Investigating Point-to-point MPI Communications for Modeling Purposes

Arnaud Legrand

February 8, 2012

Contents

1 Exploratory Analysis of SkaMPI Measurements by Stéphane Génaud (09/07/10) 1
1.1 Setup . . . 1
1.2 Global Comments . . . 2
1.3 Small messages . . . 3
1.4 Large messages . . . 5
1.5 Medium messages . . . 6
1.6 Conclusion . . . 7

2 Exploratory Analysis of SkaMPI Measurements by Mark Stillwell (15/09/11) 7
2.1 Conclusion . . . 9

3 Exploratory Analysis of plogp Measurements by Stéphane Génaud (10/2008) 10
3.1 Setup . . . 10

4 Exploratory Analysis of SkaMPI Measurements by Arnaud Legrand (23/01/12) 13
4.1 Short messages . . . 16
4.2 Large messages . . . 17
4.3 Medium Messages . . . 18
4.4 Recap . . . 20

5 Zoo: Communication time classification 22
5.1 Experiment description . . . 22
5.2 Exploratory Analysis . . . 22
5.3 A simple two-range piece-wise linear model . . . 31

1 Exploratory Analysis of SkaMPI Measurements by Stéphane Génaud (09/07/10)

1.1 Setup

In this section, I analyze the measurements performed by Stéphane Génaud on Griffon using SkaMPI. The analysis was done on 11/01/12. The output of SkaMPI was converted into a readable R format using the following command:

```
grep count griffon_skampi_pt2pt.ski.dat | sed 's/count= //' > griffon_skampi_pt2pt.2011-09-15.dat
```

1.2 Global Comments

Let's start by loading the SkaMPI measurements:

```
> df <- read.table("griffon_skampi_pt2pt.2011-09-15.dat")
> names(df) <- c("count", "countn", "time", "stddev", "iter", "mini", "maxi")
> df <- df[!(names(df) %in% c("countn", "iter", "mini", "maxi"))]
```

Now, let's have a first look at the data frame (see Figure 1): Figure 1 shows some "outliers"

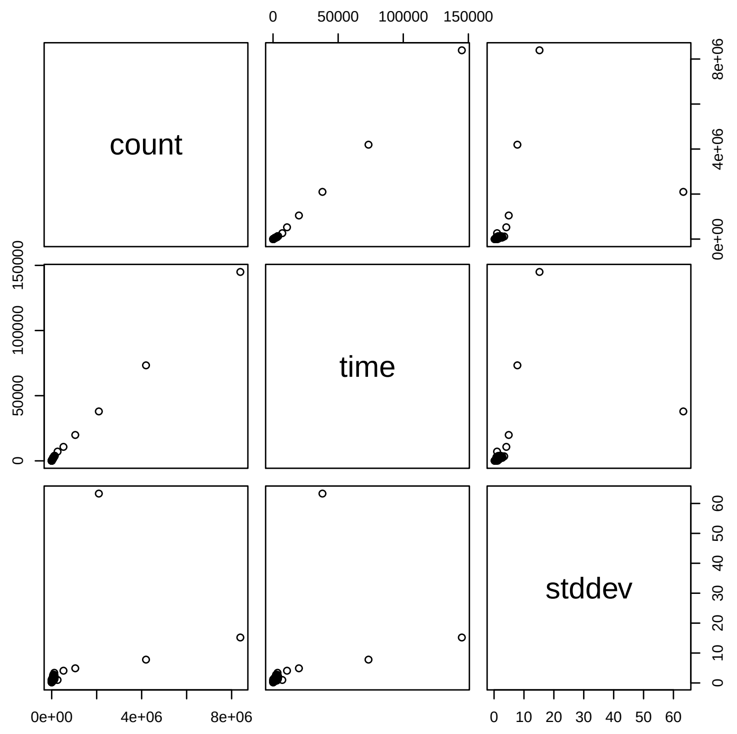


Figure 1: Scatter plot of data to get a first feeling.

for the last three values of stddev. Also it's hard to tell from such view and distribution, we need to keep in mind that some timing averages may be weird (especially the third last one).

Figure 1 also indicates that time is indeed roughly linear with count. Yet, the sampling of count was exponential, which is not the best for doing linear regression. Furthermore, we know the prediction error is actually rather relative to the time. A small error (in magnitude) has a complete different impact for small transfers and for large ones. So let's have a look at the same data but in log scale (see Figure 2): Now, we can distinguish the relation between time and count much better.

Let's only keep this one (see Figure 3):

This new plot definitely looks like piece-wise linear. From what I see, I decided to perform three regressions (in log).

```
> fit_small = lm(log10(time) ~ log10(count), data = df[df$count < 100, ])
> fit_medium = lm(log10(time) ~ log10(count), data = df[df$count < 20000 & df$count >
+     100, ])
> fit_large = lm(log10(time) ~ log10(count), data = df[df$count > 20000, ])
```

Let's look at these regressions

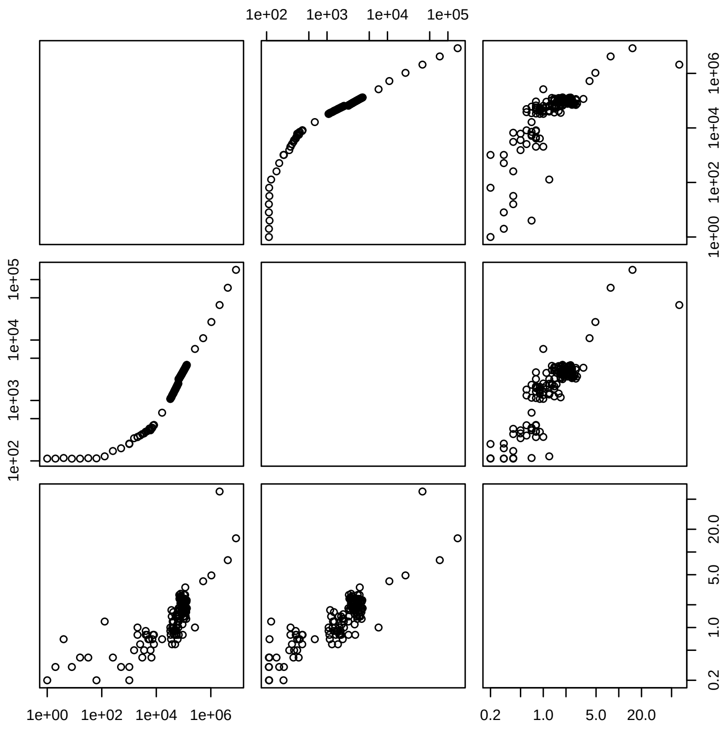


Figure 2: Scatter plot of data in log scale.

1.3 Small messages

```
> summary(fit_small)
```

```
Call:
lm(formula = log10(time) ~ log10(count), data = df[df$count <
    100, ])

Residuals:
         1          2          3          4          5          6          7
-0.0012296 -0.0013977  0.0074698 -0.0037225 -0.0038898  0.0034339 -0.0006641

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.039452   0.003055 667.666 1.43e-13 ***
log10(count) 0.001879   0.002814   0.668    0.534
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.004483 on 5 degrees of freedom
Multiple R^2:  0.08184,    Adjusted R^2:  -0.1018
F-statistic: 0.4457 on 1 and 5 DF,  p-value: 0.534
```

Sweet. It's almost constant (slope is very close to 0). Let's make sure about this without the log.

```
> summary(lm(time ~ count, data = df[df$count < 100, ]))
```

```
Call:
lm(formula = time ~ count, data = df[df$count < 100, ])
```

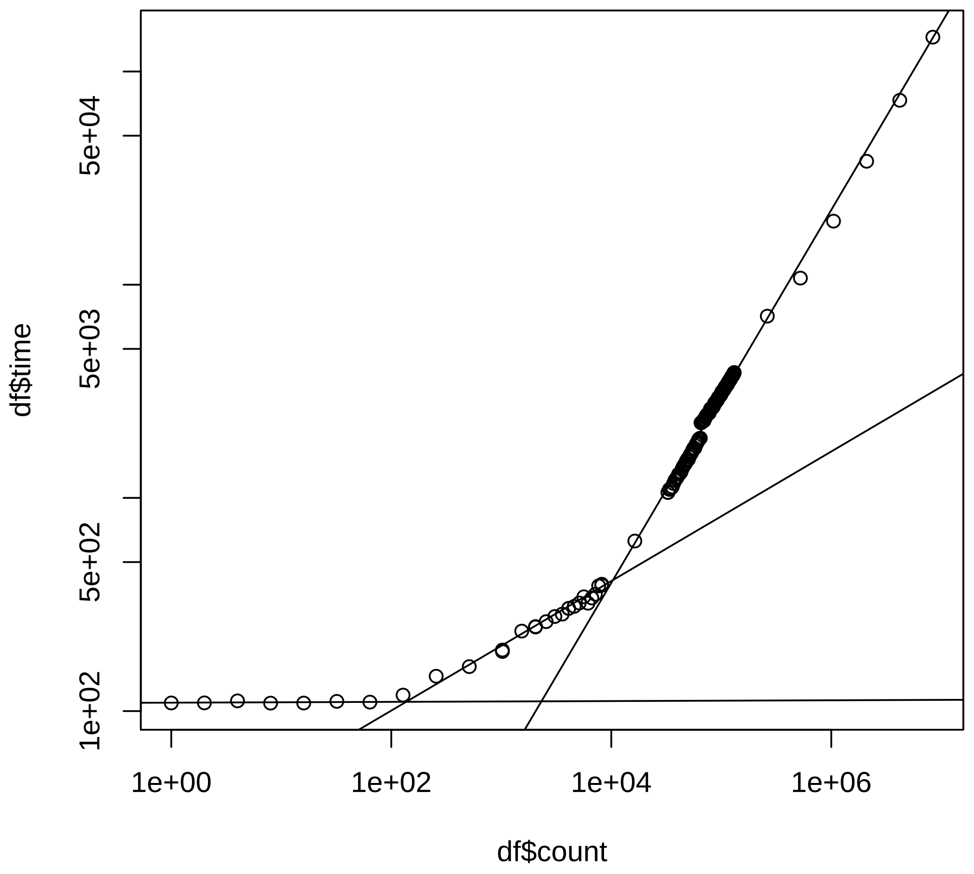


Figure 3: Communication time (time) versus data size (count) in log scale

```
Residuals:
      1        2        3        4        5        6        7
-0.5395  -0.4513   1.9250  -0.8225  -0.8174   0.9927  -0.2870

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 109.72759    0.57169  191.93 7.28e-11 ***
count         0.01187    0.02047    0.58    0.587
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.15 on 5 degrees of freedom
Multiple R^2: 0.06298,    Adjusted R^2: -0.1244
F-statistic: 0.3361 on 1 and 5 DF,  p-value: 0.5873
```

Cool. Slope is still indistinguishable from 0 (no `***` for the count slope). P-value is pretty large though, which means that there is a lots of noise. Let's look at the corresponding data from closer:

```
> df2 = df[df$count < 100, ]
> plot(df2$count, df2$time)
> abline(lm((time) ~ 1, data = df2))
```

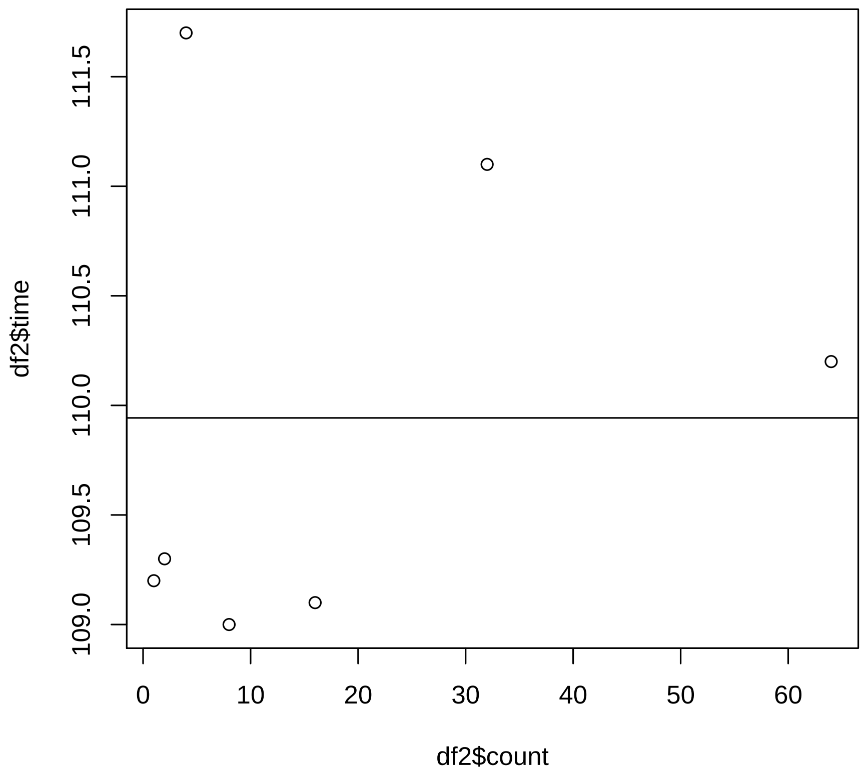


Figure 4: Communication time (time) versus data size (count) for small messages.

As can be seen, there is a lot of noise. Thus, it would be much better to work on the original measurements instead of the statistics provided by SkaMPI.

My conclusion is that for message size smaller than 1E02 (this value is crude and should obviously be refined), we can consider that communication is constant. Simple experiments to confirm this could be designed.

1.4 Large messages

```
> summary(fit_large)
```

```
Call:
lm(formula = log10(time) ~ log10(count), data = df[df$count >
    20000, ])

Residuals:
     Min       1Q   Median       3Q      Max
-0.07410 -0.02507  0.01030  0.01636  0.03774

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.898185   0.032321  -27.79   <2e-16 ***
```

```
log10(count)  0.874772   0.006507  134.44   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02429 on 104 degrees of freedom
Multiple R^2: 0.9943,    Adjusted R^2: 0.9942
F-statistic: 1.808e+04 on 1 and 104 DF,  p-value: < 2.2e-16
```

Damned. I wish the slope had been 1 and not 0.89. The fact that the slope is not 1 means that the original phenomenon is not well linear...

Yet, note that the sampling of count in this range is odd. Let me illustrate this (see Figure 5):

```
> hist(log(df$count), col = "gray")
```

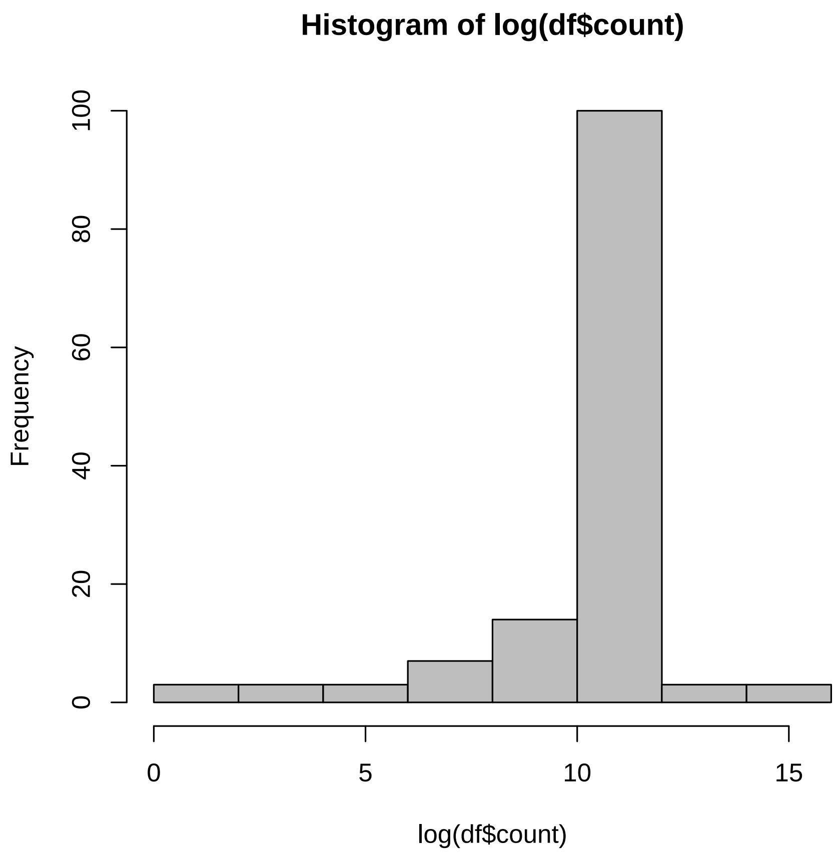


Figure 5: Histogram of the logarithm of message size. For some reason a range is much more sampled than the rest. This kind of things biases regressions a lot.

Such a sampling of count is very likely to mess the regressions so I would not trust what we can derive from these measurements.

1.5 Medium messages

```
> summary(fit_medium)
```

```
Call:
lm(formula = log10(time) ~ log10(count), data = df[df$count <
    20000 & df$count > 100, ])

Residuals:
      Min         1Q     Median         3Q        Max
-0.039489  -0.017025  -0.010072  0.009067  0.122748

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.39459    0.04871   28.63  < 2e-16 ***
log10(count)  0.30382    0.01399   21.72  7.17e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03376 on 21 degrees of freedom
Multiple R^2: 0.9574,     Adjusted R^2: 0.9553
F-statistic: 471.7 on 1 and 21 DF,  p-value: 7.17e-16
```

With a slope of 0.30, this means the original model is really not linear. This range is not too large and fortunately, it is straight in logscale, hence polynomial (that's our real slow-start). This slow-start could be approximated using piece-wise lines but at least, now, we know where, how and why to do it.

1.6 Conclusion

- The measurements need to be conducted again and the analysis should be done with the original data, not with SkaMPI statistics.
- The sampling of message size should be better organized and correctly randomized.
- We may expect a constant model for very small messages, and a piece-wise linear model with three ranges for other messages.
- The gap/overhead phenomenon may be incorporated to these experiments.

2 Exploratory Analysis of SkaMPI Measurements by Mark Stillwell (15/09/11)

Now, I look at the ping-pong measurements provided by Mark Stillwell obtained on the griffon cluster using a custom MPI code. The analysis was done on 12/01/12.

```
> df <- read.table("pingpong-in.dat")
> names(df) <- c("count", "time")
```

Now, let's have a first look at the data frame (see Figure 6):

```
> plot(df)
```

```
> summary(lm(time ~ count, data = df))
```

```
Call:
lm(formula = time ~ count, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-303.43 -192.25  -47.19  181.63 2646.58

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.921e+02  2.420e+00   120.7   <2e-16 ***
count       8.565e-03  1.266e-06  6765.8   <2e-16 ***
---
```

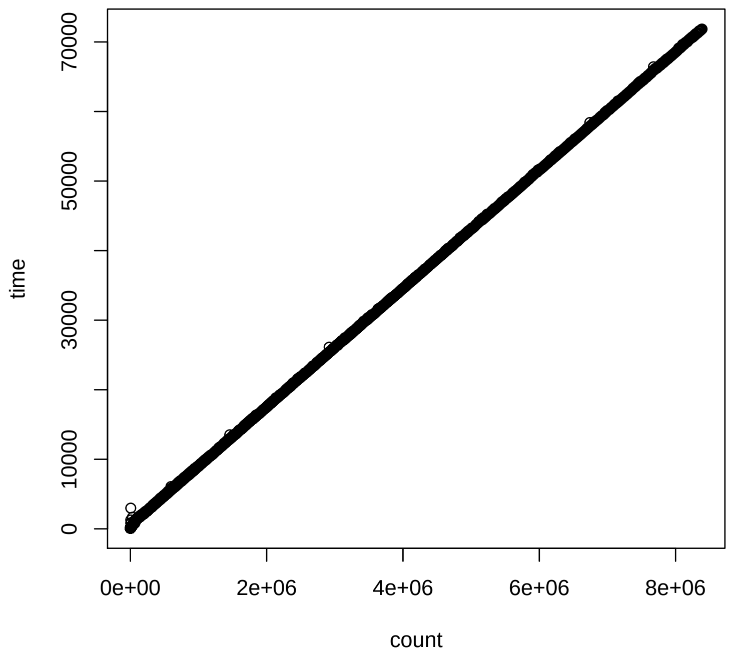


Figure 6: Communication time (time) versus data size (count) in linear scale.

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 190.5 on 7679 degrees of freedom
Multiple R^2: 0.9998,     Adjusted R^2: 0.9998
F-statistic: 4.578e+07 on 1 and 7679 DF,  p-value: < 2.2e-16
```

Figure 7 depicts the quality of the linear regression.

In linear scale, time seems perfectly linear with size. Unfortunately residuals are correlated with fitted values (let's call this slow-start...) and residuals are not well distributed at all. This is annoying as, we need a very good modeling for any message size, so let's have a look at it in log scale:

```
> plot(df, log = "xy")
> abline(v = 10000, col = "blue")
> abline(v = 65000, col = "blue")
> abline(v = 300)
> abline(v = 1400, col = "green")
> abline(v = 1)
> abline(-2.1, 1)
> curve(100 + x * 0.008565, col = "red", add = TRUE)
```

- Interestingly, we can see two clear gaps (discontinuity) when count is roughly equal to 1E+04 and 6.5E04 (blue vertical lines). It is unclear whether these gaps should be modeled or not. Remember we had some criticism about the piece-wise linear model because it was not necessarily continuous. Well, reality is not continuous either.
- There also seems to be some kind of gap for count roughly equal to 1.4E03 (green vertical line). Is it real or is it due to the way the measurements were done ? It's unclear.

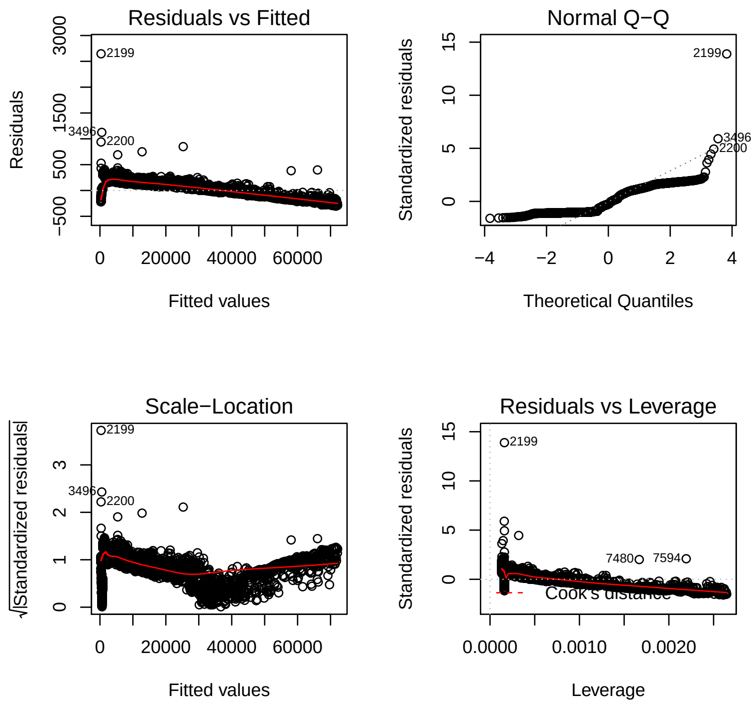


Figure 7: Graphical analysis of the quality of the linear regression.

- The time seems constant with some noise for count roughly smaller than 3E02. Even though the time is constant to 100 and then drops and goes up and down again. It is not clear that we would have the same drop if we performed the measurements again. Measurements should be randomized to make sure this is not an artifact.
- There seem to be a slight slope modification at count roughly equal 2.2E05
- As expected (this is obvious since it is perfectly linear in linear scale), the end of the plots has a slope of 1.

Analyzing in log scale is not easy and is made slightly more painful because the initial sampling is uniform. It makes the identification of slopes (hence the speed of slow-start) more difficult. Ideally, I think we should rather use an exponential random sampling.

2.1 Conclusion

I'd love to have the same kind of measurements but where log of message size is uniformly sampled. E.g., say we want message sizes to be between 1 byte and 10Mbytes, we would repeatedly send messages of size $exp(rand(1, 10^7))$. This require the sender and the receiver to use the same seed and the same random generation algorithm so that both sender and receiver declare the same count in `MPI_Send` and `MPI_Recv`. I'm not saying there will be a miracle and it will solve every issue. This is just my intuition telling me we could see something interesting with such measurements and get rid of potential measurement artifacts.

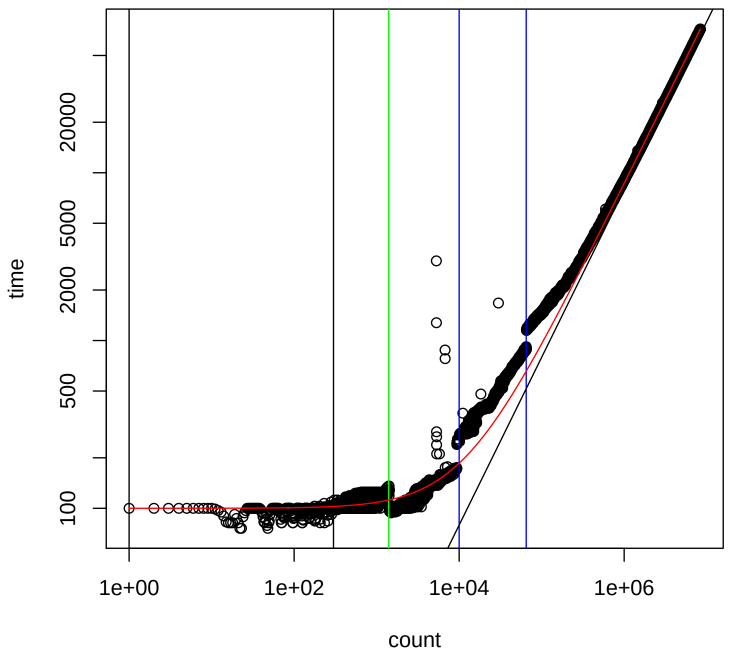


Figure 8: Communication time (time) versus data size (count) in logarithmic scale.

3 Exploratory Analysis of plogp Measurements by Stéphane Génaud (10/2008)

3.1 Setup

In this section, I analyze the measurements performed by Stéphane Génaud on Griffon using plogp (by Bal and Kielman) in October 2008. The Latency is constant and computed only for 0-length messages. The other parameters (o_r, o_s and g) are computed automatically by the logp program. The analysis was done on 23/01/12 while visiting Martin at Nancy.

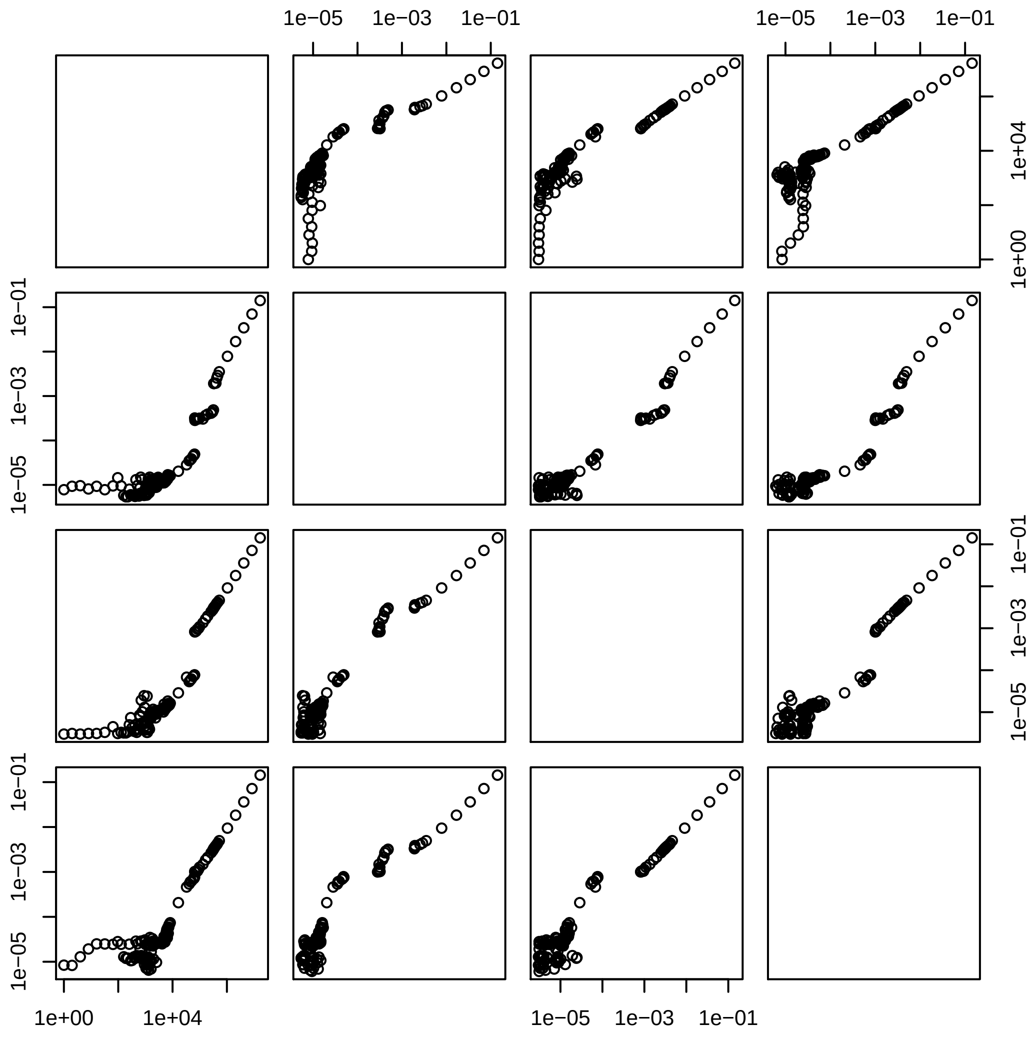
Let's put it in a suitable format:

```
grep -v '#' logp_test-griffon-1switch.Send.Recv.eth0 | sed -e 's/  */ /g' -e 's/://g' | cut -d ' ' -
```

```
> df <- read.table("logp_test-griffon-1switch.dat")
> names(df) <- c("size", "os", "or", "g")
```

Let's check for general shape in log-scale (the damn `log="xy"` option removes the labels and I don't know how to put them back but it's the usual order "size", "os", "or", "g").

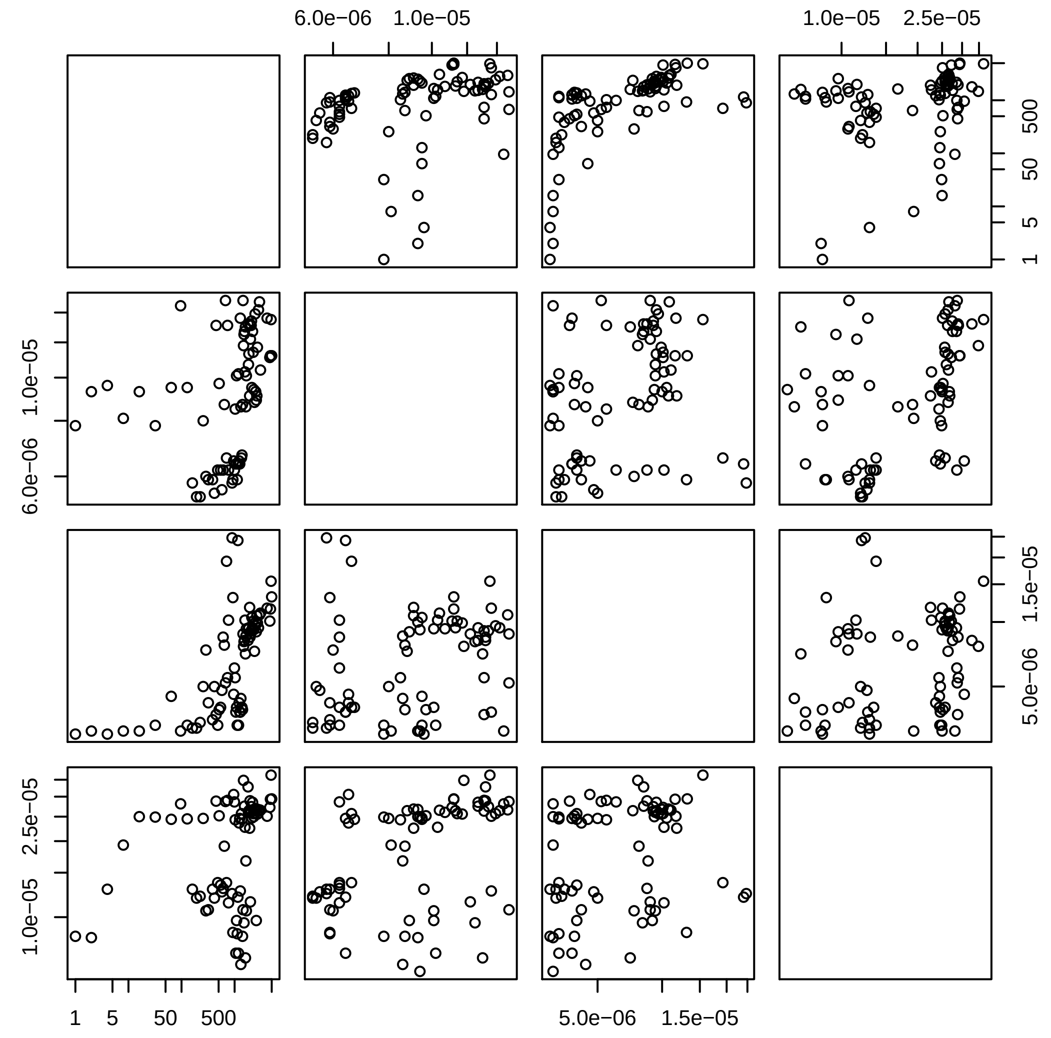
```
> plot(df, log = "xy")
```



There seem to be at least two areas. On that seems quite linear and another one that seems constant and noisy. It seems that the “noisy” part was over-sampled, which could explain this noise feeling.

Let’s zoom.

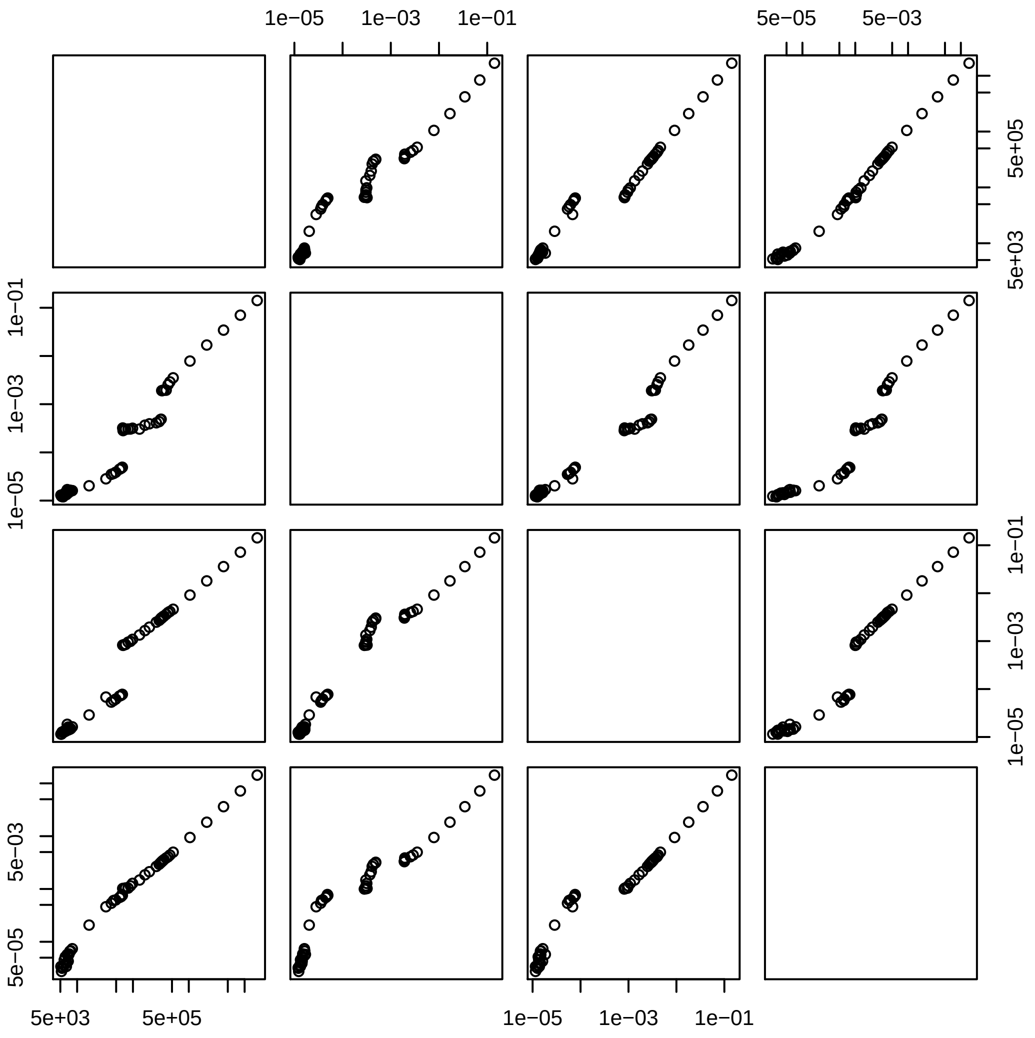
```
> plot(df[df$size < 5000, ], log = "xy")
```



Wow. It is very noisy indeed and I'm not sure I would trust a deterministic model with such values (remember that all these values are already ageraged values from measurements and are meant to be used to instantiate the LogP model). I don't know if such numbers would be reproducible (i.e., would we get the same numbers if we ran the experiments again today) but if so, trying to account for this seems hopeless to me. I think it would be better to have a stochastic model.

Let's look at larger messages.

```
> plot(df[df$size > 5000, ], log = "xy")
```



It seems much more regular, but with discontinuities and slope variations (could be something like a protocol change).

4 Exploratory Analysis of SkaMPI Measurements by Arnaud Legrand (23/01/12)

In this section, I analyze the measurements performed by Arnaud Legrand on Griffon using SkaMPI:

```
mpirun --mca btl self,tcp -machinefile machinefile ./skampi -i ski/skampi_pt2pt_stephane.ski -o grif
```

This command was lauched twice in a row, hence two measurement files. The ski file was:

```
set_min_repetitions(32)
set_max_repetitions(64)
set_max_relative_standard_error(0.03)
set_skampi_buffer(32768kb)
```

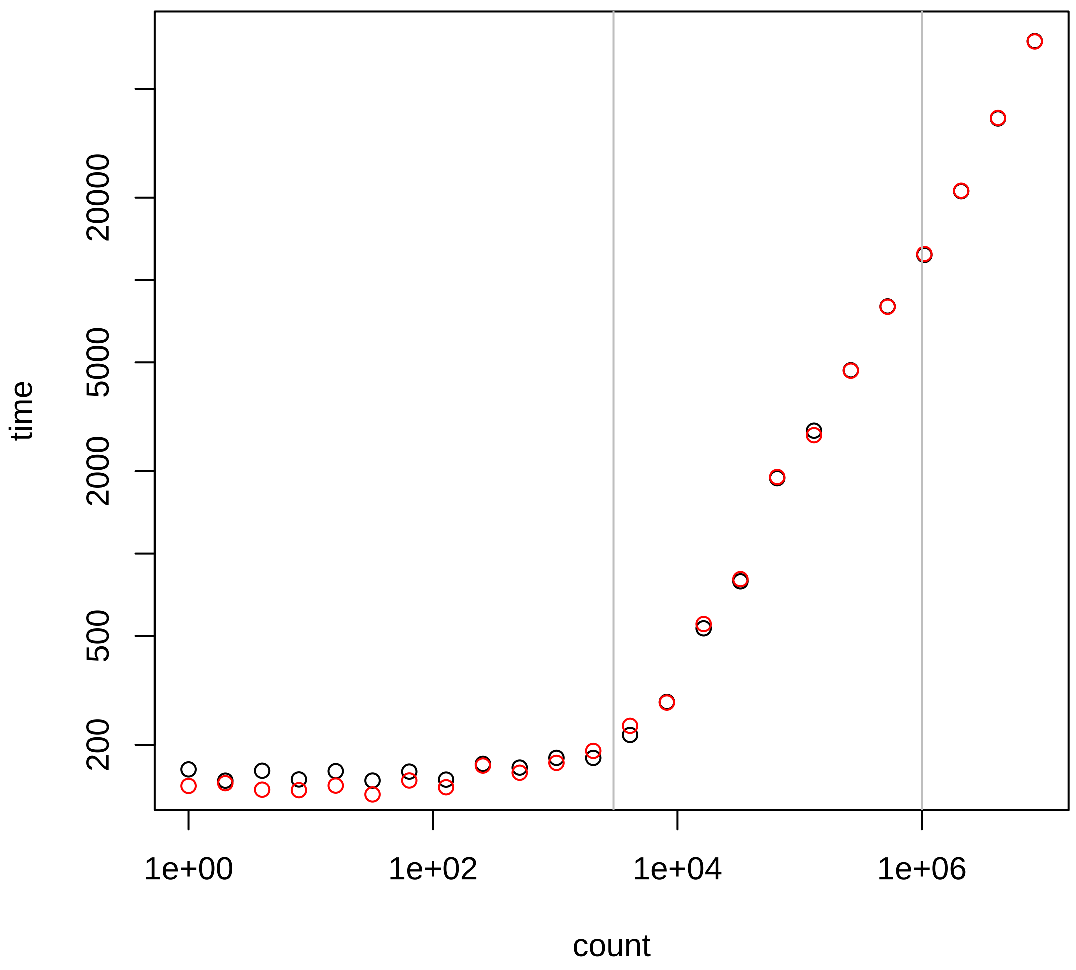
```
datatype = MPI_CHAR
comm_pt2pt = comm2_max_latency_with_root()

begin measurement "Pingpong_Send_Recv"
   for count = 1 to ... step *2  do
      measure comm_pt2pt : Pingpong_Send_Recv(count, datatype, 0, 1)
   od
end measurement
```

The bandwidth for large messages is around 112 Mbytes/s, which is reasonable for TCP over a 1GB Ethernet but does not compare to the 57.8 Mbytes/s measured by Stéphane on the same machine in July 2010 (see Section 1). I don't know why.

```
> df <- read.table("griffon_skampi_pt2pt.2012-01-23.18:34:25.dat")
> df2 <- read.table("griffon_skampi_pt2pt.2012-01-23.18:35:16.dat")
> names(df) <- c("count", "countn", "time", "stddev", "iter", "mini", "maxi")
> names(df2) <- names(df)
> df <- df[!(names(df) %in% c("countn", "iter", "mini", "maxi"))]
> df2 <- df2[!(names(df2) %in% c("countn", "iter", "mini", "maxi"))]
> df <- df[(names(df) %in% c("count", "time"))]
> df2 <- df2[(names(df2) %in% c("count", "time"))]
```

Let's plot the two measurements on the same graph in log scale.

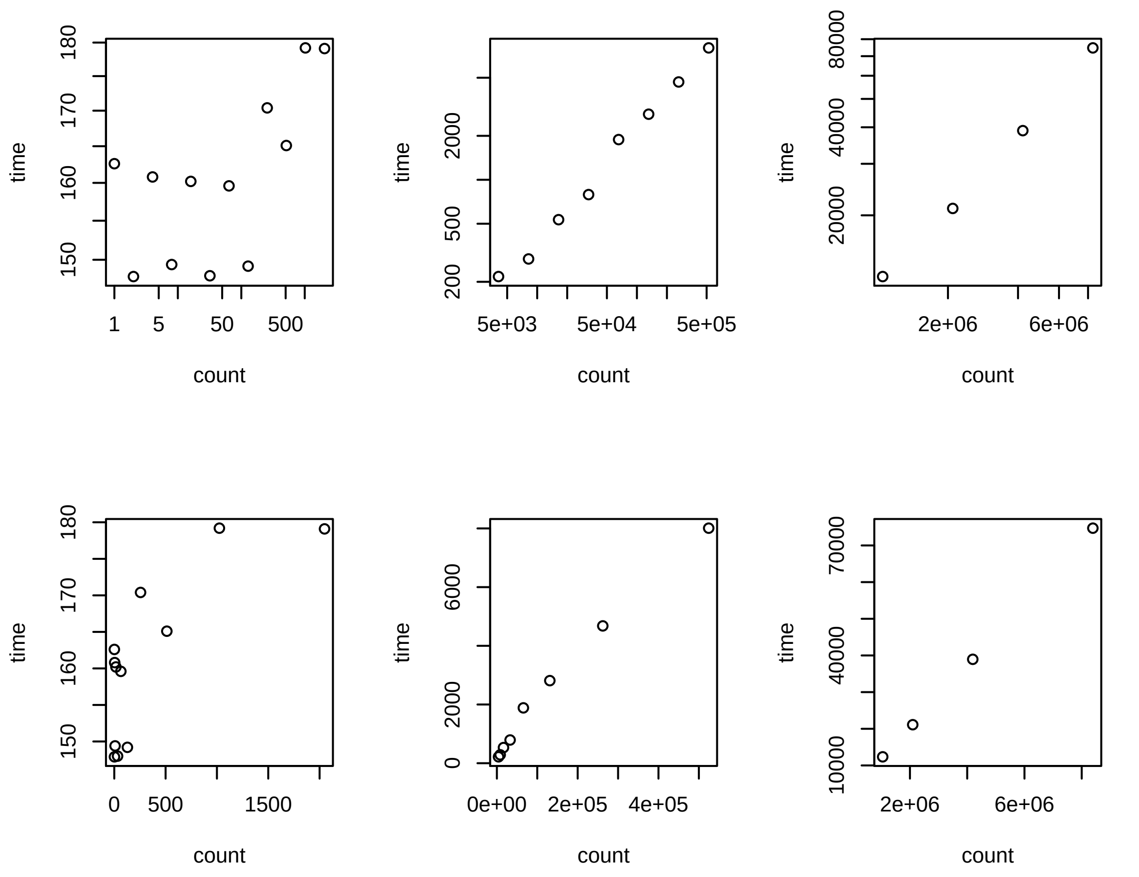
```
> plot(df, log = "xy")
> points(df2, log = "xy", col = "red")
> abline(v = 3000, col = "gray")
> abline(v = 1e+06, col = "gray")
```



Note that each measurement is already an average of 64 samples. There is still some noise (which is not surprising). That's why instead of having the average timing, it would be much better to have the original measurements and work directly with it. This would enable us to model the noise to account for. Interested users can have a look at Figure 1(d) of `http://www.cs.uiuc.edu/~wgropp/bib/papers/1999/pvmmpi99/mpptest.pdf`.

Let's look at the 3 different ranges in more detail.

```
> par(mfrow = c(2, 3))
> df_short <- df[df$count < 3000, ]
> df_medium <- df[df$count > 3000 & df$count < 1e+06, ]
> df_large <- df[df$count > 1e+06, ]
> plot(df_short, log = "xy")
> plot(df_medium, log = "xy")
> plot(df_large, log = "xy")
> plot(df_short)
> plot(df_medium)
> plot(df_large)
```



4.1 Short messages

```
> lm_short ← lm(log10(time) ~ log10(count), data = df_short)
> summary(lm_short)
```

```
Call:
lm(formula = log10(time) ~ log10(count), data = df_short)

Residuals:
      Min         1Q    Median        3Q       Max
-0.039951  -0.013464  0.000743  0.018385  0.034554

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.17657    0.01316  165.45   <2e-16 ***
log10(count)  0.01763    0.00673    2.62   0.0256 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02423 on 10 degrees of freedom
Multiple R^2: 0.407,     Adjusted R^2: 0.3477
F-statistic: 6.863 on 1 and 10 DF,  p-value: 0.02561
```

The slope is very close to 0. This means it's almost constant and it may not be worth considering it. Note that we cannot conclude anything from such measurements (we should use the original ones, not averaged values to discriminate real trends from noise). Anyway, given the range, constant plus some noise should be a pretty good model.

```
> lm_short <- lm(time ~ 1, data = df_short)
> summary(lm_short)
```

```
Call:
lm(formula = time ~ 1, data = df_short)

Residuals:
     Min       1Q   Median       3Q      Max
-13.0583 -11.6083  -0.4583   5.4667  18.2417

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  160.958      3.238   49.72 2.68e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.22 on 11 degrees of freedom
```

To get a good instantiation of this model, we should perform a uniform sampling of this range. We should also be careful that the noise is probably not random.

4.2 Large messages

It looks pretty linear. Is the slope 1 in log space ?

```
> lm_large_log <- lm(log10(time) ~ log10(count), data = df_large)
> summary(lm_large_log)
```

```
Call:
lm(formula = log10(time) ~ log10(count), data = df_large)

Residuals:
      21       22       23       24
 0.01329 -0.01488 -0.01010  0.01169

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.14424    0.17178  -6.661 0.021804 *
log10(count)  0.86746    0.02651  32.727 0.000932 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01784 on 2 degrees of freedom
Multiple R^2: 0.9981,    Adjusted R^2: 0.9972
F-statistic:  1071 on 1 and 2 DF,  p-value: 0.0009324
```

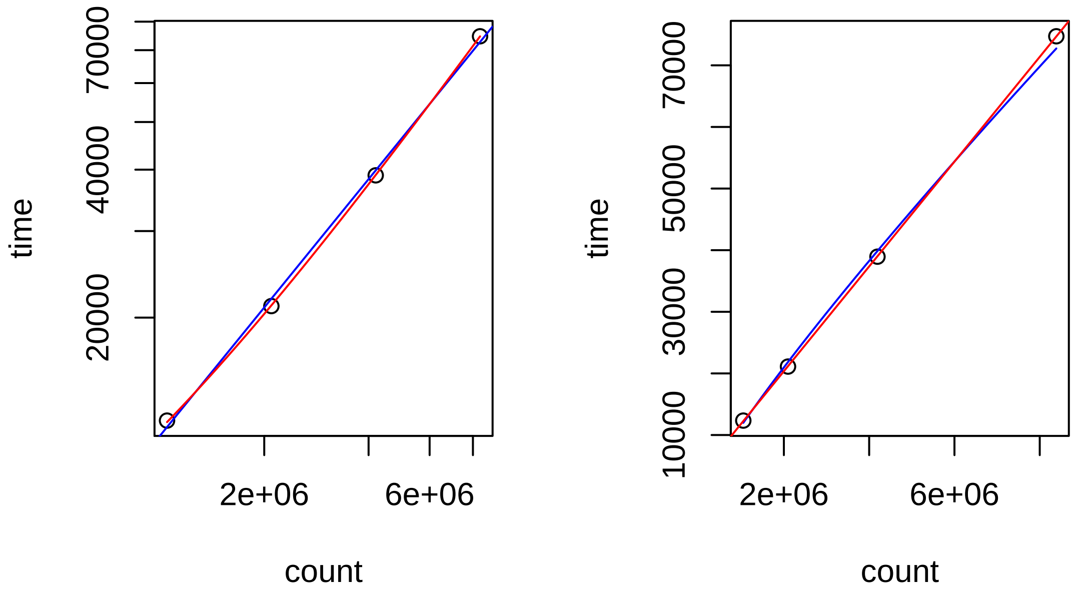
The slope is 0.86746, which is close to one but not that much... Let's look at the true linear regression:

```
> lm_large <- lm(time ~ count, data = df_large)
> summary(lm_large)
```

```
Call:
lm(formula = time ~ count, data = df_large)

Residuals:
    21     22     23     24
 89.09 -62.41 -62.30  35.61

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.343e+03  7.922e+01   42.19 0.000561 ***
count       8.505e-03  1.639e-05  518.93 3.71e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 92.14 on 2 degrees of freedom
Multiple R^2:      1,    Adjusted R^2:      1
F-statistic: 2.693e+05 on 1 and 2 DF,  p-value: 3.713e-06
```

Wow. That's just perfect. Let's plot it (linear regression in red and log-linear regression in blue).

```
> par(mfrow = c(1, 2))
> plot(df_large, log = "xy")
> abline(lm_large_log, col = "blue")
> curve(coefficients(lm_large)[1] + coefficients(lm_large)[2] * x, col = "red",
+     add = T)
> plot(df_large)
> curve(10^coefficients(lm_large_log)[1] * x^coefficients(lm_large_log)[2], col = "blue",
+     add = T)
> abline(lm_large, col = "red")
```



The standard linear regression (red) is just great whereas the logarithmic one (blue) is not that good. So we should use a simple linear model for this range.

4.3 Medium Messages

```
> lm_medium_log <- lm(log10(time) ~ log10(count), data = df_medium)
> summary(lm_medium_log)
```

```
Call:
lm(formula = log10(time) ~ log10(count), data = df_medium)

Residuals:
      Min        1Q    Median        3Q       Max
-0.074790 -0.022556 -0.002006  0.023176  0.069232

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.52172    0.12897  -4.045  0.00676 **
log10(count)  0.77403    0.02734  28.308 1.29e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05334 on 6 degrees of freedom
Multiple R^2: 0.9926,    Adjusted R^2: 0.9913
```

```
F-statistic: 801.3 on 1 and 6 DF,  p-value: 1.286e-07
```

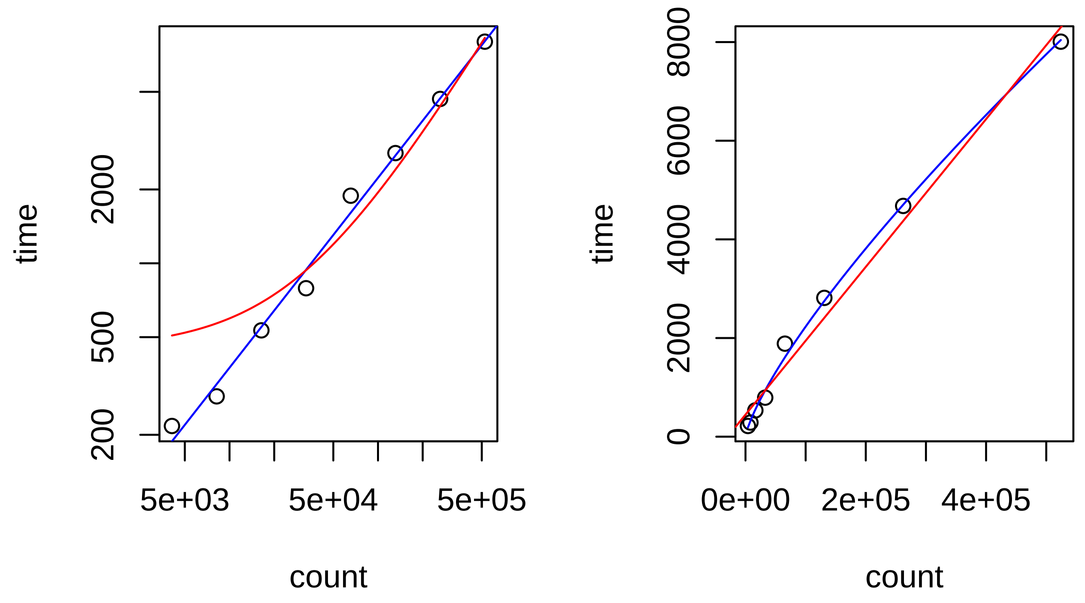
The slope is 0.77403, which is getting far from 1, especially for medium messages. look at the true linear regression:

```
> lm_medium <- lm(time ~ count, data = df_medium)
> summary(lm_medium)
```

```
Call:
lm(formula = time ~ count, data = df_medium)

Residuals:
   Min      1Q  Median      3Q     Max
-290.6  -284.4  -152.3   330.2   458.0

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 4.466e+02  1.590e+02   2.809   0.0308 *
count       1.498e-02  7.427e-04  20.169 9.65e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 356.3 on 6 degrees of freedom
Multiple R^2: 0.9855,    Adjusted R^2: 0.983
F-statistic: 406.8 on 1 and 6 DF,  p-value: 9.649e-07
```

Of course, the R^2 is good but the prediction is not really good for small messages. This residual does not characterize the noise but the non-linearity so it should not be taken into account. As can be seen on these plots, the logarithmic linear regression is pretty good.

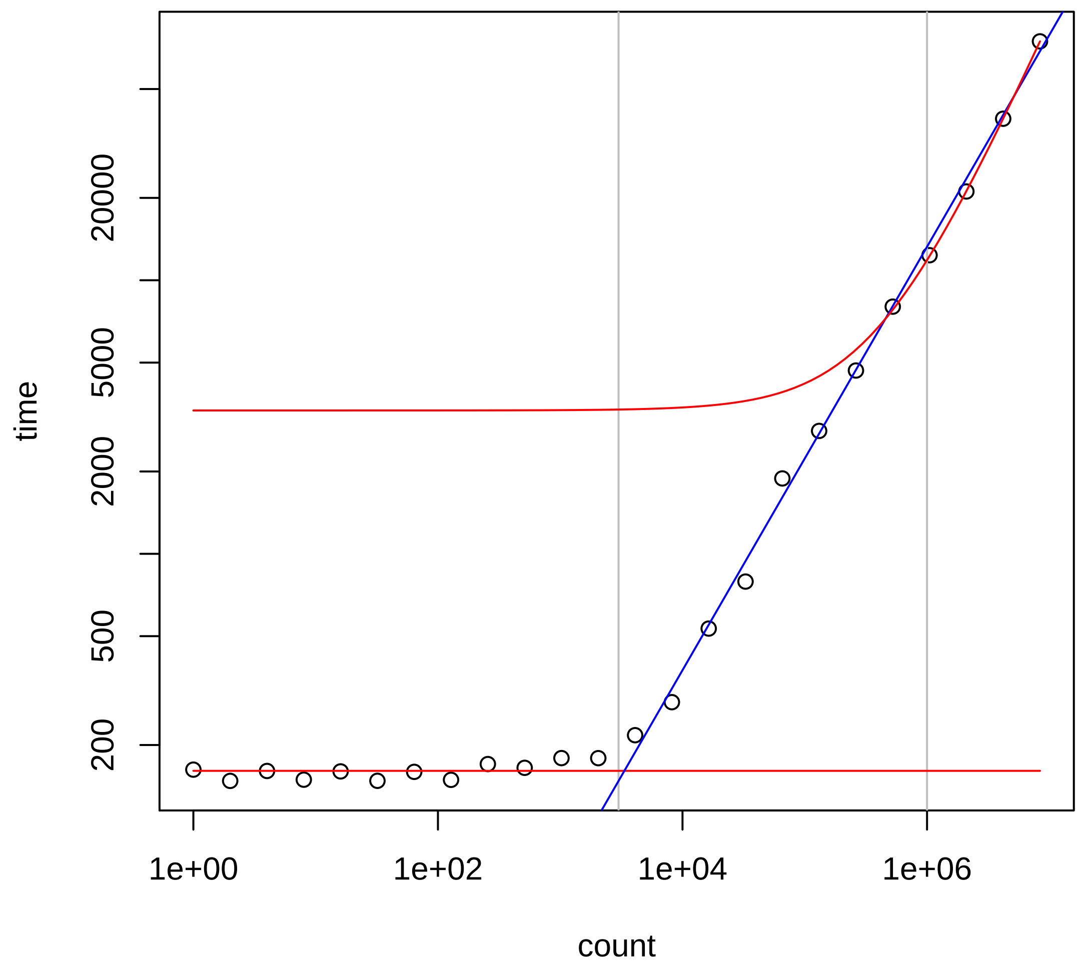
```
> par(mfrow = c(1, 2))
> plot(df_medium, log = "xy")
> abline(lm_medium_log, col = "blue")
> curve(coefficients(lm_medium)[1] + coefficients(lm_medium)[2] * x, col = "red",
+     add = T)
> plot(df_medium)
> curve(10^coefficients(lm_medium_log)[1] * x^coefficients(lm_medium_log)[2],
+     col = "blue", add = T)
> abline(lm_medium, col = "red")
```



This corresponds to the fact that there is some kind of slow start and the protocol is “speeding up”. It is not linear and piecewise linear models will not capture this. A logarithmic linear regression seems good, and an exponential sampling of the range should be used to get a good regression.

4.4 Recap

```
> par(mfrow = c(1, 1))
> plot(df, log = "xy")
> abline(v = 3000, col = "gray")
> abline(v = 1e+06, col = "gray")
> curve(coefficients(lm_short)[1] + 0 * x, col = "red", add = TRUE)
> abline(lm_medium_log, col = "blue")
> curve(coefficients(lm_large)[1] + coefficients(lm_large)[2] * x, col = "red",
+     add = T)
```



This analysis should be conducted again with all measurements (instead of aggregates) but it's not that obvious to do with SkaMPI (there would be synchronizations automatically inserted between each Send/Recv). An exponential sampling should be used to derive the model and find adequate range. It may reveal gaps or protocol changes that advocate for either fewer and more intervals to consider. Then for each interval we should check whether a standard linear regression should be used or whether a preliminary logarithmic transformation should be done.

Note that this would advocate for the following model:

- For small messages, it is constant plus some noise (probably heavy tailed but that should be characterized)
- For large messages, it is linear (as in the current SURF model). A uniform sampling should be used on this range to double-check the potential lack of fit. Then sampling solely at both ranges of the interval may produce the better result for instantiating the model.
- For medium messages, using a linear model does not seem like a good solution. A delay is well modeled by $\alpha.size^{\beta}$. Such a delay is easy to account for (just like for small messages) but this mean that the potential interference small/medium messages and large messages is not taken into account. Would it be harmful ? We need to set up experiments to check

this. Interestingly, such a model would speed up simulations since the bandwidth sharing of large messages would not be disrupted by small messages. This may look a bit extreme but disrupting all large messages and sharing bandwidth under the steady-state assumption is actually not sounder than this...

5 Zoo: Communication time classification

As we have seen previously, communication time is not linear with message size. Yet previous experiments did aggregate timing and measurements, which may bias results and hinders statistical analysis. Martin and Arnaud have set up an easy experimental framework based on coNCePTuaL (for describing the experiment to perform), akypuera (for tracing the experiment), Paje (to visualize the corresponding trace), and R (for performing the analysis). coNCePTuaL makes a bunch of initial measurements at first to obtain information on the machines and synchronize clocks. We also run `rastrotimesync` before and after launching coNCePTuaL so as to possibly resynchronize traces. akypuera traces are then merged into a Paje trace that can be either visualized or converted to a simple text file easily readable by R using a simple perl script. The timer resolution is around 1E-6.

5.1 Experiment description

We set up coNCePTuaL with two MPI process on separate machines from the griffon cluster, using tcp (and disabling infiniband). Process 0 sends N messages of size 10^x with x a random variable in $U(0,7)$. There is no message from process 1 to process 0 except for synchronization at the beginning and at the end.

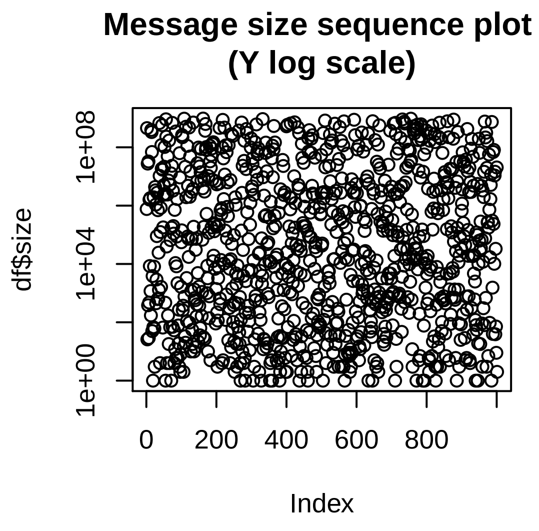
5.2 Exploratory Analysis

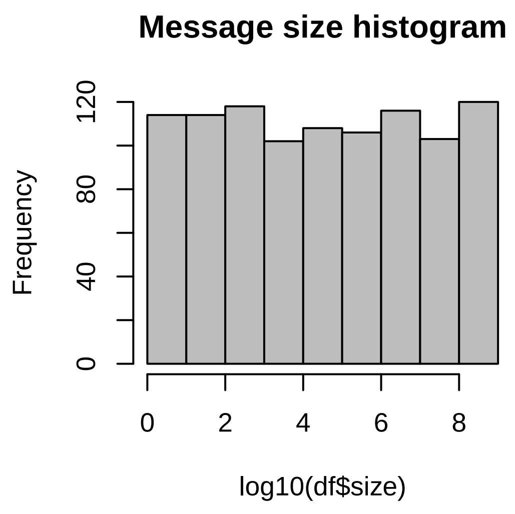
```
> df <- read.table("zoo_small.0.dat")
> names(df) <- c("size", "type0", "start0", "end0", "type1", "start1", "end1")
> df <- df[!(names(df) %in% c("type1"))]
> df <- df[df$type0 == "MPI_Send", ]
> df <- df[!(names(df) %in% c("type0"))]
> df$send <- df$end0 - df$start0
> df$recv <- df$end1 - df$start1
> df$comm <- df$start1 - df$end0
> df <- df[!(names(df) %in% c("start0", "end0", "start1", "end1"))]
```

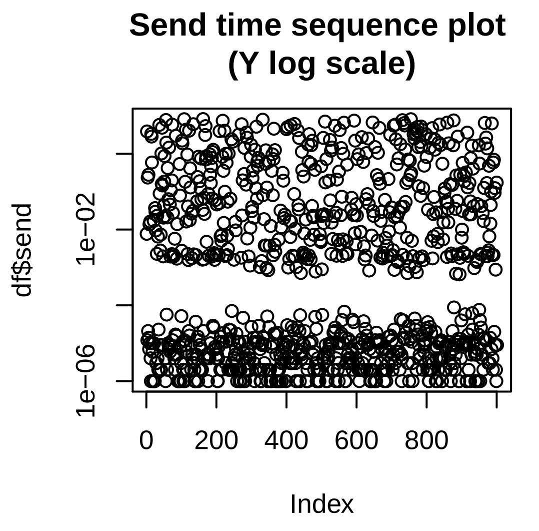
Note that, computed this way, `df$comm` has no meaning since receives are often posted before sends, especially under such a random workload. Since input workload is uniform in log space, most views should be done in log scale.

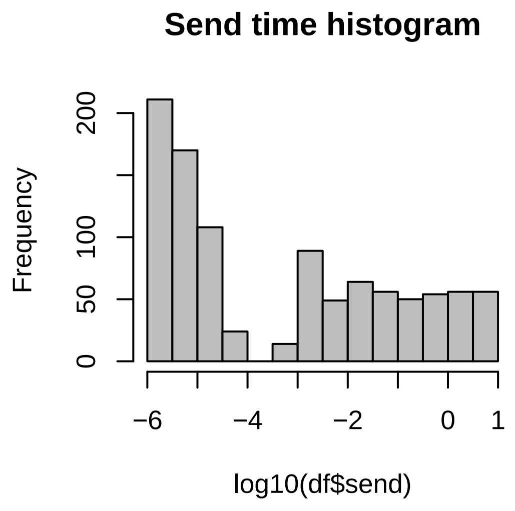
Let's check a few things

```
> par(mfrow = c(2, 2))
> plot(df$size, main = "Message size sequence plot \n(Y log scale)", log = "y")
> hist(log10(df$size), main = "Message size histogram", col = "gray")
> plot(df$send, main = "Send time sequence plot \n(Y log scale)", log = "y")
> hist(log10(df$send), main = "Send time histogram", col = "gray")
```

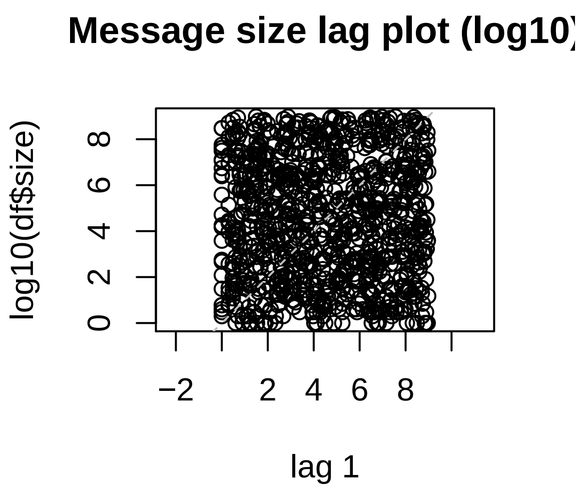


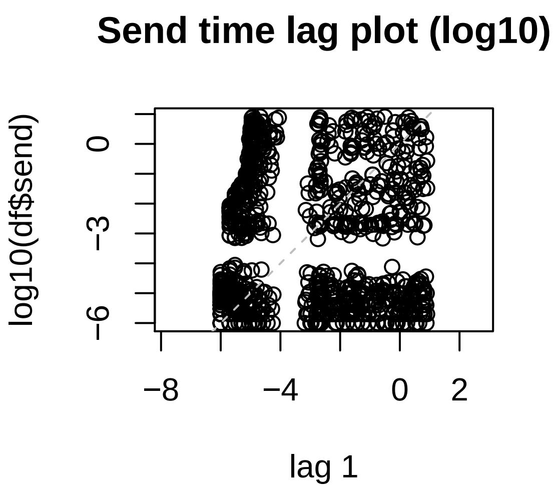






The Lag plot, is an easy way to check whether the i-th value depends on the $i-1$-th value.

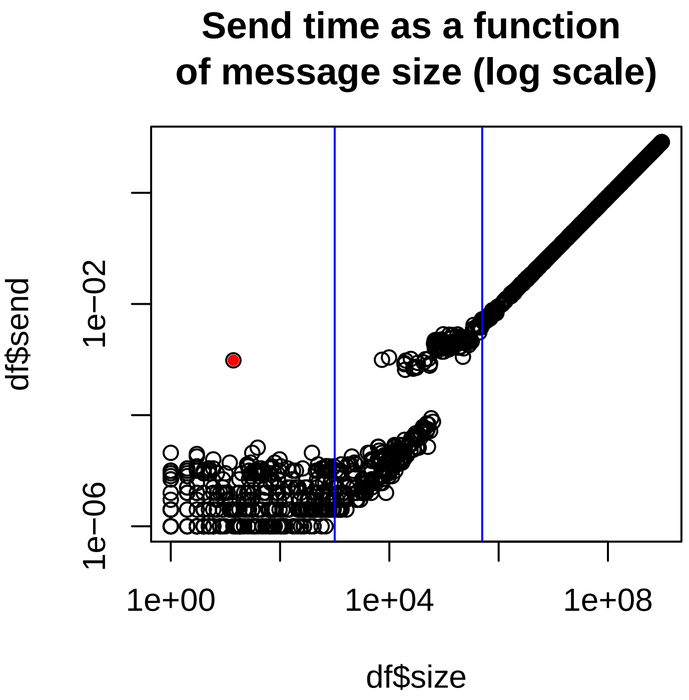




As expected, it is uniform for message size (this is how we generated it). Regarding send time, there is no obvious bias. It looks almost like the product of the distributions of the send time histogram, except maybe for the upper left "rectangle" that is not well rectangle-shaped.

Now, let's try to look at the relation between `size` and `send`.

```
> plot(df$size, df$send, log = "xy", main = "Send time as a function \nof message size (log scale)")
> outlier = df[df$size < 100 & df$send > 1e-04, ]
> points(outlier$size, outlier$send, col = "red", pch = 20)
> small_boundary = 1000
> medium_boundary = 5e+05
> abline(v = small_boundary, col = "blue")
> abline(v = medium_boundary, col = "blue")
```



There is at least one outlier, which I propose to remove to help analysis in the sequel.

```
> df ← df[!(df$size < 100 & df$send > 1e−04), ]
```

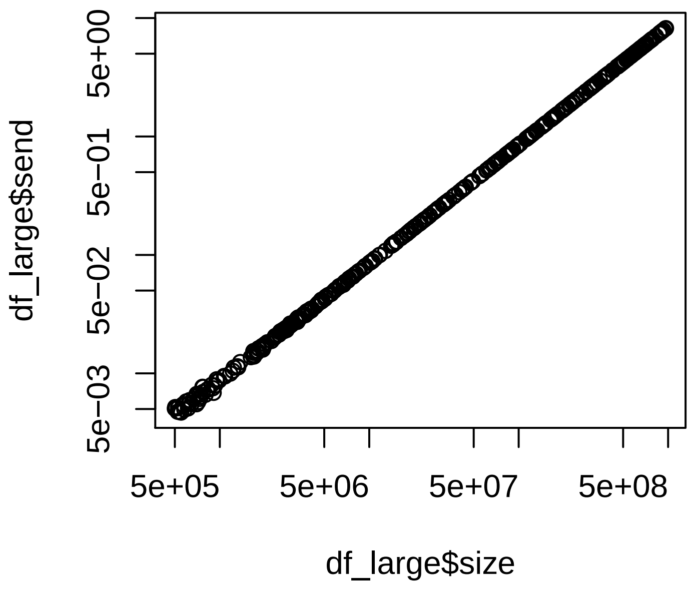
We can still distinguish between small, medium and large message although the behavior of medium messages seems quite strange.

```
> df_small ← df[df$size < small_boundary, ]
> df_medium ← df[df$size ≥ small_boundary & df$size < medium_boundary, ]
> df_large ← df[df$size > medium_boundary, ]
```

Large messages

```
> plot(df_large$size, df_large$send, log = "xy")
> summary(lm(log10(send) ~ log10(size), data = df_large))
```

```
Call:
lm(formula = log10(send) ~ log10(size), data = df_large)

Residuals:
      Min         1Q     Median         3Q        Max
-0.063999  -0.003956  -0.000676   0.002957   0.070706

Coefficients:
              Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  -8.0239321  0.0047010    -1707    <2e-16 ***
log10(size)   0.9942388  0.0006322     1573    <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0119 on 374 degrees of freedom
Multiple R^2: 0.9998,    Adjusted R^2: 0.9998
F-statistic: 2.473e+06 on 1 and 374 DF,  p-value: < 2.2e-16
```



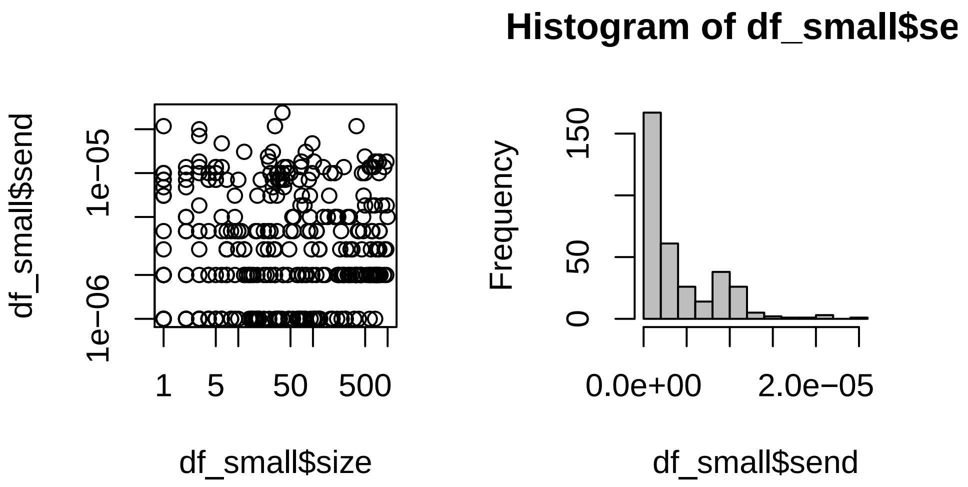
Sweet!

Small messages

```
> summary(lm(log10(send) ~ log10(size), data = df_small))
```

```
Call:
lm(formula = log10(send) ~ log10(size), data = df_small)

Residuals:
     Min       1Q   Median       3Q      Max
-0.53505 -0.43468 -0.05592  0.35661  0.92392

Coefficients:
             Estimate Std. Error  t value Pr(>|t|)
(Intercept) -5.56532    0.04168 -133.520   <2e-16 ***
log10(size)  0.03543    0.02330    1.521    0.129
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3891 on 343 degrees of freedom
Multiple R^2: 0.006696,  Adjusted R^2: 0.0038
F-statistic: 2.312 on 1 and 343 DF,  p-value: 0.1293
```

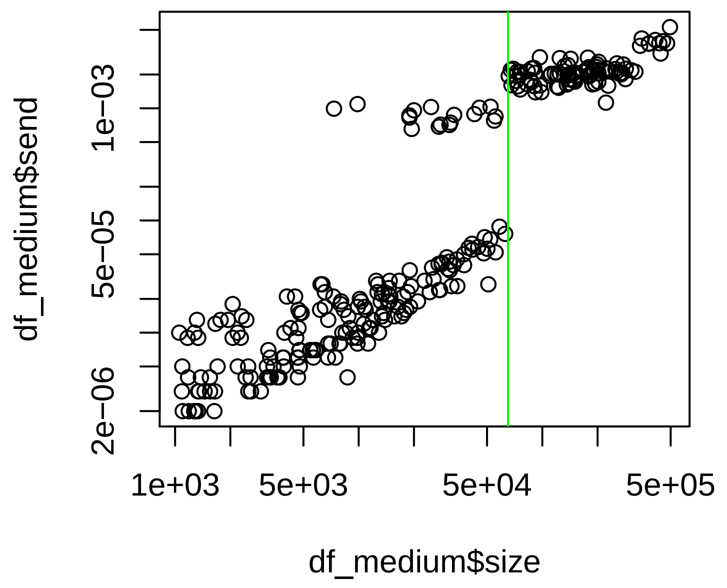
```
> par(mfrow = c(1, 2))
> plot(df_small$size, df_small$send, log = "xy")
> hist(df_small$send, , col = "gray")
```



As expected, size has no influence. We are close to the timer resolution but the time it takes is quite heavy tailed...

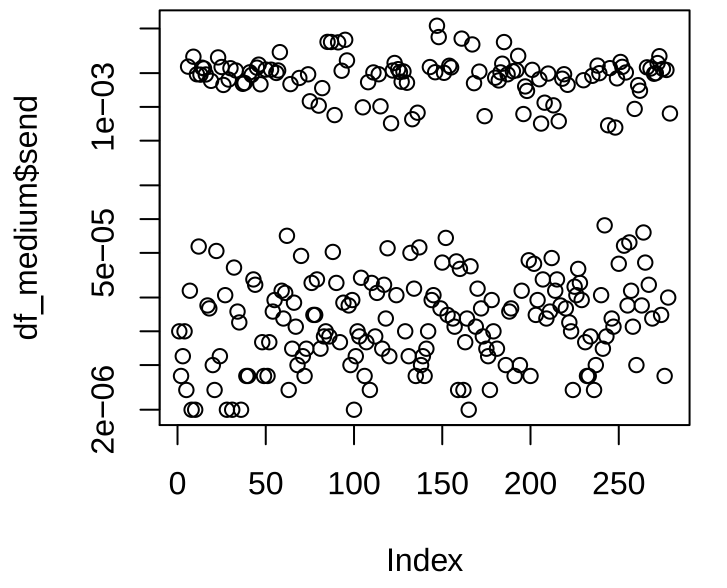
Medium messages

```
> plot(df_medium$size, df_medium$send, log = "xy")
> abline(v = 65000, col = "green")
```



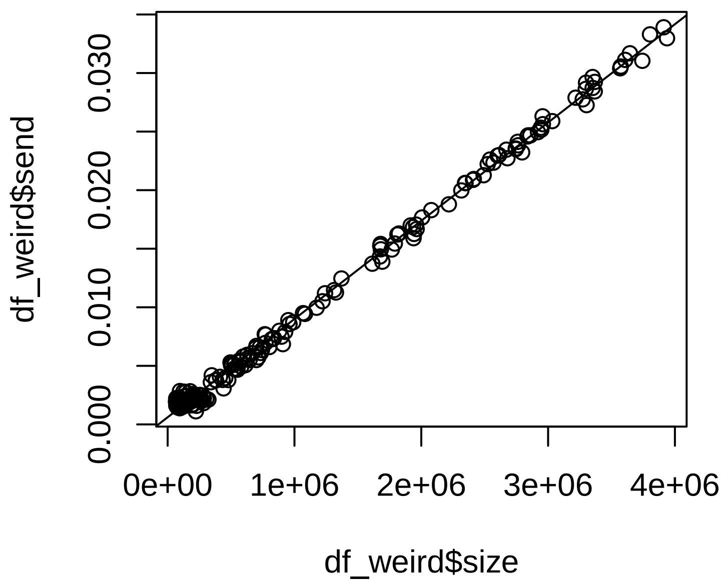
Well, that does not look good and it's clearly not continuous... Let's check that the behavior did not change over time (we already checked before but not in this range):

```
> plot(df_medium$send, log = "y")
```



Well, no. There is a strange and ugly gap for message size larger than 6.5E4. Actually, they could be incorporated to large messages:

```
> df_weird <- df[df$size > 65000 & df$size < 4e+06, ]
> plot(df_weird$size, df_weird$send)
> abline(lm(send ~ size, data = df_weird))
```

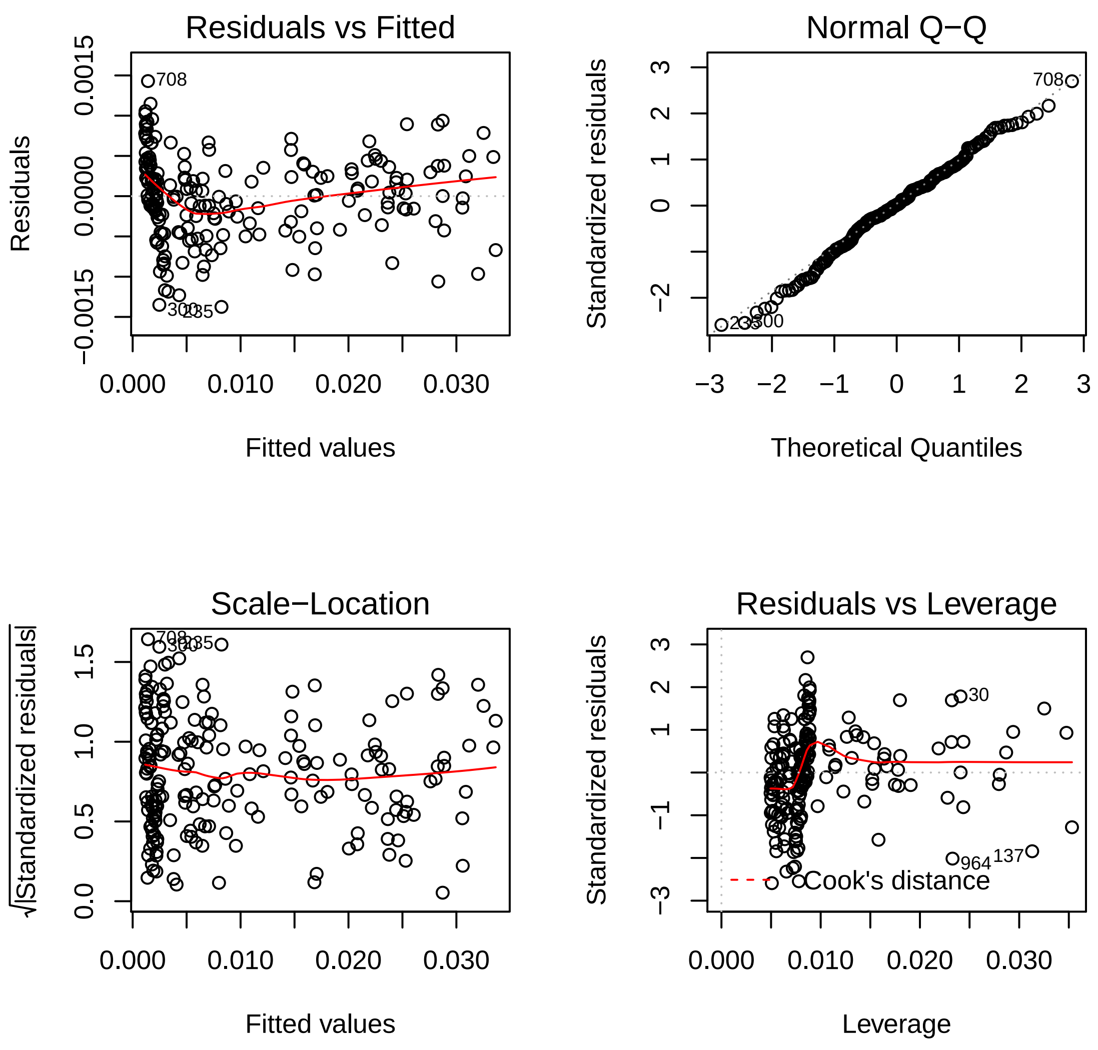


The linear regression is actually great although it could be redone with a different sampling:

```
> summary(lm(send ~ size, data = df_weird))
```

```
Call:
lm(formula = send ~ size, data = df_weird)

Residuals:
       Min         1Q     Median         3Q        Max
-1.375e-03 -3.171e-04  9.480e-06  3.600e-04  1.430e-03

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 6.093e-04  5.172e-05   11.78   <2e-16 ***
size        8.390e-09  3.262e-11  257.19   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0005324 on 200 degrees of freedom
Multiple R^2: 0.997,     Adjusted R^2: 0.997
F-statistic: 6.615e+04 on 1 and 200 DF,  p-value: < 2.2e-16
```

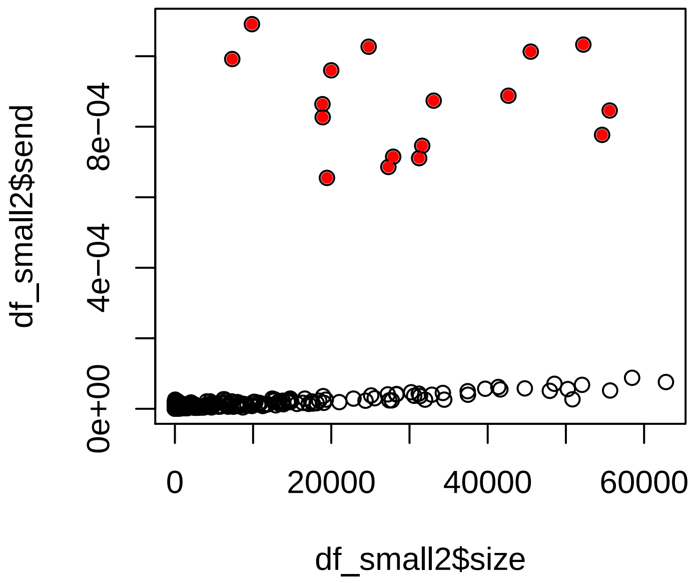
```
> par(mfrow = c(2, 2))
> plot(lm(send ~ size, data = df_weird))
```



The dispersion is nice. Perfect linear model for this range.

Not that small messages Since we have promoted half of medium messages to the large category, let's try to see whether the other half can go with the small ones.

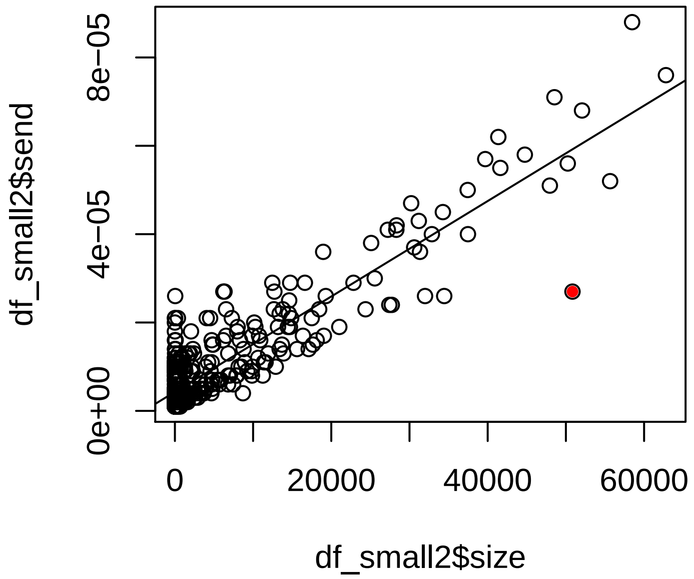
```
> df_small2 ← df[df$size < 65000, ]
> plot(df_small2$size, df_small2$send)
> outlier = df_small2[df_small2$send > 4e−04, ]
> points(outlier$size, outlier$send, col = "red", pch = 20)
```



Well, there are a few points with quite a strange behavior. I isolated them and could not find anything they had in common (e.g., when they were sent or whether the previous message sizes had a particular size). Further measurements could be done in this range but I don't know what happened for these 17 points.

Again, since their behavior seems very different from the others, I propose to remove them

```
> df_small2 <- df[df$size < 65000 & df$send < 4e-04, ]
> plot(df_small2$size, df_small2$send)
> abline(lm(send ~ size, data = df_small2))
> outlier = df_small2[df_small2$size > 40000 & df_small2$send < 4e-05, ]
> points(outlier$size, outlier$send, col = "red", pch = 20)
```



The linear regression looks pretty good although there is a lot of noise that needs to be incorporated in the model.

```
> df_small2 ← df_small2[!(df_small2$size > 40000 & df_small2$send < 4e−05), ]
> plot(df_small2$size, df_small2$send)
> abline(lm(send ∼ size, data = df_small2))
```

For small messages, it looks like it is roughly linear but the noise really matters. And the parameters of the slope and intercept are also quite different.

```
> summary(lm(send ∼ size, data = df_small2))
```

```
Call:
lm(formula = send ∼ size, data = df_small2)

Residuals:
       Min          1Q      Median          3Q         Max
−1.653e−05  −3.223e−06  −2.002e−06   3.337e−06   2.174e−05

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 4.219e−06  2.396e−07   17.61   <2e−16 ***
size        1.112e−09  2.306e−11   48.24   <2e−16 ***
−−−
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.967e−06 on 510 degrees of freedom
Multiple R^2: 0.8202,     Adjusted R^2: 0.8199
F−statistic:  2327 on 1 and 510 DF,  p−value: < 2.2e−16
```

5.3 A simple two-range piece-wise linear model

From what we just saw, we could go for a simple piece-wise linear model with just two ranges:

- Small messages (when message size is smaller than 6.5E4):

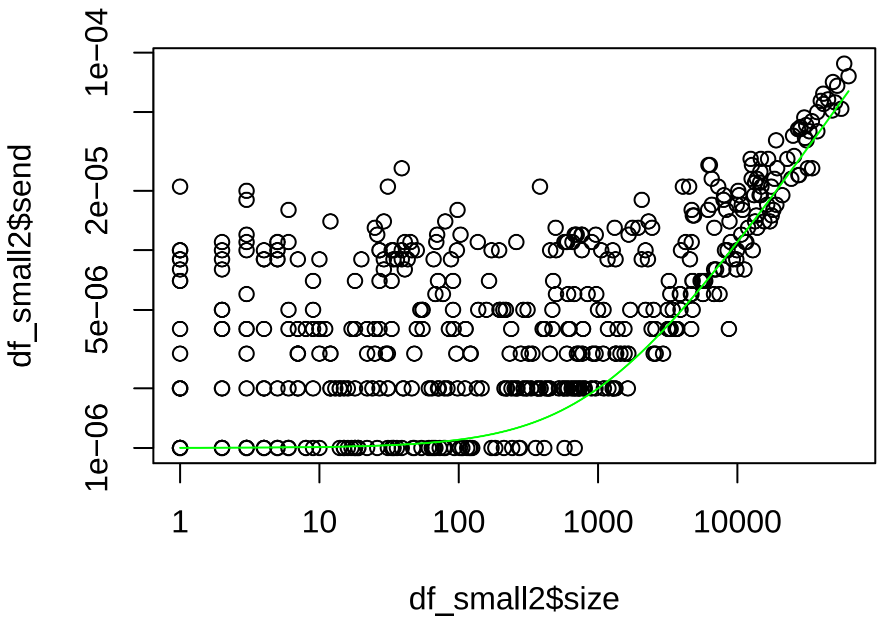
```
> lm_small ← lm(send ~ size , data = df_small2)
> summary(lm_small)
```

```
Call:
lm(formula = send ~ size , data = df_small2)

Residuals:
       Min          1Q      Median          3Q         Max
-1.653e-05  -3.223e-06  -2.002e-06   3.337e-06   2.174e-05

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 4.219e-06  2.396e-07   17.61   <2e-16 ***
size        1.112e-09  2.306e-11   48.24   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.967e-06 on 510 degrees of freedom
Multiple R^2: 0.8202,     Adjusted R^2: 0.8199
F-statistic:  2327 on 1 and 510 DF,  p-value: < 2.2e-16
```

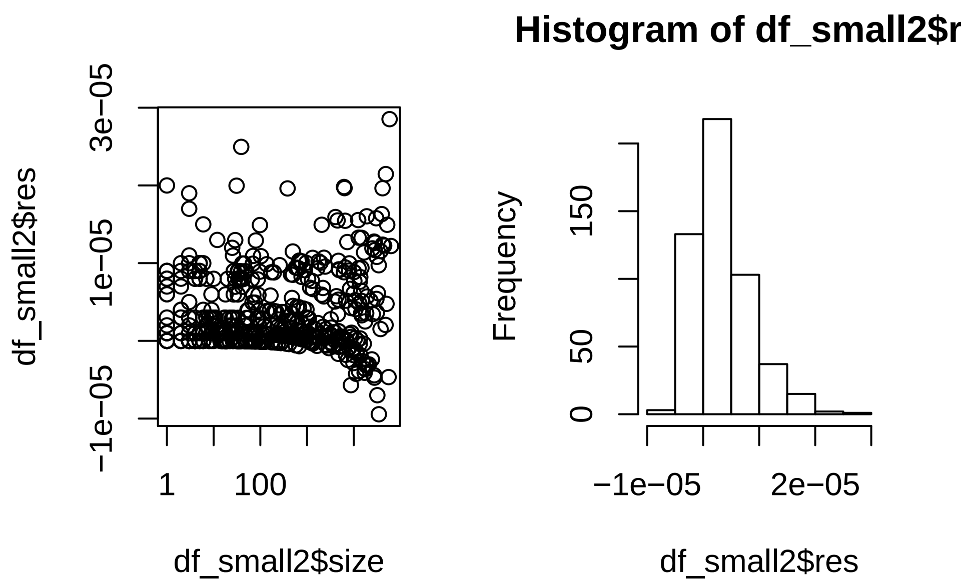
Small latency (4.21e-06) and 700% of bandwidth! This can be considered as very surprising at first sight but in such message size range, talking about bandwditdh does not really make sense anyway. Note that the way we perform the regression is not good since the assumption on the noise is that it is gaussion of mean 0. In our case it looks more like we have something linear plus some heavy-tailed strictly positive noise. I don't know how to handle this in a clean way but this cannot explain the "surprising" 700% of bandwidth.

A possible way to deal with this could be the following:

```
> plot(df_small2$size , df_small2$send , log = "xy")
> curve(1e-06 + x/1e+09, col = "green", add = TRUE)
```



```
> df_small2$res = df_small2$send - (1e-06 + df_small2$size/1e+09)
> par(mfrow = c(1, 2))
> plot(df_small2$size, df_small2$res, log = "x")
> hist(df_small2$res)
```



This way, we would have a base communication time plus some skewed noise. Anyway, before jumping to any such conclusion, we should perform new measurements in this range.

- Large messages (when message size is larger than 6.5E4):

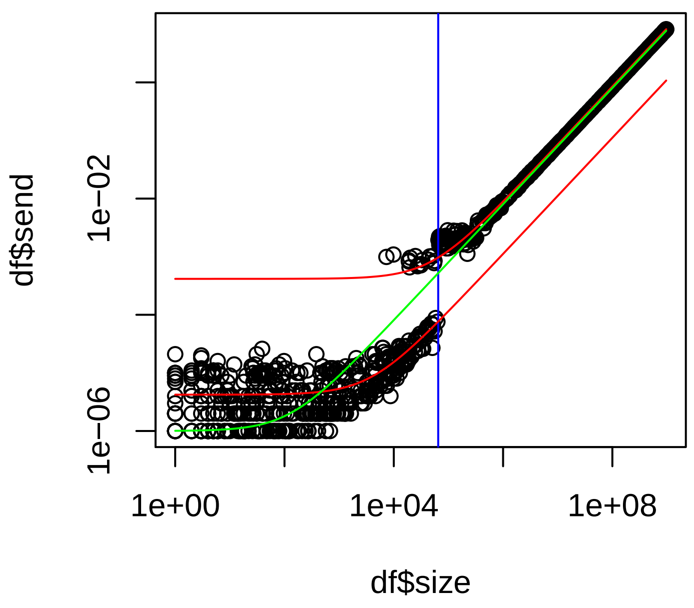
```
> boundary_small = 65000
> lm_large <- lm(send ~ size, data = df[df$size > boundary_small, ])
> summary(lm_large)
```

```
Call:
lm(formula = send ~ size, data = df[df$size > boundary_small,
    ])

Residuals:
       Min         1Q     Median         3Q        Max
-0.0017544 -0.0003514 -0.0000173  0.0002594  0.0142778

Coefficients:
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) 4.155e-04  4.341e-05     9.573   <2e-16 ***
size        8.497e-09  1.796e-13 47319.819   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0008302 on 468 degrees of freedom
Multiple R^2:     1,     Adjusted R^2:     1
F-statistic: 2.239e+09 on 1 and 468 DF,  p-value: < 2.2e-16
```

Large latency (4.15E-04) but 94% of bandwidth.

Now, let's step back (the green curve would be a dumb linear model using peak characteristics).

```
> plot(df$size, df$send, log = "xy")
> abline(v = 65000, col = "blue")
> curve(coefficients(lm_small)[1] + coefficients(lm_small)[2] * x, col = "red",
+     add = TRUE)
> curve(coefficients(lm_large)[1] + coefficients(lm_large)[2] * x, col = "red",
+     add = TRUE)
> curve(1e-06 + x * 8/1e+09, col = "green", add = TRUE)
```



- Interestingly, Martin reran the same experiment slightly later and got a second set of data `zoo_small.1.dat`. When using this set instead, we get exactly the same analysis except that the first outlier (page 24) does not appear anymore. The other few "outliers" are still there though.
- The behavior is **not** continuous. I remember discussions about enforcing that piece-wise linear models are continuous. It's a nice wrong idea with no justification.
- It took me the afternoon to write this data analysis but I think the insights it provided were worth the effort. Two ranges seem way sufficient. What matters is to incorporate some noise because it really matters when message size is not very large.
- Interestingly the boundary is not that clear. When message sizes become close to 6.5E4, some messages follow the "short messages" model while others follow the "long messages" model. It's hard to tell why and I could not find any way to discriminate from this kind of messages. If we had worked on averaged values, we would never had spotted this. Instead, we would have opted for some continuous model with a strange behavior.
- I doubt this new model will help when simulating Sweep3D and a more "precise" one using more ranges would not help either. The real problem of Sweep3D we identified when looking at Paje traces is the ability to overlap and possibly aggregate small messages. This will be the subject of our next set of experiments.
- Let me be provocative now. Is the green curve that bad ? ;)