บทที่ 2

ความรู้พื้นฐาน

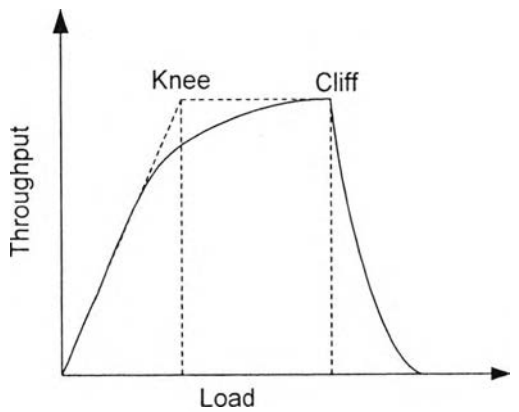
ในบทนี้จะได้กล่าวถึงความรู้พื้นฐานและทฤษฎีที่เกี่ยวข้องกับกรรมวิธีควบคุมความแออัด (congestion control mechanism) ของโพรโทคอล TCP ที่ได้มีผู้เสนอไว้ในบทความของงานวิจัยในอดีตที่ผ่านมา เนื้อหาหลัก ๆ ในบทนี้ประกอบด้วย เทคนิคการส่งข้อมูลบนช่องสัญญาณไร้สายบนโครงข่าย TCP แบบที่มีการส่งซ้ำบนระดับชั้นลิงก์ (link layer retransmission) หรือที่เรียกโดยย่อว่า โพรโทคอลสนูป (Snoop protocol) การนิยามคำจำกัดความของค่าความเท่าเทียมกันของโฟลว์ต่าง ๆ ที่ส่งผ่านภายในโครงข่าย และรวมไปถึงความสัมพันธ์ระหว่างตัวแปรต่าง ๆ ที่เกี่ยวข้องภายในระบบโดยเฉพาะในกรณีที่มีความแออัดเกิดขึ้นภายในโครงข่าย

2.1 ความแออัดในโครงข่าย

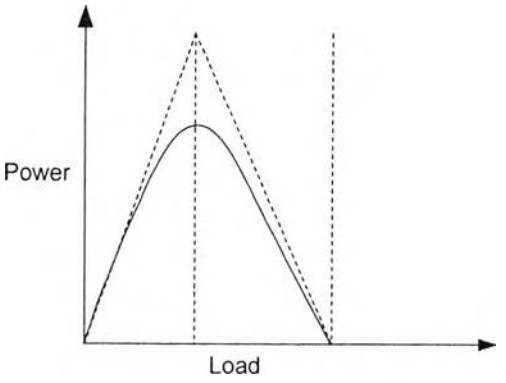
ความแออัดที่เกิดขึ้นภายในโครงข่ายโดยส่วนใหญ่มีสาเหตุมาจากการที่ผู้ใช้มีการส่งข้อมูลในบางช่วงเวลาผ่านโครงข่ายในปริมาณที่สูงเกินกว่าที่อุปกรณ์สื่อสารโดยเฉพาะเราเตอร์ (router) จะสามารถรองรับและประมวลผลได้ทัน เป็นเหตุให้เราเตอร์จำเป็นต้องนำข้อมูลส่วนที่เกินที่ไม่สามารถประมวลผลมาไว้ในหน่วยความจำหรือบัฟเฟอร์เพื่อรอการประมวลผลเพื่อส่งต่อไปยังเราเตอร์ตัวถัดไปในเส้นทาง ด้วยขนาดบัฟเฟอร์ที่มีอยู่จำกัดเมื่อมีข้อมูลส่วนเกินผ่านมายังเราเตอร์เป็นจำนวนมากจนบัฟเฟอร์ไม่เพียงพอต่อการจัดเก็บจึงจำเป็นต้องการละทิ้งข้อมูลบางส่วนที่กำลังผ่านเข้ามา ซึ่งส่งผลให้ประสิทธิภาพของโครงข่ายมีค่าลดลง ดังนั้นจึงต้องมีกลไกสำหรับคอยควบคุมความแออัดในโครงข่ายให้อยู่ในบริเวณที่เหมาะสม

ความสัมพันธ์ระหว่างทรูพุทของโครงข่ายซึ่งเป็นฟังก์ชันของอัตราการส่งข้อมูลเมื่อมีแนวโน้มเกิดความแออัดในโครงข่ายดังที่แสดงในรูปที่ 2.1 ขณะที่อัตราการส่งข้อมูลค่าน้อยๆ ความสัมพันธ์ระหว่างทรูพุทและอัตราการส่งเป็นแบบเชิงเส้นในลักษณะแปรผันซึ่งกันและกัน ถ้าเพิ่มอัตราการส่งขึ้นจนกระทั่งมีค่าเท่ากับขนาดของแบนด์วิดท์ ขนาดของทรูพุทจะไม่เพิ่มไปมากกว่านี้และเราเตอร์เริ่มจัดเก็บข้อมูลไว้ในบัฟเฟอร์ในระบบของแถวคอย ถ้าอัตราการส่งแพ็กเกตที่เข้ามายังโครงข่ายมีอัตราเพิ่มมากขึ้นและขนาดทรูพุทจะมีค่าสูงสุดเท่ากับขนาดแบนด์วิดท์ของลิงค์ขาออกซึ่งเป็นลักษณะเฉพาะที่เกิดขึ้นตรงบริเวณที่เรียกว่า "knee" จากจุดนี้หากเพิ่มอัตราการส่งจนกระทั่งบัฟเฟอร์ไม่เพียงพอสำหรับการจัดเก็บข้อมูลที่ผ่านมาได้มากกว่านี้และจะเริ่มละทิ้งข้อมูลที่กำลังผ่านเข้ามาทั้งหมดจนกว่าบัฟเฟอร์จะมีพื้นที่ว่างจากประมวลผลข้อมูลที่คงค้างในบัฟเฟอร์ ซึ่งส่งผลให้อัตราส่วนของทรูพุทต่ออัตราการส่งลดลงอย่างกระทันหันตรงบริเวณที่

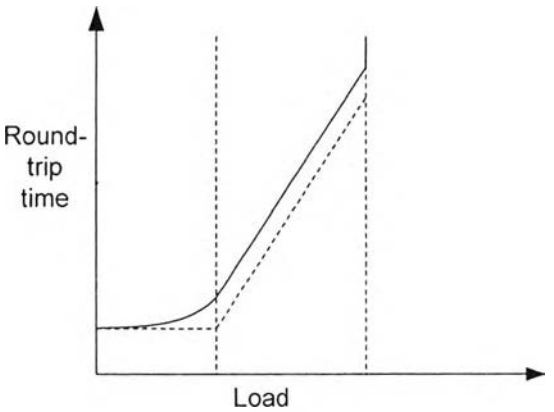
เรียกว่า "cliff" ซึ่งเป็นจุดเริ่มต้นของการพังทลายของความแออัดในโครงข่าย (congestion collapse) [8] เช่นเดียวกันกับค่าประวิงเวลาจริงในการเข้าถึงข้อมูล และค่ากำลังงานที่ได้ (power) มีลักษณะดังรูปที่ 2.2 และรูปที่ 2.3 ตามลำดับ ในช่วงต้นๆ ที่อัตราการส่งยังมีค่าน้อยๆ และเราเตอร์ยังสามารถที่ประมวลผลข้อมูลที่ผ่านเข้ามาเพื่อส่งต่อโดยที่ยังไม่จำเป็นต้องเก็บข้อมูลในบัฟเฟอร์ ค่าประวิงเวลาจริงในการเข้าถึงข้อมูลจะมีค่าคงที่ ส่วนค่ากำลังงานมีอัตราการเพิ่มขึ้นอย่างเห็นเชิงเส้น จนกระทั่งอัตราการส่งมีค่าเพิ่มขึ้นจนเราเตอร์เริ่มจัดเก็บข้อมูลที่ผ่านเข้ามาในบัฟเฟอร์เมื่อเข้าสู่จุดที่เรียกว่า "knee" ค่าประวิงเวลาจริงในการเข้าถึงข้อมูลจะมีการเพิ่มขึ้นแบบเชิงเส้นจนกว่าจะเกิดการพังทลายของความแออัด ส่วนค่ากำลังงานที่ได้ออกมาในตอนที่โครงข่ายจะเข้าถึงจุด "knee" ที่อัตราการส่งข้อมูลยังมีค่าน้อยจะมีอัตราการเพิ่มขึ้นอย่างเป็นเชิงเส้นเมื่อเทียบกับอัตราการส่งจนกระทั่งโครงข่ายเข้ามาสู่จุดที่เรียกว่า "knee" ค่ากำลังงานที่ได้ออกมาจะมีค่าคงที่ เพราะว่าแพ็กเกตส่วนใหญ่ของโครงข่ายถูกจัดเก็บไว้บนบัฟเฟอร์เพื่อรอการส่งออกจากระบบแถวคอย จึงส่งผลให้อัตราส่วนของกำลังงานที่ได้ต่ออัตราการส่งมีค่าลดลง ดังนั้นบริเวณที่เหมาะสมสำหรับการทำงานของกลไกการควบคุมความแออัดเพื่อให้สภาวะของโครงข่ายมีประสิทธิภาพและกำลังงานที่สูงสุดควรจะทำงานบริเวณจุด "knee" ก่อนที่มีอัตราการส่งเพิ่มมากขึ้นจนเกิดการพังทลายของโครงข่าย



รูปที่ 2.1 ความสัมพันธ์ระหว่างทรูพุทและโหลด (อัตราการส่งข้อมูล)



รูปที่ 2.2 ความสัมพันธ์ระหว่างค่าประวิงเวลาจริงในการเข้าถึงข้อมูลและโหลด



รูปที่ 2.3 ความสัมพันธ์ระหว่างกำลังงานและโหลด

2.2 หลักการทำงานและกรรมวิธีควบคุมความแออัดของ TCP รูปแบบต่าง ๆ

แนวความคิดหลักๆ ของการควบคุมความแออัดในโครงข่าย TCP คืออาศัยการป้อนกลับเมื่อข้อมูลไปถึงผู้ใช้ปลายทางที่เรียกแพ็กเกตตอบรับ (acknowledgement packet) และความต่อเนื่องของลำดับหมายเลขแพ็กเกตตอบรับไปควบคุมอัตราการส่งข้อมูล อัลกอริทึมในการปรับเปลี่ยนอัตราการส่งให้เหมาะสมกับสภาพของความแออัดในโครงข่ายคือ Additive increase/Multiplicative decrease (AIMD) ซึ่งจะมีตัวแปรสถานะ (state variable) ของแต่ละโฟลว์ที่เรียกว่าหน้าต่างความแออัด ใช้สำหรับจำกัดขนาดของข้อมูลหรือจำนวนแพ็กเกตที่ทางผู้ใช้ต้นทางสามารถส่งได้โดยไม่ก่อให้เกิดความแออัดแก่โครงข่าย และหน้าต่างการประกาศ (advertising window) เป็นส่วนที่บ่งบอกว่าจำนวนข้อมูลหรือแพ็กเกตที่ทางด้านผู้รับปลายทางสามารถรองรับได้ ดังนั้นในหัวข้อต่อไปจะกล่าวถึงรายละเอียดของเทคนิคต่างๆ ที่ใช้ในการควบคุมความแออัดของการส่งข้อมูลของโพรโทคอล TCP

2.2.1 Additive increase / Multiplicative decrease (AIMD)

ในเทคนิค Additive increase/Multiplicative decrease จะอาศัยสถานะของความแออัดในโครงข่ายเป็นตัวกำหนดขนาดหน้าต่างความแออัด และลดขนาดหน้าต่างความแออัดเมื่อโครงข่ายเกิดความแออัด ส่วนในกรณีที่เมื่อไม่มีความแออัดเกิดขึ้นในโครงข่ายจะทำการเพิ่มขนาดหน้าต่างความแออัด ซึ่งทางด้านผู้ใช้ต้นทางจะอาศัยแพ็กเกตตอบรับเป็นตัวบ่งบอกว่าเกิดความแออัดขึ้นในโครงข่าย เมื่อแพ็กเกตตอบรับส่งไม่ถึงมือผู้รับจนกระทั่งเกิดหมดเวลาการรอคอย (time out) ทางต้นทางจะสรุปทันทีว่าแพ็กเกตตัวนั้นสูญหายระหว่างทางเนื่องจากเกิดความแออัดในโครงข่ายและจะลดอัตราการส่งลง (เพราะในโครงข่ายแบบใช้สายสามารถสมมุติฐานได้ว่าแพ็กเกตที่สูญหายจากความผิดพลาดของการส่งมีน้อยมากเมื่อเทียบกับการสูญหายที่เกิดเนื่องมาจากความแออัดในโครงข่าย) ในแต่ละครั้งที่เกิดหมดเวลาการรอคอยขึ้น ทางด้านผู้ใช้ปลายทางจะลดขนาดหน้าต่างความแออัดลงครึ่งหนึ่งของค่าปัจจุบัน จึงเรียกวิธีในส่วนนี้ว่า multiplicative decrease ถึงแม้ว่าขนาดของหน้าต่างความแออัดถูกกำหนดในเทอมของไบต์ แต่เพื่อง่ายต่อความเข้าใจใน multiplication decrease เราจะพูดถึงขนาดของหน้าต่างความแออัดในเทอมของจำนวนแพ็กเกตแทน ดังนั้นเมื่อขนาดของหน้าต่างความแออัดถูกกำหนดให้มีค่าเท่ากับ 16 แพ็กเกต เมื่อเกิดการสูญหายของแพ็กเกตเกิดขึ้นขนาดของหน้าต่างความแออัดจะถูกลดขนาดลงเหลือ 8 แพ็กเกต ถ้าการสูญหายของแพ็กเกตเกิดขึ้นอย่างต่อเนื่อง ขนาดของหน้าต่างความแออัดจะถูกลดจนมีค่าเท่ากับ 4, 2 และ 1 ตามลำดับ โดยขนาดของหน้าต่างความแออัดที่น้อยที่สุดมีค่าเท่ากับ 1 แพ็กเกตหรือค่า maximum segment size (MSS)

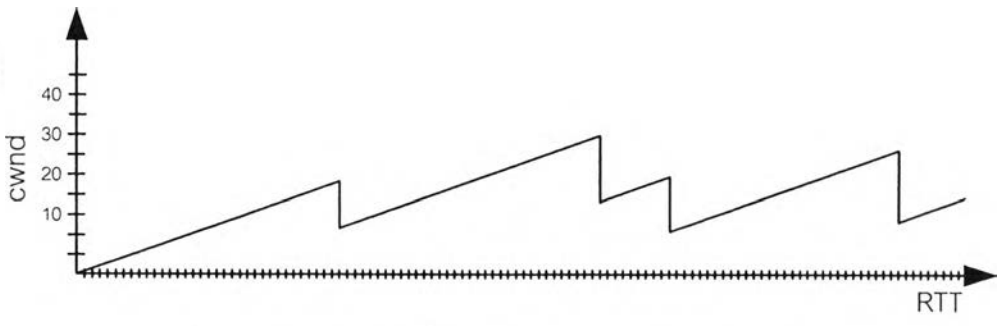
ในส่วนของ additive increase ทุกๆ ครั้งที่ต้นทางส่งแพ็กเกตในทุกๆรอบของหนึ่งช่วงค่าประวิงเวลาจริงในการเข้าถึงข้อมูลไปยังปลายทาง เมื่อได้รับแพ็กเกตตอบรับของแพ็กเกตทั้งหมดที่ส่งออกก่อนหน้านี้ไปเป็นที่เรียบร้อยจะทำการเพิ่มขนาดของหน้าต่างความแออัดด้วยค่าที่เทียบเท่า 1 แพ็กเกต ซึ่งอัตราการเพิ่มขนาดหน้าต่างความแออัดจะเป็นสัดส่วนกับขนาดหน้าต่างความแออัดเดิมทุกๆ ครั้งเมื่อได้รับแพ็กเกตตอบรับตามสมการ 2.1 และสมการ 2.2

$$Increment = mms * \left(\frac{mms}{cwnd} \right) \quad (2.1)$$

$$cwnd += Increment \quad (2.2)$$

ถ้าเรานำรูปแบบของการเพิ่มขึ้นและลดลงของขนาดหน้าต่างความแออัด ที่เกิดขึ้นอย่างต่อเนื่องตลอดต้นจนจบมาพล็อต โดยอยู่ในรูปฟังก์ชันของเวลาจะได้เป็นรูปแบบฟันปลา ดังที่แสดงในรูปที่ 2.4 ซึ่งหลักสำคัญของ additive increase/multiplicative decrease คือเมื่อความแออัดเกิดขึ้นในโครงข่ายทางด้านผู้ใช้ปลายทางสามารถที่จะลดอัตราการส่งหรือขนาดของหน้าต่างความแออัด ให้เร็วกว่าการเพิ่มอัตราการส่งหรือขนาดหน้าต่างความแออัด จึงเห็นได้ว่า

additive increase/ multiplicative decrease เป็นเงื่อนไขที่จำเป็นสำหรับการควบคุมความแออัดในโครงข่ายที่ต้องการให้ระบบมีเสถียรภาพ



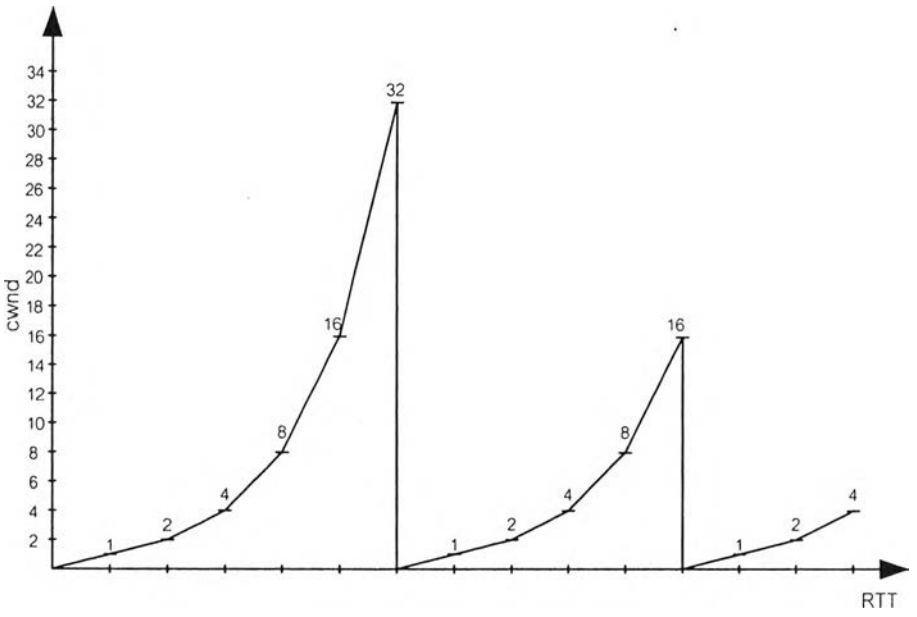
รูปที่ 2.4 รูปแบบการเพิ่มขึ้นและลดลงในการส่งข้อมูลด้วย AIMD

2.2.2 การเริ่มต้นอย่างช้าๆ (slow start)

การส่งข้อมูลหลังจากที่มีการลดขนาดหน้าต่างความแออัดหรือเมื่อตอนเริ่มต้นการส่งข้อมูล เทคนิค Additive increase/Multiplicative decrease ที่กล่าวไว้ก่อนหน้านี้เป็นวิธีที่เหมาะสมเมื่อผู้ใช้ต้นทางเริ่มต้นส่งข้อมูลด้วยขนาดของหน้าต่างความแออัดที่มากพอสมควรเพื่อให้อัตราการส่งข้อมูลลู่เข้าสู่จุดที่เหมาะสมกับสภาพโครงข่ายอย่างเร็วที่สุด ทั้งนี้เพราะว่าถ้าเริ่มต้นด้วยขนาดของหน้าต่างความแออัดที่มีค่าเท่ากับ 1 แพ็กเกต กว่าอัตราการส่งจะลู่เข้าสู่จุดที่เหมาะสมกับการส่งข้อมูลต้องใช้เวลานาน ดังนั้นจึงได้เสนอเทคนิคอีกวิธีหนึ่งที่เรียกว่าการเริ่มต้นอย่างช้า ๆ เพื่อใช้สำหรับเพิ่มขนาดของหน้าต่างความแออัดจากจุดเริ่มต้นอย่างรวดเร็ว และรูปแบบของการเพิ่มขนาดหน้าต่างความแออัดของเทคนิคเริ่มต้นอย่างช้า ๆ จะมีลักษณะคล้าย exponential มากกว่าเชิงเส้นดังรูปที่ 2.5 เนื่องจากอัตราการเพิ่มขึ้นของหน้าต่างความแออัดเป็นแบบทวีคูณทุกๆรอบของค่าประวิงเวลาจริงในการเข้าถึงข้อมูล

เมื่อผู้ใช้ต้นทางเริ่มต้นส่งข้อมูลจะกำหนดค่าเริ่มต้นของหน้าต่างความแออัดด้วยค่าที่เท่ากับ 1แพ็กเกต เมื่อได้รับแพ็กเกตตอบรับจะทำการเพิ่มขนาดของหน้าต่างความแออัดอีก 1 แพ็กเกต และทำการส่งข้อมูลออกไป 2 แพ็กเกต ตามขนาดของหน้าต่างความแออัดที่เพิ่มขึ้นมาในช่วงที่ใช้เทคนิคเริ่มต้นอย่างช้าๆ นี้จะเพิ่มขนาดของหน้าต่างความแออัดด้วยค่า 1 แพ็กเกตเสมอในทุกๆ ครั้งที่ได้รับแพ็กเกตตอบรับจากผู้ใช้ปลายทาง เมื่อเป็นเช่นนี้ขนาดของหน้าต่างความแออัดจะถูกเพิ่มจากข้างต้นเป็น 4 8 และ 16 แพ็กเกต ด้วยอัตราที่เพิ่มขึ้นเป็น 2 เท่าทุกๆ รอบของ RTT แต่เนื่องจากว่าการเพิ่มขนาดของหน้าต่างความแออัดด้วยอัตรา 2 เท่าทุกๆ RTT ตลอดการส่งข้อมูลตั้งแต่ต้นจนจบ ไม่เป็นผลดีแน่เพราะเป็นการเพิ่มขนาดอย่างรวดเร็วจะเป็นเหตุให้โครงข่ายเกิดความแออัดอย่างรวดเร็วตามไปด้วย ดังนั้นขีดจำกัดของการเริ่มต้นอย่างช้าๆ (slow start

threshold) ได้ถูกกำหนดขึ้นมาเพื่อการสำหรับเป็นจุดเปลี่ยนจากใช้เทคนิคการเริ่มต้นอย่างช้าๆ ไปยังเทคนิค additive increase ซึ่งมีอัตราการเพิ่มขนาดของหน้าต่างความแออัดที่ช้ากว่าเทคนิคการเริ่มต้นอย่างช้าๆ



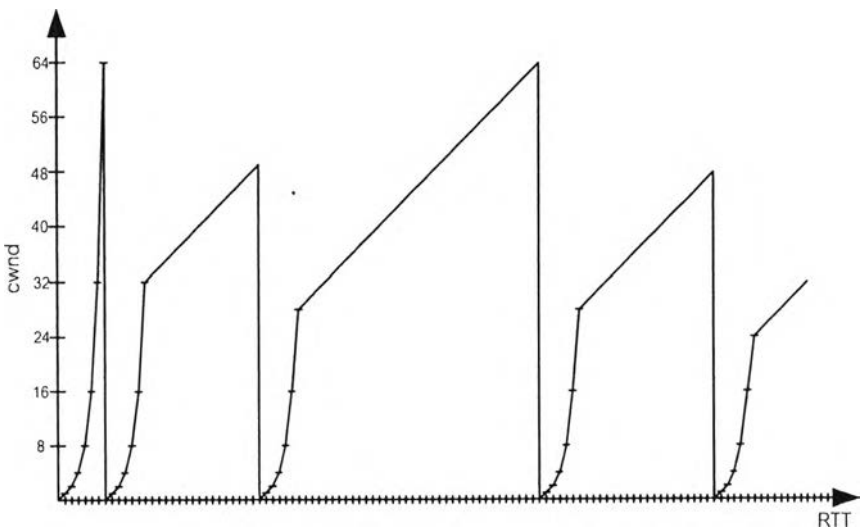
รูปที่ 2.5 รูปแบบการเพิ่มขึ้นของขนาดหน้าต่างความแออัดด้วยเทคนิคการเริ่มต้นอย่างช้าๆ

วิธีการกำหนดค่าขีดจำกัดของการเริ่มต้นอย่างช้าๆ จะอาศัยการส่งข้อมูลด้วยเทคนิคการเริ่มต้นอย่างช้าๆ ไปเรื่อยๆจนกว่าจะเกิดการชนกันของข้อมูลขึ้น และจะนำค่าของขนาดหน้าต่างความแออัด ในขณะนั้นมาลดลงครึ่งหนึ่งมาเป็นขนาดของขีดจำกัดของการเริ่มต้นอย่างช้าๆ พร้อมทั้งทำการรีเซตค่าหน้าตางความแออัดลงเหลือ 1 แพ็กเกต หลังจากนั้นก็จะทำการส่งด้วยเทคนิคการเริ่มต้นอย่างช้าๆ ต่อ จนกระทั่งขนาดของหน้าต่างความแออัด มีขนาดเท่ากับค่าของขีดจำกัดของการเริ่มต้นอย่างช้าๆ แล้วจะทำการสับเปลี่ยนเทคนิคการส่งข้อมูลต่อด้วย additive increase จนกว่าจะสิ้นสุดการส่งข้อมูลหรือเกิดการสูญหายของข้อมูล ซึ่งแสดงให้เห็นในรูปที่ 2.6

ซึ่งมีการใช้เทคนิคการเพิ่มขึ้นอย่างช้าๆ นี้จะใช้ใน 2 สถานการณ์คือ หนึ่ง ทุกๆ ครั้งที่มีการเริ่มต้นการส่งข้อมูลในครั้งแรกของการเชื่อมต่อ เนื่องจากว่าในช่วงเริ่มต้นของการส่งข้อมูลนี้ผู้ใช้ต้นทางไม่ทราบว่าอัตราการส่งข้อมูลที่โครงข่ายสามารถรองรับได้เพราะการส่งข้อมูลแบบ TCP นี้อาจมีการส่งข้อมูลผ่านบนสื่อที่หลากหลายตั้งแต่สายโทรศัพท์ไปถึงสายใยแก้วนำแสง ซึ่งมีความเร็วและความจุของแบนด์วิดท์ที่แตกต่างกันดังนั้นจึงไม่มีทางที่จะทราบขนาดแบนด์วิดท์ได้ ในสถานการณ์นี้เทคนิคการเพิ่มขึ้นอย่างช้าๆ จะมีการเพิ่มขนาดหน้าต่างความแออัด 2 เท่าทุกๆ RTT จนกระทั่งเกิดการสูญหายของแพ็กเกต ส่งผลให้ผู้ใช้ต้นทางหมดเวลาการรอคอยแพ็กเกต

ตอบรับและใช้เทคนิค multiplicative decrease ลดขนาดหน้าต่างความแออัดเหลือเท่ากับ 1 แพ็กเกต

ส่วนสถานการณ์ที่สองของการใช้เทคนิคการเริ่มต้นอย่างช้าๆ เกิดขึ้นเมื่อการเชื่อมต่อไม่ได้แพ็กเกตตอบรับ จากผู้ใช้ปลายทางจนกระทั่งเกิดหมดเวลาการรอคอย และเนื่องจากไม่มีแพ็กเกตตอบรับจึงเป็นเหตุให้ไม่มีสัญญาณที่ให้ผู้ใช้ต้นทางส่งข้อมูลใหม่ออกไปยังปลายทางได้ เมื่อเลยช่วงเวลาการอคอย ทางต้นทางจะลดขนาดของหน้าต่างความแออัดลงเหลือ 1 แพ็กเกต และเริ่มต้นการส่งข้อมูลใหม่อีกครั้งหนึ่งด้วยเทคนิคการเริ่มต้นอย่างช้าๆ แทนที่จะส่งข้อมูลด้วยขนาดของหน้าต่างความแออัดด้วยค่าเดิมทั้งหมดภายในครั้งเดียวอาจจะทำให้เกิดความแออัดในโครงข่ายขึ้นอีกได้



รูปที่ 2.6 ลักษณะการส่งข้อมูลด้วยเทคนิคการเริ่มต้นอย่างช้า ๆ

2.2.3 **การส่งซ้ำอย่างรวดเร็วและการกู้คืนอย่างรวดเร็ว** (fast retransmit and fast recovery)

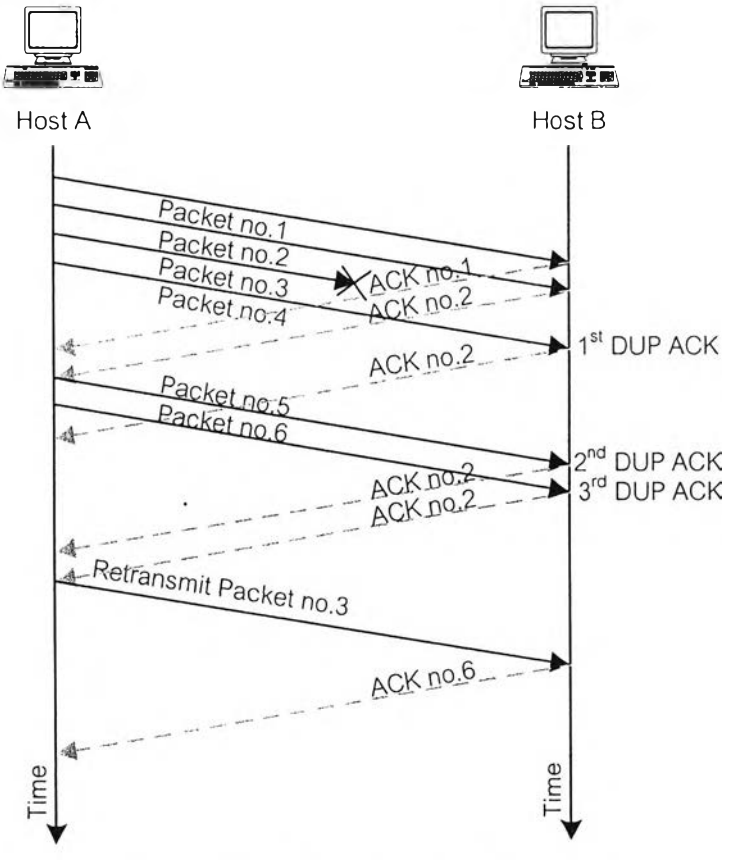
เนื่องจากว่าเทคนิคการควบคุมความแออัดที่กล่าวมาข้างต้นยังมีปัญหาในส่วนที่เมื่อเกิดการสูญหายของข้อมูลขึ้นต้องรอจนกระทั่งเกิดหมดเวลาการรอคอยทำให้เสียเวลามากถึงจะตัดสินได้ว่าเกิดการสูญหายของแพ็กเกตเนื่องจากการชนกันของข้อมูล จึงได้เกิดเทคนิคที่เรียกว่าการส่งซ้ำอย่างรวดเร็ว (fast retransmit) ขึ้นมาเพื่อให้มีการส่งซ้ำแพ็กเกตที่แน่ใจว่าสูญหายให้เร็วที่สุดก่อนที่จะเกิดหมดเวลาการรอคอยขึ้น ซึ่งแนวความคิดของการส่งซ้ำอย่างรวดเร็วนี้เป็นไปอย่างตรงไปตรงมา เพราะว่าทุกๆครั้งที่แพ็กเกตไปถึงมือผู้รับปลายทาง ทางด้านปลายทางจะส่งแพ็กเกตตอบรับที่บรรจุข้อมูลของลำดับหมายเลขแพ็กเกต (sequence number) ที่เพิ่งได้รับไปยังผู้ใช้ต้นทาง ดังนั้นเมื่อแพ็กเกตที่มีลำดับหมายเลขแพ็กเกต ถัดจากแพ็กเกตที่สูญหายถึงมือผู้รับ

ปลายทาง ทางผู้ใช้ปลายทางจะไม่ส่งแพ็กเกตเกตตอบรับที่มีลำดับหมายเลขของแพ็กเกตนี้ แต่จะส่งแพ็กเกตที่มีลำดับหมายเลขแพ็กเกตอันล่าสุดที่ได้ส่งไป ซึ่งแพ็กเกตที่ส่งไปครั้งนี้จะถูกเรียกว่า แพ็กเกตตอบรับสำเนา (duplicate acknowledgment) เมื่อทางด้านผู้ส่งได้รับแพ็กเกตตอบรับสำเนาจะรู้ว่าแพ็กเกตที่มีลำดับหมายเลขถัดมาจากที่แจ้งไปมาไม่ถึงมือผู้รับ ซึ่งอาจเป็นไปได้ว่าแพ็กเกตอาจจะตกค้างที่ไหนสักแห่งมากกว่าเกิดการสูญหาย ทางด้านผู้ส่งปลายทางจะรอจนกระทั่งได้รับแพ็กเกตตอบรับสำเนาที่มีลำดับหมายเลขเดียวกันครั้งที่ 3 จึงทำการส่งแพ็กเกตที่สูญหายไปอีกครั้ง ดังรูปที่ 2.7 แสดงให้เห็นว่าแพ็กตอบรับสำเนาทำให้เกิดการส่งซ้ำอย่างรวดเร็วได้อย่างไร เมื่อทางด้านผู้ใช้ปลายทางได้รับแพ็กเกตที่มีลำดับหมายเลข 1และ 2 แต่แพ็กเกตที่มีลำดับหมายเลข 3 เกิดการสูญหายในระหว่างทาง เมื่อแพ็กเกตลำดับหมายเลข 4 ส่งมาถึงมือผู้รับแล้วจะส่งแพ็กเกตตอบรับสำเนาด้วยลำดับหมายเลข 2 ออกไปแทนแพ็กเกตตอบรับที่มีลำดับหมายเลขแพ็กเกตของตัวเองและจะส่งแพ็กเกตตอบรับสำเนาออกไปอีกครั้งเมื่อแพ็กเกตที่มีลำดับหมายเลข 5 ไปถึงมือผู้รับปลายทาง เมื่อทางด้านต้นทางได้รับแพ็กเกตสำเนาหมายเลขเดียวกันครั้งที่ 3 จะเริ่มทำการส่งซ้ำแพ็กเกตที่สูญหายคือแพ็กเกตที่มีลำดับหมายเลข 3 เมื่อไปถึงยังปลายทาง ทางด้านผู้ใช้ปลายทางจะทำการส่งแพ็กเกตตอบรับที่มีลำดับหมายเลข 6 ซึ่งเป็นแพ็กเกตตอบรับอันล่าสุดก่อนที่จะมีการส่งซ้ำ จาก [2] เมื่อมีการเพิ่มเทคนิคการส่งซ้ำอย่างรวดเร็วได้ถูกปรับปรุงให้ดีขึ้นโดยลดระยะเวลาในการส่งแพ็กเกตที่สูญหายได้มากกว่าครึ่งซึ่งส่งผลให้ ทรูพุทมีค่าเพิ่มขึ้นจากเก่าประมาณ 20 เปอร์เซ็นต์

แต่อย่างไรก็ตามเมื่อเกิดการสูญหายของแพ็กเกตในขณะที่ขนาดของหน้าต่างความแออัดมีขนาดเล็กเกินไปทำให้ไม่มีแพ็กเกตตอบรับสำเนามากเพียงพอเพื่อแจ้งไปยังปลายทาง ส่วนมากแล้วเหตุการณ์อย่างนี้สามารถจะเกิดขึ้นได้ในช่วงที่ใช้เทคนิคการเริ่มต้นอย่างช้าๆ ในช่วงต้นๆของการเริ่มต้นส่งข้อมูล (ซึ่งในทางปฏิบัติเทคนิคการส่งซ้ำอย่างรวดเร็ว ยังคงทำงานได้เป็นปกติโดยสามารถรองรับการสูญหายของแพ็กเกตสูงสุดได้ 3 แพ็กเกต เมื่อมีขนาดหน้าต่างการประกาศเท่ากับ 64 แพ็กเกต) ซึ่งเมื่อเกิดเหตุการณ์เช่นนี้ขึ้นจะทำการหยุดส่งข้อมูลจนกระทั่งเกิดหมดเวลาการรอคอย และเริ่มทำการส่งข้อมูลด้วยเทคนิคการเริ่มต้นอย่างช้าๆออกไปใหม่อีกครั้งหนึ่ง

หลังจากเกิดการสูญหายของแพ็กเกตและผู้ใช้ต้นทางได้รับแพ็กเกตตอบรับสำเนาครั้งที่ 3 แล้ว เทคนิคของการส่งซ้ำอย่างรวดเร็วของโพรโทคอล TCP จะทำการลดขนาดของหน้าต่างความแออัดลงเหลือ 1 แพ็กเกต และเริ่มการส่งข้อมูลโดยใช้เทคนิคการเพิ่มขึ้นอย่างช้าๆ ซึ่งถ้ามีขนาดของหน้าต่างความแออัดที่สูงๆแล้ว การลดขนาดของหน้าต่างความแออัดลงเหลือ 1 แพ็กเกตนั้น จะลดทอนประสิทธิภาพในการส่งข้อมูล ดังนั้นจึงได้นำเอาเทคนิคการส่งข้อมูลที่เรียกว่าเทคนิคการกู้คืนอย่างรวดเร็ว (fast recovery) มาใช้ร่วมกับเทคนิคการส่งซ้ำอย่างรวดเร็ว โดยหลังจากที่ได้รับแพ็กเกตตอบรับสำเนาครั้งที่ 3 แทนที่จะลดขนาดของหน้าต่างความแออัดลงเหลือเพียง 1

แพ็กเกต เหมือนดังที่กล่าวไว้ข้างต้น แต่จะลดขนาดของหน้าต่างความแออัดลงเหลือครึ่งหนึ่งและเริ่มต้นการส่งข้อมูลต่อด้วยเทคนิค additive increase แทน ดังนั้นเทคนิคการเริ่มต้นอย่างรวดเร็วเมื่อนำมาใช้ร่วมกับการส่งซ้ำและการกู้คืนอย่างรวดเร็ว จะถูกใช้ก็ต่อเมื่อเริ่มต้นการส่งข้อมูลครั้งแรกและการส่งข้อมูลหลังจากการเกิดหมดเวลาการรอคอย



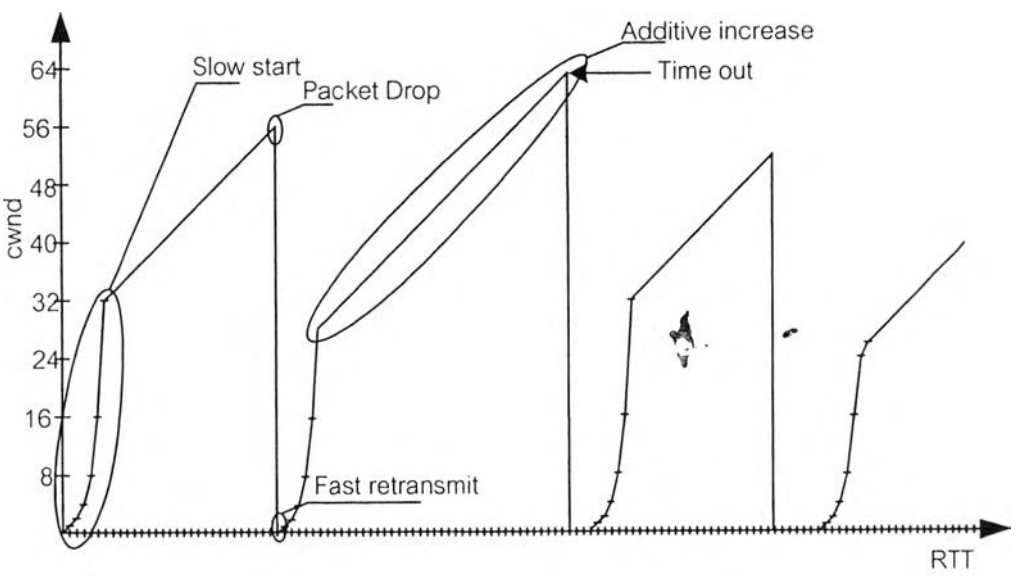
รูปที่ 2.7 การส่งแพ็กเกตด้วยเทคนิคการส่งซ้ำอย่างรวดเร็ว

2.3 รูปแบบของการส่งข้อมูลด้วย TCP ที่ใช้อยู่ในปัจจุบัน

จากที่กล่าวไว้ข้างต้นซึ่งเป็นส่วนของความรู้พื้นฐาน ความสัมพันธ์ระหว่างตัวแปรต่าง ๆ ที่เกี่ยวข้องภายในระบบ ความสำคัญของกลไกควบคุมความแออัด รวมทั้งทฤษฎีที่เกี่ยวข้องและหลักการที่สำคัญๆ ในการส่งข้อมูลบนโครงข่าย TCP ดังนั้นเนื้อหาในส่วนนี้จะกล่าวถึงกรรมวิธีควบคุมความของโพรโทคอล TCP รูปแบบต่าง ๆ ที่ได้มีผู้เสนอไว้ในบทความของงานวิจัยในอดีตที่ผ่านมาซึ่งประกอบไปด้วย Tahoe TCP และ Reno TCP และ SACK TCP

2.3.1 Tahoe TCP

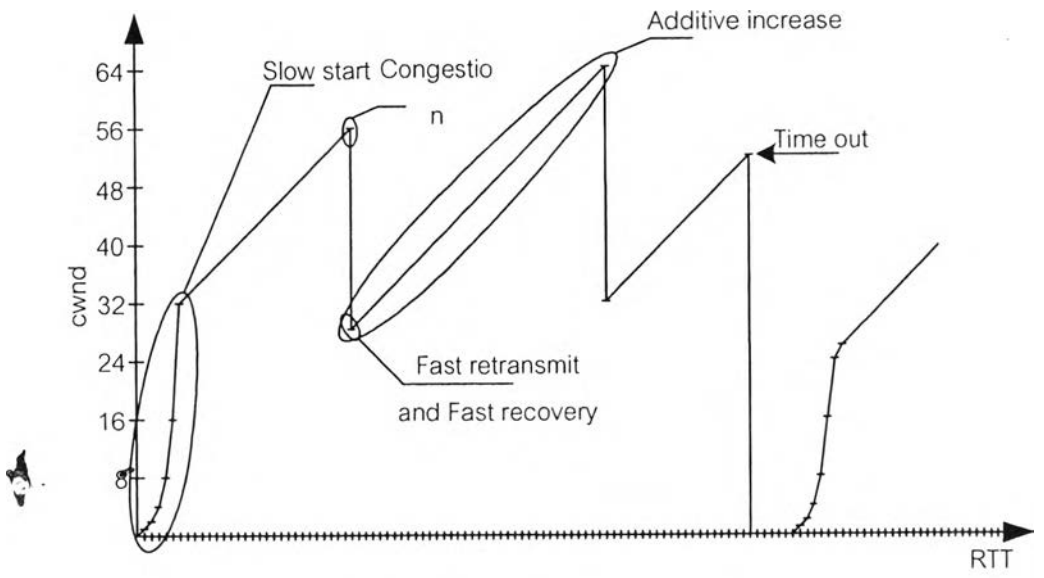
Tahoe TCP ถูกออกแบบโดย Jacobson ในปี ค.ศ. 1988 ซึ่งมีพื้นฐานมาจากการส่งข้อมูลแบบหน้าต่างเลื่อน (sliding window) โดยใช้เทคนิคการเริ่มต้นอย่างช้าๆ ทำงานร่วมกับเทคนิค additive increase/multiplicative decrease และเทคนิคการส่งซ้ำอย่างรวดเร็ว เมื่อมีการสูญหายของข้อมูลในการส่งข้อมูลบนโครงข่าย TCP จนเกินช่วงเวลาการรอคอยหรือในกรณีที่ได้รับแพ็กเกตตอบรับสำเนาครั้งที่ 3 ก่อนที่จะพ้นช่วงเวลาการรอคอย จะเริ่มต้นส่งข้อมูลใหม่ตั้งแต่ข้อมูลที่สูญหายด้วยเทคนิคการเริ่มต้นอย่างช้าๆ ซึ่งรวมถึงการคำนวณค่าใหม่ของขีดจำกัดของการเริ่มต้นอย่างช้าๆ และกำหนดให้ขนาดหน้าต่างความแออัดในการส่งใหม่นี้เท่ากับ 1 ดังรูปที่ 2.8 ซึ่งเป็นการแสดงลักษณะการทำงานของ Tahoe TCP ตั้งแต่เริ่มส่งแพ็กเกตจนกระทั่งหมดเวลาการรอคอยและเริ่มต้นส่งข้อมูลใหม่อีกครั้ง แต่ Tahoe TCP ยังมีข้อเสียในการส่งซ้ำข้อมูลใหม่อีกครั้งเมื่อเกิดการสูญหายหลายๆ แพ็กเกตใน 1 ช่วงของหน้าต่างข้อมูล



รูปที่ 2.8 รูปแบบการส่งข้อมูลด้วย Tahoe TCP

2.3.2 Reno TCP

Reno TCP ได้ถูกปรับปรุงและพัฒนาต่อจาก Tahoe TCP เพื่อที่แก้ไขวิธีในการกู้คืนแพ็กเกตที่สูญหายจากแพ็กเกตตอบรับสำเนาให้มีประสิทธิภาพยิ่งขึ้นโดยเพิ่มเทคนิคการกู้คืนอย่างรวดเร็วเข้ามา ซึ่งทำให้ผู้ใช้ต้นทางสามารถกู้คืนแพ็กเกตที่สูญหายด้วยขนาดของความแออัดใหม่เท่ากับครึ่งหนึ่งขนาดหน้าต่างความแออัดเดิมได้อย่างต่อเนื่อง เมื่อได้รับแพ็กเกตตอบรับสำเนาจากผู้ใช้ปลายทางก่อนที่จะหมดเวลาการรอคอยแทนที่จะกลับไปใช้เทคนิคการเริ่มต้นอย่างช้า ๆ ในการส่งข้อมูลใหม่อีกครั้ง ถึงแม้ว่าถูกปรับปรุงให้ดีกว่า Tahoe TCP โดยเฉพาะอย่างยิ่งเมื่อกล่าวถึงการกู้คืนแพ็กเกตในกรณีที่แพ็กเกตเกิดสูญหายเพียงแพ็กเกตเดียว ซึ่งรูปที่ 2.9 เป็นการแสดงการทำงานของโพรโทคอล Reno TCP ซึ่งแสดงให้เห็นถึงลักษณะการเพิ่มและลดลงของขนาดหน้าต่างความแออัดเมื่อเริ่มต้นการส่งแพ็กเกตและการกู้คืนแพ็กเกตเมื่อเกิดการสูญหาย แต่โพรโทคอล Reno TCP ยังคงตอบสนองต่อการสูญหายแบบหลายๆแพ็กเกตในหนึ่งช่วงค่าประวิงเวลาจริงในการเข้าถึงข้อมูลได้ไม่ดีเท่าที่ควร ข้อได้เปรียบในการปรับปรุงการส่งข้อมูลด้วยเทคนิคการกู้คืนอย่างรวดเร็วนี้เพื่อให้แบนด์วิดท์มาการใช้งานอยู่ตลอดเวลาโดยไม่ปล่อยให้สูญเสียโดยเปล่าประโยชน์ในขณะที่รอแพ็กเกตตอบรับสำเนา แต่อย่างไรก็ตามเมื่อเกิดแพ็กเกตสูญหายจนกระทั่งหมดเวลาการรอคอย Reno TCP ยังคงใช้เทคนิคการเริ่มต้นอย่างช้าๆ อยู่เหมือนเดิมด้วยขนาดหน้าต่างความแออัดที่มีค่าเท่ากับ 1 แพ็กเกตเช่นเดียวกับ Tahoe TCP



รูปที่ 2.9 รูปแบบการส่งข้อมูลด้วย Reno TCP

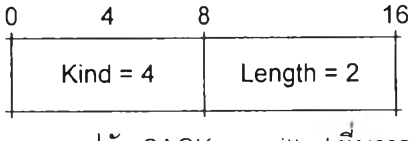
2.3.3 SACK TCP (Selective ACKnowledgment TCP)

SACK TCP (selective acknowledgement TCP) เป็นเทคนิคที่ถูกพัฒนาขึ้นเพื่อแก้ไขปัญหาเมื่อเกิดการสูญหายหลายๆแพ็กเกตใน 1 รอบของค่าประวิงเวลาจริงในการเข้าถึงข้อมูล เพราะว่ากรรมวิธีการส่งข้อมูลด้วย Tahoe TCP และ Reno TCP นั้น ใช้การส่งแพ็กเกตตอบรับแบบ cumulative เพื่อบ่งบอกให้ผู้ใช้ต้นทางรู้ว่าแพ็กเกตที่มีลำดับหมายเลขต่ำกว่าค่าในขอบซ้ายสุดของหน้าต่างทางด้านรับ (left edge of the receive window) ยังไม่ได้รับการตอบรับและผู้ใช้ต้นทางจะส่งซ้ำแพ็กเกตที่ไม่ได้รับการตอบรับด้วยเทคนิคการส่งซ้ำอย่างรวดเร็วและการกู้คืนอย่างรวดเร็ว แต่เทคนิคที่กล่าวมาสามารถกู้คืนแพ็กเกตที่สูญหายได้เพียง 1 แพ็กเกตต่อ 1 รอบของค่าประวิงเวลาจริงในการเข้าถึงข้อมูลเท่านั้น เมื่อมีส่งข้อมูลที่สูญหายและยังไม่ได้ทำการกู้คืนต้องรอจนกระทั่งหมดเวลาการรอคอย ถึงจะทำการส่งซ้ำแพ็กเกตที่สูญหายได้อีกครั้ง

โพรโทคอล SACK เป็นโพรโทคอลที่ถูกออกแบบมาเฉพาะเจาะจงเพื่อรับมือกับการสูญหายของแพ็กเกตจำนวนมาก ๆ โดยอาศัยเฮดเดอร์ของ TCP ในส่วนของออปชันฟิลด์อยู่สองส่วน คือ ส่วนของออปชัน SACK permitted ซึ่งเป็นออปชันชนิดที่ 4 และมีขนาด 2 ไบต์ และส่วนออปชัน SACK เป็นออปชัน ชนิดที่ 5 และมีขนาดที่สามารถเปลี่ยนแปลงได้ตามความเหมาะสม

2.3.3.1 ออปชัน SACK permitted

Sack permitted เป็นออปชันที่ถูกกำหนดค่าไปพร้อมกับแพ็กเกตซิงค์ (SYN) เพื่อบอกให้ผู้ใช้ปลายทางเปิดการทำงานของออปชัน SACK ที่อยู่ในฟิลด์ออปชันของไอพีดาทาแกรมและเปิดใช้การทำงานอัลกอริทึมของ SACK เมื่อการเชื่อมต่อเกิดขึ้นเสร็จสมบูรณ์จะไม่มีการกำหนดค่าออปชันนี้ในแพ็กเกตชนิดอื่นที่ไม่ใช่แพ็กเกตซิงค์ (SYN packet)

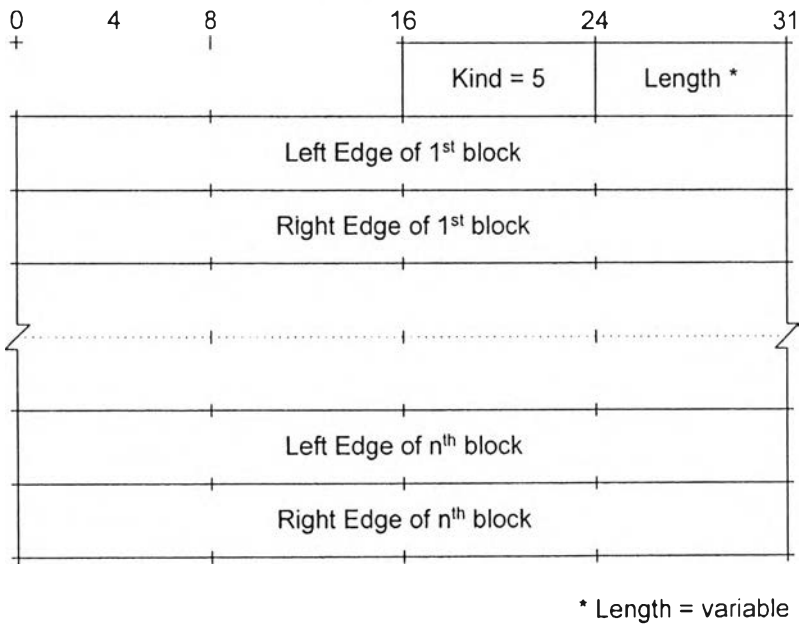


รูปที่ 2.10 รูปแบบของออปชัน SACK permitted ที่บรรจุอยู่ในฟิลด์ออปชัน

2.3.3.2 ออปชัน SACK

ออปชัน SACK เป็นส่วนที่บรรจุข้อมูลเพิ่มเติมของการตอบรับจากผู้ใช้ปลายทางไปยังผู้ใช้ต้นทางซึ่งอยู่ในรูปของบล็อกที่ไม่ต่อเนื่องหลายๆ บล็อกของชุดลำดับหมายเลขแพ็กเกตที่ได้รับการตอบรับจากผู้ใช้ปลายทางแล้ว ส่วนช่องว่างที่เกิดขึ้นระหว่างบล็อกคือชุดของลำดับหมายเลขแพ็กเกตที่เกิดสูญหายระหว่างทาง แต่ละบล็อกที่ไม่ต่อเนื่องนี้ถูกแบ่งเป็นขอบทางซ้ายสุดของบล็อก (left edge of block) และขอบทางขวาสุดของบล็อก (right edge of block) ซึ่งต่างใช้เนื้อที่

ส่วนละ 4 ไบต์ที่มีตัวแปรแบบ unsigned integer เพื่อบอกจำนวนไบต์ (ในที่นี่จะแทนด้วยหมายเลขแพ็กเกตแทนเพื่อความสะดวกในการเข้าใจและอ้างถึง) ในกรณีมีบล็อกเพียง 1 บล็อก ขอบซ้ายสุดของบล็อกและขอบขวาสุดของบล็อกจะระบุลำดับหมายเลขแพ็กเกตตัวแรกของบล็อกนั้นและลำดับหมายเลขแพ็กเกตตัวถัดไปของแพ็กเกตตัวล่าสุดในบล็อกนั้นตามลำดับ ซึ่งลำดับหมายเลขแพ็กเกตที่มีค่าน้อยกว่าค่าขอบซ้ายสุดของบล็อกนั้น (left edge of block - 1) คือลำดับหมายเลขแพ็กเกตที่สูญหายไป และในกรณีที่มีบล็อกมากกว่า 1 บล็อก ลำดับหมายเลขแพ็กเกตที่มีค่าเท่ากับค่าขอบขวาสุดของบล็อกนั้น คือลำดับหมายเลขแพ็กเกตที่สูญหาย โดยที่แพ็กเกตตอบรับที่ส่งมายังผู้ใช้ต้นทางนั้นจะระบุค่าของขอบซ้ายที่มีค่าต่ำที่สุดเป็นลำดับหมายเลขแพ็กเกต (ค่าขอบซ้ายของบล็อกอันดับสุดท้ายในกรณีที่เรียงลำดับบล็อกแบบ LILO: Last In Last Out)



รูปที่ 2.11 รูปแบบของออปชัน SACK ที่บรรจุอยู่ในฟิลด์ออปชัน

ในส่วนนี้จะกล่าวถึงการทำงานของออปชัน SACK โดยคร่าวๆเพื่อให้เห็นการทำงานเมื่อเกิดการสูญหายของแพ็กเกต ซึ่งเริ่มแสดงให้เห็นตั้งแต่แพ็กเกตลำดับหมายเลข 11 หรือแพ็กเกตไบต์ที่ 5000 และส่งข้อมูลออกไปอีกครั้งละ 8 แพ็กเกต โดยกำหนดให้ข้อมูลแต่ละแพ็กเกตมีขนาด 500 ไบต์

กรณีที่ 1 เมื่อสมมุติว่าแพ็กเกต 4 แพ็กเกตแรกได้รับแต่ข้อมูลที่เหลืออีก 4 แพ็กเกตสูญหาย

ในกรณีนี้ทางด้านผู้ใช้ต้นทางจะทำการส่งข้อมูลแบบปกติโดยเริ่มส่งแพ็กเกตซ้ำตั้งแต่ลำดับหมายเลข 15 หรือแพ็กเกตไบต์ที่ 7000 โดยไม่ได้ใช้ในส่วนการทำงานของออปชัน SACK

กรณีที่ 2 เมื่อสมมุติว่าแพ็กเกตตัวแรกสูญหายแต่ข้อมูลอีก 7 แพ็กเกตได้รับเป็นปกติ

ในกรณีนี้ทางด้านผู้ใช้ปลายทางจะเรียกร้องให้ทางผู้ใช้ต้นทางทำการส่งข้อมูลแพ็กเกตหมายเลข 11 หรือแพ็กเกตไบต์ที่ 5000 ดังนั้นในส่วนของออปชัน SACK จะมีเพียงแค่ 1 บล็อกเท่านั้นซึ่งมีข้อมูลที่บรรจุในขอบขวาสุดของบล็อก และขอบซ้ายของบล็อก ดังตาราง 2.1 จนกว่าผู้รับปลายทางได้รับข้อมูลที่สูญหายเป็นที่เรียบร้อย

ตารางที่ 2.1 รูปแบบของฟิลด์ออปชัน SACK ในกรณีเมื่อเกิดการสูญหายของแพ็กเกตลำดับหมายเลขที่ 11

Triggering Segment (Sequence No.)	ACK	First Block	
		Left Edge	Right Edge
5000 (11)	Lost		
5500 (12)	5000	5500	6000
6000 (13)	5000	5500	6500
6500 (14)	5000	5500	7000
7000 (15)	5000	5500	7500
7500 (16)	5000	5500	8000
8000 (17)	5000	5500	8500
8500 (18)	5000	5500	8500

กรณีที่ 3 เมื่อสมมุติว่าแพ็กเกตลำดับหมายเลข 12 14 16 และ 18 เกิดการสูญหาย

ในกรณีนี้ทางด้านผู้ใช้ปลายทางทำการตอบรับแพ็กเกตลำดับหมายเลขที่ 11 อย่างปกติจนเมื่อเกิดการสูญหายแพ็กเกตลำดับหมายเลขที่ 12 และผู้ใช้ปลายทางตอบรับแพ็กเกตลำดับหมายเลขที่ 13 พร้อมกับข้อมูลในส่วนออปชัน SACK ดังตารางที่ 2.2

สมมุติว่าผู้ใช้ต้นทางได้ทำการส่งแพ็กเกตลำดับหมายเลข 12 และ 14 ไปยังผู้ใช้ปลายทางซ้ำอีกครั้ง แต่ผู้ใช้ปลายทางได้รับแพ็กเกตลำดับหมายเลขที่ 14 เพียงอย่างเดียว (เนื่องจาก

แพ็กเกตลำดับหมายเลขที่ 12 อาจจะสูญหายระหว่างทางอีกครั้ง) ดังนั้นหลังจากที่ได้รับแพ็กเกตลำดับหมายเลขที่ 14 ผู้ใช้ปลายทางส่งแพ็กเกตตอบรับ พร้อมข้อมูลในส่วนของออปชัน SACK ที่ได้รับการปรับปรุงใหม่อีกครั้งดังตารางที่ 2.3 ซึ่งจากตารางดังกล่าวสังเกตเห็นได้ว่าบล็อกที่ 2 และบล็อกที่ 3 ถูกยุบมารวมเป็นบล็อกเดียวกัน

ตารางที่ 2.2 รูปแบบของฟิลด์ออปชัน SACK ในกรณีเมื่อแพ็กเกตลำดับหมายเลขที่ 12 14 16 และ 18 สูญหาย

Triggering Segment (Sequence No.)	ACK	First Block		2nd Block		3rd Block	
		Left Edge	Right Edge	Left Edge	Right Edge	Left Edge	Right Edge
5000(11)	5500						
5500(12)	(Lost)						
6000(13)	5500	6000	6500				
6500(14)	(Lost)						
7000(15)	5500	7000	7500	6000	6500		
7500(16)	(Lost)						
8000(17)	5500	8000	8500	7000	7500	6000	6500
8500(18)	(Lost)						

ตารางที่ 2.3 รูปแบบของฟิลด์ออปชัน SACK เมื่อได้รับแพ็กเกตตอบรับลำดับหมายเลขที่ 14 โดยที่แพ็กเกตลำดับหมายเลขที่ 12 เกิดการสูญหาย

Triggering Segment (Sequence No.)	ACK	First Block		2nd Block		3rd Block	
		Left Edge	Right Edge	Left Edge	Right Edge	Left Edge	Right Edge
6500(14)	5500	8000	8500	6000	7500		

หลังจากที่ผู้ใช้ปลายทางได้ส่งแพ็กเกตตอบรับของแพ็กเกตลำดับหมายเลขที่ 14 ไปแล้ว ทางด้านผู้ใช้ต้นทางส่งแพ็กเกตลำดับหมายเลขที่ 12 ซ้ำอีกครั้ง เมื่อถึงมือผู้รับปลายทางอย่างเป็น

ที่เรียบร้อย ดังนั้นผู้ใช้ปลายทางจึงส่งแพ็กเกตตอบรับไปยังผู้ใช้ต้นทางพร้อมด้วยข้อมูลในส่วนของออปชัน SACK ดังตารางที่ 2.4

ตารางที่2. 4 รูปแบบของฟิลด์ออปชัน SACK เมื่อได้รับแพ็กเกตตอบรับที่มีลำดับหมายเลขที่ 12 หลังจากที่ได้รับแพ็กเกตตอบรับลำดับที่หมายเลข 14 เป็นที่เรียบร้อย

Triggering Segment (Sequence No.)	ACK	First Block		2nd Block		3rd Block	
		Left Edge	Right Edge	Left Edge	Right Edge	Left Edge	Right Edge
5500(12)	7500	8000	8500				

หลังจากที่กู้คืนแพ็กเกตทั้งหมดได้อย่างเรียบร้อยแล้ว ผู้ใช้ต้นทางจะกลับไปใช้กรรมวิธีการส่งแบบปกติจนกว่าจะเกิดการสูญหายของแพ็กเกตเกิดขึ้นจึงจะมาใช้ กรรมวิธีการส่งข้อมูลแบบ SACK TCP อีกครั้งหนึ่ง

2.4 การประเมินความเท่าเทียม (fairness measure)

การประเมินความเท่าเทียมที่ใช้ในแวดวงการสื่อสารบนโครงข่ายคือการตรวจวัดการส่งข้อมูลผู้ใช้ว่าได้แบ่งกันใช้ทรัพยากรของโครงข่ายอย่างเท่าเทียมกันหรือไม่ ซึ่งเป็นสิ่งที่จำเป็นมากสำหรับการส่งข้อมูลแบบ TCP ที่วิ่งอยู่บนโครงข่ายมากกว่าโพรโทคอลอื่นเพราะว่าถ้าไม่สัดสรรให้ทุกๆ โฟลว์ใช้แบนด์วิดท์อย่างเท่าเทียมแล้ว โครงข่ายจะมีแนวโน้มเกิดปัญหาเช่นการพังทลายของโครงข่ายได้ง่าย วิธีสำหรับประเมินความเท่าเทียมของโครงข่ายที่ใช้กันอย่างแพร่หลายอยู่ 2 วิธีดังนี้

2.4.1 เทคนิคการประเมินความเท่าเทียมของ Jain (Jain's fairness)

Jain's fairness ถูกออกแบบโดย Raj Jain เพื่อใช้สำหรับประเมินความเท่าเทียมของเทคนิคการควบคุมความแออัดในโครงข่าย โดยมีสมมุติฐานว่าทุกโฟลว์มีอัตราการส่งที่เท่ากันและแบนด์วิดท์มีค่าเท่ากันทุกๆ เส้นทาง ดังสมการ

$$f(x_1, x_2, ..., x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n\sum_{i=1}^{n} x_i^2} \qquad (2.3)$$

โดยที่ $x_1, x_2, ..., x_n$ คือทรูพุทของผู้ใช้รายที่ n มีหน่วยเป็น $bits/second$ และค่าที่ได้จากการคำนวณด้วยสมการข้างต้นจะมีค่าอยู่ระหว่าง 0 และ 1 ถ้าค่าที่ได้จากการคำนวณมีค่าเท่ากับ 1 แสดงให้เห็นว่าโครงข่ายจัดสรรความเท่าเทียมให้แก่ผู้ใช้ทุก ๆ รายอย่างยุติธรรม

2.4.2 เทคนิคการประเมินความเท่าเทียมแบบสูงสุดและต่ำสุด (Max-min fairness)

ในกรณีนี้ใช้สำหรับพิจารณาในกรณีที่โครงข่ายมีผู้ใช้จำนวนหนึ่งต้องการแบนด์วิดท์น้อยกว่าค่าที่โครงข่ายจัดสรรไว้อย่างเท่าเทียม และผู้ใช้ที่เหลือต้องการแบนด์วิดท์มากกว่าค่าที่โครงข่ายสามารถจัดสรรไว้ โดยจะจัดสรรแบนด์วิดท์ให้แก่ผู้ใช้ในกลุ่มที่ต้องการน้อยกว่าที่โครงข่ายจัดสรรให้ตามที่ผู้ใช้ต้องการ และแบ่งแบนด์วิดท์ที่เหลือให้แก่ผู้ใช้อีกกลุ่มอย่างเท่าๆกัน จนกระทั่งจัดสรรแบนด์วิดท์จนเป็นที่พอใจทั้งหมดโดยไม่มีแบนด์วิดท์เหลือโดยไม่ได้ใช้ประโยชน์

2.5 โพรโทคอลสนูป (Snoop protocol)

โพรโทคอลสนูป คือ tcp-aware link layer ที่ถูกออกแบบมาเพื่อปรับปรุงและเพิ่มประสิทธิภาพการส่งข้อมูลบนโครงข่ายแบบใช้สายซึ่งมีฮอปสุดท้ายเชื่อมต่อต่ออยู่กับผู้ใช้ปลายทางที่มีช่องสัญญาณแบบไร้สาย โดยไม่ขัดแย้งกับหลักการของการส่งข้อมูลแบบจากต้นถึงปลาย โดยการติดตั้งโพรโทคอลสนูปบนสถานีฐานเพื่อคอยส่งซ้ำแพ็กเกตที่สูญหายบนช่องสัญญาณไร้สายด้วยหลักการเช่นเดียวกับโพรโทคอล TCP ที่ใช้ลำดับหมายเลขและแพ็กเกตตอบรับจากผู้ใช้ปลายทางและทำการคำนวณค่าค่าประวิงเวลาจริงในการเข้าถึงข้อมูลท้องถิ่น (locally last hop round trip time)

หลักการทำงานของโพรโทคอลสนูปในการส่งข้อมูลจากผู้ใช้ต้นทางไปยังผู้ใช้ปลายทางที่ช่องสัญญาณสุดท้ายก่อนโหนดปลายทางเป็นช่องสัญญาณแบบไร้สาย เมื่อมีการส่งข้อมูลจากผู้ใช้ต้นทางผ่านสถานีฐาน โพรโทคอลสนูปจะตรวจจับและจัดเก็บข้อมูลทุก ๆ แพ็กเกตที่วิ่งเข้ามายังสถานีฐาน หลังจากนั้นจะส่งผ่านแพ็กเกตที่ได้รับการจัดเก็บเรียบร้อยไปยังผู้ใช้ปลายทางต่อไป เช่นเดียวกันในส่วนของแพ็กเกตตอบรับโพรโทคอลสนูปจะทำการตรวจอ่านลำดับหมายเลขเมื่อตรวจจับการสูญหายโดยแพ็กเกตตอบรับสำเนาครั้งที่ 3 ถ้าแพ็กเกตที่สูญหายนั้นตรวจพบว่ายังถูกจัดเก็บอยู่ในบัฟเฟอร์ โพรโทคอลสนูปจะจัดการส่งแพ็กเกตที่สูญหายซ้ำอีกครั้ง เมื่อทำการส่งแพ็กเกตที่สูญหายซ้ำไปแล้วจะทำการลบแพ็กเกตตอบรับสำเนาฉบับนั้นทิ้งไป เพื่อป้องกันผู้ใช้ต้นทางส่งแพ็กเกตซ้ำซ้อนอย่างเกินความจำเป็น ซึ่งสามารถแบ่งการทำงานหลักของโพรโทคอลสนูปออกเป็น 2 ส่วนใหญ่ๆ คือ ส่วนที่จัดการกับแพ็กเกตที่เรียกว่า สนูปดาตา และส่วนที่คอยจัดการกับแพ็กเกตตอบรับที่เรียกว่า สนูปเอค

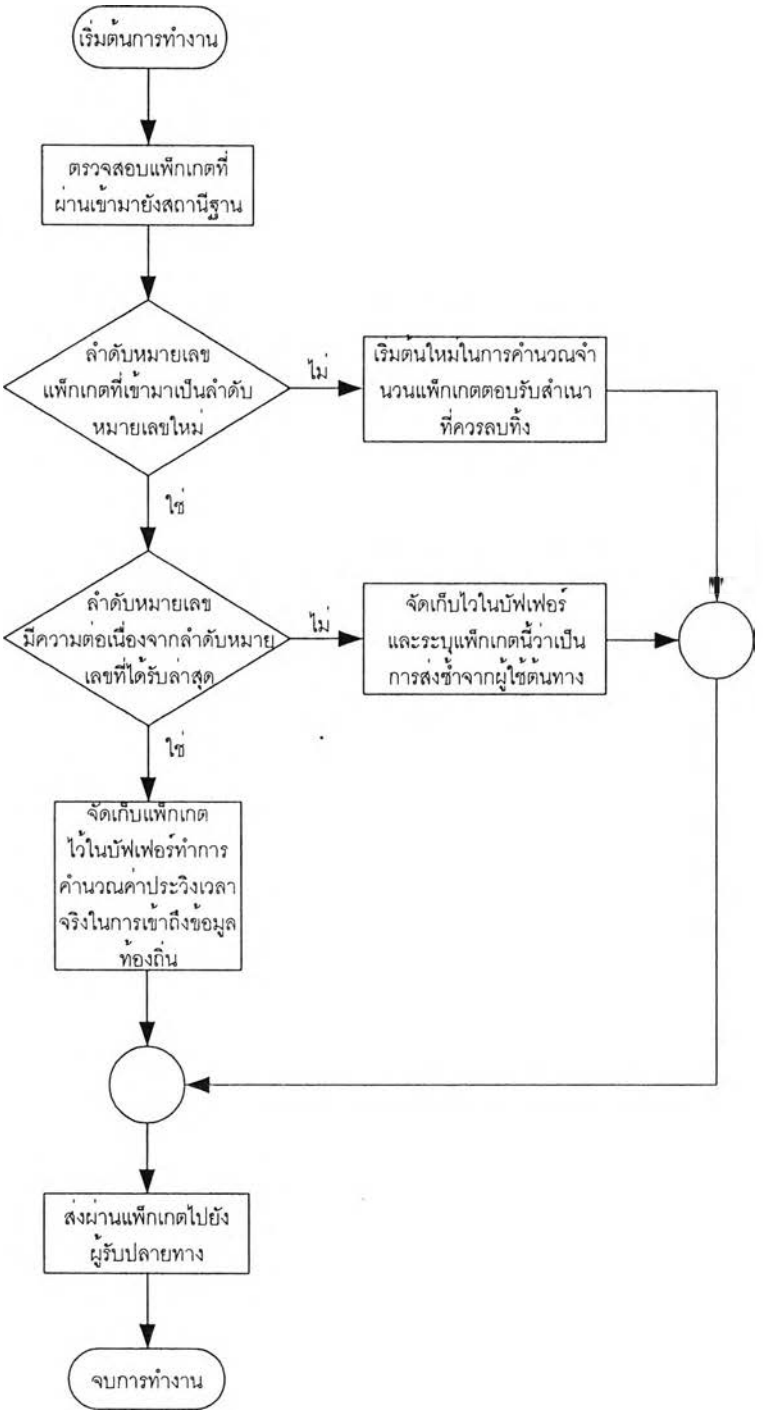
2.5.1 **สนูปดาตา** (Snoop Data)

สนูปดาตาเป็นส่วนที่ควบคุมและจัดการข้อมูลในส่วนของแพ็กเกตขาเข้า ซึ่งจะมีหน้าที่ในการจัดเก็บแพ็กเกตในหน่วยความจำ การจัดการแพ็กเกตโดยขึ้นอยู่กับลำดับหมายเลขแพ็กเกตตอบรับอันล่าสุด ซึ่งมีรายละเอียดเงื่อนไขและขั้นตอนดังต่อไปนี้

1. กรณีที่ลำดับหมายเลขแพ็กเกตเรียงลำดับต่อจากหมายเลขแพ็กเกตอันล่าสุดที่ถูกจัดเก็บไว้ในบัฟเฟอร์ของสนูป โพรโทคอลสนูปจะจัดเก็บแพ็กเกตและทำการคำนวณค่าเวลาจริงในการเข้าถึงข้อมูลท้องถิ่นโดยการกำหนดค่า time stampลงในแพ็กเกตบางตัวอย่างน้อยในหนึ่งช่วงขนาดหน้าต่างความแออัด เพื่อส่งผ่านแพ็กเกตไปยังผู้ใช้ปลายทางต่อไป

2. กรณีที่ลำดับหมายเลขแพ็กเกตที่เข้ามาเป็นลำดับหมายเลขที่ถูกจัดเก็บไว้ในบัฟเฟอร์ของสนูปไว้แล้ว ซึ่งมีสาเหตุเนื่องจากผู้ใช้ต้นทางรอคอยแพ็กเกตตอบรับของลำดับหมายเลขนั้นจนกระทั่งหมดเวลาในการรอคอยและทำการส่งซ้ำแพ็กเกตไปยังผู้รับปลายทางอีกครั้ง ดังนั้นเมื่อแพ็กเกตที่ถูกส่งซ้ำมายังสถานีฐาน โพรโทคอลสนูปจะทำการระบุแพ็กเกตนี้ว่าเป็นแพ็กเกตที่ถูกส่งซ้ำโดยผู้ใช้ต้นทางพร้อมทั้งส่งต่อแพ็กเกตลำดับหมายเลขนี้ไปยังผู้ใช้ปลายทาง เนื่องจากแพ็กเกตที่ถูกส่งซ้ำมาจากผู้ใช้ต้นทางซึ่งจะมีการตรวจสอบสถานะของแพ็กเกตตัวนี้อยู่จึงไม่สามารถลบแพ็กเกตนี้และจะทำการรีเซตตัวนับจำนวนแพ็กเกตตอบรับสำเนาที่สมควรลบทิ้ง

3. กรณีที่ลำดับหมายเลขแพ็กเกตที่เข้ามาไม่มีความต่อเนื่อง จากลำดับหมายเลขแพ็กเกตอันล่าสุดที่ถูกจัดเก็บและแพ็กเกตไม่ได้ถูกจัดเก็บไว้ในบัฟเฟอร์ของสนูป เพราะว่าเป็นแพ็กเกตใหม่ เหตุการณ์ดังกล่าวเช่นนี้สามารถสรุปได้ทันทีว่าเป็นแพ็กเกตลำดับหมายเลขนี้เคยถูกส่งมายังสถานีฐานแล้วแต่เกิดการสูญหายเนื่องจากความแออัดภายในโครงข่ายและผู้ใช้ต้นทางรอคอยจนกระทั่งหมดเวลาการรอคอยและส่งแพ็กเกตที่มีลำดับหมายเลขเดียวกันนี้ซ้ำอีกครั้งหนึ่งจนผ่านมายังสถานีฐานโดยไม่เกิดการสูญหายอีกครั้ง ในกรณีจะทำการจัดเก็บแพ็กเกตลำดับหมายเลขนี้ไว้ในบัฟเฟอร์ของสนูปพร้อมทั้งระบุว่าเป็นการส่งและทำการส่งผ่านผู้ใช้ต้นทางเป็นอันดับต่อไป



รูปที่ 2.12 ขั้นตอนการทำงานของสนูปดาดา

2.5.2 **สนูปเอค** (Snoop ACK)

สนูปเอคเป็นส่วนที่ควบคุมและจัดการข้อมูลในส่วนของแพ็กเกตตอบรับที่ผ่านเข้ามา ซึ่งจะมีหน้าที่ในการจัดการแพ็กเกตตอบรับ และปฏิบัติการโดยขึ้นอยู่กับลำดับหมายเลขแพ็กเกตตอบรับและชนิดของแพ็กเกตตอบรับ ตามเงื่อนไขดังต่อไปนี้

1. กรณีที่ลำดับหมายเลขแพ็กเกตตอบรับที่ผ่านเข้ามาเป็นแพ็กเกตลำดับหมายเลขค่าใหม่ เมื่อโพรโทคอลสนูปได้รับแพ็กเกตที่มีลำดับหมายเลขใหม่ที่ถัดจากลำดับหมายเลขแพ็กเกตตอบรับอันล่าสุด เนื่องมาจากแพ็กเกตที่มีลำดับหมายเลขเดียวกันนี้ส่งไปถึงมือผู้รับและได้รับการตอบรับจากผู้ใช้ปลายทางทางโพรโทคอลสนูปจะทำการลบข้อมูลของแพ็กเกตที่มีลำดับหมายเลขเดียวกันออกจากบัฟเฟอร์ของสนูป และทำการคำนวณค่าประวิงเวลาจริงในการเข้าถึงข้อมูลท้องถิ่นโดยใช้ค่าจาก time stamp ที่เพิ่มลงไปในส่วนของแพ็กเกตก่อนหน้านี้และส่งผ่านแพ็กเกตตอบรับนี้ไปยังผู้ใช้ต้นทาง

2. กรณีที่ลำดับหมายเลขของแพ็กเกตตอบรับมีค่าน้อยกว่าลำดับหมายเลขแพ็กเกตตอบรับอันล่าสุด ซึ่งเหตุการณ์ในกรณีเกิดขึ้นได้ค่อนข้างยากเพราะลำดับหมายเลขแพ็กเกตมีค่ามากกว่าหรือเท่ากับค่าของลำดับหมายเลขแพ็กเกตอันล่าสุดเท่านั้น ซึ่งในกรณีจะทำการลบแพ็กเกตตอบรับของลำดับหมายเลขนี้ไม่ให้ผ่านไปยังผู้ใช้ต้นทางและปล่อยให้โพรโทคอลยังคงทำงานอย่างปกติ

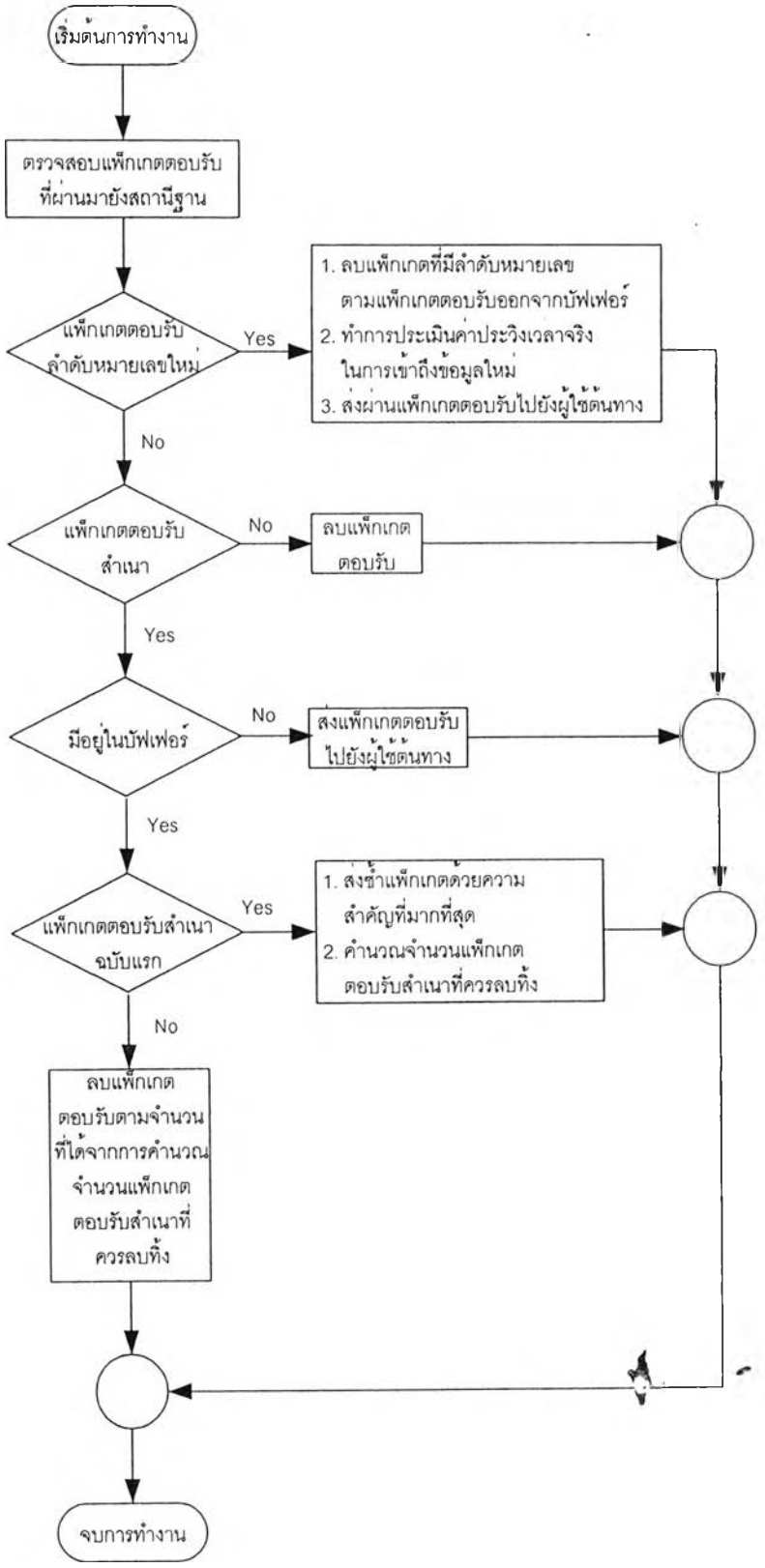
3. กรณีที่แพ็กเกตตอบรับที่ผ่านเข้ามาเป็นแพ็กเกตตอบรับลำเนา แพ็กเกตตอบรับสำเนาคือแพ็กเกตตอบรับที่มีลำดับหมายเลขเดียวกันและที่ได้รับติดต่อกันเป็นระยะเวลาหนึ่ง ซึ่งเป็นแพ็กเกตตอบรับที่ผู้ใช้ปลายทางร้องขอให้ส่งซ้ำแพ็กเก็ตที่มีลำดับหมายเลขตามที่ระบุไว้อีกครั้งเนื่องจากเกิดการสูญหาย เมื่อแพ็กเกตตอบรับประเภทนี้ผ่านเข้ามาโพรโทคอลสนูปแบ่งการทำงานออกเป็น 3 กรณีย่อยๆ ขึ้นอยู่กับสถานะของโพรโทคอลสนูปดังนี้คือ

3.1 กรณีแรกเกิดขึ้นเมื่อแพ็กเกตตอบรับสำเนาของลำดับหมายเลขที่ร้องขอไม่ได้ถูกจัดเก็บไว้ในบัฟเฟอร์ของสนูปหรือแพ็กเกตนั้นถูกระบุไว้ว่าเป็นการส่งซ้ำโดยผู้ใช้ต้นทางอย่างใดอย่างหนึ่ง ทั้งสองกรณีโพรโทคอลสนูปส่งผ่านแพ็กเกตตอบรับสำเนาทั้งหมดจะถูกส่งไปยังผู้ใช้ต้นทาง โดยกรณีที่แพ็กเกตตอบรับสำเนาร้องขอแพ็กเกตที่ไม่ได้ถูกจัดเก็บในบัฟเฟอร์ ซึ่งสรุปได้ทันทีว่าแพ็กเกตลำดับหมายเลขนั้นเกิดการสูญหายก่อนที่จะถึงสถานีฐาน และแพ็กเกตตอบรับสำเนาที่ถูกส่งไปจะเป็นการร้องขอให้ผู้ใช้ต้นทางส่งซ้ำแพ็กเกตตามลำดับหมายเลขที่ระบุอีกครั้ง แต่ในกรณีที่แพ็กเกตถูกระบุไว้ว่าเป็นการส่งซ้ำจากผู้ใช้ต้นทาง เพราะผู้ใช้ต้นทางนั้นมีการตรวจสอบสถานะของแพ็กเกตตอบรับสำเนาเมื่อมีการส่งซ้ำ ดังนั้นจึงไม่สามารถลบทิ้งและต้องส่งผ่านแพ็กเกตตอบรับนี้ไปยังผู้ใช้ต้นทาง

3.2 กรณีที่สองเกิดขึ้นเมื่อโพรโทคอลสนูปได้รับแพ็กเกตตอบรับสำเนาครั้งแรกและเพ็กเกตลำดับหมายเลขดังกล่าวถูกจัดเก็บไว้ในบัฟเฟอร์ของสนูป เมื่อโพรโทคอลสนูปได้รับแพ็กเกตตอบรับสำเนาฉบับแรก ซึ่งบางครั้งแพ็กเกตที่ถูกระบุว่าสูญหายตามแพ็กเกตตอบรับสำเนาอาจจะมาถึงผู้ใช้ปลายทางช้ากว่าแพ็กเกตอื่น ๆ ด้วยสาเหตุบางประการเช่นเกิดการสูญหายอันเนื่องมาจากช่องสัญญาณไร้สาย ดังนั้นแพ็กเกตที่วิ่งไปถึงผู้ใช้ปลายทางก่อนหน้าจะทำเกิดแพ็กเกตตอบรับสำเนาเป็นจำนวนหนึ่งถูกส่งไปยังโพรโทคอลสนูปและส่งผ่านไปยังผู้ใช้ต้นทางเพื่อให้ผู้ใช้ต้นทางได้รับแพ็กเกตตอบรับสำเนาเฉพาะแพ็กเกตที่ไม่ได้อยู่ในบัฟเฟอร์ของสนูป ดังนั้นโพรโทคอลสนูปจะทำการประเมินจำนวนแพ็กเกตตอบรับสำเนาที่ควรลบทิ้งก่อนส่งผ่านไปยังผู้ใช้ต้นทางโดยไม่ส่งผลให้ผู้ใช้ต้นทางทำการส่งซ้ำแพ็กเกตด้วยตัวเองแทนโพรโทคอลสนูป (ซึ่งเป็นช่วงเวลาในขณะที่รอแพ็กเกตตอบรับลำดับหมายเลขใหม่หลังจากที่ส่งซ้ำแพ็กเกตตามที่ระบุในแพ็กเกตตอบรับสำเนาโดยโพรโทคอลสนูป) โดยการคำนวณจำนวนของแพ็กเกตที่ถูกส่งระหว่างช่วงเวลาหลังจากเกิดการสูญหายและก่อนที่จะมีการส่งแพ็กเกตซ้ำ

3.3 ในกรณีที่สามเป็นกรณีซึ่งเกิดหลังจากที่ได้รับแพ็กเกตตอบรับสำเนาครั้งแรกและเพ็กเกตลำดับหมายเลขดังกล่าวถูกจัดเก็บไว้ในบัฟเฟอร์ของสนูป เมื่อแพ็กเกตตอบรับสำเนาฉบับอื่นๆ ตามมาหลังจากที่ทำการส่งซ้ำแพ็กเกตตามที่ระบุโดยมีค่าความสำคัญมากกว่าแพ็กเกตส่วนอื่นโพรโทคอลสนูปจะทำการลบทิ้งแพ็กเกตเกตตอบรับสำเนาที่ตามมาทั้งหมดตามวิธีการคำนวณที่ได้กล่าวไว้ในหัวข้อที่ 3.2 ดังนั้นเมื่อแพ็กเกตตอบรับของเพ็กเกตที่ได้ทำการส่งซ้ำมาถึงยังสถานีฐานและผ่านไปยังผู้ใช้ต้นทาง ทางผู้ใช้ต้นทางจะเห็นว่าแพ็กเกตที่ส่งไปก่อนหน้าไปถึงผู้ใช้ต้นทางอย่างปกติโดยไม่มีการสูญหายเกิดขึ้น

โพรโทคอลสนูปจะทำการติดตามจำนวนแพ็กเกตตอบรับสำเนาและลบทิ้งตามค่าที่คำนวณทุกๆครั้งเมื่อได้รับแพ็กเกตตอบรับฉบับแรกในระหว่างการส่งซ้ำแพ็กเกตที่สูญหาย และทำการรีเซตค่าที่คำนวณได้นั้นให้เป็นศูนย์ถ้ามีการส่งมาอีกครั้งจากผู้ใช้ต้นทางตามกลไกของการหมดเวลาหรือกลไกการส่งซ้ำอย่างรวดเร็วจากผู้ใช้ต้นทาง ซึ่งการส่งซ้ำแพ็กเกตที่สูญหายในช่องสัญญาณไร้สายด้วยค่าความสำคัญที่มากกว่าแพ็กเกตอื่นๆ สามารถปรับปรุงประสิทธิภาพได้ทุกสภาพอัตราการสูญหายของช่องสัญญาณ ทำให้แพ็กเกตที่ถูกส่งซ้ำไปถึงผู้ใช้ปลายทางได้เร็วขึ้นและลดการเกิดแพ็กเกตตอบรับสำเนาส่งผลให้ทรูพุทมีประสิทธิผลที่ดีขึ้น



รูปที่ 2.13 ขั้นตอนการทำงานของสนูปเอค