# บทที่ 2

## การควบคุมความคับคั่งและการส่งข้อมูล TCP

การจัดการทราฟฟิกเป็นปัญหาหนึ่งในการออกแบบโครงข่ายความเร็วสูง วัตถุประสงค์การจัดการทราฟฟิกเพื่อเป็นการควบคุมความคับคั่งของโครงข่าย หรือการใช้ทรัพยากรของโครงข่ายอย่างมีประสิทธิภาพ และจัดการคุณภาพของการบริการให้กับผู้ใช้บริการ ATM สามารถรองรับคุณภาพการบริการสำหรับการส่งข้อมูลในระดับสูงและเตรียมการหลายๆอย่างให้กับการบริการที่ใช้เวลาจริง (real time) หรือไม่ใช้เวลาจริง (non-real time) TCP จะมีกลไกการควบคุมความคับคั่งและการไหลข้อมูลหลายๆแบบ เช่น slow start/congestion avoidance, fast retransmit/fast recovery และ selective acknowledgments ในโครงข่ายยังมีเครื่องมือเกี่ยวกับหน้าที่หลายอย่างเพื่อใช้งานสำหรับการจัดการทราฟฟิก เช่น connection admission control (CAC), policing, shaping, scheduling, buffer management และ feedback control หน้าที่ต่างๆเหล่านี้สามารถใช้เพื่อทำให้โครงข่ายมีประสิทธิภาพและการใช้ทรัพยากรดีที่สุดขณะมีการรับประกันคุณภาพการบริการให้กับผู้ใช้บริการ

ในวิทยานิพนธ์นี้จะมุ่งเจาะจงไปที่ปัญหาการจัดการทราฟฟิกสำหรับ TCP/IP บนการบริการUBR ในโครงข่าย ATM วัตถุประสงค์การออกแบบสถาปัตยกรรมของโครงข่ายและกลไกที่จะทำให้การส่งข้อมูล TCP มีประสิทธิภาพบนโครงข่าย ATM ที่มีการบริการหลายๆแบบ เราจะมุ่งไปที่การแก้ปัญหาการจัดการคิว (queues) และการทำให้โพรโตคอล TCP มีสมรรถนะสูงสุด ในบทนี้จะอธิบายถึง TCP/IP บนโครงข่าย ATM โดยในบทนี้จะมีการเรียงหัวข้อดังนี้ หัวข้อแรกจะพูดถึงมาตรฐานการบริการแบบต่างของโครงข่าย ATM หัวข้อที่ 2 จะอธิบายจะเกี่ยวกับ การควบคุมการส่งข้อมูลและความคับคั่งของ TCP โดยแยกออกเป็นดังนี้

- การควบคุมการส่งข้อมูล TCP (TCP Flow Control)
- พื้นฐานการควบคุมความคับคั่งของ TCP
- Slow Start และ Congestion Avoidance
- Fast Retransmission และ Fast Recovery

### 2.1 การบริการของ ATM

ในหัวข้อนี้จะพูดถึงมาตรฐานของการบริการใน ATM โดยมาตรฐานนี้ ATM Forum [1] และ International Telecommunication Union (ITU) เป็นผู้กำหนด ในโครงข่าย ATM สามารถรองรับการบริการได้หลายแบบและรองรับความต้องการคุณภาพการบริการของการบริการแต่ละแบบ การบริการแต่ละแบบ

จะมีพารามิเตอร์ของคุณภาพการบริการ โดยพารามิเตอร์นี้จะถูกกำหนดอยู่ในทราฟฟิกของแหล่งกำเนิด พารามิเตอร์นี้จะถูกจัดการโดยแหล่งกำเนิดในโครงข่ายและค่าพารามิเตอร์กำหนดให้เป็นคุณภาพการบริการของโครงข่าย ค่าพารามิเตอร์ที่ใช้มีดังนี้ Maximum Cell Transfer Delay (max CTD), Peak to Peak Cell Delay Variation (peak-to-peak CDV), และ Cell Loss Ratio (CLR) แต่ละการบริการ โครงข่ายจะจัดการเกี่ยวกับค่าพารามิเตอร์จำพวกนี้ ส่วนข้อกำหนดคุณลักษณะของทราฟฟิกทางด้านแหล่งกำเนิดจะประกอบด้วย Peak Cell Rate (PCR), Sustained Cell Rate (SCR) และ Maximum Burst Size (MBS)

*การบริการแบบ Constant Bit Rate (CBR)* เป็นการบริการที่รับประกันอัตราการส่งที่คงที่ซึ่งระบุโดย Peak Cell Rates (PCR) โครงข่ายจะรับประกันทุกๆเซลล์ที่แหล่งกำเนิดส่งออกมาที่อัตรา PCR และถูกส่งเข้าไปในโครงข่ายที่อัตรา PCR นี้ตลอด

*การบริการแบบ real time Variable Bit Rate(rt-VBR)* ซึ่งจะมีพารามิเตอร์ PCR, SCR และ MBS เป็นพารามิเตอร์ที่ควบคุมทราฟฟิกที่เป็น burst โดยโครงข่ายพยายามส่งเซลล์ให้อยู่ในขอบเขตของค่า cell delay (max CTD) และค่า delay variation (peak-to-peak CDV)

*การบริการแบบ non real time VBR* แหล่งกำเนิดจะถูกกำหนดโดยค่า PCR, SCR และ MBS แต่โครงข่ายจะไม่กำหนดให้ส่งข้อมูลอยู่ภายในขอบเขตค่า CTD และ CDV เหมือนกับการบริการแบบ rt-VBR

*การบริการแบบ Available Bit Rate (ABR)* เป็นการบริการที่โครงข่ายมีการรับประกันค่า Minimum Cell Rate (MCR) โดย ABR connection จะใช้การควบคุมความคับคั่งแบบ rate-based ซึ่งโครงข่ายพยายามที่จะทำให้มีค่า CLR ต่ำโดยให้มีการเปลี่ยนแปลง Allowed Cell Rate (ACR) ที่แหล่งกำเนิด

*การบริการแบบ Unspecified Bit Rate (UBR)* เป็นการบริการที่ไม่มีการรับประกันการบริการใดๆ แหล่งกำเนิดของ UBR จะไม่ต้องปรับอัตราการส่งข้อมูลให้อยู่ในค่าพารามิเตอร์ต่างๆ (CTV, CVD, CLR) สวิตช์จะไม่มีการควบคุมความคับคั่งใดๆให้กับแต่ละ VC ของการบริการแบบ UBR เมื่อบัฟเฟอร์เต็ม สวิตช์ก็จะทำการทิ้งเซลล์ออกไป เทคนิคการจัดสรรบัฟเฟอร์จึงมีความจำเป็นและมีการปรับปรุงเพิ่มขึ้นจากการบริการแบบ UBR เดิม ซึ่งเทคนิคการจัดสรรบัฟเฟอร์ที่ปรับปรุงเพิ่มขึ้นมาจะกล่าวถึงในบทที่ 3

การบริการแบบ ABRจะมีการควบคุมความคับคั่งโดยใช้การควบคุมความคับคั่งแบบ rate-based และ credit-based ทำให้มีการควบคุมการส่งข้อมูลที่แหล่งกำเนิด ซึ่งจะทำให้มีการสูญเสียต่ำ ฉะนั้นการบริการแบบ ABR จึงคาดหมายว่าจะเป็นการบริการที่ดี แต่การบริการนี้ก็มีข้อเสียคือมีความยุ่งยากซับซ้อนจึงทำให้มีต้นทุนสูงในการสร้าง การบริการแบบ UBR เป็นการบริการที่ไม่มีการควบคุมความคับคั่งแต่ก็เป็นการบริการที่ไม่มีความซับซ้อนจึงมีต้นทุนต่ำ และคาดหมายว่าการบริการแบบ UBR จะถูกนำไปใช้กับเทคโนโลยีโครงข่ายความเร็วสูงในอนาคต ฉะนั้นวิทยานิพนธ์นี้จึงได้ทำการศึกษาเกี่ยวกับการบริการแบบ UBR

## 2.2 การควบคุมความคับคั่งและการส่งข้อมูลของ TCP

การส่งข้อมูลในโครงข่ายเพื่อให้มีความน่าเชื่อถือ จึงได้นำโพรโตคอล TCP มาใช้งานกันอย่างกว้างขวาง โดยโพรโตคอล TCP ได้มีการใช้การควบคุมการส่งข้อมูลและการควบคุมความคับคั่งเป็นเครื่องมือในการทำให้โครงข่ายมีวิสัยสามารถและความน่าเชื่อถือสูง ในหัวข้อนี้จะกล่าวถึงการควบคุมการส่งข้อมูลและกลไกการควบคุมความคับคั่งของแบบต่างๆ

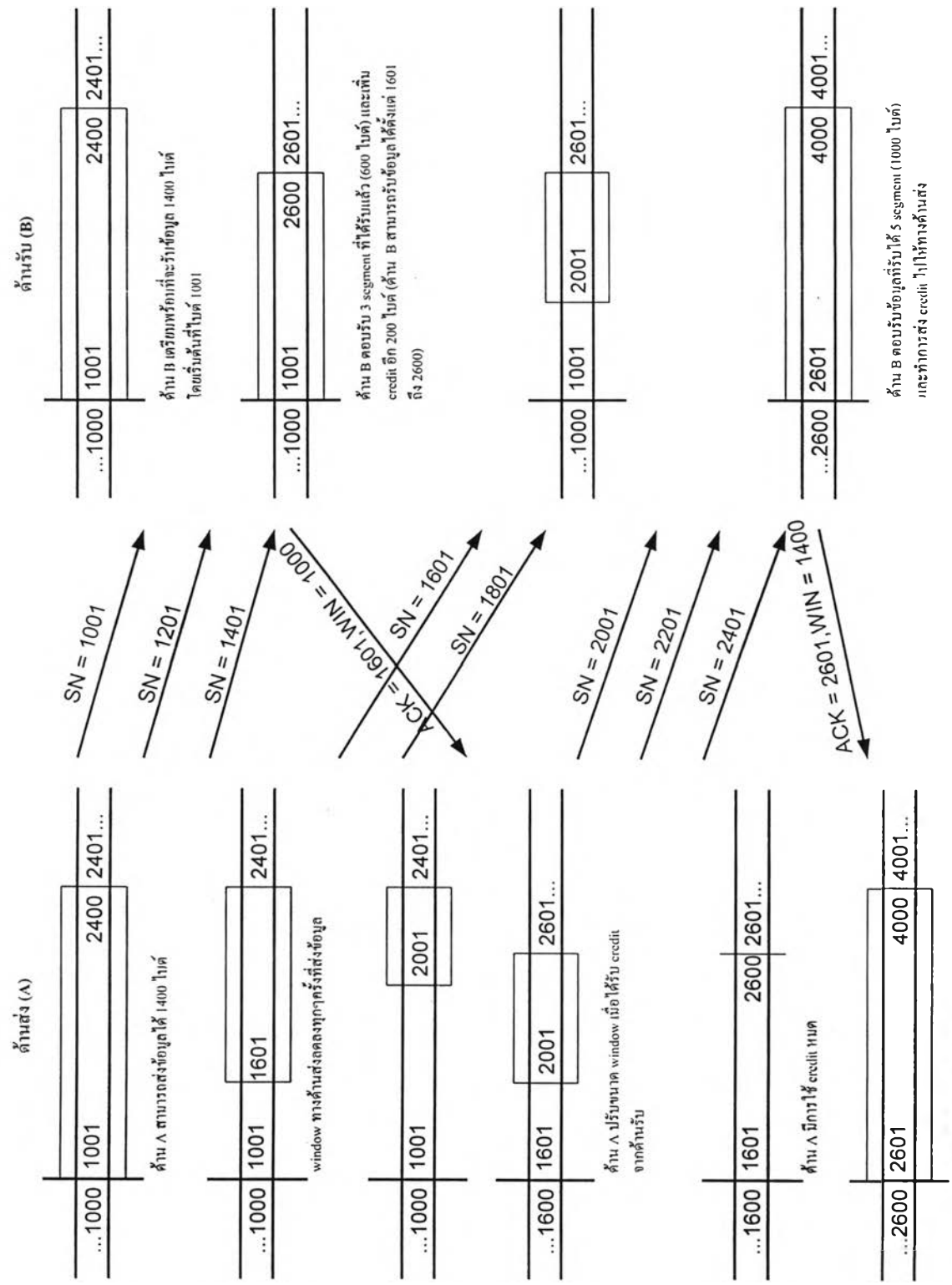
### 2.2.1 การควบคุมการส่งข้อมูลของ TCP (TCP Flow Control)

โพรโตคอลโดยทั่วไปจะต้องมีการควบคุมการส่งข้อมูล โพรโตคอล TCP จะใช้กลไก sliding window เป็นตัวควบคุมการส่งข้อมูลซึ่งแตกต่างจากโพรโตคอลอื่นๆ การควบคุมการส่งข้อมูลของ TCP ใช้เป็นแบบ credit allocation เมื่อต้นทาง TCP ส่งข้อมูล ด้นทางจะส่งไบต์แรกของ sequence number ลงในส่วนหัวของข้อมูล ต้นทาง TCP ได้รับ acknowledge (ACK) ซึ่งเป็นข่าวสารที่อยู่ในรูป (ACK = i, W IN= j) ทุกๆครั้งที่ข้อมูลมีลำดับข้อมูล i-1 ได้รับ ACK ข้อมูลลำดับต่อไปที่จะส่งคือ i และมีการอนุญาตให้ทางด้านส่งเพิ่มขนาด window (WIN) เป็น j ไบต์

ในรูปที่ 2.1 แสดงกลไกควบคุมการส่งข้อมูล โดยจะแสดงการส่งข้อมูลในทิศทางเดียวเท่านั้น และสมมติให้ส่งข้อมูล 200 ไบต์ในแต่ละ segment ขณะเริ่มต้นกระบวนการสร้างการเชื่อมต่อเสร็จเรียบร้อยก็จะมีการ synchronized กันระหว่างด้านรับและด้านส่ง ทางด้านส่ง (A) จะตกลง credit allocation ให้เป็น 1400 ไบต์และเริ่มส่งข้อมูลที่ลำดับ 1001 หลังจากส่งข้อมูลออกไป 3 segment ( 600 ไบต์) window จะมีการลดลงจนเหลือขนาด 800 ไบต์ (จำนวน 1601 จนถึง 2400) เมื่อปลายทาง (B) ได้รับข้อมูลทุกๆไบต์จนถึง 1601 และจะทำการเสนอ credit ( ขนาด window) ให้ต้นทางเป็น 1000 ไบต์ซึ่งหมายความว่าด้าน A สามารถส่งข้อมูลได้ตั้งแต่ไบต์ที่ 1601 จนถึง 2600 (5 segment) ในขณะที่ ACK ไปถึงด้าน A ทางด้านต้นทางได้มีการส่งข้อมูลเป็นจำนวน 2 segment เรียบร้อยแล้วคือตั้งแต่ 1601 จนถึง 2000 (สามารถส่งข้อมูลได้เนื่องมาจากอยู่ในจำนวน credit ที่ยอมให้ส่งได้ในขณะเริ่มต้น) ดังนั้นในขณะนี้ทางด้าน A มี credit เหลืออยู่ 400 ไบต์ (2 segment) เมื่อด้าน A ได้รับ ACK แล้วก็จะมีการปรับขนาด window ออก ปลายทั้งสองด้านของ window จะเพิ่มขึ้นหรือลดลงขึ้นอยู่กับจำนวนข้อมูลที่ส่งและ credit ที่ได้รับ

กลไกการจัดสรร credit เป็นกลไกที่มีความยืดหยุ่น เมื่อด้าน B ส่ง ACK ที่มีข่าวสารเป็น (ACK = i, WIN = j) และข้อมูลสุดท้ายที่ด้าน B ได้รับมีจำนวน i-1 ไบต์ ดังนั้น credit ที่เพิ่มขึ้นเป็นจำนวน k (k > j) เมื่อด้านรับไม่ได้รับข้อมูลแล้ว ด้าน B จะส่ง ACK (ACK = i, WIN = k) ส่วน ACK ของข้อมูล m ไบต์ (m < j) โดยที่ไม่มีการเพิ่ม credit ด้าน B จะส่ง ACK ที่มีข่าวสารเป็น (ACK = i + m, WIN = j - m) บางครั้งทางด้าน

รับไม่ต้องการส่ง ACK ทันทีทันใดเมื่อได้รับข้อมูลแล้ว แต่อาจจะรอและ ส่ง ACK ของจำนวน segment ที่เพิ่มขึ้น




รูปที่ 2.1 แสดงกลไกการควบคุมการส่งข้อมูลของ TCP

ทางด้านรับได้นำเอาวิธีเกี่ยวกับจำนวนข้อมูลที่อนุญาตให้ด้านส่งส่งได้มาใช้ โดยวิธีนี้จะยอมให้ส่ง segment ใหม่ได้ก็ต่อเมื่อมีที่ว่างในบัฟเฟอร์เท่านั้น ในรูปที่ 2.1 credit แรกที่ด้านรับส่งให้ด้านส่งหมายความว่า ด้าน B มีขนาดบัฟเฟอร์ที่ใช้งานได้ 1000 ไบต์ และ credit ที่สอง ด้าน B สามารถใช้งานได้ 1400 ไบต์ วิธีการควบคุมการส่งข้อมูลอาจจะเป็นการจำกัดค่าวิสัยสามารถและทำให้มีความล่าช้ามาก การเพิ่มวิสัยสามารถทำได้โดยทางด้านรับมีที่ว่างในบัฟเฟอร์ของ credit ที่อนุญาตให้เหมาะสม ถ้าทางด้านส่งมีความเร็วในการส่งข้อมูลมากกว่าทางด้านรับบางครั้งอาจทำให้มีการทิ้ง segment ออกไป จึงจำเป็นต้องมีการส่งข้อมูลใหม่ วิธีการควบคุมการไหลให้เหมาะสมจะทำให้โพรโตคอลนั้นมีความซับซ้อน

### 2.2.2. พื้นฐานการควบคุมความคับคั่งของ TCP

ความคับคั่งเกิดขึ้นเมื่อมีความต้องการใช้ทรัพยากรของโครงข่ายมากเกินกว่าที่จะมีทรัพยากรให้ใช้ได้ เมื่อภาระของผู้ใช้มีจำนวนมากเกินกว่าความจุที่โครงข่ายจะสามารถรองรับได้ โครงข่ายก็จะเกิดความคับคั่ง ในรูปที่ 2.2 [7] แสดงสมรรถนะของโครงข่ายเทียบกับภาระที่เพิ่มขึ้น จากรูปแสดงค่าวิสัยสามารถและค่าความล่าช้าเทียบกับภาระ ขณะเมื่อมีภาระต่ำ วิสัยสามารถจะเพิ่มขึ้นหรือลดลงตามภาระ แต่ค่าเวลาตอบสนอง (response time) หรือค่าความล่าช้า (delay) จะไม่เพิ่มขึ้นมาก ขณะเมื่อภาระมากเท่ากับความจุของโครงข่าย ภาระเพิ่มขึ้นจะทำให้ค่าความล่าช้าเพิ่มขึ้นตามไปด้วยแต่ค่าวิสัยสามารถจะไม่เพิ่มขึ้น การเพิ่มขึ้นของความล่าช้าเกิดเนื่องมาจากมีการเข้าคิวในโครงข่าย จุดที่ภาระมากเท่ากับความจุของโครงข่ายในรูป 2.2 เรียกว่าจุด "knee" เมื่อภาระมากเกินไปจะเป็นผลทำให้ค่าวิสัยสามารถลดลงและค่าความล่าช้าเพิ่มขึ้นอย่างมาก เพราะเนื่องมาจากคิวเกิดการล้นจึงทำให้เกิดการส่งแพ็กเกจใหม่และมีการใช้งานโครงข่ายลดลง จุดนี้ที่ทำให้ค่าวิสัยสามารถลดลงเรียกว่าจุด "cliff" โดยวัตถุประสงค์ของการหลีกเลี่ยงความคับคั่ง (congestion avoidance) คือรักษาการทำงานของโครงข่ายให้อยู่ที่จุด knee ขณะที่วัตถุประสงค์ของการควบคุมความคับคั่ง (congestion control) คือรักษาจุดการทำงานของโครงข่ายให้อยู่ทางด้านซ้ายมือของจุด cliff [7] โดย Jain. [8] ได้อธิบายถึง 2 องค์ประกอบของการหลีกเลี่ยงความคับคั่ง ดังนี้

1. Network policies มีหลายๆองค์ประกอบของโครงข่ายที่สามารถวัดระดับภาระ Jain [8] แสดงให้เห็นว่าความยาวคิวเฉลี่ยเป็นสิ่งที่แสดงให้เห็นขนาดภาระของโครงข่าย จากนั้น Jain ยังได้เสนอให้ใช้บิทในส่วนหัวของแพ็กเกจเป็นตัวชี้ว่าเกิดความคับคั่งเมื่อถูกเซตที่ routers แหล่งกำเนิดจะใช้บิตที่ป้อนกลับมานี้มาทำการเพิ่มหรือลดอัตราการส่ง (หรือขนาด window) TCP ได้ใช้การเพิ่มขึ้นของค่าความล่าช้า [9] และ timeout [10] เป็นเครื่องแสดงว่าขณะนี้เกิดความคับคั่ง การป้อนกลับให้ผู้ใช้งานไม่ว่าจะเป็นต้นทางหรือปลายทางจากโครงข่ายเพื่อเป็นการปรับตัวหรืออัตราการส่งให้เหมาะสมต่อโครงข่ายขณะนั้นๆ

2. End system policies วัตถุประสงค์ของการศึกษาที่ end system เพื่อให้การทำงานอยู่ในสถานะอยู่ตัว (steady state) โครงข่ายที่อยู่ในสถานะอยู่ตัวจะมีจำนวนแพ็กเกจที่อยู่ในโครงข่ายเท่ากับความจุของโครง

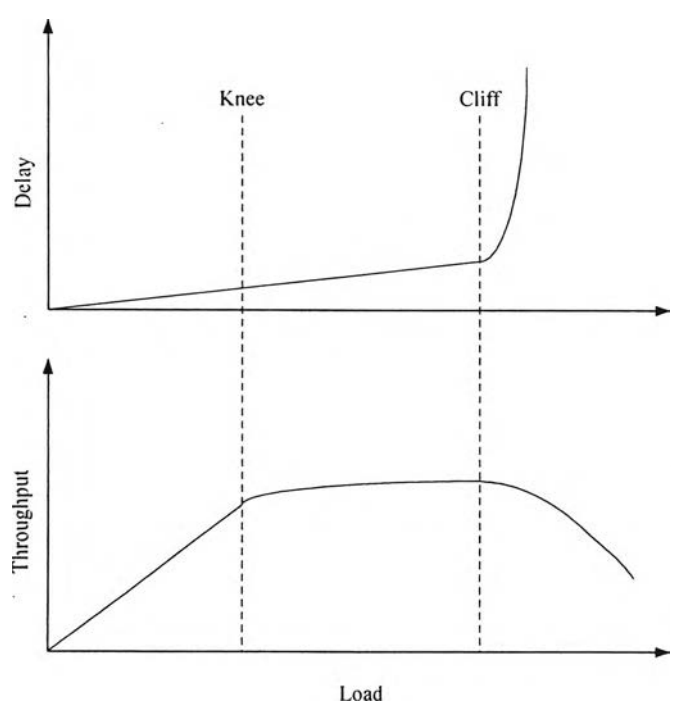
ข่าย (network capacity) ความจุของโครงข่ายกำหนดให้เป็นผลคูณของ แบนด์วิดท์กับความล่าช้า (bandwidth-delay product) [2] ดังนี้

$$Capacity\,(bits) = bandwidth\,(bit/\sec) \times round\ trip\ time\,(\sec)$$

สำหรับการหลีกเลี่ยงความคับคั่งสถานะอยู่ตัวจะอยู่ที่จุด knee ของกราฟในรูปที่ 2.2 การหลีกเลี่ยงความคับคั่งที่ระบบปลายทางจะมี 2 วิธี คือ

- ทำอย่างไรจึงจะทำให้โครงข่ายอยู่ในสถานะอยู่ตัวอย่างรวดเร็ว
- ทำอย่างไรจึงจะยังรักษาให้โครงข่ายอยู่ในสถานะอยู่ตัวถึงแม้ว่าจะมีการเปลี่ยนสภาวะของโครงข่าย

ระบบปลายทางจะรู้ว่าสภาพของโครงข่ายมีการเปลี่ยนแปลงจากกลไกการป้อนกลับ เมื่อปลายทางได้รับการป้อนกลับจากโครงข่าย ระบบปลายทางก็จะทำการตัดสินใจดังนี้ ระบบปลายทางจะเปลี่ยนอัตราการส่งข้อมูลเพิ่มขึ้นหรือลดลง และเปลี่ยนอัตราการส่งข้อมูลบ่อยขนาดไหน ขณะเกิดความคับคั่งจะมีแพ็กเกจสูญเสียและเราจะพยายามให้โครงข่ายอยู่ในสถานะอยู่ตัวอย่างรวดเร็วเมื่อเกิดความคับคั่ง โดยระบบปลายทางจะทำการส่งแพ็กเกจที่สูญเสียนั้นใหม่




รูปที่ 2.2 แสดงการเปรียบเทียบ ภาระ (Load) กับ ความล่าช้า (Delay) และ วิสัยสามารถ (Throughput)

TCP จะใช้ acknowledgment เป็นพื้นฐานสำหรับกลไกการควบคุมความคับคั่งและการไหลของข้อมูล การควบคุมการไหลข้อมูลมีเพื่อทำให้ด้านส่งไม่ทำให้เกิดการล้น (overflow) ของข้อมูลที่ทางด้านรับ ส่วนการควบคุมความคับคั่งมีเพื่อทำให้ด้านส่งไม่ทำให้เกิดการล้นของข้อมูลในโครงข่าย การควบคุมการไหลจะถูกบังคับโดย receiver window (RCVWND) ซึ่งจะเป็นตัวบอกขนาดของบัฟเฟอร์ที่ใช้งานได้ที่ด้านรับของ TCP การควบคุมความคับคั่งจะเป็นไปตาม congestion window (CWND) ซึ่งจะเป็นตัวบอกความจุของโครงข่ายโดยประมาณ

ณ เวลาใดๆ TCP จะเลือกค่าที่น้อยที่สุดระหว่าง RCVWND และ CWND ในวิทยานิพนธ์นี้เราจะเน้นในโครงข่ายที่เป็นคอขวด (bottlenecks) โดยสมมติให้ RCVWND > CWND เสมอ การทำงานของ TCP ในสถานะอยู่ตัวเกิดขึ้นเมื่อ CWND เท่ากับความสามารถที่จะรับข้อมูลได้ของโครงข่าย Jacobson และ Karels[11] ได้เสนอพื้นฐานการควบคุมความคับคั่งของ TCP คือ แพ็กเกจจะไม่เข้าไปในโครงข่ายจนกระทั่งมีแพ็กเกจออกจากโครงข่าย ซึ่งเรียกหลักการนี้ว่า "packet conservation principle" [11]

### 2.2.3 Slow Start และ Congestion Avoidance

พื้นฐานการควบคุมความคับคั่งของ TCP [11] จะมีระยะ "Slow Start" และ "Congestion Avoidance" (ซึ่งเรียกว่า vanilla TCP)โดยจะสอดคล้องกับระยะสถานะไม่อยู่ตัวและสถานะอยู่ตัวตามลำดับ วัตถุประสงค์ของระยะ slow start เพื่อทำให้โครงข่ายอยู่ในสถานะอยู่ตัวอย่างรวดเร็ว ส่วนระยะ congestion avoidance คือระยะที่โครงข่ายอยู่ในสถานะอยู่ตัว ตัวแปร SSTHRESH เป็นตัวแปรที่บอกความแตกต่างระหว่าง 2 ระยะ SSTHRESH คือค่าโดยประมาณของความจุโครงข่ายขณะอยู่ในสถานะอิ่มตัว ในสภาวะเริ่มต้นค่า SSTHRESH ถูกกำหนดให้มีค่าคงที่ค่าหนึ่งคือ 65535 ไบต์

แหล่งกำเนิดจะเริ่มต้นส่งข้อมูลในระยะ slow start เพียง 1 segment (CWND = 1 TCP segment) เมื่อแหล่งกำเนิดได้รับ acknowledgment (ACK) สำหรับ segment ใหม่ที่จะต้องส่ง แหล่งกำเนิดจะเพิ่ม CWND อีก 1 segment โดยขณะนี้แหล่งกำเนิดสามารถส่งข้อมูลได้ 2 segment (CWND = 2) เข้าไปในโครงข่าย ในระหว่างที่อยู่ในระยะ congestion avoidance แหล่งกำเนิดจะเพิ่มจำนวน CWND ทีละ 1 segment ทุกครั้งที่แหล่งกำเนิดได้รับ ACK ส่วนระหว่างที่อยู่ในระยะ slow start ค่า CWND จะเพิ่มขึ้นทีละ 2 เท่าทุกๆ round trip time โดยค่า Round Trip Time (RTT) ของ connection คือเวลาตั้งแต่เริ่มส่งข้อมูลจนถึงต้นทางได้รับ ACK ของ segment นั้น เมื่อค่า CWND เพิ่มขึ้นจนค่า CWND เท่ากับค่า SSTHRESH ระยะ slow start ก็จะเปลี่ยนไปเป็นระยะ congestion avoidance

ในระหว่างระยะ congestion avoidance แหล่งกำเนิดจะเพิ่ม CWND ไปเป็น 1/CWND ทุกๆครั้งที่ได้รับ ACK ในแต่ละ round trip time ในระหว่างระยะ congestion avoidance CWND จะเพิ่มขึ้นทีละ 1

segment ในระยะ slow start กราฟของ CWND จะมีลักษณะการเพิ่มขึ้นแบบ exponential ส่วนระยะ congestion avoidance กราฟของ CWND มีลักษณะการเพิ่มขึ้นเป็นแบบ linear ทุกๆ round trip time

ถ้า TCP connection มีการสูญเสียแพ็กเกจ ปลายทางจะส่ง duplicate ACK ของแพ็กเกจที่หายไป ต้นทางจะทำการ retransmission เนื่องมาจากการหมดเวลา (timeout) เมื่อได้รับแพ็กเกจสุดท้ายที่ไม่ได้รับ ACK โดยค่าเวลา timeout จะถูกเคลียร์ทุกๆครั้งที่ได้รับ ACK ของแพ็กเกจใหม่ ต้นทางรู้ว่าเกิดความคับคั่งเมื่อ retransmission timeout เกิดการ trigger ต้นทางจะไม่ส่งแพ็กเกจใหม่เมื่อได้รับ duplicate ACK โดยต้นทางรอจนกว่าจะได้รับ ACK ใหม่หรือเกิดการ timeout

เมื่อเกิด timeout ขึ้น ต้นทางจะว่างหรือไม่ส่งข้อมูลประมาณ 1 round trip time แพ็กเกจที่มีการ timeout สมมติว่ามีการสูญเสียแพ็กเกจเนื่องมาจากความคับคั่ง ช่วงเวลาที่ว่างไม่ส่งข้อมูลจนถึงเวลา timeout จะต้องมีช่วงเวลาที่เพียงพอกับความคับคั่งถูกเคลียร์ออกไป เวลา timeout ไม่ได้เป็นเพียงแค่เครื่องบ่งบอกว่าเกิดความคับคั่งเท่านั้นเมื่อเกิดการ trigger ยังจะเป็นเครื่องบอกให้แหล่งกำเนิดส่งข้อมูลใหม่

เมื่อเกิดการ timeout แล้ว แหล่งกำเนิดตั้งค่า SSTHRESH ให้มีค่าเป็นครึ่งหนึ่งของ CWND โดยให้ละเอียดแล้วค่า SSTHRESH จะตั้งค่าให้เป็น

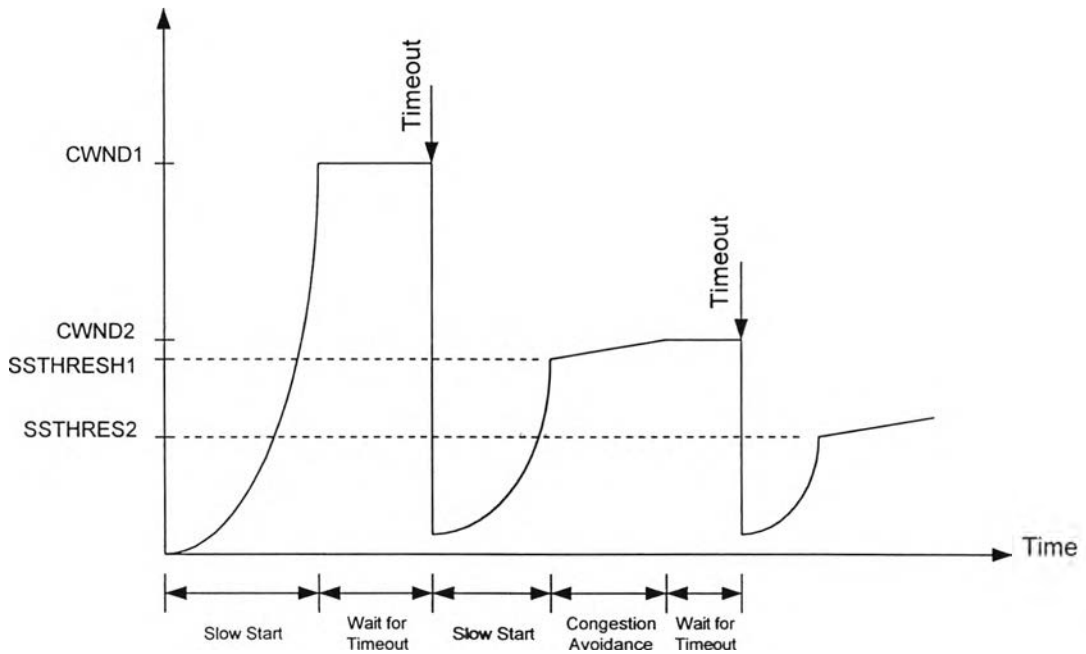
$$SSTHRESH = \max\left\{2, \min\left\{\frac{CWND}{2}, RCVWND\right\}\right\}$$

เมื่อ CWND < SSTHRESH และแหล่งกำเนิดเข้าสู่ระยะ slow start ดังนั้นแหล่งกำเนิดเริ่มการส่งข้อมูลใหม่ (retransmit) และเพิ่ม CWND ทีละหนึ่งทุกๆครั้งที่แพ็กเกจใหม่ได้รับ ACK

ถ้ามีการสูญเสียหนึ่งแพ็กเกจและถ้าบัฟเฟอร์ทางด้านรับไม่พอดังนั้นทางด้านส่งจะได้รับ cumulative acknowledgment และกู้ข้อมูลเดิมที่สูญเสียกลับมาจากความคับคั่ง ด้านส่งพยายามส่งข้อมูลใหม่ทุกแพ็กเกจโดยเริ่มต้นจากแพ็กเกจที่สูญเสีย ไม่ว่ากรณีใดๆ CWND จะเพิ่มขึ้นที่หนึ่งแพ็กเกจทุกๆครั้งที่ได้รับ ACK ถึงแม้ว่า CWND อาจจะเพิ่มหลังจาก advertised receiver window (RCVWND) แต่ window ของแหล่งกำเนิดก็ยังถูกจำกัดโดยค่าที่น้อยที่สุดของ window ทั้งสอง การเปลี่ยนแปลงขนาด CWND เทียบกับเวลาแสดงในรูปที่ 2.3

โดยทั่วๆไป TCP จะใช้ค่า retransmission timeout เป็น 500 ms แหล่งกำเนิด TCP จะประมาณค่า round trip time (RTT) ของ connection โดยการวัดเวลา (จำนวนเครื่องหมายของ timer) ระหว่างการส่งแพ็กเกจจนถึงได้รับ ACK ของแพ็กเกจนั้นกลับมา retransmission timeout ถูกคำนวณเป็นฟังก์ชั่นโดยประมาณของค่าเฉลี่ยและค่า mean-deviation ของ RTT [11] เมื่อมีการสูญเสียเนื่องมาจากความคับคั่ง เพราะ TCP timer มีค่าหยาบจึงทำให้มีการสูญเสียเวลาขณะที่ retransmission timeout รอการ trigger เมื่อก่อนแหล่ง

กำเนิดส่งข้อมูลออกไปเป็นจำนวนตามที่ window อนุญาตให้ส่งได้และแหล่งกำเนิดจะไม่ยอมส่งข้อมูลใหม่เมื่อได้รับ duplicate ACK เมื่อ retransmission timeout มีการ trigger เกิดขึ้น connection นั้นก็เข้าสู่ระยะ slow start ซึ่งเป็นผลให้ link อาจจะว่างเป็นระยะเวลานานได้และทำให้มีการใช้งานต่ำ




รูปที่2.3 แสดงกราฟ Congestion Window (CWND) ขณะ Slow Start และ Congestion Avoidance

พฤติกรรมการควบคุมความคับคั่งของ TCP ที่กล่าวมาแล้วนั้นเรียกว่า "Vanilla TCP" หลายๆครั้งได้มีงานวิจัยที่เสนอการแก้ไข Vanilla TCP [2] บางครั้งก็ได้มีการนำงานวิจัยกำหนดเป็นมาตรฐานโดย IETF (Internet Engineering Task Force) ในวิทยานิพนธ์นี้จะสนใจในพื้นฐานของ TCP ที่ยังไม่ได้มีการแก้ไขเป็นหลัก ในหัวข้อต่อไปเป็นการแก้ไข มาตรฐาน TCP เพิ่มขึ้นเรียกว่า TCP Reno โดยมีการเพิ่มกลไก Fast retransmission และ recovery

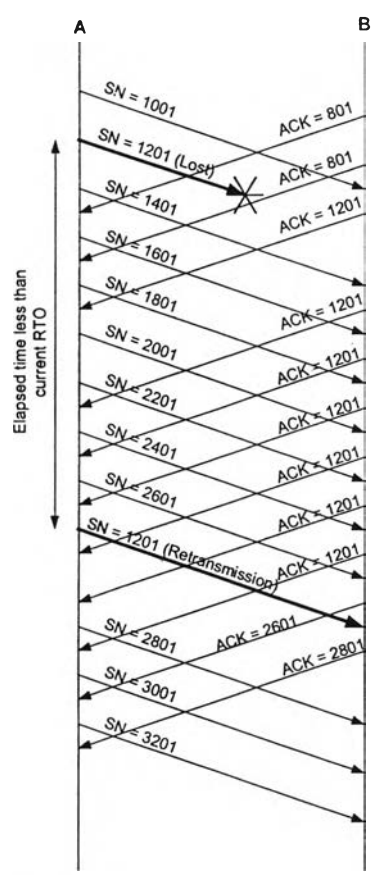
### 2.2.4 Fast Retransmission และ Recovery (TCP Reno)

TCP Reno มีการเพิ่มเติมในส่วนการควบคุมความคับคั่งซึ่งเรียกว่า Fast Retransmit and Recovery (FRR) Fast Retransmit และ Fast Recovery ถูกออกแบบมาเพื่อปรับปรุงสมรรถนะ TCP เมื่อมีการสูญเสียเพียงหนึ่งแพ็กเกจ ปัจจุบันค่า retransmission timeout ของ TCP จะใช้ timer เป็น 500 ms เมื่อเกิดความคับคั่ง TCP connection จะสูญเสียเวลาขณะรอ timeout ในรูปที่ 2.3 แกนนอนแสดงเวลาที่สูญเสียขณะรอให้เกิด timeout ในช่วงเวลานี้ TCP จะไม่ทำการส่งแพ็กเกจใหม่หรือส่งแพ็กเกจที่สูญเสียใหม่ อย่างไรก็ตามเมื่อเกิด

timeout ค่า CWND จะตั้งให้เป็น 1 segment จะเห็นว่า connection มีการใช้โครงข่ายอย่างมีประสิทธิภาพก็ต่อเมื่อผ่านไปหลายๆ round trip time TCP Reno มีอัลกอริทึม Fast Retransmit และ Recovery เพื่อทำให้ connection มีการนำแพ็กเกจที่สูญเสียกลับมาอย่างรวดเร็ว

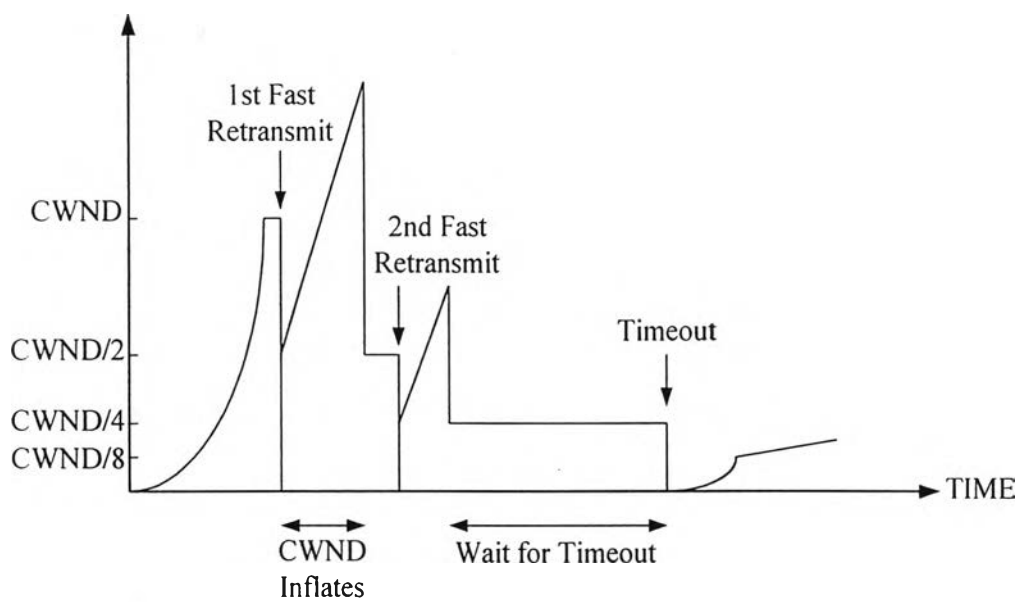
เมื่อปลายทางได้รับแพ็กเกจที่เสีย ปลายทางก็จะทำการส่ง duplicate ACK ทันทีให้กับด้านส่ง เมื่อด้านส่งได้รับ 3 duplicate ACK มันก็จะรู้ว่าแพ็กเกจไหนที่เสียจาก ACK แล้วก็จะทำการส่งแพ็กเกจที่เสียใหม่ทันทีโดยไม่ต้องรอ timeout ก่อนซึ่งจะเรียกกลไกนี้ว่า "Fast Retransmit" จากนั้นก็เริ่มระยะ congestion avoidance โดยไม่มีระยะslow start ก่อนจึงเรียกกลไกนี้ว่า "Fast Recovery" ดังนั้นต้นทางจะรู้ว่าเกิดความคับคั่งก็ต่อเมื่อได้รับ duplicate ACK 3 ครั้ง

ด้านส่งจะทำการลด CWND ลงครึ่งหนึ่งบวกด้วย 3 segment (ที่บวกด้วย 3 segment เพราะเป็นจำนวน segment ที่ออกจากโครงข่ายแล้วปลายทางรับได้) และเก็บค่านี้ไว้ใน SSTHRESH แต่ละครั้งที่ได้รับ 3 duplicate ACK ส่วนทางด้านส่งจะขยาย CWND ทีละหนึ่งและพยายามที่จะส่งแพ็กเกจใหม่ทุกครั้งที่ได้รับ ACK ด้านส่งจะรอจนถึงครึ่งหนึ่งของ round trip time ก่อนที่จะส่งข้อมูลหนึ่งแพ็กเกจใหม่สำหรับแต่ละครั้งที่ได้รับ duplicate ACK หลังจากนี้ ผลก็คือในระหว่างเกิดความคับคั่ง ด้านส่งยังคงรักษาการส่งข้อมูลเข้าไปในโครงข่ายไว้ที่ครึ่งหนึ่งของความจุขณะทำการ fast retransmit




รูปที่ 2.4 แสดงกลไก Fast retransmit

เมื่อเกิดความคับคั่งก็จะมี timeout เกิดขึ้น CWND ก็จะถูกตั้งไว้ที่ 1 segment และเริ่มดำเนินการระยะ slow start เมื่อ CWND เพิ่มขึ้นจนถึง SSTHRESH ดังนั้นก็จะเข้าสู่ระยะ congestion avoidance แต่ถ้าต้นทางได้รับ duplicate ACK อย่างน้อย 3 ครั้งเริ่มการส่งข้อมูลใหม่โดยเริ่มดำเนินการระยะ congestion avoidance โดยไม่มีระยะ slow start ซึ่งจะเรียกกลไกที่กล่าวมานี้ว่า "Fast Recovery" ในรูปที่ 2.4 แสดงกระบวนการ Fast Retransmit โดยด้านส่งจะส่งจำนวนข้อมูล 200 ไบต์ต่อหนึ่ง segment ซึ่งจากรูป segment 1201 มีการสูญหาย ทางด้านต้นทาง (A) จะยังคงส่งข้อมูลอย่างต่อเนื่องจนกว่าส่งข้อมูลครบตาม CWND ทางด้านรับ (B) ได้รับข้อมูล segment 1001 (1001 จนถึง 1200 ไบต์) ก็จะส่ง ACK ของแพ็กเกจต่อไปคือ segment 1201 จากนั้นทางปลายทางได้รับข้อมูล segment 1401 (1401 จนถึง 1600 ไบต์) ด้านรับยังคงส่ง ACK ของ segment 1201 ซ้ำทุกครั้งที่ได้รับ segment ใหม่เข้ามาจนกระทั่งได้รับข้อมูล segment 1201จึงจะหยุดส่ง ในรูปที่ 2.5 แสดงกราฟ CWND ของกลไก Fast retransmit และ recovery




รูปที่ 2.5 แสดง Congestion window ของกลไก Fast retransmit and recovery