บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของวิทยานิพนธ์

ในปัจจุบันโลกของการสื่อสารโทรคมนาคมแบบไร้สาย ได้มีการแข่งขันและพัฒนาในการให้บริการขึ้นอย่างรวดเร็ว เพื่อตอบสนองความต้องการของผู้ใช้ที่อยู่ไม่จำกัด ตั้งแต่การเปลี่ยนแปลงเทคโนโลยีจากระบบอนาล็อกมาเป็นระบบดิจิตอล การเพิ่มความเร็วในการโอนถ่ายข้อมูล การเพิ่มจำนวนช่องสัญญาณเพื่อตอบสนองความต้องการของผู้ใช้ที่มีเพิ่มมากขึ้นเรื่อย ๆ เกิดการเปลี่ยนยุคของระบบการสื่อสารไร้สายจากระบบสื่อสารเคลื่อนที่ยุคที่ 1 เป็นยุคที่ 2 จากยุคที่ 2 ก้าวสู่ยุคที่ 3 ในปัจจุบัน (3G: Third Generation Mobile Phone System) โดยระบบสื่อสารโทรศัพท์เคลื่อนที่ในยุคที่ 3 นี้จะเน้นการพัฒนาในเรื่องของการโอนถ่ายข้อมูลที่มีความเร็วสูงขึ้น การเพิ่มจำนวนช่องสัญญาณ ความปลอดภัยของข้อมูล โดยระบบโทรศัพท์เคลื่อนที่ยุคที่ 3 นี้ได้มีการแบ่งโครงสร้างการทำงานออกเป็นหลายส่วนอันได้แก่ชั้นที่ 1 ซึ่งเรียกว่า ชั้นกายภาพ (Physical Layer) ชั้นที่ 2 หรือชั้นเชื่อมโยงข้อมูล (Data Link Layer) และชั้นที่ 3 หรือชั้นควบคุมทรัพยากรคลื่นวิทยุ (Radio Resource Control : RRC) แต่ละชั้นก็จะมีหน้าที่การทำงานที่แตกต่างกัน สำหรับวิทยานิพนธ์นี้จะให้ความสนใจมุ่งเน้นเฉพาะชั้นเชื่อมโยงข้อมูล อันประกอบด้วยโปรโตคอลเอนทิตี (Protocol Entity) ย่อย ๆ คือ โปรโตคอลควบคุมการเชื่อมโยงคลื่นวิทยุ (Radio Link Control : RLC) โปรโตคอลการลู่เข้าของกลุ่มข้อมูล (Packet Data Convergence Protocol : PDCP) โปรโตคอลควบคุมบรอดคาสและมัลติคาส (Broadcast and Multicast Control : BMC) และ โปรโตคอลควบคุมการเข้าถึงสื่อกลาง (Medium Access Control : MAC) ทำการศึกษาและออกแบบวิธีการเพื่อใช้ในการประเมินสมรรถนะของโปรโตคอลในชั้นนี้ โดยนำเอา network simulator tool เข้ามาช่วยในการสร้าง simulator model ที่ใช้ในการทดสอบวิธีการที่ได้ออกแบบและประเมินสมรรถนะของโปรโตคอล เนื่องจากปัจจุบันระบบโทรศัพท์เคลื่อนที่ยุคที่ 3 ได้มีหลายกลุ่ม หลายองค์กรได้ทำการพัฒนาขึ้นเพื่อนำมาใช้งาน ซึ่งแต่ละกลุ่มสนใจเฉพาะการพัฒนาเพื่อให้ได้มาใช้งาน แต่ไม่ได้มีการประเมินคุณภาพหรือสมรรถนะการทำงานของมัน จึงไม่อาจสรุปได้ว่าโปรโตคอลที่ถูกพัฒนานั้นมีข้อดี ข้อด้อยต่างกันอย่างไร การประเมินสมรรถนะของโปรโตคอลจะส่งผลให้สามารถรู้ได้ว่าโปรโตคอลชั้นที่ 2 ที่ถูกพัฒนาขึ้นตามเอกสารของ 3GPP (The 3rd Generation Partnership Project)[1] นั้นมีประสิทธิภาพเพียงไร ค่าพารามิเตอร์ใดบ้างที่มีผลต่อสมรรถนะของระบบ หากต้องการให้ระบบทำงานได้ดีขึ้นต้องปรับค่าใด เพื่อเป็นแนวทางในการพัฒนาโปรโตคอลให้มีประสิทธิภาพมากขึ้นต่อไป โดยใช้กรณีศึกษาการสื่อสารวีดิทัศน์ที่มีการบีบอัดแบบ MPEG-4[2] เป็นสื่อทดสอบ

1.2 การตรวจเอกสาร (Literatures Review)

ในการตรวจเอกสารของวิทยานิพนธ์นี้ ได้ศึกษาเทคนิคที่เกี่ยวกับการทำงานภายในโปรโตคอล RLC ที่มีผลต่อประสิทธิภาพการทำงาน ทั้งที่เป็นกลไกและพารามิเตอร์ของโปรโตคอลเองและกลไกเพิ่มเติมที่มีผู้ได้นำเสนอไว้ โดยสามารถแบ่งออกได้เป็นหัวข้อดังต่อไปนี้

1.2.1 การวิเคราะห์กลไกและพารามิเตอร์ของโปรโตคอลที่มีผลต่อประสิทธิภาพ

ความเจริญก้าวหน้าอย่างรวดเร็วในการติดต่อสื่อสารแบบไร้สาย UMTS (Universal Mobile Telecommunications Systems) มีบทบาทหลักในการพัฒนาเครือข่ายไร้สายยุคที่ 3 (3G) ซึ่งมีจุดประสงค์หลักเพื่อให้การบริการข้อมูลและข้อมูลเสียงมีความเร็วสูง UMTS พัฒนาโปรโตคอล Radio Link Control (RLC) เพื่อสนับสนุนความน่าเชื่อถือของโปรโตคอลในชั้นที่สูงขึ้น เช่น TCP (Transmission Control Protocol) โดย RLC จะมีการทำงานการส่งซ้ำของข้อมูลเพื่อกู้ข้อมูลที่ผิดพลาดที่เกิดขึ้นที่ระดับชั้นลิงค์ (Link Layer) เป็นการปิดบังความผิดพลาดจากระดับชั้นบน เนื่องจากความซับซ้อนของโปรโตคอลและความหลากหลายในการปรับแต่งค่าของพารามิเตอร์ ดังนั้นการปรับแต่งค่าของโปรโตคอลที่ต่างกันจึงมีผลต่อประสิทธิภาพการทำงานของโปรโตคอลเองและมีผลต่อความน่าเชื่อถือในการส่งข้อมูลของโปรโตคอลระดับชั้นบนด้วย จึงมีผู้เสนอการสร้างแบบจำลองของโปรโตคอล RLC พร้อมกับการวิเคราะห์ผลของการปรับแต่งค่าพารามิเตอร์ต่าง ๆ ภายในระดับชั้นที่กระทบต่อประสิทธิภาพของโปรโตคอลในเชิงของ throughput, goodput, delay และ discard rate ซึ่งสามารถสรุปโดยรวมได้ดังนี้

แบบจำลองของ Qinging Zhang และ Hsuan-Jung Su [3] ให้ RLC ทำงานในโหมด AM (Acknowledged Mode) และ traffic model จากโปรโตคอลชั้นบนเป็นผลจากโปรแกรมประยุกต์ web browsing ได้ทำการเปรียบเทียบผลกระทบของกลไกการ Poll (Polling Mechanisms) โดยสนใจ *POLL_SDU* trigger, *last PDU in transmission buffer* และ *last PDU in retransmission buffer* ที่มีต่อ delay, discard rate และ throughput ของโปรโตคอล RLC และเปรียบเทียบประสิทธิภาพของโปรโตคอลเมื่อให้ค่าของพารามิเตอร์ *timer_discard* เปลี่ยนไปรวมถึงค่าของ service rate ด้วย โดยแบบจำลองที่สร้างขึ้นมุ่งเน้น RLC ฝั่งส่งซึ่งการทำงานหลักของฝั่งส่งที่ถูกสร้างไว้ได้แก่ segmentation/concatenation, padding, transmission และ retransmission buffer management และ acknowledgement handling สำหรับการทำงานในฝั่งรับถูกสร้างเพียงบางส่วนคือ status reports และชั้น MAC จะต้องได้รับจำนวน PDU สำหรับทุก ๆ TTI ที่คงที่ขึ้นกับ service rate ผลของการทดลองและวิเคราะห์จากแบบจำลองสามารถแยกผลกระทบที่มีต่อประสิทธิภาพของโปรโตคอลชั้น RLC ออกเป็น 3 หมวดคือ

1.2.1.1 ผลกระทบของ Polling Mechanism

จากที่ได้กล่าวไปแล้วข้างต้นว่าพารามิเตอร์ที่ใช้ในการทดลองและวิเคราะห์ได้แก่ *POLL_SDU*, *last PDU in transmission buffer* และ *last PDU in retransmission buffer* ผลที่ได้

- เปรียบเทียบ throughput และ delay เมื่อทำการเพิ่มค่า *POLL_SDU* จาก 1 – 10 พร้อมกับเปรียบเทียบเมื่อมีและไม่ *last PDU* trigger ทั้งสองตัว ผลคือ ค่า *POLL_SDU* ที่ต่ำจะให้ throughput สูงและ delay ต่ำ ในทางกลับกัน *POLL_SDU* ที่สูงจะให้ throughput ต่ำและ delay สูง เป็นทั้งในกรณีที่มีและไม่ *last PDU* trigger แต่การมี *last PDU* triggers จะให้ throughput สูงและ delay ต่ำ ซึ่งจะดีกว่าการที่ไม่มี *last PDU* triggers อย่างเห็นได้ชัด เนื่องมาจาก *POLL_SDU* ค่ามากที่ทำให้ช่วงของการ poll จะกว้าง แต่ *last PDU* triggers จะช่วยชดเชยทำให้มีการส่ง polling message ที่เพียงพอ ค่าของ throughput และ delay จึงดีขึ้น (ความถี่ของการ poll ช่วยเพิ่มประสิทธิภาพได้)
- ค่าของ status report overhead จากข้อข้างต้น ความถี่ของการ poll สูง (*POLL_SDU* ต่ำ และการมี *last PDU* triggers) จะช่วยทำให้ throughput และ delay ดีขึ้น แต่การ poll บ่อยครั้งจะเป็นตัวทำให้ status report overhead สูงตามไปด้วยเนื่องจากมีการสร้างและส่ง status report มากตามจำนวนครั้งการ poll นั่นคือ ค่า *POLL_SDU* ที่ต่ำจะก่อให้เกิด overhead ที่สูงกว่า *POLL_SDU* ที่สูง และการที่มี *last PDU* triggers จะยิ่งทำให้มี overhead ที่สูงขึ้น
- เปรียบเทียบ SDU discard rate โดยการ discard จะเกิดจาก *timer_discard* expire เพื่อลดการติดตายของโปรโตคอลและการหน่วงเวลา ค่าของ *POLL_SDU* ที่ต่ำจะทำให้ SDU discard rate ต่ำกว่าการที่ *POLL_SDU* มีค่าสูง เนื่องจากการ poll ที่เกิดขึ้นเร็ว ฝั่งส่งจะสามารถรับรู้สถานะการรับแพ็กเกตของฝั่งรับแล้วทำการส่งซ้ำให้รับข้อมูลได้ถูกต้องก่อนที่ timer จะ expire แล้วทำให้เกิดการ discard ขึ้น

1.2.1.2 ผลกระทบของ SDU discard timer

ทำการทดลองเปลี่ยนแปลงค่า *timer_discard* เป็น 2 วินาที และ 4 วินาที ผลที่ได้คือ ค่า timer ที่เปลี่ยนไปไม่ส่งผลกระทบต่อ throughput และ status report overhead ค่ายังคงเท่ากัน สำหรับเวลาหน่วงนั้น timer ที่ 4 วินาที จะให้เวลาหน่วงที่สูงกว่า 2 วินาที เนื่องจาก timer ที่สูงจะมีแพ็กเกตจำนวนมากคอยในบัฟเฟอร์ เวลาหน่วงที่ใช้จึงเพิ่มขึ้น แต่ค่า timer ที่สูงนี้สามารถลดจำนวน SDU discard rate ได้

1.2.1.3 ผลกระทบของ service rate

ทดลองเปลี่ยน service rate เป็น 64 และ 384 kbps พร้อมกับเพิ่มค่า POLL_SDU จาก 1 – 10 ซึ่งผลที่ได้ service rate ที่ 384 kbps จะให้ throughput และ delay ที่ดีกว่า 64 kbps และสำหรับ SDU discard rate จะมีค่าต่ำมากๆ ที่ 384 kbps เนื่องจากมี link capacity ที่ใหญ่ขึ้น ทำให้เวลาที่ใช้ใน queue ของแต่ละ SDU น้อยตามไปด้วย

อีกแบบจำลองหนึ่งที่ให้ traffic model ของข้อมูลที่มาจากโปรโตคอลชั้นบนเป็น web browsing และให้ RLC ทำงานในโหมด AM เช่นกัน โดยสนใจผลกระทบของ *poll prohibit timer* และ *poll timer* [4] ซึ่ง *poll prohibit timer* ถูกใช้เพื่อจัดการกับปัญหาการส่ง poll และ status report ที่มากเกินไป ที่ฝั่งส่ง *poll prohibit timer* จะเริ่มนับเมื่อ PDU ที่ถูกกำหนดบิต poll ใน header ถูกส่งออกไป จะไม่มีการ poll ใดๆที่ได้รับอนุญาตจนกว่า timer นี้จะ expire เมื่อทำการศึกษาผลกระทบของ *poll timer* จะกำหนดค่าของ *poll prohibit timer* เป็น 0.1 วินาทีคงที่ และเมื่อทำการศึกษาผลกระทบของ *poll prohibit timer* จะทำการกำหนดค่าของ *poll timer* ให้คงที่ที่ 0.5 วินาที โดยผลจากการทดลองสามารถแบ่งได้เป็น

1.2.1.4 ผลกระทบของ poll prohibit timer

ถ้าค่าของ *poll prohibit timer* มีค่าสูง (การ poll ถูกห้ามเป็นเวลานาน) throughput และ goodput จะต่ำ ในกระบวนการของ ARQ (Automatic Repeat Request) จำเป็นอย่างมากสำหรับการรอรับ status report เพื่อที่จะทำการเลื่อน transmission window ได้ ค่า *poll prohibit timer* ที่ใหญ่จะทำให้ความถี่ในการส่งการ poll น้อยเป็นผลให้การตอบกลับ status report น้อยลงตามไปด้วยรวมถึงการเลื่อน transmission window ก็จะถูกหน่วงเช่นกัน นอกจากนี้การลดค่าของ throughput และ goodput มีความรุนแรงมากขึ้นที่ error rate ที่สูงขึ้นเพราะการร้องขอการ poll และ status report มีแนวโน้มที่จะสูญหายในอากาศได้มากขึ้น และสำหรับผลกระทบต่อ RLC SDU delay เมื่อ *poll prohibit timer* มีค่าสูงจะส่งผลให้ delay มีค่าสูงเพิ่มมากขึ้นด้วย จากการทดลองพบว่าการกำหนดค่า *poll prohibit timer* เป็น 100 ms ที่เหมาะสม ได้ SDU delay ต่ำในขณะที่ throughput ไม่สูงจนผิดปกติ

1.2.1.5 ผลกระทบของ poll timer

สำหรับการวิเคราะห์ค่า *poll timer* ทำการกำหนด *poll prohibit timer* เป็น 100 ms จากการทดลอง ค่าของ *poll timer* น้อย จะได้ throughput ที่สูง เมื่อ *poll timer* มีค่าน้อย (โดยเฉพาะอย่างยิ่งน้อยกว่า RLC Round Trip Time, RTT) *poll timer* expire ในไม่ช้าหลังจาก

PDU ที่ถูกกำหนดบิต poll ใน header ถูกส่ง ไม่มี status report ที่เกี่ยวข้องรับเข้ามาภายในช่วงเวลาดังกล่าวเนื่องจากช่วงเวลาสั้นกว่า RTT Polling PDU จะถูกส่งซ้ำก่อนเวลาที่สมควรและบางส่วนของการส่งซ้ำของ polling PDU ไม่จำเป็นที่จะต้องส่ง PDU ซ้ำที่ซึ่งอาจเพิ่ม throughput แต่ไม่ผลต่อการเพิ่ม goodput หรือ SDU delay การกำหนดค่าของ *poll timer* ที่น้อยกว่า 100 ms จะให้ throughput ที่เท่ากัน เนื่องจาก *poll timer* และ *poll prohibit timer* เริ่มต้นจับเวลาพร้อมกันเมื่อ PDU ที่ถูกกำหนดบิต poll ใน header ถูกส่งออกไป และ *poll prohibit timer* มีค่าเป็น 100 ms การกำหนดค่า *poll timer* ที่น้อยกว่า 100 ms มีค่าเทียบเท่ากับให้ค่ามันเท่ากับ 100 ms เพราะแม้ว่ามันจะ expire ก่อน 100 ms มันก็จะต้องคอยให้ *poll prohibit timer* ก่อนด้วยจึงจะทำให้การ trigger มีผลใช้งาน ความสัมพันธ์ระหว่างค่า *poll timer* และ RLC SDU delay จะแปรผันตรงกัน กล่าวคือ ค่าของ *poll timer* ที่เพิ่มขึ้น RLC SDU delay ก็จะเพิ่มขึ้นด้วย ค่าของ *poll timer* ที่ต่ำจะช่วย RLC กู้ข้อมูลที่สูญหายหรือผิดพลาดได้เร็วขึ้น ค่าของ *poll timer* ที่น้อยกว่า 100 ms จะให้ค่าของ SDU delay ที่ใกล้เคียงกันด้วยเหตุผลเดียวกับในการวิเคราะห์ throughput ด้วยเหตุนี้ค่าของ *poll timer* โดยทั่วไปไม่ควรกำหนดให้น้อยกว่าค่าของ *poll prohibit timer* จากผลข้างต้นสามารถสรุปโดยรวมได้ว่าค่าของ *poll prohibit timer* และ *poll timer* ที่ต่ำทำให้ได้ delay ที่ต่ำแต่จะให้ throughput สูง

แบบจำลองของโปรโตคอล RLC เพื่อใช้ในการศึกษาประสิทธิภาพนอกจากให้ traffic model เป็น web browsing แล้ว ยังมีผู้เสนอใช้ FTP traffic model ในการทดลองอีกด้วย [5] โดยศึกษาผลกระทบของ *poll timer* และ *status period timer* ที่วัดภายใต้การเปลี่ยนแปลงค่า BLER (Block Error Rate) หลาย ๆ ค่า มี service rate ที่ 384 kbps และให้ RLC ทำงานในโหมด AM เช่นเดียวกับ 2 แบบจำลองที่ได้กล่าวมาแล้ว และการทดลองทั้งหมดสมมติให้ขนาดของ transmission window และ receiving window ใหญ่เพียงพอในการรับ-ส่งข้อมูลแล้วจะไม่ก่อให้เกิดการล้นของบัฟเฟอร์ที่อาจทำให้ข้อมูลสูญหายได้ ผลของการทดลองและวิเคราะห์จากแบบจำลองสามารถแยกผลกระทบที่มีต่อประสิทธิภาพของโปรโตคอลชั้น RLC ออกเป็น 2 หมวด คือ

1.2.1.6 ผลกระทบของ status period timer

ทำการวิเคราะห์ผลกระทบของ *status period timer* โดยดูจากค่าของ RLC SDU delay, RLC throughput และ RLC goodput ได้ผลดังนี้

- RLC SDU delay เมื่อทำการเปลี่ยนค่า status period timer และ BLER
  - เมื่อ BLER = 0% นั่นคือ ไม่มี PDU สูญหาย แสดงว่า PDU ทั้งหมดจะสามารถถูกรับได้ถูกต้องโดยใช้การส่งเพียงครั้งเดียว ดังนั้นค่า *status period timer* ไม่มีผลใด ๆ

ต่อ SDU delay ค่าของ SDU delay กรณีที่ไม่มีข้อผิดพลาดในการส่งจะเท่ากับผลรวมของเวลา one-way latency และเวลา 1 TTI

- เมื่อ BLER > 0% ค่า *status period timer* ที่น้อยๆ เช่น 40 ms จะให้ค่า SDU delay ที่สูง เมื่อ *status period timer* เพิ่มค่าขึ้นถึงค่า threshold ของแต่ละค่า BLER (ในขณะนี้ยังไม่สามารถหาที่มาของค่านี้ได้) จะทำให้ได้ SDU delay ที่ต่ำที่สุด และเมื่อเพิ่มค่าเลยจากค่า threshold นี้ ค่า SDU delay จะเพิ่มสูงขึ้นอีกครั้ง เนื่องจาก *status period timer* ค่าต่ำจะทำให้ฝั่งรับส่ง status report กลับมาหาฝั่งส่งบ่อยครั้งเกินไป ทำให้ RLC PDU จำนวนมากถูกกันไว้ที่บัฟเฟอร์ฝั่งส่ง และในบางกรณี เช่น *status period timer* มีค่าต่ำกว่า RTT อาจทำให้เกิดการส่ง status report ซ้ำซ้อนหรือส่งโดยไม่จำเป็น
- ที่ BLER ค่าสูง ค่าของ *status period timer* ที่เหมาะสมสำหรับทำให้ SDU delay น้อยที่สุดจะอยู่ใกล้ค่า RTT

- RLC throughput
  - ค่าสูงสุดของ throughput ที่เป็นไปได้คือ S*(1-p) เมื่อ S = physical layer data rate และ p = block error rate
  - ดังนั้นเมื่อ BLER = 0% จะได้ RLC throughput = 384 kbps
  - เมื่อ BLER > 0% ถ้า *status period timer* มีค่าต่ำ ทำให้ไม่มีโอกาสที่การเลื่อนของ transmission window จะถูกขัดขวาง เพราะฉะนั้นค่าของ throughput จะเป็นไปตามสมการ S*(1-p) แต่เมื่อค่า *status period timer* เพิ่มขึ้นจนถึง threshold ค่า throughput จะเริ่มลดลง เนื่องจาก transmission window ถูกขัดขวางเพราะมี RLC PDU ที่ต้องถูกทำการส่งซ้ำ
- RLC goodput
  - ที่ BLER = 0% ไม่มีการขัดขวางการเลื่อน transmission window หรือการส่งซ้ำเกิดขึ้น ดังนั้น bandwidth ของ wireless link ทั้งหมดจะถูกใช้ในการส่ง RLC PDU ตัวใหม่เสมอ ซึ่งแต่ละ PDU ในโหมด AM นี้มี header ขนาด 2 byte เพราะฉะนั้น RLC goodput จะได้มาจาก 384 คูณด้วย 38/40 ซึ่งจะได้เท่ากับ 364.8 kbps (เมื่อ 40 คือขนาดทั้งหมดของ PDU 1 ตัว) ค่า goodput นี้จะคงที่ตลอด การเปลี่ยนแปลงค่า *status period timer* ไม่มีผลกระทบใดๆ
  - เมื่อ BLER > 0% ค่าของ *status period timer* ที่จะให้ค่า goodput สูงสุดประมาณมากกว่าค่า RTT เล็กน้อย และ
  - ที่ *status period timer* ต่ำ ให้ throughput สูง แต่ goodput ต่ำ เพราะ bandwidth ของ wireless link ถูกใช้เพื่อส่ง RLC PDU ที่ร้องขอให้มีการส่งซ้ำโดยไม่จำเป็นด้วย

- BLER ที่สูงจะให้ goodput ที่ต่ำกว่าเมื่อ BLER ต่ำ

1.2.1.7 ผลกระทบของ poll timer เมื่อมีค่า BLER มาเกี่ยวข้อง

ทำการวิเคราะห์ผลกระทบของ *poll timer* โดยดูจากค่าของ RLC SDU delay, RLC throughput และ RLC goodput เช่นกันได้ผลดังนี้

- RLC SDU delay เมื่อทำการเปลี่ยนค่า *poll timer* และ BLER
  - เมื่อ BLER = 0% นั่นคือ ค่า *poll timer* ไม่มีผลใดๆ ต่อ SDU delay เช่นกัน ค่าของ SDU delay จึงเท่ากับผลรวมของเวลา one-way latency และเวลา 1 TTI
  - เมื่อ BLER > 0% ค่า SDU delay จะสูงเมื่อ *poll timer* มีค่าต่ำ เพราะการส่งซ้ำของ RLC PDU ที่ไม่จำเป็นจำนวนมาก ทำให้ PDU ตัวใหม่ที่ต้องการส่งออกถูกกันไว้ที่ฝั่งส่ง
  - ค่า SDU delay ต่ำสุดเมื่อค่า *poll timer* สูงกว่า RTT เล็กน้อย
  - ค่า *poll timer* เพิ่มขึ้น SDU delay จะเริ่มเพิ่มขึ้นด้วยสำหรับ BLER ที่ใหญ่ (15%) หรือค่า SDU delay คงที่สำหรับ BLER ที่ต่ำกว่า (10%)
- RLC throughput
  - ค่าสูงสุดของ throughput ที่เป็นไปได้คือ S*(1-p) เมื่อ S = physical layer data rate และ p = block error rate
  - เมื่อ BLER = 0% นั่นคือมีการรับ status report 1 ตัวใน 1 TTI ซึ่งเป็น status report ที่เนื่องมาจากการ trig ของ *last PDU in buffer* ในกรณีนี้จะได้ throughput เต็มที่ ที่ 384 kbps
  - เมื่อ BLER > 0% ค่า throughput จะให้ผลการวิเคราะห์ที่คล้ายกับ *status period timer*
- RLC goodput
  - เมื่อ BLER = 0% ค่า goodput สูงสุดไม่ว่า *poll timer* จะเป็นเท่าใดคือ 364.8 kbps ด้วยเหตุผลเดียวกับใน *status period timer*

เมื่อ BLER > 0% จะได้ goodput ต่ำเมื่อ *poll timer* มีค่าต่ำ และค่า goodput สูงสุดจะอยู่บริเวณที่ *poll timer* มากกว่า RTT เล็กน้อย

1.2.2 วิธีที่ถูกนำเสนอเพิ่มเติมเพื่อทำให้ประสิทธิภาพของโปรโตคอลดีขึ้น

นอกจากการปรับแต่งค่าพารามิเตอร์ของโปรโตคอลเองแล้ว ในส่วนของ RLC ฝั่งรับ เช่น ขนาดบัฟเฟอร์ และกลไกการร้องขอให้มีการส่งซ้ำ (Automatic Repeat Request, ARQ) ก็มีผลต่อประสิทธิภาพการทำงานของโปรโตคอลเช่นกัน โดยปกติตามเอกสารกำหนด

คุณลักษณะสำหรับ RLC ที่ 3GPP ได้ระบุไว้ให้ใช้วิธีการควบคุมความผิดพลาดด้วยการจัดการ window ตามแบบของ Selective-repeat ARQ ซึ่งมีปัญหาในเรื่องเกี่ยวกับการเรียงลำดับข้อมูลที่ฝั่งรับ และที่ฝั่งส่งเองก็มีโอกาสที่จะเกิดการติดตายเนื่องจาก RLC เป็นโปรโตคอลที่การส่งซ้ำจะเกิดขึ้นตามการตอบกลับที่ได้รับ ดังนั้นจึงมีผู้นำเสนอวิธีต่าง ๆเพื่อแก้ไขปัญหาเหล่านี้

1.2.2.1 วิธีควบคุมการครอบครองบัฟเฟอร์จัดเรียงลำดับใหม่ (Re-ordering Buffer)

ใน ARQ ของ 3GPP ฝั่งรับจำเป็นต้องมีการเรียงลำดับใหม่ของข้อมูล PDU ที่รับเข้ามา ดังนั้นฝั่งรับจึงต้องมีบัฟเฟอร์สำหรับเก็บข้อมูล PDU ชั่วคราวที่รับเข้ามาแล้วไม่มี error ใด ๆเกิดขึ้น ข้อมูล PDU จะคงอยู่ที่บัฟเฟอร์จัดเรียงลำดับใหม่ไปจนกระทั่งข้อมูลของ PDU ทั้งหมดลำดับก่อนหน้าได้รับมาอย่างถูกต้อง เพื่อป้องกันการเกิด overflow ของบัฟเฟอร์จัดเรียงลำดับใหม่เท่ากับเป็นการลดการคงอยู่ของข้อมูล PDU ที่บัฟเฟอร์นี้ จึงมีความจำเป็นที่จะต้องลดการครอบครองที่บัฟเฟอร์จัดเรียงลำดับใหม่ ใน ARQ ของ 3GPP การลดการครอบครองบัฟเฟอร์จัดเรียงลำดับใหม่ สามารถทำได้โดยการลดขนาดของ window อย่างไรก็ตามการลดขนาด window นั้นจะเป็นการลดประสิทธิภาพของ throughput และเวลาหน่วงด้วย

วิธีการแรก [6] จุดประสงค์เพื่อที่จะลดการครองบัฟเฟอร์จัดเรียงลำดับใหม่พร้อมกับการหยุดการเสื่อมประสิทธิภาพลงของเวลาหน่วงและ throughput ได้นำเสนอวิธีการควบคุมการครองบัฟเฟอร์ โดยสร้างค่า threshold ใน window ของฝั่งรับ และข้อมูล PDU ที่นอกเหนือจาก threshold (แต่อยู่ใน window) ถูกจัดการโดยใช้ go-back-N ARQ โดยเป็นการเพิ่มเติมเสริมวิธีการ ARQ เดิมของ 3GPP ที่จากการทำงานเดิมฝั่งส่งทำการส่ง data PDUs ไปยังฝั่งรับ แล้วฝั่งรับก็จะทำงานโดยขึ้นอยู่กับการรับ data PDUs จะมีการตรวจสอบความผิดพลาดอย่างละเอียดและบันทึกผลการดักจับความผิดพลาดที่พบไว้บน status PDUs ในการทดสอบจำลองระบบสมมติว่า status PDU จะถูกส่งเฉพาะเมื่อถึงคาบเวลาไปยังฝั่งส่ง เมื่อ data PDU มาถึงพร้อมที่ RLC ฝั่งส่งก็จะถูกเก็บไว้ที่ท้ายสุดของบัฟเฟอร์ฝั่งส่ง data PDUs ที่ถูกเก็บไว้ในบัฟเฟอร์ส่งนี้จะถูกนำส่งออกไปตามกระบวนการ FCFS (First Come First Serve) และหากฝั่งส่งได้รับ status PDU ที่ฝั่งรับส่งมาให้ PDU ใดที่ถูกตอบกลับว่ามีความผิดพลาดหรือสูญหายเกิดขึ้น ฝั่งส่งจะนำ PDU นั้นมาเก็บไว้ที่ส่งหัวของบัฟเฟอร์ส่งซึ่งมีความสำคัญสูงสุดและจะถูกนำส่งออกไปก่อน นั้นคือเพื่อให้ PDU นั้นได้รับการส่งซ้ำก่อนที่จะทำการส่ง PDU อื่นต่อไป ทั้งฝั่งส่งและฝั่งรับจะมีการใช้งาน window เพื่อควบคุมการส่ง โดยสมมติว่า W_T และ W_R แทนขนาดของ window ฝั่งส่งและฝั่งรับตามลำดับ สมมติว่า data PDU ที่มี sequence number เท่ากับ *SN* มาถึงที่ฝั่งรับ ณ เวลาที่ t และให้ *HSN* เป็น sequence number สูงสุดในจำนวนของ sequence number N ที่ซึ่ง data PDU ที่มี sequence number อยู่ในช่วง [1,N] ถูกรับไว้เรียบร้อยแล้วโดยไม่มีความผิดพลาดใด ๆที่เวลา t ดังนั้นตามค่าของ *SN* ฝั่งรับจะมีการกระทำหนึ่งในสองวิธีดังนี้

(1) ถ้า $HSN+1 \leq SN \leq HSN+W_R$ ฝั่งรับยอม data PDU ที่มี sequence number เท่ากับ SN และถ้าข้อมูลไม่มีข้อผิดพลาดใด ๆ ก็จะถูกเก็บไว้ที่บัฟเฟอร์จัดเรียงลำดับเพื่อรอ data PDU อื่นก่อนจะทำการรวมกลับเป็น SDU แล้วส่งต่อให้โปรโตคอลชั้นบนต่อไป

(2) ถ้า $SN \geq HSN+W_R+1$ ฝั่งรับจะละทิ้ง data PDU ที่มี sequence number เท่ากับ SN ด้วยเหตุที่ว่า PDU นั้นอยู่นอกเหนือ window รับ

วิธีการที่นำเสนอเพื่อเสริมจากการทำงานดังกล่าว คือ เพิ่มการกำหนดค่า threshold (ν) ขึ้นใช้ที่ window ฝั่งรับ โดยค่า $\nu \in \{0,...,W_R\}$ การพิจารณา sequence number ของ data PDU ที่รับมาก็จะเปลี่ยนไป โดย data PDU ที่ sequence number อยู่ใน window รับแต่มีค่าไม่เกิน threshold จะถูกจัดการด้วย selective repeat ARQ และหากมี sequence number มากกว่า threshold แต่ยังอยู่ใน window รับ PDU นั้นจะถูกจัดการด้วย go-back-N ARQ แทน วิธีการที่ใช้ในกาพิจารณา ณ ฝั่งรับเปลี่ยนไปดังนี้

(1) ถ้า $HSN+1 \leq SN \leq HSN+\nu$ ฝั่งรับยอม data PDU ที่มี sequence number เท่ากับ SN และถ้าข้อมูลไม่มีข้อผิดพลาดใด ๆ ก็จะถูกเก็บไว้ที่บัฟเฟอร์จัดเรียงลำดับเพื่อรอ data PDU อื่นก่อนจะทำการรวมกลับเป็น SDU แล้วส่งต่อให้โปรโตคอลชั้นบนต่อไป

(2) ถ้า $HSN+\nu+1 \leq SN \leq HSN+W_R$ ฝั่งรับทำการตรวจสอบ ถ้าทุก data PDUs ที่มี sequence number ในช่วง $[HSN+\nu+1, SN-1]$ ถูกรับเรียบร้อยโดยไม่มีข้อผิดพลาด ฝั่งรับก็จะยอมรับ data PDU ที่มี sequence number นี้ และทำการเก็บไว้ที่บัฟเฟอร์จัดเรียงลำดับถ้ามันไม่มีข้อผิดพลาดใด ๆ สำหรับกรณีอื่น ๆ ฝั่งรับก็จะทำการละทิ้ง data PDU นั้น

(3) ถ้า $SN \geq HSN+W_R+1$ ฝั่งรับจะละทิ้ง data PDU ที่มี sequence number เท่ากับ SN ด้วยเหตุที่ว่า PDU นั้นอยู่นอกเหนือ window รับ

สำหรับการประเมินประสิทธิภาพ ได้เลือกวัดประสิทธิภาพโดยใช้แบบจำลองและสังเกตจากค่าการครอบครองบัฟเฟอร์จัดเรียงลำดับสูงสุด เวลาหน่วงเฉลี่ยของ data PDU และ normalized maximum throughput โดยเวลาหน่วงของ data PDU ถูกนิยามด้วยเวลาตั้งแต่ data PDU นั้นมาถึงที่ฝั่งส่งจนกระทั่งได้รับการตอบกลับที่ไม่มีข้อผิดพลาดจาก status PDU ที่ฝั่งรับส่งมาให้ สภาวะในการจำลองคือ ให้ความยาวของ data PDU คงที่ และแต่ละ data PDU ถูกส่งด้วย data rate คงที่เช่นกัน ความน่าจะเป็นที่จะเกิดความผิดพลาดกับแต่ละ data PDU บน forward channel (ε) เท่ากับความน่าจะเป็นที่จะเกิดความผิดพลาดกับแต่ละ status PDU บน reverse channel (δ) และเป็นอิสระแก่กัน

เพื่อทดสอบวิธีการที่นำเสนอ (กำหนด threshold) ต่อผลกระทบค่าครอบครอง

บัฟเฟอร์จัดเรียงลำดับสูงสุดและเวลาหน่วงเฉลี่ยของ data PDU ได้ออกแบบการทดลองโดยกำหนดขนาด window ฝั่งรับและฝั่งส่งให้คงที่และมีค่าเท่ากันเท่ากับ 32 PDU จากนั้นทดลองเปลี่ยนค่า threshold เป็น 2, 4, 8 และ 16 แล้วทำการแปรค่าปริมาณข้อมูล (data PDUs/time unit) เรื่อยๆ ผลที่ได้คือ การครอบครองบัฟเฟอร์จัดเรียงลำดับลดลงตามค่าของ threshold กล่าวคือ ยิ่งให้ threshold มีค่าต่ำ การครองบัฟเฟอร์ก็จะยิ่งน้อยลงด้วย แต่ threshold ค่าต่ำกลับทำให้เวลาหน่วงเฉลี่ยของ data PDU เพิ่มสูงขึ้น จากนั้นได้ทำการทดลองโดยเอาขนาดของ window มาใช้ในการวิเคราะห์ผลกระทบด้วย ได้แบ่งพิจารณาจาก 4 กรณี คือ (1) W=16, ν = 8, (2) W=8, no threshold, (3) W=32, ν =16 และ (4) W=16, no threshold พบว่าปริมาณการครองบัฟเฟอร์ของกรณีที่ 1 และ 2 ไม่แตกต่างกันมากนัก เช่นเดียวกับเมื่อเปรียบเทียบกรณีที่ 3 และ 4 แต่เมื่อทำการเปรียบเทียบเวลาหน่วงของ data PDU เฉลี่ย เมื่อปริมาณข้อมูลมากขึ้น กรณีที่ 1 จะให้ผลดีกว่ากรณีที่ 2 (เวลาหน่วงต่ำกว่า) และกรณีที่ 3 ดีกว่ากรณีที่ 4 นั่นคือ การกำหนดขนาด window ใหญ่พร้อมกับค่า threshold จะให้ผลปริมาณการครองบัฟเฟอร์เทียบเท่ากับขนาด window เล็กแต่ไม่มี threshold แต่เวลาหน่วงสำหรับกรณีที่ window มีขนาดใหญ่จะดีกว่า และเมื่อเปรียบเทียบกรณีที่ 1 กับกรณีที่ 4 ซึ่งให้ขนาดของ window เท่ากัน กรณีที่ 1 กำหนด threshold เท่ากับ 8 ส่วนกรณีที่ 4 ไม่มี threshold พบว่า การมี threshold ช่วยลดขนาดของการครองบัฟเฟอร์ได้ดังที่ได้ทำการวิเคราะห์ไว้ก่อนหน้านี้ และยังช่วยทำให้เวลาหน่วงของ data PDU ดีขึ้นอีกด้วย หากวิเคราะห์ผลของ normalized maximum throughput พบว่า การลดขนาดของ window โดยตรง (เทียบกรณีที่ 2 และ 4) ส่งผลทำให้ throughput ลดลง มากกว่าการที่ให้ขนาด window เท่าเดิมแต่เพิ่มการกำหนดขนาด threshold (เทียบกรณีที่ 1 และ 4)

จากที่กล่าวมาสามารถสรุปได้ว่าปัญหาเกี่ยวกับบัฟเฟอร์จัดเรียงลำดับสามารถแก้ได้โดยลดขนาดของ window ลงแต่การลดขนาด window ทำให้ประสิทธิภาพเวลาหน่วง และ throughput ลดลง การแก้ปัญหาโดยให้ขนาด window คงที่และเสริมด้วยการมีค่า threshold มาจัดการด้วย go-back-N ARQ ช่วยลดการครองบัฟเฟอร์พร้อมกับไม่ส่งผลต่อประสิทธิเวลาหน่วง และ throughput ของโปรโตคอลมากนักด้วย

นอกจากวิธีกำหนด threshold ที่ window ฝั่งรับแล้ว ยังมีผู้นำเสนอวิธีที่เรียกว่า Extended Go-bace-N [5]โดยมีเงื่อนไขการทำงานว่าฝั่งรับจะละทิ้งไม่สนใจ data PDU ที่รับมาถ้า sequence number ของมันอยู่ระหว่างค่า sequence number สูงสุดและต่ำสุดของ data PDU ที่พบว่ามีความผิดพลาดเกิดขึ้น ที่เวลา t สมมติว่า data PDU ที่มี sequence number เท่ากับ SN มาถึงที่ฝั่งรับ และสมมติว่ามี data PDU บางส่วนถูกเก็บในบัฟเฟอร์เรียงลำดับก่อนแล้ว ณ เวลา t ให้ $N_R^I(t)$ เป็น sequence number ต่ำสุดของ data PDU ทั้งหมดที่อยู่แล้วในบัฟเฟอร์เรียงลำดับ และ $N_T^F(t)$ เป็น sequence number ต่ำสุดใน data PDU ทั้งหมดที่ฝั่งส่งกำลังจะได้รับการตอบกลับว่าเกิดข้อผิดพลาด ณ เวลา t

(1) ถ้า $N_R^F(t)+1 \leq SN \leq N_R^I(t)-2$ ฝั่งรับละทิ้ง data PDU ที่มี sequence number เท่ากับ SN เนื่องจากเป็น PDU ที่อยู่ในช่วงที่มีความผิดผลาดเกิดขึ้น

(2) ถ้า SN = $N_R^I(t)-1$ และ data PDU นั้นไม่มีข้อผิดพลาด ฝั่งรับจะยอมรับ data PDU นั้น เพราะเป็น PDU ที่มีความต่อเนื่องกับ PDU ที่ได้รับมาก่อนหน้านี้แล้ว

ในการประเมินวิธีการที่นำเสนอทำโดยสร้างแบบจำลองแล้วทำการวัดค่าการครอบครองบัฟเฟอร์สูงสุดและเวลาเฉลี่ยที่ใช้อยู่ชั่วคราวในบัฟเฟอร์เรียงลำดับ ค่า throughput สูงสุด และค่าเวลาหน่วงเฉลี่ยของ data PDU ที่ฝั่งส่ง สิ่งแวดล้อมที่ใช้ในการจำลองคือ ให้ความยาวของ data PDU คงที่และแต่ละ data PDU ถูกส่งด้วย data rate คงที่ นั่นคือ transmission time ของ data PDU จะคงที่ด้วย data PDU ถูกส่งผ่าน forward channel ที่ซึ่งความน่าจะเป็นของความผิดพลาดที่จะเกิดขึ้นในแต่ละ data PDU (ε)เป็นอิสระแก่กัน และมีค่าเท่ากับความน่าจะเป็นของความผิดพลาดที่จะเกิดขึ้นในแต่ละ status PDU (δ) ที่จะถูกส่งผ่าน reverse channel สมมติว่าฝั่งรับสามารถตรวจจับความผิดพลาดของแต่ละ data PDU ได้อย่างสมบูรณ์และทำการส่ง status PDU กลับไปฝั่งส่งเป็นช่วง ๆคาบเวลา เมื่อเปรียบเทียบปริมาณการครองบัฟเฟอร์และเวลาเฉลี่ยที่ PDU ใช้ในบัฟเฟอร์เรียงลำดับของวิธีที่นำเสนอกับวิธีของ 3GPP เดิมพบว่า วิธีการที่นำเสนอให้ค่าทั้งสองดีกว่า แต่กลับทำให้เวลาหน่วงเฉลี่ยของ data PDU ที่ฝั่งส่งและ throughput แย่ลงกว่าเดิม เมื่อทดลองเปลี่ยนขนาด window ให้เล็กลงพบว่าทั้งวิธีเดิมและวิธีการที่นำเสนอทำให้ปริมาณการครองบัฟเฟอร์และเวลาเฉลี่ยที่ PDU ใช้ในบัฟเฟอร์จัดเรียงลำดับลดลง (ให้ผลดีขึ้น) แต่ในทางกลับกันจะยิ่งทำให้เวลาหน่วงเฉลี่ยที่ PDU ใช้ตั้งแต่มาถึงฝั่งส่งจนได้รับการตอบกลับว่าถูกต้อง และ throughput แย่มากกว่าเก่า

1.2.2.2 วิธีป้องกันการหยุดหรือหน่วงการทำงานของโปรโตคอล RLC

เนื่องจาก RLC สำหรับ UMTS เป็นโปรโตคอลที่การส่งซ้ำจะขึ้นกับการตอบกลับจากฝั่งรับ ดังนั้นอาจทำให้เกิดการหยุดชั่วคราวหรือการติดตายของโปรโตคอลขึ้นได้และทำให้ประสิทธิภาพของระบบลดลง การหยุดของโปรโตคอลคือ สถานการณ์พิเศษซึ่งไม่อนุญาตให้มีการส่งแม้ว่าฝั่งส่งจะมีแพ็กเก็ตรอส่งอยู่ที่โหนด เป็นการลด throughput ของเครือข่ายและเพิ่มเวลาหน่วงของการส่งแพ็กเก็ต และอาจเกิดการ trig ให้โปรโตคอลทำงานในโหมดที่ไม่ถูกต้องจนเป็นเหตุให้การทำงาน reset ดังนั้นไม่ควรที่จะให้เหตุการณ์เช่นนี้เกิดขึ้นบ่อย ๆ เพื่อรักษาไว้ซึ่งความคงตัวและประสิทธิภาพของระบบ โดยปกติแล้วการหยุดการทำงานของโปรโตคอลเกิดได้จากหลายสาเหตุ เช่น ขนาดของ window ส่งไม่เพียงพอ และการสูญหายหรือการมาไม่เป็นลำดับของการ

ตอบกลับ

ตามที่ RLC เป็นโปรโตคอลที่การส่งซ้ำจะขึ้นกับการตอบกลับจากฝั่งรับที่ซึ่งไม่อนุญาตให้มีการส่งซ้ำนอกจากจะได้รับการตอบกลับว่าเกิดความผิดพลาด การหยุดการทำงานของโปรโตคอลอาจเกิดขึ้นถ้าไม่มีการกำหนดให้ใช้งานกลไกการ polling ตัวอย่างของปัญหานี้ เช่น พิจารณาสถานการณ์ที่ซึ่งไม่มีข้อมูลใหม่เข้ามาที่โหนดฝั่งส่ง (จบการขนถ่ายข้อมูล web page เป็นต้น) และ RLC PDUs ที่เกิดขึ้นใหม่ทั้งหมดได้ถูกส่งออกไปแล้ว บัฟเฟอร์ส่งว่าง แต่ยังมีบาง PDUs ที่ตกค้างในบัฟเฟอร์ส่งซ้ำเพราะยังไม่ได้รับการตอบกลับอาจเป็นผลจากการสูญหายของการตอบกลับหรือ status PDU จากฝั่งรับ PDUs ที่ค้างเหล่านี้จะไม่ถูกส่งซ้ำเพราะไม่มีการตอบกลับว่าเกิดความผิดพลาด (*NACKed*) และไม่ถูกนำออกหรือละทิ้งจากบัฟเฟอร์เนื่องจากจำนวนการส่งของแต่ละ PDU น้อยกว่าจำนวนสูงสุดที่อนุญาตให้ถูกส่งได้ (*MAXDAT*) แม้จะมีการกำหนดให้ใช้งานฟังก์ชันละทิ้งตามเวลา (timer based SDU discard function) ที่จะทำการละทิ้ง PDUs ที่ค้างเหล่านั้นหากรอคอยเป็นเวลาเกินกำหนด ซึ่งช่วงเวลาที่รอคอยจะเป็นช่วงเวลาที่ช่องสัญญาณส่งว่างแม้ว่าจะมีแพ็กเก็ตอยู่ในฝั่งส่ง และที่ฝั่งรับแม้ว่าจะมีกลไกที่ฝั่งรับสามารถบังคับให้ฝั่งส่งทำการส่งซ้ำได้ (ส่ง status PDU ไปให้ฝั่งส่ง) ผ่านการใช้ trigger 2 ตัว ได้แก่ *Detection of missing PDUs* และ *Timer based status transmission* แต่การหยุดการทำงานของโปรโตคอลก็อาจจะยังปรากฏอยู่เนื่องจากในบางครั้ง PDUs ที่ฝั่งส่งทำการส่งมาที่มี sequence number สูง ๆ ทั้งหมดอาจจะสูญหายและไม่เคยมาถึงหรือถูกถอดรหัสตีความที่ฝั่งรับเลย ฝั่งรับจะไม่สามารถตรวจจับ PDUs ที่สูญหายเหล่านั้นและไม่สามารถส่ง status report กลับไปแจ้งฝั่งส่งด้วย *Timer based status transmission* ฝั่งรับจะส่ง status PDU ไปหาฝั่งส่งทุกครั้งเมื่อ timer หมดอายุ เพื่อว่าในบางครั้งจะเป็นการกู้การตอบกลับที่อาจจะสูญหายไป อย่างไรก็ตามฝั่งรับจะสามารถส่งเฉพาะ status ของ PDU ที่มันรับหรือตรวจเจอเรียบร้อยแล้วเท่านั้น สำหรับ PDU ที่สูญหายไปที่ไม่สามารถรับได้นั้นฝั่งรับเองก็ไม่สามารถแจ้งฝั่งส่งเกี่ยวกับ status ของ PDU เหล่านั้น ดังนั้นการหยุดการทำงานของโปรโตคอลจึงอาจจะยังปรากฏอยู่ ในที่นี้จึงได้มีการนำเสนอ 3 วิธีที่จะเป็นทางเลือกในการป้องกันการหยุดการทำงานของโปรโตคอล RLC และเปรียบเทียบผลผ่านการใช้แบบจำลอง

1.2.2.2.1 การส่งซ้ำของ Last transmitted packet

ในวิธีนี้ PDU ที่ถูกส่งออกไปครั้งแรกเมื่อเร็ว ๆนี้ (เช่น PDU ที่มี sequence number เท่ากับ *VT(S)-1* เมื่อ *VT(S)* เป็นตัวแปรสถานะของฝั่งส่งที่บอกถึง sequence number ของ PDU ถัดไปที่จะถูกส่งเป็นครั้งแรก) จะไม่ถูกนำออกจากบัฟเฟอร์ส่งซ้ำ นอกเสียจากว่า PDUs ทั้งหมดที่มี sequence number ต่ำกว่าถูกส่งออกไปสำเร็จสมบูรณ์ หรือถูกละทิ้ง กำหนดให้ PDU ที่มี sequence number *VT(S)-1* สามารถถูกส่งซ้ำได้แม้ว่ามันจะไม่ได้รับ NACKed หรือ

ได้รับเป็นการตอบกลับที่เป็น ACKed ก่อนหน้าเรียบร้อยแล้ว ถ้ามีเหตุการณ์ที่เป็นไปตามเงื่อนไขดังนี้

- ไม่มี PDU ใหม่ หรือ PDU ที่ได้รับ NACKed กำลังรออยู่ที่ฝั่งส่ง
- การส่ง PDU ที่ *V(S)-1* ครั้งก่อนใช้เวลายาวนานเกินกว่าช่วงเวลาที่กำหนดไว้
- PDU ที่ *V(S)-1* จะถูกส่งพร้อมด้วยบิต poll ของมันที่ถูกกำหนดตาม polling trigger เช่น *Timer_Poll* เพื่อหลีกเลี่ยงการเกิดการ poll มากเกินไปและ overhead ของการตอบกลับ

ข้อดีของการ polling ลักษณะนี้คือ เป็นการเพิ่มความน่าจะเป็นที่ฝั่งรับจะสามารถตรวจจับ PDUs ที่มี sequence number ต่ำกว่า PDU ที่ถูกส่งตัวล่าสุดทั้งหมดที่สูญหาย ฝั่งรับจะส่ง status report แจ้งสถานะของ PDUs ไปจนถึงตัวที่ sequence number เท่ากับ *V(S)*-1 เพื่อว่าฝั่งส่งจะได้โต้ตอบได้อย่างทันท่วงที

1.2.2.2.2 การส่งซ้ำของ Earliest transmitted packet

ในวิธีนี้ ฝั่งส่งจะไม่เก็บ PDU ใด ๆที่มีการตอบกลับถูกต้องเรียบร้อยแล้วไว้ในบัฟเฟอร์ส่งซ้ำ อย่างไรก็ตามสำหรับ PDU ที่ยังคงค้างอยู่ในบัฟเฟอร์ส่งซ้ำจะได้รับอนุญาตให้เกิดการส่งซ้ำได้แม้ว่ามันจะไม่ได้รับ NACKed ถ้ามีเหตุการณ์ที่เป็นไปตามเงื่อนไขดังนี้

- ไม่มี PDU ใหม่ หรือ PDU ที่ได้รับ NACKed กำลังรออยู่ที่ฝั่งส่ง
- การส่งของ PDU ใด ๆ ที่ค้างอยู่และไม่ได้รับ NACKed ครั้งก่อนใช้เวลายาวนานเกินกว่าช่วงเวลาที่กำหนดไว้
- PDU ที่ค้างอยู่ที่มีเวลาส่งครั้งก่อนแรกสุดจะถูกส่งซ้ำพร้อมกับการกำหนดบิต poll ของมันที่ถูกตั้งค่าจาก polling trigger

ข้อดีของวิธีนี้คือ การส่งซ้ำแพ็กเก็ตแรกส่งช่วยลดเวลาหน่วงของแพ็กเก็ต และลดความน่าจะเป็นที่แพ็กเก็ตจะถูกละทิ้งจากบัฟเฟอร์ของฝั่งส่งถ้าแพ็กเก็ตนั้นไม่ได้รับการตอบกลับเป็นเวลานาน

1.2.2.2.3 รวมการส่งซ้ำของทั้งสองวิธีข้างต้น

ในวิธีนี้ ฝั่งส่งจะไม่เก็บ PDU ใด ๆที่มีการตอบกลับถูกต้องเรียบร้อยแล้วไว้ในบัฟเฟอร์ส่งซ้ำ เมื่อใดก็ตามที่ไม่มี PDU ใหม่หรือ PDU ที่ได้รับ NACKed กำลังรออยู่ที่ฝั่งส่ง ฝั่งส่งสามารถส่ง PDU ที่ค้างอยู่ในบัฟเฟอร์ส่งซ้ำได้ ถ้า PDU ตัวที่ถูกส่งตัวล่าสุดอยู่ค้างอยู่ในบัฟเฟอร์ มันจะถูกส่งซ้ำพร้อมกับบิต poll ของมันที่ถูก set หลังจากตัวจับเวลาของมันหมดอายุ

แต่ถ้า PDU ตัวที่ถูกส่งตัวล่าสุดไม่ได้ค้างอยู่ในบัฟเฟอร์ แล้ว PDU ที่มีเวลาส่งครั้งก่อนแรกสุดที่ยังคงค้างในบัฟเฟอร์จะถูกส่งแทนพร้อมกับบิต poll ของมันที่ถูก set วิธีนี้รวมข้อดีของ 2 วิธีก่อนหน้าไว้ด้วยกัน โดยเพิ่มส่วนที่จะทำการเลือกแพ็กเก็ตที่จะทำการส่งซ้ำ ถ้า PDU ตัวที่ถูกส่งตัวล่าสุดไม่ได้รับการตอบกลับมันจะถูกส่งซ้ำเป็นอันดับแรกเพื่อเพิ่มความน่าจะเป็นที่จะสำเร็จของมัน เพื่อว่าฝั่งรับจะสามารถปรับค่าตัวแปรสถานะของมันได้เร็วและส่งสถานะกลับไปยังฝั่งส่ง อย่างไรก็ตามถ้า PDU ตัวที่ถูกส่งตัวล่าสุดได้รับการตอบกลับ มันก็จะส่งผลให้ฝั่งรับรู้ถึง sequence number สูงสุดปัจจุบันของ PDU ที่ถูกส่ง ในขณะที่การส่ง PDU ที่มีเวลาส่งครั้งก่อนแรกสุดจะมีประโยชน์ในแง่ของการลดเวลาหน่วงและลดความน่าจะเป็นที่แพ็กเก็ตจะถูกละทิ้ง

ในแบบจำลอง สิ่งหลักที่สนใจคือเวลาหน่วงและ throughput ของการดาวน์โหลด web page ใช้การ poll ทุกๆ การ trig ของ *POLL_SDU* trigger และนอกจากนี้ยังใช้ trigger *last PDU in transmission buffer* และ *last PDU in retransmission buffer* ในแบบจำลองด้วย เพราะว่า trigger 2 ตัวนี้ สามารถป้องกันการเกิดการติดตายของโปรโตคอล และมีการกำหนดใช้งาน *Timer based SDU discard function* โดยให้ตัวจับเวลามีค่าเป็น 4 วินาที ในการทดลองได้ทำการเปรียบเทียบ throughput, เวลาหน่วง, overhead ของ status report และเปอร์เซ็นต์การละทิ้ง SDU ของทั้ง 3 วิธีเมื่อให้ช่วงเวลาของการ poll แตกต่างกัน ผลที่ได้พบว่าวิธีที่ 3 ให้ throughput สูงที่สุด และวิธีแรกให้ผลดีกว่าวิธีที่ 2 เพราะเป็นการเพิ่มความน่าจะเป็นที่ฝั่งรับสามารถรับ PDU ที่ถูกส่งสุดท้ายสำเร็จ ทำให้ฝั่งรับสามารถส่งการตอบกลับของ PDUs ที่สูญหายทั้งหมดไปยังฝั่งส่ง เพื่อให้การทำงานของโปรโตคอลทำงานได้ต่อไปหลังจากฝั่งรับปรับค่าตัวแปรสถานะของตัวเองแล้ว การส่งซ้ำ PDU แรกสุดที่ค้างอยู่จะเป็นการเร็วกว่าที่จะส่ง PDU สุดท้ายเพราะตัวจับเวลาของตัวแรกจะหมดอายุก่อน ดังนั้นวิธีการที่ 3 จึงป้องกันการเกิดการหยุดการทำงานของโปรโตคอลได้ มีประสิทธิภาพที่สุดและให้ throughput สูงสุดด้วย และพบว่า throughput จะลดลงตามช่วงเวลาการ poll ที่เพิ่มขึ้น

วิธีที่ 3 ยังให้ประสิทธิภาพของเวลาหน่วงดีที่สุดเช่นกัน วิธีที่ 2 จะให้ค่าเวลาหน่วงที่ใหญ่เพราะว่า PDU ที่ถูงส่แรกสุดสามารถถูกส่งซ้ำได้เฉพาะหลังจากที่ตัวจับเวลาหมดอายุ ซึ่งเป็นเหมือนกันหมดสำหรับทุก PDUs ที่คงค้างอยู่ เมื่อเปรียบเทียบกับวิธีอื่น ฝั่งส่งต้องรอเป็นเวลายาวนานกว่าเพื่อที่จะจบช่วงการหยุดการทำงานและกลับไปดำเนินการต่อไป โดยปกติเวลาหน่วงเพิ่มขึ้นตามช่วงเวลาการ poll ที่สูงขึ้น และเมื่อเปรียบเทียบปริมาณของ status overhead พบว่า วิธีที่ 1 และ 2 ให้ค่าเกือบจะเหมือนกัน แต่จากกรณีที่ 3 จะสูงกว่า 2 วิธีแรกเล็กน้อย โดยเฉพาะเมื่อช่วงเวลาการ poll กว้างมากๆ เหตุผลเป็นเพราะหลังจากการส่งแพ็กเกตสุดท้ายออกไปสำเร็จ ฝั่งส่งจะส่งแพ็กเกตแรกสุดที่ค้างอยู่ถ้าตัวจับเวลาของมันหมดอายุและไม่จำเป็นต้องคอยให้ PDUs ที่ค้างอยู่ตัวอื่นหมดเวลาด้วย และสุดท้ายเปรียบเทียบอัตราการละทิ้ง SDU เป็นที่น่าสนใจว่าทั้ง 3 วิธีให้ผลเหมือนกัน ทั้งนี้ก็เนื่องมาจากการใช้งาน *Timer based SDU discard function* และคิดเป็นเปอร์เซ็นต์มีการละทิ้ง SDU น้อยมาก โดยเปอร์เซ็นต์การละทิ้ง SDU เพิ่ม

ตามช่วงเวลาการ poll ที่กว้างขึ้น

จากผลการวิเคราะห์ข้างต้นสามารถสรุปได้ว่า การรวมการส่งซ้ำแพ็กเกตล่าสุดหรือส่งแพ็กเกตแรกสุดให้ผลดีกว่าอีก 2 วิธีที่เหลือ โดยให้ค่า throughput สูงที่สุดและเวลาหน่วงต่ำที่สุด เนื่องจากความสามารถของอัลกอริทึ่มที่ทำการตรวจจับสถานการณ์การหยุดค้างได้เร็วและปรับค่าสถานะการส่งได้อย่างฉับไว ในการพัฒนาวิธีการส่งซ้ำแพ็กเกตที่ถูกส่งแรกสุดซับซ้อนมากที่สุดเพราะต้องมีการพิจารณาตัวจับเวลาของทุก ๆ แพ็กเกตที่ค้างอยู่

1.3 วัตถุประสงค์

เพื่อศึกษา ออกแบบวิธีการในการประเมินสมรรถนะของโปรโตคอลชั้นที่ 2 ของระบบโทรศัพท์เคลื่อนที่ยุคที่ 3

1.4 ขอบเขต

1) วิธีการที่ออกแบบสามารถประเมินสมรรถนะของโปรโตคอลชั้นที่ 2 ของระบบโทรศัพท์เคลื่อนที่ยุคที่ 3 (อ้างอิงตามเอกสาร 3GPP Release 5) เพื่อให้ได้แบบจำลองของต้นแบบสำหรับใช้ในการวิเคราะห์
2) ใช้ Network Simulation tool เพื่อทดสอบวิธีการที่ออกแบบ และสร้าง simulation model ที่สามารถประเมินสมรรถนะของโปรโตคอลและปรับเปลี่ยนสมรรถนะของโปรโตคอลนั้นให้ดีขึ้นได้

1.5 ขั้นตอนและวิธีการดำเนินการวิจัย

1) ศึกษาโครงสร้างของระบบเครือข่าย UMTS ในชั้นที่ 2 หรือชั้นเชื่อมโยงข้อมูลและศึกษาข้อมูลเพิ่มเติมถึงพารามิเตอร์หรือตัววัดต่าง ๆที่เป็นมาตรฐานใช้ในการวัดสมรรถนะของระบบโทรศัพท์เคลื่อนที่
2) ศึกษาในขั้นลึกถึงพารามิเตอร์หรือตัววัดต่าง ๆที่นำมาใช้สำหรับวัดสมรรถนะของเครือข่ายโทรศัพท์เคลื่อนที่ที่จำเพาะเจาะจงสำหรับชั้นที่ 2 เท่านั้น ค้นหาบทความทางวิชาการที่เกี่ยวกับการวัดสมรรถนะของระบบโทรศัพท์เคลื่อนที่ รวมทั้งจัดทำโครงร่างวิทยานิพนธ์
3) ออกแบบวิธีการที่จะนำมาใช้ตรวจวัดสมรรถนะของชั้นที่ 2 ของระบบโทรศัพท์เคลื่อนที่ยุคที่ 3 พร้อมกับสร้างระบบทดสอบ
4) ทดสอบวัดสมรรถนะของระบบทดสอบที่สร้างขึ้นด้วยวิธีการที่ได้ทำการศึกษาและออกแบบไว้ เปรียบเทียบผลที่ได้กับค่าทางทฤษฎี

5) วิเคราะห์ความสัมพันธ์ของค่าที่ตรวจวัด เพื่อวิเคราะห์สาเหตุที่ทำให้ประสิทธิภาพการทำงานของโปรโตคอลเปลี่ยนแปลงไป
6) ออกแบบปรับปรุงกลไกการทำงานภายในโปรโตคอล ทดสอบผลการจำลองวิเคราะห์ถึงความแตกต่างของการทำงานทั้งสองรูปแบบ
7) รวบรวมผลการทดสอบ สรุปผล และจัดทำรายงานฉบับสมบูรณ์

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้ simulation model ที่มีความสามารถในการประเมินสมรรถนะของโปรโตคอลชั้นที่ 2 ของระบบโทรศัพท์เคลื่อนที่ยุคที่ 3 (อ้างอิงตามเอกสาร 3GPP Release 5)
2) จากการประเมินทำให้รู้ถึงพารามิเตอร์ที่มีผลต่อสมรรถนะของโปรโตคอล ดังนั้นจะส่งผลให้เราสามารถรู้ได้ว่าควรปรับพารามิเตอร์ใดเพื่อทำให้โปรโตคอลมีสมรรถนะที่ดีขึ้น ช่วยให้มีการใช้ทรัพยากรของระบบอย่างคุ้มค่า
3) ผลงานในระหว่างที่ทำวิทยานิพนธ์สามารถนำไปตีพิมพ์ลงในบทความทางวิชาการ