บทที่ 2

หลักการและทฤษฎี

ในบทนี้จะบรรยายถึงหลักการและทฤษฎีที่เป็นพื้นฐานสำหรับการทำวิทยานิพนธ์นี้ ซึ่งจะเกี่ยวข้องกับพื้นฐานที่เกี่ยวข้องกับการจำลองแบบกระจาย ด้วยโครงร่างสถาปัตยกรรมระดับสูง แนวคิดการจำลองอ็อบเจกต์และการนิยามอ็อบเจกต์ การส่งผ่านข้อมูลระหว่างหน่วยการจำลอง และการจัดการเวลา

2.1 พื้นฐานการจำลองแบบกระจายด้วยโครงร่างสถาปัตยกรรมระดับสูง

HLA เป็นข้อกำหนดมาตรฐานสำหรับการจำลองแบบกระจาย ทำให้หน่วยการจำลองสามารถทำงานร่วมกันได้ แม้ว่าจะอยู่ต่างสภาพแวดล้อมกัน HLA ถูกพัฒนาโดยกระทรวงกลาโหมประเทศสหรัฐอเมริกา ภายใต้หน่วยงานย่อยที่เรียกว่า Defense Modeling and Simulation Office (DMSO) เพื่อช่วยในการจำลองแบบกระจาย โดยในระยะแรกเน้นด้านการทหารเป็นหลัก (Kuhl, Weatherly and Dahmann, 1999)

ในอดีตแบบจำลอง (Simulation Model) เมื่อถูกออกแบบและสร้างบนโปรแกรมจำลอง (Simulator) ใดๆ จะไม่สามารถย้ายไปยังโปรแกรมจำลองอื่นๆได้ ทำให้การนำตัวแบบการจำลองมาทำงานร่วมกันเป็นไปได้ยาก และผลกระทบลูกโซ่คือตัวแบบการจำลองมีอัตราการนำมาใช้ใหม่ต่ำ HLA จึงถูกสร้างขึ้นมาเพื่อแก้ปัญหาดังกล่าว (IEEE 1516-2000, 2000) ซึ่งพอจะกล่าวแยกได้ดังนี้

ประการแรก เรื่องการทำงานร่วมกัน (Interoperability) ในระบบการจำลองแบบกระจายโดยทั่วไป มักจะประสบปัญหาเรื่องของความเข้ากันได้ระหว่างหน่วยการจำลองย่อย อันเนื่องมาจากการผูกติดกับระบบ (System Dependence) เช่น ภาษาช่วยการจำลอง ระบบปฏิบัติการ หรือ รูปแบบการประสานเวลา (Synchronization) ที่ต่างกัน เป็นต้น สำหรับระบบการจำลองที่เป็นไปตาม HLA สามารถแก้ปัญหาเหล่านี้ได้ซึ่งจะนำไปสู่การทำงานร่วมกันได้อย่างอิสระ

ประการที่สอง การนำกลับมาใช้ใหม่ (Reusability) กล่าวคือ ในระบบการจำลองที่เป็นไปตามHLA ย่อมสามารถนำหน่วยการจำลองย่อยที่ได้ออกแบบและสร้างไว้แล้วมาใช้ใหม่ โดยไม่จำเป็นต้องมีการแก้ไขโปรแกรมอีก เนื่องจาก HLA เน้นให้การออกแบบโปรแกรมการจำลองให้เป็นแบบเชิงอ็อบเจกต์ (Object-Oriented) ซึ่งจะมีอัตราการนำกลับมาใช้ใหม่สูงกว่าการออกแบบ

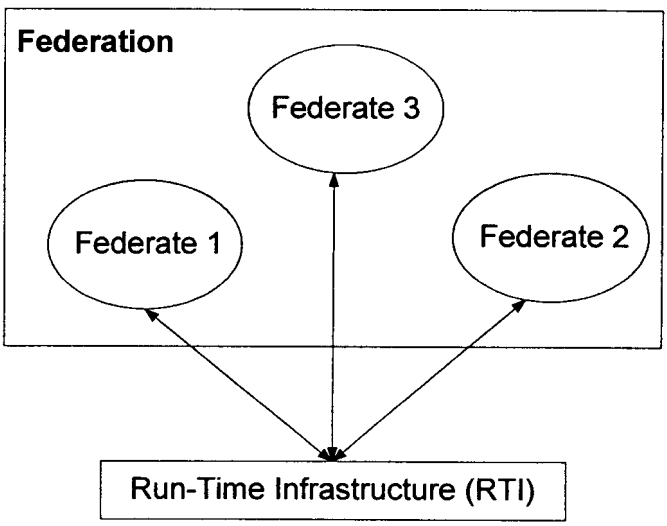
เชิงโครงสร้าง ทำให้ลดความซับซ้อนในการพัฒนาระบบการจำลองขนาดใหญ่ลงได้

HLA ได้กำหนดข้อกำหนดต่อไปนี้เพื่อให้บรรลุความปรารถนาสองประการข้างต้น

1. HLA Rule (IEEE Standard 1516-2000, 2000) คือ กฎพื้นฐานสำหรับส่วนการจำลองที่จะต้องทำงานร่วมกัน โดยจะอธิบายถึงภาระหน้าที่ ความรับผิดชอบของแต่ละหน่วยการจำลอง (Federate) และกลุ่มการจำลอง (Federation)

2. Federate Interface Specification (IEEE Standard 1516.1-2000, 2000) คือ การกำหนดส่วนต่อประสานโปรแกรมประยุกต์ (Application Program Interface-API) เช่น การกำหนดว่า หาก Federate หนึ่งต้องการส่งข้อมูลไปสู่ Federate อื่นๆ จะมีรูปแบบการเรียกใช้คำสั่งอย่างไร ซึ่งในทางปฏิบัติจะถูกกำหนดอยู่ในส่วนของ RTI Ambassador และ Federate Ambassador

3. Object Model Template (OMT) Specification (IEEE Standard 1516.2-2000, 2000) คือ การกำหนดวากยสัมพันธ์ของแบบจำลองอ็อบเจกต์ (Object Model) ที่ใช้ใน HLA ซึ่งเป็นวิธีการที่ทำให้หน่วยการจำลองทั้งหมดเข้าใจรูปแบบข้อมูลที่จะต้องแลกเปลี่ยนกัน นำไปสู่ความสามารถในการทำงานร่วมกันได้แม้จะอยู่ต่างแพลตฟอร์มกัน

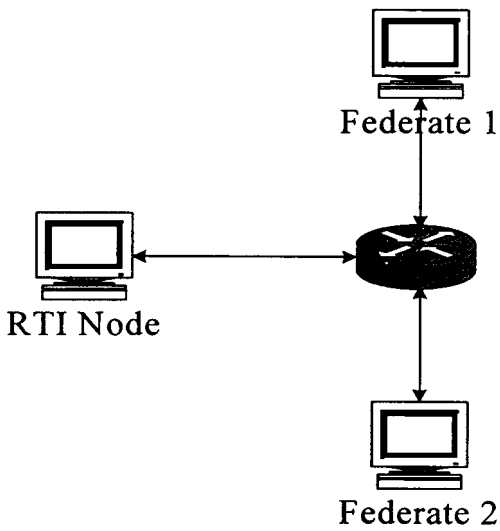


รูปที่ 2.1 แสดงองค์ประกอบเชิงกายภายของการจำลองแบบกระจายด้วย HLA

จากข้อกำหนดต่างๆ พอที่จะสรุปออกมาเป็นภาพรวม ดังแสดงในรูปที่ 2.1 ซึ่งแสดงองค์ประกอบเชิงกายภายของการจำลองแบบกระจายด้วย HLA ประกอบด้วย 3 ส่วนดังนี้

1. Federate คือ หน่วยการจำลองซึ่งจะรับภาระในการคำนวณ จากรูปที่ 2.1 ประกอบด้วย 3 Federate
2. Federation คือ การจำลองร่วมกันของ Federate โดยระหว่าง Federate จะมีการส่งข้อมูลและเหตุการณ์ถึงกันตลอดเวลา กลายเป็นกลุ่มของหน่วยการจำลอง
3. Run Time Infrastructure (RTI) คือ ตัวกลางที่ทำให้แต่ละ Federate สามารถทำงานร่วมกันอย่างเข้าจังหวะ (Synchronization)ได้ อย่างไรก็ดีสำหรับ RTI สามารถประสานการทำงานได้มากกว่า 1 Federation

HLA กำหนดให้การออกแบบแบบจำลองเป็นแบบเชิงอ็อบเจกต์ (Fujimoto, 2000) ซึ่งแต่ละอ็อบเจกต์อาจจะแทนวัตถุในโลกจริง เช่น การจำลองการทำงานของสนามบินประกอบด้วยอ็อบเจกต์ของเครื่องบิน และ อ็อบเจกต์ของลานบิน ซึ่งจะถูกกระจายไปตามแต่ละ Federate ซึ่งขณะทำการจำลองจะต้องรับภาระในการคำนวณและส่งข้อมูลการเปลี่ยนแปลงสถานะของอ็อบเจกต์ให้ Federate อื่นรับรู้ อย่างไรก็ตามแต่ละ Federate จะรับผิดชอบอ็อบเจกต์ได้อย่างไม่จำกัด ขึ้นอยู่กับผู้ออกแบบระบบ



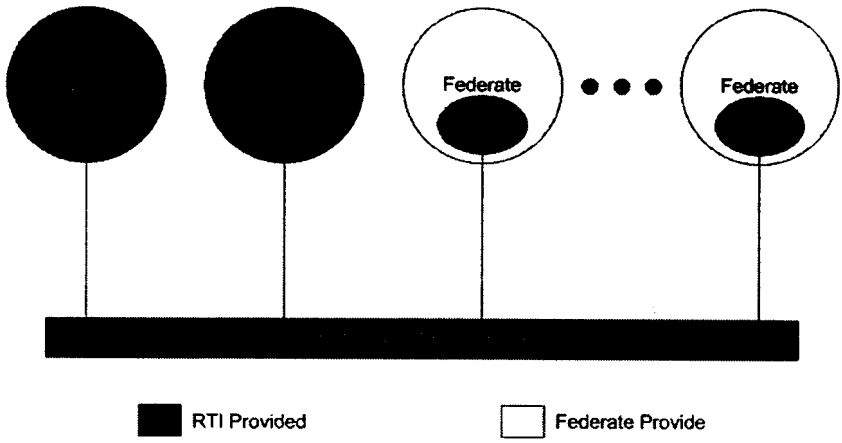
รูปที่ 2.2 แสดง HLA Federation ในการทำงานจริง

รูปแบบของ HLA Federation ในการใช้งานจริงก็จะทำงานประสานงานผ่านทางเครือข่ายคอมพิวเตอร์จากรูปที่ 2.2 แสดงให้เห็นภายใน Federation ประกอบด้วย 2 Federate ซึ่งทุก Federate ถูกกระจายออกให้อยู่ในเครื่องคอมพิวเตอร์แยกออกจากกันเป็นอิสระ แต่ในเชิงหลักการ Federate สามารถอยู่ภายในคอมพิวเตอร์เครื่องเดียวกันก็ได้ หรืออาจมองได้ว่า Federate คือ โปรเซส

ที่ทำงานแยกออกจาก Federate อื่นอย่างอิสระเพื่อทำการจำลองในส่วนที่ตนได้รับมอบหมาย โดยทั้งนี้เมื่อทำการคำนวณหรือผลิตเหตุการณ์ออกมาจะถูกส่งต่อให้กับ RTI ซึ่งทำหน้าที่เป็นผู้ประสานการทำงานทั้งหมด

ภาระความรับผิดชอบของ RTI นั้นไม่ได้จำกัดบริเวณอยู่ภายใน RTI node เท่านั้นแต่บางส่วนต้องกระจายอยู่ใน Federate ด้วย โดยแยกได้เป็น 3 ส่วน (แสดงด้วยสีทึบในรูป 2.3) คือ

1. RtiExec เป็นโปรเซสหลักสำหรับควบคุมการทำงานภายในทั้งหมดทุกๆ Federation
2. FedExec เป็นโปรเซสที่ใช้เป็นส่วนในการควบคุมการทำงานภายใน Federation
3. libRTI เป็นส่วนการทำงานที่ใช้เชื่อมต่อระหว่าง Federate กับ RTI node



รูปที่ 2.3 แสดงการแบ่งแยกส่วนของ RTI และ Federate

(ที่มา : HLA-RTI1.3-NG Programmer's Guide Version 5, 2002)

สรุปภาพรวมของ HLA Federation ออกมาได้เป็นแบบจำลองคร่าวๆ ดังรูปที่ 2.4 ซึ่งประกอบด้วย 3 ส่วน(Kolek, Boswell and Wolfson, 2000) คือ

1. แบบจำลองทางเครือข่าย (Network Model) จะแทนรูปแบบการประสานงานระหว่าง Federate และ RTI ซึ่งในวิทยานิพนธ์นี้ได้อาศัยการนำซอฟต์แวร์ ns ซึ่งมีแบบจำลองทางเครือข่ายอยู่แล้ว

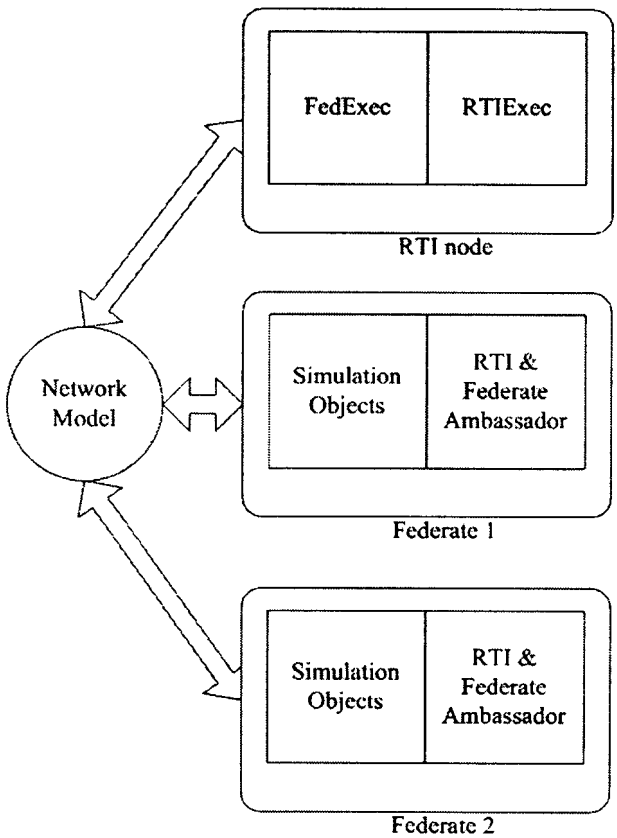
2. แบบจำลอง RTI (RTI Model) จะแทนการทำงานของ RTI

3. แบบจำลอง Federate (Federate Model) จะแทนการทำงานของ Federate และ พฤติกรรมการส่งข้อมูลของอ็อบเจกต์การจำลอง (Simulation Object)

จากรูป 2.4 Federate 1 และ 2 ซึ่งเป็นตัวอย่างของ Federate Model มีองค์ประกอบที่ใช้ในการทำงานสำหรับหน่วยการจำลองย่อยของตนเองอยู่ 2 ส่วน คือ Simulation Model เป็นส่วนของ

รูปแบบรายละเอียดการจำลองภายใน Federate นั้น ๆ และ RTI & Federate Ambassador เป็นส่วนโครงสร้างพื้นฐานที่ใช้เป็นช่องทางในการติดต่อรับส่งข้อมูลกันระหว่าง RTI node และ Federate

หากทำการขยายส่วนของ Federate ก็จะเห็นรายละเอียดภายใน ซึ่งประกอบด้วย RTI Ambassador จะถูกรับหน้าที่เป็นประหนึ่งทูตสำหรับติดต่อกับ RTI โดยหาก Simulation Object ต้องการส่งข้อมูลออก ก็สามารถเรียกใช้บริการจาก RTI Ambassador ได้ และส่วนของ Federate Ambassador ซึ่งจะรับข้อมูลจาก RTI เพื่อทำการส่งต่อให้ Virtual Object ซึ่งหมายถึง อ็อบเจกต์ที่ประมวลผลที่ Federate อื่น แต่สามารถถูกมองเห็นหรือให้ข้อมูลผลลัพธ์จากการประมวลผลแก่ Federate อื่นๆ

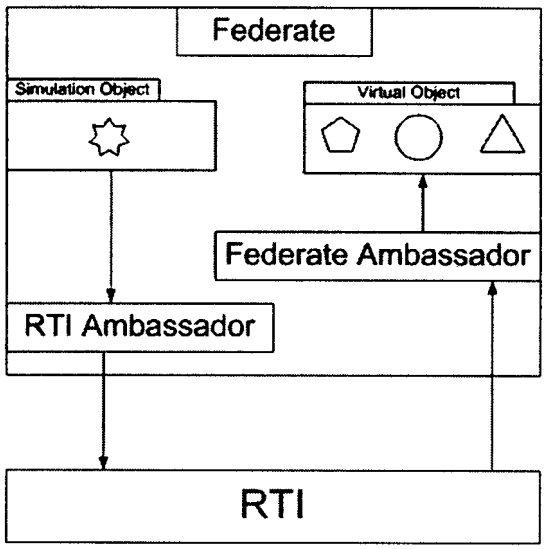


รูปที่ 2.4 แสดงองค์ประกอบสำคัญของแบบจำลองของ HLA

รูปที่ 2.5 เป็นจุดมุ่งหมายหลักประการหนึ่งของการจำลองแบบกระจายที่ใช้ HLA นั่นก็คือ การแยก Simulation Object ออกจากโครงสร้างพื้นฐานการจำลองอื่นๆ หรืออาจกล่าวว่าข้อกำหนดมาตรฐาน HLA ทำให้ระบบการจำลองยังคงทำงานติดต่อสื่อสารกันได้แม้จะถูกสร้างมาต่างแพลตฟอร์มกันก็ตาม ซึ่งทำให้เกิดผลในแง่การทำงานร่วมกัน และ การนำกลับมาใช้ใหม่ โดยการ

กำหนดให้ RTI Ambassador และ Federate Ambassador เป็นซอฟต์แวร์อีกชั้นหนึ่งรองรับการทำงานของ Simulation Object ซึ่งเป็นซอฟต์แวร์ตัวแทนของตัวแบบการจำลอง เป็นการบังคับให้ Simulation Object ต้องเรียกส่วนเชื่อมต่อที่เป็นมาตรฐาน ซึ่งง่ายต่อการนำกลับมาใช้ใหม่ในอนาคต

ทั้งนี้ในทางปฏิบัติ RTI Ambassador และ Federate Ambassador เป็นคลาสต้นแบบซึ่งจะถูกสร้างขึ้นอ้างอิงตาม interface specification (IEEE 1526.2-2000) ที่ถูกจัดเตรียมไว้แล้ว ทั้งนี้ส่วนของซอฟต์แวร์ Federate จะต้องทำการสืบทอดคลาสดังกล่าวก่อนการใช้งาน



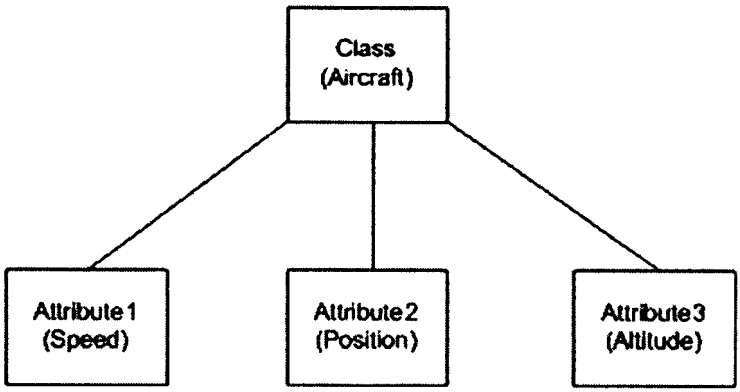
รูปที่ 2.5 แสดงองค์ประกอบภายใน Federate

2.2 อ็อบเจกต์ใน HLA

เนื่องจากตามแนวคิดของ HLA ได้นำแนวคิดของการออกแบบเชิงอ็อบเจกต์เข้ามาเกี่ยวข้องด้วยจำนวนมาก ดังนั้นในส่วนนี้จะขออธิบายแนวคิดดังกล่าว โดยจะเริ่มจากแนวคิดการนิยามคลาส การสืบทอด เพื่อจะโยงสู่แนวคิดการประกาศ (Publication) คลาสและสมัครสมาชิก (Subscription) ของคลาสซึ่งเป็นส่วนหนึ่งในแนวคิดเบื้องต้นของ HLA

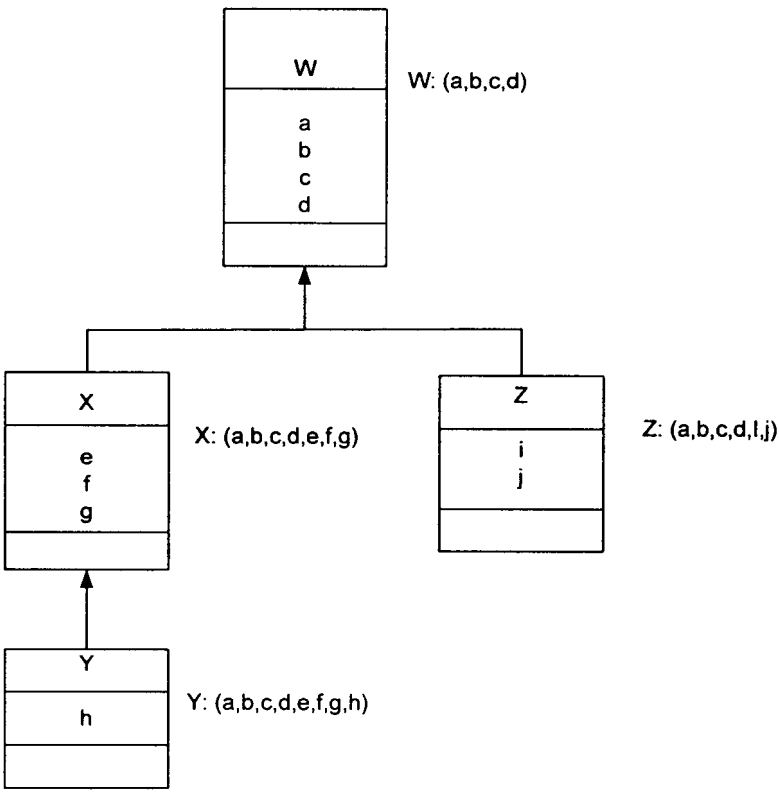
2.2.1 คลาสและการนิยาม

คลาสใดๆ จะต้องถูกนิยามลักษณะหรือคุณสมบัติประจำนั้น ๆ เช่น คลาสของเครื่องบินอาจจะมีคุณสมบัติของความเร็ว ตำแหน่ง และระดับความสูง ดังรูปที่ 2.6 ขณะที่ Federate ทำการคำนวณเพื่อทำการจำลอง คุณสมบัติต่างๆ ของอ็อบเจกต์ก็จะถูกเปลี่ยนค่าอยู่ตลอดเวลา



รูปที่ 2.6 แสดงคลาสและคุณสมบัติของคลาส

2.2.2 คลาสและการสืบทอด

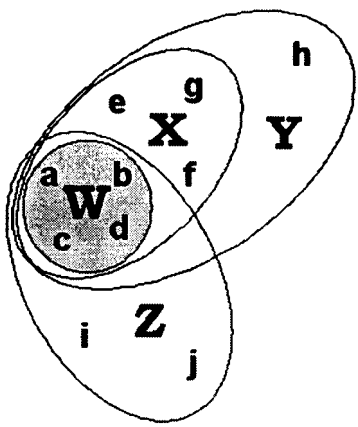


รูปที่ 2.7 แสดงแผนผังลำดับชั้นของคลาส

(ที่มา : HLA-RTI1.3-NG Programmer's Guide Version 5, 2002)

การสืบทอด (Inheritance) คลาสสามารถแสดงด้วยแผนผังลำดับชั้น (Class Hierarchy) ดังแสดงในรูป 2.7 และแผนภาพของเวน (Venn diagram) ดังแสดงในรูป 2.8 ยกตัวอย่างเช่น คลาส X สืบทอดจาก W จึงได้รับคุณสมบัติทั้งหมดจาก W และนอกจากนี้ X ยังมีสมบัติเพิ่มเติมคือ e f และ g ในกรณีนี้ เราเรียก W ว่า คลาสบรรพบุรุษ (parent class) ของคลาส X และ เรียกคลาส X ว่า คลาสลูก (Child Class) ของคลาส W

การสืบทอดสามารถกระทำได้หลายระดับไม่มีที่สิ้นสุด การแสดงการสืบทอดสามารถแสดงได้อีกลักษณะหนึ่งคือในรูปของแผนภาพของเวนน์ซึ่งจะทำให้สังเกตได้ง่ายในกรณีต้องการมองในแง่การเป็นสับเซตหรือซูเปอร์เซต เช่น W เป็นสับเซตของของทุกเซต



รูปที่ 2.8 แสดงแผนภาพของเวนน์

(ที่มา : HLA-RTI1.3-NG Programmer's Guide Version 5, 2002)

2.2.3 การประกาศและสมัครสมาชิก

การประกาศและการสมัครสมาชิก เป็นหลักการอันสำคัญของ HLA เพราะต้องการลดจำนวนข้อมูลที่จะต้องถูกส่งไปมาภายในเครือข่ายอันก่อนให้เกิดความสูญเปล่าของทรัพยากรด้านเครือข่าย (Tacic and Fujimoto, 1997) ดังนั้น HLA ได้กำหนดให้มีกระบวนการประกาศและสมัครสมาชิกขึ้น ซึ่งพอจะอธิบายได้ดังนี้

1. การประกาศ คือการบอกให้ทราบโดยทั่วกันภายใน Federation ว่าต้องการแจกจ่ายข่าวสารการเปลี่ยนแปลงค่าคุณสมบัติของอ็อบเจกต์ เช่น Federation 1 ประกาศคลาส W และ X แสดงว่า Federate 1 ต้องการแจกจ่ายข่าวสารการเปลี่ยนแปลงของทุกอ็อบเจกต์ที่เกิดจากคลาส W และ X

2. การสมัครสมาชิก คือการบอกรับเป็นสมาชิกของคลาส ซึ่งจะได้รับข่าวสารการเปลี่ยนแปลงค่าของคุณสมบัติของอ็อบเจกต์ เช่น หาก Federate 2 บอกรับเป็นสมาชิกคลาส W ทุกๆ ครั้งที่อ็อบเจกต์ของคลาส W เกิดการเปลี่ยนแปลงค่าคุณสมบัติ อ็อบเจกต์ของคลาส W ก็จะส่งข่าวสารของการเปลี่ยนแปลงไปยัง Federate 2 หากพิจารณาในทางกลับกันหากคลาส W ไม่ถูกสมัครสมาชิกโดย Federate 2 ก็ไม่มีความจำเป็นใดๆที่ Federate 2 จะต้องรับรู้ข่าวสารของอ็อบเจกต์ของคลาส W

กระบวนการประกาศและสมัครสมาชิกนั้นแท้จริงอาศัยการทำงานของ RTI ทั้งสิ้น กล่าวคือระบบจะมอบหมายให้ RTI ทำหน้าที่เป็นตัวกรองข่าวสารเพื่อลดความซ้ำซ้อนของข้อมูล ดังนั้น จึงอาจกล่าวได้ว่า ทุกๆ การประกาศและ ทุกๆ การสมัครสมาชิกล้วนแล้วมีกระบวนการภายในที่ต้องแจ้งให้ RTI ทราบทั้งสิ้น เพื่อให้ RTI ทำการกรองข่าวสารได้ถูกต้อง และสำหรับเรื่องกระบวนการภายในของ RTI จะกล่าวถึงโดยละเอียดภายหลัง

ในรูปที่ 2.9 เป็นกรณีที่ซับซ้อนขึ้นของการประกาศและสมัครสมาชิก ซึ่งจะอธิบายลำดับเหตุการณ์เรียงตามลำดับหมายเลขดังต่อไปนี้

เหตุการณ์ที่ 1 คือ เหตุการณ์ที่ Federate 1 ประกาศคลาส X และ Z

เหตุการณ์ที่ 2 คือ เหตุการณ์ที่ RTI รับรู้ถึงการประกาศ คลาส X และ Z ดังแสดงโดยเส้นทึบ หมายความว่า RTI ยอมรับการมีตัวตนของคลาส X และ Z เท่านั้น

เหตุการณ์ที่ 3 คือ การที่ Federate 2 ต้องการสมัครเป็นสมาชิกคลาส Y แต่เนื่องจากคลาส Y ไม่ได้ถูกประกาศ มีเพียงคลาสแม่ของคลาส Y คือ คลาส X เท่านั้นที่ถูกประกาศ ดังนั้น Federate 2 จึงมองเห็นอ็อบเจกต์ของคลาส Y ในรูปของอ็อบเจกต์ของคลาส X

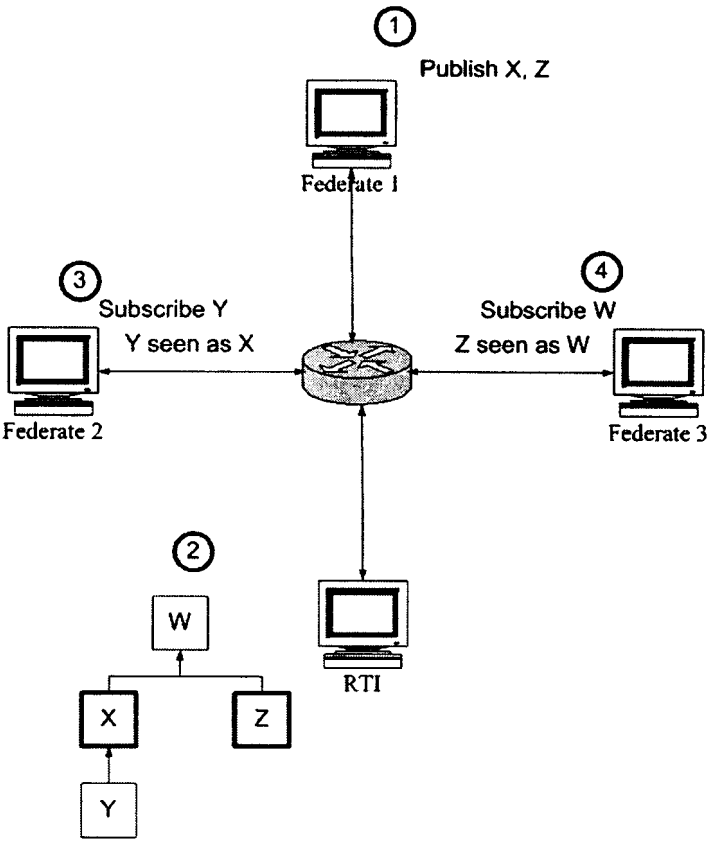
เหตุการณ์ที่ 4 คือ การที่ Federate 3 สมัครเป็นสมาชิกคลาส W เท่านั้น แม้ว่าคลาส Z ถูกประกาศออกมา แต่ Federate 3 จะมองเห็นอ็อบเจกต์ของคลาส Z ในรูปอ็อบเจกต์ของคลาส W เท่านั้น

จากคำอธิบายรูป 2.9 ได้สะท้อนถึงจุดมุ่งหมายอันสำคัญยิ่งของของกระบวนการประกาศและสมัครสมาชิก 3 ประการ คือ

1. การลดจำนวนข้อมูลที่ต้องการส่งไปมาภายในเครือข่าย (Tacic and Fujimoto, 1997)

2. การสร้างขอบเขตของการรับรู้ของแต่ละ Federate เช่น คลาส Y ซึ่งสืบทอดจาก X แต่ถ้าคลาส Y มีคุณสมบัติบางอย่างที่เป็นความลับ เป็นไปได้ว่า Federate 1 อาจประกาศคลาส X เท่านั้นเพื่อให้ Federate อื่นๆ มองเห็นคลาส Y เป็น X

3. การสร้างความยืดหยุ่นให้กับการออกแบบระบบ เช่น การที่คลาส Z ถูกเพิ่มเข้าสู่ระบบได้แต่ Federate 3 ก็ยังไม่ต้องปรับโปรแกรมแต่อย่างใด



รูปที่ 2.9 แสดงตัวอย่างการสมัครสมาชิกและการสมัครสมาชิก

2.3 การส่งผ่านข้อมูลใน Federation

การส่งผ่านข้อมูลภายใน Federation ถูกกำหนดไว้ในโครงร่างสถาปัตยกรรมระดับสูงไว้ 2 บริการด้วยกัน คือ ในด้าน Object Management เป็นการกำหนดการส่งผ่านข้อมูลเชิงคลาส (Class-Base Filtering) และ Data Distribution Management เป็นการกำหนดการส่งผ่านข้อมูลเชิงค่า (Value-Base Filtering) ทั้งสองหัวข้อจะอธิบายโดยลำดับดังนี้

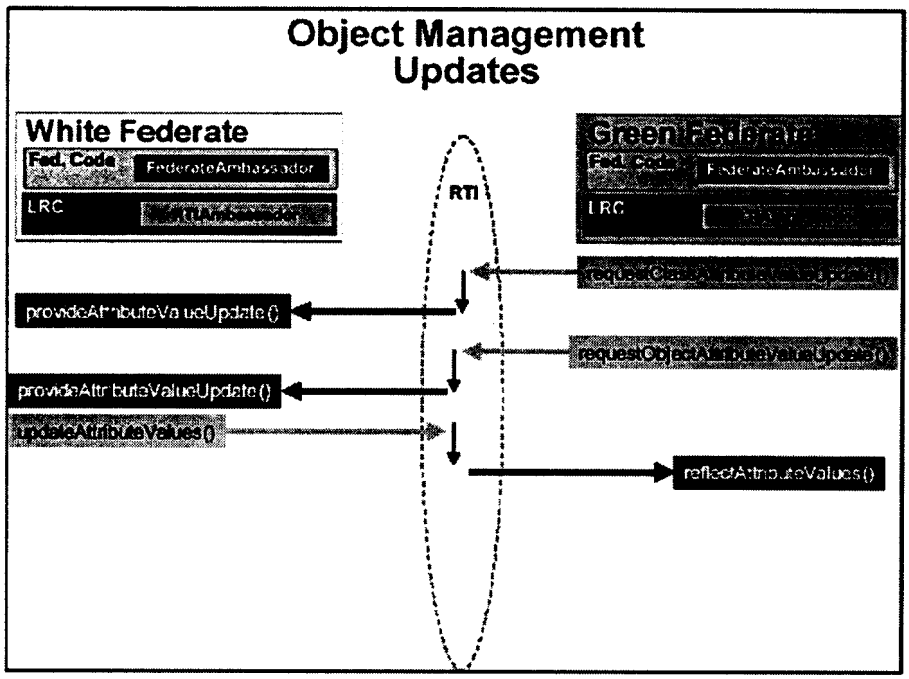
2.3.1 การส่งผ่านข้อมูลเชิงคลาสใน Object Management

โดยหลักการแล้วแต่ละ Federate จะแลกเปลี่ยนข้อมูลระหว่างกันผ่าน RTI โดยการเชื่อมต่อผ่านระบบเครือข่ายดังที่กล่าวไว้แล้วข้างต้น ข้อมูลที่แลกเปลี่ยนกันระหว่าง Federate คือข้อมูลอัน

เกิดจากการเปลี่ยนแปลงของอ็อบเจกต์ของคลาสหลังจากอ็อบเจกต์ผ่านการคำนวณแล้ว ในรายละเอียดของกระบวนการแลกเปลี่ยนข้อมูลแยกได้ 2 กระบวนการย่อยดังนี้

1. การปรับปรุงข้อมูล (Update Attribute Value) คือ กระบวนการที่ Federate ที่เป็นเจ้าของอ็อบเจกต์ส่งข้อมูลของอ็อบเจกต์ที่มีการเปลี่ยนแปลงออก สู่ RTI ทั้งนี้การปรับปรุงข้อมูลจะเกิดได้ก็ต่อเมื่อมีการประกาศคลาสก่อนแล้วเท่านั้น

2. การสะท้อนข้อมูล (Reflect Attribute Value) คือ กระบวนการคัดกรองข้อมูลของ RTI และ ทำการสะท้อนข้อมูลที่ได้รับเข้ามาจากกระบวนการปรับปรุงข้อมูลไปยัง Federate ที่สนใจหรือได้ทำการสมัครสมาชิกไว้แล้ว



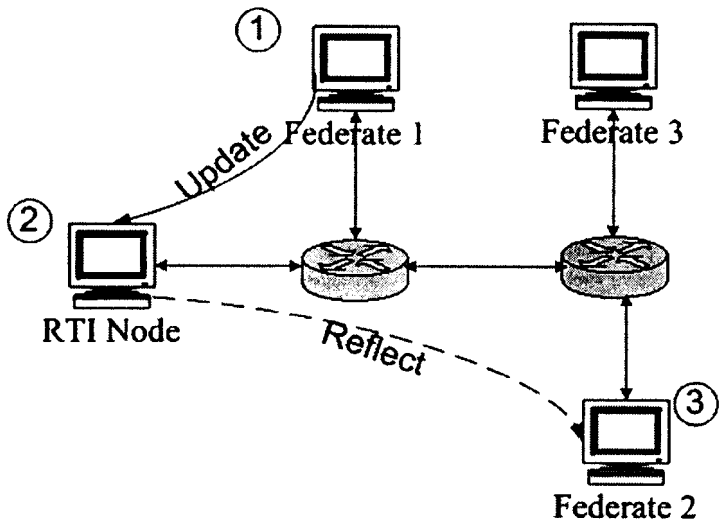
รูปที่ 2.10 แสดงการปรับปรุงและสะท้อนข้อมูล

(ที่มา : HLA-RTI1.3-NG Programmer's Guide Version 5, 2002)

รูปที่ 2.10 ได้แสดงการปรับปรุงและการสะท้อนข้อมูลที่ได้กำหนดไว้ในมาตรฐาน ซึ่งประกอบด้วย 2 กระบวนการย่อย คือ

- การเริ่มต้นกระบวนการปรับปรุงและสะท้อนข้อมูล ในส่วนนี้ประกอบด้วยส่วนการทำงานย่อย คือ requestClassAttributeValueUpdate, requestObjectAttributeValueUpdate, และ provideAttributeValueUpdat
- การปรับปรุงและสะท้อนข้อมูล ซึ่งจะมีหลักการดังที่ได้อธิบายไว้ข้างต้น และที่สำคัญคือเป็นกระบวนการที่ต่อเนื่อง โดยข้อมูลที่ถูกผลิตจะออกมาด้วยความถี่ และ ขนาด ใดๆ ซึ่ง

อาจส่งกระทบภาวะความคับคั่งของเครือข่าย (Zhao and Georganas, 2001) ซึ่งในวิทยานิพนธ์นี้จะมุ่งเน้นตรงกระบวนการปรับปรุงและสะท้อนข้อมูลเป็นหลัก โดยไม่คำนึงถึงการเริ่มต้นกระบวนการ



รูปที่ 2.11 แสดงการปรับปรุงและสะท้อนข้อมูล และมี 1 Federate สมัครสมาชิก

ตัวอย่างแสดงในรูปที่ 2.11 สถานะตั้งต้น สมมติให้ ภายใน Federation ประกอบด้วย 3 Federate ซึ่ง Federate 1 ประกาศคลาส W สู่สาธารณะ และ RTI ได้รับรู้การประกาศคลาส W แล้ว จะอธิบายกระบวนการปรับปรุงและสะท้อนข้อมูลได้ดังนี้

เหตุการณ์ที่ 1 อ็อบเจกต์ของคลาส W ถูกคำนวณและถูกปรับปรุงข้อมูลออกมา สู่ RTI

เหตุการณ์ที่ 2 RTI รับข้อมูลปรับปรุงของอ็อบเจกต์ของคลาส W แล้วทำการตรวจสอบว่าคลาส W ถูกสมัครสมาชิกโดย Federate ใดบ้าง ซึ่งพบว่าขณะนี้ Federate 2 ได้สมัครสมาชิก ดังนั้นจึงเป็นเป้าหมายของการสะท้อนข้อมูล

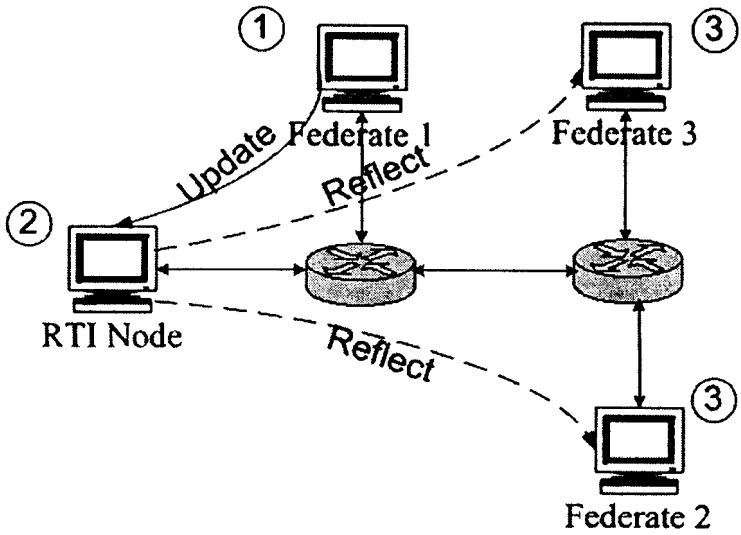
เหตุการณ์ที่ 3 เฉพาะ Federate 2 เท่านั้นรับข้อมูล

และตัวอย่างในรูปที่ 2.12 สถานะตั้งต้นกำหนดให้ดังตัวอย่างที่แล้ว

เหตุการณ์ที่ 1 อ็อบเจกต์ของคลาส W ถูกคำนวณและถูกปรับปรุงข้อมูลออกมา สู่ RTI

เหตุการณ์ที่ 2 RTI รับข้อมูลปรับปรุงของอ็อบเจกต์ของคลาส W แล้วทำการตรวจสอบว่าคลาส W ถูกสมัครสมาชิกโดย Federate ใดบ้าง ขณะนี้ Federate 2 และ Federate 3 ได้สมัครสมาชิก ดังนั้นจึงเป็นเป้าหมายของการสะท้อนข้อมูล

เหตุการณ์ที่ 3 Federate 2 และ Federate 3 รับข้อมูล เหมือนกัน

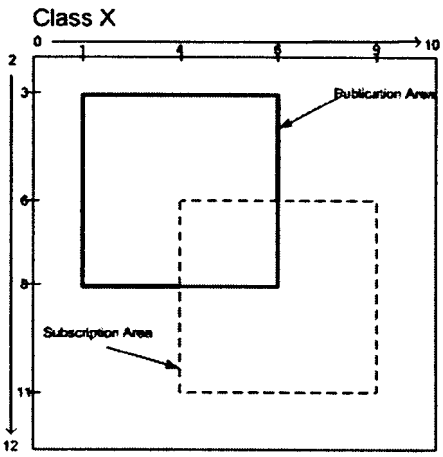


รูปที่ 2.12 แสดงการปรับปรุงและสะท้อนข้อมูล และมี 2 Federate สมัครสมาชิก

จากที่กล่าวมาทั้งหมดในหัวข้อนี้ จะเห็นว่า RTI จะส่งผ่านข้อมูลโดยอาศัยข้อมูลของการประกาศและการสมัครสมาชิกต่อคลาสเป็นแผนที่ในการชี้ทิศทางการสะท้อนข้อมูล ซึ่งถูกเรียกว่า การส่งผ่านข้อมูลเชิงคลาส ในหัวข้อต่อไปจะได้กล่าวถึงการส่งผ่านเชิงค่า

2.3.2 การส่งผ่านข้อมูลเชิงค่าใน Data Distribution Management

การส่งผ่านข้อมูลเชิงค่านั้นถูกกำหนดให้อยู่ใน Data Distribution Management โดยกระบวนการสำคัญอยู่ที่การประกาศและสมัครสมาชิกเชิงค่า ซึ่งจะแตกต่างจากที่กล่าวมาในหัวข้อที่แล้วเล็กน้อยกล่าวคือ การประกาศและสมัครสมาชิกเชิงค่า จะมีการประกาศและสมัครสมาชิกโดยคลาสและช่วงค่าของอ็อบเจกต์ ดังรูป 2.13 โดยอธิบายเชิงยกตัวอย่างได้ดังนี้คือ คลาส X มีช่วงค่าที่เปลี่ยนแปลงสองมิติ คือ (0...10,2...12) แต่การประกาศทำอยู่ในช่วงหนึ่งเท่านั้น มิได้ประกาศทั้งหมด เช่น (1..6,3..8) และสำหรับการสมัครสมาชิกทำอยู่ในช่วง (4..9,6..11) จะเห็นว่ามีส่วนที่ซ้อนทับกันระหว่างพื้นที่ทั้งสองคือ (4..6,6..8) หรือจะมองว่าคือส่วนที่ อินเตอร์เซกชันเท่านั้นที่จะถูกส่งผ่านโดย RTI



รูปที่ 2.13 แสดงการประกาศและสมัครสมาชิกเชิงค่า

ตัวอย่างการส่งผ่านข้อมูลเชิงค่าแสดงในรูปที่ 2.14 โดยการประกาศและสมัครสมาชิกเป็นดังที่กำหนดไว้ก่อนหน้านี้ ซึ่งจะอธิบายเรียงตามลำดับเหตุการณ์ได้ดังนี้

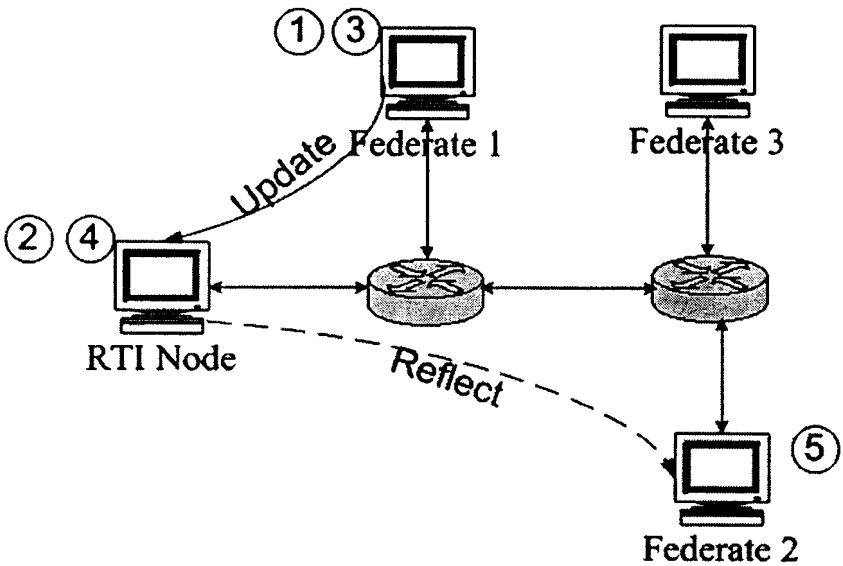
เหตุการณ์ที่ 1 อ็อบเจกต์ของคลาส X ปรับปรุงข้อมูลออกมาโดยมีค่าเป็น (1,3)

เหตุการณ์ที่ 2 RTI รับข้อมูลแต่ไม่มีการสะท้อนค่าข้อมูลออกเนื่องจากอยู่นอกเขตสนใจหรือ ส่วนที่เกิดจากการอินเตอร์เซกชัน

เหตุการณ์ที่ 3 อ็อบเจกต์ของคลาส X ปรับปรุงข้อมูลออกมาโดยมีค่าเป็น (5,7)

เหตุการณ์ที่ 4 RTI รับข้อมูลและสะท้อนค่าข้อมูลออกสู่ Federate 2 เนื่องจาก (5,7) อยู่ในส่วนที่อินเตอร์เซกชัน

เหตุการณ์ที่ 5 Federate 2 รับข้อมูล (5,7)



รูปที่ 2.14 แสดงการส่งผ่านข้อมูลเชิงค่า

2.4 องค์ประกอบภายใน RTI

การส่งผ่านข้อมูลที่เกิดขึ้นภายใน Federation ทั้งที่อยู่ในส่วน Object Management หรือ Data Distribution Management ล้วนต้องอาศัยการทำงานของ RTI ทั้งสิ้น สำหรับ Object Management กระบวนการทำงานของ RTI คือ การตรวจสอบผู้ที่ทำการสมัครสมาชิก และเมื่อได้รับข้อมูลจากกระบวนการปรับปรุงข้อมูล ก็จะทำการสะท้อนข้อมูลออกสู่ผู้ที่สมัครสมาชิกแล้วเท่านั้น แต่สำหรับ Data Distribution Management RTI จะต้องทำการกรองข้อมูลตามช่วงข้อมูลที่ได้ทำการสมัครสมาชิกก่อน เพื่อทำการสะท้อนข้อมูลออกได้ถูกต้อง

ในตอนนี้จะได้แสดงกระบวนการทำงานภายใน RTI ดังรูปที่ 2.15 เป็นแบบจำลองกลไกภายใน RTI Node ซึ่งประกอบด้วยองค์ประกอบสำคัญคือ ตารางค้นหา (Lookup Table) ซึ่งจะเชื่อมโยงความสัมพันธ์ระหว่างอ็อบเจกต์ที่ถูกประกาศกับผู้ที่สมัครสมาชิกแล้ว โดยในการออกแบบและเขียนโปรแกรมของตารางค้นหานี้ ได้ถูกออกแบบให้ทำงานเป็นคลาสอิสระ ซึ่งทำให้สามารถประยุกต์คลาสตารางค้นหานี้ไปทำงานได้อย่างยืดหยุ่น สำหรับเรื่องคลาสสำหรับตารางค้นหาได้อธิบายไว้ในบทที่ 4 ที่ว่าด้วยการออกแบบและสร้างโมดูล

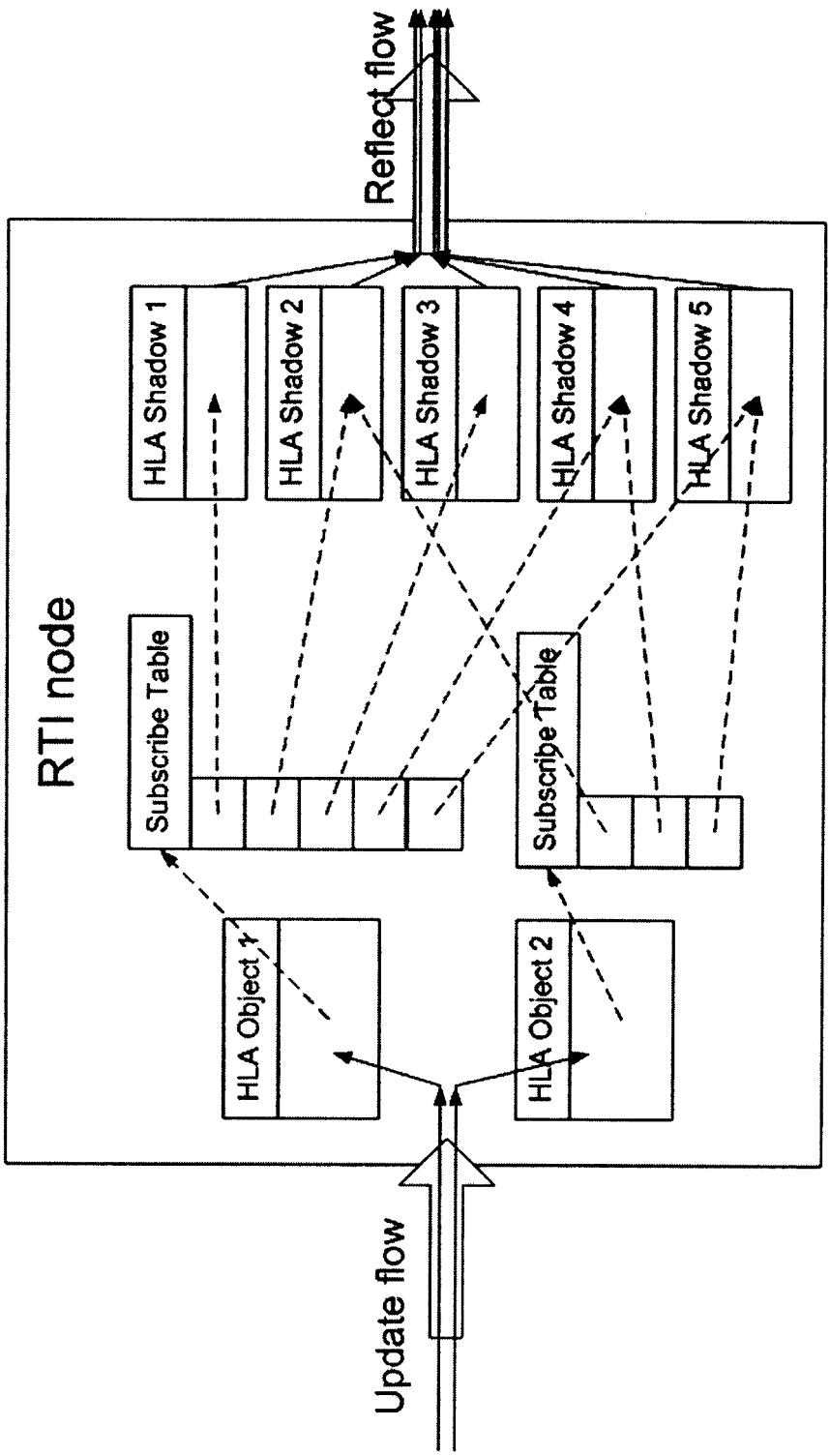
ดังแสดงในรูป HLA Object 1 เมื่อถูกประกาศและถูกสมัครสมาชิกโดย Federate 5 ตัว (ดังแสดงในภาพทางขวามือ) และ HLA Shadow แสดงความความหมายว่า Federate ที่ได้ทำการสมัคร

สมาชิกแล้วจะสามารถถูกมองเห็นได้ ในอีกด้านหนึ่งอาจพิจารณาได้ว่า HLA Shadow เป็นตัวแทนของ Federate ที่อยู่ที่ RTI เมื่อรับข้อมูลการสะท้อนจาก RTI แล้ว ก็จะรับหน้าที่นำพาข้อมูลนั้นสู่ Federate ต่อไป

หากพิจารณา HLA Object 2 เมื่อถูกประกาศ และถูกสมัครสมาชิกโดย HLA Shadow 2, 4 และ 5 แล้ว ดังนั้น เมื่อ HLA Object 2 มีการปรับปรุงข้อมูล RTI ก็จะทำการสะท้อนโดยทำข้อมูลสู่ HLA Shadow 2, 4 และ 5 เท่านั้น

ตามโครงสร้างในรูป 2.15 นั้น อาจจะสังเกตเห็นได้ว่าไม่มีส่วนใดเลยที่ทำหน้าที่ในการกรองข้อมูล แต่ก็ไม่ได้หมายความว่า โครงสร้างนี้ไม่สนับสนุนการทำงานของ Data Distribution Management ในบทต่อไปเรื่องการออกแบบและสร้างโมดูล จะกล่าวถึงเหตุผลและแนวคิดที่บ่งบอกว่าทำไมโครงสร้างของ RTI ในรูป 2.15 จึงสามารถตอบสนองได้ทั้ง Object Management และ Data Distribution Management

นอกจากนี้ RTI ยังมีภาระที่สำคัญอีกอย่างหนึ่งคือ การเป็นผู้ประสานงานด้านเวลา ซึ่งจะถูกกล่าวถึงรวมอยู่ในหัวข้อการจัดการเวลา



รูปที่ 2.15 แสดงลำดับการทำงานภายในของ RTI

2.5 การจัดการเวลา (Time Management) ใน HLA

ตามข้อกำหนดของ HLA มิได้กำหนดขั้นตอนวิธีสำหรับจัดการเวลาที่ระบบจะต้องใช้สำหรับการเข้าจังหวะระหว่าง Federate (IEEE 1516-2000, 2000) ทำให้ผู้ผลิตซอฟต์แวร์ RTI มีอิสระที่จะเลือกใช้นโยบายการจัดการเวลาแบบใดๆ ก็ได้ แต่ทั้งนี้นโยบายการจัดการเวลาที่แตกต่างกัน จะทำให้ระบบแตกต่างกันด้วย ในวิทยานิพนธ์ฉบับนี้ได้เลือกการจำลองการทำงานของ RTI โดยใช้ขั้นตอนวิธีแบบอนุรักษ์นิยม (Conservative) โดยใช้การคำนวณ Lower Bound on the Time Stamp (LBTS) (Fujimoto, 2000)

ในที่นี้องค์ประกอบของการจัดการเวลาประกอบด้วย 2 องค์ประกอบหลัก ดังนี้ ข้อความที่ใช้ร้องขอและตอบรับการขอเคลื่อนเวลา และ การคำนวณ LBTS โดยจะแยกอธิบายในรายละเอียดในหัวข้อย่อย 2.5.2 และ 2.5.3 แต่ใน หัวข้อย่อย 2.5.1 จะแสดงหลักการเกี่ยวกับเหตุการณ์ก่อนเพราะจะเป็นพื้นฐานสำหรับหัวข้อต่อไป

2.5.1 เหตุการณ์ (Event/Interaction)

ใน HLA ได้แบ่งแยกข้อความหรือข้อมูลที่ส่งระว่าง Federate ออกเป็น 2 ชนิด คือ

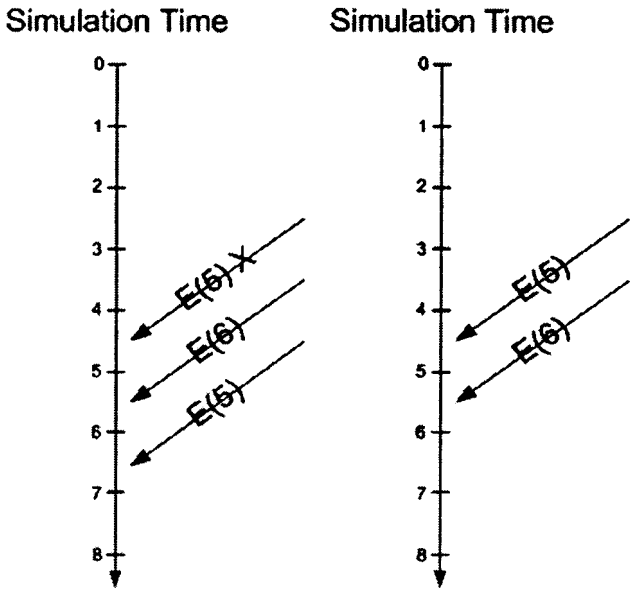
1. ข้อความชนิดที่ไม่มีเวลากำกับ หมายถึง ภายในข้อความหนึ่งหน่วย มีเพียงข้อมูลของคุณสมบัติของอ็อบเจกต์ที่ถูกปรับปรุงเท่านั้น ดังได้ถูกกล่าวถึงแล้วในตอนต้น
2. ข้อความที่มีเวลากำกับ หมายถึง ภายในข้อความหนึ่งหน่วย มีทั้งข้อมูลและเวลากำกับไปด้วย หรือ เรียกข้อความที่มีเวลากำกับไปด้วยนี้ว่า เหตุการณ์

เหตุการณ์จะถูกพรรณนาออกมาในรูปของคลาส เช่นเดียวกับอ็อบเจกต์ทั่วไป เช่น หากมีคลาสของเครื่องบินซึ่งอธิบายถึงคุณสมบัติของเครื่องบิน ในทำนองเดียวกันอาจมีคลาสของการยิงปืนซึ่งก็จะอธิบายลักษณะของการยิงปืนก็ได้ สำหรับคลาสของเหตุการณ์สามารถถูกกระทำด้วยกระบวนการต่างๆ เช่น การประกาศหรือการสมัครสมาชิกก็สามารถทำได้ในทำนองเดียวกัน

โดยหลักใหญ่แล้วเหตุการณ์และข้อมูลธรรมดาจะแตกต่างกันตรงที่ การรับและการแปลความหมาย กล่าวคือ เหตุการณ์นั้นต้องถูกรับและแปลความหมายโดย Federate ปลายทาง (ผู้สมัครสมาชิก) ภายในเวลาที่กำหนด เช่น

กำหนดให้ E(5) คือ เหตุการณ์ที่มีเวลากำกับเท่ากับ 5 จะต้องถูกแปลความหมายและมีปฏิกริยาต่อเหตุการณ์นั้นโดย Federate ปลายทาง หลังเวลาการจำลอง (Simulation Time) ที่ 4 และ

ก่อนเวลาการจำลอง 6 จะมาถึง มิฉะนั้นอาจเกิดความผิดพลาดได้ เช่น หากรับ E(5) ขณะที่เวลาการจำลองกำลังจะก้าวเข้าสู่เวลาที่ 7 ก็เปรียบเสมือนเครื่องจำลองนั้นมองเห็นเหตุเหตุการณ์อดีต ดังรูป 2.16แสดงความผิดปรกติที่อาจเกิดขึ้นได้ โดยทางซ้ายมือแสดงความผิดปรกติคือ E(5) เดินทางมาถึงล่าช้ากว่าที่ควร คือมาถึงหลัง E(6) ซึ่งเป็นเหตุการณ์ที่ผิดลำดับ ซึ่งตามที่ควรจะเป็นคือรูปขามือ E(5) ควรจะมาถึงและถูกแปลความหมายก่อน E(6) เสมอ



รูปที่ 2.16 แสดงความผิดปรกติอันเกิดจาก เหตุการณ์เดินทางมาถึงล่าช้า

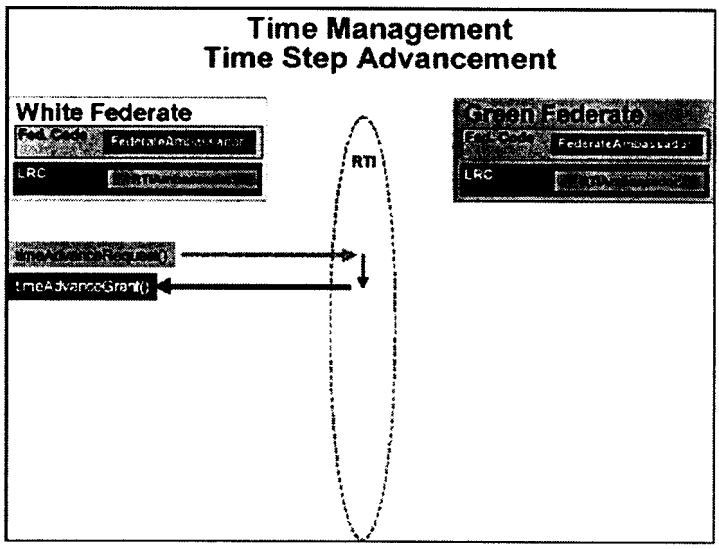
เพื่อป้องกันการเกิดเหตุการณ์ดังรูป 2.16 จึงเกิดการจัดการเวลาแบบอนุรักษ์นิยมขึ้น ซึ่งจะป้องกันการผิดลำดับของเหตุการณ์ ในตอนต่อไปจะได้อธิบายถึงข้อความที่ใช้ร้องขอและตอบรับการขอเคลื่อนเวลา และต่อด้วยหลักการคำนวณสำคัญ คือ การคำนวณ Lower Bound on the Time Stamp (LBTS) ซึ่งเป็นพื้นฐานของกระบวนการจัดการเวลาแบบอนุรักษ์นิยม

2.5.2 ข้อความที่ใช้ร้องขอและตอบรับการขอเคลื่อนเวลา

HLA มิได้กำหนดขั้นตอนวิธีสำหรับการจัดการเวลา แต่ได้กำหนดให้ใช้ข้อความในการร้องขอและตอบรับการขอเคลื่อนเวลา โดยมีรายละเอียดดังนี้ (Carothers, Fujimoto, Weatherly and Wilson, 1997)

- ข้อความร้องขอการเคลื่อนเวลา ประกอบด้วย

1. ข้อความ Time Advance Request (TAR) คือ ข้อความที่ Federate ต้องส่งให้ RTI เพื่อร้องขอให้ RTI พิจารณาการเคลื่อนเวลา โดยจะต้องมีพารามิเตอร์ของเวลาที่ต้องการร้องขอกำกับไปด้วย โดยมีลักษณะ คือ TAR (t)โดย t คือ เวลาที่ร้องขอ ผลตอบสนองจาก RTI คือการคำนวณ LBTS เพื่อพิจารณาการเคลื่อนเวลาและจะส่งเหตุการณ์ที่มีเวลากำกับทั้งหมดที่มีค่าเวลากำกับน้อยกว่าหรือเท่ากับเวลาร้องขอ t มาให้ทั้งหมด โดยในรูป 2.17 แสดงลักษณะทางตรรกะของการร้องขอด้วย TAR

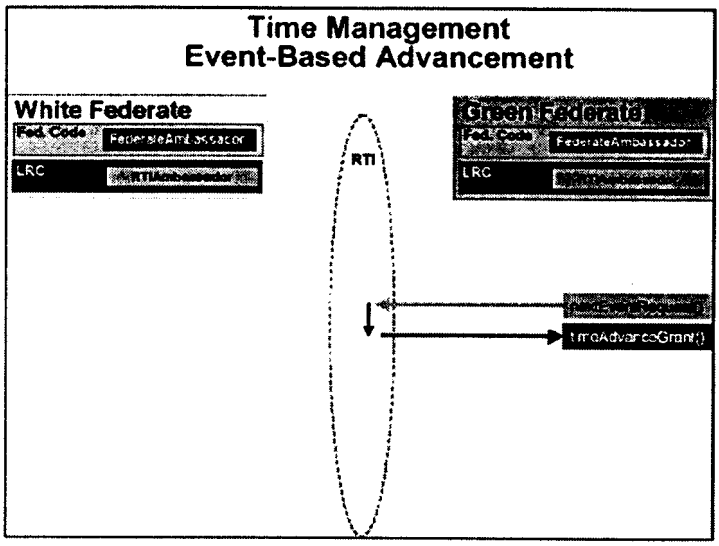


รูปที่ 2.17 แสดงการจัดการเวลาด้วย TAR

(ที่มา : HLA-RTI1.3-NG Programmer's Guide Version 5, 2002)

2. ข้อความ Next Event Request (NER) คือ ข้อความที่ Federate ต้องส่งให้ RTI เพื่อร้องขอให้ RTI พิจารณาการเคลื่อนเวลาไปยังเหตุการณ์ถัดไป โดยจะมีลักษณะคือ NER(t, one or all) สำหรับพารามิเตอร์ t คือเวลา พารามิเตอร์ที่สองเพื่อบอก RTI ว่าต้องการให้ส่งผ่านเหตุการณ์เพียงหนึ่งหรือทั้งหมด สมมติว่า Federate ส่ง NER(t, one) ให้กับ RTI ผลตอบสนองคือ RTI จะส่งเหตุการณ์ที่มีค่าเวลากำกับน้อยที่สุดและน้อยกว่าหรือเท่ากับ t มาให้ โดยในรูป 2.18 แสดงลักษณะทางตรรกะอย่างง่ายของการร้องขอด้วย NER

- ข้อความตอบรับการขอเคลื่อนเวลา หรือ Time Advance Grant (TAG) คือ ข้อความที่ RTI ส่งให้กับ Federate เมื่อต้องการอนุญาตให้เคลื่อนเวลาได้ โดยมีลักษณะเป็น TAG (t) โดย t คือ เวลาที่อนุญาตให้เคลื่อนไป



รูปที่ 2.18 แสดงการจัดการเวลาด้วย NER

(ที่มา : HLA-RTI1.3-NG Programmer's Guide Version 5, 2002)

สำหรับขั้นตอนวิธีติดต่อระหว่าง Federate และ RTI เพื่อจัดการเวลาจะได้ยกตัวอย่างให้เห็นอย่างเป็นรูปธรรมต่อไป

ก่อนจะทำการยกตัวอย่างขึ้นตอนการติดต่อเพื่อร้องขอการเคลื่อนเวลา จะขออธิบายความหมายของ Federate 2 ชนิด ซึ่งจะต้องอ้างอิงถึงในส่วนต่อไป ดังต่อไปนี้

1. Regulating Federate หมายถึง Federate ผู้ทำการประกาศคลาสของอ็อบเจกต์

2. Constrained Federate หมายถึง Federate ผู้ทำการสมัครสมาชิกคลาสของอ็อบเจกต์

จากรูปที่ 2.19 ได้แสดงถึงขึ้นตอนการติดต่อระหว่าง Constrained Federate, Regulating Federate และ RTI โดยข้อกำหนดเบื้องต้นคือ Constrained Federate จะทำการสมัครสมาชิกต่อคลาสของเหตุการณ์ E ซึ่งเป็นของ Regulating Federate และมีเหตุการณ์ที่จะเกิดขึ้นต่อเนื่องมาอีกดังอธิบายต่อไปนี้

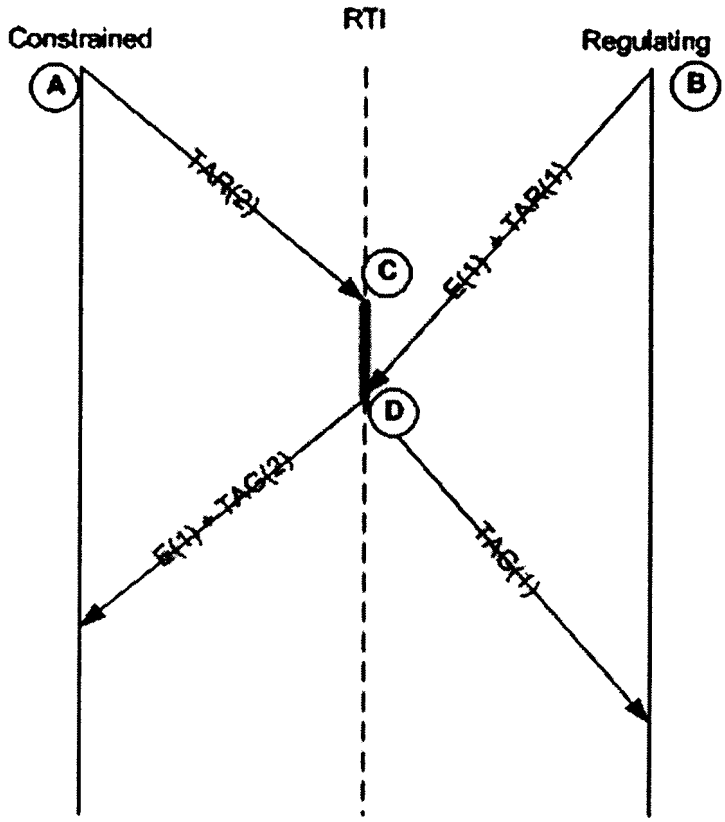
เหตุการณ์ A แสดง Constrained Federate ส่ง TAR(2) นั่นคือ ขอเคลื่อนเวลาไปยังเวลาที่ 2

เหตุการณ์ B แสดง Regulating Federate ได้ส่ง E(1) + TAR(1) ซึ่งหมายถึง Regulating Federate ได้ส่งเหตุการณ์ซึ่งมีเวลากำกับเท่ากับ 1 ออกมาและร้องขอการเคลื่อนเวลาไปยัง 1

เหตุการณ์ C แสดงการรับ TAR จาก Constrained Federate แต่ RTI จะยังไม่อนุญาตการเคลื่อนเวลาให้แก่ Constrained Federate ในทันที อันเป็นผลจากการคำนวณ LBTS ซึ่งจะทำให้ RTI หน่วงเวลาการส่ง TAG ออกไปชั่วคราวดังแสดงด้วยเส้นทึบ

เหตุการณ์ D แสดงการมาถึงของ E(1) และ TAR(1) จาก Regulating Federate ซึ่งจะกระตุ้น RTI ให้คำนวณ LBTS แต่ผลการคำนวณครั้งนี้ RTI ได้อนุญาตให้ทั้ง Constrained Federate และ Regulating Federate เคลื่อนเวลาได้

รายละเอียดเรื่องการคำนวณ LBTS จะแสดงในหัวข้อถัดไป



รูปที่ 2.19 แสดงตัวอย่างการจัดการเวลาใน HLA กรณีร้องขอการเคลื่อนเวลาด้วย TAR

2.5.3 การคำนวณ LBTS

ในทางปฏิบัติการคำนวณ LBTS จะถูกกระทำที่ RTI โดยจะคำนวณเมื่อได้รับ TAR(t) ซึ่งจะใช้การคำนวณดังปรากฏในสมการที่ (1) ดังนี้

$$LBTS = \min (T_{REG} + L_{REG}) \quad \ldots\ldots\ldots (1)$$

T_{REG} คือ เวลาของ Regulating Federate ที่ร้องขอ

L_{REG} คือ ค่า Lookahead ของ Regulating Federate

โดยถ้า t น้อยกว่าหรือเท่ากับ LBTS จะได้รับพิจารณาให้เคลื่อนเวลาได้

เพื่อให้เห็นรูปธรรมของการคำนวณ LBTS และ ขั้นตอนการร้องขอการเคลื่อนเวลาได้อย่างชัดเจนขึ้น จึงได้ยกตัวอย่างการร้องขอโดยการใช้ TAR และ NER

สำหรับตัวอย่างของการขอการเคลื่อนเวลาโดย TAR จะใช้รูป 2.19 เป็นตัวอย่าง โดยสถานะเริ่มต้นกำหนดให้เวลาเท่ากับ 0 ค่า Lookahead เท่ากับ 1 และ Regulating Federate เป็นผู้ประกาศคลาสของเหตุการณ์ E และ Constrained Federate เป็นผู้สมัครสมาชิกต่อคลาสของเหตุการณ์ E โดยจะอธิบายเป็นลำดับเหตุการณ์ได้ดังนี้

เหตุการณ์ A แสดง Constrained Federate ส่ง TAR(2) นั่นคือ ขอเคลื่อนเวลาไปยังเวลาที่ 2

เหตุการณ์ B แสดง Regulating Federate ได้ส่ง E(1) + TAR(1) ซึ่งหมายถึง Regulating Federate ได้ส่งเหตุการณ์ซึ่งมีเวลากำกับเท่ากับ 1 ออกมาและร้องขอการเคลื่อนเวลาไปยัง 1

เหตุการณ์ C แสดง RTI รับ TAR(2) จาก Constrained Federate แต่ RTI จะยังไม่อนุญาตการเคลื่อนเวลาให้แก่ Constrained Federate ในทันทีอันเป็นผลจากการคำนวณ LBTS ซึ่งได้ผลลัพธ์คือ min(0+1) = 1 จะเห็นว่า 2 (พารามิเตอร์เวลาของ TAR) มากกว่า LBTS ซึ่งจะทำให้ RTI หน่วงเวลาการส่ง TAG ออกไปชั่วคราวดังแสดงด้วยเส้นทึบ

เหตุการณ์ D แสดงการมาถึงของ E(1) และ TAR(1) จาก Regulating Federate ซึ่งจะกระตุ้น RTI ให้คำนวณ LBTS อีกครั้ง แต่ผลการคำนวณครั้งนี้ RTI ได้อนุญาตให้ทั้ง Constrained Federate เคลื่อนเวลาได้เพราะ 2 (พารามิเตอร์เวลาของ TAR) เท่ากับ min(1+1) = 2 และสำหรับ Regulating Federate เคลื่อนเวลาได้ ข้อสังเกตคือ Regulating Federate อาจเป็น Constrained Federate ด้วยก็ได้ในเวลาเดียวกัน แต่ขณะนี้ Regulating Federate ไม่ได้เป็น Constrained Federate ทำให้ RTI ไม่ต้องคำนวณ LBTS ทำให้สามารถอนุญาตการเคลื่อนที่เวลาได้ทันที

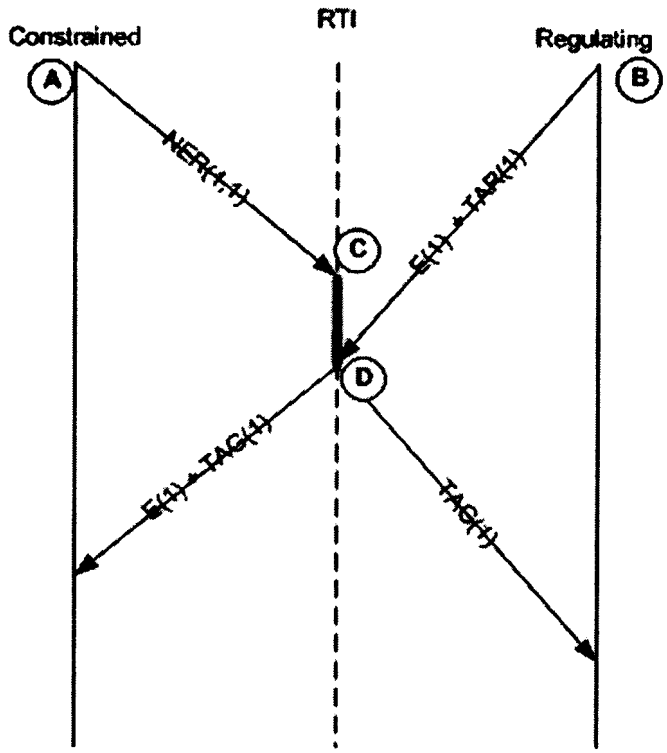
ต่อไปจะเป็นตัวอย่างของ NER กำหนดให้ทุก Federate เริ่มต้นที่เวลาเท่ากับ 0 และ Regulating Federate เป็นผู้ประกาศคลาสของเหตุการณ์ E และ Constrained Federate เป็นผู้สมัครสมาชิกต่อคลาสของเหตุการณ์ E โดยจะอธิบายเป็นลำดับเหตุการณ์ได้ดังนี้

เหตุการณ์ A แสดง Constrained Federate ส่ง NER(1,1) นั่นคือ ขอเคลื่อนเวลาไปยังที่เวลา 1 และขอเหตุการณ์ถัดไปหนึ่งเหตุการณ์

เหตุการณ์ B แสดง Regulating Federate ได้ส่ง E(1) + TAR(1) ซึ่งหมายถึง Regulating Federate ได้ส่งเหตุการณ์ซึ่งมีเวลากำกับเท่ากับ 1 ออกมาและร้องขอการเคลื่อนเวลาไปยัง 1

เหตุการณ์ C แสดงการรับ NER จาก Constrained Federate แต่ RTI จะยังไม่อนุญาตการเคลื่อนเวลาให้แก่ Constrained Federate ในทันที อันเป็นผลจากการคำนวณ LBTS ซึ่งจะทำให้ RTI หน่วงเวลาการส่ง TAG ออกไปชั่วคราว

เหตุการณ์ D แสดงการมาถึงของ E(1) และ TAR(1) จาก Regulating Federate ซึ่งจะกระตุ้น RTI ให้คำนวณ LBTS แต่ผลการคำนวณครั้งนี้ RTI ได้อนุญาตให้ Constrained Federate เคลื่อนเวลาสู่ 1 ด้วยการส่ง TAG(1) เนื่องจาก LBTS = min (T_{REG} + L_{REG}) = 1 + 1 = 2 และได้อนุญาตให้ Regulating Federate เคลื่อนเวลาสู่ 1 ด้วย TAG(1)

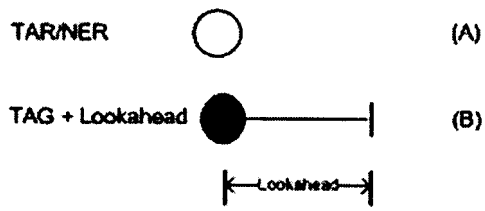


รูปที่ 2.20 แสดงตัวอย่างการจัดการเวลาใน HLA กรณีร้องขอการเคลื่อนเวลาด้วย NER

2.5.4 วิธีการอ่านกราฟการเคลื่อนที่ของเวลา

กราฟการเคลื่อนที่ของเวลาจะใช้เพื่อแสดงลักษณะการเคลื่อนที่ของเวลาอย่างเป็นรูปธรรมของ Federate ต่างๆ ภายใน Federation ซึ่งจะช่วยให้เกิดความเข้าใจต่อลักษณะของการเคลื่อนที่ของเวลาได้ดียิ่งขึ้น โดยได้นิยามสัญลักษณ์ไว้ในรูป 2.21 รูปย่อย A วงกลมขาว แสดงถึงการที่ Federate ได้ส่งข้อความร้องขอการเคลื่อนเวลาออกมาซึ่งอาจเป็นได้ทั้ง TAR หรือ NER รูปย่อย B แสดงการที่

RTI ได้ส่ง TAG ออกเพื่อให้ Federate เคลื่อนเวลาได้ สำหรับก้านที่ยื่นออกมาทางขวาแสดง Lookahead ของ Federate นั้น



รูปที่ 2.21 แสดงสัญลักษณ์ที่ใช้ในกราฟการเคลื่อนที่ของเวลา

ต่อไปจะแสดงตัวอย่างการอ่านกราฟ พร้อมแสดงเหตุผลการคำนวณประกอบตั้งแต่เหตุการณ์ A ถึง G ดังรูปที่ 2.22

เหตุการณ์ A ทั้ง Constrained Federate และ Regulating Federate อยู่ในสถานะตั้งต้นและได้ทำการส่ง TAR หรือ NER ออกมา

เหตุการณ์ B ทั้ง 2 ได้รับ TAG(0) นั่นคือการอนุญาตให้เริ่มต้นทำการคำนวณได้

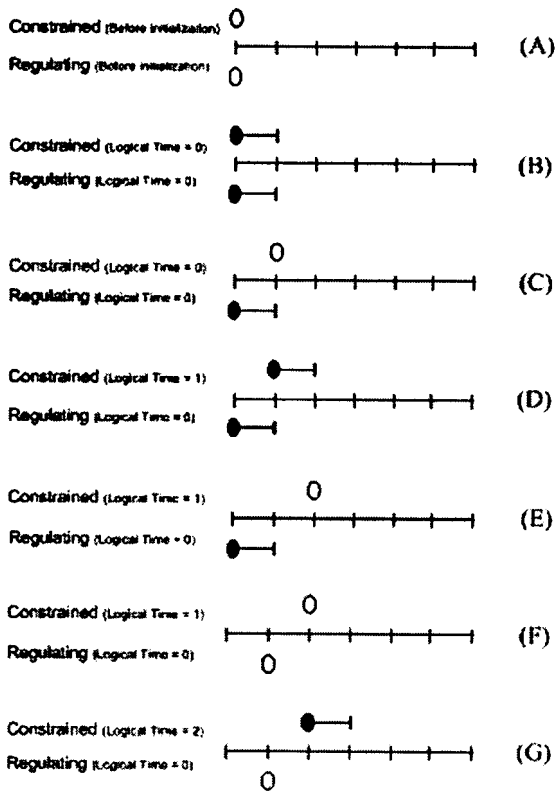
เหตุการณ์ C Constrained Federate ส่ง TAR(1) หรือ NER ออกสู่ RTI

เหตุการณ์ D RTI ส่ง TAG(1) ออกสู่ Constrained Federate ได้เนื่องจาก TAR(1) เท่ากับ LBTS = ($T_{REG} + L_{REG}$) = min (0 + 1) = 1

เหตุการณ์ E Constrained Federate ส่ง TAR(2) หรือ NER ออกสู่ RTI แต่ไม่ได้รับอนุญาตในทันทีเพราะ TAR(2) มากกว่า LBTS = ($T_{REG} + L_{REG}$) = min (0 + 1) = 1

เหตุการณ์ F Regulating Federate ส่ง TAR(1) ออกสู่ RTI

เหตุการณ์ G RTI ส่ง TAG(2) สู่ Constrained Federate เพราะ TAR(2) เท่ากับ LBTS = ($T_{REG} + L_{REG}$) = min (1 + 1) = 2



รูปที่ 2.22 แสดงกราฟการเคลื่อนที่ของเวลา

2.6 สรุป

ในบทนี้ได้กล่าวถึงทฤษฎีและหลักการพื้นฐานสำหรับการจำลองแบบกระจายที่ใช้ HLA ทั้งนี้มุ่งเน้นให้ครอบคลุมทั้ง 3 บริการหลักคือ Object Management, Data Distribution Management และ Time Management และมีความจำเป็นอย่างยิ่งสำหรับการวิจัยเพื่อสร้างระบบการจำลองทุกชนิดที่จะต้องนิยามระบบการจำลองให้ชัดเจน ดังนั้นโดยเนื้อหาได้พยายามผูกโยงความสัมพันธ์ระหว่าง HLA ในระบบการจำลองจริงกับระบบสำหรับการจำลอง HLA ซึ่งจะต้องถูกออกแบบและสร้างขึ้นเพื่อแทนระบบจริงให้ได้