บทที่ 3
การปรับปรุงและการทำแบบจำลองของอัลกอริทึมแบบปรับตัว

เนื่องจากว่าการสื่อสารพหุสื่อบนเครือข่ายอินเตอร์เน็ตนั้นไม่สามารถที่หลีกเลี่ยงปัญหาการสูญหายของแพ็กเก็ตได้ ซึ่งการสูญหายของแพ็กเก็ตข้อมูลเสียงเป็นสาเหตุสำคัญที่ทำให้คุณภาพเสียงลดลง ดังนั้นจึงจำเป็นที่จะต้องมีกลไกที่จะช่วยลดผลกระทบจากการสูญหายแพ็กเก็ต ซึ่งในที่นี้จะเรียกกลไกที่ทำหน้าที่นี้ว่า การควบคุมความผิดพลาด (Error Control) โดยเนื้อหาของบทนี้เริ่มต้นด้วยการกล่าวถึง การควบคุมความผิดพลาดแบบปรับตัว (Adaptive Error Control) ในหัวข้อที่ 3.1โดยจะกล่าวถึงเทคนิคการควบคุมความผิดพลาด Forward Error Correction (FEC) และประเภทของ FEC หัวข้อที่ 3.2 จะกล่าวถึง อัลกอริทึมที่ใช้ในการศึกษา ส่วนหัวข้อที่ 3.3 และ3.4 จะเป็นเนื้อหาเกี่ยวกับ การใช้เทคนิควิธีกำลังสองน้อยที่สุด (Least Square Method) มาประยุกต์ใช้ในอัลกอริทึมแบบปรับตัว CNR (Centre Network Research) และ การตรวจสอบความถูกต้องของแบบจำลองที่ใช้ในการศึกษาตามลำดับ

3.1 การควบคุมความผิดพลาดแบบปรับตัว

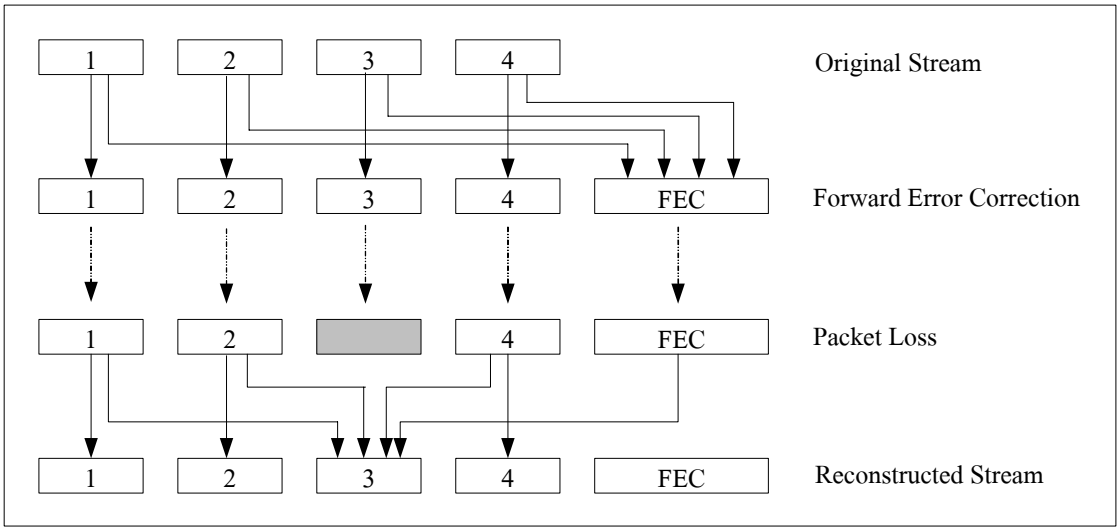
ในการสื่อสารเสียงบนเครือข่ายที่ไม่มีการรับประกันคุณภาพการบริการอย่างเช่นเครือข่ายอินเตอร์เน็ตนั้น ปัญหาอย่างหนึ่งที่ส่งผลกระทบต่อคุณภาพของเสียงก็คือการสูญหายของแพ็กเก็ต ถึงแม้ว่าในการสื่อสารเสียงนั้นไม่จำเป็นจะต้องได้รับข้อมูลครบทั้งหมดก็ตาม แต่การที่มีแพ็กเก็ตสูญหายมากเกินไปก็ทำให้คุณภาพเสียงต่ำเกินกว่าที่ยอมรับได้ ดังนั้นจึงจำเป็นต้องมีกลไกในการลดผลกระทบจากการสูญหายของแพ็กเก็ต ซึ่งกลไกดังกล่าวก็คือ การควบคุมความผิดพลาด ซึ่ง Forward Error Correction (FEC) ก็เป็นวิธีการควบคุมความผิดพลาดวิธีการหนึ่งที่สามารถลดผลกระทบจากการสูญหายของแพ็กเก็ตได้โดยไม่เพิ่มค่าเวลาหน่วงมากนัก[11] ซึ่งคุณสมบัตินี้เหมาะสมสำหรับการสื่อสารเสียงที่ต้องการความเป็นเวลาจริง

3.1.1 ประเภทคือ FEC

FEC สามารถแบ่งออกได้เป็น 2 ประเภท คือ

1. ประเภทที่ไม่ขึ้นกับสื่อ (Media Independent FEC)

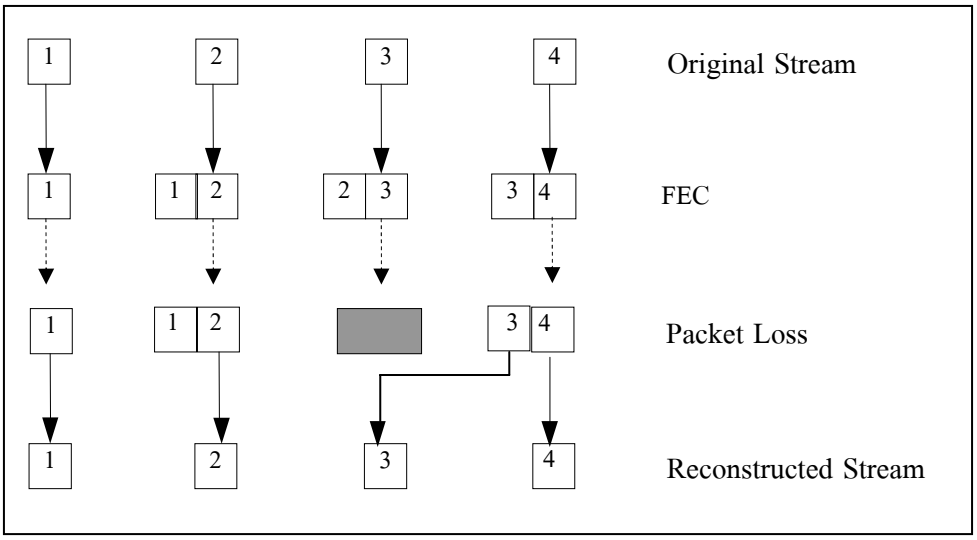
มีหลักการคือ เมื่อมีการส่งแพ็กเก็ตได้หนึ่งชุด จะมีการส่งแพ็กเก็ตพิเศษเพิ่มเข้าไป ดังรูปที่ 3.1 โดยแพ็กเก็ตพิเศษนี้ได้จากการกระทำทางคณิตศาสตร์ของข้อมูลจากทุกแพ็กเก็ตที่อยู่ในชุดเดียวกัน ดังนั้นเมื่อมีแพ็กเก็ตใดสูญหายไป ข้อมูลในแพ็กเก็ตที่สูญหายสามารถกู้คืนได้โดยใช้แพ็กเก็ตพิเศษนี้ แต่วิธีการนี้ใช้ไม่ได้ผลถ้ามีแพ็กเก็ตในชุดเดียวกันสูญหายมากกว่า 1 แพ็กเก็ต [12]



รูปที่ 3.1 Media Independent FEC

2. ประเภทที่ขึ้นกับสื่อ (Media Specific FEC)

มีหลักการคือ ในแต่ละแพ็กเก็ตที่ผู้ส่งจะส่งไปให้ผู้รับ จะมีการบรรจุข้อมูลซ้ำ (Redundancy) ของแพ็กเก็ตที่อยู่ก่อนหน้า แต่อาจใช้การบีบอัดด้วยอัตราบิตที่ต่ำกว่าเพื่อไม่ให้ปริมาณการใช้แบนด์วิดท์เพิ่มขึ้นมากเกินไป จากรูปที่ 3.2 จะเห็นว่าเมื่อแพ็กเก็ตที่ 3 สูญหาย ข้อมูลเสียงที่อยู่ในแพ็กเก็ตนี้สามารถกู้คืนได้จากข้อมูลซ้ำที่อยู่ในแพ็กเก็ตที่ 4 ซึ่ง FEC ประเภทที่สองนี้เป็นวิธีการที่ได้รับความนิยมมากกว่า FEC ประเภทแรก และมีมาตรฐานที่กำหนดรูปแบบของการบรรจุข้อมูลซ้ำของเสียงลงในแพ็กเก็ต RTP อีกด้วย ซึ่งมาตรฐานดังกล่าวก็คือ RFC2198[10] แต่มาตรฐานนี้ก็กำหนดเพียงแค่รูปแบบของแพ็กเก็ตเท่านั้น แต่ไม่ได้กำหนดว่าต้องใช้ปริมาณข้อมูลซ้ำเท่าใด และข้อมูลซ้ำแต่ละชุดใช้วิธีการบีบอัดเสียงแบบใด



รูปที่ 3.2 Media Specific FEC

เนื่องจากการควบคุมความผิดพลาดโดยใช้ FEC มีการส่งข้อมูลซ้ำของเสียงซึ่งทำให้ปริมาณการใช้แบนด์วิดท์ของสื่อสารเสียงเพิ่มขึ้น ดังนั้นจึงควรกำหนดปริมาณข้อมูลซ้ำให้เหมาะสมกับสภาพของเครือข่าย ในกรณีที่มีการสูญหายของแพ็กเก็ตเพียงเล็กน้อย หากกำหนดปริมาณข้อมูลซ้ำมากเกินไปจะเป็นการสิ้นเปลืองแบนด์วิดท์โดยไม่จำเป็น ดังนั้นในการสื่อสารเสียงจึงควรมีอัลกอริทึมในการกำหนดรูปแบบและปริมาณของข้อมูลซ้ำที่สามารถปรับตัวได้ตามปริมาณการสูญหายของแพ็กเก็ต ซึ่งอัลกอริทึมที่ได้มีการเสนอได้แก่ อัลกอริทึม Bolot[13], อัลกอริทึม RCCS[14] และอัลกอริทึม CNR[15] โดยทุกบทความนั้นมีส่วนที่เหมือนกันก็คือ อัลกอริทึมที่เสนอขึ้นนั้นล้วนมีพื้นฐานจาก FEC และจะมีการกำหนดรูปแบบการบรรจุข้อมูลมูลซ้ำลงในแพ็กเก็ตหรือที่เรียกว่า Combination เอาไว้ก่อน และจัดลำดับของแต่ละ Combination เอาไว้ โดย Combination ในลำดับหลังสามารถลดผลกระทบจากการสูญหายของแพ็กเก็ตได้ดีกว่า Combination ที่อยู่ลำดับแรกๆ แต่ต้องใช้ปริมาณข้อมูลซ้ำที่มากขึ้นด้วย โดยพารามิเตอร์ที่ใช้ในการตัดสินใจว่าควรจะใช้ Combination ใดก็คือปริมาณการสูญหายของข้อมูล สำหรับส่วนที่แตกต่างกันของแต่ละบทความก็คืออัลกอริทึมที่ใช้ในการเลือก Combination เมื่อใดควรที่จะเพิ่ม Combination และเมื่อใดควรที่จะลด เนื่องจากว่าอัลกอริทึมของการปรับตัวที่จัดอยู่ในประเภทนี้มีเป็นจำนวนมาก และแต่ละอัลกอริทึมก็ใช้ได้ผลในเงื่อนไขที่แตกต่างกัน วิทยานิพนธ์นี้จึงได้การศึกษาและปรับปรุงจากอัลกอริทึมที่มีอยู่แล้ว โดยเพิ่มเทคนิควิธีกำลังสองน้อยที่สุด (Least Square Method) เพื่อให้กำหนดพารามิเตอร์ที่เหมาะสมกับการสื่อสารเสียงในกรณีที่ใช้การบีบอัดแบบ G.723.1

3.1.2 อัตราการสูญหายของข้อมูลเสียงเมื่อมีการใช้ FEC

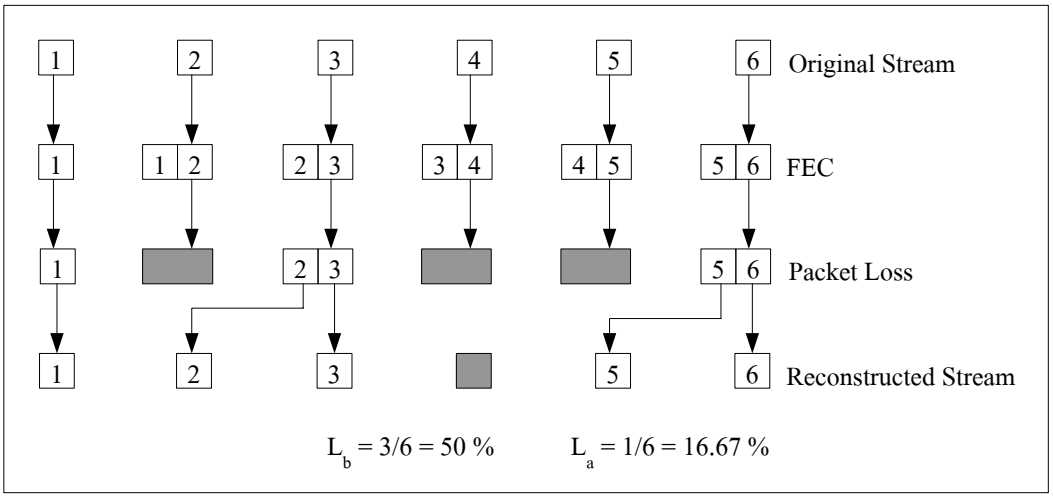
ก่อนจะกล่าวถึงอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวที่ใช้ในการศึกษา ในหัวข้อนี้จะอธิบายเกี่ยวกับค่าอัตราการสูญหายของข้อมูลเมื่อมีการใช้เทคนิค FEC ทั้งนี้เพื่อให้สามารถเข้าใจถึงรายละเอียดของอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวที่จะกล่าวถึงในหัวข้อถัดไปได้ง่ายขึ้น ซึ่งเมื่อมีการใช้เทคนิค FEC แล้ว ค่าอัตราการสูญหายของข้อมูลที่จะต้องพิจารณามีอยู่ 2 ค่าคือ

L_b คือ ค่าอัตราการสูญหายก่อนการจัดเรียง (Loss Rate before Reconstruction)

L_a คือ ค่าอัตราการสูญหายหลังการจัดเรียง (Loss Rate after Reconstruction)

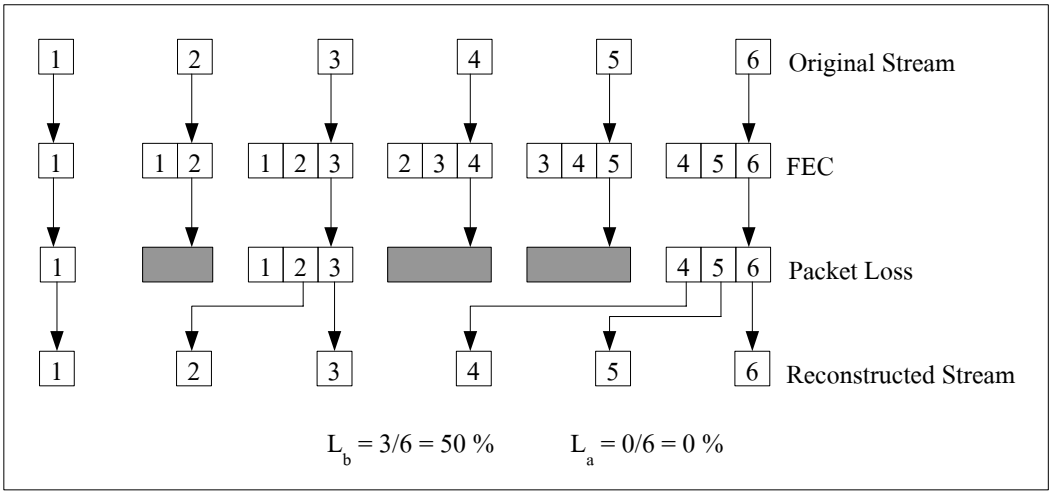
ซึ่งค่า L_b นั้นหมายถึงค่าอัตราการสูญหายของข้อมูลเสียงโดยที่ยังไม่ได้นำข้อมูลซ้ำมาจัดเรียงใหม่ หรืออาจจะกล่าวได้ว่าค่า L_b คืออัตราการสูญหายของแพ็กเก็ตนั่นเอง จากรูปที่ จะเห็นว่ามีการส่งแพ็กเก็ตเสียงทั้งหมด 6 แพ็กเก็ต แต่เกิดการสูญหาย 3 แพ็กเก็ต ดังนั้นค่า L_b จึงมีค่าเท่ากับ 3/6 หรือ 50 % นั่นเอง ส่วนค่า L_a จะคำนวณจากจำนวนข้อมูลเสียงที่สูญหายหลังจากที่ได้มีการจัดเรียงข้อมูลแล้ว ค่า L_a จึงมักจะมีค่าน้อยกว่า L_b เนื่องจากอาจมีการกู้คืนข้อมูลในแพ็กเก็ตที่

สูญหายบางส่วนได้จากข้อมูลซ้ำ จากรูปที่ 3.3 จะเห็นว่าเมื่อมีการจัดเรียงข้อมูลแล้วมีข้อมูลเสียงสูญหายไปเพียง 1 เฟรมเท่านั้นคือเฟรมที่ 4 ดังนั้นค่า L จึงมีค่าเท่ากับ 1/6 หรือ 16.67 % ซึ่งค่า L_a นี้เองจะเป็นตัวบอกถึงปริมาณข้อมูลเสียงที่สูญหาย ถ้า L_a มีค่ามากจะทำให้เสียงที่ได้รับมีคุณภาพต่ำ ส่วนค่า L_b เป็นค่าที่บอกถึงปริมาณการสูญหายของแพ็กเก็ตเท่านั้น



รูปที่ 3.3 การกู้คืนโดยใช้ข้อมูลซ้ำสามารถทำให้ค่า L_a น้อยกว่า L_b

การเพิ่มจำนวนชุดของข้อมูลซ้ำสามารถลดค่า L_a ได้ถึงแม้ว่าค่า L_b จะมีค่าเท่าเดิม จากรูปที่ 3.4 จะเห็นว่าเมื่อเพิ่มจำนวนข้อมูลซ้ำเป็น 3 ชุด สามารถกู้คืนข้อมูลเสียงที่สูญหายได้หมด ทำให้ค่า L_a เท่ากับ 0 % ดังนั้นในกรณีที่เครือข่ายมีความมีคับคั่งสูงและมีปริมาณแพ็กเก็ตสูญหายมากจึงควรจะมีเพิ่มจำนวนชุดข้อมูลซ้ำ เพื่อให้สามารถกู้ข้อมูลเสียงในแพ็กเก็ตที่สูญหายได้มากขึ้น และสามารถรักษาระดับของอัตราการสูญหายของข้อมูลเสียงไม่ให้สูงเกินไปได้



รูปที่ 3.4 การเพิ่มจำนวนข้อมูลซ้ำสามารถช่วยลดค่า L_a ได้

3.2 อัลกอริทึมแบบปรับตัวที่ใช้ในการศึกษา

3.2.1 อัลกอริทึม Bolot

อัลกอริทึมนี้ได้รับการเสนอโดย Bolot และได้รับการตีพิมพ์ลงใน[13] ซึ่งในบทความนี้ได้มีการกำหนดรูปแบบของการจัดวางข้อมูลซ้ำ (Combination) เอาไว้หลายแบบ และได้ทำการทดลองเพื่อหาค่า Reward ของแต่ละ Combination โดยค่า Reward (R_c) คืออัตราส่วนระหว่างค่า L_b และค่า L_a

$$R_c = L_b/L_a \tag{3.1}$$

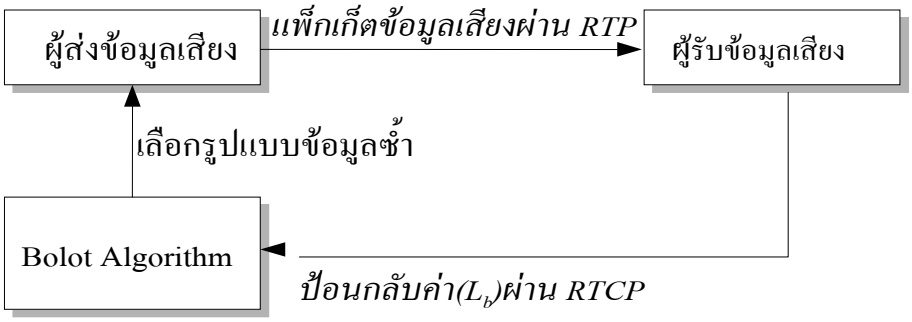
ตารางที่ 3.1 Combination ที่ใช้อัลกอริทึม Bolot

No.	Combination	Reward
0	(PCM)	1
1	(PCM, ADM4(1))	2.5
2	(PCM, GSM(1))	2.5
3	(PCM, LPC(1))	2.5
4	(PCM, ADM4(2))	6
5	(PCM, ADM4(1), ADM2(2))	6
6	(PCM, ADM4(1), ADM2(3))	10
7	(PCM, ADM4(1), ADM2(2), ADM2(3))	18

ค่า Reward ของแต่ละ Combination ที่ปรากฏใน ตารางที่ 3.1 เป็นค่าที่ได้จากการทดลองโดยใช้เครือข่ายระหว่างสถาบัน INRIA ในประเทศฝรั่งเศสกับ University College London (UCL) ในประเทศอังกฤษ สำหรับสัญลักษณ์ (PCM) หมายถึง ทุกแพ็กเก็ตใช้การบีบอัดแบบ PCM โดยไม่มีการบรรจุข้อมูลซ้ำ ส่วนสัญลักษณ์ (PCM, ADM4(1)) หมายถึง ในแต่ละแพ็กเก็ตลำดับที่ N ส่วนที่เป็นข้อมูลหลักใช้การบีบอัดแบบ PCM แต่ส่วนที่เป็นข้อมูลซ้ำใช้การบีบอัดแบบ ADM4 และเป็นข้อมูลซ้ำของแพ็กเก็ตที่ N-1 (ตัวเลข 1 ที่อยู่ในวงเล็บ) และสัญลักษณ์ของ Combination แบบอื่นๆ สามารถแปลความหมายได้ในทำนองเดียวกัน ซึ่งค่า Reward นี้เป็นค่าที่สามารถบอกได้ว่าแต่ละ

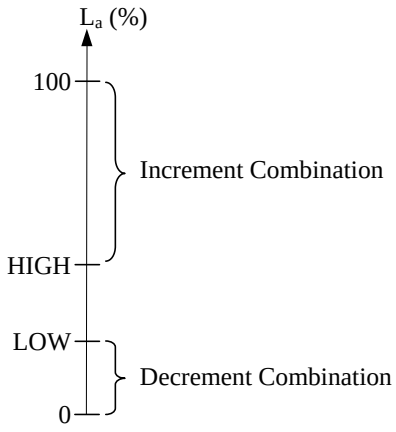
Combination สามารถกู้ข้อมูลเสียงในแพ็กเก็ตที่สูญหายได้ดีเพียงใด และจะเห็นว่า Reward จะเพิ่มขึ้นเมื่อมีการเพิ่มจำนวนชุดของข้อมูลซ้ำ ยกเว้นในกรณีของ Combination ที่ 4 และ 5 ซึ่งมีค่า Reward เท่ากัน และในกรณีของ Combination ที่มีการใช้จำนวนข้อมูลซ้ำเท่ากัน Combination ที่ใช้ข้อมูลซ้ำจากแพ็กเก็ตที่อยู่ห่างจากแพ็กเก็ตปัจจุบันมากกว่าจะมีค่า Reward สูงกว่า ตัวอย่างเช่น Combination ที่ 4 มีค่า Reward สูงกว่า Combination ที่ 3 แต่อย่างไรก็ตามค่า Reward ที่ได้แสดงในตารางที่ 3.1 นี้เป็นค่าที่ได้จากการทดลองเท่านั้น ซึ่งอาจจะไม่เป็นจริงเสมอไปในทุกสภาพแวดล้อม

จากแผนภาพการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม Bolot ในรูปที่ 3.5 จะเห็นว่าผู้รับจะมีการรายงานเฉพาะค่า L_b ให้ผู้ส่งได้ทราบเท่านั้นโดยจะระบุค่าของ L_b ลงฟิลด์ Fraction Lost ของแพ็กเก็ต RTCP ชนิด Receiver Report ซึ่งปกติฟิลด์ Fraction Lost นี้จะใช้ในการระบุค่าอัตราการสูญหายของแพ็กเก็ต และเนื่องจากค่า L_b มีค่าเท่ากับอัตราการสูญหายของแพ็กเก็ต จึงสามารถบรรจุลงในฟิลด์นี้ได้เลย แต่ผู้รับจะไม่รายงานค่า L_a ทั้งนี้เนื่องจากว่าอัลกอริทึม Bolot ไม่ได้ใช้ค่าจริงของ L_a แต่จะคำนวณค่า L_a โดยใช้ค่า L_b หารด้วยค่า Reward ของ Combination ที่กำลังใช้งาน และนำค่า L_a ไปใช้ในการตัดสินใจว่าควรเลือกที่จะเพิ่มหรือลด Combination



รูปที่ 3.5 แผนภาพการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม Bolot

จากรูปที่ 3.6 จะเห็นว่าอัลกอริทึม Bolot มีค่าเทรสโฮลด์ที่ใช้ในการพิจารณาค่า L_a อยู่ 2 ค่า คือ HIGH และ LOW ถ้าหากค่า L_a สูงเกินกว่าค่าเทรสโฮลด์ HIGH ให้เพิ่ม Combination โดยการเพิ่ม Combination หมายถึงการเลือกใช้ Combination ในตารางที่ 3.1 ในขั้นที่สูงขึ้น เช่น ถ้าหากเดิมใช้ Combination หมายเลข 1 เมื่อใดที่ค่า L_a สูงกว่าเทรสโฮลด์ HIGH ก็จะเลื่อนขึ้นไปใช้ Combination หมายเลข 2 เป็นต้น แต่ถ้าค่า L_a ต่ำกว่าค่าเทรสโฮลด์ LOW ให้ลด Combination โดยใน [13]ได้กำหนดให้ค่าเทรสโฮลด์ HIGH และ LOW มีค่าเท่ากันคือ 3% เพื่อควบคุมให้ค่า L_a อยู่ที่ 3% แต่การกำหนดค่าเทรสโฮลด์ทั้งสองให้มีค่าเท่ากันจะทำให้มีการเปลี่ยน Combination บ่อยครั้ง เพราะไม่มีช่วงของค่า L_a ช่วงใดเลยที่อัลกอริทึมนี้ไม่ต้องมีการเปลี่ยนแปลง Combination



รูปที่ 3.6 การเพิ่มและลด Combination ในอัลกอริทึม Bolot

หลักการทำงานของอัลกอริทึม Bolot สามารถสรุปเป็น Pseudo Code ได้ดังรูปที่ 3.7 โดยขั้นตอนที่ 1 เป็นคำนวณค่า L_b จากค่าที่อ่านได้จากแพ็กเก็ต Recevier Report โดยค่าของ L_b จะระบุอยู่ในในฟิลด์ Fraction Lost ซึ่งปกติแล้วการระบุค่าอัตราการสูญหายในแพ็กเก็ตในฟิลด์ Fraction Lost จะไม่ระบุเป็นค่าทศนิยมหรือค่าที่เป็นเปอร์เซ็นต์โดยตรง แต่จะระบุเป็นอัตราส่วนเมื่อเทียบกับ 256[8] เนื่องจากฟิลด์นี้มีขนาด 1 ไบต์ ตัวอย่างเช่น ถ้าค่าในฟิลด์ Fraction Loss เท่ากับ 128 หมายถึงอัตราการสูญหายของแพ็กเก็ตเท่ากับ 128/256 = 50%

ในขั้นตอนที่ 2 เป็นการคำนวณหาค่า L_a โดยใช้ค่า L_b หารด้วย Reward ของ Combination ปัจจุบัน จากนั้นในชั้นตอนที่ 3 และขั้นตอนที่ 4 จะนำค่า L_a ที่คำนวณได้ไปใช้ในการตัดสินใจว่าจะเพิ่มหรือลด Combination

```
For each RTCP packet received do
   1. Calculate loss rate before reconstruction, Lb
   2. Calculate loss rate after reconstruction
      La = Lb / Reward associated with current combination
number
   3. If (La > HIGH) then
         Increment combination
   4. If (La < LOW) then
         Decrement combination
```

รูปที่ 3.7 Pseudo Code อัลกอริทึม Bolot

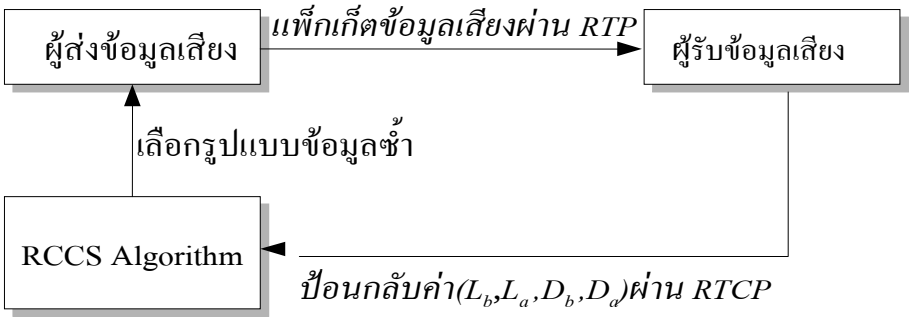
3.2.2 อัลกอริทึม RCCS

อัลกอริทึม RCCS[14] ซึ่งย่อมาจาก Redundant Codec Combination Selection เป็นอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวที่มีการเลือก Combination โดยใช้พารามิเตอร์

สองค่าคือ อัตราการสูญหายของข้อมูลและค่าเวลาหน่วงระหว่างปลายทาง (End-to-End Delay) ซึ่งแตกต่างจากอัลกอริทึม Bolot ที่ใช้เพียงอัตราการสูญหายของข้อมูลเพียงอย่างเดียว ตารางที่ 3.2 แสดง Combination ที่ใช้ในอัลกอริทึม RCCS ซึ่งจะเห็นว่ารูปแบบของการจัดวางข้อมูลซ้ำ รวมทั้งชนิดของการบีบอัดเสียงในแต่ละ Combination นั้นแตกต่างไปจากอัลกอริทึม Bolot ในคอลัมน์ Reward แสดงค่า Reward ของแต่ละ Combination ซึ่งค่าเหล่านี้ได้มาจากค่า Reward ของ Bolot นั่นเอง โดยค่า Reward ของแต่ละ Combination ในตารางที่ 3.2 ของอัลกอริทึม RCCS นำมาจากค่า Reward ของ Combination ในตารางที่ 3.1 ของอัลกอริทึม Bolot ที่มีจำนวนข้อมูลซ้ำและตำแหน่งของข้อมูลซ้ำตรงกัน โดยไม่สนใจชนิดของการบีบอัดเสียง แต่ค่า Reward ในตารางที่ 3.2 นี้เป็นเพียงค่า Reward เริ่มต้นเท่านั้น เพราะอัลกอริทึม RCCS จะมีการเปลี่ยนแปลงค่า Reward ของ Combination ที่กำลังใช้งานทุกครั้งที่ได้รับแพ็กเก็ต Receiver Report และในคอลัมน์ขวาสุดคือคอลัมน์ Penalty เป็นค่าอัตราส่วนระหว่างค่าเวลาหน่วงระหว่างปลายทางหลังการจัดเรียงข้อมูลและค่าเวลาหน่วงระหว่างปลายทางก่อนการจัดเรียงข้อมูลซ้ำ

$$\text{Penalty} = \frac{\text{ค่าเวลาหน่วงระหว่างปลายทางหลังการจัดเรียงข้อมูลซ้ำ}}{\text{ค่าเวลาหน่วงระหว่างปลายทางก่อนการจัดเรียงข้อมูลซ้ำ}} \qquad (3.2)$$

ซึ่งค่า Penalty ที่อยู่ในตารางที่ 3.2 เป็นค่าเฉลี่ยที่ได้จากการทดลองหลายๆ ครั้ง แต่ใน[14] ก็ไม่ได้อธิบายเอาไว้อย่างชัดเจนว่าการหาค่าเวลาหน่วงก่อนและหลังการจัดเรียงมีวิธีการอย่างไร ซึ่งการหาค่าเวลาหน่วงระหว่างปลายทางนั้นทำได้ยาก เนื่องจากว่านาฬิกาของเครื่องผู้ส่งและเครื่องผู้รับไม่ประสานกัน จากแผนภาพการควบคุมความผิดพลาดแบบปรับตัวของอัลกอริทึม RCCS ในรูปที่ 3.8 จะเห็นว่าค่าที่ผู้รับจะต้องรายงานให้กับผู้ส่งมีอยู่ 4 ค่า คือ L_b, L_a, D_b (เวลาหน่วงระหว่างปลายทางก่อนการจัดเรียงข้อมูล) และ D_a (เวลาหน่วงระหว่างปลายทางหลังการจัดเรียงข้อมูล)



รูปที่ 3.8 แผนภาพการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม RCCS

ตารางที่ 3.2 ค่า Reward และ Penalty ของแต่ละ Combinationในอัลกอริทึม RCCS

No	Combination	Reward	Penalty
0	(G.711)	1	1
1	(G.711,GSM(1))	2.5	1.5
2	(G.711,G.723(1))	2.5	2
3	(GSM,G.723(1))	2.5	4
4	(G.711,GSM(1),G.729(2))	6	2.4
5	(G.729,G.723(1),LPC10(2))	6	4.5
6	(G.711,GSM(1),G.729(2),LPC10(3))	18	3.4
7	(G.711,GSM(1),G.723(2),LPC10(3))	18	4.5

รูปที่ 3.9 เป็น Pseudo Code ของอัลกอริทึม RCCS ซึ่งจะเห็นว่ามีการใช้พารามิเตอร์หลายตัว โดยคำอธิบายของพารามิเตอร์แต่ละตัวนั้นอยู่ในตารางที่ 3.3 จาก Pseudo Code จะเห็นเมื่อได้ค่า L_b, L_a, D_b และ D_a ในขั้นตอนที่ 1 - 2 แล้ว ในขั้นตอนที่ 3 จะมีการแก้ไขค่า Reward (R) และ Penalty (P) ของ Combination ปัจจุบัน โดยใช้สมการตัวกรองดังนี้

$$R_i = \alpha(L_b / L_a) + (1-\alpha)R_{i-1} \quad (3.3)$$

$$P_i = \alpha(D_a / D_b) + (1-\alpha)P_{i-1} \quad (3.4)$$

ตารางที่ 3.3 คำอธิบายของพารามิเตอร์ที่ใช้ในอัลกอริทึม RCCS

พารามิเตอร์	คำอธิบาย
R_i	ค่า Reward ของ Combination ในลำดับที่ i
P_i	ค่า Penalty ของ Combination ในลำดับที่ i
L_a (L_a')	อัตราการสูญหายของแพ็กเก็ตหลังการจัดเรียง (ค่า L_a' คือค่าทำนายของ L_a)
L_b (L_b')	อัตราการสูญหายของแพ็กเก็ตก่อนการจัดเรียง (ค่า L_b' คือค่าทำนายของ L_b)
D_a	ค่าเวลาหน่วงระหว่างปลายทางหลังการจัดเรียง
D_b	ค่าเวลาหน่วงระหว่างปลายทางก่อนการจัดเรียง
α	ค่าสำหรับกำหนดความเร็วของการเปลี่ยนแปลงค่า R_i และ P_i

```
For each RTCP packet received do
1.Calculate packet loss and delay before reconstruction Lb and
Db
  Lb = Number of packet loss before reconstruction / Number of
       packet expected
  Db = end-to-end delay before reconstruction

2.Calculate packet loss and delay after reconstruction La and
Da
  La = Number of packet loss after reconstruction / Number of
       packet expected
  Da = end-to-end delay after reconstruction

3.Update reward and penalty of current combination with Ri and
Pi
  in combination table

4.If(La > HIGH or La < LOW)
    For each combination j in the combination table
       Calculate and predict La', La' = Lb/Rj
       Calculate and predict Da', Da' = Db*Pj
          If(La' < HIGH and La' > LOW)
             If(Da' < DMin)
                selected combination = j
                Dmin = Da'

5.combination = selected combination
```

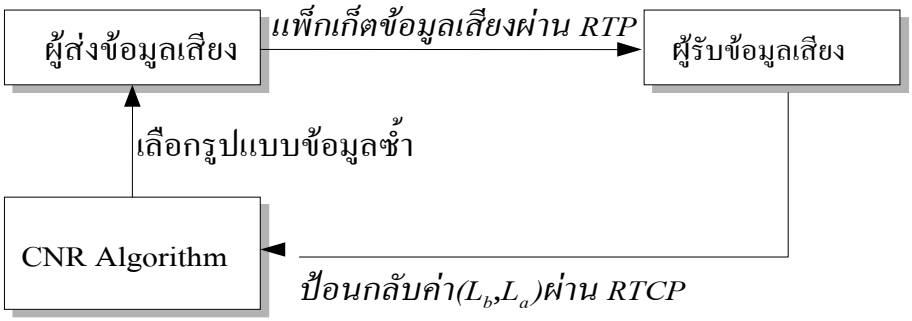
รูปที่ 3.9 Pseudo Code ของอัลกอริทึม RCCS

การแก้ไขค่า Reward นี้เพื่อให้ค่า Reward มีค่าเปลี่ยนแปลงไปตามสภาพเครือข่ายจริง เนื่องจากว่าค่า Reward ที่ได้จากการทดลองของ Bolot อาจจะไม่ได้ถูกต้องเสมอไปในทุกสภาพแวดล้อม จากนั้นในขั้นตอนที่ 4 เป็นการเลือก Combination โดยจะพิจารณาจากตารางที่ 3.2 ตั้งแต่ Combination แรกจนถึง Combination สุดท้าย และจะเลือก Combination แรกที่ให้ค่า L_a' (ค่าทำนายของ L_a ซึ่งได้จากการนำ L_b หารด้วยค่า Reward) อยู่ในช่วงระหว่างเทรสโฮลด์ LOW และ HIGH และ Combination ดังกล่าวจะต้องให้ค่า D_a' (ค่าทำนายของเวลาหน่วงระหว่างปลายทางหลังการจัดเรียงข้อมูล) ไม่เกินค่าเทรสโฮลด์ DMin ด้วย

3.2.3 อัลกอริทึม CNR

อัลกอริทึม CNR (Centre for Network Research) ได้แก้ไขจุดบกพร่องของอัลกอริทึม Bolot โดยการใช้ค่าจริงของ L_a ในการตัดสินเปลี่ยน Combination และได้แก้ไขปัญหาการแกว่งของค่า L_a โดยการนับจำนวนครั้งที่ค่าของ L_a ต่ำกว่าเทรสโฮลด์ LOW โดยจะลด Combination ก็ต่อเมื่อ L_a มีค่าต่ำกว่าเทรสโฮลด์ LOW ติดต่อนานเพียงพอแล้ว เพื่อให้แน่ใจความคับคั่งของเครือข่ายลดลงจริง และอัลกอริทึม CNR ได้แก้ไขจุดบกพร่องของอัลกอริทึม RCCS โดยการแยกเงื่อนไขในการตัดสินใจลดและเพิ่ม Combination ออกจากกัน

จากแผนภาพการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม CNR ใน[8] จะเห็นว่าผู้รับจะมีการรายงานค่า L_b และ L_a ให้กับผู้ส่งโดยใช้โปรโตคอล RTCP ซึ่งผู้ส่งจะนำค่าทั้งสองนี้ไปใช้ในการตัดสินใจว่าควรจะเลือก Combination ใด โดยแพ็กเก็ต RTCP ที่ใช้ในการรายงานค่าทั้งสองนี้ก็คือแพ็กเก็ต Receiver Report ผู้รับจะนำค่า L_b วางในฟิลด์ที่ชื่อ Fraction Lost ส่วนค่า L_a นั้นจะถูกนำไปอาไว้ในส่วนขยายเฮดเดอร์ RTCP



รูปที่ 3.10 แผนภาพการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม CNR

รูปที่ 3.10 เป็นรูปแบบของแพ็กเก็ต Receiver Report ที่มีการระบุทั้งค่า L_b และ L_a โดยระบุที่ฟิลด์ fraction lost และ fraction L_a ตามลำดับ ฟิลด์ fraction L_a อยู่ในส่วนขยายเฮดเดอร์ และมีขนาด 8 บิตเช่นเดียวกับฟิลด์ fraction lost แต่เนื่องจากใน[8] ได้ระบุได้ว่าความยาวของแพ็กเก็ต RTCP จะต้องมีขนาดที่หาร 4 ลงตัว ดังนั้นจึงมีส่วนที่ไม่ได้ใช้งานเหลืออยู่ 3 ไบต์ และเนื่องจากว่าใน [8]ได้กำหนดเอาไว้ว่าค่าในฟิลด์ fraction lost คือค่าอัตราการสูญหายของแพ็กเก็ตคูณด้วย 256 ดังนั้นการระบุค่า L_b และ L_a จึงใช้วิธีการเดียวกันคือ

$$\text{fraction lost} = L_b \times 256 \quad (3.5)$$

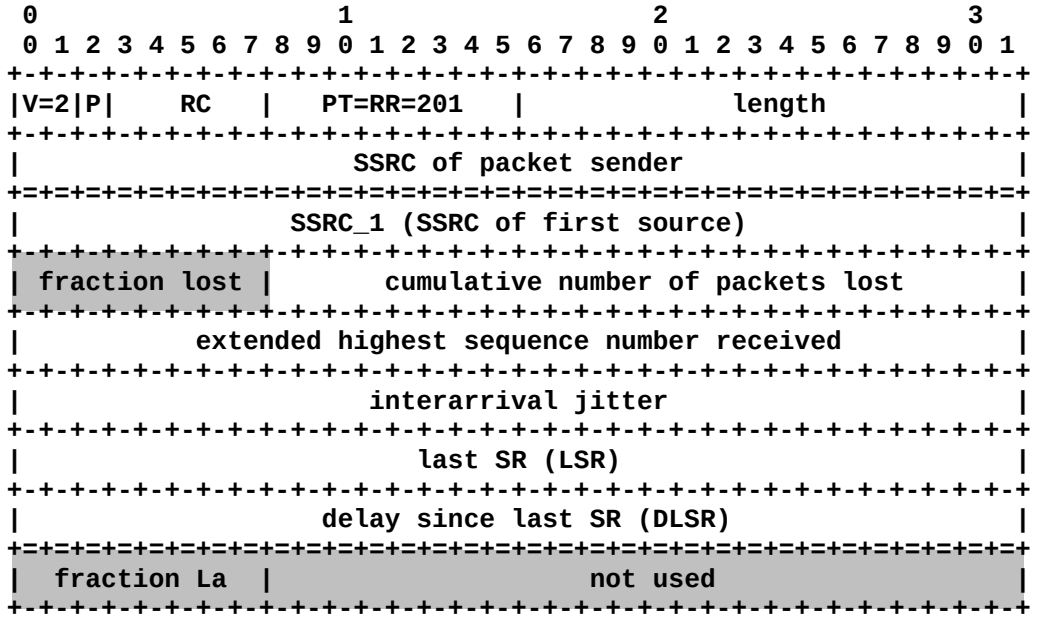
$$\text{fraction } L_a = L_a \times 256 \quad (3.6)$$

และในการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม CNR เมื่อผู้ส่งได้รับแพ็กเก็ต Receiver Report ผู้ส่งก็จะอ่านค่า L_b และ L_a ที่อยู่ในแพ็กเก็ตดังนี้

$$L_b = \text{fraction loss} / 256 \quad (3.7)$$

$$L_a = \text{fraction } L_a / 256 \quad (3.8)$$

ซึ่งค่า L_b และ L_a จะถูกนำไปใช้ในการตัดสินเลือกว่าจะใช้ Combination ใด แต่อัลกอริทึม CNR นั้นไม่ได้มีการเพิ่ม Combination ครั้งละหนึ่งขั้นเช่นเดียวกับอัลกอริทึม Bolot แต่อัลกอริทึม CNR จะนำค่า Reward ของแต่ละ Combination มาใช้ในการเลือกว่า Combination ที่เหมาะสมกับสภาพเครือข่ายคือ Combination ใด และสิ่งที่อัลกอริทึม CNR แตกต่างจากอัลกอริทึมอื่นอีกประการหนึ่งคือ ในอัลกอริทึมอื่นๆ ข้อมูลซ้ำมักจะใช้การบีบอัดเสียงที่มีอัตราบิตต่ำกว่าข้อมูลหลักเพื่อเป็นการประหยัดแบนด์วิดท์ แต่อัลกอริทึม CNR ใช้การบีบอัดเสียง G.723.1 ทั้งข้อมูลหลักและข้อมูลซ้ำ เนื่องจากการบีบอัดเสียงแบบนี้มีอัตราบิตที่ต่ำอยู่แล้ว (6.3 kbps) และการที่ข้อมูลหลักและข้อมูลเสียงใช้การบีบอัดเสียงที่แตกต่างกันก็ทำให้ผู้ส่งต้องเสียเวลามากขึ้น เนื่องจากต้องมีการเข้ารหัสเสียงในแต่ละเฟรมมากกว่า 1 ครั้ง แต่ถ้าใช้การบีบอัดเสียงชนิดเดียวกันก็สามารถเข้ารหัสเพียงครั้งเดียวและเก็บข้อมูลหลังการบีบอัดเอาไว้เป็นข้อมูลซ้ำในภายหลังได้เลย



รูปที่ 3.11 การระบุค่า L_b และ L_a ในแพ็กเก็ต Receiver Report

สำหรับ Combination ของข้อมูลซ้ำที่ใช้ในอัลกอริทึม CNR มี 6 Combination ดังแสดงในตารางที่ 3.4 โดยในคอลัมน์ Combination Format ได้กำหนดสัญลักษณ์เป็นจำนวนเต็มลบเพื่อใช้แทนรูปแบบของข้อมูลซ้ำ ตัวอย่างเช่น สัญลักษณ์ -1 หมายความว่า ในแต่ละแพ็กเก็ตลำดับที่ N จะมีข้อมูลซ้ำ ของแพ็กเก็ตลำดับที่ N-1 อยู่ และสัญลักษณ์ -1-2 หมายความว่า ในแต่ละแพ็กเก็ต N จะมีข้อมูลซ้ำของแพ็กเก็ต N-1 และ N-2 อยู่ สำหรับสัญลักษณ์ของ Combination อื่นๆ ก็สามารถแปลความหมายได้ในทำนองเดียวกันนี้ ส่วนในคอลัมน์ Initial Reward เป็นค่าเริ่มต้นของ Reward ในแต่ละ Combination ซึ่งก็คือค่า Reward ที่ได้จากการทดลองของ Bolot นั่นเอง แต่ในอัลกอริทึม CNR นี้ค่า Reward จะไม่คงที่ โดยจะถูกแก้ไขให้เปลี่ยนแปลงไปตามสภาพของเครือข่าย

ตารางที่ 3.4 Combination ของข้อมูลซ้ำที่ใช้ในอัลกอริทึม CNR

No.	Combination Format	Initial Reward (Bolot Reward)	Bit Rate include Overhead (kbps)
0	-	1	17.1
1	-1	2.5	24.8
2	-2	6	24.8
3	-1-2	6	32.3
4	-1-3	10	32.3
5	-1-2-3	18	39.7

รูปที่ 3.12 เป็น Pseudo Code ของอัลกอริทึม CNR เมื่อผู้ส่งได้รับแพ็กเก็ต Receiver Report ก็จะอ่านค่า L_b และ L_a ในขั้นตอนแรกจะแก้ไขค่า Reward ของ Combination ปัจจุบัน ส่วนในขั้นตอนที่สองเป็นการนับจำนวนครั้งที่ค่า L_b มีค่าต่ำกว่าเทรสโฮลด์ LOW (เก็บไว้ในตัวแปร count_L_b_under_low) ซึ่งตัวแปรตัวนี้จะมีการตรวจสอบอีกครั้งเมื่อมีการตัดสินใจลด Combination ในขั้นตอนที่สามเป็นการตรวจสอบว่าถ้า L_a ที่อ่านจากแพ็กเก็ต Receiver Report มีค่าสูงกว่าเทรสโฮลด์ HIGH ก็จะต้องมีการเพิ่ม Combination ซึ่งทำได้โดยนำค่า L_b และค่า Reward ของ Combination ที่อยู่ในลำดับถัดจาก Combination ปัจจุบันเป็นต้นไปมาประมาณว่าถ้าเลือกใช้แต่ละ Combination แล้วจะได้ค่า L_a เท่าไร จากนั้นก็จะเลือก Combination แรกที่ให้ค่าประมาณของ L_a (ตัวแปร L_{ai} ใน Pseudo Code) ไม่เกินเทรสโฮลด์ HIGH ซึ่งการเพิ่ม Combination แบบนี้สามารถประหยัดเวลาได้มากกว่าการเพิ่ม Combination ครั้งละหนึ่งขั้น

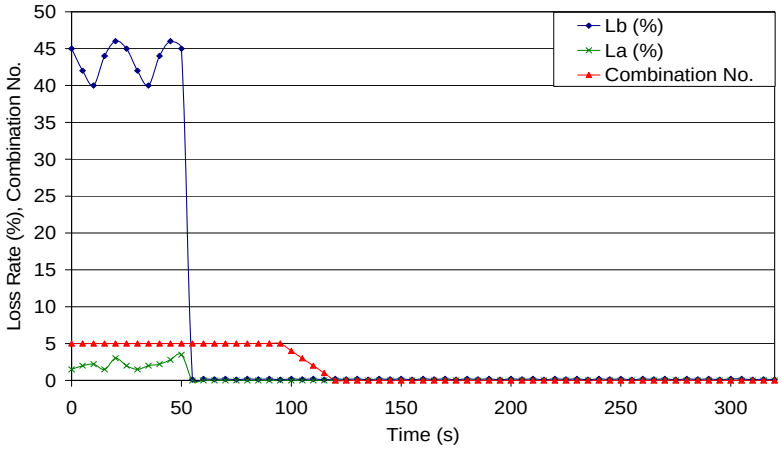
ส่วนในกรณีที่ค่า L_a ที่อ่านมาจากแพ็กเก็ต Receiver Report มีค่าต่ำกว่าเทรสโฮลด์ LOW อัลกอริทึม CNR จะไม่ลด Combination ทันที แต่จะนับจำนวนครั้งที่ L_a มีค่าต่ำกว่าเทรสโฮลด์ LOW ในช่วงเวลาติดกัน (เก็บไว้ในตัวแปร count_L_a_under_low) การลด Combination ในอัลกอริทึมจะเกิดขึ้นได้ก็ต่อเมื่อค่า count_L_a_under_low หรือค่า count_L_b_under_low มีค่ามากกว่าหรือเท่ากับ MIN_UNDER_LOW โดยการที่ค่าของ count_L_a_under_low มีค่ามากกว่าหรือเท่ากับ MIN_UNDER_LOW หมายความว่า Combination ปัจจุบันที่ใช้อยู่สามารถทำให้ค่า L_a ต่ำกว่าเทรสโฮลด์ LOW เป็นระยะเวลานานพอแล้ว และปลอดภัยพอที่จะลองลด Combination ซึ่งเทคนิคนี้จะช่วยลดผลกระทบที่เกิดจากการแกว่งของค่า L_a ได้

```
For each receiver report packet received do
1.Update reward of the current combination
    //Let Rc stands for reward of the current combination
    Rc = Lb/La

2.if(Lb < LOW)
    increment count_Lb_under_low
  else
    count_Lb_under_low = 0

3.if(La > HIGH)
    selected_combination = MAX_COMBINATION;
    for(i = current_combination+1 to MAX_COMBINATION)
        Ri = reward of combination No.i
        Lai = Lb/Ri
        if(Lai <= HIGH) {
            selected_combination = i
            break;
    current_combination = selected_combination;
    count_La_under_low = 0

  else if(La < LOW)
    increment count_La_under_low

  else
    count_La_under_low = 0

4.if((count_La_under_low >= MIN_UNDER_LOW)OR
     (count_Lb_under_low >= MIN_UNDER_LOW))
    decrement current_combination
    count_La_under_low = 0
```

รูปที่ 3.12 Pseudo Code ของอัลกอริทึม CNR

การลด Combination เมื่อค่า count_L_a_under_low หรือค่า count_L_b_under_low มีค่าต่ำกว่าค่า MIN_UNDER_LOW ดังแสดงในรูปที่ 3.13 ซึ่งค่าของ MIN_UNDER_LOW นั้น ถ้าหากมีค่ามากจะทำให้ป้องกันการแกว่งของค่า L_a ได้ดี แต่การลด Combination ทำได้ช้า และส่งผลให้มี

ปริมาณการใช้แบนด์วิดท์สูงเนื่องจากใช้ปริมาณข้อมูลซ้ำมาก ในที่นี้เลือกใช้ค่า MIN_UNDER_LOW เท่ากับ 10 ซึ่งมีค่าเท่ากับระยะเวลา 50 วินาทีเนื่องจากเป็นค่าที่มากพอที่จะลดปัญหาการแกว่งของค่า L_a ได้ดี



การลด Combination โดยพิจารณาทั้ง count_L_a_under_low และ count_L_b_under_low

รูปที่ 3.13 การเปลี่ยนแปลงการลด Combination (โดยพิจารณาทั้ง count_L_a_under_low และ count_L_b_under_low)

3.3 การประยุกต์ใช้วิธีกำลังสองน้อยที่สุดในอัลกอริทึมแบบปรับตัว CNR

ในอัลกอริทึม Bolot และ RCCS ค่า reward value ถูกกำหนดไว้แล้ว ทั้งอัลกอริทึม Bolot และอัลกอริทึม RCCS ต่างก็มีวิธีการที่คล้ายคลึงกันก็คือ มีการสร้างตาราง Combination เอาไว้ก่อน สิ่งที่แตกต่างกันก็คือวิธีการที่จะเป็นตัวกำหนดว่าเมื่อใดควรจะเพิ่มหรือลด Combination ซึ่งอัลกอริทึม Bolot พิจารณาจากค่า L_a ส่วนอัลกอริทึม RCCS นั้นพิจารณาจากทั้ง L_a และค่าเวลาหน่วงระหว่างปลายทาง ซึ่งอัลกอริทึม RCCS มีจุดบกพร่องในส่วนของการเลือก Combination ซึ่งไม่ได้มีการแยกเงื่อนไขในการลดและเพิ่ม Combination จาก Pseudo Code ของอัลกอริทึม RCCS ในรูปที่ 3.9 จะเห็นว่าอัลกอริทึมนี้จะใช้ค่า Reward เพื่อค่า L_a' (ค่าทำนายของ L_a)ในแต่ละ Combination และจะเลือก Combination ที่มีค่า L_a' ที่มีค่าต่ำกว่าเทรสโฮลด์ HIGH และสูงกว่าเทรสโฮลด์ LOW ซึ่งการใช้เงื่อนไขร่วมกันอย่างนี้อาจทำให้อัลกอริทึมนี้ไม่สามารถเพิ่ม Combination ได้ในบางครั้ง เช่น ถ้า Combination ในขั้นที่สูงกว่า Combination ปัจจุบันทุก Combination ต่างก็มีค่า L_a' ที่ต่ำกว่าเทรสโฮลด์ LOW ก็ถือว่าไม่ตรงตามเงื่อนไขแล้ว ซึ่งทำให้ไม่สามารถเพิ่ม Combination ได้ถึงแม้ว่าในขณะนั้นค่า L_a จะมีค่าสูงเกินกว่าเทรสโฮลด์ HIGH ก็ตาม การที่จะวัดว่าอัลกอริทึมใดดีกว่ากัน ต้องดูว่าอัลกอริทึมเหล่านี้สามารถเลือกใช้ Combination ที่สามารถรักษา

ระดับของ L_a ไม่ให้สูงเกินไปได้หรือไม่ รวมทั้งต้องไม่ใช้ปริมาณข้อมูลซ้ำมากเกินความจำเป็นอีกด้วย เพราะถ้าไม่คำนึงถึงปริมาณแบนด์วิดท์ที่เพิ่มขึ้นจากการใช้ข้อมูลซ้ำแล้ว ก็ไม่จำเป็นจะต้องใช้อัลกอริทึมใดเลยก็ได้ เพราะยิ่งใช้ข้อมูลซ้ำมากก็ยิ่งสามารถกู้ข้อมูลที่สูญหายได้มาก แต่การใช้อัลกอริทึมในการควบคุมความผิดพลาดเหล่านี้ก็เพื่อที่จะกำหนดให้ใช้ปริมาณข้อมูลซ้ำเท่าที่จำเป็นเท่านั้นและเพื่อให้ใช้แบนด์วิดท์ได้อย่างคุ้มค่า ส่วนอัลกอริทึม CNR มีการปรับค่าของ reward value ที่เป็นอัตราส่วนของ L_b ต่อ L_a เพื่อให้มีการปรับค่า reward value ให้เหมาะสมกับเครือข่ายในขณะนั้น ๆ ในวิทยานิพนธ์ฉบับนี้จึงนำเทคนิคการประมาณค่าด้วยวิธีกำลังสองน้อยที่สุดมาใช้ในการปรับค่า reward ของอัลกอริทึม CNR แล้วเปรียบเทียบกับการใช้ค่า reward value แบบกำหนดค่าในอัลกอริทึม Bolot และจากรูปที่ 3.7 ซึ่งแสดง Pseudo Code อัลกอริทึม Bolot จะเห็นได้ว่า อัลกอริทึม Bolot มีจุดบกพร่องในส่วนของการตัดสินใจเพิ่มหรือลด Combination โดยพิจารณาจากค่า L_a แต่ค่า L_a ที่ใช้ไม่ใช่ค่าจริง แต่เป็นค่า L_a ที่ได้จากการคำนวณโดยการนำค่า L_b หารด้วยค่า Reward ของ Combination ปัจจุบัน ซึ่งค่า L_a จากการคำนวณดังกล่าวอาจจะไม่ถูกต้องเสมอไปในทุกสภาพแวดล้อม และอาจส่งผลให้อัลกอริทึมนี้ทำงานผิดพลาดได้ และเงื่อนไขในการปรับเพิ่มหรือลด Combination นั้นจะปรับเพิ่มหรือลดที่ละ 1 Combination ซึ่งถ้าค่า reward value ในขณะนั้นยังไม่เหมาะสมกับสภาพเครือข่ายในขณะนั้น ก็จะใช้เวลานานกว่าจะทำการปรับเลือกค่า Combination เพื่อควบคุมความผิดพลาดโดยการกู้คืนแพ็กเก็ตที่สูญเสียให้เหมาะกับสภาพเครือข่ายในขณะนั้น

โดยในอัลกอริทึม C NR ได้นำเทคนิคการประมาณค่าด้วยวิธีกำลังสองน้อยที่สุดมาใช้ โดยเปลี่ยนค่า reward value จากสมการที่ 3.1 ซึ่งเป็นค่าเดียวกันกับการหาค่า reward value (R_c)ในรูปที่ 3.12 ซึ่งแสดง Pseudo Code ของอัลกอริทึม CNR ดังนั้นจะได้ค่า reward value ที่นำไปปรับค่า combination number ของแต่ละรอบการทำงานถัด ๆ ไปดังสมการ 3.9

$$R_c = A + B(L_b/L_a) \tag{3.9}$$

โดยค่า A และ B เป็นค่าคงที่ ที่หาได้จากสมการที่ 2.6

3.4 การตรวจสอบความถูกต้องของแบบจำลอง

เนื่องจากในการทดลองเพื่อประเมินผลอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวแต่ละอัลกอริทึมทำโดยการใช้แบบจำลอง ดังนั้นก่อนที่จะทำการทดลองจำเป็นที่ต้องแน่ใจเสียก่อนว่าแบบจำลองที่จะนำมาใช้ในการทดลองนั้นสามารถทำงานได้อย่างถูกต้อง ซึ่งการที่จะยืนยันว่าแบบจำลองที่ใช้ในการทดลองนั้นสามารถทำงานได้อย่างถูกต้องจำเป็นที่จะต้องมีการตรวจสอบสองขั้นตอน โดยขั้นตอนแรกคือตรวจสอบการคำนวณค่า L_b และ L_a ของผู้รับ และ

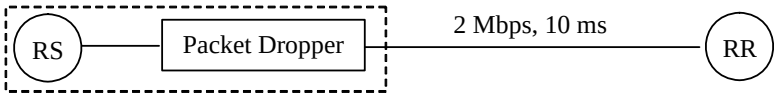
ขั้นตอนที่สองคือตรวจสอบการทำงานของอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวแต่ละอัลกอริทึม

3.4.1 การตรวจสอบการคำนวณค่า L_b และ L_a ของผู้รับ

เนื่องจากว่าค่า L_b และค่า L_a เป็นค่าที่จำเป็นสำหรับอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวที่จะต้องนำไปใช้ในการตัดสินใจเลือก Combination ของข้อมูลซ้ำ ดังนั้นหากเกิดข้อผิดพลาดในการคำนวณค่าทั้งสอง จะส่งผลให้การทำงานในขั้นตอนต่อไปเกิดข้อผิดพลาดตามไปด้วย

ในการทดลองนี้ได้กำหนดรูปแบบการเชื่อมต่อภายในเครือข่าย ดังรูปที่ 3.14 ซึ่งจะเห็นว่ามีโหนดเพียง 2 โหนดเท่านั้น โดยโหนด RS จะส่งแพ็กเก็ตเสียงไปให้โหนด RR แต่ที่โหนด RS จะมีตัวตัดทิ้งแพ็กเก็ต (Packet Dropper) ที่สามารถตัดทิ้งแพ็กเก็ตได้ตามรูปแบบที่กำหนด ทั้งนี้เพื่อใช้ในการทดสอบว่าการทำงานของแบบจำลองในส่วนของการคำนวณค่า L_a และ L_b ทำได้ถูกต้องหรือไม่ โดยได้กำหนดรูปแบบการตัดทิ้ง (Drop) แพ็กเก็ตทั้งหมด 5 รูปแบบดังตารางที่ 3.5 ซึ่งจากรูปแบบการตัดทิ้งแพ็กเก็ตดังกล่าวสามารถที่จะคำนวณค่า L_b และ L_a ที่ถูกต้องได้เลย ซึ่งหากแบบจำลองสามารถทำงานได้อย่างถูกต้อง ค่า L_b และ L_a ที่ได้จากการทดลองจะต้องมีค่าตรงกัน โดยในการตัดทิ้งแพ็กเก็ตแต่ละรูปแบบจะมีการจำลอง 6 ครั้ง (สำหรับ 6 Combination) และในการจำลองแต่ละครั้ง พฤติกรรมในการส่งแพ็กเก็ตของโหนด RS และโหนด RR เป็นดังนี้

- โหนด RS ส่งแพ็กเก็ตเสียงโดยใช้โปรโตคอล RTP ไปให้โหนด RR โดยส่งแพ็กเก็ตทุกๆ 30 ms ซึ่งเป็นระยะเวลาของเสียงพูด 1 เฟรมสำหรับการบีบอัดเสียงด้วย G.723.1 ส่วนขนาดแพ็กเก็ตขึ้นอยู่กับ Combination ที่ใช้
- โหนด RR จะมีการรายงานค่า L_b และ L_a โดยส่งมาในแพ็กเก็ต Receiver Report ให้กับโหนด RS ซึ่งแพ็กเก็ตนี้มีการส่งทุก 5 วินาที
- โหนด RS เริ่มส่งแพ็กเก็ตเสียงตั้งแต่วินาทีที่ 0 และหยุดที่วินาทีที่ 500
- ค่า L_b และ L_a ที่ได้จากการทดลองเป็นค่าเฉลี่ยของค่า L_b และ L_a ที่ RR รายงานให้กับ RS ทุกครั้ง (100 ครั้ง)



รูปที่ 3.14 การเชื่อมต่อภายในเครือข่ายที่ใช้ในการจำลอง

ตารางที่ 3.5 เป็นผลการตรวจสอบความถูกต้องในการคำนวณค่าของ L_b ของผู้รับ เนื่องจากว่าค่า L_b คืออัตราการสูญหายของข้อมูลเสียงก่อนการจัดเรียง ซึ่งก็คืออัตราการสูญหายของ

แพ็กเก็ตนั่นเอง ดังนั้นในกรณีนี้ค่า L_b ที่ถูกต้องจะต้องมีค่าเท่ากับอัตราการตัดทิ้งแพ็กเก็ต เนื่องจากในการทดลองการสูญหายของแพ็กเก็ตเกิดจากการตัดทิ้งเพียงอย่างเดียวเท่านั้น และไม่ว่าจะใช้จำนวนข้อมูลซ้ำเท่าใดก็จะไม่มีผลต่อค่า L_b เพราะว่าเป็นการพิจารณาในขณะที่ยังไม่มีการนำข้อมูลซ้ำมากู้ข้อมูลในแพ็กเก็ตที่สูญหาย จากผลการทดลองในตารางที่ 3.5 จะเห็นว่าการคำนวณค่า L_b ของผู้รับสามารถทำได้ถูกต้องในทุกกรณี นั่นคือค่า L_b มีค่าเท่ากับอัตราการตัดทิ้งแพ็กเก็ตเสมอไม่ว่าจะใช้รูปแบบการตัดทิ้งแพ็กเก็ตแบบใด

ตารางที่ 3.5 การตรวจสอบความถูกต้องในการคำนวณค่า L_b ของผู้รับในแบบจำลอง

รูปแบบในการตัดทิ้ง	**หมายเลขแพ็กเก็ตที่ถูกตัดทิ้ง (ในทุก 100 แพ็กเก็ต)**	**อัตราการตัดทิ้งแพ็กเก็ต (%)**	**ค่า L_b (%) จากการทดลองของแต่ละ Combination**					
			0	**1**	**2**	**3**	**4**	**5**
D01	10N; N = 1, 2, …,10	10	10	10	10	10	10	10
D02	10N-2, 10N-1, 10N; N = 1, 2,…,10	30	30	30	30	30	30	30
D03	10N-3, 10N-2, 10N; N = 1, 2,…,10	30	30	30	30	30	30	30
D04	10N-2, 10N-1, 10N; N = 1, 2,…,10 10N-3; N=1,2,3	33	33	33	33	33	33	33
D05	10N-1, 10N; N = 1, 2,…,10 10N-2; N=1,2,3,4	24	24	24	24	24	24	24

ตารางที่ 3.6 แสดงผลการตรวจสอบความถูกต้องในการคำนวณค่า L_a ของผู้รับในแบบจำลอง ซึ่งมีการตัดทิ้งแพ็กเก็ตตามรูปแบบเดียวกันกับที่ใช้ในตารางที่ 3.5 ซึ่งการตัดทิ้งแพ็กเก็ตทั้ง 5 รูปแบบสามารถคำนวณได้ล่วงหน้าแล้วว่าค่า L_a ที่ถูกต้องของแต่ละ Combination มีค่าเท่าใด ดังนั้นหากแบบจำลองสามารถทำงานได้ถูกต้อง ค่า L_a ที่จากการทดลองจะต้องมีค่าตรงกันกับค่าที่ถูกต้อง

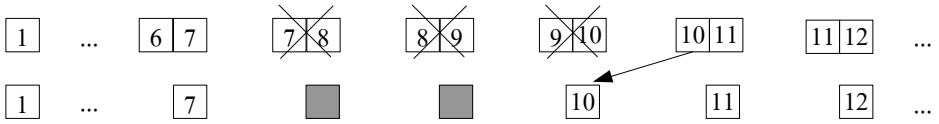
รูปที่ 3.15 เป็นตัวอย่างของการหาค่า L_a เมื่อกำหนดให้มีการตัดทิ้งแพ็กเก็ตแบบ D02 ซึ่งจากตารางที่ 3.6 จะเห็นว่าการตัดทิ้งแพ็กเก็ตแบบนี้ ในทุก 100 แพ็กเก็ตจะมีการตัดทิ้งแพ็กเก็ตเป็นจำนวน 30 แพ็กเก็ต (แพ็กเก็ตในหมายเลขที่ลงท้ายด้วย 8, 9 และ 0) หรืออาจจะกล่าวได้ว่าในทุก 10 แพ็กเก็ตจะมีการตัดทิ้ง 3 แพ็กเก็ต ดังนั้นในการหาค่า L_a ที่ถูกต้องก็สามารถพิจารณาเฉพาะ 10 แพ็กเก็ต และหาว่าหลังจากมีการกู้ข้อมูลในแพ็กเก็ตที่สูญหายแล้วมีข้อมูลเสียงสูญหายไปกี่เฟรมก็สามารถหาค่า L_a ได้แล้ว ซึ่งจะเห็นได้ว่าค่า L_a จากการทดลองของ Combination 0, 1 และ 2 ที่ได้จากการทดลองมีค่าถูกต้อง สำหรับในกรณีอื่นๆ หลังจากที่ได้มีการตรวจสอบแล้วพบว่าการ

คำนวณค่า L_a ของแบบจำลองสามารถทำงานได้ถูกต้องในทุกกรณี

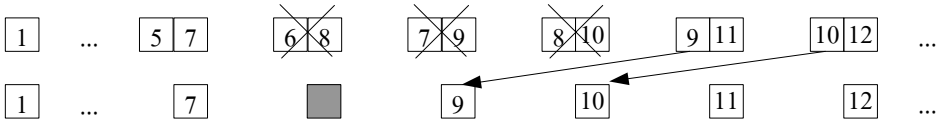
ตารางที่ 3.6 การตรวจสอบความถูกต้องในการคำนวณค่า L_a ของผู้รับในแบบจำลอง

รูปแบบในการตัดทิ้ง	หมายเลขแพ็กเก็ตที่ถูกตัดทิ้ง (ในทุก 100 แพ็กเก็ต)	อัตราการตัดทิ้งแพ็กเก็ต (%)	ค่า L_a (%) จากการทดลองของแต่ละ Combination					
			0	1	2	3	4	5
D01	10N; N = 1, 2, …,10	10	10	0	0	0	0	0
D02	10N-2, 10N-1, 10N; N = 1, 2,…,10	30	30	20	10	10	0	0
D03	10N-3, 10N-2, 10N; N = 1, 2,…,10	30	30	10	10	0	10	0
D04	10N-2, 10N-1, 10N; N = 1, 2,…,10 10N-3; N=1,2,3	33	33	23	13	13	3	3
D05	10N-1, 10N; N = 1, 2,…,10 10N-2; N=1,2,3,4	24	24	14	4	4	0	0

(a) Combination 0, ทุก 10 แพ็กเก็ตจะมีข้อมูลเสียงสูญหาย 3 เฟรม, L_a = 30%



(b) Combination 1, ทุก 10 แพ็กเก็ตจะมีข้อมูลเสียงสูญหาย 2 เฟรม, L_a = 20%



(c) Combination 2, ทุก 10 แพ็กเก็ตจะมีข้อมูลเสียงสูญหาย 1 เฟรม, L_a = 10%

รูปที่ 3.15 ตัวอย่างการหาค่า L_a เมื่อกำหนดรูปแบบการตัดทิ้งแพ็กเก็ตแบบ D02

3.4.2 การตรวจสอบการทำงานของอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัว

เมื่อแน่ใจแล้วว่าการคำนวณค่า L_b และ L_a ในแบบจำลองสามารถทำได้อย่างถูกต้อง ขั้นตอนต่อไปคือการตรวจสอบการทำงานของอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัว

ของ 2 อัลกอริทึมคือ อัลกอริทึม Bolot และอัลกอริทึม CNR ซึ่งการทดลองในหัวข้อนี้ได้กำหนดให้ใช้เครือข่ายในการจำลอง ดังรูปที่ 3.14 โดยมีการจำลอง 4 ครั้ง (สำหรับ 4 Combination) และเพื่อที่จะตรวจสอบว่าการทำงานของแบบจำลองในส่วนของอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวว่าสามารถทำงานได้อย่างถูกต้องตามวิธีการที่ได้อธิบายในหัวข้อ 3.1 หรือไม่ จึงได้กำหนดรูปแบบการตัดทิ้งแพ็กเก็ตในแต่ละช่วงเวลาให้แตกต่างกัน เพื่อจะได้เห็นว่าการเปลี่ยนแปลง Combination ของข้อมูลซ้ำเป็นไปอย่างถูกต้องหรือไม่ และในการจำลองแต่ละครั้งพฤติกรรมการส่งข้อมูลของโหนด RS และ RR เป็นดังนี้

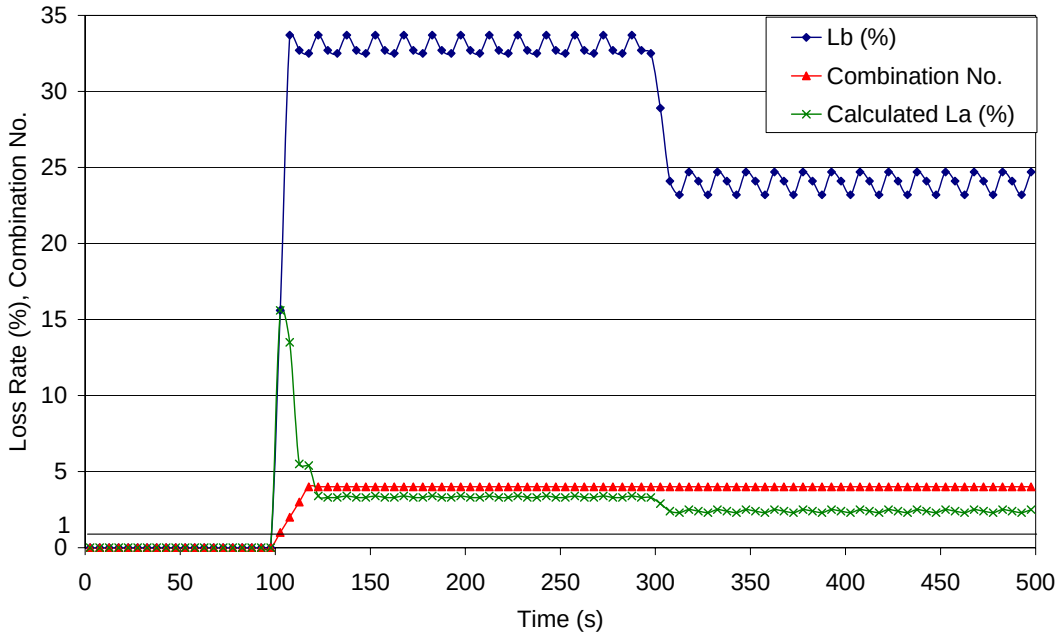
- โหนด RS ส่งแพ็กเก็ตเสียงโดยใช้โปรโตคอล RTP ไปให้โหนด RR โดยส่งแพ็กเก็ตทุกๆ 30 ms ซึ่งเป็นระยะเวลาของเสียงพูด 1 เฟรมสำหรับการบีบอัดเสียงด้วย G.723.1 ส่วนขนาดแพ็กเก็ตขึ้นอยู่กับ Combination ที่ใช้
- โหนด RR จะมีการรายงานค่า L_b และ L_a โดยส่งมาในแพ็กเก็ต Receiver Report ให้กับโหนด RS ซึ่งแพ็กเก็ตนี้มีการส่งทุก 5 วินาที
- โหนด RS เริ่มส่งแพ็กเก็ตเสียงตั้งแต่วินาทีที่ 0 และหยุดที่วินาทีที่ 500
- ในช่วงวินาทีที่ 0 – 100 ไม่มีการตัดทิ้งแพ็กเก็ต
- ในช่วงวินาทีที่ 101 – 300 มีการตัดทิ้งแพ็กเก็ตแบบ D04 (ดูรายละเอียดในตารางที่ 3.5)
- ในช่วงวินาทีที่ 301 – 500 มีการตัดทิ้งแพ็กเก็ตแบบ D05 (ดูรายละเอียดในตารางที่ 3.5)

การทดลองนี้กำหนดให้ทุก Combination ใช้การบีบอัดเสียง G.723.1 ทั้งข้อมูลหลักและข้อมูลซ้ำ เนื่องจากในวิทยานิพนธ์นี้เลือกใช้การบีบอัดเสียง G.723.1 ในการสื่อสารเสียง ประกอบกับการบีบอัดเสียงแบบนี้มีอัตราบิตที่ต่ำอยู่แล้ว (6.3 kbps) และการที่ข้อมูลหลักและข้อมูลเสียงใช้การบีบอัดเสียงที่แตกต่างกันก็ทำให้ผู้ส่งต้องเสียเวลามากขึ้น เนื่องจากต้องมีการเข้ารหัสเสียงในแต่ละเฟรมมากกว่า 1 ครั้ง แต่ถ้าใช้การบีบอัดเสียงชนิดเดียวกันก็สามารถเข้ารหัสเพียงครั้งเดียวและเก็บข้อมูลหลังการบีบอัดเอาไว้เป็นข้อมูลซ้ำในภายหลังได้เลย สำหรับ Combination ของข้อมูลซ้ำที่ใช้ในการทดลองมี 6 Combination ดังแสดงในตารางที่ 3.4 ส่วนค่าเทรสโฮลด์ที่ใช้ในทุกอัลกอริทึมนั้นกำหนดให้เท่ากัน โดยค่าเทรสโฮลด์ HIGH เท่ากับ 5% เนื่องจากว่าหากมีข้อมูลเสียงสูญหายมากกว่า 5% จะทำให้รู้สึกได้ว่าเสียงมีคุณภาพลดลง[6] และกำหนดค่าเทรสโฮลด์ LOW เท่ากับ 1% ซึ่งถือว่าเป็นค่าที่ต่ำพอที่จะลองเพิ่ม Combination ได้แล้ว

3.4.2.1 การตรวจสอบการทำงานของอัลกอริทึม Bolot

รูปที่ 3.16 เป็นกราฟแสดงผลการจำลองโดยใช้อัลกอริทึม Bolot ซึ่งจะเห็นว่าค่า L_b ในแต่ละช่วงเวลาตรงกันกับอัตราการตัดทิ้งแพ็กเก็ตตามที่ได้กำหนดไว้ข้างต้น นั่นคือช่วง 100 วินาทีแรกค่า L_b เป็น 0 % ส่วนช่วงวินาทีที่ 101 – 300 ค่า L_b เฉลี่ยมีค่าประมาณ 33 % ส่วนวินาทีที่ 301 – 500 ค่า L_b เฉลี่ยมีค่าประมาณ 24 % อัลกอริทึม Bolot ไม่ได้ค่าจริงของ L_a แต่คำนวณค่า L_a โดยใช้ค่า L_b

หารด้วยค่า Reward ของ Combination ที่ใช้อยู่ จากกราฟในรูปที่ 3.16 สามารถยืนยันได้ว่าค่า L_a จากการคำนวณของแบบจำลองมีค่าถูกต้อง ตัวอย่างเช่น หลังจากวินาทีที่ 300 เป็นต้นไปมีการใช้ Combination ที่ 4 ซึ่งจากตารางที่ 3.4 ค่า Reward ของ Combination นี้เท่ากับ 10 และในช่วงเวลานี้ค่า L_b เฉลี่ยอยู่ที่ 24% ดังนั้นค่า L_a จากการคำนวณจะต้องอยู่ที่ 2.4 % ซึ่งตรงกับที่ได้แสดงอยู่ในกราฟรูปที่ 3.16



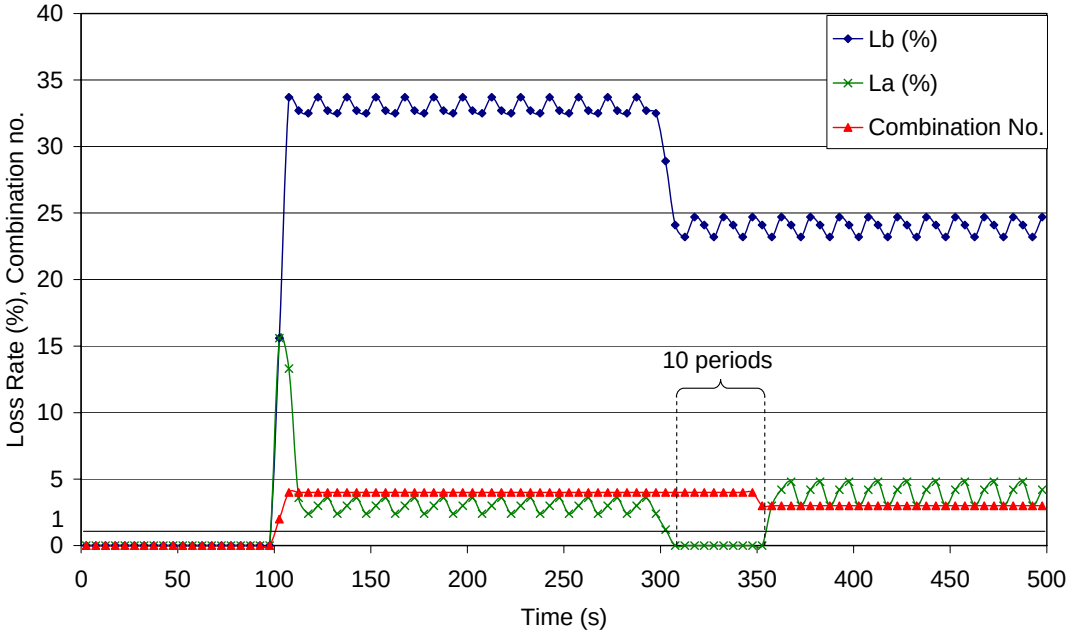
รูปที่ 3.16 ค่า L_b และค่า L_a จากการคำนวณในระหว่างการจำลองโดยใช้อัลกอริทึม Bolot

หลังจากที่ได้ตรวจสอบแล้วค่า L_a จากการคำนวณนั้นได้มาอย่างถูกต้อง ขั้นตอนต่อไปก็คือการตรวจสอบการเปลี่ยน Combination ว่าถูกต้องตาม Pseudo Code ของอัลกอริทึม Bolot ในรูปที่ 3.7 หรือไม่ ซึ่งกราฟในรูปที่ 3.16 ก็สามารถยืนยันได้ว่าแบบจำลองสามารถทำงานได้ถูกต้อง โดยในการทดลองนี้ได้กำหนดให้เทรสโฮลด์ LOW และ HIGH เป็น 1% และ 5% ตามลำดับ จะเห็นได้ว่าหลังจากวินาทีที่ 100 เมื่อค่า L_a (จากการคำนวณ) สูงเกินกว่า 5% จะมีการเพิ่ม Combination ขึ้นทีละหนึ่งขั้นและหยุดที่ Combination ที่ 4 เนื่องว่าค่า L_a ลดลงมาต่ำกว่า 5% และหลังจากวินาทีที่ 300 ถึงแม้ว่าค่า L_a จะลดลงแต่ไม่กียังไม่ต่ำกว่า 1% ดังนั้นจึงไม่มีการลด Combination

3.4.2.2 การตรวจสอบการทำงานของอัลกอริทึม CNR

จากการตรวจสอบการทำงานของแบบจำลองเมื่อมีการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม CNR การเปลี่ยนแปลง Combination เป็นไปตามกราฟในรูปที่ 3.17 ซึ่งจะเห็นว่า

แบบจำลองสามารถทำงานได้ถูกต้องตาม Pseudo Code ของอัลกอริทึม CNR ในรูปที่ 3.12



รูปที่ 3.17 การเปลี่ยนแปลงของ Combination ในระหว่างการจำลองโดยใช้อัลกอริทึม CNR

โดยหลังจากวินาทีที่ 100 ซึ่งเริ่มมีการตัดทิ้งแพ็กเก็ต ได้มีการเพิ่ม Combination ไปหยุดที่ Combination หมายเลข 4 เนื่องจากค่า L_a มีค่าต่ำกว่า 5% แล้ว และหลังจากวินาทีที่ 300 ซึ่งได้มีการลดการตัดทิ้งแพ็กเก็ตลง ค่า L_a จึงมีค่าลดลง และเมื่อค่า L_a ต่ำกว่า 1% เป็นจำนวน 10 ครั้ง ก็ได้มีการลด Combination ลงมาอยู่ที่ Combination หมายเลข 3 ทั้งนี้เนื่องจากในที่นี้ได้กำหนดค่า MIN_DURATION เท่ากับ 10