CARNEGIE MELLON UNIVERSITY
APPLIED STOCHASTIC PROCESSES
(COURSE 18-751)
HOMEWORK 3

Daniel Marew

September 18, 2017

I collaborated with :

Nebyou Yismaw
Daniel Nkemelu
Agatha Niwomugizi

Q.1 Multiplexing

We can compute the mean 95% data rate range by approximating it with the normal distribution. The mean of a bernoulli process is given by

$$\mu = Np$$

where N is the number of trails and p is the probability of success. The 95 % number of users range is given by

$$Np \pm 1.96 * \sqrt{Np(1-p)} \tag{1}$$

(a) Base design $N_b = 100$ $R_b = 1Mbps$ and $p_b = 0.3$

$$\mu_b = Np = 100 * 0.3 = 30$$

using eqn 1 number of users range

$$N_b p_b \pm 1.96 * \sqrt{N_b p_b(1-p_b)}$$

from these we get that 95% of the time the number of users range between 21.01 to 38.98. The data rate range is just the link data rate divided by the number of users.hence The lower limit of the data rate the user will see 95% of the time = $\frac{1Mbps}{38.98} = 25.6kbps \approx 26kbps$
The upper limit of the data rate the user will see 95% of the time = $\frac{1Mbps}{21.01} = 47.57kbps \approx 48kbps$

Ans. lower limit = 26kbps upper limit =48kbps , range = 48kbps-26kbps=22kbps

(b) $p_1 \to 0$, $\mu_1 = \mu_b$

$$\mu_1 = \mu_b = N_1 p_1 = 100 * 0.3 = 30$$

$$N_1 p_1 \pm 1.96 * \sqrt{N_1 p_1(1-p_1)} = \mu_1 \pm 1.96\sqrt{\mu_1}$$

since $\mu_1 = N_1 p_1$
when we substitute all the values we will get
from these we get that 95% of the time the number of users range between 19.264 to 40.73.
The upper limit of the data rate the user will see 95% of the time = $\frac{1Mbps}{19.264} = 51.9kbps \approx 52kbps$
The lower limit of the data rate the user will see 95% of the time = $\frac{1Mbps}{40.73} = 24.55kbps \approx 25kbps$

Ans. lower limit = 25kbps upper limit =52kbps , range = 52kbps-25kbps=27kbps

(c) $p_1 = p_b = 0.3$, $R_2 = R_b c$, $N_2 = 100c$, $c = 10$

$$\mu_1 = \mu_b = N_1 p_1 = 100 * 0.3 = 30$$

95% range pf users

$$N_2 p_b \pm 1.96 * \sqrt{N_2 p_b (1 - p_b)} = 100c * 0.3 \pm 1.96 * \sqrt{100c * 0.3(1 - 0.3)}$$

$$= 30c \pm 1.96 * \sqrt{21c}$$

when c=10

$$= 300 \pm 1.96 * \sqrt{210} = 300 \pm 28.4$$

from these we get that 95% of the time the number of users range between 271.59 to 328.40.
hence

The upper limit of the data rate the user will see 95% of the time = $\frac{1Mbps*10}{271.59} = 36.8kbps \approx 37kbps$
The lower limit of the data rate the user will see 95% of the time = $\frac{1Mbps*10}{321.40} = 31.11kbps \approx 31kbps$

Ans. lower limit = 31kbps upper limit =37kbps , range = 37kbps-31kbps=6kbps

(d) $p_1 = p_b = 0.3$, $R_2 = R_b c$, $N_2 = 100c$,$c \to \infty$

95% range of datarates

$$\frac{R_b c}{100 c p_b \pm 1.96 * \sqrt{100 c p_b (1 - p_b)}}$$

if we divid both numerator and denomerator with c

$$\frac{R_b}{100 p_b \pm \frac{1.96}{c} * \sqrt{100 c p_b (1 - p_b)}}$$

as $c \to \infty$ we are left with

$$\frac{R_b}{100 p_b}$$

$$= \frac{1000kbps}{100 * 0.3} = 33.33kpbs$$

The upper and lower limits would be the same 33.33kpbs
Ans. upper limit = lower limit = 33.3kpbs range = 0

	Lower limit(kbps)	Upper limit(kpbps)
Base Design	26	48
New Design 1	25	52
New Design 2	31	37
New Design 2'	33.3	33.3

Table 1: data rate range for different designs

(e) Compare the designs

From Table 1 we can clearly observe that the best design is the last one (d).Since our confidence interval is almost zero we can say with high degree of certainity what the user will be getting 95% of the time(most of the time).But the problem is, we cannot infinitly add bigger capacity links thus (d) is not a realistic solution. So I would recommend the design that is the closest to (d) but realstic. In this case, it is New Design 2(c).

Q.2 Generalized CLT

(a) Show GCLT reduces to the usual i.i.d version of CLT

$$Yn = (X_1 + X_2 + \cdots + X_n - m_n)/Sn$$

where X_k is an independent random variable with $E[X_k] = \mu_k$ and $Var[X_k] = \sigma_k^2$, $m_n = (\mu_1 + \mu_2 + \cdots + \mu_n)/s_n$ and $s_n^2 = \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2$

since i.i.ds have identical distribution the have the same mean and variance
i.e $\mu_1 = \mu_2 = \cdots = \mu_n = \mu$ and $\sigma_1^2 = \sigma_2^2 = \cdots = \sigma_n^2 = \sigma^2$

$$m_n = (\mu_1 + \mu_2 + \cdots + \mu_n) = n\mu$$
$$s_n^2 = (\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2) = n\sigma^2$$
$$Yn = \frac{(X_1 + X_2 + \cdots + X_n - n\mu)}{\sqrt{n\sigma^2}}$$
$$Yn = \frac{(X_1 + X_2 + \cdots + X_n - n\mu)}{\sigma\sqrt{n}}$$

hence , GCLT reduces to CLT when $\{X_k\}$ are i.i.ds.

(b) X_k uniformly distributed from 1 to 5

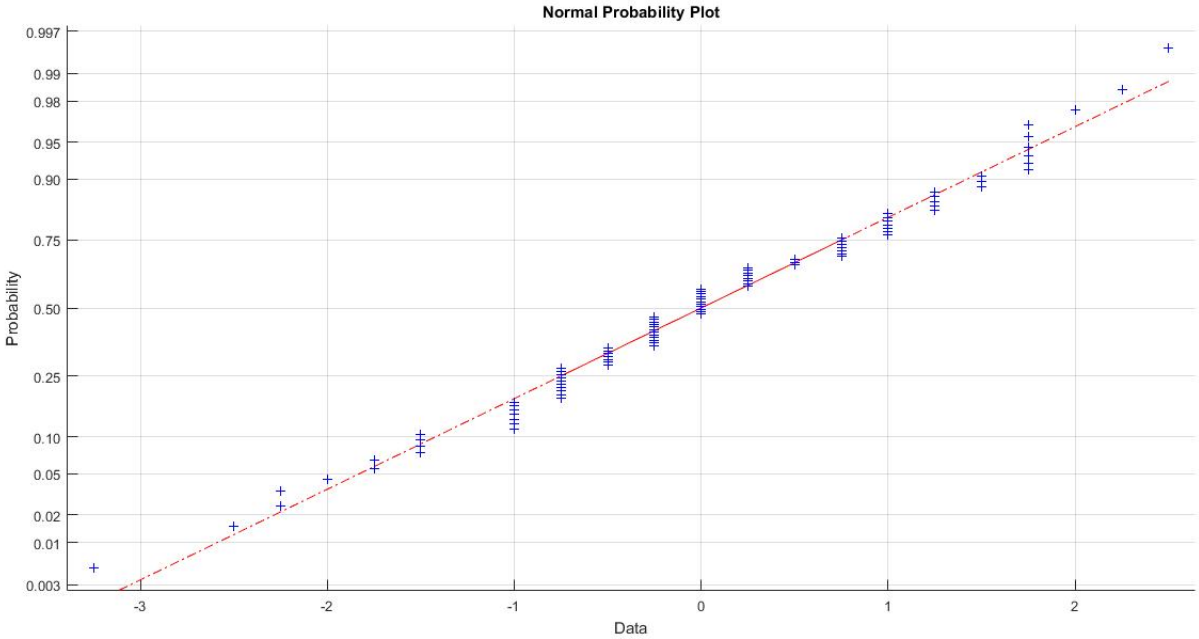

Figure 1: Normalized histogram Vs pdf of standard normal distribution for (b)

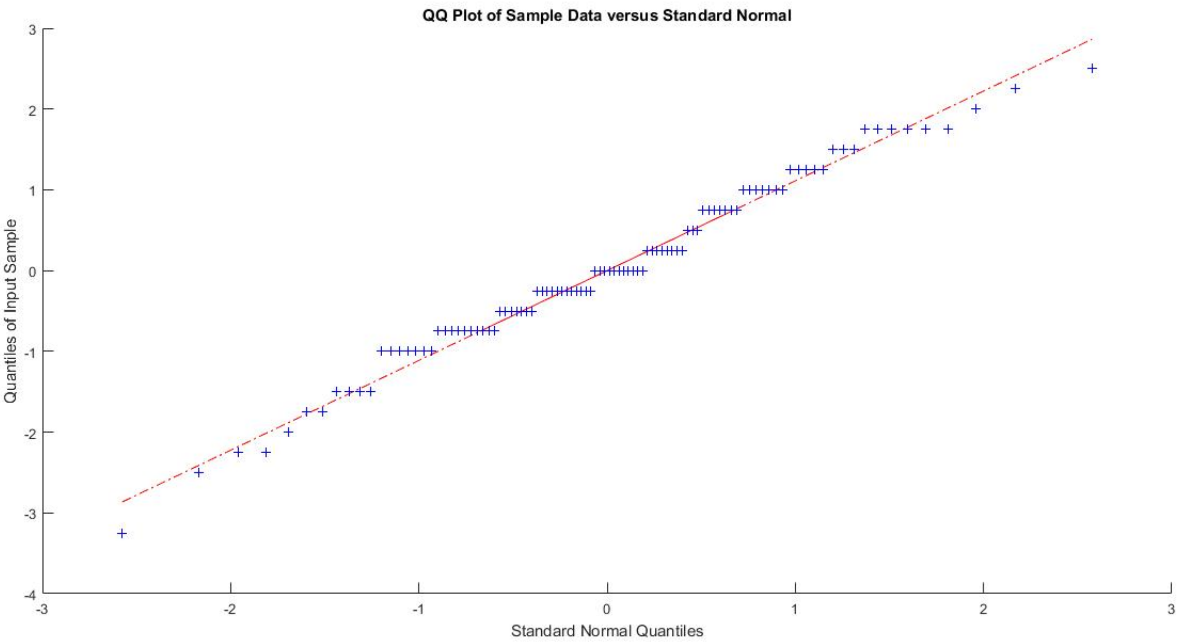

Figure 2: q-q plot for (b)

(c) X_k uniformly distributed from 1 to k

GCLT applies to (c) if for any $\epsilon > 0$ there exists a large enough n such that $\sigma_k < \epsilon S_n$.

$$\sigma_k < \epsilon S_n \implies \sigma_n^2 < \epsilon S_n^2$$

the variance for a uniform distribution is given by

$$\sigma_k^2 = \frac{(k-1)^2}{12}$$

$$S_n^2 = \sigma_1^2 + \sigma_1^2 \cdots + \sigma_n^2 = \sum_{k=1}^{n} \sigma_k^2 = \sum_{k=1}^{n} \frac{(k-1)^2}{12}$$

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(n+2)}{6}$$

$$\implies S_n^2 = \frac{(n-1)(n)(n+1)}{72}$$

$\sigma_k < \epsilon S_n$ for any $\epsilon > 0 \iff \lim_{n\to\infty} \frac{\sigma_k^2}{S_n^2} = 0$
$max\{\sigma_k\} = \sigma_n$

$$\frac{\sigma_n^2}{S_n^2} = \frac{\frac{n-1}{12}}{\frac{(n+1)(n)(n-1)}{72}} = 6 * \frac{n-1}{n(n+1)}$$
$$= \frac{6 * (1 - \frac{1}{n})}{(n+1)}$$

hence

$$\lim_{n\to\infty} \frac{\sigma_n^2}{S_n^2} = 0$$

$\implies$ GCLT applies to (c).

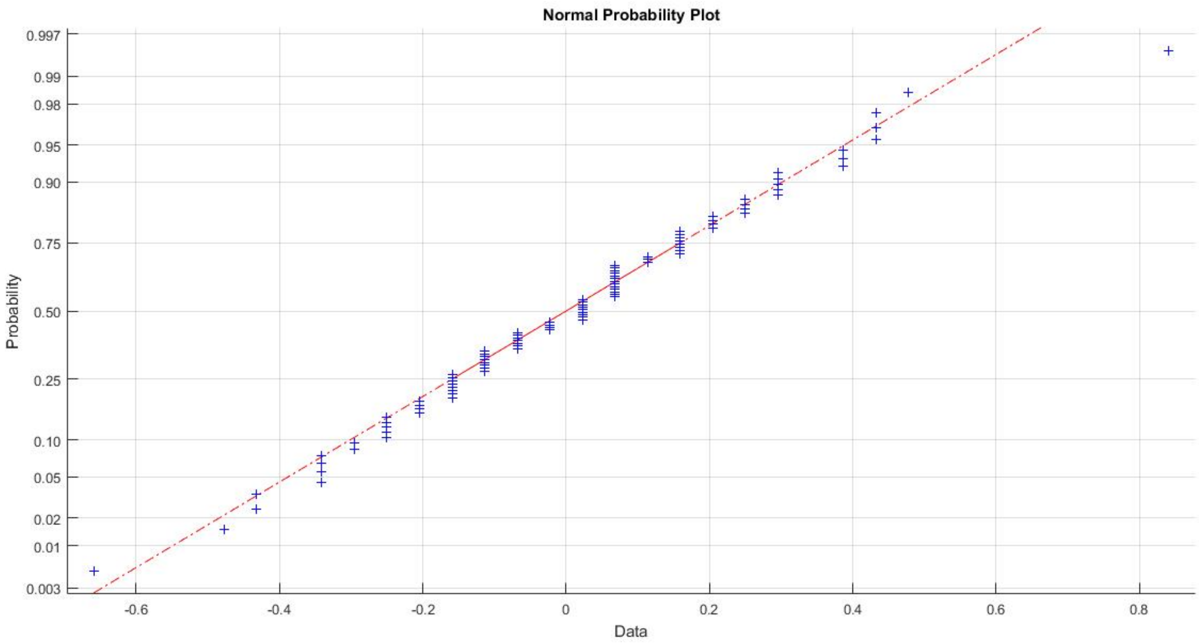

Figure 3: Normalized histogram Vs pdf of standard normal distribution for (c)

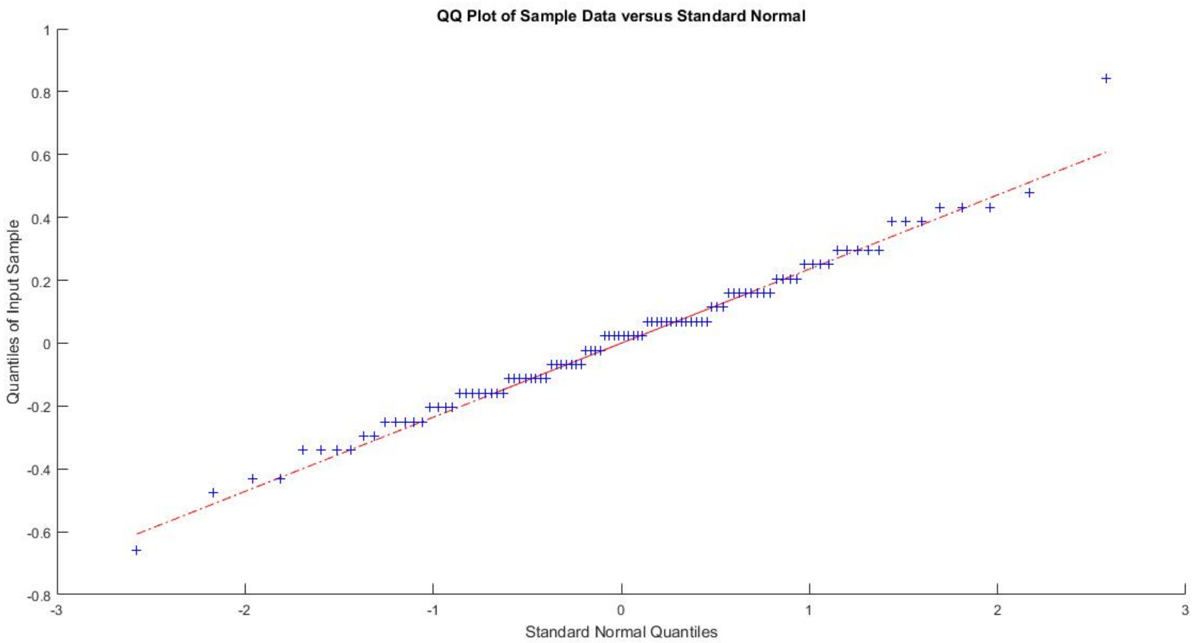

Figure 4: q-q plot for (c)

(d) $X_k = \frac{B_k}{2^k}$ **where** B_k **is a Bernoulli trail with** $p = 0.5$

To show that GCLT applies to (d) we need to show for any $\epsilon > 0$ there exists a sufficiently large n such that $\sigma_k < \epsilon S_n$.

$$B_k = \begin{cases} 1, & p \\ 0, & 1-p \end{cases}$$

$$X_k = \begin{cases} \frac{1}{2^k}, & p \\ 0, & 1-p \end{cases}$$

$$\mu_k = \frac{1}{2^k} * p + 0 * (1-p) = \frac{p}{2^k}$$

$$\sigma_k^2 = E[X_k^2] - E[X_k]^2 = \frac{p(1-p)}{4^k}$$

$$S_n^2 = \sigma_1^2 + \sigma_1^2 \cdots + \sigma_n^2 = \sum_{k=1}^{n} \sigma_k^2 = \sum_{k=1}^{n} \frac{p(1-p)}{4^k}$$

S_n^2 is a geometric series. as $n \to \infty$

$$S_n^2 = \frac{p(1-p)}{(1-\frac{1}{4})} = p(1-p) * \frac{4}{3}$$

lets take $\epsilon = \frac{1}{4}$ and $k = 1$

$$\sigma_1^2 = \frac{p}{2}$$

$$\epsilon S_n^2 = \frac{1}{4} * p(1-p)$$

$\frac{p}{2} > \frac{1}{4} * p(1-p)$ for any $p > 0 \implies \sigma_1^2 > \epsilon S_n^2$ when $n \to \infty$
Hence we can not find a large enough n such that $\sigma_1 < \frac{1}{4} S_n$

$\implies$ GCLT doesn't apply to (d).

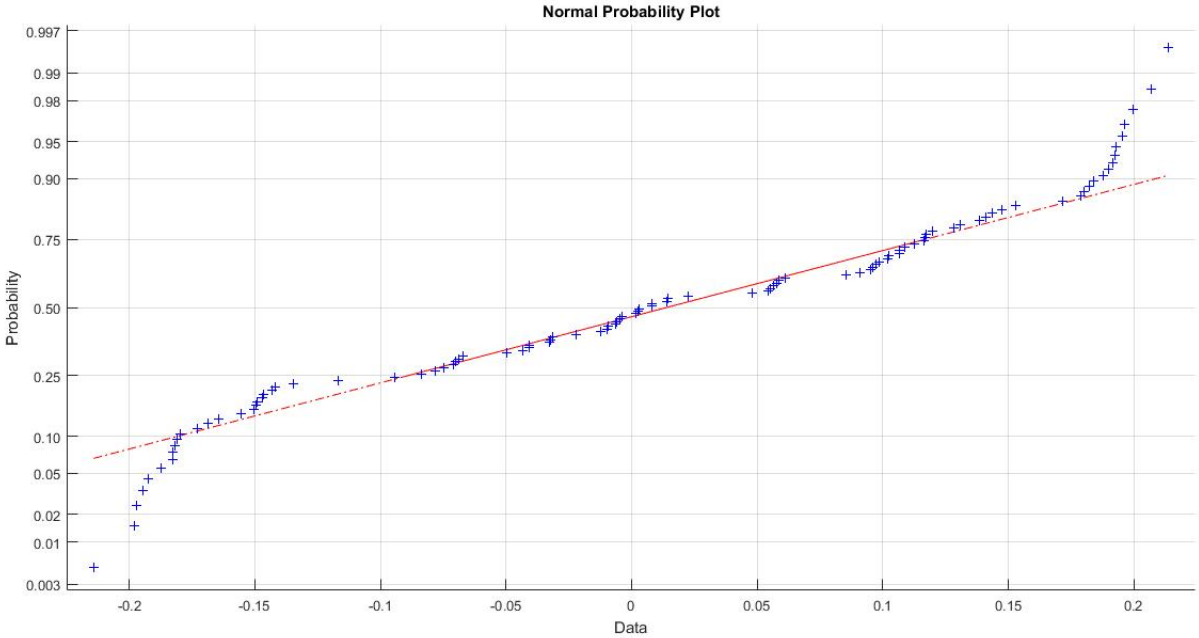

Figure 5: Normalized histogram Vs pdf of standard normal distribution for (d)

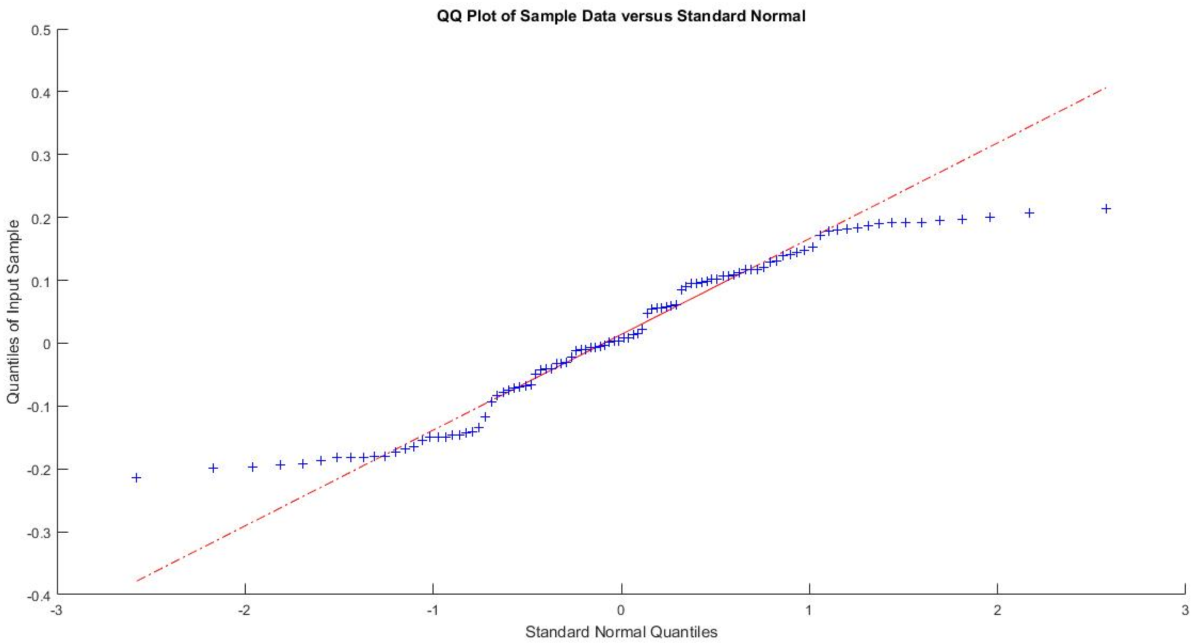

Figure 6: q-q plot for (d)

(e) How well the sum of the first ten elements is approximated by a normal distribution?

The sum of the first 10 elements in (b) and (c) approximated the normal distribution really well as we can see from the q-q plots (specially (c)) but for (d) the first 10 random variables are not enough to approximate a normal distribution.

Average Number of packets in Buffer	7.81
Fraction of time the buffer is empty	0.01
The fraction of packet Arrivals that are blocked	0.00

Table 2: Question 3b Answer

Q3 Buffers

(a) for $\lambda = 0.1$, $\mu = 0.05$, $BufferSize = 10$ and $NumberOfSteps = 200$ we get Table 2 and figure 7

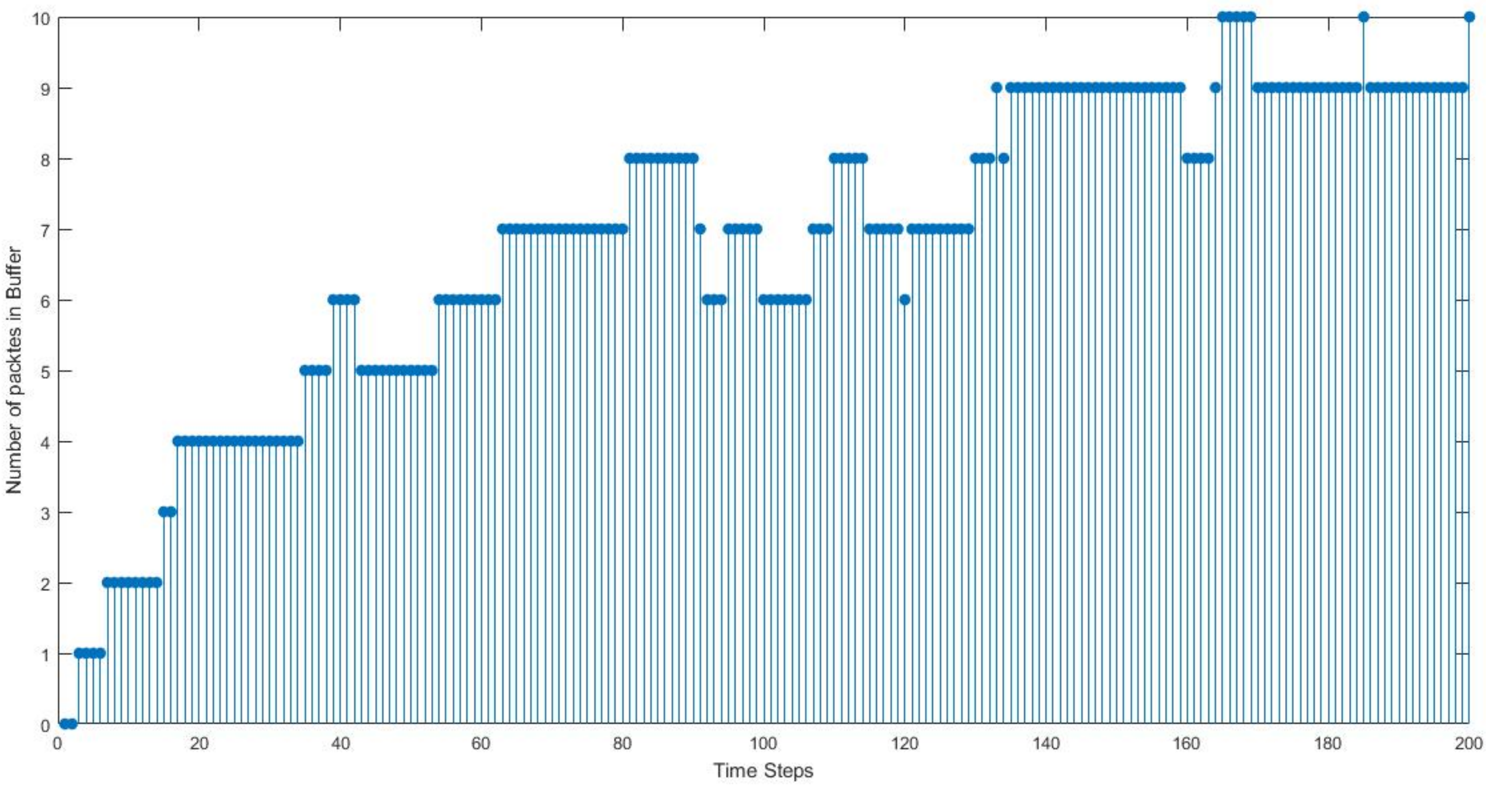

Figure 7: Number of packets in the buffer Vs Time steps

Average Number of packets in Buffer	6.23
Fraction of time the buffer is empty	0.04
The fraction of packet Arrivals that are blocked	0.2300

Table 3: Question 3 Answer

(b) for $\lambda = 0.1$, $\mu = 0.01$, $BufferSize = 10$ and $NumberOfSteps = 200$ we get table3 and figure 8

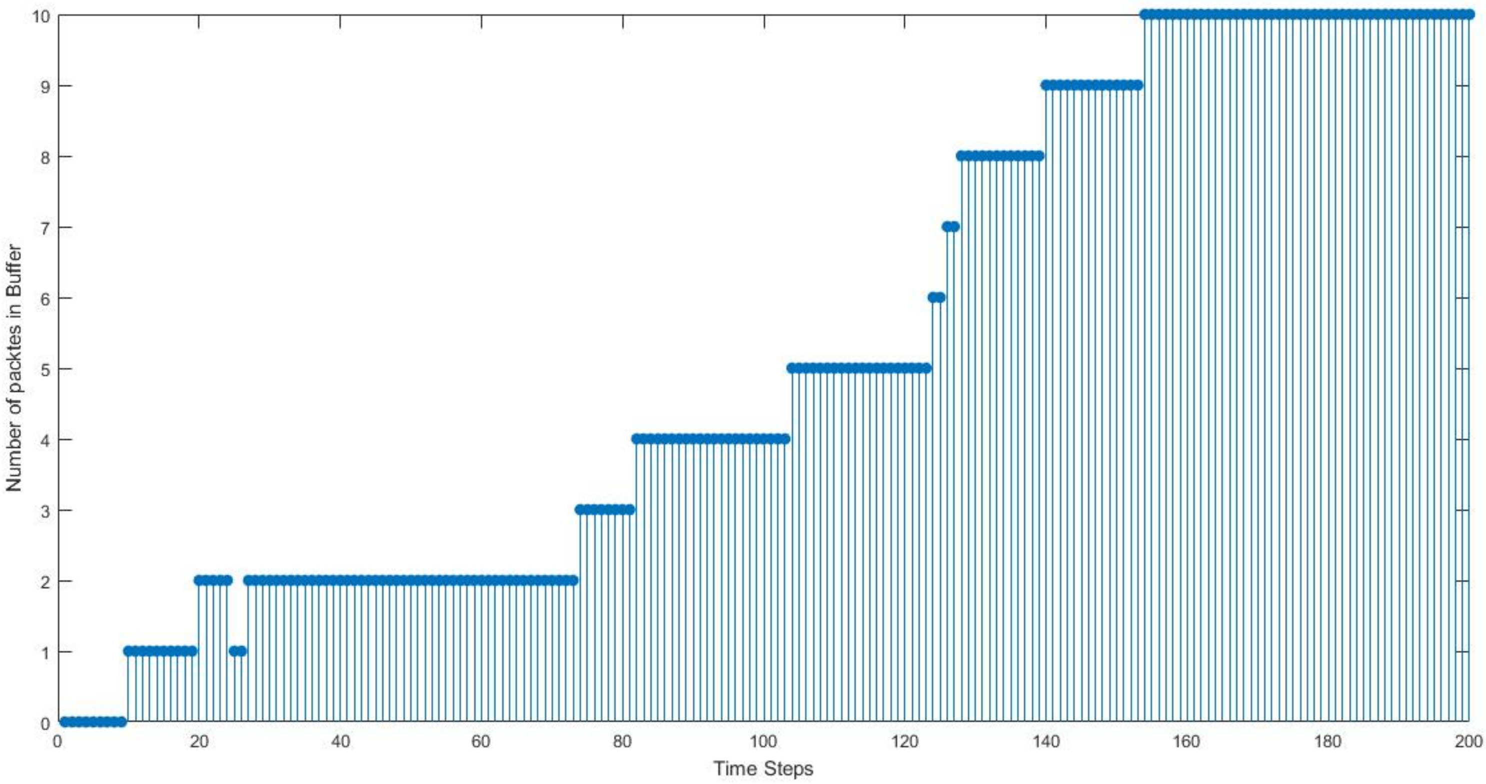

Figure 8: Number of packets in the buffer Vs Time steps

(c) Littel Law

Little Law is given by the following formula [1]

$$L = \lambda T \tag{2}$$

Where L is the average backlog(the average number of packets in the buffer) , T the delay in the system and λ is the average arrival rate.

hence to get the average delay in the system,

$$T = \frac{L}{\lambda} \tag{3}$$

for (a)

$$T = \frac{L}{\lambda} = \frac{7.81}{0.1} = 78.1$$

timesteps
for (b)

$$T = \frac{L}{\lambda} = \frac{6.23}{0.1} = 62.3$$

timesteps

Code Appendix

2. b, c and d

```
function [X,Mean,Var] = generate_random_variables_d(N,p)
    X = zeros(1,N);
    Mean = zeros(1,N);
    Var = zeros(1,N);
    for i=1:N
        X(i)    = (discrete([p,1-p])-1)/2^i;
        Mean(i) = p/2^i;
        Var(i)  = p*(1-p)/4^i;
    end
end
```

```
function [X,Mean,Var] = generate_random_variables(N)
X = zeros(1,N);
Mean = zeros(1,N);
Var = zeros(1,N);
for i=1:N
    X(i) = randi([1 i],1,1);
    Mean(i) = mean(1:i);
    Var(i)  = var(1:i);
end
end
```

```
function Yn = computeGCLT(X,Mean,Var)
    Yn = (sum(X)-sum(Mean))/sqrt(sum(Var));
end
```

```

%************************(b)********************
N = 100;
x = -N/2:N/2-1;
Ndpdf = normpdf(x);
Yns_b = zeros(N,1);
mu     = mean(1:5);%mean of uniform dist
sigma  = var(1:5);%var of uniform dist
for i=1:length(Yns_b)
    X = randi([1 5],1,10);%uniform distribution 1-5
    Yns_b(i) = computeGCLT(X,mu*ones(1,10),sigma*ones(1,10));
end
Yns_b = Yns_b/sum(Yns_b);
%************************(c)*********************
Yns_c = zeros(N,1);
for i=1:length(Yns_c)
    [X,mu,sigma]=generate_random_variables(10);
    Yns_c(i) = computeGCLT(X,mu,sigma);
end
Yns_c = Yns_c/sum(Yns_c);
%************************(d)*********************

Yns_d = zeros(N,1);
```

```
for i=1:length(Yns_d)
    [X,mu,sigma]=generate_random_variables_d(10,0.5);
    Yns_d(i) = computeGCLT(X,mu,sigma);
end
Yns_d = Yns_d/sum(Yns_d);
%************************Ploting*****************
%************************Plot Histogram***********
figure
subplot(2,2,1);
hist(Yns_b);
title('b');
subplot(2,2,2);
hist(Yns_c);
title('c');
subplot(2,2,3);
hist(Yns_d);
title('d');
%************************Normal Plot**************
figure
subplot(2,2,1);
normplot(Yns_b);
title('b');
subplot(2,2,2);
normplot(Yns_c);
title('c');
subplot(2,2,3);
normplot(Yns_d);
title('d');
%************************q-q Plot****************
figure;
subplot(2,2,1);
qqplot(Yns_b);
title('b');
subplot(2,2,2);
qqplot(Yns_c)
title('c')
subplot(2,2,3)
qqplot(Yns_d)
title('d')
```

3. a and b

```
function P = get_stochastic_matrix(buffer_size,lamda,mu)
    P = zeros(buffer_size+2,buffer_size+2);
    a = lamda*(1-mu);
    b = mu*(1-lamda);
    c = 1-(a+b);
    P(1,1) = 1-a;
    P(1,2) = a;
    P(buffer_size+2,buffer_size+2) = 1-mu;
    P(buffer_size+2,buffer_size+1) = lamda*mu;
    P(buffer_size+2,buffer_size)   = b;
    for i=2:buffer_size+1
        P(i,i) = c;
```

```
        P(i,i+1) = a;
        P(i,i-1) = b;
    end

end
```

```
DEBUG = 1;
N = 200;%time steps
State0 = 1;
lamda = 0.1;
mu = 0.001:0.001:0.01; % 10% mu
%mu = 0.02:0.01:0.2;      % 1% mu
low_load_mu = 0;
BUFFER_SIZE = 10;
percentage = 10;
MU_FOUND = 0;
for i=1:length(mu)
    P = get_stochastic_matrix(BUFFER_SIZE,lamda,mu(i));
    StateTrans = simMC(N,State0,P);
    lost_packets = mean(StateTrans==(BUFFER_SIZE+2))*100;%loss of packets
    if DEBUG
    fprintf('mu %0.4f buffer size %i lost packets %4.4f percent\n',mu(i),BUFFER_SIZE,lost_packets);
    end
    if lost_packets>percentage
    %if lost_packets<percentage
        low_load_mu = mu(i);
        MU_FOUND = 1;
        if DEBUG
            fprintf('mu %0.3f satisfies loss value of %i percent with packet loss of %4.4f\n',...
            mu(i),percentage,lost_packets);
        end


    end
end
if MU_FOUND
    % Some stat before modifying StateTrans

    StateTrans(StateTrans==(BUFFER_SIZE+2))=BUFFER_SIZE+1;%dropped == full
    Avg_Number_Of_Packets    = mean(StateTrans);
    Fraction_Of_Time_BEmpty = mean(StateTrans==1);
    Fraction_Of_Time_BBlocked = lost_packets/100;
    result = fopen('result_b.txt','w');
    fprintf(result,' Average Number of packets in Buffer: %2.2f\n Fraction of time the buffer is empty: %2.2f\n',...
        Avg_Number_Of_Packets,Fraction_Of_Time_BEmpty);
    fprintf(result,' The fraction of packet Arrivals that are blocked %2.4f\n',Fraction_Of_Time_BBlocked);
    fprintf(result,'\n MU:%2.2f\n Buffer Size: %i\n Lamda:%1.2f\n Number of Steps:%i\n Packet Loss:%2.2f percent\n',...
```

```
        low_load_mu,BUFFER_SIZE,lamda,N,lost_packets);
        StateTrans = StateTrans-1;% get rid of the bias so that it starts at state 0
    fclose(result);
    stem(1:N,StateTrans,'filled');
    xlabel('Time Steps');
    ylim([0 10]);
    ylabel('Number of packtes in Buffer');
else
    fprintf('appropriate mu not found try again!!\n')
end
```

```
function X = simMC(M,A,P)
X = zeros(1,M);
X(1) = A;
for m=1:M-1
      X(m+1) = discrete(P(X(m),:));
end
end
```

```
function T = discrete(P)
Pnorm = [0 P]/sum(P);
Pcum  = cumsum(Pnorm);
R = rand(1);
[~,T] = histc(R,Pcum);
end
```

References

[1] Jean Walrand. *Probability in Electrical Engineering and Computer science*. Jean Walrand, 2014.