

AD-A255 250

2

ADST/WDL/TR-92-003010



AUG 1 1992

**Strawman
Distributed Interactive Simulation
Architecture Description Document
Volume II: Supporting Rationale
Book I: Time/Space Coherence
and Interoperability**

**Loral Systems Company
ADST Program Office
Orlando, Florida**

31 March 1992

Prepared for

**PMTRADE
Program Manager - Training Devices
Orlando, Florida**

LORAL

Systems Company

92-21858



92 8 6 031

ADST/WDL/TR--92-003010

Strawman Distributed Interactive Simulation Architecture Description Document Volume II: Supporting Rationale Book I: Time/Space Coherence and Interoperability

Loral Systems Company
ADST Program Office
12443 Research Parkway
Orlando, Florida 32826

31 March 1992

Contract No. N61339-91-D-0001
CDRL A002

Prepared for

PMTRADE
Program Manager - Training Devices
Naval Training Systems Center
12350 Research Parkway
Orlando, Florida 32826-3275

DTIC QUALITY INSPECTED 3

LORAL
Systems Company

Approved For	
Special	
Dist. and/or	
Availability Codes	
Dist. and/or	
Special	
A-1	

REPORT DOCUMENTATION PAGE			Form approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget Project (0704-0188), Washington, DC 20503.			
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 03/31/92	3. REPORT TYPE AND DATES COVERED Version 0 03/91 -03/92	
4. TITLE AND SUBTITLE Strawman Distributed Interactive Simulation Architecture Description Document Volume II: Supporting Rationale, Book I: Time/Space Coherence and Interoperability		5. FUNDING NUMBERS Contract No. N61339-91-D-0001 CDRL A002	
6. AUTHOR(S) Beaver, Ray; Brasch, Randy; Burdick, Chuck; Butler, Brett; Downes-Martin, Stephen; Eller, Tim; Ferguson, Bob; Gotuby, Dave; Huber, Joe; Katz, Warren; Miller, Bill; Morrison, John; Sherman, Richard; Stanley, Walt; Yelowitz, K.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Loral Systems Company ADST Program Office 12443 Research Parkway, Suite 303 Orlando, FL 32826		8. PERFORMING ORGANIZATION REPORT NUMBER ADST/WDL/TR-92-003010	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Project Manager for Training Devices PM TRADE Naval Training Systems Center 12350 Research Parkway Orlando, FL 32826-3275		10. SPONSORING ORGANIZATION REPORT	
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION/AVAILABILITY STATEMENT		12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The DIS architecture defines a time and space coherent representation of a virtual battlefield environment, measured in terms of the human perception and behaviors of warfighters interacting in free play. It provides a structure by which independently developed systems may interact with each other in a well managed and validated combat simulation environment during all phases of the development process. A fundamental objective of the DIS architecture is to provide a blueprint to guide development of a general purpose system meeting the needs of a wide range of users. The architecture must therefore support the broadest possible range of user needs. At the same time, the architecture and the associated standards must provide sufficient design definition to achieve the goal of transparent interoperability of a very wide range of simulators, simulations, and actual equipment operating on instrumented ranges.			
14. SUBJECT TERMS		15. NUMBER OF PAGES	
		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	17. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	17. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-80)
Prescribed by ANSI Std Z39-18
298-102

Table of Contents

1. Introduction.....	1
2. Interoperability and Time and Space Coherence in DIS.....	3
2.1 Introduction.....	3
2.2 Perception and Cueing in Simulation.....	4
2.2.1 General Cueing Requirements.....	5
2.2.2 Time/Space Coherence in Stand-Alone Simulation.....	6
2.3 DIS Paradigm.....	6
2.4 Time/Space Coherence in DIS—the "Fair Fight".....	7
2.4.1 Latency and Simultaneity Between Observers.....	8
2.4.2 Scene Integrity.....	8
2.4.3 Contributions to Inequity.....	10
2.5 A Correlation Construct.....	11
3. Temporal Correlation.....	13
3.1 Overview.....	13
3.2 Symptoms of Temporal Disruption.....	13
3.2.1 Tracking Error.....	13
3.2.2 Image Oscillation (Jumps And Jitter).....	13
3.2.3 Out-Of-Sequence Events.....	13
3.3 Latency—Cause of Temporal Disruption.....	13
3.3.1 Network Latency.....	14
3.3.2 Internal Processing Latency.....	24
3.3.3 End-to-end Latency.....	26
3.4 Correlation solutions.....	27
3.4.1 Timestamping.....	27
3.4.2 Remote Entity Approximation Algorithms (REAs).....	28
3.4.3 Smoothing.....	44
3.4.4 Timebase Correction.....	58
3.4.5 Forward Reckoning Algorithms.....	59
3.4.6 Object Handover.....	68
4. Visual/Sensor Correlation.....	71
4.1 Effects of Visual/Sensor Inconsistencies.....	72
4.1.1 Terrain.....	72
4.1.2 Visuals.....	73
4.1.3 High Resolution Sensors.....	74
4.1.4 Electronic Warfare.....	75
4.1.5 Environment.....	76
4.2 Terrain Data Processing.....	77
4.2.1 Off-line Data Base Processing.....	78
4.2.2 Real-time Simulator Processing.....	79
4.3 Correlation Metrics and Interoperability.....	79
5. Overload Management.....	82
5.1 Network Overload Management.....	82
5.1.1 DIS Systems Are Queueing Systems.....	82
5.1.2 What Queues Can be Overloaded?.....	82
5.1.3 Techniques for Overload Management.....	83

5.1.4	Architecture Support for Network Overload Management....	85
5.2	CPU.....	85
6.	Seamless Simulation.....	88
6.1	Introduction.....	88
6.2	Overview of Seamless Simulation Issues.....	94
6.3	Manned Platform Simulators.....	99
6.4	Computer Generated Forces.....	99
6.5	Unit Level Simulations and Wargames.....	99
6.5.1	Basic Interoperability Issues.....	99
6.5.2	Unit /Platform Translation.....	101
6.5.3	DeAggregation.....	104
6.5.4	Impact on the DIS Message Protocol.....	105
6.6	Live Ranges and Operational Platforms.....	106
6.6.1	Basic Interoperability Issues.....	106
6.6.2	Large Battlefield Infrastructure.....	109
6.6.3	Area Weapon Interaction.....	111
6.6.4	Indirect Fire Effects.....	113
6.6.5	Specialized Platforms.....	113
6.6.6	Major Technical Challenges.....	113
6.6.7	An Example Scenario.....	114
6.6.8	Operational Platforms.....	114
6.6.9	Impact on the DIS Standards.....	116
6.7	Conclusions.....	116
7.	Special Problems.....	117
7.1	Dynamic Terrain.....	117
7.1.1	What is Dynamic Terrain?.....	117
7.1.2	Scope.....	117
7.1.3	How Dynamic Terrain Relates to Space-Time Coherence and Interoperability.....	120
7.1.4	Technical Challenges.....	121
7.1.5	Approach(es).....	126
7.2	Electronic Warfare.....	128
7.2.1	Overview.....	128
7.2.2	Electronic Warfare and tactical communications.....	133
7.2.3	EW Databases.....	136
7.3	Exercise Management.....	139

1. Introduction

Volume II provides supporting rationale for the DIS architecture which is presented in Volume I. Some of the key technical problems inherent in DIS technology are identified and discussed in Volume II, in addition to supporting rationale for the proposed DIS Architecture as a suitable framework for describing, designing, and specifying "DIS-compliant" systems. Volume II is divided into two books, described below.

Book One addresses some of the issues surrounding the problem of interoperability and time/space coherence. This problem is central to DIS, and sharply differentiates the DIS environment from that of more traditional stand-alone simulation.

Book Two focuses on some of the major system components that must be considered in a DIS implementation—the network, Computer Generated Forces (CGF), and Higher-Order Models (HOM). Book Two also examines the software considerations that cut across all of these system components.

Taken as a whole, Volume II should be viewed as a collection of DIS position papers which provide supporting rationale for the architecture described in Volume I. In the two Books one will find in depth discussions of some of the relevant problems and in many cases potential solutions. One will also find minority opinions which shed light on the nature of the tradeoffs intrinsic to the world of Distributed Interactive Simulation.

The DIS architecture work draws heavily from the body of work in process by the DIS Conference for the Interoperability of Defense Simulation, which is jointly sponsored by the Army Program Manager for Training Devices (PM TRADE) and the Defense Advanced Research Project Agency (DARPA). Under the auspices of the DIS Conference, representatives of the military services, industry, and associated research organizations have been working toward definition of an industry standard for Protocol Data Unit (PDU) messages. These PDU messages provide the basic means of interaction between DIS simulation entities. The architecture described by the two volumes of this document is generally consistent with the work underway by the conference, but attempts to provide a capstone document which defines the context for the work underway. In some areas the strawman architecture extends beyond the work of the conference by proposing establishment of a specific reference model context for the PDU Standard, definition of some standard terminology for the components of the reference model, and creation of an additional standard governing the set of databases required to support DIS exercises.

Comments and recommendations for changes and improvements are welcome and encouraged. Comments may be submitted to:

Loral ADST Program Office
12443 Research Parkway

Suite 303
Orlando FL 32826
attn: DIS Architecture

Comments may also be submitted via the ADST Bulletin Board System. Post comments in the DIS Architecture Comments area of the BBS. This area is found within the following structure:

ADST Bulletin Board System
System Description and Technical Information
DIS Architecture
DIS Architecture Comments

If you are not already a registered user of the ADST BBS, send a registration request to:

Loral ADST Program Office
12443 Research Parkway
Suite 303
Orlando FL 32826
attn: BBS Administration

2. Interoperability and Time and Space Coherence in DIS

2.1 Introduction

This section defines the relationships between interoperability, time and space coherence, and DIS technology. We explore the nature of these relationships by describing the following: perception and cueing in simulation, the core principles of DIS technology (the operating paradigm), and manifestations of time/space disruption in DIS applications. We end this section by describing a correlation construct that promises to offer a consistent framework for discussing and ultimately measuring correlation between DIS entities. The later sections explore different dimensions of the interoperability problem.

For the purpose of discourse, we define our usage of the following three terms: actual battlefield, virtual battlefield, and physical realization. The *Actual Battlefield* is the combat reality that simulation technology attempts to replicate. Successful simulation will cause its participants to believe that they are immersed in the Actual Battlefield. The *Virtual Battlefield* (also referred to as the Electronic Battlefield) is the simulation illusion itself. The *Physical Realization* refers to the details and mechanics of the underlying networked simulation system which supports the illusion of the Virtual Battlefield.

Against the backdrop of these definitions, interoperability refers to the working together of the components of the Physical Realization to produce a harmonious and useful simulation exercise. To see this more clearly, one needs to reexamine the controlling vision for the the DIS architecture:

The DIS architecture defines a time and space coherent representation of a Virtual Battlefield environment, measured in terms of the human perception and behaviors of warfighters interacting in free play.

Three key points are made concerning the vision.

First, we must decouple the notions of fidelity and time/space coherence. Fidelity describes how well the Virtual Battlefield maps to the Actual Battlefield. The degree of fidelity can be objectively measured in such things as weapon ranges, vehicle dynamic performance, and terrain database correlation to actual terrain. Time/space coherence is concerned with preserving intact the simulation illusion. The degree of time/space coherence is also subject to objective measures—such as correct sequencing of discrete events (missile launch, missile flyout, missile impact), and correlation of a vehicle's position as perceived by different entities on the Virtual Battlefield. However, one can have a highly coherent, logically consistent Virtual Battlefield, and yet have low fidelity.

Second, time/space coherence must be evaluated in terms of human perception and behaviors. Time/space coherence becomes a quality of the

representation of the Virtual Battlefield, found only in this end-product depiction and its associated impact on the warfighter participants. This means that measurements of time/space coherence in the Physical Realization of DIS technology—computers, devices, networks, CIG systems, etc—must be tempered with considerations of human perception. Attempted measurements of time/space coherence, based on analysis of these implementation mechanics—measurements such as network delay time, database correlation degree, CIG transport delay time—serve only as approximations of the true degree of time/space coherence, and one must weigh these numbers against warfighter participant acceptance of the simulation system and its intended application.

Third, while time/space coherence has an objectively measurable aspect, the degree required for a simulation task is application dependent. The DIS user, in the context of his intended application, determines his required degree of time/space coherence. This coherence requirement defines the threshold for interoperability. If the components of the Physical Realization support the time/space coherence requirement, we say the implementation is interoperable.

2.2 Perception and Cueing in Simulation

To explore further, we must illuminate the role of perception (subject response) and cueing (system stimulus) in simulation. One finds that acceptable quality perception and cueing is application dependent. Acceptable cueing is shaped by the expectations of the training session, combat doctrinal experiment, or weapon test.

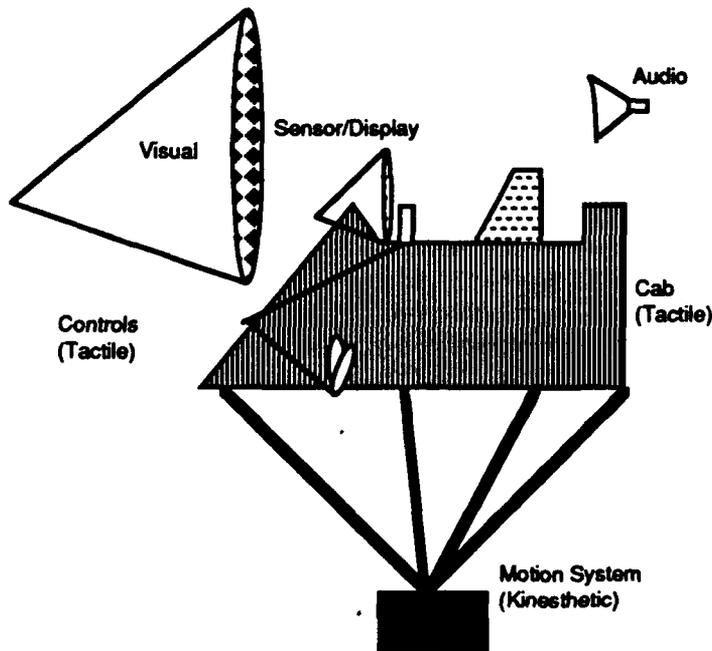


Fig 2.1: Simulator Cueing System

Fig. 2.1 depicts the components of the cueing system of a simulator. Establishment of acceptable time/space coherence means that the various cueing components collectively present an accurate picture of the state of the ownship simulator, as well as of the state of the Virtual Battlefield. "Accuracy" is defined by the simulation application and its objectives.

2.2.1 General Cueing Requirements

Within this wide range of potential expectations, the perception and cueing capabilities of any simulation implementation must demonstrate the following three capabilities:

- **Suspension of Disbelief.** The perception/cueing system must make the intended application "work". The simulated environment must present a rich variety of realistic cues to support the participant's total immersion into the Virtual Battlefield.
- **Selective Fidelity.** In any environment characterized by limited resources, successful systems are those optimized to meet their performance requirements while not dissipating resources on less important goals. The same holds true in simulation. The perception/cueing system must concentrate on faithfully rendering those aspects of reality which bear directly on the outcome of the intended application. These aspects must be enhanced, while other extraneous aspects, must be sacrificed.
- **Elimination of Erroneous Cues and Anomalies.** The cueing system must suppress erroneous cues and distracting anomalies. Intrusion of erroneous cues degrades the outcome of the simulation session, contributing to negative training and poor experimental results.

The last requirement, the elimination of erroneous cues, is where the onus of time/space coherence really comes home. This requirement can be further refined into the following:

- **Non-contradictory cue set.** The cue stimulations must be physically non-contradictory. The set of cues generated by an event in Virtual Reality must comply with the semantics of the event. For example, pulling back on the stick of a fighter-aircraft simulator should cause the following three cues to simultaneously occur: (1) visual scene rolls downward (to simulate upward pitch) (2) motion base tilts backwards (to simulate the cab pitch motion), and (3) the back and seat panels of the G-seat inflate (to simulate the increased g-loading experienced by the pilot). Failure of these three cues to respond as expected will cause pilot disorientation—distracting him from the simulation task of intent.
- **Correlated in time.** The cue stimulations must be acceptably sequenced and correlated in time. For example, a missile firing event in the same fighter-aircraft simulator should be accompanied by the following

sequence of cues: (1) motion platform/G-seat/G-suit stimulation to simulate the reaction forces of the missile leaving the airframe, (2) visual cues representing the missile flyout, (3) visual cues representing the missile's detonation.

2.2.2 Time/Space Coherence in Stand-Alone Simulation

Establishment of acceptable time/space coherence in a stand-alone simulator (not connected to an outside network) depends on faithful rendering of the cues corresponding to events caused only by the stand-alone host. One of the main obstacles to cue correlation in the stand-alone setting is the visual transport delay, the time that must elapse from when the visual scene is computationally determined--with its targets, terrain, cultural features, etc.--to when it is displayed. This time delay stems from the processing lags inherent in the CIG hardware. Typically, the lag ranges from about 200 ms for ground combat applications to 50 ms for high performance fixed wing applications.

Another obstacle to cue correlation is the use of high resolution sensors with Out The Window (OTW) Visuals. For example, Synthetic Aperture Radar (SAR) may have a range well beyond visual range, and yet exhibit a ground resolution measured in feet. Similarly, narrow field-of-view EO sensors such as FLIR's and LLLTV's may be able to see well beyond visual range under poor OTW visibility conditions (smoke, darkness) with very high resolution. More discussion is provided on this topic in paragraph 4.1.3.

2.3 DIS Paradigm

To understand the additional ramifications for time/space coherence to DIS technology, one must clearly understand the fundamentals of DIS. This subsection puts forth an axiomatic definition of DIS, and then highlights the time and space coherence problem in the light of this new paradigm.

The following four principles characterize DIS technology.

Distributed Autonomous Simulation Entities The Physical Realization of the Virtual Battlefield is distributed among disparate system components, disparate communication systems, and physically-separated sites. The collection of entities forming a DIS implementation are supported by a network of processing resources bundled into autonomous nodes--each node supporting one or several entities. Here, autonomy means that each host processor assumes sole responsibility to register its (battlefield) entity interactions with the Virtual Battlefield, and that it does so without depending on any higher level of coordination, synchronization, or control. A node processor may utilize network services and common servers to accomplish its tasks, but it does so on a demand basis, deciding for itself when to access the network.

"Ground Truth" Communication Protocol Each battlefield entity registers its interaction with the Virtual Battlefield by means of a common communication

protocol. Each entity transmits "ground truth", or an absolute frame-of-reference description of its current state and its combat operations with respect to the Virtual Battlefield.

Receiving Simulation Entity Determines Perception As events unfold on the Virtual Battlefield, perceptual cues are generated solely by the receiving simulation node. No network service or device participates in cue generation or display. Network resources function solely to maintain "ground truth." Depiction of that "ground truth" to the human subjects belongs solely to the host simulation node.

Deviation Reporting Simulation entities communicate only significant changes in their own state. This behavior carries two implications. First, entities communicate only about themselves: what they are doing and what is happening to them. Entities do not communicate their perceptions about the Virtual Battlefield or their perceptions about other players. Second, entities only communicate when appreciable changes have occurred to their state variables. Both of these points are slanted to ease the network communication burden. Point one has the additional impact of logically decoupling the interlocking pieces (entities) of the simulation, thus contributing to a cleaner object-oriented structure.

2.4 Time/Space Coherence in DIS—the "Fair Fight"

Time/space coherence for stand-alone simulators means primarily cue correlation. The consequences of poor cue correlation are negative training for training applications, and false experimental results for engineering design applications. In the realm of DIS technology however, additional causes for time/space disruption can occur, and the consequences of these disruptions can be more insidious—undermining the validity of the application.

DIS technology is vulnerable to time/space disruption from additional causes such as: network communication and processing latency, non-correlation of databases, and differences in Computer Image Generator (CIG) scene-rendering algorithms. Besides faulty cue correlation, the consequences of time/space disruption in DIS applications now include the specter of the "unfair fight"—a case in which each individual participant perceives a cohesive, plausible Virtual Battlefield, while time/space disruptions insidiously undermine the usefulness of the exercise. Assurance of a "fair fight" demands appropriately matching both fidelity and time/space coherence among the participating entities. The problem has corrupting dimensions because seemingly successful simulation exercises—characterized by consistent cue correlation for each of the participating entities—could have covertly tainted results for the intended application.

Successful attainment of the "fair fight" is an important aspect of interoperability.

2.4.1 Latency and Simultaneity Between Observers

Latency is one of the basic and most persistent causes of time/space disruption—persistent because it will always exist as a cause, given the distributed nature of the DIS paradigm.

On the Actual Battlefield, the occurrence of an event (plane crash, weapon release, bomb burst, etc.) is immediately and simultaneously apparent to all who possess the perceptual/sensor apparatus to observe it. There is an absolute battlefield frame of reference which defines ground truth. Each warfighter strains to perceive the absolute frame of reference, for accurate perception is a determinant of success or failure. However, on the Virtual Battlefield, observation of an event is dependent upon network communication.

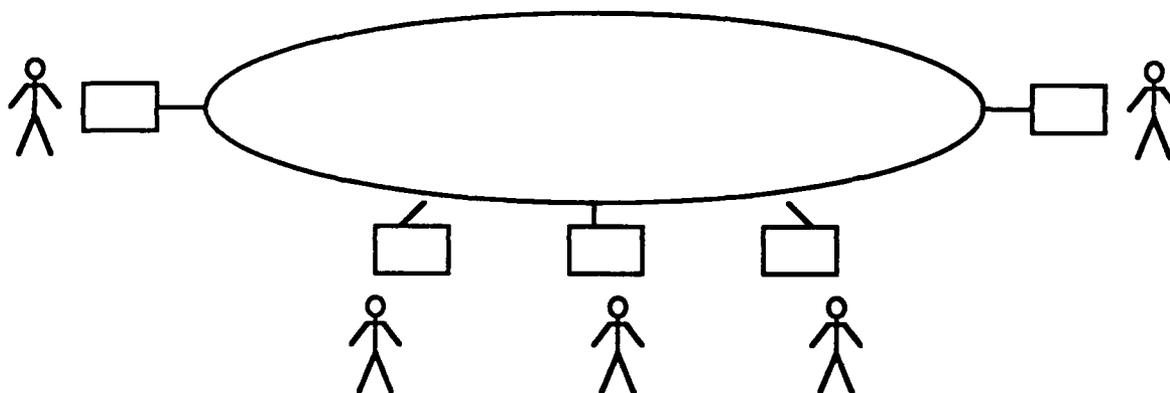


Fig 2.2: Simultaneity between observers

Figure 2.2 shows a virtual DIS network with human warfighters at several remote sites. The network exhibits "simultaneity" when all of the observers experience a given Virtual Battlefield event at the same time. However, due to the fact of network latency, simultaneity can only be approximated, never achieved. Clearly, an absolute frame of reference, common to all observers, can not exist since information takes time to flow through the network. While perfect simultaneity is not required for an effective simulation, reasonably good approximations of simultaneity are required. 300 and 700 ms have been suggested as upper bounds for air and ground applications, respectively.

2.4.2 Scene Integrity

Latency can be manifested as a loss of scene integrity.

The Virtual Battlefield is presented to the warfighter primarily through a sequence of visual frames. To understand how time/space disruption can affect the "fair fight", we need to examine the scene integrity of the presentation. We define "scene integrity" to be the logical consistency:

- Between components that make up a given "still" image frame, and

- Across a sequence of frames.

2.4.2.1 Scene Integrity Within a 'Still' Frame

Figure 2.3 depicts a scene presented to the warfighter. This scene is a composite of several different items:

- Ownship kinematic state
- Othership kinematic states
- Othership events (fire, detonation, etc.)
- Ownship events

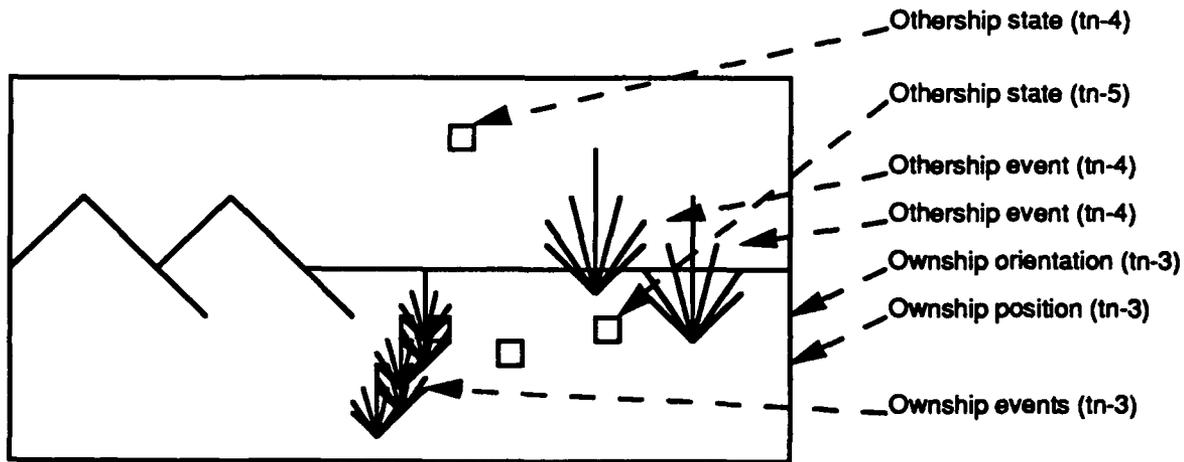
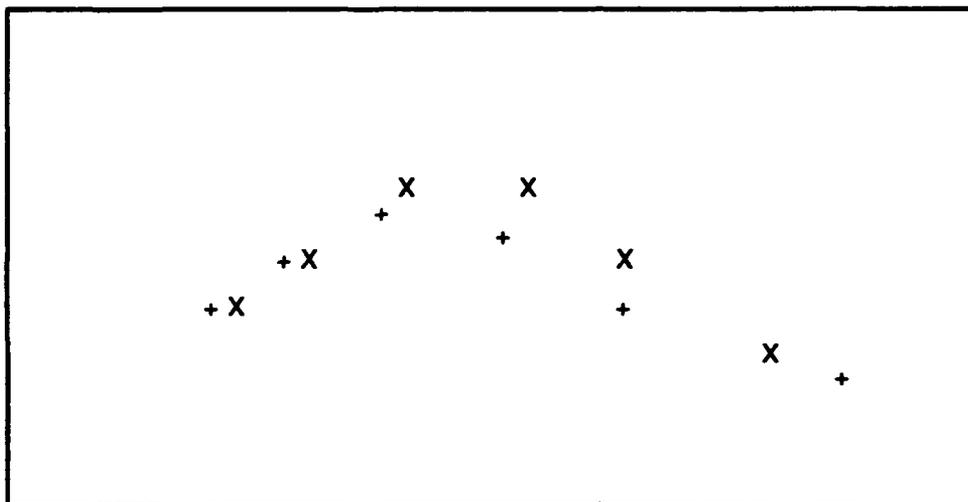


Fig 2.3: Scene integrity within a frame.

Each of these items has a "time" attached, relative to the current time (t_n). For instance, the positions of the other vehicles might be older (have lesser time value) than the ownship position because they were computed before the ownship position, and communicated to the ownship host node under conditions of network delay.

2.4.2.2 Scene Integrity Across Frames



X = Displayed Position
+ = Actual Reported Position

Fig 2.4: Sequential frames of display

Figure 2.4 depicts the sequence of positions ("track") of a given entity over successive frames. Several conditions which will be discussed in the following sections will cause the displayed and actual tracks to differ. In the worst case these displayed tracks may not reflect kinematic realities, destroying the simulation's illusion for the warfighter.

2.4.3 Contributions to Inequity

The consequences resulting from these latency-caused time/space disruptions may include a skewing of results or a tainting of the outcome from a DIS application. When warfighters are behaving according to their perceptions, and yet each warfighter's perspective is shifted from one another's because of time/space disruption on the Virtual Battlefield, then a recording of their aggregate actions may be confusing and inconsistent.

If the level of required time/space coherence is neither specified nor implemented to a stringent enough level, results from the experiment, exercise, or application cannot be trusted. An example best illustrates what can go wrong. Suppose one wishes to test the effectiveness of a new high-performance missile. A DIS application is designed to perform the test. The dynamic performance of the new missile is faithfully simulated. An exercise is conducted in which the friendly side, armed with the new missile, meets the enemy side, armed with more conventional weapons. The exercise is recorded and the outcome evaluated. However, suppose the record contains events such as friendly forces capitalizing on illegitimate firing opportunities because they "saw" the enemy through

conditions of network delay which lagged the enemy's successful and timely search for cover. The enemy would perceive successful evasion, while the friendly side would perceive successful missile impacts. What objective evaluations could missile designers make from such data?

2.5 A Correlation Construct

Time and space coherence can be represented notionally as a two dimensional correlation space, where the two axes are defined by time and space fidelity vectors for a given application or exercise. Each axis can in turn be viewed as a composite expression of the numerous factors that affect time and space coherence - time coherence by factors such as update rates, latencies, vehicle dynamics and network bandwidth, and space coherence by factors such as location, attitude, geometry, and appearance. Figure 2-5 illustrates.

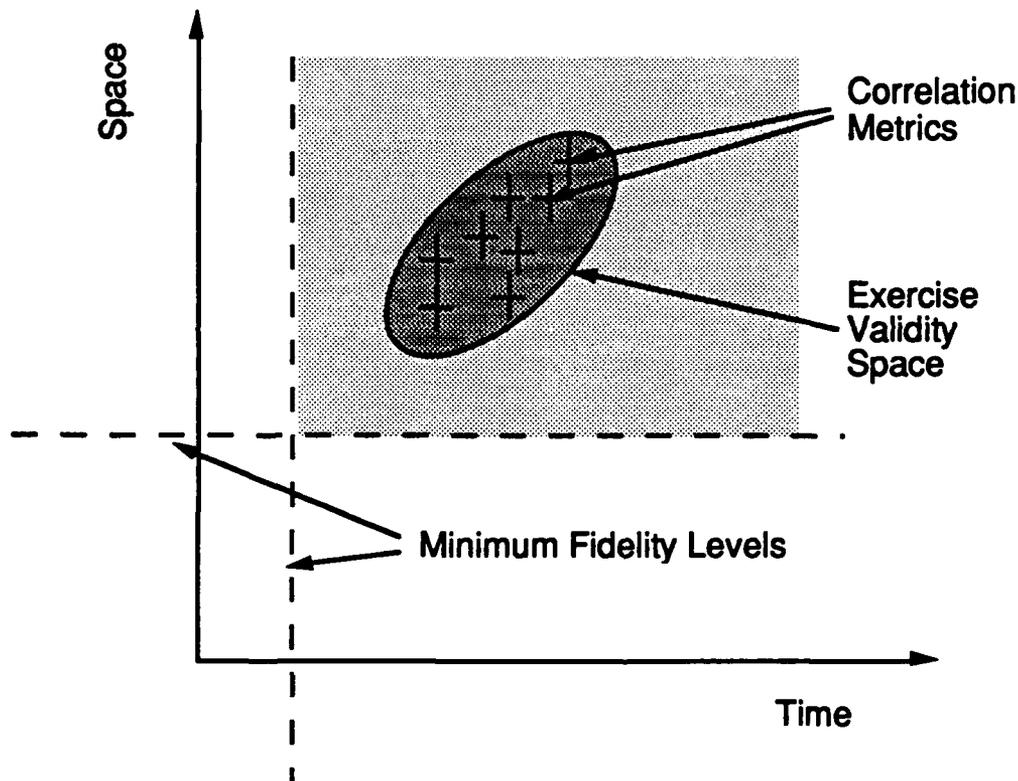


Figure 2-5: Correlation Space

Within this correlation plane we define a minimum level of fidelity on each axis that must be achieved for a given exercise (shown shaded in figure 2-5). The simulation entity correlation metrics are then plotted in the time/space plane, and tested to determine whether or not they fall within an "Exercise Validity Space", shown notionally in figure 2-5 as an ellipse. The notion is that the various entities that make up an exercise must lie relatively close together in the time/space plane for a valid exercise to take place. The location of the ellipse boundary will ultimately be determined by warfighter-in-the-loop experiments to test the correlation hypotheses for each application.

This correlation construct introduces the concepts of relative and absolute correlation. Relative correlation is defined as "closeness" in the time/space plane, the tendency of the metrics to cluster. Relative correlation says something about the similarity between two or more simulation entities, whereas absolute correlation measures the simulation entity against a fixed reference, such as an external source data base. In this sense absolute correlation describes the fidelity of a simulation entity. Clearly it is necessary to consider both aspects of correlation to determine interoperability for a given exercise.

Correlation Metrics

Interoperability between heterogeneous simulation systems requires that an adequate level of correlation be achieved for the intended application. Correlation in DIS is defined as time and space coherence of the simulation entities. It is our contention that correlation can be quantified by measuring the degree of coherence of the simulation Entity's Local Data Base and the Entity Processes.

Entity Local Data Bases are the private data bases associated with the simulation devices; they are derived from public, common source data bases. For example, terrain stored as a polygon mesh is typically proprietary to an IG vendor (the private data base) that derives from DMA or Project 2851 elevation grid data (the public data base). Entity Processes include the rendering algorithms, model dynamics and appearance functions, and in general all of the processing that is performed by a simulator from the retrieval of the local data base data to the output of the processed scene. Figure 2-6 is a simplified diagram of a DIS Entity, which is shown to consist of Entity Processing and a Local Data Base bounded by message and data base standards. Clearly both standards are necessary to permit a consistent and meaningful set of correlation metrics to be defined. The concept of correlation metrics is discussed further in Section 4.

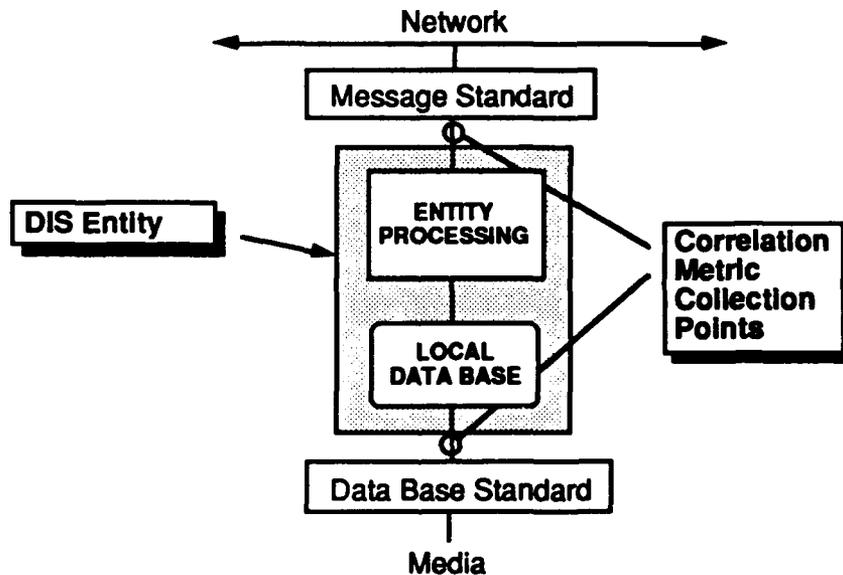


Figure 2-6: Correlation Metrics and DIS Entities

3. Temporal Correlation

3.1 Overview

The lag between when an event happens on the Virtual Battlefield and when it is presented to the warfighter by his simulator cueing systems can cause interoperability problems in DIS applications. Synchronization and timely delivery of messages are some of the most basic challenges in DIS technology. Because the system is distributed, each entity is highly dependent on accurate and reliable communication with external entities, so that it can maintain a current, coherent picture of the Virtual Battlefield.

This section addresses the symptoms, causes, and solutions of temporal disruption, and then discusses the means by which the new DIS architecture supports the implementation and coordination of the temporal coherence solutions.

3.2 Symptoms of Temporal Disruption

3.2.1 Tracking Error

Temporal disruption appears as *tracking error* when the images of external entities, presented to the warfighter by his simulator cueing system, are lagged in time and therefore differ in position from where the sending entities consider themselves to be.

3.2.2 Image Oscillation (Jumps And Jitter)

Image oscillation is a symptom of temporal disruption perceived as jumps and jitters. *Jumps* appear as entities blinking through space, from one position to the next, without seeming to cover the intervening space. A rapid sequence of jumps is perceived as *jitter*.

3.2.3 Out-Of-Sequence Events

Out-of-sequence events are occurrences on the Virtual Battlefield which defy logic by improper sequencing of cause-and-effect actions—like a missile destroying a target before it is even launched.

3.3 Latency—Cause of Temporal Disruption

The main source of temporal disruption in Distributed Simulation is latency—network communication latency, internal processing latency, and the combined end-to-end latency. Tracking error is an obvious symptom resulting from this latency. In this section, we will explore the multiple facets of latency, and come to understand how latency contributes to tracking error, as well as the other symptoms of temporal disruption.

3.3.1 Network Latency

(In order to focus on network issues, we will assume in this subsection that all other time latencies in the system are zero. The validity of this assumption—that network latency can be separately considered—will be evaluated when combined latency effects from multiple sources are examined.)

Network latency will always be with us, given the fact that information must flow over media (softwired or hardwired) between geographically-dispersed sites in a DIS networked implementation. This section will explore some of the dimensions of network latency.

	T	O	O-S-E
REA	✓	✓	
L	✓		✓
VL			✓
Q	✓	✓	
L + REA	✓	✓	
L + REA + Q	✓		

Legend:

T: tracking error

O: oscillation

O-S-E :
out-of-sequence events

L: network latency

REA: Remote Entity Approximation

VL: variable network latency

Q : quantization of time

✓ : sufficient cause

✓ : combined cause

Fig 3.1: Table Of Network Latency Effects

The Table in Fig 3.1 indicates the different aspects of network latency and the temporal disruption symptoms that are associated with them. A thick check marks indicate "sufficient" causes of the indicated effects. The effects would always be present in some degree due to the presence of the cause. The thin check

marks indicate combined causes that work in conjunction with each other to produce the indicated effects.

3.3.1.1 Different Frames of Reference

Primarily, network latency prohibits the existence of an absolute frame of reference of the DIS virtual battlefield.

An observer's (entity's) *frame of reference* is defined by the sequence of packets it receives. In this subsection, we will show that the packet sequence can be different for each observer, and thus *each observer has his own unique frame of reference on the virtual battlefield.*

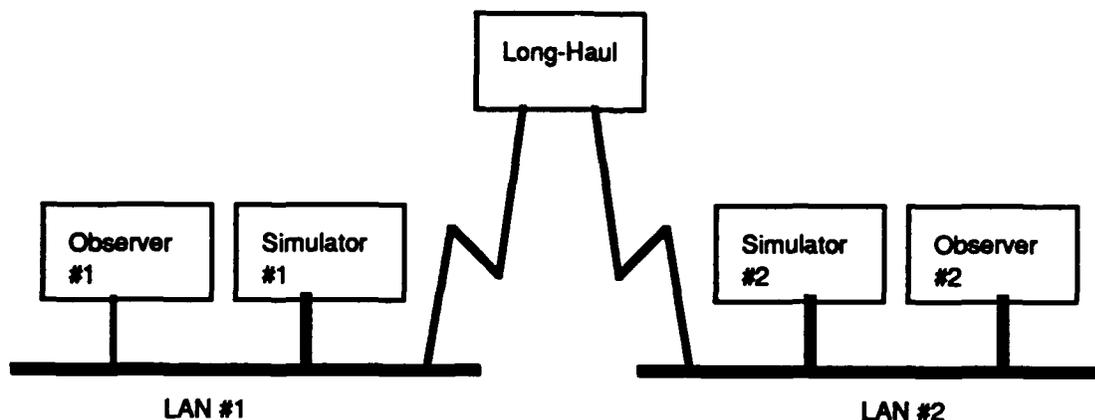
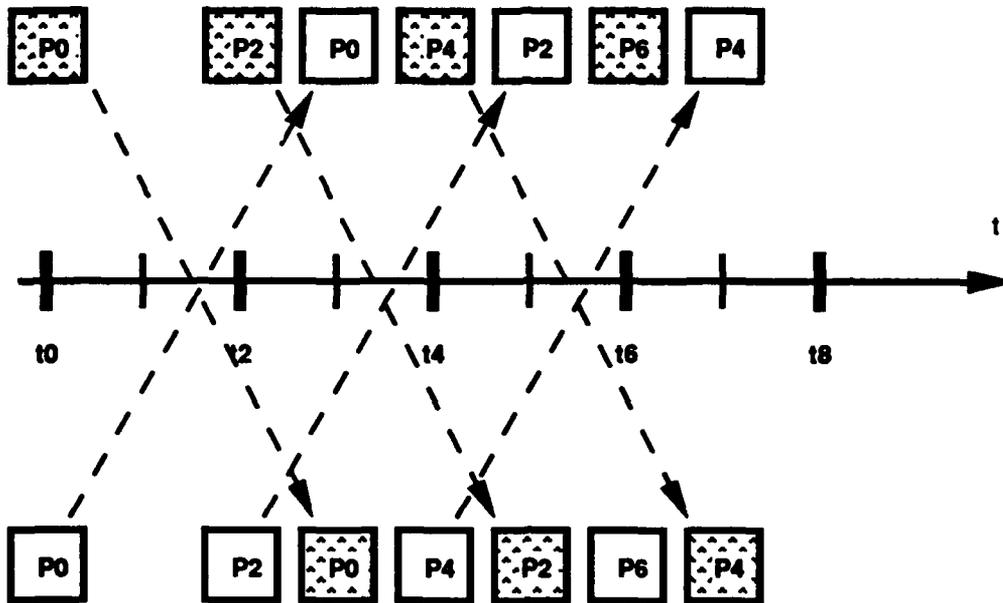


Fig 3.2: Two Lans Connected By Satellite Long-haul Connection

Figure 3.2 shows a DIS network composed of two broadcast LANS connected into a WAN. To heighten the effects of latency, the figure depicts a long-latency (≈ 100 ms. delay) satellite link to connect the two LANS.

Observer #1



Observer #2

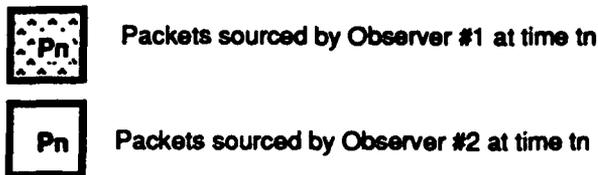


Fig 3.3: Divergent Frames of Reference

Figure 3.3 shows the data streams seen by observers on each of the LANs. The top sequence shows what observer #1 sees, and the bottom sequence shows what observer #2 sees. Note that network latency causes remote traffic to be dislocated in time with respect to local traffic. This causes the two network frames of reference to be different, causing observers on each of the LANs to observe different scenes.

3.3.1.2 Remote Entity Approximation (REA)

Network bandwidth is a scarce resource. The fact of this scarcity has been enshrined in DIS by the principle that a battlefield entity only communicates significant change of its state. In the intervening time, DIS entities must account for each other (in an approximate fashion) by using a DIS-defined extrapolation

technique (known here as Remote Entity Approximation, or REA) to project out a consistent time/space view of the kinematic state of all pertinent external entities.

Much more will be said about REA, but at this point it suffices to describe REA as a common approximation model which is shared by the host entity and all external entities which need to account for the host's kinematic state. The REA is characterized by a common extrapolation technique, and a host-entity-determined error threshold. The host entity maintains its "true" kinematic state on the Virtual Battlefield by means of a high-fidelity kinematic model. In addition, the entity maintains an approximated kinematic state by means of the common extrapolation technique. The host entity continuously compares these two states. When they differ by more than the allowable error tolerance, the host entity corrects its internal extrapolation state by updating with the true state, and continues subsequent extrapolation from this updated state. In addition, the host entity broadcasts its true state to all external entities, so that they in turn may update their approximated state, and henceforth extrapolate from the new true state

As the Table in Fig. 3.1 indicates, REA by itself can contribute to tracking error and oscillation, even without the presence of network latency. Figure 3.4 illustrates how this can occur.

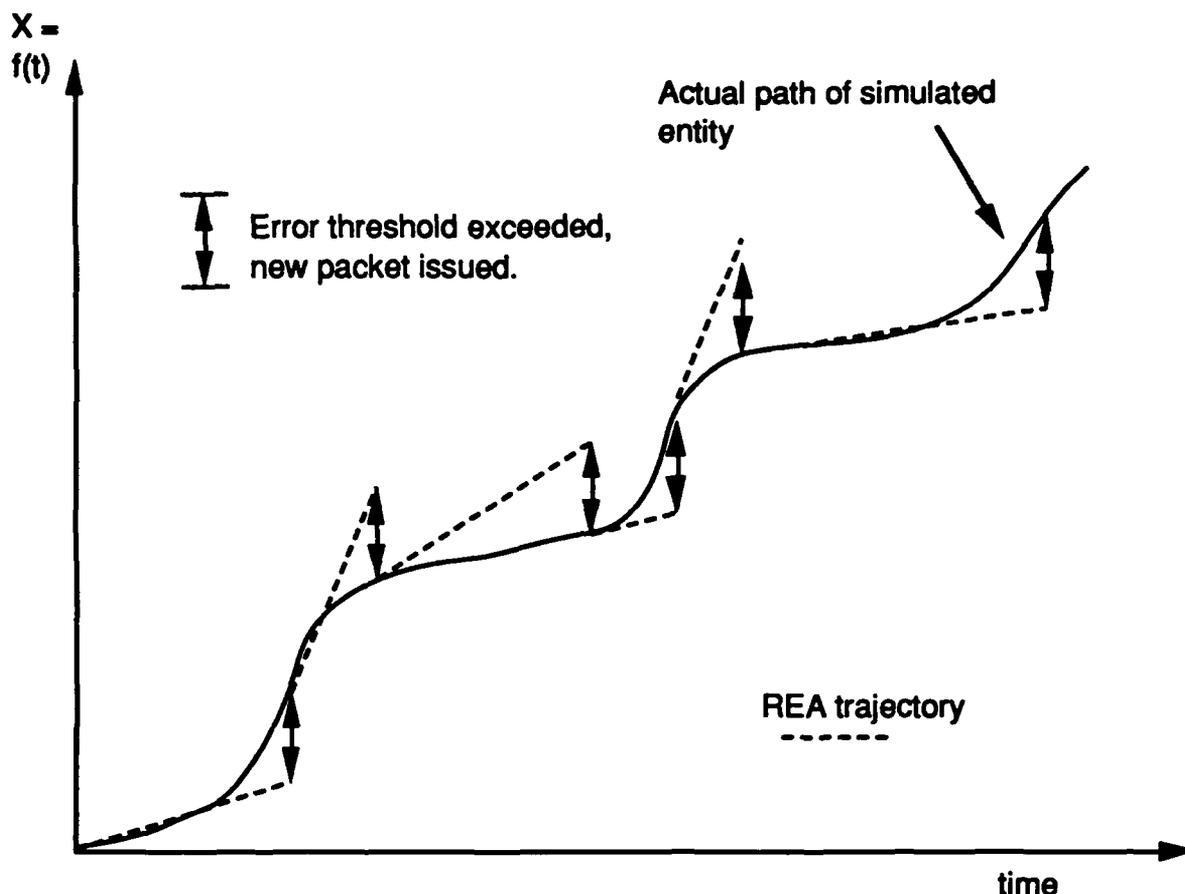


Figure 3.4. - REA Tracking Error and Oscillation. Each double arrow represents the time at which the REA position differs from the entity's actual position by a value greater than the threshold. A new packet is sent by the entity, and if the entity is instantly placed at its updated position, a visible position jump the size of the threshold will result. Also, between the times of the position updates, the remotes entity's position model will differ from the actual position model—representing tracking error.

3.3.1.3 Fixed-Time Latency

Fixed-time latency, by itself, can cause both tracking error and out-of-sequence events.

The occurrence of tracking error is due to the lagged information that the receiving entity receives from the sending entity. Kinematic state updates do not immediately reach their destination. Therefore receiving entities, by applying just-received information to the time of reception, instead of to the time of

transmission, must present a perspective that is lagged in time from the true entity position.

Out-of-sequence events can occur, even under conditions of fixed-time latency, when the path-traversal times associated with various entity-to-entity connections vary per connection. This kind of variation depends on the network topology, implementation, and the presence of switching devices. An example of this phenomenon is the case of a missile attack by entity "A", against entity "B", as viewed by entity "C". The missile entity is controlled by the node which supports entity "A". Missile PDUs are broadcast which describe the missile's flight path. At the terminus of the missile flight, "A" issues the detonation PDU which described the hit. Entity "B" broadcasts a stream of PDUs which describe its positional track. Assume these PDUs arrive at "C" such that the missile PDUs are significantly delayed compared to the PDUs from "B". This case may result in "C" perceiving the destruction of "B" (a "B"-issued PDU which describes damage to "B") before the missile even hits "B", and before the consequent detonation.

3.3.1.4 Variable Latency

Another dimension of network latency is its variableness.

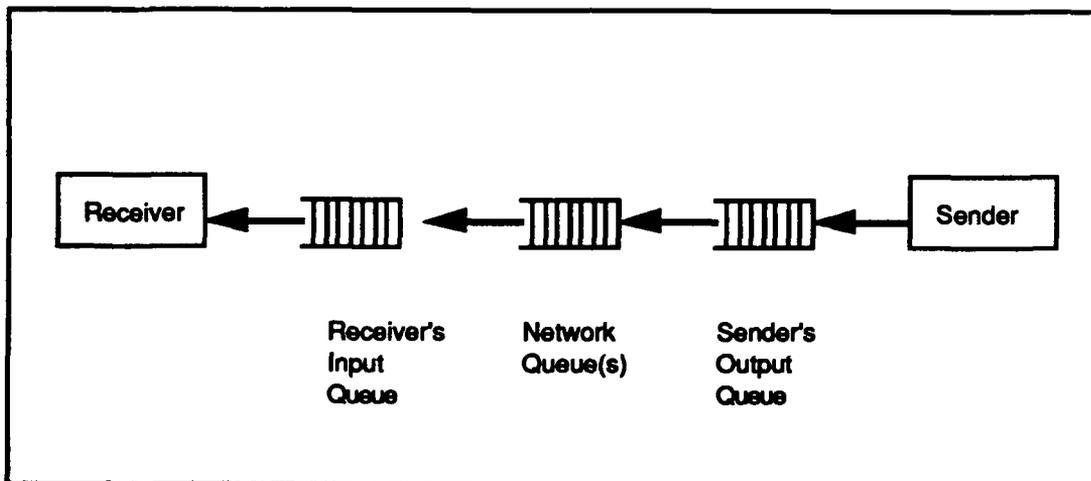
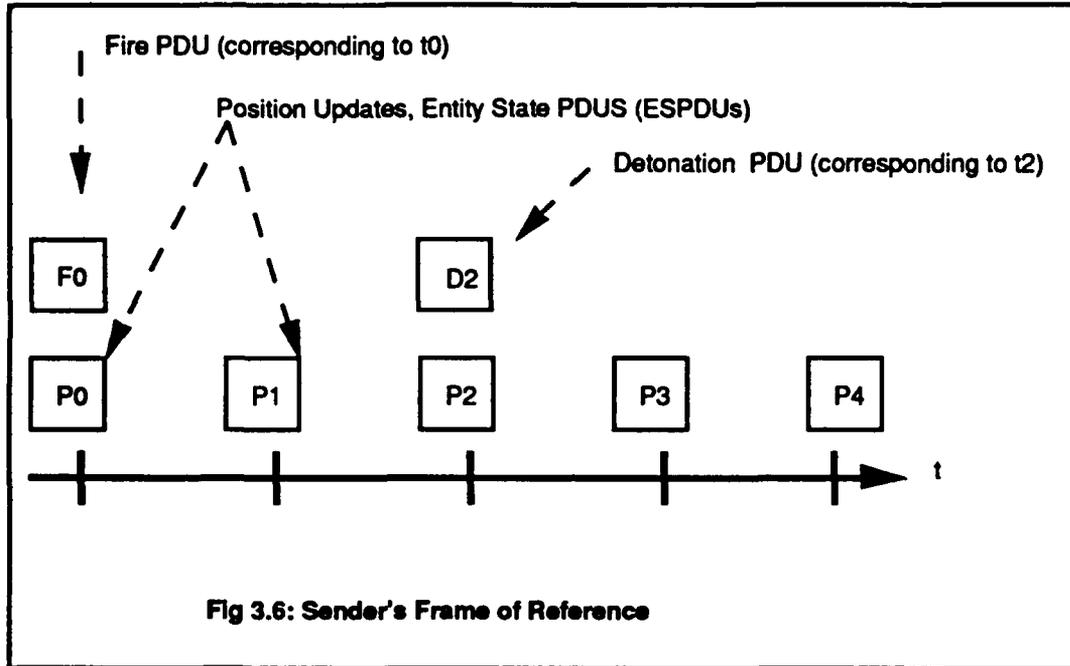


Fig 3.5: Variable Latency Due To Service Queues

Figure 3.5 illustrates one cause of this variableness. It shows a simple queueing model of the path from sender to receiver. The latency for this path traversal may vary because the service times for each of the queues may vary. This is especially true in the case of long-haul networks, packet-switched networks, and networks with "intelligent" gateways or "intermediaries."

Variable latency by itself may cause the perception of out-of-sequence events coming from a *single* entity. Individual PDUs emitted from a single entity may experience different transit delay as they traverse the network. The potential exists that the packets may arrive at a receiver out of order.

To better understand this phenomena of incorrectly-sequenced events, refer to Figure 3.6. The figure shows the stream of PDUs from one transmitting entity out to all receiving entities. This stream (when received) will define the receiver's frame of reference about the sending entity.



Note that the packet stream is composed of two different types of PDUs:

- Entity state PDUs (ESPDU), which define a kinematic state as a snapshot in time, and
- "Event" PDUs, such as the firing event and subsequent detonation event.

The crux of the problem is that:

- The correct sequence of arrival of ESPDU and event PDUs cannot be guaranteed,
- Event PDUs and ESPDU are related in time. (In the figure, the fire PDU "F0" occurs when the entity attains the firing position/orientation described in ESPDU "P0" at time t_0 . The consequent detonation occurs at time t_2 , when the entity attains the position/orientation described by ESPDU "P2".)

3.3.1.5 Fixed-Time Latency with REA

We now begin to investigate some of the combined causes.

Fig 3.7 illustrates how latency and dead reckoning contribute to jumps and tracking error that is substantially greater than what would be present under each of the conditions separately.

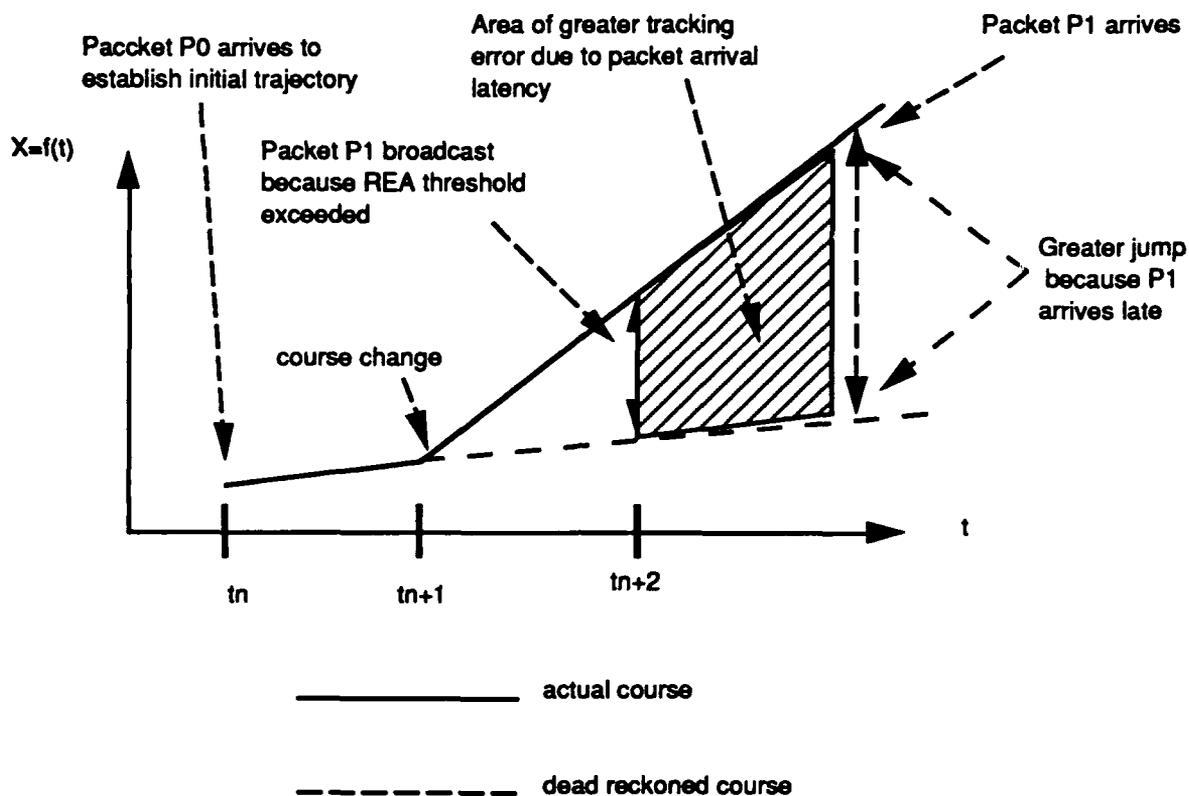


Fig 3.7: Latency And REA On X=F(T)

At time t_n , a kinematic state update (in the form of packet P0) establishes the REA trajectory shown by the solid line from t_n to t_{n+1} . At time t_{n+1} , a change in course by the simulated entity causes a new course update to be required. Because of network latency, the new course information, contained in packet P1, does not arrive immediately, but is instead delayed. When packet P1 does arrive, the old course has been extrapolated ahead. The sudden correction from the dead reckoned position back to the correct position is perceived as a jump—greater than what would have occurred under REA without delay. And the tracking error that occurs between the time of P1's transmission and its reception is also greater than what would have occurred under REA without delay.

3.3.1.6 Quantization Of Time

Real-time computer simulations mark time by discrete steps, or "frames". The "frame rate" (number of frames processed per second) of a simulation may be constant or may vary, depending on design and instantaneous computational load. The Distributed Simulation paradigm allows simulations with different frame rates to run unsynchronized in networked exercises. This means that

sending entities compute and transmit information according to an internal frame structure which may not be synchronized with that of receiving entities. Also, the information arrives at the receiving entity after a duration of network traversal that again may be unsynchronized with the frame structure of the receiver.

Taken as a whole, these characteristics (discrete marking of time, unsynchronized frame rates, and unsynchronized network traversal time) may cause temporal disruption, or may exacerbate disruption due to latency-related problems. To illustrate, we examine two cases for simulated vehicles in which there exists positional discrepancy between their host entity and remote entities. Both cases concern REA.

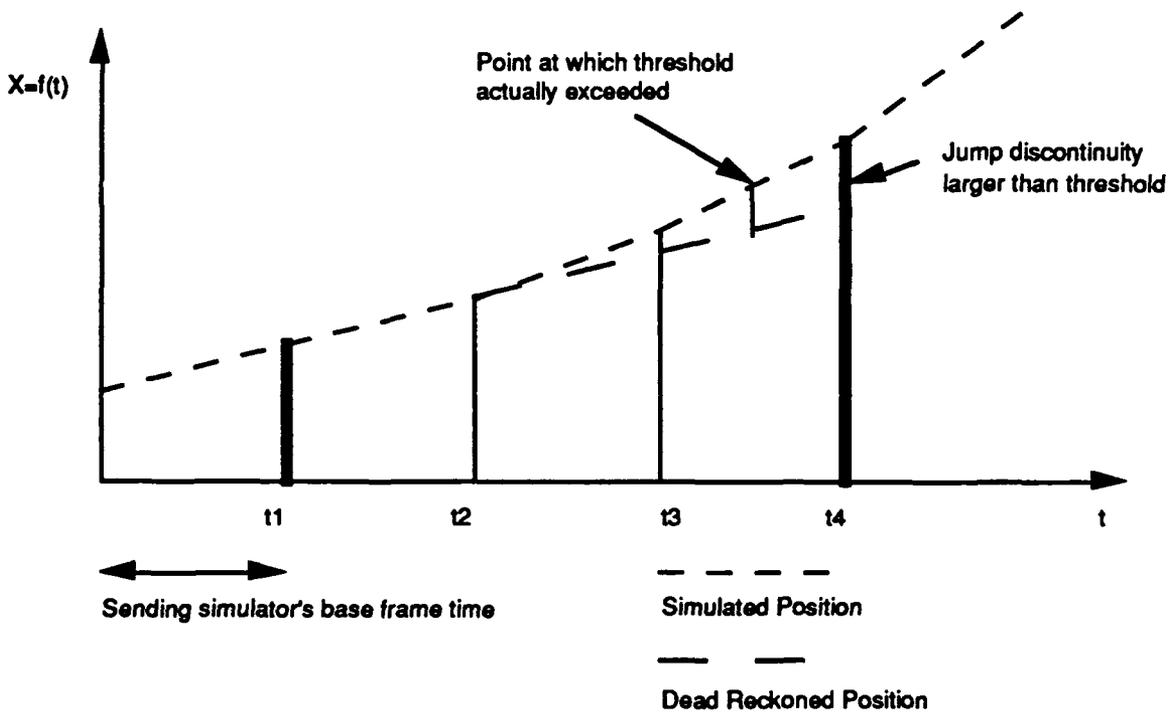


Fig 3.8: Exacerbated Position Error Due To Time Quantization

Figure 3.8 illustrates the greater error that can result when a continuous phenomenon like vehicle position is digitally simulated by marking time in discrete steps. The figure shows the position of the simulated entity as a function of time (x-position only to simplify the analysis). ESPDUs are issued at t_1 (to start) and at t_4 . The t_4 communication is necessary because the threshold has been exceeded. Note however that the threshold was actually exceeded earlier, sometime in the middle of the previous simulation frame. The resultant jump discontinuity between dead-reckoned position and actual position exceeds the threshold by a substantial amount because the initial threshold departure occurred in the middle of the frame and was consequently not processed immediately upon occurrence.

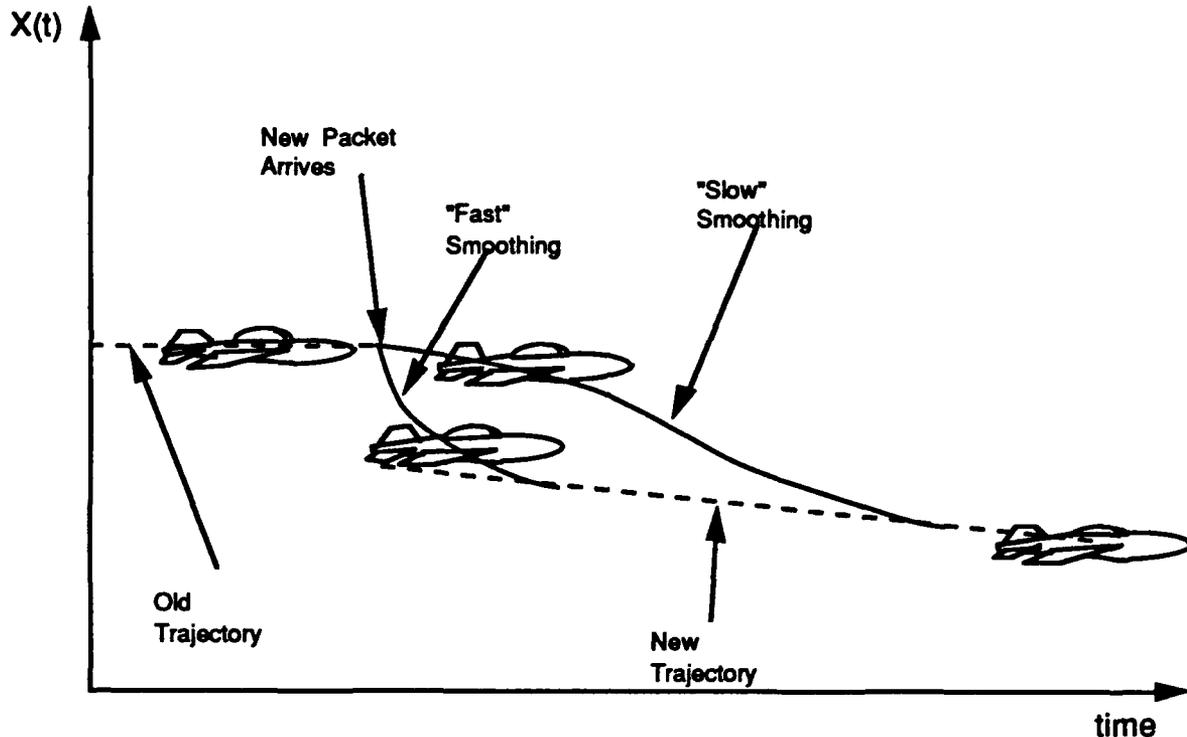


Figure 3.10 - Algorithmic Differences Cause Time/Space Inconsistencies. The above picture shows how two different simulation nodes can view the same plane in two different places because of different smoothing algorithms. If one node uses a "fast" smoothing algorithm, and the other uses a "slow" algorithm, the position of the plane will be different for a short time after a packet is received.

The Entity Appearance packet in the DIS protocol provides a data field for describing the dead reckoning algorithm being used by the transmitting entity. Standardization of other algorithms such as smoothing, forward reckoning, etc., can be accomplished by use of a similar mechanism, or, algorithm specification can become part of the session database. In either case, it is important to minimize the inconsistencies caused by mismatched algorithms to the greatest extent possible.

3.3.2 Internal Processing Latency

In this section, we will examine the visual anomalies that arise due to latencies within a given simulation host.

At left, $a(t_0)$ is received. It takes $L1-3$ time to arrive where ballistics are performed. It takes Lb time to compute ballistics against $a(t_0)$. After a hit is determined, it takes $L3-1$ time to get the detonation onto the network.

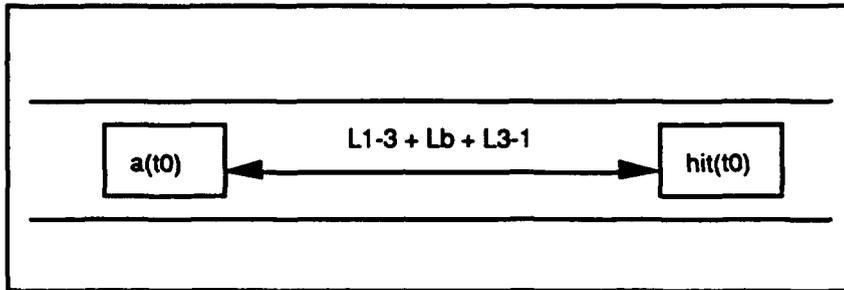


Figure 3-13: Network Frame of Reference

The network's frame of reference is pictured in figure 3-13. Note how the entity appearance packet $a(t_0)$ and the hit packet $hit(t_0)$ are displaced in time $(L1-3) + Lb + (L3-1)$, independent of and unrelated to any network latencies.

Using $(L1-3) = Lb = (L3-1) = 67ms$, the hit and the entity appearance are displaced in time approximately 200ms, For a fast-moving A-10, 200ms = 1/5sec, at 400m/sec = 80m dislocation.

3.3.3 End-to-end Latency

In this section, we will use our direct fire example again, this time with a network between shooter and victim. We will superimpose the external network latency effects on top of the internal processing latency effects.

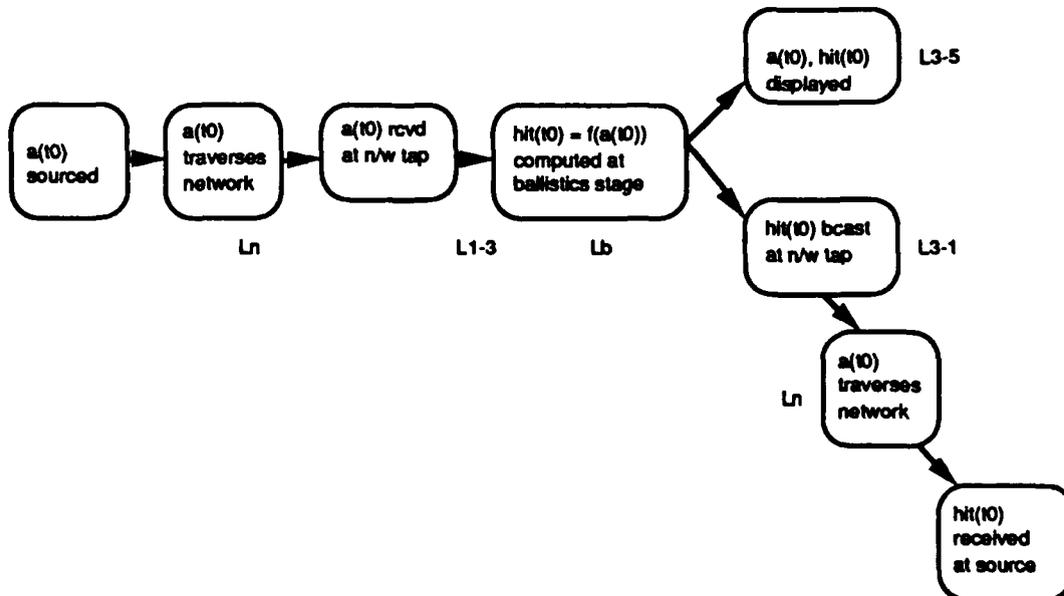


Figure 3-14: Causality Diagram with Network Latencies

The causality diagram is updated to include network latencies; see figure 3-14. The network frame of reference on the victim's net thus looks like figure 3-15.

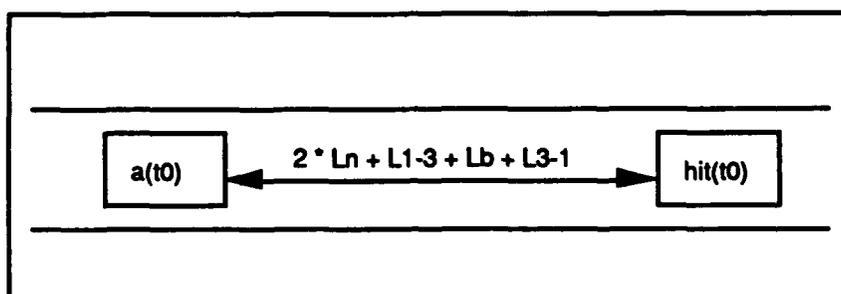


Figure 3-15: Network Frame of Reference on Victim's Net

For a fast LAN ($L_n=2\text{ms}$), total displacement = 204ms, and the extra 4ms causes only 1.6 m additional dislocation. However, with a long-latency satellite delay ($L_n=250\text{ms}$), total displacement = 700 ms. For a fast moving A-10, the additional 500ms latency causes 200m additional dislocation.

3.4 Correlation solutions

In this section, we will examine several algorithms that purport to solve some of the problems we have examined above. Each has advantages and disadvantages, in some cases introducing different, and perhaps less objectionable anomalies.

3.4.1 Timestamping

The DIS Protocol specifies the use of a timestamp field in most DIS packets. There are two types of timestamp values, relative and absolute. Both timestamps indicate the time that the data in the packet is valid.

Absolute timestamps are used when all simulators are synchronized. This can be accomplished through a variety of mechanisms such as common radio reception of a synchronizing signal by simulation nodes, hardwiring a timing signal between nodes (only useful in Local Area Networks), etc. Absolute timestamps in DIS are represented by Universal Time Coordinates (UTC), with an accuracy of 32 bits. By placing an absolute timestamp in a network packet, receiving nodes can determine the time at which the data is valid relative to the time at which the effects will be painted. This technique is described in detail in [Katz, 1992]. There are limitations on the use of absolute timestamps to correct for network and processing latencies. These are discussed in detail in section 3.4.5, Forward Reckoning Algorithms.

Relative timestamps are used when the simulation nodes are unsynchronized, and indicates the time, relative to a particular node's internal clock, that data is valid. It can be used to determine the time correction necessary to account for unsynchronized frames. By relating a remote node's relative timestamp to a local

entity updates are translated to the preferred CIG representation R_c , and are then displayed.

There are two bottlenecks. One is the network bandwidth, and the other is the local computation power available to perform the processing. Next, we quantify those bottlenecks.

B= Network Bandwidth

M= available MIPS/MFLOPS

N= number entities

P= packets per second. For our example, equal to the basic frame rate.

R_n =representation of entity state update on network (XDR)

R_c =representation of entity state information necessary to display entity on CIG

S= size of each entity state update = $f_0(R_n)$

C_n = Cost of entity state update service time = $f_1(R_n)$

C_c = Cost of conversion from entity state format to CIG format = $f_2(R_n, R_c)$

Thus, bandwidth on the network is:

$$B = NPS = NPf_0(R_n)$$

Here, we can see that bandwidth on the network is a linear function of the size of R_n . Thus it is an overriding desire to make R_n as compact as possible.

The computation cost without the filtering optimization is:

$$M = NPC_n + NPC_c$$

If we have the network interface filter out "uninteresting" entities, we reduce the number of CIG format conversions, but arguably we increase the per-packet processing slightly.¹

¹ CIGs typically have a maximum number of moving models and effects they can display. Processing updates from any more entities is thus pointless. Those systems without CIGs, such as SAFOR, do not have the luxury of such an optimization.

3.4.2.2 DRAs Trade-off Bandwidth vs. MIPS

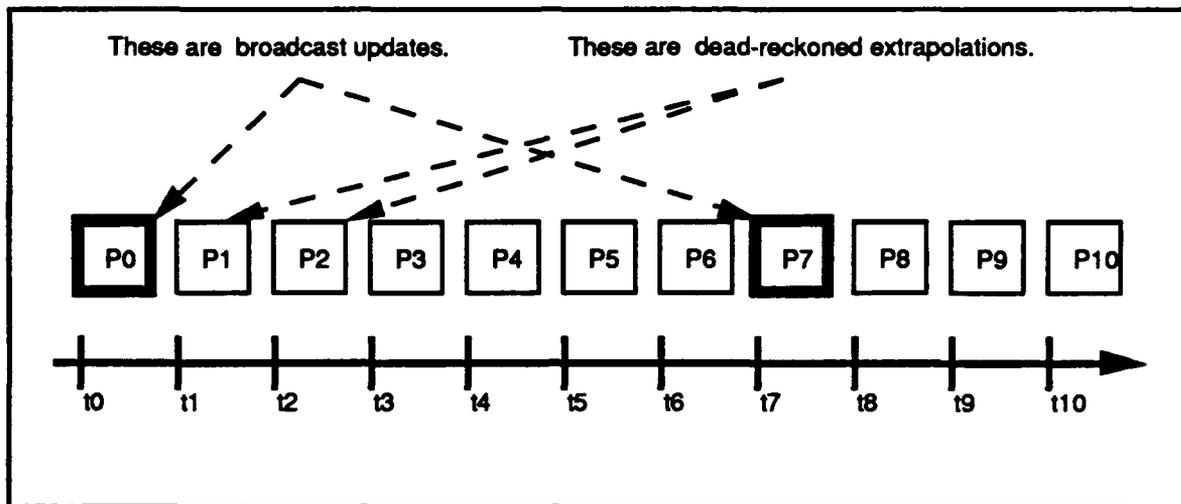


Figure 3-18: Dead Reckoning

3.4.2.2.1. How DRAs Work

In order to employ DRAs to save network bandwidth, the observed entity and its observer agree upon:

- New state variables and their XDRs. These are time-derivatives of the previous state variables (e.g., $\frac{dX}{dt}$)
- A set of state equations (e.g., $X' = X + \frac{dX}{dt} \cdot \Delta t$)
- A set of "thresholds" for selected state variables. (e.g., 1 meter)

The observed entity:

1. Uses its internal kinematics and dynamics models to compute the "actual" values of the state variables.
2. Runs the simple state equations to compute the "lower fidelity" values of the state variables.
3. Computes the discrepancy between the "actual" state variable values and the "lower fidelity" state variable values. The observed entity guarantees that it will send out an entity state update only if

the discrepancy exceeds the agreed-upon threshold.³ (P0 and P7 in Figure (above).)

The observers make a decision each frame:

1. If an update is received, use its state variable values.
2. If no update is received, run the simple state equations and use the resultant state variable values. (P1-P6, and P8, etc. in the Figure are such extrapolations.)

In SIMNET, this cut the number of updates from 15Hz, to 1Hz to 3Hz.

3.4.2.2.2. DRAs in a Manned Simulator

Figure 3-19 is a version of our previous data flow diagram, updated to include Dead Reckoning Algorithms. Briefly, entity state packets update DRA data structures. In the absence of network updates, the DRAs update the data structures each frame. Also each frame, the CIG interface reads the DRA data structures to provide the CIG with display information.

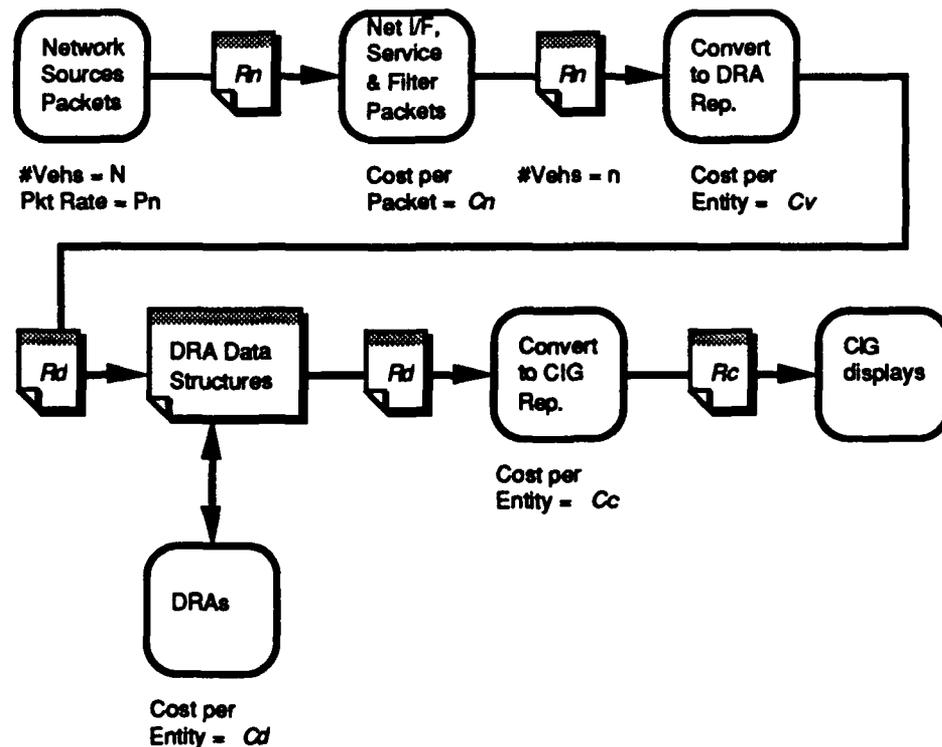


Figure 3-19: DRA Data Flow Diagram

³ Entity state updates are also sent out periodically for several other reasons. 1, so that late-arriving observers learn of the sending entity. 2, so that errors due to missed or dropped updates are eventually corrected. Hence, the "self-healing" nature of the protocol.

Next, we quantify the effects of DRAs:

R_d =representation of entity state information in DRA data structures

C_v = Cost of conversion from entity state format to DRA format = $f_3(R_n, R_d)$

C_c = Cost of conversion from DRA format to CIG format = $f_4(R_d, R_c)$

C_d = Cost of DRA application = $f_5(R_d)^4$

P_n = entity state update rate per entity

Thus,

$$B = NP_n S' = NP_n f_0(R_n')$$

Because R_n' now contains time-derivative state variable parameters, S' is now larger, by 32 bits for each of:

- first and possibly second order position updates
- first order rotation roll, pitch, and yaw
- first order rotation and position rates for, say, 2 articulated parts

$13 * 32 \text{ bits} = 416 \text{ bits}$. Thus, although S' has doubled in size, P_n has dropped by a factor of at least 6. Thus, total bandwidth is cut by at least one-third. What is the computation cost of such a bandwidth-savings?⁵

$$M = NP_n C_n + NP_n C_v + NPC_d + NPC_c$$

Making the same optimization we did before:

$$M = NP_n C_n' + nP_n C_v + nPC_d + nPC_c$$

$$M = NP_n f_1(R_n') + nP_n f_3(R_n, R_d) + nP f_5(R_d) + nP f_4(R_d, R_c)$$

⁴ Includes smoothing, etc.

⁵ We ignore the cost of DRA computation by the sender because it does not grow with either N or n except in CAUs or CIUs. It is extremely important not to concentrate on $O(1)$ processing instead of $O(N)$ and $O(n)$ processing.

3.4.2.4.2. State Equations

The DRA state equations and thresholds are important in that they indirectly determine P_n , the frequency at which entities will broadcast updates, using up network bandwidth and simulator compute power. Examples of DRA state equations can be found in a number of DIS position papers.

3.4.2.4.3. Required Data Types

We require a network data type to represent position and its time-derivatives. There is not much debate over using (X, Y, Z) 3-tuples or vectors.

For 2D orientations and rates, options include:

- Direction Cosine Matrices (DCMs). Ideal for 3D rotations of vectors, overkill for 2D, and very large (9 floating point numbers).
- Sin/Cos pairs. Can do transformation in a plane with 2 multiplies. Requires 2 floating-point numbers.
- Radian angular measures. Compact (1 floating-point number), but requires 2 transcendental functions (or table lookup and interpolations) to get to Sin/Cos pair.
- Binary Angle Measures (BAMs). Typically fixed-point fractions of a circle. Requires 1 fixed-point number. Advantages include conciseness and maximal precision per bit. Same cost as radians (except arguably faster table lookup), and any extra precision is lost in the translation.

For 3D orientations and rates, options include:

- Direction Cosine Matrices (DCMs). Ideal, but large.
- Quaternions. 4-tuple defining a vector and a rotation about it. Compact 4 floating-point number representation.
- True Euler Angles. Compact 3-tuple.
- Tait-Bryan Angles. Compact 3-tuple, roll, pitch, and yaw.

3.4.2.4.4. Required Operations

- add/subtract multiply/divide
- rotate vector / transform coordinate
- scale by constant

3.4.2.5 Evaluating Different Data Type Representations

3.4.2.5.1. Candidate Representations

- position: (coordinates systems: Z up vs. Z down)
- dp/dt d2p/dt2 (world vs body coordinates)
- 2D: BAMs, radians, sin/cos
- 3D: Tait-Bryan, True Euler, Quaternions, Direction Cosine Matrices

There has been vigorous discussion of these decisions:

- Quaternions (Burchfiel, Saunders)
- Fixed vs Floating (Smith)
- BAMs

3.4.2.5.2. Cost of Operations

In order to determine the optimal representation at each point in the system, we need to quantify the cost functions. Each row of tables 3-1, 3-2, and 3-3, when they are complete, can be plugged into the bandwidth and computation-cost equations above. The minimal bandwidth and minimal compute-cost representations are the optimal choices.

The tables assume that Direction Cosine Matrices and World Coordinates are the preferred input formats to most CIGs.⁸

⁸ Even those CIGs that accept other representations must convert them to DCMs in order to transform points from world coordinates to screen coordinates. *IS THIS TRUE???*

Table 3-3: Cost Contribution of Various Position Rate Representations

Rn	Rd	f3	f4	f5
World Coords	World Coords	0		
World Coords	Body Coords			
Body Coords	World Coords			
Body Coords	Body Coords	0		

3.4.2.6 Dead Reckoning Fundamental Relationships

We now examine some fundamental relationships or rules of thumb that have use in evaluating Dead Reckoning requirements. For illustrative purposes, we will restrict the following analysis to the case of the single x -coordinate only.

3.4.2.6.1 Dead Reckoning Types

We begin by reviewing the mechanics of the two DR types: first-order DR and second-order DR.

As background, we describe the details of positional update in the entity vehicle math models. Typically, the entity positional math model computes, on a frame-by-frame basis, the x -forces acting on the vehicle. By means of Newton's Second Law ($f=ma$), x -acceleration for simulation frame n , a_n , is computed based on these forces. x -velocity v_n and x -position x_n for frame n are then computed by numerical (digital) integration over the time-step interval t_s . The equations are

$$v_n = v_{n-1} + (t_s) a_n$$

$$x_n = x_{n-1} + (t_s) v_n$$

where $t_s = \frac{1}{t_n - t_{n-1}}$ is the (assumed to be constant, but not necessarily so) time-step interval. This system of equations simplifies to

$$x_n = x_{n-1} + (t_s) [v_{n-1} + (t_s) a_n]$$

First-order DR is merely replication of this numerical integration process, with the the simplifying assumption that x -acceleration is zero. This process

becomes mere linear extrapolation of the last-received positional information according to the last-received velocity information. At time t_n , the correct x -position x_n and x -velocity v_n are received. Until the next correcting update arrives, the DR model position \hat{x}_{n+m} for time t_{n+m} is computed as

$$\hat{x}_{n+m} = x_n + m t_s v_n .$$

Second-order DR is linear extrapolation of the received velocity v_n according to the received acceleration a_n . The relevant equations at time t_{n+m} are

$$\hat{v}_{n+m} = v_n + (m t_s) v_n$$

$$\hat{x}_{n+m} = x_n + (m t_s) \hat{v}_n + \left(\frac{m^2 + m}{2}\right) (t_s)^2 a_n$$

where \hat{v}_{n+m} is the DR velocity.

3.4.2.6.2. Required Update Frequency

DIS users and developers must specify the Battlefield Database parameter values that will shape the fidelity of the intended exercise. Dead Reckoning error thresholds (the position update criteria) stand as key parameters of interest in this specification process. While it is tempting to set arbitrarily tight error thresholds to amply ensure entity positional accuracy throughout the conduct of the exercise, developers must guard against the side effects of greater network traffic which will necessarily occur because of greater amount of entity state PDUs required to support the stricter error thresholds.

Let I represent the time interval between two successive required updates under dead reckoning. The cognizant user/developer, having specified the DR error threshold T , and having some knowledge of the maximum acceleration capability of the combat platform, should be able to derive an estimate for the minimum (worst-case) possible value of I , and thereby estimate the maximum possible (worst-case) transient update frequency F required by the DR update model.

We now derive the worst-case estimate for I and F .

For the first-order DR case, assume time t_n represents the time of the last positional update. A new positional update will be required at time t_{n+m} for which $|\hat{x}_{n+m} - x_{n+m}| > T$ (for smallest possible integral value of m). The shortest time interval between two consecutive updates must occur when the absolute value of acceleration is at a sustained maximum, a_H . For acceleration at this sustained extreme value, the x -position at time t_{n+m} is equal to

$$x_{n+m} = x_n + m t_s v_n + \left(\frac{m^2 + m}{2}\right) (t_s)^2 a_H .$$

Therefore, the positional error at time t_{n+m} is equal to $| \hat{x}_{n+m} - x_{n+m} |$ and can be computed by

$$\begin{aligned} | \hat{x}_{n+m} - x_{n+m} | &= | x_n + mt_s v_n + (\frac{m^2+m}{2})(t_s)^2 a_H - x_n - mt_s v_n | \\ &= | (\frac{m^2+m}{2})(t_s)^2 a_H |. \end{aligned}$$

Therefore, an update is required whenever

$$| \frac{m(m+1)}{2}(t_s)^2 a_H | > T.$$

When the above equation is solved for m , the resulting answer becomes

$$m > \frac{1}{2} \sqrt{1 + \frac{8T}{(t_s)^2 a_H}} - \frac{1}{2}$$

for smallest integral value of m . When m is found, then the worst case (transient) update frequency F that can result is $F = \frac{1}{m}$.

For the second-order case, the analysis becomes a bit more difficult. For this case, the DR model position at time t_{n+m} (for the most recent update occurring at time t_n) is equal to

$$\hat{x}_{n+m} = x_n + mt_s v_n + (\frac{m^2+m}{2})(t_s)^2 a_n.$$

Therefore the positional error at time t_{n+m} will be equal to

$$| \hat{x}_{n+m} - x_{n+m} | = | (\frac{m^2+m}{2})(t_s)^2 (a_H - a_n) |.$$

Proceeding as in the first-order case, the minimum integral value of m that will cause an update to be required will be

$$m > \frac{1}{2} \sqrt{1 + \frac{8T}{(t_s)^2 (a_H - a_n)}} - \frac{1}{2}$$

3.4.2.6.3. Worst Case Positional Error Under Latency

We now bring latency into the analysis. We ask the question: what is the worst-case positional error that can occur for dead reckoned positions when faced with latency? Again, we assume that this worst case error will occur when the subject vehicle is undergoing maximum acceleration a_H . We will again examine both types of dead reckoning.

As we saw above, under Type 1 dead reckoning, the maximum amount that the DR model position can depart from the actual position (under digital simulation) in m simulation frames is equal to

$$| \frac{m^2+m}{2} (t_s)^2 a_H |.$$

The maximum amount of departure under Type 2 DR (given the same assumptions) will be

$$| \frac{m^2+m}{2} (t_s)^2 (a_H - a_n) |.$$

Under digital simulation, latency must be measured in integral frames. Thus, m frames of latency will produce a positional displacement error of magnitude

$$m | \frac{m^2+m}{2} (t_s)^2 a_H |$$

and

$$m | \frac{m^2+m}{2} (t_s)^2 (a_H - a_n) |$$

respectively under Type 1 and Type 2 DR.

3.4.3 Smoothing

As previously discussed, distributed simulation network traffic can be reduced through the use of dead reckoning. By providing time derivatives of position (velocity, acceleration, angular velocity, etc.) in the appearance packet broadcast by an entity onto the network, remote entities can extrapolate the broadcasting entity's kinematic state into the future. When the error between an entity's internally simulated position, and the perceived position as extrapolated by remote entities, exceeds a predetermined threshold, the entity broadcasts an update with new position and time derivative information. When remote entities receive this update they have to somehow correlate their current dead reckoned position of the broadcasting entity, and the updated information just received over the network. The simplest approach is to place the entity at its new location and continue to dead reckon from there. Though this method requires no computational power, it leads to a disconcerting jump in the position of the remote entity. An observer will perceive the entity instantly blinking from one position to another. This effect jeopardizes the believability of the simulation, distracting the user from his tasks. It also may produce negative training. Figure 3-20 shows how a "jump discontinuity" is produced.

It is clearly desirable to eliminate these "jump discontinuities" to insure that the effectiveness and believability of the simulation is not compromised.

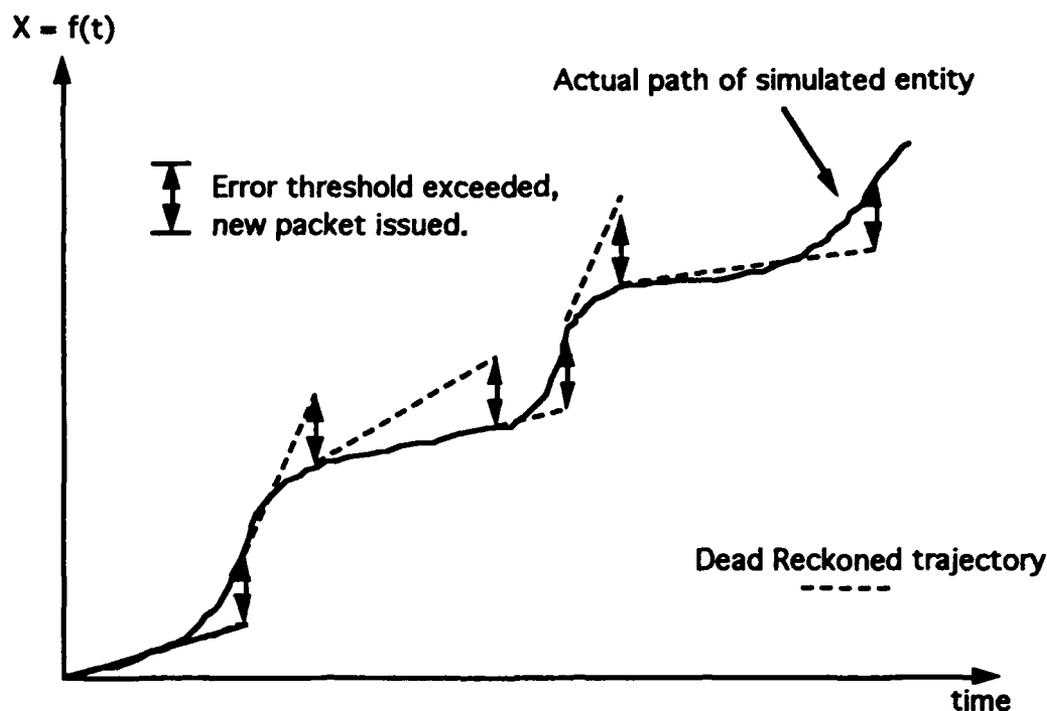


Figure 3-20. - Jump discontinuities resulting from exceeding the dead reckoning threshold. Each double arrow represents the time at which the dead reckoned position differs from the entity's actual position by a value greater than the threshold. A new packet is sent by the entity, and if the entity is instantly placed at its updated position, a visible position jump the size of the threshold will result. This assumes negligible network delay.

3.4.3.1 Prior Attempts at Smoothing

An early attempt to provide a so called "smoothing" function in distributed simulation was embarked upon in the SIMNET Stealth Vehicle. The Stealth has the unique capability to "tether" on to another entity, providing the Stealth operator an easy way to observe the battlefield with respect to the tethered entity. Since the position and orientation of the Stealth is dependent on the position and orientation of the tethered entity, the effects of jump discontinuities were greatly exaggerated. When a tethered entity jumped in position, instead of just observing a disconcerting position change in the remote entity, the entire Stealth platform underwent the same jump. The Stealth operator saw his whole world lurch in an extremely disorienting way. For this reason it became necessary to "smooth" out the effects of these position jumps between network updates.

The first smoothing algorithm developed for the Stealth Vehicle is graphically depicted in Figure 3-21. Upon receipt of a network update from a remote entity, the receiving entity selects a time in the future, t_{sm} , at which the remote entity position will coincide with the kinematic state dead reckoned from the newly received appearance packet. This time period is the time over which the

smoothing algorithm operates, hence is called the smooth time. By extrapolating the newly received kinematic information over the smooth time, the position in space at the end of the smooth time, which would have resulted if the entity was dead reckoned with the new dead reckoning parameters, is computed. For SIMNET first order dead reckoning, this intercept position, in one dimension, would be:

$$X_{int} = X_{new} + V_{new} * t_{sm} \quad (1)$$

Where:

X_{int} = Intercept position at desired time t_{sm} .

X_{new} = Updated position of remote entity from packet.

V_{new} = Updated velocity of remote entity from packet.

Using the last dead reckoned position of the remote entity, and knowing the target position in space, X_{int} , at which the two kinematic states will coincide, and knowing the time, t_{sm} , over which that distance has to be traversed, a smoothing velocity is computed:

$$V_{sm} = \frac{X_{int} - X_{old}}{t_{sm}} \quad (2)$$

Where:

V_{sm} = Velocity during smoothing.

X_{old} = Last dead reckoned position of remote entity.

This velocity, V_{sm} , is used to dead reckon (1st order) the entity over the smooth time, from the position X_{old} , to the position X_{int} , where the entity intercepts the trajectory from the packet. Upon interception, normal dead reckoning using the parameters from the packet takes over from the smoothing algorithm. This smoothing algorithm is used on orientation as well as position.

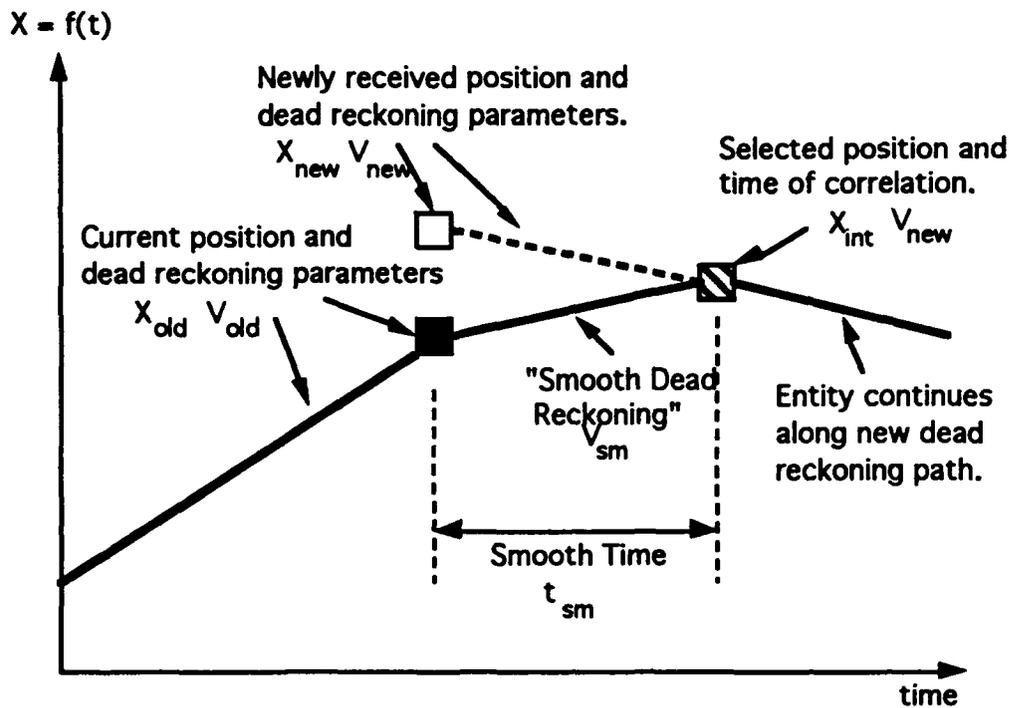


Figure 3-21 - First "smoothing" algorithm in SIMNET . A smooth time, t_{sm} , is selected, and the newly received position and dead reckoning parameters are extrapolated into the future to the "intercept" position, X_{int} . "Smooth" dead reckoning velocity is computed and substituted in the entity's dead reckoning equations. When the entity "intercepts" its new course, the new dead reckoning parameters take over from the smoothing parameters.

In the first iteration of this algorithm, a constant smooth time, t_{sm} , of seven frames, or about half a second was selected. Upon analysis, the constant smooth time appeared to cause two undesirable side effects. Firstly, if appearance packets are typically received before the smoothing is finished, a steady state position error will accumulate. Though the entity will look smooth, it could build up a positional error much greater than the discrepancy threshold. Secondly, if new appearance packets are not required until long after the smoothing is finished, such as in straight line, non-accelerating ground traversal, or straight-and-level flight, the termination of the smoothing algorithm can be visibly discerned because of its relatively large deviation from the otherwise uniform trajectory of the entity. To solve these problems the smooth time was made variable, and was set equal to the time delay between the last two received appearance packets. It was found in SIMNET that the difference in packet rate with respect to time was fairly small, meaning that one could approximately predict when the next packet would arrive based upon the delay between the two most recently received packets. This improvement solved the aforementioned problems and is currently the algorithm in use in SIMNET today. Smoothing of turrets on tanks was also implemented using this same basic algorithm.

Though the SIMNET smoothing algorithm works very well in most cases, it does still produce some visible anomalies. Figure 3-22 shows the smoothing trajectory from Figure 3-21, as well as the time derivatives of that trajectory. The figure shows that the velocity of the entity changes as a step function while the acceleration consists of two impulses. An impulse in acceleration implies that an infinite force was applied to the entity for an infinitesimally small time, the product of the two being the magnitude of the impulse. Infinite forces don't occur very often in nature. When observing the SIMNET smoothing algorithm in action, the instantaneous step change in velocity is actually noticeable, and distracting on occasion.

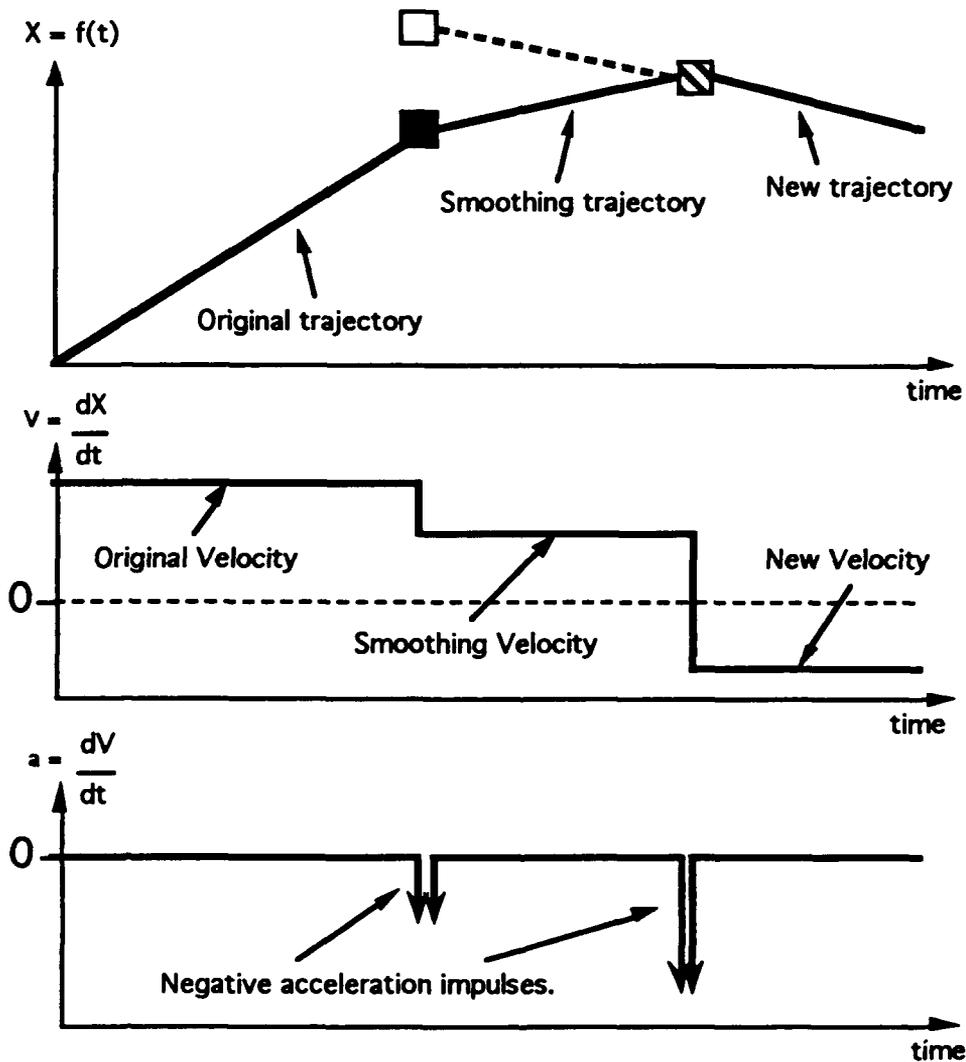


Figure 3-22. - Time Derivatives of SIMNET Smoothing Algorithm. The top graph shows the change in position due to smoothing (from Figure 3-21). The second graph shows the first derivative of position (velocity), and the third graph shows acceleration. The acceleration profile indicates that two impulses are generated by the smoothing algorithm. This is an unrealistic dynamical event for entities and can be visually disconcerting.

3.4.3.2 Optimized Smoothing Algorithms

In the quest for an "optimal" smoothing algorithm, there are three main requirements which may be traded off against each other:

- The smoothing should look realistic, should not be visually noticeable, and should not distract the user.
- The position error between the smoothed entity and the straight dead reckoned entity should be minimized.
- The computational load of the smoothing algorithm should be minimized.

Section 5.0 discusses in detail the issue of overload management as it relates to computational burdens in DIS simulators. Clearly, if an entity is too far away to benefit from smoothing, or dead reckoning for that matter, the simulator should switch to a less computationally intensive algorithm. For the purpose of this discussion we will assume that an overload management policy will degrade our "optimal" smoothing algorithm to another algorithm when convenient. The first two requirements, visual realism and minimized position error, then become our primary concerns.

3.4.3.2.1. Visual Realism in Smoothing

Visual realism in smoothing is achieved when a dead reckoned entity switches from one kinematic state to an updated one in a natural enough way such that the user is not alerted to the transition. In order to design an algorithm to accomplish this task we must first analyze human perception as it pertains to visual understanding of dynamic scenes. There are three key visual mechanisms developed over millions of years of evolution to cope with the tracking of objects in motion: saccade, smooth pursuit, and optokinetic nystagmus. There is also the intellectual process of determining that the visual scene is a plausible reality.

The saccade is a rapid eye movement which corresponds to jumping from one object to another over a relatively large distance. When the eye performs a saccade, the acuity threshold of the visual system drops an order of magnitude so as not to overload the brain with too much information. This is called saccadic suppression. When the eye comes to rest in the area of interest, visual acuity returns to normal after a small delay. This is usually followed by a corrective saccade to account for any error in targeting. Saccades are both voluntary and involuntary. The saccade, in and of itself, does not participate in tracking of objects in motion, but is used by smooth pursuit and optokinetic nystagmus.

Smooth pursuit is the motion of the eyes when tracking a moving object. Pursuit is initiated when an object of interest begins to move out of the fovea region (highest acuity) of the eye. The brain, detecting the error in position of the object, initiates a saccade which jumps the eye just ahead of the moving object. The eye then smoothly tracks the object, remaining just ahead of it, during its

travel. If the object begins to accelerate, the brain compensates for the change in velocity such that the object remains in the desired view spot. If the object accelerates too quickly such that it successfully departs the foveal region, the brain initiates another saccade to correct for the displacement, then compensates for the new velocity as best it can. These corrective saccades are very noticeable, such as when a person is trying to track a rapidly moving fly and can't "maintain lock" on the target..

Optokinetic nystagmus is a phenomena which occurs when a person watches a rapidly moving pattern. Since the brain cannot process all the information associated with a rapidly moving pattern within its visual field, it forces the eye to track a fixed point on the moving object, then performs a saccade to return to the original position in space, where it acquired the original fixed point, to acquire another fixed point. This repetitive motion resembles that of a mechanical typewriter carriage return. Examples of optokinetic nystagmus are: watching a picket fence while driving in a car, watching the turbine of a jet engine spin up, or watching a merry-go-round.

Intellectual verification of scene plausibility is the comparison of a viewed scene with memories of similar scenes and behaviors, and authenticating or dismissing the comparison. Many optical illusions can be created to illustrate this point. Figure 3-23 is a commonly known optical illusion that violates spatial relations. Other violations of scene plausibility occur when objects in motion violate the laws of Newtonian physics, such as an object moving from one place to another in an unreasonably short period of time, or a cartoon character running off a cliff but not falling until he looks down.

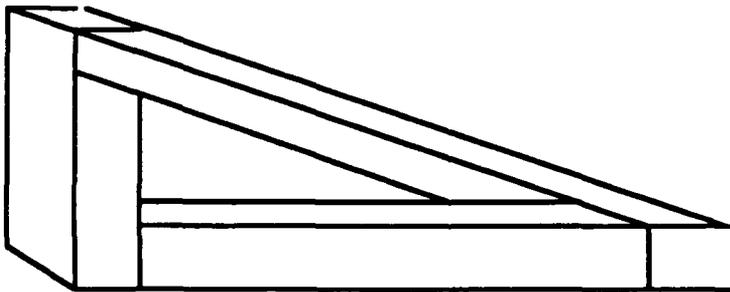


Figure 3-23 - Optical Illusion . The human brain can reason that this object cannot exist based on past experience in spatial relations.

These three visual mechanisms, coupled with reasoning ability, dictate some basic requirements that an acceptable smoothing algorithm must address:

- Entities must not instantaneously change position. This comes from the need to avoid saccadic corrections during smooth pursuit, and avoid violating scene plausibility.

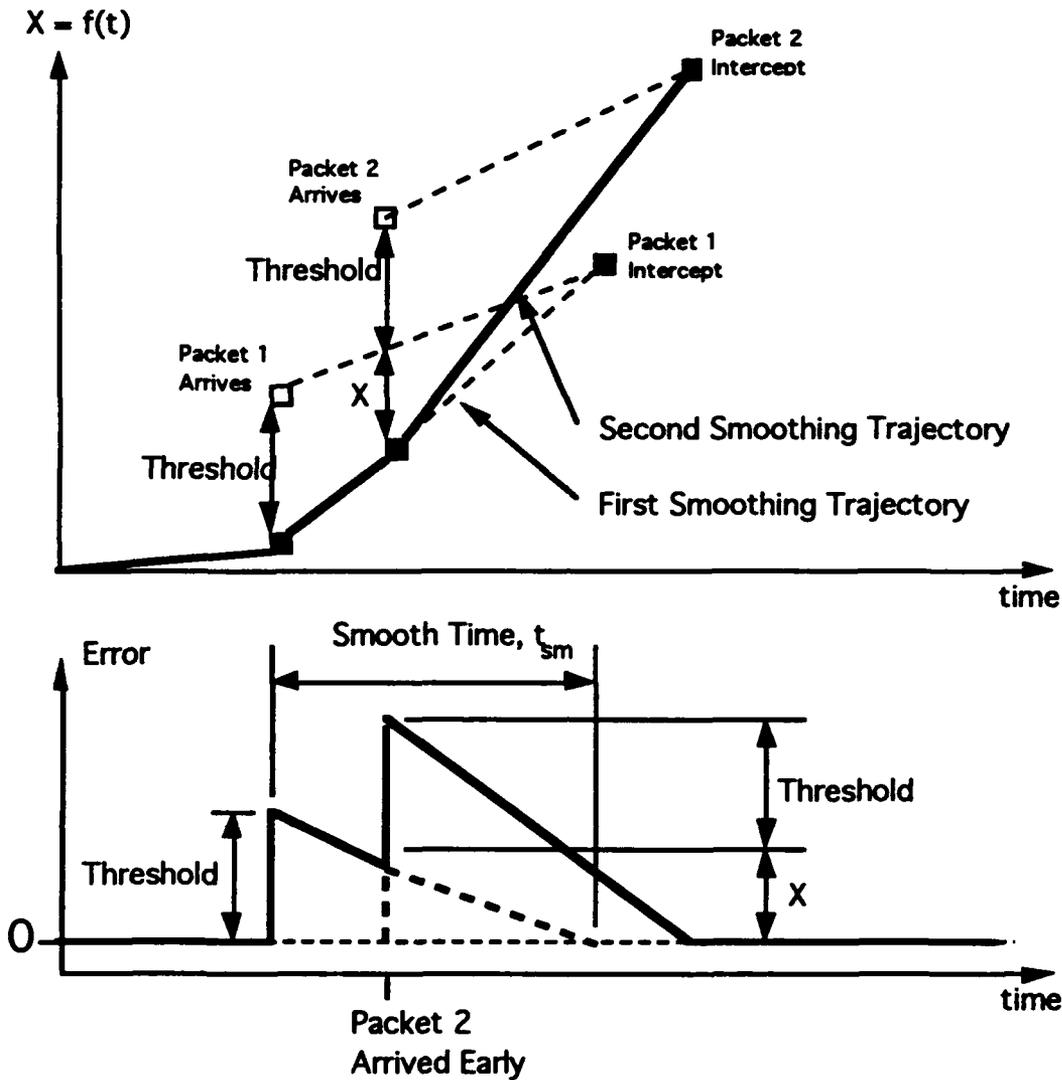


Figure 3-26 - Error Between Smoothed Trajectory and Dead Reckoned Trajectory for SIMNET Smoothing When Packet Arrives Early. Packet 2 arrived before the smoothing trajectory could reach the intercept point. The position error is $Threshold + X$. If this happens repeatedly, the error will continue to grow. For this reason it is very important to choose t_{sm} sufficiently small such that the smoothing has a chance to finish before the arrival of the next packet.

- The smoothing should look realistic, should not be visually noticeable, and should not distract the user.
- The position error between the smoothed entity and the straight dead reckoned entity should be minimized.

Guidelines for evaluating potential smoothing algorithms are:

- Accelerations on an entity should be minimized during smoothing.
- Unpredictable packet arrival time should not induce cumulative position error.

With these guidelines, smoothing algorithms may be quantitatively, as well as qualitatively evaluated.

3.4.4 Timebase Correction

This algorithm corrects the visual anomalies caused by different quantizations of time, due to different base frame rates at sender and receiver. See figure 3-28.

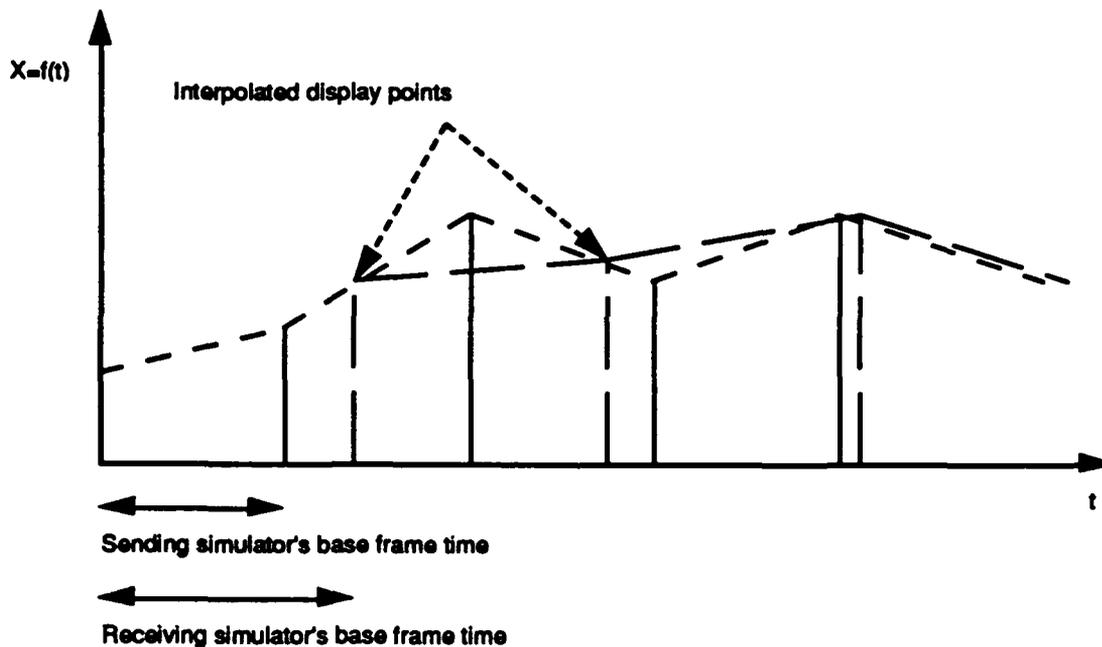


Figure 3-28: Timestamping

Relative timestamping, defined in later versions of SIMNET and in DIS, is sufficient to define the track as a function of time, and thus allow the receiving simulation to determine where to display the entity.

3.4.5 Forward Reckoning Algorithms

As stated earlier in this section, latencies associated with network delay and internal processing, cause events to be displayed to the user which are outdated by some small period of time. Some simulator makers have proposed a concept known as "forward reckoning" to compensate for this delay. Entity states are extrapolated into the future, and display devices are primed with this predicted image, such that the events are presented to the user at approximately the same time as they occur. Figure 3-29 shows the problem that forward reckoning tries to solve, and Figure 3-30 shows the forward reckoning algorithm. At first glance this seems to be a clever way to compensate for network and processing latency. There are some artifacts created by this paradigm, however, which make it less appealing.

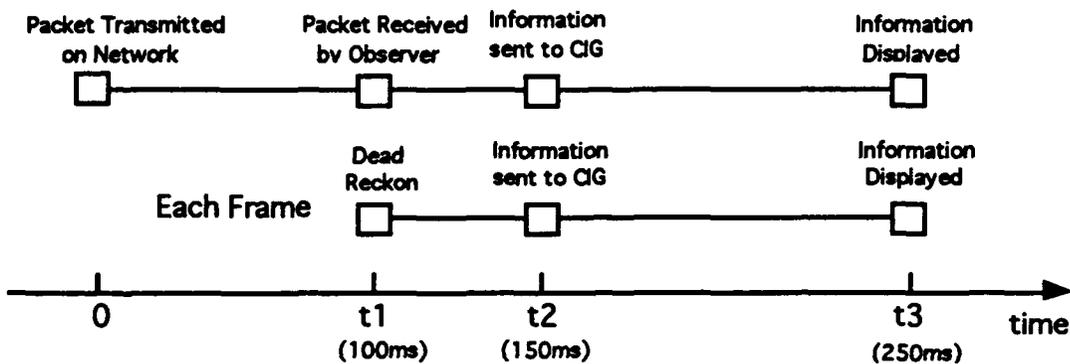
A good way to analyze the effects of forward reckoning is to take a sample case, and compare the actual trajectory of an entity to a normal dead reckoned display and to a forward reckoned display. By plotting position as a function of time for each of these cases, the time and position errors can be compared. For this example we will examine a missile, initially at rest, fired at a stationary tank 50 meters away. The missile undergoes a constant acceleration of 100 m/s^2 for a period of 1 second before it hits the target. The dead reckoning model will be second order, and the threshold at which a new packet is sent is 1 meter. Figure 3-31 shows the actual trajectory of the missile as simulated in its host computer. We will assume a total latency, t_3 , of 250 milliseconds. We will also assume, for this analysis, that the simulation frame time is small compared to the latency, so that effects of a discrete frame will not be a factor.

The first task in the analysis is to determine at what times appearance packets are issued by the missile simulator. At time 0, the missile begins to accelerate. The time at which the threshold is exceeded can be computed with the equation:

$$s = \frac{1}{2} a t^2 \quad (1)$$

Where s is the threshold (1 meter), a is the acceleration (100 m/s^2), and t is the time at which the threshold is reached, computed to be 0.14142 seconds. The first packet, which is transmitted at time t , contains:

Normal Processing and Display of Remote Entities



Forward Reckoning Entity Information

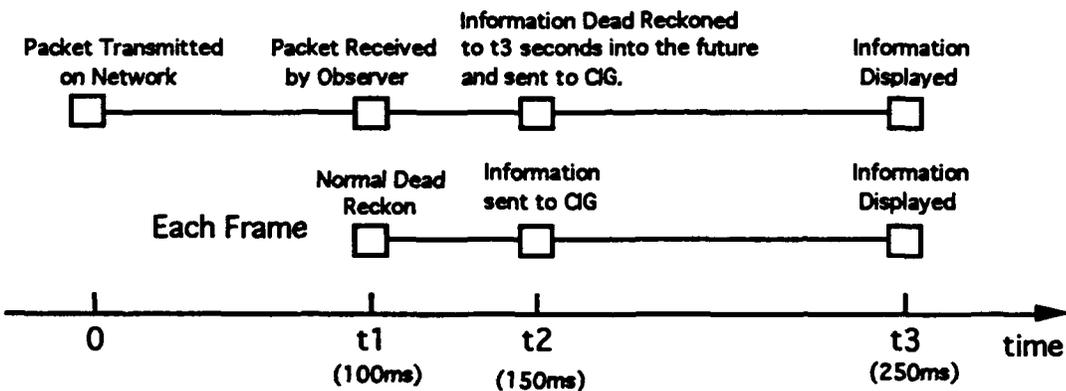


Figure 3-30 - Forward Reckoning. The top timeline shows that by the time entity position information is displayed to the user, it has been delayed by t3 seconds from the time it was put out on the network. The bottom timeline shows the forward reckoning paradigm. Upon receipt of a packet, the entity is dead reckoned t3 seconds into the future before being sent to the CIG. When the entity is displayed it should be closer in space to its actual position. The entity is dead reckoned in the normal way from that point onward.

position = 50 meters

velocity = 0 m/s

acceleration = 0 m/s²

Knowing what the network packets contain, and knowing when they are transmitted, the dead reckoned and forward reckoned trajectories can be constructed. Figure 3-32 shows the actual trajectory of the vehicle alongside the computed position at the receiver's host, and the final displayed trajectory, for normal dead reckoning. The first packet is transmitted at 0.1414 seconds and is received at $t1 + 0.1414$. This causes a 1 meter jump in the position of the missile. The computed missile trajectory then follows the actual trajectory with a lag of $t1$. The displayed trajectory is exactly the same as the computed trajectory, with an additional lag of $t3 - t1$, or 150ms. The total effect of latency and dead reckoning is a time lag of 250 ms and two 1 meter jump discontinuities in the displayed trajectory. This solution represents the worst case with respect to time lag, but the best case as far as position accuracy. The largest position error that will occur is the threshold.

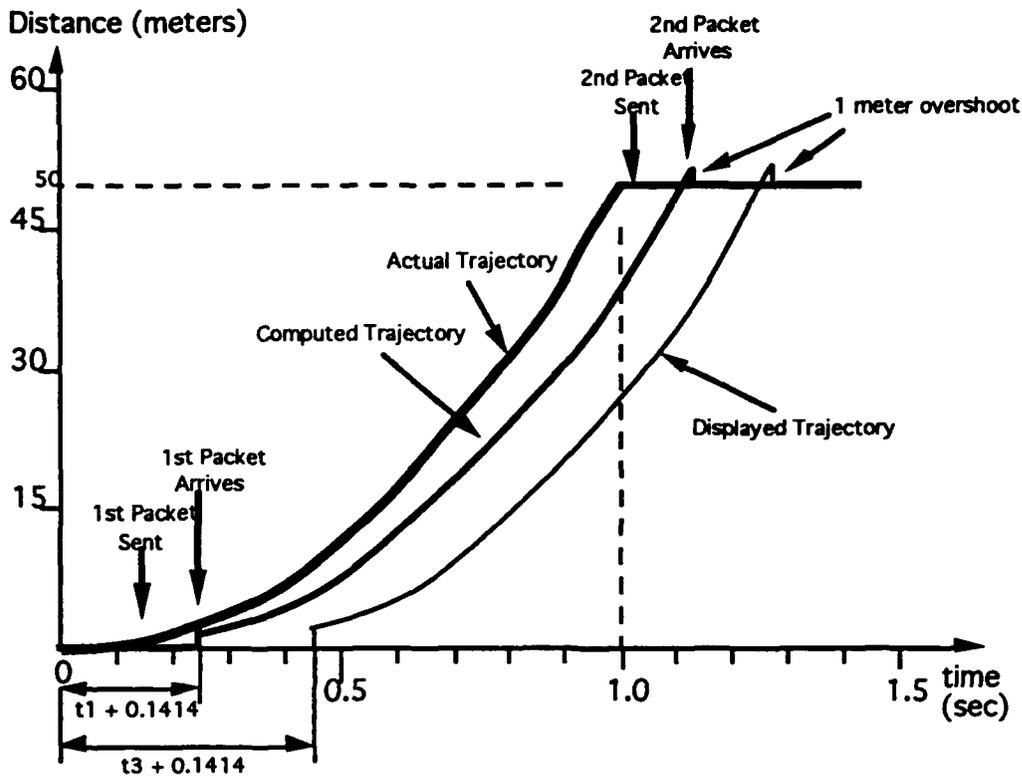


Figure 3-32 - Displayed Position of Missile as Compared to Actual Trajectory for Normal Dead Reckoning. The computed trajectory has a latency of t_1 , which is the network latency, with respect to the actual trajectory. The first packet is sent at 0.1414 seconds, so it does not arrive until $t_1 + 0.1414$, causing the sharp 1 meter correction in the computed trajectory. The displayed trajectory is exactly the same as the computed trajectory, with a time lag of $t_3 - t_1$, or 150ms in this case. The total time lag from the actual trajectory to the displayed trajectory is t_3 , or 250 ms. The positional error, however, is never more than the threshold, which is 1 meter.

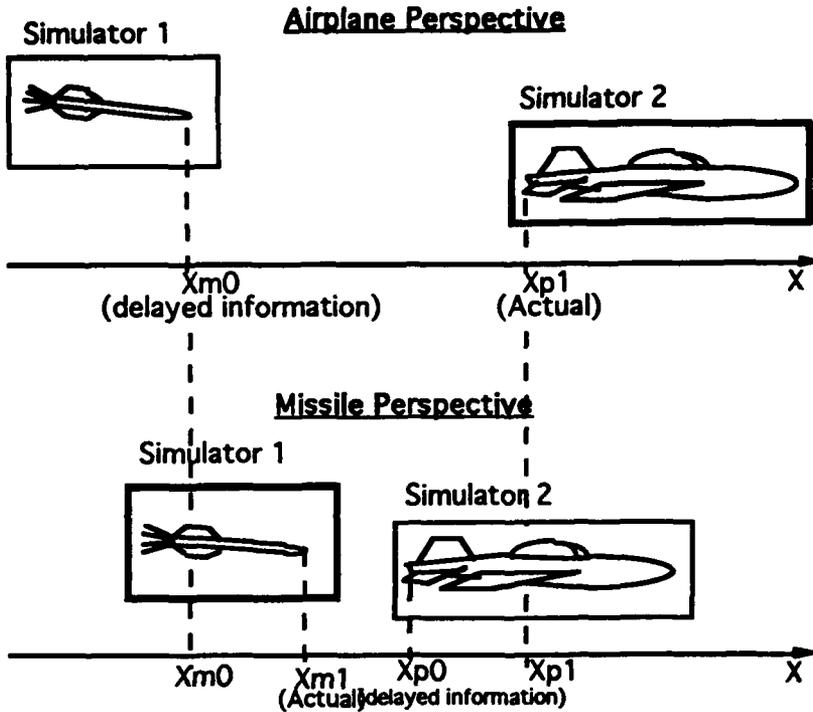


Figure 3-35 - Latency Problem in Terminal Guidance. The top picture shows the impression of the state of both entities from the airplane's point of view. The airplane sees itself in its actual position, while the missile is rendered further away due to network and processing delays. The second picture, however, shows that the missile sees itself in the correct position but sees an old version of the airplane position. The missile will decide it has hit the airplane before the airplane has a chance to make a successful final avoidance maneuver.

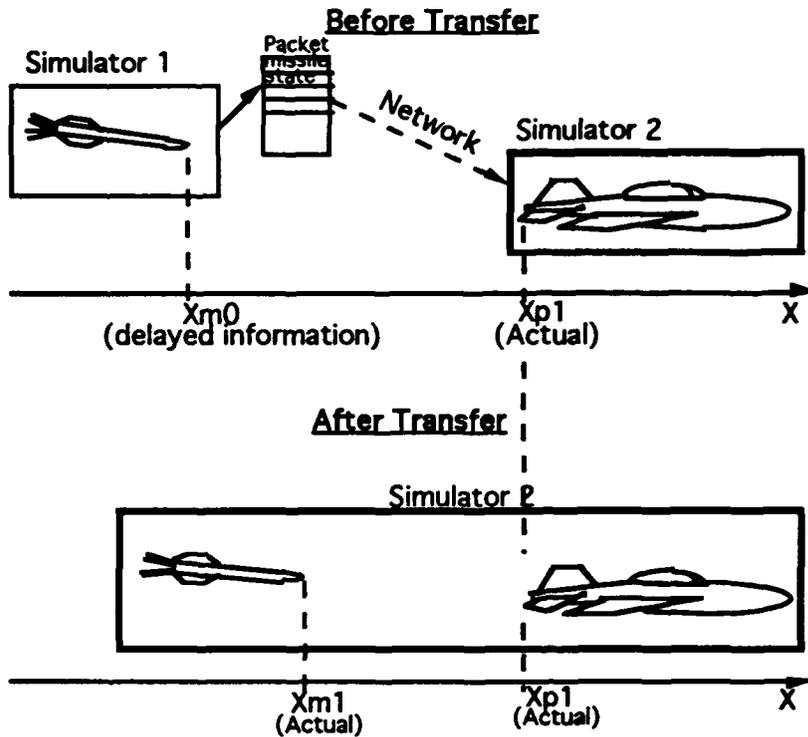


Figure 3-36 - Object Migration Solves Terminal Guidance Problem. By packaging up the missile state and sending it to the airplane node across the network, the airplane node can simulate the airplane as well as the missile. All problems associated with network and processing latency disappear when both entities are simulated on the same node. This is a general solution for many problems which require a tighter coupling between two remote entities.

patches in a gradual fashion, and reduce false cueing. The off-line and real-time management of these data sets of varying size and resolution presents a significant correlation challenge.

4.1.4 Electronic Warfare

The outcome of modern battlefield engagements is becoming increasingly reliant on effective use of the electromagnetic spectrum. For this reason extensive use of ECM and ECCM techniques have become an integral part of any comprehensive battle plan. Consequently the EW (Electronic Warfare) portion of simulation efforts has also increased in scope and in priority. Four aspects of the EW simulation problem are mentioned below. Additional information is provided in section 7.2.

First, in considering electronic warfare systems, it must be recognized that faint signals must be considered and that a very significant amount of data on the signal structure and the spatial/temporal characteristics of the beam are required. Hence antenna patterns and signatures must cover a sizeable fraction of 4π space. At the same time, many radiations exist, including numerous ones from highly complex emitters (jammers, advanced radars, etc.).

Second, regardless of the update rate on information between linked simulators, there are always processes in EW which are significantly faster than any update rate (up to 10 KHz rates would be required if a brute force approach were used). At the same time there are long, complex actions which can adequately be described by defining action, initiation time, and electronic dead reckoning.

Further, EW simulation does not permit an approach in which a sensor communicates only with its target entity. Every entity with an intercept receiver is a potential receiver of every signal. The path from every emitter to every appropriate receive entity must be considered to determine whether the received signals are above the thresholds of the equipment when considering weather and such terrain effects as ducting and diffraction (if these are part of the simulator).

Still a third part of the problem lies in the effect of weather and natural environment effects on the electromagnetic elements. Advanced simulators will contain a complex weather model, varying in three dimensions and time. This weather produces an appreciable effect on electromagnetic propagation which can be unique between each pair of entities even if they are at the same range (so that the r -squared losses are identical).

This imposes a coherence problem between classes of simulators similar to that between simulators with different terrain fidelities. In the terrain case, visibility (radar or optic) is different for the two entities in different simulators because one simulator recognizes a level of detail which includes an occulting feature. In the weather effects case, the problem is that of differential visibility in

A secondary problem lies in retaining temporal and spatial coherence, particularly for violent, moving weather cells which can have major impacts on IR, EO, and RF.

Weapon Effects Weapon effects are similar to weather effects, the primary difference being that they are man-made. Weapon effects include dust, smoke, flares, chaff, flame, explosion, tracers, rotor wash, sea state, and dynamic shadows. They exhibit many of the same properties as weather effects. The simulation of these effects can vary widely from simulator to simulator.

Dynamic Terrain When a weapon detonates on the ground it changes the terrain - craters are formed, buildings are destroyed, and dams burst. Berms may be erected or trenches dug for defensive purposes. The occurrence of these transient changes in the environment during an exercise is referred to as dynamic terrain. Within the DIS environment we will be faced with the task of determining how differing approaches to implementing dynamic terrain can be integrated into the same exercise. This topic is discussed further in section 7.1.

4.2 Terrain Data Processing

Terrain data processing from off-line data preparation to real-time processing and rendering is now discussed, primarily in the context of visual systems. This will lead to a description of a proposed correlation measurement methodology.

Figure 4-2 illustrates the processing flow experienced by terrain data from the acquisition of source data in grid form to the intermediate polygonal mesh form of the Local Data Base and finally to the rendered result as viewed by the warfighter on the simulator display. The gridded terrain data is similar to Project 2851 SSDB terrain data; in DIS terminology it is referred to as the BATTLEFIELD Data Base. The processing that is performed on this gridded terrain data by the Off-line Data Base Processing and Real-time Simulator Processing devices is controlled by standardized specifications, called SIMWORLD data, for the off-line and real-time systems. The intermediate form of the terrain data, the polygon mesh in IG specific format, is referred to as the simulator Local Data Base.

4.2.2 Real-time Simulator Processing

Figure 4-4 illustrates a typical real-time visual system processing flow. Terrain polygons are culled based on field of view and range, discarded based on projected size, orientation, or occlusion by another feature, and transitioned from higher to lower detail representations by Level Of Detail (LOD) and range controls. The net result is that the terrain surface dynamically changes in real-time; the number of terrain polygons that survive to the display continuously varies under the control of the real-time load management controls. Therefore, the load management algorithm will bear heavily on the ultimate interoperability of a given visual system.

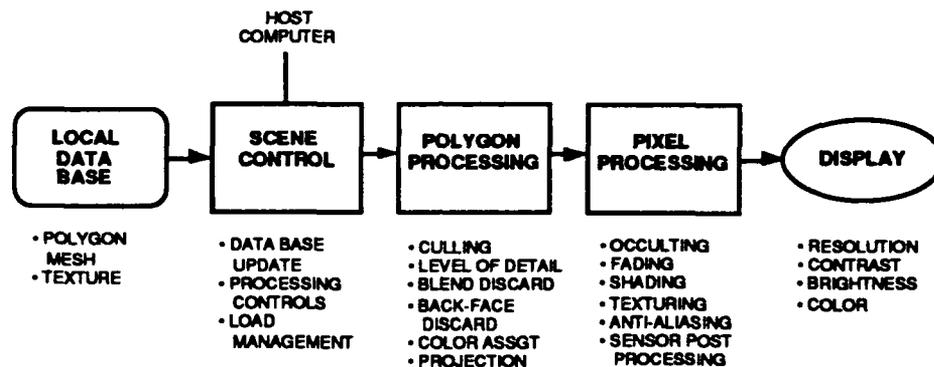


Figure 4-4: Typical Real-time Visual System Processing Flow

4.3 Correlation Metrics and Interoperability

The overall framework for developing and testing terrain correlation metrics and for determining interoperability is now presented; see figure 4-5. Data is captured at three points: source data in grid form, local data base data in polygon form, and the end result of the rendering process in pixel form. Correlation metrics are required to compare the local data base with the source data, and to compare the processing results with the local data base. These metrics are then used to determine interoperability by comparing them with "exercise validity" criteria which are derived from consideration of the application and the types of simulation devices that are planned for the exercise. It is expected that exercise validity criteria will need to be developed empirically, and then tested and refined based on subsequent warfighter-in-the-loop experiments.

5. Overload Management

5.1 Network Overload Management

5.1.1 DIS Systems Are Queueing Systems

DIS is a network. Like any network, it can be thought of as a system of queues. Both communications links (e.g., T-1, 56Kbit) and communications processors (e.g., routers, CIUs/CAUs, simulation nodes) can be modeled as queues. Each queue has a maximum number of elements (a length), and a service time for each element.

This Section examines what queues can be overloaded in a DIS system, what anomalies are manifest to the warfighters as a result, and what techniques can be used to minimize these anomalies.

5.1.2 What Queues Can be Overloaded?

Two types of queues can become overloaded. Communications link queues have limited bandwidth with which to transmit packets, and communications processor queues have limited compute power with which to route or otherwise process packets.

In order to better separate the effects of these overloads, we first examine the effects of limited bandwidth while ignoring any effects of limited compute power. Next, we examine the effects of limited compute power while ignoring limited bandwidth.

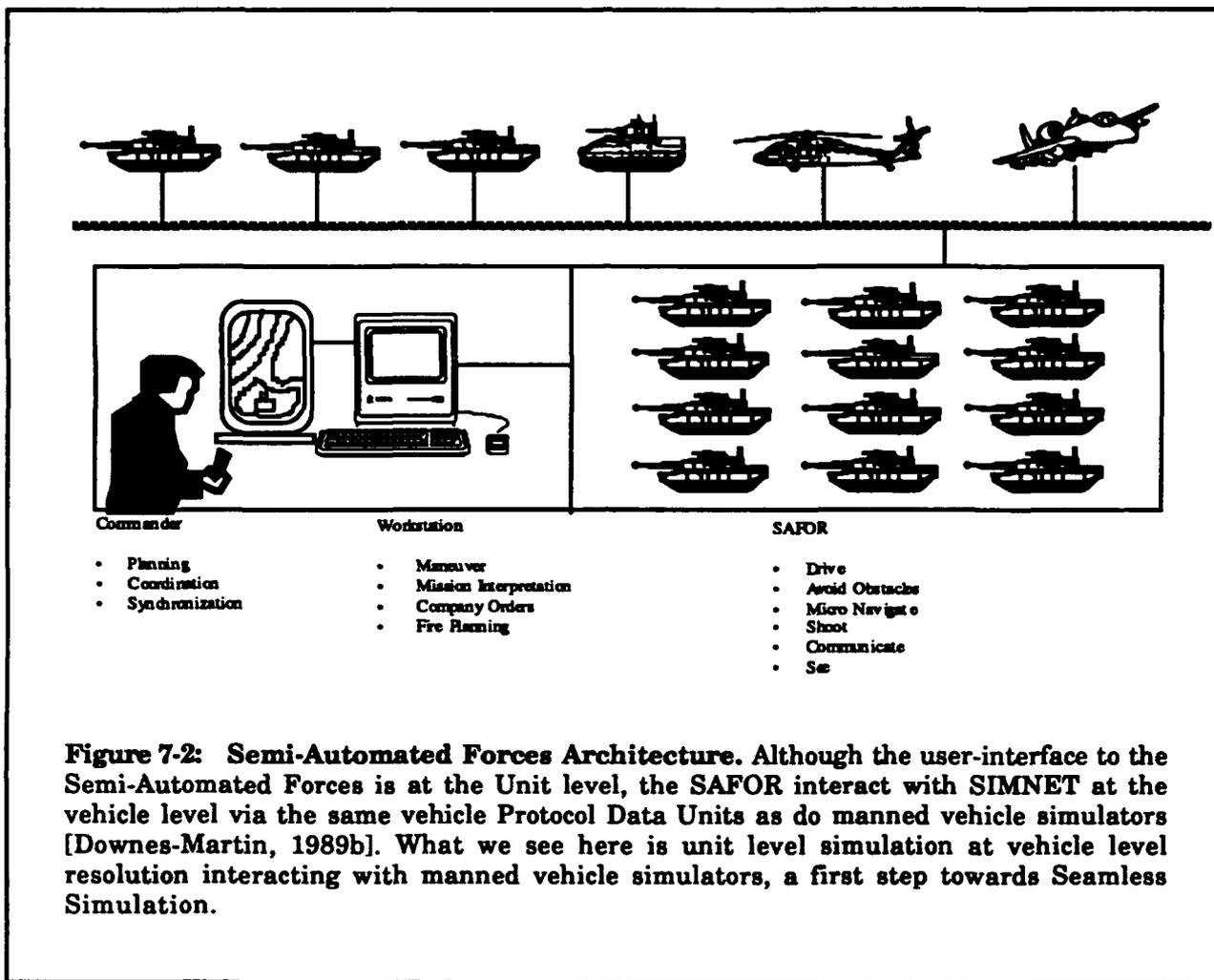
5.1.2.1 Communications Link Queues

DIS traffic is routed over an Internet of WANs and LANs. If the packet arrival rate at a given communications link queue exceeds its capacity for any length of time, the size of the queue will tend towards infinity. This means that packets may sit on the queue for different lengths of time, thus experiencing varying latencies. Further, since the queue is necessarily of finite size, packets will be dropped.

Such will happen, however temporarily, when the underlying network is not sized for the maximum burst rate of traffic. This may be likely if either dynamic multicasting or dynamic CIU-to-CIU routing is used, because then the traffic pattern over any given link in the network may be very hard to estimate *a-priori*.

Varying latency is manifest to the warfighter as discontinuous battlefield entity trajectories on the out-the-window display (See Section 3.) Smoothing may make the trajectory appear more continuous, but it may be computationally

The table indicates one possible ordering of algorithm degradation to provide a continuous reduction of computational load. For a discussion of the metrics of computational loading due to dead reckoning and smoothing see section 3.3.1. The driving requirement for degradation ordering is the desire for the simulation task to be minimally impacted by the degradation.



Both DARPA and PMTRADE have recently announced major initiatives in distributed simulation applied to team training. These initiatives will be in addition to those already seen as extensions and expansions of the SIMNET distributed simulation technology [Pasha, 1991a, 1991b]. PMTRADE will extend the state of the art in DIS under their Battlefield Distributed Simulation -- Developmental (BDS-D) program, while DARPA continues its interest in the area under the umbrella initiative "Advanced Warfighting Simulation" (AWS). These two umbrella initiatives overlap each other considerably, although it is PMTRADE who is responsible for producing training products based on distributed simulation (such as CCTT).

For these reasons, DARPA and PMTRADE are interested, along with the services, in expanding the current DIS technology along a number of dimensions (see Figure 7-3) [McBride, 1990]. These dimensions include: vertical expansion from the current tactical battalion sized battles of hundreds of vehicles to Joint/Theater with tens of thousands of vehicles; horizontal expansion to include

command and the vehicle level of combat execution increases beyond the normal "look down two, command down one". The CGF commander cannot effectively supervise his subordinates below two levels down without becoming overloaded, and thus the CGF must become autonomous and smart at the lower levels of command in order to be "indistinguishable from manned simulators".

- **Live Ranges and Operational Equipment.** As weaponry increases in complexity, lethality and range, it becomes increasingly harder to provide a complete battlefield environment on live ranges. Integration of live range platforms and distributed simulation is considered a possible solution to this problem.

The basic technical challenge here is to provide an interface between simulations of heterogeneous granularity such that each simulation or piece of equipment sees a rational world according to its expectations, and that the various worlds seen by all the participants are consistent with each other both in time and space. Aggregation and de-aggregation of simulation entities must be carried out dynamically, and the distributed simulation network protocol must be extended to take account of the new non-vehicle types of object on the net.

The Industry/DoD Standards for Interoperability of Defense Simulations Workshop, funded by DARPA and PMTRADE, is administrated by the Institute for Simulation and Training at the University of Central Florida. The goal is to develop standards that will allow vendors to internet their own vehicle simulators to those from any other vendor without knowledge of the internals of their competitors machines. The effort includes working groups to handle human-machine interfaces, network protocols, and terrain databases. This effort has so far restricted itself to interconnecting vehicles only, but will shortly be considering interconnecting wargames with vehicles. A number of draft standards and position papers have been published [IST, 1989, 1990, 1991; Pope 1989].

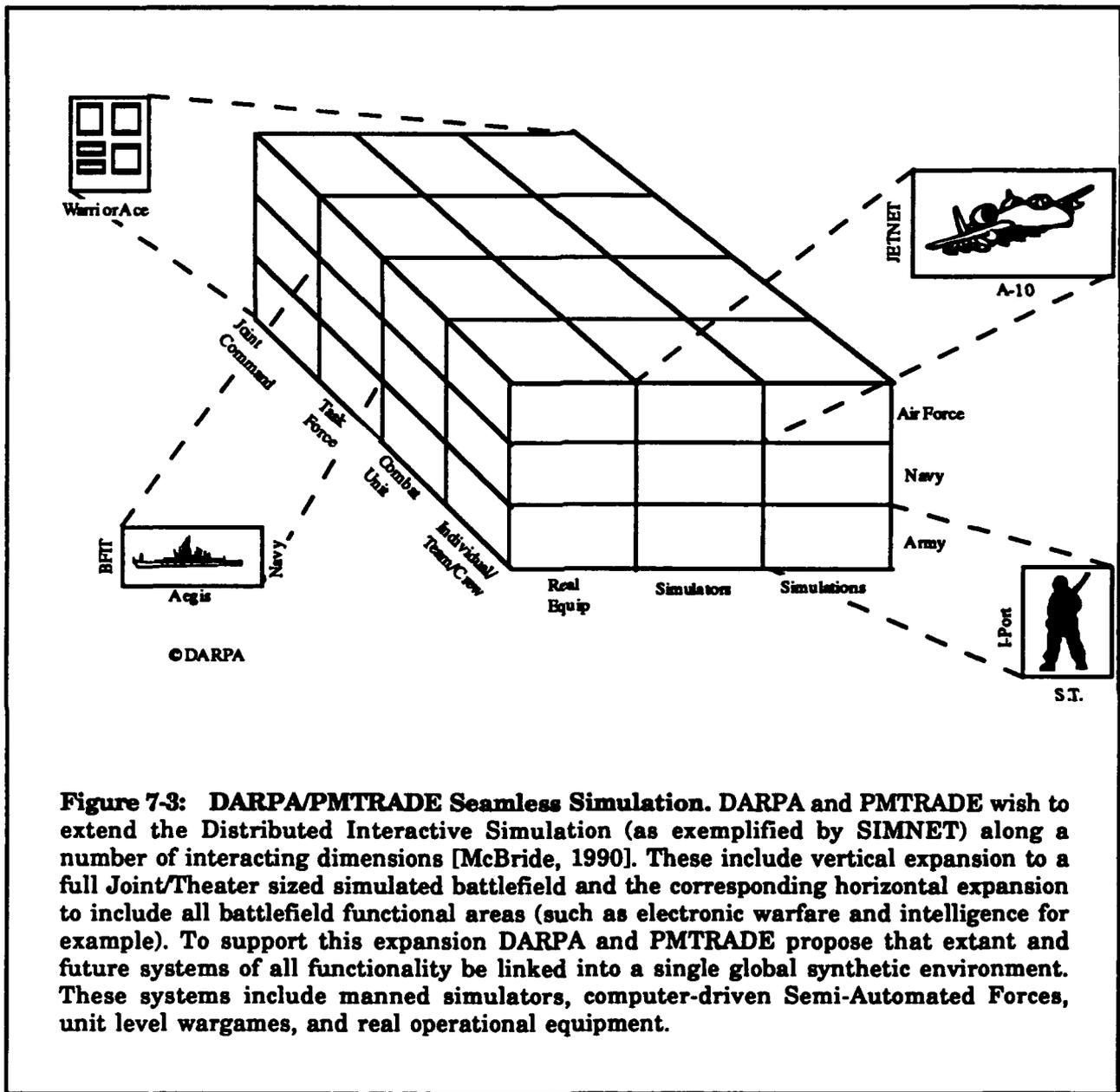


Figure 7-3: DARPA/PMTRADE Seamless Simulation. DARPA and PMTRADE wish to extend the Distributed Interactive Simulation (as exemplified by SIMNET) along a number of interacting dimensions [McBride, 1990]. These include vertical expansion to a full Joint/Theater sized simulated battlefield and the corresponding horizontal expansion to include all battlefield functional areas (such as electronic warfare and intelligence for example). To support this expansion DARPA and PMTRADE propose that extant and future systems of all functionality be linked into a single global synthetic environment. These systems include manned simulators, computer-driven Semi-Automated Forces, unit level wargames, and real operational equipment.

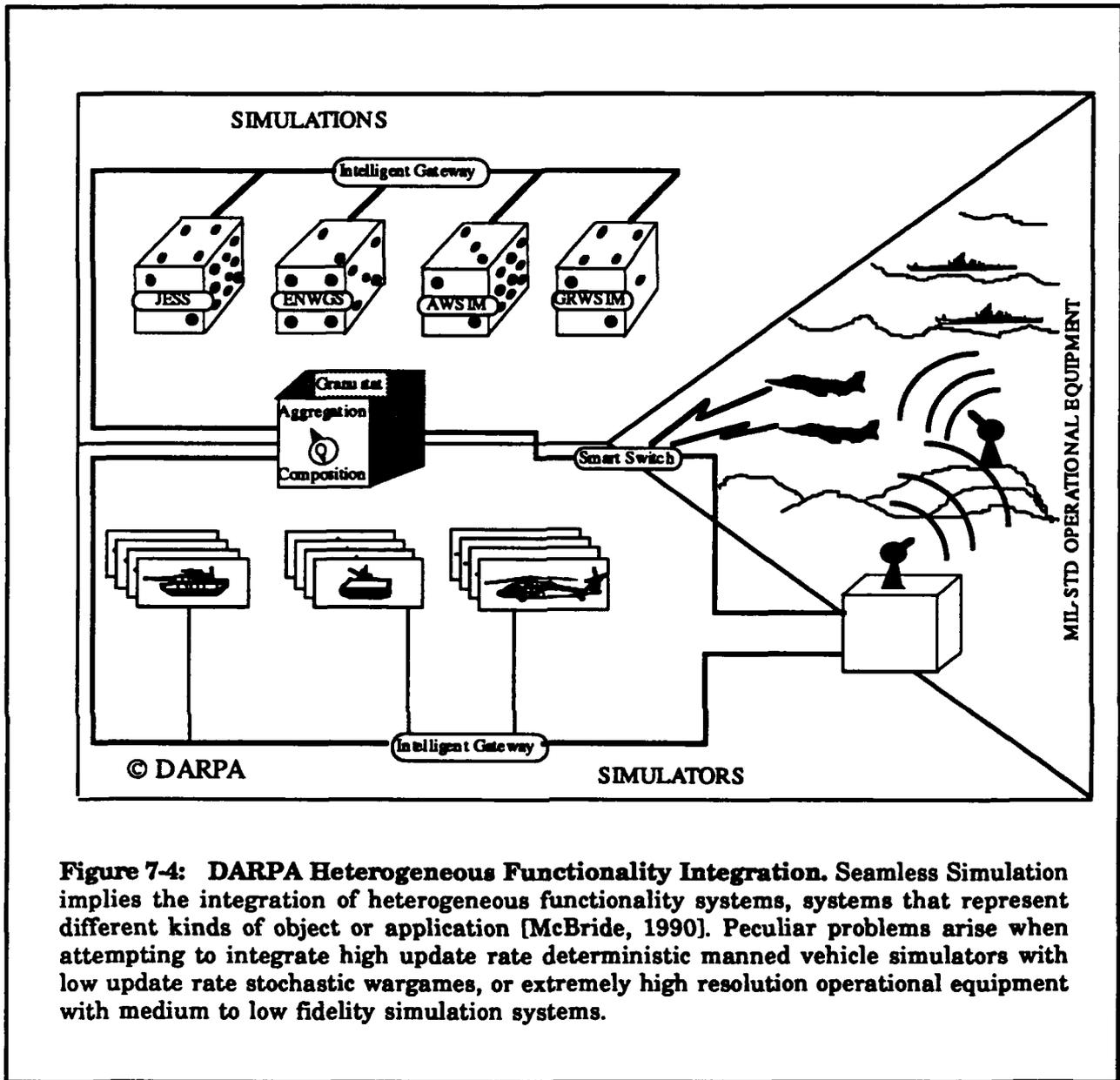


Figure 7-4: DARPA Heterogeneous Functionality Integration. Seamless Simulation implies the integration of heterogeneous functionality systems, systems that represent different kinds of object or application [McBride, 1990]. Peculiar problems arise when attempting to integrate high update rate deterministic manned vehicle simulators with low update rate stochastic wargames, or extremely high resolution operational equipment with medium to low fidelity simulation systems.

6.2 Overview of Seamless Simulation Issues

Seamless Simulation involves the integration into a single synthetic environment of many distributed objects with heterogeneous functionality and heterogeneous implementation. Some of the issues are common to those involved in integrating vehicle level simulations, such as terrain data bases, communications architectures, and interfaces. However, the current DARPA/PMTRADE sponsored workshop on DoD/Industry Standards for the Interoperability of Defense Simulations is already handling these in great detail. This detail is firmly based on the assumption of homogeneous functionality

systems, i.e. interaction of vehicles at the vehicle level of resolution. The overview of Seamless Simulation issues provided here is an attempt to review from literature a general set of issues that cover heterogeneous functionality systems. These issues should subsume those for the vehicle level. It should be noted however, that the issues discussed here are drawn from a variety of sources, and do not necessarily comprise a systematic theory of Seamless Simulation. The overview provided here is for the purpose of providing a background for the literature review. An attempt is made in this report to place these issues into a broad framework, however a proper theory of Seamless Simulation will be attempted in a later report.

One goal of the DIS Architecture is to support Seamless Simulation, the linking of heterogeneous functionality heterogeneous implementation systems into a consistent synthetic environment. This goal breaks down into two subgoals, both of which must be considered:

- **Synthetic Environment Construction.** The DIS will make use of a large number of heterogeneous systems to create a synthetic environment. Selection must be made before run time of which applications, processes, databases, and user interfaces (and which versions) are going to be used. A large number of general architecture issues are relevant, including the construction of new applications before run time by integrating components from multiple applications across the network.
- **Synthetic Environment Dynamics.** The DIS during run time must maintain a global internal consistency and satisfy user criteria of military reality. Application interactivity at the conceptual and dynamic level are the fundamental issues applicable to this subgoal.

These subgoals in turn generate three classes of requirements:

- **Simulation Truth.** There exists a simulated ground truth, that is the same no matter which system is used to interface to the DIS. Each interlinked system may or may not deal with uncertainty, intelligence, or the fog of war by suitably filtering this ground truth in some way.
- **Conceptual Consistency.** The DIS must be globally consistent in conceptual terms. Each interlinked system must be able to interact with the DIS using its own conceptual structures. For example, a vehicle simulation interacting with a unit level simulation must be able to interact with the units in terms of vehicles. Conversely, the unit simulation must be able to interact with the vehicle simulation in terms of units. This means that computer driven units or vehicles must interact with humans in a credibly realistic fashion.
- **Temporal Consistency.** The DIS must be globally consistent in temporal terms [Weatherley et al, 1991]. Each interlinked system must interact

with the DIS in a timely and causal fashion, maintaining temporal logic. For example, a fast update-rate vehicle simulation interacting with a slow update-rate unit simulation must perceive the DIS as containing fast update-rate vehicles, and those (unit generated) vehicles must respond to the vehicle simulation at the same update rate.

These requirements in turn generate four major technical dimensions to the problem of generating a DIS architecture that supports Seamless Simulation:

- **System Architecture.** System architecture determines the mechanics of how the distributed system is put together, including the communications between each interlinked system. Hardware and software issues are dealt with here. [Weatherley et al, 1991].
- **Data Management.** "Data Management is responsible for the rules by which data can be changed, and the interpretation of data values." Conceptual consistency issues are dealt with here, including aggregation and deaggregation of units, and terrain consistency. [Weatherley et al, 1991].
- **Human Machine Interaction.** Human Machine Interaction covers issues such as how simulation agents are to be built that simulate human performance to an acceptable degree of realism, the knowledge bases required to support the simulation of human behavior, and the cognitive models. [Downes-Martin, 1989a], [Deutsch, 1989], [Abrett et al, 1990a, 1990b].
- **Time Management.** Time management deals with the temporal consistency issues, timeliness and temporal logic, or causality, of the Seamless Simulation. This includes the different time rates at which say a unit simulation runs compared with a vehicle simulation. [Weatherley et al, 1991].

These dimensions in turn break down into component issues, and a certain amount of flexibility occurs in determining to which dimension each component issue belongs. It should be noted that the System Architecture and Data Management dimensions are similar to those raised by the Object management Group [Soley, 1990] in their architecture for integrating heterogeneous business applications across networks. Time Management arises in Seamless Simulation due to the competitive nature of combat and the resultant requirement for simultaneity and temporal logic.

Seamless Simulation generalizes interconnected vehicle simulations to interconnected defense simulations. The goal is to provide a simulated battlefield at theater level in which aggregate wargames (both current and future), vehicle simulators, actual field equipment (both combat vehicles and C3I assets), individual soldier "virtual reality" ports, and computer generated forces all interact in such a way that the seams between the technologies are hidden from

the participants. This includes interconnecting real platforms on live ranges with manned platform simulators, and integrating the individual soldier on the simulated battlefield as though he were on foot and operating battlefield equipment. The types of system that are eligible for seamless integration fall into five broad categories:

- **Manned platform simulators.** These interact directly with other platforms with the humans in the loop making all decisions. These systems provide user interfaces to platform level simulations.
- **Computer Generated Forces (CGF).** These are commander interfaces to simulations of C3I hierarchies, with code that drives platform level simulations. All interactions are evaluated at the platform level. Any aggregation is strictly at the level of simulating the aggregation that takes place in the real world intelligence process (the perception of the world versus ground truth). The human in the loop is making unit level planning and execution decisions, and the system is interpreting these in order to drive many unmanned platform level simulations. The system must now be able to carry out decision making and decision interpretation (decomposition of mission into subordinate missions) as well as handle platform level simulation. The CGF is a unit level simulation at the platform level of resolution and interaction.
- **Unit level simulations and wargames.** These are commander interfaces to simulations of C3I hierarchies, with code that drives unit level simulations. These systems do not continuously maintain platform levels of resolution at all times. Interaction between battlefield entities is at some unit level of aggregation above platform. Those systems that dynamically provide selective units at platform levels of interaction are carrying out deaggregation (and aggregation if they later recombine the platforms into units). In many cases, because of the aggregated level of interaction, these systems use stochastic processes to replace the causal relationships that have been hidden by the aggregation. Furthermore, they often run at faster than real time. Thus these systems differ from DIS in two critical areas, they run at different speeds and at different levels of granularity.
- **Operational platforms.** Four requirements are generated by the integration of operational platforms to DIS. First, the physical system must be represented by software within the DIS battlefield so that it can be perceived by the other participants, and if appropriate its battlefield vulnerability be simulated. Second, the actual movement of the operational platform in the real world (particularly if we are dealing with an air asset) must be correlated to its movement in the DIS world. The major point here is the correlation of real world terrain to the DIS terrain database. Third, the informational interaction of the operational platform (especially if it plays a major C3I role) must be simulated. The

DIS world must have the assets and be able to respond in ways expected by the operational platform. Finally and most critical, the operational platform must be able to have a DIS interface package of hardware and software placed on board without danger to the crew or platform, and without major impact on the cost of the platform.

- **Live ranges.** For the purposes of seamless simulation, live ranges are considered to be already instrumented with a Range Central Command and Control (RCC) system. Integration with DIS means the linking of the Range Central Command and Control system with DIS. In addition to the problems of integrating operational platforms to DIS, the characteristics of the RCC must be dealt with. The more critical characteristics are those that differ significantly from DIS, ie low bandwidth on the RCC net and low broadcast rate and reliability of transmission by the range platforms.

The DIS architecture must provide support for seamless integration of these classes of system into the DIS network. This implies a linking technology that handles system to DIS rather than system to system pairwise connection. The classes of problem that have to be dealt with are:

- **Temporal Consistency.** The Seamless Simulation must be globally consistent in temporal terms. Each interlinked system must interact within the DIS in a timely and causal fashion, maintaining temporal logic. For example, a fast update-rate vehicle simulation interacting with a slow update-rate unit simulation must perceive the Seamless Simulation as containing fast update-rate vehicles, and those (unit generated) vehicles must respond to the vehicle simulation at the same update rate.
- **Spatial Consistency.** The Seamless Simulation must maintain spatial consistency in the face of different spatial resolutions. Unit level simulations may not only operate on different time aggregations, but also with different spatial aggregations.
- **Functional Consistency.** Simulation objects representing the same real world objects may be represented in different ways and with different functions and behaviors in different systems. When these systems are linked, these differences must be dealt with. Polymorphism is one way of handling this if and only if the differences are acceptable in operational terms for the purposes of the simulation.
- **Organizational Aggregation.** Different systems may aggregate the battlefield objects into different levels of interaction, for example the platform level of execution of DIS and unit levels of execution of higher order models. Seamless integration must provide all systems with what appears to be a battlefield at the level of interaction expected by that system.

The proposed DIS Architecture must support the linking of each of the five classes of system to DIS in a seamless fashion. The architecture must indicate where the solutions to the linking problems should be embedded, and provide a mechanism for open development of the linking technologies.

6.3 Manned Platform Simulators

The seamless simulation issues of manned simulators are dealt with elsewhere in this document under Time and Space Coherency.

6.4 Computer Generated Forces

Computer Generate Forces generate platform levels of execution, and are by design DIS compliant. They are dealt with in detail elsewhere in this document. To summarize however, the existence of the CGF will require two changes to the DIS message protocol, both dealing with simulated communications. First simulated tactical communications by CGF means that the radio message packet must be able to carry data structures representing CGF generated radio messages. Second, if the CGF software command posts are distributed, then simulated face to face communications between simulated staff functions must be carried by the inter-cell tier of the network.

6.5 Unit Level Simulations and Wargames

6.5.1 Basic Interoperability Issues

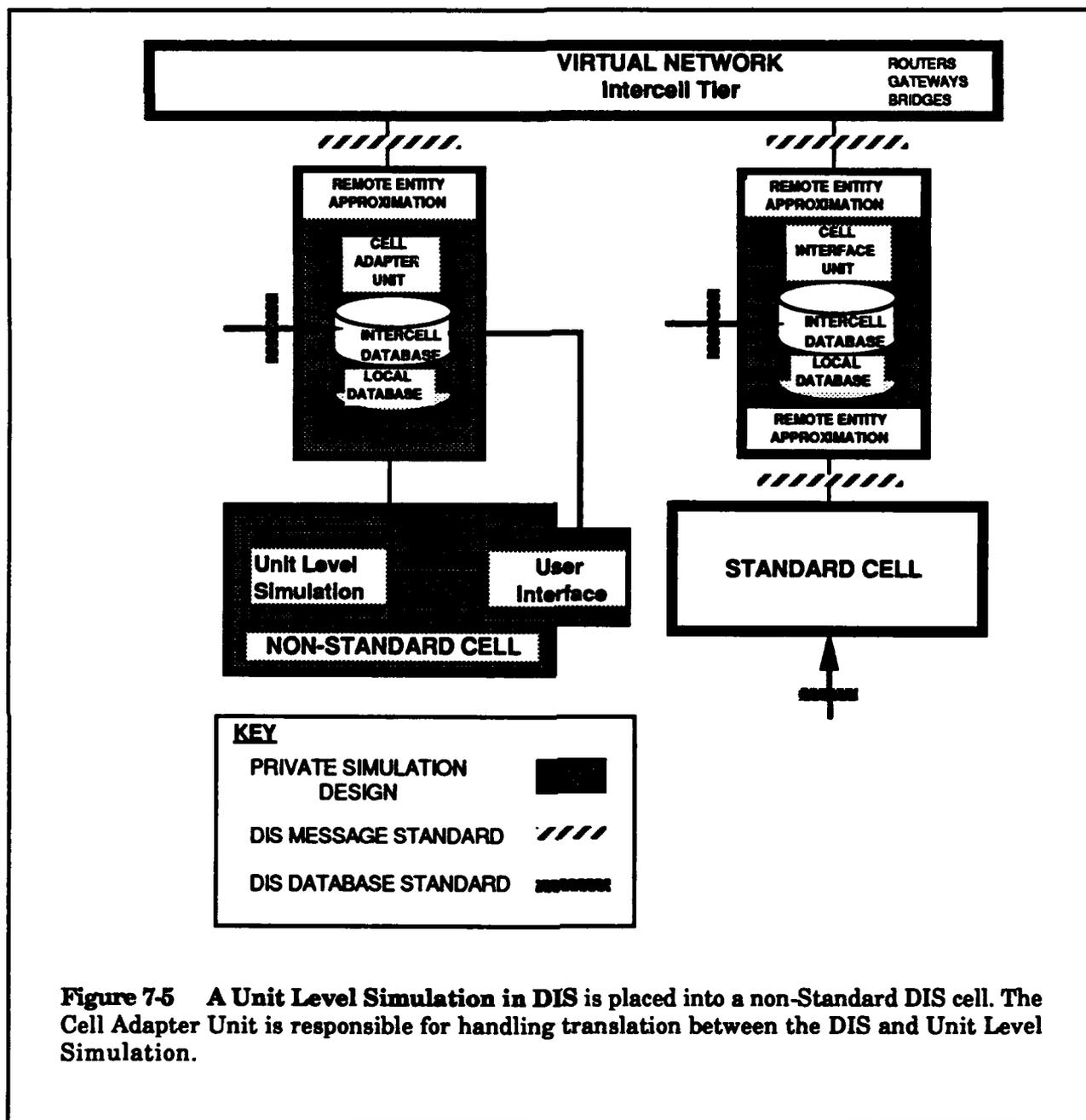
Some of the major operational objectives in integrating unit level simulations (higher order models) with DIS are:

- provide a C3I interface to large units on the DIS battlefield, inspectable at the platform level of resolution.
- make use of extant and accepted wargaming algorithms.
- examine and assess wargaming algorithms within a platform level of resolution environment.
- expand the scale of the DIS battlefield with less than linear rise in hardware and personnel costs.

The unique technical problem that has to be solved for integrating unit level simulations to DIS deals with the level of interaction. Time and space coherency are dealt with elsewhere in this document. DIS currently assumes a common level of interaction at the platform level of resolution. Any extant system which is to be integrated on DIS must have its internals preserved as much as possible, and thus be presented with information which is consistent with its internal data structures and internal assumptions. Otherwise, it would be more rational to

simply rewrite the unit level simulation. The DIS architecture supports the integration of unit level simulations by placing them into non-standard DIS cells, and places to translation and filtering between DIS and the unit level simulation into the cell adapter unit (see Figure 7-5). Note that the Unit Levels Simulation's own user interface should probably be extended with data from the intercell database, since the DIS system will contain objects unknown to the Unit Level Simulation but which the user will want to interact with. The Cell Adapter Unit is responsible for handling these additional objects.

In this architecture, the distributed Cell Adapter Units across the DIS system will contain knowledge of all objects interacting on the DIS. The intercell databases will eventually be the repository of most knowledge about the internals of the non-standard DIS cells, and eventually it will become easier to modify and extend the Cell Adapter Units than the original contents of the non-standard cells. A common set of tools and techniques must be developed for the CAUs if seamless integration of extant unit level models is to be practical.



6.5.2 Unit /Platform Translation

Interaction between units and platforms will be handled by the Cell Adapter Unit and must abide by two conditions. First the CAU must translate the DIS environment into one of units for the Unit Level Simulation and translate the Unit Level Simulation environment into one of platforms for the DIS. Second, the

interaction between the objects must follow the DIS paradigm of each object maintaining its own state and broadcasting changes to the net.

The CAU associated with the unit level simulation is responsible for deaggregating the units into platforms and maintaining the platform level of simulation in its own processors, using its own databases (see Figure 7-6). This is because the own-CAU is the only place that has the necessary information required for deaggregation into platforms. It therefore requires considerable knowledge of the internals of the unit level simulation to be placed in the intercell and local databases. It is arbitrary whether or not the CAU continuously maintains all platforms, or only generates platforms which are interacting with other objects on the DIS network (see Figure 7-7). A trade-off is required between the processing power required to deaggregate versus maintain all platforms. This latter is often called variable granularity simulation, and has been proposed as a technique for generating the appearance of massive numbers of platforms at lower processing costs.

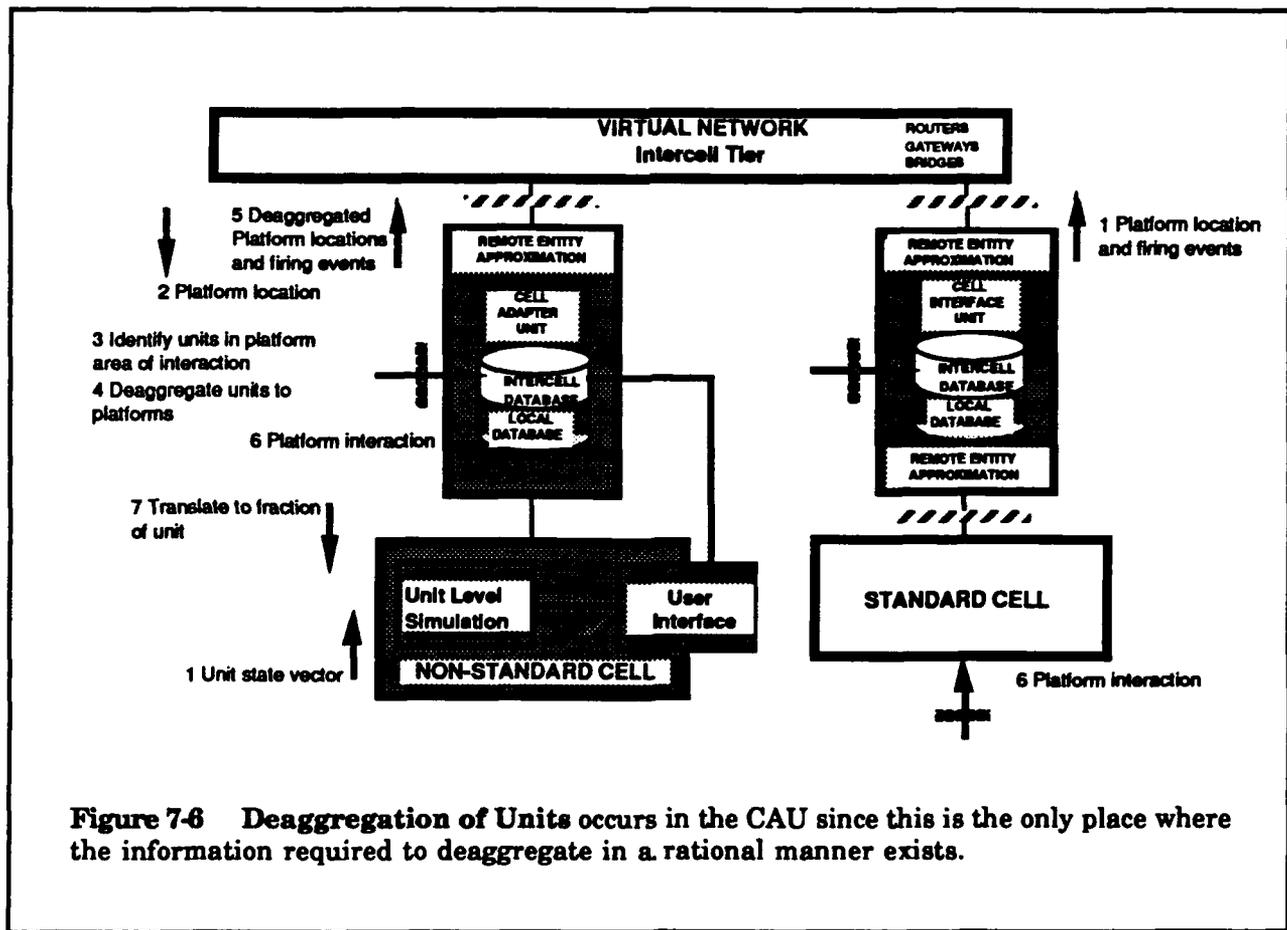


Figure 7-6 Deaggregation of Units occurs in the CAU since this is the only place where the information required to deaggregate in a rational manner exists.

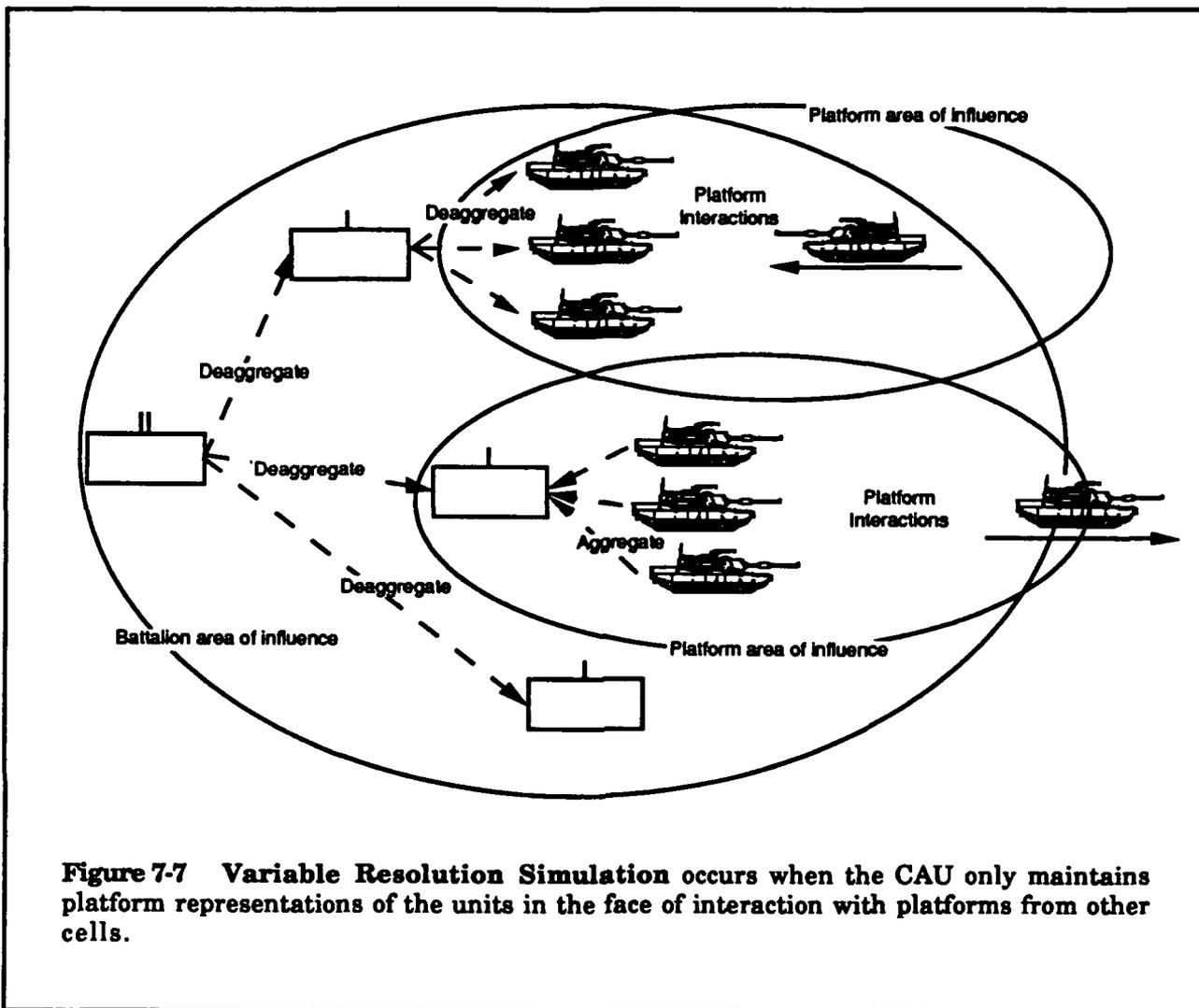


Figure 7-7 Variable Resolution Simulation occurs when the CAU only maintains platform representations of the units in the face of interaction with platforms from other cells.

Note that interaction at some intermediate level of unit aggregation (ie as might be assumed if two unit level simulations at different levels of aggregation were interacting with no platform level simulations involved) is currently assumed not to take place in DIS, although the architecture does not forbid this. The DIS Architecture will support unit levels of interaction (e.g. a simulation at battalion aggregation interacting with one at company aggregation via platoons) if the DIS message standard is extended to handle control messages informing the network what levels of aggregation are expected by the other CAUs since only the own-CAU may deaggregate the unit (see Figure 7-8).

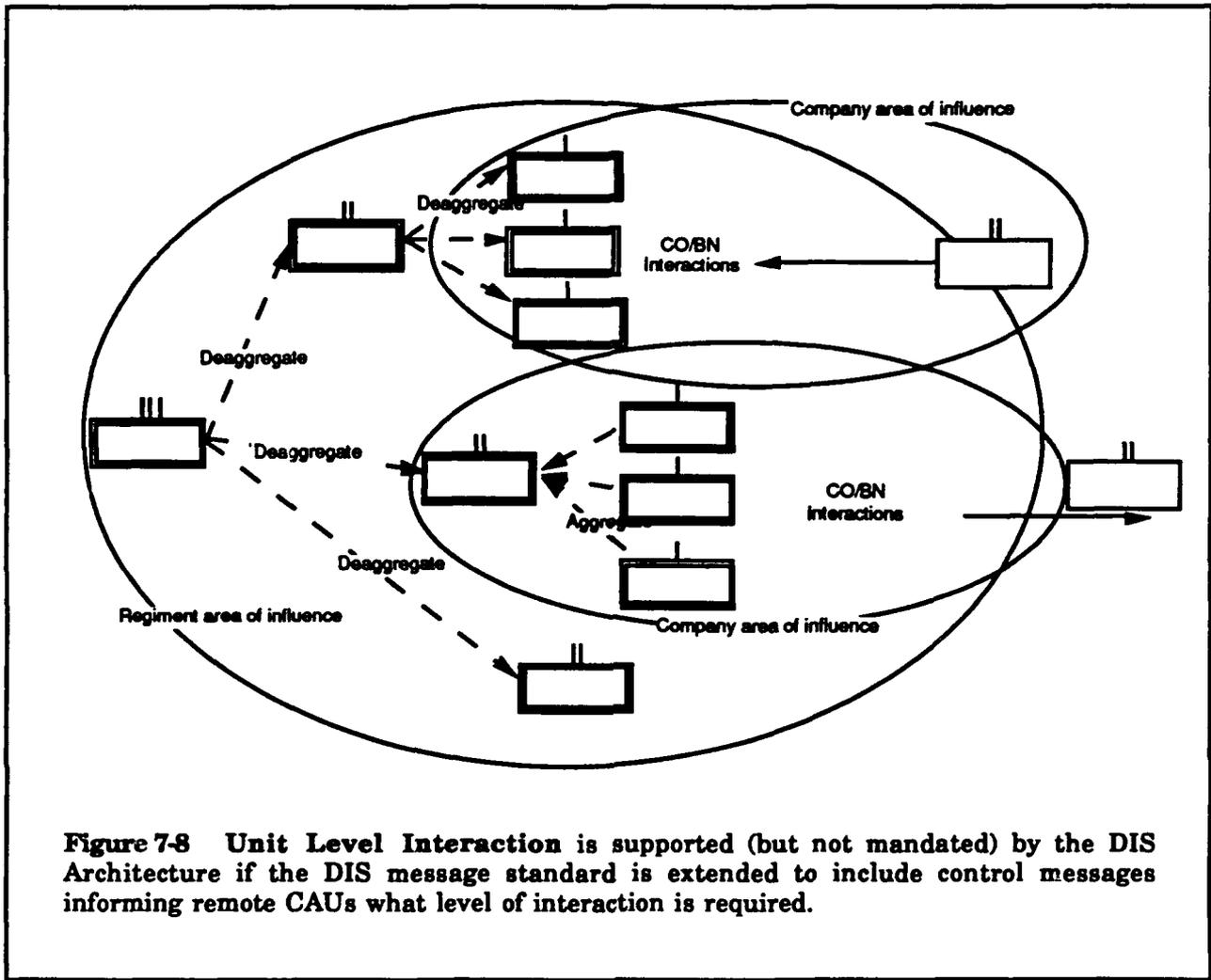


Figure 7-8 Unit Level Interaction is supported (but not mandated) by the DIS Architecture if the DIS message standard is extended to include control messages informing remote CAUs what level of interaction is required.

6.5.3 DeAggregation

One mechanism for handling deaggregation into platforms within the CAU is to embed in the CAU computer generated forces software. The role of the CGF is to provide platform level execution of unit level missions input by human operators. If the C3I structure of the unit level simulation can be used to feed CGF code, then the CGF can be used as a CAU (see Figure 7-9).

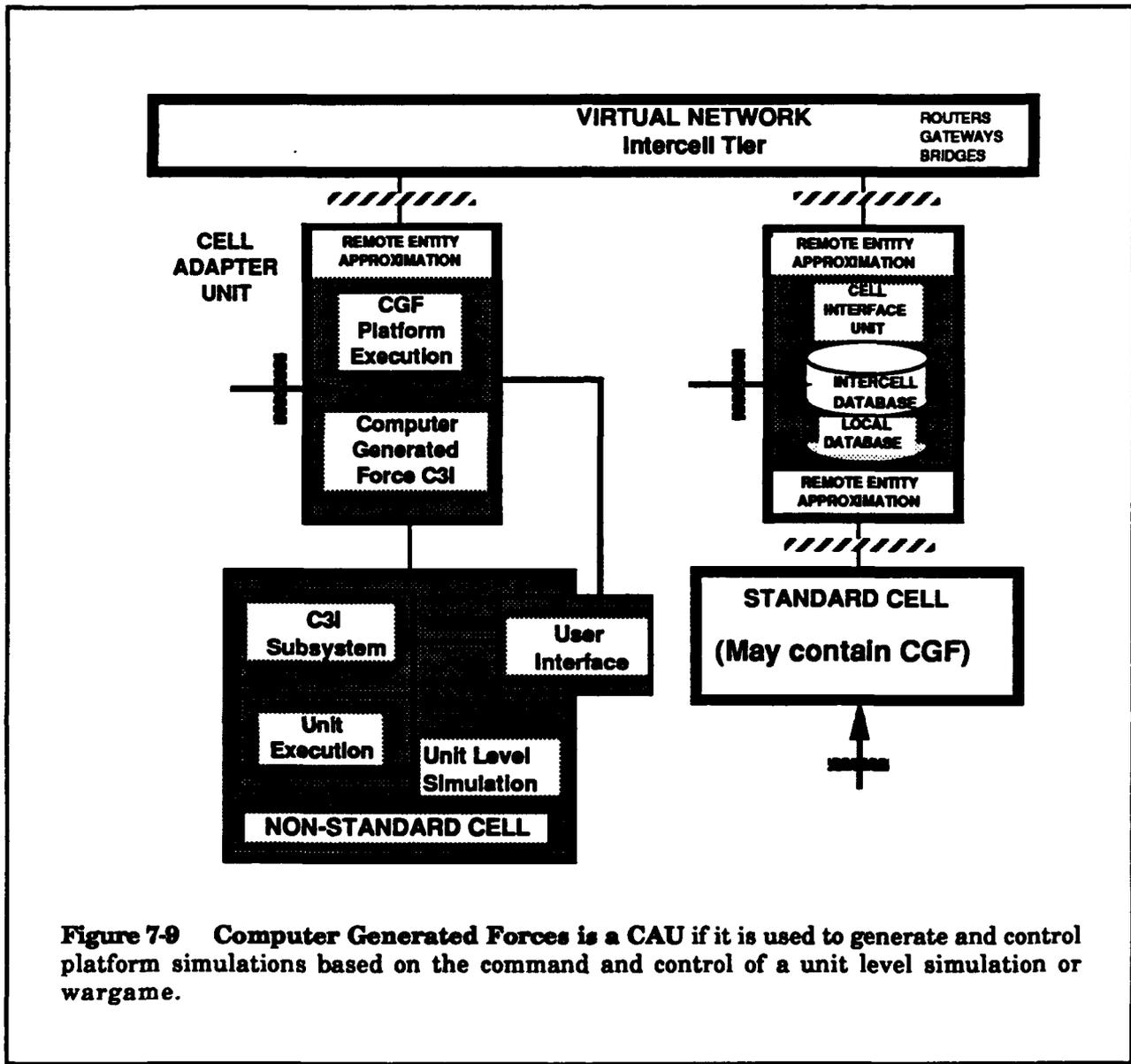


Figure 7-9 Computer Generated Forces is a CAU if it is used to generate and control platform simulations based on the command and control of a unit level simulation or wargame.

6.5.4 Impact on the DIS Message Protocol

Integrating unit level simulations will require an extension to the DIS message standard to take account of control messages between CAUs. These messages are those that deal with informing the network of the required level of deaggregation (for unit levels of interaction). If platform levels of interaction are the only levels permitted, then these messages are not necessary.

6.6 Live Ranges and Operational Platforms

6.6.1 Basic Interoperability Issues

This section discusses some of the critical issues involved in the interoperability of live ranges with simulation devices from the Distributed Interactive Simulation (DIS) community. Example scenarios are used to focus the discussion. Seamless simulation with DIS may provide a solution to four operational objectives of live ranges:

- Use simulation assets to provide range participants with the infrastructure of a larger battlefield than that possible using field equipment at the range.
- Use simulation assets to provide range participants with nuclear strike calculations and effects (similarly for chemical or any area weapon).
- Use simulation assets to provide range participants with indirect fire calculations and effects.
- Use simulation assets to provide range participants direct interaction with specialized platforms not available at the range.

In order to provide seamless simulation of live ranges with DIS, six assumptions are made concerning the operation of the range:

- Range platforms are networked in the range system to a Range Control Center (RCC).
- Range platforms transmit location and velocity information whenever sudden changes in those parameters occur.
- The platforms transmit own platform firing events whenever they occur.
- Latency from man machine interface to man machine interface within the range or to man machine interface within the DIS is within 300ms.
- The RCC handles hit calculations and informs the platforms accordingly.
- An accurate terrain database of the range is available to the simulation assets.

These assumptions mean that the DIS simulation assets can use DIS Dead Reckoning Algorithms (DRA) to model the range platform movement smoothly within their own image generation systems. An accurate terrain database of the

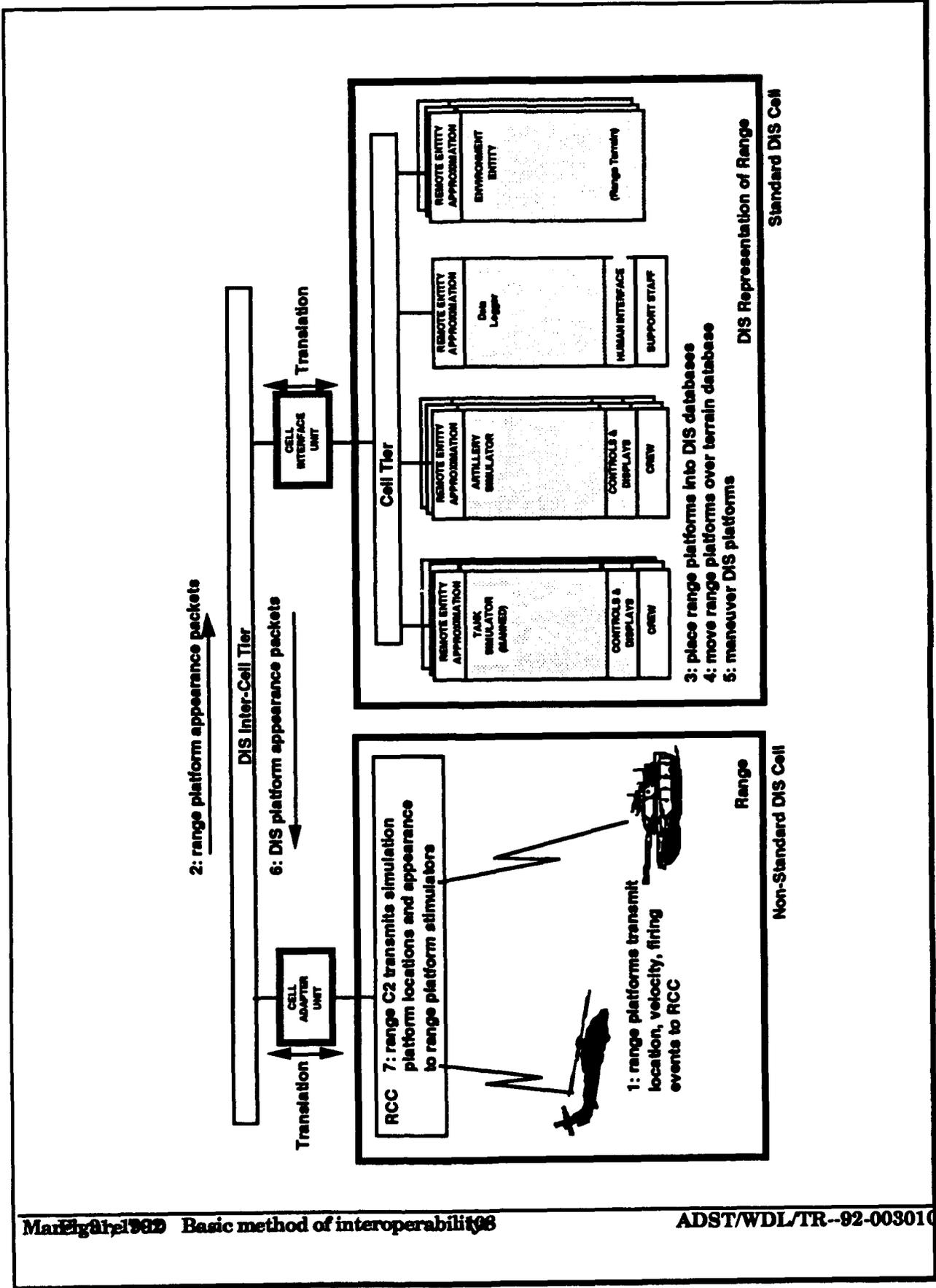
range is necessary to provide an adequate degree of correlation between the simulation entity and its representation in the range RCC database.

The DIS standard itself is based on a number of assumptions. Any networked system that proposes to use this standard must examine the DIS assumptions to determine if they are valid for that system.

- DIS is not bandwidth constrained.
- DIS objects are constrained in their I/O.
- The DIS network is extremely reliable, message loss is virtually zero.
- DIS platforms use a dead reckoning model of themselves to determine when to transmit an appearance packet. Transmission rates vary from one every five seconds for a static uneventful platform to a maximum of 15Hz for moving platforms (current SIMNET, may increase to 60Hz for fast movers). Transmission is reliable.
- Latency from one platform user interface to another platform user interface is less than 300ms

If a networked system is bandwidth constrained, or if the objects are unable to transmit appearance data at a rate that permits credible dead reckoning, then another message standard is advised. Since bandwidth between RCCs and range platforms is constrained, and the message data rates low and unreliable, the DIS protocol standard is not optimum and probably inappropriate for platform/RCC communications within ranges..

Range platforms transmit their location, velocity and firing events to the RCC (see Figure 7-10). The RCC transmits range platform appearance packets to the DIS Intercell Tier. The DIS asset reads these packets and creates shadow simulation platforms corresponding to the range platforms, and places them on the simulation terrain database. The DIS asset uses DIS dead reckoning algorithms to move the shadow platforms on the terrain database, and uses new updates from the range to correct locations. The DIS asset handles its own platforms, and transmits their appearance packets over the Intercell Tier. The RCC is responsible for transmitting to the range platforms the locations and appearances of the simulation platforms, for inserting these images into the sensors of the range platforms, and for informing the range platforms of damage caused to them by the simulation platforms. The Cell Adapter Unit (CAU) is responsible for translation between the range internal data and database representations and the DIS standard data protocol and databases.



6.6.2 Large Battlefield Infrastructure

The DIS simulation can support the RCC by embedding the range units in a larger battlefield consisting of simulated man in the loop units. For example (see Figure 7-11), the range unit requests support in the form of assets from higher command not physically available at range. The RCC play the command and control (since they are in charge of training), and decide which assets will be made available and when. These assets are simulated in the DIS system, and must fight their way across the larger simulated battlefield in order to support the range unit. They are attrited en route (by computer generated forces or by other manned simulator platforms in DIS), and arrive with realistic and validatable lateness and reduced force.

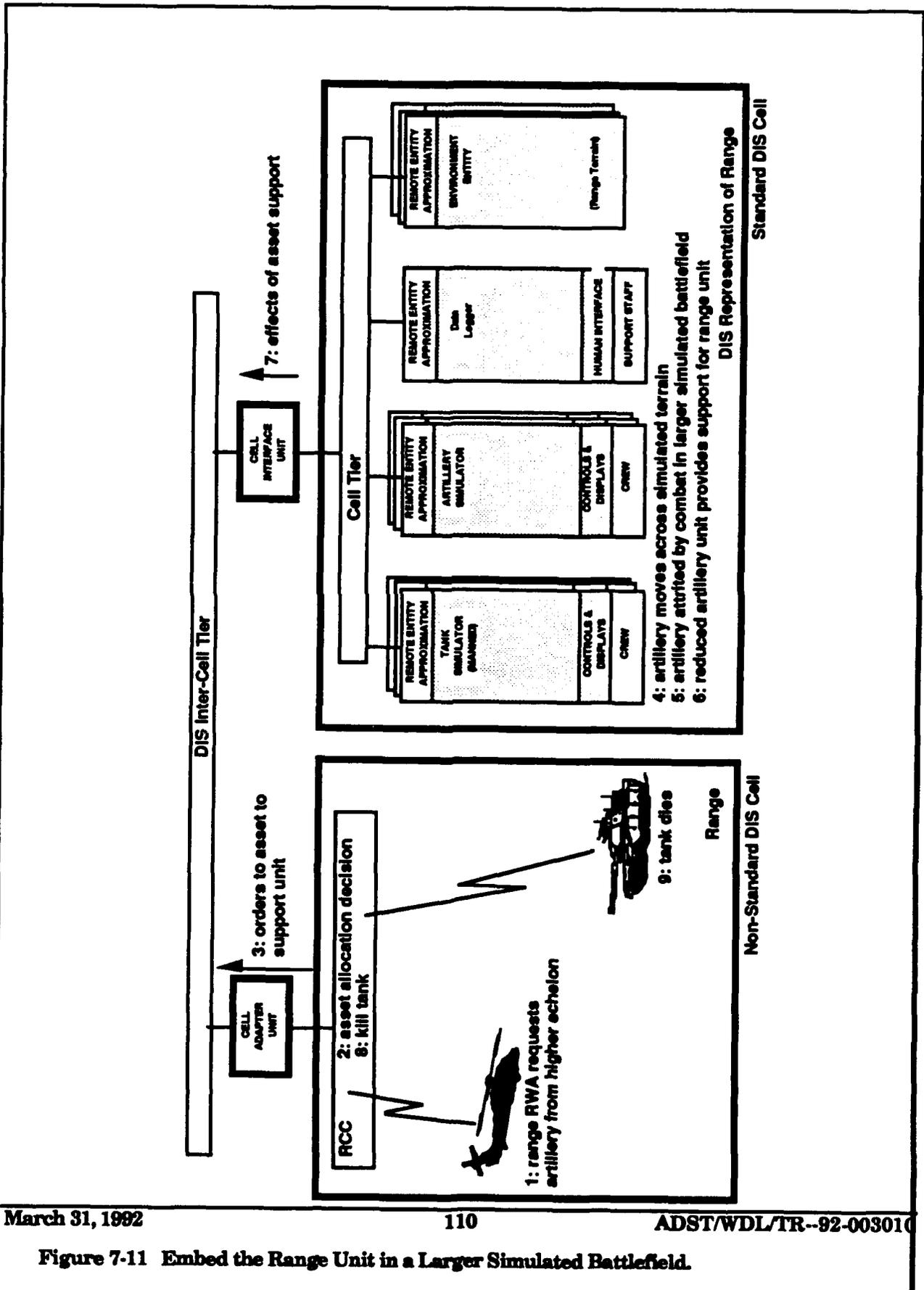
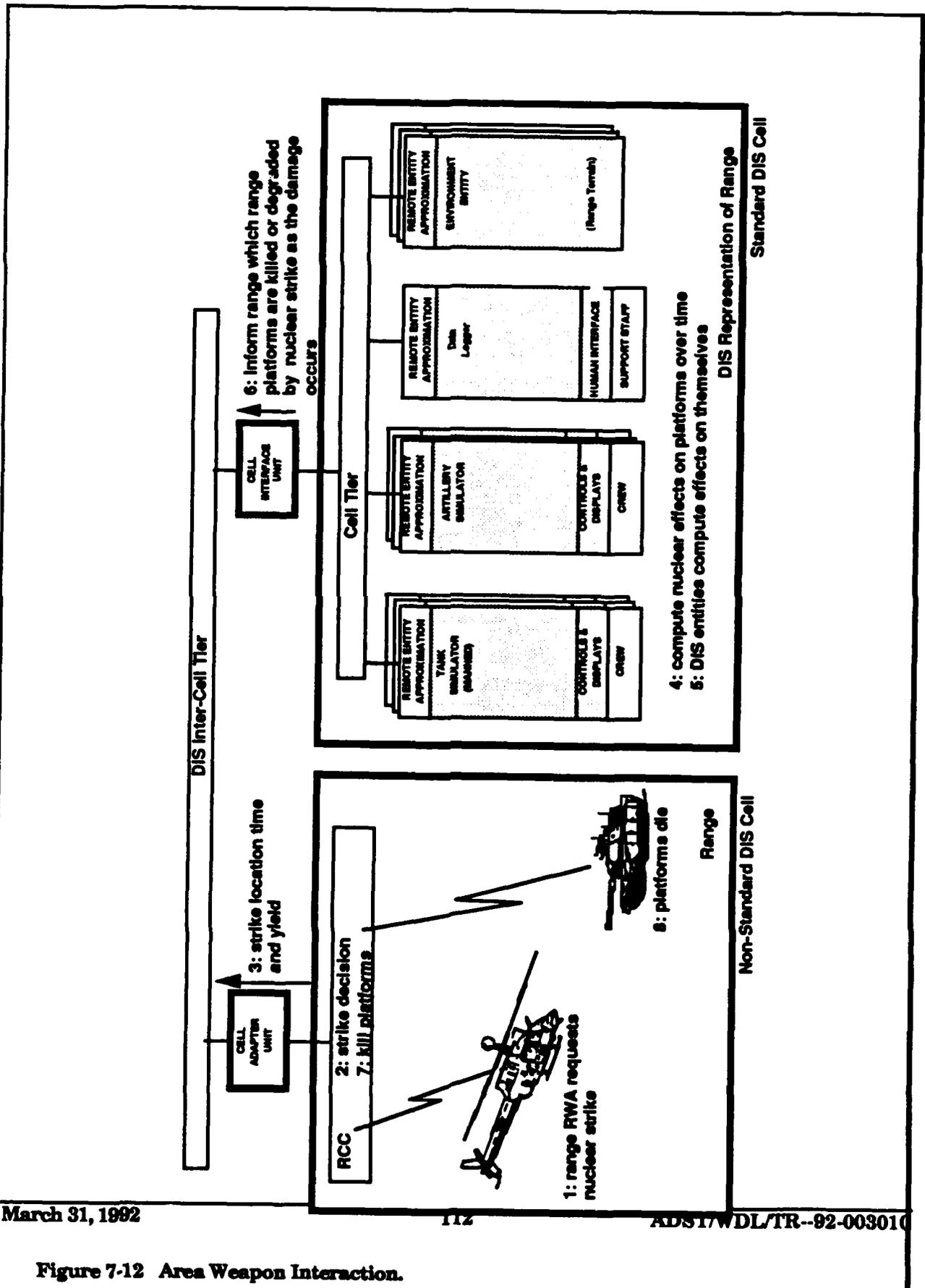


Figure 7-11 Embed the Range Unit in a Larger Simulated Battlefield.

6.6.3 Area Weapon Interaction

An example of area weapon interaction is shown in Figure 7-12. A decision is made on the range to request a nuclear strike, and this decision passed back to the RCC to simulate the processing of that request. If a strike decision is made, then the parameters of the strike are transmitted over the DIS Intercell Tier to the DIS asset. The DIS environment entity computes the effects of the blast on all platforms, and the effects of radiation (modified by atmosphere and terrain) on all platforms over time. These effects on range platforms are transmitted over the DIS Intercell Tier to the range as they occur (since the effects will vary over time as the platforms move in and out of the moving radiation cloud for example). The RCC system transmits kill or partial damage information to the range platforms.



March 31, 1992

Figure 7-12 Area Weapon Interaction.

6.6.4 Indirect Fire Effects

Indirect fire (artillery) is handled similarly to area weapon effects, with the difference that indirect fire effects are immediate and localized.

6.6.5 Specialized Platforms

If the DIS system is to provide the appearance on the range battlefield of platform or event types not available to the range, then the RCC and the DIS databases must agree on the specifications and the physical effects of the simulated systems.

6.6.6 Major Technical Challenges

Two major technical challenges are of interest in dealing with interoperability of ranges and DIS:

- Superimposing images in the range sensor systems. The DIS system platforms must have their appearances and behavior superimposed on the fields of view of the relevant range sensor systems. Direct view visual interaction between actual equipment on the range and DIS objects is not feasible with available technology. While it is theoretically possible to superimpose computer generated images on the viewports of actual vehicles, such image generators with sufficient levels of realism are currently too complex, bulky, power-hungry, and expensive to operate on field equipment.
- Database correlation. In particular, it is critical that the terrain database available to the DIS system be closely correlated to the actual range terrain. This is required so that the images of the DIS platforms may be clamped to the real terrain surface (if appropriate). Otherwise platforms will appear to float above or burrow below the terrain surface in each others system.

Increasingly, modern war is fought at night, at long ranges, and using sensors. Under these circumstances terrain is not viewed or detected with the extreme detail available to the human eye at close ranges. Also, air platforms do not need to have their images clamped to the terrain surface and often engage at beyond visual ranges. If properly designed, it will be possible to inject images into sensor systems with credible realism. With these restrictions in mind, the following scenario indicates the kind of interactions possible using current technology.

6.6.7 An Example Scenario

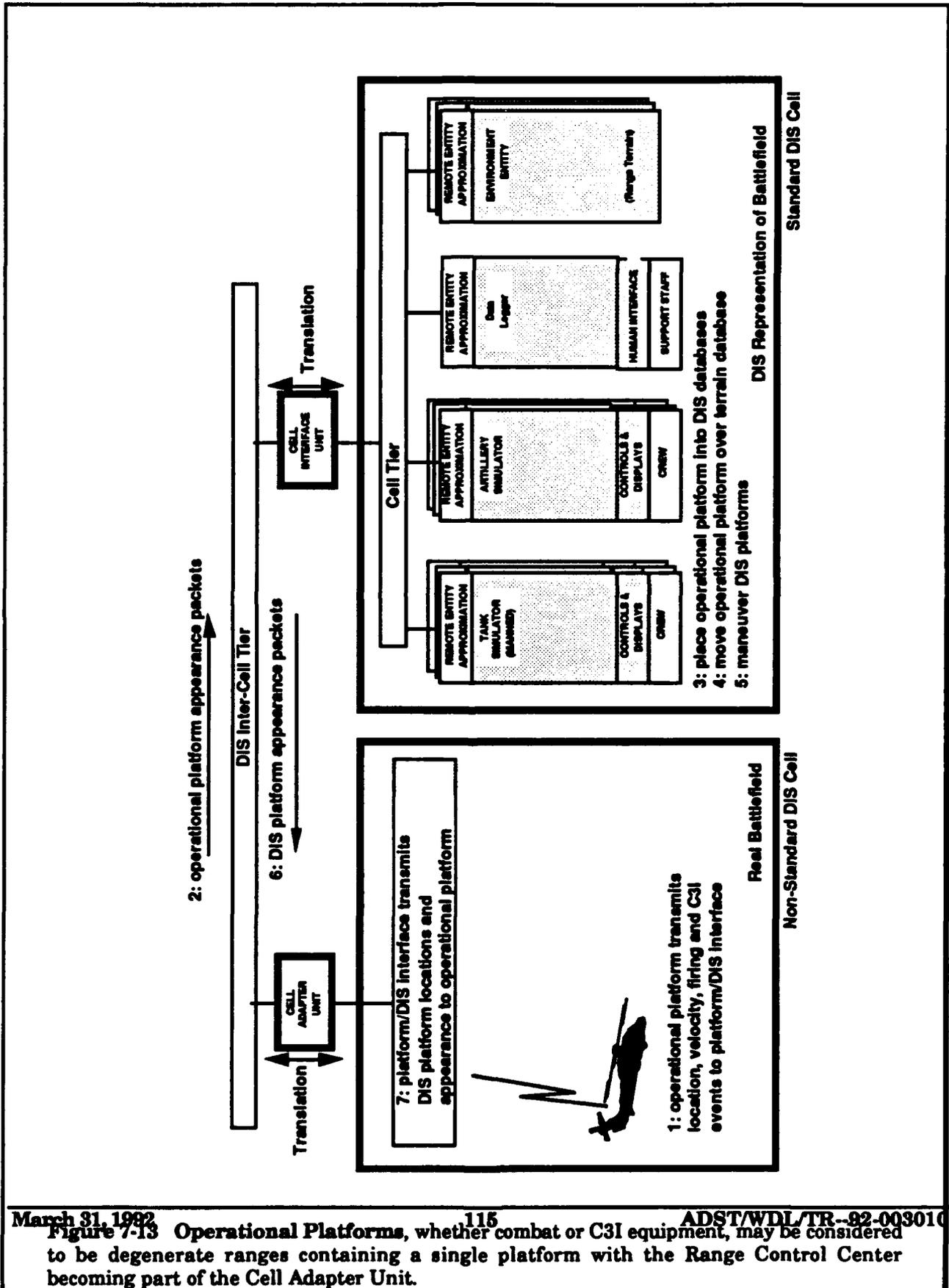
This example suggests some of the possible operational interactions between DIS and range. It assumes that many of the technical challenges described above have been solved, or at least handled in such a way that the operational objectives of the range and DIS participants are met.

A Mechanized CO is operating on range at dusk. They call for a Aviation Scout mission to attempt to locate enemy defensive positions. Via DIS, a scout helicopter mission is defined and dispatched at the BDS-D site at Ft. Rucker. The mission flies over the simulated range coordinates (by this time it is dark and the helicopters use thermal and radar for navigation), and locates the enemy locations (which appear on the DIS database via DIS messages from the Range Control Center. The helicopters could only have been seen by the range enemy actors if the helicopter images are injected into the thermal viewers available to the range enemy actors.). The scout mission reports the position to the Mech Company, who calls for an artillery strike which is handled by a Fire Control Center located at Ft. Sill. The DIS artillery simulation determines the times and locations of the artillery rounds hitting the ground, and the damage to the enemy locations. The damage and time of damage is sent via the network to the Range Control Center which then "kills" the relevant enemy locations. The Mech CO then carries out a night attack on the damaged enemy locations.

While direct visual interaction between live range platforms and DIS system platforms is currently out of the question, a great deal of valuable interaction can occur via sensor viewing. This requires the injection and manipulation of images into the sensor systems, which is an advanced research topic for LORAL. Warfare is increasingly carried out using sensors, and this will make the problem of range and DIS interaction both easier and more valuable.

6.6.8 Operational Platforms

Operational platforms are a degenerate case of live ranges, in which the range contains only a single platform (the operational platform under consideration). The operational platform is considered to be a non-standard DIS cell. The Cell Adapter Unit contains the RCC, which may or may not be loaded on the platform (see Figure 7-13).



March 31, 1982

Figure 7-13 **Operational Platforms**, whether combat or C3I equipment, may be considered to be degenerate ranges containing a single platform with the Range Control Center becoming part of the Cell Adapter Unit.

6.6.9 Impact on the DIS Standards

Internal range message protocols must be developed that provide sufficient data, data rates and data reliability to interact with DIS. If the range system fails to track a platform for more than a few seconds then its appearance in the DIS system will be anomalous. Arbitrarily large jerks in movement may be observed depending on the reliability of the range tracking system.

Trade-offs between types of interaction, required data rates and reliabilities, and required range equipments must be identified and categorized.

The cell adapter unit for the range must be built to handle translation between the DIS PDU standard and the internal range protocol.

The range and DIS must share databases that describe the entities in both the range and DIS system, and their possible modes of interaction.

The range system must be able to track which simulated platform is being fired at by which range platform.

Insertion of simulated DIS events and platforms into the range platform sensors must be possible.

Real range terrain must be correlated with the DIS terrain databases of the range.

6.7 Conclusions

Seamless Simulation of arbitrary heterogeneous systems into DIS is a requirement if prior investments in system building, VV&A and analysis are to be exploited. The role of the DIS architecture is to support the seamless integration of these systems, providing architectural constructs for the location of seamless integration techniques and solutions. The Cell Adapter Unit, with its cell and local databases, and the encapsulation of heterogeneous systems into cells, provide a mechanism for integration. The paradigm of each object maintaining its own state and broadcasting changes can be maintained under this architecture. However, the DIS message standard will need to be extended to take account of control messages between CAUs coordinating the seamless interactions, to take account of simulated face to face and tactical communications between software representations of staff and commanders, and to take account of different families of network (such as those contained in ranges).

7. Special Problems

7.1 Dynamic Terrain

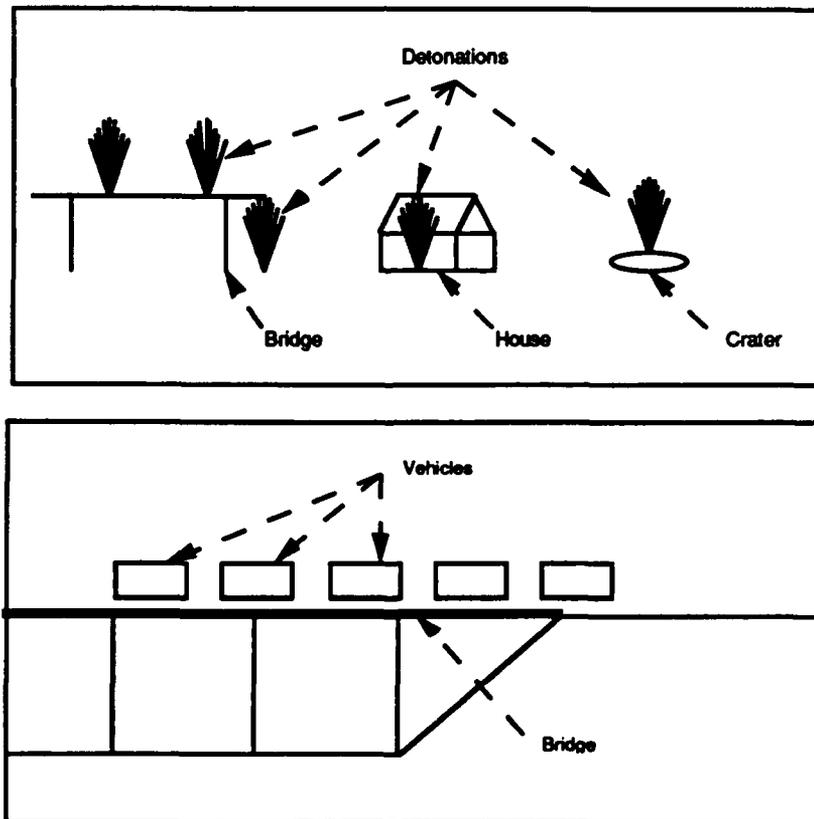
7.1.1 What is Dynamic Terrain?

In the short history of Distributed Interactive Simulation, terrain databases have been static. This precludes combat engineering operations and fire damage on database objects (e.g., craters, damaged buildings, and blown bridges).

"Dynamic Terrain" would remove those limitations, and provide a more realistic training environment.

7.1.2 Scope

Figures 7-1 and 7-2 depict examples we will use to highlight the technical problems associated with dynamic terrain. The first figure shows detonations, and some objects they can damage. The second figure shows a bridge that will collapse when enough weight is applied to it.



Figures 7-1,2: Dynamic Terrain Effects

In this section, we'll explain the need for dynamic terrain, the types of dynamic terrain, and the relationship of dynamic terrain to space-time coherence and interoperability.

The problem of dynamic terrain can be broken down into three subproblems.

1. How do we specify those portions of the terrain that can be affected by the evolution of the simulated battle? Further, how do we communicate this specification to all scenario nodes so that all Virtual Battlefield entities interact with the dynamic terrain in a globally-consistent manner?
2. How do we describe the changes that occur to dynamic terrain components during the course of the battlefield evolution?
3. How do we communicate the component changes to all Virtual Battlefield entities so that all entities share a coherent view of battlefield terrain?

An important benefit results from the object-oriented design of the DIS architecture; since terrain, with all its aspects (geographical, geometrical, cultural, emissivity) is an Actual Battlefield object, it can correspondingly be rendered in the architecture as a Virtual Battlefield object (entity). This is entirely consistent with the way warfighters view terrain--it possesses behaviors and state that must be reckoned with. Accounting for terrain in this fashion puts it on an equal footing with all other battlefield entities, and allows DIS entities to interact with terrain via the architectural communication paradigm (PDUs). This is a roundabout way of saying that the only obvious solution for coherent maintenance of dynamic terrain, the terrain server, is clearly and fully supported by the DIS architecture. The terrain server will inherit the characteristics of a battlefield entity.

7.1.2.1 Specification

The details of dynamic terrain specification are yet to be reached. Where those specifications will reside however, has been identified by the architecture. Those specifications, once developed, will reside in the Battlefield database, and as such, will be available for distribution to all participating simulation nodes, and the battlefield entities they represent. This will ensure terrain correlation and hence space coherence of the exercise.

Description

Change description involves two aspects. The first is identification and classification of the ways that a terrain object can change--i.e. enumeration of all the forms it can assume. For a cultural feature such as a building, it can be whole (undamaged), or can be damaged in a number of ways: blown-up, run-over, on fire, burned, etc. Following this enumeration of possible object states,

comes the specification of how those states appear, and how they differ from each other.

An interoperability problem that crops up at this point is that some CIGs may not be able to depict all possible enumerated forms, or the depictions may vary from CIG to CIG to such an extent that the differences affect the Virtual Battlefield and hence simulation outcome.

Communication

One must define PDUs which are analogous to the entity state packet and which can be used by the dynamic terrain entity to register its changes on the Virtual Battlefield. On the surface, the terrain entity will be very similar in operation to vehicle entities. Both types will receive PDUs, compute the effects which bear on themselves, and issue PDUs in response that relate their current status. However, a fundamental difference will exist between the two classes. The terrain entity will mark its evolution through time by discrete events while vehicle evolution (movement) will be marked by a continuously changing progression.

The problem here is missed packet updates. If entity A fails to receive an entity state PDU from vehicle entity B, the damage is somewhat contained in that entity A: has been aware of the existence of B; has been dead reckoning entity B's position and therefore knows B's position to within some (now possibly large) error tolerance; has the chance to receive the next entity state PDU from B when the next broadcast becomes necessary. This containment has been previously referred to as the "self-healing" nature of the protocol. With the terrain object however, no such self-healing exists. Dynamic terrain entity changes are discrete events triggered by dramatic battlefield occurrences: collapsed bridge, blown-up building, explosion-caused crater. If other entities fail to recognize these occurrences, then their perception of the Virtual Battlefield will dramatically differ from the perception of the other, recognizing entities.

One solution is to adopt a reliable transmission protocol for dynamic terrain messages. This solution, however, imposes added burden on the communication network and stands as a glaring exception to the message-transmission paradigm adopted for communication between other entities.

Another solution is to impose a minimum broadcast horizon time on the dynamic terrain entity. Such a mechanism already exists for vehicle entities. A time interval is selected which is appropriate for the vehicle. If the interval expires, and there has been no entity state PDUs issued by the vehicle during that time (either because no significant movement has occurred during the interval, or because the DR model position is still valid), then an updating broadcast is forced. This mechanism ensures that active entities maintain a presence on the network. However, some problems exist in implementing this solution. To begin with, an entity state PDU contains all the necessary information to correctly render the vehicle in both appearance and position. Only one broadcast is

required to repair any breach in time/space coherence. The dynamic terrain PDU must be so self-contained. However, the terrain entity may have a lot of information to report. Essentially the terrain PDU should describe the current changed state of all terrain components which have been dynamically affected on the Virtual Battlefield and which differ from their static state as described in the Battlefield database.

So this solution brings us to face another problem, inventing a concise descriptive notation for the dynamic terrain, and packing this notation into a PDU.

However, this solution finds support in the DIS architecture. Much work needs to be accomplished to implement this solution, but the details fit within the architectural scheme.

7.1.3 How Dynamic Terrain Relates to Space-Time Coherence and Interoperability

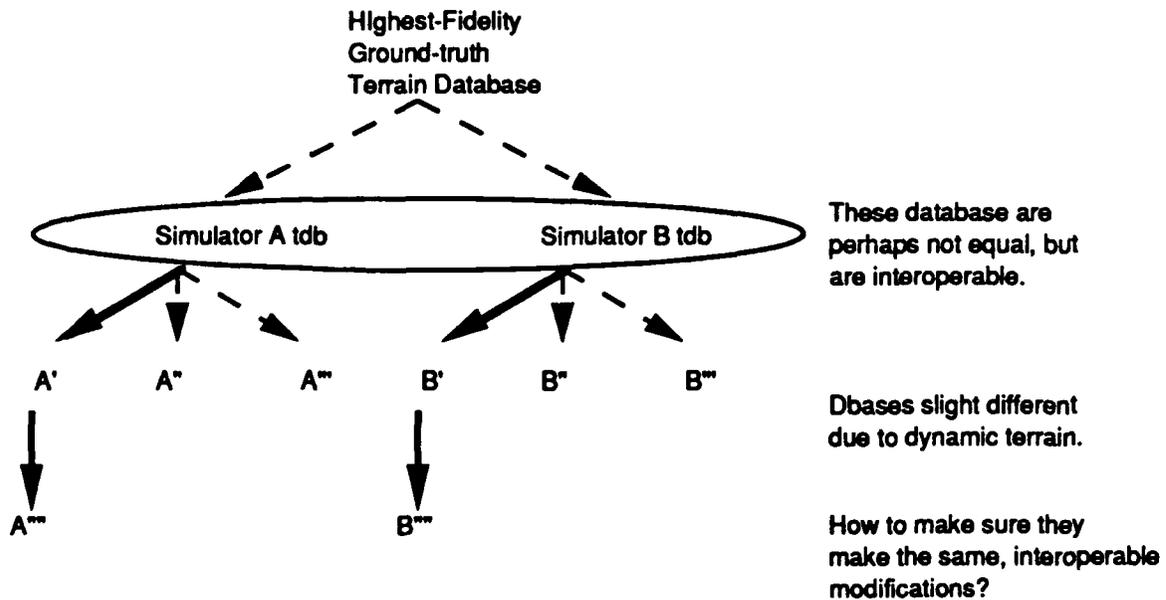


Figure 7-3: Heterogeneous Simulators

Dynamic terrain involves successive changes to a given set of databases occurring over the course of an exercise/session. Figure 7-3 shows two heterogeneous but interoperable simulators with different but interoperable terrain databases. Dynamic terrain is an incremental change to the databases. In order for interoperability to be maintained during the exercise/session, each simulator will have to make incremental changes which preserve interoperability (e.g., A goes to A', and B goes to B', with A' and B' being "equal" enough to be interoperable).

7.1.4 Technical Challenges

In this section, we'll examine some of the technical challenges presented by dynamic terrain, with emphasis on those of space-time coherence.

7.1.4.1 Balancing Antagonistic Real-time, Concurrency, and Serializability

We would like to guarantee that all observers see equivalent dynamic terrain changes. We would also like them to see them as soon as they happen. This section explains why these goals are antagonistic.

Real-time

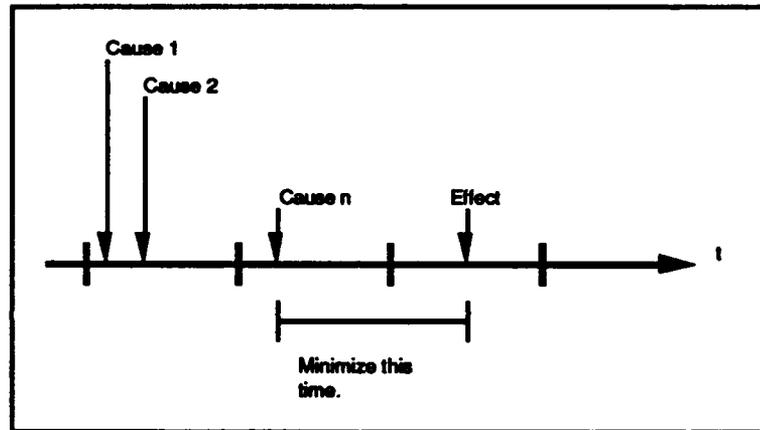


Figure 7-4: Dynamic Terrain Timeline

Figure 7-4 shows a timeline of causes which combine to produce a given effect on the terrain. We wish to minimize the time and distance between the causes and effects, to avoid introducing visual anomalies to the observer (e.g., a long time between detonations and crater formation). As we have seen in previous sections (*ref space-time coherence*), these latencies are due to internal processing and external network latencies.

