Group 15
Stefan Åhman 900326-2376 sahman@kth.se
Marcus Wallstersson 880301-6099 mwallst@kth.se

Assignment 2: Simulation of a Mobile Communication System

The goal of this assignment is to help a phone company to test if their network meets the quality of service requirements (QoS) for dropped calls and blocked calls. Today the network consists of 20 base stations along a highway of 40km. The range of each base station is 2km and can hold up to 10 concurrent calls. If this setup isn´t enough to meet the QoS what changes to the network is needed to meet the requirements.

The limits defined as QoS:

- Percentage of blocked calls = 2.0%
- Percentage of dropped calls = 1.0%

Model of the system

To simulate this, a model of the problem was developed. The range of all base stations is equal and where one station ends the next begins. The area of the range was simulated to be quadratic instead of circular. The highway was assumed to be one-way to make the simulation less complex. In the simulation the highway is 40km, the 20 base stations has a range of 2 km each.

To get rid of the boundary problem the road was implemented as a circular road. All cars driving past the 40km mark will be wrapped around and continue from the beginning of the road.

- State variables
  - Total calls: Total started calls in the system, including dropped calls.
  - Dropped calls: Total active calls dropped by full base station at handover.
  - Blocked calls: Total call initiations blocked by full base stations.
- Events
  - Start call: A call is planned to start.
  - Handover: A call is planned to do a handover.
  - End call: A call will end.

When an event is created it´s placed in a future event list (FEL). All the events in FEL is sorted so the event with the lowest timestamp is always first. The simulation will handle one event at a time and then remove it from FEL when the event handler is done. The event handler is implemented like this:

If the event is of the type ”Start call”:

- A call is started at a position on the highway, if the current base station covering this position don’t has any free channels the call is blocked. If there is a free channel its allocated and the call is started. Next step is to plan the next event, if the call will end inside the

current base station an end event is created. If the call will end outside, a handover event is created.

If the event is of the type ”Handover”:

- Unallocated channel in current base station. Next step is to plan the next event, if the call will end inside the next base station an end event is created. If the call will end outside, a handover event is created. But only if there is free channels in the next base station. Otherwise the call will be dropped.

If the event is of the type ”End call”:

- Unallocated channel in the base station connected to the call.

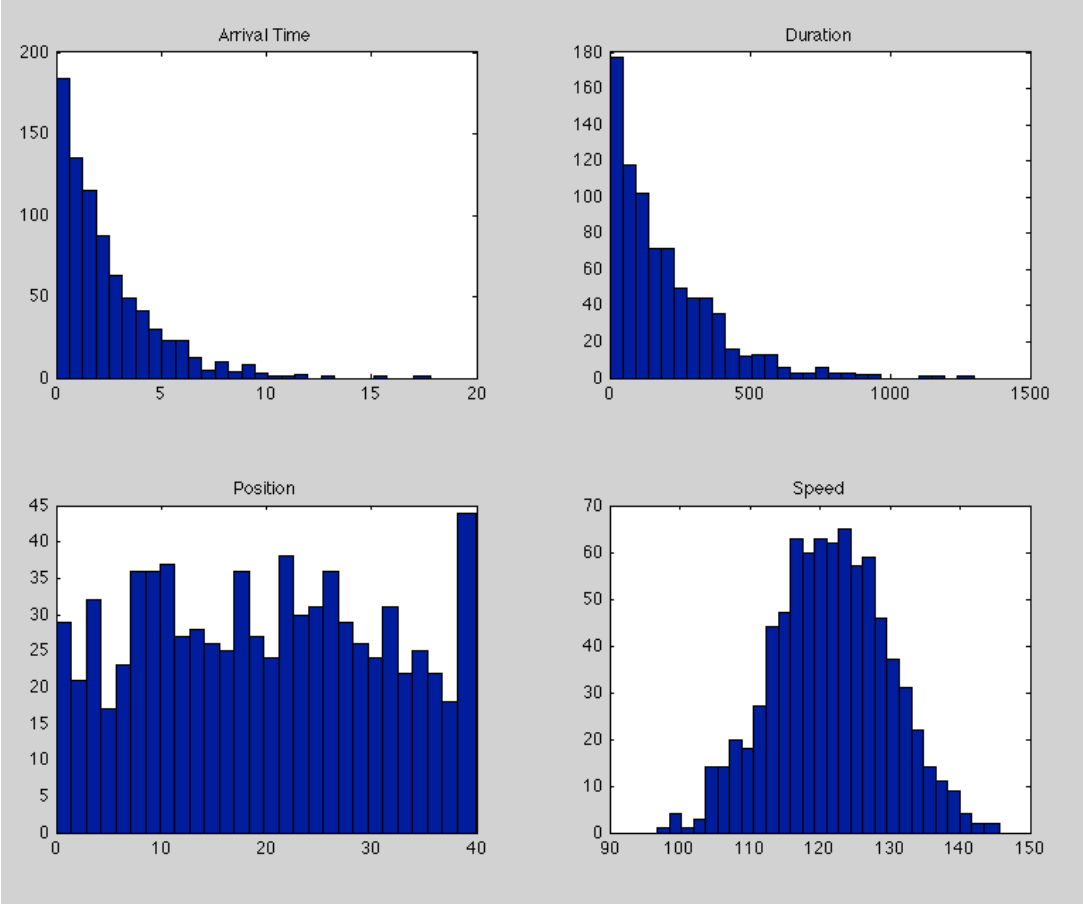
The handover process can be done in two ways.

1: FCA, The number of channels in the base station is fixed and can’t be changed. New calls and handovers allocates the same channels

2: FCA with handover reservation scheme, a chosen number of the channels are reserved only for handovers. The other channels is available for both handovers and new calls

Input data modelling

We made histogram of the data we've got. By plotting in Matlab we could estimate each distribution.



Group 15
Stefan Åhman 900326-2376 sahman@kth.se
Marcus Wallstersson 880301-6099 mwallst@kth.se

Since Matlab only got Chi-Square test for normal distributions (chi2gof), we created our own exponential and uniform Chi-Square test (see appendix). For input we have the list of data, number of intervals, the parameter and the estimated.

To estimate the required parameters for each distribution, we used the following estimators found in exercise slides for input data modelling:
Lambda = 1/mean(X), b = (n+1)n*max(X), my = mean(X) and sigma^2 = var(X)

Arrival time: Exponential

Lambda = 1/mean(X) = 0,4213

Calculate the mean value of X (list of data). Divide 1 by the mean of X to get lambda.
We ran the implemented function “chi2testexp” with input: list of inter arrival times, 29 intervals, 1 parameter and lambda.

The calculated chi squared value was 15.2625 with 27 degrees of freedom. From chi-squared table we get the value of chi2 significance level 0,05 equals to 40,1. Since our calculated chi-squared is lower than the table value we don’t need to reject our estimated lambda.

Duration: Exponential

Lambda = 1/mean(X) = 0,0051

Calculate the mean value of X (list of data). Divide 1 by the mean of X to get lambda.
We ran the implemented function “chi2testexp” with input: list of duration times, 29 intervals, 1 parameter and lambda.

The calculated chi squared value was 31.9375 with 27 degrees of freedom. From chi-squared table we get the value of chi2 significance level 0,05 equals to 40,1. Since our calculated chi-squared is lower than the table value we don’t need to reject our estimated lambda.

Position: Uniform

b = (n+1)n*max(X) = 39,9829

To get 'b', multiply the top value in your list (X) with (n+1)/n.
We ran the implemented function “chi2testuni” with input: list of position data, 29 intervals and parameter b.

The calculated chi squared value was 28.8200 with 27 degrees of freedom. From chi-squared table we get the value of chi2 significance level 0,05 equals to 40,1. Since our calculated chi-squared is lower than the table value we don’t need to reject our estimated value b.

Group 15
Stefan Åhman 900326-2376 sahman@kth.se
Marcus Wallstersson 880301-6099 mwallst@kth.se

Speed: Normal

my = mean(X) = 121,495, sigma^2 = var(X) = 67,5578
Calculate the mean and variance to get the two parameters.
We ran the function "chi2gof" with input: list of inter arrival times. The calculated chi squared value was 3.9871 with 7 degrees of freedom. From chi-squared table we get the value of chi2 significance level 0,05 equals to 14,1. Since our calculated chi-squared is lower than the table value we don't need to reject our estimated values 'my' and 'sigma^2'.

Conclusion of goodness-of-fit test

Since all the estimated values were non-rejectable we can use them in our simulation.

Verification an Validation of model and implementation

We ran multiple simulations with different values of the seed, simulation time. The results got very similar each run except in the case of very short simulation time. This is expected because in the beginning is not many calls active on the road, later on when the simulation reaches a steady state the blocked and dropped % gets similar (see Table 1)

Seed 1, Replications 100, Warm-up 0, Channels 10, Reserved 0

Simulation time	Blocked calls %	Dropped calls %
1000	0.2502	0.5996
10 000	0.4837	1.279
40 000	0.4854	1.299

Table 1.

We did a stress test of the simulation by decreasing the inter arrival times between calls, this was done by changing the estimated lambda value to 1. This nearly cut down the inter arrival times by half. The expected results of this were that the percentage of blocked and dropped calls would rise very much. This was confirmed by running the simulation with lambda set to 1. Results presented in Table 2.

Seed 1, Length 10000 Replications 100 and Warm-up 0

Channel configurations	Blocked calls %	Dropped calls %
10 ch 0 res	9.616	20.67
18 ch 1 res	1.085	0.7533

Table 2.

By choosing 18 channels and 1 reserved we could get results that satisfies the QoS values. This shows that the simulation is valid for different variables.

Output analysis

The warm-up time was estimated by running simulations with different warm-up time trying to find the steady state. The parameters for all runs in Table 3 were: seed 1, simulation length 10000, replications 100, channels 10 and reserved 0. From the test results steady state begins at approx. 800 sec but to be

Stefan Åhman 900326-2376 sahman@kth.se
Marcus Wallstersson 880301-6099 mwallst@kth.se

on the safe side we estimated it to 1000 sec. It´s better to have a value that's a little too high than a too small value because the system will be in steady state.

Warm-up (sec)	Blocked calls %	Dropped calls %
8000	0,51645	1,2809
7000	0,50619	1,3022
6000	0,50487	1,3256
5000	0,51572	1,3385
4000	0,51595	1,3502
3000	0,51673	1,3609
2000	0,51218	1,3495
1000	0,50916	1,3548
900	0,50769	1,3525
800	0,50559	1,3478
700	0,50377	1,342
600	0,50303	1,341
500	0,50264	1,3337
400	0,50065	1,3261
300	0,4982	1,3173
200	0,49349	1,3048
100	0,48855	1,2923
0	0,48369	1,2794

Table 3.

In following tests in Table 4 the seed was 1, simulation length 10 000 sec, replications 100 and warm-up length 1000 sec.

Channel configurations	Blocked calls % QoS 2%	Dropped calls % QoS 1%
10 ch 0 res	0.5092 ± 0.0283	1.3548 ± 0.0573
10 ch 1 res	1.6211 ± 0.0550	0.8065 ± 0.414
10 ch 2 res	3.8111 ± 0.0933	0.3549 ± 0.0243
11 ch 0 res	0.2033 ± 0.0177	0.5726 ± 0.0392

Table 4.

To calculate the confidence intervals for the results we implemented our own confidence interval calculation function in Matlab (see appendix). To get the confidence interval for a simulation we collected the data of blocked and dropped calls in every replication and stored them in separate lists. After that we did calculations from this lists with our confidence function. It gave us the results in Table 4.

According to the results in Table 4 our suggestions to the phone company is either to use 10 channels and 1 reserved or 11 channels and 0 reserved to satisfy the QoS requirements.

Group 15
Stefan Åhman 900326-2376 sahman@kth.se
Marcus Wallstersson 880301-6099 mwallst@kth.se

Appendix

---------- chi2testuni ----------

```
function [ sumchi,df ] = chi2testuni(list,interval,b)
%Chi2Test
%Input:     [list,interval,parameters,lambda]
%Output:    [X0,Degrees of freedom]

interval_a = 1/interval;
lengthoflist = length(list);
expected = lengthoflist/interval;

a = [];
for x = 0:interval_a:1
   a = [a x*b]; %#ok<AGROW>
end

obs = zeros(1,interval);
j = 1;
while(j < lengthoflist+1)
    for i = 1:interval
        if (a(i) <= list(j) && list(j) < a(i+1))
            obs(i) = obs(i) + 1;
        end
    end
    j = j + 1;
end

chilist = [];
for l = 1:interval
    chilist = [chilist ((obs(l)-expected)^2/expected)]; %#ok<AGROW>
end

sumchi = sum(chilist);
end
```

---------- chi2testexp ----------

```
function [ sumchi,df ] = chi2testexp( list,interval,param,lambda )
%Chi2Test
%Input:     [list,interval,parameters,lambda]
%Output:    [X0,Degrees of freedom]

interval_a = 1/interval;
lengthoflist = length(list);
expected = lengthoflist/interval;

a = [];
for x = 0:interval_a:1
   a = [a -1*(log(1-x)/lambda)]; %#ok<AGROW>
end

obs = zeros(1,interval);
j = 1;
while(j < lengthoflist+1)
```

Stefan Åhman 900326-2376 sahman@kth.se
Marcus Wallstersson 880301-6099 mwallst@kth.se

```
    for i = 1:interval
        if (a(i) <= list(j) && list(j) < a(i+1))
            obs(i) = obs(i) + 1;
        end
    end
    j = j + 1;
end

chilist = [];
for l = 1:interval
    chilist = [chilist ((obs(l)-expected).^2/expected)]; %#ok<AGROW>
end

sumchi = sum(chilist);

df = interval - param - 1;

end
```

---------- Confidence interval function ----------

```
function [ X ] = conf( list )
% Confidence interval

alpha = 0.05;

t_alpha_2_120 = 1.98;

meanv = mean(list);
lengthv = length(list);
a = [];

for x = list
   a = (x-meanv).^2;
end

asum = sum(a);

s = sqrt(asum/(lengthv-1));

plusminus = (t_alpha_2_120*(s/sqrt(lengthv)));

X = [meanv,plusminus];

end
```
