Introduction to CSIM

TELCOM2120 Network Performance
Instructor: Anotai Srikitja
Telecommunications Program
University of Pittsburgh

Introduction to CSIM

- Process-oriented simulation package based on C.
  - www.mesquite.com
- A set of libraries and procedures to implement necessary structures and operations in simulation.
- Model a system as a collection of processes which interact with each other through CSIM structures
  - Ex 1. Model a single queue with two processes: traffic generator and server.
  - Ex 2. Model a single queue with a packet process.
Example 1 (mm1_csim.c)

```c
#include "csim.h"                  /* Required header file */
FACILITY f;                        /* CSIM data structure */
sim()                               /* equivalent to main() in C */
{ create("sim");                  /* Required statement (exactly shown) */
f = facility("server");           /* Create a server facility */
while (simtime() < 2000.0)         /* Get simulation time */
{ hold(exponential(1.0));         /* Let simulation time to pass */
packet();                         /* Create a packet process */
} report();                        /* Generate output report */
}
packet()                           /* Can be any string */
{ create("packet");             /* Enter the facility queue */
reserve(f);                       /* Use server facility */
hold(exponential(0.5));          /* Use server facility */
release(f);                      /* Use server facility */
}
command to compile the program: csim mm1_csim.c -o mml_csim.out
```

Process

- Used to model active elements of a system.
- A process must have a `create()` statement.
- Can be in the following states:
  - Executing
  - Holding (letting time elapses)
  - Waiting (for some event to occur)
- A process is generated by another process – `packet()` is generated by `sim()`.
- Processes are executed in parallel (in simulated time) – packets are using the facility while `sim()` process keeps generating packets.
- Note: a `csim` process can have arguments but cannot return a value.
Facility

- Entities which processes “use” or occupy.
- \( f = \text{facility(“server”)} \) creates a single server facility.
- \( f = \text{facility_ms(“server”, n)} \) creates a facility with \( n \) servers
- When \( \text{reserve}(f) \) is called, the process enters the facility queue. The process will exit this statement only if it enters one of the idle servers in the facility.
- \( \text{hold()} \) to let simulation time pass.
- When \( \text{release}(f) \) is called, the process leaves the facility. If the process terminates without calling \( \text{release}(f) \), the server is permanently occupied and becomes inaccessible to other processes.

Example 2: M/M/3/5 (mm3_5.c)

```c
#include "csim.h" /* Required header file */
FACILITY f; /* CSIM data structure */
int blocked, num_arrive, num_wait;
double wait_time;

sim() /* equivalent to main() in C */
{
    create("sim"); /* Required statement (exactly shown) */
    f = facility_ms("multi server", 3); /* Create a facility with three servers */
    while (simtime() < 2000.0) /* Get simulation time */
    {
        hold(exponential(1.0)); /* Let simulation time to pass */
        packet(); /* Create a packet process */
    }
    report(); /* Generate output report on the screen */
    printf("blocking rate %f, mean waiting time %f\n", blocked/num_arrive, wait_time/num_wait);
}
```
M/M/3/5 (cont.)

```c
packet()
{
    double enqueue_time;
    create("packet"); /* can be any string */

    num_arrive++;  
    if (qlength(f) == 3)  
        blocked++;
    else
    {
        enqueue_time = simtime();
        reserve(f); /* Enter the facility queue */
        num_wait++;
        wait_time += simtime() – enqueue_time;
        hold(exponential(0.5)); /* Use server facility */
        release(f); /* Leave the facility */
    }

}
```

**Event**

EVENT is used to synchronize and control interactions between processes.

EVENT ev;
```
    ev = event("event_name");
```

Event has two states – occurred and not occurred.

wait(ev); The process is suspended and waits for event ev to occur.

set(ev); Make event ev occur. The process waiting for this event will proceed.
Mailbox

? Allows for the synchronous exchange of data between CSIM processes. Mailbox is basically a FIFO queue.
? Any process may send a message to any mailbox and any process may attempt to receive a message from any mailbox.
? Used to model a queue – packets are sent to a mailbox and the server fetches packets from the mailbox.
? Data can be a single integer, pointer to some other data object.

Mailbox Example

MBOX mb; /* Declare a mailbox variable */
mb = mailbox("My message queue"); /* Initialize */
....
struct
  { int pkt_size;
  } pkt;
struct pkt *p; /* Data to send to mb */
p = (struct pkt *) malloc(sizeof(*p));
p->pkt_size = exponential(1.0);
send(mb, (long) p); /* Send data to mb */
....
struct pkt *p;
receive(mb, (long *) &p); /* Fetch data from mb, process waits for the data if nothing in mb */
Table

? Collects statistics on a sequence of discrete values – interarrival time, service time, delay.

? Table tbl; /* Declare a table variable
  tbl */
  tbl = table("table_name"); /* Initialize tbl */
  record((double) delay, tbl); /* Record delay value */

  table_min(tbl); /* minimum value */