INTERFACING OF THE SILICON GRAPHICS NETWORKABLE FLIGHT SIMULATOR WITH SIMNET

Technical Report

PUBLICATION NUMBER IST-TR-89-1

JORGE CADIZ, RUEY OUYANG, JACK THOMPSON

NETWORKING AND COMMUNICATIONS TECHNOLOGY LABORATORY

The Institute for Simulation and Training
12424 Research Parkway, Orlando, FL 32826

October 5, 1989

University of Central Florida

REPORT DOCUMENTATION PAGE

2a SECURITY CLASSIFICATION AUTHORITY N/A	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) IST-TR-89-1	5 MONITORING ORGANIZATION REPORT NUMBER(S) IST-TR-89-1
6a NAME OF PERFORMING ORGANIZATION Institute for Simulation and Training 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Project Manager for Training Devices
6c ADDRESS (City, State, and ZIP Code) 12424 Research Parkway, Suite 300 Orlando, Fl 32826	7b ADDRESS (City, State, and ZIP Code) 12350 Research Parkway Orlando, Fl 32826
8a NAME OF FUNDING/SPONSORING ORGANIZATION DARPA / TTO 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N61339-89-C-0043
8c ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, VA 22209	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Interfacing of the Silicon Graphics Networkable Flight Simulator with SIMNET

12. PERSONAL AUTHOR(S)

Cadiz, Jorge; Ouyang, Ruey; Thompson, Jack

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM 4/89 TO 10/89	October 5, 1989	24

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Network Protocol Translation, Simulator Networking, Non-homogenous Simulation Networking

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

One of the efforts of the Network & Communications Technology Research Lab is to interface dissimilar simulators in the "SIMNET World". As part of this effort we have networked (via ETHERNET) a flight simulation program running on a Silicon Graphics 4D70GT Workstation with the SIMNET M1 modules that are located in the IST Research Laboratory. Various steps were taken to achieve this task, and the keypoints are documented herein.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Mike Garnsey	22b. TELEPHONE (Include Area Code): 407-380-4816 / 22c. OFFICE SYMBOL: AMCPM-TND-EN

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

INTERFACING OF THE SILICON GRAPHICS NETWORKABLE FLIGHT SIMULATOR WITH SIMNET

Technical Report

PUBLICATION NUMBER IST-TR-89-1

Introduction

One of the efforts of the Network & Communications Technology Research Lab is to interface dissimilar simulators in the "SIMNET World". As part of this effort we have networked (via ETHERNET) a flight simulation program running on a *Silicon Graphics 4D70GT Workstation* with the SIMNET M1 modules that are located in the IST Research Laboratory. Various steps were taken to achieve this task, and the keypoints are documented herein.

SIMNET Vehicle Appearance Protocol Data Unit (PDU)

Our first task was to gain a thorough understanding of the SIMNET *Vehicle Appearance PDU*. PDU's are data units which are exchanged between simulators. The Vehicle Appearance PDU's describe the visual appearance of a vehicle at the moment of broadcast. Each PDU is composed of a discrete number of fields. The fields for the SIMNET PDU's are shown below (byte length in brackets):

- SIMNET Package Header
  - ethernet destination [6]
  - ethernet source [6]
  - ethernet packet type [2]
  - user data length [2]
- Vehicle Appearance data fields [1512 - header size] (see appendix for description of Vehicle Appearance PDU fields)

For a more detailed description of the SIMNET Protocol Data Elements refer to BBN Report No. 7102, The SIMNET Network and Protocols, section 5.

Silicon Graphics (SG) "Dogfight" Flight Data

In this section we discuss certain characteristics of interest of the SG Networkable "Dogfight" Flight Simulation. These characteristics are the coordinate system, initialization, and vehicle orientation (attitude).

- **Coordinate System**: The origin of the SG Terrain Data Base is in the center of the runway. The X-axis points North, the Z-axis point West, and the Y-axis points opposite the direction of gravity (see Figure 1).

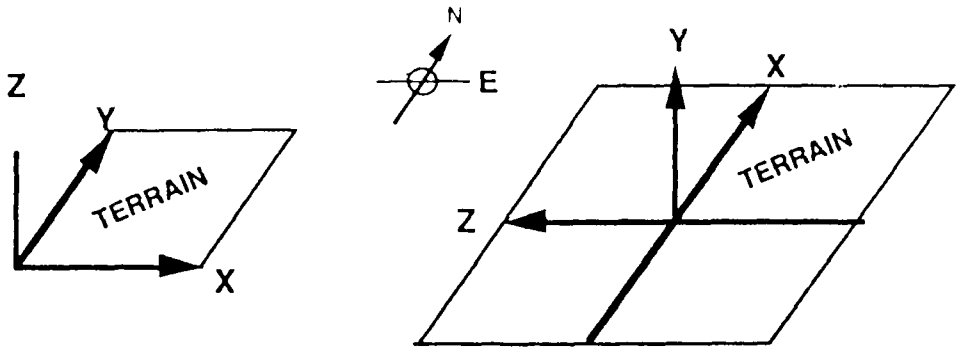

Figure 1. SIMNET & Silicon Graphics World Coordinate Systems

- **Initialization**: The SG airplane initializes on the taxi-way at X = 850, Y = 0, and Z = -2050 (feet). Air speed is zero.

- **SG Attitude Data**: In the SG simulation, the pitch, roll, and yaw of the vehicle are expressed as elevation, twist, and azimuth respectively. The conventions for these attitude angles are given in Figure 2 below.

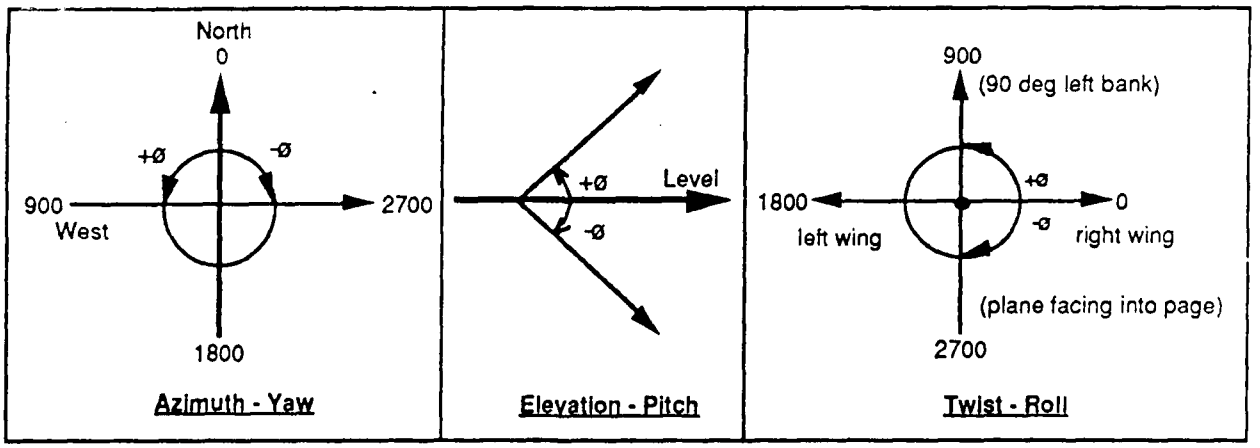

Figure 2. Silicon Graphics Attitude Data

SIMNET Characteristic Data:
The values of the SIMNET data base range from zero to 75,000 in the X and Y directions. All SIMNET distance values are in meters. The conventions for vehicle pitch, roll, and yaw are illustrated in Figure 3.

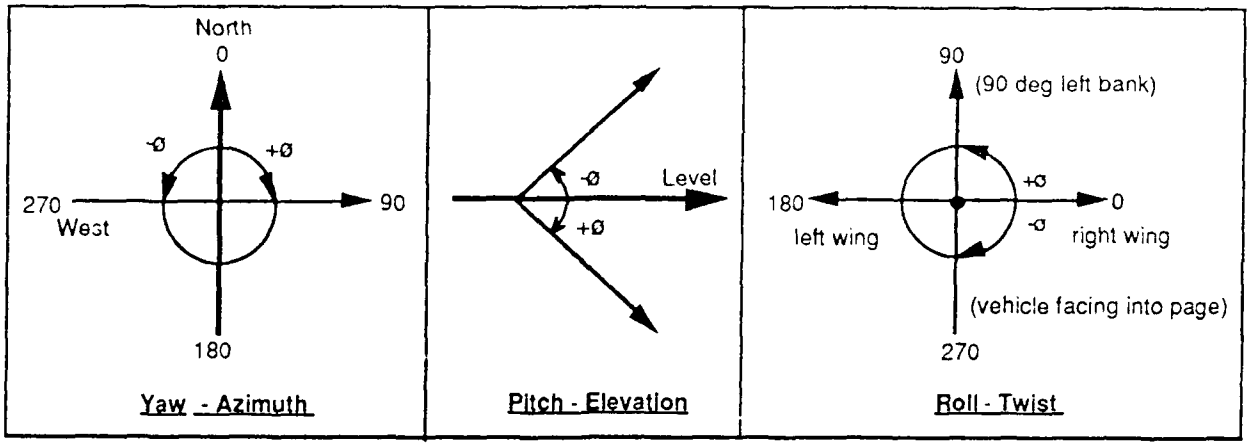

Figure 3. SIMNET Attitude Data

Silicon Graphics to SIMNET Conversion
The SG "Dogfight" PDU contains similar types of information as the SIMNET PDU. However, some calculations and transformations are required to convert the SG PDU information into SIMNET PDU format. Unlike SIMNET, the SG uses the angles of azimuth, elevation, and twist (yaw, pitch, roll) to describe the orientation of the airplane in free space. In order to interface the SG with the SIMNET, we needed a program which calculates the SIMNET required nine element (3 x 3) rotation matrix given these angles. The program which performs this calculation is provided as an attachment to this document.

The SG simulation uses feet as its distance unit of measure, whereas, SIMNET uses meters. Therefore, the conversion of feet to meters was another necessary transformation.

Translation of Silicon Graphics "Dogfight" PDU
We were able to translate the SG "Dogfight" PDU's by "stealing" packets from the ETHERNET via a PC/AT with an Excelan 205E ETHERNET board installed. The software written to perform this operation uses UDP/IP to capture the SG broadcast packets and saves them to a disk file. Information contained in the SG PDU was reformatted or translated into SIMNET type packets, and then retransmitted over the ETHERNET to the SIMNET M1 tank modules.

CONCLUSION
As a result from this task we were able to prove our capabilities of building a protocol translator using the resources that exist in the IST laboratory. These projects use a PC/AT-based platform that will translate or manipulate the ETHERNET packets allowing us to experiment with new implementations of SIMNET technology.

APPENDIX A
SIMNET Vehicle Appearance PDU

PDU Header							
Simulation Protoocol Header							
version	length	protocol	kind	exercise ID	padding	vehicle ID	
bits 1 thru 4	bits 5 thru 16	octet 2	octet 3	octet 4	octet 5	octet 6	octet 7

role	battalion	company	bumper	class	appearance	
octet 8	octet 9	octet 10	octet 11	octet 12	octet 13	octet 14

rotation				location			
octet 15	octet 16	octet 17	octet 18	octet 19	octet 20	octet 21	octet 22

grid		engine speed		padding
octet 23	octet 24	octet 25	octet 26	octet 27

STATIC CLASS VEHICLES

padding
octet 28

SIMPLE CLASS VEHICLES

padding	velocity			
octet 28	octet 29	octet 30	octet 31	octet 32

TANK CLASS VEHICLES

padding	velocity				turret azimuth		gun elevation
octet 28	octet 29	octet 30	octet 31	octet 32	octet 33	octet 34	octet 35

gun elevation
octet 36

APPENDIX B

Silicon Graphics "Dogfight" to SIMNET Interfacing Software

```
/*****************************************************************************
 * Name:
 *
 *    This file is the header file for the airpalne running on
 *    the SiliconGraphics
 *
 *****************************************************************************/

#define NAME_LENGTH 15
/*
#define MYPLANEID 16
*/
#define ADJUSTX -850
#define ADJUSTZ 2050
#define AIRPORTX 40000.0
#define AIRPORTY 220.0
#define AIRPORTZ 30000.0
/*
#define F2M 3.281
*/
#define F2M 5.0

struct plane {
    long planeid;

    char version;                       /* flight version */
    char cmd;                           /* type of packet */
    short type;                         /* plane type */
    short alive;                        /* alive */
    char myname[NAME_LENGTH+1];

    unsigned short status;
    unsigned short won;                 /* for msgs these 2 shorts */
    unsigned short lost;                /* hold the plane id */

    float x;                            /* plane position */
    float y;
    float z;
    short azimuth;
    short elevation;
    short twist;

    short mstatus;                      /* missile data */
    float mx;
    float my;
    float mz;
    float last_mx;
    float last_my;
    float last_mz;
    long kill;
    float tps;
    int airspeed;
    int thrust;
    short wheels;                       /* wheel position */
    short elevator;                     /* elevator position */
    char mtype;
};

struct plane plane;
short port=0x140a;                      /* port address for udp/ip connection */
```

```
/***********************************************************************
[illegible]

*   This file contains the c code to handle the airplane flying on the SG

***********************************************************************/

/* Initialize a synchronous/blocking udp/ip connection for input */
sginitin()

    /* Check that the driver is loaded, and get our own ethernet MAC
       address from the EXOS board */
    if  (!loaded()) errexit("driver NOT loaded");
    if  (ipinfo(&opt) < 0) errexit("could not get own ethernet MAC address");
    memcpy(my_addr, opt.xo_eaddr, sizeof(my_addr));

    /* Display my address */
    fprintf(stderr, "my addr = %02x-%02x-%02x-%02x-%02x-%02x\n",
                    my_addr[0], my_addr[1], my_addr[2],
                    my_addr[3], my_addr[4], my_addr[5]);

    /* Open input disk file */
    diskfd = open(inputfile, FILEOFLAG, FILEPMODE);
    if  (diskfd < 0) errexit("cannot open diskfile");
    fprintf(stderr, "disk file fd = %d\n", diskfd);

    /* UDP/IP specification */
    send_socket_sg.sin_port = htons(port);
    send_socket_sg.sin_addr.s_addr = 0x00000000;
    recv_socket_sg.sin_port = htons(port);
    recv_socket_sg.sin_addr.s_addr = 0xffffffff;

    /* Make a udp socket call */
    if  ((netfdsg = socket(SOCK_DGRAM, (struct sockproto *) 0,
                           &send_socket_sg, 0)) < 0) {
        fprintf(stderr, "ERRNO %d\n", errno);
        errexit("socket");
    }
    fprintf(stderr, "sg socket fd = %d\n", netfdsg);
    return(0);

/* Read synchronous/blocking udp/ip packet */
sgreadin()

    int cnt;

    /* if  ((cnt = soreceive(netfdsg, &recv_socket_sg, buf, sizeof(buf))) < 0)
        errexit("soreceive");
    fprintf(stderr, "read %d bytes from sg\n", cnt); */
    if  ((cnt = read(diskfd, buf, 100)) < 0)
        errexit("read");
    /* fprintf(stderr, "read %d bytes from disk\n", cnt); */
    return(cnt);

/* Close connection */
sgfiniin()

    soclose(netfdsg);
```

```
    int i, j;
    union {
        char  *tmpc;
        float *tmpf;
    } tmp;
    union {
        char  *tmpc;
        short *tmps;
    } tmps;

    tmp.tmpf = &plane.x;
    swap4(tmp.tmpc);
    tmp.tmpf = &plane.y;
    swap4(tmp.tmpc);
    tmp.tmpf = &plane.z;
    swap4(tmp.tmpc);
    tmps.tmps = &plane.azimuth;
    swap2(tmps.tmpc);
    tmps.tmps = &plane.elevation;
    swap2(tmps.tmpc);
    tmps.tmps = &plane.twist;
    swap2(tmps.tmpc);
}

/* Host order to network order transform */
hton_flight ()
{
    int i, j;
    union {
        char  *tmpc;
        float *tmpf;
    } tmp;
    union {
        char  *tmpc;
        short *tmps;
    } tmps;

    tmp.tmpf = &plane.x;
    swap4(tmp.tmpc);
    tmp.tmpf = &plane.y;
    swap4(tmp.tmpc);
    tmp.tmpf = &plane.z;
    swap4(tmp.tmpc);
    tmps.tmps = &plane.azimuth;
    swap2(tmps.tmpc);
    tmps.tmps = &plane.elevation;
    swap2(tmps.tmpc);
    tmps.tmps = &plane.twist;
    swap2(tmps.tmpc);
}

/* This subroutine is here for documentation, it is on simnet.ccd */
/*
swap4(char *ptr)
{
```

```
    tmp = *(ptr+3);
    *ptr = *(ptr+3);
    *(ptr+3) = tmp;
    tmp = *(ptr+1);
    *(ptr+1) = *(ptr+2);
    *(ptr+2) = tmp;
}
*/

/* This subroutine is here for documentation, it is on simnet.ccd */
/*
swap2(char *ptr)
{
    char tmp;

    tmp = *ptr;
    *ptr = *(ptr+1);
    *(ptr+1) = tmp;
}
*/

display_plane()
{
    fprintf(stderr, "plane id %ld\n", plane.planeid);
    fprintf(stderr, "version %c\t cmd %c\t type %d\t alive %d\t myname %s\n",
                    plane.version, plane.cmd, plane.type, plane.alive,
                    plane.myname);
    fprintf(stderr, "status %ud\t won %ud\t lost %ud\n",plane.x,plane.y,
                    plane.z);
    fprintf(stderr, "x %f\t y %f\t z %f\n",plane.x,plane.y,plane.z);
    fprintf(stderr, "azimuth %d\t elevation %d\t twist %d\n",plane.azimuth,
                    plane.elevation,plane.twist);
    fprintf(stderr, "mstatus %d\t mx %f\t my %f\t mz %f\n",plane.mstatus,
                    plane.mx,plane.my,plane.mz);
    fprintf(stderr, "last_mx %f\t last_my %f\t last_mz %f\n", plane.last_mx,
                    plane.last_my,plane.last_mz);
    fprintf(stderr, "kill %id\t tps %f\n", plane.kill, plane.tps);
    fprintf(stderr, "air speed %d\t thrust %d\n",plane.airspeed,
                    plane.thrust);
    fprintf(stderr, "wheels %d\t elevator %d\t mtype %c\n",plane.wheels,
                    plane.elevator, plane.mtype);
}
```

```
/*****************************************************************************
 * netio.c:

 *      This file contains the C code for the simnet M1 tank simulator.

 *****************************************************************************/

/* Initialize the synchronous/non-blocking link-level socket connection */
netinit()
{
    int rc, on=1;

    /* Check that the driver is loaded, and get our own ethernet MAC
       address from the EXOS board */
    if  (!loaded()) errexit("driver NOT loaded");
    if  (ipinfo(&opt) < 0) errexit("could not get own ethernet MAC address");
    memcpy(my_addr, opt.xo_eaddr, sizeof(my_addr));

    /* Display my address */
    fprintf(stderr, "my addr = %02x-%02x-%02x-%02x-%02x-%02x\n",
                    my_addr[0], my_addr[1], my_addr[2],
                    my_addr[3], my_addr[4], my_addr[5]);

    /* Initialize the simnet receiver/sender socket type */
    recv_socket.sl_types[0] = ETYPE;

    /* Make a link level socket call */
    if  ((netfd=socket(SOCK_ETH, (struct sockproto *)0, &recv_socket, 0)) < 0) {
        if  (errno == EACCES)
            errexit ("link-level access must be enabled with -l option on netloa
        else errexit("cannot create socket");
    }
    fprintf(stderr, "socket fd = %d\n", netfd);

    /* Synchronous/non blocking mode */
    soioctl(netfd, SIOCSLINGER, &timelimit);
    rc = soioctl(netfd, FIONBIO, &on);
    if  (rc < 0) {
        experror ("soioctl(...FIONBIO, &on)");
        return(-1);
    }
    return(0);
}

/* Read synchronous/non blocking mode packet */
/* netread (struct ether buf) */
netread ()
{
    int cnt;

    cnt = soreceive(netfd, (struct sockaddr *)0, &ether_buf, MAXPKTSIZE);
    if  ((cnt < 0) && (errno == EWOULDBLOCK))
        ;    /* No network data */
    else
        if  (cnt < 0) experror("soreceive read error"); /* Error condition */
    return (cnt);
}

/* Write synchronous/non blocking mode packet */
/* netwrite (struct ether *buf) */
```

```
netwrite ()
{
    int cnt, netcnt;

    datalength.p_datalength = ntohs (ether_buf.simnet_data.e_datalength);
    cnt = datalength.i_datalength.length;
    netcnt = sosend(netfd, (struct sockaddr *)0, &ether_buf,cnt + HEADER_SIZE);
    if  ((netcnt < 0) && (errno == EWOULDBLOCK)) netcnt = 0;
    if  (netcnt < 0)
        errexit("sosend write error");
    else
        if  ((netcnt >= 0) && (netcnt < cnt))
            fprintf(stderr, "sosend : some data has been lost\n\007\007");
}

/* Close synchronous/non blocking socket connection */
netfini ()
{
    int off = 0;

    if  (netfd >= 0) {
        fprintf(stderr, "Please wait up to %d seconds for completion\n",
                        timelimit);
    soioctl(netfd, FIONBIO, &off);
    soclose(netfd);
    netfd = -1;
    }
}

/* Network order to host order transform, not all of the data field are included
   yet.  Add more statements if needed and modify the hton_simnet() too */
/* ntoh_simnet (PDU buf) */
ntoh_simnet ()
{
    int i, j;
    union {
        char  *tmpc;
        unsigned short *tmpui;
    } tmpui;
    union {
        char  *tmpc;
        float *tmpf;
    } tmp;

    tmp.tmpf = &pdu_buf.VAPDU.VADATA.location[0];
    swap4(tmp.tmpc);
    tmp.tmpf = &pdu_buf.VAPDU.VADATA.location[1];
    swap4(tmp.tmpc);
    tmp.tmpf = &pdu_buf.VAPDU.VADATA.location[2];
    swap4(tmp.tmpc);
    tmpui.tmpui = &pdu_buf.VAPDU.VADATA.hdr.vehicleID;
    swap2(tmpui.tmpc);
    for (i=0; i<=2; i++)
        for (j=0; j<=2; j++) {
            tmp.tmpf = &pdu_buf.VAPDU.VADATA.rotation[i] [j];
            swap4(tmp.tmpc);
    }
    return(pdu_buf.ANYPDU.ANYHDR.kind);
}
```

```
/* host order to network order transform, not all of the data field are include
   yet.  Add more statements if needed and modify the ntoh_simnet() too */
/* hton_simnet (struct PDU buf) */
hton_simnet ()
{
    int i, j;
    union {
        char  *tmpc;
        unsigned short *tmpui;
    } tmpui;
    union {
        char  *tmpc;
        float *tmpf;
    } tmp;

    tmp.tmpf = &pdu_buf.VAPDU.VADATA.location[0];
    swap4(tmp.tmpc);
    tmp.tmpf = &pdu_buf.VAPDU.VADATA.location[1];
    swap4(tmp.tmpc);
    tmp.tmpf = &pdu_buf.VAPDU.VADATA.location[2];
    swap4(tmp.tmpc);
    tmpui.tmpui = &pdu_buf.VAPDU.VADATA.hdr.vehicleID;
    swap2(tmpui.tmpc);
    for (i=0; i<=2; i++)
        for (j=0; j<=2; j++) {
            tmp.tmpf = &pdu_buf.VAPDU.VADATA.rotation[i] [j];
            swap4(tmp.tmpc);
    }
    return(0);
}

/* This subroutine does the same work as ntohl(), htonl(). */
swap4(char *ptr)
{
    char tmp;

    tmp = *ptr;
    *ptr = *(ptr+3);
    *(ptr+3) = tmp;
    tmp = *(ptr+1);
    *(ptr+1) = *(ptr+2);
    *(ptr+2) = tmp;
}

/* This subroutine does the same work as ntohs(), htons(). */
swap2(char *ptr)
{
    char tmp;

    tmp = *ptr;
    *ptr = *(ptr+1);
    *(ptr+1) = tmp;
}

/* This subroutine is for debugging purpose only, it will DUMP the content of a
   link level packet in hexdecimal*/
/* dump_ether (struct ether ether_buf) */
dump_ether ()
{
    int i, j, netcnt;
```

```
    fprintf(stderr,"ETHER content \n");
    datalength.p_datalength = ntohs (ether_buf.simnet_data.e_datalength);
    fprintf(stderr,"Source addr      : %2x-%2x-%2x-%2x-%2x-%2x\n",
        ether_buf.e_shost [0], ether_buf.e_shost [1], ether_buf.e_shost [2],
        ether_buf.e_shost [3], ether_buf.e_shost [4], ether_buf.e_shost [5]);
    fprintf(stderr,"Destination addr : %2x-%2x-%2x-%2x-%2x-%2x\n",
        ether_buf.e_dhost [0], ether_buf.e_dhost [1], ether_buf.e_dhost [2],
        ether_buf.e_dhost [3], ether_buf.e_dhost [4], ether_buf.e_dhost [5]);
    fprintf(stderr,"%2x ",datalength.p_datalength);
    netcnt = datalength.i_datalength.length;
    for (i=0, j=3; i<(netcnt-HEADER_SIZE-2); i++, j++) {
        fprintf(stderr,"%2x ", ether_buf.simnet_data.e_data[i]);
        if  (j >= 17) {
            j=0;
            fprintf(stderr,"\n");
        }
    }
    fprintf(stderr,"\n");
}

/* This subroutine is for debugging purpose only, it will DUMP the content of a
   pdu packet in hexdecimal*/
dump_pdu ()
{
    int i, j, netcnt;

    fprintf(stderr,"PDU content\n");
    datalength.p_datalength = ntohs (ether_buf.simnet_data.e_datalength);
    netcnt = datalength.i_datalength.length;
    for (i=0, j=1; i<(netcnt-HEADER_SIZE-2); i++, j++) {
        fprintf(stderr,"%2x ", pdu_buf.DATAONLYPDU.DATAONLY[i]);
        if  (j >= 17) {
            j=0;
            fprintf(stderr,"\n");
        }
    }
    fprintf(stderr,"\n");
}

/* This subroutine is for debugging purpose only, it will DISPLAY the content of
   a pdu packet */
display_pdu ()
{
    int i, j;
    union {
        char  *tmpc;
        float *tmpf;
    } tmp;

    fprintf(stderr, "Rotation\n");
    for (i=0; i<=2; i++)
        for (j=0; j<=2; j++)
            fprintf(stderr,"%d %d %lf\n",i,j,pdu_buf.VAPDU.VADATA.rotation[i][j]
    fprintf(stderr, "Location\n");
    fprintf(stderr, "%lf\n",pdu_buf.VAPDU.VADATA.location[0]);
    fprintf(stderr, "%lf\n",pdu_buf.VAPDU.VADATA.location[1]);
    fprintf(stderr, "%lf\n",pdu_buf.VAPDU.VADATA.location[2]);
    fprintf(stderr, "%u\n",pdu_buf.VAPDU.VADATA.hdr.vehicleID);
}
```

```
/**************************************************************************
 * flight.c
 *
 * This program displays the airplane controled by the SiliconGraphics on
 * the simnet.
 *
 *     simnet: Link Level Raw Ethernet Packets / Synchronous Non-blocking
 *
 *     SiliconGraphics: Synchronous-blocking UDP/IP or
 *                      (disk file)
 *
 **************************************************************************/
#include <sys/extypes.h>

#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <sys/exerrno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <signal.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/exosopt.h>
#include <sys/exos.h>
#include <ex_ioctl.h>
#include <sys/soioctl.h>
#include <sys/dcb.h>
#include "..\simnet.h\simnet2.h"
#include "..\flight.h\flight.h"

struct sockaddr_link recv_socket = { AF_ETYPEFILTER };
struct sockaddr_link send_socket = { AF_ETYPEFILTER };
struct sockaddr_in recv_socket_sg = { AF_INET };
struct sockaddr_in send_socket_sg = { AF_INET };

#define FILEOFLAG (O_RDONLY | O_BINARY)
#define FILEPMODE (0)

#define PI 3.14159

extern int errno;
extern int break_enabled;
extern int abort_op;

int     diskfd = -1;            /* disk file */
int     netfd = 1;              /* simnet file */
int     netfdsg = -1;           /* udp/ip file */
int     timelimit = 30;
char    *inputfile;

char    SENDIT;
char    buf[1024];

int  break_handler();

main (argc, argv)
char **argv;
```

```
    int an, i, j, pktcnt, netcnt;

    signal(SIGINT, break_handler);
    break_enabled = 1;
    inputfile = argv[1];

    sginitin();

    netinit();

    /* Capture a simnet packet first, so we don't have to fill all of the data
           field */
    fprintf(stderr, "wait for simnet\n");
    while(1) {
        /* netcnt=netread(inbuf); */
        netcnt=netread();
        datalength.p_datalength= ntohs (ether_buf.simnet_data.e_datalength);
        netcnt=datalength.i_datalength.length + HEADER_SIZE;
        memcpy (&pdu_buf, &ether_buf.simnet_data, netcnt - HEADER_SIZE);
        pdukind = ntoh_simnet();
        if  (pdukind == vehicleAppearancePDUKind) {
            SENDIT = ' ';
            if  (ether_buf.e_shost [5] == TANKA)
                SENDIT = 'A';
            if  (ether_buf.e_shost [5] == TANKB)
                SENDIT = 'B';
        }
        if  ((SENDIT == 'A') || (SENDIT == 'B')) break;
    }

    fprintf(stderr, "Got a vehicle appearance packet from tank %c\n", SENDIT);
    pdu_buf.VAPDU.VADATA.hdr.vehicleID = MYTANKID;
    pdu_buf.VAPDU.VADATA.appearance.vehKindMask  = A10;
    memcpy (ether_buf.e_shost, my_addr, sizeof(my_addr));

    while (1) {
        netcnt = sgreadin();
        if  (netcnt <= 0) break;
        memcpy(&plane, buf, netcnt);
        ntoh_flight();

        pdu_buf.VAPDU.VADATA.location[0] =
            AIRPORTX + ((plane.x + ADJUSTX)/F2M);
        pdu_buf.VAPDU.VADATA.location[1] =
            AIRPORTZ - ((plane.z + ADJUSTZ)/F2M);
        pdu_buf.VAPDU.VADATA.location[2] = AIRPORTY + (plane.y/F2M);
        calrotation();
        hton_simnet();
        memcpy (&ether_buf.simnet_data, &pdu_buf, netcnt - HEADER_SIZE);
        netwrite();
    }
    fprintf (stderr, "End of input sg packet\n");
    close(diskfd);
    sgfiniin();
    netfini();
}

errexit(errstring)
char *errstring;
```

```
        if (errno) experror(errstring);
        else fprintf(stderr, "%s: diskfile : filename\n", errstring);
        close(diskfd);
        soclose(netfdsg);
        netfini();
        exit(1);
}

break_handler()         /* break handler ... control-break or control-c */
{
        static int break_count = 0;

        if  (++break_count == 1) {
            /* first time, just try to stop current network operation */
            abort_op = 1;
            signal(SIGINT, break_handler);      /* reset trap */
            return;
        }
        else {
            /* second time, try to clean up, then quit */
            errexit("user abort");
        }
}

pinfo(optp)
struct exosopt *optp;
{
        /* note that this routine will not return valid results
         * if used with a pre-3.3 driver, which interpreted the
         * board memory address as absolute, rather than relative
         * to the beginning of the data segment
         */
        long    optaddress = 0;         /* location of options */
        int     id;

        if  ((id = brdopen(0, 1)) < 0) {
            experror("brdopen");
            return(-1);
        }
        if  (brdioctl(id, BRDADDR, &optaddress) < 0) {
            experror("brdioctl(,BRDADDR,...)");
            return(-1);
        }
        if  (brdread(id, optp, sizeof(struct exosopt)) < 0) {
            experror("brdread");
            return(-1);
        }
        brdclose(id);
        return 0;
}

#include "..\simnet.h\simnet.ccd"
#include "..\flight.h\flight.ccd"

/*  This subroutine computes the rotation matrix (3x3) for the SIMNET PDU's */
/*  given the pitch, roll and yaw of the vehicle. */

alrotation()
```

```
    int i,j,k;
    float R,P,Y;
    float RC,RS,PC,PS,YC,YS;
    float A [3] [3];
    float z [3] [3];
    float x [3] [3];
    float y [3] [3];

    /* In Silicon Graphics DogFight: Roll=Twist; Pitch=Elevation; Yaw=Azimith */

    R=(plane.twist/10*PI)/180;
    P=-(plane.elevation/10*PI)/180;
    Y=-(plane.azimuth/10*PI)/180;

    RC=cos(R);
    RS=sin(R);

    PC=cos(P);
    PS=sin(P);
    YC=cos(Y);
    YS=sin(Y);

    z[0] [0]=YC;
    z[0] [1]=-YS;
    z[0] [2]=0;
    z[1] [0]=YS;
    z[1] [1]=YC;
    z[1] [2]=0;
    z[2] [0]=0;
    z[2] [1]=0;
    z[2] [2]=1;

    x[0] [0]=1;
    x[0] [1]=0;
    x[0] [2]=0;
    x[1] [0]=0;
    x[1] [1]=PC;
    x[1] [2]=-PS;
    x[2] [0]=0;
    x[2] [1]=PS;
    x[2] [2]=PC;

    y[0] [0]=RC;
    y[0] [1]=0;
    y[0] [2]=RS;
    y[1] [0]=0;
    y[1] [1]=1;
    y[1] [2]=0;
    y[2] [0]=-RS;
    y[2] [1]=0;
    y[2] [2]=RC;

    for (i=0; i<=2; i++) {
        for (j=0; j<=2; j++) {
            A [i][j]=0;
            for (k=0; k<=2; k++)
                A[i][j] += x[i][k] * y[k][j];
        }
    }
```

```
    for (i=0; i<3; i++) {
        for (j=0; j<3; j++){
            pdu_buf.VAPDU.VADATA.rotation[i][j] = 0;
            for (k=0; k<3; k++)
                pdu_buf.VAPDU.VADATA.rotation[i][j] += A[i][k] * z[k][j];
        }
    }
}
```

```
*****************************************************************************/
#define TANKA 0x68                      /* 02-cf-1f-30-27-68 */
#define TANKB 0xff95                    /* 02-cf-1f-30-27-95 */
#define MCC   0x09                      /* 02-cf-1f-30-28-09 */
#define ANZR  0x14                      /* 08-00-09-00-ba-14 */

typedef struct {
        unsigned version  :4;           /* version of protocol */
        unsigned length   :12;          /* length of PDU in octets */
        unsigned protocol :8;           /* protocol PDU belongs to */
        unsigned kind     :8;           /* type of PDU within protocol */
} PDUHeader;

/* version field */
#define protocolVersionFeb87 0          /* the Feb. 1987 version of the protocols */
#define protocolVersionNov87 1          /* the Nov. 1987 version of the protocols */

/* protocol field */
#define protocolNone 0                  /* no protocol -- PDU used for padding */
#define protocolMgmt 1                  /* the Network Management Protocol */
#define protocolSim  2                  /* the Simulation Protocol */
#define protocolData 3                  /* the Data Collection Protocol */
#define protocolXfer 4                  /* the File Transfer Protocol */
#define protocolDiag 5                  /* the Diagnosis Protocol */

/* kind field */
#define activatePDUKind 1               /* Activate PDU */
#define activatingPDUKind 2             /* Activating PDU */
#define deactivatePDUKind 3             /* Deactivate PDU */
#define vehicleAppearancePDUKind 4      /* Vehicle Appearance PDU */
/* #define UNUSED 5                     /* Unused PDU */
#define vehicleImpactPDUKind 6          /* Vehicle Impact PDU */
#define groundImpactPDUKind 7           /* Ground Impact PDU */
#define indirectFirePDUKind 8           /* Indirect Fire PDU */
#define serviceRequestPDUKind 9         /* Service Request PDU */
#define resupplyOfferPDUKind 10         /* Resupply Offer PDU */
#define resupplyReceivedPDUKind 11      /* Resupply Received PDU */
#define repairPDUKind 12                /* Repair PDU */
#define repairedPDUKind 13              /* Repaired PDU */
#define collisionPDUKind 14             /* Collision PDU */
#define firePDUKind 15                  /* Fire PDU */
#define radiatePDUKind 16               /* Radiate PDU */
#define resupplyCancelPDUKind 17        /* ResupplyCancel PDU */

/* Vehicle Type Identifier Field */
#define vehMainBattleTank       1       /* M1 or T72 main battle tank */
#define vehPersonnelCarrier     2       /* M2, M3 or BMP */
#define vehCommandPost          3       /* M577 Command Post */
#define vehAmmunitionTruck      4       /* M977 Ammo Truck */
#define vehFuelTruck            5       /* M978 Fuel Truck */
#define vehSupplyTruck          6       /* M35-A2 Truck */
#define vehMortatCarrier        7       /* M106 Carrier */
#define vehSPHowitzer           8       /* M109 Howitzer */
#define vehRecoveryVehicle      9       /* M88 Recovery */
#define vehFISTVehicle          10      /* Fire Support */
```

```
* Appearance Field Descriptions *

typedef struct {
        PDUHeader pduHdr;           /* version, length, protocol, PDUkind */
        unsigned char exerciseID;   /* exercise identifier */
        unsigned char padding;
        unsigned short vehicleID;   /* vehicle identifier */
} SimPDUHeader;

typedef struct {
        unsigned char role;         /* role of vehicle: ammo truck,
                                                        fuel truck, etc */
        unsigned char batallion;    /* batallion (task force) vehicle belongs
                                       to */
        unsigned char company;      /* company (team) vehicle belongs to */
        unsigned char bumper;       /* bumper number within company */
} VehicleRole;

/* role field */
#define roleSimulator      0        /* a vehicle operated by a full crew,
                                       simulated by a crewed vehicle
                                       simulator */
#define roleOPFOR          1        /* a vehicle simulated by a Semi-automated
                                       Forces system */
#define roleGunneryTarget  2        /* a gunnery target, such as that simulated
                                       by an MCC system */
#define roleAmmoTruck      3        /* an ammunition truck, such as that
                                       simulated by an MCC system */
#define roleFuelTruck      4        /* a fuel truck, such as that simulated by
                                       an MCC system */
#define roleMaintTeam      5        /* a maintenance team , such as that
                                       simulated by an MCC system */
#define roleS2             6        /* a batallion S2's vehicle, such as that
                                       simulated by an MCC system as part of a
                                       tactical operations center (TOC) */
#define roleS3             7        /* a batallion S3's vehicle, such as that
                                       simulated by an MCC system as part of a
                                       TOC */
#define roleFSE            8        /* a batallion fire support officer's
                                       vehicle, such as those simulated by an
                                       MCC system as part of a TOC */
#define roleTACP           9        /* a batallion tactical air control party
                                       vehicle, such as those simulated by an
                                       MCC system as part of a TOC */
#define roleAdminLogCenter 10       /* a batallion admin/log center vehicle,
                                       such as that simulated by an MCC
                                       system */
#define roleOther          99       /* any other vehicle not in one of the above
                                       categories */

/* company field */
#define assignedBattalion  1        /* the vehicle is assigned to no unit in
                                       particular within the batallion */
#define assignedScoutPlt   2        /* the vehicle belongs to the batallion's
                                       scout platoon */
#define assignedTACP       3        /* the vehicle belongs to the batallion's
                                       tactical air control party */
```

```
typedef struct {
        EntityID  id;                   * include host : described number *

        /* Common to all vehicles */
        VehicleRole  role;              /* include ID of described number */
        unsigned char alignment;        /* offense, defense, friend, or foe */
        unsigned char vehicleClass; /* class of vehicle */
        /* unsigned short appearance;  /* type of vehicle and appearance */
        /* struct {
                unsigned vehKindMask : 6;
                unsigned un1             : 1;
                unsigned vehDestroyed    : 1;
                unsigned vehSmokePlume   : 1;
                unsigned vehFlaming      : 1;
                unsigned vehDustCloudMask : 2;
                unsigned un2             : 1;
                unsigned vehTOWLauncherUp : 1;
                unsigned vehEngineSmoke : 1;
                unsigned un3             : 1;
        } appearance; */
        struct {
                unsigned vehSmokePlume   : 1;
                unsigned vehFlaming      : 1;
                unsigned vehDustCloudMask : 2;
                unsigned un2             : 1;
                unsigned vehTOWLauncherUp : 1;
                unsigned vehEngineSmoke : 1;
                unsigned un3             : 1;
                unsigned vehKindMask : 6;
                unsigned un1             : 1;
                unsigned vehDestroyed    : 1;
        } appearance;
        float rotation [3][3];          /* vehicle rotation */
        float location [3];             /* exact vehicle location */
        short grid [2];                 /* approximate vehicle location */
        unsigned short engineSpeed; /* engine speed, in RPM */
        /* unsigned short padding; */
        unsigned short sequence;        /* sequence # for vehicleAppearancePDU */

        /* Depending on vehicle class */
        union {

              /* If a simple moving vehicle, without turret ... */
              struct {
                     float velocity [3];    /* velocity (m/sec/15) */
              } simple;

              /* If a tank */
              struct {
                     float velocity [3];    /* velocity (m/sec/15) */
                     unsigned short turretAzimuth;
                                            /* turret/hull orinntation */
                     unsigned short gunElevation;    /* gun/turret elevation */
              } tank;
        } u;
} VehicleAppearancePDU;

/* alignment field */
#define alignedFoe          0       /* the vehicle appears unfriendly to all
                                       participants */
```

```
#define alignmentAttack [illegible]       /* the vehicle is on the attack team */
#define alignmentDefense [illegible]      /* the vehicle is on the defense team */
#define alignmentNeutral [illegible]      /* the vehicle appears friendly to all
                                             participants */

/* vehicle class field */
#define vehicleClassStatic 1         /* the vehicle is always stationary when
                                        visible, and it has no independently
                                        movable parts */
#define vehicleClassSimple 2         /* the vehicle can move, but is has no
                                        independently movable parts */
#define vehicleClassTank   3         /* the vehicle can move, and it has a turret
                                        and a gun barrel */

typedef struct {
        unsigned char ammunition;    /* type of ammunition fired */
        unsigned char fuze;          /* type of fuze used */
        unsigned char quantity;      /* number of rounds in burst */
        unsigned char rate;          /* rate of fire, rounds per second */
} BurstDescriptor;

/* ammunition field */
#define ammoHEi25 1                  /* 25 mm high exposive incendiary shell */
#define ammoHEAT105 2                /* 105 mm high exposive anti-tank shell */
#define ammoAPDS25 3                 /* 25 mm armor piercing discarding sabot
                                        shell */
#define ammoAPDS105 4                /* 105 mm armor piercing discarding sabot
                                        shell */
#define ammoTP25 5                   /* 25 mm target practice shell */
#define ammoBomb500 6                /* 500 lb. bomb */
#define ammoHE107 7                  /* 107 mm (4.2in.) high exposive mortar
                                        shell */
#define ammoHE155 8                  /* 155 mm high exposive howitzer shell */
#define ammoMissileTOW 9             /* TOW anti-tank missile */
/* fuze field */
#define fuzePointDetonating 1        /* point detonating fuze */
#define fuzeProximity 2              /* proximity fuze */

typedef struct {
        unsigned char targetType:2;  /* what is known about the target */
        unsigned : 14;
        unsigned short vehicleID;    /* ID of target vehicle, if known */
} TargetDescriptor;

/* targetType field */
#define targetUnknown 0              /* the target vehicle is not known */
#define targetNotVehicle 1           /* the target is known, but it is not a
                                        vehicle */
#define targetVehicle 2              /* the target is known and it is not a
                                        vehicle */

/* */
#define MYTANKID 16
#define MAXBUF          8192
#define HEADER_SIZE 14  /* ethernet header size including our header */

struct  ether { /* first three fields required for any link level packet */
        char  e_dhost[6];       /* 00-05   ethernet destination */
        char  e_shost[6];       /* 06-11   ethernet source */
        short e_type;           /* 12-13   ethernet packet type */
```

```
        unsigned data;
};

union {
        struct {
                unsigned length  :12;
                unsigned version :4;
        } i_datalength;
        short p_datalength;
} datalength;

typedef union {
        struct {
                char DATAONLY [1512 - HEADER_SIZE];
        } DATAONLYPDU;
        struct {
                PDUHeader ANYHDR;
                char data [1512 - HEADER_SIZE - 4];
        } ANYPDU;
        struct {
                VehicleAppearancePDU VADATA;
        } VAPDU;
} PDU;

#define MAXPKTSIZE 1514          /* total size of largest possible packet */
/* char send_addr[6];            /* our ethernet MAC address */
/* char recv_addr[6];            /* his ethernet MAC address */
char    my_addr[6];              /* my ethernet MAC address */
struct  exosopt opt;             /* EXOS board options include own address */
#define ETYPE htons(0x5208)      /* arbitrary unused ethernet type */
#define HELICOPTER11 11
#define HELICOPTER12 12
#define A10 13
PDU     pdu_buf;
struct  ether ether_buf;
```