การค้นหาภาพแบบอัตโนมัติบน Peer – to – Peer network

AUTOMATIC RETRIVEVAL ON PEER – TO – PEER NETWORK

นาย อนุรักษ์ นันตา รหัส 45380160

นาย อนุสรณ์ ใฝ่จิตร์ รหัส 45380162

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2549

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	การค้นหาภาพแบบอัตโนมัติบน Peer to Peer Network
ผู้ดำเนินโครงงาน	นายอนุรักษ์ นันตา รหัส 45380160
	นายอนุสรณ์ ใฝ่จิตร์ รหัส 45380162
อาจารย์ที่ปรึกษา	ดร. ไพศาล มุณีสว่าง
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2548

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มม่น)

.....กรรมการ

(ดร.ไพศาล มุณีสว่าง)

.....กรรมการ

(ดร.พนมขวัญ ริยะมงคล)

หัวข้อโครงงาน การค้นหาภาพแบบอัตโนมัติบน Peer to Peer Network

ผู้ดำเนินโครงงาน นายอนุรักษ์ นันตา รหัส 45380160

นายอนุสรณ์ ใฝ่จิตร์ รหัส 45380162

อาจารย์ที่ปรึกษา ดร. ไพศาล มุณีสว่าง

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2549

.....

บทคัดย่อ

โครงงานนี้ศึกษาและพัฒนาโปรแกรมการค้นหารูปภาพแบบอัตโนมัติบน Peer to peer Network โดยใช้หลักการของ Contest – Based Methods โดยใช้วิธีการของ Color Analysis ของระบบสี RGB เพื่อทำการเปรียบเทียบค่าที่ได้จากการ Quantize HSV Color Histogram ของแต่ละไฟล์ภาพแล้วหาค่าผลต่างเรียงลำดับจากน้อยไปมาก ทำให้ได้ผลการค้นหาที่เกิดจากการเปรียบเทียบค่าของการ Quantize HSV Color Histogram ซึ่งโปรแกรมนี้จะมีสองวิธีการที่จะทำการศึกษาและทดลอง คือ การค้นหารูปภาพแบบ Local และการค้นหาบน Peer to Peer Network โดยโปรแกรมที่พัฒนานี้จะใช้ C# Window Application ของ Microsoft visual studio .NET และมีการจัดการฐานข้อมูลโดยใช้ Microsoft Access 2003 เป็นเครื่องมือในการจัดการ

โดยผลการทดลองที่ได้สามารถเปรียบเทียบและค้นหารูปภาพแบบ Local และ Peer to Peer Network ได้โดยใช้หลักการของ Color Analysis จากการทดลองผลที่ได้อาจจะไม่แม่นยำเมื่อมองในลักษณะของภาพชนิดเดียวกัน แต่ถ้ามองในลักษณะขององค์ประกอบโดยรวมของสีโดยไม่สนใจว่าภาพจะเป็นชนิดเดียวกันหรือเปล่าจะได้ผลที่น่าพอใจ และผลที่ได้อาจไม่แม่นยำเมื่อเปรียบเทียบกับการใช้หลักการอื่นๆ อย่างเช่น Shape feature และ Texture feature

Project title	Automatic retrieval on Peer – to – Peer network
Name	Mr. Anuruk Nunta ID. 45380160
	Mr. Anusorn Faijit ID. 45380162
Project advisor	Dr. Paisarn Muneesawang
Major	Computer Engineering
Department	Electrical and Computer Engineering
Academic year	2006

.....

Abstract

The purposes of this project are to study and develop the program of picture searching for automatic retrieval on Peer – to - Peer network. This program employs the principle of Content – based method by RGB color analysis. The program makes a comparison of each picture file by quantize HSV color histogram. Then, the program provides the values of difference and sorts them from the least to the mast. As a result, this program is able to search the picture files by the comparison of quantize HSV color histogram. There are two methods of this study and application: Local search and P2P search. The program was conducted on C# window application of Microsoft visual studio.Net, and the data base was administrated by Microsoft Access 2003.

The finding of this project shows that this program can make the comparison and search the picture files by Local and Peer to Peer network. However, the searched picture files may not have the same character as the original ones. In other words, if there is no interest in the picture's characteristic, this program provides the satisfaction. This study may not provide the accurate searching when the comparison is conducted with other principles such as the shape and the texture.

กิตติกรรมประกาศ

ขอขอบพระคุณ ดร. ไพศาล มุณีสว่าง อาจารย์ที่ปรึกษา ที่คอยให้คำปรึกษาและให้ความช่วยเหลือตลอดจนคำแนะนำต่างๆ ในการทำโครงงานชิ้นนี้ สุดท้ายต้องขอขอบพระคุณอาจารย์ทุกท่านและเพื่อนๆ พี่ ๆทุกคนที่ยังไม่ได้เอ่ยนามที่ให้คำแนะนำและให้การสนับสนุนผู้จัดทำโครงงานให้สามารถทำโครงงานชิ้นนี้จนสำเร็จลุล่วงไปได้ด้วยดี

นายอนุรักษ์ นันตา
นายอนุสรณ์ ใฝ่จิตร์

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ก
บทคัดย่อภาษาอังกฤษ ..... ข
กิตติกรรมประกาศ ..... ค
สารบัญ ..... ง
สารบัญตาราง ..... ฉ
สารบัญรูป ..... ช

บทที่ 1 บทนำ
1.1 ที่มาและความสำคัญของโครงงาน 1
1.2 วัตถุประสงค์ของโครงงาน 2
1.3 ขอบข่ายงานของโครงงาน 2
1.4 กิจกรรมการดำเนินงาน 3
1.5 ผลที่คาดว่าจะได้รับ 3
1.6 งบประมาณในการดำเนินงาน 4

บทที่ 2 ทฤษฎีเบื้องต้น
2.1 ชนิดของการเชื่อมต่อในระบบเครือข่าย 5
2.2 โครงสร้างโดยรวมของ Network 7
2.3 พื้นฐานเกี่ยวกับ โปรโตคอล TCP/IP 8
2.4 มาตรฐานของสี 16
2.5 ระบบสี HSV/I 18
2.6 Content Base Image Retrieval 20
2.7 ทำความรู้จักกับ C# 22

บทที่ 3 วิธีการดำเนินการ
3.1 การจัดการฐานข้อมูล 23
3.2 การหาค่า RGB Color histogram 24
3.3 Index Server 27
3.4 วิธีการค้นหารูปภาพ 28
3.5 โครงสร้างการทำงาน 32

สารบัญ (ต่อ)

หน้า

บทที่ 4 ผลการทดลอง

4.1 สิ่งที่ต้องเตรียมก่อนทำการทดลอง 33

4.2 ผลการทดลอง 33

4.3 เปรียบเทียบผลการทดลอง 42

บทที่ 5 สรุปผลและข้อเสนอแนะ

5.1 สรุปผล 43

5.2 ปัญหาที่พบ 43

5.3 ข้อเสนอแนะ 44

เอกสารอ้างอิง 45

ภาคผนวก ก. Source Code การจัดการฐานข้อมูล 46

ภาคผนวก ข. Index Server 66

ประวัติผู้เขียนโครงงาน 70

สารบัญตาราง

ตารางที่ หน้า

1.1 ขั้นตอนการดำเนินโครงการ 3

2.1 ตารางเปรียบเทียบการเชื่อมต่อแบบ SERVER - BASED เทียบกับ PEER - TO PEER 6

2.2 รายละเอียดของ FLAG แต่ละชนิด 15

4.1 บันทึกการทดลองค้นหารูปภาพจากไฟล์รูปภาพบนCLIENT นั้นๆ (SEARCH LOCAL) 37

4.2 บันทึกการทดลองค้นหารูปภาพจากไฟล์รูปภาพบนNETWORK (SEARCH VIA P2P) 41

สารบัญรูป

รูปที่		หน้า
2.1	ลักษณะโครงสร้างของ DECENTRALIZED NETWORK	7
2.2	ลักษณะโครงสร้างของ CENTRALIZED NETWORK	7
2.3	ขั้นตอนการ ENCAPSULATION และ DEMULTIPLEXING	9
2.4	โครงสร้าง TCP/IP	10
2.5	IP HEADERS	11
2.6	ICMP HEADERS	13
2.7	UDP HEADERS	14
2.8	TCP HEADERS	15
2.9	RGB COORDINATES SYSTEM	16
2.10	RGB COLOR MODEL เมื่อมองจากทางด้าน WHITE	17
2.11	เวกเตอร์ของแผนภูมิค่าความสดใสและค่าโทนสี	18
2.12	แสดงระบบสี HSV	19
3.1	ส่วนประกอบของขั้นตอนการค้นหารูปภาพ	23
3.2	ขั้นตอนการวิเคราะห์องค์ประกอบสี H, S, V ของรูปภาพ	24
3.3	วิเคราะห์หาค่าสีของรูปภาพ	24
3.4	แสดงค่าองค์ประกอบของสี H, S, V เมื่อทำการ QUANTIZE เป็นเวกเตอร์	25
3.5	แสดงตัวอย่างฐานข้อมูลเก็บค่า HSV HISTOGRAM ของรูปภาพ	26
3.6	โครงสร้าง PEER TO PEER	27
3.7	ตัวอย่างของ INDEX SERVER ที่มี PEER เชื่อมต่ออยู่ 10 PEER	28
3.8	ขั้นตอนการค้นหารูปภาพแบบ LOCAL	29
3.9	ขั้นตอนการค้นหารูปภาพแบบ P2P	29
3.10	แสดงตัวอย่างการค้นหารูปภาพแบบ LOCAL	30
3.11	แสดงตัวอย่างการค้นหารูปภาพแบบ P2P	31
3.12	แสดงขั้นตอนการทำงาน	32
4.1	แสดงการ QUERY จากไฟล์รูปตัวอย่าง (SEARCH LOCAL)	34
4.2	แสดงการ QUERY จากไฟล์รูปตัวอย่าง (SEARCH LOCAL)	35
4.3	แสดงการ QUERY จากไฟล์รูปตัวอย่าง (SEARCH LOCAL)	36
4.4	การค้นหารูปภาพบน CLINET2 เครื่อง (SEARCH VIA P2P)	38

สารบัญรูป (ต่อ)

รูปที่		หน้า
4.5	การค้นหารูปภาพบนCLIENT 2 เครื่อง (SEARCH VIA P2P)	39
4.6	การค้นหารูปภาพบนCLIENT 2 เครื่อง (SEARCH VIA P2P)	40
4.7	กราฟแสดงการเปรียบเทียบการทำงานของแต่ละแบบ	42

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบันจะเห็นได้ว่ารูปภาพได้เข้ามามีส่วนเกี่ยวข้องกับงานทางด้านต่าง ๆ อย่างมากมาย เช่น งานด้านการแพทย์ พิพิธภัณฑ์ภาพ เป็นต้น ดังนั้นรูปภาพจึงถือเป็นข้อมูลที่สำคัญอีกประเภทหนึ่ง ซึ่งในการเก็บรวบรวมรูปภาพไว้ด้วยกันเป็นจำนวนมาก ๆ นั้นจะทำให้เกิดปัญหาในการค้นหารูปภาพต่าง ๆ ที่ต้องการนำมาใช้งานในภายหลัง ดังนั้นเราจึงได้นำเอาเรื่องของ Network เข้ามาช่วยแก้ปัญหาในเรื่องนี้ สาเหตุที่ใช้ Network เข้ามาช่วยแก้ปัญหาเพราะว่าในปัจจุบันนี้ Network ได้เข้ามามีบทบาทต่อความสะดวกสบายในการติดต่อสื่อสาร และค้นหาข้อมูลในองค์กรต่าง ๆ ไม่ว่าจะเป็นองค์กรเล็กหรือองค์กรใหญ่ ๆ ซึ่งองค์กรเหล่านี้ได้ให้ความสำคัญและนำNetwork เข้ามาใช้งานในองค์กรกันมากขึ้น จึงทำให้ผู้สนใจเกิดแนวคิดที่จะศึกษาทางด้าน Network และแก้ปัญหาเกี่ยวกับประสิทธิภาพในการค้นหาข้อมูลบนNetwork เพราะการค้นหาข้อมูลในปัจจุบันส่วนมากจะใช้ทรัพยากรของมนุษย์และเครื่องคอมพิวเตอร์มาก ซึ่งจะทำให้เสียเวลาและเสียค่าใช้จ่ายสูง ดังนั้นจึงได้คิดวิธีการค้นหาข้อมูลแบบอัตโนมัติขึ้น ในที่นี้จะศึกษาและทำโครงงานการค้นหาข้อมูลรูปภาพแบบอัตโนมัติบน Network ชนิด Peer-to-Peer

สาเหตุที่เลือกพัฒนาการค้นหาข้อมูลบน Peer-to-Peer network เพราะว่า Decentralized network ชนิด Peer-to-Peer client แต่ละตัวจะมี Data base อยู่ในตัวของมันเอง และ client แต่ละตัวสามารถประมวลผลในตัวมันเองได้ ซึ่ง client แต่ละตัวจะทำหน้าที่เป็น client – server โดยมีหลักการทำงานในการค้นหาข้อมูลบน peer – to – peer network client แต่ละตัวจะทำงานตามคำร้องขอของ client ที่ทำการร้องขอนั้น ๆ โดย client ที่ได้รับการร้องขอจะทำหน้าที่เป็น server และทำการประมวลผลค้นหาข้อมูลภายใน Data base ของตัวมันเอง เมื่อได้ข้อมูลตามที่ต้องการแล้วจึงส่งข้อมูลไปยัง client ที่ร้องขอ ดังนั้นจะเห็นได้ว่าการทำงานของ peer – to – peer network จะใช้เวลาในการประมวลผลน้อย เพราะ client แต่ละตัวสามารถประมวลผล Data base ของตัวมันเองได้ ซึ่งแตกต่างจาก Centralized network ที่มีการประมวลผลที่ server เพียงตัวเดียว และในระบบ Data base ของ client แต่ละตัวจะถูกนำมาเก็บรวมไว้ใน server เดียวกัน ถ้ามีข้อมูลจำนวนมาก ๆ ถูกเก็บไว้ใน server จะต้องใช้เวลามากในการค้นหาข้อมูลและจะต้องใช้หน่วยความจำที่มีขนาดใหญ่ในการเก็บข้อมูล เพราะฉะนั้นจึงทำให้เสียค่าใช้จ่ายเป็นจำนวนมาก

ซึ่งในการศึกษาโครงงานการค้นหาข้อมูลรูปภาพแบบอัตโนมัติบน peer – to – peer network จะใช้ทฤษฎีของ Content – Based Methods โดยมีการเขียนโปรแกรมควบคุมการทำงานของ peer – to – peer network โดยใช้ภาษา C# และค้นหารูปภาพ โดยใช้วิธี Content – based Image retrieval (CBIR) ซึ่ง Content – based Image retrieval (CBIR) มีอยู่ 2 ขั้นตอนด้วยกัน คือ ทำ Indexing โดยใช้วิธีการวิเคราะห์สี (Color Analysis) แล้วหลังจากที่ได้ Index ออกมาแล้วจึงนำ Index ที่ได้ไปทำการเปรียบเทียบ (Matching) กับรูปภาพใน Data base ของ client แต่ละตัวที่เราต้องการค้นหา

1.2 วัตถุประสงค์ของโครงงาน

- เพื่อศึกษาการค้นหารูปภาพโดยอัตโนมัติโดยใช้ทฤษฎีของ Contest – Based Methods และการนำมาประยุกต์บน peer-to-peer network
- เขียนโปรแกรมเพื่อทำการค้นหารูปภาพโดยอัตโนมัติ โดยใช้ภาษา C#
- เพื่อศึกษาวิธีการทำ Content – based Image retrieval (CBIR)

1.3 ขอบเขตของโครงงาน

- เขียนโปรแกรมควบคุมการทำงานของ peer – to – peer network ในการค้นหารูปภาพ โดยใช้วิธีการของ Content – based Image retrieval (CBIR)
- ใช้วิธีการวิเคราะห์สี (Color Analysis)
- ใช้ Network ชนิด peer – to – peer ในการพัฒนา

1.4 กิจกรรมการดำเนินงาน

กิจกรรม	เดือน-ปี									
	ธ.ค. 2547	ม.ค. 2548	ก.พ. 2548	มี.ค. 2548	เม.ย. 2548	พ.ค. 2548	มิ.ย. 2548	ก.ค. 2548	ส.ค. 2548	ก.ย. 2548
ศึกษาทฤษฎีและหลักการของ peer – to – peer network	←	—	—	→						
ศึกษาทฤษฎีของ Contest – Based Methods		←	—	—	→					
ศึกษาวิธีการทำ Content – based Image retrieval (CBIR)			←	—	—	→				
ศึกษาภาษา C# และการเขียนโปรแกรมควบคุม peer – to – peer network				←	—	—	→			
ศึกษา Algorithm ในการวิเคราะห์ภาพแบบต่าง ๆ					←	—	—	→		
ประยุกต์และพัฒนาแนวคิดที่จะนำมาใช้วิเคราะห์ภาพ						←	—	—	→	
ทดสอบ Algorithm ในการค้นหารูปภาพแบบอัตโนมัติบน peer – to – peer network							←	—	—	→
ทดสอบใช้งานจริง								←	—	→
จัดทำรายงาน										← — →

ตารางที่ 1.1 ขั้นตอนการดำเนินโครงการ

1.5 ผลที่คาดว่าจะได้รับ

- โปรแกรมที่สามารถสั่งงานให้ client แต่ละตัวทำการประมวลผลค้นหารูปภาพได้ตามที่ต้องการ
- ได้รับความรู้เกี่ยวกับทฤษฎีของของ Content – based Image retrieval (CBIR)
- ได้รับความรู้เกี่ยวกับทฤษฎีของของ Content – based methods
- สามารถนำความรู้ที่ได้ไปพัฒนาระบบการค้นข้อมูลรูปแบบอื่นได้

1.6 งบประมาณในการดำเนินงาน

1. ค่าวัสดุอุปกรณ์ และเครื่องมือ	1500	บาท
2. ค่าจัดทำรูปเล่ม	500	บาท
รวมทั้งหมด	2,000	บาท

บทที่ 2
ทฤษฎีเบื้องต้น

ในโลกปัจจุบันจะเห็นว่าผู้คนสนใจที่จะพัฒนาโปรแกรมในการ Share ข้อมูลกันบนระบบ Network และ Internet อย่างมากมาย เช่น โปรแกรม Bit Comet ซึ่งเป็นโปรแกรมที่ Client มีการเชื่อมต่อและ Share ข้อมูลซึ่งกันและกันทุกชนิดที่ Client Share ให้ในระบบ ไม่ว่าจะเป็น รูปภาพ เพลง หนัง และโปรแกรมอื่นๆ จึงทำให้การได้มาซึ่งข้อมูลที่ต้องการเร็วและไวกว่าการหาและ Download จากที่ที่เดียว จากตรงนี้จึงทำให้ผู้สนใจเกิดแนวคิดที่จะสร้างโปรแกรมเพื่อ Share ข้อมูลกันโดยใช้วิธีการของ Search Engine ในการค้นหาข้อมูลที่ต้องการในระบบ หรือวง Network ซึ่งโปรแกรมจะทำให้ Client มีคุณสมบัติเป็น Client – server

จากแนวคิดของ Search Engine และข้อดีของ Decentralized network ชนิด peer – to – peer ที่มีประสิทธิภาพในการค้นหาข้อมูลที่มีปริมาณมากๆใน Network ดีและเร็วกว่า Centralized network จึงทำให้ผู้สนใจนำ Network ชนิดนี้เป็นตัวพัฒนาโปรแกรม และใช้วิธีการของ Content – based Image retrieval (CBIR) วิเคราะห์ข้อมูลชนิดรูปภาพเพื่อให้โปรแกรมมีความถูกต้องของข้อมูลรูปภาพที่ต้องการมากขึ้น

2.1 ชนิดของการเชื่อมต่อในระบบเครือข่าย

การเชื่อมต่อคอมพิวเตอร์เข้าด้วยกันเป็นเครือข่ายเฉพาะบริเวณนั้น จุดประสงค์หลักอย่างหนึ่งก็คือการแบ่งกันใช้ทรัพยากรที่มีอยู่ โดยทรัพยากรเหล่านั้นอาจเป็นหน่วยประมวลผลกลาง CPU ความเร็วสูง ฮาร์ดดิสก์ เครื่องพิมพ์ หรือแม้แต่อุปกรณ์สื่อสารต่าง ๆ ซึ่งอุปกรณ์เหล่านี้จะเชื่อมอยู่กับคอมพิวเตอร์เครื่องใดเครื่องหนึ่ง วิธีการเชื่อมต่อเครือข่ายคอมพิวเตอร์ เพื่อจัดสรรการใช้งานทรัพยากรในระบบเครือข่ายสามารถจำแนกได้เป็น 2 รูปแบบคือ

2.1.1 เครือข่ายแบบพึ่งเครื่องบริการ (Server - based networking)

เป็นการเชื่อมต่อโดยมีเครื่องบริการอยู่ศูนย์กลาง ทำหน้าที่ในการให้บริการต่าง ๆ ที่เครื่องผู้ใช้หรือสถานีงาน (Workstation) ร้องขอ รวมทั้งเป็นผู้จัดการดูแลการจราจรในระบบเครือข่ายทั้งหมด นั่นคือการติดต่อกันระหว่างเครื่องต่าง ๆ จะต้องผ่านเครื่องเซิร์ฟเวอร์ เครื่องผู้ใช้จะทำการประมวลผลในงานของตนเท่านั้น ไม่มีหน้าที่ในการให้บริการกับเครื่องอื่น ๆ ในระบบ เครื่องผู้บริการในระบบเครือข่ายชนิดนี้อาจมีได้ 2 รูปแบบคือ

1. **เครื่องบริการแบบอุทิศ (Dedicated Server)** หมายถึงเครื่องบริการทำหน้าที่บริการอย่างเดียวเท่านั้น ไม่สามารถนำไปใช้ในงานทั่ว ๆไปได้ ข้อดีคือทำให้ระบบมีเสถียรภาพและมีประสิทธิภาพสูง ข้อเสียคือไม่สามารถใช้งานเครื่องที่มีราคาสูงได้

2. **เครื่องบริการแบบไม่อุทิศ (Non - Dedicated Server)** หมายถึงเครื่องบริการยังสามารถใช้งานได้ตามปกติเหมือนเครื่องลูกข่าย ซึ่งมีข้อเสียที่สำคัญคือมีประสิทธิภาพของเครือข่ายจะลดลง ทำให้วิธีนี้ไม่เป็นที่นิยมในการใช้งาน

2.1.2 เครือข่ายแบบเท่าเทียม (Peer – to – Peer Network)

เป็นการเชื่อมต่อที่เครื่องทุกเครื่องในระบบเครือข่ายมีสถานะเท่าเทียมกันหมด โดยเครื่องทุกเครื่องสามารถเป็นได้ทั้งเครื่องผู้ใช้และเครื่องบริการในขณะใดขณะหนึ่ง นั่นคือเครื่องทุกเครื่องเปรียบเสมือนกับเป็นเครื่องบริการแบบไม่อุทิศ (Non - Dedicated Server) นั่นเอง ในระบบเครือข่ายประเภทนี้การติดต่อระหว่างแต่ละเครื่องจะสามารถติดต่อกันได้โดยตรง

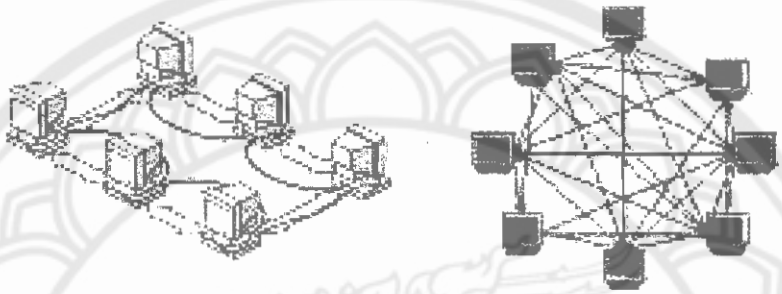
Network	Advantages	Disadvantages
Server - Based	• มีประสิทธิภาพสูงกว่าโดยเฉพาะอย่างยิ่งถ้าเป็นแบบ Dedicates Server • การดูแลระบบสามารถทำได้ง่ายกว่า	• เสียค่าใช้จ่ายสูงสำหรับเครื่อง server โดยเฉพาะอย่างยิ่งหารเป็นแบบ Dedicates Server ซึ่งไม่สามารถนำไปใช้งานอย่างอื่นได้ • ไม่สามารถใช้งานทรัพยากรที่เชื่อมอยู่กับ Workstation ได้ • ถ้า Server เสียระบบจะหยุดหมด
Peer - to - Peer	• สามารถใช้งานทรัพยากรซึ่งเชื่อมอยู่กับเครื่องใด ๆ ในเครือข่าย • ประหยัดค่าใช้จ่ายในส่วนของ Server • สามารถกระจายโปรแกรมประยุกต์ไปไว้ยังเครื่องต่าง ๆ เพื่อลดการจราจรในเครือข่ายได้	• การดูแลระบบทำได้ยาก เนื่องจากทรัพยากรกระจัดกระจายกันไปในเครื่องต่าง ๆ • มีประสิทธิภาพที่ต่ำกว่าแบบ Server - based มาก • เครื่องทุกเครื่องต้องมีหน่วยความจำและประสิทธิภาพสูงกว่าเครื่อง ในแบบ Server - based

ตารางที่ 2.1 ตารางเปรียบเทียบการเชื่อมต่อแบบ Server - based เทียบกับ Peer - to – Peer

2.2 โครงสร้างโดยรวมของ Network

2.2.1 Decentralized network

Decentralized network เป็นเทคโนโลยีที่นำเอาคอมพิวเตอร์ทุกเครื่องมาเชื่อมต่อกันโดยไม่มี Server เป็นศูนย์กลางในการเชื่อมต่อ แต่ Client แต่ละตัวจะทำหน้าที่เป็น Server หรือเลือกอีกอย่างหนึ่งว่า Client – Server และ Client แต่ละตัวจะมี Data base หรือข้อมูลที่ใช้สำหรับ Share อยู่ในตัวของของ Client เอง ตัวอย่างของ Decentralized network เช่น Peer – to – peer network



2.2.1(ก) 2.2.1(ข)

รูปที่ 2.1 ลักษณะโครงสร้างของ Decentralized network

2.2.2 Centralized network

Centralized network เป็นเทคโนโลยีที่นำเอาคอมพิวเตอร์มาเชื่อมต่อกัน โดยมีเครื่อง Server เป็นศูนย์กลางในการเชื่อมต่อ และ Share ทรัพยากรให้กับ Client

2.2.2 (ก) 2.2.2(ข)

รูปที่ 2.2 ลักษณะโครงสร้างของ Centralized network

2.3 พื้นฐานเกี่ยวกับ โปรโตคอล TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) เป็นชุดของโปรโตคอลที่ถูกใช้ในการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต โดยมีวัตถุประสงค์เพื่อให้สามารถใช้สื่อสารจากต้นทางข้ามเครือข่ายไปยังปลายทางได้ และสามารถหาเส้นทางที่จะส่งข้อมูลไปได้เองโดยอัตโนมัติ ถึงแม้ว่าในระหว่างทางอาจจะผ่านเครือข่ายที่มีปัญหา โปรโตคอลก็ยังคงหาเส้นทางอื่นในการส่งผ่านข้อมูลไปให้ถึงปลายทางได้

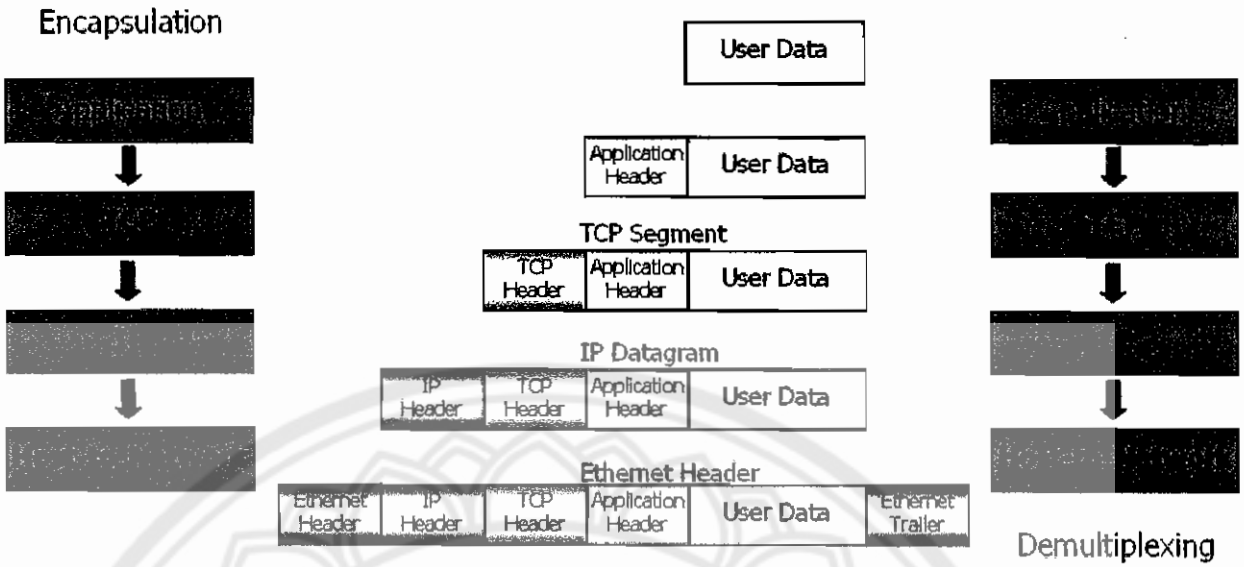
ชุดโปรโตคอลนี้ได้รับการพัฒนามาตั้งแต่ปี 1960 ซึ่งถูกใช้เป็นครั้งแรกในเครือข่าย ARPANET ซึ่งต่อมาได้ขยายการเชื่อมต่อไปทั่วโลกเป็นเครือข่ายอินเตอร์เน็ต ทำให้ TCP/IP เป็นที่ยอมรับอย่างกว้างขวางจนถึงปัจจุบัน

TCP/IP มีจุดประสงค์ของการสื่อสารตามมาตรฐาน สามประการคือ

1. เพื่อใช้ติดต่อสื่อสารระหว่างระบบที่มีความแตกต่างกัน
2. ความสามารถในการแก้ไขปัญหาที่เกิดขึ้นในระบบเครือข่าย เช่นในกรณีที่ผู้ส่งและผู้รับยังคงมีการติดต่อกันอยู่ แต่โหนดกลางที่ใช้เป็นผู้ช่วยรับ-ส่งเกิดเสียหายใช้การไม่ได้ หรือสายสื่อสารบางช่วงถูกตัดขาด กฎการสื่อสารนี้จะต้องสามารถจัดหาทางเลือกอื่นเพื่อทำให้การสื่อสารดำเนินต่อไปได้โดยอัตโนมัติ
3. มีความคล่องตัวต่อการสื่อสารข้อมูลได้หลายชนิดทั้งแบบที่ไม่มีความเร่งด่วน เช่น การจัดส่งแฟ้มข้อมูล และแบบที่ต้องการรับประกันความเร่งด่วนของข้อมูล เช่น การสื่อสารแบบ real-time และทั้งการสื่อสารแบบเสียง (Voice) และข้อมูล (data)

2.3.1Encapsulation/Demultiplexing

การส่งข้อมูลผ่านในแต่ละเลเยอร์ แต่ละเลเยอร์จะทำการประกอบข้อมูลที่ได้รับมา กับข้อมูลส่วนควบคุมซึ่งถูกนำมาไว้ในส่วนหัวของข้อมูลเรียกว่า Header ภายใน Header จะบรรจุข้อมูลที่สำคัญของโปรโตคอลที่ทำการ Encapsulate เมื่อผู้รับได้รับข้อมูล ก็จะเกิดกระบวนการทำงานย้อนกลับคือโปรโตคอลเดียวกัน ทางฝั่งผู้รับก็จะได้รับข้อมูลส่วนที่เป็น Header ก่อนและนำไปประมวลและทราบว่าข้อมูลที่ตามมามีลักษณะอย่างไร ซึ่งกระบวนการย้อนกลับนี้เรียกว่า Demultiplexing

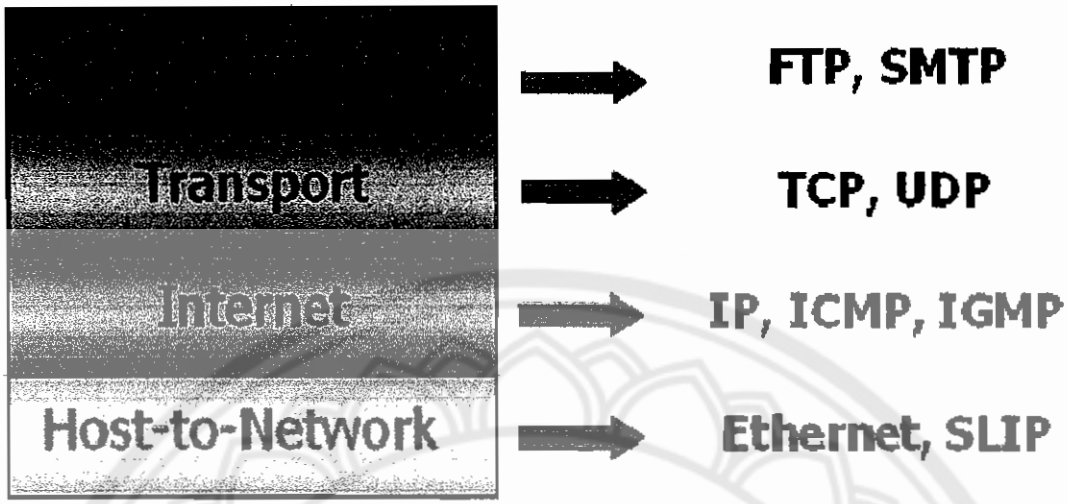


รูปที่ 2.3 ขั้นตอนการ Encapsulation และ Demultiplexing

ข้อมูลที่ผ่านการ Encapsulate ในแต่ละเลเยอร์มีชื่อเรียกแตกต่างกัน ดังนี้

- ข้อมูลที่มาจาก User หรือก็คือข้อมูลที่ User เป็นผู้ป้อนให้กับ Application เรียกว่า User Data
- เมื่อแอพพลิเคชั่นได้รับข้อมูลจาก User ก็จะนำมาประกอบกับส่วนหัวของแอพพลิเคชั่น เรียกว่า Application Data และส่งต่อไปยังโปรโตคอล TCP
- เมื่อโปรโตคอล TCP ได้รับ Application Data ก็จะนำมารวมกับ Header ของ โปรโตคอล TCP เรียกว่า TCP Segment และส่งต่อไปยังโปรโตคอล IP
- เมื่อโปรโตคอล IP ได้รับ TCP Segment ก็จะนำมารวมกับ Header ของ โปรโตคอล IP เรียกว่า IP Datagram และส่งต่อไปยังเลเยอร์ Host-to-Network Layer
- ในระดับ Host-to-Network จะนำ IP Datagram มาเพิ่มส่วน Error Correction และ flag เรียกว่า Ethernet Frame ก่อนจะแปลงข้อมูลเป็นสัญญาณไฟฟ้า ส่งผ่านสายสัญญาณที่เชื่อมโยงอยู่ต่อไป

ในแต่ละเลเยอร์ของโครงสร้าง TCP/IP สามารถอธิบายได้ดังนี้



รูปที่ 2.4 โครงสร้าง TCP/IP

2.3.1.1ชั้นโฮสต์-เครือข่าย(Host-to-NetworkLayer)

โพรโตคอลสำหรับการควบคุมการสื่อสารในชั้นนี้เป็นสิ่งที่ไม่มีการกำหนดรายละเอียดอย่างเป็นทางการ หน้าที่หลักคือการรับข้อมูลจากชั้นสื่อสาร IP มาแล้วส่งไปยังโหนดที่ระบุไว้ในเส้นทางเดินข้อมูลทางด้านผู้รับก็จะทำงานในทางกลับกัน คือรับข้อมูลจากสายสื่อสารแล้วนำส่งให้กับโปรแกรมในชั้นสื่อสาร

2.3.1.2ชั้นสื่อสารอินเทอร์เน็ต(TheInternetLayer)

ใช้ประเภทของระบบการสื่อสารที่เรียกว่า ระบบเครือข่ายแบบสลับช่องสื่อสารระดับแพ็กเก็ต (packet-switching network) ซึ่งเป็นการติดต่อแบบไม่ต่อเนื่อง (Connectionless) หลักการทำงานคือการปล่อยให้ข้อมูลขนาดเล็กที่เรียกว่า แพ็กเก็ต (Packet) สามารถไหลจากโหนดผู้ส่งไปตามโหนดต่างๆ ในระบบจนถึงจุดหมายปลายทางได้โดยอิสระ หากว่ามีการส่งแพ็กเก็ตออกมาเป็นชุดโดยมีจุดหมายปลายทางเดียวกันในระหว่างการเดินทางในเครือข่าย แพ็กเก็ตแต่ละตัวในชุดนี้ก็จะเป็นอิสระแก่กันและกัน ดังนั้น แพ็กเก็ตที่ส่งไปถึงปลายทางอาจจะไม่เป็นไปตามลำดับก็ได้

a.IP(InternetProtocol)

IP เป็นโปรโตคอลในระดับเน็ตเวิร์คเลเยอร์ ทำหน้าที่จัดการเกี่ยวกับแอดเดรสและข้อมูล และควบคุมการส่งข้อมูลบางอย่างที่ใช้ในการหาเส้นทางของแพ็กเก็ต ซึ่งกลไกในการหาเส้นทางของ IP จะมีความสามารถในการหาเส้นทางที่ดีที่สุด และสามารถเปลี่ยนแปลงเส้นทางได้ในระหว่างการส่งข้อมูล และมีระบบการแยกและประกอบดาต้าแกรม (datagram) เพื่อรองรับการส่งข้อมูลระดับ data link ที่มี

ขนาด MTU (Maximum Transmission Unit) ที่แตกต่างกัน ทำให้สามารถนำ IP ไปใช้บนโปรโตคอลอื่นได้หลากหลาย เช่น Ethernet ,Token Ring หรือ Apple Talk

การเชื่อมต่อของ IP เพื่อทำการส่งข้อมูล จะเป็นแบบ connectionless หรือเกิดเส้นทางการเชื่อมต่อในทุกๆครั้งของการส่งข้อมูล 1 ดาต้าแกรม โดยจะไม่ทราบถึงข้อมูลดาต้าแกรมที่ส่งก่อนหน้าหรือส่งตามมา แต่การส่งข้อมูลใน 1 ดาต้าแกรม อาจจะเกิดการส่งได้หลายครั้งในกรณีที่มีการแบ่งข้อมูลออกเป็นส่วนย่อยๆ (fragmentation) และถูกนำไปรวมเป็นดาต้าแกรมเดิมเมื่อถึงปลายทาง

รูปที่ 2.5 IP Headers

เฮดเดอร์ของ IP โดยปกติจะมีขนาด 20 bytes ยกเว้นในกรณีที่มีการเพิ่ม Option บางอย่าง ฟิลด์ของเฮดเดอร์ IP จะมีความหมายดังนี้

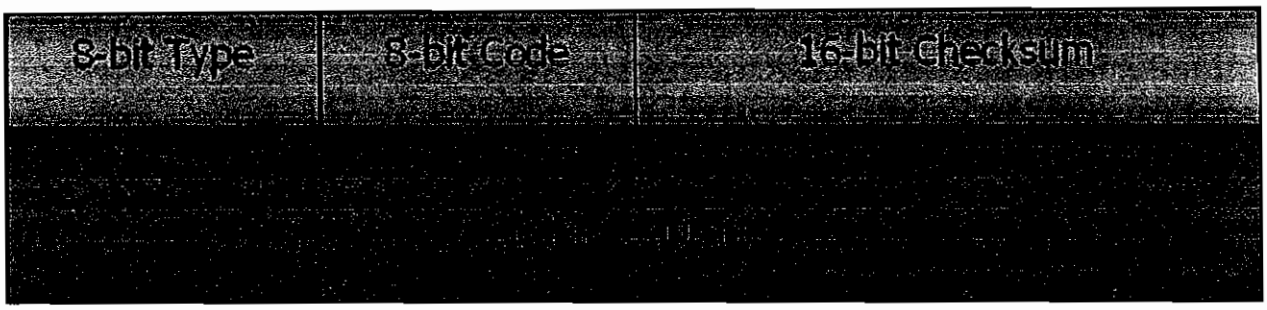
a. **Version** : หมายเลขเวอร์ชันของโปรโตคอล ที่ใช้งานในปัจจุบันคือ เวอร์ชัน 4 (IPv4) และเวอร์ชัน 6 (IPv6)
b. **Header Length** : ความยาวของเฮดเดอร์ โดยทั่วไปถ้าไม่มีส่วน option จะมีค่าเป็น 5 (5*32 bit)
c. **Type of Service (TOS)** : ใช้เป็นข้อมูลสำหรับเราเตอร์ในการตัดสินใจเลือกการเราต์ข้อมูลในแต่ละดาต้าแกรม แต่ในปัจจุบันไม่ได้มีการนำไปใช้งานแล้ว
d. **Length** : ความยาวทั้งหมดเป็นจำนวนไบต์ของดาต้าแกรม ซึ่งด้วยขนาด 16 บิตของฟิลด์ จะหมายถึงความยาวสูงสุดของดาต้าแกรม คือ 65535 byte (64k) แต่ในการส่งข้อมูลจริง ข้อมูลจะถูกแยกเป็นส่วนๆตามขนาดของ MTU ที่

กำหนดในลิงค์เลเยอร์ และนำมารวมกันอีกครั้งเมื่อส่งถึงปลายทาง แอพพลิเคชั่นส่วนใหญ่จะมีขนาดของดาต้าแกรมไม่เกิน 512 byte

e. **Identification** : เป็นหมายเลขของดาต้าแกรมในกรณีที่มีการแยกดาต้าแกรม เมื่อข้อมูลส่งถึงปลายทางจะนำข้อมูลที่มี identification เดียวกันมารวมกัน
f. **Flag** : ใช้ในกรณีที่มีการแยกดาต้าแกรม
g. **Fragment offset** : ใช้ในการกำหนดตำแหน่งข้อมูลในดาต้าแกรมที่มีการแยกส่วน เพื่อให้สามารถนำกลับมาเรียงต่อกันได้อย่างถูกต้อง
h. **Time to live (TTL)** : กำหนดจำนวนครั้งที่มากที่สุดที่ดาต้าแกรมจะถูกส่งระหว่าง hop (การส่งผ่านข้อมูลระหว่างเน็ตเวิร์ค) เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลโดยไม่สิ้นสุด โดยเมื่อข้อมูลถูกส่งไป 1 hop จะทำการลดค่า TTL ลง 1 เมื่อค่าของ TTL เป็น 0 และข้อมูลยังไม่ถึงปลายทาง ข้อมูลนั้นจะถูกยกเลิก และเราเตอร์สุดท้ายจะส่งข้อมูล ICMP แจ้งกลับมายังต้นทางว่าเกิด time out ในระหว่างการส่งข้อมูล
i. **Protocol** : ระบุโปรโตคอลที่ส่งในดาต้าแกรม เช่น TCP ,UDP หรือ ICMP
j. **Header checksum** : ใช้ในการตรวจสอบความถูกต้องของข้อมูลในเฮดเดอร์
k. **Source IP address** : หมายเลข IP ของผู้ส่งข้อมูล
l. **Destination IP address** : หมายเลข IP ของผู้รับข้อมูล
m. **Data** : ข้อมูลจากโปรโตคอลระดับบน

b.ICMP(InternetControlMessageProtocol)

ICMPเป็นโปรโตคอลที่ใช้ในการตรวจสอบและรายงานสถานภาพของดาต้าแกรม(Datagram) ในกรณีที่เกิดปัญหากับดาต้าแกรม เช่น เราเตอร์ไม่สามารถส่งดาต้าแกรมไปถึงปลายทางได้ ICMP จะถูกส่งออกไปยังโฮสต้นทางเพื่อรายงานข้อผิดพลาดที่เกิดขึ้นอย่างไรก็ดี ไม่มีอะไรรับประกันได้ว่า ICMP Message ที่ส่งไปจะถึงผู้รับจริงหรือไม่ หากมีการส่งดาต้าแกรมออกไปแล้วไม่มี ICMP Message ฟ้อง Error กลับมา ก็แปลความหมายได้สองกรณีคือ ข้อมูลถูกส่งไปถึงปลายทางอย่างเรียบร้อย หรืออาจจะมีปัญหา ในการสื่อสารทั้งการส่งดาต้าแกรม และ ICMP Message ที่ส่งกลับมาก็มีปัญหาระว่างทางก็ได้ ICMP จึงเป็นโปรโตคอลที่ไม่มีความน่าเชื่อถือ (unreliable) ซึ่งจะเป็นหน้าที่ของ โปรโตคอลในระดับสูงกว่า Network Layer ในการจัดการให้การสื่อสารนั้นๆ มีความน่าเชื่อถือ ในส่วนของ ICMP Message จะประกอบด้วย Type ขนาด 8 บิต Checksum ขนาด 16 บิต และส่วนของ Content ซึ่งจะมีขนาดแตกต่างกันไปตาม Type และ Code ดังรูป



รูปที่ 2.6 ICMP Headers

2.3.1.3ชั้นสื่อสารนำส่งข้อมูล(TransportLayer)

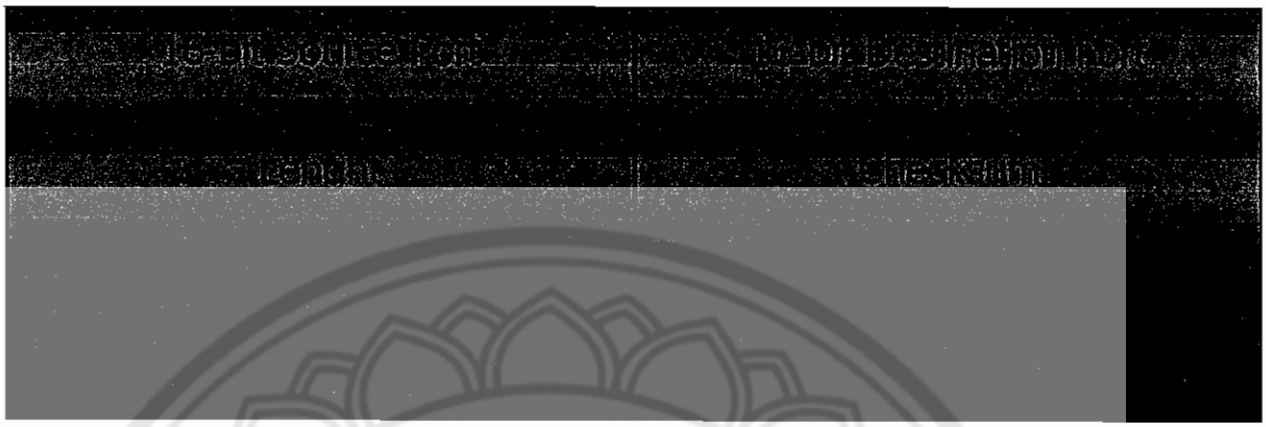
แบ่งเป็นโพรโตคอล 2 ชนิดตามลักษณะ ลักษณะแรกเรียกว่า Transmission Control Protocol (TCP) เป็นแบบที่มีการกำหนดช่วงการสื่อสารตลอดระยะเวลาการสื่อสาร (connection-oriented) ซึ่งจะยอมให้มีการส่งข้อมูลเป็นแบบ Byte stream ที่ไว้ใจได้โดยไม่มีข้อผิดพลาด ข้อมูลที่มีปริมาณมากจะถูกแบ่งออกเป็นส่วนเล็กๆ เรียกว่า message ซึ่งจะถูกส่งไปยังผู้รับผ่านทางชั้นสื่อสารของอินเทอร์เน็ต ทางฝ่ายผู้รับจะนำ message มาเรียงต่อกันตามลำดับเป็นข้อมูลตัวเดิม TCP ยังมีความสามารถในการควบคุมการไหลของข้อมูลเพื่อป้องกันไม่ให้ผู้ส่ง ส่งข้อมูลเร็วเกินกว่าที่ผู้รับจะทำงานได้ทันอีกด้วย

โปรโตคอลการนำส่งข้อมูลแบบที่สองเรียกว่า UDP (User Datagram Protocol) เป็นการติดต่อแบบไม่ต่อเนื่อง (connectionless) มีการตรวจสอบความถูกต้องของข้อมูลแต่จะไม่มีการแจ้งกลับไปยังผู้ส่ง จึงถือได้ว่าไม่มีการตรวจสอบความถูกต้องของข้อมูล อย่างไรก็ตาม วิธีการนี้มีข้อดีในด้านความรวดเร็วในการส่งข้อมูล จึงนิยมใช้ในระบบผู้ให้และผู้ใช้บริการ (client/server system) ซึ่งมีการสื่อสารแบบ ถาม/ตอบ (request/reply) นอกจากนั้นยังใช้ในการส่งข้อมูลประเภทภาพเคลื่อนไหวหรือการส่งเสียง (voice) ทางอินเทอร์เน็ต

a.UDP:(UserDatagramProtocol)

เป็นโปรโตคอลที่อยู่ใน Transport Layer เมื่อเทียบกับโมเดล OSI โดยการส่งข้อมูลของ UDP นั้นจะเป็นการส่งครั้งละ 1 ชุดข้อมูล เรียกว่า UDP datagram ซึ่งจะไม่มีความสัมพันธ์กันระหว่างดาต้าแกรมและจะไม่มีกลไกการตรวจสอบความสำเร็จในการรับส่งข้อมูล กลไกการตรวจสอบโดย checksum ของ UDP นั้นเพื่อเป็นการป้องกันข้อมูลที่อาจจะถูกแก้ไข หรือมีความผิดพลาดระหว่างการส่ง และหากเกิดเหตุการณ์ดังกล่าว ปลายทางจะได้รู้ว่ามีข้อผิดพลาดเกิดขึ้น แต่มันจะเป็นการตรวจสอบเพียงฝ่ายเดียวเท่านั้น โดยในข้อกำหนดของ UDP หากพบว่า Checksum Error ก็ให้ผู้รับปลายทางทำการทิ้งข้อมูลนั้น แต่จะไม่มีการแจ้งกลับไปยังผู้ส่งแต่อย่างใด การรับส่งข้อมูลแต่ละครั้งหากเกิดข้อผิดพลาดในระดับ IP เช่น ส่งไม่ถึง, หมดเวลา ผู้ส่งจะได้รับ Error Message จากระดับ IP เป็น ICMP Error

Message แต่เมื่อข้อมูลส่งถึงปลายทางถูกต้อง แต่เกิดข้อผิดพลาดในส่วนของ UDP เอง จะไม่มีการยืนยัน หรือแจ้งให้ผู้ส่งทราบแต่อย่างใด



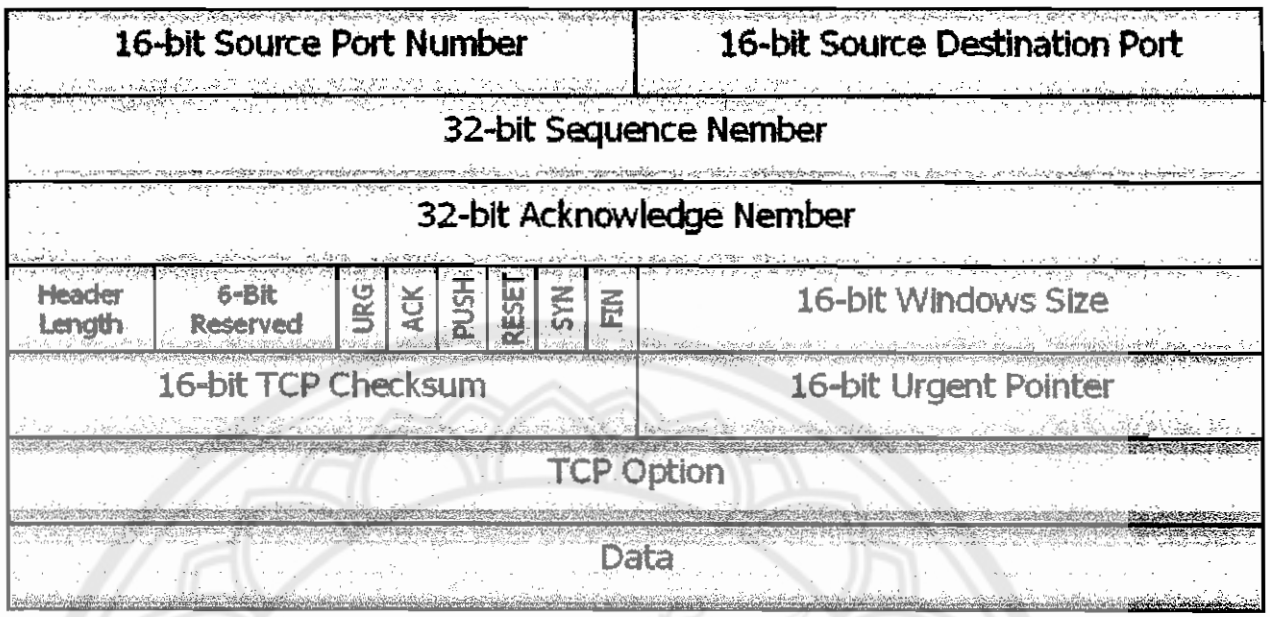
รูปที่ 2.7 UDP Headers

มีรายละเอียด ดังนี้

- **Source Port Number** : หมายเลขพอร์ตต้นทางที่ส่งดาต้าแกรมนี้
- **Destination Port Number** : หมายเลขพอร์ตปลายทางที่จะเป็นผู้รับดาต้าแกรม
- **UDP Length** : ความยาวของดาต้าแกรม ทั้งส่วน Header และ data นั่นหมายความว่า ค่าที่น้อยที่สุดในฟิลด์นี้คือ 8 ซึ่งเป็นขนาดของ Header
- **Checksum** : เป็นตัวตรวจสอบความถูกต้องของ UDP datagram และจะนำข้อมูลบางส่วนใน IP Header มาคำนวณด้วย

h. TCP: (Transmission Control Protocol)

อยู่ใน Transport Layer เช่นเดียวกับ UDP ทำหน้าที่จัดการและควบคุมการรับส่งข้อมูล ซึ่งมีความสามารถและรายละเอียดมากกว่า UDP โดยดาต้าแกรมของ TCP จะมีความสัมพันธ์ต่อเนื่องกัน และมีกลไกควบคุมการรับส่งข้อมูลให้มีความถูกต้อง (reliable) และมีการสื่อสารอย่างเป็นกระบวนการ (connection-oriented)



รูปที่ 2.8 TCP Headers

มีรายละเอียด ดังนี้

- **Source Port Number** : หมายเลขพอร์ตต้นทางที่ส่งดาต้าแกรมนี้
- **Destination Port Number** : หมายเลขพอร์ตปลายทางที่จะเป็นผู้รับดาต้าแกรม
- **Sequence Number** : ฟิลด์ที่ระบุหมายเลขลำดับอ้างอิงในการสื่อสารข้อมูลแต่ละครั้ง เพื่อใช้ในการแยกแยะว่าเป็นข้อมูลของชุดใด และนำมาจัดลำดับได้ถูกต้อง
- **Acknowledgment Number** : ทำหน้าที่เช่นเดียวกับ Sequence Number แต่จะใช้ในการตอบรับ
- **Header Length** : โดยปกติความยาวของเฮดเดอร์ TCP จะมีความยาว 20 ไบต์ แต่อาจจะมากกว่านั้น ถ้ามีข้อมูลในฟิลด์ option แต่ต้องไม่เกิน 60 ไบต์
- **Flag** : เป็นข้อมูลระดับบิตที่อยู่ในเฮดเดอร์ TCP โดยใช้เป็นตัวบอกคุณสมบัติของแพ็กเก็ต TCP ขณะนั้นๆ และใช้เป็นตัวควบคุมจังหวะการรับส่งข้อมูลด้วย ซึ่ง Flag มีอยู่ทั้งหมด 6 บิต แบ่งได้ดังนี้

Type	Description
URG	ใช้บอกความหมายว่าเป็นข้อมูลด่วน และมีข้อมูลพิเศษมาด้วย (อยู่ใน Urgent pointer)
ACK	แสดงว่าข้อมูลในฟิลด์ Acknowledge Number นำมาใช้งานได้
DSH	เป็นการแจ้งให้ผู้รับข้อมูลทราบว่าควรจะส่งข้อมูล Segment นี้ไปยัง Application ที่กำลังรออยู่โดยเร็ว
RST	ยกเลิกการติดต่อ (reset) เนื่องจากในกรณีที่เกิดการสับสนขึ้นด้วยเหตุผลต่างๆ เช่น โฮสต์มีปัญหา ให้เริ่มต้นสื่อสารกันใหม่
SYN	ใช้ในการเริ่มต้นขอติดต่อกับปลายทาง
FIN	ใช้ส่งเพื่อแจ้งให้ปลายทางทราบว่ายุติการติดต่อ

ตารางที่ 2.2 รายละเอียดของ Flag แต่ละชนิด

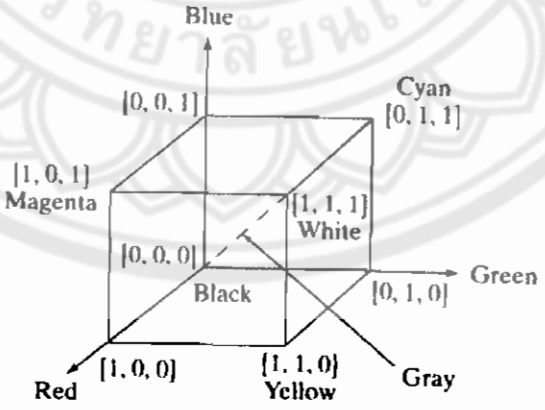
Flag ในเฮดเดอร์ของ TCP มีความสำคัญในการกำหนดการทำงานของ TCP segment เนื่องจากข้อมูลในเฮดเดอร์ของ TCP จะมีข้อมูลครบถ้วนทั้งการรับและการส่งข้อมูล ซึ่งในการทำงานแต่ละอย่างจะมีการใช้งานฟิลด์ไม่เหมือนกัน flag จะเป็นตัวกำหนดว่าให้ใช้งานฟิลด์ไหน เช่น ฟิลด์ Acknowledgment number จะไม่ถูกใช้ในขั้นตอนการเริ่มต้นการเชื่อมต่อ แต่จะมีข้อมูลในฟิลด์ ซึ่งเป็นข้อมูลที่ไม่มีความหมายใดๆ ซึ่งถ้าไม่มี flag เป็นตัวกำหนดก้ออาจจะมีการนำข้อมูลมาใช้ และก่อให้เกิดความผิดพลาดได้

2.4 มาตรฐานของสี

มาตรฐานของสีที่ใช้อยู่ในปัจจุบันมีอยู่หลายระบบด้วยกัน ทั้งนี้จะขึ้นอยู่กับการนำไปใช้ แต่โดยทั่วไปแล้ทุกมาตรฐานจะมีแนวคิดเดียวกันคือ การแทนจุดสีด้วยจุดที่อยู่ภายในสเปส 3 มิติ โดยจะมีแกนอ้างอิงสำหรับจุดสี นั้นในสเปสซึ่งแต่ละแกนจะมีความเป็นอิสระต่อกัน ตัวอย่างเช่นในระบบ RGB จะมีแกนสีคือ แกนสีแดง เขียว และน้ำเงินในระบบ HLS จะมีแกนเป็น ค่าสี (hue) ความสว่าง (lightness) และความบริสุทธิ์ของสี (saturation) ตัวอย่างระบบสีที่เราสนใจในการพิจารณาคือ ระบบ RGB (Red Green Blue)

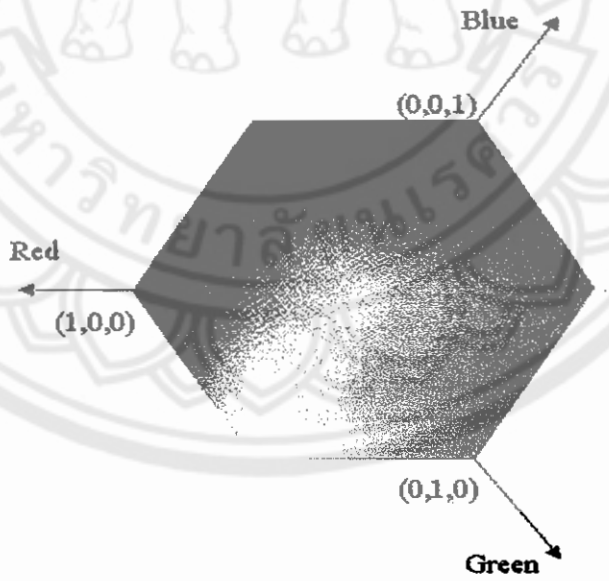
2.4.1 ระบบสี RGB

ระบบสี RGB เป็นระบบสีที่เกิดจากการรวมกันของแสงสีแดง เขียวและน้ำเงินโดยมีการรวมกันแบบ Additive ก็จะเป็นกราฟที่มี 3 แกน โดยแต่ละแกนก็จะแทนค่าสี R, G, B โดยไล่จากค่า 0 ที่มีความเข้มสีมาก จนไปถึงค่า 1 ที่มีความเข้มสีน้อยดังรูป



รูปที่ 2.9 RGB coordinates system

การบอกหรือระบุสีต่างๆ โดยใช้ "พิกัด" หรือ "ค่าของแม่สี" สามค่า ในการแทนค่าของสีต่างๆ ที่เราใช้งาน เช่น ในงานพิมพ์แยกสี จึงทำให้ดูเหมือนว่า "สี" นั้นมี สามมิติ ตามมิติของ "แม่สี"ทั้งสาม คือ แดง เหลือง และฟ้า โดยสมมติ ให้เป็นค่า R Y B ต่างๆ (ตัวย่อของ Red Yellow และ Blue) หรือเรา อาจจะใช้ "ค่าแม่สีของแสง" คือ R G B (Red Green Blue) ในการสร้างแสงสีจาก "แม่สีของแสง" สามสี ซึ่งต่างจาก "แม่สีของสี" ตรงที่สีเหลืองกับ สีเขียว กล่าวคือแม่สีของสีจะเป็น "สีเหลือง" แต่ แม่สีของแสงจะเป็น "สีเขียว ซึ่งสามารถเทียบเคียงได้กับการระบุพิกัดแบบใช้ค่า X Y Z ในการระบุตำแหน่งของวัตถุต่างๆในระบบพิกัดแบบคาร์ทีเซียนนั่นเอง เมื่อเป็นเช่นนี้ จึงทำให้ดูเหมือนว่า สี ต่างๆนั้นมี สามมิติ คือค่า R Y B หรือ ค่า R G B ต่างๆ ตามส่วนประกอบของมัน ซึ่งเราก็สามารถ สร้าง "ลูกบาศก์ของสี" (Color Cube) ขึ้นได้จากค่า R Y Bหรือ R G Bทั้งสามค่านั้นได้ ดังนั้นสีจึงดูกลายเหมือนเป็น"สามมิติ" ไปในทันที ทั้งๆที่ความจริงแล้ว สีต่างๆนั้นมันอยู่ในมิติเดียวกัน คือถ้าหาก เราเอาสีต่างๆมาเรียงต่อๆกัน มันจะ กลายเป็น "แถบสเปคตรัมของแสง"หรือ"แถบสีรุ้ง" อย่างที่เราคุ้นเคยกันดี เพราะในธรรมชาติมันก็เป็นเช่นนั้นจริงๆคือ เวลาที่เราเห็น รุ้งกินน้ำบนท้องฟ้าหรือ เราเห็นแสงหักเหผ่านแท่งแก้วปริซึมที่กระจายออกเป็นแถบสีรุ้ง ตาม Spectrum ของแสงคือ VIBGYOR หรือ ROYGBIV กล่าวคือ แดง ส้ม เหลือง เขียว น้ำเงิน คราม ม่วง(Red Orange Yellow Green Blue Indigo Violet)เรียงลำดับตามความยาวคลื่นหรือความถี่ของแสงนั่นเอง นั่นคือแสงสีต่างๆนั้นต่างอยู่ในมิติเชิงเส้นอันเดียวกัน ดังนั้นการที่ เกิดค่า R Y B หรือ R G B ของสีต่างๆ นั้นจึงเป็น "รงคมิติ" หรือเกิด"มิติเทียม" ของสี ขึ้น จากส่วนผสมของแม่สีทั้งสาม ซึ่งไม่ใช่มิติที่แท้จริง!



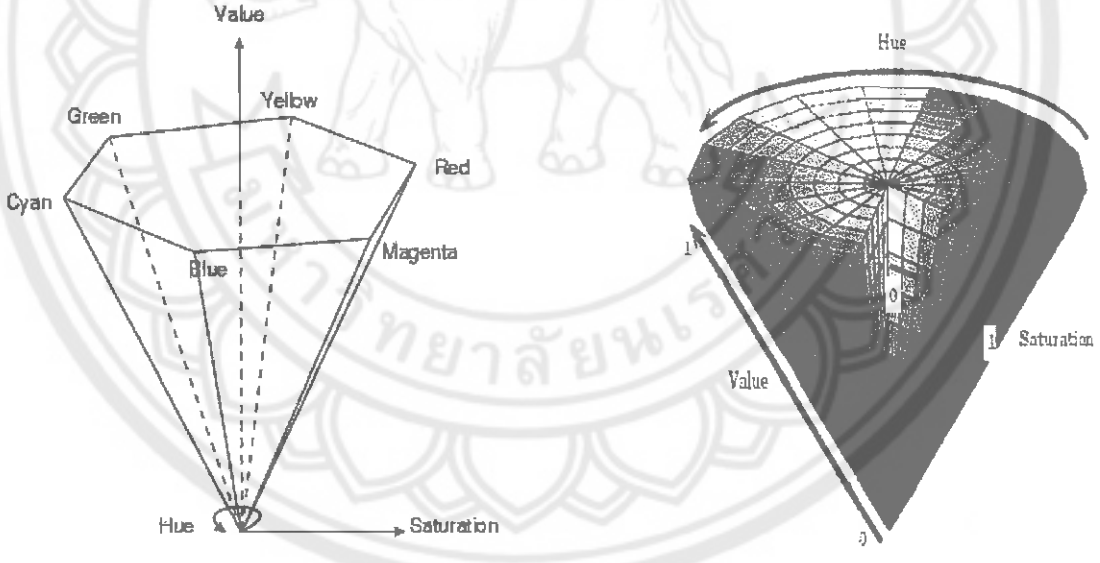
รูปที่ 2.10 RGB color model เมื่อมองจากทางด้าน White

2.5 ระบบสี HSV/I

HSV (Hue Saturation and Value) หรือ HIS (Hue, Saturation and Intensity) ยึดหลักการแลกเปลี่ยนใน ความเข้มแสง ค่า HSV สามารถรับจาก ค่า RGB และ HSV color space สามารถหาค่าง่ายกว่า RGB color space

สีเป็นรูปแบบที่เห็นได้ชัดที่สุดสำหรับข้อมูลรูปภาพ เนื่องจากสิ่งแรกที่สังเกตเห็นในรูปภาพคือสี ก่อนที่สามารถจะวิเคราะห์รายละเอียดภายในรูปภาพได้ ในการดึงข้อมูลเพื่อมาเปรียบเทียบความแตกต่างระหว่างสีจะต้องมีโมเดลของสีที่ใช้ อย่างเช่น การแบ่งช่องว่างระหว่างสีให้สอดคล้องกับการแยกความแตกต่างของสีโดยสายตาของมนุษย์ โมเดลสีที่ใช้ชื่อว่า HSV (hue, saturation, value) ค่า hue หมายถึงโทนสี ค่า saturation หมายถึง ค่าความสดใสที่เกิดจากแสงขาว และค่า value แสดงถึงความเข้มแสง มนุษย์จะรับรู้สีจากโทนสี, ค่าความสดใส และความเข้มแสง

รูปภาพทุกรูปจะต้องแปลงจากโมเดลสี RGB ให้อยู่ในโมเดลสี HSV ค่าความเข้มแสงมีผลกระทบไม่มากต่อการมองเห็นของมนุษย์ การไม่พิจารณาค่าความเข้มแสงจะช่วยในการคำนวณให้เร็วขึ้นและเนื้อที่ในการเก็บข้อมูลน้อยลง ในการคำนวณจะนำค่าแผนภูมิของค่าโทนสีกับแผนภูมิของค่าความสดใสแปลงให้อยู่ในรูปเวกเตอร์



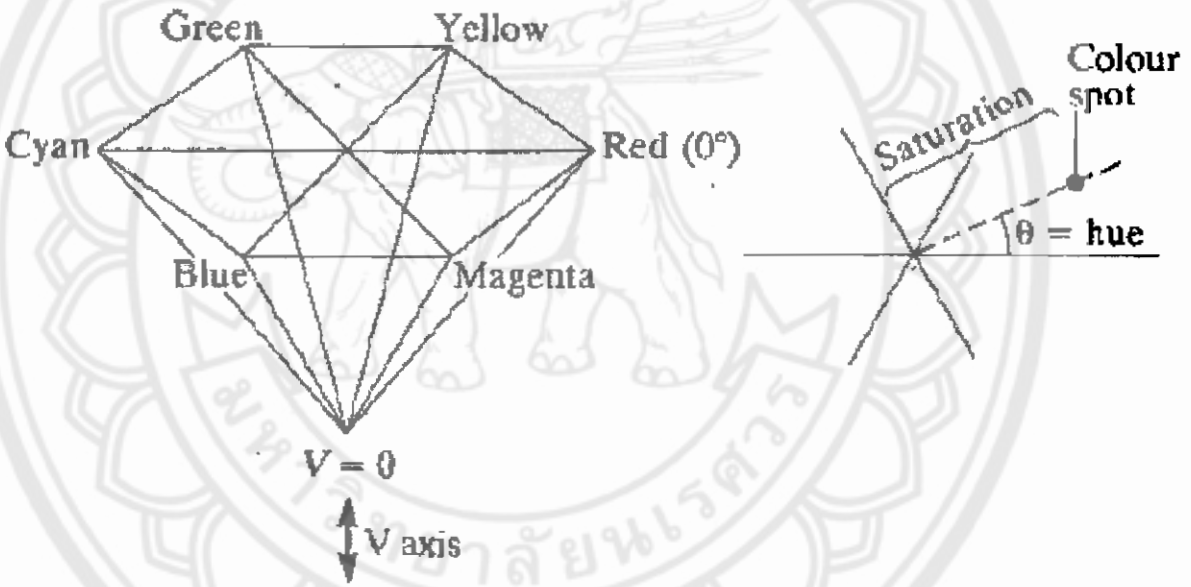
2.11(ก) HSV coordinates system **2.11(ข)** HSV color model

รูปที่ 2.11 เวกเตอร์ของแผนภูมิค่าความสดใสและค่าโทนสี

ระบบสี HSV (Hue Saturation Value) เป็นการพิจารณาสีโดยใช้ Hue Saturation และ Value ซึ่ง Hue คือค่าสีของสีหลัก(แดง เขียวและน้ำเงิน) ในทางปฏิบัติจะอยู่ระหว่าง 0 และ 255 ซึ่งถ้า Hue มีค่าเท่ากับ 0 จะแทนสีแดงและเมื่อ Hue มีค่าเพิ่มขึ้นเรื่อย ๆ สีก็จะเปลี่ยนแปลงไปตามสเปกตรัมของสีจนถึง 256 จึงจะกลับมาเป็นสีแดงอีกครั้งซึ่งสามารถแทนให้อยู่ในรูปขององศาได้ ดังนี้คือ สีแดง = 0 องศา สีเขียวเท่ากับ 120 องศา สีน้ำเงินเท่ากับ 240องศา

Hue สามารถคำนวณได้จากระบบสี RGB ได้ดังนี้

$$red_h = red - \min(red, green, blue)$$
$$green_h = green - \min(red, green, blue)$$
$$blue_h = blue - \min(red, green, blue) \quad (2.1)$$



รูปที่ 2.12 แสดงระบบสี HSV

จากลักษณะโมเดลของระบบ Hue พบว่าจะมีค่าอย่างน้อยหนึ่งค่าที่จะเท่ากับ 0 แต่ถ้ามีสองค่าเท่ากับ 0 แล้ว hue จะเป็นมุมของสี(ค่าสี)มีค่าเป็นไปตามสี่ที่สามและถ้าทั้งสามสีมีค่าเท่ากับ 0 แล้วจะทำให้ไม่มีค่าของ Hue หรือสีที่ได้จะมีค่าเท่ากับสีขาวนั่นเอง ตัวอย่างเช่น จอภาพขาว-ดำ ถ้าเกิดมีสีใดสีหนึ่งมีค่าเท่ากับ 0 จะทำให้ค่าสีที่ได้เป็นไปตามสีที่เหลือ การให้น้ำหนักในการพิจารณาเมื่อสีแดงมีค่าเท่ากับ 0

$$\frac{(240 x blue_h) + (120 x green_h)}{blue_h + green_h} \quad (2.2)$$

Saturation คือความบริสุทธิ์ของสีซึ่งถ้า Saturation มีค่าเท่ากับ 0 แล้วสีที่ได้จะไม่มี Hue ซึ่งจะเป็นสีขาว ล้วนแต่ถ้า Saturation มีค่าเท่ากับ 255 แสดงว่าจะไม่มีแสงสีขาวผสมอยู่เลย

Saturation สามารถคำนวณได้ดังนี้

$$Saturation = \frac{\max(red, green, blue) - \min(red, green, blue)}{\max(red, green, blue)} \quad (2.3)$$

Value คือความสว่างของสี ซึ่งสามารถวัดได้โดยค่าความเข้มของความสว่างของแต่ละสีที่ประกอบกัน สามารถคำนวณได้จาก

$$Value = \max(red, green, blue) \quad (2.4)$$

2.6 Content Base Image Retrieval

Color retrieval(Color histogram) เป็นเทคนิคที่สำคัญสำหรับการสร้างดัชนีฐานข้อมูลรูปภาพ และการค้นหา โดยหลักการคือ ทำการวิเคราะห์รูปภาพแต่ละรูปเพื่อคำนวณหากราฟแท่งแสดงค่าของสถิติความถี่สีซึ่งแสดงสัดส่วนของ pixels ของแต่ละสีภายในรูปภาพ กราฟแท่งแสดงค่าของสถิติความถี่สีสำหรับรูปภาพแต่ละรูปถูกวิเคราะห์แล้วจะถูกเก็บไว้ที่ฐานข้อมูลและเมื่อเวลาค้นหาผู้ใช้สามารถจะเลือกภาพเจาะจงลงไปอันใดอันหนึ่งตามสัดส่วนที่ต้องการของแต่ละสี (เช่น สีเขียว 75% สีแดง 10% และสีน้ำเงิน 15%) หรือส่งกลับรูปภาพตัวอย่างที่เลือกซึ่งกราฟแท่งแสดงค่าของสถิติความถี่สีถูกคำนวณไว้แล้วทำการการจับคู่รูปภาพที่กราฟแท่งแสดงค่าของสถิติความถี่สีมีความใกล้เคียงกันมากที่สุดแล้วคืนรูปภาพเหล่านั้นให้ผู้ใช้งาน พัฒนาโดย Swain และ Ballard ในปี1991 เทคนิคนี้ถูกใช้ในส่วนของระบบ CBIR ปัจจุบัน

2.6.1 Color histogram in RGB space

RGB space (สีแดง, สีเขียวและสีน้ำเงิน) สนับสนุนโปรแกรมเกี่ยวกับ image processing ดังนั้นจึงไม่ยากที่จะสร้างโปรแกรมซึ่งอ่านรูปภาพจาก Diskถึง RAM ของคอมพิวเตอร์

RGB space มีข้อเสียหาย 2 ข้อคือ

1. RGB space ไม่ใช่ perceptually รูปแบบเดียวกัน

2. ส่วนประกอบทั้งหมด (R, G, B) มีความสำคัญที่เท่ากันดังนั้นค่าเหล่านั้นต้อง บอกค่าจำนวนกับความประณีตเดียวกัน

ปัญหา Perceptual สามารถหลีกเลี่ยงโดยการใช้ weight matrix ระหว่าง 2 Histogram

$$d_{hist}(x, y) = (x-y)' A (x-y)$$

โดยที่ A คือ weight matrix x และ y คือกราฟแท่งแสดงค่าของสถิติความถี่ ถ้าตัวเลขที่แตกต่างของ binsทีสถานะแตกต่างกัน สถานะทั้งหมดมันก็ต้องการ matrixของมันเองตัวอย่างเช่น 64 bins histogram ก็จะต้อง weight matrix ซึ่งมีขนาดคือ 64*64

ปัญหา 2 วิธีผลรวม binsที่เป็นไปได้นั้นคือของจำนวนเต็ม (เช่น 13=1, 23= 8,33=27, 43=64, 53=125, 63=216, 73=343...)ขั้นตอนเหล่านี้เราสามารถหลีกเลี่ยงปัญหาโดยการกำหนด color mapเพื่อรูปภาพทุกรูปได้ใช้ร่วมกันก่อนที่ จะถูก histogramแปลงเป็นค่า color space ซึ่งตามความเป็นจริงค่า color spaceนี้ มีหน้าที่ตรงกันกับbinsและค่าของ Histogramสามารถได้รับการเพิ่มค่าที่ Histogram ระหว่างการแปลงซึ่งทำให้ Algorithmกระชับขึ้น

การทำ Normalize RGB

Intensity	I	= (R+G+B)/3
Normalize red	r	= R/(R+G+B)
Normalize green	g	= G/(R+G+B)
Normalize blue	b	= B/(R+G+B)
	L	= r+g+b

Histogram Distance Measure

Histogram: $h_{A,B,C}(a,b,c) = N \cdot \Pr ob(A=a, B=b, C=c)$

Histogram distance

Euclidean distance

$$d^2(h,g) = \sum_A \sum_B \sum_C (h(a,b,c) - g(a,b,c))^2$$

Intersection distance

$$d(h,g) = \frac{\sum_A \sum_B \sum_C \min(h(a,b,c), g(a,b,c))}{\min(|h|,|g|)}$$

Quadratic distance (Cross distance)

$$d(h,g) = (h-g)^t A (h-g)$$

Difference distance

$$dif(h,g) = \sqrt{(h_a - g_a)^2 + (h_b - g_b)^2 + (h_c - g_c)^2}$$

2.7 ทำความรู้จักกับ C#

ภาษา C# เป็นภาษาที่ถูกสร้างขึ้นมาเพื่อทำงานบน .NET Platform สร้างและมีการทำงานโดยใช้หลักการแบบ Object Oriented ได้อย่างสมบูรณ์ (ซึ่งเมื่อเปรียบเทียบกับ C++ ที่ยังทำงานในลักษณะของ Object Oriented Programming (OOP) ได้แค่บางส่วน) ซึ่งไลบรารี่ของC# ถูกสร้างขึ้นเพื่อที่ให้ทำงานได้ครอบคลุมตั้งแต่การสร้างรูปแบบการติดต่อแบบ GUI ไปจนถึงการแอ็คเซสฐานข้อมูลผ่านอินเตอร์เน็ตหรือแม้แต่การทำงานร่วมกับXML เพื่อทำให้การแลกเปลี่ยนข้อมูลระหว่างแอพพลิเคชั่นทำได้อย่างสมบูรณ์ไม่ว่าข้อมูลนั้นจะอยู่บนแพลทฟอร์มใดก็ตาม

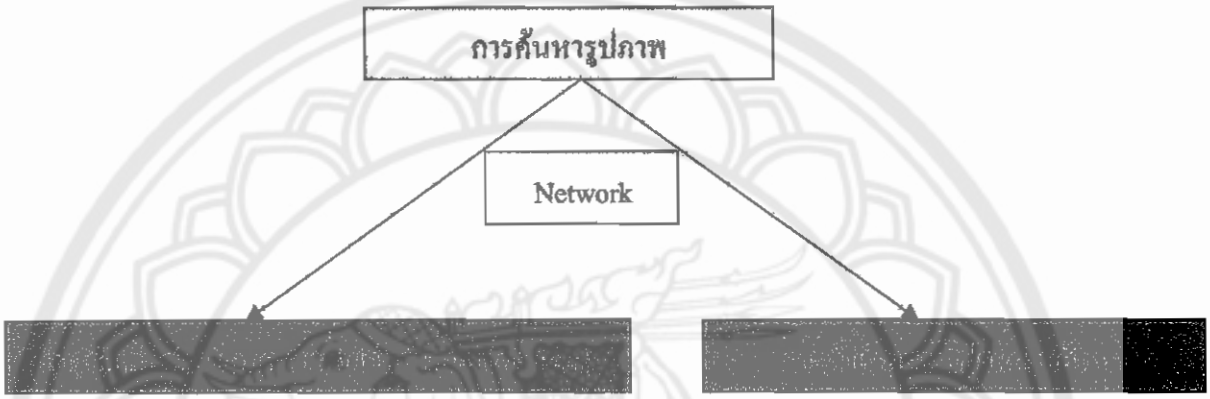
เมื่อเปรียบเทียบกับC++แล้วการสร้างแอพพลิเคชั่นจะทำได้ง่ายกว่ามาก เนื่องจากC#ถูออกแบบมาเพื่อการสร้างแอพพลิเคชั่นให้ทำงานบนอินเตอร์เน็ต(Network)โดยตรง(.NETFramework) นอกจากนี้ C# เป็น Object Oriented Programming (OOP) อย่างสมบูรณ์ ไม่ว่าจะเป็น

- Encapsulation การรวมกลุ่มฟังก์ชั่นการทำงานของออบเจ็กต์ต่างๆ (Object Blueprint, Class) เพื่อทำให้โค้ดถูกเขียนขึ้นมาอย่างเป็นระเบียบ
- Polymorphism (Inheritance, Interfacing และ Overloading) การนำโค้ดที่เขียนขึ้นมาแล้วนั้นมาใช้ในงานอื่นได้อีก

ซึ่งการเขียนโปรแกรม Visual Basic ทำได้ง่ายกว่าแต่ประสิทธิภาพของโปรแกรมมีข้อด้อยกว่าโปรแกรมที่เขียนขึ้นมาจาก C++ ในบางกรณี อย่างเช่น โปรแกรมที่ต้องติดต่อกับฮาร์ดแวร์ จะเลือกใช้ C++ แต่ถ้าต้องการความง่ายในการเขียนโปรแกรม โดยไม่ต้องคำนึงถึงประสิทธิภาพการทำงานมากนักจะเลือกใช้ Visual Basic, C# จะรวมเอาลักษณะการเขียนโปรแกรมจากภาษาทั้งสองเข้ามาไว้ เช่น C# จะไม่มี Overhead มากนัก เมื่อเทียบกับ Visual Basic เป็นต้น

บทที่ 3
วิธีการดำเนินการ

ในการทดลองของการสืบค้นหารูปภาพแบบอัตโนมัติบน Peer to Peer network นั้น จะมีการแบ่งขั้นตอนของการดำเนินงานออกเป็นส่วนต่างๆ ได้ดังนี้

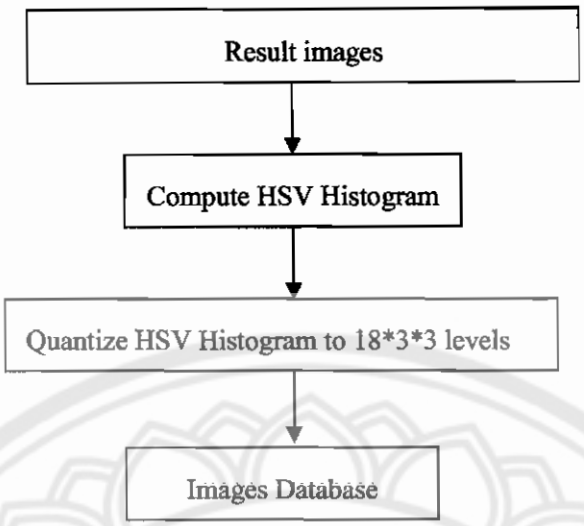


รูปที่ 3.1 ส่วนประกอบของขั้นตอนการค้นหารูปภาพ

ซึ่งในการทำงานจริงจะมีส่วนของขั้นตอนการทำงานต่างๆ ย่อยลงไปอีก โดยมีขั้นตอนการทำงานดังต่อไปนี้

3.1 การจัดการฐานข้อมูล

โปรแกรมจะทำการวิเคราะห์องค์ประกอบของภาพโดยส่วนนี้จะเลือกองค์ประกอบของสี HSV แล้วทำการเก็บข้อมูลองค์ประกอบของสีดังกล่าว รวมถึงข้อมูลที่สำคัญเพื่อที่จะช่วยดึงรูปภาพจากฐานข้อมูลขึ้นมาแสดงในโปรแกรมได้ เช่น ตำแหน่งที่อยู่ของรูปภาพ เป็นต้น ซึ่งการวิเคราะห์ค่าองค์ประกอบของสีนั้นจะมีหลักการดังนี้



รูปที่ 3.2 ขั้นตอนการวิเคราะห์องค์ประกอบสี H, S, V ของรูปภาพ

3.2 การหาค่า RGB Color histogram

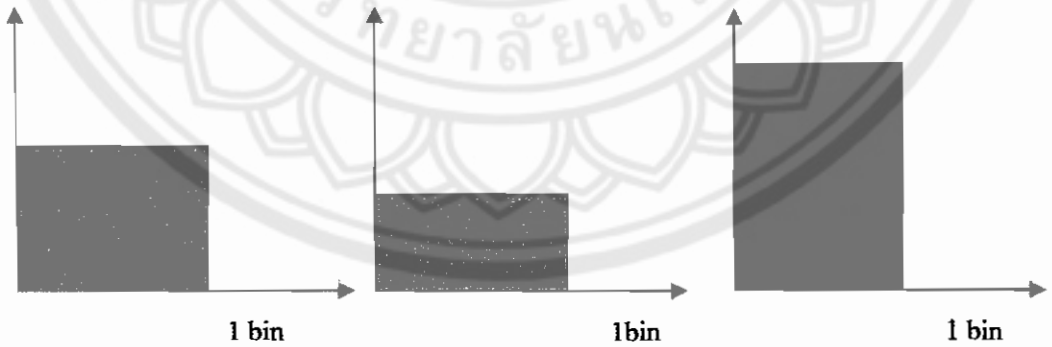
3.2.1 Normalization RGB

โดยจะทำการวิเคราะห์ค่าสีแต่ละ Pixel จะทำให้ได้ค่าสีที่ออกมาเป็น Histogram ซึ่งแต่ละรูปนั้นจะมีค่า Histogram 3 ค่า คือ

Red ซึ่งจะเป็นค่าสีแดงที่เป็นองค์ประกอบของภาพนั้น

Green ซึ่งจะเป็นค่าสีเขียวที่เป็นองค์ประกอบของภาพนั้น

Blue ซึ่งจะเป็นค่าสีน้ำเงินที่เป็นองค์ประกอบของภาพนั้น



รูปที่ 3.3 วิเคราะห์หาค่าสีของรูปภาพ

เมื่อทำการวิเคราะห์แล้วจะได้กราฟแสดงความถี่ของค่าสีแต่ละสีดังรูปที่ 3.3 แล้วทำการหาค่าเฉลี่ยของแต่ละสีจากสมการ

$$R = \frac{\sum_{i=0}^{j} p_{ij}}{p}, \qquad G = \frac{\sum_{i=0}^{j} p_{ij}}{p}, \qquad B = \frac{\sum_{i=0}^{j} p_{ij}}{p}$$

โดยที่

i, j คือขอบเขตการวิเคราะห์

p_{ij} คือจุดPixelที่ทำการวิเคราะห์

p คือจำนวนPixelทั้งหมดที่ทำการวิเคราะห์

จากนั้นทำการ Normalization ค่าของ RGB Color Histogram จากสมการ

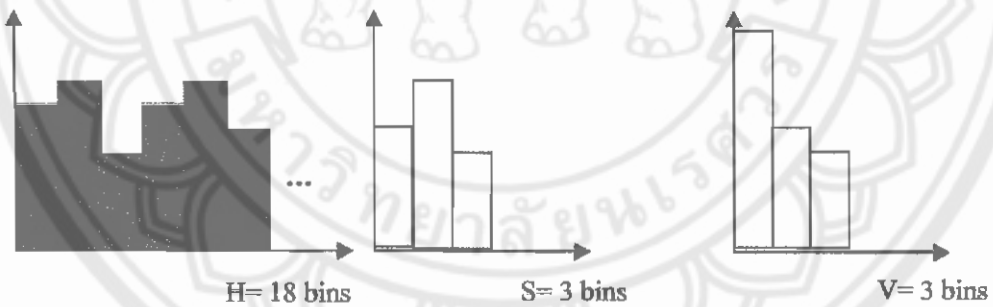
Normalize red r = R/(R+G+B)

Normalize green g = G/(R+G+B)

Normalize blue b = B/(R+G+B)

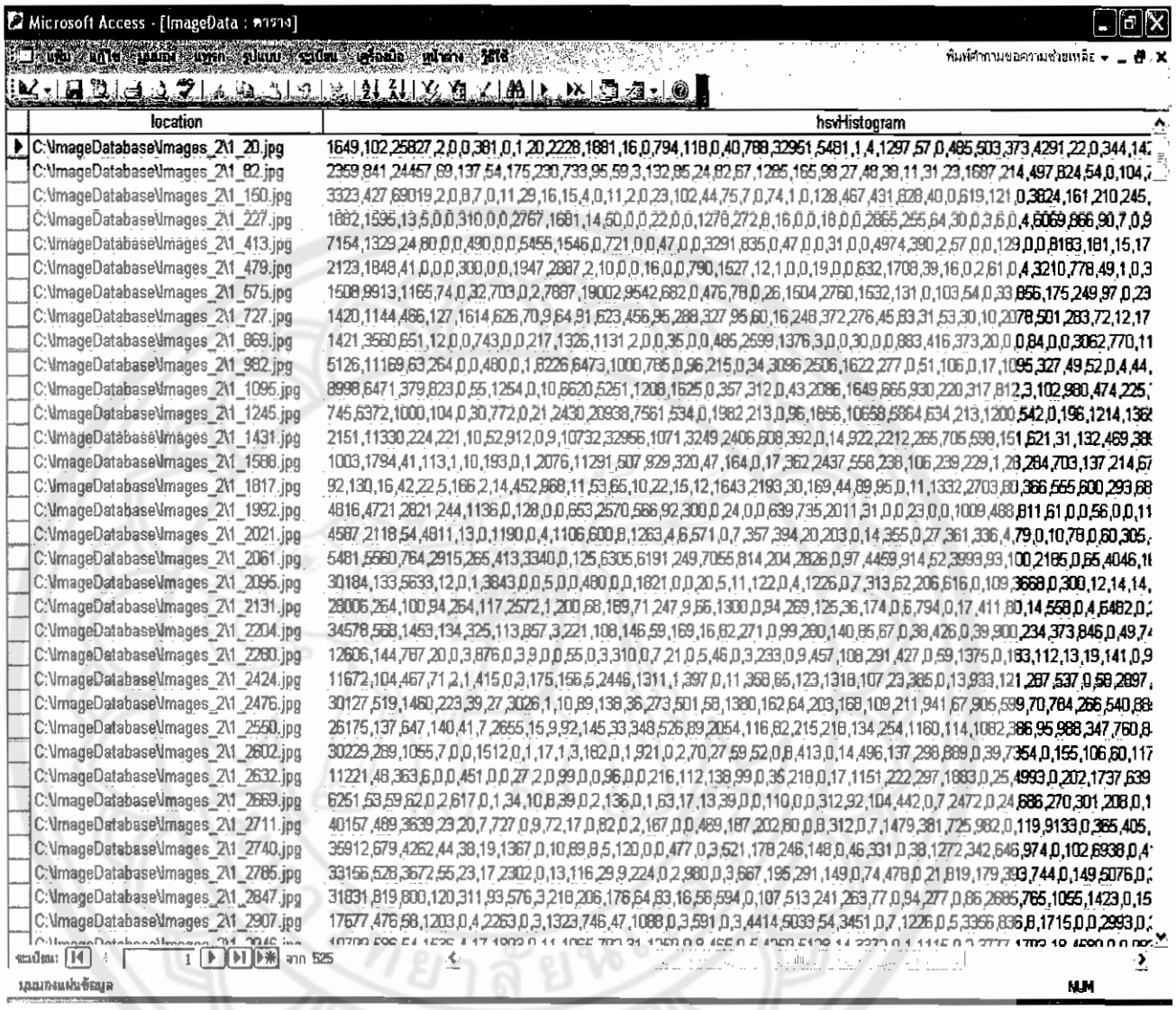
3.2.2 Quantize HSV to 18*3*3 Levels

แบ่งระดับสีของแต่และสีเป็น 18, 3, 3 ตามลำดับ



รูปที่ 3.4 แสดงค่าองค์ประกอบของสี H, S, V เมื่อทำการ Quantize เป็นเวกเตอร์

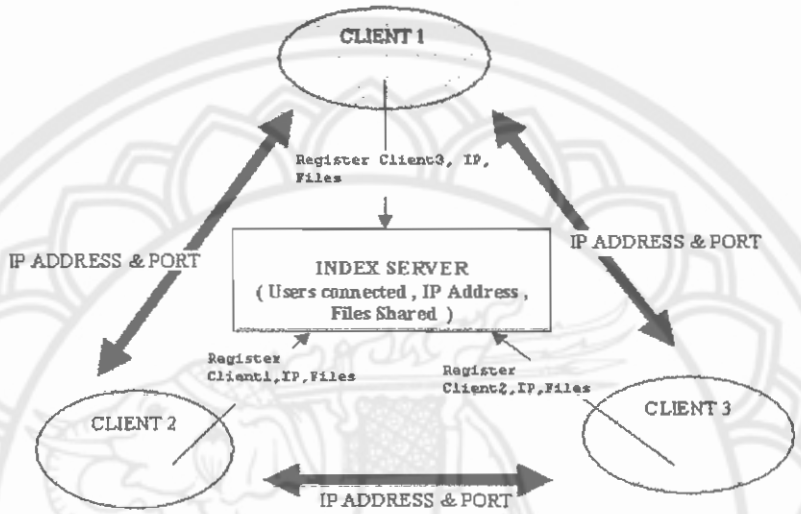
ซึ่งการทำ Normalize HSV จะทำในลักษณะเดียวกันกับการNormalize RGBและค่าของ H, S, V ก็จะเก็บในฐานข้อมูลดังนี้



รูปที่ 3.5 แสดงตัวอย่างฐานข้อมูลเก็บค่า HSV Histogram ของรูปภาพ

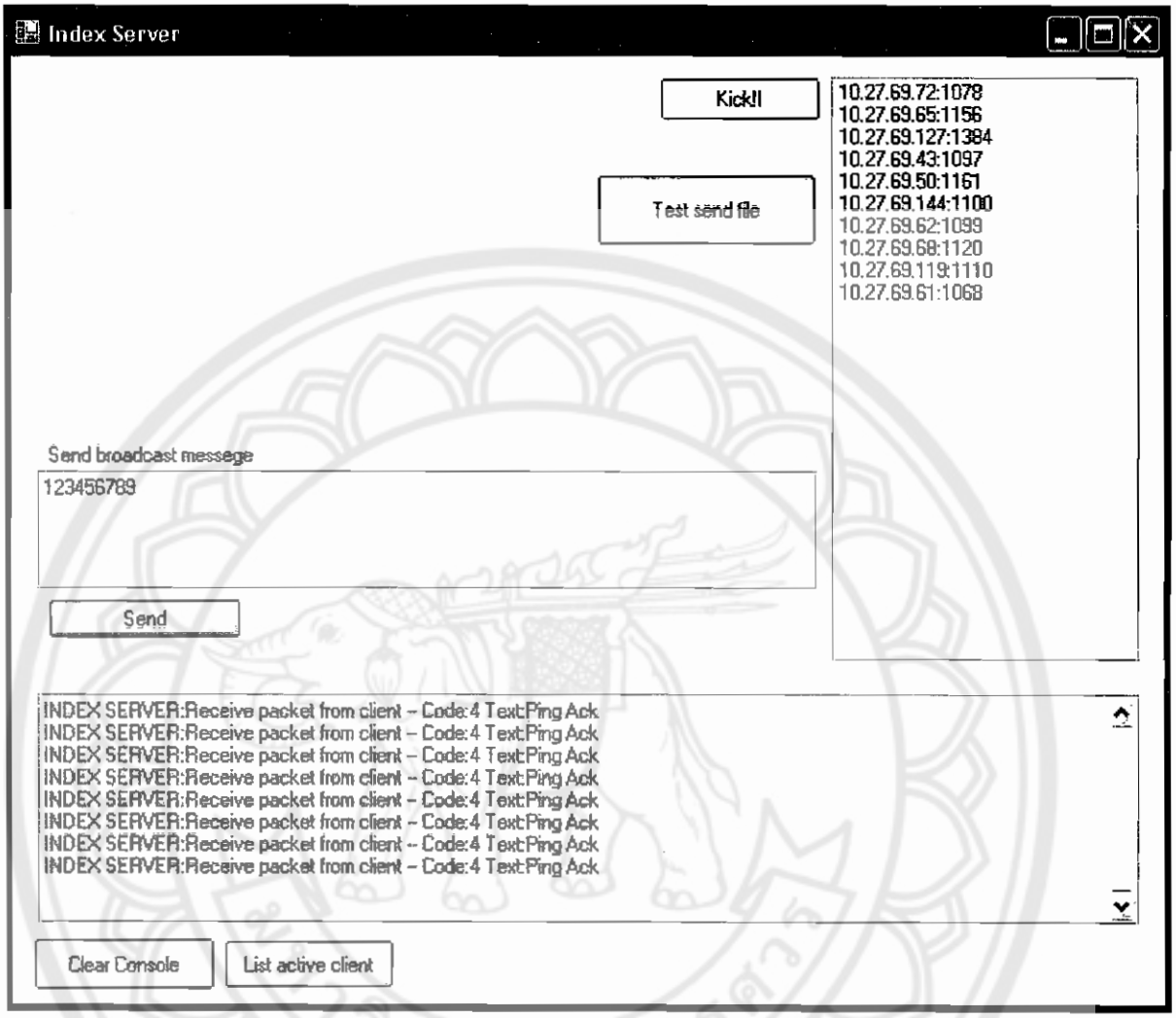
3.3 Index Server

Index Server ทำไว้เพื่อให้ Peer ต่างๆ Login เข้ามาในระบบ โดยตัวของ Index Server เองจะไม่มีหน้าที่ในการประมวลผลข้อมูล แต่ Index Server จะมีหน้าที่บอกว่าขณะนี้มี Peer IP Address อะไรบ้างที่ Connect อยู่กับตัว Index Server แล้ว Index Server จะเป็นตัวสั่งให้ Peer ที่ Connect อยู่กับตัวมันเชื่อมต่อกับ Peer อื่นๆโดยตรงดังรูป



รูปที่ 3.6 โครงสร้าง Peer to Peer

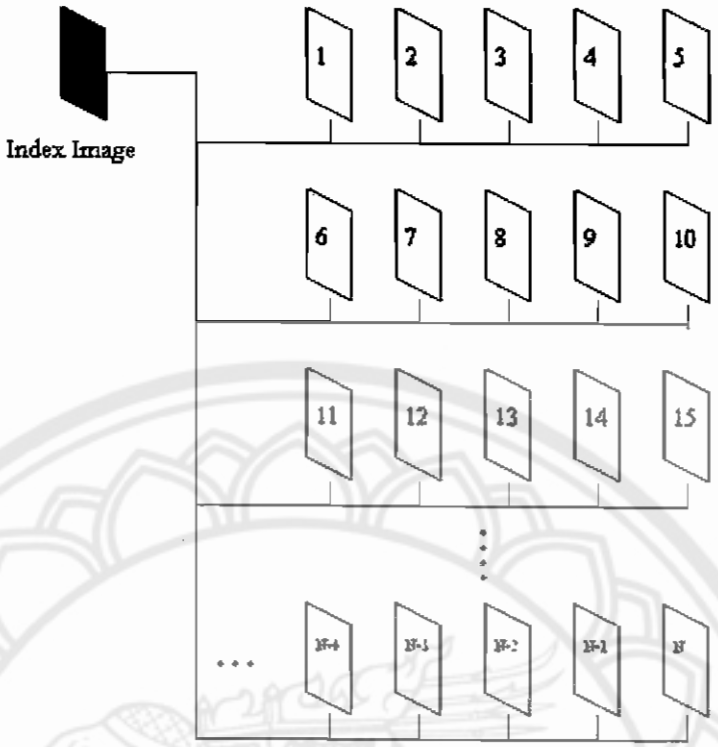
ในการทดลองนี้จะใช้ Peer ทั้งหมด 10 Peer มาเชื่อมต่อเพื่อค้นหารูปภาพโดยจะมี Index Server เป็นตัวบอกว่ามี Peer อะไรบ้างที่เชื่อมต่ออยู่ ดังรูป



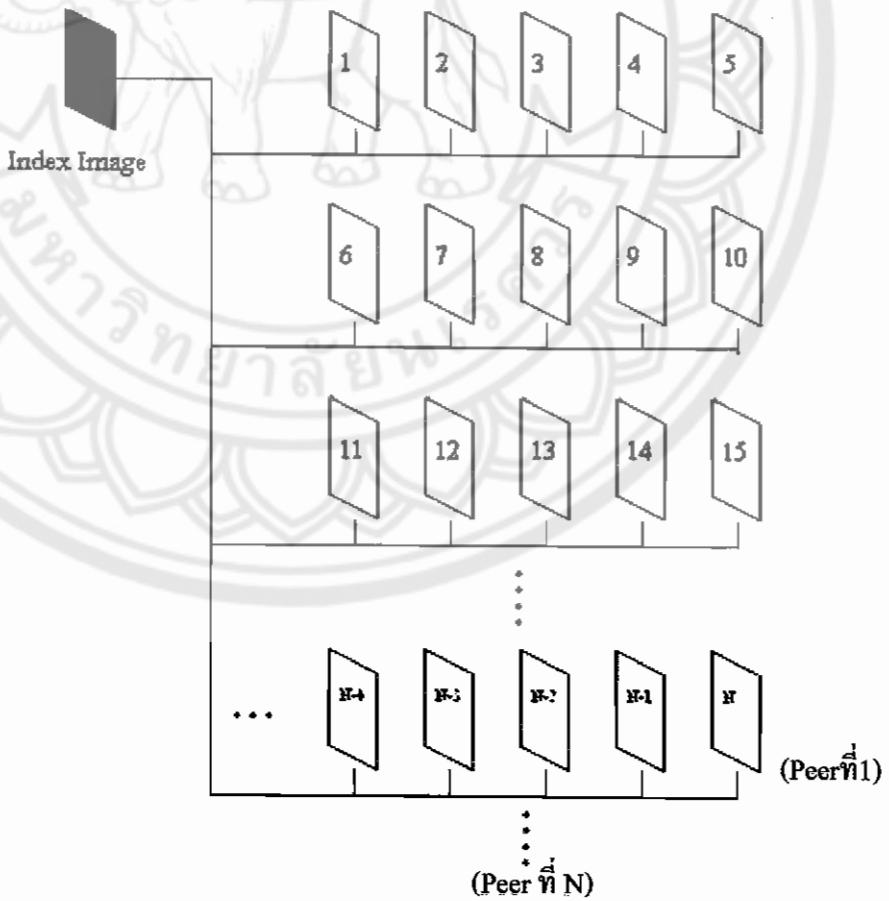
รูปที่ 3.7 ตัวอย่างของ Index Server ที่มี Peer เชื่อมต่ออยู่ทั้งหมด 10 Peer

3.4 วิธีการค้นหารูปภาพ

ในส่วนของตัวโปรแกรมเองจะมีวิธีการค้นหารูปภาพอยู่สองลักษณะคือ ค้นหาแบบ Local และค้นหาแบบ P2P และโปรแกรมนี้จะใช้วิธีการเปรียบเทียบการค้นหารูปภาพที่ต้องการ โดยเราจะนำค่า Color Histogram ของภาพที่ต้องการนำไปเปรียบเทียบกับภาพอื่นๆในฐานข้อมูล โดยใช้หลักการเปรียบเทียบ ดังนี้



รูปที่ 3.8 ขั้นตอนการค้นหารูปภาพแบบ Local



รูปที่ 3.9 ขั้นตอนการค้นหารูปภาพแบบ P2P

โดยการเปรียบเทียบแต่ละครั้ง จะได้ค่า Distancc มา 1 ค่า

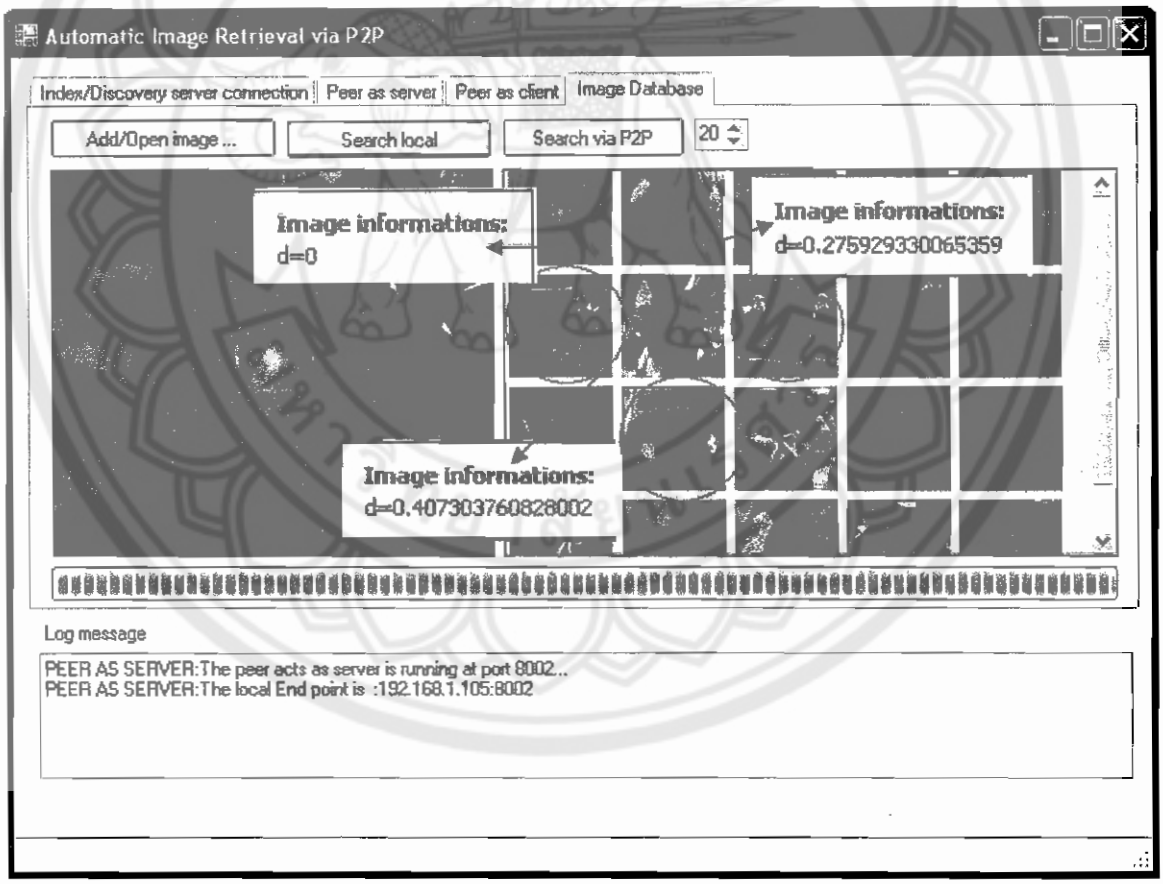
$$\text{Distance1}=\sum_{i=1}^{24}(\bar{b}1_i-\bar{v}i) = \left|(\bar{b}1_1-\bar{v}_1)+(\bar{b}1_2-\bar{v}_2)+....+(\bar{b}1_{24}-\bar{v}_{24})\right|$$

$$\text{Distance2}=\sum_{i=1}^{24}(\bar{b}2_i-\bar{v}i) = \left|(\bar{b}2_1-\bar{v}_1)+(\bar{b}2_2-\bar{v}_2)+....+(\bar{b}2_{24}-\bar{v}_{24})\right|$$

$$\vdots$$

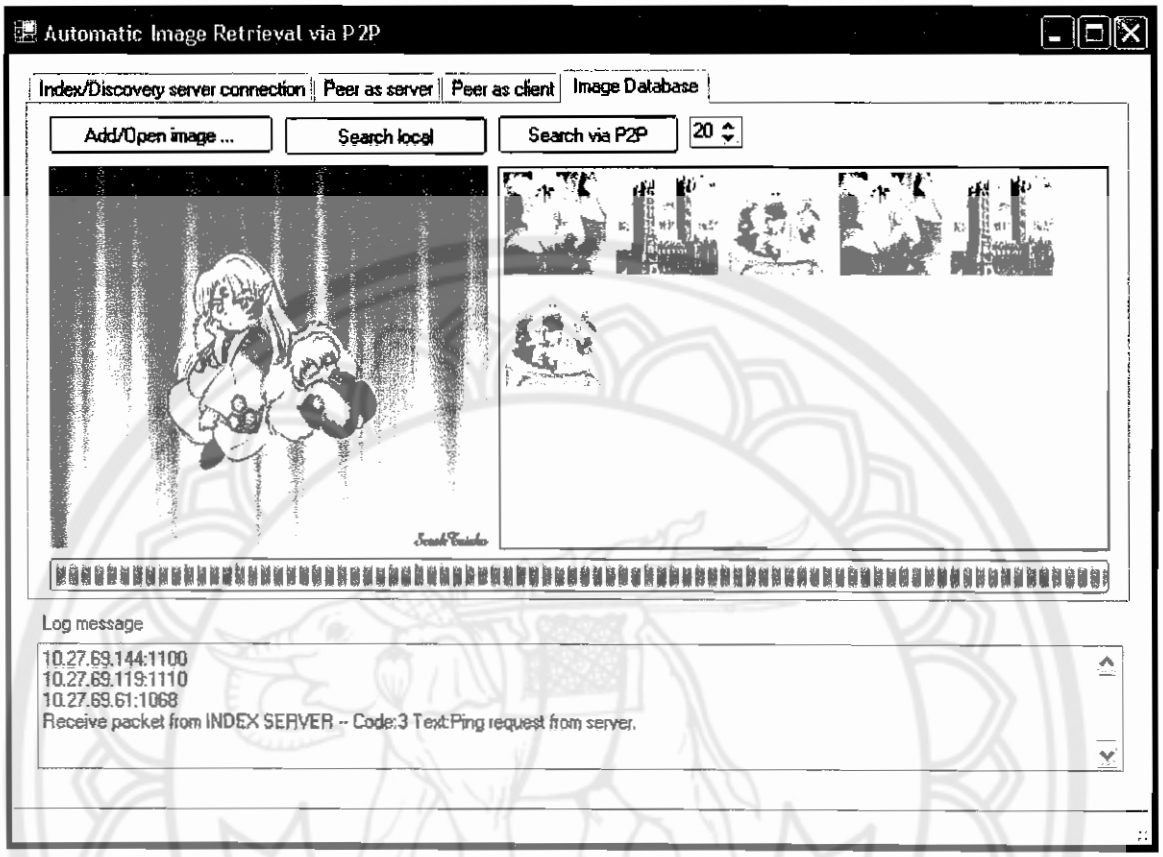
$$\text{Distance N}=\sum_{i=1}(\bar{b}N_i-\bar{v}i) = \left|(\bar{b}N_1-\bar{v}_1)+(\bar{b}N_2-\bar{v}_2)+....+(\bar{b}N_{24}-\bar{v}_{24})\right|$$

ในโปรแกรมนี้ค่า Distance ที่นำมาเปรียบเทียบในฐานข้อมูลแล้วได้ค่าเท่ากับ 0 หรือใกล้เคียง 0 ถือว่าภาพนั้นเป็นภาพเดียวกันซึ่งจะมองในลักษณะขององค์ประกอบโดยรวมของสีถึงแม้ภาพจะไม่ใช่ภาพชนิดเดียวกันก็ตาม เช่นดังรูป



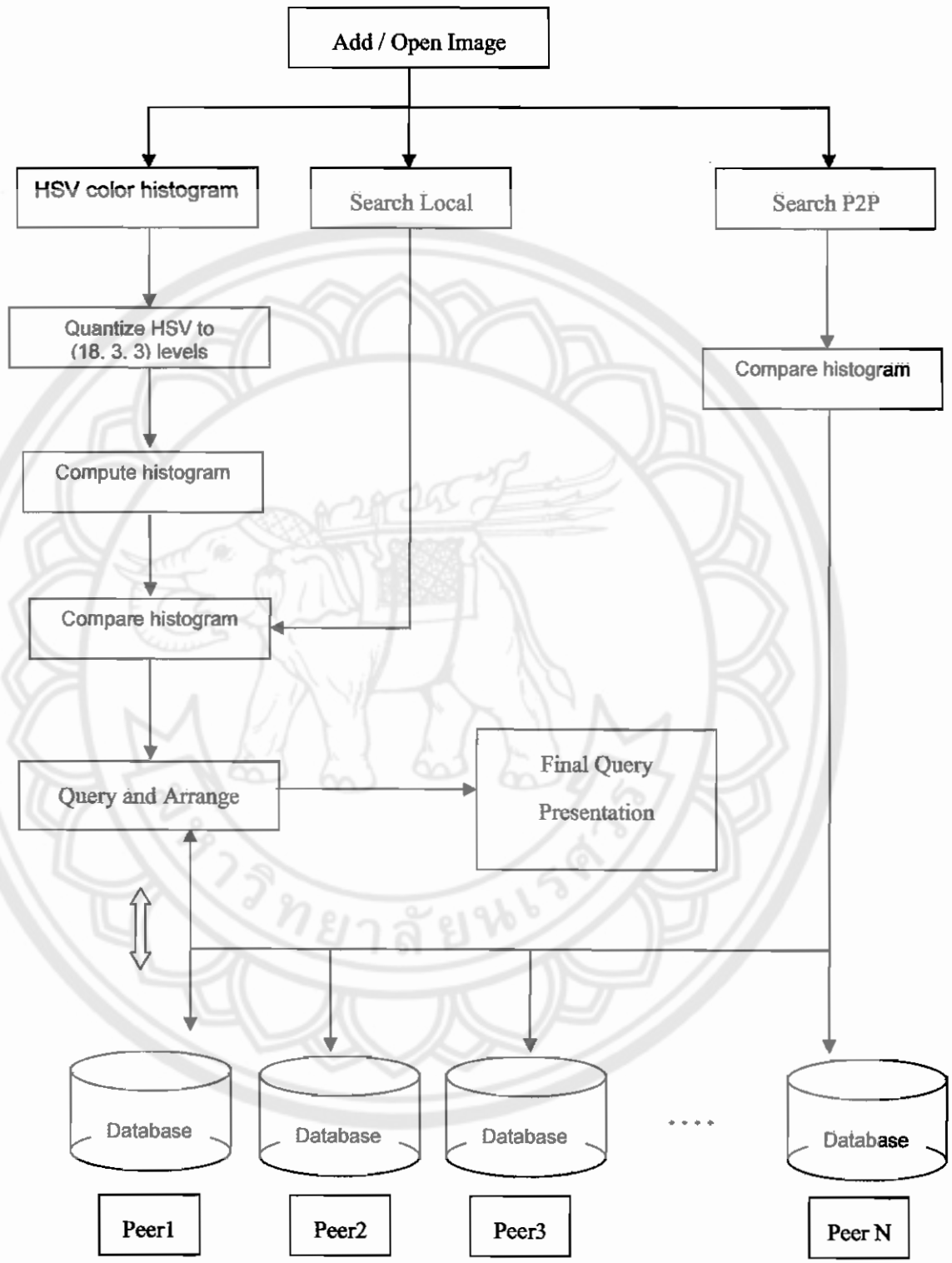
รูปที่ 3.10 แสดงตัวอย่างการค้นหารูปภาพแบบ Local

ในการค้นหารูปภาพบน Peer โปรแกรมนี้จะมีข้อจำกัดอยู่ที่ว่าจะเอา Index Image ไปเปรียบเทียบกับ Peer ที่ Connect เพียงสามรูปเพื่อประหยัดในเรื่องของเวลาในการค้นหา



รูปที่ 3.11 แสดงตัวอย่างการค้นหารูปภาพแบบ P2P

3.5 โครงสร้างการทำงาน



รูปที่ 3.12 แสดงขั้นตอนการทำงาน

บทที่ 4

ผลการทดลอง

ในบทนี้จะเป็นผลของการทำการทดลองค้นหารูปภาพแบบอัตโนมัติ จากฐานข้อมูลที่ได้สร้างไว้ในตอนแรก โดยใช้หลักการเปรียบเทียบผลของค่า Precision เป็นค่าที่ได้จากการคำนวณจากสมการดังต่อไปนี้คือ

$$\%\text{ความถูกต้อง} = (\text{จำนวนรูปภาพที่ถูกเลือก} / \text{จำนวนรูปภาพทั้งหมด}) \times 100 \qquad (4.1)$$

$$\underline{\text{ตัวอย่างเช่น}}\ \%\text{ความถูกต้อง} = \left(\frac{18}{20}\right) \times 100 = 90\%$$

ซึ่งถ้านำค่าความถูกต้อง (Precision) ของ Query ทั้งหมด มารวมกันแล้วหาค่าเฉลี่ย ก็จะได้เป็น Precision ของMethod นั้นๆ ซึ่งเมื่อนำไปเปรียบเทียบกับ Method อื่นๆ ก็จะเห็นความแตกต่างของการค้นหาได้จากการสร้างกราฟ

4.1 สิ่งที่ต้องเตรียมก่อนทำการทดลอง

1) ไฟล์รูปภาพที่ใช้เป็นฐานข้อมูล ซึ่งจะเป็นไฟล์ชนิด .JPEGและ GIF ทั้งหมด และต้องมีขนาดเท่ากัน

2) โปรแกรม Search รูปภาพที่พัฒนามาจากภาษา C#.Net

3) คอมพิวเตอร์อย่างน้อย 2 เครื่อง โดยมีการเชื่อมต่อด้วยระบบ LAN

4) โปรแกรมหาค่า HSV Histogram เพื่อเก็บไว้ในฐานข้อมูล

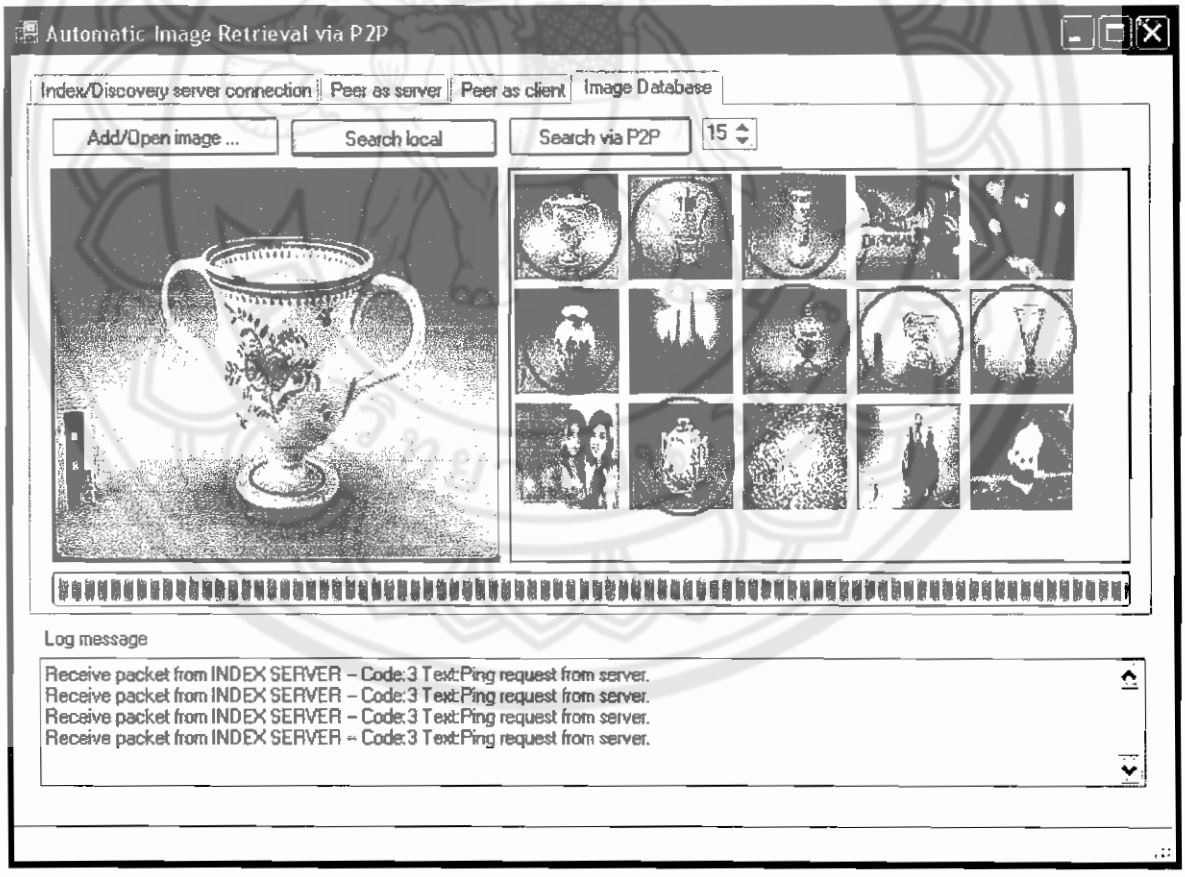
4.2 ผลการทดลอง

การทดลองค้นหารูปภาพจากไฟล์รูปภาพมีอยู่ 2 แบบด้วยกัน คือ

1. **Search local** ซึ่งจะเป็นการค้นหารูปภาพจากฐานข้อมูลที่อยู่ใน Client นั้นๆ
2. **Search via P2P** เป็นการค้นหารูปภาพจากฐานข้อมูลที่อยู่ใน Network

4.2.1 การค้นหารูปภาพแบบ Search local

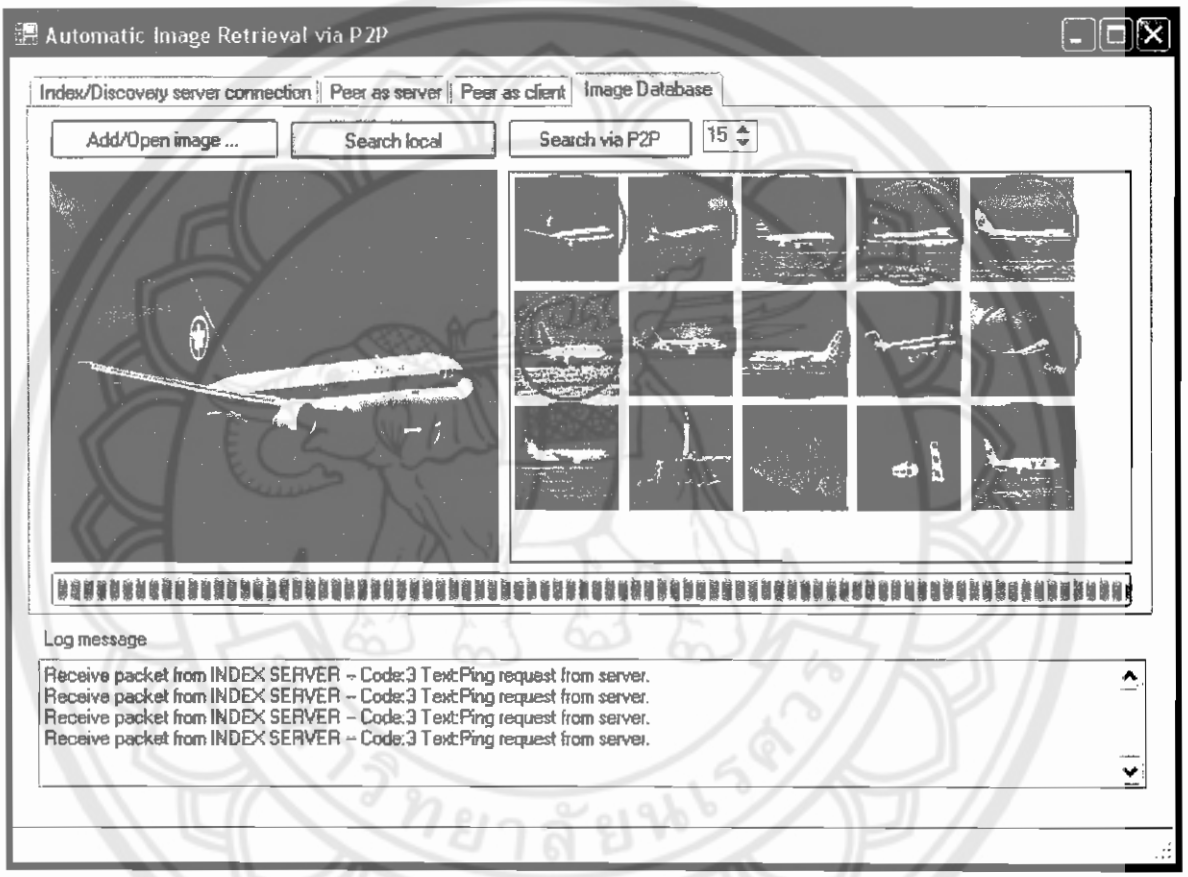
1) นำฐานข้อมูลที่ได้ทำไว้แล้ว ไปวางไว้ในไดรฟ์ C: ของ Client แต่ละเครื่อง
2) ทำการเปิดตัวโปรแกรม ซึ่งในการเปิดโปรแกรมจะต้องทำการเชื่อมต่อโดยใช้โปรโตคลอชนิด TCP พอร์ต 8001หรือพอร์ตอื่นๆ ก็ได้ที่ไม่ไปซ้ำกับโปรแกรมอื่นที่รันอยู่ และทำการใส่ IP Address ของเครื่องที่เป็น Index Server
3) ทำการเลือกรูปภาพตัวอย่างที่เราต้องการหา แล้วสั่งให้โปรแกรมทำการค้นหา ซึ่งสามารถกำหนดจำนวนของรูปภาพที่ต้องการค้นหาได้
4) โปรแกรมจะทำการค้นหารูปภาพ โดยใช้วิธี วิเคราะห์สีและขนาดของรูปภาพ ในที่นี้กำหนดจำนวนภาพที่ต้องการเท่ากับ 15 ไฟล์
5) ทำการเลือกรูปภาพที่ได้จากการทดสอบ ในกรณีนี้จะพิจารณาองค์ประกอบของรูปภาพที่ได้จากการรันโปรแกรมว่ามีส่วนเกี่ยวข้องกับรูปภาพที่เราต้องการหรือไม่



รูปที่ 4.1 แสดงการ Query จากไฟล์รูปตัวอย่าง (Search local)

จากรูปการทดสอบที่ 4.1 จะเห็นได้ว่ามีรูปที่มีลักษณะ ที่อยู่ในกลุ่มเดียวกันกับไฟล์รูปตัวอย่างทั้งหมด 8 ไฟล์ เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

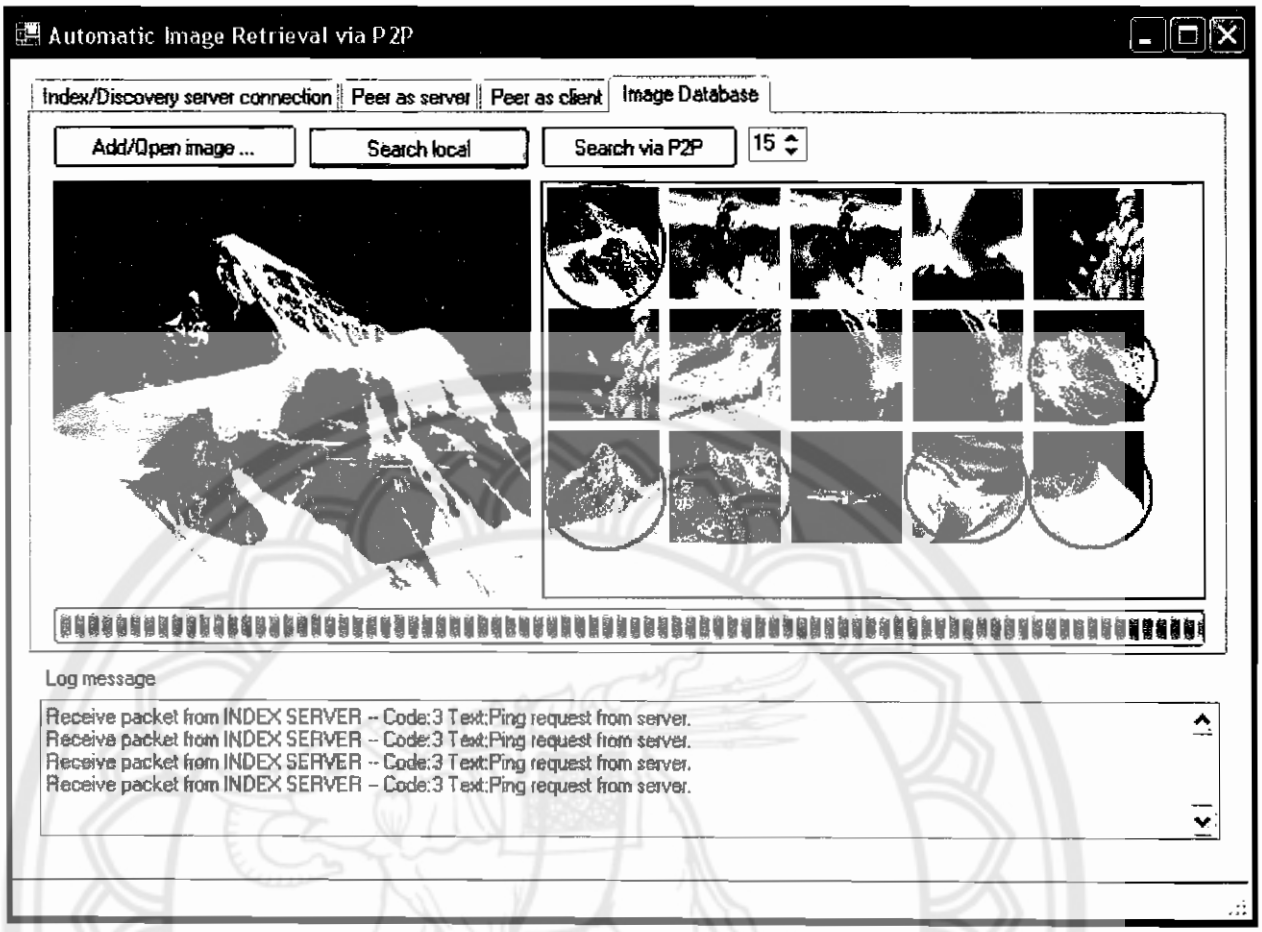
$$\%\text{Precision} = \left(\frac{8}{15}\right)\times 100 = 53.33\ \%$$



รูปที่ 4.2 แสดงการ Query จากไฟล์รูปตัวอย่าง (Search local)

จากรูปการทดสอบที่ 4.2 จะเห็นได้ว่ามีรูปที่มีลักษณะ ที่อยู่ในกลุ่มเดียวกันกับไฟล์รูปตัวอย่างทั้งหมด 12 ไฟล์ เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

$$\%\text{Precision} = \left(\frac{12}{15}\right)\times 100 = 80\ \%$$



รูปที่ 4.3 แสดงการ Query จากไฟล์รูปตัวอย่าง (Search local)

จากรูปการทดสอบที่ 4.3 จะเห็นได้ว่ามีรูปที่มีลักษณะ ที่อยู่ในกลุ่มเดียวกันกับไฟล์รูปตัวอย่าง ทั้งหมด 6 ไฟล์ เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

$$\%\text{Precision} = \left(\frac{6}{15}\right) \times 100 = 40\,\%$$

ตารางที่ 4.1 บันทึกการทดลองการค้นหารูปภาพจากไฟล์รูปภาพบนClient นั้นๆ (Search local)

ลำดับ	ชื่อไฟล์รูปภาพ	จำนวนรูปภาพที่เลือก	ตำแหน่งรูปภาพที่เลือก															Precision (%)
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	35s	9	*	*	*	*		*		*	*	*		*				60.00
2	38s	12	*	*	*	*	*	*	*	*	*	*	*				*	80.00
3	41s	6	*									*	*	*		*	*	40.00
4	42s	4	*											*	*	*		26.66
5	36s	9	*							*	*	*	*	*	*	*	*	60.00
6	37s	6	*							*			*	*		*	*	40.00
7	39s	9	*			*	*	*	*	*		*	*			*		60.00
8	40s	2	*										*					13.00
9	43s	10	*	*	*	*	*	*			*			*	*	*		66.66
10	46s	3	*												*	*		20.00
11	1_23	9	*	*		*		*	*	*		*		*	*			60.00
12	2_15	11	*	*	*	*		*	*		*		*	*	*	*		73.33
13	2_18	9	*			*	*	*	*			*	*			*	*	60.00
14	2_22	9	*	*	*			*		*		*	*	*			*	60.00
15	2_29	7	*			*				*	*	*			*	*		46.66
16	3_1	11	*	*	*	*	*			*	*		*	*		*	*	73.33
17	3_10	5	*	*		*					*				*			33.33
18	3_15	11	*		*	*	*	*	*	*	*	*				*	*	73.33
19	4_50	11	*	*	*	*		*		*	*	*	*			*	*	73.33
20	4_55	12	*	*	*	*		*	*	*	*			*	*	*	*	80.00
SUM		165	20	10	9	13	6	11	7	11	11	11	11	11	9	14	10	1099.63
AVG		8.25																54.98

4.2.2 การค้นหารูปภาพแบบ Search via P2P

1) ในการทดลองนี้จะทำการทดลองในระบบNetwork

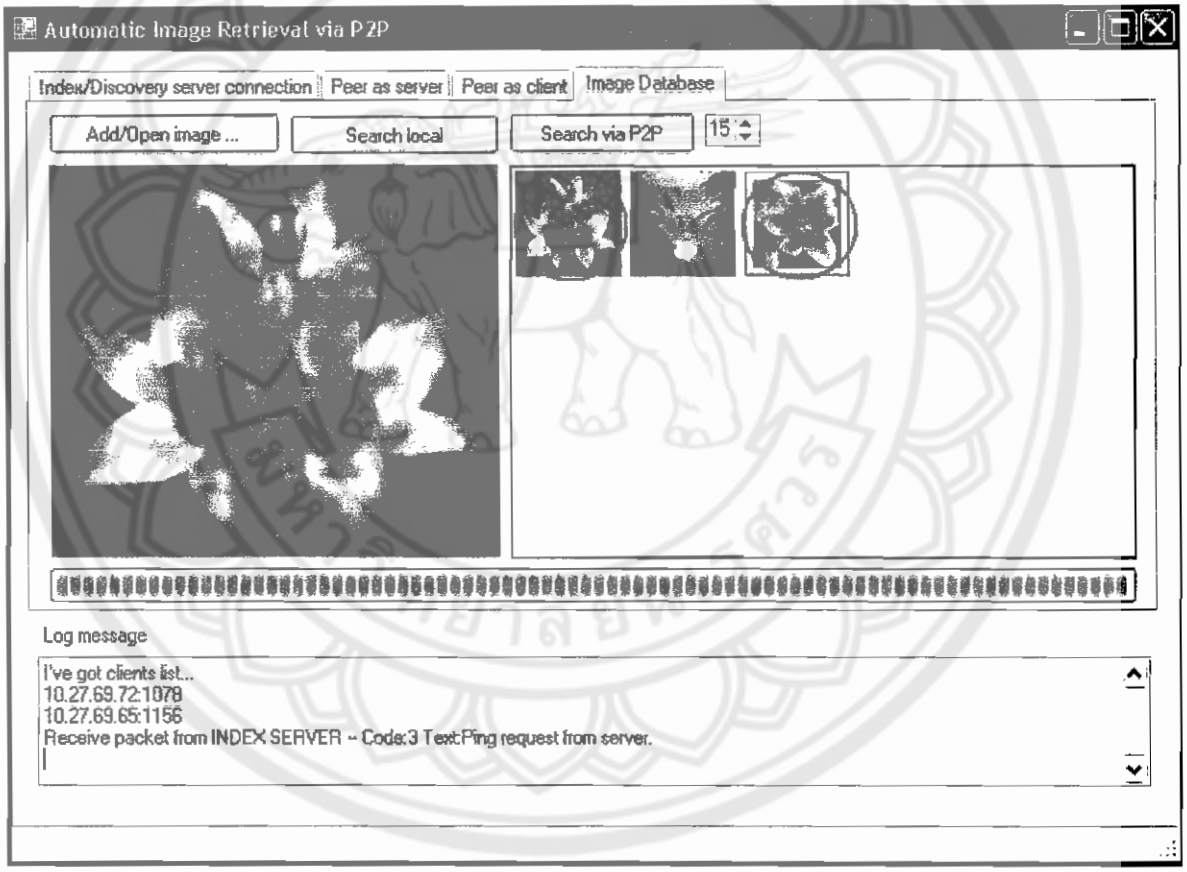
2) ใช้คอมพิวเตอร์ ทั้งหมด 10 เครื่อง (ใช้เป็น Index server 1 เครื่อง)

3) แต่ละ Client ต้องติดต่อกับ Index server โดยใช้โปรโตคลอชนิด TCP พอร์ต 8001

4) ทำการกำหนด IP Address ของเครื่อง Index server ให้แก่ Client แต่ละตัว

5) ทำการทดลองโดยทำการค้นหาภาพจาก Client ตั้งแต่ 2 เครื่องไปจนถึง 10 เครื่อง ซึ่งจะพิจารณาจากค่าความถูกต้อง และเวลาที่ใช้ในการค้นหา

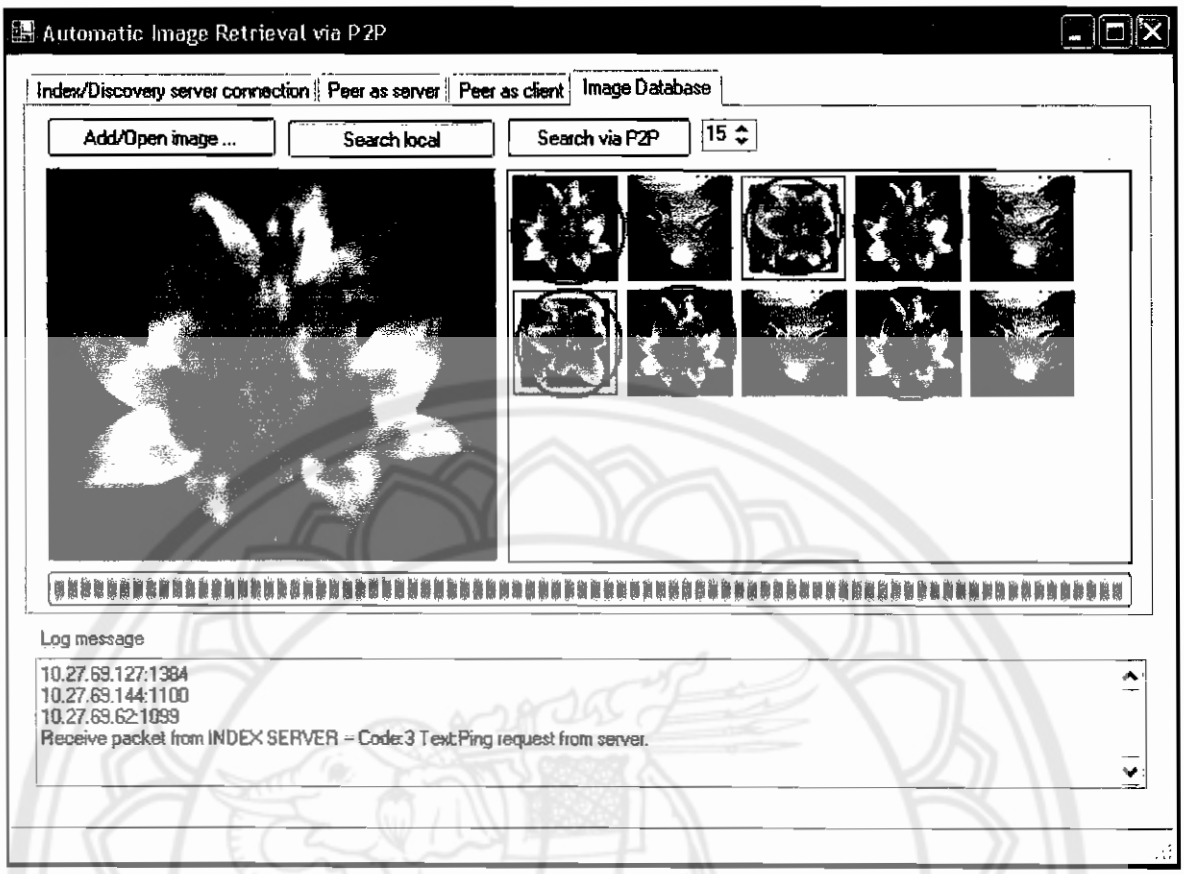
6) การค้นหารูปภาพแบบ Search via P2P จะไม่สามารถกำหนดจำนวนภาพที่ต้องการหาได้ และโปรแกรมนี้จะจำกัดจำนวนภาพที่ต้องการค้นหาในแต่ละ Peer 3 ภาพเพื่อช่วยในเรื่องของเวลาในการค้นหา



รูปที่ 4.4 การค้นหารูปภาพบนClient 2 เครื่อง (Search via P2P)

จากรูปการทดสอบที่ 4.4 จะเห็นได้ว่ามีรูปที่มีลักษณะ ที่อยู่ในกลุ่มเดียวกันกับไฟล์รูปตัวอย่างทั้งหมด 2 ไฟล์ ซึ่งเวลาที่ใช้ในการค้นหาภาพเท่ากับ 6.82 วินาที เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

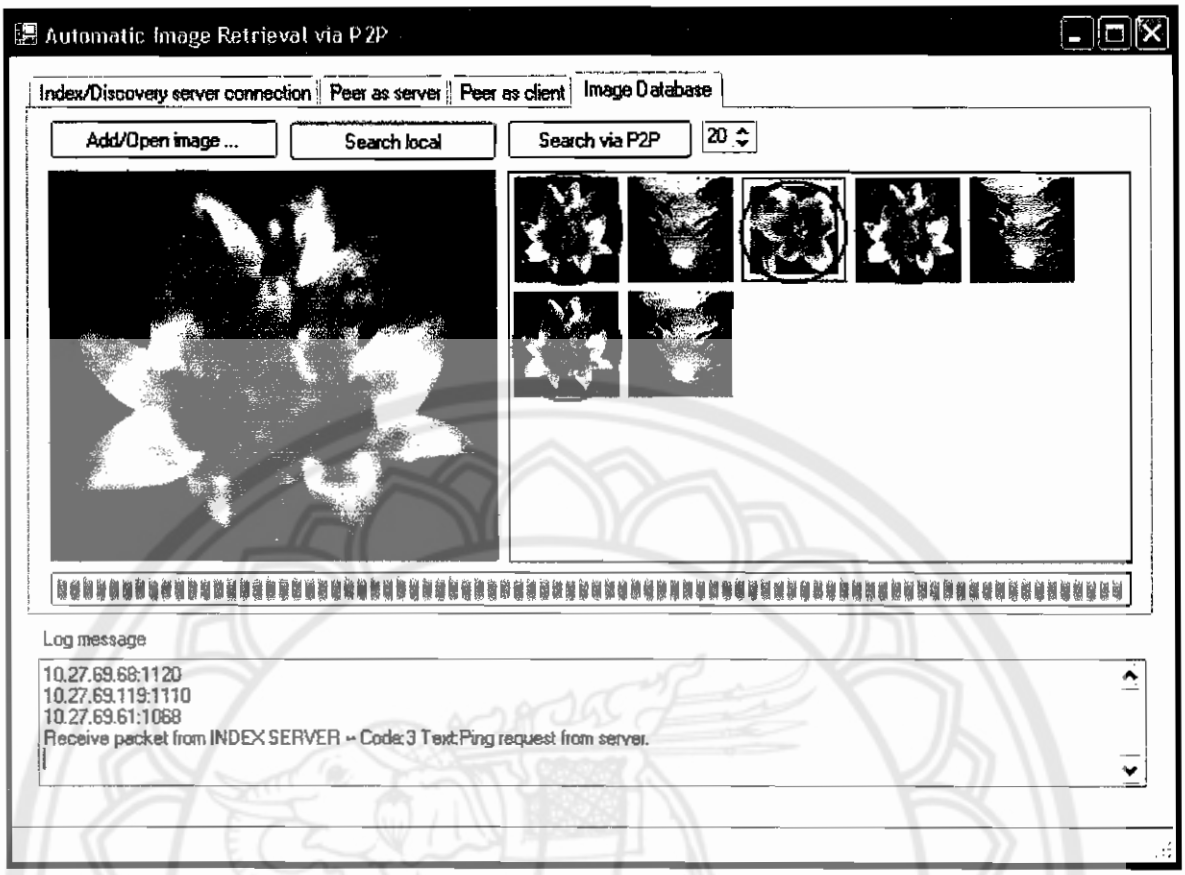
$$\%\text{Precision} = \left(\frac{2}{3}\right) \times 100 = 66.66\ \%$$



รูปที่ 4.5 การค้นหารูปภาพบนClient 7 เครื่อง (Search via P2P)

จากรูปการทดสอบที่ 4.5 จะเห็นได้ว่ามีรูปที่มีลักษณะ ที่อยู่ในกลุ่มเดียวกันกับไฟล์รูปตัวอย่าง ทั้งหมด 6 ไฟล์ ซึ่งเวลาที่ใช้ในการค้นหาภาพเท่ากับ 21.99 วินาที เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

$$\%\text{Precision} = \left(\frac{6}{10}\right) \times 100 = 60\,\%$$



รูปที่ 4.6 การค้นหารูปภาพบนClient 10 เครื่อง (Search via P2P)

จากรูปการทดสอบที่ 4.6 จะเห็นได้ว่ามีรูปที่มีลักษณะ ที่อยู่ในกลุ่มเดียวกันกับไฟล์รูปตัวอย่าง ทั้งหมด 4 ไฟล์ ซึ่งเวลาที่ใช้ในการค้นหาภาพเท่ากับ 23.31 วินาที เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

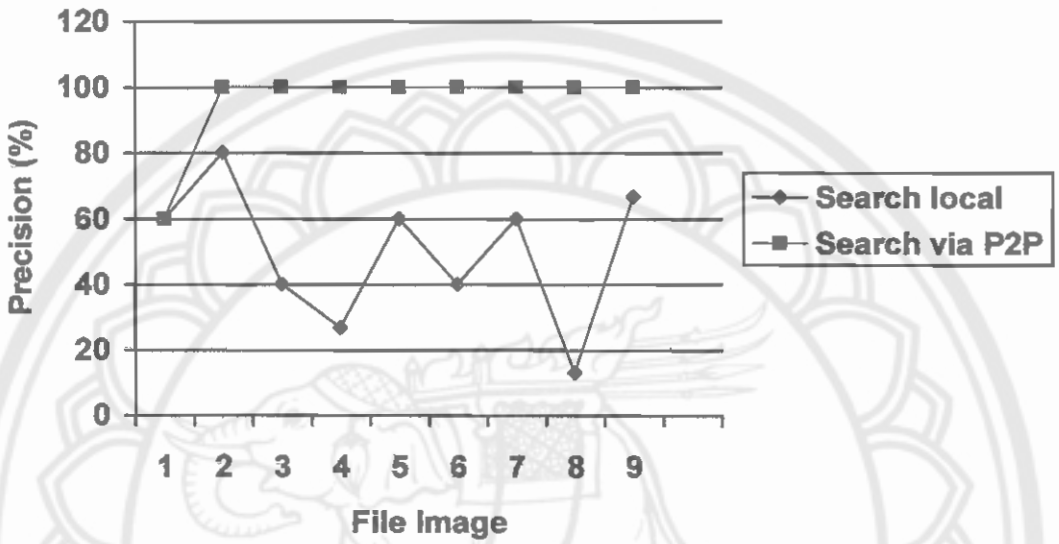
$$\%\text{Precision} = \left(\frac{4}{7}\right) \times 100 = 57.14\,\%$$

ตารางที่ 4.2 บันทึกการทดลองการค้นหารูปภาพจากไฟล์รูปภาพบนNetwork (Search via P2P)

ลำดับ	จำนวน Client	ชื่อไฟล์รูปภาพ	จำนวนรูปภาพที่เลือก	ตำแหน่งรูปภาพที่เลือก					เวลาที่ใช้ในการค้นหา (วินาที)	Precision (%)
				1	2	3	4	5		
1	2	2_3	3	*	*	*			6.82	60.00
2	3	2_16	5	*	*	*	*	*	7.12	100.00
3	4	3_41	5	*	*	*	*	*	9.55	100.00
4	5	5_10	5	*	*	*	*	*	12.00	100.00
5	6	10_26	5	*	*	*	*	*	14.64	100.00
6	7	22_3	5	*	*	*	*	*	22.99	100.00
7	8	26_15	5	*	*	*	*	*	23.16	100.00
8	9	26_19	5	*	*	*	*	*	23.22	100.00
9	10	29_30	5	*	*	*	*	*	23.31	100.00
SUM			43	9	9	9	8	8	142.63	860.00
AVG			4.77						15.84	95.55

4.3 เปรียบเทียบผลการทดลอง

นำผลการทอลองที่หาค่าความถูกต้อง (Precision) ของการค้นหาไฟล์รูปภาพแต่ละไฟล์มา plot กราฟ เปรียบเทียบการค้นหาทั้ง 2 วิธี จะเห็นความแตกต่างของการทดลองโดยที่เส้นกราฟที่ได้จากการค้นหารูปภาพในแต่ละแบบ ดังรูปที่ 4.3



รูปที่ 4.7 กราฟแสดงการเปรียบเทียบการทำงานของแต่ละวิธี

เมื่อได้ผลการเปรียบเทียบการทดลองในแต่ละขั้นตอนออกมาแล้ว จะสังเกตเห็นถึงความแตกต่างของแต่ละวิธีได้อย่างชัดเจน

บทที่ 5
สรุปผลและข้อเสนอแนะ

ในบทนี้จะเป็นการสรุผลของโครงงานนี้ ซึ่งจะกล่าวถึงการสรุปผลของโครงงาน ปัญหาในขั้นตอนการทำงาน ข้อเสนอแนะและแนวทางในการพัฒนาเพื่อเป็นประโยชน์สำหรับผู้สนใจจะพัฒนาโครงงานนี้ต่อไป

5.1 สรุปผล

1. ตัวโปรแกรมสามารถค้นหาไฟล์รูปภาพชนิดต่างๆ ได้
2. การค้นหาแบบ Search local จะสามารถกำหนดจำนวนรูปภาพที่เราต้องการค้นหาได้
3. การค้นหาแบบ Search via P2P จะไม่สามารถกำหนดจำนวนภาพที่ต้องค้นหาได้ เนื่องจากมีข้อจำกัดในส่วนของเรื่องเวลาในการค้นหา ดังนั้นโปรแกรมจะทำการเปรียบเทียบใน Data base ให้ค้นหาในแต่ละ Peer ได้แค่Peer ละ 3 ภาพเท่านั้น
4. ถ้ามองในเรื่องของการวิเคราะห์สี หรือค่าเฉลี่ยสีของภาพที่มาเปรียบเทียบกันโปรแกรมนี้จะให้ผลได้ดี ถ้าไม่ยึดติดว่าจะต้องเป็นภาพชนิดเดียวกัน
5. เวลาในการค้นหารูปภาพจะขึ้นอยู่กับจำนวน Peer ที่เราจะค้นหา คือถ้าจำนวน Peer มากขึ้นเวลาก็จะเพิ่มขึ้นตาม
6. การค้นหาแบบ Local จะใช้เวลาน้อยกว่าการค้นหาแบบ Peer

5.2 ปัญหาที่พบ

1. เนื่องจากโปรแกรมนี้เป็นโปรแกรมที่ใช้วิธีการวิเคราะห์ค่าเฉลี่ยของสี จึงมีบางกรณีคือภาพที่ได้จากการค้นหาอาจไม่ใช่ภาพชนิดเดียวกันกับภาพที่ต้องการ แต่ถ้ามองในเรื่องสีและผลการคำนวณหาค่า Distance ถือว่าภาพนั้นมีลักษณะใกล้เคียงกัน
2. ในการค้นหาในระบบNetworkจะมีปัญหาในเรื่องของเวลา และจำนวนภาพที่นำไปเปรียบเทียบใน Data base จึงกำหนดให้มีการค้นหาในแต่ละ Peer เพียงสามรูป
3. ในส่วนของรูปภาพจะทำให้ผลการค้นหาลดลงเนื่องจากขนาดของfile และSize ของภาพ เพราะโปรแกรมนี้จะได้ผลดีก็ต่อเมื่อขนาดของ file และ Size ภาพเท่ากัน
4. ในส่วนของโปรแกรมนี้ไม่สามารถทำให้ได้เหมือนโปรแกรม Bit Comet ที่มีการ Share ข้อมูลกันตลอดเวลา
5. ไม่สามารถบอกได้ว่าภาพที่ได้มาจากการค้นหาแบบ P2P ว่าภาพมาจาก Client ตัวไหน

5.3 ข้อเสนอแนะ

1. ควรมีข้อมูลในฐานข้อมูลที่ดีพอ เพราะผลการค้นหาที่ได้ภาพไม่ตรงตามความต้องการนั้นอาจมีผลมาจากข้อมูลในฐานข้อมูล ที่มีขนาดของภาพที่ไม่เท่ากัน
2. หากมีการพัฒนาต่อไปควรศึกษาในส่วนของ P2P Network และ Shapes Analysis
3. ควรจะพัฒนาให้มีคุณสมบัติเหมือนโปรแกรม Bit Comet และสามารถค้นหารูปภาพที่ไม่จำกัดขนาดได้และเวลาที่ใช้ในการค้นหารูปภาพนั้นควรไม่นานเกินไป

เอกสารอ้างอิง

[1] Corinna John . "A Simmple C# Wrapper for the AviFile Library." [Online], Available: http://www.codeproject/.2004

[2] Daniel Strigl. "DirectShow MediaPlayer in C#." [Online]. Available: http://www.codeproject.com/.2004

[3] Eric-Paul. "ImageConverter – Converts images to a specific image format, changing sizes on the flow." [Online]. Available: http://www.codeproject.com/.2004

ภาคผนวก ก

Source Code การจัดการฐานข้อมูล

ก-1 การรับค่าไฟล์รูปภาพที่จะเอาไปหาค่า hsvHistogram แล้ว Add ลง Data base

```
private void buttonOpenImageFile_Click(object sender, EventArgs e)
{
if (openFileDialog1.ShowDialog() == DialogResult.OK)
  {
    BackgroundWorker computeHistogramWorker = new BackgroundWorker();
    computeHistogramWorker.WorkerReportsProgress = true;
    computeHistogramWorker.DoWork += new
    DoWorkEventHandler(computeHistogramWorker_DoWork);
    computeHistogramWorker.RunWorkerCompleted += new
    RunWorkerCompletedEventHandler(computeHistogramWorker_RunWorkerCompleted);
    computeHistogramWorker.ProgressChanged += new
    ProgressChangedEventHandler(computeHistogramWorker_ProgressChanged);
    Bitmap bmp = new Bitmap(openFileDialog1.FileName);
    pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);
    computeHistogramWorker.RunWorkerAsync(new object[] { bmp, computeHistogramWorker
});
    buttonAddImageToDB.Enabled = false;
    buttonCBIRSearchLocal.Enabled = false;
    buttonSearchViaP2P.Enabled = false;
  }
}
```

ก-2 การนำไฟล์ภาพไปคำนวณหาค่า hsvHistogram โดย เลือก class ImageUtils

```
void computeHistogramWorker_DoWork(object sender, DoWorkEventArgs e)
    {
      Bitmap bmp = (Bitmap)((object[])e.Argument)[0];
      BackgroundWorker bgw = (BackgroundWorker)((object[])e.Argument)[1];
      int[,,] hsvHist = ImageUtils.GetHSV18x3x3Histogram(bmp, bgw);
      e.Result = hsvHist;
    }
```

ก-3 Code ImageUtils เป็น class การหาค่า hsvHistogram

```
public class ImageUtils
  {
public statie int[,,] GetHSV18x3x3Histogram(Bitmap bmp,BackgroundWorker bgw)
    {
      int[,,] hsvHistogram = new int[18,3,3];
      float hInterval = 360.0f / 18.0f;
      float sInterval = 1.0f / 3.0f;
      float vInterval = sInterval;

      for (int y = 0; y < bmp.Height; y++)
        {
          bgw.ReportProgress((int)(((float)y /
        (float)bmp.Height)*100.0f));
          Thread.Sleep(5);

        //Loop through all image pixels
        for (int x = 0; x < bmp.Width;x++ )
        {
          Color c = bmp.GetPixel(x,y);
          float h = c.GetHue();
          float s = c.GetSaturation();
```

```
float v = c.GetBrightness();
int hBinIdx = 0;
int sBinIdx = 0;
int vBinIdx = 0;
float tmp;

//Hue quantization
tmp = 0.0f;
do{
   if (h >= tmp && h < (tmp + hInterval))  break;
   tmp += hInterval;
   hBinIdx++;
} while (hBinIdx <= 17);

//Saturation quantization
tmp = 0.0f;
do
{
   if(s >= tmp && s < (tmp + sInterval)) break;
   tmp += sInterval;
   sBinIdx++;
} while (sBinIdx <= 2);

//Valuc quantization
tmp = 0.0f;
do
{
   if (v >= tmp && v < (tmp + vInterval)) break;
   tmp += vInterval;
   vBinIdx++;
} while (vBinIdx <= 2);
```

```
            hsvHistogram[hBinIdx, sBinIdx, vBinIdx]++;
        }
    }

    return hsvHistogram;
}
public static int[, ,] GetHSV18x3x3Histogram(string hsvHistogramString)
    {
    string[] split = hsvHistogramString.Split(",".ToCharArray());
    int[,,] hsvHistogram=new int[18,3,3];
    int idx = 0;

    for (int h = 0; h < 18; h++)
      for (int s = 0; s < 3; s++)
        for (int v = 0; v < 3;v++ )
        {
          hsvHistogram[h, s, v] = int.Parse(split[idx]);
          idx++;
        }

    return hsvHistogram;
}

public static double GetL2Distance(int[, ,] histH, int[, ,] histG)
{
    double sum = 0.0d;

    for (int h = 0; h < 18; h++)
    {
      for (int s = 0; s < 3; s++)
      {
        for (int v = 0; v < 3; v++)
```

```
            {
                double x = (double)histH[h, s, v];
                double y = (double)histG[h, s, v];

                sum += Math.Pow((x - y), 2);
            }
        }
    }

    sum = Math.Sqrt(sum);
    return sum;
}

public static double GetHistogramIntersection(int[,,] histH,int[,,] histG)
{
    double hi = 0.0d;
    double histHMagnitude = 0.0d;
    double histGMagnitude = 0.0d;

    for (int h = 0; h < 18; h++)
    {
        for (int s = 0; s < 3; s++)
        {
            for (int v = 0; v < 3; v++)
            {
                double x = (double)histH[h, s, v];
                double y = (double)histG[h, s, v];
                histHMagnitude += x;
                histGMagnitude += y;

                hi += Math.Min(x,y);
            }
```

```
            }
        }

        hi /= Math.Min(histHMagnitude, histGMagnitude);
        hi = 1 - hi;

        return hi;
    }
}
```

ก-4 Progress bar เริ่มเปลี่ยนจนกว่าการคำนวณหาค่า hsvHistogram จะเสร็จแล้วจะ Add ค่าของ hsvHistogram ลง Data base ในส่วนของการ Add ค่าของ hsvHistogram ลง Data base จะเลือก class ImageDatabase มาช่วย

```
void computeHistogramWorker_DoWork(object sender, DoWorkEventArgs e)
    {
        Bitmap bmp = (Bitmap)((object[])e.Argument)[0];
        BackgroundWorker bgw =
        (BackgroundWorker)((object[])e.Argument)[1];
        int[,,] hsvHist = ImageUtils.GetHSV18x3x3Histogram(bmp, bgw);

        e.Result = hsvHist;

    }
void computeHistogramWorker_ProgressChanged(object sender,
    ProgressChangedEventArgs e)
    {
        progressBarOpenImageFile.Value = e.ProgressPercentage;
    }

void computeHistogramWorker_RunWorkerCompleted(object sender,
    RunWorkerCompletedEventArgs e)
```

```
{
    int[,,] hsvHist = (int[,,])e.Result;

    textBoxHSVHistogram.Text = "";
    StringBuilder sb = new StringBuilder();
    foreach(int x in hsvHist)
        sb.Append(x.ToString() + ",");
    textBoxHSVHistogram.AppendText(sb.ToString());

    //Add image data into local database with its HSV-histogram data
    ImageDatabase imgDB = new ImageDatabase();
    imgDB.Open( ImageDatabase.ImageDatabaseFile);
    imgDB.AddImage(openFileDialog1.FileName, sb.ToString());
    imgDB.Close();

    buttonAddImageToDB.Enabled = true;
    buttonCBIRSearchLocal.Enabled = true;
    buttonSearchViaP2P.Enabled = true;
}
```

ก-5 Code ImageDatabase เป็น class ในการจัดการ Data base ทั้งหมด

```
public class ImageDatabase
  {
    public static string ImagesPath;
    public static string ImageDatabaseFile;

    OleDbConnection m_dbConn = null;

    public ImageDatabase() {
      m_dbConn = new OleDbConnection();
    }

public void Open(string dbLocation) {
      m_dbConn.ConnectionString =
      "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dbLocation ;
      m_dbConn.Open();
    }

public void Close()
    {
      m_dbConn.Close();
    }

    public void AddImage(string location,string hsvHistogram) {

      lock (m_dbConn)
      {

        if (m_dbConn.State == System.Data.ConnectionState.Closed)
        return;
```

```
string sqlCheckExist = "SELECT * FROM [ImageData] WHERE
location='" + location + "'";
OleDbCommand cmdCheckExist = new OleDbCommand(sqlCheckExist,
m_dbConn);
cmdCheckExist.ExecuteNonQuery();
OleDbDataReader dr = cmdCheckExist.ExecuteReader();



if (dr.HasRows)
{
   dr.Read();
   string oldHistogram =
   (string)dr.GetValue(dr.GetOrdinal("hsvHistogram"));
   if (String.Compare(hsvHistogram, oldHistogram) == 0)
   {
      //histogram are equal
      //do nothing
   }
   else
   {
      //not equal
   string sqlUpdateHist = "UPDATE [ImageData] SET
   hsvHistogram='" + hsvHistogram + "' WHERE location='" +
   location + "'";
      OleDbCommand cmdUpdateHist = new
      OleDbCommand(sqlUpdateHist, m_dbConn);
      cmdUpdateHist.ExecuteNonQuery();
   }
}
else
{
//Insert new
```

```
            string sqlInsert = "INSERT INTO
            [ImageData](location,hsvHistogram) VALUES('" + location
            + "','" + hsvHistogram + "')";
            OleDbCommand cmdInsert = new OleDbCommand(sqlInsert,
            m_dbConn);
            cmdInsert.ExecuteNonQuery();
        }
    }
}

public ArrayList SearchImageByHSVHistogram(string hsvHistogram,int
method,int numImage)
    {
      lock (m_dbConn)
      {

        if (m_dbConn.State == System.Data.ConnectionState.Closed)
        return null;

      //transform source histogram from string-based to int-array-based
        int[, ,] queryHistogram =
        ImageUtils.GetHSV18x3x3Histogram(hsvHistogram);
        ArrayList resultList = new ArrayList();

        string sqlAll = "SELECT * FROM [ImageData]";
        OleDbCommand cmdAll = new OleDbCommand(sqlAll, m_dbConn);
        int numRow = cmdAll.ExecuteNonQuery();
        OleDbDataReader dr = cmdAll.ExecuteReader();

        List<int> recordToBeDelete = new List<int>();

        while (dr.Read())
```

```
        {
      //Get fields data from database.
            string strCompareHist =
           (string)dr.GetValue(dr.GetOrdinal("hsvHistogram"));
            string location =
           (string)dr.GetValue(dr.GetOrdinal("location"));
            int recordID = (int)dr.GetValue(dr.GetOrdinal("id"));
            double distance = 0.0d;

            if (!File.Exists(location)) {
              recordToBeDelete.Add(recordID);
              continue;
            }

       //Convert string histogram to int-array
            int[, ,] compareHist =
            ImageUtils.GetHSV18x3x3Histogram(strCompareHist);

            if ((int)DISTANCE_MEASURE_METHOD.L2DISTANCE == method)
              distance = ImageUtils.GetL2Distance(queryHistogram,
              compareHist);
            else

if ((int)DISTANCE_MEASURE_METHOD.HISTOGRAM_INTERSECTION == method)
  distance = ImageUtils.GetHistogramIntersection(queryHistogram,
  compareHist);
  P2PImage p2pi = new P2PImage(distance, location, strCompareHist);
            resultList.Add(p2pi);
          }
          if (recordToBeDelete.Count > 0)
          {
            StringBuilder sbRecordList = new StringBuilder();
```

```
            for (int i = 0; i < recordToBeDelete.Count;i++ )
            {
              if (i == recordToBeDelete.Count-1)
                sbRecordList.Append(recordToBeDelete[i].ToString());
              else
                sbRecordList.Append(recordToBeDelete[i].ToString() + ",");
            }
          string sqlDelete = "DELETE FROM [ImageData]WHERE id in(" + sbRecordList.ToString()
        + ")";
            OleDbCommand cmdDelete = new OleDbCommand(sqlDelete, m_dbConn);
            cmdDelete.ExecuteNonQuery();
          }
//Sort the result by distance (P2PImage implements IComparable interface)
          resultList.Sort();
          if (numImage > 0 && resultList.Count > numImage)
          {
            int numTailToRemove = resultList.Count - numImage;
            resultList.RemoveRange(numImage, numTailToRemove);
          }
          return resultList;
        }
      }
    }
```

Source code การจัดการโปรแกรมการค้นหารูปภาพ

ก-6 เป็นการค้นหารูปภาพแบบ Local โดยเอาไฟล์ของรูปภาพที่ต้องการค้นหามาหาค่า hsvHistogram แล้วนำเอาค่าของ hsvHistogram ของไฟล์นั้นไปเปรียบเทียบใน Data base ในการเปรียบเทียบจะใช้ Class P2PImage

```
private void buttonCBIRSearchLocal_Click(object sender, EventArgs e)
    {
      ImageDatabase imgDB = new ImageDatabase();
      imgDB.Open( Application.StartupPath + "\\ImageDB.mdb" );
      ArrayList imgList =   imgDB.SearchImageByHSVHistogram(textBoxHSVHistogram.Text,
     (int)DISTANCE_MEASURE_METHOD.HISTOGRAM_INTERSECTION,
      Convert.ToInt32(numericUpDown1.Value));
      imgDB.Close();
      flowLayoutPanel1.Controls.Clear();
      foreach (P2PImage x in imgList)
      {
        double dist = x.Distance;
        string loc = x.Location;
        PictureBox p = new PictureBox();
        p.MouseDoubleClick += new
        MouseEventHandler(pictureboxes_MouseDoubleClick);
        p.MouseLeave += new EventHandler(pictureboxes_MouseLeave);
        p.MouseMove += new
        MouseEventHandler(pictureboxes_MouseMove);
        p.SizeMode = PictureBoxSizeMode.StretchImage;
        p.Size = new Size(64,64);
        p.Image = Image.FromFile(loc);
        p.Image.Tag = loc;
        p.Tag = dist.ToString();
        flowLayoutPanel1.Controls.Add(p);          //Add PictureBox to FlowLayoutPane
      }
```

ก-7 Class P2PImage เป็น Class ใช้หาและเปรียบเทียบ Distance ของภาพที่ต้องการกับภาพใน Data base

```
public class P2PImage : IComparable
  {
    double m_distance = 0.0;
    string m_location = String.Empty;
    string m_imageBinaryInBase64String = String.Empty;
    string m_hsvHistogram = String.Empty;
    public double Distance {
      get {
        return m_distance;
      }
      set {
        m_distance = value;
      }
    }
    public string Location {
      get {
        return m_location;
      }
      set {
        m_location = value;
      }
    }
    public string ImageBinaryInBase64String {
      get {
        return m_imageBinaryInBase64String;
      }
      set {
        m_imageBinaryInBase64String = value;
      }
```

```
public string HSVHistogram {
    get {
        return m_hsvHistogram;
    }
    set {
        m_hsvHistogram = value;
    }
}
public P2PImage() {

}
public P2PImage(double distance,string location,string hsvHistogram) {
    this.m_distance = distance;
    this.m_location = location;
    this.m_hsvHistogram = hsvHistogram;
}
//implement IComparable's CompareTo method for sort by distance-measurement
public int CompareTo(object obj)
{
    P2PImage img = (P2PImage)obj;
    if (this.m_distance > img.m_distance)
    {
        return 1;
    }
    else if (this.m_distance < img.m_distance)
    {
        return -1;
    }
    else
    {
        return 0;
```

ก-8 การค้นหาแบบ P2P Network จะเป็นการร้องขอให้ Peer Ip address ที่ Connect อยู่กับ Index Server ค้นหาภาพที่ต้องการแล้วส่งมาให้ Peer ที่ร้องขอโดยจะรับ IP address จาก Class P2PUtils

```
private void buttonSearchViaP2P_Click(object sender, EventArgs e)
    {
      buttonSearchViaP2P.Enabled = false;
      buttonAddImageToDB.Enabled = false;
      buttonCBIRSearchLocal.Enabled = false;
      flowLayoutPanel1.Controls.Clear();
      m_bSearchingViaP2P = true;

      BackgroundWorker bgwSearchViaP2P = new BackgroundWorker();
      bgwSearchViaP2P.DoWork += new DoWorkEventHandler(bgwSearchViaP2P_DoWork);
      bgwSearchViaP2P.RunWorkerCompleted += new
RunWorkerCompletedEventHandler(bgwSearchViaP2P_RunWorkerCompleted);
      bgwSearchViaP2P.RunWorkerAsync();
    }

    void bgwSearchViaP2P_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs
e)
    {
      buttonSearchViaP2P.Enabled = true;
      buttonAddImageToDB.Enabled = true;
      buttonCBIRSearchLocal.Enabled = true;
    }

    void bgwSearchViaP2P_DoWork(object sender, DoWorkEventArgs e)
    {
      m_searchViaP2PStatus = REQUESTING_CLIENT_LIST;
      m_peerServerConnection.RequestClientList();
```

```
while (m_bSearchingViaP2P)
{
  switch (m_searchViaP2PStatus)
  {
    case REQUESTING_CLIENT_LIST:
      break;
    case REQUESTING_CLIENT_LIST_COMPLETE:
      IPAddress locIP = P2PUtils.getFirstLocalIPAddress();
      int locIPIdx = -1;
      for (int i = 0; i < m_cList.Count;i++ )
      {
        m_cList[i] = m_cList[i].Split(":".ToCharArray())[0];
        if (String.Compare(locIP.ToString(), m_cList[i]) == 0)
        {
          locIPIdx = i;
        }
      }
      if (locIPIdx != -1) { m_cList.RemoveAt(locIPIdx); }
      m_searchViaP2PStatus = SEARCHING_IMAGE_VIA_P2P;
      break;
    case SEARCHING_IMAGE_VIA_P2P:

      foreach (string ip in m_cList)
      {
        PeerAsClient pac = new PeerAsClient();
        pac.LogMessegeEvent += new LogMessegeEventHandler(pac_LogMessegeEvent);
        pac.ConnectedEvent += new ConnectedEventHandler(pac_ConnectedEvent);
        pac.ConnectFailEvent += new ConnectFailEventHandler(pac_ConnectFailEvent);
        pac.DisconnectedEvent += new
DisconnectedEventHandler(pac_DisconnectedEvent);
        pac.ImageObtainedEvent += new
```

```
            ImageObtainedEventHanlder(pac_ImageObtainedEvent);
        pac.ConnectToPeer(ip, 8002);
            pac.SearchImage(textBoxHSVHistogram.Text);
            pac.Disconnect();
        }
        m_bSearchingViaP2P = false;
        break;
    }

    Thread.Sleep(500);
  }
}
void pac_ConnectedEvent(object obj)
{

}
void pac_ConnectFailEvent(string errorMsg)
{

}
void pac_DisconnectedEvent(object obj)
{

}
void pac_LogMessegeEvent(string str)
{

}
void pac_ImageObtainedEvent(object obj)
{
  if (flowLayoutPanel1.InvokeRequired)
  {
```

```
        ObjectAsParamCallBack d = new
        ObjectAsParamCallBack(m_peerAsClient_ImageObtainedEvent);
          this.Invoke(d, new object[] { obj });
      }
      else
      {
        P2PImage pi = (P2PImage)obj;

        PictureBox p = new PictureBox();
        p.MouseDoubleClick += new MouseEventHandler(pictureboxes_MouseDoubleClick);
        p.MouseLeave += new EventHandler(pictureboxes_MouseLeave);
        p.MouseMove += new MouseEventHandler(pictureboxes_MouseMove);
        p.SizeMode = PictureBoxSizeMode.StretchImage;
        p.Size = new Size(64, 64);
        try
        {
          p.Image = Image.FromFile(pi.Location);
        }
        catch (FileNotFoundException fnfe)
        {

        }
        p.Image.Tag = pi.Location;
        p.Tag = pi.Distance.ToString();

        flowLayoutPanel1.Controls.Add(p);
      }
    }
```

ก-9 Code Class P2PUtils เป็น Class ที่ใช้รับ IP Addresses ของ Peer ที่เชื่อมต่ออยู่

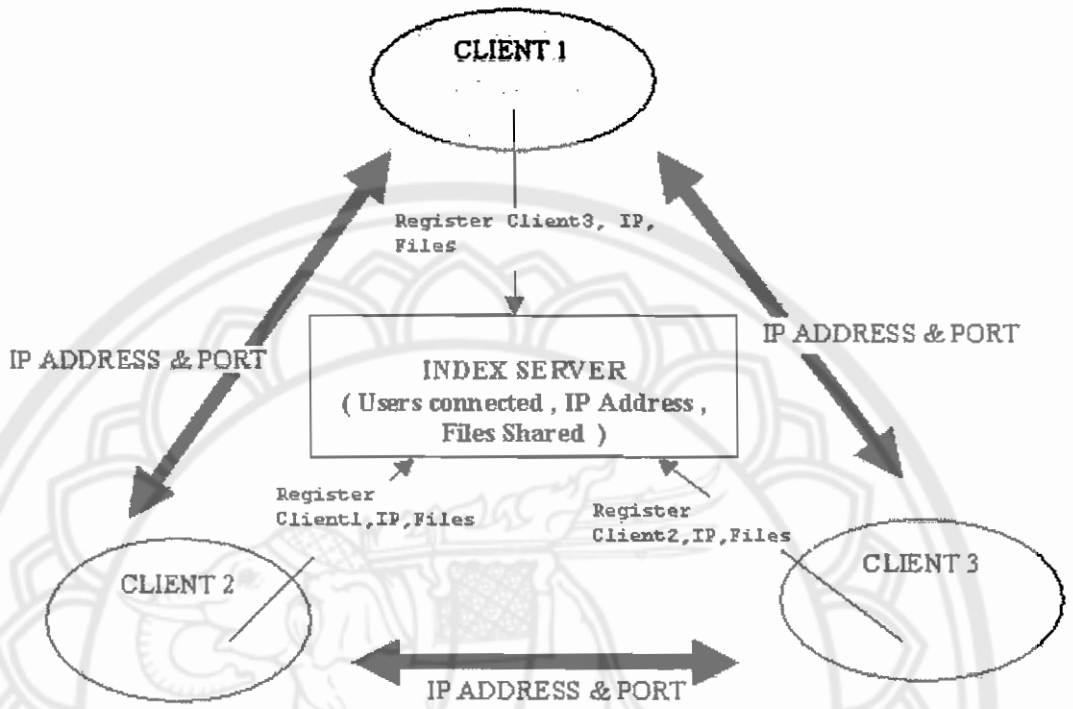
```
public class P2PUtils
  {
    public static IPAddress[] getLocalIPAddresses()
    {
      String strHostName = Dns.GetHostName();
      IPHostEntry ipHostEntry = Dns.GetHostEntry(strHostName);

      return ipHostEntry.AddressList;
    }

    public static IPAddress getFirstLocalIPAddress() {
      IPAddress localIP = null;
      String strHostName = Dns.GetHostName();
      IPHostEntry ipHostEntry = Dns.GetHostEntry(strHostName);
      foreach (IPAddress ipaddress in ipHostEntry.AddressList)
      {
        localIP = ipaddress;
        break;
      }
      return localIP;
    }
  }
```

ภาคผนวก ข

Index Server



Index Server จะเป็นตัว Register หรือรักษาคำแนะนำเกี่ยวกับผู้ใช้ ที่ Login เข้ามาใน Network โดยจะ Register ตำแหน่งที่อยู่ IP ของ Client และรายชื่อของไฟล์ที่แชร์ไว้ ณ เวลาขณะนั้น และ Index Server จะเป็นตัวจัดการให้ Client เชื่อมต่อกันเองโดยอัตโนมัติ

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net.Sockets;
using P2PFramework;
namespace IndexServer
```

```
{
    public partial class formIndexServer : Form
    {
        delegate void ConsoleAppendTextCallBack(string text);
        delegate void ClientListChangeCallBack(P2PClient clientSocket);

        private P2PIndexServer m_p2pIndexServer;

        public formIndexServer()
        {
            InitializeComponent();
            this.Disposed += new EventHandler(formIndexServer_Disposed);

            m_p2pIndexServer = new P2PIndexServer( 8001);
            m_p2pIndexServer.LogMessegeEvent += new
LogMessegeEventHandler(m_p2pIndexServer_LogMessegeEvent);
            m_p2pIndexServer.ClientConnectedEvent += new
ClientConnectedEventHandler(m_p2pIndexServer_ClientConnectedEvent);
            m_p2pIndexServer.ClientDisconnectedEvent +=new
ClientDisconnectedEventHandler(m_p2pIndexServer_ClientDisconnectedEvent);
        }

        void m_p2pIndexServer_ClientDisconnectedEvent(P2PClient p2pc)
        {
            if (listBoxClient.InvokeRequired)
            {
                ClientListChangeCallBack deleg = new
ClientListChangeCallBack(m_p2pIndexServer_ClientDisconnectedEvent);
                this.Invoke(deleg, new object[] { p2pc });
            }
            else
            {
```

```
            listBoxClient.Items.Remove(p2pc.ClientSocket.RemoteEndPoint.ToString());
        }
    }

    void m_p2pIndexServer_ClientConnectedEvent(P2PClient p2pc)
    {
        if (listBoxClient.InvokeRequired)
        {
            ClientListChangeCallBack deleg = new
ClientListChangeCallBack(m_p2pIndexServer_ClientConnectedEvent);
            this.Invoke(deleg, new object[] { p2pc });
        }
        else {
            listBoxClient.Items.Add(p2pc.ClientSocket.RemoteEndPoint.ToString());
        }
    }

    void m_p2pIndexServer_LogMessegeEvent(string str)
    {
        if (this.textBoxConsole.InvokeRequired)
        {
            ConsoleAppendTextCallBack deleg = new
ConsoleAppendTextCallBack(m_p2pIndexServer_LogMessegeEvent);
            this.Invoke(deleg, new object[] { str });
        }
        else {
            textBoxConsole.AppendText(str);
        }
    }
    void formIndexServer_Disposed(object sender, EventArgs e)
    {
```

```
        m_p2pIndexServer.StopServer();
    }

    private void formIndexServer_Load(object sender, EventArgs e)
    {
        m_p2pIndexServer.StartServer();
    }

    private void buttonClearConsole_Click(object sender, EventArgs e)
    {
        textBoxConsole.Text = "";
    }
    private void buttonSendBroadCastMessage_Click(object sender, EventArgs e)
    {
        m_p2pIndexServer.SendBroadcastMessage(textBoxBroadCastMessege.Text);
    }

    private void buttonListActiveClient_Click(object sender, EventArgs e)
    {
        m_p2pIndexServer.ListActiveClient();
    }
  }
}
```