1

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

**A CAPACITY PLANNING MODEL FOR
A LARGE CLIENT-SERVER SYSTEM**

by

Yizhong Shen

**A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of
Master of Science**

**Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, ontario**

April 1996

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395, Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file - votre référence

Our file - Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-612-13872-0

Canada

Name _____

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided

--	--	--	--

U·M·I

SUBJECT TERM

SUBJECT CODE

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture	0729
Art History	0377
Cinema	0900
Dance	0378
Fine Arts	0357
Information Science	0723
Journalism	0391
Library Science	0399
Mass Communications	0708
Music	0413
Speech Communication	0459
Theater	0465

EDUCATION

General	0515
Administration	0514
Adult and Continuing	0516
Agricultural	0517
Art	0273
Bilingual and Multicultural	0282
Business	0688
Community College	0275
Curriculum and Instruction	0727
Early Childhood	0518
Elementary	0524
Finance	0277
Guidance and Counseling	0519
Health	0680
Higher	0745
History of	0520
Home Economics	0278
Industrial	0521
Language and Literature	0279
Mathematics	0280
Music	0522
Philosophy of	0988
Physical	0523

Psychology	0525
Reading	0535
Religious	0527
Sciences	0714
Secondary	0533
Social Sciences	0534
Sociology of	0340
Special	0529
Teacher Training	0530
Technology	0710
Tests and Measurements	0288
Vocational	0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language	
General	0679
Ancient	0289
Linguistics	0290
Modern	0291
Literature	
General	0401
Classical	0294
Comparative	0295
Medieval	0297
Modern	0298
African	0316
American	0591
Asian	0305
Canadian (English)	0352
Canadian (French)	0355
English	0593
Germanic	0311
Latin American	0312
Middle Eastern	0315
Romance	0313
Slavic and East European	0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy	0422
Religion	
General	0318
Biblical Studies	0321
Clergy	0319
History of	0320
Philosophy of	0322
Theology	0469

SOCIAL SCIENCES

American Studies	0323
Anthropology	
Archaeology	0324
Cultural	0326
Physical	0327
Business Administration	
General	0310
Accounting	0272
Banking	0770
Management	0454
Marketing	0338
Canadian Studies	0385
Economics	
General	0501
Agricultural	0503
Commerce Business	0505
Finance	0508
History	0509
Labor	0510
Theory	0511
Folklore	0358
Geography	0366
Gerontology	0351
History	
General	0578

Ancient	0579
Medieval	0581
Modern	0582
Black	0328
African	0331
Asia Australia and Oceania	0332
Canadian	0334
European	0335
Latin American	0336
Middle Eastern	0333
United States	0337
History of Science	0585
Law	0398
Political Science	
General	0615
International Law and	
Relations	0616
Public Administration	0617
Recreation	0814
Social Work	0452
Sociology	
General	0626
Criminology and Penology	0627
Demography	0938
Ethnic and Racial Studies	0631
Individual and Family	
Studies	0628
Industrial and Labor	
Relations	0629
Public and Social Welfare	0630
Social Structure and	
Development	0700
Theory and Methods	0344
Transportation	0709
Urban and Regional Planning	0999
Women's Studies	0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture	
General	0473
Agronomy	0285
Animal Culture and	
Nutrition	0475
Animal Pathology	0476
Food Science and	
Technology	0359
Forestry and Wildlife	0478
Plant Culture	0479
Plant Pathology	0480
Plant Physiology	0817
Range Management	0777
Wood Technology	0746
Biology	
General	0306
Anatomy	0287
Biostatistics	0308
Botany	0309
Cell	0379
Ecology	0329
Entomology	0353
Genetics	0369
Limnology	0793
Microbiology	0410
Molecular	0307
Neuroscience	0317
Oceanography	0416
Physiology	0433
Radiation	0821
Veterinary Science	0778
Zoology	0472
Biophysics	
General	0786
Medical	0760

EARTH SCIENCES

Biogeochemistry	0425
Geochemistry	0996

Geodesy	0370
Geology	0372
Geophysics	0373
Hydrology	0388
Mineralogy	0411
Paleobotany	0345
Paleoecology	0426
Paleontology	0418
Paleozoology	0985
Palynology	0427
Physical Geography	0368
Physical Oceanography	0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences	0768
Health Sciences	
General	0566
Audiology	0290
Chemotherapy	0992
Dentistry	0567
Education	0350
Hospital Management	0769
Human Development	0758
Immunology	0982
Medicine and Surgery	0564
Mental Health	0347
Nursing	0569
Nutrition	0570
Obstetrics and Gynecology	0380
Occupational Health and	
Therapy	0354
Ophthalmology	0381
Pathology	0571
Pharmacology	0419
Pharmacy	0572
Physical Therapy	0382
Public Health	0573
Radiology	0574
Recreation	0575

Speech Pathology	0460
Toxicology	0383
Home Economics	0386

PHYSICAL SCIENCES

Pure Sciences	
Chemistry	
General	0485
Agricultural	0749
Analytical	0486
Biochemistry	0487
Inorganic	0488
Nuclear	0738
Organic	0490
Pharmaceutical	0491
Physical	0494
Polymer	0495
Radiation	0754
Mathematics	0405
Physics	
General	0605
Acoustics	0986
Astronomy and	
Astrophysics	0606
Atmospheric Science	0608
Atomic	0748
Electronics and Electricity	0607
Elementary Particles and	
High Energy	0798
Fluid and Plasma	0759
Molecular	0609
Nuclear	0610
Optics	0752
Radiation	0756
Solid State	0611
Statistics	0463
Applied Sciences	
Applied Mechanics	0346
Computer Science	0984

Engineering	
General	0537
Aerospace	0538
Agricultural	0539
Automotive	0540
Biomedical	0541
Chemical	0542
Civil	0543
Electronics and Electrical	0544
Heat and Thermodynamics	0348
Hydraulic	0545
Industrial	0546
Marine	0547
Materials Science	0794
Mechanical	0548
Metallurgy	0743
Mining	0551
Nuclear	0552
Packaging	0549
Petroleum	0765
Sanitary and Municipal	0554
System Science	0790
Geotechnology	0428
Operations Research	0796
Plastics Technology	0795
Textile Technology	0994

PSYCHOLOGY

General	0621
Behavioral	0364
Clinical	0622
Developmental	0620
Experimental	0623
Industrial	0624
Personality	0625
Physiological	0989
Psychobiology	0349
Psychometrics	0632
Social	0451

**The undersigned hereby recommend to the Faculty of Graduate
Studies and Research acceptance of the thesis,**

**A CAPACITY PLANNING MODEL FOR
A LARGE CLIENT-SERVER SYSTEM**

submitted by Yizhong Shen

in partial fulfillment of the requirements

for the degree of Master of Science

**Dr. C. M. Woodside,
Thesis Supervisor**

**Dr. S. A. Mahmoud,
Chairman,
Department of Systems and
Computer Engineering**

**Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University
April 1996**

ABSTRACT

A capacity planning model, partially symmetric model, for a large client-server system is proposed in this work. The model is designed for a real large database system belonging to a major Canadian company.

This model can be used:

- to obtain performance information (response time, throughput) of the system with the different workloads;
- to determine the bottleneck of the system;
- to get performance information for the suggested architecture.

The advantages of the partially symmetric model is:

- can solve large models which over the limitation of the most of analysis tools;
- efficient;
- easy to use.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. C. Murray Woodside for the advice, guidance and encouragement which he has given me during this project.

I also would like to thank Mr. Dave Rypma, Mr. Paul Sehra, Mr. Greg Franks and Mr. Alex Hubbard for their support and discussions.

Finally, special appreciation goes to my husband Deyan and to my daughter Donna, to whom this thesis is dedicated.

Table of Contents

CHAPTER 1	Introduction	1
	1.1 Large systems	1
	1.2 Capacity planning	2
	1.3 The Bell Sygma database system	3
	1.4 This work	6
CHAPTER 2	Theory	8
	2.1 Queueing network model	8
	2.2 Decomposition	12
	2.3 Algorithm	16
	2.3.1 Product form queueing network	18
	2.3.2 Mean value analysis (MVA)	19
	2.3.3 Heuristic algorithm Linearizer	22
CHAPTER 3	A Model Exploiting the Symmetry of a Large Client-Server System	24
	3.1 Performance measurements of the Bell Sygma database system	24
	3.1.1 Components of the data	24
	3.1.2 Transaction types	26
	3.2 Performance modelling of the Bell Sygma database system	30
	3.2.1 Modelling assumptions	30

- 3 2 2 Service centers of the model 34
- 3 2 3 3 - layer model 42
- 3 2 4 Model validation 43

3.3 A second, modified architecture of four-layer model 45

3.4 Capacity planning tool: CPT_SCSN 48

- 3 4 1 Instructions for use 48
- 3 4 2 System requirements 50

CHAPTER 4

Case Study 51

4.1 Assumptions and parameters 51

4.2 Performance analysis 52

- 4 2 1 Database servers 52
- 4 2 2 Local server 55
- 4 2 3 Intermediate server 59
- 4 2 4 Analysis 62

CHAPTER 5

Conclusions 63

5.1 Advantages of the Partially Symmetric Model 63

5.2 Capacity Planning Tool 64

5.3 Experimental Results 64

5.4 Weaknesses and Future Work 65

FIGURES

FIGURE 1-1	The database system components	5
FIGURE 2-1	An open queueing network model	10
FIGURE 2-2	An closed queueing network model	11
FIGURE 2-3	Model 1: Database system with mutex control	15
FIGURE 2-4	Model 2: Surrogate delay (EDSC) model	16
FIGURE 2-5	Iterative procedure to get service time of DESC	17
FIGURE 3-1	Main components of workstation data	25
FIGURE 3-2	Path of transaction in the database system	29
FIGURE 3-3	Structure of the database system	32
FIGURE 3-4	Partially symmetric model of the database system	33
FIGURE 3-5	Service Center of workstation	34
FIGURE 3-6	Service Center of token ring LAN	36
FIGURE 3-7	Service Center of LAN server	38
FIGURE 3-8	Procedure to get service time of LAN server (DESC)	39
FIGURE 3-9	Service Center of BackBone and WAN	40
FIGURE 3-10	Service Center of database server	41
FIGURE 3-11	Database system for validation	43
FIGURE 3-12	Service Center of intermediate server	46
FIGURE 3-13	The 4 -layer server model	47
FIGURE 4-1	Performance of RF database server	53
FIGURE 4-2	Performance of BC database server	54
FIGURE 4-3	Performance of LAN server (traffic factor)	56
FIGURE 4-4	Performance of LAN server (service time)	57
FIGURE 4-5	Performance of LAN server (workload)	58
FIGURE 4-6	Performance with and without intermediate server	60
FIGURE 4-7	Effect of fraction on performance	61

REFERENCES	67
-------------------	----

APPENDIX A - The source code of partially symmetric model solver	68
APPENDIX B - GUI of CPT_SCSN	79
APPENDIX C - The source code of GUI	89

1.1 Large systems

What is a system?

A system includes a collection of objects which work together to complete certain tasks. Computer systems, communication networks, database systems, and manufacturing lines are examples of systems. The behavior of these systems needs to be understood in order to be able to design new systems or to improve the performance of existing systems.

What is a large system?

A large system has many objects and performs complex operations. The workload of a large system is either in terms of many users, or a heavy workload for a few users. A database system which includes two main-frame computers, many disks, a wide area network, hundreds of local area networks, thousands of workstations and PCs, and performs hundreds of kinds of transaction with high workload, is an example of a large system.

In this work, we study the performance of a very large database system. We pay attention to capacity of the system, but we do not consider the detailed operation of the individual objects in the system.

1.2 Capacity planning

The capacity of a system is defined as the greatest workload at which the response time of the system remains acceptable. For a database system, the capacity of the system is the maximum of transactions at which the response time of the database system is acceptable.

To increase the capacity of the database system, we can increase the capacity of the individual objects such as LAN, WAN or the mainframe computer, or add new components to the system.

Here we will suggest adding an "intermediate" database server to the system, which implies:

- partitioning the database or duplicating the database;
- changing the software structure.

All these suggested changes are based on the analysis of the performance of the system.

Two techniques can be used to get the performance information of a system.

1. Measurement is used to determine the behavior of an existing system. Measurement includes using hardware monitors and software monitors. A hardware monitor has a limited capability for the performance measurement; a software monitor can produce much more detailed information about the system performance. But it is difficult to use the data obtained from the measurements to predict the future behavior of the system.
2. Performance modeling is used to determine the behavior of an existing system and also to predict the future performance of the system. Capacity planning usually uses performance modeling because it has to predict the future performance of the system. Modeling makes it

possible to study a system in a controlled way and to gain a deeper understanding of how a system works.

This work uses limited measurements to create a model which is primarily used to make predictions for cases with different parameters that represent possible future workloads.

1.3 The database system to be studied

The database system has two properties:

1. The database system is very large: some thousands of PCs and workstations doing transactions; all transactions go through the network (token ring, FDDI or WAN); transaction type is more than hundreds; the workload is very high.
2. The database system is not isolated. For performance analysis, other traffic in the network and other utilization of the CPU must be considered. The other traffic and utilization are not related to the database system, but they affect the performance of the database system.

Figure 1-1 shows the components of the database system.

Two mainframe computers are used to control two databases: the RF database and the BC database. The Front End Processor (FEP) is connected to two mainframe computers by high speed cable (150 Mbit/second). A part of the workstations are connected via a "backbone" campus network; others via a wide-area network. These will be called "local" (backbone) and "remote" (WAN) clients, respectively.

A client has three level of servers:

- a workstation - set up client program
- a LAN server - switch the transactions to the right database
- a mainframe - query or update database

Figure 1-1 does not show the multiplicity of the WAN connections to various local networks far from the mainframe or the multiplicity of LANs attached via LAN servers to the "backbone" subnet.

The details of equipments used in the mainframe computer servers, disk storage and front-end processors are not of primary importance in this model and will not be explained. The LAN servers and workstations are PCs, and the LANs are token rings.

It is already mentioned that the database system is not isolated. Other traffic exists in LAN, WAN and backbone, and other applications use the mainframe computers. We will call them as "external traffic" and "external applications", respectively.

It is mentioned that the database system to be studied is a real system belonging to a major Canadian company, however the parameters used in the model were only loosely based on measured values, because the available measurements are incomplete.

FIGURE 1-1 The database system components

1.4 This work

The goals of this work are:

1. To build a performance analysis model for the database system.
2. To determine the points of saturation in the distributed part of the database system by the model.
3. To suggest an architecture to improve the performance of the database system.
4. To create a GUI tool for the analysis model.

My contribution includes:

1. To create a decomposed symmetric queueing network model which can be used for performance analysis in large systems. The algorithm used in the model is Delay Equivalent Service Center (DESC). With the principles of the decomposed symmetric queueing network model, I designed and implemented a model for the database system. This model exploits symmetry in the real system which greatly simplifies the analysis; and it can be used for a very large system which is over the limitation of the analysis tool.
2. To determine parameters of the model. These parameters are based on analysis of measured data from the real database system.
3. In a case study loosely related to the present workload, the bottleneck of the database system was found to be at the mainframe servers rather than in the distributed part of the system.
4. To suggest an architecture that would be more scalable, based on moving some transactions from the mainframe computer servers to "intermediate servers."
5. To develop a planning tool based on the partially symmetric model which could be used to experiment with different system components and workload alternatives. The tool is called CPT_SCSN (Capacity Planning Tool for Symmetric Client-Server Networks) and is tailored to the structure of the database system.

The thesis is divided into five chapters.

Chapter 2 describes some of the modelling theory used in the study.

Chapter 3 describes a simplified abstract view of the system which is adopted for capacity planning. In this view the system is made up of M

identical local areas each with N user workstations, whereas in reality every local area is a little different. Many other details are suppressed in this abstract view, including:

- all details of the large and complex mainframe computer database servers that pose their own capacity planning problem which is relatively well understood;
- fine details of media access delays in the local area, backbone and wide-area nets which are far from saturated;
- details of competing “external” traffic which is represented only in its aggregate workload in the networks and the mainframe computers;
- some of the detail of the user transaction types.

Chapter 3 also describes the performance model. The model exploits the symmetry in the abstract view to give very fast cheap solutions, even though the system is so large and the workload so complex.

Chapter 3 introduces graphic user interface of the capacity planning tool, CPT_SCSN.

Chapter 4 describes a case study with parameters loosely based on the present situation. At the time of writing, more precise parameter values are not available however, the case study shows the effect of the given parameters and of certain shifts in parameters which might result from changes in user behaviour, user needs, or the redesign of the program.

Chapter 5 gives conclusions.

2.1 Queueing network models

Queueing network modeling [1][2] is widely used to model computer systems, network systems and database systems.

In a queueing network model, a system is presented as a network of queues which evaluated analytically [1] or simulated [3] by the model. A queueing network model is composed by a set of service centers or stations which represents system objects (cpu, disk, LAN, WAN,...), and a set of jobs which belong to one or more classes according to their operation.

Analytic evaluation uses certain algorithms, such as mean value analysis [1] or Linearizer (a heuristic algorithm) [5], to solve efficiently a set of equations induced by the queueing network model (station parameters, job parameters and operation)

Simulation uses the queueing network model to mimic the system behavior.

Several books [1][2][4] describe the both techniques.

Basically there are two types of queueing network model: open and closed. An open queueing network is characterized by one or more sources of job arrivals and one or more sinks that absorb jobs leaving the network, and the number of jobs in the system varies with time. In a closed queueing network, jobs neither enter or leave the network, jobs are present initially and do not need to be generated. The jobs in the model are keeping circulating from one queue to the next, the total number of jobs is constant. These two kind of behavior also can coexist together in the same model.

Defining a queueing network model of a system is made relatively straightforward by the close correspondence between the service centers or stations of queueing network model and the objects of the system. For example, service center in a queueing network model correspond to hardware resources and their queues (such as cpu, disk or LAN), and jobs in the model correspond to transactions or programs.

Here is an example taken from to show how to use queueing network model to represent a simple computer system[1].

A computer system includes 1 cpu, 3 disks, and several terminals. The jobs of the system are generated by the terminals, go to cpu, go to one of the three disk, come back to cpu and return to terminal.

Figure 2-1 gives an open queueing network model. In this model, there are no stations to represent the terminals, the parameter rate of job arriving is used to represent the performance of terminals.

Figure 2-2 gives a closed queueing network model. This model includes all objects of the system, including the terminals.

FIGURE 2-1 **An open queueing network model**

FIGURE 2-2 **An closed queueing network model**

Whether to use an open or a closed model is a decision which should depend on the situation. In this example, the closed model is better because the real system is closed. That is, there is a set of users which is nearly constant over time and which use the system nearly continuously. Our complex system is also more naturally modelled as a closed system for the same reason.

For analyzing a large system, an important strategies for the solution is decomposition. Section 2.2 will give the basic concepts of decomposition which include Flow Equivalent Service Center (FESC) and Delay Equivalent Service Center (DESC). Section 2.3 will discuss algorithms used for analytic evaluation of a closed queueing network model.

2.2 Decomposition

Decomposition is the process of partitioning a model into a number of smaller submodels. Each of these submodels is evaluated analytically, and those individual solutions are combined to obtain the solution of the original model. Each of these submodels also can be partitioned, so this method is also called hierarchical modelling.

A decomposition algorithm can be used to solve a closed, separable queueing network model.

The advantages of decomposition are:

- to reduce the size of the model and
- to simplify analytical evaluation.

The general approach of decomposition methods for queueing network follows these steps:

- decompose the network into appropriate sub-network;
- solve each sub-network independently;
- aggregate the sub-network solution to form an approximate solution of the entire network;
- validate the aggregate solution with a simulation model or measurement results.

The detailed algorithm of decomposition is presented in the book[1].

FESC

In a decomposition, some service centers are grouped together, then replaced by a special type of service center called a Flow Equivalent Service Center (FESC). The FESC is a load dependent service center. This means that the service rate is a function of the number of jobs waiting at the center.

DESC (Surrogate delay)

Surrogate delay is a simple decomposition technique which uses a Delay Equivalent Service Center (DESC) instead of a FESC. The technique can be used for queueing network model which includes simultaneous or overlapped possession of resources. Simultaneous resource possession poses difficulties in analytical modeling because it violates certain assumptions (section 2.3) that are essential to separable or product form queueing network model [6].

The service time of DESC can be determined by an iterative procedure. Here is an example to show how to use iterative procedure to get service time of the DESC from a database system.

The database system includes workstations, database server, and a disk. The transactions perform two kind of operation: read and write. For write, the transaction must hold the mutex first, then write, after write, the transaction return the mutex. To obtain the solution of the database system, we use surrogate delay.

Figure 2-3 shows a model (call it as model 1) for the above database system. The model has a critical section controlled by a mutex flag. The mutex control is shown as a service center where a non-mutex-holding job may go (shown by a solid arrow arriving at the center) and wait to obtain the mutex flag. The event (a) shows procedure that a job waits (FCFS) to get mutex flag (flag set to 1) then leaves, following the dashed arrows around the system as it executes in the critical section. When it returns to the mutex control to release the flag (Delay) at event (b). The dashed path from (a) to (b), which passes through the database server and

the disk some number of times, represents the execution path of the critical section. The solid path from (b) to (a) represents the other operation (such as read). We need to determine the queueing time to obtain the flag, but the mutex controller is not a queueing network server of a standard kind.

Figure 2-4 shows how approach this problem by DESC. We introduce another model (model 2), which only has 2 service centers: a "hold mutex" service center and a "non-hold mutex" service center. The hold service center is FCFS, such that only one job can be in service at a time, with a service time equal to the mean path from (a) to (b); the non-hold service center is a DESC with mean delay equal to the mean path from (b) to (a). To solve this model, we must determine the service times of the two service centers.

FIGURE 2-3 Model 1: Database system with mutex control

FIGURE 2-4 Model 2: Surrogate delay (EDSC) model

The model in Figure 2-4 gives another picture of the system. When a transaction holds the mutex, so it stays at the hold service center; when transaction does not hold the mutex, it stays at the non-hold service center. Only one transaction can hold the mutex, so hold service center has queue and it is FCFS.

Combining these two models, we can obtain the solution. The process is shown at Figure 2-5.

First input parameters to model 1, the mutex waiting time is an arbitrary starting value, perhaps zero (it is unknown at that time). From the solution of model 1, make parameters of model 2, e.g. the service time of hold service center and non-hold service center of model 2 are determined by the results of model 1.

After getting the solution of model 2, compare the solution with the input parameters. If the input parameters which are determined by the solution of mode 1 are close enough to the solution of model 2 (relative error is 0.1% or less), the solutions of model 1 and model 2 are considered as final solution. Else the mutex waiting time of model 1 is determined by the solution of model 2, e.g. the mutex waiting time of model 1 equals the

waiting time of hold service center of model 2. The input all parameters to model 1 (now, the mutex waiting time is known), and repeat.

This sort of iterative procedure is quite robust in performance calculations, and usually converges quickly.

FIGURE 2-5 Iterative procedure to get service time of DESC

2.3 Algorithms

2.3.1 Product form queueing network

The solution of a queueing network model is a set of performance measures that describe the time averaged (or long term) behavior of the model. Computing these measures is quite expensive and complicated. But for product form queueing network model, solution can be obtained simply.

The product form queueing network is a series of M single-server queues with exponential service time and poisson arrivals, the jobs leaving a queue immediately join the next queue. The joint probability of queue length of M queues can be computed simply by multiplying individual probabilities [7].

Product form networks are easier to analyze than non product form networks. Product form solution exist for an broader class of network which must satisfy the following 5 criteria:

1. **Service Disciplines:** All service centers have one of following four type of service disciplines: First Come First Served (FCFS), Processor Sharing (PS), Infinite Server (IS or delay), and Last Come, First Served Preemptive resume (LCFS-PR).
2. **Job Classes:** The jobs belong to a single class while awaiting or receiving service at a service center but may change classes and service centers according to fixed probability at the completion of a service request.
3. **Service Time Distribution:** At FCFS service centers, the service time distribution must be identical and exponential for all classes of jobs. At other service centers, where the service times should have probability distributions with rational Laplace transforms, different classes of jobs may have different distribution.
4. **State-dependent Service:** The service time at FCFS service center can dependent only on the total queue length of the center. The service time for a class at PS, LCFS-PR, and IS centers can also dependent on the queue length for that class, but not on the queue length of the other classes.

-
5. **Arrival Process:** In open networks, the time between successive arrivals of a class should be exponentially distributed. No bulk arrivals are permitted.

2.3.2 Mean value analysis (MVA) [1]

The mostly used algorithm to evaluate closed queueing network is mean value analysis (MVA).

The performance parameters determined by MVA are:

- **throughput (X):** number of job are finished in unit time (one second, for example)
- **residence time (R):** the total time of a job stay at a service center in a life cycle (the job can visit the center more than once)
- **utilization (U):** for a service center, U equals to total working time divide by total time ($U \leq 1$)

MVA for a single-class model is based on the following 3 equations:

$$X(N) = \frac{N}{Z + \sum_k R_k(N)} \quad (\text{EQ 1})$$

where $X(N)$ is the system throughput, $R_k(N)$ is the residence time at service center k when there are N jobs in the system, Z is the think time of the user (it can be 0).

$$Q_k(N) = X(N) R_k(N) \quad (\text{EQ 2})$$

where $Q_k(N)$ is job length at service center k .

$$R_k(N) = D_k \quad (\text{delay center}) \quad (\text{EQ 3})$$

$$R_k(N) = D_k(1 + A_k(N)) \quad (\text{queueing center}) \quad (\text{EQ 4})$$

where D_k is the service demand at service center k (service time multiple by visit ratio). For delay service center $R_k = D_k$ (first solution). For queueing service center (second solution), $A_k(N)$ is the average number of job seen at k when a new job coming.

Evaluating a model, the first thing is to determine $A_k(N)$. There are two basic techniques, exact and approximate.

The exact solution is based on the form:

$$A_k(N) = Q_k(N-1) \quad (\text{EQ 5})$$

This means that queue length seen at arrival to a queue when there are N jobs in the system is equal to the time averaged queue length there with one less job in the system.

Mean value analysis is a recursive algorithm. Computation of performance with N jobs in the system requires knowledge of performance with $N-1$ jobs. Since performance with $N=0$ is well known, so we can start from $N=0$.

Here is exact MVA solution technique for 1 class job model.

```
FOR k = 1 to K do Qk = 0
FOR n = 1 to N do
BEGIN
  FOR k = 1 to K do Rk = {  $\frac{D_k}{D_k(1 + Q_k)}$       delay
                           queueing
  X =  $\frac{n}{Z + \sum R_k}$ 
  FOR k = 1 to K do Qk = XRk
END
```

The exact MVA algorithm for multi-class job model is introduced in [1].

Even for 1 class job model, exact MVA algorithm needs to evaluate 2K+1 quantities (i.e. R_k, X, Q_k) for each population between 1 and N. In order to evaluate more efficiently, we can use approximate MVA.

The approximate MVA is based on the equation:

$$A_k(N) \approx h[Q_k(N)] \quad (\text{EQ 6})$$

for some suitable function h, a more efficient, iterative algorithm can be obtained.

One of the important approximate algorithms is Schweitzer-Core algorithm. The algorithm uses information about population N to “guess at” statistics for population N-1 and the uses SI-MVA algorithm to compute new statistic for population N [1][6].

2.3.3 Heuristic algorithm: Linearizer [6]

The Linearizer algorithm is an improvement over the Schweitzer-Core algorithm by obtaining more accurate estimate of $D_k(N)$ (see equation 3 and 4). It invokes the Schweitzer-Core algorithm as a subroutine and compute its own $D_k(N)$. It assumes that $D_{mk}(N - e_i) = D_{mk}(N)$ which implies that h (see equation 6) is a linear function of population, hence the name Linearizer.

Here is algorithm of the Linearizer for a multi-class model.

Linearizer

INPUT

K : number of class of job

M : number of service center

v_{mk} : number of visit class k job to center m

s_{mk} : mean service time class k job at center m

Q_m : type of queue (delay or queueing)

N_k : number of jobs of class k

for all m, k

STEP 0 (Initialize)

All jobs are uniformly distributed among all the queues, i.e. $L_{mk}(N) = N_k/M$, L_{mk} is number of class k job at service center m

Assume that $L_{mk}(N - e_i) = (N - e_i)_k/M$, here i can be any class

Set $D_{mk}(N) = 0$, where D_{mk} is the demand of a class k job for service center m at population N

Assume $D_{mk}(N - e_i) = 0$, here i can be any class

Set index $I = 1$

STEP 1

Apply the Schweitzer-Core algorithm to population N :
input $D_{mk}(N)$ and $L_{mk}(N)$, then get solution of the Schweitzer-Core algorithm

If index $I = 3$ then stop; otherwise go to STEP 2.

STEP 2

Invoke the Schweitzer-Core algorithm for each $(N - e_i)$ population. Use the most recent available estimates of $L_{mk}(N - e_i)$ and $D_{mk}(N - e_i)$ as input.

STEP 3

Compute estimate function $F_{mk}(N)$ and $F_{mk}(N - e_i)$ for each class:

$$F_{mk}(N) = L_{mk}(N) / N_k$$

$$F_{mk}(N - e_i) = L_{mk}(N - e_i) / N_k$$

and compute estimates of $D_{mk}(N)$ for each class:

$$D_{mk}(N) = F_{mk}(N - e_i) - F_{mk}(N)$$

Then assume that $D_{mk}(N - e_i) = D_{mk}(N)$.

Set $I = I + 1$. Then goto STEP 1.

OUTPUT

$L_{mk}(N)$: mean number of class k job at m

$W_{mk}(N)$: mean waiting time k at m

$Y_{mk}(N)$: throughput of class k at m

$U_{mk}(N)$: unilization of class k at m

for all m, k

A Model Exploiting the Symmetry of a Large Client-Server System

This chapter briefly describes the database system to be studied, the available data, and a simplified abstract symmetrical structure on which we can base an understandable capacity planning model.

3.1 Performance measurements of the database system

Performance measurements have been done on the database system. The measurements record traffic at each LAN, in two real parts: one part recording all transactions to RF database (RF system data), another one recording all transactions to BC database (BC system data).

3.1.1 Components of the data

The main components of data to be measured are shown in Figure 3-1.

Note: transaction time = host time + workstation time

FIGURE 3-1 Main components of workstation data

A transaction starts at t_0 , processing time is the time used to set up transaction program at work station. After set up the transaction program, the moment at which user press <Return> is t_1 . The host time is the time to wait transaction go to the database and come back to the work station, i.e., in the host time, the work station does nothing. The host time includes: message transmission time, waiting time and service time. The work station time is the screen initialization time, after it, the next screen appears and the new transaction will start at t_3 (like t_0).

Following are attributes of the data which collected every week:

- geographic area (domain)
- transaction type (gclcmd name and screen name)
- number of querying (count) of the week
- average "transaction time" of the week (average of $t_3 - t_1$)
- total transaction time of the week.
- average host time of the week (AVEHOST, average of $t_2 - t_1$)
- total "host time" of the week
- average workstation time of the week
- total workstation time for of week

-
- average processing time of the week
 - total processing time of the week

The parameters used for modeling the database system are based on these data.

With reference to Figure 1-1, t_1 is the time a request is sent from the workstation, t_2 is the time the reply is received, t_3 is the time the user is able to enter input for the next request, and t_3 becomes t_0 for the next cycle. The value of this data therefore is to capture the user "think and type" time from t_0 to t_1 , and the response time of the rest of the system (t_1 to t_2).

3.1.2 Transaction types

The system accepts transactions which can be classified by the screen type and the command type, into a very large number of different classes of transaction. It was felt that this was far too much detail for planners to work with, and for a reasonably simple solver to compute the performance for the separate classes, so an effort was made to extract relatively common transactions, and to cluster transactions with similar parameters as seen from the workstation into a few classes.

For clustering, we examined the data and found that some of transactions are similar (average processing time, average host time, average workstation time, average transaction time). So it is possible to use fewer number of transaction types to represent all transactions.

We process the measurement data and get:

- General information of the database system which includes: total count and average throughput, total host time and average host time, total transaction time and average transaction time
- General information of each domain
- More frequent transactions (count) in each domain
- More time summing transactions (host time and transaction time) in each domain
- Detailed information of parts of transactions

According to the results, we chose 5 significant transactions from RF system data and another 5 significant transactions from BC system data.

The number of the 5 transactions from the RF system data are ~22% of the total transactions of RF system data. The number of the 5 transaction from BC system data are ~45% of the total transactions of BC system data.

The transaction types for the model are based on these significant transactions.

Table 3-1 describes the five “significant” transactions from RF database system.

DATE	TYPE	COUNT(/week)	AVEHOST(second)
week1	R1	204847	1.05
week1	R2	56642	1.16
week1	R3	63473	1.114
week1	R4	138945	0.48
week1	R5	56916	0.63
week2	R1	276156	1.07
week2	R2	71582	1.16
week2	R3	69954	1.115
week2	R4	196143	0.49
week2	R5	62312	0.65

TABLE 3-1 Five significant transactions of RF database.

Table 3-2 describes five "significant" transactions from the BC database system.

DATE	TYPE	COUNT(/week)	AVEHOST(second)
week1	B1	44859	1.73
week1	B2	51247	1.05
week1	B3	17774	1.22
week1	B4	41349	1.34
week1	B5	22483	0.95
week2	B1	48472	1.63
week2	B2	55487	0.99
week2	B3	21182	1.16
week2	B4	63955	1.31
week2	B5	47645	0.91

TABLE 3-2 Five significant transactions of BC database.

By the average host time (AVEHOST), these transactions approximately can be classified to two parts: one with longer host time (1.0 second for RF system data and 1.65 second for BC system data) and another one with shorter host time (0.5 second for RF system data and 1.0 second for BC system data).

According to the above analysis, we further clustered the transactions into just two types for each database:

- heavy transaction to RF database: T-H-R
- light transaction to RF database: T-L-R
- heavy transaction to BC database: T-H-B
- light transaction to BC database: T-L-B

FIGURE 3-2 Path of transaction in the database system

Figure 3-2 shows the path of the transactions in the system.

We assume that the processing time, workstation time, LAN delay, LAN server service time, and the backbone delay or WAN delay are the same for the four type of transactions; the only difference for these 4 types of transactions is the database server service time. The host time includes transmission time, queueing delay time and service time.

Although we do not have data on it, there is also the possibility of a fifth type of transaction which goes only to the local server, to log in, set connection parameters, and (potentially) to retrieve locally held data. This is designated as a Local Transaction, and may be heavy or light.

- light transaction to local server only: T-L-L.
- heavy transaction to local server only: T-H-L.

3.2 Performance modeling of the database system

3.2.1 Modelling assumptions

to separate database system from the other applications

The database system to be studied is not an isolated system, there are non-database applications in the system. To set up a model for the database system, the first thing is to separate servers of the database system from the other servers.

The non-database applications affect the performance of the database system, such as LAN traffic, mainframe computer CPU utilization. All these effects are represented by parameters of the model which will be described later.

to reduce the number of service center of the model

There are more than hundred of LANs and several thousands of workstations in the database system. If each of them is represented by an individual service center, there will be thousands of service centers and the model will be very difficult (or even impossible) to solve by analytical methods or simulation.

To reduce the number of service centers, and to take advantage of the symmetric aspects of the system (some of workstations have similar work load, some of LANs have similar structure), a partially symmetric model is used to represent the system.

simplification of the system

The model is based on the following simplifications:

- All LANs in this system are token rings.
- The number of workstation in each token ring is the same for the whole system.

-
- In the local area (in which token rings are connected to the backbone), all token rings have the same traffic factor and workload.
 - In the remote area (in which token rings are connected to the WAN), all token rings have the same traffic factor and workload

parameters of the model

In the model, the following service parameters will be adjustable:

- the number of workstation in a token ring: N
- the number of token rings in the local area: M_{BB}
- the number of token rings in the remote area: M_{WA}
- the local area token ring traffic factor (the utilization of each ring connected to the backbone): ρ_{BB}
- the remote area token ring traffic factor (the utilization of each ring connected to the wide area net): ρ_{WA}
- the utilization of the mainframe computer: ρ_U
- the number of workstations in each ring which are currently using each server (RF database, BC database, or the LAN server)
- the service time of each server

Figure 3-3 shows structure of the database system.

Figure 3-4 shows the model based on the above assumptions.

Section 3.2.2 describes each service center of the model.

FIGURE 3-3 Structure of the database system

FIGURE 3-4 Partially symmetric model of the database system

3.2.2 Service centers of the model

workstation

Since there is no queue at each workstation, a workstation can be considered as a delay server. In this model, one service center is used to represent all work stations in whole system.

A transaction program is set up at workstation, the time for this process is the processing time in the measurement data. After the transaction comes back to workstation, the workstation initialize for the next screen. This time is the workstation time in the measurements data.

For a transaction, if it is not the first transaction in a workstation, we can consider that the total preparing time is the workstation time (previous transaction) plus the processing time (self). To simplify the parameter, in this model, the preparing time to represent the processing time and workstation time. In this model, the preparing time is fixed for all transactions. Its value equals the average processing time plus the average workstation time. All transactions have the same preparing time.

FIGURE 3-5 Service center of workstation

token ring LAN

The token ring LAN in the system does not deal only with the database system transactions, the other applications also use the token ring. The effects of other applications on token ring LAN are determined by Equation 9[9].

The processing rate of the token ring used in calculation is of 4 Mbit/second.

The calculation is based on the following assumptions:

- the length of each token ring LAN is 2 km
- propagating rate is 0.000005 second/km
- every station transmits 1 package once and the length of the package is 1000 bit
- token is 24 bit

The delay in a token ring is determined by:

$$\text{delay} = \text{processing} + \text{waiting} \quad (\text{EQ 7})$$

$$\text{processing} = (\text{token} + \text{data}) / \text{rate} \quad (\text{EQ 8})$$

where $\text{token} = 24$ bit, $\text{data} = 1000$ bit, $\text{rate} = 4\text{M}$ bit/second

The waiting (W) is determined by:

$$W = \frac{R(1 + \frac{\rho}{N}) + N\lambda\tau^2}{2(1 - \rho - \lambda R)} \quad (\text{EQ 9})$$

where R is latency of the token ring, ρ is traffic factor (utilization of token ring), N is number of workstation connected to token ring LAN, λ is number of jobs arriving for each workstation per second, τ is processing time of one job.

In our assumption, we have:

$$\tau^2 = \bar{\tau}^2 = (\text{processing})^2 \quad (\text{EQ 10})$$

and

$$\rho = N\lambda\bar{\tau} = N\lambda(\text{processing}) \quad (\text{EQ 11})$$

Finally, we get:

$$\text{waiting} = \frac{\text{latency} \left(1 + \frac{\rho}{N}\right) + \rho \cdot \text{processing}}{2 \left(1 - \rho - \frac{\rho \cdot \text{latency}}{N \cdot \text{processing}}\right)} \quad (\text{EQ 12})$$

N is number of workstations at each token ring LAN; ρ is the traffic factor (utilization of token ring). N and ρ represent traffic for the database system application plus the other "external" applications, and they can be determined by user.

Latency is determined by:

$$\text{latency} = 5 \times 10^{-6} (\text{second/km}) \cdot 2\text{km} + N/\text{rate} \quad (\text{EQ 13})$$

In this model the service center representing the token ring LAN is a delay server with the value of delay given by EQ 7.

FIGURE 3-6 Service center of token ring LAN

LAN server

For the transactions, the LAN server is like a switch. The service time for each transaction is fixed (at 0.01 second, a value reported to us as the representative value for its operation).

In the real system, the LAN servers also do some work for local jobs. These local jobs do not go to the mainframe computers, but they affect the performance of the local server. The transaction types T-L-L and T-H-L describe these local jobs, and their service time at the local server must be estimated by the user of the model.

Each token ring LAN has one LAN server, all of the LAN servers ($M_{BB} + M_{WA}$) should be FCFS server. Surrogate delays (section 2.3) are used to model these LAN servers; they are represented by 4 service centers:

- 2 FCFS servers and the
- 2 DESC servers

For M_{BB} LAN server (connected to the backbone), a FCFS service center represents a LAN server and a DESC server represents other ($M_{BB}-1$) LAN servers, as shown in Figure 3-4.

For M_{WA} LAN server (connected to WAN), a FCFS service center represents a LAN server and a DESC server represents other ($M_{WA}-1$) LAN servers, as shown in Figure 3-4.

FIGURE 3-7 Service center of LAN server

The process to get service time of the LAN servers (DESC) is shown in figure 3-8.

It includes the following steps:

- **initialize** - set service time for FCFS and DESC service center
- **solve** - execute the model with initialized values
- **set value** - the service time of the DESC = the response time of the FCFS
- **solve** - execute the model for the new value
- **check** - compare the current response time of FCFS with the service time of DESC
- if the relative error $< 0.1\%$ goto **end**, else goto **set value**

FIGURE 3-8 Procedure to get service time of LAN server (DESC)

BackBone and WAN

There is some external traffic in BackBone and WAN. In this model, the parameter: ρ (external traffic factor) is used to represent the effect of the external traffic.

The response time back bone or WAN to transaction is:

$$response = \frac{processing}{1-\rho} \quad (EQ 14)$$

Here ρ can be different for BackBone and WAN and the ρ is decided by user.

The BackBone (FDDI) and WAN are represented as delay server.

FIGURE 3-9 Service center of BackBone and WAN

database server

There are two mainframe computers in the system. One is for RF system data, one is for BC system data.

A transaction first goes to the FEP (front end processor), goes through the high speed cable (150Mbit/second), then goes to the corresponding main-frame computer.

In this model, all these processes are put together and represented by the database server. There are two service centers in this model for the database server: RF database server and BC database server. Each server has different service time for heavy transaction and light transaction. All these service times are determined by user.

In the real system, the mainframe computer for BC system data also works for other applications and these applications take more than half of the CPU time. A parameter r_u gives the utilization by these external applications. In the model, the effective service time of the mainframe computer for BC system data becomes:

$$effectiveServiceTime = \frac{serviceTime}{1 - r_u} \quad (EQ 15)$$

The two databases are represented each by a FCFS server.

FIGURE 3-10 Service center of database server

Table 3-3 gives visit ratio of the transactions for the service centers.

TYPE	work-station	LAN	LAN server	back bone or WAN	R-db	B-db
heavy tran. to R-db	1	2	2	2	1	0
light tran to R-db	1	2	2	2	1	0
heavy tran. to B-db	1	2	2	2	0	1
light tran to B-db	1	2	2	2	0	1
local job	1	2	1	0	0	0

TABLE 3-3 visit ratio of the transaction for each service center

3.2.3 3 - layer model

To summarize, the simplified planning model for the existing system has the structure shown in Figure 3-3, with M local area networks each with N workstations. There is a set of M_{BB} LANs connected by the backbone FDDI net and a set of M_{WA} LANs connected by wide area network ($M_{BB} + M_{WA} = M$). There are 6 transaction types from any workstation, combining "heavy" or "light" requests to the RF database, BC database or LAN server. There is external traffic competing for the networks, front-end processors and servers, at stated levels of utilization of these resources.

In this model, each client has three layer of servers:

- workstation
- LAN server
- database server

This model is also called 3 - layer model.

3.2.4 Model validation

Model validation is done at a small database system.

FIGURE 3-11 Database system for validation

in Figure 3-11, Each LAN is connected to 50 workstations.

Using QNAP2[11], performance analysis of the small database system is done by:

- simulation
- Heuristic algorithm with separate LAN server (without DESC)
- Heuristic algorithm with aggregate LAN server (with DESC)

For simulation (QNAP2):

- confidence level is 95%
- replication method is used and number of replication is 10
- mean simulation time is 5000

We compare average response times (for each service center) obtained from simulation and the analysis methods, the relative error $< 1\%$.

Heuristic algorithm (without and with DESC) is also done by linearizer[8]. We compare response times obtained from the two analysis methods (without and with DESC). the relative error $< 0.1\%$.

Convergence of Heuristic algorithm (with DESC) is fast for both QNAP2 and linearizer, interactive step < 5 .

3.3 A second, modified architecture of 4 - layer model

Looking into the future we wish to consider how the potential bottleneck at the mainframe server might be alleviated. Additional servers might share the network in various ways:

replicate the database

There are multiple similar servers. This leads to problems in maintaining consistent information.

partition the database

Certain frequently-used data is maintained separately, perhaps in multiple copies. If there is only a small amount of such data the consistency problem is somewhat simplified.

partition the system

The certain read-only data is maintained in duplicate form on "assistant" servers. This would be updated periodically, perhaps daily from the mainframe.

For each client, there are 4 layer of servers:

- workstation
- LAN server
- database server
- intermediate server

The 4-layer model considered here is intended to describe the second or third of these possibilities. In response to case study results showing (for particular parameters chosen) a potential bottleneck at the RF database server, a single "intermediate server" was introduced to share some of the RF database transaction load. Figure 3-13 shows one way of accessing it.

The LAN server or the workstation determines whether a request should be routed to the intermediate server.

There are 3 additional parameters for the 4 - layer model:

- fraction: fraction of the RF database transactions that go from each workstation to intermediate server
- service time of the intermediate server for heavy transactions
- service time of the intermediate server for light transaction.

The intermediate server is a FCFS service center, the other service centers are the same as in the 3 - layer model.

FIGURE 3-12 Service center of intermediate server

The visit ratio is changed for some service centers in 4 layer server model (see Table 3-4).

TYPE	R - db	B -db	int. server
heavy tran. to R-db	1 - fraction	0	fraction
light tran to R-db	1 - fraction	0	fraction
heavy tran to B-db	0	1 - fraction	fraction
light tran. to B-db	0	1 - fraction	fraction

TABLE 3-4 visit ratio of 4 layer server mode

FIGURE 3-13 The 4 - layer server model

3.4 Capacity planning tool: CPT_SCSN

A prototype tool, CPT-SCSN (Capacity Planning Tool for Symmetric Client-Server Network), was written to describe this particular system, to present the parameter choices in terms of the system under study.

CPT-SCSN uses queueing network model as a basic theoretical model. It is assuming that the system satisfies the five conditions which were mentioned at 2.3.1.

The prototype assumes the system structure of Figure 3-3 or Figure 3-4, and allows complete freedom in sizing the workload by number of LANs or workstations, or the execution time parameters of workstations, local servers, or the intermediate server. Extensions to more general configurations of local server would be straightforward.

The prototype was written in TCL/TK, a new prototyping system created by Ousterhont. It calls an advanced queueing solver written in C++ by G Franks[8], and uses a decomposition and surrogate delay approach to give efficient solutions. In effect just one local area net is solved and others are assumed to have identical parameters and performance (Section 3.2.2).

3.4.1 Instructions for use

The tool is intended to be self-documenting, with the nine different screens shown in the appendix of this thesis. Graphic User Interface is implemented for it.

The name of the capacity planning tool is: `cpt_scsn`. After typing: `cpt_scsn`, the main window will appear on the screen.

There are four parts in the main window:

1. README

This part handles the usage information for the tool, it includes:

-
- **usage** gives the options of the other three parts of the main windows
 - **input** gives usage information about the input part
 - **solver** gives usage information about the solver part
 - **output** gives usage information about the output part

2. INPUT

This part deals with input parameters.

User can set parameters by two ways: using an input file or setting parameters at GUI. User also can edit the input file at GUI.

Default input file is **input_file**

3. SOLVER

User selects solver model (3 layer server model or 4 layer server model) in this part.

4. OUTPUT

This part deals with the output of the modelling results. There are three ways to get the output:

- put the output to an output file
- show the output at GUI
- print the output

Default output file is **output_file**

3.4.2 System requirements

To run `cpt_scsn` needs the following files:

- `cpt_scsn`
- `usage`
- `browse_input`
- `edit_file`
- `edit_help`
- `edit_input`
- `input_help`
- `input_p_help`
- `model_help`
- `output_help`
- `parameters_input`
- `put_to_file`
- `save_to_file`
- `show_output`
- `3_level_model`
- `4_level_model`

For the last two files, the source code (C++) are put at the appendix.

To run `cpt_scsn` also needs the TCL/TK run-time software which consists of:

- TCL library
- TK library
- executable file: `wish`

The information about TCL/TK can be obtained from [10] and:

<http://web.cs.ualberta.ca/~wade/HyperTcl/>

also

http://www.x.co.uk/of_interest/tcl/Tcl.html

4.1 Assumptions and parameters.

From examining the data it seems likely that only a fraction of the workstations are operating in a tight “think-compute” cycle at any one time. Presumably each individual workstation has periods of RF database/BC database activity, interspersed with periods of other kinds of computation (e.g. word processing) or other work. We have the choice of modelling this by choosing a subset of active workstations, or of increasing the average think time.

For this case study, various reduced numbers of active workstations were used and each one dedicated to a server:

N-RDB =	workstations accessing	RF database: 100 to 400
N-BDB =	workstations accessing	BC database: 20 to 80
N-LOC =	workstations accessing	the local server only.

The ratio $N\text{-RDB}/N\text{-BDB} = 5$ was used in all the cases.

In this case, number of token ring LANs connected to the backbone is the same as that connected to WAN, i.e., $M_{BB} = M_{WA}$ (section 3.2.1).

The service time parameters used (except where noted) were

RF database server	heavy transactions	50%
	light transactions	50%
BC database server	heavy transactions	50%
	light transactions	50%
Local server	heavy transactions	50%
	light transactions	50%

Fraction of heavy transactions at all workstations = 0.5.

External traffic factors (partial utilization) at

- local networks: 0.5
- BackBone: 0.5
- WAN: 0.5
- BDB servers: 0.5

4.2 Performance analysis

4.2.1 Performance of database server

Figure 4-1 and 4-2 show performance of RF database server and BC database server, respectively.

The workload of each token ring LAN is fixed:

- 10 work stations are connected to RF database
- 2 work stations are connected to BC database

The total workload connected to the database server is dependent on the number of token ring LANs in the system (M_{BB} and M_{WA}). Figure 4-1 and 4-2 show the case that the number of token ring is 10, 20, 30 and 40, respectively.

The results indicate that the bottleneck of the system is the RF database server.

In Figure 4-1: for service time at 0.04s(50%) 0.02s(50%), the response time is 0.1s (for total 100 client), and the response time is 0.2s (for total 200 client).

FIGURE 4-1 Performance of RF database server

FIGURE 4-2 Performance of BC database server

4.2.2 Performance of LAN server

The following three figures show the performance of the LAN server in various conditions.

Figure 4-3 shows response time of the LAN server for different lan traffic factor. The workload (10 work station connected to the local server) and the service time (0.03s) are fixed.

Figure 4-4 shows response time of the LAN server at the different service time. The workload (10 work station connected to the local server) and the traffic factor (0.8) are fixed.

Figure 4-5 shows response time of the LAN server at different workload. The traffic factor (0.8) and the service time (0.05s) are fixed.

The results show that the response time of the LAN server is acceptable even that the workload is very high (30 work station are connected to it) or the traffic factor is very high (up to ~ 0.9)

FIGURE 4-3 Performance of LAN server (traffic factor)

FIGURE 4-4 Performance of LAN server (service time)

FIGURE 4-5 Performance of LAN server (workload)

4.2.3 Intermediate server

Figure 4-6 shows response time of RF database server in two cases: with and without intermediate server.

The workload is:

- 10 workstations connected to RF database system for each token ring LAN
- the number of token ring LANs is 10, 20, 30 and 40, respectively

The fraction is 0.2, it means that 20% of workload is moved from the RF database server to the intermediate server.

It is clear that the performance of the system is improved by adding an intermediate server.

Figure 4-7 shows the relationship between performance and fraction.

In Figure 4-6, at the case with intermediate server, the response time is 0.21s (for total 100 client); at the case without intermediate server, response time is 0.31s (for total 100 client).

FIGURE 4-6 Performance with and without Intermediate server

FIGURE 4-7 Effect of fraction on performance

4.2.4 Analysis

There are three ways to improve the performance of the system.

to improve performance of RF database server

The bottleneck of the system is the RF database server, to improve performance of the RF database server is the key point for the system. Figure 4-1 shows that if the service time of the RF database server can be reduced, the response time of the RF database system is acceptable at higher workload.

to move part of workload to the local server

The performance of the local server shows that the local server can afford the higher workloads. To remove part of workload from RF database server to the local server can improve the performance of the system.

to add an intermediate server

To add an intermediate server is another way to reduce workload of the RF database server.

To remove workloads from RF database server to other servers (local server or intermediate server) will be concerned with replication the database or partition the database. Using intermediate server will be easier to maintain the database consistent. But using the local server will be cheaper.

5.1 Advantages of the Partially Symmetric Model

The partially symmetric model can be used for a large system

For the most of the analysis tools (based on queueing network model), the limitation of number of service center is much less than 1000 (usually about 100).

There are more than thousand of devices (PC, work station, mainframe computer, lan server, wan server, network) in the database system, it is impossible to put all these devices as service centers to a queueing network model.

The partially symmetric model deals with thousands of service centers and hundred type of transactions by symmetric simplification. In the partially symmetric model, the thousands of devices are represented by about 20 service centers with different service types (Delay server, FCFS server); hundred types of transactions are represented by 5 type of transactions.

The partially symmetric model is very efficient.

The calculation of the partially symmetric model is based on queueing network model. For one group of parameters, to get the results takes a few minutes. The results include:

- response time of each service center for different type of transactions;
- system response time for different type of transactions;
- utilization of each service center;
- throughput of each service center.

The case study (Chapter 4) shows that the partially symmetric model can be used for performance analysis and capacity planning study.

The partially symmetric model is easy to use, because

- structure of the model is easy to understand;
- on-line help is available.

5.2 Capacity Planning Tool

An easy to use of capacity planning tool (CPT_SCSN) for experimenting with parameters:

- customized to the local structure and very portable
- easily changed, because written in the modern supporting language TCL/TK
- easily extended

5.3 Experimental Results

The studies described in chapter 4 are limited by the fact that actual parameters were not available for all the same times. However they demonstrates the use of the tool.

To the external that they response the real system, they show:

-
- that RF database server is more sensitive to load than BC database server, so that 200 active users at one time produce significant delay (several seconds) even with favorable assumption on database servers delays (Figure 4-1 and 4-2).
 - that the cluster size (number of work station per local server) cannot be greatly increased without noticeable delay (Figure 4-5).
 - similarly, that the local server execution time per response cannot be increased either, without a noticeable impact (Figure 4-4)
 - that overall lan traffic isn't a factor, up to the 80% saturated level (Figure 4-3).

That is, the RF database server is a limiting factor and the local server is near to second in a limiting factor, in further growth.

To afford the database servers an intermediate server was modelled, which can take 20% of the RF requests (Figure 4-6). One such server can increase the system capacity by about 25 active RF users. If it could take 40% of RF requests, it would have much greater impact on response time (Figure 4-7).

5.4 Weaknesses and Future Work

This section discusses weaknesses of the partially symmetric model and the possible future work.

In the partially symmetric model, the thousands of devices are represented by about 20 service center, the hundreds type of transaction are represented by 5 type of transaction. It makes the system very simple and easy to calculate, but it also loses some information. The solver (Linearizer) used in this work can handle about 100 service centers and more than 20 type of transactions in a model, so the partially symmetric model does not reach the limitation of the solver.

To add more service centers and more transaction types to the model, so the model will includes more information about the real system and the model will represent more precisely the real system.

In this model, the structure of the system is fixed, only values of the parameters can be changed. Another future work is to make model more flexible.

To make the structure of the model being changeable and to set up the model by GUI. Therefore the model will be more powerful for performance analysis and capacity planning study, and the model also will be easy to control.

REFERENCES

- [1] E.D. Lazowska, J. Zahorjan, G.S. Graham, K.C. Sevcik, *Quantitative System Performance*, Prentice-Hall, Englewood Cliffs, N.J. 1984.
- [2] C.H. Sauer, K.M. Chandy, *Computer Systems Performance Modeling*, Prentice Hall, Englewood Cliffs, N.J. 1981.
- [3] J.A. White, J.W. Schmidt, G.K. Bennett, *Analysis of Queueing Systems*, Academic Press, New York 1975
- [4] S.S. Lavenberg, *Computer Performance Modeling HandBook*, Academic Press, New York 1983
- [5] K.M.Chandy and D. Neuse, Linearizer: A heuristic Algorithm for Queueing Network Models of Computing Systems, *Communications of the ACM February 1982 Volume 25 Number 2* 126-134
- [6] P.A.Jacobson and E.D.Lazowska, Analyzing Queueing Networks with Simultaneous Resource Possession, *Communications of the ACM February 1982 Volume 25 Number 2* 142-151
- [7] *Comput. Surv. 10.3 (Sept. 78)*, Special Issue on Queueing Network Models of Computer System Performance
- [8] Greg Franks, Layered Queueing Network Solver Software Design, *Systems and Computer Engineering Department, Carleton University* 1994
- [9] Dimitri Bertsekas, Robert Gallager, *Data Networks*, Prentice-Hall, 1992
- [10] John K. Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley Publishing Company, 1994
- [11] *QNAP2 Reference Manual*, Simulog, Saint-Quentin-Yvelines, France, 1985

APPENDIX A

The source code of partially symmetric model solver

1995/07/03

3_level C

This code is implemented for 3_level for non-gui user
input from standard input
output to specified file

no assistant server
add traffic at token ring
add random traffic at back bone (FDDI)
add random traffic at wan
add random utilization at mainframe of b-db
add local user (add 4 class)
add error check for total user <- NUM_WS

1995/07/22

local lan server service time can be set by user, using default value is same as before

total number of station is 11

```
1 ws (INF1)
2 local token ring delay (INF1)
3 remote token ring delay (INF1)
4 local lan server1 (FCFS)
5 remote lan server2 (FCFS)
6 local global delay1 (INF1)          only use 1 global delay
7 remote global delay2 (INF1)
8 r-db disk (FCFS)
9 b-db disk (FCFS)
10 u lan server1 (surrogate delay)    local
11 u lan server2 (surrogate delay)    remote
```

total number of class is 20

```
1 ws -> lan delay -> local server1 -> global delay1 -> r-db light (11)
2 ws -> lan delay -> local server1 -> global delay1 -> r-db heavy (12)
3 ws -> lan delay -> local server1 -> global delay1 -> b-db light (13)
4 ws -> lan delay -> local server1 -> global delay1 -> b-db heavy (14)
5 ws -> lan delay -> local server1 (15)

6 ws -> lan delay -> local server2 -> global delay2 -> r-db light (21)
7 ws -> lan delay -> local server2 -> global delay2 -> r-db heavy (22)
8 ws -> lan delay -> local server2 -> global delay2 -> b-db light (23)
9 ws -> lan delay -> local server2 -> global delay2 -> b-db heavy (24)
10 ws -> lan delay -> local server2 (25)

11 ws -> lan delay -> u local server1 -> global delay1 -> r-db light (31)
12 ws -> lan delay -> u local server1 -> global delay1 -> r-db heavy (32)
13 ws -> lan delay -> u local server1 -> global delay1 -> b-db light (33)
14 ws -> lan delay -> u local server1 -> global delay1 -> b-db heavy (34)
15 ws -> lan delay -> u local server1 (35)

16 ws -> lan delay -> u local server2 -> global delay2 -> r-db light (41)
17 ws -> lan delay -> u local server2 -> global delay2 -> r-db heavy (42)
18 ws -> lan delay -> u local server2 -> global delay2 -> b-db light (43)
19 ws -> lan delay -> u local server2 -> global delay2 -> b-db heavy (44)
20 ws -> lan delay -> u local server2 (45)
```

It is based on 'linearizer C' by Greg Franks

Yizhong Shen

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <iostream.h>
#include <fstream.h>
#include "mva.h"
#include "server.h"
#include "population.h"
#include "vector.h"

#define LIMIT 0.0001
#define CYCLE 4

#define TOKEN 24.0 // # of bit per token
#define DATAL 1000.0 // # of bit per data
#define TR_L 2.0 // # length of token ring (km)
#define SPEED 0.000005 // second/km
#define TR_RATE 4000000.0 // tr proc. data rate bit/sec
#define FDDI_RATE 100000000.0 // FDDI proc. data rate bit/sec
#define TRRT 0.01 // FDDI Target Token Rotation Time
```

double tr_delay(float, int); // f: traffic factor int: # of ws

main()

```
{
    unsigned classes = 20,
    unsigned stations = 11,
    PopVector NCust(0),
    Server** Q,
    MVA * model;
    char OUTPUT_FILE[20];

    int NUM_WS, // # ws per tr
    num_l_tr, num_r_tr, // user input

    int num_c_11, num_c_12, num_c_13, num_c_14, num_c_15, // user input
    num_c_21, num_c_22, num_c_23, num_c_24, num_c_25,
    int num_c_31, num_c_32, num_c_33, num_c_34, num_c_35, // auto
    num_c_41, num_c_42, num_c_43, num_c_44, num_c_45,

    double think = 15.0, // exp. data
    double l_lan_delay, // network
    double r_lan_delay, // network
    double g_delay1, // network
    double g_delay2, // network
    double lan_serv = 0.01, // exp. data
    double lan_r_serv = 0.01, // surou
    double lan_l_serv = 0.01, // surou
    double ass_lan_l_original = 0.03, // surou
    double ass_lan_r_original = 0.03, // surou
    double rdb_l,
    double rdb_h,
    double bdb_l,
    double bdb_h,
    double ass_lan_ser_t;

    float lan_traffic_l; // tr traffic (local)
    float lan_traffic_r; // tr traffic (remote)
    float fddi_random_traffic; // random traffic (FDDI)
    float wan_random_traffic; // random traffic (WAN)

    float random_u_bdb;

    float limit1, limit2, limit3, limit4, // for sur
    int count = 0, // for sur

    double host_time; // print out
    double tot_throu, // print out

    NCust.grow(classes+1),
    Q = new Server * [stations+1];

    cout << "\n Number of client in one token ring ";
    fflush(stdin);
    cin >> NUM_WS;

    cout << "\n Number of token ring at local area ";
    fflush(stdin);
    cin >> num_l_tr;

    cout << "\n Traffic factor of token ring at local area ";
    fflush(stdin);
    cin >> lan_traffic_l;

    cout << "\n Number of token ring at remote area ";
    fflush(stdin);
    cin >> num_r_tr;

    cout << "\n Traffic factor of token ring at remote area ";
    fflush(stdin);
    cin >> lan_traffic_r;

    cout << "\n Number of client go to r-db with light SQL at local tr ";
    fflush(stdin);
    cin >> num_c_11;

    cout << "\n Number of client go to r-db with heavy SQL at local tr ";
    fflush(stdin);
    cin >> num_c_12;

    cout << "\n Number of client go to b-db with light SQL at local tr ";
    fflush(stdin);
    cin >> num_c_13;

    cout << "\n Number of client go to b-db with heavy SQL at local tr ";
    fflush(stdin);
    cin >> num_c_14;

    cout << "\n Number of client need local server at local tr ";
    fflush(stdin);
    cin >> num_c_15;

    cout << "\n Number of client go to r-db with light SQL at remote tr ";
```

```

fflush(stdin);
cin >> num_c_21;

cout << "Number of client go to r db with heavy SQL at remote tr ",
fflush(stdin);
cin >> num_c_22;

cout << "Number of client go to b db with light SQL at remote tr ",
fflush(stdin);
cin >> num_c_23;

cout << "Number of client go to b db with heavy SQL at remote tr ",
fflush(stdin);
cin >> num_c_24;

cout << "Number of client need local server at remote tr ",
fflush(stdin);
cin >> num_c_25;

cout << "Service time of light SQL at r-db ",
fflush(stdin);
cin >> rdb_l;

cout << "Service time of heavy SQL at r-db ",
fflush(stdin);
cin >> rdb_h;

cout << "Service time of light SQL at b-db ",
fflush(stdin);
cin >> bdb_l;

cout << "Service time of heavy SQL at b-db ",
fflush(stdin);
cin >> bdb_h;

cout << "Service time of assistant lan server (default 0.0) ",
fflush(stdin);
cin >> ass_lan_ser_t;

cout << "Random traffic factor at WAN ",
fflush(stdin);
cin >> wan_random_traffic;

cout << "Random traffic factor at back bone ",
fflush(stdin);
cin >> fddi_random_traffic;

cout << "Random ut fr factor at b-db ",
fflush(stdin);
cin >> random_u_bdb;

l_lan_delay = tr_delay(lan_traffic_l, NUM_WS);
g_delay1 = (TTRT + (DATA_L + TOKEN + S)/FDDI_RATE)/(1-fddi_random_traffic);
r_lan_delay = tr_delay(lan_traffic_r, NUM_WS);
g_delay2 = 0.1/(1-wan_random_traffic);

if ((num_c_11 + num_c_12 + num_c_13 + num_c_14 + num_c_15) > NUM_WS) {
    cout << "Total workstation of per token ring is " << NUM_WS << "\n";
    exit(1);
}

if ((num_c_21 + num_c_22 + num_c_23 + num_c_24 + num_c_25) > NUM_WS) {
    cout << "Total workstation of per token ring is " << NUM_WS << "\n";
    exit(1);
}

num_c_31 = num_c_11 * (num_l_tr - 1);
num_c_32 = num_c_12 * (num_l_tr - 1);
num_c_33 = num_c_13 * (num_l_tr - 1);
num_c_34 = num_c_14 * (num_l_tr - 1);
num_c_35 = num_c_15 * (num_l_tr - 1);

num_c_41 = num_c_21 * (num_r_tr - 1);
num_c_42 = num_c_22 * (num_r_tr - 1);
num_c_43 = num_c_23 * (num_r_tr - 1);
num_c_44 = num_c_24 * (num_r_tr - 1);
num_c_45 = num_c_25 * (num_r_tr - 1);

NCust[1] = num_c_11;
NCust[2] = num_c_12;
NCust[3] = num_c_13;
NCust[4] = num_c_14;
NCust[5] = num_c_15;

NCust[6] = num_c_21;
NCust[7] = num_c_22;
NCust[8] = num_c_23;
NCust[9] = num_c_24;
NCust[10] = num_c_25;

NCust[11] = num_c_31;
NCust[12] = num_c_32;

```

```

NCust[13] = num_c_33;
NCust[14] = num_c_34;
NCust[15] = num_c_35;

```

```

NCust[16] = num_c_41;
NCust[17] = num_c_42;
NCust[18] = num_c_43;
NCust[19] = num_c_44;
NCust[20] = num_c_45;

```

```

bdb_l = bdb_l/(1-random_u_bdb);
bdb_h = bdb_h/(1-random_u_bdb);

```

```

/**
total number of station is 11

```

```

1  w s. (INF1)
2  local token ring delay (INF1)
3  remote token ring delay (INF1)
4  lan server1 (FCFS) local
5  lan server2 (FCFS) remote
6  global delay1 (INF1) local
7  global delay2 (INF1) remote
8  r db disk (FCFS)
9  b db disk (FCFS)
10 u lan server1 (surrogate delay) local
11 u lan server2 (surrogate delay) remote
**/

```

```

Q[1] = new Delay_Server(classes); // think
Q[2] = new Delay_Server(classes); // l_lan_delay
Q[3] = new Delay_Server(classes); // r_lan_delay
Q[4] = new FCFS_Server(classes); // lan_serv (local)
Q[5] = new FCFS_Server(classes); // lan_serv (remote)
Q[6] = new Delay_Server(classes); // g_delay1 (local)
Q[7] = new Delay_Server(classes); // g_delay2 (remote)
Q[8] = new FCFS_Server(classes); // disk_ac (r-db)
Q[9] = new FCFS_Server(classes); // disk_ac (b-db)
Q[10] = new Delay_Server(classes); // lan_l_serv
Q[11] = new Delay_Server(classes); // lan_r_serv

```

```

Q[1]->setService(1,think) setVisits(1,1); // work station
Q[1]->setService(2,think) setVisits(2,1);
Q[1]->setService(3,think) setVisits(3,1);
Q[1]->setService(4,think) setVisits(4,1);
Q[1]->setService(5,think) setVisits(5,1);
Q[1]->setService(6,think) setVisits(6,1);
Q[1]->setService(7,think) setVisits(7,1);
Q[1]->setService(8,think) setVisits(8,1);
Q[1]->setService(9,think) setVisits(9,1);
Q[1]->setService(10,think) setVisits(10,1);
Q[1]->setService(11,think) setVisits(11,1);
Q[1]->setService(12,think) setVisits(12,1);
Q[1]->setService(13,think) setVisits(13,1);
Q[1]->setService(14,think) setVisits(14,1);
Q[1]->setService(15,think) setVisits(15,1);
Q[1]->setService(16,think) setVisits(16,1);
Q[1]->setService(17,think) setVisits(17,1);
Q[1]->setService(18,think) setVisits(18,1);
Q[1]->setService(19,think) setVisits(19,1);
Q[1]->setService(20,think) setVisits(20,1);

```

```

Q[2]->setService(1,l_lan_delay) setVisits(1,2); // local tr delay
Q[2]->setService(2,l_lan_delay) setVisits(2,2);
Q[2]->setService(3,l_lan_delay) setVisits(3,2);
Q[2]->setService(4,l_lan_delay) setVisits(4,2);
Q[2]->setService(5,l_lan_delay) setVisits(5,2);
Q[2]->setService(6,0) setVisits(6,1);
Q[2]->setService(7,0) setVisits(7,1);
Q[2]->setService(8,0) setVisits(8,1);
Q[2]->setService(9,0) setVisits(9,1);
Q[2]->setService(10,0) setVisits(10,1);
Q[2]->setService(11,l_lan_delay) setVisits(11,2);
Q[2]->setService(12,l_lan_delay) setVisits(12,2);
Q[2]->setService(13,l_lan_delay) setVisits(13,2);
Q[2]->setService(14,l_lan_delay) setVisits(14,2);
Q[2]->setService(15,l_lan_delay) setVisits(15,2);
Q[2]->setService(16,0) setVisits(16,1);
Q[2]->setService(17,0) setVisits(17,1);
Q[2]->setService(18,0) setVisits(18,1);
Q[2]->setService(19,0) setVisits(19,1);
Q[2]->setService(20,0) setVisits(20,1);

```

```

Q[3]->setService(1,0) setVisits(1,1); // remote tr delay
Q[3]->setService(2,0) setVisits(2,1);
Q[3]->setService(3,0) setVisits(3,1);
Q[3]->setService(4,0) setVisits(4,1);
Q[3]->setService(5,0) setVisits(5,1);
Q[3]->setService(6,r_lan_delay) setVisits(6,2);
Q[3]->setService(7,r_lan_delay) setVisits(7,2);
Q[3]->setService(8,r_lan_delay) setVisits(8,2);
Q[3]->setService(9,r_lan_delay) setVisits(9,2);

```



```

Q[11]->setService(12,0,0) setVisits(12,1),
Q[11]->setService(13,0,0) setVisits(13,1),
Q[11]->setService(14,0,0) setVisits(14,1),
Q[11]->setService(15,0,0) setVisits(15,1),

model = new Linearizer(stations, classes, Q, Ncust),
cout << "OK!" << "\n",
model->solve(),

limit1 = fabs((lan_l_serv*2.0 - Q[4])>R(1))/Q[4]>R(1)),
limit2 = fabs((lan_r_serv*2.0 - Q[5])>R(6))/Q[5]>R(6)),
limit3 = fabs((ass_lan_l_original - Q[4])>R(5))/Q[4]>R(5)),
limit4 = fabs((ass_lan_r_original - Q[5])>R(10))/Q[5]>R(10)),

while ( ( limit1 > LIMIT1 ) || ( limit2 > LIMIT2 ) || ( limit3 > LIMIT3 ) || ( limit4 > LIMIT4 ) )
{
    count = CYCLE1;

    lan_l_serv = Q[4]>R(1)/2.0,
    lan_r_serv = Q[5]>R(6)/2.0,
    ass_lan_l_original = Q[4]>R(5),
    ass_lan_r_original = Q[5]>R(10),

    Q[10]->setService(11,lan_l_serv) setVisits(11,2),
    Q[10]->setService(12,lan_l_serv) setVisits(12,2),
    Q[10]->setService(13,lan_l_serv) setVisits(13,2),
    Q[10]->setService(14,lan_l_serv) setVisits(14,2),
    Q[10]->setService(15,ass_lan_l_original) setVisits(15,1),

    Q[10]->setService(1,0,0) setVisits(1,1),
    Q[10]->setService(2,0,0) setVisits(2,1),
    Q[10]->setService(3,0,0) setVisits(3,1),
    Q[10]->setService(4,0,0) setVisits(4,1),
    Q[10]->setService(5,0,0) setVisits(5,1),
    Q[10]->setService(6,0,0) setVisits(6,1),
    Q[10]->setService(7,0,0) setVisits(7,1),
    Q[10]->setService(8,0,0) setVisits(8,1),
    Q[10]->setService(9,0,0) setVisits(9,1),
    Q[10]->setService(10,0,0) setVisits(10,1),
    Q[10]->setService(16,0,0) setVisits(16,1),
    Q[10]->setService(17,0,0) setVisits(17,1),
    Q[10]->setService(18,0,0) setVisits(18,1),
    Q[10]->setService(19,0,0) setVisits(19,1),
    Q[10]->setService(20,0,0) setVisits(20,1),

    Q[11]->setService(16,lan_r_serv) setVisits(16,2),
    Q[11]->setService(17,lan_r_serv) setVisits(17,2),
    Q[11]->setService(18,lan_r_serv) setVisits(18,2),
    Q[11]->setService(19,lan_r_serv) setVisits(19,2),
    Q[11]->setService(20,ass_lan_r_original) setVisits(20,1),

    Q[11]->setService(1,0,0) setVisits(1,1),
    Q[11]->setService(2,0,0) setVisits(2,1),
    Q[11]->setService(3,0,0) setVisits(3,1),
    Q[11]->setService(4,0,0) setVisits(4,1),
    Q[11]->setService(5,0,0) setVisits(5,1),
    Q[11]->setService(6,0,0) setVisits(6,1),
    Q[11]->setService(7,0,0) setVisits(7,1),
    Q[11]->setService(8,0,0) setVisits(8,1),
    Q[11]->setService(9,0,0) setVisits(9,1),
    Q[11]->setService(10,0,0) setVisits(10,1),
    Q[11]->setService(11,0,0) setVisits(11,1),
    Q[11]->setService(12,0,0) setVisits(12,1),
    Q[11]->setService(13,0,0) setVisits(13,1),
    Q[11]->setService(14,0,0) setVisits(14,1),
    Q[11]->setService(15,0,0) setVisits(15,1),

    cout << "OK!" << "\n",
    model = new Linearizer(stations, classes, Q, Ncust),
    cout << "OK!" << "\n",
    model->solve(),

    limit1 = fabs((lan_l_serv*2.0 - Q[4])>R(1))/Q[4]>R(1)),
    limit2 = fabs((lan_r_serv*2.0 - Q[5])>R(6))/Q[5]>R(6)),
    limit3 = fabs((ass_lan_l_original - Q[4])>R(5))/Q[4]>R(5)),
    limit4 = fabs((ass_lan_r_original - Q[5])>R(10))/Q[5]>R(10)),

    count =
    cout << "count " << count << "\n"

}

cout << "in give output file name ",
fflush(stdin),
cin << OUTPUT FILE

ofstream outFile, // output file ID
outFile.open(OUTPUT FILE, ios::out);
if(!outFile) {
    cerr << "can not open output file\n",
    exit(1),
}

```

[illegible]

```

uti = Q[4]-utilization();
outFile << "utilization of local lan server is " << uti;
uti = Q[5]-utilization();
outFile << "utilization of remote lan server is " << uti;
uti = Q[8]-utilization();
outFile << "utilization of database (remote form) is " << uti;
uti = Q[9]-utilization();
outFile << "utilization of database (BSC) is " << uti;

outFile close();

delete model;
return 0;
}

```

...../

```
double tr_delay(float a, int b)
```

```

{
    double R, W, proctime;

    R = SPEED*TR_L + b/TR_RATE;           // latency
    proctime = (TOKEN + DATAL)/TR_RATE;    // proce time
    W = (R*(1+a/b) + a*proctime)/
        (2.0*(1 - a - a*R/b/proctime));    // waiting time

    return (W*proctime);
}

```

...../

1995/07/03

4_level_model.c

This code is implemented for 4_level for non gui user
input from standard input
output to specified file

add 1 assistant server
add traffic at token ring
add random traffic at back bone (FDDI)
add random traffic at wan
add random utilization at mainframe
add local user (add 4 class)
add error check for total user <= NUM_WS

total number of station is 11

```
1 ws (INF1)
2 local token ring delay (INF1)
3 remote token ring delay (INF1)
4 local lan server1 (FCFS)
5 remote lan server2 (FCFS)
6 local global delay1 (INF1)          only use 1 global delay ?
7 remote global delay2 (INF1)
8 r-db disk (FCFS)
9 b-db disk (FCFS)
10 u lan server1 (surrogate delay)    local
11 u lan server2 (surrogate delay)    remote
12 intermidate server
```

total number of class is 20

```
1 ws -> lan delay -> local server1 -> global delay1 -> r-db light (11)
2 ws -> lan delay -> local server1 -> global delay1 -> r-db heavy (12)
3 ws -> lan delay -> local server1 -> global delay1 -> b-db light (13)
4 ws -> lan delay -> local server1 -> global delay1 -> b-db heavy (14)
5 ws -> lan delay -> local server1 (15)

6 ws -> lan delay -> local server2 -> global delay2 -> r-db light (21)
7 ws -> lan delay -> local server2 -> global delay2 -> r-db heavy (22)
8 ws -> lan delay -> local server2 -> global delay2 -> b-db light (23)
9 ws -> lan delay -> local server2 -> global delay2 -> b-db heavy (24)
10 ws -> lan delay -> local server2 (25)

11 ws -> lan delay -> u local server1 -> global delay1 -> r-db light (31)
12 ws -> lan delay -> u local server1 -> global delay1 -> r-db heavy (32)
13 ws -> lan delay -> u local server1 -> global delay1 -> b-db light (33)
14 ws -> lan delay -> u local server1 -> global delay1 -> b-db heavy (34)
15 ws -> lan delay -> u local server1 (35)

16 ws -> lan delay -> u local server2 -> global delay2 -> r-db light (41)
17 ws -> lan delay -> u local server2 -> global delay2 -> r-db heavy (42)
18 ws -> lan delay -> u local server2 -> global delay2 -> b-db light (43)
19 ws -> lan delay -> u local server2 -> global delay2 -> b-db heavy (44)
20 ws -> lan delay -> u local server2 (45)
```

It is based on "linearizer C" by Greg Franks

wing Shen

```
...../

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <iostream.h>
#include <fstream.h>
#include "mva.h"
#include "server.h"
#include "population.h"
#include "vector.h"

#define LIMIT 0.0001
#define CYCLE 4

#define TOKEN 24.0 // # of bit per token
#define DATA 1000.0 // # of bit per data
#define TR 1.2 // # length of token ring (km)
#define SPEED 0.000005 // second/km
#define TR_RATE 4000000.0 // tr proc. data rate bit/second
#define FDDI_RATE 100000000.0 // FDDI proc. data rate bit/second
#define TTR 0.01 // FDDI Target Token Rotation Time

double tr_delay(float, int); // t traffic factor int. # of ws

main()
```

```
unsigned classes = 20;
unsigned stations = 12;
PopVector NCust(0);
Server** Q;
MVA * model;
char OUTPUT_FILE[20];
```

```
int NUM_WS; // # ws per tr
int num_l_tr, num_r_tr; // user input

int num_c_11, num_c_12, num_c_13, num_c_14, num_c_15, // user input
    num_c_21, num_c_22, num_c_23, num_c_24, num_c_25,
int num_c_31, num_c_32, num_c_33, num_c_34, num_c_35, // auto
    num_c_41, num_c_42, num_c_43, num_c_44, num_c_45,

double think = 15.0; // exp data
double l_lan_delay; // network
double r_lan_delay; // network
double g_delay1; // network
double g_delay2; // network
double lan_serv = 0.01; // exp data
double lan_r_serv = 0.01; // surou
double lan_l_serv = 0.01; // surou
double rdb_l; // r-db
double rdb_h; // r-db
double bdb_l; // b-db
double bdb_h; // b-db
double int_l; // int server
double int_h;

float lan_traffic_l; // tr traffic (local)
float lan_traffic_r; // tr traffic (remote)
float fddi_random_traffic; // random traffic (FDDI)
float wan_random_traffic; // random traffic (WAN)
float random_u_bdb; // random user b-db main
float fraction; // to intermidate server

float limit1, limit2; // for sur
int count = 0; // for sur

double host_time; // print out

NCust grow(classes+1);
Q = new Server * (stations+1);

cout << "In Number of client in one token ring " << endl;
fflush(stdin);
cin >> NUM_WS;

cout << "In Number of token ring at local area " << endl;
fflush(stdin);
cin >> num_l_tr;

cout << "In Traffic factor of token ring at local area " << endl;
fflush(stdin);
cin >> lan_traffic_l;

cout << "In Number of token ring at remote area " << endl;
fflush(stdin);
cin >> num_r_tr;

cout << "In Traffic factor of token ring at remote area " << endl;
fflush(stdin);
cin >> lan_traffic_r;

cout << "In Number of client go to r-db with light SQL at local tr " << endl;
fflush(stdin);
cin >> num_c_11;

cout << "In Number of client go to r-db with heavy SQL at local tr " << endl;
fflush(stdin);
cin >> num_c_12;

cout << "In Number of client go to b-db with light SQL at local tr " << endl;
fflush(stdin);
cin >> num_c_13;

cout << "In Number of client go to b-db with heavy SQL at local tr " << endl;
fflush(stdin);
cin >> num_c_14;

cout << "In Number of client need local server at local tr " << endl;
fflush(stdin);
cin >> num_c_15;

cout << "In Number of client go to r-db with light SQL at remote tr " << endl;
fflush(stdin);
cin >> num_c_21;

cout << "In Number of client go to r-db with heavy SQL at remote tr " << endl;
```

```

fflush(stdin),
cin >> num_c_22,

cout << "\n Number of client go to b db with light SQL at remote tr ",
fflush(stdin),
cin >> num_c_23,

cout << "\n Number of client go to b db with hevy SQL at remote tr ",
fflush(stdin),
cin >> num_c_24,

cout << "\n Number of client need local server at remote tr ",
fflush(stdin),
cin >> num_c_25,

cout << "\n service time of light SQL at a db ",
fflush(stdin),
cin >> tdb_l,

cout << "\n service time of hevy SQL at a db ",
fflush(stdin),
cin >> tdb_h,

cout << "\n service time of light SQL at b db ",
fflush(stdin),
cin >> bdb_l,

cout << "\n service time of hevy SQL at b db ",
fflush(stdin),
cin >> bdb_h,

cout << "\n Random traffic factor at LAN ",
fflush(stdin),
cin >> wan_random_traffic,

cout << "\n Random traffic factor at back bone ",
fflush(stdin),
cin >> fddi_random_traffic,

cout << "\n Random utils factor at b-db ",
fflush(stdin),
cin >> random_u_bdb,

cout << "\n service time of intermidate server for light SQL ",
fflush(stdin),
cin >> int_l,

cout << "\n service time of intermidate server for hevy SQL ",
fflush(stdin),
cin >> int_h,

cout << "\n fraction (0 1) of the total jobs go to intermidate server ",
fflush(stdin),
cin >> fraction,

l_lan_delay = tr_delay(lan_traffic_l, NUM_WS),
g_delay1 = (TTRT + (DATA_L + TOKEN + 5)/(FDDI_RATE))/(1-fddi_random_traffic),
r_lan_delay = tr_delay(lan_traffic_r, NUM_WS),
g_delay2 = 0.1/(1-wan_random_traffic),

if ((num_c_11 + num_c_12 + num_c_13 + num_c_14 + num_c_15) > NUM_WS) {
    cout << "\n Total workstation of per token ring is " << NUM_WS << "\n",
    exit(1),
}

if ((num_c_21 + num_c_22 + num_c_23 + num_c_24 + num_c_25) > NUM_WS) {
    cout << "\n Total workstation of per token ring is " << NUM_WS << "\n",
    exit(1),
}

num_c_31 = num_c_11 * (num_l_tr - 1),
num_c_32 = num_c_12 * (num_l_tr - 1),
num_c_33 = num_c_13 * (num_l_tr - 1),
num_c_34 = num_c_14 * (num_l_tr - 1),
num_c_35 = num_c_15 * (num_l_tr - 1),

num_c_41 = num_c_21 * (num_t_tr - 1),
num_c_42 = num_c_22 * (num_t_tr - 1),
num_c_43 = num_c_23 * (num_t_tr - 1),
num_c_44 = num_c_24 * (num_t_tr - 1),
num_c_45 = num_c_25 * (num_t_tr - 1),

NCust[1] = num_c_11,
NCust[2] = num_c_12,
NCust[3] = num_c_13,
NCust[4] = num_c_14,
NCust[5] = num_c_15,

NCust[6] = num_c_21,
NCust[7] = num_c_22,
NCust[8] = num_c_23,
NCust[9] = num_c_24,

```

```

NCust[10] = num_c_25,

NCust[11] = num_c_31,
NCust[12] = num_c_32,
NCust[13] = num_c_33,
NCust[14] = num_c_34,
NCust[15] = num_c_35,

NCust[16] = num_c_41,
NCust[17] = num_c_42,
NCust[18] = num_c_43,
NCust[19] = num_c_44,
NCust[20] = num_c_45,

bdb_l = bdb_l/(1-random_u_bdb),
bdb_h = bdb_h/(1-random_u_bdb),

/**
total number of station is 11

1  ws (INFI)
2  local token ring delay (INFI)
3  remote token ring delay (INFI)
4  lan server1 (FCFS) local
5  lan server2 (FCFS) remote
6  global delay1 (INFI) local
7  global delay2 (INFI) remote
8  r-db disk (FCFS)
9  b-db disk (FCFS)
10 u lan server1 (surrogate delay) local
11 u lan server2 (surrogate delay) remote
**/

Q[1] = new Delay_Server(classes), // think
Q[2] = new Delay_Server(classes), // l_lan_delay
Q[3] = new Delay_Server(classes), // r_lan_delay
Q[4] = new FCFS_Server(classes), // lan_serv (local)
Q[5] = new FCFS_Server(classes), // lan_serv (remote)
Q[6] = new Delay_Server(classes), // g_delay1 (local)
Q[7] = new Delay_Server(classes), // g_delay2 (remote)
Q[8] = new FCFS_Server(classes), // disk_ac (r-db)
Q[9] = new FCFS_Server(classes), // disk_ac (b-db)
Q[10] = new Delay_Server(classes), // lan_l_serv
Q[11] = new Delay_Server(classes), // lan_r_serv
Q[12] = new FCFS_Server(classes), // int_server

Q[1] -> setService(1,think) setVisits(1,1), // work station
Q[1] -> setService(2,think) setVisits(2,1),
Q[1] -> setService(3,think) setVisits(3,1),
Q[1] -> setService(4,think) setVisits(4,1),
Q[1] -> setService(5,think) setVisits(5,1),
Q[1] -> setService(6,think) setVisits(6,1),
Q[1] -> setService(7,think) setVisits(7,1),
Q[1] -> setService(8,think) setVisits(8,1),
Q[1] -> setService(9,think) setVisits(9,1),
Q[1] -> setService(10,think) setVisits(10,1),
Q[1] -> setService(11,think) setVisits(11,1),
Q[1] -> setService(12,think) setVisits(12,1),
Q[1] -> setService(13,think) setVisits(13,1),
Q[1] -> setService(14,think) setVisits(14,1),
Q[1] -> setService(15,think) setVisits(15,1),
Q[1] -> setService(16,think) setVisits(16,1),
Q[1] -> setService(17,think) setVisits(17,1),
Q[1] -> setService(18,think) setVisits(18,1),
Q[1] -> setService(19,think) setVisits(19,1),
Q[1] -> setService(20,think) setVisits(20,1),

Q[2] -> setService(1,l_lan_delay) setVisits(1,2), // local tr delay
Q[2] -> setService(2,l_lan_delay) setVisits(2,2),
Q[2] -> setService(3,l_lan_delay) setVisits(3,2),
Q[2] -> setService(4,l_lan_delay) setVisits(4,2),
Q[2] -> setService(5,l_lan_delay) setVisits(5,2),
Q[2] -> setService(6,0 0) setVisits(6,1),
Q[2] -> setService(7,0 0) setVisits(7,1),
Q[2] -> setService(8,0 0) setVisits(8,1),
Q[2] -> setService(9,0 0) setVisits(9,1),
Q[2] -> setService(10,0 0) setVisits(10,1),
Q[2] -> setService(11,l_lan_delay) setVisits(11,2),
Q[2] -> setService(12,l_lan_delay) setVisits(12,2),
Q[2] -> setService(13,l_lan_delay) setVisits(13,2),
Q[2] -> setService(14,l_lan_delay) setVisits(14,2),
Q[2] -> setService(15,l_lan_delay) setVisits(15,2),
Q[2] -> setService(16,0 0) setVisits(16,1),
Q[2] -> setService(17,0 0) setVisits(17,1),
Q[2] -> setService(18,0 0) setVisits(18,1),
Q[2] -> setService(19,0 0) setVisits(19,1),
Q[2] -> setService(20,0 0) setVisits(20,1),

Q[3] -> setService(1,0 0) setVisits(1,1), // remote tr delay
Q[3] -> setService(2,0 0) setVisits(2,1),
Q[3] -> setService(3,0 0) setVisits(3,1),
Q[3] -> setService(4,0 0) setVisits(4,1),

```

```

Q[3]->setService(5,0,0) setVisits(5,1);
Q[3]->setService(6,r_lan_delay) setVisits(6,2);
Q[3]->setService(7,r_lan_delay) setVisits(7,2);
Q[3]->setService(8,r_lan_delay) setVisits(8,2);
Q[3]->setService(9,r_lan_delay) setVisits(9,2);
Q[3]->setService(10,r_lan_delay) setVisits(10,2);
Q[3]->setService(11,0,0) setVisits(11,1);
Q[3]->setService(12,0,0) setVisits(12,1);
Q[3]->setService(13,0,0) setVisits(13,1);
Q[3]->setService(14,0,0) setVisits(14,1);
Q[3]->setService(15,0,0) setVisits(15,1);
Q[3]->setService(16,r_lan_delay) setVisits(16,2);
Q[3]->setService(17,r_lan_delay) setVisits(17,2);
Q[3]->setService(18,r_lan_delay) setVisits(18,2);
Q[3]->setService(19,r_lan_delay) setVisits(19,2);
Q[3]->setService(20,r_lan_delay) setVisits(20,2);

```

```

Q[4]->setService(1,lan_serv) setVisits(1,2); // lan server of local tr
Q[4]->setService(2,lan_serv) setVisits(2,2);
Q[4]->setService(3,lan_serv) setVisits(3,2);
Q[4]->setService(4,lan_serv) setVisits(4,2);
Q[4]->setService(5,lan_serv) setVisits(5,1);

```

```

Q[4]->setService(6,0,0) setVisits(6,1);
Q[4]->setService(7,0,0) setVisits(7,1);
Q[4]->setService(8,0,0) setVisits(8,1);
Q[4]->setService(9,0,0) setVisits(9,1);
Q[4]->setService(10,0,0) setVisits(10,1);
Q[4]->setService(11,0,0) setVisits(11,1);
Q[4]->setService(12,0,0) setVisits(12,1);
Q[4]->setService(13,0,0) setVisits(13,1);
Q[4]->setService(14,0,0) setVisits(14,1);
Q[4]->setService(15,0,0) setVisits(15,1);
Q[4]->setService(16,0,0) setVisits(16,1);
Q[4]->setService(17,0,0) setVisits(17,1);
Q[4]->setService(18,0,0) setVisits(18,1);
Q[4]->setService(19,0,0) setVisits(19,1);
Q[4]->setService(20,0,0) setVisits(20,1);

```

```

Q[5]->setService(6,lan_serv) setVisits(6,2); // lan server of remote tr
Q[5]->setService(7,lan_serv) setVisits(7,2);
Q[5]->setService(8,lan_serv) setVisits(8,2);
Q[5]->setService(9,lan_serv) setVisits(9,2);
Q[5]->setService(10,lan_serv) setVisits(10,1);

```

```

Q[5]->setService(1,0,0) setVisits(1,1);
Q[5]->setService(2,0,0) setVisits(2,1);
Q[5]->setService(3,0,0) setVisits(3,1);
Q[5]->setService(4,0,0) setVisits(4,1);
Q[5]->setService(5,0,0) setVisits(5,1);
Q[5]->setService(11,0,0) setVisits(11,1);
Q[5]->setService(12,0,0) setVisits(12,1);
Q[5]->setService(13,0,0) setVisits(13,1);
Q[5]->setService(14,0,0) setVisits(14,1);
Q[5]->setService(15,0,0) setVisits(15,1);
Q[5]->setService(16,0,0) setVisits(16,1);
Q[5]->setService(17,0,0) setVisits(17,1);
Q[5]->setService(18,0,0) setVisits(18,1);
Q[5]->setService(19,0,0) setVisits(19,1);
Q[5]->setService(20,0,0) setVisits(20,1);

```

```

Q[6]->setService(1,g_delay1) setVisits(1,2); // FDDI delay (global 1)
Q[6]->setService(2,g_delay1) setVisits(2,2);
Q[6]->setService(3,g_delay1) setVisits(3,2);
Q[6]->setService(4,g_delay1) setVisits(4,2);
Q[6]->setService(11,g_delay1) setVisits(11,2);
Q[6]->setService(12,g_delay1) setVisits(12,2);
Q[6]->setService(13,g_delay1) setVisits(13,2);
Q[6]->setService(14,g_delay1) setVisits(14,2);

```

```

Q[6]->setService(5,0,0) setVisits(5,1);
Q[6]->setService(6,0,0) setVisits(6,1);
Q[6]->setService(7,0,0) setVisits(7,1);
Q[6]->setService(8,0,0) setVisits(8,1);
Q[6]->setService(9,0,0) setVisits(9,1);
Q[6]->setService(10,0,0) setVisits(10,1);
Q[6]->setService(15,0,0) setVisits(15,1);
Q[6]->setService(16,0,0) setVisits(16,1);
Q[6]->setService(17,0,0) setVisits(17,1);
Q[6]->setService(18,0,0) setVisits(18,1);
Q[6]->setService(19,0,0) setVisits(19,1);
Q[6]->setService(20,0,0) setVisits(20,1);

```

```

Q[7]->setService(6,g_delay2) setVisits(6,2); // wan delay (global 2)
Q[7]->setService(7,g_delay2) setVisits(7,2);
Q[7]->setService(8,g_delay2) setVisits(8,2);
Q[7]->setService(9,g_delay2) setVisits(9,2);
Q[7]->setService(16,g_delay2) setVisits(16,2);
Q[7]->setService(17,g_delay2) setVisits(17,2);
Q[7]->setService(18,g_delay2) setVisits(18,2);

```

```

Q[7]->setService(19,g_delay2) setVisits(19,2);

```

```

Q[7]->setService(1,0,0) setVisits(1,1);
Q[7]->setService(2,0,0) setVisits(2,1);
Q[7]->setService(3,0,0) setVisits(3,1);
Q[7]->setService(4,0,0) setVisits(4,1);
Q[7]->setService(5,0,0) setVisits(5,1);
Q[7]->setService(10,0,0) setVisits(10,1);
Q[7]->setService(11,0,0) setVisits(11,1);
Q[7]->setService(12,0,0) setVisits(12,1);
Q[7]->setService(13,0,0) setVisits(13,1);
Q[7]->setService(14,0,0) setVisits(14,1);
Q[7]->setService(15,0,0) setVisits(15,1);
Q[7]->setService(20,0,0) setVisits(20,1);

```

```

Q[8]->setService(1,ldb_1) setVisits(1,1-fraction); // remote form db
Q[8]->setService(2,ldb_h) setVisits(2,1-fraction);
Q[8]->setService(6,ldb_1) setVisits(6,1-fraction);
Q[8]->setService(7,ldb_h) setVisits(7,1-fraction);
Q[8]->setService(11,ldb_1) setVisits(11,1-fraction);
Q[8]->setService(12,ldb_h) setVisits(12,1-fraction);
Q[8]->setService(16,ldb_1) setVisits(16,1-fraction);
Q[8]->setService(17,ldb_h) setVisits(17,1-fraction);

```

```

Q[8]->setService(3,0,0) setVisits(3,1);
Q[8]->setService(4,0,0) setVisits(4,1);
Q[8]->setService(5,0,0) setVisits(5,1);
Q[8]->setService(8,0,0) setVisits(8,1);
Q[8]->setService(9,0,0) setVisits(9,1);
Q[8]->setService(10,0,0) setVisits(10,1);
Q[8]->setService(13,0,0) setVisits(13,1);
Q[8]->setService(14,0,0) setVisits(14,1);
Q[8]->setService(15,0,0) setVisits(15,1);
Q[8]->setService(18,0,0) setVisits(18,1);
Q[8]->setService(19,0,0) setVisits(19,1);
Q[8]->setService(20,0,0) setVisits(20,1);

```

```

Q[9]->setService(3,bdb_1) setVisits(3,1-fraction); // B & C db
Q[9]->setService(4,bdb_h) setVisits(4,1-fraction);
Q[9]->setService(8,bdb_1) setVisits(8,1-fraction);
Q[9]->setService(9,bdb_h) setVisits(9,1-fraction);
Q[9]->setService(13,bdb_1) setVisits(13,1-fraction);
Q[9]->setService(14,bdb_h) setVisits(14,1-fraction);
Q[9]->setService(18,bdb_1) setVisits(18,1-fraction);
Q[9]->setService(19,bdb_h) setVisits(19,1-fraction);

```

```

Q[9]->setService(1,0,0) setVisits(1,1);
Q[9]->setService(2,0,0) setVisits(2,1);
Q[9]->setService(5,0,0) setVisits(5,1);
Q[9]->setService(6,0,0) setVisits(6,1);
Q[9]->setService(7,0,0) setVisits(7,1);
Q[9]->setService(10,0,0) setVisits(10,1);
Q[9]->setService(11,0,0) setVisits(11,1);
Q[9]->setService(12,0,0) setVisits(12,1);
Q[9]->setService(15,0,0) setVisits(15,1);
Q[9]->setService(16,0,0) setVisits(16,1);
Q[9]->setService(17,0,0) setVisits(17,1);
Q[9]->setService(20,0,0) setVisits(20,1);

```

```

Q[10]->setService(11,lan_l_serv) setVisits(11,2); // sur lan serv of local tr
Q[10]->setService(12,lan_l_serv) setVisits(12,2);
Q[10]->setService(13,lan_l_serv) setVisits(13,2);
Q[10]->setService(14,lan_l_serv) setVisits(14,2);
Q[10]->setService(15,lan_l_serv) setVisits(15,1);

```

```

Q[10]->setService(1,0,0) setVisits(1,1);
Q[10]->setService(2,0,0) setVisits(2,1);
Q[10]->setService(3,0,0) setVisits(3,1);
Q[10]->setService(4,0,0) setVisits(4,1);
Q[10]->setService(5,0,0) setVisits(5,1);
Q[10]->setService(6,0,0) setVisits(6,1);
Q[10]->setService(7,0,0) setVisits(7,1);
Q[10]->setService(8,0,0) setVisits(8,1);
Q[10]->setService(9,0,0) setVisits(9,1);
Q[10]->setService(10,0,0) setVisits(10,1);
Q[10]->setService(16,0,0) setVisits(16,1);
Q[10]->setService(17,0,0) setVisits(17,1);
Q[10]->setService(18,0,0) setVisits(18,1);
Q[10]->setService(19,0,0) setVisits(19,1);
Q[10]->setService(20,0,0) setVisits(20,1);

```

```

Q[11]->setService(16,lan_r_serv) setVisits(16,2); // sur lan serv of remote tr
Q[11]->setService(17,lan_r_serv) setVisits(17,2);
Q[11]->setService(18,lan_r_serv) setVisits(18,2);
Q[11]->setService(19,lan_r_serv) setVisits(19,2);
Q[11]->setService(20,lan_r_serv) setVisits(20,1);

```

```

Q[11]->setService(1,0,0) setVisits(1,1);
Q[11]->setService(2,0,0) setVisits(2,1);

```

```
Q[11]->setService(3,0,0) setVisits(3,1),
Q[11]->setService(4,0,0) setVisits(4,1),
Q[11]->setService(5,0,0) setVisits(5,1),
Q[11]->setService(6,0,0) setVisits(6,1),
Q[11]->setService(7,0,0) setVisits(7,1),
Q[11]->setService(8,0,0) setVisits(8,1),
Q[11]->setService(9,0,0) setVisits(9,1),
Q[11]->setService(10,0,0) setVisits(10,1),
Q[11]->setService(11,0,0) setVisits(11,1),
Q[11]->setService(12,0,0) setVisits(12,1),
Q[11]->setService(13,0,0) setVisits(13,1),
Q[11]->setService(14,0,0) setVisits(14,1),
Q[11]->setService(15,0,0) setVisits(15,1),
```

```
Q[12] -> setService(1, int_l) setVisits(1, fraction), // int server
Q[12] -> setService(2, int_h) setVisits(2, fraction),
Q[12] -> setService(3, int_l) setVisits(3, fraction),
Q[12] -> setService(4, int_h) setVisits(4, fraction),
Q[12] -> setService(5, 0) setVisits(5, 1),
Q[12] -> setService(6, int_h) setVisits(6, fraction),
Q[12] -> setService(7, int_l) setVisits(7, fraction),
Q[12] -> setService(8, int_h) setVisits(8, fraction),
Q[12] -> setService(9, int_l) setVisits(9, fraction),
Q[12] -> setService(10, 0) setVisits(10, 1),
Q[12] -> setService(11, int_l) setVisits(11, fraction),
Q[12] -> setService(12, int_h) setVisits(12, fraction),
Q[12] -> setService(13, int_l) setVisits(13, fraction),
Q[12] -> setService(14, int_h) setVisits(14, fraction),
Q[12] -> setService(15, 0) setVisits(15, 1),
Q[12] -> setService(16, int_h) setVisits(16, fraction),
Q[12] -> setService(17, int_l) setVisits(17, fraction),
Q[12] -> setService(18, int_h) setVisits(18, fraction),
Q[12] -> setService(19, int_l) setVisits(19, fraction),
Q[12] -> setService(20, 0) setVisits(20, 1),
```

```
model = new Linearizer(stations, classes, Q, NCust),
model.solve(),
```

```
limit1 = fabs((lan_l_serv*2 0 - Q[4]->R[1])/Q[4]->R[1]),
limit2 = fabs((lan_r_serv*2 0 - Q[5]->R[6])/Q[5]->R[6]),
```

```
while (((limit1 > LIMIT) || (limit2 > LIMIT)) && (count < CYCLE)) {
```

```
lan_1_serv: Q[4] -> R[1]/2 0.
lan_1_serv: Q[5] -> R[6]/2 0.
```

```
Q(10) >getService(11,lan_1_serv) getVisits(11,2).
Q(10) >getService(12,lan_1_serv) getVisits(12,2).
Q(10) >getService(13,lan_1_serv) getVisits(13,2).
Q(10) >getService(14,lan_1_serv) getVisits(14,2).
Q(10) >getService(15,lan_1_serv) getVisits(15,1).
```

```

Q[10] ->setService(1,0,0) setVisits(1,1)
Q[10] ->setService(2,0,0) setVisits(2,1)
Q[10] ->setService(3,0,0) setVisits(3,1)
Q[10] ->setService(4,0,0) setVisits(4,1)
Q[10] ->setService(5,0,0) setVisits(5,1)
Q[10] ->setService(6,0,0) setVisits(6,1)
Q[10] ->setService(7,0,0) setVisits(7,1)
Q[10] ->setService(8,0,0) setVisits(8,1)
Q[10] ->setService(9,0,0) setVisits(9,1)
Q[10] ->setService(10,0,0) setVisits(10,1)
Q[10] ->setService(11,0,0) setVisits(11,1)
Q[10] ->setService(16,0,0) setVisits(16,1)
Q[10] ->setService(17,0,0) setVisits(17,1)
Q[10] ->setService(18,0,0) setVisits(18,1)
Q[10] ->setService(19,0,0) setVisits(19,1)
Q[10] ->setService(20,0,0) setVisits(20,1)

```

```
Q[11]->getService(16,lan_r_serv) getVisits(16,2),
Q[11]->getService(17,lan_r_serv) getVisits(17,2),
Q[11]->getService(18,lan_r_serv) getVisits(18,2),
Q[11]->getService(19,lan_r_serv) getVisits(19,2),
Q[11]->getService(20,lan_r_serv) getVisits(20,2),
```

```
Q[11] ->setService(1,0,0) setVisit(1,1,1)
Q[11] ->setService(2,0,0) setVisit(2,1,1)
Q[11] ->setService(3,0,0) setVisit(3,1,1)
Q[11] ->setService(4,0,0) setVisit(4,1,1)
Q[11] ->setService(5,0,0) setVisit(5,1,1)
Q[11] ->setService(6,0,0) setVisit(6,1,1)
Q[11] ->setService(7,0,0) setVisit(7,1,1)
Q[11] ->setService(8,0,0) setVisit(8,1,1)
Q[11] ->setService(9,0,0) setVisit(9,1,1)
Q[11] ->setService(10,0,0) setVisit(10,1,1)
Q[11] ->setService(11,0,0) setVisit(11,1,1)
Q[11] ->setService(12,0,0) setVisit(12,1,1)
Q[11] ->setService(13,0,0) setVisit(13,1,1)
Q[11] ->setService(14,0,0) setVisit(14,1,1)
Q[11] ->setService(15,0,0) setVisit(15,1,1)
```

```
model = new Linearizer(stations, classes, Q, Hcost);
model.solve();
```

```
limit1 = [abs((lan_1_serv*2 0 - Q[4])>R[1])/Q[4])>R[1]],
limit2 = [abs((lan_1_serv*2 0 - Q[5])>R[6])/Q[5])>R[6]],
count++;
```

```
cout << '\n Please enter the output file name ' ,
fflush(stdin),
cin >> OUTPUT_FILE.
```

```
ofstream outFile, // output file ID
outFile open(OUTPUT_FILE, ios_out);
if(!outFile) {
    cerr << "can not open output_file in",
    exit(-1);
}
```

```
outFile << "\t\t<PARAMETERS>\n";
```

```
outFile << '\n number of workstation per lan is ' << NUM_WS;
```

```
outFile << '\n number of token ring at local area is %t' << num_r1_t;
outFile << '\n lan traffic at local area is %t' << lan_traffic_l;

outFile << '\n number of token ring at remote area is %t' << num_r1_r;
outFile << '\n lan traffic at remote area is %t' << lan_traffic_r;
```

```

outFile << '\n number of user go to r_db with light transaction\n';
outFile << ' at local area (total) is :%d\t' << num_c_11*num_l_tr;
outFile << '\n number of user go to r_db with heavy transaction\n';
outFile << ' at local area (total) is :%d\t' << num_c_12*num_l_tr;
outFile << '\n number of user go to b_db with light transaction\n';
outFile << ' at local area (total) is :%d\t' << num_c_13*num_l_tr;
outFile << '\n number of user go to b_db with heavy transaction\n';
outFile << ' at local area (total) is :%d\t' << num_c_14*num_l_tr;
outFile << '\n number of user only go to lan server\n';
outFile << ' at local area (total) is :%d\t' << num_c_15*num_l_tr;

```

```
outFile << "in number of user go to r_db with light transaction\n";
outFile << "at remote area (total) is %t" << num_c_21*num_r_tr;
outFile << "in number of user go to r_db with heavy transaction\n";
outFile << "at remote area (total) is %t" << num_c_22*num_r_tr;
outFile << "in number of user go to h_db with light transaction\n";
outFile << "at remote area (total) is %t" << num_c_23*num_r_tr;
outFile << "in number of user go to b_db with heavy transaction\n";
outFile << "at remote area (total) is %t" << num_c_24*num_r_tr;
outFile << "in number of user only go to lan server\n";
outFile << "at remote area (total) is %t" << num_c_25*num_r_tr;
```

```
outFile << '\n r_db service time for light transaction is:\t' << rdb_l << " sec",  
outFile << '\n r_db service time for heavy transaction is:\t' << rdb_h << " sec",  
outFile << '\n b_db service time for light transaction is \t' << bdb_l << "sec",  
outFile << '\n b_db service time for heavy transaction is \t' << bdb_h << "sec",  
outFile << '\n lan server service time is\t\t\t\t\t0.0 sec'
```

```
outFile << '\n random traffic of back bone (FDDI) is ' << fddi_random_traffic,
outFile << '\n random traffic of wan is ' << wan_random_traffic,
outFile << '\n random utilization of b-db cpu is ' << random_u_bdb,
```

```
outFile << '\n inter server service time for light transaction is ' << int_l;
outFile << '\n inter server service time for heavy transaction is ' << int_h;
outFile << '\n fraction of job to inter server is ' << fraction;
```

```
outFile << '\n\n\\t\\t<RESULTS>\n'
```

```
host_time = Q[2] * R[1] + Q[4] * R[1] + Q[6] * R[1] + (Q[8] * R[1]) / (1 - fraction),
outFile << '\n local ws to rdb with light transaction ' << host_time << ' good',
```

```
host time = Q(2)->R(2) + Q(4)->R(2) + Q(6)->R(2) + (Q(8)->R(2))/(1 fraction),
outfile << '\n local we to 1 db with heavy transaction \t' << host time << ' seed',
```

```

host time = Q[2]->R[3] + Q[4]->R[3] + Q[6]->R[3] + (Q[9]->R[1])/(1 fraction),
outFile << "\n local wg to b-db with light transaction it" << host time << " seed",

```

```

hott_time = Q[2] * R[4] + Q[4] * R[4] + Q[6] * R[4] + (Q[9] * R[4]) / (1 - fraction),
outFile << '\n local wg to b-db with heavy transaction: t' << hott_time << ' sec'.

```

```
host_time = Q(3)->R(6) + Q(5)->R(6) + Q(7)->R(6) + Q(8)->R(6))/4 (fraction),
outFile <- 'remote wg to x-db with light transaction V' <-> host_time <-> 'sec'.
```

```
host_time = Q[1] * R[1] + Q[5] * R[1] + Q[7] * R[1] + (Q[8] - 17) / (1 fraction),
outFile << "remote wq to rdb with heavy transaction at " << host_time << " sec".
```

```
host_time = Q(3)->R(0) + Q(5) ->R(0) + Q(7)->R(0) + (Q(9)->R(0))/(1-fraction),
outFile << "\remote wg to b db with light transaction at" << host_time << " send",
```

```
host time = Q(3)->R(9) + Q(5)->R(9) + Q(7)->R(9) + (Q(9)->R(9))/4 fraction!
outFile << "\nnote we to b db with heavy transaction 1" << host time << "\nend"
```

$$\text{host time} = Q[2] \rightarrow R[1] + Q[4] \rightarrow R[1] + Q[6] \rightarrow R[1] + (Q[12] \rightarrow R[1]) / \text{fraction}$$

```

outFile << "(n local ws to int_server with light tran  \t" << host_time << " seed",
host_time = Q[12]->R(2) + Q[4]->R(2) + Q[6]->R(2) + (Q[12]->R(2))/fraction,
outFile << "(n local ws to int_server with heavy tran  \t" << host_time << " seed",

host_time = Q[3]->R(5) + Q[5]->R(5) + Q[7]->R(5) + (Q[12]->R(6))/fraction,
outFile << "(remote ws to int_server with light tran  \t" << host_time << " seed",

host_time = Q[3]->R(6) + Q[5]->R(6) + Q[7]->R(6) + (Q[12]->R(7))/fraction,
outFile << "(remote ws to int_server with heavy tran  \t" << host_time << " seed",

double utl,

utl = Q[4]->utilization(),
outFile << "(utilization of local lan server is %f)" << utl,
utl = Q[5]->utilization(),
outFile << "(utilization of remote lan server is %f)" << utl,
utl = Q[8]->utilization(),
outFile << "(utilization of database (remote form) is %f)" << utl,
utl = Q[9]->utilization(),
outFile << "(utilization of database (B/C) is %f)" << utl,
utl = Q[12]->utilization(),
outFile << "(utilization of intermediate server is %f)" << utl,

outFile.close(),

delete model,
return 0,
}

```

...../

```
double tr_delay(float a, int b)
```

```

{
    double R, W, proctime

    R = SPEED*TP_L + b/TR_RATE,           // latency
    proctime = (TOKEN + DATA)/TR_RATE,    // proce time
    W = (R*(1+a/b) + a*proctime)/
        (2*Q[1] - a - a*R/b/proctime),    // waiting time

    return (W*proctime),
}

```

...../

2 of/de 2

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

APPENDIX B

GUI of CPT_SCSN

▽

cpt_scsn

Capacity Planning Tool
for
Symmetric Client-Server Network

README

INPUT

input by file

input by parameters

edit input_file

SOLVER

3-layer model

4-layer model

OUTPUT

output to file

show output

print output

quit

INPUT_HELP

There are two way to input
by a file or
by parameters
default: by file <== input_file

<Input by file>

browse file list and dubble click the selected file

<Input by parameters>

- (1) open input window
- (2) click open input file button
- (3) input parameter then <return>

.....

- (4) click close input file button
- (5) click OK button

NOTE

the defference between 3-layer model and 4-layer model!!

OK

parameters_input	
<input type="button" value="help"/>	<input type="button" value="quit"/>
number of work station per token ring	<input type="text"/>
number of token ring in local area	<input type="text"/>
local token ring traffic ($0 < \&\& < 1.0$)	<input type="text"/>
number of token ring in remote area	<input type="text"/>
remote token ring traffic ($0 < \&\& < 1.0$)	<input type="text"/>
per local token ring:	
number of user to remote form database with light transaction	<input type="text"/>
number of user to remote form database with heavy transaction	<input type="text"/>
number of user to B & C database with light transaction	<input type="text"/>
number of user to B & C database with heavy transaction	<input type="text"/>
number of user to local server only	<input type="text"/>
per remote token ring:	
number of user to remote form database with light transaction	<input type="text"/>
number of user to remote form database with heavy transaction	<input type="text"/>
number of user to B & C database with light transaction	<input type="text"/>
number of user to B & C database with heavy transaction	<input type="text"/>
number of user to local server only	<input type="text"/>
light transaction service time of remote form database(second)	<input type="text"/>
heavy transaction service time of remote form database(second)	<input type="text"/>
light transaction service time of B & C database(second)	<input type="text"/>
heavy transaction service time of B & C database(second)	<input type="text"/>
random traffic of WAN ($0 < \&\& < 1.0$)	<input type="text"/>
random traffic of back bone ($0 < \&\& < 1.0$)	<input type="text"/>
random utilization of mainframe for B & C database ($0 < \&\& < 1.0$)	<input type="text"/>
following 4-layer model only	
intermidate server light transaction service time (second)	<input type="text"/>
intermidate server heavy transaction service time (second)	<input type="text"/>
fraction(of users to intermidate server)	<input type="text"/>
<input type="button" value="open input file"/>	<input type="button" value="close input file"/>
<input type="button" value="save to file"/>	<input type="button" value="clear"/>
<input type="button" value="OK"/>

MODEL_HELP

There are two models in the system:

**3-layer model and
4-layer model**

**3-layer model:
wkst == lan server == mainframe**

**4-layer model:
wkst == lan server == intermidate server == mainframe**

**NOTE:
Input parameters are different for these two models.**

OK

☒ **put_to_file**

put output to file:

<PARAMETERS>

number of workstation per lan is: 30
number of token ring at local area is: 20
lan traffic at local area is: 0.5
number of token ring at remote area is: 20
lan traffic at remote area is: 0.5
number of user go to r_db with light transaction
at local area (total) is: 80
number of user go to r_db with light transaction
at remote area (total) is: 80
r_db service time for light transaction is: 0.02 secd
number of user go to r_db with heavy transaction
at local area (total) is: 80
number of user go to r_db with heavy transaction
at remote area (total) is: 80
r_db service time for heavy transaction is: 0.04 secd
random traffic for back bone (FDDI) is: 0.5
random traffic for wan is: 0.5
number of user only go to lan server
at local area (total) is: 160
number of user only go to lan server
at remote area (total) is: 160
number of user go to b_db with light transaction
at local area (total) is: 80
number of user go to b_db with light transaction
at remote area (total) is: 80
b_db service time for light transaction is: 0.06 secd
number of user go to b_db with heavy transaction
at local area (total) is: 80
number of user go to b_db with heavy transaction
at remote area (total) is: 80
b_db service time for heavy transaction is: 0.12 secd

<RESULTS>

local ws to r-db with light transaction: 0.0981204 secd
local ws to r-db with heavy transaction: 0.118041 secd
local ws to b-db with light transaction: 13.6056 secd
local ws to b-db with heavy transaction: 13.6055 secd
remote ws to r-db with light transaction: 0.477998 secd
remote ws to r-db with heavy transaction: 0.458077 secd
remote ws to b-db with light transaction: 13.9669 secd
remote ws to b-db with heavy transaction: 13.9663 secd
total throughput (go to r-db) is: 11.0988 /secd
total throughput (go to b-db) is: 11.1102 /secd
utilization of local lan server is: 0.019568
utilization of remote lan server is: 0.019429
utilization of database (remote form) is: 0.332926
utilization of database (B&C) is: 0.999917

OK

APPENDIX C

The source code of GUI

TCL-TK CODE

CODE1: cpt_scon

```
#!/usr/local/bin/wish -f

# set .frame1      put title
# .frame1.title    title
# set .frame2      put menu button
# .frame2.help     help menu button

frame .frame1 -relief flat
frame .frame2 -borderwidth 1
pack .frame1 .frame2 -side top -fill y -anchor center

label .frame1.title1 -text "Capacity Planning Tool" -pady 1
pack .frame1.title1 -side top -anchor center
label .frame1.title2 -text "for" -pady 1
pack .frame1.title2 -side top -anchor center
label .frame1.title3 -text "Symmetric Client-Server Network" -pady 1
pack .frame1.title3 -side top -anchor center
label .frame1.title4 -text "\n\n" -pady 1
pack .frame1.title4 -side top -anchor center

menubutton .frame2.help -text "README" -menu .frame2.help.m
menu .frame2.help.m
.frame2.help.m add command \
    -label "usage" \
    -command "exec usage" \
    -underline 0
.frame2.help.m add command \
    -label "input" \
    -command "exec input_help" \
    -underline 0
.frame2.help.m add command \
    -label "solver" \
    -command "exec model_help" \
    -underline 0
.frame2.help.m add command \
    -label "output" \
    -command "exec output_help" \
    -underline 0
pack .frame2.help -side left
```

```
# set .frame3      put 3 frame which are for button
# .frame3.input     put input button
# .frame3.model     put model button
# .frame3.output    put output button
#
# .frame3.input.1   label
# .frame3.input.f   input by file
# .frame3.input.p   input by parameters
# .frame3.input.ll  take space
#
# .frame3.model.1   label
# .frame3.model.m1  exec model1
# .frame3.model.m2  exec model2
# .frame3.model.ll  take space
#
# .frame3.output.1  label
# .frame3.output.f  output to file
# .frame3.output.s  show output
# .frame3.output.p  print output
```

```
frame .frame3 -relief flat
pack .frame3 -side top -fill y -anchor center

frame .frame3.input -borderwidth 24
frame .frame3.model -borderwidth 16
frame .frame3.output -borderwidth 24
pack .frame3.input .frame3.model .frame3.output -side left
```

```
label .frame3.input.1 -text "INPUT" -width 10 -pady 1
button .frame3.input.f \
    -text "input by file" \
    -width 10 \
    -anchor center \
    -command "exec browse_input"

button .frame3.input.p \
    -text "input by parameters" \
    -width 10 \
    -anchor center \
    -command "exec parameters_input"

button .frame3.input.e \
    -text "edit input_file" \
    -width 10 \
    -anchor center \
    -command "exec edit_input"

label .frame3.input.ll -width 10
pack .frame3.input.1 \
    .frame3.input.f \
```

```
.frame3.input.p \
.frame3.input.e \
.frame3.input.ll \
    -side top
```

```
label .frame3.model.1 -text "SOLVER" -width 10 -pady 1
button .frame3.model.m1 \
    -text "3-layer model" \
    -width 10 \
    -anchor center \
    -command "exec 3_level_model"

button .frame3.model.m2 \
    -text "4-layer model" \
    -width 10 \
    -anchor center \
    -command "exec 4_level_model"

label .frame3.model.ll -width 10
label .frame3.model.12 -width 10
pack .frame3.model.1 \
    .frame3.model.m1 \
    .frame3.model.m2 \
    .frame3.model.ll \
    .frame3.model.12 \
    -side top
```

```
label .frame3.output.1 -text "OUTPUT" -width 10 -pady 1
button .frame3.output.f \
    -text "output to file" \
    -width 10 \
    -anchor center \
    -command "exec put_to_file"

button .frame3.output.s \
    -text "show output" \
    -width 10 \
    -anchor center \
    -command "exec show_output"

button .frame3.output.p \
    -text "print output" \
    -width 10 \
    -anchor center \
    -command "exec lpr -Pfamily output_file"

label .frame3.output.ll -width 10
```

```
pack .frame3.output.1 \
    .frame3.output.f \
    .frame3.output.s \
    .frame3.output.p \
    .frame3.output.ll \
    -side top
```

```
button .quit -text "quit" -command "exit"
pack .quit -side top
```

CODE 2: edit_file

```
#!/usr/local/bin/wish -f
#
# This script generates a directory browser, which lists the working
# directory and allows you to copy file to input_file by double-clicking
#
# Create a scrollbar on the right side of the main window and a listbox
# on the left side
```

```
# frame      put .frame.scroll and frame list
# quit      put quit button
```

```
# frame .frame -relief flat
# pack .frame -side top -fill y -anchor center
```

```
scrollbar .scroll -command ".list yview"
pack .scroll -side right -fill y
listbox .list -yscroll "$scroll set" -relief raised -geometry 20x20 \
    -setgrid yes
pack .list -side left -fill both -expand yes
wm minsize 1 1
```

```
button .quit -text "quit" -command "exit"
pack .quit -side bottom
```

```
# The procedure below is invoked to open a browser on a given file. If the
# file is a directory then another instance of this program is invoked, if
# the file is a regular file then the Mx editor is invoked to display
# the file
```

```
proc browse (dir file) {
    global env
    if ([string compare $dir "."] != 0) {set file $dir/$file}
    if ([file isdirectory $file]) {
        exec browse $file &
    } else {
        if ([file isfile $file]) {
            if [info exists env(EDITOR)] {
```



```

frame .frame -relief flat
pack .frame -side top -fill y -anchor center

frame .frame.f1
pack .frame.f1 -side top -pady 2 -anchor e
label .frame.f1.label1\
    -text 'number of work station per token ring' -anchor e
entry .frame.f1.a -width 10 -relief sunken -textvariable A
pack .frame.f1.label1 .frame.f1.a -side left

frame .frame.f2
pack .frame.f2 -side top -pady 2 -anchor e
label .frame.f2.label2\
    -text 'number of token ring in local area' -anchor e
entry .frame.f2.b -width 10 -relief sunken -textvariable B
pack .frame.f2.label2 .frame.f2.b -side left

frame .frame.f3
pack .frame.f3 -side top -pady 2 -anchor e
label .frame.f3.label3\
    -text 'local token ring traffic (0 <= < 1.0)' -anchor e
entry .frame.f3.c -width 10 -relief sunken -textvariable C
pack .frame.f3.label3 .frame.f3.c -side left

frame .frame.f4
pack .frame.f4 -side top -pady 2 -anchor e
label .frame.f4.label4\
    -text 'number of token ring in remote area' -anchor e
entry .frame.f4.d -width 10 -relief sunken -textvariable D
pack .frame.f4.label4 .frame.f4.d -side left

frame .frame.f5
pack .frame.f5 -side top -pady 2 -anchor e
label .frame.f5.label5\
    -text 'remote token ring traffic(0 <= < 1.0)' -anchor e
entry .frame.f5.e -width 10 -relief sunken -textvariable E
pack .frame.f5.label5 .frame.f5.e -side left

label .frame.note1 -text 'per local token ring'
pack .frame.note1 -side top

frame .frame.f6
pack .frame.f6 -side top -pady 2 -anchor e
label .frame.f6.label6\
    -text 'number of user to remote form database with light transaction'\
    -anchor e
entry .frame.f6.f -width 10 -relief sunken -textvariable F
pack .frame.f6.label6 .frame.f6.f -side left

frame .frame.f7
pack .frame.f7 -side top -pady 2 -anchor e
label .frame.f7.label7\
    -text 'number of user to remote form database with heavy transaction'\
    -anchor e
entry .frame.f7.g -width 10 -relief sunken -textvariable G
pack .frame.f7.label7 .frame.f7.g -side left

frame .frame.f8
pack .frame.f8 -side top -pady 2 -anchor e
label .frame.f8.label8\
    -text 'number of user to B & C database with light transaction'\
    -anchor e
entry .frame.f8.h -width 10 -relief sunken -textvariable H
pack .frame.f8.label8 .frame.f8.h -side left

frame .frame.f9
pack .frame.f9 -side top -pady 2 -anchor e
label .frame.f9.label9\
    -text 'number of user to B & C database with heavy transaction'\
    -anchor e
entry .frame.f9.i -width 10 -relief sunken -textvariable I
pack .frame.f9.label9 .frame.f9.i -side left

frame .frame.f21
pack .frame.f21 -side top -pady 2 -anchor e
label .frame.f21.label21\
    -text 'number of user to local server only'\
    -anchor e
entry .frame.f21.u -width 10 -relief sunken -textvariable U
pack .frame.f21.label21 .frame.f21.u -side left

label .frame.note2 -text 'per remote token ring'
pack .frame.note2 -side top

frame .frame.f10
pack .frame.f10 -side top -pady 2 -anchor e
label .frame.f10.label10\
    -text 'number of user to remote form database with light transaction'\
    -anchor e
entry .frame.f10.j -width 10 -relief sunken -textvariable J
pack .frame.f10.label10 .frame.f10.j -side left

```

```

frame .frame.f11
pack .frame.f11 -side top -pady 2 -anchor e
label .frame.f11.label11\
    -text 'number of user to remote form database with heavy transaction'\
    -anchor e
entry .frame.f11.k -width 10 -relief sunken -textvariable K
pack .frame.f11.label11 .frame.f11.k -side left

frame .frame.f12
pack .frame.f12 -side top -pady 2 -anchor e
label .frame.f12.label12\
    -text 'number of user to B & C database with light transaction'\
    -anchor e
entry .frame.f12.l -width 10 -relief sunken -textvariable L
pack .frame.f12.label12 .frame.f12.l -side left

frame .frame.f13
pack .frame.f13 -side top -pady 2 -anchor e
label .frame.f13.label13\
    -text 'number of user to B & C database with heavy transaction'\
    -anchor e
entry .frame.f13.m -width 10 -relief sunken -textvariable M
pack .frame.f13.label13 .frame.f13.m -side left

frame .frame.f22
pack .frame.f22 -side top -pady 2 -anchor e
label .frame.f22.label22\
    -text 'number of user to local server only'\
    -anchor e
entry .frame.f22.v -width 10 -relief sunken -textvariable V
pack .frame.f22.label22 .frame.f22.v -side left

label .frame.note3 -text ' '
pack .frame.note3 -side top

frame .frame.f14
pack .frame.f14 -side top -pady 2 -anchor e
label .frame.f14.label14\
    -text 'light transaction service time of remote form database(second)'\
    -anchor e
entry .frame.f14.n -width 10 -relief sunken -textvariable N
pack .frame.f14.label14 .frame.f14.n -side left

frame .frame.f15
pack .frame.f15 -side top -pady 2 -anchor e
label .frame.f15.label15\
    -text 'heavy transaction service time of remote form database(second)'\
    -anchor e
entry .frame.f15.o -width 10 -relief sunken -textvariable O
pack .frame.f15.label15 .frame.f15.o -side left

frame .frame.f16
pack .frame.f16 -side top -pady 2 -anchor e
label .frame.f16.label16\
    -text 'light transaction service time of B & C database(second)'\
    -anchor e
entry .frame.f16.p -width 10 -relief sunken -textvariable P
pack .frame.f16.label16 .frame.f16.p -side left

frame .frame.f17
pack .frame.f17 -side top -pady 2 -anchor e
label .frame.f17.label17\
    -text 'heavy transaction service time of B & C database(second)'\
    -anchor e
entry .frame.f17.q -width 10 -relief sunken -textvariable Q
pack .frame.f17.label17 .frame.f17.q -side left

label .frame.note4 -text ' '
pack .frame.note4 -side top

frame .frame.f23
pack .frame.f23 -side top -pady 2 -anchor e
label .frame.f23.label23\
    -text 'random traffic of WAN (0 <= < 1.0)'\
    -anchor e
entry .frame.f23.x -width 10 -relief sunken -textvariable X
pack .frame.f23.label23 .frame.f23.x -side left

frame .frame.f24
pack .frame.f24 -side top -pady 2 -anchor e
label .frame.f24.label24\
    -text 'random traffic of back bone (0 <= < 1.0)'\
    -anchor e
entry .frame.f24.y -width 10 -relief sunken -textvariable Y
pack .frame.f24.label24 .frame.f24.y -side left

frame .frame.f25
pack .frame.f25 -side top -pady 2 -anchor e
label .frame.f25.label25\
    -text 'random utilization of mainframe for B & C database (0 <= < 1.0)'\
    -anchor e
entry .frame.f25.z -width 10 -relief sunken -textvariable Z

```

```

pack .frame.f25 label25 .frame.f25.z -side left

label .frame.note5 -text 'following 4-layer model only'
pack .frame.note5 -side top

frame .frame.f18
pack frame.f18 -side top -pady 2 -anchor e
label .frame.f18.label18\
    -text 'intermediate server light transaction service time (second)' -anchor e
entry .frame.f18.r -width 10 -relief sunken -textvariable R
pack .frame.f18.label18 frame.f18.r -side left

frame .frame.f19
pack frame.f19 -side top -pady 2 -anchor e
label .frame.f19.label19\
    -text 'intermediate server heavy transaction service time (second)' -anchor e
entry .frame.f19.s -width 10 -relief sunken -textvariable S
pack .frame.f19.label19 frame.f19.s -side left

frame .frame.f20
pack frame.f20 -side top -pady 2 -anchor e
label .frame.f20.label20\
    -text 'fraction of users to intermediate server' -anchor e
entry .frame.f20.t -width 10 -relief sunken -textvariable T
pack frame.f20.label20 frame.f20.t -side left

# set .menu help button
# set .menu help m help
# quit

frame .menu -relief raised -borderwidth 1
pack .menu -before frame -side top -fill x

button .menu.help -text 'help' -command 'exec input_p_help'
button .menu.quit -text 'quit' -command 'exit'
pack .menu.help -side left
pack .menu.quit -side right

# set buttons put buttons
# .buttons.start open input file
# .buttons.end close input file
# .button.clear clear screen
# button.ok exit the window

frame .buttons
pack .buttons -side bottom -pady 2 -anchor center

button .buttons.start\
    -text 'open input file'\
    -command {set inf [open "input_file" w]}
button .buttons.end -text 'close input file'\
    -command {close $inf}
button .buttons.save -text 'save to file'\
    -command {exec save_to_file}
button .buttons.clear\
    -text clear
button .buttons.ok\
    -text 'OK'\
    -command 'exit'
pack .buttons.start .buttons.end .buttons.save .buttons.clear .buttons.ok\
    -side left -padx 2

# set bind for all input entry

bind .frame.f1.a <Return> {puts $inf A; puts $inf $A}
bind .frame.f2.b <Return> {puts $inf B; puts $inf $B}
bind .frame.f3.c <Return> {puts $inf C; puts $inf $C}
bind .frame.f4.d <Return> {puts $inf D; puts $inf $D}
bind .frame.f5.e <Return> {puts $inf E; puts $inf $E}
bind .frame.f6.f <Return> {puts $inf F; puts $inf $F}
bind .frame.f7.g <Return> {puts $inf G; puts $inf $G}
bind .frame.f8.h <Return> {puts $inf H; puts $inf $H}
bind .frame.f9.i <Return> {puts $inf I; puts $inf $I}
bind .frame.f10.j <Return> {puts $inf J; puts $inf $J}
bind .frame.f11.k <Return> {puts $inf K; puts $inf $K}
bind .frame.f12.l <Return> {puts $inf L; puts $inf $L}
bind .frame.f13.m <Return> {puts $inf M; puts $inf $M}
bind .frame.f14.n <Return> {puts $inf N; puts $inf $N}
bind .frame.f15.o <Return> {puts $inf O; puts $inf $O}
bind .frame.f16.p <Return> {puts $inf P; puts $inf $P}
bind .frame.f17.q <Return> {puts $inf Q; puts $inf $Q}
bind .frame.f18.r <Return> {puts $inf R; puts $inf $R}
bind .frame.f19.s <Return> {puts $inf S; puts $inf $S}
bind .frame.f20.t <Return> {puts $inf T; puts $inf $T}
bind .frame.f21.u <Return> {puts $inf U; puts $inf $U}
bind .frame.f22.v <Return> {puts $inf V; puts $inf $V}
bind .frame.f23.x <Return> {puts $inf X; puts $inf $X}
bind .frame.f24.y <Return> {puts $inf Y; puts $inf $Y}
bind .frame.f25.z <Return> {puts $inf Z; puts $inf $Z}

# clear button function

```

```

proc clearAction {} {
    frame.f1.a delete 0 end
    frame.f2.b delete 0 end
    frame.f3.c delete 0 end
    frame.f4.d delete 0 end
    frame.f5.e delete 0 end
    frame.f6.f delete 0 end
    frame.f7.g delete 0 end
    frame.f8.h delete 0 end
    frame.f9.i delete 0 end
    frame.f10.j delete 0 end
    frame.f11.k delete 0 end
    frame.f12.l delete 0 end
    frame.f13.m delete 0 end
    frame.f14.n delete 0 end
    frame.f15.o delete 0 end
    frame.f16.p delete 0 end
    frame.f17.q delete 0 end
    frame.f18.r delete 0 end
    frame.f19.s delete 0 end
    frame.f20.t delete 0 end
    frame.f21.u delete 0 end
    frame.f22.v delete 0 end
    frame.f23.x delete 0 end
    frame.f24.y delete 0 end
    frame.f25.z delete 0 end
}
.buttons.clear.config -command clearAction

```

CODE 10 put_to_file

```

#!/usr/local/bin/wish -f

# set frame for input file name
# frame.label label
# .frame.name entry

frame frame -relief flat
pack frame -side top -fill y -anchor center

label frame.label\
    -text 'put output to file:' -anchor e
entry frame.name -width 15 -relief sunken -textvariable A
pack frame.label frame.name -side left

# set buttons put buttons
# buttons.input copy input file to INPUT
# button.ok exit the window

frame buttons
pack buttons -side bottom -pady 2 -anchor center

button buttons.ok -text 'OK'
button buttons.exit -text 'quit' -command 'exit'
pack buttons.ok buttons.exit -side left -padx 6 -anchor e

# set bind for all input entry

bind buttons.ok <Button-1> {exec cp output_file $A}

```

CODE 11 put_to_file

```

#!/usr/local/bin/wish -f

# set frame for input file name
# frame.label label
# frame.name entry

frame frame -relief flat
pack frame -side top -fill y -anchor center

label frame.label\
    -text 'save input as file:' -anchor e
entry .frame.name -width 15 -relief sunken -textvariable A
pack frame.label frame.name -side left

# set buttons put buttons
# buttons.input copy input file to INPUT
# button.ok exit the window

frame buttons
pack buttons -side bottom -pady 2 -anchor center

button buttons.ok -text 'OK'
button buttons.exit -text 'quit' -command 'exit'
pack buttons.ok buttons.exit -side left -padx 6 -anchor e

# set bind for all input entry

```



```
bind buttons ok <Button-1> {exec cp input_file $A}
```

CODE 12: show_output

```
#!/usr/local/bin/wish -f
# wish -f
# read from a file and then write to a message
```

```
wm title . ' '

set new {}
lappend new ' '
set trsf [open "output_file" r]
while {[gets $trsf line] >= 0} {
    lappend new $line, lappend new "\n"
}
lappend new ' '
close $trsf
set new [join $new]
message show -width 700 -text $new
button .quit -text OK -width 5 -command 'exit'
pack .show -side top -anchor center
pack .quit -side top -anchor center
```

CODE 13: usage

```
#!/usr/local/bin/wish -f
# wish -f
# set up usage message
wm title . ' '

set new {}
(1) select input {\n
    file input or {\n
        parameters input {\n
            default: input by file == input_file {\n
                and you can edit input_file {\n
(2) select solver {\n
    3-layer model or {\n
    4-layer model {\n
(3) select output {\n
    output to a file or {\n
    output to a window or {\n
    print output {\n
    default: output to file == output_file"
message .title -text USAGE -width 50
message .show -width 400 -text $new
button .q -text OK -width 5 -command 'exit'
pack .tit .show quit -anchor center
```

CODE 14: browse_input

```
#!/usr/local/bin/wish -f
#
# This script generates a directory browser, which lists the working
# directory and allows you to copy file to input_file by double-clicking
#
# Create a scrollbar on the right side of the main window and a listbox
# on the left side.
```

```
# frame      put .frame scroll and frame list
# quit      put quit button
```

```
# frame .frame -relief flat
# pack .frame -side top -fill y -anchor center
```

```
scrollbar .scroll -command 'list yview'
pack .scroll -side right -fill y
listbox .list -yscroll 'scroll se' -relief raised -geometry 20x20 \
    -setgrid yes
pack .list -side left -fill both -expand yes
wm minsize 1 1
```

```
button .quit -text 'quit' -command 'exit'
pack .quit -side bottom
```

```
# The procedure below is invoked to open a browser on a given file: if the
# file is a directory then another instance of this program is invoked, if
# the file is a regular file then the Mx editor is invoked to display
# the file
```

```
proc browse (dir file) {
    global env
    if {[string compare $dir ' ' ''] > 0} {set file $dir/$file}
    if {[file isdirectory $file]} {
        exec browse $file &
    } else {
        if {[file isfile $file]} {
```

```
        if {[info exists env(EDITOR)]} {
            eval exec $env(EDITOR) $file &
        } else {
            if {[string match $file input_file]} {
                exit
            } else { exec cp $file input_file
                exit
            }
        }
    } else {
        puts stdout "'$file' isn't a directory or regular file"
    }
}
```

```
# Fill the listbox with a list of all the files in the directory (run
# the 'ls' command to get that information)
```

```
if $argc>0 {set dir [lindex $argv 0]} else {set dir ' '}
foreach i [exec ls -a $dir] {
    list insert end $i
}
```

```
# Set up bindings for the browser
```

```
bind list <Control-q> {destroy }
bind .list <Control-c> {destroy }
focus .list
bind list <Double-Button-1> {foreach i [selection get] {browse $dir $i}}
```

END

2 1 - 1 1 - 9 6

FIN