

Computer Engineering 4DK4
Computer Communication Networks

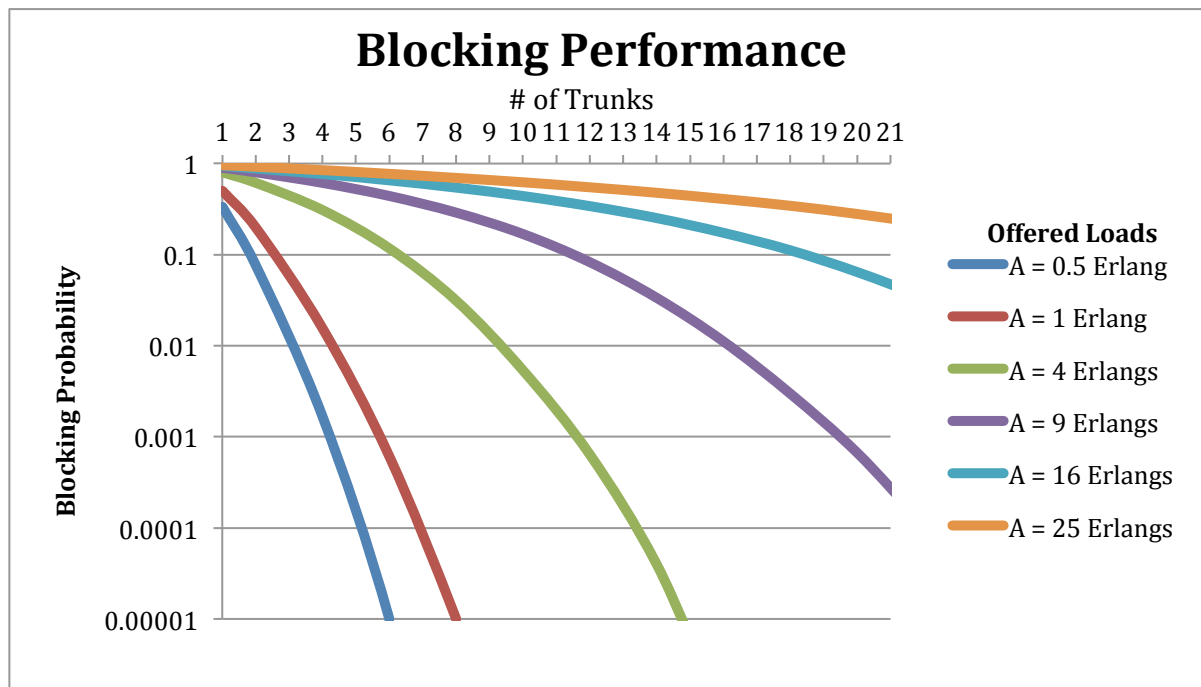
Lab 3 Report

Matthew Wong
1141381

Roy Perez
001132788

Experiments

2. The plot below shows the tradeoffs between the blocking probabilities and the number of channels for various offered loads. We can see from the plot that as we increase the offered load, the number of trunks required to maintain the 1% performance also increases. We also can see that as the offered load increases, the number of trunks does not increase linearly. For example, at an offered load of 4 Erlangs, 9 trunks are needed in order to maintain the 1% probability performance. At an offered load of 9 Erlangs, 16 trunks are needed. This is due to the multiplexing gain or the trunking efficiency. As the system size increases, it becomes more efficient in managing the use of the number of trunks. We have compared our results with an online Erlang B calculator and we can confirm that they are consistent.



Offered Load	0.5	1	4	9	16	25
Random Seed	1141381					
1	0.33264	0.49974	0.7999	0.90018	0.94121	0.96157
2	0.07675	0.19914	0.61527	0.80172	0.88274	0.9232
4	0.00167	0.01554	0.30849	0.61362	0.76728	0.84688
6	0.00001	0.00061	0.1171	0.44064	0.65414	0.77123
8	0.00000001	0.00001	0.03132	0.28916	0.54428	0.69576
10	0	0.0000001	0.00537	0.16838	0.4401	0.6231
12	0	0	0.00064	0.08273	0.34095	0.54876
14	0	0	0.00004	0.03358	0.25211	0.47785
16	0	0	0.000001	0.01114	0.17366	0.40833
18	0	0	0.0000001	0.00298	0.11164	0.344
20	0	0	0	0.00067	0.06448	0.27956
22	0	0	0	0.00011	0.03363	0.22014
24	0	0	0	0.00002	0.01501	0.16858

Erlang B Results Table

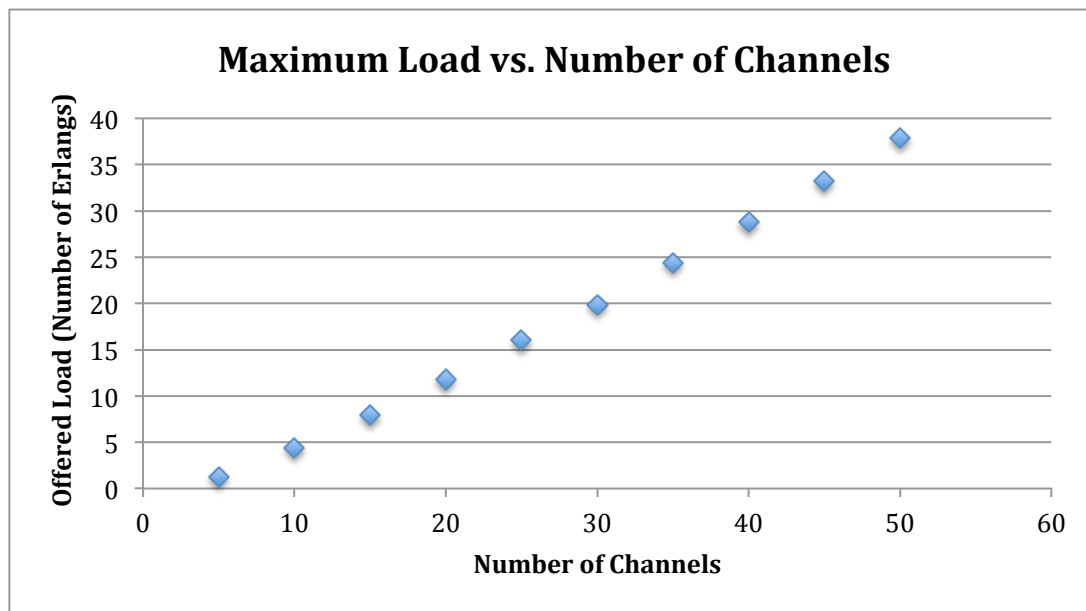
Here are the results (max 20) of the Erlang B Calculator. The unknown figures are shown in red.

B.H.T.	Blocking	Lines
0.500	0.010	4
1.000	0.010	5
4.000	0.010	10
9.000	0.010	17
16.000	0.010	25

© Westbay Engineers Ltd. 2001.
Results displayed - 11/4/2015, 11:01:53 PM

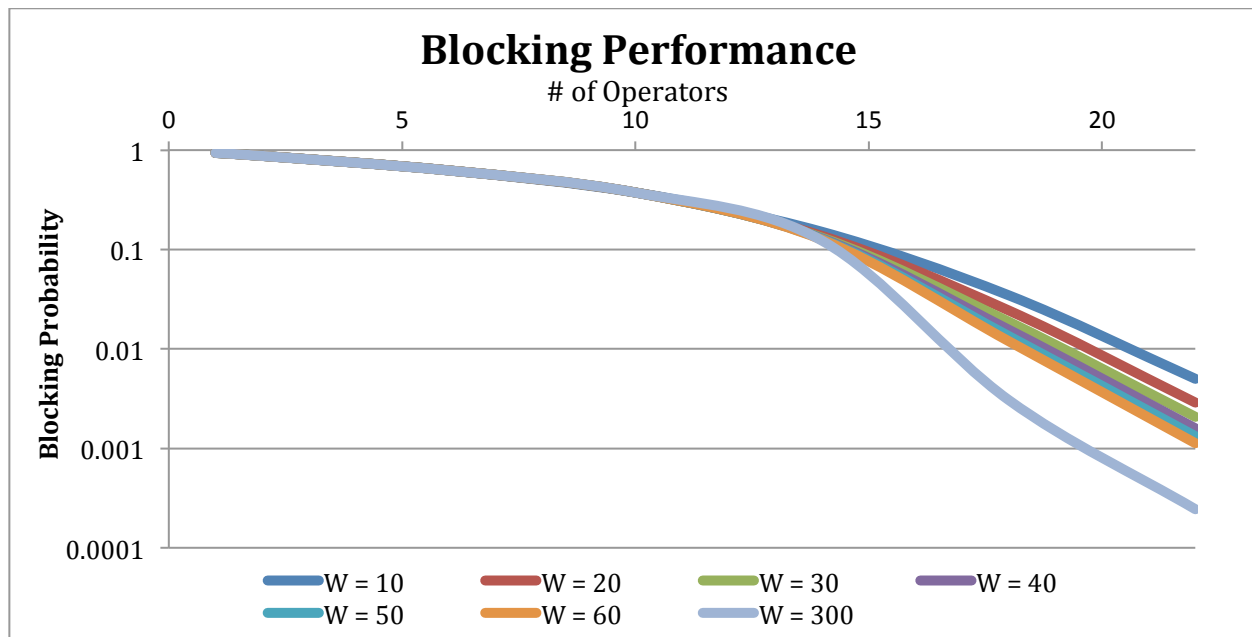
Online Erlang B Calculator: <http://www.erlang.com/calculator/erlb/>

3. Below is a plot of the maximum offered loading (in Erlangs) versus the number of cellular channels needed to achieve this blocking probability performance. As you increase the number channels, the offered load also increases.



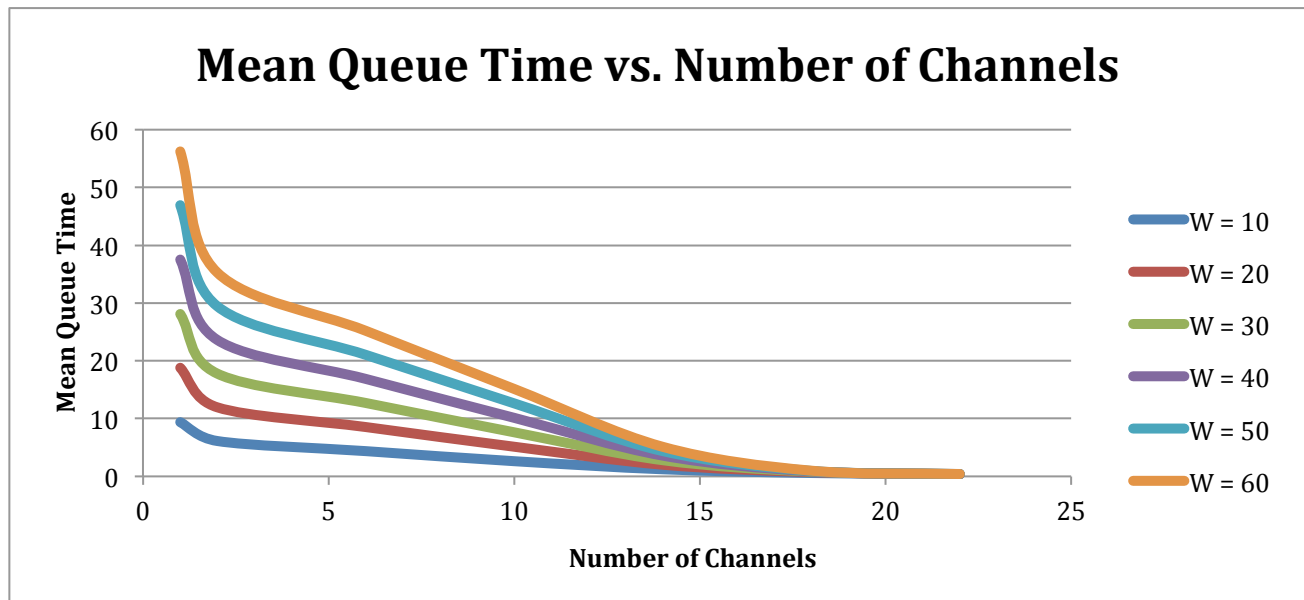
Number Of Channels	Max Erlangs
5	1.3
10	4.4
15	7.9
20	11.8
25	16
30	19.9
35	24.4
40	28.8
45	33.2
50	37.83

4. The plot below shows that as the average time a customer is willing to wait increases, the number of operators required is reduced. This makes sense practically, because if a call center is very busy with few staff to answer calls, customers must wait longer in the queue (or if they don't want to wait, they will eventually hang up).



Number of Trunks							
	1	2	6	10	14	18	22
10	0.937605	0.87506	0.62489	0.37448	0.150765	0.03469	0.005015
20	0.93761	0.874995	0.625155	0.374885	0.137135	0.02443	0.002895
30	0.937575	0.87495	0.625185	0.374875	0.131535	0.018815	0.00208
40	0.93757	0.87492	0.625165	0.37499	0.129125	0.01564	0.001595
50	0.93758	0.874905	0.62506	0.375025	0.127525	0.01351	0.00133
60	0.937545	0.874885	0.62506	0.374905	0.12566	0.012	0.00113
300	0.937245	0.874365	0.624155	0.374535	0.12279	0.00307	0.000245

Here we can see that with a low number of channels available, the mean queue waiting time is very close to their respective W value.



	1	2	6	10	14	18	22
W	Random Seeds: 1141381, 1132788						
10	9.37908	6.124673333	4.37159	2.628553333	1.228503333	0.58431	0.338293333
20	18.7578	11.95631667	8.539823333	5.13042	2.017466667	0.745233333	0.37376
30	28.13487	17.7885	12.70559667	7.62629	2.798876667	0.828066667	0.391586667
40	37.513545	23.62035667	16.86833333	10.12922667	3.594873333	0.884236667	0.402683333
50	46.89222	29.45183	21.03274333	12.63041333	4.376983333	0.942303333	0.406863333
60	56.26813	35.28242	25.19567333	15.12881667	5.115773333	0.989333333	0.413063333
300	281.25513	175.14512	124.9948167	75.07396667	24.62498	1.185363333	0.43977

5. Using the Erlangs C formula with the following parameters, we get an average failure rate of 15.4% for $t = 15$ second service target time.

Number of Channels: 55

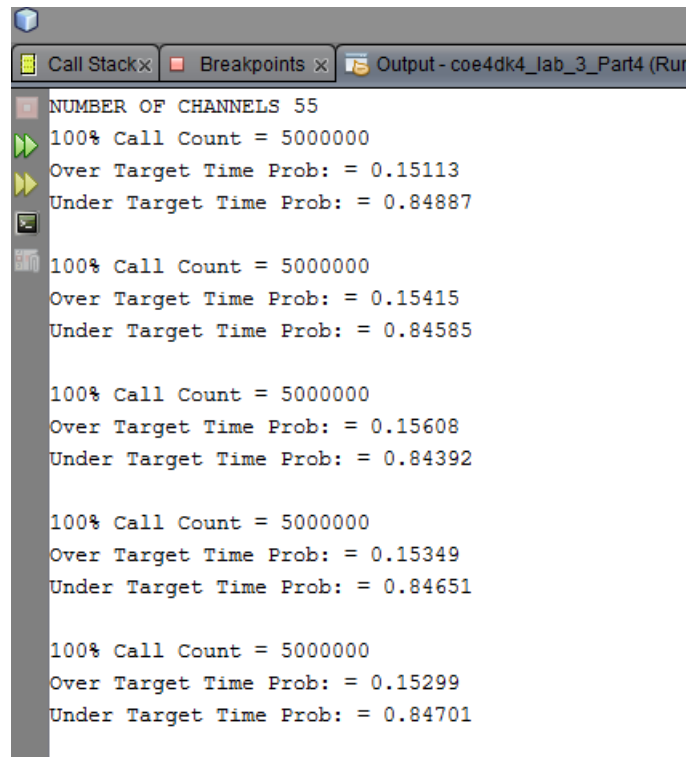
Mean Call Time: 4 Minutes

Calls per Minute: 12

Erlangs (A) = $4 * 12 = 48$

W(t) = 84.6% success, 15.4% failure.

In our simulation, we get an output of:



The screenshot shows a window titled 'Output - coe4dk4_lab_3_Part4 (Run)'. It displays the following output:

```

NUMBER OF CHANNELS 55
100% Call Count = 5000000
Over Target Time Prob: = 0.15113
Under Target Time Prob: = 0.84887

100% Call Count = 5000000
Over Target Time Prob: = 0.15415
Under Target Time Prob: = 0.84585

100% Call Count = 5000000
Over Target Time Prob: = 0.15608
Under Target Time Prob: = 0.84392

100% Call Count = 5000000
Over Target Time Prob: = 0.15349
Under Target Time Prob: = 0.84651

100% Call Count = 5000000
Over Target Time Prob: = 0.15299
Under Target Time Prob: = 0.84701

```

The average if the under target time is the same as expected 84.6%.

Code Changes

Part 2: -Add loop for number of channels used

- add **number_of_channels** field for **sim_data** to be fetched instead of constant
- change output functions for more easily parsed data

Part 3: -add for loop to increment **meanCallTime** by small fraction

- change output functions for more easily parsed data

Part 4: -add fields to sim data: **number_of_hangups**, **accumulated_queue_time**,

busyOperatorFifoQueue

- add field to Call_Ptr: **giveUpTime**
- initialize these fields in Main.C
- in **call_arrival_event()**, if call is blocked, set **giveUpTime**, place call in queue
- in **end_call_on_channel_event()**, after call leaves channel, check

busyOperatorFifoQueue's for calls, if it is not timed out, put it in channel and schedule end call on channel.

- change output functions for new variables and more easily parsed data

Part 5 (from code in Part 4):

- in **end_call_on_channel_event()**, check if call has been queued for more than target time, if so, increment failed target time counter
- ignore **giveUpTime**, always place in channel on departure