Performance Modeling of Computer Systems and Networks

Prof. Vittoria de Nitto Personè

Case study 1

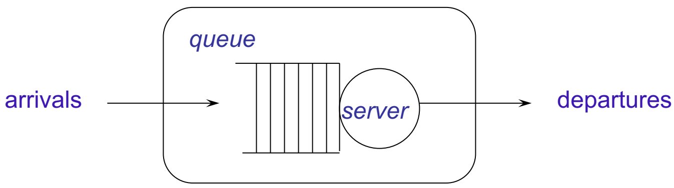
A singol server queue

Università degli studi di Roma Tor Vergata
Department of Civil Engineering and Computer Science Engineering

1

Discrete-event simulation
Trace-driven simulation

Single server queue



Consider *n*=10 job and given arrival and service times:

Arrival times:

15 47 71 111 123 152 166 226 310 320

Service times:

43 36 34 30 38 40 31 29 36 30

Prof. Vittoria de Nitto Personè 2

2

Algorithm 2: *trace-driven simulation*

$c_0 = 0.0;$ /* assumes that $a_0 = 0.0$ */
$i = 0;$
while (more jobs to process) {
    i++;
    a_i = GetArrival();
    if ($a_i < c_{i-1}$) $d_i = c_{i-1} - a_i$;
    else $d_i = 0.0$;
    s_i = GetService();
    $c_i = a_i + d_i + s_i$;
    }
    $n = i$;
return $d_1, d_2, \ldots, d_n$;

Read data from a file

ssq1.c

```
#include <stdio.h>

#define FILENAME   "ssq1.dat"    /* input data file */
#define START      0.0

  double GetArrival(FILE *fp)   /* read an arrival time */
{   double a;
    fscanf(fp, "%lf", &a);
    return (a);}

double GetService(FILE *fp)   /* read a service time */
{   double s;
    fscanf(fp, "%lf\n", &s);
    return (s);}
```

```
int main(void)
{  FILE   *fp;                  /* input data file */
   long   index     = 0;       /* job index */
   double arrival   = START;   /* arrival time*/
   double delay;               /* delay in queue*/
   double service;              /* service time*/
   double wait;                 /* delay + service*/
   double departure = START;    /* departure time*/
   struct {                     /* sum of ...   */
       double delay;     /*delay times */
       double wait;      /*wait times*/
       double service;   /*service times */
       double interarrival; /*   interarrival times */
   } sum = {0.0, 0.0, 0.0};
```



```
fp = fopen(FILENAME, "r");
if (fp == NULL) {
   fprintf(stderr,"Cannot open input file %s\n",FILENAME);
   return (1);  }
while (!feof(fp)) {
   index++;
   arrival      = GetArrival(fp);
   if (arrival < departure)
       delay   = departure - arrival; /* delay in queue   */
   else  delay       = 0.0;            /* no delay   */
   service = GetService(fp);
   wait   = delay + service;
   departure     = arrival + wait;   /* time of departure */
   sum.delay    += delay;
   sum.wait     += wait;
   sum.service += service;  }
```

```
sum.interarrival = arrival - START;

printf("\nfor %ld jobs\n", index);
printf("average interarrival time = %6.2f\n", sum.interarrival / index);
printf("   average service time .... = %6.2f\n", sum.service / index);
printf("   average delay .......... = %6.2f\n", sum.delay / index);
printf("   average wait ........... = %6.2f\n", sum.wait / index);

fclose(fp);
return (0);}
```


	i	1	2	3	4	5	6	7	8	9	10
read from file	a_i	15	47	71	111	123	152	166	226	310	320
from algorithm	d_i	0	11	23	17	35	44	70	41	0	26
read from file	s_i	43	36	34	30	38	40	31	29	36	30

- average interarrival time $\bar{r} = \frac{a_n}{n} = \frac{320}{10} = 32.0$ sec
- average service time $\bar{s} = 34.7$ sec
- arrival rate $\frac{1}{\bar{r}} \approx 0.031$ job/sec
- service rate $\frac{1}{\bar{s}} \approx 0.029$ job/sec
- traffic intensity $\frac{\bar{s}}{\bar{r}} = 1.084375$
- utilization $\bar{x} = \frac{n}{c_n}\bar{s} = 0.92287$

Insight: The server is not quite able to process jobs at the rate they arrive on average.

	i	1	2	3	4	5	6	7	8	9	10
read from file	a_i	15	47	71	111	123	152	166	226	310	320
from algorithm	d_i	0	11	23	17	35	44	70	41	0	26
read from file	s_i	43	36	34	30	38	40	31	29	36	30
completion time		58	94	128	158	196	236	267	296	346	376

Job-averaged statistics

- average delay and service time $\bar{d} = 26.7$, $\bar{s} = 34.7$ sec
  therefore, the average wait time is:

$$\bar{w} = \bar{d} + \bar{s} = 26.7 + 34.7 = 61.4 \text{ sec}$$

- verification is a difficult step

$\downarrow$

Consistency check:
used to verify that a simulation satisfies known equations

- compute $\bar{w}, \bar{d}, \bar{s}$ *independently*
- then verify that $\bar{w} = \bar{d} + \bar{s}$

$$\bar{l} = \frac{1}{\tau}\int_0^\tau l(t)dt \qquad \bar{q} = \frac{1}{\tau}\int_0^\tau q(t)dt \qquad \bar{x} = \frac{1}{\tau}\int_0^\tau x(t)dt$$

With $\tau = c_{10} = 376$

$$\bar{l} = \frac{n}{c_n}\bar{w} \qquad \bar{l} = \frac{10}{376}61.4 = 1.633 \qquad \bar{q} = 0.710 \qquad \bar{x} = 0.923$$

The average of numerous random observations (samples) of the number in the service node should be close to $\bar{l}$.
(Same holds for $\bar{q}, \bar{x}$)

Service times:

43 36 34 30 38 40 31 29 36 30

63 16 54 10 18 60 51 9 56 10	*alta variabilità*
9 10 10 16 18 51 54 56 60 63	*fair*
63 60 56 54 51 18 16 10 10 9	*unfair*

Dati originari
for 10 jobs
average interarrival time = 32.00
average service time .... = 34.70
average delay .... = 26.70
average wait .... = 61.40

$$\bar{l} = \frac{10}{376} 61.4 = 1.633$$

Dati con alta variabilità
for 10 jobs
average interarrival time = 32.00
average service time .... = 34.70
average delay .... = 32.70
average wait .... = 67.40

$$\bar{l} = \frac{10}{376} 67.4 = 1.793$$

Ordinamento unfair
for 10 jobs
average interarrival time = 32.00
average service time .... = 34.70
average delay .... = 77.70
average wait .... = 112.40

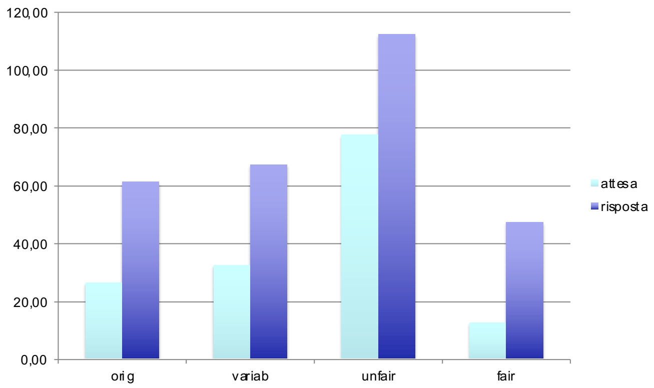
$$\bar{l} = \frac{10}{376} 112.4 = 2.989$$

Ordinamento fair
for 10 jobs
average interarrival time = 32.00
average service time .... = 34.70
average delay .... = 12.80
average wait .... = 47.50

$$\bar{l} = \frac{10}{376} 47.50 = 1.263$$

Prof. Vittoria de Nitto Personè 13

13



Prof. Vittoria de Nitto Personè 14

14

Case Study

- The owners of an ice cream shop are considering adding additional flavors and cone options
- But they are concerned about the effect of the resultant increase in service times on queue length

The case can be studied as a single-server queue

• ssq1.dat represents 1000 customer interactions at the shop (arrival times and the corresponding service times)
• for the study, the service times are sistematically increased (and decreased) by a common multiplicative factor

Exercise 1

By running ssq1 for the datafile ssq1.dat, the following can be observed:

Observed arrival rate $\frac{1}{\bar{r}} \approx 0.10$ job/sec

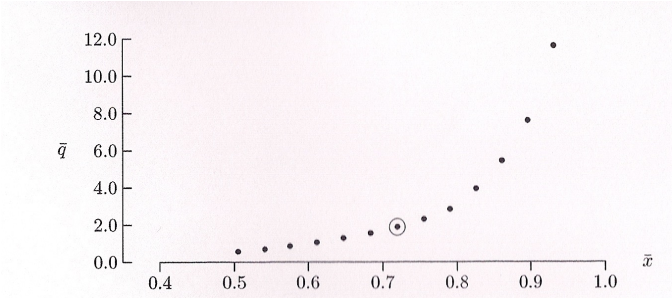
Observed service rate $\frac{1}{\bar{s}} \approx 0.14$ job/sec

- Modify the program to compute $\bar{l}, \bar{q}, \bar{x}$
- You will find $1 - \bar{x} \approx 0.28$

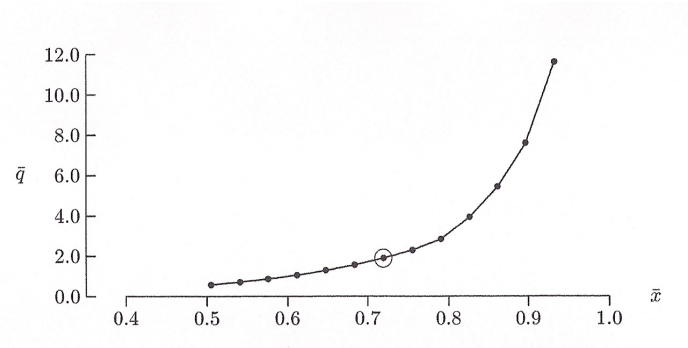
Despite this significant idle time (28%), enough jobs are delayed so that the average number in the queue is

$$\bar{q} \approx 2$$

15% ↑ service times 109% ↑ queue length
30% ↑ service times 518% ↑ queue length



- even a modest increase in service times will produce a significant increase in the average queue length
- nonlinear relationsheep between $\bar{q}, \bar{x}$ particularly pronounced when utilization approaches 1



- $\bar{q}, \bar{x}$ are continuous in the values → many additional points more
- few would question the validity of “connetting the dots”

Guidelines

- If the data have essentially no uncertainty and the resulting is smooth OK connecting the dots, but the original are left
- If the data have essentially no uncertainty but the resulting is not smooth more dots need to be generated
- If the dots correspond to ucertain data, NO interpolation! (approximation)
- If the data is inherently discrete, NEVER interpolation!

Exercises

- Ex 1.1.2 and Ex 1.1.3 on p.11 from textbook
- Study the program ssq1.c
- Run with ssq1.dat; analyze the results
- Generate a data file with the values on p.8
- Run with the new data, verify that the results confirm the expectations
- Ex 1.2.2 and Ex 1.2.3 on p.24 of the textbook