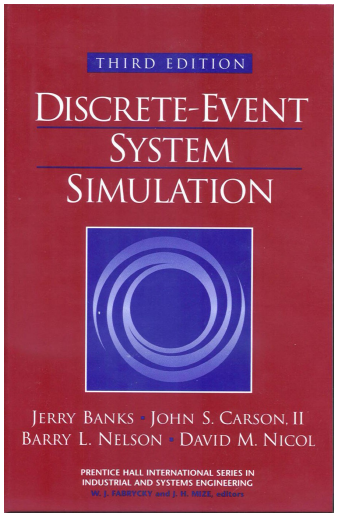
Computer Systems Lab.

http://tolerance.ajou.ac.kr

DISCRETE-EVENT SYSTEM SIMULATION



Third Edition

Jerry Banks • John S. Carson II
Barry L. Nelson • David M. Nicol

2.2 Simulation of Inventory Systems (1)

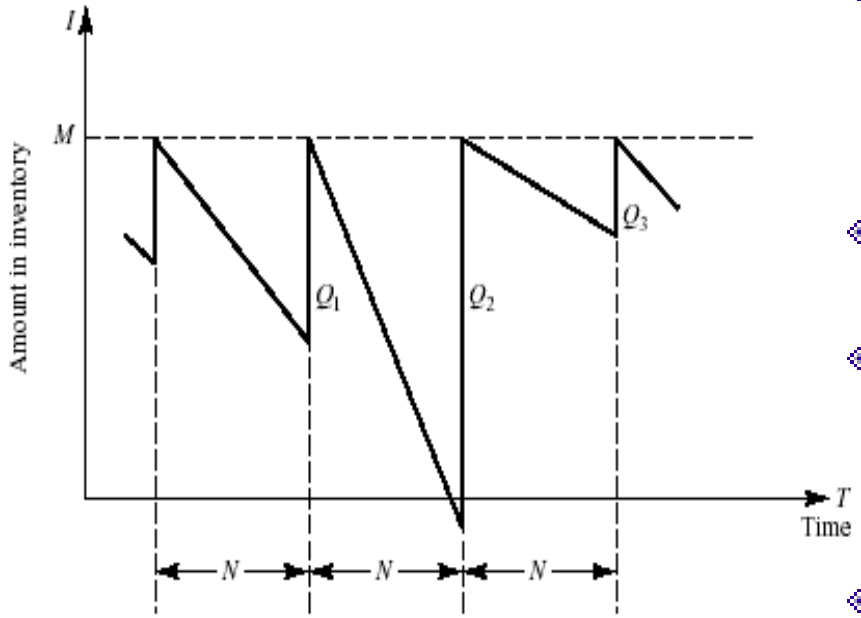


Figure 2.7 Probabilistic order-level inventory system.

- This inventory system has a periodic review of length N, at which time the inventory level is checked.
- An order is made to bring the inventory up to the level M.
- In this inventory system the lead time (i.e., the length of time between the placement and receipt of an order) is zero.
- Demand is shown as being uniform over the time period

2.2 Simulation of Inventory Systems (2)

- Notice that in the second cycle, the amount in inventory drops below zero, indicating a shortage.
- Two way to avoid shortages
  - Carrying stock in inventory

    : cost - the interest paid on the funds borrowed to buy the items, renting of storage space, hiring guards, and so on.
  - Making more frequent reviews, and consequently, more frequent purchases or replenishments

    : the ordering cost
- The total cost of an inventory system is the measure of performance.
  - The decision maker can control the maximum inventory level, M, and the length of the cycle, N.
  - In an (M,N) inventory system, the events that may occur are: the demand for items in the inventory, the review of the inventory position, and the receipt of an order at the end of each review period.

2.2 Simulation of Inventory Systems (3)

- Example 2.3 The Newspaper Seller's Problem
  - A classical inventory problem concerns the purchase and sale of newspapers.
  - The paper seller buys the papers for 33 cents each and sells them for 50 cents each. (The lost profit from excess demand is 17 cents for each paper demanded that could not be provided.)
  - Newspapers not sold at the end of the day are sold as scrap for 5 cents each. (the salvage value of scrap papers)
  - Newspapers can be purchased in bundles of 10. Thus, the paper seller can buy 50, 60, and so on.
  - There are three types of newsdays, "good," "fair," and "poor," with probabilities of 0.35, 0.45, and 0.20, respectively.

2.2 Simulation of Inventory Systems (4)

- Example 2.3 (Cont.)
  - The problem is to determine the optimal number of papers the newspaper seller should purchase.
  - This will be accomplished by simulating demands for 20 days and recording profits from sales each day.
  - The profits are given by the following relationship:

$$Pofit = \left[\begin{pmatrix} revenue \\ from\ sales \end{pmatrix} - \begin{pmatrix} \text{cost of} \\ newspapers \end{pmatrix} - \begin{pmatrix} lost\ profit\ from \\ excess\ demand \end{pmatrix} + \begin{pmatrix} salvage\ from\ sale \\ of\ scrap\ papers \end{pmatrix}\right]$$

  - The distribution of papers demanded on each of these days is given in Table 2.15.
  - Tables 2.16 and 2.17 provide the random-digit assignments for the types of newsdays and the demands for those newsdays.

2.2 Simulation of Inventory Systems (5)

Computer Systems Lab.

http://tolerance.ajou.ac.kr

Table 2.15 Distribution of Newspapers Demanded

	Demand Probability Distribution		
Demand	*Good*	*Fair*	*Poor*
40	0.03	0.10	0.44
50	0.05	0.18	0.22
60	0.15	0.40	0.16
70	0.20	0.20	0.12
80	0.35	0.08	0.06
90	0.15	0.04	0.00
100	0.07	0.00	0.00

Table 2.16 Random-Digit Assignment for Type of Newsday

Type of Newsday	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
Good	0.35	0.35	01–35
Fair	0.45	0.80	36–80
Poor	0.20	1.00	81–00

Table 2.17 Random-Digit Assignments for Newspapers Demanded

	Cumulative Distribution			*Random-Digit Assignment*		
Demand	*Good*	*Fair*	*Poor*	*Good*	*Fair*	*Poor*
40	0.03	0.10	0.44	01–03	01–10	01–44
50	0.08	0.28	0.66	04–08	11–28	45–66
60	0.23	0.68	0.82	09–23	29–68	67–82
70	0.43	0.88	0.94	24–43	69–88	83–94
80	0.78	0.96	1.00	44–78	89–96	95–00
90	0.93	1.00	1.00	79–93	97–00	
100	1.00	1.00	1.00	94–00		

2.2 Simulation of Inventory Systems (6)

- Example 2.3 (Cont.)
  - The simulation table for the decision to purchase 70 newspapers is shown in Table 2.18.
  - The profit for the first day is determined as follows:

    Profit = $30.00 - $23.10 - 0 + $.50 = $7.40
    - On day 1 the demand is for 60 newspapers. The revenue from the sale of 60 newspapers is $30.00.
    - Ten newspapers are left over at the end of the day.
    - The salvage value at 5 cents each is 50 cents.
  - The profit for the 20-day period is the sum of the daily profits, $174.90. It can also be computed from the totals for the 20 days of the simulation as follows:
  - Total profit = $645.00 - $462.00 - $13.60 + $5.50 = $174.90
  - The policy (number of newspapers purchased) is changed to other values and the simulation repeated until the best value is found.

Table 2.18 Simulation Table for Purchase of 70 Newspapers

Day	*Random Digits for Type of Newsday*	*Type of Newsday*	*Random Digits for Demand*	*Demand*	*Revenue from Sales*	*Lost Profit from Excess Demand*	*Salvage from Sale of Scrap*	*Daily Profit*
1	94	Poor	80	60	$30.00	–	$0.50	$7.40
2	77	Fair	20	50	25.00	–	1.00	2.90
3	49	Fair	15	50	25.00	–	1.00	2.90
4	45	Fair	88	70	35.00	–	–	11.90
5	43	Fair	98	90	35.00	$3.40	–	8.50
6	32	Good	65	80	35.00	1.70	–	10.20
7	49	Fair	86	70	35.00	–	–	11.90
8	00	Poor	73	60	30.00	–	0.50	7.40
9	16	Good	24	70	35.00	–	–	11.90
10	24	Good	60	80	35.00	1.70	–	10.20
11	31	Good	60	80	35.00	1.70	–	10.20
12	14	Good	29	70	35.00	–	–	11.90
13	41	Fair	18	50	25.00	–	1.00	2.90
14	61	Fair	90	80	35.00	1.70	–	10.20
15	85	Poor	93	70	35.00	–	–	11.90
16	08	Good	73	80	35.00	1.70	–	10.20
17	15	Good	21	60	30.00	–	0.50	7.40
18	97	Poor	45	50	25.00	–	1.00	2.90
19	52	Fair	76	70	35.00	–	–	11.90
20	78	Fair	96	80	35.00	1.70	–	10.20
					$645.00	$13.60	$5.50	$174.90

2.2 Simulation of Inventory Systems (7)

- Example 2.4 Simulation of an (M,N) Inventory System
  - This example follows the pattern of the probabilistic order-level inventory system shown in Figure 2.7.
  - Suppose that the maximum inventory level, M, is11 units and the review period, N, is 5 days. The problem is to estimate, by simulation, the average ending units in inventory and the number of days when a shortage condition occurs.
  - The distribution of the number of units demanded per day is shown in Table 2.19.
  - In this example, lead time is a random variable, as shown in Table 2.20.
  - Assume that orders are placed at the close of business and are received for inventory at the beginning of business as determined by the lead time.

2.2 Simulation of Inventory Systems (8)

- Example 2.4 (Cont.)
  - For purposes of this example, only five cycles will be shown.
  - The random-digit assignments for daily demand and lead time are shown in the rightmost columns of Tables 2.19 and 2.20.

Table 2.19 Random-Digit Assignments for Daily Demand

Demand	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
0	0.10	0.10	01–10
1	0.25	0.35	11–35
2	0.35	0.70	36–70
3	0.21	0.91	71–91
4	0.09	1.00	92–00

Table 2.20 Random-Digit Assignments for Lead Time

Lead Time (Days)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
1	0.6	0.6	1–6
2	0.3	0.9	7–9
3	0.1	1.0	0

Table 2.21 Simulation Tables for (M, N) Inventory System

Cycle	Day	Beginning Inventory	Random Digits for Demand	Demand	Ending Inventory	Shortage Quantity	Order Quantity	Random Digits for Lead Time	Days until Order Arrives
1	1	3	24	1	2	0	–	–	1
	2	2	35	1	1	0	–	–	0
	3	9	65	2	7	0	–	–	–
	4	7	81	3	4	0	–	–	–
	5	4	54	2	2	0	9	5	1
2	1	2	03	0	2	0	–	–	0
	2	11	87	3	8	0	–	–	–
	3	8	27	1	7	0	–	–	–
	4	7	73	3	4	0	–	–	–
	5	4	70	2	2	0	9	0	3
3	1	2	47	2	0	0	–	–	2
	2	0	45	2	0	2	–	–	1
	3	0	48	2	0	4	–	–	0
	4	9	17	1	4	0	–	–	–
	5	4	09	0	4	0	7	3	1
4	1	4	42	2	2	0	–	–	0
	2	9	87	3	6	0	–	–	–
	3	6	26	1	5	0	–	–	–
	4	5	36	2	3	0	–	–	–
	5	3	40	2	1	0	10	4	1
5	1	1	07	0	1	0	–	–	0
	2	11	63	2	9	0	–	–	–
	3	9	19	1	8	0	–	–	–
	4	8	88	3	5	0	–	–	–
	5	5	94	4	1	0	10	8	2
					88				

2.2 Simulation of Inventory Systems (9)

Example 2.4 (Cont.)

- The simulation has been started with the inventory level at 3 units and an order of 8 units scheduled to arrive in 2 days' time.

Beginning Inventory of Third day	=	Ending Inventory of 2 day in first cycle	+	new order

- The lead time for this order was 1 day.
- Notice that the beginning inventory on the second day of the third cycle was zero. An order for 2 units on that day led to a shortage condition. The units were backordered on that day and the next day also. On the morning of day 4 of cycle 3 there was a beginning inventory of 9 units. The 4 units that were backordered and the 1 unit demanded that day reduced the ending inventory to 4 units.
- Based on five cycles of simulation, the average ending inventory is approximately 3.5 (88 ÷ 25) units. On 2 of 25 days a shortage condition existed.

Chapter. 3 Selection of Techniques and Metrics

3.1 Selecting an Evaluation Technique (1)

Table 3.1 Criteria for Selecting an Evaluation Technique

Criterion	Analytical Modeling	Simulation	Measurement
1. Stage	Any	Any	Postprototype
2. Time Required	Small	Medium	Varies
3. Tools	Analysts	Computer language	Instrumentation
4. Accuracy	Low	Moderate	Varies
5. Trade-off evaluation	Easy	Moderate	Difficult
6. Cost	Small	Medium	High
7. Saleability	Low	Medium	High

3.1 Selecting an Evaluation Technique (2)

- Life-cycle stage
  - Measurement : only if something similar to the proposed system already exists
  - Analytical modeling and Simulation : if it is a new concept
- The time available for evaluation
  - Measurements generally take longer than analytical modeling but shorter than simulations.
- The availability of tools
  - Modeling skills, Simulation languages, and Measurement instruments

3.1 Selecting an Evaluation Technique (3)

- Level of accuracy
  - Analytical modeling requires so many simplifications and assumptions that if the results turn out be accurate.
  - Simulations can incorporate more details and require less assumptions than analytical modeling, and thus more often are closer to reality.
  - Measurements may not give accurate results simply because many of the environmental parameters, such as system configuration, type of workload, and time of the measurement, may be unique to the experiment. Thus, the accuracy of results can vary from very high to none.

3.1 Selecting an Evaluation Technique (4)

- Trade-off evaluation
  - The goal of every performance study is either to compare different alternatives or to find the optimal parameter value.
  - Analytical models provide the best insight into the effects of various parameters and their interactions.
  - With simulations, it may be possible to search the space of parameter values for the optimal combination, but often it is not clear what the trade-off is among different parameters.
  - Measurement is the least desirable technique in this respect. It is not easy to tell if the improved performance is a result of some random change in environment or due to the particular parameter setting.

3.1 Selecting an Evaluation Technique (5)

- Cost
  - Measurement requires real equipment, instruments, and time. It is the most costly of the three techniques.
  - Cost, along with the ease of being able to change configurations, is often the reason for developing simulations for expensive systems.
  - Analytical modeling requires only paper and pencils. Thus, It is the cheapest alternative.
- Saleability of results
  - The key justification when considering the expense and the labor of measurements
  - Most people are skeptical of analytical results simply because they do not understand the technique or the final result.

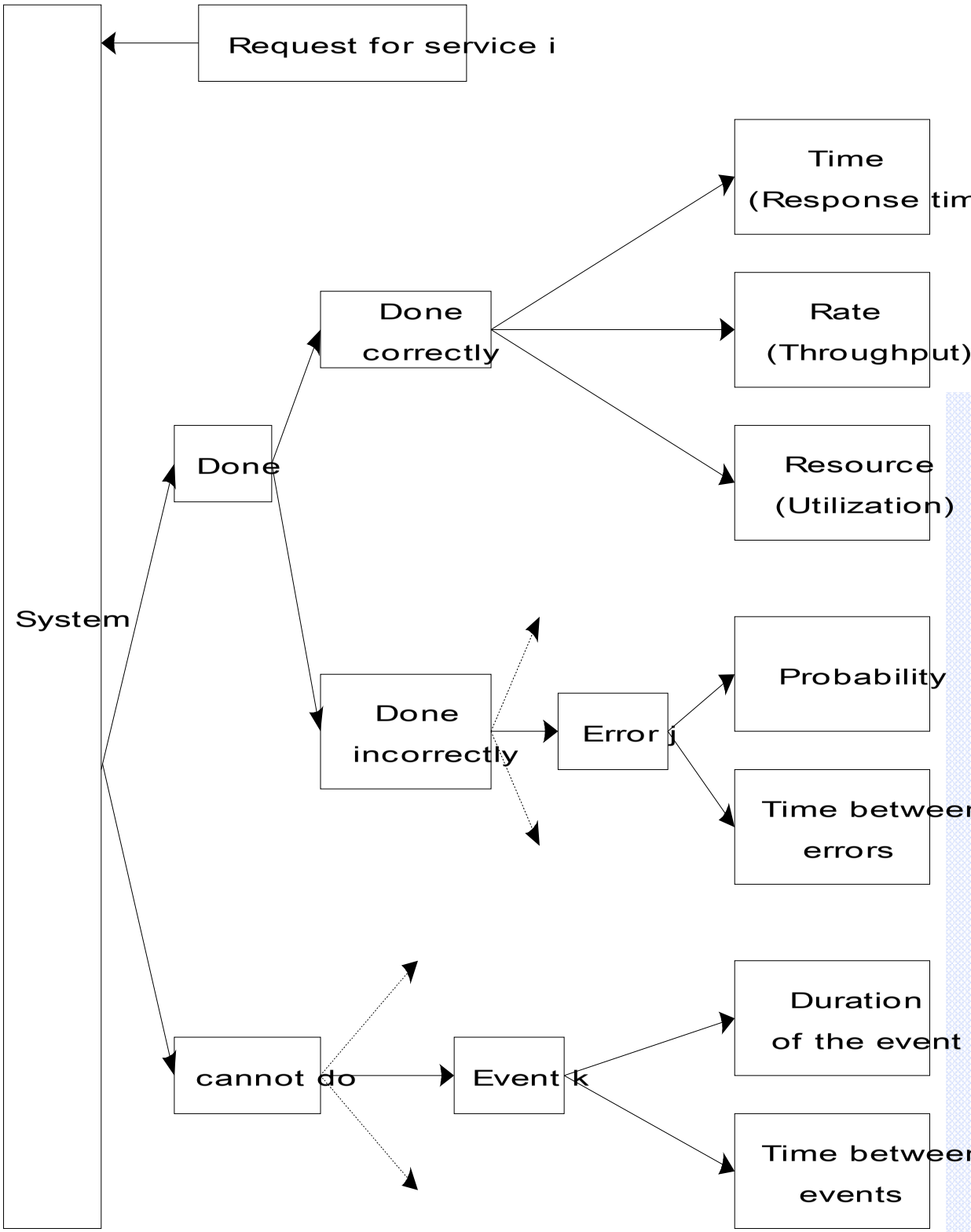
3.1 Selecting an Evaluation Technique (6)

- Three rules of validation
  - Do not trust the results of a simulation model until they have been validated by analytical modeling or measurements.
  - Do not trust the results of an analytical model until they have been validated by a simulation model or measurements.
  - Do not trust the results of a measurement until they have been validated by simulation or analytical modeling.
- Two or more techniques can also be used sequentially or simultaneously.
  - For example, a simple analytical model was used to find the appropriate range for system parameters and a simulation was used later to study the performance in that range.

3.2 Selecting performance Metrics (1)

- One way to prepare a set of performance criteria or metrics
  : to list the services offered by the system

- The outcomes can be classified into three categories, as shown in Figure 3.1.
  : The system may perform the service correctly, incorrectly, or refuse to perform the service.


Request for service i
System
Done
Done correctly
Time (Response tim
Rate (Throughput)
Resource (Utilization)
Done incorrectly
Error j
Probability
Time betwee errors
cannot do
Event k
Duration of the event
Time betwee events

3.2 Selecting performance Metrics (2)

- If the system performs the service correctly
  - Performance is measured by time-rate-resources. (responsiveness, productivity, and utilization)
  - The responsiveness of a network gateway
    : response time (the time interval between arrival of a packet and its successful delivery)
  - The gateway's productivity
    : throughput (the number of packets forwarded per unit of time)
  - The utilization gives an indication of the percentage of time the resources of the gateway are busy for the given load level.
    - The resource with the highest utilization is called the bottleneck.

3.2 Selecting performance Metrics (3)

- If the system performs the service incorrectly
  - An error is said to have occurred.
  - Classify errors and to determine the probabilities of each class of errors. Ex) the probability of single-bit errors for the gateway
- If the system does not perform the service
  - It is said to be down, failed, or unavailable
  - Classify the failure modes and to determine the probabilities of each class. Ex) The gateway may be unavailable 0.01% of the time due to processor failure and 0.03% due to software failure.

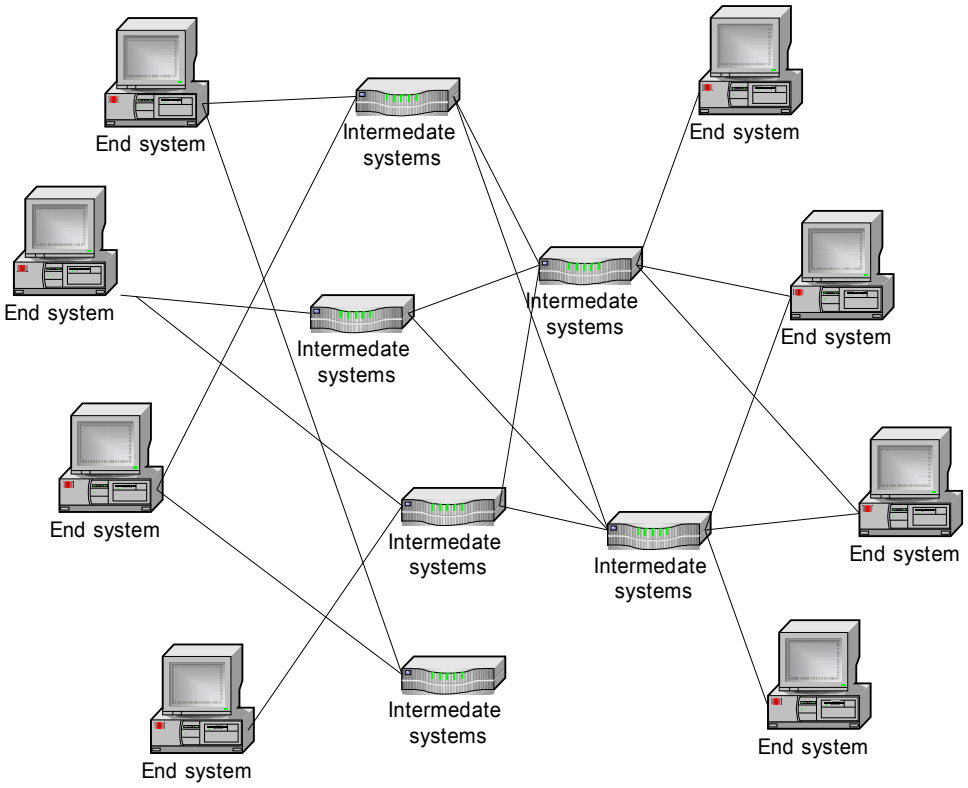
3.2 Selecting performance Metrics (4)

- The metrics associated with the three outcomes, namely successful service, error, and unavailability, are so called speed, reliability, and availability.
- For many metrics, the mean value is all that is important. However, do not overlook the effect of variability.
- In computer systems shared by many users, two types of performance metrics need to be considered : individual and global.
- Individual metrics reflect the utility of each user
  - Response time and Throughput
- Global metrics reflect the systemwide utility.
  - Response time and Throughput
  - Resource utilization, Reliability, and Availability

3.2 Selecting performance Metrics (5)

- Given a number of metrics, use the following considerations to select a subset: low variability, nonredundancy, and completeness.
- Low variability helps reduce the number of repetitions required to obtain a given level of statistical confidence.
- If two metrics give essentially the same information, it is less confusing to study only one.
- The set of metrics included in the study should be complete. All possible outcomes should be reflected in the set of performance metrics.

Case Study 3.1 (1)



- Consider the problem of comparing two different congestion control algorithms for computer networks.
- The problem of congestion occurs when the number of packets waiting at an intermediate system exceed the system's buffering capacity and some of the packets have to be dropped.

Case Study 3.1 (2)

- Four possible outcomes
  - Some packets are delivered in order to the correct destination.
  - Some packets are delivered out of order to the destination.
  - Some packets are delivered more than once to the destination (duplicate packets).
  - Some packets are dropped on the way (lost packets).
- Time-rate-resource metrics
  - Response time: the delay inside the network for individual packets.
  - Throughput: the number of packets per unit of time.
  - Processor time per packet on the source end system.
  - Processor time per packet on the destination end systems.
  - Processor time per packet on the intermediate systems.

Case Study 3.1 (3)

- The variability of the response time is important since a highly variant response results in unnecessary retransmissions. Thus, the variance of the response time became the sixth metric.
- In many systems, the out-of-order packets are discarded at the destination end systems. In others, they are stored in system buffers awaiting arrival of intervening packets. Thus, the probability of out-of-order arrivals was the seventh metric.
- Duplicate packets consume the network resources without any use. The probability of duplicate packets was the eighth metric.
- Lost packets are undesirable for obvious reasons. The probability of lost packets is the ninth metric.
- Excessive losses could cause some user connections to be broken prematurely. The probability of disconnect is the tenth metric.

Case Study 3.1 (4)

- It is necessary that all users be treated fairly in the network. Thus, fairness was added as the eleventh metric. It is defined as a function of variability of throughput across users.
- For any given set of user throughputs (x_1,x_2, … ,x_n), the following function can be used to assign a fairness index to the set:

$$f(x_1, x_2, \cdots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n\sum_{i=1}^{n} x_i^2}$$

- For all nonnegative values of x_i's, the fairness index always lies between 0 and 1.
- If only k of the n users receive equal throughput and the remaining n-k users receive zero throughput, the fairness index is k/n.

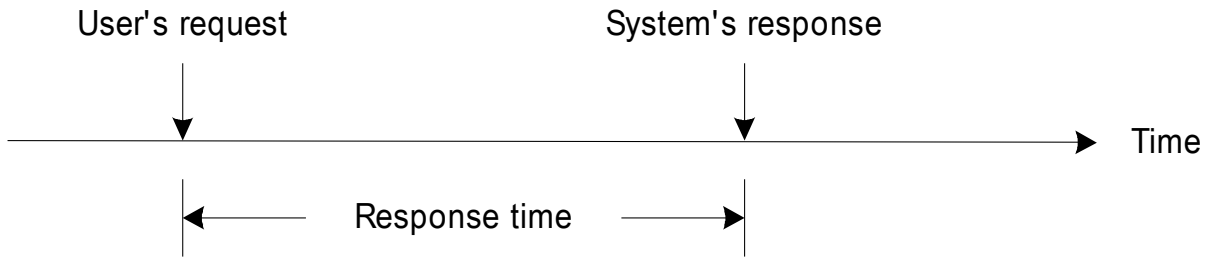
Case Study 3.1 (5)

- After a few experiments, it was clear that throughput and delay were really redundant metrics. ⇔ All schemes that resulted in higher throughput also resulted in higher delay.

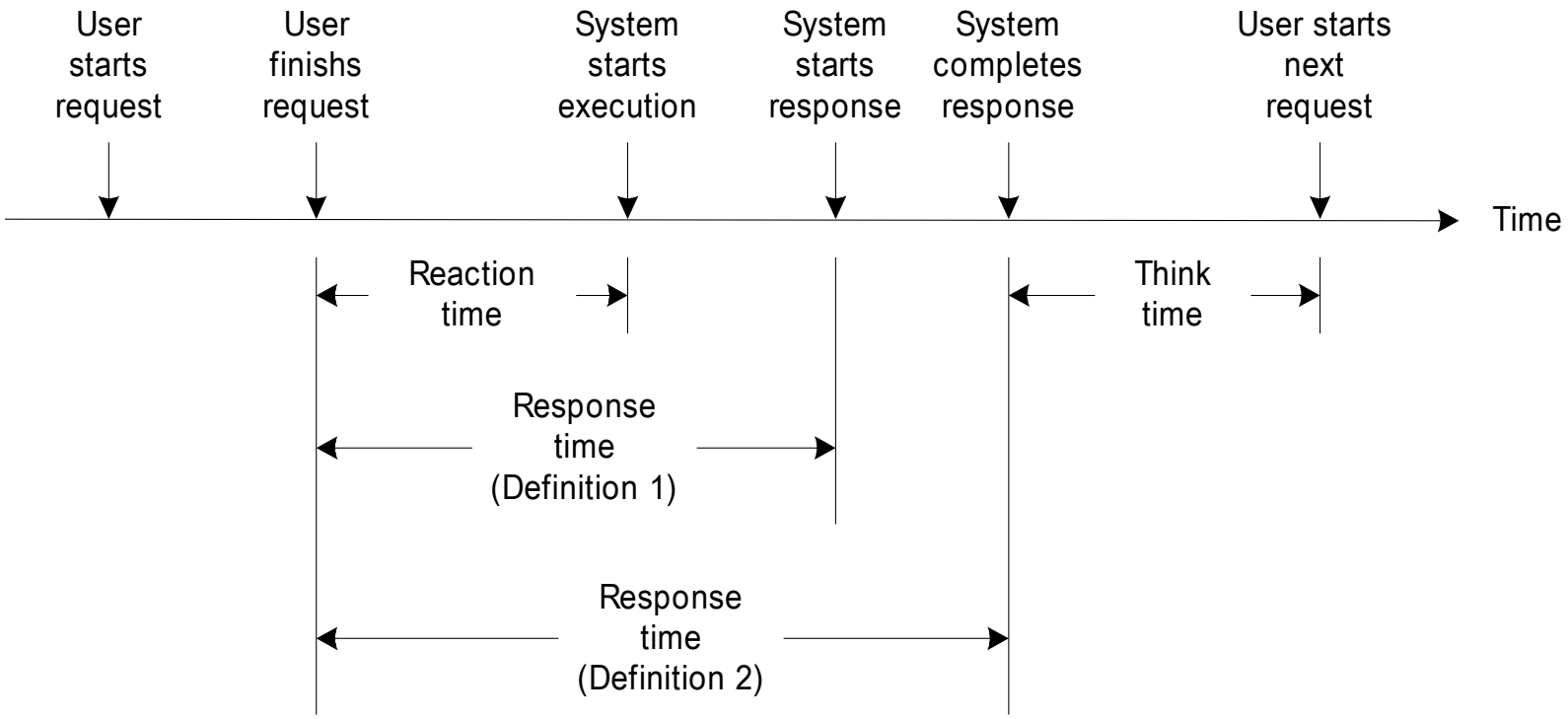
- The variance in response time was dropped since it was redundant with the probability of duplication and the probability of disconnection.

3.3 Commonly Used Performance Metrics (1)

- Response time : the interval between a user's request and the system response, as shown in Figure 3.2a.
  - This definition is simplistic since the requests as well as the responses are not instantaneous.
- The user spend time typing the request and the system takes time outputting the response, as show in Figure 3.2b.
  - It can be defined as either the interval between the end of a request submission and the beginning of the corresponding response from the system or as the interval between the end of a request submission and the end of the corresponding response form the systems.



(a) Instantaneous request and response



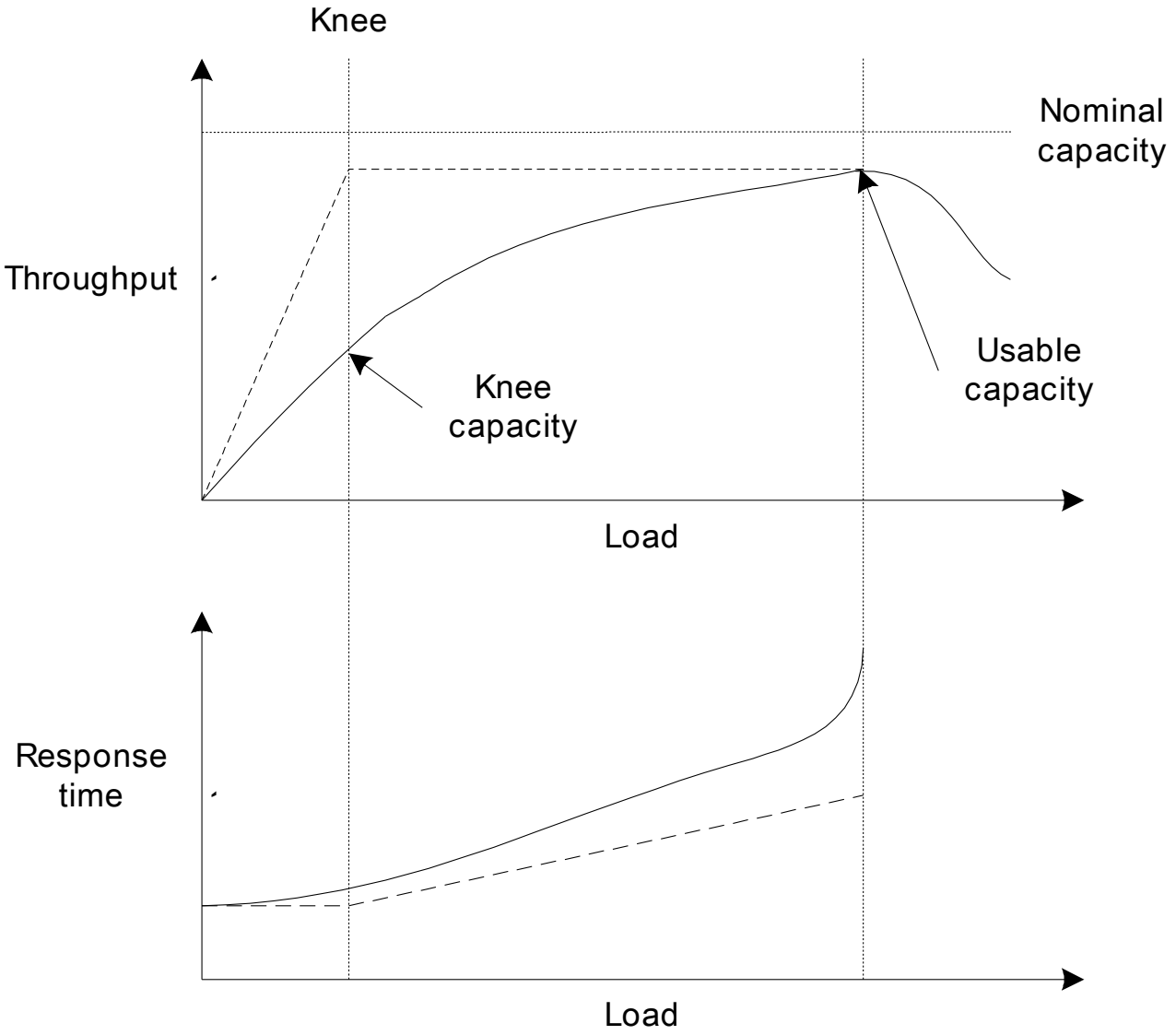
(b) Realistic request and response

3.3 Commonly Used Performance Metrics (2)

- Turnaround time : the time between the submission of a batch job and the completion of its output.
  - Notice that the time to read the input is included in the turnaround time.
- Reaction time : the time between submission of a request and the beginning of its execution by the system
  - To measure the reaction time, one has to able to monitor the actions inside a system since the beginning of the execution may not correspond to any externally visible event.
- Stretch factor : the ratio of response time at a particular load to that at the minimum load
  - The response time of a system generally increases as the load on the system increases.

3.3 Commonly Used Performance Metrics (3)

- Throughput is defined as the rate (requests per unit of time) at which the requests can be serviced by the system.
  - For batch systems, jobs per second.
  - For interactive systems, requests per second.
  - For CPU, MIPS(Millions of Instructions Per Second), or MFLOPS (Millions of Floating-Point Operations Per Second)
  - For networks, packets per second(pps) or bits per second(bps)
  - For transactions processing system, TPS(Transactions Per Second)
- After a certain load, the throughput stops increasing; in most cases, it may event start decreasing, as shown in Figure 3.3.


Knee
Nominal capacity
Throughput
Knee capacity
Usable capacity
Load
Response time
Load

3.3 Commonly Used Performance Metrics (4)

- Nominal capacity : the maximum achievable throughput under ideal workload conditions
- Usable capacity : It is more interesting to know the maximum throughput achievable without exceeding a prespecified response time limit.
- Knee capacity : the throughput at the knee
  - In many applications, the knee of the throughput or the response time curve is considered the optimal operating point.
- Efficiency : the ratio of maximum achievable throughput (usable capacity) to nominal capacity
- The utilization of a resource is measured as the function of time the resource is busy servicing requests. ⇔ the ratio of busy time and total elapsed time over a given period.

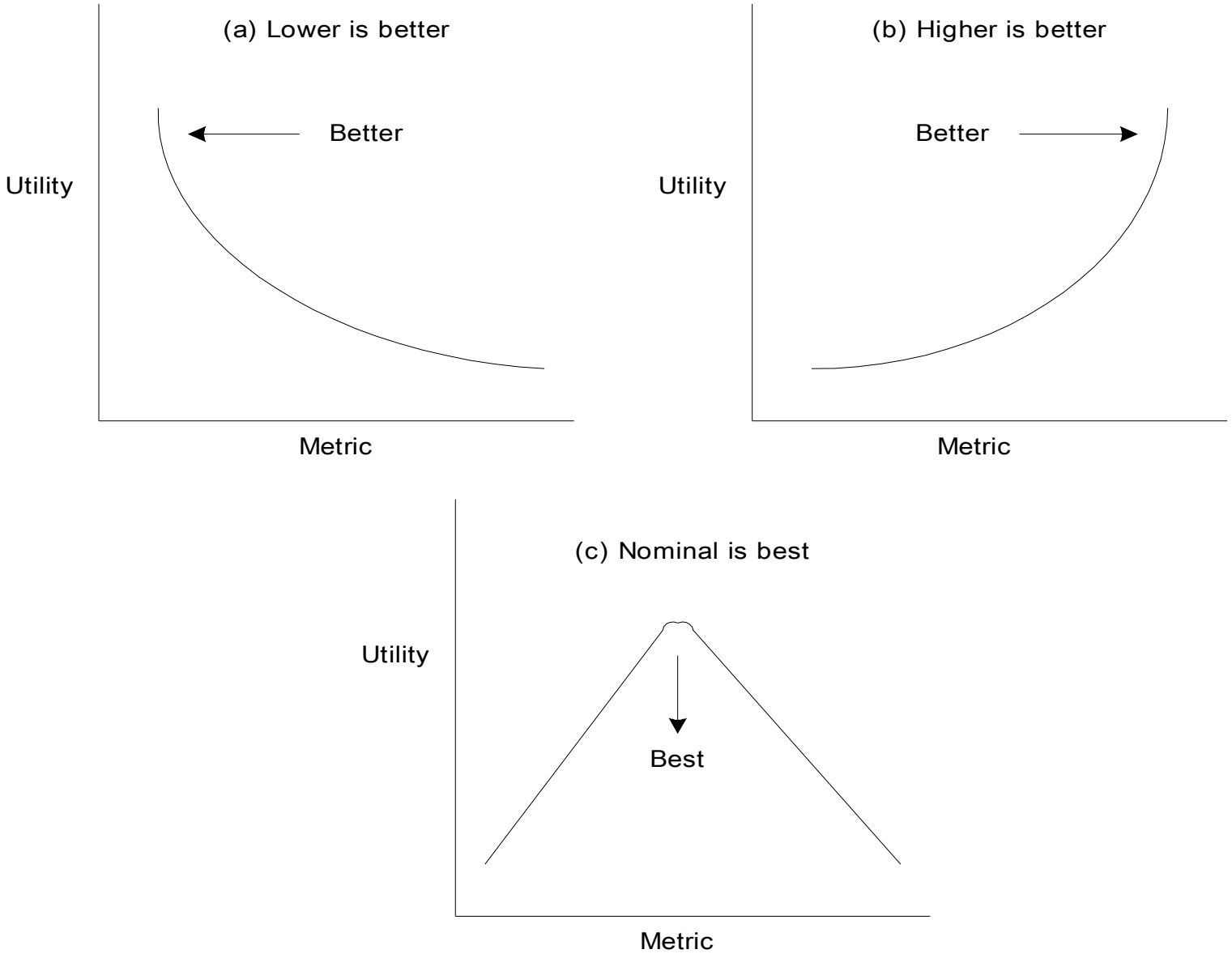
3.3 Commonly Used Performance Metrics (5)

- Idle time : the period during which a resource is not being used.
- Reliability : the probability of errors or by the mean time between errors.
- Availability : the fraction of the time the system is available to service user's requests.
- Downtime : the time during which the system is not available.
- Uptime : the time during which the system is available(MTTF-Mean Time To Failure).
- Cost/performance ratio : a metric for comparing two or more systems.

3.4 Utility Classification of Performance Metrics

- Higher is Better or HB.
  : System users and system managers prefer higher values of such metrics. Ex) System throughput
- Lower is Better or LB.
  : System users and system managers prefer smaller values of such metrics. Ex) Response time
- Nominal is Best or NB.
  : Both high and low values are undesirable. Ex) Utilization

- Figure 3.5 shows hypothetical graphs of utility of the three classes of metrics.


(a) Lower is better
Better
Utility
Metric
(b) Higher is better
Better
Utility
Metric
(c) Nominal is best
Utility
Best
Metric

3.5 Setting Performance Requirements (1)

- Typical requirement statements
  - The system should be both processing and memory efficient. It should not create excessive overhead.
  - There should be an extremely low probability that the network will duplicate a packet, deliver a packet to the wrong destination, or change the data in a packet.
- These requirement statements are unacceptable since they suffer from one or more of the following problems.
  - Nonspecific : No clear numbers are specified.
  - Nonmeasurable
  - Nonacceptable
  - Nonrealizable
  - Nonthroughput

3.5 Setting Performance Requirements (2)

- What all these problems lack can be summarized in one word : SMART(Specific, Measurable, Acceptable, Realizable, Thorough)
  - Specificity precludes the use of words like “low probability” and “rate”.
  - Measurability requires verification that a given system meets the requirement.
  - Acceptability and Realizability demand new configuration limits or architectural decisions so that the requirements are high enough to be acceptable and low enough to be achievable.
  - Thoroughness includes all possible outcomes and failure modes.

Case Study 3.2 (1)

- Consider the problem of specifying the performance requirements for a high-speed LAN system.
  - The performance requirements for three categories of outcomes were specified as follows:
  - Speed : If the packet is correctly delivered, the time taken to deliver it and the rate at which it is delivered are important. This leads to the following two requirements:

    (a) The access delay at any station should be less than 1 second.

    (b) Sustained throughput must be at least 80 Mbits/sec.
  - Reliability : Five different error modes were considered important. Each of these error modes causes a different amount of damage and, hence, has a different level of acceptability. The probability requirements for each of these error modes and their combined effect are specified as follows

Case Study 3.2 (2)

(a) The probability of any bit being in error must be less than 10^{-7}.
(b) The probability of any frame being in error (with error indication set) must be less than 1%.
(c) The probability of a frame in error being delivered without error indication must be less than 10^{-15}.
(d) The probability of a frame being misdelivered due to an undetected error in the destination address must be less than 10^{-18}.
(e) The probability of a frame being delivered more than once (duplicate) must be less than 10^{-5}.
(f) The probability of losing a frame on the LAN (due to all sorts of errors) must be less than 1%.

Case Study 3.2 (3)

- Availability : Two fault modes were considered significant. The first was the time lost due to the network reinitializations, and the second was time lost due to permanent failures requiring field service calls. The requirements for frequency and duration of these fault modes were specified as follow:

  (a) The mean time to initialize the LAN must be less than 15 milliseconds.

  (b) The mean time between LAN initializations must be at least 1 minute.

  (c) The mean time to repair a LAN must be less than 1 hour. (LAN partitions may be operational during this period.)

  (d) The mean time between LAN partitioning must be at least half a week.

Computer Systems Lab.

http://tolerance.ajou.ac.kr

다중프로세서 시스템 모델

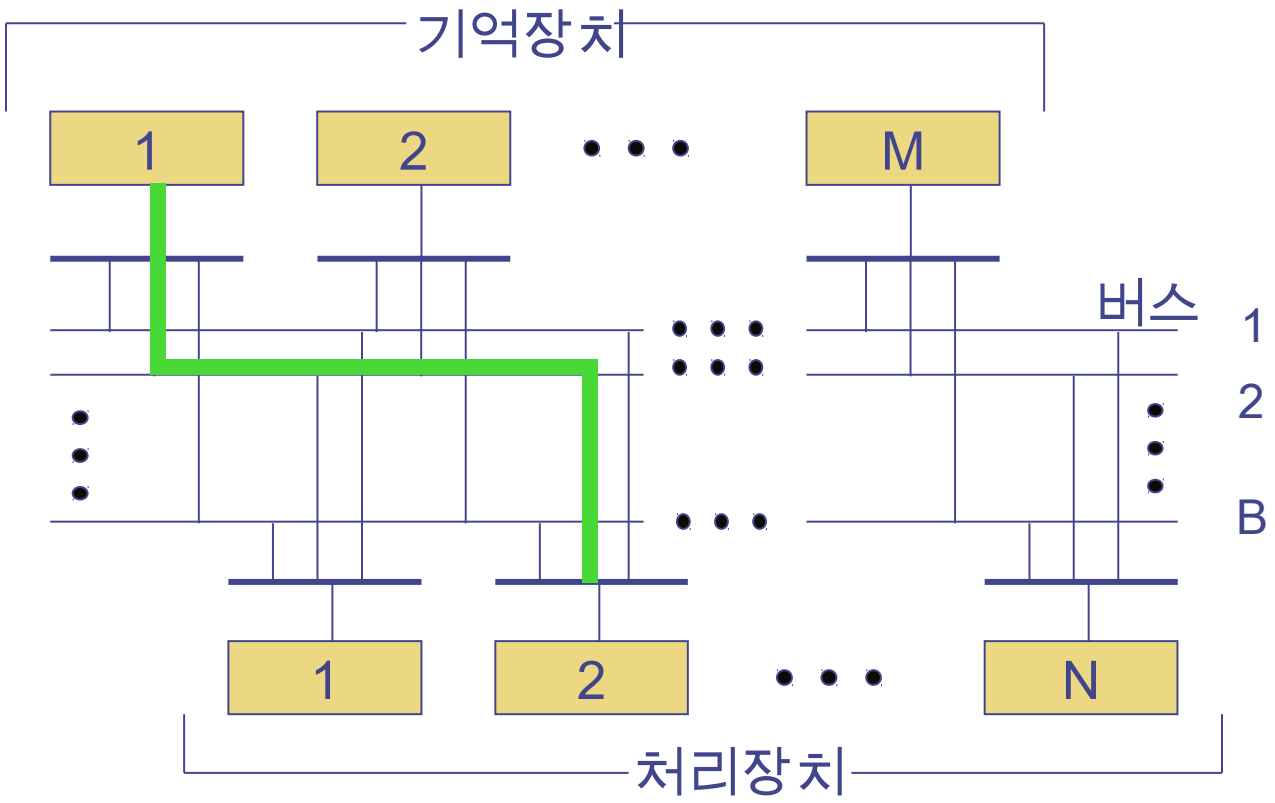
정 지 영

목차

1. 개요

- 다중프로세서 시스템에서의 메모리, 버스 경쟁 모델
- 시스템의 분석적 모델 개발
- 분석 결과를 검사하기 위한 시뮬레이션 모델 개발

2. 다중프로세서 시스템



멀티프로세서 시스템 요소

2. 다중프로세서 시스템

- N 개의 프로세서가 물리적으로나 기능적으로 동일하다고 가정
- 프로세서는 자신의 지역 메모리를 가지고 있으며 이들 메모리는 캐쉬이거나 자료 레지스터와 명령 버퍼의 형태를 취할 수 있다 .
- 실행 시 프로세서는 각 머신 싸이클 동안에 명령어 인출이나 연산자 인출 또는 저장 요청 발생

- 요청확률 h 는 프로세서의 지역 메모리에서 응해지고 , 확률 p=1-h 로 메모리로의 접근 요청
- h: 적중률
- p: 비 적중률

2. 다중프로세서 시스템

메모리 요청 처리

1. 프로세서는 머신 싸이클이 시작될 때 메모리 요청을 초기화하는 동시에 실행을 일시 정지 시킨다 . 동일 모듈에 대한 다중 요청은 중재 메커니즘에 의해 해결한다 .

2. 요청이 성공하면 해당 모듈은 프로세서에 대해 점유되고 , 그렇지 않으면 다음 싸이클의 시작에서 다시 발생한다 . 두번째 중재 메커니즘은 M 개까지의 성공적 요청의 집합으로부터 B 개까지의 요청을 선택하여 이들 요청에 대한 버스들을 점유한다 . 여기서 버스가 점유되는 순서는 모듈이 점유되었던 순서와 동일하다고 가정

2. 다중프로세서 시스템

3. 성공적인 요청에 대해서 그 요청이 저장이라면 , 주소와 자료가 버스를 통해 메모리로 전송되고 만일 그 요청이 인출이라면 해당 싸이클의 종료 시 버스를 통하여 자료가 반환된다 .

4. 버스 싸이클이 끝날 때 , 요청들이 성공적으로 완료된 프로세서들은 실행으로 되돌아가고 , 이들 요청에 의해 점유된 버스와 모듈은 해제된다 . 성공하지 못한 요청은 다음 싸이클의 시작에서 새로운 요청과 함께 재발생된다 .

- 목적 : 프로세서 성능이 메모리 모듈과 버스에 대한 경쟁에 의해서 얼마나 영향을 받는가를 결정하는 것

3. 근사 분석 모델

- 시스템 대역폭 (BW)
  - 프로세서와 메모리 사이의 전체 전송률로 단위 시간당 전송의 관점으로 표현
  - 전송 시간이 1 싸이클이기 때문에 전체 버스 활용은 BW 와 같고 BW 는 종종 사용중인 버스의 평균 수로서 정의

- 어떤 한 싸이클 동안 발생하는 요청의 확률이 다른 싸이클에서 발생하는 확률과 같고 또 독립적이라 가정하면, 프로세서의 실행 간격은 Bernoulli 실행열에 해당
- 실행간격 평균 : (1-p)/p
- 메모리 요청은 메모리에 대해 독립, 일양 분포를 한다고 가정

3. 근사 분석 모델

- 프로세서 i 가 메모리 j 를 요청할 확률 : p/M
- 프로세서 i 가 메모리 j 를 요청하지 않을 확률 : 1- p/M
- 메모리 j 에 적어도 하나의 요청이 있을 확률

$$q = 1 - (1 - p/M)^N \qquad (\text{식 } 5.1)$$

- 메모리 요청확률이 동일하고 독립적이라고 가정하면 , M 개의 메모리 중 i 번째를 요청할 확률 f_i 는 이항분포이다 .

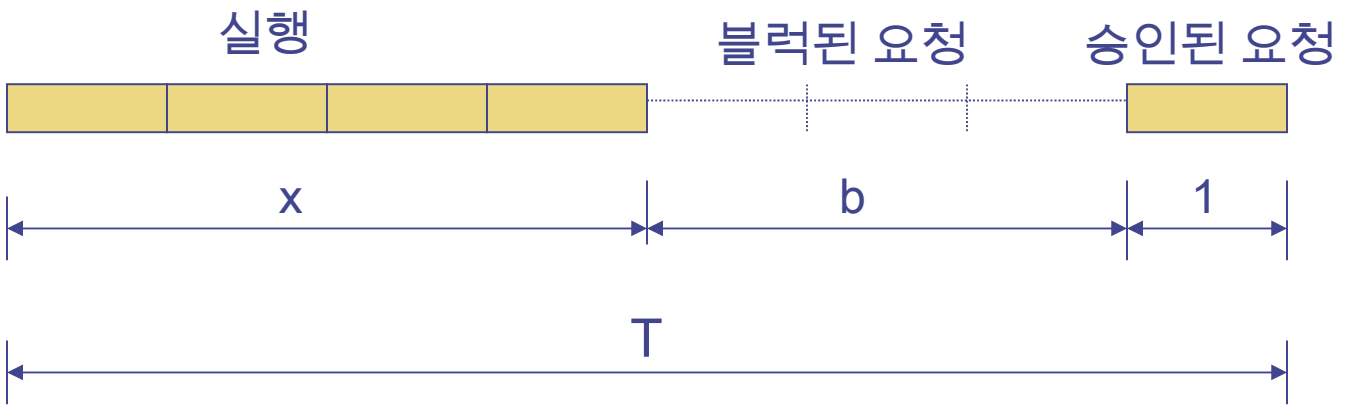
$$f_i = \binom{M}{i} q^i (1-q)^{M-i} \qquad (\text{식 } 5.2)$$

- 한 싸이클에서 승인되는 버스 요청의 기대값

$$BW = B\sum_{i=B}^{M} f_i + \sum_{i=1}^{B-1} i \times f_i \qquad (\text{식 } 5.3)$$

3. 근사 분석 모델

- 추정한 대역폭은 요청이 재발생되지 않는 싸이클에만 적용
- 프로세서당 평균 요청률 (비 적중률): p, 전체 요청률 : Np
- 일반적인 경우에 모든 싸이클을 고려해 보면 프로세서당 요청률 r 은 비 적중률보다 크다 .



프로세서 상호요청 간격 타이밍

3. 근사 분석 모델

- 단일 프로세서 요청률
  - r=(b+1)/T = (b+1)/(x+b+1)
  - 분모와 분자를 b+1 로 나누면
  - r=1/[1+x/(b+1)]
  - b+1=rT : T 동안에 발생된 요청의 전체 횟수
  - 프로세서당 요청완료율 : BW/N
  - T=N/BW
  - b+1=Nr/BW
  - **r= 1/[1+xBW/Nr]**

3. 근사 분석 모델

- BW 를 추정하기 위한 단순 고정 소수점 반복 알고리즘

1. 식 (5.1) ~(5.3) 을 사용하여 초기 대역폭 BW_0 의 추정값을 계산한다 .
2. 다음에서 r 의 개선된 추정값을 계산한다 .
   - $r_i = 1/[1+xBW_{i-1}/Nr_{i-1}]$
1. $q=1-(1-r_i/M)^N$ 을 계산하고 식 (5.2) 와 (5.3) 을 사용하여 새로운 추정값 BW_i 를 계산한다 .
2. $|Bw_i-Bw_{i-1}| < e$ 라면 종료하고 그렇지 않으면 단계 2 로 돌아간다 .

3. 근사 분석 모델

- e=0.005 인 경우 C 알고리즘

```
real BW(p,B,M,n)
  real p; intB, M, N;
    {
       real bw0, bw1=p*N, r=p, x=1.0/p-1.0, Bwi();
       do
         {
            bw0 = bw1; r=1.0/(1.0+x*bw0/(N*r));
            bw1=BWi(r,B,M,N);
          }
       while (fabs(bw1-bw0) > 0.005);
       return(bw1);
     }
```

3. 근사 분석 모델


```
real Bwi (r,B,M,N)
   real r; intB, M, N;
     { /* compute bandwidth for request rate r */
        int I; real q, bw=0.0, f();
        q=1.0-pow(1.0-r/M, (real)N);
        for(i=1; i<B; i++) bw += i*f(i,M,q);
        for(i=B; i<=M; i++) bw += B*f(i,M,q);
        return (bw);
      }

real Fact(n)
   int n;
     { /* compute n factorial */
       real z=1.0;
       while (n) {z*=n; n--;}
       return (z);
      }
```

3. 근사 분석 모델


```
real C(n,k)
   int n,k;
      { /* compute binomial coefficient */
         return (Fact (n)/Fact(k) * Fact(n-k)));
       }

real f(i,M,q)
   int i, M; real q;
      { /* compute binomial probability */
         real z;
         z=C(M,i)*pow(q,(real)i)*pow(1.0-q,(real)(M-i));
         return(z);
       }
```

3.1 활용과 대기시간

- 단일 버스 , 단일 메모리 모듈 그리고 단일 프로세서의 평균 활용은 각각 $U_b=BW/B$, U_m 그리고 $U_p=xBW/N$ 이다 .
- 요청당 평균 대기 시간은 b 이고 b 는 T-x-1 이다 .
- T=N/BW 이고 x+1=1/p 이므로 b 는 다음과 같다 .
- b=(N/BW)-(1/p)

3.2 평균 큐 길이

- L_b를 프로세서당 블럭된 요청의 평균 숫자라고 하자 . 리틀의 법칙으로부터

$$L_b = bBW / N$$

- 얻어지며 , 이것으로부터 다음을 구할 수 있다 .

$$L_b = 1 - BW / Np$$

3.3 시스템 처리량

- 시스템에서 작업의 단위를 태스크라고 하고 각 태스크는 평균 n 실행시간 간격을 요구한다고 하면, 태스크당 평균 서비스 시간은 nx 이다.
- 활용 법칙으로부터 시스템 처리량은 U_p 태스크임을 알 수 있다.

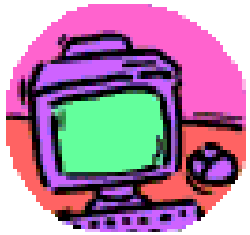
- 설계 비교를 위하여 nx=1 을 취함으로써 얻어진 정규화된 시스템 처리량 XP 를 이용하는 것이 유용하다.

$$XP = NU_p = N[xBW/N] = BW[(1/p)-1]$$

4. 시뮬레이션 모델

- 분석 모델 검증에 대해 개발하고자 하는 시뮬레이션 모델의 종류는 목적에 따라 달라진다 .

- 다중 프로세서 시스템에 있어서 목적은 분석 모델이 시스템의 행위를 적절하게 표현하는가를 검증하는 것이다 .



4.1 시뮬레이션 모델 1


```
#include <smpl.h>
#define busy 1

real
   p=0.250,          /* local memory miss rate              */
   treq[17]          /* next request time for processor     */
   tn=1.0E6;         /* earliest-occurring request time     */
int
   N=8, M=4, nB=2,   /* no. processors, memories, & buses   */
   modole[17],bus,   /* memory & bus facility descriptors   */
   nbs=0,            /* no. busy buses current cycle        */
   req[17],     /* currently-requested memory module */
   next=1,      /* arbitration scan starting point */
```

4.1 시뮬레이션 모델 1


```
/*----------- MEMORY-BUS BANDWIDTH MODEL-----*/
main() {
    int event, i,n;
    smpl  (0, “bandwidth Model”);
    for (i=1; i<=M, i++)  module [i]=facility(“module”,1);
    for (n=1; n<=N; n++) {req[n++] {req[n]=0; next_access  (n) ;}
    schedule(1,tn,0);
    while (time() < 10000.0)
      {
         cause (&event,&n) ;
         switch (event) {
            case 1: begin_cycle() ; break;
            case 2: req_module(n) : break;
            case 3: end cycle(n); break;
         }
      }
   printf(“BW=%.3f\n”, U(bus));
```

4.1 시뮬레이션 모델 1

Computer Systems Lab.

http://tolerance.ajou.ac.kr

```
/----------COMPUTE NEXT ACCESS TIME---------*/
nest_access(n)
int n;
{
   real t;
   t=floor(log(ranf())/log(1.0-p))+time();
   treq[n]=t; if(t<tn) then tn=t;
}
```

- next_access() 함수가 각 프로세서에 대한 초기 접근 시간을 결정하고 tn 을 계산하기 위하여 호출
- treq[n] 은 사건 발생 시간이지 사건간 시간은 아님 , 프로세서 n 의 다음 요청 발생 시간
- tn 은 최초로 발생한 요청의 발생시간

4.1 시뮬레이션 모델 1

- 이 모델에서는 시간의 단위가 버스 싸이클이므로, 시간은 이 단위의 정수배로 진행하고, 사건은 싸이클의 시작이나 끝에서 발생한다.

- begin_cycle() 은 시간 tn 에 발생하는 요청에 대해 N 개 프로세서들을 스캔하여 이러한 각 프로세서에 대해서 임의 메모리 모듈을 요청 목적지로서 할당하고 메모리 (그리고 버스) 요청 즉, 사건 2 를 스케줄한다.

4.1 시뮬레이션 모델 1

- req_module() 은 요청된 메모리 모듈이 사용 가능하고 , 버스가 사용 가능한가를 보기 위해 검사한다 .
- 이들 조건이 만족되면 모듈과 버스는 점유되고 이 싸이클에서의 사용중인 bus 의 숫자 , 즉 nbs 가 증가된다 . 요청의 완료는 그 싸이클의 끝에서 발생되도록 스케줄 된다 .
- 요청된 모듈이 사용 중이거나 버스가 얻어질 수 없다면 , 요청은 다음 싸이클의 시작에서 재발생되기 위해 블럭된다 .

- 사건 3 은 한 버스 싸이클의 끝에서 요청의 완료를 표시한다 . 버스와 메모리 모듈이 해제되고 , req[n] 은 0 으로 설정되며 next_access() 는 프로세서에 대한 다음 요청 발생 시간을 계산하기 위하여 호출된다 .

4.1 시뮬레이션 모델 1

Computer Systems Lab.

http://tolerance.ajou.ac.kr

```
/*----EVENT 1: BEGIN CYCLE-------*/
begin_cycle() {
   int i,n=next: real t, tmin=1.0E6;
   for (i=0; i<N; i++) {
      if (!req[n]) then {/* in this version, req[n] always is 0 here */
         if ((t=treq[n])==tn)
            then
               {req[n]=random(1,M); schedule(2,0.0n);}
            else if (t<tmin) then tmin=t;
      }
   n=(n%N)+1;
   }
next=(next%N)+1; tn=tmin;
}
```

4.1 시뮬레이션 모델 1


```
/*------EVENT 2: REQUEST MEMORY AND BUS----------*/
req_module(n)
int n;
{
   if (status (module[req[n]]!=busy&&status(bus)!=busy)
      then {
         request(module[req[n]],n,0); request(bus,n,0);
         nbs++; schedule(3,1.0,n);
      }
   else
      {req[n]=0; if (++treq[n]<tn) then tn=treq[n];}
}
```

4.1 시뮬레이션 모델 1


```
/*---------EVENT 3: END CYCLE---------*/
end-cycle(n)
{
   release(bus,n);
   release(module[req[n]].n);
   req[n]=0;
   next_access(n);
   if (--nbs==0) then schedule(1, tn-time(),0);
}
```

4.2 시뮬레이션 모델 2

- 분석 모델에서 블럭된 요청은 재발생될 때 M 개 모듈중의 어느 하나에 임으로 재할당된다고 가정하고 시뮬레이션 모델 1 에 반영하였으나 쉽게 제거될 수 있다 .
- 실제 시스템은 요청 큐잉의 몇몇 타입을 제공할 수 있는데 반하여 , 결과 모델은 큐와 버스에 대한 요청에 대해 다소 임의적인 선택을 제공한다 .
- 이것이 성능에 얼마나 영향을 주는가를 보기 위하여 , 모듈과 버스 요청이 선입선출에 기반해서 큐잉되는 시스템의 모델을 살펴본다 .

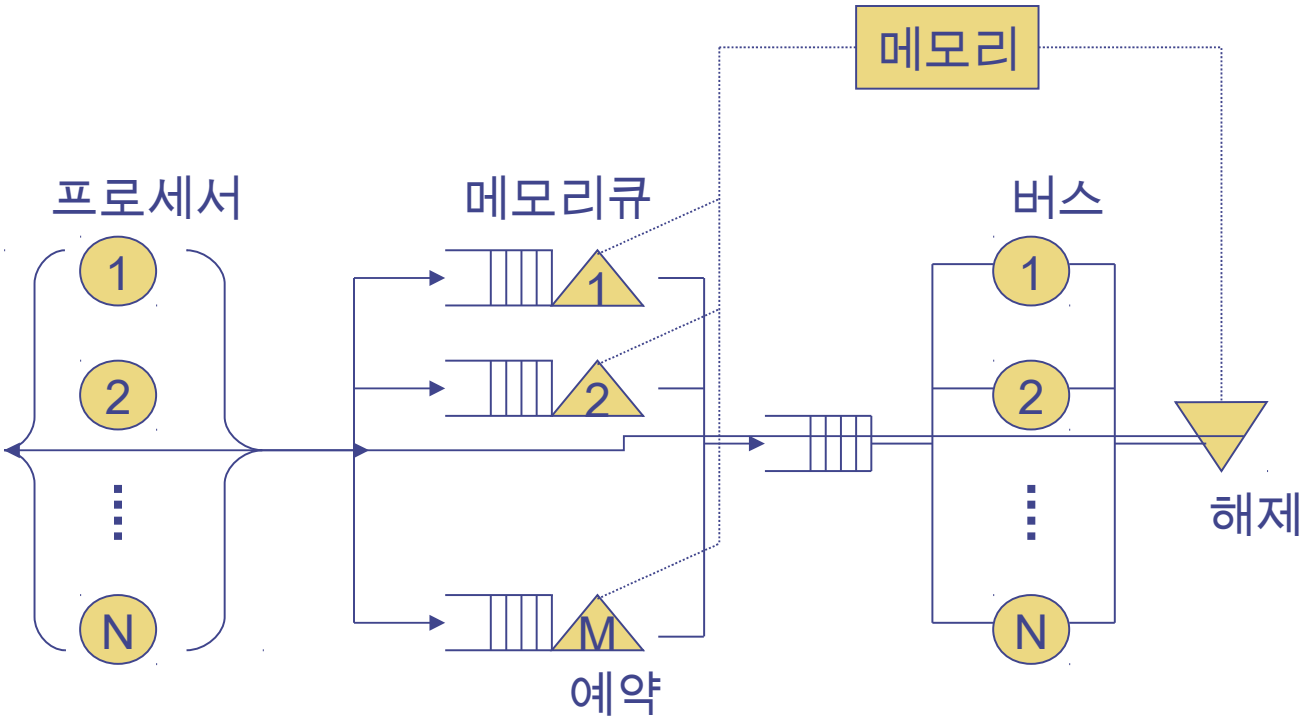
4.2 시뮬레이션 모델 2

- 프로세서는 원의 집합인 지연서버에 의해 표현된다 . 지연 서버는 항상 요청을 지원하기 위한 충분한 서버를 가지고 있어서 큐가 필요 없다 .

- 프로세서가 메모리 요청을 발생시키면 , 요청은 선택된 메모리 모듈에 대해 큐잉되고 , 그것을 점유하고 , 버스에 대해서 큐잉되고 , 버스를 점유한다 .

- 메모리 모듈과 버스는 전송을 수행하기 위하여 동시에 점유되어야 한다 .

4.2 시뮬레이션 모델 2



다중프로세서 시스템 큐잉 모델 다이어그램

4.2 시뮬레이션 모델 2

- 이 모델의 장점은 요청당 평균 지연이 메모리 지연과 버스 지연으로 분리될 수 있다는 것이다 .

```
/*-------EVENT2: REQUEST MEMORY------------*/
req_module(n)
int n;
{
   if (request (module[req[n]],n,0)==0) then
      schedule”(3,0.0,n);
}
```

4.2 시뮬레이션 모델 2


```
/*---------EVENT 3: REQUEST BUS-----------------*/
req_bus(n)
int n;
{
   if (request (bus,n)==0) then
      {nbs++; schedule(4,1.0,n);}
}
```

4.2 시뮬레이션 모델 2


```
/*------------EVENT 4: END CYCLE---------------*/
end_cycle(n)
int n;
{
   req[n]=-req[n]; nbs--;
   if (nbs==0) then {
      for (n=1; n<=N; n++)
         if (req[n]<0) then {
            release(bus,n);
            release(module[-req[n]],n);
            req[n]=0; next access(n);
      }
   schedule(1,tn-time(),0);
   }
}
```

4.3 시뮬레이션 모델 3

- 시뮬레이션 모델 1과 2는 실 시스템의 동기화 작동을 표현하기 위하여 구성되었기 때문에 고정된 단위 시간 진행을 사용하였고 싸이클 끝점에서 요청 초기화와 완료를 정렬하였다 .

- 큐잉 네트워크 모델은 다중프로세서 시스템을 모델화하는데 있어서 성공적으로 사용되어 왔고 , 또 이들은 동기화 행위를 표현하도록 제한되어 오지도 않았다 .

- 비동기화 큐잉 모델로부터의 결과와 이러한 모델이 어떻게 비교되는가 본다 .

4.3 시뮬레이션 모델 3

Computer Systems Lab.

http://tolerance.ajou.ac.kr

```
#include <smpl.h>
#define queued 1
real p=0.250;              /* local memory 비적중율              */
int  N=8, M=4, nB=2,       /* no. processors, memories, & buses  */
     module[17],           /* facility descroptors for modules   */
     bus,                  /* focility descriptors for buses     */
     req[17];              /* currently-requested memory module */
```

4.3 시뮬레이션 모델 3


```
main()
{
   int event, I, n; real x=1.0/p-1.0;
   smpl(0,"Bandwidth Model") ;
   bus=facility("bus",nB) ;
   for(i=1; i<=M, i++) module[i]=facility("module",1);
   for(n=1; n<=N; n++)  {
      req[n]=random(1,M); schedule(1, expntl(x),n; )
   }
```

4.3 시뮬레이션 모델 3


```
while (time()<10000.0) {
   cause(&event,&n);
   switch(event) {
      case 1:
        if (request(module[req[n]], n, 0)!=queued)
        then schedule(2, 0.0, n);          break;
      case 2: /* reserve bus & initiate transfer   */
        if (request(bus, n, 0) !=queued) then
        schedule(3, 1.0, n);          break;
      case 3: /* complete:  schedule next request   */
        release(bus, n);
        release(module[req[n]], n);
        req[n]=random(1, M);
        schedule(1,espntl(x), n);         break;
   }
}/* end-while */
report();
}/* end-main */
```

5. 분석 모델과 시뮬레이션 모델 결과

Computer Systems Lab.

http://tolerance.ajou.ac.kr

N	M	B	P	ana	sim1	sim2	sim3
4	4	4	1.000	2.734	2.739	2.619	2.613
4	4	2	.500	1.583	1.668	1.664	1.665
4	4	1	.250	.807	.327	.927	.339
4	2	1	.250	.818	.327	.927	.339
4	2	1	.251	.481	.487	.137	.484
8	8	8	1.000	5.251	5.253	4.984	4.934
8	8	4	.500	3.273	3.379	3.334	3.352
8	8	2	.250	1.706	1.774	1.718	1.739
8	4	2	.250	1.890	1.711	1.713	1.709
8	4	1	.251	.860	.866	.993	.861

6. 다중프로세서 시스템 모델의 확장

- 일정하지 않은 요청률
  - 중앙 및 입출력 프로세서의 혼합과 같은 다른 요청률을 가진 서로 다른 프로세서 타입으로 구성된 시스템을 모델링하기를 원할 수 있다 .

- 비임의적이거나 비일양적인 메모리 주소지정
  - 특정 모듈에 대한 요청의 쇄도나 블록 전송에 의해 생성된 순차 메모리 주소에 대한 참조의 형태를 표현하고자 할 수 있다 .

6. 다중프로세서 시스템 모델의 확장

Computer Systems Lab.

http://tolerance.ajou.ac.kr

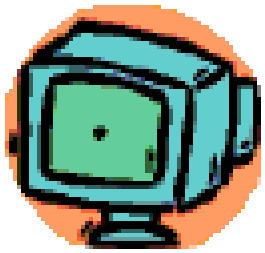
- 다양한 전송 길이
  - 요청당 하나의 단위의 전송을 가정했었다 . 이것을 요청당 여러 단위가 전송될 수 있도록 확장시키길 원할 수 있다 .

- 서로 다른 버스와 메모리 서비스 시간
  - 하나의 전송이 한 싸이클 동안 메모리 모듈과 버스 모두를 사용중 상태로 유지한다고 가정했으나 메모리 모듈 사용중인 시간이 자료전송 시간보다 긴 메모리 서브 시스템의 모형을 만들기를 원할 수 있다 .

Computer Systems Lab.

http://tolerance.ajou.ac.kr

SimScript II.5



정 지 영

1. Introduction

1. Introduction to SimScript II.5

- SimScript II.5 는 CACI Products Company 에서 만든 범용 시뮬레이션 언어로서 프로세스 지향, 사건 지향으로 만들어지는 모델을 구현하는데 적합하다

1.1 변수 (Variable)

- 문자 (letter), 숫자 (digit), 마침표 (period) 를 조합해 만든다. 대소문자의 구별은 없다.

1.2 입력데이터 읽기 (Reading Input Data)

- 데이터를 읽는 문은 READ 이다.

1. Introduction

1.3 수식 표현 (Arithmetic Expression)

수식 연산자는 다음과 같다.

+ (add), - (subtract), * (multiply), / (divide), ** (exponentiate)

example :

read x and y
add x to y
print 1 line with y thus
The sum is : ***

1. Introduction

1.4 변수값 계산 (Computing Variable Values)

- 변수에 값을 할당하는 문은 LET 이다. 그러나 굳이 이 문을 쓸 필요는 없다.

example : **let** $x = x + 1$

1.5 특별한 계산문 (Special Computation Statements)

Add / Subtract

example : **add** *1 to counter*

1. Introduction

1.6 계산 결과 보여주기 (Displaying the Results of Computation)

example : print 1 line with *PRICE/ITEMS* thus

PRICE/ITEM = $*.***

1.7 반복문 (Repetition)

for 문을 사용할 수 있다. (do loop 문으로 반복 구간을 정할 수 있다.)

example :

```
for i=1 to 5 by 1
  do
        read X
        read Y
  loop
```

1. Introduction

1.8 프로그램 종료

- stop 문은 프로그램을 논리적으로 끝내는 명령문이고,
- end 문은 물리적으로 끝내는 명령문이다.

1.9 Variable Modes

- SimScript II.5 의 numerical variable 은 REAL / INTEGER 2 가지밖에 없다.
- 표현 가능한 숫자의 크기는 Computer 에 dependent 하다.
- variable type definition 은 Preamble 에서 이루어진다.

1. Introduction

1.10 Routines

- CALL *routine name* : routine 을 호출할 때
- RETURN 은 call 된 routine 을 종료할 때 쓴다.
- argument passing
  - routine <name> given <argument> yielding <argument>
- function 을 사용하여 routine 을 표현할 수가 있다.

  preamble 에서 "DEFINE *name* AS *mode function*" 으로 정의하고 return value 는 function 내에서 "RETURN WITH *arithmetic expression*" 으로 한다.

  example : function Absolute(Number)

  ...

  return with Number

  end

1. Introduction

1.11 Library Functions

○○○.f 로 이루어져 있다. 예를 들면 abs.f 는 주어진 argument 의 절대값을 return 한다.

1.12 Text Mode Variables

텍스트를 표현하는 변수 모드이다. real / integer 처럼 선언한다.

1.13 Alpha Variables

문자 하나를 변수로 선언할 때 사용되는 모드.

1.14 Adding Performance Measurement

- U.resource : 현재 이용 가능한 자원의 수
- N.Q.resource : 큐에 있는 자원의 수
- N.X.resource : 현재 실행되고 있는 자원의 수

2. Elementary modeling concept

Model Structure

시뮬레이션을 하는 모델은 다음의 구성요소를 가져야 한다.

1) 새로운 객체의 도착을 표현하는 메카니즘
2) 모델된 시스템 안에서 그 객체에서 일어나는 일의 표현
3) 시뮬레이션을 종료시키는 메카니즘

Process Concept

프로세스는 모델 안에서 시뮬레이션이 수행되는 시간동안 능동적으로 행동하는 개체이다

2. Elementary modeling concept

Resource Concept

- 자원 (resource) 은 모델 안에서 프로세스가 요구하는 일을 행하는 수동적인 개체이다.

Program Structure

1) Preamble : C 의 Header File 과 유사하다.

2) Main program : 시뮬레이션이 수행되도록 하는 절차를 밟는 부분이다. 시스템의 컨트롤이 Timing Routine 으로 넘어가는 동작으로 수행한다.

3) Process routine : preamble 에서 선언된 process 의 동작을 표현하는 routine

Timing routine

- Discrete-event simulation 의 심장부로 모델 개발자에게 투명

예제 : A Simple Gas Station Model

Computer Systems Lab.

http://tolerance.ajou.ac.kr

[Model 개요]

- 주유펌프가 2 개인 주유소가 있다 . 이 주유소에는 고객이 random 하게 찾아온다 . 고객이 주유소에 도착하는 경우 먼저 서비스를 기다리고 , 서비스를 받은 후 떠나게 된다 . 이러한 시스템으로부터 이 주유소에 주유펌프가 효율적으로 작동하는지를 검사하고 주유펌프를 추가할 것인가 , 제거할 것인가를 결정하려고 한다 .

- 실제로 효율성 검사를 하지 않고 주유펌프를 추가 / 제거하는 것은 비용 문제가 있기 때문에 우리는 이 결정을 위해 시뮬레이션을 한다 .

예제 : A Simple Gas Station Model

- 시뮬레이션에서 사용된 가정
  - 시뮬레이션 시간은 고객 1000 명을 기준으로 한다 .
  - 이 주유소에 도착하는 고객들의 시간 간격은 2 분에 8 분 사이로 uniform 하게 분포되어 있다 .
  - 고객 서비스 시간은 5 분에 15 분 사이로 uniform 하게 분포되어 있다 .

예제 : A Simple Gas Station Model

PREAMBLE

PROCESSES INCLUDE GENERATOR AND CUSTOMER

RESOURCES INCLUDE ATTENDANT

ACCUMULATE AVG.QUEUE.LENGTH AS THE AVERAGE

AND MAX.QUEUE.LENGTH AS THE MAXIMUM

OF N.Q.ATTENDANT

ACCUMULATE UTILIZATION AS THE AVERAGE OF N.X.ATTENDANT

END

예제 : A Simple Gas Station Model


```
MAIN
    CREATE EVERY ATTENDANT(1)
    LET U.ATTENDANT(1) = 2
    ACTIVATE A GENERATOR NOW
    START SIMULATION
    PRINT 4 LINES WITH AVG.QUEUE.LENGTH(1),
    MAX.QUEUE.LENGTH(1),
      AND UTILIZATION(1) * 100. / 2 THUS

       SIMPLE GAS STATION MODEL WITH 2 ATTENDANTS
       AVERAGE CUSTOMER QUEUE LENGTH IS      *.***
       MAXIMUM CUSTOMER QUEUE LENGTH IS      *
       THE ATTENDANTS WERE BUSY **.** PER CENT OF THE TIME.
END
```

예제 : A Simple Gas Station Model


```
PROCESS GENERATOR
    FOR I = 1 TO 1000,
    DO
        ACTIVATE A CUSTOMER NOW
        WAIT UNIFORM.F(2.0,8.0,1) MINUTES
    LOOP
END

PROCESS CUSTOMER
    REQUEST 1 ATTENDANT(1)
    WORK UNIFORM.F(5.0,15.0,2) MINUTES
    RELINGQUISH 1 ATTENDANT(1)
END
```

3. Modeling Individual Objects

3.1. Attribute Concept

- 프로세스나 자원(resource)은 속성이 주어질 수 있다.
- Resources
- Every Pump has a Grade
- Create Every Pump (3)

1 2 3

U.Pump

N.X.Pump

N.Q.Pump

Grade

3. Modeling Individual Objects

3.2 Variables

- 변수는 전역 또는 지역변수(default)로 될 수 있다. 전역변수는 Preamble 에 정의된다.
- 모든 변수는 mode 를 가지고 있다.(integer, real, alpha, text)
- Background mode 는 real 이며 다음의 문장에 의해 변경된다.
  - NORMALLY, MODE IS mode

- 변수의 길이는 전형적으로 80 자 이내이며 문자, 숫자, 마침표의 조합이다.

  올바른 예) ABC, NO.OF.CUSTOMERS, 5.12.38, ABC...

  틀린 예) 567, 2+2, 5.12

3. Modeling Individual Objects

3.3 Program Control Structures

IF Statement

```
IF STATUS = BUSY
    ADD 1 TO BACK.LOG
ALWAYS
```

LOOPING

```
FOR EACH resource
```

is equivalent to

```
FOR resource = 1 TO N.resource
```

```
FOR EACH resource CALLED name
```

is equivalent to

```
FOR name = 1 TO N.RESOURCE
```

```
FOR EACH PUMP,
    WITH GRADE(PUMP) = DESIRED.GRADE
    AND RESERVE(PUMP) >= 10.0,
FIND THE FIRST CASE
```

3. Modeling Individual Objects

3.4 The Representation of Time

- 시뮬레이션 시계(clock)는 시스템에서 정의한 Real 변수 TIME.V 에 의해 표현되며 초기에 0의 값을 가진다.
- 시간의 기본 값 단위는 일(day)이다.
- HOURS.V = 24
- MINUTES.V = 60

- 시스템 설계자는 이 기본 값을 원하는 단위로 변경할 수 있다. 컴퓨터 시스템을 생각해 보면, DAYS를 SECONDS로 HOURS를 MILLISECONDS, MINUTES를 MICROSECONDS로 바꿀 수 있다.

3. Modeling Individual Objects


```
PREAMBLE
    DEFINE .seconds TO MEAN days
    DEFINE .milliseconds TO MEAN hours
    DEFINE .microseconds TO MEAN minutes
END

MAIN
    LET HOURS.V = 1000
    LET MINUTES.V = 1000
END
```

예제 : A Bank with a Separate Queue for Each Teller

- 일반적인 은행의 경우에, 고객은 은행에 도착해서 바로 이용 가능한 은행원에게 서비스를 받고 은행을 떠나게 된다. 그러나 만약 모든 은행원들이 이용가능하지 않다면 고객은 가장 짧은 줄에 줄을 서게 될 것이다.
- 이러한 은행을 시뮬레이션 해 보자. 성능 측정의 요소는 큐(대기열)의 평균, 최대 길이, 은행원 각각의 이용률, 그리고 전체 고객의 평균 대기 시간이다.
- 이 시뮬레이션에서 사용되는 파라미터는 모델 설계자가 직접 입력한다.
- 은행원의 수 (Teller), 고객 도착 시간(λ: 지수 분포를 따라 도착한다), 은행의 영업 시간

예제 : A Bank with a Separate Queue for Each Teller


```
PREAMBLE
    PROCESSES INCLUDE GENERATOR AND CUSTOMER
    RESOURCES INCLUDE TELLER

    DEFINE MEAN.INTERARRIVAL.TIME, MEAN.SERVICE.TIME,
    DAY.LENGTH AND WAITING.TIME AS REAL VARIABLES

    ACCUMULATE UTILIZATION AS THE AVERAGE OF N.X.TELLER
    ACCUMULATE AVG.QUEUE.LENGTH AS THE AVERAGE,
        MAX.QUEUE.LENGTH AS THE MAXIMUM OF N.Q.TELLER
    TALLY MEAN.WAITING.TIME AS THE MEAN OF WAITING.TIME
END
```

예제 : A Bank with a Separate Queue for Each Teller


```
MAIN
    READ N.TELLER, MEAN.INTERARRIVAL.TIME, MEAN.SERVICE.TIME,
      AND DAY.LENGTH
    CREATE EVERY TELLER
    FOR EACH TELLER,
      LETU.TELLER(TELLER) = 1

    PRINT 8 LINES WITH N.TELLER, MAEN.INTERARRIVAL.TIME,
      MEAN.SERVICE.TIME AND DAY.LENGTH THUS
SIMULATION OF A BANK WITH * TELLERS
  (EACH WITH A SEPARATE QUEUE)
CUSTOMERS ARRIVE ACCORDING TO AN EXPONENTIAL DISTRIBUTION
  OF INTER ARRIVAL TIMES WITH A MEAN OF *.** MINUTES.
SERVICE TIME IS ALSO EXPONENTIALLY DISTRIBUTED
  WITH A MEAN OF *.** MINUTES.
THE BANK DOORS ARE CLOSED AFTER *.** HOURS.
  (BUT ALL CUSTOMERS INSIDE ARE SERVED.)
```

예제 : A Bank with a Separate Queue for Each Teller


```
  ACTIVATE A GENERATE NOW
  START SIMULATION

  PRINT 6 LINES WITH TIME.V * HOURS.V,
       AND MEAN.WATING.TIME * HOURS.V * MINUTES.V THUS
THE LAST CUSTOMER LEFT THE BANK AT *.** HOURS.
THE AVERAGE CUSTOMER DELAY WAS *.** MINUTES.

TELLER          UTILIZATION              QUEUE LENGTH
                                 AVERAGE      MAXIMUM

FOR EACH TELLER,
  PRINT 1 LINE WITH TELLER, UTILIZATION(TELLER),
    AVG.QUEUE.LENGTH(TELLER), MAX.QUEUE.LENGTH(TELLER) THUS
  *              *.**               *.**              *
END
```

예제 : A Bank with a Separate Queue for Each Teller


```
PROCESS GENERATOR
    DEFINE ARRIVAL.TIME AS A REAL VARIABLE
    LET TIME.TO.CLOSE = DAY.LENGTH / HOURS.V

     UNTIL TIME.V >= TIME.TO.CLOSE,
    DO
      ACTIVATE A CUSTOMER NOW
      WAIT EXPONENTIAL.F(MEAN.INTERARRIVAL.TIME,1) MINUTES
    LOOP
END
```

예제 : A Bank with a Separate Queue for Each Teller


```
PROCESS CUSTOMER
    DEFINE ARRIVAL.TIME AS A REAL VARIABLE
    DEFINE MY.CHOICE AS A INTEGER VARIABLE
    LET ARRIVAL.TIME = TIME.V
    FOR EACH TRELLER, WITH N.X.TELLER(TELLER) = 0,
      FIND THE FIRST CASE
     IF FOUND,
      LET MY.CHOICE = TELLER
    ELSE
      FOR EACH TELLER,
            COMPUTE MY.CHOICE AS THE MINIMUM(TELLER)
              OF N.Q.TELLER(TELLER)
    ALWAYS
    REQUEST 1 TELLER(MY.CHOICE)
    LET WAITING.TIME = TIME.V - ARRIVAL.TIME
    WORK EXPONENTIAL.F(MEAN.SERVICE.TIME,2) MINUTES
     RELINQUISH 1 TELLER(MY.CHOICE)
END
```

예제 : A Bank with a Separate Queue for Each Teller

[예제의 OUTPUT]

SIMULATION OF A BANK WITH 2 TELLERS
(EACH WITH A SEPARATE QUEUE)
CUSTOMERS ARRIVE ACCORDING TO AN EXPONENTIAL DISTRIBUTION
OF INTER ARRIVAL TIMES WITH A MEAN OF 5.00 MINUTES.
SERVICE TIME IS ALSO EXPONENTIALLY DISTRIBUTED
WITH A MEAN OF 10.00 MINUTES.
THE BANK DOORS ARE CLOSED AFTER 8.00 HOURS.
(BUT ALL CUSTOMERS INSIDE ARE SERVED.)
THE LAST CUSTOMER LEFT THE BANK AT *.** HOURS.
THE AVERAGE CUSTOMER DELAY WAS *.** MINUTES.

TELLER	UTILIZATION	QUEUE LENGTH	
		AVERAGE	MAXIMUM
1	.97	1.73	6
2	.91	2.06	7