BBN 6369

2

# THE SIMNET NETWORK AND PROTOCOL

12 February 1987

BBN Laboratories Incorporated

Contract MDA903-86-C-0309

SIMNET

**DARPA**
LT. COL JACK A. THORPE
(202) 694-8232, AUTOVON 224-8232
TACTICAL TECHNOLOGY OFFICE
DEFENSE ADVANCED RESEARCH PROJECT AGENCY (DARPA)
1400 WILSON BLVD.,ARLINGTON, VA 22209-2308

**CONSULTANT**
GARY W. BLOEDORN, COL (USA RET)

**PERCEPTRONICS**
MR. RICHARD TAYLOR
(818) 884-3485
21122 ERWIN ST, WOODLAND HILLS, CA 91367-3713

**BBN LABORATORIES INCORPORATED**
DR. DUNCAN C. MILLER
(617) 497-3334
10 MOULTON STREET, CAMBRIDGE MA 02238

**BBN DELTA GRAPHICS INCORPORATED**
MR. MICHAEL CYRUS
(206) 746-6800
14100 S E 36th ST, BELLEVUE, WA 98006

# The SIMNET Network and Protocol

Contract MDA903-86-C-0309

Arthur R. Pope

October 1986

Prepared by:

BBN Laboratories Incorporated
10 Moulton Street
Cambridge, Massachusetts 02238

Prepared for:

Defense Advanced Research Projects Agency
Tactical Technology Office
1400 Wilson Boulevard
Arlington, Virginia 22209-2308

(F

## Preface

### SIMNET: Advanced Technology for the Mastery of War Fighting

which has as its [illegible]

SIMNET is an advanced research project sponsored by the Defense Advanced Research Projects Agency (DARPA) in partnership with the United States Army. Currently in its third year, the goal of the program is to develop ~~the~~ technology to build a large-scale network of interactive combat simulators. This simulated battlefield will provide, for the first time, an opportunity for fully-manned platoon-, company-, and battalion-level units to fight force-on-force engagements against an opposing unit of similar composition. Furthermore, it does so in the context of a joint, combined arms environment with the complete range command and control and combat service support elements essential to actual military operations. All of the elements that can affect the outcome of a battle are represented in this engagement, with victory likely to go to that unit which is able to plan, orchestrate, and execute their combined-arms battle operations better than their opponent. Whatever the outcome, combat units will benefit from this opportunity to practice collective, combined arms, joint war fighting skills at a fraction of the cost of an equivalent exercise in the field.

While simulators to date have been shown to be effective for training specific military skills, their high costs have made it impossible to buy enough simulators to fully train the force. Further, because of the absence of a technology to link them together, they have not been a factor in collective, combined arms, joint training. SIMNET addresses both of these problems by aiming its research at three high payoff areas:

1) Better and cheaper collective training for combined arms, joint war fighting skills;

2) A testbed for doctrine and tactics development and assessment in full combined arms joint setting; and

3) A "simulate before you build" development model. Keywords: [illegible] (KR)

These payoffs are achievable because of recent breakthroughs in several core technologies which been applied to the SIMNET program:

- High speed microprocessors
- Parallel and distributed multiprocessing
- Local area and long haul networking
- Hybrid depth buffer graphics
- Special effects technology
- Unique fabrication techniques

These technologies, applied in the context of "selective fidelity" and "rapid prototyping" design philosophies, have enabled SIMNET development to proceed at an unprecedented pace, resulting in the fielding of the first production units at Fort Knox, KY just three years into the development cycle.

In addition to the basic training applications, work is underway to apply SIMNET technology in the area of combat development to aid in the definition and acquisition of weapon systems. This is made possible because of the low cost of the simulators, the ease with which they can be modified, and the ability to network them to test the employment of a proposed weapon system in the tactical context in which it will be used, i.e., within the context of the combined arms setting.

Work on SIMNET is being carried out by co-contractors Bolt Beranek and Newman, Inc. (BBN) and Perceptronics, Inc. Perceptronics is responsible for training analysis, overall system specification, and the physical simulators, and BBN is responsible for the data communication and computer-based distributed simulation and the computer image generation (CIG) subsystems. The project is a total team effort.

DARPA is the DoD agency chartered with advancing the state of the art in military technology by sponsoring innovative, high risk/high payoff research and development.

## TABLE OF CONTENTS

## LIST OF FIGURES

# 1. INTRODUCTION

SIMNET is a project aimed at developing technologies for large scale networks of interactive vehicle simulators. Many computers must work together to perform a SIMNET simulation. Each computer has a particular role to play, such as simulating a single combat vehicle or monitoring the health of other computers in the simulation. These computers must communicate with each other while carrying out a simulation. They do so by means of a *network* that interconnects them. The network allows any computer to send information to any other computer quickly and efficiently. The dialog between computers on the network is governed by a set of rules and conventions that are collectively known as the SIMNET *protocol.*

The purpose of this document is to describe the way in which the computers participating in a SIMNET simulation communicate with each other in order to carry out that simulation. In chapter 2 we introduce the network and the various computers attached to it, which are described in terms of the roles they play in a simulation. In the following chapter we list the technical capabilities the network must have in order to support a simulation, and we identify the particular network technology chosen for SIMNET. Chapter 4 introduces some terminology and conventions used in discussing the SIMNET protocol. Chapter 5 defines the protocol rules that dictate how the computers participating in a simulation communicate with each other.

The SIMNET protocol incorporates contributions from several individuals at BBN. They are James Chung, David Epstein, Duncan Miller, John Morrison, Arthur Pope, Brian O'Toole, Daniel Van Hook, and Philip Yoo. The author collected, synthesized, and documented their ideas.

Some notes about terminology and notation: in this document, the term *exercise* refers to any SIMNET simulation irrespective of whether the purpose of the simulation is training. Referenced publications are listed in chapter 6 and noted in the text thus: [1].

# 2. THE SIMNET NETWORK

The network that interconnects the computers located at a single SIMNET site (or installation) is called a *local area network*, for it spans a limited, local area. When two or more SIMNET sites are involved in a joint exercise, their respective local area networks are interconnected by a *long haul network* that allows communication over great distances.

Supporting an exercise is the principle purpose of the network. The roles of computer systems involved in an exercise, and the manner in which the network supports an exercise, are discussed in the first section of this chapter. Subsequent sections discuss other purposes of the network: those of network management, and data collection.

## 2.1 Simulation

Every exercise must involve the participation of at least one computer system called a *Management, Command and Control (MCC)* system. The MCC system is responsible, among other things, for beginning and ending the exercise under the direction of someone called the *Battlemaster*. The MCC system may itself be composed of several computers, but for the purposes of this document we will treat the MCC system as a single computer attached to the network at a single point. (It is not relevant to either the network or its protocol whether the task of an MCC system is performed by a single computer or several working in concert.)

Every exercise also involves the participation of zero or more combat vehicle simulators, called *simulators* for short. Each simulator is responsible for performing the simulation of a single combat vehicle. There may be various types of simulators corresponding to various types of combat vehicles, such as tanks and personnel carriers, all taking part in the exercise. Each simulator may itself be composed of several computers but, again, for the purposes of this document we will treat a simulator as a single computer attached to the network at a single point.

Simulators are initialized at the beginning of an exercise by the MCC system. When it is initialized, a simulator is provided with an initial position and orientation on the terrain, and with initial quantities of ammunition and fuel. Once initialized, each simulator carries out the simulation of a single combat vehicle under the direction of that vehicle's crew. It periodically reports the position, orientation and appearance of the simulated vehicle, via the network, to all other simulators and to the MCC system.

During the exercise the MCC system simulates indirect fire, close air support, resupply, and repair. All of these activities are simulated under the direction of individuals, such as the fire support and air liaison personnel, who enter their decisions at consoles. The simulators are notified of these activities via the network.

The MCC system also simulates vehicles such as command posts, howitzers, mortars, and supply trucks. Unlike the simulated combat vehicles, which are directed by crews, the vehicles simulated by the MCC system are not operated by individual crews. The general behavior of these vehicles is controlled by personnel working with the MCC system, but fine details—such as the precise locations of vehicles, or the exact moments at which mortars fire—are determined by the MCC system. To reflect their distinction from crew-operated combat vehicles, the vehicles simulated by the MCC system are called *computer controlled vehicles.*

At the end of the exercise, the MCC system issues disabling orders to each of the participating simulators, and ceases its own activities and simulations of computer controlled vehicles.

Figure 1 shows, as an example, the components supporting a small SIMNET exercise that involves an MCC system and several simulators of various types interconnected by a network. Note that the network interconnects all components, allowing any one to communicate with all others.

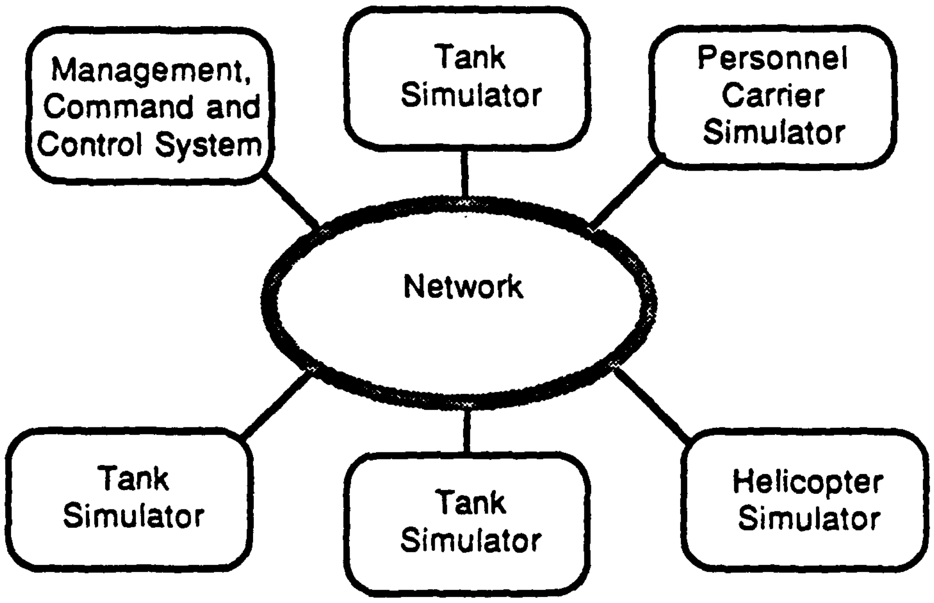


Figure 1. An MCC system and five vehicle simulators interconnected by a network.

An MCC system allows organization, initialization, combat support, and combat service support of up to one full battalion (or task force) of simulators. When two or more battalions are engaged in a joint exercise, each must be supported by its own MCC system operating under the direction of that battalion's staff. All of the participating simulators and MCC systems communicate with each other via the same network for the duration of the joint exercise. Although these computers may all be attached to a single local area network, the battalions may also be located at separate physical sites, on separate local area networks that are linked together by a long haul network.

Figure 2 depicts a network comprised of two battalions of simulators with their respective MCC systems. The two battalions are at separate physical sites and each site has its own local area network. The local area networks are themselves connected to a single long haul network by *gateway* computers that relay information between the networks. This entire collection of networks can be viewed as a single, logical network allowing any computer at either site to communicate with all other computers at both sites.

The long haul network can interconnect any number of local area networks to form a single, logical network. At each site the simulators and MCC systems attached to the local area network may comprise one entire battalion, a part of a batallion, or multiple batallions. Altogether, many sites and many battalions can participate in a single, joint exercise.

The long haul network permits the following communication among sites:

1. The MCC systems at the various sites use the long haul network to coordinate among themselves such things as when the joint exercise will begin, and upon what terrain the exercise will take place.

2. Each simulator and MCC system uses the long haul network to remain informed of the position and appearance of all simulated vehicles at all sites.

3. An MCC system at one site can use the long haul network to resupply and repair simulated vehicles at other sites.

4. Announcements of direct fire, artillery fire, mortar fire and bombing are conveyed by the long haul network to all simulators and MCC systems at all sites.

The network can be used to perform more than one exercise at any one time, although no single simulator or MCC system can simultaneously participate in more than one exercise.

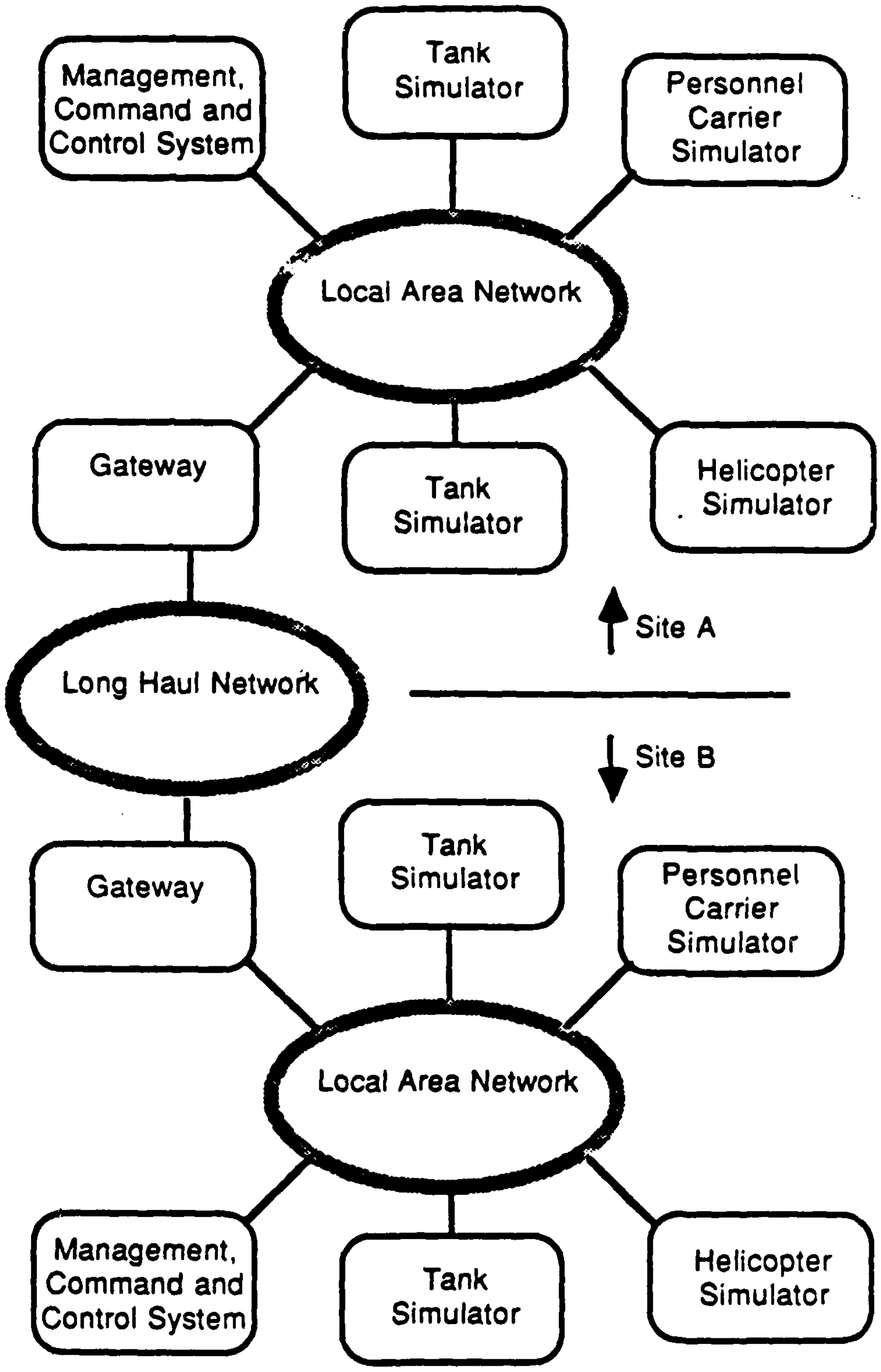


Figure 2. Two SIMNET local area networks joined by a long haul network

## 2.2 Network Management

Sizeable SIMNET installations may include a computer system called a *Network Operations and Maintenance (NOM)* system. The purpose of the NOM system is to assist the installation staff who are responsible for ensuring the reliability and availability of the simulators and MCC systems at that site. The NOM system monitors the health of simulators and MCC systems, via the network, to detect and report hardware failures.

## 2.3 Data Collection

A SIMNET installation used for research may include a computer system called a *Data Logger*. This system records data—such as the location of a simulated vehicle—as it becomes available on the network in the course of an exercise. Such data can be used to support research based on experiments done with SIMNET.

A Data Logger does not interfere with exercises in any way. It is merely listens, passively, on the network to the information produced during exercises.

# 3. TECHNICAL CHARACTERISTICS OF THE SIMNET NETWORK

Certain characteristics are required of any network used to convey the SIMNET protocol. These characteristics are described in this chapter. The implementation of a SIMNET local area network is also documented in this chapter. The SIMNET long haul network is still under design as part of the SIMNET research and development project.

A SIMNET network must meet the following criteria:

1. Each computer attached to the network must be identified by a unique *network address*.
2. The network must allow *connectionless data transfer*, also known as a *datagram service*. This means that a computer on the network must be able to to transfer data to another computer on the network in a single operation, without first establishing a connection with the destination computer. The unit of data transfered in a single operation is called a *datagram*.
3. The delay incurred by a datagram travelling from its source to its destination should be at most 500 milliseconds. Moreover, the delay should not vary widely among datagrams.
4. There is no need for the network to retain the ordering of datagrams. Two datagrams sent in quick succession, either by the same or different computers, can be delivered to their destinations in any order. (Note that a corollary to criterion 3 is that two datagrams separated by an interval of more than 500 milliseconds will be delivered in the correct order.)
5. A datagram must be able to convey 1024 bits of information.
6. The network must allow any datagram to be sent to a single destination computer (called a *point-to-point* transfer), to a set of destination computers simultaneously (called *multicasting*), or to all computers on the network simultaneously (called *broadcasting*).
7. When delivering a datagram to its destination computer, the network must supply the network address of the computer that sent the datagram.
8. The network must be able to detect, though not necessarily correct, errors in transmission.

## 3.1 The Local Area Network

The SIMNET local area network is an Ethernet™ corresponding to the Ethernet Version 2.0 [1] or IEEE 802.3 [2] specifications. This local area network meets all of the criteria listed above.

A full description of Ethernet may be found in the referenced documents [1] [2]. What follows is a brief summary of this network. An Ethernet uses a shielded coaxial cable as its medium. As many as 1024 computers may be attached to the cable. The maximum distance between the points of attachment of any two computers (as measured along the cable) is 1500 meters. If repeaters are inserted in the cable, this distance can be extended to 2800 meters. The network provides a single physical channel operating at a fixed data rate of 10 Megabits per second.

## 3.2 The Long Haul Network

The SIMNET long haul network and its gateway computers are still under design as part of the SIMNET research and development project. However the SIMNET protocol assumes that the long haul network meets all of the criteria listed above. In particular, it assumes that any computer on any local area network can send a datagram to any other computer on any of the local area networks joined by the long haul network. It further assumes that a broadcast datagram will be conveyed by the long haul network to all other computers on all other local area networks.

TM Ethernet is a registered trademark of the Xerox Corporation.

# 4. PROTOCOL TERMINOLOGY AND CONVENTIONS

This chapter introduces terminology and conventions used in defining the SIMNET protocol.

## 4.1 Simulator States

When discussing the SIMNET protocol, we must sometimes view a simulator as a simplified three-state system in order to describe certain aspects of its behavior. The three states of the simplified simulator and the transitions between those states are shown in Figure 3. The states represent the following conditions:

| | |
|---|---|
| *unavailable* | the simulator is powered down, broken, or otherwise unavailable. |
| *idle* | the simulator is available on the network, but is has not yet been instructed to begin simulating a vehicle (i.e., the simulated vehicle has not yet been providedwith a starting position and initial quantities of supplies). |
| *active* | the simulator is presently simulating a vehicle under the direction of its crew. |

The SIMNET protocol provides a way for an MCC system to change a simulator's state from idle to active (a process called *activating*) and back again (*deactivating*). The protocol doesn't, however, provide a way to change a simulator's state to or from unavailable—a simulator cannot be powered on or off remotely using the network.

## 4.2 Procedures

The SIMNET protocol is logically divided into several subsets of dialog, called *procedures*. Each procedure has a certain purpose that is fulfilled by a particular set of computers on the network exchanging certain information. What follows is a summary of these procedures:

1. The *Equipment Monitoring Procedure* is used by the NOM system to monitor the health and availability of simulators and MCC systems on a local area network.

2. The *Vehicle Activation Procedure* changes an idle simulator to an active simulator, and the *Vehicle Deactivation Procedure* does the opposite. In other words, these procedures start and stop a combat vehicle simulation (though they do not power-up or power-down a simulator).

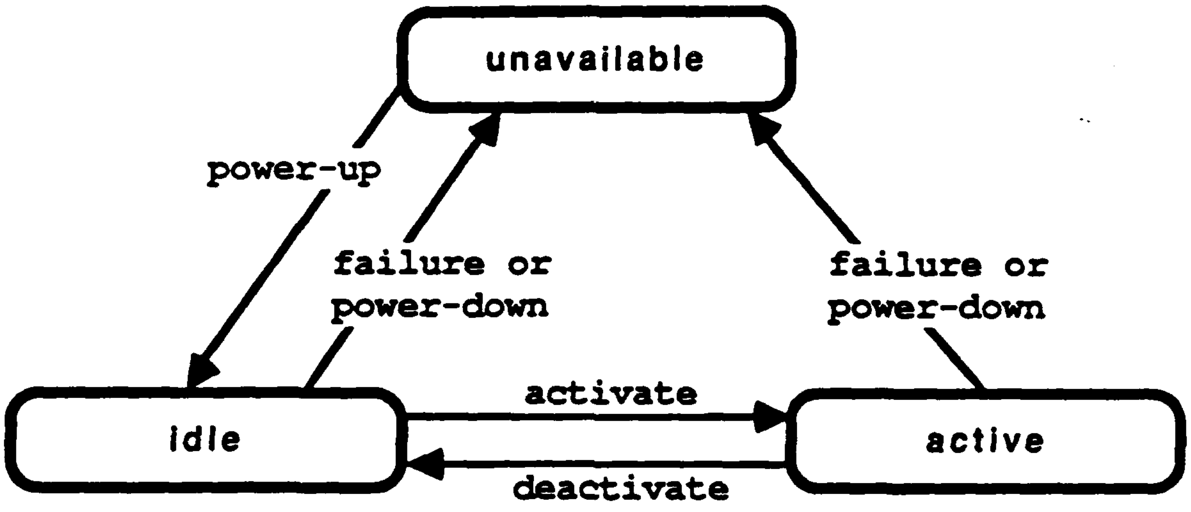


Figure 3. Simplified view of a simulator.

3. The *Vehicle Update Procedure* allows the simulators and MCC systems participating in an exercise to remain informed of the positions and appearances of all simulated vehicles, including both combat vehicles and computer controlled vehicles. This procedure is performed continuously by all MCC systems and all simulators in the active state.

4. The *Direct Fire Procedure* is used by a simulator to notify all others of direct fire caused by its crew. Direct fire weapons include cannons, machine guns and missiles.

5. The *Indirect Fire Procedure* is used by an MCC system to notify all others of artillery fire, mortar fire, and close air support bombing caused by fire support or air liaison personnel.

6. The *Vehicle Resupply Procedure* is used by an MCC system to resupply a simulated combat vehicle with ammunition or fuel, as directed by combat service support personnel.

7. The *Vehicle Repair Procedure* is used by an MCC system to repair a simulated combat vehicle, as directed by combat service support personnel.

8. The *Collision Procedure* is used by a simulator to ensure that, when its simulated combat vehicle collides with another vehicle, the computer simulating that other vehicle is also aware of the collision.

## 4.3 Protocol Data Units

The information exchanged by computers over the network is packaged in units called *protocol data units (PDUs).* The SIMNET protocol requires each PDU to be conveyed in a single network datagram.

Several kinds of PDUs are used. The first portion of the PDU always contains a number identifying the kind of the PDU; the size and format of the remainder of the PDU depend on the kind. The various kinds of PDUs are summarized here:

| | |
|---|---|
| EquipStatusPDU | is sent by an MCC system or simulator to the NOM system to report the health of its various subsystems. |
| ActivatePDU | is sent by an MCC system to a simulator to start the simulation of a combat vehicle. |
| ActivePDU | is returned by a simulator to an MCC system to acknowledge receipt of an ActivatePDU. |
| DeactivatePDU | is broadcast by an MCC system to discontinue the simulation of a combat vehicle or computer controlled vehicle. |
| VehicleAppearancePDU | is broadcast by a simulator, describing the position and appearance of its simulated combat vehicle. It is also broadcast by an MCC system for those vehicles (such as fuel trucks and ammunition carriers) that it simulates. |
| VehicleStatusPDU | is multicast by a simulator to the MCC system that activated it and to any Data Loggers on the network. This PDU checkpoints the maintenance and supplies status of a simulated combat vehicle. |
| VehicleImpactPDU | is broadcast by a simulator when the combat vehicle it simulates hits another vehicle with direct fire. |
| GroundImpactPDU | is broadcast by a simulator when the combat vehicle it simulates generates direct fire that does not strike another vehicle. |
| IndirectFirePDU | is broadcast by an MCC system when the fire support or air liaison personnel using that MCC system cause artillery rounds, mortar rounds, or bombs to explode. |
| ServiceRequestPDU | is broadcast by a simulator when the combat vehicle it simulates is capable of accepting supplies or repairs from a nearby service truck. |

| | |
|---|---|
| ResupplyOfferPDU | is sent by an MCC system to a simulator, offering that simulator's combat vehicle some quantity of supplies from a service truck. |
| ResupplyReceivedPDU | is sent by a simulator to an MCC system after having accepted supplies offered by a ResupplyOfferPDU. |
| RepairPDU | is sent by an MCC system to a simulator, describing a repair that has been completed on that simulator's combat vehicle. |
| RepairedPDU | is sent by a simulator to an MCC system, to acknowledge the receipt of a RepairPDU. |
| CollisionPDU | is broadcast by a simulator to indicate that its simulated combat vehicle has collided with another vehicle. |

## 4.4 Network Errors

It is possible that a computer may receive a PDU from the network that has become corrupted in transit. The network itself provides a means for detecting and reporting most instances of datagram corruption. An NOM system may choose to log these erroneous PDUs as an indication of the network's reliability. Other computers participating in a SIMNET exercise can safely ignore erroneous PDUs, discarding them without action. The protocol is sufficiently robust that it can, in most cases, tolerate occasional network errors without human crews participating in the simulation becoming aware of the errors.

## 4.5 Timers and Counters

Some protocol procedures call for the repeated, periodic transmission of PDUs. For example, the Equipment Status Procedure requires each computer to transmit an EquipStatusPDU periodically for the benefit of any NOM system monitoring the local area network. The definition of the protocol makes use of a conceptual device called a *timer* to describe the periodicity of a repeated transmission. Another conceptual device called a *counter* is used to describe how many times the transmission is repeated. The use of counters and timers is explained in the following paragraphs.

A *timer* is a variable used to determine the instants at which a periodically transmitted PDU must be sent. When the PDU is first sent, the timer is set to the number of seconds that must elapse before the PDU is sent again. The timer is then decremented by one unit each second. When the timer reaches zero the PDU is resent, the timer is set back to the beginning of its count, and the cycle repeats.

Some protocol procedures call for repeating the transmission of a PDU up to a specified maximum number of times. This is usually done when the sender is not sure that a PDU is being correctly conveyed to its recipient. A *counter* is a variable used to keep track of the number of times the PDU has been sent. When the PDU is first sent, the counter is set to the maximum number of times that the PDU may be sent. When the PDU is resent (usually after a timer has expired) the counter is decremented by one unit and, if it hasn't reached zero, the PDU is sent again.

Several timers and counters are involved in the definition of the SIMNET protocol. The durations of the timers and the maximum (initial) values of the counters are given names. Their names are listed here and defined completely in Appendix A. The durations of the timers are referred to as EquipStatusTime, ActivateTime, DeactivateTime, VehicleAppearanceTime, VehicleStatusTime, ResupplyTime and RepairTime. The maximum value of the counters are referred to as ActivateCount, DeactivateCount, ResupplyCount and RepairCount.

## 4.6 Concurrent Exercises

The SIMNET network and protocol allow multiple exercises to be performed simultaneously. Concurrent exercise are kept from interfering with each other on the network by assigning to each exercise a distinct integer called an *exercise identifier*. All PDUs transmitted over the network (with the exception of EquipStatusPDUs, which don't concern exercises) bear the exercise identifier of the exercise to which they pertain. The recipient of a PDU simply ignores the PDU if it bears an exercise identifier for an exercise other than the one in which the recipient is currently involved.

The procedure for generating exercise identifiers and distributing them to the MCC systems participating in an exercise has not yet been specified. Exercise identifiers are distributed to the simulators participating in an exercise by the MCC systems that initialize them, as part of the Activate Procedure.

## 4.7 Vehicle Identifiers

To every vehicle participating in an exercise is assigned a distinct integer called a *vehicle idenrifier*. Both combat vehicles and computer controlled vehicles (those simulated by an MCC system) have vehicle identifiers. No two vehicles in the same exercise have the same vehicle identifier.

Each MCC system chooses vehicle identifiers for its computer controlled vehicles and the combat vehicles it initializes. The procedure for ensuring that multiple MCC systems participating in the same exercise assign disjoint sets of vehicle identifiers has not yet been specified.

# 5. PROTOCOL PROCEDURES

In this chapter are described each of the protocol procedures governing interactions of computer systems on the SIMNET network.

## 5.1 Equipment Status Procedure

While idle or active, each MCC system and simulator periodically sends an EquipStatusPDU describing the health of its various subsystems. This PDU specifies:

- whether or not each of the various subsystems (such as the computer image generator, the interaction device controllers, and the sound generator) is operational
- in the case of a report from a simulator, the current values of various DC power supply voltages
- in the case of a report from a simulator, the current temperature within the simulator, as an indication of the correct operation of the simulator's air conditioner

This EquipStatusPDU is sent to the NOM system on the local area network every EquipStatusTime seconds. Should the NOM system not receive one of these PDUs from a particular computer for two such time periods, it may assume that that computer or its network connection has failed and notify the installation staff of the problem.

## 5.2 Vehicle Activation Procedure

The MCC system starts a simulator simulating a combat vehicle by sending an ActivatePDU to it. This PDU specifies:

- the exercise identifier of the exercise in which the vehicle will be participating
- the vehicle identifier that the simulator is to adopt for its vehicle
- the vehicle's task force alignment (either offense or defense)
- the type of vehicle (M1, M2 or M3, etc.)
- the identity of the terrain database chosen for the exercise
- the dimensions of the terrain covered by the terrain database
- the starting location of the vehicle on the terrain

- the azimuth of the vehicle's hull and turret relative to grid North
- the maintenance status of the vehicle, including the repair status of all vehicle subsystems
- the vehicle's initial fuel and ammunition loads
- the network address to which the vehicle is to send its VehicleStatusPDUs

A simulator that correctly receives an ActivatePDU immediately responds with an ActivePDU. The ActivePDU is simply an acknowledgement that the simulator has received the ActivatePDU; it doesn't imply that the initialization process was completed successfully.

The MCC system, will wait ActivateTime seconds for the ActivePDU response, then resend the ActivatePDU. It will resend ActivateCount times before concluding that the simulator is unavailable due to an equipment failure.

## 5.3 Vehicle Deactivation Procedure

The Vehicle Deactivation Procedure serves two closely related purposes:

- it allows an MCC system to terminate a simulator's simulation of a combat vehicle, and
- it allows the MCC system to notify all other computers participating in an exercise that a vehicle—either a combat vehicle or one simulated by the MCC system itself—is no longer visible on the terrain.

The MCC system may terminate a simulator's simulation of a combat vehicle by broadcasting a DeactivatePDU identifying the vehicle to be deactivated. Simulators other than the one deactivated note the deactivation by ceasing to track and display that vehicle.

By monitoring whether a simulator continues to perform the Vehicle Update Procedure, the MCC system can determine whether it is responding to the DeactivatePDU. The MCC computer periodically resends (every DeactivateTime seconds, for DeactivateCount times) a DeactivatePDU when deactivating a vehicle to ensure that all parties become aware of the deactivation.

A vehicle simulated by an MCC system, such as a fuel tanker or ammunition carrier, can be made to disappear from the terrain by broadcasting a DeactivatePDU for that vehicle and ceasing its broadcast of VehicleAppearancePDUs (described in the next section). The MCC system can make the vehicle reappear again by resuming broadcasts of the VehicleAppearancePDUs.

## 5.4 Vehicle Update Procedure

Each active simulator periodically informs others of the appearance of its simulated combat vehicle, including the position and orientation of that vehicle. Also, the MCC system periodically relates to others the appearances of the computer controlled vehicles that it simulates.

The appearance of a vehicle is described by a VehicleAppearancePDU containing the following information:

- the exercise identifier of the exercise in which the vehicle is participating
- the vehicle's vehicle identifier
- the vehicle's task force alignment (offense, defense, or both in the case of MCC vehicles serving both side in a local force-on-force exercise)
- the type of vehicle (M1, M2 or M3, etc.)
- the current location of the vehicle on the terrain
- the orientation of the vehicle's hull and its turret (if it has one)
- the elevation of the vehicle's gun barrel (if it has one)
- whether the vehicle is on fire, whether it is smoking, whether it is raising a dust cloud, and whether it is emitting exhaust smoke
- whether the vehicle has just fired a round producing a muzzle flash and, if so, the location of the muzzle flash
- the velocity of the vehicle

A VehicleAppearancePDU must be broadcast whenever any of the following conditions is met:

- VehicleAppearanceTime seconds have elapsed since the last VehicleAppearancePDU was broadcast for the vehicle;
- the vehicle's velocity differs from that last broadcast by some threshold amount;
- the orientation of the vehicle's hull or turret, or the elevation of the vehicle's gun barrel, differs from that last broadcast by some threshold amount;
- the vehicle's gun has just been fired; or
- there is a change in whether the vehicle is on fire, smoking, raising a dust cloud, or emitting engine exhaust smoke.

All simulators and MCC systems receive the broadcast VehicleAppearancePDUs and use the information contained in these PDUs to update their tables describing other vehicles in the exercise. The simulators additionally rely on these PDUs to know when to display muzzle flashes.

By monitoring the frequency of VehicleAppearancePDUs from each simulator, an MCC system can determine which simulators are active. An MCC system may assume a simulator is no longer active when it fails to receive a VehicleAppearancePDU from that simulator for two successive periods of VehicleAppearanceTime seconds. On detecting that a simulator is no longer active, the MCC system should notify all other simulators of the loss by broadcasting DeactivatePDUs according to the Vehicle Deactivation Procedure.

Each active simulator also sends a VehicleStatusPDU approximately every VehicleStatusTime seconds. This PDU contains the following information about the combat vehicle simulated:

- the exercise identifier of the exercise in which the vehicle is participating
- the vehicle's vehicle identifier
- the vehicle's task force alignment (either offense or defense)
- the type of vehicle (M1, M2 or M3, etc.)
- the maintenance status of the vehicle, including the repair status of all simulated vehicle subsystems such as drive train components
- the quantity of fuel contained in each of the vehicle's fuel tanks
- the quantity and type of ammunition contained in each of the vehicle's ammunition racks

A simulator multicasts its VehicleStatusPDUs to a subset of the computers on the network. This subset includes the MCC system that activated the simulator, so that that MCC system has recent information about the simulated combat vehicle for use in reconstituting or restarting the simulation. The subset may also include any Data Loggers on the network.

## 5.5 Direct Fire Procedure

When a combat vehicle engages another vehicle with ballistic direct fire, the firing simulator determines what has been struck by that fire. It then broadcasts either a VehicleImpactPDU or a GroundImpactPDU according to whether or not any vehicle has been struck.

A VehicleImpactPDU is broadcast if ballistic direct fire strikes a vehicle. This PDU specifies:

- the exercise identifier of the exercise in which the vehicle is participating
- the vehicle identifier of the vehicle doing the firing
- the location of the vehicle doing the firing
- the vehicle identifier of the vehicle struck
- the location of the vehicle struck
- whether the struck vehicle's hull or turret was hit
- which side of the struck vehicle was hit
- the angle of incidence of the projectile striking the vehicle's side
- the kind of ammunition fired

All simulators use the information contained in this PDU to display an explosion at the appropriate location. In addition, the simulator whose vehicle was struck uses the information to determine what, if any, damage it has sustained.

If ballistic direct fire does not strike a vehicle, a GroundImpactPDU is broadcast instead. This PDU specifies:

- the exercise identifier of the exercise in which the vehicle is participating
- the vehicle identifier of the vehicle doing the firing
- the location of the vehicle doing the firing
- the location of the impact of the projectile
- the kind of ammunition fired

## 5.6 Indirect Fire Procedure

Bombing or shelling by aircraft, howitzer or mortar is described by an IndirectFirePDU. The MCC system simulating the indirect fire broadcasts this PDU to all other MCC systems and simulators. A single IndirectFirePDU can describe several bomb or shell bursts, specifying both the spatial and temporal distributions of the bursts. In contrast to the VehicleImpactPDU, this PDU does not identify the particular vehicle(s) hit by the indirect fire; every simulator computes its own vehicle's distance from the bursts and assesses any damage.

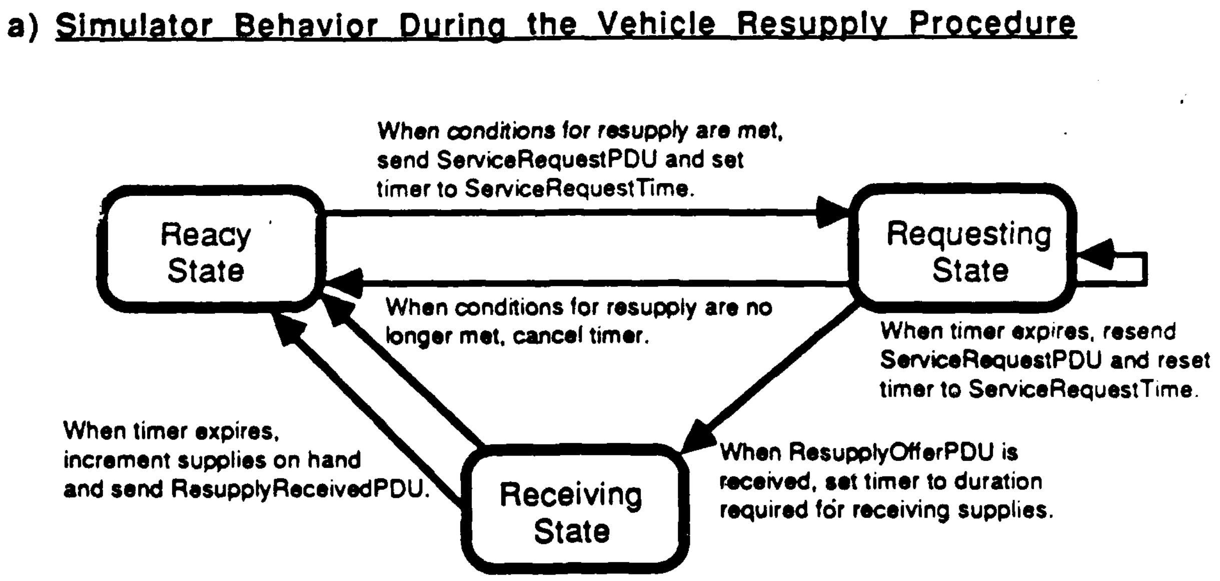
The contents of an IndirectFirePDU are:

- the exercise identifier of the exercise in which the MCC system is participating
- the kind of ordnance delivered (bomb, mortar shell, howitzer shell, etc.)
- a description of each burst, sorted in temporal order, including:
    - the amount of time by which this burst follows the previous burst (or, in the case of the first burst, the amount of time by which this burst follows broadcast of the PDU)
    - the location of the burst on or above the terrain

## 5.7 Vehicle Resupply Procedure

The Vehicle Resupply Procedure is used to transfer supplies of fuel or ammunition from a simulated service truck, such as a HEMTT, to a simulated combat vehicle. Figures 4 and 5 summarize the behavior of the combat vehicle simulator and the MCC system simulating the service truck when the Vehicle Resupply Procedure is being performed.

Briefly, the procedure is carried out as follows. At the beginning of the procedure, both the simulator and the MCC system are in their respective Ready States. A simulator transitions to Requesting State when it requests some supplies from the MCC system, and it remains in that state while awaiting a reply to its request. When that reply arrives with an offer of supplies, the simulator transitions to Receiving State and remains in that state for whatever time is required to load some portion of the supplies offered. After that time has elapsed, the simulator returns to Ready State and sends an acknowledgement to the MCC system for the portion of supplies taken. The MCC system, meanwhile, waits in Offering State from the time it offers supplies until the time an acknowledgement is received from the simulator for some portion of those supplies.


a) Simulator Behavior During the Vehicle Resupply Procedure
Reacy State
Requesting State
Receiving State
When conditions for resupply are met, send ServiceRequestPDU and set timer to ServiceRequestTime.
When conditions for resupply are no longer met, cancel timer.
When timer expires, resend ServiceRequestPDU and reset timer to ServiceRequestTime.
When ResupplyOfferPDU is received, set timer to duration required for receiving supplies.
When timer expires, increment supplies on hand and send ResupplyReceivedPDU.

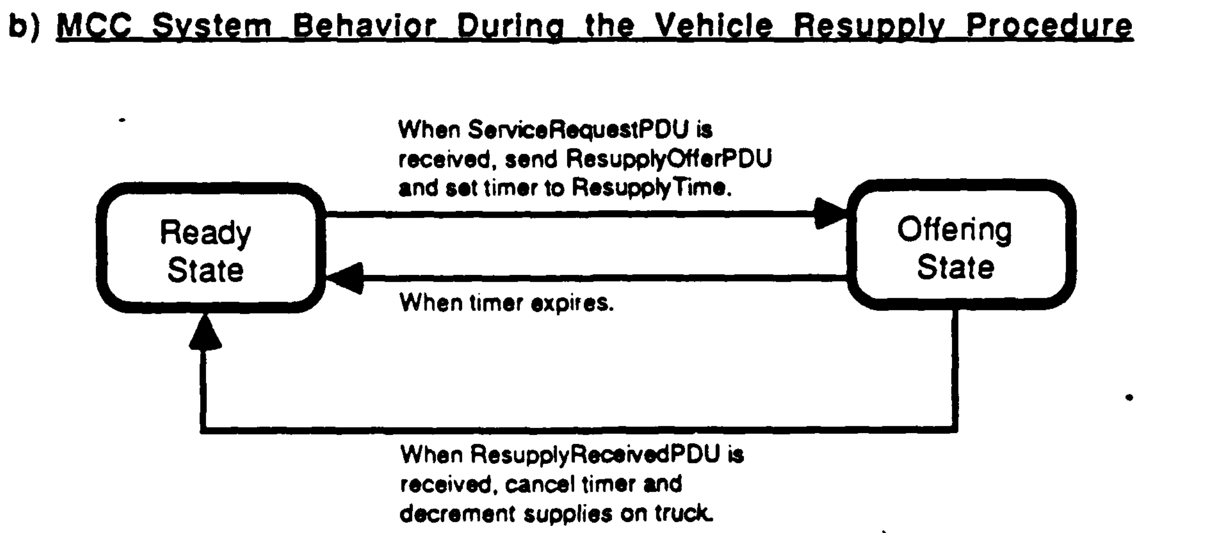

b) MCC System Behavior During the Vehicle Resupply Procedure
Ready State
Offering State
When ServiceRequestPDU is received, send ResupplyOfferPDU and set timer to ResupplyTime.
When timer expires.
When ResupplyReceivedPDU is received, cancel timer and decrement supplies on truck.

Figure 4. Vehicle Resupply Procedure states.

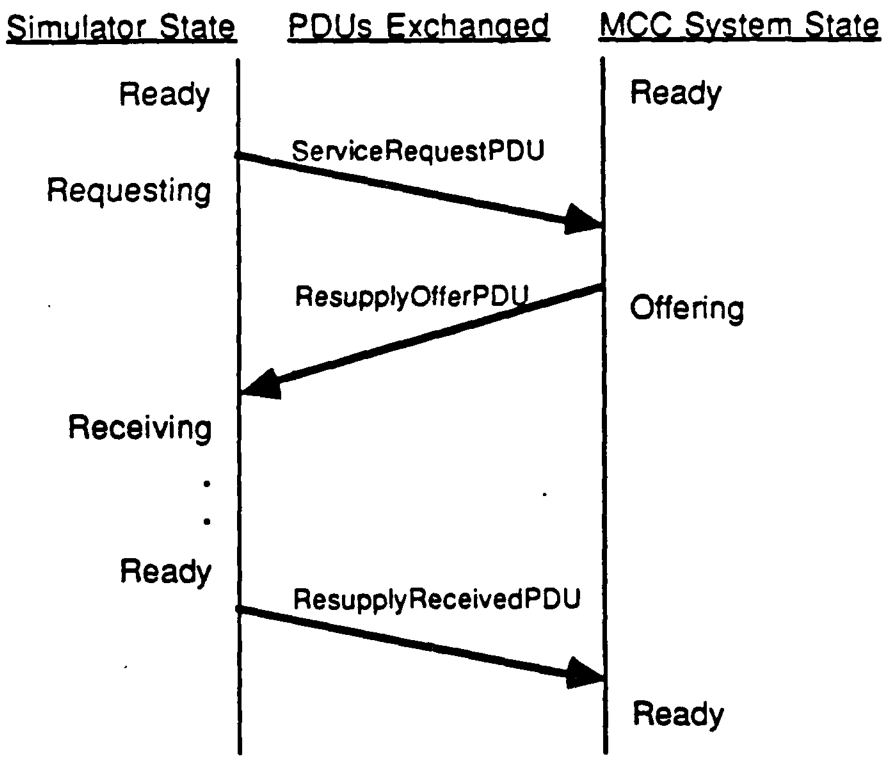


Figure 5. Vehicle Resupply Procedure protocol.

We now describe the procedure in greater detail.

The procedure is initiated by a simulator's broadcasting of a ServiceRequestPDU whenever it determines that the following conditions are all true:

- the simulator is in Ready State; and
- the combat vehicle needs ammunition, and there is a service truck of the type carrying ammunition within 20 meters; or the combat vehicle needs fuel, and there is a service truck of the type carrying fuel within 20 meters; and
- the service truck is not destroyed;
- the combat vehicle is stationary;
- the combat vehicle is not destroyed;

- the transfer of fuel or ammunition has been enabled by any necessary crew action appropriate to the simulator. For example, the transfer of ammunition to an M1 tank is enabled by the crew's setting of an ammunition resupply/distribution switch to its RECV position.

When these conditions are satisfied, the combat vehicle simulator broadcasts a ServiceRequestPDU containing the following information:

- the exercise identifier of the exercise in which the vehicle is participating
- the combat vehicle's vehicle identifier
- the service truck's vehicle identifier

Upon broadcasting the ServiceRequestPDU, the simulator changes state to Requesting State.

An MCC system receives the ServiceRequestPDU and, noticing that this PDU identifies its own service truck, responds by offering whatever supplies are currently loaded on that truck. In Figure 4, this is shown as a transition from Ready to Offering State. Meanwhile the simulator continues to broadcast the ServiceRequestPDU every ServiceRequestTime seconds until it receives an offer of supplies. The offer takes the form of a ResupplyOfferPDU, sent from the MCC system to the simulator, containing the following information:

- the exercise identifier
- the combat vehicle's vehicle identifier
- the service truck's vehicle identifier
- if the service truck is a fuel truck, the quantity of fuel currently on the truck and available to the combat vehicle
- if the service truck is an ammunition carrier, the quantities of each type of ammunition currently on the truck and available to the combat vehicle

Upon receiving the offer of supplies, the simulator changes state from Requesting to Receiving State. The simulator then has up to ResupplyTime seconds to acknowledge the receipt of those supplies by returning to the MCC system a ResupplyReceivedPDU that lists the exact supplies accepted by the combat vehicle. The simulator needn't accept all of the supplies offered but can indicate, in its receipt, just how much it did take. After delaying for up to ResupplyTime seconds, the simulator sends its ResupplyReceivedPDU and returns to Ready State. When the MCC system receives the ResupplyReceivedPDU it also returns to Ready State, and the procedure is complete.

The time required for the simulator to return the ResupplyReceivedPDU depends on the rate at which the combat vehicle it simulates can load supplies. For example, an M1 tank simulator accepting main gun ammunition from a HEMTT will acknowledge receipt of a single round after 40 seconds, as one round every 40 seconds is the simulated rate of resupply for the M1 tank.

The ResupplyReceivedPDU contains the following information:

- the exercise identifier
- the combat vehicle's vehicle identifier
- the service truck's vehicle identifier
- if the combat vehicle accepted fuel, the quantity of fuel taken
- if the combat vehicle accepted ammunition, the quantities of each type of ammunition taken

Throughout the Vehicle Resupply Procedure the simulator continues to monitor the conditions required for resupply. If any of these conditions becomes false the simulator can abort the resupply either by sending a ResupplyReceivedPDU for any supplies taken up to the time of the abort (such as a partial load of fuel), or by simply terminating the procedure and sending no ResupplyReceivedPDU at all. In the latter case, the MCC system will wait for ResupplyTime seconds, and, receiving no acknowledgement for the supplies it offered, correctly assume that no supplies were taken.

## 5.8 Vehicle Repair Procedure

The Vehicle Repair Procedure allows an MCC system to carry out simulated repairs on a combat vehicle. The repairs are performed by a maintenance team simulated by the MCC system under the direction of maintenance personnel. The maintenance team is represented as a service truck in the simulation.

Each simulator is responsible for identifying times when repairs could be performed on its combat vehicle. It does this by checking, every ServiceRequestTime seconds, whether the following conditions are all true:

- a service truck of the type used by maintenance teams is within 20 meters of the combat vehicle;
- the service truck is not damaged;

- the combat vehicle is stationary; and
- the combat vehicle is not destroyed.

When these conditions are satisfied, the combat vehicle simulator broadcasts a ServiceRequestPDU containing the following information:

- the exercise identifier of the exercise in which the vehicle is participating
- the combat vehicle's vehicle identifier
- the service truck's vehicle identifier

An MCC system receiving a ServiceRequestPDU identifying a service truck it simulates can allow maintenance personnel to start a repair on the combat vehicle mentioned in the PDU. As long as the MCC system continues to receive the ServiceRequestPDUs about every ServiceRequestTime seconds, it can allow the repair process to continue. However if the MCC system stops receiving these PDUs it must assume that the conditions listed above are no longer all true, and it must therefore abort the repair.

If the simulated repair runs to completion in the MCC system, that MCC system sends a RepairPDU to the combat vehicle simulator notifying it that the repair is complete. The RepairPDU contains:

- the exercise identifier
- the vehicle identifier of the combat vehicle repaired
- a code specifying the type of repair completed (e.g., battery replaced, oil filter replaced, etc.)

The simulator acknowledges receipt of the RepairPDU by returning a RepairedPDU. (This acknowledgement indicates that the simulator has effected the specified repair; not that the repair was appropriate or that the combat vehicle has become operational as a result of the repair.) A RepairedPDU contains the following information:

- the exercise identifier
- the vehicle identifier of the combat vehicle repaired

If the MCC system receives no response to its RepairPDU within RepairTime seconds, it assumes that the PDU was lost and resends it. It resends the PDU up to RepairCount times before giving up and reporting to the installation staff a failure of the network or of the combat vehicle simulator.

## 5.9 Collision Procedure

The Collision Procedure ensures that when two vehicles collide, both are aware of the collision. Only moving vehicles, which are simulated by combat vehicle simulators, can cause collisions. The computer controlled vehicles simulated by MCC systems are prevented from colliding into other vehicles.

The simulator of a combat vehicle involved in a collision with another vehicle will, upon detecting the collision, broadcast a CollisionPDU containing the following information:

- the exercise identifier of the exercise in which the simulator is participating
- the vehicle identifier of the combat vehicle whose simulator is sending the PDU
- the vehicle identifier of the other vehicle involved in the collision

## 6. REFERENCES

[1] "The Ethernet: A Local Area Network: Data Link Layer and Physical Layer Specifications". Digital Equipment Corporation, Intel Corporation, and Xerox Corporation; Version 2.0; November, 1982.

[2] The Institute of Electrical and Electronics Engineers, Inc. "IEEE Standards for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications". The Institute of Electrical and Electronics Engineers, Inc. New York, New York, 1985.

# APPENDIX A: SUMMARY OF TIMERS AND COUNTERS

The following timers are used in defining the SIMNET protocol:

| | |
|---|---|
| EquipStatusTime | the period with which MCC systems and combat vehicle simulators send EquipStatusPDUs to an NOM system. Duration: 30 seconds. |
| ActivateTime | how long an MCC system waits for a response to an ActivatePDU. Duration: 5 seconds. |
| DeactivateTime | how lcng an MCC system waits between sending DeactivatePDUs when attempting to deactivate a simulator, or when notifying that a computer controlled vehicle has disappeared. Duration: 10 seconds. |
| VehicleAppearanceTime | the maximum time between successive VehicleAppearancePDUs transmitted for any simulated vehicle. Duration: 5 seconds. |
| VehicleStatusTime | the maximum time between successive VehicleStatusPDUs transmitted by any combat vehicle simulator. Duration: 30 seconds. |
| ServiceRequestTime | the period between successive ServiceRequestPDUs broadcast by a combat vehicle simulator. Duration: 5 seconds. |
| ResupplyTime | how long an MCC system waits for a response to a ResupplyOfferPDU. Duration: 60 seconds. |
| RepairTime | how long an MCC system waits for a response to a RepairPDU. Duration: 10 seconds. |

The following counters are used in defining the SIMNET protocol:

| | |
|---|---|
| ActivateCount | how many times an MCC system repeats an ActivatePDU. Value: 3. |
| DeactivateCount | how many times an MCC system repeats a DeactivatePDU. Value: 3. |
| RepairCount | how many times an MCC system repeats a RepairPDU. Value: 3. |