

Volatility of Communication Edge to the Robustness of Trusted Spanning Forest

Final Report

Georges Toth

Supervisors: Apivadee Piyatumrong, Guillaume-Jean Herbiet

University of Luxembourg
Faculty of Sciences, Technology and Communication
Master in Information and Computer Sciences
Optimization Techniques
Academic Year 2008/2009

Abstract

Robustness and stability in Delay Tolerant Mobile Ad-Hoc Networks (DTMs) is a crucial issue. Nodes are moving at different speeds, different directions and rely solely on their neighbour nodes for communication support, those who may or may not behave in a cooperative manner. Hence it is important to have solid algorithms for organizing these networks and deciding whom to connect to or not, based on different parameters. The work presented in this paper builds upon an existing algorithm for building trusted spanning trees, and tries to improve on its effectiveness by investigating the introduction of a notion of trust based on edges.

1. Introduction

The research work presented in this paper analyzes delay tolerant mobile ad-hoc networks (DTMs) and how to form stable and robust DTMs.

MANETs represent an emerging and important new field in network technologies, as there is an ever growing number of mobile devices and hence the desire to not only have the classic base-station to client and vice-versa communication model, where the base station is fixed and the clients move, but also client to client communication and the forming of spontaneous mobile networks. The clients of these networks are often using technologies such as WiFi (IEEE802.11) and Bluetooth. What makes these networks so interesting and at the

same time difficult to characterize is that they are formed spontaneously, are self configuring and don't rely on a global infrastructure. The latter also means that the members of a network and the data flow are not controlled or shaped and cannot be influenced on by any kind of infrastructure. Thus these networks rely on the cooperativeness of their nodes. Cooperativeness means that a node must not be selfish, only using services without providing them, like for example using its neighbors for downloading data but refusing to relay packets between it's neighbors.

Another property which makes MANETs hard to describe is their topology, which is only to a certain degree predictable as the networks are highly dynamic, and may experience a complete change within a very short period of time.

DTMs are a subclass of mobile ad-hoc networks, which includes temporary stable connections as well as frequent reorganizations and partitions[4].

Delay tolerance introduces many new obstacles which the underlying protocols have to deal with. What we want to achieve with DTMs is that we have a highly dynamic network, which includes unpredictable connection disruption, and create a trusted spanning forest on top, which is aware of the underlying properties. Thus a major challenge in these networks is to achieve efficient communications. In that context, new algorithms and routing protocols have to be developed in order to allow for a maximum survivability and stability of these networks[1]. It is also important to note that the solution provided to the network has to be computed locally

but yet effective globally.

The work presented in this paper builds upon the G-Trust[4] algorithm, that tries to attribute weight values to the nodes in a network. These values are used for rating the behaviour of the different nodes, and allows for deciding whether it is worth establishing a connection to a specific node or not.

As this resulted in building more stable networks, it was decided to apply the same method on the connection links, i.e. edges. The goal was to better understand what was happening at the edge level, and how one could use that information for further improving the overall performance of the network.

This work studies a number of different metrics applied to both communication edges and nodes, analyzing them and how they can be interpreted and useful for optimizing the connection links. Section 2 explains background information needed to understand the concepts of this work. Section 3 explains the different models which have been used for analyzing the different proposed metrics. Section 4 explains the metrics which have been collected during the simulations. Section 5 describes the tools and environment used during this work. Finally the study is concluded in Section 6.

2. Tree concept

2.1. Tree

In graph theory, a graph is an ordered pair $G := (V, E)$, where V is a set of vertices and E is a set edges. In an undirected graph G , two vertices u and v are said to be connected if there is a path of edges from u to v . Hence a connected graph is a graph where any distinct pair of vertices can be connected through some path. A cycle in a connected graph is path with repeated vertices.

A tree T , is a cycle free graph where any two vertices are connected by exactly one path. Hence any connected graph without cycles is a tree.

2.2. Spanning Tree

A spanning tree of a connected, undirected and cycle free graph G is a tree composed of all the vertices V , and a subset of edges E of G .

2.3. Communication graph

In static networks, spanning trees have long been used for improving robustness and fault-tolerance, but also for improving on routing protocols. The same idea of spanning trees has been applied to dynamic networks

in order to improve different routing algorithms.

In a communication graph G , vertices represent the communication devices. If a vertex u and a vertex v can communicate with each other, the communication link is represented by an edge. Hence the set of edges E , is the set of communication links.

2.4. Spanning Forest

Let $G(t)$ be a communication graph at time t and a pair $V_t(G), E_t(G)$. $G(t)$ is a disjoint set of m connected subgraphs $\{G_1, \dots, G_m\}$. A spanning forest is a graph whose components are trees. Hence a spanning forest embedded in a graph $G(t)$ is a graph $S(t) = (V_t(G), E'_t(G))$ such that $E'_t \subset E_t$, or a set of k spanning trees $\{\gamma_1, \dots, \gamma_k\}$ at moment t of $G(t)$.

3. Mobility Models

While researching on MANETs, the simulation of mobility using mobility models as close as possible to real world environments is crucial. In the literature one can find many different models, of varying complexity, trying to model the different aspects of human mobility[2]. In practice though the random waypoint mobility model is the most commonly used one. To reflect natural human movement, more accurate models have to be used.

In order to realistically simulate the DTMs while applying the G-Trust algorithm, and the improvements to G-Trust, proposed in this work, two real world models were used.

The mobility models used in this work have been generated using the metropolitan ad hoc network simulator (Madhoc[2]).

The first model tries to reflect a highway environment, where we have multiple lanes with cars moving alongside and in opposing directions, and with different speeds. While moving, they try to establish connections to their neighbors and keep those connections alive.

The second model discussed in this paper simulates a shopping mall environment. Here we have a large space where we mimic the movement of people moving around in this shopping area and trying to establish connections with each other.

4. Metrics

To be able to monitor and understand the behaviour of a simulation, metrics are collected. These metrics allow for better understanding the simulation model, verify the performance of the applied algorithms and protocols, and how to possibly find new ways for improving

them.

The following sub sections will give an overview of the different metrics used in this work. We have metrics for nodes, edges and trusted spanning tree links. Most of them (unless specified) are based on the same definition (e.g. *node :: averageAge()*, *edge :: averageAge()*), hence they are no separately explained.

The following introduced metrics are based on the work of [3] and [4]

4.1. Appearances

A counter variable is used for collecting this metric, and it is defined as follows. If an element¹ changes its state from non-existence to existence, a counter variable associated with that specific element gets incremented by one; hence this is a local and not a global variable.

The optimum for this metric is a constant value of one, because that means that we have only one appearance of all edges over the simulation period.

This metric is very useful for distinguishing the behaviour of different mobility models. As we can see while comparing the results for the highway and mall mobility model, the value of this metric climbs much faster for the former one. This can be related to the nature of this mobility model, which is defined as using high speeds for the nodes, opposing directions and non-constant speeds.

Figure 1 shows an illustrative plot of this metric.

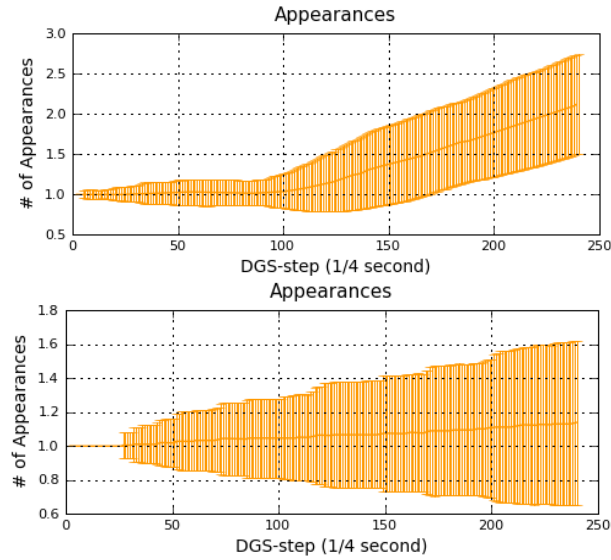


Figure 1. appearances(), top: highway, bottom: mall

¹edge, node, trusted spanning tree link

4.2. Accumulated Age

For a given element e , the accumulated age is defined as $accumulatedAge(e) = \sum_{i=0}^n t_{present_i}$, hence the sum of the time intervals the element has existed during the simulation. For the implementation of this metric, a counter variable was used which gets incremented after every simulation step, if the element has been active² for that previous step.

The optimal value for this metric is the total time of the simulation as this means that the observed element have been active during the whole simulation period. When summing this metric over all similar elements (e.g. all edges), the steeper the values increase, the more stable the network can be interpreted to be. If they increase only steadily, more links break and hence the network is less stable.

Figure 2 shows an illustrative plot of this metric.

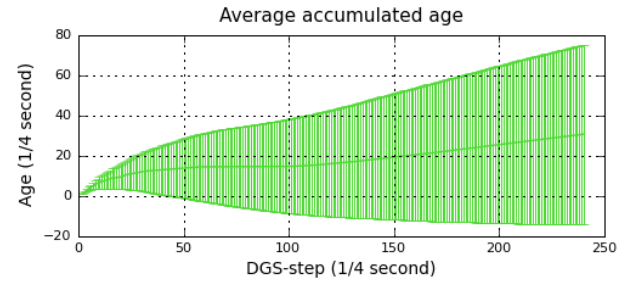


Figure 2. accumulatedAge()

4.3. Average Age

The average age of an element e is defined as $averageAge(e) = \frac{accumulatedAge(e)}{appearances(e)}$. This metric is based on two previously defined metrics. After each simulation step it is being calculated and saved.

The optimal value of this metric is, as with the accumulated age, the total time of the simulation, which means that the element has been active over the whole simulation. If the observer value stable or decreasing, that means that the network is rather unstable, if it is on the other hand constantly increasing this means that the network gets more stable.

Figure 3 shows an illustrative plot of this metric.

4.4. Number of elements

The number of elements (e.g. edges, nodes or neighbours) represents the number of active elements at

²Active means the same as existence, i.e. the element was actively present in the simulation.

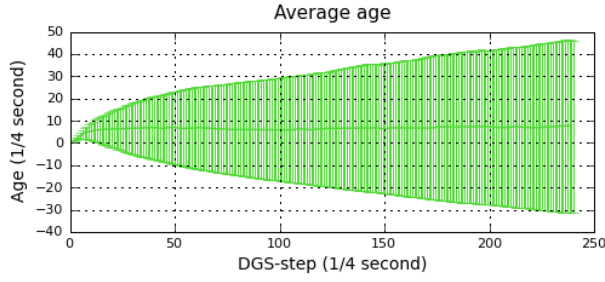


Figure 3. averageAge()

a specific simulation step. The meaning of this metric depends on the type of element being represented.

- Nodes: A high value means that there are a lot of nodes active.
- Neighbours: A high value means that the nodes have many connected neighbours.
- Edges: A high value means that there are a lot of connections between the nodes.

There is no optimal high or low value for this metric, but a constant value might indicate that the links in the network are more stable, compared to a highly fluctuating value.

Figure 4 shows an illustrative plot of this metric.

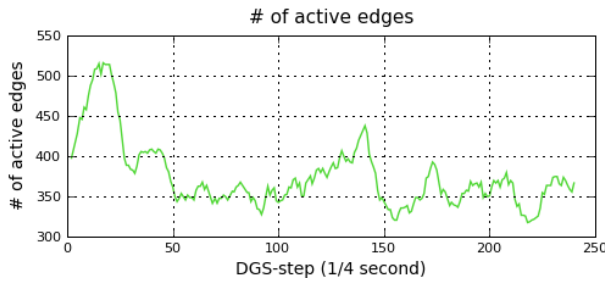


Figure 4. numberActiveEdges()

4.5. Volatility

The volatility of an element is defined as $volatility(e) = \frac{appearances(e)}{accumulatedAge(e)}$. It should be noted that this metric is the inverse of the *averageAge()* metric.

The optimal value of this measure, tends to zero. This can be explained in that it is optimal to have a as high as possible value for *accumulatedAge()* and a as low as possible value for *appearances()*, which evidently leads to that statement. In this sense, a lower value means that an element has been active for a longer

time with a few appearances, hence summed over all related elements (e.g. edges), this gives a measure about the stability of the network.

Figure 5 shows an illustrative plot of this metric.

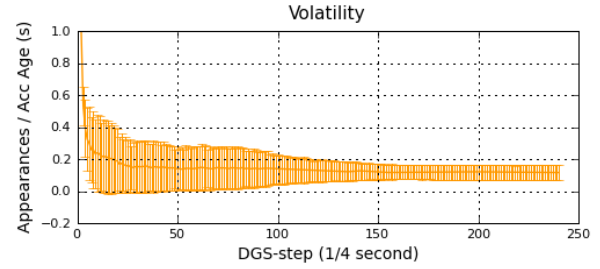


Figure 5. volatility()

4.6. Number of new neighbours

This metric is only being used for nodes, in order to keep track of the number of neighbour nodes they have. Regarding the implementation, every node keeps records of what nodes they have connected to, and each time they connect to a new node, they increment a counter.

This metric gives an overview on how many neighbours the nodes have on average over a simulation. Our results show that for example for the highway mobility model, all nodes have seen each other after a certain period of time.

Figure 6 shows an illustrative plot of this metric.

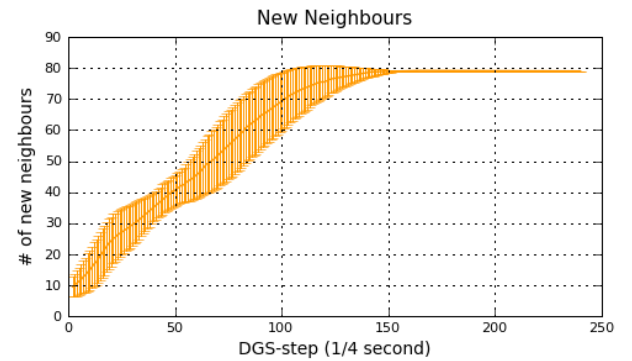


Figure 6. numberNewNeighbors()

4.7. Cumulative Distribution Function

The cumulative distribution function describes the distribution of the nodes over the simulation. The CDF is defined as follows. For any average age x , the CDF of x gives the number of nodes having an average age smaller equal x .

As per the definition of this metric, it gives us the distribution of all the nodes over a simulation based on their average age.

Figure 9 and 10 show plots of this metric.

4.8. Performance Ratio

This metric is used based on a graph. The performance ratio for a graph G at time t , is defined as $performanceRatio(G(t)) = \left(\frac{\Gamma}{m}\right)$, where Γ is the number of subgraphs and m is the number of connected components.

This metric is used as a quality assessment for the modified algorithm. The optimal value for the $performanceRatio()$ method is 1, as this means that there is only one subgraph and one connected component.

Figure 7 shows an illustrative plot of this metric.

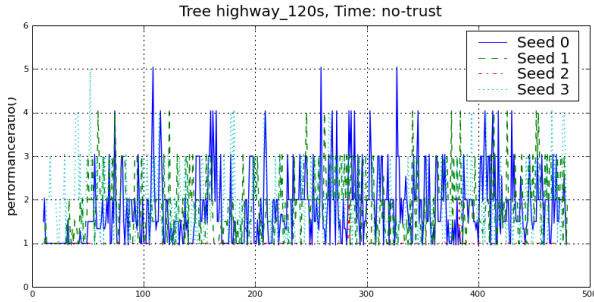


Figure 7. $performanceRatio()$

4.9. Tree Link Disappearances

This metric is used on the trusted spanning tree created by G-Trust. A tree link, i.e. edge, of the trusted spanning tree disappears if it changes its state from active to inactive. In other words, considering an edge e_{ab} between two nodes a and b at time t_1 . If this connection breaks at time t_2 , the tree link effectively disappears.

This metric is implemented as a counter variable which gets incremented for a given tree link, if it disappears.

This metric gives how many disappearances the observed trusted spanning tree has over a simulation. The optimal value for this measure should be as low as possible as this means that the trusted spanning tree is stable. High values indicate an unstable network.

Figure 8 shows an illustrative plot of this metric.

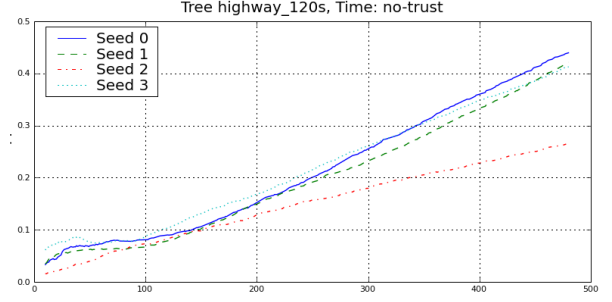


Figure 8. $treelinkDisappearances()$

	Shopping Mall	Highway
Surface (km^2)	0.32	1.0
Node Density (per km^2)	1000	80
Number of Nodes	100	80
Avg. Number of Partitions	2.68	1.7
Number of Connections	389	405
Average Degrees	7.82	10.17
Velocity of Nodes ($\frac{m}{s}$)	0.3-3	20-40
Radio Transmission Range	40-80 m	

Table 1. Parametrization used in Madhoc

5. Simulation environment

In order to simulate DTMs, the used mobility models should reflect real-world environments as close as possible, which includes the lay-out of nodes (e.g. citizens or cars), environmental properties (e.g. obstacles, limited paths, roads) and radio propagation (communication link)[4]. The networks used in this work were generated using Madhoc [2], an ad-hoc networks simulator (introduced in 3) that provides mobility models allowing realistic motion of nodes (here citizens in the mall model and cars in the highway model) in a variety of environments. The parameters for the two mobility models, “shopping mall” and “highway” (introduced in 3) are summarized in Table 1. The generated networks were saved in the DGS³ file format. Each simulation step (i.e. DGS-step) represents a time interval of $\frac{1}{4}s$.

The actual simulations were carried out using the GraphStream library. GraphStream is an event based Java library for construction dynamic graphs. For simulating the evolution of a graph, it supports different

DGS is a file format allowing to store graphs and dynamic graphs in a textual human readable way, yet with a small size allowing to store large graphs. Graph dynamics is defined using events like adding, deleting or changing a node or edge. With DGS, graphs will therefore be seen as stream of such events.

events like *elementAdded()*, *elementRemoved()* and *elementChanged()*. Besides that, it also allows for assigning arbitrary attributes to the elements of the graph.

The latter property is for example used for assigning trust values to nodes and edges.

The simulations for both mobility models (“shopping mall” and “highway”) were done using 100 runs and for a time period of 120s, which equals to 480DGS – steps.

5.1. Experimentation Steps

The work was started with implementing different Java classes for collecting the metrics described in section 4 and writing that data, depending on the context, per DGS-step or per simulation run, out to files. Those files were then used for generating the graphs using custom written scripts and the matplotlib⁴ Python graphing library. Those graphs were then used for analyzing the behaviour of the algorithm.

The CDF (see 4.7) metric was chosen to be used as link-stability value for deciding whether to establish a connection between two nodes. The idea of choosing this metric was to wait for a certain amount of DGS-steps, to allow the nodes to stabilize with respect to their neighbours, and only then form new connections. The reason for this to further help stabilizing networks was that if we don’t allow to form new links if the the opposing node hasn’t reached a certain threshold, we can avoid to create too many short-lived connections.

The experimentation continued with applying the chosen threshold values followed by an analysis of the results.

5.2. Results

Experimentation has been done with applying threshold values to the algorithm, starting from 5% up to 95%, in 5% increments. The percentage values translate to a specific average age value the nodes have to satisfy in order for connections to be formed. Table 2 summarizes the threshold values for the highway mobility model, table 3 the values for the mall mobility model. Figure 9 and 10 illustrate the CDF plot for both mobility models, which were used for determining the used threshold values.

For assessing the quality of the algorithm after having applied the previously presented threshold values, three functions (*performanceRatio()*, *treelinkAverageAge()*, *treelinkDisappearances()*), introduced in section 4 were used.

threshold()	averageAge()
95%	4.73
90%	5.1
85%	5.43
70-80%	6.1
65%	6.43
60%	6.77
45-55%	7.1
40%	7.77
30-35%	8.1
25%	8.77
20%	9.1
15%	10.1
10%	12.77
5%	108.77

Table 2. “highway” threshold values

threshold()	averageAge()
95%	9.6
90%	23.5
85%	39.0
80%	56.5
75%	76.55
70%	97.35
65%	117.6
60%	138.1
55%	160.6
50%	181.35
45%	204.1
40%	237.85
35%	272.08
30%	306.8
25%	356.75
20%	416.25
15%	460.5
10%	478.0

Table 3. “mall” threshold values

⁴<http://matplotlib.sourceforge.net>

Figures 11 and 12 both show the plots of the *performanceRatio()* function, of the respective models. Figures 13 and 14 both show the plots of the *treelinkAverageAge()* function, of the respective models. Figures 15 and 16 both show the plots of the *treelinkDisappearances()* function, of the respective models.

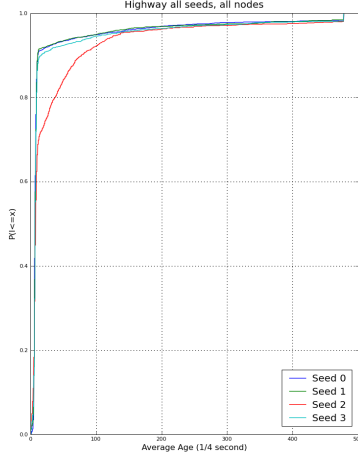


Figure 9. “highway” CDF plot

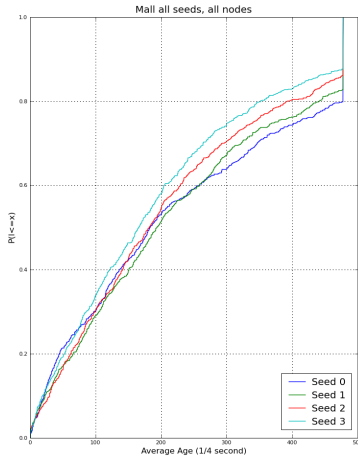


Figure 10. “mall” CDF plot

For the experimentation four different seeds⁵ were analyzed for each mobility model. The plots of the quality assessment functions don’t include “Seed 2” of the highway mobility model as it’s behaviour was too different⁶ from the remaining 3 seeds and thus found to be unsuited for including in the quality assessment.

Analyzing the plots and further analyzing the raw results of the executed experiments indicates that there

⁵In the plots marked as “Seed 0”, “Seed 1”, etc.

⁶There was probably a parameter wrongly set in Madhoc.

has been no stability improvement from applying the different threshold values. The values of the quality assessment functions fluctuate around the same values, hence show neither a consistent improvement nor regression.

6. Conclusion and Perspectives

The objective of this work was to extend the G-Trust algorithm by introducing model specific threshold values which ought to be used for keeping connections from being formed unless the involved nodes have stabilized within the range of their connection partners. The notion of trust has been applied to the edges, which in this work translates to the average age of an edge, and is being checked against the applied threshold value. Further the collection of many different metrics have been implemented, based on edges, nodes and trusted spanning tree. This work ended with the conclusion that the way the threshold was chosen did not yield any stability improvement, thus further investigation on how to construct stronger networks in respect to edges is needed.

In terms of perspectives, future work may include longer simulation times than those used in this work (i.e. 120s), which might show improvements. Including new mobility models, and the introduction of new metrics might be interesting. A proposal for a new metric for influencing the trust value for edges, is the Signal/Noise ratio, which gives the distance between two nodes. The closer two nodes are, the higher the throughput might be, hence those connections should be favoured over connections to nodes with a higher distance.

7. Acknowledgment

The author gratefully acknowledges the continuous help and support during this work, by his two supervisors Apivadee Piyatumrong and Guillaume-Jean Herbiet.

References

- [1] Vassilis Tsaoussidis Giorgos Papastergiou, Ioannis Psaras. Deep-space transport protocol: A novel transport scheme for space dtns. *Computer Communications*, 2009.
- [2] Hogie Luc and Bouvry Pascal & Guinand F. *The Madhoc simulator* <http://www-lih.univ-lehavre.fr/~hogie/madhoc>. PhD thesis, April 2007.
- [3] Yoann Pigné. *Modélisation et Traitement Décentralisé des Graphes Dynamiques - Application Aux Réseaux Mobiles Ad Hoc*. PhD thesis, L’Université du Harve, December 2008.

- [4] Apivadee Piyatumrong, Pascal Bouvry, Frederic Guinand, and Kittichai Lavangnananda. Trusted spanning trees for delay tolerant mobile ad hoc networks. In *2008 IEEE Conference on Soft Computing in Industrial Applications (SMCia/08)*, pages 131–136, June 2008.

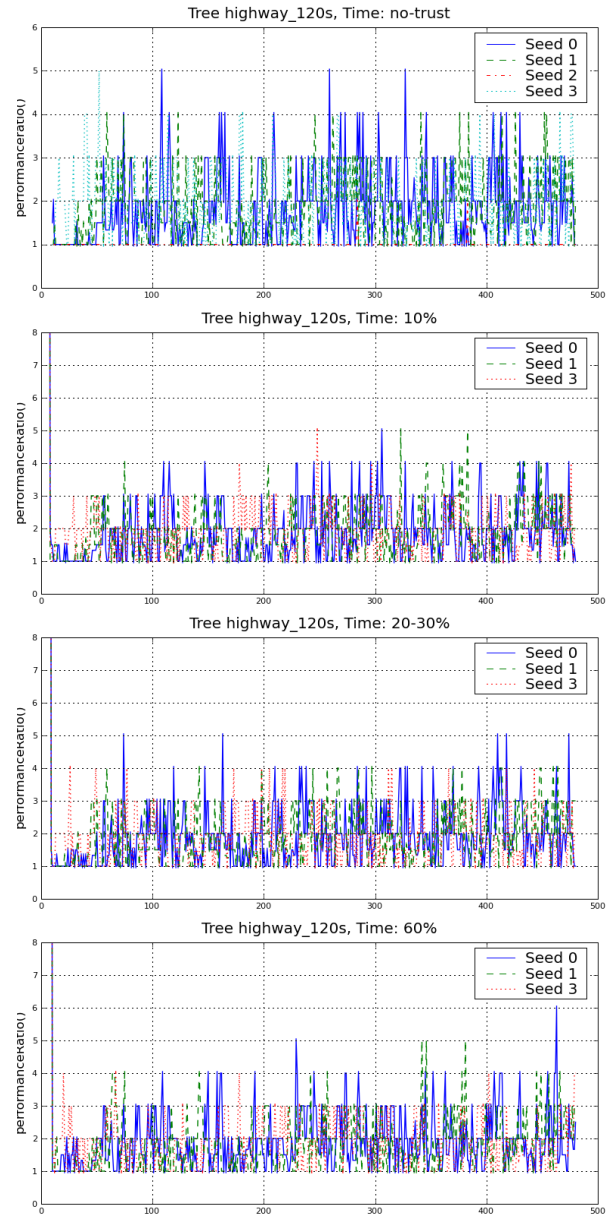


Figure 11. “highway” *performanceRatio()* (no threshold, 90%, 70%, 40%)

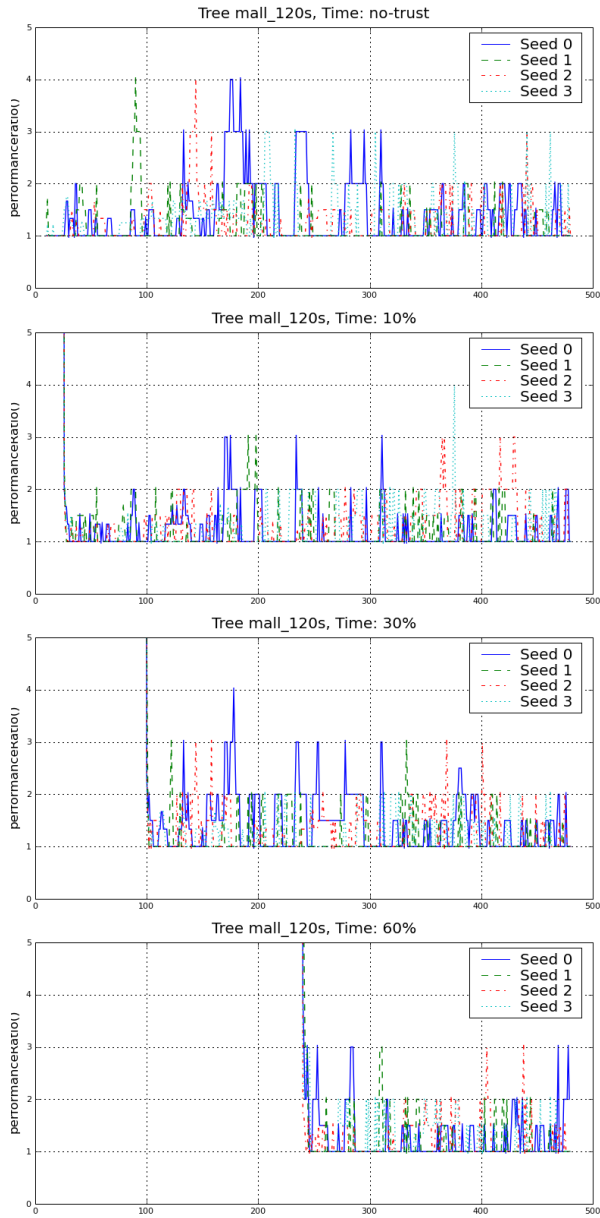


Figure 12. “mall” *performanceRatio()* (no threshold, 90%, 70%, 40%)

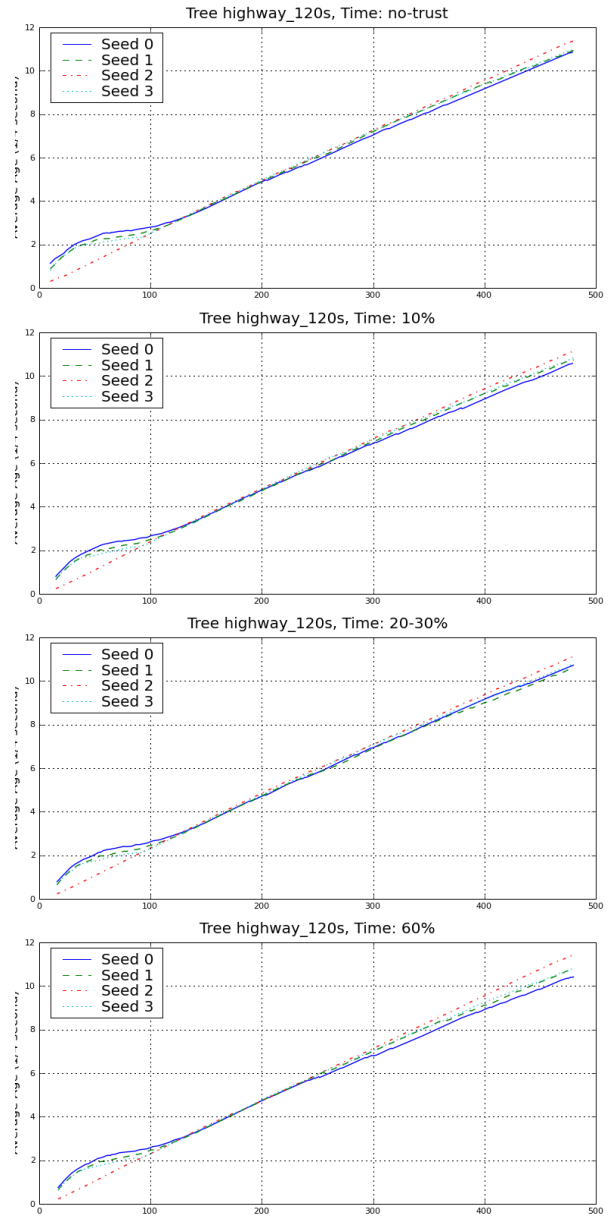


Figure 13. “highway” *treelinkAverageAge()* (no threshold, 90%, 70%, 40%)

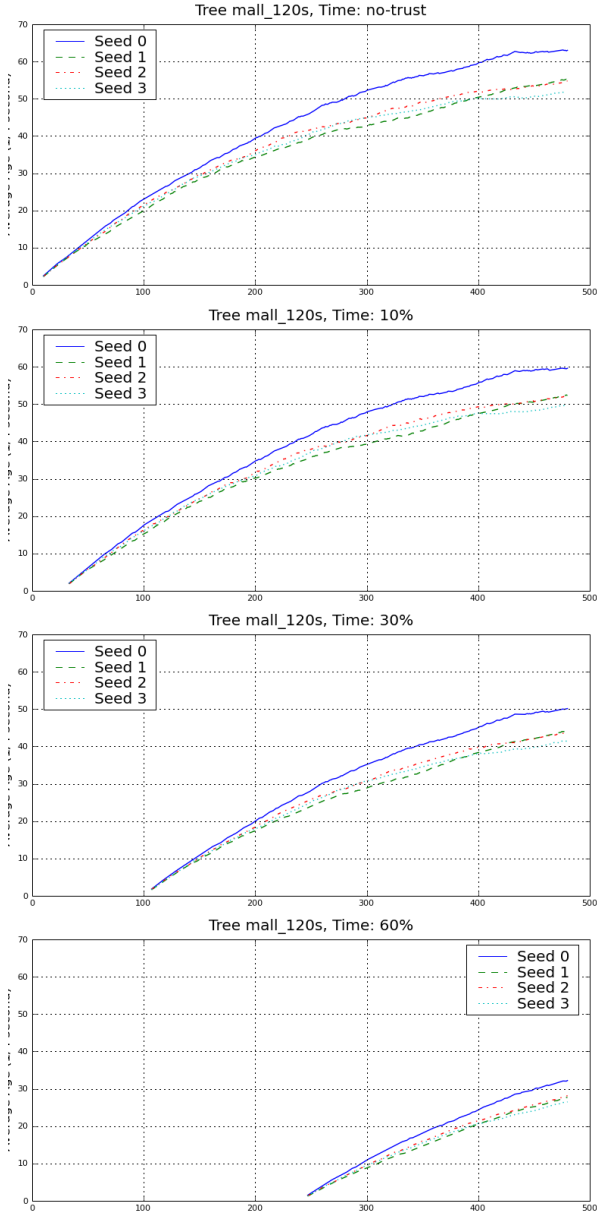


Figure 14. “mall” *treelinkAverageAge()* (no threshold, 90%, 70%, 40%)

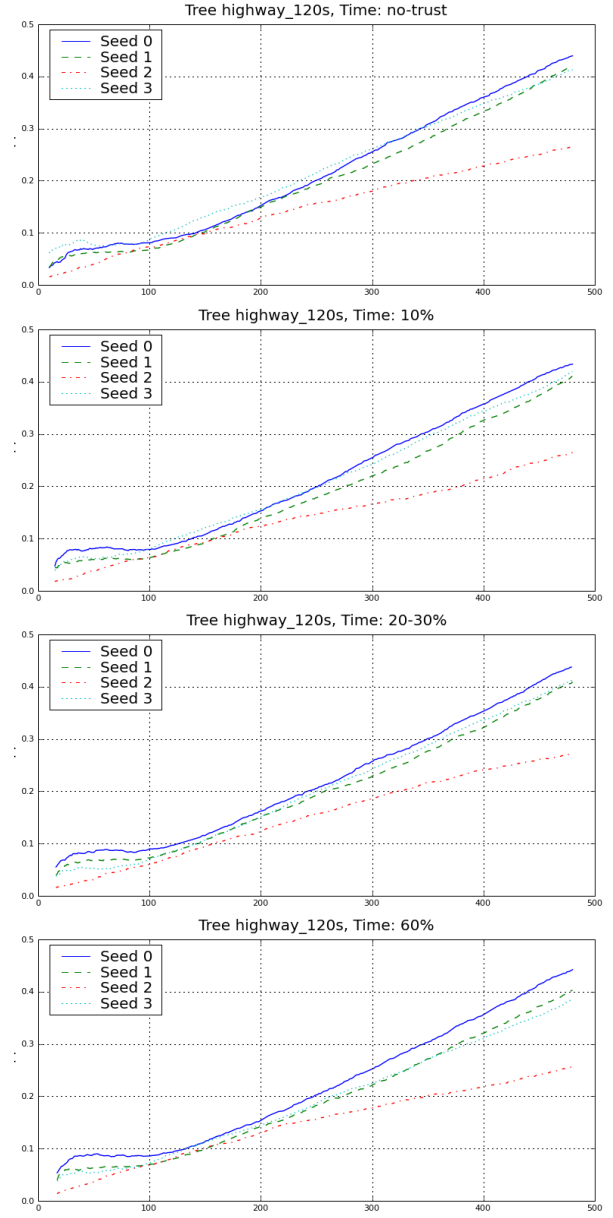


Figure 15. “highway” *treelinkDisappearances()* (no threshold, 90%, 70%, 40%)

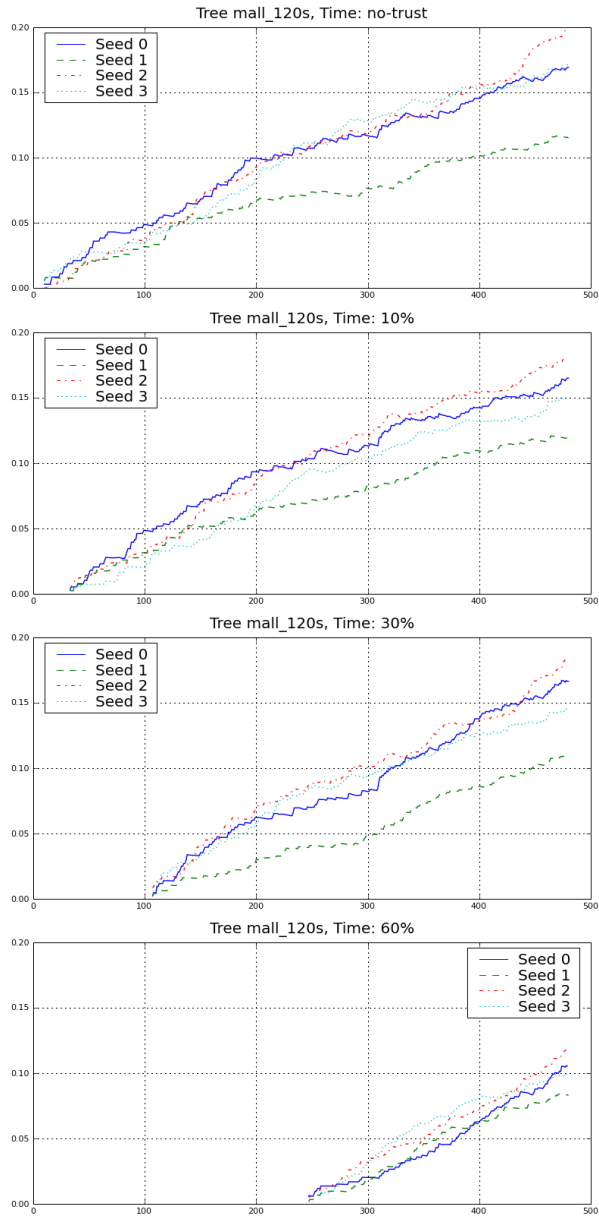


Figure 16. “mall” *treelinkDisappearances()* (no threshold, 90%, 70%, 40%)