Plots

September 3, 2017

1 Some plots and analysis of simulation data

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import math
        %matplotlib inline
```

1.1 Simulation Parameters

```
In [2]: test_info = pd.read_csv("../data/backup/stats_mlfd/test_info.csv")
        test_info
```

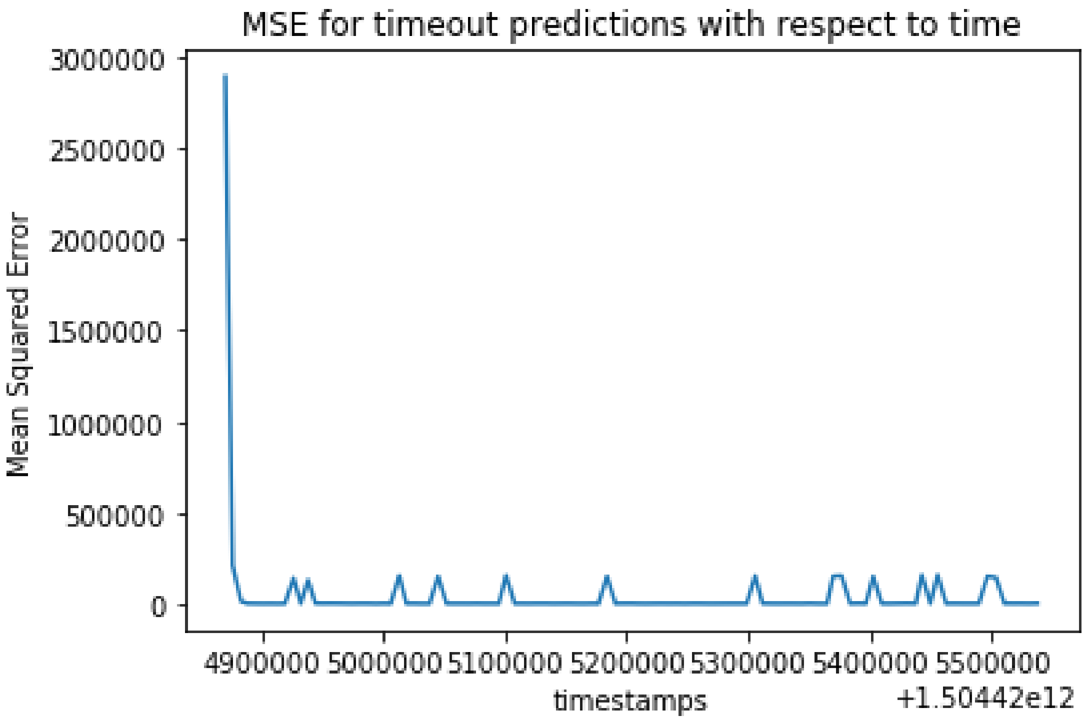
```
Out[2]:        test  workersCount  locationsCount  sampleSize  defaultMean  \
        0  mlfd_test           100             100         100       3000.0

           hbTimeout  randomMillis  geoFactor  crashProb  delta
        0  4 seconds           200      100.0      0.001    NaN
```

1.2 Prediction Evaluation of the Online-learner in the FD

```
In [3]: test_evaluations = pd.read_csv("../data/backup/stats_mlfd/testresults.csv")
        test_evaluations.head()
```

```
Out[3]:    meanSquaredError  rootMeanSquaredError  rSquared  meanAbsoluteError  \
        0      2.891005e+06           1700.295700 -0.815136        1418.753397
        1      2.135465e+05            462.110885  0.974527         349.727503
        2      1.632067e+04            127.752381  0.996729         105.777594
        3      4.299215e+03             65.568400  0.999504          56.537756
        4      3.624345e+03             60.202532  0.999099          48.894390

           explainedVariance      timestamp
        0       6.844245e+06  1504424870000
        1       1.041643e+07  1504424876000
        2       5.328207e+06  1504424883000
        3       8.820089e+06  1504424888000
        4       4.148113e+06  1504424895000
```

In [4]:
```
time = test_evaluations["timestamp"].values
mse = test_evaluations["meanSquaredError"].values
plt.plot(time, mse)
plt.ylabel('Mean Squared Error')
plt.xlabel('timestamps')
plt.title('MSE for timeout predictions with respect to time')
plt.show()
```



1.3 RTT Distributions related to geographic location

In [5]:
```
rtt_data = pd.read_csv('../data/backup/stats_mlfd/rtt_data.csv')
```

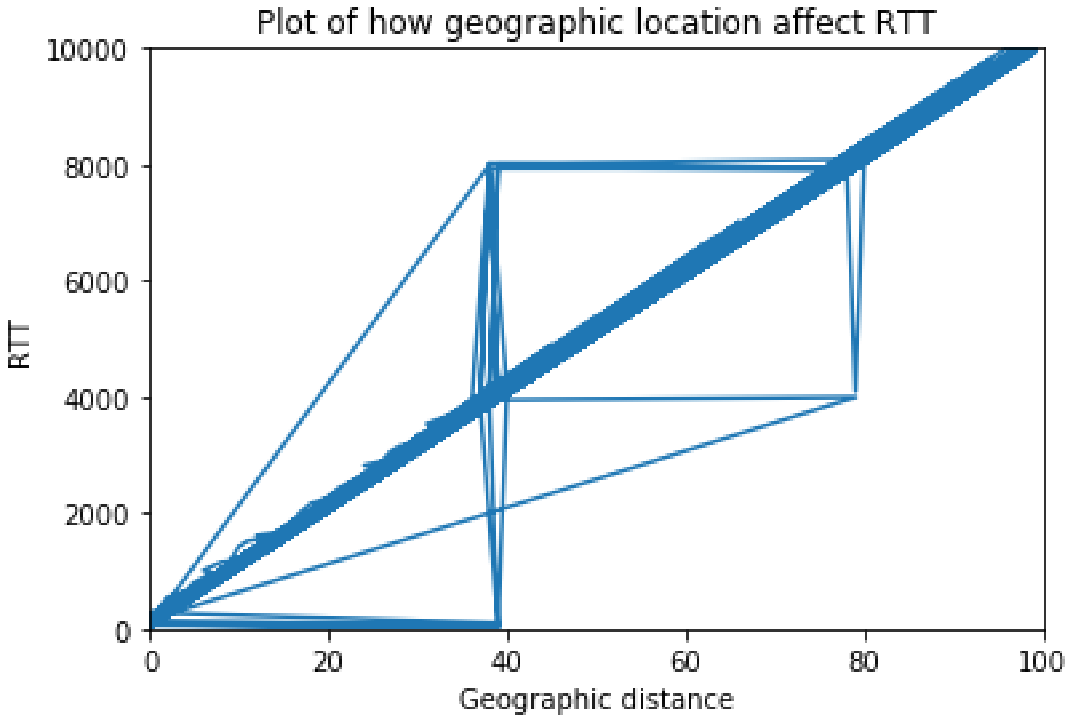
In [6]:
```
rtt_data.head()
```

Out[6]:

	node	geoLocation	rtt	timestamp	mean
0	100	0	270.0	1504424862481	270.0
1	1	1	338.0	1504424862550	338.0
2	2	2	548.0	1504424862760	548.0
3	3	3	599.0	1504424862811	599.0
4	4	4	648.0	1504424862860	648.0

In [7]:
```
geo = rtt_data["geoLocation"].values
rtt = rtt_data["rtt"].values
plt.plot(geo, rtt)
```

```
plt.axis([0, 100, 0, 10000])
plt.ylabel('RTT')
plt.xlabel('Geographic distance')
plt.title('Plot of how geographic location affect RTT')
plt.show()
```



In [8]:
```
rtt_data_with_mean = rtt_data.groupby(['geoLocation', 'rtt'], as_index=False).mean().gro
rtt_data_with_mean.head()
```

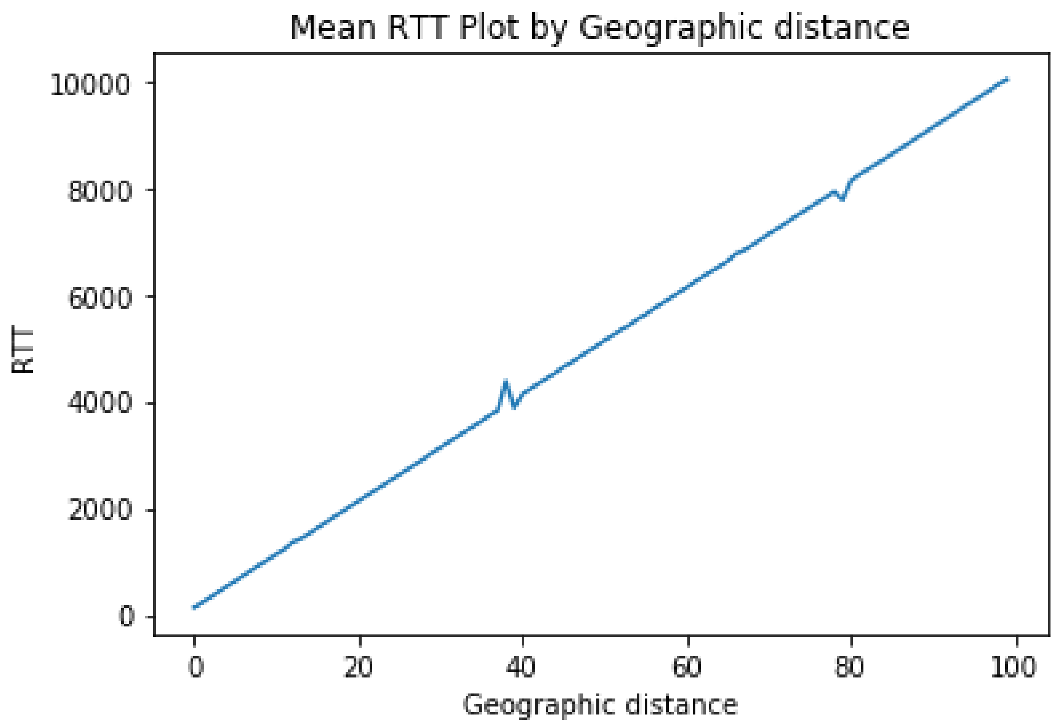
Out[8]:
```
geoLocation
0    131.860000
1    226.456522
2    325.627907
3    427.609756
4    530.630435
Name: rtt, dtype: float64
```

In [9]:
```
rtt_data_with_std = rtt_data.groupby('geoLocation')[['rtt']].std()
rtt_data_with_std.head()
```

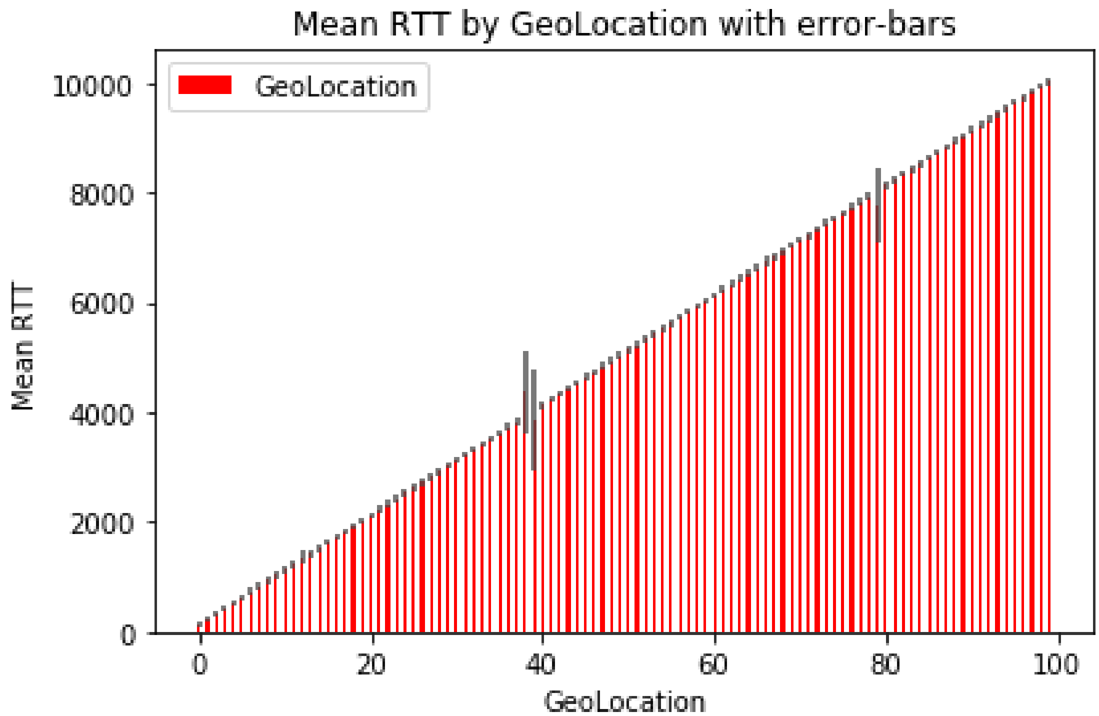
Out[9]:
```
                   rtt
geoLocation
0            58.987944
1            57.103708
```

```
2              61.323999
3              59.254465
4              60.819616
```

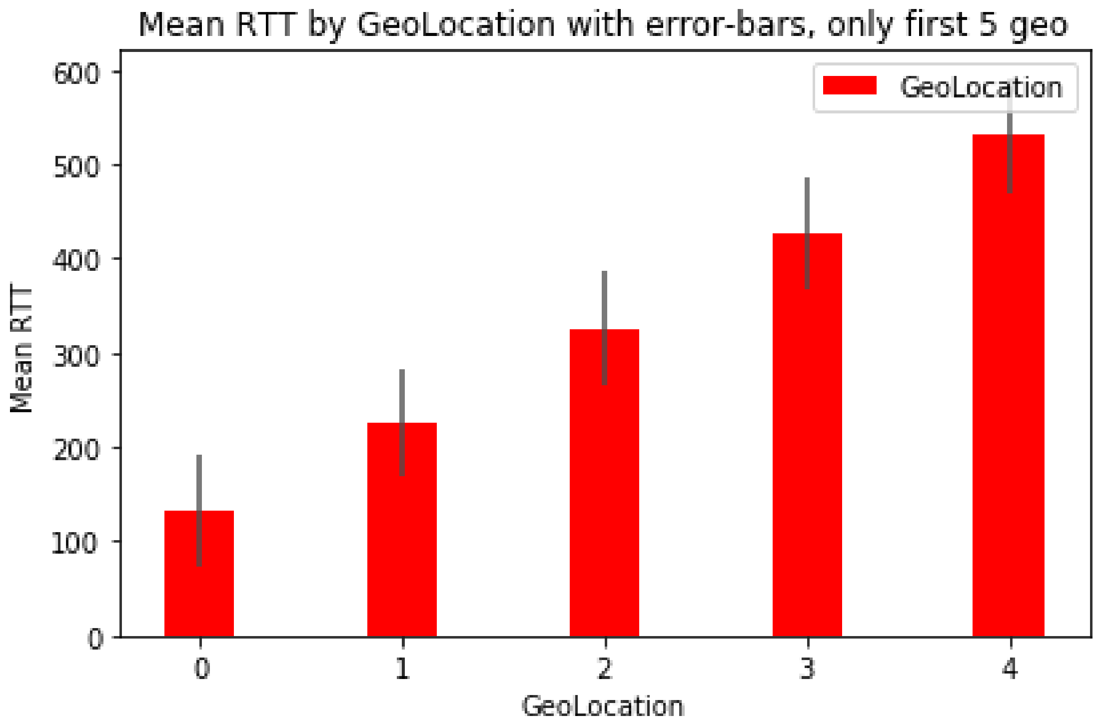
In [10]:
```
plt.plot(rtt_data_with_mean)
#plt.axis([0, 100, 0, 10000])
plt.ylabel('RTT')
plt.xlabel('Geographic distance')
plt.title('Mean RTT Plot by Geographic distance')
plt.show()
```



In [11]:
```
rtt = rtt_data_with_mean.values
std = rtt_data_with_std.values
l = len(rtt)
width = 0.35
error_config = {'ecolor': '0.3'}
plt.bar(range(len(rtt)), rtt, width=width, color='r', yerr=std, error_kw=error_config,
plt.xlabel('GeoLocation')
plt.ylabel('Mean RTT')
plt.title('Mean RTT by GeoLocation with error-bars')
plt.legend()
plt.tight_layout()
```

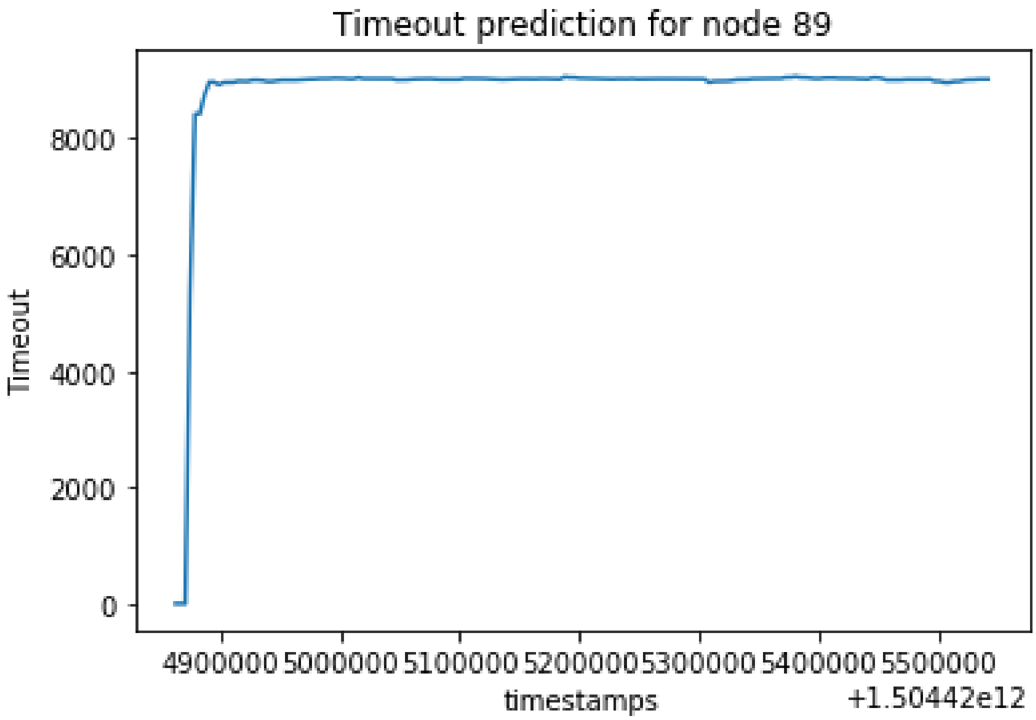



```
In [12]: rtt = rtt_data_with_mean.head().values
         std = rtt_data_with_std.head().values
         l = len(rtt)
         width = 0.35
         error_config = {'ecolor': '0.3'}
         plt.bar(range(len(rtt)), rtt, width=width, color='r', yerr=std, error_kw=error_config,
         plt.xlabel('GeoLocation')
         plt.ylabel('Mean RTT')
         plt.title('Mean RTT by GeoLocation with error-bars, only first 5 geo')
         plt.legend()
         plt.tight_layout()
```



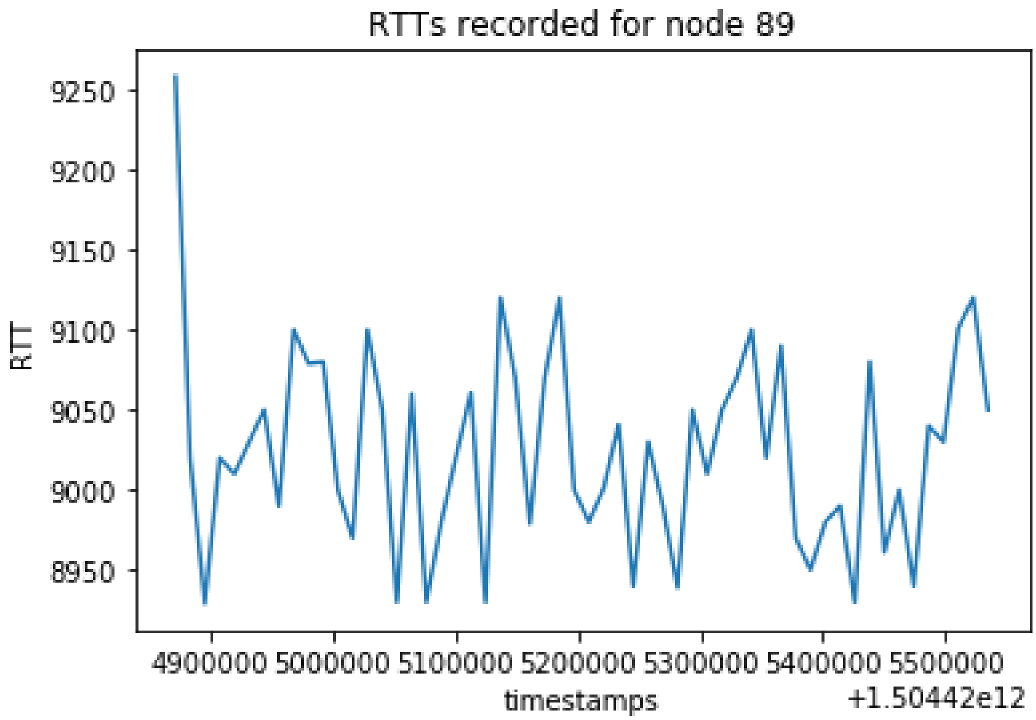
1.4 Predictions for a single Node

```
In [13]: prediction_data = pd.read_csv('../data/backup/stats_mlfd/prediction_data.csv')
         node89 = prediction_data.loc[prediction_data['node'] == 89]
         time = node89["timestamp"].values
         prediction = node89["prediction"].values
         plt.plot(time, prediction)
         plt.ylabel('Timeout')
         plt.xlabel('timestamps')
         plt.title('Timeout prediction for node 89')
         plt.show()
```



1.5 RTT-data for a single Node

```
In [14]: node89 = rtt_data.loc[rtt_data['node'] == 89]
         time = node89["timestamp"].values
         prediction = node89["rtt"].values
         plt.plot(time, prediction)
         plt.ylabel('RTT')
         plt.xlabel('timestamps')
         plt.title('RTTs recorded for node 89')
         plt.show()
```



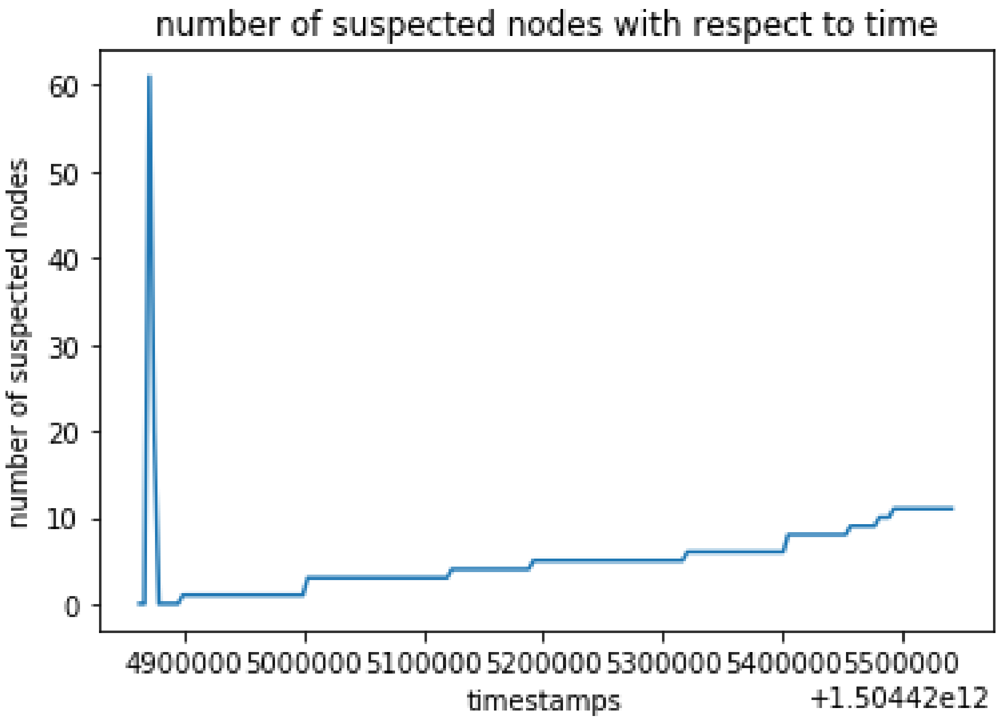
1.6 Analysis of number of suspected nodes with respect to time

```
In [15]: suspected_nodes_data = pd.read_csv('../data/backup/stats_mlfd/suspected_nodes.csv')
         suspected_nodes_data.head()
```

Out[15]:

	timestamp	numberOfSuspectedNodes
0	1504424862212	0
1	1504424866440	0
2	1504424870471	61
3	1504424874490	20
4	1504424878510	0

```
In [16]: time = suspected_nodes_data["timestamp"].values
         suspected = suspected_nodes_data["numberOfSuspectedNodes"].values
         plt.plot(time, suspected)
         plt.ylabel('number of suspected nodes')
         plt.xlabel('timestamps')
         plt.title('number of suspected nodes with respect to time')
         plt.show()
```




```
In [17]: node_crashes = pd.read_csv('../data/backup/stats_mlfd/node_crashes.csv')
         node_suspicions = pd.read_csv('../data/backup/stats_mlfd/node_suspicions.csv')

In [18]: node_suspicions = pd.merge(node_suspicions, node_crashes, how="inner", on="node")
         node_suspicions = node_suspicions.sort_values('suspected', ascending=False).drop_duplic

In [19]: node_suspicions["detection_time"] = node_suspicions["suspected"] - node_suspicions["tim

In [20]: node_suspicions = node_suspicions[node_suspicions["detection_time"]>0] #Remove prematur

In [21]: node_suspicions
```

Out[21]:

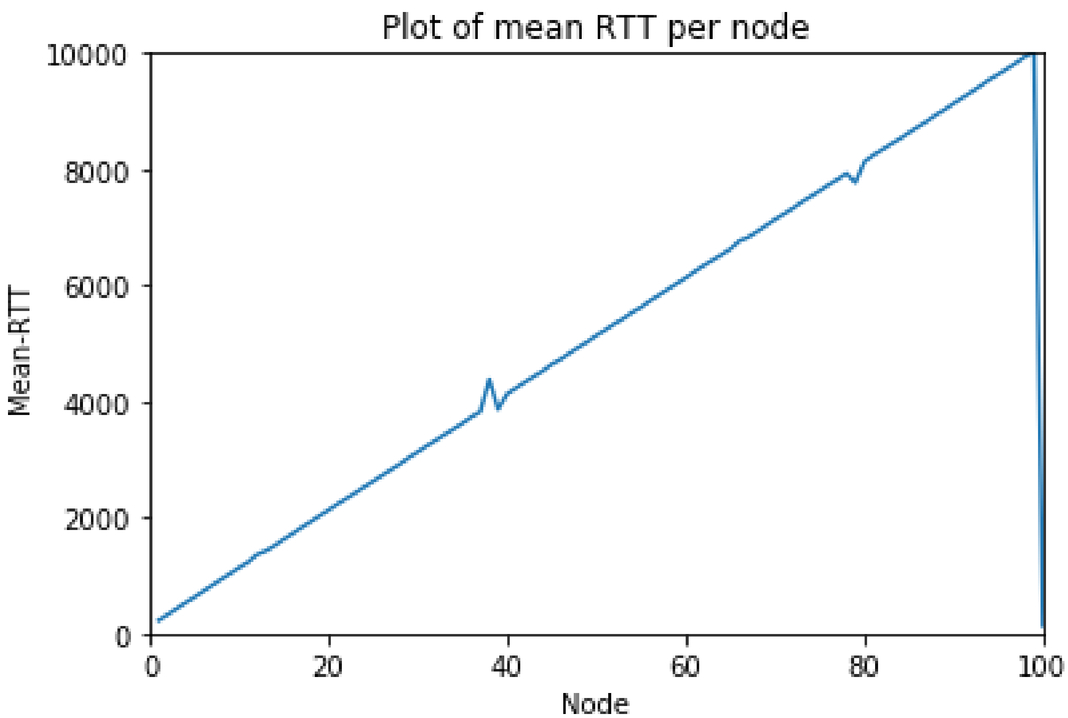
	node	suspected	timestamp	detection_time
1	91	1504425188050	1504425184031	4019
3	54	1504425119710	1504425115691	4019
6	66	1504424999110	1504424995091	4019
8	97	1504425453370	1504425441311	12059
10	73	1504424999110	1504424991072	8038
12	63	1504425401110	1504425393071	8039
14	74	1504425489550	1504425481511	8039
15	12	1504424894591	1504424890574	4017
16	14	1504425316690	1504425312671	4019
17	22	1504425401110	1504425397092	4018
18	20	1504425477490	1504425473471	4019

1.7 Mean RTT per node

In [22]:
```
#rtt_data calculate mean per node and merge
mean_rtt = rtt_data.groupby(['node', 'rtt'], as_index=False).mean().groupby('node')['rt
```

In [23]:
```
matrix = mean_rtt.as_matrix
mean = np.array(mean_rtt)
```

In [24]:
```
plt.plot(mean_rtt)
plt.axis([0, 100, 0, 10000])
plt.ylabel('Mean-RTT')
plt.xlabel('Node')
plt.title('Plot of mean RTT per node')
plt.show()
```

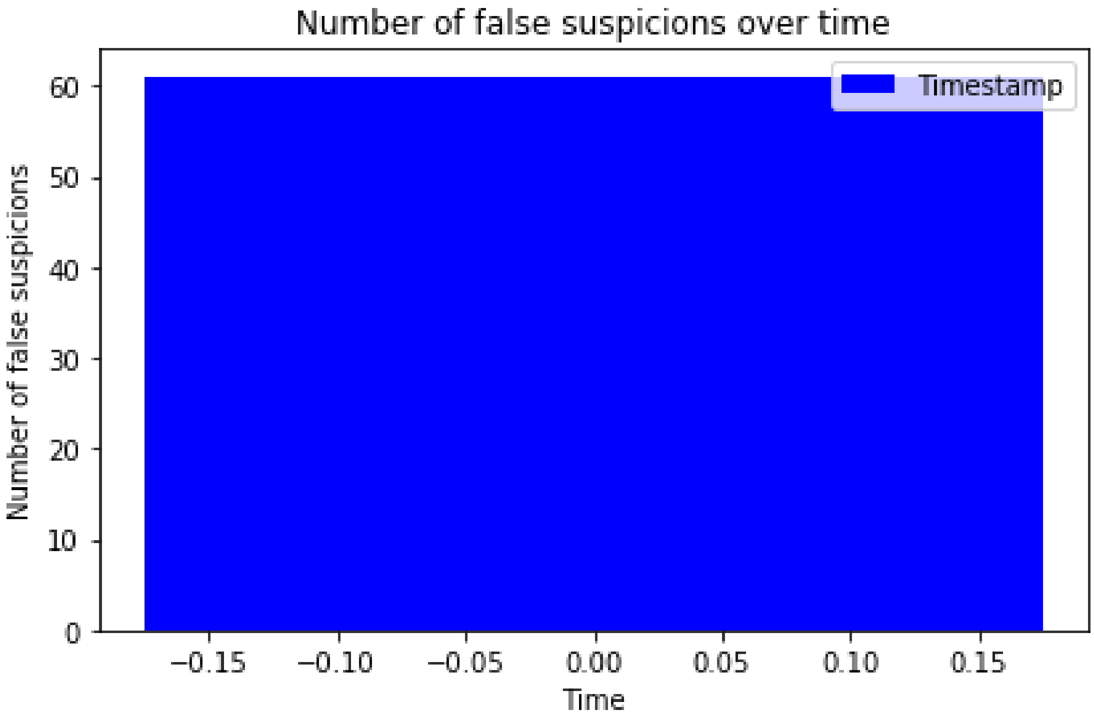


1.8 False suspicions per timeout

In [25]:
```
node_suspicions2 = pd.read_csv('../data/backup/stats_mlfd/node_suspicions.csv')
keys = ['suspected', "node"]
i1 = node_suspicions2.set_index(keys).index
i2 = node_suspicions.set_index(keys).index
false_suspicions = node_suspicions2[~i1.isin(i2)]
false_suspicions = false_suspicions.groupby("suspected").count()
false_suspicions
```

```
Out[25]:                 node
         suspected
         1504424866440    61
```

```
In [27]: mlfd_false_suspicions = false_suspicions.values
         l = len(mlfd_false_suspicions)
         width = 0.35
         error_config = {'ecolor': '0.3'}
         plt.bar(range(len(mlfd_false_suspicions)), mlfd_false_suspicions, width=width, color='b
         plt.xlabel('Time')
         plt.ylabel('Number of false suspicions')
         plt.title('Number of false suspicions over time')
         plt.legend()
         plt.tight_layout()
```

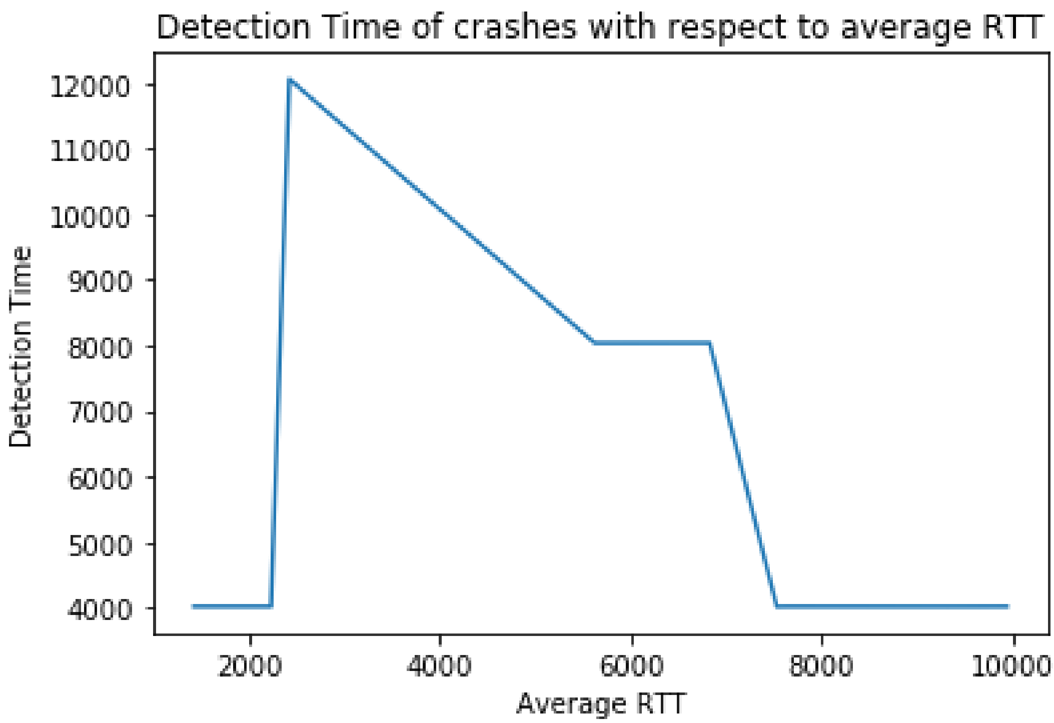


1.9 Time a node was suspected - time node actually crashed, plotted with respect to mean RTT

```
In [28]: mean_n = np.zeros(len(node_suspicions["node"]))
         j = 0
         for i in range(0, len(mean)):
             if(i in node_suspicions["node"].values):
                 #print(i)
```

```
            mean_n[j] = mean[i]
            j = j+1
```

```
In [29]: plt.plot(mean_n, node_suspicions["detection_time"])
         plt.ylabel('Detection Time')
         plt.xlabel('Average RTT')
         plt.title('Detection Time of crashes with respect to average RTT')
         plt.show()
```



1.10 Implementation of an Offline-model to analyze the data, using Keras+Tensorflow

```
In [30]: from keras.models import Sequential
         from keras.layers import Dense
         from sklearn.metrics import r2_score

         mean_geo_data = rtt_data.drop(["node", "timestamp"], 1)
         #mean_geo_data = mean_geo_data[["geoLocation","rtt","mean"]].apply(pd.to_numeric)
         df_train, df_test = np.split(mean_geo_data.sample(frac=1), [int(.8*len(mean_geo_data))]
         Y_train = df_train["rtt"].values
         Y_test = df_test["rtt"].values
         X_train = df_train.drop("rtt", 1).values
         X_test = df_test.drop("rtt", 1).values
```

```
Using TensorFlow backend.
```

```
In [31]: mean_geo_data.head()
```

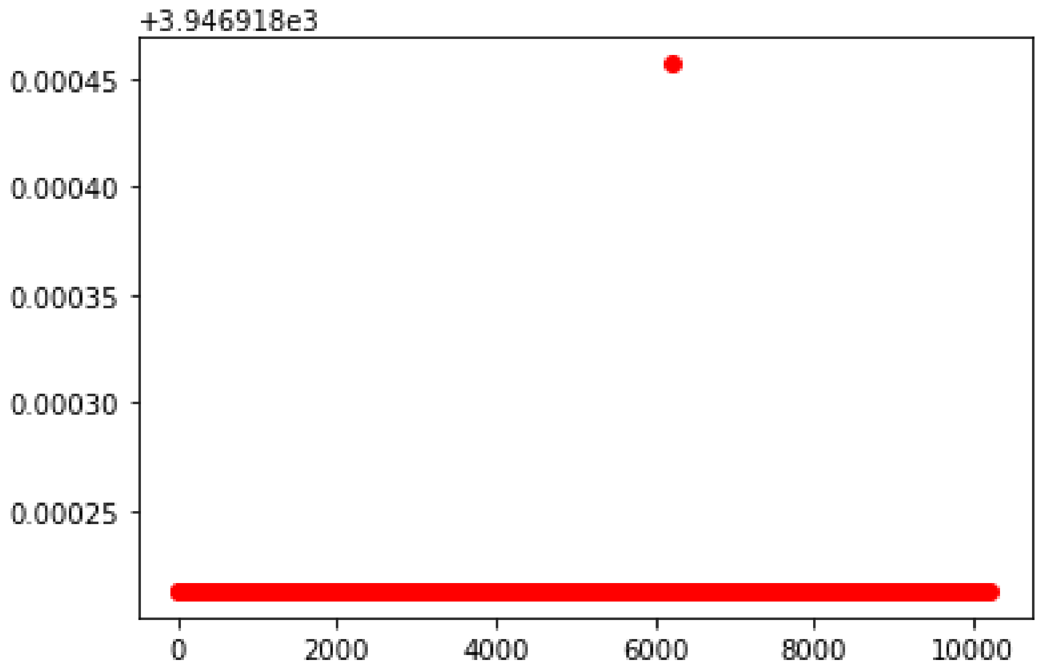
```
Out[31]:    geoLocation    rtt   mean
         0            0  270.0  270.0
         1            1  338.0  338.0
         2            2  548.0  548.0
         3            3  599.0  599.0
         4            4  648.0  648.0
```

```
In [32]: # create model
         model = Sequential()
         model.add(Dense(20, input_dim=2, init='uniform', activation='tanh'))
         model.add(Dense(1, init='uniform', activation='linear'))

         # Compile model
         model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])

         # Fit the model
         model.fit(X_train, Y_train, nb_epoch=1000, batch_size=10,  verbose=0)

         # Calculate predictions
         PredTestSet = model.predict(X_train)
         PredValSet = model.predict(X_test)

         # Save predictions
         np.savetxt("trainresults.csv", PredTestSet, delimiter=",")
         np.savetxt("valresults.csv", PredValSet, delimiter=",")
```

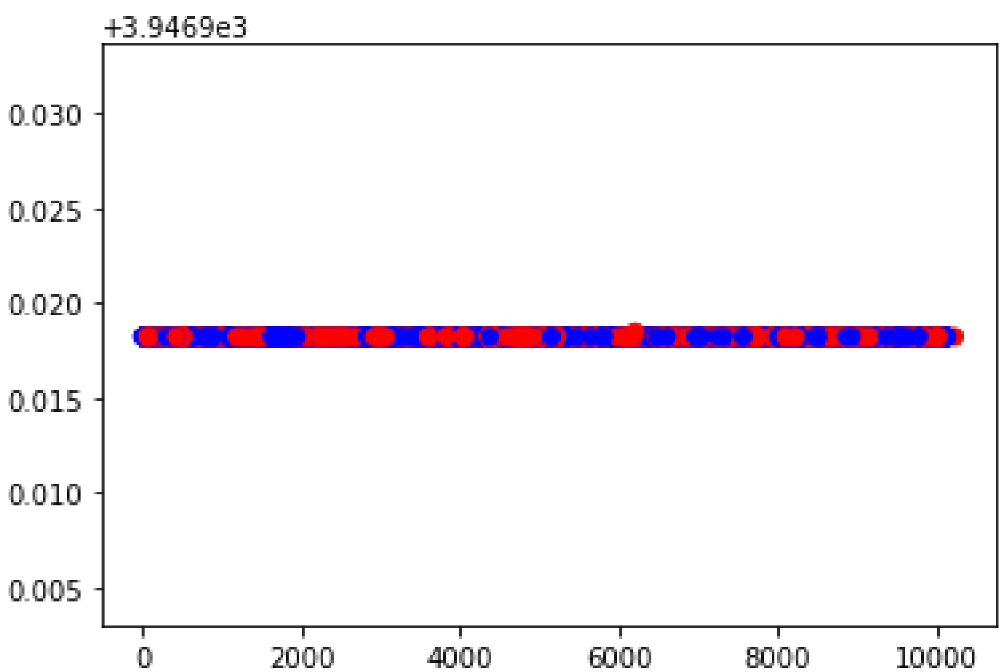
```
/home/limmen/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: UserWarning: Update
  This is separate from the ipykernel package so we can avoid doing imports until
/home/limmen/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:4: UserWarning: Update
  after removing the cwd from sys.path.
/home/limmen/anaconda3/lib/python3.6/site-packages/keras/models.py:844: UserWarning: The `nb_epo
  warnings.warn('The `nb_epoch` argument in `fit` '
```

```
In [33]: #Plot actual vs predition for training set
         TestResults = np.genfromtxt("trainresults.csv", delimiter=",")
         plt.plot(Y_train,TestResults,'ro')

         #Compute R-Square value for training set
         TestR2Value = r2_score(Y_train,TestResults)
         print("Training Set R-Square=", TestR2Value)
```

```
Training Set R-Square= -7.45569161964e-09
```



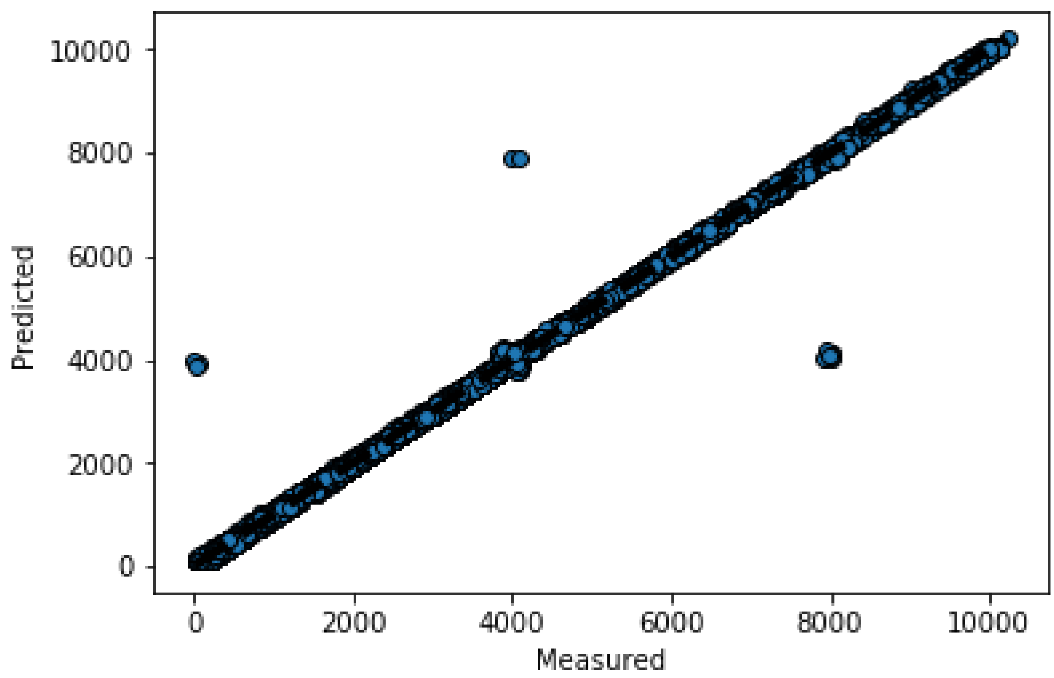

```
In [34]: plt.scatter(Y_train, TestResults,color=["red", "blue"])
         plt.show()
```



1.11 Another offline model, using sklearn

In [35]:
```
from sklearn.model_selection import cross_val_predict
from sklearn import linear_model

lr = linear_model.LinearRegression()

predicted = cross_val_predict(lr, X_train, Y_train, cv=10)
```

In [36]:
```
fig, ax = plt.subplots()
ax.scatter(Y_train, predicted, edgecolors=(0, 0, 0))
ax.plot([Y_train.min(), Y_train.max()], [Y_train.min(), Y_train.max()], 'k--', lw=4)
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.show()
```



1.12 Comparison MLFD vs EPFD

In [37]:
```
node_crashes_epfd = pd.read_csv('../data/backup/stats_epfd/node_crashes.csv')
node_suspicions_epfd = pd.read_csv('../data/backup/stats_epfd/node_suspicions.csv')
node_suspicions_epfd
node_suspicions_epfd = pd.merge(node_suspicions_epfd, node_crashes_epfd, how="inner", o
node_suspicions_epfd = node_suspicions_epfd.sort_values('suspected', ascending=False).d
node_suspicions_epfd["detection_time"] = node_suspicions_epfd["suspected"] - node_suspi
node_suspicions = node_suspicions[node_suspicions["detection_time"]>0] #Remove prematur
node_suspicions_epfd
```

Out[37]:

	node	suspected	timestamp	detection_time
4	45	1504433437756	1504433427237	10519
11	79	1504433921665	1504433911146	10519
15	42	1504433479836	1504433469317	10519
22	78	1504433574516	1504433563996	10520
25	20	1504433490356	1504433479836	10520
27	4	1504433711275	1504433700756	10519
33	71	1504433490356	1504433479836	10520
41	92	1504433753356	1504433742836	10520
45	34	1504433183715	1504433178697	5018

In [38]:
```
#rtt_data calculate mean per node and merge
mean_rtt_epfd = rtt_data.groupby(['node', 'rtt'], as_index=False).mean().groupby('node'
matrix_epfd = mean_rtt.as_matrix
mean_epfd = np.array(mean_rtt_epfd)
```

In [39]:
```
mean_n_epfd = np.zeros(len(node_suspicions_epfd["node"]))
j = 0
for i in range(0, len(mean)):
    if(i in node_suspicions_epfd["node"].values):
        #print(i)
        mean_n_epfd[j] = mean_epfd[i]
        j = j+1
```

In [40]:
```
plt.plot(mean_n_epfd, node_suspicions_epfd["detection_time"], label="EPFD")
plt.plot(mean_n, node_suspicions["detection_time"], label="MLFD")
plt.ylabel('Detection Time')
plt.xlabel('Average RTT')
plt.title('Detection Time of crashes with respect to average RTT')
plt.legend()
plt.show()
```

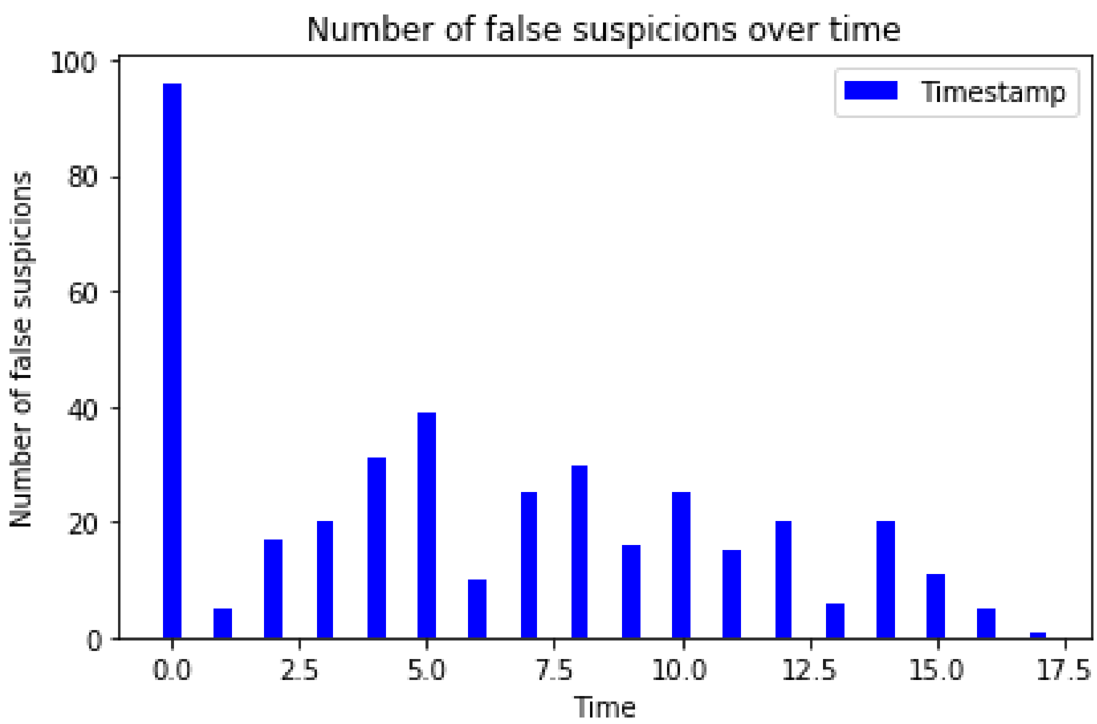
In [41]:
```
node_suspicions2_epfd = pd.read_csv('../data/backup/stats_epfd/node_suspicions.csv')
keys_epfd = ['suspected', "node"]
i1_epfd = node_suspicions2_epfd.set_index(keys_epfd).index
i2_epfd = node_suspicions_epfd.set_index(keys_epfd).index
false_suspicions_epfd = node_suspicions2_epfd[~i1_epfd.isin(i2_epfd)]
false_suspicions_epfd = false_suspicions_epfd.groupby("suspected").count()
false_suspicions_epfd
```

Out[41]:

	node
suspected	
1504433156016	96
1504433159076	5
1504433161096	17
1504433163616	20
1504433166636	31
1504433170156	39
1504433174176	10
1504433178696	25
1504433183715	30
1504433189236	16
1504433195256	25
1504433201776	15
1504433208796	20
1504433216315	6

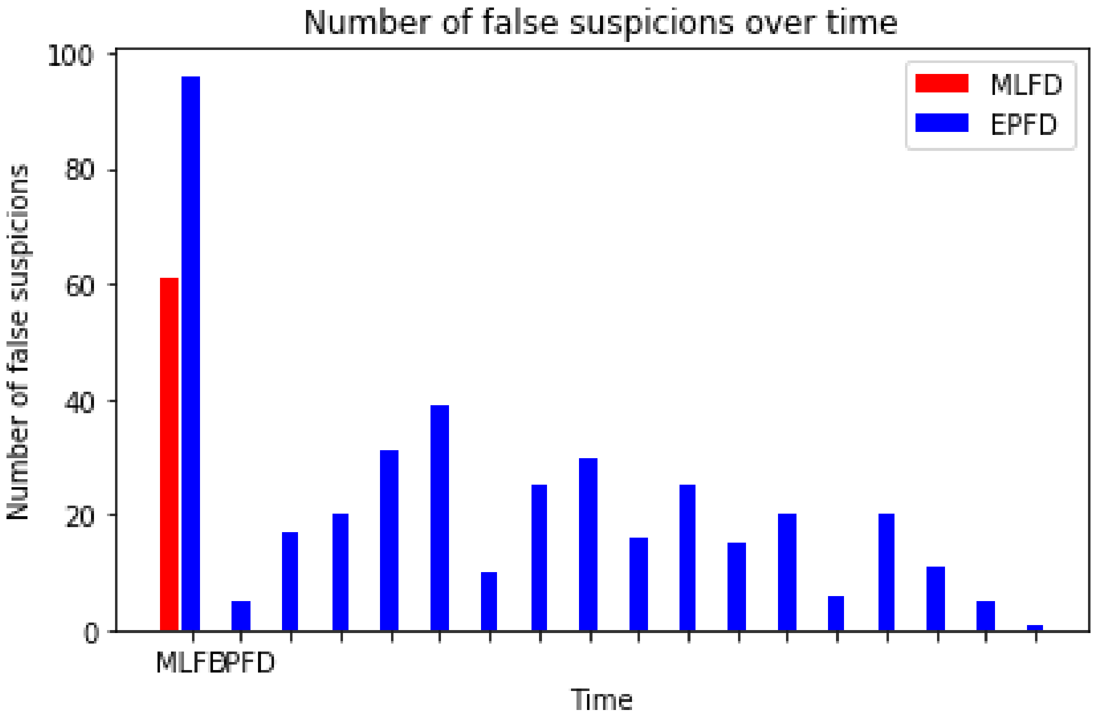
```
1504433224336    20
1504433241876    11
1504433260416     5
1504433269936     1
```

```
In [42]: epfd_false_suspicions = false_suspicions_epfd.values
         l = len(epfd_false_suspicions)
         width = 0.35
         error_config = {'ecolor': '0.3'}
         plt.bar(range(len(epfd_false_suspicions)), epfd_false_suspicions, width=width, color='b
         plt.xlabel('Time')
         plt.ylabel('Number of false suspicions')
         plt.title('Number of false suspicions over time')
         plt.legend()
         plt.tight_layout()
```




```
In [43]: bar_width = 0.45
         index1 = np.arange(l)
         index2 = np.arange(len(mlfd_false_suspicions))
         plt.bar(index2, mlfd_false_suspicions, width=width, color='r', error_kw=error_config, l
         plt.bar(index1 + bar_width, epfd_false_suspicions, width=width, color='b', error_kw=err
         plt.xlabel('Time')
         plt.ylabel('Number of false suspicions')
         plt.title('Number of false suspicions over time')
```

```
plt.legend()
plt.xticks(index1 + bar_width, ('MLFD', 'EPFD'))
plt.tight_layout()
```




```
In [ ]:
```