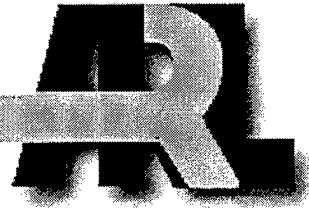


ARMY RESEARCH LABORATORY



The Distributed Interactive Simulation
(DIS) Lethality Communication Server
Volume I: Overview

Geoffrey C. Sauerborn

ARL-TR-1775

FEBRUARY 1999

19990324 005

DTIC QUALITY INSPECTED 1

Approved for public release; distribution is unlimited.

VR Link® is a registered trademark of MäK.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Abstract

A new approach to handling battle simulation lethality is presented. In this approach, a single vulnerability/lethality server provides standard distributed interactive simulation (DIS) damage states to entities fast enough for most real-time applications. Benefits include DIS simulations being freed from the burden of maintaining damage state tables and DIS exercises being easier to configure as a whole. A DIS software implementation is freely available by contacting the author.

TABLE OF CONTENTS

	<u>Page</u>
LIST OF FIGURES	v
LIST OF TABLES	vii
1. PURPOSE	1
2. INTRODUCTION	1
3. OBJECTIVE	1
4. LIMITATIONS	2
4.1 Limitation on Environmental Awareness	2
4.2 Data Latency	2
5. DESIGN	3
5.1 Callable Subroutine Inside the Simulation	4
5.2 Networked Server, With DIS Traffic Monitored	4
6. FEATURES AND EXPANDABILITY	6
6.1 Criteria for Determining Which Data are Delivered	6
6.2 Criteria for Determining Lethal Parameters	7
6.3 Returned Results	8
6.4 Beyond DIS	10
7. SUMMARY	11
REFERENCES	13
APPENDIX	
A. DIS Lethality Server Architecture	15
DISTRIBUTION LIST	21
REPORT DOCUMENTATION PAGE	25

INTENTIONALLY LEFT BLANK

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1.	DIS Server Response Latency	3
2.	Networked Server, DIS Traffic Not Monitored	5
3.	Networked Server Topology	5
4.	Non-Network Server Topology	6
5.	Default Results	9

INTENTIONALLY LEFT BLANK

LIST OF TABLES

<u>Table</u>	<u>Page</u>
1. MFK "Probability" Space	9

INTENTIONALLY LEFT BLANK

THE DISTRIBUTED INTERACTIVE SIMULATION (DIS) LETHALITY COMMUNICATION SERVER VOLUME 1: OVERVIEW

1. PURPOSE

This report presents an overview of the distributed interactive simulation (DIS) lethality communication server (the server). The report's scope is focused on the server's purpose, overall design, and some particular features. However, this text cannot be used to effectively prepare, operate, or modify the server. (Refer to Volume 2 of this report for these practical considerations.)

2. INTRODUCTION

The ARL DIS lethality communication server is a combination of application program interface (API) libraries and utility programs that make it possible to allow multiple applications to access a single lethality data source. The server is designed for the DIS environment. As such, the server returns lethality results as described by (the DIS) Institute of Electrical and Electronics Engineers (IEEE) Standard 1278.1 [1]. Furthermore, the server expects input in DIS standard protocol data unit (PDU) format, although the equivalent input may be greatly condensed at more abstracted layers within the APIs. The DIS server has demonstrated a data latency of less than 1/100th of a second and thus may be useful for a wide variety of applications. This project was jointly sponsored by the U.S. Army Modeling & Simulation Office (as a 1997 Army Modeling Improvement Program [AMIP] project) and by the U.S. Army Research Laboratory (ARL).

3. OBJECTIVE

The motivation for the server stemmed from the author's experiences in conducting DIS exercises among different types of simulations. A frustrating problem was to discover "holes" in one or another simulation's lethality data set. Naturally, different simulations handled this situation differently. This resulted in "unfair" play between simulators (or occasionally, a fatal error in a simulator). Until the problem could be fixed (and the lethality holes "plugged"), the impact on the DIS exercise was usually catastrophic. Thus, the original objective of the server was to alleviate this situation and thereby increase the amount of "free play flexibility" and decrease the amount of overall pre-exercise preparation. After a little thought, it became evident that there are several serendipitous benefits to having a lethality server:

- The server could eliminate DIS interoperability variances in lethality results (remove “unfair” weapons effects play), since all DIS simulations will resolve lethality effects through a single, repeatable means.
- It would allow increased ease of verification, validation, and accreditation (VV&A) for battle simulations exercises as a result of having a standard (and centralized) set of unclassified lethality calculations.
- It could decrease DIS simulation development time by providing a completed, computer platform-generic, lethality/vulnerability-handling mechanism. That is, because the lethality issue has been decoupled from the rest of the simulation, the lethality-handling mechanisms may be “stubbed out,” allowing more time to devote to the rest of the simulation development process.

4. LIMITATIONS

Every automated system must have design trade-offs and performance limitations. This section mentions two of the most important.

4.1 Limitation on Environmental Awareness

There is currently no provision for making the server aware of the synthetic environment (terrain). The server’s knowledge of what and where things are is limited to the DIS network traffic. Therefore, only distance and orientation from the point of detonation are considered for area (indirect) detonations against a platform. External influences that may affect the outcome (e.g., walls, earth, etc.) are not considered since the lethality server will have no knowledge of them.

4.2 Data Latency

The DIS server should be acceptably fast for real time, training, and most other applications. Informal tests demonstrate the server’s response time to be faster than normal human reaction speed. Naturally, data latency depends on computer processing speeds, network bandwidth, and network traffic. The informal tests were conducted with an unoptimized¹ server compilation

¹ No special considerations were made to “tweak” the server or client programs; they were compiled without optimization. Furthermore, both the DIS server and the client application output verbose “debugging” information during this test. Therefore, these results could be viewed as a data latency upper boundary for these processors over a standard 10-mbps (millions of bits per second) Ethernet.

across a populated network during “normal business hours.” Figure 1 shows the distribution of time taken to send (and receive the answer to) 440 separate server queries. For these tests, the average data latency was about 6 microseconds (μsec), but a very few tests (about 0.7% of the queries) took much longer (between 20 and 30 μsec).

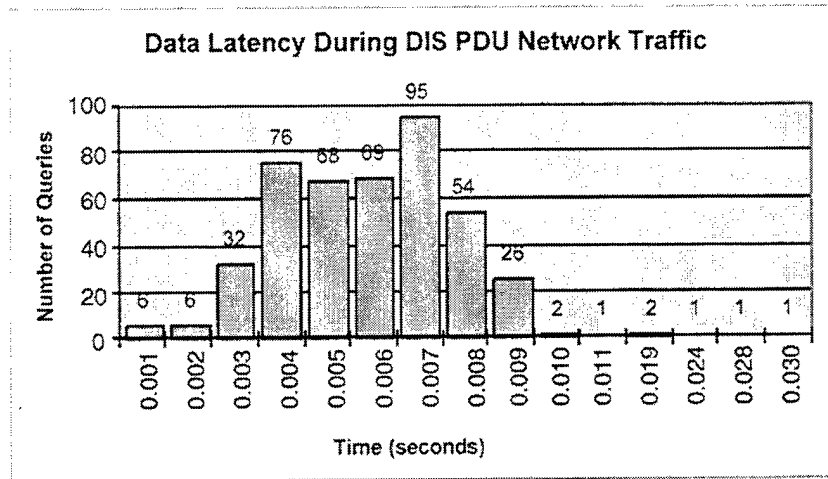


Figure 1. DIS Server Response Latency.

The DIS exercise included 79 DIS entities (67 tanks, 10 infantry fighting vehicles, and 2 mobile anti-aircraft platforms). Together, they generated 20,281 PDUs during a heavy 16-minute battle. This took place across a standard 10base2 (thin Ethernet) network that was not dedicated to the DIS server. That is, other data beside the server’s queries and the DIS PDUs also competed for network bandwidth. Approximately 98 other network hosts (other computers, printers, etc.) resided on the same network and produced normal uniform data protocol (UDP) and transmission control protocol/internet protocol (TCP/IP) traffic on the network, although an unknown number were probably inactive at the time. The DIS server (and the client application) resided on Silicon Graphics Incorporated (SGI) Indigo2 computers that used 150-MHz R4400 processors.

5. DESIGN

First and foremost, from an implementation point of view, the server is designed to be expandable. Many extensions can be accomplished by manipulating system parameters and initialization data. Other enhancements require additional software. For example, an unsupported

vulnerability data format may be accommodated by adding two functions to a software library. (More of these details are discussed in Section 6.)

At the most general point of view, the server is a system designed to provide “data,” based on some criterion. Currently, the criterion is focused on DIS battlefield “munition” type events. However, the server may be expanded to include other criteria. The data that the server provides are given somewhere in a local (to the server) file (or potentially at a remote site). The appropriate data are delivered to the client upon request.

Operationally, the server is designed to be flexible to meet the needs of client applications by allowing networked or non-networked use. In network mode, clients communicate with the server via (a standard and platform-independent protocol) TCP/IP sockets. Optionally, the server can be implemented as a callable subroutine (inside the client application) with no external networking traffic necessary. This allows flexibility for the client. The network connection will be slightly slower; however, its use shall ensure that all exercise participants using it are accessing the same lethality data sets. These two communication configurations are explained now.

5.1 Callable Subroutine Inside the Simulation (fastest communication speed)

In this schema (shown in Figure 4), data relating to the vulnerable platform and the threatening munition are a binary stream representing standard DIS PDUs. This information is passed directly to the server via an API function call. The server returns the lethality result (the damage state occurring against the threatened entity platform as a result of the munition detonation or other event).

5.2 Networked Server. With DIS Traffic Monitored

In this topology (shown in Figure 3), the server monitors DIS PDU traffic on the UDP DIS network. Communication with client occurs via TCP/IP sockets. The client requests the result from a detonation event by referencing the identifier (ID) for the detonation event and the ID of the threatened entity. These IDs are part of the DIS standard, and because the server is monitoring the DIS traffic, it knows which type of platform is referred to by the entity ID. The server also monitors other data that were broadcast on the DIS network and are relevant to a vulnerability calculation. The server then returns the result to the client via the TCP/IP network link.

Originally, a third topology was planned as shown in Figure 2.

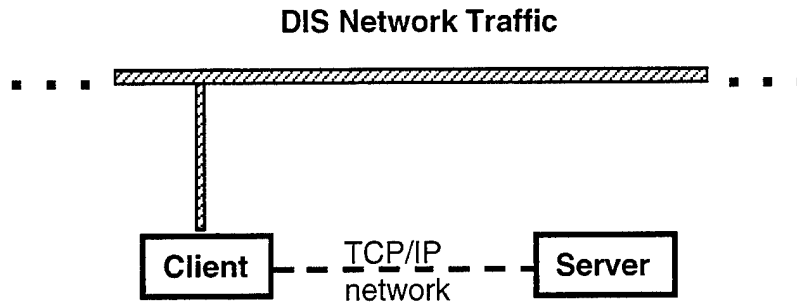


Figure 2. Networked Server, DIS Traffic Not Monitored (unimplemented).

This topology differs from Figure 3 in that the server would have no prior knowledge of detonations or entities. In this case, the client would be required to send more detailed information concerning the event. This topology was proposed as a building block before the topology of Figure 3 was achieved. However, once the server's capability to monitor DIS network traffic was completed, it made no sense to implement this topology for the reasons stated below.

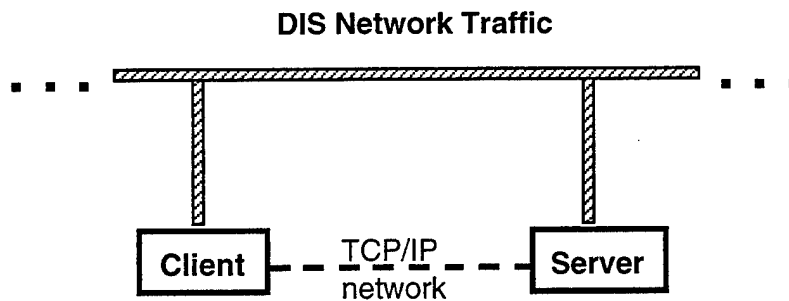


Figure 3. Networked Server Topology.

a. In the topology of Figure 4, the data required by the server would be delivered in the form of DIS PDUs. These very same PDUs were already broadcast on the DIS network. With the capability to monitor DIS PDUs completed (Figure 3), it is unnecessary for the client to resend them (since the server has been monitoring the network and already has these PDUs).

b. If the client wishes to simulate a DIS exercise by sending complete PDUs, this may be done by using the topology of Figure 4.

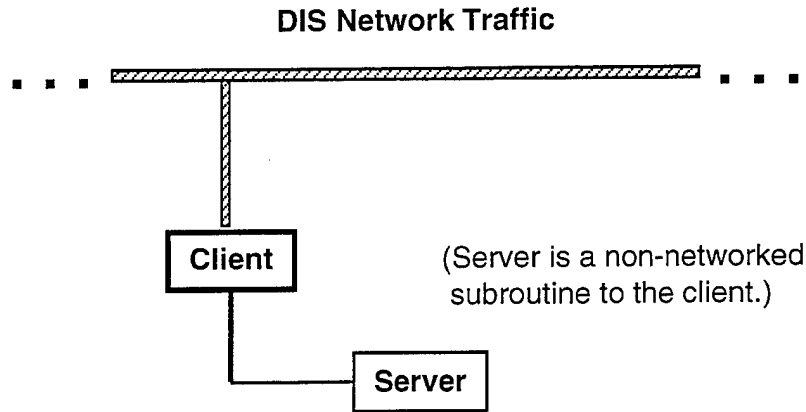


Figure 4. Non-Network Server Topology.

6. FEATURES AND EXPANDABILITY

This section highlights features and outlines the type of expandability that is built into the server's design at the time of this writing.

6.1 Criteria for Determining Which Data are Delivered

The server needs to know which type of data a client application wishes to retrieve. This is accomplished in two steps. The first step is to associate the data with a "threat" and "target." These data describe a target-threat interaction of a certain lethality "class." The second step involves monitoring the battlefield for the triggering event.

Currently, the server only handles one type of triggering event, namely, "munition"/ "detonation" type threats against an entity. There is no provision for damage resulting from collisions, biological/chemical exposure, or other kinds of damage mechanisms. However, the server is expandable and there is no reason why these (and other means of damage assessment) may not be added. However, this would require additional code to make the server "look for" and process the particular battlefield events that the lethality result depends upon (see Section 6.2).

It was mentioned that the target-threat interaction is a certain lethality "class." This class identifier allows the server to respond differently for different situations. As an illustration, suppose the data source for a ballistic munition divides lethality results according to dispersion at impact time. Without a lethality class, the server will return the lethality result from that data source even if the munition (was close but) missed the target. On the other hand, if the server knows that this munition-target interaction is of the "direct fire hit-to-kill" class, then it will

correctly bypass the data source and return a “no further damage” result (since the munition missed). This flexibility allows the server to be expanded to other lethality classes as they develop. There are currently only two lethality classes (“indirect fire” and “direct fire hit to kill”).

6.2 Criteria for Determining Lethal Parameters

Knowing how to behave for certain “lethality classes” is not enough. The server also needs to know upon what environmental factors and initial conditions the lethality calculation depends. The server accomplishes this by using the “lethality class” identifier to associate which parameters are important to that particular type of lethality result. Currently, the server only handles one type of triggering event—that of “munition”/“detonation” type threats, of which, direct and indirect fire munitions are sub-categories. These threats are characterized by an explosion or explosive/nonexplosive ballistic impact. The server knows which environmental parameters are necessary to complete the lethality calculation for these types of threats. Specifically, the server tracks the appropriate DIS PDUs and extracts the required environmental parameters from them.

In the DIS environment, the simulation that controls such a direct or indirect fire munition announces the munition’s explosion or impact by broadcasting a detonation PDU to all the exercise participants. Any munition that can be simulated in this manner can currently be serviced by the server. A provision of this is that the resulting damage to a threatened entity would then occur as a relatively immediate reaction to the detonation as opposed to one requiring prolonged exposure to a damage-inflicting environment (such as a dust storm, forest fire, noxious gas, etc.). This is because the lethality server does not account for damage as a function of time for these lethality classes. It merely returns lethality data based on an event (a denotation). These returned data are then used by the client in an appropriate manner.[2] The server does not continue to monitor the environment for “continued degradation” resulting from prolonged exposure. This type of behavior *could* be added by creating a new “lethality class.” Naturally, it would also require adding new behavior code to the server for that class.

One exception to the immediate lethality reaction just mentioned is that the server maintains a record of the number of munition “hits” occurring against any entity. While this information is available for clients to use as they see fit, it is not currently used by the server to assess damage. Furthermore, current APIs do not distinguish which types of munition accounted for the various hits; however, all this information is maintained by the DIS monitor module, and a summary output of all munitions and their effects is output in a report file (called `rollup.det`).

6.3 Returned Results

6.3.1 *Types of Vulnerability Methods (Results)*

When a lethality event occurs, many results are possible. Analysts usually divide the number of “end states” into more manageable sets. We refer to a set of possible outcomes as the vulnerability methodology. The only criterion is that the outcome sets describe the complete universe of possible outcomes. A simple example would be a binary methodology (“dead” or “alive” states). Most often, it is useful to have a more detailed division of outcomes. The server is designed to accommodate a variety of “kinds” of vulnerability methodologies.

6.3.1.1 One Implemented Vulnerability Methodology (the MFK class)

While the server is designed to accommodate a variety of vulnerability methods, as of this writing, only one methodology is implemented. This methodology conforms to DIS standard terms that describe the damage state of an entity. (Specifically, lethality results are a subset of damage “states” in the DIS entity state appearance field.) This methodology divides lethality results in terms of mobility, fire power, and catastrophic “kills” (MFK). Expanding the server to handle other methodologies would require adding another probability (result) classes in addition to MFK, along with APIs to support them. (Volume 2 of this report explains how this may be done.)

6.3.1.2 A Variety of Answers for Each Vulnerability Methodology

Within the context of a vulnerability method, effects may be returned in a variety of formats, depending on the form that the query takes (or which API is called). In addition to (or instead of) returning the results, the server can return the raw probabilities and let the client decide how to use them. For example, the client may query what is the probability that no significant damage occurred (a single probability), or the client may query what is the probability of each of the possible outcomes occurring (multiple probabilities). However, it is recommended that for DIS battle simulations, clients should request the results (and not the probability of a result occurring). In this manner, the probabilities are organized and the same random number generator is used by every client.

As concerns the one implemented vulnerability methodology (MFK), if a client requests an MFK lethality result (as opposed to the probability of a result), the answer will be one and only one of the items listed in Table 1.

Table 1. MFK “Probability” Space

Damage State	Explanation
MKILL	Mobility and only mobility kill.
FKILL	Firepower and only firepower kill.
MFKILL	Both mobility and firepower kills.
KKILL	Catastrophic kill.
NoDamage	No additional damage inflicted.

The results in Table 1 describe the universe of all possible outcomes under the MFK methodology.

Along with each of the results, a flag will be returned, which informs the client if an actual lethality data source was used to determine the weapon effects. The only reason that a source would not be used is because it was not found by the server (because it did not exist) or because it was inaccessible (by the server). In either case, when a valid lethality source cannot be accessed, then the result is generated from a random draw of a single fixed (invalid) distribution. In the case of MFK, this distribution is portrayed in Figure 5.

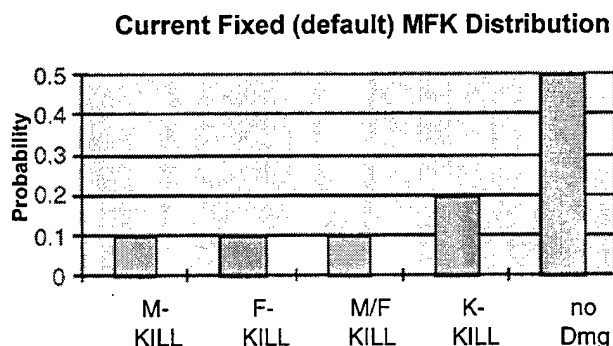


Figure 5. Default Results (when a valid lethality data source is not available).

In this way, the server can guarantee that all data “holes” are filled.² Also, all clients will know when they are receiving results from an (invalid) default distribution.

²This distribution is maintained in the API function `VL_mfkDIS_ResultGenericRandomDraw()` and may be modified there. See Volume 2 of this report for details (Appendix B, vl(3)).

6.3.2 Look-up Table

The server returns damage results found by accessing “look-up” tables. At least one data file is associated with each target-threat combination. (However, the same file may be used for any number of targets and threats.) Currently, these files must be accessed within the file system of the host computer, but the server may be relatively easily extended to add remote file access. This is because files are specified by a uniform resource locator (URL). Thus, a natural extension is to allow look-up table file access over a network by a number of protocols (http, gopher, ftp, etc.). Naturally, proper security measures must be maintained if any of these data are classified.

Currently, the server only implements local file access (e.g., *file:///tmp/file123.dat*) and will return an error if a datum source is specified as a remote host. URL access is serviced by employing a tiny subset of the Libwww (a general purpose web API provided by the World Wide Web Consortium). Therefore, expanding the capability of the server (by adding more functionality to the Libwww) should be relatively easy. The current (and final version of Libwww) is available at URL address <http://www.w3.org/Library>.

Once further development is made to allow for other types of URL access, the server would not be limited to just static “look-up” tables. The URL could reference another database server with queries, or even another application that could return calculations “on the fly” from higher fidelity vulnerability/lethality models. ARL’s Survivability/Lethality Analysis Directorate (SLAD) has an ongoing project to make models available for “on-the-fly” analysis in the future. However, it is unclear what form the “front-end” server for this capability will assume at this time.

6.4 Beyond DIS

While numerous server components are tailored to the DIS environment, the server or portions of the server could be expanded for use in other environments. This is allowed by the layered design of the server. For example, an HLA³ “front end” may be added to the server simply by changing one component (the DIS monitor component) to listen to “HLA” traffic instead of “DIS” traffic. In the language of HLA, the server’s clients would become “federates” which “subscribed” to the server’s “published” lethality results. The appendix explains in a little more detail the division of the components and the services each provides. (Another HLA compatibility option would be to leave the server as a “DIS only” tool and insert a translation

³ The U.S. Department of Defense high level architecture (HLA); first release specification was in August 1996.

utility on the network such as the HLA Gateway⁴.) This tool translates relevant HLA network traffic into DIS traffic (PDUs). Reference [3] suggests some possible approaches toward HLA migration.

7. SUMMARY

A DIS lethality server has been implemented. It has the potential to eliminate differences in how lethality effects are treated and to remove variances in which data sources are used among DIS exercise participants. This supports a tighter and more easily handled DIS exercise configuration management (since all lethality sources of a distributed exercise are handled in a single location) and conforms to recommended standards [4].

The server is designed to be flexible and expandable. Detailed examples of how to expand it are presented in Volume 2.

The server can increase the ease and amount of DIS exercise “free play” flexibility and can decrease the amount of overall pre-exercise preparation.

⁴ Developed at the Institute for Simulation and Training, Orlando, Florida
(<http://www.ist.ucf.edu/labsproj/projects/hgprode.htm>).

INTENTIONALLY LEFT BLANK

REFERENCES

1. IEEE Computer Society. "Standard for Distributed Interactive Simulation-Application Protocols." IEEE Standard 1278.1-1995, Institute of Electrical and Electronics Engineers, Inc., NY, 1995.
2. Deitz, P.H., et al. "The Generation, Use, and Misuse of 'PKS' in Vulnerability/Lethality Analysis." ARL-TR-1640, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, 1997.
3. Sauerborn, G.C. "Communicating Platform Vulnerability in a Distributed Environment." Paper: 98F-SIW-130, Simulation Interoperability Workshop Papers, The Simulation Interoperability Standards Organization (SISO), pp. 809-815, September 1998.
4. IEEE Computer Society. "Recommended Practice for Distributed Interactive Simulation—Verification, Validation, and Accreditation." IEEE Standard 1278.4-1996 (Section 6.5.4.3 Verify Data and Databases), Institute of Electrical and Electronics Engineers, Inc., NY, 1996.

INTENTIONALLY LEFT BLANK

APPENDIX A
DIS LETHALITY SERVER ARCHITECTURE

INTENTIONALLY LEFT BLANK

DIS LETHALITY SERVER ARCHITECTURE

Figure A-1 displays a view of the server's architectural design. Boxes enclosed by solid lines represent independent processes. Each of these processes may be run on separate computers. The one exception is the **DIS Server** and **DIS Monitor**; these two processes must reside on the same host machine, as indicated by the dotted box. Dashed lines separating the **VL API** and **Data Manager** indicate that these represent DIS server service layers (APIs) which must reside within a parent process.

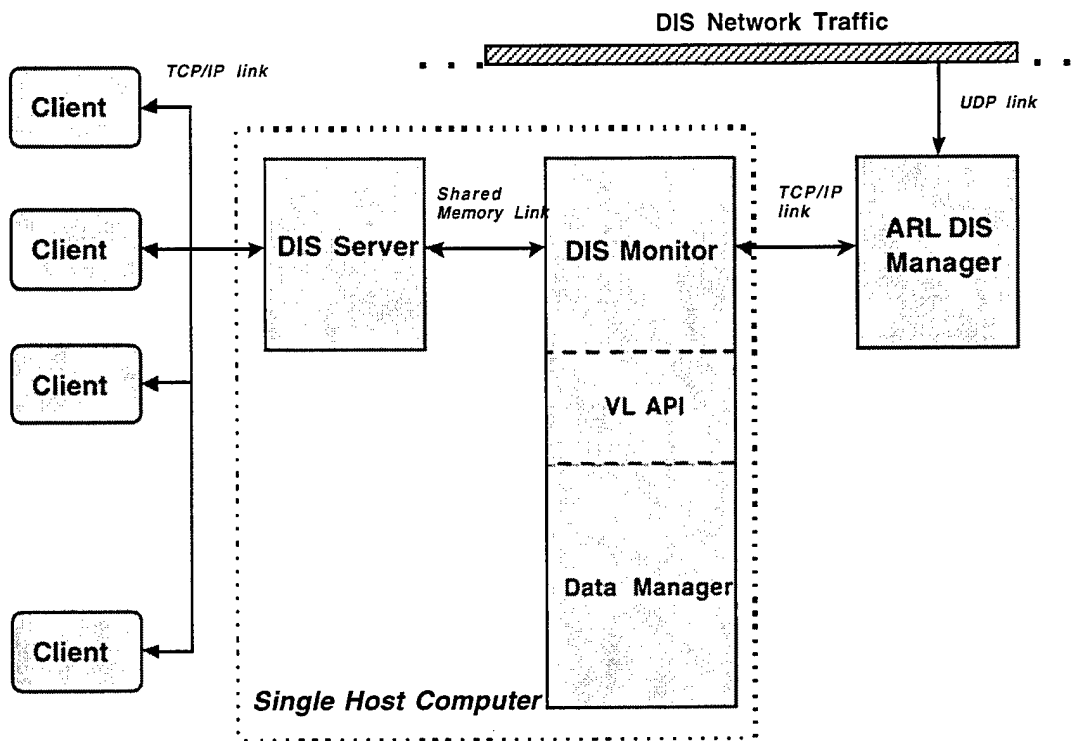


Figure A-1. DIS Server Architecture.

The architecture layout seen in Figure A-1 is a more detailed depiction of the networking topology shown in Figure 3. The clients' connectivity to the DIS network is not shown. To connect to the DIS network, clients may choose to use the **ARL DIS Manager** (which is freely provided with the lethality server), use a commercially available product (such as VR Link[®]), or their own in-house DIS networking library. It is not the responsibility or within the scope of the lethality server to decide how clients prefer to connect to the DIS network.

An explanation of the components follows:

- The **ARL DIS Manager** monitors DIS PDUs and sends them to its own clients. In this case, the DIS manager has one client (the **DIS Monitor**). Because the DIS monitor currently only is concerned with MFK vulnerability, it only requests to receive entity state, fire, and detonation PDUs (since these are the only PDUs necessary to calculate MFK results). The DIS monitor may request other PDU types from the DIS manager as necessary.
- The **DIS Monitor** monitors all fire/detonation events (along with information concerning any entities involved). It maintains cached records of these events. In this way, the parameters involved will be available when the **DIS Server** queries it for the results of a particular detonation.
- Upon receipt of a query from the DIS server, the DIS monitor calls the **VL API** which sets the appropriate parameters that describe the conditions at the time of the detonation (e.g., munition type, velocity, etc.). The **VL API** then calls the **Data Manager API** which provides data. The **VL API** layer then returns these data in a format appropriate to query.
- The **Data Manager API** manages many types of low level data. It maintains records of where to find data sources for each entity and threatening munition. It keeps track of which functions are used to read each type of data source into memory and (once read) which function to use to find results in the cached memory data structures. It is also responsible for maintaining which DIS enumerations are used to describe a particular vehicle, munition, or other item.
- The job of the **DIS Server** component is relatively simple. This component merely passes client queries to the **DIS Monitor** and returns the DIS monitor's results to the client. This raises the question, why is the DIS server component necessary? There are several benefits to using a networked approach. First, it allows a "fair fight" if all clients are assured they are receiving data from the same source and in the same manner. Second, VV&A of the simulation exercise is made easier since only one lethality data set needs to be checked. Simulation development time might be decreased. These points were made in Section 3.

Despite these advantages, some applications may find it beneficial (for speed or other reasons) to use a non-networked server architecture (and thereby bypass the **DIS Server** component). The server's modular design allows this. All that is necessary is for the application to use the API calls directly. Doing so will tremendously reduce data latency. The drawback is

that the application will have to handle many of the bookkeeping services provided by the DIS monitor, as well as provide its own interface to the DIS traffic.⁵ Figure A-2 displays the architectural layout of this approach.

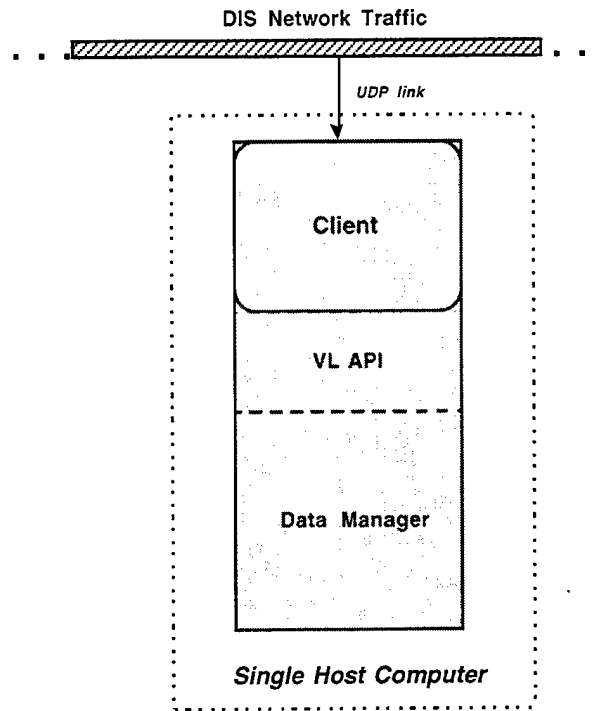


Figure A-2. Non-Networked Connection to DIS Server APIs.

Figure A-2 is a more detailed version of the topology seen in Figure 4. Here, the client is responsible for monitoring the DIS network traffic and passing to the **VL API** the appropriate parameters. The parameters passed to the API functions would be in the form of PDUs.

⁵Although as mentioned, the **ARL DIS Manager** may be used to connect to the DIS network if the user so desires.

INTENTIONALLY LEFT BLANK

<u>NO. OF COPIES</u>	<u>ORGANIZATION</u>	<u>NO. OF COPIES</u>	<u>ORGANIZATION</u>
2	ADMINISTRATOR DEFENSE TECHNICAL INFO CENTER ATTN DTIC OCP 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218	1	AMCOM MRDEC ATTN AMSMI RD W C MCCORKLE REDSTONE ARSENAL AL 35898-5240
1	DIRECTOR US ARMY RESEARCH LABORATORY ATTN AMSRL CS AL TA REC MGMT 2800 POWDER MILL RD ADELPHI MD 20783-1197	1	CECOM ATTN PM GPS COL S YOUNG FT MONMOUTH NJ 07703
1	DIRECTOR US ARMY RESEARCH LABORATORY ATTN AMSRL CI LL TECH LIB 2800 POWDER MILL RD ADELPHI MD 207830-1197	1	CECOM SP & TERRESTRIAL COMMCTN DIV ATTN AMSEL RD ST MC M H SOICHER FT MONMOUTH NJ 07703-5203
1	DIRECTOR US ARMY RESEARCH LABORATORY ATTN AMSRL D R WHALIN 2800 POWDER MILL RD ADELPHI MD 20783-1197	1	US ARMY INFO SYS ENGRG CMND ATTN ASQB OTD F JENIA FT HUACHUCA AZ 85613-5300
1	DIRECTOR US ARMY RESEARCH LABORATORY ATTN AMSRL DD J J ROCCHIO 2800 POWDER MILL RD ADELPHI MD 20783-1197	1	US ARMY NATICK RDEC ACTING TECHNICAL DIR ATTN SSCNC T P BRANDLER NATICK MA 01760-5002
1	DOD JOINT CHIEFS OF STAFF ATTN J39 CAPABILITIES DIV CAPT J M BROWNELL THE PENTAGON RM 2C865 WASHINGTON DC 20301	1	US ARMY RESEARCH OFC 4300 S MIAMI BLVD RESEARCH TRIANGLE PARK NC 27709
1	OFC OF THE DIR RSRCH AND ENGRG ATTN R MENZ PENTAGON RM 3E1089 WASHINGTON DC 20301-3080	1	US ARMY SIMULATION TRAIN & INSTRMNTN CMD ATTN J STAHL 12350 RESEARCH PARKWAY ORLANDO FL 32826-3726
2	OFC OF THE SECY OF DEFNS ATTN ODDRE (R&AT) G SINGLEY ODDRE (R&AT) S GONTAREK THE PENTAGON WASHINGTON DC 20301-3080	1	US ARMY TANK-AUTOMOTIVE & ARMAMENTS CMD ATTN AMSTA AR TD M FISETTE BLDG 1 PICATINNY ARSENAL NJ 07806-5000
1	OSD ATTN OUSD(A&T)/ODDDR&E(R) ATTN R J TREW WASHINGTON DC 20301-7100	1	US ARMY TANK-AUTOMOTIVE CMD RD&E CTR ATTN AMSTA TA J CHAPIN WARREN MI 48397-5000
		1	US ARMY TRAINING & DOCTRINE CMD BATTLE LAB INTEGRATION & TECH DIR ATTN ATCD B J A KLEVECZ FT MONROE VA 23651-5850
		1	NAV SURFACE WARFARE CTR ATTN CODE B07 J PENNELLA 17320 DAHLGREN RD BLDG 1470 RM 1101 DAHLGREN VA 22448-5100

<u>NO. OF COPIES</u>	<u>ORGANIZATION</u>	<u>NO. OF COPIES</u>	<u>ORGANIZATION</u>
1	DARPA ATTN B KASPAR 3701 N FAIRFAX DR ARLINGTON VA 22203-1714	1	OPTOMETRICS INCORPORATED ATTN FREDERICK G SMITH 3115 PROFESSIONAL DRIVE ANN ARBOR MI 48104-5131
1	UNIV OF TEXAS ARL ELECTROMAG GROUP CAMPUS MAIL CODE F0250 ATTN A TUCKER AUSTIN TX 78713-8029	1	DIR US ARL ATTN AMSRL SL EP (G MAREZ) WHITE SANDS MISSILE RANGE NM 88002
1	HICKS & ASSOCIATES, INC. ATTN G SINGLEY III 1710 GOODRICH DR STE 1300 MCLEAN VA 22102	1	DIR US ARMY TRAC ATTN ATRC WE (LOUNELL SOUTHARD) WHITE SANDS MISSILE RANGE NM 88002
1	ARL ELECTROMAG GROUP CAMPUS MAIL CODE F0250 A TUCKER UNIVERSITY OF TEXAS AUSTIN TX 78712	4	DIR US ARMY TRAC ATTN ATRC WEC JOE AGUILAR CARROL DENNY DAVID DURDA PETER SHUGART WHITE SANDS MISSILE RANGE NM 88002
1	SPECIAL ASST TO THE WING CDR 50SW/CCX CAPT P H BERNSTEIN 300 O'MALLEY AVE STE 20 FALCON AFB CO 80912-3020	3	CDR TARDEC ATTN AMSTA TR D M/S 207 FSCS ROGER HALLE GEORGE SIMON WARREN MI 48397-5000
1	HQ AFWA/DNX 106 PEACEKEEPER DR STE 2N3 OFFUTT AFB NE 68113-4039	3	CDR ARDEC ATTN AMSTA AR FSS JULIE CHU DON MILLER BILL DAVIS PICATINNY ARSENAL NJ 07806-5000
1	APPLIED RESEARCH ASSOCIATES INC ATTN MR. ROBERT SHANKLE 219 W BEL AIR AVENUE SUITE 5 ABERDEEN MD 21001	1	DEFENSE THREAT REDUCTION AGENCY ATTN SWE (WALTER ZIMMERS) 6801 TELEGRAPH ROAD ALEXANDRIA VA 22310
1	CDR US ARMY AVIATION RDEC CHIEF CREW ST R7D (DR N BUCHER) MS 243-4 AMES RESEARCH CENTER MOFFETT FIELD CA 94035		<u>ABERDEEN PROVING GROUND</u>
1	ITT INDUSTRIES ATTN CHARLES WOODHOUSE 2560 HUNTINGTON AVE ALEXANDRIA VA 22303	2	DIRECTOR US ARMY RESEARCH LABORATORY ATTN AMSRL CI LP (TECH LIB) BLDG 305 APG AA
1	ITT INDUSTRIES ATTN MICHAEL O'CONNOR 600 BLVD SOUTH SUITE 208 HUNTSVILLE AL 35802	1	US ARMY EDGEWOOD RDEC ATTN SCBRD TD J VERVIER APG MD 21010-5423
1	RAYTHEON SYSTEMS COMPANY ATTN JOHN D POWERS 6620 CHASE OAKS BLVD MS 8518 PLANO TX 75023		

<u>NO. OF COPIES</u>	<u>ORGANIZATION</u>	<u>NO. OF COPIES</u>	<u>ORGANIZATION</u>
8	DIR AMSAA ATTN P DEITZ M BORROUGHS B BRADLEY J BREWER D HODGE D JOHNSON R NORMAN A WONG	1	PRIN DPTY FOR ACQTN HDQ US ARMY MATL CMND ATTN AMCDCG A D ADAMS 5001 EISENHOWER AVE ALEXANDRIA VA 22333-0001
26	US ARMY RESEARCH LABORATORY ATTN AMSRL WM BF J LACETERA AMSRL WM BF G SAUERBORN (25 CYS) BLDG 120	1	DPTY CG FOR RDE HDQ US ARMY MATL CMND ATTN AMCRD BG BEAUCHAMP 5001 EISENHOWER AVE ALEXANDRIA VA 22333-0001
4	US ARMY RESEARCH LABORATORY ATTN AMSRL SL BV R SANDMEYER AMSRL SL BV J ANDERSON AMSRL SL BV C KENNEDY AMSRL SL BV M MUUSS BLDG 238	1	COMMANDER US ARMY MATERIEL COMMAND ATTN AMCDE AQ 5001 EISENHOWER AVENUE ALEXANDRIA VA 22333-0001
4	US ARMY RESEARCH LABORATORY ATTN AMSRL M SMITH AMSRL G MOSS BLDG 321		
2	DIR USARL AMSRL WM W DR INGO MAY LARRY JOHNSON BLDG 4600		
1	DIR USARL AMSRL WM B A. HORST BLDG 4600		
1	DIR USARL AMSRL-SL-B J SMITH BLDG 328		
	<u>ABSTRACT ONLY</u>		
1	DIRECTOR US ARMY RESEARCH LABORATORY ATTN AMSRL CS AL TP TECH PUB BR 2800 POWDER MILL RD ADELPHI MD 20783-1197		
1	COMMANDER US ARMY MATERIEL COMMAND ATTN AMCRDA TF 5001 EISENHOWER AVENUE ALEXANDRIA VA 22333-0001		
1	PRIN DPTY FOR TECH GY HDQ US ARMY MATL CMND ATTN AMCDCG T 5001 EISENHOWER AVE ALEXANDRIA VA 22333-0001		

INTENTIONALLY LEFT BLANK

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, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE February 1999	3. REPORT TYPE AND DATES COVERED Final	
4. TITLE AND SUBTITLE The Distributed Interactive Simulation (DIS) Lethality Communication Server Volume I: Overview			5. FUNDING NUMBERS PR: 1L162618AH80	
6. AUTHOR(S) Sauerborn, G.C. (ARL)				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory Weapons & Materials Research Directorate Aberdeen Proving Ground, MD 21010-5066			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory Weapons & Materials Research Directorate Aberdeen Proving Ground, MD 21010-5066			10. SPONSORING/MONITORING AGENCY REPORT NUMBER ARL-TR-1775	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) A new approach to handling battle simulation lethality is presented. In this approach, a single vulnerability/lethality server provides standard distributed interactive simulation (DIS) damage states to entities fast enough for most real-time applications. Benefits include DIS simulations being freed from the burden of maintaining damage state tables and DIS exercises being easier to configure as a whole. A DIS software implementation is freely available by contacting the author.				
14. SUBJECT TERMS client server DIS lethality degraded states distributed simulation vulnerability			15. NUMBER OF PAGES 35	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT	