บทที่ 3

การควบคุมคุณภาพแบบปรับตัวสำหรับการสื่อสารเสียง

หากกล่าวถึงคุณภาพของเสียงแล้ว สามารถจำแนกวิธีการวัดคุณภาพเสียงได้เป็น 2 ประเภทคือ การวัดแบบ Subjective และการวัดแบบ Objective โดยประเภทแรกคือการวัดแบบ Subjective เป็นการวัดที่ใช้มนุษย์เป็นผู้ตัดสินจากเสียงที่ได้ยิน โดยอาจมีการแบ่งระดับ (Rating) ของคุณภาพเสียง เช่น ดีมาก ดี พอใช้ แย่ และแย่มาก เป็นต้น ส่วนการวัดแบบ Objective เป็นการวัดคุณภาพที่สามารถระบุค่าที่วัดเป็นตัวเลขได้ เช่น ความแตกต่างระหว่างสัญญาณเสียงที่ได้ยินกับสัญญาณเสียงต้นฉบับ เป็นต้น แต่การวัดคุณภาพเสียงดังกล่าวต่างก็ต้องใช้เวลานานและไม่เหมาะกับการสื่อสารเสียงที่มีลักษณะเป็นเวลาจริง ดังนั้นการควบคุมคุณภาพเสียงในที่นี้จึงใช้การควบคุมปริมาณการสูญหายของข้อมูลเสียงแทน เพราะการสูญหายของข้อมูลเสียงเป็นสาเหตุสำคัญที่ทำให้คุณภาพเสียงลดลง และเนื่องจากการสื่อสารเสียงบนเครือข่ายอินเตอร์เน็ตไม่สามารถที่จะหลีกเลี่ยงปัญหาการสูญหายของแพ็กเก็ตได้ ดังนั้นจึงจำเป็นที่จะต้องมีกลไกในการลดผลกระทบจากการสูญหายของแพ็กเก็ต ซึ่งในที่นี้จะเรียกกลไกที่ทำหน้าที่นี้ว่า การควบคุมความผิดพลาด (Error Control)

เนื้อหาของบทนี้เริ่มต้นด้วยการกล่าวถึงอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัว (Adaptive Error Control) ในหัวข้อที่ 3.1 โดยมีการอธิบายถึงอัลกอริทึมที่มีการเสนอขึ้นมาก่อนหน้านี้ แต่อัลกอริทึมเหล่านี้ก็ยังมีจุดบกพร่องอยู่บ้าง วิทยานิพนธ์นี้จึงได้เสนอวิธีการควบคุมความผิดพลาดแบบปรับตัวขึ้นมาใหม่ชื่อว่าอัลกอริทึม CNR โดยการปรับปรุงจากข้อบกพร่องของอัลกอริทึมที่ได้เสนอก่อนหน้านี้ โดยรายละเอียดของอัลกอริทึมใหม่นี้อยู่ในหัวข้อที่ 3.2 และเนื่องจากมีการใช้แบบจำลองในการประเมินผลอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวแต่ละอัลกอริทึม จึงต้องมีการตรวจสอบความถูกต้องของแบบจำลองที่ได้จัดทำขึ้น ซึ่งเนื้อหาส่วนนี้อยู่ในหัวข้อที่ 3.3 ส่วนในหัวข้อที่ 3.4 เป็นการทดลองเพื่อประเมินผลอัลกอริทึม CNR และได้นำเสนอผลการทดลองในหัวข้อที่ 3.5 สุดท้ายคือหัวข้อที่ 3.6 เป็นการสรุปเนื้อหาของบทนี้

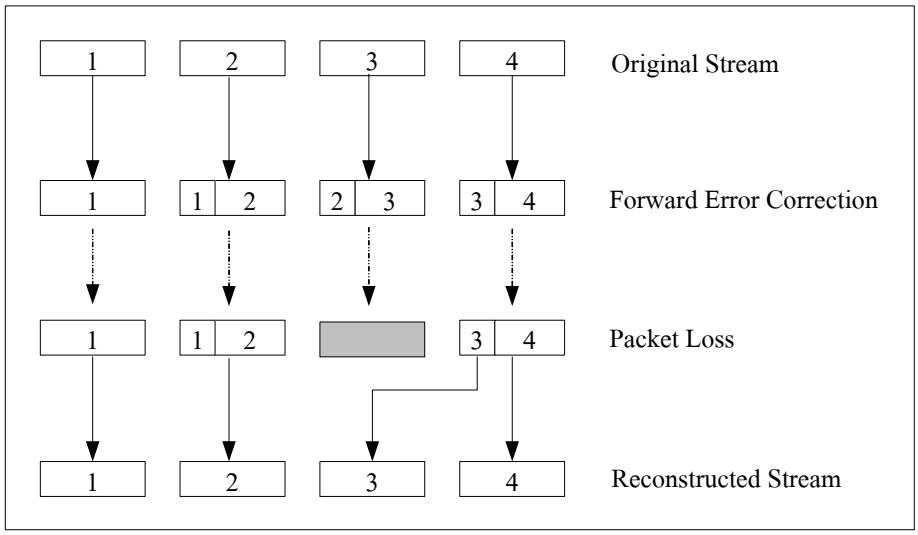
3.1 การควบคุมความผิดพลาดแบบปรับตัว

ในการสื่อสารเสียงบนเครือข่ายที่ไม่มีการรับประกันคุณภาพการบริการอย่างเช่นเครือข่ายอินเตอร์เน็ตนั้น ปัญหาอย่างหนึ่งที่ส่งผลกระทบต่อคุณภาพของเสียงก็คือการสูญหายของแพ็กเก็ต ถึงแม้ว่าในการสื่อสารเสียงนั้นไม่จำเป็นจะต้องได้รับข้อมูลครบทั้งหมดก็ตาม แต่การที่มีแพ็กเก็ต

สูญหายมากเกินไปก็ทำให้คุณภาพเสียงต่ำเกินกว่าที่ยอมรับได้ ดังนั้นจึงจำเป็นต้องมีกลไกในการลดผลกระทบจากการสูญหายของแพ็กเก็ต ซึ่งกลไกดังกล่าวก็คือ การควบคุมความผิดพลาด ซึ่ง Forward Error Correction (FEC) ก็เป็นวิธีการควบคุมความผิดพลาดวิธีการหนึ่งที่สามารถลดผลกระทบจากการสูญหายของแพ็กเก็ตได้โดยไม่เพิ่มค่าเวลาหน่วงมากนัก[16] ซึ่งคุณสมบัตินี้เหมาะสมสำหรับการสื่อสารเสียงที่ต้องการความเป็นเวลาจริง

FEC สามารถแบ่งออกได้เป็น 2 ประเภทคือ FEC ที่ไม่ขึ้นกับสื่อ (Media Independent FEC) และ FEC ที่ขึ้นกับสื่อ (Media Specific FEC) โดยประเภทแรกคือ FEC ที่ไม่ขึ้นกับสื่อ มีหลักการคือ เมื่อมีการส่งแพ็กเก็ตได้หนึ่งชุด จะมีการส่งแพ็กเก็ตพิเศษเพิ่มเข้าไป โดยแพ็กเก็ตพิเศษนี้ได้จากการกระทำทางคณิตศาสตร์ของข้อมูลจากทุกแพ็กเก็ตที่อยู่ในชุดเดียวกัน ดังนั้นเมื่อมีแพ็กเก็ตใดแพ็กเก็ตหนึ่งสูญหายไป ข้อมูลในแพ็กเก็ตที่สูญหายสามารถกู้คืนได้โดยใช้แพ็กเก็ตพิเศษนี้ แต่วิธีการนี้ใช้ไม่ได้ผลถ้ามีแพ็กเก็ตในชุดเดียวกันสูญหายมากกว่า 1 แพ็กเก็ต

FEC ประเภทที่สองคือ FEC ที่ขึ้นกับสื่อมีหลักการคือ ในแต่ละแพ็กเก็ตที่ผู้ส่งจะส่งไปให้ผู้รับ จะมีการบรรจุข้อมูลซ้ำ (Redundancy) ของแพ็กเก็ตที่อยู่ก่อนหน้า แต่อาจใช้การบีบอัดด้วยอัตราบิตที่ต่ำกว่าเพื่อไม่ให้ปริมาณการใช้แบนด์วิดท์เพิ่มขึ้นมากเกินไป จากรูปที่ 3.1 จะเห็นว่าเมื่อแพ็กเก็ตที่ 3 สูญหาย ข้อมูลเสียงที่อยู่ในแพ็กเก็ตนี้สามารถกู้คืนได้จากข้อมูลซ้ำที่อยู่ในแพ็กเก็ตที่ 4 ซึ่ง FEC ประเภทที่สองนี้เป็นวิธีการที่ได้รับความนิยมมากกว่า FEC ประเภทแรก และมีมาตรฐานที่กำหนดรูปแบบของการบรรจุข้อมูลซ้ำของเสียงลงในแพ็กเก็ต RTP อีกด้วย ซึ่งมาตรฐานดังกล่าวก็คือ RFC2198 [17] แต่มาตรฐานนี้ก็กำหนดเพียงแค่รูปแบบของแพ็กเก็ตเท่านั้น แต่ไม่ได้กำหนดว่าต้องใช้ปริมาณข้อมูลซ้ำเท่าใด และข้อมูลซ้ำแต่ละชุดใช้วิธีการบีบอัดเสียงแบบใด



รูปที่ 3.1 Media Specific FEC

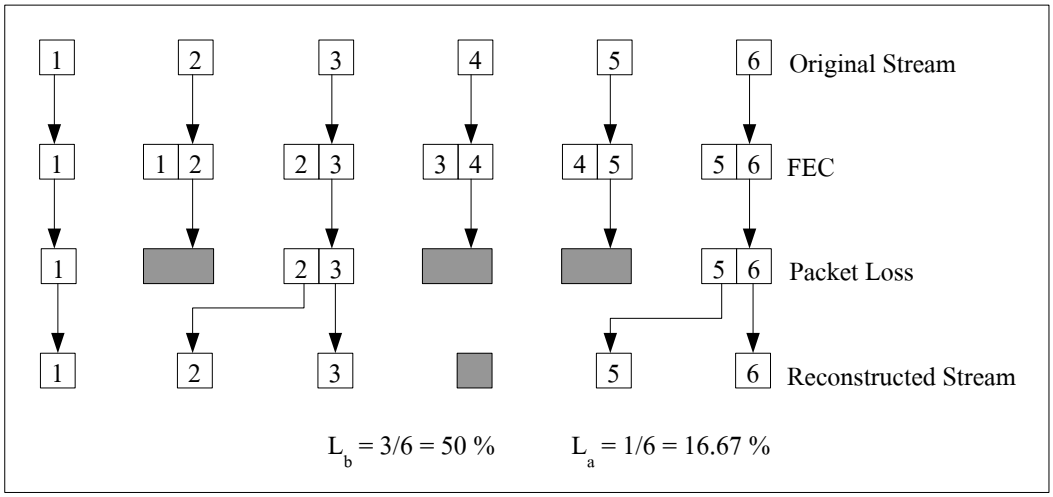
เนื่องจากการควบคุมความผิดพลาดโดยใช้ FEC มีการส่งข้อมูลซ้ำของเสียงซึ่งทำให้ปริมาณการใช้แบนด์วิดท์ของสื่อสารเสียงเพิ่มขึ้น ดังนั้นจึงควรกำหนดปริมาณข้อมูลซ้ำให้เหมาะสมกับสภาพของเครือข่าย ในกรณีที่มีการสูญหายของแพ็กเก็ตเพียงเล็กน้อย หากกำหนดปริมาณข้อมูลซ้ำมากเกินไปจะเป็นการสิ้นเปลืองแบนด์วิดท์โดยไม่จำเป็น ดังนั้นในการสื่อสารเสียงจึงควรมีอัลกอริทึมในการกำหนดรูปแบบและปริมาณของข้อมูลซ้ำที่สามารถปรับตัวได้ตามปริมาณการสูญหายของแพ็กเก็ต ซึ่งอัลกอริทึมที่ได้มีการเสนอก่อนหน้านี้ได้แก่ อัลกอริทึม Bolot[18], อัลกอริทึม USF[19] และอัลกอริทึม RCCS[20] อัลกอริทึมทั้งสามนี้ล้วนแต่ใช้พื้นฐานของ FEC ประเภทที่ขึ้นกับสื่อ แต่สิ่งที่แตกต่างกันก็คือวิธีการที่จะเลือกว่าเมื่อใดควรที่จะเลือกรูปแบบของข้อมูลซ้ำแบบใดและใช้ปริมาณข้อมูลซ้ำเท่าใด

3.1.1 อัตราการสูญหายของข้อมูลเสียงเมื่อมีการใช้ FEC

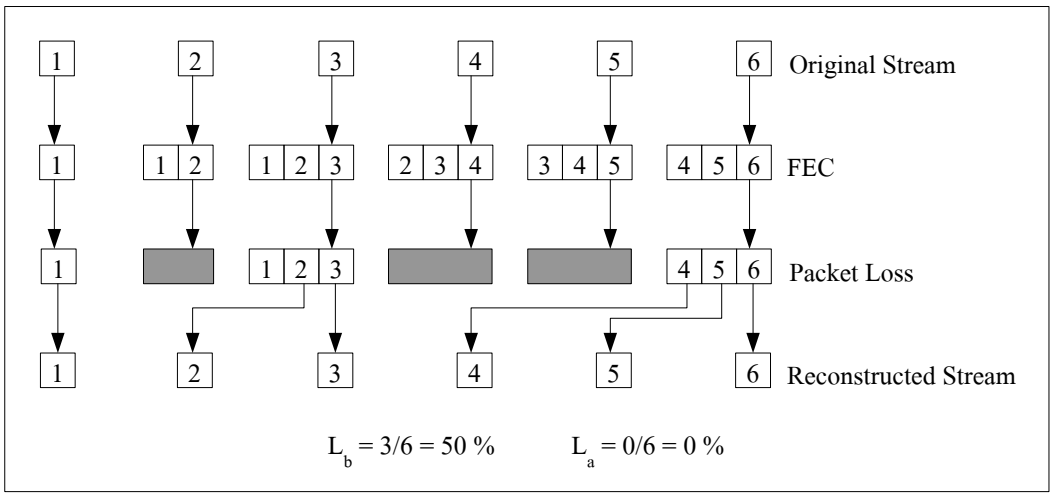
ก่อนจะกล่าวถึงอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวในแต่ละอัลกอริทึมในหัวข้อนี้จะอธิบายเกี่ยวกับค่าอัตราการสูญหายของข้อมูลเมื่อมีการใช้เทคนิค FEC ทั้งนี้เพื่อให้สามารถเข้าใจถึงรายละเอียดของอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวที่จะกล่าวถึงในหัวข้อถัดไปได้ง่ายขึ้น ซึ่งเมื่อมีการใช้เทคนิค FEC แล้ว ค่าอัตราการสูญหายของข้อมูลที่จะต้องพิจารณามีอยู่ 2 ค่าคือ ค่าอัตราการสูญหายก่อนการจัดเรียง (Loss Rate before Reconstruction) ซึ่งในที่นี้จะแทนด้วยสัญลักษณ์ L_b และค่าอัตราการสูญหายหลังการจัดเรียง (Loss Rate after Reconstruction) ซึ่งในที่นี้จะแทนด้วยสัญลักษณ์ L_a

ค่า L_b หมายถึงค่าอัตราการสูญหายของข้อมูลเสียงโดยที่ยังไม่ได้นำข้อมูลซ้ำมาจัดเรียงใหม่ หรืออาจจะกล่าวได้ว่าค่า L_b คืออัตราการสูญหายของแพ็กเก็ตนั่นเอง จากรูปที่ 3.2 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 เนื่องจากอาจมีการกู้ข้อมูลในแพ็กเก็ตที่สูญหายบางส่วนได้จากข้อมูลซ้ำ จากรูปที่ 3.2 จะเห็นว่าเมื่อมีการจัดเรียงข้อมูลแล้วมีข้อมูลเสียงสูญหายไปเพียง 1 เฟรมเท่านั้นคือเฟรมที่ 4 ดังนั้นค่า L จึงมีค่าเท่ากับ 1/6 หรือ 16.67 % ซึ่งค่า L_a นี้เองจะเป็นตัวบอกถึงปริมาณข้อมูลเสียงที่สูญหาย ถ้า L_a มีค่ามากจะทำให้เสียงที่ได้รับมีคุณภาพต่ำ ส่วนค่า L_b เป็นค่าที่บอกถึงปริมาณการสูญหายของแพ็กเก็ตเท่านั้น

การเพิ่มจำนวนชุดของข้อมูลซ้ำสามารถลดค่า L_a ได้ถึงแม้ว่าค่า L_b จะมีค่าเท่าเดิม จากรูปที่ 3.3 จะเห็นว่าเมื่อเพิ่มจำนวนข้อมูลซ้ำเป็น 3 ชุด สามารถกู้คืนข้อมูลเสียงที่สูญหายได้หมด ทำให้ค่า L_a เท่ากับ 0 % ดังนั้นในกรณีที่เครือข่ายมีความมีคับคั่งสูงและมีปริมาณแพ็กเก็ตสูญหายมากจึงควรจะมีเพิ่มจำนวนชุดข้อมูลซ้ำ เพื่อให้สามารถกู้ข้อมูลเสียงในแพ็กเก็ตที่สูญหายได้มากขึ้น และสามารถรักษาระดับของอัตราการสูญหายของข้อมูลเสียงไม่ให้สูงเกินไปได้



รูปที่ 3.2 การกู้คืนโดยใช้ข้อมูลซ้ำสามารถทำให้ค่า L_a น้อยกว่า L_b



รูปที่ 3.3 การเพิ่มจำนวนข้อมูลซ้ำสามารถช่วยลดค่า L_a ได้

3.1.2 อัลกอริทึม Bolot

อัลกอริทึมนี้ได้รับการเสนอโดย Bolot และได้รับการตีพิมพ์ลงใน [18] ซึ่งในบทความนี้ได้มีการกำหนดรูปแบบของการจัดวางข้อมูลซ้ำ (Combination) เอาไว้หลายแบบ และได้ทำการทดลองเพื่อหาค่า Reward ของแต่ละ Combination โดยค่า Reward คืออัตราส่วนระหว่างค่า L_b และค่า L_a

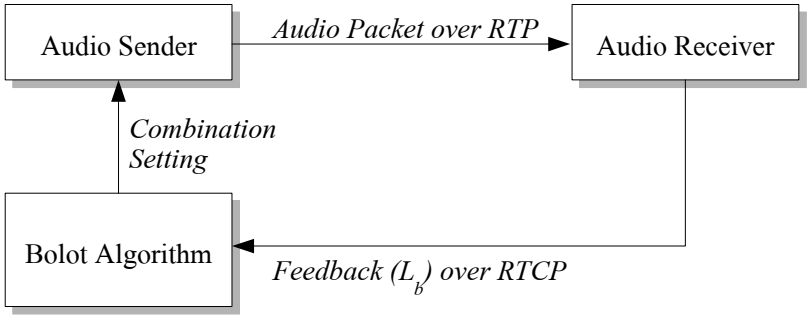
$$\text{Reward} = L_b / L_a$$

ตารางที่ 3.1 ตาราง 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

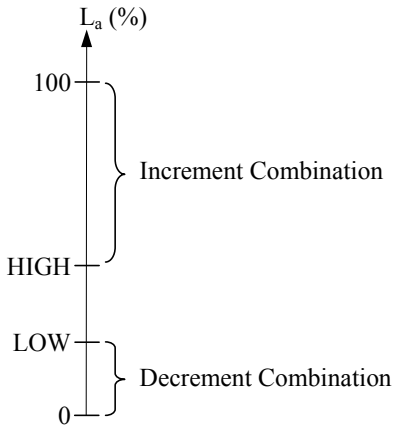
ค่า Reward ของแต่ละ Combination ที่ปรากฏในตารางที่ 3.1 เป็นค่าที่ได้จากการทดลองโดยใช้เครือข่ายระหว่างสถาบัน INRIA ในประเทศฝรั่งเศสกับ University College London (UCL) ในประเทศอังกฤษ สำหรับสัญลักษณ์ (PCM) หมายถึง ทุกแพ็กเก็ตใช้การบีบอัดแบบ PCM โดยไม่มีการบรรจุข้อมูลซ้ำ ส่วนสัญลักษณ์ (PCM, ADM4(1)) หมายถึง 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 3.4 จะเห็นว่าผู้รับจะมีการรายงานเฉพาะค่า L_b ให้ผู้ส่งได้ทราบเท่านั้นโดยจะระบุค่าของ L_b ลงฟิลด์ Fraction Lost ของแพ็กเก็ต RTCP ชนิด Receiver Report ซึ่งปกติฟิลด์ Fraction Lost นี้จะใช้ในการระบุค่าอัตราการสูญหายของแพ็กเก็ต และเนื่องจากค่า L_b มีค่าเท่ากับอัตราการสูญหายของแพ็กเก็ตจึงสามารถบรรจุลงในฟิลด์นี้ได้เลย 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 3.4 แผนภาพการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม Bolot

จากรูปที่ 3.5 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 [18] ได้กำหนดให้ค่าเทรสโฮลด์ HIGH และ LOW มีค่าเท่ากันคือ 3% เพื่อควบคุมให้ค่า L_a อยู่ที่ 3% แต่การกำหนดค่าเทรสโฮลด์ทั้งสองให้มีค่าเท่ากันจะทำให้มีการเปลี่ยน Combination บ่อยครั้ง เพราะไม่มีช่วงของค่า L_a ช่วงใดเลยที่อัลกอริทึมนี้ไม่ต้องมีการเปลี่ยนแปลง Combination



รูปที่ 3.5 การเพิ่มและลด Combination ในอัลกอริทึม Bolot

หลักการทำงานของอัลกอริทึม Bolot สามารถสรุปเป็น Pseudo Code ได้ดังรูปที่ 3.6 โดยขั้นตอนที่ 1 เป็นคำนวณค่า L_b จากค่าที่อ่านได้จากแพ็กเก็ต Recevier Report โดยค่าของ L_b จะระบุอยู่ในในฟิลด์ Fraction Lost ซึ่งปกติแล้วการระบุค่าอัตราการสูญหายในแพ็กเก็ตในฟิลด์

Fraction Lost จะไม่ระบุเป็นค่าทศนิยมหรือค่าที่เป็นเปอร์เซ็นต์โดยตรง แต่จะระบุเป็นอัตราส่วนเมื่อเทียบกับ 256[10] 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

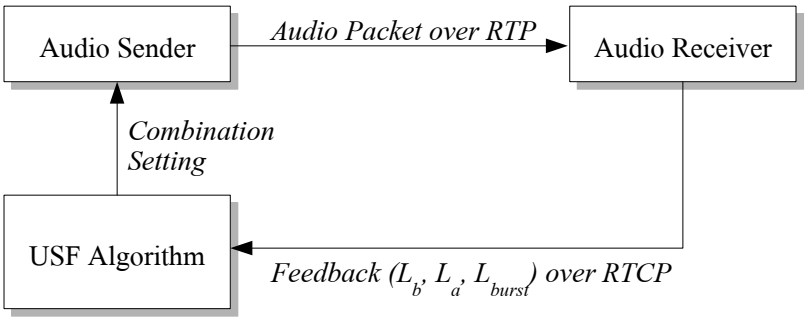
```
For each RTCP packet received do
   1. Calculate loss rate before reconstruction, Lb
   2. Calculate loss rate after reconstruction
      La = Lb / Reward associated with current combination number
   3. If (La > HIGH) then
         Increment combination
   4. If (La < LOW) then
         Decrement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

รูปที่ 3.6 Pseudo Code อัลกอริทึม Bolot

3.1.3 อัลกอริทึม USF

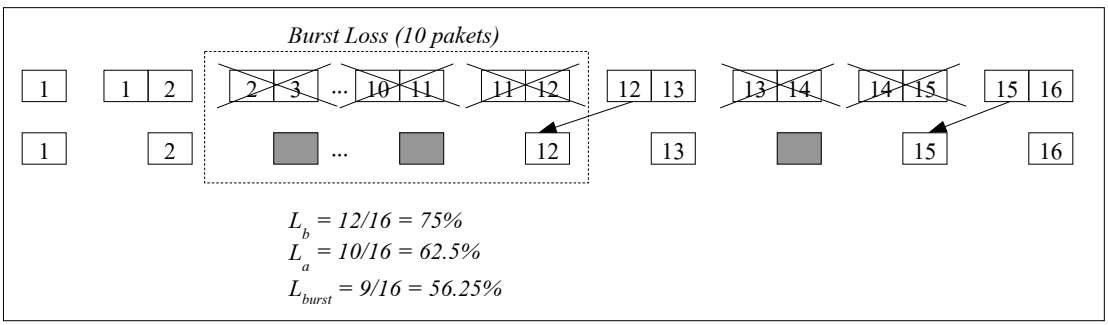
อัลกอริทึม USF ได้มีการเสนอใน [19] ซึ่งชื่อของอัลกอริทึมนี้คาดว่าน่าจะย่อมาจาก University of South Florida ซึ่งเป็นสถาบันการศึกษาของผู้เขียนบทความดังกล่าว อัลกอริทึม USF ได้มีการปรับปรุงอัลกอริทึม Bolot โดยได้ระบุว่าอัลกอริทึม Bolot มีข้อเสียอยู่สองประการ โดยประการแรกคือการที่อัลกอริทึม Bolot คำนวณค่า L_a โดยใช้ค่า Reward ที่ได้จากการทดลอง ซึ่งค่านี้ไม่ได้ตรงตามความเป็นจริงเสมอไป ดังนั้นอัลกอริทึม USF จึงเลือกใช้ค่าจริงของ L_a แทนการใช้ค่าจากการคำนวณโดยใช้ Reward ข้อเสียประการที่สองของอัลกอริทึม Bolot คือ ทำให้เกิด Cyclical Behavior เนื่องจากว่าเมื่อค่า L_a สูงกว่าค่าเทรสโฮลด์ HIGH จะมีการเพิ่ม Combination ขึ้นหนึ่งขั้น ทำให้การสูญหายของข้อมูลมีน้อยลง และเมื่อได้รับแพ็กเก็ต Receiver Report ในรอบถัดมาปรากฏว่า ค่า L_a ลดลงจนเหลือน้อยกว่าเทรสโฮลด์ LOW ส่งผลให้มีการลด Combination ลงหนึ่งขั้น ซึ่งการใช้ Combination ในขั้นที่ต่ำลงมานี้อาจทำให้ไม่สามารถกู้ข้อมูลที่สูญหายได้เพียงพอและทำให้ค่า L_a กลับมาสูงเกินค่าเทรสโฮลด์ HIGH อีกครั้ง และจะเป็นเช่นนี้ไปเรื่อย ๆ ในวิธีการของ USF จึงแก้ไขปัญหานี้โดยการเปรียบเทียบความแตกต่างของค่า L_b ในรอบปัจจุบันกับค่า L_b ในรอบก่อนหน้าว่ามีความแตกต่างเพียงพอหรือไม่ จากนั้นจึงค่อยตัดสินใจลด Combination แต่ความเป็นจริงแล้วปัญหา Cyclical Behavior ของอัลกอริทึม Bolot มีสาเหตุมาจากการที่ใช้ค่าเทรสโฮลด์ HIGH และ LOW ใกล้เคียงกันมากเกินไป และโดยเฉพาะใน [19] ได้กำหนดให้ค่า HIGH และ LOW เท่ากัน ดังนั้นย่อมจะต้องเกิดปัญหา Cyclical Behavior ขึ้นอย่างแน่นอน การแก้ปัญหานี้ก็สามารถทำได้โดยกำหนดค่าเทรสโฮลด์ HIGH และ LOW ให้แตกต่างกันพอที่จะมีช่วงที่ไม่ต้องมีการเปลี่ยน Combination

นอกจากนี้อัลกอริทึม USF ยังได้พิจารณาถึงกรณีของ Burst Loss ซึ่งได้กำหนดเอาไว้ว่า Burst Loss หมายถึง การสูญหายของแพ็กเก็ตติดกันตั้งแต่ 10 แพ็กเก็ตขึ้นไป โดยจะไม่มีการนำการสูญหายของแพ็กเก็ตในส่วนที่จัดเป็น Burst Loss มาพิจารณาในการตัดสินใจเพิ่ม Combination เนื่องจากว่าหากมีการเกิด Burst Loss แล้ว การใช้ Combination ในขั้นที่สูงขึ้นก็ไม่สามารถกู้ข้อมูลที่สูญหายกลับมาได้หมด จากหลักการทำงานของอัลกอริทึม USF สามารถสรุปเป็นแผนภาพได้ดังรูปที่ 3.7 ซึ่งจะเห็นว่าสิ่งที่ผู้รับจะต้องรายงานให้ผู้ส่งได้ทราบมีอยู่ 3 ค่า คือ L_b, L_a และ L_{burst} ผ่านทางแพ็กเก็ต RTCP ชนิด Receiver Report โดยค่า L_b นั้นจะถูกบรรจุอยู่ในฟิลด์ Fraction Lost ส่วนค่า L_a และ L_{burst} จะถูกบรรจุอยู่ในส่วนขยายเฮดเดอร์



รูปที่ 3.7 แผนภาพการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม USF

โดย L_{burst} หมายถึงอัตราการสูญหายของข้อมูลเสียงหลังการจัดเรียงโดยพิจารณาเฉพาะส่วนที่อยู่ใน Burst Loss เท่านั้น จากตัวอย่างในรูปที่ 3.8 จะเห็นว่ามีแพ็กเก็ตที่สูญหายในส่วนที่เป็น Burst Loss อยู่ 10 แพ็กเก็ต และหลังจากการจัดเรียงใหม่แล้วสามารถกู้ข้อมูลเสียงที่สูญหายมาได้หนึ่งเฟรมคือเฟรมที่ 12 ดังนั้นข้อมูลเสียงที่สูญหายในส่วนของ Burst Loss มี 9 เฟรม และเนื่องจากข้อมูลเสียงทั้งหมดมี 16 เฟรม ดังนั้น L_{burst} จึงมีค่าเท่ากับ 9/16 หรือ 56.25% นั่นเอง



รูปที่ 3.8 ตัวอย่างการคำนวณค่า L_b, L_a และ L_{burst}

สำหรับ Pseudo Code ของอัลกอริทึม USF ได้แสดงไว้ในรูปที่ 3.9 ในขั้นตอนที่ 1 และขั้นตอนที่ 2 เป็นการคำนวณค่า L_a และ L_b ตามลำดับ ในขั้นตอนที่ 3 และ 4 เป็นการตัดสินใจว่าควรจะเพิ่ม Combination หรือไม่โดยพิจารณาจากค่า L_a ถ้าหาก L_a สูงกว่าเทรสโฮลด์ HIGH ก็ให้ลดค่า L_a ด้วย L_{burst} เพื่อตัดการสูญหายของข้อมูลในส่วนที่อยู่ใน Burst Loss จากนั้นนำ L_a ที่ได้มาพิจารณาอีกครั้ง ถ้า L_a ยังมีค่ามากกว่าเทรสโฮลด์ HIGH อยู่ก็ให้เพิ่ม Combination ขึ้นหนึ่งขั้น และในขั้นตอนที่ 5-7 เป็นการตัดสินใจว่าจะลด Combination หรือไม่ โดยการลด Combination จะเกิดขึ้นก็ต่อเมื่อ L_a มีค่าน้อยกว่าเทรสโฮลด์ LOW และค่าความแตกต่างของ L_b ในรอบปัจจุบันและรอบก่อนหน้ามีค่ามากกว่า MINIMUM_THRESHOLD ซึ่งเงื่อนไขหลังนี้ใช้เพื่อให้แน่ใจว่าเครือข่ายมีความคับคั่งลดลงจริง แต่แนวคิดนี้อาจไม่ได้ถูกต้องเสมอไปในทุกสภาพแวดล้อม

```
For each RTCP packet received do
   1. Calculate loss rate after reconstruction, La
   2. Calculate loss rate before reconstruction, Lb
   3. If (La > HIGH) then
         La = La - Lburst
   4. If (La > HIGH) then
         Increment combination
   5. If (La < LOW) then
         Loss difference = Lb(previous) - Lb
   6. If (Loss difference > MINIMUM_THRESHOLD) then
         Decrement combination
   7. Set Lb(previous) = Lb
```

รูปที่ 3.9 Pseudo Code ชองอัลกอริทึม USF

ตารางที่ 3.2 ตาราง Combination ของข้อมูลซ้ำที่ใช้ในอัลกอริทึม USF

Combination No.	Combination Format
0	-
1	-1
2	-2
3	-1-2
4	-1-3
5	-1-2-3
6	-1-2-4
7	-1-3-4
8	-1-2-3-4

สำหรับ Combination ของข้อมูลซ้ำที่ใช้ในอัลกอริทึม USF มี 9 Combination ดังที่แสดงในตารางที่ 3.5 โดยในคอลัมน์ Combination Format ได้กำหนดสัญลักษณ์เป็นจำนวนเต็มลบเพื่อใช้แทนรูปแบบของข้อมูลซ้ำ ตัวอย่างเช่น สัญลักษณ์ -1 หมายความว่า ในแต่ละแพ็กเก็ตลำดับที่ N จะมีข้อมูลซ้ำของแพ็กเก็ตลำดับที่ N-1 อยู่ และสัญลักษณ์ -1-2 หมายความว่า ในแต่ละแพ็กเก็ต N จะมีข้อมูลซ้ำของแพ็กเก็ต N-1 และ N-2 อยู่ สำหรับสัญลักษณ์ของ Combination อื่น ๆ ก็สามารถแปลความหมายได้ในทำนองเดียวกันนี้ แต่ใน [19] ไม่ได้มีการระบุว่าข้อมูลซ้ำแต่ละชุดใช้วิธีการบีบอัดเสียงชนิดใดบ้าง

3.1.4 อัลกอริทึม RCCS

อัลกอริทึม RCCS[20] ซึ่งย่อมาจาก Redundant Codec Combination Selection เป็นอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวที่มีการเลือก Combination โดยใช้พารามิเตอร์สองค่าคือ อัตราการสูญหายของข้อมูลและค่าเวลาหน่วงระหว่างปลายทาง (End-to-End Delay) ซึ่งแตกต่างจากอัลกอริทึม Bolot และ USF ที่ใช้เพียงอัตราการสูญหายของข้อมูลเพียงอย่างเดียว ตารางที่ 3.3 แสดง Combination ที่ใช้ในอัลกอริทึม RCCS ซึ่งจะเห็นว่ารูปแบบของการจัดวางข้อมูลซ้ำ รวมทั้งชนิดของการบีบอัดเสียงในแต่ละ Combination นั้นแตกต่างไปจากอัลกอริทึม Bolot และ USF ในคอลัมน์ Reward แสดงค่า Reward ของแต่ละ Combination ซึ่งค่าเหล่านี้ได้มาจากค่า Reward ของ Bolot นั่นเอง โดยค่า Reward ของแต่ละ Combination ในตารางที่ 3.3 ของอัลกอริทึม RCCS นำมาจากค่า Reward ของ Combination ในตารางที่ 3.1 ของอัลกอริทึม Bolot ที่มีจำนวนข้อมูลซ้ำและตำแหน่งของข้อมูลซ้ำตรงกัน โดยไม่สนใจชนิดของการบีบอัดเสียง แต่ค่า Reward ในตารางที่ 3.3 นี้เป็นเพียงค่า Reward เริ่มต้นเท่านั้น เพราะอัลกอริทึม RCCS จะมีการเปลี่ยนแปลงค่า Reward ของ Combination ที่กำลังใช้งานทุกครั้งที่ได้รับแพ็กเก็ต Receiver Report และในคอลัมน์ขวาสุดคือคอลัมน์ Penalty เป็นค่าอัตราส่วนระหว่างค่าเวลาหน่วงระหว่างปลายทางหลังการจัดเรียงข้อมูลและค่าเวลาหน่วงระหว่างปลายทางก่อนการจัดเรียงข้อมูลซ้ำ

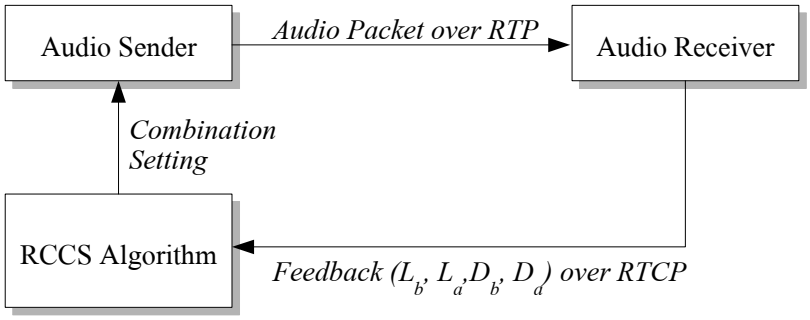
$$\text{Penalty} = \frac{\text{End-to-End Delay after Reconstruction}}{\text{End-to-End Delay before Reconstruction}}$$

ซึ่งค่า Penalty ที่อยู่ในตารางที่ 3.3 เป็นค่าเฉลี่ยที่ได้จากการทดลองหลาย ๆ ครั้ง แต่ใน [20] ก็ไม่ได้อธิบายเอาไว้อย่างชัดเจนว่าการหาค่าเวลาหน่วงก่อนและหลังการจัดเรียงมีวิธีการอย่างไร ซึ่งการหาค่าเวลาหน่วงระหว่างปลายทางนั้นทำได้ยาก เนื่องจากว่านาฬิกาของเครื่องผู้ส่งและเครื่องผู้รับไม่ประสานกัน จากแผนภาพการควบคุมความผิดพลาดแบบปรับตัวของ

อัลกอริทึม RCCS ในรูปที่ 3.10 จะเห็นว่าค่าที่ผู้รับจะต้องรายงานให้กับผู้ส่งมีอยู่ 4 ค่า คือ L_b, L_a, D_b (เวลาหน่วงระหว่างปลายทางก่อนการจัดเรียงข้อมูล) และ D_a (เวลาหน่วงระหว่างปลายทางหลังการจัดเรียงข้อมูล)

ตารางที่ 3.3 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 3.10 แผนภาพการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม RCCS

รูปที่ 3.11 เป็น Pseudo Code ของอัลกอริทึม RCCS ซึ่งจะเห็นว่ามีการใช้พารามิเตอร์หลายตัว โดยคำอธิบายของพารามิเตอร์แต่ละตัวนั้นอยู่ในตารางที่ 3.4 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

$$R_i = \alpha(L_b / L_a) + (1-\alpha)R_{i-1}$$
$$P_i = \alpha(D_a / D_b) + (1-\alpha)P_{i-1}$$

ตารางที่ 3.4 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 3.11 Pseudo Code ของอัลกอริทึม RCCS

การแก้ไขค่า Reward นี้เพื่อให้ค่า Reward มีค่าเปลี่ยนแปลงไปตามสภาพเครือข่ายจริง เนื่องจากว่าค่า Reward ที่ได้จากการทดลองของ Bolot อาจจะไม่ได้ถูกต้องเสมอไปในทุกสภาพแวดล้อม จากนั้นในขั้นตอนที่ 4 เป็นการเลือก Combination โดยจะพิจารณาจากตารางที่ 3.5 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

3.1.5 จุดบกพร่องของอัลกอริทึม Bolot, USF และ RCCS

ทั้งอัลกอริทึม Bolot, อัลกอริทึม USF และอัลกอริทึม RCCS ต่างก็มีวิธีการที่คล้ายคลึงกันก็คือ มีการสร้างตาราง Combination เอาไว้ก่อน สิ่งที่แตกต่างกันก็คือวิธีการที่จะเป็นตัวกำหนดว่าเมื่อใดควรจะเพิ่มหรือลด Combination ซึ่งอัลกอริทึม Bolot และ USF พิจารณาจากค่า L_a ส่วนอัลกอริทึม RCCS นั้นพิจารณาจากทั้ง L_a และค่าเวลาหน่วงระหว่างปลายทาง การที่จะวัดว่าอัลกอริทึมใดดีกว่ากัน ต้องดูว่าอัลกอริทึมเหล่านี้สามารถเลือกใช้ Combination ที่สามารถรักษาระดับของ L_a ไม่ให้สูงเกินไปได้หรือไม่ รวมทั้งต้องไม่ใช้ปริมาณข้อมูลซ้ำมากเกินความจำเป็นอีกด้วย เพราะถ้าไม่คำนึงถึงปริมาณแบนด์วิดท์ที่เพิ่มขึ้นจากการใช้ข้อมูลซ้ำแล้ว ก็ไม่จำเป็นจะต้องใช้อัลกอริทึมใดเลยก็ได้ เพราะยิ่งใช้ข้อมูลซ้ำมากก็ยิ่งสามารถกู้ข้อมูลที่สูญหายได้มาก แต่การใช้อัลกอริทึมในการควบคุมความผิดพลาดเหล่านี้ก็เพื่อที่จะกำหนดให้ใช้ปริมาณข้อมูลซ้ำเท่าที่จำเป็นเท่านั้นและเพื่อให้ใช้แบนด์วิดท์ได้อย่างคุ้มค่า แต่อย่างไรก็ตามอัลกอริทึมทั้งสามก็ยังมีจุดบกพร่องอยู่ โดยจุดบกพร่องของแต่ละอัลกอริทึมมีดังนี้

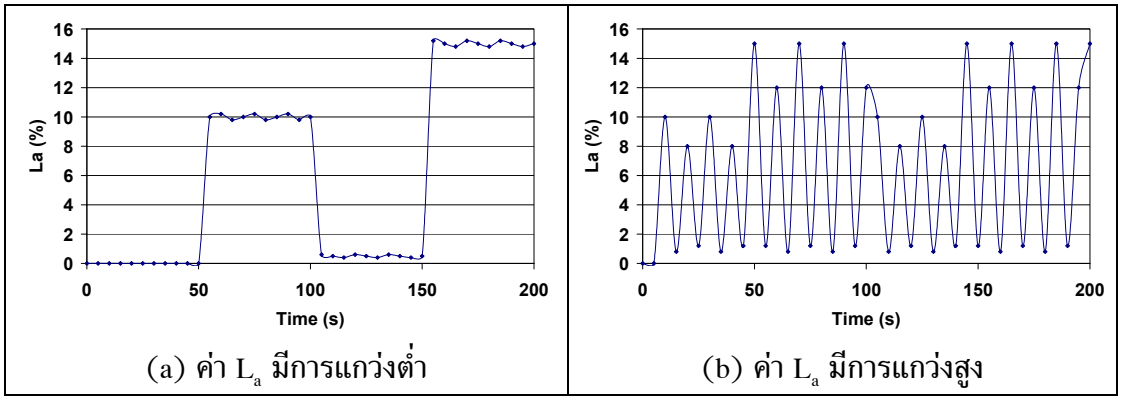
3.1.5.1 จุดบกพร่องของอัลกอริทึม Bolot

ในการตัดสินใจเพิ่มหรือลด Combination ของอัลกอริทึม Bolot นั้น ใช้พิจารณาค่า L_a แต่ค่า L_a ที่ใช้ไม่ใช่ค่าจริง แต่เป็นค่า L_a ที่ได้จากการคำนวณโดยการนำค่า L_b หารด้วยค่า Reward ของ Combination ปัจจุบัน ซึ่งค่า L_a จากการคำนวณดังกล่าวอาจจะไม่ถูกต้องเสมอไปในทุกสภาพแวดล้อม และอาจส่งผลให้อัลกอริทึมนี้ทำงานผิดพลาดได้

3.1.5.2 จุดบกพร่องของอัลกอริทึม USF

อัลกอริทึม USF ได้แก้ไขจุดบกพร่องของอัลกอริทึม Bolot โดยเปลี่ยนมาใช้ค่าจริง L_a แทนค่าที่ได้จากการคำนวณ แต่เมื่อใช้ค่าจริงของ L_a ปัญหาที่ตามก็คือ ถ้าค่า L_a มีการแกว่ง (Oscillation) มาก อัลกอริทึมนี้อาจจะทำงานผิดพลาดได้ ซึ่งการแกว่งของ L_a ย่อมเกิดขึ้นได้ในกรณีที่เครือข่ายคับคั่ง รูปที่ 3.12 แสดงให้เห็นลักษณะการแกว่งของค่า L_a ซึ่งหากค่า L_a มีการแกว่งต่ำก็จะไม่ส่งผลกระทบต่ออัลกอริทึม USF เนื่องจากว่าหาก L_a มีค่าต่ำแสดงว่าเครือข่ายมีความคับคั่งต่ำด้วย และสามารถลด Combination ได้ แต่ในกรณีที่ค่า L_a มีการแกว่งสูง การที่ L_a

มีค่าต่ำไม่สามารถระบุได้ว่าเครือข่ายมีความคับคั่งต่ำไปด้วย สถานการณ์เช่นนี้จะทำให้อัลกอริทึม USF ทำงานผิดพลาดได้ เพราะเมื่อ L_a มีค่าต่ำกว่าเทรสโฮลด์ LOW อาจจะมีการลด Combination เนื่องจากว่าเข้าใจผิดว่าเครือข่ายมีความคับคั่งลดลงแล้ว แต่ความเป็นจริงแล้วเครือข่ายยังมีความคับคั่งอยู่ในระดับเดิม การลด Combination ลงทำให้ไม่สามารถกู้ข้อมูลที่สูญหายได้เพียงพอ



(a) ค่า L_a มีการแกว่งต่ำ (b) ค่า L_a มีการแกว่งสูง

รูปที่ 3.12 ลักษณะการแกว่งของค่า L_a

3.1.5.3 จุดบกพร่องของอัลกอริทึม RCCS

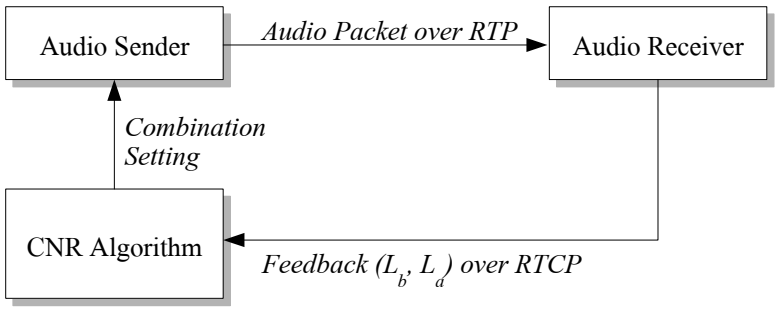
อัลกอริทึม RCCS มีจุดบกพร่องในส่วนของการเลือก Combination ซึ่งไม่ได้มีการแยกเงื่อนไขในการลดและเพิ่ม Combination จาก Pseudo Code ของอัลกอริทึม RCCS ในรูปที่ 3.11 จะเห็นว่าอัลกอริทึมนี้จะใช้ค่า Reward เพื่อค่า L_a' (ค่าทำนายของ L_a)ในแต่ละ Combination และจะเลือก Combination ที่มีค่า L_a' ที่มีค่าต่ำกว่าเทรสโฮลด์ 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

3.2 อัลกอริทึม CNR

เนื่องจากอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวที่ได้รับการเสนอก่อนหน้านี้ คือ อัลกอริทึม Bolot, USF และ RCCS ต่างก็ยังมีจุดบกพร่องตามที่ได้กล่าวแล้วในหัวข้อ 3.1.5 ในวิทยานิพนธ์นี้จึงได้มีการพัฒนาอัลกอริทึมสำหรับการการควบคุมความผิดพลาดแบบปรับตัวขึ้นมาใหม่ชื่อว่า อัลกอริทึม CNR ซึ่งเป็นชื่อห้องวิจัยที่ใช้ในการทำวิทยานิพนธ์นี้ โดยชื่อ CNR นี้

ย่อมาจาก Centre for Network Research โดยอัลกอริทึม CNR ได้แก้ไขจุดบกพร่องของอัลกอริทึม Bolot โดยการใช้ค่าจริงของ L_a ในการตัดสินเปลี่ยน Combination และได้แก้ไขปัญการแกว่งของค่า L_a ซึ่งเกิดกับอัลกอริทึม USF โดยการนับจำนวนครั้งที่ค่าของ L_a ต่ำกว่าเทรสโอลด์ LOW โดยจะลด Combination ก็ต่อเมื่อ L_a มีค่าต่ำกว่าเทรสโฮลด์ LOW ติดต่อนานเพียงพอแล้ว เพื่อให้แน่ใจความคับคั่งของเครือข่ายลดลงจริง และอัลกอริทึม CNR ได้แก้ไขจุดบกพร่องของอัลกอริทึม RCCS โดยการแยกเงื่อนไขในการตัดสินใจลดและเพิ่ม Combination ออกจากกัน

จากแผนภาพการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม CNR ในรูปที่ 3.13 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 3.13 แผนภาพการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม CNR

รูปที่ 3.14 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[10] ได้ระบุได้ว่าความยาวของแพ็กเก็ต RTCP จะต้องมีขนาดที่หาร 4 ลงตัว ดังนั้นจึงมีส่วนที่ไม่ได้ใช้งานเหลืออยู่ 3 ไบต์ และเนื่องจากว่าใน [10] ได้กำหนดเอาไว้ว่าค่าในฟิลด์ fraction lost คือค่าอัตราการสูญหายของแพ็กเก็ตคูณด้วย 256 ดังนั้นการระบุค่า L_b และ L_a จึงใช้วิธีการเดียวกันคือ

fraction lost = L_b x 256

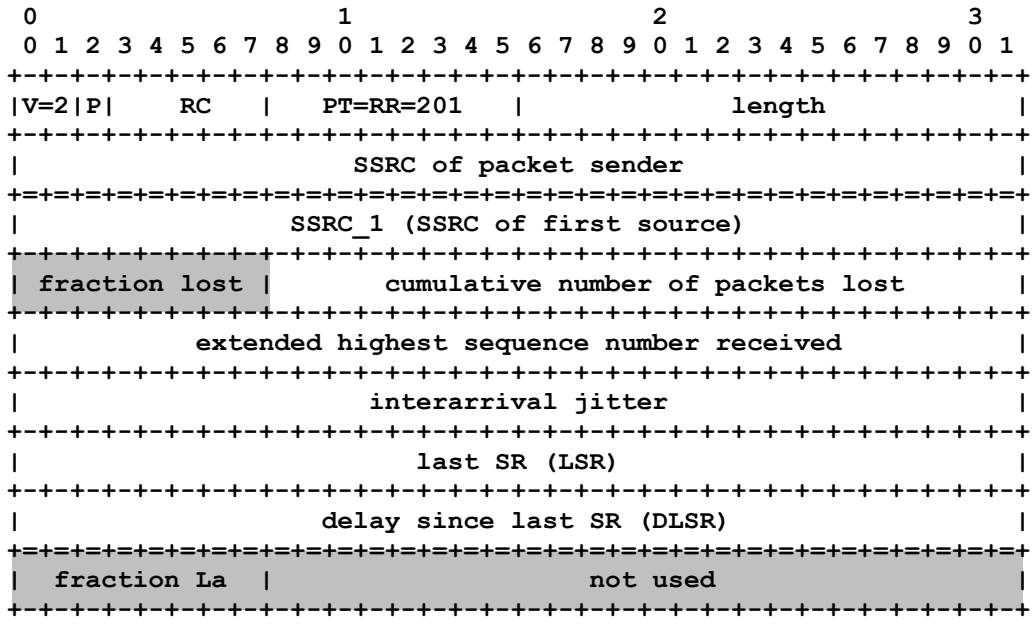
fraction L_a = L_a x 256

และในการควบคุมคุณภาพแบบปรับตัวโดยใช้อัลกอริทึม CNR เมื่อผู้ส่งได้รับแพ็กเก็ต Receiver Report ผู้ส่งก็จะอ่านค่า L_b และ L_a ที่อยู่ในแพ็กเก็ตดังนี้

L_b = fraction loss / 256

L_a = fraction L_a / 256

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



รูปที่ 3.14 การระบุค่า L_b และ L_a ในแพ็กเก็ต Receiver Report

สำหรับ Combination ของข้อมูลซ้ำที่ใช้ในอัลกอริทึม CNR มี 6 Combination ดังแสดงในตารางที่ 3.5 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 อื่น ๆ ก็สามารถแปลความหมายได้ในทำนองเดียวกันนี้ ส่วนในคอลัมน์ Initial Reward เป็นค่าเริ่มต้น

ของ Reward ในแต่ละ Combination ซึ่งก็คือค่า Reward ที่ได้จากการทดลองของ Bolot นั่นเอง แต่ในอัลกอริทึม CNR นี้ค่า Reward จะไม่คงที่ โดยจะถูกแก้ไขให้เปลี่ยนแปลงไปตามสภาพของเครือข่าย

ตารางที่ 3.5 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 3.15 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

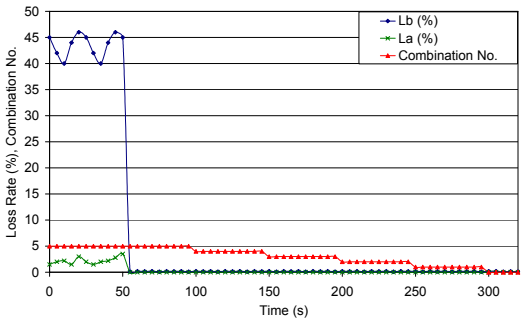
แต่การพิจารณาเฉพาะ count_L_a_under_low เพียงอย่างเดียวอาจจะทำให้การลด Combination ทำได้ช้าเกินไปในบางสถานการณ์ ตัวอย่างเช่น ในรูปที่ 3.16(a) หลังจากวินาทีที่ 50 เครือข่ายมีความคับคั่งต่ำมาก (L_b เกือบจะเป็น 0%) ซึ่งการใช้ Combination หมายเลข 0 (ไม่มีข้อมูลซ้ำเลย) ก็เพียงพอแล้ว แต่การลด Combination จากCombination หมายเลข 5 ไปเป็นหมายเลข 0 กลับต้องใช้เวลาถึง 250 วินาที เนื่องจากในที่นี้กำหนดค่า MIN_UNDER_LOW เท่ากับ 10 ซึ่งหมายความว่าการลด Combination จะเกิดขึ้นเมื่อ L_a หรือ L_b มีค่าต่ำกว่าเทรสโฮลด์ LOW อย่างน้อย 10 ครั้ง และแพ็กเก็ต Receiver Report จะมาถึงทุก 5 วินาที รวมแล้วต้องใช้เวลา 250 วินาที

```
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
```

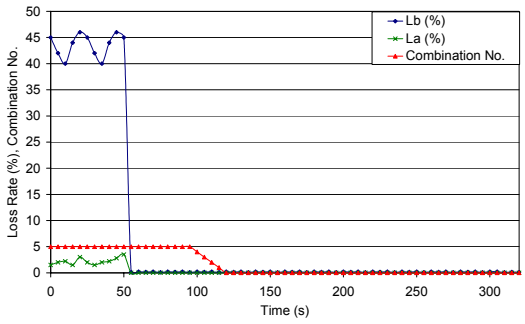
รูปที่ 3.15 Pseudo Code ของอัลกอริทึม CNR

ดังนั้นในอัลกอริทึม CNR จึงเพิ่มเงื่อนไขในการลด Combination อีกหนึ่งเงื่อนคือ ถ้าหากค่า count_L_b_under_low มีค่ามากกว่าหรือเท่ากับ MIN_UNDER_LOW แสดงว่าเครือข่ายมีความคับคั่งต่ำเป็นระยะเวลานานพอแล้ว และปลอดภัยพอที่จะลองลด Combination ได้ ซึ่ง

วิธีการนี้ช่วยให้การลด Combination ทำได้เร็วขึ้นในสภาพที่เครือข่ายมีความคับคั่งต่ำ จากรูปที่ 3.16(b) จะเห็นว่าการลด Combination จาก Combination หมายเลข 5 ไปเป็น Combination หมายเลข 0 ใช้เวลาเพียง 70 วินาที



(a) การลด Combination โดยพิจารณาเฉพาะ count_L_a_under_low



(b) การลด Combination โดยพิจารณาทั้ง count_L_a_under_low และ count_L_b_under_low

รูปที่ 3.16 การลด Combination โดยใช้ count_L_a_under_low อย่างเดียวอาจจะล่าช้าเกินไป

สำหรับค่าของ MIN_UNDER_LOW นั้น ถ้าหากมีค่ามากจะทำให้ป้องกันการแกว่งของค่า L_a ได้ดี แต่การลด Combination ทำได้ช้า และส่งผลให้มีปริมาณการใช้แบนด์วิดท์สูงเนื่องจากใช้ปริมาณข้อมูลซ้ำมาก ในที่นี้เลือกใช้ค่า MIN_UNDER_LOW เท่ากับ 10 ซึ่งมีค่าเท่ากับระยะเวลา 50 วินาทีเนื่องจากเป็นค่าที่มากพอที่จะลดปัญหาการแกว่งของค่า L_a ได้ดีซึ่งผลการทดลองในหัวข้อที่ 3.5 สามารถยืนยันได้ว่าเมื่อใช้ค่า MIN_UNDER_LOW เท่ากับ 10 แล้วทำให้สมรรถนะของอัลกอริทึม CNR ดีกว่าอัลกอริทึมอื่น รวมทั้งอัตราบิตที่ใช้ก็สูงกว่าอัลกอริทึมอื่นไม่มากนัก

3.3 การตรวจสอบความถูกต้องของแบบจำลอง

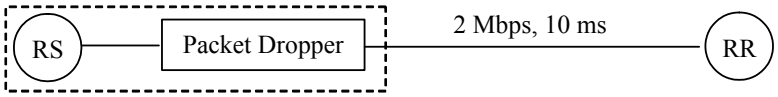
เนื่องจากในการทดลองเพื่อประเมินผลอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวแต่ละอัลกอริทึมทำโดยการใช้แบบจำลอง ดังนั้นก่อนที่จะทำการทดลองจำเป็นที่ต้องแน่ใจเสียก่อนว่าแบบจำลองที่จะนำมาใช้ในการทดลองนั้นสามารถทำงานได้อย่างถูกต้อง ซึ่งการที่จะยืนยันว่าแบบจำลองที่ใช้ในการทดลองนั้นสามารถทำงานได้อย่างถูกต้องจำเป็นที่จะต้องมีการตรวจสอบสองขั้นตอน โดยขั้นตอนแรกคือตรวจสอบการคำนวณค่า L_b และ L_a ของผู้รับ และขั้นตอนที่สองคือตรวจสอบการทำงานของอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวแต่ละอัลกอริทึม

3.3.1 การตรวจสอบการคำนวณค่า L_b และ L_a ของผู้รับ

เนื่องจากว่าค่า L_b และค่า L_a เป็นค่าที่จำเป็นสำหรับอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวที่จะต้องนำไปใช้ในการตัดสินใจเลือก Combination ของข้อมูลซ้ำ ดังนั้นหากเกิดข้อผิดพลาดในการคำนวณค่าทั้งสอง จะส่งผลให้การทำงานในขั้นตอนต่อไปเกิดข้อผิดพลาดตามไปด้วย

ในการทดลองนี้ได้กำหนดรูปแบบการเชื่อมต่อภายในเครือข่ายดังรูปที่ 3.17 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 3.6 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 3.17 การเชื่อมต่อภายในเครือข่ายที่ใช้ในการจำลอง

ตารางที่ 3.6 เป็นผลการตรวจสอบความถูกต้องในการคำนวณค่าของ L_b ของผู้รับ เนื่องจากว่าค่า L_b คืออัตราการสูญหายของข้อมูลเสียงก่อนการจัดเรียง ซึ่งก็คืออัตราการสูญหายของแพ็กเก็ตนั่นเอง ดังนั้นในกรณีนี้ค่า L_b ที่ถูกต้องจะต้องมีค่าเท่ากับอัตราการตัดทิ้งแพ็กเก็ต เนื่องจากในการทดลองการสูญหายของแพ็กเก็ตเกิดจากการตัดทิ้งเพียงอย่างเดียวเท่านั้น และไม่

ว่าจะใช้จำนวนข้อมูลซ้ำเท่าใดก็จะไม่มีผลต่อค่า L_b เพราะว่าเป็นการพิจารณาในขณะที่ยังไม่มีการนำข้อมูลซ้ำมากู้ข้อมูลในแพ็กเก็ตที่สูญหาย จากผลการทดลองในตารางที่ 3.6 จะเห็นว่าการคำนวณค่า L_b ของผู้รับสามารถทำได้ถูกต้องในทุกกรณี นั่นคือค่า L_b มีค่าเท่ากับอัตราการตัดทิ้งแพ็กเก็ตเสมอไม่ว่าจะใช้รูปแบบการตัดทิ้งแพ็กเก็ตแบบใด

ตารางที่ 3.6 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

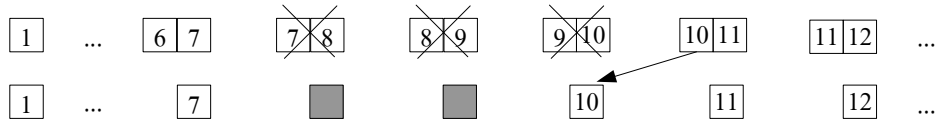
ตารางที่ 3.7 แสดงผลการตรวจสอบความถูกต้องในการคำนวณค่า L_a ของผู้รับในแบบจำลอง ซึ่งมีการตัดทิ้งแพ็กเก็ตตามรูปแบบเดียวกันกับที่ใช้ในตารางที่ 3.6 ซึ่งการตัดทิ้งแพ็กเก็ตทั้ง 5 รูปแบบสามารถคำนวณได้ล่วงหน้าแล้วว่าค่า L_a ที่ถูกต้องของแต่ละ Combination มีค่าเท่าใด ดังนั้นหากแบบจำลองสามารถทำงานได้ถูกต้อง ค่า L_a ที่จากการทดลองจะต้องมีค่าตรงกันกับค่าที่ถูกต้อง

รูปที่ 3.18 เป็นตัวอย่างของการหาค่า L_a เมื่อกำหนดให้มีการตัดทิ้งแพ็กเก็ตแบบ D02 ซึ่งจากตารางที่ 3.7 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 สำหรับในกรณีอื่น ๆ หลังจากที่ได้มีการตรวจสอบแล้วพบว่าการคำนวณค่า L_a ของแบบจำลองสามารถทำงานได้ถูกต้องในทุกกรณี

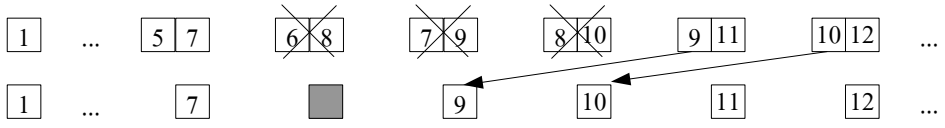
ตารางที่ 3.7 การตรวจสอบความถูกต้องในการคำนวณค่า L_a ของผู้รับในแบบจำลอง

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	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 3.18 ตัวอย่างการหาค่า L_a เมื่อกำหนดรูปแบบการตัดทิ้งแพ็กเก็ตแบบ D02

3.3.2 การตรวจสอบการทำงานของอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัว

เมื่อแน่ใจแล้วว่าการคำนวณค่า L_b และ L_a ในแบบจำลองสามารถทำได้อย่างถูกต้อง ขั้นตอนต่อไปคือการตรวจสอบการทำงานของอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวทั้ง 4 อัลกอริทึมคือ อัลกอริทึม Bolot, อัลกอริทึม USF, อัลกอริทึม RCCS และ

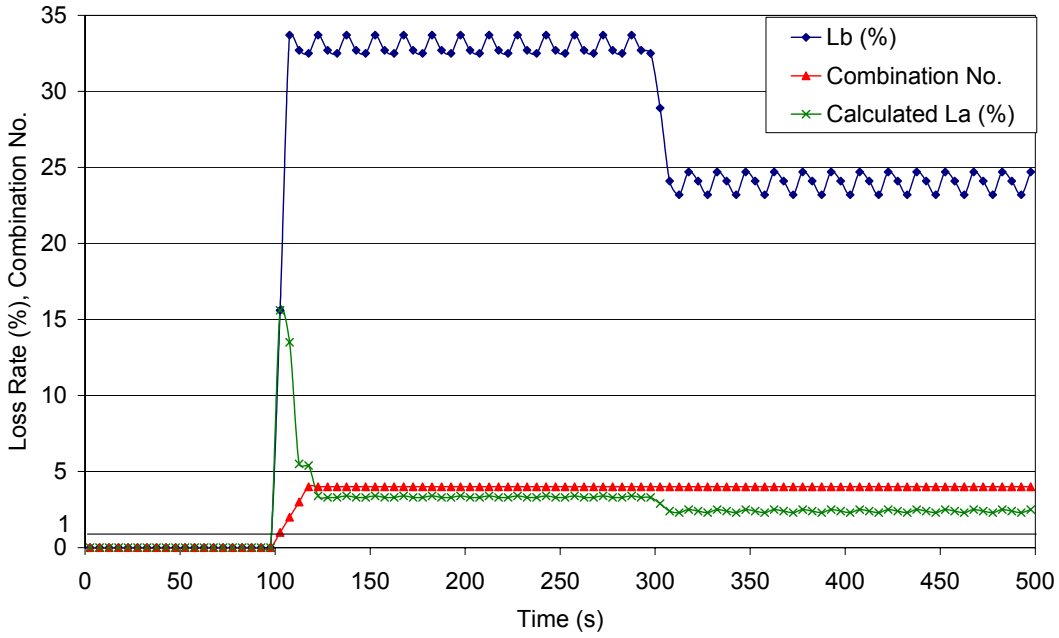
อัลกอริทึม CNR ซึ่งการทดลองในหัวข้อนี้ได้กำหนดให้ใช้เครือข่ายในการจำลองดังรูปที่ 3.17 โดยมีการจำลอง 4 ครั้ง (สำหรับ 4 Combination) และเพื่อที่จะตรวจสอบว่าการทำงานของแบบจำลองในส่วนของอัลกอริทึมในการควบคุมคุณภาพแบบปรับตัวว่าสามารถทำงานได้อย่างถูกต้องตามวิธีการที่ได้อธิบายในหัวข้อ 3.1 หรือไม่ จึงได้กำหนดรูปแบบการตัดทิ้งแพ็กเก็ตในแต่ละช่วงเวลาให้แตกต่างกัน 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 3.6)
- ในช่วงวินาทีที่ 301 – 500 มีการตัดทิ้งแพ็กเก็ตแบบ D05 (ดูรายละเอียดในตารางที่ 3.6)

ถึงแม้ว่าในบทความของอัลกอริทึม Bolot, USF และ RCCS ได้กำหนดตาราง Combination เอาไว้แล้วว่าในแต่ละ Combination ใช้การบีบอัดเสียงแบบใดสำหรับข้อมูลหลักและข้อมูลซ้ำ ซึ่งแทบทุก Combination ส่วนที่เป็นข้อมูลหลักและข้อมูลซ้ำใช้การบีบอัดเสียงแตกต่างกัน โดยการบีบอัดเสียงในส่วนของข้อมูลซ้ำจะมีอัตราบิตต่ำกว่า แต่ในการทดลองนี้กำหนดให้ทุก 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 3.5 ส่วนค่าเทรสโฮลด์ที่ใช้ในทุกอัลกอริทึมนั้นกำหนดให้เท่ากัน โดยค่าเทรสโฮลด์ HIGH เท่ากับ 5% เนื่องจากว่าหากมีข้อมูลเสียงสูญหายมากกว่า 5% จะทำให้รู้สึกได้ว่าเสียงมีคุณภาพลดลง[29] และกำหนดค่าเทรสโฮลด์ LOW เท่ากับ 1% ซึ่งถือว่าเป็นค่าที่ต่ำพอที่จะลองเพิ่ม Combination ได้แล้ว

3.3.2.1 การตรวจสอบการทำงานของอัลกอริทึม Bolot

รูปที่ 3.19 เป็นกราฟแสดงผลการจำลองโดยใช้อัลกอริทึม Bolot ซึ่งจะเห็นว่าค่า L_b ในแต่ละช่วงเวลาตรงกันกับอัตราการตัดทิ้งแพ็กเก็ตตามที่ได้กำหนดไว้ข้างต้น นั่นคือช่วง 100 วินาทีแรก ค่า 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 3.19 สามารถยืนยันได้ว่าค่า L_a จากการคำนวณของแบบจำลองมีค่าถูกต้อง ตัวอย่างเช่น หลังจากวินาทีที่ 300 เป็นต้นไปมีการใช้ Combination ที่ 4 ซึ่งจากตารางที่ 3.5 ค่า Reward ของ Combination นี้เท่ากับ 10 และในช่วงเวลานี้ค่า L_b เฉลี่ยอยู่ที่ 24% ดังนั้นค่า L_a จากการคำนวณจะต้องอยู่ที่ 2.4 % ซึ่งตรงกับที่ได้แสดงอยู่ในกราฟรูปที่ 3.19



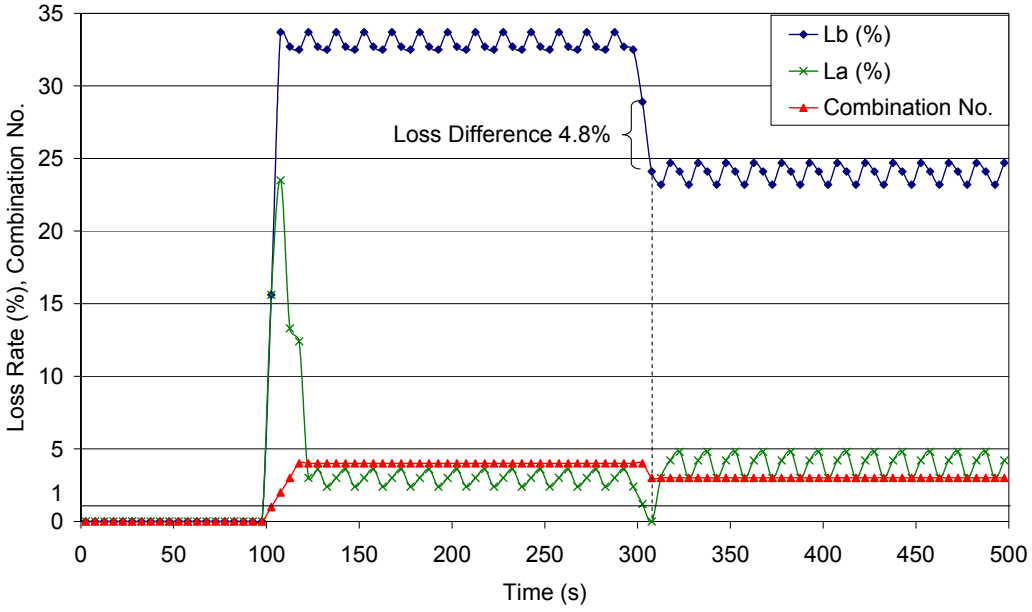
รูปที่ 3.19 ค่า L_b และค่า L_a จากการคำนวณในระหว่างการจำลองโดยใช้อัลกอริทึม Bolot

หลังจากที่ได้ตรวจสอบแล้วค่า L_a จากการคำนวณนั้นได้มาอย่างถูกต้อง ขั้นตอนต่อไปก็คือการตรวจสอบการเปลี่ยน Combination ว่าถูกต้องตาม Pseudo Code ของอัลกอริทึม Bolot ในรูปที่ 3.6 หรือไม่ ซึ่งกราฟในรูปที่ 3.19 ก็สามารถยืนยันได้ว่าแบบจำลองสามารถทำงานได้ถูกต้อง โดยในการทดลองนี้ได้กำหนดให้เทรสโฮลด์ LOW และ HIGH เป็น 1% และ 5% ตามลำดับ จะเห็นได้ว่าหลังจากวินาทีที่ 100 เมื่อค่า L_a (จากการคำนวณ) สูงเกินกว่า 5% จะมี

การเพิ่ม Combination ขึ้นทีละหนึ่งขั้นและหยุดที่ Combination ที่ 4 เนื่องว่าค่า L_a ลดลงมาต่ำกว่า 5% และหลังจากวินาทีที่ 300 ถึงแม้ว่าค่า L_a จะลดลงแต่ไม่ก็ยังไม่ต่ำกว่า 1% ดังนั้นจึงไม่มีการลด Combination

3.3.2.2 การตรวจสอบการทำงานของอัลกอริทึม USF

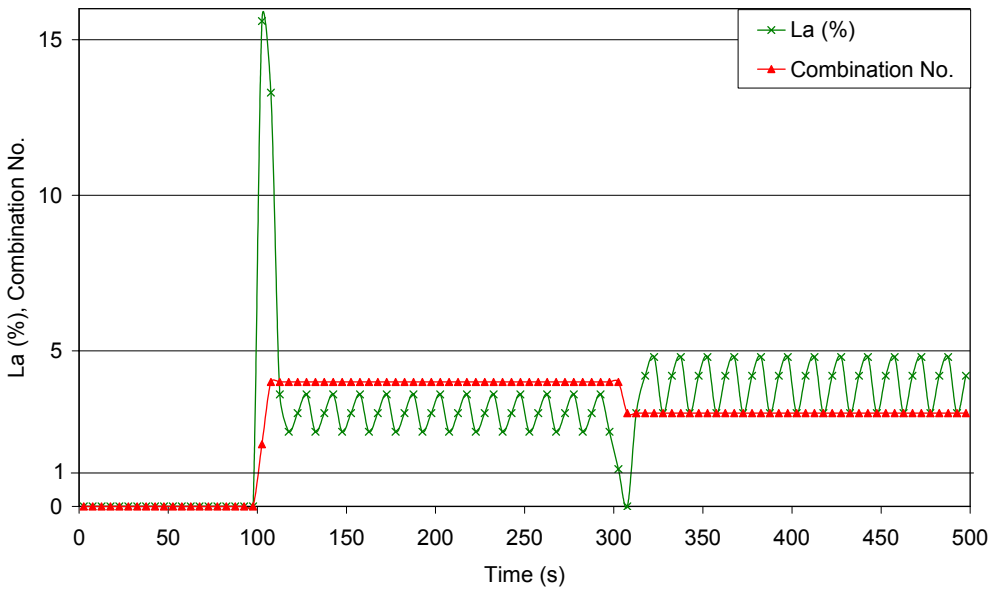
การควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม USF จะแตกต่างกับอัลกอริทึม Bolot ตรงที่อัลกอริทึม USF ใช้ค่าจริงของ L_a ในการพิจารณาเพิ่มหรือลด Combination ในรูปที่ 3.20 เป็นกราฟแสดงการเปลี่ยนแปลง Combination เมื่อมีการจำลองโดยใช้อัลกอริทึม USF ซึ่งจะเห็นว่าการทำงานของแบบจำลองเป็นไปตาม Pseudo Code ของอัลกอริทึม USF ที่อธิบายไว้ในรูปที่ 3.9 โดยในการทดลองได้กำหนดค่าเทรสโฮลด์ LOW และ HIGH เท่ากับ 1% และ 5% ตามลำดับ ส่วนค่า MINIMUM_THRESHOLD ใช้ตามที่กำหนดไว้ใน [19] คือ 3% จากกราฟจะเห็นว่าเริ่มมีการเพิ่ม Combination หลังจากวินาทีที่ 100 เนื่องจาก L_a สูงกว่า 5% และจะไปหยุดที่ Combination หมายเลข 4 เพราะว่าที่จุดนี้ L_a ต่ำกว่า 5% และหลังจากวินาทีที่ 300 ได้มีการลด Combination ลงมาอยู่ที่ Combination หมายเลข 3 เนื่องจากว่าค่า L_a ลดลงมาต่ำกว่า 1% แล้ว รวมทั้งค่าความแตกต่างของ L_b ก็มีค่าสูงกล่าวค่า MINIMUM_TRESHOLD (จากกราฟความแตกต่างของ L_b ในรอบปัจจุบันกับรอบก่อนหน้าต่างกัน 4.8%) ซึ่งเป็นเงื่อนไขในการลด Combination ของอัลกอริทึมนี้



รูปที่ 3.20 การเปลี่ยนแปลงของ Combination ในระหว่างการจำลองโดยใช้อัลกอริทึม USF

3.3.2.3 การตรวจสอบการทำงานของอัลกอริทึม RCCS

อัลกอริทึม RCCS มีการใช้ทั้งค่า L_a และค่าเวลาหน่วงระหว่างปลายทางในการเลือก Combination จาก Pseudo Code ของอัลกอริทึม RCCS ในรูปที่ 3.11 จะเห็นว่าอัลกอริทึมนี้จะพิจารณาตาราง Combination ตั้งแต่ Combination แรกจนถึง Combination สุดท้าย และจะเลือก Combination แรกที่มีค่า L_a' (ค่าทำนายของ L_a ซึ่งได้จากการนำ L_b หารด้วยค่า Reward) และค่า D_a' (ค่าทำนายของเวลาหน่วงระหว่างปลายทางหลังการจัดเรียงข้อมูลซ้ำ) ไม่เกินค่าที่กำหนด แต่เนื่องจากว่าการหาค่าเวลาหน่วงระหว่างปลายทางนั้นทำได้ยากเพราะนาฬิกาของผู้ส่งกับผู้รับนั้นไม่ประสานกัน อีกทั้งใน [20] ยังอธิบายวิธีการหาค่าเวลาระหว่างปลายทางได้ไม่ชัดเจนพอที่จะนำมาสร้างแบบจำลองได้ ดังนั้นในการเลือก Combination ของอัลกอริทึม RCCS ในแบบจำลองที่สร้างขึ้น จึงตัดเงื่อนไขของการตรวจสอบค่าเวลาหน่วงระหว่างปลายทางออกไป โดยถือว่าเวลาหน่วงที่เกิดขึ้นจากการใช้ Combination ในตารางที่ 3.5 ถือว่าเป็นค่าที่ยอมรับได้ทั้งหมด อย่างไรก็ตามการตัดเงื่อนไขของการตรวจสอบค่าเวลาหน่วงระหว่างปลายทางก็ไม่ได้ทำให้อัลกอริทึมนี้เสียเปรียบอัลกอริทึมอื่น เนื่องจากว่าหากยังมีการตรวจสอบค่าเวลาหน่วงระหว่างปลายทางอยู่ ในบางครั้งอัลกอริทึมนี้อาจจะไม่สามารถเพิ่ม Combination ได้เนื่องจากค่าเวลาหน่วงสูงเกินค่าเทรสโฮลด์ ทำให้ไม่สามารถกู้ข้อมูลที่สูญหายได้เพียงพอ แต่หากไม่มีการตรวจสอบค่าเวลาหน่วงระหว่างปลายทาง สามารถเพิ่ม Combination ได้จนถึง Combination สูงสุด และสำหรับค่า α ซึ่งเป็นพารามิเตอร์ของสมการตัวกรองในการแก้ไขค่า Reward ของอัลกอริทึมนี้ใช้ค่าตามที่กำหนดเอาไว้ใน [20] คือ 0.98

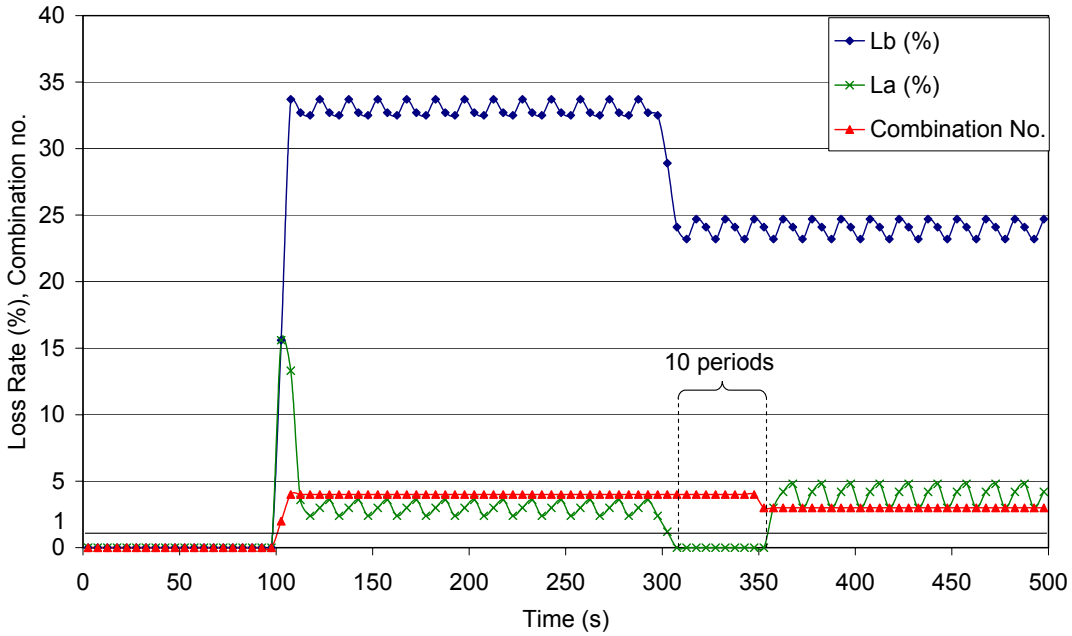


รูปที่ 3.21 การเปลี่ยนแปลงของ Combination ในระหว่างการจำลองโดยใช้อัลกอริทึม RCCS

จากการตรวจสอบการทำงานของแบบจำลองเมื่อใช้อัลกอริทึม RCCS ปรากฏว่าแบบจำลองสามารถทำงานได้ถูกต้อง ซึ่งจะเห็นได้จากกราฟแสดงผลการทดลองในรูปที่ 3.21 โดยหลังจากวินาทีที่ 100 ซึ่งเริ่มมีการตัดทิ้งแพ็กเก็ตแบบ D04 (ดูรายละเอียดได้ในตารางที่ 3.6) ซึ่งมีการตัดทิ้งแพ็กเก็ต 33% ค่า L_a ก็เพิ่มขึ้นมากกว่า 5 % อัลกอริทึม RCCS จึงได้มีการเปลี่ยน Combination ไปเป็น Combination หมายเลข 4 และหลังจากนั้นก็ไม่มีการเปลี่ยน Combination เนื่องจากค่า L_a มีค่าต่ำกว่า 5% และหลังจากวินาทีที่ 300 ซึ่งได้มีการลดการตัดทิ้งแพ็กเก็ตลงโดยใช้การตัดทิ้งแบบ D05 ซึ่งมีอัตราการตัดทิ้งแพ็กเก็ต 24% ค่า L_a จึงมีค่าลดลง และเมื่อค่า L_a ต่ำกว่า 1% อัลกอริทึม RCCS ก็ได้มีการเปลี่ยน Combination อีกครั้ง โดยลด Combination ลงมาอยู่ที่ Combination หมายเลข 3

3.3.2.4 การตรวจสอบการทำงานของอัลกอริทึม CNR

จากการตรวจสอบการทำงานของแบบจำลองเมื่อมีการควบคุมความผิดพลาดแบบปรับตัวโดยใช้อัลกอริทึม CNR การเปลี่ยนแปลง Combination เป็นไปตามกราฟในรูปที่ 3.22 ซึ่งจะเห็นว่าแบบจำลองสามารถทำงานได้ถูกต้องตาม Pseudo Code ของอัลกอริทึม CNR ในรูปที่ 3.15



รูปที่ 3.22 การเปลี่ยนแปลงของ Combination ในระหว่างการจำลองโดยใช้อัลกอริทึม CNR

โดยหลังจากวินาทีที่ 100 ซึ่งเริ่มมีการตัดทิ้งแพ็กเก็ต ได้มีการเพิ่ม Combination ไปหยุดที่ Combination หมายเลข 4 เนื่องจากค่า L_a มีค่าต่ำกว่า 5% แล้ว และหลังจากวินาทีที่ 300 ซึ่ง

ได้มีการลดการตัดทิ้งแพ็กเก็ตลง ค่า L_a จึงมีค่าลดลง และเมื่อค่า L_a ต่ำกว่า 1% เป็นจำนวน 10 ครั้ง ก็ได้มีการลด Combination ลงมาอยู่ที่ Combination หมายเลข 3 ทั้งนี้เนื่องจากในที่นี้ได้กำหนดค่า MIN_DURATION เท่ากับ 10

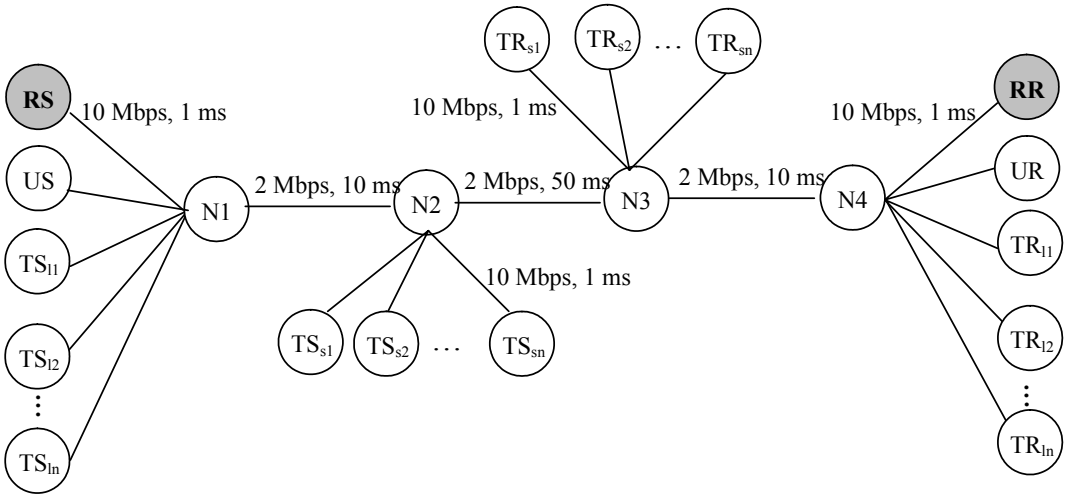
3.4 การทดลองเพื่อประเมินผลอัลกอริทึม CNR

การทดลองในหัวข้อนี้เป็นการประเมินผลอัลกอริทึม CNR โดยมีการเปรียบเทียบกับอัลกอริทึม Bolot, USF และ RCCS ในกรณีที่ใช้การบีบอัดเสียง G.723.1 ทั้งข้อมูลหลักและข้อมูลซ้ำ สำหรับ Combination ของข้อมูลซ้ำที่ใช้ในการทดลองมี 6 Combination ดังแสดงในตารางที่ 3.5 และเนื่องจากว่าแต่ละอัลกอริทึมต่างก็มีพิจารณาค่าเทรสโฮลด์ HIGH และ LOW ในการเลือก Combination ดังนั้นในการเปรียบเทียบสมรรถนะของแต่ละอัลกอริทึมนั้น จำเป็นจะต้องกำหนดค่าเทรสโฮลด์ให้เท่ากัน ใน [29] ได้ระบุเอาไว้ว่าหากมีข้อมูลเสียงสูญหายมากกว่า 5 % จะทำให้รู้สึกได้ว่าเสียงมีคุณภาพลดลง ดังนั้นการทดลองนี้จึงเลือกใช้ที่จะใช้ค่าเทรสโฮลด์ HIGH เท่ากับ 5% และเพื่อให้ผลการทดลองมีความน่าเชื่อถือมากขึ้น ในการทดลองนี้[29]จึงได้มีการจำลองในกรณีที่ใช้ค่าเทรสโฮลด์ HIGH เท่ากับ 3% ด้วย ซึ่งกรณีนี้ก็อาจจะเป็นไปได้ในการใช้งานจริง เช่น ผู้ใช้ต้องการคุณภาพเสียงที่ดีมากโดยยอมรับการสูญหายของข้อมูลเสียงได้ไม่เกิน 3% เป็นต้น ส่วนค่าเทรสโฮลด์อีกค่าหนึ่งคือค่าเทรสโฮลด์ LOW กำหนดให้มีค่าเท่ากับ 1% ซึ่งค่าของเทรสโฮลด์ LOW นี้ไม่ควรกำหนดให้มีค่าใกล้เคียงกับค่าเทรสโฮลด์ HIGH มากนัก เพื่อป้องกันไม่ให้เกิดการเปลี่ยน Combination บ่อยเกินไป

ในการทดลองนี้ไม่ได้มีการส่งข้อมูลในเครือข่ายจริง แต่เป็นการจำลองบน NS-2[36] ในรูปที่ 3.23 ได้แสดงการเชื่อมต่อของแต่ละโหนดที่อยู่ภายในเครือข่ายที่ใช้ในการจำลอง ลิงค์ระหว่างโหนด N2 และ N3 มีแบนด์วิดท์ 2 เมกะบิตต่อวินาที (Mbps) เวลาหน่วง 50 มิลลิวินาที (ms) ซึ่งลิงค์นี้ถือเป็นลิงค์คอขวดของระบบ ส่วนลิงค์ระหว่างโหนด N1 กับ N2 และลิงค์ระหว่างโหนด N3 กับ N4 มีแบนด์วิดท์ 2 เมกะบิตต่อวินาที เวลาหน่วง 10 มิลลิวินาที ส่วนลิงค์อื่นที่เหลือมีความจุ 10 เมกะบิตต่อวินาที เวลาหน่วง 1 มิลลิวินาที และลิงค์ทุกลิงค์เป็นลิงค์แบบ Drop Tail โดยโหนด RS เป็นโหนดที่ส่งแพ็กเก็ตของเสียง และมีการกำหนด Combination ของข้อมูลซ้ำตามอัลกอริทึมที่เลือกใช้ นอกจากนี้ยังมีโหนดที่คอยส่งทราฟฟิกชนิดอื่น ๆ เพื่อให้เครือข่ายมีสภาพแวดล้อมเหมือนมีผู้ใช้งานคนอื่นกำลังใช้เครือข่ายอยู่ด้วย โดยมีทั้งผู้ใช้ที่ส่งทราฟฟิก UDP และผู้ใช้ที่ส่งทราฟฟิก TCP

ในที่นี้ได้แบ่งการทดลองออกเป็น 4 ตอน โดยสิ่งที่แตกต่างกันก็คือ พฤติกรรมในการส่งข้อมูลของโหนดที่ส่งทราฟฟิก TCP โดยในตอนที่ 1 โหนดที่ส่งทราฟฟิก TCP จะมีการส่งข้อมูลติดต่อกันเป็นเวลานานโดยไม่หยุด ส่วนในตอนที่ 2 โหนดที่ส่งทราฟฟิก TCP จะมีการส่งและหยุดส่งข้อมูลสลับกันไปเป็นระยะ ส่วนในตอนที่ 3 และตอนที่ 4 นั้นพฤติกรรมการส่งทราฟฟิก

TCP จะเหมือนกับตอนที่ 1 และตอนที่ 2 ตามลำดับ แต่ในตอนที่ 3 และตอนที่ 4 ตำแหน่งที่ตั้งของโหนดที่ส่งทราฟฟิก TCP จะแตกต่างจากตอนที่ 1 และตอนที่ 2 ทั้งนี้เพื่อที่จะประเมินสมรรถนะของอัลกอริทึม CNR ในสภาพเครือข่ายที่แตกต่างกัน เพื่อเพิ่มความน่าเชื่อถือของผลการทดลอง



รูปที่ 3.23 การเชื่อมต่อภายในเครือข่ายที่ใช้ในการจำลอง

3.4.1 การทดลองตอนที่ 1

ในการทดลองตอนที่ 1 เป็นการจำลองให้มีการสื่อสารเสียงในเครือข่ายที่มีผู้ที่ส่งทราฟฟิก TCP เป็นช่วงเวลานาน และมีปริมาณการใช้แบนด์วิดท์เต็มความจุของเครือข่าย ซึ่งในสภาพเครือข่ายเช่นนี้สามารถทำให้เกิดการแกว่งของค่าอัตราการสูญหายของแพ็กเก็ตในการสื่อสารเสียงได้ โดยมีจุดประสงค์เพื่อประเมินสมรรถนะของแต่ละอัลกอริทึมว่าจะได้รับผลจากปัญหาการแกว่งของค่าอัตราการสูญหายของแพ็กเก็ตมากน้อยเพียงใด และในกรณีที่สภาพเครือข่ายเป็นเช่นนี้อัลกอริทึมใดสามารถทำงานได้ผลดีที่สุด ซึ่งการส่งข้อมูลของแต่ละโหนดเป็นดังนี้

- โหนด RS ส่งเสียงโดยใช้โปรโตคอล RTP ไปให้โหนด RR โดยส่งแพ็กเก็ตทุก ๆ 30 มิลลิวินาที ซึ่งระยะเวลาเท่ากับขนาดเฟรมของ G.723.1 ส่วนขนาดแพ็กเก็ตขึ้นอยู่กับ Combination ที่ใช้ ถ้า Combination ใดมีจำนวนข้อมูลซ้ำมาก แพ็กเก็ตเสียงที่ส่งก็จะมีขนาดใหญ่
- โหนด RR ส่งแพ็กเก็ต Receiver Report ให้กับโหนด RS และเนื่องจากในที่นี้เป็นการติดต่อแบบจุดต่อจุด ดังนั้นแพ็กเก็ต Receiver Report นี้จะมีการส่งทุก ๆ 5 วินาที

- โหนด US ส่งทราฟฟิก UDP ด้วยอัตราบิต 400 กิโลบิตต่อวินาที โดยใช้แพ็กเก็ตขนาด 1024 ไบต์ ซึ่งโหนด US ทำหน้าที่เป็นตัวแทนของโปรแกรมประยุกต์ในการส่งเสียงหรือวีดิทัศน์ที่ไม่มีการปรับตัวตามสภาพเครือข่าย
- โหนด TS_{l1} – TS_{ln} และ TS_{s1} - TS_{sn} ส่งไฟล์โดยใช้โปรโตคอล FTP ไปให้กับโหนด TR_{l1} – TR_{ln} และ TR_{s1} - TR_{sn} ตาลำดับ โดยเริ่มส่งที่วินาทีที่ 100 และหยุดส่งเมื่อการจำลองยุติ ดังนั้นในช่วง 100 วินาทีแรกจะไม่มีแพ็กเก็ตเสียงสูญหายเลย
- การจำลองยุติที่วินาทีที่ 1200

โดยในการทดลองตอนนี้กำหนดให้การจำลองในแต่ละอัลกอริทึมมีการรันแบบจำลอง 2 ครั้ง โดยครั้งแรกใช้จำนวนโหนด TCP เท่ากับ 20 โหนด (TS_{l1}-TS_{l10} และ TS_{s1}-TS_{s10}) และครั้งที่สองใช้จำนวนโหนด TCP เป็น 40 โหนด (TS_{l1}-TS_{l20} และ TS_{s1}-TS_{s20}) ทั้งนี้เพื่อให้จะได้พิจารณาผลการทดลองว่าในกรณีที่มีจำนวนโหนด TCP ต่างกัน ผลการทดลองจะเป็นไปในทำนองเดียวกันหรือไม่

3.4.2 การทดลองตอนที่ 2

ในการทดลองตอนที่ 2 นี้ ลักษณะการส่งข้อมูลของโหนด RS และ US ยังเหมือนกับในการทดลองตอนที่ 1 แต่มีการเปลี่ยนแปลงที่โหนด TS_{l1} – TS_{ln} และ TS_{s1} - TS_{sn} ซึ่งจะเริ่มส่งและหยุดส่งข้อมูลพร้อมกันทุก ๆ 200 วินาที (ส่ง 200 วินาทีและหยุดส่ง 200 วินาที) ทั้งนี้เนื่องจากว่าต้องการประเมินสมรรถนะของแต่ละอัลกอริทึมในสภาพแวดล้อมที่ระดับความคับคั่งของเครือข่ายไม่คงที่ ซึ่งในการทดลองนี้ก็มีการรันแบบจำลอง 2 ครั้งต่อหนึ่งอัลกอริทึมเช่นเดียวกับในตอนที่ 1 โดยครั้งแรกใช้จำนวนโหนด TCP เท่ากับ 20 โหนด และครั้งที่สองใช้จำนวนโหนด TCP เป็น 40 โหนด

3.4.3 การทดลองตอนที่ 3

ข้อกำหนดสำหรับการทดลองในตอนนี้ส่วนใหญ่เหมือนกันการทดลองในตอนที่ 1 สิ่งที่แตกต่างกันก็คือการทดลองตอนนี้โหนดที่มีการส่งทราฟฟิก TCP มีเฉพาะโหนด TS_{l1} – TS_{ln} เท่านั้น โดยโหนด TS_{s1} – TS_{sn} ไม่มีการส่งข้อมูล

3.4.4 การทดลองตอนที่ 4

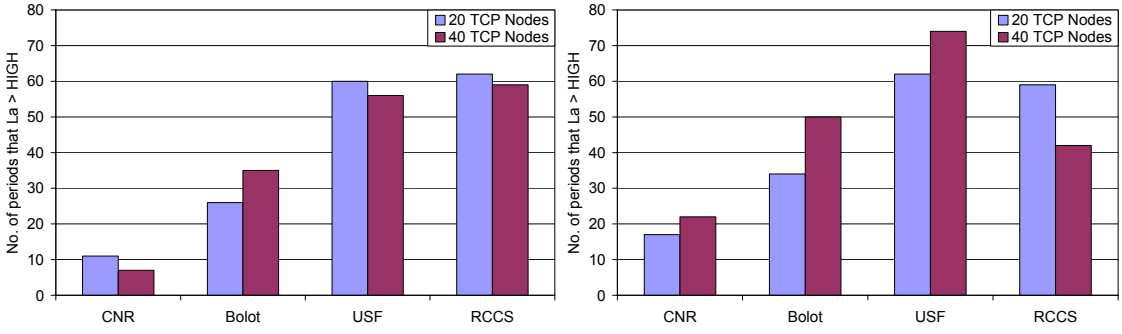
ข้อกำหนดสำหรับการทดลองในตอนนี้ส่วนใหญ่เหมือนกันการทดลองในตอนที่ 2 สิ่งที่แตกต่างกันก็คือการทดลองตอนนี้โหนดที่มีการส่งทราฟฟิก TCP มีเฉพาะโหนด TS_{l1} – TS_{ln} เท่านั้น โดยโหนด TS_{s1} – TS_{sn} ไม่มีการส่งข้อมูล

3.5 ผลการทดลอง

ในการประเมินสมรรถนะของอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวแต่ละอัลกอริทึม ใช้วิธีการนับจำนวนครั้งที่ L_a มีค่าเกินค่าเทรสโฮลด์ HIGH ซึ่งเป็นการนับว่ามีกี่ครั้งที่มีการสูญหายของข้อมูลเสียงมากจนส่งผลกระทบต่อคุณภาพเสียงได้ รวมทั้งพิจารณาค่าเฉลี่ยของ L_a จากการจำลองของแต่ละอัลกอริทึม และพิจารณาถึงอัตราบิตที่ใช้ในการที่ส่งเสียงซึ่งเป็นผลจากการเลือก Combination ของแต่ละอัลกอริทึม ซึ่งผลการทดลองในแต่ละตอนมีดังนี้

3.5.1 ผลการทดลองตอนที่ 1

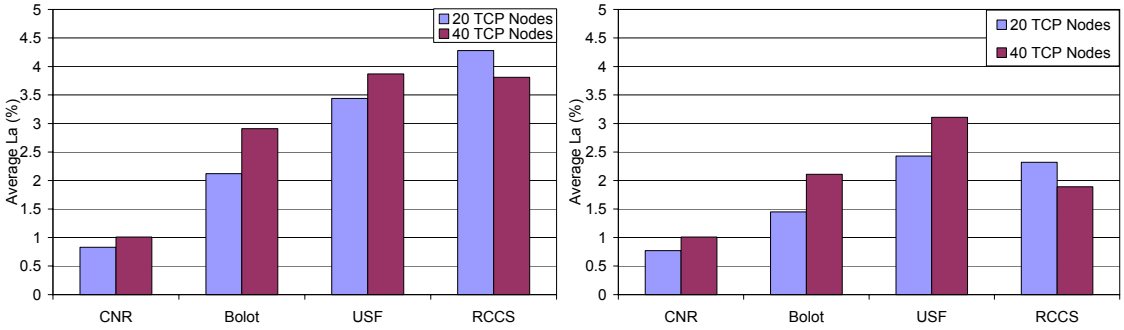
เมื่อพิจารณาจำนวนครั้งที่ที่ L_a มีค่าเกินเทรสโฮลด์ HIGH ซึ่งทำให้เสียงที่มีคุณภาพที่ต่ำเกินกว่าที่จะยอมรับได้ จากแผนภูมิในรูปที่ 3.24 จะเห็นว่าเมื่อลดค่าเทรสโฮลด์จาก 5% เป็น 3% จะทำให้จำนวนครั้งที่ L_a มีค่าเกินค่าเทรสโฮลด์ HIGH ของแต่ละอัลกอริทึมมีค่าเพิ่มขึ้น เนื่องจากว่าปกติแล้วเมื่อค่าเทรสโฮลด์ HIGH ลดลงทำให้โอกาสที่ L_a จะสูงกว่าเทรสโฮลด์ HIGH มีมากขึ้น ยกเว้นอัลกอริทึม RCCS ที่มีผลแตกต่างจากอัลกอริทึมอื่น เนื่องจากว่าในการทดลองนี้จุดบกพร่องของอัลกอริทึม RCCS (ตามที่ได้อธิบายไว้ในหัวข้อ 3.1.5.3) เกิดขึ้นบ่อยครั้งเมื่อกำหนดค่าเทรสโฮลด์ HIGH เท่ากับ 3% อย่างไรก็ตามอัลกอริทึม CNR ก็ให้ผลดีที่สุดทั้งในกรณีที่จำนวนโหนดที่ส่งทราฟฟิก TCP เป็น 20 และ 40 โหนด ไม่ว่าจะใช้ค่าเทรสโฮลด์ HIGH เท่ากับ 5% หรือ 3%



(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

รูปที่ 3.24 จำนวนครั้งที่ L_a เกินเทรสโฮลด์ HIGH (ได้รับ Receiver Report 240 ครั้ง)

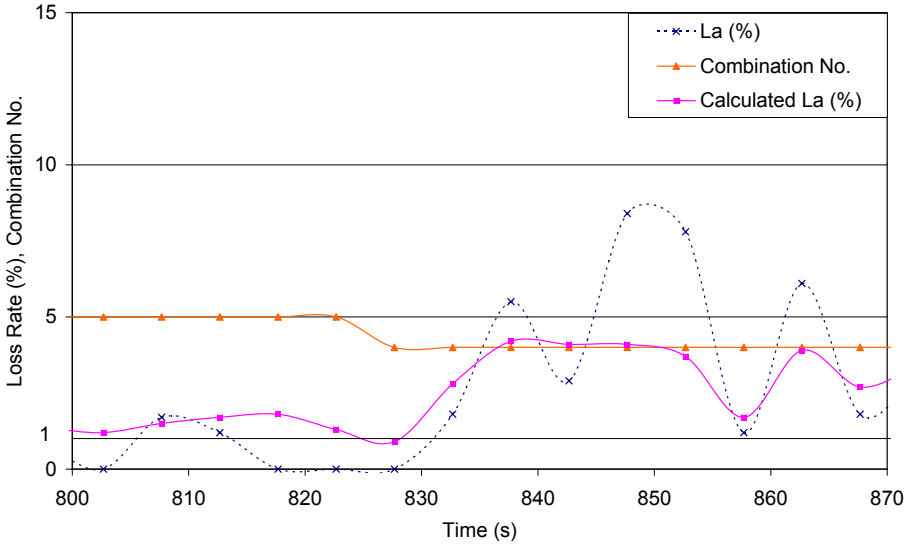
เมื่อพิจารณาที่ค่าเฉลี่ยของ L_a จากแผนภูมิในรูปที่ 3.25 จะเห็นว่าส่วนใหญ่แล้วค่าเฉลี่ยของ L_a จะไม่เกินค่าเทรสโฮลด์ HIGH ดังนั้นเมื่อลดค่าเทรสโฮลด์ HIGH จาก 5% เป็น 3% ค่าเฉลี่ยของ L_a ในแต่ละอัลกอริทึมจึงลดลง และเมื่อเปรียบผลการทดลองของทุกอัลกอริทึมแล้วจะเห็นว่าอัลกอริทึม CNR มีค่าเฉลี่ยของ L_a ต่ำที่สุดในทุกกรณี



(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

รูปที่ 3.25 ค่าเฉลี่ยของ L_a จากการจำลองในแต่ละอัลกอริทึม

เมื่อพิจารณาโดยละเอียดถึงการเปลี่ยน Combination ในการจำลองของแต่ละอัลกอริทึม เพื่อหาเหตุผลว่าเหตุใดอัลกอริทึม CNR จึงมีควบคุมการสูญหายของข้อมูลเสียงได้ดีกว่าอัลกอริทึมอื่น สาเหตุหลักก็คืออัลกอริทึม Bolot, USF และ RCCS ต่างก็ยังมีจุดบกพร่องอยู่นั่นเอง โดยอัลกอริทึม Bolot นั้นมีจุดบกพร่องอยู่ที่การใช้ค่า Reward มาคำนวณค่า L_a แทนการใช้ค่าจริงของ L_a ซึ่งจากผลการทดลองพบว่ามีหลายช่วงเวลาที่ค่า L_a จากการคำนวณมีความคลาดเคลื่อนจากค่าจริง และทำให้การความผิดพลาดในการตัดสินใจเพิ่มหรือลด Combination

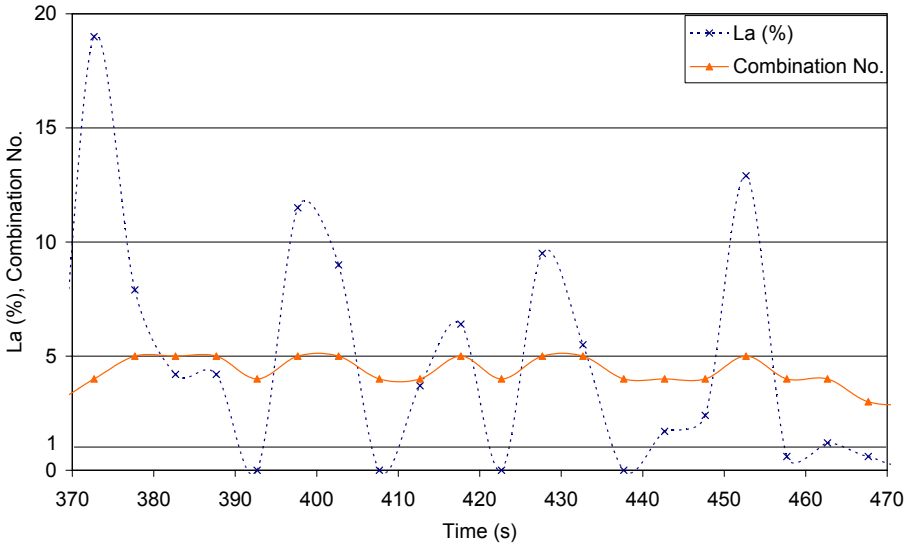


รูปที่ 3.26 การใช้ค่า Reward ของ Bolot ในการคำนวณค่า L_a อาจคลาดเคลื่อนจากค่าจริง

จากรูปที่ 3.26 ซึ่งเป็นกราฟแสดงการเปลี่ยนแปลง Combination ของการจำลองโดยใช้อัลกอริทึม Bolot เมื่อกำหนดจำนวนโหนด TCP เท่ากับ 40 โหนด และค่าเทรสโฮลด์เท่ากับ 5% จะเห็นว่าหลังจากวินาทีที่ 830 มีหลายครั้งที่ค่าจริงของ L_a สูงกว่า 5% ซึ่งสมควรที่จะต้องเพิ่ม

Combination แต่จากกราฟจะเห็นว่าอัลกอริทึม Bolot ยังคงใช้ Combination หมายเลข 4 โดยไม่เพิ่ม Combination ไปเป็น Combination หมายเลข 5 ทั้งนี้เนื่องจากว่าค่า L_a จากการคำนวณยังไม่เกิน 5 % ซึ่งจะเห็นว่าความคาดเคลื่อนการคำนวณทำให้อัลกอริทึม Bolot ตัดสินใจผิดพลาด

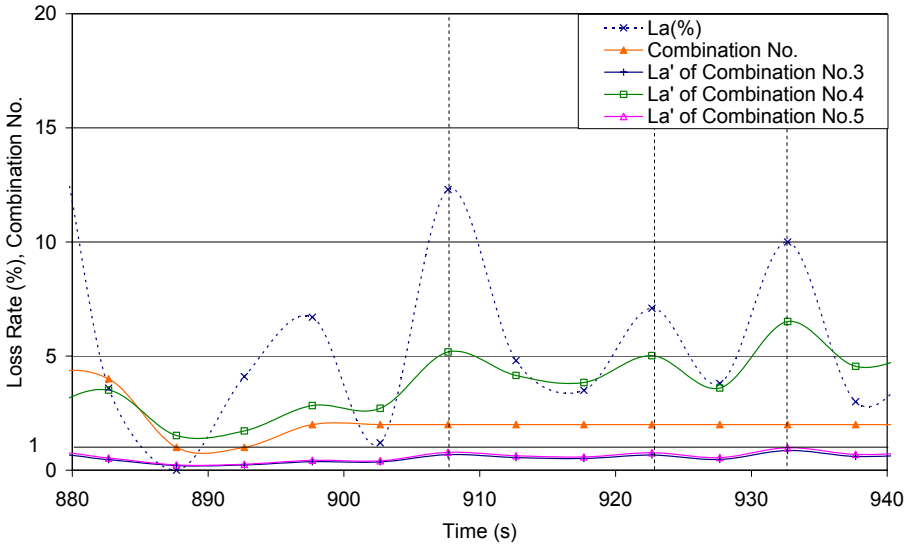
สำหรับอัลกอริทึม USF มีจุดบกพร่องตรงที่อัลกอริทึมนี้จะได้รับผลกระทบจากการแกว่งของค่า L_a เนื่องจากอัลกอริทึมนี้ใช้ค่าจริงของ L_a ในการเลือก Combination จากรูปที่ 3.27 ซึ่งเป็นกราฟแสดงการเปลี่ยนแปลง Combination ในการจำลองโดยใช้อัลกอริทึม USF เมื่อกำหนดจำนวนโหนด TCP เท่ากับ 40 โหนด และค่าเทรสโฮลด์ HIGH เท่ากับ 5% จะเห็นว่ามีหลายครั้งที่ค่า L_a ลดลงจนต่ำกว่า 1% (ค่าเทรสโฮลด์ LOW) แล้วอัลกอริทึม USF ตัดสินใจลด Combination จาก Combination หมายเลข 5 ไปเป็น Combination หมายเลข 4 แต่ Combination นี้ไม่สามารถกู้ข้อมูลที่สูญหายได้มากพอ และทำให้ค่า L_a สูงขึ้นในช่วงเวลาถัดมา ซึ่งทางที่ดีแล้วในสภาพเครือข่ายเช่นนี้ควรใช้ Combination หมายเลข 5 โดยไม่ต้องมีการเปลี่ยนแปลง



รูปที่ 3.27 อัลกอริทึม USF ได้รับผลกระทบจากการแกว่งของค่า L_a

อัลกอริทึม RCCS มีจุดบกพร่องในส่วนของการเลือก Combination ซึ่งไม่ได้มีการแยกเงื่อนไขในการลดและเพิ่ม Combination จาก Pseudo Code ของอัลกอริทึม RCCS ในรูปที่ 3.11 จะเห็นว่าอัลกอริทึมนี้จะใช้ค่า Reward เพื่อหาค่า L_a' (ค่าทำนายของ L_a) ในแต่ละ Combination และจะเลือก Combination ที่มีค่า L_a' ต่ำกว่าเทรสโฮลด์ HIGH และสูงกว่าเทรสโฮลด์ LOW ซึ่งการใช้เงื่อนไขร่วมกันอย่างนี้อาจทำให้อัลกอริทึมนี้ไม่สามารถเพิ่ม Combination ได้ในบางครั้ง เช่น ถ้า Combination ในขั้นที่สูงกว่า Combination ปัจจุบันทุก Combination ต่างก็มีค่า L_a' ที่ต่ำ

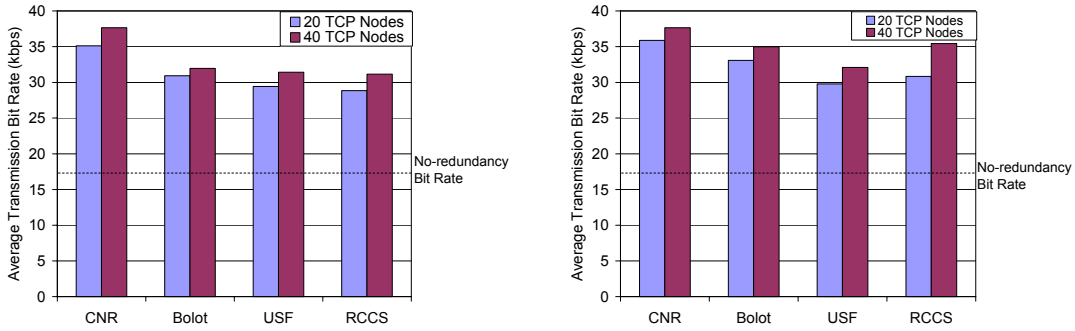
กว่าเทรสโฮลด์ LOW ก็ถือว่าไม่ตรงตามเงื่อนไขแล้ว ซึ่งทำให้ไม่สามารถเพิ่ม Combination ได้ จากรูปที่ 3.28 ซึ่งเป็นกราฟแสดงการเปลี่ยนแปลง Combination ในการจำลองโดยใช้อัลกอริทึม RCCS โดยกำหนดจำนวนโหนด TCP เท่ากับ 40 โหนด และค่าเทรสโฮลด์ HIGH เท่ากับ 5% ซึ่งจะเห็นว่ามีหลายครั้งที่ค่า L_a มีค่าสูงเกินกว่า 5% (ตำแหน่งที่มีเส้นประในแนวตั้ง) แต่กลับไม่มีการเพิ่ม Combination เลย จากกราฟจะเห็นว่าในช่วงเวลาตั้งแต่วินาทีที่ 900 – 940 มีการใช้ Combination หมายเลข 2 ตลอด โดยไม่มีการเพิ่ม Combination เนื่องจากว่า L_a' ของ Combination หมายเลข 3 มีค่าสูงเกินไป (สูงกว่า 5%) ส่วน Combination หมายเลข 4 และ 5 ก็ไม่สามารถเลือกได้เช่นกันเนื่องจากว่า L_a' มีค่าเกินไป (ต่ำกว่า 1%)



รูปที่ 3.28 จุดบกพร่องที่ทำให้อัลกอริทึม RCCS ไม่สามารถเพิ่ม Combination ได้

นอกจากนี้สิ่งที่จะต้องพิจารณาอีกอย่างหนึ่งในการประเมินผลอัลกอริทึมในการควบคุมความผิดพลาดแบบปรับตัวคือ ค่าอัตราบิตเฉลี่ยของการส่งเสียงในการจำลองของแต่ละอัลกอริทึม ซึ่งอัตราบิตจะมากหรือน้อยก็ขึ้นอยู่กับปริมาณข้อมูลซ้ำที่ใช้ ซึ่งเป็นที่แน่นอนว่าการส่งข้อมูลซ้ำปริมาณมากขึ้นทำให้สามารถกู้ข้อมูลที่สูญหายมากขึ้น และการที่อัลกอริทึม CNR มีค่าเฉลี่ย L_a น้อยกว่าอัลกอริทึมอื่น สาเหตุสำคัญก็มาจากอัลกอริทึมนี้มีการลด Combination ช้ากว่าอัลกอริทึมอื่น หรืออาจจะกล่าวอีกนัยหนึ่งว่า อัลกอริทึม CNR ใช้ปริมาณข้อมูลซ้ำมากกว่าอัลกอริทึมอื่นนั่นเอง ซึ่งส่งผลให้อัตราบิตในการส่งเสียงมีค่าสูงตามไปด้วย จากแผนภูมิในรูปที่ 3.29 ถึงแม้ว่าการใช้อัลกอริทึม CNR จะมีอัตราบิตสูงกว่าอัลกอริทึมอื่น แต่ก็สูงกว่าในปริมาณไม่มากนัก ซึ่งอัตราบิตที่เพิ่มขึ้นนี้ถือว่าคุ้มค่ากับการที่อัลกอริทึม CNR สามารถรักษาระดับของค่า L_a ให้ต่ำกว่าอัลกอริทึมอื่นได้ จะเห็นว่าแผนภูมิใน รูปที่ 3.29 ทั้งสองแผนภูมิจะมีเส้นประที่ระบุว่า No-redundancy Bit Rate ซึ่งเส้นประนี้นี้แสดงให้เห็นถึงอัตราบิตที่ใช้ในการส่งเสียงกรณีที่ไม่

มีการใช้ข้อมูลซ้ำ ซึ่งมีค่าเท่ากับ 17.07 กิโลบิตต่อวินาที ซึ่งจะเห็นว่าแต่ละอัลกอริทึมมีอัตราบิตของข้อมูลที่ถือเป็นโอเวอร์เฮดเกือบหนึ่งเท่าตัว

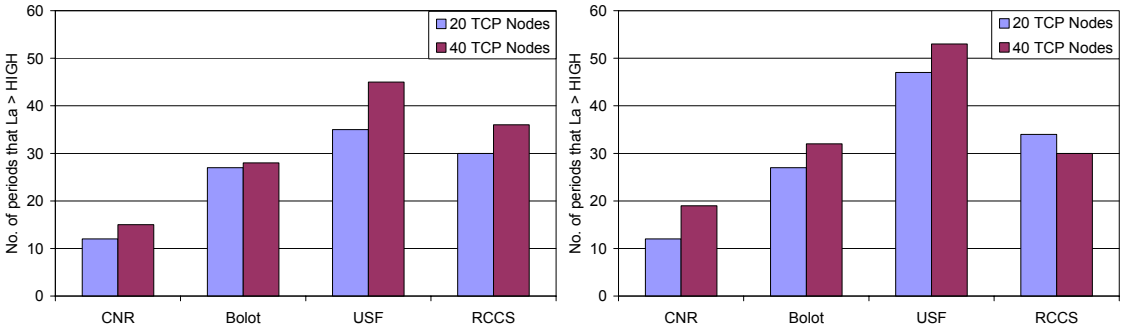


(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

รูปที่ 3.29 ค่าอัตราบิตเฉลี่ยของการส่งเสียงจากการจำลองในแต่ละอัลกอริทึม

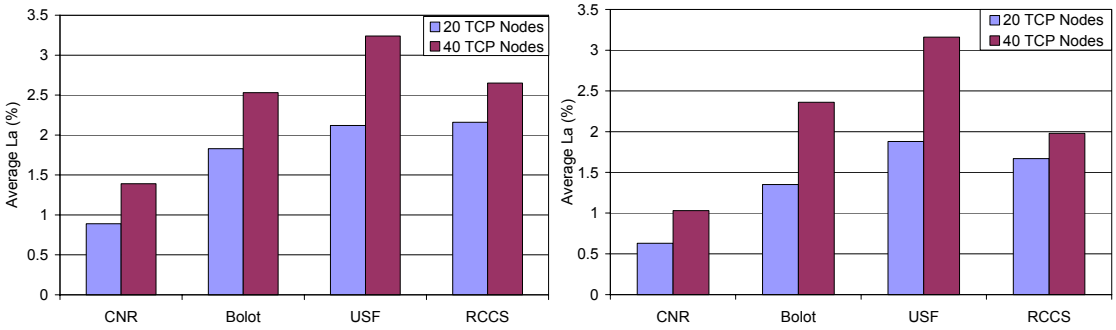
3.5.2 ผลการทดลองตอนที่ 2

ในการทดลองตอนที่ 2 สภาพความคับคั่งของเครือข่ายจะแตกต่างจากตอนที่ 1 โดยเครือข่ายจะมีความคับคั่งมากและน้อยสลับกันไป เนื่องจากโหนดที่ส่ง TCP จะเริ่มส่งและหยุดส่งทุก 200 วินาที แต่อย่างไรก็ตามผลการทดลองของตอนที่ 2 นี้ก็ยังคงเป็นไปในทำนองเดียวกันกับในตอนที่ 1 นั่นคืออัลกอริทึม CNR สามารถควบคุมความผิดพลาดได้ดีที่สุด ซึ่งพิจารณาได้จากแผนภูมิในรูปที่ 3.30 อัลกอริทึม CNR มีจำนวนครั้งที่ L_a สูงกว่าค่าเทรสโฮลด์ HIGH น้อยที่สุด และจากแผนภูมิแสดงค่าเฉลี่ยของ L_a ในรูปที่ 3.31 อัลกอริทึม CNR ก็มีเฉลี่ยของ L_a ต่ำที่สุดในทุกกรณี



(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

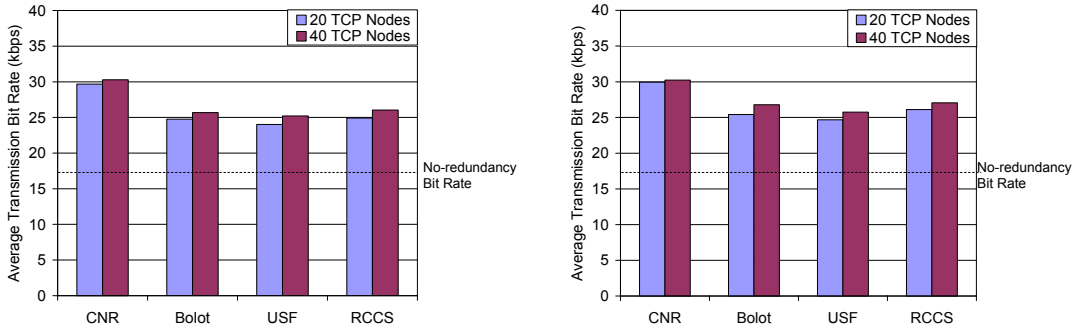
รูปที่ 3.30 จำนวนครั้งที่ L_a เกินเทรสโฮลด์ HIGH (ได้รับ Receiver Report 240 ครั้ง)



(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

รูปที่ 3.31 ค่าเฉลี่ยของ L_a จากการจำลองในแต่ละอัลกอริทึม

สำหรับผลการทดลองในส่วนของค่าอัตราบิตเฉลี่ยก็เป็นไปในทำนองเดียวกับตอนที่ 1 นั่นคืออัตราบิตเฉลี่ยเมื่อใช้อัลกอริทึม CNR มีค่ามากกว่าเมื่อใช้อัลกอริทึมอื่นไม่มากนัก

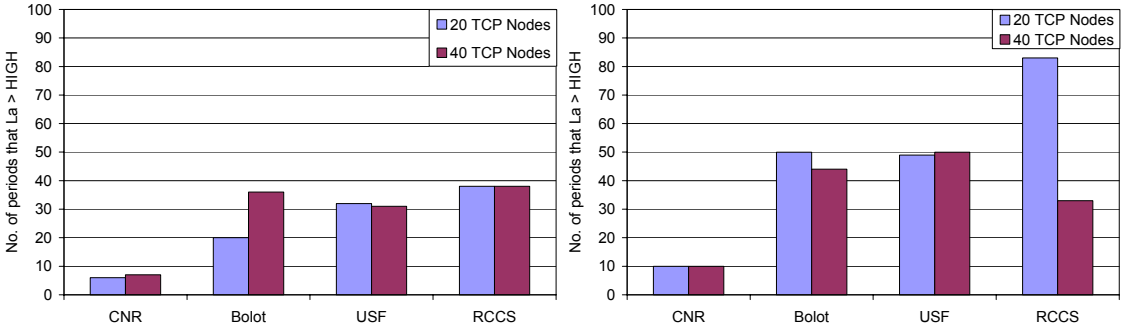


(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

รูปที่ 3.32 ค่าอัตราบิตเฉลี่ยของการส่งเสียงจากการจำลองในแต่ละอัลกอริทึม

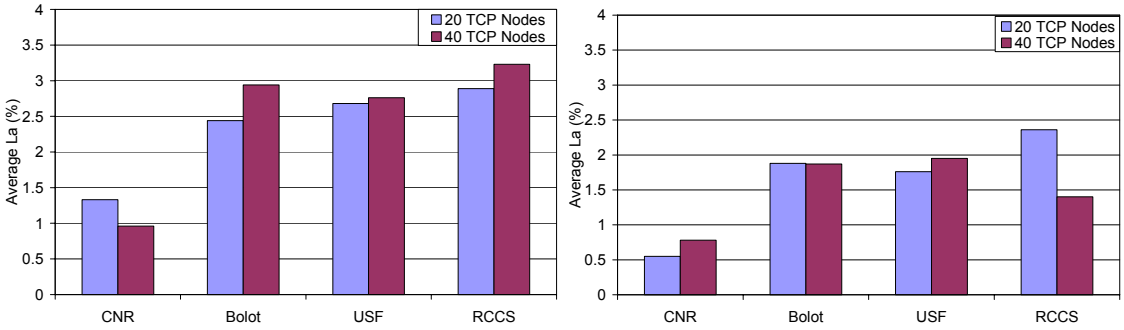
3.5.3 ผลการทดลองตอนที่ 3

ในการทดลองตอนที่ 3 โหนดที่ส่งทราฟฟิก TCP มีเฉพาะโหนด TS_{l1} – TS_{ln} เท่านั้น ส่วนข้อกำหนดอื่น ๆ เหมือนกับข้อกำหนดในตอนที่ 1 โดยผลการทดลองก็ยังเป็นไปในทำนองเดียวกันกับตอนที่ 1 นั่นคือ อัลกอริทึม CNR สามารถควบคุมความผิดพลาดได้ดีที่สุด โดยมีจำนวนครั้งที่ L_a เกินค่าเทรสโฮลด์ HIGH ต่ำสุด รวมทั้งมีค่า L_a เฉลี่ยต่ำที่สุด แต่ก็ยังมีอัตราบิตเฉลี่ยในการส่งเสียงสูงกว่าอัลกอริทึมอื่น



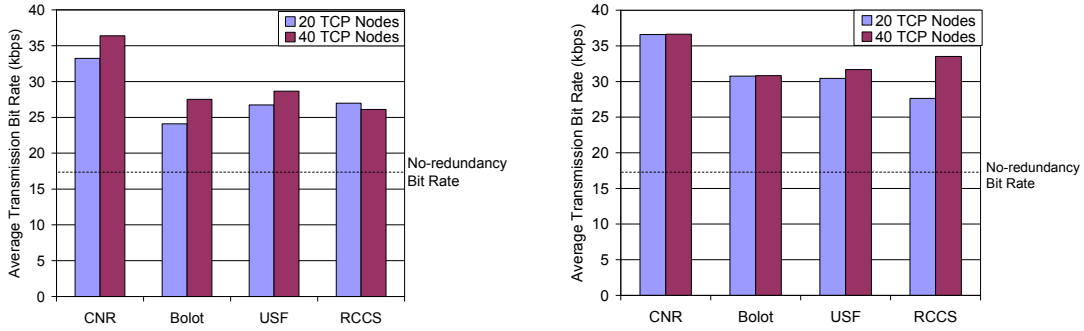
(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

รูปที่ 3.33 จำนวนครั้งที่ L_a เกินเทรสโฮลด์ HIGH (ได้รับ Receiver Report 240 ครั้ง)



(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

รูปที่ 3.34 ค่าเฉลี่ยของ L_a จากการจำลองในแต่ละอัลกอริทึม



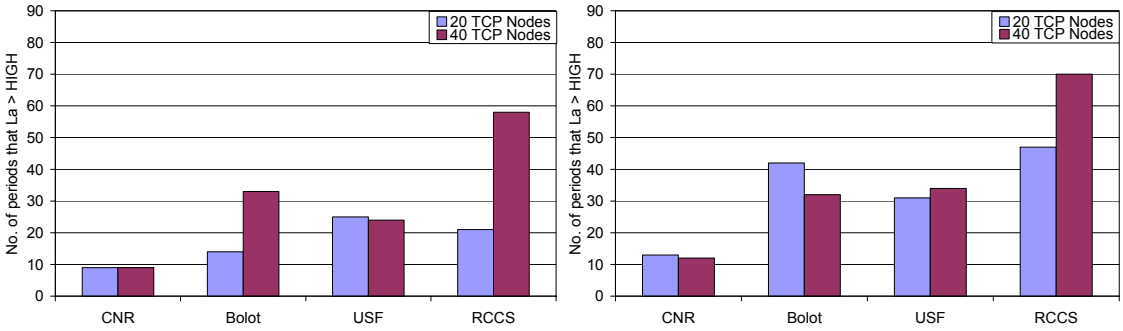
(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

รูปที่ 3.35 ค่าอัตราบิตเฉลี่ยของการส่งเสียงจากการจำลองในแต่ละอัลกอริทึม

3.5.4 ผลการทดลองตอนที่ 4

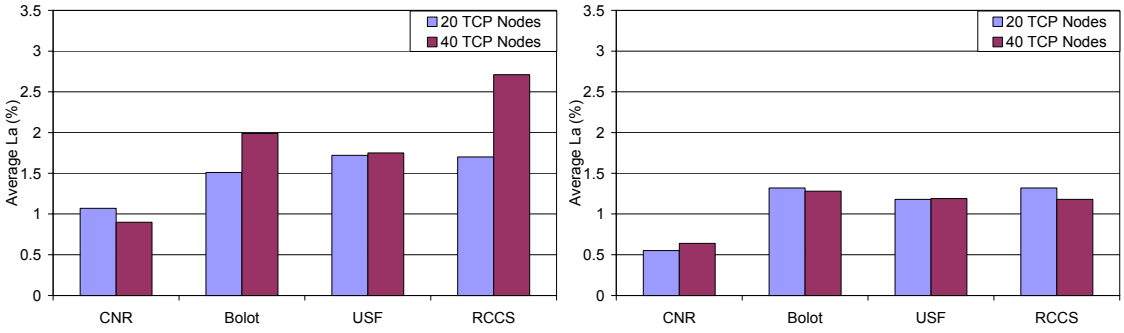
ในการทดลองตอนที่ 4 โหนดที่ส่งทราฟฟิก TCP มีเฉพาะโหนด TS_{l1} – TS_{ln} เท่านั้น ส่วนข้อกำหนดอื่น ๆ เหมือนกับข้อกำหนดในตอนที่ 2 โดยผลการทดลองก็ยังเป็นไปในทำนอง

เดียวกันกับตอนที่ 2 นั่นคือ อัลกอริทึม CNR สามารถควบคุมความผิดพลาดได้ดีที่สุด โดยมีจำนวนครั้งที่ L_a เกินค่าเทรสโฮลด์ HIGH ต่ำที่สุด รวมทั้งมีค่า L_a เฉลี่ยต่ำที่สุด แต่ก็ยังมีอัตราบิตเฉลี่ยในการส่งเสียงสูงกว่าอัลกอริทึมอื่น



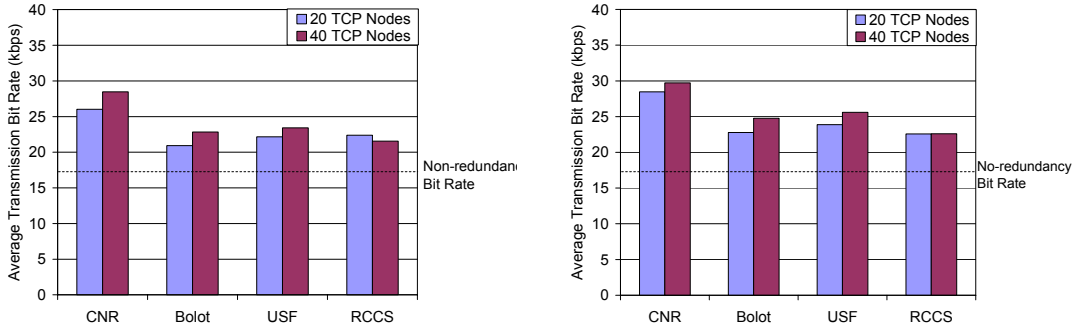
(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

รูปที่ 3.36 จำนวนครั้งที่ L_a เกินเทรสโฮลด์ HIGH (ได้รับ Receiver Report 240 ครั้ง)



(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

รูปที่ 3.37 ค่าเฉลี่ยของ L_a จากการจำลองในแต่ละอัลกอริทึม



(a) เทรสโฮลด์ HIGH = 5% (b) เทรสโฮลด์ HIGH = 3%

รูปที่ 3.38 ค่าอัตราบิตเฉลี่ยของการส่งเสียงจากการจำลองในแต่ละอัลกอริทึม

3.5.5 สรุปผลการทดลอง

จากผลการทดลองทั้ง 4 ตอนเพื่อประเมินผลอัลกอริทึม CNR ในสภาพเครือข่ายที่แตกต่างกัน สามารถสรุปผลการทดลองได้ดังตารางที่ 3.8 โดยผลการทดลองทั้ง 4 ตอนนั้นเป็นไปในทำนองเดียวกัน นั่นคือคือ อัลกอริทึม CNR สามารถควบคุมความผิดพลาดได้ดีที่สุด โดยมีจำนวนครั้งที่ L_a เกินค่าเทรสโฮลด์ HIGH ต่ำสุด รวมทั้งมีค่า L_a เฉลี่ยต่ำที่สุด แต่ก็ยังมีอัตราบิตเฉลี่ยในการส่งเสียงสูงกว่าอัลกอริทึมอื่น สำหรับคอลัมน์ Candidate Algorithm หมายถึงอัลกอริทึมที่มีผลการทดลองเกี่ยวกับปริมาณการสูญหายของข้อมูลใกล้เคียงกับอัลกอริทึม CNR ในการทดลองแต่ละตอน ซึ่งในตารางที่ 3.8 ได้มีการเปรียบเทียบว่าอัตราบิตเฉลี่ยของอัลกอริทึม CNR มีค่าสูงกว่าอัตราบิตของ Candidate Algorithm เท่าใด

ตารางที่ 3.8 สรุปผลการทดลอง

ตอน	เทรสโฮลด์ HIGH (%)	จำนวนครั้งที่ L_a > HIGH	L_a เฉลี่ย	Candidate Algorithm	$R_{CNR} - R_{candidate}$ (kbps)	
					20 TCP Nodes	40 TCP Nodes
1	5	ต่ำสุด	ต่ำสุด	Bolot	4.2	5.69
	3	ต่ำสุด	ต่ำสุด	Bolot	2.8	2.67
2	5	ต่ำสุด	ต่ำสุด	Bolot	4.91	4.62
	3	ต่ำสุด	ต่ำสุด	Bolot	4.53	3.44
3	5	ต่ำสุด	ต่ำสุด	Bolot	9.14	8.86
	3	ต่ำสุด	ต่ำสุด	Bolot	5.84	5.82
4	5	ต่ำสุด	ต่ำสุด	Bolot	5.1	5.63
	3	ต่ำสุด	ต่ำสุด	Bolot	5.69	4.97

3.6 สรุป

ในบทนี้ได้นำเสนอวิธีการควบคุมคุณภาพของสื่อสารเสียง โดยใช้วิธีการควบคุมความผิดพลาดแบบปรับตัวซึ่งเป็นวิธีการที่ใช้พื้นฐานของ FEC และจากการศึกษาอัลกอริทึมของการควบคุมความผิดพลาดแบบปรับตัวที่ได้เสนอก่อนหน้านี้ 3 อัลกอริทึมคือ อัลกอริทึม Bolot, อัลกอริทึม USF และอัลกอริทึม RCCS พบว่าอัลกอริทึมเหล่านี้ยังมีจุดบกพร่องอยู่ และเพื่อที่จะแก้ปัญหาดังกล่าว ในวิทยานิพนธ์นี้จึงได้เสนออัลกอริทึมในการควบคุมคุณภาพแบบปรับตัวขึ้นมาใหม่ชื่อว่าอัลกอริทึม CNR และจากการประเมินผลอัลกอริทึมนี้โดยใช้การจำลองบน NS-2 พบว่าอัลกอริทึมนี้ให้ผลเป็นที่น่าพอใจกว่าอัลกอริทึม Bolot, อัลกอริทึม USF และอัลกอริทึม RCCS

โดยเมื่อใช้อัลกอริทึม CNR จะทำให้จำนวนครั้งที่ค่า L_a สูงกว่าเทรสโฮลด์ HIGH น้อยกว่าอัลกอริทึมอื่น รวมไปถึงค่าเฉลี่ยของ L_a ของอัลกอริทึม CNR ก็ต่ำกว่าอัลกอริทึมอื่นด้วย แต่จุดอ่อนอย่างหนึ่งของอัลกอริทึม CNR ก็คือมีค่าอัตราบิตเฉลี่ยสูงกว่าอัลกอริทึมอื่น ทั้งนี้เนื่องจากว่าอัลกอริทึมนั้นต้องการแก้ปัญหาการแกว่งของค่า L_a จึงมีการลด Combination ช้ากว่าอัลกอริทึมอื่น ส่งผลให้มีการใช้จำนวนข้อมูลซ้ำมากกว่าอัลกอริทึมอื่น ซึ่งเป็นสาเหตุให้อัตราบิตเฉลี่ยของอัลกอริทึม CNR สูงกว่าอัลกอริทึมอื่น แต่อย่างไรก็ตามจากการผลการทดลองจะเห็นว่าอัลกอริทึม CNR มีอัตราบิตสูงกว่าอัลกอริทึมอื่นเพียงเล็กน้อยเท่านั้น และยังถือว่าคุ้มค่ากับการที่อัลกอริทึม CNR สามารถลดผลกระทบจากการสูญหายของแพ็กเก็ตและสามารถกู้ข้อมูลในแพ็กเก็ตที่สูญหายได้ดีกว่าอัลกอริทึมอื่น