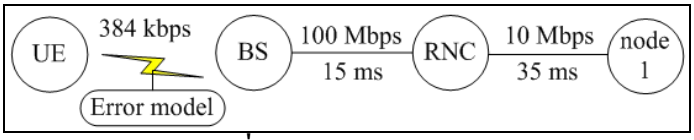
บทที่ 5

การทดลองและผลการทดลอง

เนื้อหาในบทนี้จะกล่าวถึงการสร้างระบบทดสอบ และผลการทดสอบการทำงานที่แสดงให้เห็นประสิทธิภาพการทำงานของระบบจากการใช้แบบจำลองที่ได้กล่าวถึงในบทที่ 4 โดยหัวข้อ 5.1 อธิบายวิธีการทดลองและแบบจำลองที่ใช้ ค่าพารามิเตอร์ รวมถึงรูปแบบเครือข่าย และในหัวข้อ 5.2 จะกล่าวถึงผลการทดลองและการวิเคราะห์ ซึ่งแบ่งออกเป็น เวลาหน่วงในการรับเฟรม เวลาหน่วงในระดับแพ็กเกตย่อย PDU การใช้งานบัฟเฟอร์ RLC การใช้งานช่องสัญญาณ ประสิทธิภาพการทำงานของโปรโตคอล RLC และผลการทำงานของโปรโตคอลเมื่อเปลี่ยนกลไกการเลือกส่งข้อมูลภายในชั้น RLC

5.1 วิธีการทดลอง

การทดลองประสิทธิภาพการทำงานของระบบ มาทดสอบภายใต้เครือข่ายจำลองที่ซึ่งประกอบด้วยโหนด 4 โหนดที่ทำหน้าที่เป็น UE, Base station, RNC และโหนดบนเครือข่ายอินเตอร์เน็ต ซึ่งลิงค์ระหว่างแต่ละโหนดมีแบนด์วิดท์และเวลาที่ใช้ส่งข้อมูล ดังแสดงในรูปที่ 5.1 ระหว่าง UE และ BS เชื่อมต่อกันผ่านช่องสัญญาณ DCH ของ UMTS ที่ความเร็ว 384 kbps



รูปที่ 5.1 เครือข่ายจำลอง

ในการทดลองสมมติเหตุการณ์ที่ UE1 ทำการร้องขอเปิดดูไฟล์วิดีโอจาก node1 ที่อยู่บนอินเทอร์เน็ต ซึ่งวิดีโอที่ทำการร้องขอมีความยาวมากกว่า 5 นาที เพื่อเป็นการทดสอบผลกระทบของการทำงานของระบบเมื่อไฟล์วิดีโอที่ต้องการมีเวลายาวนาน

แบบจำลองและการทดลองทั้งหมดกระทำอยู่ภายใต้ข้อจำกัดของอุปกรณ์ที่ทดสอบดังต่อไปนี้

- เครื่องคอมพิวเตอร์แบบตั้งโต๊ะจำนวน 1 เครื่อง CPU Intel Celelon D2.4G, Ram 896 MB
- กำหนดให้แบบจำลองถูกสร้างอยู่บน virtual mechineที่เป็นระบบปฏิบัติ

การ Linux Tle 5.0 ได้รับการให้สิทธิใช้หน่วยความจำ 512 MB

- Virtual mechine รันอยู่บนเครื่องโฮสต์ที่เป็นระบบปฏิบัติการ Windows XP

5.1.1 การอธิบายแบบจำลอง (Simulation description)

ไฟล์วิดีโอที่ใช้ในการทดลองเป็นวิดีโอจำลอง MPEG-4 กล่าวคือ ไม่ได้เป็นข้อมูลวิดีโอจริงแต่เป็น trace file ที่ถูกสร้างขึ้นเพื่อใช้ในการส่งบน ns-2 ซึ่งจำลองให้มีชนิดของเฟรมวิดีโอเป็น i-frame และ p-frame เช่นเดียวกับวิดีโอจริงเมื่อถูกบีบอัดเป็นชนิด mpeg-4 แต่ความยาวของเฟรมแต่ละชนิดจะคงที่เสมอ กล่าวคือ ความยาวของ i-frame เท่ากับ 9000 ไบท์ และความยาวของ p-frame เท่ากับ 2000 ไบท์ ทั้งนี้ก็เพื่อความสะดวกในการวิเคราะห์ผล ในการทดลองจะใช้ไฟล์วิดีโอที่มี key frame interval เท่ากับ 5 นั่นคือ จะมีข้อมูลชนิดที่เป็น i-frame ทุก ๆ 5 เฟรม และเฟรมอื่น ๆ จะเป็น p-frame รายละเอียดข้อมูลวิดีโอที่ใช้ในการทดลองแสดงไว้ในตารางที่ 5.1

ตารางที่ 5.1 พารามิเตอร์สำหรับวิดีโอ MPEG-4

จำนวนเฟรมที่ส่ง (เฟรม)	1200
Key frame interval (เฟรม)	5
ความยาวของ I-frame (ไบท์)	9000
ความยาวของ P-frame (ไบท์)	2000
อัตราการส่ง (เฟรมต่อวินาที)	10
เวลาที่ใช้ส่ง (วินาที)	120

การทดลองกำหนดให้ข้อมูลวิดีโอที่ใช้รับ-ส่งคงที่ กล่าวคือ ใช้ข้อมูลวิดีโอเดียวกันตลอดการทดลอง แต่ทำการเปลี่ยนค่า poll_timer เป็น 50, 200 และ 500 มิลลิวินาที และเปลี่ยนค่า error rate ตั้งแต่ 0 ไปจนถึง 25% เนื่องจากจุดประสงค์ของการทดลองเพื่อวิเคราะห์ว่ามีปัจจัยใดบ้างที่มีผลต่อประสิทธิภาพการทำงานของโปรโตคอลและเหตุใดผลประสิทธิภาพจึงเป็นเช่นนั้น และนอกจากนี้ก็เพื่อดูผลกระทบของการกำหนดค่า poll_timer ซึ่งเป็นพารามิเตอร์หนึ่งของกลไกการ poll ในชั้น RLC ฝั่งส่งที่มีต่อการส่งข้อมูลเมื่อ error rate มีค่าต่าง ๆ พารามิเตอร์ที่ใช้ในการทดลองแสดงได้ดังตารางที่ 5.2

ตารางที่ 5.2 พารามิเตอร์ที่ใช้ในการจำลอง

<table>
<tr><td>App</td><td colspan="5">MPEG-4</td></tr>
<tr><td>RTP</td><td>RTP Header size (bytes)</td><td colspan="4">12</td></tr>
<tr><td>UDP</td><td>Maximum Segment Size (bytes)</td><td colspan="4">1460</td></tr>
<tr><td></td><td>UDP Header size (bytes)</td><td colspan="4">8</td></tr>
<tr><td>IP</td><td>IP Header size (bytes)</td><td colspan="4">20</td></tr>
<tr><td>RLC</td><td>RLC Mode</td><td colspan="4">AM</td></tr>
<tr><td></td><td>Payload size (bytes)</td><td colspan="4">40</td></tr>
<tr><td></td><td>RLC header (bytes)</td><td colspan="4">2</td></tr>
<tr><td></td><td>Transmission window (PDUs)</td><td colspan="4">1024</td></tr>
<tr><td>MAC</td><td>MAC Header (bytes)</td><td colspan="4">0</td></tr>
<tr><td></td><td>MAC Multiplexing</td><td colspan="4">Not required for DCH</td></tr>
<tr><td>PHY</td><td rowspan="4">Physical Channel Type</td><td colspan="4">DCH</td></tr>
<tr><td></td><td colspan="2">Uplink</td><td colspan="2">Downlink</td></tr>
<tr><td></td><td>Bit rate (kbps)</td><td>TTI (ms)</td><td>Bit rate (kbps)</td><td>TTI (ms)</td></tr>
<tr><td></td><td>384</td><td>10</td><td>384</td><td>10</td></tr>
<tr><td></td><td>Transport BLER</td><td colspan="4">0 - 25%</td></tr>
<tr><td></td><td>Error Model</td><td colspan="4">Exponential Distribution</td></tr>
</table>

นอกจากพารามิเตอร์สำหรับสิ่งแวดล้อมระบบโดยรวมตามตารางที่ 5.1 และตารางที่ 5.2แล้ว จำเป็นต้องมีการกำหนดทำงานและพารามิเตอร์ภายในโปรโตคอล RLC ด้วยเช่นกัน ซึ่งค่าพารามิเตอร์ที่กำหนดใช้งานแสดงไว้ในตารางที่ 5.3 และตารางที่ 5.4

ตารางที่ 5.3 การกำหนด STATUS และ Poll timer สำหรับช่องสัญญาณ DCH 384kbps

โครงแบบการจำลอง RLC UMTS	✕ ไม่ใช้งาน ✓ ใช้งาน
ฝั่งส่ง	
Last PDU	✓
Poll Timer (มิลลิวินาที)	50, 200, 500
Poll every X PDU	✓

โครงแบบการจำลอง RLC UMTS	✕ ไม่ใช้งาน ✓ ใช้งาน
Poll every X SDU	✕
Last PDU in retransmission Buffer	✓
X% of transmitted window [0...100]	✕
Periodic Poll (มิลลิวินาที)	✕
Poll prohibit (มิลลิวินาที)	✕
ฝั่งรับ	
EPC Timer	✕
Check for Missing PDU	✓
STATUS every X PDU	✕
STATUS every X SDU	✕
X% of receiver window [0...100]	✕
Periodic STATUS (มิลลิวินาที)	✕
STATUS prohibit (มิลลิวินาที)	150

ตารางที่ 5.4 ค่าพารามิเตอร์สำหรับโปรโตคอล RLC ที่ทำงานในโหมด AM

พารามิเตอร์	ค่าที่ใช้
ack_mode_	negative
maxRBSize_ (kbytes)	100
poll_PDU_	256
poll_timeout_ (มิลลิวินาที)	50, 200, 500
stprob_timeout_ (มิลลิวินาที)	150
TTI_ (มิลลิวินาที)	10
length_indicator_ (บิต)	7
min_concat_data_ (แพ็กเกต)	3
max_ack_delay_	10ms

พารามิเตอร์จากตารางที่ 5.4 สามารถอธิบายได้ดังนี้

- *ack_mode_* เป็นพารามิเตอร์ที่มีผลเกี่ยวข้องกับ Poll Timer กล่าวคือ Poll timer จะเริ่มนับเมื่อ AMD PDU ที่มีการกำหนดบิต poll ถูกรับที่ชั้น MAC และถ้า x เป็น sequence number ของ AMD PDU นี้ การที่ timer จะหยุดนับจะเกิดขึ้นเมื่อได้รับ acknowledgement ของ AMD PDU ที่มี sequence number เท่ากับ x สำหรับกรณี ack_mode_ เป็น negative

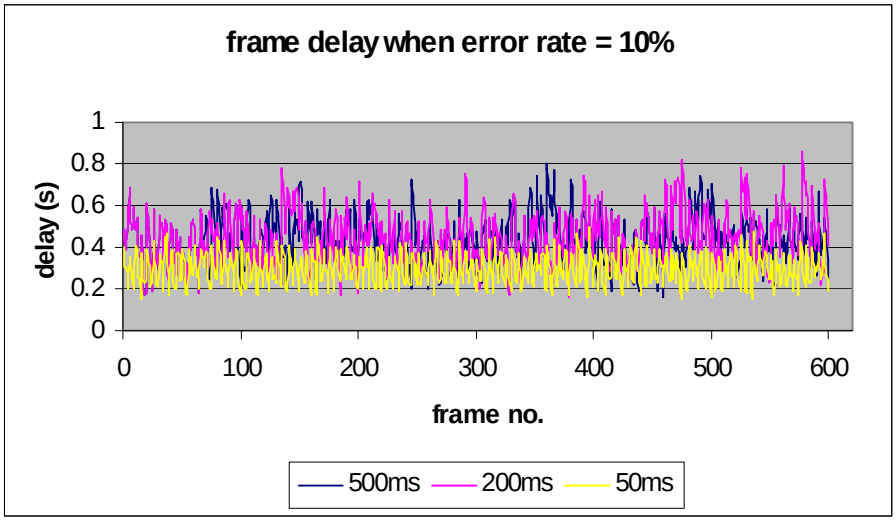
แต่สำหรับ ack_mode_ ที่กำหนดเป็น positive นั้น timer จะหยุดนับก็ต่อเมื่อได้รับ acknowledgement ของ AMD PDUs ทั้งหมดที่มี sequence number จนถึงค่า x (รวม x ด้วย)

- *maxRBSize_* คือ ค่าขนาดของ Radio Bearer หรือเป็นขนาดปริมาณความสามารถที่ RLC จะสามารถรับข้อมูลได้ หากแพ็กเกต SDU ที่จะส่งเข้าสู่ RLC มีขนาดใหญ่เกินกว่าค่านี้ SDU นั้นจะถูก drop กลายเป็นข้อมูลที่สูญหาย
- *poll_PDU_* เมื่อ PDU ถูกส่งเท่ากับจำนวนที่กำหนด จะมีการส่งบิต poll เพื่อร้องขอ acknowledgement ตามกลไกการ poll
- *poll_timeout_* ค่า poll timer ที่มีการกำหนดใช้งานในการทดลอง
- *length_indicator_* จำนวนบิต LI ใน header ของ AM PDU
- *min_concat_data_* จำนวนแพ็กเกตต่ำสุดที่อนุญาตให้ทำการต่อกับ PDU ที่กำลังสร้างได้ หากเนื้อที่ PDU ที่กำลังสร้างเหลือไม่เพียงพอสำหรับการนำแพ็กเกตถัดไปอย่างน้อย 3 แพ็กเกตมาต่อ ก็จะไม่อนุญาตให้เกิดการต่อแพ็กเกตใน PDU นั้น
- *max_ack_delay_* กำหนดเวลาช่วงระยะห่างอย่างน้อยสุดระหว่างการส่งแพ็กเกต acknowledge แต่ละครั้ง หาก acknowledge แพ็กเกตใหม่ต้องการส่งแต่ช่วงเวลาที่ห่างจาก acknowledge ก่อนหน้ายังต่ำกว่าค่าที่กำหนด acknowledge แพ็กเกตใหม่นี้ก็จะไม่อนุญาตให้เกิดการส่งขึ้น

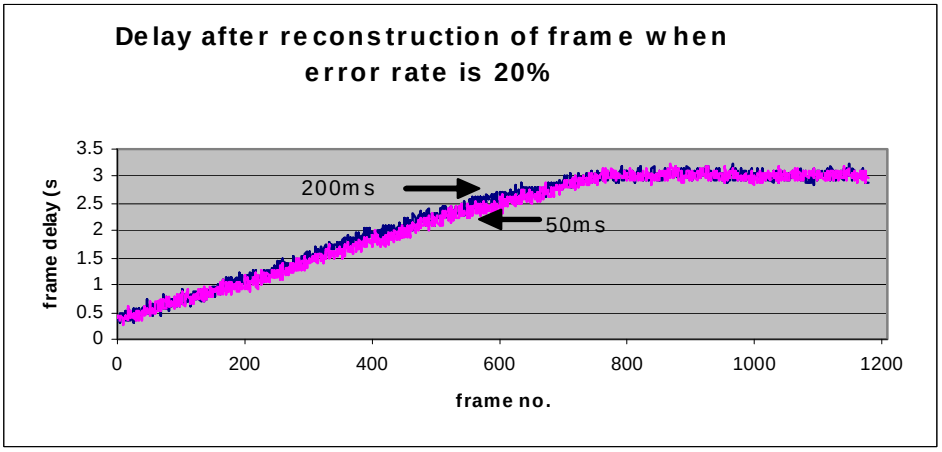
5.2 ผลการทดลองและวิเคราะห์ผลการทดลอง

5.2.1 เวลาหน่วงในการรับเฟรม (Received frame delay)

ในหัวข้อนี้แสดงผลการทดลองให้เห็นถึงผลกระทบของ poll timer ที่มีต่อเวลาหน่วงในการรับเฟรมที่ UE เมื่อทดลองใช้ค่า poll timer ที่แตกต่างกันแต่มีอัตราความผิดพลาดในช่องสัญญาณ (link error rate) เท่ากัน และเมื่อใช้อัตราความผิดพลาดในช่องสัญญาณต่างกันแต่ให้ค่า poll timer เท่ากัน ในกรณีแรกแสดงผลได้ตามรูปที่ 5.2 ซึ่งพบว่า poll timer ค่าต่ำกว่าจะให้ค่า frame delay ดีกว่าแม้ว่าอัตราความผิดพลาดในช่องสัญญาณจะเพิ่มขึ้น แต่ไม่ทุกกรณีไป ดังผลการทดลองตามรูปที่ 5.3 เมื่อ error rate เพิ่มเป็น 20% ค่า poll timer ใด ๆ ไม่มีผลเวลาหน่วงในการรับเฟรม นั่นหมายถึงว่าจะมีข้อจำกัดของ error rate ที่ poll timer สามารถช่วยเสริมความสามารถการทำงานของระบบได้ ค่าเวลาหน่วงเฟรมนี้มีความสำคัญมากสำหรับการเล่นหรือการแสดงผลวิดีโอที่ฝั่งรับเนื่องจากหากเวลาหน่วงมีค่ามากกว่าเวลา play back buffer ฝั่งรับก็จะคิดว่าเฟรมนั้นได้สูญหายไปแม้ว่ามันจะไม่ได้สูญจริงก็ตาม เป็นผลให้ภาพวิดีโอที่ผู้ใช้งานมองเห็นไม่ราบเรียบ ณ เวลานั้น ๆ แต่ในการจำลอง ค่าเวลา play back buffer ไม่ได้ถูกกำหนดนำมาใช้ในการเปรียบเทียบผล

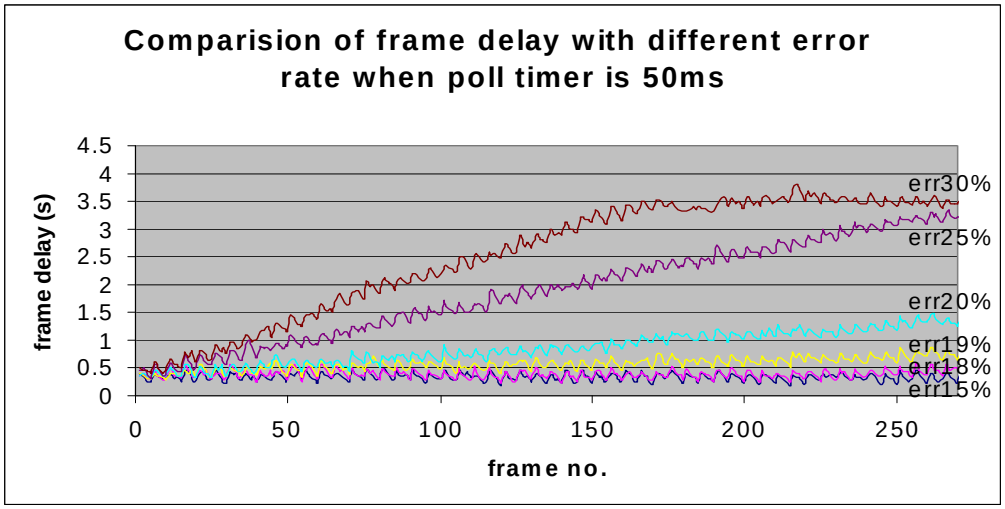


รูปที่ 5.2 เปรียบเทียบค่า frame delay เมื่อค่า poll timer แตกต่างกัน ที่ 10% link error rate

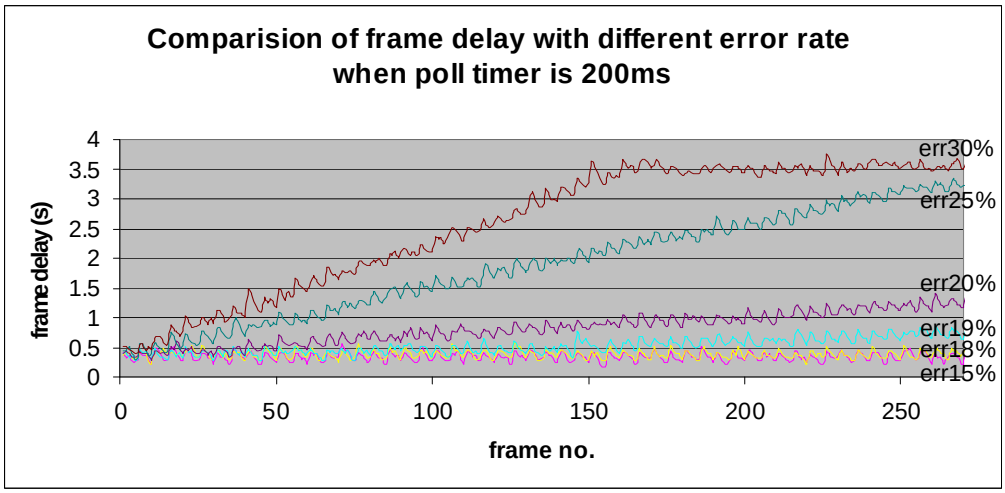


รูปที่ 5.3 เปรียบเทียบค่า frame delay เมื่อค่า poll timer แตกต่างกัน ที่ 20% link error rate

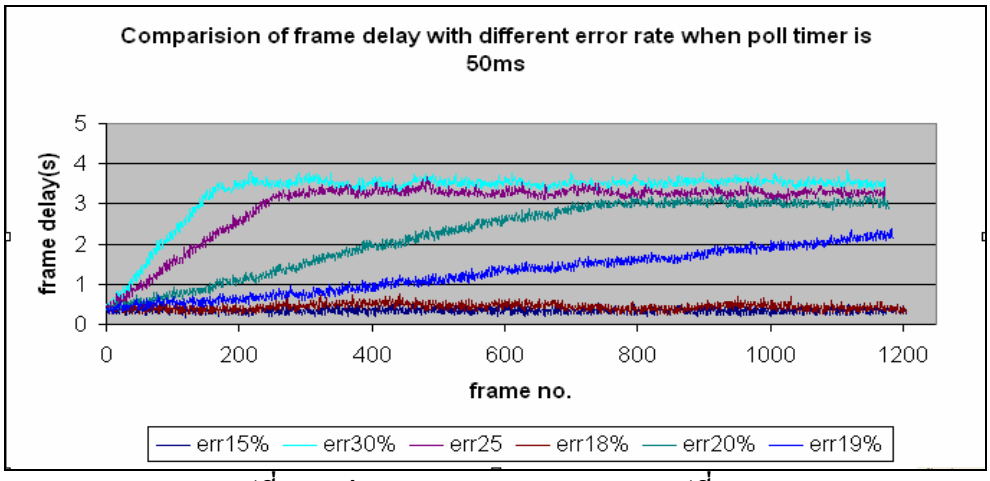
และในกรณีที่สองสำหรับการทดลองแปรเปลี่ยนค่า error rate แต่ใช้ค่า poll timer เท่ากันสามารถมองเห็นค่า error rate สูงสุดที่ความถี่ในการ poll สามารถมีผลต่อเวลาหน่วงเฟรมรูปที่ 5.4 และรูปที่ 5.5 แสดงผลเมื่อกำหนดค่า poll timer คงที่แล้วพยายามเพิ่มค่า error rate ในแต่ละครั้งของการทดลองพบว่า poll timer ทั้ง 50 ms และ 200 ms สามารถให้ค่าเวลาหน่วงในการรับเฟรมต่ำเฉพาะเมื่อ error rate มีค่าไม่มากกว่า 18% ค่าเวลาหน่วงเฟรมจะสูงขึ้นถ้า error rate เพิ่มขึ้น แต่ความถี่ในการ poll สูงเป็นผลให้มีการส่ง status report packets จำนวนมากจากฝั่งรับกลับไปยังฝั่งส่ง นั่นคือแบนด์วิดท์ในช่องสัญญาณย้อนกลับ (backward channel) ถูกใช้ไปจำนวนมากเช่นกัน แต่ในแบบจำลองไม่ได้มีการตรวจวัดค่านี้ เนื่องจากสนใจเฉพาะการสื่อสารในช่องสัญญาณไปข้างหน้า (forward channel) เท่านั้น ดังนั้นผลการทดลองจึงไม่ได้แสดงว่าแบนด์วิดท์ในช่องสัญญาณย้อนกลับเป็นอย่างไร



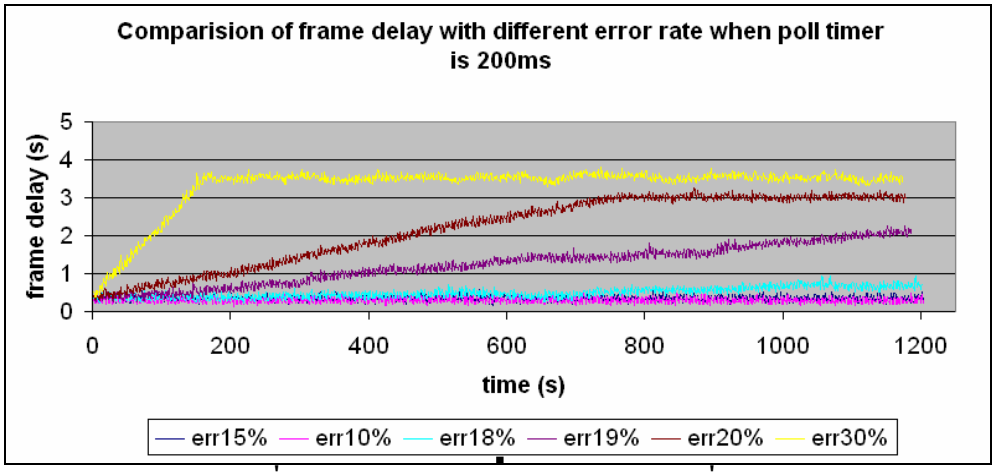
รูปที่ 5.4 เปรียบเทียบค่า frame delay เมื่อให้ link error rate ต่างกัน ที่ 50ms poll timer



รูปที่ 5.5 เปรียบเทียบค่า frame delay เมื่อให้ link error rate ต่างกัน ที่ 200ms poll timer



รูปที่ 5.6 ส่วนขยายผลการทดลองจากรูปที่ 5.4

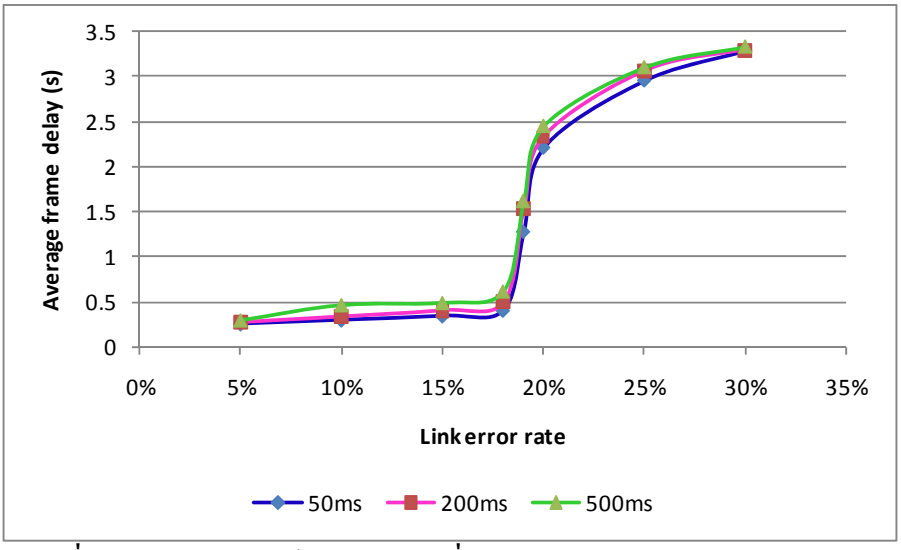


รูปที่ 5.7 ส่วนขยายผลการทดลองจากรูปที่ 5.5

รูปที่ 5.6 และรูปที่ 5.7 แสดงส่วนขยายผลการทดลองของรูปที่ 5.4 และรูปที่ 5.5 ซึ่งทำให้เห็นแนวโน้มการทำงานได้ชัดเจนขึ้น โดยจากรูปที่ 5.2 ถึงรูปที่ 5.5 สามารถสรุปเป็นค่าเฉลี่ยได้ดังตารางที่ 5.5 ซึ่งแสดงให้เห็นว่า การเพิ่มความถี่ในการ poll ไม่ช่วยลดค่าเฉลี่ยเวลาหน่วงเฟรมเมื่อ error rate สูงถึง threshold หนึ่ง ตัวอย่างเช่น 19% ตามรูปที่ 5.8

ตารางที่ 5.5 เปรียบเทียบค่าเฉลี่ยเวลาหน่วงเฟรม

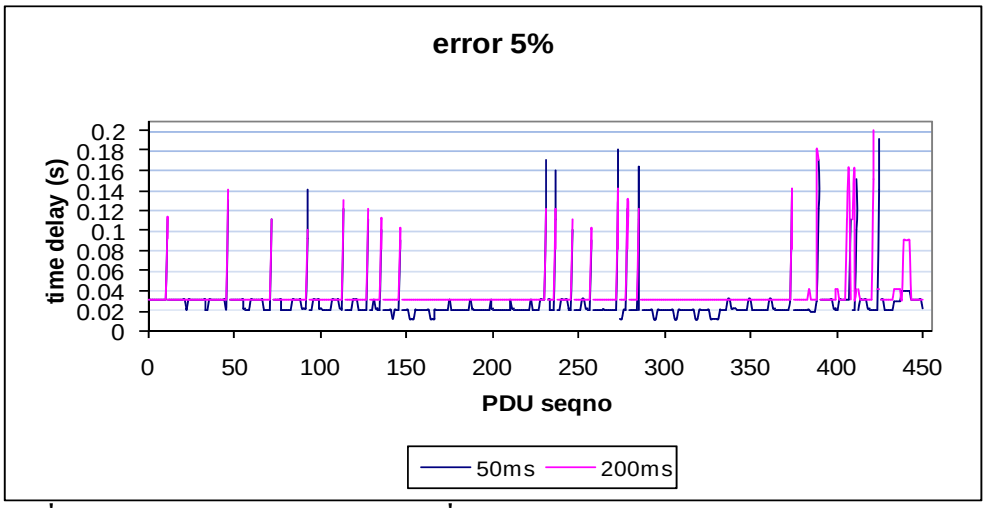
Poll timer (ms)	Average frame delay (s)							
	Err5%	10%	15%	18%	19%	20%	25%	30%
50	0.257	0.299	0.345	0.408	1.282	2.214	2.959	3.292
200	0.280	0.343	0.412	0.504	1.540	2.340	3.065	3.302
500	0.291	0.460	0.485	0.613	1.616	2.451	3.102	3.331



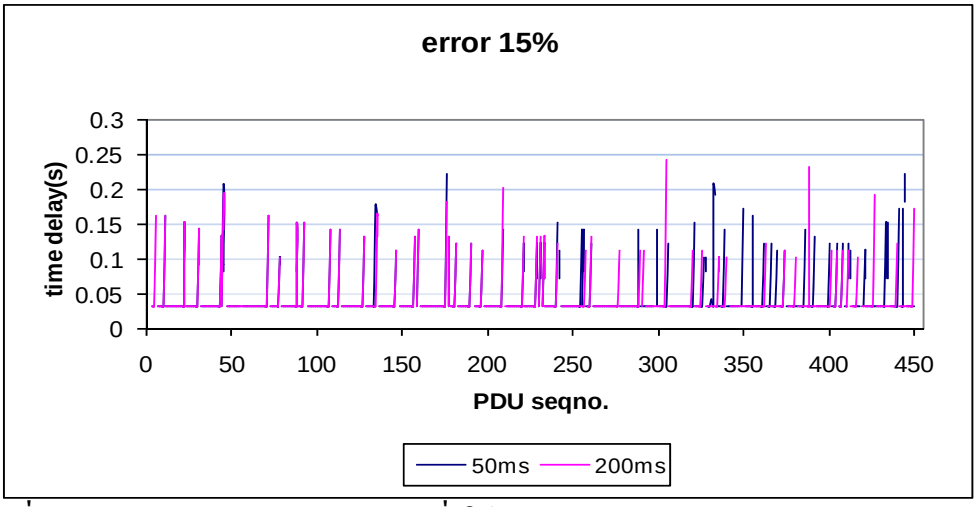
รูปที่ 5.8 ความสัมพันธ์ระหว่างค่าเฉลี่ยเวลาหน่วงเฟรมกับ link error rate เมื่อให้ poll timer ต่างกัน

5.2.2 เวลาหน่วงในระดับแพ็กเกตย่อย PDU

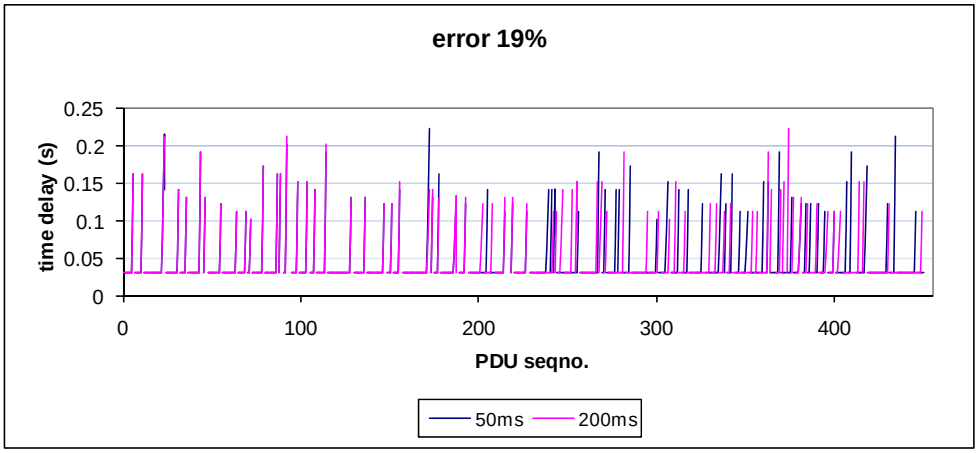
อันเนื่องมาจากว่าเมื่อช่องสัญญาณมี error rate สูงจะทำให้มีแพ็กเกตจำนวนมากอยู่ค้างในบัฟเฟอร์ของ RLC เพื่อรอคอยการส่งซ้ำ แต่ละ PDU จำเป็นต้องคอยจนกว่าส่วนของแพ็กเกตก่อนหน้าทั้งหมดจะถูกรับสมบูรณ์โดยไม่มีความผิดพลาด เมื่อ error rate เพิ่มขึ้น จำนวนครั้งของการส่งซ้ำก็จะเพิ่มขึ้นด้วย โดยสามารถยืนยันได้ด้วยผลการวัดเวลาหน่วงในระดับแพ็กเกตย่อย PDU ตามรูปที่ 5.9 ถึงรูปที่ 5.11ที่แสดงให้เห็นว่าเมื่อ error rate เพิ่มขึ้น จะมี PDU ที่ต้องใช้เวลาสูงเพื่อทำการส่งให้สำเร็จจำนวนมากขึ้น ตามรูปที่ 5.9 สำหรับกรณีที่ error rate ต่ำกว่าค่า threshold เวลาหน่วงของ PDU ที่ถูกส่งด้วยค่า poll timer ต่ำกว่า จะใช้เวลาในการส่งน้อยกว่า



รูปที่ 5.9 เวลาหน่วงของแต่ละ PDU เมื่อให้ค่า poll timer ต่างกัน กรณี error rate 5%



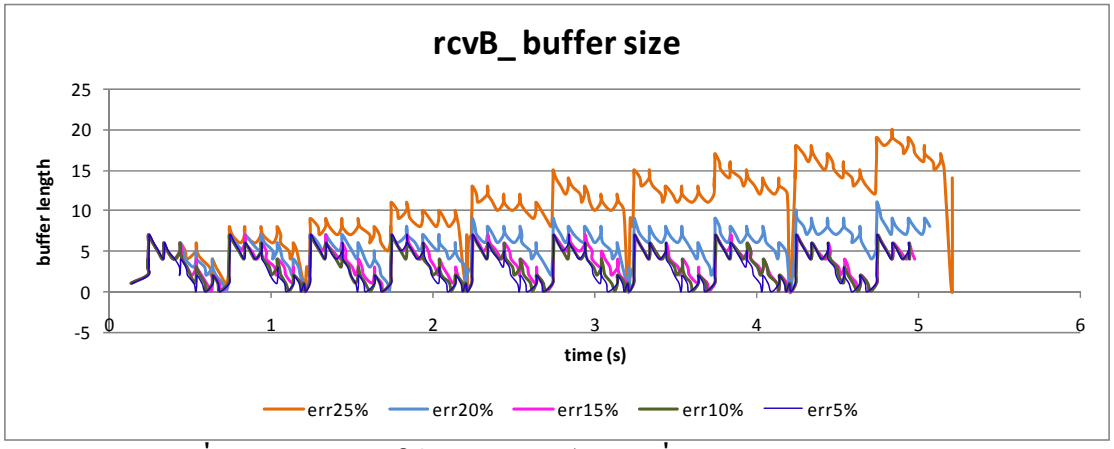
รูปที่ 5.10 เวลาหน่วงของแต่ละ PDU เมื่อให้ค่า poll timer ต่างกัน กรณี error rate 15%



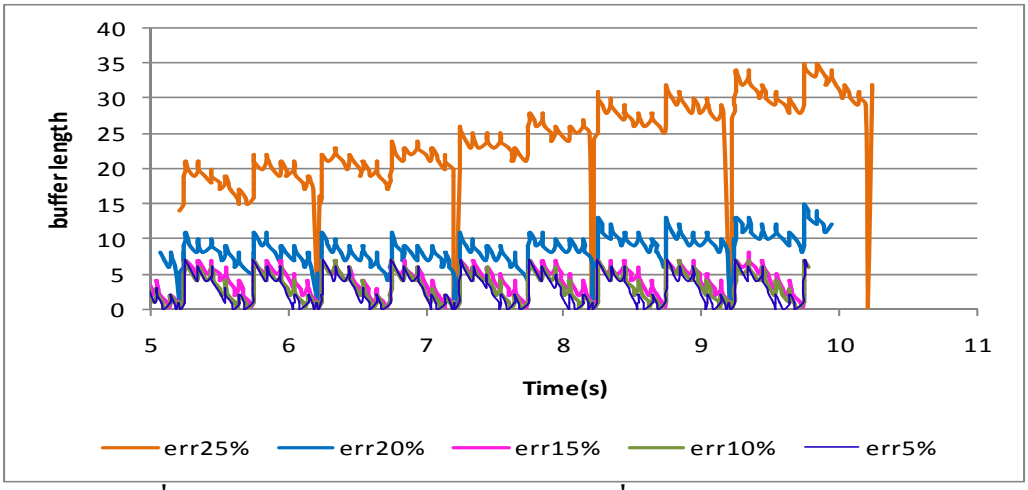
รูปที่ 5.11 เวลาหน่วงของแต่ละ PDU เมื่อให้ค่า poll timer ต่างกัน กรณี error rate 19%

5.2.3 การใช้งานบัฟเฟอร์ RLC

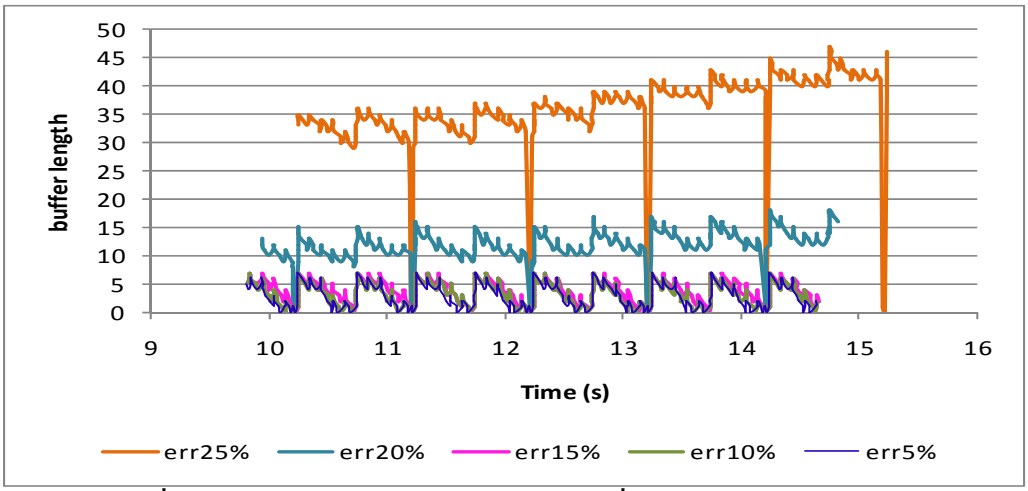
เมื่อ PDU จำเป็นต้องใช้เวลานานขึ้นในการส่งเป็นผลให้ PDU นั้น ๆ จะต้องค้างในบัฟเฟอร์ส่งเป็นเวลายาวนานขึ้น ทำให้ขนาดการใช้งานบัฟเฟอร์ RLC เพิ่มสูงขึ้นด้วย เนื่องจากไม่สามารถลบ PDU นั้นออกไปจากบัฟเฟอร์ได้เพราะไม่ได้รับการยืนยันจากฝั่งรับว่า PDU นั้น ๆ ถูกทำการส่งได้เสร็จสมบูรณ์ ดังรูปที่ 5.12 ถึงรูปที่ 5.19 จะเห็นว่าขนาดของบัฟเฟอร์จะโตขึ้นเรื่อย ๆ จนคงที่ ณ เวลาหนึ่ง นั่นเป็นผลให้เวลาหน่วงเฟรมที่ได้รับคงที่ไปด้วย (1 แพ็กเกตในบัฟเฟอร์มีขนาด 1500 ไบท์)



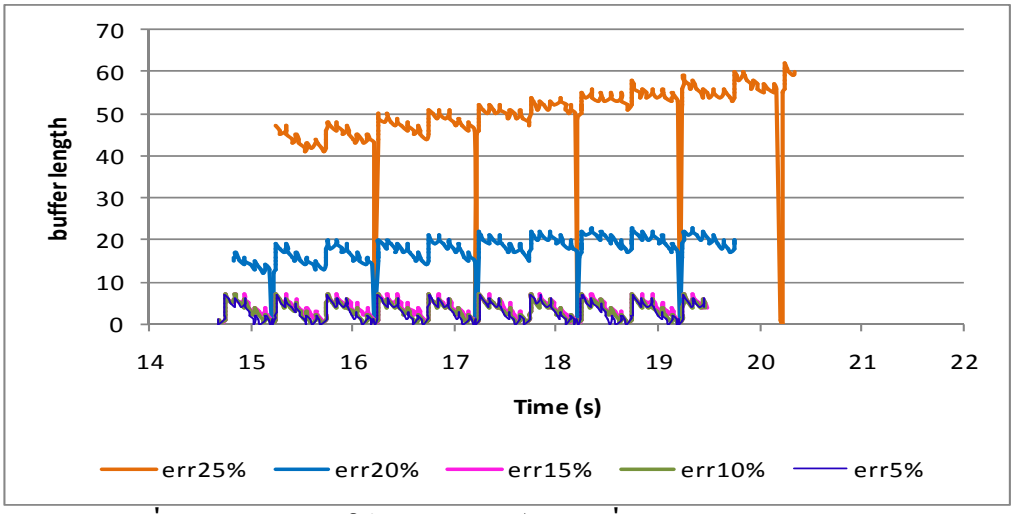
รูปที่ 5.12 ขนาดการใช้งานบัฟเฟอร์ RLC ที่ poll timer ค่าต่าง ๆ (1)



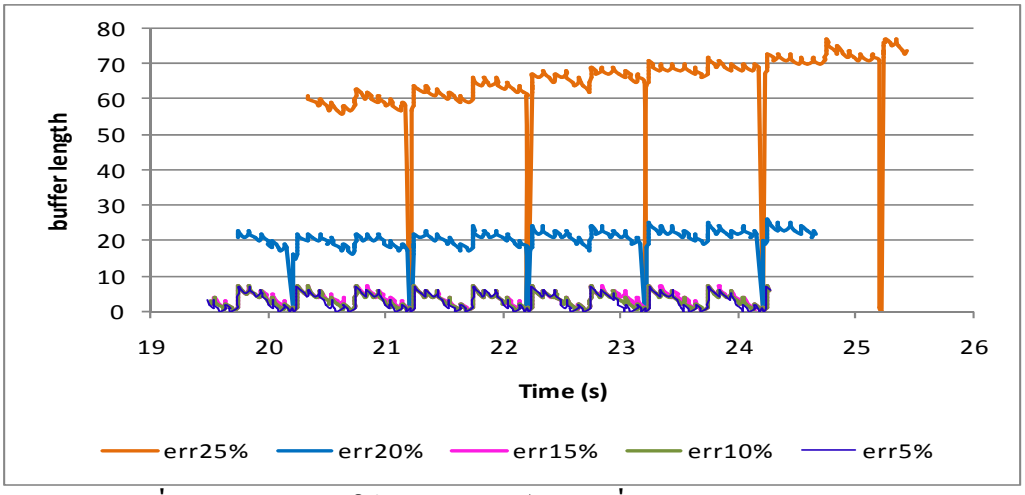
รูปที่ 5.13 ขนาดการใช้งานบัฟเฟอร์ RLC ที่ poll timer ค่าต่าง ๆ (2)



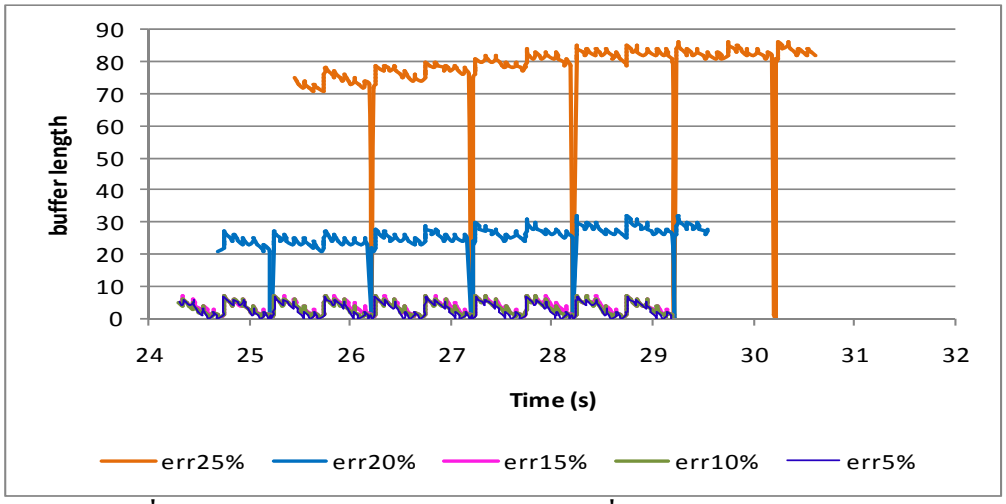
รูปที่ 5.14 ขนาดการใช้งานบัฟเฟอร์ RLC ที่ poll timer ค่าต่าง ๆ (3)



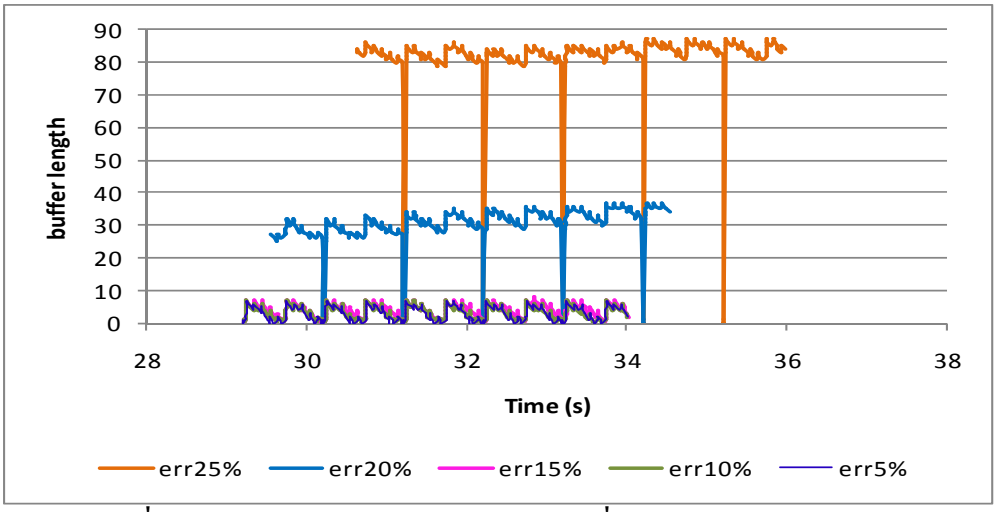
รูปที่ 5.15 ขนาดการใช้งานบัฟเฟอร์ RLC ที่ poll timer ค่าต่าง ๆ (4)



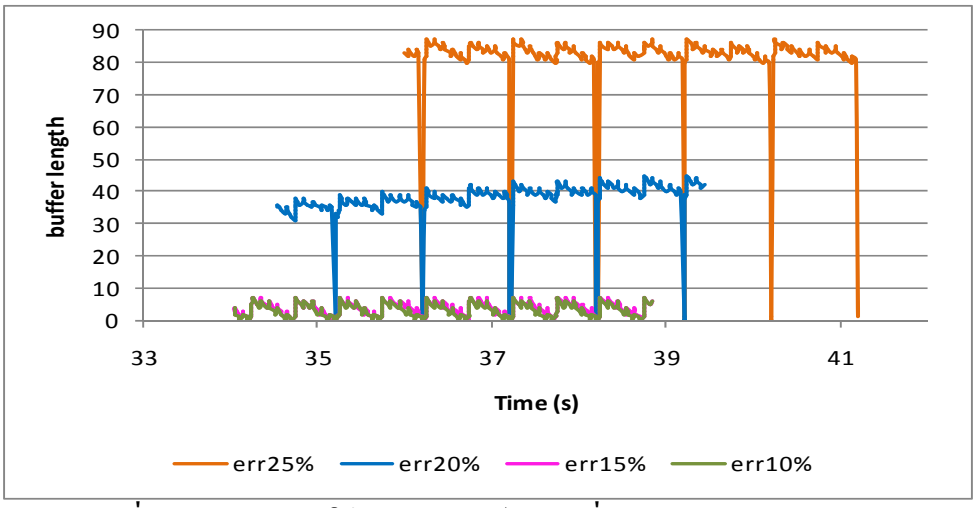
รูปที่ 5.16 ขนาดการใช้งานบัฟเฟอร์ RLC ที่ poll timer ค่าต่าง ๆ (5)



รูปที่ 5.17 ขนาดการใช้งานบัฟเฟอร์ RLC ที่ poll timer ค่าต่าง ๆ (6)



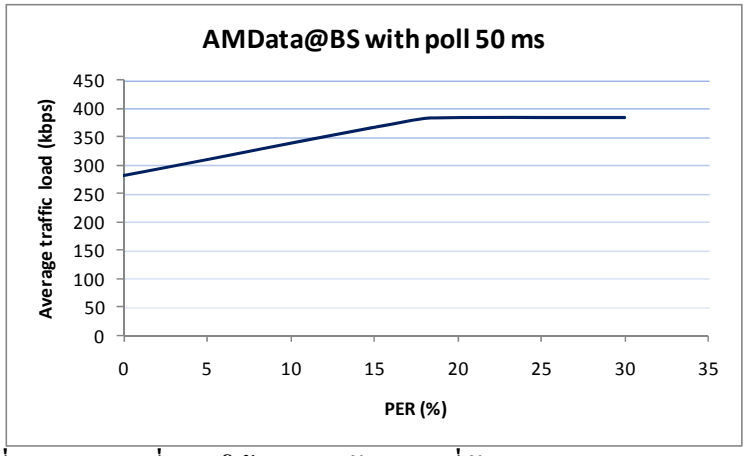
รูปที่ 5.18 ขนาดการใช้งานบัฟเฟอร์ RLC ที่ poll timer ค่าต่าง ๆ (7)



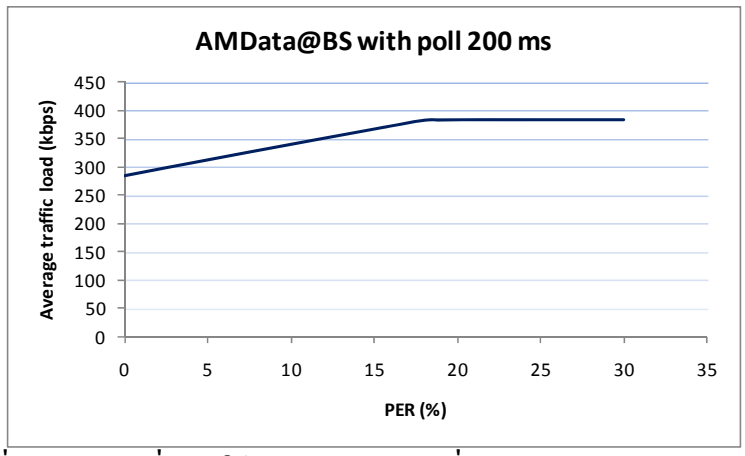
รูปที่ 5.19 ขนาดการใช้งานบัฟเฟอร์ RLC ที่ poll timer ค่าต่าง ๆ (8)

5.2.4 การใช้งานช่องสัญญาณ

การที่ error rate มีค่าสูงมาก ๆ ปริมาณข้อมูลที่ส่งปกติรวมกับข้อมูลที่ต้องส่งซ้ำอาจสูงเกินกว่าแบนด์วิดท์ของช่องสัญญาณที่ใช้งานอยู่ ซึ่งเป็นอีกเหตุผลหนึ่งที่ทำให้แพ็กเกตที่ส่งเกิดสูงหายทำให้ต้องมีการส่งซ้ำมากขึ้นกว่าเดิม ซึ่งเมื่อทดลองเก็บข้อมูลการใช้งานของช่องสัญญาณ DCH ที่ error rate ค่าต่าง ๆ พบว่าเป็นไปจริงตามข้อสันนิษฐาน กล่าวคือ ทั้ง poll ที่มีความถี่ 50 และ 200 มิลลิวินาที มีการใช้งานช่องสัญญาณเพิ่มสูงขึ้นตาม error rate ที่เพิ่มขึ้น และช่องสัญญาณมีการใช้งานเต็มแบนด์วิดท์เมื่อ error rate สูงเข้าใกล้ค่า threshold ที่ 19%ดังรูปที่ 5.20 และรูปที่ 5.21



รูปที่ 5.20 ค่าเฉลี่ยการใช้งานช่องสัญญาณที่อัตราความผิดพลาดค่าต่าง ๆ เมื่อ poll timer เป็น 50 ms



รูปที่ 5.21 ค่าเฉลี่ยการใช้งานช่องสัญญาณที่อัตราความผิดพลาดค่าต่าง ๆ เมื่อ poll timer เป็น 200 ms

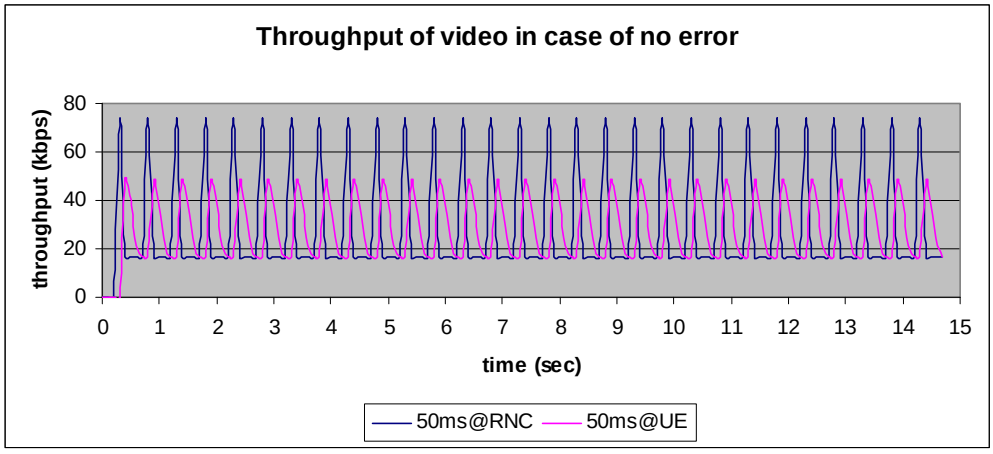
การใช้งานช่องสัญญาณสามารถวิเคราะห์จากการคำนวณได้ดังต่อไปนี้

เนื่องจากกำหนดให้ส่งวิดีโอเฟรมด้วยความเร็ว 10 เฟรมต่อวินาที และข้อมูลวิดีโอมี key frame interval เป็น 5 นั่นหมายถึง ใน 1 วินาทีจะมีการส่งวิดีโอชนิด I-frame จำนวน 2 เฟรม และชนิด P-frame จำนวน 8 เฟรม

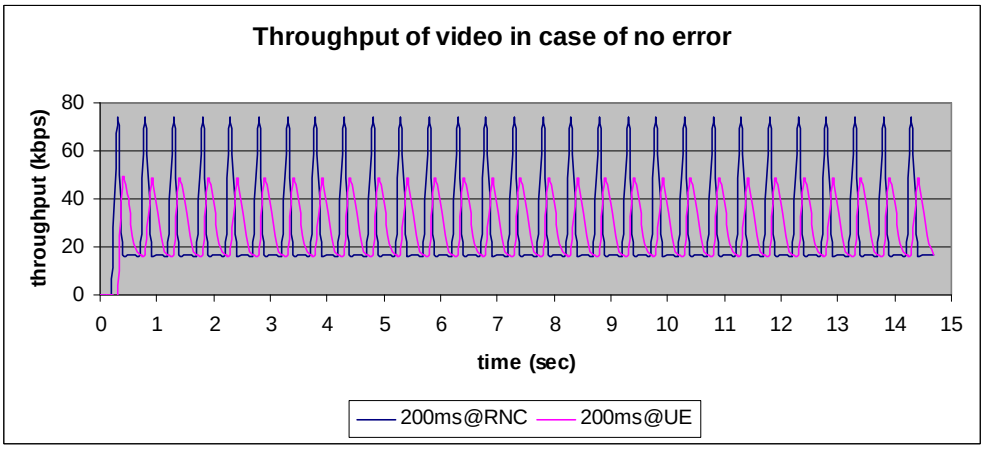
- I-frame มีความยาว 9000 ไบท์ ถูกส่ง 2 เฟรมต่อวินาที คิดเป็น 18000 ไบท์
- P-frame มีความยาว 2000 ไบท์ ถูกส่ง 8 เฟรมต่อวินาที คิดเป็น 16000 ไบท์
- นั้นคือ ใน 1 วินาทีจะข้อมูลวิดีโอเป็น 18000 + 16000 = 34000 ไบท์ หรือ 272 kbps

- 1 แพ็กเกตมีความยาว 1460 ไบท์ เพราะฉะนั้น 1 I-frame จะถูกแบ่งเป็น 7 แพ็กเกต และ 1 P-frame จะถูกแบ่งเป็น 2 ไบท์
- 1 แพ็กเกตจะมี header ขนาด 40 ไบท์ เพราะฉะนั้นในการส่ง 1 วินาทีจะมีข้อมูลที่เป็น header ทั้งหมดขนาด (7pkt * 40bytes/pkt * 2 frame) + (2pkt * 40bytes/pkt * 8 frame) = 1200 ไบท์ หรือ 9.6 kbps
- ใน 1 วินาทีมีข้อมูลส่งออกเท่ากับ 18000 + 16000 + 1200 = 35200 ไบท์
- 35200 bytes/sec * 8 bits/byte = 281.6 kbps

ซึ่งเมื่อเทียบผลการทดลองที่ทำการวัดค่า throughput ของข้อมูลวิดีโอกรณีไม่มีความผิดพลาดในช่องสัญญาณกับค่าจากการคำนวณข้างต้นจะให้ค่าเท่ากันดังแสดงในรูปที่ 5.22 และรูปที่ 5.23



รูปที่ 5.22 Throughput ของวิดีโอเมื่อ poll timer เท่ากับ 50ms วัดที่โหนด RNC และ UE



รูปที่ 5.23 Throughput ของวิดีโอเมื่อ poll timer เท่ากับ 200ms วัดที่โหนด RNC และ UE

จากกราฟรูปที่ 5.22 และรูปที่ 5.23 อาจเห็นผลที่ไม่ชัดเจนนัก เพื่อให้ได้ค่าเปรียบเทียบที่ชัดเจนขึ้นสามารถแสดงและคำนวนได้จากตารางที่ 5.6 ซึ่งเป็นข้อมูลดิบที่นำมาเขียนกราฟ

ตารางที่ 5.6 Throughput วิดีโอที่โหนด RNC กรณีไม่มี error rate

Poll 50ms @RNC		Poll 200ms @RNC	
เวลา (sec)	kbps	เวลา (sec)	kbps
0	0	0	0
0.1	0	0.1	0
0.2	0	0.2	0
0.3	74.24	0.3	74.24
0.4	16.64	0.4	16.64
0.5	16.64	0.5	16.64
0.6	16.64	0.6	16.64
0.7	16.64	0.7	16.64
0.8	74.24	0.8	74.24
0.9	16.64	0.9	16.64
1	16.64	1	16.64
1.1	16.64	1.1	16.64
1.2	16.64	1.2	16.64
1.3	74.24	1.3	74.24
1.4	16.64	1.4	16.64
1.5	16.64	1.5	16.64
...	...	...	...

ทั้ง 50ms และ 200ms poll timer ให้ผลเท่ากัน จากผลการทดลอง จะได้ว่าใน 1 วินาทีมีการใช้งานแบนด์วิทด์ไป 74.24 + 16.64 + 16.64 + 16.64 + 16.64 + 74.24 + 16.64 + 16.64 + 16.64 + 16.64 = 281.6 kbps ตรงกับผลจากการคำนวณทางทฤษฎี

เช่นเดียวกับที่โหนด UE ที่ยังคงสามารถรับข้อมูลที่ความเร็วเท่ากับที่ข้อมูลถูกส่งมาจากแหล่งกำเนิดดังแสดงและคำนวนได้จากตารางที่ 5.7

ตารางที่ 5.7 Throughput วิดีโอที่โหนด UE กรณีไม่มี error rate

50ms @UE		200ms @UE	
เวลา (sec)	kbps	เวลา (sec)	kbps
0	0	0	0
0.1	0	0.1	0
0.2	0	0.2	0
0.3	0	0.3	0
0.4	48	0.4	48
0.5	38.24	0.5	38.24
0.6	21.28	0.6	21.28
0.7	16.64	0.7	16.64
0.8	16.64	0.8	16.64
0.9	48	0.9	48
1	38.24	1	38.24
1.1	21.28	1.1	21.28
1.2	16.64	1.2	16.64
1.3	16.64	1.3	16.64
1.4	48	1.4	48
1.5	38.24	1.5	38.24
1.6	21.28	1.6	21.28
1.7	16.64	1.7	16.64
1.8	16.64	1.8	16.64
...	...	...	...

ทั้ง 50ms และ 200ms poll timer ให้ผลเท่ากัน จากผลการทดลอง จะได้ว่าใน 1 วินาทีมีการใช้งานแบนด์วิทด์ไป 48 + 38.24 + 21.28 + 16.64 + 16.64 + 48 + 38.24 + 21.28 + 16.64 + 16.64 = 281.6 kbps ตรง

หากพิจารณาในโปรโตคอลชั้นที่ 2 แพ็กเกตวิดีโอเฟรมที่ส่งจากชั้นบนลงมาจะถูกตัดแบ่งออกเป็น PDU ย่อยขนาด 40 ไบท์ ซึ่งเป็น RLC header 2 ไบท์ คิดเป็น 5% ของขนาด 40 ไบท์ (data 38 ไบท์ + header 2 ไบท์)

- ถ้าข้อมูลขนาด 281.6 kbps หลังจากโดนแบ่งเป็น PDU แล้ว ก็จะมี header ของชั้น RLC อยู่ประมาณ 281.6 kbps * 5% = 14 kbps
- ดังนั้นจะมีข้อมูลที่ถูกส่งออกไปในช่องสัญญาณ = 281.6 + 14 = 295.6 kbps

- เนื่องจากความจุของช่องสัญญาณมีค่าเป็น 384 kbps ถูกใช้งานไป 295.6 kbps จะเหลือช่องสัญญาณว่างให้ใช้งานได้อีก 384 – 295.6 = 88.4 kbps คิดเป็น 23.02% (88.4 * 100 / 384)
- นั่นคือมีช่องสัญญาณว่างสำหรับจัดการกับการส่งข้อมูลซ้ำได้ที่ error rate ไม่เกิน 23% (กรณีที่มีการส่งข้อมูลจากแหล่งเดียว)

ซึ่งผลจากการทดลองที่ได้ threshold เท่ากับ 19% ให้ผลสอดคล้องกับค่าที่คำนวณจากทฤษฎี ความแตกต่าง 4% อาจเกิดจากค่าทฤษฎีมีการชดเชย processing overhead ไม่ครบถ้วน

5.2.5 ประสิทธิภาพการทำงานของโปรโตคอล RLC

หากพิจารณาถึงประสิทธิภาพในการทำงานของโปรโตคอล RLC ที่มีกลไกสำหรับการส่งซ้ำแบบ Selective Repeat ARQ สามารถคำนวณทางทฤษฎีได้จากสมการที่ 5.1

$$\eta_{SR} = \frac{\frac{n_f - n_o}{t_f / (1 - P_f)}}{R} = \left(1 - \frac{n_0}{n_f}\right)(1 - P_f)$$ .....สมการที่ 5.1

เมื่อ n_f = ขนาดของเฟรม (บิต)

n_o = จำนวนของบิตที่เป็น header และ CRC (overhead)

R = อัตราการส่งข้อมูล (transmission rate)

t_f = เวลาที่ใช้ในการส่งเฟรม = $\frac{frame\_size}{tx\_rate} = \frac{n_f}{R}$

$(1 - P_f)$ = frame loss probability

= $(1-p)^{n_f} \approx e^{-n_f p}$ สำหรับ n_f ที่มีขนาดใหญ่และ p ที่มีขนาดเล็ก และ

p = bit error rate (BER)

ค่าที่ใช้ในการทดลอง เพื่อนำมาแทนค่าในสมการได้แก่

n_f = 1500 ไบท์ = 1500 * 8 = 12000 บิต

n_o = 40 ไบท์ = 40*8 = 320 บิต

R = 384 kbps

Packet error rate (PER) = 0 – 25%

แต่จากสมการต้องแทนค่าด้วย BER ซึ่งสามารถแปลงค่าได้ด้วยสมการที่ 5.2

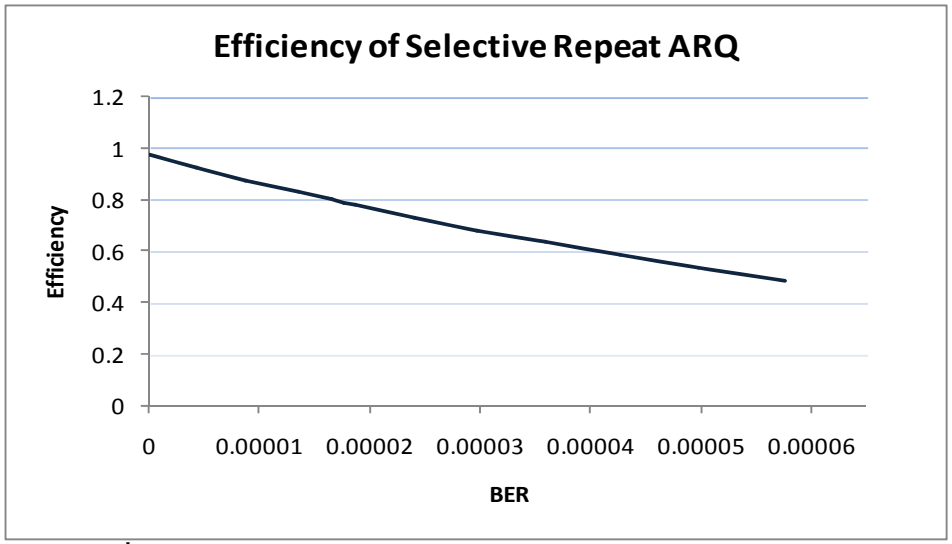
$\boxed{PER = 1-(1-BER)^{N}}$, N = packet length (บิต).....สมการที่ 5.2

หลังจากคำนวณตามสมการที่ 5.1 แล้วได้ผลตามตารางที่ 5.8 ซึ่งสามารถแสดงในรูปแบบของกราฟได้ดังรูปที่ 5.24

ตารางที่ 5.8 ประสิทธิภาพของ Selective Repeat ARQ

PER(%)	0	5	10	15	18	19	20
BER	0	0.00000421	0.00000877	0.0000135	0.0000165	0.0000176	0.0000186
Efficiency	0.9733	0.9244	0.8758	0.8275	0.7979	0.7876	0.7784

PER(%)	25	30	35	40	45	50
BER	0.000024	0.0000297	0.0000359	0.0000426	0.0000498	0.0000576
Efficiency	0.7298	0.6811	0.6325	0.5838	0.5352	0.4865

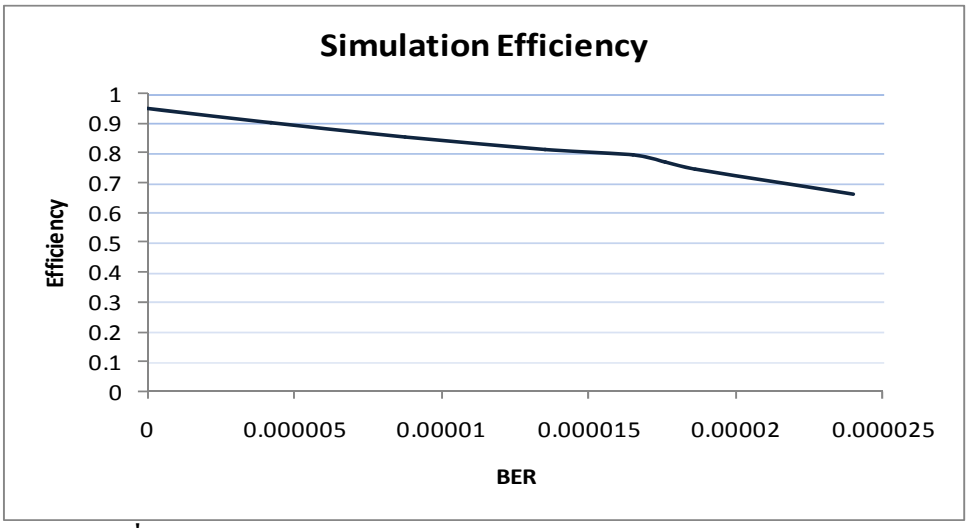


รูปที่ 5.24 ประสิทธิภาพของ Selective Repeat ARQ จากการคำนวณ

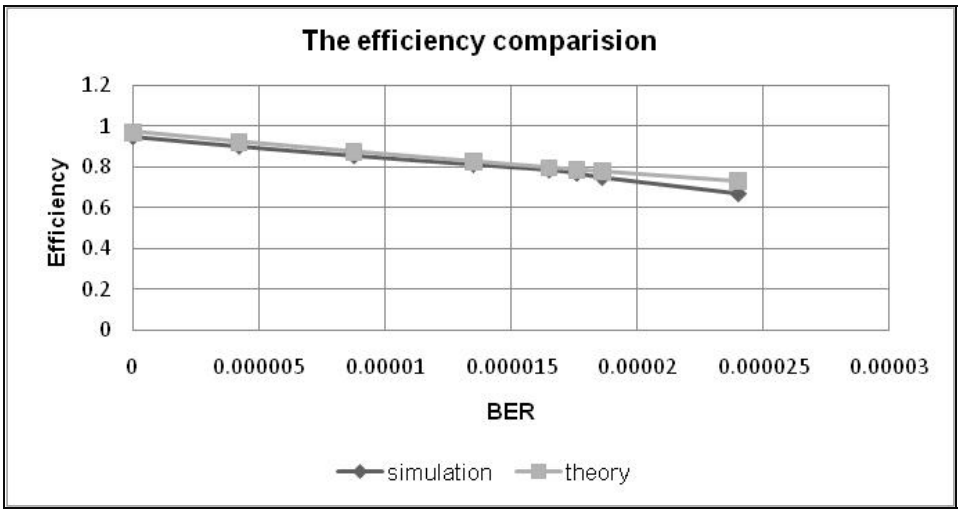
สำหรับประสิทธิภาพจากการทดลองแสดงได้ดังตารางที่ 5.9 และรูปที่ 5.25 ซึ่งจะเห็นว่าให้ผลไปในทิศทางเดียวกันกับผลที่ได้จากการคำนวณทางทฤษฎีดังรูปที่ 5.26

ตารางที่ 5.9 ประสิทธิภาพของโปรโตคอล RLC ที่ได้จากผลการทดลอง

		no error	err 5%	err 10%	err 15%	err 18%	err 19%	err 20%	err 25%
50 ms	header+data(kbyte/s)	34.27	32.56	30.89	29.33	28.54	27.75	26.97	23.93
	header (byte/s)	1754.624	1667.072	1581.568	1501.696	1461.248	1420.8	1380.864	1225.216
	data (kbyte/s)	32.5565	30.932	29.3455	27.8635	27.113	26.3625	25.6215	22.7335
	Eff	0.95	0.902597	0.856303	0.813058	0.791158	0.769259	0.747636	0.663364



รูปที่ 5.25 ประสิทธิภาพของ Selective Repeat ARQ จากการทดลอง



รูปที่ 5.26 การเปรียบเทียบประสิทธิภาพระหว่างค่าจากการทดลอง
และค่าจากการคำนวณทางทฤษฎี

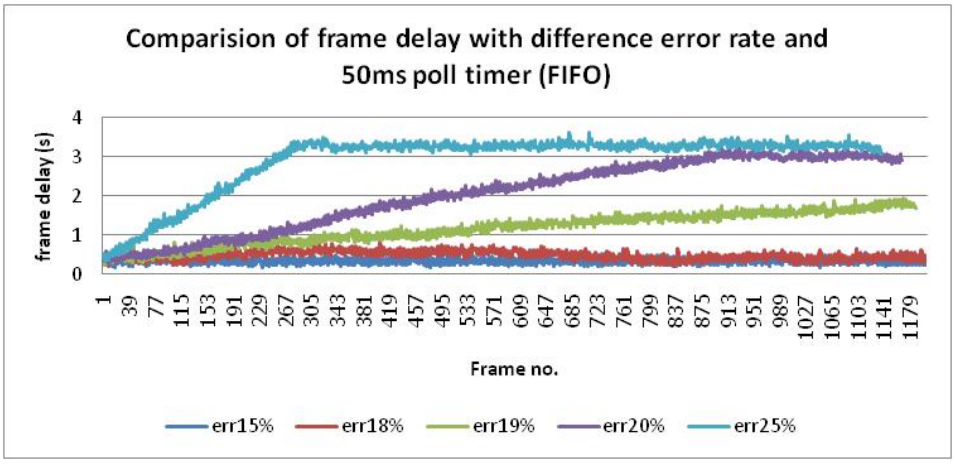
นอกจากนี้ได้ทำการทดลองเปลี่ยนระบบจำลองที่ใช้ในการทดสอบจากเดิมที่ RLC จัดการบัฟเฟอร์สำหรับส่งข้อมูลแบบ priority queue ตามที่เอกสารกำหนดมาตรฐานของ 3G กำหนดไว้ ทดลองปรับเปลี่ยนให้จัดการคิวแบบ First in first out (FIFO) แทน เพื่อทำการเปรียบเทียบว่าลักษณะคิวแบบใดเหมาะสมกว่า สำหรับผลการทดลองเปรียบเทียบเป็นไปตามหัวข้อ 5.2.6

5.2.6 การปรับเปลี่ยนกลไกการเลือกส่งข้อมูลภายใน RLC

กลไกการเลือกส่งข้อมูลของ RLC ตามที่ถูกระบุไว้ในเอกสารกำหนดมาตรฐานของกลุ่ม 3GPP นั้น กำหนดไว้ว่า หากแพ็กเกตหรือ PDU ใดที่เป็นการส่งซ้ำอันเนื่องมาจากความผิดพลาดไม่ว่ากรณีใด ๆ ให้มีลำดับความสำคัญสูงที่สุด กล่าวคือ จะต้องถูกส่งออกไปก่อน PDU ที่ถูกสร้างขึ้นใหม่และจะส่งเป็นครั้งแรกเสมอ ดังนั้น แม้ว่าในบัฟเฟอร์ส่งจะมี PDU ที่รอการส่งเป็นครั้งแรกรออยู่ในคิว หากตรวจพบว่ามี PDU ที่ถูกส่งไปไม่สำเร็จเกิดขึ้นและต้องการทำการส่งซ้ำ PDU นั้นก็จะถูกนำส่งออกก่อนทันทีแล้วให้ PDU อื่นในคิวรอ ซึ่งผลการทดลองสำหรับการใช้กลไกวิธีนี้ได้กล่าวถึงไปแล้วในหัวข้อก่อนหน้า จึงเกิดข้อสงสัยว่า หากทำการเปลี่ยนแปลงกลไกการเลือกส่งจะมีผลต่อการทำงานของโปรโตคอล RLC มากน้อยเพียงใด โดยกลไกใหม่ที่ออกแบบ อ้างอิงตามการทำงานของคิวในลักษณะ FIFO กล่าวคือ หากตรวจพบว่ามี PDU ที่ถูกส่งไปไม่สำเร็จเกิดขึ้นและต้องการทำการส่งซ้ำ PDU นั้นจะถูกนำไปต่อท้ายคิวเพื่อทำการรอส่งแทนการส่งทันที โดยผลการเปรียบเทียบการทำงานของกลไกทั้งสอง สามารถแสดงได้ดังต่อไปนี้

5.2.6.1 เวลาหน่วงในการรับเฟรม

เมื่อทำการทดลองวัดเวลาหน่วงของวิดีโอเฟรมแต่ละเฟรมที่ได้รับหลังจากทำการเปลี่ยนแปลงกลไกการเลือกส่งข้อมูลของโปรโตคอล RLC แล้ว ได้ผลดังรูปที่ 5.27 ซึ่งเป็นไปในลักษณะเดียวกันกับการจัดการคิวแบบ priority แต่ให้ค่าเวลาหน่วงต่ำกว่าเล็กน้อยอย่างไม่มีนัยสำคัญ



รูปที่ 5.27 เปรียบเทียบค่า frame delay เมื่อให้ link error rate ต่างกัน ที่ 50ms poll timer

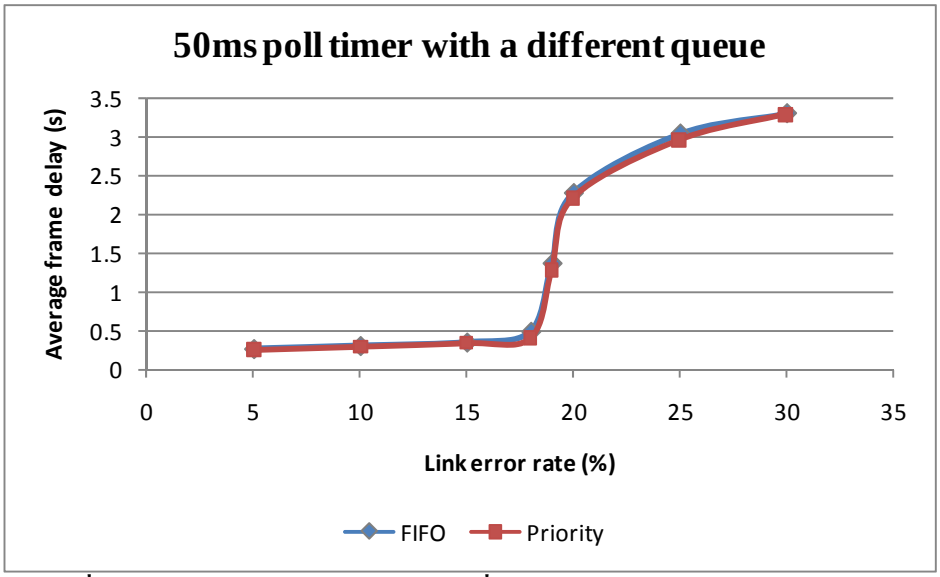
และเมื่อทำการคำนวณหาค่าเฉลี่ยของเวลาหน่วงเฟรมที่ link error rate ต่าง ๆ จะได้ผลดังตารางที่ 5.11 ซึ่งเมื่อนำมาเปรียบเทียบกับค่าจากตารางที่ 5.10 ที่เป็นผลการทำงานแบบ priority queue เดิม พบว่าแตกต่างกันน้อยมากทั้ง 50ms และ 200ms poll timer สามารถ plot กราฟแสดงการเปรียบเทียบได้ดังรูปที่ 5.28 และรูปที่ 5.29 สำหรับ 50ms poll time และ 200ms poll timer ตามลำดับ

ตารางที่ 5.10 เปรียบเทียบค่าเฉลี่ยเวลาหน่วงเฟรมเมื่อใช้กลไกแบบ priority queue

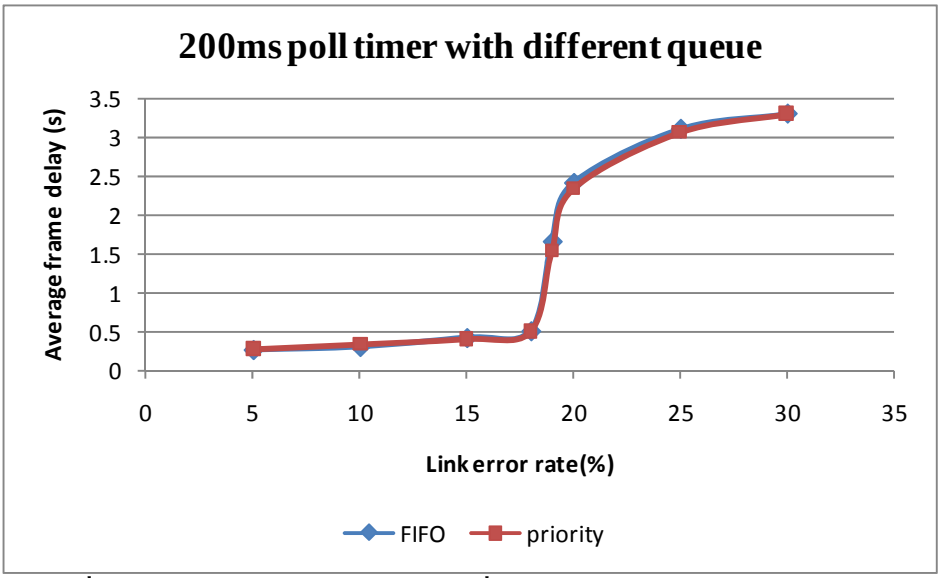
Poll timer (ms)	Average frame delay (s) priority queue							
	Err5%	10%	15%	18%	19%	20%	25%	30%
50	0.257	0.299	0.345	0.408	1.282	2.214	2.959	3.292
200	0.280	0.343	0.412	0.504	1.540	2.340	3.065	3.302
500	0.291	0.460	0.485	0.613	1.616	2.451	3.102	3.331

ตารางที่ 5.11 เปรียบเทียบค่าเฉลี่ยเวลาหน่วงเฟรมเมื่อใช้กลไกแบบ FIFO queue

poll timer (ms)	Average frame delay (s) FIFO queue							
	Err5%	10%	15%	18%	19%	20%	25%	30%
50	0.267	0.305	0.350	0.489	1.366	2.268	3.031	3.292
200	0.267	0.305	0.350	0.497	1.655	2.407	3.101	3.296



รูปที่ 5.28 ความสัมพันธ์ระหว่างค่าเฉลี่ยเวลาหน่วงเฟรมกับ link error rate เมื่อ poll time = 50ms



รูปที่ 5.29 ความสัมพันธ์ระหว่างค่าเฉลี่ยเวลาหน่วงเฟรมกับ link error rate เมื่อ poll time = 200ms

จากผลของเวลาหน่วงเฟรมแสดงให้เห็นว่าทั้งสองกลไกสามารถทำงานได้ไม่แตกต่างกัน โดยการเลือกส่งแบบ priority ให้ผลดีกว่าเพียงเล็กน้อย เนื่องจากการจัดการคิวแบบ FIFO เป็นการจัดการคิวที่แย่ที่สุด การส่งแบบ priority จะทำให้การส่งซ้ำของแพ็กเกตที่ส่งผิดพลาดหรือการกู้คืนข้อมูลเกิดขึ้นได้เร็วกว่าแบบ FIFO แต่ทั้งนี้สาเหตุที่ทั้งสองกลไกให้ผลที่ใกล้เคียงกันเป็นเพราะ โดยปกติการกู้คืนแพ็กเกตแบบนำไปวางที่ส่วนหน้าของคิว และส่วนท้าย

ของคิว จะมีข้อแตกต่างในเรื่องของเวลาที่อยู่ในคิว ซึ่งกลไกที่นำแพ็กเกตที่ต้องการการส่งซ้ำไปวางที่ส่วนหน้า แพ็กเกตนั้นจะถูกส่งออกไปได้ทันทีโดยไม่จำเป็นต้องรอ (waiting time) แต่สำหรับกลไกที่นำแพ็กเกตที่ต้องการการส่งซ้ำไปวางที่ส่วนท้ายของคิว แพ็กเกตนั้นจำเป็นต้องรอจนกว่าแพ็กเกตที่อยู่ในคิวก่อนหน้าจะถูกส่งออกไปก่อน ซึ่งเวลาที่แพ็กเกตนั้นใช้ในการรออยู่ในคิวจะขึ้นอยู่กับขนาดของคิว ณ เวลานั้น ๆ จาการทดลอบวัดค่าเฉลี่ยของเวลาที่แพ็กเกตใช้รออยู่ในคิวของทั้งสองกลไกพบว่า เมื่ออัตราความผิดพลาดของช่องสัญญาณมีค่าต่ำ เวลาที่แพ็กเกตใช้รออยู่ในคิวก็มีค่าต่ำ และเวลาที่แพ็กเกตใช้รออยู่ในคิวจะมีค่าสูงขึ้น เมื่ออัตราความผิดพลาดของช่องสัญญาณเพิ่มมากขึ้น ทั้งสำหรับ poll time ที่ 50ms และ 200ms โดยกลไกการส่งแบบ priority มีค่าเวลาที่แพ็กเกตใช้รออยู่ในคิวต่ำกว่าแบบ FIFO เล็กน้อย ดังตารางที่ 5.12 และตารางที่ 5.13 ตามลำดับ ทั้งนี้ เวลาที่แพ็กเกตใช้ในการรออยู่ในคิวจะถือว่าน้อยมากเมื่อเปรียบเทียบกับเวลาทั้งหมดที่ใช้ในการรับส่งเฟรม จึงถือว่าเวลาที่แพ็กเกตรอในคิวนี้ไม่มีนัยสำคัญ ดังนั้น ทั้งสองกลไกจึงให้ค่าเวลารวมของการรับส่งเฟรมไม่แตกต่างกัน เพราะค่าเวลาที่แพ็กเกตรอในคิวไม่ได้ส่งผลอย่างมีนัยสำคัญต่อเวลารวมของการรับส่งเฟรม

ตารางที่ 5.12 ค่าเฉลี่ยของเวลาที่แพ็กเกตใช้ในการรออยู่ในคิว เมื่อ poll timer เท่ากับ 50ms

Algorithms	Average waiting time in queue (s) Poll 50ms							
	Err 5%	10%	15%	18%	19%	20%	25%	30%
priority	0.00886	0.01072	0.01232	0.01528	0.04142	0.0778	0.1031	0.1175
FIFO	0.01061	0.01274	0.01431	0.02261	0.05577	0.0913	0.1221	0.1332

ตารางที่ 5.13 ค่าเฉลี่ยของเวลาที่แพ็กเกตใช้ในการรออยู่ในคิว เมื่อ poll timer เท่ากับ 200ms

Algorithms	Average waiting time in queue (s) Poll 200ms							
	Err 5%	10%	15%	18%	19%	20%	25%	30%
priority	0.01064	0.01203	0.01471	0.01863	0.05923	0.09361	0.1135	0.1222
FIFO	0.01112	0.01285	0.01545	0.02963	0.06899	0.09826	0.1322	0.1349

5.3 สรุป

การจำลองการส่งไฟล์วิดีโอ MPEG-4 บทเครือข่าย UMTS เพื่อทำการวัดและวิเคราะห์ประสิทธิภาพการทำงานของโปรโตคอล RLC และนอกจากนี้ก็เพื่อดูผลกระทบของการกำหนดค่า poll_timer ซึ่งเป็นพารามิเตอร์หนึ่งของกลไกการ poll ในชั้น RLC ฝั่งส่งที่มีต่อการส่งข้อมูลเมื่อ error rate มีค่าต่าง ๆ โดยการสมมติเหตุการณ์ที่ UE ทำการร้องขอเปิดดูไฟล์วิดีโอที่มีความยาวมากกว่า 10 นาทีจากโหนดที่อยู่บนอินเทอร์เน็ต เพื่อเป็นการทดสอบผลกระทบของการทำงานของระบบเมื่อไฟล์วิดีโอที่ต้องการมีเวลายาวนาน ผลที่พบคือ poll timer ค่าต่ำกว่าจะให้ค่า

เวลาหน่วงในการรับเฟรมดีกว่าแม้ว่าอัตราความผิดพลาดในช่องสัญญาณจะเพิ่มขึ้น แต่เมื่ออัตราความผิดพลาดในช่องสัญญาณสูงจนเกินค่า threshold ค่า poll timer ก็จะไม่มีผลต่อเวลาหน่วงในการรับเฟรมอีกต่อไป ซึ่งค่า threshold นี้เกิดจากข้อจำกัดของขนาดช่องสัญญาณ เพราะเมื่ออัตราความผิดพลาดในช่องสัญญาณสูง จะเป็นผลให้มีแพ็กเกตจำนวนมากที่ต้องทำการส่งซ้ำ ด้วยเหตุนี้ทำให้ข้อมูลที่ต้องส่งและต้องส่งซ้ำมีปริมาณมากจนเกินความสามารถที่ช่องสัญญาณจะส่งได้ ทำให้แพ็กเกตค้างอยู่ในบัฟเฟอร์เพื่อรอการส่งให้สำเร็จเป็นจำนวนมาก ส่งผลต่อค่าเวลาหน่วงในการรับเฟรมของฝั่งรับที่จะเพิ่มสูงขึ้น นอกจากนั้นเมื่อทำการเปลี่ยนกลไกการเลือกส่งข้อมูลภายในโปรโตคอล RLC ให้แตกต่างจากที่มาตรฐานระบุ โดยกลไกใหม่ที่ออกแบบ อ้างอิงตามการทำงานของคิวในลักษณะ FIFO กล่าวคือ หากตรวจพบว่ามี PDU ที่ถูกส่งไปไม่สำเร็จเกิดขึ้นและต้องการทำการส่งซ้ำ PDU นั้นจะถูกนำไปต่อท้ายคิวเพื่อทำการรอส่งแทนการส่งทันที ผลที่ได้จากการทดลองไม่แสดงให้เห็นถึงความแตกต่างอย่างมีนัยสำคัญระหว่างทั้งสองกลไก

สำหรับผลการทดลองที่ทดสอบโดยใช้ไฟล์วิดีโอจริงส่งผ่านเครือข่ายจำลอง สามารถดูรายละเอียดได้ในภาคผนวก ข.