```
matrix = Table[CalculaV[n, w, 1.2, 2/3],{n, 40, 200000, 134},{w, 1,
   1000, 20}]; //AbsoluteTiming
{4984.239674, Null}

Occ = matrix[[All, All, 1]];
Arr = matrix[[All, All, 2]];

Export["Occ.mat", Occ];
Export["Arr.mat", Arr];
```

2. Matlab:

2.1. Tráfico uniforme:

Este código simula las funciones de OCC y ARR de acuerdo a los parámetros de S, OCC_{MAX}, ARR_{MAX} y μ.

Permite hallar los parámetros ideales del sistema: OCC_{IDEAL}, ARR_{IDEAL}, ω_{IDEAL} y N_{MAX}.

```
clc;
close all;
clear all;

%Declaramos las constantes
%%Numero de hosts por switch
s = 48;
%%Maximo numero de TCAM por switch
max_OCC = 8192;
%%Maximo de paquetes que puede recibir el controlador (por segundo)
max_ARR = 51880;
%%Factor de uso deseado (ya que no vamos a utilizar el 100% de los
%%recursos disponibles)
fact_uso = 0.8;
%%Departure rate
mu = 1/5;
%%Cantidad de sesiones unicas en las que podria participar un host
```

```
avg_sessions_per_dest = 4;

%Inicializamos contadores locales para las funciones
%%Cantidad de simulaciones a realizar
SIM = 100;
%%Cantidad de hosts activos en la red (minimo 40, maximo 80000)
%%considerando un aumento de 20 hosts por punto a evaluar
N = 40:20:80000;

%Inicializamos algunas funciones que usaremos
%%Cantidad de TCAM usadas
OCC = zeros(length(N), SIM);
%%Cantidad de paquetes que llegan al controlador
ARR = zeros(length(N), SIM);

for n = 1:100
   %Calculamos el factor rho_w, considerando incrementos del orden
   %de 10
   rho_w(1, n) = 1 / (10 * n * mu);
   %Calculamos el factor rho_arr
   rho_a = (s * avg_sessions_per_dest) ./ (N -
      avg_sessions_per_dest);
   %Se calcula los valores de OCC y de ARR
   OCC(:, n) = N .* (1 - exp(-1 * rho_a)) .* (1 + (1 ./ (exp(rho_a)
      - 1)) .* (1 ./ (1 + rho_w(1, n) ./ rho_a)));
   ARR(:, n) = (mu / s) * (N .^ 2) .* rho_a .* exp(-1 * rho_a) .*
      (1 - (1 ./ (1 + (rho_w(1, n) ./ rho_a))));
end

%Normalizamos los valores de OCC y ARR
OCC = OCC / max_OCC;
ARR = ARR / max_ARR;

%Comparamos con el factor de uso.
%Si un valor es mayor al factor de uso, se setea un valor ‘0’ o ‘1’
%para OCC y ARR, respectivamente
OCC(OCC > fact_uso) = 0;
```

```
ARR(ARR > fact_uso) = 1;

%El punto óptimo de trabajo se encuentra al intersectar las
%funciones normalizadas de OCC y ARR.
%Para hallar las intersecciones, hallamos la distancia en el eje
%vertical entre estas, y tomamos el menor valor
[DIF, I] = min(abs(OCC - ARR), [], 1);

%Debido a que ambas funciones son crecientes, el valor que más se
%acerca al limite (factor de uso) se encuentra en una posicion
%mayor del eje horizontal
[VAL, POS] = max(I);

%Al tener la posicion en los ejes x e y, ya podemos evaluar las
%Funciones OCC y ARR con respecto a estos puntos, hallando asi sus
%valores ideales
OCC_ideal = OCC(VAL, POS) * max_OCC
ARR_ideal = ARR(VAL, POS) * max_ARR
w_ideal = 1 / (rho_w(1, POS) * mu)
N_max = avg_sessions_per_dest * (1 + s / rho_a(1, VAL))
```

2.2. Tráfico Pareto:

Este código simula las funciones de OCC y ARR de acuerdo a los parámetros de S, OCC_{MAX}, ARR_{MAX} y μ, y a las matrices generadas utilizando el script de *Wolfram Mathematica*.

Permite hallar los parámetros ideales del sistema: OCC_{IDEAL}, ARR_{IDEAL}, ω_{IDEAL} y N_{MAX}.

```
clc;
close all;
clear all;

%Declaramos las constantes
%%Numero de hosts por switch
s = 48;
```

```
%%Maximo numero de TCAM por switch
max_OCC = 8192;
%%Maximo de paquetes que puede recibir el controlador (por segundo)
max_ARR = 51880;
%%Factor de uso deseado (ya que no vamos a utilizar el 100% de los
%%recursos disponibles)
fact_uso = 0.8;
%%Departure rate
mu = 1/5;
%%Cantidad de sesiones unicas en las que podria participar un host
avg_sessions_per_dest = 4;

%Inicializamos contadores locales para las funciones
%%Cantidad de simulaciones a realizar
SIM = 50;
%%Cantidad de hosts activos en la red (minimo 40, maximo 80000)
%%considerando un aumento de 20 hosts por punto a evaluar
N = 40:134:200000;

%Inicializamos algunas funciones que usaremos
%%Cantidad de TCAM usadas
OCC = zeros(length(N), SIM);
%%Cantidad de paquetes que llegan al controlador
ARR = zeros(length(N), SIM);

for n = 1:100
  %Calculamos el factor rho_w, considerando incrementos del orden
  %de 10
  rho_w(1, n) = 1 / (20 * n * mu);
  %Calculamos el factor rho_arr
  rho_a = (s * avg_sessions_per_dest) ./ (N -
    avg_sessions_per_dest);
  %Se calcula los valores de OCC y de ARR
  OCC(:, n) = N .* (1 - exp(-1 * rho_a)) .* (1 + (1 ./ (exp(rho_a)
    - 1)) .* (1 ./ (1 + rho_w(1, n) ./ rho_a)));
  ARR(:, n) = (mu / s) * (N .^ 2) .* rho_a .* exp(-1 * rho_a) .*
    (1 - (1 ./ (1 + (rho_w(1, n) ./ rho_a))));
```

```
end

OCC = importdata('Occ.mat');
ARR = importdata('Arr.mat');
%Normalizamos los valores de OCC y ARR
OCC = OCC / max_OCC;
ARR = ARR / max_ARR;

%Comparamos con el factor de uso.
%Si un valor es mayor al factor de uso, se setea un valor 0 o 1
%para OCC y ARR, respectivamente
OCC(OCC > fact_uso) = 0;
ARR(ARR > fact_uso) = 1;

%El punto óptimo de trabajo se encuentra al intersectar las
%funciones normalizadas de OCC y ARR.
%Para hallar las intersecciones, hallamos la distancia en el eje
%vertical entre estas, y tomamos el menor valor
[DIF, I] = min(abs(OCC - ARR), [], 1);

%Debido a que ambas funciones son crecientes, el valor que más se
%acerca al limite (factor de uso) se encuentra en una posicion
%mayor del eje horizontal
[VAL, POS] = max(I);

%Al tener la posicion en los ejes x e y, ya podemos evaluar las
%funciones OCC y ARR con respecto a estos puntos, hallando asi sus
%valores ideales
OCC_ideal = OCC(VAL, POS) * max_OCC
ARR_ideal = ARR(VAL, POS) * max_ARR
w_ideal = 1 / (rho_w(1, POS) * mu)
N_max = avg_sessions_per_dest * (1 + s / rho_a(1, VAL))
```