upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

Área de Ingeniería Telemática

Simulación de eventos discretos

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Grupo de Redes, Sistemas y Servicios Telemáticos

Simulación

- Imitación del funcionamiento de un proceso real con el tiempo
- Se necesita un *modelo* del sistema real
- Se genera una historia artificial de sucesos en el sistema y sus repercusiones
- Simulación a mano o por ordenador
- Se obtienen medidas de prestaciones
- Utilidades
  - Contestar a preguntas tipo “what if” sobre el sistema real
  - Estudio de sistemas en fase de diseño (no existe el real)
- Si el modelo es muy simple se puede resolver matemáticamente
- Modelos realistas son demasiado complejos para una solución analítica

Componentes

- Entidad (*entity*)
  - Un objeto de interés en el sistema
- Atributo (*attribute*)
  - Propiedad de una entidad
- Actividad (*activity*)
  - Un periodo de tiempo de una longitud especificada

- Ejemplo: Un banco
  - Los clientes podrían ser entidades
  - Su saldo en cuenta sería un atributo
  - Hacer depósitos una actividad

Estado del sistema

- Grupo de variables necesarias para describir el sistema en un momento cualquiera, en relación con los objetivos del estudio

- Ejemplo: banco
  - Número de cajeros ocupados
  - Número de clientes esperando en cola
  - Instante en que llegará el siguiente cliente

- **Evento**: suceso instantáneo que puede cambiar el estado del sistema

- Ejemplo:
  - Llegada de un nuevo cliente (exógeno)
  - Cliente termina de ser atendido (endógeno)

Tipos de sistemas

- Sistema **discreto**
  - Las variables de estado cambian solo en un conjunto discreto de puntos en el tiempo
  - Ejemplo: banco
- Sistema **continuo**
  - Las variables de estado cambian de forma continua con el tiempo
  - Ejemplo: nivel de agua en un pantano

Modelo del sistema

- Simplificación del sistema
- Considera solo los aspectos que afectan al problema en estudio
- Debe ser lo suficientemente detallado para poderse obtener conclusiones que apliquen al sistema real

- Tipos:
  - Discreto/Continuo
  - Estático/Dinámico
  - Determinista/Estocástico

- Nos interesan los estocásicos, dinámicos y discretos.

Simulación de eventos discretos

- Las variables de estado cambian solo en un conjunto discreto de puntos en el tiempo
- Empleo de métodos numéricos
  - En vez de métodos analíticos
  - El modelo se “corre” en vez de se “resuelve”

Future (pending) Events List (FEL)

- Cada evento contiene el instante de tiempo en que sucede
- La FEL contiene los eventos planificados para este instante o posteriores aún sin procesar
- Ordenados por instante de tiempo de menor a mayor

```
Mientras queden eventos en la FEL
    Retira el primero
    Avanza la variable de reloj hasta el instante del evento
    Procesa el evento: puede modificar el estado del sistema e
      introducir otros eventos futuros en la FEL
```

Ejemplo

- Servidor web
- Recibe peticiones cada X segundos
- X es una v.a. exponencial de media 3
- La petición es de un fichero que cuesta Y segundos enviarlo
- Y es una v.a. exponencial de media 5
- El servidor es capaz de antender 2 peticiones a la vez (2 CPUs)
- Si llegan peticiones mientras las 2 CPUs están ocupadas se pierden
- ¿Cuántas peticiones se pierden cada hora?

OMNeT++

http://www.omnetpp.org/

Introducción

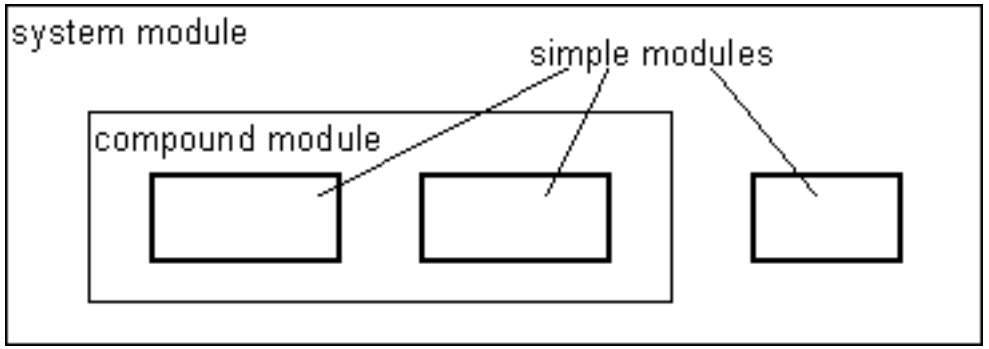
- C++ class library
  - Simulation kernel (eventos discretos)
  - Utility classes (number generation, statistics collection, etc)
- Permite crear simuladores de eventos discretos
- Al crear un simulador creas un programa
- La librería gestiona la lista de eventos
- Creamos los módulos que generan y consumen los eventos
- Eventos → mensajes:
  - Tienen una referencia al módulo que debe procesarlo
  - Los módulos “se pasan mensajes”, es decir, planifican eventos que debe procesar otro módulo

Introducción

- El main del programa lo pone la librería
- Funcionamiento básico:
  - Inicializar los módulos (crear objetos y llamar a funciones de inicialización)
  - Repetir
    - Obtener siguiente evento/mensaje
    - Procesar el evento con la función de manejar mensajes del módulo/objeto al que hace referencia el mismo
  - Finalizar los módulos

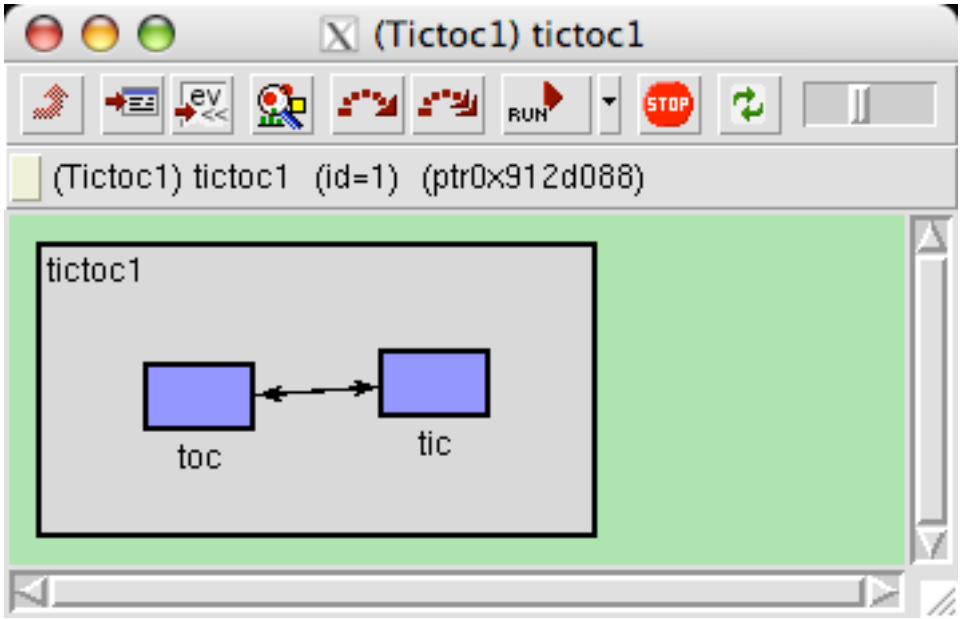
¿ Qué se desarrolla ?

- Módulos simples:
  - Escritos en C++
  - Un fichero (.ned) describe sus parámetros
- Módulos compuestos:
  - Agregación de otros módulos (simples o compuestos)
  - Interconexión de esos módulos
  - Asignación de parámetros a los mismos
- Módulo del sistema:
  - A.k.a. Network
  - Simple o compuesto



Ejemplo

- `samples/tictoc`
  - `tictoc1`



Ejemplo

- tictoc1.ned:

```
simple Txc1
    gates:
        in: in;
        out: out;
endsimple

//
// Two instances (tic and toc) of Txc1 connected both ways.
// Tic and toc will pass messages to one another.
//
module Tictoc1
    submodules:
        tic: Txc1;
        toc: Txc1;
    connections:
        tic.out --> delay 100ms --> toc.in;
        tic.in <-- delay 100ms <-- toc.out;
endmodule

network tictoc1 : Tictoc1
endnetwork
```

Ejemplo

- txc1.cc

```
class Txc1 : public cSimpleModule
{
  protected:
  // The following redefined virtual function holds the algorithm.
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};


// The module class needs to be registered with OMNeT++
Define_Module(Txc1);
```

Ejemplo

- txc1.cc

```
void Txc1::initialize()
{
    // Initialize is called at the beginning of the simulation.
      // To bootstrap the tic-toc-tic-toc process, one of the
  modules needs
    // to send the first message. Let this be `tic'.

    // Am I Tic or Toc?
    if (strcmp("tic", name()) == 0)
    {
            // create and send first message on gate "out".
  "tictocMsg" is an
       // arbitrary string which will be the name of the message
  object.
        cMessage *msg = new cMessage("tictocMsg");
        send(msg, "out");
    }
}
```

Ejemplo

- txc1.cc

```
void Txc1::handleMessage(cMessage *msg)
{
  // The handleMessage() method is called whenever a message arrives
  // at the module. Here, we just send it to the other module, through
  // gate `out'. Because both `tic' and `toc' does the same,
  // the message will bounce between the two.
    send(msg, "out");
}
```

Ejemplo

- Compilación:
  - Tengo `txc1.cc`

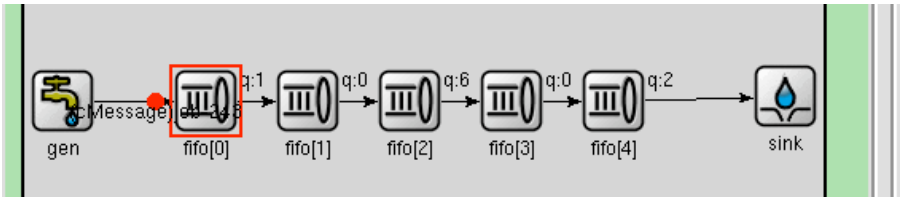
```
$ opp_makemake
$ make
```

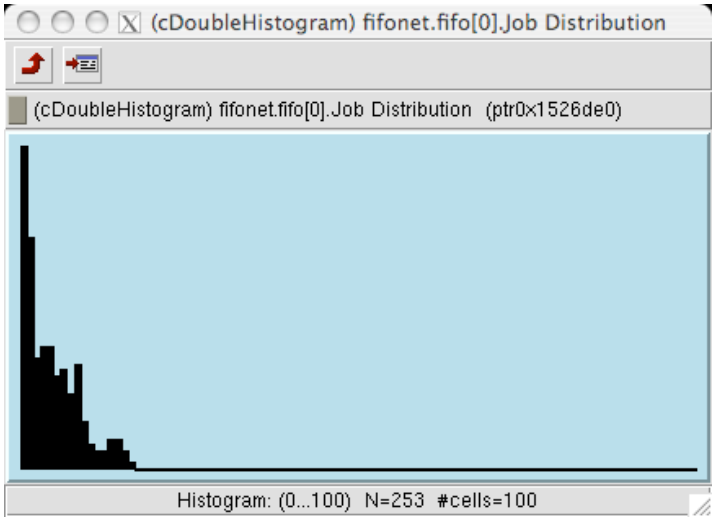
  - Fin
- Ejecución:
  - Tengo el ejecutable
  - Necesito:
    - tictoc1.ned
    - omnetpp.ini

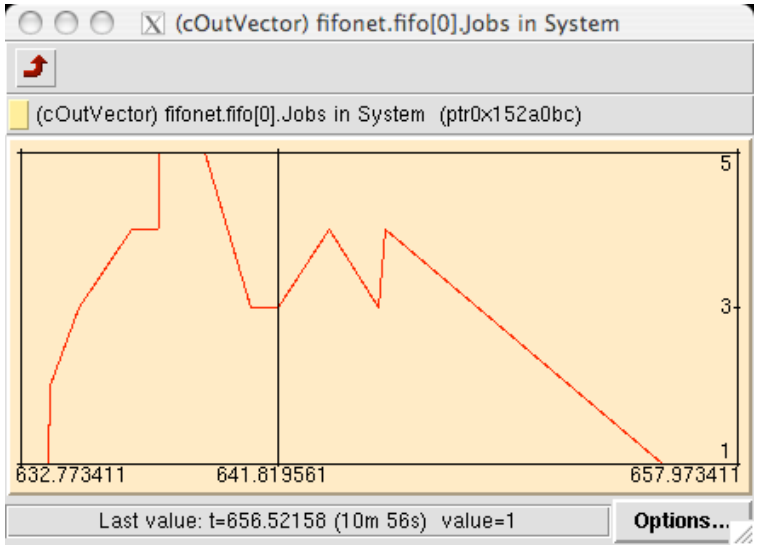
```
[General]
preload-ned-files=*.ned
network=tictoc1   # this line is for Cmdenv, Tkenv will
  still let you choose from a dialog
```

Ejemplo 2

- samples/queues
  - Histogramas
  - Vectores







Interfaz

- Gráfico: Tkenv (por defecto)
- Texto: Cmdenv (opp_makemake -u Cmdenv)

INET

- The INET Framework is suited for simulations of wired, wireless and ad-hoc networks
- Beyond IP and UDP/TCP there is 802.11, Ethernet, PPP, IPv6, OSPF, RIP, MPLS with LDP and RSVP-TE signalling, and several other protocols.
- Son un montón de módulos para OMNeT++

- Ejemplo (INETbasico)