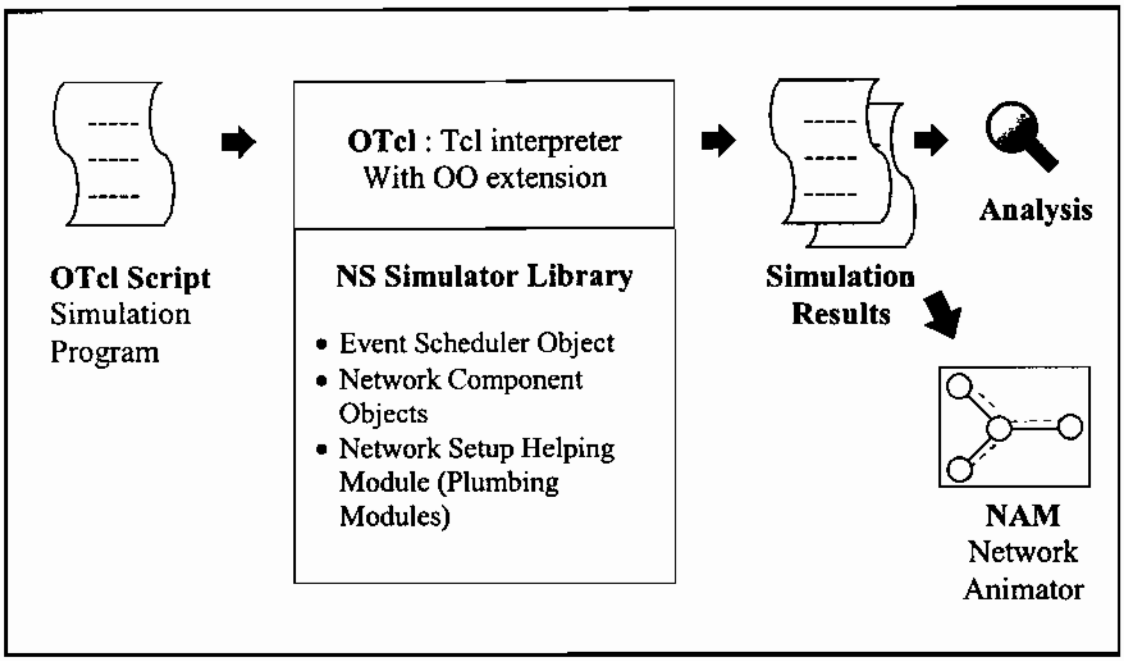
บทที่ 3
วิธีดำเนินการวิจัย

ในบทนี้จะกล่าวถึงการสร้างแบบจำลองเครือข่ายเซ็นเซอร์ไร้สายที่ใช้ในการทดลองโดยใช้วิธีการเขียนโปรแกรมชุดคำสั่งและจำลองการรับส่งข้อมูลระหว่างโหนดด้วยโปรแกรม The Network Simulator ขั้นตอนการกำหนด Location Information Error ให้กับตำแหน่งของโหนด การกำหนด Simulation Parameter ที่ใช้ในการทดลอง ขั้นตอนการแก้ปัญหาเมื่อเกิด Location Information Error ด้วยข่ายงานระบบประสาท และการทดลองด้วยข่ายงานระบบประสาท

3.1 การสร้างแบบจำลองเครือข่ายด้วยโปรแกรม The Network Simulator

3.1.1 โครงสร้างและสถาปัตยกรรมของ The Network Simulator

โปรแกรม The Network Simulator หรือเรียกย่อว่า NS ได้รับการพัฒนาจากนักวิจัยของ University of California, Berkeley [19] ที่ทำการวิจัยเกี่ยวกับเครือข่ายคอมพิวเตอร์โดยการสร้างแบบจำลองการทำงานบนเครือข่ายคอมพิวเตอร์ ซึ่งใช้ NS เป็นเครื่องมือจำลองการทำงานของโพรโทคอลต่างๆ ตัวอย่างเช่น TCP/IP (Transmission Control Protocol/Internet Protocol) UDP (User Datagram Protocol) พฤติกรรมการจราจรของเครือข่าย เช่น FTP (File Transfer Protocol) Telnet WWW CBR (Constant Bit Rate) และ VBR (Variable Bit Rate) วิธีการในการบริหารจัดการคิวของ Router เช่น Drop Tail RED (Random Early Detection) CBQ (Class-Based Queuing) ขั้นวิธีการค้นหาเส้นทาง (Routing Algorithm) ตัวอย่างเช่น Dijkstra Algorithm และอื่นๆ NS สามารถจำลองการทำงานแบบ Multicasting และโพรโทคอลบางตัวที่ทำงานในระดับ MAC Layer สำหรับเครือข่ายคอมพิวเตอร์เฉพาะบริเวณ (Local Area Network : LAN) NS เป็นส่วนหนึ่งของโครงการ VINT (Virtual InterNetwork Testbed)[19] ที่พัฒนาเครื่องมือสำหรับสร้างแบบจำลองในการแสดงผล การวิเคราะห์ และทำหน้าที่แปลง Topology ของเครือข่ายให้อยู่ในรูปแบบของ NS ซึ่งปัจจุบัน NS ที่มีใช้งานอยู่ (เวอร์ชัน 2) เขียนด้วย C++ และ OTcl (Tcl Script language with Object-oriented พัฒนาเพิ่มเติมโดย MIT) ซึ่งในเอกสารฉบับนี้ผู้วิจัยจะอธิบายสรุปโครงสร้างพื้นฐานของ NS และขั้นตอนการใช้งานโดยมีตัวอย่างประกอบ

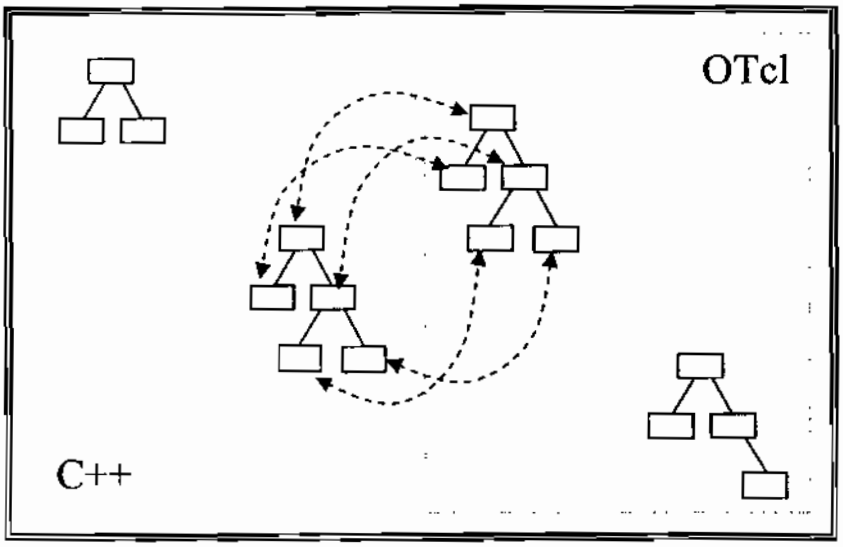


ภาพที่ 3-1 ลำดับการทำงานของ NS [19]

จากภาพที่ 3-1 แสดงให้เห็นถึงลำดับการทำงานของ NS ซึ่ง NS ก็คือ Object – oriented TCL (OTCL) ที่ทำหน้าที่ในการสร้างสคริปต์และแปลการทำงานจากสคริปต์ให้อยู่ในรูปของการสร้างแบบจำลองเกี่ยวกับเหตุการณ์ที่เกิดขึ้นและองค์ประกอบต่างๆ ของระบบเครือข่ายและโมดูลการสร้างเครือข่าย หรือกล่าวอีกนัยหนึ่ง คือการใช้ NS กับโปรแกรมที่เขียนด้วยสคริปต์ของภาษา OTcl ในการสร้างและดำเนินการเกี่ยวกับการสร้างแบบจำลองเครือข่าย ผู้ใช้งานอาจจะเริ่มต้นการเขียนสคริปต์ OTcl ด้วยการกำหนดตารางเวลาในการเกิดเหตุการณ์ (Event Scheduler Object) สร้าง Topology ของเครือข่ายโดยใช้วัตถุของเครือข่าย (Network Component Object) และฟังก์ชันพลัมบิ่ง (Plumbing Function) ที่อยู่ในไลบารี่ของ NS (NS Simulator Library) จากนั้นกำหนดการจราจรที่เกิดขึ้นบนเครือข่ายเมื่อต้องการจะเริ่มหรือหยุดรับส่ง Packet ตลอดทั้งตารางเวลาเหตุการณ์ที่เกิดขึ้น การเรียกใช้พลัมบิ่งสำหรับการติดตั้งเครือข่าย เนื่องจากในขั้นตอนการติดตั้งเครือข่ายจะมีการกำหนดเส้นทางในการเดินของข้อมูลที่อาจจะเป็นไปได้ในระหว่างวัตถุของเครือข่ายโดยการกำหนดตัวชี้ของวัตถุไปยังที่อยู่ "เพื่อนบ้าน (neighbor)" ของวัตถุที่ได้จัดสรรไว้ เมื่อผู้ใช้ต้องการที่จะสร้างเครือข่ายขึ้นมาใหม่ ก็สามารถทำได้ง่ายจากการสร้างวัตถุแบบเดียวกันโดยการเขียนวัตถุขึ้นมาใหม่หรือกำหนดส่วนประกอบของวัตถุด้วยไลบรารี่วัตถุ (Object library) และส่งข้อมูลผ่านทางวัตถุนั้น

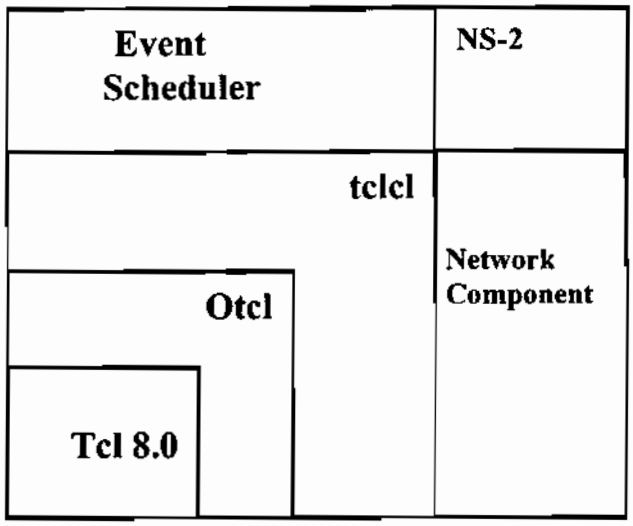
องค์ประกอบอย่างอื่นของ NS ที่นอกเหนือจากวัตถุของเครือข่าย (Network Object) คือ ตารางเวลาลำดับเหตุการณ์ (Network Scheduler Event) ซึ่งเหตุการณ์ที่อยู่ใน NS คือ Packet ID ของแต่ละ Packet กับช่วงเวลาและตัวชี้ตำแหน่งของวัตถุที่รองรับกับเหตุการณ์นั้นๆ ใน NS ตารางเวลา

ลำดับเหตุการณ์จะทำการเก็บลำดับการเกิดเหตุการณ์ต่างๆ ตามช่วงเวลาที่เกิดขึ้นและทำหน้าที่กระตุ้นการเกิดเหตุการณ์ต่างๆ ด้วยตารางคิวการเกิดเหตุการณ์สำหรับช่วงเวลาปัจจุบันโดยการอ้างอิงกับองค์ประกอบของเครือข่ายที่จัดสรรไว้



ภาพที่ 3-2 ความสัมพันธ์ระหว่าง C++ และ OTcl [19]

NS ไม่ได้ถูกพัฒนาขึ้นด้วยภาษา OTcl เท่านั้นแต่มีส่วนของการพัฒนาด้วยภาษา C++ ร่วมด้วยเหตุผลเพื่อสร้างความเสถียรภาพในการทำงาน NS มีการสร้างในส่วนของ Data Path และ Control Path แยกออกจากกัน เพื่อลดจำนวน Packet และเวลาที่ใช้ในการประมวล สำหรับตารางเวลาการเกิดเหตุการณ์และส่วนประกอบพื้นฐานของวัตถุเครือข่าย จะถูกพัฒนาและคอมไพล์ด้วย C++ Compiler เนื่องจาก C++ สนับสนุนในเรื่องการพัฒนาโปรแกรมเชิงวัตถุ ซึ่งวัตถุที่ผ่านการคอมไพล์แล้วจะถูกนำไปใช้ใน OTcl ทำหน้าที่เป็นตัวแปลง (Interpreter) และเชื่อมโยงวัตถุเหล่านั้นเพื่อสร้างเป็นฟังก์ชั่นควบคุมสำหรับใช้งานใน NS จากภาพที่ 3-2 เป็นการแสดงให้เห็นถึงความสัมพันธ์ระหว่างวัตถุที่สร้างด้วย C++ และการเชื่อมโยงโดย OTcl



ภาพที่ 3-3 สถาปัตยกรรมของ NS [19]

ภาพที่ 3-3 แสดงให้เห็นถึงสถาปัตยกรรมของ NS ซึ่งผู้ใช้งานทั่วไป ซึ่งไม่ใช่ NS Developer ที่ตำแหน่งมุมบนขวามือของภาพ สามารถออกแบบและใช้งาน Tcl โดยใช้ Simulator Object ที่อยู่ใน OTcl library ตารางเวลาเหตุการณ์และส่วนประกอบอื่นๆ ของเครือข่ายโดยมากจะถูกพัฒนาด้วยภาษา C++ และปรากฏอยู่ใน OTcl ตลอดจนการเชื่อมต่อที่พัฒนาขึ้นด้วย Tclcl

เมื่อกลับไปพิจารณาภาพที่ 3-1 จะพบว่า เมื่อการสร้างแบบจำลองเสร็จสิ้น NS จะสร้างไฟล์ข้อมูลที่ประกอบไปด้วยข้อความที่แสดงรายละเอียดการสร้างแบบจำลองของข้อมูล ซึ่งข้อมูลดังกล่าวสามารถไปใช้ในการสร้างแบบจำลองการวิเคราะห์หรือสร้างแบบจำลองทางด้านกราฟิกส์ ซึ่งใช้เครื่องมือที่เรียกว่า Network Animator(NAM)[19] ที่ได้รับการพัฒนามาจากโครงการ VINT NAM จะช่วยให้ผู้ใช้งานสามารถมองเห็นภาพรวมในการทำงานของ Topology ที่อยู่ในรูปกราฟิกส์ได้ง่ายและชัดเจน เนื่องจากสามารถควบคุมการแสดงผลต่างๆ ได้ เช่น Play Forward Reverse Pause หรืออื่นๆ และที่เพิ่มเติมคือ การแสดงผลข้อมูลที่อยู่ในรูปของกราฟิกส์ ตัวอย่างเช่น ปริมาณของ Packet ที่ Drop ในแต่ละ Link ถึงแม้ว่าการแสดงผลด้วยกราฟิกส์ไม่สามารถใช้สำหรับการสร้างแบบจำลองที่ต้องการวิเคราะห์ผลแม่นยำ

3.1.2 แบบจำลองไร้สายใน NS Simulator

โดยพื้นฐานแล้ว Wireless Model จะประกอบด้วย Mobile Node ที่สนับสนุนการสร้างแบบจำลอง Multi-Hop Ad-Hoc Network , Wireless LANs หรืออื่นๆ The Mobile Node Object จะแตกออกเป็นวัตถุ โดย Class Mobile Node นั้นมาจาก Class Node ที่เป็น Parent Class ของ Mobile Node ซึ่งในความเป็นจริงแล้วนั้น Mobile Node ก็คือ Basic Node Object ที่สามารถเพิ่ม

หน้าที่การทำงานด้าน Wireless และ Mobile, การรับ-ส่งสัญญาณผ่านทางช่องสัญญาณแบบไร้สาย เป็นต้น ซึ่งความแตกต่างสำคัญของ Mobile Node จาก Node คือ การที่ไม่จำเป็นต้องสร้างการเชื่อมโยงกับโหนดอื่น หรือ Mobile Node และส่วนที่เพิ่มเข้ามาในเรื่องความสามารถในการเคลื่อนที่ได้ง่ายประกอบไปด้วย การเคลื่อนที่ของโหนด การปรับปรุงตำแหน่งที่ตั้งตามช่วงเวลา การบำรุงรักษาขอบเขตของ Topology เป็นต้น ซึ่งสิ่งเหล่านี้ถูกพัฒนาด้วยภาษา C++ ที่ฟังก์ชั่นพลัมบิ่งของส่วนประกอบเครือข่าย ที่อยู่ภายใน Mobile Node ซึ่ง Ad-Hoc Routing Protocol ที่สนับสนุนการทำงานในปัจจุบัน ได้แก่ Destination Sequence Distance Vector (DSDV), Dynamic Source Routing (DSR), Temporally ordered Routing Algorithm (TORA) และ Ad-Hoc On-demand Distance Vector (AODV) โดยการสร้าง Mobile node พร้อมทั้งกำหนด Routing Protocol ที่ต้องใช้ดังนี้

```
set mnode [$opt(rp) –create-mobile-node $id]
```

ภาพที่ 3-4 Tcl Script การสร้าง Mobile Node [19]

โดยที่ $opt(rp) คือ การกำหนด Routing Protocol ที่ใช้ เช่น “dsdv”, “aodv”, “tora” หรือ “dsr” และการกำหนดค่าเริ่มต้นให้ Mobile Node ทำได้โดยการให้ข้อมูลของ Ad-Hoc Routing Protocol, Network Stack, Channel, Topography, Propagation Model และ Tracing ดังแสดงตามภาพที่ 3-5 จากนั้นจึงทำการสร้าง Mobile Node ขึ้นเพื่อใช้งานจริง โดยการใช้คำสั่งตามที่แสดงในภาพที่ 3-6

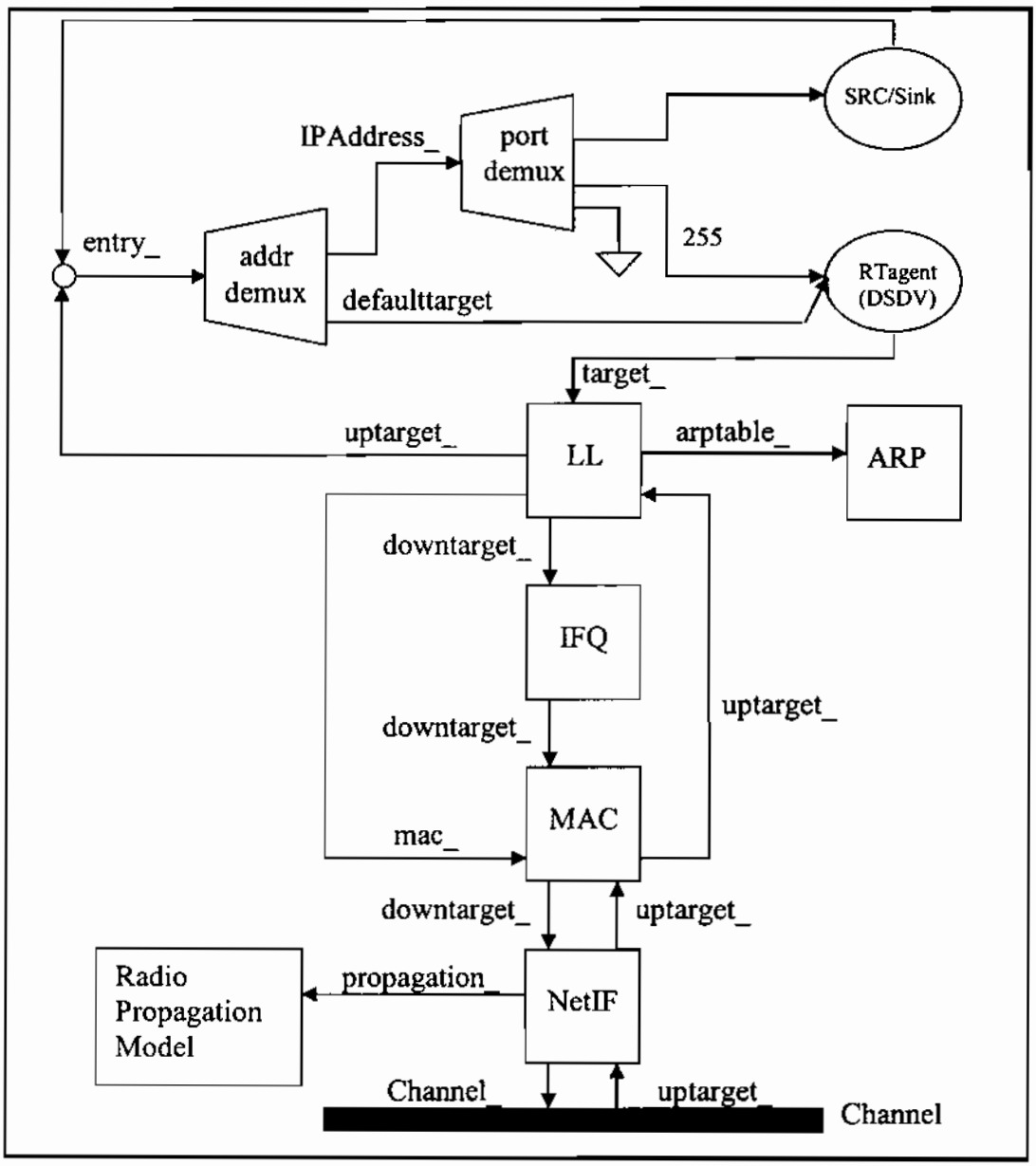
```
$ns_node-config -adhocRouting $opt (adhocRouting)
                -llType $opt(ll)
                -macType $opt(mac)
                -ifqType $opt(ifq)
                -ifqLen $opt(ifqlen)
                -antType $opt(ant)
                -propInstance [new $opt(prop)]
                -phyType $opt(netif)
                -channel [new $opt(chan)]
                -topoInstance $topo
                -wireRouting OFF
                -agentTrace ON
                -routerTrace OFF
                -macTrace OFF
```

ภาพที่ 3-5 Tcl Script กำหนดค่าเริ่มต้นให้กับ Mobile Node [19]

```
For { set j 0} { $j < $opt(nn)} {incr j} {
        set node_($j) [$ns_node]
        $node_($i) random-motion 0; #disable random motion
    }
```

ภาพที่ 3-6 Tcl Script การสร้าง Mobile Node เพื่อใช้งานจริง [19]

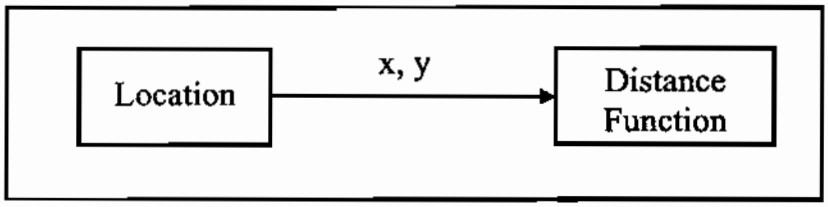
ภายใต้กระบวนการสร้าง Mobile Node Object จะทำการสร้าง Ad-Hoc Routing Agent สร้าง Network Stack ที่ประกอบด้วย Link Layer, Interface Queue, Mac Layer, และ การเชื่อมต่อเครือข่ายกับเสาอากาศ โดยการกำหนด Propagation Model ซึ่งการเชื่อมโยงแสดงดังภาพที่ 3-7



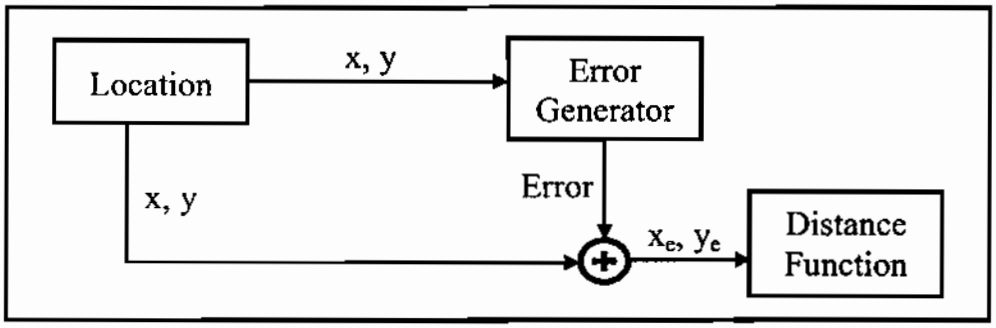
ภาพที่ 3-7 แผนผังของ Mobile Node ภายใต้การพัฒนาของ CMU Monarch's Wireless บน NS [19]

3.2 การสร้าง Location Information Error ให้กับตำแหน่งของโหนด

ข้อมูลเกี่ยวกับตำแหน่งที่ตั้ง (Location Information) ของโหนด บนเครือข่ายเซ็นเซอร์ไร้สาย เป็นปัจจัยสำคัญที่มีส่วนกำหนดประสิทธิภาพในการทำงานของขั้นตอนวิธีการค้นหาเส้นทางเชิงภูมิศาสตร์ เนื่องจากขั้นตอนการตัดสินใจทำ Greedy Forwarding จะต้องใช้ตำแหน่งที่ตั้งของแต่ละโหนดคำนวณระยะทาง ตามแสดงในภาพที่ 3-8 เพื่อพิจารณาเลือกโหนดที่จะส่ง Packet ต่อไป ซึ่งในการทดลองเราจะทำการเปรียบเทียบประสิทธิภาพการทำงานของ GPSR เมื่อเกิด Location Information Error กับ AODV และ DSR ซึ่งค่า Location Information Error คือ การที่แต่ละ Node จะให้ข้อมูลตำแหน่งที่ตั้งของตัวเองในการคำนวณระยะทางผิดพลาดไปจากความเป็นจริง ซึ่งเป็นสิ่งที่อาจเกิดขึ้นได้ในเครือข่ายเซ็นเซอร์ไร้สาย เมื่อทำการการส่งค่าตำแหน่งที่ผิดพลาดออกไปจะส่งผลให้โหนดข้างเคียงได้รับข้อมูลตำแหน่งที่ผิดพลาด ซึ่งจะส่งผลให้ประสิทธิภาพในการรับ-ส่งข้อมูลลดต่ำลง โดยในการทดลองนี้ผู้วิจัยได้สร้างค่าผิดพลาดของตำแหน่งของโหนดเพื่อจำลองการเกิดค่าความผิดพลาดในการบอกตำแหน่งของโหนด ดังแสดงในภาพที่ 3-9 จากการทำ Error Generator เพื่อกำหนดความน่าจะเป็นของความถี่ในการเกิดค่าผิดพลาดตามระดับความผิดพลาดที่ต้องการ ซึ่งในการทดลองนี้ได้กำหนดระยะทางของการเกิดผิดพลาด ที่ระยะ $\pm$5 เมตร $\pm$10 เมตร $\pm$25 เมตร และ $\pm$50 เมตร เนื่องจากโดยปกติแล้วการเกิดความผิดพลาดของตำแหน่งของอุปกรณ์ GPS จะเกิดอยู่ในช่วงตั้งแต่ $\pm$2 ไปจนถึง $\pm$30 เมตร [3] สำหรับข้อมูลนำเข้าจะผ่านฟังก์ชันความหนาแน่นของความน่าจะเป็น (Probability Density Function : PDF) ซึ่งมาจากการคำนวณด้วย Two-dimensional Gaussian Function ตามสมการ 3-1



ภาพที่ 3-8 การคำนวณตำแหน่งของ GPSR เมื่อทำงานแบบปกติ NS2 [19]



ภาพที่ 3-9 การสร้างค่าผิดพลาดตำแหน่งก่อนส่งเข้าสู่ Distance Function ที่ผู้วิจัยสร้างขึ้น

$$f(x,y) = A \exp\left[- \frac{\left((x-a)^2 + (y-b)^2\right)}{c(r)^2} \right]$$ โดยมีเมตริกซ์เป็น $\begin{bmatrix} a & b \\ b & c \end{bmatrix}$ (3-1)

และ

A คือ ความกว้าง (Amplitude)

exp คือ ฟังก์ชั่นเอ็กซ์โพแนนเชียล

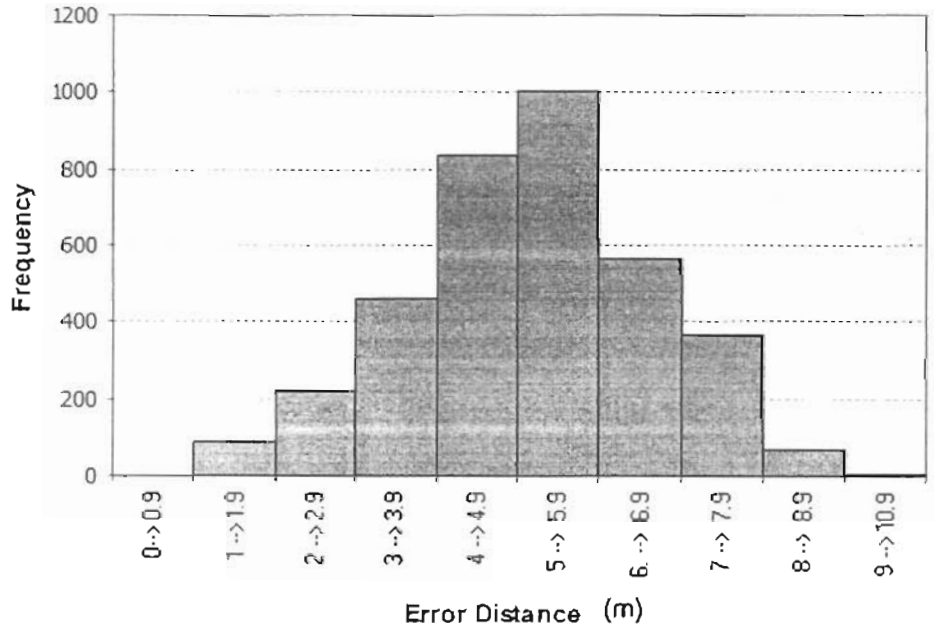
x , y คือ ข้อมูลนำเข้า

a คือ $\left(\frac{\cos\theta}{\sigma_x}\right)^2 + \left(\frac{\sin\theta}{\sigma_y}\right)$

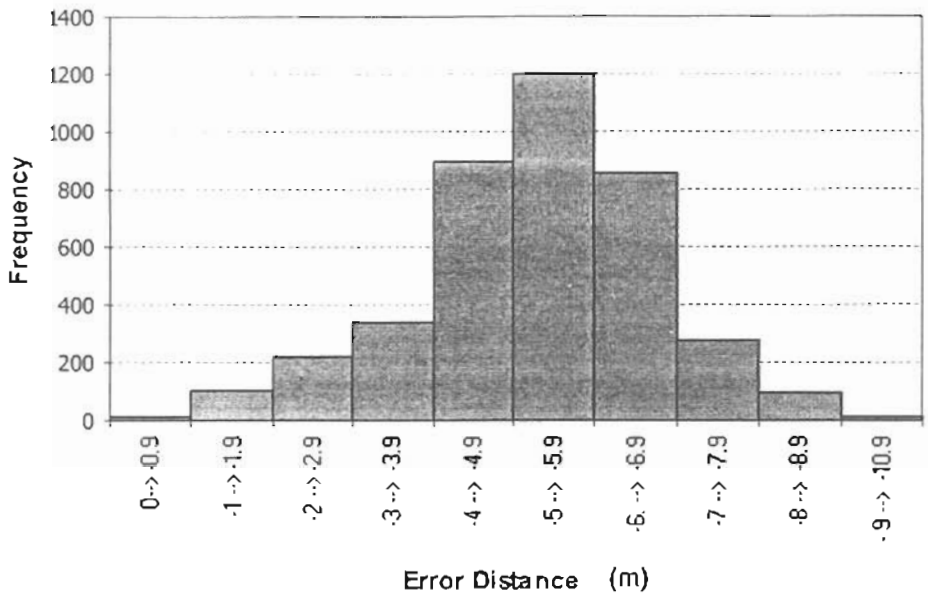
b คือ $-\frac{\sin 2\theta}{\sigma_x^2} + \frac{\sin 2\theta}{\sigma_y^2}$

c คือ $\left(\frac{\sin\theta}{\sigma_x}\right)^2 + \left(\frac{\cos\theta}{\sigma_y}\right)$

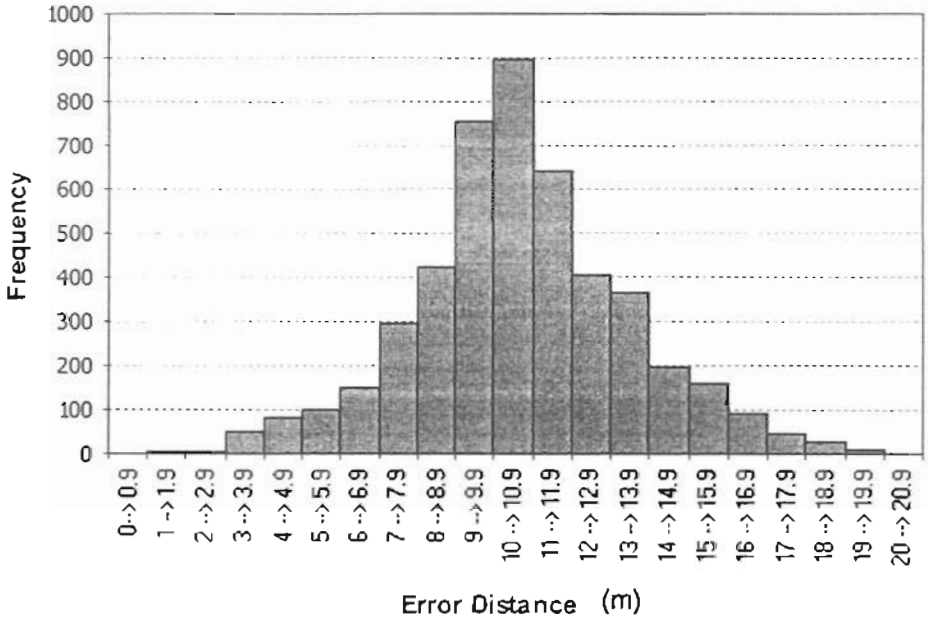
ซึ่งผลการเกิดค่าความผิดพลาดเมื่อผ่านฟังก์ชันความหนาแน่นของความน่าจะเป็น (Probability Density Function : PDF)แล้ว ได้แสดงให้เห็นในภาพที่ 3-10 ถึง ภาพที่ 3-17 โดยที่ค่าความผิดพลาดที่ระยะ ±5 เมตร แสดงให้เห็นตามภาพที่ 3-10 และภาพที่ 3-11 ค่าความผิดพลาดที่ระยะ ±10 เมตร แสดงให้เห็นตามภาพที่ 3-12 และ ภาพที่ 3-13 ค่าความผิดพลาดที่ระยะ ±25 เมตร แสดงให้เห็นตามภาพที่ 3-14 และภาพที่ 3-15 ค่าความผิดพลาดที่ระยะ± 50 เมตร แสดงให้เห็นตามภาพที่ 3-16 และภาพที่ 3-17



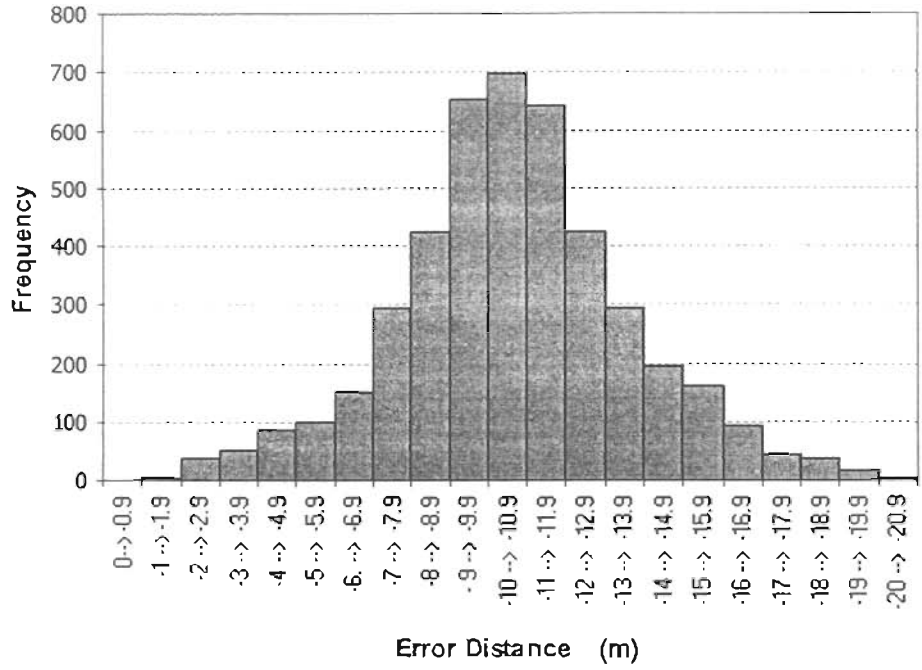
ภาพที่ 3-10 ความน่าจะเป็นของความถี่ในการเกิดค่าความผิดพลาดที่ระยะ +5 เมตร



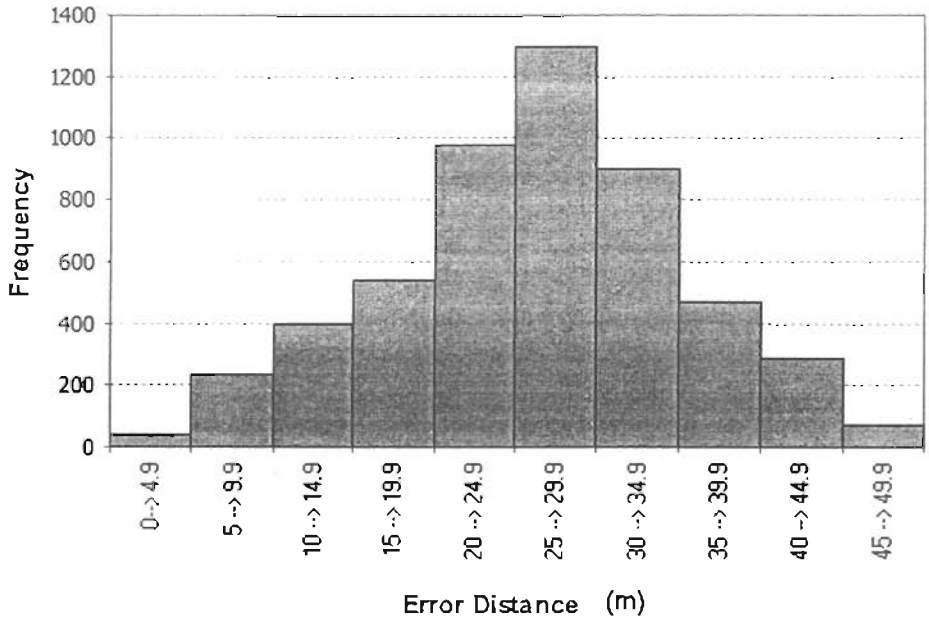
ภาพที่ 3-11 ความน่าจะเป็นของความถี่ในการเกิดค่าความผิดพลาดที่ระยะ -5 เมตร



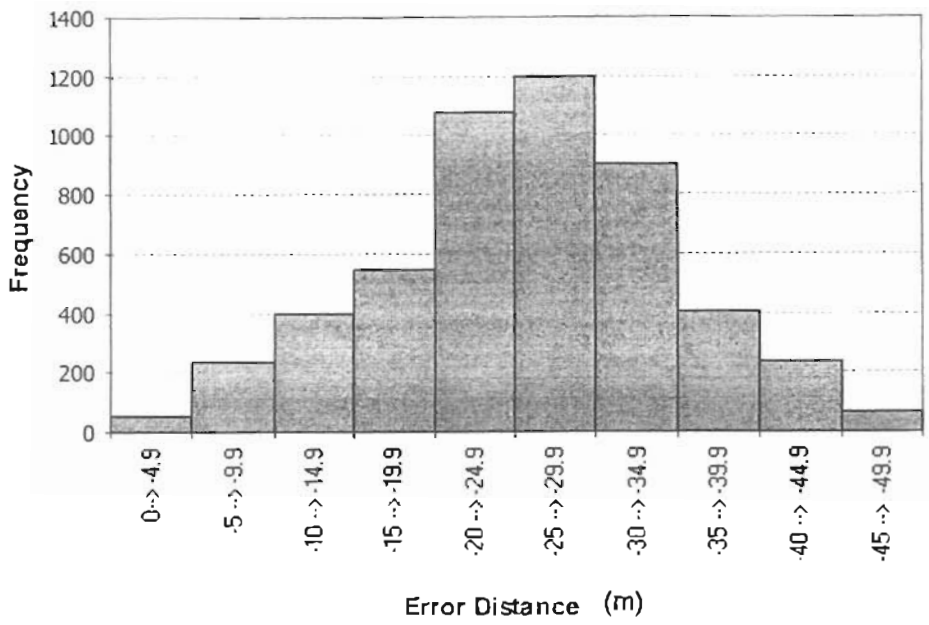
ภาพที่ 3-12 ความน่าจะเป็นของความถี่ในการเกิดค่าความผิดพลาดที่ระยะ +10 เมตร



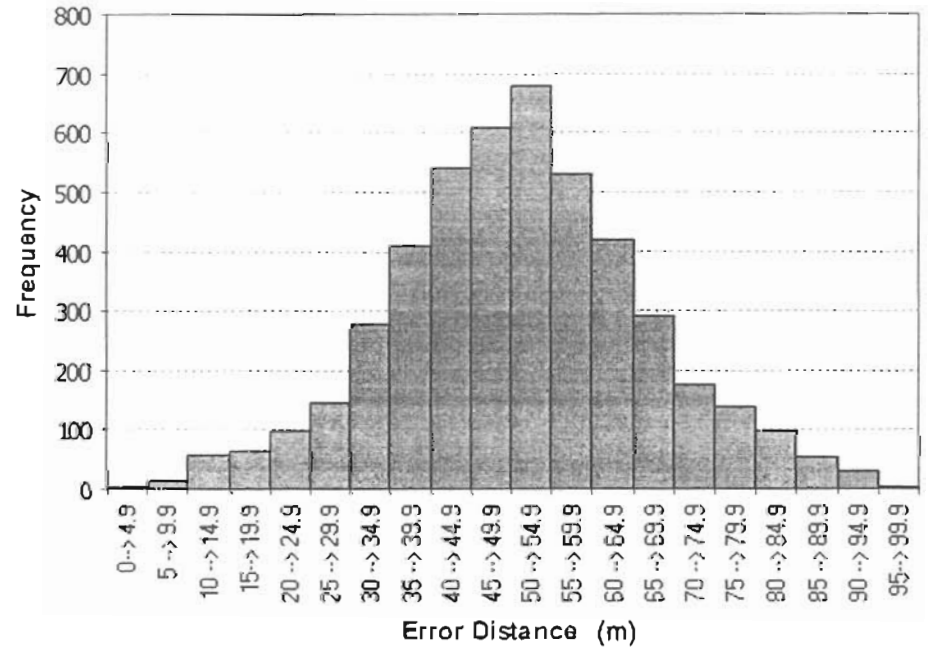
ภาพที่ 3-13 ความน่าจะเป็นของความถี่ในการเกิดค่าความผิดพลาดที่ระยะ -10 เมตร



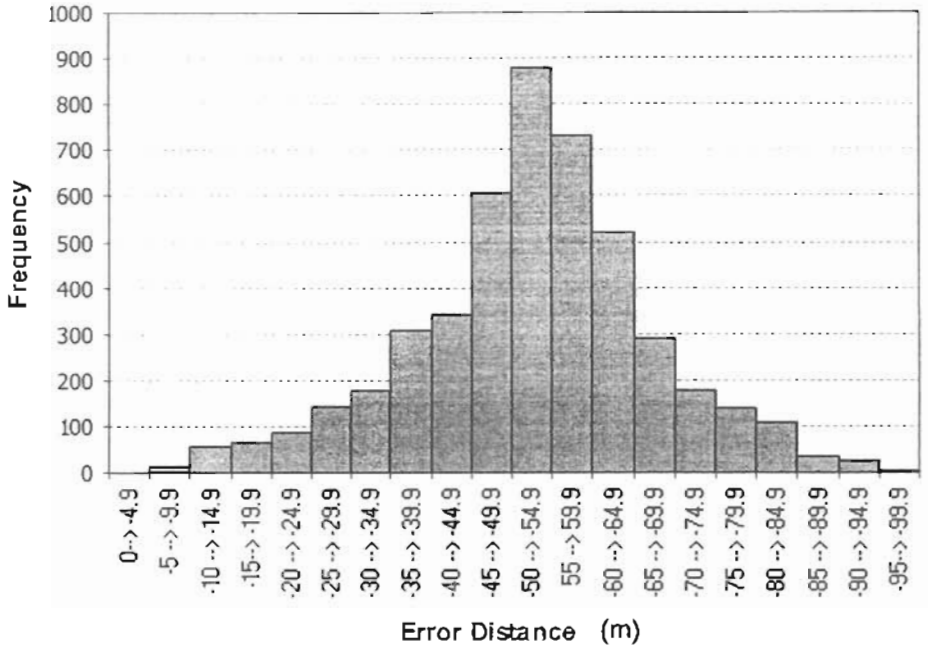
ภาพที่ 3-14 ความน่าจะเป็นของความถี่ในการเกิดค่าความผิดพลาดที่ระยะ +25 เมตร



ภาพที่ 3-15 ความน่าจะเป็นของความถี่ในการเกิดค่าความผิดพลาดที่ระยะ -25 เมตร



ภาพที่ 3-16 ความน่าจะเป็นของความถี่ในการเกิดค่าความผิดพลาดที่ระยะ +50 เมตร



ภาพที่ 3-17 ความน่าจะเป็นของความถี่ในการเกิดค่าความผิดพลาดที่ระยะ -50 เมตร

จากนั้นนำค่าความน่าจะเป็นของความถี่ในการเกิดค่าความผิดพลาดตามระยะที่ต้องการที่ได้บวกเพิ่มกับข้อมูลตำแหน่งของโหนดปัจจุบันที่ใช้งาน ซึ่งสามารถเขียนให้อยู่ในรูปของสมการดังแสดงในสมการที่ 3-2 หรือตามที่แสดงให้เห็นในภาพที่ 3-9 เพื่อกำหนดให้ตำแหน่งของโหนดเกิดค่าความผิดพลาด ก่อนที่จะส่งตำแหน่งของโหนดที่มีค่าความผิดพลาดแล้ว ไปคำนวณระยะทางตามสมการที่ 3-3 เพื่อตัดสินใจเลือกโหนดที่ดีที่สุดเป็นโหนดลำดับต่อไปสำหรับส่ง Packet

$$\text{Location } (x_e, y_e) = (x + e_x, y + e_y) \tag{3-2}$$

โดยที่ x_e คือ ตำแหน่ง x ที่บวกค่าความผิดพลาดแล้ว

y_e คือ ตำแหน่ง x ที่บวกค่าความผิดพลาดแล้ว

e_x คือ ค่าความผิดพลาดที่ ตำแหน่ง x

e_y คือ ค่าความผิดพลาดที่ ตำแหน่ง y

$$\text{Distance } (e,f) = \sqrt{(x_e - x_f)^2 + (y_e - y_f)^2} \tag{3-3}$$

โดยที่ x_e, y_e คือ ตำแหน่งที่ผ่านการคำนวณค่าผิดพลาดด้วย GPSR

x_f, y_f คือ ตำแหน่งของ Node ที่จะส่งข้อมูลเป็นลำดับถัดไป

3.3 การกำหนดค่าพารามิเตอร์ที่ใช้ในแบบจำลอง

การวิจัยนี้ทำการทดลองเพื่อเปรียบเทียบประสิทธิภาพของ GPSR , AODV , และ DSR ซึ่งทดลองโดยการสร้างแบบจำลองด้วยโปรแกรม The Network Simulation: NS2 ทดลองซ้ำอย่างน้อย 20 ครั้งในแต่ละ Scenario และวัดระดับค่าความเชื่อมั่นที่ 95% ในการทดลองนี้พิจารณาจาก Successful Rate และ End to End Delay โดยที่ Successful Rate คือ อัตราการรับ Packet ที่โหนดปลายทางหารด้วยจำนวนของ Packet ที่ส่งออกมาจากโหนดต้นทาง และ End-to-End Delay คือ ช่วงเวลาระหว่างที่ Packet ถูกส่งออกมาจากโหนดต้นทางไปจนถึงระยะเวลาเมื่อตำแหน่งปลายทางได้รับ Packet โดยการทดลองนี้จะทดสอบกับเครือข่ายที่มีโหนดจำนวน 50, 100, 150 และ 200 กระจายอยู่ในเครือข่ายที่มีขนาดใหญ่คือ 1500 x 1200 เมตร และกำหนดช่วงสัญญาณการรับส่งข้อมูล(Transmission Range) ของแต่ละโหนดที่ 100 เมตร ซึ่งการกำหนดค่าพารามิเตอร์ต่างๆ ที่ใช้ในการทดลองจะแสดงไว้ในตารางที่ 3-1

ตารางที่ 3-1 ค่าพารามิเตอร์ที่ใช้ในการทดลอง

Parameter	Value (s)
Number of Nodes	50 , 100 , 150 , 200
Number of CBR source	20
Simulation Time	1,000 Second
Routing Protocol	AODV , DSR , GPSR
NS2 Version	ns 2.28
Transmission Range	100 m
Inter Arrival Time	0.1 Second
Mobility Models	Random Waypoint Model
Topologies	1500m X 1200m
Performance Metrics	Successful Rate, End to End delay

สำหรับการทดลองที่จำนวนโหนดในเครือข่ายที่แตกต่างกันนี้ พบว่าในเครือข่ายจะมีจำนวนโหนดข้างเคียง (Neighbor Node) เท่ากับ 1, 2, 3 และ 4 โหนด ซึ่งแสดงให้เห็นถึงความสัมพันธ์ของปริมาณโหนดในเครือข่ายกับจำนวนโหนดข้างเคียงตามตารางที่3-2 โดยที่ปริมาณโหนดข้างเคียงจะมีผลต่อประสิทธิภาพในการค้นหาเส้นทางสำหรับการส่งข้อมูล

ตารางที่ 3-2 ปริมาณโหนดข้างเคียงที่พบในเครือข่ายที่มีจำนวนโหนดแตกต่างกัน

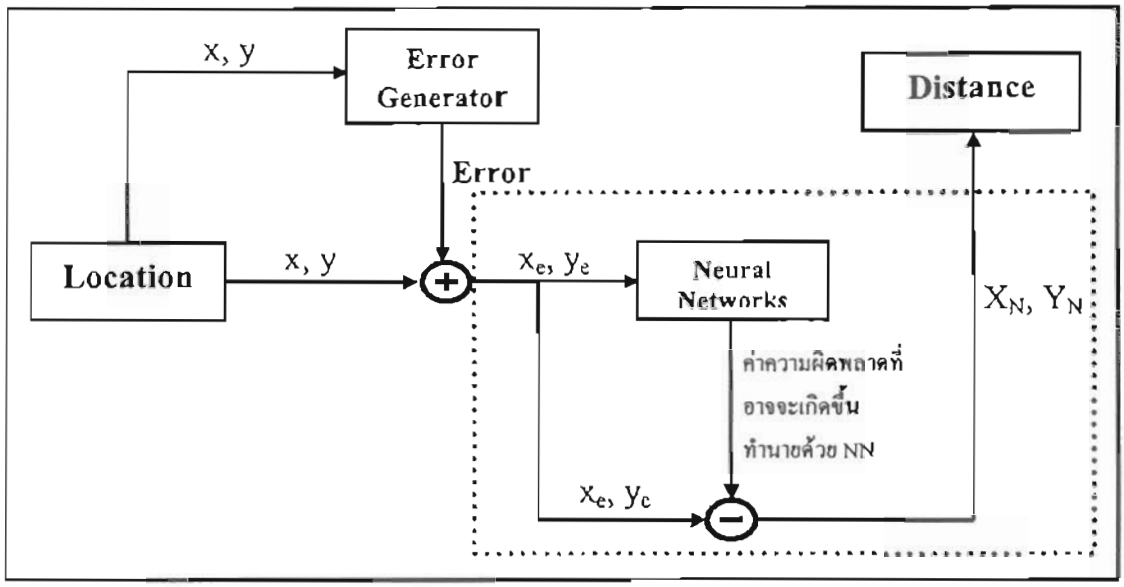
ปริมาณโหนดในเครือข่าย	ปริมาณโหนดข้างเคียงที่พบ
50	0.978
100	1.875
150	2.121
200	3.895

จากตารางที่ 3-2 แสดงให้เห็นว่าปริมาณโหนดข้างเคียงที่พบในเครือข่ายจะเปลี่ยนแปลงตามปริมาณของโหนดที่มีอยู่ในเครือข่าย นั่นคือ ในเครือข่ายที่มีปริมาณโหนดในเครือข่ายมาก ปริมาณโหนดข้างเคียงที่แต่ละโหนดจะพบก็มากตามไปด้วย ในทำนองเดียวกันหากในเครือข่ายมีปริมาณโหนดในเครือข่ายน้อย ปริมาณโหนดข้างที่แต่ละโหนดจะพบก็น้อยลงไปด้วย ซึ่งในตารางที่ 3-2 จะเป็นว่า ในเครือข่ายที่มีจำนวนโหนดน้อยในที่นี้เท่ากับ 50 โหนด แต่ละโหนดมีโอกาสที่จะพบโหนดข้างเคียงเประมาณ 1 โหนด ในขณะที่เครือข่ายมีจำนวนโหนดมาก เท่ากับ 200 โหนด แต่ละโหนดก็จะมีโอกาสที่จะพบโหนดข้างเคียงเท่ากับ 4 โหนด

3.4 ขั้นตอนการแก้ปัญหาเมื่อเกิด Location Information error ด้วยข่ายงานระบบประสาท (Neural Networks)

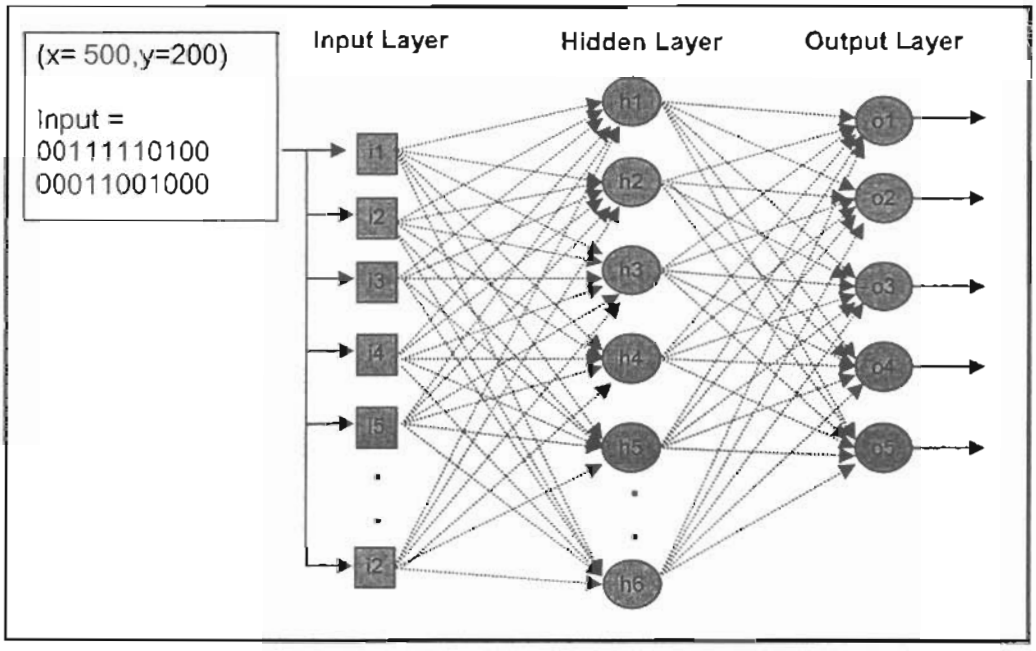
3.4.1 กรอบความคิดในการใช้ Neural Network

การแก้ปัญหาเมื่อเกิดความผิดพลาดของตำแหน่ง (Location Information Error) ด้วยข่ายงานระบบประสาท (Neural Network) โดยพิจารณาจากการคำนวณหาระยะทางของแต่ละโหนด ซึ่งในการสร้างแบบจำลองด้วยโปรแกรม The Network Simulator แต่ละโหนดจะถูกกำหนดตำแหน่งที่จะโปรยไปในเครือข่ายโดยการสุ่มค่าตำแหน่งแบบ Random Waypoint Mobility Model ซึ่งจากภาพที่ 3-18 ค่า Location คือ ตำแหน่งของแต่ละโหนดซึ่งจะแทนด้วยค่า x, y ซึ่งในการทดลองนี้ผู้วิจัยต้องการให้แต่ละตำแหน่งเกิดความผิดพลาดในการบอกตำแหน่งของตัวเอง จึงกำหนดให้แต่ละตำแหน่งจะถูกบวกเพิ่มด้วยค่าความน่าจะเป็นในการเกิดความผิดพลาดที่ระยะต่างๆ ให้กับตำแหน่ง x, y ข้างต้น ทำให้ค่าตำแหน่งมีค่าความผิดพลาดเกิดขึ้น และแทนค่าด้วย x_e, y_e จากนั้นนำข่ายงานระบบประสาท (Neural Network) มาทำนายเพื่อหาค่า Error Value ที่เกิดขึ้นจากค่าตำแหน่งผิดพลาดที่ส่งเข้ามา และนำค่า Error Value ที่ได้หักออกจากค่าตำแหน่ง เพื่อทำให้ค่าตำแหน่งมีความผิดพลาดลดลงก่อนที่จะส่งเข้าสู่การคำนวณระยะทางในส่วนของ Distance ซึ่งจากกรอบความคิดที่กล่าวมานั้นสามารถเขียนเป็นขั้นตอนได้ ตามภาพที่ 3-18



ภาพที่ 3-18 ขั้นตอนการแก้ปัญหาเมื่อเกิด Location Information Error ด้วยข่ายงานระบบประสาท

3.4.2 การประยุกต์ใช้ข่ายงานระบบประสาท (Neural Network) เพื่อทำนายค่าความผิดพลาด

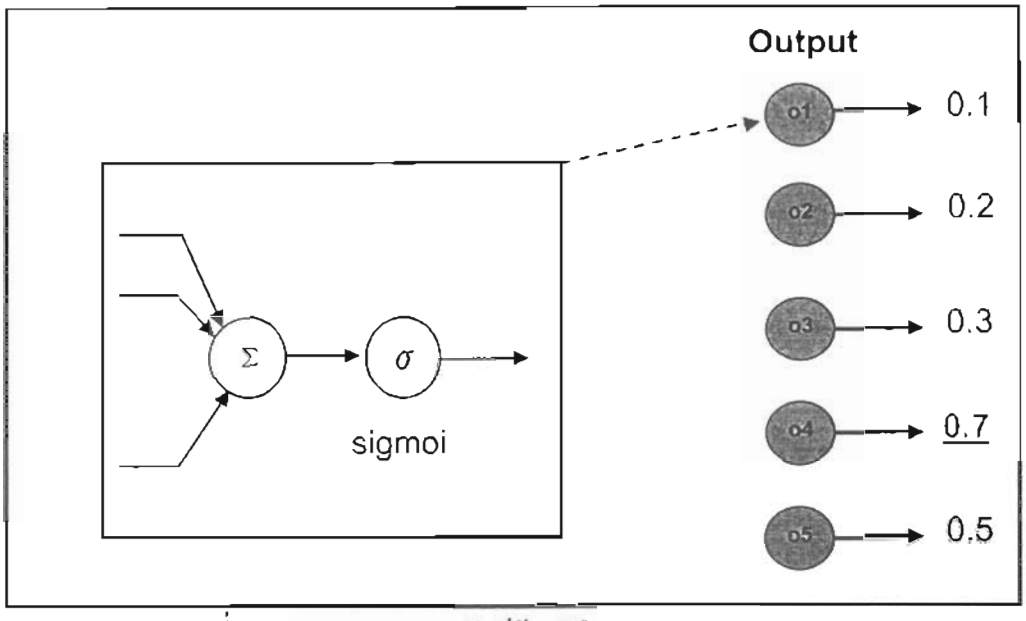


ภาพที่ 3-19 โครงสร้างของข่ายงานระบบประสาทที่ใช้ในการทดลอง

ค่าตำแหน่งของโหนด จะได้จากการสุ่มด้วย Random Waypoint Mobility Model มี 2 ค่า คือ x และ y แต่ค่าตำแหน่งที่ได้มานั้น ต่างไม่มีความสัมพันธ์หรือความขึ้นต่อกัน (Independent) เพราะฉะนั้นผู้วิจัยจึงนำค่าตำแหน่งดังกล่าวมาแปลงให้อยู่ในรูปแบบของเลขฐานสอง (Binary Code) เพื่อทำให้เกิดความสัมพันธ์ภายในค่าแต่ละค่า ดังนั้น จากเดิมที่มีข้อมูลเข้า 2 ค่า คือ x และ y ตามพื้นที่ที่ใช้ทดลองคือ 1500 X 1200 จึงเปลี่ยนเป็น 22 ค่า โดยกำหนดให้จากข้อมูลเข้าเดิมแต่ละค่า (x, y) ใช้เลขฐานสองจำนวน 11 บิต เนื่องจากค่าสูงสุดของตำแหน่ง x คือ 1500 สามารถแปลงให้อยู่ในรูปของเลขฐานสอง คือ 10111011100 และตำแหน่ง y คือ 1200 สามารถแปลงให้อยู่ในรูปของเลขฐานสอง คือ 10010110000 นั่นคือ เลขฐานสองที่ได้ใช้จำนวนไม่เกิน 11 หลัก ทำให้ได้ข้อมูลเข้าจำนวน 22 ค่า สำหรับข่ายงานระบบประสาท ตามที่แสดงให้เห็นในภาพที่ 3-19 เพื่อนำเข้าสู่ Hidden Layer สำหรับคำนวณหาผลลัพธ์ที่เหมาะสม เนื่องจากข้อมูลออกแต่ละชุดจะสัมพันธ์กับการทำนายค่าความผิดพลาดของตำแหน่งที่แตกต่างกัน ดังแสดงในตารางที่ 3-3 โดยพื้นที่ที่ผู้วิจัยแทนค่าพื้นที่ด้วย 0 คือ พื้นที่ที่เกิดความผิดพลาดของตำแหน่งไปด้านลบ และแทนค่าพื้นที่ด้วย 1 คือ พื้นที่ที่เกิดความผิดพลาดของตำแหน่งไปด้านบวก

ตารางที่ 3-3 การกำหนดค่าความผิดพลาดให้กับ Output ของข่ายงานระบบประสาท

Output : โหนดคำตอบ	**การกำหนดค่าความผิดพลาด**	
	พื้นที่ที่เกิดความผิดพลาดของตำแหน่งไปด้านลบ (0)	พื้นที่ที่เกิดความผิดพลาดของตำแหน่งไปด้านลบ (1)
O1	0 เมตร	0เมตร
O2	-5 เมตร	5 เมตร
O3	-10 เมตร	10 เมตร
O4	-15 เมตร	15 เมตร
O5	-20 เมตร	20 เมตร



ภาพที่ 3-20 การคำนวณผลลัพธ์ด้วยวิธีการ Winner Take All

ข่ายงานระบบประสาทที่ใช้ กำหนดให้มีข้อมูลออกจำนวน 5 โหนด โดยใช้เทคนิคของ Winner Take All ในการเลือกโหนดคำตอบจากโหนดที่มีค่ามากที่สุด ดังแสดงในภาพที่ 3-20 ซึ่งแสดงให้เห็นว่าระบบจะเลือกค่าลำดับที่ 4 เป็นคำตอบของระบบ ดังนั้น ค่าผิดพลาดที่ระบบทำนายได้แก่ ±15 ซึ่งจะเป็นค่าบวกหรือลบจะขึ้นอยู่กับค่าตำแหน่งของโหนดที่ได้จากการทำ Random Waypoint ในแต่ละครั้ง โดยพิจารณาจาก

- หากค่าตำแหน่งที่ Random มาได้ มีค่า y อยู่ในช่วง 0-600 ค่าความผิดพลาดจะมีค่าลบ
- หากค่าตำแหน่งที่ Random มาได้ มีค่า y อยู่ในช่วง 601-1500 ค่าความผิดพลาดจะมีค่าบวก

3.5 การทดลองข่ายงานระบบประสาท

ในส่วนนี้กล่าวถึงการประยุกต์ใช้ข่ายงานระบบประสาท ในการจำแนกหรือเลือกโหนดที่เหมาะสมสำหรับการส่งต่อข้อมูล ข่ายงานระบบประสาทนั้นมีรูปแบบและสถาปัตยกรรม โดยข่ายงานระบบประสาทที่ได้รับความนิยมและหลายใช้งานกันอย่างแพร่หลายคือ ข่ายงานแบบป้อนกลับไปข้างหน้าหลายชั้น (Multi-Layer Perceptron) และเทคนิคการปรับค่าน้ำหนักการเชื่อมโยงหรือการฝึกสอนแบบป้อนกลับ (Back-Propagation Training) การฝึกสอนข่ายงานระบบประสาทจะใช้วิธีทำซ้ำ การนำชุดข้อมูลฝึกสอนป้อนเข้าสู่กระบวนการฝึกสอนครั้งละ 1 รายการตามลำดับ และทำการปรับค่าน้ำหนักเชื่อมโยง ซึ่งจำนวนรอบของการทำซ้ำนั้นขึ้นอยู่กับการพิจารณาจากค่า Mean Square Error(MSE) หรือการลู่เข้าคำตอบที่ให้ค่าผิดพลาดน้อยจนเป็นที่พอใจ

3.5.1 โครงสร้างของข่ายงานระบบประสาท

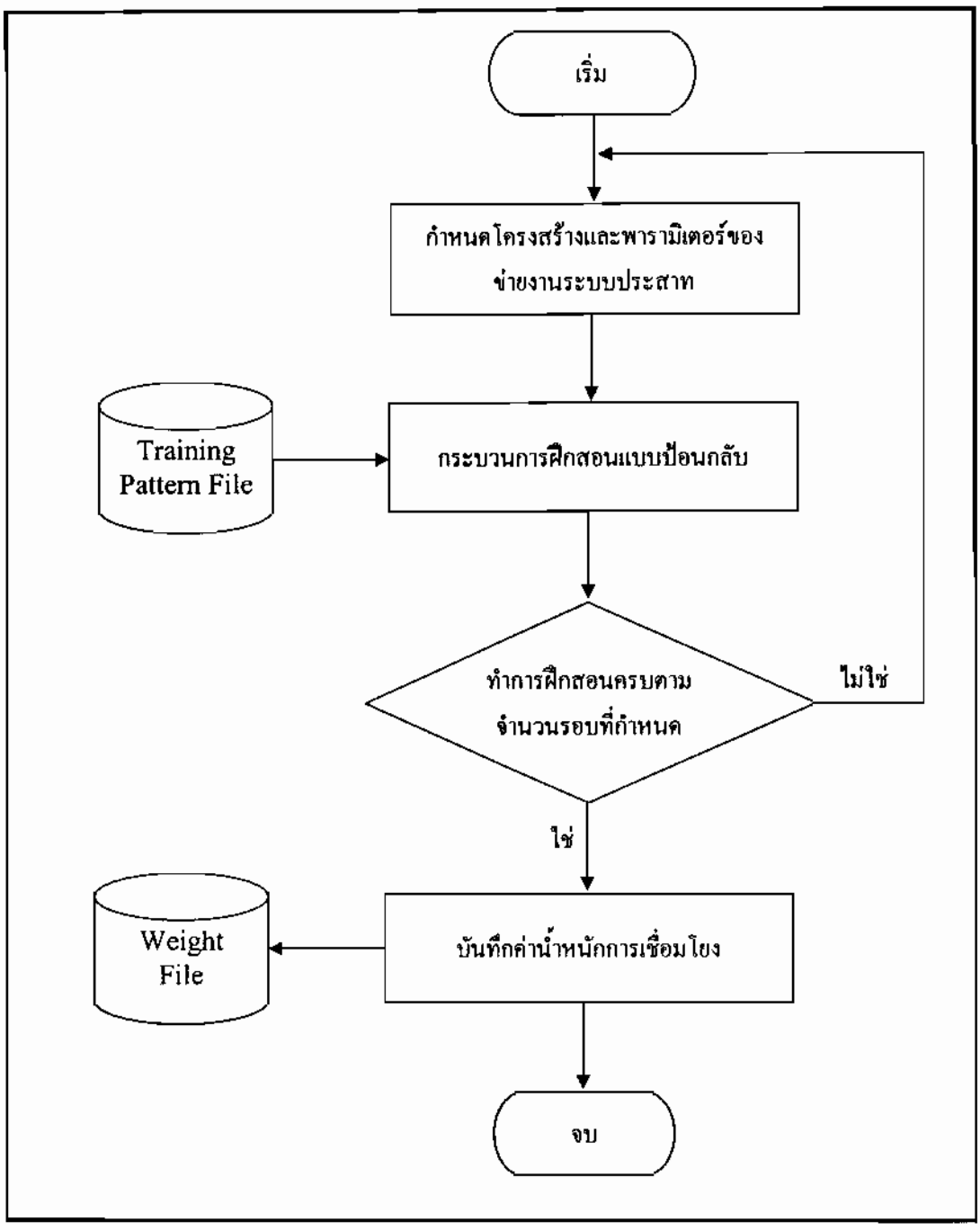
ตารางที่ 3-4 โครงสร้างของข่ายงานระบบประสาท

โครงสร้างของข่ายงานระบบประสาท	**ค่า**
จำนวนข้อมูลนำเข้า	22
จำนวนข้อมูลนำออก	5
จำนวนชั้นซ่อนของข่ายงาน	1
จำนวนหน่วยประมวลผลในชั้นซ่อน	45
ฟังก์ชันกระตุ้น	ฟังก์ชันเชิงตรรกะ (Sigmoid)

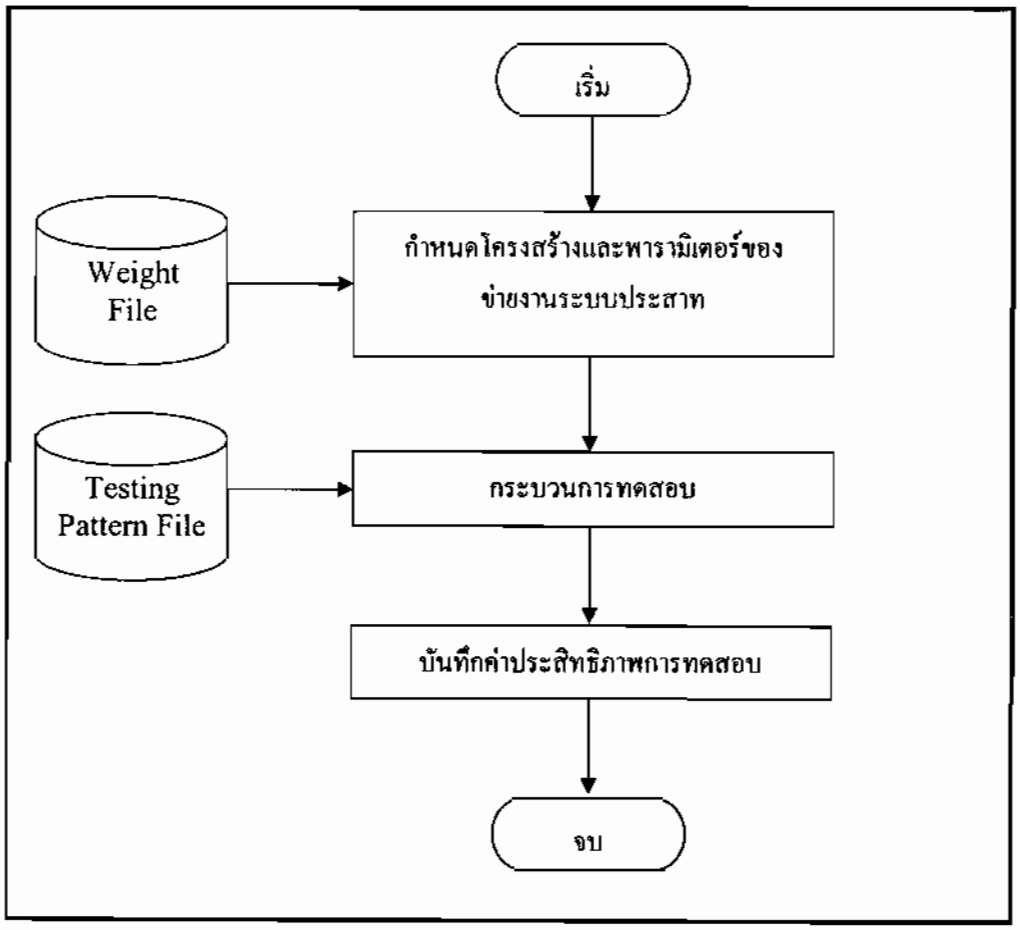
ตารางที่ 3-5 พารามิเตอร์สำหรับข่ายงานระบบประสาท

พารามิเตอร์ของข่ายงานระบบประสาท	ค่า
ค่าอัตราการเรียนรู้	0.015
จำนวนรอบการสอน	500,000 รอบ และ 1,000,000 รอบ

การทดลองกับข่ายงานระบบประสาทใช้แฟ้มข้อมูลสำหรับฝึกสอน (Training Patterns File) ที่ทำการแบ่งจากชุดข้อมูลที่รวบรวมได้ทั้งหมดจากแฟ้มชุดข้อมูล หลังจากทำการฝึกสอนแล้วทดสอบด้วยชุดข้อมูลที่รวบรวมได้ทั้งหมดจากแฟ้มชุดข้อมูลทดสอบ (Testing Pattern File) โดยแผนภาพแสดงขั้นตอนการฝึกสอนและทดสอบข่ายงานระบบประสาทแสดงในภาพที่ 3-19 และ ภาพที่ 3-20 ตามลำดับ



ภาพที่ 3-21 แผนผังขั้นตอนการฝึกสอนข่ายงานระบบประสาท



ภาพที่ 3-22 แผนผังขั้นตอนการทดสอบข่ายงานระบบประสาท