Simulación (y 2)

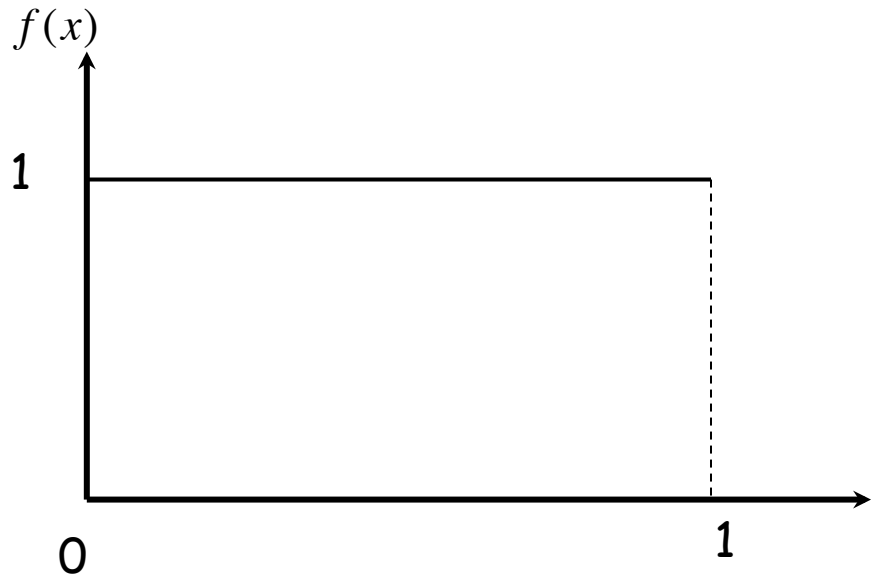
Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Grado en Ingeniería en Tecnologías de Telecomunicación, 4º

Generación de números aleatorios

Generación de números aleatorios

- Primero intentamos generar números aleatorios de una distribución uniforme
- Independientes
- Empleamos generadores de números pseudo-aleatorios (...)



Números pseudo-aleatorios

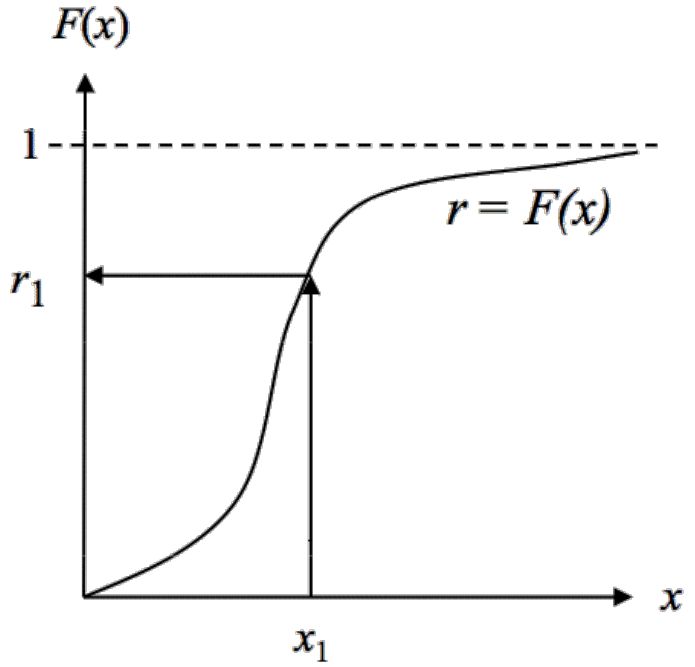
- Parece que fueran aleatorios
- Pero conocida la *semilla* se pueden predecir
- Incluso tienen un periodo
- Ejemplo: Linear Congruential Method

$$X_{i+1} = \left(aX_i + c\right) \bmod m$$

- ¿Y para una distribución no uniforme?

Inverse-transform Technique

- X variable aleatoria deseada
- F(x) = P[X<x]
- U variable aleatoria uniforme en [0,1]
- Se genera una muestra r_1 de U
- Se obtiene mediante la inversa de F(x) : $x_1 = F^{-1}(r_1)$
- x_1 es una muestra de la variable aleatoria X
- Esto es más sencillo si F(x) tiene una inversa “simple”



Ejemplo: Distribución exponencial

$$f(x) = \lambda e^{-\lambda x}$$

$$F(x) = 1 - e^{-\lambda x}$$

$$R = 1 - e^{-\lambda X}$$

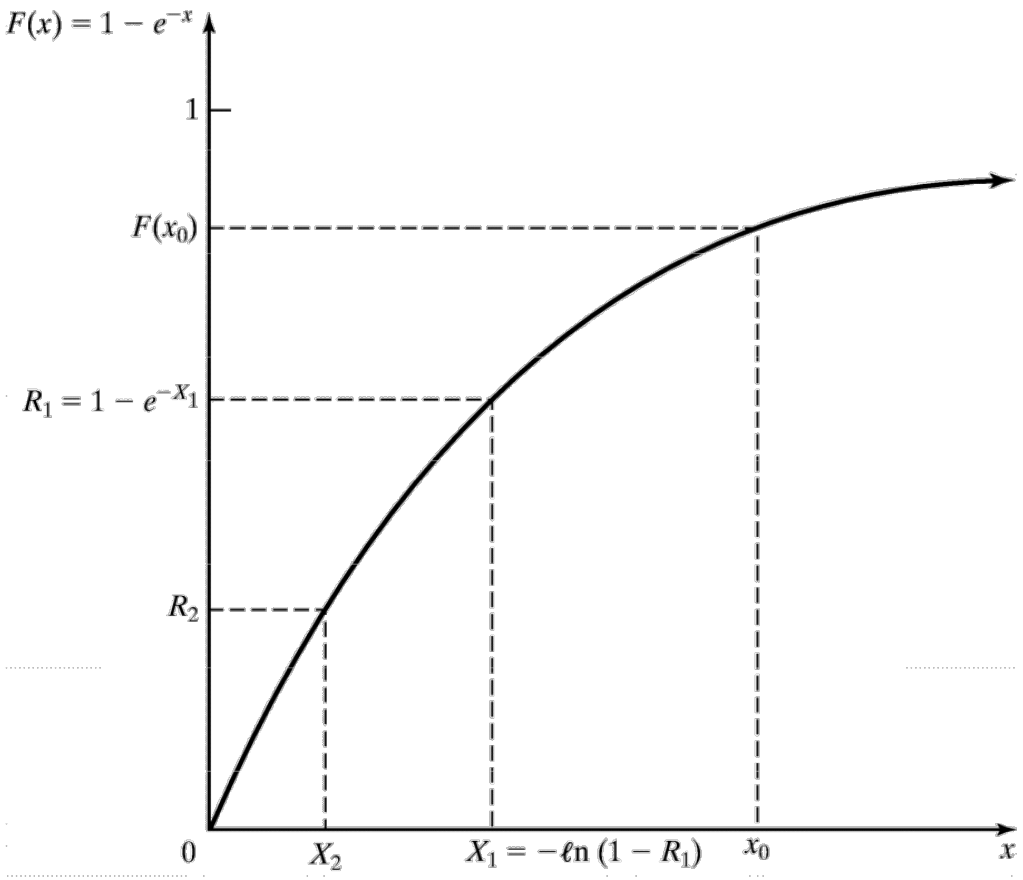
$$1 - R = e^{-\lambda X}$$

$$\ln(1-R) = -\lambda X$$

$$X = -\frac{\ln(1-R)}{\lambda} = F^{-1}(R)$$

ó

$$X = -\frac{\ln(R)}{\lambda} = F^{-1}(R)$$



(Tanto R como 1-R son variables aleatorias uniformes)

Inverse-transform Technique

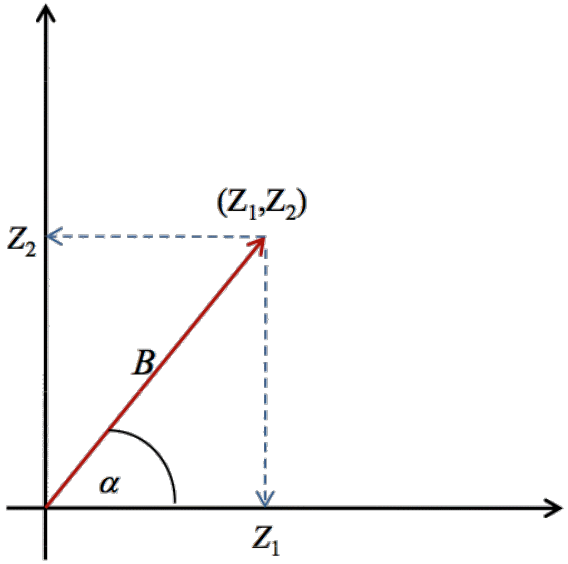
- Distribuciones “sencillas”: Triangular, Weibull, Pareto
- F(x) podría venir de una aproximación con datos experimentales
- Entonces se podría emplear interpolación para mejorar el resultado
- Para una variable aleatoria discreta es suficiente con una tabla
- “Difíciles”: Gamma, Normal, Beta
- Se tienen que emplear aproximaciones a F(x) o a $F^{-1}(x)$

Técnicas a partir de propiedades

Ejemplo: Gaussian distribution

- Z_1 y Z_2 variables aleatorias N(0,1)
- Supongamos que son coordenadas rectangulares de un punto (Z_1,Z_2)
- En coordenadas polares: $\begin{cases} Z_1 = B\cos(\alpha) \\ Z_2 = B\sin(\alpha) \end{cases}$
- El radio B es una variable aleatoria exponencial
- El ángulo α es una variable aleatoria uniforme
- Son independientes
- Así que se pueden obtener dos muestras de N(0,1) con dos muestras de una variable aleatoria uniforme

$$Z_1 = \sqrt{-2\ln(R_1)}\cos(2\pi R_2)$$
$$Z_2 = \sqrt{-2\ln(R_1)}\sin(2\pi R_2)$$



- Y para Y = N(μ,σ) :

$$Y = \mu + \sigma Z_i$$

Simulación de eventos discretos

Simulación de eventos discretos

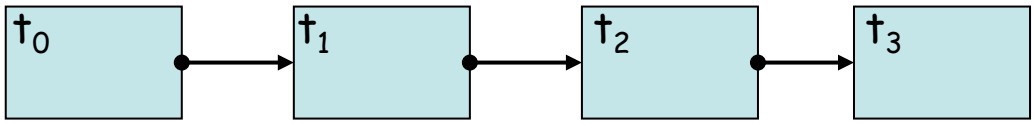
- Modelado con el tiempo de un sistema en el que todos los cambios de estado se producen en un conjunto discreto de puntos en el tiempo
- Empleo de métodos numéricos
  - En vez de métodos analíticos
  - El modelo se “corre” en vez de se “resuelve”
- Se lleva a cabo produciendo una secuencia de *snapshots* del sistema con el tiempo
- El *snapshot* en un instante *t* incluye
  - El estado del sistema en el instante *t*
  - Una lista de las actividades en progreso y cuándo terminarán
  - El estado de todas las entidades
  - Los valores de todos los contadores estadísticos

Future (pending) Events List (FEL)

- Es el mecanismo para hacer avanzar la simulación
- La FEL contiene los eventos planificados para este instante o posteriores aún sin procesar
- Cada evento contiene el instante de tiempo en que sucede
- Ordenados por instante de tiempo de menor a mayor
- Garantiza que los eventos tienen lugar en orden cronológico

Gestión de actividades con una FEL

- La duración se conoce al comenzar (determinista o aleatoria)
- En algunos entornos existe la posibilidad de cancelar
- Al comenzar la actividad se introduce un evento de finalización de la actividad en la FEL
- Ejemplo: Nueva llegada, actividad el tiempo hasta la siguiente (...)



CLOCK = $t < t_0$

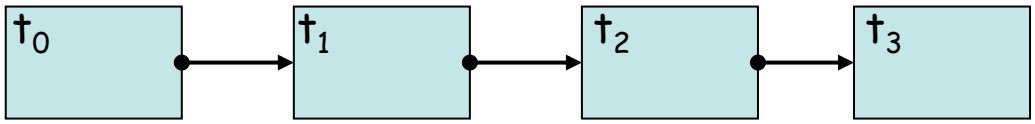
$$t_0 \leq t_1 \leq t_2 \leq t_3$$

Información fundamental a definir

- ¿Efectos de cada tipo de evento?
  - Cambios de estado
  - Cambios de atributos de entidades
- ¿Cómo se definen las actividades?
  - Deterministas, probabilísticas, ecuaciones
  - Qué tipo de evento marca su pricipio/final
  - Su comienzo es condicional al estado
- ¿Cómo comienza la simulación?
  - Primeros eventos
- ¿Cuándo finaliza la simulación?

Avance de la simulación

- *Snapshots* del sistema con el tiempo
- *Snapshot* incluye el estado del sistema y la FEL
- Esa FEL contiene las actividades en progreso y cuándo finalizan
- CLOCK = t = instante actual en la simulación
- Evento en t_0 = Evento inminente
- Se actualiza CLOCK = t_0
- Se retira el evento inminente de la FEL
- Se "ejecuta" el evento
- Eso crea un nuevo *snapshot* del sistema



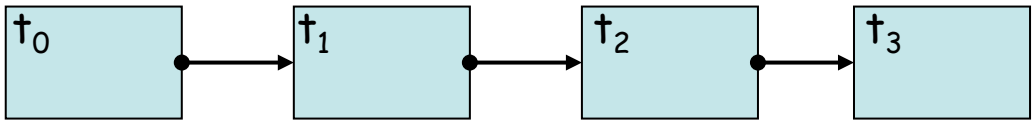
$CLOCK = t < t_0$

$t_0 \leq t_1 \leq t_2 \leq t_3$

Event-scheduling/Time-advance

Mientras queden eventos en la FEL

- Retirar el primero (evento inminente)
- Avanzar la variable de CLOCK hasta el instante del evento
- Procesar el evento: puede modificar el estado del sistema e introducir otros eventos futuros en la FEL manteniéndola ordenada
- Actualizar los contadores y estadísticos



¿ Fin de la simulación ?

- Cuando no queden eventos en la FEL
- En la inicialización introducir un evento futuro de finalización
  - Limita el tiempo simulado
  - No limita el tiempo real
- Detenerla al alcanzar una duración (tiempo real)
- Detenerla al alcanzar unas medidas una cierta precisión.

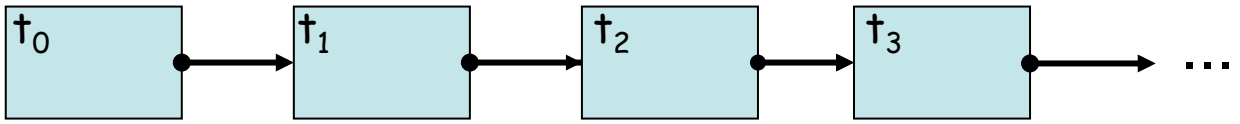
Gestión de la FEL

- Su longitud cambia durante toda la simulación
- Su gestión eficiente es vital
- Operaciones más frecuentes:
  - Retirar el primero
  - Insertar manteniendo el orden
- Puede soportar el eliminar un evento en concreto

Ejemplo de simulación con FEL

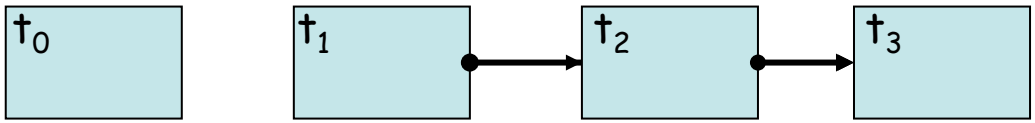
Ejemplo

- Iniciamos la FEL con todos los eventos de llegadas que se vayan a producir
- Esto es muy ineficiente
- Es común que trabajemos con millones, decenas de millones o centenares de millones de llegadas
- Un gran gasto de memoria
- ¿Mejor aproximación al problema? (...)



Ejemplo: Avance con llegadas

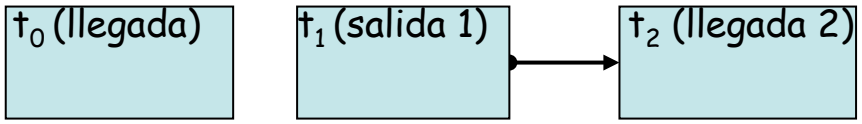
- Evento inicial: una llegada
- Se procesa
  - Introduce eventos consecuencia de ella (…)
  - Se introduce un nuevo evento que es la siguiente llegada donde le corresponda (…)



Ejemplo: Avance con llegadas

Ejemplo: uso de un cajero automático

- Evento inicial: llegada del primer cliente (t_0)
- Se procesa:
  - Introduce eventos consecuencia de ella: evento de cuándo termina de usar el cajero (t_1) (…)
  - Se introduce un nuevo evento que es la siguiente llegada (t_2) (…)

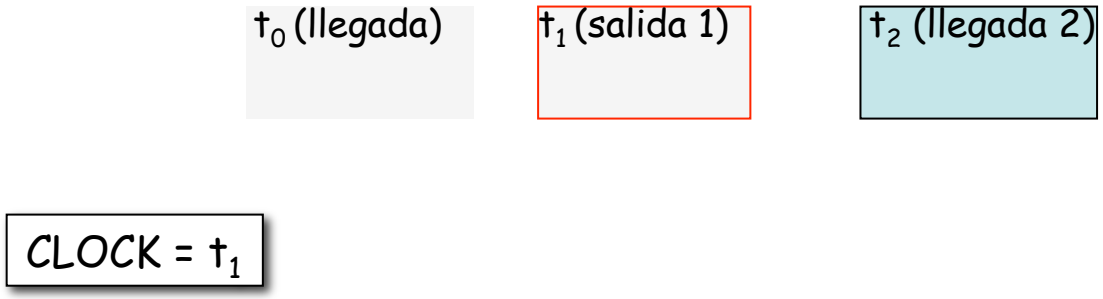


CLOCK = t_0

Ejemplo: Avance con llegadas

Ejemplo: uso de un cajero automático

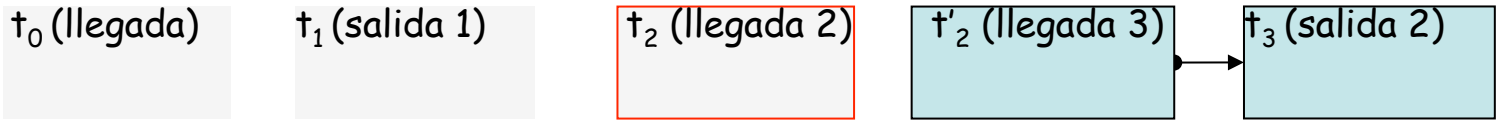
- Evento inminente: salida del primer cliente (t_1)
- Se procesa:
  - Actualización de estadísticas



Ejemplo: Avance con llegadas

Ejemplo: uso de un cajero automático

- Evento inminente: llegada del segundo cliente (t_2)
- Se procesa:
  - Introduce eventos consecuencia de ella: evento de cuándo termina de usar el cajero (t_3) (…)
  - Se introduce un nuevo evento que es la siguiente llegada (t'_2) (…)

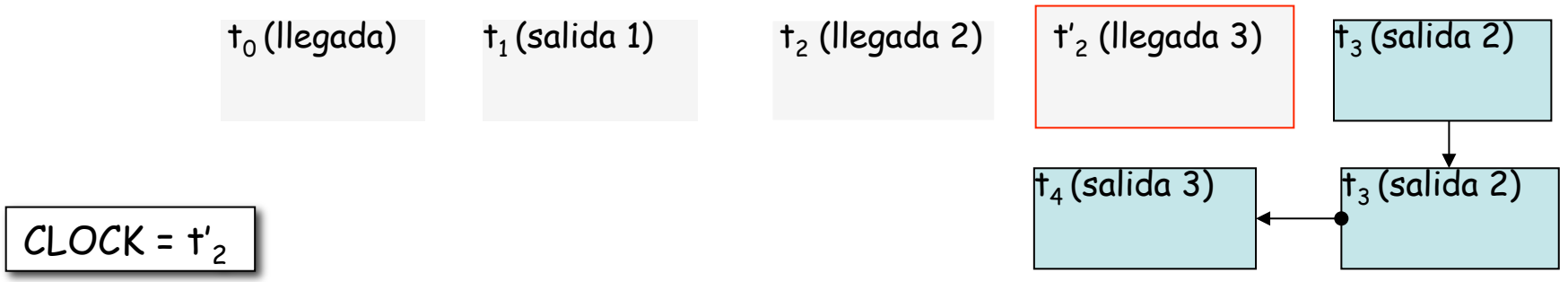


CLOCK = t_2

Ejemplo: Avance con llegadas

Ejemplo: uso de un cajero automático

- Evento inminente: llegada del tercer cliente (t'_2)
- Se procesa:
  - Introduce eventos consecuencia de ella: evento de cuándo termina de usar el cajero (t_4) (…)
  - Se introduce un nuevo evento que es la siguiente llegada (t''_2) (…)



Simulation Tools

- Librerías de utilidades
- Simuladores programables
- Simuladores controlables (gráfico, script)
- Simuladores de redes (ns2, OMNeT++, SSFNet, Parsec, Qualnet, OPNET, JiST/SWANS …)

Práctica

Dos entregables

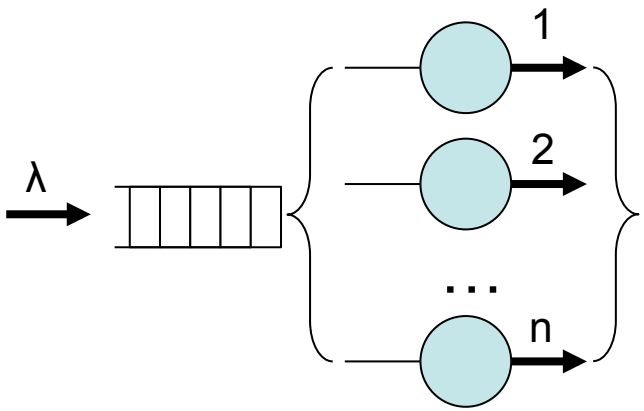
1. Un programa que saque por pantalla números aleatorios independientes según una distribución exponencial
   - Parámetros en la línea al ejecutarlo
   - Dos parámetros, ambos obligatorios:
     1. λ (tasa media de llegadas por unidad de tiempo)
     2. Número de valores a sacar, que pueden ser millones
   - La salida son líneas de texto (en unidades de tiempo)
   - Un número en cada línea (el resultado de la realización de la v.a.)
   - Ejemplo:

```
% valoresalea 12.5 10
0.0214674218
0.1150884632
0.0579031792
0.0283326041
0.1489137588
0.0332117244
0.1868370616
0.0529504899
0.0087550381
0.0944970608
```

   - Esto es el 10% de la práctica
   - Esta es la parte sencilla…

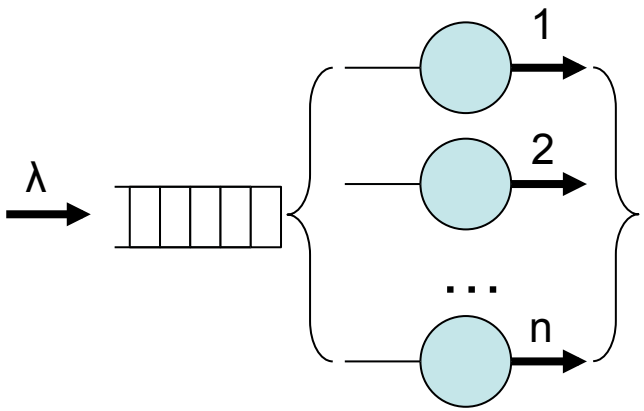
Dos entregables

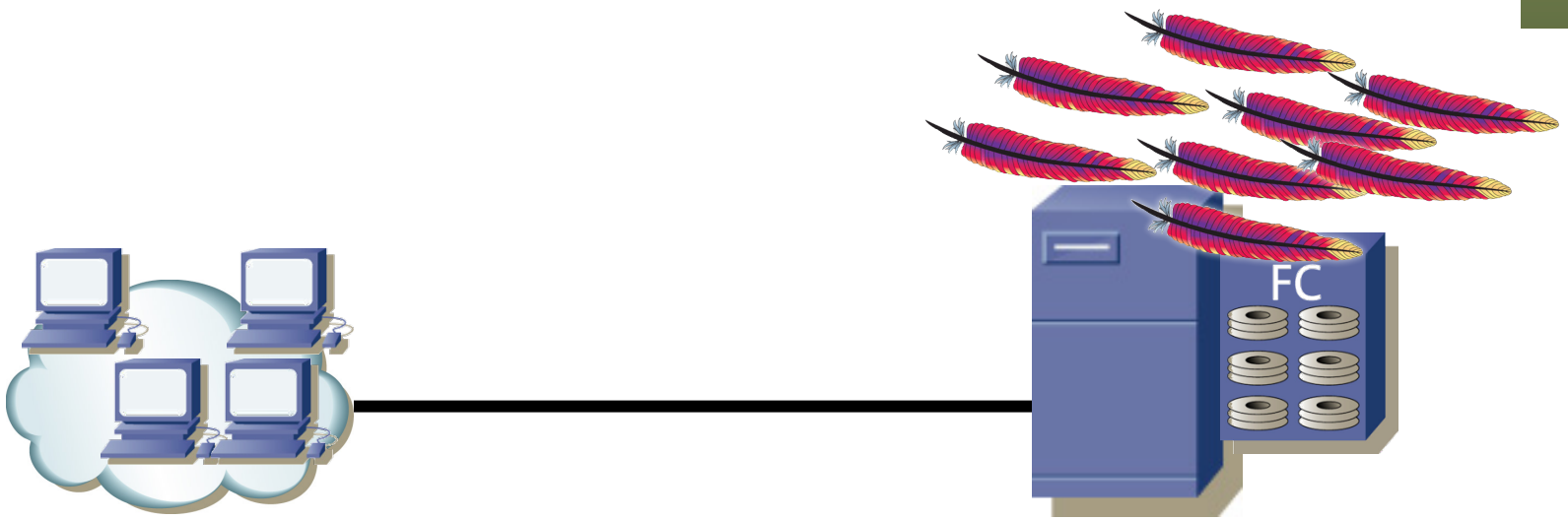
2. Un simulador de eventos discretos de un sistema con cola finita y n servidores



Simulador

- Esto podrían ser peticiones de ficheros a un servidor web que lanza múltiples hilos





Parámetros

- En los argumentos al ejecutarlo
- Todos obligatorios
- En el orden especificado
- Parámetros:
  1. Nombre (path) al fichero con los tiempos entre llegadas, en texto, uno por línea (o sea, el formato de salida del primer entregable)
  2. Nombre (path) al fichero con los tiempos de servicio (idem)
  3. Número de servidores (1+)
  4. Número máximo de clientes que pueden estar en la cola (0+)

  Ejemplo

  ```
  % misimula fichTinter.txt fichTserv.txt 6 45
  ```

¿Qué debe calcular?

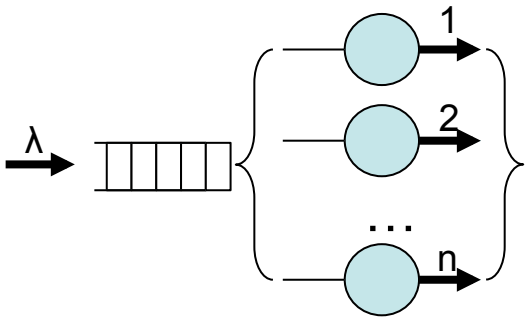
- Para cada llegada se vuelca a fichero una línea de texto con 3 columnas:
  - El instante de tiempo de la llegada
  - El número de clientes que había en cola en ese instante
  - 1 si se cursa, 0 si se descarta por cola llena
  - Es decir: Tiempo_llegada nº_en_cola 1o0
  - Ejemplo:

    23.5546 12 1

¿Qué hacer con el resultado?

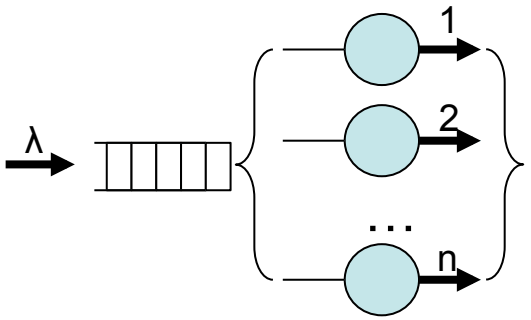
- En primer lugar verificar que lo hace bien
  - Por ejemplo comprobar que los resultados en casos sencillos cuadran con la teoría
  - Por ejemplo con 1 solo servidor y cola muy grande podemos verificar el número medio de clientes si las llegadas son de Poisson y los tiempos de servicio exponenciales
  - Con n servidores y sin cola las pérdidas deberían ajustarse a…

  - Que funcione bien es lo que más se valorará (70% de la práctica)
- En segundo lugar
  - Comparar el comportamiento del sistema entre llegadas de Poisson y tamaños exponenciales y el tráfico de la práctica anterior hacia el NAT (o desde él)
  - Esto, junto con la descripción del simulador se valorará un 20%



Eventos (propuesta)

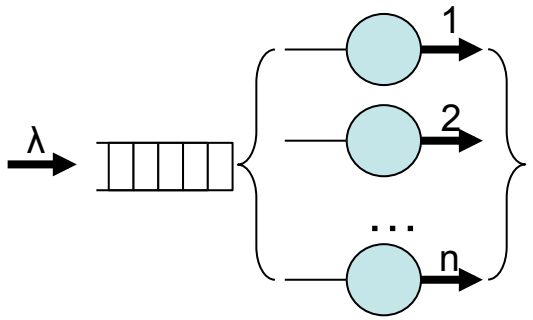
- Llegada al sistema
- Salida de un servidor



Procesado de evento de llegada

1. Planificar un nuevo evento de llegada en el futuro:
   1. Calcular tiempo entre llegadas
   2. Planificar evento de llegada en instante actual + ese tiempo
2. ¿Hay algún servidor libre? (Contador de ocupados)
   - Sí :
     1. Calcular su tiempo de servicio
     2. Planificar un evento de salida en instante actual + tiempo de servicio
     3. Incrementar número de servidores ocupados
   - No : ¿Queda espacio libre en la cola? (contador en cola)
     - Sí : Incrementar contador de clientes en cola
     - No : Apuntar una llegada perdida
3. Volcar línea de información

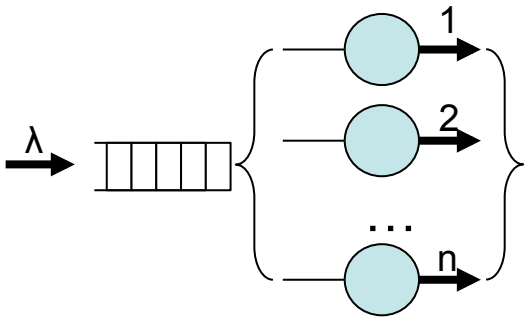
Contadores:
- Servidores ocupados
- Clientes en cola



Procesado de evento de salida

1. Decrementar contador de número de servidores ocupados
2. ¿Hay clientes en la cola?
   - Sí :
     1. Eliminar un cliente de la cola (contador clientes en cola)
     2. Incrementar número de servidores ocupados
     3. Calcular su tiempo de servicio
     4. Planificar un evento de salida en instante actual + tiempo de servicio
   - No : nada

Contadores:
- Servidores ocupados
- Clientes en cola



Práctica: Entregables

- Código fuente e instrucciones simples de compilación de ambos programas
- Deben compilar y funcionar en máquinas Linux del laboratorio
- Breve explicación del diseño del segundo programa (1-2 páginas)
- Explicación de las comprobaciones llevadas a cabo para validarlo (2-3 páginas)
- Documento comparando resultados entre llegadas de Poisson (+duraciones exponenciales) y tráfico real (3-5 páginas)

Opcional

- Que calcule para cada salida el tiempo que pasó en el sistema
- Emplear alguna librería (por ejemplo SIM.JS)
- (Se puede flexibilizar el formato de parámetros bajo petición razonada)