Πανεπιστήμιο Αθηνών

Τμήμα Πληροφορικής

Μεταπτυχιακό Πληροφορικής

Προσομοίωση

Δημοσθένης Αναγνωστόπουλος

Επ. Καθηγητής

Χαροκόπειο Πανεπιστήμιο

Αντικείμενο

Πως πλάθουμε το υπολογιστικό μοντέλο ενός (μη) υπαρκτού κόσμου, ώστε μέσω αυτού να τον μελετήσουμε με τρόπο που να εξάγονται ενδιαφέροντα, μη αμφισβητούμενα συμπεράσματα

Περιεχόμενα (1)

- Βασικές έννοιες προσομοίωσης
- Προσομοίωση διακριτών συμβάντων και συνεχής προσομοίωση
- Μοντελοποίηση και όψεις μοντελοποίησης
- Στάδια εκπόνησης μελέτης προσομοίωσης
- Πειραματισμός (experimentation)
- Μοντελοποίηση εισόδου
- Δημιουργία τυχαίων αριθμών
- Μετατροπή αριθμού σε δείγμα κατανομής
- Ανάλυση εξόδου
- Αποτίμηση και επαλήθευση

Περιεχόμενα (2)

- Αντικειμενοστραφής μοντελοποίηση
- Γλώσσα προσομοίωσης Modsim
- Εργαλεία συνεχούς προσομοίωσης
- Σχεδίαση και υλοποίηση μοντέλων
- Σχεδίαση και υλοποίηση μελέτης προσομοίωσης

Ενδεικτική βιβλιογραφία

Βασικό σύγγραμμα: Simulation modelling and analysis, Law and Kelton, McGraw-Hill

Simulation modelling with Pascal, Davis and O'Keefe, McGraw-Hill

Theory of Modeling and Simulation, 2nd Edition, 2000, B. Zeigler, Academic Press

Simulation Model Design and Execution, P. Fishwick, Prentice Hall

Δημοσιευμένα άρθρα

Βασικές Έννοιες Προσομοίωσης Περιγραφή

- Μέθοδος μελέτης συστημάτων
  - υπολογιστική, ποσοτική μέθοδος
- Σύστημα - διεργασία υπό μελέτη, υιοθέτηση υποθέσεων, σχεδίαση και κατασκευή μοντέλου (μαθηματικές-λογικές συσχετίσεις)

> Αναλυτική λύση (μαθηματικό μοντέλο, για χαμηλής πολυπλοκότητας σύστημα)

  > παράδειγμα: $u=\gamma t=Ft/m$

  (σώμα μάζας m, ασκούμε δύναμη F, καθόλου τριβή)

> Υπολογιστικό μοντέλο (προσομοίωση)

  - εκτέλεση μοντέλου
  - συλλογή και επεξεργασία αποτελεσμάτων
  - εξαγωγή συμπερασμάτων

- Σκοπός: Εκτίμηση συμπεριφοράς συστήματος υπό συνήθεις και ασυνήθεις συνθήκες
- Τελευταία (και καλή) επιλογή!

Βασικές Έννοιες Προσομοίωσης Εφαρμογές

- Συστήματα παραγωγής
- Υπολογιστικά συστήματα (υλικό, λογισμικό)
- Συστήματα μεταφορών
- Οργάνωση και σχεδίαση υπηρεσιών (νοσοκομεία, ταχυδρομεία κτλ)
- Οικονομικά και χρηματοοικονομικά συστήματα
- Στρατιωτικά συστήματα

τομές με επιχειρησιακή έρευνα (operations research) και επιστήμη διοίκησης (management science)

Βασικές Έννοιες Προσομοίωσης

- Σύστημα:
  σύνολο οντοτήτων (entities) που αλληλεπιδρούν μεταξύ τους
- Κατάσταση (state):
  σύνολο μεταβλητών που περιγράφουν την κατάσταση του συστήματος σε κάθε στιγμή
  - διακριτή ή συνεχής μεταβολή
- Μεταβλητές εισόδου
- Μεταβλητές εξόδου
- Μεταβλητές απόφασης
- Τρόποι μελέτης συστημάτων

Βασικές Έννοιες Προσομοίωσης
Κατηγορίες Μοντέλων

- Στατικά και δυναμικά μοντέλα
  - κριτήριο: μεταβολή κατά τη διάρκεια του χρόνου
- Στοχαστικά και ντετερμινιστικά μοντέλα
  - κριτήριο: τμήματα μοντέλου που εμφανίζουν πιθανοθεωρητική συμπεριφορά

* *Προσομοίωση: ποσοτική και υπολογιστική μέθοδος που χρησιμοποιεί δυναμικά και στοχαστικά μοντέλα*

- Διακριτά και συνεχή μοντέλα
  - κριτήριο: μεταβολή κατάστασης μοντέλου σε διακριτές ή συνεχείς χρονικές στιγμές.

* *Συνεχής προσομοίωση*

* *Προσομοίωση διακριτών συμβάντων (discrete event simulation, computer simulation)*

Βασικές Έννοιες Προσομοίωσης
Συστήματα και Μοντέλα

- Διακριτά και συνεχή μοντέλα
  - κριτήριο: μεταβολή κατάστασης μοντέλου σε διακριτές ή συνεχείς χρονικές στιγμές.
- Υφίστανται αντίστοιχα:
  διακριτά και συνεχή συστήματα
- Αντιστοίχιση μεταξύ κατηγοριών μοντέλων και συστημάτων δεν είναι μονοσήμαντη
  - Παράδειγμα: τρόπος μελέτης κίνησης οχημάτων

* Συνεχής προσομοίωση: εξισώσεις ροής
* Προσομοίωση διακριτών συμβάντων: μεταβολή μοντέλου σε διακριτές χρονικές στιγμές

Προσομοίωση Διακριτών Συμβάντων Μηχανισμοί Εξέλιξης Χρόνου

- Ρολόι προσομοίωσης (μεταβλητή), χρόνος προσομοίωσης
- Μονάδες χρόνος (καθαρός αριθμός)
- Δεν υφίσταται συσχέτιση μεταξύ χρόνου προσομοίωσης και πραγματικού χρόνου
- Μηχανισμοί εξέλιξης χρόνου
  - επόμενο συμβάν
  - προκαθορισμένη χρονική εξέλιξη
- Παράδειγμα: μοντελοποίηση οδικού κόμβου

> μοντέλο επόμενου συμβάντος

> > άφιξη, αναχώρηση, εκκίνηση πράσινου και κόκκινου χρώματος στο φανάρι

> μοντέλο προκαθορισμένης χρονικής εξέλιξης: κυψελωτό αυτόματο

> > νέο συμβάν κάθε 0+ν*περίοδος εξέλιξης

Προσομοίωση Διακριτών Συμβάντων
Εξέλιξη με Μηχανισμό Επόμενου Συμβάντος

- Βήματα:
    1. θέτουμε αρχικό χρόνο ίσο με το 0
    2. χρονοπρογραμματισμός συμβάντων
    3. αύξηση του ρολογιού προσομοίωσης στον χρόνο του επικείμενου συμβάντος
    4. εκτέλεση συμβάντος και χρονοπρογραμματισμός νέων συμβάντων
    5. αν έχει ολοκληρωθεί το πείραμα, τερματισμός, αλλιώς επιστροφή στο βήμα 3

Προσομοίωση Διακριτών Συμβάντων Όψεις Μοντελοποίησης

- Τρόπος σύλληψης λειτουργίας συστήματος

> Προσέγγιση συμβάντων (event approach)

> Προσέγγιση τριών φάσεων (three-phase approach)

  - και οι δύο προβαίνουν στην αναπαράσταση μέσω του χρονοπρογραμματισμού συμβάντων
  - επιπρόσθετα: υπό συνθήκη συμβάντα
  - διαφορετική κωδικοποίηση (υλοποίηση) λειτουργικότητας των συμβάντων

> Προσέγγιση διεργασιών (process approach)

  - περιγραφή δραστηριότητας οντοτήτων κατά τη λειτουργία του συστήματος
  - παράδειγμα: *aircraft object* καλεί μέθοδο ανεφοδιασμού *tank object*
  - *Τελικός χρόνος = αρχικός χρόνος + χρόνος μεθόδου (χρόνος προσομοίωσης)*

Προσομοίωση Διακριτών Συμβάντων
Παράδειγμα: ένας εξυπηρέτης/ μία ουρά

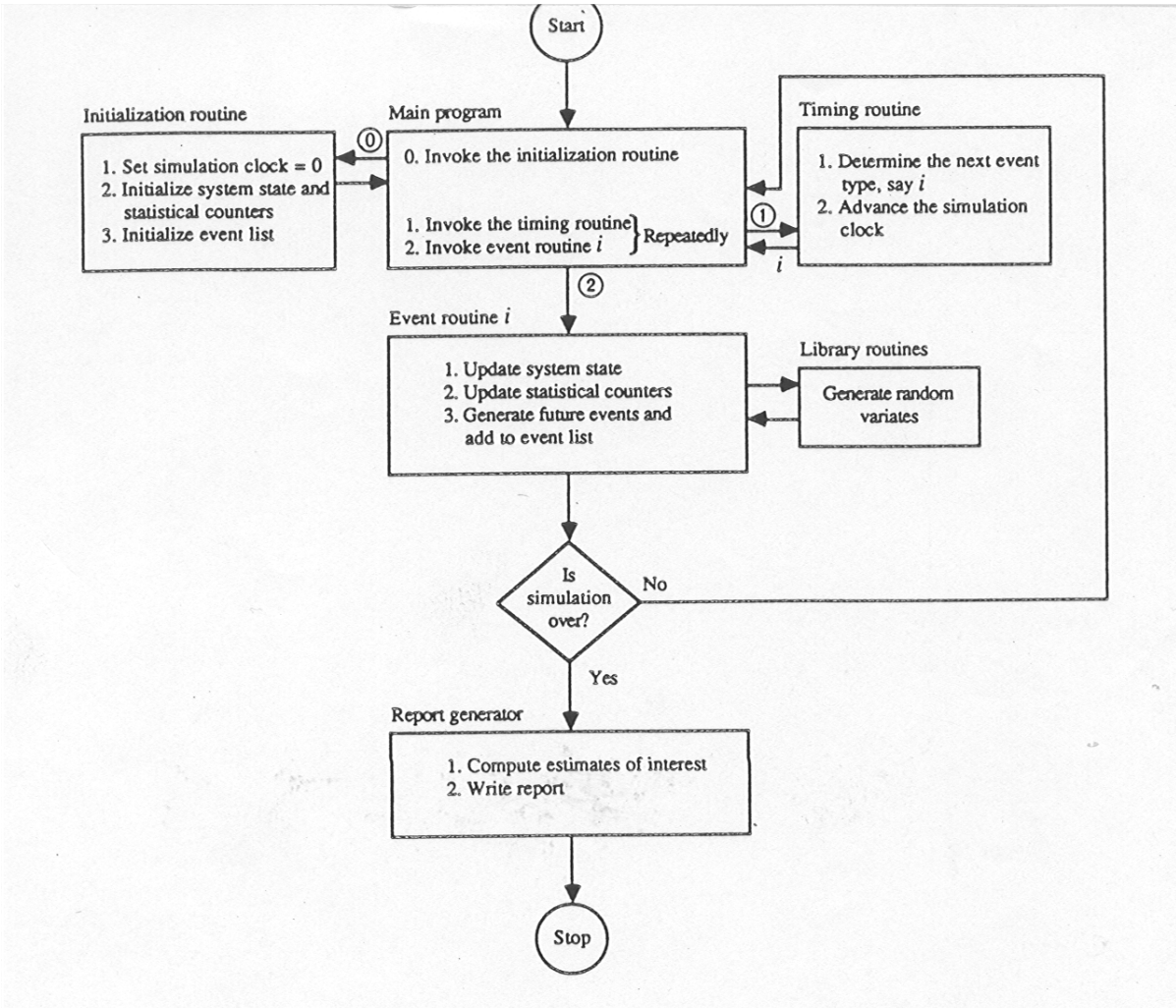
- Χρόνος μεταξύ αφίξεων (A): *A1, A2, A3, .., An*
  - IID: independent, identically distributed
- Χρόνοι εξυπηρέτησης (S): *S1, S2, S3, …, Sn*
- FIFO οργάνωση ουράς, αρχικά άδειο σύστημα
- Αποτελέσματα προσομοίωσης:
  - καθυστέρηση στην ουρά (D): *D1, .., Dn*
  - μέσος αριθμός πελατών στην ουρά (Q), *Pi* το ποσοστό του χρόνου που είναι *i* πελάτες στην ουρά.
  - αξιοποίηση εξυπηρέτη (U), *Pi* το ποσοστό του χρόνου που είναι *busy* ή *idle* ο εξυπηρέτης

(εκτ.) $$d(n) = \sum_{i=1}^{n} Di \Big/ n$$

(εκτ.) $$q(n) = \sum_{i=1}^{\infty} iPi = \sum_{i=1}^{\infty} iTi/T = \frac{\int_o^T Q(t)dt}{T}$$

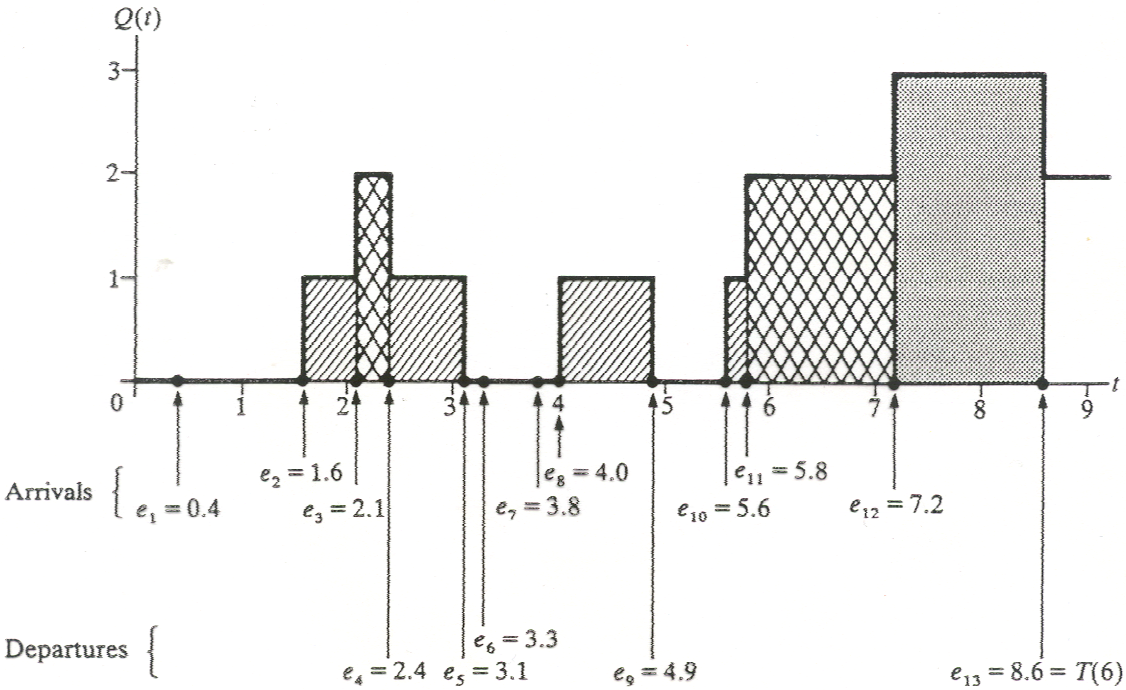
(εκτ.) $$u(n) = \sum_{i=0}^{1} iPi = \sum_{i=0}^{1} iTi/T = \frac{\int_o^T B(t)dt}{T}$$

Προσέγγιση Συμβάντων
Εργασίες Εκτέλεσης Πειράματος

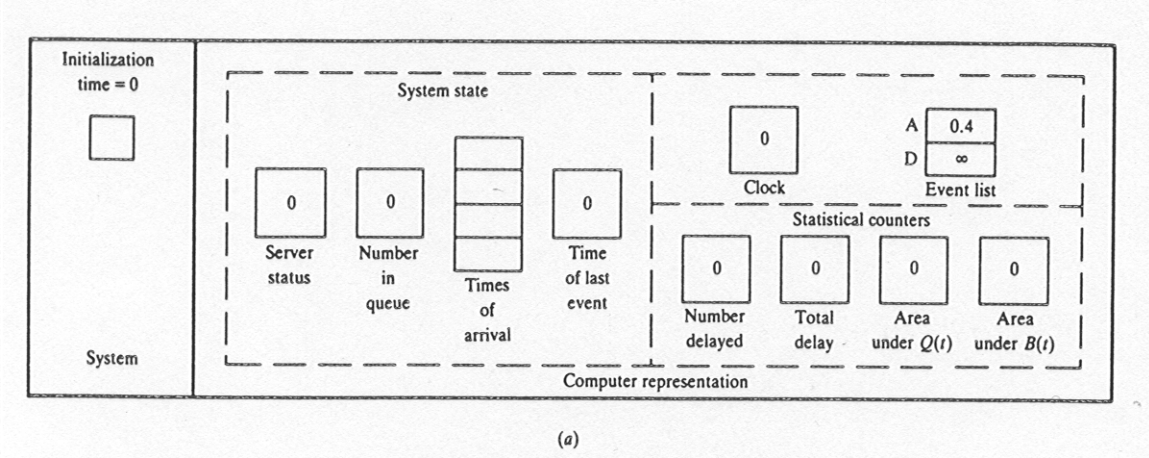


Εκτέλεση Πειράματος

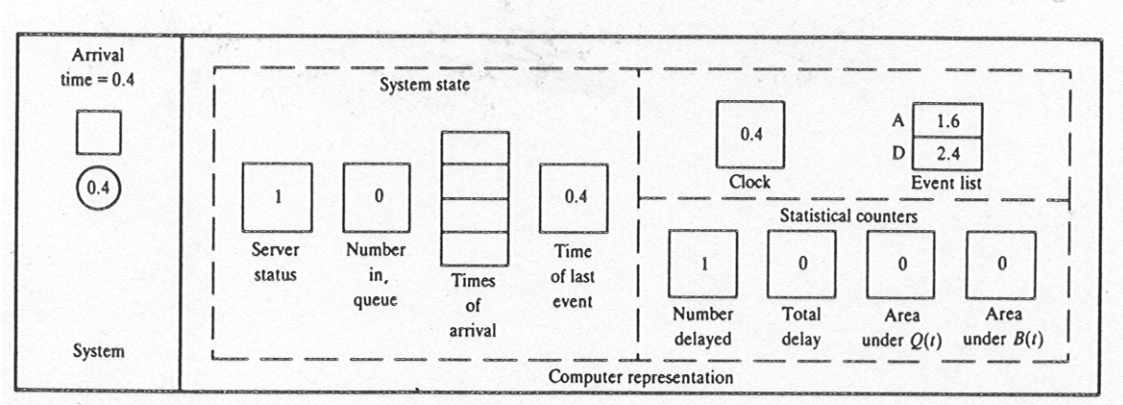
Παράδειγμα: Ένας Εξυπηρέτης/Μία Ουρά



Εκτέλεση Πειράματος
(διαδοχικές καταστάσεις-1)

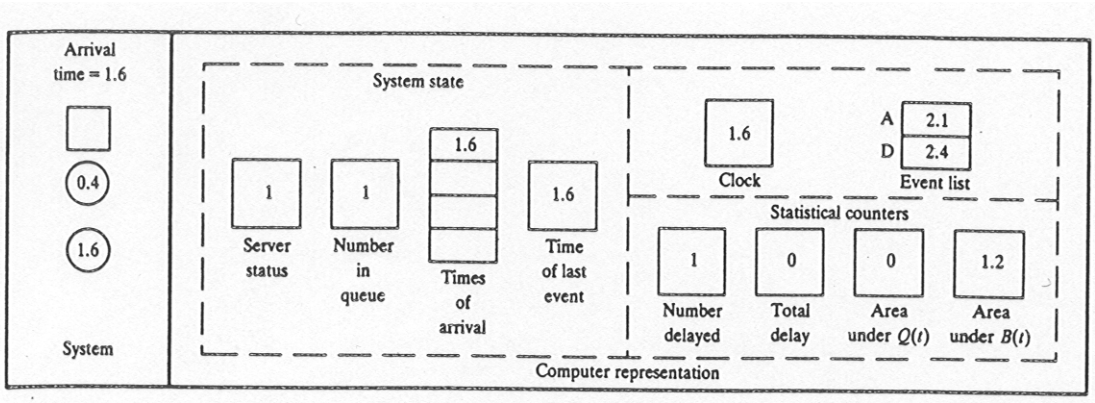


(a)

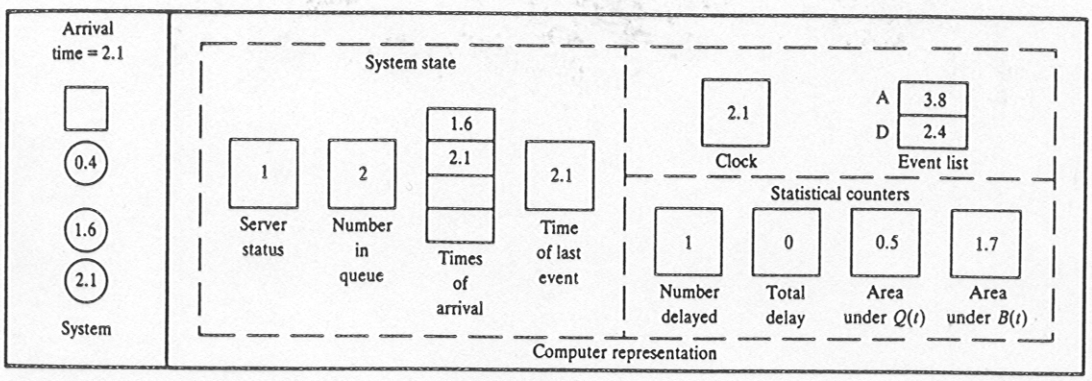


(b)

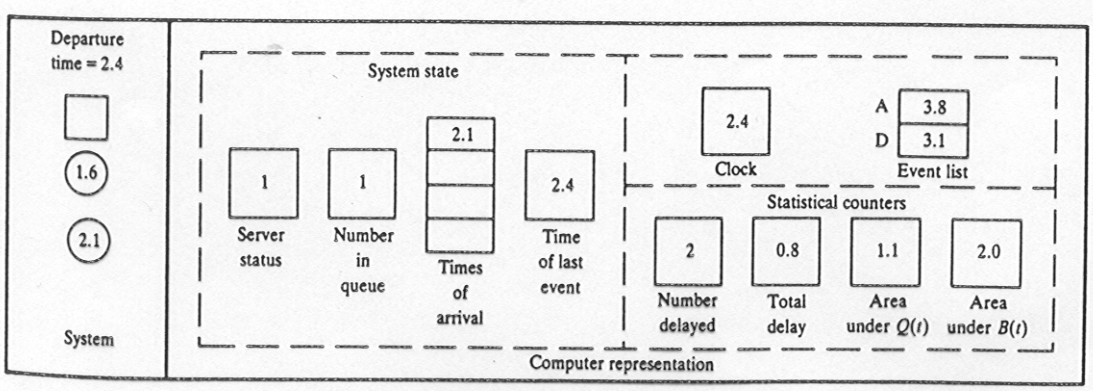
Εκτέλεση Πειράματος
(διαδοχικές καταστάσεις-2)



(c)

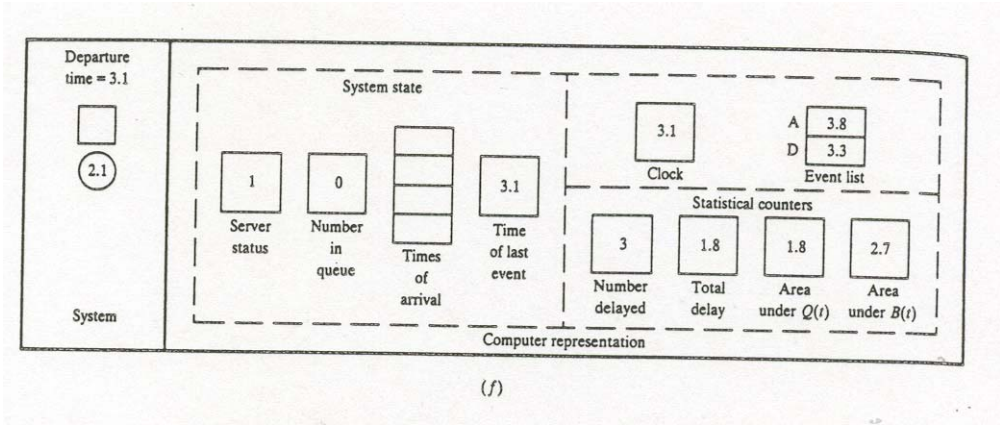


(d)

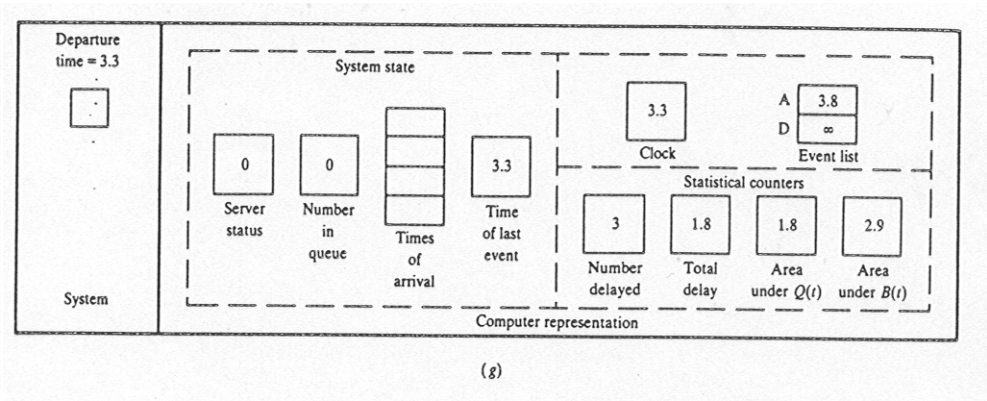


(e)

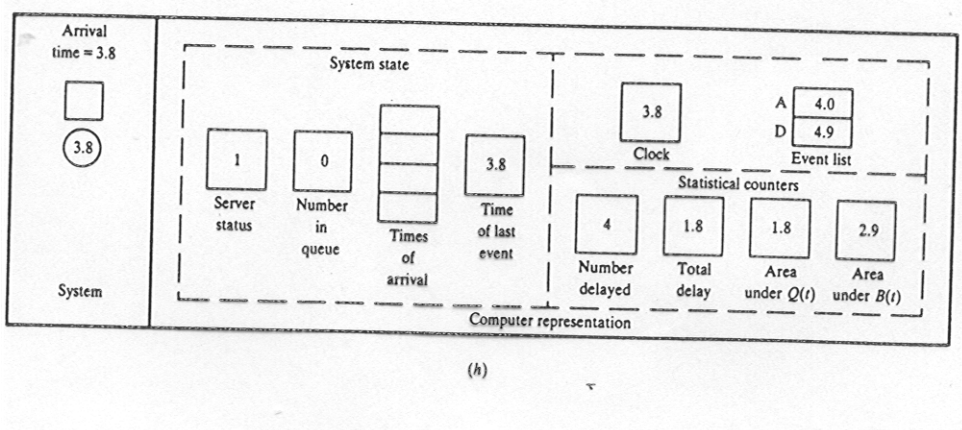
Εκτέλεση Πειράματος
(διαδοχικές καταστάσεις-3)



(f)

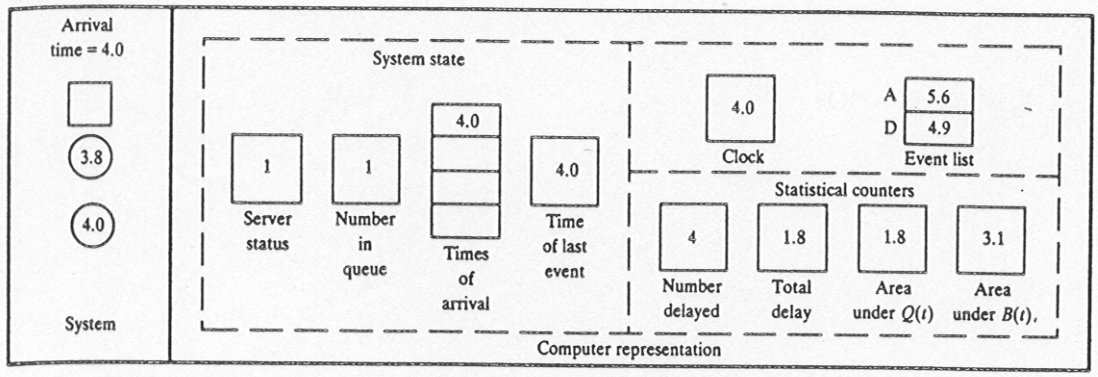


(g)

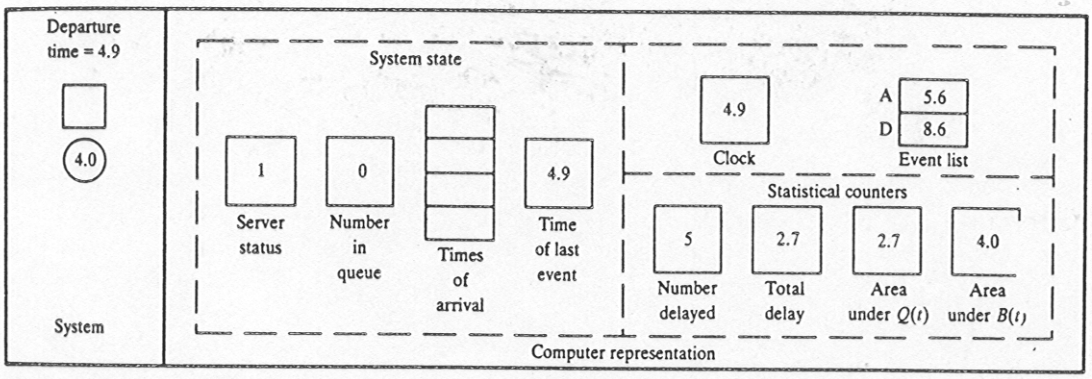


(h)

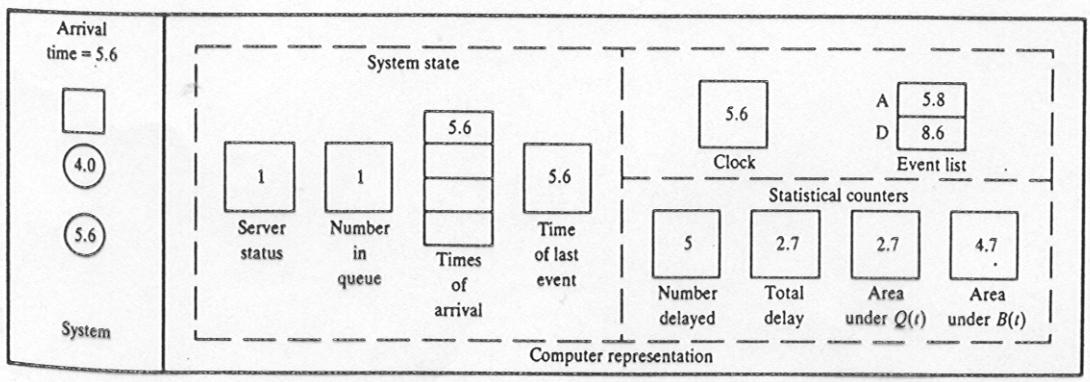
Εκτέλεση Πειράματος (διαδοχικές καταστάσεις-4)



(i)

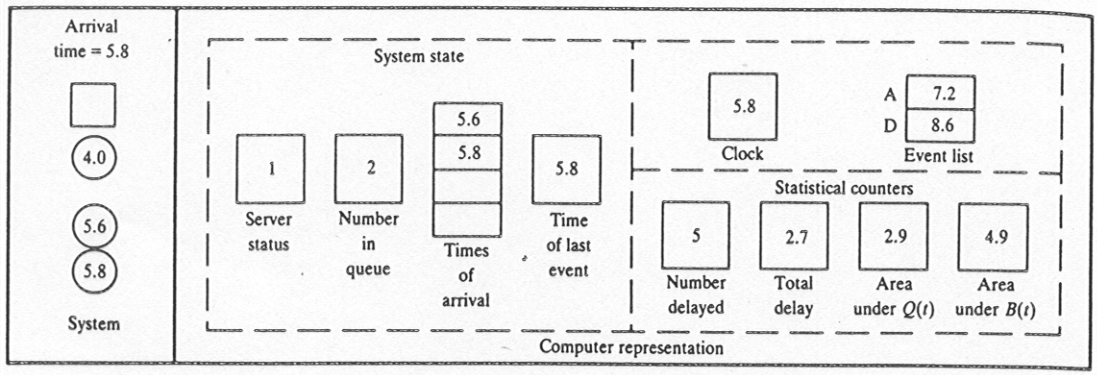


(j)

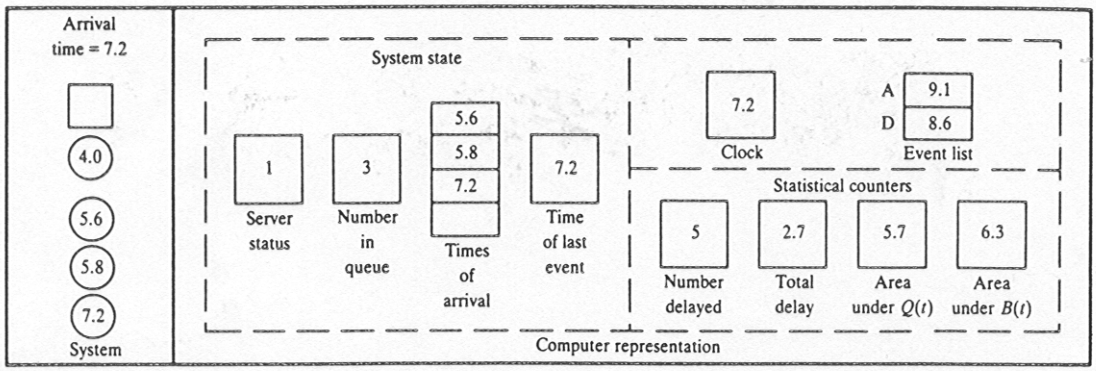


(k)

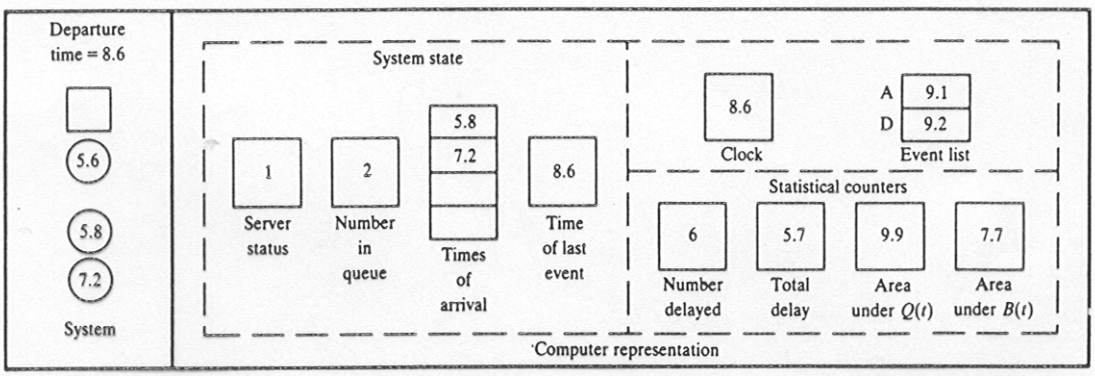
Εκτέλεση Πειράματος (διαδοχικές καταστάσεις-5)



(*l*)



(*m*)

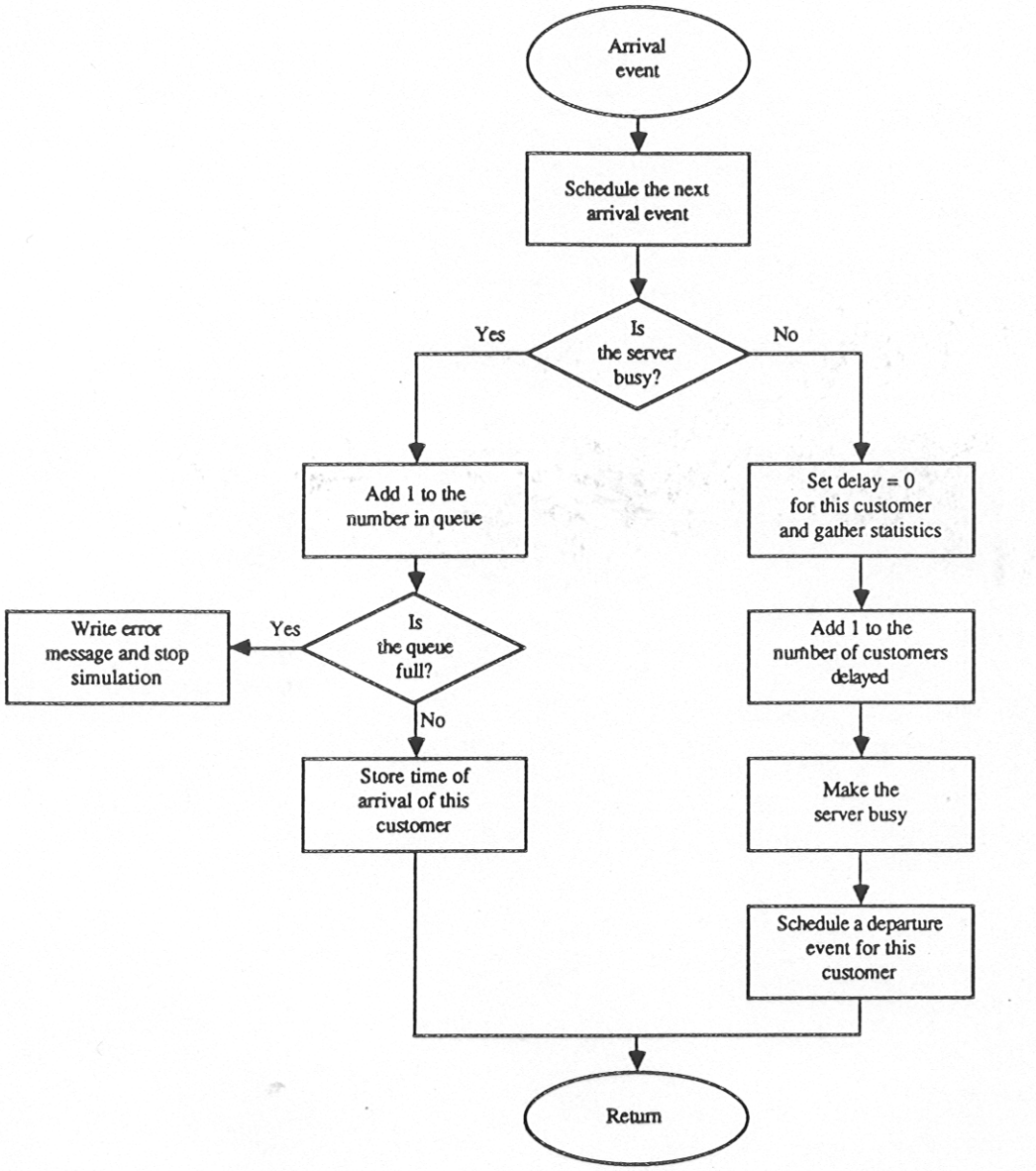


(*n*)

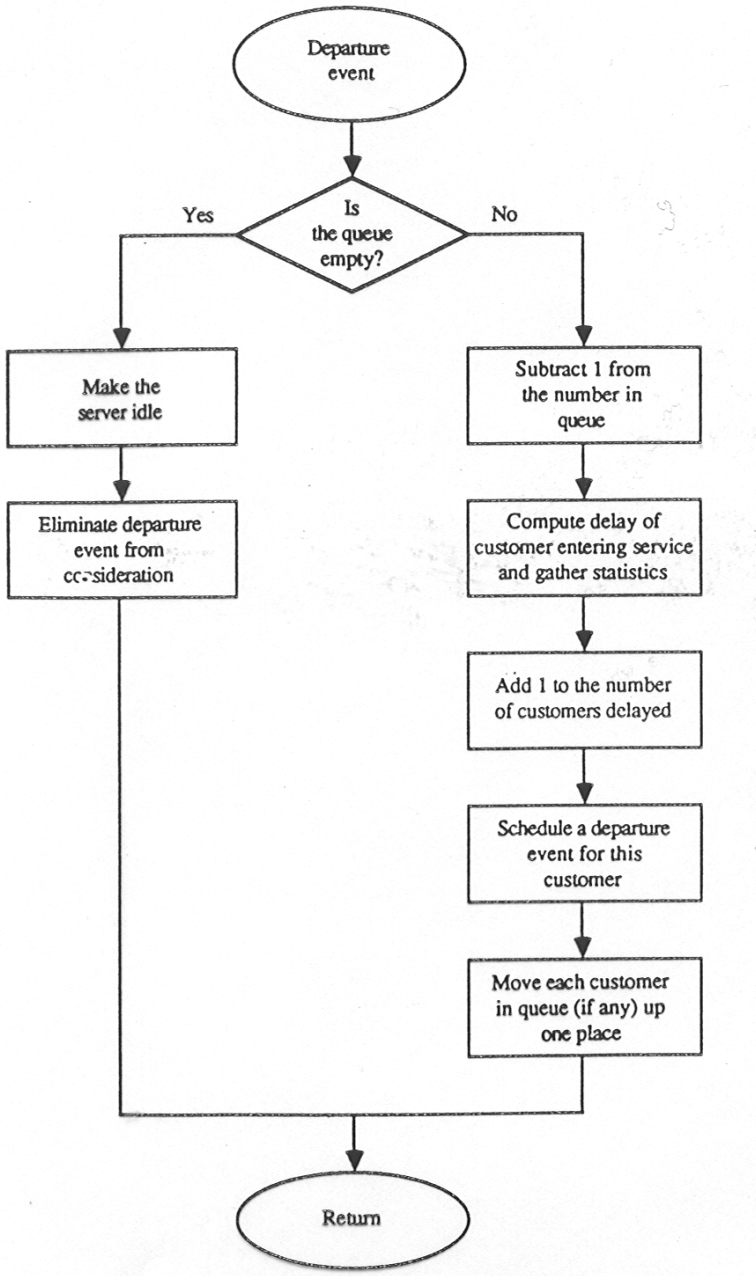
Παράδειγμα: Ένας Εξυπηρέτης/Μία Ουρά Συζήτηση Πειράματος

- Πότε προχωρά ο χρόνος προσομοίωσης;
- Διάρκεια συμβάντος
- Χρονοπρογραμματισμένα συμβάντα για την ίδια χρονική στιγμή
  - υιοθέτηση κανόνα απόφασης
- Τερματισμός πειράματος
  - επίτευξη σκοπού (π.χ. εξυπηρέτηση 1000 πελατών)
    - περιλαμβάνονται μόνο οι οντότητες που έχουν ολοκληρώσει τον κύκλο δραστηριότητας τους
  - άνω χρονικό όριο
  - συνθήκη τερματισμού
- Μοντελοποίηση με την προσέγγιση συμβάντων
  - άφιξη πελάτη
  - αναχώρηση πελάτη

Παράδειγμα: Ένας Εξυπηρέτης/Μία Ουρά
Διάγραμμα ροής: arrival event



Παράδειγμα: Ένας Εξυπηρέτης/Μία Ουρά
Διάγραμμα ροής: departure event



```
/* External definitions for single-server queueing system. */

#include <stdio.h>
#include <math.h>
#include "rand.h"     /* Header file for random-number generator. */

#define Q_LIMIT 100  /* Limit on queue length. */
#define BUSY      1  /* Mnemonics for server's being busy */
#define IDLE      0  /* and idle. */

int   next_event_type, num_custs_delayed, num_delays_required,
      num_events, num_in_q, server_status;
float area_num_in_q, area_server_status, mean_interarrival,
      mean_service, time, time_arrival[Q_LIMIT + 1],
      time_last_event, time_next_event[3], total_of_delays;
FILE  *infile, *outfile;

void  initialize(void);
void  timing(void);
void  arrive(void);
void  depart(void);
void  report(void);
void  update_time_avg_stats(void);
float expon(float mean);

main()  /* Main function. */
{
    /* Open input and output files. */

    infile  = fopen("mm1.in",  "r");
    outfile = fopen("mm1.out", "w");

    /* Specify the number of events for the timing function. */

    num_events = 2;

    /* Read input parameters. */

    fscanf(infile, "%f %f %d", &mean_interarrival, &mean_service,
           &num_delays_required);

    /* Write report heading and input parameters. */

    fprintf(outfile, "Single-server queueing system\n\n");
    fprintf(outfile, "Mean interarrival time%11.3f minutes\n\n",
            mean_interarrival);
    fprintf(outfile, "Mean service time%16.3f minutes\n\n",
            mean_service);
    fprintf(outfile, "Number of customers%14d\n\n",
            num_delays_required);

    /* Initialize the simulation. */

    initialize();

    /* Run the simulation while more delays are still needed. */

    while (num_custs_delayed < num_delays_required) {

        /* Determine the next event. */

        timing();

        /* Update time-average statistical accumulators. */

        update_time_avg_stats();

        /* Invoke the appropriate event function. */

        switch (next_event_type) {
            case 1:
                arrive();
                break;
            case 2:
                depart();
                break;
        }
    }

    /* Invoke the report generator and end the simulation. */

    report();

    fclose(infile);
    fclose(outfile);

    return 0;
}
```

```
void initialize(void)  /* Initialization function. */
{
    /* Initialize the simulation clock. */

    time = 0.0;

    /* Initialize the state variables. */

    server_status   = IDLE;
    num_in_q        = 0;
    time_last_event = 0.0;

    /* Initialize the statistical counters. */

    num_custs_delayed  = 0;
    total_of_delays    = 0.0;
    area_num_in_q      = 0.0;
    area_server_status = 0.0;

    /* Initialize event list.  Since no customers are present, the
       departure (service completion) event is eliminated from
       consideration. */

    time_next_event[1] = time + expon(mean_interarrival);
    time_next_event[2] = 1.0e+30;
```

```
void timing(void)  /* Timing function. */
{
    int   i;
    float min_time_next_event = 1.0e+29;

    next_event_type = 0;

    /* Determine the event type of the next event to occur. */

    for (i = 1; i <= num_events; ++i) {
        if (time_next_event[i] < min_time_next_event) {
            min_time_next_event = time_next_event[i];
            next_event_type     = i;
        }
    }

    /* Check to see whether the event list is empty. */

    if (next_event_type == 0) {

        /* The event list is empty, so stop the simulation. */

        fprintf(outfile, "\nEvent list empty at time %f", time);
        exit(1);
    }

    /* The event list is not empty, so advance the simulation clock. */

    time = min_time_next_event;
```

```
void arrive(void)  /* Arrival event function. */
{
    float delay;

    /* Schedule next arrival. */

    time_next_event[1] = time + expon(mean_interarrival);

    /* Check to see whether server is busy. */

    if (server_status == BUSY) {

        /* Server is busy, so increment number of customers in queue.
           */

        ++num_in_q;

        /* Check to see whether an overflow condition exists. */

        if (num_in_q > Q_LIMIT) {

            /* The queue has overflowed, so stop the simulation. */

            fprintf(outfile, "\nOverflow of the array time_arrival at");
            fprintf(outfile, " time %f", time);
            exit(2);
        }

        /* There is still room in the queue, so store the time of
           arrival of the arriving customer at the (new) end of
           time_arrival. */

        time_arrival[num_in_q] = time;
    }

    else {

        /* Server is idle, so arriving customer has a delay of zero.
           (The following two statements are for program clarity and do
           not affect the results of the simulation.) */

        delay            = 0.0;
        total_of_delays += delay;

        /* Increment the number of customers delayed, and make server
           busy. */

        ++num_custs_delayed;
        server_status = BUSY;

        /* Schedule a departure (service completion). */

        time_next_event[2] = time + expon(mean_service);
    }
}
```

```
void depart(void)  /* Departure event function. */
{
    int   i;
    float delay;

    /* Check to see whether the queue is empty. */

    if (num_in_q == 0) {

        /* The queue is empty so make the server idle and eliminate the
           departure (service completion) event from consideration. */

        server_status      = IDLE;
        time_next_event[2] = 1.0e+30;
    }

    else {

        /* The queue is nonempty, so decrement the number of customers
           in queue. */

        --num_in_q;

        /* Compute the delay of the customer who is beginning service
           and update the total delay accumulator. */

        delay            = time - time_arrival[1];
        total_of_delays += delay;

        /* Increment the number of customers delayed, and schedule
           departure. */

        ++num_custs_delayed;
        time_next_event[2] = time + expon(mean_service);

        /* Move each customer in queue (if any) up one place. */

        for (i = 1; i <= num_in_q; ++i)
            time_arrival[i] = time_arrival[i + 1];
    }
}

void report(void)  /* Report generator function. */
{
    /* Compute and write estimates of desired measures of performance.
       */

    fprintf(outfile, "\n\nAverage delay in queue%11.3f minutes\n\n",
            total_of_delays / num_custs_delayed);
    fprintf(outfile, "Average number in queue%10.3f\n\n",
            area_num_in_q / time);
    fprintf(outfile, "Server utilization%15.3f\n\n",
            area_server_status / time);
    fprintf(outfile, "Time simulation ended%12.3f", time);
}


float expon(float mean)  /* Exponential variate generation function.
                            */
{
    float u;

    /* Generate a U(0,1) random variate. */

    u = rand(1);

    /* Return an exponential random variate with mean "mean". */

    return -mean * log(u);
}
```

Παράδειγμα: Ένας Εξυπηρέτης/Μία Ουρά Αποτελέσματα Πειράματος

- Simulation results

mean interarrival: 1000min
mean service time: 0.500min
number of customers: 1000
avg delay in queue: 0.430min
avg number in queue: 0.418
server utilization: 0.460
time simulation ended: 1027.915

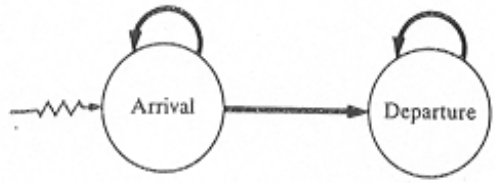
- Θέματα συζήτησης
  - χρόνος ολοκλήρωσης πειράματος
  - τυχαίοι αριθμοί και ρεύματα (streams)
  - σταθερή κατάσταση μεγεθών (steady state)
  - εκτίμηση *d(n)* αντί για *w(n)*
  - προσδιορισμός (εκτίμηση) *w(n)*

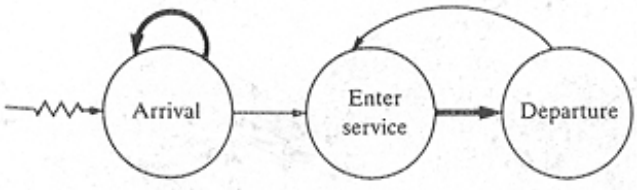
$$w(n) = d(n) + E(S)$$

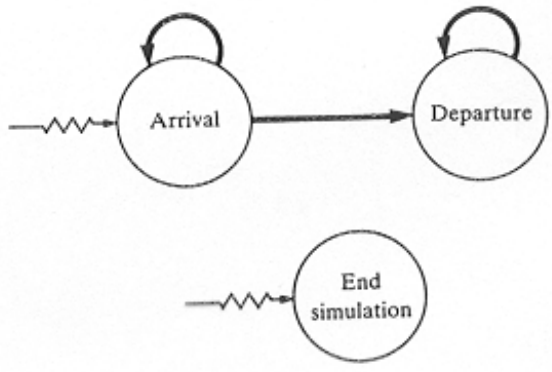
Προσομοίωση Διακριτών Συμβάντων
Προσδιορισμός συμβάντων και μεταβλητών (1)

- Μέθοδος γράφου συμβάντων (Schruben, Sargent)
  - προσέγγιση συμβάντων
  - συμβάντα -- κόμβοι
  - χρονοπρογραμματισμός συμβάντων -- κατευθυνόμενες συνδέσεις, παχύ βέλος
  - αρχικός χρονοπρογραμματισμός -- λεπτό βέλος με τεθλασμένη γραμμή
- Προσέγγιση τριών φάσεων
  - επιπρόσθετα τα υπό συνθήκη συμβάντα
  - χρονοπρογραμματισμός συμβάντων υπό συνθήκη χωρίς να παρέλθει χρόνος -- λεπτό βέλος
- Υψηλότερος βαθμός πολυπλοκότητας
  - κανόνες απλούστευσης (π.χ. απομάκρυνση κόμβων με είσοδο μόνο κανονικές γραμμές με λεπτά βέλη)

Προσδιορισμός Συμβάντων και Μεταβλητών Μέθοδος Γράφου – Παράδειγμα G/Gn/1







Προσομοίωση Διακριτών Συμβάντων
Προσδιορισμός συμβάντων και μεταβλητών (2)

- Ισχυρά συνδεδεμένα συνθετικά τμήματα (strongly connected components)
  - διάσχιση κόμβων σε κάθε συνθετικό τμήμα
  - πρώτος γράφος: 2 τμήματα
  - δεύτερος γράφος: 2 τμήματα (arrival, enter service-
  - departure
- Κανόνας αρχικοποίησης
  - κάθε ισχυρά συνδεδεμένο συνθετικό τμήμα που δεν έχει κόμβο που δέχεται εξωτερική σύνδεση εισόδου, πρέπει να έχει τουλάχιστον ένα αρχικά χρονοπρογραμματισμένο κόμβο - συμβάν
- Εναλλακτικά σχήματα αναπαράστασης δραστηριοτήτων
  - διάγραμμα δραστηριότητας (activity diagram)
- Συμβάντα (2 τύποι), πόροι, ουρές, κτλ.

Κατανεμημένη Προσομοίωση

- Εκτέλεση - τρέξιμο μοντέλου
  - σειριακή και κεντρικοποιημένη εκτέλεση, μοναδικός υπολογιστής
  - παράλληλη και κατανεμημένη προσομοίωση
- Κατανομή εργασιών σε πολλούς επεξεργαστές
- Σημαντικό πεδίο έρευνας (ACM/IEEE PADS)
- Εναλλακτικοί μέθοδοι κατανεμημένης προσομοίωσης

1. Εκχώρηση λειτουργιών υποστήριξης
2. Κατανομή μοντέλου
3. Μηχανισμός χρόνου-δίνης (time-warp simulation)

Κατανεμημένη Προσομοίωση
Εκχώρηση λειτουργιών υποστήριξης

- Λειτουργίες υποστήριξης
  - δημιουργία τυχαίων αριθμών, μετατροπή σε δείγμα κατανομής, διαχείριση λίστας συμβάντων, ανάλυση εξόδου, κτλ
- Σειριακή εκτέλεση
- Μικρότερος φόρτος κεντρικού επεξεργαστή
- Διάταξη επεξεργαστών τύπου επόπτη/εκτελεστή (master/slave)

Προσομοίωση Διακριτών Συμβάντων Κατανομή μοντέλου

- Εκχώρηση εκτέλεσης υπομοντέλων
- Ανεξάρτητη εκτέλεση
- Απαίτηση επικοινωνίας μεταξύ επεξεργαστών για την επίτευξη ορθής σειράς εκτέλεσης συμβάντων
- Αναμονή επεξεργαστή για τη λήψη μηνυμάτων
- Απουσία καθολικού ρολογιού και πλήρους λίστας συμβάντων
  - πλεονέκτημα
- Αντικατάσταση με ανταλλαγή μηνυμάτων (message passing)
  - Χρονική σφραγίδα (time stamp)
- Μειονέκτηματα

1. Εμφάνιση αδιεξόδων (deadlock)
   - π.χ. δύο επεξεργαστές περιμένουν ο ένας τον άλλο
   - απαίτηση για μέθοδο εντοπισμού και ανάκαμψης ή πρόληψης από αδιέξοδα
2. Χαμηλός βαθμός αξιοποίησης

Προσομοίωση Διακριτών Συμβάντων Μηχανισμός χρόνου-δίνης

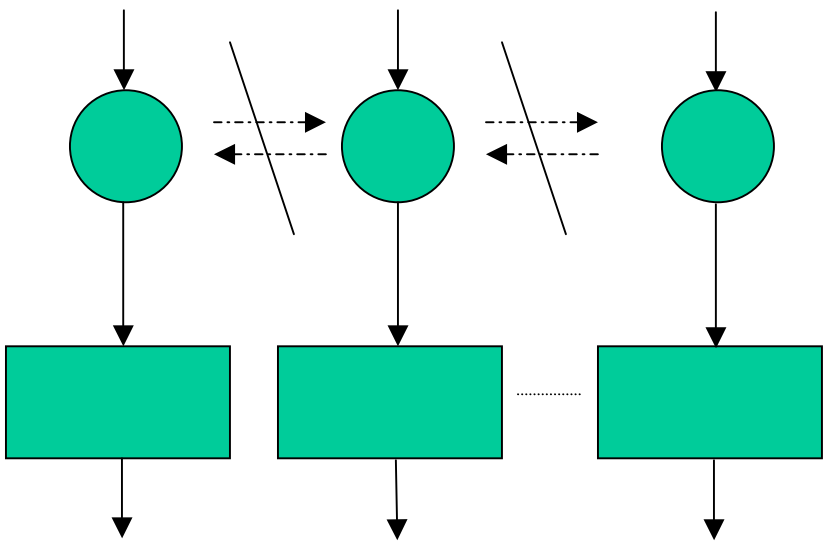
- Κατανομή μοντέλου, ανεξάρτητη εκτέλεση
- Επικοινωνίας μεταξύ επεξεργαστών

> Διαφορά: ο επεξεργαστής *δεν* σταματά αναμένοντας τη λήψη μηνυμάτων

- Μηνύματα με προγενέστερη χρονική σφραγίδα
  - πισωγύρισμα (rollback) στην χρονική αυτή στιγμή
  - αποστολή αντιμηνυμάτων (antimessages) για αντίστοιχη ενέργεια στα υπόλοιπα μοντέλα
- Ζητήματα
  - κατάλληλη κατανομή μοντέλου σχετίζεται με τη συχνότητα πισωγυρισμάτων
  - αποφυγή αδιεξόδων
  - υψηλός βαθμός αξιοποίησης επεξεργαστή
  - υψηλό κόστος πισωγυρίσματος (CPU, μνήμη)
  - αποθήκευση των απεσταλμένων μηνυμάτων για την πιθανότητα εκτέλεσης πισωγυρίσματος

Προσομοίωση Διακριτών Συμβάντων Απόδοση μηχανισμού χρόνου-δίνης

- Τυχαία απόδοση μηχανισμού χρόνου-δίνης
- Συνάρτηση βαθμού συσχέτισης μεταξύ υπομοντέλων
- Παράδειγμα: δίκτυο ουρών/επεξεργαστών χωρίς συχνή αλλαγή ουράς από τους πελάτες



Στάδια Εκπόνησης Μελέτης Προσομοίωσης

1. Ορισμός προβλήματος και σχεδιασμός της μελέτης
2. Συλλογή δεδομένων και ορισμός ιδεατού μοντέλου
3. Κατασκευή υπολογιστικού μοντέλου
4. Αποτίμηση
5. Εκτέλεση πιλοτικών πειραμάτων
6. Επαλήθευση
7. Σχεδιασμός πειραματισμού
8. Εκτέλεση πειραμάτων
9. Ανάλυση δεδομένων εξόδου
10. Τεκμηρίωση, παρουσίαση και υλοποίηση των αποτελεσμάτων

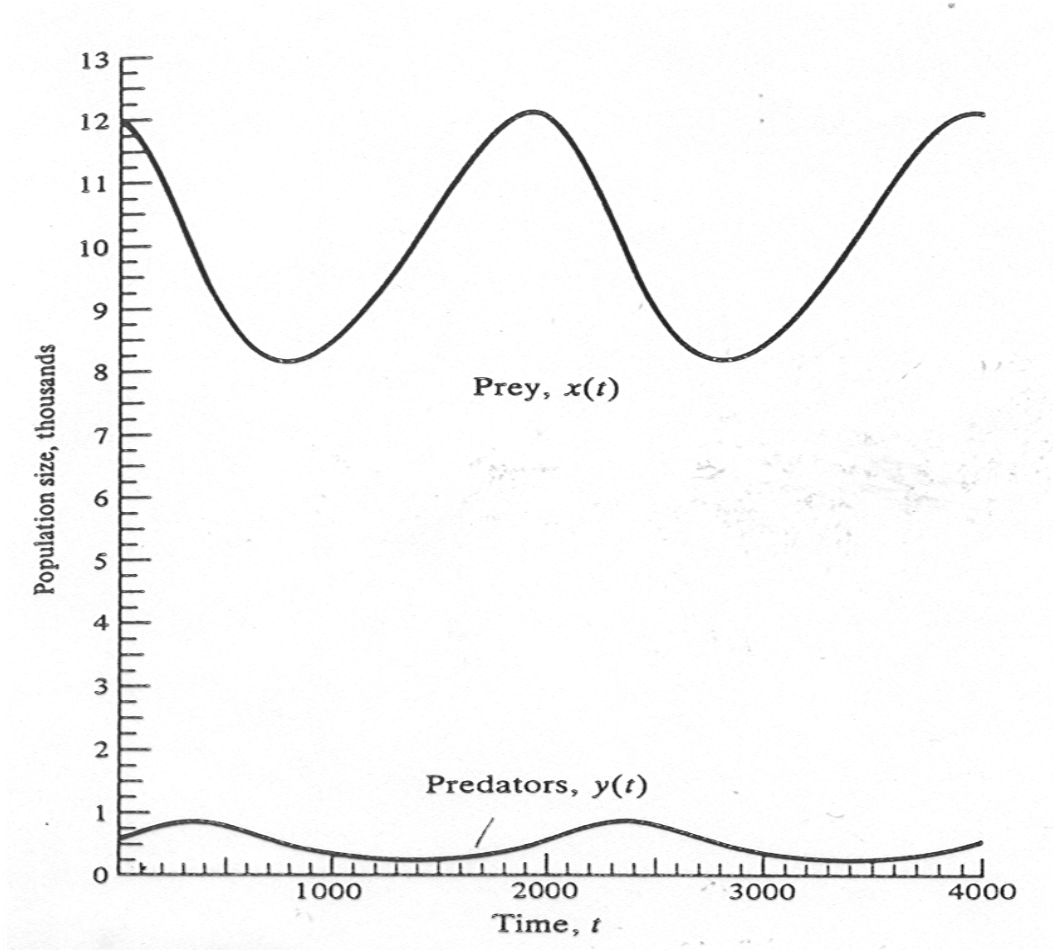
Συνεχής Προσομοίωση

Παράδειγμα: Ανταγωνισμός μεταξύ πληθυσμών
θηράματα- αρπακτικά (prey - predator)
πληθυσμός: x(t), y(t)
ρυθμός μεταβολής θηραμάτων:
dx/dt= ρ.γεννήσεων-ρ.θανάτων=rx(t)-ax(t)y(t)
a, r: θετικοί πραγματικοί
ρυθμός μεταβολής αρπακτικών:
dy/dt=ρυθμός αύξησης (παρουσία θηραμάτων)-
ρ.μείωσης(έλλειψη θηραμάτων)=bx(t)y(t)-sy(t)
s: θετικός πραγματικός
αρχικές συνθήκες: x(t)>0, y(t)>0

- περιοδική συνάρτηση του χρόνου

∃ T>0: x(t+nT)=x(t), ∀t, n θετικός ακέραιος

Συνεχής Προσομοίωση
Παράδειγμα



Στατιστική Προσομοίωση (Monte Carlo Simulation)

Παράδειγμα:
Υπολογισμός εμβαδού κλειστού σχήματος

Δημιουργία τετραγώνου πλευράς α
Δημιουργία ν τυχαίων σημείων (x,y)
Ισχύει ότι: $E/E_\tau = hits/\nu$, όταν $\nu \to \infty$
άρα $E = \alpha^2 * hits/\nu$