บทที่ 3
การออกแบบแพลตฟอร์ม

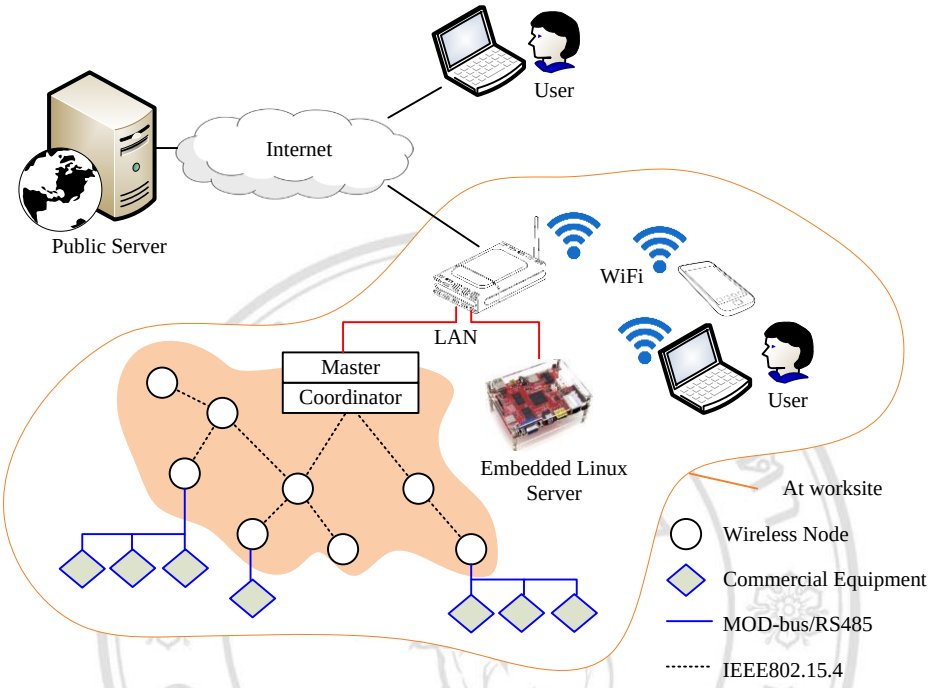
เนื้อหาในบทนี้เกี่ยวกับการออกแบบแพลตฟอร์ม “เครือข่ายเซนเซอร์ที่สามารถโปรแกรมซ้ำได้สำหรับมอนิเตอร์ระบบพลังงานทดแทน” แต่เพื่อความสะดวกในการอ้างถึงจะใช้คำว่า “แพลตฟอร์ม”, แพลตฟอร์มที่นำเสนอในงานวิจัยนี้ ครอบคลุมตั้งแต่การสร้างฮาร์ดแวร์ของอุปกรณ์ต่างๆ มาจนถึงชั้นโพรโทคอลสำหรับเครือข่ายเซนเซอร์ไร้สายที่อิงมาตรฐาน IEEE 802.15.4 รวมถึงโพรโทคอลที่ใช้ในการส่งข้อมูลต่อเข้าไปยังเครือข่ายที่ใช้งานกันทั่วไป เช่น LAN/WiFi และอินเทอร์เน็ต เป็นต้น, ส่วนสุดท้ายคือการพัฒนาเฟิร์มแวร์ที่โปรแกรมลงในโหนด และการโปรแกรมแอพพลิเคชันซ้ำลงไปหลังติดตั้ง, การโปรแกรมซ้ำเป็นส่วนที่เกี่ยวข้องกับแอพพลิเคชันซึ่งจะกล่าวถึงในบทที่ 4

3.1 โครงสร้างของแพลตฟอร์ม

โครงสร้างของแพลตฟอร์มในภาพที่ 3.1 แบ่งออกเป็น 2 ส่วน ส่วนแรกคือ เครือข่ายเซนเซอร์ไร้สายที่ใช้โพรโทคอลแบบโพรไพรทารี (Proprietary) เรียกได้อีกอย่างหนึ่งว่า PAN (Personal Area Network) และ ส่วนที่สองเป็นเครือข่ายมาตรฐาน LAN/WiFi รวมถึงอินเทอร์เน็ตด้วย เครือข่ายเซนเซอร์ไร้สายประกอบด้วย “โหนด” หลายตัว และ “โคออร์ดิเนเตอร์” จำนวน 1 ตัว โคออร์ดิเนเตอร์ทำหน้าที่ควบคุมและจัดการเครือข่าย รวมทั้งรวบรวมข้อมูลจากโหนดต่างๆ ไว้รอส่งต่อให้ “มาสเตอร์” (Master), โคออร์ดิเนเตอร์และมาสเตอร์อยู่ที่จุดเดียวกัน สามารถสื่อสารกันได้โดยตรง เมื่อมาสเตอร์ได้รับข้อมูลแล้วจะนำส่งต่อไปยังเซิร์ฟเวอร์ผ่าน LAN/WiFi และบางครั้งต้องผ่านอินเทอร์เน็ต การสื่อสารระหว่างมาสเตอร์กับเซิร์ฟเวอร์จะอิงอยู่บนโพรโทคอล HTTP (Hypertext Transfer Protocol) ซึ่งเป็นโพรโทคอลที่ใช้งานกันมากบนอินเทอร์เน็ต

หน้าที่หลักของโหนดคือ การเสาะหาข้อมูล (Data acquisition) โหนดสามารถอ่านข้อมูลได้จากแหล่งข้อมูลหลายประเภท แหล่งข้อมูลประเภทที่ใช้งานบ่อยคือ “เซนเซอร์” มันจะเปลี่ยนปริมาณทางกายภาพ (Physical quantity) ที่ต้องการวัดให้เป็นสัญญาณไฟฟ้า ซึ่งปกติจะอยู่ในรูปแรงดัน โหนดจะแปลงแรงดันและตีความให้เป็นข้อมูลเก็บไว้ก่อน, โหนดหนึ่งโหนดสามารถวัดข้อมูลได้หลายอย่างโดยใช้เซนเซอร์หลายตัว นอกจากนี้โหนดยังสามารถใช้ RS485 เพื่อสื่อสารกับอุปกรณ์เชิงพาณิชย์ ที่

อนุญาตให้สื่อสารข้อมูลด้วย MOD-bus ทำให้โหนดสามารถใช้อุปกรณ์เชิงพาณิชย์จากหลากหลายผู้ผลิตเป็นแหล่งข้อมูลของมันได้



ภาพที่ 3.1 โครงสร้างของแพลตฟอร์ม

ภายในหนึ่งเครือข่ายเซนเซอร์ไร้สายจะมีโคออร์ดิเนเตอร์ 1 ตัว ทำหน้าที่จัดการเครือข่าย ตัวอย่างเช่น การรับโหนดเข้าร่วมเครือข่าย, การสั่งให้สร้างเครือข่ายใหม่, การโปรแกรมแอพพลิเคชันให้โหนดในเครือข่าย, การตั้งนาฬิกาและปฏิทินให้กับไอซี RTCC (Real Time Clock and Calendar) บนโหนด, และการตั้งพารามิเตอร์ต่างๆ เป็นต้น เมื่อจัดการสร้างเครือข่ายและทุกโหนดอยู่ในสถานะพร้อมทำงานแล้ว ข้อมูลจากแต่ละโหนดจะถูกส่งต่อกันมารวมไว้ที่โคออร์ดิเนเตอร์ ดังนั้นโคออร์ดิเนเตอร์จึงมีหน้าที่พักข้อมูลเหล่านั้นไว้ที่บัฟเฟอร์ของมันด้วย ก่อนที่มาสเตอร์จะร้องขอข้อมูลออกไป นอกจากนี้ โคออร์ดิเนเตอร์ยังต้องร้องขอ “โปรไฟล์” (Profile) จากโหนดที่เป็นต้นกำเนิดของข้อมูลนั้น เพื่อใช้ตีความข้อมูลที่มันได้รับ โคออร์ดิเนเตอร์ตีความข้อมูลเพื่อแสดงผลและสถานะเครือข่าย ณ หน้าจอของมาสเตอร์ ผู้ใช้จึงสามารถทราบถึงสถานะการทำงานของแพลตฟอร์มในเบื้องต้นได้จากหน้าจอของมาสเตอร์

มาสเตอร์เป็นส่วนที่ต่อตรงกับโคออร์ดิเนเตอร์ มันจึงสามารถดึงข้อมูลและโปรไฟล์ที่พักไว้ในโคออร์ดิเนเตอร์มาเพื่อเตรียมส่งต่อไปให้เซิร์ฟเวอร์ และ นำข้อมูลบางส่วนมาแสดงผลบนหน้าจอได้สะดวก, การติดต่อกับเซิร์ฟเวอร์จะต้องผ่าน LAN จึงต้องใช้โพรโทคอลมาตรฐาน อย่างเช่น HTTP ซึ่งอยู่บนโพรโทคอล TCP/IP ในรายงานนี้จะไม่กล่าวถึงรายละเอียดของโพรโทคอลในฝั่ง LAN เพราะ

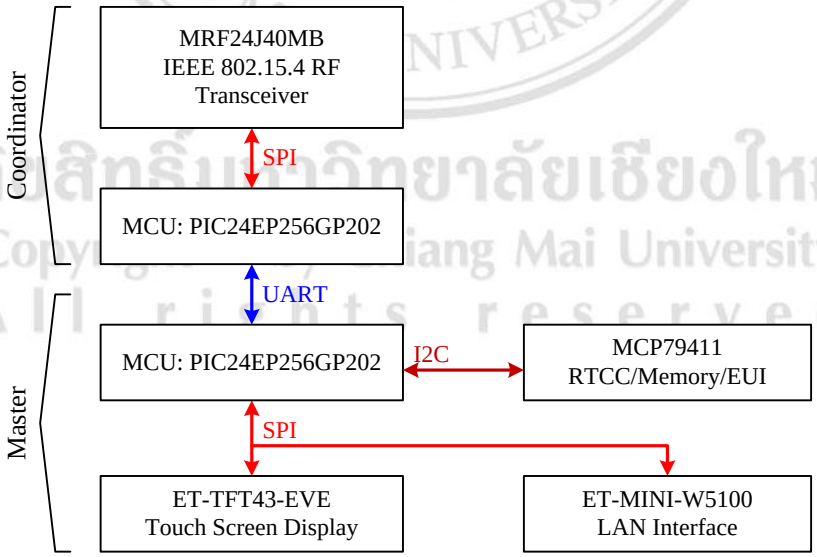
ไม่มีการออกแบบสิ่งใหม่เป็นเพียงการนำ อุปกรณ์/โมดูล/ไอซี มาใช้งานเพื่อจัดการโพรโทคอลเหล่านี้ แต่จะกล่าวถึงรายละเอียดของรูปแบบข้อมูลที่ฝากไปบน HTTP เท่านั้น

เซิร์ฟเวอร์ที่ใช้งานในแพลตฟอร์มนี้ จะต้องทำงานเป็น Web server ได้ ซึ่งอาจจะเป็นเซิร์ฟเวอร์ทั่วไปที่เชื่อมต่ออยู่กับอินเทอร์เนต (Public Server) หรือเป็น คอมพิวเตอร์, คอมพิวเตอร์ขนาดเล็ก (Embedded Linux Server) ที่ลงซอฟต์แวร์ Web server และติดตั้งอยู่ในวง LAN เดียวกันกับมาสเตอร์ก็ได้, ไม่ว่าจะเป็นเซิร์ฟเวอร์แบบใด มาสเตอร์จะติดต่อกับเซิร์ฟเวอร์ด้วยวิธีการเดียวกัน

ในกรณีที่ใช้งานเซิร์ฟเวอร์ที่อยู่ในวง LAN เดียวกัน ผู้ใช้สามารถใช้คอมพิวเตอร์ หรืออุปกรณ์ไอทีทั่วไป เชื่อมต่อเข้าวง LAN โดยผ่านเราเตอร์ (Router) ที่มี WiFi เพื่อมอนิเตอร์ข้อมูลจากเซิร์เวอร์ได้ โครงสร้างในลักษณะนี้เหมาะสำหรับการมอนิเตอร์ในไซต์งานที่ต้องการความเบ็ดเสร็จในตัว

3.1.1 ฮาร์ดแวร์ของโคออร์ดิเนเตอร์และมาสเตอร์

โคออร์ดิเนเตอร์และมาสเตอร์ถูกสร้างไว้บนบอร์ดเดียวกันดังโครงสร้างในภาพที่ 3.2 และภาพที่ 3.3 ส่วนของโคออร์ดิเนเตอร์ใช้ไมโครคอนโทรลเลอร์ขนาด 16 บิต ไอซีเบอร์ PIC24EP256GP202 เป็นตัวควบคุมหลัก มีบริวารเพียงตัวเดียวคือ ตัวรับส่งวิทยุ โมดูล MRF24J40MB ซึ่งรับ/ส่งสัญญาณวิทยุตามมาตรฐาน IEEE 802.15.4 ไมโครคอนโทรลเลอร์จะติดต่อกับตัวรับส่งวิทยุด้วย SPI และติดต่อกับมาสเตอร์ด้วย UART



ภาพที่ 3.2 ผังโครงสร้างของโคออร์ดิเนเตอร์และมาสเตอร์

ภาพที่ 3.3 ภาพถ่ายของโคออร์ดิเนเตอร์และมาสเตอร์

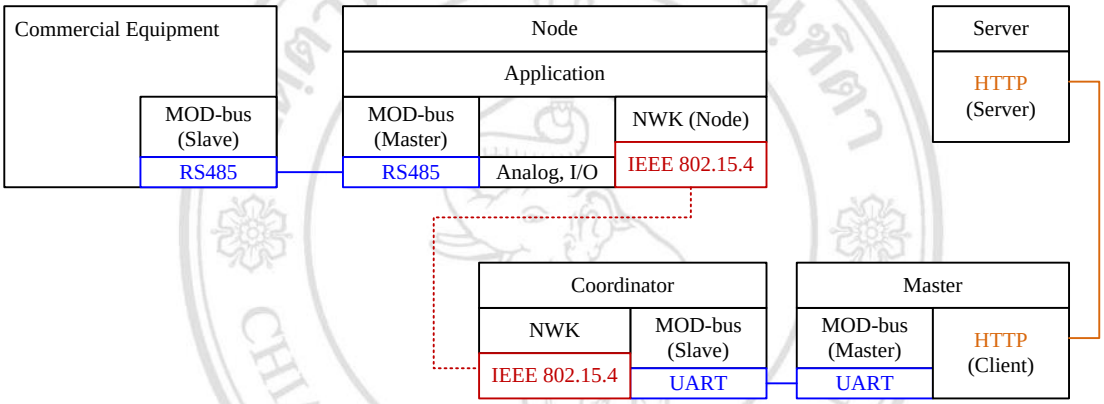
ส่วนของมาสเตอร์ก็ใช้ไอซีเบอร์ PIC24EP256GP202 เป็นตัวควบคุมหลักเช่นเดียวกัน แต่มีอุปกรณ์บริวาร 3 ตัว ได้แก่ โมดูล ET-W5100, ET-TFT43-EVE และ ไอซี MCP79411, สองโมดูลแรกใช้ SPI สำหรับการอินเตอร์เฟส ส่วนไอซี MCP79411 อินเตอร์เฟสด้วย I^2C, อุปกรณ์บริวารเหล่านี้มีหน้าที่ช่วยไมโครคอนโทรลเลอร์ประมวลผลในแต่ละด้านดังต่อไปนี้ โมดูล ET-W5100 ทำหน้าที่จัดการโพรโทคอลเพื่อการสื่อสารผ่าน LAN, โมดูล ET-TFT43-EVE ทำหน้าที่ติดต่อกับผู้ใช้ ผ่านหน้าจอ LCD กราฟฟิคความละเอียด 480x270 จุด และ Touch screen แบบ Resistive, และสุดท้ายไอซี MCP79411 ใช้เป็นนาฬิกาและปฏิทินเพื่อรักษาเวลาในแพลตฟอร์ม สำหรับ “ประทับเวลา” (Time stamp) ให้กับเรคอร์ดข้อมูล

3.2 ชั้นโพรโทคอลภายในแพลตฟอร์ม

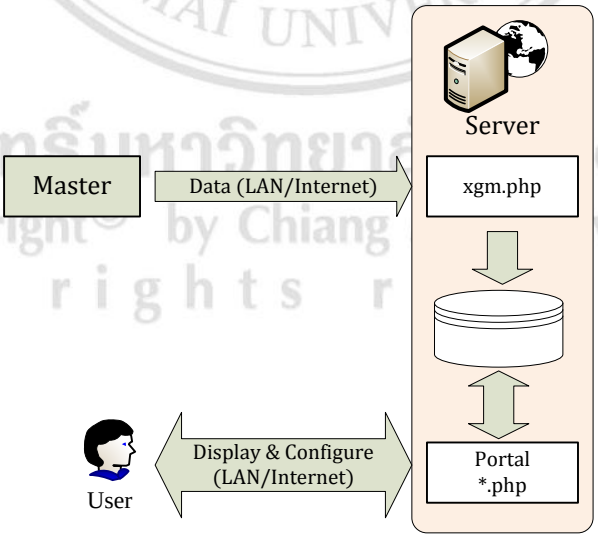
ผังในภาพที่ 3.4 แสดงชั้นโพรโทคอลที่ใช้งานในส่วนต่างๆ ของแพลตฟอร์ม ส่วนแรกการสื่อสารระหว่างโหนด กับอุปกรณ์เชิงพาณิชย์ใช้ RS485 เป็นชั้นกายภาพ และ MOD-bus ใช้เป็นดาต้าลิงค์ โหนดซึ่งทำหน้าที่อ่านข้อมูลจากอุปกรณ์ รับบทเป็นมาสเตอร์ของบัส อุปกรณ์เชิงพาณิชย์ถูกเซตให้ทำหน้าที่เป็นสเลฟของบัสมาจากโรงงานอยู่แล้ว โหนด 1 ตัว สามารถเชื่อมต่อกับอุปกรณ์ได้มากกว่า 1 ตัว แต่ต้องมี Address ไม่ซ้ำกัน, ส่วนที่สองเป็นการสื่อสารระหว่างโหนดกับโคออร์ดิเนเตอร์ ชั้นโพรโทคอลแบ่งออกเป็น 3 ชั้น ใน 2 ชั้นล่าง PHY และ MAC เป็นไปตามมาตรฐาน IEEE 802.15.4 ส่วนชั้นที่ 3 เป็นชั้นเครือข่าย (Network) ซึ่งถูกพัฒนาในงานวิจัยนี้ คุณสมบัติของชั้นเครือข่ายรองรับโทโพโลยีแบบทรี และสามารถสร้างเครือข่าย, เข้าร่วมเครือข่าย รวมทั้งซ่อมแซมได้อัตโนมัติ ออกแบบมาใช้สำหรับรวบรวมข้อมูลจากโหนดต่างๆ ไปยังโคออร์ดิเนเตอร์ และส่งสนับสนุนการ รหัสจากโคออร์ดิเนเตอร์มาโปรแกรมแอพพลิเคชัน ให้โหนด, เมื่อ

พิจารณาผังในภาพที่ 3.4 โคออร์ดิเนเตอร์เป็นเสมือนสะพานของข้อมูลจากฝั่ง PAN ไปหามาสเตอร์เพื่อเข้าสู่ฝั่ง LAN, การสื่อสารระหว่างมาสเตอร์กับโคออร์ดิเนเตอร์ จะใช้ UART ในชั้นกายภาพ และใช้ MOD-bus เป็นดาต้าลิงค์เช่นเดียวกับส่วนแรก, การสื่อสารส่วนสุดท้ายคือระหว่างมาสเตอร์กับเซิร์ฟเวอร์จะอิงอยู่บนมาตรฐาน HTTP โดยมาสเตอร์จะทำตัวเป็นเวปไคลเอนต์

ปริมาณงานส่วนใหญ่ของงานวิจัยนี้เป็นการพัฒนาโพรโทคอลบนเครือข่าย IEEE 802.15.4 รายละเอียดการทำงานของเครือข่ายไร้สายจะกล่าวถึงในหัวข้อ 3.3 ส่วนโพรโทคอลอีกสองส่วนเป็นการนำ MOD-bus และ HTTP ที่รู้จักกันดีอยู่แล้วเรานำมาประยุกต์ใช้งานเพื่อขยายขอบเขตการวัดข้อมูล และ ช่วยนำข้อมูลขึ้นสู่เซิร์ฟเวอร์ ตามลำดับ จึงละเว้นการกล่าวถึงรายละเอียด



ภาพที่ 3.4 ผังชั้นโพรโทคอลสำหรับการสื่อสารข้อมูลในแพลตฟอร์ม



ภาพที่ 3.5 สคริพต์ PHP บนฝั่งเซิร์ฟเวอร์

3.2.1 การส่งข้อมูลระหว่างมาสเตอร์กับเซิร์ฟเวอร์

การสื่อสารระหว่างมาสเตอร์กับเซิร์ฟเวอร์ มีจุดประสงค์หลักคือ นำข้อมูลเข้าไปเก็บในฐานข้อมูลบนเซิร์ฟเวอร์ และร้องขอเวลาจากเซิร์ฟเวอร์มาใช้เป็นเวลามาตรฐานในแพลตฟอร์ม ในที่นี้เลือกใช้ HTTP เพราะเป็นโพรโทคอลที่ง่ายแต่มีเสถียรภาพสูงในภาพที่ 3.5 ใช้สคริพต์ PHP ทำงานที่ฝั่งเซิร์ฟเวอร์ รอรับคำร้องขอจากไคลเอนต์ โดยใช้ GET method เพื่อส่งข้อมูล ตามรูปแบบข้างล่าง (สคริพต์ชื่อxgm.php)

```
xgm.php?c=9&g=2800&d=33402445160408151 2FB0000|2140244516040
815123B0000|
xgm.php?c=1&g=2800
```

c คือหมายเลขคำสั่ง, 1 ร้องขอเวลา และ 9 ส่งข้อมูล

g คือหมายเลขไซต์งาน, เป็นเลขฐานสิบหกจำนวน 4 หลัก

d คือข้อมูล, แยกแต่ละเรคอร์ดด้วย '|' ภายในแต่ละเรคอร์ดแบ่งออกเป็นสี่ส่วนดังนี้

สีเขียว หมายเลขโหนด, เลขฐานสิบหกจำนวน 4 หลัก

สีแดงเข้ม หมายเลขประเภทข้อมูล (เอ็นพอยนต์), เป็นเลขฐานสิบหกจำนวน 2 หลัก

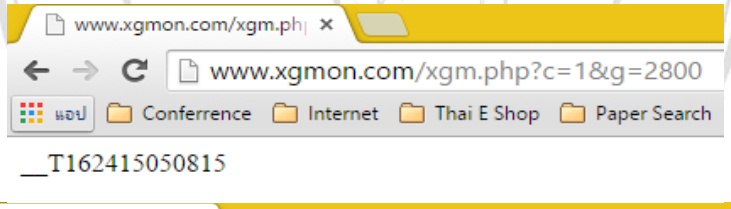
สีฟ้า เวลาประทับ, เป็น BCD เรียงจากหน่วย นาที จนถึง ปี ค.ศ.

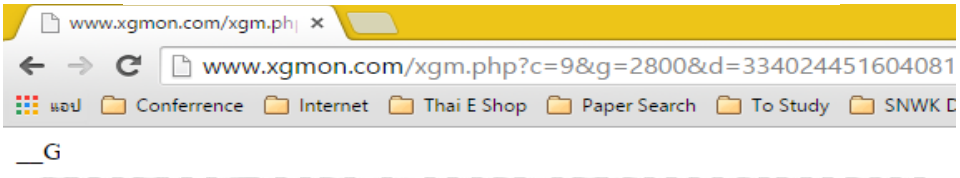
สีส้ม ข้อมูล, ตัวอย่างข้อมูลขนาด 32 บิต

ภาพที่ 3.5 แสดงให้เห็นว่าบนฝั่งเซิร์ฟเวอร์มีสคริพต์ PHP อยู่สองส่วน ส่วนแรก xgm.php เป็นสคริพต์อย่างง่ายใช้เพื่อรับข้อมูลจากมาสเตอร์ มีรูปแบบตามที่กล่าวมา ซอร์สโค้ดของ xgm.php แสดงในภาพที่ 3.6, xgm.php ทำหน้าที่เพียงนำข้อมูลดิบที่ไม่มีการประมวลผลใดๆ ลงเก็บในฐานข้อมูล, สคริพต์ส่วนที่สอง ทำหน้าที่ประมวลผลข้อมูลดิบ แล้วแยกแยะหมวดหมู่ (Web Portal) เพื่อแสดงผลให้กับผู้ใช้ สคริพต์ในส่วน Web Portal นี้ จะถูกพัฒนาขึ้นมาเฉพาะสำหรับความต้องการของผู้ใช้แต่ละคน และไม่ได้อยู่ในขอบเขตของแพลตฟอร์มนี้

```php
<?php
    $cmd = $_GET['c'];
    $site = $_GET['g'];
    if (!isset($cmd)) {     echo "__X";     die;  }
    if (!isset($site)) {    echo "__X";     die;  }
    if ($cmd == 1) { //get time
            $tm = date('siHdmY');
            $rt = "__T".substr($tm,0,10).substr($tm,12,2);
            echo "$rt"; die;
    }
    if ($cmd != 9) die; //Insert data record (cmd = 9)
    $drec = $_GET['d'];
    if (!isset($drec)) {    echo "__X"; die;    }
    $host = "localhost"; $user = "[redacted]"; $password ="
    $link = mysql_connect($host,$user,$password);
    mysql_query("USE sensorne_d4m;");
    $str = "INSERT INTO updata (rdata) VALUES ('$drec');";
    $u = @mysql_query($str);
    if ($u)
        echo "__G";
    else
        echo "__X";
?>
```

ภาพที่ 3.6 สคริพต์ xgm.php





ภาพที่ 3.7 ตัวอย่างผลการเรียกสคริพต์ xgm.php โดยใช้ Web browser

ภาพที่ 3.7 แสดงตัวอย่างการทดสอบสคริพต์ xgm.php โดยการนำไปไว้บนเซิร์ฟเวอร์ชื่อ www.xgmon.com และเรียกใช้โดยโปรแกรม Google Chrome, ภาพด้านบนเป็นการร้องขอเวลา ด้วย c=1 สคริพต์จะตอบกลับด้วย “__T” แล้วตามด้วยเวลาในรูปแบบ BCD เรียงตั้งแต่ วินาที ไปจนถึง ปี ค.ศ. ซึ่งการร้องขอเวลาด้วยวิธีนี้ กว่ามาสเตอร์จะนำเวลาที่ได้รับไปตั้งนาฬิกาของตัวเองได้เวลาอาจผ่านไปแล้วหลายวินาที เนื่องจากเวลาหน่วงของการสื่อสาร แต่ว่าความคลาดเคลื่อนในระดับวินาทีไม่กระทบต่อแอพพลิเคชันที่มอนิเตอร์ปริมาณต่างๆ ด้วยคาบการสุ่มในระดับ 5 นาทีขึ้นไป, ภาพที่ 3.7 ล่างเป็นการส่งข้อมูลเข้าเก็บในฐานข้อมูล หากการจัดเก็บข้อมูลเรียบร้อยสคริพต์จะตอบกลับด้วย

“__G” ในกรณีที่มาสเตอร์ไม่ได้รับข้อความ “__G” ภายในเวลาที่กำหนด มันจะส่งเรคอร์ดเดิมขึ้นไปให้เซิร์ฟเวอร์อีกครั้ง

3.2.2 โพรโทคอลระหว่างมาสเตอร์กับโคออร์ดิเนเตอร์

ถึงแม้ว่ามาสเตอร์กับโคออร์ดิเนเตอร์จะสื่อสารกันด้วยพอร์ต UART แต่เราก็จัดรูปแบบเฟรมตาม MOD-bus โดยให้โคออร์ดิเนเตอร์เป็นสเลฟ, รูปแบบเฟรมตาม MOD-bus โดยเลือกใช้ Function Code ในช่วง “User Defined Function codes” ที่กำหนดไว้ใน [16] ส่วนข้อมูล Payload ขนาดขึ้นอยู่กับ Function Code ของเฟรมนั้นๆ ดังแสดงในตารางที่ 3.1 แสดง Function Code ที่ใช้สื่อสารระหว่างมาสเตอร์กับโคออร์ดิเนเตอร์ แบ่งออกได้เป็น 6 กลุ่ม ตามหน้าที่, กลุ่มแรกคือ General settings ประกอบด้วย PIndex และ MRF

- Pindex มาสเตอร์ใช้เพื่อสอบถามหมายเลข (SN: serial numner) และอินเด็กซ์ (Index) รุ่นของโคออร์ดิเนเตอร์
- MRF มาสเตอร์ใช้กำหนด ช่องสัญญาณ (Channel) และ หมายเลขเครือข่าย (PAN ID) ให้กับตัวรับส่งวิทยุ MRF24J40

ตารางที่ 3.1 เฟรม MOD-bus ที่สื่อสารระหว่างมาสเตอร์กับโคออร์ดิเนเตอร์

Function Code	Payload (Command)	Payload (Response)
General settings		
PIndex	-	Index:2, SN:2
MRF	Channel, PANid:2	-
Buffer control		
BufferClear	-	
BufferStatus	-	drp, dwp
BufferFetch	Pctrl	Pres, drp, [packet]
Frame transmission control		
TrFloodAll	FCommand, {...}	-
TrFloodPAN	FCommand, {...}	-
TrNode	Node:2, {MSDU}	-
TrCode	Node:2, Page	-
Reprogram code control		
RPGcode0	CODE_SEG_0	-
RPGcode1	CODE_SEG_1	-
RPGcode2	CODE_SEG_2	-
RPGcode3	CODE_SEG_3	-
Endpoints		
EPtable	Index	NofEP, Index, EP_Profile:20
GetEPinfo	Node:2, EP	Status, EP_Profile:20
GetEPstatus	Node:2, EP	Status
State control		
SetState	State, [Node:2]	
GetState		State

กลุ่มที่สอง Buffer control มาสเตอร์ใช้เพื่อควบคุมบัฟเฟอร์ที่พักข้อมูลจากเครือข่าย กำหนดให้ข้อมูล 1 ชุด เรียกว่า 1 แพคเกต (Packet), บัฟเฟอร์ดังกล่าวถูกออกแบบให้จุข้อมูลได้ 256 แพคเกต เป็นบัฟเฟอร์แบบ Circular มีพอยน์เตอร์สำหรับอ่านและเขียน ชื่อว่า drp และ dwp ตามลำดับมี Function code อยู่ 3 แบบ ได้แก่

- BufferClear ล้างข้อมูลในบัฟเฟอร์ทิ้งโดยการ ตั้งค่า drp และ dwp เท่ากับศูนย์
- BufferStatus มาสเตอร์ใช้สอบถามค่า drp และ dwp จากโคออร์ดิเนเตอร์
- BufferFetch มาสเตอร์สั่งอ่าน ข้อมูล จากบัฟเฟอร์ โดยระบุโหมดการอ่านด้วย Pctrl, หาก Pctrl = 0 เป็นการสั่งอ่าน ณ ข้อมูลชุดที่ drp ชี้อยู่, แต่ถ้าหาก Pctrl ≠ 0 เป็นการสั่งอ่านข้อมูล เริ่มจาก drp ถอยหลังไปเป็นจำนวนชุดเท่ากับ Pctrl, แต่ Pctrl สั่งถอยหลังได้ไม่เกิน 63 ชุด, ส่วนการตอบสนอง โคออร์ดิเนเตอร์จะส่ง Pres (0: หมายถึงอ่านข้อมูลได้ตามคำร้องขอ, 0x40 ไม่มีข้อมูลให้อ่าน บัฟเฟอร์ว่าง, 0x41 ไม่สามารถอ่านข้อมูลชุดที่สั่งถอยหลังได้ เนื่องจากบัฟเฟอร์ถูกเขียนทับไปแล้ว, 0x42 ไม่สามารถอ่านข้อมูลย้อนหลังได้ตามจำนวนที่ขอ แต่ย้อนหลังได้บางชุด) และ

ส่ง drp กลับมาด้วยเป็นการบอกว่า แพคเกตที่ตามมา เป็นชุดที่เท่าไหร่, ในกรณี Pres เป็น 0x40 หรือ 0x41 จะไม่มีแพคเกตตามมา, ในการอ่าน ค่า drp จะเพิ่มขึ้นหนึ่งค่าทุกครั้งที่อ่านข้อมูลได้

กลุ่มที่สาม Frame transmission control มาสเตอร์ใช้สั่งโคออร์ดิเนเตอร์ให้ส่งเฟรมเข้าไปในเครือข่ายเซนเซอร์ไร้สาย ประกอบด้วย

TrFloodPAN มาสเตอร์สั่งโคออร์ดิเนเตอร์ให้กระจายเฟรมให้ทั่วถึงทุกโหนดในเครือข่าย โดยมีจุดประสงค์ที่ระบุด้วย FCommand ที่แสดงในตารางที่ 3.2 และ {...} เป็นฟิลด์เพิ่มเติมที่จะใช้ส่งออกอากาศ มีรายละเอียดในตารางที่ 3.2 เช่นกัน

- TrFloodAll เหมือนกัน TrFloodPAN ทุกประการ เว้นแต่จะกระจายข้อความถึงทุกโหนด โดยไม่สนใจว่าโหนดอยู่ในเครือข่ายหรือไม่

- TrNode มาสเตอร์สั่งโคออร์ดิเนเตอร์ส่งข้อความถึงโหนดที่ระบุด้วยฟิลด์ Node, ส่วนฟิลด์ MSDU จะเป็นส่วนหนึ่งของเฟรมที่ส่งไป ดูรายละเอียดในหัวข้อ 3.3.2

- TrCode มาสเตอร์สั่งโคออร์ดิเนเตอร์ส่งรหัสเพื่อโปรแกรมแอพพลิเคชันซ้ำ โดยระบุหมายเลขโหนด (Node) และ หมายเลขเพจ (Page), ในหนึ่งเพจจะมีรหัสยาว 64 ไบต์ ซึ่งโคออร์ดิเตอร์มีรหัสสำหรับโปรแกรมอยู่ด้วยด้วย กลุ่มที่สี่ Reprogram code control

กลุ่มที่สี่ Reprogram code control มาสเตอร์จะใช้เพื่อส่งรหัสสำหรับโปรแกรมซ้ำเข้าไปพักไว้ในบัฟเฟอร์ภายในโคออร์ดิเนเตอร์ก่อน

กลุ่มที่ห้า Endpoints มาสเตอร์ใช้เพื่อสอบถามโปรไฟล์ของข้อมูลชั้นแอพพลิเคชัน เรียกว่า “เอ็นพอยนต์” (Endpoint) เราจะกล่าวถึงรายละเอียดของ EP_Profile ในบทที่ 4

- EPtable มาสเตอร์สอบถาม EP_Profile จากโคออร์ดิเนเตอร์ โดยระบุเอ็นพอยต์ด้วยอินเด็กซ์, โคออร์ดิเนเตอร์ จะตอบกับด้วย NofEP-จำนวนเอ็นพอยนต์ทั้งหมด อินเดกซ์ และ ตามด้วย EP_Profile

- GetEPinfo มาสเตอร์สอบถาม สถานะ และ EP_Profile จากโคออร์ดิเนเตอร์ โดยระบุเอ็นพอยนต์ด้วย EP-หมายเลขเอ็นพอยนต์ และ Node-หมายเลขโหนด, โคออร์ดิเนเตอร์จะตอบกับด้วย Status-ซึ่งบ่งบอกได้ว่า ข้อมูลของเอ็นพอยนต์ดังกล่าว ถูกส่งมายังโคออร์ดิเนเตอร์ครั้งล่าสุดนานเกิน 1 นาทีหรือไม่ และตามด้าย EP_Profile

- GetEPstatus มาสเตอร์สอบถาม สถานะ เช่นเดียวกับ GetEPinfo แต่โคออร์ดิเนเตอร์จะตอบกลับเพียงแค่ Status

กลุ่มที่หก State control มาสเตอร์ใช้เพื่อ สอบถามและสั่งเปลี่ยน สถานะการทำงานของโคออร์ดิเนเตอร์ ซึ่งมีอยู่ 3 สถานะ ทำงานปกติ (รวบรวมข้อมูลจากเครือข่าย), หยุดทำงาน (หยุดรวบรวมข้อมูล), และโปรแกรมซ้ำ

- SetState มาสเตอร์สั่งเปลี่ยนสถานะของโคออร์ดิเนเตอร์, สำหรับการสั่งเปลี่ยนสถานะให้เป็นการโปรแกรมซ้ำ มาสเตอร์จะต้องระบุ Node-หมายเลขโหนดมาด้วย

- GetState มาสเตอร์สอบถาม สถานะการทำงาน จากโคออร์ดิเนเตอร์

ตารางที่ 3.2 แสดงฟิลด์ที่ตาม FCommand ในเฟรมที่มี Function code เป็น TrFloodAll และ TrFloodPAN เพื่อใช้กระจายข้อความในเครือข่ายนั้น มีจุดประสงค์การใช้งาน

- Application ถวงถาม/ส่งข้อมูลของชั้นแอพพลิเคชัน PCount คือความยาวของแพคเกต

- Activity กำหนดกิจกรรมการทำงานของเครือข่าย

- Realign สั่งให้สร้างเครือข่ายใหม่ กำหนด Delay-เวลาหน่วง นับตั้งแต่เวลาที่ได้รับเฟรมจนถึงเริ่มต้นขบวนการ และ TH-LQI ค่าคุณภาพลิงค์ขั้นต่ำ ใช้ในกระบวนการเลือกอัพลิงค์

- Reentry กำหนดให้โหนดที่ถูกระบุ เริ่มขบวนการเข้าร่วมเครือข่ายอีกครั้ง

- DefinePAN กำหนดโหนดที่ถูกระบุ เลือกเข้าร่วมเฉพาะเครือข่ายหมายเลข PAN ID

ตารางที่ 3.2 คำอธิบาย FCommand ใน TrFloodAll และ TrFloodPAN

FCommand	Description and Payload
Application (Application layer)	โคออร์ดิเนเตอร์ใช้ส่ง เพคเกจ ให้กับโหนด (หนึ่งโหนดหรือทุกโหนด) `PCount, [packet]`
Activity	โคออร์ดิเนเตอร์ใช้กำหนดกิจกรรมของโหนด (หนึ่งโหนดหรือทุกโหนด) `Activity-Code, Node:2` Activity-Code ∈ {0, 1, 2} {Idle, Run, Reprogram}
Realign	โคออร์ดิเนเตอร์สั่งให้ทุกโหนด เตรียมตัวสร้างเครือข่ายใหม่ `Delay, TH-LQI`
Reentry	โคออร์ดิเนเตอร์สั่งโหนดทำขบวนการเข้าร่วมเครือข่ายอีกครั้ง `Node:2`
DefinePAN	โคออร์ดิเนเตอร์สั่งกำหนดค่าหมายเลขเครือข่ายถาวร (Fixed PAN ID) ให้แก่โหนด (หนึ่งโหนด/ทุกโหนด) `Node:2, PANid:2`

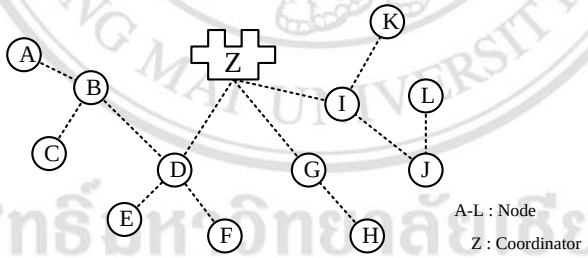
3.3 ชั้นโพรโทคอลเครือข่ายเซนเซอร์ไร้สาย

การสื่อสารในเครือข่ายไร้สายมีพฤติกรรมแตกต่างจากเครือข่ายใช้สายเป็นอย่างมาก เนื่องด้วยปัจจัยหลายประการ อาทิเช่น เครือข่ายไร้สายจำเป็นต้องอาศัยกลไกส่งต่อข้อมูลกันหลายฮอพ (Multi-hop) เพราะกำลังส่งวิทยุแต่ละโหนดมีขีดจำกัด การสื่อสารกับโหนดที่อยู่ไกลกันจำเป็นต้องอาศัยโหนดอื่นที่อยู่ในรัศมีช่วยส่งต่อให้ จึงต้องมีโพรโทคอลเข้ามาจัดการเครือข่าย

เครือข่ายไร้สายในแพลตฟอร์มนี้ใช้ตัวรับส่งวิทยุ MRF24J40MB ที่ทำงานตามมาตรฐาน IEEE 802.15.4 ตัวรับส่งมีโพรเซสเซอร์ภายในช่วยจัดการการสื่อสารในชั้น กายภาพ และ MAC ให้แล้ว ซอฟต์แวร์ของไมโครคอนโทรลเลอร์จึงมีหน้าที่จัดการตั้งแต่ชั้นเครือข่ายไปจนถึงชั้นแอพพลิเคชัน

3.3.1 โทโพโลยี

ในการขยายพื้นที่การมอนิเตอร์ให้กว้างขวาง จึงต้องติดตั้งโหนดให้กระจายกันทั่วพื้นที่ ทำให้การส่งข้อมูลเข้าหาโคออร์ดิเนเตอร์ต้องส่งต่อกันหลายฮอพ แพลตฟอร์มนี้เลือกใช้โทโพโลยีแบบทรี ซึ่งรองรับการสื่อสารแบบหลายฮอพ และโคออร์ดิเนเตอร์ทำหน้าที่เป็น Sink node (โหนดที่รวบรวมข้อมูล) ด้วย

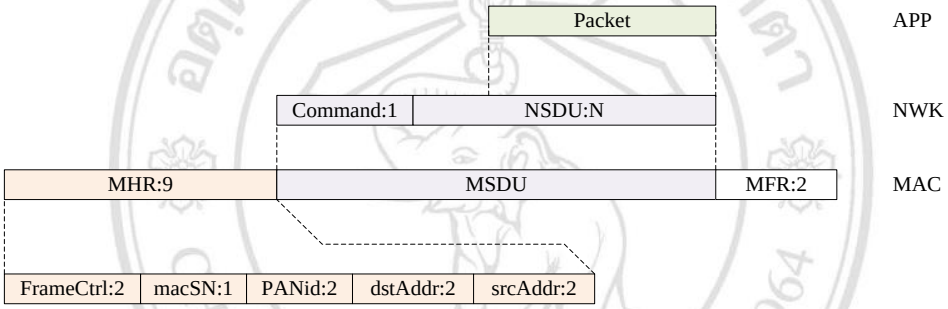


ภาพที่ 3.8 ตัวอย่างโทโพโลยีแบบทรี

พิจารณาตัวอย่างเครือข่ายโทโพโลยีแบบทรีในภาพที่ 3.8 ประกอบด้วย โคออร์ดิเนเตอร์-Z และโหนด A ถึง L จำนวน 12 โหนด, โหนด D G และ I อยู่ในระดับที่ 1 ส่วนโหนด B, E, F, H, J, K อยู่ในระดับที่ 2 ถ้าโหนด H ต้องการติดต่อกับโคออร์ดิเนเตอร์จะต้องผ่านโหนด G, เราจะเรียกโหนด D ว่าเป็นอัพลิงค์ (Uplink) ของโหนด F, และในทำนองเดียวกัน อัพลิงค์ของ D, G และ I ก็คือโคออร์ดิเนเตอร์ จากตัวอย่างโหนดในระดับที่ 3 ได้แก่ A, C และ L จะต้องใช้ 3 ฮอพจึงจะติดต่อกับโคออร์ดิเนเตอร์ได้ ในแพลตฟอร์มนี้กำหนดระดับสูงสุดของโหนดเท่ากับ 6

3.3.2 รูปแบบเฟรมและฟิลด์

รูปแบบเฟรมที่ส่งออกอากาศอิงมาตรฐาน IEEE 802.15.4 [1] พิจารณาตัวอย่างรูปแบบเฟรมชั้น MAC ในภาพที่ 3.9 แบ่งออกเป็น ส่วนหัว (MHR: MAC Header), ส่วนข้อมูล (MSDU: MAC Service Data Unit) และ ส่วนท้ายเฟรม (MFR: MAC Footer) ซึ่งเป็น CRC ใช้ตรวจสอบความถูกต้องของเฟรม, MSDU เป็นเฟรมของชั้นเครือข่ายถูกใช้งานเพื่อจุดประสงค์ต่างๆ ตามโพรโทคอลในแพลตฟอร์มนี้ ไบต์แรกเป็นฟิลด์ Command ใช้เพื่อแยกแยะคำสั่งต่างๆ ดังแสดงในตารางที่ 3.3 ไบต์ที่เหลือเป็นฟิลด์ NSDU (Network Service Data Unit) มีโครงสร้างขึ้นอยู่กับฟิลด์ Command สำหรับเฟรมที่ใช้ส่งข้อมูลชั้นของแอพพลิเคชันนั้นจะตัดเอาบางส่วนของ NSDU ไปใช้เราเรียกข้อมูลชั้นแอพพลิเคชันว่า “แพคเกต” (Packet) ซึ่งจะกล่าวถึงรายละเอียดของแพคเกตในบทที่ 4



ภาพที่ 3.9 รูปแบบเฟรมสำหรับชั้นโพรโทคอลไร้สาย

สำหรับ MHR ตามมาตรฐาน IEEE802.15.4 สามารถเลือกได้หลายรูปแบบ แต่แพลตฟอร์มนี้เลือกใช้รูปแบบตามภาพที่ 3.9 ซึ่งประกอบด้วย 5 ฟิลด์ ได้แก่

- Frame Ctrl ขนาด 2 ไบต์ ใช้ระบุประเภท และ พารามิเตอร์ของเฟรม
- MAC Sequence No. ขนาด 1 ไบต์ หมายเลขเฟรม, ถูกนับขึ้นทีละหนึ่งค่าต่อการส่งหนึ่งเฟรม
- PAN ID ขนาด 2 ไบต์ หมายเลขเครือข่าย
- Destination Address ขนาด 2 ไบต์ หมายเลขโหนดผู้รับเฟรม
- Source Address ขนาด 2 ไบต์ หมายเลขโหนดผู้ส่งเฟรม

ตารางที่ 3.3 Command และ NSDU ของเฟรมในชั้นเครือข่าย

Command	NSDU	TM (Transmission Mode)
Flood		
FloodPAN	Index, FCommand, {...}	flood
FloodAll	Index, FCommand, {...}	flood
Network organize		
UplinkRQ	-	broad
UplinkRS	Level, PANid:2	direct
JoinRQ	-	up
JoinRS	Level	down
Entry	Node:2	trace
Unknown	Node:2	trace
IdleIN	-	down
ExitIN	-	broad
Data transfer		
SinkDATA	Level, Index, Count, [packet]	up
AckDATA	Index	down
Query	Count, [packet]	single
Report	Count, [packet]	root
Reprogram		
RPGcmd	Command, Page:2	pipe
RPGres	Response, Page:2	root
CODE	Code:64	pipe

ฟิลด์ Command ตามตารางที่ 3.3 แบ่งออกได้เป็น 4 กลุ่ม ได้แก่ กลุ่มกระจายข้อความ (Flooding), กลุ่มจัดการเครือข่าย (Network Organization), กลุ่มส่งข้อมูล (Data Transfer), และ กลุ่มโปรแกรมซ้ำ (Reprogram) จุดประสงค์ของแต่ละกลุ่มใช้เพื่อตามชื่อของกลุ่มนั้นๆ, ในตารางที่ 3.3 มีสดมภ์ Transmission Modes ระบุวิธีการส่งต่อเฟรมนั้นและแสดงรายละเอียดในหัวข้อ 3.3.3

ภายในฟิลด์ NSDU ของแต่ละเฟรมประกอบด้วยฟิลด์ย่อยต่างๆ ซึ่งจะถูกใช้งานแตกต่างกันออกไปในแต่ละเฟรม คำอธิบายพอสังเขปดังต่อไปนี้

- Node หมายเลขโหนดขนาด 2 ไบต์
- PAN ID หมายเลขเครือข่ายขนาด 2 ไบต์
- Level หมายถึงระดับความลึก หรือจำนวนฮอพนับจากโคออร์ดิเนเตอร์, โหนดที่ติดต่อกับ โคออร์ดิเนเตอร์โดยตรงจะมี Level = 1, ส่วนโหนดที่อยู่ถัดออกไป Level = 2
- LQI คุณภาพสัญญาณขนาด 1 ไบต์ ตามมาตรฐาน IEEE 802.15.4
- [packet] เพคเกต ข้อมูลของชั้นแอพพลิเคชัน

- Count ความยาวของเพคเกต ขนาด 1 ไบต์
- Index ขนาด 1 ไบต์ ควบคุมเฟรม Flooding ไม่ให้ซ้ำซ้อน และ ควบคุมโฟลว์ข้อมูล
- Command, Response ขนาด 1 ไบต์ ฟิลด์คำสั่งย่อยและผลตอบสนอง ใช้สั่งงาน/ตอบโต้ในกลุ่มงานที่เฟรมนั้นๆ เกี่ยวข้อง
- Page ขนาด 2 ไบต์ เป็นหมายเลข Memory page ในหน่วยความจำสำหรับโปรแกรมซ้ำ
- Code ขนาด 64 ไบต์ เป็นรหัสสำหรับโปรแกรมซ้ำ

กลุ่มกระจายข้อความ (Flooding) มีเฟรมอยู่ 2 คำสั่ง ได้แก่ FloodPAN และ FloodAll, ทั้งสองรูปแบบมีกลไกการกระจายข้อความเหมือนกัน แต่ FloodPAN จะกระจายข้อความไปถึงโหนดที่อยู่ในเครือข่ายเท่านั้น ส่วน FloodAll จะกระจายข้อความไปถึงทุกโหนดแม้ว่าโหนดนั้นไม่ได้อยู่ในเครือข่ายก็ตาม จุดประสงค์ของการกระจายข้อความมีทั้งการบริการชั้นแอพพลิเคชัน เช่น การตั้งเวลา, การร้องขอข้อมูล และ การตั้งค่า เป็นต้น นอกจากนี้ยังใช้บริการชั้นเครือข่ายด้วย เช่น การสร้างเครือข่ายใหม่, การกำหนดหมายเลขเครือข่ายให้โหนด เป็นต้น (ดูรายละเอียดของ FCommand ในตารางที่ 3.2)

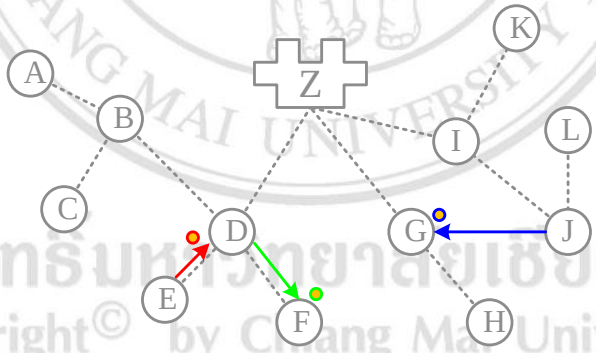
กลุ่มจัดการเครือข่าย (Network Organization) มีเฟรมอยู่ 8 คำสั่ง แบ่งออกเป็น 4 คู่, คู่แรก UplinkRQ – UplinkRS (Uplink Request, Uplink Response) เป็นเฟรมที่โหนดใช้ร้องขอโหนดอื่นให้เป็นอัพลิงค์ของมัน และ เป็นเฟรมที่โหนดที่พร้อมเป็นอัพลิงค์ใช้ตอบกลับการร้องขอ ตามลำดับ, คู่ที่สอง JoinRQ – JoinRS (Join Request, Join Response) เมื่อโหนดเลือกอัพลิงค์ได้แล้ว มันจะส่งเฟรม JoinRQ ออกไปให้โหนดที่มันเลือกเป็นอัพลิงค์ และโหนดอัพลิงค์ก็จะตอบกลับด้วย JoinRS, คู่ที่สาม Entry – Unknown เป็นเฟรมที่โหนดส่งไปแจ้งโคออร์ดิเนเตอร์และทุกโหนดที่อยู่ในเส้นทางถึงการเข้าร่วม และ หายไป ของโหนดในระดับที่มากขึ้นในเส้นทางนั้น สองเฟรมนี้จะมีผลต่อ ตารางส่งต่อ (Next-hop Table) ของโหนดในเส้นทางที่เฟรมนั้นผ่านไป, คู่สุดท้าย IdleIN – ExitIN (Idle Indicate, Exit Indicate) โหนดใช้แจ้งโหนดอื่นว่ามัน อยู่ในสถานะไม่พร้อมทำงาน และ ออกจากเครือข่ายแล้ว ตามลำดับ ซึ่งจะมีผลให้โหนดที่เป็นโหนดลูกของมันเริ่มกระบวนการค้นหาเครือข่ายใหม่อีกครั้ง

กลุ่มส่งข้อมูล (Data Transfer) มีเฟรมอยู่ 4 คำสั่ง แบ่งเป็น 2 คู่ SinkDATA – AckDATA และ Query – Report สำหรับส่งข้อมูลแบบ กระตุ้นด้วยเหตุการณ์ (Event Driven) และ ทวงถาม (Query Based) ตามลำดับ, รายละเอียดจะกล่าวถึงในลำดับถัดไป

กลุ่มโปรแกรมซ้ำ (Reprogram) มีเฟรมอยู่ 3 คำสั่ง RPGcmd (Reprogram Command), RPGres (Reprogram Response), และ CODE สำหรับไว้ให้โคออร์ดิเนเตอร์สั่งงาน และโหนดเป้าหมายตอบสนองต่อการโปรแกรมซ้ำ ส่วนเฟรม CODE ใช้ส่งรหัสสำหรับโปรแกรมซ้ำ

3.3.3 วิธีการส่งต่อเฟรม

วิธีการส่งต่อเฟรม (TM: Transmission Mode) จะบ่งบอกลักษณะ การส่งเฟรมระหว่างโหนด และ การส่งต่อเฟรมแบบหลายฮอพ รวมไปถึงการบริการต่อเฟรม (หมายถึงการกระทำคำสั่ง หรือคำร้องขอ ที่มากับเฟรมนั้น) สรุปแล้วเมื่อโหนดใดๆ ได้รับเฟรมจะกระทำหรือไม่กระทำใน 2 ประเด็น ได้แก่ บริการ (Service) และ ส่งต่อ (Forward), การส่งเฟรมออกอากาศไปโดยเจาะจงหมายเลขโหนดในฟิลด์ Destination Address นั้นจะมีโหนดผู้รับเพียงโหนดเดียว คือ โหนดที่มีหมายเลขตรงกับฟิลด์เท่านั้น ส่วนการกระจายเฟรม (Broadcast) จะระบุค่า 0xFFFF ไว้ในฟิลด์ Destination Address ซึ่งโหนดทุกโหนดที่อยู่ในรัศมีจะเป็นผู้รับเฟรม



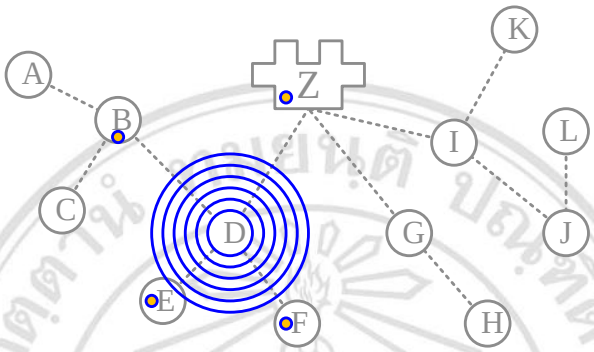
ภาพที่ 3.10 วิธีการส่งเฟรมแบบ Up, Down และ Direct

ในภาพที่ 3.10 สัญลักษณ์วงกลมเล็กพื้นสีเหลืองที่อยู่ในหรือข้างโหนดใด หมายความว่าโหนดนั้นจะต้องให้บริการต่อเฟรมที่ได้รับนั้นด้วย, ลูกศร แสดงเฟรมจากผู้ส่งไปหาผู้รับ, เส้นประ แสดงลิงค์ที่สร้างขึ้นในโทโพโลยี, และวงกลมหลายชั้น แสดงการกระจายเฟรม

Up: ส่งขึ้นหนึ่งฮอพ ส่งจากโหนดไปหาอัพลิงค์ของมัน สังเกตเฟรมสีแดงในภาพที่ 3.10 โหนดที่เป็นอัพลิงค์เป็นฝ่ายรับ มันจะบริการต่อเฟรมที่บรรจุคำร้องขามาจากโหนดลูกด้วย

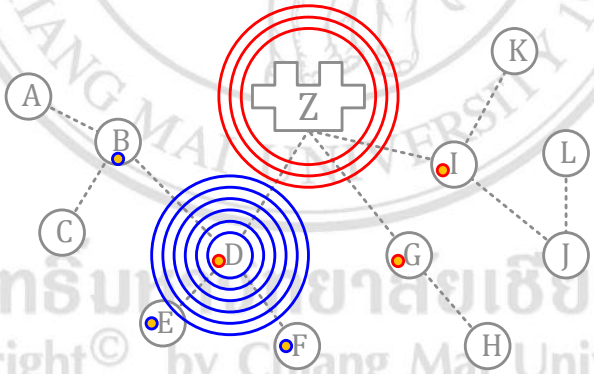
Down: ส่งลงหนึ่งฮอพ ส่งจากอัพลิงค์ไปหาโหนดลูก สังเกตเฟรมสีเขียวในภาพที่ 3.10 โหนดจะเป็นฝ่ายรับ และ บริการต่อเฟรมที่บรรจุ คำสั่ง/ผลตอบสนอง ที่มาจากอัพลิงค์ของมัน

Direct: ส่งตรงจากโหนดไปหาอีกโหนดหนึ่ง โดยไม่คำนึงลิงค์ในโทโพโยลี เฟรมนี้จะถูกเพียงหนึ่งฮอพ ไม่มีการส่งเฟรมนี้ต่อ สังเกตเฟรมสีน้ำเงินในภาพที่ 3.10



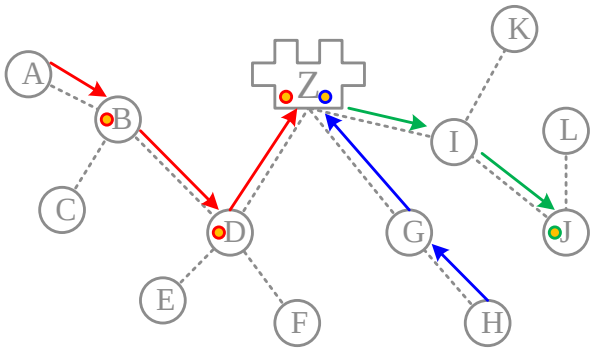
ภาพที่ 3.11 วิธีการส่งเฟรมแบบ Broad

Broad: เฟรมจะถูกกระจาย (Broadcast) จากโหนดใดโหนดหนึ่ง ไปหาทุกโหนดในรัศมีที่รับสัญญาณวิทยุของโหนดผู้ส่งได้ เฟรมนี้จะถูกเพียงหนึ่งฮอพ ไม่มีการส่งเฟรมนี้ต่อ สังเกต ภาพที่ 3.11



ภาพที่ 3.12 วิธีการส่งเฟรมแบบ Flood

Flood: เป็นการกระจายข้อความจากโคออร์ดิเนเตอร์ โดยมีเป้าหมายเป็นทุกโหนดในเครือข่าย เริ่มแรกโคออร์ดิเนเตอร์จะกระจายเฟรมออกไปให้ทุกโหนด จากนั้นโหนดที่ได้รับเฟรมจะตรวจสอบจากฟิลด์ Index ว่าเป็นเฟรมที่มันเคยได้รับแล้วหรือยัง ถ้าไม่เคยได้รับ มันจะบริการตามเฟรมนั้นและกระจายเฟรมนั้นออกไปอีก แต่ถ้าเป็นเฟรมที่มันเคยได้รับแล้ว มันก็จะโยนเฟรมนั้นทิ้งไป ภาพที่ 3.12 แสดงกลไกการส่งเฟรมแบบ Flood, โหนด D รับและบริการเฟรมแล้วกระจายเฟรมต่อไปอีก ด้วยกลไกลเช่นนี้ โหนดทุกโหนดจะได้รับเฟรมที่กระจายจากโคออร์ดิเนเตอร์



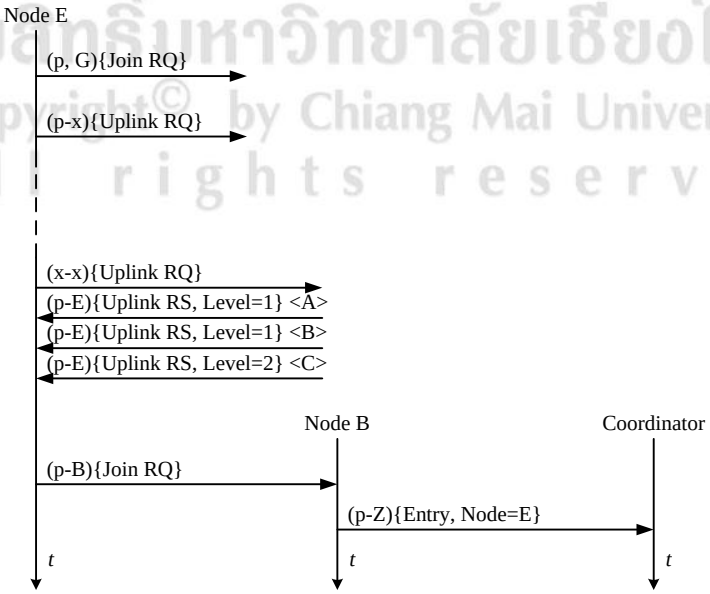
ภาพที่ 3.13 วิธีการส่งเฟรมแบบ Trace, Root และ Single

Trace: โหนดจะส่งเฟรมไปหาอัพลิงค์ของมัน อัพลิงค์จะบริการเฟรมนั้นและส่งต่อไปยังอัพลิงค์ของมันในชั้นถัดไป เป็นเช่นนี้ไปเรื่อยๆ จนถึงโคออร์ดิเนเตอร์ภาพที่ 3.13 เส้นสีแดง แสดงวิธีการส่งเฟรมแบบ Trace

Root: เริ่มต้นจากโหนดไปหาโคออร์ดิเนเตอร์คล้ายกับ Trace แต่โหนดที่อยู่ระหว่างทางจะไม่บริการเฟรมนั้น เป็นเพียงผู้ส่งต่อ โหนดที่จะให้บริการคือโคออร์ดิเนเตอร์เท่านั้นภาพที่ 3.13 เส้นสีน้ำเงินแสดงวิธีการส่งเฟรมแบบ Root

Single: คล้ายกับ Root แต่กลับทิศทางกัน เริ่มต้นจากโคออร์ดิเนเตอร์ไปหาโหนดใดโหนดหนึ่งเพียงโหนดเดียว โหนดที่อยู่ระหว่างทางเป็นผู้ส่งต่อเฟรม โหนดที่จะให้บริการต่อเฟรมนั้นมีเพียงโหนดปลายทาง (Target) เท่านั้นภาพที่ 3.13 เส้นสีเขียวแสดงวิธีการส่งเฟรมแบบ Single

Pipe: คล้าย Single แต่ใช้สำหรับใช้ สำหรับโปรแกรมซ้ำเท่านั้น



ภาพที่ 3.14 ตัวอย่างผังเวลาแสดงลำดับเฟรมชั้นเครือข่าย

ภาพที่ 3.14 แสดงตัวอย่างผังเวลาแสดงลำดับการส่งเฟรมในชั้นเครือข่าย เส้นลูกศรในแนวดิ่งแนวแกนเวลา ด้านบนของเส้นแกนเวลา เป็นชื่อโหนด, ลูกศรแนวนอนแสดงเฟรม ต้นลูกศรลากออกจากแกนเวลาของโหนดผู้ส่ง หัวลูกศรชี้ไปทางโหนดผู้รับ, ผังดังกล่าวแสดงเฉพาะเฟรม ข้อมูลของชั้น MAC ซึ่งในความเป็นจริงแล้วจะมีเฟรม Acknowledgement จากฝั่งรับตอบกลับด้วย แต่ได้ละเลยไม่แสดงเฟรม Acknowledgement ไว้, บนลูกศรแสดงเฟรม มีอักษรกำกับอยู่ภายในเครื่องหมาย ( ) และเครื่องหมาย { }, ในเครื่องหมาย ( ) มีหมายเลขเครือข่าย-หมายเลขโหนดผู้รับ ซึ่งเป็นส่วนหนึ่งของ MHR ตัวอย่างเช่น (p-G) หมายถึงโหนด G ในเครือข่ายที่มี PAN ID = p, ในกรณี (x-x) หมายถึงโหนดใดก็ได้ ใน PAN ID ใดก็ได้ อีกนัยหนึ่งที่ หมายเลขโหนดแสดงด้วย x คือการกระจายเฟรม

อักษรที่อยู่ในเครื่องหมาย { } เป็นส่วนของ MSDU, เริ่มต้นด้วย Command และ ตามด้วยฟิลด์ต่างๆ ของ NSDU

ในบางกรณี แสดงโหนดผู้ส่งเฟรมในเครื่องหมาย < > เช่น <A> แสดงโหนดเฟรมนั้นถูกส่งโดยโหนด A

3.3.4 สถานะของโหนด

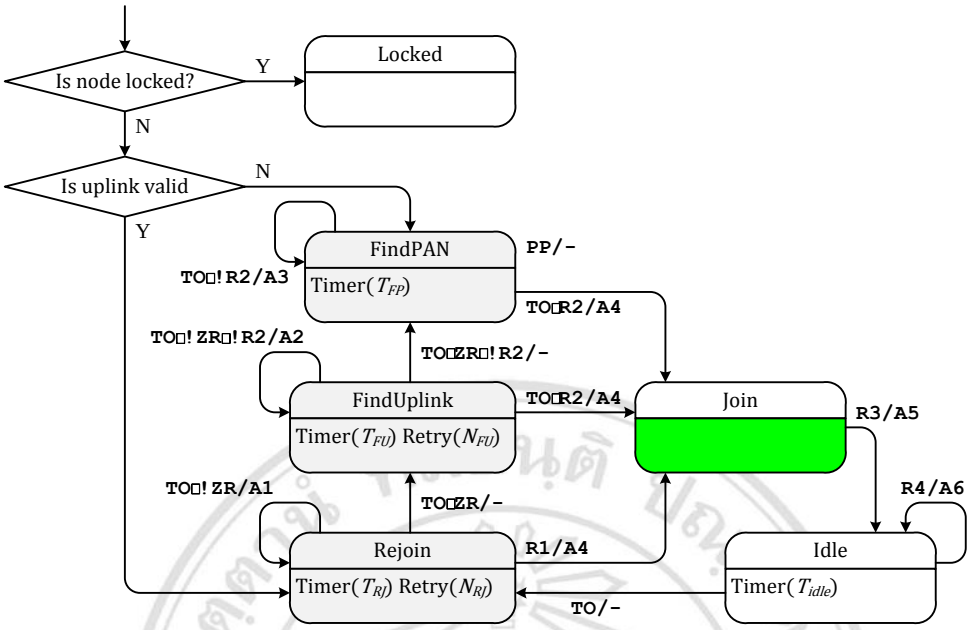
ก่อนที่โหนดจะส่งข้อมูลสู่โคออร์ดิเนเตอร์ได้ มันจะต้องการเข้าร่วมเครือข่ายก่อน พิจารณาผังสเตตของโหนดสำหรับการเข้าร่วมเครือข่ายในภาพที่ 3.15, เริ่มต้นเมื่อโหนดได้รับไฟเลี้ยง และเริ่มทำงานมันจะตรวจสอบก่อนว่า ตัวโหนดเองถูกล็อคไว้หรือไม่ ถ้าถูกล็อคไว้ก็จะเข้าสู่สเตต Locked และไม่สามารถเข้าร่วมเครือข่ายได้ ต้องรอการปลดล็อคอย่างเดียว การล็อคโหนดกระทำเพื่อป้องกันการคัดลอก (รายละเอียดการปลดล็อค ไม่เปิดเผยในรายงานนี้)

ในกรณีที่โหนดถูกปลดล็อคแล้ว จะไปตรวจสอบ “Is uplink valid” ถ้ามีค่าหมายเลขโหนดที่เคยเป็นอัพลิงค์ของมันที่ใช้ได้ แสดงว่าโหนดนั้นแบ็คอัพหมายเลขเครือข่าย (PAN ID) และหมายเลขอัพลิงค์ เดิมไว้ได้ จึงเข้าสู่สเตต “Rejoin” แต่ถ้าโหนดไม่ได้แบ็คอัพหมายเลขเครือข่ายและอัพลิงค์ไว้ หรือ แบ็คอัพแล้วผิดพลาด โหนดก็จะเข้าสู่สเตต “FindPAN”

สเตต “Rejoin” โหนดจะขอเข้าร่วมเครือข่าย โดยร้องขอไปยังอัพลิงค์เดิม ถ้าได้รับการตอบรับมันก็จะเข้าสู่สเตต “Join” ซึ่งถือว่าเข้าร่วมเครือข่ายสำเร็จ, แต่ถ้าโหนดร้องขอไปยังอัพลิงค์เดิมไป N_{RJ} ครั้ง แต่ละครั้งห่างกันเป็นเวลา T_{RJ} แต่ไม่มีการตอบสนอง โหนดจะเข้าสู่สเตต “FindUplink”

สเตต “FindUplink” โหนดจะร้องขอ อัพลิงค์ใหม่ ในเครือข่ายเดิม โดยโหนดจะร้องขอด้วยการกระจายเฟรมออกไป หากได้รับเฟรมตอบรับจากโหนดอื่น ตั้งแต่ 1 โหนดขึ้นไป มันเลือกอัพลิงค์จากโหนดเหล่านั้นตามกระบวนการ และขอเข้าร่วมเครือข่าย เข้าสู่สเตต “Join”, แต่ถ้าในกรณีตรงข้าม ไม่มีโหนดใดตอบรับเลย หลังจากร้องขอไป N_{FU} ครั้ง แต่ละครั้งห่างกันเป็นเวลา T_{FU} โหนดจะเข้าสู่สเตต “FindPAN”

สเตต “FindPAN” โหนดจะทำงานคล้ายสเตต “FindUplink” แต่จะร้องขออัพลิงค์ใหม่ ในเครือข่ายใดก็ได้ (PAN ID = 0xFFFF) จะร้องขอทุกๆ T_{FP} จนกว่าจะได้รับการตอบรับ



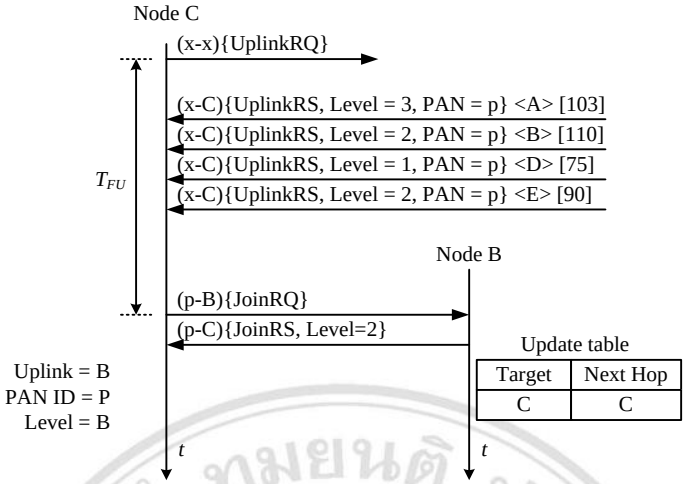
ภาพที่ 3.15 สเตตการทำงานของชั้นเครือข่ายที่ฝั่งโหนด

สเตต “Join” โหนดอยู่ในเครือข่าย สามารถรับ/ส่งข้อมูลได้ แต่เมื่อพบข้อผิดพลาดร้ายแรง หรืออัพลิงค์ของมันหลุดออกจากสเตต Join โหนดนั้นก็ต้องออกจากสเตต “Join” เข้าสู่สเตต “Idle”, ในสเตต “Idle” นี้โหนดจะยุติการส่งข้อมูล รักษาข้อมูลไว้ในบัฟเฟอร์ไว้ รอเวลา T_{idle} แล้วเข้าสู่สเตต “Rejoin” เพื่อเริ่มขบวนการเข้าสู่เครือข่ายใหม่อีกครั้ง

3.3.5 การเข้าร่วมเครือข่าย

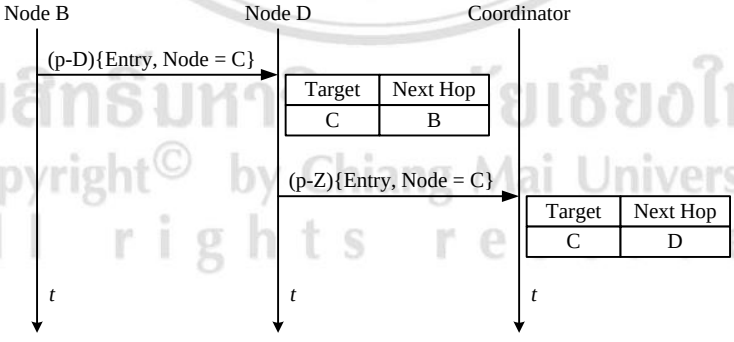
กลไกเข้าร่วมเครือข่ายของโหนดขึ้นอยู่กับสเตตตามที่กล่าวมาแล้ว แบ่งออกเป็นขั้นตอนย่อยได้แก่ การหาเครือข่าย/อัพลิงค์, การร้องขอเข้าร่วมเครือข่าย, และ การปรับตารางส่งต่อ

การหาเครือข่ายและการหาอัพลิงค์โหนด โหนดจะใช้เฟรม UplinkRQ เหมือนกัน เพียงแต่ว่าการหาเครือข่ายจะตั้งค่าฟิลด์ PAN ID เป็น 0xFFFF ส่วนการหาอัพลิงค์จะระบุค่า PNA ID ที่ต้องการ, พิจารณากรณีตัวอย่างโดยอาศัยโทโพโลยีจากภาพที่ 3.8 สมมติให้โหนด C เป็นโหนดที่เริ่มต้นทำงานและไม่มีหมายเลขเครือข่าย/อัพลิงค์แบ็คอัพไว้โหนด C จะส่งเฟรมตามขั้นตอนดังแสดงในภาพที่ 3.16 เริ่มจากส่งเฟรม UplinkRQ ออกไป โหนด A, B, D, E ได้รับเฟรมและตอบด้วยเฟรม UplinkRS, โหนดที่จะตอบรับด้วยเฟรม UplinkRS ได้นั้นจะต้องอยู่ในสเตต “Join” เท่านั้น จากนั้นโหนด C จะเลือกโหนดอัพลิงค์โดยพิจารณาจาก Level และ LQI ของโหนด กล่าวคือจะเลือกโหนดที่มี Level ต่ำสุดแต่ทว่าจะต้องมี LQI สูงกว่า LQITH (Threshold LQI) แต่ถ้าหากมีโหนดตอบมาเพียงโหนดเดียว โหนด C ก็จำเป็นต้องเลือกโหนดนั้นเป็นอัพลิงค์



ภาพที่ 3.16 การส่งเฟรม UplinkRQ-RS / JoinRQ-RS

เมื่อโหนด C เลือกอัพลิงค์ได้แล้วในกรณีนี้สมมติว่าเลือกโหนด B ก็จะส่งเฟรม JoinRQ ไปให้โหนด B, จากนั้นโหนด B จะตอบกลับด้วยเฟรม JoinRS การที่โหนด C ได้รับเฟรม JoinRS นี้ มันจะถือว่าเข้าร่วมเครือข่ายเรียบร้อยแล้ว และตั้งค่าหมายเลขอัพลิงค์ให้เป็นโหนด B รวมทั้งแบ็คอัพหมายเลขเครือข่ายไว้ด้วย มันจะตั้งค่า Level ของมันให้มากกว่า Level ของโหนด B อยู่ 1 ค่า ตัวอย่างนี้ Level ของโหนด C เท่ากับ 3 สังเกตตัวอย่างนี้ โหนด C ไม่เลือกโหนด D ที่มี Level ต่ำกว่าโหนด B ก็เพราะค่า LQI ของเฟรมที่ได้รับจากโหนด D ต่ำกว่า LQI_{TH}, (กำหนดไว้ 80) เมื่อโหนด B รับโหนด C เป็นโหนดลูกแล้วมันก็จะปรับปรุงตารางส่งต่อของมันเอง พร้อมทั้งรายงานการมีโหนดใหม่เข้าร่วมเครือข่ายไปยังโคออร์ดิเนเตอร์ด้วย



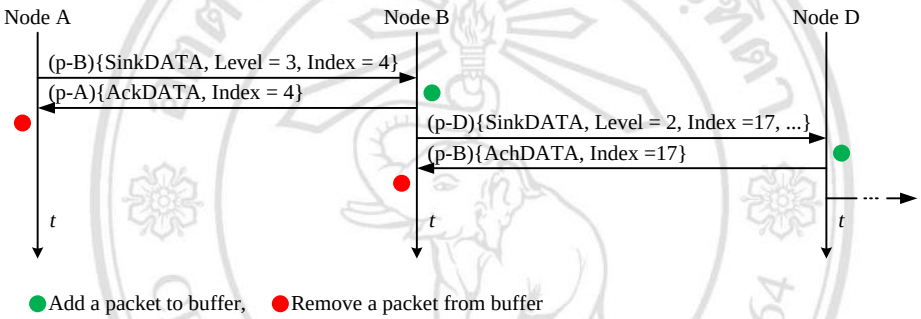
ภาพที่ 3.17 การรายงานโหนดใหม่เข้าร่วมเครือข่าย และการส่งเฟรม Entry

ภาพที่ 3.17 แสดงตัวอย่างโหนด B รายงานการเข้าร่วมเครือข่ายของโหนด C ไปยังอัพลิงค์ของมัน คือ โหนด D, เมื่อโหนด C ปรับปรุงตารางส่งต่อของตัวเองเรียบร้อยแล้ว มันจะส่งเฟรม Entry ไปให้โหนด D ซึ่งโหนด D ก็จะปรับปรุงตารางส่งต่อของตัวเองเช่นกัน แล้วรายงานต่อไปยังอัพลิงค์ของมัน ซึ่งก็คือโคออร์ดิเนเตอร์ เมื่อขบวนการเสร็จเรียบร้อยจะมี 3 โหนดที่ปรับปรุงตารางส่งต่อ คือ

โหนด B, D, และ โคออร์ดิเนเตอร์ ซึ่งค่า Next hop เมื่อ Target เป็นโหนด C จะมีค่าเป็น โหนด C, โหนด B, และ โหนด D ตามลำดับ

3.3.6 การรับส่งข้อมูลแบบกระตุ้นด้วยเหตุการณ์

โหนดที่อยู่ในสเตต “Join” และทำงานตามฟังก์ชันของมันปกติ เมื่อมีเหตุการณ์ที่กำหนดไว้เกิดขึ้น เช่น ครบคาบเวลาสุ่ม, ปริมาณที่วัดเกินระดับ, ได้รับฝากแพคเกตจากโหนดลูก, ตรวจจับสัญญาณที่กำหนดไว้ได้ เป็นต้น โหนดนั้นจะสร้างแพคเกตขึ้นมา, จากนั้นโหนดจะส่งแพคเกตไปให้อัพลิงค์ของมัน โดยใช้เฟรม SinkDATA และ AckDATA และมีขบวนการ Flow Control ดังตัวอย่างต่อไปนี้

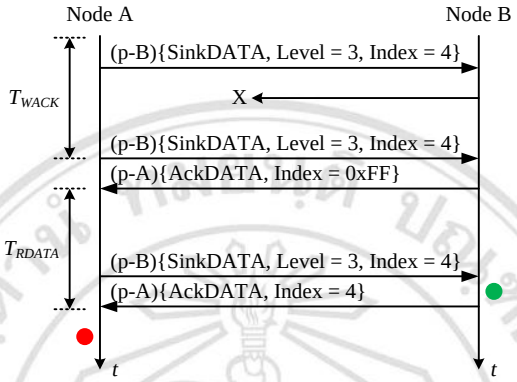


ภาพที่ 3.18 การส่งข้อมูลที่กระตุ้นด้วยเหตุการณ์ โดยใช้เฟรม SinkDATA/AckDATA

อาศัยตัวอย่างโทโพโลยีจากภาพที่ 3.18 สมมติให้โหนด A เป็นต้นกำเนิดแพคเกต และเริ่มส่งเฟรม SinkDATA ตามภาพที่ 3.18 ภายในเฟรมประกอบด้วยฟิลด์ Level = 3 (ของโหนด A) ฟิลด์ Index ซึ่งเป็นตัวเลขที่โหนด A สังเคราะห์ขึ้นและเพิ่มค่าขึ้นทุกครั้งเมื่อส่งแพคเกตสำเร็จ Index มีค่าอยู่ในช่วง 0x00-0x7F, เมื่อโหนด B รับแพคเกตจากโหนด A และมีที่ว่างในบัฟเฟอร์ที่จะเก็บแพคเกตของนั้นได้ มันจะตอบกลับด้วยเฟรม AckDATA ที่มีฟิลด์ Index ตรงกับ Index ของเฟรม SinkDATA ที่โหนด A ส่งมา, ในฝั่งของโหนด A เมื่อได้รับเฟรม AckDATA ด้วย Index ที่ตรงกัน มันจะถือว่าแพคเกตนั้นได้ถูกส่งไปอย่างเรียบร้อยแล้ว และมันจะนำออกจากบัฟเฟอร์ของมัน ในขณะที่โหนด B ถือว่าแพคเกตที่มันได้รับนั้น กลายเป็นแพคเกตของมันไปแล้ว และมีหน้าที่ที่จะต้องส่งต่อไปให้อัพลิงค์ของมัน กลไกการส่งต่อจากโหนด B ไปยังโหนด D จะมีลักษณะเดียวกันกับที่กล่าวไปแล้ว และด้วยกลไกนี้จะทำให้ทุกแพคเกตที่เกิดขึ้น สุดท้ายจะถูกส่งไปถึง โคออร์ดิเนเตอร์

ฟิลด์ Level ที่ใส่ไว้ในเฟรม SinkDATA ใช้สำหรับตรวจสอบความผิดพลาดของเครือข่าย หากโหนดได้รับเฟรม SinkDATA ที่มี Level เท่ากันหรือน้อยกว่ามันจะออกจากเครือข่ายทันทีเข้าสู่สเตต “Idle” และส่งเฟรม ExitIN ออกไป และรอเวลากลับเข้าเครือข่ายใหม่

ในกรณีที่บัฟเฟอร์ของอัพลิงค์เต็มไม่พร้อมที่จะรับแพคเกตได้ มันจะตอบด้วยเฟรม AckDATA ด้วยค่า Index = 0xFF, โหนดที่เป็นฝ่ายส่งแพคเกต จะรอเวลา T_{RDATA} แล้วค่อยส่งเพคเกตเดิมไปอีกครั้ง , ในทำนองเดียวกัน ถ้าโหนดฝ่ายส่งไม่ได้รับเฟรม AckDATA ภายในเวลา T_{WACK} มันก็จะส่งแพคเกตเดิมอีกครั้ง ดูตัวอย่างในภาพที่ 3.19



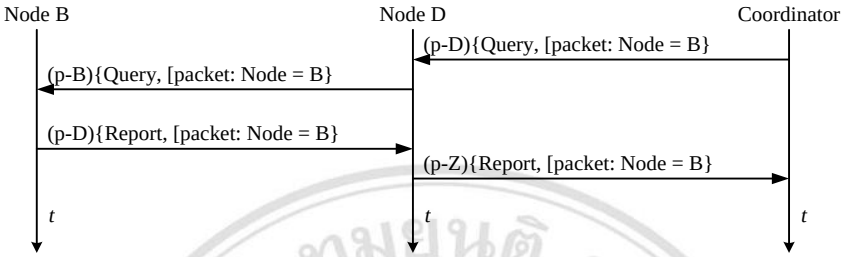
ภาพที่ 3.19 การส่งข้อมูลที่กระตุ้นด้วยเหตุการณ์ กรณีบัฟเฟอร์เต็มและไม่ได้รับเฟรมตอบรับ

3.3.7 การรับส่งข้อมูลแบบทวงถาม

โคออร์ดิเนเตอร์สามารถทวงถามข้อมูลจากโหนดใดโหนดหนึ่งภายในเครือข่ายได้ หรือ ส่งข้อมูลไปให้โหนดก็ได้เช่นกัน โดยส่งเฟรม Query ออกไป และโหนดจะตอบกลับด้วยเฟรม Report, การควบคุม Flow control เป็นหน้าที่ของโคออร์ดิเนเตอร์เอง โหนดที่อยู่ระหว่างทางเป็นผู้ส่งต่อเฟรมทั้งสองเท่านั้น พิจารณาตัวอย่างในภาพที่ 3.20 และ โทโพโลยีในภาพที่ 3.8 โคออร์ดิเนเตอร์ต้องการข้อมูลจากโหนด B มันจึงส่งเฟรม Query ผ่านโหนด D, เมื่อโหนด D ได้รับเฟรม Query ซึ่งภายในฟิดล์ [packet] จะมีฟิลด์ย่อย Node ที่ระบุเป้าหมายว่าเป็นโหนด B มันจะเปิดตารางส่งต่อ เพื่อหา Next hop ที่จะไปยังโหนด B จากนั้นก็ส่งเฟรม Query ต่อไปยังโหนด B, เมื่อโหนด B ได้รับเฟรม Query และตรวจสอบแล้วว่าเฟรมนั้นมีเป้าหมายเป็นตัวมันเอง จึงรับเฟรมนั้น แล้วส่งเฟรมนั้นขึ้นไปบริการชั้นแอพพลิเคชัน

เมื่อชั้นแอพพลิเคชันของโหนด B บริการเฟรม Query เรียบร้อยแล้ว มันอาจจะตอบกลับไปยังโคออร์ดิเนเตอร์ด้วยเฟรม Report หรือไม่ขึ้นอยู่กับ [packet] นั้นๆ จากตัวอย่างโหนด B จะตอบกลับไปยังโคออร์ดิเนเตอร์ด้วยเฟรม Report โดยผ่านโหนด D, เมื่อได้รับเฟรม Report โหนด D จะส่งต่อไปยังอัพลิงค์ของมันทันที

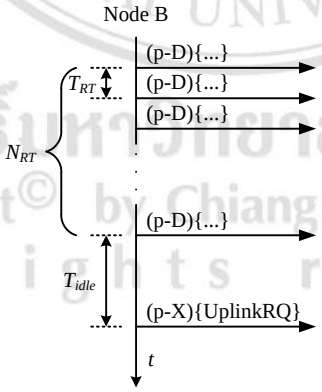
การส่งเฟรม Query เป็นการช่วยตรวจสอบ การมีอยู่ของโหนดด้วย ถ้าหากโหนดปลายทาง หรือ โหนดใดโหนดหนึ่งระหว่างทาง เกิดหายไป (ไม่ทำงาน) ก็จะเกิดเฟรม Unknown ขึ้นมาแจ้งโคออร์ดิเนเตอร์



ภาพที่ 3.20 การส่งข้อมูลที่แบบทวงถาม

3.3.8 การซ่อมแซมเครือข่าย

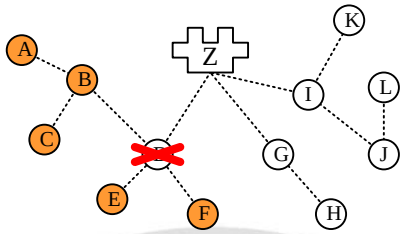
พิจารณากรณีที่อัพลิงค์หายไป (ไม่ทำงาน) โหนดลูกจึงไม่สามารถส่งข้อมูลไปยังโคออร์ดิเนเตอร์ได้ ตัวอย่างภาพที่ 3.21 โหนด B มีโหนด D เป็นอัพลิงค์ และโหนด D ไม่ทำงาน เมื่อโหนด B ส่งเฟรมใดๆ ที่มี TM=Up ไปให้โหนด D และไม่ได้รับ Acknowledgement มันจะพยายามส่งเฟรมนั้นใหม่อีกครั้งเมื่อเวลาผ่านไป T_{RT} และถ้าส่งไปอีก N_{RT} ครั้งแล้วยังไม่ได้รับ Acknowledgement โหนด B จะถือว่าโหนด D ไม่สามารถเป็นอัพลิงค์ให้มันได้อีกต่อไป มันจึงกระจายเฟรม ExitIN ออกอากาศ และเข้าสู่สเตต Idle เป็นระยะเวลา T_{Idle} จากนั้นโหนด B จะเริ่มขบวนการค้นหาอัพลิงค์ใหม่



ภาพที่ 3.21 กรณีโหนดอัพลิงค์ไม่ทำงาน

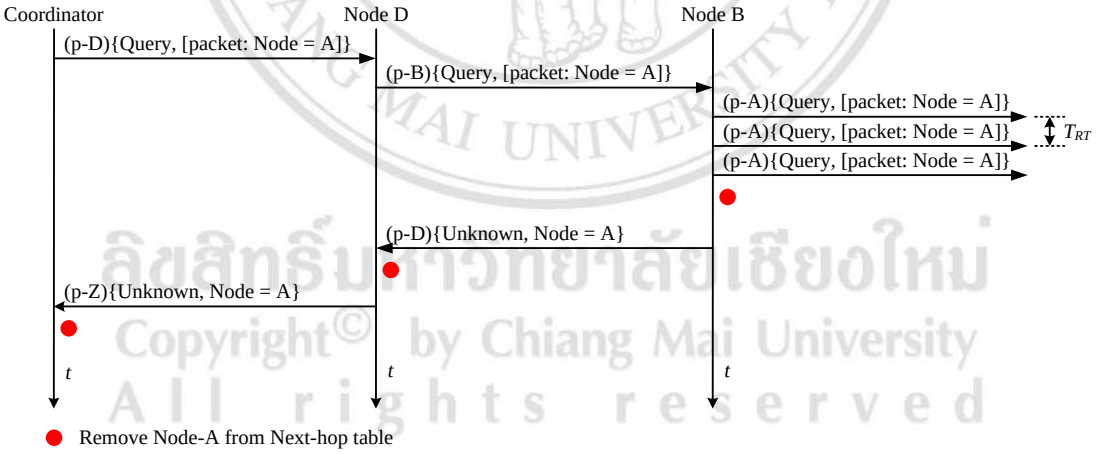
สำหรับโหนดลูกของโหนด B เมื่อได้รับเฟรม ExitIN จากโหนด B มันก็จะกระจายเฟรม ExitIN ออกอากาศ และเข้าสู่สเตต Idle เป็นระยะเวลา T_{Idle} เช่นเดียวกัน เมื่อเป็นเช่นนี้ การที่โหนด D หายไปจะทำให้โหนดทั้งหมดที่อยู่ใต้มัน จะออกจากเครือข่ายและเริ่มหาเครือข่ายใหม่ โดยที่โหนดที่

มี Level ต่ำจะออกก่อน และ เริ่มกระบวนการค้นหาอัพลิงค์ใหม่ก่อน เช่นกัน พิจาณาตัวอย่างในภาพที่ 3.22โหนด D หายไป ทำให้โหนด A, B, C, E, F ต้องค้นหาอัพลิงค์ใหม่



ภาพที่ 3.22 การซ่อมแซมเครือข่ายกรณีโหนดไม่ทำงาน

กรณีที่โหนดปลายทางของเฟรมที่ TM=Single ไม่ทำงาน พิจารณาตัวอย่างโทโพโลยีในภาพที่ สมมติให้ โหนด A หยุดทำงานไปแล้ว โคออร์ดิเนเตอร์ส่งเฟรม Query ออกไปหาตาม ตัวอย่างลำดับเฟรมในภาพที่ 3.23 เมื่อโหนด B ส่งเฟรมไปหา โหนด A แล้วไม่ได้ Acknowledge-ment กลับมา มันจะรอเวลา T_{RT} แล้วจะส่งไปใหม่อีก 2 หาไม่ได้รับ Acknowledgement โหนด B จะถือว่าโหนด A ไม่อยู่ในเครือข่ายแล้ว และจะถอดโหนด A ออกจากตารางส่งต่อ จากนั้นจะส่งเฟรม Unknown กลับไปแจ้งอัพลิงค์ของมันคือโหนด D, โหนด D ก็จะถอดโหนด A ออกจากตารางส่งต่อเช่นกัน และ ส่งเฟรม Unknown ไปยังโคออร์ดิเนเตอร์ เพื่อให้ถอดโหนด A ออกจากตาราง



ภาพที่ 3.23 กรณีโหนดปลายทางไม่ทำงาน

กรณีที่โคออร์ดิเนเตอร์ตัดสินใจที่จะล้างเครือข่ายเดิมทิ้งเนื่องจากเหตุใดก็ตาม และ สร้างเครือข่ายขึ้นมาใหม่ โคออร์ดิเนเตอร์สามารถทำได้โดยกระจายเฟรม FloodPAN ที่ FCommand = Realign ออกไป โดยระบุ Delay และ TH-LQI (ดูตารางที่ 3.2) ในกรณีเมื่อโหนดได้รับคำสั่งที่กระจายมามันจะไม่ออกเครือข่ายทันที แต่จะรอเวลาให้ผ่านไปเท่ากับที่ระบุไว้ใน Delay จากนั้นก็จะเข้าสู่สเตต FindUplink เมื่อเป็นเช่นนี้ ทุกโหนดจะเข้าสู่สเตต FindUplink เกือบจะพร้อมๆ กัน แต่โหนดที่

อยู่ใกล้โคออร์ดิเนเตอร์จะเข้าร่วมเครือข่ายโดยมีโคออร์ดิเนเตอร์เป็นอัพลิงค์ และพวกมันจะมี Level = 1 แล้วเข้าสู่สเตต Join และพร้อมที่จะเป็นอัพลิงค์ ให้โหนดที่อยู่ออกไปเรื่อยๆ ได้, ด้วยกระบวนการเช่นนี้ เครือข่ายจะค่อยๆ ถูกสร้างขึ้นมาใหม่ โดยทยอยสร้างออกไปจากโคออร์ดิเนเตอร์เป็นศูนย์กลาง