Resource Allocation Algorithms in AON Network

by

May Ku

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Science

at the

Massachusetts Institute of Technology

February 1994

© May Ku

The author hereby grants to MIT permission to reproduce and to distribute copies of this thesis document in whole or in part.

Signature of Author	
Department of Electric	al Engineering and Computer Science January 14, 1994
Certified by	•••••••••••••••
Lecturer, Department of Electric	Dr. Steven G. Finn al Engineering and Computer Science Thesis Supervisor
Accepted by	Prof. Frederic R. Morgenthaler air, Committee on Graduate Students

Resource Allocation Algorithms in AON Network by May Ku

Submitted to the Department of Electrical Engineering and Computer Science on January 14, 1994, in partial fulfillment of the requirements for the degree of

Master of Science

in Electrical Engineering and Computer Science

Abstract

This thesis presents a study of scheduling algorithms for allocating system resources in the lowest level of a wideband All Optical Network (AON) proposed by a consortium of AT&T, DEC and MIT. Three scheduling algorithms are considered and applied to uniform traffic, multiclass traffic, and client/server traffic for both blocking and queueing systems. We present mathematical approximations and bounds for several queueing and blocking systems. Simulations using OPNET software were run for these scheduling algorithms and compared to the mathematical approximations and bounds. From our study we conclude that a Random Assignment Scheduling Algorithm seems to be a very promising scheduling approach for the lowest level of the proposed AON network.

Thesis Supervisor: Dr. Steven G. Finn Title: Lecturer, Department of Electrical Engineering and Computer Science

Acknowledgments

This thesis is dedicated to my family.

I would especially like to thank my thesis advisor, Dr. Steven Finn, for guiding and encouraging me through this endeavor.

I would very much like to thank Nate Osgood for finding six SPARC 10 machines to run my simulations, for keeping crazy hours to ensure the runs would be kicked off promptly during the final rush to get the work complete. Your time and efforts are greatly appreciated.

The work was performed at the Laboratory for Information and Decision Systems (LIDS) at MIT. I would like to thank Prof. Robert Kennedy, Prof. Robert Gallager, Prof. Mitch Trott and Sheila Hegarty of LIDS for their various help.

Many thanks to my friend Hua Xu, whom I can always count on for emotional support. Thanks to my former colleagues at IBM, Darlene Shamsid-Deen, Tom Jung, Jim Kelly, Bob Lannin who were always helpful and encouraging.

This research was partially supported by the Advanced Research Project Administration Grant #MDA972-92-J-1038.

Contents

Chapt	er 1 Introduction 1
Chapte	er 2 AON Network 3
2.1	Network Architecture
2.2	Network Services4
2.3	Network Operations4
Chapt	er 3 Network Model and Problem Description 7
3.1	System Resource Allocation
3.2	Number of Transceivers
3.3	Tuning, Modulation, Turn On/Off Time8
3.4	Session Distribution between OTs10
3.5	Queueing / Blocking System
3.6	Scheduling Approach
3.7	Scheduling Algorithms Considered
3.8	System Characteristics and Traffic Models12
3.9	Problem Summary14
Chapte	er 4 Mathematical Approximations and Bounds 17
4.1	Offered Load17
4.2	Single Class, Uniform Traffic System184.2.1Blocking System184.2.2Queueing System23
4.3	Two Class, Uniform Traffic System

Chapte	er 5 (OPNET Simulation Model	31
5.1	Blocki	ng System	32
5.2	Queue	ing System	33
5.3	Schedu	lling Algorithms	33
	5.3.1	Single Class System	34
	5.3.2	Two Class System	37
Chapte	er 6 S	Simulation Results	39
6.1	Single	Class, Uniform Traffic System	39
	6.1.1	Blocking System	39
	6.1.2	Queueing System	44
6.2	Two C	lass, Uniform Traffic System	48
	6.2.1	Blocking System	48
	6.2.2	Queueing System	51
6.3	Single	Class, Client/Server System	53
	6.3.1	Blocking System	53
	6.3.2	Queueing System	55
Chapte	er7S	Summary	59
Appen	dix A	Mathematical Calculations	61
A.1	Sessio	on Blocking Probability	61
A.2	Single	e Class, Uniform Traffic System	64
	A.2.1	Blocking System	64
	A.2.2	Queueing System	68
A.3	Two (Class, Uniform Traffic System	68
	A.3.1	Blocking System	68
Appen	dix B	OPNET Models	71
B.1	Netwo	ork Model	71
B.2	Node	Model	71
B.3	Single	e Class, Uniform Traffic System	74

	B.3 .1	Blocking System	74
	B.3.2	Queueing System	87
B.4	Two C	lass, Uniform Traffic System	
	B.4 .1	Blocking System	
	B.4.2	Queueing System	115
B.5	Single	Class, Client/Server System	
	B.5 .1	Blocking System	
	B.5.2	Queueing System	
eferer	nces		135

References

iv

List of Figures

Figure 2.1.1	AON Network	3
Figure 3.1.1	System Resources	7
Figure 3.3.1	Session Assignment	9
Figure 4.2.1.1	M / M / WT' / WT' System	18
Figure 4.2.1.2	Mathematical Approximation for M/M/1/B/L+1 System	20
Figure 4.2.1.3	Mathematical Approximation for M/M/1/B/CL System	21
Figure 4.2.1.4	Mathematical Bounds for M/M/1/B/RL System	22
Figure 4.2.2.1	Mathematical Bounds for M/M/1/Q/L+1 System	24
Figure 4.2.2.2	Mathematical Bounds for M/M/1/Q/CL System	25
Figure 4.2.2.3	Mathematical Bounds for M/M/1/Q/RL System	26
Figure 4.3.1.1	Markov Model for M/M/2/B/* System	27
Figure 4.3.1.2	Mathematical Analysis for M/M/2/B/* System	29
Figure 5.0.1	Node Module	31
Figure 5.1.1	Release Process Module	32
Figure 5.1.2	Scheduler Process Module for Blocking System	32
Figure 5.2.1	Scheduler Process Module for Queueing System	33
Figure 6.1.1.1	M/M/1/B/L+1 System Results	40
Figure 6.1.1.2	M/M/1/B/CL System Results	41
Figure 6.1.1.3	M/M/1/B/RL System Results	42
Figure 6.1.2.1	M/M/1/Q/L+1 System Results	45
Figure 6.1.2.2	M/M/1/Q/CL System Results	46

Figure 6.1.2.3	M/M/1/Q/RL System Results	47
Figure 6.2.1.1	M/M/2/B/* System	50
Figure 6.2.1.2	M/M/2/B/* System	51
Figure 6.2.2.1	M/M/2/Q/RL System	52
Figure 6.3.1.1	M/M/1/B/RL Client-Server System	54
Figure 6.3.1.2	M/M/1/B/RL Client-Server System	55
Figure 6.3.2.1	M/M/1/Q/RL Client-Server System	56
Figure 6.3.2.2	M/M/1/Q/RL Client-Server System	57
Figure 6.3.2.3	M/M/1/Q/RL Client-Server System	58

Recent advances in optical fiber technology makes it the preferred transmission medium for long-distance, point-to-point communications links. In U.S. alone, more than two million miles of fiber has been installed by long distance phone companies[1]. However, they are mostly operated at a capacity much lower than their terahertz potential. One current research topic is how to build an optical fiber communication network that will use the fiber bandwidth more effectively. Various ideas have been proposed. A popular approach is to employ *wavelength-division multiplexing* (WDM) which divides the optical spectrum into many different wavelengths, each corresponding to a different communications channel.

WDM networks can be further categorized. One could use a bus or star topology. The transmitter and receiver can be fixed or dynamically tuned to available wavelength channels. Various media access (MAC) protocols from fixed assignment to random access can be used. There are single-hop or multihop networks, where in a single-hop network, any two nodes can talk directly to each other via a wavelength channel. In multihop networks some node pairs may need to route through intermediary node(s) since they don't share the same wavelength channels. (For a review of various proposed WDM network, see [1,2,3].)

A consortium of AT&T, DEC, and MIT has proposed a WDM based wideband All-Optical Network (AON)[4]. It uses tree-of-stars topology at its lowest level and has a hierarchical structure. Each node has a tunable transmitter and receiver, thus all nodes can talk directly to each other (i.e. single-hop network). A demand assigned "scheduled TDM" MAC protocol is proposed for local communications, where a central agent(s) is responsible for allocating the time slots to requesting terminals.

This thesis will study scheduling algorithms for allocating the system resource in the lowest level of the AON network. We will conclude from our study that a *Random Assignment Scheduling Algorithm* developed in this thesis seems to be a very promising scheduling approach for the AON level 0 subnetwork. It appears to work across a wide variety of traffic requirements including uniform traffic, multiclass traffic, and client/server traffic.

The remainder of the thesis is organized as follows. Chapter 2 gives an overview of the AON network's architecture, service, and network operations. Chapter 3 establishes the network model and defines the problem. Chapter 4 derives the mathematical approximations or bounds for systems studied. Chapter 5 describes the OPNET simulation package that we will be using, specifies the network and node models used for the simulation. It also discuss the three scheduling algorithms used. Chapter 6 discuss the simulation results and compare them to mathematical bounds. Chapter 7 discusses our conclusions. Appendix A includes the C programs used to calculate the mathematical formulas, and in Appendix B are the reports from OPNET.

1

An All-Optical Network (AON) has been architected, and a test bed based on this design will be built to investigate the utilization of terahertz bandwidth capacity of optical networks by a ARPA sponsored consortium of AT&T, DEC, and MIT[4].

In the AON network, optical signals flow across the network without being converted to electrical signals. The network is designed to be scalable in the dimensions of geographic span, the number of users, and data rate. It employs wavelength division multiplexing (WDM) and time division multiplexing (over each wavelength) techniques to access the fiber bandwidth. Frequency reuse is utilized to enable network expansion over multiple geographical areas.

2.1 Network Architecture



Figure 2.1.1 AON Network

The AON is a hierarchical network with three levels (L0, L1, L2) of sub-networks as shown in Figure 2.1.1. It is designed to scale gracefully to hundreds of thousands of all optical end nodes. One can consider L2 as the backbone of a national or worldwide network, L1 as a Metropolitan Area Network (MAN), and L0 as a Local Area Network (LAN). The lowest level L0 is a "local" broadcast star network. Optical Terminals (OTs) are attached to the AON via an L0 subnet. Within the L0 subnet, optical wavelengths are divided into three sets.

• L0 wavelengths: this wavelength set is used for local traffic between OTs within the same L0 subnet. L0 wavelengths are blocked from entering the L1 level by a frequency selective local bypass element located at the exit link of L0 to L1 subnet. These wavelengths may be reused in L2 and L1 subnets, as well as other L0 sub-networks.

• L1 wavelengths: this wavelength set is used for communication between OTs in different L0 subnets, which requires transmission through an L1 subnet.

• Control wavelength: this wavelength is dedicated for control, scheduling, network management, and datagram services.

The Media Access Control (MAC) protocol for the control wavelength channel is designed not to require a central resource or central timing since this channel is used for power-on configuration of the network. An Ethernet protocol based upon IEEE standard 10Broad36 will be used in the test-bed.

The L0 and L1 wavelength channels are allocated by a central scheduling agent located within the respective L0 and L1 subnet. Depending on the incoming request, a wavelength channel may be allocated as a whole, or as subunits by using "scheduled" time division multiplexing (TDM) techniques. The L0 subnet is a broadcast star network and doesn't support wavelength routing. We will discuss "scheduled TDM" in more detail in Chapter 3.

Multiple L0's may be connected to a L1, which is connected to a L2. There is a single L2 subnet in the network acting as the backbone. In each L0, L1 and L2 subnet, there is a dedicated control wavelength in addition to data wavelengths. In each subnet, there is a scheduling agent responsible for allocating the data wavelength channels as requested. Both L1 and L2 subnets support wavelength routing.

2.2 Network Services

Three basic services are provided by the AON network.

• Type-A "switched - physical circuit" services provide point-to-point or point-to-multipoint high speed circuit switched photonic sessions. It uses the entire bandwidth of a wavelength channel. The scheduling agent will allocate an entire wavelength channel to Type-A session.

• Type-B "scheduled TDM" services provide time division multiplexed (TDM) circuitswitched sessions in the range of a few Mbps to the full optical channel rate. It uses a portion of a wavelength channel. When we have Type-B session requests coming in, the scheduling agent will divide the wavelength channel(s) into slots using "scheduled TDM", and allocate slot(s) as requested.

• Type-C "unscheduled datagram" services use a dedicated "well-known" wavelength (i.e. control wavelength) for control, scheduling, network management, and datagram services. No scheduling is necessary for Type-C communication packets.

2.3 Network Operations

Optical Terminals send Type-A and Type-B session requests to the L0 scheduler via Type-

C packets. Upon receiving the request, the scheduler determines if adequate resource is available. If so, the scheduler allocates the resource and informs the destination(s) of the new session request. If the destination subsequently accepts the connection request, the requesting OT is reliably informed and the session begins. All sessions are unidirectional.

Scheduling of Type-A session is relatively straightforward. The scheduler needs to know that both the source and the destination have a free transmitter and receiver respectively, and a free wavelength channel is available. The scheduler informs the source and the destination of the wavelength channel to use, so they can tune their respective transmitter and receiver to the wavelength to start the session. For type-B sessions, since wavelength channels are time divided into slots, the scheduler needs to find enough slots to satisfy the session throughput request. It also has to make certain that both the source's transmitter and the destination's receiver are free during these slot intervals.

The frequency reuse property of the network gives the scheduler full control of its own resource. Therefore a session between OTs in the same L0 (intra-L0 session) can be established by the local L0 scheduler. Sessions between OTs in different L0's require L1 and possibly L2 resource, and cannot be scheduled by the local L0 scheduler alone. We will limit our study to the first case, and leave L0/L1/L2 scheduler cooperation to future investigation. From now on, network or system means a L0 subnet and available wavelength channels are the L0 wavelength channels for use of intra-L0, point-to-point Type-B sessions.

In this thesis, we will study the scheduling algorithm used for resource allocation for intra-L0, point-to-point Type-B sessions.

In this chapter we discuss the aspects of the AON network that are relevant to our study of intra-L0, point-to-point, Type-B session scheduling, and define the problem we will study.

3.1 System Resource Allocation

To access the system resource, wavelength division multiplexing (WDM) and time division multiplexing (over each wavelength) techniques are used. The available fiber bandwidth is divided into W wavelength channels of equal bandwidth. Transmission over each wavelength channel is organized in frames of equal size T. All frames over different wavelength channels are aligned.

A frame is said to have size *T* if it is divided into *T* slots of equal time duration. For a system with *W* wavelength channels and frame size *T*, the total number of available slots is *WT*. They are demand assigned to sessions by the scheduler. Each slot is referred to by its wavelength number and the position in the frame, (w, t), where $w \in (\lambda_1, ..., \lambda_w)$ and $t \in (1, ..., T)$. Slot A in Figure 3.1.1 is referred to as $(\lambda_3, 2)$, or simply as (3,2). A row of slots means all slots with the same wavelength number and different slot number. A column of slots means all slots with the same slot number and different wavelength channels. So row $i = \{(\lambda_i, j) : j = 1, ..., T\}$, and column $j = \{(\lambda_i, j) : i = 1, ..., W\}$.

When an Optical Terminal(OT) needs to establish a session, it sends the scheduler information on itself (source), the destination, and the throughput requirement L in terms of the number of slots needed per frame. The scheduler is responsible for allocating the required resource. Once the slot(s) is allocated, the session will use the same slot(s) in all subsequent frames until it terminates. Since we are primarily concerned here with the efficiency of the scheduling algorithm, we ignore the processing time it takes to establish a session.



Figure 3.1.1 System Resources

7

3.2 Number of Transceivers

In the study, we assume each OT has only one tunable transmitter and one tunable receiver for Type-B sessions. Consequently, each OT can transmit or receive only on one wavelength at a time. Referring to Figure 3.1.1, a session which needs two slots can be assigned slot (3,2) and (1,4), but not (3,2) and (1,2). However an OT may be transmitting at (3,2) and receiving at (1,2). If an OT is transmitting over multiple slots, all slots must have different slot numbers. This constitutes the most basic constraint on our scheduling algorithm, and gives arise to the concept of "column conflict", "pre-column conflict", and "post-column conflict".

• Column Conflict

A session (s,d) is one with node s as source and node d as destination. When allocating resource for session (s,d), if either node s is transmitting or node d is receiving over column j involving some other session, we say there is a column conflict over j due to transmitter or receiver conflict, and none of the slots in column j can be assigned to session (s,d).

• Pre-column Conflict

When allocating resource for session (s,d), if either node s is transmitting or node d is receiving over column j - l involving some other session(s) on wavelength λ_t or λ_r , we say there is a *pre-column conflict* over j due to transmitter or receiver conflict on wavelength λ_t or λ_r . It is possible that none of the slots in column j can be assigned to session (s,d) due to constraints on transmitter or receiver tuning times. Sometimes we will be able to resolve pre-column conflict as discussed in the next section.

Post-column Conflict

When allocating resource for session (s,d), if either node s is transmitting or node d is receiving over column j + 1 involving some other session(s) on wavelength λ_t or λ_r , we say there is a *post-column conflict* over j due to transmitter or receiver conflict on wavelength λ_t or λ_r . It is possible that none of the slots in column j can be assigned to session (s,d) due to constraints on transmitter or receiver tuning times. Sometimes we'll be able to resolve post-column conflict as discussed in the next section.

3.3 Tuning, Modulation, Turn On/Off Time

Since each transmitter and receiver can operate on any of the W wavelength channels, there is a finite tuning/modulation/turn on-off overhead time (will be referred to as tuning overhead time) required as it moves from wavelength channel to wavelength channel. This overhead varies as a function of the distance between the two wavelengths, and can be a source of inefficiency in the network. The following describes some possible methods that can be used to reduce the capacity lost to this tuning overhead (Refer to Figure 3.3.1 as we gradually build up our system from an empty one).



Figure 3.3.1 Session Assignment

• Avoid Tuning Overhead

Consider an empty system and a request of one slot for session $(a,e)^1$. Assume slot (3,1) is assigned to this session. A new request of one slot for session (a,f) comes in, and assume this is the only session involving node f's receiver. If we assign session (a,f) to any of the empty slots in row 3, no tuning overhead will occur since node a's transmitter is already at wavelength 3 and node f's receiver can be set to this wavelength. Any other assignment will result in tuning overhead for node a's transmitter. Let's assign the session to slot (3,4) as in Figure 3.3.1.

• Resolve Pre-column Conflict / Post-column Conflict

If request for a one slot session (b,e) comes in, there is *Pre-column Conflict* over column 2 due to node *e*'s receiver conflict on wavelength λ_3 . In this case we can still assign the session to slot (3,2) since node *e*'s receiver is already tuned to wavelength 3, so no additional tuning is needed. We say the *Pre-column Conflict* over column 2 is resolved. However, if slot (3,2) is already assigned to some other session, the *Pre-column Conflict* over column 2 is unresolvable, and none of the slots in column 2 can be used. The same applies to *Post-column Conflict*. The most restrictive situation is when *Pre-column Conflict* due to transmitter and receiver as well as *Post-column Conflict* due to transmitter and receiver occur. There are essentially three independent sessions between the same source and destination nodes, two already assigned and one needs to be assigned. The two assigned sessions are one slot number apart. The conflicts are resolvable only if the two assigned sessions have the same wavelength channel, and the slot between them with that wavelength channel is free to be assigned to the new session. We will end up with three sessions assigned to consecutive slots.

^{1.} session (a,e) is the notation for a communications channel from source node 'a' to destination node 'e'.

• Off-line Tuning

If the tuning can be done off-line, and assuming the modulation and turn on-off time is negligible, we can substantially reduce tuning overhead conflicts. A request for a one slot session (c,e) comes in, and can be assigned to slot (3,3) using off-line tuning technique. As shown in Figure 3.3.1, even though node c's transmitter is tuned to wavelength W for session (c,b) during slot 1 time, it can be tuned to wavelength 3 during its idle period of slot 2, and be ready for session (c,e) assigned slot (3,3).

• Combine Tuning into Data Slot

If the tuning time and the requested slots can be a fractional number, we can combine them into a slot. To illustrate the point, let's assume the tuning overhead uses 0.3 of a slot, and the session requires 1.6 slots. Instead of assigning one slot for tuning and another two slots for data for a total of three slots, we can assign two slots. The session will use the first 0.3 of the slot for tuning, followed by data immediately.

For our study, we will assume all tuning overhead and sessions require an integer number of slots. More specifically we will assume one slot for tuning overhead should it ever be needed.

3.4 Session Distribution between OTs

The L0 subnet is basically considered a campus-wide Local Area Network. The traditional LAN is built upon the client/server model. A server could be a file server, printer, gateway, time-sharing system, etc. One study of LAN traffic has shown that for one network measured, these identifiable servers sent about 69% and received about 73% of the packets over one typical day[5].

For either a client/server or distributed model, the L0 subnet may be highly compartmentalized. The users in the same department are more likely to talk to each other then to users outside their department. The study previously referred to reported that 72% of traffic measured was intranet or intradepartmental packets[5].

In our study, we will initially concentrate on a uniformly distributed traffic model. After we get an understanding of the uniform system, we will introduce a client/server traffic model into the subnet and see how it influences the system characteristics.

3.5 Queueing / Blocking System

In the queueing system model, the scheduler has a queue(s) to hold requests that cannot be immediately satisfied. It will try repeatedly until resource is found. In the blocking system, the scheduler does not have any queue. Any unsatisfied request is discarded, and needs to be regenerated by the terminal.

3.6 Scheduling Approach

• Contiguous-Slot (CS) and Random-Slot (RS) Assignment:

A session requesting multiple slots $\{(\lambda_i, j)\}$, tuning overhead included, is said to have CS assignment if, when arranging j in increasing order, the increment is always one. A scheduling algorithm is implementing CS if all sessions have CS assignment, otherwise it is implementing RS.

• Single-Wavelength (SW) and Multiple-Wavelength (MW) Assignment:

A session requesting multiple slots $\{(\lambda_i, j)\}$, tuning overhead included, is said to have SW assignment if all λ_i 's are identical. A scheduling algorithm is implementing SW if all sessions have SW assignment, otherwise it is implementing MW.

By combining variations on wavelength and slots, a scheduling algorithm can implement one of the four assignments, *SW-CS*, *SW-RS*, *MW-CS*, and *MW-RS*. In this work, we will be focusing on the most constraint *SW-CS* and the least constraint *MW-RS* assignment.

3.7 Scheduling Algorithms Considered

The following is the outline of the three different scheduling approaches to tuning overhead that we will consider.

• Contiguous L Assignment Algorithm (CL Assignment)

This scheduling algorithm allows no tuning overhead time on the common channel, and is used in conjunction with SW-CS assignment. When a session requests L slots per frame, it gets L slots. The tuning overhead is avoided by using the off-line tuning technique or by resolving *Pre-column* and *Post-column Conflicts* if possible. Otherwise the session request is either rejected or queued depending on the system. The advantage of this approach is algorithmic simplicity. There is no fragmentation to worry about, and a single class (L slots per session per frame) system with W wavelength channels and frame size T can be treated as a single class of one slot per session per frame system with W wavelength channels and frame size $\lfloor T/L \rfloor$ due to the SW-CS approach used. The disadvantage of this approach is that in a system with a small number of users, this algorithm can have low channel utilization because of rejections due to unresolved *Pre-column* or *Post-column Conflicts*. So that even if a block of L slots is free, it goes unused and thus reduces the channel utilization. The larger the L, the more pronounce the effect becomes. However when the number of users in the system is large, the algorithm should perform well since minimal tuning conflict is expected.

• Contiguous L+1 Assignment Algorithm (L+1 Assignment)

In this algorithm we always allocate one additional slot for tuning overhead, therefore

there is no *Pre-column* or *Post-column Conflicts* to worry about. It is used in conjunction with *SW-CS* algorithm, so all sessions are assigned a block of L+1 slots. As in the *Contiguous L Assignment Algorithm*, a single class (*L* slots per session per frame) system with *W* wavelength channels and frame size *T* can be treated as a single class of one slot per session per frame system with *W* wavelength channels and frame size $\lfloor T/(L+1) \rfloor$. Compared to *Contiguous L Assignment Algorithm*, since an overhead of one slot per *L* slots is introduced, the channel utilization will degrade in a system with large number of users. In a system with small population the cost in overhead can be traded off against the loss due to rejection resulting from unresolved *Pre-column* and *Post-column Conflicts*. The channel utilization can actually improve compared to *Contiguous L Assignment Algorithm*. The exact nature of the improved efficiency also depends on the value of *L*.

• Random L Assignment Algorithm (RL Assignment)

This algorithm uses MW-RS assignment and off-line tuning technique. The slot used for one session's off-line tuning can be used for another session's on-line data. This is made possible by the use of RS assignment. Off-line tuning also means that all sessions are assigned L slots as requested. This approach introduces fragmentation and additional algorithm complexity, but we hope it will utilize the available slots more fully and give added performance.

3.8 System Characteristics and Traffic Models

In this section, we define system characteristics and traffic models for a system with W number of wavelength channels and frame size T. The number WT describes the total number of slots per frame available for use. Both W and T are deterministic. We assume that each wavelength channel has fixed capacity, so the system capacity is linearly proportional to the number of wavelength channels.

A/B/C/D/E System Characteristics

• The first parameter A indicates the session arrival process. It is G for a general distribution of interarrival times, M for memoryless, specifically the Poisson process; and D for deterministic interarrival time.

• The second parameter B indicates the distribution of session service time (session hold time). It will be M, G, and D for exponential, general, and deterministic probability distribution, respectively.

We assume that successive interarrival times and service times are statistically independent of each other.

• The third parameter C indicates the distribution of the number of slots required per session. It will be M, G, and D for exponential, general, and deterministic probability distribution, respectively. When the number of slots required per session per frame is one of s predetermined values $\{L_1, \ldots, Ls\}$, we say the system is s-class, and is denoted by

numeral s. So a number 1 means single class, and 2 means two class system.

• The fourth parameter D indicates if it is a blocking(B) or a queueing(Q) system.

• The fifth parameter E gives the scheduling algorithms described in Section 3.7. It will be CL for "Contiguous L Assignment Algorithm", L+1 for "Contiguous L+1 Assignment Algorithm", and RL for "Random L Assignment Algorithm".

In this thesis, mathematical estimations/bounds are derived and OPNET simulations are run for the following system traffic models. The models are chosen since they represent some typical aspects of the system. The simplest model of single class and uniform traffic is first analyzed to give us some understanding of the efficiency of the different algorithms. To get a more accurately approximation of the real system traffic, we analyzed the two class and uniform traffic system. This system is simple enough to study yet it represents the multiclass SONET traffic that AON may carry. Finally to approximate the client/server situation, we analyze the single class and client/server traffic and compare that to single class and uniform traffic. From the study of these three systems, we want to observe how well the three scheduling algorithms perform, and when it is important to take tuning overhead into consideration.

System Traffic

1. Single Class and Uniform Traffic System

We will analyze the blocking and queueing systems respectively for the single class and uniform traffic system. For blocking system, M/M/1/B/CL, M/M/1/B/L+1, and M/M/1/B/RL characteristics are used. Sessions arrive according to a Poisson process with rate λ , exponential session hold time with mean $1/\mu$, and all sessions require L slots per frame. We also assume all sessions are uniformly distributed among optical terminals in the network. We will obtain a mathematical approximation and simulation result for M/M/1/B/L+1 and M/M/1/B/RL systems, mathematical bounds and simulation results in terms of channel utilization and blocking probability are obtained. The queueing system is the same as the blocking one except the unsatisfied session requests are queued. Again, mathematical bounds and simulation results for queueing delay and the average queue size are obtained for M/M/1/Q/CL, M/M/1/Q/L+1, and M/M/1/Q/RL systems.

This study will allow us to check our simulation and mathematical results against each other, and show how effectively the three scheduling algorithms deal with tuning overhead.

2. Two Class and Uniform Traffic System

We will study M/M/2/B/CL, M/M/2/B/L+1, and M/M/2/B/RL for blocking system and M/M/2/Q/RL for queueing system. Sessions arrive according to a Poisson process with rate λ_1 for those requiring L_1 slots per frame and λ_2 for those requiring L_2 slots per frame.

Session hold time is exponential with mean $1/\mu$ for both types of sessions. All sessions are uniformly distributed among optical terminals in the network. In the *Contiguous L* and L+1 Assignment Algorithms, the system is divided into two subsystems according to the relative traffic load of each type of sessions. Each subsystem serves only one type of sessions using Contiguous L or L+1 Assignment Algorithm. The advantage of this approach is that each type of sessions has its fair share of the system resource. The disadvantage is that accurate estimate of the relative traffic is needed in order to divided up the system resource. This problem can be solved by using Random L Assignment Algorithm which has additional complexity and may not be as fair. We will compare the three scheduling algorithms for the blocking system, and show that the Random L Assignment Algorithm has higher efficiency. For the queueing system, only M/M/2/Q/RL will be simulated.

3. Single Class and Client/Server Traffic System

This system is essentially the same as system 1. Only we will introduce server nodes into the system (N_s server nodes among a total of N nodes including servers). So in addition to uniform traffic among regular nodes, we have client/server traffic between the server nodes and the regular nodes (i.e. client nodes). The traffic break up is such that each server generates *st* percentage of sessions to and another *st* percentage from clients. The rest of the $1-N_s*2*st$ percentage of the traffic are among clients (this is the uniform part). Notice that since each transmitter or receiver can only use one out of W wavelength channels at a given time, *st* has to be less than or equal to 1/W before server's transmitter and receiver become system's bottleneck. We want to see how robust the result is from system 1.

3.9 Problem Summary

In this thesis, we will study scheduling algorithms for intra-L0, point-to-point, Type-B sessions in the AON network under the following assumptions:

- Zero propagation delay in the network.
- Zero processing delay in session scheduling.
- One tunable transmitter and tunable receiver per OT.
- The total system capacity is proportional to the number of wavelength channels.
- All tuning overhead and sessions require an integer number of slots.
- Poisson arrival process and exponentially distributed session service time.
- Successive interarrival time and session service time are statistically independent.

The approach we take is by studying the three "typical" system traffic models specified in Section 3.8, the single class uniform traffic blocking and queueing system, the two class uniform traffic blocking and queueing system, and the single class client/server traffic blocking and queueing system. The scheduling algorithms employed in studying these models are specified in Section 3.7, the Contiguous L Assignment Algorithm, Contiguous L+1 Assignment Algorithm, and Random L Assignment Algorithm.

What we will conclude from our study is that when the number of users in the system is relatively large compared to the number of wavelength channels, the effect of tuning overhead is minimal, and the efficiency closed to the bound can be obtained by *Random L* Assignment Algorithm which uses off-line tuning and MW-RS assignment approach.

The simulations will be run for system with frame size 128 as in the testbed built by the AON consortium. The number of wavelength channels will be 2, 4, and 8. The number of nodes in the system will be 8 and 40. So the ratio of nodes to wavelength channels takes on the value of 1, 2, 4, 5, 10, and 20. From this range of ratios we will be able to conclude how large the number of nodes in the system has to be, compared to the number of wavelength channels, for the effect of tuning overhead to be negligible. Finally the session throughput requirement will take on the value of 1, 3, and 12 in single class system to help us understand the effect of throughput requirement on the efficiency of scheduling algorithms. For two class system, the session throughput requirement will be 3 and 12. We vary the relative traffic load of each type of the session to understand its effect on the efficiency of scheduling algorithms. For single class client/server system, we fixed the total number of nodes to 40, and wavelength channels to 8, while the number of servers takes on 1 and 3, and server traffic percentage *st* takes on the value of 0.05 and 0.1 (the maximum *st* is 1/8 before server's transmitter and receiver become system bottleneck). We will compare the results to single class uniform traffic situations.

In this chapter we derive mathematical approximations or bounds for single class uniform traffic and two class uniform traffic systems.

4.1 Offered Load

Single Class

Assuming sessions arrive with Poisson rate λ , service time (i.e. session holding time) is exponentially distributed with mean $1/\mu$, and the session throughput requirement is L slots per frame.

Let the capacity of each wavelength channel be C_s bps, so the total system capacity is WC_s bps. Let the number of bits transmitted by each session be exponentially distributed with mean of K bits per session. If a session can use an entire wavelength channel, it takes on average $1/\mu_0 = K/C_s$ second to transmit. If it acquires only L slots per frame, it takes on average $1/\mu = KT/C_sL$ second to transmit. Therefore $1/\mu = T/\mu_0L$. The offered load is the percentage of system capacity used, $\rho = \lambda(\text{sessions/sec}) \cdot K(\text{bits/session}) / WC_s(\text{bps}) = \lambda L/\mu WT$.

In our simulations, we normalize $1/\mu_0$ to one, meaning the mean session holding time is one second had it been using an entire wavelength channel. When sessions only use L of the T slots, the mean session holding time is $1/\mu = T/L$.

$$\rho = \frac{\lambda}{\mu} \cdot \frac{L}{WT} \qquad and \qquad \frac{1}{\mu} = \frac{1}{\mu_0} \cdot \frac{T}{L}$$
(4.1.0.1)

Two Class

Assuming sessions with throughput requirement L_1 slots per frame arrive with Poisson rate $\lambda_1 = \alpha \lambda$, and those with throughput requirement L_2 slots per frame arrive with Poisson rate $\lambda_2 = (1-\alpha)\lambda$. Where λ is the overall session arrival rate, and α is the percentage of arrivals that are L_1 type sessions. Assume the service time (i.e. session holding time) is exponentially distributed with mean $1/\mu$ for all sessions.

Let the capacity of each wavelength channel be C_s bps, so the total system capacity is WC_s bps. Let the number of bits transmitted by type L_1 and L_2 sessions be exponentially distributed with mean of K_1 bits and K_2 bits per session respectively. Since type L_2 sessions have throughput requirement L_2/L_1 times that of type L_1 sessions and the session holding time is the same for both, we must have $K_2 / K_1 = L_2 / L_1$. When a type L_1 session acquires only L_1 slots per frame, it will take $1/\mu = K_1 T/C_s L_1$ second to transmit. Similarly $1/\mu = K_2 T/C_s L_2$, The offered load is the percentage of system capacity used, $\rho =$

{ $\lambda_1(\text{sessions/sec}) \cdot K_1(\text{bits/session}) + \lambda_2(\text{sessions/sec}) \cdot K_2(\text{bits/session})$ } / $WC_s(\text{bps}) = \lambda \{\alpha L_1 + (1 - \alpha)L_2\} \} / \mu WT.$

If we normalize the service time against type L_I sessions (i.e. assuming that each type L_I session can use an entire wavelength channel), the mean session holding time is $l/\mu_0 = K_I/C_s$. Since in reality each type L_I session uses only L_I of T slots, the mean session holding time is $l/\mu = T/\mu_0 L_I$. Similarly we will have $l/\mu = T/\mu_0 L_2$ and $l/\mu_0 = K_2/C_s$ if normalizing the service time against L2 type sessions. In either case we will set l/μ_0 to one when running simulations.

$$\rho = \frac{\lambda}{\mu} \cdot \frac{\alpha L_1 + (1 - \alpha) L_2}{WT} \qquad where \qquad \frac{1}{\mu} = \frac{1}{\mu_0} \cdot \frac{T}{L_1} \qquad or \qquad \frac{1}{\mu} = \frac{1}{\mu_0} \cdot \frac{T}{L_2}$$
(4.1.0.2)

depending on normalization of session service time against type L_1 or L_2 services.

4.2 Single Class, Uniform Traffic System

In this section, we derive mathematical approximation for blocking system and mathematical upper bound for queueing system.

4.2.1 Blocking System

In Appendix A.1 we derived $P_{b|Ns}$, the probability that a new session request will be rejected given the number of sessions in service Ns. Two assumptions were made, the existence of column conflict only and that all Ns sessions are equally distributed over slots in the system. The column conflict only assumption is valid for Contiguous L+1 Assignment Algorithm but not for Contiguous and Random L Assignment Algorithms. The assumption of equally distributed sessions over all slots in the system is a statistical approximation of the system. Therefore the $P_{b|Ns}$ thus derived is a conservative approximation but not the minimum for the system. And the channel utilization and the blocking probability derived below using this value of $P_{b|Ns}$ are also not the lower bounds but conservative approximations only.

Mathematical Approximation for M/M/1/B/L+1

By using *SW-CS* algorithm, each row can accommodate a maximum of $T' = \lfloor T/(L+1) \rfloor$ sessions. If we align these sessions at the boundary, we can consider the system as that of frame size T' and all sessions require one slot per frame. The system could be modeled by using Markov Chain.



Figure 4.2.1.1 *M / M / WT' / WT'* System

As shown in Figure 4.2.1.1, the system is equivalent to M/M/m/m blocking system with m = WT'. Let state *i* be the number of sessions in service, P_i be the probability that system will be in state *i* when in equilibrium. Obviously the system will advance from state *i* to state i+1 only if the new session request is accepted given state *i*. And the probability of acceptance is $1-P_{b|i}$, where $P_{b|i}$ is derived in Appendix A.1. Solving the following balance equation, and express it in terms of offered load $\rho = (\lambda/\mu)(L/WT)$,

$$Pi \cdot \lambda (1 - Pb|i) = P(i+1) \cdot (i+1) \mu$$

we have

$$Pi = Po \cdot \left(\rho \frac{WT}{L}\right)^{i} \cdot \frac{1}{i!} \cdot \prod_{j=0}^{i-1} \left(1 - Pb|j\right) \quad for \quad i = 0, ..., WT \quad (4.2.1.1)$$

where

$$Po^{-1} = 1 + \sum_{i=1}^{WT} \left(\rho \frac{WT}{L}\right)^{i} \cdot \frac{1}{i!} \cdot \prod_{j=0}^{i-1} \left(1 - Pb\right|j)$$
(4.2.1.2)

$$CU = \frac{L}{WT} \cdot \sum_{i=0}^{WT} i \cdot Pi \qquad and \qquad Pb = \sum_{i=0}^{WT} Pi \cdot Pb|i \qquad (4.2.1.3)$$

Where CU is the channel utilization, and P_b is the blocking probability of the system.

In Figure 4.2.1.2 we have shown the channel utilization and the blocking probability for systems with throughput requirement of one, three, and twelve slots per frame, frame size of 128, two, four, or eight wavelength channels, and eight or forty nodes. The same set of parameters will be used for simulation. The calculation is based upon the formulas above, the program is shown in Appendix A.2.1.

We can see from our analysis that the system performance depends on the ratio of N/W, where N is the number of nodes and W the wavelength channels in the system. We denote this ratio as γ . It describes the size of the system population relative to the wavelength number. The larger the population the less likely the session request is blocked by transmitter or receiver conflict, and the better the performance. From Figure 4.2.1.2 we see that a system with small γ performs considerably worse. However once $\gamma >>1$, the relative size of γ does not seem to be critical.

Notice that due to the SW-CS assignment nature of the algorithm, there are wasted slots in each row. Combining this inefficiency and the one slot tuning overhead per session, we can derive the maximum channel utilization for sessions when ignoring the transmitter and receiver conflicts. Assuming frame size of 128 and throughput requirement of L slots per frame, the maximum channel utilization is $L^*[T/(L+1)]/T$. So for L equals to one, three and twelve, the maximum channel utilization is 0.5, 0.75 and 0.84 respectively (shown as solid horizontal lines in Figure 4.2.1.2). The results for systems with large γ is very close to this absolute bound.



Figure 4.2.1.2 Mathematical Approximation for *M/M/1/B/L+1* System

Mathematical Approximation for M/M/1/B/CL

For Contiguous LAssignment Algorithm, the existence of pre-column and post-column conflict increases the probability that a new session will be rejected to above the $P_{b|Ns}$ value derived in Appendix A.1. Therefore $P_{b|Ns}$ is a conservative approximate lower bound for the system. Under the condition that we have large number of nodes (compared to the wavelength number), the occurrence of column conflict as well as pre-column and post-column conflict is rare. Therefore $P_{b|Ns}$ can be a tight approximate lower bound for

the system.

We use the same Markov Chain approach as in M/M/1/B/L+1 to solve for the system parameters. The only difference is now $T' = \lfloor T/L \rfloor$.

In Figure 4.2.1.3, we presented the results for systems with forty nodes, four wavelength channels, frame size of 128, and throughput requirement of one, three, and twelve slots per frame. Notice that M/M/1/B/CL system of large number of users is more efficient at steady state compared to M/M/1/B/L+1. The program in Appendix A.2.1 was used to do the calculations.

Notice that due to the SW-CS assignment nature of the algorithm, there are wasted slots in each row. Assuming frame size of 128 and throughput requirement of L slots per frame, the maximum channel utilization is $L^* \lfloor T/L \rfloor / T$. So for L equals to one, three and twelve, the maximum channel utilization is 1.0, 0.98 and 0.94 respectively (shown as dotted horizontal lines in Figure 4.2.1.3). The results shown in Figure 4.2.1.3 are close to this absolute bound.



Figure 4.2.1.3 Mathematical Approximation for M/M/1/B/CL System

Mathematical Bounds for M/M/1/B/RL

Ignoring transmitter and receiver conflicts, we can use M/M/m/m model to obtain the bound for the M/M/1/B/RL system. The number of servers $m = \lfloor WT/L \rfloor$. The channel utilization CU and the blocking probability Pb are as follows,

$$CU = \frac{L}{WT} \cdot \frac{\sum_{n=0}^{m} n \cdot (\lambda/\mu)^{n}/n!}{\sum_{n=0}^{m} (\lambda/\mu)^{n}/n!} \quad and \quad Pb = \frac{(\lambda/\mu)^{m}/m!}{\sum_{n=0}^{m} (\lambda/\mu)^{n}/n!} \quad (4.2.1.4)$$

where $\lambda/\mu = \rho WT/L$.

A system of WT slots can accommodate a maximum of $\lfloor WT/L \rfloor$ sessions as in a system of $\lfloor WT/L \rfloor$ servers. When taking transmitter and receiver conflicts into consideration, the system resource would be used less efficiently. So M/M/m/m model provides the upper bound for channel utilization and lower bound for blocking probability. And the absolute maximum channel utilization is $L^* WT/L WT$. In our system, the frame size is 128. For L of 1, the maximum channel utilization is 1.0; for L of 3, the maximum channel utilization is 0.996, 0.996, 0.999 for W of 2, 4 and 8 respectively; for L of 12, the maximum channel utilization is 0.984, 0.984, 0.996 for W of 2, 4 and 8 respectively.

In Figure 4.2.1.4 channel utilization and blocking probability are plotted for systems with two, four and eight wavelength channels, frame size of 128, and throughput requirement of one, three, and twelve slots per frame. Notice that the channel utilization approaches the absolute maximum value. The program for calculating the result is shown in Appendix A.2.1.



Figure 4.2.1.4 Mathematical Bounds for M/M/1/B/RL System

4.2.2 Queueing System

Ignoring transmitter and receiver conflicts, results from M/M/m system are used as bounds for M/M/1/Q/* systems. For M/M/m system, the following equations hold[7]:

$$Pq = \frac{p_0 (\lambda/\mu)^m}{m! (1 - \lambda/m\mu)}; \qquad Nq = \frac{(\lambda/m\mu) Pq}{1 - \lambda/m\mu}; \qquad Wq = \frac{(\lambda/m\mu) Pq}{\lambda (1 - \lambda/m\mu)} \qquad (4.2.2.1)$$

$$p_0 = \left\{ \sum_{k=0}^{m-1} \frac{(\lambda/\mu)^k}{k!} + \frac{(\lambda/\mu)^m}{m! (1 - \lambda/m\mu)} \right\}^{-1}$$
(4.2.2.2)

where P_q is the probability that an arriving session has to wait in queue, N_q is the average number of sessions in queue, W_q is the average waiting time in queue of a session, and $\lambda/\mu = \rho WT/L$.

For M/M/1/Q/L+1 system, the number of servers $m = W \lfloor T/(L+1) \rfloor$; for M/M/1/Q/CL system, $m = W \lfloor T/L \rfloor$; and for M/M/1/Q/RL system, $m = \lfloor WT/L \rfloor$. The results thus obtained, N_q and W_q , are lower bounds for the system since the value used for m is the maximum number of sessions the system can serve at any time for each of the scheduling algorithm. With the transmitter and receiver conflicts taken into consideration, the equivalent number of servers are smaller and the system is less efficient, which translates into larger queue size and longer waiting time in queue.

The program used for calculation is included in Appendix A.2.2.

Mathematical Bounds for M/M/1/Q/L+1

The number of servers *m* equals to W[T/(L+1)] as mentioned earlier. The arrival rate used in calculation is $\lambda = \rho W$ as obtained from Equation (4.1.0.1) and by setting $1/\mu_0$ to one as in simulations (*W* is the number of wavelength channels).

Same as that in M/M/1/B/L+1 system, the maximum load the system can handle without running into infinite queue and delay is $L^*[T/(L+1)]/T$. So for L equals to one, three and twelve, the maximum channel utilization is 0.5, 0.75 and 0.84 respectively (shown as vertical solid lines in Figure 4.2.2.1).

The results for system with frame size 128, wavelength channels two, four, eight, and session throughput requirement of one, three, twelve slots per frame are shown in Figure 4.2.2.1. Notice that the system approaches the absolute bounds mentioned.



Figure 4.2.2.1 Mathematical Bounds for *M/M/1/Q/L+1* System

Mathematical Bounds for M/M/1/Q/CL

The number of servers *m* equals to $W \lfloor T/L \rfloor$ as mentioned earlier. The arrival rate used in calculation is $\lambda = \rho W$ as obtained from Equation (4.1.0.1) and by setting $1/\mu_0$ to one as in simulations (*W* is the number of wavelength channels).

Same as that in M/M/1/B/L system, the maximum load the system can handle without running into infinite queue and delay is $L^{*}[T/L]/T$. So for L equals to one, three and

twelve, the maximum channel utilization is 1.0, 0.98 and 0.94 respectively (shown as vertical solid lines in Figure 4.2.2.2).

The results for system with frame size 128, wavelength channels two, four, eight, and session throughput requirement of one, three, twelve slots per frame are shown in Figure 4.2.2.2. Notice that the system approaches the absolute bounds mentioned.



Figure 4.2.2.2 Mathematical Bounds for *M/M/1/Q/CL* System

Mathematical Bounds for M/M/1/Q/RL

The number of servers *m* equals to $\lfloor WT/L \rfloor$ as mentioned earlier. The arrival rate used in calculation is $\lambda = \rho W$ as obtained from Equation (4.1.0.1) and by setting $1/\mu_0$ to one as in simulations (*W* is the number of wavelength channels).

Same as that in M/M/1/B/R system, the maximum load the system can handle without running into infinite queue and delay is L* WT/L WT. In our system, the frame size is 128. For L of 1, the maximum channel utilization is 1.0; for L of 3, the maximum channel utilization is 0.996, 0.996, 0.999 for W of 2, 4 and 8 respectively; for L of 12, the maximum channel utilization is 0.984, 0.984, 0.996 for W of 2, 4 and 8 respectively.



Figure 4.2.2.3 Mathematical Bounds for M/M/1/Q/RL System

The results for systems with frame size 128, two, four, eight wavelength channels, and session throughput requirement of one, three, twelve slots per frame are shown in Figure 4.2.2.3. Notice that the system approaches the absolute bounds mentioned.

4.3 Two Class, Uniform Traffic System

In this section we derive mathematical bounds for two class, uniform traffic, blocking systems by Markov Chain approach. We will also look at mathematical bounds for two class, uniform traffic, blocking system.

4.3.1 Blocking System

Again we use Markov Chain to analyze the system while ignoring transmitter and receiver conflicts. Let (i_1, i_2) be the state variables for the number of type L_1 and L_2 sessions in service respectively. Let λ_1 and λ_2 be the corresponding arrival rate, and $1/\mu$ the average session holding time, identical for both type of sessions. Furthermore, assume $L_1 < L_2$, and define $i_{1m} = \lfloor WT/L_1 \rfloor$ and $i_{2m} = \lfloor WT/L_2 \rfloor$. We have a two dimensional Markov Chain bounded by $i_1L_1 + i_2L_2 \le WT$ as shown in Figure 4.3.1.1.



Figure 4.3.1.1 Markov Model for *M/M/2/B/** System

Instead of obtaining an overall blocking probability for the system, we use a weighted blocking probability. The reason is that since type L_1 and L_2 sessions have different throughput requirements, blocking one type of session is not the same as blocking the other as far as the overall impact on the system throughput is concerned. The weighted blocking probability takes that into consideration. We will show shortly that the weighted blocking probability derived here is the lower bound for the system, thus can be used as an
yard stick to measure simulation results.

$$\lambda_1 \cdot P_{i_1, i_2} = (i_1 + 1) \mu \cdot P_{i_1 + 1, i_2}$$
 and $\lambda_2 \cdot P_{i_1, i_2} = (i_2 + 1) \mu \cdot P_{i_1, i_2 + 1}$

Solving above balance equations, we can obtain the channel utilization CU, the blocking probability Pb_1 and Pb_2 for type L_1 and L_2 sessions, the weighted blocking probability Pb of the system, and the steady state probability $P_{i1, i2}$ of system in state (i_1, i_2) , assuming $L_1 < L_2$.

$$P_{i1,i2} = \frac{\rho_1^{i1}}{i_1!} \cdot \frac{\rho_2^{i2}}{i_2!} \cdot P_{0,0}$$
(4.3.1.1)

$$CU = \sum_{i_2=0}^{\lfloor WT/L_2 \rfloor} \sum_{i_1=0}^{\lfloor (WT-i_2L_2)/L_1 \rfloor} \frac{i_1L_1 + i_2L_2}{WT} \cdot P_{i1, i2}$$
(4.3.1.2)

$$Pb = \{\alpha L_1 \cdot Pb_1 + (1 - \alpha)L_2 \cdot Pb_2\} / \{\alpha L_1 + (1 - \alpha)L_2\}$$
(4.3.1.3)

$$Pb_{1} = \sum_{i_{2}=0}^{\lfloor WT/L_{2} \rfloor} P_{\lfloor (WT-i_{2}L_{2})/L_{1} \rfloor, i2}$$
(4.3.1.4)

$$Pb_{2} = \sum_{i_{1}=0}^{\lfloor WT/L_{1} \rfloor} P_{i1, \lfloor (WT-i_{1}L_{1})/L_{2} \rfloor}$$
(4.3.1.5)

where

$$P_{0,0} = \left\{ \sum_{i_2=0}^{\lfloor WT/L_2 \rfloor} \frac{\rho_2^{i_2}}{i_2!} \sum_{i_1=0}^{\lfloor (WT-i_2L_2)/L_1 \rfloor} \frac{\rho_1^{i_1}}{i_1!} \right\}^{-1}$$
(4.3.1.6)

$$\rho_1 = \frac{\lambda_1}{\mu} = \frac{\alpha W T \rho}{\alpha L_1 + (1 - \alpha) L_2} \qquad and \qquad \rho_2 = \frac{\lambda_2}{\mu} = \frac{(1 - \alpha) W T \rho}{\alpha L_1 + (1 - \alpha) L_2} \qquad (4.3.1.7)$$

The blocking probability Pb_1 and Pb_2 for type L_1 and L_2 sessions are not the lower bounds for the system. They are what one would get when using a "fair" scheduling algorithm which gives equal access of system resources to both type L_1 and L_2 sessions. (By contrast, an "unfair" scheduling algorithm prefers one type of the session over the other. In our system, most of the scheduling algorithms are inherently unfair since sessions with larger throughput requirements are more likely to be rejected not only because they ask more slots at once but also because they are more likely to encounter transmitter and receiver conflicts).

However the weighted blocking probability Pb is the lower bound for the system, it does not matter if the system uses fair or unfair scheduling algorithms. (i.e. Pb = Pb(fair) = Pb(unfair), where Pb(fair) and Pb(unfair) are weighted blocking probabilities for fair and unfair systems respectively). Only when taking transmitter and receiver conflicts into consideration, the weighted blocking probability of the system will be lower than that obtained here. Thus Pb from Equation (4.3.1.3) is indeed the lower bound for the system.

Assuming we are using an unfair scheduling algorithm that increases the blocking probability for type L1 sessions from $Pb_1(fair)$ by δ (i.e. $Pb_1(unfair) = Pb_1(fair) + \delta$), so for every type L1 sessions, an additional δL_1 slots are freed up for type L2 sessions' use. Since for every type L1 session request there is α^{-1} -1 type L2 sessions, the blocking probability for type L2 sessions decreases from $Pb_2(fair)$ by $\varepsilon = (\delta L_1/L_2)/(\alpha^{-1}-1)$, that is

 $\begin{aligned} Pb_2(unfair) &= Pb_2(fair) - \varepsilon. \text{ From Equation (4.3.1.3), } Pb(unfair) &= \{\alpha L_l Pb_l(unfair) + (1-\alpha)L_2 Pb_2(unfair)\} / \{\alpha L_l + (1-\alpha)L_2\} = Pb(fair) = Pb. \end{aligned}$



Figure 4.3.1.2 Mathematical Analysis for M/M/2/B/* System

In Figure 4.3.1.2, we showed the channel utilization, weighted blocking probability, and blocking probability for type L_I and L_2 sessions for a system of $L_I=3$, $L_I=12$, W=8, T=128. Notice that CU and Pb vary little as a function of α . The program used is included in Appendix A.3.1.

OPtimized Network Engineering Tools (OPNET) is a comprehensive software program capable of simulating large communications networks with detailed protocol modeling and performance analysis[6]. Its features include: graphical specification of models; a dynamic, event-scheduled Simulation Kernel; integrated data analysis tools; and hierarchical, object-based modeling. We will use OPNET v2.3/M(c) on SUN SPARC stations to model our systems.

The highest "domain" in OPNET's hierarchical modeling structure is the network model. Nodes and links can be placed directly in a network or within subnetworks, which can be treated as single objects in the network model. A node is defined by connecting various module types with packet streams and statistics wires. The connection between modules allow for guided packet and status information exchange between modules. Each module placed in a node serves a unique purpose, such as generating packets, queueing packets, processing packets, or transmitting and receiving packets. At the core of most OPNET simulations are user-defined process models in addition to that provided by OPNET. Process models can represent the logic of communications hardware, network protocol, distributed algorithms, or high-level server-client processes. OPNET allows construction of process models by graphical representation of extended finite state machine.

In our system, since we ignore propagation delay of the packet, we can collapse physical nodes (terminals) into one logical node. So we've designed our network consisting of only one node, shown as in Figure 5.0.1. This node is made up of three modules, a source, the *scheduler* and the *release* process modules. The source, *src*, uses the ideal source generator provided by OPNET, it generates session requests according to a Poisson process with a specified rate. The *scheduler* model processes a session request by making appropriate resource assignments. It differs slightly for the blocking system and the queueing system. The *release* module erases the session after it's finished, it is also different for blocking and queueing system.

In Appendix B.1, we presented the network model used in OPNET provided report form, and in Appendix B.2 is the OPNET report form of the node model used.



Figure 5.0.1 Node Module

5.1 Blocking System

The *release* module is shown in Figure 5.1.1. Upon entering the module, it erases the packet that represents session request and clear the associate variables(i.e. release the system resource used by the session).



Figure 5.1.1 Release Process Module

The *scheduler* module for blocking system is shown in Figure 5.1.2. At the beginning of the simulation, the system will first enter the *init* state, where variables are initialized. Environmental variables such as the number of wavelength channels in the system W, the frame size T, number of nodes in the system N, the session throughput requirement in terms of number of slots per frame L are registered. At the end of the init state, if there is a session request arrival, it transits into the *pk_prepare* state, else it goes into the *idle* state. In the *pk_prepare* state, the session request gets assigned the source and destination address and the session holding time before getting sent into the schedule state. Once in the schedule state, based on the source and destination address and the available system resource, the session request is either rejected or accepted. In the former case, the session request is registered as "blocked" and deleted. In the later case, the scheduler send a delayed interrupt to the release module. So after a delay period equal to the session hold time, the release module will be notified and delete the expired session. Once the scheduler has finished with a session request, it goes back to the *idle* state, where it waits for the next arrival. When the simulation ends, the function record_stats records system statistics such as channel utilization, blocking probability before exiting.



Figure 5.1.2 Scheduler Process Module for Blocking System

5.2 Queueing System

The queueing system behaves very much like the blocking system, the only difference is that it keeps a queue of session requests that cannot be immediately satisfied.

The *release* process model for the queueing system is almost the same as that shown in Figure 5.1.1. In addition to erasing the expired session and freeing associated system resource, it also sends an interrupt to the *scheduler* module to notify it of the additional resource available. Upon receiving such an interrupt, if the *scheduler* finds itself with an non-empty queue, it will go into state *schedule*, and process the session requests hold in the queue.

The scheduler process model for the queueing system is shown in Figure 5.2.1. As in the blocking system, the module will enter from the *idle* state into the $pk_prepare$ state when a new session request comes in, and proceed to the schedule state to process the request. Additionally when an interrupt from the *release* module comes in signaling additional resource available, and the queue is not empty at the time, the module enters from the *idle* state to the schedule state to process session requests hold in the queue. Everything else is identical to that in the blocking system.

There is a slight difference between single class and two class systems. In single class system a single sub-queue is maintained. While in two class system two sub-queues are maintained, one for each type of the sessions. Upon receiving the interrupt from the *release* module signaling the release of a session, the system will try to fill the "vacancy" with the same type of session from the sub-queue as the one just released. The different type of session in sub-queue has lower priority to be filled.



Figure 5.2.1 Scheduler Process Module for Queueing System

5.3 Scheduling Algorithms

In this section we describe the three scheduling algorithms used. And point out the differences between the single class and the two class systems.

5.3.1 Single Class System

Contiguous L+1 Assignment Algorithm

Since all sessions require L slots for data and one additional slot for overhead tuning, we can regard the system as consisting of W rows and $\lfloor T/(L+1) \rfloor$ columns of cells. Each cell consists of contiguous L+1 slots. The scheduling algorithm searches through each column and row to find a free cell without Column Conflict and assign it to the session. The following is the general description of the algorithm.

- for each column i (0 ≤ i < LT / (L+1)) do
 determine if there is a Column Conflict and find a free cell in the column.
 - if without Column Conflict and a free cell is found, mark cell found, end the search; else, continue to next column.
- if cell found at the end of the search, assign the cell to the session; else, reject the session request and mark it as blocked.

Contiguous L Assignment Algorithm

Similar to the L+1 Assignment, this system can be regarded as consisting of W rows and $\lfloor T/L \rfloor$ columns of cells. Each cell consists of contiguous L slots. However a free cell is not necessarily usable due to Column, Pre-column and Post-column Conflict as discussed in Section 3.2. From now on we call a free cell without Column Conflict an available cell. An available cell encountering Pre-column or Post-column Conflict and unable to resolve it cannot be used either, and we call it a rejected cell. An non-rejected cell is a usable cell ready for assignment to the session. We will assign priority to each usable cell depending on the number of conflicts it resolved. For example a usable cell that encounters none of the Pre-column and Post-column Conflict has zero priority. A usable cell that encounters and resolves a Pre-column Conflict due to a transmitter has priority one. A usable cell that encounters and resolves all of the Pre-column Conflicts due to a transmitter and a receiver, and Post-column Conflicts due to a transmitter and a receiver has priority four. The algorithm will search through all cells in the system and replace the keeper cell with higher priority usable cell. This priority system favors maximum conflicts resolution, and hopefully helps us lowering session blocking due to unresolved conflicts. The search will end when the keeper cell reaches the highest priority of four or when all cells are searched. At the end of the search, the session will be assigned to the keeper cell if it exists, or marked as rejected. The following is the general description of the algorithm.

- for each column i $(0 \le i < [T/L])$ do
 - determine if there is a Column Conflict and find all free cells in the column.
 - if there is Column Conflict or no free cell is found, continue to next column.
 - for all available cells in the column, determine their usability by checking Pre-column and Post-column Conflicts and assign priority to usable ones, replace the keeper cell with higher priority usable cell.
 - if the keeper cell's priority reaches four (highest priority), end the search; else, continue to next column.
- if keeper cell exists at the end of the search, assign it to the session; else, reject the session request and mark it as blocked.

Random L Assignment Algorithm

This algorithm allows for MW-RS resource allocation as discussed in Section 3.6. Given a request of L slots per sessions, the final assignment may consist of slots spread across all wavelength channels. The algorithm described below is not limited to single or two class traffic. It works for any value of L.

A block (l, w, s, o) is made of slots with identical wavelength channel w and consecutive slot numbers starting at position s in the frame. The *size* of a block l is the number of slots contained in it. The overhead indicator of a block o can be represented by a binary number xy, where x and y indicate if the block encounters *Pre-column* and *Post-column Conflicts* respectively. It takes on the value of one for unresolved conflict and zero otherwise. Notice that o with decimal value 2 represents binary 10. The actual overhead of the block has the decimal value of x+y which represents the number of slots needed for tuning. So a block with o of decimal 2 needs 1 slot for overhead due to Pre-column Conflict. The *available size* of a block is the size of the block (l) minus the actual overhead of the block. The available slots of a block are all slots in the block except the first one if x is one and the last one if y is one.

What the algorithm does first is to find all slots that could be used for the session and sort them into a list of blocks in order of decreasing size. From this list of blocks, the algorithm will allocate slots to the session using criteria of minimizing waste and using the smallest block possible. The concept of waste is as following. The number of slots provided by a block is the lesser of the available size of the block and the number of slots needed. When the size of a block is greater than the number of slots provided by it, the difference between the two is called the waste associated with the block. For example, three slots are needed while the block has size seven. So the number of slots provided by the block is three while the waste is four. The occurrence of waste indicates fragmentation of the block, which we want to avoid. Another way of reducing fragmentation is to reserve larger blocks for sessions with larger L, which means using the smallest block possible when making assignment.

Since we are using MW-RS approach, multiple blocks can be assigned to a session. There are also possibly many ways to assign blocks to a session. What we are going to do is to work down the list of blocks until we find a "possible assignment" (pa) which may contain one or more blocks. We assign priority to the pa, and may replace "keeper assignment" (ka) with the pa if the pa has higher priority. We will continue this process until the end of the list, and assign ka to the session if one exists. The criteria for updating pa is to compare successive pa's waste, update only when new pa has less waste. Since a pa may contain more than one block, the waste of a pa is the cumulated waste of all blocks in it.

The process of finding a pa is as following. As we moved down the list, examine the block against the ones already assigned to the pa to determine if any slots have identical position in the frame. If that is the case, discard the block due to *Column Conflict* at that position. We call this "*usability test*". Next step is to determine if the block is bigger than needed. If the available size of the block is smaller than the number of slots needed, we

add the block to the pa and update the number of slots needed. Repeat the process as we move down the list until enough blocks are assigned to the pa to provide a total of L slots. However if the available size of the block is bigger than the number of slots needed, we mark this block as "temporary assignment" (ta) since we don't want to break up a big block. We move to the next block which has a smaller size and so on until we get to the smallest block possible and assign it to the pa. The criteria for updating ta is to compare successive ta's waste and overhead, update only when the newer one has less waste and no bigger overhead. We call this "ta-update process". The following is the general description of the algorithm.

- step 1:
 - find all free slots without Column Conflict and express them in terms of blocks.
 - discard all blocks whose available size is less than one.
 - sort remaining blocks into a list first by decreasing size and second by increasing overhead.
- step 2:
 - fetch the next available block in the list, repeat step 2 if failing "usability test".
 - if the block is a ta, repeat step 2 until completion of "ta-update process"; else, assign the block to pa.
 - if the pa is complete, replace ka with it if it has less waste; else, repeat step 2.
- step 3:
 - if ka exists when reaching end of the list, assign these slots to the session by rule 1; else, reject the session request and mark it as blocked.
- ♦ rule 1:
 - for all blocks in the ka except the last one, assign their available slots to the session.
 - the number of slots needed (ns) is L minus the number of slots provided thus far.
 - for the last block in the ka, we only need ns slots starting from position 1 if overhead of the block is 0; position 2 if overhead is 1 and 3; position "second from the last" and counting backwards if overhead is 2.

We will use an example to illustrate step 2. Assume we have following list of blocks: $\{(10, 1, 1, 0) (8, 2, 8, 0) (7, 4, 13, 0) (6, 3, 13, 0) (6, 1, 20, 1) (6, 2, 27, 2) (5, 1, 40, 0) (3, 4, 1, 0)\}$ and we need L equals to 16 slots.

- 1) fetch block 1 (10,1,1,0), pass "usability test".
- 2) block 1 is a non-ta since available size 10 is less than needed 16 slots, assign block 1

- 2) block 1 is a non-ta since available size 10 is less than needed 16 slots, assign block 1 to pa 1, waste 0, number of slots needed is now 6.
 3) fetch block 2 (8,2,8,1), fail "usability test" since *Column Conflict* at position 8, 9 and 10.
 4) fetch block 3 (7,4,13,0), pass "usability test", make it ta since available size 7 is greater than needed 6, waste 1.
 5) fetch block 4 (6,3,13,0), pass "usability test", waste 0, make it the new ta and the end of "ta-update process", assign block 4 to pa1, number of slots needed is now 0.
 6) pa1 complete with block 1 and 4 with a total waste of 0, assign pa1 to ka.
- 7) fetch block 5 (6,1,20,1), pass "usability test".
- 8) block 5 is non-ta, waste 1, assign to pa2, number of slots needed 11.
- 9) fetch block 6 (6,2,27,2), pass "usability test".
 10) block 6 is non-ta, waste 1, assign to pa2, number of slots needed 6.
- 11) fetch block 7 (5,1,40,0), pass "usability test".
- 12) block 7 is non-ta, waste 0, assign to pa2, number of slots needed 1.
 13) fetch block 8 (3,4,1,0), pass "usability test".
 14) block 8 is ta, waste 2.

- 15) end of list, assign block 8 to pa 2.
- 16) pa 2 complete with block 5, 6, 7 and 8 with a total waste of 4, ka=pa1.
 17) assign available slots in block 1 and 4 to the session.

5.3.2 Two Class System

Contiguous L+1(L) Assignment Algorithm

To ensure fair distribution of system resource among type L_1 and L_2 sessions, we partition the frame of size T into two independent portions of size T_1 and T_2 serving type L_1 and L_2 sessions respectively. The system is in essence two independent single class uniform traffic subsystems. We will use *Contiguous* L+1(L) Assignment Algorithm described in previous section for each of the two subsystems.

Let α be the percentage of the total sessions that is type L_1 sessions. A system of W wavelength channels and size T_1 frame can support on average $WT_1/(L_1+b)$ number of type L_1 sessions. A system of size T_2 frame can support on average $WT_2/(L_2+b)$ number of type L_2 sessions. In both cases b takes on the value of one for L+1 Assignment and zero for L Assignment. We have,

$$\{WT_1/(L_1+b)\}/\{WT_2/(L_2+b)\} = \alpha/(1-\alpha)$$
(5.3.2.1)

which gives

$$T_2/(L_2+b) = T/\{(L_2+b) + (L_1+b)\alpha/(1-\alpha)\}$$
(5.3.2.2)

The subsystem for L_2 sessions can be regarded as c_2 columns of cells, where c_2 equals to $\lfloor T_2/(L_2+b) \rfloor$ as specified in Equation (5.3.2.2). The subsystem for L_1 sessions can be regarded as c_1 columns of cells, where c_1 equals to $\lfloor (T-(L_2+b)c_2)/(L_2+b) \rfloor$. The following is the general description of the algorithm.

- ♦ determine the number of cells c1, c2 for subsystem 1 and 2 respectively.
- if type L₁ sessions, using Contiguous L+1(L) Assignment Algorithm (Section 5.3.1) to schedule the session in subsystem 1.
- if type L_2 sessions, using Contiguous L+1(L) Assignment Algorithm (Section 5.3.1) to schedule the session in subsystem 2.

Random LAssignment Algorithm

The Contiguous L+l(L) Assignment Algorithms outlined above are fair for both type L_1 and L_2 sessions, however an accurate estimate of the parameter α is needed. By using Random L Assignment Algorithm no estimation of α is needed. For blocking system, the Random L Assignment Algorithm outlined in the previous section can be used exactly as it is. For queueing system, notice that two separate queues for L_1 and L_2 sessions will be kept. So when a session request gets queued, it gets sent to the queue for that particular session type. Everything else is the same as described in the previous section.

In this chapter we present the simulation results for single/two class uniform traffic systems and single class client/server systems, and compare them to mathematical approximations/bounds derived in Chapter 4.

6.1 Single Class, Uniform Traffic System

6.1.1 Blocking System

The simulation results for *M/M/1/B/L+1*, *M/M/1/B/CL and M/M/1/RL* systems are shown in Figure 6.1.1.1 through Figure 6.1.1.3. The simulations are run for eight and forty users; two, four and eight wavelength channels; and throughput requirement of one, three, and twelve slots per frame. The solid lines in the graphs are calculated results based on the mathematical approximations/bounds obtained in Section 4.2.1. The dotted lines are simulation results.

As far as achieving maximum channel utilization is concerned, two factors come in to play, the number of wavelength channels W and the ratio of the number of nodes to the number of wavelength channels represented by parameter γ .

When γ is much greater than one, adding more wavelength channels to the system yields more efficiency as in the case of the forty user & eight wavelength system, the forty user & four wavelength system, and the forty user & two wavelength system. The reason is that when γ is large, session rejection due to transmitter and receiver conflicts is negligible. Therefore adding more wavelength channels translates directly into more system resources for the users.

When is γ close to one, systems with higher γ yields higher efficiency as in the case of the eight user & four wavelength system ($\gamma = 2$), and the eight user & eight wavelength system ($\gamma = 1$). The reason is that smaller γ indicates more session rejection due to transmitter or receiver conflicts and therefore less channel utilization rate. When γ is less than one, the transmitter and receiver become the bottleneck of the system, adding more wavelength channels will not give more resources to each of the users.

We can also see that systems with γ larger than two have relatively the same channel utilization rate regardless the number of wavelength channels. Only when γ is significantly small (less than two) does the system have a much lower channel utilization rate.

The figures also showed that systems with larger L reach the maximum channel utilization more slowly (i.e. has a larger 'knee').





For M/M/1/B/L+1 systems, simulation results approach the derived mathematical approximations rather closely. For systems with relatively large number of users ($\gamma > 1$), the channel utilization approaches that of the absolute maximum derived in Section 4.2.1 (0.5, 0.75 and 0.84 for L equals to 1, 3 and 12 respectively).





For M/M/1/B/CL system, the same two factors (number of wavelength channels W and the ratio γ of the number of nodes to wavelength channels) influence the system performance. It's clear that when the number of users is relatively large ($\gamma > 1$), the channel utilization approaches the mathematical bounds as well as the absolute maximum derived in Section 4.2.1 (1.0, 0.98 and 0.94 for L equals to 1, 3 and 12 respectively). Notice that the systems with L of 12 approaches the maximum much more slowly.





For M/M/1/B/RL system, the same two factors (number of wavelength channels W and the ratio γ of the number of nodes to wavelength channels) influence the system performance. It's clear that when the number of users is relatively large ($\gamma > 1$), the channel utilization approaches the mathematical bounds as well as the absolute maximum derived in Section 4.2.1 (1.0, 0.999 and 0.996 for L equals to 1, 3 and 12 respectively). Notice that the systems with L of 12 approaches the maximum much more slowly.

We want to compare the efficiency of the three algorithms for systems represented by large and small γ .

For systems with relatively large number of nodes (e.g. N = 40, $W = 8 : \gamma = 5$), session rejection due to transmitter or receiver conflicts is relatively rare. Therefore *Contiguous* L+1 Assignment Algorithm performs the worst due to the added tuning overhead of one slot. The smaller the L, the more of the overhead there is, and the lower the channel utilization it achieves. Random L Assignment Algorithm performs the best while Contiguous L Assignment Algorithm trails behind. This is expected since Contiguous L Assignment Algorithm wastes slots at the end of the frame when L doesn't divide T evenly.

For systems with relatively small number of nodes (e.g. N = 8, W = 8: $\gamma = 1$), Contiguous L+1 Assignment Algorithm performs much better than the other two algorithms especially for systems with large number of L. The reason is two fold. First for Contiguous L+1 Assignment Algorithm the percentage of tuning overhead is 1/L, smaller for larger L. Which means higher channel utilization for larger L. Secondly for the Contiguous and Random L Assignment Algorithms, whenever a group of L free slots are rejected due to Column, Pre-column or Post-column Conflicts, these L slots are "wasted". The larger the L, the higher the chance of these slots being rejected, and also more slots wasted when rejected. Which again means worse channel utilization for the Contiguous and Random L Assignment Algorithms for systems with larger L. From the figure, it is clear that Contiguous L+1 Assignment Algorithm performs the best for L equals to three and twelve, and the worst for L equals to one where the tuning overhead is 100%. When comparing Contiguous and Random L Assignment Algorithms, notice that for the system shown Contiguous L Assignment Algorithm performs better than Random L Assignment Algorithm for smaller L (L = 1 & 3) while the opposite is true for larger L (L = 12). Again this can be attributed to the trade-offs between two factors. First, the random algorithm fragments the system resources, which leads to more rejections due to transmitter and receiver conflicts and therefore lower channel utilization. On the other hand, it is harder to find L contiguous slots than L random slots. The larger the L, the harder it is to find L contiguous slots even if the system has the resources available. Which means lower channel utilization for the contiguous algorithms. When L is large enough, the second effect dominates and Random LAssignment Algorithm outperforms Contiguous LAssignment Algorithm (L=12).

When comparing the channel utilization (or blocking probability) achieved by systems with small γ (e.g. N = 8, $W = 8 : \gamma = 1$) to that with large γ (e.g. N = 40, $W = 8 : \gamma = 5$), we see that systems using *Contiguous* L+1 *Assignment Algorithm* has the smallest difference while those using *Contiguous* or *Random L Assignment Algorithms* have similarly larger differences. The reason is that by adding one slot for tuning overhead, *Contiguous* L+1 *Assignment Algorithm* helps systems with small γ while degrades systems with larger γ , therefore narrows the difference between the two.

Hopefully the load of the system is proportional to the number of users in the system. So when the users are few, the load is low and the system operates at below the knee of the curve and we don't have to worry about achieving maximum channel utilization. Only when more users come into the system, the load increases and we want to operate at the higher "plateau" of the curve. In this situation we see that *Random L Assignment Algorithm* gives the best performance.

The OPNET reports of the process models used for *M/M/1/B/L+1*, *M/M/1/B/CL*, *M/M/ 1/B/RL* are presented in Appendix B.3.1. The termination method and seeds selection are also discussed.

6.1.2 Queueing System

The results of the simulations are presented in Figure 6.1.2.1 through Figure 6.1.2.3. The solid lines are mathematical bounds based on results in Section 4.2.2. The dotted lines are simulation results. Notice that the simulation results are very close to the calculated bounds for all three systems when the number of nodes in the system is relatively large $(\gamma>1)$. The reason is the same as that discussed in the previous section for the blocking system.



Figure 6.1.2.1 M/M/1/Q/L+1 System Results

For M/M/1/Q/L+1 system, when the number of users is relatively large ($\gamma > 1$), the channel utilization approaches the mathematical bounds as well as the absolute maximum derived in Section 4.2.2 (0.5, 0.75 and 0.84 for *L* equals to 1, 3 and 12 respectively). In case of γ equals to one (N=8 & W=8), the "knee of the curve" occurs at load around 0.4, 0.55 and 0.5 for *L* equals to 1, 3 and 12 respectively due to heavy transmitter and receiver conflicts. Refer to Figure 6.1.1.1, we see that M/M/1/B/L+1 systems start to reach channel utilization "plateau" (or non-zero blocking probability) at the same load values. Also systems with larger *L* have slower rising curves. So the results are consistent.



Figure 6.1.2.2 M/M/1/Q/CL System Results

For M/M/1/Q/CL system, when the number of users is relatively large ($\gamma > 1$), the channel utilization approaches the mathematical bounds as well as the absolute maximum derived in Section 4.2.2 (1.0, 0.98 and 0.94 for L equals to 1, 3 and 12 respectively). In case of γ equals to one (N=8 & W=8), the "knee of the curve" occurs at load around 0.55, 0.45 and 0.35 for L equals to 1, 3 and 12 respectively due to heavy transmitter and receiver conflicts. Refer to Figure 6.1.1.1, we see that M/M/1/B/L+1 systems start to reach channel utilization "plateau" (or non-zero blocking probability) at the same load values. Also systems with larger L have slower rising curves. So the results are consistent.



Figure 6.1.2.3 M/M/1/Q/RL System Results

For M/M/1/Q/RL system, when the number of users is relatively large ($\gamma > 1$), the channel utilization approaches the mathematical bounds as well as the absolute maximum derived in Section 4.2.2 (1.0, 0.999 and 0.996 for L equals to 1, 3 and 12 respectively). In case of γ equals to one (N=8 & W=8), the "knee of the curve" occurs at load around 0.45, 0.5 and 0.5 for L equals to 1, 3 and 12 respectively due to heavy transmitter and receiver conflicts. Refer to Figure 6.1.1.1, we see that M/M/1/B/L+1 systems start to reach channel utilization "plateau" (or non-zero blocking probability) at the same load values. Also systems with larger L have slower rising curves. So the results are consistent.

Again we want to compare the efficiency of the three scheduling algorithms for systems with large number of users (e.g. N = 40, $W = 8 : \gamma = 5$) and small number of users (e.g. N = 8, $W = 8 : \gamma = 1$). As in the single class uniform traffic blocking systems *Random L Assignment Algorithm* for the queueing systems also gives the best performance for systems with $\gamma > 1$. For systems with smaller γ *Contiguous L+1 Assignment Algorithm* seems to give the best performance when L is large for the same reason as discussed in the previous section. When comparing the performance (e.g. the load value at the "knee of the curve") of systems with small γ (e.g. N = 8, $W = 8 : \gamma = 1$) to that with large γ (e.g. N = 40, $W = 8 : \gamma = 5$), we see that systems using *Contiguous L+1 Assignment Algorithm* has the smallest difference while those using *Contiguous* or *Random L Assignment Algorithms* have similarly larger differences. The reason is again the same as that for the blocking system. We can therefore conclude that M/M/1/B/* and M/M/1/Q/* systems have similar performance characteristics. In both systems *Random L Assignment Algorithm* gives the best performance when $\gamma > 1$ while *Contiguous L+1 Assignment Algorithm* seems to give the best performance when $\gamma > 1$ while *Contiguous L+1 Assignment Algorithm* seems to give the best performance when $\gamma > 1$ while *Contiguous L+1 Assignment Algorithm* seems to give the best performance when $\gamma > 1$ while *Contiguous L+1 Assignment Algorithm* seems to give the best performance when $\gamma > 1$ small and *L* is large.

In Appendix B.3.2 are the OPNET reports of the process models used for M/M/1/Q/C+1, M/M/1/Q/CL, M/M/1/Q/RL. The termination criteria and seeds selection are also discussed.

6.2 Two Class, Uniform Traffic System

6.2.1 Blocking System

The simulation results for M/M2/B/L+1, M/M/2/B/CL and M/M/2/B/RL systems are shown in Figure 6.2.1.1 and Figure 6.2.1.2. The simulations are run for forty users, eight wavelength channels, and throughput requirement of three (L_1) and twelve (L_2) slots per frame. The parameter α represents the percentage of session arrivals that is type L1. Given α , the percentage of system resources used by type L_1 sessions is β { $\beta = \alpha L_1/(\alpha L_1 + (1-\alpha)L_2)$ }.

In Figure 6.2.1.1, channel utilization and weighted blocking probability are plotted. The solid lines are mathematical bounds obtained in Section 4.3.1, the dotted lines are simulation results. It is clear that for systems with relatively large number of nodes (e.g. N = 40, $W = 8 : \gamma = 5$) Random L Assignment Algorithm gives the best performance and is close to the mathematical bound. Contiguous L+1 Assignment Algorithm has the poorest performance due to tuning overhead. For systems with small number of nodes (e.g. N = 8, $W = 8 : \gamma = 1$) Contiguous L+1 Assignment Algorithm performs the best while Contiguous L Assignment Algorithm the worst. Again the same reasons discussed in Section 6.1.1 apply here.

In Figure 6.2.1.2, blocking probabilities for type L1 and L2 sessions are plotted. The dotted lines are simulation results. The solid lines are derived in Section 4.3.1. They represent a system with large number of nodes and where both type of sessions have equal access to *all* system resources. We call them "fair" blocking probabilities.

For systems with relatively large number of users (e.g. N = 40, $W = 8 : \gamma = 5$), Random

L Assignment Algorithm yields better than "fair" blocking probability for type L1 sessions and worse than "fair" one for type L2 sessions. This is expected since sessions with larger value of L are more likely to be rejected than those with smaller value simply because it is harder to find more slots at once. For Contiguous L+1 and L Assignment Algorithms, the system is divided into two independent subsystems each serving either type L1 or L2 sessions. There are lots of waste in this arrangement since different type of sessions do not share resources with each other. The blocking probabilities for type L1 and L2 sessions are both worse than the "fair" ones most of the time. The exception is noted in the case of α equals to 0.8. The blocking probability for type L2 sessions cross over the "fair" one. The reason is that in the "fair" system a higher proportion of the system resources is used by type L1 sessions when α is large. Therefore the system becomes more fragmented and less favorable for type L2 sessions which have larger throughput requirement. In contiguous L and L+1 systems, a fixed amount of resources are reserved for type L2 sessions. So when the offered load on the system becomes high, type L2 sessions still have their share of the system resources, and thus have lower than the "fair" blocking probability.

For systems with small number of users (e.g. N = 8, $W = 8 : \gamma = 1$), blocking probabilities for type L1 and L2 sessions are higher than the "fair" ones for all three algorithms due to heavy transmitter and receiver conflicts. The blocking probability for type L1 sessions is the highest for Contiguous L Assignment Algorithm, and the lowest for Random L Assignment Algorithm. The reason is that Random L Assignment Algorithm allows sharing of all system resources and is the most efficient. For type L2 sessions, the blocking probability by Random L Assignment Algorithm starts lower than that by Contiguous L+1 and L Assignment Algorithms, but eventually becomes higher. The reason is the same as that explained earlier when comparing the blocking probability for type L2 sessions by Contiguous L+1 and L Assignment Algorithm to the "fair" one.

To summarize, for systems with γ greater than one, the *Random L Assignment Algorithm* achieves the highest channel utilization, also has close to the "fair" blocking probabilities for both type L1 and L2 sessions.

In Appendix B.4.1 is the OPNET process model report and the termination criteria.



Figure 6.2.1.1 M/M/2/B/* System (Channel Utilization & Weighted Blocking Probability)



Figure 6.2.1.2 *M/M/2/B/** System (Blocking Probability for $L_1 \& L_2$ Sessions)

6.2.2 Queueing System

Simulation results for M/M/2/Q/RL systems are shown in Figure 6.2.2.1. The weighted average queue size (Q) and time in queue (D) are obtained the same way as the weighted blocking probability for the two class blocking system. Let Q_1 and Q_2 be the average queue size for type L1 and L2 sessions, D_1 and D_2 the average time in queue. We have $Q = {\alpha L_1 Q_1 + (1-\alpha) L_2 Q_2}/{\alpha L_1 + (1-\alpha) L_2}$, and $D = {\alpha L_1 D_1 + (1-\alpha) L_2 D_2}/{\alpha L_1 + (1-\alpha) L_2}$.



Weighted Average Queue Size & Time in Queue



As in M/M/2/B/RL systems, sessions with larger L do not fare as well as those with smaller value. However for systems with relatively large number of users (40 users, 8 wavelength channels), the *Random L Assignment Algorithm* does perform well. A crude comparison is to M/M/1/Q/RL systems.

First look at forty users and eight wavelength channels system. For L equals to three and at load of 0.9, the average queue size and time in queue are 0.37 and 0.05 respectively for M/M/1/Q/RL system. In this system, the average queue size for type L1 (3) sessions at load of 0.9 varies from 0.04 to 0.17 (approximate reading from Figure 6.2.2.1) depending on the value of α , and the average time in queue varies from around 0.01 to 0.03. For L equals to twelve and at load of 0.9, the average queue size and time in queue are 4.0 and 0.56 respectively for M/M/1/Q/RL system. In this system, the average queue size for type L2 (12) sessions at load of 0.9 varies from 2 to 4 (approximate reading from Figure 6.2.2.1) depending on the value of α , and the average time in queue is approximately 0.7. So the performance characteristics of both type of classes in the M/M/2/Q/RL systems correspond closely to that in M/M/1/Q/RL systems.

Next look at eight users and eight wavelength channels system. It's clear that the average queue size and time in queue rise sharply at around load of 0.5. In M/M/1/Q/RL system the average queue size and time in queue also rise sharply at around load of 0.5 for both L equals to three and twelve cases.

We therefore conclude that this system compares closely to M/M/1/Q/RL system for each type of the sessions.

In Appendix B.4.2 OPNET process model report is shown, and termination criteria discussed.

6.3 Single Class, Client/Server System

6.3.1 Blocking System

Simulation results for M/M/1/B/RL uniform traffic and client/server systems are shown in Figure 6.3.1.1 and Figure 6.3.1.2. The solid lines in Figure 6.3.1.1 are the channel utilization and blocking probability for M/M/1/B/RL uniform traffic system. The dotted lines are results for M/M/1/B/RL client/server traffic systems. The simulations were run for systems with eight wavelength channels, one or three server nodes, and a total of forty nodes including the servers. The parameter *st* represents the percentage of sessions involving each server's transmitter or receiver. It takes on the value of 5% and 10% in our simulations. For our system of eight wavelength channels, when *st* approaches 1/8, the server's transmitter and receiver become the bottle neck of the system. It's clear when *st* is small (5%) the channel utilization and the blocking probability of the client/server systems are almost identical to the uniform systems. When *st* becomes larger (10%), the client/server systems performs less well than the uniform systems especially for larger number of servers (three servers system is much worse than one server system).

In Figure 6.3.1.2, blocking probabilities for server and uniform traffic are show. Given the same number of servers (1 or 3 here), the blocking probability for server is smaller for system with smaller *st*, and the reverse is true for uniform traffic's blocking probability. The reason is that smaller *st* means that the server's transmitter and receiver are less congested, thus decreasing the server traffic's blocking probability. It also means that higher

percentage of the traffic is uniform in nature, thus increasing uniform traffic's blocking probability. For system with the same *st* parameter (0.05 or 0.1 here), the blocking probability for server and uniform traffic is smaller for system with more servers. The reason is that more servers means that lower percentage of the traffic is uniform in nature, thus decreasing uniform traffic's blocking probability. For server traffic, more server means higher percentage of the server traffic (even though the server traffic handled by each server is constant), thus higher blocking probability.





Figure 6.3.1.1 M/M/1/B/RL Client-Server System (Channel Utilization & Blocking Probability)



Figure 6.3.1.2 M/M/1/B/RL Client-Server System (Blocking Probability for Server & Uniform Sessions)

6.3.2 Queueing System

Simulation results for M/M/1/Q/RL uniform traffic and client/server system are shown in Figure 6.3.2.1 through Figure 6.3.2.2. In Figure 6.3.2.1, average queue size (Q) and time in queue (D) for the overall system are shown. In Figure 6.3.2.2 and Figure 6.3.2.2, average queue size for server (Q_s) and uniform (Q_u) traffic, average time in queue for server (D_s) and uniform (D_u) traffic are show, respectively. Since a single queue is maintained for the server and the uniform traffic, $Q = Q_s + Q_u$ and $D = 2*st*N_sD_s + (1-2*st*N_s)D_u$,

where st is the percentage of total sessions that each server's transmitter or receiver carries and N_s is the number of servers. By comparing Figure 6.3.2.1 to Figure 6.3.1.1 and Figure 6.3.2.2 to Figure 6.3.1.2, we see that the queueing systems have similar relative performance characteristics as the blocking systems, and similar conclusion can be drawn.



In Appendix B.5.2 OPNET reports for the processor models are included.

Figure 6.3.2.1 *M/M/1/Q/RL* Client-Server System (for overall system)



Figure 6.3.2.2 M/M/1/Q/RL Client-Server System (Average Queue Size for Server & Uniform Traffic)



Figure 6.3.2.3 M/M/1/Q/RL Client-Server System (Average Time in Queue for Server & Uniform Traffic)

From our study of single, two class uniform traffic systems and single class client/server systems, we have shown that a *Random L Assignment Scheduling Algorithm* for level 0 AON subnetworks gives close to the optimal utilization of system resources when the ratio of users to wavelengths γ is large. In which case tuning overhead due to transmitter and receiver conflicts was shown to be a minimal issue. Fragmentation caused by using random slots does not seem to degrade the system performance either. We have also shown that for single class systems with large γ , the *Contiguous L Assignment Scheduling Algorithm* performs only slightly worse than the best *Random L Assignment Scheduling Algorithm*. The trade off here is much reduced algorithmic complexity.

For systems with small numbers of nodes ($\gamma < 2$), scheduling using Random L Assignment Algorithm does not always give the best performance compared to Contiguous L and L+1 Assignment Algorithms. In single class systems when the throughput requirement L is larger than one, Contiguous L+1 Assignment Algorithm gives the best performance compared to the other two algorithms. However we argue that since the number of nodes (users) are few in the system, the load will also be very light. So the system will be operating at below it's capacity, and Random L Assignment Scheduling Algorithm will also work well.

From the study of single class uniform traffic systems with γ greater than one, we showed that the *Random L Assignment Scheduling Algorithm* gives the best performance and it approaches the optimal system performance bound.

For two class uniform traffic systems, we showed that for blocking systems with $\gamma > 1$, the *Random L Assignment Scheduling Algorithm* gives the best performance and again approaches the optimal performance bound. For two class uniform traffic queueing systems, we ran the simulation only using *Random L Assignment Scheduling Algorithm* due to limited computational resources. The results compare well to the system capacity limits (*WT*). The average queue size and average time in queue for each type of session are close to the corresponding ones in the M/M/1/Q/RL case.

We also studied the single class client/server systems using the Random L Assignment Scheduling Algorithm. The results were compared to the corresponding single class uniform traffic systems. The parameter st, representing the percentage of the total traffic that each server's transmitter or receiver processes, indicates the degree to which the server's transmitter or receiver becomes a bottleneck. When st is small compared to 1/W (W the number of wavelengths), the system performs at close to uniform traffic level even for multiple servers. Therefore the results obtained by assuming uniform traffic pattern appear rather robust.

Our overall conclusion is that the *Random L Assignment Scheduling Algorithm* seems to be a very promising scheduling approach for the AON level 0 subnetwork for a wide variety of traffic requirements including, uniform traffic, multiclass traffic, and client/ server traffic.

A.1 Session Blocking Probability

For a blocking uniform traffic system with W wavelength channels, frame size T, and session throughput request of one slot per frame, we will first derive $P_{b|Ns}$ the probability that a new session (t,r) request will be rejected given the number of sessions in service is Ns. Assume all Ns sessions are distributed equally among the WT slots (or T columns). To simply the model, we will take into consideration of *Column Conflict*, but not *Pre-Column* and *Post Column Conflict*.

Let the Ns sessions in service be distributed among column (1, ..., T) according to $(N_1, ..., N_T)$, where N_i is the number of sessions using slots in column *i*, and $Ns = N_1 + ... + N_T$. Let N be the number of nodes in the system, the probability of blocking for the new session is:

$$P(b|Ns) = \sum_{\Sigma Ni = Ns; \ 1 \le Ni \le min(W, N)} P(b|N_1, ..., N_T, Ns) \bullet P(N_1, ..., N_T|Ns)$$
(a.1.1)

The summation is taken over $l \le N_i \le min(W, N)$. The upper bound is obtained by the fact that the number of slots in use in each columns can be no larger than the number of wavelength channels. In addition it can be no larger than the number of transmitters or receivers) in the system, which is N for a system of N nodes each with one transmitter(receiver). The lower bound is one since that N_i equals to zero means free slot available for the session, therefore it doesn't contribute to the blocking probability of the session.

To calculate the second term in equation (a.1.1), notice that under the assumption of equally distributed sessions among all columns, we have

$$P(N_1, ..., N_T | Ns) = \frac{Ns!}{N_1! ... N_T!} \left(\frac{1}{T}\right)^{N_1} ... \left(\frac{1}{T}\right)^{N_T} N(\cdot) = N(\cdot) \cdot \frac{Ns!}{T^{Ns}} \cdot \prod_{i=1}^T \frac{1}{N_i!}$$
(a.1.2)

Where N() is the normalization factor, can be obtained by summing $P_{NI, \dots, Nt|Ns}$ over all combinations of $(N1, \dots, Nt)$ to one.

$$P(N_1, ..., N_T | N_S) = \frac{1}{N(N_S, T, \min(W, N))} \cdot \prod_{i=1}^T \frac{1}{N_i!}$$
(a.1.3)

where

$$N(Ns, T, min(W, N)) = \sum_{\Sigma Ni = Ns; \ 0 \le Ni \le min(W, N)} \prod_{i=1}^{T} \frac{1}{N_i!}$$
(a.1.4)

To calculate the first term in equation (a.1.1), note a session is rejected by the system only if it is rejected by every column in the system. And since we assume only *Column Conflict*, $P_{bi|Ni}$ the probability that a session is rejected by column *i*, given there is N_i busy slots in the column, is independent of each other. Therefore,

$$P(b|N_1, ..., N_T, N_S) = \prod_{i=1}^{T} P(b_i|N_i)$$
(a.1.5)

Combine equation (a.1.3) through (a.1.5) into equation (a.1.1), we have

$$P(b|Ns) = \frac{M(Ns, T, min(W, N))}{N(Ns, T, min(W, N))}$$
(a.1.6)

where $N(N_s, T, min(W, N))$ is given in equation (a.1.4), and

$$M(Ns, T, \min(W, N)) = \sum_{\Sigma Ni = Ns; \ 1 \le Ni \le \min(W, N)} \prod_{i=1}^{T} \frac{P(b_i|N_i)}{N_i!}$$
(a.1.7)

The value of M(A,B,C) and N(A,B,C) can be obtained recursively. Here A donates the total number of sessions, B number of columns, C the upper bound for number of sessions in each column. Let $l = A \mod C$, the maximum number of columns that can take on the upper bound, then

$$M(A, B, C) = \sum_{i=0}^{l} C(B, i) \cdot \left(\frac{P(bi|C)}{C!}\right)^{i} \cdot M(A - iC, B - i, C - 1)$$
(a.1.8)

$$N(A, B, C) = \sum_{i=0}^{l} C(B, i) \cdot \left(\frac{1}{C!}\right)^{i} \cdot N(A - iC, B - i, C - 1)$$
(a.1.9)

where C(m,n) = m! / n!(m-n)! (m>=n), the initial and special conditions are:

$$M(A, B, C) = \begin{cases} 0 & A < B & or & A > C \cdot B \\ \{P(bi|1)/1!\}^B & A = B \\ \{P(bi|C)/C!\}^B & A = C \cdot B \\ P(bi|A)/A! & B = 1 & and & 1 \le A \le C \\ \{P(bi|1)/1!\}^{2B-A} \cdot \{P((bi|2)/2!)\}^{A-B} \cdot C(B, A-B) & C = 2 \end{cases}$$

$$N(A, B, C) = \begin{cases} 0 & ((B = 0 \text{ or } C = 0) \text{ and } (A \neq 0)) & \text{or } A > C \cdot B \\ 1 & (B = 0 \text{ or } C = 0) & \text{and } A = 0 \\ \{1/C!\}^B & A = C \cdot B \\ 1/A! & B = 1 \text{ and } 0 \le A \le C \\ C(B, A) & C = 1 \end{cases}$$

Now the only thing we need is $P_{bi|Ni}$ the probability that a session is blocked by column *i*. It is the same as one minus the probability that it will be accepted. For a session to be accepted, there must be free slots available, i.e $N_i \neq min(W,N)$. Given free slots available, the new session (t,r)'s transmitter and receiver cannot be in use by any of the sessions already in service associated with the busy slots in the column. Let \overline{t} and \overline{r} donate that node t's transmitter and node r's receiver are free, we have

$$1 - P(b_i | N_i) = P(N_i \neq \min(W, N) | N_i) \bullet P(\tilde{i} | N_i \neq (\min(W, N), N_i)) \bullet P(\tilde{r} | \tilde{i}, N_i \neq (\min(W, N), N_i)) = \delta(N_i - \min(W, N)) \bullet P(\tilde{i} | N_i) \bullet P(\tilde{r} | \tilde{i}, N_i)$$
(a.1.10)

where $\delta_n = \begin{pmatrix} 0 & n = 0 \\ 1 & n \neq 0 \end{pmatrix}$.

The derivation for $P_{\bar{t}|Ni}$ is straightforward, it is the probability that picking N_i transmitters out of total N transmitters resulted in node t's transmitter not picked.

$$P(\tilde{i}|Ni) = \frac{C(N-1,Ni)}{C(N,Ni)} = 1 - \frac{Ni}{N}$$
(a.1.11)

Let $L(N,N_i)$ be the number of choices of picking N_i sessions out of a system of Nnodes, where each node can transmit to any other node but itself. Let $L_I(N,N_i)$ be the number of choices of picking N_i sessions out of a system of N nodes, where node t is not transmitting at all, and all other node can transmit to any other but itself. Let $L_2(N,N_i-1)$ be the number of choices of picking N_i sessions out of a system of N nodes, where node t is transmitting to one of the possible N-1 nodes, say node r, and the rest of the nodes can transmit to each other but itself and node r. Then

$$L(N, Ni) = L1(N, Ni) + (N-1) \cdot L2(N, Ni-1)$$
(a.1.12)

The probability that node t is transmitting is $(N-1)L_2(N,N_i-1)/L(N,N_i)$ from the definition of L, L1, and L2. It is also the probability of picking N_i transmitters out of N with equal probability with node t's transmitter picked, therefore

$$\frac{(N-1)L2(N,Ni-1)}{L(N,Ni)} = \frac{C(N-1,Ni-1)}{C(N,Ni)} = \frac{Ni}{N}$$
(a.1.13)

The definition of $L_2(L, N_i-1)$ can be reiterated as the number of choices of picking N_i-1 sessions out of a system of N nodes, where a node, say t, is not transmitting and another node, say r, not receiving, and all other nodes can transmit to each others but itself. Then it's easy to see that

$$P(\bar{r}|\bar{t}, Ni) = \frac{L2(N, Ni)}{L1(N, Ni)}$$
(a.1.14)

Combine equation (a.1.12) through (a.1.14), we can solve for $P_{\overline{r}|\overline{t},Ni}$ in terms of L_2 , plug that and equation (a.1.11) into equation (a.1.10), we have:

$$1 - P(bi|Ni) = \delta(Ni - min(W, N)) \cdot \frac{Ni}{N(N-1)} \cdot \frac{L2(N, Ni)}{L2(N, Ni-1)}$$
(a.1.15)

To solve for L2, we need one more set of equation. Consider $L_2(N,N_i)$, the number of choices of picking N_i sessions out of a system of N nodes where a node, say t, is not transmitting and a different node, say r, is not receiving. It can be further broken into cases according to the status of node r's transmitter. In case of node r not transmitting, we can take this node out of the system of N nodes since neither of its transmitter or receiver is
busy. So the number of choices is the same as picking N_i sessions out of a system of N-1 nodes where node t is not transmitting, i.e $L_i(N-1,N_i)$. Now consider the case where node r is transmitting to either node t or one of the rest of the N-2 nodes. In the former case, we have node t not transmitting, node r not receiving, and a session (r,t). The rest of the N_i -1 sessions are happening among the other N-2 nodes, so the number of choices are $L(N-2,N_i-1)$. In the later case, we have node t neither transmitting nor receiving, and therefore can be discounted. Node r is transmitting to one of the N-2 nodes, but not receiving. By definition the number of choices of picking N_i session in such a system is $(N-2)L_2(N-1,N_i-1)$. Therefore

$$L2(N, Ni) = L1(N-1, Ni) + L(N-2, Ni-1) + (N-2) \cdot L2(N-1, Ni-1)$$
(a.1.16)

Combine equation (a.1.12), (a.1.13) and (a.1.16), and with initial condition of $L_2(*,0) = I$, we can solve L_2 recursively:

$$L2(N,Ni) = \frac{(N-1)(N-2)}{Ni} \cdot L2(N-1,Ni-1) + \frac{(N-2)(N-3)}{Ni-1} \cdot L2(N-2,Ni-2)$$
(a.1.17)

where

$$L2(N,0) = 1$$
 and $L2(N,1) = N^2 - 3N + 3$ (a.1.18)

A.2 Single Class, Uniform Traffic System

The C programs used to calculate mathematical approximation and bounds are included.

A.2.1 Blocking System

Mathematical Approximation for M/M/1/B/L+1 and M/M/1/B/CL

The following program is used to calculate the mathematical approximation/bounds as in Section 4.2.1 for M/M/1/B/L+1 system by setting parameter b to 1, and for M/M/1/B/L by setting b to 0. The added complexity in the program is due to resolving number overflow.

```
if (Nn<W) min1=Nn;
pbf();
for (Ns=0; Ns<=Nx; Ns++)
  {
if (Ns<T) { pa[Ns]=1; continue; }
if (Ns>=Nn*T || Ns==Nx) { pa[Ns]=0; continue; }
pb=M(Ns,T,min1); m1=N(Ns,T,min1); pb=pb/m1;
for (i=0; i<(mct+nct); i++) pb=pb/1e15;
cothcl=1 pb;
    pa[Ns]=1-pb;
 /*
        Getting ps[i]
                                                  .---- */
 for (p=0.1;p<1.6;p+=0.1)
   p1=p*W*T0/L; ps[0]=1; pb=0; m1=0; m=1; j=0;
for (i=1;i<=Nx;i++)
     {

ps[i]=ps[i-1]*p1*pa[i-1]/i;

m+=ps[i];

pb+=ps[i]*(1-pa[i]);

m1+=i*ps[i]*L/(T0*W);

if (m>1e30)

(a1ii=mm=0, a2ii=pt
        { s1[j]=m; m=0; s2[j]=pb; pb=0; s3[j]=m1; m1=0; ps[i]=ps[i]/1e30; j+=1; }
   for (i=0; i<j; i++)
     {
for (b=0; b<(j-i); b++)
{ s1[i]=s1[i]/1e30; s2[i]=s2[i]/1e30; s3[i]=s3[i]/1e30; }
m+=s1[i]; pb+=s2[i]; m1+=s3[i];
   pb=pb/m; m1=m1/m;
printf("%20.9f\t%20.9f\n",pb,m1);
}
 void pbf()
{
int i,k;
double m,l,n1,n2;
extern int Nn,min1;
extern double pbi[];
k=Nn*(Nn-1); n1=1; n2=L(Nn,1); pbi[0]=0; pbi[min1]=1;
for (i=1;i<min1;i++)
{ pbi[i]=1-i*n2/(n1*k); n1=n2; n2=L(Nn,i+1); }
}
double L(int N, int i)
 double R;
\begin{array}{l} \label{eq:starsess} \mbox{if} \ (i == 0) \ \{ \ R = 1.0; \ return(R); \ \} \\ \mbox{if} \ (i == 1) \ \{ \ R = (double)(N^*N-3^*N+3); \ return(R); \ \} \\ \ R = (N-1)^*(N-2)^*L(N-1,i-1)/i \ + \ (N-2)^*(N-3)^*L(N-2,i-2)/(i-1); \\ \ return(R); \end{array}
double comb(int x, int y)
 double R=1.0;
int i,j;
   =x-y;
j=x-y;
if (x<0 ll y<0) return(0);
if (j<0) return(0);
for (i=1;i<=j;++i) R=R*(y+i)/i;
return(R);
 double fact(int x)
 double R=1.0;
int i;
for (i=1<u>;i</u><=x;i++) R=R*i;
 return(R);
3
double N(int A, int B, int C)
int i,j,p,nct_keep,nct1_keep,nct1;
double R=0,k,l,m,n,n1;
extern int nct,nct_old;
```

nct=0; if (B==0) if (A==0) return(1); else return(0);

```
if (C==0) if (A==0) return(1); else return(0); if (A > B^2C) return(0); if (A == B^*C)
  {
i=1/fact(C); R=1.0;
for (i=1;i<=B;i++)
     \
R=R*l;
if (R<1e-15) { R=R*1e15; nct-=1; }
   return(R);
   }
if (B==1)
if (A>=0 && A<=C)
     1
R=1/fact(A);
if (R<1e-15) { R=R*1e15; nct-=1; }
return(R);
   else return(0);
if (C == 1)
  {
R=1.0;
for (i=1; i<=A; i++)
    {
R=R*(B-A+i)/i;
If (R>1e15) { R=R/1e15; nct+=1; }
   réturn(R);
  }
\label{eq:constraint} \begin{array}{l} \mbox{!=(double)A/(double)C; i=floor(i); } m=1/fact(C); \ p=A-B^*(C-1); \\ \mbox{if } (p<0) \ p=0; \end{array}
R=N(A-p*C,B-p,C-1); nct1=0; n=1;
for (i=1; i<=p; i++)
  1
m=n*m*(B-p+i)/i;
if (n<1e-15) { n=n*1e15; nct1-≈1; }
if (n>1e15) { n=n/1e15; nct+=1; }
}
nct+=nct1; R=R*n;
if (R<1e-15) { R=R*1e15; nct-=1; }
if (R>1e15) { R=R/1e15; nct+=1; }
nct_old=nct; p+=1;
for (I=p;i<=I;i++)
 else {
if (nct_old>nct)
            for (j=0; j<(nct_old-nct); j++) k=k/1e15;
R+=k;
          eise
           for (j=0; j<(nct-nct_old); j++) R≈R/1e15;
nct_old=nct; R+=k;
           }
        }
  3
nct=nct_old;
return(R);
}
double M(int A, int B, int C)
{
int i,j,p,mct_keep,mct1,mct1_keep;
double R=0,k,l,m,n;
extern int mct,mct_old;
extern double pbi[];
mct=0;
if (A<B) return(0);
if (A>C*B) return(0);
if (A==B)
  {
l=pbi[1];
R=1.0;
```

```
for (i=0; i<B; i++)
   R=R*I;
if (R<1e-15) { R=R*1e15; mct+=1; }
  réturn(R);
if (A==C*B)
 i=pbi[C]/fact(C);
R=1.0;
for (i=1;i<=B;i++)
   È=R*l;
   if (R<1e-15) { R=R*1e15; mct+=1; }
  réturn(R);
if (B==1)
if (A>=1 && A<=C)
   h=pbi[A]/fact(A);
if (R<1e-15) { R=R*1e15; mct+=1; }
return(R);
  else return(0);
if (C==1)
 {
R=1.0;
for (i≃1;i<=A;i++)
   {
R=R*pbi[1];
if (R<1e-15) { R=R*1e15; mct+=1; }
  réturn(R);
if (C==2)
 l=pbi[1]; m=pbi[2]/2; R=1.0;
for (j=1; j<=(A-B); j++)
   R=R*m*(2*B-A+j)/j;
if (R<1e-15) { R=R*1e15; mct+=1; }
  for (j=1; j<=(2*B-A); j++)
    k=R*I;
   if (R<1e-15) { R=R*1e15; mct+=1; }
  return(R);
l=(double)A/(double)C; l=floor(I); m=(double)(A-B)/(double)(C-1); m=floor(m);
if (l>m) l=m;
m=pbi[C]/fact(C); p=A-B*(C-1);
if (p<0) p=0;
R=M(A-p*C,B-p,C-1);
mct1=0; n=1;
for (i=1;i<=p;i++)
 {
n=n*m*(B-p+i)/i;
if (n<1e-15) { n=n*1e15; mct1=mct1+1; }
}
mct+=mct1; R=R*n;
if (R<1e-15) { R=R*1e15; mct+=1; }
mct_old=mct; p+=1;
for (i=p;i<=l;i++)
/
 if (k<1e-15) { k=k*1e15; mct+=1; }
if (k>1.0) { k=k/1e15; mct=1; }
if (mct_old==mct) R+=k;
 else {
if (mct_old>mct)
         for (j=0; j<(mct_old-mct); j++) R=R/1e15;
mct_old=mct; R+=k;
        eĺse
         for (j=0; j<(mct-mct_old); j++) k=k/1e15;
R+=k;
```

• - •



Mathematical Bounds for M/M/1/B/RL

```
#include <math.h>
void main()
{
    int i,j,k,b,W,T,L,m;
    double p,a,q,s,Pq,Nq,ss[20];
W=8; L=7; T=128;
p=(double)(W*T)/(double)L; m=floor(p);
for (p=0.1; p<1.6; p+=0.1)
    {
        a=p*W*T/L; q=1; s=1; jj=0;
        for (i=1; i<m; i++)
        {
        q=q*a/i; s+=q;
        if (s>1e30) { ss[j]=s; s=0; q=q/1e30; j+=1; }
    }
    for (k=0; k<(j-i); k++) ss[i]=ss[i]/1e30;
    s+=ss[i];
        q=q*a/m; Pq=q/(s+q); Nq=(1-Pq)*p;
    printf("%flt%flt%flt", Nq,Pq);
}</pre>
```

A.2.2 Queueing System

The following program is used to calculate mathematical bounds as in Section 4.2.2 for M/M/1/Q/L+1, M/M/1/Q/CL, and M/M/1/Q/RL systems. The added complexity is due to resolving number overflow.

A.3 Two Class, Uniform Traffic System

A.3.1 Blocking System

The following program is used to calculate mathematical bounds as in Section 4.3.1 for M/M/2/B/* systems. The added complexity is due to number overflow.

#define L 50 void main() Int W,T,L1,L2,i,j,k,m1,m2,il,ict,ict1,jct,jct1; double a,p,p1,p2,q,s0,s1,s2,s3,c0,c1,c2,c3,c4,cu,pb1,pb2,pb,pp; double ss2[L],ss3[L],cc2[L],cc3[L],cc4[L],pp1[L],pp2[L]; W=8; a=0.8; L1=3; L2=12; T=128; for (p=0.1; p<1.6; p+=0.1) $\begin{array}{l} p_1 = a^* p^* W^* T / (a^* L 1 + (1 - a)^* L 2); & p_2 = (1 - a)^* p_1 / a; \\ q = (double) (W^* T) / (double) L 2; & m_2 = floor(q); \\ q = (double) (W^* T) / (double) L 1; & m_1 = floor(q); \\ q = (double) (W^* T) / (double) L 1; & m_1 = floor(q); \\ q = (double) (W^* T) / (double) L q = (1 - a)^* p_1 / q_2 + (1 - a)^* p_1$ q=(totable)(v i)(totable)(r, ii)(iii)(q), so=1; c0=; jct=0; for (j=0; jct=; j++) { ss2[j]=0; cc2[j]=0; cc3[j]=0; pp1[j]=0; pp2[j]=0; } for (j=0; j<=m2; j++)</pre> q=(double)(W*T-(j+1)*L2)/(double)L1; il=floor(q); for (i=0; i<L; i++) { ss3[i]=0; cc4[i]=0; } for (i=0; i<L; i++) { ss3[i]=0; cc4[i]=0; } if (i==(il+1)) ict1=ict; pb=s1; if (ict>ict1) for (k=0; k<(ict-ict1); k++) pb*=1e15; if (ict<ict1) for (k=0; k<(ict1-ict); k++) pb/=1e15; pp+=pb; if (c1>1e15) if (c1>1e15)
{ s1=s1/1e15; c1=c1/1e15; ict+=1; }
if (s1<1e-15)
{ s1=s1*1e15; c1=c1*1e15; ict-=1; }
} /* end of for_i */
if (ict>ict1) for (k=0; k<(ict-ict1); k++) pp/=1e15;
if (ict>ict) for (k=0; k<(ict1-ict); k++) pp*=1e15;
b=ps; idt=idt</pre> pb=pp; ict1=ict; s3=0; c4=0; ict=-1; for (i=(L-1); i>=0; i--) { if (ss3[i]==0 && ict==-1) continue; if (ict==-1) ict=i; for (k=0; k<i; k++) { ss3[k]*=1e-15; cc4[k]*=1e-15; } s3+=ss3[i]; c4+=cc4[i]; $\begin{array}{l} \text{if}'(j{=}0) \; s0^*{=}p2/j; & /* \; s0{=}p2{-}j/j! \; */ \\ \text{if}\;(j{=}{=}1) \; c0{=}p2; \; \text{if}\;(j{>}1) \; c0^*{=}p2/(j{-}1); \; /* \; c0{=}jp2{-}j/j! \; */ \\ k{=}ict{+}jct; \end{array}$ if (k>=0) \$s2[k]+=s0*s3; cc2[k]+=s0*c4; cc3[k]+=c0*s3; $\label{eq:second} \begin{array}{l} /* \ s2=sum_j(\ p2\sim_i/j!.sum_i(p1\sim_i/i!)\)\ */\\ /* \ c2=sum_j(\ p2\sim_i/j!.sum_i(ip1\sim_i/i!)\)\ */\\ /* \ c3=sum_j(\ jp2\sim_j/j!.sum_i(p1\sim_i/i!)\)\ */ \end{array}$ if (j==0) jct1=ict1+jct; k=ict1+jct-jct1; if (k>=0) { pp1[k]+=s0*s1; pp2[k]+=s0*pb; } if (c0>1e15) { s0=s0/1e15; c0=c0/1e15; jct+=1; } if (s0<1e-15) { s0=s0*1e15; c0=c0*1e15; jct-=1; } m1=il; } /* end of for_i */ jct=-1; s2=0; c2=0; c3=0; for (i=(L-1); i>=0; i--) }f (ss2[i]==0 && jct==-1) continue; if (jct==-1) jct=i: for (k=0; k<; k+++) { ss2[k]/=1e15; cc2[k]/=1e15; cc3[k]/=1e15; } s2+=ss2[i]; c2+=cc2[i]; c3+=cc3[i]; if (pp1[i]==0 && ict==-1) continue; if (ict==-1) ict=i; for (k=0; k<i; k++) { pp1[k]/=1e15; pp2[k]/=1e15; }

#include <math.h>

pb1+=pp1[i]; pb2+=pp2[i];
}
ict+=jct1;
pb1/=s2; pb2/=s2;
if (ict>jct1) for (k=0; k<(jct1-ict); k++) { pb1/=1e15; pb2/=1e15; }
if (ict>jct1) for (k=0; k<(jct1-ict); k++) { pb1*=1e15; pb2*=1e15; }
pb=a^pb1+(1-a)^pb2;
printf("%ft%ft%ft%ft",cu,pb,pb1,pb2);
pb=(L1*a^pb1+L2*(1-a)*pb2)/(L1*a+L2*(1-a));
printf("%ftn",pb);
}</pre>

B.1 Network Model

The following is the network model "t2_net" used. It consists of only one node named "t2" whose model is included in Section B.2.



B.2 Node Model

The following is the node model "t2" used. Notice that the processor model "t2_scheduler" in the *queue scheduler* component and the processor model "t2_release" in the *processor release* component are to be substituted by "mm*_scheduler" and "mm*_release" in Section B.3 through Section B.5.

1 7 06:55:28 1994 Page 1 of		Streams: Jer [0]			<u>م</u>				kets)			Loth)
Fri Jan	trrbutes Attr Value: src src src src src src src src src src	Streams Output schedu	ttributes Attr Value: scheduler	0 00 01	12_schedule 0 disabled	enabled disabled disabled	duene	ue (0) Attributes	Attr Value: infinite (pacl	infinite (bits)	ue (1) Attributes	Attr Value:
8	Attr Name: name name interarrival args interarrival args pk size pof pk size args pk size args field (0) args packet format icon name	index Input Streams: 0	Attr Name: name	in strm count out strm count subqueue count	process model default priority intrpt interval	endsim introt failure introts recovery introts	icon name	anbqns	Attr Name: pk capacity	bit capacity	enbqns	Attr Name:
ode Model Report: ode Model	enerator											
ZŻ	ideal g src		queue scheduler									
z z _ j	ideal g src		queue scheduler									
1 Page 2 of 2 N	sec deal g		scheduler scheduler									
Fri Jan 7 06:55:28 1994 Page 2 of 2 N	late: strain a lateral general g	Output Streams:	gueue gueue									
Fri Jan 7 08:55:28 1994 Page 2 of 2 N	Attributes Attributes Indeals Release B B C_release disabled disab	Streams Output Streams: D	gueue gueue									

5	Output Streams:	release [0]	
Stream	Input Streams:	src [0]	
	index	0	

72

Node Model Report: 12 Node Model

;

processor **release**

		and the second se	
Packet Format Report:	8	Fri Jan 7 06:56:34 1994	Page 1 of 2
Parameter Model			
	Fie	Id O	
	Component Name:	Component Value.	-1-
		wavelength	
		integer	
	Size (Dits) Default Value	D	
	Default Set	unset	
			I
	Fie	14.1	
	Component Name:	Component Value:	-
	Field Name	timeslot	T-
	Type	integer	
	Size (bits)	0	
	Default Value		
	Default Set	unset	
	Fie	1d 2	
	Component Name:	Component Value:	1
	Field Name	SIC	T
	Type	integer	
	Size (bits)	_ 0	
	Default Value		
	Default Set	unset	
	Fie	1d 3	r
	Component Name:	Component Value:	
	Field Name	dst	
	Type	integer	
	Size (bits)	0	
	Default Value	+0001	
	Delauli Sel	lasin	7
	Fie	10 4	[
	Component Name:	Component Value:	 ,
	Field Name	SVC	_
	Type	double	
	Size (bits)	0	
	Default Set	tinset	
			_

Packet Format Report: 2	Fri Jan 7 06:56:34 1994	Page 2 of 2
Parameter Model		

Default Value Default Set unset	Fie Component Name: Field Name Type Size (bits)	<i>ld 5 Component Value:</i> type integer 0
	Default Value Default Set	unset

The packet used in the system has the following format:

B.3 Single Class, Uniform Traffic System

B.3.1 Blocking System

The termination criterion used for M/M/1/B/L+1, M/M/1/B/CL, and M/M/1/B/RL systems is to end the program when either 99999 seconds have elapsed or the "steady state" condition has been reached. The program starts to monitor periodically the channel utilization (cu), the blocking probability (pb), and the blocking probability due to transmitter or receiver conflicts (pbc) once the measured load is within 1% of the offered load derived in Section 4.1. If the newly measured cu, pb, and pbc are within 1% of the previous measured values, we say that "steady state" condition is reached. The program will be ready for termination once the measured load is within 5% of the offered load.

The seed used for each simulation has value NWL, where N is the number of users, W the wavelength channels, and L the throughput requirement. So for N equals to forty, W equals to eight and L equals to one, the seed is 4081.

<u>M/M/1/B/L+1:</u> OPNET reports for the scheduler and release processor models.



Wight+1 Uniform Traffic System eduer Model 30 fines State Variables 30 fines State Variables 30 fines Intervieweight.noc. 11 www.der.noc. Intervieweight.noc.	V1/	(B/L+1 Uniform uler Model	Traffic System	
State Variables 30 lines State Variables 30 lines 0 hjid user id: int wavelength_no: int wavelength_no: int Wit: int Nit: int Nit: int Nit: int Nit: int Nit: int Nit: int Nit: berefunden *de.dist: Discribution *de.dist.dist.dist.dist.dist.dist.dist.dist				
State Variables 30 fines Objid variables 000 int variables 000 int variables von.id int variables von.id int variables int variables int variables variables beribulion variables				
Objid in int int int int int int int int int	S	ate Variables		30 lines
Objid Src id. int Wavefendt_non. int Wavefendt_non. int Wavefendt_non. int Wavefendt_non. int Wavefendt_non. int Wavefendt_non. int Wit. int Wit. int N1. int N1. int N1. int N1. int N1. bisinbution Var_angl.00; bisinbution -var_angl.01; bisinbution -var_angl.02; bisinbution -var_angl.03; bisinbution -var_angl.04; bisinf Uouble bisinf Var_angl.04; bisinf </th <th></th> <th>Objid</th> <th>lown_id:</th> <th></th>		Objid	lown_id:	
1 Int Workergeth_no: 1 Int Warkergeth_no: 1 Int Umsize: 1 N1: Scalar 1 Kolo_no: N1: 1 N1: N1: 1		Objid	\src id;	
5 int Wavelength_no: Um_size: int 10 int Wit: NI: NI: NI: NI: NI: NI: NI: NI: NI: NI		int	\node_no:	
5 int Un size: int 1 int Un size: but we use: but we inte: but w		int	\wavelength_no:	
int 11: int N1: int N1: int N1: int N1: chait N1: chait N1: bistribution 'wr.rate: Distribution 'wr.gf20; log int 'wr.gf22; double 'wr.gf22; double 'wr.gf22; double 'wr.gf22; int 'wr.conn:: int 'wr.conn:	2	, Tur	Vim_size:	
11 Int Not_not. 11 0 doubte vvr_intre. 12 Destribution vvr_intre. 13 Distribution vvr_intre. 14 vvr_intre. vvr_intre. 15 Distribution vvr_intre. 16 vvr_intre. vvr_intre. 17 Distribution vvr_intre. 18 Distribution vvr_intre. 19 Distribution vvr_intre. 11 Distribution vvr_intre. 12 Distribution vvr_intre. 13 Distribution vvr_intre. 14 Distribution vvr_intre. 15 Distribution vvr_intre. 16 vvr_intre. vvr_intre. 14 Distribution vvr_intre. 15 Distribution vvr_intre. 16 vvr_intre. vvr_intre. 10 double vvr_intre. 11 vvr_intre. vvr		int	VTI:	
10 double char W: uvr.arg(20); bistribution 13 Distribution "uvr.arg(20); uvr.arg(20); bistribution 14 Distribution "uvr.arg(20); uvr.arg(20); bistribution 15 Distribution "uvr.arg(20); uvr.arg(20); bistribution 16 Distribution "uvr.arg(20); uvr.arg(20); bistribution 17 Distribution "uvr.arg(20); uvr.arg(20); bistribution 18 Distribution "uvr.arg(20); uvr.arg(20); bistribution 19 double "uvr.arg(20); uvr.arg(20); bistribution 20 double "uvr.arg(20); uvr.arg(20); bistribution 21 double "uvr.arg(20); bistribution 22 double "uvr.arg(20); bistribution 23 int "uvr.arg(20); bistribution 24 uvr.arg(20); int "uvr.arg(20); bistribution 25 int "uvr.arg(20); bistribution 26 int "uvr.arg(20); bistribution 27 int "uvr.arg(20); bistribution 28 int "uvr.arg(20); bistribution 29 int "uvr.arg(20); bistribution 20 int "uvr.arg(20); bistribution 23 int "uvr.arg(20); bistribution 24 uvr.arg(20); bistri "uvr.arg(20); bistribution<		int	\slot_no;	
10 int N1: 11 double ver_time: 12 Distribution "ver_tats: 13 Distribution "ver_dist: 14 Distribution "ver_dist: 15 Distribution "ver_dist: 16 Distribution "ver_dist: 17 Distribution "ver_dist: 18 Distribution "ver_dist: 19 Distribution "ver_dist: 10 orbite "ver_dist: 11 Distribution "ver_dist: 12 double "ver_dist: 13 double "ver_text: 14 ver_text: "ver_text: 15 int "ver_text: 16 ver_text: "ver_text: 11 ver_text: "ver_text: 11 <td></td> <td>int</td> <td>N:</td> <td></td>		int	N:	
01 double double wr.arg(20) Bestribution bestribution wsr.afg(20) Bestribution wsr.afg(20) bestribution wsr.afg(20)		int	NN1;	
char urv.arg(20); urv.arg(20); Distribution "urv.arg(20); urv.dist; Distribution "urv.arg(20); urv.dist; Distribution "urv.arg(20); urv.dist; 20 double urv.arg(20); double "urv.arg(20); urv.dist; 21 double 22 double 23 urv.arg(20); int 24 "urv.arg(20); urv.dist; 25 urv.arg(20); urv.com:: 26 "urv.arg(20); urv.ex.arg(20); 27 urv.arg(20); urv.urv.ex.arg(20); 28 urv.arg(20); urv.urv.ex.arg(20); 29 urv.arg(20); urv.urv.ex.arg(20); 20 urv.arg(20); 21 "urv.arg(20); 22 urv.arg(20); 23 urv.arg(20); 24 "urv.arg(20); 25 urv.arg(20); 26 "urv.arg(20); 27 "urv.arg(20); 28 urv.urv.ex.inv.urv.ex.inv	9	double	\svc_time;	
15 Descriptution "urr_arte: 16 Discriptution "dx_c.dis: 17 Discriptution "dx_c.dis: 18 Discriptution "dx_c.dis: 19 Discriptution "dx_c.dis: 10 Discriptution "dx_c.dis: 11 Discriptution "dx_c.dis: 12 Double "tx_u.prov. 13 Double "tx_u.prov. 14 Double "tx_u.prov. 15 int "tx_u.prov. 16 Obto the prov. "tx_u.prov. 17 Double "tx_u.prov. 18 Double "tx_u.prov. 19 int "tx_andy. 11 "tx_andy. "tradit." 12 Pacer "tradit."		char	\arv_arg[20];	
15 Distribution "src. dist. 15 Distribution "dr. dist. 16 Distribution "dr. dist. 17 Distribution "dr. dist. 18 Distribution "dr. dist. 19 Distribution "dr. dist. 10 Distribution "dr. dist. 11 Note Post. 11 Vec.omt. "dr. dist. 11 Vec.omt. "dr. dist. 11 Vec.omt. "dr. dist. 11 Vec.adir. "dr. dist.		double	larv_rate;	
15 Distribution "dst_tilds: 10 Instribution "dst_tilds: 10 Ing_ int Vocked_pk: 10 obtolie Vocked_pk: 10 double Vocked_pk: 11 Vocked_pk: Vocked_pk: 12 double Vocked_pk: 13 double Vocked_pk: 14 vocked_pk: Vocked_pk: 15 int Vocked_pk: 16 vocked_pk: Vocked_pk: 17 vocked_pk: Vocked_pk: 18 int Vocked_pk: 19 int Vocked_pk: 11 Vocked_pk: Vocked_pk:		Distribution	*\src_dist;	
15 Destribution long int long int long int long int long int locked.c.pk; long double both double both locked both locked.c.pk;		Distribution	*\dst_dist;	
long int woral.pk: long int blocket.pk: long int blocket.pk: louble vu.max. double vu.max. double vb.prev: double vb.prev: double vb.prev: int vb.srady: int vb.srady: int vb.srady: lint vb.srady: int vb.srady: lint v	15	Distribution	*\svc_dist;	
Dorg int Nocked_pk: 20 devide vemerc; 20 devide vemerc; 20 devide vemerc; 20 devide vemerc; 21 devide vemerc; 22 int Vemerc; 23 int vecount; 24 int vecount; 25 int vecount; 26 int vecount; 27 int vecount; 28 int vecount; 19 int ve_ady; 11 veady; pect** 11 veady; pect**		long int	viotal_pk;	
Image Molecked c.pk: 00 double vurmer: double vurmer: double vurger: double vurg.prev:		long int	Vblocked_pk:	
0 double vut prev: double vut prev: double vbt prev: double vbt prev: int vbt prev: int vbt strady; int vbt strady; 3 fines		long int	<pre>blocked_c_pk:</pre>	
20 double vu, prev.; double vp, prev.; double vp, prev.; double vp, prev.; double vp, prev.; int vu, ad, prev.; int vp, connt.; int vp, connt.; int vp, connt.; int vp, creaty; int voad_steady; int vad_steady;		double	kcu_max;	
double vb. prev.; double vb. prev.; double vb. prev.; double vb. count; int vc. count; int vc. count; int vc. steady; int vb. steady; int vc. steady;	ຊ	double	keu_prev;	
double vbc. prev. 1 vbc. prev. 1 vbc. count. 1 vbc. count. 1 vbc. seady. 1 vbc. staady.		double	\ph_prev;	
double Woad, prev: wak count: int Woad, prev: wak count: wak count: wak count: wak count: wak count: wak count: wak count: wak count: int Woad, prev: wak count: wak cou		double	\pbc_prev;	
int vk. count: time 25 int int vc. steady; vhb. steady; int 30 int int vb. steady; vbb. steady; int 1 vbb. steady; vbb. steady; 1 vbb. steady;		double	Voad_prev:	
23 int v		int	pk_count:	
int bu steady: int bu steady: int bub steady: int bub steady: int bub steady: int bub steady: Femporary Variables 3 lines Packet by: int straticular	52	Li	Vr_count:	
int vb. steaty; int vbr. steaty; int vbr. steaty; int vcady; Temporary Variables 3 fines Peckt* pkm: int scrist.j;		int	kcu_steady;	
int von steady: 30 int voad_steady: Temporary Variables 3 fines Packat hydric		int	pb_steady:	
0 int Voad_Steady: Veady: Temporary Variables 3 lines Packet Polor: int scrdst.j;		int	phc_steady:	
30 Int veady. Temporary Variables 3 fines Packet" phon. int secisti,j.		int	Voad_steady:	
Temporary Variables 3 lines 3 lines texts to a scratching the scratching text of the scratc	8	int	Vready:	
Temporary Variables 3 lines Packut pkor: int srcdst.jj:				
Packet before int secdstaji	Ē	morary Varia	ahlac	3 lines
Packst* pkptr. int src.dst.ij;	ĺ.		24MB	
int src.dst.i.j:		Packet pkptr.		
		int src.dsi	tá.j:	

Proces	ocess Model Report: mm1b11 scheduler	7 03:49:18 1994 Page 3 of 6
WW1, schedu	MV1/B/L+1 Uniform Traffic System theduler Model	
đ		
5	State U: Init (Enter Execs)	forced, 24 lines
v	99 jma sim_attr_get(OPC_IMA_INTEGER, xr. &node_no); 10 jma sint_get(OPC_IMA_INTEGER, vr. &wavelength_no); 10 jma sim attr_get(OPC_IMA_INTEGER, vr. & &nodength_no); 10 jma_sim_attr_get(OPC_IMA_INTEGER, vr. & &nodength_no);	
^	own_id=op_id_self():src_id=op_topo_in_assoc(own_id_0): op_ima_obi_attr_get(src_id.tinterarrival_args'&arv_arg(0)):	
-	arv_rate=atof(&arv_arg(0)); arv_rate=1/arv_rate;	
3	No N=wavelengub_no*fm_size: svc_time=(double)fm_size/(double)(siot_no+1): T1=floo(svc_time): N1=T1*wavelengub_no: svc_time=(double)fm_size(double)	joruo:
15	<pre>swc_dist=up_dist_boad(* exponential: swc_dimc.0); 15 src_dist=up_dist_hoad(* uniterm_int* 1,(double)mode_no); dst_dist=up_dist_hoad(* uniterm_int* 1,(double)mode_no);</pre>	
20	total_pke() blocked_pke(), blocked_c_pke(), released_pke(), on_jin_svc=0; time=0 ou_max=(doubk=)N1*stor_no)/(double)Ni load_prev=arv_rate/(double)wavelength ou_ueasiy=0; ph_steady=0; phc_steady=0; load_steady=0; ready=0;	
	<pre>for (i=0,iCT1;i++) for (j=0;)<wardength_no:j++)< td=""><td></td></wardength_no:j++)<></pre>	
Sta	State 0: init (CET's)	
B	CET Cond: (ARRIVAL)	
2	Trans: pk_prepare	
ų ž	CET Cond: (dcfault)	
:	Trans: idle	
Sta	State 1: pk prepare (Enter Execs)	forced, 6 lines
	src=(int)op_dist_outcome(src_dist); next:	
	dst=(int)op_dist_outcome(dst_dist): if (dst == stc) goto next:	
2	5 iotal pk+-i. pkpti⊐up pk get∖up intrpt strmi∪.	
Sta	State 1: pk prepare (CET's)	
មីន	CET Cond: (1) #0 Fran: .	
2	Trans: schedule	

7	5
•	~

Soche 1 <th1< th=""> 1 <th1< th=""> <th1< th=""></th1<></th1<></th1<>	2005 20 20 20 20 20 20 20 20 20 20 20 20 20	Model Report: mm1D/1 _scheduler L=1 Uniform Traffic System Nodel 2. idle (Enter Execs) if (Led. Jearly && rates -1.001) (= (ad. Stardy && rates -1.001) (= (ad. Stardy =: rate: the (NY * p. sim, time()); ratio=tem it (ratio > 0.99) && rates -1.001)) = pro-stardy & rates -1.001) = pro-stardy & rates -1.001 & ratio=tem it (ratio > 0.99) & ratio -1.001 & ratio=tem it (ratio) = 1.001	Fri Jan 7 03:49:16 1994 Fage 4 of 6 unforced. 57 lingss unforced. 57 lingss viead_preve: unforced. 57 lingss ouble/blocked_c_pW(double/blocal_pk: eased_pW(double/pk_count: tr_count=floor(temp)+20. eased_pw(double/pk_count: tr_count=floor(temp)+20. u_preve: inte() shot(ill) clune): u_preve: inte() shot(ent): tr_count=floor(temp)+20. u_preve: inte() shot(ill) clune): u_preve: inte() shot(ent): tr_count=trice(temp)+20. rut_preve: inte() shot(ent): tr_count=trice(temp)+20. rut_preve: inte() shot(ent): tr_count=trice(temp); trice(temp); tric
S	25	f (ready) f tempetoal_pt*sto_no*svc_timo(N*op_sim_time()): ratio-tem if (ratios/J995&&ratior.) (05&&ktime(load_pt*svc_time)&&kti	/load_prev: me*sloLno)/(N*0p_sim_time())<≂≎u_max&&'ENDSIM)
n	2	l If (readv)	
4	\$2	ratio=0; for (j=0; k=17; i++) for (j=0; j>wavelength_no; j++) if (sloq[1][]; sc!=(for (j=0; y=0, no "N1/N"*0, sim_tline()); p }	ratio+=(op_sin_time ();slo(1)[j],clinc); _coant=2*floor(tenp): tr_coant+=1;
		if (cu_strady && pb_strady && pbc_strady) rrady≈1: if (!rrady) f	
4	 9	if (pkc_previ=d) ratio=temp/pkc_prev: te is (interpero) ratio=1; teke ratio=0; if (ratio 0.0998 & ratio < 1.001) tes pbc_prev=temp:)	
<u> </u>		if ('p ^f steady) temp=(double)blocked_c_pk((double))otal_pk;	
(r)	30	temp=(double)blocked_pk/(double)local_pk; ti (top_prev?=0) ration=pp/p_prev; clss if (tomp==0) ration=1: else ratio=0; if (ratio > 0.099 & & ratio < 1.001) pb_strady=1: else ph_prev=temp;	
~	22	} if (!pbsteady)	
		rations: a failed in the second se	o+=(op. sim_time ()-slod[i][j];clime): ou_prev: ime)&&t temp < cu_max) cu_steady=1:
		if (:cu_steady) 	
	15	} if (load_steady && released_pk>(u_count*pk_count))	
	0]	<pre>load_steady=1: ratio=0; for (i=th, i<1); i+1) it (slot(i)[i], sc:=0) ration for (i=th, j-wavelength, no: i+1) it (slot(i)[i], sc:=0) ration cu_proved(marterialo)* slot_not(N*0p, sim_time()); ph, prever(auble) blocked_pdk(double loual_pk; pkc_prev=(for marterial)* slot_count=flore(temp); temp=(double) he for phaseu_preverN1: pk_count=flore(temp); temp=(double</pre>	o+=(o p. sim_time ()-stoti][j].tume): oubleblocked_c_pk(double)total_pk: cssd_pk/doublepk_count: tr_count=floor(temp.H-20):
vo	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	temp=total_pk*slot_no*svc_time/(N*op_sim_time()); ratio=tem if (ratio > 0.999 && ratio < 1.001) {	Moad_prev:
		ii (!load_steady && released_pk>() {	
<u> </u>	Stat	e 2: idle (Enter Execs)	unforced, 57 lines
M/M sche	V1/B/	L+1 Uniform Traffic System r Model	
Proc	Sess	Model Report: mm1bl1_scheduler	Fri Jan 7 03:49:18 1994 Page 4 of 6

/M1/	/B/L+1 Unifor	m Traffic System		
	INCURE			
Š	ate 2:	idle (CET's)		
ы В	T Cond:	(ARRIVAL)		
Ŷ	Exec:			
	Trans:	pk_prepare		
U	T Cond:	(default)		
Ŧ	Exec:			
	Trans:	idle		
U	T Cond:	(ENDSIM)		
¥	Exec:	record_stats();		
	Trans:	idle		
Sta	ate 3:	schedule (Enter Execs)	forcea	1, 4 lines
	if (no_in_svc	(==NI)		
	{blocke	:d_pk+=1;		
	if ('find	resource(src.dst.pkptr)) {blocked_pk+=1, blocked	d_c_pk+=1: op_pk_destroy(pkpu):}	
ö	ate 3:	schedule (CET's)		
뜅	ET Cond:	(1)		
¥	Exec:			
	I rans:	Idle	1.1.1.1.1.1.1	

P	Cess	Model Report: mm1bi1_scheduler	Fri Jan 7 03:49:19 1994 Page 6 o	9
WN Sche	V1/B	vL+1 Uniform Traffic System er Model		
4	E	ction Block	51 lines	Π
		record_stats()		
		int Lij;		
	5	double cu=0; extern double time:		
		extern channel_asen stort 1.30/[1.0); for (i=0: i <t1: i++)<="" td=""><td></td><td></td></t1:>		
	10	for (j=0: j <wavelength_no: (slot[i](j].src!="0)" if="" j++)="" td="" time+:<=""><td>(op_sim_time()-slot(i][j].ctime).</td><td></td></wavelength_no:>	(op_sim_time ()-slot(i][j].ctime).	
		cu=time*sloLno/(N*op_sim_time()): if (cu > cu_max) op_stat_write_scalar(*load*.total_pk*slot_no*svc_time/(N*o	cu=cu_max; _sim_time()));	
		op_stat_write_scalar(*cu*.cu); op_stat_write_scalar(*pb*.(double)blocked_pk/(double)total_p		
	15	op_stat_write_scalar("pbc".(double)blocked_c_pk/(double)tote	_pk);	
	20	int find_resource(sc.dt.packet)		
		int sc.at; Packet* packet;		
		{ int i.ik.an:		
	25	double svc:		
		extern int no_in_svc: extern channel_asgn sloq[130][10];		
		ap=0;		
	२	10f (1=01<11111++) /* for each column */		
		k=-1; for (j=0;i <wavelength_no;j++)< td=""><td></td><td></td></wavelength_no;j++)<>		
	25	{ }}		
	3	if (slot[i][j].src==sc slot[i][j].dst==dt) goto next_col		
		<pre>if (k!=-1) { ap=1: goto assign: }</pre>		
4	6	next_col: ap=0:		
	2	-		
		if (!ap) return 0: else		
	:			
	\$	assign: stotfilkl.src=sc: slotfilkl dst=dt: slotfilkl.ctime=0p sim	time():	
		op_pk_nfd_set(packct.*wavelength.*k): op_pk_nfd_se	packet timeslot i).	
	;	SVC=0p_uist_outconnetsvc_uist); no_in_svc+=1; up_pn_s	nd_detayed(packet.u.svc);	
	8			

W1/F ase	B/L+1 Uniform Traffic S Model	ystem			
	model				
			Summary		
	Alimber of	[[aadar		-	
	States	Block	State Variahlae	l emporary Variables	Function
	-	Yee	No	Vae	No
				-	
He	ader Block			9 11	Set
	tynedef struct				001
	informer series				
	int src:				
	int dst;				
5	double ctime;				
	} channel_asgn:				
	extern int	no_in_svc.released_	ok:		
	extern double	time;			
	extern channel asgn	slot[130][10];			
j a	moorary Variables			210	Seu
	Packet* pkptr: int wavelength.time	slot:			
Sta	te 0: discar	d (Enter Execs)		Inter	nrrad A linac
	if (op_intrpt_type()==OP	C_INTRPT_STRM)			
	pkptr=op_pk_get(op	intrpt_strm()):	, But the net (thread		
5	time+=(op_sim_time	()-slot[timeslot][wavele	acingur), op_ph_niu_g ngth].ctime);	seu prput " cumes lot ", & uncs	1005
	<pre>slot[timeslot][wavelc no_in_svc-=l; release</pre>	ngth].src=0: slot[timeslo sd_pk+=1: op_pk_destr	t][wavelength].dst=0; s oy(pkptr):	:lot[timeslot][wavelength].ctim	ie=0;
E S	te 0: discar	d (CET's)			
1	1000	4 44 9			
5 9	Cond: (1)				
5	Trans: discard				
	11010				

<u>M/M/1/B/CL:</u>

OPNET report for the *scheduler* processor model. The *release* model is identical to that in M/M/1/B/C+1.



5 2	ess Model Report: mm1bl_scheduler Fri Jan 7 03:36:12 1994 Page 4 of 7	Process Model Report: mm1bl_scheduler
¥ Å	1/B/CL Uniform Traffic System duler Model	MMV1/B/CL Uniform Traffic System scheduler Model
	state 2: idle (Enter Execs) unforced, 60 lines	State 0: init (Enter Execs)
	if ('load_steady & & released_ph>0) fod	op_ima_sim_attr_get(OPC_IMA_INTEGER, 'N' & code, on ima_sim_attr_cost(OPC_IMA_INTEGER_'co-& cost
	<pre>kmp=uotal_pk*slot_not*yc_time(N*op_sim_time()): ratio=temp/load_prev: if (ratio > 0.999 && ratio < 1.001)</pre>	op ima sin attr get(OPC_IMA_INTEGRR, Tr. & Ma in attr get(OPC_IMA_INTEGRR, Tr. & Ma in attr get(OPC_IMA_INTEGRR, Tr. & Statu
	{ load stradb=1; ratio=0;	5 own id=on id self(): sec id=on in assoc(own id 0
	$\int_{C^{\infty}} \int_{C^{\infty}} \int_{C$	op_ima_obj_attr_get(src_id, interarrival args",&s
	in (1-22, 2-24, 2-24, 2-27) (f (slot(])], statio+=(0p, sim_time()-slot(i])[j], ctime): 0 ctimerari(interstations): skon stations (sim fine()):	arv_rate=atof(&arv_arg[0]); arv_rate=1/arv_rate;
	pb_prev=(double)blockted_pk/(double)bloat_pk; pbc_prev=(double)blocked_c_bu/k(double)blockted_pk; temp=cu_prev*N1: pk_count=floor(temp): temp=(double)released_pk/(double)pk_count: tr_count=floor(temp)+20:	N=wavelength_no*fm_sizz: xvc_uime=(double)fm_sizz(d T1=floor(svc_uime): N1=T1*wavelength_no:
		<pre>swc_dist=op_dist_load('exponential'.svc_lime.0); src_dist=op_dist_load('meiterne 'ser' 1 (Anihbahode ' screential');</pre>
	if (load_strady & & !trady & & released_pk>(tr_count*pk_count))	dst_dist=op_dist_load(" uniform_int". 1. (double) hode_r dst_dist=op_dist_load(" uniform_int". 1. (double) hode_r
	if ('tu_steady)	total_pk=0; blocked_pk=0; blocked_c_pk=0; released_pk=
		cu_intat=(outbreytyn'r stor_novy(cuotoreytyr iotal_prev≃ary 20 cu_strady=0; pb_strady=0; pbc_strady=0; load_stcady=0;
	for (act: kc11:1+++) for (=0-); evaruselength, nor, j++) for (=0-); act=0-0: nor); ++0 (=0-); here (=k-10-1);	for (i=0;i <t] :="" i++)<="" td=""></t]>
	temp=(time+ratio)'sld, no(N*09, sim_three)): ratio=temp?cu_prev; if (ratio > 0.999 && ratio < 1.001 && time<(total_pk*sv:.time) && temp<.t	(stotil):
	II (:pb.steady)	State 0: Inft (CET'S) CET Cond: (ARRIVAL)
	0 temp=(double)blocked_pk/double)toral_pk: if (pb_prev;=0) ratio=temp/pb_prev;	#0 Exec: : Trans: bk prepare
	else if (temp==0) ratio=1: else ratio=0: if (ratio> 0.999 && ratio < 1.001) pb steadv=1:	CET Cond: (dcfault) #1 Ever:
	clse ph_prev=temp:	Trans: idle
	uf ('pbc_steady)	
	temp=(double)blocked_c_pk(double)total_pk; if (rebe: news/=0) to in-remember: - news	State 1: pk prepare (Enter Execs)
	1 concentration of the second procession of th	src=(int)op_dist_outcome(src_dist); next:
	II (TAILO > 0.337 AAX TAILO < 1.001.) PRC_SREADY=1: clse pbc_prevelentp:	dst=(int)op_dist_outcome(dst_dist); if (dst == src) goto next;
	if (ou steady & & ph_steady & & phc_steady : teady =!	5 [5 [101d] nk+= ; nkntr=00 pk gettop introt strm());
	ii (fready)	
	ratio=4); for (i=0: k;(T1: i++)	State 1: pk prepare (CET's)
	is (1-2), we according not, 1++) if (slot[i][], src[=0) ratio+=(op_sin_time()-slot[i][j] ctime):	CET Cond: (1) #0 Exec: :
	$\operatorname{contract}_{1}$	Trans: schedule
·	5 lif(ready)	
	kmp=total_pk*slot_no*svc_time/(N*op_slim_time()); ratio=temp/loal_prev; if (ratios0.993& ratio=10108.8/nime=crotal_nd*svc_time/& strim=*clot_mo/N*-no_sim_time())>======xe_0.8.8.1FNDSIM	

Process Model Report: mm1bl_scheduler	Fri Jan 7 03:36:11 1994	Page 3 of 7
M/M/1/B/CL Uniform Traffic System		
scheduler Model		

s	te 0: init (Enter Execs)	forced, 24 lines	
	(9) Ima. sim. attr. gret(OPC_IMA_INTEGER, w. & kuvele. no): (a) Ima. ima. attr. gret(OPC_IMA_INTEGER, w. & wwwelengihno). (a) Ima. sim. attr. gret(OPC_IMA_INTEGER, v. v. & film. size): (a) Ima. sim. attr. gret(OPC_IMA_INTEGER, v. v. & slot_no):		
<u>.</u>	own_id=op_id_get(Y;src.id=op_topo_in_gesoc(own_idi0); op_imma_obj_attr_get(src.id.thretarrival args.Xanv_arg(0));		
2	ary_fate=atof(&ary_arg[0]); ary_rate=1/ary_rate;		
2	N=wavelengub_nov ⁺ im_size: svc_time=(double)im_size(/double)slot_no: T1=ftoor(svc_time): N1=T1 *wavelengub_no:		
15	<pre>swc_dist=pp_dist_load(*exponential'swc_inne()); src_dist=pp_dist_load(*uniform_innt',1,(double)hoode_no); dist_dist_load(*uniform_innt',1,(double)hoode_no);</pre>		
20	Intal_pk=0: blocked_pk=0: blocked_c_pk=0: released_pk=0: no. in_sve=0: time=0: cu_max=(double/N1*sloc_no)/(double/N: taad_prev=mr_rate/(double)wavelength_no: cu_steady=0: ph_steady=0: phc_steady=0: toad_steady=0: ready=0:		
	for (i=0); <tt i±++)<br="">for (i=0;<particle=0;0;++) (slot(j)[[],sr=0; slot(j)[],ds=0; slot(j)[[],crime=0;]</particle=0;0;++) </tt>		

ļ			
3	te 0:	init (CET's)	
Ē	Cond:	(ARRIVAL)	
0	Exec:		
	Trans:	pk_prepare	
Π	Cond:	(default)	
-	Exec:		
	Trans:	idle	

forced, 6 lines

Proc	ess Model Report: mm1bl scheduler	r Fri Jan 7 03:36:12 1994 Page 6 of 7	Process Model Report: mm1bl scheduler
MM	/1/B/CL Uniform Traffic System		WM/1/B/CL Uniform Traffic System
sch	sduler Model		scheduler Model
	i,j.ap.fit.ps 5 double svc.	.pd.cf.cs.cd.nf.ns.nd.ca(8).na[8].na[2]:	00_sim_end('reaching steady state'.'.'.'.')
	extern int no_in_svc extern channel_asgn slot[130][1	: [0]:	
	ap=-1: ps=-1: ps=-1: 10 for (=0:i <t1:i++) *="" for<="" th=""><th>each column */</th><th>State 2: idle (CET's) CET (Cond: (default)</th></t1:i++)>	each column */	State 2: idle (CET's) CET (Cond: (default)
	{ if (i==0)		#0 Exec: :
	cf=0, cs=-1; cd=-1; for (j=0;j <wavelength_no< td=""><td>15;++)</td><td>CET Cond: (ENDSIM) #1 Exec: record_stats(): 7.77ars: rede</td></wavelength_no<>	15;++)	CET Cond: (ENDSIM) #1 Exec: record_stats(): 7.77ars: rede
	(if (slot[i][j].src==0) if (slot[i][j].src==sc) if (slot[i][i].stc==sc)	(ca(cf]=; cf+=1:) ce=j: cf=z:	CET Cond: (ARRIVAL) #2 Exec: : Trans: pk_prepare
	10	-(-m)	
	uf (i=(T1-1)) { ns=-1: nd=-1:		State 3: schedule (Enter Execs) If (no in sec = N1)
-	ts else		(blocked_pk+=1; op_pk_destroy(pkpt);)
•	nf=0; ns=-1; nd=-1; for (j=0;j <wavelength_no< td=""><td>sj++)</td><td>if (:find_resource(sr.dst.pkptv)) 5 (blocked_pk+=1; blocked_c_ak+=1; op_pk_destroy(pkpr);)</td></wavelength_no<>	sj++)	if (:find_resource(sr.dst.pkptv)) 5 (blocked_pk+=1; blocked_c_ak+=1; op_pk_destroy(pkpr);)
	50 if (slot[i+1][j].src== if (slot[i+1][j].src==	0) [najn(]=j: n(+=1:) sc) ns=j:	
	11 (stout)+1 JJ.0st==	(1) NG=):	State 3: schedule (CET's) CET Cond. (1)
	المالية (1 من 1 م	and the second	#0 Exercise (1) #0 Exercise (1) Trans (d)e
	for (i=0;i <cr></cr> ccfii++)		
	fil=0;		Function Block
	60 if (ps!=-1) if (calj]==ps) fit+=1	: else continue:	record_stats()
	if (pd=-1) if (ca(j)==pd) fit+=1	: else continue:	ant 1.): daulte 1.): 5 devendable inno-
	65 if (ca(j)==ns) fit+=1 if (nd!=-1)	: else continue:	contraction of the second seco
	if (ca[j]==nd) fit+=1	: else continue:	for (i=0: i <t1: i++)<br="">for (j=0; j<warekenguh_no: j++)<="" td=""></warekenguh_no:></t1:>
	if (fit>ap) 70 [as[0]=i; as[1]=calj if (an=4) ono assion]; ap=fit. }	10 if (slot(i)[j], src!=0) iune+=(op sim_itme()-slot(i)[j], ctime); cu=time stor_tor()(rev) sim_itmin(si): if (cu s-u un az) (n=ru-un az; op stat_wite statiat('to ad' und by stor_tor une(VP) op sim_itme());
) next_col: if (i!=(T1-1))		op_stat_write_scatadr**c=1*cup; op_stat_write_scatadr**c=1*cupbibliocked_pkf(doublebjotal_pk); op_stat_write_scatadr*pb=*cikobibliokickd_pkf(doublebjotal_pk);
	75 { ps=cs: pd=cd: cs=ns; cd= for (j=0;j <nf;j++) ca(j)="n</p"></nf;j++)>	ndı ci=nf. lalı):	-
			20 int find_resource(sc.dt.packet)
	if (ap=-1)		int sc.dt: Packet packet:
	ICULIE OF		

100	Model Repo	ort: mm1bl_scheduler	Fri Jan 7 03:36:12 1994	Page 5 of 7
빌	VCL Uniform er Model	Traffic System		2
	6	sim_end('reaching steady state'.'.'.'.');		
125	te 2:	idle (CET's)		
Π	Cond:	(default)		
~	Exec:			
:6	Cond:	(ENDSIM)		
-	Exec:	record_stats();		
	Trans:	idle		
Ξ	Cond:	(ARRIVAL)		
~	Exec:			
	Trans:	pk_prepare		
5	te 3	schadiila (Entar Evace)	forred	5 lines
9		Schedule (Eiliel Exers)	Inacial	2 III IES
	if (no_in_svc (blocked	== N1) _pk+=1: op_pk_destroy(pkpu);		
	if (!find_	[resource(src.dst.pkptr]) [pk+=1: blocked c .pk+=1; op .pk .destrov(nkptr);}		
3	te 3:	schedule (CET's)		
Ξ	Cond:	(1)		
•	Exec: Trans:			
5	ICUON BIOC	×	91 line	S
	record_stats()			
	t			

ate 3:	schedule (Enter Execs)	forced, 5 lines
if (no_in	$i_{\rm s} = NI$	
19) (P	{blocked_pk+=1: op_pk_destroy(pkptr): }	
else		
If (if (!find_resource(src.dst.pkptr))	
(blo	{blocked_pk+=1: blocked_c_pk+=1: op_pk_destroy(pkptr);}	

-	CUON BIOCK		91 lines
-	record_stats()		
	int		
	double	CU	
	extern double	time;	
	extern channel_asgn	slot(130)[10];	
	for (i=0: i<11 · i++)		
	for (internet of the second	and an it. ()	
	in you, you wavere if (shotfilli) si	usu_uo.jr+) uc'=0) time+=(on sim time()-shotfilfil ctime) [.]	
	cu=time*slot_no/(N*ol	p_sim_time()); if (cu > cu max) cu=cu max;	
	op_stat_write_scalar(*	<pre>loadtotal_pk*slot_no*svc_time/(N*op_sim_time()));</pre>	
	op_stat_write_scalar(".	cu".cu);	
	op_stat_write_scalar(*1	pb".(double)blocked_pk/(double)total_pk);	
	op_stat_write_scalar(*1	pbc .(double)blocked_c_pk/(double)total_pk);	
	_		
	int find_resource(sc.dt.packu	ct)	
	int sc.dt:		
	Packet* packet;		
-	-		



<u>*M/M/1/B/RL:*</u> OPNET reports for the *scheduler* and *release* processor models.



| W1/LPHL Uniform Traffic System 11 users 1 voor user 1 users 1 voor user edulier Model Objid voor uid 31 fines State Variables 31 fines 31 fines int W1.1 int Variables 5 int Variables 31 fines 10 int Variables 31 fines 11 int Variables 31 fines 12 benchulon ver uit Variables 13 benchulon ver uit Variables 14 benchulon ver uit Variables 15 Deterhulon ver dist benchulon 16 dooble ver uit ver dist 17 bengin un voor alst bengin un voor alst bendin un ver alst 18 Deterhulon ver dist bendin un voor alst bendin un ver alst 20 dooble ver dist bendin un ver alst bendin un voor alst 21 bendin un ver alst ver alst bendin un ver alst 22 bendin un ver alst bendin un ver alst bendin un ver alst 23 bendin un ver alst bendin un ver alst bendin un ver alst 24 bendin un ver alst bendin un ver alst

 | Millert Unform Traffic System
educer Model
educer Model
State Variables 31 lines 31 lines 31 lines 31 lines 31 lines 1 events to a 1 eque control
ter the transmission of the ter the transmission of the ter ter the ter ter ter ter ter ter ter ter ter te
 | n n n | N NW ALGS DELAND | | Erilan 7 03:31:37 1004 1 Pone 2 of | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

--
--
--

---|-------|------------------|------------------------|------------------------------------|---|--|---|-----|-----------|--|--|--|--|-----|-----------------|--|---|--|---|-----|-----------|--|---
---|--|-----|------|--|--|---|--|-----|----------|--|--|---|--|-----|----|--|---|--|--|----|-----|--|---|---|---|--------|------------|--
--|--|--|------|---------------|--|---|--|--|--------|------------|--|--|---|--|--------------|-------------|--
--|--|--|--------------|-------------|--|--|--|----|--------------|-------------|--|---|--|--|----------|------------|--|---|--|--|----------|--------------|--
--|--|--|----------|----------------|--|--|--|--|--------|----------|--|---|--|---|--------|----------|--|--
--|--|--------|-----------|--|---|---|--|--------|------------|--|---|---|--|--------|------------|--|--|--|--|--------|--------------|--|--
---|----|---|------------|--|---|--|--|-----|-----------|--|--|---|--|-----|------------------------|--|---|---|--|-----|-------------|--|---|--|--|-------|-------------|--|--|---|---|-----|--------------|--|-----------------------------|--|--|-----|---------|--|-----------------------------|---|--|--|--|--|--|--------------------------------|-----|---------------|------|---------|--|--|--|--|--|--|--|---|--|------------|------|--|
| State Variables 31 lines State Variables 31 lines Objid verside Int Variables Int verside Int Variables

 | State State 0 bild low_uit 1 bild low 1 bild bild 1 bild 1 bild
 | N1 | 3/RL Uniform T | Taffic System | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
| State Variables 31 lines 0 bjid trans. 10 bjid trans. 11 unde.unde.unde. trans. 12 unde.unde. trans. 13 bertibution trans. 14 unde.unde. trans. 15 bertibution trans. 16 double trans. 17 bertibution trans. 18 bertibution trans. 19 double trans. 10 double trans. 11 bertibution trans. 12 bertibution trans. 13 bertibution trans. 14 double trans. 15 bertibution trans. 16 double trans. 17 double trans. 18 bertibution trans. 19 double trans. 10 double trans. 11 double trans. 12 double trans. 13 double trans. 14 trans. trans. 15 int trans. 16 unde<

 | State Variables 31 lines Opid terr, di:
terr terr, di:
terr 0 pid terr, di:
terr terr, di:
terr 10 terr terr 11 terr terr 12 terr terr 13 terr terr 14 terr terr 15 terr terr 16 double terr, mit. 17 double terr, mit. 18 berthuion terr, dist. 19 double terr, mit. 11 double terr, mit. 12 double terr, mit. 13 Destribution
terr, dist. 14 Destribution terr, dist. 15 Destribution terr, dist. 16 double terr, mit. 17 terr terr, dist. 18 Destribution terr, dist. 19 terr terr, dist. 10 terr terr, dist. 11 terr terr, dist. 12 terr terr 13 terr terr 14 terr terr 15 terr | nedu | ler Model | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Objid two-let 10 bit two-dist 11 Ware-dist ware-dist 12 Ware-dist ware-dist 13 Distribution -wer_dist 14 Ware-dist ware-dist 15 Distribution -wer_dist 16 Distribution -wer_dist 17 Distribution -wer_dist 18 Distribution -wer_dist 19 Distribution -wer_dist 10 Mark dist Model 11 Distribution -wer_dist 12 Distribution -wer_dist 12 Distribution -wer_dist 12 Distribution -wer_dist 12 Dist -wer_dist

 | 09jd vow_it 1 undeit 1
 | Sta | te Variables | | 31 lines | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Cohjad wer_lei: in: Warenergeh_Jon; in: Warenergeh_Jon; in: Warenergeh_Jon; in: Warenergeh_Jon; in: Warenergeh_Jon; in: W1: in: W1:

 | 0 Opjid terc.jd: 1 in: Wareheigh_Jonc 1 in: Wareheigh_Jonc 1 in: Wareheigh_Jonc 1 in: Wareheigh_Jonc 1 in: Nin 1 doubt wareheigh_Jonc 1 bong int word_Jatc 1 <
 | | Objid | Nown_id: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| int Variance Variance Variance Variance Variance Deterblution Variance Variance Variance Deterblution Variance Variance Variance Deterblution Variance Variance Variance Deterblution Variance Variance Variance Longint Variance <tr td=""> Variance<td>int int worder-grip. 5 int Monte.net. int Min.str. Monte. int Monte. Monte. int Monte. Monte. int Nti. Monte. 10 doubte wer.uter. 11 doubte wer.uter. 12 Deterbinion wer.uter. 13 Deterbinion wer.uter. 14 long int blocket.plk: 15 Deterbinion wer.gist. 20 doobte wer.gist. 21 long int blocket.plk: 22 doobte wer.gist. 23 int wer.gist. 24 dootte wer.gist. 25 int wer.gist. 26 int wer.gist. 27 dootte wer.gist. 28 int wer.gist. 29 int wer.gist. 20 dootte wer.gist. 28 int <td< td=""><td></td><td>Objid</td><td>\src_id;</td><td></td></td<></td></tr> <tr><td>s int Warvelength_Dick int Num_init Nim_init int Nim_init Nim_init int Nim_init Nim_init int Nim_init Nim_init int Nim_init Niminit int Niminit Niminit int Niminit Niminit double Mar_andit Mar_andit big int Uncledic_pk: Niccled_pk: big int Uncledic_pk: Niccled_pk: big int Uncledic_pk: Niccled_pk: clouble Var_and: Niccled_pk: clouble Var_andi Niminininininininininininininininininini</td><td>5 int Warvelengt, Inc. 10 int With int. 10 doubt NV: 11 doubt NV: 12 doubt NV: 13 doubt NV: 14 doubt NV: 15 Derthholo Mat. dist. 16 doubt Vex_dist. 17 Derthholo Mat. dist. 18 Derthholo Mat. dist. 19 doubt Vex_dist. 11 borg int Wooded pt. 12 Derthholo Mat. dist. 13 Derthholo Mat. dist. 14 vex_dist. Mooded pt. 15 Derthholo Mat. dist. 20 doubt Vex_dist. 20 doubt Vex_dist. 21 doubt Vex_dist. 22 unt Vex_dist. 23 unt Vex_dist. 24 doubt Vex_dist. 25 unt Vex_dist. 26 unt Vex_dist. 27 doubt Vex_dist. 28 unt Vex_dist. 10 unt Vex_dist.</td><td>_</td><td>'nt</td><td>\node_no:</td><td></td></tr> <tr><td> 5 lint Vin_size: 10 double Vin_size: 10 double Vin_size: 10 double Vin_size: 13 Deterbution *ex_dist: 15 Deterbution *ex_dist: 16 double Vin_size: 17 Deterbution *ex_dist: 18 Deterbution *ex_dist: 19 bit Nocded_pit: 10 double Vin_max. 20 double Vin_max. 21 double Vin_size: 22 double Vin_size: 23 dist Variables 30 lint Vin_size: 31 lint Vin_size: 32 double Vin_max. 33 lint Vin_size: 34 double Vin_merce: 35 lint Vin_size: 36 lint Variables 37 fires </td><td>5 int Un_size: 10 int UT: 11 int N1: 12 doubte NN: 13 doubte NN: 14 N1: N1: 15 doubte NN: 16 doubte NN: 17 doubte NN: 18 bistribution *Nn. dist 19 bistribution *Nn. dist 11 Distribution *Nn. dist 12 Distribution *Nn. dist 13 Distribution *Nn. dist 14 Distribution *Nn. dist 15 Distribution *Nn. dist 16 Distribution *Nn. dist 17 Long int Would stat 18 Distribution *Nn. dist 19 Distribution *Nn. dist 20 double Ku, dist 21 double Ku, dist 22 double Ku, dist 23 int VL. dist 24 int VL. dist 25 int VL. dist 26 int VL. dist 27 double VL. dist</td><td></td><td>int</td><td>\wavelength_no;</td><td></td></tr> <tr><td>int VT1: int VT1: int NC int NC int NC int NC int NC char Nr Detribution Nr Sterbholon Nr Ver, dist Detribution Nr Ver, dist Detribution Nr Ver, dist Noted, pt. long int Vold, dist long int Vold, dist long int Vold, dist cooble Vp. donble Vp. long Ver, dist donble Vp. int Vc. cont: int Vc. cont: <t< td=""><td>int 111: int 111: int 111: int 111: char Nr.: char Nrmer. Char Nrmer. Dentholion Nr Lis Dentholion Dentholion Nr Dentholion Nr Dentholion Nr Dotted A Dotted A Dotted Nr Dotted P <td< td=""><td>S</td><td>int</td><td>Vfm_size:</td><td></td></td<></td></t<></td></tr> <tr><td>int blot.nox. 10 double vex.time. double vex.time. double vex.time. double vex.time. double vex.time. Destribution vex.dist. Destribution vex.dist. Destribution vex.dist. Double vex.dist. Double vex.dist. Double vex.dist. Log and vex.dist. Log</td><td>int bloc.noc. int NI: int NI: charbinicion wrv.mer; charbinicion wrv.mer; Destribution wrv.mer; Distribution wrv.dist; Distribution wrv.dist; Distribution wrv.mer; Distrin wrv.mer; <t< td=""><td></td><td>int</td><td>VTI:</td><td></td></t<></td></tr> <tr><td>10 double N: 10 double vsrimer. Distribution vsrimer. Dint vsrimer. Dint</td><td>10 double N:
clothe N:
clothe 13 Detribution *ec. dist:
brenthultion *ec. dist:
clothe 13 Detribution *ec. dist:
clothe 14 Detribution *ec. dist:
clothe 15 Detribution *ec. dist:
clothe 20 double Values:
clothe 21 Detribution *ec. dist:
clothe 22 double Values:
clothe 23 int bloc dec. plk:
clothe 24 double Values:
clothe 25 int bloc dec. plk:
clothe 26 double Values:
clothe 27 double Values:
clothe 28 int Values:
clothe 29 int Values:
clothe 20 double Values:
clothe 28 int Values:
clothe 29 int Values:
clothe 20 int Values:
clothe 21 int Values:
clothe 22 int Values:
clothe 23 int Values:
clothe 24 int Values:
clothe 25 int Values:
clothe 26 int Values:
clothe</td><td></td><td>int</td><td>klot_no:</td><td></td></tr> <tr><td>10 double W11: 11 double Wc. Jine: Destribution *wc. Jine: Destribution *wc. Jine: Distribution *wc. dist: Double to Jine: 20 double Voold *wc. dist: double to Jine: 20 double Voold to Jine: 21 double 11 Wc.const: 12 to Moded_prev: double to Moded_prev: double to Moded_prev: double to Moded_prev: 11 Wc.const: 11 to Moded_prev: 12 to Moded_prev: 13 to Moded_prev: 14 to Moded_prev: 15 to Moded_prev: 16 to Moded_prev:<td>10 dotte WN: 11 dotte WN: Detribution *ex. dat: Long int blocked_pt: Long int blocked_pt: Long int blocked_pt: Long obtic vol_pter: double vol_pter: int vol_pter: int vol_ater int</td><td></td><td>ji,</td><td>W:</td><td></td></td></tr> <tr><td>10 double lwc.time. 10 double lwc.rag(20); Distribution "dc.dis: lbs:rbiblion Distribution "dc.dis: lbs:rbiblion 15 Distribution "dc.dis: Distribution "dc.dis: lbs:rbiblion 16 Distribution "dc.dis: 17 Distribution "dc.dis: 18 Distribution "dc.dis: 19 long int Vold_sc. 10 uote tc.gat: 10 uote tc.gat: 10 uote tc.gat: 11 Word_sc.gr. tc.gat: 12 uote tc.gat: 11 Word_scat: tc.gat:</td><td>10 doubte wwtme.: 110 cdar wwme:: 111 Dentihulion *dedis: 111 Dentihulion *dedis: 112 Distribution *dedis: 113 Distribution *dedis: 114 Distribution *dedis: 115 Distribution *dedis: 116 trut blocked_ick 111 blocked_ick blocked_ick 111 block blocked_ick 111</td><td></td><td>Ďţ</td><td>NI:</td><td></td></tr> <tr><td>doubt wr_x_reg(20); Describution wr_x_reg(30); Distribution *ex. dist; Long int Wocket_pit; Lint Wood, stready; Lint Wood, stready; Lint Wood, stready; Lint Wood, stready; Lencorent Vartables 3 liftes</td><td>char un-rarg(20); Describution -un-rarg Distribution -un-raff Long int Noccher_pat Long int Noccher_pat Long int Noccher_pat Longle -un-raff Lint -un-raff<td>9</td><td>double</td><td>\svc_time;</td><td></td></td></tr> <tr><td>Distribution Var. Ant:
Constribution Var. Ant:
Var. Ant. 15 Distribution -Var. (dist.
Var. dist. 20 Distribution -Var. (dist.
Var. dist. 20 Distribution Var. (dist.
Var. dist. 20 Distribution Var. (dist.
Var. (double 21 Distribution Var. (dist.
(double 22 Distribution Var. (dist.
(double 23 Distribution Var. (dist.
(double 23 Distribution Var. (dist.
(double 23 Distribution Var. (double 24 Distribution Var. (double 25 Distribution Var. (double 26 Distribution Var. (double 27 Distribution Var. (double 28 Distribution Var. (double 29 Distribution Var. (double 29 Distribution Var. (double 29 Distribution Var. (double 20 Distribution Var. (double Distribution Var. (double <!--</td--><td>Berthaution ***. att:: Distribution ***. dist: Distribution ***. dist: Distribution ***. dist: Dorg int Vocal_sis: Long int Vocal_sis: Lint Voconi: <td< td=""><td></td><td>char</td><td>larv_arg[20];</td><td></td></td<></td></td></tr> <tr><td>Distribution "Acc. dist:
"Acc. dist: Distribution "Acc. dist:
"Acc. dist: Distribution "Acc. dist:
"Acc. dist: Distribution "Acc. dist:
"Acc. dist: Logg int Uncided_pic:
Nucleof_pic: Logg int Uncided_pic: Logg int Uncided_pic: Logg int Uncided_pic: Log unble Ver_merc; double Ver_merc; int Ver_merc; int Ver dor; int Ver dor;</td><td>Ise Distribution were, dist. Distribution were, dist. Ise Distribution were, dist. long int blocked_prix. double ver_prev. double ver_prev.</td><td></td><td>double</td><td>larv_rate:</td><td></td></tr> <tr><td> Discribution "Act, dist. Discribution "Act, dist.
long int blocked_pit. long int blocked_pit. long int blocked_pit. long int blocked_pit. double typ_prev. double typ_prev. double typ_prev. double typ_prev. double typ_prev. int blocked_pit. </td><td> Breithulion ver, dist. Distribution ver, dist. long int blocked_pkt. long int blocked_pkt. long int blocked_pkt. long int blocked_pkt. double ver_ner. double ver_ner.</td><td></td><td>Distribution</td><td>**src_dist:</td><td></td></tr> <tr><td> Distribution *eve. dist. Ibergint Notal_pit: Notal_pit: Notal_pit: Notal_pit: Notal_pit: Notext_1_pit: Notext_1_pit: Notext_1_pit: double tr_1_max; double tr_1_max</td><td>15 Distribution *ex. dist. Noted_pit: Noted_pit: Noted_pit: Noted_pit: 20 double *u_max: 20 double *u_max: 21 double *u_max: 22 double *u_max: 23 int *u_cont: 11 *u_cont: *u_max: 23 int *u_cont: 12 int *u_cont: 13 int *u_cont: 14 *u_ady: *u_ady: 15 int *u_ady: 16 *u_ady: *u_ady: 17 *u_ady: *u_ady: 18 *u_ady: *u_ady: 19 int *u_ady: 11 *u_ady: *u_ady: 12 *u_ady: *u_ady: 13 *u_ady: *u_ady: 14 *u_ady: *u_ady: 15 *u_ady: *u_ady: 16 *u_a</td><td></td><td>Distribution</td><td>*\dst_dist;</td><td></td></tr> <tr><td>Iong int Votal pi:
bong int Votal pi:
bong int Votal pi:
bong int Votal pi:
bonds 20 double Ver, prev:
double Ver, prev:
bonds Ver, prev:
bonds</td><td>Image int World pict. long int World pict. long int World car. long int World car. long int World car. long World car. double Wurg mer. double Wurg comm. int Wurg staady. int Wurg staady.</td><td>15</td><td>Distribution</td><td>*\svc_dist:</td><td></td></tr> <tr><td>Image int Uncated pit: Low point Vacator pit: Low book Vacator pit: Low book Vacator pit: double Vp_merc; double Vp_merc; double Vp_merc; double Vp_merc; double Vp_merc; double Vp_merc; low point Vp_merc; low point; Vp_merc; low point; Vp_merc; low prediv; Jintes</td><td>Image int Uncket JR:: 20 double VL_TPEY: int VL_Count: int VL_Count:</td><td></td><td>long int</td><td>Vuotal_pk:</td><td></td></tr> <tr><td>Ibidity Uncced.c.p.H: 20 double VL_max; int VL_count; int VL_count; int VL_count; <</td><td>Iboug int Uncced.c.pk; 20 double vc_max; int vc_max; int</td><td></td><td>long int</td><td>Vblocked_pk:</td><td></td></tr> <tr><td>20 double VL_max; 20 double VL_mer; int VL_count; int<td>20 double VL_mark: double VL_merk: int VL_count: int</td><td></td><td>long int</td><td>Vblocked_c_pk:</td><td></td></td></tr> <tr><td>20 double VL. prev: int VL. count: int VL. count: int VL. staady: int VD. staady:</td><td>20 double ku, prev:
double double kp, prev:
double double kp, rever:
double double kp, rever:
kit double kp, rever:
kit int kp, stady:
kit int kp, stady:
kit</td><td></td><td>double</td><td>kcu_max;</td><td></td></tr> <tr><td>double (v). prev:
double (v). prev:
double Voad. prev:
double Voad. prev:
double Voad. inne:
int Vo. graady:
int Vo. graady:
int Voad. sready:
int Voad. sre</td><td>double \u00ebuilty_prev: double \u00ebuilty_prev: int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev<</td><td>ន</td><td>double</td><td>ku_prev;</td><td></td></tr> <tr><td>double Upt. prev:
double Upt. prev:
Upt. prev: 23 double Uoral_time:
time Upt. prev:
Upt. Pre-stady:
time Upt. prev:
Upt. Upt. stady:
time Upt. prev:
Upt. Upt. stady:
time Upt. prev:
Upt. Upt. stady:
Upt. Upt. stady: Upt. prev:
Upt. Upt. Stady:
Upt. Upt. stady: Upt. Prev:
Upt. Prev:
Upt. Upt. Stady: Upt. Prev:
Upt. Pr</td><td>Couple Vpc.grev: 3 int Vpc.grev: 4 Vontile Vontile 1 V Vontil 1 Vontil Vontil</td><td></td><td>double</td><td>\pb_prev;</td><td></td></tr> <tr><td>double Uoal_inne; 25 int Voal_inne; int Voal_inne; Int int Vc_count; Int int Vc_staady; Int int Vb_staady; Int int Vb_staady; Int int Vb_staady; Int int Vb_staady; Int int Verady; Int int Verady; Int Verady; Int Verady;</td><td>South Woad, prev: 00mble Woad, prev: 1 Wacal, innec 1 Wacal, innec 1 Wacal, innec 1 Wacal, standy: 1 Wacal, standy:</td><td></td><td>double</td><td>ipbc_prev:</td><td></td></tr> <tr><td>South Vex.Linne: 1 Vex.connt: 1 Vex.connt: 1 Vex.connt: 1 Vex.standy: 1 Vex.standy:</td><td>Sample Counce to the count:
Int Vocal time:
but on seady:
int <thvocal time:<br="">int <!--</td--><td></td><td>double</td><td>Voad_prev:</td><td></td></thvocal></td></tr> <tr><td>25 int Va.count:
int Va.count:
int Va.count:
int Vpb.staady;
int Vba.staady;
int Vba.staady;
i</td><td>23 lut yb. count:
lut vc. count:
lut vc. staady:
lut vbe. staady:
lut voad staad</td><td></td><td>double</td><td>Viotal_time:</td><td></td></tr> <tr><td>int vr. staady:
int vr. staady:
int vpb. staady:
int vbad_staady:
int vbad_staady:
int vaady:
Teruporary Variables 3 lines</td><td>int v.r. count:
int v.r. count:
int vpb. steady:
int vpb. steady:
int vbad_steady:
int veady:
int v</td><td>25</td><td>Ĕ</td><td>\pk_count;</td><td></td></tr> <tr><td>int vu. steady:
int vpc. steady:
int vpc. steady:
int veady:
int veady:
Terrporary Variables 3 lines</td><td>int vu steady:
int ve steady:
int ve steady:
int ve steady:
int veedy:
Temborary Variables 3 ines
Pacter* phore:
int veedy:</td><td></td><td>int</td><td>Vr_count:</td><td></td></tr> <tr><td>int ybe_steady:
int ybe_steady:
int vead, steady:
Int veady:
Temporary Variables 3 lines</td><td>int yb. steady:
10 int ybe. steady:
11 veady:
12 remporeny Variables 3 lines
12 meter pour.
14 pour.</td><td></td><td>int</td><td><pre>\cu_stcady:</pre></td><td></td></tr> <tr><td>int yhe seady:
1 int Voad seady:
1 int Voad seady:
Terriporary Variables 3 lines</td><td>Int Whe.steady:
houd_steady: 10 Vood_steady: 11 Vood_steady: 12 Pecter* phone: 1 Pecter* phone: 1 int scredstij;</td><td></td><td>iet</td><td>\pb_steady:</td><td></td></tr> <tr><td>30 lint Voud.strady:
 int Verady:
Terriporary Variables 3 lines</td><td>30 int Voad_strady.
int veady;
Temporary Variables 3 lines
Packet" pom:
int sc.dat.it;</td><td></td><td>ji ji</td><td>pbc_steady.</td><td></td></tr> <tr><td>lint veady:
Temporary Variables 3 lines</td><td>lint veady:
Temporary Variables 3 lines
Packet pour
int secdst.j.;</td><td>ŝ</td><td>int</td><td>Voad_steady:</td><td></td></tr> <tr><td>Temporary Variables 3 lines</td><td>Temporary Variables 3 intes
Packet* piper:
int sectabil;</td><td></td><td>int</td><td>tready;</td><td></td></tr> <tr><td>Temporary Variables 3 lines</td><td>Temporary Variables 3 lines
Pactor* pipm:
int scdst.ij;</td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td>Packer* pkpr.
int srcdst.j;</td><td>Ten</td><td>noorary Varia</td><td>bles</td><td>3 lines</td></tr> <tr><td></td><td>Practices propert.
int sere.dst.i.jt;</td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td>-</td><td></td><td>INT SUCIAN</td><td>1.1;</td><td></td></tr> | int int worder-grip. 5 int Monte.net. int Min.str. Monte. int Monte. Monte. int Monte. Monte. int Nti. Monte. 10 doubte wer.uter. 11 doubte wer.uter. 12 Deterbinion wer.uter. 13 Deterbinion wer.uter. 14 long int blocket.plk: 15 Deterbinion wer.gist. 20 doobte wer.gist. 21 long int blocket.plk: 22 doobte wer.gist. 23 int wer.gist. 24 dootte wer.gist. 25 int wer.gist. 26 int wer.gist. 27 dootte wer.gist. 28 int wer.gist. 29 int wer.gist. 20 dootte wer.gist. 28 int <td< td=""><td></td><td>Objid</td><td>\src_id;</td><td></td></td<> | | Objid | \src_id; | | s int Warvelength_Dick int Num_init Nim_init int Nim_init Nim_init int Nim_init Nim_init int Nim_init Nim_init int Nim_init Niminit int Niminit Niminit int Niminit Niminit double Mar_andit Mar_andit big int Uncledic_pk: Niccled_pk: big int Uncledic_pk: Niccled_pk: big int Uncledic_pk: Niccled_pk: clouble Var_and: Niccled_pk: clouble Var_andi Niminininininininininininininininininini | 5 int Warvelengt, Inc. 10 int With int. 10 doubt NV: 11 doubt NV: 12 doubt NV: 13 doubt NV: 14 doubt NV: 15 Derthholo Mat. dist. 16 doubt Vex_dist. 17 Derthholo Mat. dist. 18 Derthholo Mat. dist. 19 doubt Vex_dist. 11 borg int Wooded pt. 12 Derthholo Mat. dist. 13 Derthholo Mat. dist. 14 vex_dist. Mooded pt. 15 Derthholo Mat. dist. 20 doubt Vex_dist. 20 doubt Vex_dist. 21 doubt Vex_dist. 22 unt Vex_dist. 23 unt Vex_dist. 24 doubt Vex_dist. 25 unt Vex_dist. 26 unt Vex_dist. 27 doubt Vex_dist. 28 unt Vex_dist. 10 unt Vex_dist. | _ | 'nt | \node_no: | | 5 lint Vin_size: 10 double Vin_size: 10 double Vin_size: 10 double Vin_size: 13 Deterbution *ex_dist: 15 Deterbution *ex_dist: 16 double Vin_size: 17 Deterbution *ex_dist: 18 Deterbution *ex_dist: 19 bit Nocded_pit: 10 double Vin_max. 20 double Vin_max. 21 double Vin_size: 22 double Vin_size: 23 dist Variables 30 lint Vin_size: 31 lint Vin_size: 32 double Vin_max. 33 lint Vin_size: 34 double Vin_merce: 35 lint Vin_size: 36 lint Variables 37 fires | 5 int Un_size: 10 int UT: 11 int N1: 12 doubte NN: 13 doubte NN: 14 N1: N1: 15 doubte NN: 16 doubte NN: 17 doubte NN: 18 bistribution *Nn. dist 19 bistribution *Nn. dist 11 Distribution *Nn. dist 12 Distribution *Nn. dist 13 Distribution *Nn. dist 14 Distribution *Nn. dist 15 Distribution *Nn. dist 16 Distribution *Nn. dist 17 Long int Would stat 18 Distribution *Nn. dist 19 Distribution *Nn. dist 20 double Ku, dist 21 double Ku, dist 22 double Ku, dist 23 int VL. dist 24 int VL. dist 25 int VL. dist 26 int VL. dist 27 double VL. dist | | int | \wavelength_no; | | int VT1: int VT1: int NC int NC int NC int NC int NC char Nr Detribution Nr Sterbholon Nr Ver, dist Detribution Nr Ver, dist Detribution Nr Ver, dist Noted, pt. long int Vold, dist long int Vold, dist long int Vold, dist cooble Vp. donble Vp. long Ver, dist donble Vp. int Vc. cont: int Vc. cont: <t< td=""><td>int 111: int 111: int 111: int 111: char Nr.: char Nrmer. Char Nrmer. Dentholion Nr Lis Dentholion Dentholion Nr Dentholion Nr Dentholion Nr Dotted A Dotted A Dotted Nr Dotted P <td< td=""><td>S</td><td>int</td><td>Vfm_size:</td><td></td></td<></td></t<> | int 111: int 111: int 111: int 111: char Nr.: char Nrmer. Char Nrmer. Dentholion Nr Lis Dentholion Dentholion Nr Dentholion Nr Dentholion Nr Dotted A Dotted A Dotted Nr Dotted P Dotted P <td< td=""><td>S</td><td>int</td><td>Vfm_size:</td><td></td></td<> | S | int | Vfm_size: | | int blot.nox. 10 double vex.time. double vex.time. double vex.time. double vex.time. double vex.time. Destribution vex.dist. Destribution vex.dist. Destribution vex.dist. Double vex.dist. Double vex.dist. Double vex.dist. Log and vex.dist. Log | int bloc.noc. int NI: int NI: charbinicion wrv.mer; charbinicion wrv.mer; Destribution wrv.mer; Distribution wrv.dist; Distribution wrv.dist; Distribution wrv.mer; Distrin wrv.mer; <t< td=""><td></td><td>int</td><td>VTI:</td><td></td></t<> | | int | VTI: | | 10 double N: 10 double vsrimer. Distribution vsrimer. Dint vsrimer. Dint | 10 double N:
clothe N:
clothe 13 Detribution *ec. dist:
brenthultion *ec. dist:
clothe 13 Detribution *ec. dist:
clothe 14 Detribution *ec. dist:
clothe 15 Detribution *ec. dist:
clothe 20 double Values:
clothe 21 Detribution *ec. dist:
clothe 22 double Values:
clothe 23 int bloc dec. plk:
clothe 24 double Values:
clothe 25 int bloc dec. plk:
clothe 26 double Values:
clothe 27 double Values:
clothe 28 int Values:
clothe 29 int Values:
clothe 20 double Values:
clothe 28 int Values:
clothe 29 int Values:
clothe 20 int Values:
clothe 21 int Values:
clothe 22 int Values:
clothe 23 int Values:
clothe 24 int Values:
clothe 25 int Values:
clothe 26 int Values:
clothe | | int | klot_no: | | 10 double W11: 11 double Wc. Jine: Destribution *wc. Jine: Destribution *wc. Jine: Distribution *wc. dist: Double to Jine: 20 double Voold *wc. dist: double to Jine: 20 double Voold to Jine: 21 double 11 Wc.const: 12 to Moded_prev: double to Moded_prev: double to Moded_prev: double to Moded_prev: 11 Wc.const: 11 to Moded_prev: 12 to Moded_prev: 13 to Moded_prev: 14 to Moded_prev: 15 to Moded_prev: 16 to Moded_prev: <td>10 dotte WN: 11 dotte WN: Detribution *ex. dat: Long int blocked_pt: Long int blocked_pt: Long int blocked_pt: Long obtic vol_pter: double vol_pter: int vol_pter: int vol_ater int</td> <td></td> <td>ji,</td> <td>W:</td> <td></td> | 10 dotte WN: 11 dotte WN: Detribution *ex. dat: Long int blocked_pt: Long int blocked_pt: Long int blocked_pt: Long obtic vol_pter: double vol_pter: int vol_pter: int vol_ater int | | ji, | W: | | 10 double lwc.time. 10 double lwc.rag(20); Distribution "dc.dis: lbs:rbiblion Distribution "dc.dis: lbs:rbiblion 15 Distribution "dc.dis: Distribution "dc.dis: lbs:rbiblion 16 Distribution "dc.dis: 17 Distribution "dc.dis: 18 Distribution "dc.dis: 19 long int Vold_sc. 10 uote tc.gat: 10 uote tc.gat: 10 uote tc.gat: 11 Word_sc.gr. tc.gat: 12 uote tc.gat: 11 Word_scat: tc.gat: | 10 doubte wwtme.: 110 cdar wwme:: 111 Dentihulion *dedis: 111 Dentihulion *dedis: 112 Distribution *dedis: 113 Distribution *dedis: 114 Distribution *dedis: 115 Distribution *dedis: 116 trut blocked_ick 111 blocked_ick blocked_ick 111 block blocked_ick 111 | | Ďţ | NI: | | doubt wr_x_reg(20); Describution wr_x_reg(30); Distribution *ex. dist; Long int Wocket_pit; Lint Wood, stready; Lint Wood, stready; Lint Wood, stready; Lint Wood, stready; Lencorent Vartables 3 liftes | char un-rarg(20); Describution -un-rarg Distribution -un-raff Long int Noccher_pat Long int Noccher_pat Long int Noccher_pat Longle -un-raff Lint -un-raff <td>9</td> <td>double</td> <td>\svc_time;</td> <td></td> | 9 | double | \svc_time; | | Distribution Var. Ant:
Constribution Var. Ant:
Var. Ant. 15 Distribution -Var. (dist.
Var. dist. 20 Distribution -Var. (dist.
Var. dist. 20 Distribution Var. (dist.
Var. dist. 20 Distribution Var. (dist.
Var. (double 21 Distribution Var. (dist.
(double 22 Distribution Var. (dist.
(double 23 Distribution Var. (dist.
(double 23 Distribution Var. (dist.
(double 23 Distribution Var. (double 24 Distribution Var. (double 25 Distribution Var. (double 26 Distribution Var. (double 27 Distribution Var. (double 28 Distribution Var. (double 29 Distribution Var. (double 29 Distribution Var. (double 29 Distribution Var. (double 20 Distribution Var. (double Distribution Var. (double </td <td>Berthaution ***. att:: Distribution ***. dist: Distribution ***. dist: Distribution ***. dist: Dorg int Vocal_sis: Long int Vocal_sis: Lint Voconi: <td< td=""><td></td><td>char</td><td>larv_arg[20];</td><td></td></td<></td> | Berthaution ***. att:: Distribution ***. dist: Distribution ***. dist: Distribution ***. dist: Dorg int Vocal_sis: Long int Vocal_sis: Lint Voconi: <td< td=""><td></td><td>char</td><td>larv_arg[20];</td><td></td></td<> | | char | larv_arg[20]; | | Distribution "Acc. dist:
"Acc. dist: Distribution "Acc. dist:
"Acc. dist: Distribution "Acc. dist:
"Acc. dist: Distribution "Acc. dist:
"Acc. dist: Logg int Uncided_pic:
Nucleof_pic: Logg int Uncided_pic: Logg int Uncided_pic: Logg int Uncided_pic: Log unble Ver_merc; double Ver_merc; int Ver_merc; int Ver dor; int Ver dor; | Ise Distribution were, dist. Distribution were, dist. Ise Distribution were, dist. long int blocked_prix. double ver_prev. double ver_prev. | | double | larv_rate: | | Discribution "Act, dist. Discribution "Act, dist.
long int blocked_pit. long int blocked_pit. long int blocked_pit. long int blocked_pit. double typ_prev. double typ_prev. double typ_prev. double typ_prev. double typ_prev. int blocked_pit. | Breithulion ver, dist. Distribution ver, dist. long int blocked_pkt. long int blocked_pkt. long int blocked_pkt. long int blocked_pkt. double ver_ner. double ver_ner. | | Distribution | **src_dist: | | Distribution *eve. dist. Ibergint Notal_pit: Notal_pit: Notal_pit: Notal_pit: Notal_pit: Notext_1_pit: Notext_1_pit: Notext_1_pit: double tr_1_max; double tr_1_max | 15 Distribution *ex. dist. Noted_pit: Noted_pit: Noted_pit: Noted_pit: 20 double *u_max: 20 double *u_max: 21 double *u_max: 22 double *u_max: 23 int *u_cont: 11 *u_cont: *u_max: 23 int *u_cont: 12 int *u_cont: 13 int *u_cont: 14 *u_ady: *u_ady: 15 int *u_ady: 16 *u_ady: *u_ady: 17 *u_ady: *u_ady: 18 *u_ady: *u_ady: 19 int *u_ady: 11 *u_ady: *u_ady: 12 *u_ady: *u_ady: 13 *u_ady: *u_ady: 14 *u_ady: *u_ady: 15 *u_ady: *u_ady: 16 *u_a | | Distribution | *\dst_dist; | | Iong int Votal pi:
bong int Votal pi:
bong int Votal pi:
bong int Votal pi:
bonds 20 double Ver, prev:
double Ver, prev:
bonds Ver, prev:
bonds | Image int World pict. long int World pict. long int World car. long int World car. long int World car. long World car. double Wurg mer. double Wurg comm. int Wurg staady. int Wurg staady. | 15 | Distribution | *\svc_dist: | | Image int Uncated pit: Low point Vacator pit: Low book Vacator pit: Low book Vacator pit: double Vp_merc; double Vp_merc; double Vp_merc; double Vp_merc; double Vp_merc; double Vp_merc; low point Vp_merc; low point; Vp_merc; low point; Vp_merc; low prediv; Jintes | Image int Uncket JR:: 20 double VL_TPEY: int VL_Count: int VL_Count: | | long int | Vuotal_pk: | | Ibidity Uncced.c.p.H: 20 double VL_max; int VL_count; int VL_count; int VL_count; < | Iboug int Uncced.c.pk; 20 double vc_max; int vc_max; int | | long int | Vblocked_pk: | | 20 double VL_max; 20 double VL_mer; int VL_count; int <td>20 double VL_mark: double VL_merk: int VL_count: int</td> <td></td> <td>long int</td> <td>Vblocked_c_pk:</td> <td></td> | 20 double VL_mark: double VL_merk: int VL_count: int | | long int | Vblocked_c_pk: | | 20 double VL. prev: int VL. count: int VL. count: int VL. staady: int VD. staady: | 20 double ku, prev:
double double kp, prev:
double double kp, rever:
double double kp, rever:
kit double kp, rever:
kit int kp, stady:
kit int kp, stady:
kit | | double | kcu_max; | | double (v). prev:
double (v). prev:
double Voad. prev:
double Voad. prev:
double Voad. inne:
int Vo. graady:
int Vo. graady:
int Voad. sready:
int Voad. sre | double \u00ebuilty_prev: int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev< | ន | double | ku_prev; | | double Upt. prev:
double Upt. prev:
Upt. prev: 23 double Uoral_time:
time Upt. prev:
Upt. Pre-stady:
time Upt. prev:
Upt. Upt. stady:
time Upt. prev:
Upt. Upt. stady:
time Upt. prev:
Upt. Upt. stady:
Upt. Upt. stady: Upt. prev:
Upt. Upt. Stady:
Upt. Upt. stady: Upt. Prev:
Upt. Prev:
Upt. Upt. Stady: Upt. Prev:
Upt. Pr | Couple Vpc.grev: 3 int Vpc.grev: 4 Vontile Vontile 1 V Vontil 1 Vontil Vontil | | double | \pb_prev; | | double Uoal_inne; 25 int Voal_inne; int Voal_inne; Int int Vc_count; Int int Vc_staady; Int int Vb_staady; Int int Vb_staady; Int int Vb_staady; Int int Vb_staady; Int int Verady; Int int Verady; Int Verady; Int Verady; | South Woad, prev: 00mble Woad, prev: 1 Wacal, innec 1 Wacal, innec 1 Wacal, innec 1 Wacal, standy: 1 Wacal, standy: | | double | ipbc_prev: | | South Vex.Linne: 1 Vex.connt: 1 Vex.connt: 1 Vex.connt: 1 Vex.standy: 1 Vex.standy: | Sample Counce to the count:
Int Vocal time:
but on seady:
int Vocal time:
but on seady:
int <thvocal time:<br="">int <!--</td--><td></td><td>double</td><td>Voad_prev:</td><td></td></thvocal> | | double | Voad_prev: | | 25 int Va.count:
int Va.count:
int Va.count:
int Vpb.staady;
int Vba.staady;
int Vba.staady;
i | 23 lut yb. count:
lut vc. count:
lut vc. staady:
lut vbe. staady:
lut voad staad | | double | Viotal_time: | | int vr. staady:
int vr. staady:
int vpb. staady:
int vbad_staady:
int vbad_staady:
int vaady:
Teruporary Variables 3 lines | int v.r. count:
int v.r. count:
int vpb. steady:
int vpb. steady:
int vbad_steady:
int veady:
int v | 25 | Ĕ | \pk_count; | | int vu. steady:
int vpc. steady:
int vpc. steady:
int veady:
int veady:
Terrporary Variables 3 lines | int vu steady:
int ve steady:
int ve steady:
int ve steady:
int veedy:
Temborary Variables 3 ines
Pacter* phore:
int veedy: | | int | Vr_count: | | int ybe_steady:
int ybe_steady:
int vead, steady:
Int veady:
Temporary Variables 3 lines | int yb. steady:
10 int ybe. steady:
11 veady:
12 remporeny Variables 3 lines
12 meter pour.
14 pour. | | int | <pre>\cu_stcady:</pre> | | int yhe seady:
1 int Voad seady:
1 int Voad seady:
Terriporary Variables 3 lines | Int Whe.steady:
houd_steady: 10 Vood_steady: 11 Vood_steady: 12 Pecter* phone: 1 Pecter* phone: 1 int scredstij; | | iet | \pb_steady: | | 30 lint Voud.strady:
 int Verady:
Terriporary Variables 3 lines | 30 int Voad_strady.
int veady;
Temporary Variables 3 lines
Packet" pom:
int sc.dat.it; | | ji ji | pbc_steady. | | lint veady:
Temporary Variables 3 lines | lint veady:
Temporary Variables 3 lines
Packet pour
int secdst.j.; | ŝ | int | Voad_steady: | | Temporary Variables 3 lines | Temporary Variables 3 intes
Packet* piper:
int sectabil; | | int | tready; | | Temporary Variables 3 lines | Temporary Variables 3 lines
Pactor* pipm:
int scdst.ij; | | | | | | Packer* pkpr.
int srcdst.j; | Ten | noorary Varia | bles | 3 lines | | Practices propert.
int sere.dst.i.jt; | | | | | | - | | INT SUCIAN | 1.1; | |
| int int worder-grip. 5 int Monte.net. int Min.str. Monte. int Monte. Monte. int Monte. Monte. int Nti. Monte. 10 doubte wer.uter. 11 doubte wer.uter. 12 Deterbinion wer.uter. 13 Deterbinion wer.uter. 14 long int blocket.plk: 15 Deterbinion wer.gist. 20 doobte wer.gist. 21 long int blocket.plk: 22 doobte wer.gist. 23 int wer.gist. 24 dootte wer.gist. 25 int wer.gist. 26 int wer.gist. 27 dootte wer.gist. 28 int wer.gist. 29 int wer.gist. 20 dootte wer.gist. 28 int <td< td=""><td></td><td>Objid</td><td>\src_id;</td><td></td></td<>

 |
 | Objid | \src_id; | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| s int Warvelength_Dick int Num_init Nim_init int Nim_init Nim_init int Nim_init Nim_init int Nim_init Nim_init int Nim_init Niminit int Niminit Niminit int Niminit Niminit double Mar_andit Mar_andit big int Uncledic_pk: Niccled_pk: big int Uncledic_pk: Niccled_pk: big int Uncledic_pk: Niccled_pk: clouble Var_and: Niccled_pk: clouble Var_andi Niminininininininininininininininininini

 | 5 int Warvelengt, Inc. 10 int With int. 10 doubt NV: 11 doubt NV: 12 doubt NV: 13 doubt NV: 14 doubt NV: 15 Derthholo Mat. dist. 16 doubt Vex_dist. 17 Derthholo Mat. dist. 18 Derthholo Mat. dist. 19 doubt Vex_dist. 11 borg int Wooded pt. 12 Derthholo Mat. dist. 13 Derthholo Mat. dist. 14 vex_dist. Mooded pt. 15 Derthholo Mat. dist. 20 doubt Vex_dist. 20 doubt Vex_dist. 21 doubt Vex_dist. 22 unt Vex_dist. 23 unt Vex_dist. 24 doubt Vex_dist. 25 unt Vex_dist. 26 unt Vex_dist. 27
doubt Vex_dist. 28 unt Vex_dist. 10 unt Vex_dist. | _ | 'nt | \node_no: | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 lint Vin_size: 10 double Vin_size: 10 double Vin_size: 10 double Vin_size: 13 Deterbution *ex_dist: 15 Deterbution *ex_dist: 16 double Vin_size: 17 Deterbution *ex_dist: 18 Deterbution *ex_dist: 19 bit Nocded_pit: 10 double Vin_max. 20 double Vin_max. 21 double Vin_size: 22 double Vin_size: 23 dist Variables 30 lint Vin_size: 31 lint Vin_size: 32 double Vin_max. 33 lint Vin_size: 34 double Vin_merce: 35 lint Vin_size: 36 lint Variables 37 fires

 | 5 int Un_size: 10 int UT: 11 int N1: 12 doubte NN: 13 doubte NN: 14 N1: N1: 15 doubte NN: 16 doubte NN: 17 doubte NN: 18 bistribution *Nn. dist 19 bistribution *Nn. dist 11 Distribution *Nn. dist 12
 Distribution *Nn. dist 13 Distribution *Nn. dist 14 Distribution *Nn. dist 15 Distribution *Nn. dist 16 Distribution *Nn. dist 17 Long int Would stat 18 Distribution *Nn. dist 19 Distribution *Nn. dist 20 double Ku, dist 21 double Ku, dist 22 double Ku, dist 23 int VL. dist 24 int VL. dist 25 int VL. dist 26 int VL. dist 27 double VL. dist | | int | \wavelength_no; | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| int VT1: int VT1: int NC int NC int NC int NC int NC char Nr Detribution Nr Sterbholon Nr Ver, dist Detribution Nr Ver, dist Detribution Nr Ver, dist Noted, pt. long int Vold, dist long int Vold, dist long int Vold, dist cooble Vp. donble Vp. long Ver, dist donble Vp. int Vc. cont: int Vc. cont: <t< td=""><td>int 111: int 111: int 111: int 111: char Nr.: char Nrmer. Char Nrmer. Dentholion Nr Lis Dentholion Dentholion Nr Dentholion Nr Dentholion Nr Dotted A Dotted A Dotted Nr Dotted P <td< td=""><td>S</td><td>int</td><td>Vfm_size:</td><td></td></td<></td></t<>

 | int 111: int 111: int 111: int 111: char Nr.: char Nrmer. Char Nrmer. Dentholion Nr Lis Dentholion Dentholion Nr Dentholion Nr Dentholion Nr Dotted A Dotted A Dotted Nr Dotted P Dotted P <td< td=""><td>S</td><td>int</td><td>Vfm_size:</td><td></td></td<>
 | S | int | Vfm_size: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| int blot.nox. 10 double vex.time. double vex.time. double vex.time. double vex.time. double vex.time. Destribution vex.dist. Destribution vex.dist. Destribution vex.dist. Double vex.dist. Double vex.dist. Double vex.dist. Log and vex.dist. Log

 | int bloc.noc. int NI: int NI: charbinicion wrv.mer; charbinicion wrv.mer; Destribution wrv.mer; Distribution wrv.dist; Distribution wrv.dist; Distribution wrv.mer; Distrin wrv.mer; <t< td=""><td></td><td>int</td><td>VTI:</td><td></td></t<>
 | | int | VTI: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
| 10 double N: 10 double vsrimer. Distribution vsrimer. Dint vsrimer. Dint

 | 10 double N:
clothe N:
clothe 13 Detribution *ec. dist:
brenthultion *ec. dist:
clothe 13 Detribution *ec. dist:
clothe 14 Detribution *ec. dist:
clothe 15 Detribution *ec. dist:
clothe 20 double Values:
clothe 21 Detribution *ec. dist:
clothe 22 double Values:
clothe 23 int bloc dec. plk:
clothe 24 double Values:
clothe 25 int bloc dec. plk:
clothe 26 double Values:
clothe 27 double Values:
clothe 28 int Values:
clothe 29 int Values:
clothe 20 double Values:
clothe 28 int Values:
clothe 29 int Values:
clothe 20 int Values:
clothe 21 int Values:
clothe 22 int Values:
clothe 23 int Values:
clothe 24 int Values:
clothe 25 int Values:
clothe 26 int Values:
clothe
 | | int | klot_no: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 double W11: 11 double Wc. Jine: Destribution *wc. Jine: Destribution *wc. Jine: Distribution *wc. dist: Double to Jine: 20 double Voold *wc. dist: double to Jine: 20 double Voold to Jine: 21 double 11 Wc.const: 12 to Moded_prev: double to Moded_prev: double to Moded_prev: double to Moded_prev: 11 Wc.const: 11 to Moded_prev: 12 to Moded_prev: 13 to Moded_prev: 14 to Moded_prev: 15 to Moded_prev: 16 to Moded_prev: <td>10 dotte WN: 11 dotte WN: Detribution *ex. dat: Long int blocked_pt: Long int blocked_pt: Long int blocked_pt: Long obtic vol_pter: double vol_pter: int vol_pter: int vol_ater int</td> <td></td> <td>ji,</td> <td>W:</td> <td></td>

 | 10 dotte WN: 11 dotte WN: Detribution *ex. dat: Long int blocked_pt: Long int blocked_pt: Long int blocked_pt: Long obtic vol_pter: double vol_pter: int vol_pter: int vol_ater int
 | | ji, | W: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 double lwc.time. 10 double lwc.rag(20); Distribution "dc.dis: lbs:rbiblion Distribution "dc.dis: lbs:rbiblion 15 Distribution "dc.dis: Distribution "dc.dis: lbs:rbiblion 16 Distribution "dc.dis: 17 Distribution "dc.dis: 18 Distribution "dc.dis: 19 long int Vold_sc. 10 uote tc.gat: 10 uote tc.gat: 10 uote tc.gat: 11 Word_sc.gr. tc.gat: 12 uote tc.gat: 11 Word_scat: tc.gat:

 | 10 doubte wwtme.: 110 cdar wwme:: 111 Dentihulion *dedis: 111 Dentihulion *dedis: 112 Distribution *dedis: 113 Distribution *dedis: 114 Distribution *dedis: 115 Distribution *dedis: 116 trut blocked_ick 111 blocked_ick blocked_ick 111 block blocked_ick 111
 | | Ďţ | NI: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| doubt wr_x_reg(20); Describution wr_x_reg(30); Distribution *ex. dist; Long int Wocket_pit; Lint Wood, stready; Lint Wood, stready; Lint Wood, stready; Lint Wood, stready; Lencorent Vartables 3 liftes

 | char un-rarg(20); Describution -un-rarg Distribution -un-raff Long int Noccher_pat Long int Noccher_pat Long int Noccher_pat Longle -un-raff Lint -un-raff <td>9</td> <td>double</td> <td>\svc_time;</td> <td></td>
 | 9 | double | \svc_time; | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Distribution Var. Ant:
Constribution Var. Ant:
Var. Ant. 15 Distribution -Var. (dist.
Var. dist. 20 Distribution -Var. (dist.
Var. dist. 20 Distribution Var. (dist.
Var. dist. 20 Distribution Var. (dist.
Var. (double 21 Distribution Var. (dist.
(double 22 Distribution Var. (dist.
(double 23 Distribution Var. (dist.
(double 23 Distribution Var. (dist.
(double 23 Distribution Var. (double 24 Distribution Var. (double 25 Distribution Var. (double 26 Distribution Var. (double 27 Distribution Var. (double 28 Distribution Var. (double 29 Distribution Var. (double 29 Distribution Var. (double 29 Distribution Var. (double 20 Distribution Var. (double Distribution Var. (double </td <td>Berthaution ***. att:: Distribution ***. dist: Distribution ***. dist: Distribution ***. dist: Dorg int Vocal_sis: Long int Vocal_sis: Lint Voconi: <td< td=""><td></td><td>char</td><td>larv_arg[20];</td><td></td></td<></td>

 | Berthaution ***. att:: Distribution ***. dist: Distribution ***. dist: Distribution ***. dist: Dorg int Vocal_sis: Long int Vocal_sis: Lint Voconi: <td< td=""><td></td><td>char</td><td>larv_arg[20];</td><td></td></td<>
 | | char | larv_arg[20]; | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | |
| Distribution "Acc. dist:
"Acc. dist: Distribution "Acc. dist:
"Acc. dist: Distribution "Acc. dist:
"Acc. dist: Distribution "Acc. dist:
"Acc. dist: Logg int Uncided_pic:
Nucleof_pic: Logg int Uncided_pic: Logg int Uncided_pic: Logg int Uncided_pic: Log unble Ver_merc; double Ver_merc; int Ver_merc; int Ver dor;

 | Ise Distribution were, dist. Distribution were, dist. Ise Distribution were, dist. long int blocked_prix. double ver_prev.
 | | double | larv_rate: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | |
| Discribution "Act, dist. Discribution "Act, dist.
long int blocked_pit. long int blocked_pit. long int blocked_pit. long int blocked_pit. double typ_prev. double typ_prev. double typ_prev. double typ_prev. double typ_prev. int blocked_pit.

 | Breithulion ver, dist. Distribution ver, dist. long int blocked_pkt. long int blocked_pkt. long int blocked_pkt. long int blocked_pkt. double ver_ner. double ver_ner. | | Distribution | **src_dist: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Distribution *eve. dist. Ibergint Notal_pit: Notal_pit: Notal_pit: Notal_pit: Notal_pit: Notext_1_pit: Notext_1_pit: Notext_1_pit: double tr_1_max; double tr_1_max

 | 15 Distribution *ex. dist. Noted_pit: Noted_pit: Noted_pit: Noted_pit: 20 double *u_max: 20 double *u_max: 21 double *u_max: 22 double *u_max: 23 int *u_cont: 11 *u_cont: *u_max: 23 int *u_cont: 12 int *u_cont: 13 int *u_cont: 14 *u_ady: *u_ady: 15 int *u_ady: 16 *u_ady: *u_ady: 17 *u_ady: *u_ady: 18 *u_ady: *u_ady: 19 int *u_ady: 11 *u_ady: *u_ady: 12 *u_ady: *u_ady: 13 *u_ady: *u_ady: 14 *u_ady: *u_ady: 15 *u_ady: *u_ady: 16 *u_a
 | | Distribution | *\dst_dist; | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Iong int Votal pi:
bong int Votal pi:
bong int Votal pi:
bong int Votal pi:
bonds 20 double Ver, prev:
double Ver, prev:
bonds

 | Image int World pict. long int World pict. long int World car. long int World car. long int World car. long World car. double Wurg mer. double Wurg comm. int Wurg staady.
 | 15 | Distribution | *\svc_dist: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Image int Uncated pit: Low point Vacator pit: Low book Vacator pit: Low book Vacator pit: double Vp_merc; double Vp_merc; double Vp_merc; double Vp_merc; double Vp_merc; double Vp_merc; low point Vp_merc; low point; Vp_merc; low point; Vp_merc; low prediv; Jintes

 | Image int Uncket JR:: 20 double VL_TPEY: int VL_Count:
 | | long int | Vuotal_pk: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Ibidity Uncced.c.p.H: 20 double VL_max; int VL_count; int VL_count; int VL_count; <

 | Iboug int Uncced.c.pk; 20 double vc_max; int
 | | long int | Vblocked_pk: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 double VL_max; 20 double VL_mer; int VL_count; int <td>20 double VL_mark: double VL_merk: int VL_count: int</td> <td></td> <td>long int</td> <td>Vblocked_c_pk:</td> <td></td>

 | 20 double VL_mark: double VL_merk: int VL_count: int | | long int | Vblocked_c_pk: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 double VL. prev: int VL. count: int VL. count: int VL. staady: int VD. staady:

 | 20 double ku, prev:
double double kp, prev:
double double kp, rever:
double double kp, rever:
kit double kp, rever:
kit int kp, stady:
kit
 | | double | kcu_max; | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| double (v). prev:
double (v). prev:
double Voad. prev:
double Voad. prev:
double Voad. inne:
int Vo. graady:
int Vo. graady:
int Voad. sready:
int Voad. sre

 | double \u00ebuilty_prev: int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev int \u00ebuilty_prevev< | ន | double | ku_prev; | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| double Upt. prev:
double Upt. prev:
Upt. prev: 23 double Uoral_time:
time Upt. prev:
Upt. Pre-stady:
time Upt. prev:
Upt. Upt. stady:
time Upt. prev:
Upt. Upt. stady:
time Upt. prev:
Upt. Upt. stady:
Upt. Upt. stady: Upt. prev:
Upt. Upt. Stady:
Upt. Upt. stady: Upt. Prev:
Upt. Prev:
Upt. Upt. Stady: Upt. Prev:
Upt. Pr

 | Couple Vpc.grev: 3 int Vpc.grev: 4 Vontile Vontile 1 V Vontil 1 Vontil Vontil | | double | \pb_prev; | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| double Uoal_inne; 25 int Voal_inne; int Voal_inne; Int int Vc_count; Int int Vc_staady; Int int Vb_staady; Int int Vb_staady; Int int Vb_staady; Int int Vb_staady; Int int Verady; Int int Verady; Int Verady; Int Verady;

 | South Woad, prev: 00mble Woad, prev: 1 Wacal, innec 1 Wacal, innec 1 Wacal, innec 1 Wacal, standy:
 | | double | ipbc_prev: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| South Vex.Linne: 1 Vex.connt: 1 Vex.connt: 1 Vex.connt: 1 Vex.standy:

 | Sample Counce to the count:
Int Vocal time:
but on seady:
int Vocal time:
but on seady:
int <thvocal time:<br="">int <!--</td--><td></td><td>double</td><td>Voad_prev:</td><td></td></thvocal>
 | | double | Voad_prev: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 25 int Va.count:
int Va.count:
int Va.count:
int Vpb.staady;
int Vba.staady;
int Vba.staady;
i

 | 23 lut yb. count:
lut vc. count:
lut vc. staady:
lut vbe. staady:
lut voad staad | | double | Viotal_time: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| int vr. staady:
int vr. staady:
int vpb. staady:
int vbad_staady:
int vbad_staady:
int vaady:
Teruporary Variables 3 lines

 | int v.r. count:
int v.r. count:
int vpb. steady:
int vpb. steady:
int vbad_steady:
int veady:
int v | 25 | Ĕ | \pk_count; | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| int vu. steady:
int vpc. steady:
int vpc. steady:
int veady:
int veady:
Terrporary Variables 3 lines

 | int vu steady:
int ve steady:
int ve steady:
int ve steady:
int veedy:
Temborary Variables 3 ines
Pacter* phore:
int veedy:
 | | int | Vr_count: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| int ybe_steady:
int ybe_steady:
int vead, steady:
Int veady:
Temporary Variables 3 lines

 | int yb. steady:
10 int ybe. steady:
11 veady:
12 remporeny Variables 3 lines
12 meter pour.
14 pour.
 | | int | <pre>\cu_stcady:</pre> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| int yhe seady:
1 int Voad seady:
1 int Voad seady:
Terriporary Variables 3 lines

 | Int Whe.steady:
houd_steady: 10 Vood_steady: 11 Vood_steady: 12 Pecter* phone: 1 Pecter* phone: 1 int scredstij;
 | | iet | \pb_steady: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30 lint Voud.strady:
 int Verady:
Terriporary Variables 3 lines

 | 30 int Voad_strady.
int veady;
Temporary Variables 3 lines
Packet" pom:
int sc.dat.it;
 | | ji ji | pbc_steady. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| lint veady:
Temporary Variables 3 lines

 | lint veady:
Temporary Variables 3 lines
Packet pour
int secdst.j.;
 | ŝ | int | Voad_steady: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Temporary Variables 3 lines

 | Temporary Variables 3 intes
Packet* piper:
int sectabil;
 | | int | tready; | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Temporary Variables 3 lines

 | Temporary Variables 3 lines
Pactor* pipm:
int scdst.ij;
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|

 | Packer* pkpr.
int srcdst.j;
 | Ten | noorary Varia | bles | 3 lines | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|

 | Practices propert.
int sere.dst.i.jt;
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|

 | -
 | | INT SUCIAN | 1.1; | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | |
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

in roce	e Model Report: mm1hr schadiiilar	Eri Ian 7 n3:31:38 1004 Dage 3 of 10
	o mouto nepon. man ou senerance	111 Jail 1 10001.00 1304 1 496 0 01 10
chedu	art Unitorim I rame System lier Model	
Į		
3	ite 0: Init (Enter Execs)	forced, 29 lines
	op Jma. sim. attr. get(OPC, IMA, INTEGER, 'N. &node_no); op. Jma. sim. attr. get(OPC, IMA, INTEGER, 'N. & wwwlength_no); op. Jma. sim. attr. get(OPC, IMA, INTEGER, 'N. & & with. aix2); op. Jma. sim. attr. get(OPC, IMA, INTEGER, 'N. & & & op.);	
s	own_id=0p_id_self(); src_id=0p_tppo_in_assoc(own_id_0); op_inna_obj_attr_get(src_id_tincerarrival args'&ar_arg(0));	
5	arv_ratc=atof(&arv_arg[0]);	
2	N=wavelength_no*fin_size: svc_time=(double)N/(double)slot_no: N1=floor(svc_time); svc_time=(double)fin_size/(double)slot_no:	
15	sve disterp dist load(* exponential: ave.time.0); sre disterp dist load(* unitern_int*1.(doubb/mode_no); dist disterp dist load(* unitern_int*1.(doubb/mode_no);	
20	ונוטון באבלו, אוסבובת, באבלו, אוסבובתו ב. באברול, הורוצאמר באבלו, ונוס באי כנו השגיע האוסר באיר השגיע באברול באברו באברו באברו באברו באברו באברו באברו כנו בגדמלו באלו איני באבולו באיר אוסב באבמלו בילו, ונוסעו באברו באילו באלו	at, time=t, wavelength_no:
	for (i=0;i<=N1;i++)	
52	pk(i)) sre-0; pk(i) ks=0; pk(i) ks=0; pk(i) crime=0; for (j=0; j<15; j++) {otherf[1]() stot=0; other[1](); wavelength=0; other[1](], ks=0; }	
	for (i=0;i <fm_sizz;i++) for (i=0;i<wavelength_no;i++) slot(i][i]="0:</td"><td></td></wavelength_no;i++)></fm_sizz;i++) 	
St	tte 0: Init (CET's)	
8	Cond: (ARRIVAL)	
2	Exec: Trans: pk. prepare	
<u>n z</u>	T Cond: (dcfault) Exec: : Trans: idle	
Sb	ite 1: pk prepare (Enter Execs)	forced, 10 lines
	src=(int)op_dist_outcome(arc_dist); next: next:	~
s	ost=(int/pp_uss_ourcome(ost_oist); if (dst == src) goto next; temp=op_dist_outcome(svc_dist);	
	pkptr=op_pk_get(op_intrpt_strm()): op_pk_ntd_pet(pkpt:*src*,src): op_pk_ntd_pet(pkpt:*dst*,dst); op	pk. nid. set (pkptr.'svc',temp);
01	total pk+=1; total time+=temp;	
St	te 1: pk prepare (CET's)	
_		_

Proc	ess M	odel Report: mm1br_scheduler Fri Jan 7 03:31:38 1994 Page 5 of 10
M/M/ scher	1/B/R	L Uniform Traffic System Model
σ	state	2: idle (Enter Execs) unforced, 56 lines
	. <u></u>	(lload_steady && reteased_pt>0)
		t Intpr=slot_no*total_time/(N*op_sim_time()); ratio=temp/load_prev: if (ratios > 0.999 && ratio = 1.001)
5		
		load_steady=1, ratio=0, for (i=1: i<=N1: i++)
		if (pkt(i)sc:)=0) ratio+=(op_sim_time()-pkt(i).cime): cu_neve=(rim-k-retion)*telor =no(NV*on eine finaci):
ž		pb_prev=double)blocked_pk(double)total_pk; pbc_prev=(double)blocked_c_pk((double)total_pk;
		cmp=cu_prevrvit; pk_count=floor(temp); temp=(double)reteased_pk/(double)pk_count: tr_count=floor(temp)+20: }
		l Itali strady && tready && released at the count to count to
	:	
		if ('cu_steady)
		t ratio=0:
×	_	for (i=1: i<=N1: i++)
		If (pk(t) src:=U) rattot=(op_sim_time()-pk(f) _ctime); temp=(time+ratio)*slot no/(N*oo sim_time()); ratio=temo/cn prev
		if (ratio > 0.999 && ratio < 1.001 && time<(total_pk*svc_time) && temp < cu_max) cu_steady=1;
25		clse cu_prev≃temp; }
		if (.jpb_steady)
		t temp=(double)blocked_pk/(double)total_pk;
		if (pb_previ=0) ratio=temp/pb_prev; else if (temp==0) ratio=1; else ratio=0;
• • •		if (ratio > 0.999 && ratio < 1.001) pb_steady=1; clse pb_prev=temp;
		if ('fpt_steady)
35		temp=(double)hlocked_c_pk(double)total_pk;
		nt (poc_previ=4.0) ratio=temprobc_prev; else if (temp=-0) ratio=1: else ratio=0:
		if (ratio > 0.999 & & ratio < 1.001) pbc_steady=1; else pbc reveience
9		
		if (cu_steady && pb_steady && pbc_steady) ready=1; if (theadw)
	-	
45		ratio=0: for (i=1: i<=N1: ii+1)
	-	if (pit(i) srcl=0) sin_time()-pit(j) clime):
		cmp=(unc+rato)^sio(_no^nNi/(N*op_sim_time());
50		
		(cady)
		temp=total_pk*stot_no*svc_time(N*0p_sim_time()): ratio=temp/load_prev: if (ratios0.995&&ratios1.005&&time <total_nk*svc_time &(time*stor_no="" (n*0n_sim_time=""))="" k="" m="vENDSIM</td"></total_nk*svc_time>
55		op_sim_end(reaching steady state(

Process Model Report: mm1br_scheduler	Fri Jan 7 03:31:38 1994	Page 4 of 10
M/M/1/B/RL Unitorm Traffic System		
scheduler Model		

¥ CEI	Cond: Exec:	()
	Trane.	schodula

팀 월 📃	/RL Uniform Traffic System sr Model
	op_pk_ntd_get(packet, src-≻): op_pk_ntd_get(packet, asr-&ut): op_pk_ntd_get(packet, src-&src-&src)
	ct=0. for (j=0.)_2×wavelength_no: j++) cmpty[j]=0:
10	for (i=0)=cfm_size:i++)
	<pre>conflict[][0]=1: conflict[][1]=1: for (A=0: K-wavelength_no: X++)</pre>
	go_back=0; if (conthicti;1[0]==-1 &&& conthict[5][1]==-1)
	for (k=0; kcwavelength_no; k++) { if ((slot[i][K]]=0 ll i==(fm_size-1)) && cmpy/k]>0)
	i (slot(i)[k]:=0) j=i=empy(k]-1: else j=i=empy(k]: if (j=c) oh=0. else if ((conflict(i)[0]==-1 conflict(i)[0]==-4 & & & & & & & & & & & & & & & & & & &
-	clse ch=2: if (ch=2 & cmpy[k]<2) (cmpy[k]=0; continue:) inser: if (ct==0)
	<pre>if (shot) k)=0 # go. back) store(0 0)=i-empy(k): clse store(0 10)=i-empy(k)+1: store(0 1)=k: store(0 12)=empy(k): store(0 13)=oh: }</pre>
	ces for (j=(ct-1); j>=0; j=) { if (store[j][2]===mpy]k] ∥ (store[j][2]===mpy]k] && store[j][3]>ob))
	<pre>store[i+1](0]=store[i](0]; store[i+1][1]=store[i](1]; store[i+1][2]=store[i](2]; store[i+1][3]=store[i][3];</pre>
	if (storetj][2)⇔empty]k] I (storetj][2)≕empty]k] & & storetj][3]<=oh)) {
	store_ct: fi (sho(i)k)=0l go_back) store[j+1][0]=i=empy]k]; cts store[j+1][0]=i=empy]k[j+1: store[j+1][1]=k: store[j+1][1]=k: break:
	$if(j=0) \{ j=1; \text{goto store_ct: } \}$
	if (ge_back; empty(k)=0: ct+=i; }

Process Mode	el Report: mm1br_scheduler	Fri Jan 7 03:31:38 1994	Page 6 of 10
M/M/1/B/RL U	niform Traffic System		
scheduler Mot	Jel Jel		
State 2:	idle (CET's)		
CET Cond	(ARRIVAL)		

₩ ₽	Exec:	
7	Trans:	pk_prepare
CET	Cond:	(default)
#	Exec:	
1	Trans:	idle
CET	Cond:	(ENDSIM)
# 2	Exec:	record_stats();
1	Trans:	idle

Schedule (Enter Execs) no.in_svc = N1) (no.in_svc = N1) (blocked_pk+=1: op_pk_destroy(pkprt);) e if (find_resource(pkprt)) (blocked_pk+=1: blocked_cpk+=1: op_pk_destroy))	forced, 4 lines				estroy(pkptr);}	
	3: schedule (Enter Execs)	(no_in_svc = NI)	{blocked_pk+=1; op_pk_destroy(pkptr); }	2	if (!find_resource(pkptr)) {blocked_pk+=1: blocked_c_pk+=1: op_pk_d	

Fur	Iction Block	230 lines
	record_stats() {	
	int double	iji. certij
2	extern double extern channel_asgn	time: pkt[1025]:
10	for (i=1; i<=N1; i++) if (pkt[i].src!=0) t	me+=(op_sim_time()_pk{[1],cine):
	cu=time*slot_no/(N*0, if (cu > cu_max) on stat write scalart"	_sim_time()): custut innet(): coset statu max:
15	op_stat_write_scalar(* op_stat_write_scalar(* op_stat_write_scalar(*; }	cores. 20. (double)blocked_c.pk(double)total_pk). 30. (double)blocked_c.pk(double)total_pk).
	_	
20	int find_resource(packet) Packet* packet;	
25	f int double	ijikapapi nd cicobgo, back waste i citificarsts.dt: conflict[130][2],empy[10],score[1025][4],temp[130],asgo[130],rmb[130],tmp_mb[130]; svc:
	cxtern int cxtern other_asgn extern channel_asgn	and an accessful 130] [0]: and all accessful 230] [1]: pixt1025]:

Prod	cess N	Model Report: mm1br_scheduler	i Jan 7 03:31:39 1994 Page 8 o	10
MN sche	V1/B/I eduler	RL Uniform Traffic System r Model		
		ala		
	8	tixe for (k=∩ k cruatelenath mo' k+++)		
		[if (cmpty/k]>0)		
	95	if (slot(i] k ==4). empty k =1; if (empty k ==0) continue: j=i-empty k -1; if (j<0) oh=0;		
	8	etse ((confinetjall0)=-1 il confinetjalj0)==b)&b etse bo=2: if (confinetjalj0)=-1 il confinetjalj0]==b)&b(s.con if (confinetjal]0)==1 il confinetjalj0]==b,&b(s.con etse if (d==0) od=1; etse od=3;	(conflict(j)[1]=-1 % conflict(j)[1]==k)) oh=0; lict(i)[1]==-1 ∥ conflict(i)[1]==k)) oh=oh:	
	105	if (((oh==2 oh==1)&& cmpy(k <2) ((oh==3 & goo insert: paos: back: cmpy(k =0: ct+=1:	& empty(k)<3)) { empty(k)=0. continue: }	
	110	-		
	115	nd=slot_not_i_ct=0; west=4; west=0; agn()=1; sgn()]=0; for (i=0; i=0; i=0; i=0; =0; imp_mb(i]=0; } for (i=0; i=0; i=0; i=0; i=0; i=0; i=0; i=0;		
	120	if (store[i][3]==0, fit = 0; else if (store[i][3]==3) fit = 2; else j=store[i][2]-fit:		
	125	if (next) () if ()>=nu) if ()>=nu) if ()>=nu)		
	130	K=pad+Ht. if (fice=ap) && k ap) f (or (oh=store[i][0]; ohc(store[i][0])+store[i][2]); of if (mb[oh]) (k=1; break; }	(#	
		11 (x:=-1) حال من الحال الحال الحال الحال الحال الحال المال الحال (مال (مال حال الحال الحال الحال الحال المال (مال - 1) المال - 10 (مال - 1) المال - 10 (مال - 10) المال	č	
	135	k=sucr[1][0]-1: if ((.k.o.) k=0) go_back=sucr[1][0]+store[1][1]=1 i: if (.go_h for (oh=k: oh-go_back: oh++1 imp_m[lob]) }	ck>fm_size) go_back=fm_size: =1 ,	
	140) } if(_jcadfl ==(ct-1))		
	¥	 wastc+=apl: i_ct+=1: if (wastc <asgn[0]< td=""><td></td><td></td></asgn[0]<>		
	<u>}</u>	t asgn[0]=waste: asgn[1]=1_ct: for (k=2: k<(i_ct+2), k++) asgn[k]=temp[k-2]:		

_



85

Process Model Report: mm1br_release	Fri Jan 7 03:33:21 1994	Page 1 of 1
M/M/1/B/RL Uniform Traffic System		
release Model		

	Function	Block	No
	Temporary	Variables	Yes
summary	State	Variables	Ŵ
	Header	Block	Yes
	Number of	States	-

2 lines			
Temporary Variables	Packet* pkptr:	int wavelength.timeslot,i.j.k:	

tate U: discard (E	if (op_intrpt_type()==OPC_IN	pkptr=op_pk_get(op_intry op_pk_nfd_get(pktt.*varv i=slot[timeslo1][wavelength for (j=0: jcpkt[i]]tet: j++)	for (k=other[i][j],slot: other[i][j],slot=0; othe }
EXECS) Unforced, 12 lines		m()): uct+i-&wavelenguh):op_pk_nfd_get(pkpu:'timesiot'.×iol): te+=(op_sim_timet)-pkt[i]ctime1:	hefil(j).slot-othefil(j).ken/k k++) slot(k] othefil(j).wavelength]=0; .wavelength=0; othefil(j).ken=0; .mad> ndfil1.clima=0 nn in suv=1: relevent oth=1: an nk detravitent }

Sta	te 0:	discard (CET's)
G	Cond:	(1)
¥	Exec:	
	Trans:	discard

Process Model Report: mm1br_scheduler	Fri Jan 7 03:31:40 1994	Page 10 of 10
M/M/1/B/RL Uniform Traffic System		
scheduler Model		



B.3.2 Queueing System

Many methods can be used to determine the stopping time for a simulation [8].

Regenerative method [9] was used as the termination criterion for M/M/1/O/L+1 system. A cycle starts when the queue changes its size from zero to one. Average queue size is measured in each of the cycle, and its half length of the 90% confidence interval is calculated. The program terminates when one of the three conditions are true: the half length interval of the average queue size for 90% confidence interval reaches within 10% value of the average queue size, or the number of the cycles gone through by the simulation reaches 20,000, or the simulation has been running for 2,000,000 seconds. The simulation will also end if it is determined that its cycle time is extremely long, that is it went through less than two cycles in 99,999 seconds. The procedure outlined in [10] is used to calculate the inverse of the cdf of the normal distribution, which seems to have a relative accuracy of about 5 decimal places.

For *M/M/1/Q/CL* and *M/M/1/Q/RL* systems, a sequential batch means procedure outlined in [11] is used to determine the run length of the simulation. The following is the description of the procedure:

- (1) Set $i \leftarrow 1$, $n_0 \leftarrow 600$, $n_1 \leftarrow 800$. Collect n_1 observations. (2) Partition the n_j observations into 400 batches, each of size n/400. If the estimated serial correlation in these batches is greater than 0.4, go to (5). If it is negative, go to
- (3) Partition the n_i observations into 200 batches, each of size n/200. If the estimated serial correlation among these 200 batches is greater than that among the 400 batches, go to (5).
- (4) Partition the n_i observations into 40 batches, each of size n/40. Construct a nominal p-percent confidence interval assuming the 40 batches are independent and ignoring that the batches were constructed sequentially. <u>If</u> the interval is acceptably small relative to the current value of X (say half-width/X < γ), stop; otherwise go to (5).
 (5) Set i ← i +1, n_i ← 2n_{i-2}. Collect n_i n_{i-1} additional observations and go to (2).

In our simulations, we used the 90% confidence interval, and the termination criteria is γ from step (4) equals to 0.1. Also the observation was made on the average queue size after the measured load comes to within 1% of the offered load obtained in Section 4.1. Again the simulation is terminated either when the above criteria is met or after it has been running for 2,000,000 seconds.

The seed used for each simulation has value NWL, where N is the number of users, W the wavelength channels, and L the throughput requirement. So for N equals to forty, W equals to eight and L equals to one, the seed is 4081.

M/M/1/O/L+1: Opnet reports for the scheduler and release processor models.

1/10.1.+ 1 Uniform Traffic System duler Model duler Model 0bjid bown_id. 0bobb bown_id. 0bjid bown_id. <th>T/OL +1 Uniform Traffic System duler Model duler Model Uniform Traffic System int Uniform System int Uniform System int Uniform System int Internation Uniform System Uniform System Uniform System</th> <th>UD.IU.40 1004 I 1 895 E VI</th>	T/OL +1 Uniform Traffic System duler Model duler Model Uniform Traffic System int Uniform System int Uniform System int Uniform System int Internation Uniform System Uniform System Uniform System	UD.IU.40 1004 I 1 895 E VI
State Variables 33 lines Objid wc.ud.d. Objid wc.ud.d. Int wordc.noc. Int Wrarc.arc. Dobbe wcmcarg100. Abstribution *ec.dist. Dobbe wc.dist. Dobbe word.gr.c. Dobbe word.gr.c. Dobbe word.gr.c. Dobbe wc.dist. Dobbe wc.dist. Dobbe wc.dist. Dobbe wc.dist. Dobbe wc.dist. Dobbe word.gr.c. <th>State Variables Objid were, id. Objid were, id. int wavetergeht, not. int wavetergeht, not. int wavetergeht, not. int wavetergeht, not. int wavetergeht, not. int wavetergeht, not. int wavetergeht, not. Bathioton wavetergehtergeht, not. Bathioton wavetergehtergeht, not. Bathioton wavetergehtergeht, not. Bathioton wavetergehtergeht, not. Bathioton wavetergehtergehtergehtergeht, not. Bathioton wavetergehtergehtergehtergehtergehtergehtergeht, not. Bathioton wavetergehter</th> <th></th>	State Variables Objid were, id. Objid were, id. int wavetergeht, not. int wavetergeht, not. int wavetergeht, not. int wavetergeht, not. int wavetergeht, not. int wavetergeht, not. int wavetergeht, not. Bathioton wavetergehtergeht, not. Bathioton wavetergehtergeht, not. Bathioton wavetergehtergeht, not. Bathioton wavetergehtergeht, not. Bathioton wavetergehtergehtergehtergeht, not. Bathioton wavetergehtergehtergehtergehtergehtergehtergeht, not. Bathioton wavetergehter	
State Variables 33 incs Objid vov.l.d. Objid vov.l.d. Ini voc.n.o. ini Vitt. Init Vitt.	State Variables Objid vown.id. Objid vown.id. int warecall, woodc.no. int fur size: int fur size: int fur size: int fur size: int fur size: int fur size: int fur size: Nhi: Char vorrg(20); barbidon varrg(20); barbidon varrg	
0bjd Vow.1d. 0bjd Vow.1d. 0bjd Vow.1d. 11 Viewsteigth.aot. 12 Jobbie 13 Distribution 14 Viewsteigt. 15 Distribution 16 Jobbie 17 Viewsteigt. 18 Distribution 19 Jobie 10 Jobie 11 Viewsteigt. 12 Jobie 13 Jobie 14 Viewsteigt. 15 Distribution 16 Jobie 17 Liewsteigt. 18 Distribution 19 Jobie 10 Jobie 10 Jobie 11 Viewsteigt. 12 Jobie 13 Jobie 14 Jobie	Objid Vown_id, Dojid Vown_id, Nec_id, int Dir Vec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Dir 10 double Vec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Nec_id, Dir 15 Distribution Nec_id, Distribution Nec_id, Distribution Nec_id, Nec, Nec, Nec, Nec, Nec, Nec, Nec, Nec	33 lines
0 bjid werkength.not. init werkength.not. init werkength.not. init Will. 10 double war.sing.init. Will. 11 war.sing.init. 11 war.sing.init. 11 war.sing.init. 11 war.sing.init. 11 war.sing.init. 12 bage init 13 Distribution 14 war.sing.init. 15 Distribution 16 war.sing.init. 17 bage init. 18 Distribution 19 bage init. 10 b	0bjid werkingth_not int workingth_not int workingth_not int Win_strest int Vortage	
int Wordengelu.not. int Vinc.not. Onder Ver.dint. Onder Ver.dint. Distribution *ex.dist. Dioptic Vinc	ini waver.nos. ini waver.nos. ini waver.nos. ini waver.nos. ini No. No. No. No. No. No. No. No.	
init Wavetergut.not. init Wavetergut.not. init With Sizes. barbit Wer.finate. barbit Wer.finate. barbit Wer.finate. barbit Wer.finate. barbit Wast.dist. barbit Wast.dist. barbit Wast.dist. barbit Wast.dist. barbit Wast.stat. barbit Wa	int Wardcreigth_noc int With jazz: int Nit. Nit. (11. jazz: Nit. (11. jazz: Nit. (12. jazz: Nit. (13. jazz	
 in VII. size: in VII. size: in VII. in VIII. 	 ini VTI jižiči ini VTI jižiči ini VTI jižiči ini VTI jižiči odoble Vev. dine. bistribuion Vev. dine. 	
int VII: int VOLOC: int VOLOC: int VII: char VV.ciffic: char VV.ciffic: char VV.ciffic: char VV.ciffic: bistribution Vec.dis: bistribution Vec.dis: bistribution Vec.dis: cycle.stat cycle.stat cycle.stat cycle.stat couble Volat. couble Volat. volat.	int VT1: int VV. int VV. char VV. char VV. Char VV. Barihuion VV. Barihuion VV. Barihuion Varag(20): Derihuion Varag(30): Derihuion Var	
int Norunce int Norunce double Vary_affact double Vary_affact Destribution *ec_dist Destribution *ec_dist double Valaps: double Valaps: double Valaps: double Valaps: double Valaps: double Valaps: double Valaps: double Valaps: Destribution *ec_dist double Valaps: double Valaps: Destribution *ec_dist double Valaps: double Valaps: double Valaps: double Valaps: double Valaps: double Valaps: double Valaps: double Valaps: Mead steady.	int Not_no: int Not_no: chaobie vvs_inne: chaobie vvs_inne: barbuion vs_inte: barbuion vs_ard.ds: barbuion vs_ard.ds: barbuion vs_ard.ds: barbuion vs_ard.ds: barbuion vs_ard.ds: barbuion vs_ard.ds: barbuion vs_ard.ds: barbuion vs_ard.ds: vsv_d.ds: barbuion vs_ard.ds: barbuion vs_ard.ds:	
Int N1: other Wer_inner, other Wer_inner, other Wer_inner, other Wer_inner, Distribution Wer_dist; Distribution Wer_dist; Distribution Wer_dist; Distribution Wer_dist; S Distribution Stati Wergat; Notal pre: Notal pre:	int N: int N: obset wsr_ang(20): double wsr_ang(20): double wsr_ang(20): Bernhuiden wsr_ang(20): S Destribution "der, dis: I Deng int "Vert_dis: Double Usat Ouble West, gapa: double West, gapa: double "der, fine: int Vood int Vood int Vood steady: int Vood steady:	
int N1: 0.deb/e Ver_inic. 0.deb/e Ver_inic. 0.deb/e Ver_inic. 0.deb/e Ver_inic. 0.deb/e Ver_offst. 0.deb/e Ver_offst. 0.deb/e Ver_offst. 0.deb/e Ver_offst. 1.def Ver_offst. 2.def Derrbuion 0.de Ver_offst. 1.de Ver	int N1: charble Vwc_innc; charble Vwc_innc; Distribution Vwc_innc; Distribution Vwc_innc; Distribution Vwc_innc; Distribution Vwc_innc; Stribution Vwc_innc; S Distribution Vwc_innc; S Distribution Vwc_innc; Operation Vwc_innc; torg int Vwcie.ct; long int Vwcie.ct; obolic Voad_prev; double Vancie.ct; double Vancie.ct; double Vancie.ct; int Vancie.ct; int Voad_sm; int Voad_sm;	
0 double ver_inne: char ver_arg20; Distribution ver_arg20; Distribution ver_arg20; Distribution ver_arg30; Distribution	0 dauble Wr. Jane: Aur. arg.10; bestimbuion 1 John wr. arg.10; bestimbuion Wr. Jane: Lar. arg.10; bestimbuion 2 Distribution "Ast. dist. "Vex. dist. 1 Distribution "Ast. dist. "Ast. dist. 1 Distribution "Ast. dist. "Ast. dist. 1 Distribution "Ast. dist. 2 double Word. June: 2 double Word. Stat. 2 double "Ast." 2 double "Ast." 3 double "Ast." 4 "Ast." "Ast." 3 double "Ast." 4 "Ast." <td< td=""><td></td></td<>	
char Mar. Jag. 201: Sathhuion Mar. Jag. 201: Sathhuion 5 Beathhuion Mar. Jag. 363: Sathhuion 5 Beathhuion Mar. Jag. 363: Sathhuion 5 Spit. Sathhuion Mar. Jag. 363: Sathhuion 6 Spit. Sath Sathhuion Mar. 364: Sathhuion 10 Spit. Sath Sathhuion Mar. 374: Sathhuion 10 Spit. Sath Sathhuion Vere. Git. Sath Sathhuion 10 Spit. Sath Sathhuion Vere. Git. Sath Sath Sath Sath Sath Sath Sath Sath	dater wer_arg(20): bestribution wer_are: Distribution wer_are: S Distribution wer, dist Distribution wer, dist port int werd(2000); long int werd(20	
double Var_affe: Destribution Var_affe: Var_affe: cycle_stat Var_affe: Var_affe: cycle_stat Var_affe: Var_affe: Var_affe: cycle_stat Var_affe: Var_affe: Var_affe: Var_affe: double Var_affe: Var_affe: Var_affe: double Var_affe: Var_aff	Bothle Warr arte: Warr dist. Distribution "An arts" "An arts" S Distribution "An arts" "An arts" S Distribution "An arts" "An arts" S Splic_stat "Var. dist. "An arts" S Splic_stat "Var. dist. "An arts" S Splic_stat "Var. dist. "An arts" Doing int "Var. dist. "Var. dist." Doing int "Var. dist." Obolic "Var. dist." Int "Var. dist." Int "Var. dist." Int "Nod." dist."	
Distribution "exc. dist. 5 Distribution "exc. dist. 5 Distribution "exc. dist. 6 Distribution "exc. dist. 1 Distribution "exc. dist. 2 double "exp. dist." 3 int "exp. dist."	Distribution wist dist. "exc. dist. - wist dist. 5 Distribution - wist dist. "exc. dist. - wist dist. 10000 init ong init ong init on gint obtoble double double double double double double double double double double double double double init bit "exc. dist. - wist.conf. vol.conf.conf.conf.conf.conf.conf.conf.conf	-
Statistion "Aut dist. 5 Distribution "Aut dist. 1 Distribution "Aut dist. 1 Long int "Ver_dist. 1 Long int "Ver_dist. 1 Long int "Ver_dist. 1 Long int "Ver_dist. 1 Long int "Ver_crist." 1 Long int "Ver_crist." 1 Long int "Ver_crist." 1 Lonble Waal_prev." 1 Lonble Waal_prev." 1 Lonble Waar." 1 Lonble Waar." 1 Lonble Waar." 1 Lonble Ver_crist." 1 Lonble <td>Distribution big int "dst_dist. 5 Distribution int "dst_dist. 0 opg int "ver_dist. 1 obje "ver_dist. 1 int "ver_dist. 1 obje "ver_dist. 1 int "ver_dist. 1 int "ver_dist. 1 obje "ver_dist. 1 obje "ver_dist. 1 int "ver_dist. <!--</td--><td></td></td>	Distribution big int "dst_dist. 5 Distribution int "dst_dist. 0 opg int "ver_dist. 1 obje "ver_dist. 1 int "ver_dist. 1 obje "ver_dist. 1 int "ver_dist. 1 int "ver_dist. 1 obje "ver_dist. 1 obje "ver_dist. 1 int "ver_dist. </td <td></td>	
5 Distribution *vec.dist. 6 evec.stat *vec.dist. 6 evec.stat *vec.dist. 1 evec.stat *vec.stat 1 evec.stat *vec.stat <td> Batrihution "ure dist. Distribution "ure dist. long int votal_26: long int votal_26: long int votal_26: long int votal_26: long int votal_27: long int votal_17: long int votal_17: double votal_17: double votal_17: double votal_16: int votal_16: int votal_16: famDoratry Variables </td> <td></td>	 Batrihution "ure dist. Distribution "ure dist. long int votal_26: long int votal_26: long int votal_26: long int votal_26: long int votal_27: long int votal_17: long int votal_17: double votal_17: double votal_17: double votal_16: int votal_16: int votal_16: famDoratry Variables 	
cycle.stat vestal[2000]; borg ini vestal[2000]; vestal[2000]; torg ini vestal[2000]; vestal[2000]; vestal[2000]; torg ini vestal[2000]; vestal[2000]; torg obset vestal[2000]; vestal[2000]; torg obset vestal[2000]; vestal[2000]; tord obset vestal[2000];<	cycle_stat vetat[2000]; long int voral_2000; long int long int voral_stx; voral_cat; double voral_stx; voral_inte; long int voral_stx; voral_inte; voral_stx; voral_inte; sobuble voral_stx; double voral_stx; voral_stat; double voral_stat; voral_stat; lint load_stead; poul lint load_stead; poul lint load_stead; poul	
long int votal pic. long int votal pic. long int votal pic. long int votal pic. long int votet.cycle. looble votal pice. double votal pice. double votal pice. double votal pice. looble votal pice. double votal pice. looble votal pice. looble votal pice. looble votal pice. looble votal pice. looble votal pice. looble votal pice. loot vo	long int voral.pk: long int voral.pk: long int vorcesed.pk; long int vorcesed.pk; long int vorcesed.pk; long int vord.procesed.pk; double vlant: double vlant: double vlant: double vlant: double vlant: double vlant: double vlant: lint vord.pk; mit vord.pk; mit vord.pk; lint vord.sim: lint vord.sim: lint vord.sim: lint vord.sim: lint vord.sim: hkett* hptor:	
Iong int one int Vercessed_pk: hore int Vercessed_pk: vercessed_pk: double Vercessed_pk: vercessed_pk: vercessed_pk: double Vercessed_pk: vercessed_pk:	Iong ini ong ini double double double double double double double double double double double double double tini tini bid ipposities double valants double valants double valants double valants valants double valants double valants	
Image in the system	00 long int long int double 'verle_cri weat_cycle; double 15 double 'verle_cri weat_sime; double 16 double 'verle_cri weat_sime; double 10 double 'verle_cri weat_sime; double 10 double 'verle_cri weat_sime; double 11 'verle_fine; double 'verle_fine; 'verle_fine; int 11 'verle_fine; int 'verle_fine; 'verle_fine; 11 'verle_fine; 'verle* 'verle* 11 'verle* 'verle* 11 'verle* 'verle*	
Openance Openance Openance 0 double Voad_prev: Voad_prev: 0 double Vaene: Voad_prev: 0 double Voad_prev: Voad_prev: 0 double Voad_prev: Voad_prev: 0 double Voad_prev: Voad_prev: 0 double Voad_prev: Voad_prev 0 double Voad_prev: Voad_prev 0 double Voad_prev Voad_prev 0 double Voad_prev Voad_prev 0 double Voad_prev Voad_prev 0 double Voad_prev Voad_prev 0 double Voad_prev </td <td>Double Double Double Double double Uoal_prev: Uoal_prev: Uoal_prev: double Uoal_prev: Uoal_prev: Uoal_prev: double Valence: Valence: Uoal_prev: double Valence: Valence: Valence: double Valence: Valence: Valence: double Valence: Valence: Valence: double Valence: Valence: Init int Uoal_steadv: Init Valence: Packet* Paper: Paper: Packet*</td> <td></td>	Double Double Double Double double Uoal_prev: Uoal_prev: Uoal_prev: double Uoal_prev: Uoal_prev: Uoal_prev: double Valence: Valence: Uoal_prev: double Valence: Valence: Valence: double Valence: Valence: Valence: double Valence: Valence: Valence: double Valence: Valence: Init int Uoal_steadv: Init Valence: Packet* Paper: Paper: Packet*	
on program on the sec to t	 Anny and Anny ann	
double war_inte. double war. double war. double var. int void strady. int void strady. int void strady. int void strady.	double Voar_Innt. double Voar_Innt. double Valay: double Valay: double Vara. double Vara. double Vara. double Vara. double Vara. int Variables Temporary Variables	
Inclusion Control Value Souble Value Value South Value South Int Value South	double double should double double double double int verage ve verage verage verage verage verage verage verage vera	
double double Valery: Valery: double Valery: Valery: valery: double Valery: Valery: valery: valery: int Valery: Valery: Valery: valery: valery: Valery: Valery: valery: valery: Valery: Valery: valery: valery: Valery: Valery: Valery: valery: valery: Valery: Valery: Valery: valery: valery: Valery: Valery: Valery: Valery: Valery: valery: Valery: Valer	Souble Veray: Veray: double Veray: Veray: veray: double Veray: Veray: veray	
S double double tobub Varpus double double bouble double int int int bouble int bouble int bouble int int bouble int int bouble int int bouble int int bouble int int int int bouble int int int int int int int int int int	S double double double lupbus: lump_alpha; double 0 users: double users: visit 0 double visits: visit 0 double visits: visit 1 used_sim: int visits: visit 1 load_stady; 1 load_stady;	
55 double Quence: double Vence: double double Vence: Vence: int Vence: Vence: int Vence: Vence: femportary Variables 3 lines	55 double Venerc. double Venerc. Venerc. int Volt. Venerc. int Voad steadyr. Venerc. int Voad steadyr. Venerc. Packet* Paper. Paper.	
double term_alpha: int term_alpha:	double termp_alpha. double termp_alpha. double termp_aprotect double vargar. double vargar. double vargar. double vargar. double vargar. int vard_sint. int load_sint. int load_sint. int load_sint. int load_sint. int load_sint.	
double terms_preue: double vara: double vara: 30 double vara: int vard_sim: vara int load vara int vara vara	double termgenere: double varar: 30 double varar: int varar: varar:	
double Usar: double Vara double Vara int Vard_sm: int Vard_sm: int Vard_sm: int Vard_sm: int Vard_sm: int Vard_sm: int Variables Temporary Variables 3 lines	double Varat: 00 double Varat: 01 double Varat: int Vood sim: Varat: int Vood sim: Varat: int Vood sim: Varat: int Vood sim: Vood sim: int Vood sim: Vood sim: int Nood sim: Vood sim:	
double Vaire: bit 30 double Vaire: bit int Variation: bit Variation: Variation int Variation 3 lines Temporary Variables 3 lines	About be value value value 00 double value value value value int value value int load steady: fam. int load steady: fam. int load steady: fam. int load steady: fam.	
30 doubte b: bint byte, inc. bint byte, inc. int byte, inc.	30 double bit int byte_inc. int byte_inc. int boad stradyr. int boad stradyr. Femborary Variables	
int bout steady. Int bout steady. Temporary Variables 31 mes Packer byter:	int vycle_ine: int vend_sim: lint lood steady; Temporary Variables Packet* piper:	
int voul_sin: int voul_seafr. Temporary Variables 3 lines Packer	int ved_sim: int load_steady: Temporary Variables	
int kead steady. Temporary Variables 3 lines Packet aptor:	int local steady: Temporary Variables	
Temporary Variables 3 lines Packet pour	Temporary Variables	
Temporary Variables 3 lines Packet aptor: Int orderity	Temporary Variables	
I emporary variables Jenes Jen	Temporary Variables	0 linco
Packet* pkptr: int scredstitk:	Packet* pkptr:	3 lines
int sredsticity		
	int src.dst.ij.k:	

Process Model Report: mm1q1_scheduler	Fri Jan 7 05:10:45 1994	Page 1 of 7
M/M/1/Q/L+1 Uniform Traffic System		
scheduler Model		

	Function	Block	Yes	
	Temporary	Variables	Yes	
SUITING	State	Variables	Yes	
	Header	Block	Yes	
	Number of	States	4	

Ηe	ider Block	26 lines
	#include <stdlib.h></stdlib.h>	
	#include <stdio.h></stdio.h>	
	#include <math.h></math.h>	
5	typedef struct	
	int src;	
	int dst;	
	double ctime;	
9) channel_asgn:	
	typedet struct	
	double a:	
15	double q:) cycle_stat:	
ę	ctrame1_asgn slot[130][10]: ini 	
3	double unter	
	#define ARRIVAL (op_intrpt_type()==OPC_INTRPT_STRM) #define ENDSIM (op_intrpt_type()==OPC_INTRPT_ENDSIM)	
25	#define EMPTYQ (op_subq_empty(0)) #define NEXTQ (op_intrpt_type()==OPC_INTRPT_REMOTE && \	
_	(EMPTYO)	

Pro	s Model Report: mm1q1_scheduler Fri Jan 7 05:10:46 1994 Page 4 of 7	Process Model Report. mm1q11_scheduler Fri Jan 7 05:10:46 1994 Page 3 of 7
ww. sche	20.+1 Uniform Traffic System ler Model	MM/1/Q/L+1 Uniform Traffic System scheduler Model
J	.Trans' schedule	State 0: init (Enter Execs) forced, 27 lines [f_inverse(0.1.&2): /* 90% confidence interval V
<u> </u>	te 2: Idle (Enter Execs) unforced, 30 lines	op_ima_sim_attr_get(OPC_IMA_INTEGER.vr.¬c_no): op_ima_sim_attr_get(OPC_IMA_INTEGER.vr.&vavetength_no):
	u (toda secon) (tech_poterioto.cyte_iterioto.cyte_iterioto.c) 11 ((cwje context owch && load stradiv)	9 optima_smi_artr_get(OFIMA_INTEGER.T.*.&mi_smc; op_ima_smi_attr_get(OFC_IMA_INTEGER.t.*.&silo_no);
\$		<pre>own_id=op.id_set(): src_id=op_topo_in_assoc(own_id():</pre>
	tor ()=0.50%c_c16.66% ()=4.5 [a]+=5840[j]=2.24=5840[j]=2.34=5840[j]=7.844=5840[j]a, 7645[j]a, 764=5840[j]a, 764=5840[j]a, 7545[j]a; =20[j]+=29406-er-611424-234540[j]a, 254454545454545454545454545454545454545	10 av.rat=ato(&av.arg0); av.rat=1/av.rate:
	restances and the second state of the second state of the second state of the second state of the second state is the second state of the second state state of the second state of the se	N=wavelength_not*fm_size: svc_time=(double)fm_size/(double)slot_not1): T1=floor(svc_time): N1=T1*wavelength_not svc_time=(double)fm_size/(double)slot_no: load_prev=arv_rate/(double)wavelength_not
	cos al=sv(0.1*r): if (al<1.0) end_sim=1:	<pre>13 3 sec_dist=op_dist_load(*exponential'svc_lintc(); src_dist=op_dist_load(*unitorm_int*1.1(doubb)mode_no); dst_dist=op_dist_load(*unitorm_int*1.(doubb)mode_no); dst_dist=op_dist_load(*unitorm_int*1.(doubb)mode_no);</pre>
	tise 12=j*a1*a1:j=foor(a2)-j: iff (xvache inc Inent realex==reale next realex==:	20 total_pk=0; processed_pk=0; no_in_swe=0; cycle_ct=0; end_sim=0; detay=0; time=0; total_time=0; load_steady=0; alpha=0; queuc=0; temp_alpha=0; temp_queuc=0; tstart=0; op_subq_flush(0);
<u> </u>		for (i=0;i <t):i++)< td=""> for (i=0;i<t):i++)< td=""> for (i=0;i<t,wavelength_noi)++)< td=""> for (i=0;i<t,wavelength_noi)++)< td=""> 25 (i=0;i<t,wavelength_noi)++)< td=""></t,wavelength_noi)++)<></t,wavelength_noi)++)<></t,wavelength_noi)++)<></t):i++)<></t):i++)<>
	<pre>if(cnd_sim && !ENDSIM) op_sim_end('reaching state''.');</pre>	for (i=0):cl20000;i++) (cstad(i).a=0: cstad(i).q=0:)
7	ii (cycle. ci==20000 & & ENDSIM) • p. sim_end (* 2000 cycle:).	State 0: init (CET's)
	ii (op sim_time()>99999 && cyclc_ctc2 && ENDSIM) op_sim_end(*cycle_too long);	CET Cond: (ARRIVAL) #0 Exec: : Trans: DK prepare
U	to 9. Maio (PETra)	Cond: (dcfault) Trans: idle
10	Cond: (default)	
*	Exec: : Trans: idie	State 1: pk prepare (Enter Execs) forced. 10 lines
U #	1 Cond: (ENDSIM) Evec: record.stats():	stc=(ini)op_dist_outcome(src_dist); next:
10	Trans: idle Cond: (NEXTQ &&.ARRIVAL)	$ds = (at \log of dist_outcome(ds, dist);$ if $(ds) = sc_0 st_0 on cvt$:
*	Exec: : Trans: schedule	5 temp=up_dist_outcome(svc_dist):
0¥	Cone: (ARRIVAL) Exee: :	pkptr=op_pk_get(op_intrpl.strm()): on ob ridd andreber recent new one ridd naderheber recent deity one ridd naderheber recent tame):
]	Tans: pk prepare	up_per_unc_satisfier at act, up_per_unc_satisfier dat act, up_per_unc_satisfier. 10 101a1_pk+=1: total inne+=temp:
		State 1: pk prepare (CET's) CETI Cond: (1)
		#0 Exec:

7	Process M/M/1/C	cess Model Report: mm1q1_scheduler Fri Jan 7 05:10:46 15 1/1/0/ + 1 1 Initioner Traffic System	k4 Page 5 of 7
	schedul	eduler Model	
	40	<u> </u>	
	PIO		rcea, 50 lines
		it (no_in_sec = NI)	
	<u>v</u>	<pre>5 if (m, subu_pk_insert().pkpur.OPC_QPOS_TAIL):=OPC_QINS_OK) [print((*error inserting into queue.ur): op_pk_destroy(pkpu); } cke</pre>	
	10	f ratio=op_subq_stat(0.0PC_QSTAT_PKSIZE); temp=op_sim_time(): if (ratio=1.0)	
		if (load_steady) {abph+=(temp-tstart): cstat(cyte_ct].a=apha; cstat(cycle_ct]) if (thoad_steady) { temp_apha=apha=apha=e-queue: } tstart=temp; qsize=1: apha=a?, queue=?; load_steady=1:	-queue. cycle_ct+=1.}
	15	15 celse	
		alpha+=(temp-tizat); queue+=qsize*(temp-tstart); tstart=(temp: qsize=ratio; if (tstart==0.0) print(r = r = 0 - 1.);	
	3	07	
		telse fit (Jfind_resource(pkpu))	
	22	25 if A subm_pk insert(a,pkpr.OPC_QPOS_TALL)=OPC_QDNS_OK) if (mult'server inserting inco queue.n.); op_pL.deruny(pkpu);)	
	30	cise f(==0, subq_stat(0,0PC_QSTAT_PKSIZE); temp=op_sim_time(): alpha+=(temp stat(); onecet=auxis [*] (temp stat): stat=temp: sist=rait)	
		torne work draw inter draw work to and to an a	
	<u> </u>	by if (NEXTQ)	
	40	<pre>ruio=0p_subq_stat(0.0PC_0STAT_PKSIZE); for (i=0:icratio:i++) if (no_in_svc < N1)</pre>	
		k k (find_resource(pkpr)) if (find_resource(pkpr)) if (op_sub_nk insert(0,tknr.OPC OPOS TAIL):=OPC ONS OK)	
-2:	45	45 [printif(error inserting into queue\n'). op_pk_destroy(p)	u); }
	50	raio-op_subq_stat(0.0PC_QSTAT_PKSIZE): temp=op_sim_time(): if (raioi=spire && hout_stated) [alpta+=(temp-tatert): queue+=qsize*(temp-tatert): tstar=temp: qsize=ratio;) 30 }	
	Sta	State 3: schedule (CET's)	
	¥ CEI	CET Cond: (1) #0 Exee: :	
	-	Trans. Mia	

x=1-x/2: p(t=0.32222431088e0t, p1=+1.0; p2=0.34224208847e0t, p3=0.02042311210245e0t, p4=0.453642210148e-4: q(t=0.099348462666e0t, q1=0.888881570495e0t, q2=0.531103462566e0t, q3=0.10353775285e0t, q4=0.38586706534e Fri Jan 7 05:10:47 1994 Page 6 o 87 lines a=a1/cycle_ct; j=cycle_ct; s11=(a+a2*a2);N(j-1); s22=(a3+a1*a1/5)V(j-1); s12=(a5*a1*a2);N(j-1); s=s11-2**s12+t**s22; s=z*sqrt(s*j)Na1; al +=(double/stat(j) a: a2+=(double/stat(j) q: a3+=(double/stat(j) a*(double/stat(j) a: a4+=(double/stat(j) q*(double/stat(j) q; double/stat(j) q: p=x: if (p=0.5) p=1.0p; y=sqn(40.9f*p)); z1=y+(f0x+Y^f(p2+y^*(p2+y^*p4))))(q0+y^*(q1+y^*(q2+y^*(q3+y^*q4)))); if (x<0.5)*z1=*z1: op_stat_write_scalar' t lead 'total_itime'slot_nol(N'op_sim_time()); op_stat_write_scalar('c = v'time'slot_nol(N'op_sim_time()); op_stat_write_scalar('c = v'time'slot_nol(N'op_sim_time()); op_stat_write_scalar('c = v'time'slot_scalar('slot_scalar('c = v'time'slot_scalar('c = v'time'slot_sca for (i=0; i<T): i++)
for (i=0; i<T+)
for (i=0; i<T+)
if (sloqi)[j]; acc=0) time+=(op_sim_time()-sloqi)[j], ctime);</pre> if (cycle, ct=0) (a l=abhta+temp_abhta, a2=quouc+temp_queue:) if (cycle, ct=0) & & al=0.0) a l=op_sim_time(); r=a2al: if (cycle, cp-1) if (cycle, cp-1) i.j.k: a=0.a1.a2.a3.a4.a5.s=0.s11.s12.s22.r.temp; double y.p.p0.p1.p2.p3.p4.q0.q1.q2.q3.q4: t int ij.k: double a=0.a12.a3.a4.a5.s=0.s11.s extern double time: extern channel_asgn slot[130][10]: al=0: a2=0: a3=0: a4=0: a5=0: for (j=0: j<cycle_ct && j<20000: j++) Process Model Report: mm1q11_scheduler MM/1/Q/L+1 Uniform Traffic System scheduler Model i.j.k.ap.sc.dt: int find_resource(packet) Packet* packet; { int f_inverse (x,zl) double x,*zl; Function Block 15 55 2 50 25 30 35 40 45 50 s

.

]				
Page 1 of 1					Function	Block	£
Fri Jan 7 05:13:51 1994					Temporary	Variables	Yes
				Summary	State	Variables	Ŷ
1 release	ystern				Header	Block	Yes
ss Model Report: mm1ql	/Q/L+1 Uniform Traffic S	e Model			Number of	States	-
Proce	IMM	releas					

			_in_svc;	, c;
			ou	ţ.

orary	Variables	3 lines
acket.	pkptr:	
ŗ	wavelength, timeslot;	
biid	own id.src id:	

discard (type()==OPC_J type()==OPC_J fid_get(pkptr.*w p_sim_time()*s stolf(wavetengti c=1: op_pk_de t_schedule_ren	discard (Enter Execs) unforced. 9 lines	type()==OPC_INTRPT_STRM)	nk wet(an intrut strm()).		p_sim_time()-slot(timeslot][wavelength].ctime);	slot][wavelength].src=0: slot[timeslot][wavelength].dst=0: slot[timeslot][wavelength].ctime=0:	c-=l: op_pk_destroy(pkptr); own_id=op_id_self(): src_id=op_topo_in_assoc(own_id.0);	t_schedule_remote(op_sim_time().1.src_id);
	State				Ś			
State				-	_		_	_

먍	ö	discard (CET's)	
E	:ond:	(1)	
-01	:xec:		
F	rans:	discard	

Process Model Report: mm1ql1_scheduler Fri Jan 7 05:10:47 1994 Page 7 of 7 MM/1/0/L+1 Uniform Traffic System scheduler Model			
MMM1/0/L+1 Uniform Traffic System scheduler Model	Process Model Report: mm1q11_scheduler	Fri Jan 7 05:10:47 1994	Page 7 of 7
scheduler Model	M/M/1/Q/L+1 Uniform Traffic System		
	scheduler Model		

	double	SVC:	
9	extern int extern channel_asgn	no_in_see: step1 30[10]:	
	op_pk_nfd_get(packet	.*src*.&xc); op_pk_nfd_get(packet.*dst*.&d); op_pk_nfd_get(packet.*svc*.&svc);	
~	ap=0. tor (=0;i <t1;i++) t</t1;i++) 	/* for each column */	
ç	k=-1: for (j=0;j <waveler< td=""><td>gth_noj∔+)</td><td></td></waveler<>	gth_noj∔+)	
,	if (slot[i][j].s if (slot[i][j].s	c==0) k=j: c==s ll sloqij[j].dsi==di) goto next_col:	
5	if (k!=-1) { a next_col: ap= }	⊨l: golo assign:) A.	
	if (!ap) return 0; else		
•	{ assign:		
~ ~	slottill(k).src=sc: s op_pk.nfd_se(p, no_in_svc+=1; pr op_pk_send_dela }	oolijisidserentiisodojijkikienimeeopisam, <u>innee):</u> cosket.wevelengtn.kkioopi pk.nld.ettiga cket.timesiot.ij cosket.jaketi.dakyvenopi <mark>sim.jinne()-op.pk</mark> .creation.jinne_gettipackec): yedi(packet.0.svc): return 1:	

<u>M/M/1/Q/CL:</u>

OPNET reports for the *scheduler* processor model. The *release* model is identical to that in M/M/1/Q/C+1.



Proce	s Model Report: mmtal scheduler Eri Ian 7 04:58-54 1904 Parce 4 of 8	Doctors Madel David and a stratic
M/M/ schec	O/CL Uniform Traffic System	Process mover report. mm.rg_scneourer Fri Jan / 04:56:34 1994 Fage 3018 MM/1/DCL Uniform Traffic System schendulzer Model
S	ite 2: idle (Enter Execs) unforced, 35 lines	State 0: init (Enter Execs) forced, 25 lines
	if (!load_steady)	f_inverse(0.1.&z): /* 90% confidence interval */
	temp=total_time*slot_not(N*0p_sim_timet)*load_prev); if (temp30,939 && tempc1.001)	op ima sim attr ge(OPC_IMA_INTEGER.rs. & and co):
ý.	[ad] steady=1: itstart-up, sim_time(), qsiziz-up, subq, stat(o,OPC, QSTAT, PKSIZE), numerical stat right new state8000 and intrust schedule sufflow else, interaction of the numerical stat right new state8000 and intrust schedule sufflow else.	9 prime sine and gettor C. P.M. LINEGER, "The Active gettor and an and gettor and an article gettor (OPC, JMA, INTEGER, "The Active gettor and article gettor (OPC, JMA, INTEGER, "The Active gettor and article gettor article article gettor
		own_id=op_idd_set();scc_id=op_topo_in_assoc(own_idd(); ono inna ohi aftr eet(scrid 'threasartisal asrcs'& & scry no(0)):
	ii (MEASURE)	10 av. rate=doi(&av. arg0); av. rate=jav.rate;
	temp=pp_sim_time(); queuet-spisor*(comp-tstart); statication; cti=queue/inc; stat_ct+=1; queue24; start=temp; qsize=up_subq_stat(0,0PC_QSTAT_PKSIZE); if (stati_rt=mever_rtor);	N=wavekangth_no*fm_size: svc_time=(double)fm_size/double)slot_no: T1=floor(svc_time): N1=T1+wavelength_no:
	l (15 swc_dist=op_dist_load(*exponential:swc_innc.0); src_dist=op_dist_load(*uniform_int:1.(doubb)mode_no); dst_dist=op_dist_load(*uniform_int:1.(doubb mode_no);
20	ii (computera) gous sept. step3: step3:	20 inc=10: total_pt=rf, processed_pt=0: no_in_svc=0; time=0; total_inne=0; total_pt=rf. load_prev=arv_rate/double/bwavefeneth_no: load_steady=0; red. sim=0: nb=600; nl=800;
25	strai_col/200.ktratuo; if ratios/stemp) goto step5: step4: col(440.ktemp):	for (=0: <t11;++)< td=""> for (=0;<t11;++)< td=""> for (=0;<www.elength_mo;++)< td=""> 25 [(=0)][(==0;<www.elength_mo;++)< td=""></www.elength_mo;++)<></www.elength_mo;++)<></t11;++)<></t11;++)<>
	ut (tempet).)) end sam≃1: stops: M((abent) { n1=n1*2; next.stat=n1: }	
8	clse {n0=n0*2; next_state=n0; }	State 0: init (CET's)
	op_intrpt_schedule_self(op_sim_time()+inc.()); }	CET Cond: (ARRIVAL) 40 Exec: : Arrive Centre Trans: Discontenare
35	if (end.sim && IENDSIM) op sim end('reaching steady state);	CET Cond: (default) #1 Exec: : Trans: idle
ŝ	e2: idle (CET's)	
U	Cond: (default)	State 1: pk prepare (Enter Execs) forced, 10 lines
¥	Exec: : Trans. Idla	StC=(titt)/0p_aist_outconne(stc_aist); next:
U	Cond: (ENDSIM)	dst=(int)op_dist_outcome(dst_dist); if (data ==) and a mathematical dist);
Ŧ	Exec: record stats(): Trans: idle	5 [temp=op_dist_outcome(svc_dist);
5 2 2	Cond: (ARRIVAL) Exec:	pkpu=-op_pk_get(op_intrpt_strm()):
ł	Tans: pk. prepare	op_pk_nid_set(pkpur*src*src); op_pk_nid_set(pkpur*dsr*dsr; dst; dst_nid_set(pkpur*src*temp);
걸육	Cond: (NEXTQ &&: ARRIVAL) Exec: :	10 total pk+=1: total time+=temp:
<u>:</u>	Trans: schedule	

Sta	te 1:	p k	prepare (CET's)			Γ
CET	Cond:	E				
¥	Exec:					
	Trans:	schet	dule			

MMM/DCL Unform Traffic System schoolder Model 10 schoolder Model 11 schoolder Model 12 schoolder Model 13 schoolder Model 14 schoolder Model 15 schoolder Model 16 schoolder Model 17 18 schoolder Model 19 schoolder Model 10 schoolder Model 11 schoolder Model 12 schoolder Model 13 schoolder Model 14 schoolder Model 15 16 16 17 17 18 18 19 19 10 10 11 11 11 11 12 12 13 14 14 15 15 15 16 16 16 16 16	Proce	200	Model Report: mm1cil scheduler	f a
10 Implementation 11 Implementation 12 Implementation 13 Implementation 14 Implementation 15 Implementation 16 Implementation 17 Implementation 18 Implementation 19 Implementation 10 Implementation 11 Implementation 12 Implementation 13 Implementation 14 Implementation 15 Implementation 16 Implementation 17 Implementation 18 Implementation 19 Implementation 10 Implementation 11 Implementation 12 Implementation 13 Implementation 14 Implementation 15 Implementation 16 Implementation 17 Implementation 18 Implementation 19 Implementation 10 Implementation 11 Implementation 12 Implementation 13 Implementation 14 Impleme				5
$ \begin{cases} \text{behavior}_{\substack{\text{rest}\\ \text{rest}}} \\ free (rest) (rest), (rest),$	schec	aulei		
$ \begin{cases} \begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 $				
$ \begin{cases} 10 & \text{vec}_{(1,2)} \\ (1,2) & \text{vec}_{(1,2)} \\ (1,2) & (1$			b=floor(s);	
$ \begin{cases} \text{ (c)} (\text{c)} (\text{c}) ($	<u> </u>		x=0.	
$ \begin{cases} \begin{array}{c} \text{Signature} \\ Signature$			ior (i=0:1<40:1++)	
[1] yil-yil/Do Areay(i); readd, site, for (all-side-distrib) ([=y](], r. se=[];) set0. for ([=](C1(1; +1)); for ([=](C1(1; +1)); fo			y(i)=0; for (j=0;5(tt);++)y(i)+=stat[i*b+j]:	
$ \begin{cases} x = x_{0} (x_{0}, x_{0}), \\ x = x_{0} (x_{0}, x_{0}),$		~ ~	y(1)=y(1)b: x+=y(1); }	
20 set05, if (Al-ad) set *prop(AD):: for (ad), Forth 1:++) for (ad), Forth 2:++) if ((a)((a)), Foreworking(h, 100;)++) if ((a)((a)), Foreworking(h, 100;)++) if ((a)((a)), Foreworking(h, 100;)++) if ((a)((a)), Foreworking(h, 100;)++) or stat, write, statier (- and, - and, inne'(b), 100;)+, 00; inne(h)); or stat, write, statier (- and, - and), inne'(b), 100; (b)((a)(b)(a)(a)(b)(b)(b)(b)(b)(b)(b)(b)(b)(b)(b)(b)(b)			x=x/40; s=0; for (i=f)i5<40; i++) (i=v1i+x: s+=i+i;)	·
$\begin{cases} \text{for } (\text{rd}, \text{i-f}(1; i_{1}, i_{1}), \\ \text{for } (\text{rd}, \text{i-constraint}(\text{int}, i_{1}+i_{1}), \\ \text{for } (\text{rd}, \text{i-constraint}(\text{int}, i_{1}+i_{1}), \\ \text{or stat.} \text{reflex statistic statistic (1, i_{1}, i_{1}+i_{2}), \\ \text{op stat.} reflex statistic (2, :1), \\ op stat., \text{reflex statistic (2, :2), \\ op stat., \\ op stat., \text{reflex statistic (2, :2), \\ op st$	20		s=s(39: if (x;=d) s=z*sqn(s(40)/x:	
$ \begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 $			for (j=0; i<11; i++) for (j=0; j=0; j=4, j=4, j=4, j=1, j=+)	
$\begin{cases} op, sat, write, scalar(') to at' at all junc * b(t, mo(N'* op, sim, time()); \\ \text{ op, sat, write, scalar(') cot, ```` delay((double)processed_p(); ``` op, sat, write, scalar(') cot, ``` delay((double)processed_p(); ``` double ``` to'` double ``` to'` at a cot, ``` op, sat, write, scalar(') cot, ``` double ``` to'` at a cot, ``` a cot, ``` op, sat, write, scalar(') cot, ``` double ``` to'` at a cot, ``` a cot, ```` a cot, ``` a cot, ```` a cot, ``` a cot, ``` a cot, ``` a cot, ```` a cot, ``` a cot, ```` a cot, ``` a cot, ``` a cot, ``` a cot, ```` a cot, ``` a cot, ```` a cot, ``` a cot$		~_~	if (slo(i)[j],src!=0) time+=(op_sim_time()-slo(i)[j],ciunc):	
30 op. stat. write. scalar(r or - xit. op. stat. write. scalar(r or - xit. or). 31 op. stat. write. scalar(r or - xit. or). 32 op. stat. write. scalar(r or - xit. or). 33 [_investe (x.1)] abolice x. x1:: double x. x1:: 40 Yr. pril(r - 4100; + 110; 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,	i	```	op.stal_write_stalar(*1oad*tutal_time*stot_no/(N*op_stm_time())); op_stal_write_stalar(*cu*time*stot_no/(N*op_stm_time()));	
<pre>20 0p.att.ortic_start(cg - st, trains, tf, train</pre>			op_stat_write_scatar(*0*.x); op_stat_write_scatar(*0*.x); op_stat_write_scatar(*0*.x);	
<pre>33 [_inverse (x1)] 40 ************************************</pre>	م	>	ດງ.sat.whne.seatar(103 - 5). ການທີ່ໃ* ຈັດເບດີ ກຳເປັນ ອຳເປັນ ອຳເປັນ ອຳເປີນ ການທີ່ໃ* ຈັດເບດີ ກຳເປັນ ອຳເປັນ ອຳເ	
<pre> 2 [inverse (x,1) 2 double x*2;: 4 double yapthp142.p3.p4.d0q142.q3.q4; 4 double yapthp142.p3.p4.d0q142.q3.q4; 4 x=x2; 4 double yapthp142.p3.p4.d0q142.q3.q4; 4 x=x2; 4 p=-0.322323108860; p1=-10, p2=0.3422420885470; p5=-0.02042112102560; q4=0.35607006546-2; 4 p=-0.322324108860; p1=-10, p2=0.3422420885470; p5=-0.020421121024560; q4=0.35607006546-2; 4 p=-0.322324108860; p1=-10, p2=0.3422420885470; p5=-0.020421121024560; q4=0.35607006546-2; 4 p=-0.322324108860; p1=-10, p2=0.3422408560; q1=0.385870506546-2; 4 p=-0.322324108860; p1=-10, p2=0.34224085660; q1=0.385870506546-2; 4 p=-0.32524108660; p1=-10, p2=0.34224085640; q1=0.385607066546-2; 4 p=-0.32524108660; p1=-10, p2=0.34224085640; q1=0.38560706546-2; 4 p=-0.325241036967; p1=-10, p2=0.342240660; q1=0.3858777525660; q1=0.38560706546-2; 4 p=-0.3252140; p1=-0; 4 p=-0.32540; p1=-0; 4 p=-0</pre>				
<pre>40 41 42 43 43 44 44 45 45 45 45 45 45 45 45 45 45 45</pre>	n	n	[_inverse (x,z)] double _x.*21:	
$\begin{cases} x = -x^{2}, \\ p = -0.322324108866, p = -10, p = -0.3422420884370, p = -0.02042121024560, p = -0.4554422101486-4; \\ p = -0.322324108866, p = -10, p = -0.3422420885470, p = -0.02042121024560; q = -0.355607006546-2; \\ p = -0.3250750666, q = -0.3885815709566, q = -0.385607006546-2; \\ p = -0.27507606, q = -0.3885815709566, q = -0.385607006546-2; \\ p = -0.27507606, q = -0.3885815709566, q = -0.385607006546-2; \\ p = -0.27507606, q = -0.3858815709566, q = -0.385607066546-2; \\ p = -0.2750266, q = -0.38568715709566, q = -0.385607066546-2; \\ p = -0.2750266, q = -0.3858815709566, q = -0.385607066546-2; \\ p = -0.275266, q = -0.3858815709566, q = -0.385607066546-2; \\ p = -0.275026, q = -0.3858815709566, q = -0.385607066546-2; \\ p = -0.27526, p = -0.47607, p = -0.476767, p = -0.476776, p = -0.2351103462, q = -0.385607065346-2; \\ p = -0.27526, p = -0.47676, p = -0.47676, p = -0.42676, q = -0.385607065346-2; \\ p = -0.27526, p = -0.2756, p = -0.47676, p = -0.4266, q = -0.2358157095, p = -0.23581570, p = -0.23581670, p = -0.23581570, p = -0.23581670, p = -0.23581670, p = -0.23581670, p = -0.2358170, p = -0.235817$			է double y.p.p0.p3.p4.q0.q1.q2.q3.q4:	
4.5 p=x if (p=0.5) p=1.0+; y=q=(0+0)(7)); *1=y=(0+0)(7)); 50 if (z=0.5)*2=*2i; if (z=0.5)*2=*2i; if	1	0	x=1-x0; p6=0.3223243108864; p1=1.0; p2=0.34224208854760; p5=-0.020423121024564; p4=-0.45364221101486-4; q0=0.09934846260664; q1=0.5858157049564; q2=0.5311024623666; q3=0.155577528564; q4=0.3856070065	e-2:
<pre>50 50 50 51 52 52 53 53 54 55 55 55 55 55 55 55 55 56 56 56 56 56</pre>		Ś	p=x; p=x,1) p=1.0p; v=(p-v=v=v=v=v=v=v=v=v=v=v=v=v=v=v=v=v=v=v=	
<pre>50] 51 serial_col(np1) 52 int</pre>			୨=qutrebgP ม). *21=y+(fp2+y*(g1+y*(g2+y*(g3+y*p4))))/(q0+y*(q1+y*(q2+y*(q2+y*(q3+)))): if (x0) *21=*21) *21=*21	
<pre>serial_col(np1) int n. double "p1:</pre>	5	0		
55 double "p1: in ij,k.b.; double p.x.1.22.c.1.23.s.1.32.y(400): b=nex_gav(n; k=n2, x_i=0; x_i=0			serial_col(np.)) seria_col(np.))	
$\begin{cases} \text{int} i_{j} k,h; \\ \text{double} p_{k,X1}, 2_{k,C1}, 2_{k,S1}, 3_{k,Y} 400 ; \\ \text{b=nex}, g_{k}n_{k}, e_{k}n_{k}, 1_{k}-h, x_{k}-h, \\ \text{for}(a_{k})(x_{k}+n), \\ \text{for}(a_{k})(x_{k}+n) \\ \text{for}(a_{k})(x_{k}+n) ; \\ \text{for}(a_{k})(x_{k}+$	-2:	5	double *p1:	
<pre>60 benext.stad/n.k=n/2.x=0.x1=0.x2=0; for (i=Di=cni++)</pre>			iri ijikib: double p.x.al.a2.cci.c2.asi.s2.yi400);	
y(j=0; for (=4);4c;j++) y(i]+=sad(*b+j): for (=4);y(ib;x+=y(i): j(i=y(ib;x+=y(i): if (<ck) cise="" x(1+="y(i):" x(2+="y(i):<br">if (<ck) cise="" x(1+="y(i):" x(2+="y(i):</td"><td></td><td>Q</td><td>benetx. Stat/rs: $k=n/2$: $x=0$; $x_1=0$; $x_2=0$; for $(i=0)$; $i=0$; $i=0$;</td><td></td></ck)></ck)>		Q	benetx. Stat/rs: $k=n/2$: $x=0$; $x_1=0$; $x_2=0$; for $(i=0)$; $i=0$;	
		~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	(1)=0. for (j=0; ckyj++) y(i]+=stat(j*b+j): y(i]=y(i)0: x+=y(i]: f(cick) t1+=y(i]): clas x2+=y(i]: }	

Process Model Report: mm1ql_scheduler	Fri Jan 7 04:56:55 1994	Page 5 of 8
M/M/1/Q/CL Uniform Traffic System		
scheduler Model		



|--|

Proces	ss Model Report: mm1ql_scheduler	Fri Jan 7 04:56:55 1994	Page 7 of 8	
WW1 schedi	(Q/CL Uniform Traffic System uler Model			
· · · · · · · · · · · · · · · · · · ·	x=x/n: xl=x1/k: x2=x2/k: s=0, s1=0, s2=0, c=0, c1=0, c2=0, (or (=0):crit++)			
10	,			
	s+=]*]; if (i<(n-1)) c+=]*(y(i+1)-x); if (i <k)< td=""><td></td><td></td><td></td></k)<>			
75	j=y(i]+xl : sl +=j*j; if (i<(k-1)) cl +=j*y(y(j+tl-x1):			
	else			
<u> </u>	[j=y(i *x2: s2+=j*(; if (i<(n-1)) c2+=j*(y(j+1]-x2):]			
	if (s==4) *p1=0: clsc *p1=2*c/5+(c1/61+c2/s2)/2: }			
8	col(n.p1)			
2	double *p1:			
	int ij,b; double p.x.s.y(400);			
2	b=next_stat/n; x=0; for (j=0;i <n;i++)< td=""><td></td><td></td><td></td></n;i++)<>			
	()=0; (0r(j)=0;4h++) (0r(j)=0;4h++) y[i]=y[i]0:x+=y(i]:			
	<pre>x=dn:s=0; t=dn:s=0; for(i=0.5xxii+) { [j=y(i] x: x+=]²; } s=s(n-1): x=sqn(s(n): ii (s=u) ? p1=0; cise "p1=x[*]9'x; piniu(i *ud: ut\utue\n.suL.cut.[*]p)); }</pre>			
<u> </u>) ini find_resource(packet) Packet* packet:			
<u> </u>	l it i jan furps på citas cå af as ad ca [8] an double svc: extern tar no_in_svc: extern charne_agen stor[130][10];	81,as(21,sc.dt.		
	op_pk_nfd_get(packct.*src*.≻): op_pk_nfd_get(packet.*	it ".&di);		
<u> </u>	ap=-1: ps=-1: pd=-1: for (j=0:j <t1:j++) *="" <="" column="" each="" for="" td=""><td></td><td></td><td>_</td></t1:j++)>			_
12	5 (i==0) cf=0; cs=-1; cd=-1;			

Process	s Model Report: mm1ql_scheduler	Fri Jan 7 04:56:56 1994	Page 8 of 8
M/M/1/C	0/CI 1 Initorm Traffic System		
schedul	er Model		
_	for (j=0;j <wavelength_no;j++)< td=""><td></td><td></td></wavelength_no;j++)<>		
	ff (slot(1)[], strc=u) { (ca(cf)=j; cf+=1; } if (slot(1)[], strc=u) (ca=; if (slot(1)], dst==dt) cd=; }		
135	if (i==(T1-1)) { ns=-1: nd=-1: } clse		
	f=0: ns=-1; nd=-1; for (j=0; v=wetength, no;)++) if (slot(j+11[]), arc=0; [ndn1]=j; nf+=1;] if (slot(j+11[]), arc=2; ns=1; if (slot(j+11[]), arc=2; ns=1; nd=1; if (slot(j+11[]), arc=2; nd=1; nd=1;		
145	-		
	if (cf==0 cs!=-1 cd!=-1) goto next_col:		_
150	$\begin{array}{l} for (j=0)_{j}cr(j] + +) \\ f(=) \\ f(=) \\ f(=) \\ f((y) = -1) \\ f((y$		
160	it ((fibag) (as(0)=i: as(1)=ca[j]: ap=fit:) if (ap==4) goto assign: 1		
165	next_ot: if (i!=(T1-1)) { p==x; pd=ct: cs=ns: od=nd; cf=nf; for (j=2x; pd=ct) : cs=ns: od=nd; cf=nf;		
170			
	if (ap=-1) return 0: clse f		
175	assign: stolated [1].sec=x: stolatol[1].a(1]).dst=dt. stolatol op. pk _infd_set(packet: "acvet=negth: stol[1]); op. pk_inf no_in_sect=1; processed_pkt=1; delay=ap_sim_time())] as 1].ctime=op_sim_time(): 1_set(packet.ttmesiot.as[0]): op_pk_creation_time_get(packet):	
180	op_pk_send_delayed(packct.0.svc);		

<u>M/M/1/Q/RL:</u>

OPNET reports for the *scheduler* and *release* processor models.



svc_time;

double

2

Ï

otal_pk:

2

20

\slot_no:

State Variables Objid init init init init init init

ន

R

Temporary Variables Packet* pkpr: int src.dst.ij; double ratio.temp;

Pro	ss Model Report: mm1qr_scheduler Fri Jan 7 04:26:57 1994 Page 4 of 11	Process Model Report: mm1ar scheduler	+
MM	//Q/RL Uniform Traffic System	MM/1/0/RI Inition Traffic Sustain	_
sche	tuler Model	scheduler Model	
<i>.</i>	tate 1: pk_prepare (CET's)	State 0: init (Enter Exers) fundation	Γ
<u> </u>	ET Cond: (1)	[_inverse(0.1.&2); /* 90% confidence interval */	1
	Trans: schedule	on ima sim attr set(OPC IMA INTEGED as & and not	
		9. Jma sim attr get(OPC_IMA_INTEGER, vr. & & wavelength_in); 9. op_ima sim attr get(OPC_IMA_INTEGER, vr. & & avelength_in); 9. op_ima sim attr get(OPC_IMA_INTEGER, vr. & & avelength_in);	
20	late 2: idle (Enter Execs) unforced, 35 lines	op_ima_sim_attr_get(OPC_IMA_INTEGER, 1.2. & siot_no);	
	if (lhad_steady)	own_idia=op_idd_set();scc_idia=op_idopo_int_assoc(own_idd.); oo inna ooli attr zee(sccid:int:interarrival_arrss*Aaax sro(0));	
	temp=total_time*stol_no(N*op_sim_time()*toad_prev): if (temp>0.999 && temp<1.001)	10 arr_rate=alo((&arr_rag(0)); arr_rate=1/arr_rate:	
^	koad_suzady=1: tstart=op_sim_time(): qsizz=op_subq_stat(0,OPC_QSTAT_PKSIZE):	N=wavekragth_no*fm_size: svc_time=(double)N(double)slot_no:	
	queue=0: stat_ct=6: next_stat=800: op_intrpt_schedule_self(op_stm_ttme()+inc.0): }	NI=floor(svc_time); svc_time=(double)/m_size/(double)slot_no; 15	
	l lí (MEASURE)	<pre>svc_dist=op_dist_load('exponential'svc_time(0); src_dist=op_dist_load('uniform_int'1.(double)mode_mo); dst_dist=op_dist_load('uniform_int'1.(double)mode_mo);</pre>	
	<pre>temp=op.sim_time(): queue+=qsize*(temp-tstart); stal(stat_ct)=queue/inc: stat_ct+=1; queue=4b: tstat=temp; qsize==psubh_stat(0.OPC_QSTAT_PKSIZE); if (stat_ct==nex_stat)</pre>	20 inc=10; total_pk=0; processed_pk=0; no. in_svc=0; time=0; total_time=0; dotal_time=0; dotay=0; load_prev=arv_rate/(dotable)/wavelength_no; load_steady=0; end_stim=0; n0=600; n1=800; op_subq_fhush(0);	
	l step2:	for (i=0;i<=N1;i++)	
	scrial_col(400.&temp); if (temp>0.4) goto step5.	25 { { } { } { } { } { } { } { } { } { }	
ř.	if (temp<0) goto step4: step3:	for $(j=0; j<15; j++)$ (other[i][j], stot=0; other[i][j], wavelength=0; other[i][j], len=0;)	
	serial col(200,&raio); if (ratio:stemp) goto step5:	for (=dix(=m_size:i++) 30 for (=d); <eventement_morphite)< td=""><td></td></eventement_morphite)<>	
57	sterpt: col/d0.&tterp): if ((ctende(1)) cted star=1:	sociality=0.	
	step5.	Quete 0. 1-1-0000-1-1	ſ
	u (uozu) { n1=n *: cick(_sia=n); } cick (n0=n0*2; nexsia=n0; }	State U: Init (JET S) CET Cond: (ARRIVAL)	-
ň	op_intrpt_schedule_self(op_sim_time()+inc.0);	#0 Exec: : Tans: pk_prepare	
	if (end. sim && : ENDSIM)	CET Cond: (defaul) #1 Exec: : Trans: idle	
ć	op.xm. tuo('reachnig steady state'''''''''		
S	ite 2: idle (CET s)	State 1: pk prepare (Enter Execs) forced, 10 lines	ГТ
õ	T Cond: (ARRIVAL)	src=(nt)op_dist_outcome(src_dist): next:	
*	Exec: : Trans: pk_prepare	$dst=(int) pop. dist_outcome(dst_dist),$ if $(dat == a ex_0 = a hoto meets)$	
<u>0</u> #	T Cond: (default) Fran-	5 temp-op_dist_outcome(svc_dist);	
	Tans de	pkptr=op_pk_get(op_intrpt_strm());	
2 5	Lona: (ENDSIM) Exec: record.stats():	op_pk_ndd_set(pkpur "src" src): op_pk_ndd_set(pkpur" dst " dst " dst), op_pk_ndd_set(pkptr." svc" temp);	
3	Trans idle 1 Cond NEXTO & A RRIVAL)	10 total pk+=1: total time+=temp:	
÷	Trans schaduld		

Proc	cess	: Model Report: mm1qr_scheduler	04:26:58 1994 Page 6 of 11
MW.	110	J/RL Uniform Traffic System	
Ś	edul	er Model	
		extern double time: extern channel_agn pkt[1025];	
	10	$s=(double)stat_cu(double)H0$; $b=floor(s)$; $x=0$; for $(i=0;i<40;i++)$	
		{ 1}]=0; for (j=0;j=0;)++) y[i]+=stat[i+b+j]; y[i]=y[i]/b: xi=y[i];	
	2	x=x/40; s=0; for (q=is=dbi++) { j=y(i) x: s+=j*j; } s=29; if (xi=0) s=2*eqfi(s40)K;	
	07	for (i=1. i<=N1: i++) if (pkt(i].src=d) inne+=(opsim_time()-pkt(i].cume):	
	25	<pre>op_stat_write_scalar(*load_time*slot_no(N*op_stm_fime())); op_stat_write_scalar(*cu:time*slot_no(N*op_stm_fime())); op_stat_write_scalar(*o",*);</pre>	
1.000 C	30	op_sat_mite_stain(*) delay((double)processed_pK); op_sat_mite_stain(* of *.s); op_stat_write_stain(* c*.stat_ct); print(* eA\tCp: *K\tCp: *K\tCp: *K\tCaev: *K\ta*.stat_ctx.(delay/processed_pk),s	
	35	f_inverse (x,z1) double x,zz1: (
<u> </u>	40	ძითს დ. ჯაღსის განანანოს კანაკაფა. x=1-x/2; p0=.032222431088ch; p1=.10; p2=0.342242088547cb; p3=.0.0204231210 დ0=0.09934842606cch; q1=0.8885815704956ch; q2=0.351103462566ch; q3=0	45ch; p4=0.453642210148e-4: 0353775285ch; q4=0.38560700634e-2:
A.C	45	p=x; if (p=0.5) p=1.0-p; y=sqr(-log(p*p)); *21=y+f(p2+y*(p2+y*(p3+y*p4))))(q0+y*(q1+y*(q2+y*q2)))); f(x<0.5) *21=*21; }	
	20	scrial_col(n,p1) int n; double *p1:	
	55	l int ijikd: double paratazaciazasiazy(400);	
	99	b=next_stad/n; k=n/2; x=0; x1=0; x2=0; for (i=0):cn:i++) f(1=0; f(1=0; k=0; k=0; k=1); f(1=0; k=0; k=1; k=x=0; k=1; k=1; k=1; k=1; k=1; k=1; k=1; k=1	

Process Model Report: mm1qr_scheduler	Fri Jan 7 04:26:57 1994	Page 5 of 11
M/M/1/Q/RL Uniform Traffic System		
scheduler Model		



Sta	ite 3:	schedule (CET's)
S	r Cond:	(1)
¥	Exec:	
	Trans:	idle
3	nction Block	320 lines

	record_stats()		l
	_		
	int	i.j.b.	
	double	x.s.y[40];	
S	extern int	slot(130)[10];	

Report: mm1qr_scheduler Fri Jan 7 04:26:59 1994 Page 8 of 11	Process	Model Report: mm1ar scheduler E1.Jan 7.04:26:58.1994 Page 7.of 11	
iform Traffic System	M/M/1/C schedul	J/RL Uniform Traffic System	
(j=0; j≤wawelengsh_no: j++) empy(j)=0; (j=0; j≤msize; i++)	65	ן x=v/a: x1=x1/k: x2=x2/k: s=0; s1=0; s2=0; c=0; c1=0; c2=0; for (i=05:qn:i++)]
contact(i)[0]= 1: contact(i)[1]= 1: for (k=0; k <vvertength_nor, k++)<="" td=""><td>20</td><td>j=y(i)+x: s+=j*); if (s(0=1): c+=j*(y(i+1)-x): if (i<k)< td=""><td></td></k)<></td></vvertength_nor,>	20	j=y(i)+x: s+=j*); if (s(0=1): c+=j*(y(i+1)-x): if (i <k)< td=""><td></td></k)<>	
if (stot(i) k =0) empty(k+=1; if (pkt(stot(i) k).sc==sc) conflict(i)[0]=k; if (pkt(stot(i) k).dst==dt) conflict(i)[1]=k;)	75	[j=y(li+xl:sl+=j*ty(j+1}-xl): if (i<(k-1)):cl+=j*ty(j+1}-xl): clsc	
go_back=0: if (conflict[i](0)==1 && conflict[i][1]==-1) f (f=0: k-wavelength_no: k++) for (s=0: k-wavelength_no: k++)	80	$\begin{cases} \frac{1}{2} x (1+x^2, x^2+y^2)^* \\ 1 x (1+x^2, x^2+y^2)^* \\ 1 x (1+x^2) + \frac{1}{2} x \\ 1 + x^2 \\ 1 \end{cases}$	
if ((slot(j) k]=0 li==(fm_size1)) & & empy(k)=0) { ((slot(j) k)=0) j=:empy(k)=1 : else j=i=empy(k): if (j=0) oh=0:	85	$\pi_1(s=0), p_1=0;$ $p_1=2^* v_2 s_1(c_1(s_1+c_2)s_2)/2;$	
cise if ((conflict[j][0]==-1 l conflict[j][0]==k)&&(conflict[j][1]==-1 l conflict[j][1]==k)) ah=0; else ah=2; if ch==2 && empy[k]<2) empy[k]=0; confinue:] iff ch==0) if (ch==0)	8	col(r.p.1) int n: double "p1: [int.j.b: double p.xs.y(400):	
if (shot j k)=0 lt go_back) store[0][0]=i=empty[k]; etse store[0][0]=i=empty[k]+1; store[0][1]=k: store[0][2]=empty[k]; store[0][3]=oh.	95	b=next, stad/n; x=0; for (i=Oisiani++)	
cke ' for (j={(x-1); j>=0; j) if (store(j)[2]<=mpy(k) # (store(j)[2]==empy(k) && store(j)[3]>oh))	100	y(1)=0: for (j=0;4:4:) y(1)=y(10; x+=y(1); }	
<pre>ktrif+1[0]=store[j](0]: store[j+1](1]=store[j][1]: store[j+1][2]=store[j][2]: store[j+1][3]=store[j][3]:</pre>	105	<pre>x=vtn: s=d. f(i=0:ca:it+i) f(j=y(i):x:s+i)*j: s=s(u=1)*p[=0:clss*p]=z*s: f(i(s=0)*p]=0:clss*p]=z*s: print(*a: *t:vt:vt:n*sal_ctx*p]);</pre>	
if (slot(1)(k))=0 1go hack) store(j,+1)(0)=i-empoy(k) cise store(j+1)(0)=i-empoy(k)+1): store(j+1)(1)=k: store(j+1)(2)=empoy(k): store(j+1)(3)=oh: break:)	110	int find_resource(packet) Packet: "packet:	
if (go. back) []=1; goto store_ct;] if (go. back) goto back: empty[k]=0; ct+=1:]	115	(i, i, k.ap.ap.l.ad.ct.ob.go. back.wastei_ct.ft.i.next.sc.dt: int conflict(1301/2).empty[10].store[1025][4].i.emp[130].asgn[130].mb[130].mp_mb[130]: double svc: cvc: extern int no_in_svc:scal(130[10]: extern int area of 1025[115].	
cise f for (k=f: k-wavelength_no: k++)	120	extern channe_segn pk(0.25]; 0p_pk_nfd_gettpacket 'src .≻); 0p_pk_nfd_gettpacket 'dst .&dt), 0p_pk_nfd_gettpacket 'svc .&svc); ct=0;	

clse for (j=(ct-1); j>=0: j--) { if (store[j][2]<cmpt if (slot(i][k]==0) cmpty[k]+=1; if (pkt[slot(i]][k]], src==sc) conflict(i][0 if (pkt[slot[i][k]]], dst==dt) conflict(i][1) if (slot(i][k]=0) j=j=empy[k if (j<0) oh=0; els ((conflict[j][0]==-1 || c else oh=2; if (oh=2; insert=2 && empy[k]<2) (insert=1 (c==0) if (c==0) { store[j+1][0]≕ store[j+1][2]= } if (store[j][2]>emp if (slot(i][k]!=0 || go_ba clsc store[0][0]=i-c store[0][1]=k: store } store_ct: if (slotfil[k]!= else stor store[j+1][1]= go_back=0: if (conflict(i][0]==-1 && conflict(i][1]== if ((slot[i][k]!=0 || i==(fm_sizefor (j=0: j<wavelength_no: j++) empty(j)=0;</pre> { for (k=0: k<wavelength_no: k++) conflict(i][0]=-1; conflict(i][1]=-1; for (k=0; k<wavelength_no; k++)</pre> for (i=0;i<fm_size;i++) 125 130 135 140 145 150 155 160 165 170 175

180

Process Model Report: mm1qr_scheduler MW/1/Q/RL Uniform Traffic System scheduler Model
Pro	cess	Model Report: mm1qr_scheduler Fri Jan 7 04:26:59 1994 Page 9 of 11
N de	V1/0	/RL Uniform Traffic System
, N		
		if (cmpy(k)>0)
	185	ii (sloti]Rh==0) canpy[k];=1: ii (canpy[k]==0) continue: j=i-ampy[k]-1: ii (j=d) viets
	961	cks ((conthet)[10]=-1 confict[j][0]=±k)&k(conthet[j] 1]=-1 confict[j][1]==k) oh=0. else oh=2. if ((conthet)[i][0]=-1 confict[i][0]=±k)&k(conthet][i] 1]==1 confict[i] 1]==k); oh=oh: else of ((n)==0) oh=1 - (n) oh=2.
	195	<pre>if (((oh==1)&& empty k <2) (oh==3 && empty k <3)) { empty k =0; continue: } go.back=1: gooi insert: back: back: compatibulation: for entry is a set of the set</pre>
	200	
	205	nd=slot_not_i_ct=d>, waste=d: next=d>, agn[0]=-1; agn[1]=d). for (i=Q: i <dm_size: [=""]<br="" i++)="" mb[i]="0;" unp_mb[i]="0;">for (i=Q: i<cc: i++)<="" td=""></cc:></dm_size:>
	210	if (store[j]]]==0) fit = 0: ets: if (store[j]]]==3) fit = 2: ets: fit = 1: j=store[j]]2-fit:
	215	if (next) () if ()=nd)
····.		k=pan+Nt: if (fic= ap) && k cap) for (ob=store[i][0]; obc(store[i]][0]); ob++)
	220	if (mb[oh]) (k=1: break:) if (k=-1) f
	225	ap=k: apl ≤fit: temp[i_ct]=i: for (obs./or size: op+1-1, mp_mp[ob]=0; k=senct[i](0p+i:i (t<0) k=0; go_bad<=suce[i](0p+suce[i](2p+1:if (go_bad>fin_size) go_badk=fin_size: or (ob=k: ob <go_badk; ob++)="" unp_mp[ob]="1:</td"></go_badk;>
	230	از (از (جماط ال:=(د-1)) از (جماط ال:=(د-1))
	235	waste+=apl: i_c(+=1: if (waste-agn[0]] a sgn[0]==-1) f ggn[0]=waste: agn[1]=i_c t: for (k=2: kc+(_ct+2); k++) asgn[k]=emp[k-2]:
	240	ut axgn(U) <u>→</u> U) goto axsgn:) if (j <nd)<="" td=""></nd>



e Fri Jan 7 04:35:29 1994 Page 1 of 1			
Process Model Report: mm1qr_releas	M/M/1/Q/RL Uniform Traffic System	release Model	

ber of	Header	State	Temporary	Function
es	Block	Variables	Variables	Block
	Yes	£	Yes	Ŷ

Hei	ader Block	17 lines	
	typedef struct		
	int src;		
	int dst;		
S	int len;		
	double ctime:		
) channel_asgn;		
	typedef struct		
10	int slot:		
	int wavelength:		
	int len;		
	} other_asgn:		
	extern int	no_in_svc.slot[130][10];	
15	extern double	time;	
	extern channel_asgn	pkt[1025];	
	extern other asgn	other[1025][15];	

porary	Variables	3 lines
Packet*	pkptr;	
ũ	wavelength.timeslot.i.j.k;	
Obiid	own id.src id:	

discard (Enter Exace)	pLtype()==OPC_INTRPT_STRM)	enp_pkk_get(op_intrpt_strmt)): k_ind_get(jskpt.*vavelaenyth.&&vavelength):op_pk_indd_get(jskpt.*t.imeslot.&×lot): timeslot[jvavelength]:time+=(op_sim_time()-pkt(it].ctime): 40:j <pkt[j].kor.j++)< th=""><th>or (k=oher[i]U] slot: k=(oher[i]U]) kot+oher[i]U] ken; k++) slot[k]oher[i]U] wavelength]=0: her[i]U] slot=0: oher[i]U] wavelength=0; oher[i]U] ken=0; sr=0; pkt[i] dst=0; pkt[i] kenne=0; no_in_zve=1: sp_pk_destroy(pkpro); dd=op_id_zef(D; src_id=op_topp_in_zesso(covn_id:0); op_intrpt_schedule_remote(op_sim_time().1.src_id);</th></pkt[j].kor.j++)<>	or (k=oher[i]U] slot: k=(oher[i]U]) kot+oher[i]U] ken; k++) slot[k]oher[i]U] wavelength]=0: her[i]U] slot=0: oher[i]U] wavelength=0; oher[i]U] ken=0; sr=0; pkt[i] dst=0; pkt[i] kenne=0; no_in_zve=1: sp_pk_destroy(pkpro); dd=op_id_zef(D; src_id=op_topp_in_zesso(covn_id:0); op_intrpt_schedule_remote(op_sim_time().1.src_id);
e O: disc	if (op_intrpt_type()==	pkptr=op_pk_get op_pk_nfd_get(p i=stot{timestot][w for (j=0; j <pkt[i].1< th=""><th>for (k=other[other[i][j].slc } pkt[i].src=0: pkt[i own_id=op_id_se</th></pkt[i].1<>	for (k=other[other[i][j].slc } pkt[i].src=0: pkt[i own_id=op_id_se
Stat		s.	10

* CET	le 0: Cond: Exec:	discard (CET's) ()
	Irans:	discard

Process Model Report: mm1qr_scheduler	Fri Jan 7 04:26:59 1994	Page 11 of 11
M/M/1/Q/RL Uniform Traffic System		
scheduler Model		



B.4 Two Class, Uniform Traffic System

The termination criterion used for M/M/2/B/L+1, M/M/2/B/CL, and M/M/2/B/RL systems is to end the program when either 99999 seconds have elapsed or the "steady state" condition has been reached. The program starts to monitor periodically the channel utilization (cu), the blocking probability for the system and for each type of the sessions (pb, pb_1, pb_2) once the measured load is within 1% of the offered load derived in Section 4.1. If the newly measured cu, pb, pb_1 , and pb_2 are within 1% of the previous measured values and the total number of sessions processed by the system is more than 20,000, we say that "steady state" condition is reached. The program will be ready for termination once the measured load is within 5% of the offered load.

The seed used for each simulation has value NWs, where N is the number of users, W the wavelength channels, and s equals to one if the offered load is normalized against sessions with smaller of the two throughput requirements and zero otherwise (refer to Section 4.1). So for N equals to forty, W equals to eight and s equals to one, the seed is 4081.

B.4.1 Blocking System

<u>M/M/2/B/L+1:</u> OPNET reports for the scheduler and release processor models.



Proce	Model Report: mm2bf1_scheduler Fri Jan 7 05:31:26 1994 Page 3 of 7	Process	Aodel Report	mm2bit scheduler	Fri Jan 7 05:31:26 1994 Page 2 of	12
WW ² sched	/L+1 Uniform Traffic System	M/M/2/B/ schedule	-+1 Uniform	Fraffic System		
Ś	te 0: init (Enter Execs) forced, 36 lines	State	Variables		40 lines	
	/r assume Ll <= L2 4/		Dijid Dijid	Nown_id:		
	op_ima_sim_attr_get(OPC_IMA_INTEGERn. & node_no);			Vinode_no;		
	op_ima_sim_attr_get(OPC_IMA_INTEGERw. & wavelength_no):		Ħ	\wavelength_no:		
Ŷ	op_ima_sinattr_get(OPC_IMA_INTEGER, 'T', &fm_size);	2	Ħ	Vfm_size;		
	00_00035.00_0000000000000000000000000000		81	VIZ: telse mor		
	op inte sim attr seitOPC IMA INTEGER. s. & Assoc_nue)		= =	Velor mol:		
	op_ima_sim_attr_get(OPC_IMA_DOUBLE, a. &a);		1 2	vslot_no2:		
2		10	Ħ	LN:		
_	own_id=op_id_self(); src_id=op_lopo_in_assoc(own_id_0);		ti	NI:		
	op_inma_obj_attr_get(src_d.'interarrival args'.&arv_arg[0]);		ਸ :	IN2:		
	av rat=atof(&av aro(0)); av rat=1/av rate:		nt breihle			
15		15	louble	lsvc_time:		
	N≃wavekength_no*fm_size: svc_time=fm_size#((1+slot_no2)+(1+slot_no1)*a/(1-a)); T2=floor(svc_time);		thar	\arv_arg[20];		
	N2=T2*wavelength_no: svc_time=(double)(fm_size-T2*(1+slot_no2))(double)(1+slot_no1): T1=floor(svc_time);		louble	larv_rate;		
	Nl=Tl*wavelength_no;		Distribution	*\src_dist;		
20		5	Distribution	**dst_dist;		
3	II (\$20 - 1) \$2-1; 12 (\$10 - 1) \$2-1; 13 (\$10 - 1) \$2 - 10; 10 - 1	07	Distribution	* Seve_dist:		
	a (asoc.nor-asoc.nor) s >==) svc_mmc=(uousic.jum_sizze(couosic.jsu_goz. else svc_time=/double/fm_size//double/stor_mo1*		one int	Viis_uist; Vhicebard at 1:		
			ong int	Volocked pk2:		
	svc_dist=op_dist_hoad(*exponential*svc_time.0);		ong int	vtotal_pk1;		
52	<pre>src_dist=pp_dist_lowed(`uniform_int`.l.(double)nock_no);</pre>	25	ong int	Viotal_pk2;		
	dst_dist=op_dist_howd(*uniform_int*,1.(double)mode_mo);		louble	keu_prev;		
	CK_DIST=DP_QUS0000("Dernoull1'.a.U); had mevundstelv nolit('sivelvi nov)***** rutateuv imma//doukla)N:		louble buble	vob_prev:		
			louble	vol_prev.		
8	blocked_pk1=0: blocked_pk2=0; totai_pk1=0: totai_pk2=0; released_pk1=0; released_pk2=0;	30	louble	Voad_prev;		
-	no_in_svc1=0: no_in_svc2=0; time1=0; time2=0;		n	pk_count1;		
	cu_steady=0; pb_steady=0; pbi_steady=0; pb2_steady=0; load_steady=0; ready=0;		Ħ	\pk_count2:		
			= 1	Vr_count1;		
35	ing (i=0,cc,t,t,t+t,t+t) for (i=0)i≤wavelength mori+t)	22	= =	ven steady:		
	<pre>(slot)[1] sread(1)[1] ds=0; slot(1)[1] crime=0;</pre>	3		why steady:		
J			. 2	vpb1_steady;		
			E	\pb2_steady:		
Ĕ			Ħ	Voad_steady;		
5	e 0: init (CET's)	40	ŧ	vready;		Ì
បី	Cond: (ARRIVAL)					
¥	Exec: :					
12		Tem	oran Varis	hlae	3 linas	
3 5	Cond: (detault)		Mat [*] abut	000	011110	
;	Trans: Idle		attact pape.	ist.tvne:		
J			iouble temp.r	atio:		

Sta	te 1: pk prepare (Enter Execs) forced. 8 lin
	src=(int)op_dist_outcome(src_dist);
	acxt:
	dst=(int)op_dist_outcome(dst_dist);
	if ($dst = src$) goto next:
ŝ	
	type=(int)op_dist_outcome(cls_dist);
	$[if(type=1) total_pk1+=1; clse total_pk2+=1;$
	pknt=op pk set(op introt strm());

Chedu	/B/L+1 Uniform] uler Model	raffic System	
ŝ	ate Variables		40 lines
	Objid	lown_id:	
	Ohjid	\src_id;	
	int	Vnode_no;	
	iit	\wavelength_no:	
Ś	int	Vfm_size;	
	int	VT2;	
	int	Valor_no;	
	int	kiot_no1;	
	int	kslot_no2;	
2	int	N:	
	Iu	SN1:	

int worde_not. int Virg. int Virg. int Virg. int Virg. int Virg. int Virg. int Virg. int Virg. int Virg. int Virg. char Virg. char Virg. barrhoution Virg. Distribution Virg			
int Workengelt, Doc. int VLL, arkers int VLL, arkers int VLL, arkers int VLL, arkers int VLL, arkers int VLL, arkers int VLL, arkers double VLL, arkers Conche VLL, arkers Distribution VLL, arkers Conche VLL, arkers Distribution VLL, arkers double VLL, prevers double VLL, prevers doubl		int	knode_no:
 int (171. size: int (172. size: char (172. size: char (172. size: Conde (172. size: Distribution (172. size: Distribution (172. size: Distribution (172. size: Distribution (172. size: Line: conde (172. size: conde (172. size: co		ij	\wavelength_no:
int NT2: int NSC. DOC: int NSC. DOC: int NSC. DOC: int NSC. DOC: int NSC. DOC: int NSC. DOC: int NSC. DOC: double NSC. Inter- tor NSC. Inter- double NSC. Inter- berthultion NSC. dist: Distribution NSC. dist: NSC. dist: Long int NSC. dist: Long in		int	Vfm_size;
int bioLos; int Victor; int Victor; int Victor; int Victor; int Victor; int Victor; double Victor; Distribution Victor; Long int Victor; double Victor; double Victor; double Victor; int Victor; int Victor; int Victor; victor;		int	112:
int biol		int	kslot_no;
int Nichords Nichold N		int	kslot_no1:
 and N1; and N1; and N1; and N1; and N2; and N2;		int	klot_no2;
ini NN1: ini NN2: donbie w. Mv2.mic. donbie w. mv2.mic. Distribution w.r. mic. Distribution w.r. dist. Distribution w.r. dist. double w.r. prev. double Wa2.prev. double Wa2.prev	0	int	N:
att VV2: donkte VV2: chorbite VV2: chorbite VV2: Destribution VV2: double VV2. double		Ē.	WI:
int and the second seco		int	W2:
double wr_argine; double wr_argine; Diseribution wr_argine; Diseribution wr_argine; Diseribution wr_argin; Diseribution wr_argin; Diseribution wr_argin; brog int buicted_gk1; korg int buicted_gk2; korg int buicted_gk2; korg int buicted_gk2; korg int buicted_gk2; double by_grev; double		int	55
 double bwc_inne; double bwc_aits: double bwc_aits: Distribution wc_aits: borg in Underlykt; borg in Underlykt; bord pint wcal_pkt; bord pint wcal_pkt; bord pint wcal_pkt; bord pint wc_annt2; b		double	la:
char war_arter double war_arter Distribution *ac. dat: Distribution *ac. dat: Distribution *ac. dat: Distribution *ac. dat: Distribution *ac. dat: Distribution *ac. dat: Distribution *ac. dat: browg int bucked_gkt; kong int bucked_gkt; double by_prev; double by_prev; bit by_prev;	Ś	double	lsvc_time;
double Var. Ame: Distribution Var. dist. Distribution Var. dist. Distribution Var. dist. Distribution Var. dist. Distribution Var. dist. Nong int Voord_pt1; long int Voord_pt2; long int Voord_pt2; double Vp1_prev; double Vp2_prev; double Vp3_prev; double Vp3_p		char	karv_arg[20];
Describution were date: Describution were date: Describution were date: Describution were date: brog int Vuoteder gat: long int Vuoteder gat: long int Vuoteder gat: long int Vuoteder gat: double ver_preve double ver_preve doubl		double	larv_rate:
Distribution Mat. dist. Distribution "Arc. dist. Distribution "Arc. dist. Distribution "Arc. dist. Dag in Ublocked_pk(: borg in: Ublocked_pk(: borg in: S long in: Word_stat.; Jong in: Word_stat.; double Vpl_prev. int Vc_countl.; int Vpl_prev.		Distribution	*\src_dist:
 Distribution *ex. dist. Distribution *ex. dist. Distribution *bicked_dist. long int blocked_dist. double block_count: blocked_dist. blocked_dist.<		Distribution	*\dst_dist;
Distribution "dcl_dist: long int blocked_gdk1; long int blocked_gdk1; double vural_gdk2; double vb_gb_grev; double vb_gb_gb_grev; double vb_gb_gb_gb_grev; double vb_gb_gb_gb_gb_gb_gb_gb_gb_gb_gb_gb_gb_gb	•	Distribution	*svc_dist
long int blocked pkt: long int blocked pkt: long int broad pkt: double by the prev; double by prev; double by prev; double by prev; double by pkt_prev; double by prev; double by pkt_prev; int by count; int by count; int by steady; int by steady;		Distribution	*cis_dist;
long int blocket JAC2: long int blocket JAC2: long int blocket JAC2: double by Larver; double by Larver; double by Larver; double by Larver; double by Larver; int by Larve		kong int	blocked_pk1:
long int Voad phi: double thur Voad phi: double by prev; double by prev; doubl		long int	blocked_pk2;
5 long int Word, PC2, double Vu, greer; double VM_, prev; double VM2, prev; double VM2, prev; double VM2, prev; int VL_count1; int VL_count2; int VL_coun		long int	kotal_pk1;
double Veruper; double Veruper; double Veluper; double Veluper; double Veluper; int Vercount; int Vercount; int Vercount; int Vercount; int Velupe; int Velupe; in	ŝ	long int	Votal_pk2:
double (vh.)_prev: double (vh.)_prev: double (vh.2_prev: nt (vh.2_prev: nt vh.c.comt1; int v.c.comt1; int v.c.comt2; int v.g.stady; int (vh.3_stady; int (vh.3_stady; int (vo.3_stady; int (vo.4_stady; int (vo.4_stady; int (vo.4_stady;		double	cu_prev;
double (val. prev: double (val. prev: auti (val. prev: auti (val. prev: auti (val. prev: auti (val. prev: auti (val. prev: auti (val. steady: int (val. steady: int (val. steady: auti (val. steady: auti (val. steady: attady		double	lpb_prev:
double (val. prev. double (val. prev. int (v. countl.; int (v. countl.; int v. countl.; int (val. steady; int (val. stead		double	ipb1_prev;
0 double Voad prev: int Vac count1; int Vac count2; int Vac count2; int Vac count2; int Vac staady; int Vac staady; in		double	ipb2_prev;
int pl.count: int pl.count: int bl.count: int v.count: int v.a.sount: int v.b.steady: int pb.steady: int pb.steady: int v.a.steady: int v.a.steady: i	0	double	Voad_prev;
int yb_count2: int u_count2: 5 int u_count3. int yb_steady; int yb1_steady; int yb3_steady; int yb3_steady; int yb3_steady;		int	ptk_count1;
int Vr. count1; int Vr. count2; 3 int Vr. steady; int yby steady; int yb2. steady; int Yb2. steady; int Vbod. steady; int Vbod. steady; int Vbod. steady;		int	pk_count2:
int Vr_count2: 5 int Vr_count2: int Vol_steady; int Vol_steady; int Vol_steady; int Vood_steady; of int Vood_steady;		int	Vr_countl;
5 int burgetady; int by basedoy; int yob_astady; int yob2_astady; int yob2_astady;		iit	\rcount2;
int yoh_steady; int yoh_steady; int yoh_steady; int Yuod_steady; of int Yuod_steady;	ŝ	iit	kcu_steady;
int VebL. steady: int Veb2_steady: int Veb2_steady; on inv		ii.	lob_steady;
int Vpb2_steady: int Vpb2_steady: on version:		int	pb1_stcady;
int Voad_stready:		ii	\pb2_steady:
D int headur		İİ	Voad_steady;
	0	int.	lready:

r Execs)	forced, 8 lines	

Process	Model Report: mm2bh1_scheduler Fri Jan 7 05:31:27 1994 Page 5 of 7	Process	Model Report: mm2bi1_scheduler Fri Jan 7 05:31:27 1994	Page 4 of 7
M/M/2/B	L+1 Uniform Traffic System r Model	M/M/2/F schedul	vL+1 Unitorm Traffic System er Model	
	else if (temp=0) ratio=1: else ratio=0. 27 cer:s=0.000 R. endic=1: 01001			
8	$ (100 > 0.777) \propto \propto 7.000) \qquad pull steady=1.008 pull provide the provided of the provid$	į		
	If ('hb2_steady)	Sta	le 1: pk prepare (CET'S)	
22	temp=(double)hocked_pk2/tdouble)utel_pk2: ti (pp2:ev=49) transmp/pk2_prev: clss — if (neuro=0)station=1, clss traite=0, ti (raito > 0.999 && raito < 1.001) pb2_steady=1; clss pb2_prev=temp;	0	Exec: : : Trans: schedule	
90	if (u. strady && ph_strady && ph1_strady && ph2_strady) ready=1: if (trady)	Sta	te 2: idle (Enter Execs) unforcec 1 ('load, steady && released pk1-0 && released_pk2-0)	. 80 lines
	ratio=61.temp=0. for (i=62, t <tt): i++)<="" td=""><th></th><td>{ raio=(total_pk1*stot_no1+total_pk2*stot_no2)*svc_time/(N*op_sim_timet)*toad_prev): if (raio>0.999 && raio<1.001)</td><td></td></tt):>		{ raio=(total_pk1*stot_no1+total_pk2*stot_no2)*svc_time/(N*op_sim_timet)*toad_prev): if (raio>0.999 && raio<1.001)	
65	for $ ac_{1}$, (ac_{1}) , $($	2	load_steady=1; ratio=0; temp=0; ov.ci.ev.st.cs.st.cv.st.cs.st.cv.st.cv.st.cv.st.cv.st.cv.st.cv.st.cv.st.cv.st.cv.st.cv.st.cv.st	
	ic (=0.1, ×.wavelength, no: ++) ic ([stot]][], sc:=0) ratio+=(op. sim_time()-stot]][], ctime);		for (j=0, j=0, v=0, v=0, mo; j++) for (j=0, j=0, v=0, v=0, v=0, j=0, v=0, v=0, v=0, v=0, v=0, v=0, v=0, v	
70	ratio=(slot_nol *(timel +temp)+slot_no2*(time2+ratio))(N*op_slim_time()): temp=ratio*(N1: %t_count=2*flot(note); t_count=2+lot(note); t_count=2+lot(): temp=ratio*(transformer); t_count=2+lot(note); t_count=2+lot	10	for $(1=T)$: $i_{n}(T)$, i_{n+1}); i_{n+1} ; for $(1=T)$; i_{n+1} ; i_{n+1} ; for $(1=T)$; i_{n+1} ; i	
			i (suqtij),sic:=0) auto=10p.sum_numer/>suqtij),counec. cu_prev=[slot_nol*(inite1+temp)+sic) non2*(inite2+temp)*kic) eh. sus=24eabbeerde at h21kiensheitenet at h21eabbeerde at 2000.	
75	if (ready && (total_pk1+total_pk2)>20000)	15	Purpretrements meterized in the resource structure structure and an international provident and an and and and and and and and and	k2; at1=floor(temp)+20; at2=floor(temn)+20;
	tempe(coal_pk1*s)c_no1+total_pk2*ist_no2)*svc_timv(N*op_sim_time());ratio=tempfload_prev: if (ratio=5956.&& tailo = 1.006.&& tENDS10; on oil menditriventing restarts=1	20		
80		24	ii (load_steady && !ready && released_pk1>(tr_count1*pk_count1) && released_pk2>(tr_count2*pk_coun	2))
			if (:cu_steady) f	
St	le 2: idle (CET's)	25	ratio=0. temp=0.	
₩ ¥	Cond: (ARRIVAL)		for (i=0; i <t1; i++)<br="">for (i=0; i<wavelength i++)<="" no:="" td=""><td></td></wavelength></t1;>	
2	Trans. pk. prepare		if $(s)(i)(j)(x)(x) = 0$, $temp+=(op_sim_time()-s)(i)(j)(j)$ ctime);	
ដ ដ	Cond: (default)	01	for (i=T1: i<(T1+T2): i++) for (i=C1: i <wavelength i++)<="" no:="" td=""><td></td></wavelength>	
	Trans: Idle	3	if (slot[i][],src1=0) rato+=(op_sim_time()-slot[i][j];ctime):	
锐	Cond: (ENDSIM) Exec: record_stats():		temp=(stot_not*(tume1+temp)+stot_not*(tume2+tatto))/(N*Op_sun_tume()); fatto=tempver_prev; if (ratios50.999 && ratio<1.001 && temp<(total_pk1*stot_not1+total_pk2*stot_not2)*svc_tume(N*	p_sim_time()))
	Trans: idle	35	cu_stcady=1:	
			cu_prev=temp;	
			if (!pb_steady)	
		40	temp=(double)(blocked_pk1+blocked_pk2)/(double)(total_pk1+total_pk2);	
			if (pb_prev)=0) ratio=temp/pb_prev: else if (temp=0) ratio=1; else ratio=0; if (ratio 3000 & & ratio -1 (01) - ch stradie=1; else ratio=0;	
		45	if (14h) thratic voice and	
			temp=(double)blocked_pk1/(double)total_pk1:	
			if (pbl_previ=0) ratio=temp/pbl_prev;	_

MM2BL+1 Uniform Traffic System Enrection Block record stato) record stato record stato </th <th>Proces</th> <th>s Model Report: mm2bi1_scheduler Fri Jan 7 05:31:28 1994 Page 7 of 7</th> <th></th>	Proces	s Model Report: mm2bi1_scheduler Fri Jan 7 05:31:28 1994 Page 7 of 7	
Function Block iii record stats() iiii record stats() iiii record stats() iiii iiii iiii record stats() iiiii iiii iiiii record stats() iiiii iiii iiiii iiii iiiii (or (ac), irT1: irt): iiii (slot(1)))(srift): iirt): (or (ac), irt1: irt): iiii (slot(1)))(srift): iirt): (or (ac), irt1: irt): iiii (slot(1))(srift): iirt): (or (ac), irt1: irt): iiii (slot(1))(srift): (or (ac), irt1: irt): iiii (slot(1))(srift): (or (ac), irt): iiit (slot(1))(srift): op stat_wrife. scalar(* or : (dot, lool)*bhocked_ps2)(double)(hold_ps2)(double)(hold_ps2) op stat_wrife. scalar(* pr2 : (double)bhocked_ps2)(double)(hold_p)(srift): op stat_wrife. scalar(* pr2 : (double)bhocked_ps2)(double)(hold_1): op stat_wrife. scalar(* pr2 : (double)bhocked_ps2)(double)(hold_1): <th>M/M/2/ schedu</th> <th>B/L+1 Uniform Traffic System lier Model</th> <th></th>	M/M/2/ schedu	B/L+1 Uniform Traffic System lier Model	
Function Block record.stat() ime: lime: (re(ide), if revertength_mo: j++) ime(ide) (juttice): (re(ide), if revertength_mo: j++) ime(ide) (juttice): (re(ide), if revertength_mo: j++) ime(juttice): (re(ide), if revertend) ime: stat(rev: (stolend)) (re(ide), if revertend) ime: stat(rev: (stolend)) (re(re), if revertend) ime: stat(revertend) (re(re), if revertend) ime: stat(revertend)			1
record.state() in: [i]; c:erren danote, soil [30](10]; in: [4]; for (a-0; i=T1; i+1) for (a-0; i=T1; i+1) for (a-1);	5	nction Block 54 lines	
S time init in init init centro donobic inite 1.1mc2; centro channed. agen slot[130](10; for(i=0; i <t1; i+1)<="" td=""> init(i=0); i<t1; i+1)<="" td=""> for(i=0; i<t1; i+1)<="" td=""> init(i=0); i<t1; i+1)<="" td=""> for(i=1); i<t1; i+1)<="" td=""> io(i); i</t1;></t1;></t1;></t1;></t1;></t1;></t1;></t1;></t1;></t1;></t1;>		record_stats()	
10 (or (j=0; j <varvelength_not; j++)<="" td=""> 10 (or (j=0; j<varvelength_not; j++)<="" td=""> 10 (or (j=1); i<d(1+1); i<d(1+1);="" i<d(1+<="" th=""><th></th><th>it it; extern double itme!.time2: extern channel_agen sloq130[[10];</th><th></th></d(1+1);></varvelength_not;></varvelength_not;>		it it; extern double itme!.time2: extern channel_agen sloq130[[10];	
20 int (idu)(i)(i)(i)(i)(i)(i)(i)(i)(i)(i)(i)(i)(i)	10	<pre>for (i=0: i<1): i++) for (i=0: j<vvvelength_no: (i="T1:" for="" i++)="" i++)<="" i<(1+1):="" j++)="" j<vvvelength_no:="" pre=""></vvvelength_no:></pre>	
20 initial_resource/scatpacket.pp) Packet* packet: i, i, kap.high.low: 23 iii stap.high.low: 24 soc. 25 iii si.k.ap.high.low: 26 iii si.k.ap.high.low: 27 auoble 28 iii (j) 29 apr0. 30 apr0. 31 for (=10w: ichigh.i+) 50 if (j) 51 if (=10w: ichigh.i+) 52 if (=100;10].src=0.1k=) 53 if (=100;10].src=0.1k=) 54 if (=100;10].src=0.1k=) 56 if (=100;10].src=0.1k=) 57 if (=100;10].src=0.1k=) 58 if (=100;10].src=0.1k=) 59 op.Ph.fidd.src=1.sol(1]k.lds=dt: sol(1]k].dst=mop. shc_1]k].chee.check.townelsength.r.b) 50 op.Ph.fidd.src=0.r.sol(1]k.lds=dt: sol(1]k].stence.t.sol(1]k].ldst=mop. shc_1]k].check.townelsength.r.b) 50 op.Ph.fidd.src=0.r.sol(1]k.lds=dt: sol(1]k].stence.t.so	15	ii (stoli)[], sct=0; imr&=iep_sim_time(>stoli)[], clime, with [], sct=0; imr&=iep_sim_time(N*op_sim_time()); op_sim_write_scalar(* ex-side) on the firsto, but head ph2*shot, conv. it imr@(); op_sim_write_scalar(* ex-side) on the intervention of the intervention intervention of the	
mm sc.dt.pr Packet* packet. int i,i,k.ap.high.low. ouble svc: cutern int T.J.ao.in.svc.l. ap=0. if (p=1) { low=0. high=T1: } lok { low=T1: high=T+T2: } for (j=(w:ichight.i+) /* for each column / for (j=(w:ichight.i+)) /* for each column / if (j=1) { low=0. high=T1: } lok { low=0. high=T+12: } if (slot(j][]) src==0. hoi j+1) if (slot(j)[]] src==0. hoi j+1) if (slot(j)[]] src==0. hoi j+1) if (slot(j)[]] src==0. hoi j+1) if (slot(j)[]] src==0. hoi j+1)	20	int find_resource(sc.dtpacket.tp)	
 ²⁵ int i, i,k,ap,high,how. ³⁰ doubt xvc.: ³⁰ cutern int xvc.: ³¹ cutern channel_agen so(130(10): ³² cutern channel_agen so(130(10): ³² for each column / ³⁵ for (=low: ichigh: i++) / / or each column / ³⁵ k=1: ³⁶ for (=low: ichigh: i=+) ³⁶ (i=0; i=0; i=0; i=0; i=0; i=0; i=0; i=0;		int scolup: Packet: Packet:	
 ap=0. if (p=1) { (ow-0, high=T1:) dsc { (ow=T1: high=T1+T2:) for (i=low: ichigh: i++) / / or each column / i (i-obj-cwavelength_noi)++) if (i=l) / (i=l)	25	int ij,k.aphigh.how: double svc: extern int T1.no_in_svc1.ao_in_svc2: extern channel_agen stor[130][[10]:	
for (i=low:i:dight, i++) // for each column '/ 35 [c] (i 15 for (j=low;i:dight, noi)++) 16 [(slot][j]] sc==0, j++) 17 [(slot][j]] sc==0, k=]; 18 [(slot][j]] sc==0, k=]; 19 [(slot][j]] sc==0, k=]; 11 [(slot][j]] sc==0, k=]; 16 [(slot][j]] sc==0, k=]; 17 [(slot][j]] sc==0, k=]; 18 [(slot][j]] sc==0, k=]; 19 [(slot][j]] sc==0, k=]; 11 [(slot][j]] sc==0, k=]; 11 [(slot][j]] sc==0, k=]; 12 [(slot][j]] sc==0, k=]; 13 [(slot][j]] sc==0, k=]; 14 [(slot][j]] sc==0, k=]; 15 [(slot][j]] sc==0, k=]; 16 [(slot][j]] sc==0; j]] 17 [(slot][j]] sc==0; j]] 18 [(slot][j]] sc==1; slot][j]] 19 [(slot][j]] sc==1; slot][j]] 10 [(slot][j]] sc==1; slot][j]] 11 [(slot][j]] sc==1; slot][j]] 12 [(slot][slot][sc==1; slot][slot][sc==1; slot][sc==1; slot][slot][sc==1; slot][slot][sc==1; slot][slot][sc==1;	30	ap=0; if (p==1) { low=0; high=T1; } clss { low=T1: high=T1+T2; }	
35 k=1: in (j=0;/cwavelength_noj++) 1 [(=0;/cwavelength_noj++) 1 [((=0);/cwavelength_noj++) 1 [(((=0);/cwavelength_noj++) 1 [(((=0);/cwavelength_noj++) 40 [((((=0);/cwavelength_noj++) 1 [(((=0);/cwavelength_noj++) 1 [(((=0);/cwavelength_noj++) 1 [(((=0);/cwavelength_noj++) 1 [(((=0);/cwavelength_noj++) 1 [(((=0);/cwavelength_noj++) 1 [(((=0);/cwavelength_no)) 20 [(((=0);/cwavelength_no)) 21 [(((=0);/cwavelength_no)) 22 [(((=0);/cwavelength_no)) 1 [(((=0);/cwavelength_no)) 29 [((((=0);/cwavelength_no)) 20 [((((=0);/cwavelength_no)) 21 [((((=0);/cwavelength_no)) 22 [((((=0);/cwavelength_no)) 23 [((((=0);/cwavelength_no)) 24 [((((=0);/cwavelength_no)) 25 [((((=0);/cwavelength_no)) 26 [(((=0);/cwavelength_no)) 27 [((=0);/cwavele		for (i=low: :chigh: i++) /* for each column */	
 if (slot[ij]) arc==0) k=j: if (slot[ij]) arc==s l slot(ij]) dst==dt) goto next_col: if (k=-1) [ap=1, goto assign:] next_col: ap=0, next_col: ap=0, if (ap) return 0. if (ap=1, not d set[packet, "awai serget: 43; op 1 k, nof a set[packet, t = ineal assign: assign: if (p=1) no.in_svet i=i; cle no.in_sve2 ==i; op_1 k, and delayed[packet] 	35	k=1: ior(je0ij≤wavelength_norj++)	
 45 if (1,2p) reum 0; 45 if (1,2p) reum 0; 46 class is a segimation of the sequence	40	if (sloqii)[],src=0) k=j: if (sloqii)[j,src=sc # sloqii][j]dst==d0 gato next_col: if (k:=1) { ap=1; goto assign: } next_col: ap=0;	
assign: soliditiki krasses: sholijiki dasadi: sholijiki krime-op. sim. time() soliditiki krasses: sholijiki dasadi: sholijiki krime-op. sim. time() soliditiki krasses: sholijiki dasadi: "solo p. k. nid. set(parket: t: inesi if (ty==1) no. in. svet +=1; eise no. in. sve2 +=1; op. pk. nid. set(parket: t: inesi runn 1: runn 1:	45	ii ('lap) return 0. else	
	20	asign: stol(i)[k] scr=sc: stol(i)[k].ds=dt: stol(i)[k].ctime=op_sim_time(): op_kb_nfdset(packet: *avva!angth: 4.b); op_pk_andf_set(packet: t=imesior: 4.); svc=op_dist_outconne(svc_dist); (t[q==1)] no_in_svc[4=1; else no_in_svc24=1; op_pk_send_delayed(packet().svc); return 1;	

if (type==1 && no.in_svc1==N1) thocked pk1+=1: op.pk.destroy(pkprt);) else if (type==0 && no.in_svc2==N2) thocked_pk2=1: op.pk_destroy(pkprt);) else if (find_resource(src.dst,pkprt,typc)) if (find_resource(src.dst,pkprt,typc)) if (type==1) blocked_pk1+=1; clse blocked_pk2+=1; if (type==1) blocked_pk2+=1; clse blocked_pk2+=1; if (type==1) blocked_pk2+=1; clse blocked_pk2+=1; clse blocked_pk2+=1; if (type==1) blocked_pk2+=1; clse blocked_pk2+=1; clse blocked_pk2+=1; if (type==1) blocked_pk2+=1; clse blocked_pk2+=1; clse blocked_pk2+=1; if (type==1) blocked_pk2+=1; clse blocked_pk2+=1; clse blocked_pk2+=1; if (type==1) blocked_pk2+=1; clse blocked_pk2+=1; clse blocked_pk2+=1; if (type==1) blocked_pk2+=1; clse b	cate 3:	schedule (Enter Execs)	forced, 11 lines
<pre>(blocked pk1+=1: op_pk_destroy(pkprt);) eke if (type==0 && no_in_svc2==N2) blocked_pk2+=1: op_pk_destroy(pkprt);) eke if (find_resource(srcdstpkprt,type)) op_pk_destroy(pkprt); if (type==1) blocked_pk1+=1: clsc blocked_pk2+=1: if (type==1) blocked_pk1+=1: clsc blocked_pk2+=1:)</pre>	if (ty	c==1 && no_in_svc1==N1)	
else if (type==0 && no.inswc2==N2) if (type==0 && no.inswc2==N2) else if (find_resource(srdstt,p(prt.type)) op_pk_destroy(phpt): if (type==1) blocked_pk(1+=1: cise blocked_pk2+=1: if (type==1) blocked_pk(1+=1: cise blocked_pk2+=1:		<pre>blocked pk1+=1: op_pk_destroy(pkptr):)</pre>	
<pre>if (type=0 && no. in_wc2==N2) f (type=0 && no. in_wc2==N2) f (hocked_pk2+=i; op_ph_destroy(pkpr);</pre>	else		
5 [blocked_pk2+=1: op_pk_destroy(pkpr);) else if (!find_resource(src.dst,pkpri.typc)) op_pk_destroy(pkpr); if (type=1) blocked_pk(1+=1: clsc blocked_pk2+=1: j ('type=1) blocked_pk(1+=1: clsc blocked_pk2+=1: j ('type=1) blocked_pk1+=1: clsc blocked_pk2+=1:		f (type==0 & & no_in_svc2==N2)	
<pre>clsc if (!find_resource(src.ds.tpkprt.type)) op_pk_detroy(pkprt): if (!yppe=1) blocked_pk[1=1: clsc blocked_pk2+=1:</pre>		<pre>[blocked_pk2+=1: op_pk_destroy(pkptr);]</pre>	
if (!find_resource(src.dst.phpt.t.ype)) {		tise	
0 0 _ pk_destroy(hçur): 10 _ if (yppe=1) blocked_pk[+=1; clsc blocked_pk2+=1; 1 _ if (yppe=1) blocked_pk2+=1; clsc		if (!find_resource(src.dst,pkptr.type))	
op_pk_destroy(pkpt): 10 if (type==1) blocked_pk1+=1: clsc blocked_pk2+=1: 1			
10 if (type==1) blocked_pk1+=1: else blocked_pk2+=1: }		op_pk_destroy(pkptr);	
		<pre>if (type==1) blocked_pk1+=1: else blocked_pk2+=1;</pre>	

EXe Cor



OPNET report for the *scheduler* processor model. The *release* model is identical to that in M/M/2/B/L+1.



<u>M/M/2/B/CL:</u>

L Uniform Traffic System Model	MM/2/B/CL Uniform Traffic Syst scheduler Model	wa
0. [alt [] attached [] and [] attached []		
U: Init (Entuer Exects) torced, 34 lines	State Variables	41 lines
assume $LI <= L2$ Y	Objid Nown_id: Objid Narc_id:	
Jina sim attr get(OPC_IMA_INTEGER, N, & coode_no);	int vnode_no	
Juna sin, aur Settore. ImA_Intelect. "A"& vavecagh_no):	int Wavelen	gth_no;
Jume sim attr get(UPC_IMA_IN)EGEK."T. &tm_size); 1 ime sim attr ant/OPC_TMA_INTEGED :::. & i.e	5 int Vin_size	
Jima sim attr est(DPC IMA INTEGRE rts. & Cancella).		
Lina sim attr ret(OPC_IMA_DOUBLE.a. (ka);	int solution	
	10 int W.	-
vn_id=op_id_self(); src_id=op_tapo_in_assoc(own_id.0);	int WI:	
lima_obj_attr_get(src_id.'interarrival args'.&arv_arg(0));	int W2;	
	int is:	
v_rate=atot(&arv_arg(0)); arv_rate=1/arv_rate;	double la:	
	15 double svc_tim	
-watering	char arg	20];
	double tary rate	
	Distribution *Asr dis	
$((slot_nol-slot_no2)^* s >=0)$ svc time=(double)fin size/(double)slot_no2:	20 Distribution Asc. di	τ. τ.
se svc_time=(double)fim_size/(double)shot_no1;	Distribution *Cls dis	
	long int Vblocked	Det:
c_dist =op_dist_load(*exponential*svc_time.0):	long int blocked	pt2:
dist=op_dist_load(*uniform_int*1.(double)node_no):	long int votal pk	
t_dist=op_dist_homd('uniform_int'.1.(double)node_no);	25 kong int kotal_pk	2:
dist=op_dist_load(*bernoulli*a_0);	double ku_prev	
ad_prev=(a*8100_n01+(1-a)*8101_n02)*arv_rate*8ve_tunev(double)N;	double vpb_prev	
rcked nkl=0: hlocked nk2=0: total nk1=0: total nk2=0: released nk1=0: سامت ملك2=0: بنسماح0: بنسمك2-0:	double voltage	
_in_svc1=0; no_in_svc2=0; cu_steady=0; pb_steady=0; pb1_steady=0; pb2_steady=0; load_steady=0; ready=0;	30 double Noad mr	
	int ok cour	
r (i=0:i<(T1+T2);i++)	int by cour	12:
for (j=0;j <wavelength_no;j++)< td=""><td>int Vr_count</td><td></td></wavelength_no;j++)<>	int Vr_count	
{ slot[i][i].src=0; slot[i][i].dst=0; slot[i][rime=0; }	int Vr_count	2;
	35 int kcu_stcad	ly:
	int vob_stead	by.
	int vpb1_ster	tdy;
	int vob2_ster	udy;
DIG: (ARRIVAL)	int Voad ste	ady;
	40 Int Vready:	
are. M. Lievane Xd: (default)		
190C:		
ans: idle		
	Temporary Variables	3 lines
	Packet* pkptr.	
: Dix prepare (Enter Execs) forced 8 lines	int i.j.src.dst.type: double temo ratio:	
	UNUDIC ICHIPURHO	

Proce	s Model Report: mm2bil scheduler Eri, Jan 7 05:54-13 1994 Page 5 of 7	Process Mor	al Banot: mm2hi schadular Edi lan 7 05-54-13 1004 Banot 447
C/W/W		AAAA/D/B/CI	ici report: minizur_schedurer r11.dai: / 03.34.13.1334 raye 4.01./
sched	ler Model	scheduler M	
	(iboz_steauy)		
	temp=(double)blocked_pk2/(double)total_pk2:		
55	if (pb2_prev!=0) ratio=temp(pb2_prev: else if (/enn==-1) ratio=) - else ratio=)	Trai	is schedule
	tas utturpe=0; nator=1; tas nator=1; tele ph2_prev=1; else ph2_prev=1emp; if (ratio > 0.999 && ratio < 1.001) ph2_steady=1; else ph2_prev=1emp;		
	if (cu_steady && ph_steady && pb1_steady && ph2_steady) ready=1:	State 2:	idle (Enter Execs) unforced, 80 lines
96	if (Jready)	If (1)	oad strady && released_pk1>0 && released_pk2>0)
	ratio=0; temp=0; for (i=0+2/T) : i=4.)		temp=(total_pk1*stot_no1+total_pk2*stot_no2)*svc_time2(N*op_sim_time()): ratio=tempfload_prev:
	for (j=0; j=wavelengh_no; j++)	5	if (ratio>0.999 && ratio<1.001)
65	if (slot[i][j].srci=()) temp+=(op_sim_time()+slot[i][j].ctume): for (i=T1; i<(T1+T2): i++)		load_steady=1; tatio=0; temp=0;
	for (j=0, j-cwarelength_nov, j++) st/statististic-statistic statistic statistic statistic statistic statistics		ior (1=0. jc+avelength_no: j++)
	u tstoptightster=0.prattor=0.prattor=0.prattor=0.prattor=0.protect; ratio=(slot_no1*(time1+temp)+slot_no2*(time2+ratio))/(N*op_stm_time());	10	if (slot(j)[[]).sct:=0) temp+=(op_sim_time()-slot(j)[]).ctime): for (i=T1: i_TT); i_1_1_1
70	temp=tatio*N1; pk_countl=2*floor(temp); tr_countl+=1;	2	for (j=0; j=wavelength_no; j++)
			if (stodi)ij(j]src!=(0) ratio+=[op_sim_fine()-stodi]j[j].ctime); cu_prev=(stot_no1*(time1+temp)+stot_no2*(time2+ratio))(N*op_sim_time());
		:	ph_prev=(double)(blocked_pk1+blocked_pk2)/(double)(total_pk1+total_pk2);
75	if (ready && (total_pk1+total_pk2)>20000)	15	pb1_prev=(double)blocked_pk1/(double)tota_pk1; pb2_prev=(double)blocked_pk2/(double)pt2; temp=cu_prev*N1: pk_count1=floor(temp): temp=(double)released_pk1/(double)pk_count1: rr_count1=floor(temp)+2);
	tammericanal nel #olor nel ±tered nel 3%olor nel 3%one time/10% nel immerities nel immerities and neuvo		temp=cu_prev*N2; pk_count2=floor(temp); temp=(double)released_pk2/(double)pk_count2: tr_count2=floor(temp)+2);
	usupetuvat. par suo no revat. paz suo 1.002 & 2.500 (1027) w. unaetuvo 1.00 op. sun . unaetu): rauo e comprosa . prev if (ratio > 0.995 & & ratio < 1.005 & & PDDSIM)		
80	<pre>op_sim_end('reaching steady state');</pre>	20	
J			aa steady aan iteady aan iteasta pki shu counti "pk_counti) aan releasted pk 25 (tr_counti2 pk_counti2))
ē	1		nt (.cu_steady)
ηζ	116 2: 1016 (CE1 S)	25	ratio=0: temp=0;
¥ ز	CORD: (AKKIVAL) EXEC: :		for (i=0; i <t1: i++)<="" td=""></t1:>
	Trans: pk prepare		ig (slot[i][j])src=d) temp+=(op_sim_time()-slot[i][i];ctime);
0 ¥	T Cond: (default)	:	for $(i=T1: i<(T1+T2): i++)$
•	Trans: idle	30	for (j=0; j <wavelength_no; j++)<br="">if (storfilli) srei=1(nn sim time()-storfilli) reime):</wavelength_no;>
ΰ	Cond: (ENDSIM)		temp=(slot_no1*(time1+temp)+slot_no2*(time2+ratio))/(N*op_sim_time()); ratio=temp/cu_prev;
*	Trans: idle		if (ratio>0)999 && ratio<1.001 && temp<(total_pk1*slot_no1+total_pk2*slot_no2)*svc_time(N*op_sim_time())) en stradv=1*
		35	clse en mon-monte
ίΩ.	ite 3: schedule (Enter Execs) forced 11 lines		vdrov
1] if (type==1 && no_in_sec1==N1)		II (: DD_SKEAU)
	{blocked_pk1+=1:op_pk_destroy(akpt);}	40	temp=(duble)(blocked_pk1+blocked_pk2)((double)(total_pk1+total_pk2).
	if (type=0 && no_in_swc2=NZ)		ii (po_proved) raucedup/po_prove clsc if (temp==0) rauce1; clsc rauce0;
<u>~</u>	{blocked_pk2+=1; op_pk_destroy(pkpu);} else		if (ratio > 0.999 && aratio < 1.001) pb_steady=1: else pb_prev≈temp:
	if ('find_ressurce(src.dst.ptptr.type))	45	if (!pb]_strady)
	op_pk_destroy(kpkpr);		ternp=(double)blocked_pk1/(double)total_pk1:
<u> </u>	II ($h_{\text{pre=1}}$) bioxxou_pk($h=1$, ene bioxxou_pk_{n}=1;		if (pb1_previ=0) ratoo=emp/pb1_prev; else if (leenp=0) ratio=1; else ratio=0;
		20	if (ratio > 0.999 & & ratio < 1.001) pb1_steady=1; else pb1_prev=temp: }
		-	-

Proce	ss Model Report: mm2bi scheduler Fri Jan 7 05:54:14 1994 Page 7 of 7	Process Model Report: mm2bi scheduler Eri, Jan 7 05:54:14 1994 Page 6 of 7	1
WW.	B/CL Uniform Traffic System	MM/2/B/CL Uniform Traffic System	
8 CIE		scheduler Model	
-	if (stot(i)[j].dst==dt) cd=j;	State 3: schedule (CET's)	
		CET Cond: (1)	
÷.	_	#U Exec: ; Trans: idle	
	if (i==:(hgh+1) && the=-1: hd=-1: } clsc	!	
·	{ nf=0: ns= 1: nd≈ 1:	Function Block 103 lines	
3	for (j=0;j <wavelength_no;j++)< td=""><td>record_stats()</td><td></td></wavelength_no;j++)<>	record_stats()	
	if (slot(++1)[),src==0; [adir]=j; af+=1; } if (slot(++1)[j],src==5; hs=j; if (slot(++1)[j],dsr==d); ad=j;	t int extern double time2: 5 extern harmed asen stall 101:101:	
<u>نو</u>		for (1=0; it.11; i++)	
	if (cf==0 cd==1) goon next_coi:	for (j=ft) ; <pre>cwarvelength_no: j++) if (sloud)ij[]:sct=d)) ime1+=(op.sim_tme()-sloud)ij[]; ctime):</pre>	
×	for (j=b);cet(j++)	10 for (=T1: i<(T1+T2): i++) for (j=0; j <wavelength_no: j++)<="" td=""><td></td></wavelength_no:>	
	fi⊨0; if (pst=-1)	if (sloti)[j]:xc'=0) imc2+=(op.sim.thme()-slot(j][j].cimc): op_stat_write_statar(1-tota-(sloti) j] + slot(on-tota) p22*sto(1-202)*vc_imac(N*op.sim_time())): os stat write-statar(-vc-ist) no 1-imn()+slot(n no2"imn(2)Write os itm time())):	
75	if (qu)]==ps) it+=1: else continue: if (pu[=-1] if (calif==n) fin+=1: else continue:	15 0p_stal_write_scalar(*pb'.(double/blocked_pk1+blocked_pk2)(double)(total_pk1+total_pk2)); 0p_stal_write_scalar(*pb1*(double/blocked_pk1)(double)total_pk1);	
	if (al:	op_stal_write_scatar("pb2",(double/blocked_pk2/(double)total_pk2);	
	if $(nd)=1$	20	
×	If $\mathbf{f}([\mathbf{f}(\mathbf{f}(\mathbf{f}))] = \mathbf{f}(\mathbf{f}(\mathbf{f})) = \mathbf{f}(\mathbf{f}(\mathbf{f}))$ if $(\mathbf{f}(\mathbf{f}(\mathbf{f})) = \mathbf{f}(\mathbf{f}) = \mathbf{f}(\mathbf{f})$; $\mathbf{g}(\mathbf{f}(\mathbf{f}) = \mathbf{f}(\mathbf{f}))$	int find_resource(s: dt.packet.tp) ints.c.dt.packet.tp	
		Packet* packet:	
85	next_col:	25 int i,jap.fit.ps.pd.cf.cs.cd.nf.as.nd.ca[8].na[8].na[2].high.low: double vc:	
	if ([:=(high-1)) {	externing to a set of the set of	
	ps=rs: pd=cd: cs=nd: cl=nd: for (j=0;2cnf;j++) ca[j]=na[j]:	twent thattet_ment	
रु		if (tp==1) (Iow=0; high=T1; ps=-1; pd=-1:)	
	if (ap=-1)	clse	
56	clsc 	35 Iow=T1: high=T1+T2: ps=1: pd=-1: for (j=d5)=cwarckength_no.j++)	
10	assign: stol(ast0)[[ast1]].acr=sc: stol(ast0]][ast1]].dsr=dt: stor(ast0)[[ast1]].crime=up_sim_time(); up_that(a_st(upsct): "asve1=norpth: ast1]]; up_ph_ntd1_set(packet: "t:in=siot: "ast0]); svc=up_dist_outcome(svq_dist if (pa=1) no im_svc1=t1; esc no im_svv2+=1; op_thst ast0_delaret(asve1:ent) :	if (stot(tow-1](j) stc==sc) ps=j: if (stot(tow-1](j) dst==dt) pd=j: }	
		for (i=low: i <drigh: *="" <="" column="" each="" for="" i++)="" td=""><td></td></drigh:>	
		45 if (i==low)	
		c1=0; cs=-1; cd=-1; for (j=0;j=wavelength_po;j++)	
		50 I (slotijij)src==50) { ca[cf]=j; cf+=1; } if (slotijij]src==50) { ca[cf]=j; cf+=1; }	
			_



dist:

3

double double double double Distribution Distribution Distribution Distribution long int long int long int long int long int long int double double double int int long int lo

2

<u>M/M/2/B/RL:</u>

Process Model Report: mm2br_scheduler MM/2B/RL Uniform Traffic System scheduler Model

State Variables

kown_id: krc_id: hrode_no; \wavelength_no; tfm_size: klot_no1: klot_no2:

Objid in the set of th

2

OPNET reports for the *scheduler* and *release* processor models.

ន

d_prev; _count1 _count2

R

ph2_steady: load_steady: \ready:

<u>.</u> 1. 2. 1. 1. 1. 1. 1. 1. 1. 1.

33

Temporary Variables Packet* pkpur, int i,j.src.dst.type: double temp.ratio;

roces	s Model Report: mm2br_scheduler Fri Jan 7 06:52:19 1994 Page 4 of 10	Process	Model Report: mm2
VM/2/	S/RL Unitorm Traffic System	M/M/2/B/	/RL Uniform Traffic ?
chedu	ler Model	schedule	er Model
Ċ		ľ	
2	tte 1: pk prepare (Enter Execs) forced, 8 lines	Stat	te 0: init
v	rece(int) up_dist_outcome (src_dist): text_dist_f(int)up_dist_outcome(ds_dist): if (dst == src) goto next.		op_ima_sim_attr_get(op_ima_sim_attr_get(op_ima_sim_attr_get(op_ima_sim_attr_get(
n	type=(int)op_dist_ontcome(cis_dist); if (typt==1) {slot_noeslot_no1; total_pk1+=1;} cise {slot_no2; total_pk2+=1;;} biktre=op_kk_ee(con_intrat_non);;	<u>n</u>	op_ima_sum_attr_get(op_ima_sim_attr_get(op_ima_sim_attr_get(
]		10	own_id=op_id_self(); { op_ima_obj_attr_get(
S S	te 1: pk prepare (CET's)		arv_ratc=atof(&arv_arg
¥	Exec: : Trans: schedule	15	N=wavelength_no*fm_ svc_time=(double)N/(d
			if (s==0) s=-1; if ((slot_no1-slot_no2)
20	II (Iload steady && released pk1>0) && released pk1>0)	20	else svc_time=(double
~ ~	temp=(total_pk1*stot_no1+total_pk2*stot_no2)*svc_time(N*op_sim_time()); ratio=tempNtoad_prev: if (ratio=0.999 && ratio=(1.001)		svc_dist= op_dist_load src_dist= op_dist_load dst_dist= op_dist_load cls_dist= op_dist_load
	load steady=: ratio=0; temp=0; for (i=1: i<=N2: i++)	25	load_prev=(a*slot_nol
	if (pkt[i] src!=0) if (pkt[i] type==1) temp+=(op_ sim_time()-pkt[i] ctime):		blocked_pk1=0; blocke time1=0; time2=0; cu_
0	clse if (pktli)type==2) ratio+=(op_sim_time()-pktli)tctime): else print((*error finding pkt type\n*);	30	for (i=0:i<=N1 # i<=N2
	cu_prec=[slo_ton+t(mic+temp)+slo_too2*(tim2+ratio)/N*vg_sim_time(); ph_prev=[double/blocked_pk1/blocked_pk2/globe/blocked_pt+trau_gr2f_pt+trau_gr2f ph_prev=[double/blocked_pk1/blockbloral_gr4f, ph_p_prev=[double/blocked_pk2/globe/blocked_pk2/globe/blocked_pk2		{ pkt[i].src=0; pkt[i for (j=0; j<5; j++)
15	temp=time(+temp)/op_sim_time(): pk_counti=floor(temp); temp=tionabicire(essed_pk1/doubic)pk_counti: tr_counti=floor(temp)+20; temp=time(2-teatio)/op_sim_time(): bk_counti=floor(temp=time=time(texpset)=tk2)/doubic)pk_counti>tr_counti=floor(temp=4.0)	35	{ouncr(1JJJ.s } for (i=0:i <fm_size:i++ for (j=0;j<wavele< td=""></wavele<></fm_size:i++
20			slot[i][j]=0;
	ii (load_steady && !teady && teleased_pkt>(tr_count1*pk_count1) && teleased_pkt>/tr_count2*pk_count2))	Stat	te 0: init
ž	if (keu, steady)	CET #0	Cond: (AR) Exec.
	ratio=0: temp=0; for (i=1 : i <=N2 : i++) if (i pdf1 = xr(-1=)) if (pdf1 = xr(-1=))	#1 #1	Trans: pk. J Cond: (defa Exec: ; Trans: idle
30	<pre>creation is the strate of a strate is a strate is the strate is it is a strate is it</pre>]	
35	ה המשבר היה ההיה היה היה היה היה היה היה היה ה		
3	ter beveration:		
	ii (/bb_steady) I		

Provace Model Report: mm2hr schadular	Eri lan 7 06-52-10 1004	Paria 3 of 10
	1110011 1 00:00:10 1001	
WW/2/B/RL Uniform Traffic System		
scheduler Model		

00,00 00,00 01,00 01,00 01,00 01,00 01,00 00,00 01,00 01,00 01,00 01,00 01,00 01,00 01,00 01,00 01,00 01,00 00,000 00,000 00,000000	<pre>rm_sim_attr_get(OPC_IMA_INTEGER, rv. & mode_no); mm_sim_attr_get(OPC_IMA_INTEGER, rv. & knock_no); mm_sim_attr_get(OPC_IMA_INTEGER, rv. & kin_stor; mm_sim_attr_get(OPC_IMA_INTEGER, rv. & kin_stor; mm_sim_attr_get(OPC_IMA_INTEGER, rv. & kin_stor); mm_sim_attr_get(OPC_IMA_INTEGER, rv. & kin_n_no); id=op_id_set(f); src_id=op_idpo_In_assek(own_idi0); mm_sim_attr_get(SC_IMA_INTEGER, rv. & kin_n_no); attr_get(src_id': interarrival args' & kav_urg(0); atelof_dar_urg(0); arv_rate=l/arv_rate: ate=ato(i& arv_urg(0)); arv_rate=l/arv_rate: ate=dothe)N(dothe)slo(_no?; N2=floor(svc_ime); ime=(dothe)N(dothe)slo(_no?; N2=floor(svc_ime); ==0) s=1;</pre>
90-10 90-10 90-10 90-10 90-10 10 90-10 10 90-10 10 10 10 10 10 10 10 10 10	ma_sim_att_get(OPC_INA_INTEGER, '', & & & & & & & & & & & & & & & & &
5 00_10 00_10 00_10 00_10 00_10 10 00_10 247_7	<pre>ma_sim_attr_get(OPC_IMA_INTEGER, 1:1: &slot_nol); ma_sim_attr_get(OPC_IMA_INTEGER, 1:2: &slot_nol); ma_sim_attr_get(OPC_IMA_INTEGER, 1:2: &slot_nol); an_sim_attr_get(OPC_IMA_INTEGER, 1:2: &slot_nol); id=op_idi_set(0); src_id=op_lop_lin_asswc(own_id10); ma_obj_attr_get(src_idi': interartival args': &swr_arg(0)); atc=ato(&sar_arg(0)); srv_interartival args': &swr_arg(0)); arteelof(&arr_arg(0)); srv_interartival args': &swr_arg(0)); arteelof(&arr_arg(0)); srv_interartival; args': &swr_arg(0); arteelof(&sar_arg(0)); srv_interartival; args': &swr_arg(0); arteelof(belb()V(double)slot_nol; N2=floor(svc_ime); mic=(double)VV(double)slot_nol; N2=floor(svc_ime);</pre>
5 op_in inop_in 	ma_sim_attr_get(OPC_IMA_INTEGER: ::1: *&iol. no.2); ma_sim_attr_get(OPC_IMA_INTEGER: ::1: *&iol. no.2); ma_sim_attr_get(OPC_IMA_DOUBLE: ::1: *&iol. id=op_idset(1); src_id=op_topo_in_assoc(own_id:0); ma_obj_attr_get(src_id: :tnc=arr1va].args :*&av_arg(0); ma_obj_attr_get(src_id: :tnc=arr1va].args :*&av_arg(0); arte=ato(i&arv_arg(0); srv_inte=i/fav_r1stc. arte=ato(i&arv_arg(0); srv_inte=i/fav_r1stc. arte=ato(i&arv_arg(0); srv_inte=i/fav_r1stc. inte=(double)N(double)slo(_no?; N2=floor(svc_inte);
ni_qo ni_qo ni_qo ni_qo nrun	<pre>ma_sm_attr_get(OFC_IMA_DOUBLE.*s'.ka); ma_sim_attr_get(OFC_IMA_DOUBLE.*s'.ka); id=op_idget(xr_id:=pp_top_in_asse(svm_id(0); ma_obj_attr_get(xr_id:=incerarriva) args'.kav_arg(0); atc=ato(kav_ag(0)); av_rate=l/av_rate; arcegth_no*fm_aixc:svc_ime=double)N(double)slot_no1; N1=floor(svc_ime); ime=(double)N(double)slot_no2; N2=floor(svc_ime); ime=(double)N(double)slot_no2; N2=floor(svc_ime);</pre>
own op_in	id=0 -jdi_zeff (); src_id=0 -p _t0 po_in_assw(cwn_id()); ma_obj_attr_get(src_idi * incerarriva) args*.&av_arg(0)); atc=atot(&av_arg(0)); av_rate=1/ar_rate: avetegth_no*fm_size; svc_ime=(double)N(double)slot_no1; N1=floor(svc_ime); are=(double)N(double)slot_no2; N2=floor(svc_ime); ==0) s=-1;
10 op_in arv_r	u=op, at _set(1; sc., u=op_uop_un_assetown_u(u); ma_obj_attr_get(sc., jd. * incerarriva.) arges '&av_arg(0); atc=atot(&av_arg(0); av_arg(0); av_arg(st.) avetegth_no*ftn_sisc; svime=(double)N(double)slot_no!; N1=floor(svc_ime); ime=(double)N(double)slot_no?: N2=floor(svc_ime);
u_vis	ma_uon_aur_getoruneeaarr.va. argacanargu); ate=atoi(&ararg(0); arate=1/arate: avelegth_no*fin_airs.svc_imne=(double)N(double)slot.no!: N1=floor(svc_ime); imne=(double)N(double)slot_no?: N2=floor(svc_ime);
87. 18	ate=ato(Rarv_rgt0)); arv_rate=1/arv_rate: avelegth_no*fm_aise: svc_imme=(double)N(double)slot_no!; N1=floor(svc_ime): imme=(double)N(double)slot_no?; N2=floor(svc_ime): =40 s=-1;
	avelength_no*fin_sits: svc_imre=(double)N(double)StoLno1: N1=floor(svc_imc): imre=(double)N(double)slo1_no2: N2=floor(svc_lime): =0) s=-1:
N	avecugu_nov nu_xave.xvv_unte=euototer/vv.euotote/sov_nov.vv=avecove_nov.vv=untey. time=(double)V(double)slot_no2: N2=floor(svc_time): =0) s=-1:
15 svc_ti	=====================================
	:T==3() 2=-1:
π(S) π	the mode and the second and the standard doubt a share of a standard before and.
else s	aor_aor_aor_aor_aor_a svuue=(oouere.)uu_arze(uouere.)saor_aoz. sve_time=(douhle)fm_size(/douhle)slor_not:
20	
svc_d	dist= op_dist_load(*exponential*.svc_time.0);
src_d	list= op_dist_load("uniform_int".(double)node_no);
dst_d	uisterpologist_load("unitorm_int",",(double)node_no):
25 load	ustant_unst_unsu(nernoutti .a.u); prev=(a*slot_noi+(1-a)*slot_no2)*arv_rate*svc_time/(double)N;
block	$ccd_pki=0$; blocked_pk2=0; total_pk1=0; total_pk2=0; released_pk1=0; released_pk2=0;
time1	l=0: time2=0: cu_steady=0: pb_steady=0: pb1_steady=0: pb2_steady=0: load_steady=0: ready=0:
30 for (i=	=dric=N1 i <=N2;i++)
	pkt[i],src=0; pkt[i],dst=0; pkt[i],ten=0; pkt[i],type=0; pkt[i],ctime=0;
_	
35	{other[1][].siot=0. other[1][].wavelengtn=0. other[1][].len=0.}
for G	=dtisffm sizeri++)
	for (j=0;j <wavelength_no;j++)< td=""></wavelength_no;j++)<>
	slot(i][i]=0:

Sta	te 0:	init (CET's)
Ē	Cond:	(ARRIVAL)
9	Exec.	
	Trans:	pk_prepare
Ā	Cond:	(default)
Ξ	Exec:	
	Tranc.	

roces	ss Model Report: mm2br_scheduler Fri Jan 7 06:52:20 1994 Page 6 of 10	Pro
/M/2/	BIRL Unitom Traffic System Juler Model	MN
ŀ		
ŝ	ate 3: schedule (Enter Execs) forced, 5 lines	_
	if (!find_resource(src.dst.pkptr))	
2	op_pk_destroy(pkpus) if (type==1) blocked_pk1+=1: clse blocked_pk2+=1:	
]		
ŝ	ate 3: schedule (CET's)	
빙 육	El Cond: (1) Exec: :	
	Trans: idle	
ŭ	and the state of t	
ĩ	Incluon block	
<u>s</u>	record_stats() { int i.cl=0.c2-0; extern double time! Lime2; extern channel_agn pt(1)025;	
	for (i=1: i<=N1 ii (<=N2; i++) u (pki(i)_acc ¹ =0) ii (pki(i)_acpc=1) { cl+=1: uncl+=(0p_sim_time()-pki(i)_cunc); }	
2		
15	<pre>if (pudi).ype==2) [c2+1: imc2+imc3+imc3, public, public, public, cume(:) clear public errors. In pact. represents. op_sul_write_scalar(*: load=(load=published=public).wr_imme()): op_sul_write_scalar(*: load=(load=published=public).op_sul_timme()): op_sul_write_scalar(*: pp: load(load).pb(k)cked_public)(load=public). op_sul_write_scalar(*: pp: 2* (double)blocked_pk2/(double)load=publ): op_sul_write_scalar(*: pp: 2* (double)blocked_pk2/(double)load=publ): op_sul_write_scalar(*: pb2: (double)blocked_pk2/(double)load=publ):</pre>	
50		
	int find_resource(se.dtpacket) int se.dt; Packet* packet	
25	{	
30	double vvc. extern int sdq(130)[10]; extern to her, agen other[102][15]; extern channel_agen pd(1025];	k,
	ct=d): for (j=dr, j<+warelength_no: j++) empry(j]=dr.	
35	for (=0icfm_size:i++)	
	conflict[i][0]=1; conflict[i][1]=1; for $(k=0; k$ -evavelength, no: $k \rightarrow 1$	
<u>}</u>	if (slot(i)[k]=0) cmpy[k]+=1: sr = a transition (seccs) configuration	

Process Mo	del Report: mm2br_scheduler	Fri Jan 7 06:52:20 1994	Page 5 of 10	_
M/M/2/B/RL	Uniform Traffic System			T
scheduler M	odel			
				٦.
. 40	temp=(double)(blocked_pk]+blocked_pk2)((double)(total_pk	+total nk2);		
	if (pb_prev!=0) ratio=temp/pb_prev.			

$\begin{array}{l} u_1(p_2,p_2,r=v), and extanging p_1(rev.)\\ clse if (rank)==0) ratio=1; clse ratio=0;\\ if (ratio>0.999 & & ratio<1.001) ph_s(rady=1; clse ph_prev=temp;\\ f \end{cases}$	if (!pbl_steady) {	temp=(double)blocked_pk1(double)total_pk1: if (oph_prev=0)0 nation=temp)_prov: else	if (¦pb2.steady)	temp=(double)blocked_pk2/(double)total_pk2: if (pp2_previ=0) ratio=temp/ph2_prev: clse if (temp==0) ratio=1; clse ratio=0; if (ratio > 0.999 && ratio < 1.001) ph2_stratio=1; clse ph2_prev=temp;	if (cu. steady && ph_steady && ph_steady && ph2_steady) : if (tready)	<pre>raio=0: temp=0; for (:=1: i=N1 1:==N2; i++) if (pk(1); arc=07, i++) if (pk(1); arg=1) temp==(0p sim_time(-pk(1); cime); cise if (pk(1); type==2) raio==(0p sim_time(-pk(1); cime); cise if (pk(1); type==2) raio==(0p sim_time(-pk(1); cime); cise if (pk(1); type==2) raio=(-pk(1); trano); cise if (pk(1); type==2) raio=(-pk(1); trano); temp=(time2+raio)(0p sim_time(0; pk_count2=2*floot(temp); tr_count2+=1; })</pre>	<pre>(reary && (total_pk1+total_pk2>,0,000) temp=(total_pk1*stot_no1+total_pk2*stot_no2)**ve.time(N*op_stim_time()); ratio=temp/load_prev: if (ratio > 0.995 && ratio < 1.005 && FENDSIN). op_stim_end(* reachting steachy state*******)</pre>
if (ratio	if (!pb1_steac	temp=(d if (pb1 else if (ratio	if (!pb2_stead	temp=(d if (pb2_1 else if (ratio	if (cu_steady if (!ready)		If (ready & & & (tota { temp=(total_] if (ratio > 0); op_sim_}

Stat	te 2:	idle (CET's)
CET	Cond:	(ARRIVAL)
â	Exec:	
	Trans:	pk_prepare
CET	Cond:	(default)
Ŧ	Exec:	
	Trans:	idle
E	Cond:	(ENDSIM)
ŧ	Exec:	record_stats();
	Trans:	idle



if (j>=nd)

if (next)

if (j<nd || i==(ct-1))

-

if (j<nd)

continue:

go_back=1; goto insert; back:

MM/2/B/RL Uniform Traffic System scheduler Model

Process	: Model Report: mm2br_scheduler	Fri Jan 7 06:52:21 1994	Page 9 of 10
M/M/2/E schedul	J/RL Uniform Traffic System er Model		
	if (rmb[oh]) { k=1; break; } if (k==1)		
	continue: else		
165	-		
	temp[i_ct]=i; k=store[i][0]-1: if (k<0) k=0; so hæck=storeli][0]+store[i][2]+1: if (so back>fm siz	t) vo hack≓im size:	
	for (oh=k; oh <go_back; oh++)<="" td=""><td></td><td></td></go_back;>		
170	if (j>nd) tmp_rmb[oh]=1: clse rmb[oh]=1:		
	ii (j <nd)="" continue;="" i_ct+="1;" nd="j;" td="" waste+="fit;" {="" }<=""><td></td><td></td></nd>		
175	11 ()==nd) {		
	waste+=fit: i_ct+=1: if (waste <aspn[0])<="" aspn[0]="-1" td="" =""><td></td><td></td></aspn[0]>		
180	agn(y)=water, agn(y)=1=1.ct. for (k=2; k<1(, ct+2); k++) agn[k]=tcmp[k-2]; if ($agn(0)=-0$) goto assign:		
	http://waste=0.		
185	<pre>for (oh=0: oh<fm_size: oh++)="" rmb[oh]="0:" unp_rmb[<br="" {="">continue:</fm_size:></pre>	h]=0; }	
	} if (isnd) { armi-nd+fit an [=fit next=]' }		
	if (next && i==(ct-1))		
190	t waste+≕ap1: i_ct+=1;		
	if (waste <asgn[0])<br="" asgn[0]="-1" ="">f</asgn[0]>		
	asgn[0]=waste; asgn[1]=i_ct; 671-0-1-42 = -10141(1-2)-		
195	$100 (\text{ k}=2; \text{ ket}_{12}(1+2.); \text{ as gut } \text{k}=1.000 \text{ ket}_{12}(1+2.);$ if (asgn[0]==0) goto assign:		
	~		
200	if(asgn(0]==-1) return 0:		
	else		
	assign: for (k=1 : k<=N1 k<=N2: k++)		
205			
	ut (pkt(k].src=sc; pkt(k].dst=dt; pkt(k].len=äsgal[1]; pkt[k].src=sc; pkt(k].dst=dt; pkt[k].len=äsgal[1]; if (stol_no==stol_no1) pkt[k].type=1;		
210	clse if (slor no=slot no2) pkt[k] type=2; else print[(error in assign pkt type\n*	
	pk([k].crime=op_sim_time(): nd=slot_no: for (i=2: i<=asgn[1]; i++)		
215	oh=store[asgn[i]][0]; ct=store[asgn[i]][2]; nd=ct; if (store[asent]]][3]=1) { / cr=1 · nd+=1 · }		
	clse if (store(asgn[i)][3]=2) { ct-1: nd+= clse if (store(asgn[i)][3]=2) { ct-1: nd+= clse if (store(asgn[i)][3]=3) { ct-1: nd+=	l; ob+=l; } =2; nd+=2; ob+=l; }	
	<pre>for (j=oh; j<(oh+ct); j++) slot(j][store[asgn[i]][1] other[k][i-2].slot=oh; other[k][i-2], wavelength=st</pre>	=k; sre[asgn[i]][1];	
	-		

Process A	Aodel Report: mm2br scheduler	Fri Jan 7 06:52:21 1994	Page 10 of 10
M/M/2/B/F scheduler	AL Uniform Traffic System Model		
060			
	i=asgn[1]+1: ob=store[asgn[i]][0]; rf (store[asgn[i]][3]==3) ob+=1;		
	<pre>clsc if (store[asgn[i]][3]==2) oh+=(store[asgn[i]] for (i=oh: i<(oh+nd); i++) slot[i][store[asgn[i]][1]</pre>	[2]-nd);]=k:	
225	other[k][i-2].slot=oh; other[k][i-2].wavelength=s op_pk_nfd_set(packet 'timeslot'.oh); op_pk	bre[asgn[i]][1]; other[k][i-2].len=nd; nfd_set(packet, *wave1ength*,store[as;	gn(i)][1]):
	svc=op_dist_outcome(svc_dist):	łayed(packet.0.svc): break:	
	return 1:		
230			



B.4.2 Queueing System

The sequential batch procedure described in Section B.3.2 is used as termination criteria for the M/M/2/Q/RL simulation. The observation was made on the average queue size for both types of sessions $(Q_1 \& Q_2)$ after the measured load comes to within 1% of the offered load obtained in Section 4.1. When the 90% confidence interval for $Q_1 \& Q_2$ comes to within 10% of the value of $Q_1 \& Q_2$ respectively, the simulation will end. The simulation will also be terminated after it has been running for 2,000,000 seconds.

The seed used for each simulation has value NWs, where N is the number of users, W the wavelength channels, and s equals to one if the offered load is normalized against sessions with smaller of the two throughput requirements and zero otherwise (refer to Section 4.1). So for N equals to forty, W equals to eight and s equals to one, the seed is 4081.

<u>*M/M/2/Q/RL:*</u> OPNET Reports for the *scheduler* and *release* processor models.

Multiple of the first of the	Mutuality of loging with and other	Proce	ss Model Heport: mm	zdr. scneauler	Fri Jan / U0:501:29 1994 Mage 2 01 13	Process Model Heport: mmzqr_schequier	
Site 34 (mis) 0 min weak 1 min weak	Site Vrizibie Marchen of the work, or in work of the work, in w	M/M/S sched	VQ/RL Uniform Traffic uler Model	System		WM/2/Q/HL Untorm Iramc System scheduler Model	
Option Number of network Number of network Hadder State 1 0.00 vvc. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Optim Wurder of musicing Musicing 1 10 <th>Ś</th> <th>tate Variables</th> <th></th> <th>34 lines</th> <th></th> <th>Sun</th>	Ś	tate Variables		34 lines		Sun
Obid Internet	Obtil		Objid Vow	vn_id:		Number of Header	Stat
1 International internatinaly internationaly international international internation	a workering a workering a workering a workering a workering a workering a workering a workering a workering b		Objid \sr	c_id:		States Block	Varial
5 m workenerganian 1 m Noncomparison 1 Noncomparison	1 International internatinternational international international international inte		Int To To To To To To To To To To To To To	de_no:		4 Yes	Yes
Image: Section Sectin Sectin Section Section Section Section Section Section Section	Image: Note of the second s	v		aveiciigui_no.			
International state Note of the sector state 1 0 doller v.v., inst. 0 doller v.v., inst. 0 doller v.v., inst. 0 doller v.v., dist. 0 doller doller of v.v., inst. 0 doller doller 0 doller (0 doller doller) 0 doller doller 0 doller v.v., onto, pol(2) 0 doller doller 0	Interface Note: Note: 1 Recent control: Recent control: 1 Recent control: Recent control: 1 Destruction: Vec. data: 1 De	<u> </u>					
Internet NC: 0 doubte w: 0. doubte w: stendus catilitabi	Int NL: 0 dolte w. 0 dolte w. 0 dolte w. 1 Derhenton			1: -			
10 min wr. 0 min wr. 0 wr. mr. 1 mr. wr. 1 mr. wr. <	10 dute dute wer inter- strations wer inter- wer inter- betwein wer inter- wer inter- wer dist 15 Destribution wer inter- betwein *** dist 16 betwein *** dist 17 Destribution *** dist 18 Destribution *** dist 19 Destribution *** dist 10 betwein *** dist 11 Destribution *** dist 12 betwein *** dist 13 betwein *** dist 14 betwein *** dist 15 inter *** dist 16 inter *** 17 inter *** 18 betwein *** 19 betwein *** 10 int *** 11 betwein *** 12 doubt *** 13 doubt *** 14 *** *** 15 inter *** 16 inter 17 inter 18 *** 19 *** 10 inter 11 *** 11 ****		int W.	2;		Header Block	
10 double to the construction of the	10 double two werk constrained between the between the betwe		int \sn			#include <edlib h=""></edlib>	
doubte New vergion: www.region: www.region: filtered examines doubte New vergion: www.region: www.region: filtered examines doubte New vergion: www.region: www.region: filtered examines doubte New ministered www.region: filtered examines filtered examines 15 New ministered filtered examines filtered examines 20 kreg int wuc. www.region: filtered examines 20 kreg int wuc. www.region: filtered examines 20 kreg int wuc. filtered examines filtered examines 21 double wuc. filtered examines 22 double wuc. filtered examines 23 double wuc. filtered examines 24 double wuc. filtered examines 25 double wuc. filtered examines 30 double wuc. filtered 31 double wuc. filtered	double behaviors wr. argin wr. argin 15 Databilition wr. argin Ver. argin wr. argin 16 Databilition wr. argin Ver. argin 17 Databilition wr. argin Ver. argin 18 Databilition wr. argin Ver. argin 19 Databilition wr. argin Ver. argin 10 Databilition wr. argin Ver. argin 11 Databilition wr. argin Ver. argin 12 Databilition wr. argin Ver. argin 13 Databilition wr. argin Ver. argin 14 Databilition Ver. argin 15 Databilition Ver. argin 16 Databilition Ver. argin 17 Databilition Ver. argin 18 Databilition Ver. argin 19 Databilition Ver. argin 10 Databilition Ver. argin 11 Databilition Ver. argin 12 Databilition Ver. argin 13 Databilition Ver. argin 14 Databilition Ver. argin 14 Ver. argin Ver. argin 14 Ver. argin Ver. argin 15 Databilition Ver. argin	10) double \a:			#include <edin< td=""><td></td></edin<>	
date wr.arg10) Deterholon vc.dis:	chrait www.arg/00; Deterhuise wwww.arg/00; Deterhuise wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww		double \sv	c_time;		#include $$	
Is between every distribution ever	Is between we determined Ver determined Is betweende Ver determined Determined Ver determined Det		char \ar	v_arg[20]:			
15 Destribution "Arc. dis: 16 Destribution "Arc. dis: 17 Destribution "Arc. dis: 18 Distribution "Arc. dis: 19 Inter Inter 10 Inter Inter 11 Inter Inter 12 Inter Inter 13 Inter Inter 14 Inter Inter 15 Inter Inter 16 Inter Inter 17 Inter Inter 18 Inter Inter 19 Inter Inter 10 Inter Inter 13 Inter Inter 14 Inter Inter 15 Inter Inter 16 Inter Inter 17 Inter Inter 18 Inter Inter 19 Inter Inter 10 Inter Inter 10 Inter Inter 10 Inter Inter 11 Inter Inter 12 Inter Inter 13 Inter Inter <	15 Destribution *ex. clas: 16 Destribution *ex. dist. Destrest. *ex. dist.		double \ar	v_rate;		5 tymedief struct	
[15] Distribution "Act dis: Distribution "Act dis: Distribution "Act dis: Distribution "Act dis: Distribution "Act dis: Distribution "Act dis: Distribution "Act dis: Distribution "Act dis: Distribution "Act dis: Distribution "Act dis: Distribution "Act dis: Distribution "Act dis: Distribution "Act dist Distribution "Act distribution" Distribution" Distribution" Distribution" Distribution" Distribution" Distribution" Distribution" Distribution" Distribution" Distribution" Distribution" Distribution" Distribution"	13 Distribution "vects dist 13 Distribution "vects dist 14 Distribution "vect dist 15 Distribution "vects dist 16 Init "vects dist 17 Distribution "vects dist 18 Net dist "init 19 Net dist "init 10 Nature "init 10 Nature "init 11 Nature "init 12 Outble "init 13 Outble "init 14 Nature "init 15 Outble "init 16 init Net 17 Init "init 18 Outble "init 19 Outble "init 10 Nature "init 13 Outble "init 14 Nature "init 15 Outble "init 16 Nature Nature 16		Distribution */s	src_dist:	-		
Distribution "vec, dist. Distribution "vec. <t< td=""><td>Distribution "vec, dist. Distribution "vec, dist. <td></td><td>5 Distribution */c</td><td>dst_dist;</td><td></td><td>int cross</td><td></td></td></t<>	Distribution "vec, dist. Distribution "vec, dist. <td></td> <td>5 Distribution */c</td> <td>dst_dist;</td> <td></td> <td>int cross</td> <td></td>		5 Distribution */c	dst_dist;		int cross	
Degrin voule "ds.dist. 20 long int voule 20 long int voule 21 long int voule 22 long int voule 23 long int voil 24 long int voil 25 long int voil 26 long int voil 27 long int voil 28 dooble variation 29 long int voil 20 long int voil 21 long int voil 22 dooble variation 23 dooble variation 24 dooble variation 25 dooble variation 26 dooble variation 27 dooble variation 28 dooble variation 29 dooble variation 20 dooble variation 20 dooble variation 21 dooble variation 22 dooble variation 23 dooble variation 24 dooble variation 25 dooble variation 26 dooble variation 27 dooble variation 28 dooble variation 29 dooble variation 20 dooble variation	Destribution "ds.ds: 20 long int vuct.sds: 20 long int vuct.sds: 20 long int vuct.sds: 20 long int vuct.sds: 21 long int vuct.sds: 22 long int vuct.sds: 23 lonble vuct.sds: 24 lonble vuct.sds: 25 lonble vuct.sds: 26 lonble vuct.sds: 27 lonble vuct.sds: 28 lonble vuct.sds: 29 lonble vuct.sds: 20 lonble vuct.sds: 29 lonble vuct.sds: 20 lonble vuct.sds: 21 lonble vuct.sds: 22 lonble vuct.sds: 23 lonble		Distribution *\s	svc_dist:			
Image in the received pl(2); brog in the vector pl(2); brog in the vector pl(2); brog in the vector pl(2); brog in the vector pl(2); brown the vector pl(2);	20 long int vectorset_pt(2); wetcresset_pt(2); long int 10 int ppc. 20 long int vectorset_pt(2); wetcresset_pt(2); long int 1 1 1 23 double wat int str. 24 double wat int str. 25 double wat int str. 26 double wat int str. 27 double wat int str. 28 double wat int str. 29 double wat int str. 20 double wat int str. 29 double wat int str. 20 double wat int str. 21 double wat int str. 20 double wat int str. 21 double wat int str. 22 double wat int		Distribution */c	cls_dist;		int ust.	
20 long int Viet. Cit. 20 long int Viet. Cit. 0 long int Viet. Cit. 0 long int Viet. Cit. 0 long int Viet. Cit. 1 long int Viet. Cit.	20 long int burg int vucc.t. 20 long int vucc.t. 0 orgin in vuc.t. 0 orgin in vuc.t. 1 orgin in vuc. 1 orgin in vuc.t. 1 orgin in vuc. 1 orgin in vuc. 1 orgin in vuc.t.		long int Vto	stal_pk[2];		10 Internet	
20 long int Wat Cit. Journel Lagn: 10 long int Vaci. Journel Lagn: 10 long int Vaci. Journel Lagn: 115 double Vaci. Interc. 126 double Vaci. Interc. 127 double Vaci. Interc. 128 double Vaci. Interc. 130 double Vaci. Journel. 131 double Vaci. Journel. 132 double Vaci. Journel. 133 double Vaci. Journel. 134 double Vaci. Journel. 135 double Vaci. Journel. 130 double Vaci. Journel. 131 double Vaci. Journel. 132 double Vaci. Journel. 133 double Vaci. Journel. 144 Journel. Journel. Journel. 145 double Vaci. Journel. 146 Math. Vaci. Journel. 141 Vaci. Journel. Journel. 141 Vaci. Journel.	20 long int Vact.ct.) interct_segn: 21 long int Vact.ct.) interct_segn: 15 interct_seqn: voc. 15 doubte Veta/12; 16 int voc.agn: 17 int voc.agn: 18 int voc.agn: 19 doubte Veta/12; 10 doubte Veta/12; 11 int voc.agn: 11 int voc.agn: 12 doubte Veta/12; 13 doubte Veta/12; 14 voc.agn: int 15 doubte vot.agn: 16 vot.agn: int 17 doubte vot.agn: 18 doubte vot.agn: 19 doubte vot.agn: 10 doubte vot.agn: 10 doubte vot.agn: 10 doubte vot.agn: 10 doubte vot.agn: 11 vot.agn: int 11 vot.agn: int 11 vot.agn: int 11 vot.agn: int 11 vot.agn: </td <td>_</td> <td>long int \pr</td> <td>rocessedpk[2]:</td> <td></td> <td>10 mm type.</td> <td></td>	_	long int \pr	rocessedpk[2]:		10 mm type.	
Jong int Unct. Sat. Jong int Unct. Sat. Jong int Uit. Jondie Undie verdienstruct Jondie Variationsci.	borg int 'uct. Sat: long int 'yred is nut: 'yred is nut:	Ä	0 long int \st	al_ct;		l channel acur:	
Imagini uti Not 2.3 doobie variation Variation 2.4 doobie variation Variation 2.5 doobie variation Variation 2.6 doobie variation Variation 2.7 doobie variation Variation 2.8 doobie variation Variation 2.9 doobie variation Variation 2.0 doobie variation Variation 3.0 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 doobie variation Variation 3.1 door Variat	Imagini uti Ingeni void Viet 25 double vuod.gree: double vuod.gree:		long int Vnc	ext_stat;			
15 In a protect and the start concernent of the	1/3 In solution to unit: double varaf.000012; double varaf.000012; double varaf.21; double vara		hong int \n(Ċ.		trimodal storet	
2s doubte Vand_merc; unu surd; doubte vand_merc; unu surd; doubte vand_merc; unu surd; unu surd; unu surd; int surd;	25 double Voad_prev: 26 double Voad_prev: 27 double voad_prev: 28 double voad_prev: 29 double voad_prev: 20 double varefergh: 20 double varefergh: 20 double varefergh: 20 double varefergh: 21 double vareforgh: 22 double vareforgh: 23 double vareforgh: 24 double vareforgh: 25 double vareforgh: 26 double vareforgh: 27 double vareforgh: 28 double vareforgh: 29 double vareforgh: 20 double vareforgh: 21 in vareforgh: 23 double vareforgh: 24 double vareforgh: 25 double vareforgh: 26 double vareforgh: 27 double vareforgh: 28 double vareforgh: 29 double vareforgh: 20		long int vn	1:			
2.5 double val(20000)[2; double val(med2); double val(med2); double val(med2); double val(med2); double val(2); double val(2); in val(103); double val(103)[15; in val(103)[16], double val(103)[16], in val(103)[16], double val(103)[16], double val(103)[16], in val(103)[16], double val(103)[16], in val(103)[16], double val(103)[16	12.6 double variations: 1.16 double variations: 1.10 double variations: 12.6 double variation: 1.10 double variations: 1.10 double variations: 12.6 double variation: 1.10 double variation: 1.10 double variation: 12.6 double variation: 1.10 double variation: 1.10 double variation: 13.0 double variation: 1.10 double variation: 1.10 double variation: 13.0 double variation: 1.10 double variation: 1.10 double variation: 13.0 double variation: 1.10 double variation: 1.10 double variation: 10.10 variation: 1.00 double variation: 1.00 double variation: 10.10 variation: 1.00 variation: 1.00 double variation: 10.10 variation: 1.00 variation: 1.00 variation: 10.10 variation: 1.00 variation: 1.00 variation: 10.10 variation: 1.00 variation: 1.00 variation: 10.10 variation: 1.00 variation: 1.00 variation: 10.10 variation: 1.00 variation: 1.00 variation: 10.10 variation: 1.00 variation:		double Vio	oad_prev;		int clot	
double Voal, fine(2); double Voal, fine(2); double Vasar(2); ini Voal, staad); ini Voal,	double Vala. double vest. double visard/21; int void.acendy; int void.acendy; <t< td=""><td>4</td><td>5 double \st</td><td>at[200000][2]:</td><td></td><td>III SIOL int moraleneth:</td><td></td></t<>	4	5 double \st	at[200000][2]:		III SIOL int moraleneth:	
double Vatar(2); double unit returned. 30 double Varar(2); varar(2); double unit returned. 31 double Varar(2); varar(2); unit unit returned. 30 double Varar(2); varar(2); unit unit returned. 31 double Varar(2); unit unit returned. 32 double Varar(1035); varar(2); unit unit returned. 31 varar(2); unit unit returned. unit returned. 32 double Varar(1035); varar(2); varar(2); unit returned. 31 unit varar(2); unit unit returned. 33 double varar(1035); varar(2); unit returned. 34 double varar(1035); varar(2); unit returned. 34 double varar(2); unit returned. unit returned. 34 double varar(2); varar(2); unit returned. 34 double varar(2); unit returned. unit returned. 34 double unit returned. 34 double <td< td=""><td>double Variat[2]; double Variat[2]; in Neal, stranty; in Variat[2]; variation: obtools in Variat[2]; in Variat[2]; <</td><td></td><td>double Vic</td><td>otal_time[2];</td><td></td><td>III WAVCUIEU.</td><td></td></td<>	double Variat[2]; double Variat[2]; double Variat[2]; double Variat[2]; double Variat[2]; double Variat[2]; double Variat[2]; double Variat[2]; double Variat[2]; double Variat[2]; double Variat[2]; double Variat[2]; double Variat[2]; in Neal, stranty; in Variat[2]; variation: obtools in Variat[2]; in Variat[2]; <		double Vic	otal_time[2];		III WAVCUIEU.	
adobie ystard2; adobie ystard2; adobie ystard2; adobie ystard2; adobie ystard2; adobie ystard2; adobie ystard2; adobie ystard2; adobie ystard2; adobie ystard2; adobie wisc.san adobie wisc.san in bade in bade </td <td>adobie bytard21; 20 chaneLage pdl1(025); 30 double vpcard21; chaneLage pdl1(025); 30 double vpcard21; chaneLage pdl1(025); adobie vpcard21; chaneLage rdl1(025); adobie vpcard21; chaneLage rdl1(025); adobie vpcard1; chaneLage rdl1(025); adobie vpcard1; chaneLage rdl1(025); adobie vpcard1; chaneLage rdl1(025); adobie vpcard1; rdl1(025); chaneLage ini verd sin; vpcard1; rdl1(025); adobie rdl1(02; rdl1(02; rdl1(02; ini verd sin; verd sin; verd sin; ini <</td> <td></td> <td>double Vd</td> <td>clay[2];</td> <td></td> <td>Int. Kelt.</td> <td></td>	adobie bytard21; 20 chaneLage pdl1(025); 30 double vpcard21; chaneLage pdl1(025); 30 double vpcard21; chaneLage pdl1(025); adobie vpcard21; chaneLage rdl1(025); adobie vpcard21; chaneLage rdl1(025); adobie vpcard1; chaneLage rdl1(025); adobie vpcard1; chaneLage rdl1(025); adobie vpcard1; chaneLage rdl1(025); adobie vpcard1; rdl1(025); chaneLage ini verd sin; vpcard1; rdl1(025); adobie rdl1(02; rdl1(02; rdl1(02; ini verd sin; verd sin; verd sin; ini <		double Vd	clay[2];		Int. Kelt.	
30 double visce[2]; double visce[2]; double visc. 31 double visc.[35][15]; double visc. 31 double visc. obje visc.[35][15]; double visc. 32 double visc. obje visc.[35][15]; double visc. 31 double visc. obje visc.[35][15]; double visc. 31 double visc. double visc. 32 double visc. double visc. 33 double visc. double visc. 34 double visc. double visc. <	abobic visca21; abobic visca21; abobic visca21; abobic visca21; abobic visca31; abobic visca31;		double Vts	start[2]:		ouner_asgn:	
30 double vuene(2); 30 double vuene(2); double vie: vuene(1); in: bobb in: void void void in: void void void void void void void in: void void voi	30 doubte 'unered': 30 doubte 'unered': doubte 'unered': 'unered': doubte 'unered': 'unered': doubte 'unered': 'unered': doubte 'unered': 'unered': unit 'void: 'unered': void: 'unered': 'unered': unit 'void: 'unered': void: 'unered': 'unered': unit 'void: 'unered': void: 'unered': 'unered': unit 'void: 'unered': unit 'unered': 'unered':		double \q	size[2];			
double vinc. unctunctunctunctunct	double vin: unit	ñ	0 double Var	ucue[2];		cname_asgu pkq1020).	
double U: Month U: Month U: Month Mont	doubt v: ant im		double Vir	10.		0uter_asgu 0uter_asgu 0uter[1/2-3][1-3].	of short 201101.
int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: int Voad. steady: inter. North (inter. North) steader steady: inter. North (inter. North) Steader steady: inter. North (inter. North) Steader steady: inter. North (inter. North) Steader steady: inter. North (inter. North) Steader steady: inter. North (inter. North) Steader steady:	int Voad. steady: int Voad. steady: int Vead. steady:		double \z				for lfoct hors: zo
Int Verd sim: Add file Attribute Optime Attribute Optime Attribute Optime Attribute	int tend.sim. int tend.sim. int tend.sim. tendine tend.sim. tendine tend.sim. tendine tends.sim.set. tendine tend.sim.set. tendine tends.sim.set. tendine tends.sim.set. tendine tends.sim.set. tendine tends.set. tender tender.set. tender tender.set. tender tender.set. tender<		int Vic	oad_steady;		000Dic mmcl 2];	
Peteric ENCSIN (op. intrp./.psc)=oPC_JINTRY_ISINS Peteric ENCSIN (op. intrp./.psc)=oPC_JINTRY_ISINS Pactine ENCTYQ (op. sub_entpry()(a.k. op. sub_entpy()) && op. sub_entpy()(a.k. op. sub_entpy()) && op. sub_entpy()(a.k. op. sub_entpy()) Pacter physic 30 Pacter physic 30 Pacter physic 40	Termporary Variables 3 lines 3 lines 30 aetime ENDSM (op. langt, trype)(==0°C, LINEPT_ENDS Pactine ENDSM (op. langt, trype)(=0°C, LINEPT_ENDS aetime ENDSM (op. langt, trype)(=0°C, LINEPT_ENDS Pactine Pacter pactine ENDSM (op. langt, trype)(=0°C, LINEPT_ENDS Image 31 lines 30 Bactine NEXTQ (op. langt, trype)(=0°C, LINEPT_ENDS Image 31 lines Image 30 Bactine NEXTQ (op. langt, trype)(=0°C, LINEPT_ENDS Image aetime NEXTQ (op. langt, trype)(=0°C, LINEPT_ENDS Image langter Image aetime NEXTQ (op. langt, trype)(=0°C, LINEPT_ENDS		int	nd sim:		C7	
Temporary Variables 3 lines 1 Packet's plot	Temporary Variables 3 lines 30 #define NEXTQ (mpth) type)=0PC.NTRFT_REMC 30 #define NEXTQ (mpth) type)=0PC.NTRFT_REMC 31 #define NEXTQ (mpth) type)=0PC.NTRFT_REMC 32 #define NEXTQ (mpth) type)=0PC.NTRFT_REMC 31 #define NEXTQ (mpth) type)=0PC.NTRFT_REMC 32 #define NEXTQ (mpth) type)=0PC.NTRFT_REMC	J				acetine AKKIVAL (0)_InitrpL(ype()==0PC #define ENDSIM (0)_InitrpL(ype()==0PC	INTRPT_ENDS
30 #define NEXTO1 (go _intrpit type)==0PC_INTRPT REMC Packet* pkpr. Packet* pkpr.	30 #define NEXTQ() (op_intrpl.type()=OPC_INTRPT_REMO 31 #define NEXTQ() (op_intrpl.type()=OPC_INTRPT_REMO 20 #define NEXTQ() (op_intrpl.type()=OPC_INTRPT_REMO	Ľ	11-0-11-		0 finor	#define NEXTO (op introt type()==OPC	INTRPT REMO
Parket* pkpr:	Packa [*] pkpr: Packa [*] pkpr: int ijsr.dsUrge (op. intrpl.type)==OPC_INTRPT_REMO	-1	emporary Variable	S	S III ES	30 #define NEXTOI (an introt type()==OPC	INTRPT REMO
	int ijste.dstuppe:		Packet* pkptr:			#define NEXTO2 (on intrust type)==OPC	INTRPT REMOT

Tocess Model Report: mm2qr_scheduler Fri Jan 7 06:50:29 1994 Page 1 of 13 IMV20/RL Uniform Traffic System cheduler Model			the statement of the st
/////2/Q/RL Uniform Traffic System cheduler Model	Process Model Report: mm2qr_scheduler	Fri Jan 7 06:50:29 1994	Page 1 of 13
cheduler Model	MM/2/Q/RL Uniform Traffic System		
	cheduler Model		

_				
	Function	Block	Yes	
	Temporary	Variables	Yes	
Summary	State	Variables	Yes	
	Header	Block	Yes	
	Number of	States	4	

Ider Block 32 lines	
#include csudith.h> #include csudito.h> #include cmath.h>	
typeded struct	
it store init store init store	
int len: int type: double cinne:	
} channel_asgn:	
typedd struct { int slot: int wavchength: int wavchength:) other_aspn: ctaamel_aspnpdd[1025];	
odber_asen obert (1923)15); inter_asen obert (1923)15); int no_in_sve.slot_no1.slot_no2.slot[130][10]; double time[2];	
<pre>#define ARRIVAL (0p_intrpLtype()==OPC_INTRPT_STRM) #define ENDSIM (op_intrpLtype()==OPC_INTRPT_STRM) #define ENUTQ (op_intrpLtype()==OPC_INTRPT_RENDSIM) #define REXTQ (op_intrpLtype()==OPC_INTRPT_RENDTE && 9: intrpL code()==1 && 0: op_isuble_intpt()) #define NEXTQ (op_intrpLtype()==OPC_INTRPT_RENDTE && 0: intrpL code()==1 && 0: op_isuble_intpt()) #define NEXTQ2 (op_intrpLtype()==OPC_INTRPT_RENOTE && 0: intrpL code()==1 && 0: op_isuble_intpt()) #define NEXTQ2 (op_intrpLtype()==OPC_INTRPT_RENOTE && 0: intrpL code()==2 && 0: op_isuble_intpt()) #define NEXTQ2 (op_intrpLtype()==OPC_INTRPT_RENOTE && 0: intrpL code()=2 && 0: op_isuble_intpt())</pre>	

Î

schedu	er Model
ŝ	te 0: init (Enter Execs) forced, 40 lines
	[_inverse(0.1.&z); /* 90% confideme interval */
<u>~</u>	op ima_sim_attr_get(OPC_IMA_INTEGER.vv.&mode_no); op ima_sim_attr_get(OPC_IMA_INTEGER.vv.&modergth_no); op ima_sim_attr_get(OPC_IMA_INTEGER.vir.&modersit); op ima_sim_attr_get(OPC_IMA_INTEGER.vir.⋅_no); op ima_sim_attr_get(OPC_IMA_INTEGER.vir.⋅_no);
6	op_ima_sui_aur_gen.orimA_inter.cr.s. scsn); op_ima_sim_atr_get(OPC_IMA_DOUBLE, aka);
	own.id=op_idi_set(f):src.id=op_topo_int_assoc(own_idi0); op_inna_obl_attr_get(src.idi-intesartival_args^kan_arg[0]);
5	av_rate=atof(&av_arg[0]); av_rate=1/av_rate:
2	N=wavelength_mo*fm_sizz: svc_time=double/N(double)slot_nol ; N1=floor(svc_time); svc_time=(double)N(double)slot_no2: N2=floor(svc_time);
20	ii (sn==0) sn=-1; ii ((slot_nol-slot_no2)*sn >=0) svc_time=(double)fm_sizz(double)slot_no2; else svc_time=(double)fm_sizz(double)slot_no1;
25	svc_dist=pp_dist_load(*exponential`svc_time()); svc_dist=pp_dist_load(*unitorn=int*i.(double)mode_no); dist_app_dist_load(*unitorn=int*i.(double)mode_no); cls_dist=pp_dist_load(*bernoillit*a());
30	inc=10, no, in, svv=0t, load_prev=d.*slot, no1+(1-a)*slot, no2/*arv_rare*svc_lime(double)N; load_steady=0t; end_sim=0t, n0=600t, n1=800t, op_subq_flush(0); op_subq_flush(1);
R	for (i=0: i=2: i++) [total_pk(i)=0; processed_pk(i)=0; time(i)=0; total_time(i)=0; det(i=0;i==N1) [i<=N2:i++) [tot(i=0;i==N1) [i<=N2:i++) [tot(i=0;i=1)] [i=0, i=0;i=1)]
35	pd(i).src=0;.pd(i).dst=0;.pd(i).len=0;.pd(i).type=0;.pd(i).ctime=0; for (j=0; j<15; j++) (other(i)[j).slot=0;.other(i)[j]).wavelength=0;.other(i)[j].ten=0;.
	for (i=1). for (j=0;5:waredengh_noj++) for (j=0;5:waredengh_noj++)

-1

Fri Jan 7 06:50:30 1994 Page 4 of 13

Process Model Report: mm2qr_scheduler MM/2/Q/RL Uniform Traffic System scheduler Model forced, 10 lines

 State 1:
 pk prepare (Enter Execs)

 src=(in)/sp dist_outcome(scc_dist):
 as=(in)/sp dist_outcome(scc_dist):

 dsr=(in)/sp dist_outcome(scc_dist):
 if (dat = sc) goto next:

Ś

byre=(in)op_dist_outcome(cis_dist):temp=op_dist_outcome(svc_dist): if (type=1) [uota_pk[1]=1: total_ime(1]+=temp;] clas [total_pk[0]+=1: total_ime(0]+=temp,] pkptr=ap_pk_get(op_intrtp_strm)); op_pk_ndd_get(pkptr: type=1:op_pk_ndd_get(pkptr: dist: dist: 10_0_pk_ndd_get(pkptr: svc=itemp); op_pk_ndd_get(pkptr: type=1:opp);

pk prepare (CET's) (1) : schedule

State 1: CET Cond: #0 Exec: Trans:

ţ	.00	
old	с с.	
GET	Cond:	(ARRIVAL)
¥	Exec:	
	Trans:	pk_prepare
Ē	Cond:	(default)
ž	Exec:	
	Trans:	idle

1	1	7
÷.	-	•

M/2/C	RL Uniform Traffic System r Model	WW/2/Q schedule	/RL Uniform Traffic Sys ar Model
			in 9. Mar 14.
GET	Cond: (ARRIVAL)		e Z: 1016 (ET
¥.	Exect :		If (:IOau_Steauy)
CET	itans. Pr. prepare Cond: (default)		temp=(total_time[1]*s]
Ŧ	Exec		if (temp>0.999 & & ten
	Trans: Idle	2	
5	Cond: (ENDSIM)		for (i-0: i-2: i+1)
¥	Exec: record stats(): Trans: idie		{tstart[i]=0p
5	Cond: (NEXTQ &&: (ARRIVAL)	ġ	stat_ct=0, next_st
¥	Exec: : schedule	01	-
			if (MEASURE)
	an sheet free to be and a free of the second s	51	temeen sin time():
80	B 3: SCIPEQUIE (ETHER EXECS) IUCEU, OJ IIIES Ir (ARRIVAL)	1	for (i=0: i<2; i++)
			ilerioon fifierens
	if (type=1) i=slot_nol; clse i=slot_no2;		queue[i]=0; tstart
~	11 ((10_10_5vc+1) > N)	20	
۰ 	if (op_subq_pk_insert(type.pkpt.OPC_QPOS_TAIL))=OPC_QINS_OK)		stat_ct+=1; if (stat_ct+=newt_stat)
	{ print('error inserting into queue\n*); op_pk_destroy(pkpu);) elso if (had istady)		
_		24	stepz: eerial col/0.400.8
9	temp=op_sim_time(): queue(type)+=qsize(type)*(temp-tstart(type)): tstart(tyne=l=temn: csize(tyne=l=on subd stat(tyne.OPC_OSTAT_PKSIZE):		if (temp>0.4) goto
			if (temp<0) goto s sten3.
_			scrial_col(0,200.8
15	cuse if (!find_resource(pkpty))	30	if (ratio>temp) go
			col(0,40.&temp):
	ii (op_smbq_pk_insert(iype,pkprt.UFC_QFUS_IALL)=UFC_QUNS_UK) { mint("**ror inserting into guewe\n"); on pk destrow(nknr); }		if (temp>0.1) goto
	else if (load_steady)	35	sten2 1:
ล		3	serial_col(1,400.8
	ucmp=op_aum_unne(); queueltype_H=quiziettype[]*(ucmp=ustanttype); tstart[type]=tcmp; qsize(type]=op_subq_stat(type.OPC_QSTAT_PKSIZE);		if (temp>0.4) goto
			sten3 1:
25		40	serial_col(1.200.8
I			if (ratio>temp) go sten4 1:
	if (NEXTQ2)		col(1,40,&temp);
-	ratio-op_stat(0.0PC_QSTAT_PKSIZE);	57	if (tempc=0.1) cn
8	for (j=Chicratic);++) if (/cn in revealed mc3) ~- N)	2	step5:
	(Inc_m_svc*suc_noz) <= !v)		if (nO>n1) { n1=n
	pkpti=op_subq_pk_remove(0.0): if (find assource(spert))		cise { 110~110~2; 116
35	if (op_subq_pk_insert(0,pkptr.OPC_QPOS_TAIL):=OPC_QINS_OK)	20	op_intrpt_schedule_s
	<pre>{ print('error inserting into queue\n', op_pk_destroy(pkpt); }</pre>		_
	ratio-op_subd_stat(0.0PC_0STAT_PKSIZE); temp=op_sim_time();		if (cnd_sim & & !ENDSIM) on sim end(reachi
40	u (rauo:⊐şısıze/u) && toad_staady) {queue/0}+=qsize/0]*(tannı-tsiari(0): tsiari(0]=tennıx qsize(0]=ratio;}]	
	A DECEMBER OF A		
	ratio=op_stat(1,OrC_Q>1At_rKSUZE);	Stat	te 2: Idle (CE

Process Model Report: mm2qr_scheduler	Fri Jan 7 06:50:30 1994	Page 5 of 13
MM/2/Q/RL Uniform Traffic System		

Fri Jan 7 06:50:31 1994 Page 6 of 13

Process Model Report: mm2qr_scheduler

tte 2: Idle (Enter Execs) unfor	unforced, 54 lines
n (.ioase.aoy) kampelosal_time(11*siol_nol+toal_time(01*siol_no2))(N*op_sim_time()*load_prev): if (tamps0.999 &&k tempc1.001)	
<pre>lod stready=1: for (i=0, i=2; i++) fact (i=0, i=2; i++) stat_c=0-for stat=800, op introt schedule self(0.0PC_QSTAT_PKSIZE); queuclij=0; stat_c=0-for stat=800, op introt schedule self(0.0 sim time(DFL));</pre>	queue(i)=0: }
iť (MEASURE)	
the similar of the contract of	
quene[i]+=qsizz[i]*(emp-start[i]); stat[sat_cr[j]])=quene[j]/inc; quene[i]+@t:start[i]=(emp:qsize[i]=qp_subt_stat(i,OFOSTAT_PKSIZE);	
star_c+=1;	
If (stat_ct==cxt_stat)	
step2: seria_coi(0.400.&temp); fi (temmed): a goos step5; fi (temmed): a on cards.	
step3: scrib_c01200,∶ if (ratiossem) one step3	
stept. col(0.40.&temp): if (temp-6.1.) goto step5:	
step2_1: serial_col(1.400.&temp); if (cemped) goo step2_1; if (cemped) goo step2_1;	
step3_1: fraid-scrut(1200,&rratio); fraid-scrupp, goto step5; step4_1:	
col(1,40,&temp); if (temp==0.1) end_sim=1;	
step5: if (ab-n1) { a =n1*2; aext_stat=a1; } clsc [n4=a0*2; next_stat=n0; } op_larrpt_scheduk_set(op_sim_time()+inc.0);	
if (end. sim && !PNDSIM) op. sim end(.reach.ing.steady.state);	

State 2: idle (CET's)

Proce	es Model Benort: mm2er scheduller
WW2	33 model report impart_screeter
sched	luler Model
10	
	s≡(double)stat_cv(/double,¥0: b=1bor(s). ∧=0. for (i=0ic40):i++)
15	y(i)=0; for (j=0;2cb;++) y(i)+=stat(*b+j)[d]: y(i)=y(i)(b; xt=y(i):
20	$\begin{aligned} x = x^{-1}(t_1) := a(t_1) \\ for (1=t_0) := a(t_1) : (1=y_1(t_1) \cdot x; t_2 = t_1^{-1}); \\ x = x^{-2} \cdot y_2 \\ x = x^{-1} \cdot x = x(t_1) = x \\ x(t_1) = x \cdot x = x(t_1) = x \end{aligned}$
25	for (i=1: i<=N1 ii (<=N2: i++) if (pd(i):ac:=0) if (pd(i):yye==1) time[1]+=(op_sim_time()-pk(i).cime): cfsc time[0]+=(op_sim_time()-pd(i).cime):
30	op_stat_write_scalar(*10ad*(total_time[1]*\$100_n01+total_time[0]*\$100_n02WfW90_stat_time())); op_stat_write_scalar(*ou-time[1]*\$104_n01+time[0]*\$101_n02/N/W90_stat_time()); op_stat_write_scalar(*0=*time[1]*\$104_n01a_pt[1]V(double)(total_pt[1]);
35	op.stat_write_scalar(`02.*xt(0)); op.stat_write_scalar(`02.*delpt)[V(double)processed_pk[1]); op.stat_write_scalar(`02.*delpt)[V(double)processed_pk[0]); op.stat_write_scalar(`oq1'sst[1]);
40	op_stat_write_statat(cz. s.styl).
	f invers (12)
45	double x,*z1:
	մօսble չդրոնդելդեշրծյունգնելցնեց։
20	x=1-x02; pd=-0.32232313108840; p1=-1.0; p2=0.34224208854760; p3=-0.020423121024540; p4=-0.4536422101486-4; q0=0.09934845206666; q1=0.38858157049560; q2=0.53110346236666; q3=0.1035377228560; q4=0.38560700654c-2;
22	p=x. ii ((p>0.5) p=1.0-p: y=squt(-log(p*)): * 21=y+(p0+y*(p1+y*(p2+y*(p3+y*p4))))/(q0+y*(q1+y*(q2+y*(q3+y*q4)))): ii (x<0.5) *21=*2!: }
99	serial_col(t.t.pl) int t.n: double Pl.
65	ا int ijkkb: double p.xx1.x2.cc1.c2.ss1.s2.y(400):

Process Model Report: mm2qr_scheduler	Fri Jan 7 06:50:31 1994	Page 7 of 13
M/M/2/Q/RL Uniform Traffic System		
scheduler Model		





tion Block	ecord_stats()	-	ini.	double	extern int	extern double	extern channel
			ijihut	x.s.y[40].xx[2].ss[2]:	<pre>slot_no1.slot_no2.slot[130][10];</pre>	time[2];	_asgn pkt[1025]:

Proces	ss Model Report: mm2qr_	scheduler	Fri Jan 7 06:50:32 1994	Page 9 of 13
MM/2/ schedu	Q/RL Uniform Traffic Sys iler Model	tem		
70	<pre>b=next_stat/n; k=n/2; for (i=0;i<n;i++) pre="" {<=""></n;i++)></pre>	c=0; x1 =0; x2=0;		
<u>۲</u>	y[1]=0; for (j=0; th) y[1]=y[1]/b: x+=y[if (i k x1+=y[1];	y[i]+=stat[i*b+j][t]: i]: else x2+=y[i]:		
5	x = x/n; x1 = x1/k; x2 = x2 for (i=0:i <n:i++)< td=""><td>Øk: s=0: s1=0: s2=0: c=0: c1=0: c2=0:</td><td></td><td></td></n:i++)<>	Øk: s=0: s1=0: s2=0: c=0: c1=0: c2=0:		
80	{ j=y(i)-x: s+=j*; if (i<(n-1)) c+=j*(if (i <k)< td=""><td>(y[i+1]-x);</td><td></td><td></td></k)<>	(y[i+1]-x);		
85	{ j=y(i}-x1: s1- if (i<(k-1)) c. Plee }	+=]*); +=]*(y[i+1]-x1);		
6	;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;	+= *]; 2+=j*(y[i+1].x2):		
	} if (s==0) *p1=0; else * }	p1=2*c/s-(c1/s1+c2/s2)/2:		
56	col(t.a.p1) int t.n:			
100	double *p1: { int i,j.b: double p.x.s.y[400];			
	b=next_stat/n; x=0; for (i=0;i <n;i++)< td=""><td></td><td></td><td></td></n;i++)<>			
	y[i]=0; for (j=0;j <b;j++) y[i]=y[i]/b; x+=y[</b;j++) 	y(i)+=stat[i*b+j][t]: i);		
110	<pre>/ x=x/n; s=0; for (i=0;i<n;i++) j="y}<br" {="">s=s(n_1); s=sort(s(n)).</n;i++)></pre>	(j)-x: s+=j*j; }		
115	<pre>5-w(n-1), s-squ(w(n), if (s=0) *p1=0; else * printf(~\n\$d(\$d): 8: }</pre>	p]=z*¥X: £\t&f *,staLct.t.x.*pl):		
	int find_resource(packet) Packet* packet;			
170	, in <i>i</i> i.	i.j.k.ap.ap1.nd.ct.oh.go_back.waste.i_ct conflict[130][2],empty[10],store[1025]	ь.fit.next.sc.dt.p; [[4].temp[130].asgn[130].tmb[130].tm	
125	double extern int extern other_asgn	svc; no_in_svc,slot_no1.slot_no2.slot[130][other[1025][15];	10);	

Process	Model Report: mm2ar scheduler	Fri Jan 7 06:50:32 1994 P	age 10 of 13	
M/M/2/0	VRL Uniform Traffic System			Γ
schedult	er Model			
	extern channel_asgn pkt[1025];			
	op_pk_nfd_get(packet, *src*, &so); op_pk_nfd_get(packet, *cs op_pk_nfd_get(packet, *svc*, &svc); op_pk_nfd_get(packet, *cy	&dt): ∍ • (&tp):		
051	ci=0: for (j=0: j <wavelength_no: empty[j]="0:</td" j++)=""><td></td><td>~</td><td></td></wavelength_no:>		~	
135	for (i=0:i <fm_size:i++) {</fm_size:i++) 			
	<pre>conflict[i][0]=-1: conflict[i][1]=-1: for (k=0: k-wavelength_no: k++)</pre>			
140	if (slot(iRb=0) cmpy(k)=1: if (pdt(slot(iRb],src=sc) confic(iRb)=k: if (pdt(slot(iRb],dst=dt) confic(iRb)]=k: }			
145	وه back=0; ii (دومالندرایا[0]==-۱ گرگ دومالندرایا[1]=–۱) for (ا=0: لاحمعموادیوای ۲۵۵ k++۱)			
150	{ if ((slot[i][k]!=0 i==(fm_size-1)) && empty[k]>(f			
	if (stotfi]k}!=0) j≕-empty[k]-1: clse j≕i-emp if (j<0) oh=0: oto	(k):		
155	cise if ((conflict[j][0]==+1 conflict[j][0]==k) else oh=2:	&(conflict[j][1]==-1 conflict[j][1]==	k)) oh=0;	
99	ii (oh=.2 && empty[k <2) { empty[k]=U: co insert. if (ct==0)	mue:)		
3	if ($slot(i) k =0$ [go_hack) $store[0] 0]=i$ store[0][1]=k: $store[0] 2]=mpty[k]$; store }	mpty[k]: else store[0][0]=i-empty[k]+1 J[[3]=oh;		
165	$\int_{l}^{cise} for (j=(ct-1); j>=0; j-)$			
	if (store(j][2] <empty(k] (store(j)[2]<br="" ="">{</empty(k]>	=empty[k] && store[j][3]>oh))		
170	store[j+1][0]=store[j][0]: store[store[j+1][2]=store[j][2]: store[1][1]=store[j][1]: 1][3]=store[j][3]:		
	if (store(j)[2]>empty[k] /l (store(j)]2	=====================================		
175	where \mathbf{r}_{i} is the set of t	j+1][(0]=i-empty[k]: else store[j+1][[0]= npty[k]: store[j+1][3]=oh:	ei-empty[k]+1;	
180	$ f(j=0) j=1; goto store_ct; $			
	11 (go_natck) goto back; cmpty[k]=0; ct+=1;)			
	-			

Proc	cess	Model Report: mm2qr_scheduler	Fri Jan 7 06:50:33 1994	Page 11 of 13	
N S	V2/Q	/RL Uniform Traffic System			
	185	-			
	3	clse			
		{ for (k=0:k <wavelength_no:k++)< td=""><td></td><td></td><td></td></wavelength_no:k++)<>			
	5	() (() amoto(k)2())			
	195	i (stolijk]=0) ampy[k]=1: i (stolijk]=0) ampy[k]=1: ii (cmpy[k]=0) continue: j=i=mpy[k]-1: ii (j=0) oh=0;			
	200	cise if ((conflict[j](0)==-11 conflict[j](0)==k), cise oh=2, if ((conflict[j](0)==-11 conflict[j](0)==k)&&(c cise if (oh==-11 oh==1)&& cmpy(k)=21 (id=2) if (oh==-12 oh==1)&& cmpy(k)=22 (id=2)	&(conflict(j][1]==-1 conflict(j][1] mflict(i][1]==-1 conflict(i][1]==k) && cmpty[k]<3)) { cmpty[k]=Y. o	==k)) oh=0:) oh=oh; ontinuc: }	
	205	go, back=1: gouo insert: back: j pu/kl=0: ct+=1:			
	10				
·	1	if (tp=1) nd=stot_no1; else nd=stot_no2; (c=cd: wast=c); next=f); age[0]=-1; age[0]]=0; for (i=d); i=dn_size; i++) { mb[i]=0; imp_mb[i]=0; i for (i=d); i=cci; i++)			
-	CI7	(torofi]]3]=0) fii = 0; ets eff (snor[i]]3]=3) fii = 2; etse fit = 1; j=store[i][2]-fit;			
	220	if (next) { if(j==nd)			
	225	{ k=j-nd+fit: if (fic=apl&&&k <ap)< td=""><td></td><td></td><td></td></ap)<>			
		{ for (oh=store[i][0]; oh<(store[i][0]+store[i][2]); if (mb[oh]) { k=-1; break; } if (P'=1)	oh++)		
	230	{ ap=k: ap1=fit: temp{i_ct]=i:			
	235	for (to)= 0 , here hor (m, size: out+) imp_mb(ol k=store[10]-1: if (k=0) k=0; go_hak=store[10]+store[1][1]+1: if (go_for for (to)=k: oh=go_back: oh++) imp_mb(o	=0: aack>fm_size) go_back=fm_size: J=1:		
~	240	if (j=r(t-1)) {			
		wastc+=apl: i_ct+=1: if (wastc <asgn[0] #="")<="" asgn[0]="1" td=""><td></td><td></td><td></td></asgn[0]>			



ss Model Report: mm2qr_release Fri Jan 7 06:28:22 1994 Page 1 of 2 20.DRL Uniform Traffic System se Model	ss Model Report. mm2qr_retease Fri Jan 7 06:28:22 1994 Page 1 of 2 2/0.RL Uniform Traffic System se Model			
2/Q/RL Uniform Traffic System se Model	2/0/RL Uniform Traffic System se Model	ess Model Report: mm2qr_release	Fri Jan 7 06:28:22 1994	Page 1 of 2
se Model	se Model	2/Q/RL Uniform Traffic System		
		se Model		

	-			1
	Function	Block	No	
	Temporary	Variables	Yes	
Summary	State	Variables	No No	
	Header	Block	Yes	
	Number of	States	-	

8	der Block		18 lines	
	typedef struct	_		
	ini	STC:		
	int	dst;		
_	int	len:		
	int	type:		
	double	ctime:		
) channe	el_asgn;		
	typedef struct	2		
	int	slot;		
	int	wavelength:		
_	int	len;		
	} other_a	asgn;		
-	extern int		no_in_svc.slot_no1.slot_no2.slot[130][10];	
	extern dou	ible	time[2];	
	extern cha	unel_asgn	pkt[1025];	
-	extern othe	er asen	other[1025][15];	

t		arrived, to me
	p_intrpt_type()==OPC_INTRPT_STRM)	
	r pkptr=op_pk_get(op_intrpt_strm()): op_pk_nfd_get(pkput,"wavelength",&wave	ngth);
-	op_pk_nfd_get(pkpt.'timeslot×lot); i=slot[timeslot][wavelength];	
	<pre>if (pkt[i].type==1) { time[1]+=(op_sim_time()-pkt[i].ctime): no_in_svc-=slot_no1;</pre>	pe=1:)
_	<pre>clsc { time[0]+=(op_sim_time()-pkt[i]) ctime); no_in_svc=slot_no2; type=2; }</pre>	
	for (j=0. j <pkt[i].lcn; j++)<="" td=""><td></td></pkt[i].lcn;>	
_	<pre>{ for (k=other[i][j] slot; k<(other[i][j] slot+other[i][j].len); k++) slot[k][other[i][</pre>	wavelength]=0;
	other[i][j].slot=0; other[i][j].wavelength=0; other[i][j].len=0; }	
	pkt[i].src=0: pkt[i].dst=0: pkt[i].len=0: pkt[i].type=0: pkt[i].ctime=0: op_pk_destroy	kptr);
_	own_id=op_id_setf(); src_id=op_topo_in_assoc(own_id,0);	
_	op_intrpt_schedule_remote(op_sim_time().type.src_id);	
_		

discard (CET's)	(1)		discard	
State 0:	CET Cond:	#0 Exec:	Trans:	

Process	s Model Report: mm2qr_scheduler	Fri Jan 7 06:50:34 1994	Page 13 of 13	
M/M/2/(scheduł	2/RL Uniform Traffic System er Model			
]
	assign:			
305	for $(k=1; k \le N) \parallel k \le N2; k++)$			
	if (pkt(k].src!=0) continue;			
	<pre>pkt(k].src=sc: pkt(k].dst=dt: pkt(k].len=asgn(1); if (in=-1)</pre>	kt[k].type=tp: pkt[k].ctime=op_sim_time(
310				

5 for (k=1: k<=N1 k<=N2; k++) {	ii (pkl(kl.src=sc: pkl(kl.ds=di: pkl(kl.len=sgn(l); pkl(kl.type=fp: pkl(kl.ctime=00_sim_time(): ii ((p=1)) i	nd=slot_nol: no_in_svc+=slot_nol; processed_pk[]+=1; delay[1]+=op_sim_time(}-op_pk_creation_time_get[packct]; }	clse defailed in 20, no. in 2vet=slot, no2; processed_pk(0)+=1; defay(0)+=apsim_time()-appk, creation_time_get(packet);	<pre>for (i = 2: ic=asgn[1]; i++)</pre>	<pre>cise if (store[age(i)]] 3]=3.) (c = 1; nh = 1; oh = 1; oh = 1; i c = if (store[age(i)]] 3]=3.) (c = 2; nh = 2; nh = 1;) for (j=nh; j(oh=v; j+1; stol)] store[age(i)][1]]=4; obser[k][1,2] stor=nd; obter[k][1,2], wavelength=store[age[n]][1]]=4;</pre>	<pre>i=asgn[1]+1: on=store[asgn[1][0]: if (store[asgn[1]]]3==3) oh+=1: else if (store[asgn[1]]]3==3) oh+=1(store[asgn[1]]]-had): for [ache, j=(oh+ad); h= store[3][store[asgn[1]][1]]=k; for [ache][A=2][acheao; other[k][=2], waveferent=store[asgn[1]]], other[k][=2][ken=nd;</pre>	op_pk_ndi_set(packet 'times i ot 'ob); op_pk_ndd_set(packet 'wave iength': store[ssgn[i]][1]); op_pk_send_delayed(packet(0,svc); break:) return i:
305	310		315	320	325	330	335

B.5 Single Class, Client/Server System

B.5.1 Blocking System

The termination criteria and the seed used here is the same as that in Section B.3.1.

<u>M/M/1/B/RL:</u> OPNET report for the *scheduler* processor model. The *release* model is identical to that in M/M/1/B/RL uniform traffic system.



	Second labour -		This is a contraction of the	
33	S MOUEL LEUOIL	. mm i Dr CS screduler	1 Inu Jan 0 23:10:02 1934	Lage 2 of 10
M/1/M	3/R Client/Serv. ler Model	er System		
Sta	te Variables		38 line	S
	Objid	Nown_id;		
	Objid	\src_id;		
	ŭ	Inode_no;		
	Ħ.	Server_no;		
<u>~</u>	ti i	\wavelength_no:		
_	ij.	VIII_SIZE:		
	net			
		\slot_no;		
\$	ă.			
2	unt domble	VNI: Verseffici		
	double	kwo time:		
	char	Varv are[20]:		
	double	larv rate:		
15	Distribution	*\tfc_dist:		
	Distribution	*\srv_dist;		
	Distribution	*\src_dist:		
	Distribution	*vdst_dist:		
	Distribution	*'svc_dist;		
ន	long int	viotal_pks;		
	IUI BUON	Violal_pku:		
	long unt	Velocked_pks:		
	houg mt	Vhlocked c nks		
25	lone int	Wheeked c nku:		
	double	kcu max;		
	double	kcu_prev;		
	double	\pb_prev:		
;	double	pbc_prev;		
8	double	Voad_prev;		
	double			
	n i	Vpr_count:		
	1.1	u_count. Ven streader		
ž	1.1	table standar		
<u>.</u>		Vu_suctury.		
	1.1	Vicad stands.		
	1.j	tready.		
ē	mporary Van	ables	3 lines	
	Packet* pkptr;	:		
	int src.ds	CLJ:		
	double rano,u	emp;		

	Model Becord: mm1hr ce eched	ular	Thu lan & 23-10-02 1004	Pana 3 of 10
MW1/E	VR Client/Server System	Ba	1100 0001 0 0001 0 000	
chedu	er Model			
Sta	te 0: init (Enter Execs	(1)	forced	41 lines
<u>~</u>	op. ima. sim. attr. get(OPC. IMA. INTI op. ima. sim. attr. get(OPC. IMA. DOU	GER, 'w & Mode_no); GER, 'w & wavelength_no); GER, 'r . & kin_size); GER, 'r . & server_no); GER, 'ns . & server_no); BLE, * tp . & traffic);		
10	own_id=op_id_self(); src_id=op_topo_in_assoc(own_id.0); op_ima_obj_attr_get(src_id,*interar	:rival args'.&arg[0]):		
	arv_ratc=atof(&arv_arg[0]); arv_ratc=1/arv_rate:			
15	N=wavelength_no*fm_sizz: svc_time=(double)N(double)sloL_no: N1=floor(svc_time); svc_time=(double)fm_sizz/(double)slot.	9		
50	sw dist=op_dist_load(*oxponential tic_dist=op_dist_load(*oxponential.*(c srv_dist=op_dist_load(*uniform_int src_dist=op_dist_load(*uniform_int) dist_dist=op_dist_load(*uniform_int)	-svc_time.0): Jouble/(2*traffic*server_no).0) -1.(double/(2*server_no)); .(double/(1+server_no).(doub) .(double)(1+server_no).(doub)); le)mode_mo); le)mode_mo);	
<u>م</u>	total_pix=0; total_pix=0; blocked_pix=0; blocked_pixu=f blocked_c_pix=0; blocked_c_pixu=0; refereed_pix=0; no_in_yvv=0;); =0; timc=0;		
2	cu_max=(double/KN1*suc_no/(double) bad_prev=arv_rato((double)wavelength cu_steady=0; pb_steady=0; pbc_steady=	in; ∟no; =0: load_steady=0: ready=0:		
35	for (j=0::<=N1:;++) { pkt[i]src=0; pkt[i]ds=0; pkt[i].len for (j=0; <1.5; j++) (other[i][j].sto=0; other[i][j].	i=0: pkt[i].ctime=0: wavelength=0: other[i][j].len=0	F	
40	for (i=0:i <fm_size:i++) for (i=0;i<wavelength_no:i++) slot<="" td=""><td>(i)[i]=0:</td><td></td><td></td></wavelength_no:i++)></fm_size:i++) 	(i)[i]=0:		
45	te 0: init (CETs)			
방幕	r Cond: (ARRIVAL) Frac:			
21	Trans: pk prepare Cond: (default)			
	Trans: idle			

	LIDCESS MODEL LEPOLT. HIHILIN CS SCHEMMEN	hu Jan 6 23:10:03 1994 raye 4 0
Client/Server System Jodei	M/M/1/B/R Client/Server System scheduler Model	
	State 1: pk prepare (Enter Execs)	forced, 19 lines
if ('pb_steady)	temp=(int)op_dist_outcome(tfc_dist); if (term) /* clipm-cexer traffic */	
icmp=:double/(bicxled_pks+bicxled_pku)/(double/icoral_pks+rotal_pku). ir ich_revici-annendoh_merer	entronmarker distruction	
the product of the pr	5 if (ration=server_no)	
ctse ratio=U; if (ratio > 0.999) & & ratio < 1.001) ob stead v=1;	{srt=ratio; dst=(in)/op_dist_outcomet dst_dist; total_pks+=t; else	
else ph_prev=(emp:	[dst=ratio-server_no: src=(int)op_dist_outcome(src_dist): tot	1_pks+=1:)
if (!pc_steady)	10 else	
temp=(double)(blocked_c_pks+blocked_c_pku)(double)(total_pks+total_pku);	src=(int)op_dist_outcome(src_dist):	
ii (poc_previ=4)/rauce=cempeoc_prev; clse if (temp==0)/ratio=1;	next: dst=(int)op_dist_outcome(dst_dist);	
else ratio=0; if (ratio > 0.999 && ratio < 1.001) pbc_strady=1;	15 if (dat $==$ src) goto next: total_pku+=1:	
cise poc_prev=temp:	temp=op dist_outcome(svc_dis); total_time+=temp; pkpr=op_pkee	(op_intrpt_strm()):
n (u. sakau) aca pa_sheauy aca pa_sheauy) (cauy=1, if (fready)	op pk, nig seuppur, sice arc), op pk, nig seuppur, die arch arch arch arch ar	mu securation sec interpret
(ratios):		
for (i=1: i<=N1; i++)	State 1: pk prepare (CET's)	
if (pk(li)src!=0) ratio+=(op_sim_time()-pkt[i].ctime); tentn=(bine+ratio)*slot_no*N[/(N*oo_sim_time());	CET Cond: (1)	
pk_count=2*floor(temp); tr_count+=1;	Trans: schedule	
(ready)	State 2: idle (Enter Execs)	unforced, 64 lin
temp=slot_no*total_time(//*0p_sim_time()); ratio=temp/load_prev:	if (!load_steady && released_ph>0) {	
ii (rauo.9.1995) & & ano.c.1.000 & & 'LANDSUM) 0p.5im_cmd(*reachting steady state':):	<pre>temp=slot_no*total_time(N*op_sim_time()); cario=tempVada_prev; cario=tempVada_prev;</pre>	6 Berne (17 , 6).
	5 temp=louble/joost_pray(uouble/joost_pray(uouble/joost_pray, z. szvec_ if (ratio > 0.999) & & ratio < 1.001 & & temp > 0.999 & & temp < 1	001)
2: idle (CET's)	load_strady=1: ratio=0;	
ond: (ARRIVAL)	10 ii (pkt(i).src)=0. ratio+=(op_sim_time().pkt(i).cume):	
(ec: : : ans: ok ovepare	cu_prev=(time+ratio)*slot_no/(N*op_sim_time()):	aksatotal aku):
ond: (dclault)	pbc_prev=(double)(blocked_c_pks+blocked_c_pku)(double)	total_pks+total_pku);
kec: ans: idle	temp=cu_prev*N1: pk_count=floor(temp): 15 temp=(dubhle)released nk/(double)nk count: rr_count=floor	cmp)+20;
ond: (ENDSIM)		
tec: record_stats(): 2ns: idle		
	if (load_steady && fready && released_ptcs(r_count*pk_count)) 20	
	if ('cu_steady)	
	ratio=0; for $(z=1 ; z=N ; t+1)$	
	25 if (pk(i)] src ² -0) nation=10p. sim. timety-pk(i) fatimets: temperimetersetions) sim (control model) and timet(): ratio=temply if (ratios 0.0998 & Ratios < 1.1001)cstability	
	else cu_prev=temp;	

Process	Model Renort mm1br cs scheduler Thu Jan 6 23-10-04 1994 Pane 7 of 10	Process Model Benort: mm1br is echeduler Thu Ian 6.23-10-03 1004 Pare 6 of 10
A/////A		MM///BLR Clam/Server Server Current Cortex and Cortex and Cortex 1200 1304 1 Cortex 2011 V
schedul	n onemocarver ogsammer	www.rut.no.eeuceever.oyseen scheduler Model
35	op_pk_nfd_get(packet, src-t&st); op_pk_nfd_get(packet; dsr't&dt); op_pk_nfd_get(packet; src-t&stv);	State 3: schedule (Enter Execs) forced, 13 lines
	ct=0; for (j=0; j=vawrelength_mo; j++), empry(j=0;	if (no.jn_yvc = NI) { f (stecesener no lidst-server no) blocket nk s+=1:
40	for (i=0):cfm_size:i++)	sets bit kotted prove
	conthict[i][0]=1: conflict[i][1]=-1: for (k=0, k=wavelength_non k++)	else ' if (find_resource(pkpu)) {
45	[{	<pre>10 if (src=serve_nol fldsc=serve_nol { blocked_pks+=1: blocked c_pks+=1: } else blocked_pht+=1: blocked_c_pku+=1: }</pre>
20		
	er_patran. if (confice[]](0]=-1 && confice[][1]=-1)	State 3: schedule (CET's)
55	for (k=0; k.cwarelength_no; k++)	Cert Cond: (1)
3	if ((slot[i][k]!=0 l i==(fm_size-1)) && empty[k]>0)	I rans: Ide
	if (sold) k -i=()) j=i-empty k -1: clse j=i-empty k : if (j=0) ob=0.	Function Block 237 lines
99	دادد if ((conflict[j][0]=-1 conflict[j][0]==kk&k(conflict[j][1]]==-1 conflict[j][1]]==k)) مh=0.	record_stars()
	ctsc otr=2; if (oh=2 & & empty[k]<2) { empty[k]=ft. continue: } 	int i.j; double cu=0
65	insett. if(ct==0)	5 extern double time: extern channel_asgn pkd[1025];
	if (slot(i) k]=018_c_back) store(0][0]=i-empty k]: else store(0][0]=i-empty[k]+1: store(0][1]=k: store(0][2]=empty[k]:	for $(=1; k=N1; i++)$ if $(pkt)[3, ex:=0)$ into $+=(op, sim_{-}time(-pkt)]$ ctime):
70	clse (or (jacto-1): ja-pt); j)	10 cu=time*stot_not(N*op_sim_time()):
	if (sore[j]2]<=mpy[k] (sore[j]]==empy[k] & & sore[j]]3}>oh))	if (up cut_max) cut=ru_max: prat_mrite_xatar(1oad-sio(_no*total_time(N*up_sim_time())): on tau mriterond/inforcestering
75	<pre>store[j+1](0]=store[j](0]; store[j+1][1]=store[j](1]; store[j+1][2]=store[j](2]; store[j+1][2]=store[j][2];</pre>	15 op_latt_mic_interact, (a) (a) (b) (b) (b) (b) (b) (b) (b) (b) (b) (b
	} if (suretj]l2>empy]k] ∥ (suretj]l2]==empyJk] && suretj][3]<=ob))	op_stat_min_statar (por , quonte processorparameter_parameter_parameter), op_stat_min_statar (pos -, (double)blocked_pk()oblockal_pk()obloal_pk(); op_stat_min_statar (pos-tatar) (post-bioked c_c) c_kk((ouble)oblockal_pk();
80	store_ct: if (stori[j]k1]=01 go_bwck) store[j+1][0]=j:empty[k]; else store[j+1][0]=j:empty[k]+1; store[j+1][1]=k: store[j+1][2]=empty[k]; store[j+1][2]=ob; break:	20 op_stat_write_scalar("pbur."(double)blocked_pku/(double)total_pku): op_stat_write_scalar("pbcu".(double)blocked_c_pku/(double)total_pku): }
	$if(j=1: goto store_ct;)$	25 int find_resource(packet)
	ii (go. back) goto back: emptylk=67. ch=1;	Packet: packet:
		int utkapapi.nact.00.go/back.waser_ctrinext.sc.dt. ont conflict[130][2],empy[10],store[1023[4],temp[130],amp[130],amp_1130];amp_1130]; Acodds conditioned (130][2],empy[130][2],empy[130][2],empy[130],amp_1130];amp_1130];
8		externation of the set
	for (k=0; kewavelength_no; k++)	externet_asgn pk(1025;

Proc	ess	Model Report: mm1br_cs_scheduler	Thu Jan 6 23:10:04 1994	Page 8 of 10	
MM	1/8/1	R Client/Server System		>	
		- MODEL			
6	2	{ if (cmpty[k]>0)			
	8	i (skol []k]=0) cmpy(]k]=1; ii (smpy(]k]=0) continue. j=i=mpy(]k]-1; if (j<0) time?			
	05	clsc if ((conflict[j](0)==) II conflict[j](0)=4), clsc $nl=2$; if (conflict[j](0)==1 I conflict[j](0)= $k_{i} k_{i} k_{i}$ clsc $nl(1)(0)==1$ and i_{i} : clsc $nl=2$; k k_{i} conflict[j](0)= $k_{i} k_{i} k_{i}$ if (($nl=2$ II $nl=1$) $k_{i} k_{i}$ empty[k]<2) II ($nl=$ go $hack$:	&&(conflict[j][1]==-1 conflict[j][1 onflict[j][1]==-1 conflict[i][1]==k; 3 && empty[k]<3)) { empty[k]=0; c]==k)) oh=0:)) oh=oh: :ontinue: }	
	10	cmpylk]=0: ct+=1:			
	15) nd=slot_no: i_crt=0; waste=0; next=0: agn(0)=-1: agn(1)=0; for (i=b:for_t_size: i+++) { mb(1)=0; mpmb(i)=0; } for (i=b:rest: i++) { mb(1)=0; mpmb(i)=0; }			
	50	{ if (store(i 3)==0) fut = 0. efcs if (store(i 3)==3) fut = 2; else fut = 1; j=store(i 2+fut:			
	35	if (next) (c) (c)			
		t k=j-nd+fit: if (fit<= apl && k <ap)< td=""><td></td><td></td><td></td></ap)<>			
<u> </u>	98	{ for (oh=store[1][0]; oh<(store[1][0]+store[1][2]); if (mb[oh]) [k=-1: break: } if (k=-1)	oh++)		
<u> </u>	33	ا الحالية المالية ال المالية المالية ال المالية المالية br>المالية المالية br>المالية المالية br>المالية المالية الم مالية المالية br>مالية مالية المالية الم مالية مالية المالية r>مالية مالية مالية مالي	n]=0; back>fm_size) go_back=fm_size. bj=1:		
71	40	-			
		if (j ≤ nd ll i==(ct-1)) {			
	45	wäsic+=ap1: L_ct+=1; if (wasic <asgn[0])<br="" asgn[0]="-1" ="">ℓ</asgn[0]>			
1	8	<pre>asgn(0)=wastc: asgn[1]=i_t: for (k=2; k<(i_ct+2); k++) asgn[k]=temp[k-2] i(asgn(0)==0) goto assgn: if (j=ad) f</pre>			





B.5.2 Queueing System

The termination criteria and the seed used here is the same as that in Section B.3.2.

<u>M/M/1/Q/RL:</u> OPNET report for the *scheduler* processor model. The *release* model is identical to that in M/M/1/Q/RL uniform traffic system.

00e	ss Model Report:	: mmtar cs scheduler	Fri. Jan 7 02:35-19 1004 Page 2 of 12
/W1	/Q/R Client/Serv	er System	1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1
B			
ŝ	ate Variables		43 lines
	Objid	'lown_id:	
	Objid	\src_id:	
	i. nt	vnode_no;	
Ŷ		bet ver _filo; \wavelength_no:	
<u>,</u>	iii.	Vm size:	
	Ē	VT1:	
	int	vslot_no;	
1	int	N.	
2	iit	- NNI:	
	double	Viraffic:	
	char	larv arg[20]:	
	double	arv_ratc;	
12	Distribution	*\tfc_dist:	
	Distribution	*\srv_dist;	
	Distribution	*\src_dist;	
	Distribution	*Vast_dist; *\sur_dist	
8	kong int	Viotal rks:	
	long int	Votal_pku;	
	long int	processed_pks;	
	long int	processed_pku:	
2	long int	stat_ct;	
3	long int	hick(_stat.	
	long int	61:	
	double	Voad_prev;	
	double	\stats[200000];	
8	double	\statu[200000];	
	double	Viotal_time;	
	double	Vdelays:	
	double	Viciayu:	
35	double	Visitaris;	
3	double	ustariu. Visizre:	
	double	vosizeu:	
	double	Agueues:	
	double	Aucucu:	
4	double	Vinc;	
	double		
	Ĭ.	Voad_stcady;	
		ALIN. SILLI.	
Te	mporary Varia	bles	3 lines
	Packet* pkptr;		
	int src.dst.i	ŝ	
	double ratio.tet	np:	

Process Model Report: mm1qr_cs_scheduler	Fri Jan 7 02:35:18 1994	Page 1 of 12
WM/1/Q/R Client/Server System		
scheduler Model		

H	Parter Block 20 lines	20
2	#include <stditb.it></stditb.it>	3
	#include <stdio.h></stdio.h>	
	#include <math.i></math.i>	
Ŷ	tymedef struct	
	int src;	
	int dst;	
2	double ctime:	
	} channel_asgn;	
	typedef struct	
15	int slot;	
	int wavelength: int len:) other search	
50	channel_agn pkt[1025]; other_agn obter[1025][15]; int no_in_svc.refeased_pk.sho[130][10]; double time;	
25	#define ARRIVAL (op_intrpt_type()==OPC_INTRPT_STRM) #define ENDSIM (op_intrpt_type()==OPC_INTRPT_ENDSIM) #define EMPTYO (op_solvg_empty(0)) #define NEXTQ (op_intrpt_type()==OPC_INTRPT_REMOTE && !EMPTYQ)	

22			
1/1 1/1	2/R Client/S er Model	erver System	
E I	te 1:	pk prepare (Enter Execs)	forced. 22 lines
	pkptr=op_p	<_get(op_intrpt_strm()):	
	temp=(int)0 if (temp)	/* client_server traffic */	
	ratio=(nt)op_dist_outcome(srv_dist);	
	if (rational for the second se	<=scrvcr_no) c=ratio; dst=(int)op_dist_outcome(dst_dist); total	_pks+=i:op_pk_nfd_set(pkpu.type.i);)
0	r (c	st=ratio-server_no; src=(int)0p_dist_outcome(src_	_dist); total_pks+=1;
	elsc		
	r src=(in	:)op_dist_outcome(src_dist);	
ŝ	next		
	dst=(ir. if (dst =	.) op_dist_outcome (dst_dist); = src) goto next;	
	total_p	u+=1; op_pk_nfd_set (pkptr.*type*.0);	
0	-		
	temp=op_d	st_outcome(svc_dist): total_time+=temp: set(pkptr.*src*.src): op_pk_nfd_set(pkptr.*dst	.dst);op.pk.mfdi settpkpts.*svc°.temp);
ŝ	te 1:	pk prepare (CET's)	
μ	Cond:	(1)	
2	Exec:		
	Irans:	schedule	

ses	s Model Report: mm1 ar cs scheduler	Fri Jan 7 02:35:19 1994 Page 3 g
M/1//	J/R Client/Server System ler Model	
Sta	te (): init (Enter Exerc)	forrad 25 lines
	f_inverse(0.1.&z); /* 90% confidence interval */	
2	op.ima.sim.attr_get(OPC_IMA_INTEGER.vv.¬e_no); op.ima.sim.attr_get(OPC_IMA_INTEGER.vv.&wwwenength_no); op.ima.sim.attr_get(OPC_IMA_INTEGER.vv.&tettin_size); op.ima.sim.attr_get(OPC_IMA_INTEGER.vv.&tettin_size); op.ima.sim.attr_get(OPC_IMA_INTEGER.vv.&tettin); op.ima.sim.attr_get(OPC_IMA_INTEGER.vv.&tettin); op_ima.sim.attr_get(OPC_IMA_	
10	own_id=op_id_self(); src_id=op_topo_in_assoc(own_id.0); op_ima_obj_attr_get(src_id^*interarrival args'&arv_arg[0]);	
	arv_rate=atof(&arv_arg[0]);	
15	N=wavekength_no*fm_size: svc_time=(double)N(double)slot_no: N1=floor(svc_time): svc_time=(double)fm_size/(double)slot_no:	
20	svc. disterop. dist_load(* exponential'avc. limte(); ufc.distrop_dist_load(* exponential'svc. limte(); ufc.distrop_dist_load(* express_limte(*, idouble); distrop_dist_load(* unitor_limt*, (double); *sccret_no)); svc.distrop_dist_load(* unitor_limt*, (double)()+scret_no)(double);)mode_no): throad = no): throad = no):
25	total_pks=0; total_pku=0; processed_pks=0; processed_pku=0; inc=10; no.in_sve=0; time=0; total_time=0; delays=0; load_prev=av_rate/(double)wavelength_no; load_steady=0; end_sim=0; nD=500; np_subq_flush(0);	
30	<pre>for (i=0:i<=N1:i++) { for (i=0:i<=N1:i++) for (i=0:i<=0; pk(1), len=0; pk(1), cime=0; for (j=0:j<1; j++) (obset(i)[1], sion=0; obset(i)[1], wavelength=0; o for (j=0:j<1; j++) (obset(i)[1], sion=0; obset(i)[1], wavelength=0; o for (j=0:j<1; j++) (obset(i)[1], sion=0; obset(i)[1], wavelength=0; o for (j=0:j<1; j++) (obset(i)[1], sion=0; obset(i)[1], sion=0; for (j=0:j<1; j++) (obset(i)[1], sion=0; for (j=0:j<1; for (j=0:j)) for (j=0:j<1; for (j=0:j<1; for (j=0:j)) for (j=0:j</pre>	ber[i][].1en=0: }
35	for (i=0;i <fm_size;i++) for (i=0;i<wardetergth_no;i++) slot[i][j]="0;</td"><td></td></wardetergth_no;i++)></fm_size;i++) 	

Sta	te 0:	init (CET's)
G	Cond:	(ARRIVAL)
¥	Exec:	
	Trans:	pk_prepare
E	Cond:	(dcfault)
Ŧ	Exec:	
	Trans:	idle

State Discretion Discretion Discretion Discretion State State Discretion Dis	J/L/W/P	/R Client/Server System		0,	/D Cliant/Son/o
State 3. Schedule (File Exces) Orocol, 47 lines 1 (f(ARNAL)) (f(ARNAL)) (f(ARNAL)) (f(ARNAL)) 2 (f(ARNAL)) (f(ARNAL)) (f(ARNAL)) (f(ARNAL)) 2 (f(ARNAL)) (f(ARNAL)) (f(ARNAL)) (f(ARNAL)) 2 (f(ARNAL)) (f(ARNAL)) (f(ARNAL)) (f	chedul	sr Model		D/L/WM	ar Model
State 8. Schedule (Enter Exces) forced, 47 lines State 1 ((a) (a) (a) (a) (a) (b) (a) (a) (b) (b) (a) (b) (b) (b) (b) (b) (b) (b) (b) (b) (b					
International and the second state of the second	Stat	e 3: schedule (Enter Execs)	forced, 47 lines	Stat	e 2:
<pre> * (no.jin.vrc = Nt) * (no.jin.vrc = Nt)</pre>		if (ARRIVAL)			if (!load_steady)
⁵ if (rep. start, picture (hypert.OFC, OFOS, 1ALL)=C(PC, ORS, OK) ¹⁰ (rep. start, picture is netering in the rest in the picture of the detroy (hyper); if (rep. picture is netering in the rest in the picture of the detroy (hyper); if (rep. picture is netering in the rest in the picture picture (in the rest in the picture); if (rep. picture		$i(10_{-}in_{-}svc == Ni)$			temp=total_t
10 ckc [Printlif excret inserting into recording opticity (Print) 13 ckc [Printlif excret inserting into (Print) 14 [entropy and (Print) 15 ckc 16 [equores equication (Print) 17 [equores equication (Print) 18 [equores equication (Print) 19 [equores equication (Print) 10 [equores equication (Print) 11 [equores equication (Print) 11 [equores equication (Print) 11 [equores equication (Print) 12 [equores equication (Print) 13 [equores equication (Print) 14 [equores equication (Print) 15 [equores equication (Print) 16 [equores equication (Print) 17 [equores equication (Print) 16 [equores equication (Print) 17 [equores equication (Print) 16 [equores equication (Print) 17 [equores equication (Print) 18 [equores equication (Print) 19 [equores equication (Print) 11 [equores equication (Print) 11 [equores equication (Print) 11 [equores equication (Print) 11 [equores equication (Pr	°.	t ut (ap_subq_pk_insert(0,pkpt:OPC_QPOS_TAIL):=OPC_QINS_OK)		5	if (temp>0.9 -{
10 improve starts that hatter, increasers and factors that that that that that that that tha		<pre>{ print("error inserting into queue\n"): op_pk_destroy(pkpt clsc if (load_steady)</pre>	(, , , , , , , , , , , , , , , , , , ,		tstarts=1
10 i ((accesserer, not) i (accesserer, not) i (accesserer, not) i (not ever equicative) i (antra-atemp; quice-seried, quice(1)) 15 15 cles i ((auces-equicative) i (autra-atemp; quice-seried, quice(1)) 16 10 if (find_recorer, regiont) i (autra-atemp; quice-seried, quice(1)) 16 20 if (find_recorer, regiont) i (autra-atemp; quice-seried, quice(1)) 20 21 if (find_recorer, right) i (find_recorer,		t ccmp=op_sim_time();			op_intr *
15 0 15 16 (queners-squeer "(ramp-starm), turrum=ramp; quizeu=find, quize(0;;) 20 if (fuld_resource(phen)) if (fuld_resource(phen)) if (fuld_resource(phen)) 25 if (fuld_resource(phen)) 26 if (fuld_resource(phen)) 27 if (fuld_resource(phen)) 28 if (fuld_resource(phen)) 29 if (fuld_resource(phen)) 29 if (fuld_resource(phen)) 20 if (fuld_resource(phen)) 21 if (fuld_resource(phen)) 22 if (fuld_resource(phen)) 23 if (fuld_resource(phen)) 24 if (fuld_resource(phen)) 25 if (fuld_resource(phen)) 26 if (fuld_resource(phen)) 27 if (fuld_resource(phen)) 28 if (fuld_resource(phen)) 29 if (fuld_resource(phen)) 20 if (fuld_resource(phen)) 28 if (fuld_resource(phen)) 29 if (fuld_resource(phen)) 20 if (fuld_resource(phen)) 20 if (fuld_resource(phen)) 20 if (fuld_resource(phen)) 21 if (fuld_resource(phen)) 22 if (fuld_resource(phen)) 23 if (fuld_resource(phen)	10	if (src<=server_no dsi<=server_no) {queues+=qsizes*(temp-tstarts): tstarts=temp: qsizes=find_qsi	ze(1):}	10	-
15 15 15 20 if (full_reserver(plev)) if (full_reserver(plev)) 21 if (full_reserver(plev)) 22 if (full_reserver(plev)) 23 if (full_reserver(plev)) 24 if (full_reserver(plev)) 25 if (full_reserver(plev)) 26 if (full_reserver(plev)) 27 if (full_reserver(plev)) 28 if (full_reserver(plev)) 29 j 20 if (new plevel) 21 if (new plevel) 22 if (new plevel) 23 if (new plevel) 24 if (new plevel) 25 if (new plevel) 26 if (new plevel) 27 if (new plevel) 28 if (new plevel) 29 j 29 j 20 if (new plevel) 20 if (new plevel) 21 if (new plevel) 22 if (new plevel) 23 if (new plevel) 24 if (new plevel) 25 if (new plevel) 26 if (new plevel) 27 if (new plevel) 28 if (new plevel) 29 if (new pleve		cise {queueu+=qsizeu*(temp-tstartu): tstartu=temp: qsizeu=find_q;	ize(0); }		if (MEASURE) {
20 if (findu securciphen) 20 if (no. short, prime conceptent) 21 if (no. short, prime conceptent) 22 if (no. short, prime conceptent) 23 if (no. short, prime conceptent) 24 if (no. short, prime conceptent) 25 if (no. short, prime conceptent) 26 if (no. short, prime conceptent) 27 if (no. short, prime conceptent) 28 if (no. short, prime conceptent) 29 if (no. short, prime conceptent) 20 if (no. short, prime conceptent) 21 if (no. short, prime conceptent) 22 if (no. short, prime conceptent) 23 if (no. short, prime conceptent) 24 if (no. short, prime conceptent) 25 if (no. short, prime conceptent) 26 if (no. short, prime conceptent) 27 if (no. short, prime conceptent) 28 if (no. short, prime conceptent) 29 if (no. short, prime	15	_		15	temp=0p_sin stats[stat_ct]
20 if (no. subt. JR. Insertion into queue (n.), on JR. destroy (piper); 1 20 21 (f (no. d. g. add)) is (no. d. g. add)) 23 (f (no. d. g. add)) 24 (no. d. g. add)) 25 (f (no. d. g. add)) 26 (f (no. d. g. add)) 27 (f (no. d. g. add)) 28 (f (no. d. g. add)) 29 (f (no. d. g. add)) 20 (gueuses=spicas' (temp-start); startu-etemp, qsizes" (ind. gaize(1);) 28 (gueuses=spicas' (temp-start); startu-etemp, qsizes" (ind. gaize(1);) 29 (gueuses=spicas' (temp-start); startu-etemp, qsizes" (ind. gaize(1);) 20 (gueuses=spicas' (temp-start); startu-etemp, qsizes" (ind. gaize(1);) 20 (gueuses=spicas' (temp-start); startu-etemp, qsizes" (ind. gaize(1);) 21 (gueuses=spicas' (temp-start); startu-etemp, qsizes" (ind. gaize(1);) 22 (f (no. in g. et < N))		else if (!find_resource(pkptr))			stat_ct+=1; t if (stat_ct==1
20 cisc (find) second financial finto queneran's (or ph. destroy(perport; 1) 25 (find) second financial finto queneran's (or ph. destroy(perport; 1) 26 (find) second financial finto queneran's (or ph. destroy(perport; 1) 27 (find) second financial finto queneran's (starturatemp; qsize=find, qsize(1); 1) 28 (quenera-sqizes*(temp-startu); tstarturatemp; qsize=find, qsize(1); 1) 29) 20) 21 (quenera-sqizes*(temp-startu); tstarturatemp; qsize=find, qsize(0); 1) 29) 20) 21 (quenera-sqizes*(temp-startu); tstarturatemp; qsize=find, qsize(0); 1) 20 (quenera-sqises*(temp-startu); tstarturatemp; qsize=find, qsize(0); 1) 20) (quenera-sqises*(temp-startu); tstarturatemp; qsize=find, qsize(0); 1) 21 (quenera-sqises*(temp-startu); tstarturatemp; qsize=find, qsize(0); 1) 22 (quenera-sqises*(temp-startu); tstarturatemp; qsize=find, qsize(0); 1) 23 ((final) -secure; tstarturatemp; qsize=find, qsize(1); 1) 24 (final) -secure; 1) 25 (final) -secure; 1) 26 (final) -secure; 1) 27 (final) -secure; 1) 28 (final) -secure; 1) 29 (final) -secure; 1) 20 (final) -secure; 1)	ę	if (op_subq_pk_insert(0,pkpu:OPC_QPOS_TAIL))=OPC_QINS_OK)			{ step2:
25 if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 30 if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 30 if (NEXTO) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 31 if (NEXTO) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 32 if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 33 if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 34 if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 35 if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 36 if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 37 if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 38 if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 38 if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) if (arccesterer_no) 39 if (arccesterer_no) if (arccesterer_no) if (arcce	3	<pre>(print('error inserting into queue\n'); op_pk_destroy(clsc if (load_steady)</pre>	pkptr); }	20	serial_c if (temp
25 if (accessrerno) if (accessrerno) if (accessrerno) 30 if (accessrerno) (queueu+=qsized*(temp-startu): startu=temp; qsizes=find_qsize();) 25 30 if (NEXTQ) (queueu+=qsized*(temp-startu): startu=temp; qsizes=find_qsize();) 36 31 if (NEXTQ) (queueu+=qsized*(temp-startu): startu=temp; qsizeu=find_qsize(0);) 30 31 if (NEXTQ) (queueu+=qsized*(temp-startu): startu=temp; qsizeu=find_qsize(0);) 30 32 if (no_1) (no_1) (no_1) 33 if (no_1) (no_1) (no_1) 34 if (no_1) (no_1) (no_1) 35 if (no_1) (no_1) (no_1) 36 if (no_1) (no_1) (no_1) 37 if (no_1) (no_1) (no_1) 38 if (no_1) (no_1) (no_1) 39 if (no_1) (no_1) (no_1) 30 if (no_1) (no_1) (no_1) 31 if (no_1) (no_1) (no_1) 32 if (no_1) (no_1) (no_1) 33 if (no_1) (no_1) (no_1) 34 if (no_1) (no_1) (no_1) 34 if (no_1) (no_1) (no_1)		{ temp=op_sim_time();			if (temp sten3:
30)) 31 (ueueu+sqizau*(temp staru): tartu=temp; qsicue=find_qsize(0):) 32 (if (NEXTO)) 33 (if (NEXTO)) 1 (if (no.in., sve < N1))	25	if (src<=server_no dst<=server_no) {	asize(1); }	35	serial_c
30))))) (queedor + system ' (temp- isamu): isamu-etemp; qaraci0;) 33 [if (NEXTO) ([r] (no.m., statu-etemp; qaracin, isamu-etemp; qaracen, isamu-etem		clse		1	step4:
30)		{queueu+=qsizeu*(cmp-tstartu): tstartu=temp: qsizeu=tu }	id_qsize(0):}		col(40.6 if (temp
if (NEXTQ) if (NEXTQ) 35 if (NEXTQ) 10 if (no.in_swc <n1)< td=""> if (no.in_swc <n1)< td=""> if (no.in_swc <n1)< td=""> if (no.in_swc <n2)< td=""> if (no.in_swc <n1)< td=""> if (no.in_swc <n2)< td=""> if (no.in_swc <n2)< td=""> if (no.in_swc <n2)< td=""> <td< td=""><td>30</td><td></td><td></td><td>30</td><td>step5: if (n0>n</td></td<></n2)<></n2)<></n2)<></n2)<></n2)<></n2)<></n2)<></n2)<></n2)<></n2)<></n2)<></n2)<></n2)<></n1)<></n2)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<></n1)<>	30			30	step5: if (n0>n
35 Inforego subdy stat(0.0PC_OSTAT_PKSIZE); for (richt)-ranko(r++) 35 10 if (no. insvc < N1)		if (NEXTQ)			else (n0
35 ratio-pandy suff. article. To the process and the process of th					op_intrpt_se
40 if (m, in, src < N1)	35	ratio=op_subq_stat(0.OPC_QSTAT_PKSIZE); for (i=0:i <rationi++)< td=""><td></td><td>35</td><td>-</td></rationi++)<>		35	-
40 if (Ind. resource(betrof) if (Ind. resource(betrof) (g. subt. Jr. resource(betrof) (g. resource(betrof) (g. subt. Jr. resource(betrof) (g.		if $(n_0, n_s v c < N_1)$		3	if (end_sim && !]
40 If (Into: Subject, OPC, OPC, TALL):=OPC, ONS, OK) 41 If (Into: Subject, OPC, OPC, INL):=OPC, ONS, OK) 1 If (Into: Subject, OPC, OPC, INL):=OPC, OPC, OPC, OPC, OPC, OPC, OPC, OPC,		pkptr=op_subq_pk_remove(0.0);			op sim end
State (print(f'error inserting into queue)n*). op. jk_detroy(pkpr);) Siate 15 temp=op.sim_time(): ratio=find_qsize(1); ratio=ind_qsize(s & load stealy) (queue+=qsize*/temp-tstant): tstartu=temp; qsizeu=ratio;) 60 6 15 if (ratio:=qsizes & load_stealy) (queue+=qsizeu*(temp-tstant): tstartu=temp; qsizeu=ratio;) 60 6 16 if (ratio:=qsizes & load_stealy) (queue+=qsizeu*(temp-tstant): tstartu=temp; qsizeu=ratio;) 60 6 16 if (ratio:=qsizes & load_stealy) (queue+=qsizeu*(temp-tstant): tstartu=temp; qsizeu=ratio;) 60 6 17 if (ratio:=qsizes & load_stealy) (queue+=qsizeu*(temp-tstant): tstartu=temp; qsizeu=ratio;) 60 6 16 if (ratio:=qsizes & load_stealy) (queue+=qsizeu*(temp-tstant): tstartu=temp; qsizeu=ratio;) 60 6 17 if (ratio:=qsizeu & load_stealy) (queue+=qsizeu*(temp-tstant): tstartu=temp; qsizeu=ratio;) 60 6 17 if (ratio:=qsizeu & load_steal) (queue+=qsizeu*(temp-tstant): tstartu=temp; qsizeu=ratio;) 60 6 17 if (ratio:=qsizeu & load_steal) (qsizeu=ratio;) 60 6 6 17 if (ratio:=qsizeu & load_steal) (qsizeu=ratio;) 60 6 6 17 if (ratio:=qsizeu & load_steal) (qsizeu=ratio;) 6 6 6 17 if (ratio:=qsizeu & load_steal) (qsizeu=ratio;) 6 6	40	if (!tind_resource(pkptr.)) if (op_subq_pk_insert(0,pkptr.OPC_QPOS_TAIL)!=OPC_QIN	S_OK)		
45 temp-eop. sim_time(): ratio=find_size(1); ratio=find_sizes & k load steady) (queuev+syster*(temp-tstant): vtant=temp; qsizeu=ratio;) 40 80 45 ratio=find_size(s) (queuev+syster*(temp-tstant): tstant=temp; qsizeu=ratio;) 60 6 45 ratio=find_size(s) (queuev+syster*(temp-tstant): tstant=temp; qsizeu=ratio;) 60 6 45 ratio=find_size(s) (queuev+syster*(temp-tstant): tstant=temp; qsizeu=ratio;) 60 6 45 ratio=find_size(s) 6 6 7 46 ratio=find_size(s) 7 6 7 47 ratio=find_size(s) 7 7 7 48 ratio=find_size(s) 7 7 7 49 ratio=find_size(s) 7 7 7 41 ratio=find_size(s) 7 7 7 42 ratio=find_size(s) 7 7 7 43 ratio=find_size(s) 7 7 7 44 ratio=find_size(s) 7 7 7 45 ratio=find_size(s) 7 7 7 46 ratio=find_size(s) 7 7 7 47 ratio=find_size(s) 7 7 7 48 ratio=find_si		{ print((*error inserting into queue\n*). op_pk .	destroy(pkpu): }	Stat	e 2:
15 Interestington Interestington Interestington 14 Interestington Interestington Interestington 15 Interestington Interestington Interestington 16 Interestington Interestington Interestington 17 Interestington Interestington Interestington 16 Interestington Interestington Interestington 17 Interestington Interestington Interestington 17 Interestington Interestington Interestington 17 Interestington Interestington Interestington 17 Interestington Interestington Interestington		lcmp=op_in_time(); ratio=find_qsize(1);		¥ 5	Cond: Exec:
ii (ratio:=qsizeu && load_steady) [queveu+=qsizeu*(temp-tstartu): tstartu=temp; qsizeu=ratio;] ii (ratio:=qsizeu && load_steady) [queveu+=qsizeu*(temp-tstartu): tstartu=temp; qsizeu=ratio;] ii (ratio:=qsizeu && load_steady) [queveu+=qsizeu*(temp-tstartu): tstartu=temp; qsizeu=ratio;] ii (ratio:=qsizeu && load_steady) [queveu+=qsizeu*(temp-tstartu): tstartu=temp; qsizeu=ratio;] ii (ratio:=qsizeu && load_steady) [queveu+=qsizeu*(temp-tstartu): tstartu=temp; qsizeu=ratio;] ii (ratio:=qsizeu && load_steady) [queveu+=qsizeu*(temp-tstartu): tstartu=temp; qsizeu=ratio;] ii (ratio:=qsizeu && load_steady) [queveu+=qsizeu*(temp-tstartu): tstartu=temp; qsizeu=ratio;] ii (ratio:=qsizeu && load_steady) [queveu+=qsizeu*(temp-tstartu): tstartu=temp; qsizeu=ratio;] ii (ratio:=qsizeu && load_steady) [queveu+=qsizeu*(temp-tstartu): tstartu=temp; qsizeu=ratio;] ii (ratio:=qsizeu && load_steady) [queveu+=qsizeu*(temp-tstartu): tstartu=temp; qsizeu*(temp-tstartu): tstart	45	u utauo:=4sizes & & ioau_sicauyo {queues+=9sizes=7riemp-1siartsi: tstarts=1em ratio=find_gsize(0);	ال. فديدهحعديون }	120	Trans: Cond:
CET Cond: (1) CET Cond: (1) Face: (1) Fac		if (ratio!=qsizeu && load_steady) {queueu+=qsizeu*(temp-tstartu); tstartu=te	mp: qsizeu=ratio:}	,≞	Exec:
State 3: schedule (CET's) #2 CET Cond: (1) #3 M Trans. #4 #4				130	Trans: Cond:
State 3: schedule (CET's) CET Cond: (1) #3 th Trans. dia Trans. dia				1 2	Exec:
CET Cond: (1) #3 [6]	Stat	e 3: schedule (CET's)		CET	Cond:
	#0ET	Cond: (1) Exec: :		#3	Exec: Trans:
		Trans: idle]	1.101.00

Process Model Report: mm1qr_cs_scheduler Fri Jan 7 02:35:19 1994 Page 5 of 12 MMV/1/DIR Client/Server System scheduler Model

if ('load_stead	
temp=tota if (temp>(L inne*slot, no(/N* op_sim_time ()*load_prev): 1999 && tempe1.001)
tstart tstart queu op_ii	⊨op. sim. time(): tstatu=op. sim. time(): qsizes=find_ qsize(1). qsizeu=find_ qsize(0). s=dt: queueu=0: load_ steady=1: stat_ct=0: next_stat=800: atrpt_schedule_setf(op_sim_time()+inc.0):
if (MEASURE	
temp=0p_ stats[stat_ stat_ct+=1 if (stat_ct=	sim_time(): quecest-apixos*(temp-tistaris): quecest-apixos*(temp-tistariu): stl=queceston: stantajata.cpl=queceutine: queues=di; quecueu=di; : tatars=temp: tistaru=temp: qsizes=find_qsize(1): qsizeu=find_qsize(0): ==text_cast)
f step2	
seria if (ter	_col(400.&temp); nn>0.4) eoin sten5;
if (te.	npc()) golo step4:
scrial	
if (ra stend	io>temp) goto step5.
col(4	Actemp):
if (te sten5	np<0.1) end_sim=1:
if (n6	onl) { n1=n1*2: next_stat=n1; }
- cisc	nu=nu*2: next_stat≡nu: }
op_intrpt }	
if (end_sim & å op sim e	.iENDSIM) M("reaching steady state)

Sta	te 2:	idle (CET's)
5	Cond:	(ARRIVAL)
£	Exec:	
	Trans:	pk_prepare
E	Cond:	(default)
Ŧ	Exec:	
	Trans:	idle
E	Cond:	(ENDSIM)
ţ	Exec:	record_stats();
	Trans:	Idle
E	Cond:	(NEXTQ && !ARRIVAL)
£3	Exec:	
	Trans:	schedule
Í		

352 lines

Function Block

Proces	ss Model Report: mm1qr	_cs_scheduler	Fri Jan 7 02:35:20 1994	Page 7 of 12
M/M/1.	/Q/R Client/Server Syste Jer Model	Ε		
	record_stats()			
<u>.</u>	{ int double extern int extern double cxtern channel_asgn	i,j.b: x,x,xu,s,sssu,y[40],ys[40],yu[40]; stot[10][10]; time: pkt[1025];		
10	$s=(double)stat_ct/(donb=floor(s):x=0; xs=0; xu=0;for (=0:i<40:i++)$	ubie)40.		
15	(i)=0: ys(i)=0:) for (j=0; ys(i)=0:) y(i)=y(i)/b: ys(i)	u(i)=0: . {ys[i]+=stats[i*b+j]; yu(i]+=statu[i*b+j]; y =ys[i]/b; yu[i]=yu[i]/b; x+=y[i]; xs+=ys[i]; y	(ij=ys(i)+yu(i):) xu+=yu(i):	
	<pre>uxf.d0. xs=xs440; xu= for (i=0):=400; i=1) [j= s=s239; su=sy39; su== if (x!=0) s=z*sqn(s) if (xs1=0) ss=z*sqn(s) if (xu)=0) su=z*sqn(s)</pre>	xu/t0, s=0, su=0; y(i)-x: s+=j*j: j=ys(i)-x: ss+=j*j: j=yu(i)-x: yu/3> yu/0)/x: u/40)/xu:	u: su+=j*j;) بر	
25	<pre>for (i=1; i<=Nl; i++) if (pkt[i].src!=0)</pre>	time+=(op_sim_time()-pkt[i].ctime);		
30	op_stat_write_scalar op_stat_write_scalar op_stat_write_scalar op_stat_write_scalar op_stat_write_scalar	1.0ad°.total_time*stot_not(N*op_sim_tim -cu*time*stot_not(N*op_sim_time()); - r. (double/botal_pkst(double/total_pkst(-0.*x); -D*(doalys+delayu/tdouble/tprocessed_pk	e())): Nal_pku)): :s+processed_pku)):	
35	op. stat. write. scatart op. stat. write. scatart op. stat. write. scatart op. stat. write. scatart op. stat. write. scatart	°os *xs; 10s *delays(double)processed_pks); •ou *xu); •ou *delayu/(double)processed_pku); •od *s);		
40	op_stat_write_scalar op_stat_write_scalar op_stat_write_scalar	qas. ، دهر. (راه. ' دهر. - د ' ۱۹۹۰ رام.		
45	f_inverse (x.zl) double x,*zl; { double y,p,p0,p1,p2	ւթ3,թ4,զ0,գ1,գ2,գ3,գ4։		
20	x=1-x/2; p0=-0.32232431088, q0=0.099348462606e	e0. p1=-1.0. p2=-0.342242088547e0. p3=-0. 0. q1=0.588581570495e0. q2=0.531103462	(0204231210245e0; p4=-0.45364221) :366e0: q3=0.10353775285e0; q4=0.3	0148e-4; 8560700634e-2;
	p=x: if (p>0.5) p=1.0-p; y=sqrt(-log(p*p)); *2=j+*(0^4+y*(p1+y* if (x<0.5) *21=*21; }	(p2+y*(p3+y*p4)))/(q0+y*(q1+y*(q2+y*(q ³ +y*q4)));	



Proc	Cess	Model Report: mm1gr_cs_scheduler	Fri Jan 7 02:35:21 1994	Page 9 of 12
MM	V1/Q/	VR Client/Server System		
8				
	120	int i.j.b: double p.x.s.y[400]:		
		b=next_stat/n; x=0; for (j=0;i <n;i++)< td=""><td></td><td></td></n;i++)<>		
	125	√[]=0: for (]=0j±(2;++) y[i]+=(sus[i ⁺ t+j]+suu[i ⁺ t+, y[i]=y[i]b: x+=y[i]:	*	
	130	<pre>x=Vn: sel; for (i=0):cni+i) (j=y(i)-x: st=j*j;) ss(0-1): sequ(00:1): sequ(00:1): fit (s=0) *p1=0; clas *p1=2*x; primt(*st=*b1*b1*clas *p1=2*x;</pre>		
	135	_		
		int find_resource(packet) Packet* packet:		
	140	t int ij,k.ap.apl.ndc.t.oh.go_back.waste int conflict[130][2],empty[10],store[10 double svc:	_ct.fit.next.sc.dt.p; 5][4],temp[130].asgn[130].mb[130].tmp_n	mb[130];
	145	extern int no. in svc.sio(130][10]: extern other_asgn other[1025][15]: extern channel_asgn pkt[1025];		
		0p_pk_nfd_get(packet. * type * &tp): 0p_pk_nfd_get(pack 0p_pk_nfd_get(packet. * dst * & &d): 0p_pk_nfd_get(packet	ຸ" ຣາຕ - ເ ຮັນ ເ); ຣາບຕ - ເ ຮັນ ເງ;	
-	150	ct=0; for (j=0; j <wavelength_no: empty(j]="0;</td" j++)=""><td></td><td></td></wavelength_no:>		
	155	for (i=0:i <fm_size:i++) td="" {="" {<=""><td></td><td></td></fm_size:i++)>		
	3	<pre>conflict[i][0]=-1: conflict[i][1]=-1; for (k=0: k<wavelength_no: k++)<="" pre=""></wavelength_no:></pre>		
	99	if (slotij k =0, emptyk +=1; if (pd(sloti) k) src=ss) conflicti 0 =k; if (pd(sloti) k]dst=dt) conflicti 1 =k; }		
	165	وی hack=0; if (conflict[j][0]=-1 && conflict[j][]=-1) در مرمد است		
	170	co (κ. κ. κ. κ. κ. κ. κ. κ. (κ. κ. γ. κ. κ. γ. γ. γ. γ. κ. κ. γ.	kþó)) empylki:	
	175	use if ((conflict[j]]0]=-/f[conflict[j]]0] clsc ch=2: if (oh=2 && empty[k]=2) { empty[k] insert:	=k)&&(conflict[j][1]==-1lkonflict[j][1]==k) 3: continue: }	() oh=0:



Proces	s Model Report: mm1ar cs scheduler [Fri Jan 7 02:35:22 1994	Page 11 of 12
M/M/1/ schedu	O/R Client/Server System lier Model	
	j=store[1]21-fit:	
240	if (next) { if ()	
245	<pre>k=j-ad+fit: k=j-ad+fit: if (fic= apl && k-ap)</pre>	
255	ap-k; ap1=fit; temp[i, ct]=: for (ab=0; ohcfm_size; oh++) tmp_mb[oh]=0; for (ab=0; ohcfm_size; oh++) tmp_mb[oh]=0; k=shct=strot[i][0]+:store[i][2]+1; if (go_back>fm_size) go_hack=fm_size; for (oh=k; ohcgo_back; oh++) tmp_mb[oh]=1.	
	۲ if (; 5-nd ll i==(c+1)) wask+==p1: L ct+=1: if (wask<=a@p1(0) ll agg(10)=-1)	
265	agn(0)=wate: aspn[1]=i_ct: for (h=2; kc(i_ct+2); kt+1) aspn[k]=temp[k-2], f(aspn[0]=0) guo assign: f(aspn[0]=0) guo assign:	
72	 if (1,stud) if (1,stud) temp(0 =i: ad=slot_no;); waste=fit: i_tt=1: next=0; temp(0 =i: ad=slot_no;); waste=fit: i_tt=1: next=0; texture[1](0 -1: i: f(1,std)); h=0; texture[1](0 -1: i: f(1,std)); h=0;<td></td>	
27.	continue:	
28	<pre>k=0; for (ob=sore[i][0]; ob<(sore[i][0]+sore[i][2]); oh++) if (mb[ob]) [k=1; break; } if (k=1) contine; clea</pre>	
58	<pre>t temp[.ct]=i: (k<0) k=0; k=0; k=2; k=store[1][0]-1; if (k<0) k=4; k=store[1][0]-1; if (k<0) k=4; k=0; k=0; k=0; k=0; k=0; k=0; k=0; k=0</pre>	
	<pre>1 if (j=nd) if (j=nd) if (d=j; wastet=fit i_ct+=1: continue;) if (j=nd)</pre>	
29.		



- [1] R. Ramaswami, "Multiwavelength Lightwave Networks for Computer Communications", IEEE Communications Magazine, pp. 78-88, Feb 1993
- [2] B. Mukherjee, "WDM-Based Local Lightwave Networks Part I: Single-Hop Systems," *IEEE Network*, pp. 12-27, May 1992
- [3] B. Mukherjee, "WDM-Based Local Lightwave Networks Part II: Multihop Systems," *IEEE Network*, pp. 20-32, July 1992
- [4] S.B. Alesander, R.S. Bondurant, D. Byrne, V.W.S. Chan, S. Finn, R. Gallager, R.S. Kennedy, et al, "A Consortium on Wideband All-Optical Networks," to be published in *Journal of Lightwave Technology*
- [5] J.F. Shoch, J.A. Hupp, "Measured Performance of an Ethernet Local Network", *Communications of the ACM*, pp. 711-721, Dec 1980
- [6] OPNET Manuals by *Mil 3, Inc.*
- [7] D. Bertsekas, R. Gallager, "Data Networks", Prentice-Hall, Inc. NJ, p.200, 1987
- [8] R. Jain, "The Art of Computer Systems Performance Analysis", John Wiley & Sons, pp. 432-436, 1991
- [9] M.A. Crane, A.J. Lemoine, "An Introduction to the Regenerative Method for Simulation Analysis", *Springer-Verlag*, pp. 36-46, 1977
- [10] P. Bratley, B.L. Fox, L.E. Schrage, "A Guide to Simulation", Springer-Verlag, p. 327, 1983
- [11] P. Bratley, B.L. Fox, L.E. Schrage, "A Guide to Simulation", Springer-Verlag, pp. 86-87, 1983