1 COURSE CHANGE NOTIFIER

A large number of mobility models, which are commonly used in network simulation studies, utilize piecewise linear paths with no acceleration. In these mobility models, the node motion in between two consecutive updates can be represented by the speed and the direction of the node. A collection of the node speeds and the directions at all update instances and for all nodes completely describe the mobility pattern.

The built in mobility models in ns3 implement a callback mechanism that reports any changes in nodes' speed or direction. We utilize this callback mechanism by redirecting the callbacks to a function that records the new info in a file, ``CourseChangeNotifier.txt". There are 8 pieces of information that is being recorded.

1. Time of change rounded down to nearest microsecond.
2. Node ID that is going through change
3. The new speed of the node along x axis
4. The new speed of the node along y axis
5. The new speed of the node along z axis
6. X location of the node
7. Y location of the node
8. Z location of the node

The 3rd spatial dimension is reported in this file, for compatibility for future processing. Since planar mobility models are assumed in the study, this information is not used further.

2 NEIGHBORHOOD CALCULATOR

2.1 THE GOAL:

The goal of the neighborhood calculator is to calculate the time instance in which any node pair to be in the vicinity of one another.

A node pair is assumed to be in each other neighborhood when the distance between them is shorter than a fixed communication distance. This fixed distance represents the largest distance that allows direct communication between the nodes under ideal conditions and is calculated assuming zero interference and no fading. Ideally, two nodes becoming neighbors establish an ideally possible link in between themselves and that link gets broken when they leave each other's neighborhood.

The goal of the neighborhood calculator is to calculate the time instances of the ideal link establishments and breakages for all node pairs. This is implemented in a MATLAB script, ``NeighborhoodCalculator.m'' file.

2.2 THE INPUTS:

The inputs for each simulation set is defined in the ``*SimulationInfo.mat*'' file residing in the simulation directory. The Neighborhood Calculator script loads this workspace in order to gather information about the simulation. The main information used by the neighborhood calculator is described below.

2.2.1 MAIN INPUTS:

nameofCourseChangeDataFile: The name of the file storing Course Change notifier data. This is the input file containing the information about the mobility pattern.

IsInputDataInTextFormat **:** 1 if the input data is in text format and 0 if the input data is in MATLAB workspace format
N: Number of nodes
R: Maximum neighborhood distance
simulationEndtime: Simulation end time.

2.2.2 OPTIONAL INPUTS:

plotStateMatricies= Inf,x,0 (default). Triggers the creation of a graph that represent node locations and speeds at each update for x many updates. In order to create a graph for all the points, Inf can be passed as an argument.
plotonPreviousFigure = 1, 0(default). Instead of creating a new figure, the previous figure is used.

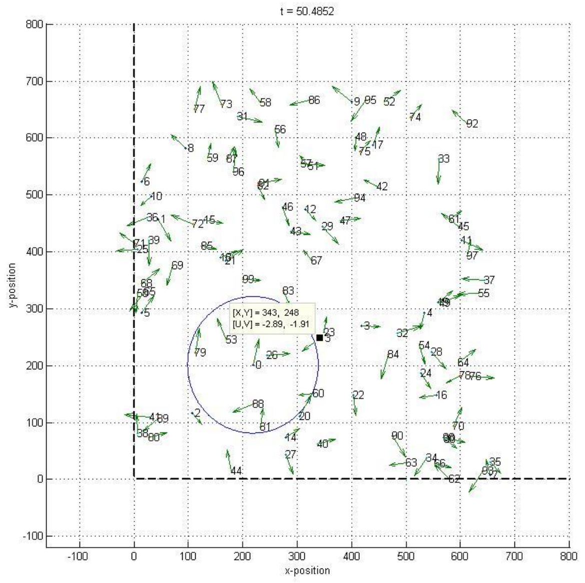

Figure 1 Node speeds and distribution using state matrix at a particular time.

2.3 OUTPUTS:

2.3.1 MAIN OUTPUT:

Cells4EaBTime{N,N} cell array that stores link establish and breakage times in the upper right half. For example Cells4EaBTime{i,j} is an array that stores the time of link establishments and link breakages between node I and j provided that i<j. The elements with odd indices record the time of link establishment and the elements with even indices record the time of link breakages.

2.3.2 OPTIONAL OUTPUTS:

Results_el(3,:) **:** Stores time of link establishments in the [node1;node2;time] format
Results_bl(3,:):Stores time of link breakages in the [node1;node2;time] format

2.4 OVERVIEW OF EXECUTION:

The execution of the program involves keeping a state matrix and updating it with each input line from the course change notifier data. The state of a node is represented by a variable array, namely [LastUpdateTime, Positionx, Positiony, Vx, Vy]. The node is known to be at the point (Positionx, Positiony) at time LastUpdateTime and moving with (Vx,Vy), the velocity components along x and y axes. The state of the network is the collection of the states of all the nodes and stored in a matrix format in the variable, *STATE_MATRIX*. *STATE_MATRIX* has a size of Nx6, where N represents the number of nodes in the network. I'th row of *STATE_MATRIX* stores the state of node with id (i-1).

For a given *STATE_MATRIX*, the square of the distance between two nodes is described by a second order polynomial and is calculated by the function, *DistSqPoly(state1, state2)*. The roots of the DistSqPoly(state1, state2) – R^2 = At^2 + Bt+ C = 0 represents the time instances where two nodes become neighbors, i.e. link establishment instant, or leave each other's neighborhood, i.e. link breakage instant. The number of roots is checked by checking the discriminant. The roots are associated with link establishment or link breakages based on the sign of the highest degree term of the polynomial.

The roots corresponding to the points earlier than the update time of the nodes' state are discarded. The remaining roots corresponding to the future link establishment and future link breakages are temporarily stored in TempResults_el(NxN) and TempResults_bl(NxN)matricies, respectively. TempResults_el(ixj), TempResults_el(ixj) shows the time of future link establishment or breakage between nodes i and j. A stored temporary result is confirmed when an update about one of the nodes is read from the input file that happens at a later time then the stored temporary value. The confirmed results are recorded into permanent result cell arrays Cells4EaBTime{i,j}.

An overview of the execution stages of the Neighborhood calculator can be summarized by the following steps

- Initialization: Initialization of the variables used in the calculations
- Read initial node locations: Read initial node locations corresponding to t=0;
- Find initial neighbors: Find initial neighbors based on the distance
- Find initial intersection times based on the initial data: Based on the initial node velocities, nodes temporary link establishment and link breakage data is calculated.
- Process movement data line by line: Start reading from the input file using ReadLineFromCourseChangeData script, which reads the input file line by line and returns the result in readdata variable. ReadLineFromCourseChangeData is capable of handling different input files (txt and mat). If *plotStateMatricies* option is active, a figure showing the state is displayed and the program is paused for 2 seconds.
- Add final neighbors and link breakages: Input file is processed until the last update or until a time value greater than *simulationEndtime* is read. After that time, the temporary results are compared to the *simulationEndtime* and are accepted if they happen before *simulationEndtime.*
- Save outputs: Save the output variables into a MATLAB workspace file whose name is defined in the ``*SimulationInfo.mat*'' file by the variable *nameofCourseChangemat* .

3 RANDOM WAYPOINT MODEL GENERATOR

In order to generate controllable input data with minimal accuracy loss, a random waypoint model generator is implemented as a function in ``*GenerateRandomWayPointModel.m''* file. This function returns a Lx6 matrix, *CourseChangeDataMatrix*, which stores the node update time instances with the nodes' new speeds and positions.

3.1 INPUTS:

N: Number of nodes in the network
maxx : The limits of the area over which nodes are distributed and moving along x axis.
maxy : The limits of the area over which nodes are distributed and moving along y axis.
speedmax : Maximum allowed node speeds
speedmin : Minimum allowed node speeds

3.2 OUTPUTS

CourseChangeDataMatrix: (Lx6) : [UpdateTime, NodeID, Vx, Vy, Positionx,Positiony]

4 VERIFICATION OF NEIGHBORHOOD CALCULATOR

4.1 VERIFICATION STRATEGY

In addition to the potential accuracy loss due to the limited significant figures used in the input mobility files, two other points of potential accuracy loss are envisioned.

- Missing critical points due to potential bugs and conceptual mistakes.
- The accuracy loss due to the round-off errors in the Neighborhood calculator.

In order to test the internal accuracy of the Neighborhood algorithm, we run a verification test using MATLAB generated mobility data file that stores the node locations and velocities using a double precision. The test involves

i. Calculate the node locations at the time instances reported by the Neighborhood algorithm
ii. Calculate the distance between the nodes going through a link change
iii. Calculate the statistics for difference between the calculated distance and the communication radius, R.

4.2 VERIFICATION TEST

The verification of neighborhood calculator is implemented in test_NeighborhoodCalculator.m file.

test_NeighborhoodCalculator reads both the output of random waypoint model generator and the neighborhood calculator. For each time instance calculated by the neighborhood calculator, the test algorithm calculates the node positions of the nodes going through a link breakage or a link establishment. The difference between the calculated distance and the communication radius R, is stored as the raw data for the statistics in arrays, *statistics_el*, *statistics_bl*.

In order to verify neighborhood calculator, using random waypoint model generator, a mobility file for 100 nodes and 120 seconds is generated. The link establishment times and link breakage times are calculated with neighborhood calculator for this input data.

The maximum absolute value of the stored statistics are 7.532e-13 and 4.547e-13 for link establishments and link breakages respectively. The median absolute value of the stored statistics are 4.263e-14 and 4.263e-14 for link establishments and link breakages respectively. The small maximum and median values of the statistics verifies the neighborhood algorithm.

5 COMPARISON OF NEIGHBORHOOD CALCULATOR WITH LINK DETECTION

The neighborhood discovery service is implemented in the core of the SAMRAKSH framework and provides a list of nodes in the local neighborhood. The links reported by this module is saved into a memory map file in predefined intervals. The default reporting interval is set to 10 ms.

The reported values are read with MATLAB and are compared to the output of Neighborhood calculator algorithm. In this comparison, we separated the link establishments (LETs) and link breakages (LBTs) and analyzed them separately. In other words LETs from course change notifier (CN_LET) are compared to the LETs from link detection algorithm (LD_LETs).

In this comparison, we investigate

- missing LETs (that are reported by CN but not by LD)
- missing LBTs (that are reported by CN but not by LD)
- fake LETs (that are reported by LD but not by CN)
- fake LBTs (that are reported by LD but not by CN)

Hence, for each node pair, the length of the arrays CN_LET and LD_LET (also CN_LBT and LD_LBT) are potentially different. In order to match the values reported by the CN and LD algorithms, we follow a min error approach. This algorithm is summarized as:

- If the length of LD_LET (CN_LET) is zero, all the values on the other array are treated as missed (fake) links
- If both arrays have elements,
    - The CN entries making LD_LET – CN_LET difference minimum are selected and repetitions are eliminated leading to u_CN_LET.
    - The set difference between u_CN_LET and CN_LET are the time values missed by the LD algorithm.
    - The LD entries making, LD_LET – u_CN_LET minimum are selected and the repetitions are eliminated, leading to u_LD_LET.
    - The set difference between u_LD_LET and LD_LET are the fake time values reported by the LD algorithm.
    - u_LD_LET and u_CN_LET records matching entries. Their difference is also stored as a separate statistics.
- The above procedure is repeated for LBT values.

Figure 2 Summary of the algorithm matching CN and LD values

5.1 MISSING LINKS

In the comparison, intuitively, we don't expect to observed any true link establishments or link breakages that are not reported by the LDs. This result will shows that, the link detection algorithm is working at least as fast as the minimum link duration at the given node speed limits and the power levels (communication radius). Considering the communication radius of 120 m, frequent beaconing and relatively slow maximum speed of the nodes, we don't expect to see any missed links.

However, we did observe 39 missing link establishment and 26 missing link breakages in the LD report. In other words there are some link establishments that are reported by the CN algorithm but not by the LD algorithm.

For example, as shown in Figure 3, node 0 discover node 9 on line 11137 of the allnodes.txt. On the other hand, node 9 does not discover node 0. Looking at both the CN results and the node locations reported in allnodes.txt file, we observe that these nodes did indeed became neighbors. In fact we observed none of the 36 nodes that came in contact with node 0 reported node 0 in their local neighborhood. The reason behind missing reports about node 0 should be investigated. BK: Check whether this was fixed in the master

```
0, 272.263000, 436.102997, 15 18 21 23 25 36 56 67 68 77 79 82 85 94 95

0, 272.272003, 436.140991, 9 15 18 21 23 25 36 56 67 68 77 79 82 85 94 95

0, 272.281006, 436.179993, 9 15 18 21 23 25 36 56 67 68 77 79 82 85 94 95
```

Figure 3 Lines 11136, 11136 and 11138 from the allnodes.txt file, corresponding to the reported information for node 0 at t=111.36

2 of the remaining 3 mismatched LETs correspond to node 40 and 79 that leave each other neighborhood at t = 0.00879. Given the very short duration for link discovery at the beginning of the simulation, this mismatch is expected. The other mismatch also correspond to a short duration of 0.2135 s between nodes 33 and 77. During this duration, node 77 discovers node 33 but node 77 does not. This duration is also typical considering the variability in link detection latency that is investigated further in section 5.3.

5.2 FLUCTUATING LINKS

Fluctuating links are expected since the link detection algorithm can go through collisions. Missing consecutive beacon packets can potentially let the node think the neighbor has left the neighborhood. Nevertheless, these points can be important to verify the correctness of the comparison algorithm. Considering a large node density, limited bandwidth and relatively frequent beacons, we expect fluctuating links.

We are interested in their existence, and their frequency.

5.2.1 EXAMPLE FLUCTUATING LINKS

An example of the fluctuating link is available in the sample all nodes file available in the matlab directory.

Figure 4 presents lines 6058,..,6066 from allnodes data corresponding to t=60.58-60.66s for node 0. We can clearly see that node 0 temporarily lose node 26 as its neighbor. However, Figure 5 that presents the corresponding node positions at that time instant, clearly show that node 0 and node 26 are well within each other's neighborhood.

```
0, 227.492004, 240.345001, 13 26 79 81 83 99

0, 227.501007, 240.382996, 13 26 79 81 83 99

0, 227.509003, 240.421997, 13 26 53 79 81 83 99

0, 227.518005, 240.460007, 13 53 79 81 83 99

0, 227.526993, 240.498993, 13 53 79 81 83 99

0, 227.535995, 240.537003, 13 53 79 81 83 99

0, 227.544998, 240.576004, 13 53 79 81 83 99

0, 227.552994, 240.613998, 13 53 79 81 83 99

0, 227.561996, 240.653000, 13 53 79 81 83 99

0, 227.570999, 240.692001, 13 26 53 79 81 83 99
```

Figure 4 All nodes file between t = 60.58 and t = 60.66 for node 0

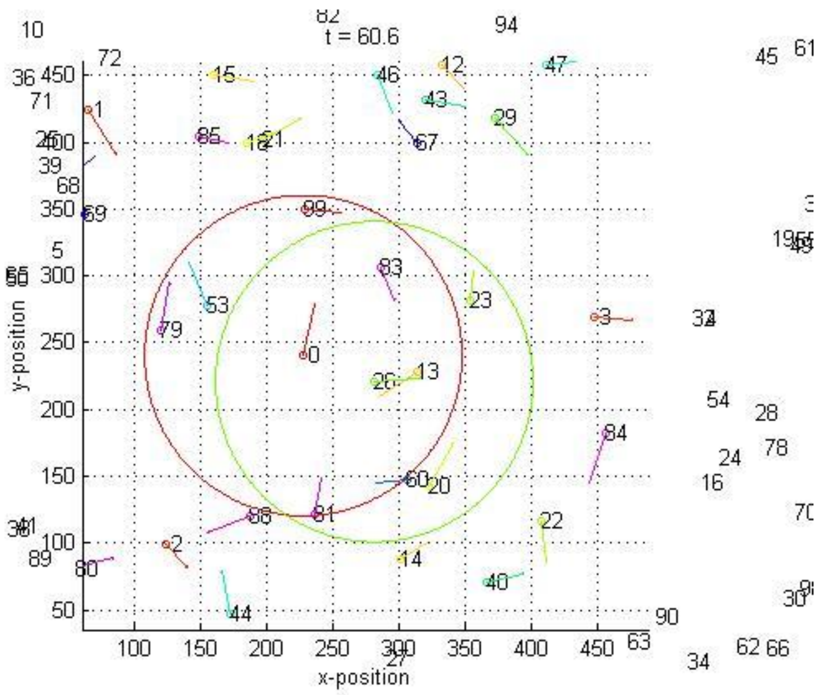

Figure 5 Node Positions at t=60.2

5.2.2 LET AND LBT VALUES DUE TO FLUCTUATING LINKS

Figure 6 and Figure 7 presents the node ID announcing the fake link establishments and node IDs announcing fake link breakages, respectively. These two figures are combined on Figure 8. Comparing the figures, it can be observed that most of the fake link breakages are detected in a short amount of time and got corrected by announcing a fake link establishment.

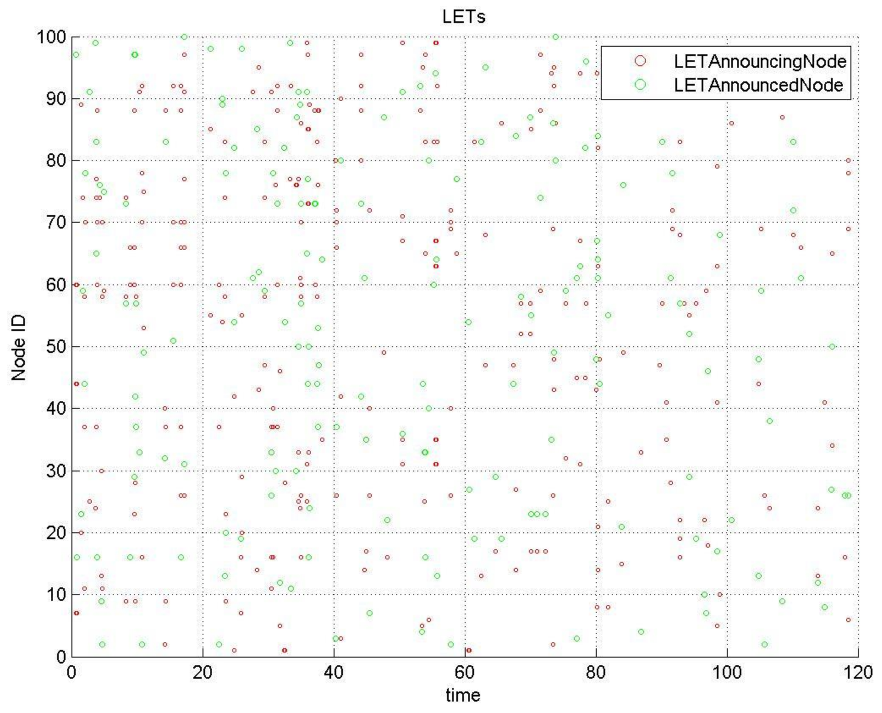

Figure 6 Fake link establishments between nodes

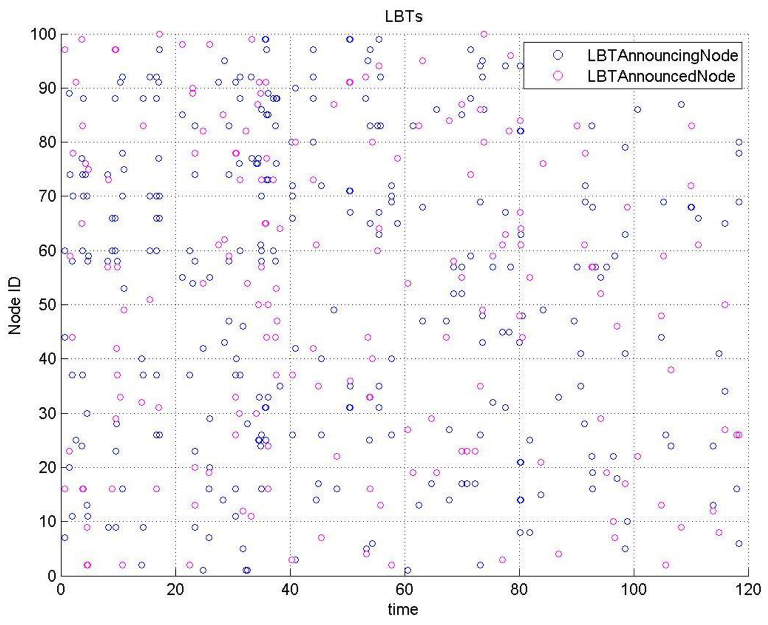

Figure 7 Fake link breakages between nodes

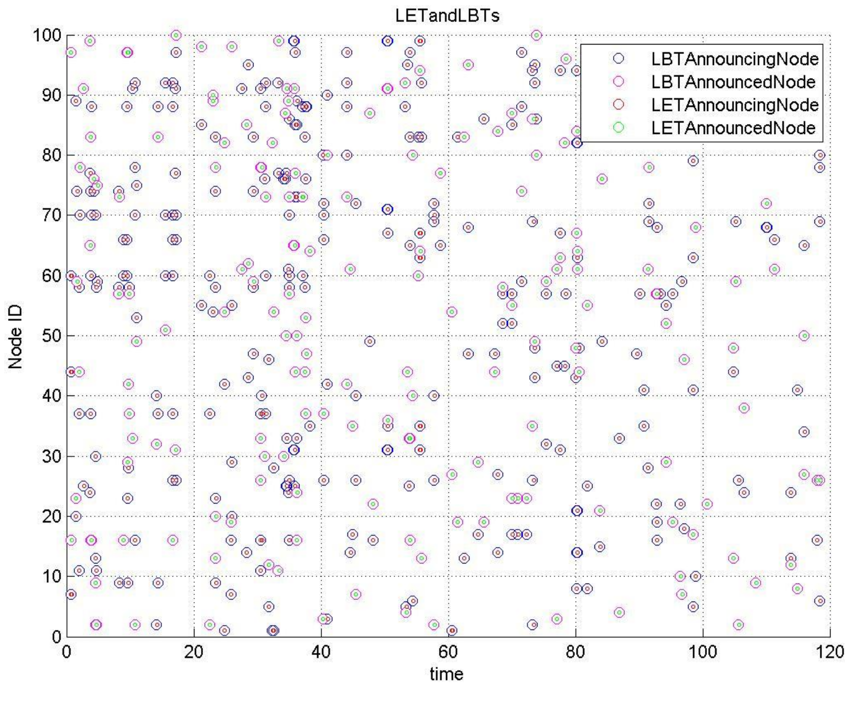

Figure 8 LETs and LBTs drawn together

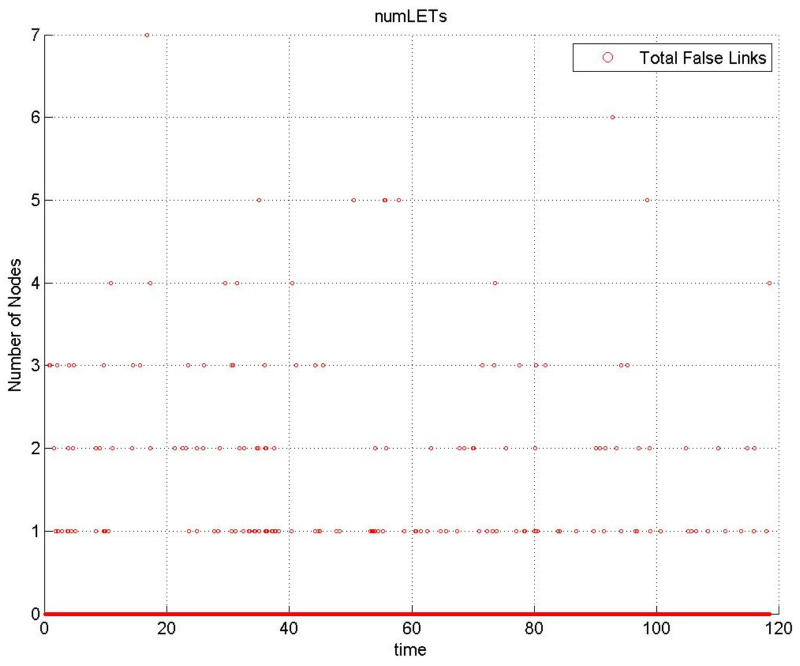

Figure 9 Number of fake link establishment in each beaconing interval

5.2.1 FLUCTUATING LINK LOSS DURATION

Although fake link breakages are detected in a short amount of time and got corrected by announcing a fake link establishment, during that interval the node is not listed in the neighborhood list. In this section, we investigate the length of this duration statistically.

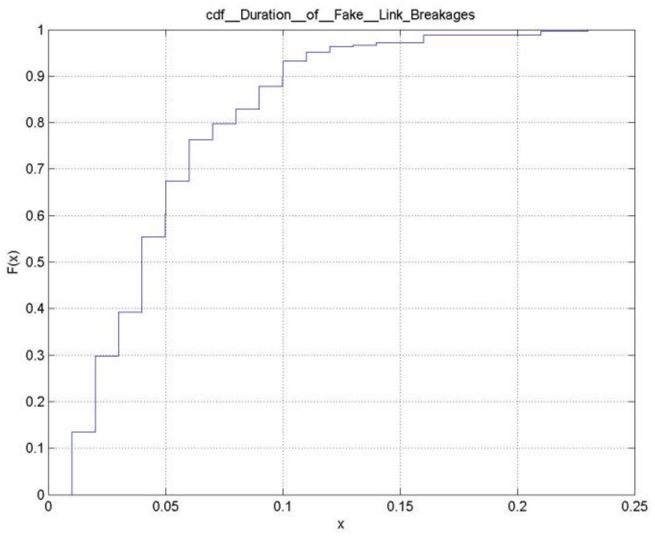

Figure 10 CDF of link loss duration due to fluctuating links

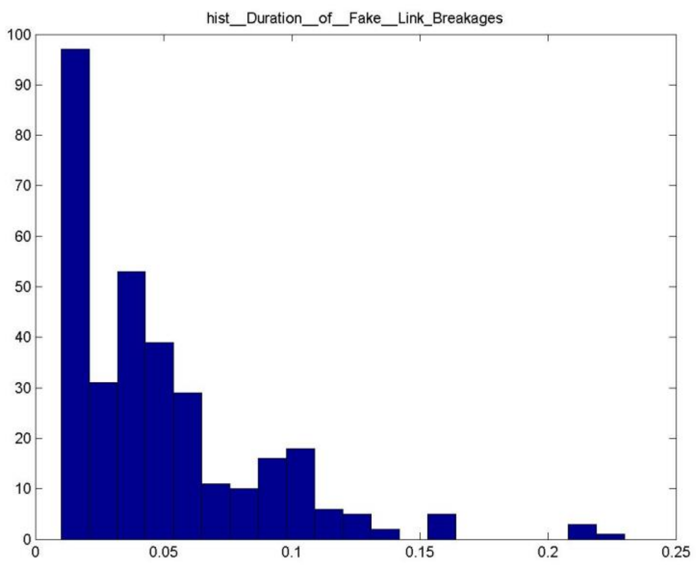

Figure 11 Histogram of link loss duration due to fluctuating links

5.3 CDF AND PDFS OF THE DIFFERENCE BETWEEN LD AND THE CN

5.3.1 LINK ESTABLISHMENTS

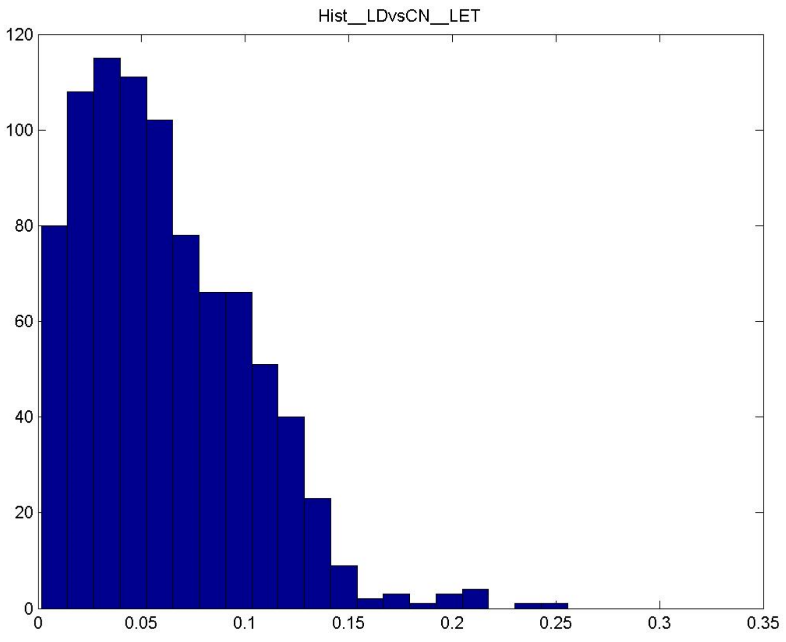

Figure 12 Histogram of the time difference between reported link establishment times of the link detection algorithm of the framework and the reported link establishment times calculated with the course change notifier data.

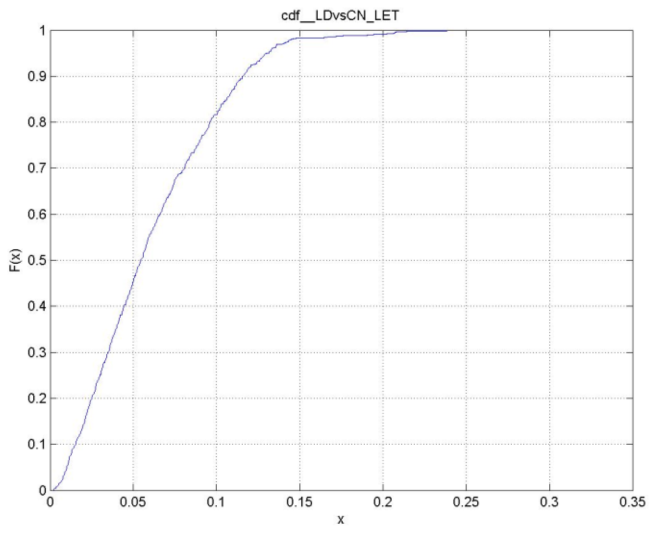

Figure 13 CDF of the time difference between reported link establishment times of the link detection algorithm of the framework and the reported link establishment times calculated with the course change notifier data.

5.3.1 LINK BREAKAGES

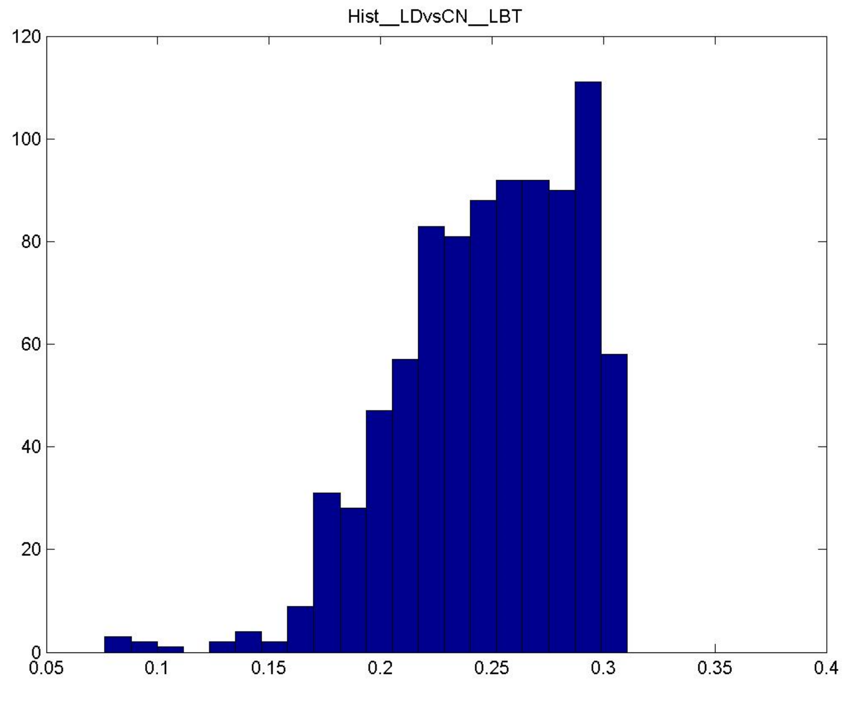

Figure 14 Histogram of the time difference between reported link breakages times of the link detection algorithm of the framework and the reported link breakages times calculated with the course change notifier data.

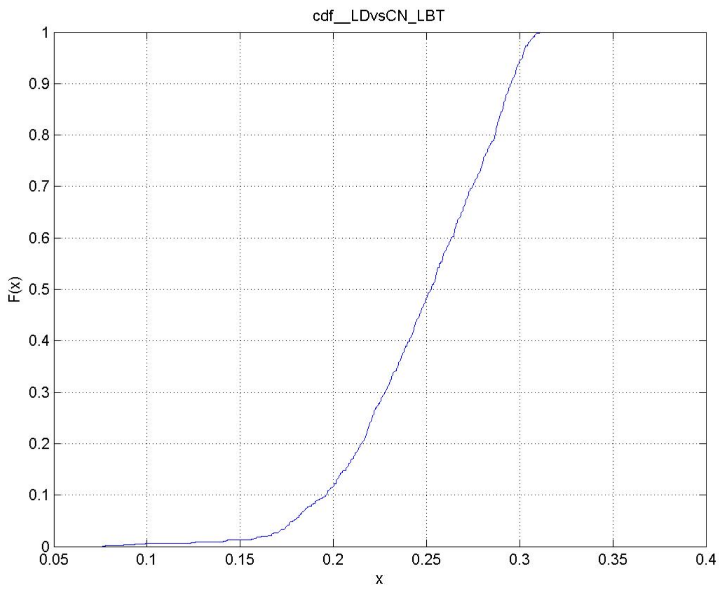

Figure 15 CDF of the time difference between reported link breakages times of the link detection algorithm of the framework and the reported link breakages times calculated with the course change notifier data.

5.4 CURVE FITTING

5.4.1 LINK ESTABLISHMENTS

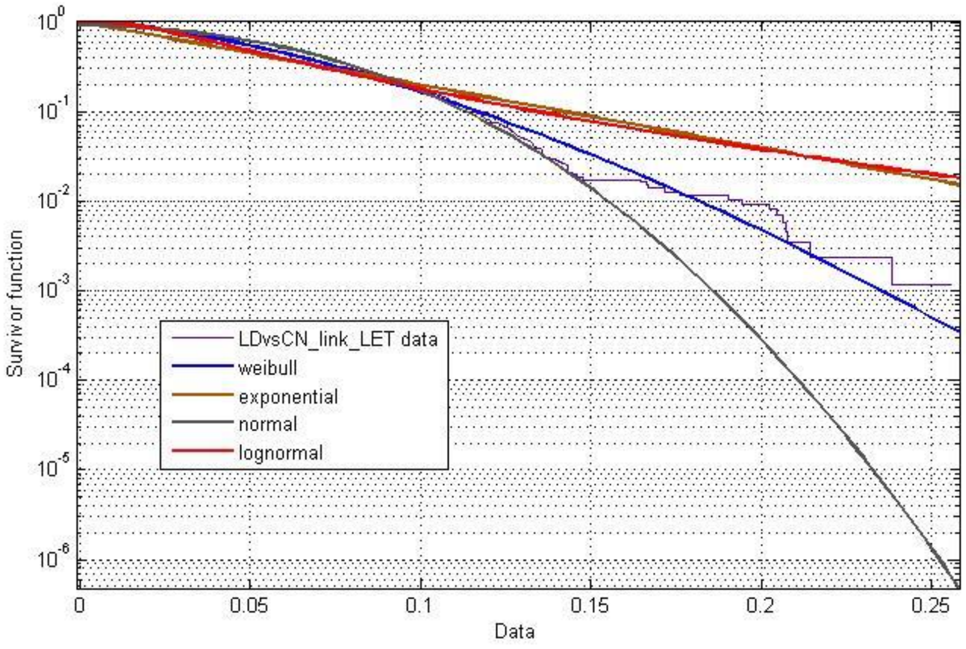

Figure 16 Survivor function (1-CDF) of the reported link establishment time difference between LD and CN together with the various fitted distributions.

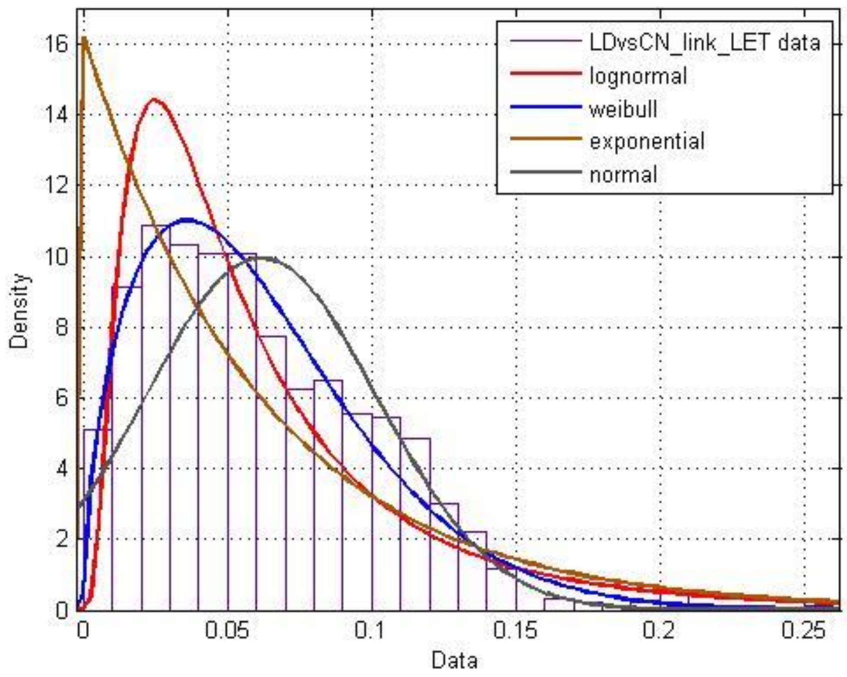

Figure 17 Histogram of the reported link establishment time difference between LD and CN data together with various fitted distributions.

From the graphs it is seems like weibull distribution is a good representation of the data. The fitted parameters for weibull are:

```
Distribution:    Weibull
Log likelihood:  1658.27
Domain:          0 < y < Inf
Mean:            0.0616435
Variance:        0.00161303

Parameter  Estimate          Std. Err.
A          0.0686245         0.00156701
B          1.56885           0.0418823

Estimated covariance of parameter estimates:
           A                 B
A          2.45551e-06       2.05603e-05
B          2.05603e-05       0.00175413
```

Figure 18 Parameters of best fit Weibull distribution for the reported link breakages time difference between LD and CN

Although a Kolmogorov-simirnov test cannot reject that the data is coming from the weibull distribution at 5% significance level, chi^2 goodness of fit test rejects it with a p value of 0.008.

We conclude that although the distribution is close to a weibull. However, further analysis is needed to determine the true underlying distribution.

Since the shape parameter of the weibull distribution is larger than 1, we can also conclude that the distribution does not have heavy tails.

5.4.2 LINK BREAKAGES

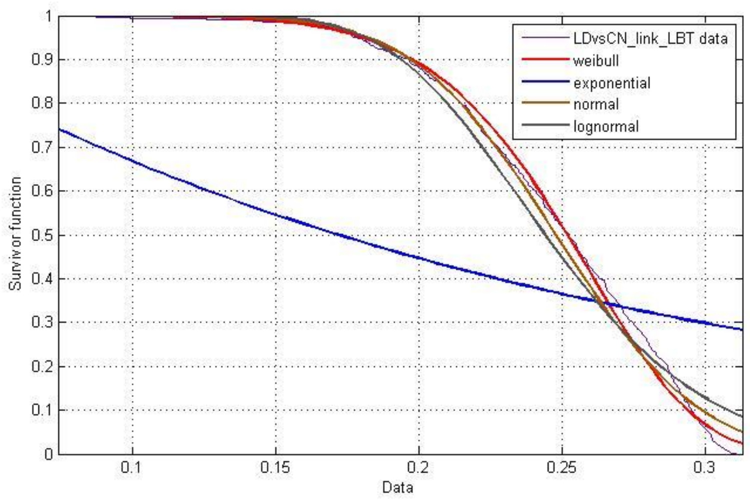

Figure 19 Survivor function (1-CDF) of the reported link breakages time difference between LD and CN together with the various fitted distributions.

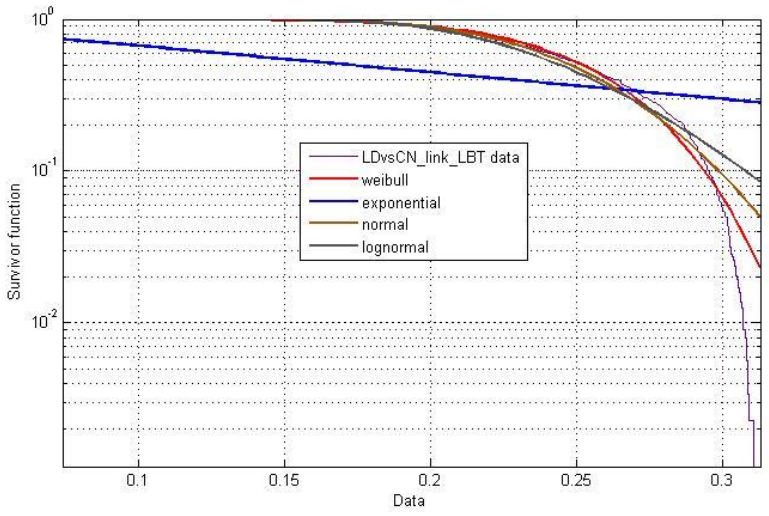

Figure 20 Survivor function (1-CDF) of the reported link breakages time difference between LD and CN together with the various fitted distributions.

```
Distribution:    Weibull
Log likelihood:  1634.8
Domain:          0 < y < Inf
Mean:            0.248175
Variance:        0.00142812

Parameter  Estimate  Std. Err.
A          0.263886  0.00119969
B           7.78056    0.212883

Estimated covariance of parameter estimates:
    A            B
A  1.43925e-06  7.78051e-05
B  7.78051e-05     0.045319
```

Figure 21 Parameters of best fit Weibull distribution for the reported link breakages time difference between LD and CN

Since the shape parameter of the weibull distribution is larger than 1, we can also conclude that the distribution does not have heavy tails. This can also be observed from the faster declining survivor function than the exponential distribution.

6 STATISTICAL ANALYSIS OF THE RANDOM WALK MOTION MODEL

6.1 LINK DURATIONS

The duration of a link is defined as the time difference between the link establishment and the link breakage. In this section, we investigate the link duration statistically. The histogram and the cdf of the link durations are available in Figure 22 and Figure 23, respectively.

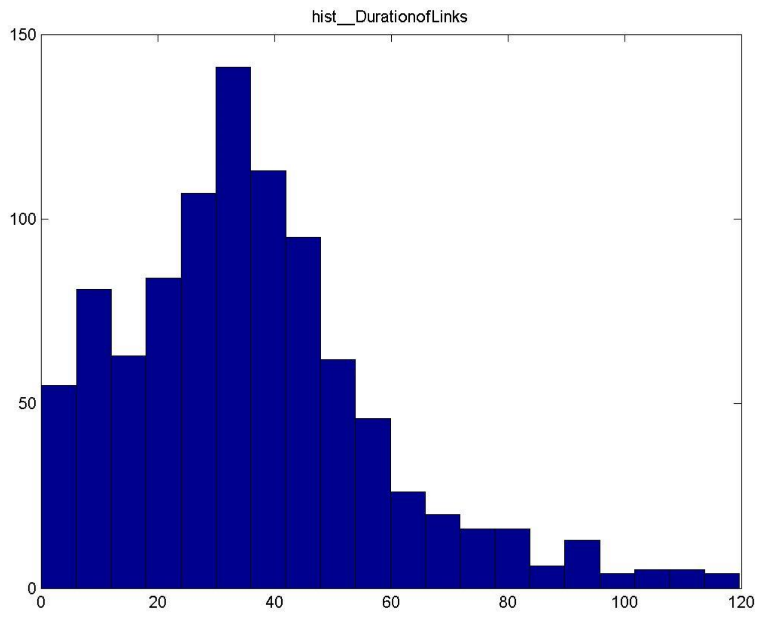

Figure 22 Histogram of Link Durations

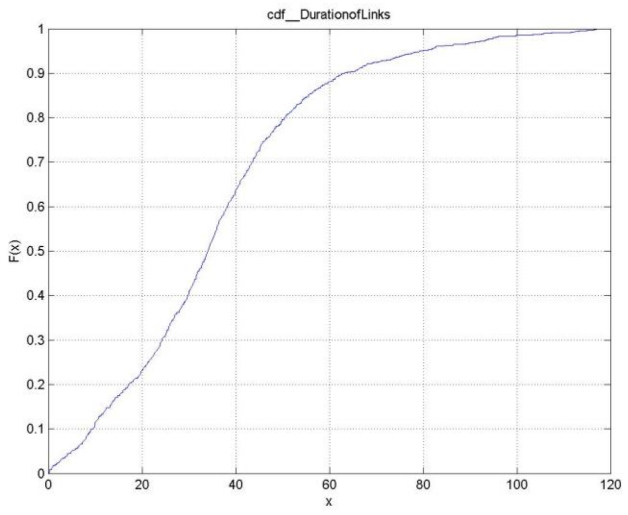

Figure 23 CDF of Link Durations

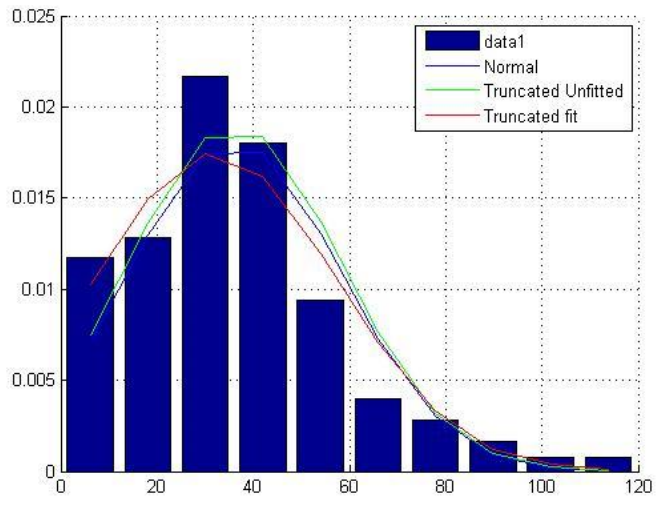

Figure 24 Estimating link durations with truncated normal

6.2 TIME INTERVAL BETWEEN A BREAK AND REFINDING THE NODE

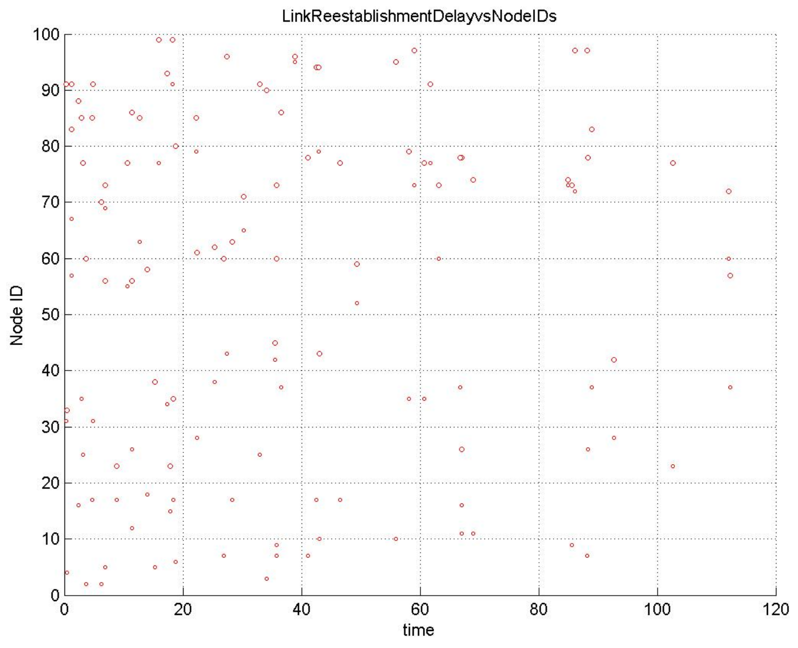

Figure 25 Node IDs vs. Link Reestablishment interval

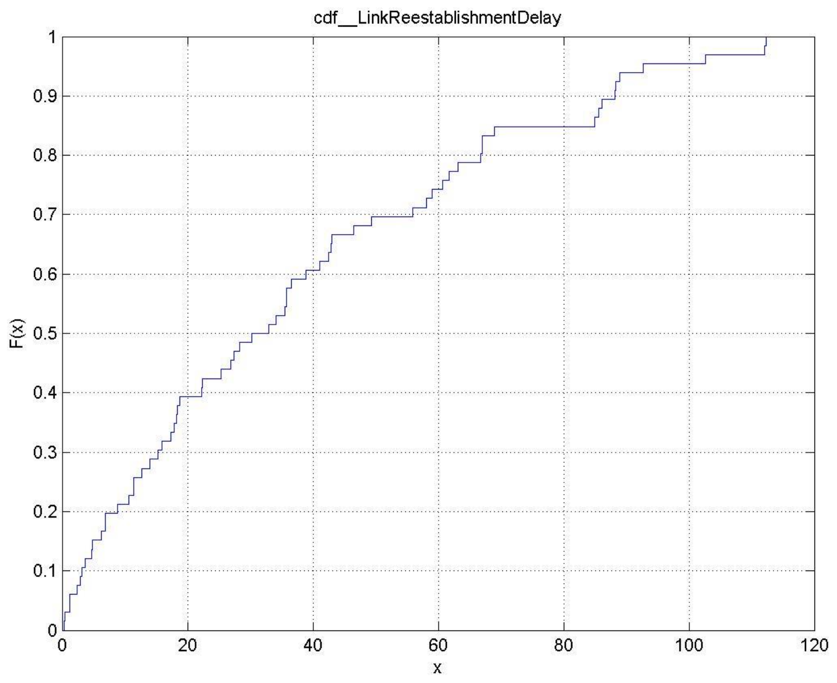

Figure 26 CDF of Link Reestablishment

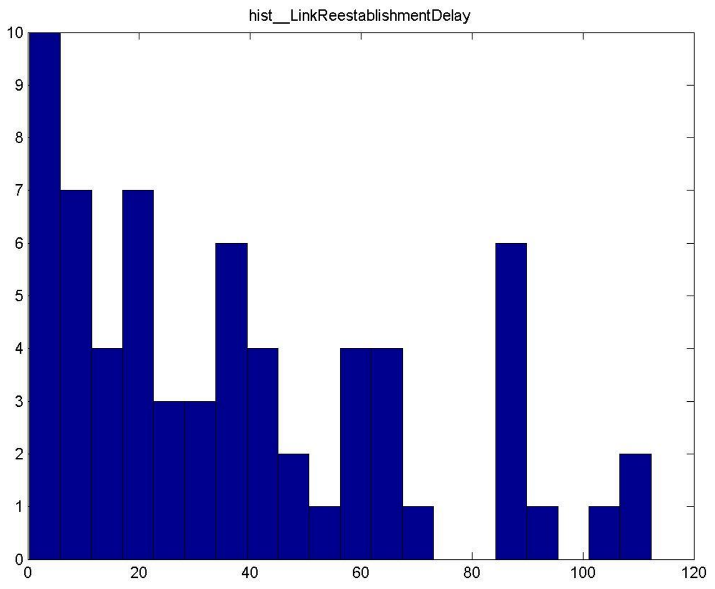

Figure 27 Histogram of Link Reestablishment Delays

6.3 AVERAGE NUMBER OF NEIGHBORS

Figure 28 presents average duration of a node having x many neighbors together with the weighted average number of neighbors which takes the durations as the weight coefficients.

Why do we have a large duration for having 20 neighbors? Check if we have enough samples.

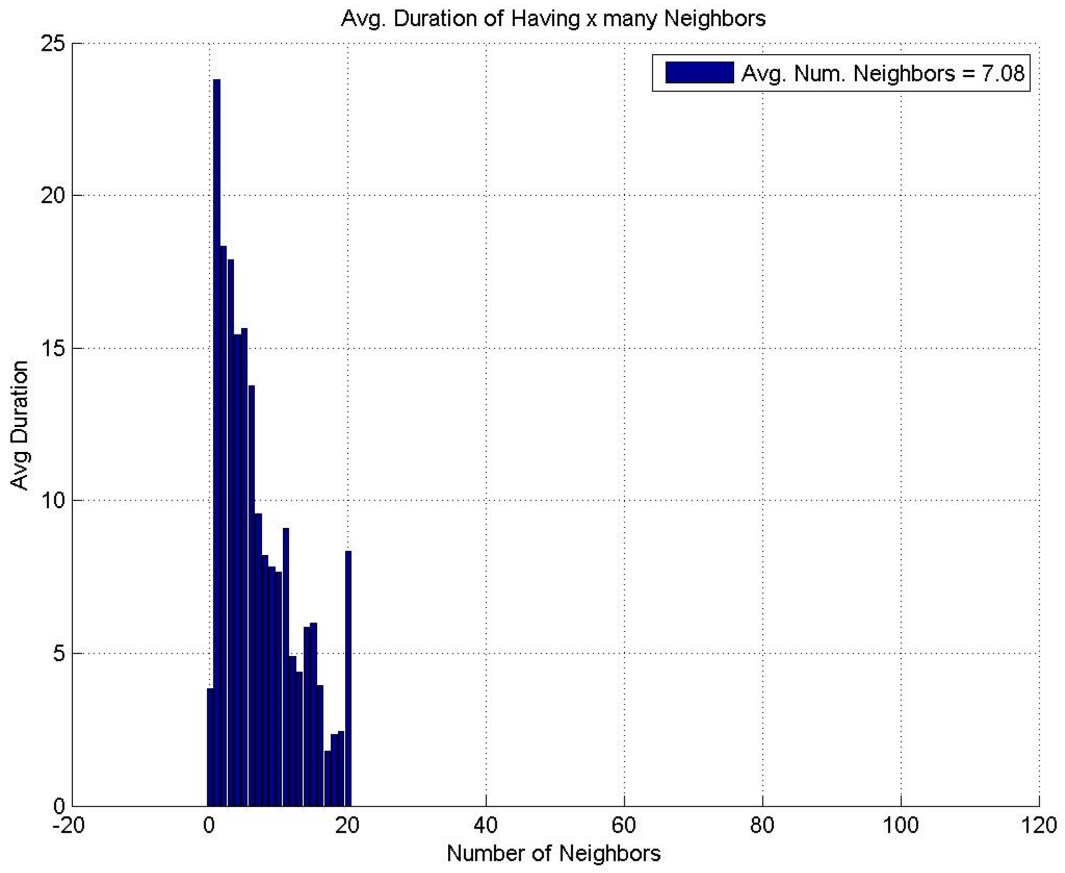

Figure 28 Average Duration of having x many neighbors