

Network II.5 tutorial

Network II.5 – Without programming

WILLIAM J. GARRISON

CACI
345 North York Road, Hatboro, PA 19040

ABSTRACT

This tutorial will acquaint the reader with a powerful modelling tool which can dramatically reduce amount of time it takes to simulate a computer system. The NETWORK II.5 world view and the class of problems it addresses will be discussed. A summary of NETWORK II.5 capabilities will be presented. To tie the tutorial concepts together, an example of using NETWORK II.5 is included.

1. BACKGROUND

Simulation languages ease the burden of programming simulation problems. However, in certain application areas, higher level facilities can be designed to speed the simulation analysis. While this higher level facility would probably be written in a simulation language, it would require no programming on the part of the user. Instead, the user is interactively guided in building a description of the system to be simulated. The higher level facility would then actually write and run the model. This higher level facility would allow someone to dive into a simulation problem without the delay inherent in becoming fluent in a simulation language and coding up a model. NETWORK II.5 is such a higher level facility for modeling computers and communication networks.

2. NETWORK II.5 WORLD-VIEW

There are two concepts at the heart of the design of NETWORK II.5. First, items to be simulated are viewed based strictly on their function. Second, timing is the most important consideration in the simulation.

There are four main functions in the computer and communications world. They are Processing, Transfer, Storage and Control (called Module in NETWORK II.5). NETWORK II.5 has a separate general purpose building block to model each one of these functions. The Processing, Transfer and Storage building blocks are used to model hardware and the Module building block models both the software and control functions that may be in either hardware, firmware or software. Every device is described in terms of the function it provides to the system to be simulated. Many real world devices require more than one NETWORK II.5 building block to be fully described because they embody more than one function. For example, a Personal Computer might require two processing elements (main processor and co-processor), three transfer devices (internal bus, RS-232 bus, parallel bus) and two storage devices (disk and main memory).

By concentrating on providing a few powerful function oriented building blocks, the explosive progress of technology will never outpace NETWORK II.5. Whether data is being moved via a bus, a satellite link,

fiber optics or some advanced technology yet to be discovered, the function is the same. Information is transferred. In addition, the limited number of building blocks makes NETWORK II.5 a very easy tool to learn to use.

NETWORK II.5 simulates a system based on timing. Instructions that run on a processing element are not executed in the sense that at the end of an add instruction, the result is 4 and the overflow bit is reset. Instead, the effect of the instruction on system operation is modelled. If the instruction sent a message, a transfer device will be tied up for the amount of time it took to send the message. If a fetch from memory is required, at the proper time and for the proper duration, a storage device will be utilized. NETWORK II.5 timing orientation facilitates measurement of such system considerations as response time, conflicts, device utilization, etc.

3.0 THE NETWORK II.5 APPLICATION DOMAIN

NETWORK II.5 was designed to simulate systems in which devices are requesting, manipulating and distributing information and making decisions based on the system state. Telephone networks, distributed database systems, automotive electronic systems, military communication systems, automated factories and local area networks are just a few of the current applications of NETWORK II.5.

NETWORK II.5 is extremely flexible, allowing the portions of a computer system of special interest to be modelled at a finely detailed level while the rest of the system is modelled at a coarser level. There is no arbitrary limit to the number of Processing Element, Transfer, Storage and Module building blocks used to describe a system. In addition, any device interconnection scheme is allowed.

4.0 WITHOUT PROGRAMMING?

You do not need to learn (or own) any computer language to run NETWORK II.5. NETWORK II.5 speaks your language! You describe the system being simulate using terms like Processing Element, Instruction and Bus Protocol = Collision. All errors are express in terms easily understood. "PE5 does not connect to TD5" is much easier to understand than "Subscript out of range."

NETWORK II.5 has an interactive front end called NETIN that guides a user when building, modifying or displaying a system description. NETIN will interactively diagnose logic errors in a user's model description, allowing quick corrections. It also can produce diagrams of both the hardware and the software the user has described. When a system description is complete, NETWORK will run the simulation using an interactive dialog to set up the

simulation parameters. If desired, the user can interactively follow the simulation's progress by requesting a narrative trace of the simulation. At the end of a NETWORK II.5 simulation, the user may request a post processed report called NETPLOT. By analyzing a log file produced during the simulation run, NETPLOT produces both timeline diagram showing simulation activity plotted along a time axis and graphs of utilization as a function of time.

NETANIMATION has recently been added to the NETWORK II.5 modeling tool. NETANIMATION allows a user to draw the computer simulated and watch the simulation graphically represented as the simulation progresses. A wide variety of styles and colors are available to represent the simulated system. Zooming and panning allow very large system to be displayed. Simulations may run automatically or single stepped and English trace messages may be displayed to explain the details of simulation activity. Currently, NETANIMATION runs on IBM compatible PCs, SUN workstations and the VAX family of computers. It can, however, animate NETWORK simulations executed on any other hardware by running in a post processing mode. A sample NETANIMATION display is given as Figure 2.

5.0 NETWORK II.5 AVAILABILITY

NETWORK II.5 is currently offered on IBM (CMS and TSO), VAX, IBM-compatible PC, SUN, CRAY, Data General and GOULD machines. Implementations for other machines will be added as required. It is identical in all implementations, so that models can be moved from one manufacturer's machine to another without any changes.

6.0 NETWORK II.5 REPORTS

There are ten basic categories of reports provided by NETWORK II.5. They are Module Summary, Processing Element Statistics, Data Transfer Statistics, Data Storage Statistics, a Narrative Trace, a Snapshot Report, a Timeline Report, a Hardware Diagram, a Software Diagram and Semaphore reports.

The Module Summary, Received Message Semaphore Statistics, Processing Element Statistics, Data Transfer Device Statistics and Data Storage Device Statistics reports are tabular in format and are produced at user specified times and at the end of the simulation to summarize the simulation results. Samples of these reports are given as figures 5 to 9. The Narrative Trace report is produced interactively upon demand and chronicles the progress of the simulation event by event as they occur. This report is interactive with the user to allow the user to stop a simulation if things are going wrong. The Snapshot Report lists the current status of every hardware device, module and semaphore in the simulation and is produced both as a part of the tabular reports and interactively during a simulation run in response to a user request. The Timeline and Utilization reports are post processed reports that act upon a database produced during a simulation run to show the status of every hardware device and every semaphore in the simulation. The time span plotted on these reports is user specified so that a user can go back and expand the time scale of a period of interest several times until the needed information is obtained.

7.0 THE TUTORIAL EXAMPLE

For the purpose of illustration, an extremely simplified example of using NETWORK II.5 is presented here. See the references for examples of a more realistic problem both presented and solved.

An office contains 2 computers on the same serial bus. Files are sent from one computer to another. Only one computer can be using the bus at a time. Computer A requests a file called "Data" from Computer B every 30 seconds. Computer B requests a file called "More Data" from Computer A every 45 seconds. "Data" is 700 bits long. "More Data" is 770 bits long. The serial bus moves data at 1200 baud. Assume the files are stored in a 7 bit code and the hardware adds a parity bit to each character transmitted. Run the simulation of this system for 1000 seconds.

7.1 PROBLEM FORMULATION

HARDWARE - Each Processing Element will need 2 instructions: A "Request file from other PE" instruction on and a "Send file to other PE" instruction. Memories need not be simulated (but they certainly could be). Processor internal speed is not significant to this simulation!

SOFTWARE - Each processing element will need a module that sends a file upon the receipt of a request and a module that requests a file every X seconds.

The user prepares the input file to NETWORK through the use of the interactive program NETIN. NETIN guides the user through building a description of the system to be simulated and provides on-line documentation. A brief example of a NETIN session is presented as Figure 1.

In NETIN a user can always get help by typing "?". Also, when the user becomes an "expert", a BRIEF level of prompting may be chosen which significantly shortens the dialog. The output of NETIN is a file which becomes the input to NETWORK. The file is easily readable and useful for documenting the simulation performed. The sample portions of the file produced by NETIN to solve this example problem are given as Figure 3.

Running the simulation involves an interactive dialog with NETWORK, an example of which is included as Figure 4. If desired, the user can also watch the simulation progress on a graphics display using NETANIMATION. As a result of running this data file, the user gets the end of simulation reports. The user also could request a timeline plot, a utilization plot, a narrative trace report, a hardware diagram or a software diagram.

REFERENCES

1. Garrison, W.J., NETWORK II.5 USER'S MANUAL, Version 3.1, CACI, Inc., June, 1988.
2. Russell, E.C., BUILDING SIMULATION MODELS WITH SIMSCRIPT II.5, CACI, Inc., January 1983.
3. CACI, Inc., A QUICK LOOK AT NETWORK II.5, Sept. 1988.
4. Karplus and Cheung, PERFORMANCE EVALUATION TOOLS FOR SIMULATORS CONSISTING OF NETWORKS OF MICRO-COMPUTERS, Computer Science Department, University of California, Los Angeles, 1984.
5. Cheung, Dimitriadis and Karplus, INTRODUCTION TO SIMULATION USING NETWORK II.5, CACI, Inc., 1987.

07.12.42 > netin
 WELCOME TO THE COMPUTER NETWORK SIMULATION INPUT PROGRAM NETIN

ENTER A NETIN TOP-LEVEL COMMAND
 >?

NETIN TOP-LEVEL COMMANDS AND THEIR USAGE

"?"	("?")	LISTS THE AVAILABLE TOP-LEVEL COMMANDS
"HELP"	("H")	PROVIDES BRIEF EXPLANATION OF COMMANDS
"PROMPT"	("P")	ALLOWS YOU TO SET THE LEVEL OF PROMPTING
"LOAD"	("LO")	ALLOWS YOU TO LOAD DATA FROM A FILE
"VERIFY"	("V")	ALLOWS YOU TO CHECK CURRENT DATA
"SAVE"	("S")	ALLOWS YOU TO WRITE TO A FILE
"DISPLAY"	("DI")	ALLOWS YOU TO DISPLAY A LIST OF BASIC ENTITIES WHICH ARE IN CORE
"FIND"	("F")	LOCATES A SPECIFIED BASIC ENTITY AND ENTERS THE MANIPULATION MODE
"CREATE"	("CR")	ALLOWS YOU TO ENTER DATA WHICH DESCRIBES A NEW BASIC ENTITY
"DELETE"	("DE")	ALLOWS YOU TO DELETE AN ENTITY
"QUIT"	("Q")	TERMINATES PROGRAM EXECUTION

ENTER A NETIN TOP-LEVEL COMMAND

>create ?

ENTER ONE OF THE BASIC ENTITY TYPES LISTED BELOW:

- PROCESSING.ELEMENT OR PE
- BUS OR B
- STORAGE.DEVICE OR SD
- MODULE OR M
- INSTRUCTION.MIX OR IM
- FILE OR F

>pe

YOU WILL BE ASKED FOR THE DATA TO MAKE A NEW
 PROCESSING.ELEMENT
 NAME (TEXT)

>computer a

BASIC.CYCLE.TIME (REAL; MICROSEC)

>30

INPUT.CONTROLLER (TEXT; YES/NO)

>pd

THE DEFAULT VALUE IS "NO".

ENTER A "YES" OR A "NO" OR ENTER "D" FOR "DEFAULT".

>d

THE DEFAULT, "NO" , HAS BEEN ASSUMED

Figure 1

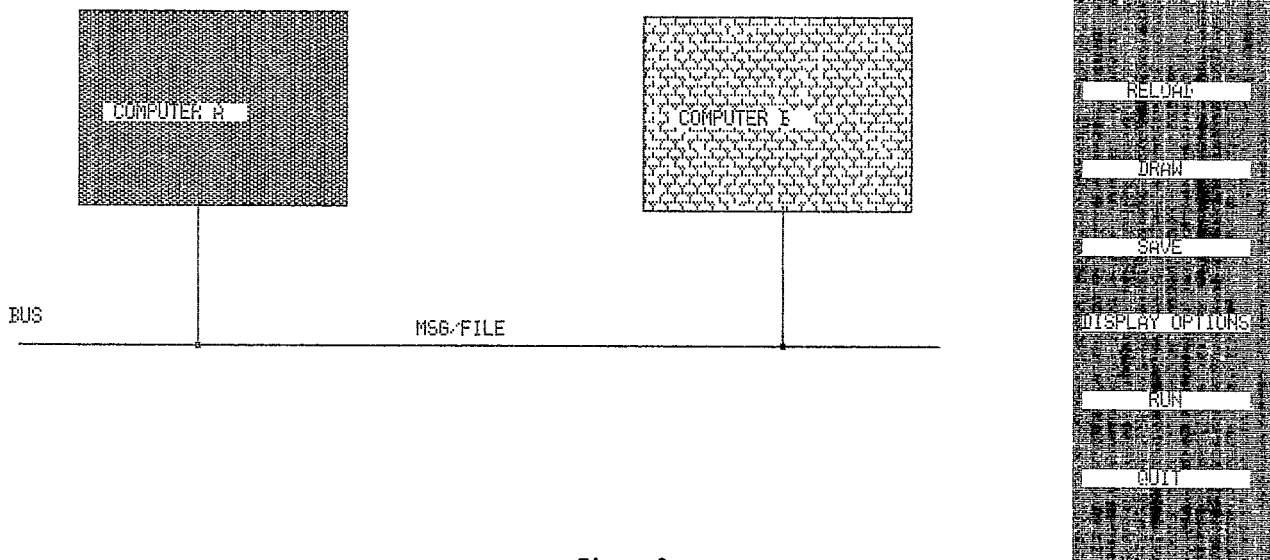


Figure 2

```

* CASE 1 - A 2 COMPUTER, 1 BUS ARCHITECTURE
*****PROCESSING ELEMENTS - SYS.PE.SET
HARDWARE TYPE = PROCESSING
NAME = COMPUTER A
  BASIC CYCLE TIME = 1.000 MICROSEC
  INPUT CONTROLLER = NO
  INSTRUCTION REPERTOIRE =
    NAME ; ASK FOR "DATA" FILE
    MESSAGE ; ASK FOR "DATA" FILE
    LENGTH ; 0 BITS
    DESTINATION PROCESSOR ; COMPUTER B
  NAME ; SEND "MORE DATA" FILE
  MESSAGE ; SEND "MORE DATA" FILE
  LENGTH ; 770 BITS
  DESTINATION PROCESSOR ; COMPUTER B
*****BUSSES - SYS.BUS.SET
HARDWARE TYPE = DATA TRANSFER
NAME = BUS
  CYCLE TIME = 833.00 MICROSEC
  BITS PER CYCLE = 1
  CYCLES PER WORD = 7
  WORDS PER BLOCK = 1
  WORD OVERHEAD TIME = 833.00 MICROSEC
  BLOCK OVERHEAD TIME = 0.0 MICROSEC
*****MODULES - SYS.MODULE.SET
SOFTWARE TYPE = MODULE
NAME = REQUEST "DATA" FILE
PRIORITY = 0
INTERRUPTABILITY FLAG = NO
CONCURRENT EXECUTION = NO
ITERATION PERIOD = 3.00E+07 MICROSEC
ALLOWED PROCESSORS =
  COMPUTER A
INSTRUCTION LIST =
  EXECUTE A TOTAL OF ; 1 ASK FOR "DATA" FILE

```

Figure 3

```

>network
ENTER NAME OF INPUT FILE
>CASE1
SOURCE MODEL IS FROM CASE1 NETWORK
OUTPUT WILL GO TO CASE1 LISTING
PLOT INPUT DATA WILL GO TO CASE1 PLOTIN.

WELCOME TO CACI NETWORK II.5.
YOU ARE USING VERSION 3.0.5.

DO YOU WANT A LISTING OF YOUR INPUT FILE (OFFLINE)? (Y/N)
>y

ENTER THE TIME UNIT WHICH YOU WISH TO EMPLOY FOR INPUT.
IT MUST BE SECOND(S), MILLISECONDS(MIL), OR MICROSECONDS(MIC).
>s
ENTER LENGTH OF SIMULATION (IN SECONDS)
>1000
DO YOU WISH TO HAVE PERIODIC REPORTING? (Y/N)
>n

DO YOU WISH TO TRACE THE EVENT FLOW? (Y/N)
>n
SIMULATION TERMINATED AT 10.000000 SECONDS.

```

Figure 4

CASE 1 - A 2 COMPUTER, 1 BUS ARCHITECTURE

PROCESSING ELEMENT UTILIZATION STATISTICS

FROM 0. TO 1000. SECONDS

(ALL TIMES REPORTED IN MICROSECONDS)

PROCESSING ELEMENT NAME	COMPUTER A	COMPUTER B
STORAGE REQUESTS GRANTED	0.	0.
INTERRUPTED REQUESTS	0.	0.
AVERAGE WAIT TIME	0.	0.
MAXIMUM WAIT TIME	0.	0.
STD DEV WAIT TIME	0.	0.
GEN STORAGE REQUESTS	0.	0.
FILE REQUESTS GRANTED	0.	0.
INTERRUPTED REQUESTS	0.	0.
AVERAGE WAIT TIME	0.	0.
MAXIMUM WAIT TIME	0.	0.
STD DEV WAIT TIME	0.	0.
TRANSFER REQUESTS GRANTED	55	55
INTERRUPTED REQUESTS	0.	0.
AVERAGE WAIT TIME	0.	0.
MAXIMUM WAIT TIME	0.	0.
STD DEV WAIT TIME	0.	0.
INPUT CONTROLLER REQUEST	0.	0.
DEST PE REQUESTS	55	55
AVERAGE WAIT TIME	0.	0.
MAXIMUM WAIT TIME	0.	0.
STD DEV WAIT TIME	0.	0.
RESTARTED INTERRUPTS	0.	0.
AVG TIME PER INTERRUPT	0.	0.
MAX TIME PER INTERRUPT	0.	0.
STD DEV INTERRUPT TIME	0.	0.
MAX INTERRUPT QUEUE SIZE	0.	0.
AVG INTERRUPT QUEUE SIZE	0.	0.
STD DEV INTERRUPT QUEUE	0.	0.
MAX MODULE QUEUE SIZE	1	1
AVG MODULE QUEUE SIZE	0.	0.
STD DEV MODULE QUEUE	0.	0.
PER CENT PE UTILIZATION	3.812	3.812

Figure 5

CASE 1 - A 2 COMPUTER, 1 BUS ARCHITECTURE

COMPLETED INSTRUCTION REPORT

FROM 0. TO 1000. SECONDS

INSTRUCTION NAME	COUNT	INSTRUCTION NAME	COUNT
COMPUTER A			
ASK FOR "DATA" FILE	33	SEND "MORE DATA" FILE	22
COMPUTER B			
ASK FOR "MORE DATA" FILE	22	SEND "DATA" FILE	33

Figure 6

CASE 1 - A 2 COMPUTER, 1 BUS ARCHITECTURE

TRANSFER DEVICE UTILIZATION STATISTICS

FROM 0. TO 1000. SECONDS

(ALL TIMES REPORTED IN MICROSECONDS)

TRANSFER DEVICE NAME	BUS
TRANSFER REQUESTS GRANTED	110
INTERRUPTED REQUESTS	0
AVG REQUEST DELAY	0.
MAX REQUEST DELAY	0.
STD DEV REQUEST DELAY	0.
INTERRUPTED TRANSFERS	0
COMPLETED TRANSFERS	110
AVG USAGE TIME	346528.000
MAX USAGE TIME	733040.000
STD DEV USAGE TIME	347296.072
AVG QUEUE SIZE	0.050
MAX QUEUE SIZE	1.000
STD DEV QUEUE SIZE	0.005
PER CENT OF TIME BUSY	3.812

Figure 7

CASE 1 - A 2 COMPUTER, 1 BUS ARCHITECTURE

COMPLETED MODULE STATISTICS

FROM 0. TO 1000. SECONDS

(ALL TIMES REPORTED IN MICROSECONDS)

MODULE.NAME	REQUEST "DATA" FILE	REQUEST "MORE DATA" FILE	TRANSMIT "DATA" FILE
HOST PE	COMPUTER A	COMPUTER B	COMPUTER B
COMPLETED EXECUTIONS	33	22	33
CANCELLATIONS DUE TO ITERATION PERIOD	0	0	0
RUN UNTIL SEMAPHORES	0	0	0
MESSAGE REQUIREMENTS	0	0	0
SUCCESSOR ACTIVATION	0	0	0
AVG PRECONDITION TIME	0.	0.	0.
MAX PRECONDITION TIME	0.	0.	0.
MIN PRECONDITION TIME	0.	0.	0.
STD DEV PRECOND TIME	0.	0.	0.
AVG EXECUTION TIME	0.	0.	666400.000
MAX EXECUTION TIME	0.	0.	666400.000
MIN EXECUTION TIME	0.	0.	666400.000
STD DEV EXECUTION TIME	0.	0.	0.
RESTARTED INTERRUPTS	0.	0.	0.
AVG TIME PER INTERRUPT	0.	0.	0.
MAX TIME PER INTERRUPT	0.	0.	0.
STD DEV INTERRUPT TIME	0.	0.	0.

Figure 8

CASE 1 - A 2 COMPUTER, 1 BUS ARCHITECTURE

COMPLETED MODULE STATISTICS

FROM 0. TO 1000. SECONDS

(ALL TIMES REPORTED IN MICROSECONDS)

MODULE.NAME	TRANSMIT "MORE DATA" FILE
HOST PE	COMPUTER A
COMPLETED EXECUTIONS	22
CANCELLATIONS DUE TO ITERATION PERIOD	0
RUN UNTIL SEMAPHORES	0
MESSAGE REQUIREMENTS	0
SUCCESSOR ACTIVATION	0
AVG PRECONDITION TIME	0.
MAX PRECONDITION TIME	0.
MIN PRECONDITION TIME	0.
STD DEV PRECOND TIME	0.
AVG EXECUTION TIME	733040.000
MAX EXECUTION TIME	733040.000
MIN EXECUTION TIME	733040.000
STD DEV EXECUTION TIME	0.
RESTARTED INTERRUPTS	0.
AVG TIME PER INTERRUPT	0.
MAX TIME PER INTERRUPT	0.
STD DEV INTERRUPT TIME	0.

Figure 9

CASE 1 - A 2 COMPUTER, 1 BUS ARCHITECTURE
 S N A P S H O T R E P O R T
 AT SIMULATED TIME 1000. SECONDS

```

PE "COMPUTER A" IS IDLE
  MESSAGE "SEND "DATA" FILE"                      700. BITS IS QUEUED
PE "COMPUTER B" IS IDLE
  MESSAGE "SEND "MORE DATA" FILE"                770. BITS IS QUEUED

BUS "BUS" IS IDLE
    
```

Figure 10

CASE 1 - A 2 COMPUTER, 1 BUS ARCHITECTURE
 HARDWARE INTERCONNECTION DIAGRAM

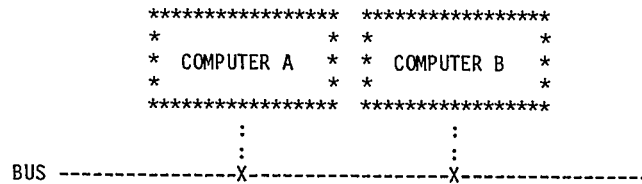


Figure 11

CASE 1 - A 2 COMPUTER, 1 BUS ARCHITECTURE
 DIAGRAMS OF ALL SOFTWARE MODULES

PARENT MODULE : REQUEST "DATA" FILE
 ITERATION PERIOD : 30 SEC

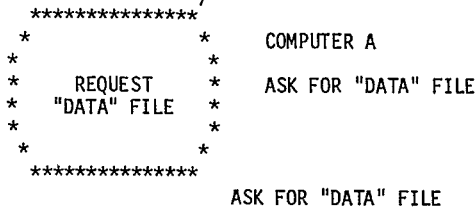


Figure 12

WILLIAM J. GARRISON is an Executive Associate in the Modelling and Simulation Department of CACI. He led the development of NETWORK II.5, a Computer and Communications Network Simulator, and has been active in applying simulation to manufacturing problems. Currently, he is involved in the development of future versions of NETWORK II.5 and teaches courses and consults on the use of both NETWORK II.5 and the simulation language SIMSCRIPT II.5.

William Garrison received the B.S. degree in Electric Engineering from the University of Pennsylvania in 1975. He received an M.S. degree in Computer Science from the University of Pennsylvania in 1980. He is a member of the engineering honor society Tau Beta Pi and a member of the IEEE Computer Society.

William J. Garrison
C.A.C.I. Inc.
345 North York Road
Hatboro, PA 19040
(215) 675-0895