหนดให้โฮสต์ทำหน้าที่เป็นเกตเวย์

เมื่อใช้คำสั่ง ***ipv6 if 3*** และ ***ipv6 if 4*** ที่โฮสต์ 2 ก็จะปรากฏ IPv6 แอ็ดเดรส 2002:A1F6:6A7:3:0:3:A1F6:6A9 ที่อินเทอร์เฟซหมายเลข 3 และแอ็ดเดรส 2002:A1F6:6A7:4:204:E2FF:FE0E:9E8 ที่อินเทอร์เฟซหมายเลข 4 แสดงดังรูป



รูปที่ 11.3.5 แสดงแอดเดรสที่เกิดจากการกระจายค่าพรีฟิกซ์จากโฮสต์ที่เป็น 6to4 เราเตอร์

การทดลองนี้จะแบ่งการติดต่อออกเป็นสองส่วนคือการติดต่อระหว่างโฮสต์ใดๆภายในเครือข่ายกับโฮสต์บริการที่เป็น 6to4 เราเตอร์และการติดต่อระหว่างโอสต์บริการกับ 6Bone สำหรับการติดต่อในส่วนแรกนั้นสามารถทำได้สองวิธีคือ

- ติดต่อโดยการใช้ลิ้งค์-โลคัลแอ็ดเดรสโดยใช้แอ็ดเดรส 2002:A1F6:6A7:4:204:E2FF:FE0E:9E8 ในอินเทอร์เฟซหมายเลข 4

- ติดต่อโดยการทำ 6 over 4 ทันเนิลระหว่างโฮสต์ใดๆกับโฮสต์ที่เป็นโฮสต์บริการผ่านอินเทอร์เฟซหมายเลข 3 โดยใช้แอ็ดเดรส 2002:A1F6:6A7:3:0:3:A1F6:6A9

จะทำการทดลองโดยใช้วิธีที่สอง

4. ที่โฮสต์ 1พิมพ์คำสั่ง

   ipv6 rtu ::/0 3/:206.123.31.114 pub life 1800

   คำสั่งนี้ทำหน้าที่ forward แพ็กเก็ตที่ได้รับไปให้กับรีเลย์เราเตอร์ของ freenet

5. จากนั้นทำการคอนฟิกให้โฮสต์ 2 ส่งแพ็กเก็ตมาให้กับโฮสต์ 1 โดยการทำ 6 over 4 ทันเนิลโดยพิมพ์คำสั่ง

   ipv6 rtu ::/0 3/2002:A1F6:6A7:3:0:3:A1F6:6A7

   การทำงานจะเริ่มเมื่อโฮสต์ 2 ต้องการติดต่อกับ 6Bone ก็จะส่งแพ็กเก็ต ipv6 ไปให้กับโฮสต์ 1 โดยการทำ 6 over 4 ทันเนิล จากนั้นโฮสต์ 1 ก็จะ forward แพ็กเก็ตส่วนที่เป็น IPv6 ไปให้กับรีเลย์เราเตอร์ของ freenet โดยการทำ 6to4 ทันเนิล เข้าสู่ 6Bone โฮสต์ที่อยู่ใน 6Bone ก็จะติดต่อกลับมาโดยการทำ 6to4 ทันเนิลมาที่โฮสต์ 1 และจากนั้นโฮสต์ 1 ก็จะทำ 6 over 4 ทันเนิลส่งแพ็กเก็ตต่อให้กับโฮสต์ 2

6. ทำการทดสอบโดยการ trace route ไปยังโฮสต์ที่อยู่ใน 6Bone โดยการพิมพ์คำสั่งที่โฮสต์ 2

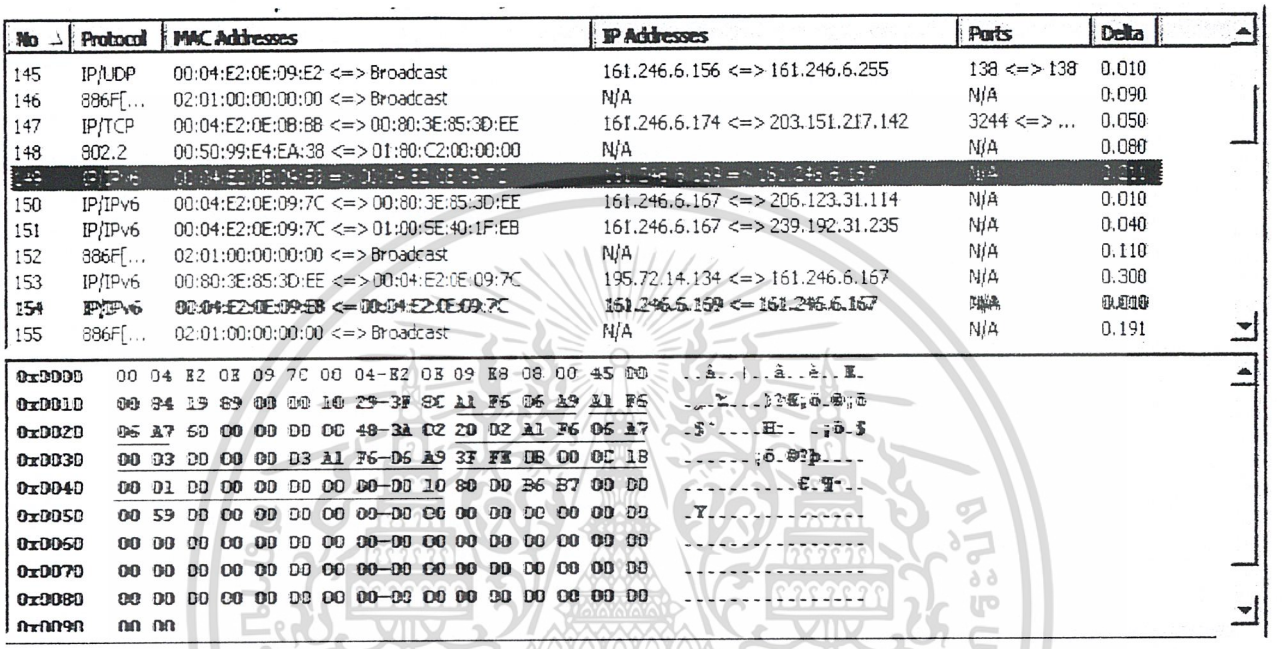
   tracert6 www.6bone.net

   ได้ผลการทดสอบดังภาพ

รูปที่ 11.3.6 ผลการทดสอบการ trace route ไปยัง 6Bone โดยผ่านโฮสต์บริการ

จากภาพเส้นทางการเดินของแพ็กเก็ตจะไปยังโฮสต์ 1 ซึ่งเป็นโฮสต์บริการเป็นอันดับแรกและจะถูกส่งต่อไปยังรีเลย์เราเตอร์ของ freenet และจึงถูกส่งต่อไปยังโฮสต์ปลายทางใน 6Bone

เมื่อใช้โปรแกรมจับแพ็กเก็ตมาตรวจสอบจะได้



No.	Protocol	MAC Addresses	IP Addresses	Ports	Delta
145	IP/UDP	00:04:E2:0E:09:E2 <=> Broadcast	161.246.6.156 <=> 161.246.6.255	138 <=> 138	0.010
146	886F[...	02:01:00:00:00:00 <=> Broadcast	N/A	N/A	0.090
147	IP/TCP	00:04:E2:0E:08:B8 <=> 00:80:3E:85:3D:EE	161.246.6.174 <=> 203.151.217.142	3244 <=> ...	0.050
148	802.2	00:50:99:E4:EA:38 <=> 01:80:C2:00:00:00	N/A	N/A	0.080
149	IP/IPv6	[illegible]	[illegible]	[illegible]	[illegible]
150	IP/IPv6	00:04:E2:0E:09:7C <=> 00:80:3E:85:3D:EE	161.246.6.167 <=> 206.123.31.114	N/A	0.010
151	IP/IPv6	00:04:E2:0E:09:7C <=> 01:00:5E:40:1F:EB	161.246.6.167 <=> 239.192.31.235	N/A	0.040
152	886F[...	02:01:00:00:00:00 <=> Broadcast	N/A	N/A	0.110
153	IP/IPv6	00:80:3E:85:3D:EE <=> 00:04:E2:0E:09:7C	195.72.14.134 <=> 161.246.6.167	N/A	0.300
154	IP/IPv6	[illegible]	[illegible]	[illegible]	[illegible]
155	886F[...	02:01:00:00:00:00 <=> Broadcast	N/A	N/A	0.191

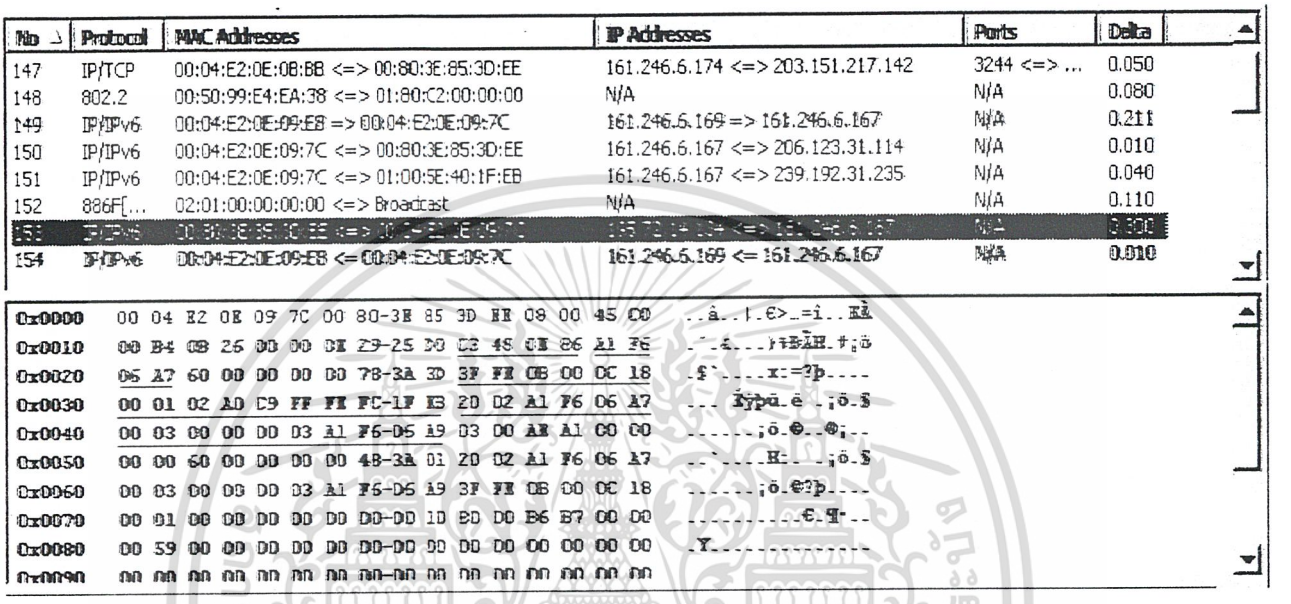
0xD000	00	04	E2	0E	09	7C	00	04	E2	0E	09	E8	08	00	45	00
0xD010	00	84	19	89	00	00	10	29	3F	9C	A1	F6	06	A9	A1	F6
0xD020	06	A7	60	00	0D	D0	0C	48	3A	02	20	02	A1	F6	06	A7
0xD030	00	03	DD	00	0D	D3	A1	F6	06	A9	3F	FE	0B	00	0C	1B
0xD040	00	01	DD	00	0D	DD	0D	00	06	10	80	0D	B6	B7	00	0D
0xD050	00	59	0D	00	0D	0D	0C	00	06	00	00	0D	0C	00	0D	0D
0xD060	00	0D	0D	00	0D	0D	0C	00	06	00	00	0D	0C	00	0D	0D
0xD070	00	0D	0D	00	0D	0D	0C	00	06	00	00	0D	0C	00	0D	0D
0xD080	00	0D	0D	00	0D	0D	0C	00	06	00	00	0D	0C	00	0D	0D
0xD090	00	0D														

รูปที่ 11.3.7 แสดงแพ็กเก็ตที่ถูกส่งจากโฮสต์ใดๆในเครือข่ายไปยังโฮสต์บริการโดยการทำ 6 over 4 ทันเนิล

จากภาพแสดงแพ็กเก็ตที่ถูกส่งไปยังโฮสต์บริการ จะเห็นได้ว่ามีการทำ 6 over 4 ทันเนิลไปยังโอสต์ 1 โดยสีน้ำเงินแสดงส่วนที่เป็น IPv4 และส่วนที่เป็นสีดำจะเป็น IPv6

- แอ็ดเดรสต้นทางของส่วน IPv4 จะเป็นของโฮสต์ 2 คือ ค่า 161.246.6.169 แสดงในภาพเป็น A1 F6 06 A9 ในตำแน่งที่ 0x001A และแอ็ดเดรสปลายทางจะเป็น
  ของโฮสต์ 1 คือ 161.246.6.167 แสดงในภาพเป็น A1 F6 06 A7 ในตำแหน่งที่ 0x001E
- ในส่วนที่เป็น IPv6 นั้นแอ็ดเดรสต้นทางจะเป็น
  2002:A1F6:6A7:3:0:3:A1F6:6A9 ในขณะที่แอ็ดเดรสปลายทางนั้นไม่ได้เป็นของโฮสต์ 1 แต่จะเป็นของโฮสต์ใน 6Bone เลย
  ซึ่งจากภาพเป็นของ www.6Bone.net คือ 3FFE:B00:C18:1::10 แสดงในภาพที่ตำแหน่ง 0x004A

เมื่อโอสต์ภายใน 6Bone ได้รับแพ็กเก็ตจากโฮสต์บริการแล้ว ก็จะตอบกลับมาโดยการทำ 6to4 ทันเนิลมายังโฮสต์บริการแสดงได้ดังภาพ



No.	Protocol	MAC Addresses	IP Addresses	Ports	Delta
147	IP/TCP	00:04:E2:0E:0B:BB <=> 00:80:3E:85:3D:EE	161.246.6.174 <=> 203.151.217.142	3244 <=> ...	0.050
148	802.2	00:50:99:E4:EA:38 <=> 01:80:C2:00:00:00	N/A	N/A	0.080
149	IP/IPv6	00:04:E2:0E:09:E8 => 00:04:E2:0E:09:7C	161.246.6.169 => 161.246.6.167	N/A	0.211
150	IP/IPv6	00:04:E2:0E:09:7C <=> 00:80:3E:85:3D:EE	161.246.6.167 <=> 206.123.31.114	N/A	0.010
151	IP/IPv6	00:04:E2:0E:09:7C <=> 01:00:5E:40:1F:EB	161.246.6.167 <=> 239.192.31.235	N/A	0.040
152	886F[...	02:01:00:00:00:00 <=> Broadcast	N/A	N/A	0.110
153	IP/IPv6	00:80:3E:85:3D:EE <=> 00:04:E2:0E:09:7C	161.246.6.167 <=> 161.246.6.167	N/A	0.000
154	IP/IPv6	00:04:E2:0E:09:E8 <=> 00:04:E2:0E:09:7C	161.246.6.169 <=> 161.246.6.167	N/A	0.010

```
0x0000  00 04 E2 0E 09 7C 00 80-3E 85 3D EE 08 00 45 00   ..â..|.€>…=î..E.
0x0010  00 B4 0B 26 00 00 3E 29-25 D0 C2 4B 0E 86 A1 F6   .´.&..>)%ÐÂK.†¡ö
0x0020  06 A7 60 00 00 00 00 78-3A 3D 3F FE 0B 00 0C 18   .§`....x:=?þ....
0x0030  00 01 02 A0 C9 FF FE FC-1F EB 2D D2 A1 F6 06 A7   ... ÉÿþüëÒ¡ö§
0x0040  00 03 00 00 00 03 A1 F6-06 A9 03 00 A8 A1 C0 00   ......¡ö.©..¨¡À.
0x0050  00 00 60 00 00 00 00 4B-3A 01 2D D2 A1 F6 06 A7   ..`....K:.-Ò¡ö.§
0x0060  00 03 00 00 00 03 A1 F6-06 A9 3F FE 0B 00 0C 18   ......¡ö.©?þ....
0x0070  00 01 00 00 00 00 00 00-00 1D ED D0 B6 E7 00 00   ..........íÐ¶ç..
0x0080  00 59 00 00 00 00 00 00-00 00 00 00 00 00 00 00   .Y..............
0x0090  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

รูปที่ 11.3.8 แสดงแพ็กเก็ตที่ตอบกลับมาจากโอสต์ปลายทางมาสู่โฮสต์บริการ

จากภาพ

- แอ็ดเดรสในส่วน IPv4 เป็นแอ็ดเดรสของโฮสต์ 1 และโฮสต์ใน 6Bone ซึ่งเป็นการทันเนิลระหว่างโฮสต์ทั้งสองเท่านั้นโดยไม่เกี่ยวกับโฮสต์ 2
- แต่แอ็ดเดรสในส่วนที่เป็น IPv6 นั้นแอ็ดเดรสต้นทางเป็น 3FFE:B00:C18:1:2A0:C9FF:FEFC:1FEB แสดงในภาพเป็นตำแหน่ง 0x002A และมีแอ็ดเดรสปลายทางเป็น 2002:A1F6:6A7:3:0:3:A1F6:6A9 แสดงในภาพเป็นตำแหน่ง 0x003A ซึ่งเป็นแอ็ดเดรสของโอสต์ 2 ที่อินเทอร์เฟซ 3 จะเห็นได้ว่าการติดต่อในส่วน IPv6 ไม่ได้เกี่ยวข้องกับโฮสต์ 1 เลย

No	Protocol	MAC Addresses	IP Addresses	Ports	Delta
135	IP/TCP	00:80:3E:85:3D:EE <=> 00:04:E2:0E:0B:BB	209.75.20.28 <=> 161.246.6.174	80 <=> 3417	0.000
150	IP/IPv6	00:80:3E:85:3D:EE <=> 00:04:E2:0E:09:7C	195.72.14.134 <=> 161.246.6.167	N/A	0.100
151	IP/IPv6	00:04:E2:0E:09:E8 => 00:04:E2:0E:09:7C	161.246.6.169 => 161.246.6.167	N/A	0.000
152	886F[...	02:01:00:00:00:00 <=> Broadcast	N/A	N/A	0.010
153	IP/TCP	00:04:E2:0E:0B:BB <=> 00:80:3E:85:3D:EE	161.246.6.174 <=> 203.151.217.142	3244 <=> ...	0.100
154	IP/TCP	00:80:3E:85:3D:EE <=> 00:04:E2:0E:0B:BB	203.151.217.142 <=> 161.246.6.174	8000 <=> ...	0.160
155	IP/IPv6	00:04:E2:0E:09:E8 => 00:04:E2:0E:09:7C	161.246.6.169 => 161.246.6.167	N/A	0.151
156	IP/IPv6	00:04:E2:0E:09:E8 <= 00:04:E2:0E:09:7C	161.246.6.169 <= 161.246.6.167	N/A	0.010
157	IP/UDP	00:E0:29:40:FD:A6 <=> Broadcast	161.246.6.201 <=> 161.246.6.255	138 <=> 138	0.060

```
0x0000  00 04 E2 0E 09 E8 00 04-E2 0E 09 7C 08 00 45 00   ..â..è..â..|..E.
0x0010  00 64 06 BC 00 00 10 29-52 79 A1 F6 06 A7 A1 F6   .d.¼...)Ry¡ö.§¡ö
0x0020  06 A9 60 00 00 00 00 28-3A 3C 3F FE 0B 00 0C 18   .©`....(:<?þ....
0x0030  00 01 00 00 00 00 00 00-20 02 A1 F6 06 A7         ........ .¡ö.§
0x0040  00 03 00 00 00 03 A1 F6-06 A9 81 00 0B 23 00 00   ......¡ö.©...#..
0x0050  00 6A 61 62 63 64 65 66-67 68 69 6A 6B 6C 6D 6E   .jabcdefghijklmn
0x0060  6F 70 71 72 73 74 75 76-77 61 62 63 64 65 66 67   opqrstuvwabcdefg
0x0070  68 69                                             hi
```

รูปที่ 11.3.9 แสดงแพ็กเก็ตที่ส่งต่อจากโฮสต์บริการมายังโฮสต์ต้นทาง

จากภาพแสดงแพ็กเก็ตที่โฮสต์บริการทำการส่งต่อมายังโฮสต์ 2 โดยการทันเนิล 6 over 4 เหมือนตอนที่ส่งไป โดยแอ็ดเดรสต้นทางในส่วนที่เป็น IPv6 ยังคงเป็นโฮสต์ที่อยู่ใน 6Bone ส่งมายังโฮสต์ 2 อยู่ แสดงว่าเส้นทางการเดินทางของแพ็กเก็ตเป็นไปตามที่ได้คอนฟิกเอาไว้และสามารถกำหนดให้โฮสต์ตัวหนึ่งทำหน้าที่เป็นเกตเวย์ให้กับโฮสต์อื่นๆภายในเครือข่ายได้

ข้อสรุป

สามารถปรับแต่งโหนดใดๆในเครือข่ายทำหน้าที่เป็น 6to4 เราเตอร์และปรับแต่ง 6to4 เราเตอร์นั้นติดต่อกับรีเลย์เราเตอร์เพื่อติดต่อกับโหนดภายใน 6Bone โดยโหนดที่เหลือภายในเครือข่ายก็จะสามารถเข้าสู่ 6Bone ได้ด้วย โดยแพ็กเก็ตจะถูกส่งไปยัง 6to4 เราเตอร์และรีเลย์เราเตอร์ตามลำดับ การทดลองนี้เหมาะสำหรับเครือข่ายที่ต้องการติดต่อกับโหนดภายใน 6Bone ก็ปรับแต่งเพียงแค่โหนดหนึ่งเป็น 6to4 เราเตอร์และปรับแต่งให้ติดต่อกับรีเลย์เราเตอร์ที่ยอมส่งแพ็กเก็ตภายในเครือข่ายเข้าสู่ 6Bone

11.3.4 การทดลองที่ 3.4 ทดสอบการใช้งาน IPv4 หลังจากการติดตั้งแสตก

ทดสอบการติดต่อกับโหนดที่เป็น IPv4 ที่อยู่ต่างเครือข่ายกันภายในอินเทอร์เน็ต

สมมติฐาน เมื่อทำการติดตั้งดูอัล-แสตกและคอนฟิก IPv6 เข้าสู่ 6Bone แล้วก็ยังสามารถใช้งาน IPv4 ได้เหมือนเดิม

วิธีทดลอง

1. ติดตั้ง IPv6 dual-stack ให้กับเครื่องที่ใช้งาน Window XP และคอนฟิกให้เชื่อมต่อกับ 6Bone ตามการทดลองที่ผ่านมาโดยพิมพ์คำสั่งดังนี้

   ipv6 install

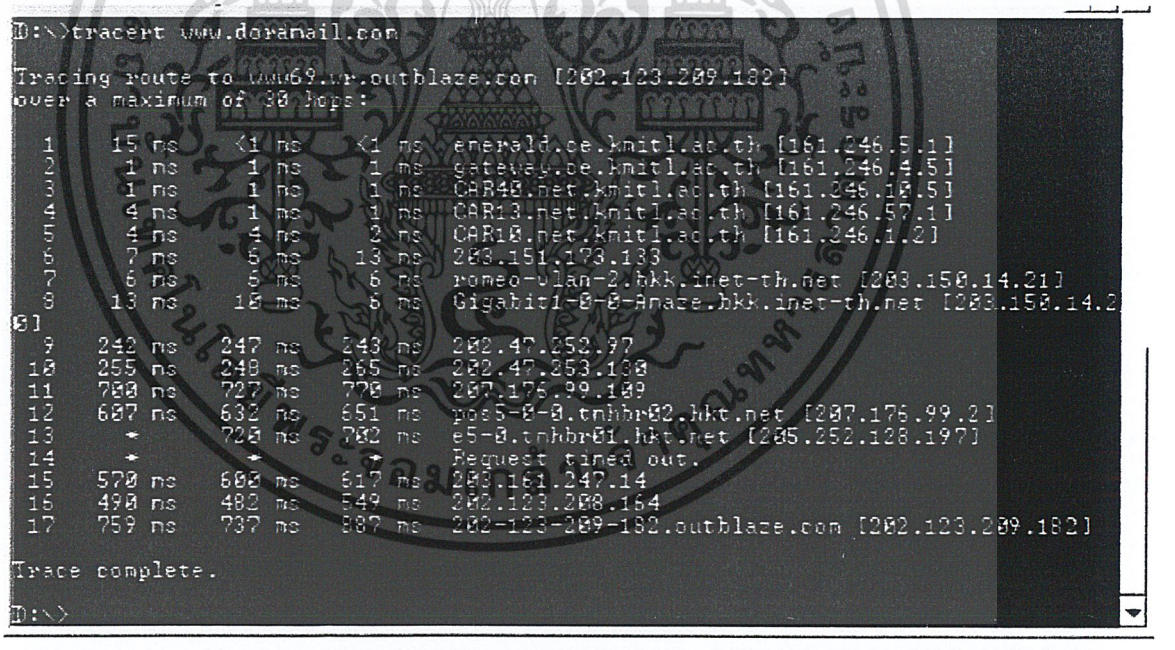
   ipv6 rtu ::/0 2/::64.71.128.26 pub

   ipv6 adu 2/3FFE:1200:3028:FF01::46D9

2. ทดสอบว่าสามารถติดต่อกับไซต์ที่เป็น IPv4 เท่านั้นได้โดยการใช้คำสั่ง

   tracert www.doramail.com

   จะได้ผลลัพธ์ดังภาพ



รูปที่ 11.3.10 แสดงผลลัพธ์การติดต่อกับ IPv4 ไซต์หลังจากการติดตั้งแสตกและเชื่อมต่อกับ 6Bone

3. จากภาพแอ็ดเดรสที่ได้รับมาจาก DNS เป็นแอ็ดเดรสแบบ IPv4 และจากเส้นทางการผ่านของแพ็กเก็ตไม่ได้ผ่านไปยังรีเลย์เราเตอร์ของ hurricane แต่ส่งผ่านไปยังเป้าหมายเลยแสดงว่าตารางการเลือกเส้นทางทั้งของแพ็กเก็ตทั้งสองเวอร์ชั่นแยกจากกันเป็นอิสระและสามารถใช้งานร่วมกันได้

ข้อสรุป

เมื่อติดตั้งดูอัล-แสตกแล้วก็ยังสามารถใช้งาน IPv4 ได้เหมือนเดิม เพราะเป็นอิมพลีเมนต์ที่ออกแบบมาให้สนับสนุนทั้ง IPv6 และ IPv4 โดยการคอนฟิกเข้าสู่ 6Bone นั้นจะถูกกำหนดสำหรับ IPv6 แพ็กเก็ตเท่านั้น

11.4 การทดลองที่ 4 : การทดสอบ IPv6 ระดับชั้นโปรแกรมประยุกต์

จุดประสงค์

ทดสอบการทำงานต่างๆในระดับชั้นบริการในสภาพแวดล้อมที่เป็น IPv6 ในแบบต่างๆ

สภาพแวดล้อมที่ 1 เครื่อง IPv6 ภายในเครือข่ายเดียวกัน (การทดลองที่ 4.2)

สภาพแวดล้อมที่ 2 เชื่อมต่อกับ 6Bone โดยวิธี 6to4 ผ่านรีเลย์เราเตอร์ (การทดลองที่ 4.1)

รูปที่ 11.4.1 แสดงการเชื่อมต่อสำหรับการทดลองที่ 4

การทดลอง

11.4.1 การทดลองที่ 4.1 ทดสอบการใช้งานโพรโตคอล HTTP ผ่านเว็บบราวเซอร์

ทดลองใช้บราวเซอร์เรียกใช้บริการ HTTP เพื่อแสดงเว็บไซต์ที่เป็น IPv6 และ IPv4

สมมติฐาน เมื่อสามารถใช้งาน IPv6 ได้แล้วก็จะสามารถเข้าใช้บริการ World Wide Web ได้ทั้งเซิฟเวอร์ที่สนับสนุน IPv4 และ IPv6

วิธีทดลอง

1. ติดตั้ง IPv6 dual-stack ให้กับเครื่องที่ใช้งาน Window XP และคอนฟิกให้เชื่อมต่อกับ 6Bone ตามการทดลองที่ผ่านมาโดยพิมพ์คำสั่งดังนี้

   ipv6 install

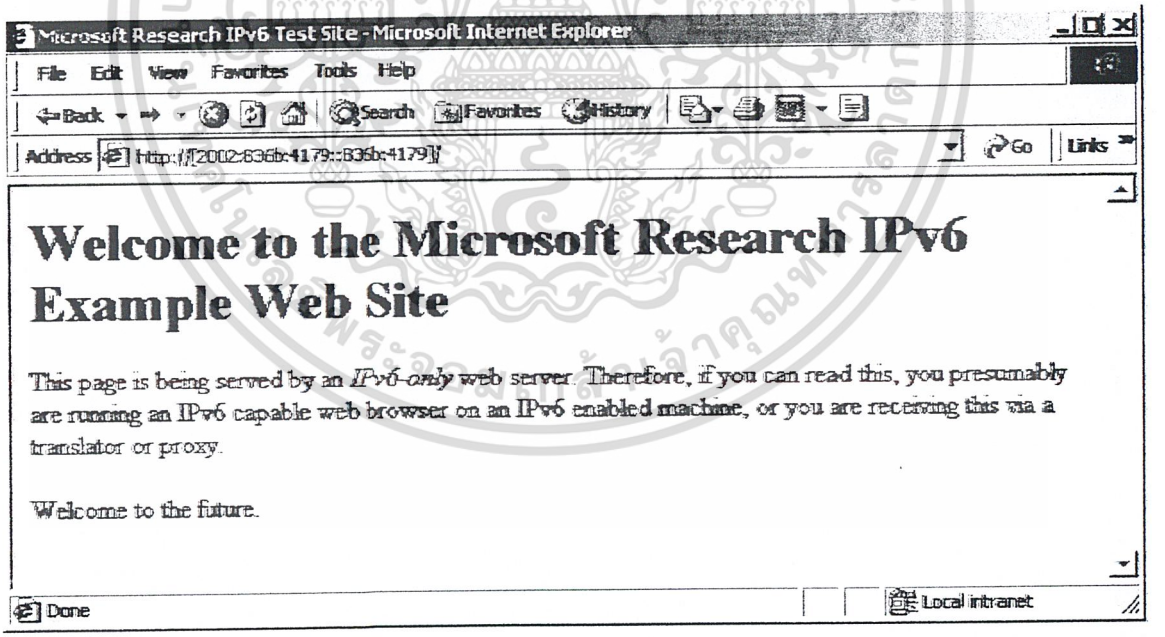
   ipv6 rtu ::/0 2/::64.71.128.26 pub

   ipv6 adu 2/3FFE:1200:3028:FF01::46D9

2. เรียกดูเว็บไซต์จากโฮสต์ใน 6Bone ผ่านบราวเซอร์เพื่อตรวจสอบว่าสามารถเรียกใช้โพรโตคอล http ผ่าน IPv6 ได้ โดยบราวเซอร์ที่ใช้เป็น Internet Explorer เวอร์ชั่น 6 ที่ช่องแอ็ดเดรสใส่แอ็ดเดรส ของ IPv6 ไซต์

   http://[2002:836B:4179::836B:4179]

   จะเข้าสู่เว็บไซต์ของ Microsoft ซึ่งสามารถเข้าถึงได้โดยโฮสต์ที่สนับสนุน IPv6 เท่านั้น

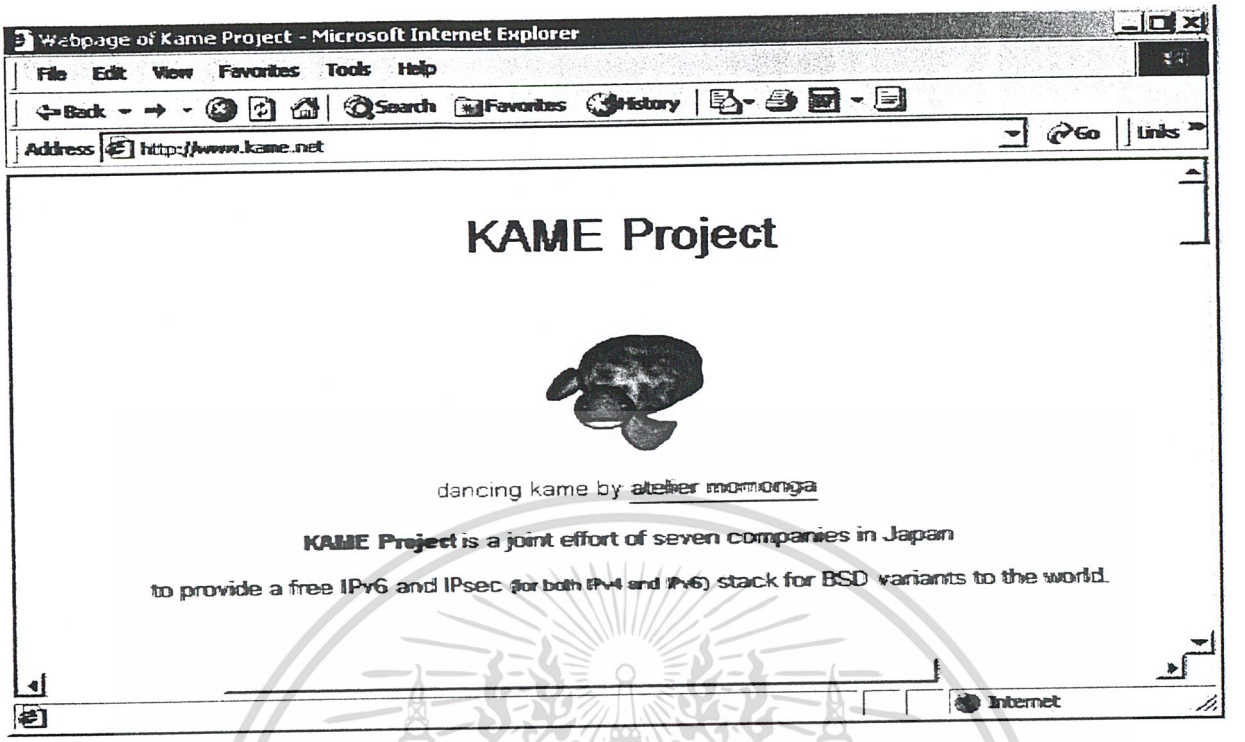


รูปที่ 11.4.2 แสดงผลการ ติดต่อกับไซต์ใน 6Bone ด้วยเว็บบราวเซอร์ โดยโพรโตคอล http

3. ทดลองเปิดเว็บโดยการใช้ domain name โดยพิมพ์

   www.kame.net

   จะได้ผลลัพธ์ดังภาพ

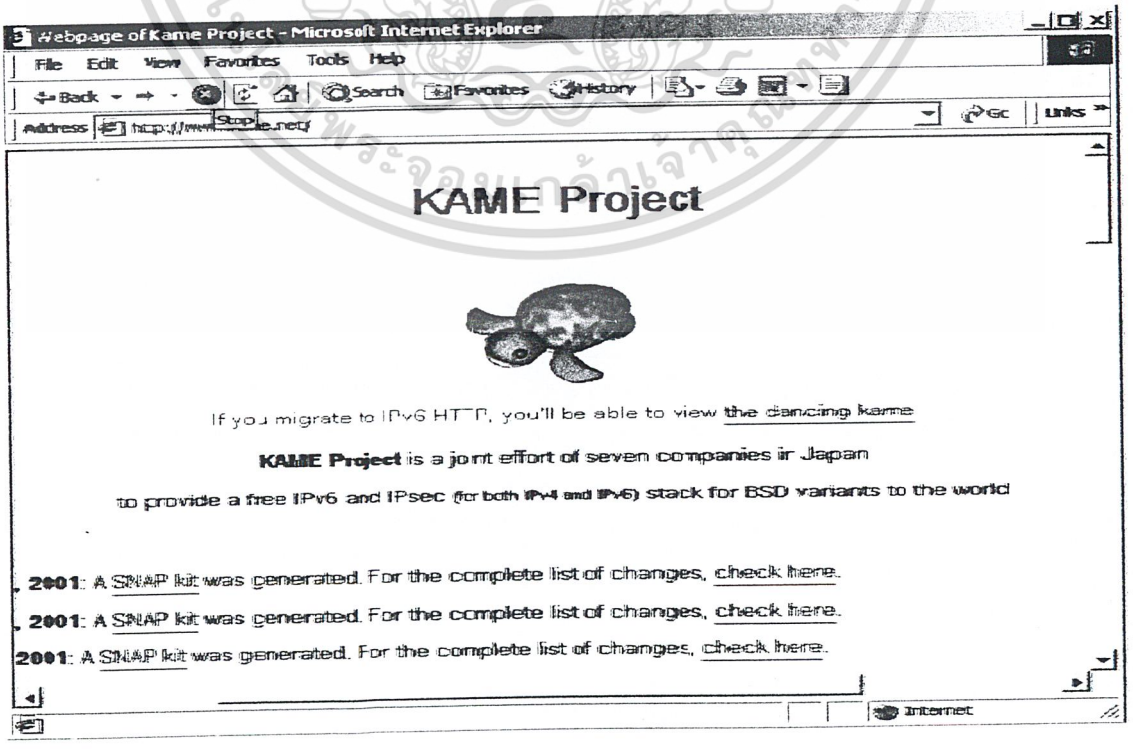


รูปที่ 11.4.3 การเปิดเว็บใน IPv6 ไซต์โดยการใช้ domain name

เว็บนี้เปิดให้บริการทดสอบว่าโฮสต์สนับสนุน IPv6 หรือไม่โดยหากสนับสนุนแล้ว ภาพของ เต่าจะสามารถเคลื่อนไหวได้ ลองทดสอบโดยการยกเลิกการใช้งาน IPv6 โดยการพิมพ์คำสั่ง

ipv6 uninstall

จากนั้นทดลองเข้าเว็บใหม่อีกครั้งผลปรากฏดังภาพ

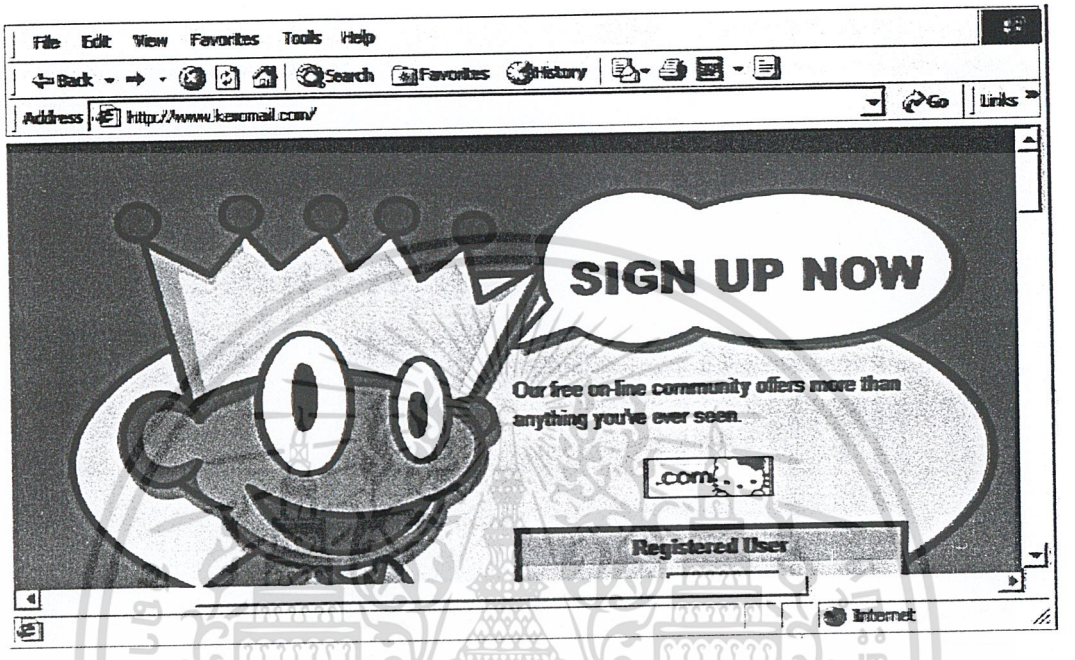


รูปที่ 11.4.4 การติดต่อไซต์เดิมหลังจากยกเลิกการใช้งาน IPv6

เต่าในภาพไม่เคลื่อนไหวแสดงว่าโฮสต์ไม่สนับสนุน IPv6 แล้ว

4. ทดลองเปิดการใช้งาน IPv6 อีกครั้งและทดลองเปิดเข้าเว็บไซต์ที่สนับสนุนเฉพาะ IPv4 อย่างเดียวเช่น

www.keromail.com



รูปที่ 11.4.5 ผลการเรียกเว็บไซต์ที่สนับสนุนเฉพาะ IPv4 ในขณะที่กำลังใช้งาน IPv6

สามารถรียกใช้งานเว็บทั่วไปได้ตามปกติแสดงว่าการใช้งาน Dual-stack ไม่ทำให้เกิดการปัญหาระหว่าง IPv4 และ IPv6

ข้อสรุป

เมื่อติดตั้งให้ใช้งาน IPv6 ได้ก็จะสามารถเข้าใช้บริการ World Wide Web ได้ แต่มีข้อแม้ว่าบราวเซอร์ที่ใช้นั้นจะต้องสนับสนุน IPv6 ด้วย ถ้าไม่สนับสนุนก็จะไม่สามารถติดต่อกับเซิฟเวอร์ที่สนับสนุน IPv6 ได้ แต่จะยังสามารถติดต่อกับเซิฟเวอร์ที่สนับสนุน IPv4 ได้เหมือนเดิม

11.4.2 การทดลองที่ 4.2 ทดสอบการใช้งาน FTP

ทดสอบการใช้งาน FTP ติดต่อระหว่างโฮสต์ IPv6 และ IPv4 ในสภาพแวดล้อมทั้ง 2 แบบ

สมมติฐาน สามารถใช้บริการ FTP ได้ทั้งเซิฟเวอร์ที่สันบสนุน IPv4 และ IPv6

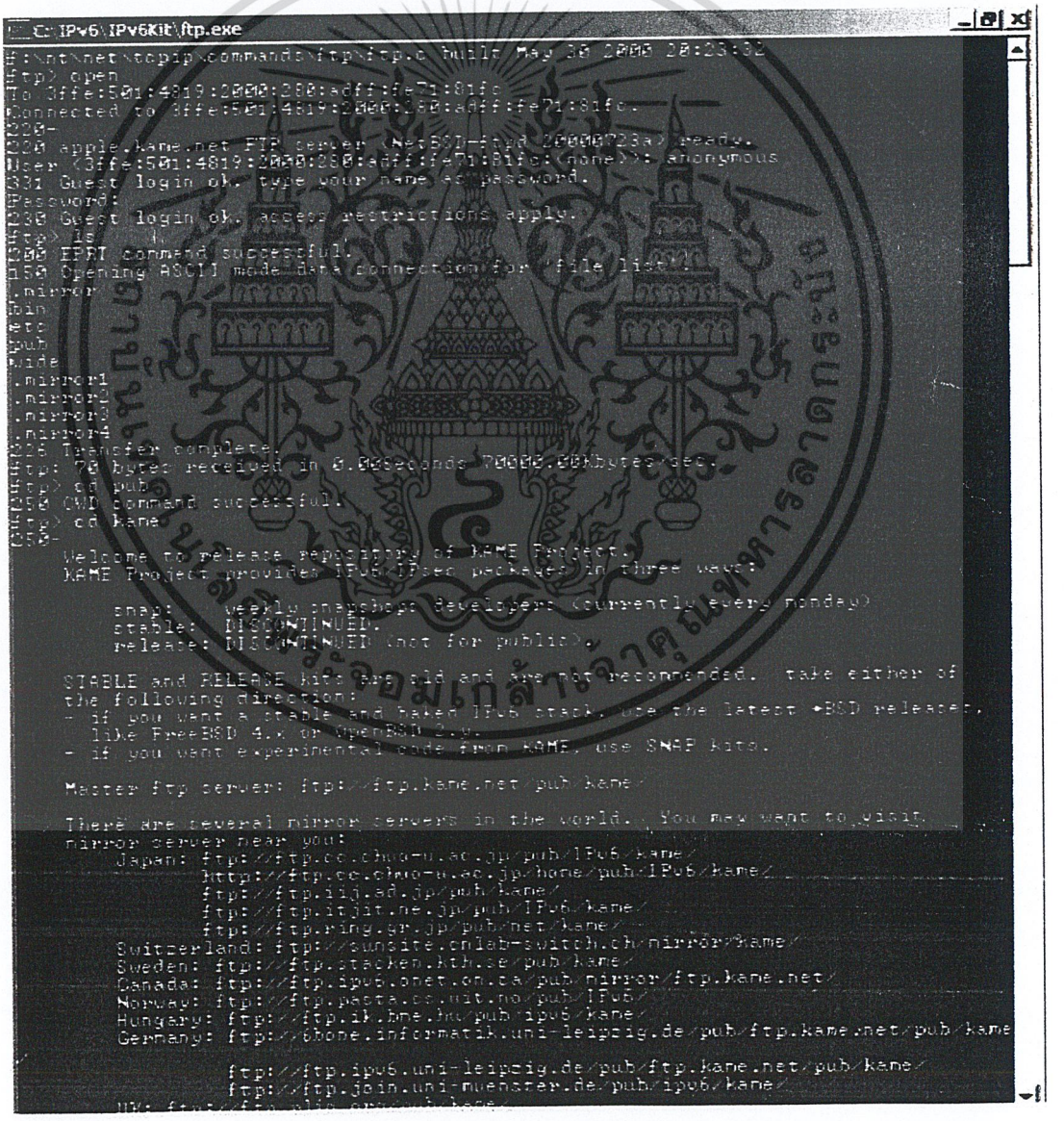
วิธีทดลอง

1. ติดตั้ง IPv6 dual-stack ให้กับเครื่องที่ใช้งาน Window 2000 และคอนฟิกให้เชื่อมต่อกับ 6Bone ตามการทดลองที่ผ่านมาโดยพิมพ์คำสั่งดังนี้

   ipv6 rtu ::/0 2/::64.71.128.26 pub

   ipv6 adu 2/3FFE:1200:3028:FF01::46D9

2. ในชุดคูอัล-แสตกของ Microsoft นั้นจะมีโปรแกรม FTP ที่สนับสนุนการใช้งาน IPv6 แล้ว การใช้งานทดลองติดต่อไปยังโฮสต์ใน 6Bone ที่เปิดให้บริการ FTP ได้ผลดังรูป

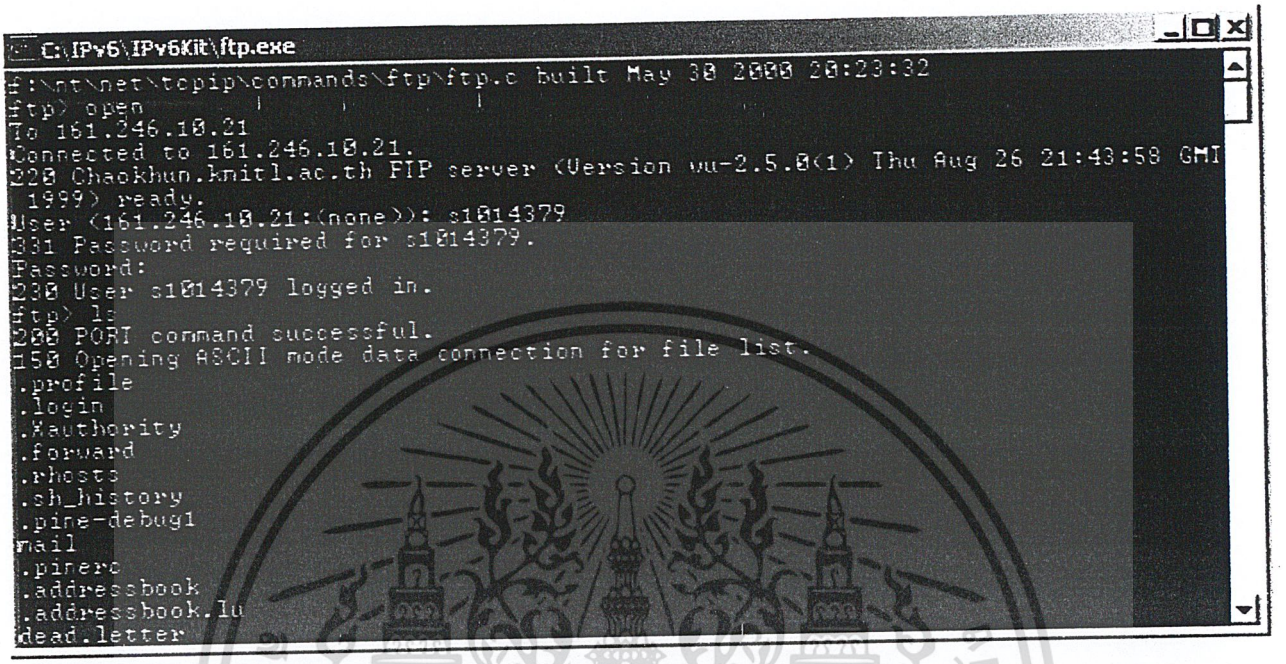


รูปที่ 11.4.6 ผลการเรียกใช้บริการ FTP ติดต่อกับเครื่องภายใน

โปรแกรมรู้จักแอ็ดเดรสแบบ IPv6 สามารถติดต่อและใช้บริการได้ตามปกติ

3. ทดลองติดต่อกับโฮสต์ IPv4 ที่เปิดบริการ FTP เพื่อทดสอบว่าสามารถใช้งานร่วมกันได้หรือไม่

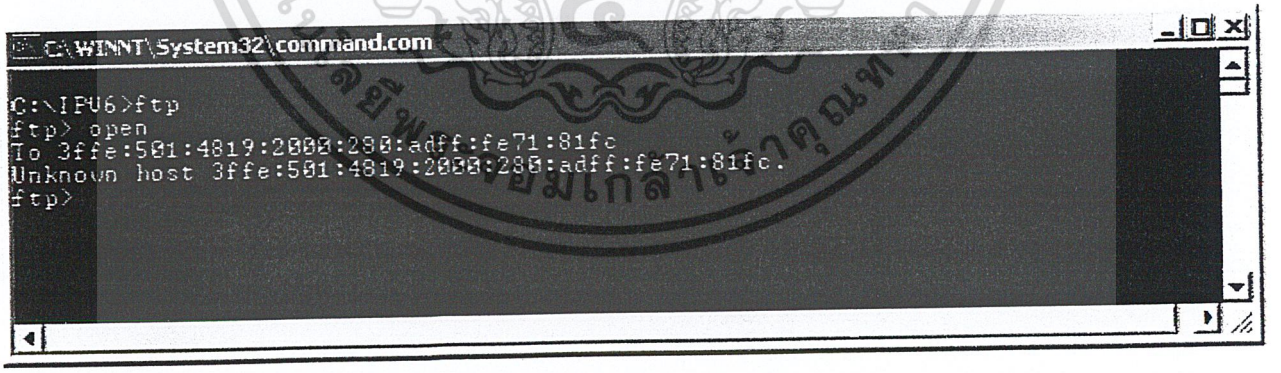
ได้ผลดังภาพ



รูปที่ 11.4.7 แสดงการใช้ FTP ที่สนับสนุน IPv6 ติดต่อกับ IPv4

สามารถเข้าติดต่อใช้งานถ่ายโอนแฟ้มข้อมูลได้ตามปกติ

4. ทดลองใช้ FTP ที่ให้มากับระบบปฏิบัติการติดต่อกับ IPv6 ไซต์



รูปที่ 11.4.8 แสดงการใช้ FTP ที่มากับระบบปฏิบัติการที่ไม่สนับสนุน IPv6

ผลปรากฏว่าโปรแกรมไม่รู้จักแอ็ดเดรสแบบ IPv6 แม้ระบบจะสนับสนุน IPv6 อยู่แล้ว

ข้อสรุป

หลังจากการติดตั้งให้ใช้งาน IPv6 ได้แล้วก็จะสามารถใช้บริการ FTP เพื่อติดต่อกับเซิฟเวอร์ที่สนับสนุน IPv4 หรือ IPv6 แต่มีข้อแม้ว่า FTP ไคลเอนต์จะต้องถูกออกแบบให้สนับสนุน IPv6 ด้วย ถ้าไม่สนับสนุนก็จะไม่สนับสนุนใช้บริการ FTP ติดต่อกับเซิฟเวอร์ IPv6 ได้

บทที่ 12
บทสรุปการทดลอง

โครงงานเครือข่ายทดลองไอพีเวอร์ชั่น 6 (IPv6 Testbed) มีจุดประสงค์เพื่อรวบรวมองค์ความรู้เกี่ยวกับอินเทอร์เน็ตโพรโตคอลเวอร์ชั่น 6 และทำการติดตั้งและทดสอบการทำงานในสภาพแวดล้อมต่างๆ เมื่อได้ศึกษาถึงการทำงานของ IPv6 อย่างละเอียดแล้ว ก็ได้ทำการทดลองตามที่ได้ออกแบบไว้เพื่อทดสอบว่าทำงานได้ตรงตามทฤษฎีและจะเกิดปัญหาใดๆขึ้นในการทำงานจริงหรือไม่ โดยได้ทำแบ่งการทดลองออกเป็น 3 รูปแบบตามลักษณะการใช้งานที่จะเกิดขึ้นจริง ดังนี้

1. การเชื่อมต่อกับโฮสต์ภายในเครือข่ายเดียวกัน

การติดต่อภายในเครือข่ายเดียวกันนั้น การส่งแพ็กเก็ตถึงกันจะสามารถส่งแพ็กเก็ตแบบ IPv6 ได้โดยตรงโดยผ่านโพรโตคอลที่ใช้ภายในเครือข่ายเช่นอีเทอร์เน็ต การทดลองการเชื่อมต่อกันกับโฮสต์ที่ทำงานภายใต้ระบบปฏิบัติการที่ต่างกันสามารถเชื่อมต่อกันได้โดยไม่เกิดปัญหาใดๆ โฮสต์ที่ติดตั้งแสตกของ IPv6 สามารถทำงานอยู่ภายในเครือข่ายเดียวกันกับโฮสต์แบบ IPv4 ได้โดยไม่เกิดการขัดแย้งกัน การติดต่อกันใช้แอ็ดเดรสแบบลิ้งค์-โลคัลแอ็ดเดรสซึ่งเป็นแอ็ดเดรสที่เกิดจากแอ็ดเดรส IPv4 หรือหมายเลขอินเทอร์เฟซ

2. การเชื่อมต่อกันระหว่างเครือข่าย

การเชื่อมต่อกันระหว่างเครือข่าย IPv6 ใดๆสองเครือข่ายโดยผ่านโครงสร้างเครือข่ายที่สนับสนุนเฉพาะ IPv4 นั้น ไม่สามารถทำได้โดยการส่งแพ็กเก็ต IPv6 ถึงกันโดยตรง เนื่องจากเราเตอร์จะไม่รู้จักแพ็กเก็ตของ IPv6 และจะทำการทิ้งแพ็กเก็ตนั้นไป จึงต้องมีการผ่านกระบวนการทำทันเนิล (IPv6 Tunneling) คือการนำแพ็กเก็ตแบบ IPv6 มาครอบด้วยแพ็กเก็ตแบบ IPv4 เสมือนว่าแพ็กเก็ต IPv6 เป็นส่วนข้อมูล (Data) ของ IPv4 โดยกำหนดแอ็ดเดรสปลายทางของส่วนหัวของ IPv4 เป็นแอ็ดเดรส IPv4 ของเครื่องปลายทางหรือเครื่องเกตเวย์ของเครือข่ายปลายทาง เพื่อให้สามารถส่งผ่านเครือข่ายของ IPv4 ได้ โดยวิธีการปรับเปลี่ยนนั้นสามารถทำได้สองวิธี

วิธีแรกทำได้โดยการคอนฟิกให้ทุกโฮสต์ภายในเครือข่ายสร้างอุโมงค์ไปยังโฮสต์ทุกๆโฮสต์ภายในเครือข่ายปลายทาง ข้อดีของวิธีการนี้คือ สามารถคอนฟิกโฮสต์แต่ละตัวแยกกันได้อย่างอิสระ แต่ข้อเสียคือจะทำให้เกิดอุโมงค์ขึ้นมากมายและการเชื่อมต่อจะเป็นแบบ Mesh และการเพิ่มโฮสต์ใหม่เข้าไปในเครือข่ายจะทำได้ลำบาก เนื่องจากต้องเพิ่มคอนฟิกให้โฮสต์ทุกโฮสต์ วิธีที่สองคือการกำหนดให้แต่ละเครือข่ายมีโฮสต์ที่ทำหน้าที่เป็นเกตเวย์เพื่อเป็นทางผ่านของแพ็กเก็ตที่เข้าและออกจากเครือข่าย เกตเวย์จะทำหน้าที่เอ็นแค็ปแพ็กเก็ตของเครื่องทุกเครื่องภายในเครือข่ายส่งไปยังเครื่องเกตเวย์ปลายทาง ทำให้อุโมงค์ที่เกิดขึ้นจะมีเพียงอุโมงค์เดียว คืออุโมงค์ระหว่างเครื่องเกตเวย์ทั้งสอง โฮสต์ทุกโฮสต์ในเครือข่ายแต่ละเครือข่ายจะถูกคอนฟิกให้ส่งแพ็กเก็ตที่ต้องการส่งออกนอกเครือข่ายไปยังเครื่องเกตเวย์

ข้อดีของวิธีการนี้มีหลายประการเช่น การคอนฟิกสามารถทำได้ไม่ยุ่งยาก เพียงแต่คอนฟิกอุโมงค์ที่เครื่องเกตเวย์ก็เพียงพอ การเพิ่มโฮสต์เข้าไปในเครือข่ายก็เพียงแต่คอนฟิกให้ส่งแพ็กเก็ตไปยังโฮสต์บริการเท่านั้น รวมทั้งยังมีข้อดีอีกประการคือ ต้องการแอ็ดเดรส IPv4 เฉพาะเครื่องที่ทำหน้าที่เป็นเกตเวย์เท่านั้นก็เพียงพอ

การส่งแพ็กเก็ตข้ามเครือข่ายอาจเกิดปัญหากรณีส่งแพ็กเก็ตที่มีขนาดใหญ่กว่าขนาด MTU ของลิ้งค์ ในการทดลองได้มีการทดสอบโดยการกำหนดขนาด MTU ให้มีขนาดเล็ก และทำการส่งแพ็กเก็ตที่มีขนาดใหญ่ไป ผลปรากฏว่าแพ็กเก็ตถูกแบ่งย่อยตั้งแต่ที่โฮสต์ต้นทางเพื่อให้ส่งผ่านลิ้งค์นั้นได้และถูกรวมที่โฮสต์ปลายทางอย่างถูกต้องตรงตามข้อกำหนดของ IPv6 เพื่อช่วยลดภาระการทำงานของเราเตอร์

นอกจากนี้ยังได้ทำการทดลองด้านความปลอดภัยของ IPv6 ด้วย โดย IPv6 ได้รวมเอา IPSec เข้ามาอยู่ในข้อกำหนดของโพรโตคอลเลย แบ่งเป็นการเข้ารหัสและการพิสูจน์ตัวจริง ในการทดลองได้ทำ IPSec ระหว่างโฮสต์ที่เป็นเกตเวย์ของแต่ละเครือข่าย ทำให้แพ็กเก็ตทั้งหมดที่ส่งถึงกันระหว่างสองเครือข่ายถูกครอบคลุมด้วยความสามารถของ IPSec ทั้งหมด การคอนฟิกในการทดลองนี้สามารถนำไปใช้จริงได้ เช่นการติดต่อกันระหว่างองค์กรสององค์กรใดๆเมื่อต้องการทำเป็นความลับ สามารถกำหนด IPSec ระหว่างเกตเวย์ของแต่ละองค์กร โดยเครื่องอื่นๆภายในองค์กรไม่ต้องคอนฟิกอะไรเพิ่มเติมเลย

ก่อนการทดลองมีคำถามเกิดขึ้นว่าหลังจากติดตั้งแสตกของ IPv6 แล้ว ยังสามารถทำงานร่วมกับโฮสต์ที่เป็น IPv4 เท่านั้นได้หรือไม่ เนื่องจากเราไม่สามารถปรับเปลี่ยนโฮสต์ทุกโฮสต์บนโลกนี้ให้เป็น IPv6 ได้พร้อมกัน การทำงานร่วมกับโฮสต์แบบ IPv4 จึงยังเป็นเรื่องสำคัญ ซึ่งปัญหาการส่งผ่านแพ็กเก็ตไปยังเครื่อข่าย IPv4 ก็สามารถทำได้โดยผ่านการทำทันเนิลแล้ว IPv6 ถูกกำหนดให้เป็นโพรโตคอลหนึ่งในระดับชั้น IP ของโพรโตคอล TCP/IP เมื่อต้องการติดต่อกับโฮสต์ที่เป็น IPv 4 ก็ใช้แอ็ดเดรสแบบ IPv4 และใช้โพรโคคอล IPv4 แต่หากต้องการติดต่อกับโฮสต์ IPv6 ก็จะใช้แอ็ดเดรส IPv6 และโพรโตคอล IPv6 ทำให้ไม่เกิดการขัดแย้งกันและสามารถทำงานร่วมกันได้ ซึ่งผลการทดลองก็สนับสนุนความจริงข้อนี้ได้ โดยการติดต่อกับโฮสต์ทั้งภายในและนอกเครือข่ายสามารถทำได้เหมือนปกติ

3. การเชื่อมต่อกับ 6Bone

6Bone คือกลุ่มของเครือข่าย IPv6 ที่เป็นเหมือนศูนย์กลางและทำหน้าที่ส่งผ่านแพ็กเก็ตไปยังเครือข่ายปลายทาง หากไม่มี 6Bone การเชื่อมต่อกันระหว่างเครือข่ายก็จะต้องทำอุโมงค์เองถึงกันโดยตรง หากมีเครือข่ายเกิดขึ้นใหม่ก็จะต้องทำการคอนฟิกเพิ่มและปรับเปลี่ยนตลอดเวลา แต่หากคอนฟิกเครือข่ายเชื่อมกับ 6Bone ก็จะสามารถติดต่อถึงกันได้ตลอดโดยไม่ต้องทำการปรับเปลี่ยนใดๆ การเชื่อมต่อเข้ากับ 6Bone จะต้องผ่านทางรีเลย์เราเตอร์ซึ่งเป็นเสมือนกับปากทางเข้าสู่ 6Bone โดยการทำทันเนิลจากโฮสต์ของเราไปยังรีเลย์เราเตอร์หรือหากเป็นเครือข่ายก็ทันเนิลจากโฮสต์เกตเวย์ไปยังรีเลย์เราเตอร์

จะทำให้เครือข่ายของเราทั้งเครือข่ายเชื่อมต่อกับ 6Bone และสามารถติดต่อกับเครือข่ายอื่นๆทั่วโลกที่เชื่อมต่อกับ 6Bone ได้

4. การใช้งาน IPv6 ในระดับชั้นโปรแกรมประยุกต์

จากการทดลองสรุปได้ว่าโปรแกรมประยุกต์นั้นๆจะต้องถูกออกแบบมาให้รู้จัก IPv6 จึงจะสามารถใช้งานได้ ในการทดลองได้ทดสอบโดยการเรียกดูเว็บเพจผ่านบราวเซอร์ Internet Explorer เวอร์ชั่น 5.5 ซึ่งรู้จัก IPv6 แล้ว ผลการทดสอบสามารถเรียกดูเว็บเพจจากโฮสต์ใน 6Bone ที่สนับสนุนเฉพาะ IPv6 ได้ โดยเมื่อสอบถามแอ็ดเดรสไปยัง DNS หากได้รับเรคคอร์ดแบบ A6 หรือ AAAA กลับมาก็จะส่งการร้องขอโดยใช้แอ็ดเดรส IPv6 ไปยังเครื่องเซิร์ฟเวอร์ ในการทดลองยังได้ทดสอบโปรแกรม FTP ด้วย ผลปรากฏว่าก็สามารถถ่ายโอนไฟล์ได้ตามปกติ จึงสรุปได้ว่าการใช้งาน IPv6 ในระดับโปรแกรมประยุกต์ทั้งกับโฮสต์ที่เป็น IPv6 และ IPv4 สามารถทำได้โดยไม่เกิดปัญหาใดๆ

หลังจากที่ทำการทดลองแล้ว จึงสรุปได้ว่า อิมพลีเมนต์ของ IPv6 แสตกของหลายๆระบบปฏิบัติการในเวลานี้ ยังไม่สามารถทำตามข้อกำหนดได้ทุกอย่าง การติดตั้ง IPv6 แสตกสามารถทำได้ไม่ยุ่งยาก การใช้งานลักษณะต่างๆรวมทั้งการทำงานร่วมกับ IPv4 สามารถทำได้โดยไม่เกิดปัญหา แต่การปรับแต่งระบบให้เป็นไปตามที่ต้องการจะมีความยากอยู่บ้าง เนื่องจากอิมพลีเมนต์ที่ต่างระบบปฏิบัติการและผู้ปรับแต่งจำเป็นต้องมีความรู้ทางทฤษฎีเกี่ยวกับข้อกำหนดต่างๆของ IPv6 อยู่พอสมควร ประโยชน์ของโครงงานนี้ช่วยให้ผู้ใช้ทั่วไป ผู้ที่สนใจ ตลอดจนผู้ดูแลระบบสามารถติดตั้งและปรับแต่งระบบให้เป็นไปตามที่ต้องการได้

ภาคผนวก ก

การใช้งาน IPv6 ในระบบปฏิบัติการ Windows 2000

ระบบที่ต้องการ

โฮสต์ติดตั้งระบบปฏิบัติการ Windows 2000 Professional , Windows 2000 Server หรือ Windows 2000 Advance Server พร้อมกับติดตั้ง Service Pack 1 รองรับอีเทอร์เน็ตอแด็ปเตอร์ทุกชนิดที่ Windows 2000 รองรับ และต้องติดตั้ง Internet Protocol (TCP/IP)

การติดตั้ง

1. สามารถดาวโหลดคูอัล-แสตกได้จาก
   http://msdn.microsoft.com/downloads/sdks/platform/tpipv6/readme.asp
2. ล็อกออนด้วยแอคเคาตน์ระดับผู้ดูแลระบบ (Adminstrator)
3. รันไฟล์ setup.exe จากไดเร็กตอรี่ที่แตกจากไฟล์ที่ดาวน์โหลดได้
4. คลิกที่ Start >> Settings >>Network and Dial-up Connections
5. เลือกการเชื่อมต่อที่ต้องการเพิ่มโพรโตคอล IPv6 เข้าไป เช่น Local Area Connection
6. คลิก Install
7. เมื่อปรากฏไดอะล็อก Select Network Component Type คลิก Protocol และคลิก Add
8. เมื่อปรากฏไดอะล็อก Select Network Protocol คลิก Microsoft IPv6 Protocol และคลิก OK
9. คลิก Close

การติดตั้งจะทำการก็อปปี้ไฟล์ tcpip6.sys ลงในไดเร็กตอรี่ %systemroot%\system32\drivers และไฟล์ wship6.dll กับไฟล์เครื่องมือต่างๆ (ipv6.exe , ping6.exe , tracert6.exe , ttcp.exe) ลงไปในไดเร็กตอรี่ %systemroot%\system32 และจะทำการแทนที่ไฟล์เหล่านี้กับเวอร์ชั่นเดิมที่อยู่ในระบบแล้ว

- Wininet.dll
- Ftp.exe
- Telnet.exe
- Tlntsvr.exe

การตั้งค่าคอนฟิกต่างๆนั้นจะสูญหายไปเมื่อทำการรีเซ็ตเครื่องหรือทำการรีสตาร์ท IPv6 โปรโตคอล ผู้ใช้งานสามารถเก็บค่าคอนฟิกเหล่านั้นไว้ในรูปชุดคำสั่ง (Command line) ในสคริปต์ไฟล์นามสกุล .cmd เพื่อเรียกใช้งานหลังจากเปิดเครื่องใหม่หรือรีสตาร์ท IPv6 โปรโตคอล การกำหนดให้เรียกใช้งานคอนฟิกไฟล์โดยอัตโนมัติหลังจากเปิดเครื่องนั้น ใช้ Window 2000 Scheduled Tasks กำหนดให้เรียกไฟล์ .cmd เมื่อคอมพิวเตอร์เริ่มทำงาน

Net.exe

ใช้เริ่มหรือสิ้นสุดการทำงานของโปรโตคอล IPv6 การรีสตาร์ทโปรโตคอล IPv6 มีผลเหมือนกับการรีเซ็ตเครื่องซึ่งอาจจะมีการเปลี่ยนหมายเลขอินเทอร์เฟซ

Net.exe มีคำสั่งย่อยหลายคำสั่ง แต่ละอันมีรูปแบบอาร์กูเมนต์ (Argument)และออปชั่นเป็นของตัวเอง ดังตัวอย่างต่อไปนี้

net stop tcpip6

หยุดการทำงานของ IPv6 และลบทิ้งจากหน่วยความจำคำสั่งนี้จะล้มเหลวเมื่อมี IPv6 ซ็อคเก็ตใดๆเปิดอยู่แล้ว

net start tcpip6

เริ่มการทำงานของ IPv6 ถ้ายังไม่มีการใช้งาน ไฟล์ Tcpip6.sys ในไดเร็กตอรี่ %systemroot%\System32\Drivers จะถูกเรียกใช้งาน

Ipv6.exe

การตั้งค่าคอนฟิกต่างๆสามารถทำได้โดยผ่าน IPv6.exe ใช้สำหรับการสอบถามหรือตั้งค่าของอินเทอร์เฟซ แอ็ดเดรส แค็ช และ เราต์ Ipv6.exe มีหลายคำสั่งย่อยดังนี้

ipv6 if [if#]

แสดงข้อมูลเกี่ยวกับอินเทอร์เฟซ ถ้ามีการระบุหมายเลขอินเทอร์เฟซข้อมูลเกี่ยวกับอินเทอร์เฟซนั้นจะถูกแสดงเท่านั้น ถ้าไม่ระบุจะแสดงข้อมูลของอินเทอร์เฟซทั้งหมด ข้อมูลที่แสดงคือ ลิ้งค์-เลเยอร์ แอ็ดเดรส และแอ็ดเดรส IPv6 ทั้งหมดที่ถูกกำหนดให้อินเทอร์เฟซนั้น รวมไปถึงขนาด MTU ปัจจุบันและขนาดของ MTU สูงสุดที่อินเทอร์เฟซนั้นสนับสนุน

อินเทอร์เฟซ #1 เป็นอินเทอร์เฟซเทียมใช้สำหรับการทำลูปแบ็ค อินเทอร์เฟซ #2 เป็นอินเทอร์เฟซเทียมใช้สำหรับการทำคอนฟิกทันเนิล ออโตเมติกทันเนิล และ 6to4 ทันเนิล หมายเลขอินเทอร์เฟซอื่นๆจะแสดงเรียงลำดับตามการถูกสร้างก่อน-หลัง

ถ้าลิ้ง-เลเยอร์แอ็ดเดรสเป็นรูปแบบ aa-bb-cc-dd-ee-ff แสดงว่าเป็นอีเทอร์เน็ตแอ็ดเดรส ถ้าอยู่ในรูปแบบ a.b.c.d เป็น 6-over-4 อินเทอร์เฟซ ข้อมูลเกี่ยวกับ 6-over-4 หาดูได้ที่ RFC 2529 อินเทอร์เฟซเทียมทั้งสองนั้นจะไม่ใช้ Neighbor Discovery

ipv6 ifc if# [forward] [advertises] [-forward] [-advertises] [mtu #bytes] [site site-identifier]

ใช้สำหรับควบคุมแอททริบิวต์ของอินเทอร์เฟซ อินเทอร์เฟซสามารถทำการส่งต่อ (forward)ในกรณีที่แอ็ดเดรสปลายทางไม่ได้ถูกกำหนดให้กับอินเทอร์เฟซนั้น หรืออินเทอร์เฟซอาจจะทำการแอ็ดเวอไทซ์ (Advertises) ในกรณีที่ทำการส่ง router advertisement แอททริบิวต์เหล่านี้สามารถควบคุมได้อย่างอิสระ การที่อินเทอร์เฟซเทียมทั้งสองไม่ใช้ Neightbor Discovery ทำให้ไม่สามารถกำหนดค่าคอนฟิกของ router advertisement ได้ ผู้ใช้สามารถกำหนด MTU สำหรับอินเทอร์เฟซได้ โดยจะ

ต้องมีค่าน้อยกว่าหรือเท่ากับค่า MTU สูงสุดที่แสดงโดย ipv6 if และจะต้องมากกว่าหรือเท่ากับ ขนาด MTU ต่ำสุดที่ IPv6 กำหนดไว้ (1280 ไบต์)

ผู้ใช้สามารถทำการเปลี่ยน site-identifier สำหรับอินเทอร์เฟซได้ site-identifer จะถูกใช้ในส่วน sin6_scope_id ของไซต์-โลคัล แอ็ดเดรส

ipv6 ifd if#

ลบอินเทอร์เฟซที่ระบุ อินเทอร์เฟซเทียมทั้งสองไม่สามารถลบได้

ipv6 nc [if# [address]]

แสดงข้อมูลใน neighbor cache ถ้ามีการกำหนดหมายเลขอินเทอร์เฟซ จะแสดงเฉพาะ neighbor cache ของอินเทอร์เฟซนั้น สามารถระบุ IPv6 แอ็ดเดรสเพื่อแสดงเฉพาะ neighbor cache เพียงอันเดียว ข้อมูลที่แสดงนั้น อินเทอร์เฟซ ลิงค์-เลเยอร์ แอ็ดเดรส และ สถานะการเข้าถึง จะถูกแสดง

ipv6 ncf [if# [address]]

ใช้ฟลัช (Flush) neighbor cache ที่ระบุ เฉพาะ neighbor cache ที่ไม่มีการอ้างถึงจะถูกขจัดทิ้ง เพราะว่าส่วน route cache จะทำการอ้างถึง neighbor cache จึงแนะนำว่าควรทำการฟลัช route cache ก่อน ตารางเราติงสามารถอ้างถึง neighbor cache ด้วย

ipv6 rc [if# [address]]

แสดงข้อมูลใน route cache ชื่อ route cache ถูกกำหนดโดย Microsoft IPv6 implementation เป็นชื่อเรียกสำหรับ destination cache ถ้ามีการระบุหมายเลขอินเทอร์เฟซและแอ็ดเดรส จะแสดงเส้นทางการเข้าถึงแอ็ดเดรส หากไม่ระบุจะเป็นการแสดงข้อมูลใน route cache ทั้งหมด

สำหรับแต่ละเรคคอร์ดของ route cache จะแสดงข้อมูล IPv6 แอดเดรส อินเทอร์เฟซที่อยู่ถัดไป และแอ็ดเดรสที่อยู่ติดกัน รวมไปถึงแอ็ดเดรสต้นทางที่จะถูกใช้สำหรับปลายทางนี้ และขนาด MTU ที่ใช้เข้าถึงปลายทางด้วย

แอ็ดเดรสปลายทางสามารถมีหลายเส้นทางตามจำนวนอินเทอร์เฟซ

ipv6 rcf [if# [address]]

ใช้ฟลัช route cache

ipv6 bc

ใช้แสดงข้อมูลของ binding cache ที่ใช้เก็บความสัมพันธ์ระหว่าง home address และ care-of-address สำหรับ Mobile IPv6

ipv6 spt

แสดงข้อมูลปัจจุบันของตารางพรีฟิกซ์ โดยปกติแล้วจะถูกกำหนดอัตโนมัติโดย router advertisement

ipv6 spu prefix if# [lifetime L]

เพิ่ม ลบ หรือแก้ไขพรีฟิกซ์ในตารางพรีฟิกซ์ โดยปกติแล้วระยะเวลาของพรีฟิกซ์(เป็นวินาที) จะกำหนดเป็นไม่สิ้นสุด หากมีค่าเป็นศูนย์จะถูกลบทิ้งไป

ipv6 rt

แสดงข้อมูลในตารางเราติง

ipv6 rtu prefix if#[/nexthop] [lifetime L] [preference P] [publish] [age] [sp1 site-prefix-length]

ใช้เพิ่ม ลบ เส้นทางในตารางเราติง ส่วนพรีฟิกซ์สามารถเป็นในลิ้งค์เดียวกันเพื่อกำหนดอินเทอร์เฟซ หรืออาจจะเป็นโหนดถัดไป สามารถกำหนด lifetime เป็นวินาที (ค่าเริ่มต้นเป็นไม่สิ้นสุด) ส่วน preference มีค่าเริ่มต้นเป็นศูนย์

หากมีการกำหนดเป็น publish หมายความว่ามันจะถูกสร้างเป็น router advisement โดยใช้ age สำหรับกำหนดอายุ ส่วน sp1 ใช้กำหนดความยาวของไซต์พรีฟิกซ์ใช้สำรับส่ง router advertisement

Ipsec6.exe

ใช้กำหนด policy และ association ความปลอดภัยสำหรับโปรโตคอล IPv6 มีคำสั่งย่อยดังนี้

ipsec6 sp [interface]

แสดง policy ที่ทำงานอยู่ สามารถระบุอินเทอร์เฟซได้

ipsec sa

แสดง association ที่ทำงานอยู่

ipsec6 c [filename]

ใช้สร้างไฟล์เพื่อคอนฟิก policy และ association กำหนดชื่อโดยค่า filename โดยไม่ต้องใส่นามสกุล นามสกุลของ policy เป็น .spd และ assocciation เป็น .sad

ipsec6 a [filname]

ใช้เพิ่ม policy ลงไปในไฟล์ filename.spd (ไม่ต้องใส่นามสกุล) และเพิ่ม association ลงในไฟล์ filename.sad

ipsec6 i [policy] [filname]

ใช้เพิ่ม policy ลงไปในไฟล์ filename.spd (ไม่ต้องใส่นามสกุล) และเพิ่ม association ลงในไฟล์ filename.sad โดยอ้างอิงตาม policy

ipsec6 d [type = sp sa] [index]

ลบ policy หรือ association ที่ทำงานอยู่ โดยระบุโดยค่า index

การใช้ Internet Explorer ติดต่อกับ IPv6 เว็บไซต์

เวอร์ชั่นใหม่ของไฟล์ Wininet.dll ทำให้เว็บบราวเซอร์สามารถเข้าถึง IPv6 เว็บเซิฟร์เวอร์ได้ Internet Explorer จะใช้ IPv6 เพื่อดาวน์โหลดเว็บเพจในสถานการณ์ต่อไปนี้

- เมื่อดีเอ็นเอสส่งค่ากลับมาเป็น IPv6 แอ็ดเดรส

- URL ใช้รูปแบบ IPv6 ตามที่กำหนดใน RFC 2732 ตัวอย่างเช่น http://[2010:836B:4179::836B:4179]

สำหรับรายชื่อเว็บไซต์ที่บริการเฉพาะ IPv6 สามารถดูได้ที่ http://www.ipv6.org/other-sites.html ใช้ ping6 เพื่อหาค่า IPv6 แอ็ดเดรส แล้วนำค่าแอดเดรสที่ได้มาใส่ในเครื่องหมาย '[ ]' การเข้าถึงเว็บไซต์ที่บริการเฉพาะ IPv6 นั้นจะติดต่อผ่านทาง 6Bone

Internet Explorer จะไม่สามารถติดต่อกับ IPv6 เว็บไซต์ได้ ถ้ามีการกำหนดให้ใช้พร็อกซี่เซิร์ฟเวอร์ที่ไม่สนับสนุน IPv6

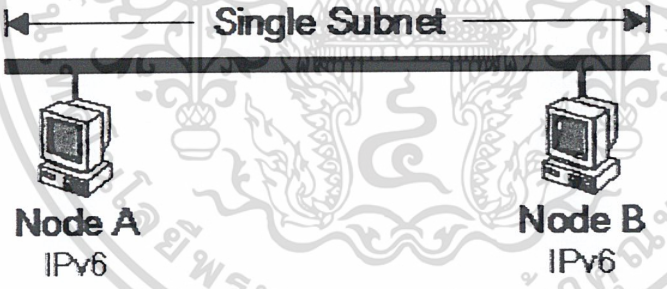
การกำหนดให้ Internet Explorer ยกเลิกการใช้พร็อกซี่เซิร์ฟเวอร์

1. คลิกเมนู Tools แล้วคลิก Internet Options
2. คลิกแถบ Connections แล้วคลิก LAN Settings
3. ยกเลิกการเลือกช่อง Use a proxy server แล้วคลิก OK
4. คลิก OK ในหน้าต่าง Internet Options

ตัวอย่างการคอนฟิก

แบบที่ 1 ภายในซับเน็ตเดียวกันโดยใช้ลิ้ง-โลคัลแอ็ดแดรส

การคอนฟิกแบบนี้ต้องการเพียงแค่ Microsoft IPv6 โปรโตคอลแสตกเท่านั้น ระบบนี้ประกอบด้วยโหนด 2 โหนด ในซับเน็ตเดียวกัน โดยเชื่อมต่อกันโดยไม่ผ่านเราเตอร์ตามรูปที่ 1



รูปที่ 1 แสดงการเชื่อมต่อ 2 โหนดภายในซับเน็ตเดียวกัน

Microsoft IPv6 กำหนดให้ลิ้ง-โลคัลแอ็ดเดรสตามเน็ตเวิร์คอะแด็ปเตอร์ โดยมีพรีฟิกซ์เป็น fe80::/64 โดย 64 บิตหลังคือ interface identifier นำมาจาก 48 บิต MAC แอ็ดเดรสของเน็ตเวิร์คอะแด็ปเตอร์ ดังนี้

- เลขฐานสิบหก 0xff-fe จะถูกแทรกระหว่างไบต์ที่ 3 และ 4 ของ MAC แอ็ดเดรส
- บิตที่สองเรียงจากบิตที่มีนัยสำคัญต่ำสุดของไบต์แรกของ MAC แอ็ดเดรส จะถูกคอมพลีเมนต์

ตัวอย่างเช่น จาก MAC แอ็ดเดรส 00-60-08-ff-fe-52-f9-d8

- แทรก 0xff-fe ระหว่างไบต์ที่ 3 และ 4 จะได้ 00-60-08-ff-fe-52-f9-d8
- ไบต์แรกของ MAC แอ็ดเดรสคือ 00 เขียนเป็นเลขฐานสองได้ 00000000 บิตที่สองเรียงจากนัยสำคัญต่ำสุดมีค่าเป็น 0 ทำการคอมพลีเมนต์ได้ 1 จะได้เป็น 00000010 เท่ากับ 0x02

ได้ interface identifier เป็น 02-60-08-ff-fe-52-f9-d8 เมื่อนำมารวมกับพรีฟิกซ์จะได้เป็นลิ้งค์-โลคัลแอ็ดเดรสเป็น fe80::260:8ff:fe52:f9d8 โดยสามารถเรียกดูลิ้ง-โลคัลแอ็ดเดรสได้โดยใช้คำสั่ง ipv6 if ได้ผลลัพธ์ตามตัวอย่างต่อไปนี้

```
C:\>ipv6 if
Interface 4 (site 1): Local Area Connection
  uses Neighbor Discovery
  link-level address: 00-10-5a-aa-20-a2
    preferred address fe80::210:5aff:feaa:20a2, infinite/infinite
    multicast address ff02::1, 1 refs, not reportable
    multicast address ff02::1:ffaa:20a2, 1 refs, last reporter
  link MTU 1500 (true link MTU 1500)
  current hop limit 128
  reachable time 43500ms (base 30000ms)
  retransmission interval 1000ms
  DAD transmits 1
Interface 3 (site 1): 6-over-4 Virtual Interface
  uses Neighbor Discovery
  link-level address: 10.0.0.2
    preferred address fe80::a00:2, infinite/infinite
    multicast address ff02::1, 1 refs, not reportable
    multicast address ff02::1:ff00:2, 1 refs, last reporter
  link MTU 1280 (true link MTU 65515)
  current hop limit 128
  reachable time 34000ms (base 30000ms)
  retransmission interval 1000ms
  DAD transmits 1
Interface 2 (site 0): Tunnel Pseudo-Interface
  does not use Neighbor Discovery
  link-level address: 0.0.0.0
    preferred address ::10.0.0.2, infinite/infinite
  link MTU 1280 (true link MTU 65515)
  current hop limit 128
  reachable time 0ms (base 0ms)
  retransmission interval 0ms
  DAD transmits 0
Interface 1 (site 0): Loopback Pseudo-Interface
  does not use Neighbor Discovery
  link-level address:
    preferred address ::1, infinite/infinite
  link MTU 1500 (true link MTU 1500)
  current hop limit 1
  reachable time 0ms (base 0ms)
  retransmission interval 0ms
  DAD transmits 0
```

จากตัวอย่าง Interface 4 เป็นลิ้ง-โลคัลแอ็ดเดรสที่เกิดจากอีเทอร์เน็ตอะแด็ปเตอร์

ทดสอบการเชื่อมต่อระหว่าง 2 ลิ้ง-โลคัลโฮสต์

ทำได้โดยการ ping ตามขั้นตอนดังนี้

1. ติดตั้ง Microsoft IPv6 แสตกสำหรับ Windows 2000 ในโฮสต์ทั้งสอง (A และ B) ที่อยู่ในลิ้งค์เดียวกัน
2. ใช้ ipv6 if ที่โฮสต์ A เพื่อดูค่า ลิ้งค์-โลคัลแอ็ดเดรส สมมติว่าได้ค่า fe80::210:5aff:feaa:20a2

3. ใช้ ipv6 if ที่โฮสต์ B เพื่อดูค่า ลิ้งค์-โลคัลแอ็ดเดรส สมมติว่าได้ค่า fe80::260:97ff:fe02:6ea5
4. จากโฮสต์ A ทำการ ping ไปที่โฮสต์ B โดยใช้ Ping6.exe

ตัวอย่าง ping6 fe80::260:97ff:fe02:6ea5

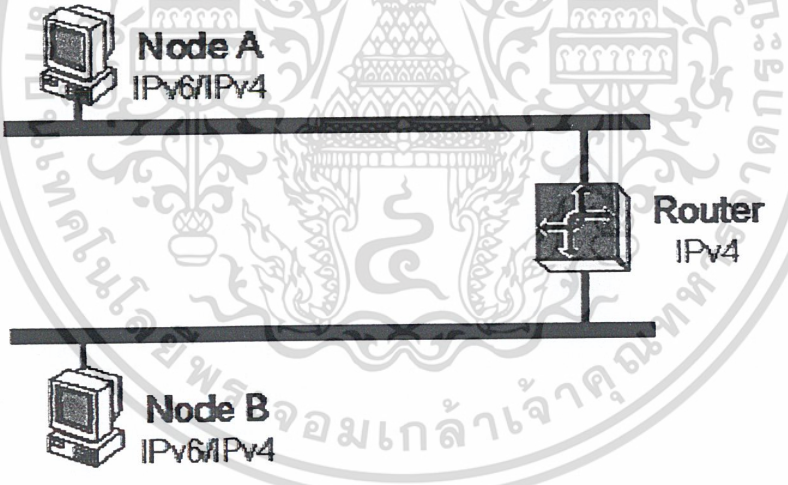
สามารถกำหนดค่าแอ็ดเดรสต้นทางได้สำหรับแพ็กเก็ตร้องขอที่ส่งออกไปได้โดยใช้ ping6 -s

ตัวอย่างเช่น ping6 -s fe80::210:5aff:feaa:20a2%4 fe80::260:97ff:fe02:6ea5

เมื่อทำการ ping ลิ้งค์-โลคัลหรือไซต์-โลคัลควรจะกำหนด scope-ID เพื่อให้แอ็ดเดรสปลายทางไม่กำกวม โดยใช้ address%scope-ID สำหรับลิ้งค์-โลคัลแอ็ดเดรสแล้ว scope-ID มีค่าเท่ากับหมายเลขอินเทอร์เฟซที่แสดงจากคำสั่ง ipv6 if

แบบที่ 2 ซับเน็ตต่างกันโดยส่งผ่านเครือข่าย IPv4 (6to4)

6to4 เป็นวิธีการติดต่อระหว่าง IPv6 โฮสต์โดยผ่านเครือข่าย IPv4 ที่มีอยู่แล้ว โดยใช้พรีฟิกซ์เฉพาะเพื่อแบ่งแยกกับ IPv6 แอ็ดเดรส 6to4 เหมือนกับไอเอสพีเทียม (pseudo-ISP) ที่ใช้ทำการเชื่อมต่อ สามารถใช้ 6to4 เพื่อติดต่อโดยตรงกับไซต์ 6to4 อื่น และสามารถใช้ 6to4 รีเลย์ เพื่อติดต่อกับ 6Bone ไซต์ วิธี 6to4 ไม่จำเป็นต้องใช้เราเตอร์ที่สนับสนุน IPv6 แพ็กเก็ตของ IPv6 จะถูกเอ็นแค็ปอยู่ภายใน IPv4 แพ็กเก็ต



รูปที่ 2 แสดงโครงสร้างของการคอนฟิกแบบ 6to4 ผ่าน IPv4 เราเตอร์

สิ่งที่จำเป็นสำหรับ 6to4 คือ แอ็ดเดรส IPv4 ที่สามารถใช้งานได้หนึ่งแอ็ดเดรส สมมติว่าไซต์ประกอบด้วยกลุ่มของเครื่องคอมพิวเตอร์ที่สนับสนุน IPv6 โดยอาจจะอิมพลีเมนต์โดยใช้ Microsoft IPv6 แสตก หรือ อิมพลีเมนต์อื่นๆ เครื่องทั้งหมดเชื่อมต่อกันโดยใช้อีเทอร์เน็ตหรือ 6-over-4 IPv4 แอ็ดเดรสที่มีจะกำหนดให้กับคอมพิวเตอร์เครื่องใดเครื่องหนึ่งที่อิมพลีเมนต์โดยใช้ Microsoft IPv6 แสตก คอมพิวเตอร์เครื่องนี้จะถูกใช้เป็น 6to4 เกตเวย์

ถ้า IPv4 แอ็ดเดรสที่มีเป็นส่วนของ 10.0.0.0/8 , 172.16.0.0/12 หรือ 192.168.0.0/16 หรือเป็น Automatic Private IP Addressing (APIPA) 169.254.0.0/16 ที่ถูกใช้โดย Windows 98 และ Windows 2000 แล้วจะไม่สามรถใช้ได้

6to4cfg.exe สามารถตั้งค่าคอนฟิกสำหรับ 6to4 ได้อย่างอัตโนมัติ และสามารถค้นหา IPv4 แอ็ดเดรสที่มีและสร้าง 6to4 พรีฟิกซ์ได้ รวมทั้งสามารถคอนฟิกได้โดยตรงและเขียนเป็นสคริปต์ไว้ใช้ภายหลังได้ รูปแบบคำสั่งของ 6to4cfg.exe เป็นดังนี้

6to4cfg [-r] [-s] [-u] [-R relay] [-b] [-S address] [filename]

6to4cfg [filename]

เขียนคอนฟิกสคริปต์ลงไฟล์หากมีการระบุ filename หากระบุค่า filename เป็น con จะเขียนผลลัพธ์ลงหน้าจอคอนโซล หากไม่ระบุชื่อไฟล์จะทำการตั้งค่าคอนฟิกใหม่

6to4cfg -r

ตั้งให้เป็น 6to4 เกตเวย์เราเตอร์ของเครือข่าย

6to4cfg -s

กำหนดให้ใช้ไซต์-โลคัลแอ็ดเดรสได้ ใช้ร่วมกับ –r

6to4cfg -u

ระบุให้ 6to4 คอนฟิกถูกกลับเป็นตรงกันข้าม (reverse)

6to4cfg -R relay

ใช้กำหนดชื่อหรือ IPv4 แอ็ดเดรสของ 6to4 รีเลย์เราเตอร์ มีค่าเริ่มต้นเป็น 6to4.ipv6.microsoft.com ซึ่งเป็น 6to4 รีเลย์เราเตอร์ของ Microsoft

6to4cfg -b

กำหนดให้ 6to4cfg.exe เลือกรีเลย์แอ็ดเดรสที่ดีที่สุดก่อน

6to4cfg -S address

กำหนด IPv4 แอ็ดเดรสสำหรับพรีฟิกซ์ของ 6to4

ตัวอย่างการคอนฟิก 6to4

สมมติให้ 6to4 เกตเวย์เป็น 172.31.42.239 แอ็ดเดรสนี้จะถูกรวมกับพรีฟิกซ์ 2002::/16 ได้เป็นพรีฟิกซ์ขนาด 48 บิตเป็น 2002:ac1f:2aef::/48 ตัวอย่างกำหนดให้ใช้ซับเน็ต 0 สำหรับการคอนฟิกโดยผู้ใช้บนเครื่องที่เป็นเกตเวย์ และซับเน็ต 1 สำหรับการคอนฟิกแบบอัตโนมัติบนเครื่องอื่นๆในลิ้งค์เดียวกัน

1. ใช้ Ipv6.exe บนเครื่องที่เป็นเกตเวย์เพื่อเริ่มการใช้งาน IPv6

   ipv6 rtu 2002::/16 2

   ค่า 2002::/16 เป็นค่าพรีฟิกซ์ และค่า 2 เป็นหมายเลขอินเทอร์เฟซที่ใช้ ซึ่งค่า 2 เป็นอินเทอร์เฟซเทียมใช้สำหรับคอนฟิกทันเนิล ออโตเมติกทันเนิล และ 6to4 เมื่อแอ็ดเดรสปลายทางมีพรีฟิกซ์เป็น 2002::/16 32 บิตที่ต่อจากพรีฟิกซ์ถูกใช้เป็น IPv4 แอ็ดเดรสสำหรับการเอ็นแค็ป

2. คอนฟิกแอ็ดเดรส 6to4 บนเครื่องที่เป็นเกตเวย์

ipv6 adu 2/2002:ac1f:2aef::ac1f:2aef

คำสั่ง ipv6 adu ทำการอัพเดตแอ็ดเดรส สามารถเพิ่ม ลบ แก้ไข แอ็ดเดรสบนอินเทอร์เฟซ จากคำสั่งเป็นการกำหนดแอ็ดเดรส 2002:ac1f:2aef::ac1f:2aef ให้กับอินเทอร์เฟซหมายเลข 2 โดยได้มาจากไซต์พรีฟิกซ์ 2002:ac1f:2aef::/48 รวมกับซับเน็ต 0 เป็น 2002:ac1f:2aef::/64 แล้วนำไปรวมกับ interface identifer ขนาด 64 บิต โดยแบบแผนแล้วกำหนดให้ interface identifer เป็น IPv4 แอ็ดเดรสสำหรับแอ็ดเดรสสำหรับอินเทอร์เฟซหมายเลขสอง

คำสั่งทั้งสองเพียงพอที่ใช้ติดต่อกับ 6to4 ไซต์อื่นๆ อาจลองทดสอบโดยการติดต่อกับ 6to4 ไซต์ของ Microsoft โดยใช้คำสั่ง

ping6 2002:836b:9820::836b:9820

เพื่อที่จะติดต่อกับ 6Bone จะต้องสร้างคอนฟิกทันเนิลสู่ 6to4 รีเลย์ โดยใช้ 6to4 รีเลย์เราเตอร์ของ Microsoft 131.107.152.32 โดยใช้คำสั่ง

ipv6 rtu ::/0 2/::131.107.152.32 pub life 1800

คำสั่ง ipv6 rtu ใช้สำหรับแก้ไขตารางเราติง

ค่า ::/0 ใช้เป็นพรีฟิกซ์ การกำหนดให้มีความยาวศูนย์หมายถึงค่าดีฟอลต์เราต์

ค่า 2/::131.107.152.32 ใช้กำหนดค่าโหนดถัดไป แพ็กเก็ตที่มีค่าพรีฟิกซ์ตรงจะถูกส่งไปยัง ::131.107.152.32 โดยใช้อินเทอร์เฟซหมายเลขสอง

ค่า pub กำหนดให้สามารถพับลิคได้โดยกำหนดระยะเวลา 30 นาที

ทำการทดสอบโดยการติดต่อกับ 6Bone ไซต์

ping6 3ffe:1cff:0:f5::1

ขั้นตอนสุดท้าย ทำการกำหนดซับเน็ตพรีฟิกซ์ให้กับอินเทอร์เฟซอื่นๆ

ipv6 rtu 2002:ac1f:2aef:1::/64 3 pub life 1800

ipv6 rtu 2002:ac1f:2aef:2::/64 4 pub life 1800

คำสั่ง ipv6 rtu นี้ใช้กำหนดพรีฟิกซ์ 2002:ac1f:2aef:1::/64 ให้กับอินเทอร์เฟซหมายเลข 3

คำสั่งต่อไปให้เกตเวย์ทำงานเป็นเราเตอร์

ipv6 ifc 2 forw

ipv6 ifc 3 forw adv

ipv6 ifc 4 forw adv

คำสั่ง ipv6 ifc ใช้กำหนดแอททริบิวต์ของอินเทอร์เฟซ

อินเทอร์เฟซหมายเลขสองไม่จำเป็นต้องกำหนดให้แอดเวอร์ไทซ์เพราะเป็นอินเทอร์เฟซเทียม หลังจากป้อนคำสั่งเหล่านี้ Microsoft IPv6 แสตกจะทำการคอนฟิกแอดเดรสบนอินเทอร์เฟซหมายเลข 3 และ 4 อินเทอร์เฟซทั้งสองจะทำการส่ง router advertisement ในช่วงเวลา 3-10 วินาที โฮสต์ที่ได้รับ

จะทำการคอนฟิกตนเองโดยอัตโนมัติกับดีฟอลต์เราต์และ 6to4 แอ็ดเดรสจากพรีฟิกซ์และจะสามารถติดต่อกับ 6to4 อื่นและ 6Bone ผ่านเครื่องที่เป็นเกตเวย์

การตรวจสอบความผิดพลาดของการคอนฟิก 6to4

สามารถทำได้ดังนี้

1. ตรวจสอบว่า IPv4 เชื่อมต่อกับ 6to4 รีเลย์เราเตอร์หรือไม่โดยใช้คำสั่ง

   ping 6to4.ipv6.microsoft.com

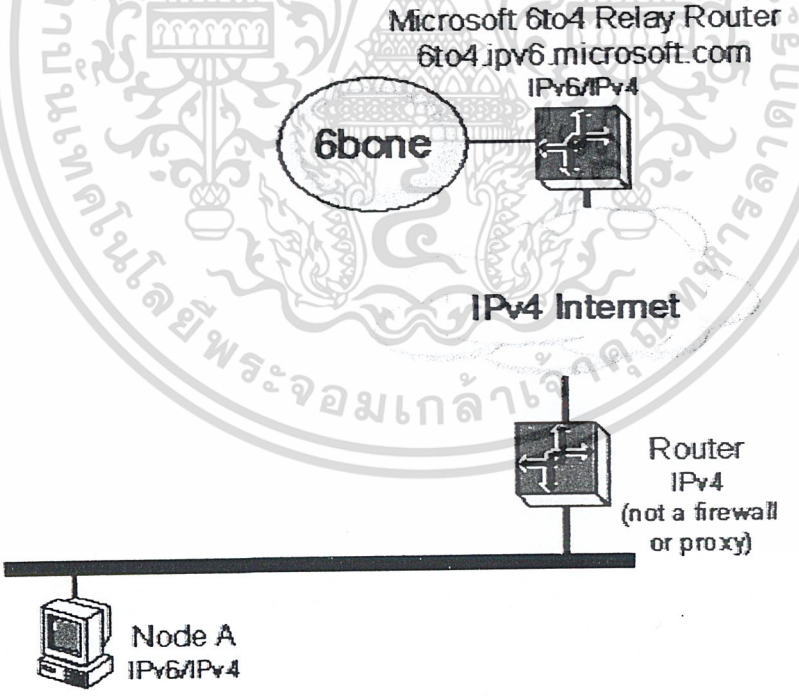
   ถ้าไม่สำเร็จแสดงว่าไม่สามารถเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตได้
2. ครวจสอบการเอ็นแค็ป IPv6 โดยใช้ออโตเมติกทันเนิล

   ping6 ::131.107.152.32

   ถ้าไม่สำเร็จ แสดงว่าอาจจะมีกำแพงไฟเสมือนหรือ NAT ระหว่างคอมพิวเตอร์และอินเทอร์เน็ต
3. ตรวจสอบผลลัพธ์ของคำสั่ง ipv6 rt ควรจะมี 2002::/16 --> 2 และตรวจสอบผลลัพธ์ของคำสั่ง ipv6 if 2 ควรจะมีแอ็ดเดรสที่มีพรีฟิกซ์เป็น 2002::/16

แบบที่ 3 ติดต่อกับ 6Bone

วิธีที่ดีที่สุดคือการคอนฟิก 6to4 ผ่านรีเลย์เราเตอร์ตามที่ได้เสนอไปแล้ว



รูปที่ 3 การเชื่อมต่อกับ 6Bone ผ่านรีเลย์เราเตอร์ของ Microsoft

สามารถทำการทดสอบได้โดย ping ติดต่อกับเครื่องคอมพิวเตอร์ใน 6Bone ตามตัวอย่างต่อไปนี้

ping6 6bone-gw.ipv6.sics.se

ping6 sipper.ipv6.zk3-x.dec.com

ping6 www.6bone.net

ping6 carmen.ipv6.cselt.it

ping6 ipv6.research.microsoft.com

ไซต์ใน 6Bone บางไซต์อาจจะไม่สามารถเข้าถึงได้ ผู้ใช้ที่มีประสบการณ์ในการแก้ปัญหาอาจจะใช้คำสั่ง tracert3 -d address ช่วยในการตรวจสอบได้

แบบที่ 4 การใช้ IPSec ระหว่าง 2 โฮสต์ภายในลิ้งค์เดียวกัน

การตั้งคอนฟิกนี้จะสร้าง IPSec Security Association (SA) ระหว่างสองโฮสต์ในซับเน็ตเดียวกัน เพื่อจะทำการพิสูจน์ตัวจริงโดยใช้ส่วนหัวของการพิสูจน์ตัวจริง (Authentication Header – AH) และ Message Digest 5 (MD5) กับแฮชชิ่งอัลกริทึ่ม ในตัวอย่างนี้การคอนฟิกจะแสดงความปลอดภัยระหว่างโฮสต์เพื่อนบ้านสองโฮสต์ โดยกำหนดให้

โฮสต์ 1 มีลิ้งค์-โลคัลแอ็ดเดรสเป็น FE80::2AA:FF:FE53:A92C และ

โฮสต์ 2 มีลิ้งค์-โลคัลแอ็ดเดรสเป็น FE80::2AA:FF:FE92:D0F1

1. ที่โฮสต์ 1 สร้างไฟล์เปล่าของ Association (.SAD) และ Policy (.SPD) โดยใช้คำสั่ง ipsec6 c ตัวอย่างเช่น

   ipsec6 c test

   จะได้ผลเป็นไฟล์สองไฟล์คือ test.sad และ test.spd

2. ที่โฮสต์ 1 ทำการแก้ไขไฟล์ test.spd โดยการเพิ่ม Policy ที่รักษาความปลอดภัยทุกการสื่อสารระหว่างโฮสต์ทั้งสอง ตารางต่อไปนี้จะแสดง Policy ที่ถูกเพิ่มเข้าไป

ชื่อฟิลด์	ค่าตัวอย่าง
Policy	2
RemoteIPAddr	- FE80::2AA:FF:FE92:D0F1
LocalIPAddr	- *
RemotePort	- *
Protocol	- *
LocalPort	- *
IPSecProtocol	AH
IPSecMode	TRANSPORT
RemoteGWIPAddr	*
SABundleIndex	NONE
Direction	BIDIRECT

Action	APPLY
InterfaceIndex	0

ตารางที่ 1 แสดงตัวอย่าง Policy บนโฮสต์ 1

ต้องใส่เซมิโคลอน (':') เมื่อสิ้นสุดเอ็นทรี่ด้วย โดยจะเรียงลำดับเอ็นทรี่จากมากไปหาน้อย

3. ที่โฮสต์ 1 ทำการแก้ไขไฟล์ test.sad โดยการเพิ่ม SA ระหว่างโฮสต์ทั้งสอง จะต้องสร้าง Association สองตัว โดยตัวแรกสำหรับการสื่อสารที่เข้าสู่โฮสต์ 2 และอีกตัวสำหรับการสื่อสารที่มาจากโฮสต์ 2 ตารางต่อไปนี้แสดงตัวอย่าง SA เอ็นทรี่สำหรับการสื่อสารที่เข้าสู่โฮสต์ 2

ชื่อฟิลด์	ค่าตัวอย่าง
SAEntry	2
SPI	3001
SADestIPAddr	FE80::2AA:FF:FE92:D0F1
DestIPAddr	POLICY
SrcIPAddr	POLICY
Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	OUTBOUND
SecPolicyIndex	2

ตารางที่ 2 แสดงตัวอย่าง SA ที่เข้าสู่โฮสต์ 2 บนโฮสต์ 1

และตารางต่อไปนี้แสดงตัวอย่าง SA ที่มาจากโฮสต์ 2

ชื่อฟิลด์	ค่าตัวอย่าง
SAEntry	1
SPI	3000
SADestIPAddr	FE80::2AA:FF:FE53:A92C
DestIPAddr	POLICY

SrcIPAddr	POLICY
Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	INBOUND
SecPolicyIndex	2

ตารางที่ 3 แสดงตัวอย่าง SA ที่มาจากโฮสต์ 2 บนโฮสต์ 1

ต้องใส่เซมิโคลอน (':') เมื่อสิ้นสุดเอ็นทรี่ด้วย โดยจะเรียงลำดับเอ็นทรี่จากมากไปหาน้อย

4. ที่โฮสต์ 1 สร้างเท็กซ์ไฟล์ที่บรรจุข้อความที่จะใช้ทำการพิสูจน์ตัวจริงกับโฮสต์ 2 ตัวอย่างเช่น ไฟล์ test.key มีข้อความ "This is a test" อยู่ โดยจะต้องมีเครื่องหมาย " " ปิดข้อความด้วยเพื่อใช้เป็นคีย์สำหรับ ipsec6

   ในชุด Microsoft IPv6 แสดก นั้นสนับสนุนการสร้างคีย์โดยผู้ใช้โดยตรงผ่านเท็กซ์ไฟล์เท่านั้น ในตัวอย่างนี้กำหนดให้ใช้คีย์เดียวกันทั้งสองทิศทาง ผู้ใช้สามารถกำหนดคีย์ไฟล์ต่างกันได้โดยผ่านค่า KeyFile ในไฟล์ .sad

5. ที่โฮสต์ 2 สร้าง Association และ Policy เปล่าโดยคำสั่ง ipsec6 c เหมือนในขั้นตอนที่ 1 ในตัวอย่างนี้กำหนดให้ใช้ชื่อ test.sad และ test.spd เหมือนกับที่โฮสต์ 1 ผู้ใช้สามารถกำหนดตามความเหมาะสมได้

6. ที่โฮสต์ 2 แก้ไขไฟล์ .spd โดยเพิ่มเอ็นทรี่ตามตารางต่อไปนี้

ชื่อฟิลด์	ค่าตัวอย่าง
Policy	2
RemoteIPAddr	- FE80::2AA:FF:FE53:A92C
LocalIPAddr	- *
RemotePort	- *
Protocol	- *
LocalPort	- *
IPSecProtocol	AH
IPSecMode	TRANSPORT
RemoteGWIPAddr	*

SABundleIndex	NONE
Direction	BIDIRECT
Action	APPLY
InterfaceIndex	0

ตารางที่ 4 แสดงตัวอย่าง Policy บนโฮสต์ 2

7. ที่โฮสต์ 2 แก้ไขไฟล์ .sad เพิ่มเอ็นทรี่ของ Association ตามตารางต่อไปนี้ตามลำดับ

SAD File Field Name	Example Value
SAEntry	2
SPI	3001
SADestIPAddr	FE80::2AA:FF:FE92:D0F1
DestIPAddr	POLICY
SrcIPAddr	POLICY
Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	INBOUND
SecPolicyIndex	2

ตารางที่ 5 แสดงตัวอย่าง SA ที่มาจากโฮสต์ 1 บนโฮสต์ 2

SAD File Field Name	Example Value
SAEntry	1
SPI	3000
SADestIPAddr	FE80::2AA:FF:FE53:A92C
DestIPAddr	POLICY
SrcIPAddr	POLICY
Protocol	POLICY

SABundleIndex	NONE
Direction	BIDIRECT
Action	APPLY
InterfaceIndex	0

ตารางที่ 4 แสดงตัวอย่าง Policy บนโฮสต์ 2

7. ที่โฮสต์ 2 แก้ไขไฟล์ .sad เพิ่มเอ็นทรี่ของ Association ตามตารางต่อไปนี้ตามลำดับ

SAD File Field Name	Example Value
SAEntry	2
SPI	3001
SADestIPAddr	FE80::2AA:FF:FE92:D0F1
DestIPAddr	POLICY
SrcIPAddr	POLICY
Protocol	POLICY
DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	INBOUND
SecPolicyIndex	2

ตารางที่ 5 แสดงตัวอย่าง SA ที่มาจากโฮสต์ 1 บนโฮสต์ 2

SAD File Field Name	Example Value
SAEntry	1
SPI	3000
SADestIPAddr	FE80::2AA:FF:FE53:A92C
DestIPAddr	POLICY
SrcIPAddr	POLICY
Protocol	POLICY

DestPort	POLICY
SrcPort	POLICY
AuthAlg	HMAC-MD5
KeyFile	Test.key
Direction	OUTBOUND
SecPolicyIndex	2

ตารางที่ 6 แสดงตัวอย่าง SA ที่เข้าสู่โฮสต์ 1 บนโฮสต์ 2

8. ที่โฮสต์ 2 สร้างเท็กซ์ไฟล์ชื่อ test.key บรรจุข้อความ "This is a test" เหมือนขั้นตอนที่ 5
9. ที่โฮสต์ 1 เพิ่ม Policy และ Association ที่สร้างมาเข้าสู่ระบบโดยพิมพ์คำสั่ง

   ipsec6 a test
10. ที่โฮสต์ 2 เพิ่ม Policy และ Association ที่สร้างมาเข้าสู่ระบบโดยพิมพ์คำสั่ง

    ipsec6 a test
11. ทดสอบ ping โฮสต์ 1 จากโฮสต์ 2 โดยใช้คำสั่ง ping6 หากใช้โปรแกรมดักจับแพ็กเก็ตดูจะพบแพ็กเก็ตร้องขอและตอบรับของ ICMPv6 โดยมีส่วนเพิ่มเติมส่วนหัวของการพิสูจน์ตัวจริง (Authentication header) อยู่ระหว่างส่วนหัวของ IPv6 และส่วนหัวของ ICMPv6

หัวข้อเพิ่มเติม

Router advertisements

ข้อมูลของ router advertisement จะนำมาจาก published route ในตารางเราติงโดยอัตโนมัติ เส้นทางที่ไม่เป็น published จะใช้ตามปกติแต่ไม่ถูกใช้ในการทำ router advertisement

Router advertisement จะบรรจุอ็อปชั่นลิ้งค์-เลเยอร์แอ็ดเดรสต้นทางและอ็อปชั่นของ MTU ค่าของอ็อปชั่น MTU จะนำมาจาก MTU ของลิ้งค์ที่เชื่อมต่ออินเทอร์เฟซนั้นอยู่ ค่านี้สามารถเปลี่ยนได้โดยใช้คำสั่ง ipv6 ifc mtu

Router advertisement จะมีเพียงแค่เส้นทางที่มีช่วงชีวิตไม่เป็นศูนยเท่านั้นถ้ามีค่าเส้นทางดีฟอลต์เราต์ ซึ่งเส้นทางที่เป็นดีฟอลต์เราต์จะมีความยาวพรีฟิกซ์เป็นศูนย์

ไซต์พรีฟิกซ์

คำสั่ง ipv6 rtu ทำให้พรีฟิกซ์ในลิ้งค์สามารถคอนฟิกความยาวของไซต์พรีฟิกซ์ได้ ตัวอย่างเช่น

ipv6 rtu 2002:836b:9820:2::/64 4 pub life 1800 sp1 48

เป็นการระบุพรีฟิกซ์ภายในลิ้งค์ของอินเทอร์เฟซหมายเลข 4 ให้เป็น published (ใช้ใน router advertisement) กำหนด lifetime เป็น 1800 วินาที และกำหนดความยาวของไซต์พรีฟิกซ์ในการทำ router advertisement เป็น 48

การกำหนดค่าไซต์พรีฟิกซ์มีผลให้เกิดเอ็นทรี่ขึ้นในตารางไซต์พรีฟิกซ์ซึ่งสามารถเรียกดูได้โดยใช้คำสั่ง ipv6 spt

แอ็ดเดรสปลายทางที่เป็นมัลติแคสต์

เมื่อมีการกำหนดแอ็ดเดรสปลายทางเป็นมัลติแคสต์ ในชุด Microsoft IPv6 แสตกต้องการแอปพลิเคชั่นที่ระบุอินเทอร์เฟซส่งออก ทำได้โดยใช้ IPV6_MULTICAST_IF หรือโดยการระบุแอ็ดเดรสต้นทางให้ซอคเก็ต

หรืออาจจะทำการระบุดีฟอลต์อินเทอร์เฟซสำหรับมัลติแคสต์แอ็ดเดรสหนึ่งใดๆ ระบุเป็นช่วงแอ็ดเดรส หรืออาจจะทั้งหมดก็ได้ ตามตัวอย่างต่อไปนี้

ipv6 rtu ff02::5/128 3

ipv6 rtu ff0e::/16 3

ipv6 rtu ff00::/8 3

ออโตเมติกทันเนิลและ 6to4

ทั้งออโตเมติกทันเนิลโดยใช้ IPv4-คอมแพติเบิลแอ็ดเดรสและ 6to4 ทำงานโดยผ่านการเลือกเส้นทางโดยพรีฟิกซ์บนอินเทอร์เฟซหมายเลข 2 เลข 32 บิตที่ต่อจากส่วนพรีฟิกซ์ถูกนำไปใช้เป็นแอ็ดเดรสปลายทางในส่วนหัวของ IPv4 ที่เอ็นแค็ป

ออโตเมติกทันเนิลใช้พรีฟิกซ์ ::/96 ซึ่งถูกกำหนดเป็นดีฟอลต์ในตารางเราติงอยู่แล้ว แต่ 6to4 ซึ่งใช้พรีฟิกซ์ 2002::/16 ไม่ได้ถูกกำหนดเป็นดีฟอลต์

ภาคผนวก ข
การใช้งาน IPv6 ในระบบปฏิบัติการ Windows NT

สามารถดาวโหลดคูอัล-แสตกได้จาก http://www.research.microsoft.com/msripv6/msripv6.htm โดยแสตกสำหรับ Windows NT ที่ได้มีการพัฒนามาล่าสุดนั้นเป็นเวอร์ชั่น 1.4 หากมีการติดตั้งในเวอร์ชั่นก่อนหน้านี้จะต้อง Uninstall ออกเสียก่อน สำหรับการใช้งานในคำสั่งต่างๆนั้นจะเหมือนกับการคอนฟิกใน Windows 2000 แต่จะมีส่วนต่างในขั้นตอนการติดตั้งและปัญหาบางประการที่เกิดขึ้น

สภาพแวดล้อมของระบบที่จำเป็น

โฮสต์ที่จะติดตั้งคูอัล-แสตกจะต้องติดตั้งระบบปฏิบัติการ Windows NT 4.0 ผู้พัฒนาได้ออกแบบให้สามารถใช้ได้กับทุก Service Pack แต่ได้มีการทดสอบและรับรองผลกับ Service Pack 3 ขึ้นไปเท่านั้น การ์ดอีเทอร์เน็ตและ FDDI ที่สามารถทำงานร่วมกับ Windows NT 4.0 ได้สามารถใช้กับแสตกนี้ได้

การติดตั้ง

เปิดหน้าต่าง Control Panel เลือกหมวด Network เลือกที่แถบ Protocol ติดตั้งได้โดยคลิกที่ปุ่ม Add ขณะติดตั้งจะมีการก็อปปี้โปรโตคอลแสตก tcpip6.sys ไปที่ %SystemRoot%\system32\drivers

การใช้งาน IPv6 ร่วมกับ Internet Exploerer

จำเป็นจะต้องแทนที่ไฟล์ wininet.dll กับเวอร์ชั่นใหม่ที่ได้มากับคูอัลแสตก เนื่องจากไฟล์นี้มีกี่ใช้งานอยู่ตลอดจึงไม่สามารถแทนที่ไฟล์ได้ทันที ต้องเปลี่ยนชื่อ wininet.dll ให้เป็นชื่ออื่นเสียก่อนจึงก็อปปี้ไฟล์ที่มากับคูอัลแสตกลงไปแทน ไฟล์นี้จะทำให้ Internet Explorer 4.01 ที่ติดตั้ง Service Pack 1 หรือเวอร์ชั่นใดๆที่สูงกว่านี้รู้จักแอ็ดเดรสแบบ IPv6 ตามแบบที่กำหนดใน RFC 2732 ตัวอย่างเช่น http://[2002:836b:4179::836b:4179]

ปัญหาที่เกิดขึ้น

การติดตั้งคูอัล-แสตกอาจจะทำให้เกิดปัญหาบางประการขึ้นซึ่งจำเป็นจะต้องได้รับการแก้ไข แต่การแก้ไขนั้นไม่รับรองผลสำหรับโฮสต์ที่ได้ติดตั้ง Service Pack ใดๆ (การแก้ไขรับรองผลกับโฮสต์ที่ไม่ได้ติดตั้ง Service Pack ใดๆเลย) ผู้พัฒนาจึงไม่ได้กำหนดให้ทำการแก้ไขปัญหานี้ตั้งแต่ตอนติดตั้ง ผู้ติดตั้งจะต้องทำการติดตั้งไฟล์แก้ไขเอง การแก้ไขสามารถทำได้ดังนี้

1. แทนที่ไฟล์ afd.sys (สำหรับ Service Pack 3 หรือน้อยกว่า) หรือ afd.sp4 (สำหรับ Service Pack 4 ขึ้นไป) ในไดเร็กตอรี่ %SystemRoot%\system32\drivers\afd.sys
2. หากติดตั้ง Service Pack ต่ำกว่าระดับ 4 จะต้องแทนที่ไฟล์ msafd.dll กับไฟล์ที่มากับคูอัล-แสตก ในไดเร็กตอรี่ %SystemRoot%\System32\msafd.dll
3. หากติดตั้ง Service Pack ต่ำกว่าระดับ 4 จะต้องแทนที่ไฟล์ nslookup.exe กับไฟล์ที่มากับคูอัล-แสตก ในไดเร็กตอรี่ %SystemRoot%\System32\nslookup.exe

ภาคผนวก ค
การใช้งาน IPv6 ในระบบปฏิบัติการ Windows XP

ใน Windows XP ได้รวม IPv6 ดูอัล-แสตกมากับระบบปฏิบัติการแล้ว แต่จะไม่ติดตั้งให้อัตโนมัติ ผู้ใช้จะต้องทำการติดตั้งเอง ทำได้โดยการพิมพ์ ***ipv6 install*** ที่ Command Prompt สำหรับรายละเอียดการคอนฟิกนั้นจะเหมือนกับใน Windows 2000 จะแตกต่างกันเพียงรายละเอียดบางอย่างเช่น หมายเลขอินเทอร์เฟซ โดยผู้ใช้สามารถดูรายละเอียดได้ที่ไฟล์ ipv6.chm ในโฟลเดอร์ Help ของ Windows XP สำหรับการยกเลิกการติดตั้งสามารถทำได้โดยพิมพ์ ***ipv6 uninstall*** ที่ Command Prompt

ภาคผนวก ง
การใช้งาน IPv6 ในระบบปฏิบัติการ Linux

ในขั้นแรกต้องตรวจสอบว่าระบบปฏิบัติการ Linux ที่ใช้อยู่นั้นมีการสนับสนุน IPv6 หรือไม่ ถ้ารุ่นที่ใช้อยู่นั้นไม่สนับสนุน IPv6 สิ่งที่ต้องทำคือหา Kernel ที่สนับสนุนมา และให้คอมไพล์ Kernel และในหัวข้อ Network Options ให้เลือกเป็น "IPv6 protocol" สำหรับการคอมไพล์และการติดตั้ง Kernel ดูได้จากคู่มือของระบบปฏิบัติการที่ให้มา สำหรับการคอนฟิกให้ใช้งาน IPv6 นั้นสามารถทำได้ดังนี้ (ทดลองโดยใช้ Linux Redhat 7.2)

ในไฟล์ /etc/sysconfig/network ให้เพิ่มคำสั่งดังนี้

NETWORKING_IPV6=yes อนุญาติให้ใช้งาน IPv6

IPV6FORWARDING=yes/no ถ้าเป็น yes หมายความว่าอนุญาติให้มีการส่งแพ็กเก็ตผ่านไปได้ (สำหรับโหนดที่จะให้ใช้ทำหน้าที่เป็นเราเตอร์) ถ้าเป็น no หมายความว่าไม่อนุญาติให้ส่งแพ็กเก็ตผ่านไป

IPV6_AUTOTUNNEL=yes/no ถ้าเป็น yes หมายความว่าอนุญาติการทำทันเนิลโดยอัตโนมัติสำหรับ IPv6-in-IPv4 tunnel interface (sit0) ถ้าเป็น no หมายความว่าไม่อนุญาติให้ใช้งาน

IPV6_MTU="ค่า MTU" ค่า MTU ที่ใช้ (ไม่จะเป็นต้องกำหนด)

สำหรับการคอนฟิกอินเทอร์เฟซอื่นๆ นั้นสามารถทำได้ดังตัวอย่างต่อไปนี้

ในไฟล์ /etc/sysconfig/network-scripts/ifcfg-***device*** (***device*** ให้เปลี่ยนตามชื่ออินเทอร์เฟสที่ต้องการคอนฟิกเช่น eth0, eth1, ppp)

IPV6INIT=yes อนุญาติการใช้งาน IPv6 สำหรับอินเทอร์เฟสนี้

IPV6ADDR="IPv6 แอดเดรส" กำหนด IPv6 แอดเดรสให้กับอินเทอร์เฟสเช่น (3ffe:ffff:0:f101::1/64)

IPV6ADDR_SECONDARIES="IPv6 แอดเดรส" กำหนด IPv6 แอดเดรสอีกแอดเดรสให้กับอินเทอร์เฟส คำสั่งนี้ไม่จะเป็นต้องกำหนด

คำสั่งต่อไปนี้สำหรับการติดตั้งอุโมงค์แบบคงที่

IPV6TUNNELIPV4="IPv4 แอดเดรส" ใช้บอก IPv4 แอดเดรสของอุโมงต์ปลายทาง

IPV6TUNNELIPV4LOCAL="IPv4 แอดเดรส" ใช้บอก IPv4 แอดเดรสของอุโมงค์ต้นทาง

คำสั่งต่อไปนี้สำหรับการติดตั้ง 6to4

IPV6TO4INIT=yes/no ใช้บอกว่าอินเทอร์เฟสนี้ให้อนุญาติใช้การทันเนิลหรือไม่

IPV6TO4_RELAY="IPv4 แอดเดรส" ใช้บอกแอดเดรสของรีเลย์เราเตอร์ (แอดเดรสแบบ IPv4)

IPV6TO4_IPV4ADDR="IPv4 แอดเดรส" จำเป็นต้องใช้คำสั่งนี้ก็ต่อเมื่อถ้ามีการใช้ NAT หรืออุปกรณ์อื่นที่ทำให้ไม่สามารถใช้ IPv4 แอดเดรสที่ได้รับอย่างอัตโนมัติจากอินเทอร์เฟส

ในไฟล์ /etc/sysconfig/static-routes-ipv6 ใช้สำหรับการเพิ่มตารางเราติงโดยมีรูปแบบดังนี้

device ***prefix*** ตัวอย่างเช่น

sit0 3ffe::/16

sit1 2000::/3

ตัวอย่างการติดตั้งอุโมงค์

ตัวอย่างที่ 1 (Freenet6 Point-to-Point-Tunnel)

IPv4 แอดเดรสของอุโมงค์ปลายทาง (Freenet6) คือ 206.1236.31.102

IPv6 แอดเดรสของอุโมงค์ปลายทางคือ 3ffe:b00:c18:1ffff:0:0:0:a1f

IPv4 แอดเดรสของอุโมงค์ต้นทาง (ของเครือข่ายเรา) คือ 62.155.142.69

IPv6 แอดเดรสของเราคือ 3ffe:b00:c18:1ffff:0:0:0:a1e

ตารางเราติง 3ffe::/16 2000::/3

การติดตั้ง

ในไฟล์ /etc/sysconfig/network-scripts/ifcfg-sit2

```
DEVICE="sit2"
BOOTPROTO="none"
ONBOOT="yes"
IPV6INIT="yes"
```

IPV6TUNNELIPV4="206.123.31.102"

IPV6ADDR="3ffe:b00:c18:1fff:0:0:0:a1f/127"

ในไฟล์ /etc/sysconfig/static-routes-ipv6

sit2 3ffe::/16

sit2 2000::/3

ตรวจสอบได้ดังนี้

```
# ip tunnel
sit0: ipv6/ip  remote any  local any  ttl 64  nopmtudisc
sit2: ipv6/ip  remote 206.123.31.102  local any  ttl 64

# ifconfig sit2
sit2      Link encap:IPv6-in-IPv4
          inet6 addr: fe80::3e9b:8e45/10 Scope:Link
          inet6 addr: fe80::c0a8:101/10 Scope:Link
          inet6 addr: 3ffe:b00:c18:1fff::a1f/127 Scope:Global

# route -A inet6 |grep sit2
3ffe:b00:c18:1fff::a1e/127    ::                  UA    256    4        0 sit2
3ffe::/16                     ::                  U     1      0        0 sit2
2000::/3                      ::                  U     1      0        0 sit2
fe80::/10                     ::                  UA    256    0        0 sit2
ff00::/8                      ::                  UA    256    0        0 sit2
```

ตัวอย่างที่ 2 (Hurricane Electric tunnel broker)

IPv4 แอดเดรสของอุโมงค์ปลายทาง (Hurricane) คือ 64.71.128.26

IPv6 แอดเดรสของอุโมงค์ปลายทางคือ 3ffe:1200:3028:ff01::2e6

IPv4 แอดเดรสของอุโมงค์ต้นทาง (ของเครือข่ายเรา) คือ 62.155.142.69

IPv6 แอดเดรสของเราคือ 3ffe:1200:3028:ff01::2e7

ตารางเราติง 3ffe::/16 2000::/3

การติดตั้ง

ในไฟล์ /etc/sysconfig/network-scripts/ifcfg-sit3

```
DEVICE="sit3"
BOOTPROTO="none"
ONBOOT="yes"
IPV6INIT="yes"
IPV6TUNNELIPV4="64.71.128.26"
IPV6ADDR="3FFE:1200:3028:FF01::2e7/127"
```

ในไฟล์ /etc/sysconfig/static-routes-ipv6

```
sit3    3ffe::/16
sit3    2000::/3
```

ตรวจสอบได้ดังนี้

```
# ip tunnel
sit0: ipv6/ip  remote any  local any  ttl 64  nopmtudisc
sit3: ipv6/ip  remote 64.71.128.26  local any  ttl 64

# ifconfig sit3
sit3      Link encap:IPv6-in-IPv4
          inet6 addr: 3ffe:1200:3028:ff01::2e7/127 Scope:Global
          inet6 addr: fe80::3e9b:8e45/10 Scope:Link
          inet6 addr: fe80::c0a8:101/10 Scope:Link

# route -A inet6 | grep sit3
3ffe:1200:3028:ff01::2e6/127          ::                    UA    256   3        0 sit3
3ffe::/16                             ::                    U     1     0        0 sit3
2000::/3                              ::                    U     1     0        0 sit3
fe80::/10                             ::                    UA    256   0        0 sit3
ff00::/8                              ::                    UA    256   0        0 sit3
```

ตัวอย่างที่3 (6to4 relay tunnel router)

IPv4 แอดเดรสของ remote 6to4 relay router คือ 194.95.108.191

IPv4 แอดเดรสที่ใช้เฉพาะในเครือข่ายคือ 192.168.1.1

IPv4 แอดเดรสที่แบบโกลเบิ้ล คือ 62.155.142.69

ตารางเราติง 3ffe::/16 2000::/3

การติดตั้ง

ในไฟล์ /etc/sysconfig/network-scripts/more ifcfg-eth0

```
DEVICE="eth0"
BOOTPROTO="none"
IPADDR="192.168.1.1"
NETMASK="255.255.255.0"
ONBOOT="yes"
IPV6INIT="yes"
IPV6FORWARDING="yes"
IPV6TO4INIT="yes"
IPV6TO4_RELAY="194.95.108.191"
IPV6TO4_IPV4ADDR="62.155.142.69"
```

ในไฟล์ /etc/sysconfig/static-routes-ipv6

```
sit0    3ffe::/16
sit0    2000::/3
```

การตรวจสอบ

```
# ip tunnel
sit0: ipv6/ip  remote any  local any  ttl 64  nopmtudisc

# ifconfig sit0
sit0      Link encap:IPv6-in-IPv4
          inet6 addr: ::62.155.142.69/96 Scope:Compat
          inet6 addr: ::127.0.0.1/96 Scope:Unknown
          inet6 addr: 2002:3e9b:8e45::1/48 Scope:Global
```

inet6 addr: ::192.168.1.1/96 Scope:Compat

```
# route -A inet6 | grep sit0
::/96                   ::                  U    256  2      0 sit0
2002:3e9b:8e45::/48     ::                  UA   256  0      0 sit0
3ffe::/16               ::194.95.108.191    UG   1    40     0 sit0
2000::/3                ::194.95.108.191    UG   1    0      0 sit0
fe80::/10               ::                  UA   256  0      0 sit0
ff00::/8                ::                  UA   256  0      0 sit0
```

ภาคผนวก จ
รายชื่อเราเตอร์ที่สนับสนุน IPv6

Hardware

Cisco

เราเตอร์ทุกรุ่นใน Cisco 12000 Series ได้ติดตั้ง Cisco Internetworking Operating System (Cisco IOS) ในเวอร์ชั่น 12.0ST ซึ่งรองรับ IPv6

- Cisco 12008
- Cisco 12012
- Cisco 12016
- Cisco 12404
- Cisco 12406
- Cisco 12410
- Cisco 12416

http://www.cisco.com

Hitachi

เราเตอร์ทุกชนิดในรุ่น GR2000

- GR2000 – 6H
- GR2000 – 10H
- GR2000 – 20H
- GR2000 – 2S
- GR2000 – 4S

http://global.hitachi.com

Ericsson

- AXI462/1
- AXI462/3
- AXI462/5

http://www.tbit.dk

6Wind

- 6211 Model
- 6221 Model
- 6231 Model

http://www.6wind.com

Software

GNU Zebra

สนับสนุนโปรโตคอล

- BGP-4 และ BGP-4+
- RIPv1, RIPv2 และ RIPng
- OSPFv2 และ OSPFv3

สามารถติดตั้งบนระบบปฏิบัติการ

- GNU/Linux 2.0.X and 2.2.X
- FreeBSD 2.2.8
- FreeBSD 3.1
- FreeBSD 4.X
- NetBSD 1.4
- OpenBSD 2.4

http://www.zebra.org

GateD

สามารถติดตั้งบนระบบปฏิบัติการ

- FreeBSD
- NetBSD
- BSDi
- Solaris
- AIX

http://www.nexthop.com

บรรณานุกรม

[1] Marcus Goncalves and Kitty Niles: "*IPv6 Networks*" , McGraw-Hill 1998.

[2] Craig Hunt: "*TCP/IP Netwok Administration, Second Edition*", O'Reilly & Associates 1998.

[3] Andrew S. Tanenbaum: "*Computer Networks*", Prentice-Hall 1996.

[4] Christian Huitema: "*IPv6: The New Internet Protocol*", Prentice-Hall 1996.

[5] William Stallings: "*Network Security Essentials: Application and standards*", Prentice-Hall 1999.

[6] R. Atkinson: "*[RFC 1825] Security Architecture for Internet Protocol*" August 1995.

[7] R. Atkinson: "*[RFC 1826] IP Authentication Header*" August 1995.

[8] R. Atkinson: "*[RFC 1827] IP Encapsulating Security Payload (ESP)"* August 1995.

[9] R. Gilligan and E. Nordmark: "*[RFC 1933] Transition Mechanisms for IPv6 Host and Routers*" April 1996.

[10] J. MaCann: "*[RFC 1981] Path MTU Discovery for IP version 6*" August 1996.

[11] S. Deering and R. Hinden: "*[RFC 2460] Internet Protocol, Version 6 (IPv6) Specification*" December 1998.

[12] T. Narten, E. Nordmark and W. Simpson: "*[RFC 2461] Neighbor Discovery for IP Version 6 (IPv6)*" December 1998

[13] S. Thomson: "*[RFC 2462] IPv6 Stateless Autoconfiguration*" December 1998.

[14] A. Conta and S. Deering: "*[RFC 2463] Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*" December 1998.

[15] M. Crawford: "*[RFC 2464] Tranmission of IPv6 Packets over Ethernet Networks*" December 1998.

[16] A. Conta and S. Deering: "*[RFC 2473] Generic Packet Tunneling in IPv6 Specification*" December 1998.

[17] R. Hinden, B. Carpenter and L. Masinter: *"[RFC 2732] Format for Literal IPv6 Addresses in URL's*" December 1999.

[18] R. Gilligan and E. Nordmark: "*[RFC 2893] Transition Mechanisms for IPv6 Hosts and Routers*" August 2000.

[19] B. Carpenter and K. Moore: "*[Internet Draft] Connection of IPv6 Domains via IPv6 Clouds without Explicit Tunnels*" October 1999.

[20] M. Crawford: "*[Internet Draft] IPv6 Node Information Queries*" August 2000.

[21] S. Deering, B. Haberman, T. Jinmei, E. Nordmark, A. Onoe and B. Zill: "*[Internet Draft] IPv6 Scoped Address Architecture*" September 2001.

[22] R. Hinden and S. Deering: "*[Internet Draft] IP Version 6 Addressing Architecture*" March 2001.

[23] R. Draves: "*[Internet Draft] Default Address Selection for IPv6*" May 2001.