Published in IET Communications
Received on 22nd November 2008
Revised on 10th February 2009
doi: 10.1049/iet-com.2008.0671

ISSN 1751-8628

Handling node churn in decentralised network coordinate system

Y. Chen[1] *G. Zhao*[2] *A. Li*[3] *B. Deng*[1] *X. Li*[1]
[1]*Department of Electronic Engineering, Tsinghua University, Beijing 100084, People's Republic of China*
[2]*Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA*
[3]*Department of Computer Science, Duke University, Durham, NC 27708, USA*
E-mail: chenyang04@mails.tsinghua.edu.cn

Abstract: A Network Coordinate (NC) system is an efficient mechanism to predict Internet distance with scalable measurements. In this paper, we focus on the node churn problem – the continuous process of nodes arrival and departure – in distributed applications. Studies on Vivaldi, a representative distributed NC system, show that under node churn the prediction accuracy of the NC system will be seriously impaired. In this paper, we focus on how to handle the impact of node churn in Vivaldi. Firstly, we propose a simple solution by directly increasing the measurement frequency. Our experiments have demonstrated that this approach can reduce the harm of node churn. However, it increases the communication overhead as the measurement frequency grows. To avoid such expensive solution, we propose the design and implementation of Myth, a decentralised and fast convergence NC system. It introduces the merit of Landmark-based NC system to shorten convergence time in Vivaldi with slight extra overhead. Our experimental results show that Myth outperforms Vivaldi a lot under node churn, without compromising the performance under stable environment. Moreover, we have found that the use of Myth is a cost-effective way to achieve higher prediction accuracy; it will not only improve the prediction accuracy but also save the communication overhead.

1 Introduction

A Network Coordinate (NC) system is a scalable mechanism to predict the network latency between any two Internet nodes without direct end-to-end measurements. Each node is assigned a set of numbers called NCs to represent its position in an Euclidean space, and the latency between any two nodes can be calculated from their coordinates with a distance function. The NC system can estimate the Internet latency accurately as well as reduce the active probing overhead significantly. Thus it is extremely useful in large-scale distributed applications. To date, NCs have demonstrated their merit in a wide variety of applications ranging from overlay multicast [1], server selection [1], distributed query optimisation [2], file-sharing via BitTorrent [3], compact routing [4] and application layer anycast [5].

Several approaches have been proposed in literature, whose core algorithms can be categorised into two classes, namely landmark-based algorithm (LBA) and simulation-based algorithm (SBA). In LBA systems [6–8], each node measures its distances to a set of reference nodes called *landmarks*, and its coordinates can be determined by minimising the difference between the measured distances and the calculated distances. LBA provides high accuracy and stability. However, the dedicated landmark nodes are under the heavy load of serving all the other nodes in the system. In other words, LBA has bad scalability. As for SBA systems, such as [9] and [10], pairwise distances are mapped into a physical spring system, so that every node's coordinates can be determined by minimising the energy of the whole simulated physical system. SBA systems distribute the measurement and computation to all participating nodes in order to lighten the load of any individual node. Thus SBA guarantees better scalability and has received more practical implementations.

However, distributed systems have to deal with churn-change in the set of participating nodes due to joins, graceful leaves and failures, as indicated in [11]. In Vivaldi, a representative SBA system, each node starts from the same initial position, updates its coordinates by referring to the measured distance to one of its neighbours and that neighbour's current coordinates and converges to the ideal position after many rounds of updates. Given the distributed nature of Vivaldi, it will in general take tens of seconds for a node to converge to its ideal position. During this period of time, the algorithm is vulnerable to the departure and arrival of its neighbours, because a converging node may lose the good reference from a departing converged neighbour, and may refer to some newly joined nodes whose coordinates are still very inaccurate. Thus, under node churn the prediction accuracy of Vivaldi will be significantly impaired. Ledlie *et al.* [12] confirms this by analysing the statistics obtained from Azureus BitTorrent client running upon Vivaldi.

This paper focuses on handling the node churn problem of Vivaldi, with the major contributions as follows:

First, we propose a simple and straightforward solution to node churn. We demonstrate that simply increasing the measurement frequency of Vivaldi can reduce the decrease of prediction accuracy caused by node churn; however, higher communication overhead is needed. Thus this approach is not a cost-effective solution.

Second, we propose Myth, a fully decentralised NC scheme, to solve this problem. Myth is fully compatible with Vivaldi. In Myth, a scalable algorithm is designed to intelligently choose the initial coordinates of nodes based on the information of existing nodes. After the initial coordinates are determined, basic Vivaldi algorithm is utilised to achieve convergence and guarantee scalability. By using the algorithm, the initial position of a Myth node is close to its final ideal position; hence the convergence duration is greatly shortened. Furthermore, because the initial coordinates of a Myth node are more accurate than the ones of a Vivaldi node, the impairment of referring to a newly joined node as neighbour is reduced. The extra overhead of using Myth is moderate. We evaluate the performance of Myth by comparing it with the Vivaldi system. The experimental results show that in node churn scenario Myth outperforms Vivaldi under every metric we collect. Moreover, we found that using Myth is a much better way to achieve high prediction accuracy instead of increasing each node's measurement frequency of Vivaldi; it can not only improve the prediction accuracy but also save the communication overhead. We believe that Myth is a step further towards a practical NC system in real Internet.

The rest of this paper is organised as follows. In Section 2, we study the relationship between the measurement frequency and the prediction accuracy of Vivaldi under node churn. Then in Section 3, we present the design and implementation of Myth, followed by its performance evaluation in Section 4. In Section 5, we summarise the conclusions.

2 Impact of node churn for Vivaldi

In this section, we first briefly introduce the definition of NCs and the basic algorithm of Vivaldi. Then we study its performance degradation under node churn.

2.1 Definition of NCs

Suppose we have N Internet nodes. Let S be the set of these N nodes. Let L be the $N \times N$ distance matrix among the nodes in S. Thus $L(i, j)$ represents the measured round-trip time (RTT) between node i and node j.

Basically, NC is an embedding of these N hosts into m-dimensional Euclidean space R^m. We define x_i as the NC of node i, we have $x_i = (r_1^i, r_2^i, \ldots, r_m^i)$.

We can use the x_i and x_j to estimate the RTT between node i and node j. We use $L^E(i, j)$ to represent this estimated RTT. The definition of $L^E(i, j)$ is as follows

$$L^E(i,j) = \| x_i - x_j \| = \sqrt{\sum_{1 \le k \le m} (r_k^i - r_k^j)^2} \tag{1}$$

To serve thousands of nodes effectively, an NC system should be scalable. Each node only does restricted measurements to calculate its NC. The existing systems [6, 7, 9, 10, 13] use only $O(N)$ measurements. Thus the total measurement overhead under NC is much lower than the $O(N^2)$ measurements required for a full mesh of N nodes.

The prediction accuracy of an NC scheme is often depicted by the relative error (RE) of predicted distance over the real latency measured on Internet. RE between node i and node j is defined as [7, 8, 14–16]

$$\mathrm{RE} = \frac{|L^E(i,j) - L(i,j)|}{L(i,j)} \tag{2}$$

Smaller RE indicates higher prediction accuracy. When measured latency equals predicted latency, the RE value will be zero.

2.2 Vivaldi algorithm

In our work, we focus on the improvement of the accuracy of Vivaldi [9], which is called as the most widely used SBA system in [17] because of its clean and decentralised implementation. Vivaldi is studied in [18, 19] as the representative NC algorithms, and it is deployed in many well-known Internet systems, such as Bamboo DHT [20], Stream-Based Overlay Network (SBON) [2] and Azureus BitTorrent [3]. Before going into the details, we first briefly introduce the basic procedure of Vivaldi.

Vivaldi characterises the whole network as a spring system. Let $L(i, j)$ be the actual distance (RTT) between node i and node j in the system, and x_i be the coordinates assigned to node i. Ideally, the coordinates of all nodes are produced by minimising the following error function which corresponds to the overall energy. We refer to those coordinates as ideal coordinates of all nodes.

$$E = \sum_i \sum_j (L(i, j) - \| x_i - x_j \|)^2 \qquad (3)$$

$||x_i - x_j||$ is the distance between node i and node j calculated based on coordinates.

In practical decentralised Vivaldi system, node i maintains its current coordinates x_i and local error e_i, and updates them for many rounds to get closer to the ideal coordinates. In each round, node i adjusts them through measuring the latency to one of the other nodes in the system. The pseudo-code of Vivaldi is shown as follows, where c_e and c_c are tunable parameters. In this pseudo-code, node i updates its coordinates x_i by probing node j. rtt means the latency between node i and node j. For each node i, the initial value of e_i is set to 1.

Algorithm 1 Vivaldi (rtt, x_j, e_j)

```
1: $w = e_i/(e_i + e_j)$
2: $e_s = |\|x_i - x_j\| - \mathrm{rtt}|/\mathrm{rtt}$
3: $e_i = e_s \times c_e \times w + e_i \times (1 - c_e \times w)$
4: $\delta = c_c \times w$
5: $x_i = x_i + \delta \times (\mathrm{rtt} - \| x_i - x_j \|) \times u(x_i - x_j)$
```

A sample weight is firstly computed based on the local and remote error (line 1), and then the RE is computed (line 2). Next node i updates its local error (line 3). Finally node i calculates and updates its coordinates (line 4 and line 5).

In Vivaldi, all nodes have the same initial coordinates. To bootstrap the algorithm Vivaldi defines $u(0)$ as a unit-length vector in a randomly chosen direction. When two nodes occupy the same location, there will be a spring pushing them away from each other in an arbitrary direction.

2.3 Evaluation of vivaldi under node churn

In most NC applications, especially peer-to-peer content distribution or file sharing, all nodes may join or leave the system at any time. This inherent dynamics and distributed nature make the deployment of NC even more challenging.

Dabek *et al.* [9] only studied Vivaldi under stable environment where all nodes will not leave the system after they join. However, this is not always the case in practice. Ledlie *et al.* [12] shows that the client lifetime in file sharing using Azureus (running upon Vivaldi) follows a long-tailed distribution typical in peer-to-peer (P2P) systems. In their study, 78% of the nodes stay in the system for less than 60 min.

We evaluate Vivaldi in two scenarios and compare the prediction accuracies.

Stable scenario: All nodes join the system at the beginning and stay throughout the whole experiment.

Churn scenario: We use the synthetic traces used in [11] to generate the node churn. As defined in [20], a node's session time is the elapsed time between its joining the network and subsequently leaving it. As in [11], we use session times with PDF $f(x) = ab^a/(x + b)^{a+1}$ with exponent $a = 1.5$ and b fixed so that the distribution has the mean of 30 min unless otherwise stated. This is a standard Pareto distribution, shifted b units (without the shift, a node would be guaranteed to be up for at least b minutes). Pareto distribution can describe the long-tailed characteristic of nodes' staying time, which is used for simulating the node staying time in both [11] and [21]. Between each session we use exponentially distributed downtimes with the mean of 2 min [11].

There are some implementations of Vivaldi protocol, such as [12, 20]. In our simulation we choose the protocol used in [20] because it is part of the Bamboo DHT, which is a mature, robust and open source DHT that has run on PlanetLab 24/7 for more than 3 years. Bamboo DHT serves as the infrastructure of the OpenDHT [22].

In the Vivaldi of Bamboo, the default way of collecting latency samples to update the local node's NC is to pick a random node in the overlay network by the fixed *Update Interval*. These sample nodes are uniformly chosen. Once one node is found, the local node then pings this random chosen node and retrieves its NC (together with the value of local error). The NC of the local node is updated and the above procedure is repeated every t seconds. We define the above procedure which is repeated every t seconds on each node as one *probe*. In other words, one probe includes the following three steps: (a) Generate one GUID (Global Unique Identifier) randomly then query the DHT for it. The DHT will return the node which is responsible of this GUID. This node will be the randomly chosen neighbour. (b) Measure the distance from itself to this randomly chosen neighbour. Also, get the current NC (together with the value of local error) of this randomly chosen neighbour. (c) Update the own NC with the distance and NC obtained in the last step.

There are two kinds of Vivaldi implementations. This kind of Vivaldi implementation is called *full Vivaldi embedding* [23, 24], – all other nodes can be used as the referenced node for every node. The other kind of implementation is that each node maintains a fixed neighbour set and updates its own NC by probing nodes only from this neighbour set. The design of this kind of implementation of Vivaldi has not considered the node churn. For example, the following two questions have not been answered in [9]. (a) Once one of a node's neighbours is down, how can it detect this? (b) How soon can it get a new neighbour to take the place of the left one? Because of these uncertain features of the implementation using a fixed neighbour set, instead, full Vivaldi embedding is used in our simulation.

We use two different data sets from real Internet measurement to study the prediction accuracy of Vivaldi. The first data set is the King data set from [9], which includes the round-trip latencies among 1740 Internet naming servers. The second data set is the PlanetLab data set, available at [25], which includes the round-trip latencies among 226 hosts of the PlanetLab. Both of these two data sets are used frequently in NCs simulation. We evaluate Vivaldi in both stable and churn scenarios. The parameters of the simulation are defined in Table 1. To implement Vivaldi, m is set as 5 [20], c_c and c_e in Vivaldi are set to 0.25 as an empirical value in [9]. Specially, in churn scenario, we set t as both 10 and 20 to see the effect of the different update intervals.

Fig. 1 and Table 2 show the comparison of the REs of Vivaldi between two different scenarios. According to Fig. 1, we can find that the performance of Vivaldi degrades in node churn scenario. In churn scenario, when t equals 10, average RE is increased by 126.32% in PlanetLab data set and by 66.67% in King data set, when t equals 20, average RE is increased by 231.58% in PlanetLab data set and by 108.33% in King data set. Vivaldi's performance degradation in node churn scenario will seriously limit its application, which confirms the conclusion in [12]. Furthermore, we can also find that smaller t can lead to better accuracy under node churn. Thus increasing the measurement frequency is a simple and straightforward way to improve the performance of Vivaldi under node churn. Obviously, this approach will definitely cause increased communication overhead because smaller t will lead to more probe.

Table 1 Parameters of vivaldi simulation

Parameter	Definition
m	dimension of the Euclidean space
c_c	tunable parameter in Vivaldi
c_e	tunable parameter in Vivaldi
t	NC update interval of a Vivaldi host

Table 2 Average RE Vivaldi

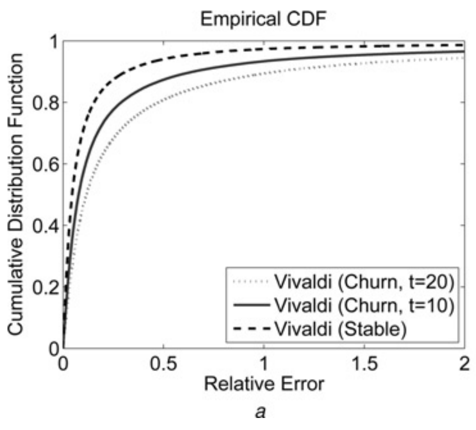
Data set	Scenario		
	Stable	Churn ($t = 10$)	Churn ($t = 20$)
PlanetLab	0.19	0.43	0.63
King	0.24	0.40	0.50

In the next section, we propose a more effective solution to node churn called Myth. In contrast to the above simple and straightforward solution, Myth can not only improve the prediction accuracy but also save the communication overhead.

3 Myth system design

3.1 Basic idea of myth

In Vivaldi, the coordinates of all the nodes start at the same initial position. These nodes update their coordinates for many rounds and their coordinates finally converge to their ideal positions. Fig. 2*a* shows the first three rounds of the



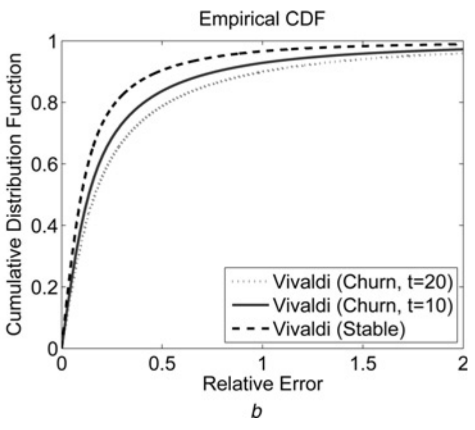


Figure 1 *Distribution of RE*

a Planet Lab
b King

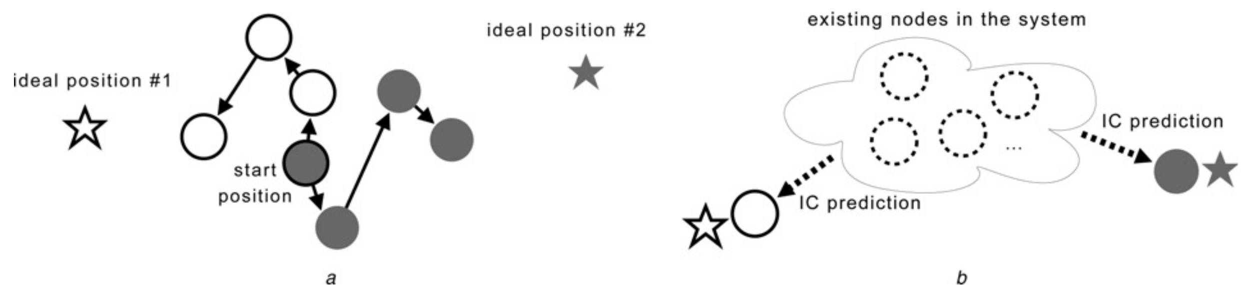


Figure 2 *Illustrations of coordinates update in Vivaldi and Myth*
a Vivaldi
b Myth

NC update of two nodes after their joining to the system. The two stars with different colours show the ideal positions of both nodes, respectively.

Here we argue that using the same initial coordinates for all nodes is not a good heuristic under node churn, even though nodes will be spread in the following update. First, extra rounds are needed to spread out the nodes from their initial coordinates, and hence the convergence time is increased. Furthermore, since a newly joined node may serve as the reference for its neighbours, its irrelevant and inaccurate initial coordinates will affect the prediction of its neighbours. Thus under high node churn rate, a large portion of the nodes are in the state of disturbance with inaccurate NCs.

To solve this problem, in Myth we manage to bootstrap each node with the initial coordinates close to its ideal coordinates. Fig. 2*b* shows the case. Myth is fully compatible with Vivaldi. In Myth, an lightweight initial coordinates (IC) prediction scheme is added before the periodical Vivaldi-like adjustment. This scheme utilises the coordinates of nodes which are already in the system. After doing this, each node needs fewer rounds to reach its ideal position. Moreover, the prediction error caused by referring to a newly joined node is reduced, which improves the stability of the system. In the following we present the scheme in details.

3.2 Initial coordinates calculation

One sparking point of Myth is to calculate the IC of each node, respectively, in a scalable fashion. When a new node A joins the overlay system, it firstly chooses L nodes ($L > m$, where m is the dimension of the space) randomly in the overlay. This random selection process can simply be done by using the DHT query since our Vivaldi implementation is built on Bamboo DHT. We can generate L random GUIDs and then query the DHT for them. The DHT will return L nodes which are, respectively, responsible of these GUIDs. After that node A measures its RTTs to these L nodes as well as retrieves the NCs of these L nodes. Using the distances and NCs collected, node A can calculate its own initial coordinates IC_A which minimise the overall error between the measured and the calculated distances from A to these L nodes. As in [6], we use Simplex Downhill Algorithm to minimise the following objective function $f_{\mathrm{obj}}(A)$.

$$f_{\mathrm{obj}}(A) = \sum_{N_i \in N_1 \ldots N_L} (1 - \min(1, e_{N_i})) \cdot \xi(A, N_i) \tag{4}$$

where $\xi(\cdot)$ is an error measurement function defined as follows.

$$\xi(N_1, N_2) = \left[\frac{L(N_1, N_2) - L^E(N_1, N_2)}{L(N_1, N_2)}\right]^2 \tag{5}$$

The idea of our IC prediction algorithm is similar to the NC calculation of the ordinary global network positioning (GNP) nodes in [6]. However, there is a significant difference between Myth and GNP or other LBAs: Myth does not require the deployment of dedicated landmark nodes. Our trick is to make use of the existing Myth nodes, which are already in the system, as landmark nodes. This solution is scalable since there are no fixed dedicated landmarks.

3.3 Myth implementation

Algorithm 2 Myth algorithm

```
Join_Myth_Overlay()
[rtt(·), x(·), e(·))] = Sample_Neighbours(L)
x_A = Initial_Coordinates_Prediction(rtt(·), x(·), e(·))
Wait(Update_Interval)
while forever do
    [rtt, x, e] = Sample_Neighbours(1)
    x_A = Vivaldi(rtt, x, e)
    Wait(Update_Interval);
end while
```

Table 3 The increase of total number of probes

Update_Interval (s)	Average stay duration		
	20, min	30, min	60, min
10	5.83%	3.89%	1.94%
20	11.67%	7.78%	3.89%

Algorithm 2 shows the procedure that a new node A joins a Myth overlay and starts its NC calculation. Node A joins Myth overlay using the procedure proposed in [20, 26]. After joining the Myth overlay A probes L nodes in its neighbour set as well as retrieves the NCs of these L nodes. These L nodes are randomly chosen by the procedure proposed in Section 3.2. Then A employs IC prediction algorithm to calculate its initial coordinates. After that node A can start the coordinates adjusting procedure and update its coordinates periodically using basic Vivaldi algorithm.

3.4 Overhead analysis

By adding the IC prediction scheme, Myth introduces extra communicational and computational overhead when a new node joins. The computational overhead is negligible because the time needed to compute an IC is far shorter than the time interval between update rounds, which is generally on the order of seconds. In this section we present a simple analysis of the communicational overhead imposed by Myth.

Suppose there are M nodes ($M > L$) in the swarm, and each node probes one of the other nodes updates its NC once per unit time. Thus for each node the communicational load (number of messages per update interval) of Vivaldi is 1. We do not take into account the sizes of messages because almost the same format of messages is used for all communications. During IC prediction, each node will contact extra $L - 1$ neighbours ($L < M$) to compute its IC (one neighbour information is shared with the first round of Vivaldi update). If one node's average staying duration is k seconds, the upper bound of the extra communicational load introduced by IC prediction is $(L-1)/(k/t) \times 100\%$, where the IC prediction happens for each node once per k/t update intervals. In practical settings, L is 8 in our experiment, and we vary the value of k to show different overheads under different node churn situations. The update interval is set to 10 and 20 s. Table 3 shows the results.

Table 4 Average RE

Data set		Scenario		
		Stable	Churn ($t = 10$)	Churn ($t = 20$)
PlanetLab	Vivaldi	0.19	0.43	0.63
	Myth	0.19	0.27	0.33
King	Vivaldi	0.24	0.40	0.50
	Myth	0.23	0.29	0.32

Even when the average node stay duration is as short as 20 min and the update interval is 20 s, the overhead of Myth is only increased by 11.67% of Vivaldi. If the average node stay duration reaches 60 min and the update interval is 10 s, Myth will add merely 1.94% communicational overhead to Vivaldi. Thus the extra overhead is acceptable.

4 Performance evaluation

4.1 Experiment setup

In our experiments, we use two data sets, the King and PlanetLab, to compare Myth and Vivaldi. Same as the parameters used in the Vivaldi simulation in Section 2.3, both systems employ five-dimension coordinates [20]. c_c and c_e in Vivaldi and Myth are set to 0.25 as an empirical value in [9]. Our experiment is performed both in stable

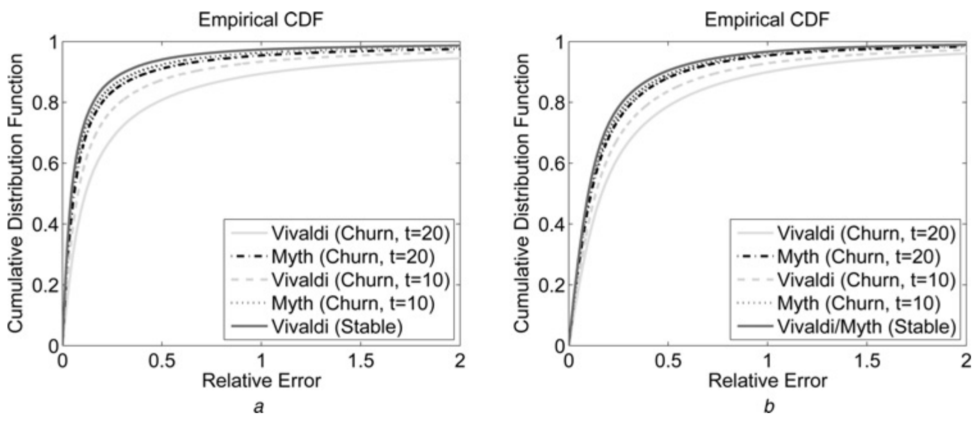


Figure 3 *Distribution of RE*

a Planet Lab
b King

Table 5 Average CNAE

Data set		Scenario		
		Stable	Churn ($t = 10$)	Churn ($t = 20$)
PlanetLab	Vivaldi	4.83	19.37	30.34
	Myth	4.17	13.65	17.41
King	Vivaldi	26.02	39.10	44.72
	Myth	24.84	33.08	36.47

Table 6 Average RRL

Data set		Scenario		
		Stable	Churn ($t = 10$)	Churn ($t = 20$)
PlanetLab	Vivaldi	0.05	0.09	0.13
	Myth	0.05	0.06	0.07
King	Vivaldi	0.11	0.16	0.19
	Myth	0.11	0.13	0.14

scenario and churn scenario. Ten runs are performed on each data set and the average results are reported.

4.2 Evaluation results of myth

RE: Fig. 3 shows the comparison of RE between Myth and Vivaldi. Table 4 shows the comparison of the average RE between them.

In stable scenario, the performance of Myth is almost equivalent to that of Vivaldi. Hardly any difference can be found between the two curves. In churn scenario, Myth outperforms Vivaldi a lot in both of the data sets. When t equals 10, Myth can reduce the average RE from Vivaldi by 37.21% in PlanetLab data set and by 27.50% in King data set. When t equals 20, Myth can reduce the average RE from Vivaldi by 47.62% in PlanetLab data set and by 36.00% in King data set.

4.3 Other metrics

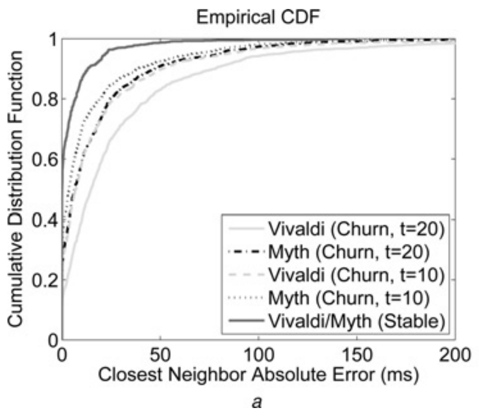
Besides RE described in Section 2, we also evaluate the performance of Myth with the following two metrics.

Closest neighbour absolute error (CNAE): Lua *et al.* [14] proposed closest neighbour loss (CNL), which indicates the probability to correctly select the closest neighbour to a given node, and is defined by the fraction of nodes where an incorrect node is chosen as the closest neighbour using predicted distances. In addition to the CNL, we measure the magnitude of the error when the wrong node is selected. More precisely, we use CNAE [27], which is defined as the gap between the distance to the incorrectly selected neighbour and the distance to the actual closest neighbour. Pietzuch *et al.* [17] uses nearest neighbour loss (NNL) to capture the application-observed latency penalty for using a node that is not the true nearest neighbour. Both of CNAE and NNL have the same definition but under different names. In this paper, we use CNAE for consistency.

Relative rank loss (RRL): Lua *et al.* [14] measures the probability to correctly select the closer node from an arbitrary node pair. It is defined as the percentage of incorrectly ordered node pairs (as perceived at a given node) based on the prediction. The value of RRL is between 0 (for no loss of relative order) and 1 (for a complete reversal of order).

Among these metrics, RE is the basic metric which is evaluated by all NC designers. RRL and CNAE focus more on application perspective where nodes only need to know the relative distances of other nodes.

CNAE: Table 5 and Fig. 4 show the comparison of CNAE between Myth and Vivaldi. In stable scenario, the CNAEs of



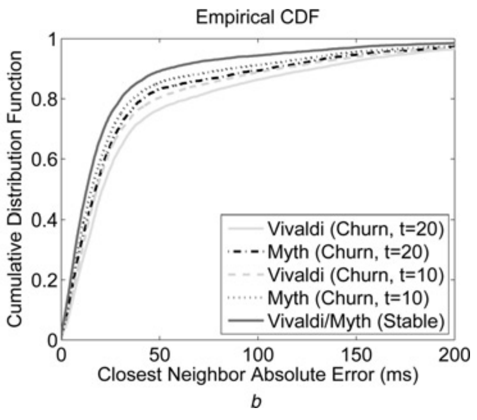


Figure 4 *Distribution of CNAE*

a Planet Lab
b King

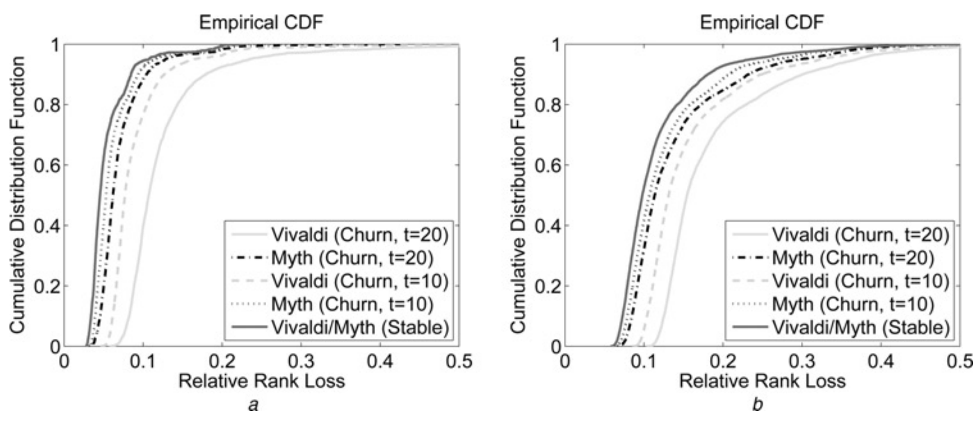


Figure 5 *Distribution of RRL*

a Planet Lab
b King

Myth and Vivaldi are also almost equivalent to each other. In churn scenario, in both PlanetLab data set and King data set, Myth improves the quality of the closest neighbour selection a lot comparing to Vivaldi. When t equals 10, Myth can reduce the average CNAE from Vivaldi by 29.53% in PlanetLab data set and by 15.40% in King data set. When t equals 20, Myth can reduce the average CNAE from Vivaldi by 42.62% in PlanetLab data set and by 18.45% in King data set.

RRL: Table 6 and Fig. 5 show the comparison of RRL between Myth and Vivaldi. Just as the results in RE and CNAE metrics, the RRL of Myth and Vivaldi in stable scenario are almost equivalent both PlanetLab data set and King data set. Myth outperforms Vivaldi with both PlanetLab data set and King data set in churn scenario. When t equals 10, Myth can reduce the average RRL from Vivaldi by 33.33% in PlanetLab data set and by 18.75% in King data set. When t equals 20, Myth can reduce the average RRL from Vivaldi by 46.15% in PlanetLab data set and by 26.31% in King data set.

4.4 Evaluation of measurement overhead

In this section, we study the measurement overhead of Vivaldi and Myth under node churn. Our simulation study the total number of messages used during 10 000 s. Let us consider the results of PlanetLab data set first, from Table 7, the total number of probes of Myth when t equals 20 is 43.01% smaller than the total number of probes of Vivaldi when t equals 10. At the same time, as demonstrated in Tables 4–6, Myth when t equals 20 performs better than Vivaldi when t equals 10 in all the metrics we used (RE, RRL, CNAE).

Table 7 Total number of probes in 10,000 s

Data set		Churn ($t = 10$)	Churn ($t = 20$)
PlanetLab	Vivaldi	1.86×10^5	0.93×10^5
	Myth	1.98×10^5	1.06×10^5
King	Vivaldi	1.43×10^6	0.72×10^6
	Myth	1.52×10^6	0.81×10^6

In other words, to improve the performance of Vivaldi under node churn, using Myth is a cost-effective way instead of simply reducing the t. It can not only improve the prediction accuracy but also save the measurement overhead. The same conclusion can be found using the results of King data set.

5 Conclusion

In this paper we study the node churn problem in a representative distributed NC system, Vivaldi. An accurate and decentralised NC system called Myth is proposed to improve the prediction accuracy under node churn.

The major contribution of this paper is twofold. (a) By evaluating the performance of Vivaldi under node churn, we find out that increasing the probe frequency of Vivaldi is a direct solution to reduce the effect of the node churn but the communication overhead is enlarged. As a simple approach, it is not a cost-effective solution to node churn. (b) We propose Myth, a fully decentralised NC scheme, to improve the accuracy of Internet distance prediction under node churn. Myth is both accurate and scalable under high node churn rate, combining merits of LBA and SBA together with moderate extra overhead. We evaluate the performance of Myth and compare it with the Vivaldi system with real Internet measurement traces. The experimental results show that Myth can remedy the node churn problem of Vivaldi. In our experiment, the performance of Myth exceeds that of Vivaldi a lot in churn scenario, and is almost equivalent to the performance of Vivaldi in the scenario where all nodes stay stable. In other words, Myth is robust to node churn. We have also demonstrated that using Myth is a more efficient way to improve the prediction accuracy under node churn instead of increasing each node's measurement frequency of Vivaldi; it can not only improve the prediction accuracy but also save the communication overhead.

To further evaluate the practicality of Myth, we currently focus on the deployment of Myth on Internet and develop some real applications based on this NC system. Also, we will involve Embeddable Overlay Networks (EON) [24] in our future research on NC to create a high quality NC system.

6 Acknowledgment

This work is supported by the National Basic Research Program of China (No. 2007CB310806) and the National Science Foundation of China (No. 60473087, No. 60703052, No. 60850003). Thanks to Prof. Pei Cao from Stanford University for her comments and suggestions. This paper was presented in part at the the 15th IEEE International Conference on Networks, Adelaide, Australia, November 2007 [28].

7 References

[1] ZHANG R.M., TANG C.Q., HU Y.C., FAHMY S., LIN X.: 'Impact of the inaccuracy of distance prediction algorithms on internet applications: an analytical and comparative study'. Proc. IEEE INFOCOM, May 2006

[2] PIETZUCH P., LEDLIE J.: 'Network-aware operator placement for stream-processing systems'. Proc. ICDE, 2006

[3] Azureus BitTorrent. http://azureus.sourceforge.net/

[4] ABRAHAM I., MALKHI D.: 'Compact routing on euclidian metrics'. Proc. PODC, 2004

[5] WANG G., CHEN Y., SHI L., *ET AL.*: 'Proxima: towards lightweight and flexible anycast service'. Proc. IEEE INFOCOM Student Workshop, April 2009

[6] NG T.S.E., ZHANG H.: 'Predicting Internet network distance with coordinates-based approaches'. Proc. INFOCOM, June 2002

[7] PIAS M., CROWCROFT J., WILBUR S., HARRIS T., BHATTI S.: 'Lighthouses for scalable distributed location'. Proc. IPTPS, February 2003

[8] TANG L.Y., CROVELLA M.: 'Virtual landmarks for the internet'. Proc. ACM IMC, 2003

[9] DABEK F., COX R., KAASHOEK F., MORRIS R.: 'Vivaldi: a decentralized network coordinate system'. Proc. ACM SIGCOMM, August 2004

[10] SHAVITT Y., TANKEL T.: 'Big-bang simulation for embedding network distances in euclidean space'. Proc. IEEE INFOCOM, April 2003

[11] GODFREY B., SHENKER S., STOICA I.: 'Minimizing churn in distributed systems'. Proc. ACM SIGCOMM, September 2006

[12] LEDLIE J., GARDNER P., SELTZER M.: 'Network coordinates in the wild'. Proc. USENIX NSDI'07, April 2007

[13] MAO Y., SAUL L., SMITH J.M.: 'IDES: an internet distance estimation service for large networks', *IEEE J. Sel. Areas Commun. (JSAC), (special issue on Sampling the Internet, Techniques and Applications)*, 2006, **24**, (12), pp. 2273–2284

[14] LUA E.K., GRIFFIN T., PIAS M., ZHENG H., CROWCROFT J.: 'On the accuracy of embeddings for internet coordinate systems'. Proc. IMC, October 2005

[15] ZHANG R., HU Y.C., LIN X., FAHMY S.: 'A hierarchical approach to internet distance prediction'. Proc. ICDCS, 2006

[16] TANG L.Y., CROVELLA M.: 'Geometric exploration of the landmark selection problem'. Proc. PAM, 2004

[17] PIETZUCH P., LEDLIE J., MITZENMACHER M., SELTZER M.: 'Network-aware overlays with network coordinates'. Proc. IWDDS, July 2006

[18] KAAFAR M.A., MATHY L., TURLETTI T., DABBOUS W.: 'Virtual networks under attack: disrupting internet coordinate systems'. Proc. ACM CoNext, December 2006

[19] KAAFAR M.A., MATHY L., BARAKAT C., SALAMATIAN K., TURLETTI T., DABBOUS W.: 'Securing internet coordinate embedding systems'. Proc. ACM SIGCOMM, August 2007

[20] RHEA S., GEELS D., ROSCOE T., KUBIATOWICZ J.: 'Handling churn in a DHT'. Proc. USENIX Annual Technical Conf., June 2004

[21] WANG F., XIONG Y.Q., LIU J.C.: 'mTreebone: a hybrid tree/mesh overlay for application-layer live video multicast'. Proc. IEEE ICDCS, June 2007

[22] RHEA S., GODFREY B., KARP B., *ET AL.*: 'OpenDHT: a public DHT service and its uses'. Proc. ACM SIGCOMM 2005, August 2005

[23] LUA E.K.: 'The structure of internet latency', PhD thesis, University of Cambridge, 2006

[24] LUA E.K., GRIFFIN T.G.: 'Embeddable overlay networks'. Proc. 12th IEEE Symp. Computers and Communications (ISCC) 2007, Aveiro, Portugal, 1–4 July 2007

[25] NC Research Group at Harvard. http://www.eecs.harvard.edu/syrah/nc/, accessed November 2008

[26] RHEA S., GEELS D., ROSCOE T., KUBIATOWICZ J.: 'Handling churn in a DHT'. U. C. Berkeley Technical Report UCB/CSD-03-1299, December 2003

[27] ZHANG R., HU Y.C., LIN X., FAHMY S.: 'A hierarchical approach to internet distance prediction'. Technical Report TR ECE 06-03, Purdue University, March 2006

[28] CHEN Y., ZHAO G.Y., LI A., DENG B.X., LI X.: 'Myth: an accurate and scalable network coordinate system under high node churn rate'. Proc. 15th IEEE Int. Conf. Networks, Adelaide, Australia, November 2007