Anexos – Código Fuente del Simulador y Ficheros de Ejecución de QNAP2

18 de junio de 2008

Índice

---

1. Código fuente del simulador

Adjuntar el código fuente del simulador diseñado con un lenguaje de propósito general, que incluya comentarios que faciliten su comprensión.

1.1. Arrivalslist.h / Arrivalslist.c

Implementación de las listas de llegadas.

```
1 // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
2 // $LastChangedRevision: 29 $
3 // $LastChangedBy: vicente@jdalmau.es $
4
5 #ifndef _ARRIVALSLIST_H_
6 #define _ARRIVALSLIST_H_
```

```
 7
 8 #include "systypes.h"
 9
10 // implementacion
11
12 // nodos de la lista
13 struct NODEAL{
14         TIME t;
15         struct NODEAL *next;
16 };
17 typedef struct NODEAL NODEAL;
18
19 // lista de tiempos de llegada
20 typedef struct ARRIVALSLIST {
21         NODEAL *first;
22         NODEAL *last;
23 }ARRIVALSLIST;
24
25
26 // operaciones
27 void initAL (ARRIVALSLIST *l);
28 bool emptyAL (ARRIVALSLIST *l);
29 void addArrivalAL (ARRIVALSLIST *l, TIME t);
30 void delArrivalAL (ARRIVALSLIST *l);
31 TIME getArrivalAL (ARRIVALSLIST *l);
32 void showAL (ARRIVALSLIST *l);
33
34 #endif // _ARRIVALSLIST_H_
```

```
 1 // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
 2 // $LastChangedRevision: 29 $
 3 // $LastChangedBy: vicente@jdalmau.es $
 4
 5 #include <stdio.h>
 6 #include <stdlib.h>
 7 #include "arrivalslist.h"
 8
 9 // Implementacion dinamica de las listas de instantes de llegadas
10
11 // Inicializa la lista de llegadas transformandola en una lista vacia
12 //
13 void initAL (ARRIVALSLIST *l) {
14
15         l->first = NULL;
16         l->last = NULL;
17 }
18
19 // Comprueba si la lista de llegadas esta vacia
20 //
21 bool emptyAL (ARRIVALSLIST *l) {
22
23         if (l->first == NULL)
```

```
24                      return true;
25              return false;
26      }
27
28      // Añade un nuevo tiempo de llegadas a la lista al final de la misma
29      //
30      void addArrivalAL (ARRIVALSLIST *l, TIME t) {
31
32              // lista vacia
33              if (l->first == NULL) {
34                      l->first = (NODEAL *)malloc(sizeof(NODEAL));
35                      (l->first)->t = t;
36                      (l->first)->next = NULL;
37                      l->last = l->first;
38              }
39              // ya tenia elementos
40              else {
41                      (l->last)->next = (NODEAL *)malloc(sizeof(NODEAL));
42                      ((l->last)->next)->t = t;
43                      ((l->last)->next)->next = NULL;
44                      l->last = (l->last)->next;
45              }
46      }
47
48      // Elimina el primer elemento de la lista si la lista no esta vacia
49      //
50      void delArrivalAL (ARRIVALSLIST *l) {
51
52              if (l->first == NULL) {
53                      printf("arrivalslist.delArrivalAL(): ERROR, the list is
                            empty\n");
54                      exit(EXIT_FAILURE);
55              }
56
57              // la lista solo tiene un elemento
58              if (l->first == l->last) {
59                      free(l->first);
60                      l->first = NULL;
61                      l->last = NULL;
62              }
63              // hay mas de un elemento
64              else {
65                      NODEAL *n = l->first;
66                      l->first = (l->first)->next;
67                      free(n);
68              }
69      }
70
71      // Obtiene el tiempo de llegada del cliente que ocupa la primera posicion
            de la cola
72      //
73      TIME getArrivalAL (ARRIVALSLIST *l) {
```

```
74
75          if (l->first == NULL) {
76                  printf("arrivalslist.getArrivalAL(): ERROR, the list is
                        empty\n");
77                  exit(EXIT_FAILURE);
78          }
79
80          return (l->first)->t;
81  }
82
83  // Muestra el contenido de la lista de llegadas
84  //
85  void showAL (ARRIVALSLIST *l) {
86
87          if (!emptyAL(l)) {
88                  NODEAL *p = l->first;
89                  while (p != NULL) {
90                          printf("arrivalslist.showAL(): <%f>\n", p->t);
91                          p = p->next;
92                  }
93          }
94  }
```

1.2. Eventlist.h / Eventlist.c

Implementación de la lista de eventos que usaremos en el simulador.

```
1  // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
2  // $LastChangedRevision: 29 $
3  // $LastChangedBy: vicente@jdalmau.es $
4
5  #ifndef _EVENTLIST_H_
6  #define _EVENTLIST_H_
7
8  #include "systypes.h"
9  // Eventos
10 #define A1 0     // llegada de un cliente al nodo 1
11 #define A2 1     // llegada de un cliente al nodo 2
12 #define A3 2     // llegada de un cliente al nodo 3
13 #define D1 3     // salida de un cliente del nodo 1
14 #define D2 4     // salida de un cliente del nodo 2
15 #define D3 5     // salida de un cliente del nodo 3
16 #define EVENTLISTSIZE 6
17
18 // implementacion
19
20 typedef int EVENT;
21
22 typedef TIME EVENTLIST[EVENTLISTSIZE];
23 typedef int POSITIONEL;
24
25 // operaciones
```

```
26 void initEL (EVENTLIST l);
27 bool emptyEL (EVENTLIST l);
28 void setArrivalEventEL (EVENTLIST l, EVENT e, TIME t);
29 void setDepartureEventEL (EVENTLIST l, EVENT e, TIME a, TIME d);
30 EVENT getClosestEventEL (EVENTLIST l);
31 TIME getTimeEL (EVENTLIST l, EVENT e);
32 TIME getArrivalEL (EVENTLIST l, EVENT e);
33 void showEL (EVENTLIST l);
34
35 #endif // _EVENTLIST_H_
```

```
1  // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
2  // $LastChangedRevision: 29 $
3  // $LastChangedBy: vicente@jdalmau.es $
4
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <strings.h>
8  #include "eventlist.h"
9  #include "systypes.h"
10 #include "sysroutines.h"
11
12 // Implementacion de la lista de eventos a usar en el simulador
13
14 // variables para almacenar los tiempos de llegada de los clientes que
       estan usando los servidores
15 TIME arrivals[3];
16
17 // Inicializa la lista de eventos estableciendo un valor temporal nulo
       para todos los eventos considerados.
18 //
19 void initEL (EVENTLIST l) {
20
21         int i;
22         for (i=0; i<EVENTLISTSIZE; i++)
23                 l[i] = NULLTIME;
24         for (i=0; i<3; i++)
25                 arrivals[i] = NULLTIME;
26 }
27
28 // Determina si una lista de eventos esta vacia o no
29 //
30 bool emptyEL (EVENTLIST l) {
31
32         int i = 0;
33         while (i < EVENTLISTSIZE) {
34                 if (l[i] != NULLTIME)
35                         return false;
36                 i++;
37         }
38         return true;
39 }
```

```
40
41 // Establece el valor temporal t para el evento de llegada e en la lista
       de eventos l.
42 // El evento podra ser la llegada de un cliente a uno de los 3 nodos de
       la red.
43 //
44 void setArrivalEventEL (EVENTLIST l, EVENT e, TIME t) {
45
46         l[e] = t;
47 }
48
49 // Establece el valor temporal d para el evento de salida e en la lista
       de eventos l.
50 // Se anotara ademas el instante de llegada a del cliente al sistema.
51 //
52 void setDepartureEventEL (EVENTLIST l, EVENT e, TIME a, TIME d) {
53
54         // instante de salida
55         l[e] = d;
56         // almacenar el instante de llegada del cliente
57         arrivals[e%3] = a;
58 }
59
60 // Retorna la posicion cuyo instante temporal es mas proximo al valor
       actual del reloj del simulador.
61 //
62 EVENT getClosestEventEL (EVENTLIST l) {
63
64         if (emptyEL(l)) {
65                 printf("eventlist.getClosestEventEL(): ERROR, the event
                       list is empty\n");
66                 exit(EXIT_FAILURE);
67         }
68
69         int e = 1;
70         int chosen = 0;
71
72         while (e < EVENTLISTSIZE) {
73                 if (((l[e] != NULLTIME)&&(l[e] < l[chosen]))||(l[chosen]
                       == NULLTIME))
74                         chosen = e;
75                 e++;
76         }
77
78         return chosen;
79 }
80
81 // Retorna el instante temporal asociado al evento e en la lista de
       eventos l.
82 //
83 TIME getTimeEL (EVENTLIST l, EVENT e) {
84
```

```
85          return l[e];
86  }
87
88  // Retorna el instante temporal en que llego el cliente que ocupaba el
        servidor asociado al evento e
89  //
90  TIME getArrivalEL (EVENTLIST l, EVENT e) {
91
92          return arrivals[e%3];
93  }
94
95  char *eventToString (EVENT e) {
96
97          switch (e) {
98                  case A1:
99                          return "A1";
100                         break;
101                 case A2:
102                         return "A2";
103                         break;
104                 case A3:
105                         return "A3";
106                         break;
107                 case D1:
108                         return "D1";
109                         break;
110                 case D2:
111                         return "D2";
112                         break;
113                 case D3:
114                         return "D3";
115                         break;
116         }
117 }
118 // Muestra el contenido de una lista de eventos.
119 //
120 void showEL (EVENTLIST l) {
121
122         int i;
123         for (i = 0; i < EVENTLISTSIZE; i++) {
124                 printf("eventlist.showEL(): %s ", eventToString(i));
125                 if (l[i] == NULLTIME)
126                         printf("<EMPTY>\n");
127                 else
128                         printf("<%f>\n", l[i]);
129         }
130 }
```

1.3. Main.c

```
1  // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
```

```
2  // $LastChangedRevision: 29 $
3  // $LastChangedBy: vicente@jdalmau.es $
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  #include "sysroutines.h"
9  #include "report.h"
10
11 int main(int argc, char *argv[]) {
12
13         if (argc != 5) {
14                 printf("main.main(): ERROR there's at least one missing
                       parameter\n");
15                 printf("main.main(): Usage: simulator lambda_value
                       replication_time min_reps_number debug(0/1)\n");
16                 exit(EXIT_FAILURE);
17         }
18         DEBUG = (strcmp(argv[4], "1")==0)?true:false;
19
20         // inicializacion del simulador
21         simulatorInit(atof(argv[1]), atof(argv[2]));
22
23         // Aplicacion del metodo secuencial con error relativo
24
25     // Numero minimo de replicas que ejecutamos
26     int i;
27     for (i=0; i<atoi(argv[3]); i++)
28         replicate();
29
30         int n = atoi(argv[3]);
31
32         while (largeRelativeError(n)) {
33                 replicate();
34                 n++;
35         }
36
37     // mostrar los resultados
38         showResults(n);
39
40         return 0;
41 }
```

1.4. Rand.h / Rand.c

Generador de números aleatorios.

```
1  // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
2  // $LastChangedRevision: 29 $
3  // $LastChangedBy: vicente@jdalmau.es $
4
5  #ifndef RAND_H_
6  #define RAND_H_
```

```
 7
 8 #define SEG 7          // segmento usado
 9
10 float urand(int stream);
11 void urandst(long zset, int stream);
12 long urandgt(int stream);
13 double exponential(double lambda);
14
15 #endif // RAND_H_
```

```
 1 // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
 2 // $LastChangedRevision: 29 $
 3 // $LastChangedBy: vicente@jdalmau.es $
 4
 5 /* Prime modulus multiplicative linear congruential generator
 6 Z[i] = (630360016 * Z[i-1]) (mod(pow(2,31) - 1)), based on Marse and
       Roberts'
 7 portable FORTRAN random-number generator UNIRAN.  Multiple (100) streams
       are
 8 supported, with seeds spaced 100,000 apart.  Throughout, input argument
 9 "stream" must be an int giving the desired stream number.  The header
       file
10 rand.h must be included in the calling program (#include "rand.h")
11 before using these functions.
12
13 Usage: (Three functions)
14
15 1. To obtain the next U(0,1) random number from stream "stream," execute
16        u = urand(stream);
17    where urand is a float function.  The float variable u will contain
          the
18    next random number.
19
20 2. To set the seed for stream "stream" to a desired value zset, execute
21        urandst(zset, stream);
22    where urandst is a void function and zset must be a long set to the
23    desired seed, a number between 1 and 2147483646 (inclusive).  Default
24    seeds for all 100 streams are given in the code.
25
26 3. To get the current (most recently used) integer in the sequence being
27    generated for stream "stream" into the long variable zget, execute
28        zget = urandgt(stream);
29    where urandgt is a long function. */
30
31 #include <stdio.h>
32 //#include <stdlib.h>
33 #include <math.h>
34 #include "rand.h"
35
36
37 /* Define the constants. */
38 #define MODLUS 2147483647
```

```
39 #define MULT1 24112
40 #define MULT2 26143
41
42 /* Set the default seeds for all 100 streams. */
43
44 static long zrng[] =
45 {
46         0,
47         1973272912, 281629770, 20006270, 1280689831, 2096730329,
               1933576050,
48         913566091, 246780520, 1363774876, 604901985, 1511192140,
               1259851944,
49         824064364, 150493284, 242708531, 75253171, 1964472944,
               1202299975,
50         233217322, 1911216000, 726370533, 403498145, 993232223,
               1103205531,
51         762430696, 1922803170, 1385516923, 76271663, 413682397,
               726466604,
52         336157058, 1432650381, 1120463904, 595778810, 877722890,
               1046574445,
53         68911991, 2088367019, 748545416, 622401386, 2122378830,
               640690903,
54         1774806513, 2132545692, 2079249579, 78130110, 852776735,
               1187867272,
55         1351423507, 1645973084, 1997049139, 922510944, 2045512870,
               898585771,
56         243649545, 1004818771, 773686062, 403188473, 372279877,
               1901633463,
57         498067494, 2087759558, 493157915, 597104727, 1530940798,
               1814496276,
58         536444882, 1663153658, 855503735, 67784357, 1432404475,
               619691088,
59         119025595, 880802310, 176192644, 1116780070, 277854671,
               1366580350,
60         1142483975, 2026948561, 1053920743, 786262391, 1792203830,
               1494667770,
61         1923011392, 1433700034, 1244184613, 1147297105, 539712780,
               1545929719,
62         190641742, 1645390429, 264907697, 620389253, 1502074852,
               927711160,
63         364849192, 2049576050, 638580085, 547070247
64 };
65
66
67 /* Generate the next random number. */
68
69 float urand(int stream)
70 {
71         long zi, lowprd, hi31;
72
73         zi = zrng[stream];
74         lowprd = (zi & 65535) * MULT1;
```

```
75          hi31 = (zi >> 16) * MULT1 + (lowprd >> 16);
76          zi = ((lowprd & 65535) - MODLUS) +
77               ((hi31 & 32767) << 16) + (hi31 >> 15);
78          if (zi < 0) zi += MODLUS;
79          lowprd = (zi & 65535) * MULT2;
80          hi31 = (zi >> 16) * MULT2 + (lowprd >> 16);
81          zi = ((lowprd & 65535) - MODLUS) + ((hi31 & 32767) << 16) + (hi31
                >>15);
82          if (zi < 0) zi += MODLUS;
83          zrng[stream] = zi;
84          return ((zi >> 7 | 1) + 1) / (float)16777216.0;
85  }
86
87  /* set the current zrng for stream "stream" to zset */
88
89  void urandst (long zset, int stream)
90  {
91          zrng[stream] = zset;
92  }
93
94  /* Return the current zrng for stream "stream" */
95
96  long urandgt (int stream)
97  {
98          return zrng[stream];
99  }
100
101 // Devuelve un valor perteneciente a una distribución exponencial de
        parametro
102 // lambda
103 double exponential(double lambda)
104 {
105         float u;
106
107         // Generamos un numero aleatorio dentro de U(0, 1)
108         u = (float) urand(SEG);          // u pertenece a U(0, 1)
109
110         // Aplicamos el metodo de la transformada inversa para convertir
                la u
111         // que acabamos de calcular a un valor perteneciente a Exp(lambda
                )
112         return (-(log(u)/lambda));
113 }
```

1.5. Report.h / Report.c

Libreria de soporte a la presentación del informe estadístico. Contiene todas las funciones auxiliares necesarias para poder presentar al usuario los resultados obtenidos en términos del intervalo de confianza.

```
1  // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
2  // $LastChangedRevision: 29 $
```

```
3  // $LastChangedBy: vicente@jdalmau.es $
4
5  #ifndef _REPORT_H_
6  #define _REPORT_H_
7
8  #include "systypes.h"
9
10 double sampleVariance (int n, TIME samples, TIME mean);
11 double confidenceInterval (int n, TIME samples, TIME mean);
12 double tstudent (int dof);
13 bool largeRelativeError (int n);
14
15 #endif // _REPORT_H_
```

```
1  // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
2  // $LastChangedRevision: 29 $
3  // $LastChangedBy: vicente@jdalmau.es $
4
5  #include "report.h"
6  #include "systypes.h"
7  #include <math.h>
8  #include <stdio.h>
9  #include <stdlib.h>
10
11 // Libreria de soporte a la presentacion del informe estadistico.
       Contiene todas las funciones auxiliares necesarias
12 // para poder presentar al usuario los resultados obtenidos en terminos
       del intervalo de confianza.
13
14 // Acceso a las variables externas de <sysroutines> para el calculo del
       error relativo.
15 extern TIME RT1, RT2, RT3;                          // tiempo de respuesta de
        cada estacion
16 extern TIME RTS;                        // tiempo de respuesta del sistema
17 extern TIME SQRTRT1, SQRTRT2, SQRTRT3;           // sumatorio con el
       cuadrado de los tiempos de respuesta de cada estacion
18 extern TIME SQRTRTS;                           // sumatorio con el cuadrado
       de los tiempos de respuesta del sistema
19
20 // Variables para mostrar los resultados de la simulacion
21 TIME MRT1, MRT2, MRT3, MRTS;
22 TIME CIRT1, CIRT2, CIRT3, CIRTS;
23
24 // Percentiles 0.975 de la distribucion t de student para distintos
       grados de
25 // libertad
26 //
27 double t_0975[] =
28 {
29         12.71, 4.30, 3.18, 2.78, 2.57, 2.45, 2.36, 2.31, 2.26, 2.23,
                          // dof = 1..30
30         2.20, 2.18, 2.16, 2.14, 2.13, 2.12, 2.11, 2.10, 2.09, 2.09,
```

```
31          2.08, 2.07, 2.07, 2.06, 2.06, 2.06, 2.05, 2.05, 2.04, 2.04,
32          2.02,

                // dof = 40
33          2.01,

                // dof = 50
34          2.00,

                // dof = 60
35          1.98,

                // dof = 120
36          1.96

                // dof = infinite
37  };
38
39  // Calcula la varianza muestral aplicando tomando para ello n como el
        numero de muestras, samples como el sumatorio
40  // de los cuadrados de las muestras y mean como la media muestral de las
        mismas muestras.
41  //
42  double sampleVariance (int n, TIME samples, TIME mean) {
43
44          return (samples - (n * pow(mean, 2))) / (n - 1);
45  }
46
47  // Calcula el intervalo de confianza teniendo en cuenta que n es el
        numero de muestras, samples el sumatorio
48  // de los cuadrados de las muestras y mean la media muestral de las
        mismas muestras.
49  //
50  double confidenceInterval (int n, TIME samples, TIME mean) {
51
52          return tstudent(n - 1) * sqrt(sampleVariance(n, samples, mean) /
                n);
53  }
54
55  // Retorna el valor percentil de la distribucion t de student para n
        grados de libertad.
56  //
57  double tstudent (int dof) {
58
59          double t;
60
61          if (0 < dof && dof <= 30) {
62                  t = t_0975[dof - 1];
63          } else if (dof <= 40) {
64                  t = t_0975[30];
65          } else if (dof <= 50) {
66                  t = t_0975[31];
```

```
67          } else if (dof <= 60) {
68                  t = t_0975[32];
69          } else if (dof <= 120) {
70                  t = t_0975[33];
71          } else if (dof > 120) {
72                  t = t_0975[34];
73          } else {
74                  printf("report.tstudent(): Invalid value for the degrees
                        of freedom.");
75                  exit(EXIT_FAILURE);
76          }
77          return t;
78  }
79
80  // Comprueba si el error relativo es adecuado  para todos los parametros
        que se manejan en la simulacion, tiempo de respuesta,
81  // numero medio de clientes y utilizacion en funcion del valor n que
        representa el numero de replicas.
82  // Devuelve un valor falso si los cocientes de todos los intervalos de
        confianza y las medias muestrales son un valor menor
83  // o igual a la cota de error (corregido) manejado por el simulador.
84  //
85  bool largeRelativeError (int n) {
86
87          // tiempo de respuesta nodo 1
88          TIME meanRT1    = RT1 / n;
89          TIME ciRT1          = confidenceInterval(n, SQRTRT1, meanRT1);
90
91          if ((ciRT1 / meanRT1) > RELATIVEERROR)
92                  return true;
93
94          // tiempo de respuesta nodo 2
95          TIME meanRT2    = RT2 / n;
96          TIME ciRT2      = confidenceInterval(n, SQRTRT2, meanRT2);
97
98          if ((ciRT2 / meanRT2) > RELATIVEERROR)
99                  return true;
100
101         // tiempo de respuesta nodo 3
102         TIME meanRT3    = RT3 / n;
103         TIME ciRT3      = confidenceInterval(n, SQRTRT3, meanRT3);
104
105         if ((ciRT3 / meanRT3) > RELATIVEERROR)
106                 return true;
107
108     // tiempo de respuesta del sistema
109     TIME meanRTS = RTS / n;
110     TIME ciRTS = confidenceInterval(n, SQRTRTS, meanRTS);
111
112         if ((ciRTS / meanRTS) > RELATIVEERROR)
113                 return true;
114
```

```
115             // el error relativo ya no es demasiado grande
116             MRT1 = meanRT1;
117             MRT2 = meanRT2;
118             MRT3 = meanRT3;
119             MRTS = meanRTS;
120             CIRT1 = ciRT1;
121             CIRT2 = ciRT2;
122             CIRT3 = ciRT3;
123             CIRTS = ciRTS;
124
125             return false;
126     }
127
128     // Muestra los resultados obtenidos en la simulacion
129     // El parametro n es el numero de replicas ejecutadas
130     //
131     void showResults (int n) {
132
133             printf("\nreport.showResults(): ___Mean Values:\n");
134             printf("report.showResults(): ___-------------\n");
135             printf("report.showResults(): ___Response Time Station 1:\t %f\n",
                    MRT1);
136             printf("report.showResults(): ___Response Time Station 2:\t %f\n",
                    MRT2);
137             printf("report.showResults(): ___Response Time Station 3:\t %f\n",
                    MRT3);
138             printf("report.showResults(): ___Response Time SYSTEM___:\t %f\n",
                    MRTS);
139             printf("\n");
140         printf("report.showResults(): _______Confidence Intervals:\n");
141             printf("report.showResults(): ___----------------------\n");
142             printf("report.showResults(): ___Executed Replications:\t\t %i\n",
                    n);
143             printf("report.showResults(): ___Confidence Level:\t\t95 %%\n");
144             printf("report.showResults(): ___Response Time Station 1:\t %f\n",
                    CIRT1);
145             printf("report.showResults(): ___Response Time Station 2:\t %f\n",
                    CIRT2);
146             printf("report.showResults(): ___Response Time Station 3:\t %f\n",
                    CIRT3);
147             printf("report.showResults(): ___Response Time SYSTEM___:\t %f\n",
                    CIRTS);
148     }
```

1.6. Sysroutines.h / Sysroutines.c

Contiene las rutinas necesarias para procesar los eventos considerados en el sistema.

```
1  // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
2  // $LastChangedRevision: 29 $
3  // $LastChangedBy: vicente@jdalmau.es $
4
```

```
5  #ifndef _SYSROUTINES_H_
6  #define _SYSROUTINES_H_
7
8  #include "eventlist.h"
9
10 void simulatorInit (double theLambda, double maxReplicationTime);
11 void replicationInit (void);
12 void replicate (void);
13 EVENT timer (void);
14 // eventos
15 void arrival1 (void);
16 void departure1 (void);
17 void arrival2 (void);
18 void departure2 (void);
19 void arrival3 (void);
20 void departure3 (void);
21
22 #endif // _SYSROUTINES_H_
```

```
1  // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
2  // $LastChangedRevision: 29 $
3  // $LastChangedBy: vicente@jdalmau.es $
4
5  #include <stdio.h>
6  #include <math.h>
7  #include "sysroutines.h"
8  #include "systypes.h"
9  #include "arrivalslist.h"
10 #include "eventlist.h"
11 #include "rand.h"
12
13 // rutinas para los eventos considerados en el sistema
14
15 // Duracion de la Replica
16 TIME MAXREPLICATIONTIME;
17
18 // replicas EJECUTADAS
19 int REPS = 0;
20
21 // distribuciones utilizadas
22 double lambda;          // Con 0 < lambda < 0.02 para que la red sea
       estable
23 double mu1      = 0.1;  // Tiempo medio servicio nodo 1 = 10 ms
24 double mu2      = 0.06; // Tiempo medio servicio nodo 2 = 15 ms
25 double mu3      = 0.05; // Tiempo medio servicio nodo 3 = 20 ms
26
27 // probabilidades de ramificacion y salida del sistema
28 float p12       = 0.3;
29 float p13       = 0.7;
30 float p30       = 0.4;
31 float p31       = 0.6;
32
```

```
33 // Variables de estado del sistema
34 TIME clock;
35 EVENTLIST eventList;              // lista de eventos del sistema
36 bool serverBusy1, serverBusy2, serverBusy3;      // ocupacion servidores
       de los 3 nodos
37 ARRIVALSLIST arrivals1, arrivals2, arrivals3;           // colas de los 3
       nodos
38 TIME lastEvent;                   // instante en que se dio el ultimo
       evento
39
40 // Contadores estadisticos (para cada una de las replicas)
41 int served1, served2, served3;  // numero de clientes que han recibido
       servicio en cada estacion
42 int served;                       // numero de clientes que han recibido
       servicio en el sistema (han salido)
43 TIME rTSum1, rTSum2, rTSum3;     // tiempo de respuesta acumulado para los
       clientes en cada estacion
44
45 // Acumuladores para calcular la varianza muestral
46 TIME RT1, RT2, RT3;                          // tiempo de respuesta de cada
       estacion
47 TIME RTS;                       // tiempo de respuesta del sistema
48 TIME SQRTRT1, SQRTRT2, SQRTRT3;          // sumatorio con el cuadrado de
       los tiempos de respuesta de cada estacion
49 TIME SQRTRTS;                           // sumatorio con el cuadrado de los
       tiempos de respuesta del sistema
50
51 // Prepara el simulador para ejecutar una nueva simulacion.
52 //
53 void simulatorInit (double theLambda, double maxReplicationTime) {
54
55         lambda = theLambda;
56         MAXREPLICATIONTIME = maxReplicationTime;
57
58         RT1 = RT2 = RT3 = RTS = INITIALTIME;
59         SQRTRT1 = SQRTRT2 = SQRTRT3 = SQRTRTS = INITIALTIME;
60
61         if (DEBUG) {
62                 printf("sysroutines.simulatorInit(): Lambda equals to <%f
                    >\n", lambda);
63                 printf("sysroutines.simulatorInit(): Time for each
                    replication <%f>\n", MAXREPLICATIONTIME);
64         }
65 }
66
67 // Prepara el simulador para ejecutar una nueva replica
68 //
69 void replicationInit (void) {
70
71         clock = INITIALTIME;
72         lastEvent = INITIALTIME;
73
```

```
74          // configurar la lista de eventos
75          initEL(eventList);
76          setArrivalEventEL(eventList, A1, exponential(lambda));
77
78          serverBusy1 = serverBusy2 = serverBusy3 = false;
79          initAL(&arrivals1);
80          initAL(&arrivals2);
81          initAL(&arrivals3);
82
83          // contadores estadisticos
84          served1 = served2 = served3 = served = 0;
85          rTSum1 = rTSum2 = rTSum3 = INITIALTIME;
86
87          if (DEBUG)
88                  printf("sysroutines.replicationInit(): Ready for a new
                        replication\n", clock);
89  }
90
91  // Rutina del temporizador que se encargara de determinar el siguiente
        evento a procesar, retorna
92  // el evento.
93  //
94  EVENT timer (void) {
95
96          EVENT e = getClosestEventEL(eventList);
97          clock = getTimeEL(eventList, e);
98
99          return e;
100 }
101
102 // Ejecuta una replica en el simulador
103 //
104 void replicate (void) {
105
106         REPS++;
107
108         replicationInit();
109         while (clock < MAXREPLICATIONTIME) {
110                 EVENT e = timer();
111                 switch (e) {
112                         case A1:
113                                 arrival1();
114                                 break;
115                         case D1:
116                                 departure1();
117                                 break;
118                         case A2:
119                                 arrival2();
120                                 break;
121                         case D2:
122                                 departure2();
123                                 break;
```

```
124                             case A3:
125                                     arrival3();
126                                     break;
127                             case D3:
128                                     departure3();
129                                     break;
130                             default:
131                                     break;
132                     }
133             }
134
135             if (DEBUG) {
136                     printf("sysroutines.replicate(): Replication <%5i> - 
                            Served Station 1: <%5i>\n", REPS, served1);
137                     printf("sysroutines.replicate(): Replication <%5i> - 
                            Served Station 2: <%5i>\n", REPS, served2);
138                     printf("sysroutines.replicate(): Replication <%5i> - 
                            Served Station 3: <%5i>\n", REPS, served3);
139                     printf("sysroutines.replicate(): Replication <%5i> - 
                            Served System:    <%5i>\n", REPS, served);
140                     printf("sysroutines.replicate(): Replication <%5i> - RT 
                            Station 1: <%3.5f>\n", REPS, rTSum1/served1);
141                     printf("sysroutines.replicate(): Replication <%5i> - RT 
                            Station 2: <%3.5f>\n", REPS, rTSum2/served2);
142                     printf("sysroutines.replicate(): Replication <%5i> - RT 
                            Station 3: <%3.5f>\n", REPS, rTSum3/served3);
143                     printf("sysroutines.replicate(): Replication <%5i> - RT 
                            System    : <%3.5f>\n", REPS, (rTSum1+rTSum2+rTSum3)/
                            served);
144             }
145
146             // acumular los contadores estadisticos
147
148             RT1 += rTSum1/served1;
149             RT2 += rTSum2/served2;
150             RT3 += rTSum3/served3;
151             RTS += (rTSum1+rTSum2+rTSum3)/served;
152
153             SQRTRT1 += pow(rTSum1/served1, 2);
154             SQRTRT2 += pow(rTSum2/served2, 2);
155             SQRTRT3 += pow(rTSum3/served3, 2);
156             SQRTRTS += pow((rTSum1+rTSum2+rTSum3)/served, 2);
157     }
158
159     // Rutinas para procesar los eventos de la red
160
161     // Llegada de un cliente al nodo 1
162     //
163     void arrival1 (void) {
164
165             if (DEBUG)
```

```
166                 printf("sysroutines.arrival1():\t<%3.2f>\tOne\tclient\t
                        arrived.\n", clock);
167
168         // el servidor esta desocupado
169         if (!serverBusy1) {
170                 serverBusy1 = true;
171                 // marcar su instante de salida
172                 setDepartureEventEL(eventList, D1, clock, clock +
                        exponential(mu1));
173         }
174         // poner el cliente en cola
175         else {
176                 addArrivalAL(&arrivals1, clock);
177         }
178         // siguiente llegada
179         setArrivalEventEL(eventList, A1, clock + exponential(lambda));
180
181         lastEvent = clock;
182 }
183
184 // Salida de un cliente del nodo 1
185 //
186 void departure1 (void) {
187
188         if (DEBUG)
189                 printf("sysroutines.departure1():\t<%3.2f>\tOne\tclient\t
                        left.\n", clock);
190
191         // actualizar contadores estadisticos (tiempo de servicio)
192         rTSum1 += clock - getArrivalEL(eventList, D1);
193         served1++;
194
195         setDepartureEventEL(eventList, D1, NULLTIME, NULLTIME);
196         if (emptyAL(&arrivals1)) {
197                 serverBusy1 = false;
198         }
199         else {
200                 // seleccionar el siguiente de la cola para entrar al
                        servidor
201                 setDepartureEventEL(eventList, D1, clock, clock +
                        exponential(mu1));
202                 // actualizar contadores estadisticos (tiempo de espera)
203                 rTSum1 += clock - getArrivalAL(&arrivals1);
204                 // eliminarlo de la cola de espera
205                 delArrivalAL(&arrivals1);
206         }
207
208         // encaminar el cliente hacia la estacion 2 o la estacion 3
209         if (urand(SEG) < p12) {
210                 if (DEBUG)
211                         printf("sysroutines.departure1():\tThe\tclient\twill
                                \tgo\tto\tstation\t2.\n");
```

```
212
213                 // servidor libre
214                 if (!serverBusy2) {
215                         serverBusy2 = true;
216                         // marcar su instante de salida
217                         setDepartureEventEL(eventList, D2, clock, clock +
                                exponential(mu2));
218                 }
219                 // poner el cliente nuevamente en cola
220                 else {
221                         addArrivalAL(&arrivals2, clock);
222                 }
223         }
224         else {
225                 if (DEBUG)
226                         printf("sysroutines.departure1():␣The␣client␣will
                            ␣go␣to␣station␣3.\n");
227
228                 // servidor libre
229                 if (!serverBusy3) {
230                         serverBusy3 = true;
231                         // marcar su instante de salida
232                         setDepartureEventEL(eventList, D3, clock, clock +
                                exponential(mu3));
233                 }
234                 // poner el cliente nuevamente en cola
235                 else {
236                         addArrivalAL(&arrivals3, clock);
237                 }
238         }
239
240         lastEvent = clock;
241 }
242
243 // Llegada de un cliente al nodo 2.
244 //
245 void arrival2 (void) {
246
247         if (DEBUG)
248                 printf("sysroutines.arrival2():␣\t<%3.2f>␣One␣client␣
                        arrived.\n", clock);
249
250         // el servidor esta desocupado
251         if (!serverBusy2) {
252                 serverBusy2 = true;
253                 // marcar su instante de salida
254                 setDepartureEventEL(eventList, D2, clock, clock +
                        exponential(mu2));
255         }
256         // poner el cliente en cola
257         else {
258                 addArrivalAL(&arrivals2, clock);
```

```
259         }
260
261         lastEvent = clock;
262 }
263
264 // Salida de un cliente del nodo 2.
265 //
266 void departure2 (void) {
267
268         if (DEBUG)
269                 printf("sysroutines.departure2():\t<%3.2f>\tOne client
                        left.\n", clock);
270
271         // actualizar contadores estadisticos (tiempo de servicio)
272         rTSum2 += clock - getArrivalEL(eventList, D2);
273         served2++;
274
275         setDepartureEventEL(eventList, D2, NULLTIME, NULLTIME);
276         if (emptyAL(&arrivals2)) {
277                 serverBusy2 = false;
278         }
279         else {
280                 // seleccionar el siguiente de la cola para entrar al
                        servidor
281                 setDepartureEventEL(eventList, D2, clock, clock +
                        exponential(mu2));
282                 // actualizar contadores estadisticos (tiempo de espera)
283                 rTSum2 += clock - getArrivalAL(&arrivals2);
284                 // eliminarlo de la cola de espera
285                 delArrivalAL(&arrivals2);
286         }
287
288         // encaminar el cliente hacia la estacion 1
289
290         // servidor libre
291         if (!serverBusy1) {
292                 serverBusy1 = true;
293                 // marcar su instante de salida
294                 setDepartureEventEL(eventList, D1, clock, clock +
                        exponential(mu1));
295         }
296         // poner el cliente nuevamente en cola
297         else {
298                 addArrivalAL(&arrivals1, clock);
299         }
300
301         lastEvent = clock;
302 }
303
304 // Llegada de un cliente al nodo 3.
305 //
306 void arrival3 (void) {
```

```
307
308         if (DEBUG)
309                 printf("sysroutines.arrival3():\t<%3.2f>\tOne client
                        arrived.\n", clock);
310
311         // el servidor esta desocupado
312         if (!serverBusy3) {
313                 serverBusy3 = true;
314                 // marcar su instante de salida
315                 setDepartureEventEL(eventList, D3, clock, clock +
                        exponential(mu3));
316         }
317         // poner el cliente en cola
318         else {
319                 addArrivalAL(&arrivals3, clock);
320         }
321
322         lastEvent = clock;
323 }
324
325 // Salida de un cliente del nodo 3.
326 //
327 void departure3 (void) {
328
329         if (DEBUG)
330                 printf("sysroutines.departure3():\t<%3.2f>\tOne client
                        left.\n", clock);
331
332         // actualizar contadores estadisticos (tiempo de servicio)
333         rTSum3 += clock - getArrivalEL(eventList, D3);
334         served3++;
335
336         setDepartureEventEL(eventList, D3, NULLTIME, NULLTIME);
337         if (emptyAL(&arrivals3)) {
338                 serverBusy3 = false;
339         }
340         else {
341                 // seleccionar el siguiente de la cola para entrar al
                        servidor
342                 setDepartureEventEL(eventList, D3, clock, clock +
                        exponential(mu3));
343                 // actualizar contadores estadisticos (tiempo de espera)
344                 rTSum3 += clock - getArrivalAL(&arrivals3);
345                 // eliminarlo de la cola de espera
346                 delArrivalAL(&arrivals3);
347         }
348
349         // encaminar el cliente hacia la estacion 1 o hacia afuera del
                sistema
350         if (urand(SEG) < p30) {
351                 if (DEBUG)
```

```
352                         printf("sysroutines.departure3():␣The␣client␣will
                                ␣go␣out␣of␣the␣system.\n");
353
354                 // actualizar contadores estadisticos
355                 served++;
356         }
357         else {
358                 if (DEBUG)
359                         printf("sysroutines.departure3():␣The␣client␣will
                                ␣go␣to␣station␣1.\n");
360
361                 // servidor libre
362                 if (!serverBusy1) {
363                         serverBusy1 = true;
364                         // marcar su instante de salida
365                         setDepartureEventEL(eventList, D1, clock, clock +
                                exponential(mu1));
366                 }
367                 // poner el cliente nuevamente en cola
368                 else {
369                         addArrivalAL(&arrivals1, clock);
370                 }
371         }
372
373         lastEvent = clock;
374 }
```

1.7. Systypes.h

Definición de algunos tipos usados en el sistema

```
1  // $LastChangedDate: 2008-06-18 21:24:02 +0200 (mié, 18 jun 2008) $
2  // $LastChangedRevision: 29 $
3  // $LastChangedBy: vicente@jdalmau.es $
4
5  // Contiene la definición de tipos usados en el sistema
6
7  #ifndef _SYSTYPES_H_
8  #define _SYSTYPES_H_
9
10 #include <stdbool.h>
11
12 // Debug
13 bool DEBUG;
14
15 // Tiempo
16 typedef double TIME;
17 #define NULLTIME -1.0
18 #define INITIALTIME 0.0
19
20 // Error relativo
```

```
21 #define ERRORLEVEL 0.1                                    // por defecto
       del 10%
22 #define RELATIVEERROR ERRORLEVEL / (1.0 + ERRORLEVEL)     // error
       corregido
23 #define MINREPLICATIONS 1                                 // numero minimo
       inicial
24                                                           //para el metodo
                                                                 secuencial
25
26 #endif // _SYSTYPES_H_
```

2. Resultados obtenidos con QNAP

A continuación mostramos los diferentes ficheros de resultados arrojados por QNAP en función del valor de λ (tráfico medio de entrada) utilizado:

2.1. Lambda = 0.002

```
1
2    SIMULOG    ***   QNAP2   ***    ( 15-09-2000   ) V 9.4
3    (C)   COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986
4
5
6    ***SIMULATION WITH SPECTRAL METHOD ***
7    ... TIME =       10000000.00  , NB SAMPLES =      512  , CONF. LEVEL = 0.95
8    (0I0A03)  ==>WARNING (OUTPUT) : SOME CONFIDENCE INTERVALS ARE NOT
         AVAILABLE
9                                        FOR QUEUE ... RT         . 0 ASSUMED THEN
10   ************************************************************************
11   *    NAME    *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
12   ************************************************************************
13   *            *          *          *          *          *          *
14   *SOURCE1     * 509.6    * 1.000    * 1.000    * 509.6    *      19624*
15   *    +/-     * 6.925    *0.0000E+00*0.0000E+00* 6.925    *          *
16   *            *          *          *          *          *          *
17   *STATION1    * 10.07    *0.7124E-01*0.7676E-01* 10.85    *      70766*
18   *    +/-     *0.7711E-01*0.1831E-02*0.2050E-02*0.1056    *          *
19   *            *          *          *          *          *          *
20   *STATION2    * 15.28    *0.3291E-01*0.3414E-01* 15.85    *      21535*
21   *    +/-     *0.2134    *0.1298E-02*0.1436E-02*0.2801    *          *
22   *            *          *          *          *          *          *
23   *STATION3    * 20.22    *0.9957E-01*0.1106    * 22.47    *      49231*
24   *    +/-     *0.1695    *0.2477E-02*0.3216E-02*0.3641    *          *
25   *            *          *          *          *          *          *
26   *RT          *0.0000E+00*0.0000E+00*0.2215    * 112.9    *      19624*
27   *    +/-     *----------*0.0000E+00*0.6321E-02* 2.490    *          *
28   *            *          *          *          *          *          *
29   ************************************************************************
30   ... END OF SIMULATION ...
31
```

```
32
33                 MEMORY USED:         9123 WORDS OF 4 BYTES
34                   (   0.06   % OF TOTAL MEMORY)
35       62 /END/
```

2.2. Lambda = 0.004

```
 1
 2    SIMULOG    ***   QNAP2   ***   ( 15-09-2000   ) V 9.4
 3    (C)  COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986
 4
 5
 6    ***SIMULATION WITH SPECTRAL METHOD ***
 7    ... TIME =       10000000.00  , NB SAMPLES =      512  , CONF. LEVEL = 0.95
 8    (0I0A03)  ==>WARNING (OUTPUT) : SOME CONFIDENCE INTERVALS ARE NOT
         AVAILABLE
 9                                        FOR QUEUE ... RT         . 0 ASSUMED THEN
10    ********************************************************************
11    *    NAME    *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
12    ********************************************************************
13    *            *           *          *          *          *          *
14    *SOURCE1     * 251.5     * 1.000    * 1.000    * 251.5    *      39765*
15    *    +/-     * 3.138     *0.0000E+00*0.0000E+00* 3.138    *          *
16    *            *           *          *          *          *          *
17    *STATION1   * 10.03     *0.1447    *0.1685    * 11.69    *     144230*
18    *    +/-     *0.6940E-01*0.2647E-02*0.3876E-02*0.1170    *          *
19    *            *           *          *          *          *          *
20    *STATION2   * 15.16     *0.6644E-01*0.7083E-01* 16.17    *      43809*
21    *    +/-     *0.1216     *0.1406E-02*0.1702E-02*0.1802    *          *
22    *            *           *          *          *          *          *
23    *STATION3   * 20.10     *0.2018    *0.2528    * 25.17    *     100421*
24    *    +/-     *0.1588     *0.3655E-02*0.5737E-02*0.2792    *          *
25    *            *           *          *          *          *          *
26    *RT          *0.0000E+00*0.0000E+00*0.4921    * 123.8    *      39765*
27    *    +/-     *----------*0.0000E+00*0.9686E-02* 1.390    *          *
28    *            *           *          *          *          *          *
29    ********************************************************************
30    ... END OF SIMULATION ...
31
32
33                 MEMORY USED:         8875 WORDS OF 4 BYTES
34                   (   0.06   % OF TOTAL MEMORY)
35       62 /END/
```

2.3. Lambda = 0.006

```
 1
 2    SIMULOG    ***   QNAP2   ***   ( 15-09-2000   ) V 9.4
 3    (C)  COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986
 4
 5
 6    ***SIMULATION WITH SPECTRAL METHOD ***
 7    ... TIME =       10000000.00  , NB SAMPLES =      512  , CONF. LEVEL = 0.95
```

```
(0I0A03)  ==>WARNING (OUTPUT) : SOME CONFIDENCE INTERVALS ARE NOT
    AVAILABLE
                                        FOR QUEUE ... RT        . 0 ASSUMED THEN
*****************************************************************
*   NAME   *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************
*          *           *          *          *          *          *
*SOURCE1   * 167.6     * 1.000    * 1.000    * 167.6    *     59656*
*   +/-    * 1.414     *0.0000E+00*0.0000E+00* 1.414    *          *
*          *           *          *          *          *          *
*STATION1  * 10.04     *0.2157    *0.2749    * 12.80    *    214716*
*   +/-    *0.4541E-01*0.2381E-02*0.4194E-02*0.1084    *          *
*          *           *          *          *          *          *
*STATION2  * 14.98     *0.9657E-01*0.1066    * 16.54    *     64447*
*   +/-    *0.1414     *0.1931E-02*0.2375E-02*0.1768    *          *
*          *           *          *          *          *          *
*STATION3  * 20.08     *0.3017    *0.4302    * 28.63    *    150268*
*   +/-    *0.1339     *0.3295E-02*0.7462E-02*0.3437    *          *
*          *           *          *          *          *          *
*RT        *0.0000E+00*0.0000E+00*0.8117    * 136.1    *     59655*
*   +/-    *----------*0.0000E+00*0.1228E-01* 1.664    *          *
*          *           *          *          *          *          *
*****************************************************************
... END OF SIMULATION ...


              MEMORY USED:        9315 WORDS OF 4 BYTES
                (   0.06   % OF TOTAL MEMORY)
     62 /END/
```

2.4. Lambda = 0.008

```

 SIMULOG    ***  QNAP2  ***   ( 15-09-2000  ) V 9.4
 (C)  COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986


***SIMULATION WITH SPECTRAL METHOD ***
... TIME =      10000000.00  , NB SAMPLES =     512  , CONF. LEVEL = 0.95
(0I0A03)  ==>WARNING (OUTPUT) : SOME CONFIDENCE INTERVALS ARE NOT
    AVAILABLE
                                        FOR QUEUE ... RT        . 0 ASSUMED THEN
*****************************************************************
*   NAME   *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************
*          *           *          *          *          *          *
*SOURCE1   * 125.6     * 1.000    * 1.000    * 125.6    *     79647*
*   +/-    *0.9658     *0.0000E+00*0.0000E+00*0.9658    *          *
*          *           *          *          *          *          *
*STATION1  * 10.05     *0.2868    *0.4017    * 14.07    *    285466*
*   +/-    *0.4364E-01*0.2971E-02*0.6583E-02*0.1480    *          *
*          *           *          *          *          *          *
```

```
*STATION2  * 15.05     *0.1296     *0.1490     * 17.31     *      86061*
*    +/-   *0.1121     *0.2475E-02*0.3110E-02*0.1517     *           *
*          *           *           *           *           *           *
*STATION3  * 20.13     *0.4013     *0.6714     * 33.67     *     199404*
*    +/-   *0.1099     *0.4348E-02*0.1346E-01*0.4988     *           *
*          *           *           *           *           *           *
*RT        *0.0000E+00*0.0000E+00* 1.222     * 153.4     *      79646*
*    +/-   *----------*0.0000E+00*0.2061E-01* 2.333     *           *
*          *           *           *           *           *           *
*****************************************************************
... END OF SIMULATION ...


              MEMORY USED:         9259 WORDS OF 4 BYTES
                (   0.06   % OF TOTAL MEMORY)
     62 /END/
```

2.5. Lambda = 0.01

```

 SIMULOG    ***   QNAP2   ***   ( 15-09-2000   ) V 9.4
 (C)  COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986


***SIMULATION WITH SPECTRAL METHOD ***
... TIME =       10000000.00  , NB SAMPLES =      512  , CONF. LEVEL = 0.95
(0I0A03)  ==>WARNING (OUTPUT) : SOME CONFIDENCE INTERVALS ARE NOT
    AVAILABLE
                                    FOR QUEUE ... RT        . 0 ASSUMED THEN
*****************************************************************
*   NAME   *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************
*          *           *           *           *           *           *
*SOURCE1   * 100.5     * 1.000     * 1.000     * 100.5     *      99539*
*    +/-   *0.8096     *0.0000E+00*0.0000E+00*0.8096     *           *
*          *           *           *           *           *           *
*STATION1  * 10.06     *0.3585     *0.5595     * 15.69     *     356520*
*    +/-   *0.3459E-01*0.4793E-02*0.8474E-02*0.1793     *           *
*          *           *           *           *           *           *
*STATION2  * 15.07     *0.1618     *0.1933     * 18.01     *     107354*
*    +/-   *0.7734E-01*0.3221E-02*0.3356E-02*0.1568     *           *
*          *           *           *           *           *           *
*STATION3  * 20.08     *0.5003     * 1.003     * 40.27     *     249165*
*    +/-   *0.1139     *0.5299E-02*0.2192E-01*0.6899     *           *
*          *           *           *           *           *           *
*RT        *0.0000E+00*0.0000E+00* 1.756     * 176.4     *      99537*
*    +/-   *----------*0.0000E+00*0.3195E-01* 2.645     *           *
*          *           *           *           *           *           *
*****************************************************************
... END OF SIMULATION ...


```

```
                MEMORY USED:         9011 WORDS OF 4 BYTES
                  (   0.06   %OF TOTAL MEMORY)
      62 /END/
```

2.6. Lambda = 0.012

```
 SIMULOG    ***  QNAP2  ***   ( 15-09-2000  ) V 9.4
 (C)  COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986


***SIMULATION WITH SPECTRAL METHOD ***
... TIME =      10000000.00  , NB SAMPLES =     512  , CONF. LEVEL = 0.95
(0I0A03)  ==>WARNING (OUTPUT) : SOME CONFIDENCE INTERVALS ARE NOT
    AVAILABLE
                                      FOR QUEUE ... RT        . 0 ASSUMED THEN
*****************************************************************
*   NAME   *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************
*          *          *          *          *          *          *
*SOURCE1   * 83.83     * 1.000    * 1.000    * 83.83    *    119287*
*   +/-    *0.4617    *0.0000E+00*0.0000E+00*0.4617    *          *
*          *          *          *          *          *          *
*STATION1  * 10.02     *0.4280    *0.7489    * 17.54    *    427002*
*   +/-    *0.4339E-01*0.3972E-02*0.1312E-01*0.2400    *          *
*          *          *          *          *          *          *
*STATION2  * 15.04     *0.1929    *0.2399    * 18.70    *    128274*
*   +/-    *0.1014    *0.3470E-02*0.3833E-02*0.1889    *          *
*          *          *          *          *          *          *
*STATION3  * 20.07     *0.5996    * 1.506    * 50.42    *    298726*
*   +/-    *0.9515E-01*0.5114E-02*0.4256E-01* 1.157    *          *
*          *          *          *          *          *          *
*RT        *0.0000E+00*0.0000E+00* 2.495    * 209.2    *    119285*
*   +/-    *----------*0.0000E+00*0.5414E-01* 3.917    *          *
*          *          *          *          *          *          *
*****************************************************************
... END OF SIMULATION ...


                MEMORY USED:       10451 WORDS OF 4 BYTES
                  (   0.07   %OF TOTAL MEMORY)
      62 /END/
```

2.7. Lambda = 0.014

```
 SIMULOG    ***  QNAP2  ***   ( 15-09-2000  ) V 9.4
 (C)  COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986


***SIMULATION WITH SPECTRAL METHOD ***
... TIME =      10000000.00  , NB SAMPLES =     512  , CONF. LEVEL = 0.95
```

```
(0I0A03)  ==>WARNING (OUTPUT) : SOME CONFIDENCE INTERVALS ARE NOT
    AVAILABLE
                                        FOR QUEUE ... RT        . 0 ASSUMED THEN
*****************************************************************
*   NAME   *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************
*          *           *          *          *          *          *
*SOURCE1   * 71.80     * 1.000    * 1.000    * 71.80    *    139271*
*   +/-    *0.3203     *0.0000E+00*0.0000E+00*0.3203     *          *
*          *           *          *          *          *          *
*STATION1  * 10.01     *0.4981    *0.9938    * 19.98    *    497328*
*   +/-    *0.4522E-01*0.4783E-02*0.2962E-01*0.4670     *          *
*          *           *          *          *          *          *
*STATION2  * 15.05     *0.2253    *0.2914    * 19.47    *    149630*
*   +/-    *0.7755E-01*0.5085E-02*0.8364E-02*0.2663     *          *
*          *           *          *          *          *          *
*STATION3  * 20.05     *0.6973    * 2.320    * 66.72    *    347694*
*   +/-    *0.1013     *0.6335E-02*0.9670E-01* 2.461     *          *
*          *           *          *          *          *          *
*RT        *0.0000E+00*0.0000E+00* 3.605    * 258.8    *    139266*
*   +/-    *----------*0.0000E+00*0.1152     * 11.72    *          *
*          *           *          *          *          *          *
*****************************************************************
... END OF SIMULATION ...


              MEMORY USED:       11467 WORDS OF 4 BYTES
                (   0.08   % OF TOTAL MEMORY)
     62 /END/
```

2.8. Lambda = 0.016

```

 SIMULOG    ***  QNAP2  ***   ( 15-09-2000  ) V 9.4
 (C)  COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986


***SIMULATION WITH SPECTRAL METHOD ***
... TIME =      10000000.00  , NB SAMPLES =     512  , CONF. LEVEL = 0.95
(0I0A03)  ==>WARNING (OUTPUT) : SOME CONFIDENCE INTERVALS ARE NOT
    AVAILABLE
                                        FOR QUEUE ... RT        . 0 ASSUMED THEN
*****************************************************************
*   NAME   *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************
*          *           *          *          *          *          *
*SOURCE1   * 62.77     * 1.000    * 1.000    * 62.77    *    159300*
*   +/-    *0.3067     *0.0000E+00*0.0000E+00*0.3067     *          *
*          *           *          *          *          *          *
*STATION1  *10.000     *0.5703    * 1.334    * 23.39    *    570285*
*   +/-    *0.3805E-01*0.4304E-02*0.3496E-01*0.5257     *          *
*          *           *          *          *          *          *
```

```
20  *STATION2  * 15.04     *0.2575     *0.3458     * 20.19     *     171249*
21  *    +/-   *0.7931E-01*0.2896E-02*0.5812E-02*0.2008     *           *
22  *          *          *          *          *           *           *
23  *STATION3  * 19.99     *0.7976     * 3.863     * 96.82     *     399032*
24  *    +/-   *0.8934E-01*0.5281E-02*0.1751     * 3.891     *           *
25  *          *          *          *          *           *           *
26  *RT        *0.0000E+00*0.0000E+00* 5.543     * 348.0     *     159294*
27  *    +/-   *----------*0.0000E+00*0.2048     * 11.96     *           *
28  *          *          *          *          *           *           *
29  ************************************************************************
30  ... END OF SIMULATION ...
31
32
33                MEMORY USED:        12003 WORDS OF 4 BYTES
34                  (   0.08   % OF TOTAL MEMORY)
35      62 /END/
```

2.9. Lambda = 0.018

```
1
2   SIMULOG    ***   QNAP2   ***   ( 15-09-2000   ) V 9.4
3   (C)  COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986
4
5
6   ***SIMULATION WITH SPECTRAL METHOD ***
7   ... TIME =       10000000.00  , NB SAMPLES =     512  , CONF. LEVEL = 0.95
8   (0I0A03)  ==>WARNING (OUTPUT) : SOME CONFIDENCE INTERVALS ARE NOT
        AVAILABLE
9                                      FOR QUEUE ... RT        . 0 ASSUMED THEN
10  ************************************************************************
11  *   NAME   *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
12  ************************************************************************
13  *          *          *          *          *           *           *
14  *SOURCE1   * 55.56     * 1.000    * 1.000    * 55.56     *     179980*
15  *    +/-   *0.2837     *0.0000E+00*0.0000E+00*0.2837     *           *
16  *          *          *          *          *           *           *
17  *STATION1  * 10.02     *0.6434    * 1.792    * 27.89     *     642394*
18  *    +/-   *0.5266E-01*0.5668E-02*0.4936E-01*0.6132     *           *
19  *          *          *          *          *           *           *
20  *STATION2  * 15.02     *0.2895    *0.4064    * 21.09     *     192678*
21  *    +/-   *0.7921E-01*0.3570E-02*0.7108E-02*0.2122     *           *
22  *          *          *          *          *           *           *
23  *STATION3  * 19.98     *0.8984    * 9.004    * 200.2     *     449715*
24  *    +/-   *0.7218E-01*0.7023E-02*0.8957     * 20.37     *           *
25  *          *          *          *          *           *           *
26  *RT        *0.0000E+00*0.0000E+00* 11.20     * 622.4     *     179978*
27  *    +/-   *----------*0.0000E+00*0.9319     * 50.90     *           *
28  *          *          *          *          *           *           *
29  ************************************************************************
30  ... END OF SIMULATION ...
31
32
```

```
                MEMORY USED:        16131 WORDS OF 4 BYTES
                  (   0.11   % OF TOTAL MEMORY)
      62 /END/
```

2.10. Lambda = 0.019

```

 SIMULOG    ***  QNAP2  ***   ( 15-09-2000  ) V 9.4
 (C)  COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986


***SIMULATION WITH SPECTRAL METHOD ***
... TIME =      10000000.00  , NB SAMPLES =     512  , CONF. LEVEL = 0.95
(0I0A03)  ==>WARNING (OUTPUT) : SOME CONFIDENCE INTERVALS ARE NOT
   AVAILABLE
                                      FOR QUEUE ... RT        . 0 ASSUMED THEN
*****************************************************************
*   NAME   *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************
*          *          *          *          *          *          *
*SOURCE1   * 52.72     * 1.000    * 1.000    * 52.72    *    189698*
*   +/-    *0.2644    *0.0000E+00*0.0000E+00*0.2644    *          *
*          *          *          *          *          *          *
*STATION1  * 10.00     *0.6784    * 2.115    * 31.18    *    678296*
*   +/-    *0.3564E-01*0.4126E-02*0.4135E-01*0.5228    *          *
*          *          *          *          *          *          *
*STATION2  * 15.01     *0.3059    *0.4416    * 21.67    *    203807*
*   +/-    *0.7516E-01*0.3263E-02*0.7010E-02*0.1877    *          *
*          *          *          *          *          *          *
*STATION3  * 20.00     *0.9491    * 17.24    * 363.3    *    474480*
*   +/-    *0.6769E-01*0.7140E-02* 2.709    * 58.54    *          *
*          *          *          *          *          *          *
*RT        *0.0000E+00*0.0000E+00* 19.79    * 1043.    *    189687*
*   +/-    *----------*0.0000E+00* 2.742    * 144.3    *          *
*          *          *          *          *          *          *
*****************************************************************
... END OF SIMULATION ...


                MEMORY USED:        18915 WORDS OF 4 BYTES
                  (   0.13   % OF TOTAL MEMORY)
      62 /END/
```