บทที่ 4

การประกอบโนดเป็นเครือข่าย

4.1 การจำลองการทำงานแบบขนาน

การนำโนดมาประกอบเป็นเครือข่ายจะมีงานที่ต้องทำขนานกันทั้งอุปกรณ์ภายในโนดและแต่ละโนดในเครือข่าย การจำลองการทำงานของระบบเครือข่ายจำเป็นต้องใช้ข้อกำหนดและตัวแปร (variable) ต่าง ๆ ที่เกี่ยวข้องในการทำงานของแต่ละโนด เช่น ขนาดของบัฟเฟอร์ของโนดหรือจำนวนของช่องสื่อสาร เป็นต้น หรือเกี่ยวข้องกับเหตุการณ์สำคัญต่าง ๆ เช่น การส่งแพ็กเก็ตจากโฮสมายังบางโนดในเครือข่าย การรับส่งข้อมูลระหว่างโนดกับโนดในเครือข่าย หรือเกิดความผิดพลาดของอุปกรณ์ต่าง ๆ ภายในโนด เป็นต้น

ในบทที่ 3 เราได้ออกแบบส่วนประกอบแบบจำลองของโนด (ดูรูปที่ 3.11) ซึ่งแบ่งออกเป็น 3 หน่วยคือ 1) หน่วยเชื่อมต่อกับผู้ใช้ 2) หน่วยเชื่อมต่อกับเครือข่าย และ 3) หน่วยควบคุมโนด ตามรายละเอียดดังกล่าวไปแล้วในหัวข้อ 3.2 โดยจำลองด้วยซอฟต์แวร์ แบบจำลองทางซอฟต์แวร์นี้จะแทนสถานะการทำงานของแต่ละหน่วย รายละเอียดได้กล่าวไปแล้วในหัวข้อ 3.3

4.1.1 การจำลองเหตุการณ์แบบขั้น ๆ

ในหัวข้อนี้จะอธิบายถึงวิธีการในการจำลองการทำงานของโนด ในการจำลองจะเกิดการเปลี่ยนสถานะจากสถานะหนึ่งไปยังอีกสถานะหนึ่ง เราเรียกว่าเหตุการณ์ (Event) ดังนั้นการใช้เทคนิคการจำลองเหตุการณ์แบบขั้น ๆ (discrete event simulation) จะมีความเหมาะสมเนื่องจากการจำลองเหตุการณ์แบบขั้น ๆ จะพิจารณาระบบที่จะจำลองนั้นจากการเปลี่ยนสถานะ ณ เวลาที่เป็นช่วงหรือเป็นขั้น และเหตุการณ์ที่ทำให้เกิดการเปลี่ยนสถานะ

ในการจำลองการทำงานของเครือข่ายที่นำมาประกอบกันนั้น จำเป็นที่จะต้องมีการจำลองแบบขนาน (Parallel discrete event simulation) กระบวนการจำลองแบบขนานนั้นถูกกำหนดในลักษณะการรวบรวมลำดับของสถานะและเหตุการณ์ต่าง ๆ ทั้งหมดของระบบที่แสดงถึงพฤติกรรมการกทำงานของระบบ สาเหตุที่เรียกว่าเป็นการจำลองแบบขนานนั้นเพราะว่ากระบวนการทำงานทั้งหมด มักจะประกอบด้วยกระบวนการทำงานย่อยในแต่ละโนดที่มีการทำงานแบบขนานกันแต่ละกระบวนการทำงานย่อมจะมีการทำงานที่เป็นอิสระของตัวเอง และประกอบด้วยสถานะและเหตุการณ์ต่าง ๆ

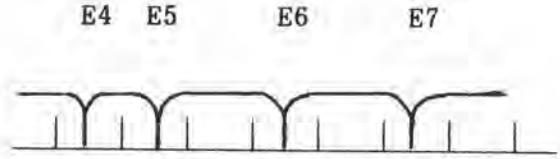
สถานะของระบบในการจำลองแบบขนานจะประกอบด้วยสถานะแบบช่วง (discrete state) ของแต่ละกระบวนการทำงานย่อยในระบบขนาน รวมทั้งค่าปัจจุบันของแต่ละตัวแปรสถานะ (state variable) สถานะแบบช่วงจะแสดงการทำงานใน 2 ลักษณะ คือ ภาวะทำงานตามหน้าที่ (mission function) ของหน่วยหรืออุปกรณ์นั้น และภาวะรอ (wait) เงื่อนไขตามที่กำหนดไว้ ในระดับตรรกะ (logic condition) เป็นที่พอใจหรือตรงเงื่อนไขก่อนที่จะเปลี่ยนสถานะไปทำงานอย่างอื่น

ในแต่ละสถานะการทำงานจะมีการบันทึกเวลาที่ถูกใช้ไปสำหรับสถานะนั้น การเปลี่ยนสถานะจะเกิดขึ้นเมื่อสถานะนั้นทำงานครบตามเวลาที่กำหนดไว้หรือ ได้รับเงื่อนไขที่ต้องการ การบันทึกเวลาที่ถูกใช้ไปนี้จะเป็นตัวชี้ว่าการทำงานของระบบหรือโนดเป็นอย่างไรเมื่อสถานะนั้นเกิดขึ้นจริงในระบบ เช่น มีการเก็บแพ็กเก็ตอยู่ในคิวนานเท่าไหร่ เวลาที่ใช้ในการส่งแพ็กเก็ตใช้ไปเท่าไรแล้ว เป็นต้น

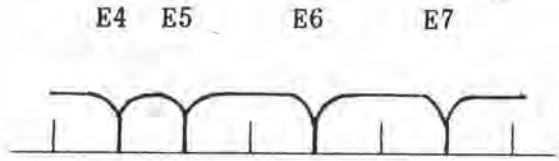
ดังนั้นการจำลองแบบขนานจะมีการแบ่งเวลาเป็นช่วงสำหรับแต่ละเหตุการณ์ วิธีในการเลือกใช้เวลานั้นมีอยู่ 2 แบบ คือ (Watson, 1989)

- กำหนดเวลาอิสระตามเหตุการณ์ เป็นการกำหนดตามเหตุการณ์จริง บางทีเรียก วิธีนี้ว่า next event simulation ดังรูปที่ 4.1 (ก)

- กำหนดเวลาที่เพิ่มขึ้นแบบคงที่ กล่าวคือ เป็นการแบ่งเวลาเป็นช่วงหรือเป็นคาบ เรียกว่า quantumtime (1 ช่วงเวลาเท่ากับ 1 quantumtime) โดยกำหนดว่าแต่ละเหตุการณ์จะใช้เวลาในการทำงานกี่ quantumtime เราเรียกวิธีนี้ว่า fixed time increment simulation ดังรูปที่ 4.1 (ข)



(ก) next event simulation



(ข) fixed-time increment simulation

รูปที่ 4.1 แสดงวิธีการกำหนดเวลาของเหตุการณ์

โดยทั่วไปวิธี fixed-time increment เป็นวิธีที่ง่ายและสะดวกมากกว่าวิธี next event วิธี next event นี้จะให้ความสะดวกสำหรับเหตุการณ์ที่ใช้เวลาในการเปลี่ยนสถานะเป็นช่วงเวลานานมาก หรือกล่าวอีกนัยหนึ่งว่า มีการเปลี่ยนแปลงของสถานะไม่บ่อยมาก แต่วิธี fixed-time increment จะให้ความสะดวกสำหรับเหตุการณ์ที่มีช่วงเวลาที่เป็นคาบ ซึ่งจุดต่างๆ ภายในช่วงเวลาที่อยู่ในคาบจะไม่มีเหตุการณ์เกิดขึ้น จะมีเหตุการณ์เกิดขึ้นในขณะตำแหน่งเริ่มต้นเวลาของช่วงที่เป็นคาบ ดังนั้นจะพบว่าวิธี fixed-time increment จะมีความเหมาะสมกว่าในการจำลองการทำงานของเครือข่าย แต่สิ่งสำคัญคือการกำหนดค่าของแต่ละช่วงเวลาว่ามีค่าน้อยหรือมากเท่าใดจึงจะเหมาะสม

4.1.2 ข้อสมมุติการกำหนดเวลาเพิ่มขึ้นแบบคงที่

ข้อสมมุติที่จะกล่าวในหัวข้อนี้จะสนับสนุนหลักการกำหนดเวลาที่เพิ่มขึ้นแบบคงที่ มีรายละเอียด ดังนี้ เมื่อมีสถานะการทำงานต่าง ๆ เป็น s_i และมีเวลาในการทำงานแต่ละสถานะเท่ากับ t_i (i=1 ถึง n) และกำหนดว่าค่าของเวลา quantum time คือ t_q ค่าของ t_q จะเหมาะสมได้ก็ต่อเมื่อสามารถแทนค่า t_i ให้อยู่ในเทอมของ $k_i * t_q$ โดยที่ k_i จะเป็นเลขจำนวนบวกเต็ม

$t_i = k_i t_q$ โดยที่ $i = 1,...n$

เมื่อ n คือสถานะการทำงาน k_i เป็นเลขจำนวนเต็มบวก กล่าวคือ ค่าของเวลาในการทำงานแต่ละสถานะ (t_i) จะอยู่ในเทอมเป็นจำนวนเท่าของค่าเวลา quantum time (t_q)

ตัวอย่างเช่น มีสถานะการทำงาน S_1, S_2, S_3 และมีเวลาในการทำงานแต่ละสถานะเท่ากับ 3, 6, 9 ตามลำดับ จะเห็นได้ว่าสามารถหาค่า $t_q = 1$ หรือ 3 เพราะสามารถแทนค่า $t_1 = 3t_q$, $t_2 = 6t_q$, และ $t_3 = 9t_q$ เมื่อ $t_q = 1$ และจะได้ $t_1 = t_q$, $t_2 = 2t_q$ และ $t_3 = 3t_q$ เมื่อ $t_q = 3$ ในทางปฏิบัติเราจะเลือกค่า $t_q = 3$ เพื่อให้ผลการทำงานในการจำลองถูกต้องและใช้เวลาในการจำลองน้อยกว่าการเลือก $t_q = 1$ ซึ่งให้ผลการทำงานที่เหมือนกัน กล่าวคือ อาจจะพิจารณาได้ว่าการหาค่า t_q นั้นหาได้จากการหาค่าหารร่วมมากของเวลาในการทำงานแต่ละสถานะ

โดยทั่วไปนั้นค่าของเวลาในการทำงานของแต่ละสถานะอาจจะไม่สามารถหาค่า t_q ที่เหมาะสม (ในกรณีที่บอกว่า $t_q = 1$ นั้นเรายังถือว่ายังไม่เป็นค่าที่ดีที่สุด) อาจจะต้องมีการประมาณค่าของสถานะใดสถานะหนึ่งให้อยู่ในกลุ่มใดกลุ่มหนึ่ง โดยกลุ่มที่กำหนดขึ้นมาใหม่นี้สามารถที่จะหาค่า t_q ได้อย่างเหมาะสม

ตัวอย่างเช่น มีสถานะการทำงาน 8 สถานะ มีเวลาในการทำงานในแต่ละสถานะเท่ากับ 8, 9, 10, 19, 20, 21, 28, 30 ตามลำดับ เราอาจจะต้องจัดให้

8, 9 และ 10 อยู่กลุ่มหนึ่ง มีค่าเท่ากับ 10

19, 20 และ 21 อยู่กลุ่มหนึ่ง มีค่าเท่ากับ 20

28 และ 30 อยู่กลุ่มหนึ่ง มีค่าเท่ากับ 30

จากนั้นหาค่า t_q ของทั้งสามกลุ่ม ได้เท่ากับ 10 ดังนั้นเวลาในการทำงานของแต่ละสถานะจะเท่ากับ t_q, t_q, $2t_q$, $2t_q$, $2t_q$, $3t_q$ และ $3t_q$ ตามลำดับ

4.2 การกำหนดเวลาของสถานะการทำงานของหน่วยต่าง ๆ ในโนด

ในแต่ละโนดจะมีอุปกรณ์ที่ทำงานอิสระต่อกันหรือทำงานแบบขนานกันได้แก่ UserRx, UserTx, NetwRx[From], NetwTx [To] และ NodeCon

4.2.1 การกำหนดเวลาพื้นฐานของอุปกรณ์

การกำหนดเวลาที่อุปกรณ์เหล่านี้ทำงานในสถานะต่าง ๆ มี 5 แบบ คือ

- เวลาแบบ vary สำหรับอุปกรณ์ที่ทำงานในสภาวะรอต่าง ๆ ได้แก่ Wait, WaitNodeCon, EndSending, NodeManager
- เวลาแบบ X rate สำหรับช่องสื่อสารในสถานะ RxPkt และ TxPkt
- เวลาแบบ NoOfMove สำหรับ NodeCon ที่ทำงาน MovePkt
- เวลาแบบ small สำหรับอุปกรณ์และสถานะต่อไปนี้ UserRx.RxPkt, UserTx.TxPkt, NodeCon.SendConPkt, NodeCon.CallRouting, NodeCon.PreRouting, NodeCon.ProcessConPkt
- เวลาแบบ very small สำหรับ NodeCon ซึ่งทำหน้าที่อื่นที่เหลือ ได้แก่ NodeCon.ChkError, NodeCon.ChkPktType, NodeCon.ChkAvailable, NodeCon.InsertPktInQ

4.2.2 ตัวอย่างการกำหนดเวลาจำลองของอุปกรณ์

เวลาจริง (Real time) จะถูกจำลองด้วยเวลาเป็นช่วง (discrete time) เช่น 1 วินาที = 1000 unit time ค่าเวลาของ very small ถือเป็น 1 quantum time ในที่นี้จะเรียกว่า unit time ดังนั้นสามารถกำหนดเวลาทำงานของสถานะต่าง ๆ เป็น unit time ได้ดังนี้

เวลา very small = 1 unit time

เวลา small = 5-20 unit time

เวลา X rate = 1000 unit time สำหรับช่องสื่อสารที่มีความเร็ว 1200 bps
500 unit time สำหรับช่องสื่อสารที่มีความเร็ว 2400 bps
250 unit time สำหรับช่องสื่อสารที่มีความเร็ว 4800 bps
125 unit time สำหรับช่องสื่อสารที่มีความเร็ว 9600 bps

เวลา NoOfMove = จำนวนแพ็คเก็ตที่ Move * 5 unit time

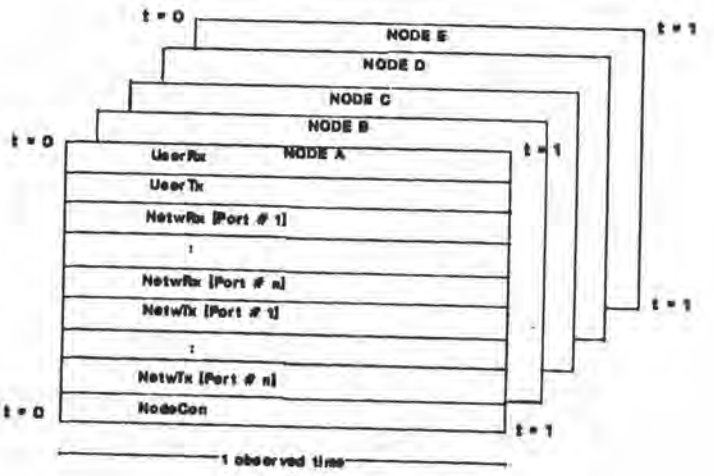
เวลา vary จะปรับตามการเปลี่ยนสถานะที่ปฏิบัติการได้จังหวะกัน (synchronize) กันระหว่างอุปกรณ์ที่ต้องทำงานประสานกัน

เมื่ออุปกรณ์มีสถานะการทำงานที่แน่นอนและเวลาที่ทำงานในแต่ละสถานะแน่ชัดสามารถคาดการณ์ได้เราก็สามารถบอกได้ว่าในช่วงเวลา 1 unit time อุปกรณ์ไหนอยู่ที่สถานะอะไร และเกิดเหตุการณ์อะไรบ้าง และทราบได้ว่าเหตุการณ์ถัดไป (next event) จะเป็นเหตุการณ์ใด สำหรับกรณีที่สถานะการทำงานของอุปกรณ์ต้องใช้เวลามากกว่า 1 unit time จะมีการบันทึกเวลาสะสมจนถึงเวลาที่งานเสร็จครบตามกำหนดจึงจะเปลี่ยนสถานะ ตัวแปรสถานะต่าง ๆ เช่น HavePkt, Error เป็นต้น จะเกิดขึ้นเมื่อทำงานสถานะที่เกี่ยวข้องแล้วเสร็จ

ในการจำลองเราจำเป็นต้องกำหนดว่าต้องการดูผลการจำลองในแต่ละช่วงเวลานานเท่าใด (กี่ unit time) เราเรียกช่วงเวลาดังกล่าวว่า เวลาในการสังเกต (observed time) ในการทดลองเรากำหนดให้ 1 observed time = 1 unit time ซึ่งถือว่าเป็นผลการทำงานแบบละเอียดมากที่สุด และจะเห็นเหตุการณ์และสถานะการทำงานต่างๆ ที่เกิดขึ้นกับอุปกรณ์ต่าง ๆ ได้อย่างละเอียด

4.2.3 การกำหนดเวลาสังเกตและการเปลี่ยนสถานะ

การกำหนดเวลาในการสังเกตจะส่งผลให้แต่ละอุปกรณ์ของโนดต่างๆ ได้รับการทำงานพร้อม ๆ กันเป็นเวลา 1 unit time ในการทำงานจริงจะเป็นการตั้งต้นการทำงานจากโนด A ไปถึงโนด E ในแต่ละโนดจะเริ่ม execute จาก UserRx ไปจนถึง NodeCon ดังรูปที่ 4.2



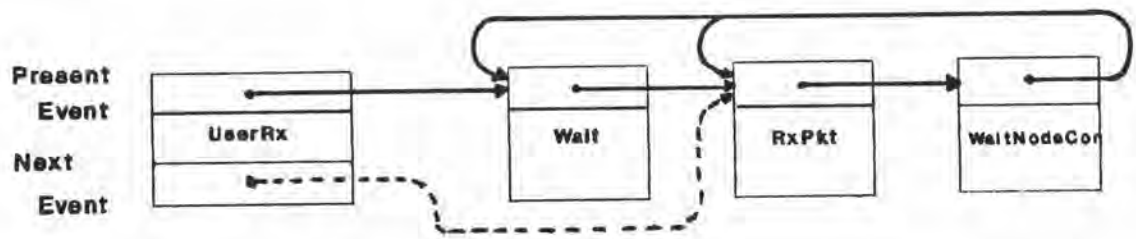
รูปที่ 4.2 การทำงานใน 1 observed time

4.3 การเชื่อมโยงเหตุการณ์ในการจำลองแบบขนาน

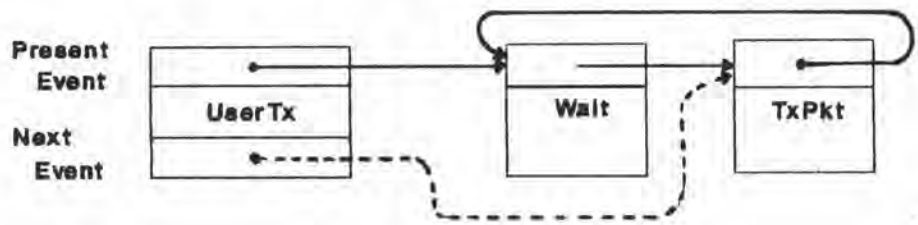
โครงสร้างของการเชื่อมโยงเหตุการณ์ (event list) จะต้องบรรจุเหตุการณ์ทั้งหมดที่เป็นไปได้ว่าจะเกิดขึ้น ซึ่งโครงสร้างที่ใช้จะเป็นโครงสร้างแบบเชื่อมโยง (list) โดยมีตัวเชื่อมโยงเพื่อชี้ไปยังสถานะถัดไป เรียกว่าตัวชี้สถานะถัดไป (next event) การเปลี่ยนแปลงสถานะจากสถานะหนึ่งไปยังอีกสถานะหนึ่งจะเกิดขึ้นได้ 2 ลักษณะ คือ แบบไม่มีเงื่อนไข และแบบมีเงื่อนไข

ถ้าเป็นการเปลี่ยนแปลงสถานะแบบมีเงื่อนไข จะมีตัวชี้สถานะถัดไปหลายตัวขึ้นอยู่กับเงื่อนไขที่เกิดขึ้น การทำงานของสถานะและเงื่อนไขในการเปลี่ยนสถานะได้กล่าวแล้วข้างต้น ในหัวข้อที่ 3.3

จากรูปที่ 4.3 แสดงการเชื่อมโยงเหตุการณ์ของหน่วยต่าง ๆ ได้แก่ UserRx, UserTx, NetwRx, NetwTx และ NodeCon โดยมีตัวชี้สถานะปัจจุบัน (present event) เป็นตัวบอกถึงสถานะการทำงานปัจจุบันของหน่วยที่จะต้องทำงาน เมื่อหน่วยทำงานต่าง ๆ ทำงานครบตามเวลาในการสังเกต หรือครบตามเวลาที่ได้รับการทำงานก็จะต้องกำหนดตัวชี้สถานะถัดไป (next event) เพื่อที่ว่าเมื่อถึงเวลาในการสังเกตถัดไปจะทำงานในสถานะที่ถูกต้อง

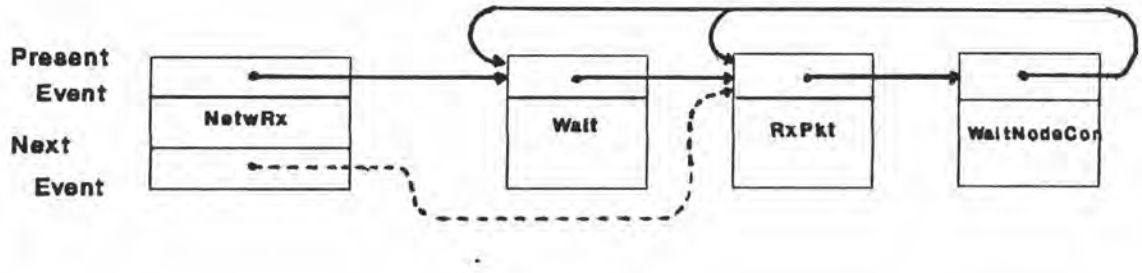


(ก) UserRx

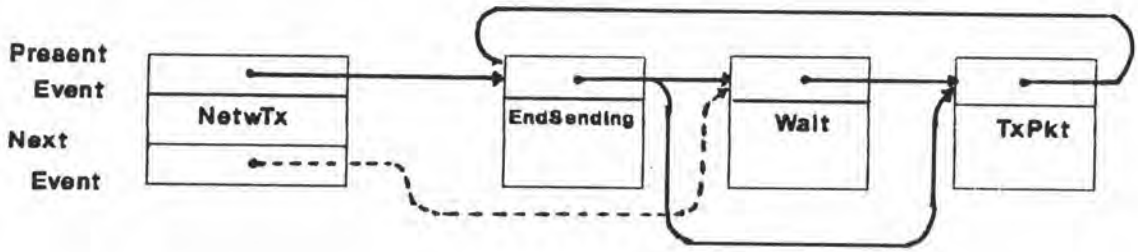


(ข) UserTx

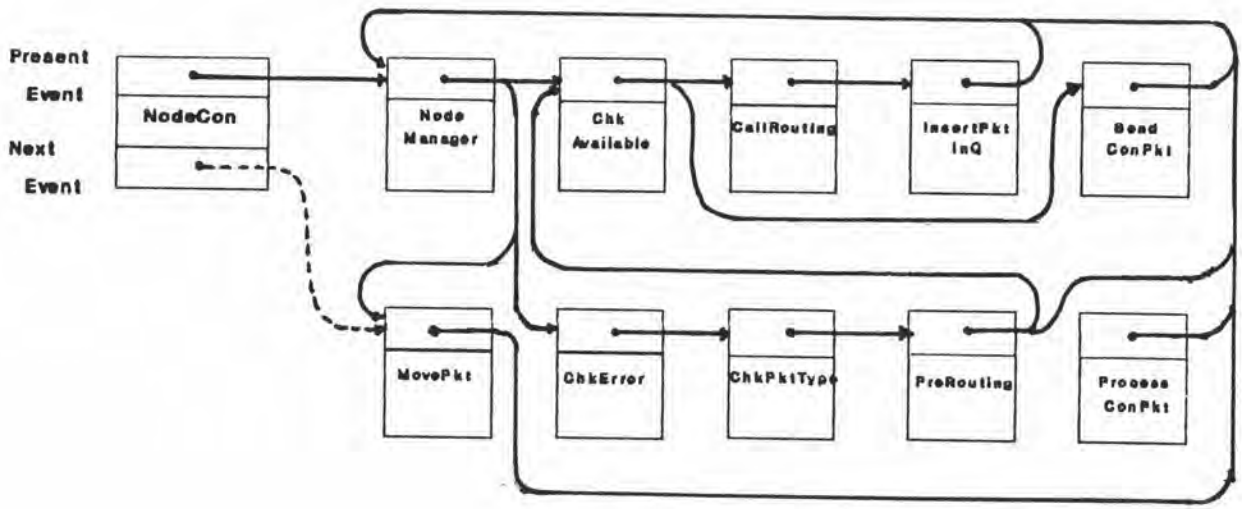
รูปที่ 4.3 การเชื่อมโยงเหตุการณ์ของหน่วยต่าง ๆ (ต่อ)



(ค) NetwRx



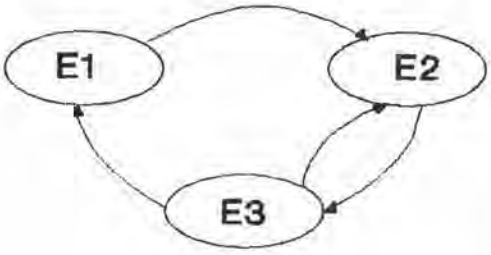
(ง) NetwTx



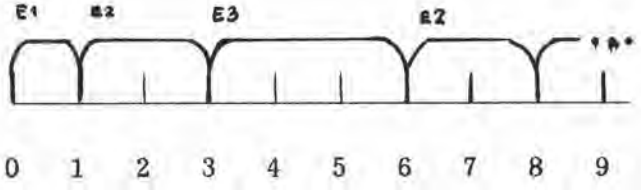
(จ) NodeCon

รูปที่ 4.3 การเชื่อมโยงเหตุการณ์ของหน่วยต่าง ๆ

ในบางสถานะอาจมีการใช้เวลามากกว่า 1 ช่วงเวลาในการสังเกต (ในการทดลองกำหนดให้เท่ากับ 1 unit time) จึงมีโอกาสที่สถานะนั้นยังทำงานไม่เสร็จ ก็จะบันทึกเวลาที่ถูกใช้งานสะสมไว้ก่อนและตัวชี้สถานะถัดไปก็จะชี้ที่สถานะเดิม จนกว่าจะทำงานครบตามเวลาของสถานะนั้น ตัวชี้สถานะถัดไปก็จะชี้ไปยังสถานะอื่น ดังรูปที่ แสดงตัวอย่างในการเปลี่ยนแปลงเหตุการณ์ต่าง ๆ โดยมีสถานะการทำงาน E1, E2 และ E3 และมีเวลาการทำงานของสถานะเท่ากับ 1, 2 และ 3 ตามลำดับ สถานะ E1 และ E2 มีการเปลี่ยนแปลงสถานะแบบไม่มีเงื่อนไข ส่วนสถาน E3 มีการเปลี่ยนแปลงสถานะแบบมีเงื่อนไข ผลการทำงานในแต่ละช่วงเวลาในการสังเกต มีผลดังรูป 4.6 (ข) และ (ค)



(ก) แผนภูมิสถานะการทำงาน



(ข) แสดงการเกิดเหตุการณ์ในเวลาต่าง ๆ

Observed Time	Present Event	Next Event
0	E1	E1
1	E1	E2
2	E2	E2
3	E2	E3
4	E3	E3
5	E3	E3
6	E3	E2
7	E2	E2
8	E2	?

(ค) ตารางแสดงตัวชี้สถานะปัจจุบันและสถานะถัดไปในแต่ละช่วงเวลาในการสังเกต

รูปที่ 4.4 แสดงตัวอย่างการเปลี่ยนแปลงเหตุการณ์ต่าง ๆ

4.4 การจำลองการทำงานของโนดเมื่อประกอบเป็นเครือข่าย

แบบจำลองของโนดที่เราทำการออกแบบไว้แล้วในหัวข้อที่ 3.2 และกำหนดสถานะการทำงานของหน่วยต่าง ๆ ในโนดในหัวข้อที่ 3.3 และจากหัวข้อที่แล้ว เราได้กล่าวถึงวิธีการต่าง ๆ ในการจำลองให้ทำงานแบบขนาน ซึ่งเกิดจากการนำโนดมาประกอบเป็นเครือข่ายเพื่อทำการทดสอบแบบจำลองของโนดว่าจะมีความถูกต้อง หรือมีความผิดพลาดจากระบบการทำงานจริงหรือไม่อย่างไร

การทดสอบแบบจำลองเหตุการณ์ต่างๆ ที่เกิดขึ้น จะต้องมีการทดสอบและวัดผลความถูกต้อง นับตั้งแต่แบบจำลองของโนดที่เราออกแบบในระดับตรรกะไว้แล้วในหัวข้อที่ 3.2 ว่าจะมีหน้าที่การทำงานครอบคลุมการทำงานในเครือข่ายจริงมากน้อยเพียงใด ซึ่งเราตรวจสอบแล้วว่าครอบคลุมการทำงานหลักของเครือข่ายจริงได้ แต่ไม่ลงรายละเอียดในกรรมวิธีมากนักจนถึงระดับกายภาพ จึงได้นำแบบจำลองของโนดดังกล่าวไปออกแบบสถานะการทำงานของหน่วยต่าง ๆ เพื่อนำไปใช้ในการสร้างแบบจำลองของโนดทางซอฟต์แวร์ ดังได้กล่าวไปแล้วในหัวข้อ 3.3 ในการที่เราเลือกใช้แผนภูมิสถานะ เป็นวิธีการหนึ่งที่จะแทนแบบจำลองของโนดในระดับตรรกะได้อย่างถูกต้องโดยสามารถวัดสมรรถะด้วยวิธีการตรวจสอบตัวแปรสถานะและค่าทางสถิติที่มีการเก็บบันทึกในขณะเสร็จสิ้นการทำงานสำหรับบางสถานะ เช่น ในสถานะ User.RxPkt ก็จะมีการเก็บค่าจำนวนแพ็กเก็ตที่ได้รับจากโฮส หรือในสถานะ UserTx.TxPkt ก็จะเก็บค่าจำนวนแพ็กเก็ตที่ส่งออกให้กับโฮส ซึ่งเราถือว่าเป็นค่า Network Throughput ของเครือข่าย หรือการติดตามการทำงานของสถานะ UserRx.RxPkt เมื่อเสร็จสิ้นการทำงาน ตัวแปรสถานะของ UserRx.HavePkt เป็นจริงหรือไม่ เป็นต้น

ความผิดพลาดพื้นฐานในการจำลองนั้นเป็นความผิดพลาดจากการบันทึกค่าของตัวแปรสถานะหรือการเก็บค่าทางสถิติที่เกี่ยวข้องกับเหตุการณ์ต่าง ๆ ที่เกิดขึ้นในขณะจำลอง

ในหัวข้อที่แล้วได้กล่าวถึงการจำลองให้ทำงานแบบขนานและการเชื่อมโยงเหตุการณ์ในการจำลองแบบขนานได้นำมาใช้ในการจำลองการทำงานของโนด โดยมีการกำหนดโครงสร้างของการเชื่อมโยงเหตุการณ์ของหน่วยต่าง ๆ ในโนด ถ้ากำหนดโครงสร้างของการเชื่อมโยงเหตุการณ์แบบง่าย เราก็อาจจะสร้างด้วยวิธีดังกล่าวไว้ในหัวข้อที่แล้ว จะได้ผลลัพธ์ดังตารางในรูปที่ 4.5 ถึง 4.9 ตารางแสดงการเชื่อมโยงเหตุการณ์ของหน่วยต่าง ๆ

Clock (t)	Present	Event Time	Used Time	State Variable (s)	Next Event	Output Variable
	UserRx. Wait	Vary	~	InQ.HavePkt	RxPkt	
	RxPkt	5	~	-	WaitNodeCon	UserRx.HavePkt
	WaitNode Con	Vary	~	((InQ.Empty) & (UserRx.HavePkt = F))	Wait	
				((InQ.HavePkt)& (UserRx.HavePkt =F))	RxPkt	

รูปที่ 4.5 ตารางแสดงการเชื่อมโยงเหตุการณ์ของหน่วย UserRx

Clock (t)	Present	Event Time	Used Time	State Variable (s)	Next Event	Output Variable
	UserTx. Wait	Vary	~	UserTx.HavePkt	TxPkt	
	TxPkt	X rate	~	UserTx.Empty	Wait	NetThroughput

รูปที่ 4.6 ตารางแสดงการเชื่อมโยงเหตุการณ์ของหน่วย UserTx

Clock (t)	Present	Event Time	Used Time	State Variable (s)	Next Event	Output Variable
	NetwRx. Wait	Vary	~	NetwTx[To]. HavePkt	RxPkt	NetwRx.Ready
	RxPkt	X rate	~	NetwTx[To]. TxDone	WaitNodeCon	NetwRx.Buffer .HavePkt
	WaitNode Con	Vary	~	NetwRx.HavePkt. = F	Wait	NoOfRR NoOfError

รูปที่ 4.7 ตารางแสดงการเชื่อมโยงเหตุการณ์ของหน่วย NetwRx

Clock (t)	Present	Event Time	Used Time	State Variable (s)	Next Event	Output Variable
	NetwTx. End Sending	Vary	~	NetwTx[To]. HavePkt	Wait	NoOfTx
				[(NetwRx[From]. Ready)&(NetwTx [To].HavePkt)]	TxPkt	
	Wait	Vary	~	NetwRx[From]. Ready	TxPkt	NetwTx[To]. HavePkt
	TxPkt	Xrate	~	NetwTx[To]. Txdone	EndSending	NetwTx[To]. TxDone

รูปที่ 4.8 ตารางแสดงการเชื่อมโยงเหตุการณ์ของหน่วย NetwTx

Clock (t)	Present	Event Time	Used Time	State Variable (s)	Next Event	Output Variable
	NodeCon. Node Manager	Vary	~	UserRx.HavePkt	ChkAvailable	
				NetwRx[From]. HavePkt	ChkError	
				NetwTx[To]. TxDone	MovePkt	
				[(NetwRx[From]. HavePkt)& Destination= this node	MovePkt	
	ChkAvai-lable	1	~	Full	SendConPkt (RNR)	
				Available	CallRouting	Usedlist. Buffer
	Call Routing	5	~		InsertPktInQ	
	Insert PktInQ	1	~		NodeManager	CurrentPktInQ

รูปที่ 4.9 ตารางแสดงการเชื่อมโยงเหตุการณ์ของหน่วย NodeCon (ต่อ)

Clock (t)	Present	Event Time	Used Time	State Variable (s)	Next Event	Output Variable
	SendCon Pkt	5	~	—	NodeManager	
	PreRout-ing	5	~	Destination = other node	ChkAvailable	
				Destination = this node	NodeManager	
	ChkError	1	~	NoError	ChkPktType	
				Error	SendConPkt (ReTx)	
	ChkPkt Type	1	~	ConPkt	ProcessCon Pkt	
				NotConPkt	PreRouting	
	Process ConPkt	5	~	ReTx or RNR or RR	NodeManager	
	MovePkt	NoOf-Move* 5	~	—	NodeManager	

รูปที่ 4.9 ตารางแสดงการเชื่อมโยงเหตุการณ์ของหน่วย NodeCon

ในตารางดังกล่าวจะมีตัวชี้สถานะปัจจุบัน การทำงานของสถานะนั้น เวลาของสถานะการทำงาน การบันทึกเวลาที่ใช้ไปกับสถานะนั้น ตัวแปรสถานะ ตัวชี้สถานะถัดไป ผลที่ได้เมื่อเสร็จการทำงาน เหตุการณ์ที่ทำให้เกิดการเปลี่ยนแปลงสถานะจากสถานะหนึ่งไปอีกสถานะหนึ่งจะมี 2 ลักษณะ ดังได้กล่าวไปแล้ว

ตัวอย่างการทำงานของหน่วย UserRx (รูปที่ 4.10) จะเริ่มต้นที่สถานะ Wait จนกระทั่งถึงเวลา t=3 โฮสมีแพ็คเก็ตที่ต้องการส่งให้ UserRx (InQ.HavePkt = T, T หมายถึง True) UserRx.Wait จะเปลี่ยนสถานะไปที่ UserRx.RxPkt โดยมีเงื่อนไขจากตัวแปรสถานะ InQ.HavePkt UserRx จะทำงานที่สถานะ UserRx.RxPkt จนกว่าจะครบ 5 unit time เมื่อถึงเวลา t=8 UserRx ทำงานที่สถานะ RxPkt เสร็จ ก็จะเปลี่ยนสถานะไปที่ UserRX.WaitNodeCon และ UserRx จะอยู่ที่สถานะนี้จนกว่าหน่วย NodeCon จะมารับแพ็คเก็ตไปเก็บในบัฟเฟอร์ของโนด (set UserRx.HavePkt = False) ก่อนจะเปลี่ยนสถานะ wait หรือ RxPkt ก็ขึ้นอยู่กับว่าโฮสมีแพ็คเก็ตรอคิวส่งให้ UserRx อีกหรือไม่ ถ้าโฮสยังมีแพ็คเก็ตที่ต้องการส่ง (InQ.HavePkt = T) ก็จะเปลี่ยนสถานะไปที่ RxPkt แต่ถ้าโฮสยังไม่มีแพ็คเก็ตที่ต้องการส่ง (InQ.HavePkt = F, F หมายถึง False) ก็จะเปลี่ยนสถานะไปที่ wait เราได้แสดงรายละเอียดของข้อมูลการทำงานของ UserRx ในแต่ละสถานะตามเวลาต่างๆ ในรูปที่ 4.10 ตัวอย่างการทำงานของหน่วย UserRx สำหรับหน่วยอื่น ๆ ก็จะมีการทำงานคล้ายกับที่ได้อธิบายไปแล้วข้างต้น

Clock (t)	Present Event	Event Time	Used Time	State Variable	Next Event	Output
0	Wait	Vary	—	(InQ.HavePkt)	—	—
1	Wait	Vary	1	InQ.HavePkt = F	Wait	—
2	Wait	Vary	2	InQ.HavePkt = F	Wait	—
3	Wait	Vary	3	InQ.HavePkt = T	RxPkt	
4	RxPkt	5	1	-	RxPk	
5	RxPkt	5	2	-	-	
6	RxPkt	5	3	-	-	UserRx.
7	RxPkt	5	4	-	-	HavePkt
8	RxPkt	5	5	-	WaitNodeCon	= T
9	WaitNodeCon	Vary	1	-		
.	.	.	.	.	.	.
.	.	.	.	.	.	.
n	.	.	.	(User Rx.HavePkt and InQ.HavePkt)	RxPkt or Wait	.

รูปที่ 4.10 ตัวอย่างการทำงานของหน่วย UserRx

จากตารางการเชื่อมโยงเหตุการณ์ของหน่วยต่าง ๆ จะนำไปใช้ในการสร้างแบบจำลองทางซอฟต์แวร์และใช้ในการกำหนดโครงสร้างข้อมูลของแต่ละหน่วย ตลอดจนถึงการสร้างโปรแกรมจำลอง ซึ่งจะได้กล่าวในบทที่ 5