Proyecto Final Simulación: Segunda Entrega

Buttignol Leandro y López Martín

13 de marzo de 2017

1 Consigna

Consideremos un ascensor que se puede comandar por los siguientes eventos: arriba,abajo,parar. Cada piso a su vez cuenta con un sensor que indica la presencia del ascensor, de modo tal que la salida del sistema "ascensor" son los eventos producidos por dichos sensores.

Algunas consideraciones:
• El ascensor sube y baja a velocidades contantes de 1 metro por segundo.
• La distancia entre un piso y otro es de 2 metros.
• El edificio tiene 10 pisos.

1.1 Extensión 1

Diseñar un modelo de un controlador para el ascensor al cual llegan eventos con números enteros, indicando que el ascensor debe ir hasta dicho piso. Dicho controlador recibe además como eventos de entrada, las salidas del ascensor. El sistema debe provocar una señal de salida "libre" cada vez que finalice una tarea y "ocupado" cada vez que comience otra. Suponer que no arriban eventos de entrada indicando que se debe ir a un nuevo piso mientras se está yendo a otro. Los pisos a los cual debe dirigirse el ascensor son igualmente probables.

1.2 Extensión 2

Obtener el modelo del tablero del ascensor, suponiendo que consta de una cola en la que almacena pedidos (observar que no debe haber dos pedidos consecutivos iguales, ya que no tiene sentido ir dos veces al mismo lugar). Los arribos de los pedidos al tablero siguen una distribución exponencial con una media de 1 cada 9 segundos.

1.3 Extensión 3

Suponiendo que se administran 2 ascensores, diseñar un modelo para el control de pedidos desde los pisos. Suponer que al mismo llegan pedidos del tipo "ir al piso j" que indican que desde el piso j se pidió el ascensor. Dichos pedidos pueden ser enviados a uno u otro tablero de ascensor según alguna estrategia adoptada.

1.4 Primer entrega

• Diseñe un modelo preliminar del problema.
• Verifique si las colas detectadas son estables.
• Estimar analíticamente el tiempo medio que una solicitud pasa en la cola (tiempo de respuesta).
• Especifique su comportamiento mediante el formalismo DEVS.

1.5 Segunda Entrega

• Implementar una simulación utilizando PowerDEVS.
• Analizar las salidas utilizando el método del Intervalo de Confianza con Precisión Específica. Considerar como error crítico un 5 % del resultado analítico.
• Graficar la trayectoria del ascensor.

Haga una comparativa del tiempo de respuesta según las siguientes estrategias de envío de pedidos del controlador:

1. Los pedidos se alternan si o si entre los 2 ascensores.
2. Se envía siempre a uno disponible, priorizando el elevador 1.
3. Se envía al elevador que según ciertos cálculos llegará más rápido al piso destino.

2 Devs atómicos.

En esta sección se denota el formalismo DEVS de cada componente desarrollado.

2.1 Dev de ascensor.

Descripción de funcionamiento: A continuación se especifica el comportamiento de un ascensor "tonto" que recibe como entrada acciones a realizar ("Parar", "Arriba", "Abajo") y como salida retorna los pisos por los que va pasando. En el momento en que el ascensor recibe una acción "Parar" se detiene y queda en espera de una nueva acción a realizar. En el momento en que el ascensor recibe una acción "Arriba" comienza si ascenso constante hasta recibir una nueva acción a ejecutar, en caso que la acción recibida sea un "Abajo" el ascensor comienza a descender hasta recibir una nueva orden.

AscensorDev $\langle X, Y, S, \delta int, \delta ext, \lambda, ta \rangle$

$X = \{"Arriba", "Abajo", "Parar"\}$
$Y = \{0..9\}$
$S = \{\{0..9\} \times R^+ \times \{ "Arriba",\ "Abajo",\ "Parar"\}\}$

El estado S está conformado por el piso actual$\times tiempo \times acción$.

$\delta ext(s, e, x) = \delta ext((piso, tiempo, accion), e, x)$

$$\begin{cases} (piso, \infty, x) & \text{si accion="Parar"} \wedge \text{x="Parar"} \\ (piso+1, \frac{alturaPiso}{velocidad}, x) & \text{si accion="Parar"} \wedge \text{x="Arriba"} \\ (piso-1, \frac{alturaPiso}{velocidad}, x) & \text{si accion="Parar"} \wedge \text{x="Abajo"} \\ (piso, tiempo-e, accion) & \text{si (accion="Abajo"} \vee \text{accion="Arriba")} \wedge \text{(x="Abajo"} \vee \text{x="Arriba")} \\ (piso, \infty, x) & \text{si ((accion="Abajo"} \vee accion = "Arriba") \wedge x = "Parar") \end{cases}$$

$\delta int(s) = \delta int(piso, tiempo, accion)$

$$\begin{cases} (piso, \infty, "Parar") & \text{si accion="Parar"} \\ (piso+1, \frac{alturaPiso}{velocidad}, accion) & \text{si accion="Arriba"} \\ (piso-1, \frac{alturaPiso}{velocidad}, accion) & \text{si accion="Abajo"} \end{cases}$$

$\lambda(s) = \lambda(piso, tiempo, accion) = piso$

$ta(s) = ta(piso, tiempo, accion) = tiempo$

$SInicial = (0, \infty, "Parar")$

2.2 Dev de controlador de 1 ascensor.

Descripción de funcionamiento: En este caso el controlador debe controlar el comportamiento de dos ascensores, por lo tanto lleva el estado de cada uno, además el controlador es el encargado de dar las órdenes al ascensor, para ello debe conocer el destino del ascensor y el piso por donde circula este para poder decirle cuando subir, bajar o detenerse.

Para ello se necesita que el controlador tenga un puerto de entradas para recibir nuevos destinos y otro que reciba las salidas del ascensor para saber por dónde va el mismo y poder realizar correctamente la tarea de control. En cuanto a puertos de salida también se registran dos, uno destinado a enviar las órdenes al ascensor, y el segundo destinado a indicar la disponibilidad del ascensor a la botonera (si el ascensor está ocupado la botonera no le envía nuevas peticiones, en cambio mientras el ascensor esté libre la botonera si las enviará).

Controlador1Dev$\langle X, Y, S, \delta int, \delta ext, \lambda, ta\rangle$

$X = \{0..9\} \times \{0, 1\}$
$Y = \{\text{"}Arriba\text{"}, \text{"}Abajo\text{"}, \text{"}Parar\text{"}\} \times \{0\} \cup \{\text{"}Libre\text{"}, \text{"}Ocupado\text{"}\} \times \{1\}$
$S = \{\{0..9\} \times \{0..9\} \times R^+ \times \{\text{"}Libre\text{"}, \text{"}Ocupado\text{"}\} \times \{0..9\}\}$

El estado S está conformado por piso actual × destino × tiempo × estado del ascensor × puerto por el que será la prox. salida.

δext(s,e,x)=δext((piso,destino,tiempo,estado,salida),e,(n,p))

$$\begin{cases} (piso, n, 0, \text{"}Ocupado\text{"}, salida) & \text{si estado="Libre"} \wedge p = 0 \\ (n, destino, 0, estado, 0) & \text{si estado="Ocupado"} \wedge p = 1 \wedge n = destino \\ (n, destino, \infty, estado, 0) & \text{estado="Ocupado"} \wedge p = 1 \wedge n \neq destino \end{cases}$$

$\delta int(s) = \delta int(piso, destino, tiempo, estado, salida)$

$$\begin{cases} (piso, destino, 0, \text{"}Libre\text{"}, 1) & \text{si salida=0} \wedge estado = \text{"}Ocupado\text{"} \wedge piso = destino \\ (piso, destino, 0, estado, 1) & \text{si salida=0} \wedge estado = \text{"}Ocupado\text{"} \wedge piso \neq destino \\ (piso, destino, \infty, estado, 0) & \text{salida=1} \end{cases}$$

$\lambda(s) = \lambda(piso, destino, tiempo, estado, salida)$

$$\begin{cases} (\text{"}Parar\text{"}, salida) & \text{si salida=0} \wedge piso = destino \\ (\text{"}Abajo\text{"}, salida) & \text{si salida=0} \wedge piso > destino \\ (\text{"}Arriba\text{"}, salida) & \text{si salida=0} \wedge piso < destino \\ (estado, paraello) & \text{si salida=1} \end{cases} \quad yelpisopordondecirculaestepara.$$

$ta(s) = ta(piso, destino, tiempo, estado, salida) = tiempo$

$SInicial = (0, 0, \infty, \text{"}Libre\text{"}, 0)$

2.3 Dev de controlador de 2 ascensores

Descripción de funcionamiento: Éste DEVS es similar al anterior solo que ahora debe controlar dos ascensores. Para ello se tiene en cuenta el piso de destino y el piso actual de un nuevo ascensor, este control se realiza a través de un nuevo puerto de entrada (Lo que forma un total de 3 puertos de entrada). También se requiere un nuevo puerto de salida para enviar las órdenes al nuevo ascensor.
A su vez se adhieren dos colas de peticiones (una para cada ascensor) que almacenan las peticiones a tratar por cada ascensor respectivamente. Estas colas se llenan de acuerdo a lo que dicta una función llamada óptimo la cuál implementa una heurística de decisión a la hora de tratar una petición.

Controlador2Dev$\langle X, Y, S, \delta int, \delta ext, \lambda, ta\rangle$

$X = \{0..9\} \times \{0, 1, 2\}$
$Y = \{"Arriba", "Abajo", "Parar"\} \times \{0, 1\} \cup \{"Libre", "Ocupado"\} \times \{2\}$
$S = (a1, a2, \{0, 1, 2\})$
$S = ((\{0..9\}\times\{0..9\}\times\{"Libre", "Ocupado"\}\times R^+)\times(\{0..9\}\times\{0..9\}\times\{"Libre", "Ocupado"\}\times R^+))\times\{0, 1, 2\})$

El estado S está compuesto por a1 × a2 × puerto de la prox.salida. Donde a1 y a2 son los estados de ascensores distintos. Dichos estados están compuesto por piso actual × piso destino × estado × tiempo.

$\delta ext((a1, a2, sal), e, (n, p)) = \delta ext((p1, d1, e1, t1), (p2, d2, e2, t2), sal), e, (n, p))$

$$\begin{cases} ((p1, n, "Ocupado", 0), (p2, d2, e2, t2 - e), 0) & \text{estado(a1,a2)="Libre"} \wedge p = 0 \wedge optimo(a1, a2, n) = "a1" \\ ((p1, d1, e1, t1 - e), (p2, n, "Ocupado", 0), 1) & \text{estado(a1,a2)="Libre"} \wedge p = 0 \wedge optimo(a1, a2, n) = "a2" \\ ((n, d1, e1, 0), (p2, d2, e2, t2 - e), 0) & \text{estado(a1,a2)="Ocupado"} \wedge p = 1 \wedge n = d1 \\ ((n, d1, e1, \infty), (p2, d2, e2, t2 - e), 1) & \text{estado(a1,a2)="Ocupado"} \wedge p = 1 \wedge n \neq d1 \\ ((p1, d1, e1, t1 - e), (n, d2, e2, 0), 1) & \text{estado(a1,a2)="Ocupado"} \wedge p = 2 \wedge n = d2 \\ ((p1, d1, e1, t1 - e), (n, d2, e2, \infty), 0) & \text{estado(a1,a2)="Ocupado"} \wedge p = 2 \wedge n \neq d2 \end{cases}$$

$\delta int(a1, a2, sal) = \delta int((p1, d1, e1, t1), (p2, d2, e2, t2), sal)$

$$\begin{cases} ((p1, d1, "Libre", 0), a2, 2) & \text{si sal=0} \wedge e1 = "Ocupado" \wedge t1 \leq t2 \wedge p1 = d1 \\ ((p1, d1, e1, 0), a2, 2) & \text{sal=0} \wedge e1 = "Ocupado" \wedge t1 \leq t2 \wedge p1 \neq d1 \\ (a1, (p2, d2, "Libre", 0), 2) & \text{sal=1} \wedge e2 = "Ocupado" \wedge t1 > t2 \wedge p2 = d2 \\ (a1, (p2, d2, e2, 0), 2) & \text{sal=1} \wedge e2 = "Ocupado" \wedge t1 > t2 \wedge p2 \neq d2 \\ ((p1, d1, e1, \infty), a2, 1) & \text{sal=2} \wedge t1 = 0 \\ (a1, (p2, d2, e2, \infty), 0) & \text{sal=2} \wedge t2 = 0 \end{cases}$$

$\lambda(a1, a2, sal) = \lambda((p1, d1, e1, t1), (p2, d2, e2, t2), sal)$

$$\begin{cases} ("Parar", sal) & \text{si sal=0} \wedge p1 = d1 \\ ("Abajo", sal) & \text{si sal=0} \wedge p1 > d1 \\ ("Arriba", sal) & \text{si sal=0} \wedge p1 < d1 \\ ("Parar", sal) & \text{si sal=1} \wedge p2 = d2 \\ ("Abajo", sal) & \text{si sal=1} \wedge p2 > d2 \\ ("Arriba", sal) & \text{si sal=1} \wedge p2 < d2 \\ (estado(a1, a2), sal) & \text{si sal=2} \end{cases}$$

$ta(a1, a2, sal) = ta((p1, d1, e1, t1), (p2, d2, e2, t2), sal) = Min(t1, t2)$

$SInicial((0, 0, "Libre", \infty), (0, 0, "Libre", \infty), 1)$

Donde,

-estado(a1,a2) es una función que recibe el estado de 2 ascensores a1 y a2. Devuelve "Libre" si entre a1 y a2 hay algún ascensor disponible. En caso contrario, devuelve "Ocupado".

-optimo(x,y,n) es una función que recibe el estado de 2 ascensores a1 y a2, y un numero entre 0 y 9 que representa el piso destino. Esta función implementará una estrategía para decidir cual ascensor debe ser en el encargado de ir hasta el piso destino. Devuelve "a1" si el ascensor óptimo es a1 o "a2" en el caso contrario.

2.4 Dev de botonera

Descripción de funcionamiento: este componente DEVS se encarga de tomar las salidas del generador, almacenarlas en una cola y entregarlas al controlador cuando este esté disponible, es decir, cuando haya ascensores libres.

BotoneraDev$\langle X, Y, S, \delta int, \delta ext, \lambda, ta \rangle$

$X = \{0..9\} \times \{0\} \cup \{"Libre", "Ocupado"\} \times \{1\}$
$Y = \{0..9\}$
$S = \{"Libre", "Ocupado"\} \times$ Cola de N $\times \mathrm{R}^+$

El estado S está compuesto por el estado de los ascensores $\times$ peticiones a tratar $\times$ tiempo.

$\delta ext(s, e, x) = \delta ext((estado, Q, t), e, (n, p))$
$$\begin{cases} (estado, Q.add(n), 0) & \text{si p=0} \wedge estado = "Libre" \\ (estado, Q.add(n), \infty) & \text{si p=0} \wedge estado = "Ocupado" \\ (n, Q, 0) & \text{si p1=} \wedge n = "Libre" \\ (n, Q, \infty) & \text{si p=1} \wedge n = "Ocupado" \end{cases}$$

$\delta int(s) = \delta int(estado, Q, t)$
$\{ \ (estado, Q.pop(), \infty)$ si estado="Libre" $\wedge \neg Q.isEmpty()$

$\lambda(s) = \lambda(estado, Q, tiempo) =$
$\{ \ Q.top()$ si estado="Libre" $\wedge \neg Q.isEmpty()$

$ta(s) = ta(estado, Q, t) = t$

$SInicial("Libre", \emptyset, \infty)$

2.5 Dev de generador

Descripción de funcionamiento: este componente describe un generador de números aleatorios enteros entre 0 y 9, en un intervalo de tiempo que sigue una distribución exponencial con una media de 1 cada 9 segundos.

GeneradorDev$\langle X, Y, S, \delta int, \delta ext, \lambda, ta \rangle$

$X = \emptyset$
$Y = \{0..9\}$
$S = R^+ \times R^+$

$\delta ext(s, e, x) = \{\}$

$\delta int(s) = \delta int(sem, tiempo) = (rn(sem, 0, 1), (\frac{-1}{\frac{1}{9}} ln(1 - sem))$

$\lambda(s) = \lambda(sem, tiempo)$

$$\begin{cases} 0 & \text{si sem} < \frac{1}{10} \\ 1 & \text{si } \frac{1}{10} \leq \text{sem} < \frac{2}{10} \\ 2 & \text{si } \frac{2}{10} \leq \text{sem} < \frac{3}{10} \\ 3 & \text{si } \frac{3}{10} \leq \text{sem} < \frac{4}{10} \\ 4 & \text{si } \frac{4}{10} \leq \text{sem} < \frac{5}{10} \\ 5 & \text{si } \frac{5}{10} \leq \text{sem} < \frac{6}{10} \\ 6 & \text{si } \frac{6}{10} \leq \text{sem} < \frac{7}{10} \\ 7 & \text{si } \frac{7}{10} \leq \text{sem} < \frac{8}{10} \\ 8 & \text{si } \frac{8}{10} \leq \text{sem} < \frac{9}{10} \\ 9 & \text{si } \frac{9}{10} \leq \text{sem} < \frac{10}{10} \end{cases}$$

$ta(s) = ta(sem, tiempo) = tiempo$

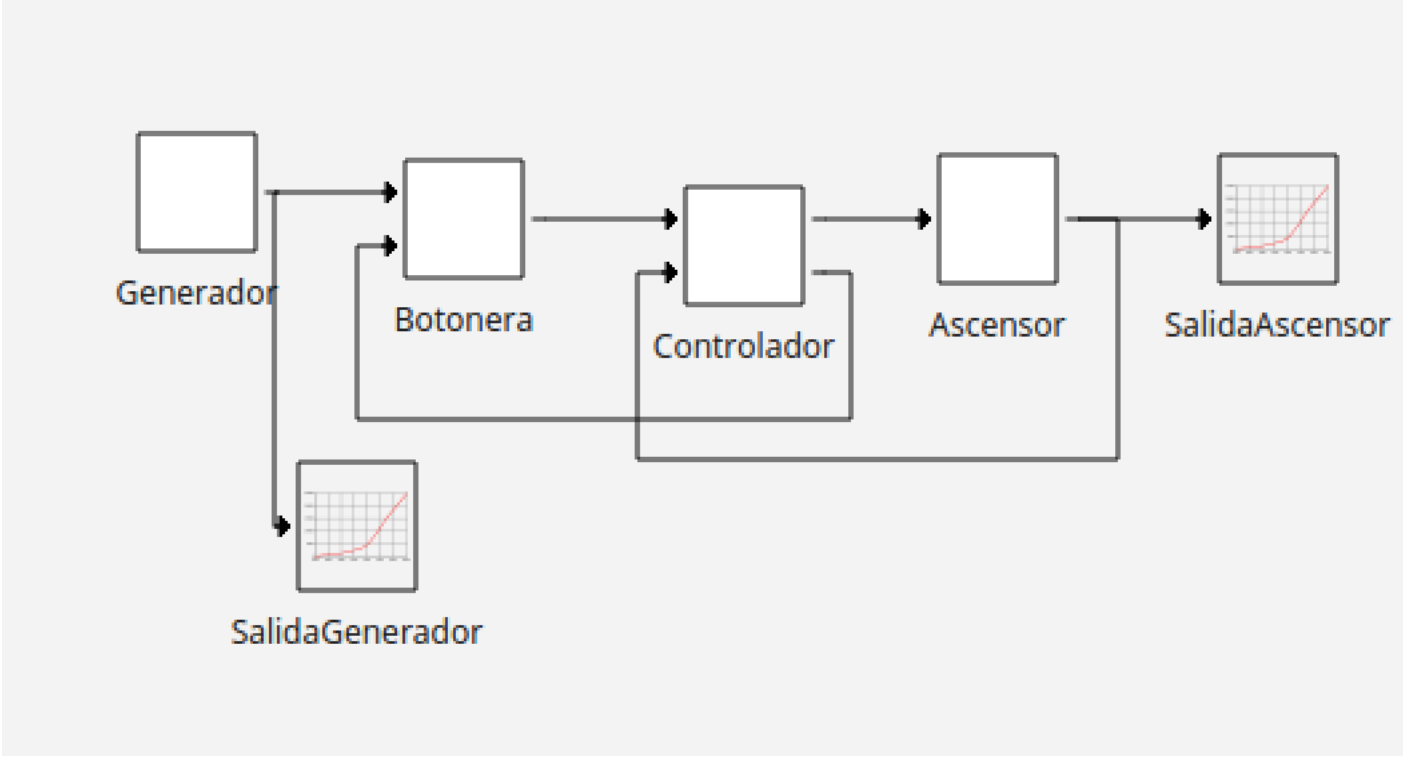
$SInicial = (4, 0)$

3 Devs acoplados

3.1 Dev acoplado para 1 ascensor

N1 $\langle X, Y, D, \{Md\}, EIC, EOC, IC, Select\rangle$

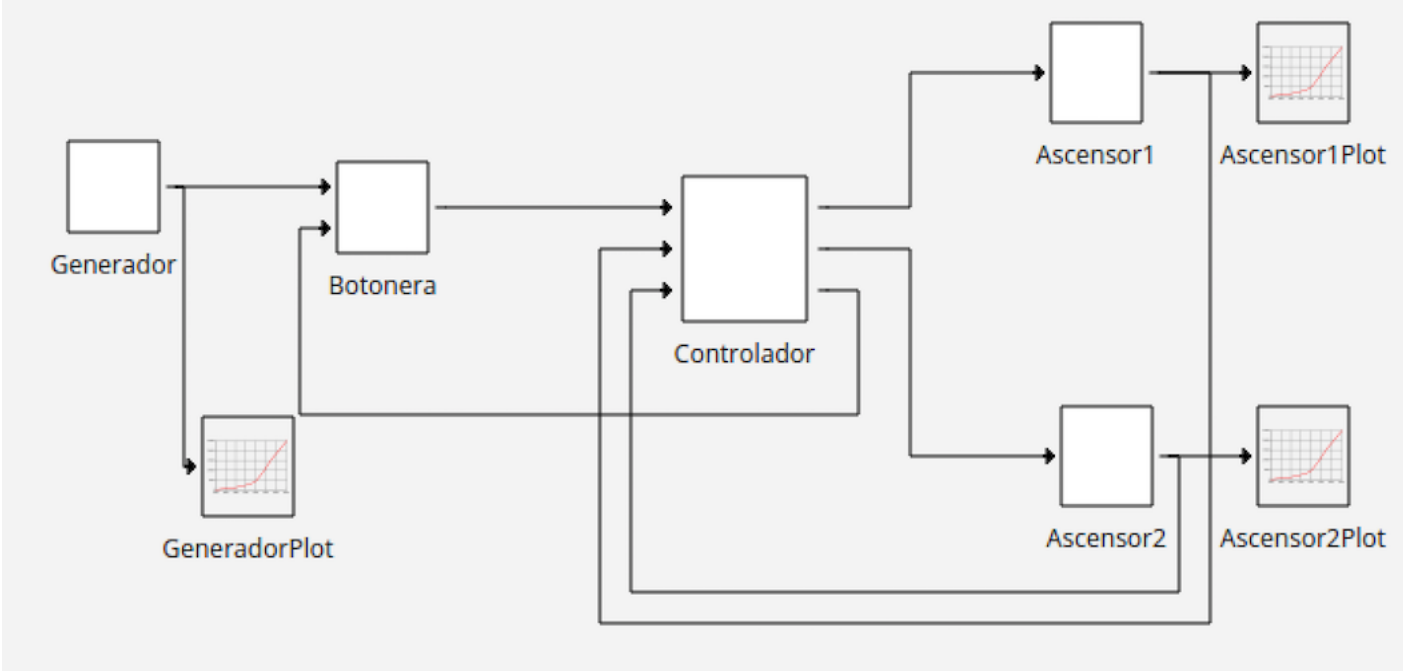
$X = \emptyset$
$Y = \emptyset$
$D = \{Ascensor, Controlador, Botonera, Generador\}$
$\{Md\} = \{Ascensor \rightarrow AscensorDev, Controlador \rightarrow Controlador1Dev, Botonera \rightarrow BotoneraDev,$
$Generador \rightarrow GeneradorDev\}$
$EIC = \emptyset$
$EOC = \emptyset$
$IC = \{[(Generador, 0), (Botonera, 0)], [(Botonera, 0), (Controlador, 0)], [(Controlador, 0), (Ascensor, 0)],$
$[(Controlador, 1), (Botonera, 1)], [(Ascensor, 0), (Controlador, 1)]\}$
$Select = (Ascensor, Controlador, Botonera, Generador)$



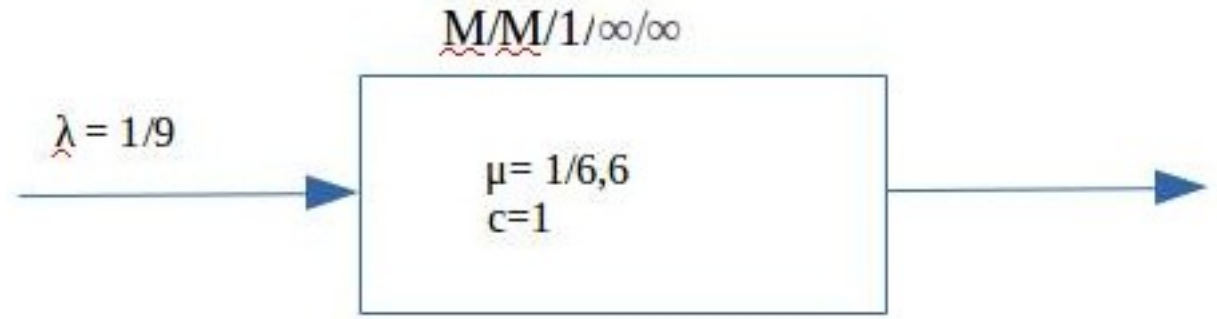
3.2 Dev acoplado para 2 ascensores

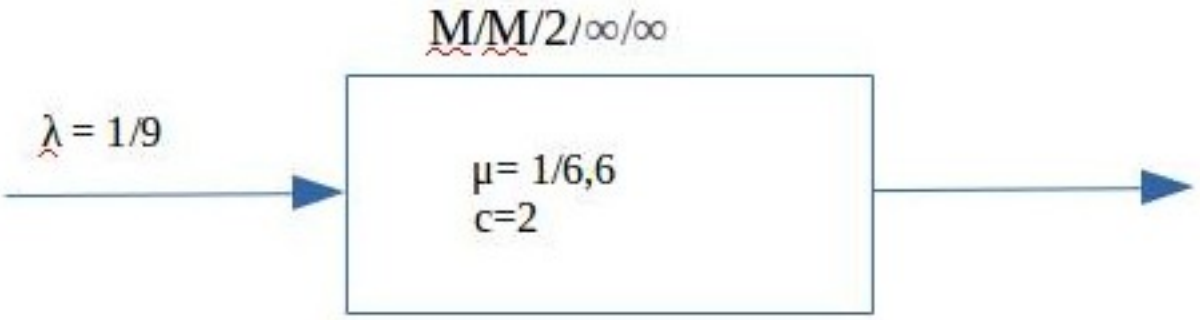
N2 $\langle X, Y, D, \{Md\}, EIC, EOC, IC, Select\rangle$

$X = \emptyset$
$Y = \emptyset$
$D = \{Ascensor1, Ascensor2, Controlador, Botonera, Generador\}$
$\{Md\} = \{Ascensor1 \rightarrow \ AscensorDev, Ascensor2 \rightarrow AscensorDev, Controlador \rightarrow Controlador2Dev,$
$Botonera \rightarrow BotoneraDev, Generador \rightarrow GeneradorDev\}$
$EIC = \emptyset$
$EOC = \emptyset$
$IC = \{[(Generador, 0), (Botonera, 0)], [(Botonera, 0), (Controlado, 0)], [(Controlador, 0), (Ascensor1, 0)],$
$[(Controlador, 1), (Ascensor2, 0)], [(Ascensor1, 0), (Controlador, 1)], [(Ascensor2, 0), (Controlador, 2)],$
$[(Controlador, 2), (Botonera, 1)]\}$
$Select = (Controlador, Ascensor1, Ascensor2, Botonera, Generador)$



4 Análisis de cola





Tenemos como dato que el ascensor se mueve a 3,3 pisos (en promedio) para atender a una petición, como cada piso tiene 2 metros entonces el ascensor recorrer 6,6 metros por cada petición a 1 m/s. Es lo mismo decir que el tiempo promedio de servicio es 6,6s.
Con lo que podemos deducir que $\mu = \frac{1}{6,6}$.

Para ver si el sistema es estable tengo que analizar si $\lambda < \mu.c$
(1) $\frac{1}{9} < \frac{1}{6,6}.1 \rightarrow 0,11 < 0,15$
(2) $\frac{1}{9} < \frac{1}{6,6}.2 \rightarrow 0,11 < 0,30$

Luego, los dos sistemas son estables.

4.1 Cálculo de las variables estadísticas

Las variables estadísticas para el sistema de colas con un sólo ascensor son

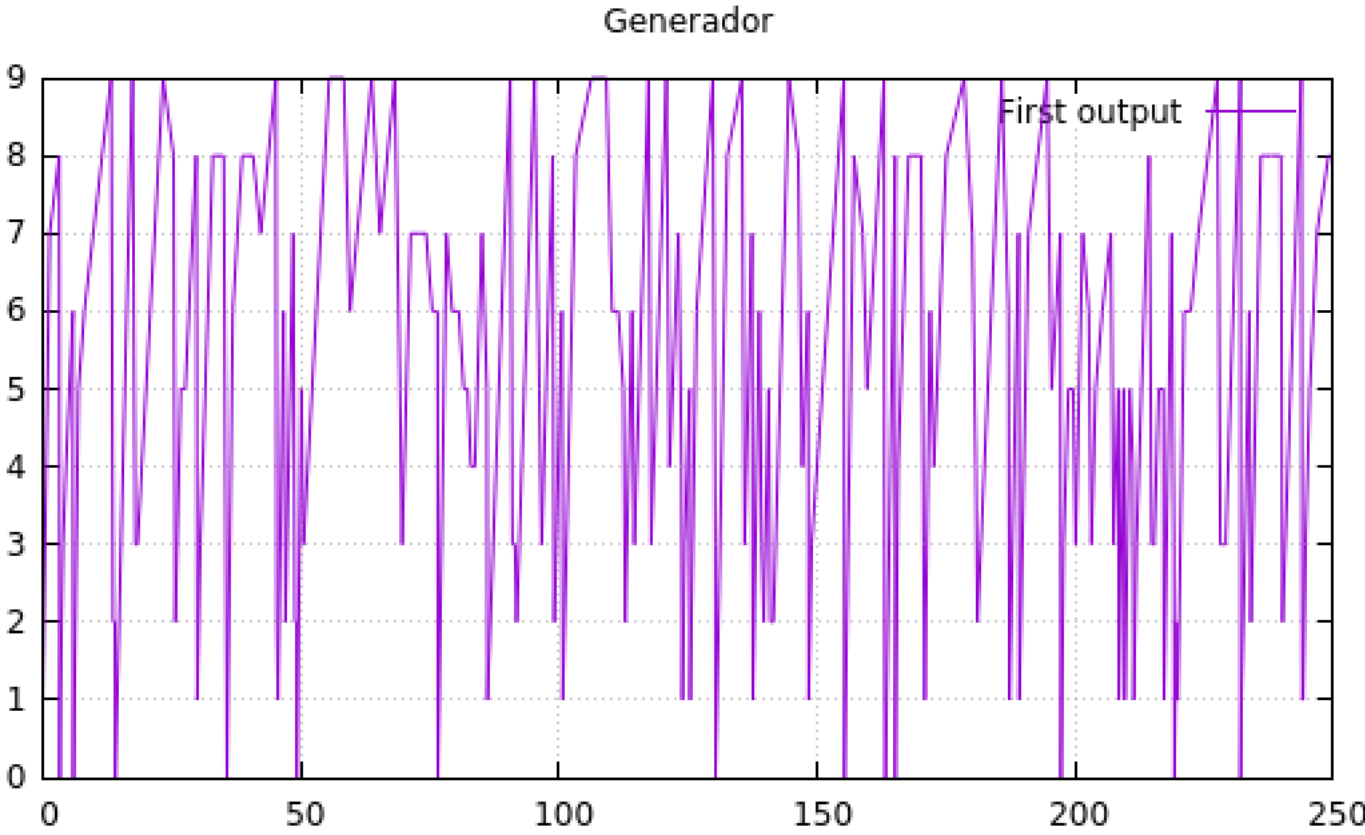
M/M/c Queue		
lambda	0.111	(arrival rate)
mu	0.152	(service rate)
c	1.000	(number of servers)
rho	0.733	(utilization)
L	2.750	(mean number in system)
w	24.750	(mean time in system)
wQ	18.150	(mean time in queue)
LQ	2.017	(mean number in queue)
P0	0.267	(probability of an empty system)

Las variables estadísticas para el sistema de colas con 2 ascensores son

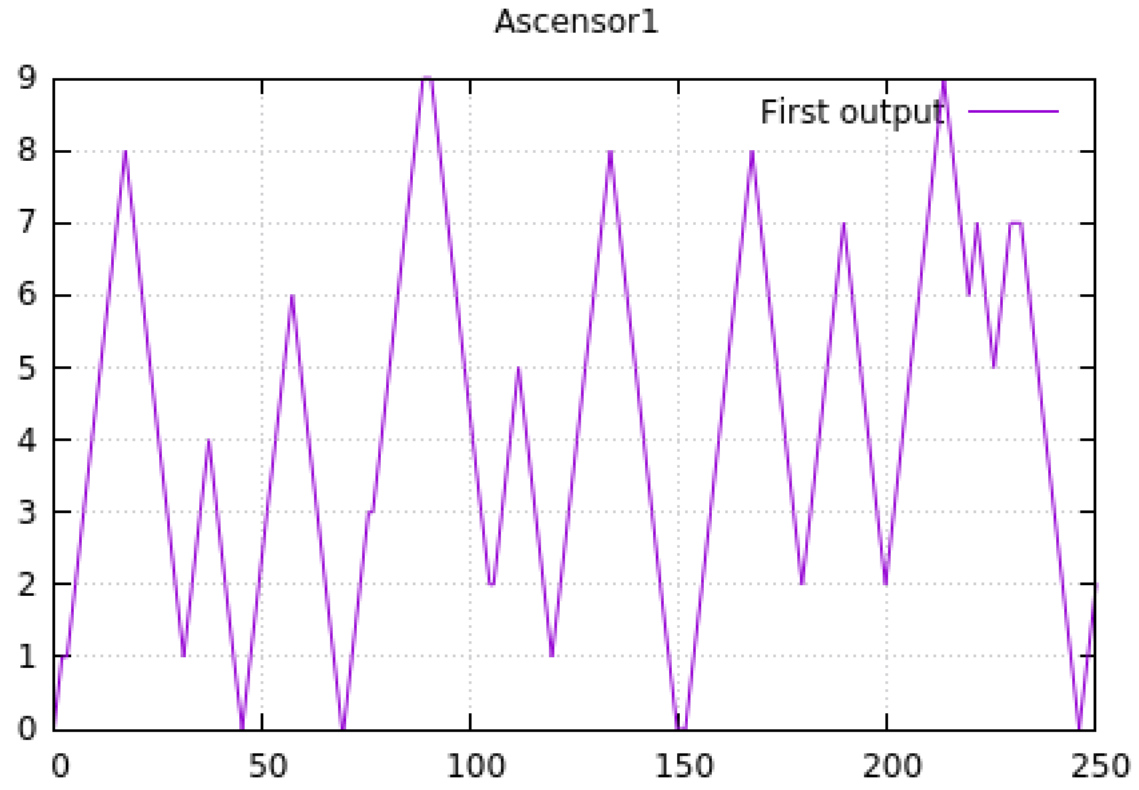
M/M/c Queue		
lambda	0.111	(arrival rate)
mu	0.152	(service rate)
c	2.000	(number of servers)
rho	0.367	(utilization)
L	0.847	(mean number in system)
w	7.625	(mean time in system)
wQ	1.025	(mean time in queue)
LQ	0.114	(mean number in queue)
P0	0.463	(probability of an empty system)

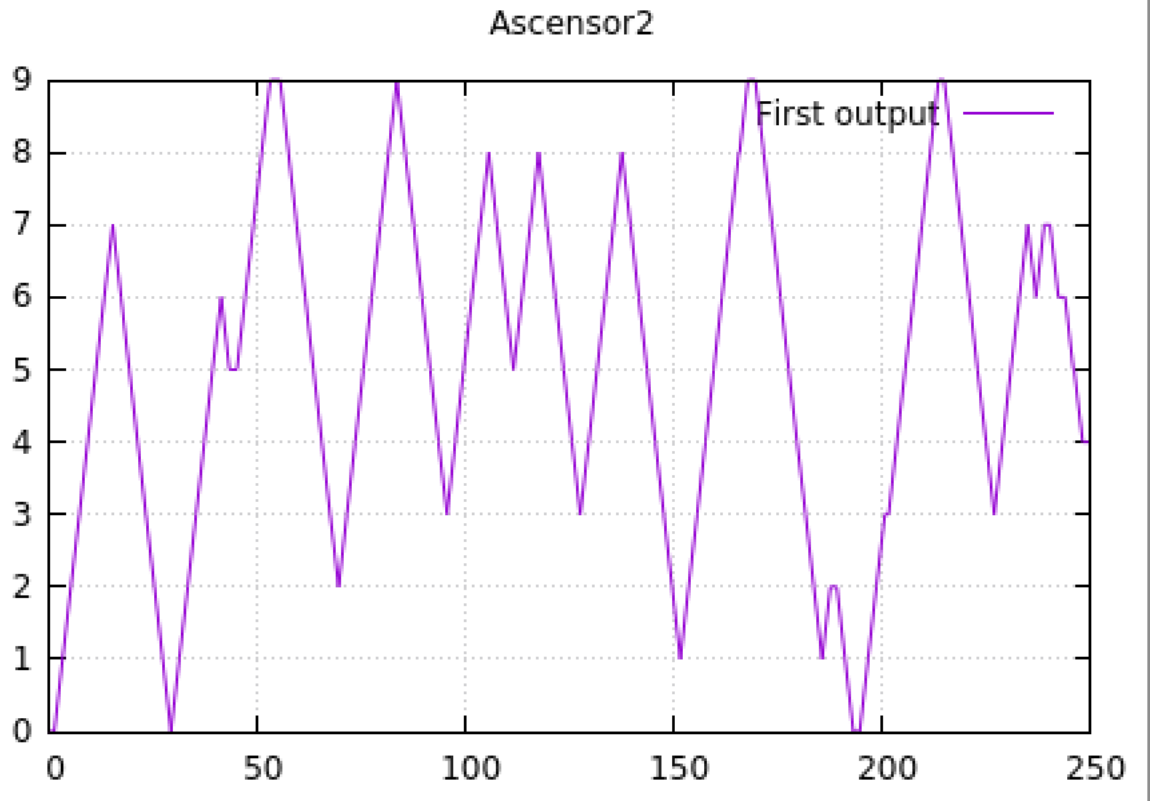
5 Comparativa del Tiempo de Respuesta

Si analizamos una ejecución con igual semilla para las tres heurísticas obtenemos las siguientes salidas:

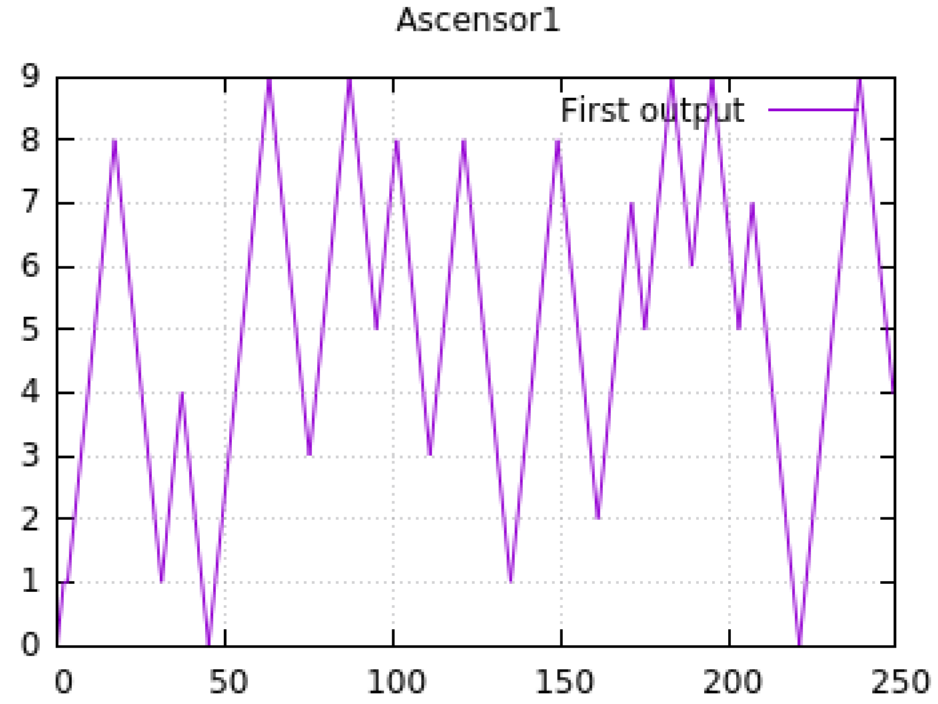


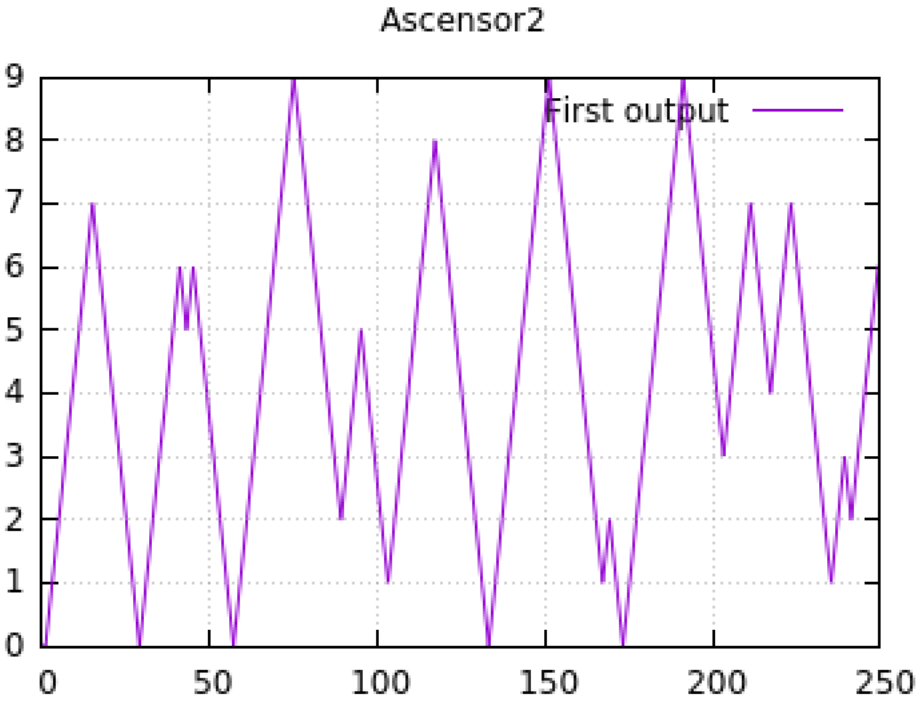
• Primera Heurística: Las peticiones se alternan entre los ascensores.



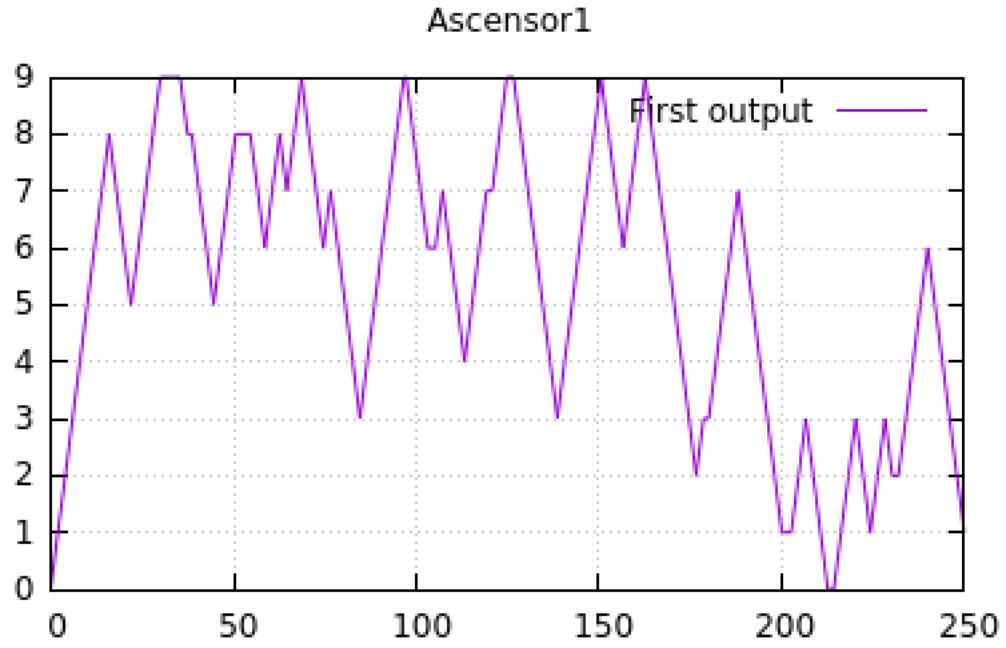


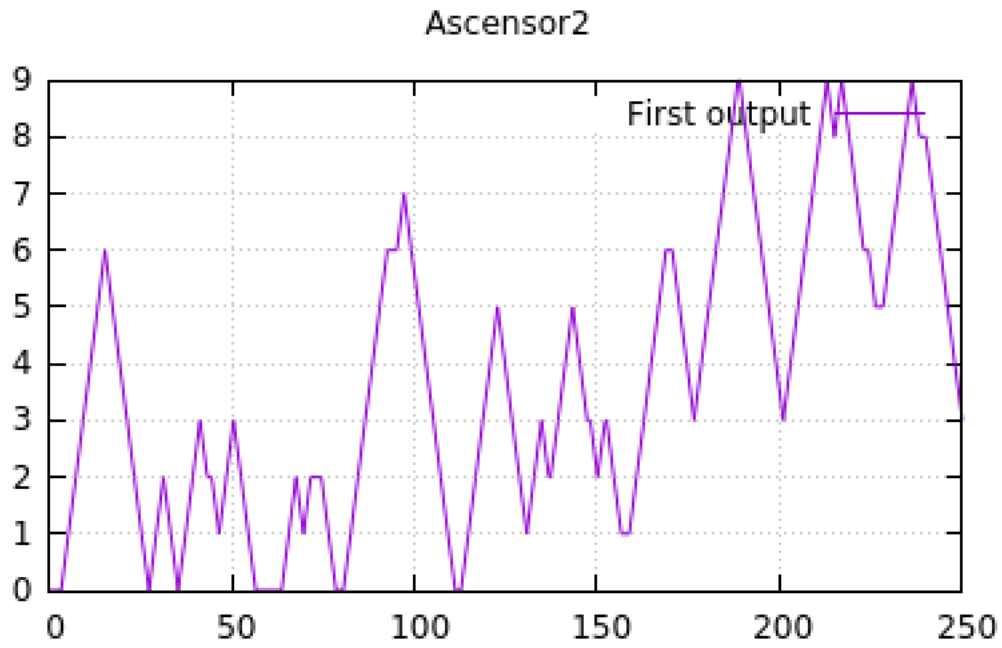
- Segunda Heurística: Las peticiones se asignan a un ascensor libre, priorizando siempre al ascensor 1.





- Tercera Heurística: Las peticiones se le asignan al ascensor que más rápido llegue a atenderlas.





En los gráficos de la heurística tres se puede notar un gran cambio en la forma en que se toman las peticiones,respecto de las de las otras dos heurísticas, podemos observar que en los primeros segundos se simulación (hasta el 170 aproximadamente) el ascensor 1 se encarga de tratar las peticiones de pisos altos, mientras que el ascensor 2 se encarga de cubrir las peticiones de pisos bajos. Una vez que pasan esos 180 segundos iniciales vemos que la tendencia se invierte haciendo que el ascensor 1 pase a cubrir las peticiones de pisos bajos mientras que el ascensor 2 se encarga de las de pisos altos.

Esto nos da lugar a pensar que el tiempo se espera de una petición en el sistema disminuye ya que cada ascensor recorre distancias más cortas para cumplir con las peticiones.

A continuación se plantearán los resultados de calcular el tiempo de espera promedio en esta simulación

teniendo en cuenta un tiempo de simulación de 2000, para corroborar analíticamente que ocurre con lo que suponemos.

Análisis de salida
• Primera heurística

Wq promedio = 654.50

• Segunda heurística

Wq promedio = 641.21

• Tercera heurística

Wq promedio = 393.79

De esta manera podemos confirmar lo que sospechábamos, la tercera heurística reduce significativamente el tiempo de espera de una petición.