


Introducción a C++ y Omnet++



Redes y Sistemas Distribuidos

Juan A. Fraire



Introducción a C++

(el lenguaje del Lab 3, 4 y 5)

Introducción

Lo Básico

Clases



Introducción

- **Superconjunto de C**
 - Cualquier programa legal de C es un programa legal de C++
- Es **compilado**
- Tiene **tipado fijo**
- Orientado a **objetos**
 - Objetos, clases, métodos
- No tiene **garbage collector**
 - Objetos creados como variables **bool**, **char**, **int...** se borran automáticamente
 - Objetos creados con **new**, se borran con **delete**
- Usa **{ }** para bloques (no indentación)

```
#include <iostream>
using namespace std;

// main() is where execution begins
int main() {
    // prints "Hello World in 2019"
    int year = 2019;
    cout << "Hello World in " << year;
    return 0;
}
```

Lo Básico

```
// statements ends with colons (;)
x = y;
y = y + 1;
add(x, y);

// statements can be in the same line
x = y; y = y + 1; add(x, y);

// Brackets defines logical blocks
{
    // prints Hello World
    cout << "Hello World";
    return 0;
}
```

- Tipos de variables

- **bool** (1 byte) true o false
 - **char** (1 byte) 127 a 127 / 0 a 255
 - **int** (4 bytes) -2147483648 a 2147483647
 - **unsigned int** (4 bytes) 0 a 4294967295
 - **float** (4 bytes) +/- 3.4e +/- 38
 - **double** (8 bytes) +/- 1.7e +/- 308
 - **void** sin tipo
- ```
int i, j, k; // definition of i, j, k
int d = 3; // definition and
float f = 5.1; // initializing d and f
char x = 'x'; // x has the value 'x'
```

# Lo Básico - Clases

---

## Declaración

```
#include <string.h>

using namespace omnetpp;

class ClassName : public masterClass {
private:
 int number = 0;
public:
 int function1(int input);
protected:
 void function2();
};
```

## Definición

```
int ClassName::function1(int input){
 number += input;
 function2();
 return number;
}

void ClassName::function2(){
 cout << "Current number: " << number << endl;
}
```

# Introducción a Omnet++

(el simulador del Lab 3, 4 y 5)

Introducción

Simulación Eventos Discretos

Omnet++

Modelo de Colas

---

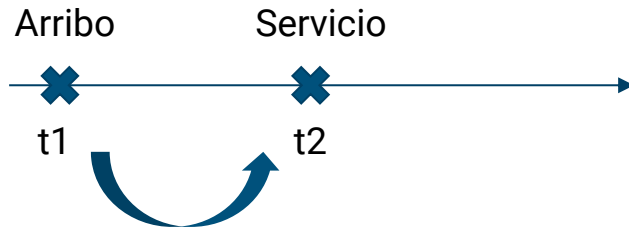
# Introducción

---

- Lab 0, 1, 2: **capa de aplicación** (cliente y servidor)
  - Perspectiva de **desarrollador**
  - La cosa tiene que **funcionar** (test, debug)
- Lab 3, 4 y 5: **capas inferiores** (transporte, red y enlace)
  - Perspectiva de **analista**
  - ~~○ La cosa tiene que funcionar (test, debug)~~
    - Implica desarrollos a nivel kernel
    - Requiere de decenas de hosts
    - Hace falta observar cientos de intercambios en poco tiempo
  - La cosa se tiene que **entender** (análisis, simulación)

# Introducción - Simulación

- Simulación de **eventos discretos**
  - El **tiempo** de simulación avanza en los eventos en una **cola de eventos**
  - Un evento suceden en momentos discretos de tiempo
  - El **estado** del sistema puede cambiar durante estos **eventos**

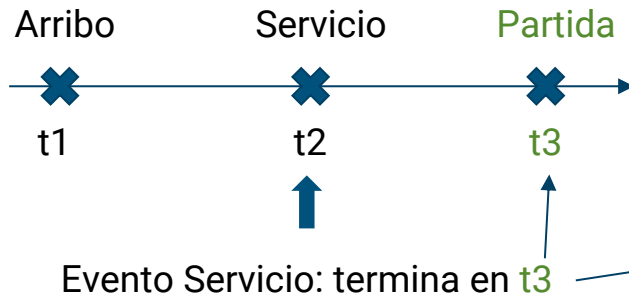


| Cola de eventos |          |
|-----------------|----------|
| t1              | Arribo   |
| t2              | Servicio |
|                 |          |



# Introducción - Simulación

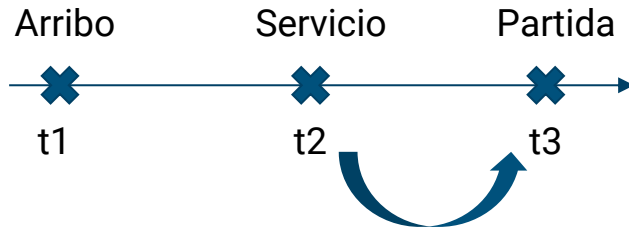
- Simulación de **eventos discretos**
  - El **tiempo** de simulación avanza en los eventos en una **cola de eventos**
  - Un evento suceden en momentos discretos de tiempo
  - El **estado** del sistema puede cambiar durante estos **eventos**



| Cola de eventos |          |
|-----------------|----------|
| t1              | Arribo   |
| ➡ t2            | Servicio |
| ➡ t3            | Partida  |

# Introducción - Simulación

- Simulación de **eventos discretos**
  - El **tiempo de simulación** avanza en los eventos en una **cola de eventos**
  - Un evento suceden en momentos discretos de tiempo
  - El **estado** del sistema puede cambiar durante estos **eventos**



| Cola de eventos |          |
|-----------------|----------|
| t1              | Arribo   |
| t2              | Servicio |
| t3              | Partida  |

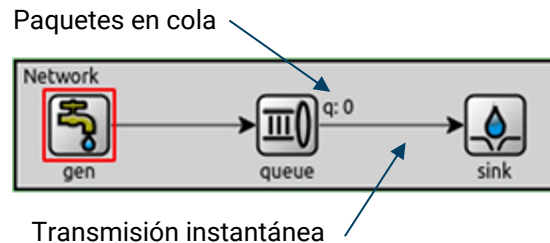
# Introducción - Simulación

---

- Ventajas
  - Estudiar **tiempos reales** prolongados en **tiempos de simulación** reducidos
  - Observar en detalle tiempos reales **pequeños** (ms, us)
  - Desarrollar **modelos de protocolos y algoritmos** definidos por eventos
  - Correr **cientos de casos** y hacer comparaciones paramétricas
  - **Corregir y actualizar** los modelos basado en los resultados
- Omnet++
  - Sistema expresado en módulos (archivo **.ned**)
  - Parámetros expresados en texto (archivo **.ini**)
  - Comportamiento (eventos) en una clase de **C++**

# Modelo de Colas

- Sistema
  - Generador (**gen**)
  - Cola (**queue**)
  - Sumidero (**sink**)
- Parámetros
  - Tiempo de generación (**generationInterval**)
  - Tiempo de servicio (**serviceTime**)
- Eventos
  - sendPacket (**gen**)
  - newPacket y endService (**queue**)
  - newPacket (**sink**)

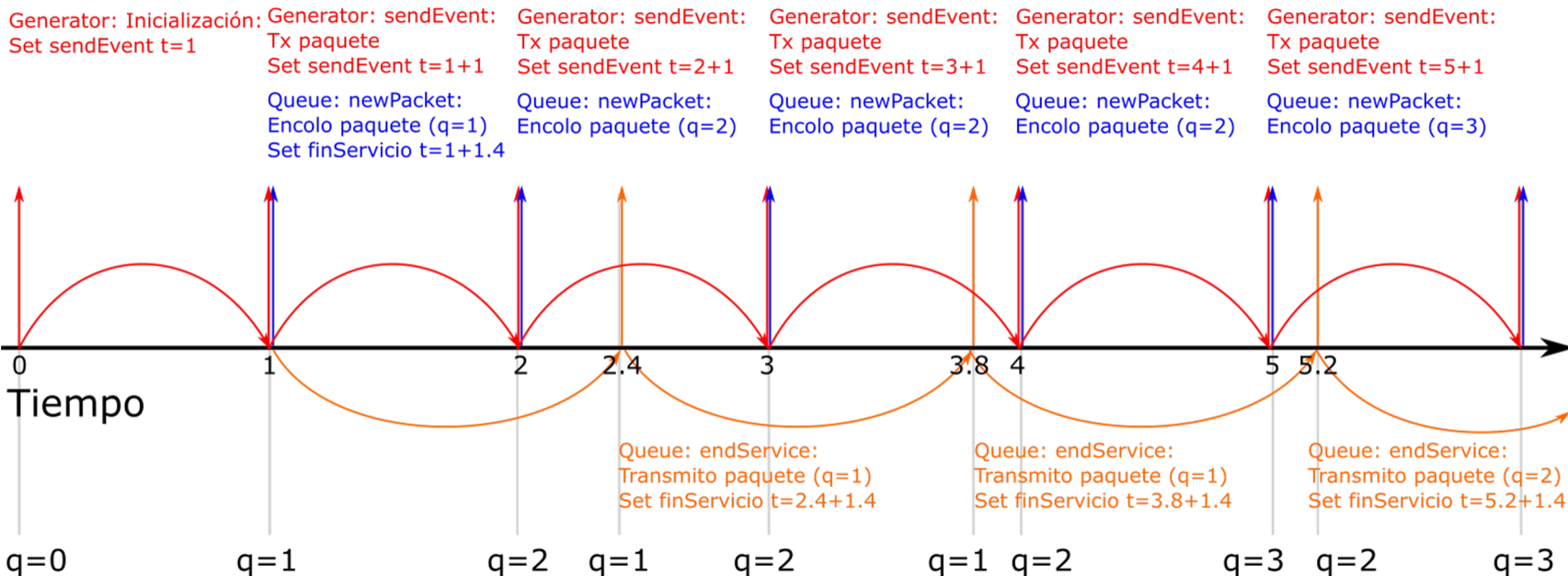


Archivo .ini

```
sim-time-limit = 200s
Network.gen.generationInterval=exponential(1)
Network.queue.serviceTime=exponential(1)
```

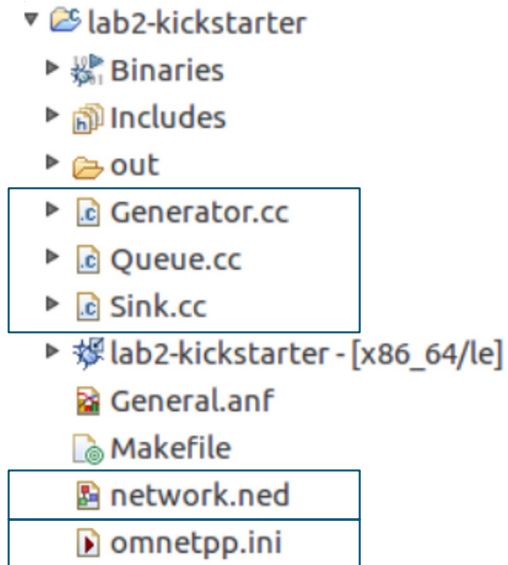
# Modelo de Colas - Ejecución Manual

- generationInterval = 1, y serviceTime = 1.4**

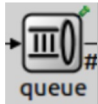


# Modelo de Colas - Ejecución Omnet++

- Descargar
  - Descargar **máquina virtual**
  - Descargar **kickstarter Lab 3**
- Ejecutar Omnet++
  - `$/omnetpp`
- Importar kickstart
  - File, Import, Existing Projects into Workspace
  - Root directory: lab3-kickstarter directory
- Ejecutar Simulación
  - 



# Modelo de Colas - Archivo .ned



```

simple Generator
{
 parameters:
 volatile double generationInterval;
 @display("i=block/source");
 gates:
 output out;
}

simple Queue
{
 parameters:
 volatile double serviceTime; // sec
 @display("i=block/queue;q=buffer");
 gates:
 input in;
 output out;
}

simple Sink
{
 parameters:
 @display("i=block/sink");
 gates:
 input in;
}

```

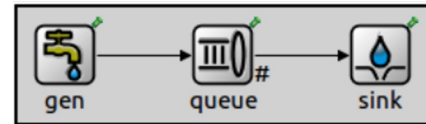
Cada módulo simple tiene una clase .cc de igual nombre que describe su comportamiento

```

network Network
{
 @display("bg1=2");
 submodules:
 gen: Generator {
 @display("p=30,30");
 }
 queue: Queue {
 @display("p=130,30");
 }
 sink: Sink {
 @display("p=230,30");
 }
 connections:
 gen.out --> queue.in;
 queue.out --> sink.in;
}

```

El módulo Network integra y conecta los módulos simples



# Modelo de Colas - Sink.cc



```
class Sink : public cSimpleModule {
private:
 cStdDev delayStats;
 cOutVector delayVector;
public:
 Sink();
 virtual ~Sink();
protected:
 virtual void initialize();
 virtual void finish();
 virtual void handleMessage(cMessage *msg);
};
```

Llamada al  
inicializar el  
módulo

```
void Sink::initialize(){
 // stats and vector names
 delayStats.setName("TotalDelay");
 delayVector.setName("Delay");
}
```

Llamada al haber  
un mensaje  
entrante

```
void Sink::handleMessage(cMessage * msg) {
 // compute queuing delay
 simtime_t delay = simTime() - msg->getCreationTime();
 // update stats
 delayStats.collect(delay);
 delayVector.record(delay);
 // delete msg
 delete(msg);
}
```

Borrar objeto msg,  
no se usará más

Clases para  
almacenar  
estadísticas

Llamada al  
finalizar el  
módulo

```
void Sink::finish(){
 // stats record at the end of simulation
 recordScalar("Avg delay", delayStats.getMean());
 recordScalar("Number of packets", delayStats.getCount());
}
```



# Modelo de Colas - Generator.cc



```
#include <string.h>
#include <omnetpp.h>

using namespace omnetpp;

class Generator : public cSimpleModule {
private:
 cMessage *sendMsgEvent;
 cStdDev transmissionStats;
public:
 Generator();
 virtual ~Generator();
protected:
 virtual void initialize();
 virtual void finish();
 virtual void handleMessage(cMessage *msg);
};
Define_Module(Generator);
```

scheduleAt(A1, A2)  
envía al mismo módulo  
el mensaje A2 en el  
tiempo A1

```
void Generator::initialize() {
 transmissionStats.setName("TotalTransmissions");
 // create the send packet
 sendMsgEvent = new cMessage("sendEvent");
 // schedule the first event at random time
 scheduleAt(par("generationInterval"), sendMsgEvent);
}

void Generator::handleMessage(cMessage *msg) {
 // create new packet
 cMessage *pkt = new cMessage("packet");
 // send to the output
 send(pkt, "out");

 // compute the new departure time
 simtime_t departureTime = simTime() + par("generationInter
 // schedule the new packet generation
 scheduleAt(departureTime, sendMsgEvent);
}
```

par("abc") trae el  
parámetro "abc"  
del archivo .ini

Nuevo paquete  
se envía por la  
gate "out"

# Modelo de Colas - Queue.cc



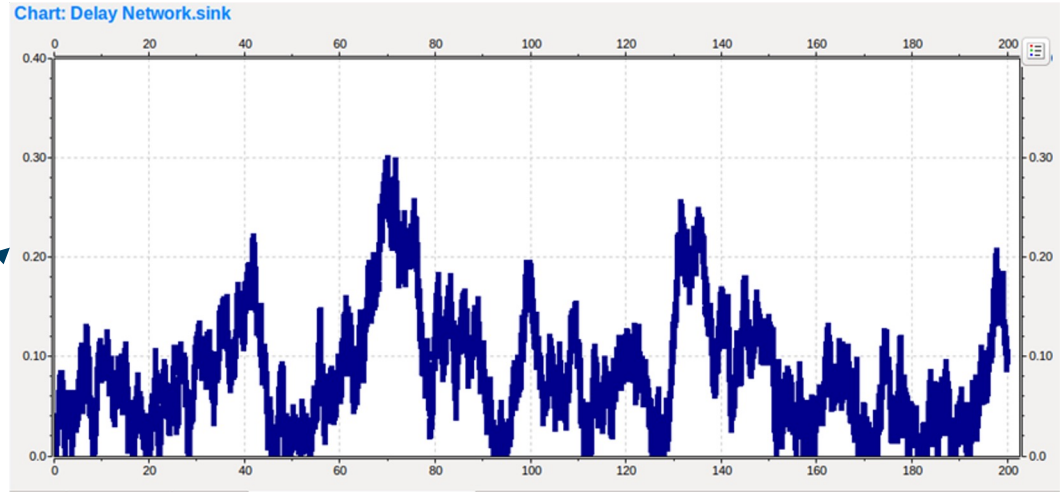
Queue tiene dos tipos de posibles eventos a) nuevo paquete y b) terminé de atender un paquete.  
Hay que diferenciarlos

```
class Queue: public cSimpleModule {
private:
 cQueue buffer;
 cMessage *endServiceEvent;
 simtime_t serviceTime;
public:
 Queue();
 virtual ~Queue();
protected:
 virtual void initialize();
 virtual void finish();
 virtual void handleMessage(cMessage *msg);
};
```

```
void Queue::handleMessage(cMessage *msg) {
 // if msg is signaling an endServiceEvent
 if (msg == endServiceEvent) {
 // if packet in buffer, send next one
 if (!buffer.isEmpty()) {
 // dequeue packet
 cMessage *pkt = (cMessage*) buffer.pop();
 // send packet
 send(pkt, "out");
 // start new service
 serviceTime = par("serviceTime");
 scheduleAt(simTime() + serviceTime, endServiceEvent);
 }
 } else { // if msg is a data packet
 // enqueue the packet
 buffer.insert(msg);
 // if the server is idle
 if (!endServiceEvent->isScheduled()) {
 // start the service
 scheduleAt(simTime(), endServiceEvent);
 }
 }
}
```

# Modelo de Colas - Métricas

- lab2-kickstarter
  - Binaries
  - Includes
  - out
  - Generator.cc
  - Queue.cc
  - Sink.cc
  - lab2-kickstarter - [x86\_64/le]
  - General.anf



- ▶ Avg delay (scalar) 0.084205399594469
- ▶ Delay (vector) 0.08420539959446845 (199239)
- ▶ Number of packets (scalar) 199239.0

# Notas Finales

---

- En los Lab 3, 4 y 5 vamos a evaluar
  - **Código claro, legible y bien comentado**
    - Que funcione y que nos permita entender su algoritmo, no hay test exhaustivos
  - **Informe con los análisis de las algoritmos creados**
    - Que nos permita entender que entendieron el problema
    - Dediquen tiempo a esto y estructuren el informe:
      - 1 - Título
      - 2 - Abstract
      - 3 - Introducción (planteo del problema)
      - 4 - Metodos (planteo de posibles soluciones)
      - 5 - Resultados (presentación y análisis de los resultados)
      - 6 - Discusión (limitaciones y posibles mejoras al/los método(s))
      - 7 - Referencias