UNIVERSITY OF CALIFORNIA,
IRVINE

An Adaptive, Distributed Algorithm for Interest Management

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Katherine Lee Morse

Dissertation Committee:
Professor Lubomir Bic, Co-Chair
Professor Michael Dillencourt, Co-Chair
Professor Michael Zyda

2000

The dissertation of Katherine Lee Morse is approved
and is acceptable in quality and form
for publication on microfilm:

_____

_____

_____

Committee Chair

University of California, Irvine
2000

DEDICATION

To

My mother, Marjorie, who taught me I could be anything I dreamed.

And to David, who ran with me every step of the way.

Table of Contents

CHAPTER 1: Related Work 5
1.1 Large Scale Virtual Environments 5
1.1.1 Scope 7
1.1.2 A Brief History 8
1.2 The Need for Interest Management 10
1.2.1 Interest Expressions 11
1.3 Taxonomy 12
1.3.1 Interest Expression Specification 12
1.3.2 Architecture 18
1.4 Implementations 21
1.4.1 ModSAF 21
1.4.2 JPSD 23
1.4.3 CCTT 24
1.4.4 NPSNET 25
1.4.5 STOW-E 26
1.4.6 STOW ED-1 27
1.4.7 ALSP 29
1.4.8 HLA 30
1.4.9 STOW ACTD 31
1.4.10 Proximity Detection 32
1.4.11 MASSIVE-2 32
1.4.12 DIVE 33
1.4.13 MAVERIK 34
1.5 Classification 35
1.5.1 Implications for Multicast Research 41
CHAPTER 2: Problem Statement 44
2.1 Overview Of Data Distribution Management In The HLA 44
2.2 Multicast Grouping 55
2.2.1 Connection Graphs 56
2.2.2 Total Possible Combinations 57
2.2.3 Formal Problem Statement 59
2.2.4 Cost Function 64
2.3 Estimating tq and tds 68
CHAPTER 3: Solution Evaluation 73
3.1 Performance Measures 73
3.2 Characterizing Scenarios 75
3.3 Benchmark Algorithm 76
3.3.1 Inputs to the Benchmark Algorithm 80
CHAPTER 4: Implemented Solutions and Contributions 84
4.1 MESSENGERS Overview 84
4.2 Baseline Prototype 85
4.3 Offline Grouping Algorithms 90
4.3.1 The Global Largest Outgoing Connection Algorithm 90

4.3.2 The Input-Restricted LOC Algorithm 99
4.4 Online Distributed IRLOC Algorithm 115
4.4.1 Testing the Online Grouping Algorithm 119
4.5 Research Contributions 125
CHAPTER 5: Future Work 128
5.1 Load Balancing 128
5.2 Varying ts and tr 130
5.3 Measuring Weights 130
5.4 Maintaining Incoming Weight Information 131
5.5 Ungrouping 131
5.6 Accounting for and Measuring tp 132
5.7 The Real World 132
APPENDIX A: Offline Simulator Test Inputs and Results 141
APPENDIX B: Large Connection Set Figures 199

List of Figures

Virtual Environment Representation in a Network Environment 7
Area of Interest Example 15
Area of Interest with Cells 15
Area of Interest with Extents 16
Source-Based Filtering 19
Destination-Based Filtering 20
Intermediate Filtering 20
HLA Federation Logical View 45
Two-Dimensional Routing Space Example 49
Geographic Routing Space Example 50
Multi-Extent Regions 54
Example Region Layout 57
Example Connection Graph 58
Connection Graph Exceeding Maximum Weight 62
Connection Graph Demonstrating Lack of Self-Reducability 64
Illustration of Time Bound for k Point-to-point Communications 66
Illustration of Time Bound for k Multicast Communications 67
Connection Graph for Estimating t_q and t_{ds} 69
Initial Region Layout 77
Accounting for Local Subscription Regions 78
Sample Region Layout from Benchmark Algorithm 80
Resulting Connection Graph 81
Federate Join and Federation Creation 87
Routing Space Initialization 88
Subscription Object Class Attributes With Region 89
Register Object Instance With Region 90
Discover Object and Establish Connectivity 91
Sparse Uniform Connectivity, Variable Weights, 1 Connection/Node 94
Non-Uniform Sparse Connectivity, Uniform Low Weights, Multiple Connections/Node 95
Fully Connected Subgraphs, Uniform Low Weights, Multiple Connections/Node 96
5 Connections, Point to Point 103
5 Connections, Broadcast 103
6 Connections, Point to Point 104
6 Connections, Broadcast 104
5 Connections, 2 Groups, LOC 105
5 Connections, 3 Groups, LOC 106
6 Connections, 2 Groups, LOC 106
5 Connections, 2 Groups, IRLOC 107
5 Connections, 3 Groups, IRLOC 107
6 Connections, 2 Groups, IRLOC 108
Static vs. Dynamic Allocation of Multicast Groups 114
Attempt to Add a Connection to a Group 116
Report Success or Failure of Grouping 118

Dropping Connections and Resigning from Groups 119
5 Connections, 2 Groups, Online Grouping 121
5 Connections, 3 Groups, Online Grouping 121
6 Connections, 2 Groups, Online Grouping 122
Physical and Logical Relationship of Research Components to DDM in the HLA 127
Pre-Engagement Snapshot 199
Engagement Snapshot 200
Post Engagement Snapshot 201

List of Tables

Implementation Classification	37
Specificity vs. Query Domain Selection Time	39
Multicast vs. Software Infrastructure Support	40
Average t_q for Example Figure 2-6 and Figure 2-11	69
Average t_{ds} for Figure 2-6 and Figure 2-11	70
Average Message Delivery Time for Figure 2-6	70
Average Message Delivery Time for Figure 2-11	71
Random Connection Set Tests	97
Entity Connection Weight by Type and Counts by Federate	110
Region Ranges	110
Class Subscriptions	111
RTI Timing Experiments	123
Initialization Times	123
Load-Balancing Experiment	129

List of Equations

Stirling Number of the Second Kind	58
Total Possible Combinations	58
Message Delivery Bound	60
t_{ds} for c Not in a Group	60
t_{ds} for c in a Group	60
t_q	61
Time Bound for k Point-to-point Communications	65
Time Bound for k Multicast Communications	66
Average Serial Send Delay Time	67
Multicast Improvement in t_{ds}	68
Average t_q with Grouping	68
Improving Delivery of c_1 to f_2	71

ACKNOWLEDGMENTS

My thanks to Steve Seidensticker who opened all the doors; Dr. Judith Dahmann for the compelling mental challenges, the inspiration of her example, and the money; Dr. Jack Thorpe for the advice to be where I wanted to be; SAIC management for letting me have enough rope to hang myself; Dr. Sandra Hutchins for her respect and example; susan@espressonet.com for opening espressoNet where virtually every word of this was written; Dr. John King for teaching me the rules of engagement; the whole "HLA gang" who honed my thinking on this subject, especially Dr. Richard Weatherly, Jim Calvin, Reed Little, James Ivers, Andreas Kemkes, Dr. Mikel Petty, James Ivers, Jurgen Schulze, and Steve Bachinsky; Dr. Duke Hong for the simulator that proved the algorithms; Dr. Munehiro Fukuda, Eugene Gendelman, and Hairong Kuang for the various versions of Messengers that supported the distributed version; Joni Currier for writing the "RTI API" much more cleanly than I would have; Juergen Schulze for debugging the benchmark algorithm and making it cleaner in C++; my committee for their patience; Dave Walter for the half-time WMD job right at the end with just the right level of coverage and mental challenge without all the airplane trips; Mike Zyda for showing me the way to closure.

My eternal love and gratitude to David Lee Drake for everything else including, but certainly not limited to, financial support, moral support, listening, understanding, asking the right questions, writing code, and reformatting the whole damn thing in FrameMaker.

To all of you who believed when I lost faith:

- David Drake
- John Chalfant
- Susan Gingrich
- Alan Evans
- Steve Manning
- Joni Currier
- Jack Thorpe
- Laurie Rogers-Webster
- Duke Hong
- Sharon Ellison
- Mike Zyda
- Jennifer Logan
- Richard Weatherly

It’s easy to make the simple opaque. The challenge is to make the complex transparent.

CURRICULUM VITAE

2000	Ph.D. in Information and Computer Science, University of California, Irvine, “An Adaptive, Distributed Algorithm for Interest Management,” Professor Lubomir Bic, Chair
1996	M.S. in Information & Computer Science, University of California, Irvine, emphasis in Computer Systems Design
1986	M.S. in Computer Science, University of Arizona
1983	B.A. in Russian, University of Arizona, Cum Laude, Phi Beta Kappa
1982	B.S. in Mathematics, University of Arizona, Cum Laude

PUBLICATIONS

Refereed Journal Publications

Katherine L. Morse, Lubomir Bic and Michael Dillencourt. Interest Mangement in Large Scale Virtual Environments. MIT PRESENCE, February 2000.

Refereed Conference Publications

Katherine L. Morse. Parallel Distributed Simulation in ModSim. Proceedings of the 10th Annual International Conference on Parallel Processing, 1990.

David L. Drake and Katherine L. Morse. The Security-Specific Eight Stage Risk Assessment Methodology. Proceedings of the 17th National Computer Security Conference, 1994.

Munehiro Fukuda, Katherine L. Morse, Lubomir Bic, Michael Dillencourt, D. Menzel and E. Lee. A Novel Approach to Toxicology Simulation Based on Autonomous Objects. Proceedings of the Conference on Simulation in the Medical Sciences, 1996.

David L. Drake and Katherine L. Morse. Applying the Eight Stage Risk Assessment Methodology to Firewalls. Proceedings of the 19th National Information Systems Security Conference, 1996.

Katherine L. Morse and Jeffrey Steinman. Data Distribution Management in the HLA: Multidimensional Regions and Physically Correct Filtering. Proceedings of the 1997 Spring Simulation Interoperability Workshop, 1997.

Katherine L. Morse, Dannie Cutts, John Hancock and Stephan Lubbers. Feasibility and Functionality of Autonomous Objects in the HLA. Proceedings of the 1997 Spring Simulation Interoperability Workshop, 1997.

Katherine L. Morse, Andreas Kemkes and Mikel Petty. Issues in the Relationship between HLA's Declaration Management and Data Distribution Management Services. Proceedings of the 1997 Fall Simulation Interoperability Workshop, 1997.

Katherine L. Morse. The Object Model Template Routing Space Table: Recording Federation-Global DDM Decisions. Proceedings of the 1999 Spring Simulation Interoperability Workshop, 1998.

Katherine L. Morse. The Object Model Template Routing Space Table: Recording Federation-Global DDM Decisions. Proceedings of the 1998 Spring Simulation Interoperability Workshop, 1998.

George J. Valentino, Todd Kniola, Shihab Khalil and Katherine L. Morse. An Agents Toolkit to Support Distributed Simulations. Proceedings of the 1998 Spring Simulation Interoperability Workshop, 1999.

Katherine L. Morse, Lubomir Bic, Michael Dillencourt and Kevin Tsai. Multicast Grouping for Dynamic Data Distribution Management. Proceedings of the 1999 Society for Computer Simulation Conference, 1999.

Invited Papers

Judith Dahmann and Katherine L. Morse. The High Level Architecture: An Update. Proceedings of the 1998 Workshop on Distributed Interactive Simulation – Real Time, 1998.

ABSTRACT OF THE DISSERTATION

An Adaptive, Distributed Algorithm for Interest Management

by

Katherine Lee Morse

Doctor of Philosophy in Information and Computer Science

University of California, Irvine, 2000

Professor Lubomir Bic, Chair
Professor Michael Dillencourt, Co-Chair

The scale of large-scale virtual environments (*LSVE*s) is limited by the ability of the supporting infrastructure to deliver data to participants in a timely manner. Multicast can improve data delivery time by minimizing message send time similarly to broadcast, while reducing the delivery of extraneous messages which goes with broadcast. However, multicast groups are typically limited resources, mostly due to hardware limitations. Significant performance improvements have been made using judicious, static assignments of multicast groups based on pre-defined criteria such as geographic location. However, such static approaches ultimately lack the flexibility to scale to meet the requirements of highly dynamic LSVEs. Dynamic multicast grouping has been considered to be too computationally expensive to be practically applicable. This dissertation derives a straightforward heuristic based on readily available data from which various computationally inexpensive algorithms can be derived. Through experiments with

simulations and a well-known LSVE environment, the feasibility of general application of these algorithms is demonstrated, as well as the significant reduction in the use of multicast groups they achieve. Finally, experimentation and analysis demonstrate that the real issue with dynamic assignment of multicast groups is the time required to reconfigure multicast hardware.

INTRODUCTION

Large-scale virtual environments (*LSVE*s) consist of thousands of complex entities moving and interacting in a three-dimensional, computer-maintained space. The physical environment is a collection of Local Area Networks (LANs) connected by a Wide Area Network (WAN), such as the Internet, spanning an area as large as a continent. Each LAN typically consists of several computers, each of which usually runs a single type of simulator. The simulator may simulate one or many entities. The total number of computers may number in the hundreds. Through this type of simulation, users and computers at geographically dispersed locations can interact with each other in a shared, simulated environment just as if they were in the same physical location. In addition, because the environment is simulated, activities can be performed which would be dangerous, expensive, or physically impossible in a real environment. For example, this approach may be used for interconnecting manned simulators which cannot be moved, or established LANs of workstations which would be expensive and impractical to move.

In [Capps99], Capps, Watsen and Zyda identify interest management as a key technology for improving the scalability of LSVEs. Interest management is a mechanism through which entities in an LSVE express, in a well defined way, their interest in receiving types or instances of data, or their ability to generate data. The interest management mechanism matches the two types of expressions and interacts with the data

delivery mechanism to ensure that the requested data is delivered to the appropriate receivers.

Most interest management systems to date have been purpose-built with relatively static architectures and static specification of filtering capabilities. The current trends for interest management systems are toward:

- Distributed and dynamic architectures

- Flexible, general purpose specification of filtering expressions

- Optimization to improve overhead of the interest management itself, especially through the use of multicast.

The research focuses on improving the performance of interest management through a distributed architecture and dynamic assignment of multicast groups. It assumes the filtering expression specification in the Data Distribution Management (DDM) services of the Department of Defense's (DoD) High Level Architecture [DMSO98].

Characteristics of scenarios which should impact the performance of interest management systems in general have been identified and quantified through analysis of simulation scenarios. A benchmark algorithm which exercises these characteristics has been built and used to test a distributed interest management system utilizing the

MESSENGERS mobile agents system. The interest management system operates both with and without multicast grouping. The success of the grouping is measured by demonstrating that data can be delivered more quickly using grouping while the additional overhead incurred is less than the amount of time it would take the participating simulations to discard the unwanted data which would have been received without the interest management system. The research contributions from this work are:

- Multicast grouping heuristic algorithms;

- Quantitative characterization system for scenarios with respect to DDM;

- Definition of the cost function for evaluating additions to multicast groups;

- Identification of the quantitative characterization of scenarios for which the grouping algorithm improves filtering efficiency without excessive overhead;

- User-controllable benchmark algorithm which exercises quantitative characterizations for performance analysis of DDM implementations.

The ultimate goal of this research is to build a flexible interest management system which users may tune based on general characteristics of their scenarios.

Chapter 1 describes related work. Chapter 2 provides the problem statement including a brief overview of the DDM services in the DoD's High Level Architecture [DMSO98] that are applicable to this research, a formal statement of the problem, and a description of the cost function for evaluating multicast grouping algorithms. Chapter 3 describes performance measures for evaluating the efficiency and effectiveness of any interest management system, a characterization of scenarios that directly addresses interest management, and a description of the benchmark algorithm which exercises the scenario characteristics. Chapter 4 describes implemented solutions and results, including the architecture of a baseline prototype built using MESSENGERS as a basis for comparison with a production RTI, two offline multicast grouping algorithms with simulated message delivery timing results, and the online distributed grouping algorithm with timing and performance results. Section 4.1 provides an overview of the MESSENGERS mobile agents system [Bic96, Bic95] used in this effort. Chapter 4 also summarizes the research contributions of this effort. Chapter 5 describes potential future work.

CHAPTER 1: Related Work

This section motivates the need for interest management, the scope of LSVE applications, and provides a brief history. Section 1.2 describes the basic mechanism of Interest Management including the interest expressions which entities use to invoke it. Section Taxonomy presents a taxonomy for classifying Interest Management systems. Section Implementations describes in detail thirteen systems which have used Interest Management. Section Classification classifies the thirteen systems surveyed according to the taxonomy described in section Taxonomy.

1.1 Large Scale Virtual Environments

The most common application of LSVEs to date has been for military war game training exercises. The technology has expanded directly from this origin into the entertainment field in the form of virtual reality games [Zyda97]. It also has potential applications to artificial life [Langston94], molecular dynamics [Hockney88], collision of space debris, and environmental simulation [Zeigler97]. [Logi99] describes a distributed, cooperative, real-time decision support system which allows traffic management experts and systems in different jurisdictions to share data and solve interconnected traffic congestion management problems. [Aguilera98, Banavar99] demonstrate a content-based subscription system for stock trading. The LSVE domain introduces two problems[1]. The first is that there is usually no correlation between an entity's physical location in the

1. For an overview of general LSVE requirements, see [Brutzman95]. For a review and taxonomy of LSVEs, see [Macedonia97].

network, i.e. where it's simulated, and its logical location in the virtual environment. And, of course, most entities are moving in the virtual environment, so any grouping relationships that may be found at one point in time won't necessarily hold for any length of time. Secondly, the fact that humans may be interacting within the virtual environment introduces the requirement that updates from entities in perceptible range must be reflected at intervals of approximately 100 ms, 80 ms of which is typically consumed by network transmission [Calvin95][2]. The communication model used may exacerbate these problems. If it were possible to migrate the simulators in the network to correspond to their proximity in the virtual environment, it would improve communication between close entities. However, such migration is usually precluded by the complexity and specialization of the models, and closeness in the virtual environment may be defined along many axes to the effect that there is no optimal migration. As a result it is necessary to improve communication through other techniques such as Interest Management.

Figure 1-1 illustrates the relationship between entities in the network environment and their location in the virtual environment. Notice that some entities simulated on the same node or LAN are far apart in the virtual environment while entities simulated far apart on the WAN may be close together in the virtual environment. The individual nodes and the associated software are referred to as simulators; the collection of simulators executing together is a simulation. The infrastructure is the hardware and software which provide supporting services to the simulation, but don't perform any of the modeling.

---

2. It takes approximately 8.25 ms to transit one time zone, one way on a WAN [Singhal99]. The remainder of the time is spent transiting the receiver's network interface hardware, OS kernel and the application layer.

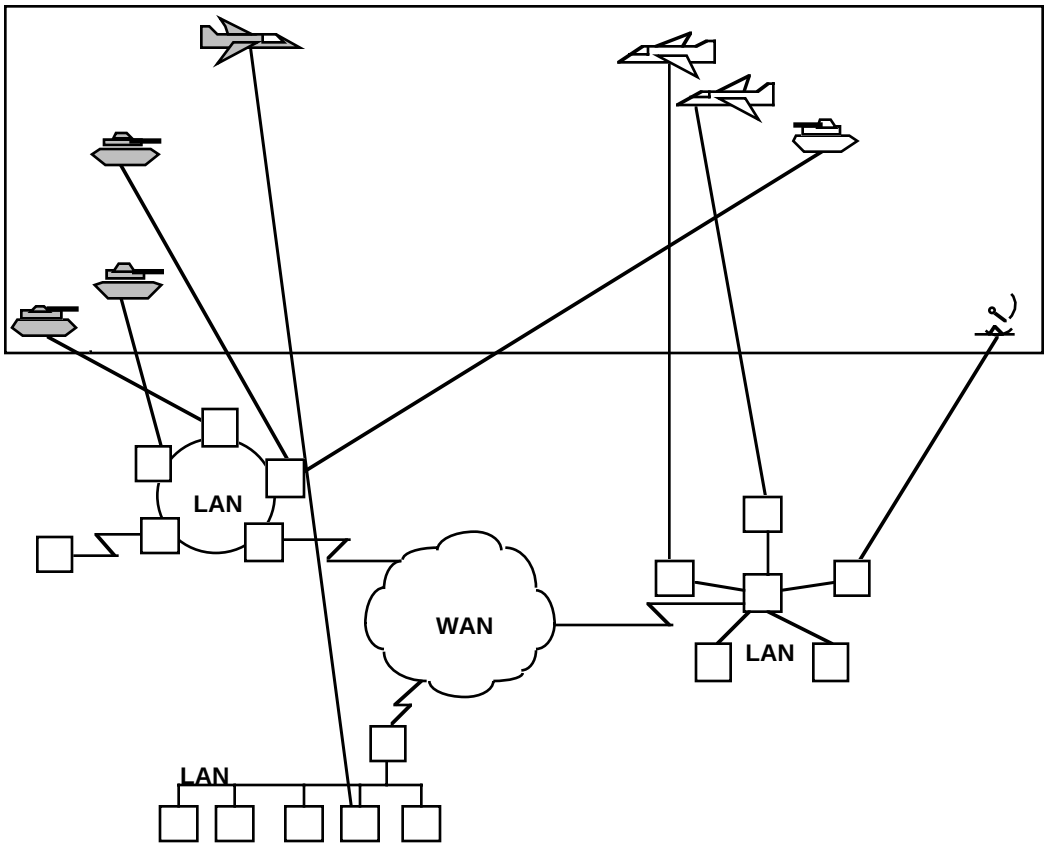

Figure 1-1. Virtual Environment Representation in a Network Environment

The LANs may be ATM, Ethernet, FDDI, ScramNet or SCI, the last for connecting via gateways to legacy systems which cannot be upgraded to interoperate with new systems and protocols. The WAN may be IP, ATM, or IP over ATM. Obviously the Internet could serve this purpose. The Defense Simulation Internet is a dedicated DoD internet which has been used for several LSVEs [DIS94a].

1.1.1 Scope

The network for an LSVE may encompass dozens of LANs supporting hundreds of host nodes. These nodes may support thousands of simulated entities. DARPA

envisions simulations encompassing 100,000 entities [DIS94a]. Simulated entities are grouped into three categories. *Live* entities are actual, physical objects which have communications support and instrumentation which allows them to interact with other entities in the simulation. *Virtual* entities are man-in-the-loop simulators such as tank or flight simulators in which the individual operator performs operations just as if the vehicle were in the field. *Constructive* entities are almost entirely synthetic. A single workstation controlled by an individual playing the role of a commander directs the operations of many simulated entities [DIS94a]. In the virtual environment, all three types of entities appear equally real to all participants. This feature supports the two primary missions of these types of simulations: training personnel on existing systems and testing the effectiveness of planned systems not yet in production.

1.1.2 A Brief History

SIMNET (1983 - 1992) was the precursor of most of the LSVEs surveyed in this paper. Based on the lessons learned with SIMNET, DARPA developed the *Distributed Interactive Simulation* (*DIS*) protocol which has subsequently become IEEE standard 1278 [DIS94]. DIS's primary goal is to support training exercises. The DIS protocol specifies the type and format of the *protocol data units* (*PDUs*) to be passed between entities such as Entity State, Fire, Action Response, Create Entity, and Transmitter. It also specifies responses to be given upon receipt of certain PDUs. Version 1.0 of the standard specified seven types of PDUs. The current version, 2.0.4 [IEEE 1278.1] specifies 27. The structure and content of the PDUs is completely specified, limiting the applications of DIS. Any time new semantics need to be added to a simulation, not only must all of the

simulation's software be changed, but the standard must be changed as well. The communication protocol for DIS is broadcast for all PDUs.

DARPA initiated the multi-service *Aggregate Level Simulation Protocol (ALSP)* program in 1990 to link both analytical and training simulations[Wilson94]. ALSP defines a general architecture for data exchange which allows the simulation (confederation) designers to decide what data will be exchanged, rather than having it specified by the protocol. At runtime, a simulator can tag data it wishes to share with other simulators and forward the tagged data to the infrastructure. The infrastructure forwards this data to other simulators which have requested data with the same tags. This process is flexible, but it has the potential to be non-deterministic because the tags can be defined at runtime. For example, if one simulator tagged a data item with the name "velocity" and another asked for "Velocity", the infrastructure would not forward the data.

Drawing on the lessons of DIS and ALSP, in 1995 the Defense Modeling and Simulation Office of DoD embarked on defining the *High Level Architecture* (*HLA*)[3], an initiative targeted at unifying almost all existing and future military simulations as well as providing an infrastructure for interoperation of non-military simulations [DMSO98]. The HLA also provides a general architecture and services for data exchange, allowing simulation (federation) designers to specify the actual data to be exchanged between simulators (federates). Like ALSP, data is tagged by simulators and sent to the infrastructure for forwarding to other simulators. However, the possible tags and types for

3. The HLA is the specification for an architecture. A simulation support system which meets the specification is referred to as a Run-Time Infrastructure.

data are specified at simulation design time, allowing the infrastructure to perform error checking and to optimize how much matching it must do before forwarding the data to other interested simulators. HLA is designed to support existing DIS and ALSP simulations through “wrappers” which convert between the DIS and ALSP protocols, and HLA service calls. For DIS, this entails breaking down PDUs into their constituent fields and tagging them individually before forwarding them to the infrastructure. For ALSP this entails deciding before execution what tags will be used and performing error-checking at runtime to preclude simulators from sending data with unknown tags. Our discussions of HLA assume the use of native HLA federates.

1.2 The Need for Interest Management

The DIS protocol is stateless and assumes unreliable, broadcast communication. Entities are required to periodically broadcast “heartbeat” *entity state PDUs* to accommodate entities joining the simulation late and to compensate for PDUs lost due to communication errors. Using this protocol for a simulation of 100,000 entities, each node in the network would require a 375 Mbps network connection [ADST92]. Entity State PDUs may account for 62 - 97% [RDTE95, RDTE96] of the data traffic. In some experiments, as much as 90% of the data is useless to the receiver [Rak96]. The receiver is responsible for sorting through and discarding the useless messages, unnecessarily consuming processor cycles. This process cripples the performance of the simulations and restricts their scalability.

ALSP and HLA do not have to contend with heartbeat state messages, but they are designed to support very large scale distributed simulations and must address scalability as simulations grow. Using broadcast protocols, the volume of data can still exceed the receivers ability to sort it and determine what is relevant, e.g. a tank simulator may care about other tanks within a radius of 5 km, but it probably doesn't care about soldiers on foot 50 km away.

The concept of Interest Management[4] was developed to address these problems by reducing the received messages to a smaller, relevant set. Under Interest Management, a simulator expresses its data interests in terms of location and other application-specific attributes. The data interests are referred to as the *area of interest (AOI)* and usually correlate with the sensing capabilities of the system being modeled. For the different sensory modalities, the area of interest may be different sized and shaped. The simulator maps its AOI into a representation that is usually simplified and is known as an interest expression (IE). Other components of the simulation infrastructure, *interest managers (IMs)*, accept the simulators' IE and use them to filter messages to sets (or reduced supersets) which meets the receivers' needs.

1.2.1 Interest Expressions

An IE is a specification of the data one simulator needs to receive from other simulators in order to interact with them correctly. IEs may refer to several attributes of the sender's simulator or entities. They may be geographic. They may refer to some

4. Interest Management is also referred to as relevance filtering, data distribution management, and data subscription.

attribute of an entity including its type. A tank may want to know about all ground vehicles within a four kilometer radius around itself, but a soldier only cares about entities within a 400 meter radius. An airborne surveillance radar may express interest in all ships within a radius of 30 nautical miles. IEs may also specify a reduced resolution or frequency of data. A wide-area viewer may need data about all entities in the simulation, but not every time new data is generated. Updates every few minutes may be sufficient. For a small virtual environment on a few hundred square kilometers the problem may not seem insurmountable, but at least one virtual environment already implemented was 15,000 square kilometers [Mastaglio95].

1.3 Taxonomy

Having described the general characteristics of all Interest Management schemes, it's now possible to discuss variations on these characteristics and to develop a framework for classifying individual implementations. After analyzing the thirteen systems surveyed, the most general observations that can be made about Interest Management schemes is that they must have an architecture for performing filtering and a means for simulators to express their interests.

1.3.1 Interest Expression Specification

An interest expression is an approximation of a real entity's area of interest. The AOI in the previous tank example was for all ground vehicles in a four kilometer radius. However, an AOI can encompass any type of data, e.g. radio frequencies. In fact, the tank example also includes the requirement that the detected entity also be a ground vehicle.

1.3.1.1 Specificity

Specificity defines the expressive capability of the language or formalism for specifying interest expressions. It is the form which the interest expression can take.

Formulas[5]. The most obvious form is a mathematical or predicate formula of some of the examples given, e.g. class(entity) = ground_vehicle & distance(entity, self) < 4, might return the state of all entities whose class is ground_vehicle and whose distance is less than 4 kilometers from the requesting entity. Conceptually this is clear, but the implementation is complex. This simple example assumes that the IM can resolve the distance calculation between the requesting entity and all other entities, and the IM knows that the units are kilometers. Notice also that the entities which match this formula when it's first expressed will almost certainly not be the entities that match it later as the requester and all other entities move in the virtual environment as the simulation progresses. The IM would have to continually update the result of the formula as the requester moves. In practice, a system using formulas would require the requester to specify the IE in absolute terms and to update it as the basis for the formula changes. The management overhead of such systems is typically high because formula evaluation is costly and each message may have to be evaluated against many formulas.

Cells. A simpler implementation is to divide up the "space" statically ahead of time into discrete cells. All participating simulators know the bounds of the cells, so they can tell by the position of their own entities which cells to send and receive data from.

5. Expressions is a more general term for this concept, but formulas are use here to avoid confusion with interest expressions.

There can be an explicit mechanism for managing which simulators are “joined” to a cell for sending or receiving, or the process can be managed implicitly using multicast. Message processing time can be improved because messages are simply sent to the cells without having to compare each message to a formula. Cell-based implementations require little overhead to manage, but the discretizing effects of the cells can negatively impact filtering quality. Even if an entity’s interest range is logically small, if the entity’s position falls near the edge of a cell it may overlap several cells causing it to receive data which is really not of interest.

Extents. Extent-based implementations can strike a balance between formulas and cells. The filtering space is decided ahead of time and IEs are limited to n-dimensional rectangles in the space, but the range and location of the IEs are expressed at run time. The extents are more precise than cells, but less precise than formulas. The extent intersection calculations are easier than formula matching and only have to be calculated when the extents change, rather than for each message sent, but the management overhead is still higher than it is for cells.

The figures below illustrate the difference between these approaches. Figure 1-2 shows a tank on a virtual battlefield. The circle around it represents its area of interest. If the system under consideration supports formula IEs, the tank’s IE would exactly correspond to the circle. If the system is using cell IEs, the tank’s area of interest would be mapped to the nine cells marked with the shaded box in Figure 1-3. This area of interest would be mapped to the square in Figure 1-4 if the system were using extents.

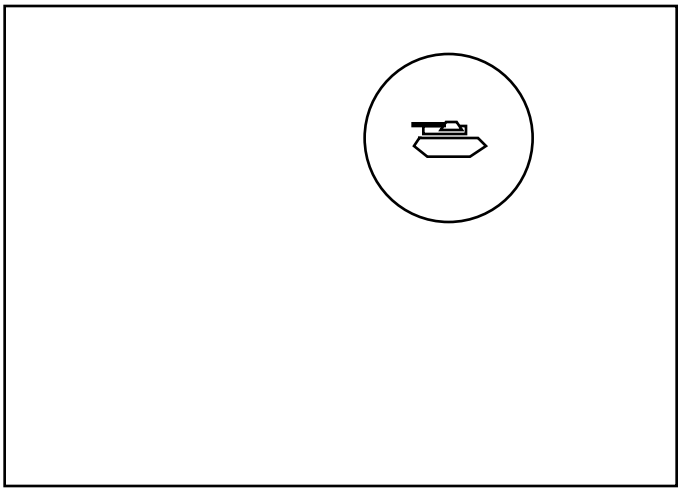

Figure 1-2. Area of Interest Example

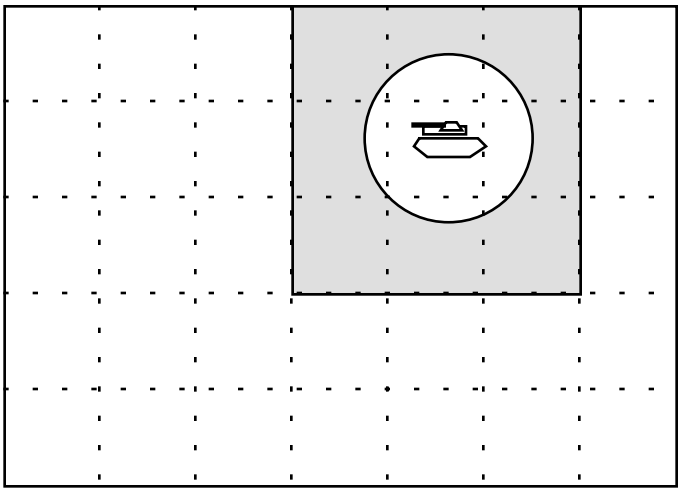

Figure 1-3. Area of Interest with Cells

1.3.1.2 Precision

Specificity describes the difference between a simulator's AOI and the form of the IEs it's allowed to use to express it, i.e. the difference between the data it wants and the data it's allowed to request. Precision is the difference between the data the simulator requests and what it receives. It is the degree to which the IM delivers only the data requested and no more. For an IM to be correct it must deliver at least the data requested.

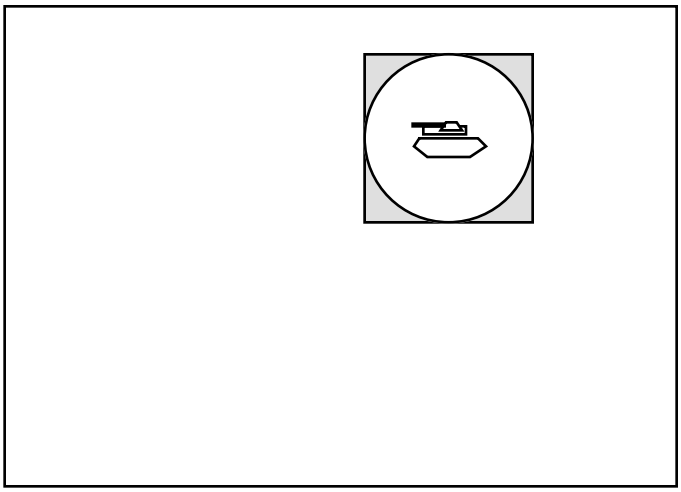

Figure 1-4. Area of Interest with Extents

Precision may be Exact or Approximate [Petty97]. To understand how this is different than specificity, consider the following implementation. The IEs are specified as formulas, but the underlying implementation maps the formulas to a predefined grid. This mapping is invisible to the user. The user only gets approximate results; in fact the results delivered would be exactly those illustrated in Figure 1-2. This is probably not a good design decision because it would pay the computational cost of evaluating formulas, but only deliver results at the precision of cells. These results could have been gotten for significantly less overhead.

1.3.1.3 Query Domain Selection Time

The query domain selection time is the time at which the domain of IEs is fixed. Choosing an early query domain selection time increases opportunities to optimize performance of the IM system while choosing a late query domain selection time increases potential flexibility.

System Design. If the query domain is selected at system design time, the IM user has no control over it. IEs may only be over values of variables fixed in the system. For example, an IM system for DIS could restrict IEs to be over the fields in the PDUs. Since the fields in the PDUs are fixed in the standard, the query domain is fixed at system design time. For example, location in the Entity State PDU is defined to have X, Y, and Z components which are 64-bit floating point numbers.

Execution Initiation. A more general approach is to allow the simulation execution designers to specify the query domain during their design. They agree about the variables for the IEs, but once the simulation execution begins no new ones can be added. This provides more flexibility than fixing the query domain at system design time, but affords some opportunities for optimization because the IM can be initialized with information about the query domain. The simulation designers could agree that all of their entities are going to operate on the ground, so only X and Y components are needed.

Runtime. The most flexible approach is to allow the query domain to be specified at runtime. The IM would provide only the syntax and framework for evaluating IEs, but any participating simulator could request any data and hope that some other simulator will provide it. Some simulators could use only X and Y components while others also use Z.

1.3.1.4 Frequency/Temporal

It may be the case that some simulations may not want or need every update from other simulators. If one participant is a plan view display, it literally only wants to see the

big picture. It needs to know about every entity being simulated, but only needs to get every n^{th} update from each one to have a sufficiently detailed view. In fact, it would be overwhelmed with data if it received all updates. An IM would support such a requirement with a frequency control. A variation on this would be a temporal control which would allow a simulator to request that it only receive updates every n units of wall clock or simulation time.

1.3.2 Architecture

1.3.2.1 Software Infrastructure Support

Although up to this point IMs have been discussed as if they are an actual component of the system, in some cases the IM is an implicit agreement between cooperating simulators without any other software infrastructure. For example, a cell-based IM with multicast groups assigned to the cells a priori may require only a local library at each node to resolve entity locations with respect to the grid. Participating simulators join and leave multicast groups directly based on the locations of their entities and interests. However, many systems have an actual software infrastructure which hides the filtering implementation from users. The shaded boxes in Figure 1-5, Figure 1-6, and Figure 1-7 indicate all the nodes in the network that can support all or part of the software infrastructure. Note also that the infrastructure supporting a simulator may or may not be collocated on the simulator's node. If it is collocated with the simulator, the infrastructure may take many forms including a library or a separate process. The separation of infrastructure from simulators in the figures is intended to be notional and does not necessarily imply physical separation.

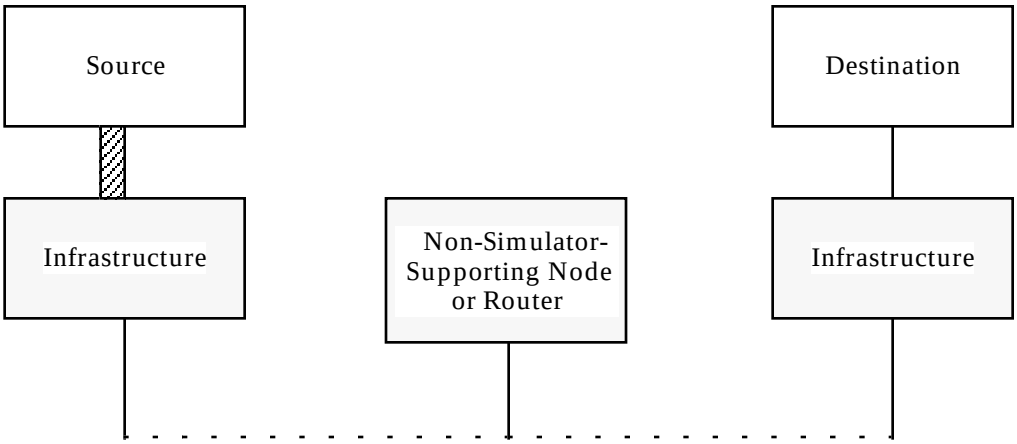

Figure 1-5. Source-Based Filtering

1.3.2.2 Network Location

Filtering may be performed at any of three locations in the network architecture: Source, Destination, or Intermediate. Source and destination represent the actual nodes where the data is sent or received, respectively. An intermediate location may either be another node or a multicast router. Source-based filtering has the highest potential for reducing network bandwidth use. As Figure 1-5 illustrates, the bulk of data generated at the source is filtered before it can reach the network.

Destination-based filtering is usually the most precise because it can be configured to support just the local simulator, but it provides no network bandwidth use reduction as Figure 1-6 shows, nor processor cycle reduction. Both approaches can be either beneficial or detrimental in terms of CPU usage, depending on whether the sender's or receiver's CPU is more heavily loaded.

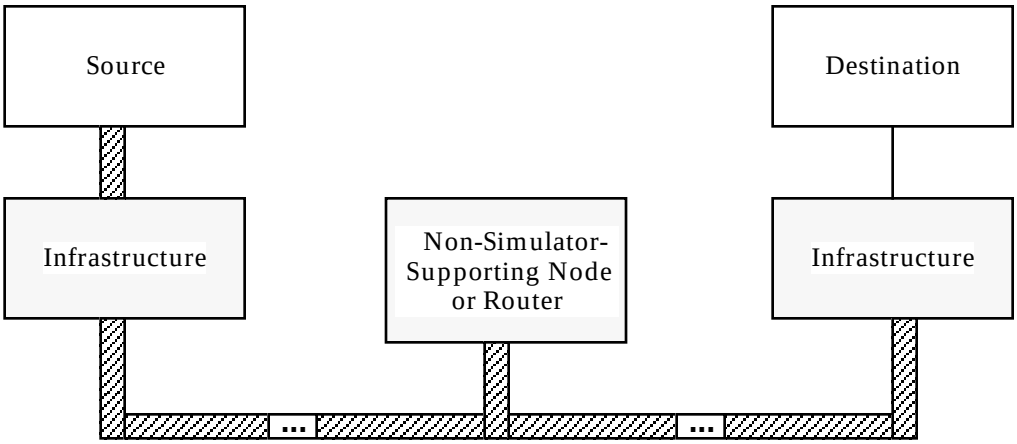

Figure 1-6. Destination-Based Filtering

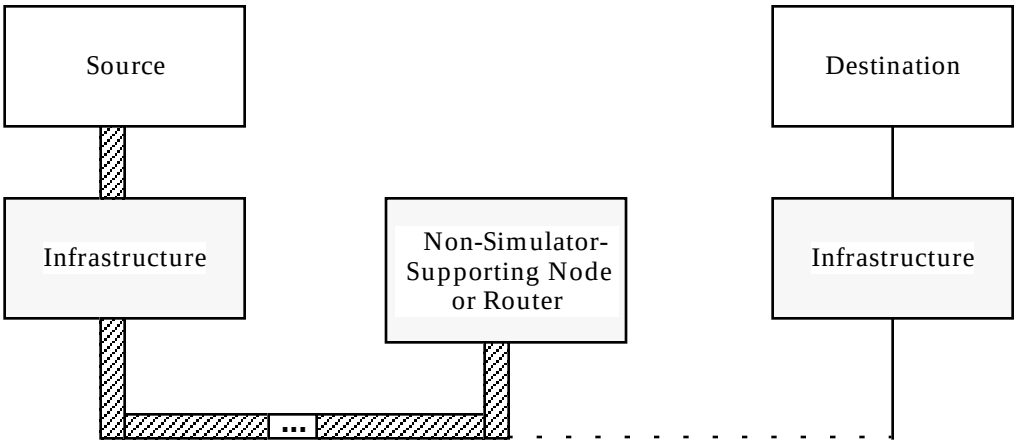

Figure 1-7. Intermediate Filtering

As Figure 1-7 illustrates, this issue can be mitigated by filtering at an intermediate location, but the resulting precision may be lower than either source-based or destination-based filtering because it may group the filtering needs of multiple senders and receivers.

1.3.2.3 Multicast Support

Multicast has arisen as an implementation option several times. This is multi-point to multi-point rather than point to multi-point. Multicast is attractive because it can reduce the network load and the processor cycle load. However, the applicability of multicast is reduced by the restriction on the number of groups that current multicast hardware can support[6] and the time it takes to reconfigure the groups. The latter can be on the same order of time as that allocated for sending a message, including formulating it, passing it through the local node's call stack, network latency, retrieving the message, and unpacking it at the receiving node [Mastaglio95].

1.4 Implementations

With this taxonomy for classifying implementations, the relevant characteristics of the thirteen systems can be reviewed and their interest management implementations classified accordingly. Since the military is currently the largest user of LSVEs, nine of the implementations are from the military simulation domain. The other four implementations are from various civilian simulation domains.

1.4.1 ModSAF

Modular Semi-Automated Forces (ModSAF) [Smith95, Russo95, Vrablik95] is a constructive simulator designed for the Army and fielded in the 1990s. It began as a simple tool to generate targets for initial operator training in tank simulators. The original implementation was in Lisp on a BBN Butterfly. It has since evolved through programs

6. [3Com] lists a limit of 6K, the highest number identified.

called SIMNET SAF, OdinSAF, and finally ModSAF. ModSAF is DIS compliant. It can scale to 850 entities running on 17 platforms, including SGI, Sun, MIPS, HP, DEC Alpha, IBM, and PowerPC. It was designed to be portable and to support many other simulators, so no specific exercises were performed. It was used in STOW ED-1 and NPSNET (both described below).

ModSAF uses the most basic cell-based scheme which filters PDUs by their type. The grid cells are assigned a priori and their bounds remain static throughout the exercise. IEs are hard-wired into code modules. The modules have code which registers receivers with a packet filtering mechanism to receive specific types of PDUs from entities in the grid cell of interest. The entities themselves do not express interest. In fact, entities are passive in terms of both movement and interest expression. Since a single simulator simulates many entities, movement and Interest Management are managed at an aggregate level rather than by individual entities. Most versions, including the current one, 2.0, use broadcast. Version 1.4 experimented with multicast to provide high- and low-fidelity information through a frequency filtering mechanism. Every other PDU is sent to alternating grids. An entity wanting low-fidelity information subscribes to only one grid. An entity wanting high-fidelity information subscribes to both. ModSAF can be characterized as having destination-based filtering. All PDUs arrive at all nodes, but are filtered by a code module before being passed to individual entities.

1.4.2 JPSD

Joint Precision Strike Demo (JPSD) [Powell96, Watson95] is a prototype simulator built between 1993 and 1996 to train Army tactical commanders. JPSD uses DIS PDUs and runs in real time, but is not DIS compliant. It was designed to support 8,300 constructive entities. A recent demonstration supported 5,000 to 6,000 entities for several hours running on 70 to 80 nodes. Forty of these nodes were located at a main site and connected to an ATM star configured with multicast groups to emulate Ethernet LANs. Six gateways connected the remaining nodes at several other sites, some via ATM.

IEs in JPSD are predicates in conjunctive normal form about attributes of entities. The terms can be is-equal, is-in-range, or function calls which return true or false. The IEs are compiled, but their variables, start time, and end time are bound at run time. Although not implemented, there were plans for an entity-centric Modeling Interest Language. Each node has a single IM for all the entities simulated on the node. Entities broadcast (or publish) their IEs to IMs on all other nodes. When a PDU is generated on a node, the local IM checks the contents of the PDU against all received IEs. A pass actor is created for each PDU which satisfies at least one IE. The pass actor collects the IDs of all remote entities whose IE the PDU passes and sends the PDU only after all IDs are collected.

Two versions of interest publication were implemented: static and dynamic. Static publication performs a priori analysis of PDU types that simulators need and sets up channels allocated at simulation startup for each type. Dynamic publication allows entities to publish IEs when they need the data. Obviously dynamic publication is a more

general approach. Although JPSD currently uses point-to-point unicast, the pass actor mechanism is designed to be integrated with multicast.

1.4.3 CCTT

Close Combat Tactical Trainer (CCTT) [Mastaglio95] is an incremental software and system integration of semi-automated forces and manned simulators to train Army tank and mechanized infantry forces. Begun in 1993 and run through 1996, it is the first fully DIS compliant system built. It can support 851 entities: 50 manned modules and 15 computer generated forces processors each able to simulate 60 vehicles. The manned simulators are connected via an FDDI LAN using TCP/IP and UDP/IP, with plans to migrate to a WAN architecture.

CCTT's Interest Management scheme uses a combination of grid-based and PDU type-based IEs. Entities can express interest in a collection of predefined square grid cells which are 5 kilometers on a side, and in a pre-specified subset of PDU types such as Entity State, Fire, etc. PDUs are filtered first on the basis of the originating cell and then on PDU type. Multicast groups are pre-assigned to grids and PDU types, and cannot be changed at run time. The local IM on each node (referred to as the network manager) joins and leaves multicast groups to meet the IEs of local entities. If the number of multicast groups a network manager needs to join exceeds the hardware limit of 64, the network manager must take up the slack by doing the filtering in software.

1.4.4 NPSNET

Naval Postgraduate School NET (NPSNET) [Macedonia94, Macedonia95, Macedonia95+, Abrams99] is a DIS 2.0.4-compliant 3D visual simulation testbed developed by the students and researchers at NPS. ModSAF is integrated with NPSNET, and 10 exercises have been conducted supporting up to 7 sites with up to 18 entities. It can support as many as 300 entities. The network architecture consists of SGIs connected via FDDI at the LAN level and using MBONE at the WAN level. An effort was planned to move to Realtime Information Transfer and Networking [Calvin95++]. NPS is currently implementing a virtual environment toolkit, Bamboo, that will allow NPSNET-V to support multiple protocols at the same time [Watsen98].

NPSNET uses hexagonal grid cells which more closely approximate a real world area of interest which would be round[7]. An entity's area of interest consists of a radius of grid cells where the entity is joining new cells at the leading edge and leaving old cells at the trailing edge as it moves forward. An entity is an "active" member of the cell in which it is resident and a "passive" member of all the other cells in its area of interest. PDUs are sent to and received from the grid cells without further filtering. Grid cells are pre-assigned to multicast groups in a straightforward, static manner. The mechanism for maintaining membership in the groups is distributed by making the oldest active member of the grid cell the group leader. The group leader is responsible for adding members, deleting members, and providing new members with an aggregate of all current Entity State PDUs for the cell. In addition, each node has a local IM. Macedonia, et al envision

7. This same idea was developed independently by the designers of the cellular phone network [MacDonald79].

IMs[8] based on spatial (geographic), functional (type), and temporal (frequency of updates) relationships. To date they have only implemented the spatial relationship with this grid cell approach.

1.4.5 STOW-E

Synthetic Theater of War-Europe (STOW-E) [VanHook94, VanHook94a, VanHook94b, RDTE95, VanHook96, Rogers95] is one in a series of STOW programs led by DARPA with participation by all the armed services. The STOW programs began in 1993 and are slated to run until at least 2000. These programs combine live, constructive, and virtual simulations to support readiness and operations, requirements determination, acquisition, and test and evaluation. Later STOWs are logical, but not necessarily physical successors of earlier STOWs, i.e. lessons learned are used, but not necessarily code. The STOW-E exercise was played in November 1994 on a European virtual battlefield using DIS 2.0.3 with experimental PDU additions. Three thousand live, constructive, and virtual entities participated at 24 sites, including 18 on the Defense Simulation Internet with the rest bridged legacy systems. STOW-E's filtering capability was provided by Realtime Information Transfer and Networking.

STOW-E uses grid-based filtering of Entity State PDUs. An entity's area of interest is referred to as its cell set. Each LAN has a node at the LAN-WAN boundary referred to as the Application Gateway. The Application Gateway supports the IM which collects the cell sets of all entities on its LAN, unions them, and broadcasts the union to all

8. Macedonia coined the term Area of Interest Manager (AOIM) in his Ph.D. dissertation [Macedonia95b].

remote Application Gateways. The remote Application Gateways union (cluster) all the received cell sets into a full accuracy region. This is the data of immediate interest to remote entities. All data generated on a LAN which is within its full accuracy region is broadcast by the Application Gateway to all other Application Gateways. Data generated on the local LAN, but outside the full accuracy region (within the reduced accuracy region) is broadcast with reduced frequency, e.g. only 1 in every n PDUs generated in the reduced accuracy region is broadcast. The reason for broadcasting the data at all is that entities may join and move, and their IEs may not take affect in time for them to get data that they may require immediately. Since an Application Gateway aggregates all local cell sets before sending them out to other Application Gateways, the data returned will not be of interest to all entities on all nodes. So, filtering must be done by both the source and destination Application Gateways using the full accuracy region and cell sets, respectively. Notice that an Application Gateway's full accuracy region changes as both local and remote entities move across cells.

1.4.6 STOW ED-1

Synthetic Theater of War Engineering Demo 1 (STOW ED-1) [VanHook96, Calvin95, Calvin95+], October 1995, was the first in a series of engineering demonstrations with increasing levels of functionality planned for the next STOW system. It uses DIS PDUs and runs in real time, but is not DIS-compliant. ED-1 encompassed 7 sites, 62 ModSAFs, and 3,000 to 5,000 entities. ED-1's filtering capability was provided by an upgraded version of Realtime Information Transfer and Networking.

ED-1 uses a two-level, grid-based filtering scheme with alternating grids (referred to as multiple fidelity/uncertainty channels) supported by multicast. This is similar to version 1.4 of ModSAF in which partial data is received by subscribing to one grid and complete data is received by subscribing to both. Several logical multicast groups were multiplexed onto a single physical group when the number of logical groups exceeded the hardware limit of approximately 1000. These groups were static.

Each node in ED-1 supports a local IM (Subscription Principal) responsible for aggregating IEs for entities on the node and forwarding them to a Subscription Agent. An Agent Host at the LAN-WAN boundary (similar to the Application Gateway in STOW-E) supports a Subscription Agent whose function is similar to the Subscription Principal's, but at the level of the LAN. The Subscription Agent collects the IEs which have been aggregated at the Subscription Principals on the nodes on the LAN. The Subscription Agent evaluates all the IEs and joins the multicast groups which coincide with the IEs. Likewise, when an entity's behavior necessitates deleting an IE, this information is passed to the Subscription Principal which forwards it to the Subscription Agent. The Subscription Agent leaves the associated multicast group if no other entity on the LAN has an outstanding IE which still requires membership in the group. Data received at the Agent Host via the multicast groups is broadcast back to the LAN where the Subscription Principals retrieve just the data coinciding with the IEs of local entities.

1.4.7 ALSP

Unlike the other systems surveyed thus far, the Aggregate Level Simulation Protocol (ALSP) is an architecture and protocol rather than a single simulation. It incorporates a protocol for message exchange between simulations as well as a distributed runtime simulation support and management infrastructure [ALSP97]. Several large simulations have been and continue to be implemented on top of ALSP, including confederations which are made up of systems large enough to be standalone simulations. The largest ALSP user is the Joint Training Confederation with twelve ALSP constituent simulators (confederates). One of the Joint Training Confederation confederates alone, Corps Battle Simulation, runs 10 to 12 exercises per year for a total of 50 exercises to date, supporting as many as 150 nodes.

ALSP recognizes two classes of data messages subject to filtering: attributes of persistent objects and interactions. For example, a tank would be represented as a persistent object and its simulator would periodically provide new values for its attributes. Interactions model non-persistent events such as radio messages or missiles. ALSP also has two types of filtering. Class filters eliminate object and interaction classes and attributes, regardless of their value. Attribute value filters eliminate attribute messages based on their values. They are specified as conjunctions of range checks for continuous-valued attributes or membership operations for discrete-valued attributes. The ALSP filtering architecture is multi-tiered. Each simulator in an ALSP simulation is logically attached to a single ALSP Common Module which performs class filtering at the sender. At the receiver the ALSP Common Modules perform class and attribute filtering. Each

ALSP Common Module is logically connected to an ALSP Broadcast Emulator. ALSP Broadcast Emulators facilitate the distribution of ALSP data, primarily by receiving data on one path and retransmitting it on another path. An ALSP Broadcast Emulator may be connected to multiple ALSP Common Modules or other ALSP Broadcast Emulators to form a multi-tiered distribution hierarchy. ALSP Broadcast Emulators also perform class filtering.

1.4.8 HLA

A key difference between the High Level Architecture (HLA) [DMSO98] and the other systems surveyed to this point is that the HLA specifies functional requirements, but not the hardware, software, or network architecture. An implementation of the HLA is referred to as a Run-Time Infrastructure. This makes it nearly impossible to survey every simulation (federation) that has been run on HLA, but early implementations of the Run-Time Infrastructure have supported legacy DIS, JPSD, training, engineering, and analysis federations. The Defense Modeling and Simulation Office envisions that some day there will be commercially available Run-Time Infrastructures designed to optimize performance for different classes of simulations.

The HLA supports two types of filtering: class-based using Declaration Management services and value-based using Data Distribution Management [Morse97] services. Declaration Management services filter object and interaction classes as well as attributes of object classes. Data Distribution Management services extend Declaration Management services using routing spaces and regions. A routing space is a

multidimensional parameter space where the dimensions are the values on which to filter. The sender (updating federate) declares that its object's position in the parameter space is within a (update) region of the routing space defined by a set of extents. A receiver (subscribing federate) declares its interest in receiving classes of data through another (subscription) region in the routing space. By calculating the intersection of update and subscription regions, the Run-Time Infrastructure can establish connectivity between senders and receivers for routing updates and interactions. Note that the sender's guarantee that the object is in the update region does not guarantee that the object is in the intersection with the receiver's subscription region, only that the two regions intersect.

1.4.9 STOW ACTD

The final STOW Advanced Concept Technology Demonstration (ACTD) was held in October 1997. It supported 7,000 to 8,000 entities running in real time on 450 nodes at 5 sites around the U.S. Approximately 400 of those nodes ran ModSAF. The WAN was IP over ATM and the LANs were 10 base T switched Ethernet. The LANs delivered 5 - 10 Mbits/second with 500 - 1000 packets/second arriving at the nodes.

Filtering was provided by a prototype HLA Run-Time Infrastructure [Rak96, Calvin 97] with update regions reduced to points and routing spaces mapped onto an underlying grid. Since the scenario was mapped out ahead of time, areas of the routing spaces which were expected to experience a high density of entity interactions were assigned smaller grid cells, or a variable density grid. Grid cells were assigned a priori to multicast groups, pushing the current multicast hardware capability with 3200 groups.

1.4.10 Proximity Detection

Unlike the other systems surveyed, this system had its genesis in parallel, optimistic simulation. It began as a non-DIS, constructive, ground war simulation executed on a Hypercube running the Time Warp Operating System [Steinman96, Steinman94]. The same technique was applied to an aircraft simulation of 1,000 to 2,000 entities on a cluster of SGIs running SPEEDES [Steinman92]. Both implementations used optimistic scheduling protocols and assumed reliable message passing. This system is included because it demonstrates the fundamental characteristics of Interest Management.

Proximity Detection uses a two-level, hierarchical filtering scheme where the coarse level is provided by active grid entities and the fine level is provided by the entities (sensors) themselves. As entities move they check in and out of the grid cells they occupy. The grids maintain lists of resident entities and manage the process of keeping all resident entities cognizant of each other. Entities use point-to-point unicast to send all their messages to all other entities in the current grid. The receiving entity performs more detailed filtering based on its own sensor model to determine if it can truly “sense” the other entity.

1.4.11 MASSIVE-2

MASSIVE-2 and its predecessor, MASSIVE-1, [Greenhalgh95, Greenhalgh97] focus on supporting awareness between participants in collaborative virtual environments. Example applications include a sports arena, a court room, and a 3D web browser. One

test supported eight participants at five sites in three countries. Current performance limits have been estimated at 20 mutually aware users and 1000 objects [Greenhalgh98].

MASSIVE-1 and MASSIVE-2 implement a spatial model in which objects have auras within three types of media: audio, visual/graphics, and text. Auras are automatically assigned and manipulated by the infrastructure. Their range is based on the medium and their location is updated when the object moves. When an object joins a world, it connects to a local aura collision manager which in turn passes the object to the appropriate world manager. World managers may be distributed throughout the network. When objects have overlapping auras, their local aura collision managers detect the collision and notify both of the objects so they can directly negotiate mutual awareness based on focus and nimbus[9]. Users may manipulate parameters in the focus and nimbus formulas to adjust their view and perception. MASSIVE-2 also implements third party objects (regions) which negotiate awareness between other pairs of objects. Other objects send via the multicast group(s) associated with its most local, fully containing regions. One multicast group is assigned to each region.

1.4.12 DIVE

The Distributed Interactive Virtual Environment (DIVE) [Hagsand96] was designed to support collaborative environments in which human avatars (actors) interact with each other and manipulate objects. Such environments are typically characterized by few avatars, but large numbers of other objects or very complex objects. So while they

9. Focus is roughly an object's "field of view". Nimbus is field of influence [Benford94].

support few large-scale objects, they must manage the distribution of large quantities of data. DIVE has a predetermined class hierarchy for representing 3-D graphical objects. It has been tested with 20 human participants on a wide-area network with Unix platforms.

One of the key classes in the DIVE hierarchy is a world. A world is a separate spatial domain and is assigned a multicast group. An actor enters a world through a gateway which is itself an object in the hierarchy. A collision manager recognizes this entrance as a collision and queries the DIVE name server for the multicast group associated with the world. Interest management within the world is typically performed on the basis of visual range. The collision manager for a world handles requests for actors and objects within a specified visual range. Based on volume intersections, the collision manager signals collisions to the colliding objects.

1.4.13 MAVERIK

Like DIVE, the Manchester Virtual Environment Interface Kernel (MAVERIK) and its distributed extension, Deva, [Cook97, Hubbold96, Hubbold98, Pettifer97, Pettifer98] are designed to support virtual environments with large, complex objects. However, MAVERIK focuses on large-scale industrial applications rather than human collaboration. MAVERIK has been tested on two applications: computer-aided design of complex process plants and building modeling. The former example has been tested with a plant model containing 5 megabytes of geometric data at an average frame rate of 10 frames per second. At least one Deva experiment supported 20 live users distributed between Germany, Sweden, and Manchester.

Deva has a client-server architecture where the global world view is maintained on a server which may itself be distributed. Client processes support individual users and download enough of the global view to maintain the user's view. The server is responsible for maintaining a consistent world view and performs all interest management functions. MAVERIK's interest management is handled by a spatial management system which can support different IM approaches. Currently it supports bounding volumes, hierarchical bounding volumes, grid cells, and hierarchical grid cells (including K-d trees and octrees). Users can also write callback functions and supply them to the spatial management system to customize filtering, e.g. by object type. The spatial management system interacts with the navigation and collision detection components as the user moves through the environment. IEs are automatically generated by the navigation component based on the user's location and the IM approach in affect. If an intersection of IEs is detected by the collision detection component, a callback is made to identify objects of interest.

1.5 Classification

Table 1-1 presents the classification of the thirteen systems surveyed according to the preceding taxonomy. The systems are arranged in increasing chronological order of their start dates.

Some simple and obvious trends can be observed by reviewing the classification in this format, such as the relationship between the use of extents and the approximate precision of the returned results, and the fact that no system has generally addressed temporal IEs. The general trend over time is away from query domain selection at system

design time toward runtime and execution initiation. Also, the cost of filtering is being spread across the network. STOW and ALSP explicitly distribute the filtering while HLA doesn't specify where it will be performed, leaving Run-Time Infrastructure developers free to decide which architecture best supports their performance requirements. Our time line also shows a trend away from building filtering directly into the simulations and toward the abstraction of software infrastructure support.

Table 1-1: Implementation Classification

		ModSAF	ALSP	DIVE	CCTT	JPSD[10]
Interest Expression Specification	**Specificity**	Cells	Formulas	Cells	Cells	Formulas
	Precision	Exact	Exact	Exact	Exact	Exact
	Query Domain Selection Time	System Design	System Design	System Design	System Design	Execution Initiation, Runtime[11]
	Temporal/ Frequency	yes[12]	no	no	no	no
Architecture	**Network Location**	Destination	All[13]	Intermediate	Intermediate	Source
	Software Infrastructure Support	no	yes	yes	no	yes
	Multicast Support	yes[14]	no	yes	yes	no
Start Date		1990	1990	1992	1993	1993

		NPSNET	MAVERIK	MASSIVE-2	STOW-E[15]	Proximity Detection
Interest Expression Specification	**Specificity**	Cells	Cells, Extents	Extents	Cells	Cells
	Precision	Exact	Exact	Exact	Exact	Exact
	Query Domain Selection Time	System Design	System Design, Execution Initiation[16]	System Design	System Design	System Design
	Temporal/ Frequency	no	no	no	yes[17]	no
Architecture	**Network Location**	Intermediate	Intermediate	Intermediate	Intermediate	Source
	Software Infrastructure Support	no	yes	yes	yes	no
	Multicast Support	yes	no	yes	no	no
Start Date		1993	1993	1993	1994	1994

		STOW ED-1	STOW ACTD	HLA
Interest Expression Specification	**Specificity**	Cells	Extents	Extents
	Precision	Exact	Approximate	Approximate
	Query Domain Selection Time	System Design	Execution Initiation	Execution Initiation
	Temporal/ Frequency	no	no	no
Architecture	**Network Location**	Intermediate	Intermediate, Destination[18]	not specified
	Software Infrastructure Support	no	yes	yes
	Multicast Support	yes	yes	not specified
Start Date		1995	1995	1995

10 11 12 13 14 15 16 17 18
,

One result which is initially deceptive is the apparent trend toward reduced precision. All of the earlier systems deliver exact results while the two newest, HLA and STOW ACTD, deliver approximate results. Except for JPSD and ALSP which use formulas, all of the earlier systems returned exact results in terms of *cells*. So a simulation could expect to get data from exactly the cells to which it subscribes, but it has no control over the precision of the cells. Because a simulation sets the size, and hence the precision, of its extents, HLA and STOW ACTD are actually capable of delivering more precise results to their simulations.

By inverting the view and looking at the systems grouped by some pairs of the IE specification and architecture characteristics, other trends become clear. Table 1-2 groups the systems by two IE Specification categories with some illuminating holes. Why have no systems been built with these pairs of characteristics? The answers lie in the relationship between the pair.

---

10. ALSP confederation.
11. Static vs. dynamic versions of JPSD.
12. Low- and high- fidelity multicast groups.
13. One type of filtering component works at the source and destination; the other type is "in between" in the logical architecture, but may also be hosted physically at the source or destination.
14. Later versions support multicast.
15. Embedded in an ALSP demo.
16. Depending on whether or not the user supplies custom filtering callback functions.
17. Reduced accuracy region only broadcasts 1 of every n PDUs.
18. Object and interactions are filtered through multicast groups, but unwanted attributes are filtered at the destination.

Table 1-2: Specificity vs. Query Domain Selection Time

Query Domain Selection Time	Specificity Cells	Extents	Formulas
System Design	ModSAF CCTT NPSNET STOW-E STOW ED-1 Proximity Detection DIVE MAVERIK[16]		
Execution Initiation	Exact	HLA STOW ACTD MASSIVE[19] MAVERIK[16]	JPSD[11] ALSP
Runtime			JPSD[11]

19

What would it mean to have extents specified at system design time? Since the extents could not be changed once they were set, this would amount to variable- rather than uniformly-sized cells, but they're not extents because they cannot be changed. So, systems that use extents must at least have the flexibility to specify them at execution initiation.

Why are there no systems that set the query domain to formulas at system design time? A system that filters DIS PDUs on the values of the fields would fall into this category. The most plausible explanation is a performance consideration. Significant coarse filtering can be achieved using class-based filtering and multicast as CCTT did. Looking at the PDU in any more detail any place other than in the receiving simulation would probably be wasted effort.

---

19. MASSIVE is a more restrictive case because the filtering space (geographicl location) is decided at system design time and only the extents are specified at run time.

The lack of systems allowing specification of cells or extents at runtime can be explained by observing what kind of information is necessary to specify them. Because the cells or extents can be any number of ranges over all possible parameter spaces, the infrastructure would have to be able to understand specifications in those terms. Being able to do that amounts to being able to specify formulas.

A similar argument can explain why there are no systems that specify cells at execution initiation time. If the infrastructure has enough flexibility to resolve cells at execution initiation time, it can specify extents which are more general.

Table 1-3 groups the systems by two Architecture classifications. Unlike the grouping in Table 1-2, there are systems in all possible groups.

Table 1-3: Multicast vs. Software Infrastructure Support

		Multicast	
		yes	**no**
Software Infrastructure Support	**yes**	HLA[20] STOW ACTD MASSIVE-2 DIVE	HLA[20] JPSD STOW-E ALSP MAVERIK
	no	ModSAF CCTT NPSNET STOW ED-1	Proximity Detection

20

The performance optimization possible with multicast is equally applicable to approaches both with and without the abstraction of a software infrastructure layer. The

20. Multicast is neither required nor precluded in HLA.

fact that Proximity Detection and STOW-E system used the cell-based approach like CCTT and NPSNET suggests that they too could have used multicast.

1.5.1 Implications for Multicast Research

The effective use of multicast for JPSD and ALSP probably would require development of efficient algorithms for grouping the results of formulas which evaluate to the same receiver. This requirement suggests interesting future research. Efficiency will be the key to these algorithms since their use must not cause realtime simulations to exceed their time constraints. This time requirement also impacts the use of multicast since the time to reconfigure routers can approach the limit of the time constraint. And as the STOW ACTD learned, the number of multicast groups that current hardware can support is limited.

This analysis shows that the systems built to date nearly cover the space of possible implementations within the range of the taxonomy. No doubt other systems could be built, but they would probably map into the taxonomy as a variation of a previously built system with the addition of application domain specific optimizations. However, this does not mean closure has been reached on all possible research in the area. A review of the systems surveyed reveals that they all share the characteristic that the elements of the IM infrastructure are statically located in the overall architecture. HLA and ALSP allow simulation designers to build different network configurations and infrastructure architectures, but once they're built the filtering responsibility is fixed in components of the architecture. Nothing in our analysis of the requirements for the

architecture suggests that it must be statically defined; it is only the case that it has been in all the systems thus far.

Research is underway on filtering systems where the responsibility for filtering is dynamically distributed throughout the architecture [Saville97, Watsen98]. The goal of these systems is to improve scalability of LSVEs even more by further decentralizing filter processing and making better use of system resources such as CPU cycles and network bandwidth as the execution proceeds. Coupled with dynamic allocation of multicast groups, this research has the potential to reduce both filter processing overhead and message latency[21].

The development of large scale distributed simulations introduced the problem of data overflow. The first generation of interest management systems sought to optimize data transmission. However, in doing so they introduced a second, but smaller problem: overhead to support interest management. Having exhausted most of the possibilities for improving data transmission, the focus has now turned to optimizing the optimizers, i.e. improving the performance of interest management systems. The use of multicast groups is the first such attempt.

There is a trend away from purpose-built simulation systems, and consequently filtering subsystems, and toward general-purpose frameworks, and consequently general-purpose filtering components with later resolution of their query domains. Later

21. See [Zyda97] pg. 46 for an overview of the need for this work in both DoD and the entertainment industry.

resolution of the query domain means more flexibility for the systems in specifying their filtering requirements, and hence better precision in the received results. Such trends will be mitigated and balanced by performance requirements, possibly pushing toward multicast solutions as that technology matures to support more multicast groups as well as faster router reconfiguration. Performance and scalability requirements are also pushing filtering systems to more distributed architectures which reduce the impact on hosts supporting simulations.

CHAPTER 2: Problem Statement

2.1 Overview Of Data Distribution Management In The HLA

The High Level Architecture describes simulations in terms of *federations* of *federates* where a federation is a single, possibly networked, simulation comprised of simulator federates executing together through a Run Time Infrastructure (RTI). The HLA is a specification, not an implementation. The specification is comprised of three documents: the Interface Specification [DMSO98], the Rules [DMSO98a], and the Object Model Template [DMSO98b]. These three documents are currently in balloting for an IEEE standard, 1516.1, 1516, and 1516.2, respectively. This research focuses on the specification of DDM described in [DMSO98].

An RTI is an implementation meeting this specification. Figure 2-1 [1]shows a logical view of an HLA federation. RTI implementations may or may not have a physical architecture which matches this logical architecture.

Entities simulated in a federation are called *objects* and are associated with federates. Objects' states are described by the values of their *attributes*. An *object instance* is an instantiation of an *object class* within the federate. Likewise, *instance attributes* are the actual attributes of *object instances*, while *class attributes* are the specifications of the

1. I'm indebted to Dr. Judith Dahmann, Chief Scientist DMSO, for this graphic.

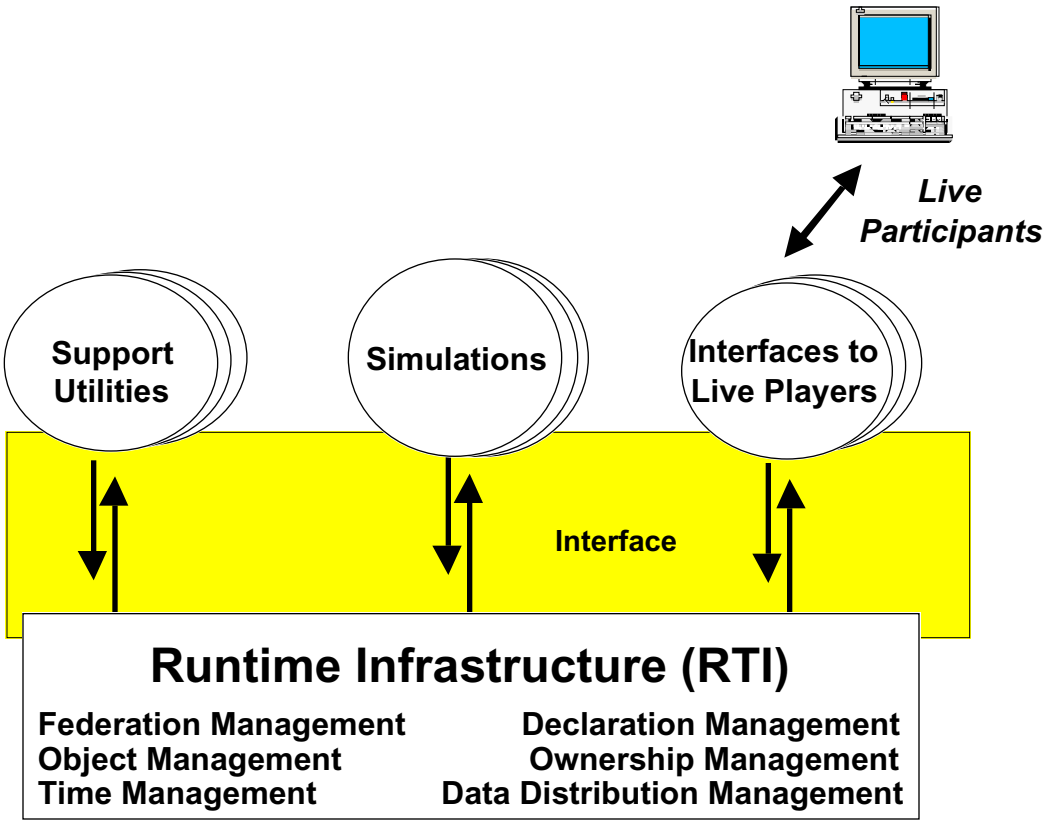

Figure 2-1. HLA Federation Logical View

attributes of an object class. The Interface Specification describes six service classes for supporting federations:

- Federation management - manages creation, control, modification and deletion of a federation execution;
- Object management - manages the creation, modification, and deletion of objects;
- Declaration management - controls the distribution of object instance attributes between federates on a *class* basis;
- Ownership management - manages transfer of instance attribute ownership between federates;

- Time management - controls the advancement of simulated (logical) time;
- Data distribution management - controls the distribution of object instance attributes between federates on an *instance* basis.

In the absence of DDM, the process for exchanging data between federates has the following basic steps:

- A federate declares its intent to send class attributes of an object class via the Declaration Management service Publish Object Class.
- Other federates declare their interest in receiving specific class attributes of an object class via the Declaration Management service Subscribe Object Class Attributes.
- The publishing federate registers individual instances of the published object class via the Object Management service Register Object Instance.
- The RTI matches subscriptions and publications and determines if attributes of registered objects are of interest to subscribing federates. If they are, these attributes are considered to be *in scope* for the subscribing federates. The RTI notifies these subscribing federates that they need to know about the appropriate object instance via the Object Management service call back Discover Object Instance. This step establishes connectivity between federates, a concept which will be of particular importance in the discussion of DDM.
- The RTI may also inform the subscribing federates which specific attributes are in scope via the Object Management service call back Attributes In Scope.
- The publishing/registering federate updates the values of the registered object instances via the Object Management service Update Attribute Values.

- The RTI checks the connectivity requirements for the updated instance attributes and sends them to the federates for which they are in scope. The receiving federates are notified via the Object Management service call back Reflect Attribute Values.

The fundamental Data Distribution Management construct is a *routing space* [Morse97]. A routing space is a multidimensional coordinate system through which federates either express an interest in receiving data (subscribe) or declare their intention to send data (update). These intentions are expressed through:

- *Subscription Regions*: Bounding routing space coordinates that narrow the scope of interest of the subscribing federate[2].
- *Update Regions*: Bounding routing space coordinates which are guaranteed to enclose an object's location in the routing space.

Both subscription and update regions can change in size and location over time as a federate's interests change or an object's location in the routing space changes.

An object is discovered by a federate when at least one of the object's attributes comes into scope for the federate, i.e. if and only if:

- the federate has subscribed to the attribute
- the object's update region overlaps the federate's subscription region.

---

2. Regions in a multidimensional routing space do not necessarily map to physical geographical regions. A region in a routing space should be thought of as an abstract volume with any number of dimensions, e.g. radio channels.

This second point is an addition to the definition of scope when only Declaration Management is used. DDM enables federates to specify by object class and attribute name the types of data they will send or receive, while also narrowing the specific instances of data. Each federate decides which of the federation routing spaces are useful to them and defines the portions of those routing spaces that specify *regions*, or logical areas of interest particular to the federate, by putting bounds (*extents*) on the dimensions of the selected routing space.

In the process of specifying a subscription region, the federate tells the RTI it is interested in data which fall within the extents of the region specified by that federate. Specifying an update region and associating that update region with a particular object instance is a contract from the federate to the RTI that the federate will ensure that the object instance is "within" the region. This means that the characteristics of the object which map to the dimensions of the routing space fall within the extents of the update region at the time that the attribute update is issued. This implies that the federate is monitoring these added characteristics for each of the attributes owned by the federate. As the state of the objects change, the federate may need to either adjust the extents on the associated regions or change the association to another region.

Figure 2-2 shows one update region (U1) and two subscription regions (S1, S2) within a two dimensional routing space. In this example, U1 and S1 overlap so attribute updates from the object associated with U1 will be routed to the federate that created S1.

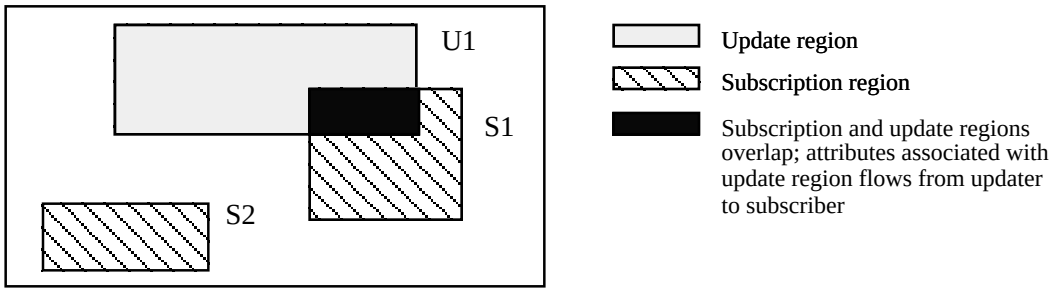

Figure 2-2. Two-Dimensional Routing Space Example

In contrast U1 and S2 do not overlap so attributes will not be routed from the federate that created U1 to the federate that created S2.

When an update region and subscription region of different federates overlap, the RTI establishes communications connectivity between the updating and subscribing federates for the attributes associated with the update region. The subscribing federates each receive only the instance attributes to which they subscribed, although they may receive individual updates outside their subscription region depending on the precision of the routing space implementation. In Figure 2-2, S1's federate will receive attribute updates from the object associated with U1 because their regions overlap, even though the object itself may not be within S1.

Each federate can create multiple update and subscription regions. Update regions are associated with individual objects registered with the RTI. A federate might have a subscription region for each sensor system being simulated.

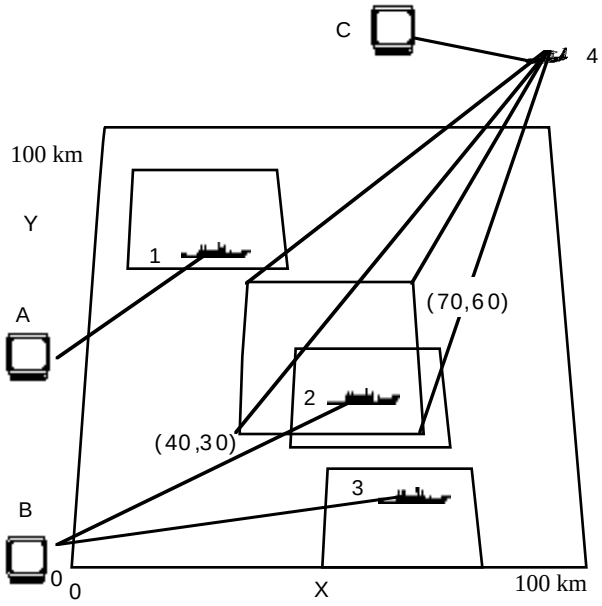

Figure 2-3. Geographic Routing Space Example

An example will serve to illustrate the specific flow and interaction of DDM, Object Management, and Declaration Management services. Figure 2-3 shows an example of a geographic routing space based on position and range.

1. The federation defines a 2 dimensional routing space with dimensions X and Y, based on the Position attribute, each with range [0 km, 100 km], for routing attributes force affiliation and velocity.
2. Federates A and B publish object class ship.
3. Federate A creates one update region and registers a single ship with the region, receiving object ID 1 in return.
4. Federate B creates two update regions and registers two ships with these regions, receiving object IDs 2 and 3 in return.

5. Federate C publishes object class aircraft and registers a single instance, receiving object ID 4 in return.
6. Federate C subscribes for velocity and force affiliation of ships in the subscription region specified to the RTI by the extent {{40, 70},{30, 60}}.
7. Federates A and B send attribute updates of force affiliation.
8. The RTI only reflects affiliation of ship 2 to federate C; ships 1 and 3 are filtered.

The federation begins the process by defining and creating the routing space with dimensions. Several important quantities are specified in each routing space:

- Name of the routing space
- Number and names of dimensions in the routing space

The federates must agree on the use of each dimension, i.e. the routing variables on each axis.

When an object that is participating in any of the routing spaces is created by a federate, its initial update region (or regions) in each of its relevant routing spaces must first be created using Create Region. The object is then registered with the RTI using Register Object Instance With Region. As a result, IDs are associated with the objects. In the example, federate A's ship has ID 1; federate B's ships have IDs 2 and 3; federate C's aircraft has ID 4. During registration an object's relevant published attributes need to are *associated* with the update regions. Federates A and B associate the force affiliation attributes of ships 1, 2, and 3 with their respective update regions as part of the object registration process.

Possibly simultaneously, other federates are creating subscription regions in the routing spaces, also using Create Region, so that it can discover object instances of interest. As time progresses, these two pieces of information can be updated using the Modify Region service according to a consistent and correct strategy that is associated with the routing space. It is up to the federates to define how to use a routing space correctly. In the example, federates A and B would create update regions around the initial positions of their ships. Federate C creates a subscription region corresponding to the sensor range of the aircraft.

The RTI determines which federates should discover which objects by matching subscription and update regions. Federates are notified by the Discover Object service of objects that meet the federate's subscription requests, i.e. are in scope. Object discovery is provided only once by the RTI to a federate even when multiple locally-defined subscription regions overlap a remote object's update region. Federate C is notified by the RTI to discover federate B's object 2.

A federate updates the attributes of an object as needed. The attributes are reflected in each of the federates that have discovered the object via the Reflect Attribute Values service. Even though federates A and B may update the attributes of ships 1, 2, and 3, federate C only receives the updated force affiliation attribute of ship 2 because it is the only one that federate C has "discovered". The updated attributes of ships 1 and 3 may be sent to other federates (not shown) that have discovered them via other subscription

regions, or their attributes may not be sent anywhere if no other federate has subscribed to them.

As an object's state within its federate changes over time, it might become necessary to modify update or subscription regions. Note that this does not have to occur every time an attribute is updated. When an update region or a subscription region is modified, the changes are passed to the RTI using the Modify Region service. The RTI then reassesses the matches between the modified region and all regions of the complementary type.

When a federate wants to delete an object or wants to stop using DDM with the object's attribute updates, it must first break any associations with update regions using the Unassociate Region For Updates service. This triggers the RTI to recalculate overlaps with subscription regions in the same routing space. Then the federate can use the Delete Region service to notify the RTI that it can delete any data structures associated with the update region. Likewise, when a federate wants to stop using a subscription region, it must first call Unsubscribe Object Class With Region before it can call Delete Region for the region.

In summary, when a federate's subscription region no longer overlaps the update region of an object, for any reason, the DDM services must coordinate actions so that the subscribing federate is notified that the attributes associated with the object instance are

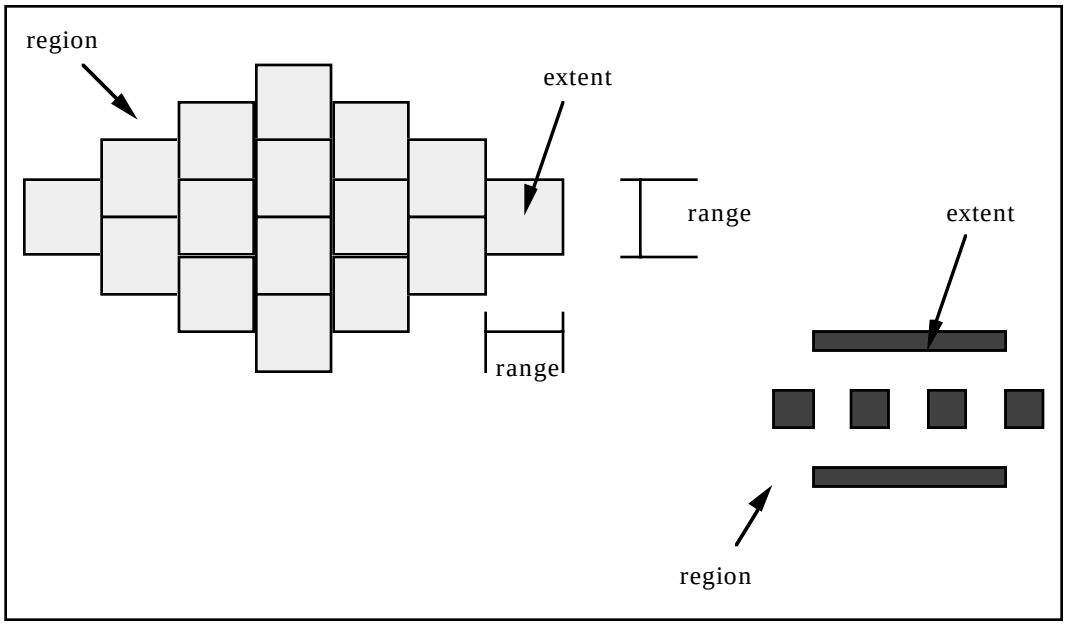

Figure 2-4. Multi-Extent Regions

out of scope (assuming that no other locally defined subscription regions overlap the update region of the object).

For the sake of simplicity, regions have been described as n-dimensional hyper-rectangles up to this point[3]. In fact, they are defined as sets of extents, or sets of n-dimensional rectangles. Regions which are not logically rectangular can be approximated by sets of smaller rectangles. Figure 2-4 shows an example of a two-dimensional routing space with two regions, one composed of 16 contiguous square extents and one composed of 6 non-contiguous rectangular extents.

3. The most common application of regions is to geographical 3-space, but the concept of regions is not limited to this.

2.2 Multicast Grouping

The most promising optimization identified to date is the use of multicast groups for routing data to a controlled subset of all simulators in a simulation [Abrams99, Calvin95b, Macedonia95, Mastaglio95, Rak96]. The ultimate measure of effectiveness of any interest management system is the latency between sending a piece of data and an interested receiver getting it. Broadcast makes sending fast, but at the expense of time spent by the receiver discarding irrelevant data. Point-to-point ensures that receivers only get relevant data, but it requires determining the destination for and sending multiple copies of messages, slowing transmission. The use of multicast strikes a balance between broadcast and point-to-point by reducing the time to send and the amount of data received. Broadcast and point-to-point represent opposite ends of the send/receive time spectrum with various applications of multicast occupying the area in between. Even though multicast has the potential of improving communication time, it is not without its own challenges.

- Network multicast hardware currently supports a limited number of multicast groups, on the order of a couple thousand.
- Network interface cards support even fewer multicast groups in hardware, sometimes as few as 16, and the trend does not appear to be toward supporting more.
- Multicast is not fully deployed on the Internet. IPv4 specified 28 bits for class D multicast addresses, while IPv6 allots 112 [Huitema96]. But, multicast has not been broadly used under IPv4.

- The time to reconfigure multicast routers can be of the same order as the total allowable latency for message delivery [Greenhalgh97].

As a result, most implementations using multicast to date have used static assignment of multicast groups, usually to fixed geographic regions. These implementations have achieved good results, but ultimately they are limited in scale as well because they do not account for changing connection patterns between senders and receivers. The next step in optimization is dynamic multicast grouping that adapts to connection patterns.

2.2.1 Connection Graphs

A federation with 11 federates each supporting 200 objects can have a connection from each object to each other federate, or approximately 11*200*10 connections. With 3,000 multicast groups available, the number of possible groupings approaches ($3{,}000^{22{,}000}$/3,000!). Clearly an optimized, heuristic approach offers potential for improved performance. By virtue of regions, the destination of attribute updates are known before they're sent. Consider the region layout illustrated in Figure 2-5. From this figure it is clear that federate f_1, which owns region U1, will send updates of attributes associated with U1 to federates $f_3$and f_2 because they own subscription regions S2 and S4, respectively.

This information is recast into a connection graph as illustrated in Figure 2-6. Federate f_1 sends the attribute update(s) represented by connection path c_1 to federates f_2 and f_3. Federate f_3 sends the attribute update(s) represented by connection path c_2 to

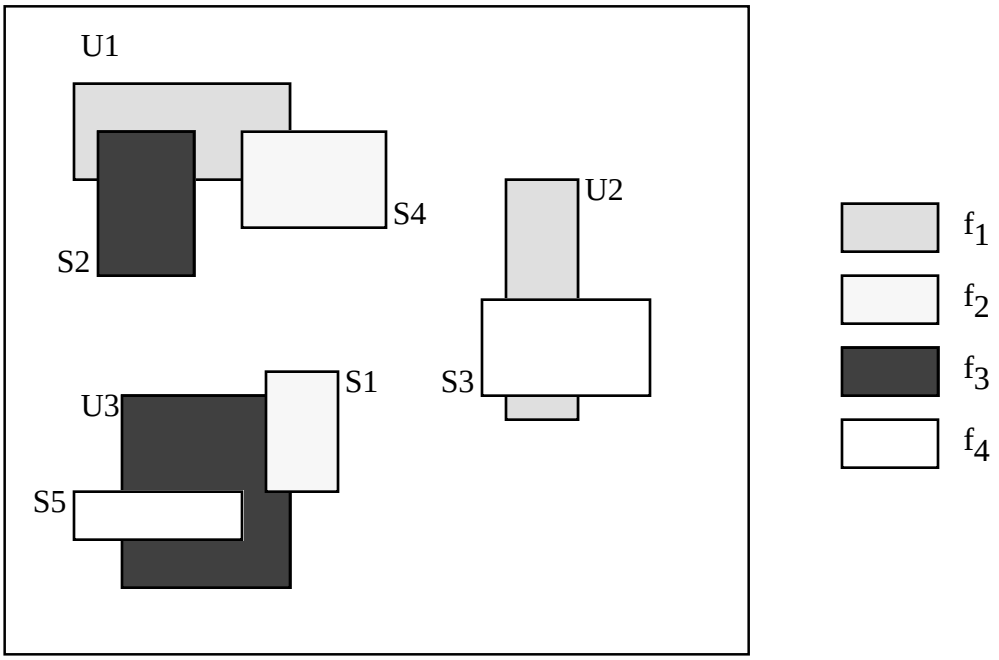

Figure 2-5. Example Region Layout

federates f_2 and f_4. Federate f_1 also sends c_3 to f_4. Alternatively, the graph can be described by the connection set, C, represented by this graph is {<f_1, c_1, w_1, (f_2, f_3)>, <f_1, c_3, w_3, (f_4)>, <f_3, c_2, w_2, (f_2, f_4)>}. Note that a connection path is not the same as a communication. A connection path doesn't represent a single message, but all updates of some set of object attributes. A communication is a single update or message. A connection always has a single sender federate and one or more receiver federates. It also has a weight which represents the frequency with which it is updated. So, connection c_1 is sent with frequency w_1. The goal is to put connections into multicast groups such that message delivery time meets a specified limit.

2.2.2 Total Possible Combinations

Assume that all connections not assigned to a multicast group are sent point-to-point. With global information, the exhaustive search solution to this problem has

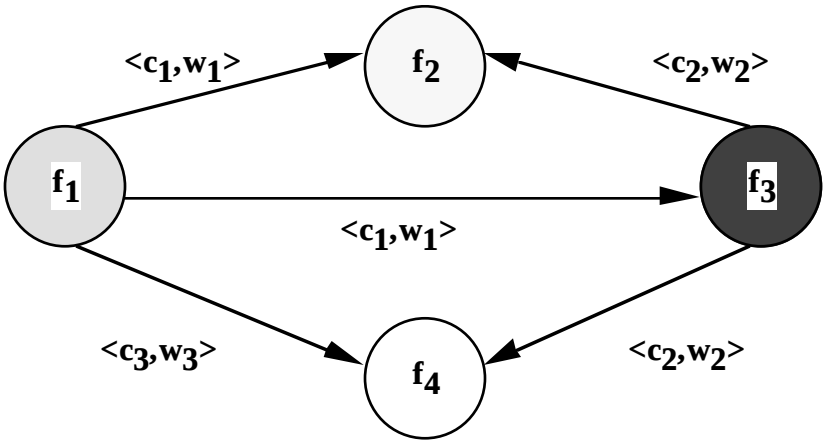

Figure 2-6. Example Connection Graph

complexity O((m+1)S(n,m+1)) for n connections and m multicast groups, where S(n, m) is the Stirling number of the second kind as given in Equation 2-1. The Stirling number of the second kind calculates the number of unique partitions of n distinct items into m indistinguishable groups [Bender91].

$$S(n, m) = \frac{1}{m!} \bullet \sum_{i=0}^{m} (-1)^i \binom{m}{i} (m - i)^n$$

Equation 2-1. Stirling Number of the Second Kind

To be precise, the number of possible combinations is given in Equation 2-2.

$$TPC(n, m) = \sum_{i=1}^{m} \left(S(n, i) + (i + 1)S(n, i + 1) \right)$$

Equation 2-2. Total Possible Combinations

This is the case for two reasons. First, for any given set of connections and groups, the combination doesn't have to use all the multicast groups available. This results in taking the sum over all possible number of multicast groups, m. Second, any connection

may be “unassigned,” i.e. it may be left point-to-point. In addition to S(n, m), which assigns all connections to an actual multicast group, TPC must add the possibility of another “group” for point-to-point, but it must be substituted for each real group since it’s special. To see the special nature of the “unassigned group,” consider the following example. Under the Stirling number, the assignment of four connections (c_1, c_2, c_3, c_4) to two multicast groups (g_1, g_2), ($c_1 \in g_1$, $c_2 \in g_1$, $c_3 \in g_2$, $c_4 \in g_2$) is equivalent to ($c_1 \in g_2$, $c_2 \in g_2$, $c_3 \in g_1$, $c_4 \in g_1$) because in both cases c_1 and c_2 are grouped together as are c_3 and c_4. If group g_1 represents point-to-point, c_1 and c_2 going point-to-point while c_3 and c_4 are grouped together is different than c_1 and c_2 being grouped together while c_3 and c_4 are sent point-to-point.

2.2.3 Formal Problem Statement

There exists a finite set F of federates, $f_1 \ldots f_g$, represented as vertices in a graph. There exist:

- a finite set C of connections, $c_1 \ldots c_n$, each represented as a collection of directed edges with the same source federate
- for each $c \in C$ a weight w(c)
- for each c a set of k(c) receivers, r(c,1),...,r(c,k) $\in$ F
- for each c a sender s(c) $\in$ F.

For each pair ($f_i \in F$, $f_j \in F$) there exists an integer $t_p(f_i, f_j)$ representing the maximum network propagation time for a message between federates f_i and f_j.

There exist positive integers m and t_{max} representing respectively the number of available multicast groups and the maximum tolerable message delivery delay.

There exist positive integers t_s and t_r representing respectively the time to traverse the message sender's protocol stack and the time to traverse the message receiver's protocol stack, the latter excluding queueing time at the receiver.

Is there an assignment, A, of all $c \in C$ into at most m groups such that Equation 2-3 holds?

$$\forall c \; \forall \{i: 1 \leq i \leq k(c)\} \; t(c, i) \equiv t_{ds}(c, A) + t_p(s(c), r(c, i)) + t_q(r(c, i), A) + t_r \leq t_{max}$$

Equation 2-3. Message Delivery Bound

$t_{ds}(c, A)$ represents the time a message is delayed at the sender and satisfies the condition in Equation 2-4 if c is not in a group, and the condition in Equation 2-5 if c is in a group. Regardless of the backup in the sending queue, all messages must be delivered eventually, i.e. $t_{ds}(c, A)$ may not be infinite.

$$t_s \leq t_{ds}(c, A) \leq k(c) \bullet t_s$$

Equation 2-4. t_{ds} for c Not in a Group

$$t_{ds}(c, A) = t_s$$

Equation 2-5. t_{ds} for c in a Group

$t_q(f, A)$ represents the queueing delay at each receiver and satisfies the inequality given in Equation 2-6.

$$0 \le t(f, A) \le t \bullet \sum_{\{c':f \in \text{ receiver set of } c'\}} w(c')$$

Equation 2-6. t_q

2.2.3.1 Relationship to Clique-Covering

This problem is similar to the clique-covering problem. Intuitively, if a clique exists in the connection pattern between a set of nodes for a set of connections, assigning a multicast group to these nodes for these connections results in optimal routing. In practice, however, it's not reasonable to expect to be fortunate enough to have many cliques in the connection graph.

2.2.3.2 Relationship to the Knapsack Problem

The multicast grouping problem is related to the knapsack problem [Papa94]: "We must select some among a set of *n* items. Item *i* has value v_i, and weight w_i, both positive integers. There is a limit *W* to the total weight of the items we can pick. We wish to pick certain items (without repetitions) to maximize the total value, subject to the constraint that the total weight is at most *G* [sic, should be *W*]."

The n connections are all unassigned connections in the system. The value of each connection is the reduction in average message delivery time due to parallel sending.

The weight of each connection is simply the connection weight. The goal is to maximize the reduction in average send time without exceeding the maximum input weight that any of the participant nodes can accept. This is where the multicast grouping problem differs from the knapsack problem. Simply adding the weight of the new connection to the total weight does not capture the effect on incoming weight to all participating nodes. Consider the example connection graph in Figure 2-7.

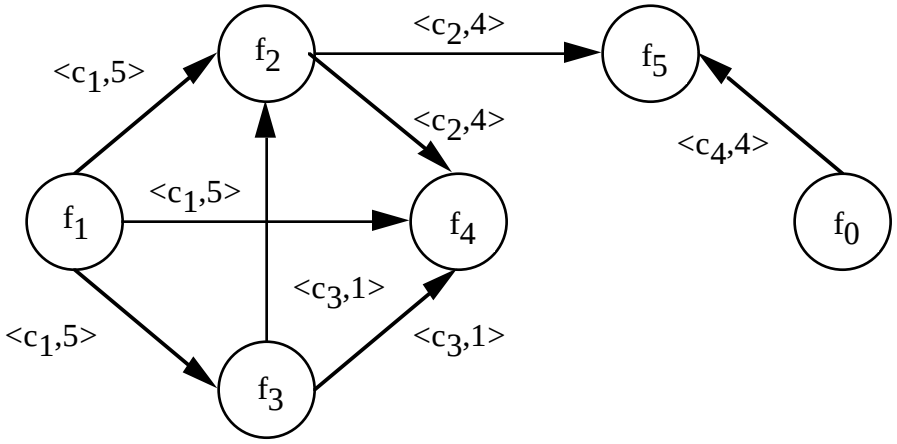

Figure 2-7. Connection Graph Exceeding Maximum Weight

If the maximum allowable weight for the graph is 10, putting connections c_1 through c_3 into a single group appears to meet the weight requirement. However, f_5 has an additional incoming connection, c_4, which exceeds f_5's limit by 4.

2.2.3.3 Application of Known Optimization Techniques

Relating the multicast grouping problem to the knapsack problem and demonstrating their differences illustrates one of reasons that this problem is not amenable to solution by optimization techniques such as backtracking. This problem lacks *self-*

reducibility [Lewis98], i.e. better (or worse) solutions to subproblems do not translate to predictably better (or worse) solutions to the larger problem. This is because the value of adding a connection to a multicast group is constant, while the cost of additions, which is dependent on previous additions, changes every time an assignment is made, i.e. the problem space changes while the problem is being solved. Additionally, the incoming weight only partially captures the queuing effects at the federates. Before any grouping, f_5 has two incoming connections with weight 4, c_2 and c_4. If messages from the two connections arrive interleaved at intervals at least as large as the message delivery time, f_5 will experience no queuing delays. If the messages always arrive at the same time, f_5 will experience queuing delays which affect average message delivery time. Finally, the results of grouping are highly dependent on the number of groups available. This is immediately obvious considering that point-to-point is the same as having zero groups available, while having n multicast groups for n connections can deliver optimal routing.

These effects come into play when branch-and-bound is applied to the example illustrated in Figure 2-8. The connection set for this graph is $<f_0, c_1, 50, (f_1, f_2)>$, $<f_0, c_2, 50, (f_3, f_4)>$, $<f_1, c_3, 50, (f_0, f_4)>$, $<f_5, c_4, 40, (f_6, f_7)>$}. Simulating message delivery of this connection set when all the connections are sent point-to-point results in an average message delivery time of 35.6174 ms, when t_s, t_r, and t_p are all 10 ms. Adding the first connection to a group reduces this time to 34.4695; and adding the second connection to the same group decreases the time again to 33.7226. This result is somewhat unexpected since the addition of the second connection significantly increases the number of irrelevant messages delivered to f_1, f_2, f_3, and f_4. The time is decreased in this particular case

because the messages from c_1 and c_2 happened to arrive interleaved. The positive impact of putting both connections into the same group to reduce sending time was realized without the negative impact of queueing effects. Putting the third connection into the same group finally has a negligible negative impact, 33.7244, because $f_2$and f_3 are finally getting three times as many messages as they should. However, adding the final connection to another group send the delivery time down again to 32.6708.

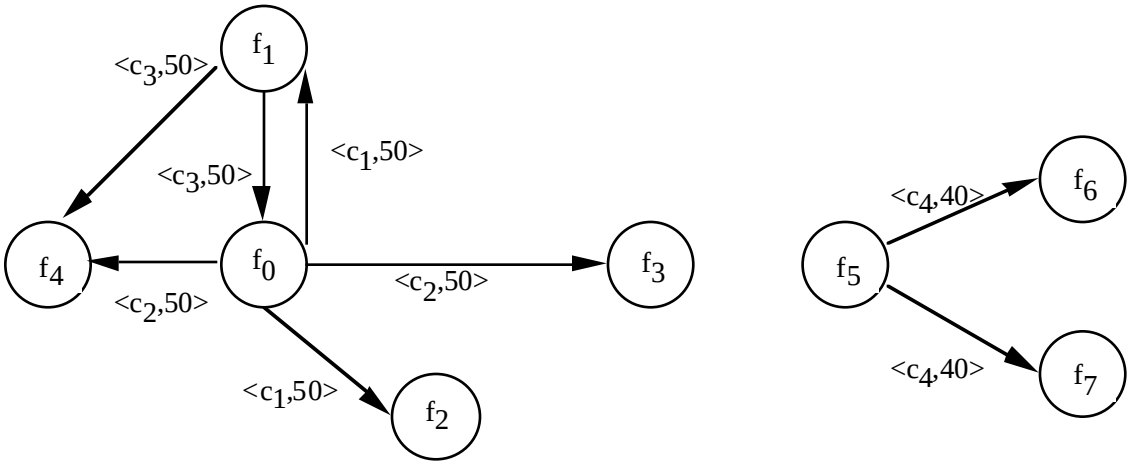

Figure 2-8. Connection Graph Demonstrating Lack of Self-Reducability

2.2.4 Cost Function

The multicast grouping algorithms use message delivery time as their cost measure. The goal is to group the n connections, c_1, ... c_n, into no more than m multicast groups, g_1, ... g_m, such that no communication arrives at its receiver in greater than the maximum tolerable latency, t_{max}, if physically possible. If two algorithms can achieve this requirement, one algorithm is considered better than the other if its average message delivery time is lower. The parameters to the algorithm are listed below.

- Number of available multicast groups (m)
- Maximum tolerable latency (t_{max})
- Time to send (t_s) - assume that t_s is roughly the same for all federates.
- Time to receive (t_r) - assume that the time to discard an irrelevant message is the same as the time to receive a relevant one; also that t_r is roughly the same for all federates.
- Time to propagate message through the network (t_p,(f_i, f_j)) - measured between federate f_i and f_j, the publisher and subscriber of the connection, respectively.

The algorithms begin by assuming point-to-point communication for all messages and falls back to this position when the network and scenario make multicast grouping impossible, i.e. when $t_{ds} + t_p + t_r + t_q > t_{max}$ for some connections.

The decision to add a connection, c, to a multicast group is based on the expected impact on t_{max} of all connections and federates already in the group. If connection c is sent to k receivers point-to-point, t(c), the time from the beginning of c's sending to the end of the last receiver's receipt is bounded by:

$$\forall\{j: 1 \le j \le k(c)\}\ k \bullet t_s + \max(t_p(s, r(c, j))) + \max(t_q(r(c, j))) + t_r$$

Equation 2-7. Time Bound for k Point-to-point Communications

This bound is based on the worst case assumption that the k^{th} communication has the longest t_p and the longest t_q.

Figure 2-9 illustrates this for a connection with three communications, where the

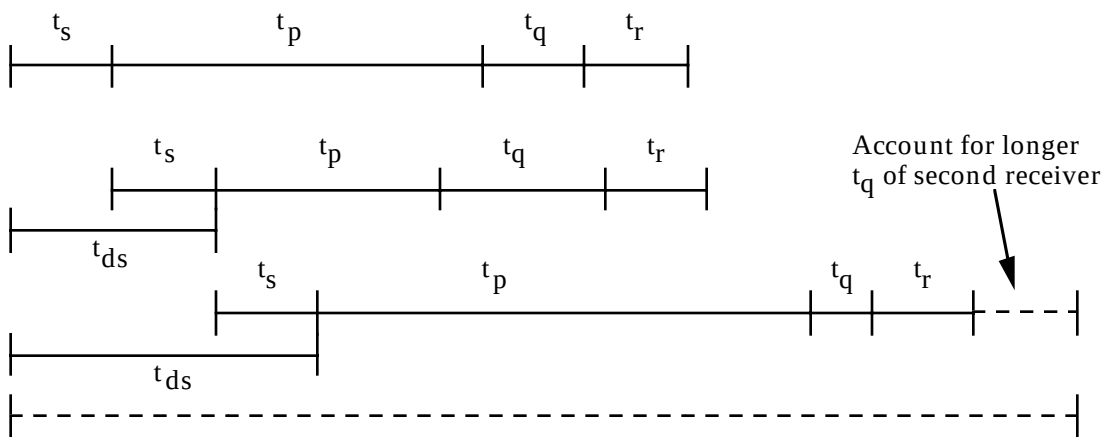

Figure 2-9. Illustration of Time Bound for k Point-to-point Communications

second communication has the longest t_q and the third communication has the longest t_p.

Assume t_S and t_r are fixed, uniform, and roughly equal. Time to propagate the update, $t_p(f_i,f_j)$ is assumed to be fixed, but different and measurable between source and destination, federates i and j. Time in the queue, t_q, varies depending on the number and frequency of messages received.

If connection c's sender and all its receivers are assigned to a multicast group, the last receiver's receipt time is bounded by:

$$\forall\{j: 1 \le j \le k(c)\}\ t_s + \max(t_p(s, r(c, j))) + \max(t_q'(r(c, j))) + t_r$$

Equation 2-8. Time Bound for k Multicast Communications

This is based on the same worst case assumptions as Equation 2-7. Figure 2-10 illustrates this time bound for the example in Figure 2-9.

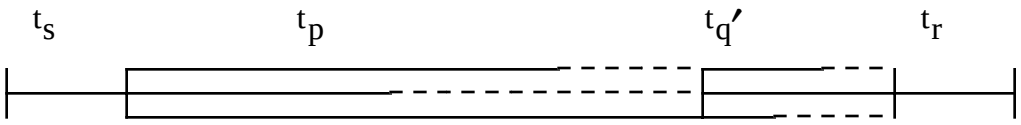

Figure 2-10. Illustration of Time Bound for k Multicast Communications

Assuming the use of multicast routers which optimize routing in a tree structure, values of t_p should be the same as for point-to-point[4]. Individual receivers are unaware of the routing, so t_r is also unchanged. Without multicast the average serial send time for the k copies is as given in Equation 2-9.

$$\text{Average } t_{ds} = \frac{t_s}{k} \bullet \sum_{i=1}^{k} 1 = \frac{k \bullet (k+1)}{2} \bullet \frac{t_s}{k} = \frac{t_s \bullet (k+1)}{2}$$

Equation 2-9. Average Serial Send Delay Time

The use of multicast improves latency for a message by reducing the average serial send time for the k copies as given in Equation 2-9 to t_s. In fact, t_{ds} is exactly t_s. Furthermore, this improvement is achieved each time a message is sent for a connection.

4. If the underlying multicast protocol is not source-based, e.g. core-based tree [Ballardie98], this assumption must be relaxed.

Taking into account the frequency with which a connection's messages are sent, by sending a connection with multicast, the average message sending delay is reduced by

$$\text{Improvement in } t_{ds} = \left[\frac{t_s \bullet (k+1)}{2} - t_s\right] \bullet w = \left[\frac{t_s \bullet (k-1)}{2}\right] \bullet w$$

Equation 2-10. Multicast Improvement in t_{ds}

2.3 Estimating t_q and t_{ds}

However, adding connections to an existing multicast group may cause extraneous communication at some receivers, increasing t_q for some valid communications. For example, putting all the connections in Figure 2-6 would result in the following extraneous communications:

- c_3 to f_2 and f_3
- c_1 to f_4
- c_2 to f_1

When a new connection is added to a group, t_q for the last message received at any time is t_r • (the new incoming weight - 1). Based on the assumption that, on average, half the messages in any time frame arrive before the one of interest and the other half arrive after it.

$$\text{Average } t_q = \frac{t_r \bullet (\text{incoming weight} - 1)}{2}$$

Equation 2-11. Average t_q with Grouping

Consider the average t_q for the connection graph in Figure 2-9 which results from this grouping. Added receivers are marked with bold arrows.

t_q at each of the federates before and after putting all the connections into a single group is given in Table 2-1. t_q is only given if the federate needs to receive relevant data, e.g. t_q is not applicable for f_1, even after grouping, since f_1 never has any relevant data to be delayed.

Table 2-1: Average t_q for Example Figure 2-6 and Figure 2-11

	Before Grouping	After Grouping	Change
f_1	N/A	N/A	N/A
f_2	$(w_1+w_2-1)\bullet t_r/2$	$(w_1+w_2+w_3-1)\bullet t_r/2$	$w_3.t_r/2$
f_3	$(w_1-1)\bullet t_r/2$	$(w_1+w_3-1)\bullet t_r/2$	$w_3.t_r/2$
f_4	$(w_2+w_3-1)\bullet t_r/2$	$(w_1+w_2+w_3-1)\bullet t_r/2$	$w_1.t_r/2$

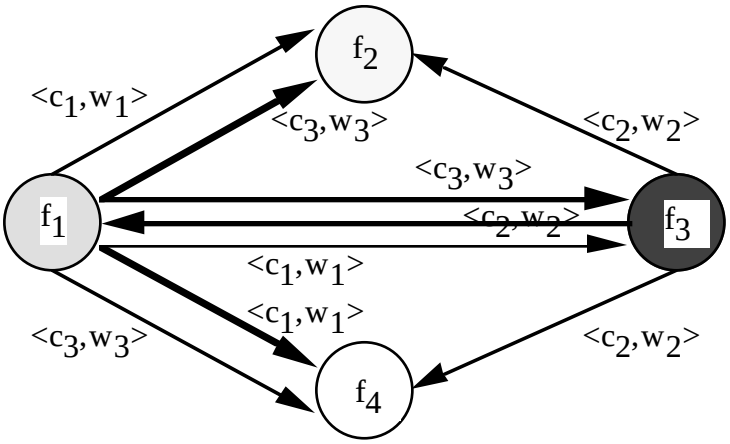

Figure 2-11. Connection Graph for Estimating t_q and t_{ds}

It is possible to estimate the average case sending delay, t_{ds}, for each connection more accurately because its value is only dependent on the state of the connection itself. If a connection is sent point-to-point it's assumed the order in which the individual messages are sent is non-deterministic. t_{ds} for each of the connections in Figure 2-6 and Figure 2-11, before and after putting all the connections into a single group, is given in Table 2-2.[5]

Table 2-2: Average t_{ds} for Figure 2-6 and Figure 2-11

	Before Grouping	After Grouping	Change
c_1	$w_1 \cdot t_s \cdot 3/2$	$w_1 \cdot t_s$	$-w_1 \cdot t_s/2$
c_2	$w_2 \cdot t_s \cdot 3/2$	$w_2 \cdot t_s$	$-w_2 \cdot t_s/2$
c_3	$w_3 \cdot t_s$	$w_3 \cdot t_s$	0^5

Since $t_p(f_i, f_j)$ and t_r are fixed, the average message delivery time for each connection at each federate can be estimated. Table 2-3 shows the average delivery time without $t_p(f_i, f_j) + t_r$ before the connections are all put in a single group.

Table 2-3: Average Message Delivery Time for Figure 2-6

	c_1	c_2	c_3
f_1	N/A	N/A	N/A
f_2	$w_1 \bullet t_s \bullet 3/2 + (w_1+w_2-1) \bullet t_r/2$	$w_2 \bullet t_s \bullet 3/2 + (w_1+w_2-1) \bullet t_r/2$	N/A
f_3	$w_1 \bullet t_s \bullet 3/2 + (w_1-1) \bullet t_r/2$	N/A	N/A
f_4	N/A	$w_2 \bullet t_s \bullet 3/2 + (w_2+w_3-1) \bullet t_r/2$	$w_3 \bullet t_s + (w_2+w_3-1) \bullet t_r/2$

5. The fact that t_{ds} is not improved in this example illustrates why a single-receiver connection should never be put in a multicast group.

Table 2-3 shows the average delivery time without $t_p(f_i, f_j) + t_r$ after the connections are all put in a single group. Once again this table looks at relevant messages, but takes into account the queueing effects of the irrelevant messages.

Table 2-4: Average Message Delivery Time for Figure 2-11

	c_1	c_2	c_3
f_1	N/A	N/A	N/A
f_2	$w_1 \cdot t_s + (w_1+w_2+w_3-1) \bullet t_r/2$	$w_2 \cdot t_s + (w_1+w_2+w_3-1) \bullet t_r/2$	N/A
f_3	$w_1 \cdot t_s + (w_1+w_3-1) \bullet t_r/2$	N/A	N/A
f_4	N/A	$w_2 \cdot t_s + (w_1+w_2+w_3-1) \bullet t_r/2$	$w_3 \cdot t_s + (w_1+w_2+w_3-1) \bullet t_r/2$

The average message delivery time of c_1 to f_2 can be improved if Equation 2-12 holds.

$$\frac{(3 \bullet w_1 \bullet t_s) + (w_1 + w_2 - 1) \bullet t_t}{2} > (w_1 \bullet t_s) + \frac{(w_1 + w_2 + w_3 - 1) \bullet t_t}{2}$$

or

$$w_1 \bullet t_s > w_3 \bullet t_t$$

Equation 2-12. Improving Delivery of c_1 to f_2

Based on the assumption that $t_s = t_r$, this implies that the average delivery time of c_1 to f_2 can only be improved if $w_1 > w_3$.

The general case for adding a connection to a multicast group can be derived from the preceding equations and examples. Assume a multicast group, g, has a set of

connections assigned to it and that the sum of their weights is w_g. Not all of the members of g will receive w_g from the group[6] in the case that they both send and receive to the group. This is because the weight of their own connections, which they don't receive, are included in w_g. Now consider what happens to the average message delivery time across the entire system when a new connection, c, with weight w_c is added to the group. The average message delivery time is improved as described in Equation 2-10. This improvement is calculated trivially at the sender. The negative impact on t_q is more complex to calculate. Each receiver of c who is not already in g will receive the full brunt of the connections in g, w_g. Each member of g who is not a receiver of c will receive the additional weight of c, w_c. As the implemented algorithms in Chapter 4 demonstrate, calculating the negative impact is somewhat harder in the offline case and much harder in the online, distributed case. However, the penalty for only accounting for the improvement in sending time when generating groups exceeds the benefits of a simple implementation.

These derivations illuminate a much larger point. This research addresses *potential* performance improvements. There are some circumstances under which multicast cannot improve over the performance of point-to-point. So, developing an algorithm or algorithms which perform the grouping is insufficient. The circumstances under which the algorithm can improve performance must be delineated and the implementation must be able to detect these situations and respond accordingly. Section 3.2 describes a taxonomy for characterizing scenarios which is used for making this selection.

---

6. This is independent of weight that members receive from other point-to-point connections and multicast connections in other groups.

CHAPTER 3: Solution Evaluation

3.1 Performance Measures

The primary goal of IM services is to reduce the amount of data received by simulations. But they should not do so at the cost of excessive overhead, i.e. they should not use more CPU cycles and/or delay message delivery more than the federates would if they received all messages and performed final filtering themselves. It goes without saying that IM services should also deliver the correct data. So, IM services should be efficient, quick, and correct.

DDM's primary goal is to reduce the amount of data received by federates, but this requirement is derived from a higher requirement to enable federates to perform their jobs of simulating their models in a timely manner. To do so, federates must receive data in a timely manner and they must have enough CPU cycles available to process the data when they receive it. Balancing these two requirements is central to the performance of a DDM implementation.

Suppose a federate is only using class-based filtering (Declaration Management services in the HLA). The infrastructure may not have to use as many CPU cycles because it doesn't have to manage regions, but the receiving federate will have to expend additional cycles to discard irrelevant data[1]. This situation may also impact the timeliness of the

1. This assumes an RTI architecture roughly like the current DMSO architecture described in [DMSO98] in which RTI components are hosted on the same processors as the federates.

federate's receipt of desired data in two ways. First, the data may be delayed in the network because the network is flooded with data which will only be discarded by the receiver. Second, the federate has to expend time as well as CPU cycles to discard unwanted data when it's received, delaying the federate's processing of the desired data. So, two measures are of interest:

- Efficiency = CPU cycles expended per desired update; an efficient DDM implementation will use fewer CPU cycles across the federation than the federates would discarding unwanted data.
- Latency = time to receive wanted data; to be fair in evaluating DDM implementations this is the time between the sending federate calling the RTI and the receiving federate determining that the data is of interest, i.e. including the time it would take the federate to discard enough unwanted data to get to the desired data.

CPU cycles are measured by running the simulation in fixed length time slices. The RTI is allowed to run at the beginning of each cycle until it processes all available tasks at which time it returns control to the simulation. The percentage of time remaining in the cycle is for the simulation to use the CPU. See [Cohen98a] for a detailed description of this measurement method for CPU usage. Notice that this measurement method captures both the overhead time for maintaining and updating DDM state and the time wasted by the infrastructure in processing incoming messages, including extraneous ones. With instrumentation it would be possible to determine exactly how many messages are delivered, however such instrumentation is not part of the HLA standard and so not generally available in RTIs. While this would be another valuable measure, the overall

goal is to reduce time taken away from the simulation by the RTI. The latency and efficiency measures capture that goal.

3.2 Characterizing Scenarios

Since this research is targeted at optimizing interest management, a characterization of scenarios should not focus on how they affect interest managers' ability to route the data itself, i.e. precision, but on their effect on the performance of interest managers. The following taxonomy characterizes scenarios:

- Number of regions (r) – more regions will require more memory to store within RTI components and more CPU cycles to search when regions are updated.
- Rate of region change (Δr) – frequent region changes will require more CPU cycles to recalculate intersections. However, an intelligent implementation will detect when region changes don't result in intersection changes, and hence don't generate connectivity changes.
- Number of intersections (i) – since all region intersections for a given object class and routing space must be rechecked when either an update or subscription region changes, a large number of intersections will require more CPU cycles to recalculate.
- Rate of region intersection change (Δi) – if the interest management system's time to establish connectivity is high, then rapid changes in the intersections will not be affected immediately, resulting in a larger percentage of incorrect messages.

This taxonomy leads to the following hypothesis with respect to the performance of multicast grouping implementation:

Hypothesis 1: Multicast grouping algorithms may produce performance benefits when Δi and t are low[2] regardless of the value of Δr.

If users are able to roughly determine the values of r, Δr, i, and Δi for their scenarios they can determine if multicast grouping can optimize performance for their scenarios.

3.3 Benchmark Algorithm

Having established the characteristics the benchmark must manifest, it is now possible to describe the algorithm for the benchmark. The size, shape, and pattern of regions in this algorithm are intentionally artificial. The goal is to be able to control accurately the DDM-specific characteristics of the scenario, not to represent a realistic simulation scenario. See [Cohen98, Cohen98a] for DDM benchmarking which more closely approximates a realistic scenario. Figure 3-1 illustrates a sample region layout for a test with 5 federates, 12 regions, and 86 intersections.

Subscription regions are laid out in non-overlapping horizontal bands. Update regions are laid out in vertical bands. Subscription regions extend the entire width of the x dimension from RTI_MIN to RTI_MAX, and their extents never change. This is the same usage scenario labeled as (e) in [Cohen98a]. Update regions are responsible for

2. This is t for a specified number of multicast groups.

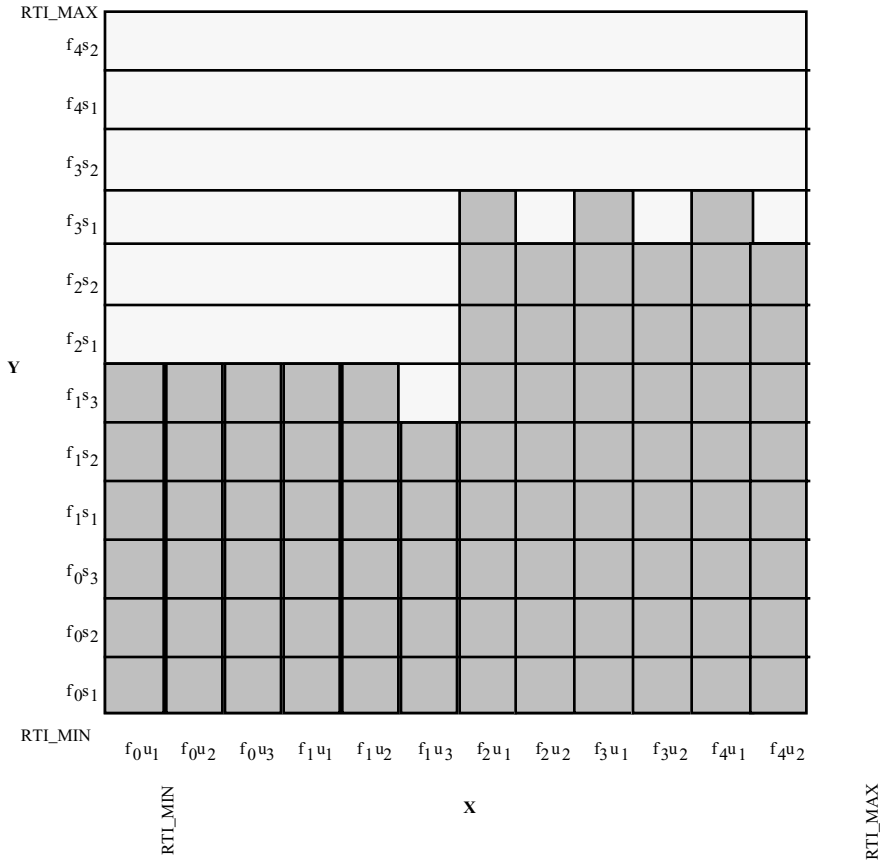

Figure 3-1. Initial Region Layout

calculating and modifying their extents to create and manage the number of intersections requested[3]. Within a federate, they start from RTI_MIN in the y dimension and extend as far toward RTI_MAX as necessary to create the number of intersections required.

Regions are uniformly assigned to federates to the extent possible given the number of regions and federates. In most cases the algorithm is always accurate, but assigning intersection changes may be off by one or two depending on the divisibility of the user's parameters. Notice in Figure 3-1 that specifying 12 regions means that federates 0 and 1 each have three regions while the remaining three federates each have

3. The algorithm doesn't have to worry about whether or not update regions overlap each other because the update regions control all region modifications leading to intersections.

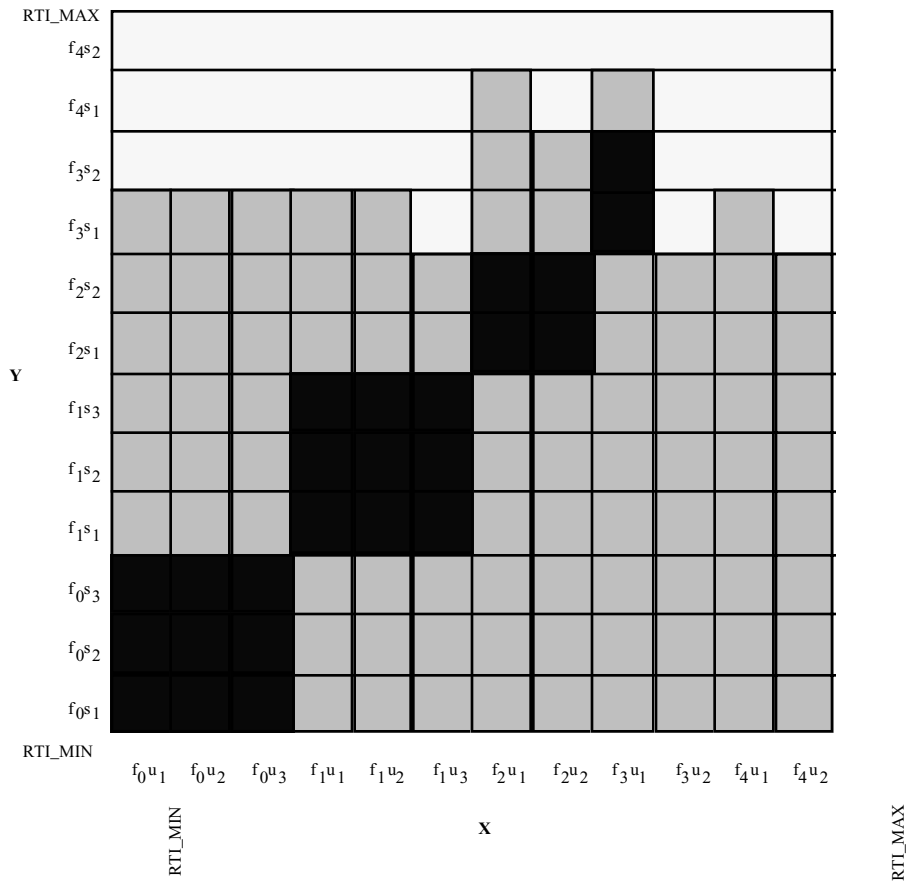

Figure 3-2. Accounting for Local Subscription Regions

two. As a result, when the 86 intersections are allocated, all federates get 17 intersections except for federate 0 which gets 18, but federates 2 through 4 have to allocate each of their 17 intersections across only two update regions rather than three.

There's only one small problem with the layout as described. "Intersections" between your own subscription and update regions don't "count", e.g. even though f_0u_1 and f_0s_1 overlap in the figure, that's not an intersection from the perspective of DDM. So, the algorithm needs to adjust the update region lengths to account for overlaps with their own federate's subscription region. Figure 3-2 illustrates this extension for the example.

The dark blocks mark the bands of local subscription regions that the update regions must extend around.

So far the algorithm only accounts for the initial placement of regions. It must also account for region modifications and intersection changes: Δr and Δi. Δr is specified in modifications per minute. Δi is specified as a percentage of i. Intersection changes are made by sliding the update regions “up” and “down” across the subscription regions, always accounting for the black hole of local subscription regions. The number of changes is controlled by how far the region slides. Observe that you can’t have a region intersection change without modifying a region, so region modifications happen automatically with intersection changes.

Notice that each federate can independently calculate region location, size, and movement based on knowing how many other federates are participating and how many total regions and intersections there are. In fact within each federate the modification of each update region is independent of the modification of every other one because once each update region “knows” how many how often it must change, and how many intersections and intersection changes it’s responsible for maintaining, it performs these actions against the static subscription regions without interaction with the other update regions.

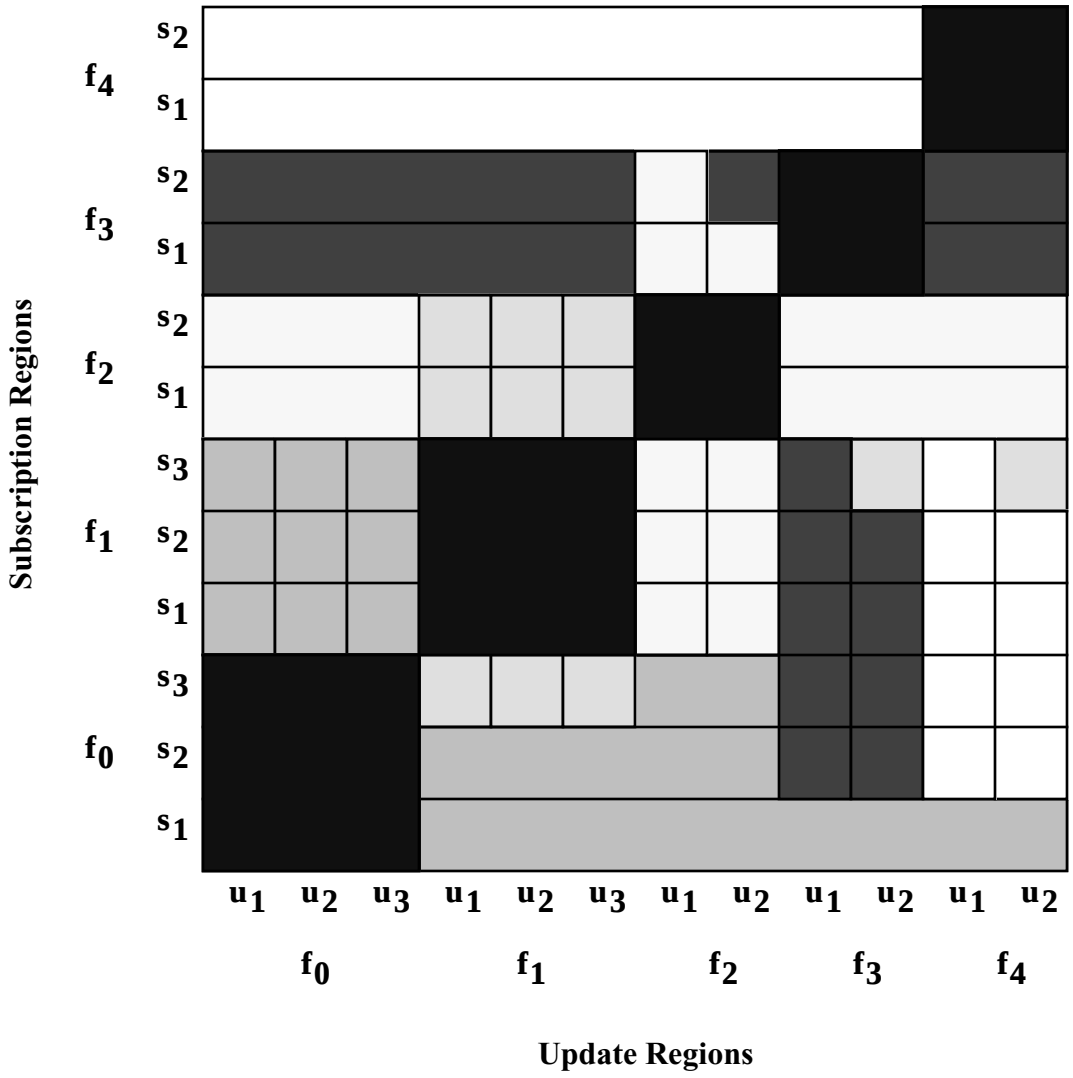

Figure 3-3. Sample Region Layout from Benchmark Algorithm

A more illustrative example is given in Figure 3-3 and Figure 3-4 . Figure 3-3 shows one possible layout generated by the benchmark algorithm. The resulting connection graph is shown in Figure 3-4.

3.3.1 Inputs to the Benchmark Algorithm

The benchmark takes the following inputs:

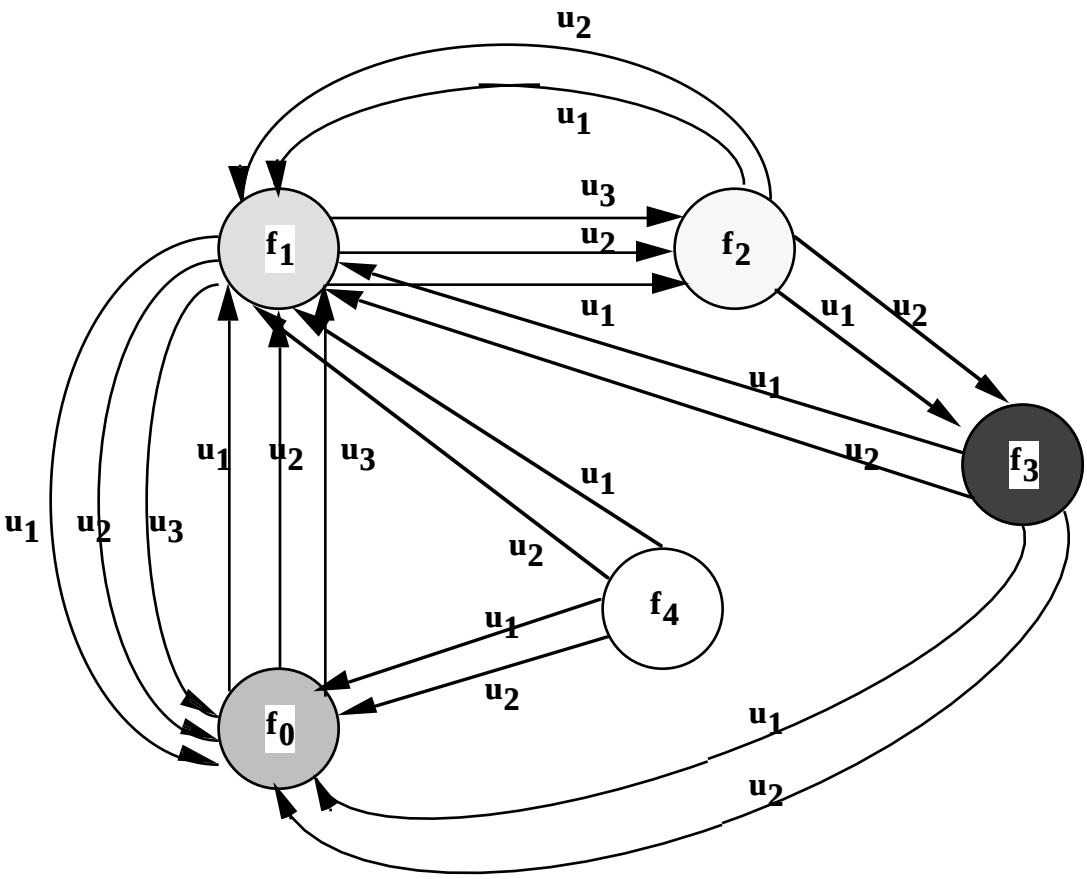

Figure 3-4. Resulting Connection Graph

- Number of federates – there is no specified limit on the number of federates which can participate.
- Federate number - 0 ≤ federate number < number of federates.
- Total regions – there is no specified limit on the number of regions which can be created, but there is a practical limit based on the RTI range for a dimension because it must be subdivided to form disjoint subscription regions. The benchmark actually creates twice this many regions: one set of update regions this size and one set of subscription regions this size. Regions are uniformly assigned to federates plus or minus one based on the ratio of regions to federates.

- Δr – region modifications are given in modifications per minute. The default cycle time for modifications is one minute, but if more region modifications are specified per minute than can be accommodated by the number of regions, the cycle time is divided until it will accommodate them.
- Total intersections – total intersections _ total regions * (total regions – local regions), i.e. each update region can intersect with every subscription region except those assigned to the local federate.
- Δi – region intersection changes are specified as a percentage of i. Δi cannot always be exactly the number requested by the user because of the relationship between Δi and the number of intersections. If the number of intersections is to remain constant, intersection changes must occur in pairs; sliding an update region up or down by one subscription region results in one new intersection and the deletion of an existing one. Once the algorithm uniformly assigns intersection changes to the federates and the federates assign them to update regions, the federates attempt to even out assignments to update regions, adjusting adjacent regions up and down by one. However, the number assigned to the federate in the first place could have been odd.
- Number of minutes to run
- Number of seconds to allow the federation to run before beginning measurement
- Number of milliseconds to allocate for a single measurement loop.
- DM/DDM switch – this compiler switch is provided to enable measurement of the difference between using just DM and using DDM as described in section Performance Measures. Even when the algorithm is just using DM, it performs all the region modification calculations because the federate would still have to use this

information to identify unwanted data. These calculations also represent the normal federate activity of moving objects with which the update regions are associated.

CHAPTER 4: Implemented Solutions and Contributions

4.1 MESSENGERS Overview

MESSENGERS is a distributed programming system based on the principles of mobile agents (individually referred to as Messengers), which carry their own behavior in the form of programs. This enables them to navigate freely in the underlying computational network, communicate with one another, and invoke pre-compiled node-resident C functions. Thus MESSENGERS represents a general coordination paradigm for distributed applications development. Its main advantages are dynamic composability of applications and the ability to operate in unknown or dynamically changing network topologies. Dynamic composability enables the development of arbitrarily extensible open-ended applications, including adding entirely new behaviors at run time. MESSENGERS provides the capability to structure the application as a collection of autonomous mobile "intelligent agents." These can be created at run time, each having its own specific behavior that permits it to pursue its specific goal. The complex overall functionality of the application thus emerges from the local interactions of the individual agents. Hence, the paradigm offers a flexible way of creating, testing, and experimenting with different designs interactively and incrementally. The MESSENGERS paradigm is a good match for the interest management problem at hand because interest expressions can be described and implemented as "intelligent agents" that cooperate in achieving the overall goal. They are also inherently capable of migration among physical nodes, permitting performance to be adjusted at runtime. In addition to the physical nodes and network, Messengers may create virtual networks including multiple virtual nodes on

individual physical nodes, and virtual links between physical and/or virtual nodes. Virtual nodes and links may be labeled, allowing the creation of virtual networks whose structures have semantic content. Messengers may also replicate themselves.

A MESSENGERS environment has been developed and is currently operational on a local area network of Sun workstations running Solaris. Its basic capabilities have been tested using a number of different applications [Bic96, Bic95].

4.2 Baseline Prototype

The baseline prototype has an architecture similar to the HLA Run Time Infrastructure (RTI) 1.3 [VanHook98]. The RTI uses the intersections of update and subscription regions in the same routing space for the same object class to determine that it should establish connectivity between the publishing and subscribing federates. The DDM process is bootstrapped by having publishing federates “subscribe” for relevant subscriptions, i.e. the DDM services only forward subscription information for a given object class to federates which are publishing the same object class, rather than to all federates. If a federate is not publishing an object class, it will never need to consider subscriptions for that class. Routing decisions are made at the publisher’s component of the RTI based on detected intersections with relevant subscription regions. Intersection is established based on four criteria:

- Routing space
- Object class

- Region extents
- Attributes

The baseline prototype is implemented with three classes of Messenger mobile agents:

1. subscription regions,
2. update regions,
3. routing initialization agents.

MESSENGERS has the capability to specify virtual networks on top of the physical network. Multiple virtual nodes may be physically hosted on a single physical node with links between them that do not necessarily correspond to the physical network lines. In the baseline prototype, virtual nodes and links labeled with the routing space and object class are created when a federate declares its intent to publish the attributes of the object class through its routing space. This allows prefiltering of subscriptions on the first two intersection criteria, i.e. subscription region Messengers only migrate to virtual nodes where they know they will encounter update regions representing objects of a class of interest in the correct routing space. The intersection calculation problem is reduced to region extent and attribute intersection calculation.

Figure 4-1 through Figure 4-2 show a simplified example of such a virtual network as it is established. Each federate is linked with the object code for a local RTI component, its interface to the RTI. When a federation is created by a federate joining it,

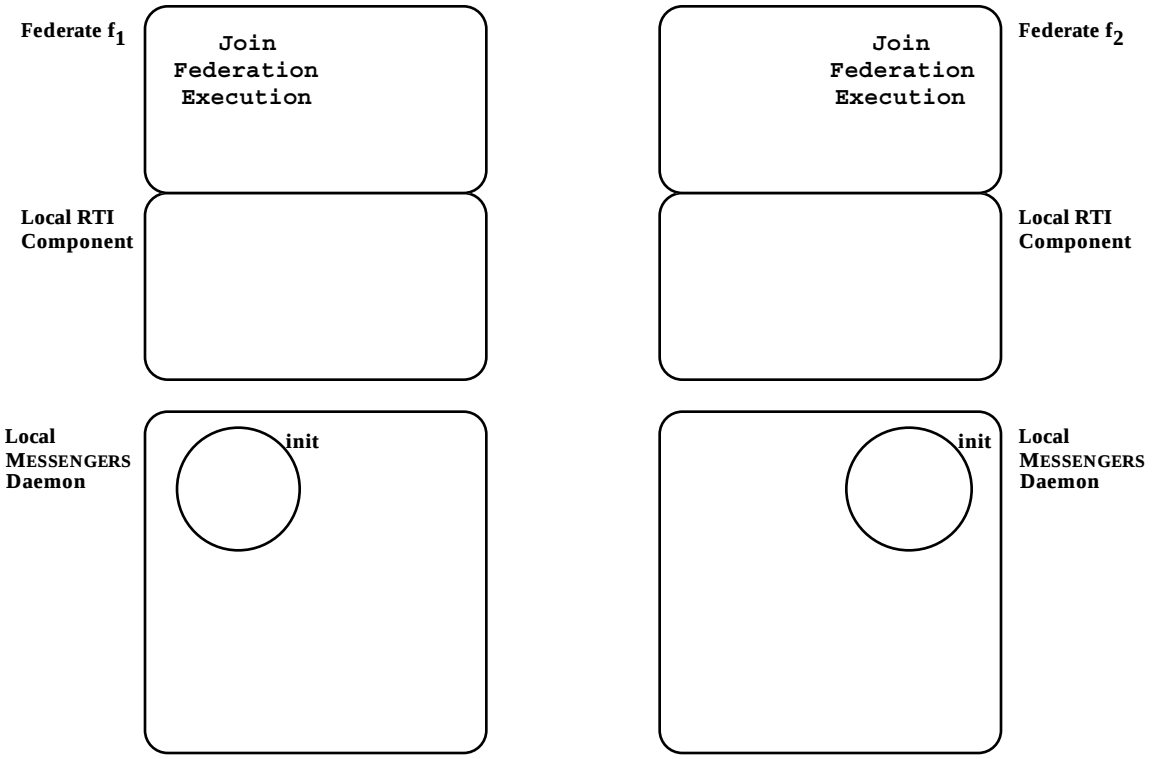

Figure 4-1. Federate Join and Federation Creation

the RTI creates a local MESSENGERS daemon associated with the federate which has an initial virtual node (init) as shown in Figure 4-1. Each federate which subsequently joins the federation also gets an init node. Communication between the local RTI component and the MESSENGERS daemon is performed via Unix IPC.

When a federate declares its intent to publish an object class through a routing space, a routing space initialization Messenger is injected into the init node from which it creates another virtual node, the routing space node, which will host all update regions for the object class/routing space pair. The routing space initialization Messenger creates links from the new routing space virtual node to all non-local init nodes as illustrated in Figure 4-2. The routing space initialization Messenger also creates a unique link, also

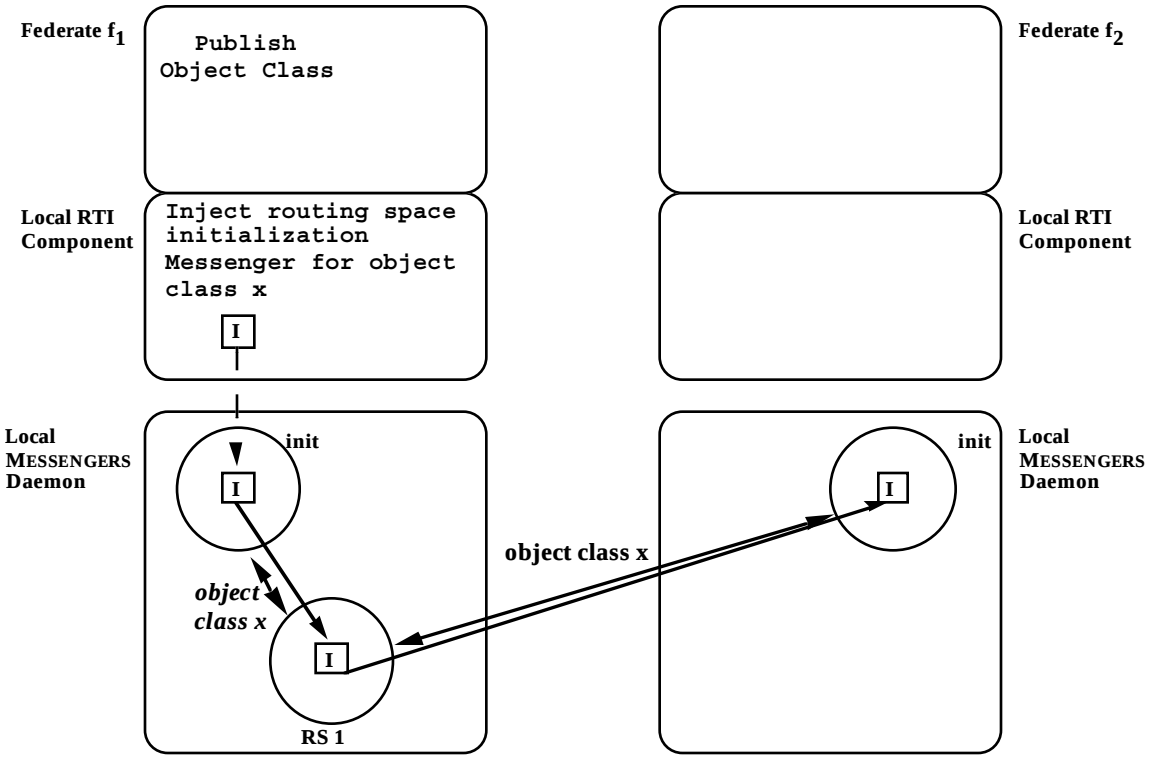

Figure 4-2. Routing Space Initialization

labeled with the object class, from the new routing space virtual node to the local init node.

Subscription region Messengers traverse the object class links to arrive at the correct routing space virtual nodes where they can expect to locate update regions matching their object class/routing space pair, as shown in Figure 4-3.

When a federate registers an object instance with an update region, an update region Messenger in injected which traverses the unique object class link in the local MESSENGERS daemon to arrive at the correct routing space virtual node as shown in

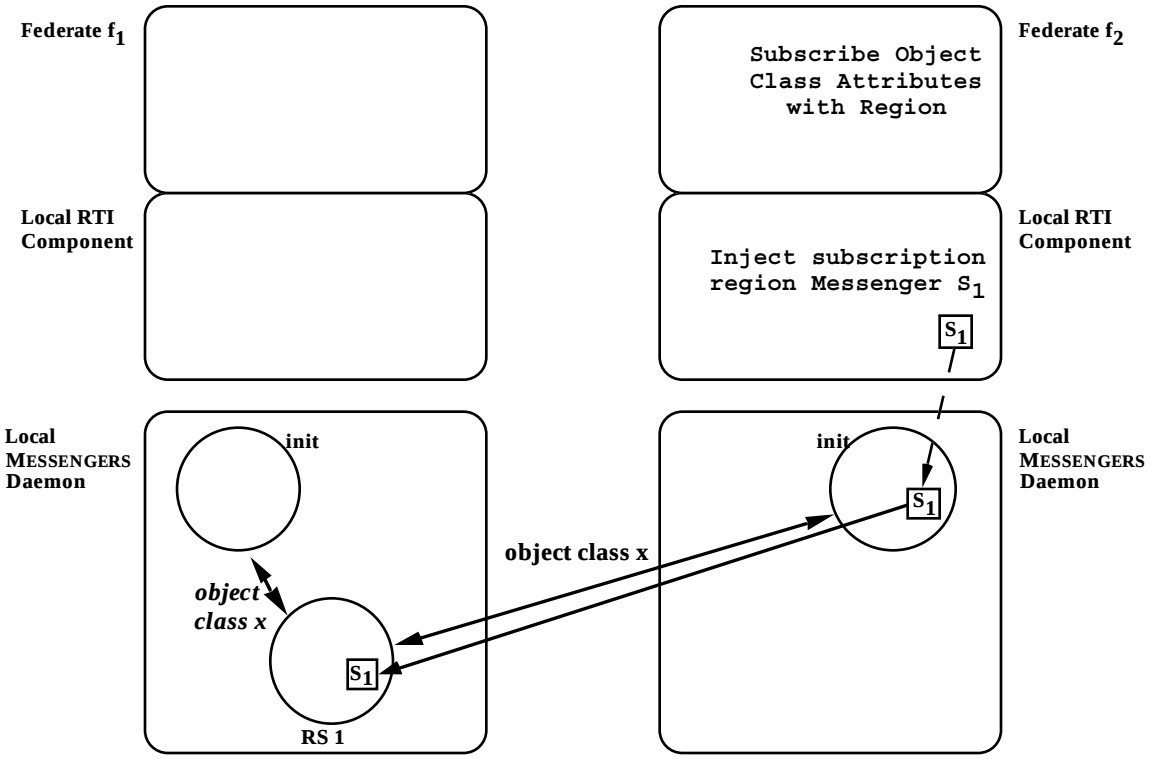

Figure 4-3. Subscription Object Class Attributes With Region

Figure 4-4. There it interacts with appropriate subscription regions to determine region overlaps.

If an overlap is found, the update region Messenger duplicates itself. The parent Messenger reports the overlap back to the local RTI component so that the local RTI component can establish connectivity for subsequent attribute updates. The child Messenger traverses the appropriate object class link to the subscribing federates MESSENGERS daemon where the object discovery is reported back to the subscribing federate, Figure 4-5.

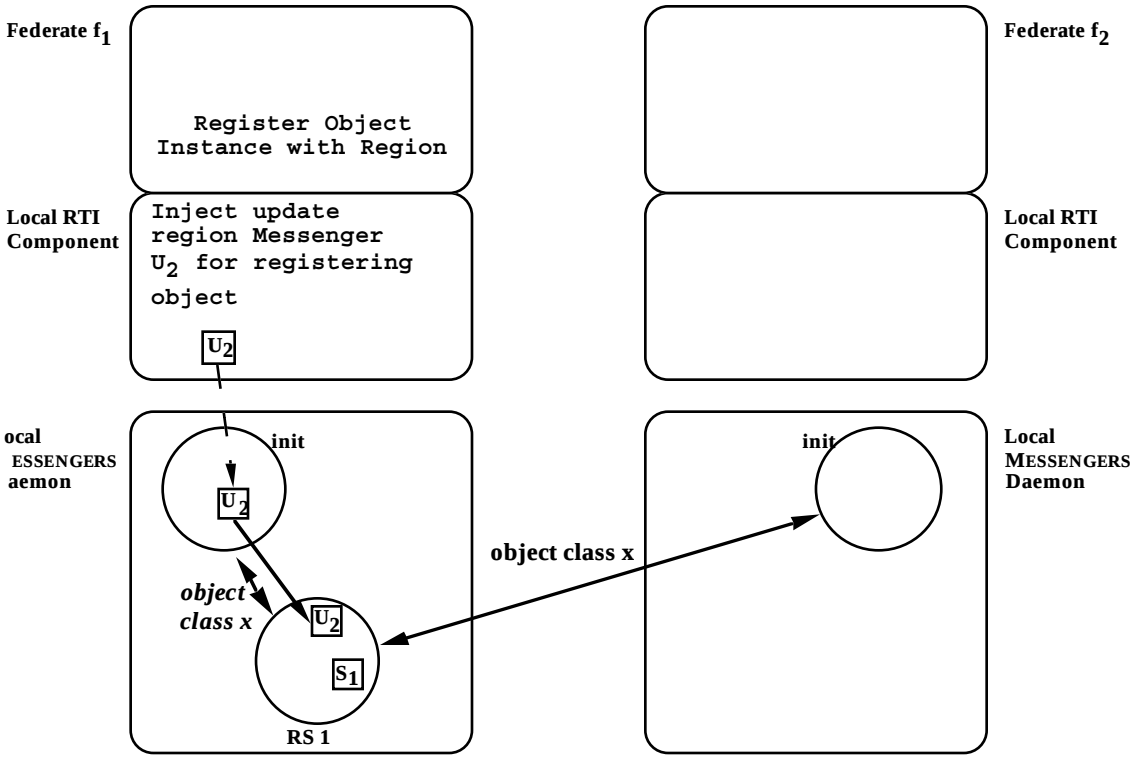

Figure 4-4. Register Object Instance With Region

4.3 Offline Grouping Algorithms

Two grouping algorithms were developed: the largest outgoing connection algorithm and the input-restricted largest outgoing connection algorithm. Both were initially tested offline with global knowledge against static connection sets. The resulting groupings were tested with an offline simulation.

4.3.1 The Global Largest Outgoing Connection Algorithm

The first grouping algorithm runs offline with global knowledge of all connections. t_s, t_r and the t_p matrix are provided as inputs. This algorithm is a greedy algorithm and is referred to as the “largest outgoing connection” (LOC) algorithm. The LOC of any node is

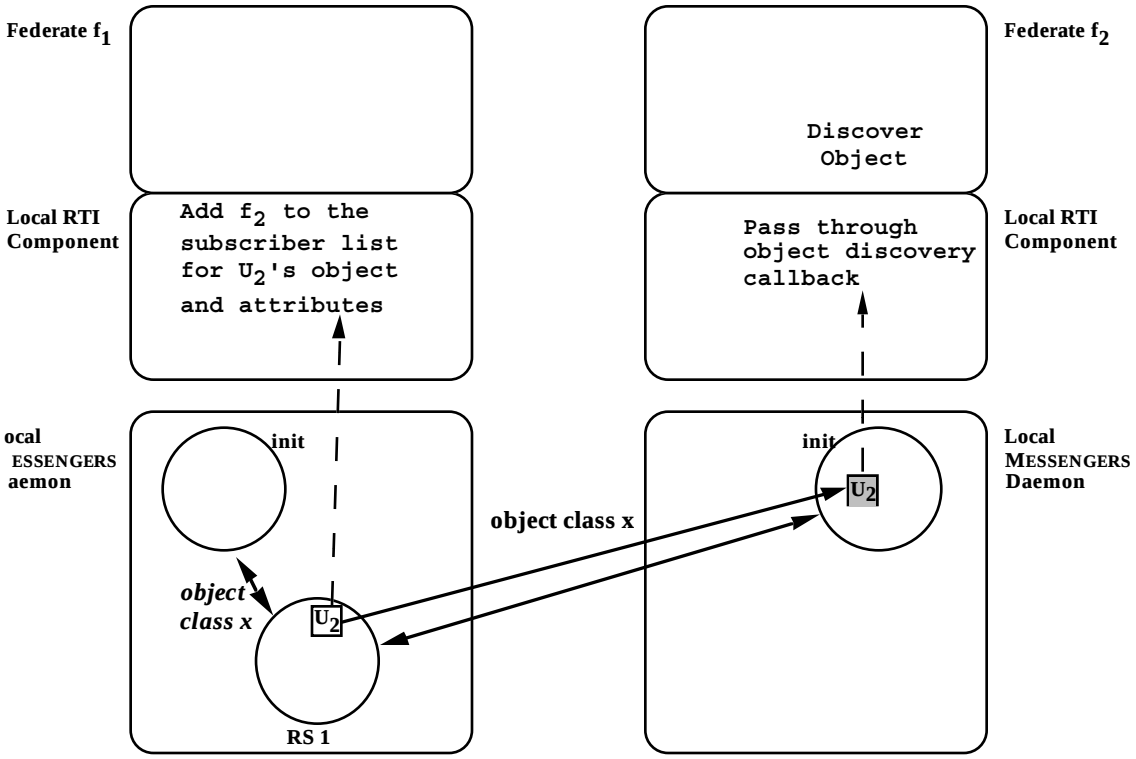

Figure 4-5. Discover Object and Establish Connectivity

calculated as its connection weight, w, multiplied by the number of receivers, k, because $k \bullet t_s \bullet w$ is the total send time for point-to-point which is one of the quantities to reduce.

The algorithm performs the following steps:

1. Add the node and connection with the largest outgoing connection weight; in the event of a tie, add the lowest numbered such connection.
2. Add all the connection's receivers.
3. Repeat steps 1 and 2 with the remaining connections from current group members based on their LOCs only as long as their addition doesn't cause the group to exceed t_{max}. Halt when all connections are assigned or all multicast groups are used.

4.3.1.1 Simulating Algorithm Goodness

The goodness of the results of the largest outgoing connections algorithm was tested with a discrete event simulation which takes as input a configuration file specifying:

- f (the number of federates)
- t_p (a pairwise matrix for all the federates)
- t_r
- t_s
- t_{max}
- a connection set with connections assigned either to multicast groups or to point-to-point communication.

The simulation simulates 10 seconds of updates at millisecond resolution according to the connection set and measures:

- Average message queue time
- Average message delivery time
- Number of messages
- Number of late messages
- Average queue length

If an update is sent point-to-point, individual copies of the update are sent to multiple receivers at intervals of t_s. Updates sent via multicast are sent simultaneously.

4.3.1.2 Expected and Actual Results on Selected Connection Sets

The first tests of the LOC algorithm were performed on selected connection sets. The connection sets were selected to exercise cases on which the algorithm was expected to perform either particularly well or poorly. The tests used combinations of:

- Uniform weights on all connections vs. variable weights
- Fully connected subgraphs vs. strongly connected subgraphs vs. sparsely connected subgraphs
- t_p = 1 millisecond, 10 millisecond, and 50 milliseconds between all pairs of nodes.

Figure 4-6, Figure 4-7, and Figure 4-8 illustrate several of these connection graphs.

Varying t_p had little effect other than to increase the average message delivery time by the change in t_p. Likewise, the difference between using uniform weights or variable weights produced little change in the results beyond the number of messages received and the expected accompanying changes in the other measures. Some of these experiments were run with low connection weights with the result that all the connections could be added to the multicast group without any receiver exceeding t_{max}. However, this still resulted in all receivers receiving a proportionally large number of messages leading to message delays.

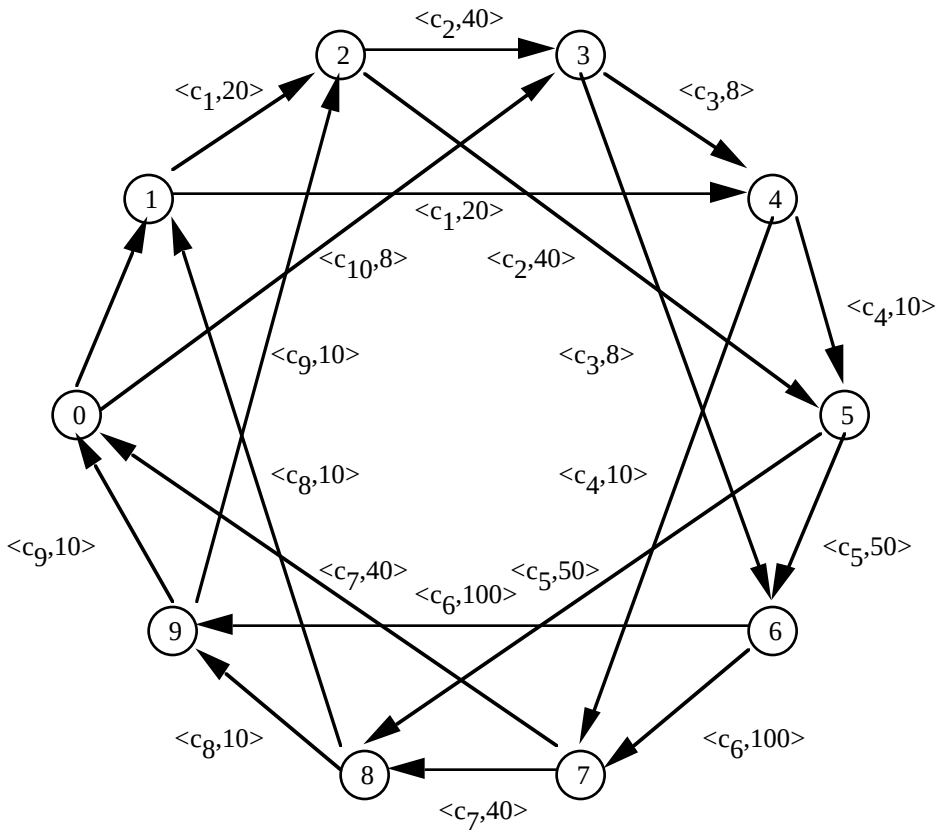

Figure 4-6. Sparse Uniform Connectivity, Variable Weights, 1 Connection/Node

When subgraphs are not strongly connected, some incoming connections to a node may be considered while others aren't. For example, if both of f_1's outgoing connections in Figure 2-6 are put into a single multicast group, f_2 will receive c_3 extraneously. While these messages may not themselves cause f_2 to exceed t_{max}, the addition of the weight of c_2 may. This leads to the following observation:

The grouping algorithm should take into account all incoming connections as well as outgoing connections because an incoming connection not included in the multicast grouping can overwhelm a receiver.

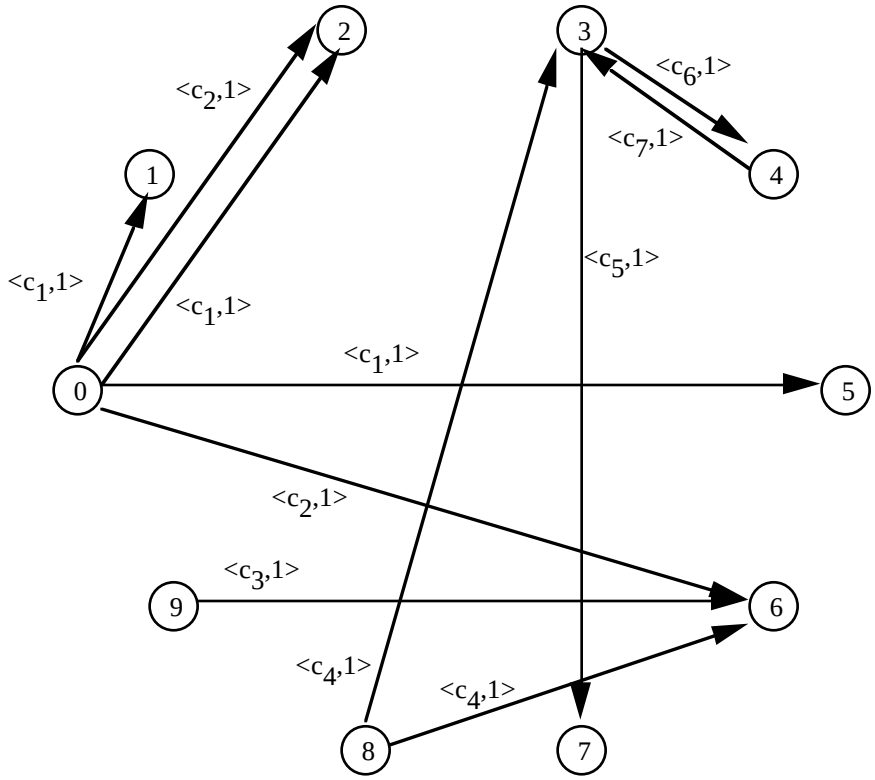

Figure 4-7. Non-Uniform Sparse Connectivity, Uniform Low Weights, Multiple Connections/Node

As expected, the biggest difference was observed when the degree of connectivity of the subgraphs varied. Predictably, the algorithm performs best when the subgraphs are completely connected, i.e. within a subgraph every node is in the receiver set of all of every other node's connections. t_s is minimized for every connection because it can be sent via multicast. No node receives any extraneous messages, so t_q is not negatively impacted. This leads to a second and more important observation:

The grouping algorithm must use empirical measures for predicting t_q to make accurate predictions about the impact of sending extraneous messages.

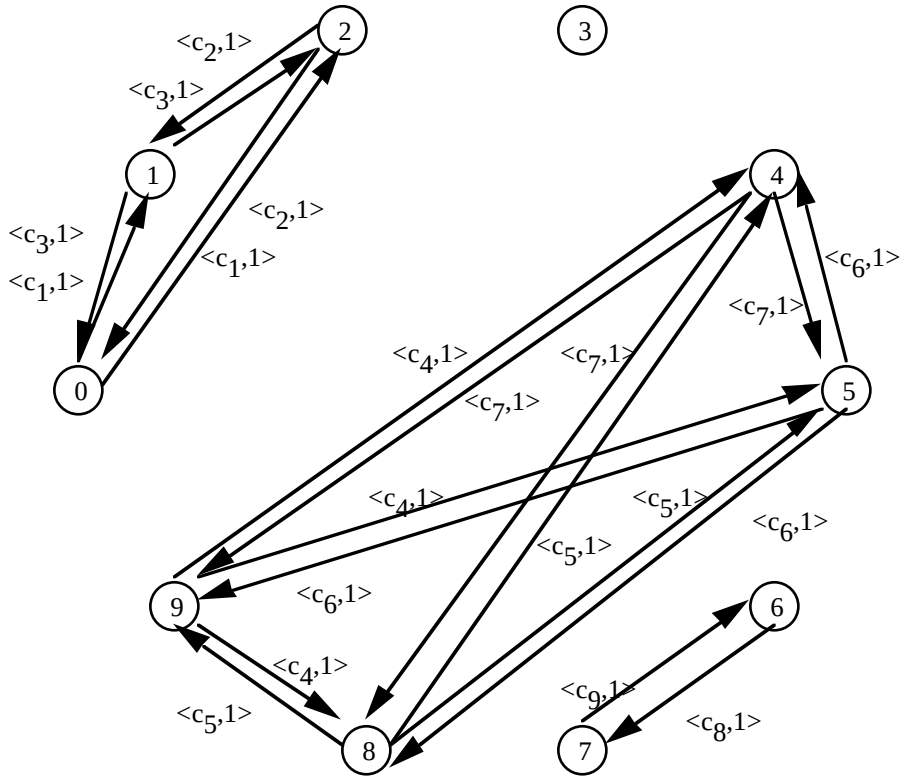

Figure 4-8. Fully Connected Subgraphs, Uniform Low Weights, Multiple Connections/Node

In summary, the LOC algorithm makes grouping decisions designed to minimize t_{ds} and the simulation results show that it's successful at doing so. However, it makes no effort to minimize t_q and the results also show that it sometimes fails to produce good results because of this.

4.3.1.3 Comparison of LOC to the Exhaustive Approach for Random Sets

The LOC algorithm was tested against random connection sets generated by a tool which takes as input a partial configuration file containing the number of federates, f, and the t_p array, and the total number of connections to generate, n. The tool generates a full configuration file for the simulation assuming point-to-point communication. The full

configuration file contains the requested number of connections where the sender ID, connection weight, number of receivers, and receiver IDs of each connection are all generated randomly (using the Unix rand() function).

A related tool was built which took these random connection sets and generated all possible groupings of the connections into a specified number of multicast groups according to the total possible combinations given in Equation 2-2.

Experiments were performed with the random connection sets in Table 4-1. n and m were necessarily kept small to compensate for the combinatorial growth of the possible combinations.

Table 4-1: Random Connection Set Tests

f	10
$t_r = t_s$	10 milliseconds
t_p	10 milliseconds between all pairs of federates
t_{max}	1000 milliseconds
(n, m)	(5, 2), (5, 3), (6, 2)

Ten random connection sets of the given sizes were generated, using different random seeds, and the results were averaged to minimize the impact on the results of randomly pathological connection sets. For each of the 30 experiments, the following measures were generated:

1. Average message delivery time for the LOC algorithm
2. Average message delivery time for point-to-point communication
3. Average message delivery times for all possible groupings as generated by the exhaustive algorithm; note that the LOC and point-to-point solutions are also contained in this set.
4. Best, worst, mean and median of Table 4-1
5. The percentage of solutions from Table 4-1 which performed better than the LOC algorithm.

A total of 6770 simulation runs were executed for the 30 experiments. The results can be summarized as follows:

1. In 22 of the 30 experiments, the LOC algorithm produced a better average message delivery time than both point-to-point and at least 50% of all possible solutions.
2. Of the remaining 8 cases, 5 failed because the LOC algorithm fails to account for overload of incoming connections.
3. In the other 3 cases, one or more federate was output-throttled by using point-to-point. An output-throttled connection is one for which k•w exceeds t_{max}. The first effect of such a connection is that the sender physically cannot send all the required updates in time without using multicast. The second side effect is that the average message delivery time at the receivers may appear lower because many messages never leave the sender. When these connections were added to multicast groups, the sending federate was able to send all messages at the desired rate, resulting in send rates as much as four times higher. Of course this resulted in slower average receive rates since the

receiving federates had to receive four times as much data.

4. In the single worst experiment, LOC performed worse than 63.8% of the solutions for one connection set.
5. In the single best experiment, LOC produced the best results for one connection set.
6. On average across all simulation runs from all the experiments, only 28.2% of the solutions were better than those generated by LOC.

4.3.2 The Input-Restricted LOC Algorithm

The input-restricted largest outgoing connection (IRLOC) algorithm seeks to minimize both t_{ds} and t_q to produce better average results than the LOC algorithm. This algorithm recognizes three facts about adding a connection to an existing group:

1. Any receivers of the connection who are not already in the group will receive additional connections equal to the sum of the connections already in the group, the group weight.
2. Any group members who are not receivers of the connection will receive the additional weight of the connection.
3. Assuming that $t_s = t_r$, improvements in average message delivery time created by sending a connection via multicast are directly offset by the "negative weight" created by the first two facts.

Taking these three facts into account, the IRLOC algorithm performs the following steps:

1. Calculate the positive cumulative effect of each connection, $(k - 1) \bullet w$.

2. Add the receivers of the connection with the largest cumulative effect; in the event of a tie, add the lowest numbered such connection.
3. Add the next largest connection such that a) the input weight of the current group members does not exceed t_{max} by the addition of the connection weight, b) the input weight of the connection's receivers not already in the group does not exceed t_{max} by the addition of the group weight, c) the positive cumulative effect is greater than the negative weight. Note that the positive cumulative effect is only a function of the connection, while the negative weight is a function of the connection and the current state of the group.
4. Repeat step 3 with the remaining connections. Halt when all connections are assigned or all multicast groups are used.

4.3.2.1 Simulation Results Comparing LOC and IRLOC

The discovery of the effects of output-throttled connections lead to the realization that the average message delivery time reported by the offline simulation doesn't capture all the important measures of goodness when the limits of message delivery are tested. And these are precisely the cases of most interest. As a result, the evaluation criteria were refined beyond just average message delivery time to include the effects of output throttling and overflowing the receiving queue. Output throttling and exceeding t_{max} are boolean failure conditions; any algorithm which produces either effect for a connection set is considered to have failed. If an algorithm succeeds on these two criteria, it is compared to other successful algorithms on the basis of average message delivery time. However, the message delivery time is tempered by the number of extraneous messages it delivers. Consider the following example. Algorithm A delivers 200 messages with an average

message delivery time of 30 ms. Algorithm B delivers 100 messages with an average message delivery time of 35 ms. Algorithm A has better average message delivery time, but it used 6 seconds to deliver the messages as compared to 3.5 seconds for Algorithm B. Furthermore, if the receiver only required the 100 messages delivered by Algorithm B, it would have had to take additional time to discard the 100 extraneous messgaes delivered by Algorithm A. So, the four evaluation criteria are:

1. No receiving federate exceeds t_{max} for its average message delivery time (Success or Failure)
2. No sending federate is output throttled for any connection or set of connections (Success or Failure)
3. Average message delivery time
4. Number of extraneous messages

The 30 random test cases were rerun for just LOC, IRLOC, point-to-point, and broadcast. Recall that point-to-point and broadcast are opposite ends of the spectrum for improving and degrading message delivery. Point-to-point is the worst case for message sending, negatively impacting t_{ds}, while it's the best case for message reception, positively impacting t_q. Broadcast is the best case for message sending, positively impacting t_{ds}, while it's the worst case for message reception, positively negatively t_q. Note that broadcast never fails to send all the required messages unless the sending federate has too many connections to be able to send even one update of each connection at the required frequency. In addition to the measures described in Section 4.3.1.1, these experiments also reported the incoming connection weight at each federate, the number of "output-

throttled" connections that each federate is sending, and the number of "output-throttled connections" of which each federate is a receiver. The percentage of extra messages, both positive and negative, was calculated for LOC, IRLOC, point-to-point, and broadcast based on the incoming weight at each federate using point-to-point. The point-to-point incoming weight reflects the number of messages which should be received, but not necessarily the number that are received with point-to-point owing to output throttling effects. The full results of these experiments are contained in Appendix A.

Any grouping which results in output-throttled connections or any receiver having an average message delivery time greater than t_{max} are considered to have failed. 26 of the point-to-point experiments and 8 of the broadcast experiments failed on one or both of these criteria. These are precisely the types of test cases to which grouping is applicable. Figure 4-9 through Figure 4-12 show the results of all the point to point and broadcast experiments Tests which failed the t_{max} or output throttling criteria are indicated by bars extending above the delivery time, solid for output throttled and checkered for t_{max}.

Among test cases which pass the first two criteria, a lower average message delivery time is preferable. In all cases, LOC produced lower average message delivery time than point to point, but often at the predictable cost of delivering extraneous messages, as illustrated in Figure 4-13 through Figure 4-15. In 24 of the 30 test cases, LOC produced lower average message delivery time than broadcast. However, in all cases it delivered less data, as much as 50% less data. The analysis of the remaining 6 test cases

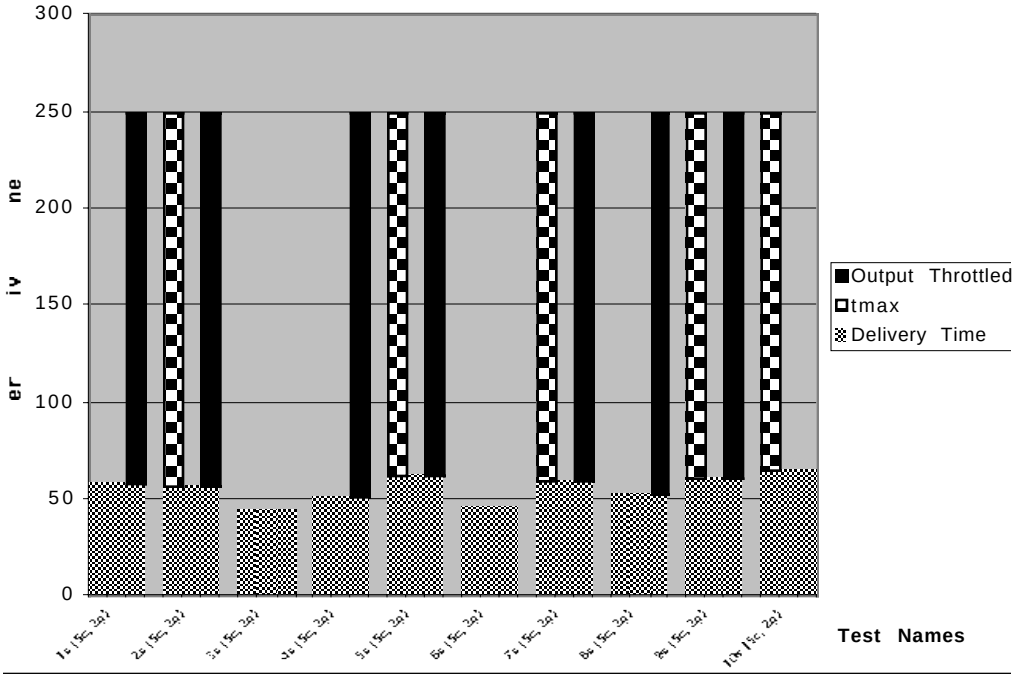

Figure 4-9. 5 Connections, Point to Point

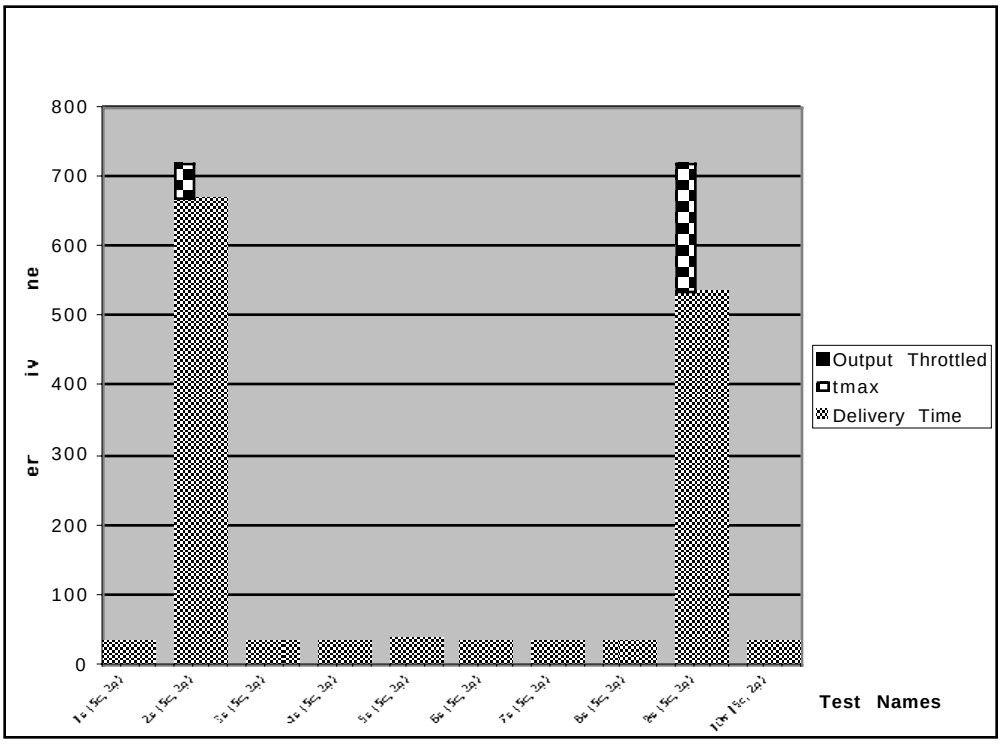

Figure 4-10. 5 Connections, Broadcast

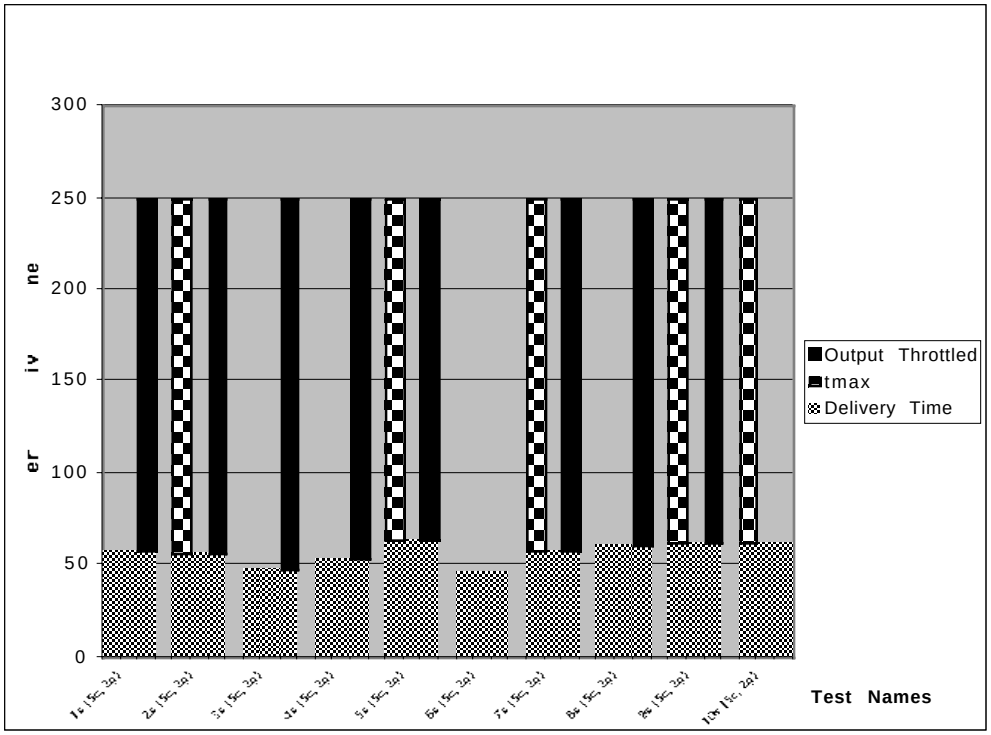

Figure 4-11. 6 Connections, Point to Point

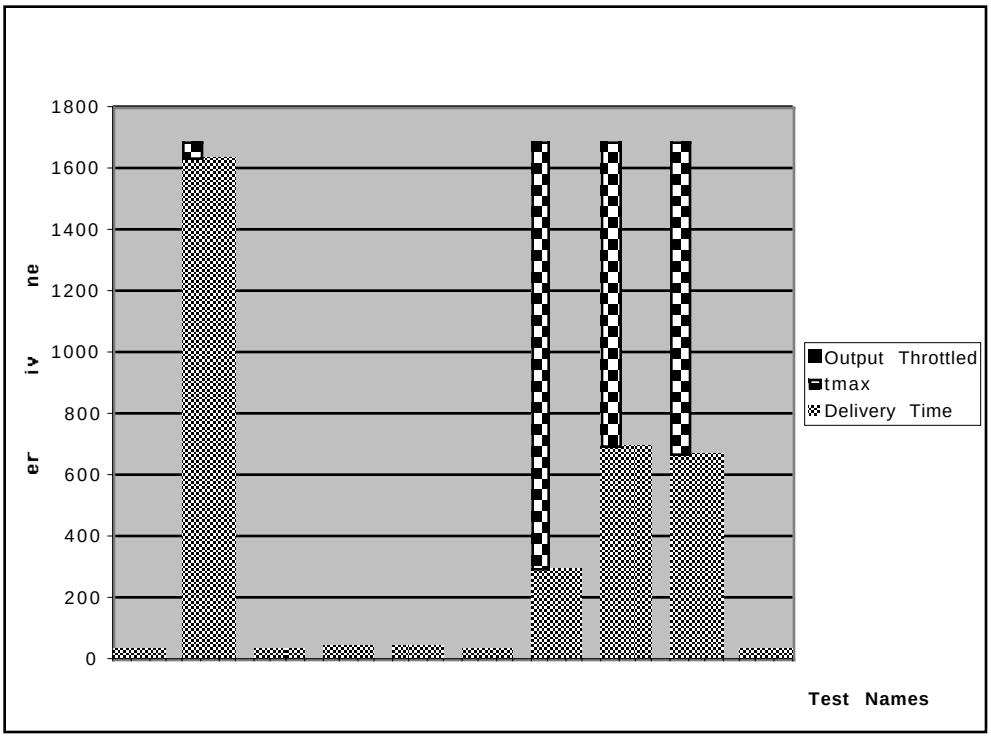

Figure 4-12. 6 Connections, Broadcast

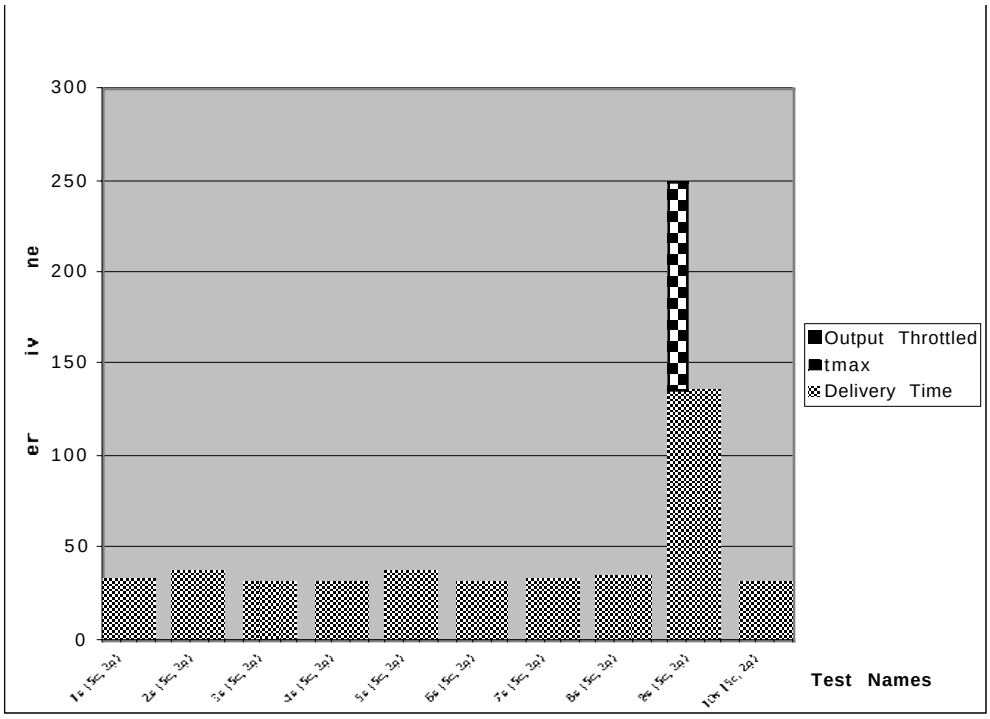

Figure 4-13. 5 Connections, 2 Groups, LOC

reveals that the LOC algorithm could be more aggressive about adding connections to groups because the time to discard messages is much lower than the time saved by reducing sends. In all cases where LOC failed the t_{max} or output-throttling criteria, no solution exists to prevent these conditions because either some receivers had point-to-point input weights which cause them to exceed t_{max} or the sender had multiple connections with output weights that they could not all be sent, even if they were all assigned to multicast groups. Details of the individual failure causes can be found in Appendix A.

The IRLOC algorithm generates consistently good results for all cases where a good solution exists. In 23 of the 30 cases, IRLOC produced average message delivery

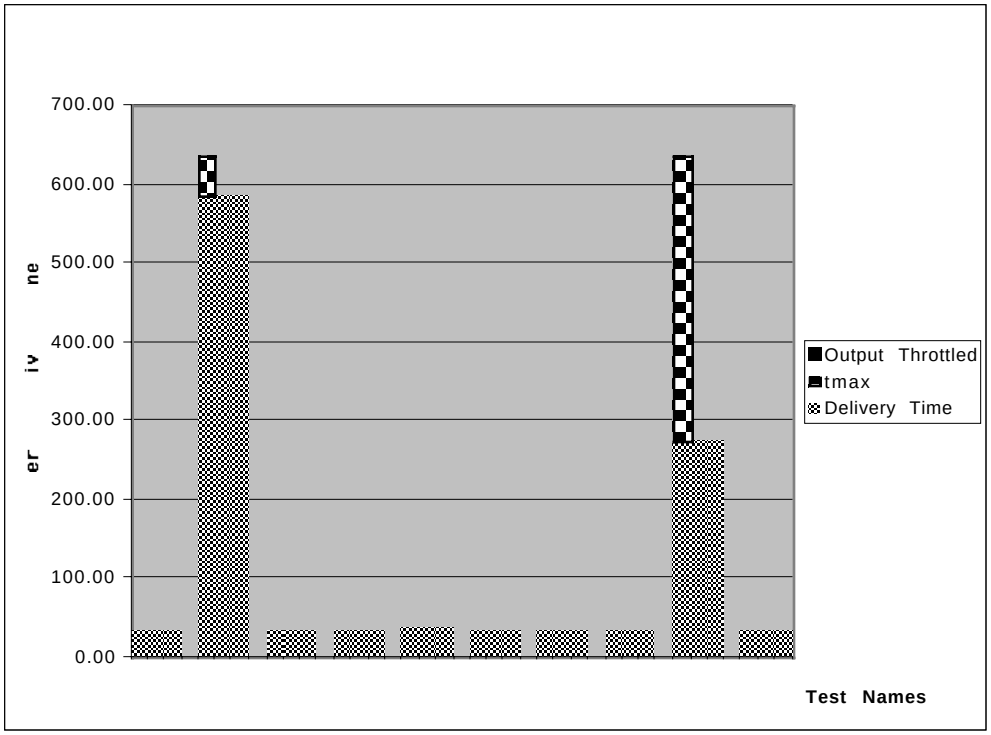

Figure 4-14. 5 Connections, 3 Groups, LOC

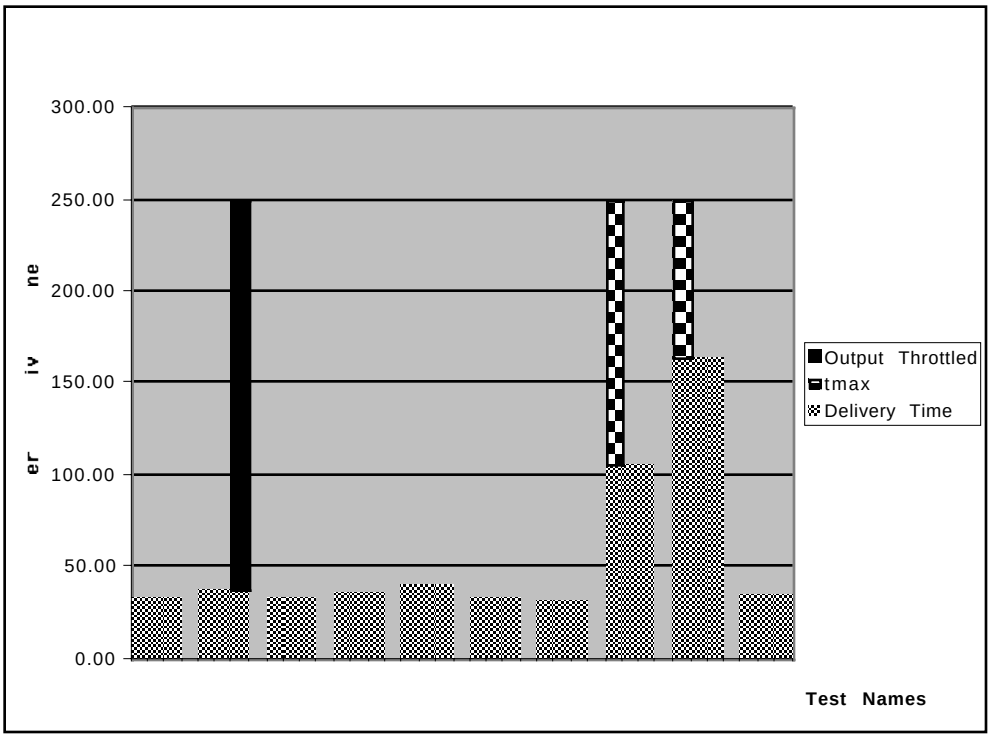

Figure 4-15. 6 Connections, 2 Groups, LOC

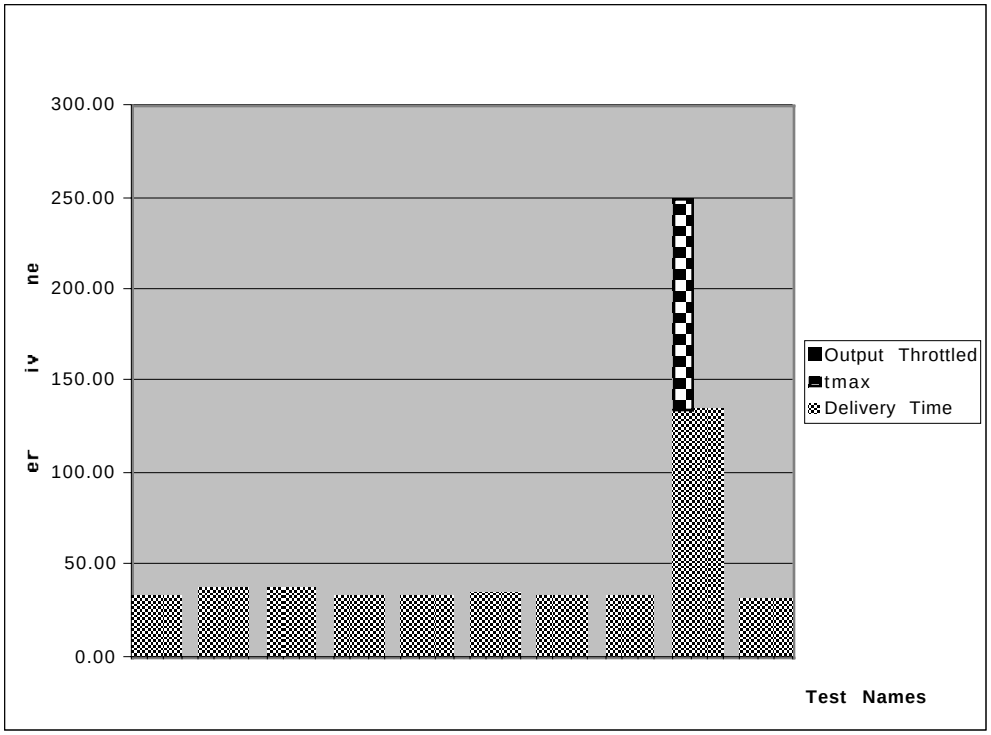

Figure 4-16. 5 Connections, 2 Groups, IRLOC

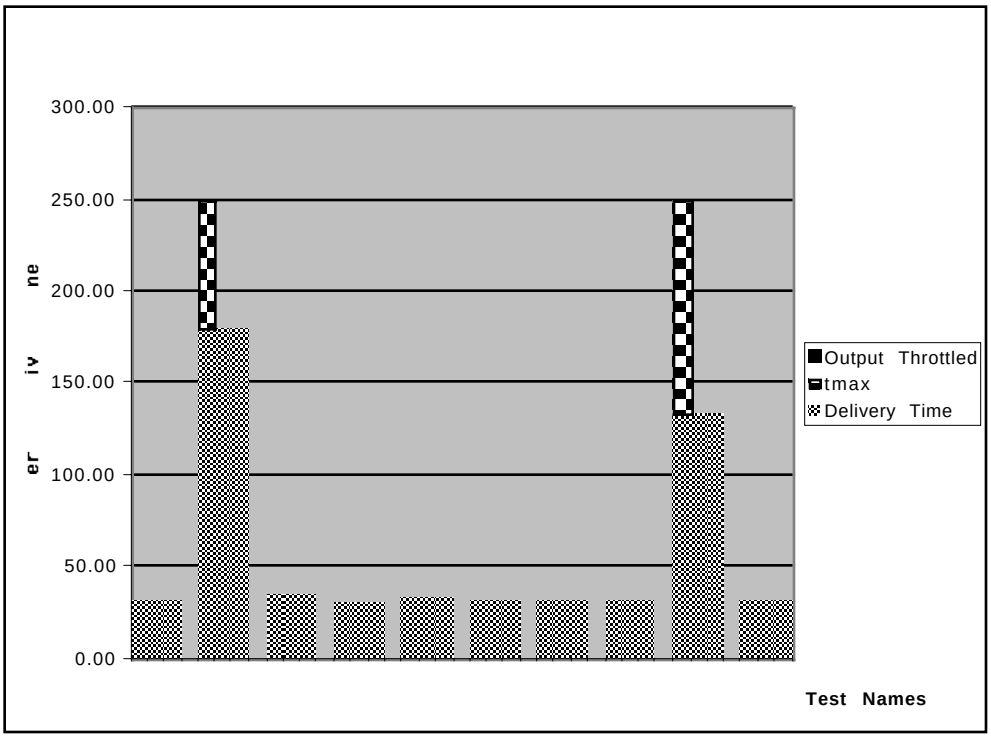

Figure 4-17. 5 Connections, 3 Groups, IRLOC

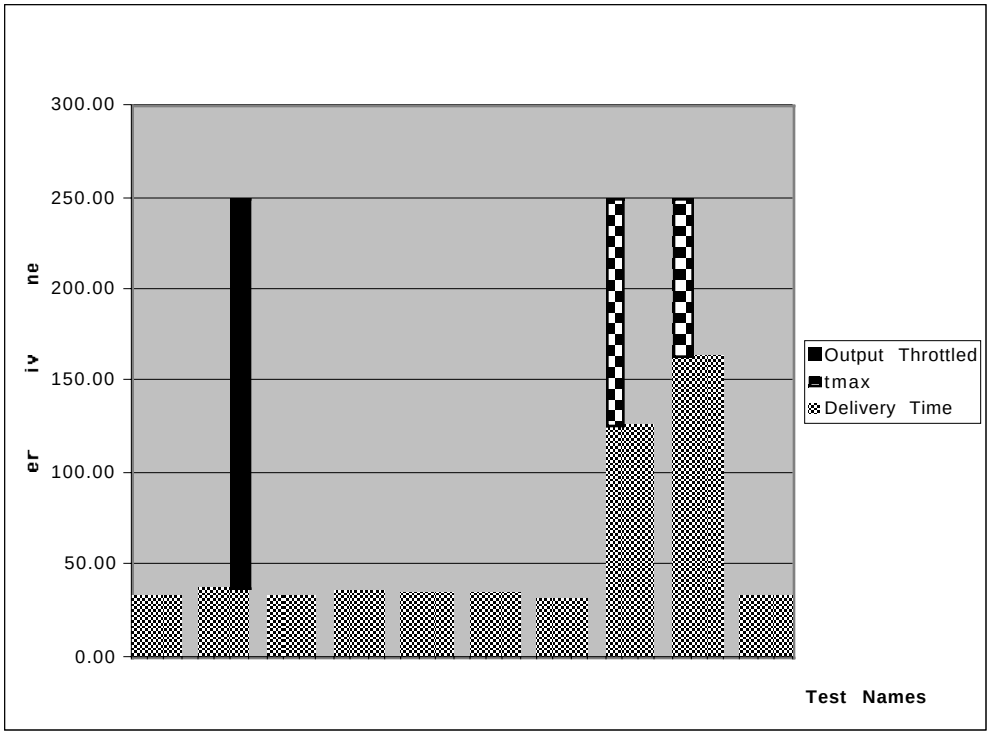

Figure 4-18. 6 Connections, 2 Groups, IRLOC

times as low or lower than IRLOC, and usually with more accurate delivery of the correct data. In three of the seven cases in which IRLOC had higher message delivery times, it delivered exactly the correct set of data in 34.82, 34.11, and 34.64 milliseconds vs. 32.03, 32.07, and 33.24 milliseconds for LOC where the fastest possible delivery time is 30 milliseconds given $t_s = t_r = t_p = 10$ milliseconds. There are two important points about these three cases. First, while IRLOC delivered exactly the right data for all three cases, LOC delivered 252%, 102%, and 268% **extraneous** messages for the same cases. Second, all seven cases had fairly light connection sets, i.e. nearly all of the connections could have been put in a single group without exceeding t_{max}. The IRLOC algorithm balances the positive effect of adding a connection to a group against the **potential** negative effect of adding it. When grouping light connection sets, this approach is overly conservative because the receivers have a lot of spare time to throw away extraneous messages, i.e. the

potential negative effect is higher than the actual negative effect. Under such circumstances, the more aggressive LOC grouping algorithm works better and the additional overhead of IRLOC is not warranted.Figure 4-16 through Figure 4-18 show the results of the IRLOC algorithm.

4.3.2.2 Larger Connection Sets

The entity counts and bit rates for the large connection set experiments were derived from STOW-E data [NCCOSC95]. The original bit rates and the round-ups used for connection weights are given in Table 4-2.

The STOW-E network analysis divided the data into eight time periods. Since there was no time period during which all entity types were present, the entity numbers used for this experiment are the averages for the individual entity types across all eight time periods. The entities were allocated to federates by side, with tanks, trucks, and dismounted infantry grouped into armored battalions. The numbers of entities and

federates are given in Table 4-2.The same 20 federates represent tanks, trucks, and dismounted infantry.

Table 4-2: Entity Connection Weight by Type and Counts by Federate

Entity Type	Kbps	Connection Weight	# of Entities	# of Federates
Submarine	1.09	2	2	1
Ship	1.16	2	11	2
Fixed-wing aircraft	3.72	4	36	2
Rotary-wing aircraft	2.40	3	35	2
Tank	1.27	2	600	24
Truck	1.09	2	456	24
Dismounted infantry	1.09	2	336	24
Total			**1476**	**31**

The battlespace is approximately 400 km by 500 km, with region ranges as given in Table 4-3.Subscriptions by class type are as given in Table 4-4. Individual subscriptions can be derived from Table 4-3 and Table 4-4, e.g. submarines subscribe to opposing ships and other submarines within 25 km.

Table 4-3: Region Ranges

Entity Type	Subscription Region Range	Update Region Range
Submarine	12 km	12 km
Ship	12 km	12 km
Fixed-wing aircraft	25 km	2550 km
Rotary-wing aircraft	20 km	20 km
Tank	12 km	12 km
Truck	12 km	12 km
Dismounted infantry	12 km	12 km

Table 4-4: Class Subscriptions

Entity Type	Subscribes to
Submarine	Submarine
	Ship
Ship	Ship
Fixed-wing aircraft	Fixed-wing aircraft
	Tank
	Ship
Rotary-wing aircraft	Rotary-wing aircraft
	Tank
Tank	Tank
Truck	Truck
	Dismounted infantry
Dismounted infantry	Dismounted infantry

Three “snapshots” were taken of an engagement: prior to engagement, at the point of engagement, at the end of the engagement. These snapshots were generated using the Integrated Theater Level-Engagement Model [SAIC99][1] and are provided in Appendix B. In the figures, semi-circles represent submarines, circles represent blue ships, and diamonds represent red ships. The small ‘m’ icons represent 6 fixed wing aircraft. The small ‘m’ icons with the bar across the top represent 7 rotary wing aircraft. The rectangles represent armored forces each consisting of 25 tanks, 19 trucks, and 14 dismounted infantry for a total of 58 entities per armored force.

1. Thanks to the ITEM team for helping me generate these graphics: Steve Vedder, Doug Boyles, and Bill Macak.

In the pre-engagement snapshot, none of the entities are close enough to opposing forces to receive any data. While this doesn't produce multicast grouping results, it is a testament to the value of interest management in general.

The engagement snapshot is predictably the most interesting one. Visual inspection of the graphic reveals the highest level of grouping between opposing forces. Because many of the groups involve 58-entity armored forces, this snapshot results in 1052 connections. The armored forces are "aggregated" entities, i.e. one icon in the snapshot represents multiple individual entities. As a result of aggregation, all entities in the aggregate have the same update regions and the same subscription regions. This results in particularly high numbers of connections. However, if the entities were individually represented, their regions would only be perturbed slightly from the aggregated region. This would result in slightly lower numbers of connections, on the order of 5% to 10%. Simulating broadcast using the offline simulator was impractical as it would have required the simulator to handle approximately 950,000 events to simulate 10 seconds. Several experiments with this connection set indicate that the maximum number of groups it can make productive use of is 8. Compare this with a static allocation of multicast groups to grid cells. With the 400 km by 500 km battlespace, 10 km by 10 km grid cells would require 2,000 multicast groups; 5 km by 5 km grid cells would require 8,000.

This is illustrated in the small for the grouping of armored forces near the left center of the engagement snapshot. Figure 4-19 shows the grouping of four red armored forces and three blue armored forces, and their update and subscription regions. Since

armored forces have update and subscription regions of the same size, only one region is shown for each force. The armored forces and their regions are shown superimposed on a 5 km by 5 km static grid. Of the 100 grid cells shown, only 33 of the cells are used, while only 12 are productively used to exchange data between interacting forces. Dynamic grouping put all of these connections into a single group, and this is an area of the scenario with very dense interaction between entities. This scenario contains over 100,000 square kilometers of empty sand and ocean to which a static grid assignment of multicast groups would have wastefully allocated thousands of multicast groups!

While no single connection was output throttled, several of the armored forces federates were output throttled by virtue of the number of entities they were simulating. The input weight of 15 federates exceeded t_{max}. Although the IRLOC delivered 50% more messages than point-to-point, it was still only able to deliver less than 50% of the required messages and the average message delivery time exceeded t_{max}. When the number of entities per armored force was successively lowered to 20 and 10, the connection set became tractable to the point where the IRLOC algorithm could deliver 90% of the messages with only one federate overruning t_{max} because its input weight was 180.

In the end snapshot, there is significantly less grouping among federates, but much of it involves armored forces with their 58 entities. As a result, the end snapshot has 350 connections. However, most of these connections only have a single receiver, so the net

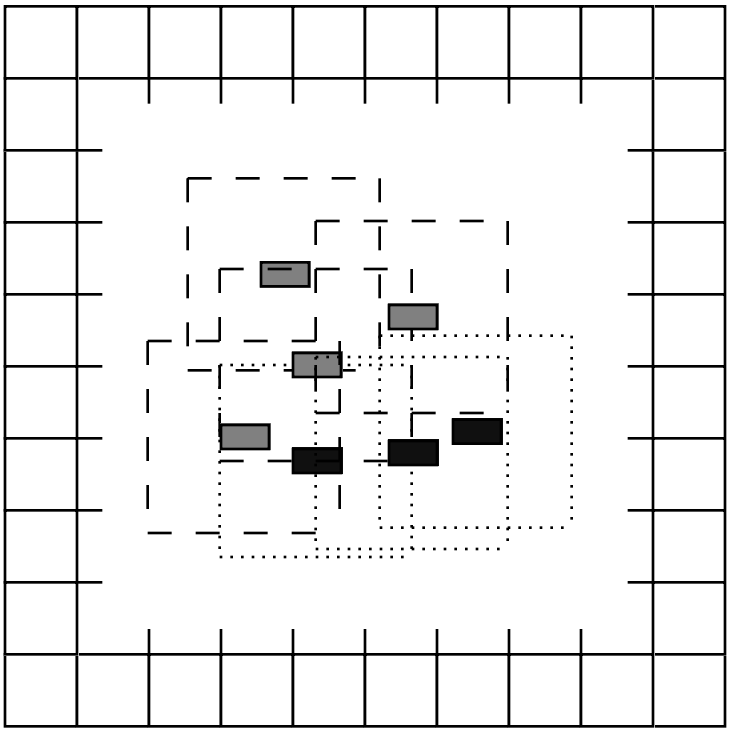

Figure 4-19. Static vs. Dynamic Allocation of Multicast Groups

gain over 10 seconds of simulation time is only 200 microseconds and only uses one multicast group.

The conclusions to be drawn from this experiment are two-fold. First, interest management problems exist for which there is no solution. In distributed simulations, these problems manifest themselves in message overflow and late messages. In the real world, they manifest themselves in overloaded individuals who make bad decisions because they cannot assimilate and analyze all the data presented to them. Second, and more important, when good grouping solutions exist for connection sets with typical chaotic clustering, the IRLOC algorithm can find one.[2]

2. A stronger assertion can probably be made about the algorithm's applicability to random connection sets, but would require more extensive analysis and experimentation.

4.4 Online Distributed IRLOC Algorithm

The online, distributed IRLOC algorithm integrates the baseline prototype with the basic structure of the IRLOC algorithm. The distributed algorithm operates with degraded information for several reasons. First, the information about connections and incoming weights is distributed among the federates, and collecting it would be prohibitive in a very large scale distributed simulation. If the simulation were small enough to be able to collect all this data at a central point and still make timely grouping decisions, it wouldn't need multicast grouping. Second, this same information is changing in real time. Regions and region intersections are changing while the grouping decisions are being made. Even if the algorithm had access to global information when it started grouping, there is a non-zero probability that the information would be out of date by the time the grouping completed. Finally, the MESSENGERS system doesn't provide a straightforward, timely mechanism for passing dynamic data structures to Messengers. Some simplifying assumptions have been made which account for this. In a production system this final constraint could be relaxed.

The online grouping algorithm is triggered by the discovery of a connection or connections. A grouping Messenger is injected which begins searching for a potential group. The grouping Messenger searches three places in the following order:

1. on the init node on the local machine;
2. at the multicast server where it may find an unused group;
3. at most one hop from the multicast server at another machine.

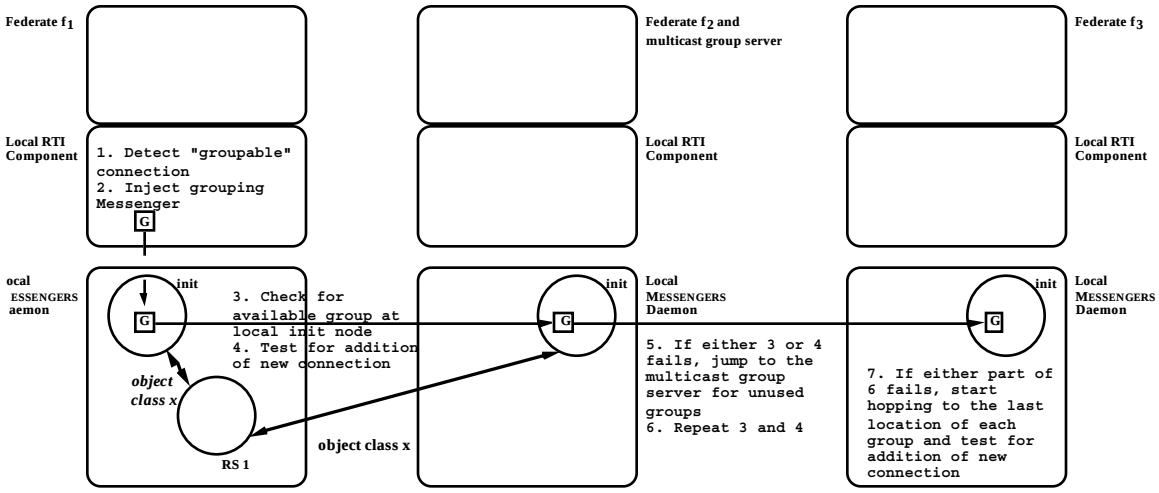

Figure 4-20. Attempt to Add a Connection to a Group

The multicast servers are selected a priori and specified in a configuration file along with the number of groups they're initially assigned. A DDM initialization Messenger reads the configuration file and initializes the data structures on each init node before any routing space virtual nodes are created or any regions are injected.

Figure 4-20 illustrates the grouping Messenger's process. If the grouping Messenger finds an unused group at the multicast server, it marks the group as taken to the requesting federate's machine and "carries" the group home. This is how groups migrate away from the multicast server. If the group is unused, there's no need to check for overflow and the group can be used immediately. Before a partially used group can be taken from another federate, it must be checked for overflow. The grouping Messengers carries the connection weight and connection receivers with it. The current group weight

and members is always stored with the group at its current init node. All of this information is consistent with the IRLOC algorithm and is always up to date. However, the IRLOC algorithm also makes use of the current incoming connection weights of both the current group members and the connection's receivers. Here is where slightly degraded information is used. Instead of having the current incoming weights of all the connection's receivers, the grouping Messenger carries the last known, largest incoming weight of all the receivers. The incoming weights of receivers are piggybacked on subscription regions, so they may be out of date due to subsequent subscriptions. Instead of the current incoming weights of all the group's members, the group is stored with the last known, largest incoming weight of any of all the group's members.

If the grouping succeeds, the group's weight and member list is updated. As illustrated in Figure 4-21, the grouping is reported to the requesting federate which changes its connectivity and adjusts its outgoing connection weight down. It also injects "join" Messengers for all the connection receivers who were not previously members of the group. In this system this Messenger only informs the receiver to adjust its incoming weight upward to account for other traffic from the group and to add to a reference count for this group. If multicast hardware were available, this Messenger would also be the trigger for the receiver to issue the appropriate system calls to join the multicast group.

Figure 4-22 shows the steps for dropping a connection and resigning from a group. When a sending federate discovers that connectivity has been lost between one of its regions and another federate, it injects a "resign" Messenger destined for the receiver. The

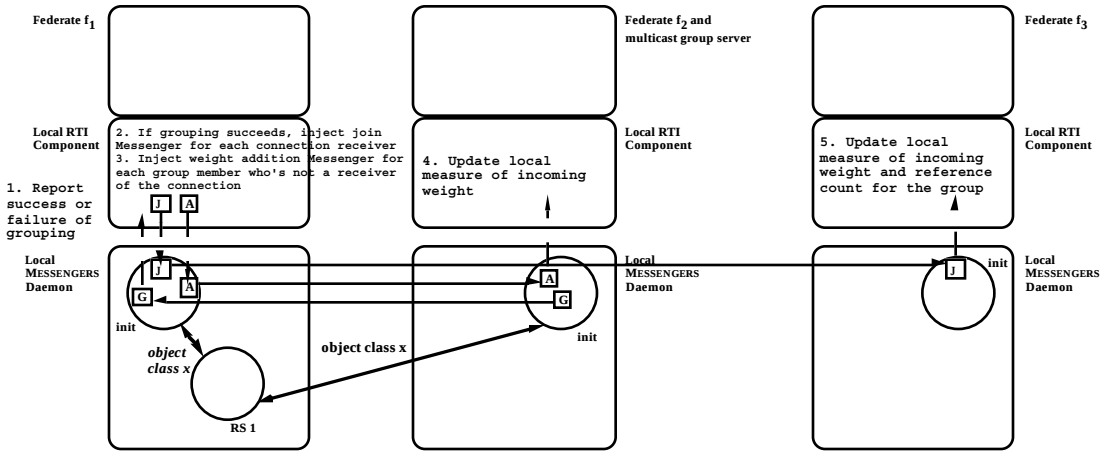

Figure 4-21. Report Success or Failure of Grouping

receiving federate decrements its reference count for the indicated group, and only actually leaves the group when the reference count reaches zero. Leaving a group is accomplished by a Messenger which travels to the multicast server to find the last location of the group. The Messenger hops forward according to the last location links until it finds the node holding the group where the Messenger removes the receiver from the group. If the Messenger removes the last federate from the group, it carries the group back to the multicast server and resets the group to its initial state so it can be reused. At the same time, the sending federate checks whether the loss of connectivity has reduced the connection's receiver set to one. If this is the case, the connection is reset to point-to-point. Through these two mechanisms, federates leave groups and connections leave groups when there is no longer any benefit to their membership. However, no attempt is made to reevaluate the cost function each time connectivity is lost because doing so would be computationally expensive and would require keeping complete knowledge of the group's connection set.

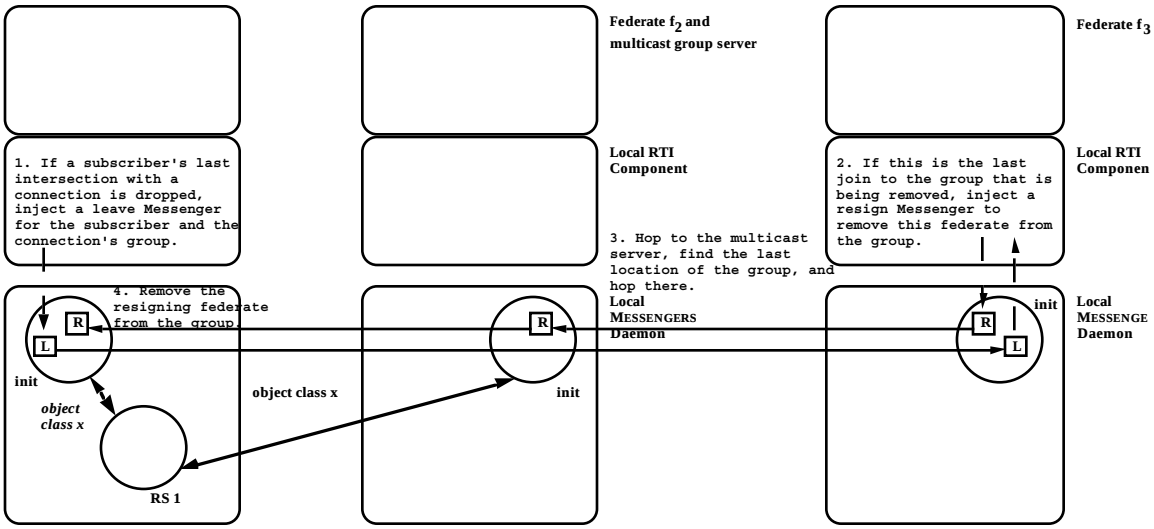

Figure 4-22. Dropping Connections and Resigning from Groups

4.4.1 Testing the Online Grouping Algorithm

There are two questions to be answered about the online grouping algorithm. Does it produce results comparable to the IRLOC algorithm? How much time does it take to do so? The first question is answered by testing it statically against the same connection sets as the IRLOC algorithm. The second question is answered by testing it dynamically against the baseline prototype and RTI 1.3.

4.4.1.1 Comparison of the Online Grouping Algorithm to IRLOC

Since the online grouping algorithm can only be run in real time with the MESSENGERS system underneath, this comparison test required manually generating regions and DDM API calls whose resulting region intersections produce the connection sets listed in Table 4-1. The groupings produced in this way were manually edited into connection set files and run through the offline simulator. These connection sets and the results of the offline simulator are included in Appendix A. In the online configuration there is no way to create the entire connection set statically. As soon as a connection or

connections are detected at any federate, the RTI component at that federate triggers grouping[3]. This required writing auxiliary Messengers which locate existing multicast groups and add new receivers to the group when they subscribe for a connection which has already been assigned to the group in question.

Even with degraded information about the input weights of the federates, the online grouping algorithm compared quite favorably to the offline IRLOC algorithm. In over half the cases, 17, the online algorithm generated the same solution as IRLOC. In six of the cases, it generated a better average message delivery time. In one case, the degraded information about input weights caused the online algorithm to generate too conservative a solution. In two cases, the order in which connections were discovered affected the order in which they were added to groups, i.e. early addition of a connection prevented later addition of a different connection which would have produced a better result. In one case, the fact that the grouping Messenger was restricted to only looking one hop from the multicast server prevented it from putting a potential connection into an existing group. In three cases, the grouping was affected by both of the previous two factors, ordering and restricted hops. Figure 4-23 through Figure 4-25 show the average message delivery times for the groupings generated by the online grouping algorithm.

3. Extra special thanks to DLD for the test harness that got all the Messengers daemons up and running at the same time and the scripter that let me run all the tests by just pressing return and watching the results scroll by.

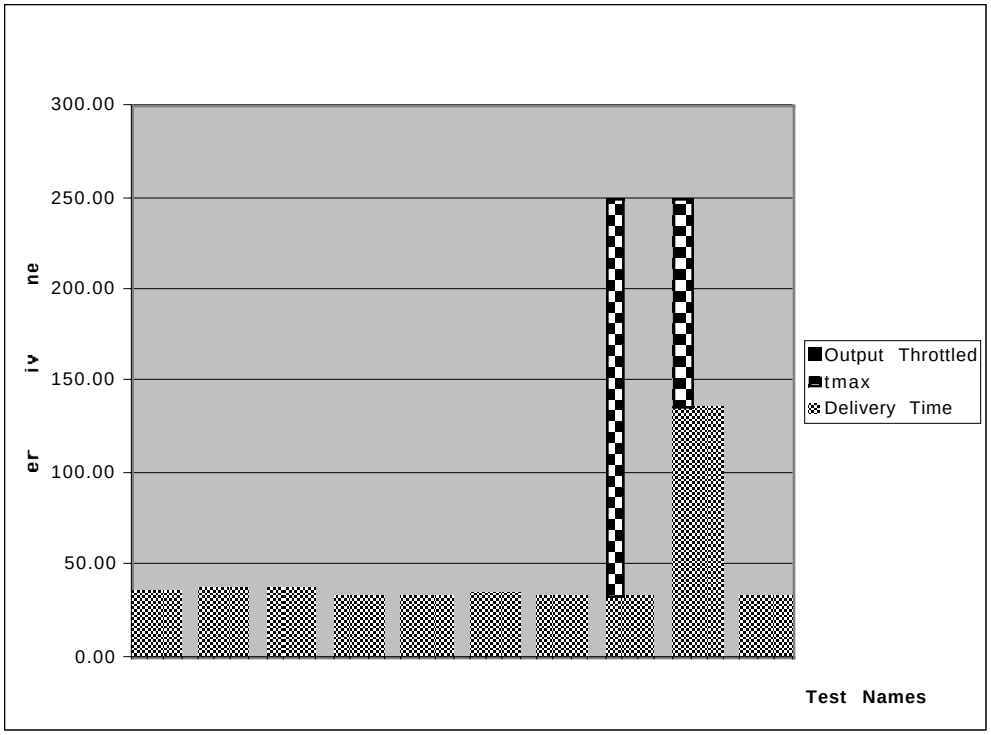

Figure 4-23. 5 Connections, 2 Groups, Online Grouping

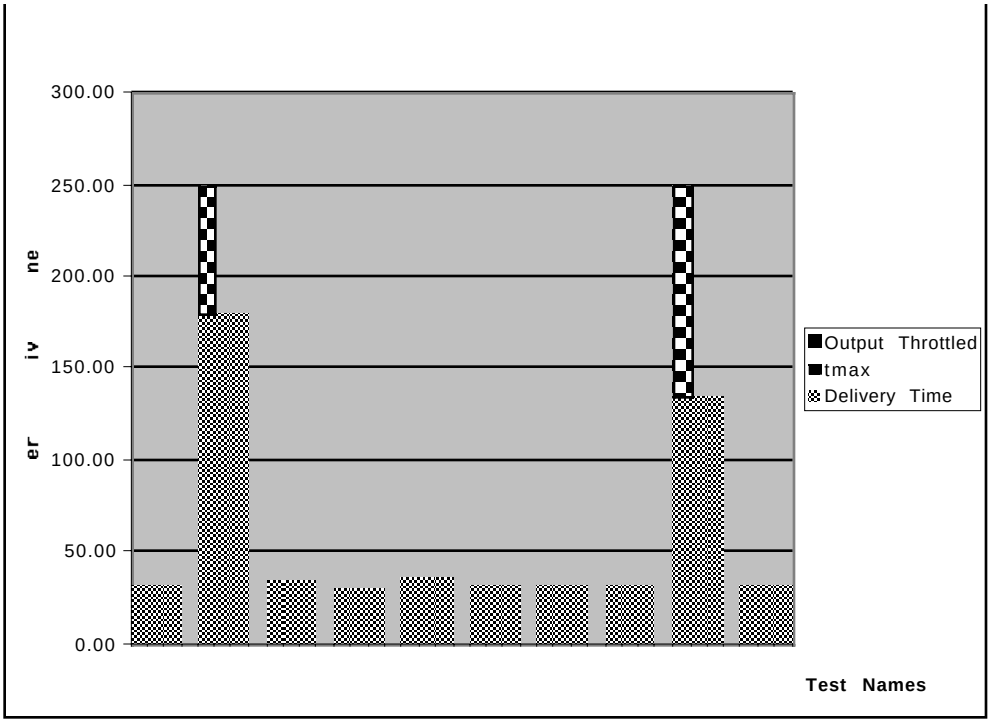

Figure 4-24. 5 Connections, 3 Groups, Online Grouping

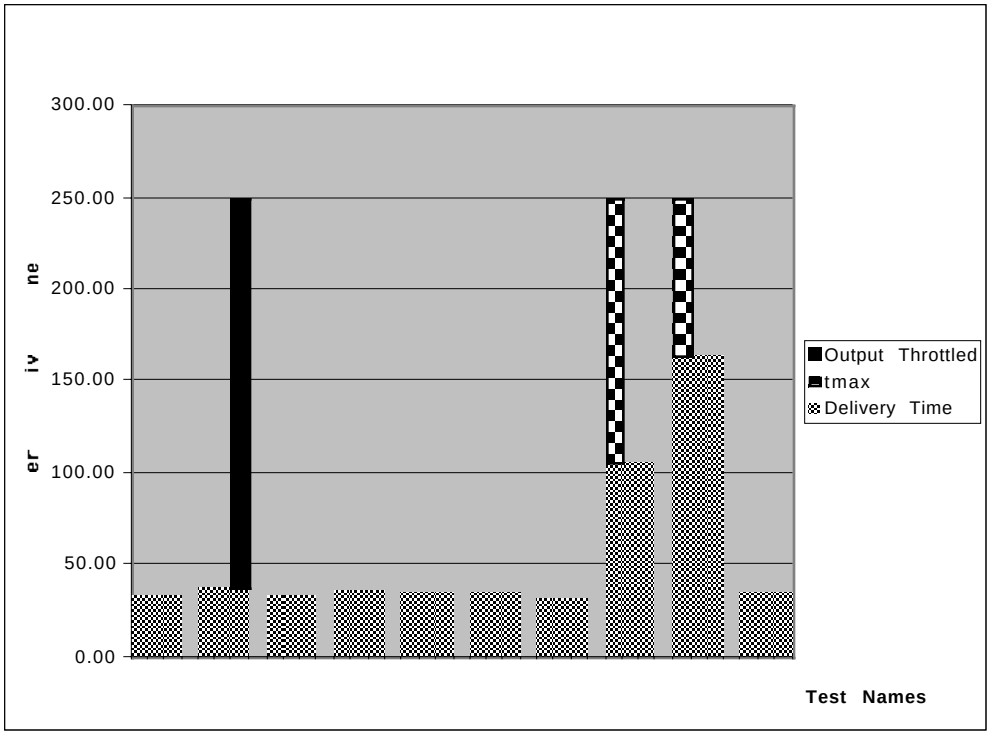

Figure 4-25. 6 Connections, 2 Groups, Online Grouping

4.4.1.2 Comparing the Online Grouping Algorithm to the Baseline Prototype and RTI 1.3

The experiments described in Table 4-5 are designed to test the hypothesis in Section 3.2 using relative measures between the online grouping algorithm, the baseline prototype, and an actual RTI implementation with DDM, RTI 1.3. The experiments use the benchmark algorithm described in Section 3.3. When interpreting the results, it's critical to remember that the baseline prototype and the online grouping algorithm implement the barest minimum of HLA functionality necessary to test the hypothesis with almost no error checking. RTI 1.3 is a robust, fully-compliant HLA 1.3 implementation with all the specified service groups and error checking, and the overhead implied by that . [4]

Table 4-5: RTI Timing Experiments

#	(federates, regions)[3]	Δr/minute	i	Δi/minute
1	(2,50), (5, 200), (10,1000)	0	r/25, r/10	0
2	(2,50), (5, 200), (10,1000)	r	r/25, r/10	0
3	(2,50), (5, 200), (10,1000)	r	r/50, r/25	i

Experiment 1 tests the impact of intersection calculations on initialization time. It establishes a basis for projection of performance of the online grouping algorithm in an actual implementation. The baseline prototype and RTI 1.3 are used because the online grouping algorithm is built on top of the baseline prototype, while the baseline prototype has an architecture for DDM which closely models the architecture of RTI 1.3. The average per federate initialization times for each of the federates in the RTI 1.3 tests are listed in Table 4-6. Separate tests verified that the growth in initialization times is due

Table 4-6: Initialization Times

f	r	Δr/minute	i	Δi/minute	RTI 1.3 (seconds)	Change for Grouping
2	50	0	2	0	13.190171	.012
2	50	0	5	0	13.103029	.008881
5	200	0	8	0	24.895595	.015031
5	200	0	20	0	24.134336	.010996
10	1000	0	40	0	132.050392	.009019
10	1000	0	100	0	129.133358	.005554
Averages					56.251147	.010247

primarily to a larger number of federates, not to a larger number of regions. The change in initialization time for grouping was calculated as the difference in initialization time

4. Federates are allocated one per workstation. Regions are uniformly distributed to federates, i.e. r/f per workstation.

between the on line grouping algorithm and the baseline prototype. Given that average per federate initialization time increase is more than three orders of magnitude smaller than the initialization time without grouping, using grouping has no appreciable impact on initialization.

Experiment 2 tests the impact of region changes without any intersection changes. As discussed in Section 3.2, this should impact CPU usage, but not severely since the system should recognize that connectivity hasn't changed. Since regions are uniformly assigned to federates, Δr per federate = r/f which ranges from 25 to to 100. Here the methodology is to determine if the RTI can do its job without robbing the federates of the CPU cycles they need to do their job. Although the Sun Sparc 5s used in the experiment are slightly underpowered compared to platforms typically used for HLA-based simulations, the RTI performed fairly well. Each experiment is run for 7 minutes with 10 loops per second for a total of 4200 loops. During each loop, the federate code performs all the calculations it requires and the remainder of the time in the loop allocated for the RTI to perform its functions. A "bad" loop is one in which the RTI fails to complete all its processing in the remaining loop time allocated to it. Across all six tests in experiment 2, the RTI only suffered an average of 2.6% bad loops. Running the benchmark algorithm with the online grouping algorithm and the baseline grouping algorithm only resulted in an average of 2173 μsec more time taken by the RTI across a 7 minute period; approximately .5 μsec per .1 sec loop. That's less time than it takes to receive a single extraneous message that would have been delivered without multicast!

Experiment 3 tests the impact of region changes with intersection changes. This should impact CPU usage more severely than experiment 2 since connectivity changes must be made. Predictably, the RTI produced more bad loops for experiment 3 than for experiment 2, 5.7% vs. 2.6%. However, the grouping algorithm only resulted in an average of 384 μsec more time. The fact that this is lower than the time for experiment 2 are initially surprising, but the numbers are so small compared to the measurable resolution that even small perturbations in the CPU load or network load on these non-dedicated machines can result in proportionally large differences.

For the sake of completeness, the additional time for all the experiments with the on line grouping algorithm and the baseline algorithm were recorded and averaged. The average additional time was 2596 μsec or .62 μsec per loop. All of this is overshadowed by the time it takes to reconfigure multicast groups in routers. According to [IETF97] and [Cisco99], joining a multicast group across a LAN can take no time at all, while leaving a multicast group across a WAN may take on the order of 260 seconds. In summary, it is not the time it takes to calculate the multicast groups which is the impediment to dynamic multicast grouping as has been asserted in the past, but the time it takes to change the groups in the routers.

4.5 Research Contributions

In summary, this research makes the following contributions to the state of interest management systems:

- Multicast grouping heuristic algorithms;
- Quantitative characterization system for scenarios with respect to Data Distribution Management;
- Definition of the cost function for evaluating additions to multicast groups;
- Identification of the quantitative characterization of scenarios for which the grouping algorithm improves filtering efficiency without excessive overhead;
- User-controllable benchmark algorithm which exercises quantitative characterizations for performance analysis of DDM implementations.

Each of these elements can be integrated with current interest management systems, specifically the data distribution management services in the High Level Architecture as illustrated physically and logically in Figure 4-26. More importantly, together they provide the capability to build a flexible interest management system which can be tuned based on simple, coarse measures of scenarios which users can reasonably expect to know.

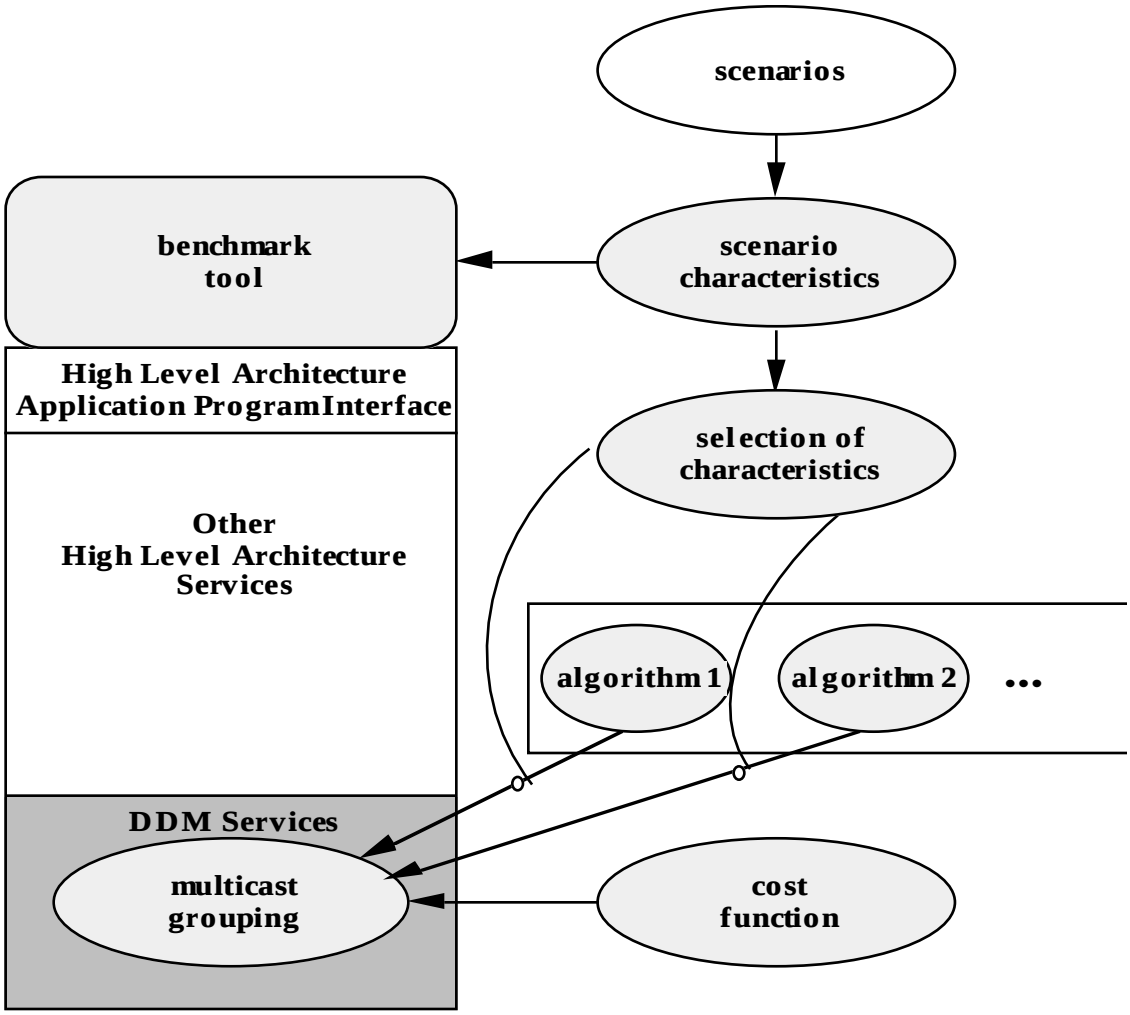

Figure 4-26. Physical and Logical Relationship of Research Components to DDM in the HLA

CHAPTER 5: Future Work

5.1 Load Balancing

All interest management systems must balance the costs of interest management against its benefits. If the overhead for interest management, in terms of CPU time which translates to latency of message delivery, is higher than what it would cost for the receiver to discard irrelevant messages itself, there is no benefit to interest management. However, this rule is only hard and fast if the time being spent on interest management could reasonably be spent by the simulator itself as is the case when the interest management infrastructure is co-resident on a system with the simulation. If the overhead can be offloaded to another system, the individual simulators can benefit from interest management even if the total overhead for the simulation as a whole is higher.

Future work could include investigating the benefit of load balancing interest management overhead using the MESSENGERS' system capability to migrate virtual nodes between physical nodes. This capability is above and beyond MESSENGERS' ability to migrate individual Messengers. Recall that each virtual node represents the publication of a single object class for a single federate. When the number of local virtual nodes or the number of regions hosted on individual local virtual nodes exceeds the local CPU's ability to process region intersections while still meeting message processing timing constraints, the virtual nodes can be moved to another system. The virtual links are elastic; they remain attached to the correct virtual node even if the virtual node changes physical nodes. So, the routing of region Messengers is not disrupted by the migration of

the virtual node. Auxiliary Messengers could explore latency to and load at neighboring physical nodes.

Unlike most other load balancing applications, load balancing the virtual nodes is constrained by the fact that migrating a virtual node far from its original physical node may have adverse effects on the ability of the interest manager to establish connectivity between federates in a timely manner. Virtual nodes should migrate to balance CPU load against the latency with the “home” node of the update regions. This work would include experimenting with the frequency of load balancing and the number of network hops separating the migrated virtual node from its “home” node.

This is based on the hypothesis that load balancing intersection calculations “away” from senders would produce performance benefits when Δi is low because changes have to be reported to the sender. A test for this would compare the baseline prototype against the load-balancing prototype for low and high values of Δi. This should result in balanced CPU loading, but higher time to establish connectivity. An experiment to test this hypothesis is detailed in Table 5-1. As with multicast grouping, this experiment should demonstrate values of load balancing parameters, such as frequency and number of network hops, which a user could tune.

Table 5-1: Load-Balancing Experiment

System	f	Δr/ minute	i	Δi/ minute
B, R, L	(2,50), (5, 200), (10,1000)	r	r/50, r/10	i, 2i

5.2 Varying t_s and t_r

The calculations of positive and negative weights, as well as t_{max}, are considerably simplified by the assumptions that t_s and t_r are equal, and that they are the same for all federates in the federation. The latter assumption is a generalization of the former. Relaxing these assumptions does not change the fundamental structure of the IRLOC algorithms, but it does simplify the mathematics. The effect of relaxing them could be achieved by multiplying the connection weights by the ratio of each federate's speed to that of the fastest federate, e.g. if one federate takes twice as long to send an update as the fastest federate, all its outgoing connection weights should be multiplied by two. This same calculation can be used to account for different size updates.

5.3 Measuring Weights

The system as described so far assumes that update frequency information is available for calculating weights. Such information would typically be found in a Federation Execution Planner's Workbook [Dahmann97]. However, preparation of such a workbook is not required to develop a federation. Nor is it a guarantee that an object's update frequency will not change during the course of federation execution. Measuring both static and dynamically changing update frequencies, and adjusting multicast groups accordingly is an area warrants further investigation.

5.4 Maintaining Incoming Weight Information

The offline grouping algorithms always have perfect global knowledge of the incoming weights of all federates. And the algorithms update this information as they build groups. The distributed online grouping algorithm has a much less consistent view of this information. A receiver could form many new connections and be added to other groups in between the time when it sends its incoming weight to a sender with its subscription region and the time when the sender begins to form a group. The accuracy of the offline grouping algorithm could be improved by periodically updating incoming weight information to receivers. One possibility is to keep track of the incoming weight sent with each subscription region as well as the lowest reported such weight. When the difference between the lowest reported weight and the current incoming weight reaches a threshold, an auxiliary Messenger with a new incoming weight would be dispatched to update all senders holding the subscription region. The threshold would have to be determined experimentally, trading off the cost of the additional overhead of tracking reported incoming weights against the cost of the extraneous messages resulting from inaccurate incoming weight information.

5.5 Ungrouping

The focus of this research has been on developing groups. Connections are dropped from groups and groups are dissolved when all the receivers have dropped out. However, the departure of only one or two receivers may adversely affect the effectiveness of the grouping. In the extreme case the cost function could be recalculated every time a receiver drops out. Doing so would require keeping complete information about the

connections in the set, including the identities of all the required receivers of each connection. The recalculation would be very expensive. Like the grouping algorithms, ungrouping could almost certainly benefit from heuristic approaches.

5.6 Accounting for and Measuring t_p

While t_p is a component of the average message delivery, none of the grouping algorithms described accounts for it. For example, if a connection had such a large value of t_p that any value of t_{ds} greater than t_s would cause some receivers to exceed t_{max}, the connection should be put into a group immediately. Furthermore, the addition of any other connection to the group which would increase t_q would have to be precluded.

It was also assumed that t_p was known and unvarying. In practice, neither is the case. t_p would have to measured between each pair of federates both at federation initialization and periodically during federation execution. As with incoming weights, the frequency of measuring t_p would have to balance the cost of the taking the measurement against the cost of less effective groupings resulting from obsolete measurements.

5.7 The Real World

Finally, the true test of this research would be to implement the distributed IRLOC algorithm in a production RTI and test it in a very large scale, dynamic virtual environment. While there has been some discussion of incorporating multicast grouping into the UK RTI [Hoare97], no concrete plans exist for doing so.

REFERENCES

[3Com] 3Com Corporation. Scaling Performance and Managing Growth with the CoreBuilder 3500 Layer 3 Switch. Available at http://www.3com.com/products/dsheets/400347a.html.

[Abrams99] Howard Abrams. Extensible Interest Management for Scalable Persistent Distributed Virtual Environments. Ph.D. Dissertation, Naval Postgraduate School, December 1999.

[ADST92] Advanced Distributed Simulation Technology Program Office. Strawman Distributed Interactive Simulation Architecture Description Document. Volume 1, Loral Systems Company, March 1992.

[Aguilera98] Marcos Aguilera, Robert E. Stron, Daniel C. Sturman, Mark Astley and Tushar D. Chandra. Matching Events in a Content-based Subscription System. In Proceedings of Principles of Distributed Computing 1999, Atlanta, GA, 1999.

[ALSP97] ALSP Architects. Personal Communication, November 1997.

[Banavar99] Guruduth Banavar, Tushar Chandra, Bodhi Mukherjee, Jay Nagarajarao, Robert E. Strom and Daniel C. Sturman. An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. In Proceedings of the International Conference on Distributed Computing Systems 1999, Austin, TX, 1999.

[Benford94] Steve Benford, Lennart E. Fahlen, and John Bowers. Supporting Social Communication Skills in Multi-Actor Artificial Realities. In Fourth International Conference on Aritficial Reality and Tele-Existence, pages 205-226, July 1994.

[Brutzman95] Donald P. Brutzman, Michael R. Macedonia, Michael J. Zyda. Internetwork Infrastructure Requirements for Virtual Environments. In Proceedings of the Virtual Reality Modeling Language (VRML) Sumposium, San Diego, CA, pages 95–104, 13 – 15 December 1995.

[Calvin95] James O. Calvin, Duncan C. Miller, Joshua Seeger, Gregory Troxel, and Daniel J. Van Hook. Application Control Techniques System Architecture. Technical Report RITN-1001-00, MIT - Lincoln Labs, February 1995.

[Calvin95a] James O. Calvin, Carol J. Chiang, and Daniel J. Van Hook. Data Subscription. In 12th Workshop on Standards for the Interoperability of Distributed Simulations, pages 807-813. March 1995.

[Bender91] Edward A. Bender and S. Gil Williamson. Foundations of Applied Combinatorics, Addison-Wesley, New York, 1991.

[Ballardie98] Tony Ballardie, Paul Francis, Jon Crowcroft. Core Based Trees (CBT) An Architecture for Scalable Inter-Domain Multicast Routing. In Proceedings of SIGCOMM ’93, pages 85-94, 1003.

[Bic96] Lubomir Bic, Munehiro Fukuda, and Michael Dillencourt. Distributed Computing using Autonomous Objects. IEEE Computer, 29(8), August 1996.

[Bic95] Lubomir Bic. Distributed Computing using Autonomous Objects. In 5th IEEE CS Workshop on Future Trends of Distributed Computing Systems, August 1995.

[Calvin95b] James O. Calvin, David P. Cebula, Carol J. Chiang, Steven J. Rak, and Daniel J. Van Hook. Data Subscription in Support of Multicast Group Allocation. In 13th Workshop on Standards for the Interoperability of Distributed Simulations, pages 367-369, September 1995.

[Calvin97] James O. Calvin, et al. Design, Implementation, and Performance of the STOW RTI Prototype (RTI-s). In Proceedings of the 1997 Spring Simulation InteroperabilityWorkshop, pages 145-154, March 1997.

[Capps99] Michael Capps, Kent Watsen and Michael Zyda. Cyberspace and Mock Apple Pie. In IEEE Computer Graphics and Applications, November/ December 1999.

[Cisco99] Cisco IOS 12.0 Solutions for Network Protocols Volume 1: IP. Cisco Press. 1999.

[Cohen98] D. Cohen and A. Kemkes. User-Level Measurements of DDM Scenarios, Proceedings of the 1998 Spring Simulation Interoperability Workshop, 98S-SIW-072, March 1998.

[Cohen98a] D. Cohen and A. Kemkes. Applying User-Level Measurements to RTI 1.3 Release 2. In Proceedings of the 1998 Fall Simulation Interoperability Workshop, 98F-SIW-132, September 1998.

[Cook97] Jon Cook, Roger Hubbold, and Martin Keates. Virtual Reality for Large-Scale Industrial Application. In Proceedings of Euro VR ‘97, pgates 157-166. November 1997.

[Dahmann97] Judith Dahmann, Ken Hunt, Robert Lutz and Jack Sheehan. In Proceedings of the 1997 Spring Simulation Interoperability Workshop, 97S-SIW-067, March 1997.

[DIS94] DIS Steering Committee. Standard for Information Technology - Protocols for Distributed Interactive Simulation, March 1994. IEEE Standard 1278.

[DIS94a] DIS Steering Committee. The DIS Vision: A Map to the Future of Distributed Simulation. Version 1, May 1994.

[DMSO98] Department of Defense High Level Architecture Interface Specification, Version 1.3, DMSO, April 1998, available at http://hla.dmso.mil.

[DMSO98a] Department of Defense High Level Architecture Rules, Version 1.3, DMSO, April 1998, available at http://hla.dmso.mil.

[DMSO98b] Department of Defense High Level Architecture Object Model Template, Version 1.3, DMSO, April 1998, available at http://hla.dmso.mil.

[Greenhalgh95] Chris Greenhalgh and Steve Benford. Virtual Reality Tele-conferencing: Implementation and Experience. In Proceedings of the Fourth European Conference on Computer Supported Cooperative Work, September 1995.

[Greenhalgh97] Chris Greenhalgh and Steve Benford. Boundaries, Awareness and Interaction in Collaborative Virtual Environments. InProceedings Sixth IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pages 193-198, June 1997.

[Greenhalgh98] Chris Greenhalgh. MASSIVE-2. Personal Communication, August 1998.

[Hagsand96] Olof Hagsand. Interactive Multiuser VEs in the DIVE System. In IEEE Multimedia, vol. 3, no. 1, January - March 1996.

[Hoare97] Peter Hoare, Gerard Magee and Ian Moody. The Development of a Prototype HLA Run Time Infrastructure (RTI-Lite) Using CORBA. In Proceedings of the 1997 Summer Computer Simulation Conference, July 1997 July, Arlington, VA.

[Hoare98] Peter Hoare and Richard Fujimoto. HLA RTI Performance in High Speed LAN Environments. In Proceedings of the 1998 Fall Simulation Interoperability Workshop, pages 501-510, September 1998.

[Hockney98] Hockney and J.W. Eastwood. Computer Simulations using Particles, IOP Publishing Ltd. Bristol, Great Britain, 1988.

[Hubbold96] Roger Hubbold, Xiao Dongbo and Simon Gibson. MAVERIK – the Manchester Virtual Environment Interface Kernel. In Proceedings of the Third Eurographics Workshop on Virtual Environments, February 1996.

[Hubbold98] Roger Hubbold. MAVERIK. Personal Communication, October 1998.

[Huitema96] Christian Huitema. IPv6: The New Internet Protocol, Prentice-Hall, New Jersey, 1996, pg. 46.

[IETF97] IETF Network Working Group. Internet Group Management Protocol, Version 2, RFC 2236. Available at http://rfc.fh-koeln.de/rfc/html/rfc2236.html, November 1997.

[Langston94] Langston. Artificial Life III, Addison Wesley, 1994.

[Lewis98] Harry R. Lewis and Christos H. Papadimitrou. Elements of the Theory of Computation, Prentice-Hall, New Jersey, 1998, pg. 340-346.

[Logi99] Filippo Logi. CARTESIUS: A Cooperative Approach to Real-Time Decision Support for Multi-Jurisdictional Traffic Congestion Management. Ph.D. Dissertation, University of California, Irvine, 1995.

[Macedonia94] Macedonia, Michael R., Zyda, Michael J., Pratt, David R., Barham, Paul T. and Zeswitz, Steven. NPSNET: A Network Software Architecture for Large Scale Virtual Environments. Presence, Vol. 3, No. 4, Fall 1994, pp.265-287.

[Macedonia95] Michael Macedonia, Michael Zyda, David Pratt, and Paul Barham. Exploiting Reality with Multicast Groups: a Network Architecture for Large Scale Virtual Environments. In Virtual Reality Annual International Symposium ‘95, pages 2-10, March 1995.

[Macedonia95a] Michael R. Macedonia, Donald P. Brutzman, Michael J. Zyda, David R. Pratt, Paul T. Barham, John Falby, and John Locke. NPSNET: A Multi-Player 3D Virtual Environment Over the Internet. In Proceedings of the 1995 Symposium on Interactive 3D Graphics, pages 93–94, Monterey, CA, 9 – 12 April 1995.

[Macedonia95b] Michael R. Macedonia. A Network Software Architecture for Large Scale Virtual Environments. Ph.D. Dissertation, Naval Postgraduate School, June 1995.

[Macedonia97] Michael Macedonia and Michael Zyda. A Taxonomy for Networked Virtual Environments. In IEEE Multimedia, vol. 4, no. 1, pages 48-56, January - March 1997.

[MacDonald79] V.H. MacDonald. The Cellular Concept. In The Bell System Technical Journal, pages 15-41, January 1979.

[Mastaglio95] Thomas W. Mastaglio and Robert Callahan. A Large-Scale Complex Virtual Environment for Team Training. IEEE Computer, 28(7), pages 49-56, July 1995.

[Morse96] Katherine L. Morse. Interest Management in Large Scale Distributed Simulations. UCI Technical Report #96-27.

[Morse97] Katherine L. Morse and Jeffrey S. Steinman. Data Distribution Management in the HLA: Multidimensional Regions and Physically Correct Filtering. In Proceedings of the 1997 Spring Simulation Interoperability Workshop, pages 343-352, March 1997.

[Morse97a] Katherine L. Morse, Dannie E. Cutts, John P. Hancock, Stephan A. Lubbers. Feasibility and Functionality of Autonomous Objects in the HLA. In Proceedings of the 1997 Spring Simulation Interoperability Workshop, pages 353-362, March 1997.

[Morse97b] Katherine L. Morse, Andreas Kemkes, and Mikel Petty. Issues in the Relationship between HLA's Declaration Management and Data Distribution Management Services. In Proceedings of the 1997 Fall Simulation Interoperability Workshop, pages 549-559, September 1997.

[Morse98] Katherine L. Morse and Judith Dahmann. The High Level Arhcitecture: An Update. In Proceedings of the 1998 Workshop on Distributed Interactive Simulation – Real Time, 1998.

[Morse99] Katherine L. Morse, Lubomir Bic, and Michael Dillencourt. Characterizing Scenarios for DDM Performance and Benchmarking RTIs. In Proceedings of the 1999 Spring Simulation Interoperability Workshop, March 1999.

[Morse99a] Katherine L. Morse, Lubomir Bic, and Michael Dillencourt. Multicast Grouping for Dynamic Data Distribution Management. In Proceedings of the 1999 Society for Computer Simulation Conference, July 1999.

[Morse00] Katherine L. Morse, Lubomir Bic, and Michael Dillencourt. Interest Mangement in Large Scale Virtual Environments. Forthcoming in MIT PRESENCE.

[NCCOSC95] Naval Command, Control and Ocean Surveillance Center. Synthetic Theater of War-Europe (STOW-E) Technical Analysis. NCCOSC, San Diego, CA, May 22, 1995.

[Papa94] Christos Papadimitriou. Computational Complexity, Addison-Wesley, Massachusetts, 1994, pg. 202.

[Pettifer97] Steve R. Pettifer. Deva: a Coherent Operating Environment for Large Scale Virtual Reality Applications. In Proceedings of Virtual Reality Universe 97. 1997.

[Pettifer98] Steve R. Pettifer. Deva. Personal Communication, October 1998.

[Petty97] Mikel D. Petty. Experimental Comparison of d-Rectangle Intersection Algorithms Applied to HLA Data Distribution. In Proceedings of the 1997 Distributed Simulation Symposium, pages 13-26, September 1997.

[Powell96] Edward T. Powell, Larry Mellon, James F. Watson, and Glenn H. Tarbox. Joint Precision Strike Demonstration (JPSD) Simulations Architecture. In 14th Workshop on Standards for the Interoperability of Distributed Simulations, pages 807-810, March 1996.

[Rak96] Steven J. Rak and Daniel J. Van Hook. Evaluation of Grid-Based Relevance Filtering for Multicast Group Assignment. In 14th Workshop on Standards for the Interoperability of Distributed Simulations, pages 739-747, March 1996.

[RDTE95] RDT&E Division and Advanced Telecommunications Inc. Synthetic Theater of War-Europe Technical Analysis. Naval Command, Control and Ocean Surveillance Center, May 1995.

[RDTE96] RDT&E Division. Engineering Demonstration #1 Bandwidth Analysis. Naval Command, Control and Ocean Surveillance Center, February 1996.

[Rogers95] Dennis Rogers. STOW-E Lessons Learned: Focused on the 3 Primary Army STOW-E sites. In 12th Workshop on Standards for the Interoperability of Distributed Simulations, pages 143-147, March 1995.

[Russo95] Kevin L. Russo, Lawrence C. Shuette, Joshua E. Smith, and Matthew E. McGuire. Effectiveness of Various New Bandwidth Reduction Techniques in ModSAF. In 13th Workshop on Standards for the Interoperability of Distributed Simulations, pages 587-591, September 1995.

[Russo95] Kevin L. Russo, Lawrence C. Shuette, Joshua E. Smith, and Matthew E. McGuire. Effectiveness of Various New Bandwidth Reduction Techniques in ModSAF. In 13th Workshop on Standards for the Interoperability of Distributed Simulations, pages 587-591, September 1995.

[SAIC99] SAIC. Integrated Theater-Level Engagement Model (ITEM) User's Manual version 8.3. SAIC, 10260 Campus Point Drive, San Diego, CA, October 25, 1999.

[Singhal99] Sandeep Singhal and Michael Zyda. Networked Virtual Environments - Design and Implementation, ACM Press Books, SIGGRAPH Series, 23 July 1999, ISBN 0-201-32557-8, 315 pages.

[Smith95] Joshua Smith, Kevin L. Russo, and Lawrence C. Schuette. Prototype Multicast IP Implementation in ModSAF. In 12th Workshop on Standards for the Interoperability of Distributed Simulations, pages 175-178, March 1995.

[Steinman92] Jeff S. Steinman. SPEEDES: A Multiple-Synchronization Environment for Parallel Discrete-Event Simulation. In International Journal in Computer Simulation, vol. 2, pages 251-286, 1992.

[Steinman94] Jeff S. Steinman and Frederick Wieland. Parallel Proximity Detection and the Distribution List Algorithm. In Proceedings of the 1994 Workshop on Parallel and Distributed Simulation, pages 3-11, July 1994.

[Steinman96] Jeff S. Steinman. Proximity Detection Interest Management. Personal Communication, February 1996.

[VanHook94] Daniel J. Van Hook and James O. Calvin. AGENTS: An Architectural Construct to Support Distributed Simulation. In 11th Workshop on Standards for the Interoperability of Distributed Simulations, pages 357-362, September 1994.

[VanHook94a] Daniel J. Van Hook, James O. Calvin, Michael K. Newton, and David A. Fusco. An Approach to DIS Scaleability. In 11th Workshop on Standards for the Interoperability of Distributed Simulations, pages 347-356, September 1994.

[VanHook94b] Daniel J. Van Hook, Steven J. Rak and James O. Calvin. Approaches to Relevance Filtering. In 11th Workshop on Standards for the Interoperability of Distributed Simulations, pages 367-369, September 1994.

[VanHook96] Daniel J. Van Hook. RITN IM and IM History. Personal Communication, January 1996.

[Vrablik95] Rob Vrablik. ModSAF Interest Management Overview. Personal Communication, November 1995.

[VanHook98] Daniel J. Van Hook and James O. Calvin. Data Distribution Management in RTI 1.3. In Proceedings of the 1998 Spring Simulation Interoperability Workshop, pages 1167-1175, March 1998.

[Watsen98] Kent Watsen and Michael Zyda. Bamboo - A Portable System for Dynamically Extensible, Real-Time, Networked, Virtual Environments. In Proceedings of VRAIS 98, pages252-259, 16 - 19 March 1998.

[Watson95] James F. Watson. JPSD Interest Management Overview. Personal Communication, November 1995.

[Wilson94] Annette Wilson and Richard Weatherly. New Traffic Reduction and Management Tools for ALSP Confederations. In 1994 Elecsim Internet Conference, April 1994. http://alsp.ie.org/alsp

[Zeigler97] Bernard P. Zeigler, Yoonkeon Moon, Doohwan Kim, and George Ball. The DEVS Environment for High-Performance Modeling and Simulation. In IEEE Computational Science and Engineering, pages 61-71, July-September 1997.

[Zyda97] “Modeling and Simulation: Linking Entertainment & Defense.” Editors: Michael Zyda and Jerry Shehan. National Academy Press, September 1997, ISBN 0-309-05842-2, 181 pages.

APPENDIX A: Offline Simulator Test Inputs and Results

The collected results, raw outputs, and inputs from the offline simulation of the thirty test cases described in Section 4.3.1.1 and Section 4.3.1.2 are provided below. Each of the test cases simulated 10 federates running for 10 seconds with t_p = 10 ms between each pair of federates and $t_s = t_r$ = 10 ms at all federates. In all of the tables, PP indicates point-to-point, BR indicates broadcast, LOC indicates largest outgoing connection algorithm, and IRLOC indicates input-restricted largest outgoing connection.

The format of the connection set inputs to the offline simulator is

```
sending_federate_ID #_of_receivers receiver_list multicast_group
```

where a multicast group of zero indicates that the connection is sent point-to-point.

5 connections, 2 groups

seed	measure	PP	BR	LOC	IRLOC	ON	comments
1	out. throt.	1	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	58.22	32	32.32	32.91	36.13	
	% extra	-33%	50%	32%	0%	0%	
2	out. throt.	2	0	0	0	0	4 federates' input weights are at least 100
	tmax	1	7	0	0	0	
	avg. delivery	56.85	670.14	37.23	37.23	37.23	
	% extra	-62%	37%	-13%	-13%	-13%	
3	out. throt.	0	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	43.68	32.08	32.07	37.25	37.25	
	% extra	0%	205%	102%	0%	0%	
4	out. throt.	1	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	51.1	31.5	31.3	33.05	33.05	

	% extra	-39%	101%	32%	0%	0%	
5	out. throt.	1	0	0	0	0	
	tmax	1	0	0	0	0	
	avg. delivery	61.85	36.51	36.88	33.6	33.58	
	% extra	-51%	30%	0%	8%	-5%	
6	out. throt.	0	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	46.24	32.22	32.03	34.82	34.69	
	% extra	0%	343%	252%	0%	1%	
7	out. throt.	1	0	0	0	0	Even with multicast, federate 2 is output throttled from multiple connections
	tmax	1	0	0	0	0	
	avg. delivery	59.68	32.69	33.02	33.02	33.02	
	% extra	-68%	32%	-16%	-16%	-16%	
8	out. throt.	1	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	52.17	34.93	34.5	33.33	33.33	
	% extra	-60%	94%	61%	-3%	-3%	
9	out. throt.	2	0	0	0	0	2 federates' input weights are at least 100
	tmax	1	6	1	1	1	
	avg. delivery	61.76	539.19	136.38	134	136.38	
	% extra	-54%	9%	-3%	-2%	-3%	
10	out. throt.	0	0	0	0	0	
	tmax	2	0	0	0	0	
	avg. delivery	64.7	31.66	31.55	31.14	32.97	
	% extra	-6%	29%	21%	4%	0%	

Offline simulator raw output

random_5c_1s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.64	275	30.64	0	0.02	12	0.14	40	1	0
Federate 1:	0.64	275	40.64	0	0.02	22	0.08	40	1	0
Federate 2:	0.00	125	50.00	0	0.00	0	0.00	25	1	0
Federate 3:	0.26	195	56.67	0	0.00	16	0.03	32	1	0
Federate 4:	1.56	375	54.76	0	0.06	64	0.09	50	1	0
Federate 5:	0.26	194	76.65	0	0.00	52	0.01	32	1	0
Federate 6:	0.54	224	78.93	0	0.01	69	0.02	35	1	0
Federate 7:	0.00	50	80.00	0	0.00	0	0.00	5	0	1
Federate 8:	0.57	294	77.78	0	0.02	98	0.02	42	1	0
Federate 9:		0								
	0.66	2007	58.22	0	0.01	33	0.04	301		

random_5c_2s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.71	311	30.71	0	0.02	11	0.20	61	1	1
Federate 1:	1.07	400	38.60	0	0.04	38	0.11	110	2	0
Federate 2:	0.52	211	41.04	0	0.01	9	0.11	50	1	0
Federate 3:	0.75	320	47.69	0	0.02	40	0.06	101	2	0
Federate 4:	0.00	110	50.00	0	0.00	1	0.00	11	0	1
Federate 5:	1.05	408	56.69	0	0.04	80	0.05	111	2	0
Federate 6:	1.03	220	71.49	0	0.02	41	0.05	51	1	0
Federate 7:	1.06	319	75.16	0	0.03	99	0.03	101	2	0
Federate 8:	1.03	220	91.49	0	0.02	88	0.03	51	1	0
Federate 9:	0.00	110	110.00	0	0.00	98	0.00	40	1	0
	0.84	2629	56.85	0	0.02	50	0.06	687		

random_5c_3s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.21	80	31.21	0	0.01	1	0.97	8	0	0
Federate 1:	0.00	60	40.00	0	0.00	2	0.00	6	0	0
Federate 2:	0.00	30	43.33	0	0.00	1	0.00	3	0	0
Federate 3:	0.00	10	60.00	0	0.00	0	0.00	1	0	0
Federate 4:	1.29	180	41.29	0	0.02	5	0.39	18	0	0
Federate 5:	0.48	130	46.63	0	0.01	4	0.12	13	0	0
Federate 6:	0.00	50	42.00	0	0.00	0	0.00	5	0	0
Federate 7:	0.00	20	70.00	0	0.00	0	0.00	2	0	0
Federate 8:	0.00	90	51.11	0	0.00	4	0.00	9	0	0
Federate 9:		0								
	0.60	650	43.68	0	0.00	2	0.15	65		

random_5c_4s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.03	330	30.03	0	0.00	13	0.01	53	1	0
Federate 1:	0.03	229	40.03	0	0.00	10	0.01	43	1	0
Federate 2:	0.00	239	46.65	0	0.00	0	-0.00	44	1	0
Federate 3:	0.39	269	56.30	0	0.01	30	0.03	47	1	0
Federate 4:	1.95	50	53.95	0	0.01	1	0.97	5	0	0
Federate 5:	1.39	70	62.82	0	0.01	4	0.22	7	0	0
Federate 6:	1.39	70	72.82	0	0.01	4	0.22	7	0	0
Federate 7:	0.00	229	71.75	0	0.00	99	0.00	43	1	0
Federate 8:	0.00	60	83.33	0	0.00	6	0.00	6	0	1
Federate 9:		0								

	0.27	1546	51.10	0	0.00	17	0.15	255		

random_5c_5s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.50	382	31.50	0	0.06	22	0.25	79	2	0
Federate 1:	1.25	342	41.25	0	0.04	30	0.14	75	2	0
Federate 2:	0.56	242	50.56	0	0.01	24	0.06	65	2	0
Federate 3:	1.17	342	58.25	0	0.04	57	0.07	75	2	0
Federate 4:	1.55	442	60.24	0	0.07	91	0.08	85	2	0
Federate 5:		0								
Federate 6:	1.42	381	74.62	0	0.05	98	0.05	79	2	0
Federate 7:	1.59	440	77.98	0	0.07	99	0.07	85	2	0
Federate 8:	0.71	310	81.39	0	0.02	99	0.02	45	1	1
Federate 9:	0.00	110	110.00	0	0.00	98	0.00	25	1	0
	1.23	2991	61.85	0	0.04	62	0.07	613		

random_5c_6s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.83	140	30.83	0	0.01	3	0.32	14	0	0
Federate 1:	1.93	150	41.26	0	0.03	6	0.46	15	0	0
Federate 2:	0.00	60	48.33	0	0.00	2	0.00	6	0	0
Federate 3:	0.10	70	57.24	0	0.00	2	0.02	7	0	0
Federate 4:	0.35	20	60.35	0	0.00	1	0.07	2	0	0
Federate 5:	0.14	70	70.14	0	0.00	2	0.05	7	0	0
Federate 6:	1.08	599	35.42	0	0.06	51	0.13	60	0	0
Federate 7:	1.57	70	84.43	0	0.01	2	0.55	7	0	0
Federate 8:	0.27	140	73.84	0	0.00	15	0.02	14	0	0
Federate 9:		0								
	0.93	1319	46.24	0	0.01	8	0.16	132		

random_5c_7s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.64	267	30.64	0	0.02	8	0.20	70	1	0
Federate 1:	0.75	183	40.75	0	0.01	8	0.17	60	1	0
Federate 2:	0.00	100	50.00	0	0.00	0	0.00	10	0	1
Federate 3:	0.00	83	50.00	0	0.00	0	0.00	50	1	0
Federate 4:	1.51	213	43.20	0	0.03	15	0.20	63	1	0
Federate 5:	0.18	183	64.72	0	0.00	20	0.02	60	1	0
Federate 6:	0.61	265	64.35	0	0.02	65	0.02	70	1	0
Federate 7:	0.18	182	84.74	0	0.00	39	0.01	60	1	0
Federate 8:	0.24	265	80.85	0	0.01	99	0.01	70	1	0
Federate 9:	0.00	83	110.00	0	0.00	0	0.00	50	1	0
	0.50	1824	59.68	0	0.01	25	0.06	563		

random_5c_8s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	20	30.00	0	0.00	0	0.00	2	0	1
Federate 1:	0.18	168	31.38	0	0.00	2	0.11	53	1	0
Federate 2:	0.18	168	41.38	0	0.00	5	0.06	53	1	0
Federate 3:	0.00	138	50.00	0	0.00	0	-0.00	50	1	0
Federate 4:	1.03	518	40.18	0	0.05	72	0.07	92	1	0
Federate 5:	0.17	168	68.98	0	0.00	13	0.02	53	1	0
Federate 6:	0.80	337	59.55	0	0.03	85	0.03	72	1	0

Federate 7:	0.06	158	90.06	0	0.00	98	0.00	52	1	0
Federate 8:	0.00	20	100.00	0	0.00	0	0.00	2	0	0
Federate 9:		0								
	0.53	1695	52.17	0	0.01	27	0.03	429		

random_5c_9s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	6.84	671	36.84	0	0.46	58	0.79	120	2	0
Federate 1:	4.22	559	44.22	0	0.24	65	0.36	95	1	1
Federate 2:	1.77	471	49.41	0	0.08	64	0.13	100	2	0
Federate 3:	0.00	222	55.00	0	0.00	13	0.00	75	2	0
Federate 4:	1.46	421	59.37	0	0.06	66	0.09	95	2	0
Federate 5:	1.46	421	69.37	0	0.06	91	0.07	95	2	0
Federate 6:	1.47	421	74.65	0	0.06	99	0.06	95	2	0
Federate 7:	0.77	360	75.54	0	0.03	99	0.03	50	1	1
Federate 8:	3.62	271	98.09	0	0.10	99	0.10	80	2	0
Federate 9:	3.73	220	113.73	0	0.08	98	0.08	75	2	0
	2.90	4037	61.76	0	0.12	75	0.17	880		

random_5c_10s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.47	158	30.47	0	0.01	3	0.24	17	0	0
Federate 1:	1.14	198	39.12	0	0.02	8	0.26	21	0	0
Federate 2:	0.43	188	44.04	0	0.01	13	0.06	20	0	0
Federate 3:	0.05	130	46.98	0	0.00	10	0.01	13	0	0
Federate 4:	0.47	158	60.47	0	0.01	13	0.06	17	0	0
Federate 5:	1.14	198	69.12	0	0.02	32	0.07	21	0	0
Federate 6:	1.15	197	79.12	0	0.02	41	0.05	21	0	0
Federate 7:	0.48	157	90.48	0	0.01	34	0.02	17	0	0
Federate 8:	0.48	157	100.48	0	0.01	56	0.01	17	0	0
Federate 9:	0.00	87	110.00	0	0.00	37	0.00	10	0	0
	0.65	1628	64.70	0	0.01	25	0.08	174		

random_5c_1s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.87	500	31.87	0	0.09	40	0.23	50	0	0
Federate 1:	1.55	500	31.95	0	0.08	40	0.19	50	0	0
Federate 2:	0.61	330	30.61	0	0.02	9	0.20	33	0	0
Federate 3:	0.94	420	31.89	0	0.04	32	0.12	42	0	0
Federate 4:	1.45	500	32.65	0	0.07	44	0.16	50	0	0
Federate 5:	1.29	500	32.89	0	0.06	44	0.14	50	0	0
Federate 6:	1.35	500	33.35	0	0.07	48	0.14	50	0	0
Federate 7:	0.00	230	30.00	0	0.00	8	0.00	23	0	0
Federate 8:	1.37	500	33.77	0	0.07	48	0.14	50	0	0
Federate 9:		0								
	1.26	3980	32.32	0	0.05	31	0.13	398		

random_5c_2s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.96	598	30.96	0	0.06	55	0.10	61	0	0
Federate 1:	8.13	884	40.27	0	0.72	96	0.74	110	0	0
Federate 2:	1.33	498	31.33	0	0.07	24	0.27	50	0	0
Federate 3:	1.75	805	33.12	0	0.14	76	0.18	101	0	0

Federate 4:	0.00	110	50.00	0	0.00	1	0.00	11	0	0
Federate 5:	8.39	894	42.95	0	0.75	99	0.75	111	0	0
Federate 6:	0.67	508	37.36	0	0.03	53	0.06	51	0	0
Federate 7:	2.69	805	38.28	0	0.22	99	0.22	101	0	0
Federate 8:	1.74	507	42.69	0	0.09	78	0.11	51	0	0
Federate 9:	0.00	398	30.00	0	0.00	0	-0.00	40	0	0
	3.45	6007	37.23	0	0.21	58	0.24	687		

random_5c_3s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 1:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 2:	0.00	70	30.00	0	0.00	3	0.00	7	0	0
Federate 3:	0.00	70	30.00	0	0.00	3	0.00	7	0	0
Federate 4:	2.75	220	32.75	0	0.06	10	0.59	22	0	0
Federate 5:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 6:	0.15	50	30.15	0	0.00	1	0.07	5	0	0
Federate 7:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 8:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 9:		0								
	2.07	1310	32.07	0	0.03	5	0.28	131		

random_5c_4s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.31	529	30.31	0	0.02	32	0.05	53	0	0
Federate 1:	1.83	569	31.83	0	0.10	37	0.28	57	0	0
Federate 2:	1.79	559	31.79	0	0.10	34	0.29	56	0	0
Federate 3:	1.83	569	31.83	0	0.10	37	0.28	57	0	0
Federate 4:	0.31	150	30.31	0	0.00	3	0.15	15	0	0
Federate 5:	0.07	170	30.07	0	0.00	5	0.02	17	0	0
Federate 6:	0.00	70	30.00	0	0.00	1	0.00	7	0	0
Federate 7:	1.83	569	31.83	0	0.10	37	0.28	57	0	0
Federate 8:	0.07	170	30.07	0	0.00	5	0.02	17	0	0
Federate 9:		0								
	1.30	3355	31.30	0	0.04	19	0.14	336		

random_5c_5s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.93	788	32.93	0	0.23	90	0.26	79	0	0
Federate 1:	3.09	748	34.43	0	0.23	86	0.27	75	0	0
Federate 2:	1.61	648	31.61	0	0.10	65	0.16	65	0	0
Federate 3:	2.96	748	35.64	0	0.22	96	0.23	75	0	0
Federate 4:	3.16	848	36.70	0	0.27	99	0.27	85	0	0
Federate 5:		0								
Federate 6:	4.40	787	39.94	0	0.35	99	0.35	79	0	0
Federate 7:	3.91	847	40.93	0	0.33	99	0.33	85	0	0
Federate 8:	1.10	449	48.78	0	0.05	99	0.05	45	0	0
Federate 9:	0.00	250	30.00	0	0.00	0	0.00	25	0	0
	2.92	6113	36.88	0	0.18	73	0.19	613		

random_5c_6s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.69	649	32.69	0	0.17	56	0.31	65	0	0

Federate 1:	1.26	150	31.26	0	0.02	4	0.47	15	0	0
Federate 2:	1.75	639	31.75	0	0.11	60	0.19	64	0	0
Federate 3:	2.69	649	32.69	0	0.17	62	0.28	65	0	0
Federate 4:	1.26	100	31.26	0	0.01	2	0.58	10	0	0
Federate 5:	0.98	569	30.98	0	0.06	38	0.15	57	0	0
Federate 6:	1.69	599	31.69	0	0.10	49	0.20	60	0	0
Federate 7:	2.69	649	32.69	0	0.17	54	0.32	65	0	0
Federate 8:	1.89	639	31.89	0	0.12	47	0.25	64	0	0
Federate 9:		0								
	2.03	4643	32.03	0	0.09	37	0.28	465		

random_5c_7s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	3.01	599	33.01	0	0.18	55	0.32	70	0	0
Federate 1:	3.62	499	33.62	0	0.18	38	0.47	60	0	0
Federate 2:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
Federate 3:	0.00	399	30.00	0	0.00	26	0.00	50	0	0
Federate 4:	0.84	529	30.84	0	0.04	33	0.13	63	0	0
Federate 5:	3.62	499	33.62	0	0.18	35	0.50	60	0	0
Federate 6:	3.01	599	34.68	0	0.18	73	0.25	70	0	0
Federate 7:	3.62	499	33.62	0	0.18	37	0.48	60	0	0
Federate 8:	3.01	599	36.35	0	0.18	69	0.26	70	0	0
Federate 9:	0.00	399	30.00	0	0.00	31	0.00	50	0	0
	2.39	4721	33.02	0	0.11	40	0.24	563		

random_5c_8s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	418	30.00	0	0.00	21	0.00	42	0	0
Federate 1:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
Federate 2:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
Federate 3:	0.00	508	30.00	0	0.00	41	0.00	51	0	0
Federate 4:	5.58	926	35.58	0	0.52	99	0.52	93	0	0
Federate 5:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
Federate 6:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
Federate 7:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
Federate 8:	0.63	428	30.63	0	0.03	22	0.12	43	0	0
Federate 9:		0								
	4.50	6910	34.50	0	0.31	68	0.32	694		

random_5c_9s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 1:	7.09	947	41.83	0	0.67	99	0.67	95	0	0
Federate 2:	8.05	997	43.04	0	0.80	99	0.81	100	0	0
Federate 3:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
Federate 4:	5.35	948	39.55	0	0.51	99	0.51	95	0	0
Federate 5:	5.11	947	41.42	0	0.48	99	0.49	95	0	0
Federate 6:	5.12	947	41.43	0	0.49	99	0.49	95	0	0
Federate 7:	0.99	499	49.95	0	0.05	99	0.05	50	0	0
Federate 8:	4.02	799	36.52	0	0.32	99	0.32	80	0	0
Federate 9:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
	102.08	8580	136.38	421	8.76	99	8.78	880		

random_5c_10s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.87	250	31.87	0	0.05	10	0.43	25	0	0
Federate 1:	1.87	250	31.87	0	0.05	10	0.43	25	0	0
Federate 2:	1.60	200	31.60	0	0.03	10	0.31	20	0	0
Federate 3:	0.54	130	30.54	0	0.01	6	0.10	13	0	0
Federate 4:	2.25	250	32.25	0	0.06	11	0.51	25	0	0
Federate 5:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 6:	2.00	250	32.00	0	0.05	10	0.46	25	0	0
Federate 7:	1.87	250	31.87	0	0.05	10	0.43	25	0	0
Federate 8:	0.33	210	30.33	0	0.01	5	0.12	21	0	0
Federate 9:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
	1.55	2100	31.55	0	0.03	8	0.35	210		

random_5c_1s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.98	400	30.98	0	0.04	20	0.19	40	0	0
Federate 1:	1.72	400	33.72	0	0.07	24	0.28	40	0	0
Federate 2:	0.00	250	30.00	0	0.00	0	0.00	25	0	0
Federate 3:	0.31	320	30.31	0	0.01	9	0.11	32	0	0
Federate 4:	2.16	500	35.36	0	0.11	46	0.23	50	0	0
Federate 5:	0.31	320	30.31	0	0.01	9	0.11	32	0	0
Federate 6:	0.75	350	37.61	0	0.03	22	0.12	35	0	0
Federate 7:	0.00	50	30.00	0	0.00	0	0.00	5	0	0
Federate 8:	0.83	420	33.21	0	0.03	26	0.13	42	0	0
Federate 9:		0								
	0.99	3010	32.91	0	0.03	15	0.12	301		

random_5c_2s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.96	598	30.96	0	0.06	55	0.10	61	0	0
Federate 1:	8.13	884	40.27	0	0.72	96	0.74	110	0	0
Federate 2:	1.33	498	31.33	0	0.07	24	0.27	50	0	0
Federate 3:	1.75	805	33.12	0	0.14	76	0.18	101	0	0
Federate 4:	0.00	110	50.00	0	0.00	1	0.00	11	0	0
Federate 5:	8.39	894	42.95	0	0.75	99	0.75	111	0	0
Federate 6:	0.67	508	37.36	0	0.03	53	0.06	51	0	0
Federate 7:	2.69	805	38.28	0	0.22	99	0.22	101	0	0
Federate 8:	1.74	507	42.69	0	0.09	78	0.11	51	0	0
Federate 9:	0.00	398	30.00	0	0.00	0	-0.00	40	0	0
	3.45	6007	37.23	0	0.21	58	0.24	687		

random_5c_3s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.04	80	30.04	0	0.00	2	0.01	8	0	0
Federate 1:	0.98	60	32.65	0	0.01	1	0.59	6	0	0
Federate 2:	0.00	30	43.33	0	0.00	1	0.00	3	0	0
Federate 3:	0.00	10	60.00	0	0.00	0	0.00	1	0	0
Federate 4:	0.48	180	34.92	0	0.01	8	0.10	18	0	0
Federate 5:	0.48	130	38.94	0	0.01	5	0.10	13	0	0
Federate 6:	0.00	50	42.00	0	0.00	0	0.00	5	0	0
Federate 7:	0.00	20	70.00	0	0.00	0	0.00	2	0	0
Federate 8:	0.00	90	34.44	0	0.00	0	0.00	9	0	0
Federate 9:		0								
	0.32	650	37.25	0	0.00	2	0.08	65		

random_5c_4s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.31	529	30.31	0	0.02	32	0.05	53	0	0
Federate 1:	0.33	429	31.03	0	0.01	8	0.16	43	0	0
Federate 2:	0.82	439	30.82	0	0.04	9	0.37	44	0	0
Federate 3:	1.19	469	32.47	0	0.06	17	0.32	47	0	0
Federate 4:	0.03	50	36.03	0	0.00	0	0.00	5	0	0
Federate 5:	0.00	70	44.29	0	0.00	2	0.00	7	0	0
Federate 6:	0.00	70	48.57	0	0.00	2	0.00	7	0	0
Federate 7:	0.57	429	34.30	0	0.02	14	0.17	43	0	0
Federate 8:	0.00	60	50.00	0	0.00	0	0.00	6	0	0
Federate 9:		0								
	0.58	2545	33.05	0	0.01	8	0.11	255		

random_5c_5s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	3.35	788	33.35	0	0.26	82	0.32	79	0	0
Federate 1:	2.64	748	32.64	0	0.20	75	0.26	75	0	0
Federate 2:	1.61	648	31.61	0	0.10	66	0.16	65	0	0
Federate 3:	2.64	748	32.64	0	0.20	77	0.25	75	0	0
Federate 4:	6.29	848	36.29	0	0.53	87	0.61	85	0	0
Federate 5:		0								
Federate 6:	3.33	788	33.83	0	0.26	82	0.32	79	0	0
Federate 7:	6.17	848	36.17	0	0.52	87	0.60	85	0	0
Federate 8:	1.46	450	31.46	0	0.07	24	0.27	45	0	0
Federate 9:	2.64	748	32.64	0	0.20	77	0.25	75	0	0
	3.54	6614	33.60	0	0.23	66	0.30	663		

random_5c_6s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.83	140	30.83	0	0.01	3	0.32	14	0	0
Federate 1:	1.83	150	32.49	0	0.03	4	0.59	15	0	0
Federate 2:	1.60	60	33.27	0	0.01	1	0.96	6	0	0
Federate 3:	2.86	70	38.58	0	0.02	1	1.02	7	0	0
Federate 4:	0.35	20	60.35	0	0.00	1	0.07	2	0	0
Federate 5:	0.10	70	41.53	0	0.00	1	0.07	7	0	0
Federate 6:	1.10	599	32.77	0	0.07	33	0.20	60	0	0
Federate 7:	0.10	70	47.24	0	0.00	1	0.07	7	0	0
Federate 8:	0.61	140	35.61	0	0.01	8	0.10	14	0	0
Federate 9:		0								
	1.10	1319	34.82	0	0.01	5	0.34	132		

random_5c_7s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	3.01	599	33.01	0	0.18	55	0.32	70	0	0
Federate 1:	3.62	499	33.62	0	0.18	38	0.47	60	0	0
Federate 2:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
Federate 3:	0.00	399	30.00	0	0.00	26	0.00	50	0	0
Federate 4:	0.84	529	30.84	0	0.04	33	0.13	63	0	0
Federate 5:	3.62	499	33.62	0	0.18	35	0.50	60	0	0
Federate 6:	3.01	599	34.68	0	0.18	73	0.25	70	0	0
Federate 7:	3.62	499	33.62	0	0.18	37	0.48	60	0	0
Federate 8:	3.01	599	36.35	0	0.18	69	0.26	70	0	0

Federate 9:	0.00	399	30.00	0	0.00	31	0.00	50	0	0
	2.39	4721	33.02	0	0.11	40	0.24	563		

random_5c_8s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 1:	0.02	518	30.41	0	0.00	41	0.00	53	0	0
Federate 2:	0.21	518	31.18	0	0.01	49	0.02	53	0	0
Federate 3:	0.00	488	30.00	0	0.00	39	0.00	50	0	0
Federate 4:	7.43	886	38.11	0	0.66	98	0.67	92	0	0
Federate 5:	0.35	518	32.28	0	0.02	50	0.04	53	0	0
Federate 6:	1.32	697	32.76	0	0.09	84	0.11	72	0	0
Federate 7:	0.35	508	32.72	0	0.02	32	0.05	52	0	0
Federate 8:	0.00	20	100.00	0	0.00	0	0.00	2	0	0
Federate 9:		0								
	1.91	4173	33.33	0	0.08	39	0.09	429		

random_5c_9s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 1:	5.32	947	37.95	0	0.50	99	0.51	95	0	0
Federate 2:	9.55	998	40.55	0	0.95	99	0.96	100	0	0
Federate 3:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
Federate 4:	5.35	948	39.55	0	0.51	99	0.51	95	0	0
Federate 5:	5.11	948	35.11	0	0.48	99	0.49	95	0	0
Federate 6:	5.12	947	41.43	0	0.49	99	0.49	95	0	0
Federate 7:	1.00	500	34.00	0	0.05	22	0.23	50	0	0
Federate 8:	4.02	799	36.52	0	0.32	99	0.32	80	0	0
Federate 9:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
	102.03	8583	134.00	421	8.76	91	8.80	880		

random_5c_10s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
Federate 1:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 2:	1.60	200	31.60	0	0.03	10	0.31	20	0	0
Federate 3:	2.24	130	35.31	0	0.03	4	0.62	13	0	0
Federate 4:	0.20	170	30.20	0	0.00	3	0.10	17	0	0
Federate 5:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 6:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 7:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
Federate 8:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
Federate 9:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
	0.92	1810	31.14	0	0.02	4	0.31	181		

random_5c_1s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.27	500	32.27	0	0.11	39	0.29	50	0	0
Federate 1:	2.27	500	32.27	0	0.11	39	0.29	50	0	0
Federate 2:	0.61	330	30.61	0	0.02	9	0.20	33	0	0
Federate 3:	1.08	420	31.08	0	0.05	28	0.16	42	0	0
Federate 4:	2.47	500	32.47	0	0.12	39	0.31	50	0	0
Federate 5:	2.27	500	32.27	0	0.11	39	0.29	50	0	0

Federate 6:	2.32	500	32.32	0	0.12	39	0.30	50	0	0
Federate 7:	1.10	250	31.10	0	0.03	11	0.24	25	0	0
Federate 8:	2.27	500	32.27	0	0.11	39	0.29	50	0	0
Federate 9:	2.27	500	32.27	0	0.11	39	0.29	50	0	0
	2.00	4500	32.00	0	0.09	32	0.27	450		

random_5c_2s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.36	607	32.36	0	0.14	51	0.28	61	0	0
Federate 1:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
Federate 2:	6.29	995	36.29	0	0.63	99	0.63	100	0	0
Federate 3:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
Federate 4:	4.33	807	34.33	0	0.35	91	0.38	81	0	0
Federate 5:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
Federate 6:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
Federate 7:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
Federate 8:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
Federate 9:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
	640.14	9388	670.14	3017	60.10	94	60.33	1089		

random_5c_3s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 1:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 2:	0.00	70	30.00	0	0.00	3	0.00	7	0	0
Federate 3:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 4:	2.75	220	32.75	0	0.06	10	0.59	22	0	0
Federate 5:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 6:	1.87	200	31.87	0	0.04	8	0.47	20	0	0
Federate 7:	0.02	170	30.02	0	0.00	7	0.00	17	0	0
Federate 8:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 9:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
	2.08	1980	32.08	0	0.04	8	0.42	198		

random_5c_4s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.31	529	30.31	0	0.02	32	0.05	53	0	0
Federate 1:	1.83	569	31.83	0	0.10	37	0.28	57	0	0
Federate 2:	1.79	559	31.79	0	0.10	34	0.29	56	0	0
Federate 3:	1.83	569	31.83	0	0.10	37	0.28	57	0	0
Federate 4:	1.36	549	31.36	0	0.07	35	0.21	55	0	0
Federate 5:	1.83	569	31.83	0	0.10	37	0.28	57	0	0
Federate 6:	1.13	469	31.13	0	0.05	17	0.31	47	0	0
Federate 7:	1.83	569	31.83	0	0.10	37	0.28	57	0	0
Federate 8:	0.07	170	30.07	0	0.00	5	0.02	17	0	0
Federate 9:	1.83	569	31.83	0	0.10	37	0.28	57	0	0
	1.50	5121	31.50	0	0.08	31	0.23	513		

random_5c_5s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	7.71	888	37.71	0	0.69	92	0.74	89	0	0
Federate 1:	7.71	888	37.71	0	0.69	92	0.74	89	0	0
Federate 2:	1.87	688	31.87	0	0.13	71	0.18	69	0	0

Federate 3:	7.71	888	37.71	0	0.69	92	0.74	89	0	0
Federate 4:	7.95	888	37.95	0	0.71	92	0.76	89	0	0
Federate 5:	2.89	638	32.89	0	0.18	48	0.38	64	0	0
Federate 6:	7.77	888	37.77	0	0.69	92	0.74	89	0	0
Federate 7:	7.71	888	37.71	0	0.69	92	0.74	89	0	0
Federate 8:	1.46	450	31.46	0	0.07	23	0.28	45	0	0
Federate 9:	7.71	888	37.71	0	0.69	92	0.74	89	0	0
	6.51	7992	36.51	0	0.52	79	0.60	801		

random_5c_6s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.69	649	32.69	0	0.17	55	0.31	65	0	0
Federate 1:	1.26	150	31.26	0	0.02	4	0.47	15	0	0
Federate 2:	1.75	639	31.75	0	0.11	63	0.18	64	0	0
Federate 3:	2.69	649	32.69	0	0.17	56	0.31	65	0	0
Federate 4:	2.96	649	32.96	0	0.19	55	0.35	65	0	0
Federate 5:	0.98	569	30.98	0	0.06	49	0.11	57	0	0
Federate 6:	1.69	599	31.69	0	0.10	44	0.23	60	0	0
Federate 7:	2.69	649	32.69	0	0.17	54	0.32	65	0	0
Federate 8:	1.89	639	31.89	0	0.12	58	0.21	64	0	0
Federate 9:	2.69	649	32.69	0	0.17	58	0.30	65	0	0
	2.22	5841	32.22	0	0.13	50	0.28	585		

random_5c_7s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	3.20	828	33.20	0	0.26	86	0.31	83	0	0
Federate 1:	3.20	828	33.20	0	0.26	85	0.31	83	0	0
Federate 2:	0.64	230	30.64	0	0.01	10	0.14	23	0	0
Federate 3:	3.20	828	33.20	0	0.26	85	0.31	83	0	0
Federate 4:	1.77	728	31.77	0	0.13	75	0.17	73	0	0
Federate 5:	1.10	748	31.10	0	0.08	73	0.11	75	0	0
Federate 6:	2.41	778	32.41	0	0.19	74	0.25	78	0	0
Federate 7:	3.20	828	33.20	0	0.26	86	0.31	83	0	0
Federate 8:	3.20	828	33.20	0	0.26	84	0.31	83	0	0
Federate 9:	3.20	828	33.20	0	0.26	87	0.30	83	0	0
	2.69	7452	32.69	0	0.20	74	0.25	747		

random_5c_8s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	418	30.00	0	0.00	21	0.00	42	0	0
Federate 1:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
Federate 2:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
Federate 3:	0.00	508	30.00	0	0.00	51	0.00	51	0	0
Federate 4:	5.58	926	35.58	0	0.52	99	0.52	93	0	0
Federate 5:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
Federate 6:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
Federate 7:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
Federate 8:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
Federate 9:	5.55	926	35.55	0	0.51	99	0.52	93	0	0
	4.93	8334	34.93	0	0.41	87	0.41	837		

random_5c_9s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 1:	7.93	947	37.93	0	0.75	99	0.75	95	0	0
Federate 2:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 3:	10.55	998	40.55	0	1.05	99	1.06	100	0	0
Federate 4:	857.74	996	887.74	430	85.43	99	85.68	120	0	0
Federate 5:	656.25	998	686.25	258	65.49	99	65.68	115	0	0
Federate 6:	846.32	997	876.32	424	84.38	99	84.62	120	0	0
Federate 7:	5.67	699	35.67	0	0.40	59	0.66	70	0	0
Federate 8:	10.15	998	40.15	0	1.01	99	1.02	100	0	0
Federate 9:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
	509.19	9627	539.19	2375	49.02	95	49.19	1080		

random_5c_10s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.87	250	31.87	0	0.05	10	0.43	25	0	0
Federate 1:	1.87	250	31.87	0	0.05	10	0.43	25	0	0
Federate 2:	1.60	200	31.60	0	0.03	10	0.31	20	0	0
Federate 3:	0.54	130	30.54	0	0.01	6	0.10	13	0	0
Federate 4:	2.25	250	32.25	0	0.06	11	0.51	25	0	0
Federate 5:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 6:	2.00	250	32.00	0	0.05	10	0.46	25	0	0
Federate 7:	1.87	250	31.87	0	0.05	10	0.43	25	0	0
Federate 8:	0.33	210	30.33	0	0.01	5	0.12	21	0	0
Federate 9:	1.87	250	31.87	0	0.05	10	0.43	25	0	0
	1.66	2250	31.66	0	0.04	9	0.39	225		

random_5c_1s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.81	400	30.81	0	0.03	21	0.15	40	0	0
Federate 1:	1.15	400	32.90	0	0.05	21	0.21	40	0	0
Federate 2:	0.00	250	30.00	0	0.00	0	0.00	25	0	0
Federate 3:	1.05	320	35.42	0	0.03	11	0.30	32	0	0
Federate 4:	1.64	500	35.84	0	0.08	56	0.15	50	0	0
Federate 5:	0.18	320	38.93	0	0.01	18	0.03	32	0	0
Federate 6:	0.70	350	33.56	0	0.02	21	0.11	35	0	0
Federate 7:	0.00	50	80.00	0	0.00	0	0.00	5	0	0
Federate 8:	1.19	420	43.57	0	0.05	74	0.07	42	0	0
Federate 9:		0								
	0.91	3010	36.13	0	0.03	22	0.10	301		

random_5c_2s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.96	598	30.96	0	0.06	55	0.10	61	0	0
Federate 1:	8.13	884	40.27	0	0.72	96	0.74	110	0	0
Federate 2:	1.33	498	31.33	0	0.07	24	0.27	50	0	0
Federate 3:	1.75	805	33.12	0	0.14	76	0.18	101	0	0
Federate 4:	0.00	110	50.00	0	0.00	1	0.00	11	0	0
Federate 5:	8.39	894	42.95	0	0.75	99	0.75	111	0	0
Federate 6:	0.67	508	37.36	0	0.03	53	0.06	51	0	0
Federate 7:	2.69	805	38.28	0	0.22	99	0.22	101	0	0
Federate 8:	1.74	507	42.69	0	0.09	78	0.11	51	0	0
Federate 9:	0.00	398	30.00	0	0.00	0	-0.00	40	0	0
	3.45	6007	37.23	0	0.21	58	0.24	687		

random_5c_3s.on

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.04	80	30.04	0	0.00	2	0.01	8	0	0
Federate 1:	0.98	60	32.65	0	0.01	1	0.59	6	0	0
Federate 2:	0.00	30	43.33	0	0.00	1	0.00	3	0	0
Federate 3:	0.00	10	60.00	0	0.00	0	0.00	1	0	0
Federate 4:	0.48	180	34.92	0	0.01	8	0.10	18	0	0
Federate 5:	0.48	130	38.94	0	0.01	5	0.10	13	0	0
Federate 6:	0.00	50	42.00	0	0.00	0	0.00	5	0	0
Federate 7:	0.00	20	70.00	0	0.00	0	0.00	2	0	0
Federate 8:	0.00	90	34.44	0	0.00	0	0.00	9	0	0
Federate 9:		0								
	0.32	650	37.25	0	0.00	2	0.08	65		

random_5c_4s.on

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.31	529	30.31	0	0.02	32	0.05	53	0	0
Federate 1:	0.33	429	31.03	0	0.01	8	0.16	43	0	0
Federate 2:	0.82	439	30.82	0	0.04	9	0.37	44	0	0
Federate 3:	1.19	469	32.47	0	0.06	17	0.32	47	0	0
Federate 4:	0.03	50	36.03	0	0.00	0	0.00	5	0	0
Federate 5:	0.00	70	44.29	0	0.00	2	0.00	7	0	0
Federate 6:	0.00	70	48.57	0	0.00	2	0.00	7	0	0
Federate 7:	0.57	429	34.30	0	0.02	14	0.17	43	0	0
Federate 8:	0.00	60	50.00	0	0.00	0	0.00	6	0	0
Federate 9:		0								
	0.58	2545	33.05	0	0.01	8	0.11	255		

random_5c_5s.on

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.93	788	32.93	0	0.23	90	0.26	79	0	0
Federate 1:	0.98	350	30.98	0	0.03	13	0.25	35	0	0
Federate 2:	1.61	648	31.61	0	0.10	66	0.16	65	0	0
Federate 3:	2.88	748	32.88	0	0.22	82	0.26	75	0	0
Federate 4:	4.43	848	34.43	0	0.38	87	0.43	85	0	0
Federate 5:		0								
Federate 6:	3.26	788	33.76	0	0.26	90	0.28	79	0	0
Federate 7:	5.00	848	36.18	0	0.42	90	0.47	85	0	0
Federate 8:	1.46	450	35.91	0	0.07	36	0.18	45	0	0
Federate 9:	0.98	350	30.98	0	0.03	13	0.25	35	0	0
	2.99	5818	33.58	0	0.17	57	0.25	583		

random_5c_6s.on

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.26	150	31.26	0	0.02	4	0.47	15	0	0
Federate 1:	1.96	150	37.29	0	0.03	2	1.04	15	0	0
Federate 2:	0.00	60	30.00	0	0.00	1	0.00	6	0	0
Federate 3:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 4:	1.54	20	31.54	0	0.00	1	0.28	2	0	0
Federate 5:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 6:	1.37	599	34.04	0	0.08	43	0.19	60	0	0
Federate 7:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 8:	0.47	140	47.61	0	0.01	0	1.00	14	0	0
Federate 9:		0								

	1.08	1329	34.69	0	0.01	5	0.31	133		

random_5c_7s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	3.01	599	33.01	0	0.18	55	0.32	70	0	0
Federate 1:	3.62	499	33.62	0	0.18	38	0.47	60	0	0
Federate 2:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
Federate 3:	0.00	399	30.00	0	0.00	26	0.00	50	0	0
Federate 4:	0.84	529	30.84	0	0.04	33	0.13	63	0	0
Federate 5:	3.62	499	33.62	0	0.18	35	0.50	60	0	0
Federate 6:	3.01	599	34.68	0	0.18	73	0.25	70	0	0
Federate 7:	3.62	499	33.62	0	0.18	37	0.48	60	0	0
Federate 8:	3.01	599	36.35	0	0.18	69	0.26	70	0	0
Federate 9:	0.00	399	30.00	0	0.00	31	0.00	50	0	0
	2.39	4721	33.02	0	0.11	40	0.24	563		

random_5c_8s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 1:	0.02	518	30.41	0	0.00	41	0.00	53	0	0
Federate 2:	0.21	518	31.18	0	0.01	49	0.02	53	0	0
Federate 3:	0.00	488	30.00	0	0.00	39	0.00	50	0	0
Federate 4:	7.43	886	38.11	0	0.66	98	0.67	92	0	0
Federate 5:	0.35	518	32.28	0	0.02	50	0.04	53	0	0
Federate 6:	1.32	697	32.76	0	0.09	84	0.11	72	0	0
Federate 7:	0.35	508	32.72	0	0.02	32	0.05	52	0	0
Federate 8:	0.00	20	100.00	0	0.00	0	0.00	2	0	0
Federate 9:		0								
	1.91	4173	33.33	0	0.08	39	0.09	429		

random_5c_9s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 1:	7.09	947	41.83	0	0.67	99	0.67	95	0	0
Federate 2:	8.05	997	43.04	0	0.80	99	0.81	100	0	0
Federate 3:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
Federate 4:	5.35	948	39.55	0	0.51	99	0.51	95	0	0
Federate 5:	5.11	947	41.42	0	0.48	99	0.49	95	0	0
Federate 6:	5.12	947	41.43	0	0.49	99	0.49	95	0	0
Federate 7:	0.99	499	49.95	0	0.05	99	0.05	50	0	0
Federate 8:	4.02	799	36.52	0	0.32	99	0.32	80	0	0
Federate 9:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
	102.08	8580	136.38	421	8.76	99	8.78	880		

random_5c_10s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
Federate 1:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 2:	1.20	200	33.20	0	0.02	12	0.19	20	0	0
Federate 3:	0.53	130	39.76	0	0.01	10	0.07	13	0	0
Federate 4:	0.20	170	30.20	0	0.00	3	0.10	17	0	0
Federate 5:	0.00	210	35.71	0	0.00	5	0.00	21	0	0
Federate 6:	0.44	210	38.06	0	0.01	7	0.13	21	0	0
Federate 7:	0.00	170	30.00	0	0.00	3	0.00	17	0	0

```
Federate 8:      0.00   170    30.00    0   0.00   3   0.00   17    0    0
Federate 9:      0.00   100    30.00    0   0.00   0   0.00   10    0    0
_____________________________________________________________________________
                 0.45  1740    32.97    0   0.01   5   0.12  174
```

Seed 1 connection sets

PP	2 10 2 4 8 0 2 2 7 0 1 3 4 5 6 8 0 2 5 7 0 1 3 4 5 7 8 0 3 8 4 0 1 4 6 0 7 25 8 0 1 2 3 4 5 6 8 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 2 9 0 1 3 4 5 6 7 8 9 1 2 5 9 0 1 3 4 5 6 7 8 9 1 3 8 9 0 1 2 4 5 6 7 8 9 1 7 25 9 0 1 2 3 4 5 6 8 9 1
LOC	2 10 8 0 1 3 4 5 6 7 8 2 2 2 7 0 1 3 4 5 6 8 0 2 5 8 0 1 3 4 5 6 7 8 2 3 8 8 0 1 2 4 5 6 7 8 2 7 25 8 0 1 2 3 4 5 6 8 1	**IRLOC**	2 10 2 4 8 0 2 2 7 0 1 3 4 5 6 8 1 2 5 7 0 1 3 4 5 7 8 2 3 8 4 0 1 4 6 0 7 25 8 0 1 2 3 4 5 6 8 1
ON	2 10 2 4 8 0 2 2 7 0 1 3 4 5 6 8 0 2 5 7 0 1 3 4 5 7 8 0 3 8 4 0 1 4 6 2 7 25 8 0 1 2 3 4 5 6 8 1		

Seed 2 connection sets

PP	0 10 6 2 3 4 6 7 8 0 0 50 4 1 3 5 7 0 2 1 7 0 3 4 5 6 7 8 0 2 20 3 0 1 5 0 4 40 9 0 1 2 3 5 6 7 8 9 0	**BR**	0 10 9 1 2 3 4 5 6 7 8 9 1 0 50 9 1 2 3 4 5 6 7 8 9 1 2 1 9 0 1 3 4 5 6 7 8 9 1 2 20 9 0 1 3 4 5 6 7 8 9 1 4 40 9 0 1 2 3 5 6 7 8 9 1
LOC	0 10 6 2 3 4 6 7 8 0 0 50 4 1 3 5 7 2 2 1 7 0 3 4 5 6 7 8 0 2 20 3 0 1 5 0 4 40 9 0 1 2 3 5 6 7 8 9 1	**IRLOC**	0 10 6 2 3 4 6 7 8 0 0 50 4 1 3 5 7 2 2 1 7 0 3 4 5 6 7 8 0 2 20 3 0 1 5 0 4 40 9 0 1 2 3 5 6 7 8 9 1
ON	0 10 6 2 3 4 6 7 8 0 0 50 4 1 3 5 7 2 2 1 7 0 3 4 5 6 7 8 0 2 20 3 0 1 5 0 4 40 9 0 1 2 3 5 6 7 8 9 1		

Seed 3 connection sets

PP	2 10 2 4 5 0 2 5 4 0 1 4 8 0 6 2 5 0 2 4 5 7 0 7 1 7 0 1 2 3 4 5 6 0 7 4 2 6 8 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 5 9 0 1 3 4 5 6 7 8 9 1 6 2 9 0 1 2 3 4 5 7 8 9 1 7 1 9 0 1 2 3 4 5 6 8 9 1 7 4 9 0 1 2 3 4 5 6 8 9 1

LOC	2 10 5 0 1 4 5 8 1 2 5 5 0 1 4 5 8 1 6 2 8 0 1 2 3 4 5 7 8 2 7 1 8 0 1 2 3 4 5 6 8 2 7 4 8 0 1 2 3 4 5 6 8 2	**IRLOC**	2 10 2 4 5 2 2 5 4 0 1 4 8 1 6 2 5 0 2 4 5 7 0 7 1 7 0 1 2 3 4 5 6 0 7 4 2 6 8 0
ON	2 10 2 4 5 2 2 5 4 0 1 4 8 1 6 2 5 0 2 4 5 7 0 7 1 7 0 1 2 3 4 5 6 0 7 4 2 6 8 0		

Seed 4 connection sets

PP	0 4 6 2 3 4 5 6 8 0 2 1 7 0 1 3 4 5 6 7 0 4 2 7 0 1 3 5 6 7 8 0 6 10 1 0 0 8 40 5 0 1 2 3 7 0	**BR**	0 4 9 1 2 3 4 5 6 7 8 9 1 2 1 9 0 1 3 4 5 6 7 8 9 1 4 2 9 0 1 2 3 5 6 7 8 9 1 6 10 9 0 1 2 3 4 5 7 8 9 1 8 40 9 0 1 2 3 4 5 6 7 9 1
LOC	0 4 8 1 2 3 4 5 6 7 8 2 2 1 8 0 1 3 4 5 6 7 8 2 4 2 8 0 1 2 3 5 6 7 8 2 6 10 8 0 1 2 3 4 5 7 8 2 8 40 5 0 1 2 3 7 1	**IRLOC**	0 4 6 2 3 4 5 6 8 2 2 1 7 0 1 3 4 5 6 7 0 4 2 7 0 1 3 5 6 7 8 0 6 10 1 0 0 8 40 5 0 1 2 3 7 1
ON	0 4 6 2 3 4 5 6 8 2 2 1 7 0 1 3 4 5 6 7 0 4 2 7 0 1 3 5 6 7 8 0 6 10 1 0 0 8 40 5 0 1 2 3 7 1		

Seed 5 connection sets

PP	2 10 3 4 7 8 0 2 10 7 0 1 3 4 6 7 8 0 5 25 9 0 1 2 3 4 6 7 8 9 0 8 4 2 0 6 0 8 40 7 0 1 2 3 4 6 7 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 10 9 0 1 3 4 5 6 7 8 9 1 5 25 9 0 1 2 3 4 6 7 8 9 1 8 4 9 0 1 2 3 4 5 6 7 9 1 8 40 9 0 1 2 3 4 5 6 7 9 1
LOC	2 10 3 4 7 8 0 2 10 7 0 1 3 4 6 7 8 0 5 25 9 0 1 2 3 4 6 7 8 9 2 8 4 2 0 6 0 8 40 7 0 1 2 3 4 6 7 1	**IRLOC**	2 10 3 4 7 8 2 2 10 8 0 1 3 4 6 7 8 9 1 5 25 9 0 1 2 3 4 6 7 8 9 1 8 4 2 0 6 0 8 40 8 0 1 2 3 4 6 7 9 1
ON	2 10 3 4 7 8 0 2 10 8 0 1 3 4 6 7 8 9 1 5 25 9 0 1 2 3 4 6 7 8 9 1 8 4 2 0 6 0 8 40 6 0 2 3 4 6 7 2		

Seed 6 connection sets

PP	1 50 1 6 0 2 1 8 0 1 3 4 5 6 7 8 0 5 8 4 0 1 6 8 0 6 5 7 0 1 2 3 5 7 8 0 8 1 7 1 2 3 4 5 6 7 0	**BR**	1 50 9 0 2 3 4 5 6 7 8 9 1 2 1 9 0 1 3 4 5 6 7 8 9 1 5 8 9 0 1 2 3 4 6 7 8 9 1 6 5 9 0 1 2 3 4 5 7 8 9 1 8 1 9 0 1 2 3 4 5 6 7 9 1
LOC	1 50 7 0 2 3 5 6 7 8 1 2 1 8 0 1 3 4 5 6 7 8 2 5 8 8 0 1 2 3 4 6 7 8 2 6 5 7 0 1 2 3 5 7 8 1 8 1 8 0 1 2 3 4 5 6 7 2	**IRLOC**	1 50 1 6 0 2 1 8 0 1 3 4 5 6 7 8 0 5 8 4 0 1 6 8 2 6 5 7 0 1 2 3 5 7 8 1 8 1 7 1 2 3 4 5 6 7 0
ON	1 50 1 6 0 2 1 8 0 1 3 4 5 6 7 8 2 5 8 4 0 1 6 8 0 6 5 7 0 1 2 3 5 7 8 1 8 1 8 0 1 2 3 4 5 6 7 2		

Seed 7 connection sets

PP	2 10 3 0 6 8 0 2 50 9 0 1 3 4 5 6 7 8 9 0 4 10 7 0 1 2 5 6 7 8 0 5 8 1 4 0 6 5 1 4 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 50 9 0 1 3 4 5 6 7 8 9 1 4 10 9 0 1 2 3 5 6 7 8 9 1 5 8 9 0 1 2 3 4 6 7 8 9 1 6 5 9 0 1 2 3 4 5 7 8 9 1
LOC	2 10 3 0 6 8 0 2 50 9 0 1 3 4 5 6 7 8 9 1 4 10 7 0 1 2 5 6 7 8 2 5 8 1 4 0 6 5 1 4 0	**IRLOC**	2 10 3 0 6 8 0 2 50 9 0 1 3 4 5 6 7 8 9 1 4 10 7 0 1 2 5 6 7 8 2 5 8 1 4 0 6 5 1 4 0
ON			

Seed 8 connection sets

PP	0 1 3 1 2 5 0 0 50 7 1 2 3 4 5 6 7 0 3 2 8 0 1 2 4 5 6 7 8 0 3 20 1 4 0 3 20 2 4 6 0	**BR**	0 1 9 1 2 3 4 5 6 7 8 9 1 0 50 9 1 2 3 4 5 6 7 8 9 1 3 2 9 0 1 2 4 5 6 7 8 9 1 3 20 9 0 1 2 4 5 6 7 8 9 1 3 20 9 0 1 2 4 5 6 7 8 9 1
LOC	0 1 8 1 2 3 4 5 6 7 8 2 0 50 7 1 2 3 4 5 6 7 1 3 2 8 0 1 2 4 5 6 7 8 2 3 20 8 0 1 2 4 5 6 7 8 2 3 20 8 0 1 2 4 5 6 7 8 2	**IRLOC**	0 1 3 1 2 5 0 0 50 7 1 2 3 4 5 6 7 1 3 2 8 0 1 2 4 5 6 7 8 0 3 20 1 4 0 3 20 2 4 6 2
ON	0 1 3 1 2 5 0 0 50 7 1 2 3 4 5 6 7 1 3 2 8 0 1 2 4 5 6 7 8 0 3 20 1 4 0 3 20 2 4 6 2		

Seed 9 connection sets

PP	1 25 9 0 2 3 4 5 6 7 8 9 0 3 20 4 0 1 4 6 0 5 5 5 0 1 2 7 8 0 7 50 9 0 1 2 3 4 5 6 8 9 0 8 20 5 0 1 2 5 7 0	**BR**	1 25 9 0 2 3 4 5 6 7 8 9 1 3 20 9 0 1 2 4 5 6 7 8 9 1 5 5 9 0 1 2 3 4 6 7 8 9 1 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 9 0 1 2 3 4 5 6 7 9 1
LOC	1 25 9 0 2 3 4 5 6 7 8 9 2 3 20 4 0 1 4 6 0 5 5 5 0 1 2 7 8 0 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 5 0 1 2 5 7 0	**IRLOC**	1 25 9 0 2 3 4 5 6 7 8 9 1 3 20 4 0 1 4 6 0 5 5 5 0 1 2 7 8 0 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 5 0 1 2 5 7 2
ON	1 25 9 0 2 3 4 5 6 7 8 9 2 3 20 4 0 1 4 6 0 5 5 5 0 1 2 7 8 0 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 5 0 1 2 5 7 0		

Seed 10 connection sets

PP	2 5 8 0 1 3 4 5 6 7 8 0 3 2 8 0 1 2 4 5 6 7 8 0 3 10 9 0 1 2 4 5 6 7 8 9 0 5 4 2 2 3 0 8 4 5 1 2 3 5 6 0	**BR**	2 5 9 0 1 3 4 5 6 7 8 9 1 3 2 9 0 1 2 4 5 6 7 8 9 1 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 9 0 1 2 3 4 6 7 8 9 1 8 4 9 0 1 2 3 4 5 6 7 9 1
LOC	2 5 8 0 1 3 4 5 6 7 8 2 3 2 8 0 1 2 4 5 6 7 8 2 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 8 0 1 2 3 4 6 7 8 2 8 4 8 0 1 2 3 4 5 6 7 2	**IRLOC**	2 5 9 0 1 3 4 5 6 7 8 9 1 3 2 9 0 1 2 4 5 6 7 8 9 1 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 2 2 3 0 8 4 5 1 2 3 5 6 2
ON	2 5 8 0 1 3 4 5 6 7 8 2 3 2 8 0 1 2 4 5 6 7 8 1 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 2 2 3 0 8 4 5 1 2 3 5 6 0		

5 connections, 3 groups

seed	measure	PP	BR	LOC	IRLOC	ON	comments
1	out. throt.	1	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	58.22	32	31.97	31.21	31.2	
	% extra	-33%	50%	33%	0%	2%	
2	out. throt.	2	0	0	0	0	4 federates' input weights are at least 100
	tmax	1	7	6	2	2	
	avg. delivery	56.85	670.14	585.1	179.25	179.25	
	% extra	-62%	37%	35%	-4%	-4%	

3	out. throt.	0	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	43.68	32.08	32.07	34.11	34.11	
	% extra	0%	205%	102%	0%	0%	
4	out. throt.	1	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	51.1	31.5	30.97	30.55	30.55	
	% extra	-39%	101%	51%	0%	0%	
5	out. throt.	1	0	0	0	0	
	tmax	1	0	0	0	0	
	avg. delivery	61.85	36.51	36.45	33.71	35.58	
	% extra	-51%	30%	10%	8%	6%	
6	out. throt.	0	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	46.24	32.22	31.32	31.18	31.18	
	% extra	0%	343%	29%	1%	1%	
7	out. throt.	1	0	0	0	0	
	tmax	1	0	0	0	0	
	avg. delivery	59.68	32.69	31.21	30.98	30.98	
	% extra	-68%	32%	14%	0%	0%	
8	out. throt.	1	0	0	0	0	
	tmax	1	0	0	0	0	
	avg. delivery	52.17	34.93	32.23	31.45	31.45	
	% extra	-60%	94%	40%	-2%	-2%	
9	out. throt.	2	0	0	0	0	2 federates' input weights are at least 100;
	tmax	1	6	3	1	1	
	avg. delivery	61.76	539.19	273.06	132.85	134	
	% extra	-54%	9%	3%	-2%	-2%	
10	out. throt.	0	0	0	0	0	
	tmax	2	0	0	0	0	
	avg. delivery	64.7	31.66	31.66	30.8	31.18	
	% extra	-6%	29%	29%	4%	1%	

Offline simulator raw output

random_5c_1s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.64	275	30.64	0	0.02	12	0.14	40	1	0
Federate 1:	0.64	275	40.64	0	0.02	22	0.08	40	1	0
Federate 2:	0.00	125	50.00	0	0.00	0	0.00	25	1	0
Federate 3:	0.26	195	56.67	0	0.00	16	0.03	32	1	0
Federate 4:	1.56	375	54.76	0	0.06	64	0.09	50	1	0
Federate 5:	0.26	194	76.65	0	0.00	52	0.01	32	1	0
Federate 6:	0.54	224	78.93	0	0.01	69	0.02	35	1	0
Federate 7:	0.00	50	80.00	0	0.00	0	0.00	5	0	1
Federate 8:	0.57	294	77.78	0	0.02	98	0.02	42	1	0
Federate 9:		0								
	0.66	2007	58.22	0	0.01	33	0.04	301		

random_5c_2s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.71	311	30.71	0	0.02	11	0.20	61	1	1
Federate 1:	1.07	400	38.60	0	0.04	38	0.11	110	2	0
Federate 2:	0.52	211	41.04	0	0.01	9	0.11	50	1	0
Federate 3:	0.75	320	47.69	0	0.02	40	0.06	101	2	0
Federate 4:	0.00	110	50.00	0	0.00	1	0.00	11	0	1
Federate 5:	1.05	408	56.69	0	0.04	80	0.05	111	2	0
Federate 6:	1.03	220	71.49	0	0.02	41	0.05	51	1	0
Federate 7:	1.06	319	75.16	0	0.03	99	0.03	101	2	0
Federate 8:	1.03	220	91.49	0	0.02	88	0.03	51	1	0
Federate 9:	0.00	110	110.00	0	0.00	98	0.00	40	1	0
	0.84	2629	56.85	0	0.02	50	0.06	687		

random_5c_3s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.21	80	31.21	0	0.01	1	0.97	8	0	0
Federate 1:	0.00	60	40.00	0	0.00	2	0.00	6	0	0
Federate 2:	0.00	30	43.33	0	0.00	1	0.00	3	0	0
Federate 3:	0.00	10	60.00	0	0.00	0	0.00	1	0	0
Federate 4:	1.29	180	41.29	0	0.02	5	0.39	18	0	0
Federate 5:	0.48	130	46.63	0	0.01	4	0.12	13	0	0
Federate 6:	0.00	50	42.00	0	0.00	0	0.00	5	0	0
Federate 7:	0.00	20	70.00	0	0.00	0	0.00	2	0	0
Federate 8:	0.00	90	51.11	0	0.00	4	0.00	9	0	0
Federate 9:		0								
	0.60	650	43.68	0	0.00	2	0.15	65		

random_5c_4s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.03	330	30.03	0	0.00	13	0.01	53	1	0
Federate 1:	0.03	229	40.03	0	0.00	10	0.01	43	1	0
Federate 2:	0.00	239	46.65	0	0.00	0	-0.00	44	1	0
Federate 3:	0.39	269	56.30	0	0.01	30	0.03	47	1	0
Federate 4:	1.95	50	53.95	0	0.01	1	0.97	5	0	0
Federate 5:	1.39	70	62.82	0	0.01	4	0.22	7	0	0
Federate 6:	1.39	70	72.82	0	0.01	4	0.22	7	0	0
Federate 7:	0.00	229	71.75	0	0.00	99	0.00	43	1	0
Federate 8:	0.00	60	83.33	0	0.00	6	0.00	6	0	1
Federate 9:		0								

	0.27	1546	51.10	0	0.00	17	0.15	255	

random_5c_5s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.50	382	31.50	0	0.06	22	0.25	79	2	0
Federate 1:	1.25	342	41.25	0	0.04	30	0.14	75	2	0
Federate 2:	0.56	242	50.56	0	0.01	24	0.06	65	2	0
Federate 3:	1.17	342	58.25	0	0.04	57	0.07	75	2	0
Federate 4:	1.55	442	60.24	0	0.07	91	0.08	85	2	0
Federate 5:		0								
Federate 6:	1.42	381	74.62	0	0.05	98	0.05	79	2	0
Federate 7:	1.59	440	77.98	0	0.07	99	0.07	85	2	0
Federate 8:	0.71	310	81.39	0	0.02	99	0.02	45	1	1
Federate 9:	0.00	110	110.00	0	0.00	98	0.00	25	1	0
	1.23	2991	61.85	0	0.04	62	0.07	613		

random_5c_6s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.83	140	30.83	0	0.01	3	0.32	14	0	0
Federate 1:	1.93	150	41.26	0	0.03	6	0.46	15	0	0
Federate 2:	0.00	60	48.33	0	0.00	2	0.00	6	0	0
Federate 3:	0.10	70	57.24	0	0.00	2	0.02	7	0	0
Federate 4:	0.35	20	60.35	0	0.00	1	0.07	2	0	0
Federate 5:	0.14	70	70.14	0	0.00	2	0.05	7	0	0
Federate 6:	1.08	599	35.42	0	0.06	51	0.13	60	0	0
Federate 7:	1.57	70	84.43	0	0.01	2	0.55	7	0	0
Federate 8:	0.27	140	73.84	0	0.00	15	0.02	14	0	0
Federate 9:		0								
	0.93	1319	46.24	0	0.01	8	0.16	132		

random_5c_7s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.64	267	30.64	0	0.02	8	0.20	70	1	0
Federate 1:	0.75	183	40.75	0	0.01	8	0.17	60	1	0
Federate 2:	0.00	100	50.00	0	0.00	0	0.00	10	0	1
Federate 3:	0.00	83	50.00	0	0.00	0	0.00	50	1	0
Federate 4:	1.51	213	43.20	0	0.03	15	0.20	63	1	0
Federate 5:	0.18	183	64.72	0	0.00	20	0.02	60	1	0
Federate 6:	0.61	265	64.35	0	0.02	65	0.02	70	1	0
Federate 7:	0.18	182	84.74	0	0.00	39	0.01	60	1	0
Federate 8:	0.24	265	80.85	0	0.01	99	0.01	70	1	0
Federate 9:	0.00	83	110.00	0	0.00	0	0.00	50	1	0
	0.50	1824	59.68	0	0.01	25	0.06	563		

random_5c_8s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	20	30.00	0	0.00	0	0.00	2	0	1
Federate 1:	0.18	168	31.38	0	0.00	2	0.11	53	1	0
Federate 2:	0.18	168	41.38	0	0.00	5	0.06	53	1	0
Federate 3:	0.00	138	50.00	0	0.00	0	-0.00	50	1	0
Federate 4:	1.03	518	40.18	0	0.05	72	0.07	92	1	0
Federate 5:	0.17	168	68.98	0	0.00	13	0.02	53	1	0
Federate 6:	0.80	337	59.55	0	0.03	85	0.03	72	1	0

Federate 7:	0.06	158	90.06	0	0.00	98	0.00	52	1	0
Federate 8:	0.00	20	100.00	0	0.00	0	0.00	2	0	0
Federate 9:		0								
	0.53	1695	52.17	0	0.01	27	0.03	429		

random_5c_9s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	6.84	671	36.84	0	0.46	58	0.79	120	2	0
Federate 1:	4.22	559	44.22	0	0.24	65	0.36	95	1	1
Federate 2:	1.77	471	49.41	0	0.08	64	0.13	100	2	0
Federate 3:	0.00	222	55.00	0	0.00	13	0.00	75	2	0
Federate 4:	1.46	421	59.37	0	0.06	66	0.09	95	2	0
Federate 5:	1.46	421	69.37	0	0.06	91	0.07	95	2	0
Federate 6:	1.47	421	74.65	0	0.06	99	0.06	95	2	0
Federate 7:	0.77	360	75.54	0	0.03	99	0.03	50	1	1
Federate 8:	3.62	271	98.09	0	0.10	99	0.10	80	2	0
Federate 9:	3.73	220	113.73	0	0.08	98	0.08	75	2	0
	2.90	4037	61.76	0	0.12	75	0.17	880		

random_5c_10s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.47	158	30.47	0	0.01	3	0.24	17	0	0
Federate 1:	1.14	198	39.12	0	0.02	8	0.26	21	0	0
Federate 2:	0.43	188	44.04	0	0.01	13	0.06	20	0	0
Federate 3:	0.05	130	46.98	0	0.00	10	0.01	13	0	0
Federate 4:	0.47	158	60.47	0	0.01	13	0.06	17	0	0
Federate 5:	1.14	198	69.12	0	0.02	32	0.07	21	0	0
Federate 6:	1.15	197	79.12	0	0.02	41	0.05	21	0	0
Federate 7:	0.48	157	90.48	0	0.01	34	0.02	17	0	0
Federate 8:	0.48	157	100.48	0	0.01	56	0.01	17	0	0
Federate 9:	0.00	87	110.00	0	0.00	37	0.00	10	0	0
	0.65	1628	64.70	0	0.01	25	0.08	174		

random_5c_1s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.27	500	32.27	0	0.11	39	0.29	50	0	0
Federate 1:	2.27	500	32.27	0	0.11	39	0.29	50	0	0
Federate 2:	0.61	330	30.61	0	0.02	9	0.20	33	0	0
Federate 3:	1.08	420	31.08	0	0.05	28	0.16	42	0	0
Federate 4:	2.47	500	32.47	0	0.12	39	0.31	50	0	0
Federate 5:	2.27	500	32.27	0	0.11	39	0.29	50	0	0
Federate 6:	2.32	500	32.32	0	0.12	39	0.30	50	0	0
Federate 7:	1.10	250	31.10	0	0.03	9	0.29	25	0	0
Federate 8:	2.27	500	32.27	0	0.11	39	0.29	50	0	0
Federate 9:		0								
	1.97	4000	31.97	0	0.08	28	0.24	400		

random_5c_2s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.36	607	32.36	0	0.14	51	0.28	61	0	0
Federate 1:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
Federate 2:	6.29	995	36.29	0	0.63	99	0.63	100	0	0
Federate 3:	859.50	997	889.50	431	85.69	99	86.00	121	0	0

Federate 4:	4.33	807	34.33	0	0.35	90	0.38	81	0	0
Federate 5:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
Federate 6:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
Federate 7:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
Federate 8:	859.50	997	889.50	431	85.69	99	86.00	121	0	0
Federate 9:	2.21	895	32.21	0	0.20	99	0.20	90	0	0
	555.10	9286	585.10	2586	51.55	93	51.75	1058		

random_5c_3s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 1:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 2:	0.00	70	30.00	0	0.00	3	0.00	7	0	0
Federate 3:	0.00	70	30.00	0	0.00	3	0.00	7	0	0
Federate 4:	2.75	220	32.75	0	0.06	10	0.59	22	0	0
Federate 5:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 6:	0.15	50	30.15	0	0.00	1	0.07	5	0	0
Federate 7:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 8:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 9:		0								
	2.07	1310	32.07	0	0.03	5	0.28	131		

random_5c_4s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.31	529	30.31	0	0.02	32	0.05	53	0	0
Federate 1:	1.13	469	31.13	0	0.05	17	0.31	47	0	0
Federate 2:	1.12	459	31.12	0	0.05	14	0.35	46	0	0
Federate 3:	1.13	469	31.13	0	0.05	17	0.31	47	0	0
Federate 4:	0.93	449	30.93	0	0.04	12	0.34	45	0	0
Federate 5:	1.13	469	31.13	0	0.05	17	0.31	47	0	0
Federate 6:	1.13	469	31.13	0	0.05	17	0.31	47	0	0
Federate 7:	1.13	469	31.13	0	0.05	17	0.31	47	0	0
Federate 8:	0.00	70	30.00	0	0.00	1	0.00	7	0	0
Federate 9:		0								
	0.97	3852	30.97	0	0.04	14	0.23	386		

random_5c_5s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	7.71	888	37.71	0	0.69	92	0.74	89	0	0
Federate 1:	7.71	888	37.71	0	0.69	92	0.74	89	0	0
Federate 2:	1.87	688	31.87	0	0.13	71	0.18	69	0	0
Federate 3:	7.71	888	37.71	0	0.69	92	0.74	89	0	0
Federate 4:	7.95	888	37.95	0	0.71	92	0.76	89	0	0
Federate 5:		0								
Federate 6:	7.77	888	37.77	0	0.69	92	0.74	89	0	0
Federate 7:	7.71	888	37.71	0	0.69	92	0.74	89	0	0
Federate 8:	1.46	450	31.46	0	0.07	22	0.29	45	0	0
Federate 9:	0.00	250	30.00	0	0.00	0	0.00	25	0	0
	6.45	6716	36.45	0	0.43	65	0.49	673		

random_5c_6s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.26	150	31.26	0	0.02	4	0.47	15	0	0

Federate 1:	1.26	150	31.26	0	0.02	4	0.47	15	0	0
Federate 2:	0.61	140	30.61	0	0.01	3	0.23	14	0	0
Federate 3:	1.26	150	31.26	0	0.02	4	0.47	15	0	0
Federate 4:	1.76	150	31.76	0	0.03	4	0.63	15	0	0
Federate 5:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 6:	1.69	599	31.69	0	0.10	27	0.36	60	0	0
Federate 7:	1.26	150	31.26	0	0.02	4	0.47	15	0	0
Federate 8:	0.83	140	30.83	0	0.01	3	0.32	14	0	0
Federate 9:		0								
	1.32	1699	31.32	0	0.02	5	0.35	170		

random_5c_7s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.15	698	31.15	0	0.08	58	0.14	70	0	0
Federate 1:	1.15	698	31.15	0	0.08	61	0.13	70	0	0
Federate 2:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
Federate 3:	1.15	698	31.15	0	0.08	62	0.13	70	0	0
Federate 4:	1.77	728	31.77	0	0.13	71	0.18	73	0	0
Federate 5:	1.15	698	31.15	0	0.08	65	0.12	70	0	0
Federate 6:	1.15	698	31.15	0	0.08	58	0.14	70	0	0
Federate 7:	1.15	698	31.15	0	0.08	61	0.13	70	0	0
Federate 8:	1.15	698	31.15	0	0.08	62	0.13	70	0	0
Federate 9:	1.15	698	31.15	0	0.08	64	0.12	70	0	0
	1.21	6412	31.21	0	0.08	56	0.12	643		

random_5c_8s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	219	30.00	0	0.00	1	0.00	22	0	0
Federate 1:	1.88	727	31.88	0	0.14	74	0.18	73	0	0
Federate 2:	1.88	727	31.88	0	0.14	90	0.15	73	0	0
Federate 3:	0.00	508	30.00	0	0.00	66	0.00	51	0	0
Federate 4:	5.58	926	35.58	0	0.52	99	0.52	93	0	0
Federate 5:	1.88	727	31.88	0	0.14	80	0.17	73	0	0
Federate 6:	1.88	727	31.88	0	0.14	88	0.15	73	0	0
Federate 7:	1.88	727	31.88	0	0.14	84	0.16	73	0	0
Federate 8:	1.88	727	31.88	0	0.14	85	0.16	73	0	0
Federate 9:		0								
	2.23	6015	32.23	0	0.13	67	0.15	604		

random_5c_9s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 1:	7.93	947	37.93	0	0.75	99	0.75	95	0	0
Federate 2:	10.55	998	40.55	0	1.05	99	1.06	100	0	0
Federate 3:	5.11	948	35.11	0	0.48	99	0.49	95	0	0
Federate 4:	673.55	996	703.55	270	67.09	99	67.28	115	0	0
Federate 5:	5.11	948	35.11	0	0.48	99	0.49	95	0	0
Federate 6:	661.93	997	691.93	264	65.99	99	66.18	115	0	0
Federate 7:	2.36	500	32.36	0	0.12	27	0.44	50	0	0
Federate 8:	4.02	799	34.02	0	0.32	99	0.32	80	0	0
Federate 9:	5.11	948	35.11	0	0.48	99	0.49	95	0	0
	243.06	9079	273.06	955	22.07	92	22.16	960		

random_5c_10s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.87	250	31.87	0	0.05	10	0.43	25	0	0
Federate 1:	1.87	250	31.87	0	0.05	10	0.43	25	0	0
Federate 2:	1.60	200	31.60	0	0.03	10	0.31	20	0	0
Federate 3:	0.54	130	30.54	0	0.01	6	0.10	13	0	0
Federate 4:	2.25	250	32.25	0	0.06	11	0.51	25	0	0
Federate 5:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 6:	2.00	250	32.00	0	0.05	10	0.46	25	0	0
Federate 7:	1.87	250	31.87	0	0.05	10	0.43	25	0	0
Federate 8:	0.33	210	30.33	0	0.01	5	0.12	21	0	0
Federate 9:	1.87	250	31.87	0	0.05	10	0.43	25	0	0
	1.66	2250	31.66	0	0.04	9	0.39	225		

random_5c_1s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.98	400	30.98	0	0.04	20	0.19	40	0	0
Federate 1:	0.98	400	30.98	0	0.04	20	0.19	40	0	0
Federate 2:	0.00	250	30.00	0	0.00	0	0.00	25	0	0
Federate 3:	0.31	320	30.31	0	0.01	9	0.11	32	0	0
Federate 4:	1.94	500	31.94	0	0.10	39	0.25	50	0	0
Federate 5:	0.31	320	30.31	0	0.01	9	0.11	32	0	0
Federate 6:	0.94	350	30.94	0	0.03	13	0.24	35	0	0
Federate 7:	0.00	50	30.00	0	0.00	0	0.00	5	0	0
Federate 8:	0.83	420	33.21	0	0.03	26	0.13	42	0	0
Federate 9:		0								
	0.87	3010	31.21	0	0.03	13	0.12	301		

random_5c_2s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.96	598	30.96	0	0.06	55	0.10	61	0	0
Federate 1:	421.50	994	453.24	0	41.90	99	42.13	110	0	0
Federate 2:	1.33	498	31.33	0	0.07	24	0.27	50	0	0
Federate 3:	50.64	996	80.74	0	5.04	99	5.07	101	0	0
Federate 4:	0.05	110	31.87	0	0.00	0	1.39	11	0	0
Federate 5:	456.42	995	490.16	0	45.41	99	45.71	111	0	0
Federate 6:	1.34	508	32.13	0	0.07	29	0.23	51	0	0
Federate 7:	51.93	996	82.43	0	5.17	99	5.22	101	0	0
Federate 8:	1.44	508	32.62	0	0.07	31	0.23	51	0	0
Federate 9:	0.00	398	30.00	0	0.00	0	-0.00	40	0	0
	148.15	6601	179.25	0	9.78	53	10.04	687		

random_5c_3s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.04	80	30.04	0	0.00	2	0.01	8	0	0
Federate 1:	0.98	60	32.65	0	0.01	1	0.59	6	0	0
Federate 2:	2.07	30	38.74	0	0.01	1	0.62	3	0	0
Federate 3:	0.00	10	60.00	0	0.00	0	0.00	1	0	0
Federate 4:	0.25	180	32.47	0	0.00	6	0.07	18	0	0
Federate 5:	0.00	130	33.85	0	0.00	1	0.00	13	0	0
Federate 6:	0.00	50	42.00	0	0.00	0	0.00	5	0	0
Federate 7:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 8:	0.00	90	34.44	0	0.00	0	0.00	9	0	0
Federate 9:		0								
	0.26	650	34.11	0	0.00	1	0.13	65		

random_5c_4s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.31	529	30.31	0	0.02	32	0.05	53	0	0
Federate 1:	0.37	429	30.37	0	0.02	7	0.21	43	0	0
Federate 2:	0.82	439	30.82	0	0.04	9	0.37	44	0	0
Federate 3:	1.13	469	31.13	0	0.05	17	0.31	47	0	0
Federate 4:	0.47	50	30.47	0	0.00	1	0.21	5	0	0
Federate 5:	0.00	70	30.00	0	0.00	1	0.00	7	0	0
Federate 6:	0.00	70	30.00	0	0.00	1	0.00	7	0	0
Federate 7:	0.37	429	30.37	0	0.02	7	0.21	43	0	0
Federate 8:	0.00	60	30.00	0	0.00	0	0.00	6	0	0
Federate 9:		0								
	0.55	2545	30.55	0	0.01	7	0.14	255		

random_5c_5s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	3.70	788	33.70	0	0.29	82	0.35	79	0	0
Federate 1:	2.70	748	32.70	0	0.20	75	0.27	75	0	0
Federate 2:	1.53	648	31.53	0	0.10	64	0.15	65	0	0
Federate 3:	2.70	748	32.70	0	0.20	75	0.27	75	0	0
Federate 4:	6.55	848	36.55	0	0.56	88	0.63	85	0	0
Federate 5:		0								
Federate 6:	3.71	788	33.71	0	0.29	82	0.35	79	0	0
Federate 7:	6.41	848	36.41	0	0.54	87	0.62	85	0	0
Federate 8:	1.46	450	31.46	0	0.07	23	0.28	45	0	0
Federate 9:	2.70	748	32.70	0	0.20	75	0.27	75	0	0
	3.71	6614	33.71	0	0.25	65	0.32	663		

random_5c_6s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.26	150	31.26	0	0.02	4	0.47	15	0	0
Federate 1:	1.26	150	31.26	0	0.02	4	0.47	15	0	0
Federate 2:	0.00	60	30.00	0	0.00	1	0.00	6	0	0
Federate 3:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 4:	1.54	20	31.54	0	0.00	1	0.28	2	0	0
Federate 5:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 6:	1.69	599	31.69	0	0.10	28	0.35	60	0	0
Federate 7:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 8:	0.83	140	30.83	0	0.01	3	0.32	14	0	0
Federate 9:		0								
	1.18	1329	31.18	0	0.02	4	0.20	133		

random_5c_7s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.15	698	31.15	0	0.08	60	0.13	70	0	0
Federate 1:	1.35	598	31.35	0	0.08	53	0.15	60	0	0
Federate 2:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
Federate 3:	0.00	498	30.00	0	0.00	51	0.00	50	0	0
Federate 4:	1.06	628	31.06	0	0.07	63	0.10	63	0	0
Federate 5:	1.35	598	31.35	0	0.08	51	0.16	60	0	0
Federate 6:	1.15	698	31.15	0	0.08	59	0.14	70	0	0
Federate 7:	1.35	598	31.35	0	0.08	37	0.21	60	0	0
Federate 8:	1.15	698	31.15	0	0.08	49	0.16	70	0	0

Federate 9:	0.00	498	30.00	0	0.00	34	0.00	50	0	0
	0.98	5612	30.98	0	0.05	46	0.11	563		

random_5c_8s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 1:	0.35	518	30.35	0	0.02	50	0.04	53	0	0
Federate 2:	0.19	518	30.39	0	0.01	61	0.02	53	0	0
Federate 3:	0.00	488	30.00	0	0.00	57	0.00	50	0	0
Federate 4:	4.77	906	34.77	0	0.43	99	0.43	92	0	0
Federate 5:	0.17	518	30.56	0	0.01	58	0.02	53	0	0
Federate 6:	1.44	707	31.44	0	0.10	83	0.12	72	0	0
Federate 7:	0.18	508	30.18	0	0.01	42	0.02	52	0	0
Federate 8:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 9:		0								
	1.38	4203	31.45	0	0.06	45	0.06	429		

random_5c_9s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 1:	7.42	947	37.95	0	0.70	99	0.71	95	0	0
Federate 2:	9.55	998	40.55	0	0.95	99	0.96	100	0	0
Federate 3:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
Federate 4:	5.49	948	35.49	0	0.52	99	0.52	95	0	0
Federate 5:	5.11	948	35.11	0	0.48	99	0.49	95	0	0
Federate 6:	5.17	948	35.17	0	0.49	99	0.49	95	0	0
Federate 7:	1.00	500	34.00	0	0.05	22	0.23	50	0	0
Federate 8:	4.02	799	36.52	0	0.32	99	0.32	80	0	0
Federate 9:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
	102.27	8584	132.85	421	8.78	91	8.82	880		

random_5c_10s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
Federate 1:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 2:	1.60	200	31.60	0	0.03	10	0.31	20	0	0
Federate 3:	0.54	130	30.54	0	0.01	6	0.10	13	0	0
Federate 4:	0.20	170	30.20	0	0.00	3	0.10	17	0	0
Federate 5:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 6:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 7:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
Federate 8:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
Federate 9:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
	0.80	1810	30.80	0	0.01	4	0.26	181		

random_5c_1s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.51	400	31.51	0	0.06	20	0.30	40	0	0
Federate 1:	1.51	400	31.51	0	0.06	20	0.30	40	0	0
Federate 2:	0.00	250	30.00	0	0.00	0	0.00	25	0	0
Federate 3:	0.13	320	30.13	0	0.00	11	0.04	32	0	0
Federate 4:	2.47	500	32.47	0	0.12	39	0.31	50	0	0
Federate 5:	0.13	320	30.13	0	0.00	11	0.04	32	0	0

Federate 6:	1.33	350	31.33	0	0.05	13	0.35	35	0	0
Federate 7:	0.00	50	30.00	0	0.00	0	0.00	5	0	0
Federate 8:	1.08	420	31.08	0	0.05	28	0.16	42	0	0
Federate 9:		0								
	1.14	3010	31.14	0	0.03	14	0.15	301		

random_5c_2s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.36	607	32.36	0	0.14	51	0.28	61	0	0
Federate 1:	455.19	995	485.19	0	45.29	99	45.50	110	0	0
Federate 2:	1.33	498	31.33	0	0.07	24	0.27	50	0	0
Federate 3:	50.69	996	80.69	0	5.05	99	5.07	101	0	0
Federate 4:	0.21	110	30.21	0	0.00	1	0.21	11	0	0
Federate 5:	491.02	996	521.02	9	48.91	99	49.08	111	0	0
Federate 6:	1.41	508	31.41	0	0.07	26	0.27	51	0	0
Federate 7:	50.69	996	80.69	0	5.05	99	5.07	101	0	0
Federate 8:	1.41	508	31.41	0	0.07	26	0.27	51	0	0
Federate 9:	0.00	398	30.00	0	0.00	0	-0.00	40	0	0
	158.27	6612	188.27	9	10.47	52	10.60	687		

random_5c_3s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.04	80	30.04	0	0.00	2	0.01	8	0	0
Federate 1:	0.00	60	30.00	0	0.00	1	0.00	6	0	0
Federate 2:	0.00	30	30.00	0	0.00	0	0.00	3	0	0
Federate 3:	0.00	10	30.00	0	0.00	0	0.00	1	0	0
Federate 4:	1.36	180	31.36	0	0.02	9	0.27	18	0	0
Federate 5:	0.45	130	30.45	0	0.01	4	0.14	13	0	0
Federate 6:	0.15	50	30.15	0	0.00	1	0.07	5	0	0
Federate 7:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 8:	0.65	90	30.65	0	0.01	1	0.59	9	0	0
Federate 9:		0								
	0.57	650	30.57	0	0.00	1	0.11	65		

random_5c_4s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.31	529	30.31	0	0.02	32	0.05	53	0	0
Federate 1:	0.37	429	30.37	0	0.02	7	0.21	43	0	0
Federate 2:	0.82	439	30.82	0	0.04	9	0.37	44	0	0
Federate 3:	1.13	469	31.13	0	0.05	17	0.31	47	0	0
Federate 4:	0.47	50	30.47	0	0.00	1	0.21	5	0	0
Federate 5:	0.00	70	30.00	0	0.00	1	0.00	7	0	0
Federate 6:	0.00	70	30.00	0	0.00	1	0.00	7	0	0
Federate 7:	0.37	429	30.37	0	0.02	7	0.21	43	0	0
Federate 8:	0.00	60	30.00	0	0.00	0	0.00	6	0	0
Federate 9:		0								
	0.55	2545	30.55	0	0.01	7	0.14	255		

random_5c_5s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	3.70	788	33.70	0	0.29	82	0.35	79	0	0
Federate 1:	2.70	748	32.70	0	0.20	75	0.27	75	0	0
Federate 2:	1.53	648	31.53	0	0.10	64	0.15	65	0	0

Federate 3:	2.70	748	32.70	0	0.20	75	0.27	75	0	0
Federate 4:	6.55	848	36.55	0	0.56	88	0.63	85	0	0
Federate 5:		0								
Federate 6:	3.71	788	33.71	0	0.29	82	0.35	79	0	0
Federate 7:	6.41	848	36.41	0	0.54	87	0.62	85	0	0
Federate 8:	1.46	450	31.46	0	0.07	26	0.25	45	0	0
Federate 9:	0.00	250	30.00	0	0.00	0	0.00	25	0	0
	3.68	6116	33.68	0	0.23	58	0.29	613		

random_5c_6s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.83	140	30.83	0	0.01	3	0.32	14	0	0
Federate 1:	1.26	150	31.26	0	0.02	4	0.47	15	0	0
Federate 2:	0.00	60	30.00	0	0.00	1	0.00	6	0	0
Federate 3:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 4:	1.54	20	31.54	0	0.00	1	0.28	2	0	0
Federate 5:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 6:	1.69	599	31.69	0	0.10	27	0.36	60	0	0
Federate 7:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 8:	0.83	140	30.83	0	0.01	3	0.32	14	0	0
Federate 9:		0								
	1.13	1319	31.13	0	0.01	4	0.19	132		

random_5c_7s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.15	698	31.15	0	0.08	60	0.13	70	0	0
Federate 1:	1.35	598	31.35	0	0.08	53	0.15	60	0	0
Federate 2:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
Federate 3:	0.00	498	30.00	0	0.00	51	0.00	50	0	0
Federate 4:	1.06	628	31.06	0	0.07	63	0.10	63	0	0
Federate 5:	1.35	598	31.35	0	0.08	51	0.16	60	0	0
Federate 6:	1.15	698	31.15	0	0.08	59	0.14	70	0	0
Federate 7:	1.35	598	31.35	0	0.08	37	0.21	60	0	0
Federate 8:	1.15	698	31.15	0	0.08	49	0.16	70	0	0
Federate 9:	0.00	498	30.00	0	0.00	34	0.00	50	0	0
	0.98	5612	30.98	0	0.05	46	0.11	563		

random_5c_8s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 1:	0.38	528	30.38	0	0.02	52	0.04	53	0	0
Federate 2:	0.38	528	30.38	0	0.02	57	0.03	53	0	0
Federate 3:	0.00	498	30.00	0	0.00	50	0.00	50	0	0
Federate 4:	4.47	916	34.47	0	0.41	99	0.41	92	0	0
Federate 5:	0.38	528	30.38	0	0.02	53	0.04	53	0	0
Federate 6:	1.41	717	31.41	0	0.10	79	0.13	72	0	0
Federate 7:	0.04	518	30.04	0	0.00	37	0.01	52	0	0
Federate 8:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 9:		0								
	1.34	4273	31.34	0	0.06	43	0.07	429		

random_5c_9s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 1:	7.93	947	37.93	0	0.75	99	0.75	95	0	0
Federate 2:	10.55	998	40.55	0	1.05	99	1.06	100	0	0
Federate 3:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
Federate 4:	5.49	948	35.49	0	0.52	99	0.52	95	0	0
Federate 5:	5.11	948	35.11	0	0.48	99	0.49	95	0	0
Federate 6:	5.17	948	35.17	0	0.49	99	0.49	95	0	0
Federate 7:	2.36	500	32.36	0	0.12	27	0.44	50	0	0
Federate 8:	4.02	799	34.02	0	0.32	99	0.32	80	0	0
Federate 9:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
	102.52	8584	132.52	421	8.80	92	8.86	880		

random_5c_10s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
Federate 1:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 2:	1.60	200	31.60	0	0.03	10	0.31	20	0	0
Federate 3:	0.54	130	30.54	0	0.01	6	0.10	13	0	0
Federate 4:	0.20	170	30.20	0	0.00	3	0.10	17	0	0
Federate 5:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 6:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 7:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
Federate 8:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
Federate 9:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
	0.83	1740	30.83	0	0.01	4	0.26	174		

random_5c_1s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.98	400	30.98	0	0.04	20	0.19	40	0	0
Federate 1:	0.98	400	30.98	0	0.04	20	0.19	40	0	0
Federate 2:	0.00	250	30.00	0	0.00	0	0.00	25	0	0
Federate 3:	0.31	320	30.31	0	0.01	9	0.11	32	0	0
Federate 4:	1.94	500	31.94	0	0.10	39	0.25	50	0	0
Federate 5:	0.31	320	30.31	0	0.01	9	0.11	32	0	0
Federate 6:	0.94	350	30.94	0	0.03	13	0.24	35	0	0
Federate 7:	0.00	50	30.00	0	0.00	0	0.00	5	0	0
Federate 8:	0.83	420	33.21	0	0.03	26	0.13	42	0	0
Federate 9:		0								
	0.87	3010	31.21	0	0.03	13	0.12	301		

random_5c_2s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.96	598	30.96	0	0.06	55	0.10	61	0	0
Federate 1:	421.50	994	453.24	0	41.90	99	42.13	110	0	0
Federate 2:	1.33	498	31.33	0	0.07	24	0.27	50	0	0
Federate 3:	50.64	996	80.74	0	5.04	99	5.07	101	0	0
Federate 4:	0.05	110	31.87	0	0.00	0	1.39	11	0	0
Federate 5:	456.42	995	490.16	0	45.41	99	45.71	111	0	0
Federate 6:	1.34	508	32.13	0	0.07	29	0.23	51	0	0
Federate 7:	51.93	996	82.43	0	5.17	99	5.22	101	0	0
Federate 8:	1.44	508	32.62	0	0.07	31	0.23	51	0	0
Federate 9:	0.00	398	30.00	0	0.00	0	-0.00	40	0	0
	148.15	6601	179.25	0	9.78	53	10.04	687		

random_5c_3s.irlo

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.04	80	30.04	0	0.00	2	0.01	8	0	0
Federate 1:	0.98	60	32.65	0	0.01	1	0.59	6	0	0
Federate 2:	2.07	30	38.74	0	0.01	1	0.62	3	0	0
Federate 3:	0.00	10	60.00	0	0.00	0	0.00	1	0	0
Federate 4:	0.25	180	32.47	0	0.00	6	0.07	18	0	0
Federate 5:	0.00	130	33.85	0	0.00	1	0.00	13	0	0
Federate 6:	0.00	50	42.00	0	0.00	0	0.00	5	0	0
Federate 7:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 8:	0.00	90	34.44	0	0.00	0	0.00	9	0	0
Federate 9:		0								
	0.26	650	34.11	0	0.00	1	0.13	65		

random_5c_4s.irlo

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.31	529	30.31	0	0.02	32	0.05	53	0	0
Federate 1:	0.37	429	30.37	0	0.02	7	0.21	43	0	0
Federate 2:	0.82	439	30.82	0	0.04	9	0.37	44	0	0
Federate 3:	1.13	469	31.13	0	0.05	17	0.31	47	0	0
Federate 4:	0.47	50	30.47	0	0.00	1	0.21	5	0	0
Federate 5:	0.00	70	30.00	0	0.00	1	0.00	7	0	0
Federate 6:	0.00	70	30.00	0	0.00	1	0.00	7	0	0
Federate 7:	0.37	429	30.37	0	0.02	7	0.21	43	0	0
Federate 8:	0.00	70	30.00	0	0.00	1	0.00	7	0	0
Federate 9:		0								
	0.55	2555	30.55	0	0.01	7	0.14	256		

random_5c_5s.irlo

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	3.70	788	33.70	0	0.29	82	0.35	79	0	0
Federate 1:	2.70	748	32.70	0	0.20	75	0.27	75	0	0
Federate 2:	1.53	648	31.53	0	0.10	64	0.15	65	0	0
Federate 3:	2.70	748	32.70	0	0.20	75	0.27	75	0	0
Federate 4:	6.55	848	36.55	0	0.56	88	0.63	85	0	0
Federate 5:		0								
Federate 6:	3.71	788	33.71	0	0.29	82	0.35	79	0	0
Federate 7:	6.41	848	36.41	0	0.54	87	0.62	85	0	0
Federate 8:	1.46	450	31.46	0	0.07	23	0.28	45	0	0
Federate 9:	2.70	748	32.70	0	0.20	75	0.27	75	0	0
	3.71	6614	33.71	0	0.25	65	0.32	663		

random_5c_6s.irlo

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.26	150	31.26	0	0.02	4	0.47	15	0	0
Federate 1:	1.26	150	31.26	0	0.02	4	0.47	15	0	0
Federate 2:	0.00	60	30.00	0	0.00	1	0.00	6	0	0
Federate 3:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 4:	1.54	20	31.54	0	0.00	1	0.28	2	0	0
Federate 5:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 6:	1.69	599	31.69	0	0.10	28	0.35	60	0	0
Federate 7:	0.14	70	30.14	0	0.00	2	0.05	7	0	0
Federate 8:	0.83	140	30.83	0	0.01	3	0.32	14	0	0
Federate 9:		0								

	1.18	1329	31.18	0	0.02	4	0.20	133		

random_5c_7s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.15	698	31.15	0	0.08	60	0.13	70	0	0
Federate 1:	1.35	598	31.35	0	0.08	53	0.15	60	0	0
Federate 2:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
Federate 3:	0.00	498	30.00	0	0.00	51	0.00	50	0	0
Federate 4:	1.06	628	31.06	0	0.07	63	0.10	63	0	0
Federate 5:	1.35	598	31.35	0	0.08	51	0.16	60	0	0
Federate 6:	1.15	698	31.15	0	0.08	59	0.14	70	0	0
Federate 7:	1.35	598	31.35	0	0.08	37	0.21	60	0	0
Federate 8:	1.15	698	31.15	0	0.08	49	0.16	70	0	0
Federate 9:	0.00	498	30.00	0	0.00	34	0.00	50	0	0
	0.98	5612	30.98	0	0.05	46	0.11	563		

random_5c_8s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 1:	0.35	518	30.35	0	0.02	50	0.04	53	0	0
Federate 2:	0.19	518	30.39	0	0.01	61	0.02	53	0	0
Federate 3:	0.00	488	30.00	0	0.00	57	0.00	50	0	0
Federate 4:	4.77	906	34.77	0	0.43	99	0.43	92	0	0
Federate 5:	0.17	518	30.56	0	0.01	58	0.02	53	0	0
Federate 6:	1.44	707	31.44	0	0.10	83	0.12	72	0	0
Federate 7:	0.18	508	30.18	0	0.01	42	0.02	52	0	0
Federate 8:	0.00	20	30.00	0	0.00	0	0.00	2	0	0
Federate 9:		0								
	1.38	4203	31.45	0	0.06	45	0.06	429		

random_5c_9s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 1:	7.42	947	37.95	0	0.70	99	0.71	95	0	0
Federate 2:	9.55	998	40.55	0	0.95	99	0.96	100	0	0
Federate 3:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
Federate 4:	5.49	948	35.49	0	0.52	99	0.52	95	0	0
Federate 5:	5.11	948	35.11	0	0.48	99	0.49	95	0	0
Federate 6:	5.17	948	35.17	0	0.49	99	0.49	95	0	0
Federate 7:	1.00	500	34.00	0	0.05	22	0.23	50	0	0
Federate 8:	4.02	799	36.52	0	0.32	99	0.32	80	0	0
Federate 9:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
	102.27	8584	132.85	421	8.78	91	8.82	880		

random_5c_10s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	170	30.00	0	0.00	3	0.00	17	0	0
Federate 1:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 2:	1.60	200	31.60	0	0.03	10	0.31	20	0	0
Federate 3:	0.54	130	30.54	0	0.01	6	0.10	13	0	0
Federate 4:	0.20	170	30.20	0	0.00	3	0.10	17	0	0
Federate 5:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 6:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 7:	0.00	170	30.00	0	0.00	3	0.00	17	0	0

```
Federate 8:      0.00   170    30.00    0   0.00   3   0.00   17    0    0
Federate 9:      0.00   170    30.00    0   0.00   3   0.00   17    0    0
_____________________________________________________________________________
                 0.80  1810    30.80    0   0.01   4   0.26  181
```

Seed 1 connection sets

PP	2 10 2 4 8 0 2 2 7 0 1 3 4 5 6 8 0 2 5 7 0 1 3 4 5 7 8 0 3 8 4 0 1 4 6 0 7 25 8 0 1 2 3 4 5 6 8 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 2 9 0 1 3 4 5 6 7 8 9 1 2 5 9 0 1 3 4 5 6 7 8 9 1 3 8 9 0 1 2 4 5 6 7 8 9 1 7 25 9 0 1 2 3 4 5 6 8 9 1
LOC	2 10 8 0 1 3 4 5 6 7 8 1 2 2 8 0 1 3 4 5 6 7 8 1 2 5 8 0 1 3 4 5 6 7 8 1 3 8 8 0 1 2 4 5 6 7 8 1 7 25 8 0 1 2 3 4 5 6 8 1	**IRLOC**	2 10 2 4 8 0 2 2 7 0 1 3 4 5 6 8 1 2 5 7 0 1 3 4 5 7 8 2 3 8 4 0 1 4 6 3 7 25 8 0 1 2 3 4 5 6 8 1
ON	2 10 2 4 8 0 2 2 8 0 1 3 4 5 6 7 8 3 2 5 8 0 1 3 4 5 6 7 8 3 3 8 4 0 1 4 6 2 7 25 8 0 1 2 3 4 5 6 8 1		

Seed 2 connection sets

PP	0 10 6 2 3 4 6 7 8 0 0 50 4 1 3 5 7 0 2 1 7 0 3 4 5 6 7 8 0 2 20 3 0 1 5 0 4 40 9 0 1 2 3 5 6 7 8 9 0	**BR**	0 10 9 1 2 3 4 5 6 7 8 9 1 0 50 9 1 2 3 4 5 6 7 8 9 1 2 1 9 0 1 3 4 5 6 7 8 9 1 2 20 9 0 1 3 4 5 6 7 8 9 1 4 40 9 0 1 2 3 5 6 7 8 9 1
LOC	0 10 8 1 2 3 4 5 6 7 8 2 0 50 9 1 2 3 4 5 6 7 8 9 1 2 1 8 0 1 3 4 5 6 7 8 2 2 20 8 0 1 3 4 5 6 7 8 2 4 40 9 0 1 2 3 5 6 7 8 9 1	**IRLOC**	0 10 6 2 3 4 6 7 8 3 0 50 4 1 3 5 7 2 2 1 7 0 3 4 5 6 7 8 0 2 20 3 0 1 5 0 4 40 9 0 1 2 3 5 6 7 8 9 1
ON	0 10 6 2 3 4 6 7 8 3 0 50 4 1 3 5 7 2 2 1 7 0 3 4 5 6 7 8 0 2 20 3 0 1 5 0 4 40 9 0 1 2 3 5 6 7 8 9 1		

Seed 3 connection sets

PP	2 10 2 4 5 0 2 5 4 0 1 4 8 0 6 2 5 0 2 4 5 7 0 7 1 7 0 1 2 3 4 5 6 0 7 4 2 6 8 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 5 9 0 1 3 4 5 6 7 8 9 1 6 2 9 0 1 2 3 4 5 7 8 9 1 7 1 9 0 1 2 3 4 5 6 8 9 1 7 4 9 0 1 2 3 4 5 6 8 9 1
LOC	2 10 2 4 5 2 2 5 4 0 1 4 8 1 6 2 5 0 2 4 5 7 3 7 1 7 0 1 2 3 4 5 6 0 7 4 2 6 8 0	**IRLOC**	2 10 2 4 5 2 2 5 4 0 1 4 8 1 6 2 5 0 2 4 5 7 3 7 1 7 0 1 2 3 4 5 6 0 7 4 2 6 8 0

ON	2 10 2 4 5 2 2 5 4 0 1 4 8 1 6 2 5 0 2 4 5 7 3 7 1 7 0 1 2 3 4 5 6 0 7 4 2 6 8 0		

Seed 4 connection sets

PP	0 4 6 2 3 4 5 6 8 0 2 1 7 0 1 3 4 5 6 7 0 4 2 7 0 1 3 5 6 7 8 0 6 10 1 0 0 8 40 5 0 1 2 3 7 0	**BR**	0 4 9 1 2 3 4 5 6 7 8 9 1 2 1 9 0 1 3 4 5 6 7 8 9 1 4 2 9 0 1 2 3 5 6 7 8 9 1 6 10 9 0 1 2 3 4 5 7 8 9 1 8 40 9 0 1 2 3 4 5 6 7 9 1
LOC	0 4 8 1 2 3 4 5 6 7 8 1 2 1 8 0 1 3 4 5 6 7 8 1 4 2 8 0 1 2 3 5 6 7 8 1 6 10 1 0 0 8 40 8 0 1 2 3 4 5 6 7 1	**IRLOC**	0 4 6 2 3 4 5 6 8 2 2 1 7 0 1 3 4 5 6 7 3 4 2 7 0 1 3 5 6 7 8 3 6 10 1 0 0 8 40 5 0 1 2 3 7 1
ON	0 4 6 2 3 4 5 6 8 2 2 1 8 0 1 3 4 5 6 7 8 3 4 2 7 0 1 3 5 6 7 8 3 6 10 1 0 0 8 40 5 0 1 2 3 7 1		

Seed 5 connection sets

PP	2 10 3 4 7 8 0 2 10 7 0 1 3 4 6 7 8 0 5 25 9 0 1 2 3 4 6 7 8 9 0 8 4 2 0 6 0 8 40 7 0 1 2 3 4 6 7 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 10 9 0 1 3 4 5 6 7 8 9 1 5 25 9 0 1 2 3 4 6 7 8 9 1 8 4 9 0 1 2 3 4 5 6 7 9 1 8 40 9 0 1 2 3 4 5 6 7 9 1
LOC	2 10 7 0 1 3 4 6 7 8 1 2 10 7 0 1 3 4 6 7 8 1 5 25 9 0 1 2 3 4 6 7 8 9 2 8 4 7 0 1 2 3 4 6 7 1 8 40 7 0 1 2 3 4 6 7 1	**IRLOC**	2 10 3 4 7 8 2 2 10 8 0 1 3 4 6 7 8 9 1 5 25 9 0 1 2 3 4 6 7 8 9 1 8 4 2 0 6 3 8 40 8 0 1 2 3 4 6 7 9 1
ON	2 10 7 0 1 3 4 6 7 8 3 2 10 7 0 1 3 4 6 7 8 3 5 25 9 0 1 2 3 4 6 7 8 9 1 8 4 2 0 6 0 8 40 7 0 1 2 3 4 6 7 2		

Seed 6 connection sets

PP	1 50 1 6 0 2 1 8 0 1 3 4 5 6 7 8 0 5 8 4 0 1 6 8 0 6 5 7 0 1 2 3 5 7 8 0 8 1 7 1 2 3 4 5 6 7 0	**BR**	1 50 9 0 2 3 4 5 6 7 8 9 1 2 1 9 0 1 3 4 5 6 7 8 9 1 5 8 9 0 1 2 3 4 6 7 8 9 1 6 5 9 0 1 2 3 4 5 7 8 9 1 8 1 9 0 1 2 3 4 5 6 7 9 1

LOC	1 50 1 6 0 2 1 8 0 1 3 4 5 6 7 8 1 5 8 8 0 1 2 3 4 6 7 8 1 6 5 8 0 1 2 3 4 5 7 8 1 8 1 8 0 1 2 3 4 5 6 7 1	**IRLOC**	1 50 1 6 0 2 1 8 0 1 3 4 5 6 7 8 3 5 8 4 0 1 6 8 2 6 5 7 0 1 2 3 5 7 8 1 8 1 8 0 1 2 3 4 5 6 7 3
ON	1 50 1 6 0 2 1 8 0 1 3 4 5 6 7 8 3 5 8 4 0 1 6 8 2 6 5 7 0 1 2 3 5 7 8 1 8 1 8 0 1 2 3 4 5 6 7 3		

Seed 7 connection sets

PP	2 10 3 0 6 8 0 2 50 9 0 1 3 4 5 6 7 8 9 0 4 10 7 0 1 2 5 6 7 8 0 5 8 1 4 0 6 5 1 4 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 50 9 0 1 3 4 5 6 7 8 9 1 4 10 9 0 1 2 3 5 6 7 8 9 1 5 8 9 0 1 2 3 4 6 7 8 9 1 6 5 9 0 1 2 3 4 5 7 8 9 1
LOC	2 10 9 0 1 3 4 5 6 7 8 9 1 2 50 9 0 1 3 4 5 6 7 8 9 1 4 10 9 0 1 2 3 5 6 7 8 9 1 5 8 1 4 0 6 5 1 4 0	**IRLOC**	2 10 3 0 6 8 3 2 50 9 0 1 3 4 5 6 7 8 9 1 4 10 7 0 1 2 5 6 7 8 2 5 8 1 4 0 6 5 1 4 0
ON	2 10 3 0 6 8 3 2 50 9 0 1 3 4 5 6 7 8 9 1 4 10 7 0 1 2 5 6 7 8 2 5 8 1 4 0 6 5 1 4 0		

Seed 8 connection sets

PP	0 1 3 1 2 5 0 0 50 7 1 2 3 4 5 6 7 0 3 2 8 0 1 2 4 5 6 7 8 0 3 20 1 4 0 3 20 2 4 6 0	**BR**	0 1 9 1 2 3 4 5 6 7 8 9 1 0 50 9 1 2 3 4 5 6 7 8 9 1 3 2 9 0 1 2 4 5 6 7 8 9 1 3 20 9 0 1 2 4 5 6 7 8 9 1 3 20 9 0 1 2 4 5 6 7 8 9 1
LOC	0 1 8 1 2 3 4 5 6 7 8 1 0 50 8 1 2 3 4 5 6 7 8 1 3 2 8 0 1 2 4 5 6 7 8 1 3 20 1 4 0 3 20 8 0 1 2 4 5 6 7 8 1	**IRLOC**	0 1 3 1 2 5 0 0 50 7 1 2 3 4 5 6 7 1 3 2 8 0 1 2 4 5 6 7 8 3 3 20 1 4 0 3 20 2 4 6 2
ON	0 1 3 1 2 5 0 0 50 7 1 2 3 4 5 6 7 1 3 2 8 0 1 2 4 5 6 7 8 3 3 20 1 4 0 3 20 2 4 6 2		

Seed 9 connection sets

PP	1 25 9 0 2 3 4 5 6 7 8 9 0 3 20 4 0 1 4 6 0 5 5 5 0 1 2 7 8 0 7 50 9 0 1 2 3 4 5 6 8 9 0 8 20 5 0 1 2 5 7 0	**BR**	1 25 9 0 2 3 4 5 6 7 8 9 1 3 20 9 0 1 2 4 5 6 7 8 9 1 5 5 9 0 1 2 3 4 6 7 8 9 1 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 9 0 1 2 3 4 5 6 7 9 1
LOC	1 25 9 0 2 3 4 5 6 7 8 9 1 3 20 4 0 1 4 6 2 5 5 5 0 1 2 7 8 3 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 9 0 1 2 3 4 5 6 7 9 1	**IRLOC**	1 25 9 0 2 3 4 5 6 7 8 9 1 3 20 4 0 1 4 6 3 5 5 5 0 1 2 7 8 0 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 5 0 1 2 5 7 2
ON	1 25 9 0 2 3 4 5 6 7 8 9 2 3 20 4 0 1 4 6 0 5 5 5 0 1 2 7 8 0 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 5 0 1 2 5 7 3		

Seed 10 connection sets

PP	2 5 8 0 1 3 4 5 6 7 8 0 3 2 8 0 1 2 4 5 6 7 8 0 3 10 9 0 1 2 4 5 6 7 8 9 0 5 4 2 2 3 0 8 4 5 1 2 3 5 6 0	**BR**	2 5 9 0 1 3 4 5 6 7 8 9 1 3 2 9 0 1 2 4 5 6 7 8 9 1 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 9 0 1 2 3 4 6 7 8 9 1 8 4 9 0 1 2 3 4 5 6 7 9 1
LOC	2 5 9 0 1 3 4 5 6 7 8 9 1 3 2 9 0 1 2 4 5 6 7 8 9 1 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 9 0 1 2 3 4 6 7 8 9 1 8 4 9 0 1 2 3 4 5 6 7 9 1	**IRLOC**	2 5 9 0 1 3 4 5 6 7 8 9 1 3 2 9 0 1 2 4 5 6 7 8 9 1 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 2 2 3 3 8 4 5 1 2 3 5 6 2
ON	2 5 8 0 1 3 4 5 6 7 8 2 3 2 9 0 1 2 4 5 6 7 8 9 1 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 2 2 3 0 8 4 5 1 2 3 5 6 3		

6 connections, 2 groups

seed	measure	PP	BR	LOC	IRLOC	ON	comments
1	out. throt.	1	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	58.08	32.54	32.55	33.22	33.2	
	% extra	-33%	50%	32%	0%	2%	
2	out. throt.	3	0	1	1	1	4 federates' input weights are at least 100
	tmax	1	9	0	0	0	2 federates' input weights are at least 90

	avg. delivery	56.61	1632.81	36.89	36.61	36.61	
	% extra	-70%	8%	-38%	-37%	-37%	
3	out. throt.	1	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	47.61	32.77	32.42	32.51	32.51	
	% extra	-38%	110%	13%	0%	0%	
4	out. throt.	2	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	53.47	39.54	36.02	36.02	36.02	
	% extra	-49%	76%	0%	0%	0%	
5	out. throt.	2	0	0	0	0	
	tmax	1	0	0	0	0	
	avg. delivery	62.88	38.37	39.71	34.86	34.77	
	% extra	-54%	29%	-13%	7%	0%	
6	out. throt.	0	0	0	0	0	
	tmax	0	0	0	0	0	
	avg. delivery	45.65	33.5	33.24	34.64	34.64	
	% extra	0%	359%	268%	0%	0%	
7	out. throt.	1	0	0	0	0	Even with multicast, federate 2 is
	tmax	1	8	0	0	0	output throttled from multiple connections
	avg. delivery	58.34	289.99	31.66	31.66	31.66	
	% extra	-69%	56%	-16%	-16%	-16%	
8	out. throt.	2	0	0	0	0	Federate 4's point-to-point input weight
	tmax	0	8	1	1	1	is 117; federate 3 is still output throttled
	avg. delivery	61.01	694.86	105.95	126.02	105.95	under LOC due to multiple connections
	% extra	-70%	40%	-8%	11%	-8%	
9	out. throt.	2	0	0	0	0	4 federates' input weights are at least 100;
	tmax	1	8	2	2	2	2 federates' input weights are at least 90
	avg. delivery	61.25	671.06	163.86	163.86	163.86	
	% extra	-53%	7%	-3%	-3%	-3%	

10	out. throt.	0	0	0	0	0	
	tmax	2	0	0	0	0	
	avg. delivery	61.99	33.13	35.18	32.93	35.34	
	% extra	-5%	41%	13%	21%	2%	

Offline simulator raw output

random_6c_1s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.12	285	31.12	0	0.03	13	0.24	41	1	0
Federate 1:	0.64	275	40.64	0	0.02	22	0.08	40	1	0
Federate 2:	0.00	135	49.26	0	0.00	1	0.00	26	1	0
Federate 3:	0.72	205	56.81	0	0.01	16	0.09	33	1	0
Federate 4:	1.56	375	54.76	0	0.06	65	0.09	50	1	0
Federate 5:	0.26	194	76.65	0	0.00	52	0.01	32	1	0
Federate 6:	0.54	224	78.93	0	0.01	69	0.02	35	1	0
Federate 7:	0.00	60	76.67	0	0.00	0	0.00	6	0	1
Federate 8:	0.57	294	77.78	0	0.02	98	0.02	42	1	0
Federate 9:		0								
	0.76	2047	58.08	0	0.02	34	0.05	305		

random_6c_2s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.71	311	30.71	0	0.02	11	0.20	61	1	2
Federate 1:	2.38	433	39.33	0	0.10	38	0.27	150	3	0
Federate 2:	0.15	244	41.96	0	0.00	11	0.03	90	2	0
Federate 3:	0.05	254	48.79	0	0.00	28	0.00	101	2	0
Federate 4:	0.00	143	50.00	0	0.00	3	0.00	51	1	1
Federate 5:	0.76	441	57.47	0	0.03	88	0.04	151	3	0
Federate 6:	0.40	253	72.10	0	0.01	52	0.02	91	2	0
Federate 7:	0.49	253	76.97	0	0.01	80	0.02	101	2	0
Federate 8:	0.30	187	92.60	0	0.01	68	0.01	51	1	0
Federate 9:	0.00	110	110.00	0	0.00	98	0.00	40	1	0
	0.73	2629	56.61	0	0.02	48	0.06	887		

random_6c_3s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.35	280	30.35	0	0.01	8	0.11	48	1	0
Federate 1:	0.00	259	40.00	0	0.00	6	0.00	46	1	0
Federate 2:	0.84	229	49.97	0	0.02	12	0.15	43	1	0
Federate 3:	0.00	10	60.00	0	0.00	0	0.00	1	0	0
Federate 4:	1.29	180	41.29	0	0.02	5	0.39	18	0	0
Federate 5:	0.48	130	46.63	0	0.01	4	0.12	13	0	0
Federate 6:	0.00	249	56.39	0	0.00	47	0.00	45	1	0
Federate 7:	0.00	219	70.00	0	0.00	99	0.00	42	1	0
Federate 8:	0.00	90	51.11	0	0.00	4	0.00	9	0	1
Federate 9:		0								
	0.36	1646	47.61	0	0.01	18	0.08	265		

random_6c_4s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.62	496	31.62	0	0.08	37	0.21	93	2	0
Federate 1:	0.03	229	40.03	0	0.00	10	0.01	43	1	0
Federate 2:	1.01	405	44.93	0	0.04	55	0.07	84	2	0
Federate 3:	0.39	269	56.30	0	0.01	40	0.03	47	1	1
Federate 4:	0.96	216	51.42	0	0.02	12	0.16	45	1	0
Federate 5:	1.25	236	61.67	0	0.03	29	0.10	47	1	0
Federate 6:	1.25	236	71.67	0	0.03	58	0.05	47	1	0
Federate 7:	1.11	395	76.32	0	0.04	99	0.04	83	2	0
Federate 8:	0.00	60	83.33	0	0.00	6	0.00	6	0	1
Federate 9:		0								

	1.00	2542	53.47	0	0.03	34	0.07	495	

random_6c_5s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.46	371	31.46	0	0.05	23	0.23	83	2	0
Federate 1:	1.49	331	41.49	0	0.05	26	0.19	79	2	0
Federate 2:	0.60	231	50.60	0	0.01	21	0.06	69	2	0
Federate 3:	0.86	330	57.83	0	0.03	53	0.05	79	2	0
Federate 4:	1.01	430	59.38	0	0.04	83	0.05	89	2	0
Federate 5:	0.00	40	80.00	0	0.00	0	0.00	4	0	1
Federate 6:	1.49	369	75.56	0	0.06	98	0.06	83	2	0
Federate 7:	1.11	429	78.08	0	0.05	99	0.05	89	2	0
Federate 8:	0.71	310	81.39	0	0.02	99	0.02	45	1	1
Federate 9:	0.31	150	110.31	0	0.00	98	0.00	29	1	0
	1.06	2991	62.88	0	0.03	60	0.07	649		

random_6c_6s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.83	140	30.83	0	0.01	3	0.32	14	0	0
Federate 1:	1.93	150	41.26	0	0.03	6	0.46	15	0	0
Federate 2:	0.00	60	48.33	0	0.00	2	0.00	6	0	0
Federate 3:	0.10	70	57.24	0	0.00	2	0.02	7	0	0
Federate 4:	0.35	20	60.35	0	0.00	1	0.07	2	0	0
Federate 5:	0.08	120	53.42	0	0.00	6	0.02	12	0	0
Federate 6:	1.08	599	35.42	0	0.06	46	0.14	60	0	0
Federate 7:	1.57	70	84.43	0	0.01	2	0.55	7	0	0
Federate 8:	0.27	140	73.84	0	0.00	15	0.02	14	0	0
Federate 9:		0								
	0.89	1369	45.65	0	0.01	8	0.16	137		

random_6c_7s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.61	254	30.61	0	0.02	9	0.17	70	1	0
Federate 1:	0.44	177	40.44	0	0.01	7	0.10	60	1	0
Federate 2:	0.00	100	50.00	0	0.00	0	0.00	10	0	1
Federate 3:	0.00	77	50.00	0	0.00	0	0.00	50	1	0
Federate 4:	1.15	283	39.31	0	0.03	16	0.20	88	1	0
Federate 5:	0.45	177	64.80	0	0.01	23	0.03	60	1	0
Federate 6:	0.62	253	64.54	0	0.02	65	0.02	70	1	0
Federate 7:	0.46	175	84.80	0	0.01	43	0.02	60	1	0
Federate 8:	0.60	252	81.40	0	0.02	89	0.02	70	1	0
Federate 9:	0.00	76	110.00	0	0.00	0	-0.00	50	1	0
	0.56	1824	58.34	0	0.01	25	0.06	588		

random_6c_8s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	90	30.00	0	0.00	0	0.00	27	1	1
Federate 1:	0.66	238	34.45	0	0.02	10	0.15	78	2	0
Federate 2:	0.66	238	44.45	0	0.02	17	0.09	78	2	0
Federate 3:	0.00	138	50.00	0	0.00	0	-0.00	50	1	1
Federate 4:	1.18	368	49.76	0	0.04	58	0.07	117	2	0
Federate 5:	0.60	238	69.76	0	0.01	54	0.03	78	2	0
Federate 6:	0.78	297	71.35	0	0.02	91	0.03	97	2	0

Federate 7:	0.58	227	90.58	0	0.01	94	0.01	77	2	0
Federate 8:	0.00	89	100.00	0	0.00	14	0.00	27	1	0
Federate 9:	0.00	69	110.00	0	0.00	0	-0.00	25	1	0
	0.63	1992	61.01	0	0.01	34	0.04	654		

random_6c_9s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	6.84	671	36.84	0	0.46	58	0.78	120	2	0
Federate 1:	4.07	609	43.25	0	0.25	76	0.33	100	1	1
Federate 2:	1.99	521	48.90	0	0.10	74	0.14	105	2	0
Federate 3:	0.00	222	55.00	0	0.00	13	0.00	75	2	0
Federate 4:	1.71	471	58.78	0	0.08	77	0.10	100	2	0
Federate 5:	1.46	421	69.37	0	0.06	89	0.07	95	2	0
Federate 6:	1.47	421	74.65	0	0.06	99	0.06	95	2	0
Federate 7:	0.80	410	73.77	0	0.03	99	0.03	55	1	1
Federate 8:	3.56	321	94.21	0	0.11	99	0.11	85	2	0
Federate 9:	3.73	220	113.73	0	0.08	98	0.08	75	2	0
	2.90	4287	61.25	0	0.12	78	0.17	905		

random_6c_10s.pp
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.05	258	32.05	0	0.05	12	0.42	27	0	0
Federate 1:	2.57	298	41.23	0	0.08	18	0.40	31	0	0
Federate 2:	0.78	288	46.61	0	0.02	31	0.07	30	0	0
Federate 3:	0.05	130	46.98	0	0.00	8	0.01	13	0	0
Federate 4:	0.47	158	60.47	0	0.01	13	0.06	17	0	0
Federate 5:	1.14	198	69.12	0	0.02	32	0.07	21	0	0
Federate 6:	1.67	297	73.59	0	0.05	61	0.08	31	0	0
Federate 7:	0.96	257	83.18	0	0.02	70	0.04	27	0	0
Federate 8:	0.48	157	100.48	0	0.01	60	0.01	17	0	0
Federate 9:	0.00	87	110.00	0	0.00	35	0.00	10	0	0
	1.24	2128	61.99	0	0.03	34	0.12	224		

random_6c_1s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.35	510	32.35	0	0.12	41	0.29	51	0	0
Federate 1:	1.55	500	31.95	0	0.08	40	0.19	50	0	0
Federate 2:	0.70	340	30.99	0	0.02	12	0.19	34	0	0
Federate 3:	1.16	430	32.55	0	0.05	33	0.15	43	0	0
Federate 4:	1.45	500	32.65	0	0.07	44	0.16	50	0	0
Federate 5:	1.29	500	32.89	0	0.06	44	0.14	50	0	0
Federate 6:	1.35	500	33.35	0	0.07	48	0.14	50	0	0
Federate 7:	0.00	240	31.25	0	0.00	12	0.00	24	0	0
Federate 8:	1.37	500	33.77	0	0.07	48	0.14	50	0	0
Federate 9:		0								
	1.35	4020	32.55	0	0.05	32	0.14	402		

random_6c_2s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.96	598	30.96	0	0.06	55	0.10	61	0	1
Federate 1:	4.00	753	36.51	0	0.30	96	0.31	150	1	0
Federate 2:	1.18	564	32.65	0	0.07	49	0.13	90	1	0
Federate 3:	1.37	574	32.99	0	0.08	41	0.19	101	0	0

Federate 4:	0.17	176	50.17	0	0.00	2	0.12	51	1	0
Federate 5:	2.74	763	41.35	0	0.21	99	0.21	151	1	0
Federate 6:	1.28	573	42.05	0	0.07	96	0.08	91	1	0
Federate 7:	1.38	574	38.03	0	0.08	65	0.12	101	0	0
Federate 8:	0.93	491	40.60	0	0.05	57	0.08	51	0	0
Federate 9:	0.00	398	30.00	0	0.00	0	-0.00	40	0	0
	1.67	5464	36.89	0	0.09	56	0.13	887		

random_6c_3s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.77	579	31.77	0	0.10	42	0.24	58	0	0
Federate 1:	1.55	559	31.73	0	0.09	39	0.22	56	0	0
Federate 2:	0.33	429	31.26	0	0.01	11	0.12	43	0	0
Federate 3:	0.00	10	60.00	0	0.00	0	0.00	1	0	0
Federate 4:	0.48	180	34.92	0	0.01	8	0.10	18	0	0
Federate 5:	0.34	180	36.46	0	0.01	9	0.06	18	0	0
Federate 6:	0.84	449	32.18	0	0.04	17	0.21	45	0	0
Federate 7:	0.46	419	32.37	0	0.02	14	0.13	42	0	0
Federate 8:	0.00	190	32.11	0	0.00	6	0.00	19	0	0
Federate 9:		0								
	0.92	2995	32.42	0	0.03	15	0.11	300		

random_6c_4s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	7.67	927	37.67	0	0.71	99	0.71	93	0	0
Federate 1:	0.33	429	31.03	0	0.01	8	0.16	43	0	0
Federate 2:	5.20	837	35.20	0	0.44	99	0.44	84	0	0
Federate 3:	0.44	469	32.57	0	0.02	17	0.12	47	0	0
Federate 4:	0.79	449	33.24	0	0.04	10	0.35	45	0	0
Federate 5:	1.47	469	36.16	0	0.07	18	0.38	47	0	0
Federate 6:	0.64	469	36.83	0	0.03	15	0.19	47	0	0
Federate 7:	5.15	827	37.09	0	0.43	99	0.43	83	0	0
Federate 8:	0.00	60	83.33	0	0.00	6	0.00	6	0	0
Federate 9:		0								
	3.53	4936	36.02	0	0.17	37	0.28	495		

random_6c_5s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	3.99	709	33.99	0	0.28	68	0.41	83	0	0
Federate 1:	3.36	668	35.46	0	0.22	68	0.33	79	0	0
Federate 2:	1.57	569	32.98	0	0.09	55	0.16	69	0	0
Federate 3:	2.41	668	37.20	0	0.16	79	0.20	79	0	0
Federate 4:	2.57	768	38.56	0	0.20	96	0.20	89	0	0
Federate 5:	0.00	40	80.00	0	0.00	0	0.00	4	0	0
Federate 6:	3.40	708	42.95	0	0.24	97	0.25	83	0	0
Federate 7:	4.59	768	45.99	0	0.35	99	0.35	89	0	0
Federate 8:	1.10	449	48.78	0	0.05	99	0.05	45	0	0
Federate 9:	0.34	290	41.38	0	0.01	20	0.05	29	0	0
	2.85	5637	39.71	0	0.16	68	0.20	649		

random_6c_6s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	4.17	699	34.17	0	0.29	60	0.48	70	0	0

Federate 1:	2.81	200	32.81	0	0.06	5	1.12	20	0	0
Federate 2:	3.03	689	33.03	0	0.21	72	0.29	69	0	0
Federate 3:	4.17	699	34.17	0	0.29	74	0.39	70	0	0
Federate 4:	2.79	150	32.79	0	0.04	3	1.31	15	0	0
Federate 5:	2.47	619	32.47	0	0.15	62	0.25	62	0	0
Federate 6:	3.07	649	33.07	0	0.20	65	0.30	65	0	0
Federate 7:	2.69	649	32.69	0	0.17	65	0.27	65	0	0
Federate 8:	3.16	689	33.16	0	0.22	63	0.34	69	0	0
Federate 9:		0								
	3.24	5043	33.24	0	0.16	47	0.48	505		

random_6c_7s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.84	598	30.84	0	0.05	51	0.10	70	0	0
Federate 1:	1.00	498	31.00	0	0.05	38	0.13	60	0	0
Federate 2:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
Federate 3:	0.00	398	30.00	0	0.00	24	0.00	50	0	0
Federate 4:	2.89	777	32.89	0	0.22	74	0.30	88	0	0
Federate 5:	1.00	498	31.00	0	0.05	42	0.12	60	0	0
Federate 6:	0.84	598	32.51	0	0.05	71	0.07	70	0	0
Federate 7:	1.00	498	31.00	0	0.05	42	0.12	60	0	0
Federate 8:	0.84	598	34.18	0	0.05	86	0.06	70	0	0
Federate 9:	0.00	398	30.00	0	0.00	29	0.00	50	0	0
	1.06	4961	31.66	0	0.05	46	0.09	588		

random_6c_8s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	239	30.00	0	0.00	7	0.00	27	0	0
Federate 1:	1.99	737	32.26	0	0.15	76	0.19	78	0	0
Federate 2:	2.13	737	32.80	0	0.16	80	0.19	78	0	0
Federate 3:	0.00	488	30.00	0	0.00	42	0.00	50	0	0
Federate 4:	442.71	992	473.28	0	43.92	99	44.35	117	0	0
Federate 5:	2.22	737	33.58	0	0.16	81	0.20	78	0	0
Federate 6:	4.62	906	37.70	0	0.42	99	0.42	97	0	0
Federate 7:	2.25	727	33.90	0	0.16	80	0.20	77	0	0
Federate 8:	0.00	239	35.86	0	0.00	21	0.00	27	0	0
Federate 9:	0.00	219	30.00	0	0.00	4	0.00	25	0	0
	74.68	6021	105.95	0	4.50	59	4.56	654		

random_6c_9s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 1:	10.88	997	45.37	0	1.08	99	1.09	100	0	0
Federate 2:	244.58	998	279.81	0	24.41	99	24.53	105	0	0
Federate 3:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
Federate 4:	21.25	996	56.25	0	2.12	99	2.12	100	0	0
Federate 5:	5.11	947	41.42	0	0.48	99	0.49	95	0	0
Federate 6:	5.12	947	41.43	0	0.49	99	0.49	95	0	0
Federate 7:	1.80	549	51.76	0	0.10	99	0.10	55	0	0
Federate 8:	4.82	849	39.53	0	0.41	99	0.41	85	0	0
Federate 9:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
	129.12	8779	163.86	421	11.34	99	11.37	905		

random_6c_10s.lo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.71	270	31.71	0	0.05	20	0.22	27	0	0
Federate 1:	1.95	310	32.59	0	0.06	22	0.26	31	0	0
Federate 2:	0.38	300	33.05	0	0.01	30	0.04	30	0	0
Federate 3:	1.82	230	37.03	0	0.04	14	0.29	23	0	0
Federate 4:	1.19	270	33.41	0	0.03	21	0.15	27	0	0
Federate 5:	0.06	210	39.59	0	0.00	10	0.01	21	0	0
Federate 6:	2.02	310	40.41	0	0.06	26	0.24	31	0	0
Federate 7:	1.11	270	35.55	0	0.03	28	0.10	27	0	0
Federate 8:	1.11	270	36.30	0	0.03	27	0.11	27	0	0
Federate 9:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
	1.24	2540	35.18	0	0.03	20	0.14	254		

random_6c_1s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.73	410	31.73	0	0.07	21	0.33	41	0	0
Federate 1:	1.72	400	33.72	0	0.07	26	0.26	40	0	0
Federate 2:	0.15	260	30.53	0	0.00	1	0.38	26	0	0
Federate 3:	0.62	330	31.22	0	0.02	8	0.24	33	0	0
Federate 4:	2.16	500	35.36	0	0.11	46	0.23	50	0	0
Federate 5:	0.31	320	30.31	0	0.01	9	0.11	32	0	0
Federate 6:	0.75	350	37.61	0	0.03	22	0.12	35	0	0
Federate 7:	0.00	60	35.00	0	0.00	0	0.00	6	0	0
Federate 8:	0.83	420	33.21	0	0.03	26	0.13	42	0	0
Federate 9:		0								
	1.12	3050	33.22	0	0.03	16	0.18	305		

random_6c_2s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.96	598	30.96	0	0.06	55	0.10	61	0	1
Federate 1:	4.00	767	36.47	0	0.31	96	0.32	150	1	0
Federate 2:	2.18	579	32.18	0	0.13	38	0.33	90	0	0
Federate 3:	1.41	589	34.65	0	0.08	63	0.13	101	1	0
Federate 4:	0.10	191	40.68	0	0.00	28	0.01	51	0	0
Federate 5:	2.95	777	40.51	0	0.23	99	0.23	151	1	0
Federate 6:	1.35	589	36.67	0	0.08	62	0.13	91	0	0
Federate 7:	1.36	589	42.97	0	0.08	99	0.08	101	1	0
Federate 8:	0.90	498	41.14	0	0.04	63	0.07	51	0	0
Federate 9:	0.00	398	30.00	0	0.00	0	-0.00	40	0	0
	1.81	5575	36.61	0	0.10	60	0.14	887		

random_6c_3s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.31	479	30.31	0	0.02	19	0.08	48	0	0
Federate 1:	0.31	459	30.52	0	0.01	14	0.10	46	0	0
Federate 2:	0.33	429	31.26	0	0.01	11	0.12	43	0	0
Federate 3:	0.00	10	60.00	0	0.00	0	0.00	1	0	0
Federate 4:	0.42	180	34.87	0	0.01	4	0.15	18	0	0
Federate 5:	0.48	130	46.63	0	0.01	4	0.12	13	0	0
Federate 6:	0.84	449	32.18	0	0.04	17	0.21	45	0	0
Federate 7:	0.46	419	32.37	0	0.02	14	0.13	42	0	0
Federate 8:	0.00	90	34.44	0	0.00	0	0.00	9	0	0
Federate 9:		0								
	0.43	2645	32.51	0	0.01	8	0.09	265		

random_6c_4s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	7.67	927	37.67	0	0.71	99	0.71	93	0	0
Federate 1:	0.33	429	31.03	0	0.01	8	0.16	43	0	0
Federate 2:	5.20	837	35.20	0	0.44	99	0.44	84	0	0
Federate 3:	0.44	469	32.57	0	0.02	17	0.12	47	0	0
Federate 4:	0.79	449	33.24	0	0.04	10	0.35	45	0	0
Federate 5:	1.47	469	36.16	0	0.07	18	0.38	47	0	0
Federate 6:	0.64	469	36.83	0	0.03	15	0.19	47	0	0
Federate 7:	5.15	827	37.09	0	0.43	99	0.43	83	0	0
Federate 8:	0.00	60	83.33	0	0.00	6	0.00	6	0	0
Federate 9:		0								
	3.53	4936	36.02	0	0.17	37	0.28	495		

random_6c_5s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	4.00	828	34.00	0	0.33	92	0.36	83	0	0
Federate 1:	4.00	788	34.00	0	0.32	85	0.37	79	0	0
Federate 2:	2.02	688	32.02	0	0.14	66	0.21	69	0	0
Federate 3:	4.00	788	34.00	0	0.32	85	0.37	79	0	0
Federate 4:	6.71	888	36.71	0	0.60	91	0.65	89	0	0
Federate 5:	0.00	40	30.00	0	0.00	0	0.00	4	0	0
Federate 6:	4.78	828	35.27	0	0.40	92	0.43	83	0	0
Federate 7:	6.51	888	37.63	0	0.58	95	0.61	89	0	0
Federate 8:	1.46	450	35.91	0	0.07	32	0.20	45	0	0
Federate 9:	4.00	788	34.00	0	0.32	85	0.37	79	0	0
	4.38	6974	34.86	0	0.31	72	0.36	699		

random_6c_6s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.83	140	30.83	0	0.01	3	0.32	14	0	0
Federate 1:	1.83	150	32.49	0	0.03	4	0.59	15	0	0
Federate 2:	1.60	60	33.27	0	0.01	1	0.96	6	0	0
Federate 3:	2.86	70	38.58	0	0.02	1	1.02	7	0	0
Federate 4:	0.35	20	60.35	0	0.00	1	0.07	2	0	0
Federate 5:	0.06	120	36.72	0	0.00	5	0.01	12	0	0
Federate 6:	1.10	599	32.77	0	0.07	35	0.18	60	0	0
Federate 7:	0.10	70	47.24	0	0.00	1	0.07	7	0	0
Federate 8:	0.61	140	35.61	0	0.01	8	0.10	14	0	0
Federate 9:		0								
	1.06	1369	34.64	0	0.01	6	0.33	137		

random_6c_7s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.84	598	30.84	0	0.05	51	0.10	70	0	0
Federate 1:	1.00	498	31.00	0	0.05	38	0.13	60	0	0
Federate 2:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
Federate 3:	0.00	398	30.00	0	0.00	24	0.00	50	0	0
Federate 4:	2.89	777	32.89	0	0.22	74	0.30	88	0	0
Federate 5:	1.00	498	31.00	0	0.05	42	0.12	60	0	0
Federate 6:	0.84	598	32.51	0	0.05	71	0.07	70	0	0
Federate 7:	1.00	498	31.00	0	0.05	42	0.12	60	0	0
Federate 8:	0.84	598	34.18	0	0.05	86	0.06	70	0	0

Federate 9:	0.00	398	30.00	0	0.00	29	0.00	50	0	0
	1.06	4961	31.66	0	0.05	46	0.09	588		

random_6c_8s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	269	30.00	0	0.00	3	0.00	27	0	0
Federate 1:	1.85	767	31.85	0	0.14	88	0.16	78	0	0
Federate 2:	1.72	767	31.85	0	0.13	89	0.15	78	0	0
Federate 3:	0.00	488	30.00	0	0.00	46	0.00	50	0	0
Federate 4:	687.11	996	717.11	295	68.44	99	68.76	117	0	0
Federate 5:	1.71	767	31.97	0	0.13	89	0.15	78	0	0
Federate 6:	6.68	955	36.68	0	0.64	99	0.64	97	0	0
Federate 7:	1.73	757	31.73	0	0.13	87	0.15	77	0	0
Federate 8:	1.73	757	31.73	0	0.13	87	0.15	77	0	0
Federate 9:	1.73	757	31.73	0	0.13	88	0.15	77	0	0
	95.98	7280	126.02	295	6.99	77	7.03	756		

random_6c_9s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 1:	10.88	997	45.37	0	1.08	99	1.09	100	0	0
Federate 2:	244.58	998	279.81	0	24.41	99	24.53	105	0	0
Federate 3:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
Federate 4:	21.25	996	56.25	0	2.12	99	2.12	100	0	0
Federate 5:	5.11	947	41.42	0	0.48	99	0.49	95	0	0
Federate 6:	5.12	947	41.43	0	0.49	99	0.49	95	0	0
Federate 7:	1.80	549	51.76	0	0.10	99	0.10	55	0	0
Federate 8:	4.82	849	39.53	0	0.41	99	0.41	85	0	0
Federate 9:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
	129.12	8779	163.86	421	11.34	99	11.37	905		

random_6c_10s.irlo
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.36	270	32.36	0	0.06	20	0.30	27	0	0
Federate 1:	3.69	310	33.69	0	0.11	22	0.50	31	0	0
Federate 2:	2.81	300	32.81	0	0.08	26	0.31	30	0	0
Federate 3:	3.62	230	35.36	0	0.08	10	0.78	23	0	0
Federate 4:	2.52	270	32.52	0	0.07	21	0.32	27	0	0
Federate 5:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 6:	3.69	310	33.69	0	0.11	22	0.50	31	0	0
Federate 7:	2.36	270	32.36	0	0.06	20	0.30	27	0	0
Federate 8:	2.36	270	32.36	0	0.06	20	0.30	27	0	0
Federate 9:	2.36	270	32.36	0	0.06	20	0.30	27	0	0
	2.78	2710	32.93	0	0.08	19	0.43	271		

random_6c_1s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.94	510	32.94	0	0.15	40	0.37	51	0	0
Federate 1:	2.94	510	32.94	0	0.15	42	0.36	51	0	0
Federate 2:	0.64	340	30.64	0	0.02	12	0.17	34	0	0
Federate 3:	1.66	430	31.66	0	0.07	29	0.24	43	0	0
Federate 4:	3.17	510	33.17	0	0.16	40	0.40	51	0	0
Federate 5:	2.94	510	32.94	0	0.15	40	0.37	51	0	0

Federate 6:	2.32	500	32.32	0	0.12	39	0.30	50	0	0
Federate 7:	1.84	260	31.84	0	0.05	10	0.45	26	0	0
Federate 8:	2.94	510	32.94	0	0.15	40	0.37	51	0	0
Federate 9:	2.94	510	32.94	0	0.15	40	0.37	51	0	0
	2.54	4590	32.54	0	0.12	33	0.34	459		

random_6c_2s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	2.36	607	32.36	0	0.14	51	0.28	61	0	0
Federate 1:	1874.52	997	1904.52	736	186.89	99	187.55	161	0	0
Federate 2:	1411.71	996	1441.71	651	140.61	99	141.25	140	0	0
Federate 3:	1874.52	997	1904.52	736	186.89	99	187.55	161	0	0
Federate 4:	865.40	996	895.40	439	86.19	99	86.50	121	0	0
Federate 5:	1874.52	997	1904.52	736	186.89	99	187.55	161	0	0
Federate 6:	1874.52	997	1904.52	736	186.89	99	187.55	161	0	0
Federate 7:	1874.52	997	1904.52	736	186.89	99	187.55	161	0	0
Federate 8:	1874.52	997	1904.52	736	186.89	99	187.55	161	0	0
Federate 9:	1874.52	997	1904.52	736	186.89	99	187.55	161	0	0
	1602.81	9578	1632.81	6242	153.52	94	154.09	1449		

random_6c_3s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	3.27	619	33.27	0	0.20	53	0.38	62	0	0
Federate 1:	3.27	619	33.27	0	0.20	53	0.38	62	0	0
Federate 2:	0.89	469	30.89	0	0.04	13	0.31	47	0	0
Federate 3:	3.27	619	33.27	0	0.20	53	0.38	62	0	0
Federate 4:	3.49	619	33.49	0	0.22	53	0.41	62	0	0
Federate 5:	3.27	619	33.27	0	0.20	53	0.38	62	0	0
Federate 6:	2.43	599	32.43	0	0.15	48	0.30	60	0	0
Federate 7:	1.29	569	31.29	0	0.07	42	0.17	57	0	0
Federate 8:	2.38	220	32.38	0	0.05	10	0.52	22	0	0
Federate 9:	3.27	619	33.27	0	0.20	53	0.38	62	0	0
	2.77	5571	32.77	0	0.15	43	0.36	558		

random_6c_4s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	7.67	927	37.67	0	0.71	99	0.71	93	0	0
Federate 1:	12.61	967	42.61	0	1.22	99	1.22	97	0	0
Federate 2:	10.93	957	40.93	0	1.05	99	1.05	96	0	0
Federate 3:	1.83	569	31.83	0	0.10	37	0.28	57	0	0
Federate 4:	8.87	947	38.87	0	0.84	99	0.84	95	0	0
Federate 5:	12.61	967	42.61	0	1.22	99	1.22	97	0	0
Federate 6:	7.02	867	37.02	0	0.61	99	0.61	87	0	0
Federate 7:	12.61	967	42.61	0	1.22	99	1.22	97	0	0
Federate 8:	2.01	569	32.01	0	0.11	37	0.30	57	0	0
Federate 9:	12.61	967	42.61	0	1.22	99	1.22	97	0	0
	9.54	8704	39.54	0	0.83	87	0.87	873		

random_6c_5s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	10.36	928	40.36	0	0.96	96	1.00	93	0	0
Federate 1:	10.36	928	40.36	0	0.96	95	1.01	93	0	0
Federate 2:	2.64	728	32.64	0	0.19	74	0.26	73	0	0

Federate 3:	10.36	928	40.36	0	0.96	96	0.99	93	0	0
Federate 4:	10.71	928	40.71	0	0.99	96	1.03	93	0	0
Federate 5:	3.44	678	33.44	0	0.23	55	0.42	68	0	0
Federate 6:	10.47	928	40.47	0	0.97	95	1.01	93	0	0
Federate 7:	7.53	888	37.53	0	0.67	92	0.72	89	0	0
Federate 8:	2.52	490	31.71	0	0.12	27	0.44	49	0	0
Federate 9:	10.36	928	40.36	0	0.96	96	1.00	93	0	0
	8.42	8352	38.37	0	0.70	82	0.79	837		

random_6c_6s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	4.17	699	34.17	0	0.29	61	0.48	70	0	0
Federate 1:	2.81	200	32.81	0	0.06	5	1.12	20	0	0
Federate 2:	3.03	689	33.03	0	0.21	65	0.32	69	0	0
Federate 3:	4.17	699	34.17	0	0.29	69	0.42	70	0	0
Federate 4:	4.50	699	34.50	0	0.31	67	0.46	70	0	0
Federate 5:	2.47	619	32.47	0	0.15	53	0.29	62	0	0
Federate 6:	3.07	649	33.07	0	0.20	55	0.36	65	0	0
Federate 7:	2.69	649	32.69	0	0.17	55	0.31	65	0	0
Federate 8:	3.16	689	33.16	0	0.22	64	0.34	69	0	0
Federate 9:	4.17	699	34.17	0	0.29	69	0.42	70	0	0
	3.50	6291	33.50	0	0.22	56	0.45	630		

random_6c_7s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	372.01	998	402.01	0	37.13	99	37.23	108	0	0
Federate 1:	372.01	998	402.01	0	37.13	99	37.23	108	0	0
Federate 2:	0.64	230	30.64	0	0.01	10	0.14	23	0	0
Federate 3:	372.01	998	402.01	0	37.13	99	37.23	108	0	0
Federate 4:	5.84	977	35.84	0	0.57	98	0.58	98	0	0
Federate 5:	4.84	997	34.84	0	0.48	99	0.48	100	0	0
Federate 6:	151.24	997	181.24	0	15.08	99	15.12	103	0	0
Federate 7:	372.01	998	402.01	0	37.13	99	37.23	108	0	0
Federate 8:	372.01	998	402.01	0	37.13	99	37.23	108	0	0
Federate 9:	372.01	998	402.01	0	37.13	99	37.23	108	0	0
	259.99	9189	289.99	0	23.89	90	23.97	972		

random_6c_8s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	667	30.00	0	0.00	57	0.00	67	0	0
Federate 1:	762.70	996	792.70	359	75.96	99	76.31	118	0	0
Federate 2:	762.70	996	792.70	359	75.96	99	76.31	118	0	0
Federate 3:	0.00	508	30.00	0	0.00	45	0.00	51	0	0
Federate 4:	764.39	996	794.39	365	76.13	99	76.48	118	0	0
Federate 5:	762.70	996	792.70	359	75.96	99	76.31	118	0	0
Federate 6:	762.70	996	792.70	359	75.96	99	76.31	118	0	0
Federate 7:	762.70	996	792.70	359	75.96	99	76.31	118	0	0
Federate 8:	762.70	996	792.70	359	75.96	99	76.31	118	0	0
Federate 9:	762.70	996	792.70	359	75.96	99	76.31	118	0	0
	664.86	9143	694.86	2878	60.79	90	61.07	1062		

random_6c_9s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t

Federate 0:	1009.98	998	1039.98	519	100.80	99	101.09	125	0	0
Federate 1:	13.98	997	43.98	0	1.39	99	1.40	100	0	0
Federate 2:	1009.98	998	1039.98	519	100.80	99	101.09	125	0	0
Federate 3:	251.55	998	281.55	0	25.10	99	25.18	105	0	0
Federate 4:	1026.79	996	1056.79	524	102.27	99	102.56	125	0	0
Federate 5:	656.25	998	686.25	258	65.49	99	65.68	115	0	0
Federate 6:	1015.51	997	1045.51	520	101.25	99	101.54	125	0	0
Federate 7:	7.95	749	37.95	0	0.60	72	0.82	75	0	0
Federate 8:	251.17	998	281.17	0	25.07	99	25.14	105	0	0
Federate 9:	1009.98	998	1039.98	519	100.80	99	101.09	125	0	0
	641.06	9727	671.06	2859	62.36	97	62.56	1125		

random_6c_10s.br
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	3.59	350	33.59	0	0.13	28	0.45	35	0	0
Federate 1:	3.59	350	33.59	0	0.13	28	0.45	35	0	0
Federate 2:	1.84	300	31.84	0	0.06	25	0.21	30	0	0
Federate 3:	2.53	230	32.53	0	0.06	12	0.48	23	0	0
Federate 4:	3.91	350	33.91	0	0.14	28	0.49	35	0	0
Federate 5:	1.63	210	31.63	0	0.03	5	0.68	21	0	0
Federate 6:	3.70	350	33.70	0	0.13	28	0.46	35	0	0
Federate 7:	3.59	350	33.59	0	0.13	28	0.45	35	0	0
Federate 8:	2.19	310	32.19	0	0.07	22	0.30	31	0	0
Federate 9:	3.59	350	33.59	0	0.13	28	0.45	35	0	0
	3.13	3150	33.13	0	0.10	23	0.44	315		

random_6c_1s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.73	410	31.73	0	0.07	21	0.33	41	0	0
Federate 1:	1.72	400	33.72	0	0.07	24	0.28	40	0	0
Federate 2:	0.15	260	30.53	0	0.00	1	0.38	26	0	0
Federate 3:	0.62	330	31.22	0	0.02	8	0.24	33	0	0
Federate 4:	2.16	500	35.36	0	0.11	46	0.23	50	0	0
Federate 5:	0.31	320	30.31	0	0.01	9	0.11	32	0	0
Federate 6:	1.03	400	37.03	0	0.04	29	0.14	40	0	0
Federate 7:	0.00	80	33.75	0	0.00	0	0.00	8	0	0
Federate 8:	0.83	420	33.21	0	0.03	26	0.13	42	0	0
Federate 9:		0								
	1.15	3120	33.20	0	0.04	16	0.18	312		

random_6c_2s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.96	598	30.96	0	0.06	55	0.10	61	0	1
Federate 1:	4.00	767	36.47	0	0.31	96	0.32	150	1	0
Federate 2:	2.18	579	32.18	0	0.13	38	0.33	90	0	0
Federate 3:	1.41	589	34.65	0	0.08	63	0.13	101	1	0
Federate 4:	0.10	191	40.68	0	0.00	28	0.01	51	0	0
Federate 5:	2.95	777	40.51	0	0.23	99	0.23	151	1	0
Federate 6:	1.35	589	36.67	0	0.08	62	0.13	91	0	0
Federate 7:	1.36	589	42.97	0	0.08	99	0.08	101	1	0
Federate 8:	0.90	498	41.14	0	0.04	63	0.07	51	0	0
Federate 9:	0.00	398	30.00	0	0.00	0	-0.00	40	0	0
	1.81	5575	36.61	0	0.10	60	0.14	887		

random_6c_3s.on

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.31	479	30.31	0	0.02	19	0.08	48	0	0
Federate 1:	0.31	459	30.52	0	0.01	14	0.10	46	0	0
Federate 2:	0.33	429	31.26	0	0.01	11	0.12	43	0	0
Federate 3:	0.00	10	60.00	0	0.00	0	0.00	1	0	0
Federate 4:	0.42	180	34.87	0	0.01	4	0.15	18	0	0
Federate 5:	0.48	130	46.63	0	0.01	4	0.12	13	0	0
Federate 6:	0.84	449	32.18	0	0.04	17	0.21	45	0	0
Federate 7:	0.46	419	32.37	0	0.02	14	0.13	42	0	0
Federate 8:	0.00	90	34.44	0	0.00	0	0.00	9	0	0
Federate 9:		0								
	0.43	2645	32.51	0	0.01	8	0.09	265		

random_6c_4s.on

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	7.67	927	37.67	0	0.71	99	0.71	93	0	0
Federate 1:	0.33	429	31.03	0	0.01	8	0.16	43	0	0
Federate 2:	5.20	837	35.20	0	0.44	99	0.44	84	0	0
Federate 3:	0.44	469	32.57	0	0.02	17	0.12	47	0	0
Federate 4:	0.79	449	33.24	0	0.04	10	0.35	45	0	0
Federate 5:	1.47	469	36.16	0	0.07	18	0.38	47	0	0
Federate 6:	0.64	469	36.83	0	0.03	15	0.19	47	0	0
Federate 7:	5.15	827	37.09	0	0.43	99	0.43	83	0	0
Federate 8:	0.00	60	83.33	0	0.00	6	0.00	6	0	0
Federate 9:		0								
	3.53	4936	36.02	0	0.17	37	0.28	495		

random_6c_5s.on

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	4.00	828	34.00	0	0.33	92	0.36	83	0	0
Federate 1:	4.00	788	34.00	0	0.32	85	0.37	79	0	0
Federate 2:	2.02	688	32.02	0	0.14	66	0.21	69	0	0
Federate 3:	4.00	788	34.00	0	0.32	85	0.37	79	0	0
Federate 4:	6.71	888	36.71	0	0.60	92	0.64	89	0	0
Federate 5:	0.00	40	30.00	0	0.00	0	0.00	4	0	0
Federate 6:	4.78	828	35.27	0	0.40	92	0.43	83	0	0
Federate 7:	6.51	888	37.63	0	0.58	95	0.61	89	0	0
Federate 8:	1.46	450	35.91	0	0.07	35	0.18	45	0	0
Federate 9:	0.34	290	30.34	0	0.01	5	0.20	29	0	0
	4.24	6476	34.77	0	0.27	65	0.34	649		

random_6c_6s.on

Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.83	140	30.83	0	0.01	3	0.32	14	0	0
Federate 1:	1.83	150	32.49	0	0.03	4	0.59	15	0	0
Federate 2:	1.60	60	33.27	0	0.01	1	0.96	6	0	0
Federate 3:	2.86	70	38.58	0	0.02	1	1.02	7	0	0
Federate 4:	0.35	20	60.35	0	0.00	1	0.07	2	0	0
Federate 5:	0.06	120	36.72	0	0.00	5	0.01	12	0	0
Federate 6:	1.10	599	32.77	0	0.07	35	0.18	60	0	0
Federate 7:	0.10	70	47.24	0	0.00	1	0.07	7	0	0
Federate 8:	0.61	140	35.61	0	0.01	8	0.10	14	0	0
Federate 9:		0								

	1.06	1369	34.64	0	0.01	6	0.33	137		

random_6c_7s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.84	598	30.84	0	0.05	51	0.10	70	0	0
Federate 1:	1.00	498	31.00	0	0.05	38	0.13	60	0	0
Federate 2:	0.00	100	30.00	0	0.00	0	0.00	10	0	0
Federate 3:	0.00	398	30.00	0	0.00	24	0.00	50	0	0
Federate 4:	2.89	777	32.89	0	0.22	74	0.30	88	0	0
Federate 5:	1.00	498	31.00	0	0.05	42	0.12	60	0	0
Federate 6:	0.84	598	32.51	0	0.05	71	0.07	70	0	0
Federate 7:	1.00	498	31.00	0	0.05	42	0.12	60	0	0
Federate 8:	0.84	598	34.18	0	0.05	86	0.06	70	0	0
Federate 9:	0.00	398	30.00	0	0.00	29	0.00	50	0	0
	1.06	4961	31.66	0	0.05	46	0.09	588		

random_6c_8s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	0.00	239	30.00	0	0.00	7	0.00	27	0	0
Federate 1:	1.99	737	32.26	0	0.15	76	0.19	78	0	0
Federate 2:	2.13	737	32.80	0	0.16	80	0.19	78	0	0
Federate 3:	0.00	488	30.00	0	0.00	42	0.00	50	0	0
Federate 4:	442.71	992	473.28	0	43.92	99	44.35	117	0	0
Federate 5:	2.22	737	33.58	0	0.16	81	0.20	78	0	0
Federate 6:	4.62	906	37.70	0	0.42	99	0.42	97	0	0
Federate 7:	2.25	727	33.90	0	0.16	80	0.20	77	0	0
Federate 8:	0.00	239	35.86	0	0.00	21	0.00	27	0	0
Federate 9:	0.00	219	30.00	0	0.00	4	0.00	25	0	0
	74.68	6021	105.95	0	4.50	59	4.56	654		

random_6c_9s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	840.62	998	870.62	421	83.89	99	84.13	120	0	0
Federate 1:	10.88	997	45.37	0	1.08	99	1.09	100	0	0
Federate 2:	244.58	998	279.81	0	24.41	99	24.53	105	0	0
Federate 3:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
Federate 4:	21.25	996	56.25	0	2.12	99	2.12	100	0	0
Federate 5:	5.11	947	41.42	0	0.48	99	0.49	95	0	0
Federate 6:	5.12	947	41.43	0	0.49	99	0.49	95	0	0
Federate 7:	1.80	549	51.76	0	0.10	99	0.10	55	0	0
Federate 8:	4.82	849	39.53	0	0.41	99	0.41	85	0	0
Federate 9:	2.49	749	32.49	0	0.19	99	0.19	75	0	0
	129.12	8779	163.86	421	11.34	99	11.37	905		

random_6c_10s.on
Averages for all federates

	tq	#_mess	avg_time	late	AQL	%TNE	AQL_NE	in_w	in_t	out_t
Federate 0:	1.71	270	31.71	0	0.05	20	0.22	27	0	0
Federate 1:	1.95	310	32.59	0	0.06	22	0.26	31	0	0
Federate 2:	0.38	300	33.05	0	0.01	30	0.04	30	0	0
Federate 3:	0.53	130	39.76	0	0.01	10	0.07	13	0	0
Federate 4:	0.15	170	33.68	0	0.00	6	0.04	17	0	0
Federate 5:	0.06	210	39.59	0	0.00	9	0.01	21	0	0
Federate 6:	2.02	310	40.41	0	0.06	26	0.24	31	0	0
Federate 7:	1.11	270	35.55	0	0.03	25	0.12	27	0	0

```
Federate 8:      0.08   170    38.31    0   0.00  12   0.01   17    0    0
Federate 9:      0.09   150    30.09    0   0.00   2   0.06   15    0    0
_____________________________________________________________________________
                 0.98  2290    35.34    0   0.02  16   0.11  229
```

Seed 1 connection sets

PP	2 10 2 4 8 0 2 2 7 0 1 3 4 5 6 8 0 2 5 7 0 1 3 4 5 7 8 0 3 8 4 0 1 4 6 0 6 1 4 0 2 3 7 0 7 25 8 0 1 2 3 4 5 6 8 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 2 9 0 1 3 4 5 6 7 8 9 1 2 5 9 0 1 3 4 5 6 7 8 9 1 3 8 9 0 1 2 4 5 6 7 8 9 1 6 1 9 0 1 2 3 4 5 7 8 9 1 7 25 9 0 1 2 3 4 5 6 8 9 1
LOC	2 10 8 0 1 3 4 5 6 7 8 2 2 2 7 0 1 3 4 5 6 8 0 2 5 8 0 1 3 4 5 6 7 8 2 3 8 8 0 1 2 4 5 6 7 8 2 6 1 4 0 2 3 7 0 7 25 8 0 1 2 3 4 5 6 8 1	**IRLOC**	2 10 2 4 8 0 2 2 7 0 1 3 4 5 6 8 1 2 5 7 0 1 3 4 5 7 8 2 3 8 4 0 1 4 6 0 6 1 4 0 2 3 7 0 7 25 8 0 1 2 3 4 5 6 8 1
ON	2 10 2 4 8 0 2 2 8 0 1 3 4 5 6 7 8 2 2 5 8 0 1 3 4 5 6 7 8 2 3 8 4 0 1 4 6 0 6 1 4 0 2 3 7 0 7 25 8 0 1 2 3 4 5 6 8 1		

Seed 2 connection sets

PP	0 10 6 2 3 4 6 7 8 0 0 50 4 1 3 5 7 0 0 40 5 1 2 4 5 6 0 2 1 7 0 3 4 5 6 7 8 0 2 20 3 0 1 5 0 4 40 9 0 1 2 3 5 6 7 8 9 0	**BR**	0 10 9 1 2 3 4 5 6 7 8 9 1 0 50 9 1 2 3 4 5 6 7 8 9 1 0 40 9 1 2 3 4 5 6 7 8 9 1 2 1 9 0 1 3 4 5 6 7 8 9 1 2 20 9 0 1 3 4 5 6 7 8 9 1 4 40 9 0 1 2 3 5 6 7 8 9 1
LOC	0 10 6 2 3 4 6 7 8 0 0 50 4 1 3 5 7 2 0 40 5 1 2 4 5 6 0 2 1 7 0 3 4 5 6 7 8 0 2 20 3 0 1 5 0 4 40 9 0 1 2 3 5 6 7 8 9 1	**IRLOC**	0 10 6 2 3 4 6 7 8 0 0 50 4 1 3 5 7 0 0 40 5 1 2 4 5 6 2 2 1 7 0 3 4 5 6 7 8 0 2 20 3 0 1 5 0 4 40 9 0 1 2 3 5 6 7 8 9 1
ON	0 10 6 2 3 4 6 7 8 0 0 50 4 1 3 5 7 0 0 40 5 1 2 4 5 6 2 2 1 7 0 3 4 5 6 7 8 0 2 20 3 0 1 5 0 4 40 9 0 1 2 3 5 6 7 8 9 1		

Seed 3 connection sets

PP	2 10 2 4 5 0 2 5 4 0 1 4 8 0 6 2 5 0 2 4 5 7 0 7 1 7 0 1 2 3 4 5 6 0 7 4 2 6 8 0 8 40 5 0 1 2 6 7 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 5 9 0 1 3 4 5 6 7 8 9 1 6 2 9 0 1 2 3 4 5 7 8 9 1 7 1 9 0 1 2 3 4 5 6 8 9 1 7 4 9 0 1 2 3 4 5 6 8 9 1 8 40 9 0 1 2 3 4 5 6 7 9 1
LOC	2 10 5 0 1 4 5 8 2 2 5 5 0 1 4 5 8 2 6 2 5 0 2 4 5 7 0 7 1 7 0 1 2 3 4 5 6 0 7 4 2 6 8 0 8 40 5 0 1 2 6 7 1	**IRLOC**	2 10 2 4 5 0 2 5 4 0 1 4 8 2 6 2 5 0 2 4 5 7 0 7 1 7 0 1 2 3 4 5 6 0 7 4 2 6 8 0 8 40 5 0 1 2 6 7 1
ON	2 10 2 4 5 0 2 5 4 0 1 4 8 2 6 2 5 0 2 4 5 7 0 7 1 7 0 1 2 3 4 5 6 0 7 4 2 6 8 0 8 40 5 0 1 2 6 7 1		

Seed 4 connection sets

PP	0 4 6 2 3 4 5 6 8 0 2 1 7 0 1 3 4 5 6 7 0 3 40 6 0 2 4 5 6 7 0 4 2 7 0 1 3 5 6 7 8 0 6 10 1 0 0 8 40 5 0 1 2 3 7 0	**BR**	0 4 9 1 2 3 4 5 6 7 8 9 1 2 1 9 0 1 3 4 5 6 7 8 9 1 3 40 9 0 1 2 4 5 6 7 8 9 1 4 2 7 9 0 1 2 3 5 6 7 8 9 1 6 10 9 0 1 2 3 4 5 7 8 9 1 8 40 9 0 1 2 3 4 5 6 7 9 1
LOC	0 4 6 2 3 4 5 6 8 0 2 1 7 0 1 3 4 5 6 7 0 3 40 6 0 2 4 5 6 7 1 4 2 7 0 1 3 5 6 7 8 0 6 10 1 0 0 8 40 5 0 1 2 3 7 2	**IRLOC**	0 4 6 2 3 4 5 6 8 0 2 1 7 0 1 3 4 5 6 7 0 3 40 6 0 2 4 5 6 7 1 4 2 7 0 1 3 5 6 7 8 0 6 10 1 0 0 8 40 5 0 1 2 3 7 2
ON	0 4 6 2 3 4 5 6 8 0 2 1 7 0 1 3 4 5 6 7 0 3 40 6 0 2 4 5 6 7 1 4 2 7 0 1 3 5 6 7 8 0 6 10 1 0 0 8 40 5 0 1 2 3 7 2		

Seed 5 connection sets

PP	2 10 3 4 7 8 0 2 10 7 0 1 3 4 6 7 8 0 5 25 9 0 1 2 3 4 6 7 8 9 0 8 4 2 0 6 0 8 40 7 0 1 2 3 4 6 7 0 8 4 9 0 1 2 3 4 5 6 7 9 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 10 9 0 1 3 4 5 6 7 8 9 1 5 25 9 0 1 2 3 4 6 7 8 9 1 8 4 9 0 1 2 3 4 5 6 8 9 1 8 40 9 0 1 2 3 4 5 6 7 9 1 8 4 9 0 1 2 3 4 5 6 7 9 1

LOC	2 10 3 4 7 8 0 2 10 7 0 1 3 4 6 7 8 0 5 25 9 0 1 2 3 4 6 7 8 9 2 8 4 2 0 6 0 8 40 7 0 1 2 3 4 6 7 1 8 4 9 0 1 2 3 4 5 6 7 9 0	**IRLOC**	2 10 3 4 7 8 0 2 10 8 0 1 3 4 6 7 8 9 1 5 25 9 0 1 2 3 4 6 7 8 9 1 8 4 2 0 6 0 8 40 8 0 1 2 3 4 6 7 9 1 8 4 9 0 1 2 3 4 5 6 7 9 2
ON	2 10 3 4 7 8 0 2 10 7 0 1 3 4 6 7 8 1 5 25 9 0 1 2 3 4 6 7 8 9 2 8 4 2 0 6 0 8 40 7 0 1 2 3 4 6 7 1 8 4 9 0 1 2 3 4 5 6 7 9 2		

Seed 6 connection sets

PP	1 50 1 6 0 2 1 8 0 1 3 4 5 6 7 8 0 5 8 4 0 1 6 8 0 6 5 7 0 1 2 3 5 7 8 0 7 5 1 5 0 8 1 7 1 2 3 4 5 6 7 0	**BR**	1 50 9 0 2 3 4 5 6 7 8 9 1 2 1 9 0 1 3 4 5 6 7 8 9 1 5 8 9 0 1 2 3 4 6 7 8 9 1 6 5 9 0 1 2 3 4 5 7 8 9 1 7 5 9 0 1 2 3 4 5 6 8 9 1 8 1 9 0 1 2 3 4 5 6 7 9 1
LOC	1 50 7 0 2 3 5 6 7 8 1 2 1 8 0 1 3 4 5 6 7 8 2 5 8 8 0 1 2 3 4 6 7 8 2 6 5 7 0 1 2 3 5 7 8 1 7 5 8 0 1 2 3 4 5 6 8 2 8 1 8 0 1 2 3 4 5 6 7 2	**IRLOC**	1 50 1 6 0 2 1 8 0 1 3 4 5 6 7 8 0 5 8 4 0 1 6 8 2 6 5 7 0 1 2 3 5 7 8 1 7 5 1 5 0 8 1 7 1 2 3 4 5 6 7 0
ON	1 50 1 6 0 2 1 8 0 1 3 4 5 6 7 8 0 5 8 4 0 1 6 8 2 6 5 7 0 1 2 3 5 7 8 1 7 5 1 5 0 8 1 7 1 2 3 4 5 6 7 0		

Seed 7 connection sets

PP	2 10 3 0 6 8 0 2 50 9 0 1 3 4 5 6 7 8 9 0 2 25 1 4 0 4 10 7 0 1 2 5 6 7 8 0 5 8 1 4 0 6 5 1 4 0	**BR**	2 10 9 0 1 3 4 5 6 7 8 9 1 2 50 9 0 1 3 4 5 6 7 8 9 1 2 25 9 0 1 3 4 5 6 7 8 9 1 4 10 9 0 1 2 3 5 6 7 8 9 1 5 8 9 0 1 2 3 4 6 7 8 9 1 6 5 9 0 1 2 3 4 5 7 8 9 1
LOC	2 10 3 0 6 8 0 2 50 9 0 1 3 4 5 6 7 8 9 1 2 25 1 4 0 4 10 7 0 1 2 5 6 7 8 2 5 8 1 4 0 6 5 1 4 0	**IRLOC**	2 10 3 0 6 8 0 2 50 9 0 1 3 4 5 6 7 8 9 1 2 25 1 4 0 4 10 7 0 1 2 5 6 7 8 2 5 8 1 4 0 6 5 1 4 0
ON	2 10 3 0 6 8 0 2 50 9 0 1 3 4 5 6 7 8 9 1 2 25 1 4 0 4 10 7 0 1 2 5 6 7 8 2 5 8 1 4 0 6 5 1 4 0		

Seed 8 connection sets

PP	0 1 3 1 2 5 0 0 50 7 1 2 3 4 5 6 7 0 3 2 8 0 1 2 4 5 6 7 8 0 3 20 1 4 0 3 20 2 4 6 0 3 25 9 0 1 2 4 5 6 7 8 9 0	**BR**	0 1 9 1 2 3 4 5 6 7 8 9 1 0 50 9 1 2 3 4 5 6 7 8 9 1 3 2 9 0 1 2 4 5 6 7 8 9 1 3 20 9 0 1 2 4 5 6 7 8 9 1 3 20 9 0 1 2 4 5 6 7 8 9 1 3 25 9 0 1 2 4 5 6 7 8 9 1
LOC	0 1 3 1 2 5 0 0 50 7 1 2 3 4 5 6 7 1 3 2 8 0 1 2 4 5 6 7 8 0 3 20 1 4 0 3 20 2 4 6 0 3 25 9 0 1 2 4 5 6 7 8 9 2	**IRLOC**	0 1 3 1 2 5 0 0 50 9 1 2 3 4 5 6 7 8 9 1 3 2 9 0 1 2 4 5 6 7 8 9 1 3 20 1 4 0 3 20 2 4 6 2 3 25 9 0 1 2 4 5 6 7 8 9 1
ON	0 1 3 1 2 5 0 0 50 7 1 2 3 4 5 6 7 1 3 2 8 0 1 2 4 5 6 7 8 0 3 20 1 4 0 3 20 2 4 6 0 3 25 9 0 1 2 4 5 6 7 8 9 2		

Seed 9 connection sets

PP	1 25 9 0 2 3 4 5 6 7 8 9 0 3 20 4 0 1 4 6 0 5 5 5 0 1 2 7 8 0 5 5 5 1 2 4 7 8 0 7 50 9 0 1 2 3 4 5 6 8 9 0 8 20 5 0 1 2 5 7 0	**BR**	1 25 9 0 2 3 4 5 6 7 8 9 1 3 20 9 0 1 2 4 5 6 7 8 9 1 5 5 9 0 1 2 3 4 6 7 8 9 1 5 5 9 0 1 2 3 4 6 7 8 9 1 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 9 0 1 2 3 4 5 6 7 9 1
LOC	1 25 9 0 2 3 4 5 6 7 8 9 2 3 20 4 0 1 4 6 0 5 5 5 0 1 2 7 8 0 5 5 5 1 2 4 7 8 0 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 5 0 1 2 5 7 0	**IRLOC**	1 25 9 0 2 3 4 5 6 7 8 9 2 3 20 4 0 1 4 6 0 5 5 5 0 1 2 7 8 0 5 5 5 1 2 4 7 8 0 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 5 0 1 2 5 7 0
ON	1 25 9 0 2 3 4 5 6 7 8 9 2 3 20 4 0 1 4 6 0 5 5 5 0 1 2 7 8 0 5 5 5 1 2 4 7 8 0 7 50 9 0 1 2 3 4 5 6 8 9 1 8 20 5 0 1 2 5 7 0		

Seed 10 connection sets

PP	2 5 8 0 1 3 4 5 6 7 8 0 3 2 8 0 1 2 4 5 6 7 8 0 3 10 9 0 1 2 4 5 6 7 8 9 0 5 4 2 2 3 0 5 10 5 0 1 2 6 7 0 8 4 5 1 2 3 5 6 0	**BR**	2 5 9 0 1 3 4 5 6 7 8 9 1 3 2 9 0 1 2 4 5 6 7 8 9 1 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 9 0 1 2 3 4 6 7 8 9 1 5 10 9 0 1 2 3 4 6 7 8 9 1 8 4 9 0 1 2 3 4 5 6 7 9 1

LOC	2 5 8 0 1 3 4 5 6 7 8 2 3 2 8 0 1 2 4 5 6 7 8 0 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 2 2 3 0 5 10 8 0 1 2 3 4 6 7 8 2 8 4 5 1 2 3 5 6 0	**IRLOC**	2 5 9 0 1 3 4 5 6 7 8 9 1 3 2 9 0 1 2 4 5 6 7 8 9 1 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 2 2 3 0 5 10 9 0 1 2 3 4 6 7 8 9 1 8 4 5 1 2 3 5 6 2
ON	2 5 9 0 1 3 4 5 6 7 8 9 1 3 2 8 0 1 2 4 5 6 7 8 0 3 10 9 0 1 2 4 5 6 7 8 9 1 5 4 2 2 3 0 5 10 5 0 1 2 6 7 2 8 4 5 1 2 3 5 6 0		

APPENDIX B: Large Connection Set Figures

Figure B-1. Pre-Engagement Snapshot

Figure B-2. Engagement Snapshot

Figure B-3. Post Engagement Snapshot