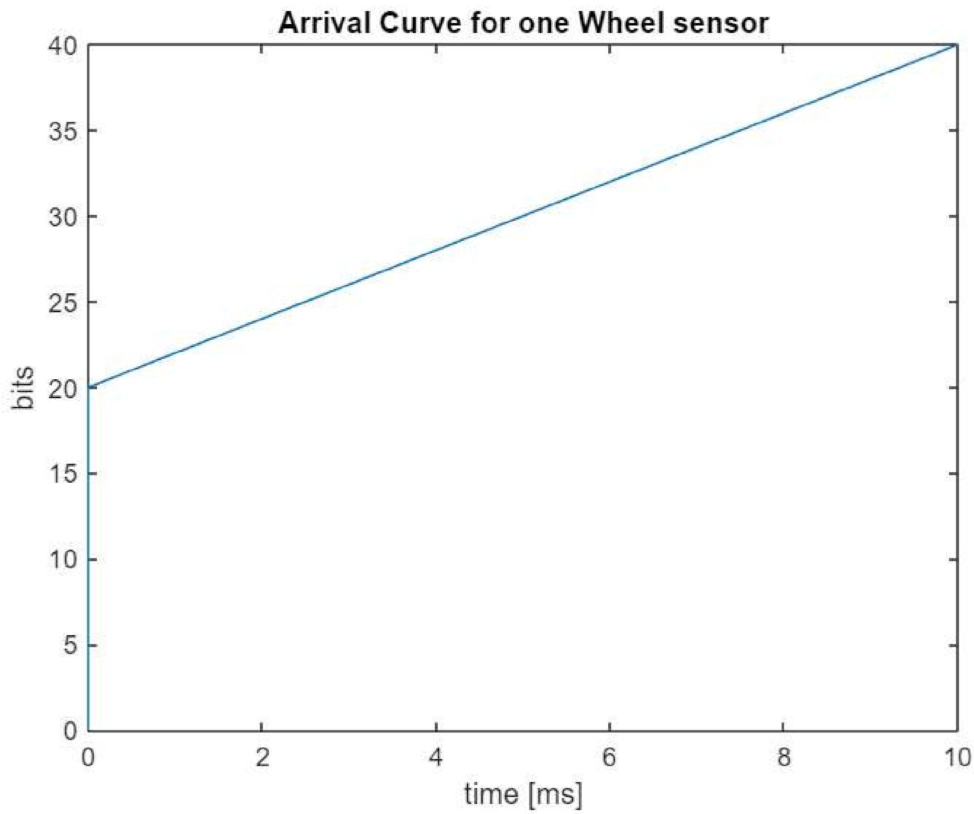
DRTS Assignment 1

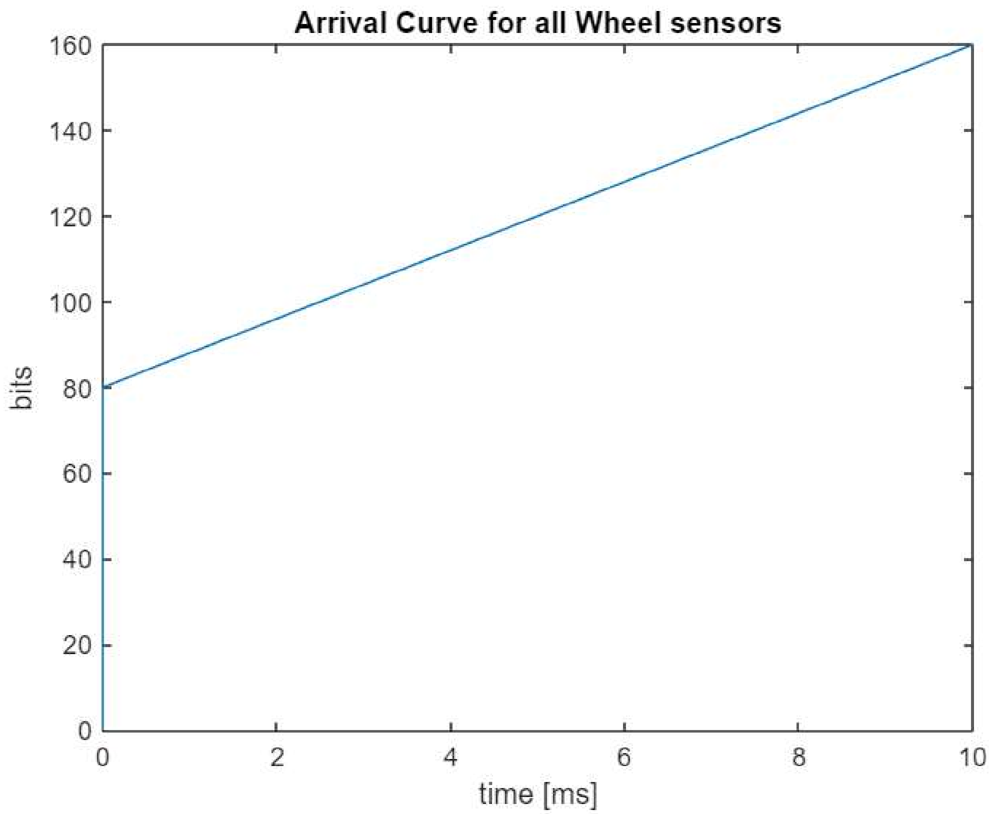
1: Create a DNC/RTC arrival-model of the periodic external traffic sources of the in-car network (choose arrival curve types and determine parameters)

Of the periodic external traffic sources there are four wheel sensors and one ESP sensor. The arrival curve is chosen to be of the affine type. The parameters of the external sources are defined in the script below.

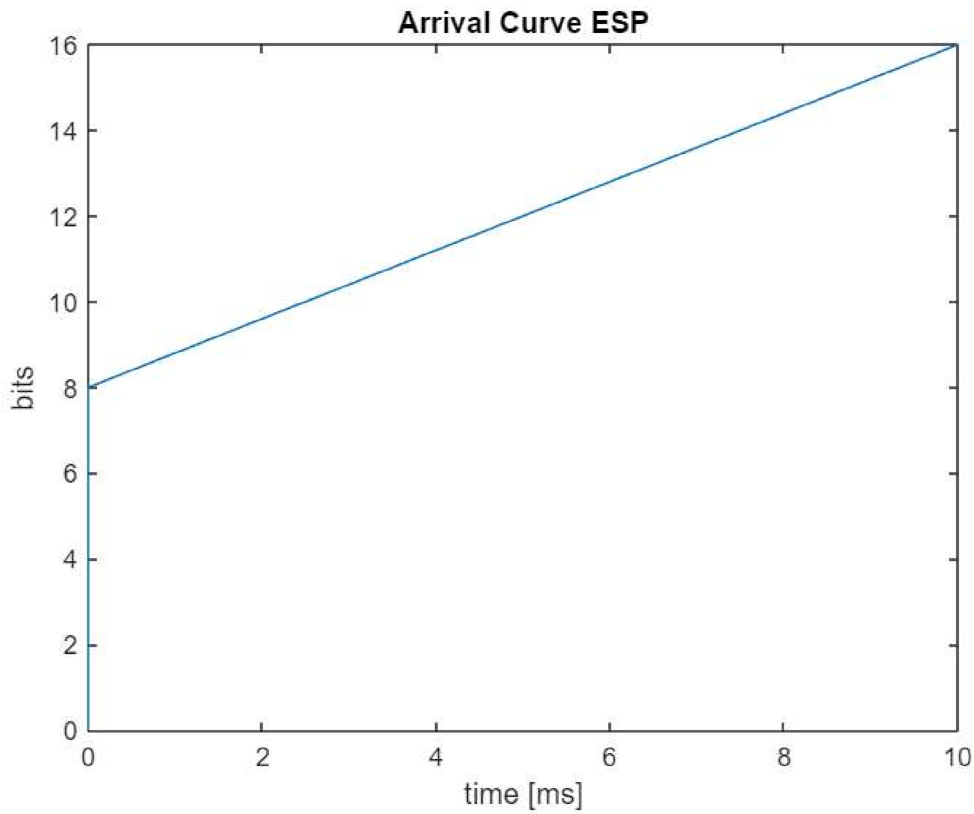
```
clc;clear;close all;

BW = 1*10^3; % [bpms] Serialization speed ms = miliseconds

% Wheel:
N_W = 4;        % Amount of wheel sensors
p_W = 20;       % Amount of bits per sensor per period
O = 0;          % Overhead
T = 10;         % [ms] Transfer period for both wheel sensor and ESP

n = 1;          % Amount of sensor packets in each dataframe (CAN packet)
k_W = p_W*n+O;  % Bits per packet
tau = 0;                % jitter

% Arrival curve - One wheel sensor:
r_W = k_W/(n*T);                        % Rate
b_W = k_W * (tau/T + 1);        % Burst
arr_W = rtccurve([0, b_W, r_W]);
figure
rtcplot(arr_W,T)
title('Arrival Curve for one Wheel sensor')
xlabel('time [ms]')
ylabel('bits')
```



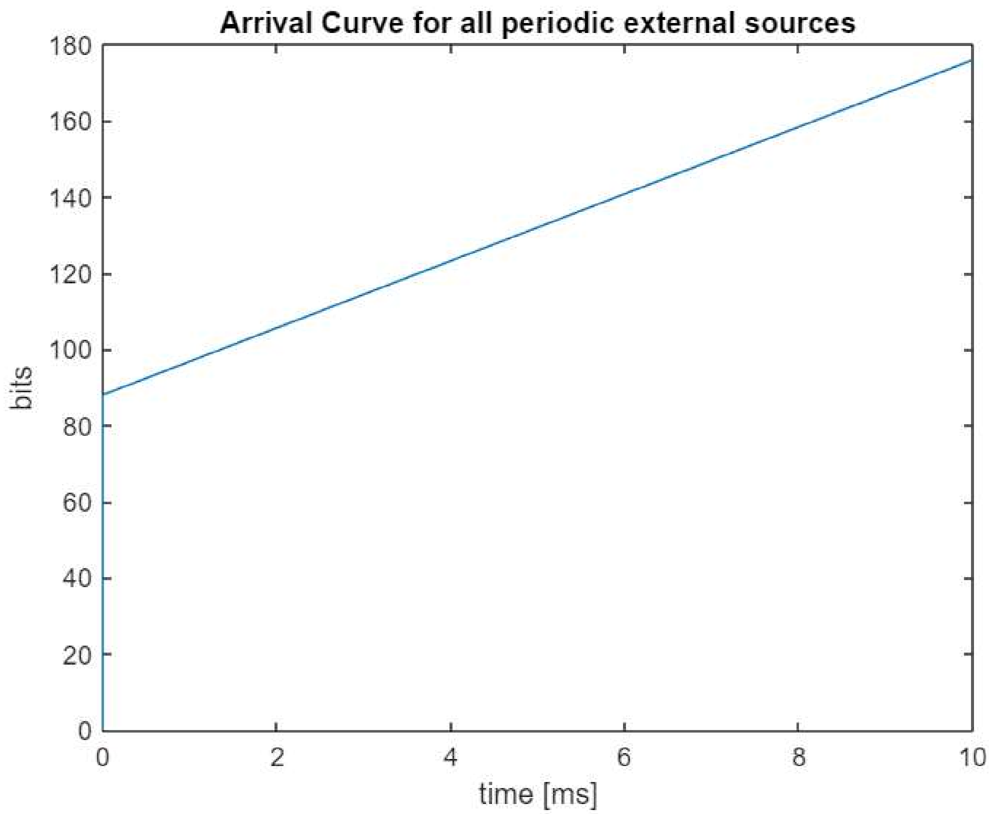

```
% Arrival curve - All wheel sensors
r_W_all = N_W*r_W;
b_W_all = N_W*b_W;
arr_W_all = rtccurve([0, b_W_all, r_W_all]);
figure
rtcplot(arr_W_all,T)
title('Arrival Curve for all Wheel sensors')
xlabel('time [ms]')
ylabel('bits')
```




```
% Arrival curve - ESP sensor
p_ESP = 8;                       % Amount of bits per sensor per period
k_ESP = p_ESP+0;

% Arrival:
r_ESP = k_ESP/(n*T);                     % Rate
b_ESP = k_ESP * (tau/T + 1);     % Burst

arr_ESP = rtccurve([0, b_ESP, r_ESP]);
figure
rtcplot(arr_ESP,T)
title('Arrival Curve ESP')
xlabel('time [ms]')
ylabel('bits')
```




```
% Combination of periodic arrival constraints:
r_per = r_W_all + r_ESP;        % All periodic rate
b_per = b_W_all + b_ESP;        % All periodic burst

figure
arr_per = rtccurve([0, b_per, r_per]);    % Periodic sources arrival curve
rtcplot(arr_per, T)
title('Arrival Curve for all periodic external sources')
xlabel('time [ms]')
ylabel('bits')
```

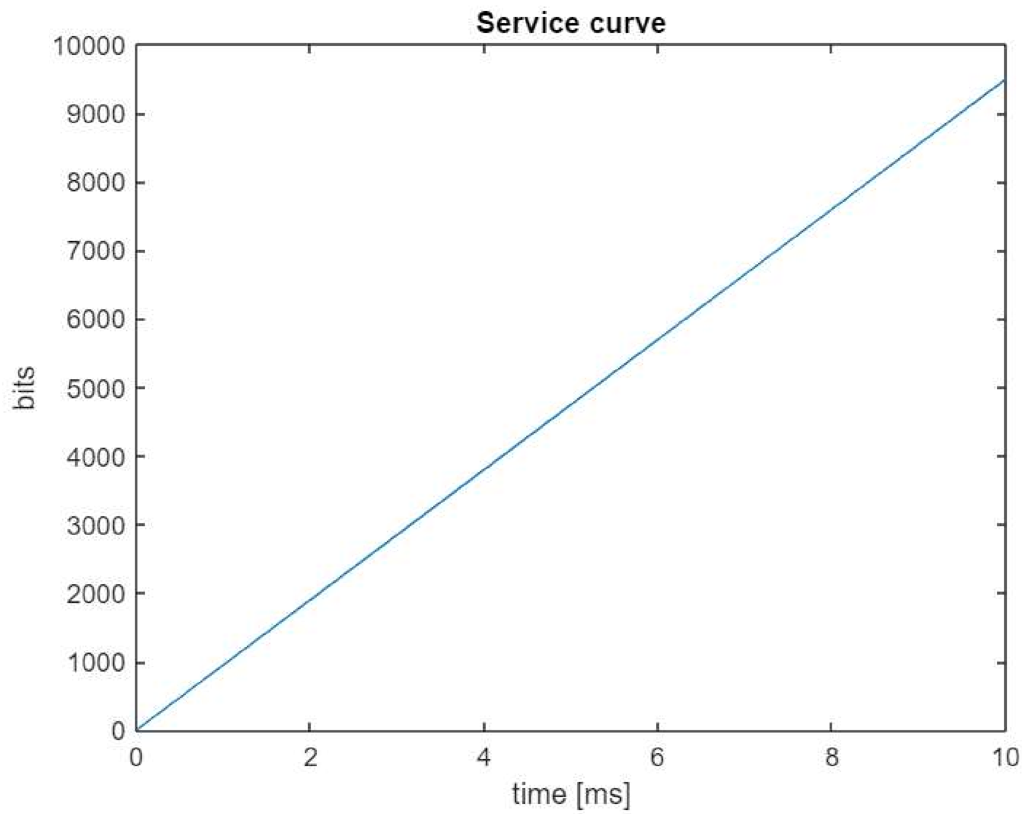


2: Create a DNC/RTC model for your communication network (Identify network elements, select curve types and parameters)

There is one network element i our communication network, nemely the CANbus interface. A service curve is defined.

Service curve rate is set a 5% lower than bandwidth to ensure we're not marginally stable.

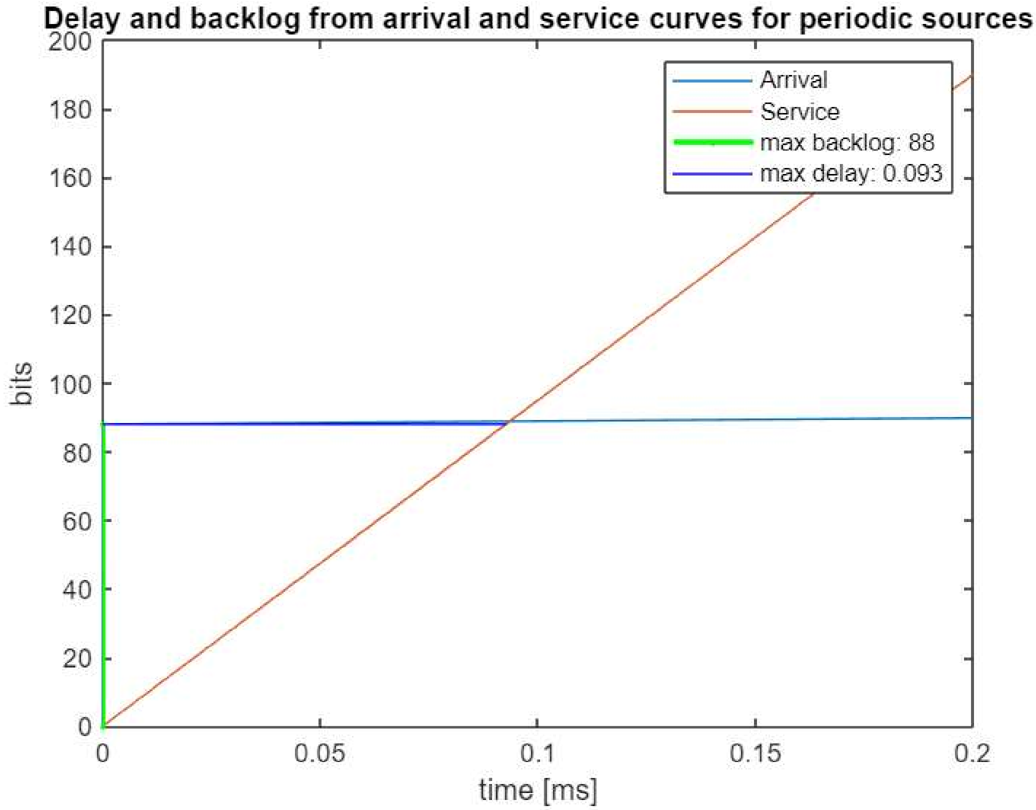
```
% Service curves:
d = (n-1)*T; % Delay - 0 since we are not packing more than one sensor
serv = rtccurve([d, 0, BW*0.95]);
figure
rtcplot(serv,T)
title('Service curve')
xlabel('time [ms]')
ylabel('bits')
```




```
% Backlog and delay of periodic sources:
figure
rtcplot(arr_per, 0.2) % Plot Arrival curve
hold on
rtcplot(serv) % Service curve
rtcplotv(arr_per,serv) % Backlog
rtcploth(arr_per,serv, 'b') % Delay
title('Delay and backlog from arrival and service curves for periodic sources')
xlabel('time [ms]')
ylabel('bits')

delay = rtch(arr_per,serv), backlog = rtcv(arr_per,serv)
```

```
delay = 0.0926
backlog = 88
```

```
str_delay = sprintf('max delay: %0.3f', delay);
str_backlog = sprintf('max backlog: %d', backlog);
legend('Arrival', 'Service', str_backlog, str_delay)
```



3. Guess initial parameters for token bucket filters (for the Poisson traffic sources) and include parameters in the DNC/RTC model.

```
% - Packet size is defined in task as 1400 bits
% - Average period is set to 30 ms (although its a poisson process)

% Rear Camera parameters
T_RC = 40% [ms]
```

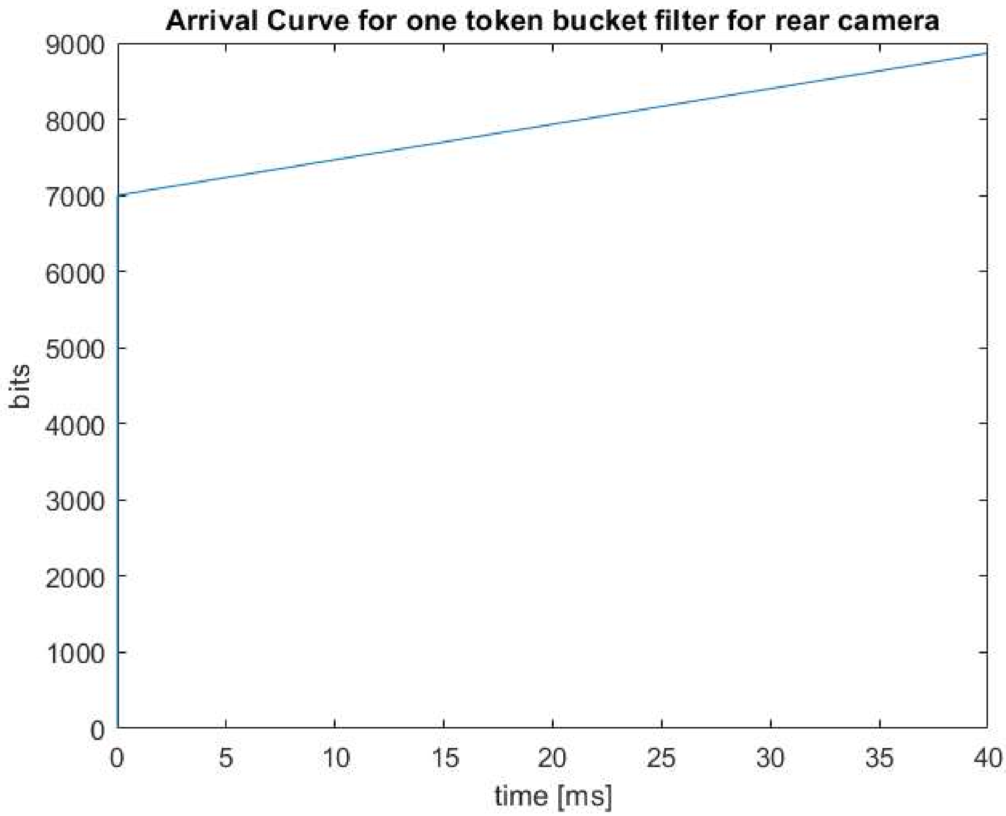
```
T_RC = 40
```

```
p_RC = 1400; % Packet size - Number of bits per packet
% Multimedia parameters
T_MM = 40; % [ms]
p_MM = 1400; % Packet size - Number of bits per packet

% Token bucket filter parameters - initial guesses
M_TB = 5; % Bucket size - Burst parameter (random guess=
T_TB = 30; % [ms] Token replenishment rate - Curve rate (Bit faster than 40ms)
```

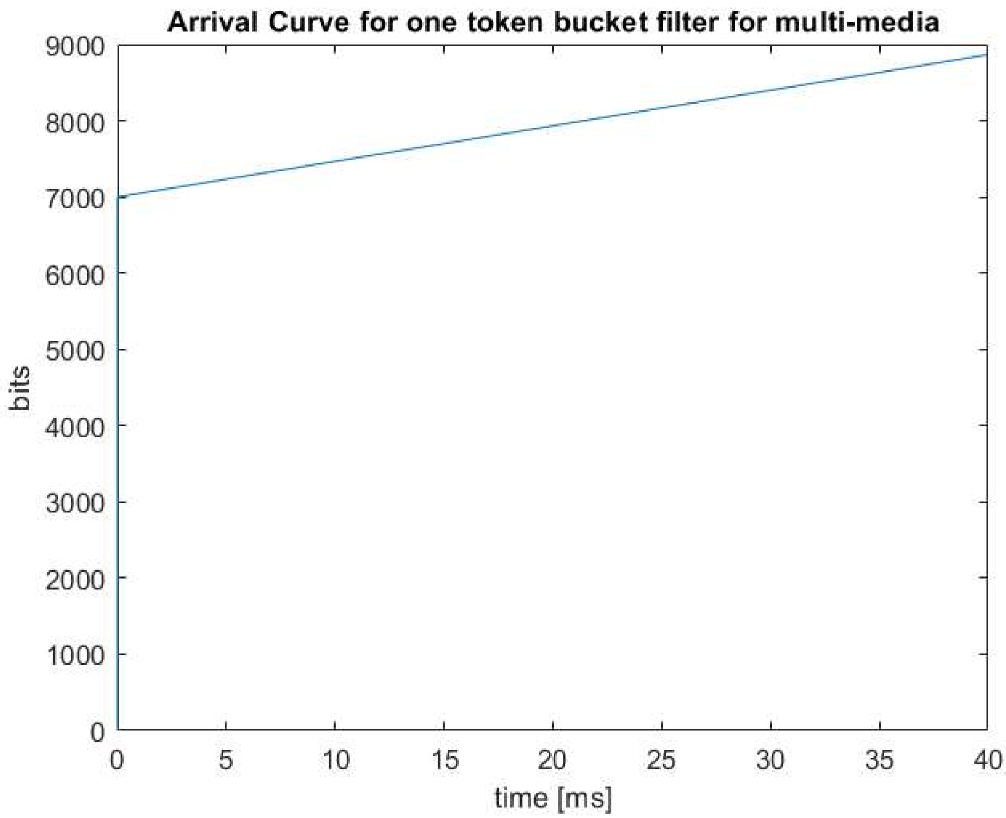
4. Determine arrival curves for outputs of token bucket filters and include in the DNC/RTC model.

```
% Arrival curve Rear Camera
r_RC = 1/T_TB * p_RC;
b_RC= p_RC * M_TB;
arr_RC = rtccurve([0, b_RC, r_RC]);
```

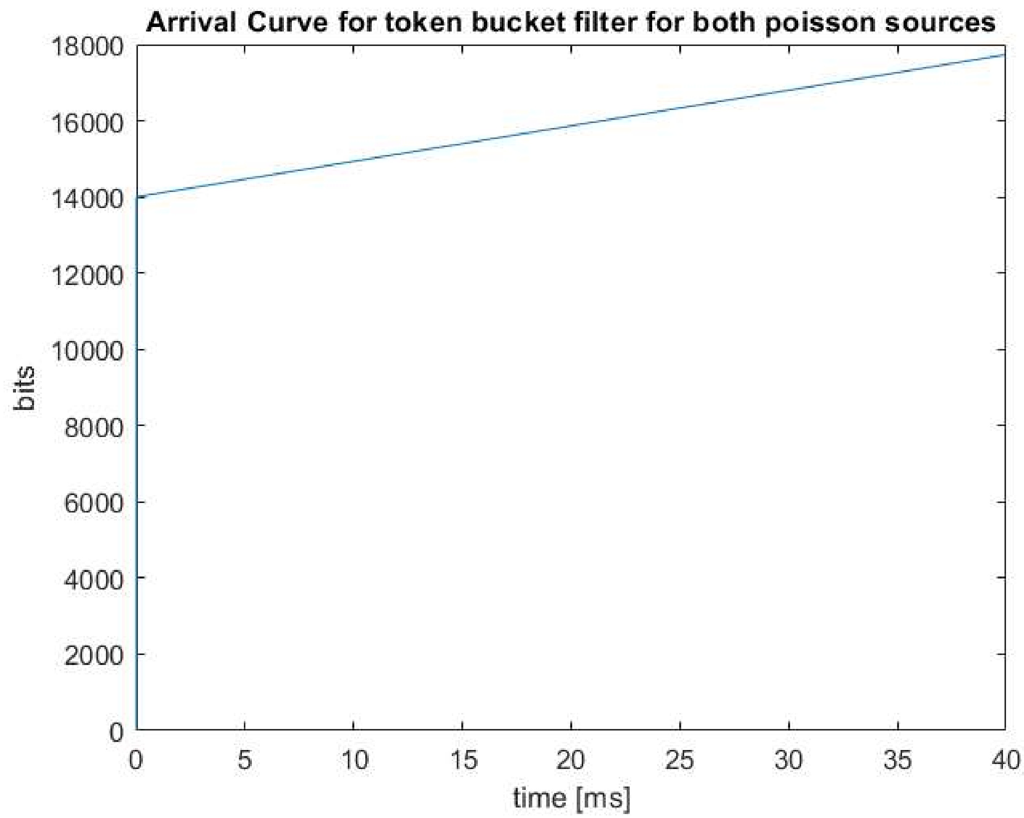
```
figure
rtcplot(arr_RC, T_RC)
title('Arrival Curve for one token bucket filter for rear camera')
xlabel('time [ms]')
ylabel('bits')
```




```
% Arrival curve Multi-media
r_MM = 1/T_TB * p_MM;
b_MM= p_MM * M_TB;
arr_MM = rtccurve([0, b_MM, r_MM]);
figure
rtcplot(arr_MM, T_MM)
title('Arrival Curve for one token bucket filter for multi-media')
xlabel('time [ms]')
ylabel('bits')
```




```
arr_TB_tot = rtcplus(arr_RC, arr_MM);
figure
rtcplot(arr_TB_tot, T_MM)
title('Arrival Curve for token bucket filter for both poisson sources')
xlabel('time [ms]')
ylabel('bits')
```

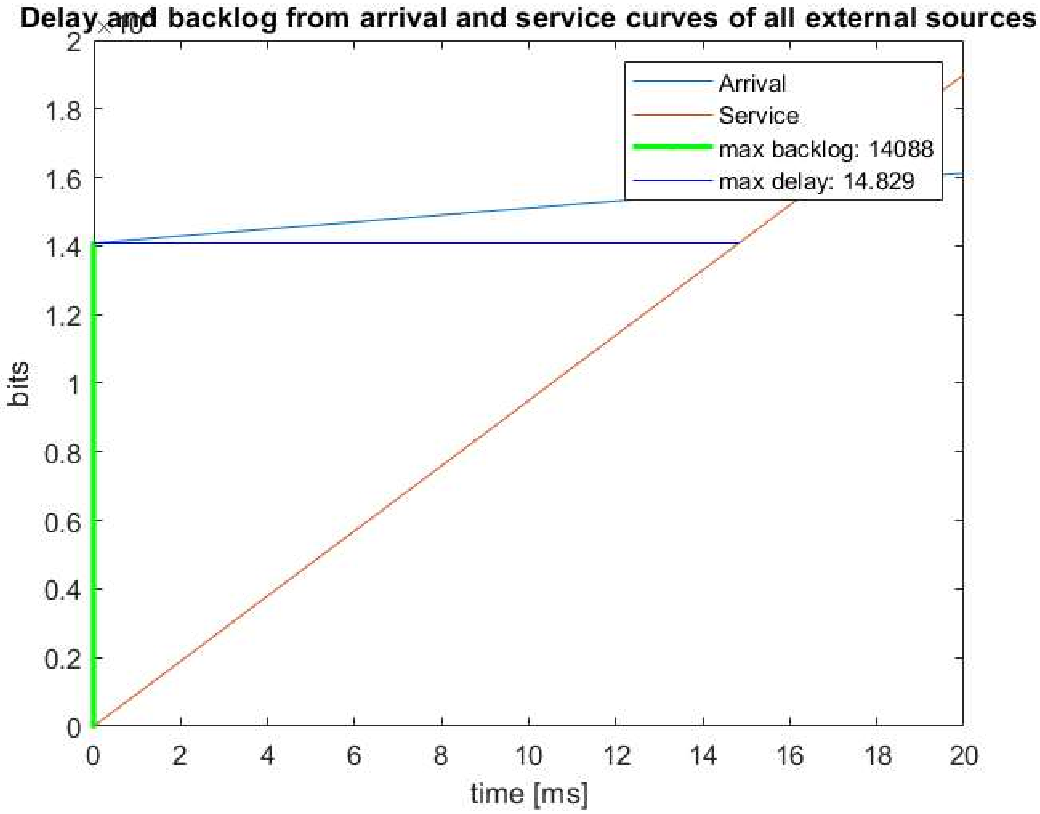


5. Compute max backlogs and max waiting times for all flows for the (non)deterministic part of the network

```
% Arrival curve: All external network sources (periodic and poisson)
arr_all = rtcplus(arr_per, arr_TB_tot);


% Backlog and delay:
figure
rtcplot(arr_all, 20) % Plot Arrival curve
hold on
rtcplot(serv) % Service curve
rtcplotv(arr_all,serv) % Backlog
rtcploth(arr_all,serv, 'b') % Delay
title('Delay and backlog from arrival and service curves of all external sources')
xlabel('time [ms]')
ylabel('bits')

delay = rtch(arr_all,serv), backlog = rtcv(arr_all,serv)
```

```
delay = 14.8295
backlog = 14088
```

```
str_delay = sprintf('max delay: %0.3f', delay);
str_backlog = sprintf('max backlog: %d', backlog);
legend('Arrival', 'Service', str_backlog, str_delay)
```

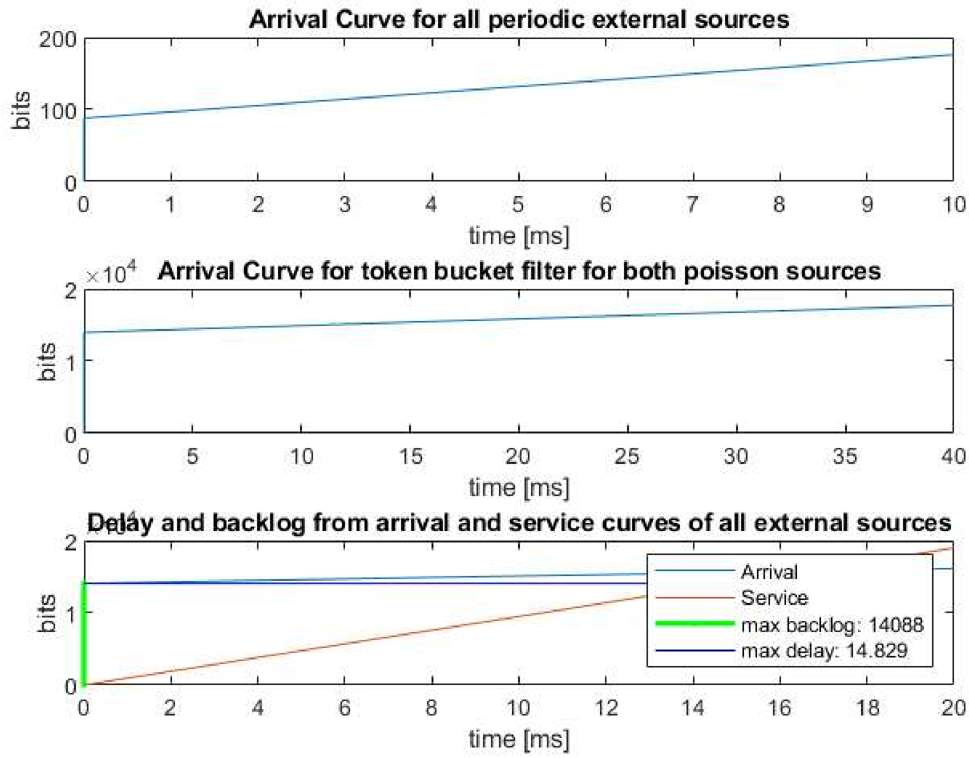



```
figure
subplot(3,1,1)

rtcplot(arr_per, T)
title('Arrival Curve for all periodic external sources')
xlabel('time [ms]')
ylabel('bits')

subplot(3,1,2)
rtcplot(arr_TB_tot, T_MM)
title('Arrival Curve for token bucket filter for both poisson sources')
xlabel('time [ms]')
ylabel('bits')

subplot(3,1,3)
rtcplot(arr_all, 20) % Plot Arrival curve
hold on
rtcplot(serv) % Service curve
rtcplotv(arr_all,serv) % Backlog
rtcploth(arr_all,serv, 'b') % Delay
title('Delay and backlog from arrival and service curves of all external sources')
xlabel('time [ms]')
ylabel('bits')

delay = rtch(arr_all,serv), backlog = rtcv(arr_all,serv)
```

```
delay = 14.8295
backlog = 14088
```

```
str_delay = sprintf('max delay: %0.3f', delay);
```

```
str_backlog = sprintf('max backlog: %d', backlog);
legend('Arrival', 'Service', str_backlog, str_delay)
```



Assignment task 3

Select a suitable queueing model for Token Bucket input queues

To accommodate a finite queue (buffer) length, the M/M/1/L is chosen as a probabilistic model for the Token Bucket Filter.

Compute mean queue lengths, mean waiting times and packet drop probabilities for input queues.

The M/M/1/L queue can be characterised in steady state based on their parameters λ(arrival intensity), μ(service intensity), and a utilisation parameter $\rho = \frac{\lambda}{\mu}$

The token bucket filters implemented in the simulation have the same parameters:

$$\lambda = \frac{1}{40\text{ ms}}$$

$$\mu = \frac{1}{30\text{ ms}}$$

$$\rho = \frac{\lambda}{\mu} = \frac{30}{40} = 0.75$$

```
lambda = 1/(T_RC*1e-3); mu = 1/(T_TB*1e-3);
rho = lambda/mu;
L = 6;  % Queue size
```

The mean queue length can be obtained by the expression

$$\mathrm{E}[Q] = \frac{\rho}{(1-\rho^{L+1})(1-\rho)}$$

```
EQ = rho / ( (1-rho^(L+1)) * (1-rho) )
```

```
EQ = 3.4621
```

The mean waiting time can be obtained by the expression

$E[T] = \frac{1}{\mu}(E[Q]+1)$ % the one is here as the new arrival to show the delay..

```
ET = (1/mu) * (EQ + 1)
```

```
ET = 0.1339
```

The probability of package loss can be obtained by the expression

$$P_{\mathrm{Loss}} = \frac{\rho^L - \rho^{L+1}}{1-\rho^{L+1}}$$

```
P_loss = (rho^L-rho^(L+1)) / (1-rho^(L+1))
```

```
P_loss = 0.0513
```