On the Performance of Centralized Task Graph Protocol*

W. Ke, P. Basu, and T.D.C. Little
Department of Electrical and Computer Engineering, Boston University
8 Saint Mary's St., Boston, MA 02215, USA
Ph: 617-353-8042, Fax: 617-353-1282
{*ke,pbasu,tdcl*}*@bu.edu*

MCL Technical Report No. 04-30-2003

Abstract–With the increase in the number of computing devices with networking capacity in our day-to-day environment, much research effort has been spent trying to tap into the full potential of the resources available. We propose the introduction of a Task layer in the devices. Such layer will abstract the resources available locally into a pre-defined task description language which can then be understood by the community of users and be requested and shared when needed.

We developed a centralized version of the Task protocol that can support the resource discovery and management required, in addition to supporting mobility management, thus making our protocol suitable in mobile ad-hoc networks (MANET) scenarios as well. We conducted performance studies of our centralized protocol and we found that: (1) Variable time-out values (which depend on the number of nodes in the TG) during resource selection process is necessary to successfully complete the process, (2) connectivity metrics that do not take application level traffic patterns into consideration may give inaccurate estimates of application throughput performance, and (3) procedures that control and estimate the probability of a task graph successfully selecting enough resources are necessary to reduce redundant traffic that keeps attempting when there are not enough resources available.

Keywords: Protocol Performance Evaluation, Ad-Hoc Networks, Application Protocols

*This work was supported by the NSF under grant No. ANI-0073843. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

One paradigm that guided the design of computer network protocols in the past has been the association of specific addresses to hosts, by which these hosts can be reached. Therefore, protocols were concerned with host discovery, and routing of data packets to specific hosts (i.e., addresses).

However, reaching a specific host is only a means to an end, which is to request a service from the host. With the increase in the number of computing devices in the environment that surrounds us, services which we want can be fulfilled by many different hosts. Therefore, currently, research effort has been directed at how to specify applications in terms of the services needed, which are then automatically transformed into available hosts [1, 2, 3, 4, 5, 6].

We propose modeling an application in terms of a Task Graph (TG), in which the nodes are computing devices which satisfy certain attributes, and edges are communication channels between the nodes which satisfy certain QoS requirements. With the application specified as a TG, a new Task layer would be introduced in the computing devices that would select proper devices and join them together to execute the application, as well as manage any possible failure during execution.

We proposed two protocols, one centralized [7] and one distributed [8]. For a detailed explanation of the distributed protocol please refer to [9]. This article details performance studies of the centralized protocol. For a more complete description of the centralized protocol please refer to [7].

In this performance study we simulated for scenarios of 100 nodes roaming in square regions of area 1 km^2, 2 km^2 and 4 km^2. The nodes were under the Random Waypoint (RWP) mobility model [10]. The maximum speed $V_{max} \in \{1, 5, 10, 15, 20\}$ m/s and the pause time $P \in \{0, 100\}$. We studied the performance of the protocol for task graphs (TG) having a binary tree format, with the number of nodes in the TG $N \in \{3, 7, 15, 31, 63\}$.

We report on Instantiation Time and Dilation metrics in Sec. 2. Dilation gives us a sense of how far spread an instantiated TG application is, and we correlate that observation with instantiation times. Sec. 3 reports Re-Instantiation Times and number of Re-Instantiations that took place during the simulations. These metrics offer a picture of the performance of the recovery process.

Moving on to metrics that affect the application we show results for delay, throughput, average number of hops traversed and percentage of connected time in Sec. 4. These are indicators of how well an application may perform when executed on top of our centralized Task protocol. The final metric we explore is the number of overhead packets used in our protocol in Sec. 5. We

conclude with some observations regarding design of centralized protocols in MANETs in Sec. 6

2 Instantiation Times, Dilation and Time-Out

In this section we study the behavior for the instantiation times of the different TGs, the dilation of the embedded task graph and the implications for time-out values. By instantiation time we mean the period of discovery and selection phases as described in [7]. Figs. 1, 2 and 3 show the scatter plot of the instantiation times.

We can see in Figs. 1 and 2 that for small number of nodes in the TG (≤ 15) many values are less than 10 seconds, independent of whether the nodes were in constant movement (pause time 0) or having pause times of 100 s. In particular, with 3 nodes (and less distinctively with 7 nodes), we can see a bi-modal behavior, in which the instantiation time seems to be either less than 1 second or around 3 seconds. This is the same phenomena as mentioned by [9], the 3 second TCP time-out value being the cause for the 3 second gap in instantiation times[1].

Performance degradation starts when the node density is low (see Fig. 3). Another crucial factor in performance degradation is the increase in the number of nodes in the TG. In particular, for the number of nodes equal to 63, there are many instances of instantiation failures, as described in the figures. We can see that instantiation failures also start with TG with 31 nodes at low density (Fig. 3).

The centralized protocol assumes the existence of only one controller, which is responsible for keeping track of the state of the task graph nodes. This means that, for the 31 node case, one node (the controller) is keeping 30 TCP connections while for the 63 node case, 62 TCP connections. The time-out value of the task layer protocol is 14 seconds. This implies that for the instantiation to be completed, within a period of 14 seconds, all 30 (or 62) nodes must have successfully received an instantiation packet from the controller and have their acknowledgment packets successfully arrive at the controller. Given that TCP performance drops drastically with increased number of hops [11], and that the dilation of the instantiated task graph is larger than one (see Figs. 4–6), the time-out value of 14 seconds is not sufficient for the instantiation to be successful.

Regarding Figs. 4–6, some observations can be made. Mobility does influence dilation. It is interesting to see that the average dilation is not 1 even for really small task graphs (3 or 7

[1] If there is a packet loss in the TCP connection, there is a 3 second time-out value before a retransmission is attempted.

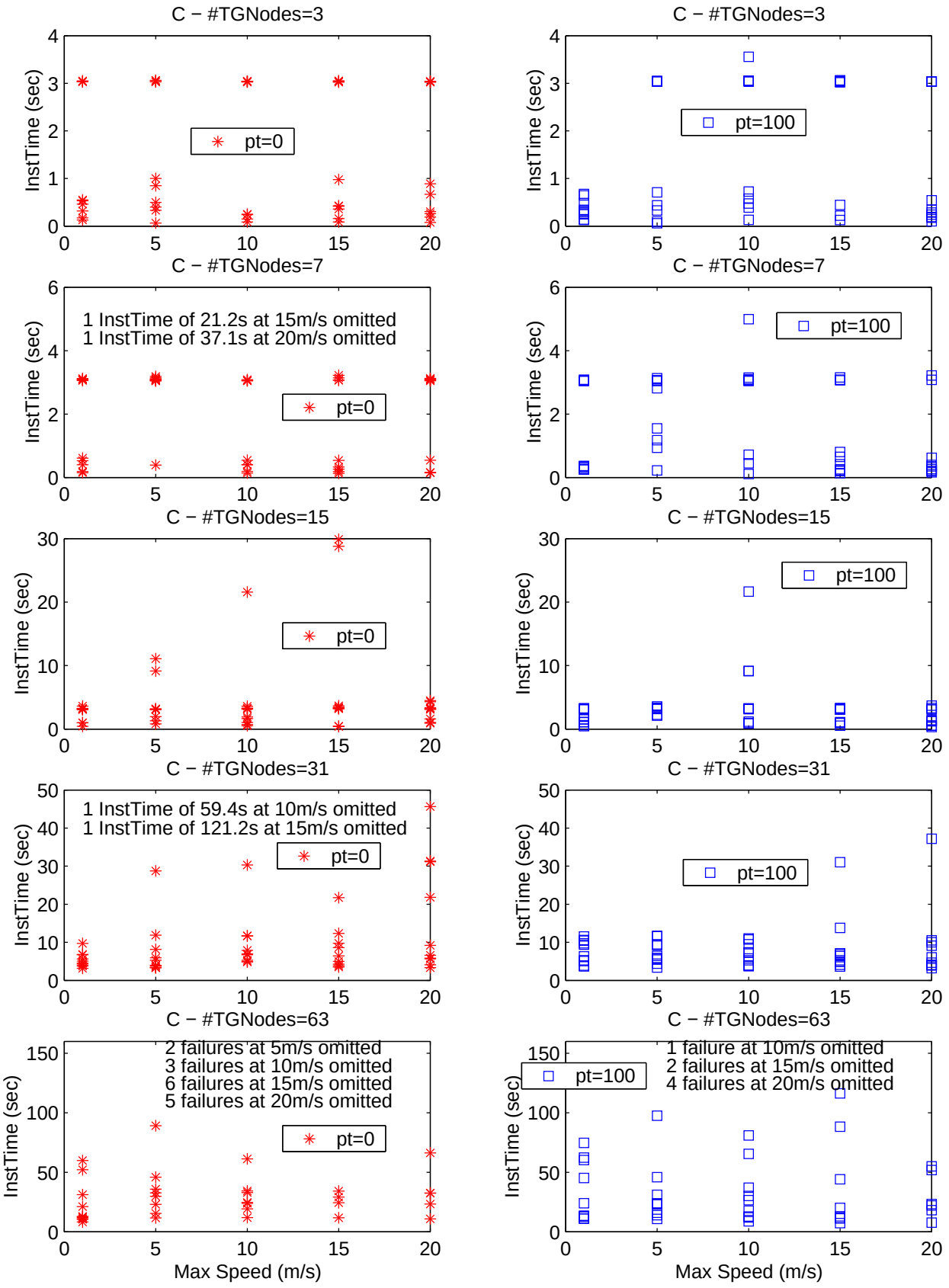

Figure 1: Instantiation Times Area = 1 km^2.

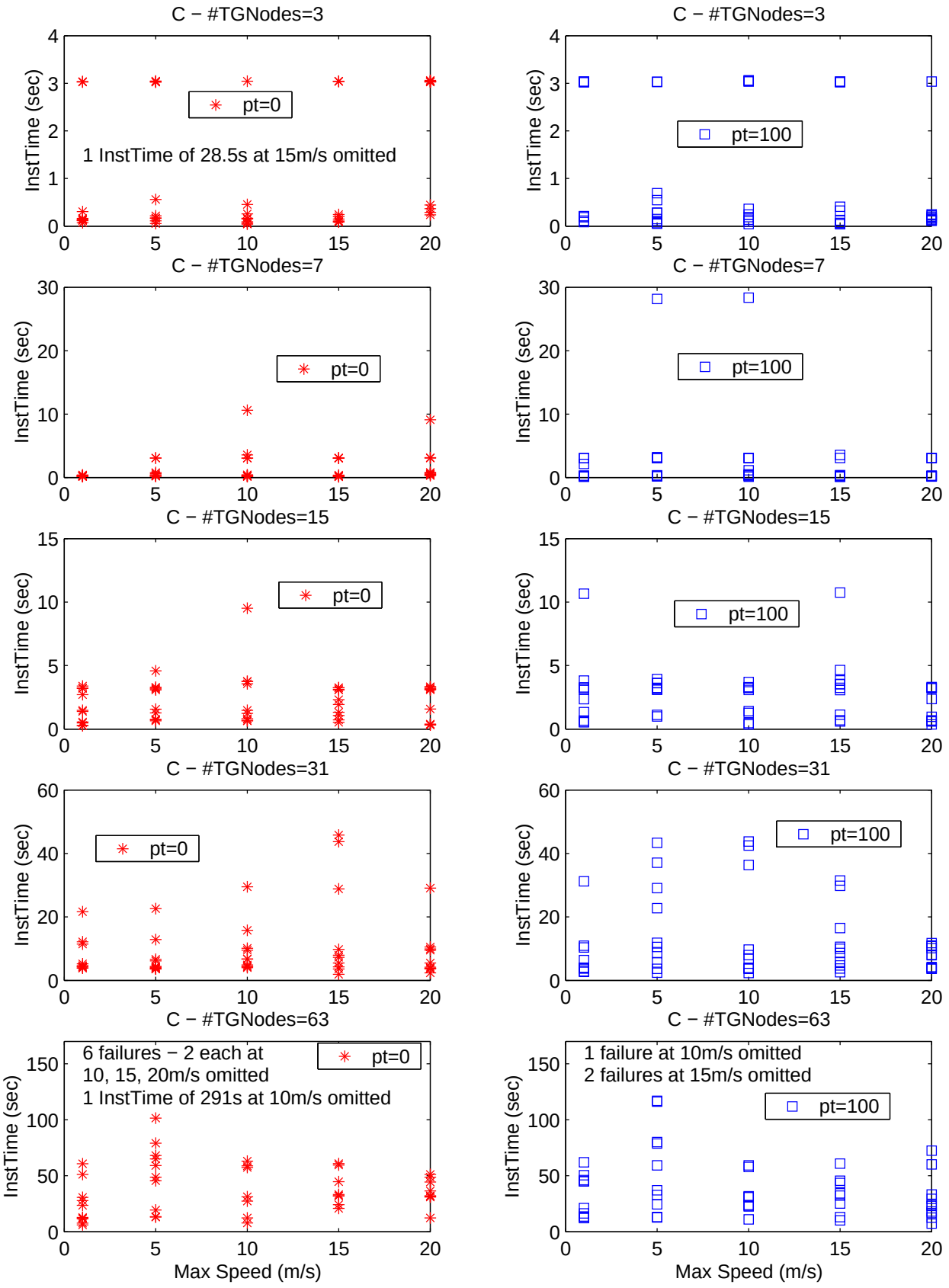

Figure 2: Instantiation Times Area = 2 km^2.

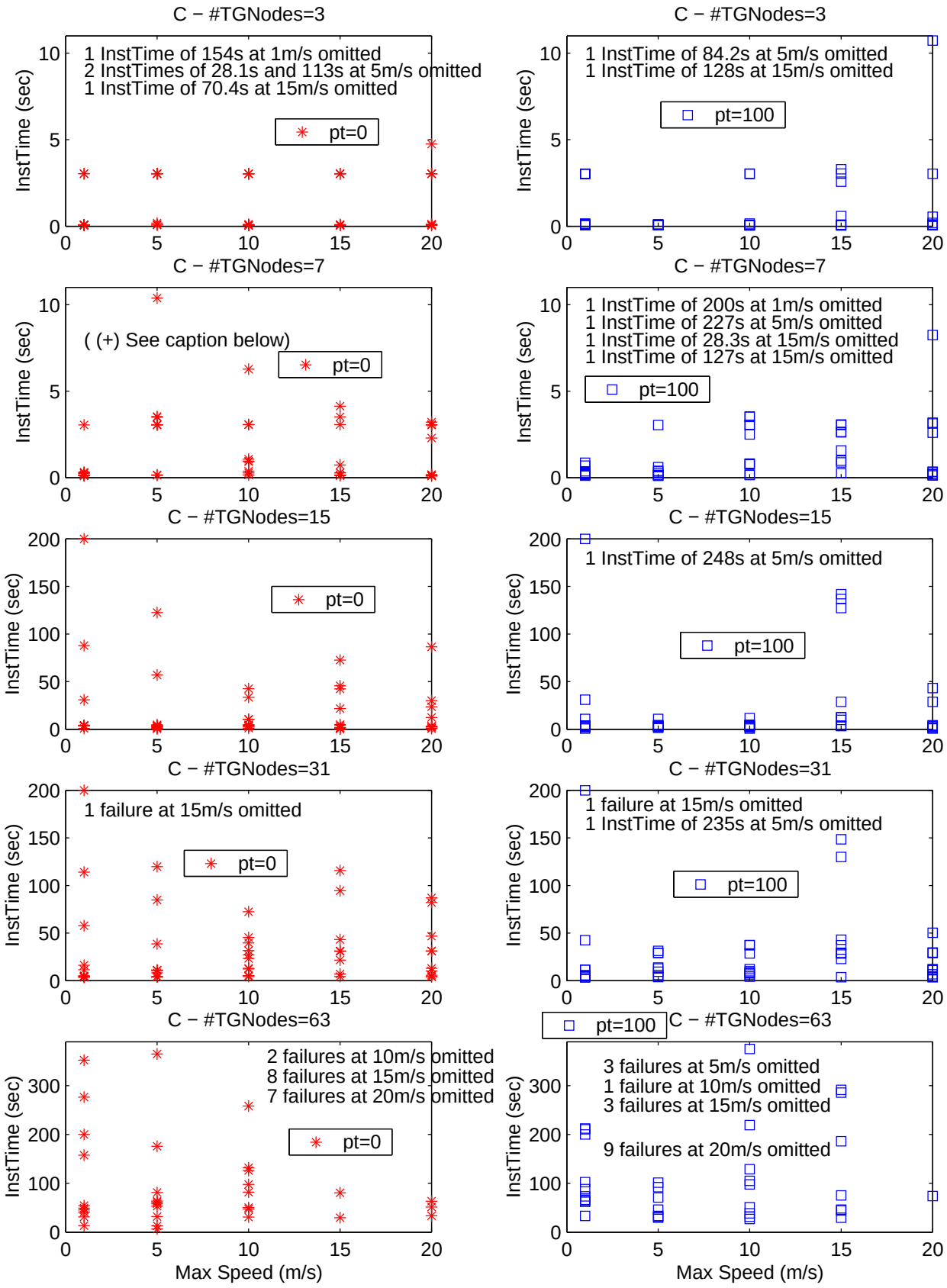

Figure 3: Instantiation Times Area = 4 km^2. (+) 1 Failure at 1m/s, 1 InstTime of 84.5s at 1m/s, 2 InstTimes of 29.5s and 113s at 5m/s, 1 InstTime of 71.3s at 10m/s, 1 InstTime of 72.4s at 15m/s and 1 InstTime of 112s at 20m/s omitted

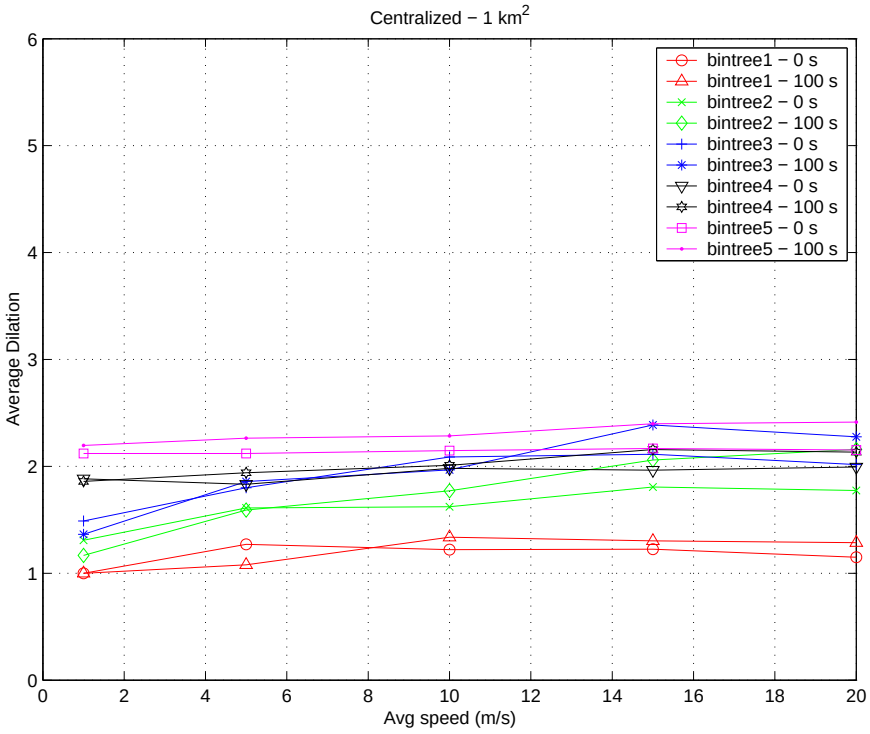

Figure 4: Average Dilation 1 km^2

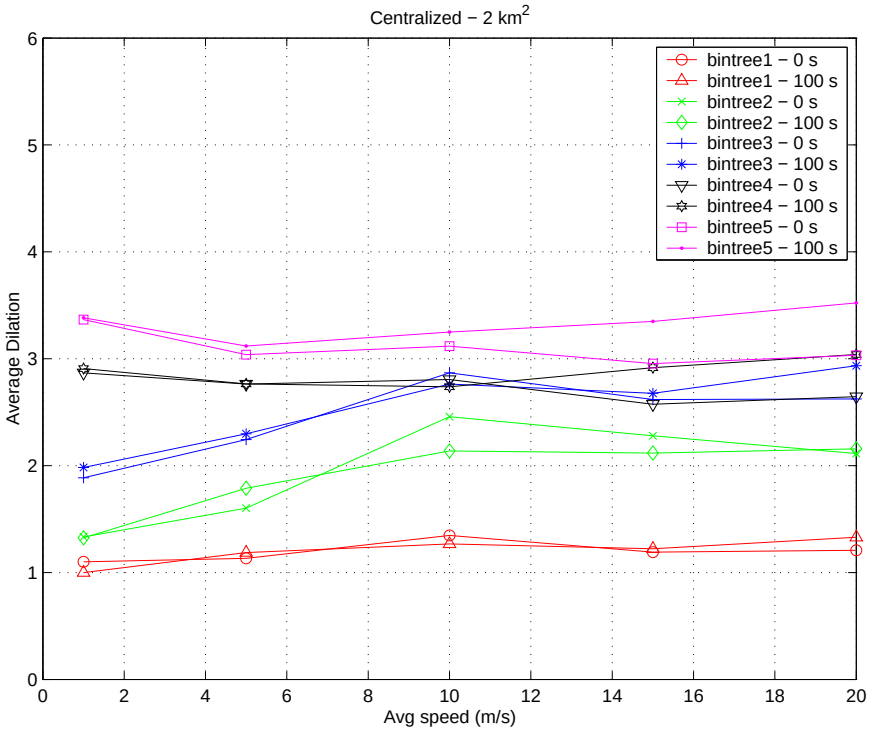

Figure 5: Average Dilation 2 km^2

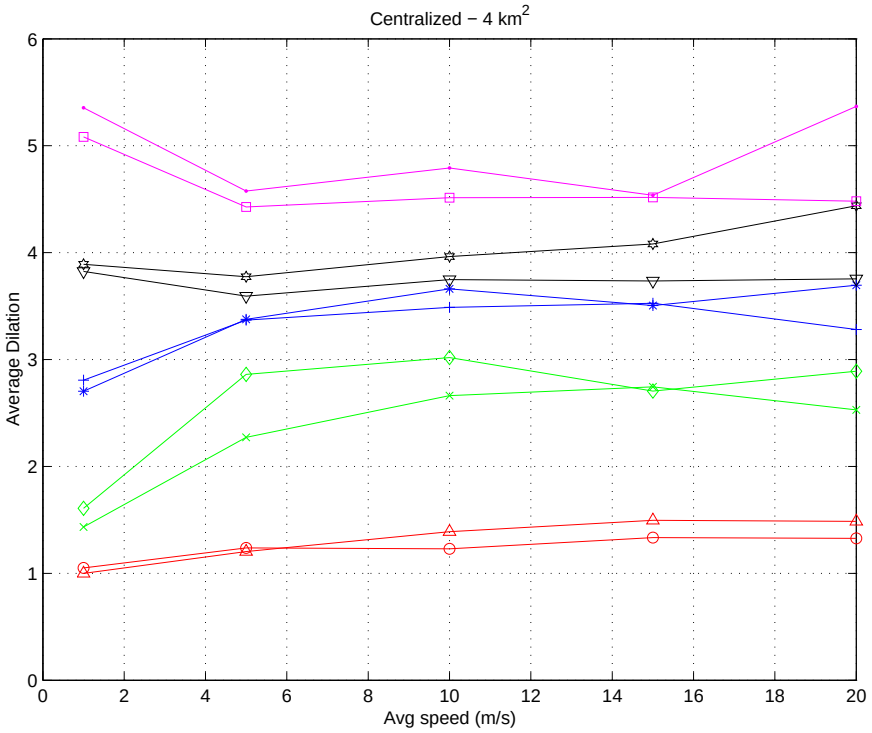

Figure 6: Average Dilation 4 km^2

nodes). If we consider that the transmission range is 250 m, then in a 1x1 km^2 the node density is 10^{-4} nodes/m^2, and the average number of neighbors is $10^{-4}\ \pi\ 250^2 \approx 19$, which is greater than 7. Dilation has to do with the embedding of the task graph onto a network, but once the nodes have been selected, unless a time-out takes place, they remain part of the instantiated task graph. This means that even if initially dilation is one (at instantiation time), with time and mobility, nodes wander away from their original positions but remain connected as components of the task graph, which result in dilation values higher than one.

Dilation values show a decrease for large task graphs when mobility increases. This happens because dilation is only computed for instantiated task graphs. And task graphs that are successfully instantiated with large number nodes have lower dilation values (if they had higher dilation values, they would not have been instantiated due to performance degradation of the network + transport layers capacity bottleneck). But at high mobility values, the lower dilation values are still "stretched" to higher values.

One observation regarding instantiation times and the task time-out value is that insufficient time-out values increase instantiation times. This happens because at each time-out (that is, every 14 seconds in our case), if a node is not instantiated but is still "alive" (the controller received one packet from that node in the past 14 seconds), a new `INSTANTIATION` packet will be sent, with a

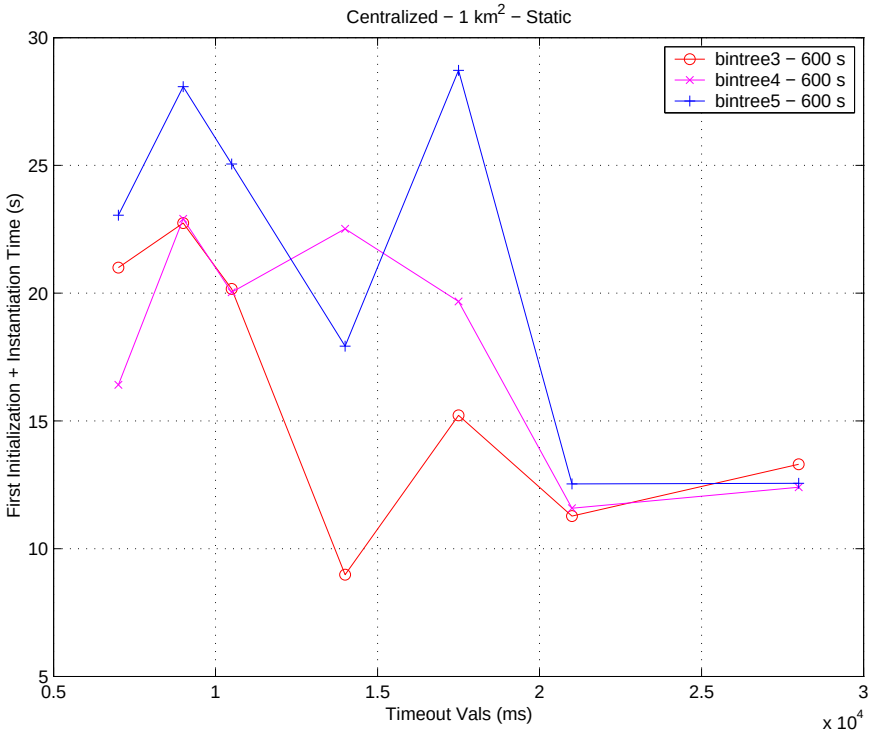

Figure 7: Average Instantiation time for different time-out values

new sequence number (this is to avoid receiving stale acknowledgement packets that were "lost" in the network when the actual node has already moved away). If an old but valid acknowledgement packet was in transit when this new `INSTANTIATION` packet was issued, then even if the ACK packet does arrive at the controller, it will be regarded as stale, and instantiation will not complete.

Fig. 7 shows that with the increase in the time-out value in a static scenario, the average instantiation time goes down, and reinstantiation time goes down as well, and the number of reinstantiations occurs much less frequently (Fig. 8)[2].

It is not clearly evident (more simulations are needed) that with the increase on the time-out values dilation increases, as seen in Fig. 10 though a slight increase can be observed. This happens because with lower time-out values higher dilation possibilities cannot be instantiated due to network + transport layer capacity limitations. This shows that a higher time-out value will have a higher chance of succeeding, and taking on the average less time. The trade-off here is on mobility detection. A high time-out value implies in a slow mobility detection. One possible solution is therefore the setting of lower period local time-out values between nodes, which is matter of future research.

[2]Note that this is the static scenario. Any re-instantiation is due to the absence of an `ALIVE` packet which got lost.

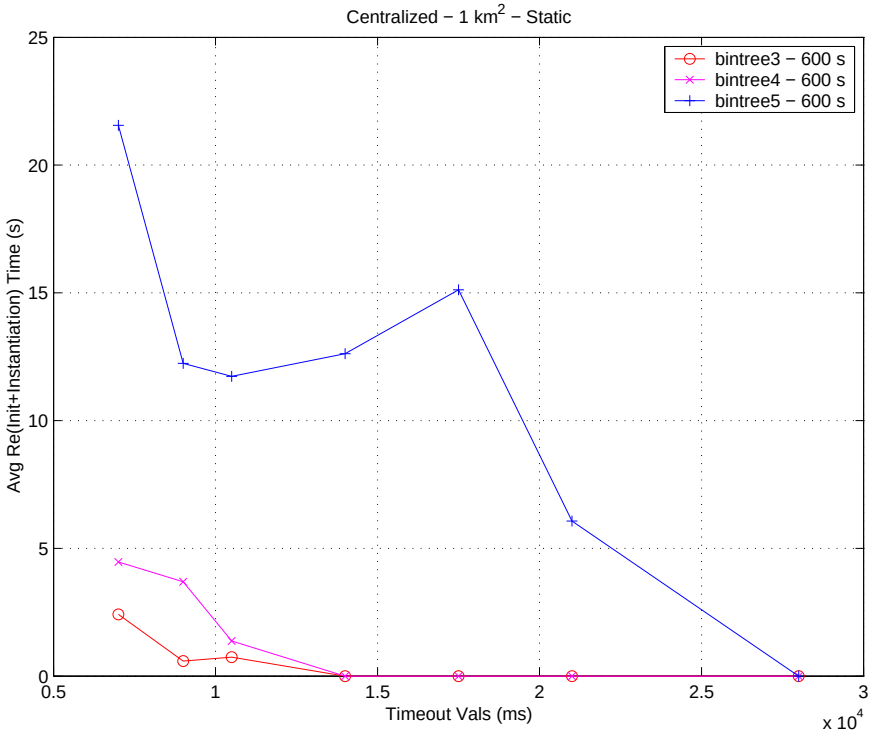

Figure 8: Average Re-Instantiation time for different time-out values

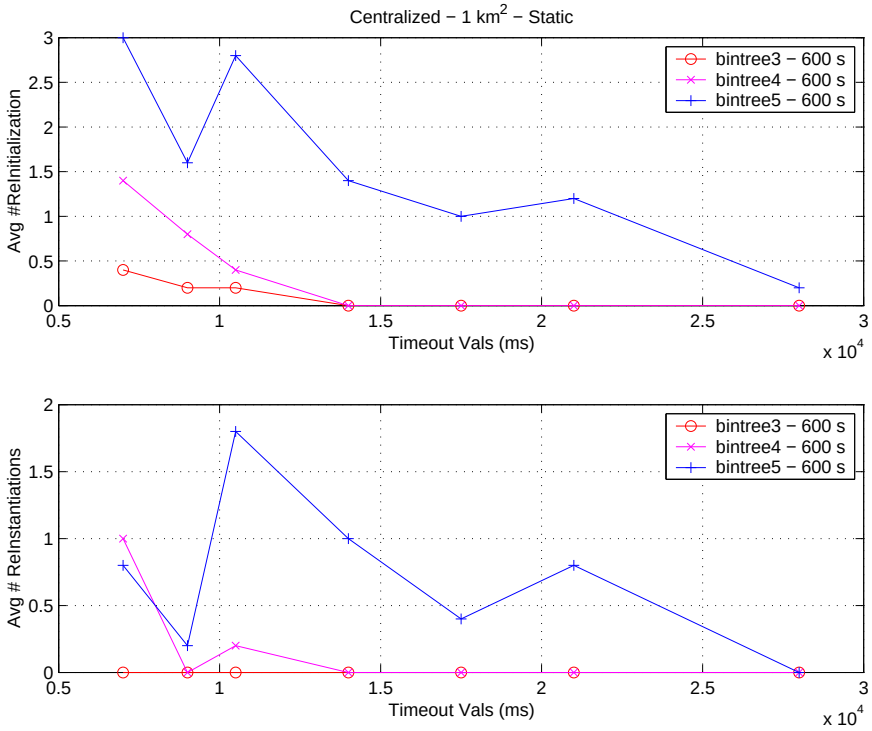

Figure 9: Average Number of Re-Instantiations for different time-out values

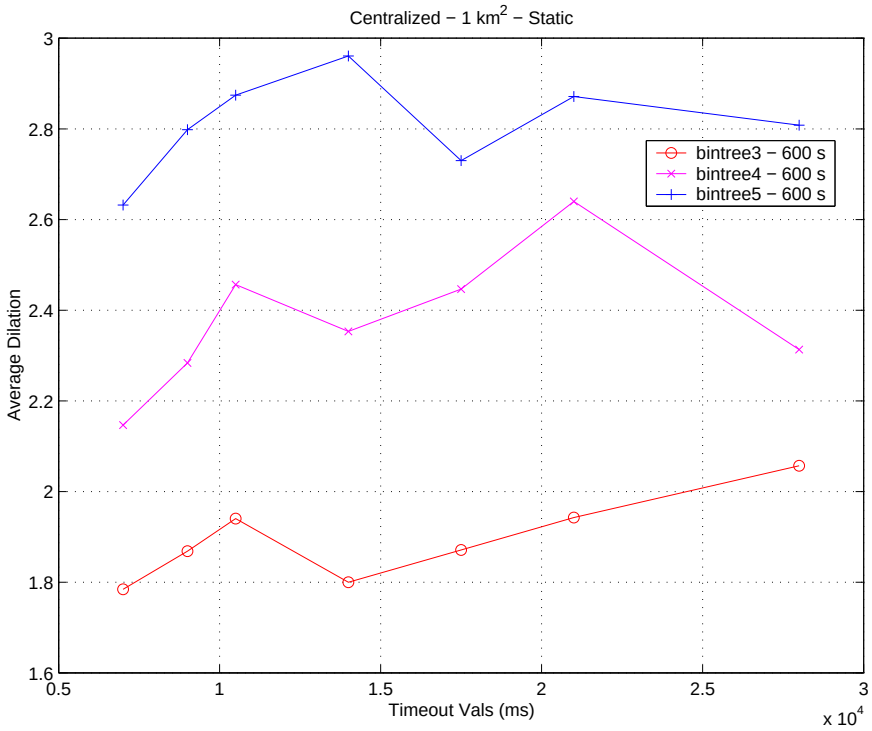

Figure 10: Average Dilation for different time-out values

Therefore, for instantiating centrally a task graph with varying number of nodes, an accurate time-out value estimation algorithm must be developed. A static time-out value essentially becomes the limiting factor on the size of the possible task graphs.

3 Re-Instantiation Times and Number of Re-Instantiations

To obtain an idea of the recovery process, together with the re-instantiation times we also plot the number of reinstantiations that took place in our simulation runs.

The decrease in the number of re-instantiations as the number of nodes in the TG increases in Fig. 11 is due to the fact that many reinstantiations did not finish once started. That is, once a reinstantiation happens, until it is completed, it is counted only one time. Since we simulated for 600 seconds, with the instantiation process starting at 200 second value, there could not have been many reinstantiations.

Again, if we exclude the 63 node case and the sparsely populated case (scenario of 100 nodes roaming in a 2x2 km^2 area), the reinstantiation times are below 20 seconds (Fig. 12 and 13) for most cases. We can notice that by decreasing node density, the average reinstantiation times

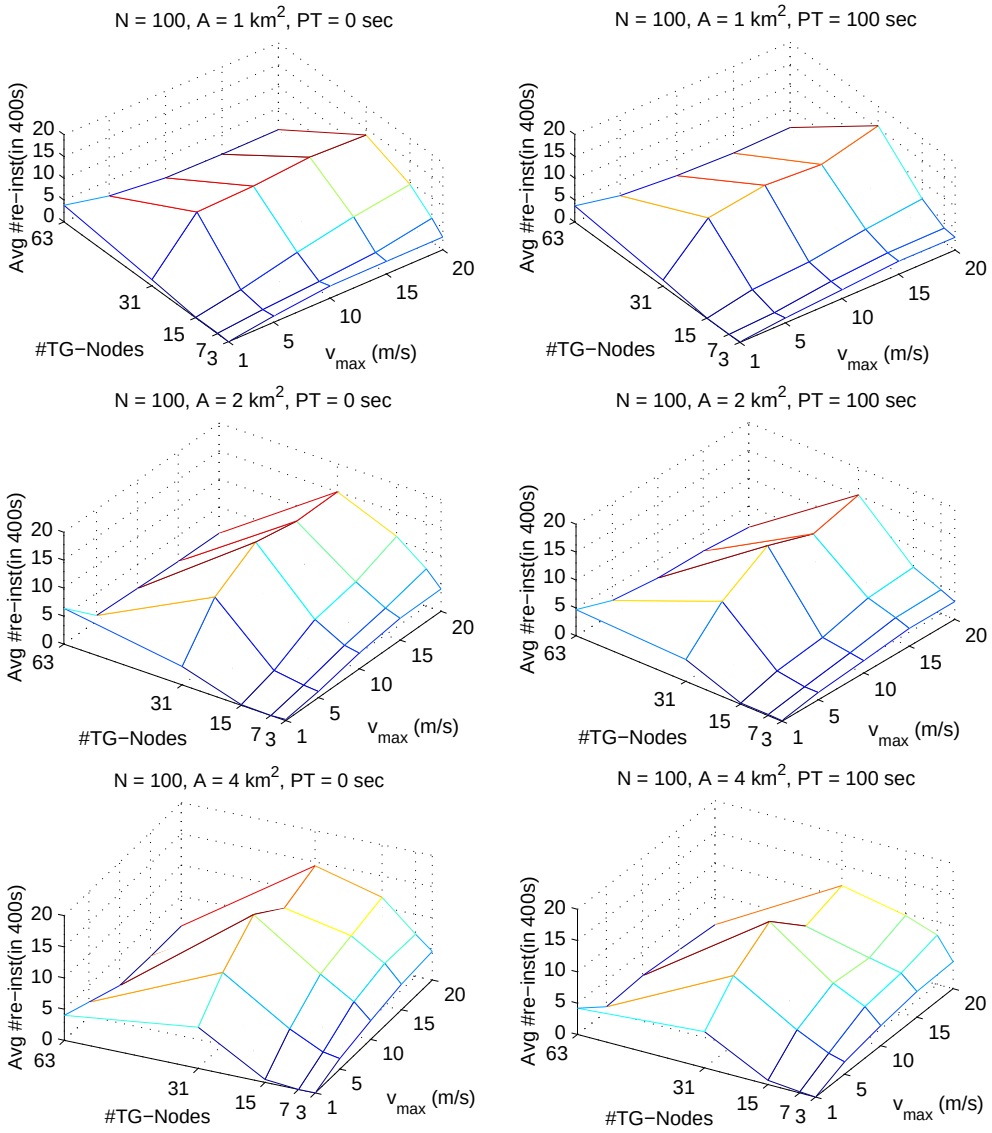

Figure 11: Number of Re-Instantiations Area $\in \{1, 2, 4\}$ km^2

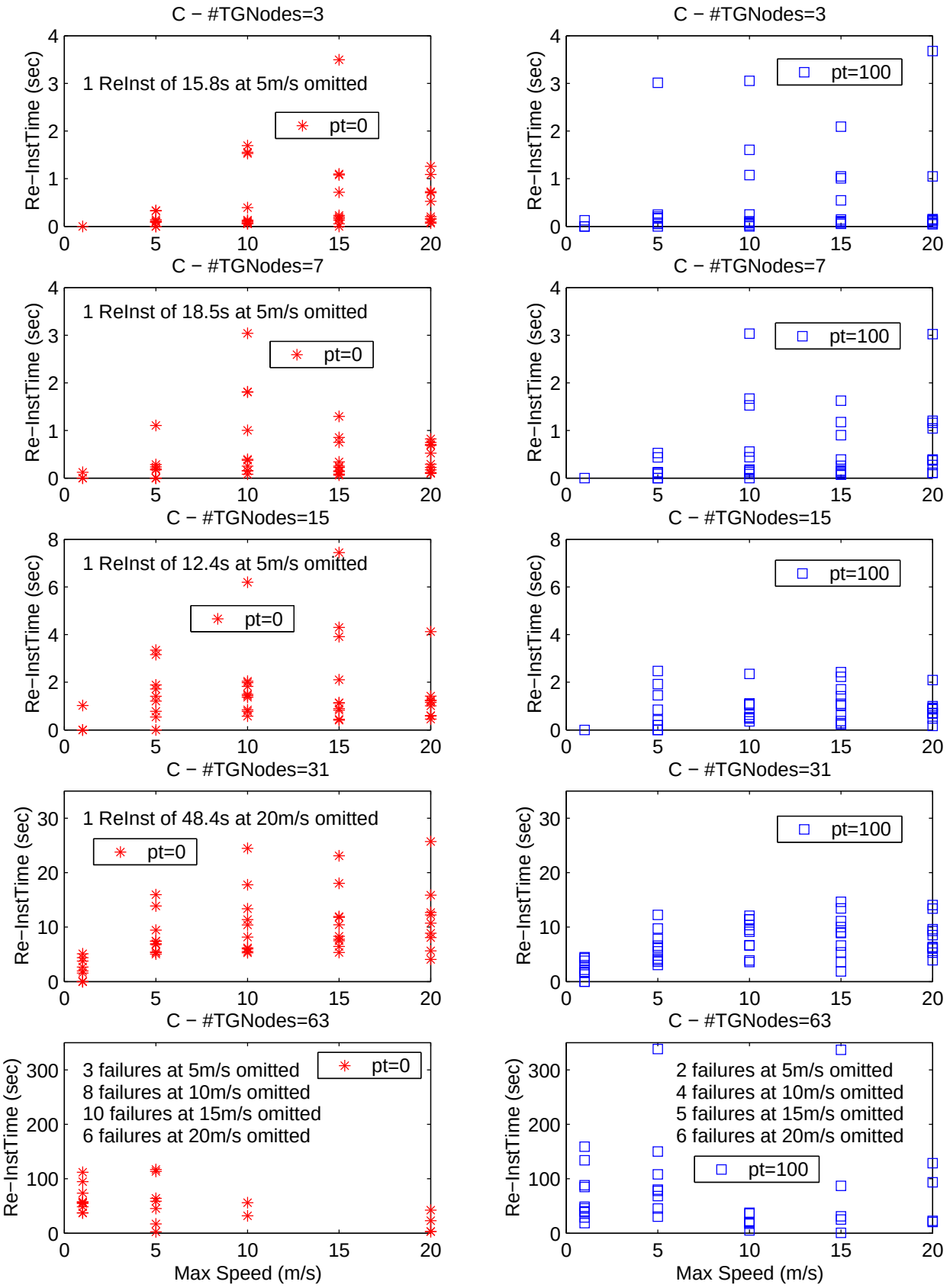

Figure 12: Re-Instantiation Times Area = 1 km^2

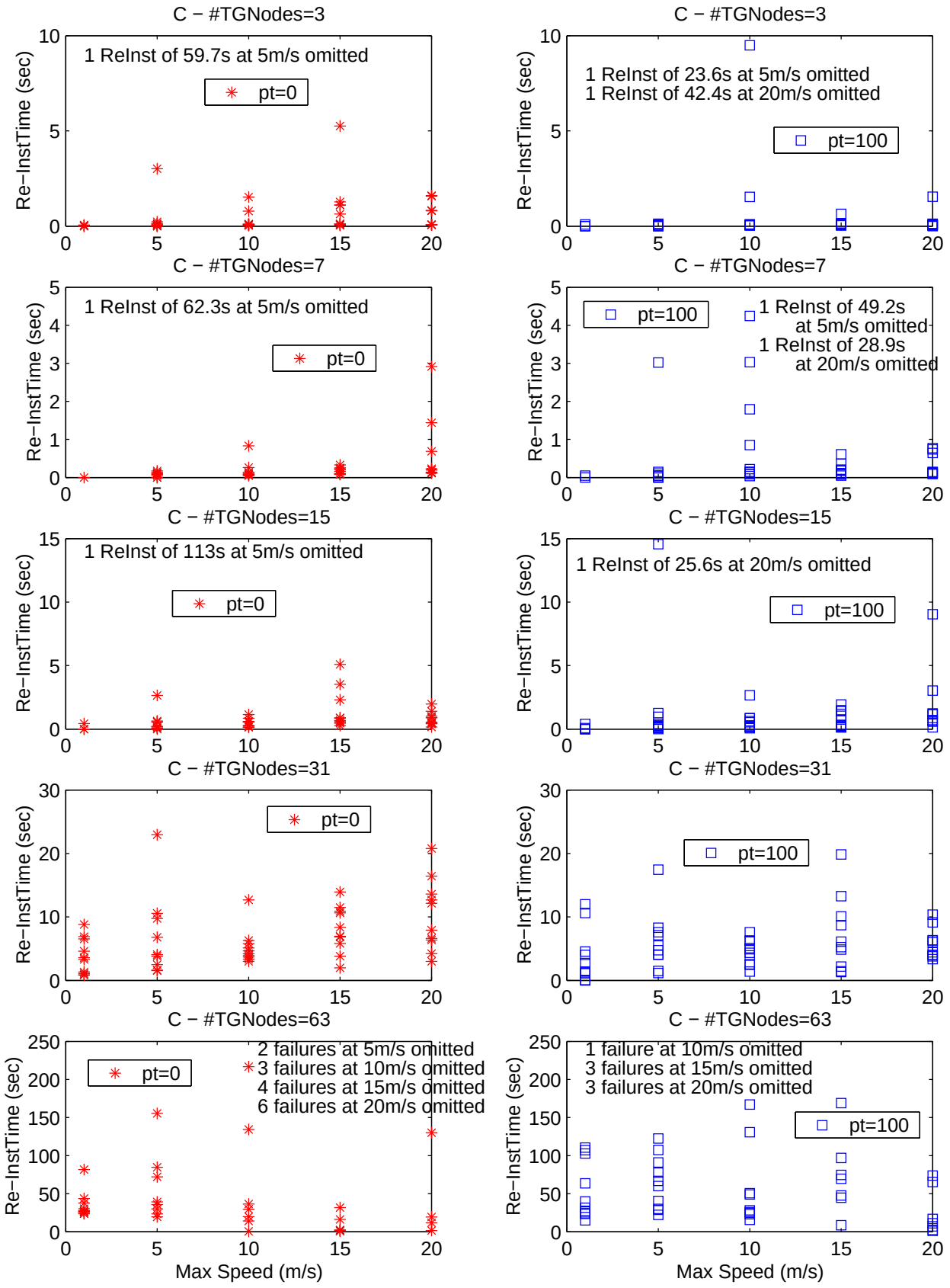

Figure 13: Re-Instantiation Times Area = 2 km^2

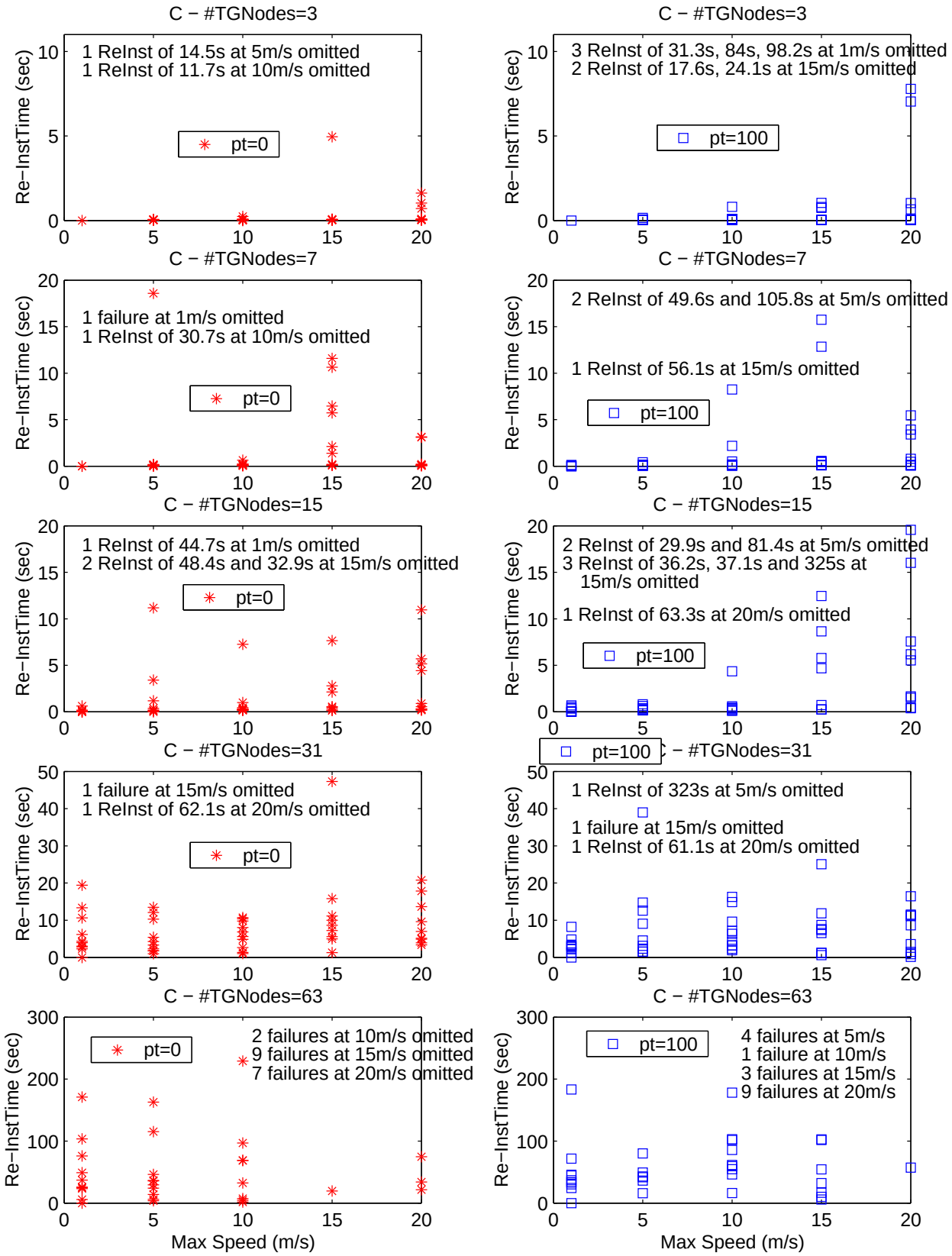

Figure 14: Re-Instantiation Times Area = 4 km^2

fluctuates more with increased mobility (Fig. 14). This shows that while highly populated areas are more prone to congestion problems, the performance of the communication channels between a source and destination pair is less dependent on the overall mobility.

This in fact suggests an "optimal" traffic pattern and/or transport control mechanism, once given a "density" condition. In highly populated areas, traffic should be controlled at source, and retransmissions should not be too frequent, while at less populated areas retransmissions should be scheduled in shorter periods, so as to detect if a new route exists (i.e., a node roamed in such a way that a path between source and destination has been established) to the desired destination.

4 Delay, Throughput, Hops traversed and Percentage Connected

In this section we plot the performance with respect to the metrics of delay and throughput of the application data units (ADU) packets. Delay is measured only of packets that successfully reach their destinations. Throughput is measured in terms of the percentage of the packets which reach their final destinations over the total number of packets that should have reached their sinks. We show also the average number of hops the ADU packet traversed in order to reach the destination. The percentage connected time is based on sampling the controller at Poisson Arrival times and checking whether the controller believes the task graph to be connected or not. It basically reflects the view the controller has of the task graph application. If it sees the application as mainly disconnected, continuous effort will be given to connect and/or reinstantiate. Due to the PASTA principle[3] [12], this metric can be used in QoS services at the controller, to decide whether a task graph should be (re)instantiated or aborted when the percentage connected time is below a threshold.

We include a 2 dimensional plot of the effective throughput, delay, hops traversed and a percentage connected time metric in Figs. 15-26.

The lower delay values (Figs. 15, 17 and 19), and corresponding lower number of hops traversed (Figs. 16, 18 and 20 for high speed and large task graph (63 nodes) is due to the fact that only successfully delivered packets are counted towards computing the two metrics above, and in a highly mobile network, the packets that are successfully delivered (note the very low corresponding throughput) are exactly those that suffer low delay and which crossed very few hops. Throughput in all cases drops with the increase of the number of nodes in the task graph and with mobility

[3]Poisson Arrivals See Time Averages - basically the percentage of times an observed variable is in a given state when observed according to a Poisson arrival process approaches the real percentage value assymptotically.

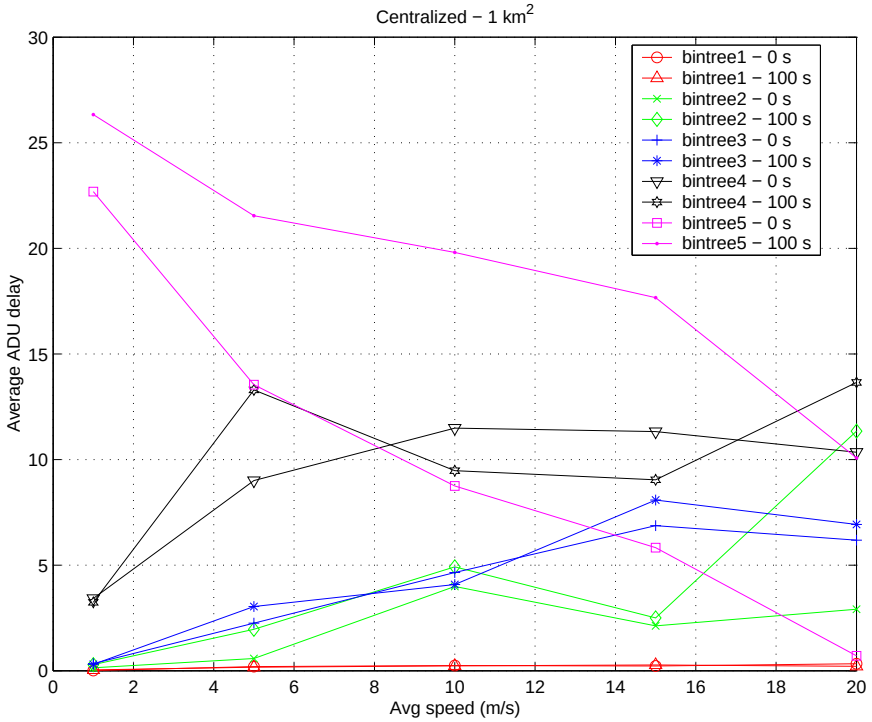

Figure 15: Centralized Protocol - Average Delay for Area = 1 km^2

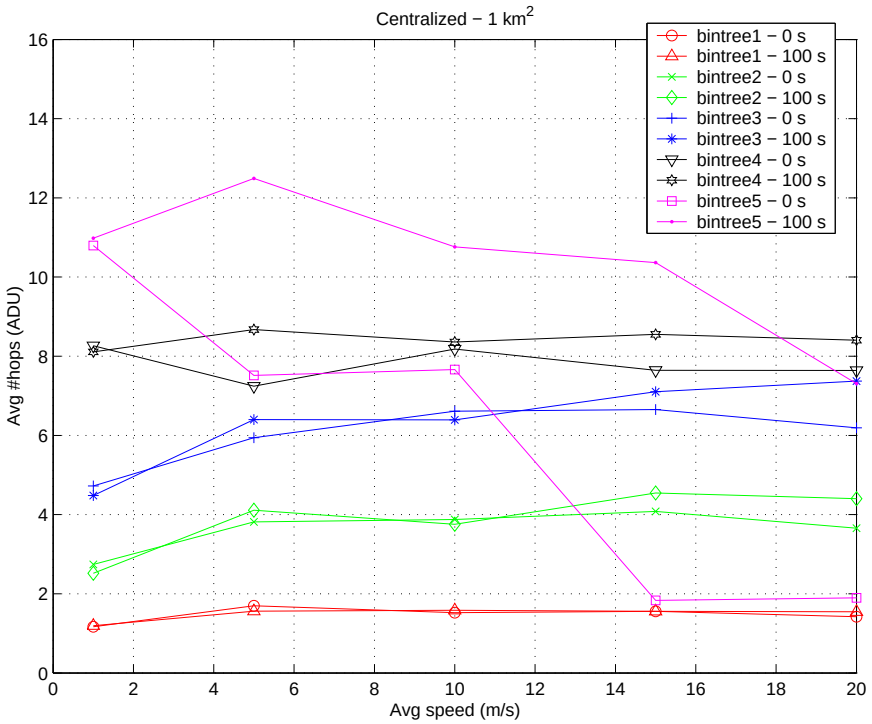

Figure 16: Centralized Protocol - Average Number of Hops traversed for Area = 1 km^2

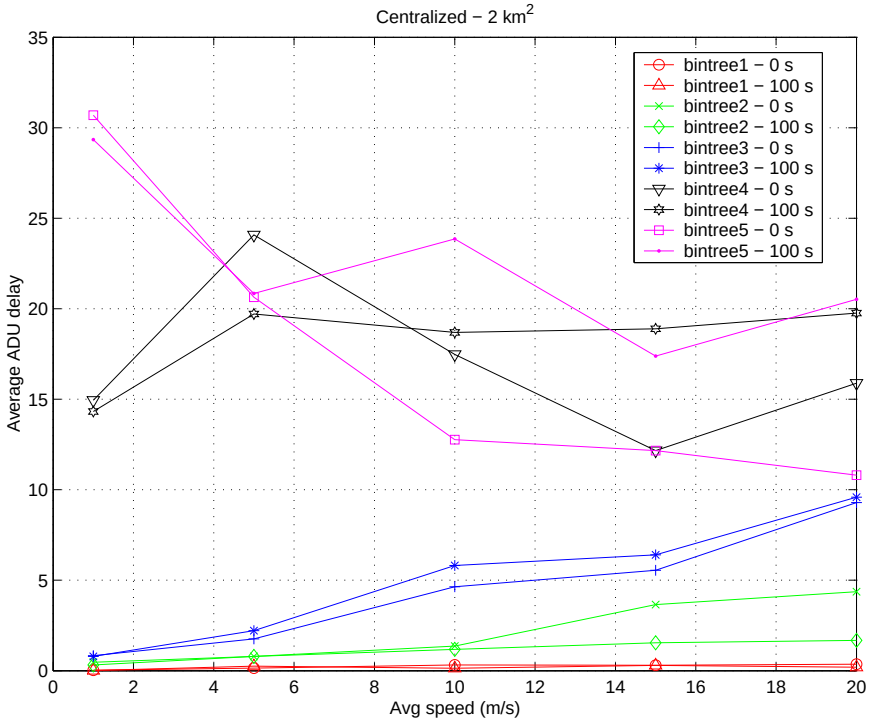

Figure 17: Centralized Protocol - Average Delay for Area = 2 km^2

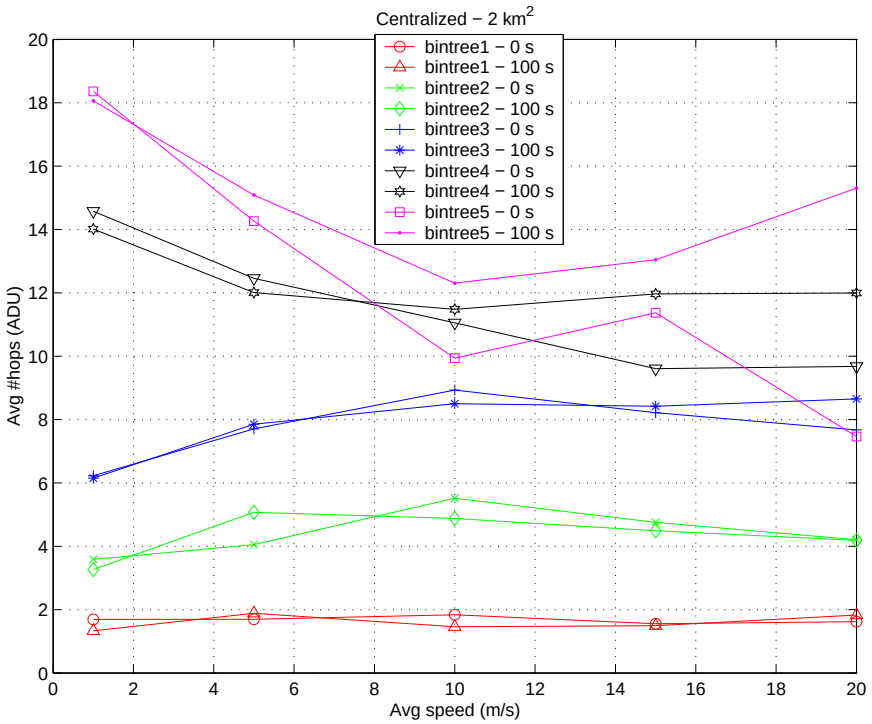

Figure 18: Centralized Protocol - Average Number of Hops traversed for Area = 2 km^2

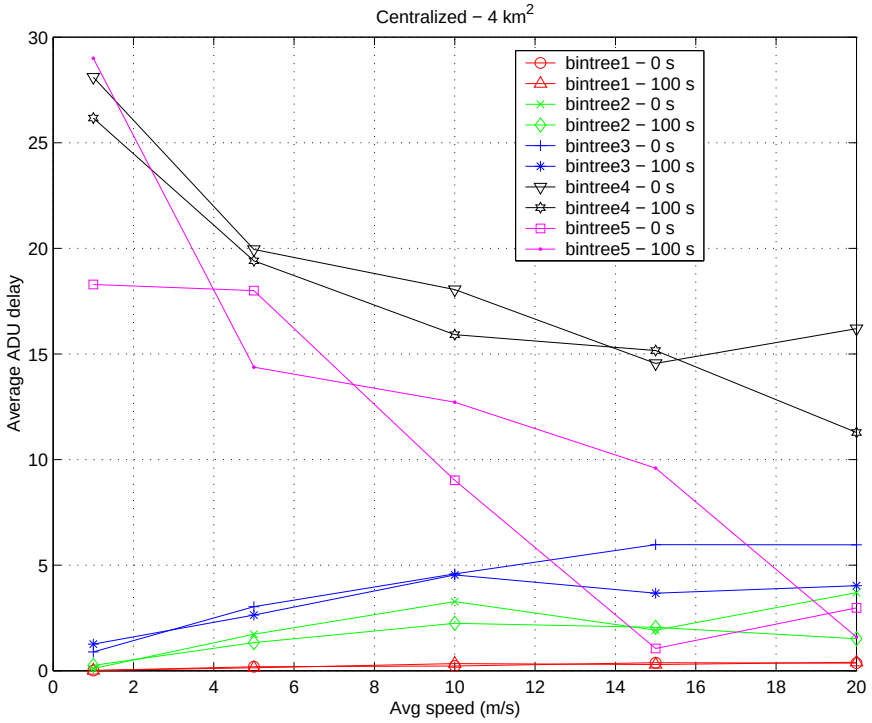

Figure 19: Centralized Protocol - Average Delay for Area = 4 km^2

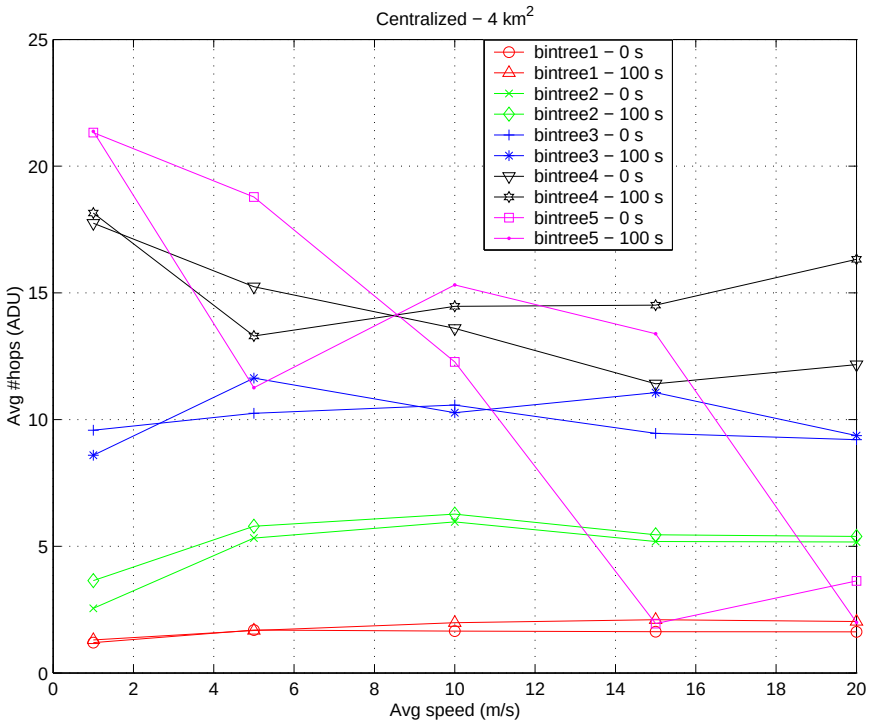

Figure 20: Centralized Protocol - Average Number of Hops traversed for Area = 4 km^2

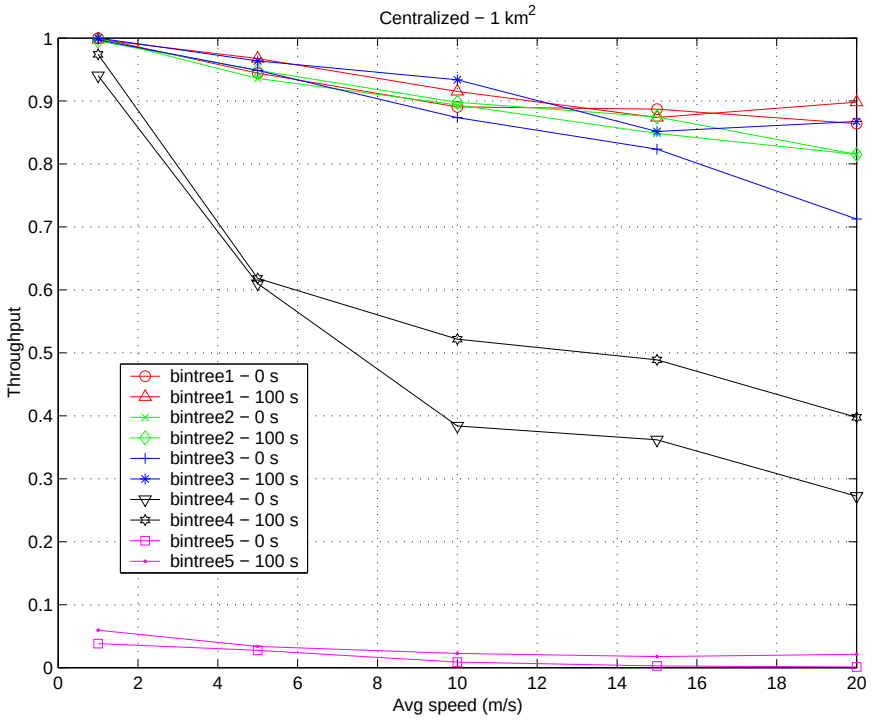

Figure 21: Centralized Protocol - Average Effective Throughput for Area = 1 km^2

(Figs. 21, 22 and 23). Throughput is almost zero for the 63 node case.

A very interesting question comes up when we see the average percentage of connected time (Figs. 24, 25 and 26) and relate them to the average throughput. One would believe that they should reflect similar values, since the average throughput depends on the connectivity of the instantiated task graph.

The key point here is that throughput is measured per packet, while connectivity is a state that depends on all the nodes. In other words, the application is deemed disconnected even if only one node is disconnected, but the disconnected node may not lie on the path of the data packet, and thus throughput can still be one. Conversely, a task graph can be fully "connected," and yet still not have throughput. This can happen when all nodes successfully exchange `ALIVE` packets with the controller but there is a local time-out period between two neighboring nodes. In this case a data packet in transit will get dropped, but the controller will not observe the disconnectivity.

We can see from these observations that a key metric to throughput in a task graph is not connectivity only, but connectivity between source and destination. In other words, the requirements on the connectivity of the whole task graph may be too stringent a requirement for specific applications that do not utilize paths between all member nodes. In other words, the study of the

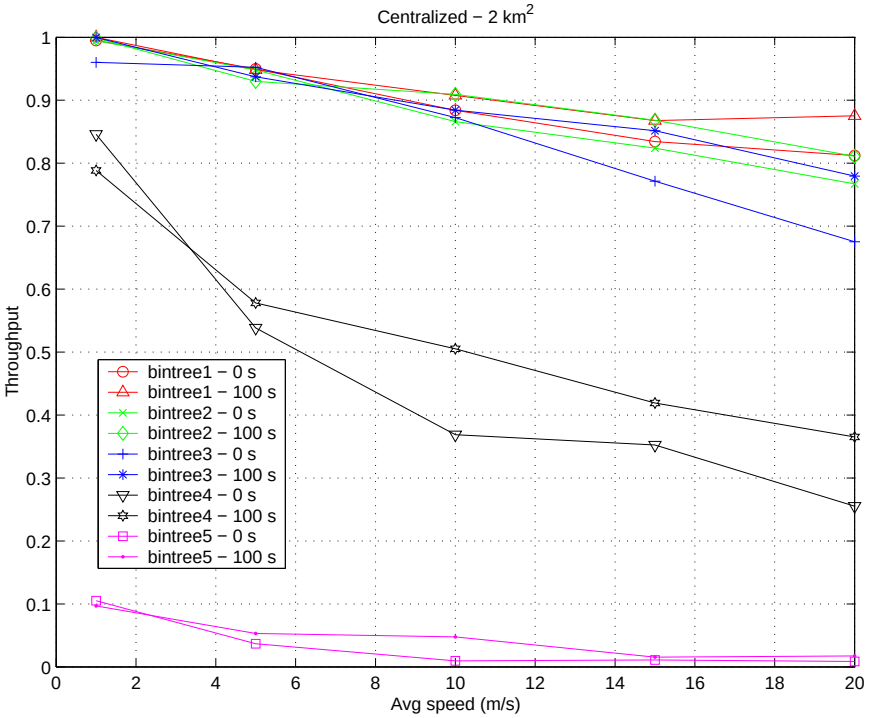

Figure 22: Centralized Protocol - Average Effective Throughput for Area = 2 km^2

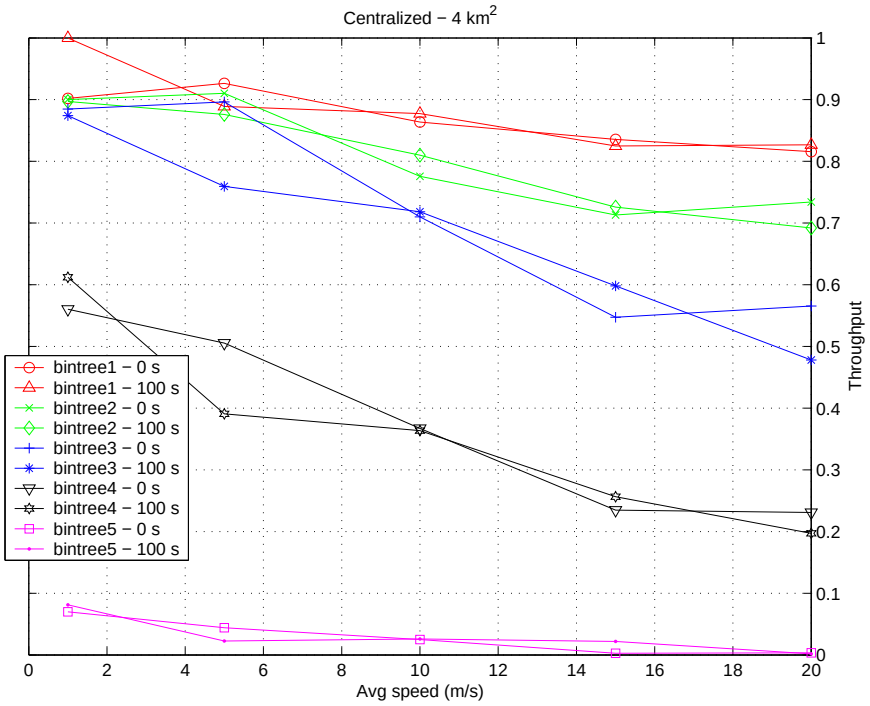

Figure 23: Centralized Protocol - Average Effective Throughput for Area = 4 km^2

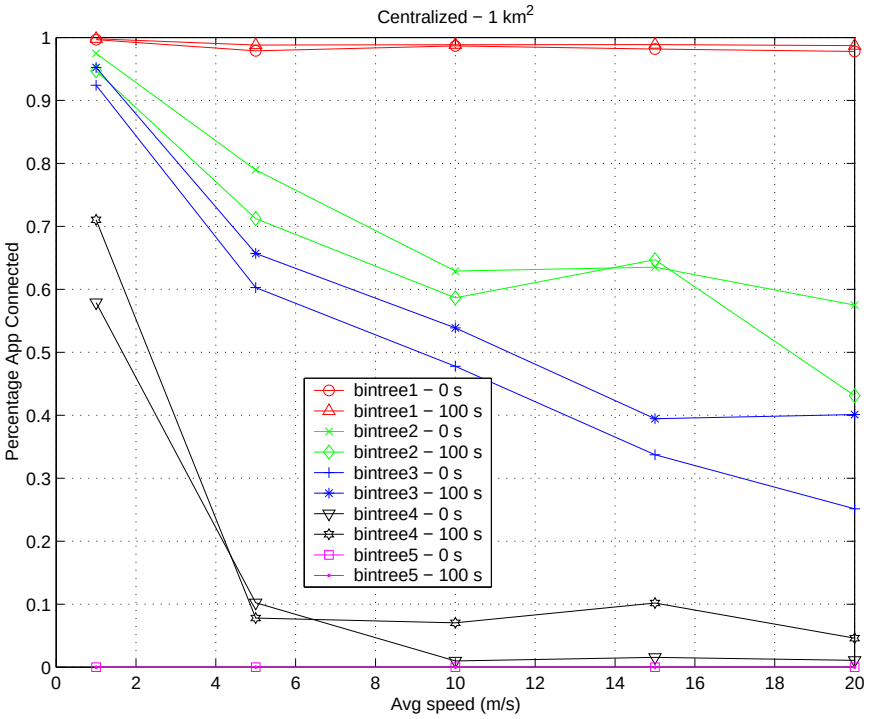

Figure 24: Centralized Protocol - Average Percentage Connected Time for Area = 1 km^2

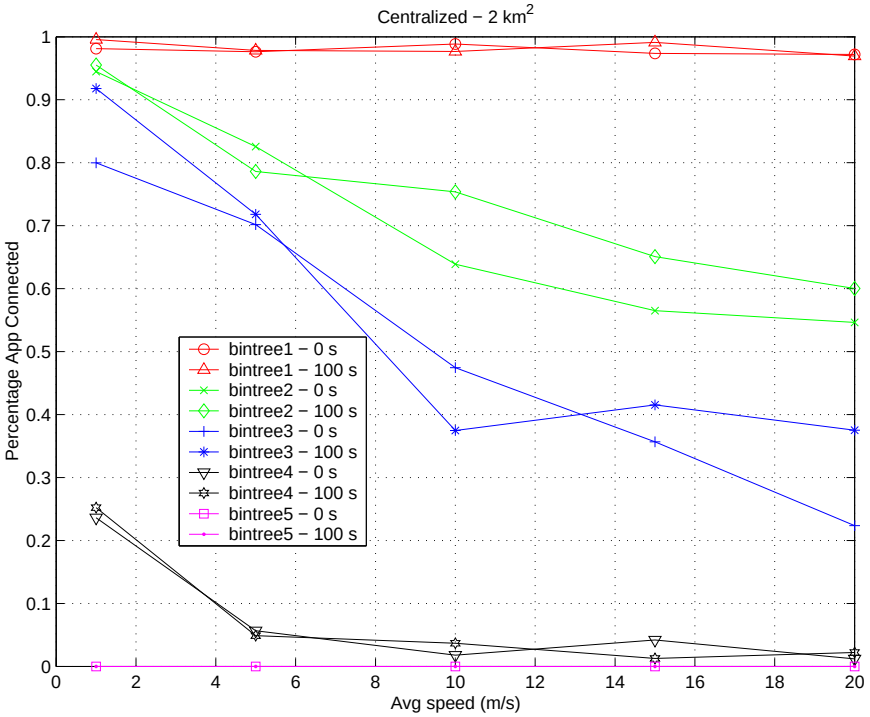

Figure 25: Centralized Protocol - Average Percentage Connected Time for Area = 2 km^2

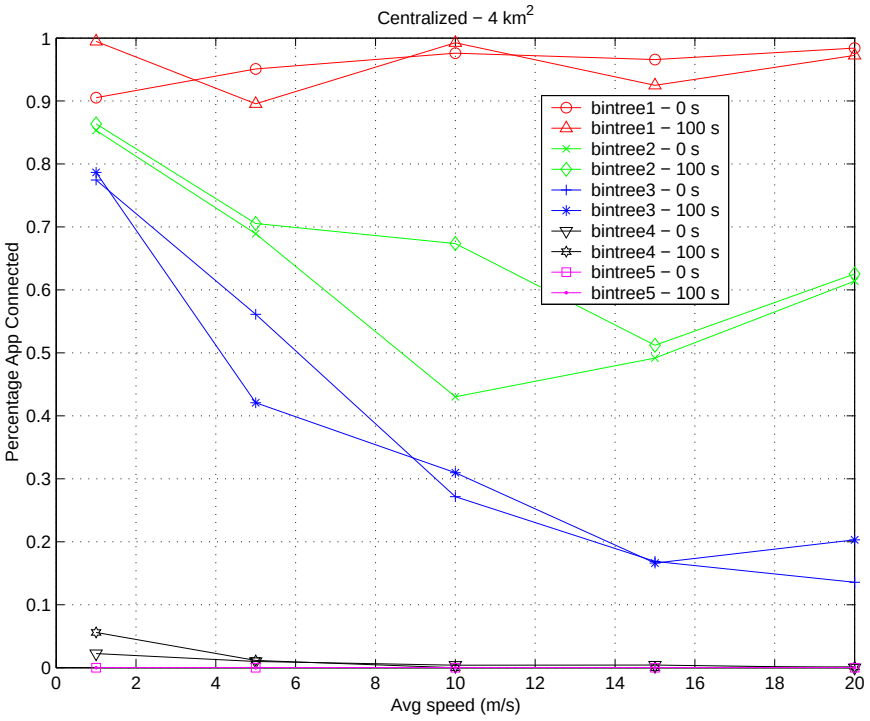

Figure 26: Centralized Protocol - Average Percentage Connected Time for Area = 4 km^2

connectivity and suitability of running (instantiating) a task graph on a MANET depending on the connectivity should take into consideration the traffic pattern as one key factor.

5 Overhead

In this section we show the overhead of our protocol. We plot the graphs with respect to the number of original broadcast packets (Fig. 27), forwarded broadcast (Fig. 28) and unicast packets (Fig. 29) needed for running our centralized protocol as defined in [9].

From our graphs, we can see that large task graphs are impacted more by mobility as far as original broadcast packets are concerned. For lower density areas (roaming areas of 2 and 4 km^2) there is a direct increase in the number of broadcast needed and the number of nodes in the task graph.

Original broadcast packets are transmitted from the controller only. The controller performs an increased ring search when not enough nodes answer to their initialization packets. When the node density is low, increased broadcasts are necessary. Also during reinitialization stages an original broadcast is performed. Essentially, the higher the number of reinitialization + reinstantiations,

the higher the number of original broadcasts (and consequent forwarded broadcasts) and unicast packets. Many of the packets may well be considered hopeless attempts in an impossible scenario (especially considering the 63 node case), a specific admission control mechanism is necessary to avoid such redundant traffic.

6 Conclusion

We have found that time out values need to be adjusted to accomodate increasingly large number of nodes if a centralized control is desired. To deal with mobility, localized mobility detection mechanisms may be put in place, but this is a matter for future research. Also, to examine only the "connectivity" of a set of nodes misses out on the fact that not all paths need be present and connected at all times. Linking the traffic pattern to throughput and/or connectivity studies would yield much more adequate results. The increase of overhead packets when attempting to instantiate large task graphs and highly mobile scenarios suggest that an admission control mechanisms, based on the variables of the environment observed, coupled with variables such as time out values, traffic characteristics, would save effort in attempting to instantiate a task graph when the probability of success is absurdly small.

References

[1] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. "The Design and Implementation of an Intentional Naming System". In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP)*, Kiawah Island, SC, December 1999.

[2] G. Banavar, J. Beck, E. Gluzberg, J. Munson, J. Sussman, and D. Zukowski. "Challenges: An Application Model for Pervasive Computing". In *Proceedings of the 6th ACM MobiCom Conference*, Boston, MA, August 2000.

[3] M. Esler, J. Hightower, T. Anderson, and G. Borriello. "Next Century Challenges: Data-Centric Networking for Invisible Computing The Portolano Project at the University of Washington". In *Proceedings of the 5th ACM MobiCom Conference*, Seattle, WA, August 1999.

[4] E. Guttman. "Service Location Protocol: Automatic Discovery of IP Network Services". *IEEE Internet Computing*, July 1999.

[5] Sun Microsystems: Jini Technology Core Platform Specification. URL. `http://www.sun.com/jini/specs`.

[6] Oxygen Project, MIT Laboratory for Computer Science. URL. `http://www.oxygen.lcs.mit.edu`.

[7] W. Ke, P. Basu, and T. D. C. Little. "A Task Graph Based Application Framework for Mobile Ad Hoc Networks". In *Proceedings of the IEEE International Conference on Communications (ICC)*, New York, NY, April-May 2002.

[8] P. Basu, W. Ke, and T. D. C. Little. "A Novel Approach for Execution of Distributed Tasks on Mobile Ad Hoc Networks". In *Proceedings of the IEEE Wireless Computing and Networking Conference (WCNC)*, Orlando, FL, March 2002.

[9] P. Basu. *A Task Based Approach for Modeling Distributed Applications on Mobile Ad Hoc Networks*. PhD thesis, Boston University, May 2003.

[10] D. B. Johnson and D. A. Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks". In T. Imielinski and H. Korth, editors, *Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[11] G. Holland and N. Vaidya. "Analysis of TCP Performance over Mobile Ad Hoc Networks". In *Proceedings of the 5th ACM MobiCom Conference*, pages 219–230, Seattle, WA, August 1999.

[12] R. W. Wolff. Poisson Arrivals See Time Averages. *Oper. Res.*, 30:223–231, 1982.

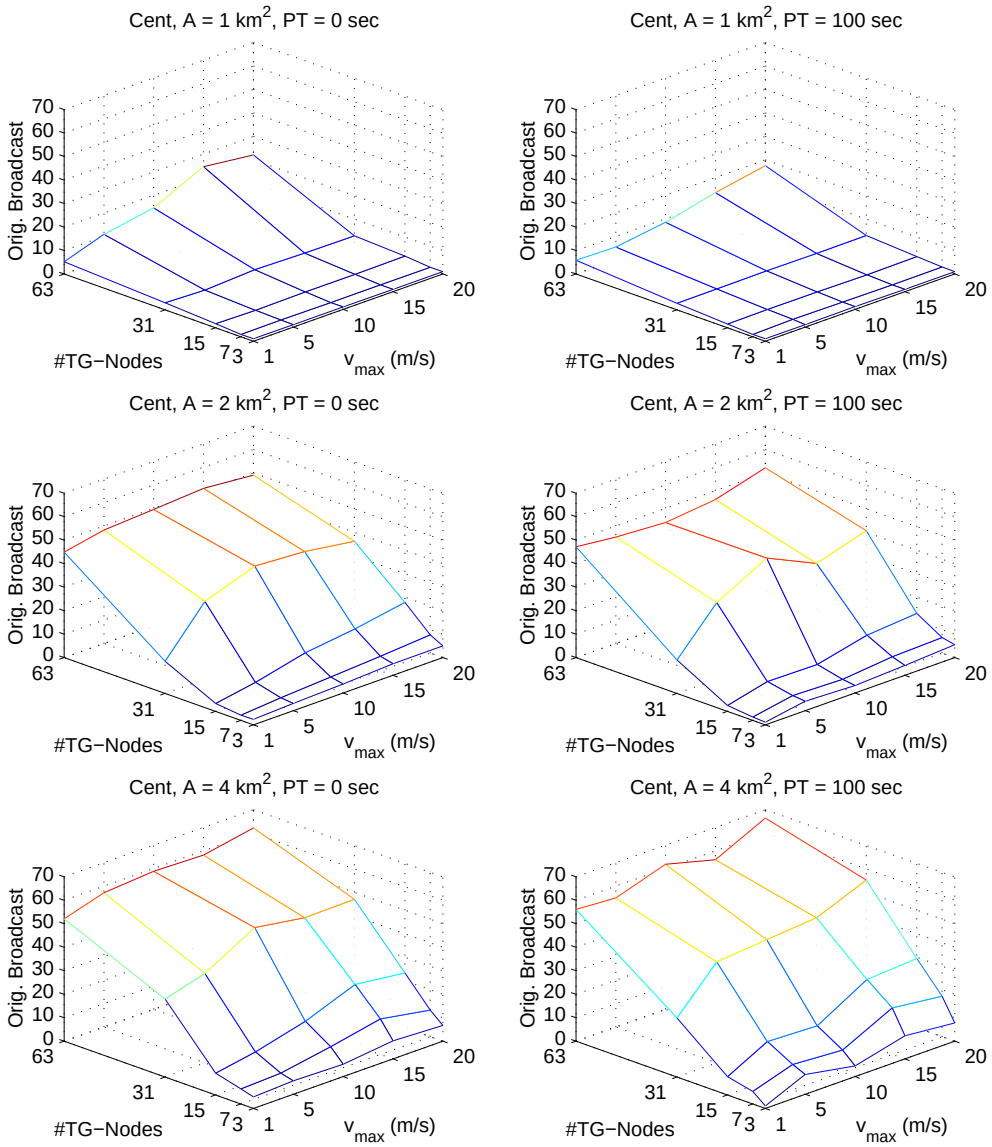

Figure 27: Centralized Protocol - Original Broadcast Packets for Area $\in \{1, 2, 4\}$ km^2 and Pause Times $\in \{0, 100\}$.

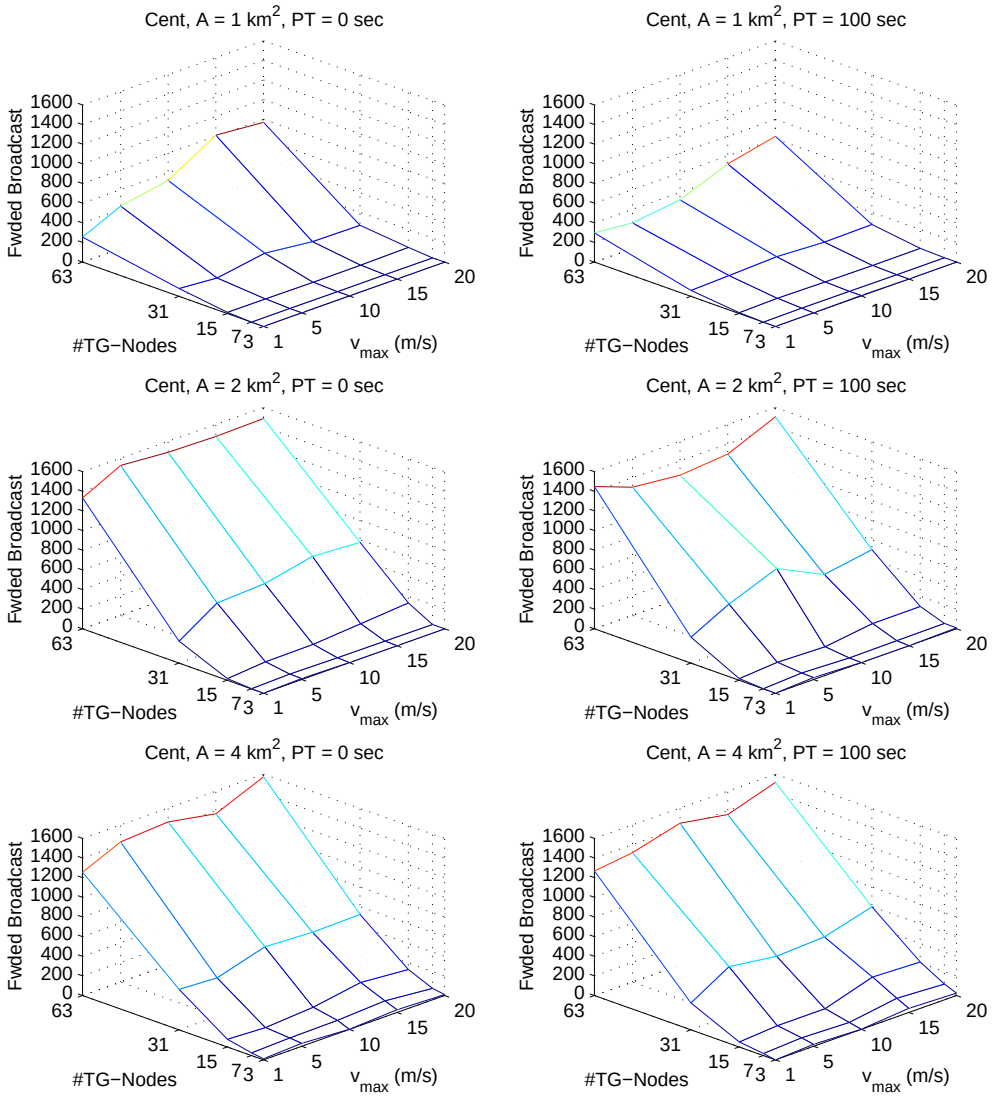

Figure 28: Centralized Protocol - Forwarded Broadcast Packets for Area $\in \{1, 2, 4\}$ km^2 and Pause Times $\in \{0, 100\}$.

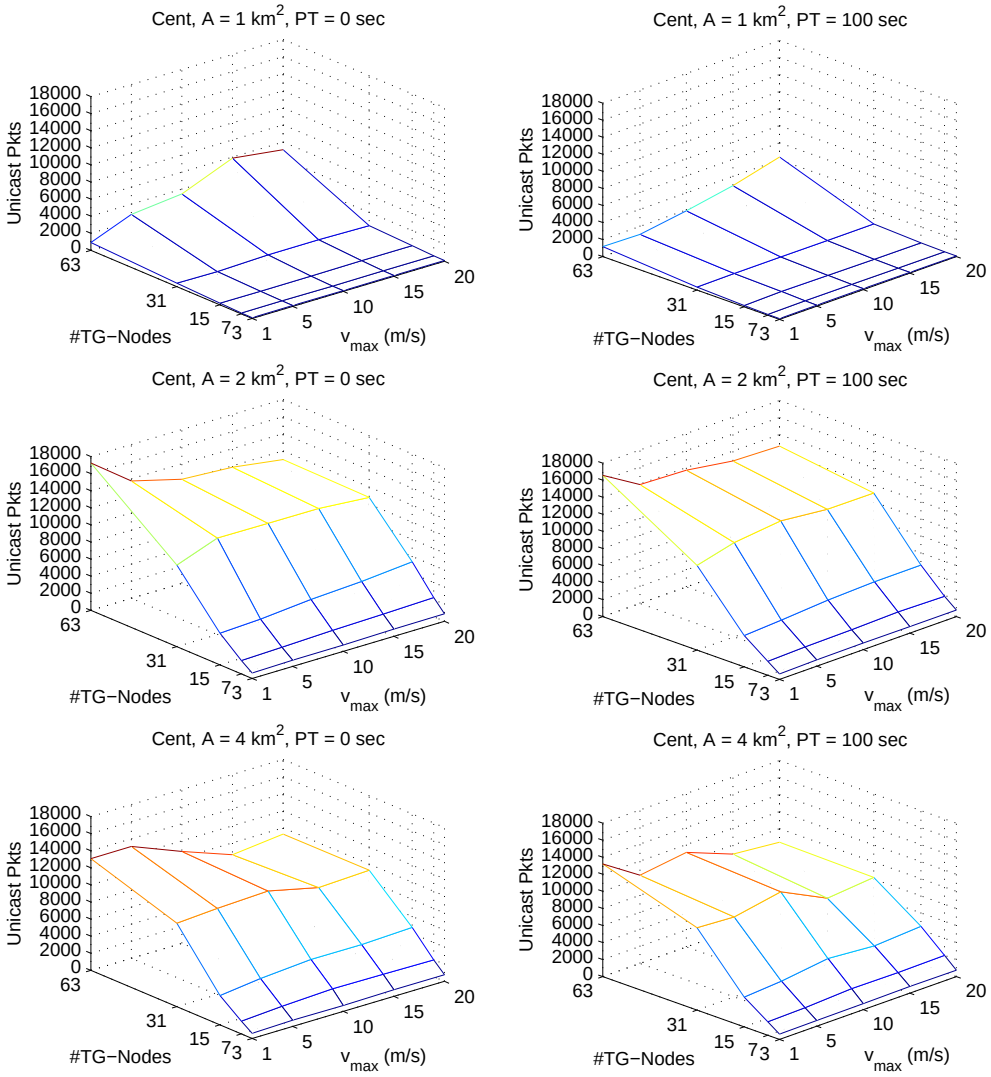

Figure 29: Centralized Protocol - Unicast Packets for Area $\in \{1, 2, 4\}$ km^2 and Pause Times $\in \{0, 100\}$.