รายงานวิจัยฉบับสมบูรณ์

โครงการ แนวทางการโปรแกรมเชิงมหภาคแบบใหม่สำหรับเครือข่ายไร้สายของระบบฝังตัว

โดย ผศ.ดร. เฉลิมเอก อินทนากรวิวัฒน์

มกราคม 2554

สัญญาเลขที่ MRG5080449

รายงานวิจัยฉบับสมบูรณ์

โครงการ แนวทางการโปรแกรมเชิงมหภาคแบบใหม่สำหรับเครือข่ายไร้สายของระบบฝังตัว

ผศ.ดร. เฉลิมเอก อินทนากรวิวัฒน์

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

สนับสนุนโดยสำนักงานคณะกรรมการการอุดมศึกษา

และสำนักงานกองทุนสนับสนุนการวิจัย

กิตติกรรมประกาศ

ผู้วิจัยขอขอบคุณ ศ.ดร.ประภาส จงสถิตย์วัฒนา Prof. Rajesh Gupta Prof. Amin Vahdat Prof. Yoshito Tobe ตลอดจนนักวิจัยทุกท่านในงานนักวิจัยรุ่นใหม่พบเมธีวิจัยอาวุโส สกว. สำหรับคำแนะนำและความคิดเห็นต่อโครงการวิจัยนี้ นอกจากนี้ ผู้เขียนขอขอบคุณครอบครัวทุกคนสำหรับเวลา กำลังใจและความเข้าใจสำหรับการทำงานชิ้นนี้

บทคัดย่อ

รหัสโครงการ: MRG5080449

ชื่อโครงการ: แนวทางการโปรแกรมเชิงมหภาคแบบใหม่สำหรับเครือข่ายไร้สายของระบบฝังตัว

ชื่อนักวิจัย: ผศ.ดร. เฉลิมเอก อินทนากรวิวัฒน์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

E-mail Address: intanago@cp.eng.chula.ac.th

ระยะเวลาโครงการ: 2 ปี

การโปรแกรมเครือข่ายไร้สายของระบบฝังตัวนั้นขึ้นชื่อว่ามีความยากลำบากและซ้ำซากในหลายด้าน เราจึงขอนำเสนอแนวทางใหม่สำหรับการโปรแกรมเครือข่ายไร้สายของระบบฝังตัวแบบองค์รวม (การโปรแกรมเชิงมหภาค) แทนแบบองค์ประกอบที่เชื่อมต่อกันเป็นเครือข่ายหลายๆองค์ประกอบ เพื่อให้การโปรแกรมเครือข่ายไร้สายของระบบฝังตัวทำได้ง่ายขึ้น แนวทางของเราอนุญาตให้กลุ่มของทรัพยากรถูกบรรยายเชิงประกาศโดยคุณสมบัติในขณะที่ทำงาน และให้กลุ่มของทรัพยากรนี้ถูกแมปไปเป็นตัวแปรตัวหนึ่ง การใช้แนวทางนี้จะทำให้การเข้าถึงทรัพยากรง่ายลงจนเป็นเพียงแค่การเข้าถึงตัวแปรซึ่งเป็นอิสระจากเครือข่ายอย่างสมบูรณ์แบบ แนวทางของเราสนับสนุนการเข้าถึงกลุ่มทรัพยากรที่ต้องการได้ทั้งแบบลำดับและแบบขนาน การเข้าถึงแบบขนานหรือแบบกลุ่มช่วยลดเวลารวมในการเข้าถึงและลดการใช้พลังงาน เพราะการเข้าถึงแบบนี้ทำให้เกิดการประมวลผลในเครือข่ายได้ นอกจากนี้ เราสามารถเชื่อมโยงกลุ่มทรัพยากรแต่ละกลุ่มเข้ากับพารามิเตอร์ปรับค่าให้สอดคล้องกัน (เช่น เมื่อใดเวลาหมดในการเข้าถึงทรัพยากร งบด้านพลังงาน) เพื่อจำกัดเวลาในการเข้าถึงหรือปรับการใช้ทรัพยากร

คำหลัก: โปรแกรมเชิงมหภาค เครือข่ายไร้สายของระบบฝังตัว เครือข่ายเซ็นเซอร์ไร้สาย

Abstract

Project Code: MRG5080449

Project Title: A Novel Macro-programming Approach for Wireless Networks of Embedded Systems

Investigator: Asst. Prof. Chalermek Intanagonwiwat

E-mail Address: intanago@cp.eng.chula.ac.th

Project Period: 2 Years

Programming Wireless Networks of Embedded Systems (WNES) is notoriously difficult and tedious. To simplify WNES programming, we propose a novel approach for programming WNES as a whole (i.e., macro-programming) instead of several networked entities. Our approach allows for a set of resources to be declaratively described by their run-time properties, and for this set to be mapped to a variable. Using this approach, resource access is simplified to only variable access that is completely network-transparent. Our approach provides both sequential and parallel accesses to the desired set. Parallel, or group, access reduces the total access time and energy consumption because it enables in-network processing. Additionally, we can associate each set with tuning parameters (e.g., timeout, energy budget) to bound access time or to tune resource consumption.

Keywords: Macro-programming, Wireless Networks of Embedded Systems, and Wireless Sensor Networks

สารบัญ

กิตติกรรมประกาศ 1

บทคัดย่อ 2

Abstract 3

สารบัญ 4

บทที่ 1 บทนำ 6

บทที่ 2 การโปรแกรมเชิงมหภาค 8

2.1 รูปแบบการเขียนโปรแกรมที่เหมาะสม 8

2.2 การเรียกชื่อทรัพยากรเชิงประกาศ 10

2.3 ตัวแปรทรัพยากร 10

2.4 การจำกัดเชิงประกาศ 11

2.5 การเข้าถึงทรัพยากร 12

2.6 การผูกทรัพยากร 13

2.7 เวลาสิ้นสุดการเข้าถึง 16

บทที่ 3 การพัฒนาระบบ 18

3.1 สถาปัตยกรรม Smart Message 18

3.2 การพัฒนา DRN บน SM 19

บทที่ 4 การทดลองและการประเมินผล 22

4.1 วัตถุประสงค์ ตัววัด และวิธีการ 22

4.2 โปรแกรมติดตามวัตถุ..... 23

4.3 การปรับแต่ง 29

4.4 การเพิ่มประสิทธิภาพโดยการระบุบริเวณ 32

4.5 ผลกระทบจากรัศมีของพื้นที่วงกลมที่ระบุ 35

บทที่ 5 งานวิจัยที่เกี่ยวข้อง 37

บทที่ 6 บทสรุป 40

เอกสารอ้างอิง 41

Output จากโครงการวิจัยที่ได้รับทุนจาก สกอ. และ สกว. 44

บทที่ 1

บทนำ

เครือข่ายไร้สายของระบบฝังตัวประกอบด้วยโหนดไร้สายจำนวนมาก แต่ละโหนดมีทรัพยากรจำกัด ถูกติดตั้งใช้งานในสภาวะแวดล้อมที่ไม่เป็นมิตรและมีพลวัตรสูง เครือข่ายไร้สายของระบบฝังตัวแตกต่างจากเครือข่ายทั่วไปตรงที่เครือข่ายของระบบฝังตัวมีลักษณะมุ่งเน้นที่คุณสมบัติ ดังนั้น โหนดที่อยู่ในความสนใจในเครือข่ายไร้สายของระบบฝังตัวจึงถูกกำหนดโดยคุณสมบัติของโหนดในขณะปฏิบัติงานแทนที่จะกำหนดโดยหมายเลขโหนด คุณลักษณะเหล่านี้ก่อให้เกิดความท้าท้ายในแง่การวิจัยเรื่องการออกแบบการโปรแกรมเครือข่ายไร้สายของระบบฝังตัวอยู่สองความท้าทายหลัก

1. เราจะเขียนโปรแกรมประยุกต์เครือข่ายไร้สายของระบบฝังตัวให้ง่ายและมีประสิทธิภาพได้อย่างไร
2. เราจะโปรแกรมเครือข่ายใหม่อีกครั้งได้อย่างไร เมื่อโหนดทั้งหลายปราศจากผู้ดูแลควบคุมหลังการติดตั้งใช้งาน

ในโครงการวิจัยนี้ เราจะมุ่งเน้นการตอบคำถามข้อแรกในเรื่องการกำหนดคุณลักษณะของโปรแกรมประยุกต์ในเครือข่ายไร้สายของระบบฝังตัวเป็นหลัก เราเสนอวิธีการเรียกชื่อทรัพยากรเชิงประกาศ หรือ Declarative Resource Naming (DRN) เพื่อโปรแกรมเครือข่ายไร้สายของระบบฝังตัวแบบองค์รวม (การโปรแกรมเชิงมหภาค) แทนการโปรแกรมระดับโหนด วิธีของเราทำให้ผู้เขียนโปรแกรมสามารถบรรยายกลุ่มของทรัพยากร (โหนด) ที่สนใจได้ในเชิงประกาศ โดยระบุคุณสมบัติในขณะทำงานและแมปกลุ่มโหนดนี้ไปยังตัวแปรตัวหนึ่ง การเข้าถึงโหนดที่ต้องการสามารถทำได้อย่างง่ายดายโดยการอ้างอิงถึงตัวแปรที่แมปไว้ ดังนั้นการเข้าถึงโหนดจะถูกทำให้ง่ายลงเหลือเป็นเพียงการเข้าถึงตัวแปรซึ่งเป็นอิสระอย่างสมบูรณ์จากเครือข่าย แนวทางของเราสนับสนุนการเข้าถึงกลุ่มที่ต้องการทั้งแบบลำดับและแบบขนาน การเข้าถึงแบบขนานช่วยลดเวลารวมในการเข้าถึงและลดการใช้พลังงานอีกด้วย เนื่องจากการเข้าถึงแบบขนานทำให้เราสามารถสรุปรวมข้อมูลในเครือข่ายได้ นอกจากนี้ เราสามารถเชื่อมโยงแต่ละกลุ่มโหนดเข้ากับพารามิเตอร์ปรับค่า เช่น เวลาการทำงาน พลังงานที่ใช้ได้ เพื่อจำกัดเวลาในการเข้าถึง หรือ ปรับการใช้ทรัพยากรระบบ

เนื่องจากเครือข่ายไร้สายของระบบฝังตัวอาจถูกติดตั้งใช้งานในสภาวะแวดล้อมที่ไม่เป็นมิตรและมีพลวัตรสูง อีกทั้งเราอาจจะไม่สามารถเข้าไปแตะต้องโหนดทางกายภาพได้ เราจึงจำเป็นต้องสามารถโปรแกรม

โหนดที่ปราศจากผู้ควบคุมดูแลนี้ได้จากระยะไกล แพลตฟอร์มที่มีพื้นฐานของการโยกย้ายโปรแกรมระหว่างโหนดได้จึงมีความเหมาะสมสำหรับโครงการนี้เพราะโปรแกรมสามารถถูกส่งต่อไปยังโหนดเป้าหมายได้โดยไม่ต้องให้มนุษย์เข้าไปช่วยเหลือ ตัวอย่างของแพลตฟอร์มดังกล่าวคือ Smart Messages (SM) [3], SensorWare [4], Mate [18], และ TML[20] ถึงแม้ว่าการโปรแกรมเครือข่ายได้ใหม่อีกครั้งหลังการติดตั้งใช้งานจะไม่ใช่จุดประสงค์ของงานเรา แต่เรายังคงเลือกออกแบบและพัฒนาระบบของเราอยู่บนแพลตฟอร์มดังกล่าวสองแพลตฟอร์มคือ Smart Messages และ Mate เพื่อให้ระบบของเราทำงานได้บนทั้ง iPAQ และ Mote อีกทั้งยังสามารถโปรแกรมได้ใหม่หลังการติดตั้งใช้งานได้อีกด้วย

นอกจากนี้ เรายังพัฒนาโปรแกรมประยุกต์ในการติดตามวัตถุอย่างง่ายโดยใช้แนวทางการโปรแกรมเชิงมหภาคเพื่อแสดงให้เห็นว่าแนวทางพัฒนาโปรแกรมแบบใหม่นี้สามารถนำไปพัฒนาโปรแกรมประยุกต์ได้จริง อีกทั้งเรายังประเมินแนวทางการโปรแกรมนี้และพารามิเตอร์ปรับค่า (อายุขัยของการผูกทรัพยากร) บนเครือข่ายของเครื่องเสมือน 20 เครื่อง เมื่อโปรแกรมถูกแคชหรือติดตั้งในเครือข่ายแล้ว การปรับค่าพารามิเตอร์ให้เหมาะสมจะสามารถช่วยให้โปรแกรมที่พัฒนาขึ้นประหยัดการส่งข้อมูลลงได้ถึง 55.2% โดยที่ไม่ทำให้ความแม่นยำลดลงอย่างมีนัยสำคัญ

บทที่ 2

การโปรแกรมเชิงมหภาค

2.1 รูปแบบการเขียนโปรแกรมที่เหมาะสม

โดยทั่วไปแล้วมีรูปแบบในการเขียนโปรแกรมหลักอยู่สองรูปแบบในวงการคอมพิวเตอร์ นั่นคือ การโปรแกรมเชิงประกาศ (Declarative Programming) และการโปรแกรมเชิงบังคับ (Imperative Programming) สำหรับการโปรแกรมเชิงประกาศนั้นจะ Abstract รายละเอียดของอัลกอริธึ่มได้ทั้งหมด โปรแกรมเมอร์เพียงแค่ระบุสิ่งที่ตนต้องการเท่านั้นโดยไม่ต้องระบุว่าจะทำอย่างไรจึงจะได้ผลลัพธ์มา ตัวแปลภาษาจะทำการเติมอัลกอริธึ่มที่จำเป็นต้องใช้ให้เอง การสร้างรายละเอียดในเชิงอัลกอริธึ่มให้อย่างอัตโนมัติเช่นนี้อาจมีประสิทธิภาพได้สำหรับงานง่ายๆเฉพาะอย่างเท่านั้น เช่น ฐานข้อมูล แต่อาจทำได้ไม่ดีนักในงานประเภทอื่น ตัวอย่างของแนวทางการเขียนโปรแกรมเชิงประกาศมีอยู่บ้างพอสมควร เช่น งานที่มีพื้นฐานมาจากภาษา SQL ได้แก่ Cougar [1] และ TAG [19] ถึงแม้ว่าการโปรแกรมเชิงประกาศจะมีข้อดีในแง่ของความง่ายในการเขียนโปรแกรม แต่ก็ไม่ได้หมายความว่าการโปรแกรมเชิงประกาศจะเหมาะสมสำหรับทุกโปรแกรมประยุกต์ในเครือข่ายไร้สายของระบบฝังตัวแต่อย่างใด

การโปรแกรมเชิงบังคับอาจจะมีความเหมาะสมมากกว่าโดยเฉพาะอย่างยิ่งในงานที่มีความซับซ้อน ซึ่งไม่อาจสร้าง Code ที่มีประสิทธิภาพให้ได้อย่างอัตโนมัติ หรือไม่อาจสร้างให้ได้โดยง่าย ยกตัวอย่างเช่น คงเป็นไปไม่ได้หรือยากมากที่จะใช้ภาษา SQL ในการเขียนหรือสร้าง Kalman Filter หรืออัลกอริธึ่ม Maximum Likelihood สำหรับการประมาณตำแหน่งวัตถุ เป็นต้น เพราะว่าภาษา SQL ไม่ได้ถูกออกแบบมาเพื่อบรรยายรายละเอียดของอัลกอริธึ่ม

การโปรแกรมเชิงประกาศและการโปรแกรมเชิงบังคับต่างก็มีข้อดีคนละด้าน หากใช้ให้เหมาะสม นอกจากนี้ การโปรแกรมทั้งสองแบบสามารถส่งเสริมกันได้ เพราะต่างทำงานได้ดีในด้านที่เป็นข้อด้อยของอีกฝ่ายหนึ่ง หากใช้ร่วมกันน่าจะสามารถกำจัดข้อด้อยของทั้งสองแบบลงได้ การบูรณาการการโปรแกรมเชิงประกาศและเชิงบังคับเข้าด้วยกันน่าที่จะก่อให้เกิดแนวทางการเขียนโปรแกรมที่ทรงพลังเหมาะสมในการ

โปรแกรมหลากหลายด้าน ในโครงการนี้ เราเสนอว่าการบูรณาการดังกล่าวนั้นทำได้ถ้าการโปรแกรมเชิงประกาศนั้นถูกใช้เฉพาะในบางส่วนของโปรแกรมเท่านั้น ไม่ใช่ทั้งหมด

โดยทั่วไปแล้ว ส่วนของโปรแกรมที่ควร Abstract คือ

1. ส่วนที่ไม่เกี่ยวข้องกับโปรแกรมหลัก
2. ส่วนที่ต้องมีในโปรแกรมประยุกต์ทั่วไป
3. ส่วนที่ซ้ำซากน่าเบื่อหน่ายสำหรับโปรแกรมเมอร์

การจะระบุว่าส่วนไหนของโปรแกรมน่าที่จะต้องทำการ Abstract ได้ จำเป็นอย่างยิ่งที่จะต้องทำความเข้าใจโปรแกรมประยุกต์ของเครือข่ายไร้สายของระบบฝังตัวให้ดีเสียก่อน โดยปกติแล้ว โปรแกรมเป็นชุดของการกระทำการบนตัวแปรและทรัพยากร เนื่องจากตัวแปรมักจะถูกเข้าถึงบ่อยครั้งกว่า ภาษาสำหรับการเขียนโปรแกรมจึงมักมีกระบวนการในการเข้าถึงตัวแปรที่ง่ายกว่าการเข้าถึงทรัพยากร

จึงไม่น่าแปลกใจนักที่ การเข้าถึงทรัพยากรโดยปกติจะยุ่งยากกว่า โดยเฉพาะอย่างยิ่งในระบบเครือข่ายที่แยกความแตกต่างชัดเจนระหว่างทรัพยากรของโหนดเอง และทรัพยากรของโหนดอื่นที่อยู่ไกลออกไป ตามธรรมดาแล้วทรัพยากรมักจะถูกผูกเข้ากับโหนดไว้ตั้งแต่ต้น ดังนั้นในการระบุทรัพยากรของโหนดอื่นจำเป็นต้องใช้หมายเลขเครื่องของโหนดนั้นร่วมด้วย ถ้าเราไม่รู้หมายเลขเครื่อง เรามักจำเป็นต้องทำกระบวนการค้นหาทรัพยากร ด้วยเหตุนี้ โปรแกรมเมอร์จึงต้องจัดการในรายละเอียดโปรแกรมจำนวนมาก ไม่ว่าจะเป็นการเชื่อมต่อ การค้นหาทรัพยากร และการเข้าถึงทรัพยากร เป็นต้น

หากเป็นเครือข่ายไร้สายของระบบฝังตัว ความยุ่งยากนี้ยิ่งมีมากขึ้นเพราะว่าทรัพยากรที่สนใจจะถูกระบุโดยคุณสมบัติในขณะทำงานแทนที่จะเป็นหมายเลขเครื่อง ยกตัวอย่างเช่น เราอาจต้องการเข้าถึงเซ็นเซอร์ที่วัดอุณหภูมิได้เกิน 30 องศาเซลเซียสเท่านั้น ในกรณีนี้ การค้นหาทรัพยากรในเครือข่ายไร้สายของระบบฝังตัวเป็นเรื่องจำเป็น ปกติสามัญ แทนที่จะเป็นเรื่องที่เลือกที่จะทำหรือไม่ก็ได้ คุณสมบัติของทรัพยากรนั้นค่อนข้างมีพลวัตรสูงเพราะสภาวะแวดล้อมอาจจะมีอุณหภูมิสูงขึ้นหรือลดลงเมื่อใดก็ได้ อาจมีความแปรปรวนสูง รวมทั้งเป็นสภาวะแวดล้อมที่ไม่เป็นมิตร การผูกหรือแมป ทรัพยากรอาจถูกต้องในขณะหนึ่ง เมื่อเวลาผ่านไปอาจไม่ถูกต้องแล้ว เพราะว่าทรัพยากรที่ตรงตามคุณสมบัติในขณะหนึ่ง อาจไม่ตรงตามคุณสมบัติอีกต่อไป

แต่ว่าต่อให้คุณสมบัติของทรัพยากรจะไม่เปลี่ยนแปลงก็ตาม ทรัพยากรที่ผูกไว้ก็อาจจะไม่สามารถเข้าถึงได้เพราะว่าพลวัตรของเครือข่าย เช่น การเคลื่อนที่ของโหนด เป็นต้น โปรแกรมสำหรับเครือข่ายไร้สายของระบบฝังตัวจะต้องสามารถจัดการกับการเปลี่ยนแปลง การผูกที่ไม่ถูกต้องอีกต่อไป การค้นหาทรัพยากรอื่นที่เท่าเทียมกันมาผูกแทน การผูกทรัพยากรใหม่ที่เพิ่งค้นพบและตรงตามคุณสมบัติที่ต้องการ เรื่องจุกจิกที่เกิดขึ้นได้บ่อยเหล่านี้ได้ เรื่องเหล่านี้ไม่ว่าจะเป็นการค้นหา การเข้าถึง การผูกใหม่ และการเชื่อมต่อ ล้วนแล้วแต่ซ้ำซากน่าเบื่อหน่ายสำหรับโปรแกรมเมอร์ ดังนั้น ส่วนของโปรแกรมที่เกี่ยวข้องกับทรัพยากรจึงเป็นตัวเลือกที่สมเหตุสมผลสำหรับการ Abstraction แบบเชิงประกาศของเรา

2.2 การเรียกชื่อทรัพยากรเชิงประกาศ

เพื่อทำให้การเขียนโปรแกรมสำหรับเครือข่ายไร้สายของระบบฝังตัวทำได้ง่ายขึ้น เราขอเสนอวิธีที่โปรแกรมเครือข่ายเป็นเสมือนหน่วยเดียวกัน โดยในที่นี้เราจะมองเสมือนว่าเครือข่ายทั้งเครือข่ายเป็นเพียง Abstract Machine เครื่องหนึ่ง ถึงแม้ในทางกายภาพแล้ว ทรัพยากรจะอยู่กันอย่างกระจัดกระจาย ทรัพยากรเหล่านั้นทั้งหมดล้วนแล้วแต่อยู่บนเครื่องเดียวกันในโมเดลของเรานี้ ดังนั้นในโมเดลของเรา จึงไม่มีเครือข่าย ไม่มีการเชื่อมต่อ ไม่มีคำว่าใกล้ ไม่มีคำว่าไกล

2.3 ตัวแปรทรัพยากร

การโปรแกรมเครือข่ายไร้สายของระบบฝังตัวสามารถถูกทำให้ง่ายลงได้โดยการทำให้การเข้าถึงทรัพยากรง่ายเหมือนกับการเข้าถึงตัวแปร ดังนั้น เราจึงขอเสนอ ตัวแปรพิเศษชนิดใหม่ชื่อ ตัวแปรทรัพยากร (ตัวแปรที่ถูกแมปและอ้างอิงถึงโหนดจริง) ยกตัวอย่างเช่น เราสามารถเขียนโปรแกรมเพื่ออ่านค่าเซนเซอร์แสงและควบคุมกล้องได้ดังนี้

```
Resource R, X;
printf("light intensity=%f", R->light);
X->camera=off;
```

ในตัวอย่างข้างต้น เราสมมุติให้ตัวแปรทรัพยากร R แมปไปยังโหนดที่มีเซนเซอร์แสง และตัวแปรทรัพยากร X แมปไปยังโหนดที่มีกล้อง การอ่านค่าความเข้มแสงสามารถทำได้อย่างง่ายดายโดยเพียงแค่การอ้างอิงถึงเซนเซอร์แสงของ R คือ R->light ในทางที่คล้ายๆกัน เราสามารถปิดกล้องได้โดยแค่ใส่ค่า Off ให้กับกล้องของ X คือ X->camera ไม่จำเป็นต้องมีรายละเอียดอัลกอริธึ่มสำหรับการควบคุมและการกระทำการกับทรัพยากร ตัวอย่างนี้แสดงให้เห็นว่าแนวทางของเราไม่เพียงแต่สามารถใช้ในการดึงหรือส่งข้อมูลไปยังโหนดที่ต้องการได้เท่านั้น มันยังสามารถใช้ในการควบคุมโหนดเหล่านั้นได้อีกด้วย

2.4 การจำกัดเชิงประกาศ

คงเป็นที่เข้าใจได้ว่า บางท่านอาจสงสัยว่าโหนดทางกายภาพหรือทรัพยากรไหนจะถูกแมปหรือผูกเข้ากับตัวแปรทรัพยากรกันแน่ และโปรแกรมเมอร์จะรู้เกี่ยวกับชนิดของเซนเซอร์แต่ละตัวได้อย่างไร ในที่นี้แทนที่เราจะระบุหมายเลขโหนดเหมือนอย่างในอินเทอร์เนต เราจะทำในแนวทางอื่นที่เหมาะสมกับเครือข่ายไร้สายของระบบฝังตัวมากกว่า นั้นคือการระบุคุณสมบัติที่ต้องการของโหนดเป้าหมายแทนในเชิงประกาศด้วยเงื่อนไขในเชิงตรรก ยกตัวอย่างเช่น เราสามารถระบุได้ว่า R จะต้องถูกผูกไปยังโหนดที่มีเซนเซอร์แสงที่อยู่ในป่าซึ่งมีอุณหภูมิมากกว่า 30 องศาเซลเซียส

```
Resource R = <within(location,forest)&&
                temperature > 30 &&exist(light)>
Resource X = <a(b,c)!=0 && exist(camera)>
```

อย่างไรก็ตาม มีความเป็นไปได้อย่างสูงที่จะมีโหนดที่สอดคล้องตามคุณสมบัติข้างต้นอยู่เป็นจำนวนมาก ดังนั้นตัวแปรทรัพยากรที่แท้จริงจะถูกผูกกับเซตของโหนดที่สอดคล้องตามคุณสมบัติ มิได้ผูกอยู่กับโหนดใดโหนดหนึ่งแต่เพียงโหนดเดียว ตำแหน่งและอุณหภูมิเป็นคุณสมบัติท้องถิ่นของโหนดที่ถูกใช้ในการตัดสินว่าโหนดนั้นๆจะเป็นสมาชิกในเซตที่ผูกกับตัวแปร R หรือไม่ นอกจากนี้ เราสามารถใช้ฟังก์ชันตรรกะใดก็ได้ในเงื่อนไขเชิงประกาศ โดยผู้ใช้อาจสร้างฟังก์ชันตรรกะขึ้นมาเองก็ได้ เช่น ฟังก์ชัน a() ในตัวอย่างข้างต้น ข้อยืดหยุ่นดังกล่าวโดยทั่วไปแล้วทรงพลังมาก ควรที่จะเพียงพอต่อการกำหนดเงื่อนไขเชิงประกาศที่ซับซ้อนหลากหลายกว่าในตัวอย่างได้อีกมาก

2.5 การเข้าถึงทรัพยากร

ในที่นี้เราจะอธิบายถึงความจำเป็นที่ระบบของเราต้องสนับสนุนการเข้าถึงข้อมูลได้หลายแบบ สำหรับใช้ในสถานะการณ์ที่แตกต่างกัน โดยจะมีการวิเคราะห์ข้อดีและข้อเสียของแต่ละแบบไว้ให้ด้วยเพื่อเป็นแนวทางในการนำไปเลือกใช้ให้เหมาะสมกับงาน ในโครงการนี้ เราขอเสนอวิธี 2 วิธีในการเข้าถึงโหนดทั้งหลายที่สอดคล้องตามเงื่อนไขเชิงประกาศ คือแบบอนุกรมและแบบขนาน

- การเข้าถึงแบบอนุกรม ในแบบนี้โหนดแต่ละโหนดในเซตสามารถถูกอ้างอิงถึงได้โดยการใช้ Iterator (คล้ายกับ Iterator ใน C++ Standard Template Library) วิธีนี้ทำให้เราสามารถเข้าถึงโหนดที่สอดคล้องตามเงื่อนไขได้ทีละตัว ยกตัวอย่างเช่น เราสามารถอ่านค่าความเข้มแสงจากเซนเซอร์แสงของโหนดในเซต R ทีละตัวได้ตามลำดับดังนี้

```
Resource R;
Iterator i;
foreach i in R {
     printf("light intensity = %f\n", i->light);
}
```

อย่างไรก็ตาม การอ่านตามลำดับไม่สามารถให้ค่าที่เวลาเดียวกันของโหนดแต่ละตัวในเซตได้ เพราะความหน่วงเวลาในการเข้าถึงโหนดทุกตัวในเซตอาจมีนัยสำคัญสูง ทำให้การอ่านค่าของโหนดแรกเกิดขึ้นที่เวลาห่างจากเวลาขณะอ่านค่าของโหนดสุดท้ายมากได้ อย่างไรก็ตามการเข้าถึงแบบนี้ยังมีประโยชน์อยู่มาก เช่น การเข้าถึงโหนดแค่บางโหนดในเซต เป็นต้น

- การเข้าถึงแบบขนาน ในแบบนี้ โหนดทุกตัวในเซตจะถูกเข้าถึงพร้อมกัน การเข้าถึงแบบขนานทำได้โดยการอ้างอิงตัวแปรทรัพยากรโดยตรงดังนี้

```
Resource R;
printf("light intensity=%f", R->light);
```

ในตัวอย่างข้างต้น โปรแกรมจะแสดงค่าความเข้มของแสงของทุกโหนดในเซต R นี้ออกมาเป็นหน้าจอ ค่าความหน่วงเวลารวมทั้งหมดเมื่อใช้การเข้าถึงแบบขนานนี้จะลดลงเหลือเพียงแค่ค่าหน่วง

เวลาของการเข้าถึงโหนดตัวที่ใช้เวลามากที่สุด แนวทางนี้ไม่เพียงลดเวลาการเข้าถึงรวมเท่านั้น มันยังสามารถให้ค่าที่เวลาเดียวกันของโหนดแต่ละตัวในเซตได้ นอกจากนี้ แนวทางแบบขนานนี้ยังแตกต่างจากแนวทางแบบอนุกรมตรงที่แนวทางแบบขนานเปิดโอกาสให้มีการประมวลผลภายในเครือข่ายได้ เช่น การรวมข้อมูลในระหว่างการเดินทางของข้อมูลในเครือข่าย ซึ่งการประมวลผลในเครือข่ายนี้สามารถลดการใช้พลังงานของระบบลงได้อย่างมีนัยสำคัญ [10, 11, 13, 17, 14, 19] ตัวอย่างเช่น การใช้ฟังชัน max(A) เพื่อการหาค่ามากที่สุดในเซต A สามารถทำได้ดังนี้

```
Resource R;
printf("max light intensity = %f", max(R->light));
```

ในทางอุดมคติ ระบบควรจะส่งเฉพาะค่าที่มากที่สุดเท่านั้นเพื่อประหยัดพลังงาน โดยไม่เสียพลังงานไปในการส่งค่าๆอื่นเลย แต่ในทางปฏิบัติแล้ว การจะทำเช่นนั้นได้โหนดทุกตัวจะต้องรู้ว่าค่าที่อ่านได้ของตนเมื่อเทียบกับโหนดอื่นแล้วมีค่ามากที่สุดหรือไม่ หากมากสุดจึงส่งออกมา ซึ่งในความเป็นจริงแล้วถ้าโหนดไม่แลกเปลี่ยนค่ากันก่อน โหนดแต่ละตัวจะไม่สามารถล่วงรู้ค่าของโหนดอื่นได้ ในที่นี้เราสามารถทำให้การใช้พลังงานใกล้เคียงกับอุดมคติได้โดยการระงับการส่งหรือส่งต่อค่าที่น้อยกว่าค่าที่เคยพบมาก่อนหน้าของการเข้าถึงเดียวกัน ระหว่างการส่งต่อค่าภายในเครือข่าย การระงับการส่งต่อค่านี้จะไม่สัมฤทธิ์ผลหรืออาจเป็นไปไม่ได้ ถ้าเราทำการเข้าถึงข้อมูลแบบอนุกรมแทนที่จะเป็นแบบขนาน

2.6 การผูกทรัพยากร

โมเดลของเราสนับสนุนการผูกทรัพยากรสองประเภทคือ แบบพลวัตและแบบสถิต

- การผูกแบบพลวัต ในวิธีของเรานั้น โปรแกรมเมอร์ไม่มีความจำเป็นที่จะต้องเขียนโปรแกรมเพื่อบำรุงรักษาให้การผูกค่าระหว่างทรัพยากรทางกายภาพและตัวแปรทรัพยากรถูกต้องอยู่เสมอด้วยตนเอง เนื่องจากของคุณสมบัติของทรัพยากรเปลี่ยนแปลงค่อนข้างบ่อย จึงจำเป็นต้องมีการผูกค่าใหม่อยู่บ่อยมาก ยกตัวอย่างเช่น เซตของโหนด R ที่เวลา t1 อาจจะแตกต่างอย่างสิ้นเชิงกับเซตของโหนด R ที่เวลา t2

```
Resource R = <expression1>
Time t1 = get_time();
x=Count(R);
     ...
Time t2 = get_time();
y=Count(R);
/* Normally, x != y */
```

จากตัวอย่างข้างต้นจะเห็นว่า เราเพียงแค่กำหนดเงื่อนไขเชิงประกาศสำหรับตัวแปรทรัพยากรเพื่อบรรยายทรัพยากรที่สนใจเท่านั้น แล้วระบบเราจะทำการบำรุงรักษาให้การผูกค่านั้นถูกต้องเองถึงแม้ว่าจะมีการเปลี่ยนแปลงเกิดขึ้นก็ตาม โดยทั่วไปแล้ว การอ้างอิงถึงตัวแปรทรัพยากรจะส่อความหมายโดยนัยถึงการเข้าถึงทรัพยากร ซึ่งการเข้าถึงตัวแปรทรัพยากรในระบบเรานี้มีความหมายที่ค่อนข้างเข้มงวด กล่าวคือ การเข้าถึงทรัพยากรจะต้องกระทำการบนทรัพยากรที่มีเงื่อนไขตรงตามที่ประกาศไว้ ณ เวลาที่ทำการเข้าถึง โดยที่เซตของโหนดในตัวแปรทรัพยากรจะต้องเปลี่ยนแปลงได้เองให้มีสมาชิกคือโหนดที่ตรงตามเงื่อนไขในขณะนั้นเท่านั้นได้โดยอัตโนมัติ โปรแกรมเมอร์ไม่ต้องเขียนโปรแกรมเพื่อจัดการในเรื่องนี้เลย ดังนั้น การจะรักษาไว้ซึ่งความหมายที่เข้มงวดนี้ ระบบจะต้องใช้พลังงานหรือโอเวอร์เฮดอย่างมีนัยสำคัญ เพื่อให้การผูกค่านี้ถูกต้องอยู่ตลอดเวลา ดังนั้น เราจึงเสนอทางเลือก หรือ พารามิเตอร์ปรับค่า ซึ่งสามารถใช้เพื่อประหยัดพลังงานในส่วนนี้โดยแลกกับความหมายของการเข้าถึงที่เข้มงวดน้อยลง ยกตัวอย่างเช่น โปรแกรมเมอร์สามารถลดความเข้มงวดได้โดยอนุญาตให้มีการเข้าถึงโหนดที่ผูกไว้ไม่เกิน t วินาทีก่อน

```
Resource R = <expression, last_bound_time > now-t>
```

นอกจากนี้ โปรแกรมเมอร์สามารถระบุงบพลังงานสำหรับการจำกัดการใช้พลังงานในการเข้าถึงทรัพยากรได้

```
Resource R = <expression, energy_budget = 100>
```

พารามิเตอร์ดังกล่าวเป็นเพียงแค่ตัวอย่างเท่านั้น อาจมีพารามิเตอร์อื่นๆอีกมากที่เป็นไปได้ ซึ่งเราจะละไว้สำหรับการวิจัยในภายหน้า

- การผูกแบบสถิต ถึงแม้การผูกแบบพลวัตจะดูสมเหตุสมผลดี อย่างไรก็ตามในบางสถานการณ์เราอาจจำเป็นต้องใช้การผูกแบบสถิต ยกตัวอย่างเช่น เราอาจต้องการเข้าถึงทรัพยากรที่ตรงตาม

เงื่อนไขมาก่อนซึ่งปัจจุบันทรัพยากรเหล่านั้นไม่มีคุณสมบัติตรงตามเงื่อนไขแล้ว เราอาจทำการเปิดกล้องในพื้นที่ A ให้ทำงานไว้ เมื่อเวลาผ่านไปเราอาจต้องการปิดมัน แต่ว่ามีกล้องบางตัวได้ถูกเคลื่อนย้ายไปบริเวณอื่นแล้ว ถ้าพื้นที่ A เป็นเงื่อนไขในการผูกทรัพยากรของเรา กล้องที่ถูกเคลื่อนย้ายไปพื้นที่อื่นจะไม่ตรงตามเงื่อนไขอีกต่อไป ทำให้เราไม่สามารถปิดกล้องเหล่านั้นได้โดยอ้างอิงถึงตัวแปรทรัพยากรเดิมได้

หนึ่งในวิธีแก้สำหรับปัญหาข้างต้นคือ การพึ่งพาระบบให้ช่วยจำคุณสมบัติของโหนดทุกตัวตั้งแต่เริ่มต้นทำงานจนถึงปัจจุบัน ยกตัวอย่างเช่น เราสามารถประกาศตัวแปรทรัพยากรตัวใหม่ที่มีเงื่อนไขเรื่องเวลามาประกอบด้วย ดังนี้

```
Resource R = <expression1>;
Time t1 = get_time();
    ....
Resource X = <expression1 && time == t1>;
```

ตราบใดก็ตามที่เรารู้เวลาที่โหนดตรงตามเงื่อนไข เราย่อมสามารถอ้างอิงถึงมันอีกในอนาคตถึงแม้ว่ามันอาจไม่ตรงตามเงื่อนไขอีกแล้วได้ โดยการนำเวลาดังกล่าวมาผูกเป็นเงื่อนไขสำหรับตัวแปรทรัพยากรตัวใหม่สำหรับการเข้าถึงในภายหลังได้

วิธีแก้อีกวิธีที่คล้ายกันคือ การให้มีฟังก์ชัน last() ที่จะให้ค่าเซตที่แล้ว ที่ตรงตามเงื่อนไข ดังนั้น เราจะสามารถกระทำการบนเซตที่ต้องการได้ ถึงแม้ว่าในปัจจุบันเซตนั้นไม่ตรงตามเงื่อนไขอีกต่อไป เช่น

```
Resource R = <expression1>;
Resource X = last(R);
```

อย่างไรก็ตาม วิธีแก้ทั้งสองก่อให้เกิดโอเวอร์เฮดอย่างมาก เนื่องจากระบบจะต้องเก็บค่าการเปลี่ยนแปลงทุกอย่างของเซตไว้ตลอดเวลา

วิธีแก้ปัญหาอีกทางเลือกหนึ่งคือ การให้มีคำสั่งที่ระบุชัดเจนลงไปว่าให้จำเซตใดไว้ เราเสนอกลไกที่ระบุชัดเจนไว้อยู่ 2 กลไกคือ การใช้ตัวแปรทรัพยากรแบบสถิต และ การใช้ตัววนซ้ำ

ในการใช้ตัวแปรทรัพยากรแบบสถิตนั้น เราสามารถระบุว่าตัวแปรจะผูกกับค่าใดไว้โดยไม่เปลี่ยนแปลงไปตามเงื่อนไขอีกต่อไป จะไม่มีการผูกใหม่ไม่ว่าจะเกิดเหตุการณ์ใดขึ้นก็ตาม ดังนั้นเราสามารถเก็บเซตของโหนดใดใดไว้ได้ ถึงแม้ว่าในเวลาต่อมา มันจะไม่ตรงตามเงื่อนไขอีกแล้ว ดังตัวอย่าง

```
Resource R1;
Static Resource R2=R1;
/* R1 changes over time but R2 does not*/
```

คำสั่งที่ระบุชัดเจนเช่นนี้ จะมีโอเวอร์เฮดน้อยกว่า last() อย่างมากเพราะว่าระบบไม่ต้องเก็บค่าทุกค่าที่เคยตรงตามเงื่อนไขมาก่อนของทุกตัวแปรทรัพยากร ยิ่งกว่านั้น ตัวแปรทรัพยากรแบบสถิตนี้เราตั้งใจให้มันจดจำเซตของโหนดที่ตรงตามเงื่อนไข แต่หากเราต้องการจะจดจำเพียงแค่โหนดเดียว การใช้ตัววนซ้ำจะเหมาะสมกว่า โดยค่าของตัววนซ้ำจะไม่เปลี่ยนแปลงแบบอัตโนมัติโดยปราศจากการหมอบหมายค่าใหม่ให้ ดังตัวอย่างต่อไปนี้

```
Iterator i1 = R1->first_element;
```

2.7 เวลาสิ้นสุดการเข้าถึง

ไม่ว่าเราจะใช้การผูกค่าตัวแปรทรัพยากรแบบใดก็ตาม การเข้าถึงทรัพยากรจะสำเร็จหรือไม่ เป็นเรื่องที่ไม่สามารถรับประกันได้ร้อยเปอร์เซ็นต์ เวลาในการเข้าถึงทรัพยากรในเครือข่ายเซ็นเซอร์ไร้สายนั้นไม่สามารถกำหนดขอบเขตได้ การล้มเหลวของการเข้าถึงทรัพยากรเป็นเรื่องที่ไม่อาจหลีกเลี่ยงได้ในทางปฏิบัติ เนื่องจากสภาพที่เป็นพลวัตของเครือข่าย ดังนั้น เราไม่สามารถแยกแยะความแตกต่างระหว่างเหตุการณ์สองเหตุการณ์นี้ได้โดยง่าย เหตุการณ์สองเหตุการณ์ดังกล่าวนั้น คือ เหตุการณ์แรกเป็นเหตุการณ์ที่การเข้าถึงนั้นสำเร็จแต่ไม่สามารถกำหนดขอบเขตว่าใช้เวลาเท่าใด หากเรารอนานพอจะสามารถเข้าถึงได้ แต่อาจจะต้องรอนานมากแล้วไม่รู้ว่าจะต้องรอนานเท่าใด ส่วนเหตุการณ์ที่สองเป็นเหตุการณ์ที่การเข้าถึงนั้นล้มเหลวไม่ว่าจะรอนานเท่าใดก็ตาม

ในที่นี้ เราขอเสนอให้ทำการสนธิตัวแปรทรัพยากรเข้ากับเวลาสิ้นสุดในการเข้าถึง หากการเข้าถึงใดกินเวลานานกว่าเวลาสิ้นสุดของการเข้าถึง ถือว่าเกิดสิ่งผิดปกติขึ้น อันนำไปสู่ Exception (ในลักษณะคล้ายกับ Java Exception) ในที่นี้โปรแกรมเมอร์ควรจะต้องสามารถระบุหรือเขียนคำสั่งเพื่อจัดการกับปัญหานี้อย่างชัดเจนได้ โดยใช้ catch statement ดังตัวอย่างต่อไปนี้

```
Resource R = <expression1, timeout = 10>
Iterator i = R->first_element;
try {
     printf("light intensity = %f", i->light);
} catch(TimeoutException) {
printf("can’t access the light sensor");
}
```

บทที่ 3

การพัฒนาระบบ

ในที่นี้ เราจะกล่าวถึงพื้นฐานของสถาปัตยกรรม Smart Message ที่เราใช้ในการพัฒนาระบบของเรา รวมถึงกล่าวถึงข้อดีของสถาปัตยกรรมดังกล่าว ก่อนที่เราจะอธิบายถึงรายละเอียดการพัฒนาระบบของเราต่อไป

3.1 สถาปัตยกรรม Smart Message

Smart Message (SM) เป็น Mobile Agent สำหรับเครือข่ายไร้สายของระบบฝังตัว แต่ละ Mobile Agent ประกอบไปด้วยส่วนที่เป็นรหัสคำสั่งและส่วนที่เป็นข้อมูล นอกจากนี้ยังมีสถานะการทำงานขนาดย่อมอยู่ด้วย สถาปัตยกรรมนี้แตกต่างจากวิธีการติดต่อสื่อสารทั่วไป หรือ วิธีการติดต่อสื่อสารโดยการส่งข้อความถามตอบกันอยู่มาก โปรแกรมประยุกต์ SM หากต้องการข้อมูลบางอย่างจากโหนดหนึ่ง จำเป็นต้องโอนย้ายตัวเองไปยังโหนดที่มีข้อมูลนั้น แล้วประมวลผลหรือรันอยู่บนโหนดนั้นอีกที การจะทำเช่นนั้นได้ SM จะต้องรันอัลกอริธึ่มในการหาเส้นทาง ขนรหัสคำสั่งของอัลกอริธึ่มนั้นเป็นส่วนหนึ่งของ SM สำหรับใช้รันบนโหนดที่อยู่ขณะนั้น เพื่อใช้คำนวนหาโหนดถัดไปที่ต้องโอนย้ายตัวเองต่อไป เพื่อเข้าสู่โหนดเป้าหมายอีกทีหนึ่ง ส่วนรหัสคำสั่งนี้จะถูกแคชไว้ที่โหนดที่เคยผ่านไป เพื่อลด Cost ที่เกิดขึ้นจากการโอนย้ายรหัสคำสั่งที่เหมือนกันในอนาคต เมื่อเวลาผ่านไป ค่า Cost นี้จะถูกเฉลี่ยออกไปเนื่องจาก Locality ในเชิงเวลาและเชิงพื้นที่ของโปรแกรมประยุกต์ SM

สถาปัตยกรรม SM ประกอบไปด้วยเครื่องจำลอง SM (SMVM) และ Tag Space โดย SMVM มีพื้นฐานมาจาก KVM ของค่าย Sun โดยมีการพัฒนาต่อให้สนับสนุน Tag Space และสามารถโอนย้ายโปรแกรมไปรันต่อบนเครื่องอื่นได้ SMVM จึงเหมาะสมสำหรับอุปกรณ์เคลื่อนที่ที่มีทรัพยากรจำกัด และมีหน่วยความจำแค่ 160KB ก็ยังทำงานได้ ส่วน Tag Space เป็นเนื้อที่ในหน่วยความจำที่มีชื่อกำกับ สามารถทำให้การเชื่อมต่อกับ I/O และหน่วยความจำของ SMVM อยู่ในรูปแบบเดียวกันได้ การเข้าถึง I/O หรือหน่วยความจำจะต้องกระทำการผ่าน Tag Object เท่านั้น การเข้าถึง I/O หรือหน่วยความจำโดยตรงจะไม่สามารถกระทำได้ใน SMVM

ใน SM ไม่มี Java thread หรือ pre-emption ทำให้โมเดลในการทำงานของ SM นั้นค่อนข้างง่าย มีเพียง SM แค่ SM เดียวเท่านั้นที่จะ active ใน SMVM ในขณะที่ SM อื่นจะรออยู่ในคิวจนกว่า SM ที่ active จะทำหนึ่งในสิ่งต่อไปนี้คือ ทำงานเสร็จ โอนย้ายตัวเองไปทำงานต่อที่โหนดอื่น หรือ หยุดรอเหตุการณ์อะไรบางอย่าง

ข้อดีของการใช้ SM เป็น Platform ในการพัฒนาของเรามีอยู่มาก ข้อดีที่มีนัยสำคัญมากอันหนึ่งคือความสามารถในการโปรแกรมใหม่ได้หลังการติดตั้งใช้งาน ฟังก์ชันสำหรับ Aggregation หรือ Predicate สามารถติดตั้งได้ในขณะทำงานตามปกติ ข้อดีอีกข้อหนึ่งคือ SM Tag ซึ่งทำให้การเชื่อมต่อหรือเรียกใช้หน่วยความจำและ I/O มีรูปแบบเดียวกัน ความสามารถข้อนี้ทำให้การพัฒนาของเราง่ายขึ้นมาก โดยเฉพาะอย่างยิ่ง ส่วนการเข้าถึงทรัพยากรในลักษณะคล้ายการเข้าถึงตัวแปรของงานเรา

3.2 การพัฒนา DRN บน SM

โปรแกรมเชิงมหภาคถูกพัฒนาเป็น Smart Message ตัวหนึ่งที่จะถูกติดตั้งบนโหนดของผู้ใช้ โดยทั่วไปแล้ว โปรแกรม SM สามารถโอนย้ายตัวเองไปทำงานที่โหนดใดก็ได้ แต่ในการพัฒนาของเรา SM หลักของโปรแกรมเชิงมหภาคของเราจะไม่ทำการโอนย้ายตัวเองแต่อย่างใด ส่วนการดึงข้อมูลจากโหนดอื่นจะทำโดยให้ SM หลักสร้าง SM ลูกออกมา แล้วให้ SM ลูกโอนย้ายตัวเองไปยังโหนดเหล่านั้นเพื่อไปนำข้อมูลกลับมาอีกทีหนึ่ง

เมื่อตัวแปรทรัพยากรถูกประกาศด้วย Predicate และอายุขัยของการผูก ปกติ SM หลักจะไม่ทำการผูกตัวแปรทันที แต่การผูกจะทำเมื่อจำเป็นเท่านั้น เช่น เมื่อตัวแปรถูกอ้างอิงถึง การที่เซตของโหนดที่ตรงตามเงื่อนไขการผูกมักเปลี่ยนแปลงบ่อย การผูกตัวแปรเผื่อไว้ล่วงหน้าก่อนการใช้งานจริงมักไม่มีประโยชน์ เพราะเมื่อถึงเวลาต้องใช้จริง ตัวแปรก็ต้องถูกผูกใหม่อยู่ดี โอเวอร์เฮดของการผูกไว้ก่อนจึงสูญเปล่า แนวคิดนี้คล้ายกับวิธีการหาเส้นทางเมื่อต้องการใช้เท่านั้นในเครือข่ายไร้สายเฉพาะกิจ

การผูกตัวแปรทรัพยากรนั้น เราจะต้องทำการค้นหาว่าโหนดใดในระบบขณะนั้นมีคุณสมบัติตรงตามเงื่อนไขบ้าง SM หลักจะทำการสร้าง SM ค้นหา ซึ่งภายในประกอบไปด้วยเงื่อนไขสำหรับการค้นหาโหนดและค้นหาเส้นทางไปยังโหนดเหล่านั้น ถ้าไม่มีเงื่อนไขเชิงภูมิศาสตร์ปนอยู่ SM ค้นหาจะทำการ Flood

เครือข่ายโดยการทำสำเนาตัวเองแล้วโอนย้ายสำเนาเหล่านั้นไปยังโหนดเพื่อนบ้านทุกตัวจนกว่าโหนดทุกตัวในเครือข่ายจะมีสำเนาของ SM ค้นหานี้

ถ้ามีเงื่อนไขเชิงภูมิศาสตร์ปรากฏอยู่และบ่งบอกพื้นที่เป้าหมายไว้ด้วย SM ค้นหาจะโอนย้ายตัวเองไปยังโหนดเพื่อนบ้านที่ใกล้จุดเป้าหมายมากที่สุด ทำเช่นนี้ไปจนกระทั่งถึงโหนดหนึ่งที่อยู่ในพื้นที่เป้าหมาย หลังจากถึงแล้ว จึง Flood ตัวเองไปยังโหนดทุกตัวในพื้นที่เป้าหมายเพื่อตรวจสอบว่าโหนดใดตรงตามเงื่อนไขบ้างอีกทีหนึ่ง

ในทุกโหนดที่ SM ค้นหาได้เคยโอนย้ายมาถึง SM ค้นหาจะสร้าง Tag เพื่อทำเครื่องหมายไว้บนโหนดเหล่านั้น Tag นี้มีประโยชน์ในการใช้แยกแยะว่าโหนดดังกล่าว SM ค้นหาได้เคยผ่านมาแล้วหรือไม่ SM ค้นหาจะจบการทำงานของตนเองหากมันมาถึงโหนดที่ตนเองเคยทำเครื่องหมายไว้แล้ว นอกจาก Tag เครื่องหมายแล้ว SM ค้นหายังสร้าง Tag เพื่อใช้เป็นเส้นทางเดินทางกลับไปหาโหนดของผู้ใช้ตลอดเส้นทางที่มันผ่านไป โดยการจดจำว่าโหนดล่าสุดก่อนหน้าที่จะมาอยู่บนโหนดปัจจุบันนี้คือโหนดใด ดังนั้น Tag เส้นทางสู่ผู้ใช้นี้บนโหนดทั้งหลายจะประกอบกันเป็นต้นไม้รวมข้อมูลสำหรับรวบรวมข้อมูลจากโหนดทุกโหนดที่ตรงตามเงื่อนไขได้

เมื่อค้นพบโหนดที่ตรงตามเงื่อนไข SM ค้นหาจะโอนย้ายตัวเองกลับไปหา SM หลักโดยอาศัยเส้นทางจากต้นไม้รวมข้อมูล ระหว่างการเดินทางกลับ SM ค้นหาจะสร้าง route-to-id tag และ route-to-resource tag บนโหนดปัจจุบันเพื่อใช้จดจำโหนดล่าสุดก่อนหน้าโหนดปัจจุบันของการเดินทางกลับนี้ route-to-id tag ทั้งหลายเมื่อต่อรวมกันจะก่อให้เกิดเส้นทางไปสู่โหนดที่ตรงตามเงื่อนไขได้ เส้นทางนี้มีประโยชน์ต่อการเข้าถึงข้อมูลแบบอนุกรม ในขณะที่ route-to-resource tag จะก่อให้เกิดต้นไม้ multicast สำหรับการส่งคำขอการเข้าถึงไปยังโหนดที่ตรงตามเงื่อนไขแบบขนาน

ทันทีที่เดินทางกลับมาถึงโหนดผู้ใช้ SM ค้นหาจะรายงาน SM หลักเกี่ยวกับโหนดที่ตรงตามเงื่อนไข SM หลักจะเพิ่มโหนดที่อยู่ในรายงานดังกล่าวลงไปในเซตที่ผูกอยู่กับตัวแปรทรัพยากร นอกจากนี้ SM หลักจะทำการตั้งเวลาผูกใหม่ของตัวแปรทรัพยากรให้มีค่าเป็นอายุขัยการผูกต่อไป

ในการพัฒนานี้ การเข้าถึงแบบอนุกรมโดยใช้ตัววนซ้ำจะไม่ก่อให้เกิดการค้นหาทรัพยากรใหม่ SM หลักจะสร้าง SM เข้าถึง โดย SM เข้าถึงนี้จะโอนย้ายตัวเองไปยังโหนดที่ผูกไว้โดยใช้ route-to-id tag สำหรับโหนดนั้น ในทางกลับกัน การเข้าถึงตัวแปรทรัพยากรอาจนำไปสู่การค้นหาทรัพยากร ถ้าอายุขัยของ

การผูกนั้นหมดลง หรือตัวแปรไม่ได้ถูกไว้ในขณะนั้น เมื่อตัวแปรถูกผูกเสร็จ SM หลักจะสร้าง SM เข้าถึงที่มีหน้าที่โอนย้ายตัวเองไปยังโหนดที่ผูกไว้ทั้งหลายแบบขนาน โดยใช้ต้นไม้ Multicast ทันทีที่ถึงโหนดที่ผูกไว้ SM เข้าถึงจะทำงานตามที่โปรแกรมไว้คือ การเข้าถึงทรัพยากรบนโหนดเหล่านั้น อาจประมวลผลบางอย่างก่อนนำผลลัพธ์ที่ได้กลับไปแจ้งโหนดผู้ใช้ต่อไป

การเข้าทรัพยากรแบบขนานทำให้เราสามารถรวมข้อมูลระหว่างทางได้เมื่อมีโอกาส อันนำไปสู่การประหยัดพลังงานได้ ดังนั้น บนโหนดข้อต่อของต้นไม้รวมข้อมูล SM เข้าถึงจะหยุดรอ SM เข้าถึงตัวอื่นจนกว่า SM เข้าถึงจากทุกกิ่งของต้นไม้จะเดินทางมาถึงโหนดข้อต่อดังกล่าว หรือจนกว่าจะหมดเวลาของการรอคอย SM ทั้งหลายที่มาถึงที่โหนดข้อต่อจะถูกรวมเข้าเป็น SM ตัวเดียว โดย SM ที่เป็นผลลัพธ์ของการรวมจะโอนย้ายตัวเองกลับไปที่โหนดผู้ใช้ เวลาของการรอคอยในการพัฒนานี้มีค่าคงที่และค่อนข้างพื้นฐานมาก การพัฒนาที่ดีกว่านี้คือการใช้ความลึกของโหนดข้อต่อในการตั้งค่าเวลาการรอคอย โหนดยิ่งอยู่ลึกจะต้องรอคอยนาน การคำนวนความลึกของโหนดข้อต่อสามารถทำได้ในช่วงที่ SM ค้นหาเดินทางกลับจาโหนดที่ตรงตามเงื่อนไขไปยังโหนดผู้ใช้ เนื่องจากโหนดที่ตรงตามเงื่อนไขคือใบไม้ของต้นไม้ ความลึกของโหนดที่ตรงตามเงื่อนไขคือศูนย์ SM ค้นหาแต่ละตัวจะมีตัวนับความลึกนี้ ซึ่งจะเพิ่มขึ้นทีละหนึ่งสำหรับทุก Hop ระหว่างการเดินทางกลับ SM ค้นหาจะสร้าง Tag ความลึกไว้บนทุกโหนดระหว่างการเดินทางกลับหากโหนดดังกล่าวยังไม่ Tag ความลึก โดยเขียนค่าปัจจุบันของตัวนับความลึกลงไปบน Tag ความลึกดังกล่าวของโหนด แต่หากโหนดมี Tag ความลึกอยู่แล้ว SM จะเปรียบเทียบค่าตัวนับความลึกของตนกับค่า Tag ความลึก ค่าใดมากกว่า จะใช้ค่านั้นเป็นค่าใหม่ของตัวนับและ Tag ความลึก

นอกจากนี้ เรายังสามารถปรับปรุงระบบให้ดีขึ้นได้ด้วยการรวม SM ค้นหาและ SM เข้าถึงให้เป็น SM ค้นหาและเข้าถึงในตัวเดียวกันเลย SM ตัวใหม่นี้จะทำงานคล้ายกับ SM ค้นหา หากแต่ว่าเมื่อมันค้นพบโหนดที่ตรงตามเงื่อนไข มันจะเข้าถึงทรัพยากรบนโหนดทันที แล้วย้ายตัวเองกลับไปหา SM หลักเพื่อรายงานเกี่ยวกับโหนดที่ตรงตามเงื่อนไขรวมทั้งผลของการเข้าถึงด้วยในขั้นตอนเดียว การรวมกันของ SM ดังกล่าวสามารถปรับปรุงเรื่องการประหยัดพลังงานและช่วยลดเวลาในการเข้าถึงของระบบ ส่วนการรวมกันของ SM ดังกล่าวเป็นเรื่องที่สามารถนำไปพัฒนาต่อได้สำหรับผู้ที่สนใจต่อไป

บทที่ 4

การทดลองและการประเมินผล

ในการทดลองและประเมินผลนี้ เราจะประเมินการทำงานของโปรแกรมประยุกต์ที่เขียนด้วย DRN ซึ่งทำงานอยู่บนระบบที่เราพัฒนาขึ้นอีกทีหนึ่ง ในที่นี้จะขออธิบายถึงวิธีการทดลองและประเมินผลของเรา รวมทั้งพิจารณาพารามิเตอร์ในการปรับปรุงสมรรถนะของ DRN ว่ามีผลต่อสมรรถนะของโปรแกรมประยุกต์อย่างไร

4.1 วัตถุประสงค์ ตัววัด และ วิธีการ

เราได้พัฒนาโปรแกรมประยุกต์สำหรับการติดตามวัตถุโดยใช้ DRN โปรแกรมประยุกต์นี้ถูกประเมินผลบนเครือข่ายขนาด 20 โหนด แต่ละโหนดถูกจำลองโดยใช้ Smart Message Virtual Machine (SVM) ที่รันอยู่บนเครื่องคอมพิวเตอร์เดียวกัน หากแต่ว่าใช้พอร์ตต่างกัน (เนื่องจาก SMVM สามารถรันได้บนเครื่อง HP iPAQ [3] โปรแกรมของเราจึงสามารถทำงานได้บนเครื่อง iPAQ โดยไม่ต้องมีการแก้ไขใดใด)

วัตถุประสงค์ของเราในการศึกษาและประเมินผลนั้นมีอยู่ 2 ส่วนคือ ข้อแรก ทดสอบว่าโมเดลของ DRN สำหรับการโปรแกรมเชิงมหภาคเครือข่ายไร้สายของระบบฝังตัวเป็นโมเดลที่ใช้งานได้จริง ข้อสองคือ ทำความเข้าใจกับค่าอายุขัยของการผูกทรัพยากรว่ามีผลกระทบต่อโปรแกรมอย่างไร

เราเลือกตัววัดในการวิเคราะห์สมรรถนะของโปรแกรมเราดังนี้: จำนวนไบต์ที่โปรแกรมประยุกต์ส่งและค่าผิดพลาดเฉลี่ยของระยะห่าง จำนวนไบต์ที่โปรแกรมประยุกต์ส่งนั้นนับทั้งเครือข่าย ตัววัดนี้สามารถบ่งบอกถึงพลังงานที่ต้องสูญเสียไปได้อย่างหยาบๆ และยังสามารถอนุมานถึงอายุขัยโดยรวมของเครือข่ายอีกด้วย ค่าผิดพลาดเฉลี่ยของระยะห่างเป็นตัววัดระยะห่างระหว่างตำแหน่งจริงของวัตถุและตำแหน่งที่รายงานจากโปรแกรม ตัววัดนี้เป็นตัวบ่งบอกถึงความเที่ยงตรงของโปรแกรมติดตาม ตัววัดที่คล้ายกันนี้ถูกนำเสนอในงานอื่นเช่นกัน [23] เราศึกษาตัววัดดังกล่าวโดยใช้ค่าอายุขัยของการผูกต่างๆกัน เพื่อดูผลกระทบที่เกิดขึ้น

ในการทดลองของเรา เราจำลองพื้นที่ซึ่งติดตั้งเครือข่ายเซนเซอร์ไร้สายแบบหลายฮอบ อันประกอบไปด้วยโหนด 20 โหนด ที่ติดตั้งในตำแหน่งที่สุ่มมาในพื้นที่ขนาด 20m x 40m โหนดแต่ละตัวมีรัศมีการส่ง

10m ส่วนรัศมีในการรับรู้คือ 5m ระยะดังกล่าวทำให้โหนดสองโหนดที่ตรวจจับวัตถุเดียวกันได้สามารถสื่อสารตรงถึงกันได้ในลิงค์เดียว รัศมีการส่งยังเป็นตัวกำหนดเพื่อนบ้านของโหนดแต่ละตัวในการจำลองของเรา

วัตถุมีการเคลื่อนที่ที่ความเร็ว 0.25m/s โดยมีการเคลื่อนที่ตามเข็มนาฬิกาบนขอบของสี่เหลี่ยมขนาด 10m x 30m ซึ่งอยู่ตรงกลางพื้นที่ตรวจจับของเรา การเคลื่อนที่ตามเข็มนาฬิกานี้ทำให้โหนดในบริเวณต่างกันมีการตรวจพบและติดตามวัตถุโดยทั่วถึงกัน โปรแกรมจะประมาณตำแหน่งของวัตถุ 25 ครั้งตลอดการทดลอง หรือ หนึ่งครั้งทุกสี่วินาที นอกจากนี้การทดลองของเราได้กำจัดผลกระทบที่อาจเกิดจากลำดับชั้น MAC ออกไป โดยกำหนดให้ไม่มีการชนกันหรือการสูญหายของ packet ในการทดลอง

4.2 โปรแกรมติดตามวัตถุ

รูปที่ 1 แสดงรหัสเทียม DRN อย่างง่ายสำหรับโปรแกรมประยุกต์ติดตามวัตถุของเรา โดยหลักๆ แล้ว โปรแกรมจะติดตามวัตถุโดยการอ่านค่าตำแหน่งของอุปกรณ์หรือโหนดที่ตรวจพบวัตถุภายในบริเวณที่สนใจ ค่าเฉลี่ยตำแหน่งของอุปกรณ์ดังกล่าวจะถูกใช้เป็นค่าประมาณตำแหน่งของวัตถุ ในตอนแรก จะไม่มีค่าประมาณตำแหน่งของวัตถุ โปรแกรมจะต้องหาวัตถุให้ทั่วพื้นที่การทดลอง เมื่อพบวัตถุแล้ว เราสามารถใช้ตำแหน่งของวัตถุมากำหนดบริเวณที่สนใจในการค้นหาครั้งต่อไปได้โดยไม่ต้องค้นหาทั้งพื้นที่อีกต่อไป ในที่นี้กำหนดให้บริเวณที่สนใจสำหรับการค้นหาครั้งต่อไปเป็นพื้นที่ในระยะ 10 m รอบตำแหน่งที่ประมาณได้ครั้งล่าสุด แนวทางนี้มีข้อดีในแง่ของการจำกัดพื้นที่ในการค้นหา ยังผลให้มีประสิทธิภาพในแง่พลังงานดีขึ้น โดยเฉพาะอย่างยิ่งเมื่อมีการใช้การหาเส้นทางเชิงภูมิศาสตร์ในระบบ หลังจากนั้น หากเราไม่สามารถหาวัตถุในพื้นที่วงกลมพลวัตนี้ได้ บริเวณที่สนใจจะถูกเซตใหม่ให้เป็นพื้นที่ทั้งหมด

```
1: Space sp=UNIVERSE;
2: Resource R1=< (within(Sp)==TRUE) & (motion>0) >;
3: Location AverageLoc;
4:
5: for (int i=1; i<= 25; i++) {
6: AverageLoc = average(R1->Location);
```

```
7: if (AverageLoc != NULL) {
8: System.out.println("Average("+i+")="+AverageLoc);
9: sp.updateRegion(AverageLoc, 10);
10: } else {
11: System.out.println("Average("+i+")=NOT FOUND");
12: sp = UNIVERSE;
13: }
14: sleep(4000);
15: }
```

รูปที่ 1. รหัสเทียมสำหรับโปรแกรมประยุกต์ติดตามวัตถุ

ส่วนโปรแกรมภาษาจาวาสำหรับติดตามวัตถุจะแสดงไว้ในรูปที่ 2 ซึ่งมีอัลกอริธึ่มการทำงานเหมือนรหัสเทียมในรูปที่ 1 จะเห็นได้ว่ามีความเป็นไปได้ที่เราจะแปลงรหัสเทียมของ DRN เป็นโปรแกรมในภาษาจาวาได้โดยไม่ยากนัก สำหรับโปรแกรมในภาษาจาวานี้ คลาส TrackingApp เป็นเพียงคลาสที่เพิ่มขยายจาก SmWrapper ซึ่งเป็นคลาสที่ซ่อนรายละเอียดเกี่ยวกับ SM จากโปรแกรมเมอร์ เพื่อที่จะทำให้เข้ากันได้กับ Syntax ของภาษาจาวา เราได้สร้างส่วนการประกาศเงื่อนไขของทรัพยากรเป็นคลาสชื่อว่า TrackingExpression ดังแสดงไว้ในรูปที่ 3 ส่วนการสร้างคลาสเงื่อนไขทรัพยากรแบบอัตโนมัติจากรหัสเทียมอาจทำได้ แต่ไม่อยู่ในขอบเขตของโครงงานนี้ คลาส Expression แต่ละคลาสจะมี evaluate () method ที่จำเป็นต้องรันบนอุปกรณ์เพื่อดูว่าอุปกรณ์ตรงตามเงื่อนไขของ Expression หรือไม่

```
1: public class TrackingApp extends SmWrapper{
2:
3: private final static int timeout = 24000; // Binding lifetime
4: private Space sp;
5: private TrackingExpression tExp;
6: private Resource resource;
7: private LocationAverage agg;
```

```
8:
9: public TrackingApp(){
10:   super("TrackingApp");
11: }
12:
13: public void run() {
14:   try {
15:         sp = new Space(null,-1); // sp = UNIVERSE
16:         tExp = new TrackingExpression(sp, "motion");
17:         resource = new Resource(tExp, timeout);
18:         agg = new LocationAverage();
19:         for (int i=1; i<= 25; i++) {
20:               agg = (LocationAverage)resource.access(agg, 4000);
21:               System.out.println("agg = "+agg);
22:               Location average = (Location)agg.evaluator();
23:               if (average != null) {
24:                     System.out.println("Average("+i+") = "+average);
25:                     sp.updateRegion(average, 10);
26:               } else {
27:                     System.out.println("Average("+i+") = NOT FOUND");
28:                     sp.updateRegion(null, -1); // sp = UNIVERSE
29:               }
30:               sleep(4000);
31:         }
32: } catch(Exception e) {}
33: }
```

```
34:
35: public static void main(String []args) {
36:   TrackingApp trackingApp = new TrackingApp();
37:   String []types;
38:   types = new String[3];
39:   types[0] = "TrackingApp";
40:   types[1] = "TrackingExpression";
41:   types[2] = "LocationAverage";
42:   trackingApp.initSM(types, trackingApp);
43:   trackingApp.run();
44: }
45: }
```

รูปที่ 2 โปรแกรมภาษาจาวาแบบ DRN สำหรับติดตามวัตถุ

ในส่วนโปรแกรมประยุกต์ติดตามวัตถุนี้ ทรัพยากรจะถูกเข้าถึงแบบขนาน ซึ่งเปิดโอกาสให้มีการประมวลผลในเครือข่ายได้ (อาทิเช่น การรวมข้อมูล) อันจะนำไปสู่การลดการใช้พลังงานโดยรวมของระบบ [10, 11, 13, 14, 17, 19] ตามปกติแล้ว ในระบบอื่นๆ โปรแกรมสำหรับการรวมข้อมูลในเครือข่ายไม่สามารถปรับแต่งหรือติดตั้งแบบพลวัตในขณะทำงาน หลังจากเริ่มติดตั้งใช้งานเครือข่ายไปแล้วได้ [19] ในบางระบบ API อาจไม่สนับสนุนให้เขียนหรือแก้โค้ดการรวมข้อมูลในเครือข่ายขึ้นมาใหม่ ซึ่งต่างจากจากงานของเรา DRN ได้สร้าง Class Aggregation ไว้ให้ ที่สามารถเพิ่มขยายให้เป็นการรวมข้อมูลแบบใหม่ได้ตามที่ต้องการ อีกทั้งยังสามารถติดตั้งได้แบบพลวัตในขณะทำงาน

```
1: public class TrackingExpression extends Expression{
2:
3: private Space sp_;
4: private String moTag_;
5:
```

```
6: public TrackingExpression(Space sp, String moTag) throws
7:                          BadSMApiUsageException {
8:    sp_=sp;
9:    moTag_=moTag;
10: }
11:
12: public boolean evaluate() {
13:   try{
14:         Integer moInt = (Integer)TagSpace.readTag(moTag_);
15:         GPSData gps = (GPSData)TagSpace.readTag("gps");
16:         if (!sp_.outside(new Location(gps.latitude, gps.longitude))
17:               && (moInt.intValue()>0) ) {
18:               return true;
19:         }
20:   }catch(Exception e) {}
21:   return false;
22: }
23: }
```

รูปที่ 3 Class TrackingExpression สำหรับโหนดที่ตรงตามเงื่อนไข

โดยทั่วไปแล้ว เทคนิคการรวมข้อมูลจะถูกสร้างโดยใช้ฟังก์ชัน 3 ฟังก์ชันได้แก่ initializer i(), merger m(), และ evaluator e() โดยที่ initialize i() จะระบุว่าจะทำการสร้างบันทึกสถานะข้อมูลสำหรับค่าของตัวรับรู้ค่าหนึ่งได้อย่างไร DRN จะเรียกฟังก์ชันนี้บนอุปกรณ์ที่มีคุณสมบัติตรงตามเงื่อนไขที่ประกาศไว้ บันทึกสถานะข้อมูลนี้จะถูกส่งกลับไปยังโหนดของผู้ใช้ ระหว่างการเดินทาง บันทึกดังกล่าวอาจไปพบเจอกับบันทึกอื่นจากเซตของโหนดที่ตรงตามเงื่อนไขเดียวกัน ในกรณีเช่นนี้ DRN จะเรียกฟังก์ชัน merger m() เพื่อรวมข้อมูลเหล่านี้ให้เป็นบันทึกเดียว เมื่อบันทึกได้มาถึงโหนดของผู้ใช้ DRN จะทำการเรียกฟังก์ชัน evaluator e() เพื่อคำนวณค่าที่แท้จริงของข้อมูลที่รวมได้

ในโปรแกรมประยุกต์ของเรา เราได้แสดงการสร้างเทคนิครวมข้อมูลใหม่ชื่อว่า LocationAverage (ในรูปที่ 4) โดยสามารถทำได้อย่างง่ายดายเพียงแค่การเพิ่มขยาย Class Aggregation และ Overload ฟังก์ชันทั้งสามที่กล่าวไปแล้วข้างต้น เราใช้ <sum_x, count_x, sum_y, count_y> แทนสถานะของข้อมูล สมมุติว่าอุปกรณ์ที่ตรงตามเงื่อนไขอยู่ที่ตำแหน่ง (x1, y1) ตัว initializer จะทำการสร้างบันทึกสถานะข้อมูลเป็น <x1, 1, y1, 1> ส่วน merger จะรวมสถานะ <x1, cx1, y1, cy1> และ <x2, cx2, y2, cy2> เป็นสถานะเดียวคือ <x1+x2, cx1+cx2, y1+y2, cy1+cy2> สำหรับ evaluator จะประเมินผลลัพธ์ที่ให้ค่าเป็น <sum_x/count_x, sum_y/count_y> ซึ่งเป็นค่าตำแหน่งเฉลี่ย

```
1: public class LocationAverage extends Aggregate{
2:
3: private GPSData gps;
4: double sum_x, sum_y;
5: int count_x, count_y;
6:
7: public void initializer() {
8:    try {
9:          gps = (GPSData)TagSpace.readTag("gps");
10:         sum_x = gps.latitude;
11:         sum_y = gps.longitude;
12:         count_x = 1;
13:         count_y = 1;
14:   } catch (Exception e) {}
15: }
16:
17: public void merger(Aggregate agg) {
18:   sum_x = sum_x+agg.sum_x;
```

```
19:   sum_y = sum_y+agg.sum_y;
20:   count_x = count_x+agg.count_x;
21:   count_y = count_y+agg.count_y;
22: }
23:
24: public Object evaluator() {
25:   try {
26:         return new Location(sum_x/count_x, sum_y/count_y);
27:   } catch (Exception e) {
28:         return null;
29:   }
30: }
31: }
```

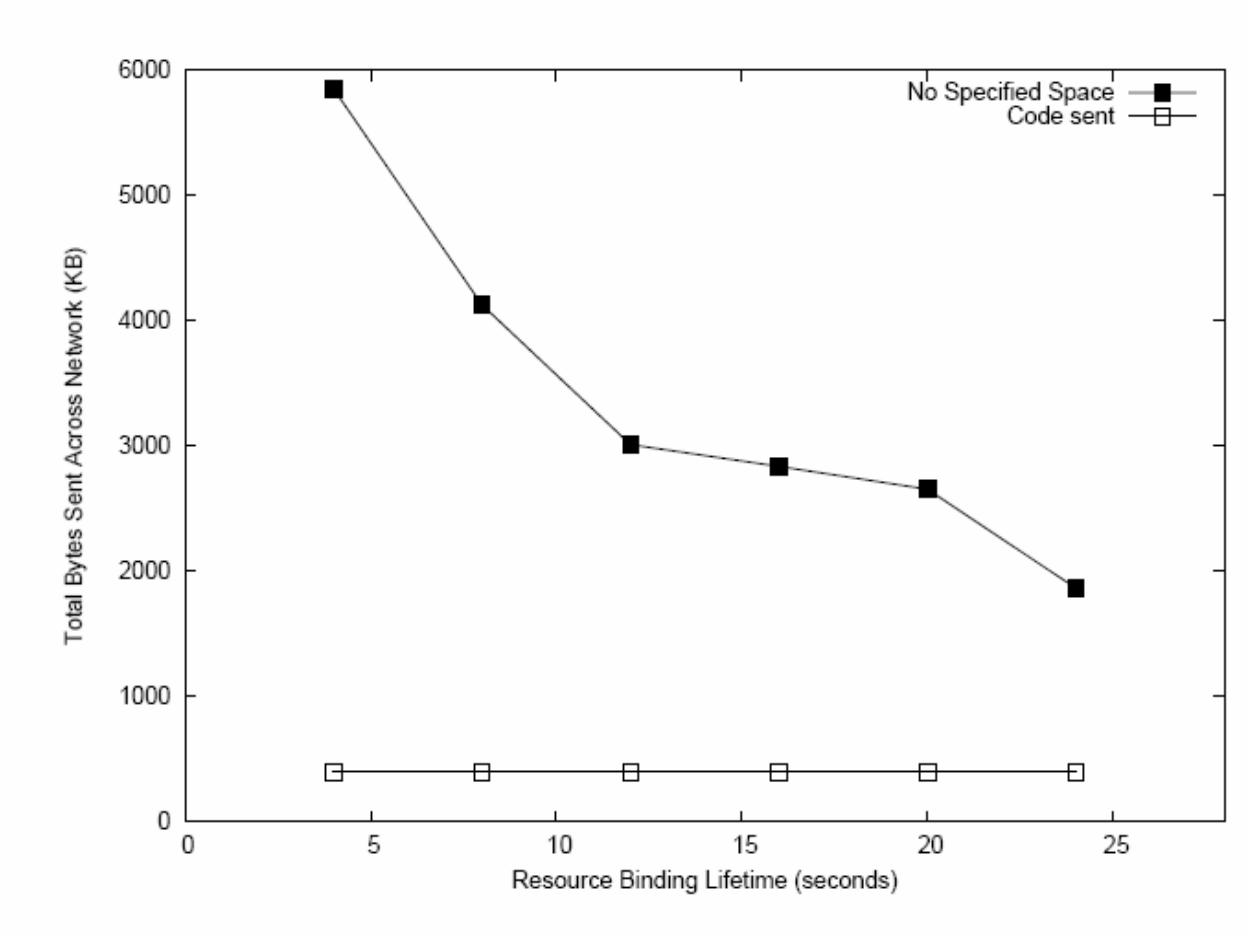
รูปที่ 4 แสดง Class LocationAverage สำหรับการประมวลผลในเครือข่าย

4.3 การปรับแต่ง

ในโมเดลของเรา หากใช้ความหมายอย่างเข้มงวด การเข้าถึงทรัพยากรจะต้องกระทำการบนโหนดที่ตรงตามเงื่อนไขเชิงประกาศในขณะที่เข้าถึงเท่านั้น การเปลี่ยนแปลงใดใดในเซตของโหนดที่ตรงตามเงื่อนไขไม่จำเป็นต้องให้โปรแกรมเมอร์มาเสียเวลาเขียนโปรแกรมรองรับไว้เอง ดังนั้น DRN จะต้องทำการผูกทรัพยากรได้ใหม่อย่างพลวัตและอัตโนมัติ ความหมายที่เข้มงวดนี้อาจนำไปสู่โอเวอร์เฮดและการใช้พลังงานอย่างมากเพื่อให้มั่นใจได้ว่าการผูกทรัพยากรนั้นถูกต้องอยู่เสมอ จึงไม่น่าประหลาดใจนักที่เรานำเสนอวิธีที่สามารถใช้ในการปรับแต่งระดับความเข้มงวดดังกล่าวเพื่อแลกกับการประหยัดพลังงานที่เพิ่มขึ้น หนึ่งในตัวปรับแต่งของเราคือ อายุขัยของการผูกทรัพยากร ยกตัวอย่างเช่น การใช้อายุขัยของการผูกทรัพยากรค่า t ถึงแม้ว่าจะลดความเข้มงวดของโมเดลลงแต่จะอนุญาตให้เข้าถึงทรัพยากรที่ถูกผูกไว้ภายใน t วินาทีที่ผ่านมา

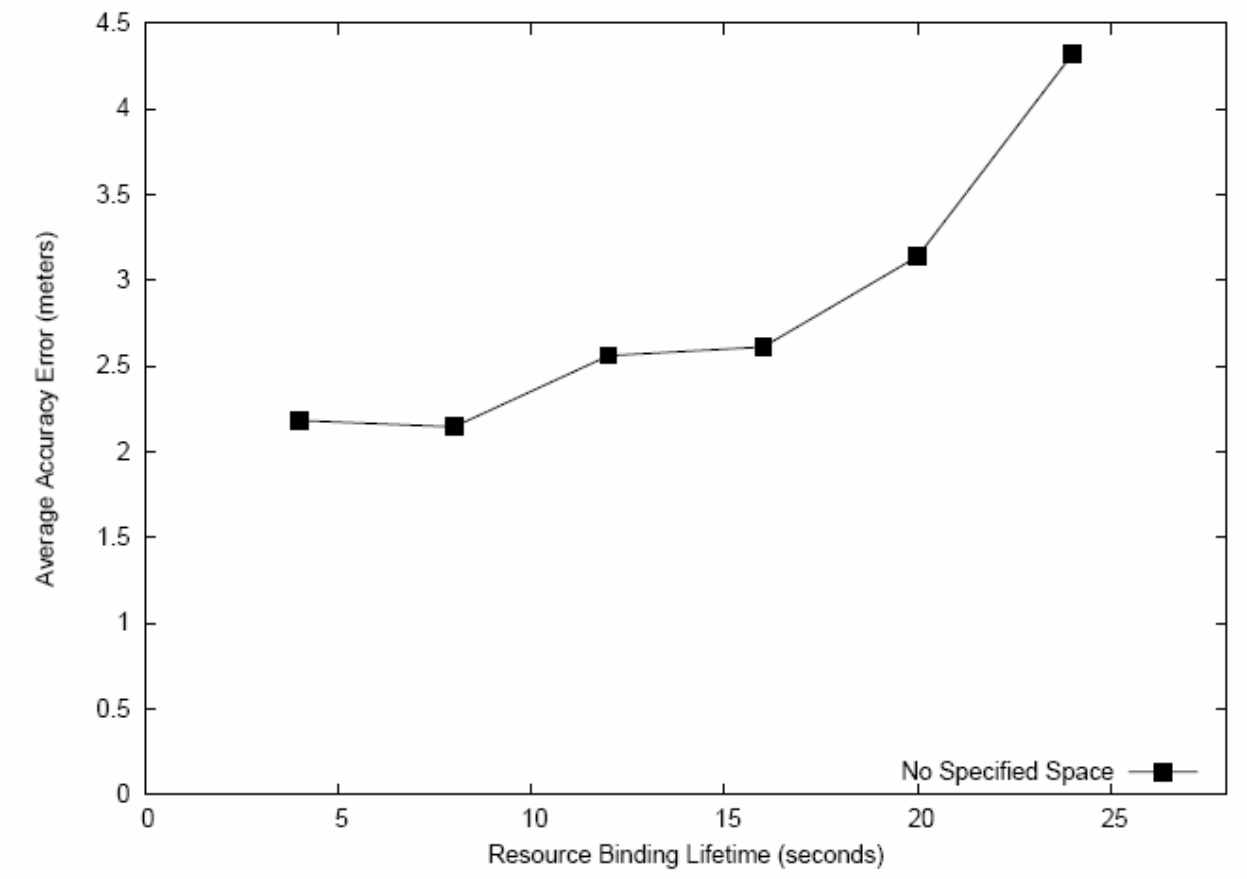
ในการทดลองนี้ เราศึกษาผลกระทบของอายุขัยการผูก ต่อการใช้พลังงานและความแม่นยำของการติดตามวัตถุของโปรแกรมประยุกต์เวอร์ชันที่ยังไม่ได้ปรับปรุงประสิทธิภาพ กล่าวคือ บรรทัดที่ 25 ในรูปที่ 2 นั้นถูกลบออกไป ดังนั้นการค้นหาวัตถุจะทำโดยทั่วเครือข่าย นอกจากนี้ เรายังมุ่งหมายที่จะแสดงให้เห็นว่าถึงแม้ว่า เงื่อนไขเชิงประกาศและตัวแปรที่เกี่ยวข้องไม่ได้ถูกแก้ไขเลย ทรัพยากรยังคงสามารถถูกผูกใหม่ได้อย่างเหมาะสมและเป็นพลวัต

รูปที่ 5 แสดงจำนวนไบต์ที่ส่งเมื่อเทียบว่าอายุขัยการผูกทรัพยากร ผลลัพธ์เป็นไปดังคาดคือ จำนวนไบต์ที่ใช้ลดลง เมื่อเราเพิ่มอายุขัยการผูกทรัพยากรขึ้น เพราะเป็นการลดจำนวนการค้นหาทรัพยากรที่ตรงตามเงื่อนไข ผลลัพธ์ชี้ให้เห็นว่ามันเป็นไปได้ที่จะประหยัดพลังงานได้อย่างมากโดยที่ไม่ทำให้ความแม่นยำลดลงอย่างมีนัยสำคัญ กล่าวคือ เราสามารถลดจำนวนไบต์ที่ต้องส่งลงได้ถึง 51.5% โดยที่ความแม่นยำลดลงเพียงเล็กน้อยหากเราเพิ่มอายุขัยการผูกทรัพยากรจาก 4 เป็น 16 วินาที การประหยัดนี้จะมีค่ามากขึ้นอีกเมื่อ โปรแกรมได้ถูกแคช หรือถูกติดตั้งในเครือข่ายเรียบร้อยแล้ว หากเราแยกจำนวนไบต์ที่ต้องใช้ในส่งโปรแกรมไปติดตั้งในเครือข่ายออกไป การประหยัดของเราจะเพิ่มขึ้นเป็น 55.2%



รูปที่ 5 แสดงจำนวนไบต์ที่โปรแกรมประยุกต์ต้องส่ง

ความผิดพลาดในการติดตามเฉลี่ยไม่ได้เพิ่มขึ้นอย่างมีนัยสำคัญ จนกว่าอายุขัยการผูกจะมีค่ามากกว่า 16 วินาที (รูป 6) ผลลัพธ์นี้ไม่น่าแปลกใจ ถ้าวัตถุเคลื่อนที่ด้วยความเร็ว 0.25 m/s ออกจากตัวรับรู้ที่ผูกไว้ มันจะใช้เวลาอย่างมากที่สุด 20 วินาที ที่จะออกจากรัศมีการรับรู้ของตัวรับรู้ ในทางกลับกัน ถ้าวัตถุเคลื่อนที่เข้าหาตัวรับรู้โดยไม่เปลี่ยนทิศทาง มันจะใช้เวลาอย่างมากที่สุด 40 วินาทีในการออกจากรัศมีการรับรู้ สำหรับการเคลื่อนที่ในการทดลองนี้ หากต้องการความแม่นยำที่พอใช้ได้ เราไม่จำเป็นต้องค้นหาทรัพยากรใหม่ภายใน 20 วินาที อย่างไรก็ตาม หลังจาก 20 วินาที ความแม่นยำจะเริ่มแย่ลงอย่างมีนัยสำคัญ ถ้าเราไม่ค้นหาทรัพยากรใหม่หลังจาก 40 วินาที เราจะไม่สามารถติดตามวัตถุได้



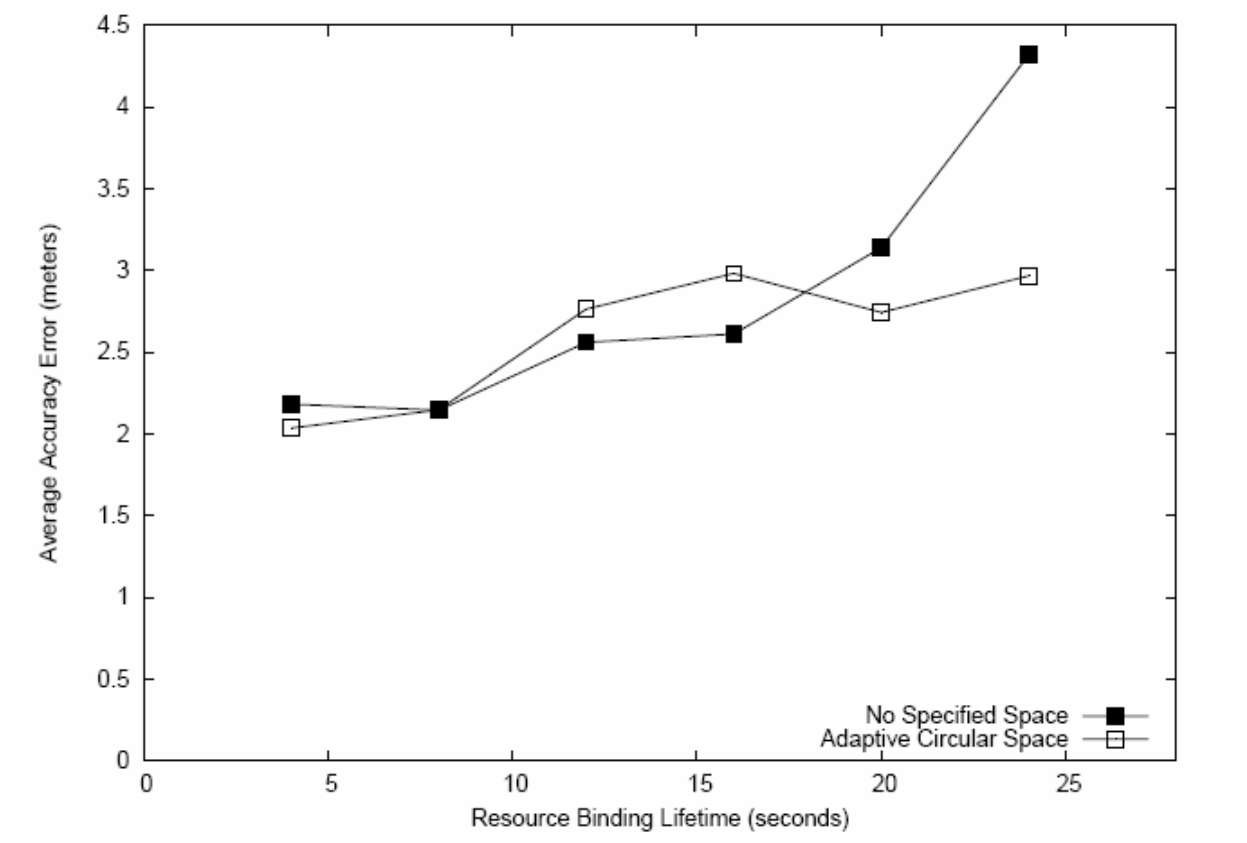
รูปที่ 6 แสดงระยะผิดพลาดเฉลี่ย

ความแม่นยำในการติดตามขึ้นอยู่กับหลายปัจจัย เช่น เทคนิคในการประมาณตำแหน่ง ความหนาแน่นของเครือข่าย และ รัศมีการรับรู้ ความผิดพลาดในการประมาณระดับ 2-3 m ในการทดลองนี้นับว่าใช้ได้ สำหรับวิธีการประมาณตำแหน่งที่เรียบง่ายเช่นนี้ ในเครือข่ายที่มีความหนาแน่นต่ำ และ มีรัศมีการรับรู้เพียง 5 m

4.4 การเพิ่มประสิทธิภาพโดยการระบุบริเวณ

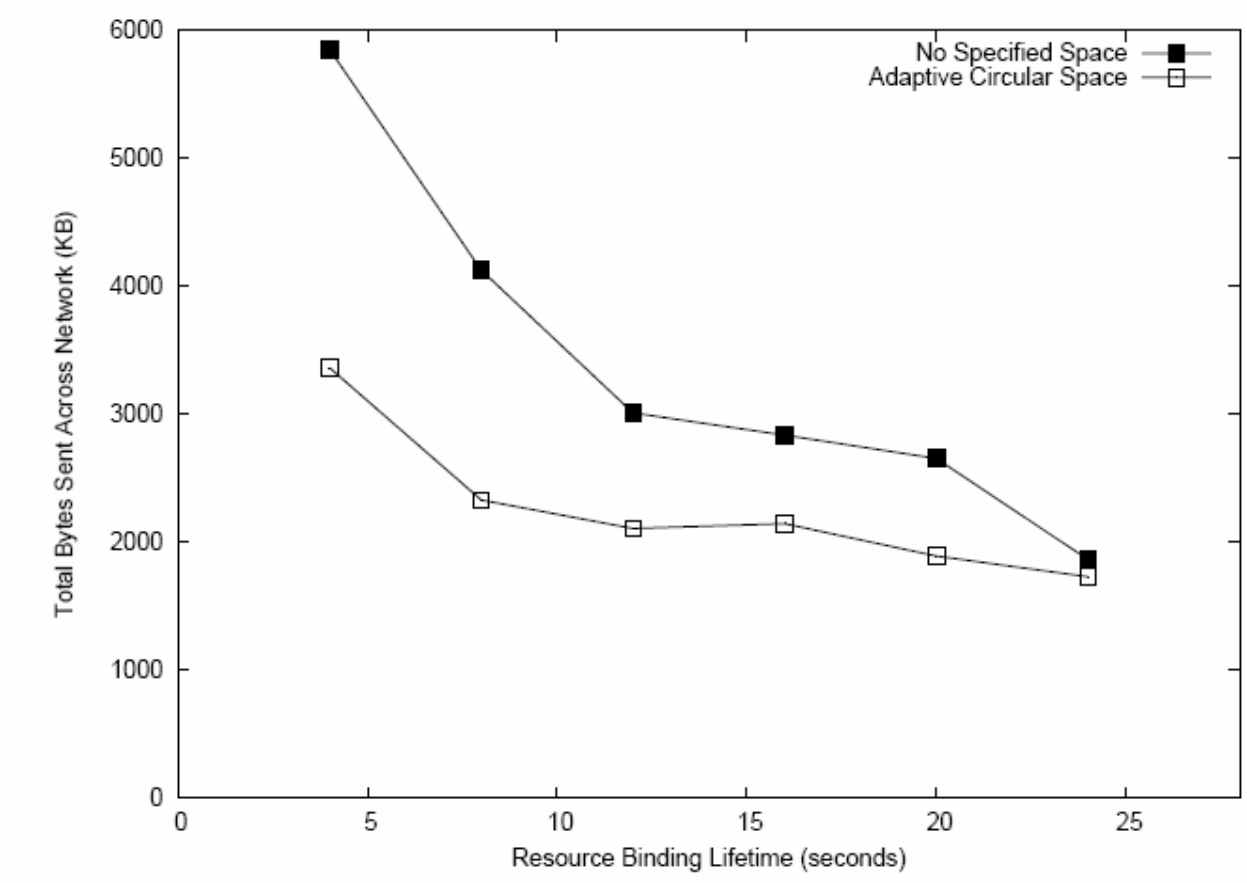
เหมือนดังเช่นวิธีการเขียนโปรแกรมอื่นๆ การเขียนโปรแกรมให้มีประสิทธิภาพจำเป็นที่จะต้องเข้าใจระบบที่ทำงานอยู่ด้วย ยกตัวอย่างเช่น ในระบบหน่วยความจำเสมือน เราควรเขียนโปรแกรมที่ทำให้เกิดจำนวน Page Faults น้อยที่สุด หากจะเขียนโปรแกรมให้กระทำการบนข้อมูลทุกตัวในอาร์เรย์สองมิติ ซึ่งรัน

บนระบบที่ใช้หน่วยความจำเสมือน เราควรเข้าถึงข้อมูลของอาร์เรย์ทีละแถว แทนที่จะเข้าถึงทีละคอลัมน์ในทางที่คล้ายกัน โปรแกรมประยุกต์ติดตามวัตถุของเราจะมีประสิทธิภาพที่มากขึ้นเมื่อมีการระบุพื้นที่ที่ต้องการค้นหา เพราะว่าไลบรารี่ของเราสนับสนุนการทำ Geographic Routing หากมีการระบุพื้นที่ คำร้องขอในการค้นหาทรัพยากรจะถูกส่งต่อตามตำแหน่งทางภูมิศาสตร์ไปยังพื้นที่เป้าหมายได้ แทนที่จะต้อง Flood คำร้องขอนี้ไปทั่วเครือข่าย



รูปที่ 7 ผลกระทบของการระบุพื้นที่ต่อระยะทางความผิดพลาดเฉลี่ย

เพื่อที่จะศึกษา ผลกระทบของการระบุตำแหน่งต่อโปรแกรมประยุกต์ติดตามวัตถุของเรา เราจึงได้ทำการทดลองคล้ายกับการทดลองที่ผ่านมา หากแต่ว่าครั้งนี้ โปรแกรมของเรามีบรรทัดที่ 25 ของรูปที่ 2 อยู่ด้วย เมื่ออายุขัยการผูกเพิ่มขึ้น เราจะประหยัดมากขึ้นเนื่องจาก เป็นการลดจำนวนครั้งในการค้นหาทรัพยากรลง นอกจากนี้ ความแม่นยำในการติดตามไม่ได้แย่ลงเมื่อมีการระบุพื้นที่ (รูปที่ 7)



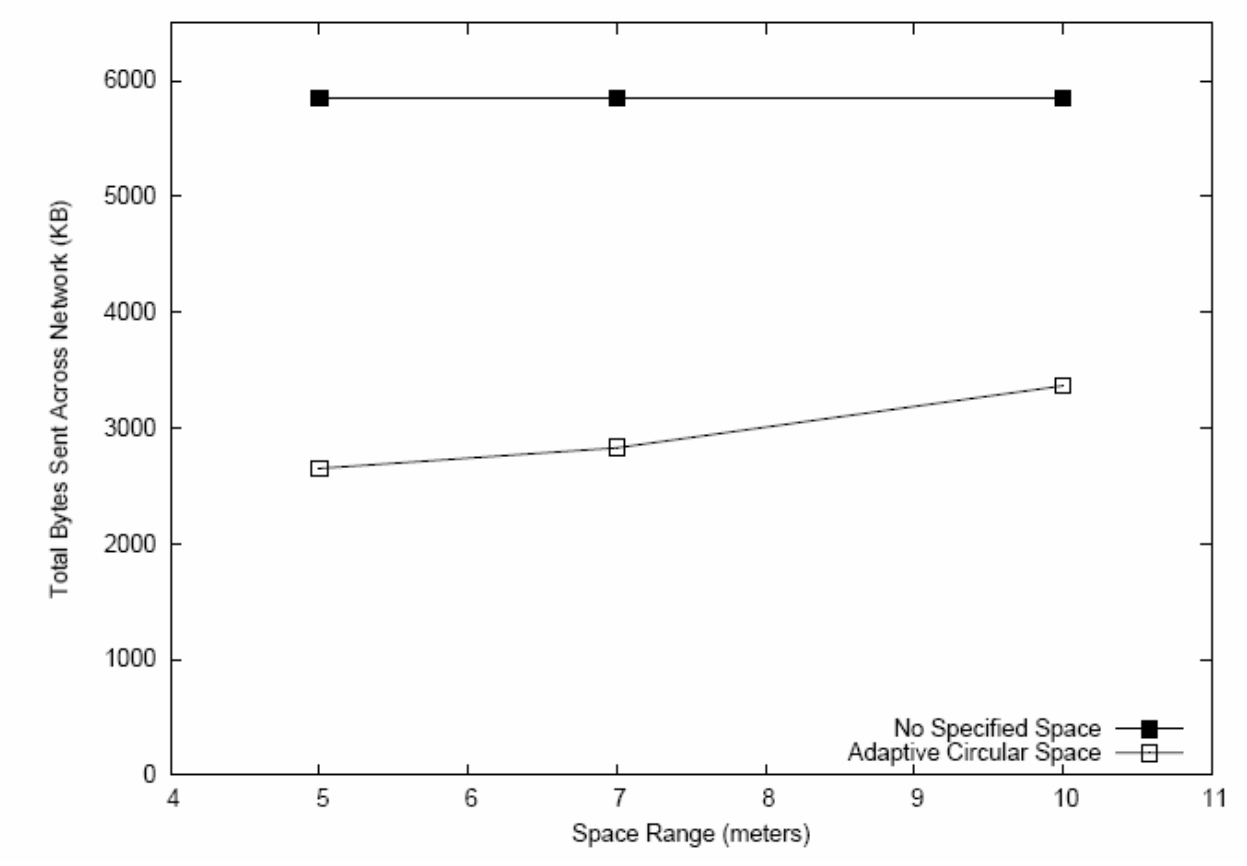
รูปที่ 8 แสดงผลกระทบของการระบุพื้นที่ต่อจำนวนไบต์ที่ต้องส่ง

ผลลัพธ์ของเราบ่งชี้ว่า การระบุพื้นที่เป้าหมายแบบพลวัตสามารถประหยัดจำนวนไบต์ในการส่งได้ถึง 42.5% เมื่อเทียบกับการไม่ระบุพื้นที่ (รูปที่ 8) ถึงแม้ว่าการประหยัดนี้จะมีนัยสำคัญ อาจมีผู้ที่คาดหวังว่าจะประหยัดได้มากกว่านี้ เพราะว่า Geographic Routing มีประสิทธิภาพมากกว่าการ Flooding มาก อย่างไรก็ตาม เมื่อคำขอค้นหาทรัพยากรได้ถูกส่งต่อเชิงภูมิศาสตร์ไปยังพื้นที่เป้าหมายแล้ว คำร้องขอนั้นจะถูก Flood ในพื้นที่จำกัดดังกล่าวอยู่ดี เพื่อค้นหาทรัพยากรทั้งหมดที่ตรงตามเงื่อนไข การ Flood แบบจำกัดพื้นที่นี้ มีโอเวอร์เฮด จึงยังผลให้ประหยัดน้อยกว่าที่คาดไว้บ้าง

4.5 ผลกระทบจากรัศมีของพื้นที่วงกลมที่ระบุ

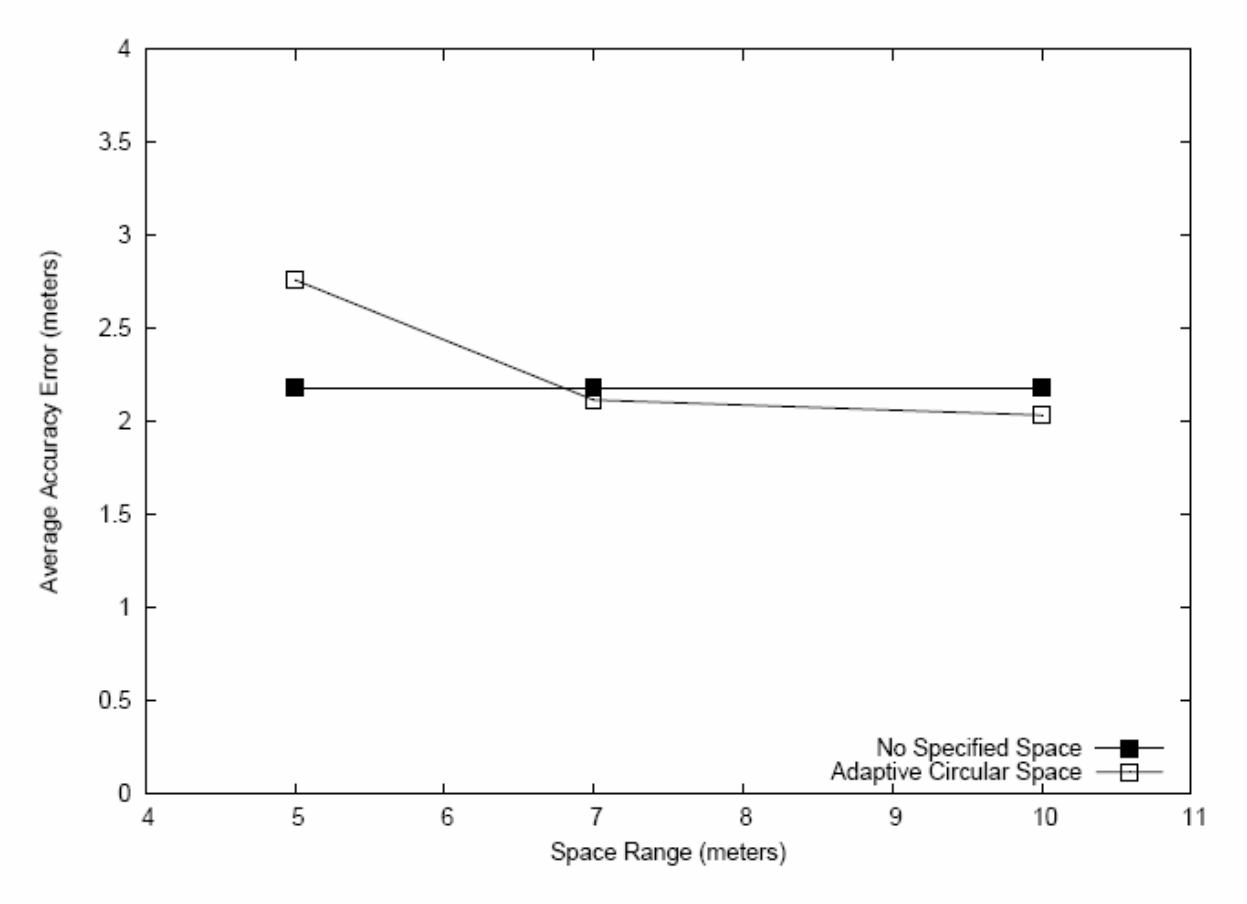
ตามธรรมดาแล้ว หากต้องการจะลดโอเวอร์เฮดให้น้อยที่สุด พื้นที่วงกลมที่ระบุควรมีขนาดเล็กที่สุดเท่าที่จะทำได้ ในการทดลองที่แล้ว เราใช้รัศมี 10 m พื้นที่ดังกล่าวดูเหมือนว่าจะมีขนาดใหญ่เกินความจำเป็น เมื่อเทียบกับรัศมีการรับรู้ 5 m ดังนั้น อาจมีผู้คาดหวังว่ารัศมีของพื้นที่เป้าหมายแค่ 5 m น่าจะเพียงพอแล้วสำหรับตัวรับรู้ทุกตัวที่ตรวจพบวัตถุในขณะหนึ่งๆ อย่างไรก็ตาม ความคาดหวังดังกล่าวค่อนข้างมองโลกในแง่ดีเกินไป เนื่องจากวัตถุมีการเคลื่อนที่อยู่ตลอดเวลา และการประมาณตำแหน่งย่อมมีความผิดพลาดอยู่ (ตำแหน่งที่ประมาณถูกใช้ในการกำหนดพื้นที่เป้าหมายในการค้นหาครั้งถัดไป)

เพื่อที่จะศึกษาผลกระทบของรัศมีพื้นที่เป้าหมายสำหรับโปรแกรมประยุกต์ติดตามวัตถุของเรา เราได้ทำการทดลองคล้ายกับการทดลองที่แล้ว หากแต่แตกต่างจากเดิมตรงที่เราใช้อายุขัยการผูก 4 วินาที และศึกษาสมรรถนะของระบบเมื่อรัศมีเปลี่ยนไป



รูปที่ 9 แสดงผลกระทบของรัศมีพื้นที่เป้าหมายที่ระบุต่อจำนวนไบต์ที่ส่ง

ผลการทดลองเป็นไปตามคาด กล่าวคือ เมื่อรัศมีเพิ่มขึ้น จำนวนไบต์ในการส่งเพิ่มขึ้นตาม (รูปที่ 9) ในขณะที่ความแม่นยำมากขึ้น (รูปที่ 10) ดังนั้น ผลการทดลองนี้บ่งชี้ว่า การใช้รัศมีพื้นที่เป้าหมาย 7 m เราสามารถรักษาความแม่นยำในระดับเดียวกับการค้นหาทั้งเครือข่ายได้ แต่จะประหยัดจำนวนไบต์ในการส่งได้ถึง 51.6% การใช้รัศมีพื้นที่ 7 m (หรือ 2m มากกว่ารัศมีการรับรู้) นั้นไม่น่าแปลกใจเช่นกัน เนื่องจากความผิดพลาดของเราอยู่ที่ประมาณ 2-3 m



รูปที่ 10 แสดงผลกระทบของรัศมีพื้นที่เป้าหมายที่ระบุต่อระยะทางผิดพลาดเฉลี่ย

บทที่ 5

งานวิจัยที่เกี่ยวข้อง

การเขียนโปรแกรมสำหรับเครือข่ายระบบฝังตัวไร้สาย ได้รับความสนใจในวงการวิจัยอย่างมากในช่วงเวลา 4-5 ปีนี้ งานวิจัยบางชิ้นจึงมีความเกี่ยวข้องและมีอิทธิพลต่อแนวทางวิจัยของเรา ซึ่งจะได้กล่าวถึงในที่นี้

งานของเราได้รับอิทธิพลอย่างมากจากงานวิจัยชื่อ Spatial Programming (SP) [2, 12] และ Spatial View [16] โดยงานของเราและงานดังกล่าวมีวิสัยทัศน์ทางด้านการโปรแกรมเครือข่ายระบบฝังตัวไร้สายที่ตรงกันว่าควรจะมีการโปรแกรมเครือข่ายเป็นกลุ่มก้อน พยายามทำให้การเข้าถึงทรัพยากรทำได้ง่ายเหมือนการเข้าถึงตัวแปรตัวหนึ่ง เปิดเผยคุณสมบัติในแง่พื้นที่ให้โปรแกรมเมอร์สามารถอ้างอิงถึงได้ในโปรแกรม ซ่อนรายละเอียดของเครือข่าย สนับสนุน Imperative Programming อย่างไรก็ตาม SP สนับสนุนแค่การเข้าถึงทรัพยากรแบบลำดับ ขณะที่ DRN สนับสนุนทั้งการเข้าถึงทรัพยากรแบบลำดับและแบบขนาน การเข้าถึงทรัพยากรแบบขนานทำให้ช่วยลดเวลาในการเข้าถึงรวมและลดการใช้พลังงานรวมลงได้ (โดยการใช้การประมวลผลในเครือข่าย) นอกจากนี้ SP เป็นการโปรแกรมแบบ Imperative Programming แต่ DRN เป็นการโปรแกรมแบบผสมระหว่าง Declarative และ Imperative Programming ยิ่งกว่านั้นคือ การผูกทรัพยากรของ SP เป็นแบบสถิตย์โดยปริยาย แต่การผูกทรัพยากรของ DRN เป็นแบบพลวัตโดยปริยาย ถึงแม้ว่าการผูกทรัพยากรแบบพลวัตอาจทำได้ใน SP แต่โปรแกรมเมอร์จะต้องระบุการทำการผูกทรัพยากรใหม่อย่างชัดเจนเองในโปรแกรม การมุ่งเน้นเรื่องการผูกทรัพยากรแบบพลวัตของ DRN นี้คล้ายกับของ Spatial View หากแต่ว่า Spatial View ไม่สนับสนุนการเข้าถึงทรัพยากรแบบขนานและไม่มี Declarative Abstraction

ในเมื่อตัวแปรคือพื้นที่ในหน่วยความจำ การ Map ทรัพยากรเข้ากับตัวแปรของ DRN จึงคล้ายกับ Memory-Mapped File อย่างไรก็ตาม DRN จำเป็นต้องจัดการเรื่องการ Map แบบพลวัตและการล้มเหลวของการติดต่อสื่อสารบ่อยครั้ง ในขณะที่ Memory-Mapped File จะไม่ต้องจัดการกับเรื่องดังกล่าวเลย

Abstract Region [22, 23] มุ่งเน้นในนิยามที่กว้างขึ้นของพื้นที่ กล่าวคือ พื้นที่ใน Abstract Region สามารถเป็นเชิงกายภาพหรือเชิงตรรกก็ได้ ยกตัวอย่างเช่น พื้นที่เชิงตรรกสามารถถูกกำหนดโดยจำนวน Hop ของการติดต่อสื่อสาร ตัวอย่างนี้ชี้ให้เห็นว่า Abstract Region นั้นไม่ได้ตั้งใจที่จะซ่อนรายละเอียดเชิงเครือข่ายจากโปรแกรมเมอร์แต่อย่างใด ยิ่งกว่านั้น พื้นที่นั้นเป็นเพียงแค่หนึ่งในเงื่อนไขเชิงประกาศในงานเรา (ถึงแม้ว่าจะเป็นส่วนที่สำคัญมากก็ตาม) ดังนั้น พื้นที่จึงไม่ใช่ประเด็นที่เราพยายามมุ่งเน้นในงานนี้

งานของเราได้รับอิทธิพลอย่างมากจากงาน Directed Diffusion [10, 14, 15] และ LEACH [11] ซึ่งจุดประเด็นเรื่องการประหยัดพลังงานโดยการประมวลผลข้อมูลในเครือข่าย นอกเหนือจากอิทธิพลในแง่ดังกล่าวแล้ว DRN ยังคล้ายกับ Diffusion ในหลายแง่มุม ยกตัวอย่างเช่น ใน API ของ Diffusion [6] เราจำเป็นที่จะต้องบรรยายข้อมูลเชิงประกาศในการทำ Publication และ Subscription ดังนั้น ทั้ง Diffusion และ DRN จึงมีลักษณะของการโปรแกรมแบบผสม นอกจากนี้ แนวทางการเน้นข้อมูลเป็นหลักของ Diffusion ช่วยให้การซ่อนรายละเอียดทางเครือข่ายทำได้อย่างมีประสิทธิภาพมาก ซึ่งเป็นหนึ่งในคุณสมบัติที่ DRN พยายามเน้นเช่นกัน อย่างไรก็ตาม DRN แตกต่างจาก Diffusion ตรงที่ DRN เน้นเรื่องตั้งชื่อทรัพยากร แต่ Diffusion เน้นเรื่องการตั้งชื่อข้อมูลเพื่ออ้างอิง

การโปรแกรมเครือข่ายไร้สายของระบบฝังตัวเป็นหน่วยเดียวกัน ได้มีงานวิจัยอยู่บ้าง ได้แก่ TAG [19] และ COUGAR [1] งานดังกล่าวเสนอให้โปรแกรมเครือข่ายไร้สายของระบบฝังตัวเหมือนฐานข้อมูลตัวหนึ่ง แต่เราเสนอให้โปรแกรมเครือข่ายเสมือนเป็นเครื่องๆเดียว

งานวิจัยทางด้านการโปรแกรมแบบมหภาคอื่นๆ ได้แก่ งานที่ชื่อว่า Kairos [9] งานนี้คล้ายกับ Split-C [21] ซึ่งเป็นงานทางด้านการโปรแกรมแบบขนาน Kairos สนับสนุนให้มีการเข้าถึงตัวแปรทางไกลได้ทีละตัว แต่ DRN สามารถเข้าถึงตัวแปรและทรัพยากรบนโหนดทางไกลที่ถูกตั้งชื่อเชิงประกาศได้พร้อมๆกันแบบขนาน

มีงานวิจัยจำนวนมากซึ่งเป็นการโปรแกรมแบบผสมระหว่าง Declarative และ Imperative Programming ยกตัวอย่างเช่น Embedded SQL [7] และ Constraint-Imperative Programming [8] ใน Embedded SQL ภาษา SQL จะถูกใช้เป็นภาษาหลักในการเข้าถึงฐานข้อมูล ส่วนการประมวลผลข้อมูลจะใช้ Imperative Programming ในแง่มุมหนึ่ง ทรัพยากรใน DRN นั้นเทียบได้กับฐานข้อมูลใน Embedded SQL ซึ่งการเข้าถึงเชิงประกาศนับว่าเหมาะสมดี สำหรับ Constraint-Imperative Programming ตัวแปรจะ

ถูกจำกัดด้วยเงื่อนไขของค่าที่เป็นไปได้หรือยอมรับได้ เนื่องจากเงื่อนไขนั้นถูกระบุเชิงประกาศ ตัวแปรทรัพยากรของเราจึงคล้ายกับตัวแปรที่ถูกจำกัดในงานดังกล่าว ถึงแม้ว่างานเราอาจมีบางแง่มุมที่คล้ายกับงานเหล่านี้ แต่ Embedded SQL และ Constraint-Imperative Programming มุ่งเน้นในการแก้ปัญหาที่แตกต่างกับของเรา ทำงานบนระบบที่ต่างกัน มีสภาวะแวดล้อมการทำงานที่ต่างกัน กล่าวคือ Embedded SQL ถูกออกแบบมาสำหรับการประมวลผลข้อมูลบนฐานข้อมูลปกติทั่วไป ส่วน Constraint-Imperative Programming ถูกออกแบบมาสำหรับการหาคำตอบที่ตรงตามข้อจำกัดในระบบทั่วไป ในทางตรงกันข้าม DRN มุ่งเน้นการตั้งชื่อและเข้าถึงทรัพยากรบนเครือข่ายไร้สายของระบบฝังตัวที่มีพลวัตสูง

X-Tree [5] เป็นงานที่เน้นการโปรแกรมเชิงมหภาคแบบผสมสำหรับระบบตัวรับรู้ในเครือข่ายระยะไกล มีลักษณะคล้ายกับ TAG ตรงที่มีการโปรแกรมระบบทั้งหมดเสมือนฐานข้อมูลตัวหนึ่ง ซึ่งแตกต่างจากงานเราที่เน้นการโปรแกรมระบบทั้งหมดเสมือนเครื่องๆเดียว นอกจากนี้ X-Tree ถูกออกแบบมาสำหรับการโปรแกรมอุปกรณ์บนอินเทอร์เนต แต่ DRN ถูกออกแบบมาสำหรับการโปรแกรมเครือข่ายไร้สายของระบบฝังตัวซึ่งมีพลวัตสูงในสภาวะแวดล้อมที่ยากลำบากและไม่เป็นมิตร

บทที่ 6

บทสรุป

เรามีความเชื่อว่าการที่พัฒนาโปรแกรมประยุกต์สำหรับเครือข่ายไร้สายของระบบฝังตัวให้มีประสิทธิภาพนั้น การใช้ Programming Abstraction ที่เหมาะสมเป็นเรื่องที่สำคัญและจำเป็นมาก DRN นับเป็น Abstraction ดังกล่าวที่ประสานรวมเงื่อนไขเชิงประกาศเข้ากับ Imperative Construct เพื่อสร้างวิธีการเขียนโปรแกรมที่ทรงพลังเหมาะสมสำหรับการโปรแกรมเชิงมหภาคเครือข่ายไร้สายของระบบฝังตัว เราได้ทำการสร้างระบบดังกล่าวโดยใช้ Smart Message (SM) ที่สามารถทำงานได้บน iPAQ ซึ่งติดต่อสื่อสารกันโดยใช้คลื่นวิทยุแบบ 802.11 เราเลือกพัฒนาระบบบน SM เนื่องจาก SM สนับสนุนการอพยพโปรแกรมซึ่งมีความจำเป็นต่อการโปรแกรมเครือข่ายในภายหลังได้ในขณะทำงานจากทางไกล ถึงแม้ว่าจะติดตั้งใช้งานไปแล้วก็ตาม สำหรับแพลตฟอร์มการโปรแกรมอีกครั้งในภายหลังจากทางไกล ยังมีอยู่อีกบ้าง ไม่ว่าจะเป็น SensorWare, Mate, และ TML ซึ่งสามารถนำมาใช้พัฒนาระบบของเราได้เช่นกัน นอกจากนี้ Abstraction ของเรานั้นเป็นอิสระต่อเครือข่ายและแพลตฟอร์มข้างล่าง ดังนั้น งานของเราน่าที่สามารถนำไปประยุกต์ใช้เพื่อโปรแกรมเชิงมหภาคเครือข่ายไร้สายหรือใช้สายอื่นๆได้ด้วย

ยิ่งกว่านั้น เราได้สร้างโปรแกรมประยุกต์ติดตามวัตถุโดยใช้ DRN เพื่อแสดงให้เห็นว่าโมเดลการโปรแกรมของเรานั้นสามารถใช้งานได้จริง เราได้ทำการประเมินผล DRN และ ตัวปรับแต่งของ DRN (ได้แก่ อายุขัยการผูกทรัพยากร) บนเครือข่ายของเครื่องเสมือน 20 เครื่อง ตัวปรับแต่งของเราหากใช้อย่างเหมาะสมสามารถทำให้โปรแกรมประยุกต์ประหยัดจำนวนไบต์ในการส่งได้ถึง 55.2% โดยไม่ทำให้ความแม่นยำแย่ลงอย่างมีนัยสำคัญเมื่อโค้ดของโปรแกรมประยุกต์ได้ถูกแคชหรือถูกติดตั้งเรียบร้อยแล้วในเครือข่าย

เอกสารอ้างอิง

[1] Philippe Bonnet, Johannes Gehrke, Tobias Mayr, and Praveen Seshadri. Query processing in a device database system. Technical Report TR99-1775, Cornell University, October 1999.

[2] Cristian Borcea, Chalermek Intanagonwiwat, Porlin Kang, Ulrich Kremer, and Liviu Iftode. Spatial programming using smart messages: Design and implementation. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS 2004)*, Tokyo, Japan, March 2004.

[3] Cristian Borcea, Deepa Iyer, Porlin Kang, Akhilesh Saxena, and Liviu Iftode. Cooperative Computing for Distributed Embedded Systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS)*, pages 227–236, July 2002.

[4] A. Boulis, C.Han, and M. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In *Proceedings of the First International Conference on Mobile Systems, Applications, and Services (Mobisys 2003)*, pages 187–200, San Francisco, CA, May 2003.

[5] Shimin Chen, Phillip B. Gibbons, and Suman Nath. Database-centric programming for wide-area sensor systems. In *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2005.

[6] Dan Coffin, Dan Van Hook, Ramesh Govindan, John Heidemann, and Fabio Silva. Network routing application programmer's interface (api) and walk through 8.0. Technical Report 01-741, USC/ISI, March 2001.

[7] Oracle Corporation. Pro*c/c++ precompiler programmer's guide release 9.2, 2002.

[8] Martin Grabmuller. *Constraint Imperative Programming*. Diploma Thesis, Technische Universitat Berlin, 2003.

[9] Ramakrishna Gummadi, Omprakash Gnawali, and Ramesh Govindan. Macro-programmingwireless sensor networks using kairos. In *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2005.

[10] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin,

and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.

[11] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences*, Maui, Hawaii, January 2000.

[12] Liviu Iftode, Cristian Borcea, Andrzej Kochut, Chalermek Intanagonwiwat, and Ulrich Kremer. Programming computers embedded in the physical world. In *Proceedings of the 9th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS)*, San Juan, Puerto Rico, May 2003.

[13] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of the International Conference on Distributed Computing Systems*, Vienna, Austria, July 2002. IEEE.

[14] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'2000)*, Boston, Massachusetts, August 2000.

[15] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva, Directed Diffusion for Wireless Sensor Networking. *IEEE/ACM Transactions on Networking*, Volume 11, Issue 1, Pages 2-16. February, 2003.

[16] Ulrich Kremer, Liviu Iftode, Jerry Hom, and Yang Ni. Spatial Views: Iterative Spatial Programming for Networks of Embedded Systems. Technical Report DCSTR-493, Rutgers University, June 2002.

[17] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. The impact of data aggregation in wireless sensor networks. In *DEBS'02*, pages 575–578, Vienna, Austria, July 2002.

[18] P. Levis and D. Culler. A tiny virtual machine for sensor networks. In *Proceedings of the ACM Conference on Architectural Support for Programming Languages and Operating Systems (APLOS)*, October 2002.

[19] Samuel Madden, Michael Franklin, Joseph Hellerstein, and Wei Hong. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

[20] Ryan Newton, Arvind, and Matt Welsh. Building up to macroprogramming: An intermediate language for sensor networks. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN'05)*, April 2005.

[21] The Castle project. http://www.cs.berkeley.edu/projects/parallel/castle/split-c/.

[22] Matt Welsh. Exposing resource tradeoffs in region based communication abstractions for sensor networks. In *Proceedings of the 2nd ACM Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

[23] MattWelsh and Geoff Mainland. Programming sensor networks using abstract regions. In *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*, March 2004.

Output จากโครงการวิจัยที่ได้รับทุนจาก สกอ. และ สกว.

1. ผลงานตีพิมพ์ในวารสารวิชาการนานาชาติ (ระบุชื่อผู้แต่ง ชื่อเรื่อง ชื่อวารสาร ปี เล่มที่ เลขที่ และหน้า)
   - กำลังอยู่ในระหว่างการจัดทำ จะส่งตามมาภายหลัง

2. การนำผลงานวิจัยไปใช้ประโยชน์
   - เชิงวิชาการ (มีการพัฒนาการเรียนการสอน/สร้างนักวิจัยใหม่) แนวคิดการโปรแกรมเชิงมหภาคนี้ นำไปสู่การสร้างนักวิจัยทางด้านนี้มากขึ้น โดยเฉพาะอย่างยิ่งนิสิตในที่ปรึกษาของหัวหน้าโครงการวิจัยนี้ ทั้งในระดับปริญญาตรี โท และ เอก ดังมีรายชื่อดังนี้
     (a) นายภูริ เฉลิมเกียรติสกุล
     (b) นายณภัทร รัตนศิรินทรวุธ
     (c) นายศุภเสฏฐ์ ชูชัยศรี