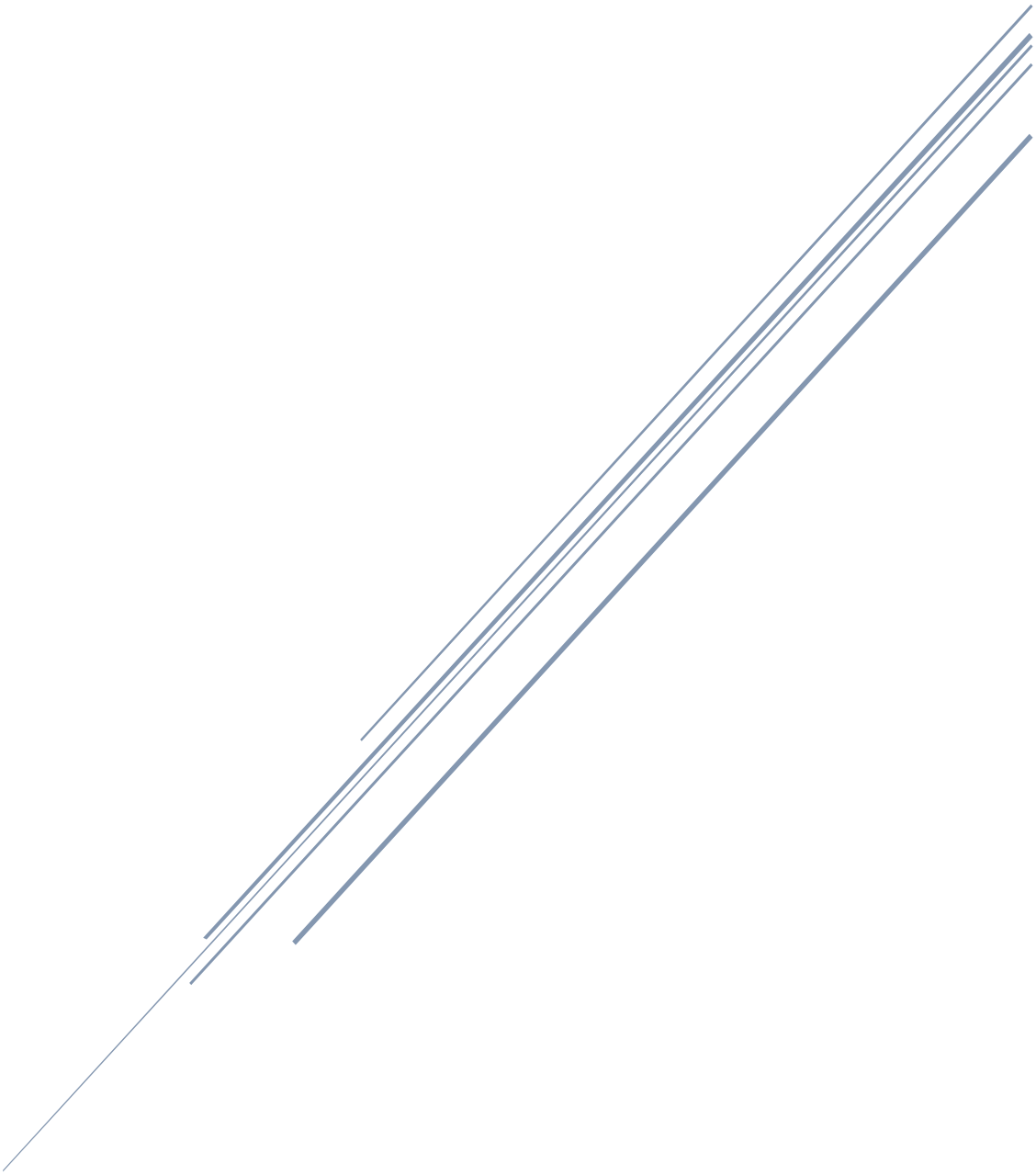
EVALUACIÓN DE CONFIGURACIONES

Práctica 6

–

Análisis operacional



FDI - UCM

Iván Aguilera Calle – Daniel García Moreno

1. PDQ

1.1.Herramienta

Pretty Damn Quick (PDQ) es una herramienta de código abierto que permite representar y solucionar sistemas informáticos modernos. La herramienta consta de una serie de funciones para poder trabajar:

`Init()`	Initializes all internal PDQ variables. Must be called prior to any other PDQ function.
`CreateOpen()`	Defines the characteristics of a workload in an open-circuit queueing model.
`CreateClosed()`	Defines the characteristics of a workload in a closed-circuit queueing model.
`CreateNode()`	Defines a single queueing-center in either a closed or open circuit queueing model.
`SetDemand()`	Defines the service demand of a specific workload at a specified node.
`SetVisits()`	Define the service demand in terms of the service time and visit count.
`SetDebug()`	enables diagnostic printout of PDQ internal variables.
`Solve()`	The solution method must be passed as an argument.
`GetResponse()`	Returns the system response time for the specified workload.
`GetThruput()`	Returns the system throughput for the specified workload.
`GetUtilization()`	Returns the utilization of the designated queueing node by the specified workload.
`Report()`	Generates a formatted report containing system, and node level performance measures.

Además, nos permite trabajar con variables.

Para comenzar, analizaremos el código de ejemplo de la sección 4.2.

```
#!/usr/bin/perl

use pdq;

# Globals
$arrivRate = 0.75;
$servTime  = 1.0;

# Initialize PDQ and add a comment about the model
pdq::Init("Open Network with M/M/1");
pdq::SetComment("This is just a very simple example.");

# Define the workload and circuit type
pdq::CreateOpen("Work", $arrivRate);

# Define the queueing center
pdq::CreateNode("Server", $pdq::CEN, $pdq::FCFS);

# Define service demand due to workload on the queueing center
pdq::SetDemand("Server", "Work", $servTime);

# Change units labels to suit
pdq::SetWUnit("Cust");
pdq::SetTUnit("Secs");

# Solve the model
#  Must use the Canonical method for an open network
pdq::Solve($pdq::CANON);

# Generate a generic performance report
pdq::Report();
```

En el código se declaran variables, que corresponde a la tasa de llegadas (0,75) y el tiempo de servicio (1). Posteriormente, con la función *Init()* inicializa el modelo. Con la función *CreateOpen()* se define la carga de trabajo y el tipo de sistema. En este ejemplo, se está definiendo un sistema abierto. Con la función *CreateNode(),* se crean los distintos elementos del sistema informático con sus respectivas cola, pasando por parámetro la política de las mismas, en este ejemplo, con política FCFS. Con *SetDemand()* ajustamos la demanda de una carga de trabajo y el tiempo de servicio requerido para dicha carga de trabajo de un nodo específico. Con la función *SetWUnit()* establecemos las unidades de las variables de trabajo que aparecerán posteriormente en el informe final. Por defecto, la unidad de trabajo es "Job". Análogamente, con la función *SetTUnit()* estableceremos la unidad de tiempo de las variables de trabajo.

Una vez se tiene definido, ajustado y creado el modelo PDQ, utilizaremos la función *Solve()* para probar el modelo y obtener las métricas, para generar, posteriormente, con la función *Report(),* el informe final. La función *Solve()* recibe como parámetro el método de resolución del modelo. Este puede ser "CANON" si se ha creado un sistema abierto o en cambio, si estamos ante un sistema cerrado, podemos utilizar "EXACT" si queremos limitar el sistema a 1000 usuario o "APPROX" si queremos usuarios ilimitados.

Para probar el modelo, ejecutamos el siguiente comando:

```
$ perl -Iperl5/blib/lib -Iperl5/blib/arch test.pl
```

Obtenemos el siguiente informe final:

```
                PRETTY DAMN QUICK REPORT
=========================================
***  on   Sun May  7 10:24:22 2017     ***
***  for  Open Network with M/M/1      ***
***  PDQ  Version 6.2.0 Build 082015   ***
=========================================

COMMENT: This is just a very simple example.

=========================================
********    PDQ Model INPUTS      ********
```

```
                    ========================================

WORKLOAD Parameters:

Node Sched Resource   Workload   Class     Demand
---- ----- --------   --------   -----     ------
  1  FCFS  Server     Work       Open      1.0000

Queueing Circuit Totals
Streams:   1
Nodes:     1

Arrivals      per Secs      Demand
--------      --------      -------
Work          0.7500        1.0000


                    ========================================
                    ********  PDQ Model OUTPUTS      ********
                    ========================================

Solution Method: CANON

                    ********  SYSTEM Performance     ********

Metric                 Value    Unit
------                 -----    ----
Workload: "Work"
Number in system         3.0000    Cust
Mean throughput          0.7500    Cust/Secs
Response time            4.0000    Secs
Stretch factor           4.0000

Bounds Analysis:
Max throughput           1.0000    Cust/Secs
Min response             1.0000    Secs


                    ********  RESOURCE Performance   ********

Metric         Resource     Work           Value    Unit
------         --------     ----           -----    ----
Capacity       Server       Work               1    Servers
Throughput     Server       Work          0.7500    Cust/Secs
In service     Server       Work          0.7500    Cust
Utilization    Server       Work         75.0000    Percent
Queue length   Server       Work          3.0000    Cust
Waiting line   Server       Work          2.2500    Cust
Waiting time   Server       Work          3.0000    Secs
Residence time Server       Work          4.0000    Secs
```

1.2.Ejercicios

1.2.1. Ejercicio 5

Dispositivo	V_i	S_i **(s)**
Procesador (1)	17	0,03
Disco (2)	6	0,04
Disco (3)	10	0,04

Estamos ante un modelo **abierto**. El tiempo medio entre llegadas de clientes es de 0,6 segundos. En la tabla anterior podemos observar las diferentes razones de visita y los tiempos de servicio de los distintos componentes del sistema. La tasa de llegada es la inversa del tiempo medio entre llegadas (1,67).

Teniendo en cuenta los siguientes datos, elaboramos el siguiente código: creamos variables globales para cada uno de los datos conocidos, inicializamos el modelo, creamos la cola de trabajo, creamos los diferentes dispositivos del modelo, ajustando sus respectivas razones de visita y tiempos de servicio, establecemos las unidades que queremos que aparezcan en el informe final y resolvemos el modelo. Utilizamos la función *SetVisits()*, tal y como se especifica en el enunciado:

```
#!/usr/bin/perl

use pdq;

# Globals

# Tasa de llegadas
$arrivRate = 1.6666666;

#Tiempos de servicio
$procesadorServTime  = 0.03;
$discoServTime  = 0.04;

#Razones de visita
$procesadorVisitRate = 17.0;
$disco1VisitRate = 6.0;
$disco2VisitRate = 10.0;
```

```
# Initialize PDQ and add a comment about the model
pdq::Init("Modelo PDQ - Ejercicio 5");
pdq::SetComment("Ejercicio 5 - ECO - TEMA 5");

# Define the workload and circuit type
pdq::CreateOpen("Work", $arrivRate);

# Define the queueing center
pdq::CreateNode("Procesador", $pdq::CEN, $pdq::FCFS);
pdq::CreateNode("Disco 1", $pdq::CEN, $pdq::FCFS);
pdq::CreateNode("Disco 2", $pdq::CEN, $pdq::FCFS);

# Used to define the service demand of a specific workload in terms of the explicit service time and
visit count.
pdq::SetVisits("Procesador", "Work", $procesadorVisitRate, $procesadorServTime);
pdq::SetVisits("Disco 1", "Work", $disco1VisitRate, $discoServTime);
pdq::SetVisits("Disco 2", "Work", $disco2VisitRate, $discoServTime);

# Change units labels to suit
pdq::SetWUnit("Trabajos");
pdq::SetTUnit("Segundos");

# Solve the model
#  Must use the Canonical method for an open network
pdq::Solve($pdq::CANON);

# Generate a generic performance report
pdq::Report();
```

Una vez resuelto el modelo, se genera el informe final, el cual podemos ver a continuación. Subrayaremos en amarillo los valores que posteriormente compararemos con los ejercicios realizados en clase. A la derecha podremos observar un número ayuda para poder encontrar el valor que posteriormente será calculado de manera manual:

PRETTY DAMN QUICK REPORT

==========

*** on Sun May 7 11:04:56 2017 ***

*** for Modelo PDQ - Ejercicio 5 ***

*** PDQ Version 6.2.0 Build 082015 ***

==========

PDQ_Report warning: No PDQ service demands defined.

COMMENT: Ejercicio 5 - ECO - TEMA 5

==========

******** PDQ Model INPUTS ********

==========

WORKLOAD Parameters:

Node	Sched	Resource	Workload	Class	Visits	Service	Demand
1	FCFS	Procesador	Work	Open	17.0000	0.0300	0.5100
1	FCFS	Disco 1	Work	Open	6.0000	0.0400	0.2400
1	FCFS	Disco 2	Work	Open	10.0000	0.0400	0.4000

1

Queueing Circuit Totals

Streams: 1

Nodes: 3

Arrivals	per Segundos	Demand
Work	1.6667	1.1500

==========

******** PDQ Model OUTPUTS ********

==========

Solution Method: CANON

******** SYSTEM Performance ********

Metric	Value	Unit
Workload: "Work"		
Number in system	8.3333	Trabajos
Mean throughput	1.6667	Trabajos/Segundos
Response time	5.0000	Segundos
Stretch factor	4.3478	

7

6

Bounds Analysis:

Max throughput	1.9608	Trabajos/Segundos
Min response	1.1500	Segundos

4

2

******** RESOURCE Performance ********

Metric	Resource	Work	Value	Unit
Capacity	Procesador	Work	1	Servers
Throughput	Procesador	Work	28.3333	Visits/Segundos
In service	Procesador	Work	0.8500	Trabajos
Utilization	Procesador	Work	85.0000	Percent
Queue length	Procesador	Work	5.6667	Trabajos
Waiting line	Procesador	Work	4.8167	Trabajos
Waiting time	Procesador	Work	2.8900	Segundos
Residence time	Procesador	Work	3.4000	Segundos
Waiting time	Procesador	Work	2.8900	Segundos
Capacity	Disco 1	Work	1	Servers
Throughput	Disco 1	Work	10.0000	Visits/Segundos
In service	Disco 1	Work	0.4000	Trabajos
Utilization	Disco 1	Work	40.0000	Percent
Queue length	Disco 1	Work	0.6667	Trabajos
Waiting line	Disco 1	Work	0.2667	Trabajos
Waiting time	Disco 1	Work	0.1600	Segundos
Residence time	Disco 1	Work	0.4000	Segundos
Waiting time	Disco 1	Work	0.1600	Segundos
Capacity	Disco 2	Work	1	Servers
Throughput	Disco 2	Work	16.6667	Visits/Segundos
In service	Disco 2	Work	0.6667	Trabajos
Utilization	Disco 2	Work	66.6667	Percent
Queue length	Disco 2	Work	2.0000	Trabajos
Waiting line	Disco 2	Work	1.3333	Trabajos
Waiting time	Disco 2	Work	0.8000	Segundos
Residence time	Disco 2	Work	1.2000	Segundos
Waiting time	Disco 2	Work	0.8000	Segundos

3

En primer lugar, calculamos las demandas de servicio de los diferentes dispositivos y observamos que los valores son idénticos a los obtenidos en el informe:

$D_i = V_i \times S_i$

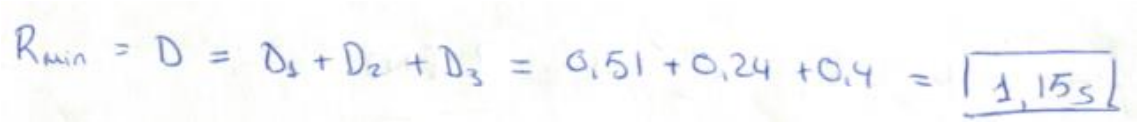
$D_1 = V_1 \times S_1 = 17 \times 0,03 = 0,51\,s$

$D_2 = V_2 \times S_2 = 6 \times 0,04 = 0,24\,s$

$D_3 = V_3 \times S_3 = 10 \times 0,04 = 0,4\,s$

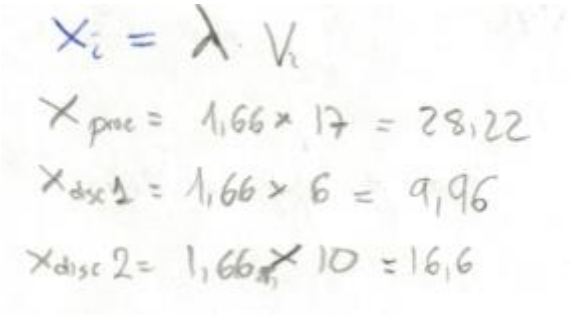
1

En segundo lugar, calculamos el tiempo de respuesta mínimo y el valor es idéntico al obtenido en el informe:



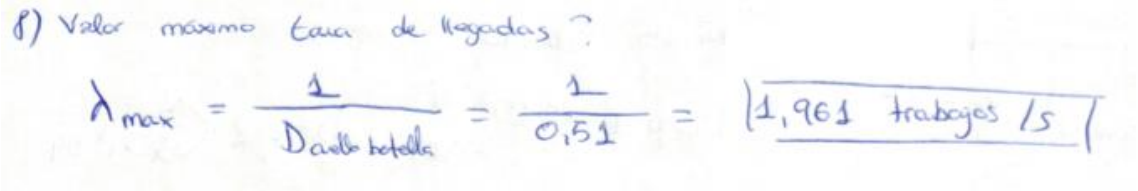
2

En tercer lugar, calculamos la productividad de los distintos dispositivos del modelo. Los valores obtenidos difieren en unos pocos decimales a los obtenidos en el informe:



3

En cuarto lugar, calculamos el valor máximo de la tasa de llegadas obteniéndose un valor casi idéntico al obtenido en el informe a falta de unos pocos decimales:



4

En quinto lugar, calculamos los tiempos de respuesta. **No se observan estos valores en el informe (la herramienta no los proporciona).**

$$R_i = \frac{S_i}{1-U_i} = \frac{S_i}{1-(X_i \cdot S_i)}$$

$$R_0 = \frac{0{,}03}{1-(28{,}22 \cdot 0{,}03)} = 0{,}195$$

$$R_2 = \frac{0{,}04}{1-(9{,}96 \cdot 0{,}04)} = 0{,}066$$

$$R_3 = \frac{0{,}04}{1-(16{,}6 \cdot 0{,}04)} = 0{,}119$$

En sexto lugar, calculamos el tiempo de respuesta del sistema, obteniendo una diferencia de 0,1 segundos respecto al valor obtenido en el informe a causa del redondeo de decimales:

$$R = (V_1 \times R_1) + (V_2 \times R_2) + (V_3 \times R_3) =$$
$$= (17 \cdot 0{,}195) + (6 \cdot 0{,}066) + (10 \cdot 0{,}119) = \boxed{4{,}95}$$

6

Por último, calculamos el número de trabajos del sistema. Obtenemos una diferencia de 0,20 peticiones, a causa de la acumulación de redondeos de decimales:

$$N = \lambda \cdot R = 1{,}66 \times 4{,}9 = \boxed{8{,}13 \text{ peticiones}}$$

Ley de Little $\rightarrow N = X_0 \cdot R = \lambda R$

7

1.2.2. Ejercicio 6

Dispositivo	V_i	S_i **(s)**
Procesador (1)	4	0,5
Disco (2)	3	0,75

Estamos ante un modelo **cerrado** interactivo con 25 usuarios. El tiempo medio de reflexión es de 6 segundos. En la tabla anterior podemos observar las diferentes razones de visita y los tiempos de servicio de los distintos componentes del sistema. En este ejercicio, solo disponemos de dos elementos (nodos).

En este ejercicio, como estamos ante un modelo cerrado, al llamar a la función *CreateClosed()*, pasaremos por parámetro el tiempo medio de reflexión a partir del parámetro "think" de la función. Como el valor de "think" va a ser distinto de cero, indicaremos, también, a la función *CreatedClosed()* mediante el parámetro "TERM".

Para solucionar el modelo, al llamar a la función *Solve()* indicaremos por parámetro "EXACT" que el sistema estará limitado, como máximo, a 1000 usuarios. En nuestro caso, tenemos 25 usuarios en nuestro sistema.

```
#!/usr/bin/perl

use pdq;

# Globals

# Numero de usuarios del sistema
$users = 25.0;

# Tiempo de reflexion
$think = 6.0;

#Tiempos de servicio
$procesadorServTime  = 0.5;
$discoServTime  = 0.75;

#Razones de visita
$procesadorVisitRate = 4.0;
$discoVisitRate = 3.0;
```

```
# Initialize PDQ and add a comment about the model
pdq::Init("Modelo PDQ - Ejercicio 6");
pdq::SetComment("Ejercicio 6 - ECO - TEMA 6");

# Define the workload and circuit type
pdq::CreateClosed("Work", $pdq::TERM, $users, $think);

# Define the queueing center
pdq::CreateNode("Procesador", $pdq::CEN, $pdq::FCFS);
pdq::CreateNode("Disco", $pdq::CEN, $pdq::FCFS);

# Used to define the service demand of a specific workload in terms of the explicit service time and
visit count.
pdq::SetVisits("Procesador", "Work", $procesadorVisitRate, $procesadorServTime);
pdq::SetVisits("Disco", "Work", $discoVisitRate, $discoServTime);

# Change units labels to suit
pdq::SetWUnit("Peticiones");
pdq::SetTUnit("Segundos");

# Solve the model
#  Must use the Canonical method for an open network
pdq::Solve($pdq::EXACT);

# Generate a generic performance report
pdq::Report();
```

Una vez resuelto el modelo, se genera el informe final, el cual podemos ver a continuación. Subrayaremos en amarillo los valores que posteriormente compararemos con los ejercicios realizados en clase. A la derecha podremos observar un número ayuda para poder encontrar el valor que posteriormente será calculado de manera manual:

PRETTY DAMN QUICK REPORT

==========

*** on Sun May 7 11:33:00 2017 ***

*** for Modelo PDQ - Ejercicio 6 ***

*** PDQ Version 6.2.0 Build 082015 ***

==========

PDQ_Report warning: No PDQ service demands defined.

COMMENT: Ejercicio 6 - ECO - TEMA 6

==========

******** PDQ Model INPUTS ********

==========

WORKLOAD Parameters:

Node	Sched	Resource	Workload	Class	Visits	Service	Demand
1	FCFS	Procesador	Work	Closed	4.0000	0.5000	2.0000
1	FCFS	Disco	Work	Closed	3.0000	0.7500	2.2500

1

Queueing Circuit Totals

Streams: 1

Nodes: 2

Client	Number	Demand	Thinktime
Work	25.00	4.2500	6.00

==========

******** PDQ Model OUTPUTS ********

==========

Solution Method: EXACT

******** SYSTEM Performance ********

Metric	Value	Unit
Workload: "Work"		
Mean concurrency	22.3566	Peticiones
Mean throughput	0.4406	Peticiones/Segundos
Response time	50.7454	Segundos
Round trip time	56.7454	Segundos
Stretch factor	11.9401	

5

Bounds Analysis:

Max throughput	0.4444	Peticiones/Segundos
Min response	4.2500	Segundos
Max Demand	2.2500	Segundos
Tot demand	4.2500	Segundos
Think time	6.0000	Segundos
Optimal clients	4.5556	Clients

4

2

3

******** RESOURCE Performance ********

Metric	Resource	Work	Value	Unit
Capacity	Procesador	Work	1	Servers
Throughput	Procesador	Work	1.7623	Visits/Segundos
In service	Procesador	Work	0.8811	Peticiones
Utilization	Procesador	Work	88.1129	Percent
Queue length	Procesador	Work	6.3936	Peticiones
Waiting line	Procesador	Work	5.5125	Peticiones
Waiting time	Procesador	Work	12.5124	Segundos
Residence time	Procesador	Work	14.5124	Segundos
Waiting time	Procesador	Work	12.5124	Segundos
Capacity	Disco	Work	1	Servers
Throughput	Disco	Work	1.3217	Visits/Segundos
In service	Disco	Work	0.9913	Peticiones
Utilization	Disco	Work	99.1270	Percent
Queue length	Disco	Work	15.9630	Peticiones
Waiting line	Disco	Work	14.9717	Peticiones
Waiting time	Disco	Work	33.9831	Segundos
Residence time	Disco	Work	36.2331	Segundos
Waiting time	Disco	Work	33.9831	Segundos

En primer lugar, calculamos las demandas de servicio de los diferentes dispositivos y observamos que los valores son idénticos a los obtenidos en el informe:

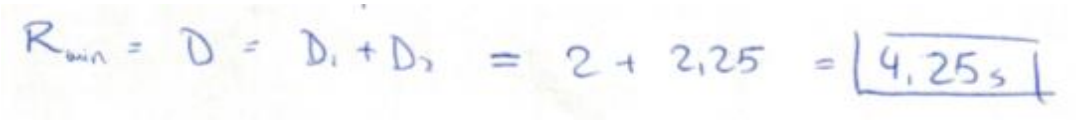
$$D_i = V_i \cdot S_i$$

$$D_{proc} = 4 \times 0{,}5 = 2_s$$

$$D_{disco} = 3 \times 0{,}75 = 2{,}25_s$$

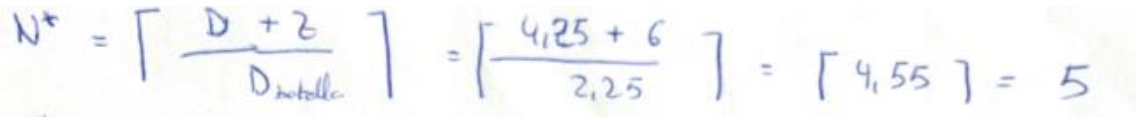
1

En segundo lugar, calculamos el tiempo mínimo de respuesta, obteniendo el mismo resultado que el podemos observar en el informe:



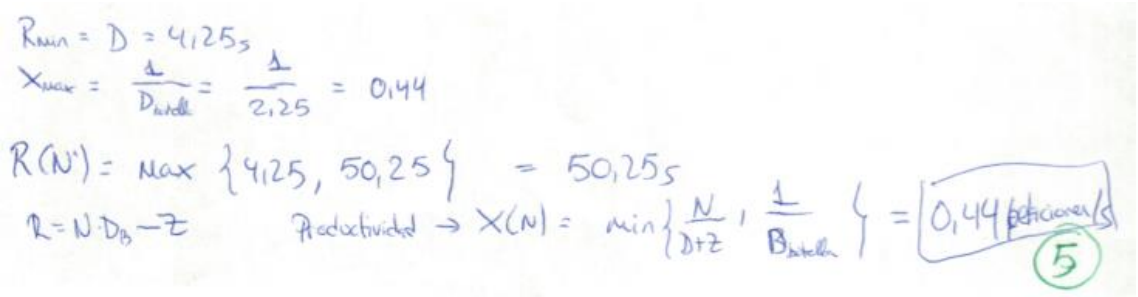
2

En tercer lugar, calculamos el punto teórico de saturación. Obtenemos el mismo resultado que en informe, con la salvedad de que no se aplica la función techo:



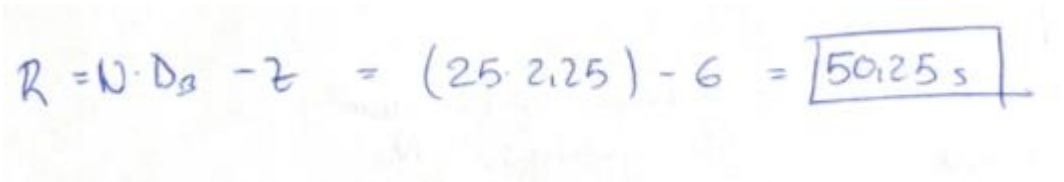
3

En cuarto lugar, el límite dek tiempo de respuesta. Obtenemos el mismo resultado que se puede observar en el informe:



4

Por último, calculamos el tiempo medio de respuesta, obteniendo una diferencia de 0,49 segundos respecto al valor obtenido en el informe:



5

Ante este desbarajuste, nos dimos cuenta que no estábamos aplicando la fórmula correcta. La fórmula a aplicar es la siguiente:

$$R = \left(\frac{N}{X}\right) - 2 = \left(\frac{25}{0{,}44}\right) - 6$$

$$\boxed{R = 50{,}74080 \text{ segundos}}$$

Ahora sí, el valor calculado es el que aparece en el informe generado por PDQ.