การศึกษาการทำงานของ FDDI โดยการสิมิวเลชั่น

THE STUDY OF FDDI BY COMPUTER SIMULATION

โดย

นางสาวดอกไม้ สนิทนิตย์

นางสาวเสาวลักษณ์ แสนวิทยากร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2534

การศึกษาการทำงานของ FDDI โดยการสิมิวเลชั่น

นางสาว ดอกไม้ สนิทนิตย์
นางสาว เสาวลักษณ์ แสนวิทยากร

อาจารย์ที่ปรึกษา
อาจารย์ สุวิพล สิทธิชีวภาค

บทคัดย่อ

โครงงานนี้ เป็นการวิเคราะห์การทำงานของ เนตเวอร์คความเร็วสูง Fiber Distributed Data Interface (FDDI) ซึ่งเป็นเนตเวอร์คที่มีอัตราการส่งข้อมูล 100 Mbps สามารถส่งได้ทั้ง สัญญาณภาพ เสียงและข้อมูล เนื่องจากการศึกษาระบบการทำงานโดยการสร้างระบบอาร์ดแวร์นั้น ทำให้ยุ่งยากและสิ้นเปลืองค่าใช้จ่ายมากนอกจากนี้ผลจากการทดลองยังสังเกตได้ยาก เนื่องจากระบบใหญ่เกินไป เกิดช้าเกินไปและเร็วเกินไปบ้าง ดังนั้นในโครงงานนี้ได้ทำการจำลองระบบ FDDI โดยใช้โปรแกรมคอมพิวเตอร์ภาษาซี ระบบจำลองในโครงงานนี้ สามารถจำลองเหตุการณ์ต่างๆ ทุกเหตุการณ์ที่เกิดขึ้นได้เหมือนกับในระบบจริง และผลจากการออกแบบนั้นสามารถใช้ตรวจสอบลักษณะการทำงานของ FDDI ในลภาวะต่างๆ โดยที่เราสามารถเปลี่ยนค่าตัวแปรที่มีผลต่อระบบ เช่น ขอบเขตของเนตเวอร์ค ขนาดแพคเกตข้อมูล จำนวนเทอร์มินอล และดูผลจากการเปลี่ยนค่าตัวแปรเหล่านี้ได้ รวมทั้งดูประสิทธิภาพการทำงานของระบบได้ทันที

THE STUDY OF FDDI BY COMPUTER SIMULATION

Ms. Dokmai Sanitnit

Ms. Saovaluk Sanwittayakon

ADVISOR

Dr. Suvepon Sittichivapak

ABSTRACT

This reserch project presents the performance of FDDI (Fiber Distributed Data Interface). The FDDIis an high speed network which sends data rate at 100 Mbps and also has a function used to send voice and video. For the fact that it is too difficult to study FDDI in hardware because of high cost. However we can simulate the system by software program simulator. So this project has simulates this system by computer software methodin C language. Our simulator can perform all events that happen in real FDDI system and in our experiment result we can check FDDI system by vary system parameters,i.e.,system network length, packet size, the number of terminals and transmission speed.

สารบัญ

บทที่ 1

บทนำ

ในปัจจุบันการสื่อสารได้พัฒนาขึ้นอย่างรวดเร็ว ทำให้มีความสะดวกในการติดต่อสื่อสาร ระบบการสื่อสารต่างๆ ถูกพัฒนาขึ้นเพื่อสนองความต้องการ และระบบการสื่อสารที่ได้รับความนิยมอย่างมากคือ การสื่อสารในระบบโลคัลแอเรียเนตเวอร์ค (Local Area Network or LAN) ซึ่งเป็นการสื่อสารผ่านข้อมูลทางคอมพิวเตอร์ โดยแรกเริ่มเนตเวอร์คนี้จะเป็นเนตเวอร์คที่ส่งข้อมูลด้วยความเร็วต่ำ คือประมาณ 1-10 Mbps ต่อมาเนื่องจากความเจริญทางด้านเทคโนโลยี และทางด้านโทรคมนาคมได้เจริญขึ้น และเพื่อสนองความต้องการเหล่านี้ จึงได้มีการพัฒนาเนตเวอร์ค ที่สามารถส่งข้อมูลได้ด้วยความเร็วสูง สามารถสนองความต้องการทั้งด้านการส่งข้อมูลที่เป็นดาต้า (Data) สัญญาณภาพ (Graphic) และสัญญาณเสียง (Voice) และตัวกลางที่จำเป็นสำหรับเนตเวอร์คความเร็วสูงนี้คือ เส้นใยแสง (Optical Fiber) ซึ่งจะได้กล่าวถึงในบทต่อไป

สำหรับเนตเวอร์คความเร็วสูงที่จะกล่าวถึงนี้คือ FDDI (Fiber Distributed Data Interface) ซึ่งพัฒนามาจาก IEEE802.5 โทเกนริงค์ (Token Ring) ลักษณะบางประการของ FDDI จะเหมือนกับ IEEE802.5 แต่บางประการจะแตกต่างกัน เช่นในเรื่องของตัวกลางในการส่ง ความเร็วในการส่งข้อมูล และการจัดการของโทเกน เป็นต้น ดังตาราง 1.1 แสดงการเปรียบเทียบระหว่าง IEEE802.5 โทเกนริงค์ กับ FDDI

ตาราง 1.1 เปรียบเทียบระหว่าง IEEE802.5 กับ FDDI

	FDDI	IEEE802.5
ตัวกลางที่ใช้	เส้นใยแสง	โคแอคเชียลเคเบิ้ล
ความเร็วในการส่งข้อมูล	100Mbps	1-10 Mbps
ขอบเขตพื้นที่	100 Km	1-5 Km
หลักการรับส่งข้อมูล	โทเกนพาสซิ่ง	โทเกนพาสซิ่ง

บทที่2

โลคัลแอเรียเนตเวอร์คสมัยใหม่ (Modern Local Area Network)

โลคัลแอเรียเนตเวอร์ค เรารู้จักกันว่าเป็นระบบการสื่อสารข้อมูลที่ยอมให้อุปกรณ์ที่แตกต่างกันสามารถติดต่อกันได้ โลคัลแอเรียเนตเวอร์คนั้นมีหลายชนิด มีขอบเขตจำกัดในบริเวณที่ไม่ไกลนัก เช่น ใช้ในอาคารสำนักงาน หรือตามมหาวิทยาลัย เป็นต้น การประยุกต์ใช้งานของเนตเวอร์คนี้ มีทั้งในด้านการพาณิชย์, การอุตสาหกรรม หรือสถาบันต่างๆ โลคัลแอเรียเนตเวอร์คนั้นจะต้องให้การสนับสนุน (support) บริการทั้งการเคลื่อนย้ายไฟล์ (file transfer) กราฟฟิค (graphics) เวิร์ดโปรเซสซิงค์ (word processing) อิเลคทรอนิคเมล์ (electronic mail) การกระจายเดต้าเบส (distributed data bases) การติดต่อระหว่างโลคัลแอเรียเนตเวอร์คด้วยกัน การบริการวิดีโอบางประเภท นอกจากนี้โลคัลแอเรียเนตเวอร์ค ยังจะต้องซัพพอร์ตอุปกรณ์เดต้าที่แตกต่างกัน คือ คอมพิวเตอร์แบบต่างๆ (ไมโคร มินิ และมินิแมคซี่) วิดีโอเทอร์มินอล (Video terminal) เครื่องพิมพ์ (Printer) และเกทเวย์ (Gateway) สำหรับติดต่อกับเนตเวอร์คอื่น

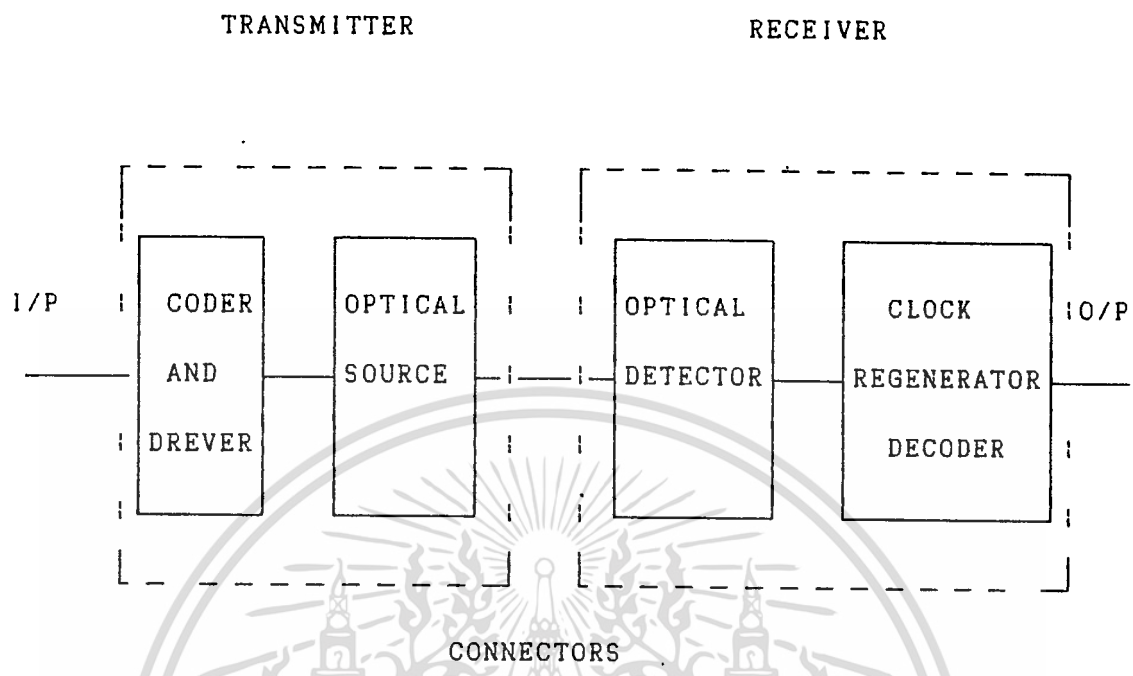
ต่อมาเนื่องจากความเจริญทางด้านเทคโนโลยี และโทรคมนาคมได้เจริญขึ้น และระบบคอมพิวเตอร์เนตเวอร์คได้พัฒนาอย่างรวดเร็ว ทำให้มีการพัฒนาระบบโลคัลแอเรียเนตเวอร์คให้สามารถส่งข้อมูลได้ด้วยความเร็วสูง ซึ่งตัวกลางที่ใช้ในการสื่อสารข้อมูลด้วยความเร็วสูงจำเป็นต้องใช้ระบบการส่ง เส้นใยแสง สำหรับเส้นใยแสงนี้ได้ถูกปรับปรุงใช้งานทางด้าน เคเบิ้ลทีวี หรือทางด้านโทรศัพท์ได้อย่างมีประสิทธิภาพ อย่างไรก็ตามเมื่อถูกนำมาใช้งานในเนตเวอร์คความเร็วสูง นั้นมีข้อจำกัดหลายประการทางด้านปริมาณการส่งข้อมูล เช่น ความสูญเสียตรงจุดต่อเชื่อม คุณสมบัติของตัวส่งและรับสัญญาณแสง และความยุ่งยากของอุปกรณ์ที่จะใช้ในการแอคเซสตัวกลาง ซึ่งปัญหาเหล่านี้ทำให้การใช้เส้นใยแสง ในโครงสร้างเนตเวอร์คแบบโลคัลแอเรียถูกจำกัดไม่ก้าวหน้าเท่าที่ควร

2.1 ระบบการสื่อสารเส้นใยแสง

เส้นใยแสงมีคุณสมบัติต่างๆ ที่เหมาะสมสำหรับใช้เป็นตัวกลางในระบบการสื่อสารที่มีความจุชัลแนลสูง คุณสมบัติที่สำคัญมีดังนี้

1. แบ่นด์วิดธ์ (bandwidth) กว้าง สามารถส่งข้อมูลด้วยอัตรา 100 Gbps ในระยะทางหนึ่งร้อยกิโลเมตร แต่ปัจจุบันที่ใช้กันอยู่ทั่วไปนั้นสามารถส่งข้อมูลด้วยความเร็วเป็นเมกะบิทต่อวินาที (Mbps)
2. เนื่องจากเส้นใยแสงทำจากแก้วซึ่งเป็นไดอิเลคตริค จึงปลอดภัยจากการรบกวนทางอิเลคโตรแมคเนติค
3. มีขนาดเล็กน้ำหนักเบา ขนาดเส้นผ่าศูนย์กลางประมาณ 5-10 ไมโครเมตร
4. การลดทอนต่ำประมาณ 0.2 dB ต่อกิโลเมตร

อุปกรณ์พื้นฐานที่ใช้ในระบบการสื่อสารเส้นใยแสงได้แก่ เครื่องส่งแสง (optical transmitter) เครื่องรับแสง (optical receiver) อุปกรณ์ต่อเชื่อม (optical connector) และคอปเลอร์ (coupler) ต่อไปนี้เราจะพิจารณารูปแบบง่ายๆ ของการติดต่อซึ่งประกอบด้วยเครื่องส่งและเครื่องรับ ซึ่งเชื่อมต่อโดยใช้ชิงเกิ้ลออฟติคไฟเบอร์ ดังรูป 2.1 เครื่องส่งนั้นจะประกอบด้วยวงจรอิเลคทรอนิค แหล่งกำเนิดแสงก็คือ LED (Light Emitting Diode) หรือเลเซอร์ไดโอด (Laser Diode) ลำแสงที่ถูกมอดูเลทแล้วก็จะถูกส่งออกไปตามเส้นใยแสง ส่วนที่ทางด้านรับเมื่อแสงกระทบกับผิวของออฟติคอลดีเทคเตอร์ ซึ่งเป็น p-i-n หรืออะวาลานซ์โฟโต้ไดโอด (avalance photodiode) พลังงานแสงก็จะถูกเปลี่ยนกลับเป็นสัญญาณไฟฟ้าตามเดิม และวงจรดีเทคสัญญาณก็จะเปลี่ยนให้เป็นสัญญาณเริ่มแรกก่อนที่จะส่งมา สำหรับคอนเนคเตอร์นั้นใช้สำหรับเชื่อมต่ออุปกรณ์ส่งกับเส้นใยแสง และเส้นใยแสงกับอุปกรณ์รับและใช้เชื่อมต่อระหว่างเส้นใยแสง



รูปที่ 2.1 แสดงตัวอย่างการสื่อสารเส้นใยแสง

พารามิเตอร์ของระบบที่สำคัญบางประการ ซึ่งจะเป็นตัวกำหนดว่าการเชื่อมโยงนั้นจะทำได้ในรูปแบบใดบ้างจะมีอยู่สามอย่างคือ

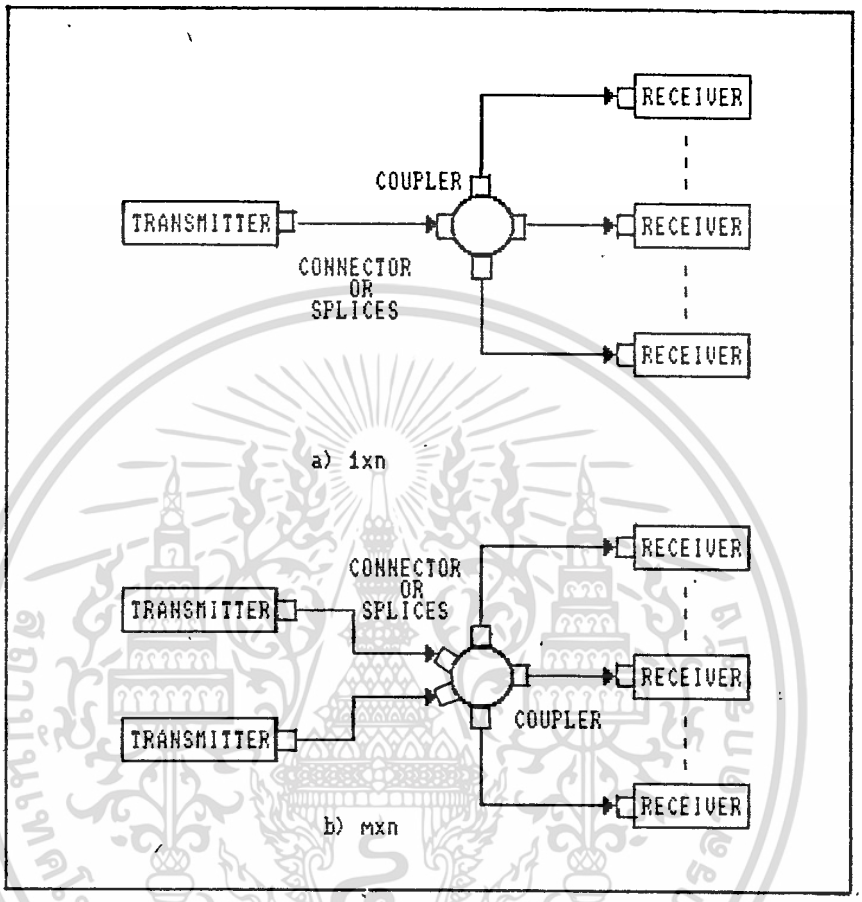
1. กำลังแสงของสัญญาณที่รับเข้ามาที่ดีเทคเตอร์
2. ความเพี้ยนของสัญญาณที่ส่งผ่านเส้นใยแสง
3. เวลาสิ้นสุดของระบบ

ซึ่งกำลังของสัญญาณที่รับเข้ามาจะต้องมีค่ามากกว่าค่าที่กำหนดไว้ค่าหนึ่ง ซึ่งเป็นฟังก์ชันของเครื่องรับแสง การสูญเสียกำลังของแสงในการเชื่อมโยงนั้นควรจะมีค่าที่เหมาะสมในวงที่จำกัดไว้ ซึ่งค่าความสูญเสียเหล่านี้รวมค่าสูญเสีย ระหว่างเครื่องส่งและเครื่องรับ ถ้าหากค่าความสูญเสียนี้เกินจำนวนของผลต่าง ระหว่างกำลังที่ส่งเข้าไปในเส้นใยแสง และค่าน้อยที่สุดที่รับได้โดยเครื่องรับแสง แสดงว่าการเชื่อมโยงนั้นไม่เป็นที่ยอมรับ ในเหตุการณ์เช่นนี้ จึงต้องใช้เครื่องส่งที่มีกำลังส่งได้เต็มที่กว่านี้ และเครื่องรับแสงจะต้องไวต่อการรับแสง หรือเส้นใยแสงจะต้องมีค่าความสูญเสียต่ำ

เวลาสิ้นสุดของระบบนั้นจะถูกกำหนดโดยบิทเรต (Bitrate) ค่าต่างๆ เหล่านี้ พารามิเตอร์ที่สำคัญที่เกี่ยวข้องคือ วัสดุที่ใช้ทำเส้นใยแสงและรูปแบบการกระจายของแสง ทั้งสองนี้สามารถทำให้เกิดการขยายของพัลส์ (Pulse Dispersion) และทำให้เกิดการรบกวนแบบอินเตอร์ซิมโบล (Intersymbol noise) ขณะที่แสงเดินทางผ่านเส้นใยแสง การกระจายของวัสดุที่ใช้ทำเส้นใยแสงนั้น อาจทำให้พัลส์ของแสงขยายออกเนื่องจากค่าดีเลย์ของความยาวคลื่นที่แตกต่างกันในรูปของลำแสง ส่วนการกระจายโหมดจะทำให้เกิดการขยายพัลซ์ซึ่งเกิดจากทางเดินของคลื่นต่างๆ กันในมัลติโหมดไฟเบอร์

การสูญเสียของแสง อาจจะเกิดที่จุดเชื่อมต่อระหว่างเครื่องส่งแสงกับเส้นใยแสง ที่จุดต่อเชื่อมทุกจุด และรอยต่อของเส้นใยแสง นอกจากนี้ยังมาจากการสูญเสียแสงเนื่องจากเส้นใยแสงเอง

การเชื่อมต่อแบบ 1xn นั้นจะแยกลำแสงจากเครื่องส่งตัวเดียวแล้วแยกออกหลายๆ เอาท์พุทพอร์ท โดยใช้อุปกรณ์เชื่อมต่อ ซึ่งจะเป็นฟังก์ชันของโครงสร้างอุปกรณ์เชื่อมต่อ โดยจะพิจารณาจากความต้องการของระบบ โดยทั่วไปจะเป็นการเชื่อมต่อแบบ mxn โดยที่ m คือจำนวนอินพุทพอร์ท และ n คือจำนวนเอาท์พุทพอร์ท ถ้าสมมติสมการการกระจายของกำลังแสงที่มีการเชื่อมต่อแบบ 1xn กำลังแสงจะลดลงเมื่อค่า 1/n มีค่าน้อย เมื่อแสงผ่านอุปกรณ์ต่อเชื่อม ซึ่งเขียนในรูปลอการิทึมได้ดังนี้คือ ลดลงเท่ากับ -10logn dB การเพิ่มของค่าความสูญเสียจะปรากฏที่ตัวต่อเชื่อม และเป็นการยากที่จะจัดให้อยู่ในรูปสมการที่แน่นอน

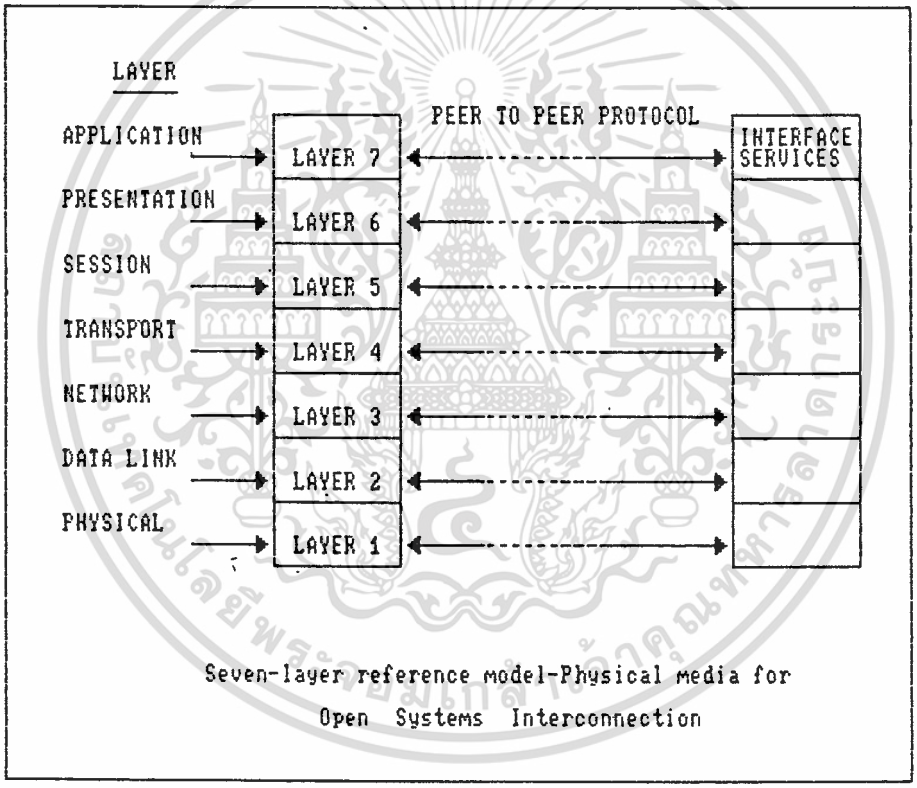


รูปที่ 2.2 การเชื่อมโยงเส้นใยแสงด้วยเพาเวอร์คอปเลอร์

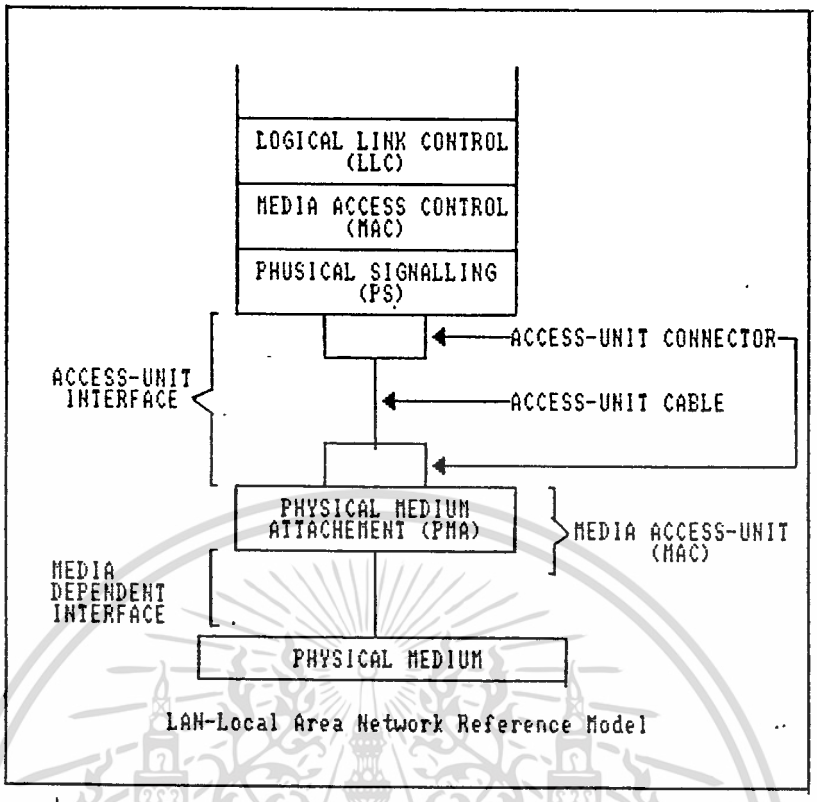
2.2 โครงสร้างของออฟติคไฟเบอร์โลคัลแอเรียเนตเวอร์ค

โครงสร้างในที่นี้หมายถึงระดับชั้นของเนตเวอร์ค ซึ่งกำหนดโดยองค์การมาตรฐานนานาชาติ(InternationalStandardOrganization หรือ ISO) เรียกว่า OSI (Open Systems Interconnection) หรือกำหนด โดย IEEE802 ซึ่งแสดงให้เห็นดังรูป 2.3 และ รูป 2.4 และความสัมพันธ์ระหว่างสองรูปนี้แสดงดังรูป 2.5 สำหรับมาตรฐานของ OSI นั้นประยุกต์ใช้งานกับคอมพิวเตอร์เนตเวอร์คทั่วๆ ไปขณะที่ LAN นั้น

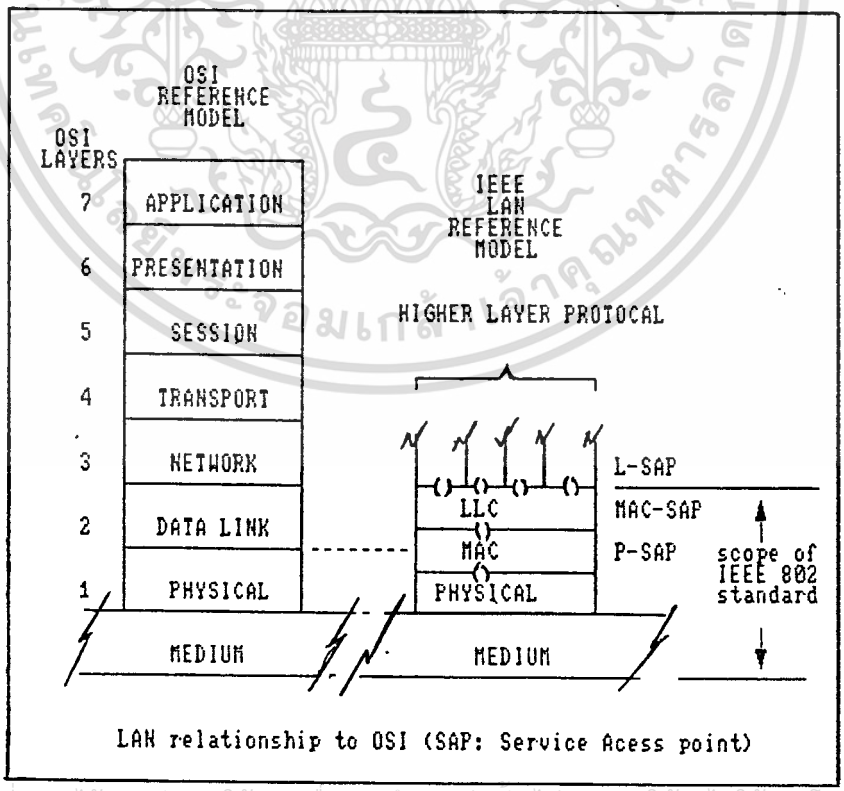
ตั้งขึ้นเพื่อพยายามให้เป็นมาตรฐานสำหรับโลคัลแอเรียเนตเวอร์ค ในที่นี้จะกล่าวถึงสองระดับชั้นแรกของมาตรฐาน คือระดับชั้นฟิสิคัลและระดับชั้นเดต้าลิงค์ของ OSI หรือระดับชั้นฟิสิคัลและระดับชั้นมีเดียมแอคเซสของ LAN



รูป2.3 แสดงโครงสร้างมาตรฐานของ OSI



รูปที่ 2.4 โครงสร้างมาตรฐานของโลคัลแอเรียเนตเวอร์ค



รูปที่ 2.5 แสดงความสัมพันธ์ระหว่าง OSI กับ LAN IEEE802 โมเดล

2.2.1 **มาตรฐานของ OSI มี 7**

1. **ระดับชั้นฟิสิคัล** (Physical Layer) ในระดับนี้จะเกี่ยวข้องกับการส่งข้อมูลดิบ กล่าวคือ พิจารณาในแง่ของบิทที่ผ่านไปตามช่องทางการสื่อสาร การออกแบบมาใช้งานจะต้องก่อให้เกิดความมั่นใจได้ว่า เมื่อปลายทางอีกด้านหนึ่งส่งบิท "1" ออกมาแล้วที่ปลายทางด้านตรงข้ามจะต้องได้รับบิท "1" ออกมาด้วย

2. **ระดับชั้นเดต้าลิงค์** (Data Link Layer) เป็นระดับที่จะทำหน้าที่อำนวยการส่งสัญญาณเข้าสู่เครือข่ายการสื่อสาร โดยปราศจากข้อผิดพลาด หน้าที่ต่างๆ ได้แก่การจัดข้อมูลให้เป็นแพคเกต (Packet) การรับรองการส่งแพคเกตให้ไปถึงสถานีรับ และการตรวจสอบแก้ไขข้อผิดพลาด

3. **ระดับชั้นเนตเวอร์ค** (Network Layer) ทำหน้าที่เกี่ยวกับการเลือกเส้นทาง (Routing) และสวิทชิ่ง (Switching)

4. **ระดับชั้นทรานสปอร์ต** (Transport Layer) ทำหน้าที่แบ่งข้อมูลข่าวสาร (massage) ออกเป็นส่วนย่อยๆ และมีวิธีการทำให้การส่งผ่านข้อมูลนั้นเป็นไปอย่างถูกต้องจนถึงสถานีรับ

5. **ระดับชั้นเซสชั่น** (Session Layer) เกี่ยวกับการจัดและรักษาการติดตั้งการคิดเงิน และการจัดชนิดของการสื่อสาร

6. **ระดับชั้นพรีเซนเตชั่น** (Presentation Layer) เกี่ยวกับการแบ่งงานออกเป็นหลายๆ ส่วน เพื่อสะดวกต่อการสื่อสารระหว่างผู้ใช้บริการ เช่น โคดคอนเวอร์ชั่น (Code Conversion) เทคคอมเพรสชั่น (Text Compression) เป็นต้น

7. **ระดับแอปปลิเคชั่น** (Application Layer) ในระดับนี้จะครอบคลุมถึงระดับผู้ใช้ที่จะประยุกต์ใช้งานแต่ละคน เช่นการส่งไฟล์ เป็นต้น

2.2.2 **มาตรฐานของ LAN มี 3 ระดับคือ**

1. **ระดับชั้นฟิสิคัล** (Physical Layer) รับผิดชอบเกี่ยวกับการสร้างสัญญาณการส่งแบบบอร์ดแบนด์ หรือเบสแบนด์

2. **มีเดียมแอคเซสคอนโทรล** (Mediam Access Control) รับผิดชอบเกี่ยวกับการแอคเซสตัวกลาง เช่น แบบ CSMA/CD หรือ โทเกนริงค์

3. **ลอจิคัลลิงค์คอนโทรล** (Logical Link Control) เกี่ยวกับการควบคุม

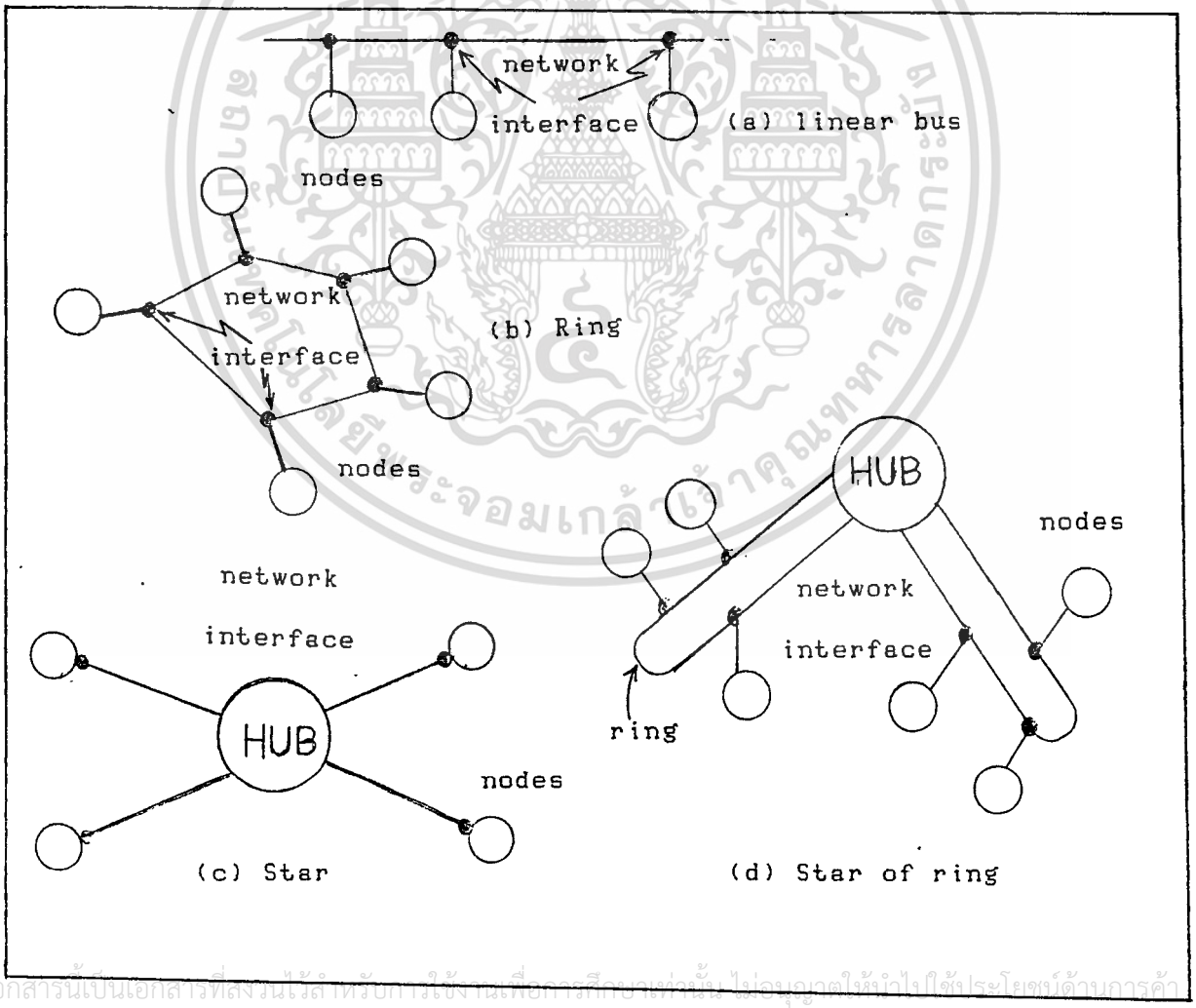
การไหลของข้อมูล (Flow Control) และการบริการ

2.2.3 การจัดโครงข่ายของออปติคไฟเบอร์โลคัลแอเรียเนตเวอร์ค มีหลายแบบ ดังแสดงในรูป 2.6

1. แบบบัส (Bus) การเชื่อมต่อจะเป็นลักษณะเรียงต่อกันตามลำดับ สัญญาณเดินทางได้สองทิศทาง

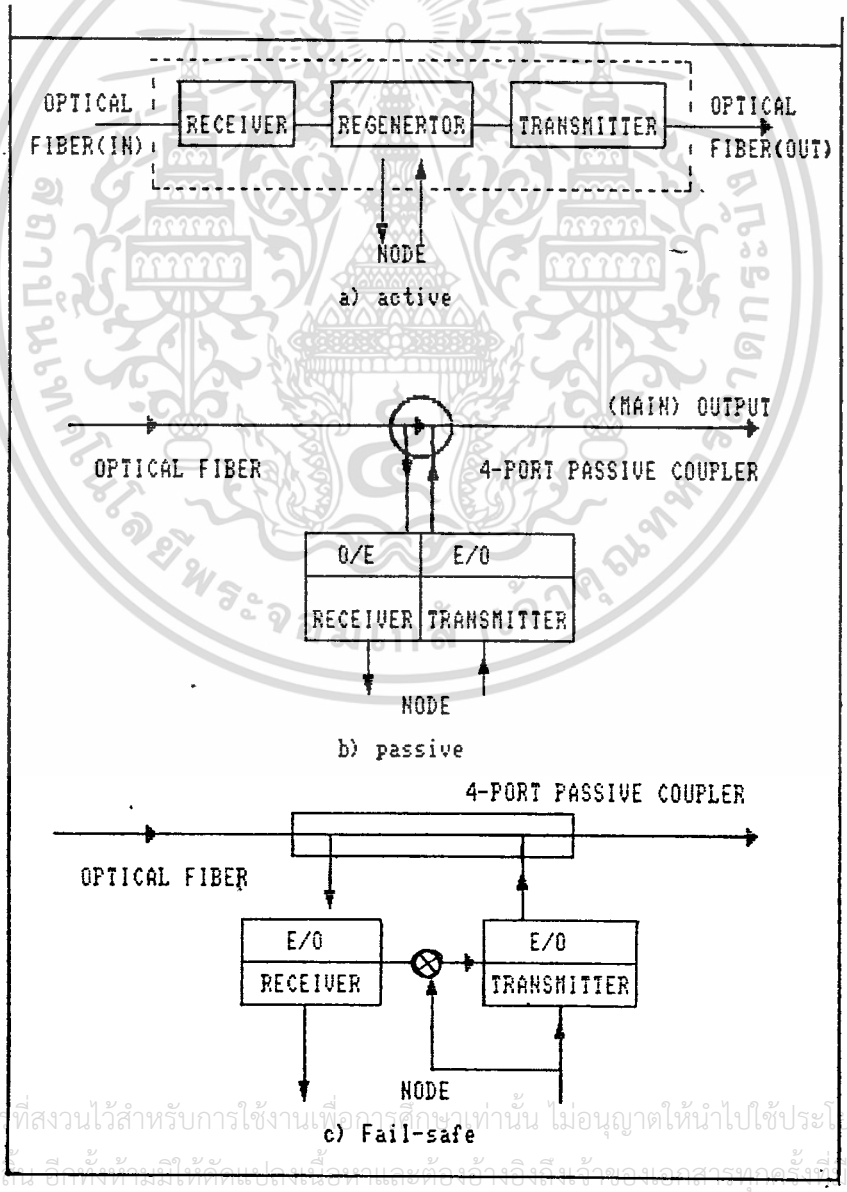
2. แบบริงค์ (Ring) การเชื่อมต่อจะเรียงต่อกันเป็นรูปวงแหวน คือตัวสุดท้ายจะวนกลับมาเชื่อมต่อกับตัวแรก ดังนั้นสัญญาณสามารถเดินทางได้ทิศทางเดียว

3. แบบสตาร์ (Star) ทุกสถานีจะเชื่อมต่อกับสถานีศูนย์กลาง (Hub) การสื่อสารจะเป็นแบบสองทางไป-กลับ



รูปที่ 2.6 แสดงการจัดโครงข่ายของโลคัลเนตเวอร์ค

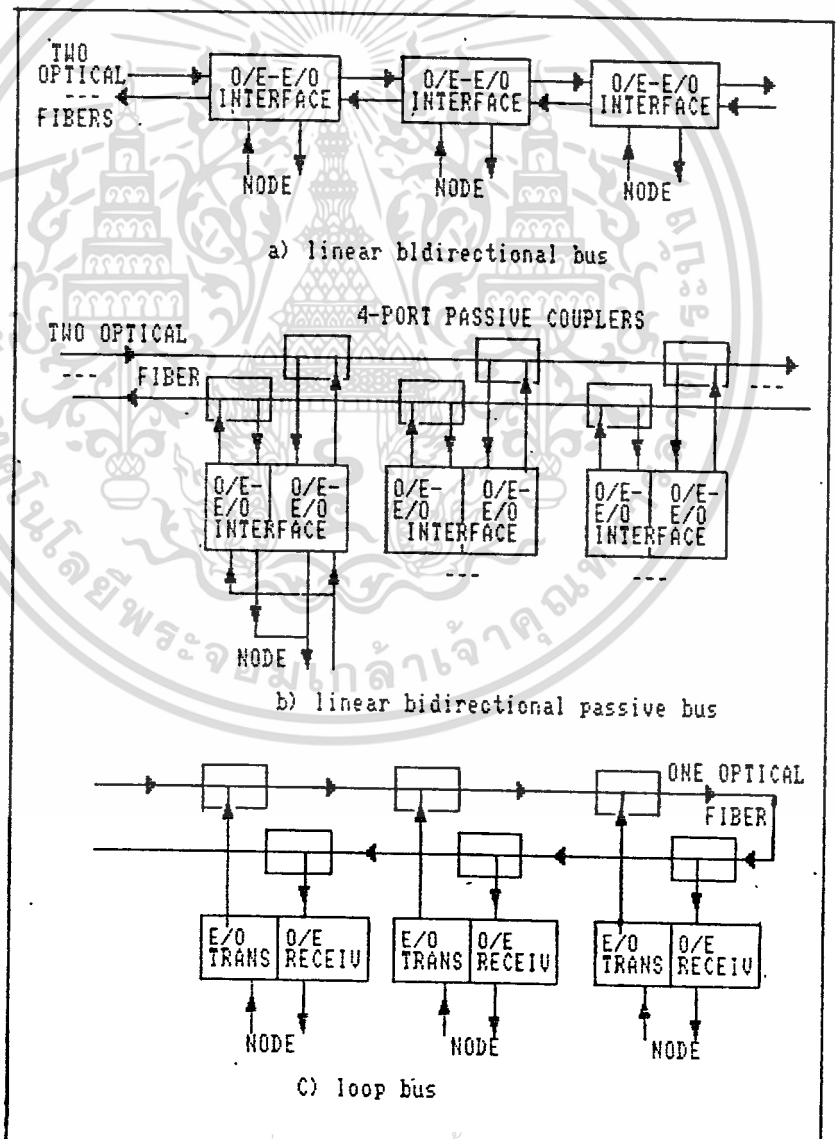
นอกจากที่กล่าวมานี้ยังมีรูปลักษณะอย่างอื่น จะเป็นการประกอบขึ้นจากการผสมแต่ละแบบเข้าด้วยกัน สำหรับบัสและริงค์นั้นสามารถแยกแยะได้อีก จากการใช้อุปกรณ์สำหรับอินเตอร์เฟสดังรูป 2.7 ซึ่งอาจจะเป็นอุปกรณ์แอคทีพ (Active) หรือแพสซีพ(Passive) ในกรณีแพสซีพนั้นอินเตอร์เฟสจะรับ-ส่งสัญญาณโดยตรง สำหรับกรณีแอคทีพนั้น สัญญาณแสงที่จะผ่านสู่สถานีนั้นจะเปลี่ยนเป็นสัญญาณไฟฟ้า เข้าออกสู่สถานี ซึ่งจะมีการเปลี่ยนสัญญาณไฟฟ้าเป็นแสงในตัวอินเตอร์เฟส ดังนั้นข้อมูลที่เข้าออกสู่สถานี สามารถที่จะเพิ่มลดหรือตบแต่งได้ โครงสร้างทั่วๆ ไปของแอคทีพบัสแสดงดังรูป 2.7a และของแพสซีพบัสแสดงดังรูป 2.7b สำหรับในรูป 2.7c นั้นเป็นไฮบริดจ์อินเตอร์เฟสซึ่งรวมเอาทั้งแบบแอคทีพบัสและแพสซีพบัส ซึ่งเรียกว่า Failsafe ถ้าหากแอคทีพบัสเสียก็จะใช้แบบแพสซีพแทน



รูปที่ 2.7 แสดงการอินเตอร์เฟส

แบบบัส

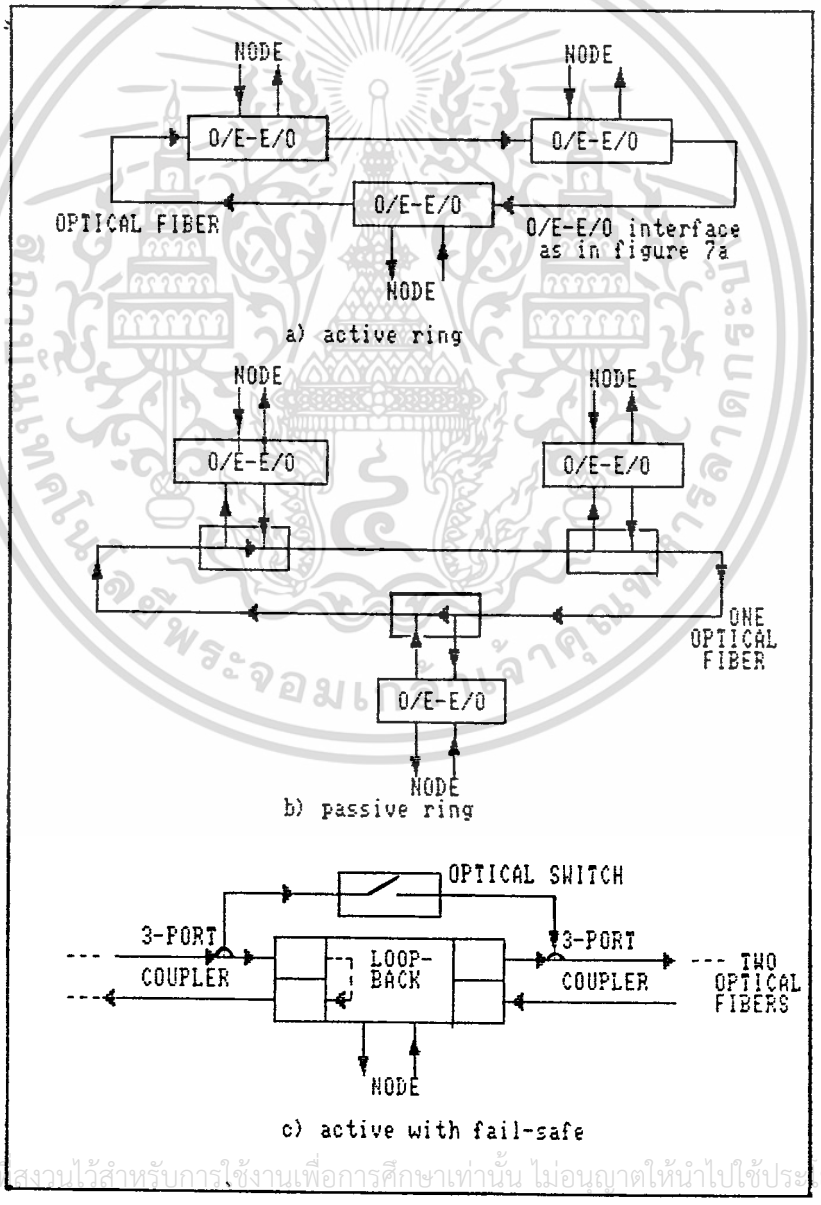
รูปร่างของบัสที่เป็นแบบแอคทีพและแพสซีพ แสดงดังรูป 2.8a และ รูป 2.8b แอคทีพบัสนั้นประกอบด้วย แอคทีพคอปเลอร์ในแต่ละจุดสัมผัสกับเนตเวอร์คโดยทิศทางละอัน ส่วนแพสซีพบัสนั้นจะประกอบจากแพสซีพคอปเลอร์ที่ซับซ้อน ในกรณีของแอคทีพบัสนั้นราคาของอินเตอร์เฟส และความซับซ้อนของวงจรอิเลคทรอนิคส์ ทำให้การพัฒนาในรูปร่างแบบนี้ ไม่ก้าวหน้าเท่าที่ควร ส่วนกรณีของแพสซีพนั้น จากการสูญเสียพลังงานของตัวคอปเลอร์ซึ่งจะเป็นตัวกำหนดจำนวนสถานีที่จะผ่านต่อกับเนตเวอร์ค ทำให้ลักษณะของบัสเปลี่ยนไปเป็นดังรูป U ดังแสดงในรูป 2.8c



รูปที่ 2.8 แสดงลีเนียร์ไฟเบอร์ออฟติคบัส (Linear Fiber Optic buses)

แบบริงค์

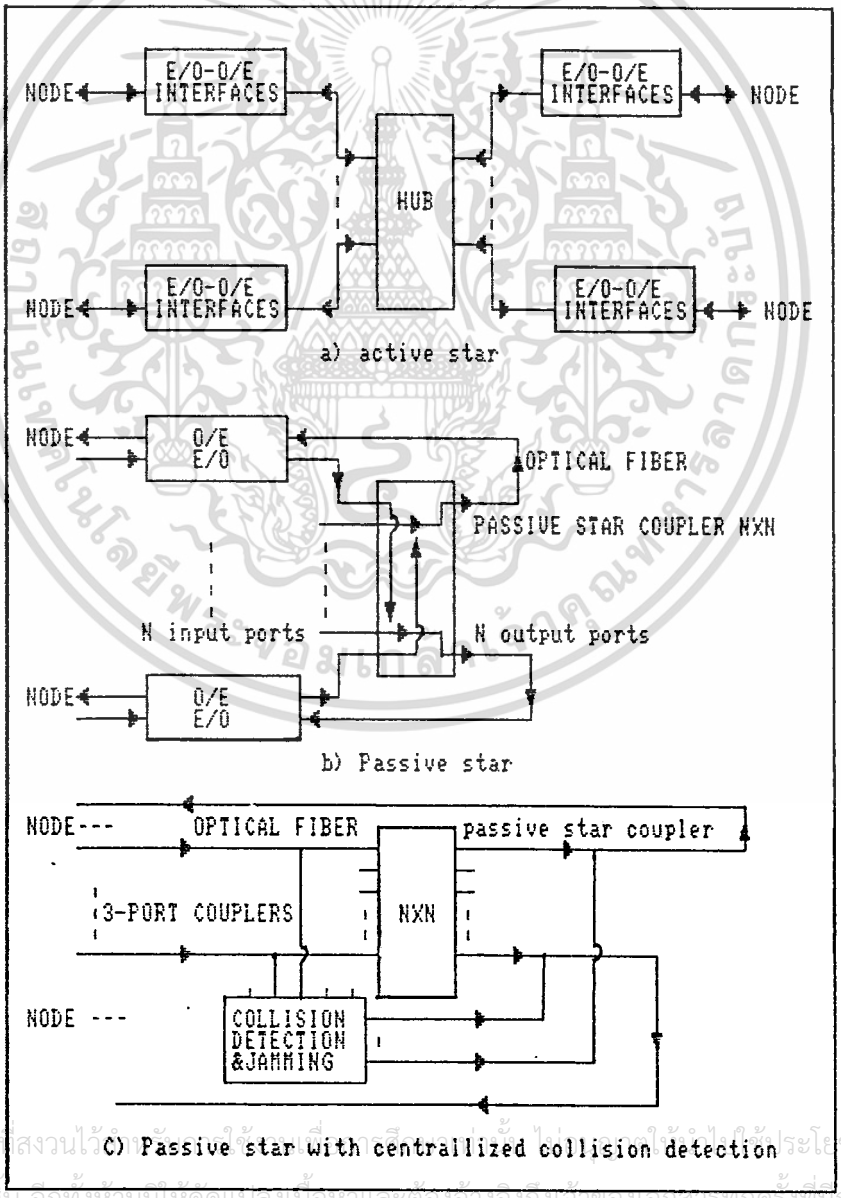
รูปร่างของริงค์นั้นลักษณะคล้ายๆ กับของบัส จะแบ่งออกเป็นสองแบบคือ แบบแอคทีพและแบบแพสซีพ ดังรูปที่ 2.9a และ 2.9b สำหรับแอคทีพริงค์นั้นเมื่อสถานีใดๆ เสียหรือเส้นใยแสงเกิดแตกหักขึ้น จะเป็นสาเหตุให้ทั้งระบบไม่ทำงาน (ปัญหานี้จะแก้ไขโดยการเพิ่มสายเคเบิ้ลเผื่อไว้) ส่วนแบบแพสซีพริงค์นั้นจะไม่มีปัญหา เมื่อสถานีเกิดเสียแต่จะมีปัญหาเมื่อเส้นใยแสงเกิดแตกหักขึ้น และจะสังเกตได้ว่า เมื่อสัญญาณแสงได้อัดฉีดเข้าไปในสายเคเบิ้ลนั้นมันก็จะผ่านไปรอบๆ ริงค์จนกว่ามันจะถูกลดทอนหมด ดังนั้นถ้าหากเราไม่สามารถที่จะนำเอาสัญญาณก่อนๆ ออกมาใช้ก็จะเกิดมีแอคโค่ขึ้น



รูปที่ 2.9 แสดงโครงข่ายไฟเบอร์ออฟติคริงค์

แบบสตาร์

สำหรับรูปร่างของเจตเวอร์คแบบสตาร์นี้ ตัวอย่างที่เราสามารถเห็นได้ง่ายๆ ก็คือระบบโทรศัพท์นั่นเอง ซึ่งแสดงได้ดังรูป 2.10 ซึ่งจะประกอบด้วยสถานีศูนย์กลางซึ่งใช้ควบคุมระบบ สำหรับแอคทีพสตาร์นั้นสถานีศูนย์กลางจะทำงานเป็นสวิชชิ่งหรือมีเดียแอคเซส ส่วนแพสซีพสตาร์นั้นจะทำงานด้วย nxn คอปเลอร์ การทำงานของเนตเวอร์คแบบสตาร์นั้นสถานีศูนย์กลางจะเป็นศูนย์กลางควบคุม ซึ่งจะถูกเชื่อมต่อไปยังทุกสถานี ดังนั้นถ้าหากจำนวนของสถานีเพิ่มขึ้น ก็จำเป็นที่จะต้องเพิ่มค่าใช้จ่ายในการก่อสร้างคู่สายใยแสง และนอกจากนี้ยังเพิ่มความยุ่งยากให้แก่อาร์ดแวร์ของสถานีศูนย์กลางอีกด้วย



รูปที่ 2.10 แสดงโครงข่ายแบบสตาร์

บทที่3

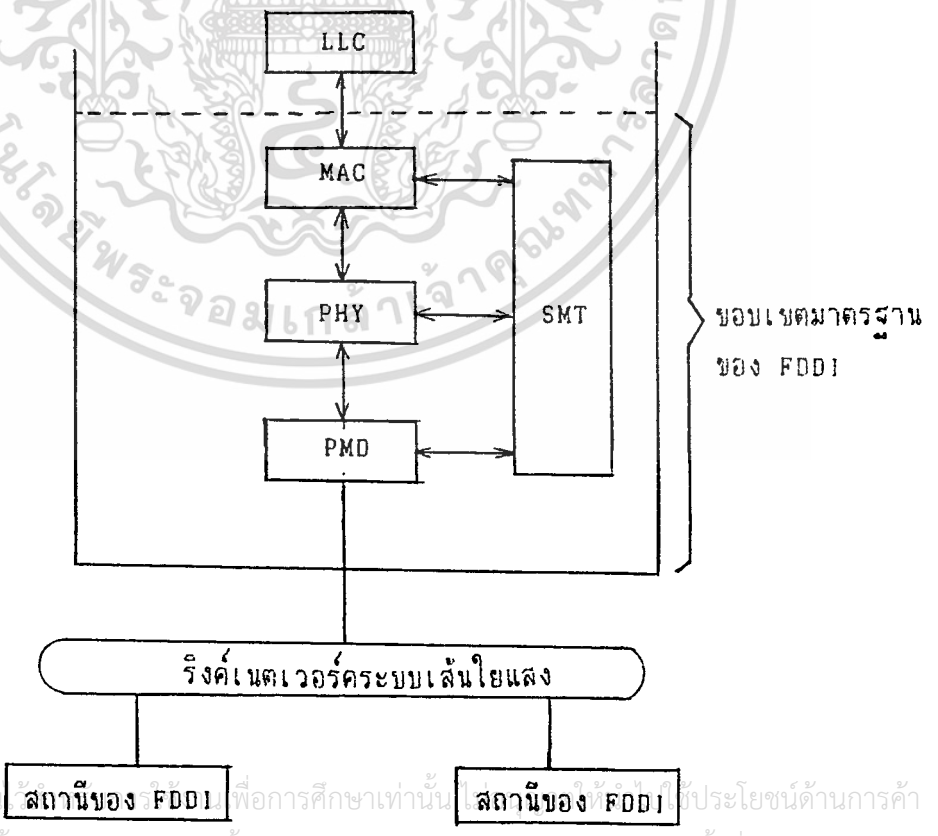
FDDI(Fiber Distributed Data Interface)

3.1 ขอบเขตมาตรฐานของ FDDI

มาตรฐานของ FDDI นั้นประกอบด้วย ระดับชั้น MAC(MAC Layer) และระดับชั้นฟิสิคัล(Physical Layer) ซึ่งโครงสร้างของ FDDI นั้นกำหนดให้ใช้มาตรฐานของ IEEE802.2 ลอจิคัลลิงค์คอนโทรล (Logical Link Control) มาตรฐานนั้นประกอบด้วยสี่ส่วนดังนี้

1. มีเดียมแอคเซสคอนโทรล (Mediam Access Control)
2. ฟิสิคัลโปรโตคอล (Physical Protocol)
3. ฟิสิคัลมีเดียมดีเพนเดนซ์ (Physical Mediam Dependent)
4. สเตชั่นแมนเนจเมนต์ (Station Management)

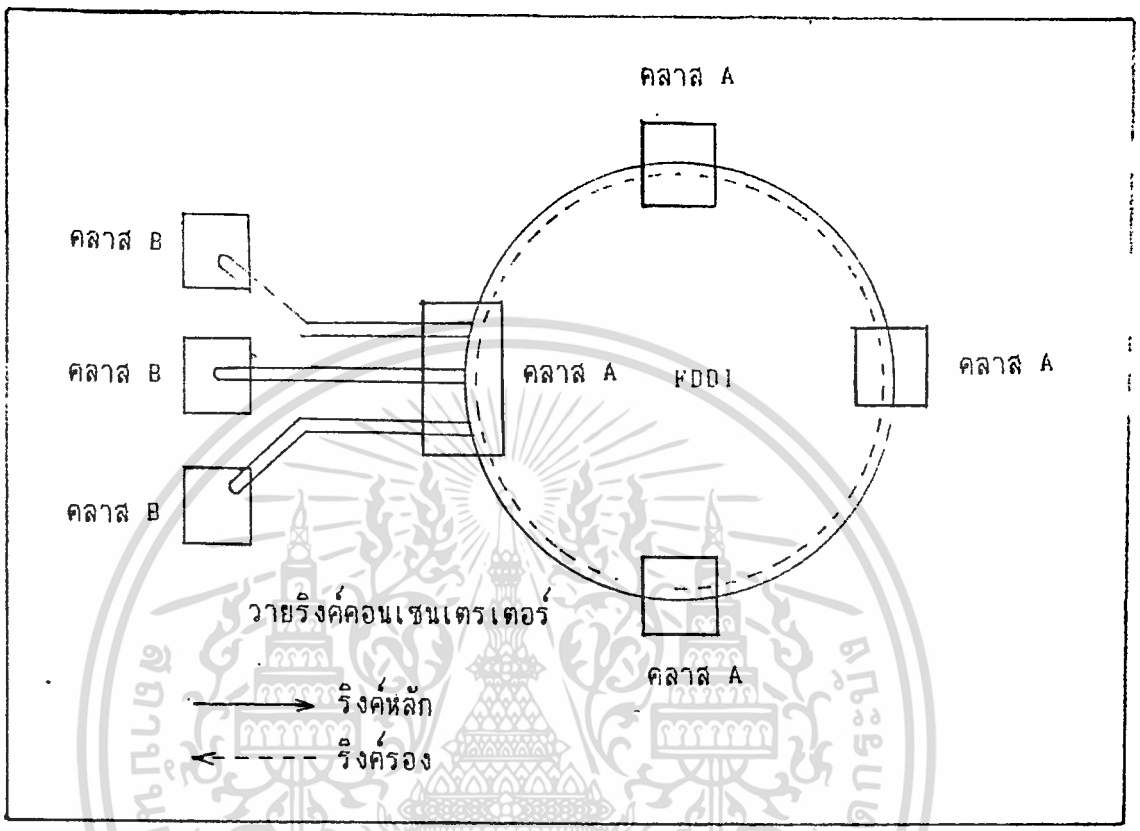
ซึ่งโครงสร้างแสดงดังรูป 3.1



รูปที่ 3.1 แสดงโครงสร้างของ FDDI

สำหรับMACเป็นส่วนหนึ่งของเดต้าลิงค์ จะเป็นตัวกำหนดโครงสร้างของเฟรมแอดเดรส การควบคุมความผิดพลาด นอกจากนี้ยังเป็นตัวควบคุมระบบการส่งแบบระบบโทเกน และเพื่อให้ระบบการจัดการข้อมูลด้วยความเร็วสูง ข้อมูลจะถูกจัดการทีละ 4 บิท หรือ 8 บิท ในโครงสร้างของ FDDI นั้นส่วนของฟิสิคัลได้ถูกแยกออกเป็นสองส่วนย่อยคือ PHY กับ PMD โดยที่ PHY จะรับข้อมูลจาก MAC เปลี่ยนเป็นระหัสแล้วส่งให้ PMD หรือจาก PMD ไปยัง MAC สำหรับข้อมูลที่ถูกเข้าระหัสนี้ จะใช้เพื่อเพิ่มข้อมูลการกำหนดจังหวะของข้อมูลจาก MAC สำหรับ IEEE802.5 นั้นใช้ระหัสดิฟเฟอเรนเชียลแมนเชสเตอร์ (Differential Manchester) ซึ่งระหัสนี้สองบิทจะแทนข้อมูลหนึ่งบิท ดังนั้นในการส่งข้อมูล 100 Mbps จำเป็นต้องใช้ความเร็วสัญญาณ 200 MHz

ส่วนของ PMD นั้นจะเป็นเพียงส่วนล่างของชั้นฟิสิคัล ซึ่งจะกำหนดความยาวคลื่นแสงและกำลังส่งออก เป็นต้น ส่วนของ SMT นั้นมีไว้เพื่อให้เนตเวอร์คสามารถปฏิบัติงานได้อย่างถูกต้อง ซึ่ง FDDI ได้ระบุการบำรุงรักษา ตัวอย่างเช่นวิธีการแยกชิ้นส่วนที่มีปัญหาการจัดการข้อมูลที่ผิดพลาด และการกำหนดการบริการแบบซิงโครนัสและอะซิงโครนัส เป็นต้น ในมาตรฐานนี้ได้ระบุสถานีไว้สองชนิดคือ สถานีคลาส-A ซึ่งมีการเชื่อมต่อสองทางส่วนสถานีคลาส-B นั้นมีการเชื่อมต่อทางเดียว และจะถูกใช้ร่วมกับตัววายริงค์คอนเซนเตรเตอร์ (Wiring Concentrator) ริงค์เนตเวอร์คของ FDDI นั้นจะประกอบด้วยริงค์ที่ใช้เป็นตัวหลักและตัวรอง แต่ริงค์ตัวรองนั้นใช้สำหรับเสริมเท่านั้น สถานีคลาส-A นั้นจะต่อเชื่อมกับริงค์ทั้งสองริงค์ ในขณะที่สถานีคลาส-B นั้นจะเชื่อมต่อกับริงค์ตัวหลักเท่านั้น ดังรูป 3.2 แสดง FDDI ริงค์เนตเวอร์ค



รูปที่ 3.2 แสดงโครงสร้างของ FDDI ริงค์เนตเวอร์ค

จากรูป FDDI ริงค์เนตเวอร์คประกอบด้วยสถานีคลาส-A และคลาส-B สำหรับวายริงค์คอนเซนเตรเตอร์นั้นจะเป็นกรณีพิเศษของสถานีคลาส-A เท่านั้นโดยมันจะเชื่อมต่อทั้งริงค์หลักและริงค์รอง แต่จะเชื่อมริงค์หลักให้กับสถานีคลาส-B เท่านั้น สถานีจะถูกต่อกับสายเคเบิลแบบดูเพล็กซ์ไฟเบอร์ (Duplex Fiber) ทั้งสองไฟเบอร์จะถูกสวมอยู่ในแพคเกตเดียว ฮาร์ดแวร์ที่เหมือนกันจะถูกต่อเชื่อมจากคลาส-A ไปยังคลาส-A หรือจากคลาส-A ไปยังตัวคอนเซนเตรเตอร์ หรือจากคอนเซนเตรเตอร์ไปยังคลาส-B ถ้าหากสายเคเบิลเกิดเสียหรือว่าสถานีเกิดเสียขึ้น FDDI เนตเวอร์คก็จะปฏิบัติการเปลี่ยนโครงสร้างเนตเวอร์คดังรูป 3.3

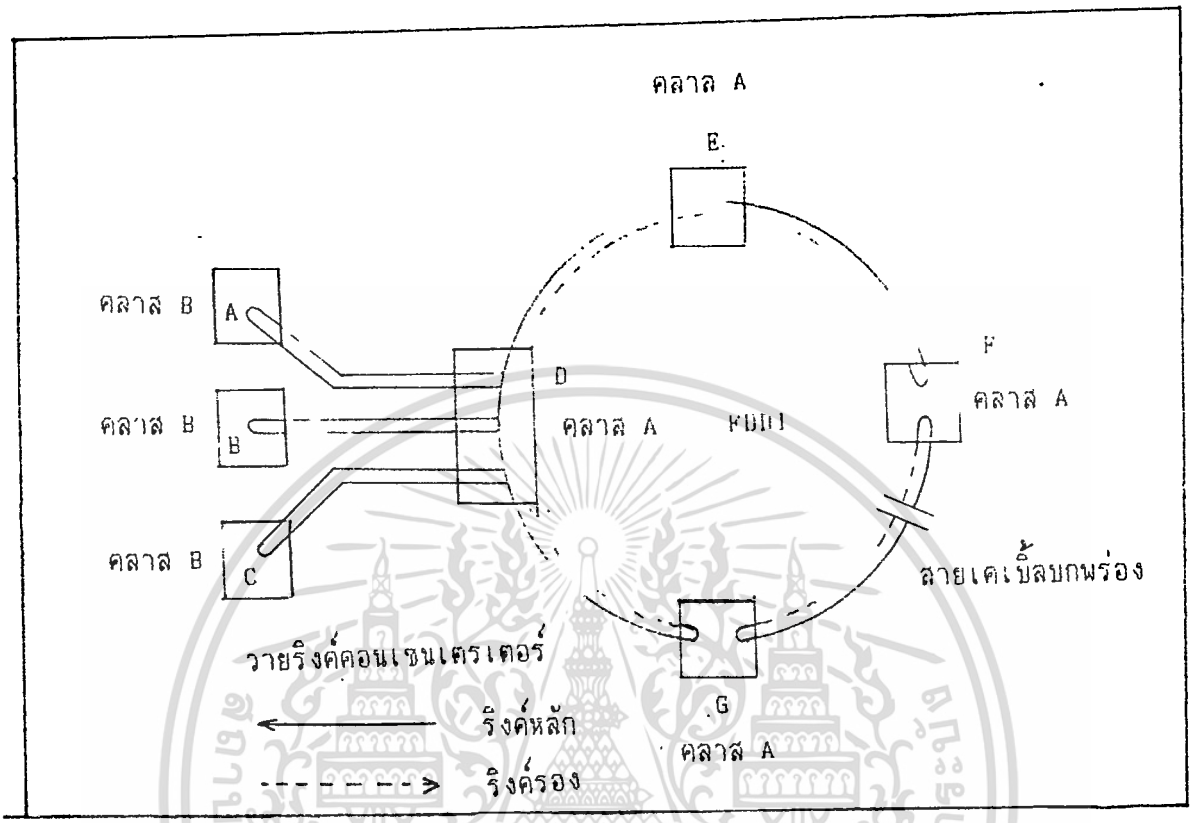
3.2 มีเดียแอคเซสคอนโทรลของ FDDI

ในเวลาเริ่มต้นเนตเวอร์ไม่ได้อยู่ในลักษณะริงค์ มันจะแยกออกเป็นส่วนย่อยๆ ประกอบด้วยชั้นฟิสิคัลสองชั้น และ MAC อีกหนึ่งหรือมากกว่า หลังจากที่ปรับเนตเวอร์คเรียบร้อยแล้วโทเกนก็จะถูกสร้างขึ้น เพื่อความน่าเชื่อถือจะมีสถานีเดียวเท่านั้นที่มีโอกาสสร้างโทเกน สำหรับเวลาสูงสุดที่ต้องการให้โทเกนวนรอบริงค์นั้น ได้ถูกกำหนดโดยสถานีซึ่งกำหนดให้ค่านั้นมีค่าน้อยที่สุดสำหรับสถานีที่ถูกกำหนดให้สร้างโทเกน สถานีพบว่าริงค์ควรจะมีการปรับค่าใหม่ และเข้าสู่สภาวะเรียกร้อง ในสภาวะนี้จะมีการส่งแพคเกตเรียกร้องอย่างต่อเนื่อง

สำหรับแพคเกตเรียกร้องนี้จะประกอบด้วยสตาร์ทดิลิมิเตอร์ (Start Delimeter) แล้วต่อด้วยส่วนควบคุม ส่วนที่อยู่รับ-ส่งนั้นจะเหมือนกันคือของผู้ส่งแพคเกต ส่วนที่เป็นข้อมูลนั้นจะประกอบด้วยค่าเป้าหมายของโทเกนวนรอบริงค์ (TTRT) เมื่อแพคเกตนี้ถูกตรวจสอบด้วยสถานีอื่นที่ส่งแพคเกตเรียกร้องด้วยค่าของ TTRT ในแพคเกตจะถูกเปรียบเทียบกับของสถานี ค่า TTRT มีค่าน้อยกว่าจะเป็นค่าสิทธิการเรียกร้องที่เหนือกว่าสถานีที่รับค่า TTRT น้อยกว่านะหยุดส่งแพคเกตเรียกร้องออกไป ถ้าหากว่าค่าของ TTRT เท่ากันสถานีก็จะเปรียบเทียบค่าแอดเดรส เพื่อตัดสินผู้ที่จะได้ส่งแพคเกตเรียกร้องต่อไป ทุกสถานีจะต้องจำค่า TTRT ของแพคเกตเรียกร้องสุดท้าย และแล้ววิธีการปฏิบัติการเรียกร้องก็สิ้นสุด ค่า TTRT นี้จะเป็นค่าปรับใช้ในการส่งแพคเกต จากนั้นสถานีที่ชนะการส่งซึ่งโดยการรับโทเกนของมันคืนมาแล้ว มันก็จะผลิตโทเกนเพื่อใช้ในการปรับค่าเริ่มแรกของริงค์ วิธีการเรียกร้องการส่งนี้จะเป็นการตัดสินถึงความถี่ที่จะได้รับโทเกนของแต่ละสถานี และเพื่อที่จะรับรองว่าสถานีนั้นสามารถรับโทเกน และส่งข้อมูล FDDI ได้แบ่งการบริการออกเป็น 2 แบบคือการบริการแบบซิงโครนัส และแบบอะซิงโครนัส อะซิงโครนัสจะเริ่มต้นส่งถ้าหากโทเกนมาถึงก่อนค่า TTRT ที่กำหนด ส่วนซิงโครนัสนั้นสามารถส่งได้ตลอดเวลาที่ได้รับโทเกน กรณีที่เลวร้ายที่สุดของเวลาที่ใช้ส่งแบบอะซิงโครนัสนั้นไม่เกินกว่า TTRT จะเห็นว่าวิธีการส่งแบบไทม์โทเกนโปรโตคอลของ FDDI นี้ เป็นการใช้แบนด์วิดธ์ที่ว่างเปล่า จากการส่งซิงโครนัสไปให้การส่งแบบอะซิงโครนัส ซึ่งจะเป็นการหลีกเลี่ยงการส่งอะซิงโครนัสในปริมาณที่คงที่ วิธีการแบ่งสรรแบนด์วิดธ์นี้ จะตัดสินอัตราการส่งของซิงโครนัสของแต่ละสถานี ซึ่งเรียกร้องแบนด์วิดธ์การส่งซิงโครนัสนั้นจะถูกกำหนดตอนปลายเช่นกัน จากนั้นก็จะสร้างเวลาสูงสุดที่สถานี

นั้นสามารถส่งแบบซิงโครนัส ปัญหาที่เกิดขึ้นบ่อยๆ ใน FDDI คือการถอดสถานีออกจากริงค์ซึ่งจะทำให้สายโดนตัดหรือเคลื่อน ถึงแม้ปัญหานี้จะแก้ไขโดยการเพิ่มริงค์สำรองก็ตาม ถ้าหากสถานีถูกต่อโดยรีเลย์ที่ใช้เปลี่ยนทาง การปิดเครื่องสามารถทำให้คลื่นแสงพุ่งไปถึงสถานีข้างเคียง แต่ถ้าหากไม่มีการติดตั้งการเปลี่ยนเส้นทางก็ไม่เกิดขึ้น ถ้าหากเปรียบเทียบวิธีนี้กับวิธีของ IEEE802.6 การอยู่หรือไม่อยู่ของสถานีจะไม่เป็นปัญหา นอกเสียจากว่าแพคเกตไม่สามารถถูกสร้างที่สถานีนั้น สายบกพร่องสามารถถูกพบโดยฝ่ายจัดการที่สถานีนั้นๆ ทั้งสองด้าน หรือโดยสถานีตรวจสอบ MAC ของสถานีสามารถตรวจสายบกพร่องได้โดยการเช็คช่วงเวลาเงียบ และขึ้นอยู่กับสภาวะของ MAC ซึ่งจะตัดสินว่าเกิดความผิดพลาดที่โทเกนโดยที่โทเกนได้ล่าช้ามาแล้ว และไทม์เมอร์ที่ใช้วัดการวนรอบของโทเกนได้หมดเวลาแล้ว ถ้าหาก MAC ได้เห็นโทเกนก่อนที่จะเกิดสายบกพร่อง มันอาจจะสรุปว่าอาจจะเป็นก่อนการส่งไม่ถูกต้อง ซึ่งสามารถวัดโดยไทม์เมอร์ที่ใช้แพคเกตพิเศษ

ทั้งสองสถานี MAC จะพยามแก้ไขโดยการเริ่มการปฏิบัติการเรียกร้อง สำหรับการปรับโครงสร้างเนตเวอร์ค ถ้าหากการดำเนินการไม่สามารถแก้ไขได้ การเรียกก็จะล้มเหลวในขณะที่มีการปฏิบัติการเรียกร้องการทำงาน จะใช้วิธีการใช้ริงค์สำรองดังแสดงในรูป 3.3 ถ้าหากการปฏิบัติการเรียกร้องไม่สำเร็จ MAC จะเริ่มการปฏิบัติการเตือนภัยในสภาวะนี้ MAC จะส่งแพคเกตพิเศษที่เรียกว่าแพคเกตเตือนภัย ถ้าหากสถานีนั้นสามารถรับแพคเกตเตือนภัยของมัน มันก็จะตระหนักว่าการบกพร่องได้ถูกแก้ไขแล้วและมันก็จะเริ่มต้นการปรับค่าของริงค์ใหม่ ถึง FDDI โปรโตคอลจะอาศัยหลักการของ 802.5 โปรโตคอล แต่ปรากฏว่าหลายอย่างที่เหมือนกันโดยที่ชั้น MAC และชั้นฟิสิคัลนั้นไม่ได้ถูกออกแบบใช้งานได้กับของ 802 อย่างไรก็ตาม FDDI ได้ยอมรับ LLC ของ 802 ดังนั้นการเชื่อมต่อของเนตเวอร์คจะเป็นไปได้โดยอาศัยบริดจ์



รูป3.3 แสดงการเปลี่ยนโครงสร้างของ FDDI

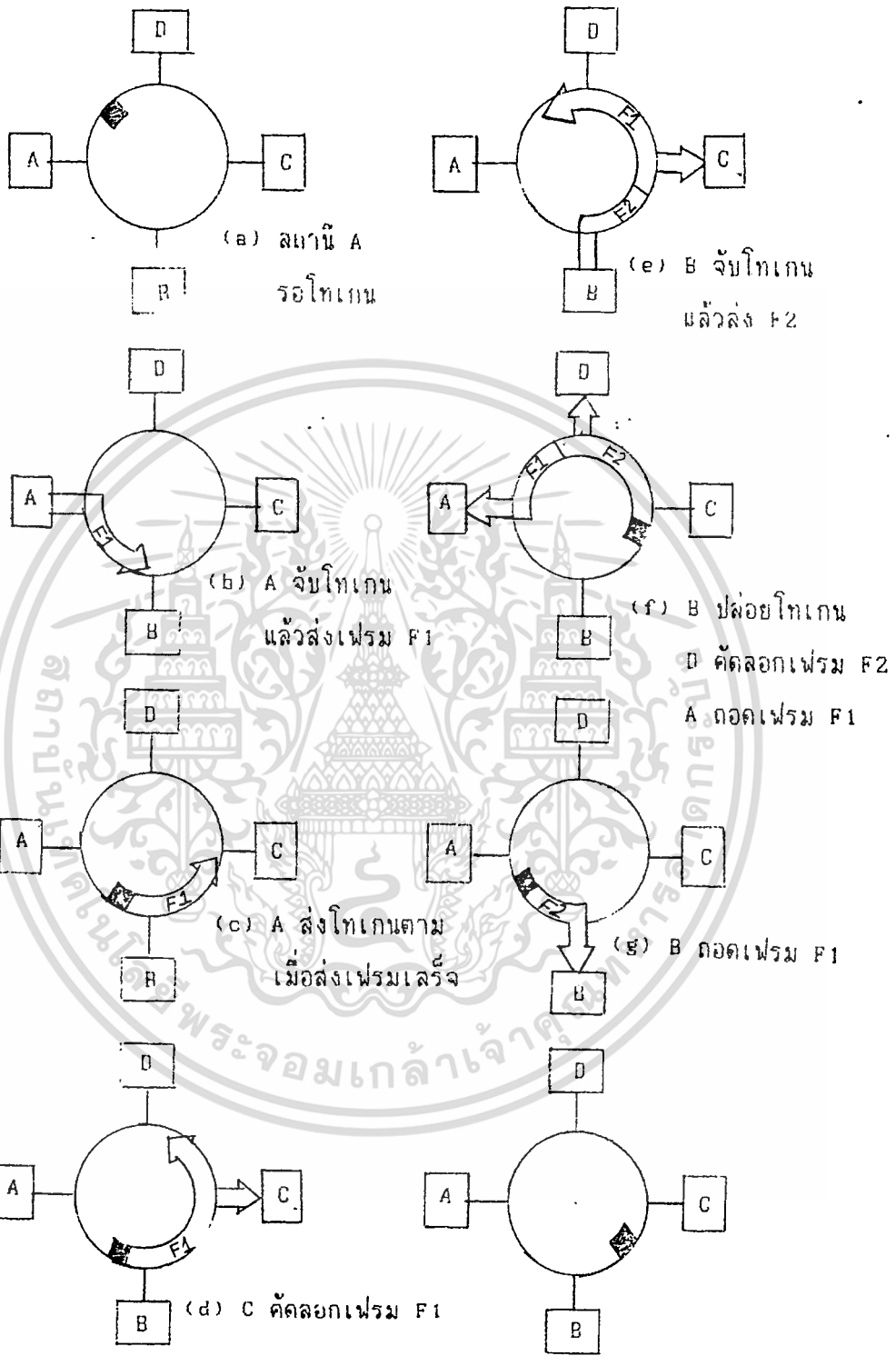
3.3 โทเกนริงค์ (Token Ring)

FDDI MAC โปรโตคอลนั้นจะเป็นแบบโทเกนริงค์เช่นเดียวกับ IEEE802.5 การจัดการขั้นต้นของโทเกนริงค์ จะคล้ายกับ IEEE802.5 มากในบทนี้จะกล่าวถึงการจัดการเบื้องต้นและจุดที่แตกต่างกันของโปรโตคอลทั้งสอง

เทคนิคของ FDDI โทเกนริงค์นั้นจะใช้โทเกนเฟรมเล็กๆ วนไปรอบๆ ริงค์ เมื่อสถานีว่างสถานีที่ต้องการที่จะส่งข้อมูลจะต้องรอ จนกระทั่งมีโทเกนผ่านมามันจะต้องยึดโทเกนไว้ และหลังจากนั้นก็จะเริ่มส่งข้อมูลหนึ่งเฟรมหรือมากกว่านั้น ซึ่งในขณะนั้นจะไม่มีโทเกนวนอยู่ในริงค์ สถานีนี่นๆ ที่ต้องการจะส่งจะต้องรอไปก่อน เฟรมข้อมูลที่ส่งเข้าไปในริงค์จะต้องวนไปจนครบรอบ และจะถูกปรับปรุงโดยสถานีส่ง สถานีส่งจะปล่อยโทเกนเข้าไปในริงค์ใหม่เมื่อมันส่งเฟรมไปเรียบร้อยแล้ว ถ้าหากความยาวบิทของริงค์มากกว่าความยาวบิทที่สถานีส่งไป ดังนั้นโทเกนที่ส่งเข้าไปใหม่นี้จะปรากฏก่อนส่วนสุดท้ายของ

เคอเรนท์เฟรม (current freme) จะกลับมาถึงเครื่องส่งของมันเพื่อปรับปรุง

รูปที่ 3.4 แสดงตัวอย่างการจัดการของริงค์ หลังจากสถานี A จับโทเกนได้แล้วมันจะส่งเฟรมสมมติเป็นเฟรม F1 และในทันทีทันใดนั้นมันก็จะส่งโทเกนเข้าไปในริงค์ทันที ซึ่งข้อมูลในเฟรม F1 นั้นจะส่งให้กับสถานี C ซึ่งสถานี C จะทำการคัดลอก F1 ไว้ เมื่อมันเคลื่อนผ่านมา แล้วเฟรม F1 ก็จะวนกลับมายัง A สถานีก็จะทำการถอดเฟรมนั้นออกจากริงค์ ขณะที่สถานี B จับโทเกนซึ่งสถานี A ปล่อยมาก็จะทำการส่งเฟรม F2 ไป ซึ่งจะทำเช่นนี้ไปเรื่อยๆ ดังนั้นในริงค์จะประกอบด้วยเฟรมหลายๆ เฟรม สถานีเหล่านี้จะเป็นผู้ถอดเฟรมของมันออกจากริงค์เอง เมื่อเฟรมวนกลับมาถึง ข้อแตกต่างระหว่าง 802.5 และ FDDI มีดังนี้คือประการแรกสถานีของ FDDI ไม่ได้พยายามที่จะยึดโทเกนโดยการฟลิปปิ้งบิท (flipping bit) เพราะอัตราข้อมูลของ FDDI สูงมันเห็นว่าไม่มีประโยชน์ที่จะทำตามวิธีนี้ ประการที่สองใน FDDI สถานีจะปล่อยโทเกนโดยทันทีที่ส่งเฟรมเสร็จโดยไม่ต้องรอให้เฟรมที่มันส่งไปวนกลับมาถึงก่อน เพราะอัตราข้อมูลสูง วิธีการเช่นนี้จึงเหมาะอย่างยิ่ง

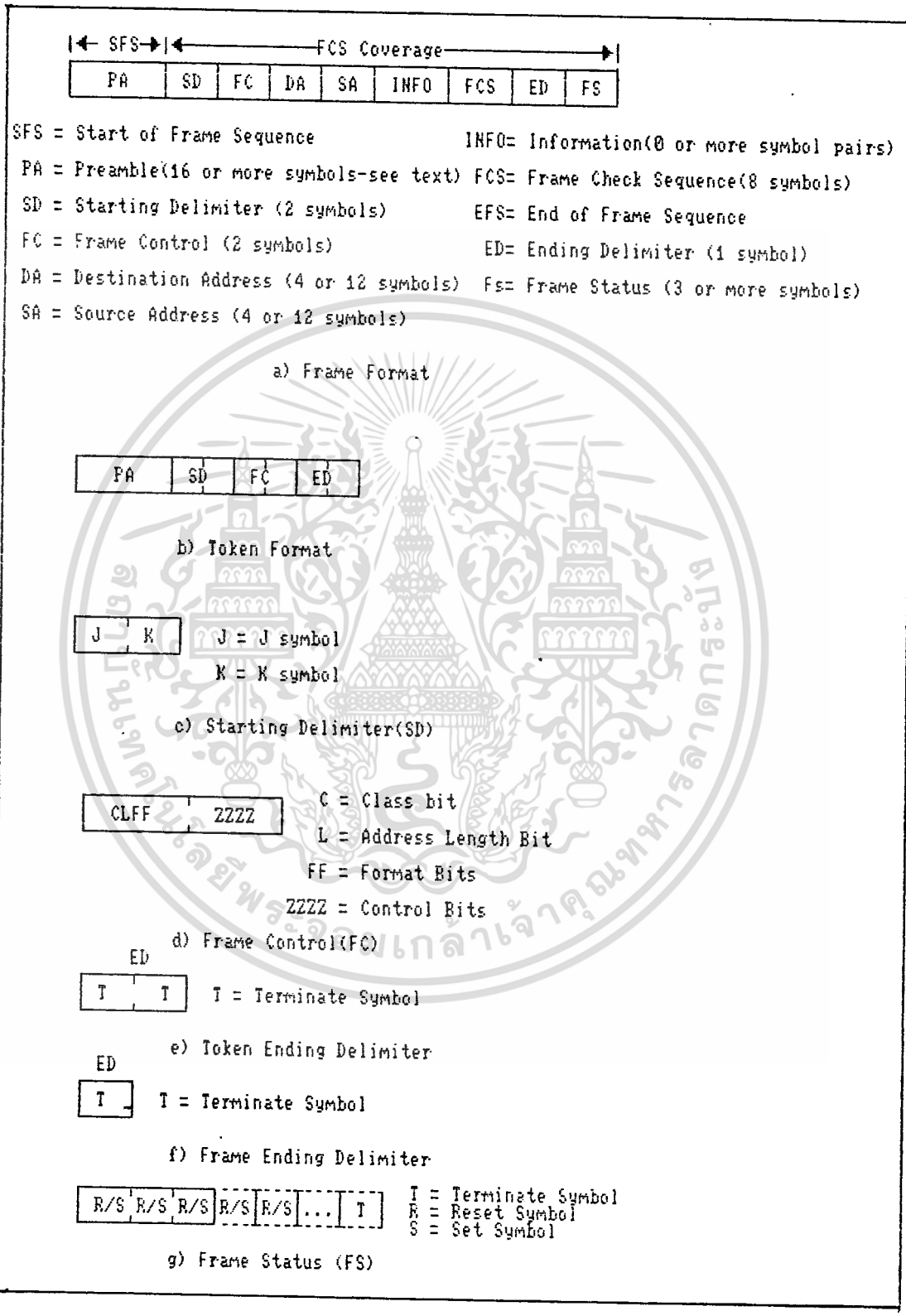


รูปที่ 3.4 แสดงการจัดการของ FDDI ริงค์เนตเวอร์ค

3.4 MAC เฟรม

จากหัวข้อ 3.2 ได้กล่าวถึงรายละเอียดของ มีเดียมแอคเซสคอนโทรลของ FDDI มาแล้ว ในหัวข้อนี้จะกล่าวถึงโครงสร้างของเฟรม ซึ่งมาตรฐานของโครงสร้างเฟรมนี้จะอยู่ในรูปของซิมโบล (symbol) โดยแต่ละซิมโบลจะประกอบด้วยสี่บิท ที่กำหนดเช่นนี้เพราะในระดับชั้นฟิสิคัลนั้น ข้อมูลจะถูกส่งอยู่ในรูปของ four-bit chunks คือจะจัดการข้อมูลทีละสี่บิท อย่างไรก็ตามรูปแบบของ MAC ตามความเป็นจริงแล้วจะต้องติดต่อกับบิทเฉพาะ (Individual bits) ดังรูป 3.5 แสดงรูปแบบ (format) ของเฟรมซึ่งสร้างโดย FDDI โปรโตคอลรูปแบบของเฟรมทั้งหมด จะประกอบด้วยฟิลด์ต่างๆ ดังต่อไปนี้

1. **พรีเอ็มเบิล** (Preamble) ทำหน้าที่ซิงโครไนซ์ระหว่างเฟรมกับคล็อคของสถานีเป็นส่วนเริ่มต้นของเฟรม ประกอบด้วย 16 Idle symbols (64 บิท) ซึ่ง Idle symbol นี้เป็นฟิลด์แพทเทิร์นที่ไม่มีข้อมูล
2. **สตาร์ติ้งดีลิมิเตอร์** (Starting Delimiter) เป็นส่วนแจ้งส่วนเริ่มต้นของเฟรม ประกอบด้วยซิกแนลลิ่งแพทเทอร์น (Signaling Pattern)
3. **เดสติเนชั่นแอดเดรส** (Destanation Address) เป็นส่วนกำหนดสถานีปลายทาง อาจจะมีเพียงสถานีเดียวหรือหลายๆ สถานีก็ได้ ประกอบด้วย 16 บิท หรือ 48 บิท
4. **ซอสแอดเดรส** (Source Address) ระบุสถานีที่ส่งเฟรม
5. **ข่าวสารข้อมูล** (Informations) จะบรรจุ LLC เดต้าหรือข่าวสารข้อมูล
6. **เฟรมเช็คซีเควนซ์** (Frame Check Sequence) สำหรับตรวจสอบความผิดพลาด ประกอบด้วย 32 บิทรีดันแดนซีเช็ค (Redundancy Check)
7. **เอนดิ้งดีลิมิเตอร์** (Ending Delimiter) บรรจุซิมโบลที่ไม่มีเดต้าสำหรับแจ้งการสิ้นสุดของเฟรม
8. **เฟรมสเตตัส** (Frame Status) บรรจุเออเรอร์ดีเทค (Error Detected) เฟรมก็อปปี้ (Frame Copied)



รูปที่ 3.5 แสดงรูปแบบของเฟรมต่างๆ

บทที่ 4

การวิเคราะห์การทำงานของ FDDI โดยวิธีคอมพิวเตอร์สิมิวเลชั่น (Simulation)

สิมิวเลชั่น คือการจำลองหรือเลียนแบบเหตุการณ์ต่างๆ ซึ่งบางครั้งเหตุการณ์นั้นไม่สามารถที่จะทดลองด้วยของจริงได้ การจำลองเหตุการณ์สามารถทำได้หลายๆ แบบ เช่น การใช้สมการทางคณิตศาสตร์ การใช้โปรแกรมคอมพิวเตอร์ (Software Simulation) การใช้คอมพิวเตอร์ในการจำลองเหตุการณ์นั้น เราสามารถที่จะทำได้ทุกเหตุการณ์ จะประสบผลสำเร็จมากน้อยแค่ไหนนั้นขึ้นอยู่กับว่า ผู้ที่เขียนโปรแกรมมีความเข้าใจเกี่ยวกับระบบหรือเหตุการณ์ที่จำลองมากน้อยแค่ไหน เพราะว่าการจำลองเหตุการณ์จริงจริงนั้นมีพารามิเตอร์มากมาย ซึ่งก็มีหลายเหตุการณ์ยากแก่การจำลองให้เหมาะสม

จุดมุ่งหมายของวิธีการคอมพิวเตอร์สิมิวเลชั่น มีดังนี้คือ

1. ใช้สำหรับพิสูจน์วิธีคำนวนทางคณิตศาสตร์
2. ใช้ทดลองกับโมเดลยากๆ ที่วิธีการคำนวนทางคณิตศาสตร์ไม่สามารถทำได้

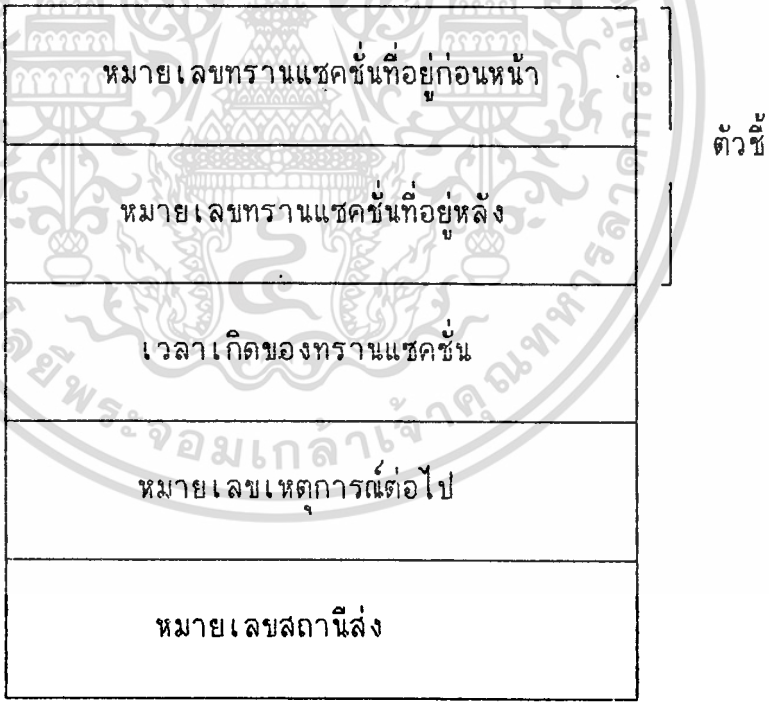
4.1 เหตุผลที่ต้องมีการจำลองเหตุการณ์

1. ในการจำลองเหตุการณ์เราสามารถเปลี่ยนพารามิเตอร์ต่างๆ เพื่อดูผลที่เกิดขึ้นได้ ถึงแม้ว่าในเหตุการณ์จริงแล้ว การทำเช่นนั้นอาจทำให้เสียค่าใช้จ่ายมาก หรืออาจเกิดอันตรายขึ้นได้ ตัวอย่าง เช่น การจำลองพลังงานนิวเคลียร์ เพื่อดูผลการเกิดการผิดพลาดโดยปราศจากอันตราย

2. การจำลองเหตุการณ์ สามารถสร้างเหตุการณ์ที่ไม่สามารถเกิดขึ้นได้ ในเหตุการณ์จริง ตัวอย่าง เช่น นักจิตวิทยาต้องการจะศึกษาการพัฒนาทางด้านการเรียนรู้ของหนู โดยให้หนูในบริเวณที่วกวน หรือเขาวงกต และดูว่าเส้นทางใดที่หนูวิ่งได้เร็วที่สุด .ถึงแม้ว่า เราไม่สามารถทดลองได้จริงๆ แต่เราก็สามารถดูผลได้จากการจำลองเหตุการณ์

4.2 หลักการสิมิวเลชั่น

ในทางปฏิบัติเราไม่สามารถที่จะเขียนโปรแกรมจำลองเหตุการณ์ ให้เหมือนกับเหตุการณ์หรือการทดลองจริงๆ ได้ แต่เราสามารถที่จะจำลองให้ใกล้เคียงมากที่สุด โดยมีหลักการคือ ในระบบจริงๆ นั้นเรามีสถานีเป็นตัวรับส่งแพคเกตทั้งหมด แต่ในการเขียนโปรแกรมนั้น เราจะใช้ทรานแซคชั่น (Transaction) แทนแพคเกต โดยการอาศัยคำสั่งของภาษาคอมพิวเตอร์ เพื่อปฏิบัติการกับทรานแซคชั่น ให้เข้าสู่เหตุการณ์ ซึ่งจำลองการเคลื่อนไหวของแพคเกตในระบบเนตเวอร์ค ทรานแซคชั่นนี้เมื่อสร้างขึ้นแล้วจะถูกเรียงในลูกโซ่เวลา (Timming Chain) ตามลำดับเวลาน้อยไปหามาก โดยแต่ละทรานแซคชั่นมีโครงสร้างดังรูป 4.1



รูป 4.1 โครงสร้างของทรานแซคชั่น

จากรูปจะเห็นว่า แต่ละทรานแซคชั่นนั้นจะแทนแพคเกตอันหนึ่ง ประกอบด้วยตัวแปรที่เป็นจำนวนจริง แต่ละทรานแซคชั่นจะถูกเรียงตามลำดับเวลา ที่จะเข้าสู่เหตุการณ์ต่อไป แต่ละทรานแซคชั่นจะถูกเชื่อมต่อกันโดยอาศัยตัวชี้ (Pointer) โปรแกรมสำหรับการจำลองเหตุการณ์นั้นประกอบด้วย โปรแกรมหลัก และโปรแกรมย่อยอีกหลายๆ กลุ่ม โดยโปรแกรมหลักจะเป็นตัวดึง ทรานแซคชั่นออกมาจากลูกโซ่เวลาเพื่อมาจัดการตามเหตุการณ์ที่ระบุไว้ในทรานแซคชั่น

4.3 การทำงานของโปรแกรม

เริ่มจากโปรแกรมหลัก (Main) เรียกโปรแกรมย่อย Generate() เพื่อเป็นการกำหนดเวลาการเกิดของแพคเกตแต่ละสถานี การกำหนดเวลาการเกิดนี้ ได้มาจากการคำนวณทางคณิตศาสตร์

$$\text{เวลาเกิดของแพคเกต} = -\log f \times 1/\lambda$$

การเกิดของแพคเกตเป็นแบบปัวซอง (Poisson) เมื่อเรากำหนดให้ ρ เป็นอัตราเวลาที่ชันแนลไม่ว่าง (Busy) หรือจะเรียกว่า ความเข้มทราฟฟิค (Traffic Intensity) จะได้ว่า

$$\rho = \lambda Y \quad \text{โดย } 0 < \rho < 1$$

ซึ่ง λ = ค่าเฉลี่ยอัตราแพคเกตเข้าระบบ (แพคเกต/วินาที)

Y = ค่าเฉลี่ยเวลาบริการต่อแพคเกต (วินาที)

ค่า f ได้จากการสุ่มค่าเวลา (Random)โดยใช้โปรแกรมย่อย ran3() เวลาเกิดของแพคเกตเหล่านี้จะถูกเรียงลำดับ เพื่อรอการเกิด ซึ่งในทางโปรแกรมทำได้โดยเรียงแพคเกตตามลำดับเวลาการเกิด เรียกว่า ลูกโซ่เวลา (Timming Chain) โปรแกรมย่อย insert_list() จะทำหน้าที่เรียงลำดับค่าเวลาเกิดของแพคเกตจากน้อยไปหามาก

เพื่อที่แพคเกตจะถูกดึงออกมาใช้งาน ตามลำดับ ก่อน-หลังอย่างถูกต้อง

สำหรับแต่ละแพคเกต ในโปรแกรมนี้จะถูกแทนด้วย ทรานแซคชั่น (Transaction) ซึ่งในทรานแซคชั่นจะกำหนดเหตุการณ์ต่างๆ ที่แพคเกตนั้นๆ ต้องกระทำ และเก็บค่าเวลาการทำงานของแต่ละแพคเกตไว้ เพื่อให้แพคเกตทำงานอย่างถูกต้อง จาก โปรแกรม Set_trans() จะเห็นว่า ทรานแซคชั่นของแพคเกต ถูกแบ่งออกเป็นส่วนๆ เพื่อเก็บค่าต่างๆ ดังนี้

ส่วนที่ 1 เก็บค่าหมายเลขแพคเกตที่ถูกดึงมาจัดการก่อนตัวมัน

ส่วนที่ 2 เก็บค่าหมายเลข แพคเกตที่ถูกดึงมาจัดการหลังตัวมัน

ส่วนที่ 3 เก็บค่าหมายเลขแพคเกตของตัวมันเอง

ส่วนที่ 4 เก็บค่าเวลาที่ แพคเกตต้องไปทำเหตุการณ์ต่างๆ

ส่วนที่ 5 กำหนดเหตุการณ์ต่อไปที่แพคเกตต้องกระทำ

ส่วนที่ 6 เก็บค่าสถานีที่แพคเกตเกิด

ในโปรแกรมย่อย Set_tok() เป็นการกำหนดเหตุการณ์ต่างๆ ที่โทเกน จะต้องกระทำและเวลาการทำงานของโทเกน เช่นเดียวกับแพคเกต ทรานแซคชั่นของโทเกนจึงถูกแบ่งออกเก็นส่วนๆ

ส่วนที่ 1 เก็บค่าหมายเลขแพคเกต ที่ถูกดึงออกมาจากลูกโซ่ของเวลา เพื่อออกมาจัดการก่อนโทเกน

ส่วนที่ 2 เก็บค่าหมายเลขแพคเกต ที่ถูกดึงออกมาจากลูกโซ่ของเวลาเพื่อออกมาจัดการหลังโทเกน

ส่วนที่ 3 เก็บค่าหมายเลขคือ "0" เนื่องจากโปรแกรมกำหนดให้หมายเลข "0" แสดงถึงทรานแซคชั่นนั้น คือโทเกน

ส่วนที่ 4 เก็บค่าเวลาที่โทเกน ต้องไปทำเหตุการณ์ต่างๆ นั่นคือเวลาที่โทเกนวนไปถึงแต่ละสถานี

ส่วนที่ 5 กำหนดเหตุการณ์ที่โทเกนต้องกระทำ

ส่วนที่ 6 กำหนดสถานีต่อไปที่โทเกนจะไปถึง

4.3.1 การทำงานของโปรแกรมย่อย event_1

การทำงานของโปรแกรมย่อย event_1() แพคเกตที่เกิด จะถูกนำเข้าไปรอการส่งในคิว นั่นคือนำไปเรียงในลูกโซ่เวลา โดยใช้ โปรแกรมย่อย insert_list() และกำหนดค่าเวลาการเกิด ของแพคเกตต่อไปของสถานีนั้น แล้วนำไปเรียงลำดับ เพื่อรอการเกิดในลูกโซ่เวลา

การกำหนดเวลาเกิดของแพคเกตตัวใหม่นั้น โดยการใช้โปรแกรมย่อย Generate เพื่อกำหนดช่วงห่างของการเกิดแพคเกต แล้วนำไปบวกกับเวลาปัจจุบัน การทำงานของโปรแกรมย่อย token(pac) เป็นการควบคุมการทำงานของโทเกน โดยโทเกนจะวนไปรอบๆ ริงค์ เมื่อถึงสถานีใดๆ โทเกนจะเช็คว่าสถานีนั้น มีแพคเกตรออยู่ในคิวเพื่อรอการส่งหรือไม่ ถ้าไม่มีโทเกนจะเคลื่อนไปยังสถานีต่อไป ซึ่งในโปรแกรมจะมีการกำหนดเวลาที่จะถึงสถานีต่อไปให้กับโทเกน โดยการคำนวนทางคณิตศาสตร์ ถ้ามีแพคเกตรอการส่งอยู่ในคิว โปรแกรมหลักจะเรียกใช้โปรแกรมย่อย event_2 ต่อไป

4.3.2 การคำนวณเวลาของโทเกน

C = ความจุชันแนลของริงค์ (Channel Capacity) หน่วยเป็นเมกกะบิทต่อวินาที (Mbps)

B = รีพีทดีเลย์ (Repeat Delay) ของแต่ละสถานี หน่วยเป็นบิท

D = ความยาวของริงค์ หน่วยเป็น กิโลเมตร km

N = จำนวณสถานีในริงค์

= ความล่าช้าของโทเกน ในการเดินทางจากสถานีหนึ่งไปสถานีต่อไป

$$= \frac{D}{N(2 \times 10^{5})} + N \times B/C$$

4.3.3 การทำงานของโปรแกรมย่อย Event_2()

แพคเกตที่รออยู่ในคิวของแต่ละสถานีนั้น จะถูกส่งได้ก็ต่อเมื่อโทเกนวนมาถึงสถานีนั้นเท่านั้น เมื่อโทเกนวนมาถึงแล้ว แพคเกตนั้นจะได้รับการส่งไปในริงค์ โปรแกรมย่อย

Event_2() นี้ทำหน้าที่ ควบคุมการส่งของแพคเกต ในโปรแกรม หมายเลขของแพคเกตที่กำลังถูกส่ง จะเพิ่มขึ้นจากหมายเลขเดิม 100 เช่น แพคเกตหมายเลข 3 เมื่อถูกส่งไปในริงค์ จะถูกเปลี่ยนเป็น 103 เพื่อแสดงให้เห็นว่า แพคเกตนั้นกำลังถูกส่ง ช่วงเวลาที่แพคเกตถูกส่งไปในริงค์ จนกระทั่งวนกลับยังสถานีเดิม หาได้จากการคำนวณ

$$= \frac{D}{2 \times 10^{6}} + N \times B/C + L/C$$

แพคเกตจะถูกเรียงในลูกโซ่เวลา เพื่อรอเวลาที่แพคเกตจะวนมาถึงสถานีเดิมอีกครั้งเมื่อแพคเกตถูกส่งแล้ว โทเกนจะถูกส่งตามแพคเกตไปทันที โดยกำหนดเวลาที่โทเกนจะเดินทางไปถึงสถานีต่อไปจากการคำนวณ แล้วนำโทเกนไปเรียงในลูกโซ่เวลา

4.3.4 การทำงานของโปรแกรมย่อย Event_3

เมื่อแพคเกตที่ถูกส่งเข้าไปในริงค์วนกลับมาถึงสถานีเดิม แพคเกตนั้นจะถูกถอดออกจากริงค์ โดยโปรแกรมย่อย delete() ค่าต่างๆ จะถูกนำมาคำนวณ เพื่อหาประสิทธิภาพการทำงานของระบบ เช่น จำนวนแพคเกตทั้งหมดในระบบ จำนวนแพคเกตที่ส่งสำเร็จในเวลาที่กำหนด ค่าความล่าช้าของการส่งแพคเกต (delay) ค่าทรูพุท (throughput)

ค่าทรูพุท คือ อัตราการส่งสำเร็จของแพคเกต หรือกล่าวอีกอย่าง คือ จำนวนแพคเกตที่ส่งสำเร็จ ต่อจำนวนแพคเกตทั้งหมดในระบบ หาได้จาก

$$\text{ทรูพุท} = \text{จำนวนแพคเกตที่ส่งสำเร็จ} \times \frac{\text{เวลาที่ส่งแพคเกต}}{\text{เวลาที่ใช้ในการสิมิวเลชั่น}}$$

```
/* sort long integer input using linked */
# include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <alloc.h>
double log(double f);
float ran3();
float trans[300][10];
float numpac=0.0;
int allpac=0;
int length ,bit,bits,lengths;
float val=0;
float i;
float nod;
float timnow=0.0;
float th_put=0.0;
float delay=0.0;
float avdelay=0.0;
float q[300];
float gen[300];
float npac[300];
float q_tim[300][20];
int   pac,n,nowtrans;
float aldelay=0.0;
float simulate=0.0;
int k;
typedef struct node *NODEPTR;
struct node
{
  float number;
  float data;
  NODEPTR next;
};
NODEPTR link=NULL;       /*the sorted linked list */

NODEPTR    curr,
           prev=NULL,
           temp;

float generate();

  main()


 {
    int station,x,m;
    float t_trans;
           clrscr();
           windo();
              gotoxy(25,1);
              printf("PARAMITER OF SIMULATION");
           randomize();
              gotoxy(20,8);
              printf("ENTER STATION\n");
              gotoxy(50,8);
              scanf("%d",&station);
                k=station;          /*pac*/
              gotoxy(20,10);
              printf("ENTER RING LENGTH (km)");
```

```
            gotoxy(50,10);
              scanf("%d",&lengths);
            length=lengths;      /*ring length (km)*/
            gotoxy(20,12);
            printf("ENTER PACKET LENGTH (bits)");
            gotoxy(50,12);
              scanf("%d",&bits);
           bit =bits;      /* packet length (bit)*/
         simulate= 5;     /*msec*/
         set_trans(k);
         set_tok(k);

         nowtrans=link->number;

         while( link->data<=simulate)
             {
                  nowtrans=link->number;
                  switch(trans[nowtrans][4])
                    {

                      case 1:  event_1();
                               break;

                      case 2:  event_3();
                               break;

                      case 3:  token(k);
                               break;
                    }
             }
for (m=1;m<10;m++)
 /*  gotoxy(15,8);
    printf("node %d:  offer load=%.0f
    success=%.0f\n",m,gen[m],npac[m]);*/
    windo();
    gotoxy(25,1);
    printf("FDDI SIMULATION");
    gotoxy(20,6);
    printf("station          =%d\n",  k);
    gotoxy(20,8);
    printf("ring lenght      =%d\n",  length);
    gotoxy(20,10);
    printf("packet length    =%d\n",  bit);
    gotoxy(20,12);
    printf("delay time       = %f\n",  avdelay);
    gotoxy(20,14);
    printf("success packet   =%.0f\n",  numpac);
    gotoxy(20,16);
    printf("All packet       =%d\n",  allpac);
    th_put=numpac/allpac;
    gotoxy(20,18);
    printf("throughput       =%f\n",th_pui);
              /*throughput value*/
}
```

```
event_1()
 {
  float time;
  NODEPTR  temp;

     nod= trans[nowtrans][5];
     q[nod]=q[nod]+1.0;
     gen[nod]=gen[nod]+1.0;
   q_tim[nod][q[nod]]= trans[nowtrans][3];
                       /* reserve  time */

     time = generate();
     allpac=allpac++;
     gotoxy(20,18);
     printf("All pac= %d\n",allpac);
     val=time+trans[nowtrans][3];
     trans[nowtrans][3]=val;
        /* gen transmit time(new pac)  */

     link->data=val;
     temp=link;
     link=link->next;
     sert_list(temp);  /*  send old pac into queue */

     return (float)  q[nod];

  }


token(pac)
{

  float time ;

      if(trans[nowtrans][5] > pac)
                 trans[nowtrans][5]=1.0;
                   /*  begin at node.1 */

    nod= trans[nowtrans][5] ;

    if(q[nod]==0)              /*  no pac to send  */
       {

         time= ( (length/(pac*2e5))+(168/1e8) )*1e3;
                                        /*msec */

         val = time+trans[nowtrans][3];
         trans[nowtrans][3] = val;
         link->data = val;
         temp = link;
         link = link->next;
         sert_list(temp);       /* insert   timimg chain */

  trans[nowtrans][5]= trans[nowtrans][5]+1.0;
                      /*to next node */
```

```
        }
    else
        {
            event_2(pac);
        }
    return (float)  trans[nowtrans][5];

  }


event_2(pac)
 {
    float t_pac,t_tok;
    int m;
*    printf(" Event_2->token time= %f",trans[0][3]);*/

    nod= trans[nowtrans][5]; /*  node which token arrive */
      q_tim[nod][0]=q_tim[nod][1];
      q[nod]=q[nod]-1.0;
         /*  printf(" Queue[%.0f]= %.0f \n" ,nod,q[nod]);*/
      for(m =0;m<=q[nod];m++)

          q_tim[nod][m]= q_tim[nod][m+1]; /* shift queue */


      i= nod+ 100;
      trans[i][4]=2;              /*  set to event_3 */
      t_pac =  (  (length/2e5)+(pac*8e-8)+(bit/100e6) )*1e3;
                                  /*msec */

     val = t_pac + trans[0][3];
        /* now time is time token arrive */

      trans[i][3]=val;            /*   success time    */
      trans[i][5]=nod;
      insert_list();              /*  transmit pac */


      t_tok = ( (length/(pac*2e5)) + (168/100e6) )*1e3;
      val= t_tok + trans[0][3];
      trans[0][5]=trans[0][5]+1.0;
      trans[0][3]=val;
      link->data=val;
      temp=link;
      link=link->next;
      sert_list(temp);                 /*transmit token */

   return(float) q[nod];
 }
```

```
delete()
{
  link=link->next;
}


event_3()
{

nod= trans[nowtrans][5];
numpac=numpac+1.0;  /*success pac*/
npac[nod]=npac[nod]+1.0;      /*success pac each node */

aldelay= (trans[nowtrans][3]-q_tim[nod][0])+aldelay;
avdelay=aldelay/numpac;                    /*average delay*/

/* printf(" Event_3->success pac= %.0f node %.0f= %.0f\n",
numpac,nod,npac[nod]);  */

delete();

 }



float  generate()

  {
  float c;
  float f=0.0;
  float prob=0.0;
  float rate=0.0;     /*rate= (pac length/capacity)/prob  */

        prob=0.5;
        f=ran3();
             if(f<=0.000000)
                  f=ran3();
        rate= k* ( (bit/100e6)/prob )  ;
          /* average time between packets  */

        c=(-1) * (log(f)/log(10));
        val=  (c* rate )*1e3;          /* generate time */
        return(float)  val;
  }

  print_list()

  {
  NODEPTR buf;
    buf = link;
   for(;link!=NULL; link=link->next)
```

```
            printf("%f  ", link->data);
            printf("\n\n");

    puts ("");
    link = buf;
}


sert_list(NODEPTR temp)
{


    curr=link;
    prev = NULL;
for(;curr!=NULL&&val>curr->data;prev=curr,curr=curr->next);

    if (temp!=NULL)
    {
        temp->next = curr;
        if (prev==NULL)
            link = temp;
        else
            prev->next=temp;
    }
    return(temp !=NULL );
}



insert_list()
{

    NODEPTR  make_node();

    curr=link;
    prev = NULL;
for(;curr!=NULL&&val>curr->data;prev=curr,curr=curr->next);

    if ((temp=make_node(val))!=NULL)
    {
        temp->next = curr;
        if (prev==NULL)
            link = temp;
        else
            prev->next=temp;
    }
    return(temp !=NULL );
}



NODEPTR make_node(value)
float value;
{

NODEPTR
```

```
        newptr;

if ((newptr=(NODEPTR) malloc(sizeof(struct node))) !=NULL)
    {
       newptr->number = i;
       newptr->data=value;
       newptr->next=NULL;

     }

 return (NODEPTR) newptr;
 }



set_trans(pac)

  {
     for (i=1;i<= pac;i++)
        {
            generate();
            insert_list();
            trans[i][0]= prev->number;
            trans[i][1]= curr->number;
            trans[i][2]= temp->number;
            trans[i][3]= temp->data; /* gen transmit time */
            trans[i][4]= 1;                  /* packet */
            trans[i][5]= i;      /* packet each node. */

        }
 }


set_tok(pac)
{

    i=0;
    val= ( (length/(pac*2e5) )+ (8/100e6) )*1e3;  /*msec */
    insert_list();
    trans[0][0]=prev->number;
    trans[0][1]=curr->number;
    trans[0][2]=temp->number;
    trans[0][3]=val;
    trans[0][4]=3;                /* token */
    trans[0][5]=1;               /* node 1 */

    return(float) val;

  }



float ran3()
{
 static long int a;
```

```
 a=rand();
 a=(a*10001+3)%1717;
 return(float) a/1717;
 }

windo()
{
  clrscr();
   window(1,5,80,40);
   textattr(BLACK*16+LIGHTGRAY);
   clrscr();
   window(1,1,75,25);
   textattr(LIGHTGRAY*16+BLACK);
   clrscr();
   window(3,2,73,24);
   textattr(BLACK*16+LIGHTGRAY);
   clrscr();

}
```

4.4 ผลการทดลอง

จากการจำลองการทำงานของ FDDI ได้ผลดังต่อไปนี้คือ

กราฟรูปที่ 1

เป็นกราฟแสดงความสัมพันธ์ระหว่าง ความยาวของริงค์ (km) และค่าความล่าช้าของการส่งแพคเกต โดยการกำหนดค่าในโปรแกรมดังนี้

ตัวแปร

capacity = 100 Mbps

จำนวนสถานี = 100 สถานี

ความยาวของแพคเกต = 1000 bits

กราฟที่ได้จากโปรแกรมจำลองระบบ ซึ่งป้อนตัวแปรดังกล่างนั้นแสดงว่า เมื่อระยะทางของริงค์เพิ่มขึ้น เวลาความล่าช้าของการส่งแต่ละแพคเกตจะเพิ่มมากขึ้นด้วย โดยที่จำนวนสถานีเท่าเดิมนั้น ระยะทางระหว่างสถานีจะเพิ่มขึ้นแต่ละสถานีจะต้องรอการมาถึงของโทเกนนานขึ้น และการที่ความยาวของริงค์มีมากนั้น ทำให้เวลาการเดินทางไปในริงค์ จนกระทั่งวนกลับมายังสถานีเดิมมีมากด้วย ด้วยเหตุผลดังกล่าวนี้ สรุปว่าค่าความล่าช้าของการส่งแพคเกตจะมากขึ้น เมื่อมีจำนวนสถานีในระบบมาก

DELAY
0.35
0.30
0.25
0.20
0.15
0.10
0.05
C = 100 Mbps
NODE = 100
Lp = 1000 bits
0
1
2
3
4
5
6
7
8
9
10
D (km)

กราฟรูปที่ 2

แสดงความสัมพันธ์ระหว่าง ค่าความล่าช้าของการส่งแพคเกต กับความยาวของแพคเกตที่ส่ง โดยกำหนดค่าตัวแปรดังนี้

capacity = 100 Mbps

ความยาวริงค์ = 5 km

จำนวนสถานี = 100 สถานี

จากกราฟ แสดงถึงผลการทดลองที่ได้จากโปรแกรมจำลองระบบ ดังนี้คือ เมื่อความยาวของแพคเกตมากขึ้น จะทำให้ค่าความล่าช้าของการส่งแพคเกตมีมาก ซึ่งสามารถอธิบายได้ว่า เมื่อสถานีที่ต้องการส่งแพคเกต ได้ส่งแพคเกตเข้าไปในริงค์แล้วสถานีนั้นจะส่งโทเกนตามหลังแพคเกตไปทันที การที่โทเกนเคลื่อนที่ต่อท้ายแพคเกตไปเช่นนี้ สถานีต่อไปที่ต้องการส่ง จะต้องรอจนกว่าแพคเกตที่ถูกส่งโดยสถานีก่อนหน้า จะผ่านพ้นไปเสียก่อน จึงจะได้รับโทเกน ดังนั้นถ้าแพคเกตมีขนาดสั้น สถานีจะได้รับโทเกนเร็วมันจึงสามารถส่งแพคเกตได้ทันที แต่ถ้าแพคเกตมีขนาดยาว สถานีจะต้องรอโทเกนเป็นเวลานานมีผลให้ค่าความล่าช้าของการส่งแพคเกตมีค่ามาก

DELAY
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
10
20
30
40
50
60
70
80
90
100
PAC_LENGTH
(x10 bits)

กราฟรูปที่ 3

แสดงความสัมพันธ์ระหว่างความล่าช้าของการส่งแพคเกต กับจำนวนสถานีในริงค์ โดยกำหนดค่าตัวแปรดังนี้

capacity = 100 Mbps

ความยาวริงค์ = 5 km

ความยาวแพคเกต = 1000 bits

จากกราฟที่ได้จากการทดลองแสดงว่า เมื่อจำนวนสถานีในริงค์เพิ่มมากขึ้น จะทำให้ค่าความล่าช้าของการส่งแพคเกตมีมากขึ้นด้วย ซึ่งอธิบายได้ว่า ขณะที่แพคเกตถูกส่งไปในริงค์ ข้อมูลในแพคเกตจะถูกรีพีทที่สถานีต่างๆ ก่อนที่จะถูกส่งไปยังสถานีต่อไป ดังนั้นเมื่อจำนวนสถานีมากขึ้น เวลาที่เสียไปทั้งหมดในการรีพีทข้อมูลในแพคเกตจะมีมาก จึงมีผลให้ค่าความล่าช้าของการส่งแพคเกตมีมากขึ้นด้วย

DELAY
0.35
0.30
0.25
0.20
0.15
0.10
0.05
C = 100 Mbps
D = 5 km
Lp = 1000 bits
0
10
20
30
40
50
60
70
80
90
100
NODE

4.5 สรุปและแนวทางพัฒนาต่อไป

โครงงานนี้เนื่องจากเป็นการศึกษาการทำงานของระบบ ดาต้าดิสทริบิวอินเตอร์เฟส โดยการเขียนโปรแกรมจำลองระบบ(simulation) เพื่อดูประสิทธิภาพการทำงานของระบบว่า มีลักษณะขั้นตอนการทำงานอย่างไร มีค่าตัวแปรใดที่มีผลทำให้ประสิทธิภาพการทำงานดีขึ้น หรือเลวลงอย่างไรบ้าง

ผลจากโปรแกรมจำลองนี้ ได้แสดงผลและเปรียบเทียบออกมาเป็นกราฟ ดังที่แสดงในผลการทดลอง ซึ่งแสดงให้เห็นถึงความสัมพันธ์ของตัวแปรที่มีผลต่อระบบ ซึ่งถ้าผลที่ได้จากการจำลองระบบ มีทิศทางการเปลี่ยนแปลงในทิศทางเดียวกับผลที่ควรจะเป็นในทางทฤษฎี และเราสามารถอธิบายได้ว่า ทำไมผลการทดลองจึงเป็นเช่นนั้น เราก็จะสรุปได้ว่า โปรแกรมจำลองระบบของเรานั้น ประสบผลสำเร็จตามจุดประสงค์ และสามารถนำไปประยุกต์ใช้กับงานอื่นได้

อย่างไรก็ตามโปรแกรมจำลองระบบ ดาต้าดิสทริบิวอินเตอร์เฟส(fddi)นี้ ถือว่ายังไม่สมบูรณ์ เพราะว่ายังขาดไทม์เมอร์(timer) เพื่อใช้ในการกำหนดความสำคัญของแพคเกตข้อมูลที่ส่ง(priority) ซึ่งเป็นสิ่งที่ควรพัฒนาต่อไป

สำหรับโปรแกรมจำลองระบบนี้ นับเป็นเรื่องที่น่าสนใจอย่างยิ่งแนวความคิดหนึ่งที่เราสามารถนำไปใช้ประโยชน์คือ การประยุกต์ใช้เป็นซอฟต์แวร์ออกแบบระบบเนตเวอร์ค ซึ่งจุดประสงค์ของซอฟแวร์นี้คือ ช่วยผู้ใช้โปรแกรมออกแบบระบบ และต้องติดต่อกับผู้ใช้โปรแกรมด้วยภาษาของผู้ใช้ (มิใช่ภาษาโปรแกรม)ในเรื่องการออกแบบระบบนั้นซอฟต์แวร์ต้องมีความสามารถ ช่วยผู้ใช้โปรแกรมวาดรูประบบ และกำหนดคุณลักษณะขององค์ประกอบในระบบ ภายหลังการออกแบบแล้ว จะต้องสามารถจำลองการทำงาน ให้ผู้ใช้เห็นพฤติกรรมของระบบ และสรุปผลการจำลอง เพื่อให้ผู้ใช้สามารถปรับปรุงการทำงานต่อไปได้

ในอนาคต การวิจัยจะได้ขยายการประยุกต์ใช้ ทั้งการจำลองแบบ และวิธีสร้างส่วนติดต่อกับผู้ใช้ กับงานด้านอื่นๆ ในเรื่องของระบบเนตเวอร์ค เราจะได้ทำการศึกษาระบบเน็ตเวอร์คต่างชนิดกัน และการรวมเนตเวอร์คเหล่านี้ เข้าเป็นระบบจำลองเดียว การสร้างระบบจำลองการทำงาน ของทั้งระบบการสื่อสารระยะใกล้และไกล จะยังมีประโยชน์อย่างมาก ต่อผู้ออกแบบในอนาคต

ภาคผนวก

สเป็คของ FDDI ริงค์เนตเวอร์ค

(a) Media-interface connector:

Duplex fiber-optic keyed connector, ferrule-in-sleeve design
Insertion loss: Not specified; included in terminal power measurements or cable loss specifications

Four types:	A:	Primary in/secondary out
	B:	Secondary in/primary out
	M:	Single attach to concentrator
	S:	On single-attach station

(b) Network performance:

Data rate:	100 Mb/s, clock rate of 125 Mb/s
Number of stations:	500 maximum
Network size:	100-km duplex ring, 200-km fiber path length
Maximum station spacing:	2 km maximum between stations
Maximum station delay:	756 ns
Maximum loop delay:	1.733 ms based on 5085-ns/km fiber delay and 200-km loop with 500 stations
Addressing:	16- or 48-bit, individual and multicast
Bit-error rate:	$< 10^{-9}$ BER total ring
BER through station:	$< 2.5 \times 10^{-10}$ under all conditions at minimum power; $< 1 \times 10^{-12}$ when input power is 2 dB above minimum
Tx coding:	NRZI 4B/5B

(c) Station optical transmitter:

Center wavelength:	1270 to 1380 nm
Average output power:	−20 to −14 dB at the output of the test connector
Duty-cycle distortion:	1 ns maximum
Data dependent jitter:	0.6 ns maximum
Random jitter:	0.76 ns maximum
Extinction ratio:	10% maximum
Spectral width:	Width plus chromatic dispersion plus source rise time must achieve an optical rise time less than 5 ns in a 2-km length of fiber; ranges from a width of 100 to 200 nm

Optical pulse:	80 ns ±500 ppm time interval; 40 ns ±0.7 ns mean pulse width; 0.6 to 3.5 ns rise-time window

(d) Station optical receiver:

Average received power:	−31 to −14 dBm at the input to the test connector
Rise/fall time:	0.6 to 5 ns
Duty-cycle distortion:	1 ns maximum
Data dependent jitter:	1.2 ns maximum
Random jitter:	0.76 ns peak-to-peak maximum

(e) Bypass switch:

Attenuation:	2.5 dB maximum
Isolation:	40 dB worst-case
Time to switch:	25 ms from command
Media interrupt:	15 ms maximum

(f) Fiber plant:

Fiber type:	Multimode
Core diameter:	62.5 μm per EIA 455-58
Cladding diameter:	122.0 to 128.0 μm per EIA 455-27, −48
Numerical aperature:	0.275 per EIA 455-177
Attenuation at 1300 nm:	<2.5 dB/km typical, 11.0 dB maximum end to end per EIA 455-53
Modal bandwidth:	500 MHz·km minimum at 1300-nm optical 3-dB BW per EIA 455-30, −51, −54
Chromatic dispersion:	See standard for profile, 0.11 ps/nm/km between 1300 and 1349 nm

Optional fiber types not specified to date:

50/125 μm at 0.2 and 0.22 N.A.
85/125 μm at 0.26 N.A.
100/140 μm at 0.29 N.A.

หนังสืออ้างอิง

[1] Andrew S.Tanenbanm "Computer Network" McGraw-Hill,1989.

[2] Donald G. Baker "Local Area Networks with Fiber-Optic Applications" Prentice-Hall, 1986.

[3] Ged E. Keiser "Local Area Networks" McGraw-Hill, 1989.

[4] Stawart V. Hoover and Ronald F. Perry "Simulation" Addison-Wisley, 1989.

[5] Ruth Davies and Robert O'Keefe "Simulation Modelling with Pascal" Prentice Hall International(UK)Ltd, 1989.

[6] William Stalling "Local Networks 2nd Eddition" Macmillan Publishing Company, 1987.

[7] สุริยัน ศรีสวัสดิ์กุล "ระบบการสื่อสายข้อมูลคอมพิวเตอร์" พิสิกส์เซ็นเตอร์การพิมพ์

กิตติกรรมประกาศ

การทำปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี ทั้งนี้เนื่องมาจากความช่วยเหลือจากผู้มีพระคุณหลายท่าน ขอขอบพระคุณ อาจารย์ สุวิพล ลิทธิชีวภาค อาจารย์ที่ปรึกษา ที่กรุณาให้ความช่วยเหลือให้คำแนะนำปรึกษา ขอขอบพระคุณ อาจารย์ กอบชัย เดชหาญ หัวหน้าภาควิชาโทรคมนาคม ที่เอื้ออำนวยอุปกรณ์ต่างๆ และอนุญาตให้ใช้ห้องคอมพิวเตอร์นอกเวลาราชการ ซึ่งทำให้การทำงานสะดวกขึ้นเป็นอย่างมาก และขอบคุณเพื่อนๆ ที่ให้คำแนะนำและช่วยแสดงความคิดเห็น ตลอดจนให้กำลังใจในการทำงานในครั้งนี้

นางสาวดอกไม้ สนิทนิตย์

นางสาวเสาวลักษณ์ แสนวิทยากร

คณะผู้จัดทำ