

```

function colaDD1_V1(lambda,mu,tiempoSimulacion)
% Simula una cola D/D/1/inf, pero sólo reproduce la dinámica
reloj = 0; % reloj de la simulación
proximaLlegada = 0; % Tiempo de la siguiente llegada
proximaSalida = inf; % Tiempo de la siguiente salida
paquetesSistema = 0; % Número de paquetes en el sistema [N(t)]
while reloj<tiempoSimulacion
    if proximaLlegada<proximaSalida % Procesa una llegada
        reloj = proximaLlegada; % Actualiza el reloj de simulación
        proximaLlegada = reloj + 1/lambda; % Programa la próxima llegada
        paquetesSistema = paquetesSistema+1;
        if paquetesSistema == 1 % De hecho, lo puede empezar a transmitir inmediatamente
            proximaSalida = reloj + 1/mu; % Programa la próxima salida
        end
    else % Procesa una salida
        reloj = proximaSalida; % Actualiza el reloj de simulación
        paquetesSistema = paquetesSistema - 1;
        proximaSalida = inf;
        if paquetesSistema > 0
            proximaSalida = reloj + 1/mu;
        end
    end
end
end

```

```

function traza = colaDD1_V2(lambda,mu,tiempoSimulacion)
% Simula una cola D/D/1/inf y dibuja N(t) vs t (el número de paquetes en el sistema)
    reloj = 0; % reloj de la simulación
    proximaLlegada = 0; % Tiempo de la siguiente llegada
    proximaSalida = inf; % Tiempo de la siguiente salida
    paquetesSistema = 0; % Número de paquetes en el sistema [N(t)]
    traza = zeros(floor(3*tiempoSimulacion*lambda),2); % El evento i ocurrió en el instante
    numeroEvento = 0; % traza(i,1) y dejó traza(i,2) paquetes en el sistema
    while reloj < tiempoSimulacion
        numeroEvento = numeroEvento+1; % Un evento más
        traza(numeroEvento,:) = [reloj paquetesSistema]; % Registra (tiempo,estado) en cada evento
        if proximaLlegada < proximaSalida % Procesa una llegada
            reloj = proximaLlegada; % Actualiza el reloj de simulación
            proximaLlegada = reloj + 1/lambda; % Programa la próxima llegada
            paquetesSistema = paquetesSistema+1;
            if paquetesSistema == 1 % De hecho, lo puede empezar a transmitir inmediatamente
                proximaSalida = reloj + 1/mu; % Programa la próxima salida
            end
        else % Procesa una salida
            reloj = proximaSalida; % Actualiza el reloj de simulación
            paquetesSistema = paquetesSistema - 1;
            proximaSalida = inf;
            if paquetesSistema > 0
                proximaSalida = reloj + 1/mu;
            end
        end
    end
    traza = traza(1:numeroEvento,:);
    stairs(traza(:,1),traza(:,2))
    axis([0 tiempoSimulacion -0.2 0.2+max(traza(:,2))])

```

```

function traza = colaDD1k(lambda,mu,cupo,tiempoSimulacion)
% Simula una cola D/D/1/cupo y dibuja N(t) vs t
    reloj = 0; % reloj de la simulación
    proximaLlegada = 0; % Tiempo de la siguiente llegada
    proximaSalida = inf; % Tiempo de la siguiente salida
    paquetesSistema = 0; % Número de paquetes en el sistema [N(t)]
    traza = zeros(floor(3*tiempoSimulacion*lambda),2); % El evento i ocurrió en el instante
    numeroEvento = 0; % traza(i,1) y dejó traza(i,2) paquetes en el sistema
    while reloj<tiempoSimulacion
        numeroEvento = numeroEvento+1; % Un evento más
        traza(numeroEvento,:) = [reloj paquetesSistema]; % Registra (tiempo,estado) en cada evento
        if proximaLlegada<proximaSalida % Procesa una llegada
            reloj = proximaLlegada; % Actualiza el reloj de simulación
            proximaLlegada = reloj + 1/lambda; % Programa la próxima llegada
            if paquetesSistema < cupo
                paquetesSistema = paquetesSistema+1;
                if paquetesSistema == 1 % De hecho, lo puede empezar a transmitir inmediatamente
                    proximaSalida = reloj + 1/mu; % Programa la próxima salida
                end
            end
        else % Procesa una salida
            reloj = proximaSalida; % Actualiza el reloj de simulación
            paquetesSistema = paquetesSistema - 1;
            proximaSalida = inf;
            if paquetesSistema > 0
                proximaSalida = reloj + 1/mu;
            end
        end
    end
    traza = traza(1:numeroEvento,:);
    stairs(traza(:,1),traza(:,2))
    axis([0 tiempoSimulacion -0.2 0.2+max(traza(:,2))])

```

```

function traza = colaDDnk_V1(lambda,mu,servidores,cupo,tiempoSimulacion)
% Simula una cola D/D/servidores/cupo y dibuja N(t) vs t
reloj = 0; % reloj de la simulación
proximaLlegada = 0; % Tiempo de la siguiente llegada
proximaSalida = inf*ones(servidores,1); % Tiempo de la siguiente salida en cada servidor
paquetesSistema = 0; % Número de paquetes en el sistema [N(t)]
traza = zeros(floor(3*tiempoSimulacion*lambda),2); % El evento i ocurrió en el instante
numeroEvento = 0; % traza(i,1) y dejó traza(i,2) paquetes en el sistema
while reloj<tiempoSimulacion % Inicia el ciclo de simulación
    numeroEvento = numeroEvento+1; % Un evento más
    traza(numeroEvento,:) = [reloj paquetesSistema]; % Registra (tiempo,estado) en cada evento
    if proximaLlegada<min(proximaSalida) % Procesa una llegada
        reloj = proximaLlegada; % Actualiza el reloj de simulación
        proximaLlegada = reloj + 1/lambda; % Programa la próxima llegada
        if paquetesSistema<cupo % El sistema puede aceptar este paquete
            paquetesSistema = paquetesSistema+1;
            if paquetesSistema <= servidores % De hecho, lo puede empezar a transmitir inmediatamente
                ss = find(proximaSalida==inf); % Busca el primer servidor desocupado
                ss = ss(1);
                proximaSalida(ss) = reloj + 1/mu; % Programa la próxima salida de este servidor
            end
        end
    else % Procesa una salida
        [reloj,k] = min(proximaSalida); % Actualiza el reloj de simulación e indica el servidor que termina el servicio
        paquetesSistema = paquetesSistema - 1;
        proximaSalida(k) = inf; % Este servidor queda desocupado
        if paquetesSistema >= servidores
            proximaSalida(k) = reloj + 1/mu; % Programa la proxima salida de este servidor
        end
    end
end
traza = traza(1:numeroEvento,:); % Fin de la simulación: Grafica la ocupación del sistema
stairs(traza(:,1),traza(:,2)) % (número de paquetes en el sistema como función del tiempo)
axis([0 tiempoSimulacion -0.2 0.2+max(traza(:,2))])

```

```

function [traza,EN] = colaDDnk_V2(lambda,mu,servidores,cupo,tiempoSimulacion)
% Simula una cola D/D/servidores/cupo, dibuja N(t) vs t y calcula E[N]
reloj = 0; % reloj de la simulación
proximaLlegada = 0; % Tiempo de la siguiente llegada
proximaSalida = inf*ones(servidores,1); % Tiempo de la siguiente salida en cada servidor
paquetesSistema = 0; % Número de paquetes en el sistema [N(t)]
traza = zeros(floor(3*tiempoSimulacion*lambda),2); % El evento i ocurrió en el instante
numeroEvento = 0; % traza(i,1) y dejó traza(i,2) paquetes en el sistema
EN = 0; % Area bajo la curva N(t) vs t
while reloj<tiempoSimulacion % Inicia el ciclo de simulación
    tue = reloj; % tiempo del último evento
    numeroEvento = numeroEvento+1; % Un evento más
    traza(numeroEvento,:) = [reloj paquetesSistema]; % Registra (tiempo,estado) en cada evento
    if proximaLlegada<min(proximaSalida) % Procesa una llegada
        reloj = proximaLlegada; % Actualiza el reloj de simulación
        EN = EN + paquetesSistema*(reloj - tue); % Actualiza el área bajo la curva N(t)
        proximaLlegada = reloj + 1/lambda; % Programa la próxima llegada
        if paquetesSistema<cupo % El sistema puede aceptar este paquete
            paquetesSistema = paquetesSistema+1;
            if paquetesSistema <= servidores % De hecho, lo puede empezar a transmitir inmediatamente
                ss = find(proximaSalida==inf); % Busca el primer servidor desocupado
                ss = ss(1);
                proximaSalida(ss) = reloj + 1/mu; % Programa la próxima salida de este servidor
            end
        end
    else % Procesa una salida
        [reloj,k] = min(proximaSalida); % Actualiza el reloj de simulación e indica el servidor que termina el servicio
        EN = EN + paquetesSistema*(reloj - tue); % Actualiza el área bajo la curva N(t)
        paquetesSistema = paquetesSistema - 1;
        proximaSalida(k) = inf; % Este servidor queda desocupado
        if paquetesSistema >= servidores
            proximaSalida(k) = reloj + 1/mu; % Programa la proxima salida de este servidor
        end
    end
end
EN = EN/reloj % Número promedio de paquetes en el sistema
traza = traza(1:numeroEvento,:); % Grafica la ocupación del sistema
stairs(traza(:,1),traza(:,2)) % (número de paquetes en el sistema como función del tiempo)
axis([0 tiempoSimulacion -0.2 0.2+max(traza(:,2))])

```

```

function [traza,EN,PB] = colaDDnk_V3(lambda,mu,servidores,cupo,tiempoSimulacion)
% Simula una cola D/D/servidores/cupo, dibuja N(t) vs t, calcula E[N] y calcula PB
    reloj = 0; % reloj de la simulación
    proximaLlegada = 0; % Tiempo de la siguiente llegada
    proximaSalida = inf*ones(servidores,1); % Tiempo de la siguiente salida en cada servidor
    paquetesSistema = 0; % Número de paquetes en el sistema [N(t)]
    traza = zeros(floor(3*tiempoSimulacion*lambda),2); % El evento i ocurrió en el instante
    numeroEvento = 0; % traza(i,1) y dejó traza(i,2) paquetes en el sistema
    EN = 0; % Area bajo la curva N(t) vs t
    NA = 0; NB = 0; % Paquetes que llegan, paquetes rechazados
    while reloj<tiempoSimulacion % Inicia el ciclo de simulación
        tue = reloj; % tiempo del último evento
        numeroEvento = numeroEvento+1; % Un evento más
        traza(numeroEvento,:) = [reloj paquetesSistema]; % Registra (tiempo,estado) en cada evento
        if proximaLlegada<min(proximaSalida) % Procesa una llegada
            reloj = proximaLlegada; % Actualiza el reloj de simulación
            EN = EN + paquetesSistema*(reloj - tue); % Actualiza el área bajo la curva N(t)
            proximaLlegada = reloj + 1/lambda; % Programa la próxima llegada
            NA = NA+1; % Llega un paquete más
            if paquetesSistema<cupo % El sistema puede aceptar este paquete
                paquetesSistema = paquetesSistema+1;
                if paquetesSistema <= servidores % De hecho, lo puede empezar a transmitir inmediatamente
                    ss = find(proximaSalida==inf); % Busca el primer servidor desocupado
                    ss = ss(1);
                    proximaSalida(ss) = reloj + 1/mu; % Programa la próxima salida de este servidor
                end
            else
                NB = NB+1; % Otro paquete bloqueado
            end
        else % Procesa una salida
            [reloj,k] = min(proximaSalida); % Actualiza el reloj de simulación e indica el servidor que termina el servicio
            EN = EN + paquetesSistema*(reloj - tue); % Actualiza el área bajo la curva N(t)
            paquetesSistema = paquetesSistema - 1;
            proximaSalida(k) = inf; % Este servidor queda desocupado
            if paquetesSistema >= servidores
                proximaSalida(k) = reloj + 1/mu; % Programa la proxima salida de este servidor
            end
        end
    end
    EN = EN/reloj % Número promedio de paquetes en el sistema
    PB = NB/NA % Fracción de paquetes rechazados
    traza = traza(1:numeroEvento,:); % Grafica la ocupación del sistema
    stairs(traza(:,1),traza(:,2)) % (número de paquetes en el sistema como función del tiempo)
    axis([0 tiempoSimulacion -0.2 0.2+max(traza(:,2))])

```

```

function [traza, EN, EW, PB] = colaDDnk_V4(lambda,mu,servidores,cupo,tiempoSimulacion)
% D/D/servidores/cupo, dibuja N(t) vs t, calcula E[N], E[W] y PB
reloj = 0; % reloj de la simulación
proximaLlegada = 0; % Tiempo de la siguiente llegada
proximaSalida = inf*ones(servidores,1); % Tiempo de la siguiente salida en cada servidor
paquetesSistema = 0; % Número de paquetes en el sistema [N(t)]
paquetesCola = 0; % Número de paquetes en la cola [Q(t)]
traza = zeros(floor(3*tiempoSimulacion*lambda),2); % El evento i ocurrió en el instante
numeroEvento = 0; % traza(i,1) y dejó traza(i,2) paquetes en el sistema
EN = 0; % Promedio de paquetes en el sistema (durante la simulación, área debajo de la curva)
NB = 0; % Número de paquetes bloqueados
NA = 0; % Número de paquetes que han llegado
ND = 0; % Número de paquetes que han salido
EW = 0; % Suma de los retardos de los paquetes que han salido de la cola
tamanoCola = cupo - servidores; % Calcula el tamaño de la cola
if tamanoCola==inf % La cola será un arreglo circular donde se almacena
    tamanoCola=ceil(lambda*tiempoSimulacion); % el instante de llegada de cada paquete
end
cola = zeros(tamanoCola,1); % Se usa un puntero ULTIMOCOLA a donde se almacenará el próximo paquete que llegue
primeroCola=1; % y un puntero PRIMEROCOLA de donde se tomará el próximo paquete que salga (FIFO)
ultimoCola=1; % Estos punteros se incrementarán circularmente
while reloj<tiempoSimulacion % Inicia el ciclo de simulación
    tue = reloj; % tiempo del último evento
    numeroEvento = numeroEvento+1; % Un evento más
    traza(numeroEvento,:) = [reloj paquetesSistema]; % Registra (tiempo,estado) en cada evento
    if proximaLlegada<min(proximaSalida) % Procesa una llegada
        NA = NA + 1; % Una llegada más
        reloj = proximaLlegada; % Actualiza el reloj de simulación
        EN = EN + paquetesSistema*(reloj - tue); % Actualiza el área bajo la curva N(t)
        proximaLlegada = reloj + 1/lambda; % Programa la próxima llegada
        if paquetesSistema<cupo % El sistema puede aceptar este paquete
            paquetesSistema = paquetesSistema+1;
            if paquetesSistema <= servidores % De hecho, lo puede empezar a transmitir ya
                ss = find(proximaSalida==inf ); % Busca el primer servidor desocupado
                ss = ss(1);
                proximaSalida(ss) = reloj + 1/mu; % Programa la próxima salida de este servidor
            else % Este paquete debe esperar en cola
                paquetesCola = paquetesCola + 1; % Lo almacena en la última posición de la cola
                cola(ultimoCola)=reloj;
                ultimoCola = ultimoCola+1; % (y actualiza circularmente dicha posición)
                if ultimoCola>tamanoCola
                    ultimoCola=1;
                end
            end
        end
    else
        NB = NB + 1; % Este paquete es rechazado porque no hay dónde ubicarlo
    end
end
else % Procesa una salida
    ND = ND + 1; % Una salida más
    [reloj,k] = min(proximaSalida); % Actualiza el reloj de simulación e indica el servidor que termina el servicio
    EN = EN + paquetesSistema*(reloj - tue); % Actualiza el área bajo la curva N(t)
    paquetesSistema = paquetesSistema - 1;
end
end

```

```

if paquetesSistema < servidores
    proximaSalida(k) = inf ;      % Este servidor queda desocupado
else
    paquetesCola = paquetesCola - 1; % Este servidor empieza a atender al primer paquete
                                   % de la cola
    proximaSalida(k) = reloj + 1/mu; % Programa la próxima salida de este servidor
    EW = EW + (reloj - cola(primerCola));
    primeroCola = primeroCola+1;      % (y actualiza circularmente dicha posición)
    if primeroCola>tamanoCola
        primeroCola=1;
    end
end
end
end
traza = traza(1:numeroEvento,:); % Fin de la simulación: Grafica la ocupación del sistema
stairs(traza(:,1),traza(:,2))    % (número de paquetes en el sistema como función del tiempo)
EN = EN/reloj                    % Promedio del número de paquetes que permanecieron en el sistema
EW = EW/ND                       % Retardo promedio en la cola
PB = NB/NA                       % Fracción de paquetes que fueron rechazados

```



```

function [traza, EN, EQ, ET, EW, PB, G2] = colaDDnk_V5(lambda,mu,servidores,cupo,tiempoSimulacion)
% D/D/servidores/cupo, dibuja N(t) vs t, calcula E[N], E[Q], E[T], E[W], PB y Gamma
    reloj = 0; % reloj de la simulación
    proximaLlegada = 0; % Tiempo de la siguiente llegada
    proximaSalida = inf*ones(servidores,1); % Tiempo de la siguiente salida en cada servidor
    paquetesSistema = 0; % Número de paquetes en el sistema [N(t)]
    paquetesCola = 0; % Número de paquetes en la cola [Q(t)]
    traza = zeros(floor(3*tiempoSimulacion*lambda),2); % El evento i ocurrió en el instante
    numeroEvento = 0; % traza(i,1) y dejó traza(i,2) paquetes en el sistema
    EN = 0; % Promedio de paquetes en el sistema (durante la simulación, área debajo de la curva)
    NB = 0; % Número de paquetes bloqueados
    NA = 0; % Número de paquetes que han llegado
    ND = 0; % Número de paquetes que han salido
    EQ = 0; % Promedio de paquetes en la cola (durante la simulación, área debajo de la curva)
    EW = 0; % Suma de los retardos de los paquetes que han salido de la cola
    tamañoCola = cupo - servidores; % Calcula el tamaño de la cola
    if tamañoCola==inf % La cola será un arreglo circular donde se almacena
        tamañoCola=ceil(lambda*tiempoSimulacion); % el instante de llegada de cada paquete
    end
    cola = zeros(tamañoCola,1); % Se usa un puntero ULTIMOCOLA donde se almacenará el próximo paquete que llegue
    primeroCola=1; % y un puntero PRIMEROCOLA de donde se tomará el próximo paquete que salga
    ultimoCola=1; % Estos punteros se incrementarán circularmente
    while reloj<tiempoSimulacion % Inicia el ciclo de simulación
        tue = reloj; % tiempo del último evento
        numeroEvento = numeroEvento+1; % Un evento más
        traza(numeroEvento,:) = [reloj paquetesSistema]; % Registra (tiempo,estado) en cada evento
        if proximaLlegada<min(proximaSalida) % Procesa una llegada
            NA = NA + 1; % Una llegada más
            reloj = proximaLlegada; % Actualiza el reloj de simulación
            EN = EN + paquetesSistema*(reloj - tue); % Actualiza el área bajo la curva N(t)
            EQ = EQ + paquetesCola*(reloj - tue); % Actualiza el área bajo la curva de Q(t)
            proximaLlegada = reloj + 1/lambda; % Programa la próxima llegada
            if paquetesSistema<cupo % El sistema puede aceptar este paquete
                paquetesSistema = paquetesSistema+1;
                if paquetesSistema <= servidores % De hecho, lo puede empezar a transmitir ya
                    ss = find(proximaSalida==inf ); % Busca el primer servidor desocupado
                    ss = ss(1);
                    proximaSalida(ss) = reloj + 1/mu; % Programa la próxima salida de este servidor
                else % Este paquete debe esperar en cola
                    paquetesCola = paquetesCola + 1; % Lo almacena en la última posición de la cola
                    cola(ultimoCola)=reloj;
                    ultimoCola = ultimoCola+1; % (y actualiza circularmente dicha posición)
                    if ultimoCola>tamañoCola
                        ultimoCola=1;
                    end
                end
            end
        else
            NB = NB + 1; % Este paquete es rechazado porque no hay dónde ubicarlo
        end
    else % Procesa una salida
        ND = ND + 1; % Una salida más
        [reloj,k] = min(proximaSalida); % Actualiza el reloj de simulación e indica el servidor que termina el servicio
    end
end

```

```

EN = EN + paquetesSistema*(reloj - tue); % Actualiza el área bajo la curva N(t)
EQ = EQ + paquetesCola*(reloj - tue); % Actualiza el área bajo la curva de Q(t)
paquetesSistema = paquetesSistema - 1;
if paquetesSistema < servidores
    proximaSalida(k) = inf ;      % Este servidor queda desocupado
else
    paquetesCola = paquetesCola - 1; % Este servidor empieza a atender al primer paquete
                                   % de la cola
    proximaSalida(k) = reloj + 1/mu; % Programa la próxima salida de este servidor
    EW = EW + (reloj - cola(primerocola));
    primerocola = primerocola+1;      % (y actualiza circularmente dicha posición)
    if primerocola>tamanoCola
        primerocola=1;
    end
end
end
end
traza = traza(1:numeroEvento,:); % Fin de la simulación: Grafica la ocupación del sistema
stairs(traza(:,1),traza(:,2))    % (número de paquetes en el sistema como función del tiempo)
EN = EN/reloj                    % Promedio del número de paquetes que permanecieron en el sistema
EQ = EQ/reloj                    % Promedio del número de paquetes que permanecieron en la cola
EW = EW/ND                       % Retardo promedio en la cola
ET = EW + 1/mu                  % Retardo promedio en el sistema
G1 = (NA - NB)/reloj            % Número de paquetes aceptados por segundo
G2 = ND/reloj                   % Número de paquetes transmitidos por segundo
PB = NB/NA                       % Fracción de paquetes que fueron rechazados

```