A view from inside a distributed Internet coordinate system

Mohamad JABER, Cao-Cuong NGO and Chadi BARAKAT
EPI Planète, INRIA, France
Email:{mjaber, cngo, cbarakat}@sophia.inria.fr

Abstract—**The Vivaldi system [1] is known to be one of the most interesting approaches for the calculation of Internet coordinates. It is a fully distributed, light-weight and adaptive algorithm. Recent studies show that host coordinates in the Vivaldi system are not stable and are drifting rapidly even when the network delays do not change. In this paper we observe that, despite the instability of Vivaldi coordinates in their absolute values, there is still a stable internal structure that can better reflect the stability of the underlying network. We proceed for this study by extensive simulations and experimentations. In a first stage, we confirm the fact that Vivaldi coordinates oscillate over time because of the adaptive nature of the system. However the variations of these coordinates are most of the time correlated with each other pointing to a stable cluster of nodes seen from inside the network. In a second stage, we present a new clustering algorithm based on the data mining Hierarchical Grouping Method [2] to identify this cluster of stable nodes once the network and the host coordinates reach their stationary regime. The metric that we use to cluster nodes in the system is the amount of variation of their Euclidean distances. Our main finding is that such stable cluster of nodes always exists and is grouping most of the nodes. We highlight the utility of such finding with an application that tracks changes in network delays. To this end, we propose to track a simple signal, which is the size of this biggest stable cluster.**

I. Introduction

During the last years, the Internet has evolved from a small academic network to a huge network interconnecting tens of thousands of autonomous systems (AS). This rapid growth of the Internet infrastructure, coupled with the flexibility of the end-to-end service it provides, has led to a vigorous development of Internet usage and the appearance of a wide variety of applications. Many of these applications like skype [3], KaZaA [4], Akamai [5] and BitTorrent [6], are based on the overlay technology. For these applications, the positioning of each host with respect to the entire overlay is important for its configuration and for the dissemination of content through it. It is also important for tracking any change in network conditions and for the reconfiguration of the overlay accordingly.

Naturally, the positioning in the Internet can be achieved by exhaustive and regular measurements between all hosts involved in the overlay communication. However, given the large size of these overlays, this solution becomes unfeasible. One must add to this difficulty the impracticality of a solution per application and the interest of a universal positioning service.

This work was supported by the French ANR CMON project on Collaborative Monitoring.

Recently, a new approach has emerged for Internet positioning having the main advantage of providing estimates for network delays between machines at a low measurement cost. This approach consists in building a coordinate system for the Internet [7]. The basic idea is to embed all the nodes of a given application (or overlay) in some Euclidean space, and to associate to each node specific coordinates in this space in such a way that the network delay between any two nodes can be approximated by the geometric distance separating them. We can distinguish two main groups of network coordinate systems: the centralized coordinate systems like GNP [8] where we have a set of nodes named landmarks that are used as references to compute the coordinates of the other nodes, and the distributed coordinate systems like Vivaldi [1] where all nodes are similar and the coordinates are calculated in a distributed way without any central unit.

The Vivaldi system [1] is known to be one of the most promising approaches for Internet coordinates. It is based on a fully distributed, light-weight and efficient algorithm. By only measuring the network delay to few neighbors, and knowing the coordinates of these neighbors, a node can adjust its coordinates in steps until reaching the stable regime. The low cost of Vivaldi and the ease of its deployment have motivated us to run exhaustive measurements and simulations to understand the behavior of node coordinates and the extent to which they can be used. In particular, we are interested evaluating how stable are these coordinates and whether they can be leveraged to monitor a large network and to detect/diagnose any change as a long-term congestion, a rerouting, a link failure or any other event causing a shift in network delays. In the literature, Internet coordinates have been often studied from the perspective of estimating network delays. Here, we invert the problem and we study them from the perspective of network monitoring, given their light-weight feature and their distributed nature.

Vivaldi coordinates are largely covered in the literature. Several papers, [9] and [10] study the problem of coordinates security. In [11], [12], the authors show that network coordinates in the Vivaldi system are not stable but rather drift rapidly even when the network delays do not change. In this paper we show that, despite the instability of Vivaldi coordinates in their absolute values, there is still a stable internal structure that can better reflect the stability of the underlying network. Our contributions can be summarized in three main points. First, we confirm the fact that Vivaldi coordinates oscillate over time because of the adaptive nature of the system. The variations of these coordinates however are most of the time correlated with each other, pointing to some stable cluster of nodes seen from inside the system. The second contribution consists of a new clustering algorithm based on the data mining Hierarchical Grouping Method [2] to identify this cluster of stable nodes once the network and the host coordinates are in their stationary regime. The metric that we use to cluster nodes in the system is the amount of variation of their Euclidean distances with respect to each other. One important observation is that such stable cluster of nodes always exists and is grouping most of the nodes. As third contribution, we highlight the utility of the stable internal structure of Vivaldi coordinates with an application that tracks changes in network delays. To this end, and instead of tracking jointly the global matrix of network delays, we propose to track a simple signal being the size of this biggest stable cluster. By artificially changing the network delays in a simulation scenario, we show that these changes are easily reflected by this body of stable nodes, hence allowing to obtain a global picture about the stability of the underlying network without the need for exhaustive delay measurements.

The remaining of the paper is organized as follows. In Section 2 we describe the Vivaldi coordinate system. Section 3 explains our experimental methodology and presents first results to motivate the work. Section 4 presents our clustering algorithm and Section 5 summarizes the experimental results confirming the presence of a stable internal structure and how network changes are reflected by this stable structure. Finally, we conclude the paper in Section 6 with some perspectives on our future research.

II. Vivaldi description

Vivaldi is a fully distributed, light-weight and efficient algorithm, requiring no fixed network infrastructure and no distinguished nodes. It is able to estimate network delays accurately while minimizing the load on the network. The basic ideas of Vivaldi can be summarized as follows:

- Every node has an estimation of its coordinates.
- It picks a random node and asks for its coordinate estimation.
- It computes the Euclidean distance between itself and this selected node and compare this distance to the network delay between them.
- It calculates the stretch of the path and decides on the change to introduce into its coordinates.
- All nodes keep applying this algorithm in a decentralized manner until all the system stabilizes.

In Vivaldi each node calculates its own coordinates by performing periodic measurements of Round-Trip Time (RTT) with a small number of randomly selected nodes called neighbors. After each measurement made with one of these neighbors, the node performs an update of its coordinates: it approaches or recedes from its neighbor in an incremental way minimizing the difference between the RTT estimated using the coordinates and the measured RTT.

III. Experimental methodology and work motivation

A. Experimentation description

We use for our experimentation the PlanetLab platform [13] which contains more than 1000 nodes spread across the world in approximately 489 sites. Over PlanetLab we deploy an implementation of Vivaldi developed by the Harvard university and named Pyxida [14]. As Pyxida is implemented in Java, it requires lot of memory and processing power. We successfully managed to get meaningful coordinates from 145 PlanetLab nodes. The other nodes are simply eliminated from the experiments. Those experiments were carried out between February and September 2009 with the following parameters for the Vivaldi algorithm. We fix the number of neighbors per node to 32 and every node chooses its neighbors arbitrary. We use an Euclidean space of four dimensions plus the height to calculate the coordinates of each node. Every 10 seconds, each node pings (we use applicative UDP ping) one of its 32 neighbors and updates its coordinates. This leads to approximately one tour every 320 seconds. For coordinate measurements and collection, every node sends its coordinates to a collection server every 10 seconds where they are stocked and used afterwards for our analysis.

B. Simulations description

Our experimentations over PlanetLab are very useful to understand the coordinate variations of the Vivaldi system in a real network and to develop our clustering algorithm. One problem with these experimentations however is the impossibility to provoke network delay changes (anomalies) inside the PlanetLab network, because it is used by hundreds of researchers around the world and we do not control the intermediate routers and links. It follows that we cannot use the experimentations over PlanetLab to study the impact of network delay changes on Vivaldi coordinates and on our clustering algorithm. This clearly limits the evaluation of the capacity of our algorithm to detect network changes.

To counter the above limitation of PlanetLab experimentations, we run extensive simulations in addition. To be close to reality, we simulate a real topology of PlanetLab provided by the iPlane [15] archive. In fact, iPlane gives us all the data about the routes between sources and destinations on PlanetLab, the list of routers on the path, the delays between all routers and all routers interfaces. From these data we extract the PlanetLab topology with all detailed paths and routers between 200 PlanetLab nodes. By this way we are able to generate all kinds of network delay changes that we want to study. In this paper we show the results for one scenario changing the network delay. We leave the complete study of network delay anomalies to a future work.

As we experience a convergence time of coordinates around 2000 seconds (similar results were found in [11]), we fix the duration of all simulations to 30000 seconds. The scenario of network delay changes that we study in this work is as follows. Among the 200 nodes that we have, we create a network delay change influencing 50 nodes. For that, we increase the delay of the links influencing these 50 nodes from their normal values to a value equal to 30 times the normal values (this can be seen as a link outage or a severe congestion on those links). From time 0s to time 5000s we run the simulation with the normal settings, then we provoke the delay anomaly from time 5000s to 15000s. From 15000s till the end of the simulation we return to the normal settings. The impact on Vivaldi coordinates will be illustrated.

C. *Work motivation*

As Vivaldi coordinates form an effective way for estimating network delays, one can make the supposition that a link exists between network delay conditions and network coordinates. To confirm this supposition and evaluate the existence of such link, we start by observing the stability of coordinates in their absolute values over a network that does not present important shifts in its delays. Unfortunately, and as we will see next, such coordinates oscillate so much that makes it impossible to link them to any change in network conditions. This is the main motivation for the clustering algorithm that will come next.

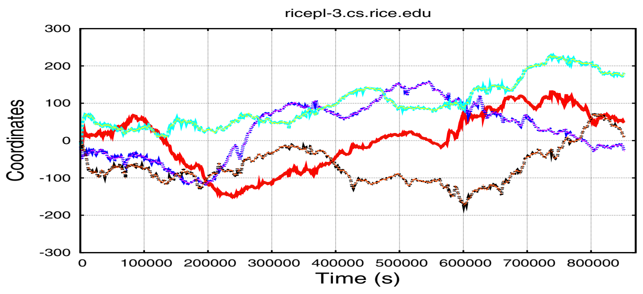


Fig. 1. Coordinate variations(Rice university)

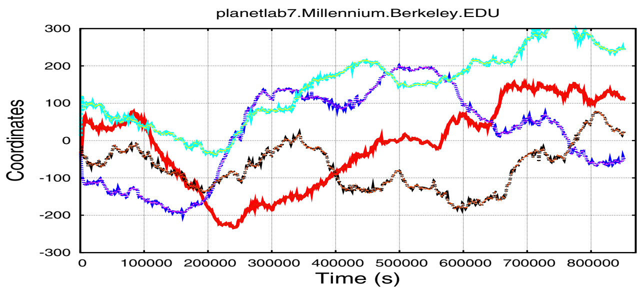


Fig. 2. Coordinate variations(Berkeley university)

In Figures 1 and 2, we present the variations of machine coordinates as a function of time for a relatively stable PlanetLab [13] node having a stable RTT between it and most of the other nodes (Figures 1) and for another instable PlanetLab node (Figures 2) experiencing an important variability in the RTT between it and all the other nodes. We can clearly observe the large variations of the four dimensions coordinates, for both the stable and instable node. This is mainly caused by the Vivaldi adaptive algorithm and the absence of any fixed reference point in the Euclidean space. We can conclude that by tracking simply the coordinates in their absolute form, we cannot differentiate between a shift due to a normal behavior or a shift caused by a change in network delay conditions (anomaly).

While observing coordinates, we discovered a strong relation between the shifts of all nodes. In particular, we noticed that generally nodes move together, which means that all the system shifts even if there is no problem. But, we also noticed that some particular nodes shift more than others because of the instability of their paths. This pushes us to introduce a clustering technique with the main goal to differentiate between nodes according to the value of their shifts with respect to each other.

IV. Clustering algorithm

We present a new clustering algorithm based on the Hierarchical Clustering Method [2]. The main idea of our clustering algorithm is to start by putting all participating nodes in the same cluster. Then in a second step, all nodes that are far from at least one other node in the cluster will be excluded from the cluster. These excluded nodes will pass another round to form their own cluster. At the end, each cluster will contain nodes which are all mutually close to each other. The metric that we use to cluster nodes in the system is the amount of variations of their Euclidean distances with respect to each other. The larger the variations of the Euclidean distance of two nodes, the farther they are from each other.

A. *Definitions*

- K: Number of participating nodes.
- N: Number of dimensions of the Vivaldi Euclidean space.
- $\vec{x}_i(t)$: The coordinates of node i at time interval t.
- $E_{ij}(t)$: The Euclidean distance of two nodes i and j at time interval t:

$$E_{ij}(t) = \sqrt{\sum_{d=1}^{N}(x_i(t)[d] - x_j(t)[d])^2}.$$

- $\Delta E_{ij}(t) = |E_{ij}(t) - E_{ij}(t-1)|$: The shift in the Euclidean distance between two nodes i and j at time interval t compared to the previous interval.
- In Vivaldi, each node has 32 neighbors. Every 10s it sends a ping to one of its neighbors and updates its coordinates. Thus, w e use t = 10s as the frequency with which our clustering algorithm is executed.
- ϵ: This is the threshold of allowed variations of Euclidean distances (expressed in ms). Nodes with variations in their Euclidean distances to each other below ϵ (with some confidence level) will be in the same cluster. We have tested our algorithm with different values of ϵ over the different experiments in our work.
- Violation indicator $s_{ij}(t)$: The membership of two nodes i and j to the same cluster at time interval t is violated when the shift in the Euclidean distance between them is larger than the threshold ϵ: $s_{ij}(t) = 1$ if $\Delta E_{ij}(t) > \epsilon$ and $s_{ij}(t) = 0$ otherwise.

- C, or the confidence level, denotes the ratio of membership tests that the nodes should pass for them to be declared in the same cluster. Differently speaking, two nodes do not belong to the same cluster, i.e., $v_{ij} = 1$, if the number of violations of their Euclidean distances satisfies:
$$\sum_{t_i=t-I+1}^{t} s_{ij}(t_i)/I > 1 - C.$$
Otherwise, we set the variable v_{ij} summarizing the result of the test to 0.
- I: Number of past time intervals of 10s to be considered for the clustering. After trying several values for I, we specify it to 100 in this paper. This means that at any time t, the clustering is performed over the 100 past intervals of 10s each.

B. *Clustering algorithm description*

The main goal of our algorithm is to find the maximum number of nodes in each cluster that satisfy the threshold ϵ for their Euclidean distance shifts. The algorithm starts from a sorted list of nodes according to their number of violations with respect to all other nodes, and it ends with a set of clusters that contain nodes without any violation with each other. The details of our algorithm are as follows:

- First, we calculate the number of violations from each node to all other nodes, i.e., $V_i = \sum_{j=1}^{K} v_{ij}$.
- We sort nodes by their decreasing number of violations, V_i. The algorithm here starts with all nodes in the same cluster.
- To find the first cluster, we exclude the node with the largest number of violations, then we recalculate the number of violations for the remaining nodes. We continue excluding nodes with large number of violations until there is no violation between the nodes in the cluster. We are sure that the distance between any two nodes in this cluster shifts by less than the threshold ϵ within the confidence level C.
- We repeat the above steps to form a second cluster with the excluded nodes.
- We stop the algorithm when we are left without any excluded node.

V. Experimental results

We run our clustering algorithm every 10s over both the PlanetLab and the iPlane traces. Next we study the distribution of the number of clusters and their sizes. We recall that a cluster is composed of nodes that are relatively stable (between them) according to the ϵ threshold used and the confidence level C. Roughly speaking, the shift of the Euclidean distance between any node i in a cluster and any other node j in the same cluster, $\Delta E_{ij}(t) = |E_{ij}(t) - E_{ij}(t-1)|$, does not exceed ϵ in more than C% of the 100 past measurement intervals.

A. *Cluster size distribution with different ϵ values*

In this subsection we study the impact of ϵ on the output of our clustering algorithm. To this end, we fix the confidence level C to 90%, and we perform the clustering while changing the value of ϵ. We present in Figures 3(a, b, c and d) the distribution of the sizes of the first four clusters that we find for different values of ϵ and as a function of time. In Figures 3(a and b) we can observe that the size of the biggest cluster is around 37 and 85 nodes respectively among the 145 PlanetLab nodes. This relatively small size of the biggest cluster is normal because we set the value of ϵ to 4ms in (a) and 8ms in (b) respectively, which means that we only allow a small shift in Euclidean distances compared to the shift inherent to Vivaldi dynamics. For the other clusters, we always have a small number of nodes, smaller than 7 and 10 nodes respectively. We can say that nodes belonging to the biggest cluster for these small values of ϵ are stable at the scale of measurement interval t and can be used to detect small shifts in the network delays between them. We can also observe that the size of the biggest cluster itself changes over time and that we have some nodes leaving and entering this cluster for these small values of ϵ. This is the result of variations in Euclidean distances that can be caused either by Vivaldi dynamics or by small shifts in network delays.

In Figure 3(c) and for ϵ equal to 16ms, we notice that the size of the biggest cluster becomes more important and more stable and by slightly oscillating around 125 nodes among the 145 total PlanetLab nodes. We also notice how the sizes of the other clusters are always smaller than 5 nodes.

In Figure 3(d) we observe that the size of the biggest cluster is even larger and has smoother oscillations around 142 nodes among the 145 total PlanetLab nodes. Here, we use an ϵ equal to 64ms, which means that we allow a shift in Euclidean distances of 64ms. The size of the other clusters than the biggest one is always smaller than 1 node. Here we can say that we approximately have all the nodes in the same cluster, and that this biggest cluster is relatively stable, with only few nodes changing the cluster to which they belong.

From the above figures we can confirm the presence of a main stable biggest cluster. When we use a small ϵ value, we obtain a small cluster containing very stable nodes that are almost fixed with respect to each other. However, when we use a large ϵ, we obtain one main biggest cluster that contains approximately all the nodes. The distance shifts between these nodes is now larger, which limits the capacity of the algorithm to track small changes in network delays. The advantage compared to the former case is that we now have more stable nodes to track.

B. *Biggest cluster stability for different C values*

In Figures 3(e) and 3(f) we study the stability of nodes in the biggest cluster as a function of the confidence level C and the ϵ threshold used. The previous results were specific to one C value, here we span the space of C and ϵ. In Figure 3(e) we present the proportion of stable nodes as a function of ϵ for different confidence levels and after averaging over the duration of the simulation. Figure 3(f) presents the results for unstable nodes in the same scenarios. We can clearly observe that when we use a low confidence level and for all ϵ values, the biggest cluster becomes more stable, especially when we use large ϵ values. For example, if we use a confidence level of 85% and an ϵ of 64ms, we obtain a very stable biggest cluster where 99.5% of nodes are inside. We can conclude from these figures that in the normal case and for a large ϵ value and a low confidence level, we have a stable biggest cluster that reflects well the state of the participating nodes. Unfortunately, the biggest cluster in this latter case has limited capacity to detect shifts in network delays since most of the shifts below ϵ will pass undetected. The capacity to detect larger shifts than ϵ is however still effective.

C. Biggest cluster stability as a function of time

In Figures 3(i and j) we plot the percentage of nodes staying in the biggest cluster for different time durations (in percentage compared to the total duration of the simulation) and for two values of the confidence level, 80% (Figure 3(i)) and 90% (Figure 3(j)). We can clearly observe that for a large value of ϵ, the majority of nodes stay all the time in the biggest cluster, and hence the size of the biggest cluster is a stable signal that, if tracked over time, should allow to detect any abnormal shift in the delay of links crossed by Vivaldi probes.

D. Impact of network changes on the size distribution of clusters

In this subsection, we study the impact of a change in network delays on the distribution of the sizes of clusters. To achieve this goal, we use the scenario of network delay change explained in Section III.B and we study the impact of this scenario on the coordinate system by making a comparison with the normal case. We fix the confidence level C to 90% for all the results in Figures 3(g and h).

In Figures 3(g) and 3(h) we present the average proportion of nodes in the biggest cluster as a function of ϵ for the normal case 3(g) and the abnormal case 3(h). The proportion of nodes in the biggest cluster clearly decreases during the abnormal case for all the values of ϵ, and this decrease becomes more and more clear when we use a larger ϵ. For example, for ϵ equal to 64ms, the proportion of nodes in the biggest cluster decreases from 99% in the normal case to 78% in the abnormal case. Indeed, for this delay anomaly pattern, increasing the value of ϵ makes more nodes involved hence easying the detection of the anomaly.

We can conclude from the above figures that the number of nodes in the biggest cluster is a good parameter to differentiate between a normal and an abnormal network scenario. This can be done by monitoring the movement of stable nodes out of the biggest cluster. The stable nodes to monitor are those declared frequent biggest cluster residents when the network is in its normal conditions. As we have seen, their number is function of ϵ, which is to be set according to the delay anomaly levels to be detected.

VI. Conclusions and future work

In this paper, we studied the Vivaldi coordinate system and we showed that despite the instability of coordinates, we can still use them to track network changes. We started by developing a new clustering algorithm that allows to group the stable nodes together following the amount of variations of their coordinates. The clustering pointed out the presence of a stable biggest cluster which contains most of the nodes. We showe that the size of this biggest cluster reflects well the state of the network, and so a node that belongs to this biggest cluster is a stable and normal node. By tracking the number of nodes in the biggest cluster, one can detect main changes in network delays. The amount of changes to detect and the confidence level of tracking can be set by the network administrator. With our algorithms, coordinates can then become an efficient way for a light-weight network diagnosis that does not require the deployment of a complex monitoring infrastructure. In our future work, we will push the study further by designing a distributed protocol for clustering and for the detection and localization of network delay anomalies. We are also planning to conduct an exhaustive experimental study to validate the performance of such protocol by considering a large set of scenarios for changing network delays.

References

[1] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *SIGCOMM '04*. New York, NY, USA: ACM, 2004, pp. 15–26.

[2] J. H. Ward, "Hierarchical grouping to optimize an objective function," *journal of the American Statistical Association*, pp. 236–244, 1963.

[3] Skype, "Skype," http://skype.com/, 2009.

[4] KaZaA, "Kazaa media dekstop," http://www.kazaa.com/, 2009.

[5] Akamai, "Akamai technologies," http://www.akamai.com/html/misc/requirements.html, 2009.

[6] BitTorrent, "Bittorrent," http://bitconjurer.org/BitTorrent/protocol.html, 2009.

[7] H. Lim, J. C. Hou, and C.-H. Choi, "Constructing internet coordinate system based on delay measurement," *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, pp. 513–525, 2005.

[8] T. NG and H. ZHANG, "Predicting internet network distance with coordinates-based approaches." in *IEEE Infocom '02*. IEEE, 2002, pp. 170–179.

[9] M. A. Kaafar, L. Mathy, C. Barakat, K. Salamatian, T. Turletti, and W. Dabbous, "Securing internet coordinate embedding systems," in *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2007, pp. 61–72.

[10] G. Wang and T. S. E. Ng, "Distributed algorithms for stable and secure network coordinates," in *Proceedings of the ACM/USENIX Internet Measurement Conference (IMC'08)*, Oct 2008.

[11] J. Ledlie, P. Gardner, and M. Seltzer, "Network coordinates in the wild," in *Proceedings of the Fourth USENIX Symposium on Network Systems Design and Implementation (NSDI)*, April 2007.

[12] J. Ledlie, P. Pietzuch, and M. Seltzer, "Stable and accurate network coordinates," in *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2006, p. 74.

[13] Planetlab, "Planetlab," http://www.planet-lab.org/, 2009.

[14] Pyxida, "Pyxida: An open source network coordinate library and application," http://pyxida.sourceforge.net/, 2009.

[15] Iplane, "Iplane, an information plane for distributed services," http://iplane.cs.washington.edu/, 2009.

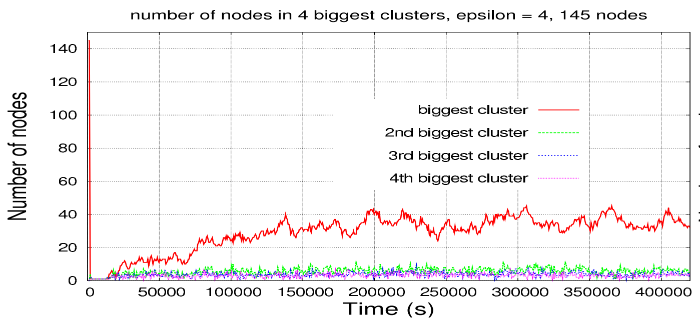


Fig.3(a) (ϵ = 4ms)

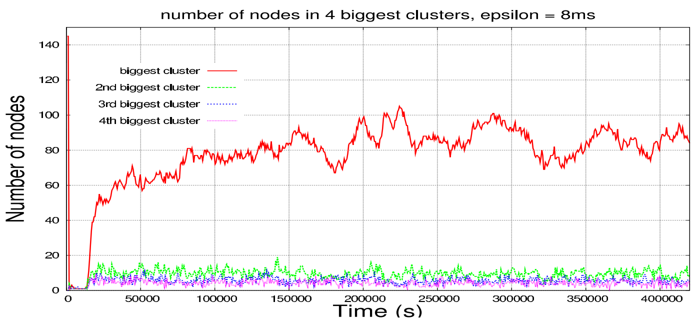


Fig.3(b) (ϵ = 8ms)

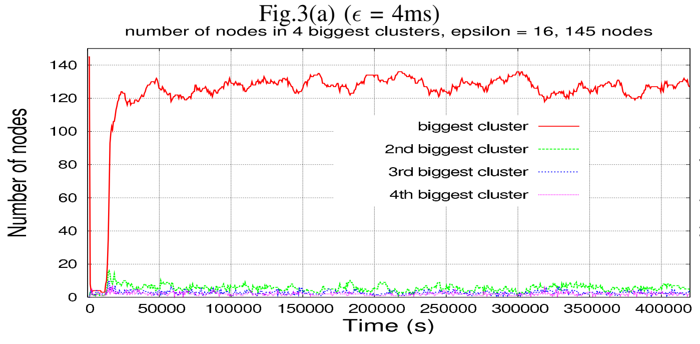


Fig.3(c) (ϵ = 16ms)

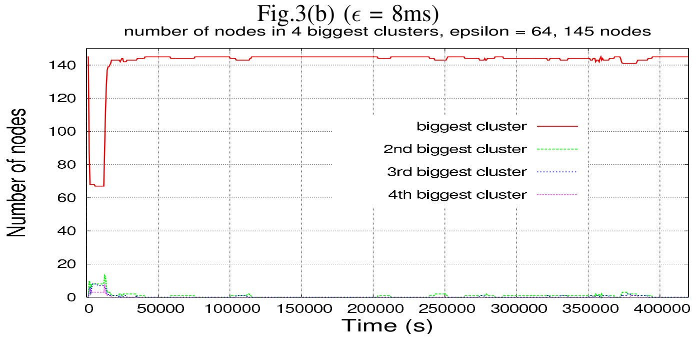


Fig.3(d) (ϵ = 64ms)

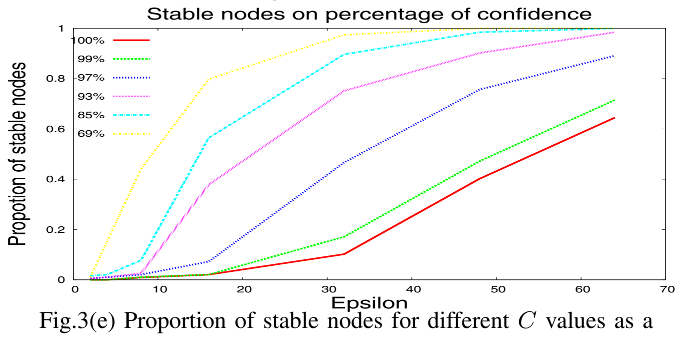


Fig.3(e) Proportion of stable nodes for different C values as a function of ϵ

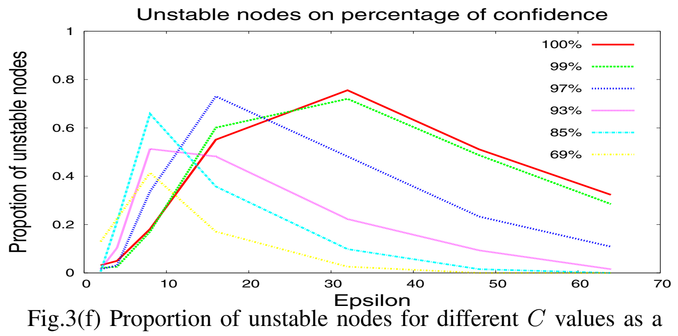


Fig.3(f) Proportion of unstable nodes for different C values as a function of ϵ

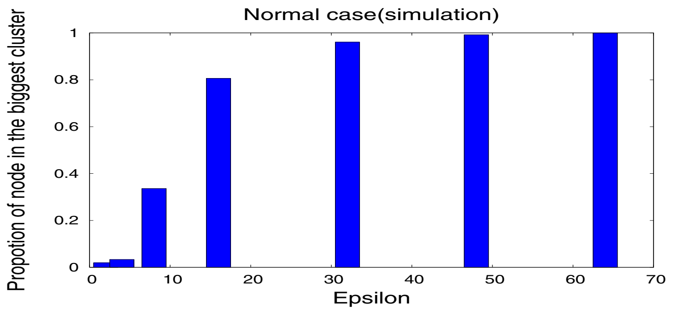


Fig.3(g) Biggest cluster(normal case)

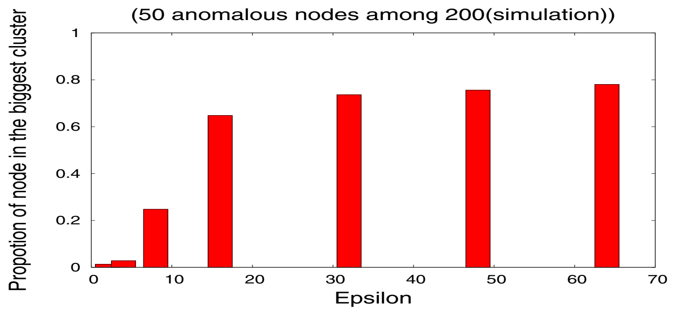
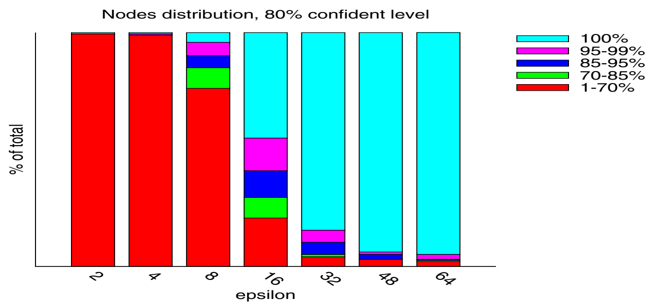


Fig.3(h) Biggest cluster (simulation, 50 abnormal nodes among 200)



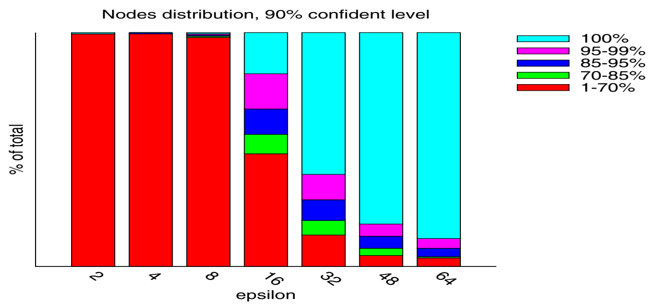


Fig.3(i) Stability of the biggest cluster as a function of time(C = 80) Fig.3(j) Stability of the biggest cluster as a function of time(C = 90)

Fig. 3. Cluster size distribution for different C and ϵ values