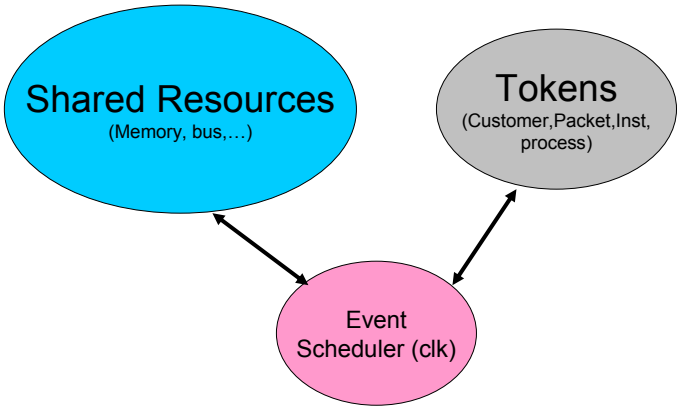
SMPL Simulation Language

Ali Mohammad Zareh Bidoki

From Simulating Computer Systems: Techniques and Tools, M.H. MacDougall , MIT Press Series in Computer Systems, 1987

Simulation System Components



Simulation Language & Operating System

- Resource Management
- Event Scheduling
- Queuing
- Priority
- Preemption

Simulating M/M/1 with SMPL

```
//----- Include files -------------------------------------------------------
#include <stdio.h>      // Needed for printf()
#include "smpl.h"       // Needed for SMPL
//====== Main program
      ============================================================
void main(void)
{
 real Ta = 200;        // Mean interarrival time (seconds)
 real Ts = 100;        // Mean service time
 real te = 1.0e6;      // Total simulation time
 int customer = 1;     // Customer id (always '1' for this simulation)
 int event;            // Event (1 = arrival, 2 = request, 3 = completion)
 int server;           // Handle for server facility
 // Initialize SMPL subsystem
 smpl(0, "M/M/1 Queue");
 // Initialize server facility (single server)
 server=facility("server", 1);
 // Schedule arrival event at time 0 to kick-off simulation
 schedule(1, 0.0, customer);
```

Simulating M/M/1 with SMPL

```
// Loop while simulation time is less than te
 while (time() < te)
 {
  // "Cause" the next event on the event list
  cause(&event,&customer);
  // Process the event
  switch(event)
  {
   case 1:  // *** Arrival
     schedule(2, 0.0, customer);
     schedule(1, expntl(Ta), customer);
     break;
   case 2:  // *** Request Server
     if (request(server, customer, 0) == 0)
       schedule(3, expntl(Ts), customer);
     break;
   case 3:  // *** Release server
     release(server, customer);
     break;
  }
 }
 // Output standard SMPL report
 report();
}
```


SMPL

- SMPL is a set of C functions for building event-based, discrete-event simulation models. SMPL was written by M. H. MacDougall and is described in *Simulating Computer Systems, Techniques and Tools*, The MIT Press, 1987.

SMPL Entities

- Facilities
- Tokens
- Events

Facilities

- A facility typically represents some work-performing resource of system being modeled, such as CPU and memory in computer system, BUS in computer networks and lock in operating system.
- SMPL provides functions to define facilities, reserve, release and preempt them and interrogate their status.
- The Interconnection of facilities is not explicit, but can be determined by the model's routing of tokens between facilities.
- A system comprises a collection of interconnected facilities.

Tokens

- Tokens represents the active entities of the system.
- The dynamic behavior of the system is modeled by the movement of tokens through set of facilities.
- A token may represent a task in a computer system model, a packet in communication model or memory access in a memory bus subsystem model.
- In SMPL a token may reserve (preempt) a facility or schedule activity of various duration.
- A token can be a single integer (customer id), an structure (enter time, size,…)or object (packet).

Events

- A change of state of any system entity, active or passive is an event.
- Some events are Task arrival , CPU completion interval, Process departure…

SMPL Functions(1/8)

- Smpl (m,s)
  - int m; char *s
  - Initialize the simulation subsystem for a simulation run (clear data structures, initialize clk to zero,..).
  - When m=1, SMPL provides an interactive interface to model execution.We use m=0
  - S shows model name
- Reset()
  - Clear all accumulated measurements (Not clk)
- Usage
  - Making multiple simulation runs during one instance of simulation program with different parameters (How response time varies when arrival rate increase)
  - Replicated runs

SMPL Functions(2/8)

- F=facility (s,n)
  - Char *s; int n
  - This function creates and names a facility with n servers.
- R=request (f,tkn,pri)
  - int f,tkn,pri
  - It requests that a server of facility f be reserved for token designed by token tkn with pri priority.
  - If the facility is not busy, a server is reserved for the requesting tokens (The first idle server will choose)
  - Each facility has a queue. When facility is busy a queue entry is constructed for the request.
  - A request which initially finds the facility busy is called blocked request (in compare with suspend-preempted-request).

SMPL Functions(3/8)

- R=preempt (f,tkn,pri)
  - int f,tkn,pri
  - If the facility is not busy and or all the tokens in busy facility have greater priority it executed like request function.
  - If the facility is busy the server with the lowest priority reserving token is located. If this priority is equal or grater than requester the request is queued and 1 is returned.
  - Else located token is suspended (preempted) and the server will be reserved for requestor.
  - The suspended token is put on top of the same priority queue with its remaining time.
- What is its difference with OS?
  - The variable, …

SMPL Functions(4/8)

- Release (f,tkn)
  - int f,tkn
  - This function release the server facility f reserved by token tkn.
  - Next, the facility queue is examined and if it sn't not empty the entry at its head is dequeued and associated event is rescheduled at current time.
  - If the entry is for a preempted request the released server is reserved for the dequeued token and the associated event is rescheduled to occur at a time equal to the current time plus remaining time.

SMPL Functions(5/8)

- N=inq (f)
  - Number of tokens currently in queue
- R=status (f)
  - The facility status (busy or idle)
- U=U (f)
  - Mean facility utilization
- B=B (f)
  - Mean busy period
- L=lq(f)
  - Mean queue length

SMPL Functions(6/8)

- Schedule (ev,te,tkn)
  - int ev,tkn; real te
  - This function schedule an event. Ev is event number, te is the inter event time and tkn is a token associated with event.
  - Then an event entry for this event is constructed
- Cause (ev,tkn)
  - Int *ev, *tkn
  - Removes the entry at the head of event list, advances simulation time to the event occurrence time and return the event number ev and token tkn.

SMPL Functions(7/8)

- Tkn=cancel (ev)
  - int tkn;
  - Search in event list for event ev and remove it from list and return its token.
- t=real time()
  - Return current simulation time.
- r= real ranf()
  - Returns a psuedo-random variate uniformly distributed in the range 0 ,1.

SMPL Functions(8/8)

- i=stream (n)
  - int n
  - Select a stream (seed) (1=<n<=15)
  - or identify the selected stream (n=0)
- R=expntl (x)
- R= erlang (x,s)
- R=hyperx(x,s)
- R=uniform (a,b)
  - Real a,b
- K=ranfom (i,j) int i,j
- R=normal (x,s)

A Queuing Network Simulation Model (closed system)

- The system comprises a CPU and four disks.
- There are two types of tasks : class 0 (n0) and class 1 (n1) that class 1 has higher priority.
- The CPU execution time for class 0 is exponentially distributed with mean 10 ms
- The CPU execution time for class 1 is exponentially distributed with mean 5 ms
- Each disk service time is erlang distribution with mean 30ms and standard deviation 7.5.
- The requests are distributed randomly and uniformly across all four disks.

SMPL code

```
define n0 6
#define n1 3
#define nt n0+n1
#define nd 4
#define n0 6
#define qd 1

struct token {
        int cls;  //class
        int un;   //disk unit
        real ts;  //tour start time

} task [nt+1];
int disk[nd+1];
int cpu, nts=1000;
real tc[2]={10.0,5.0};
real td=30.0,sd=7.5;
```

SMPL code

```
void main(void)
{
    int i,j,event,n[2];
    real t,s[2];
    struct token *p;
    n[0]=n[1]=0;
    s[0]=s[1]=0.0;
    for (i=1;i<=nt;i++) task[i].cls=(i>n0)? 1:0;
    // Initialize SMPL subsystem
    smpl(0, "Central Server Model");
    // Initialize server facility (single server)
    cpu=facility("CPU", 1);
    for (i=1;i<=nd;i++) disk[i]=facility("disk", 1);
    for (i=1;i<=nt;i++) schedule(1,0.0,i);
```



```
while(nts){
        cause (&event,&i); p=&task[i];
        switch(event)
        {
        case 1:  // begin tour
                p->ts=time();  schedule(2, 0.0,i);
                break;
        case 2:  // *** Request CPU
                j=p->cls;
                if (preempt(cpu, i, j) != qd)
                        schedule(3, expntl(tc[j]),i);
                break;
        case 3:  // *** Release cpu
                release(cpu,i); p->un=random(1,nd);
                schedule(4,0.0,i);
                break;
        case 4:  // *** Request disk
                if (request(disk[p->un], i, 0) != qd)
                        schedule(5, erlang(td,sd),i);
                break;
        case 5:  // *** Release disk
                release(disk[p->un], i); j=p->cls;
                t=time(); s[j]+=t-p->ts; p->ts=t; n[j]++;
                schedule(1,0.0,i); nts--;
                break;
                        }
    }
```

SMPL code

```
printf("class 0 tour time=%.2f\n",s[0]/n[0]);
printf("class 1 tour time=%.2f\n",s[1]/n[1]);
}

}
```