TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 12, No. 2, February 2014, pp. 1536 $\sim$ 1543
DOI: http://dx.doi.org/10.11591/telkomnika.v12.i2.3817

Distributed Regional Weighted Multidimensional Scaling for Network Coordinate System

Wang Cong*, Zhang Fengli , and Li Min
School of Computer Science and Engineering, University of Electronic Science and Technology of China
Sichuan province, Chengdu city, China
*corresponding author, e-mail: wongcong@gmail.com

Abstract

Network Coordinate System embeds all nodes into a specific metric space and estimate the latency between any node-pair through computing the corresponding metric distance with low-cost. But the metric space is usually uniform while the Internet is heterogeneous since different autonomous systems (AS) always have different routing strategies. To make full use of the clustering characteristic for improving the latency estimating accuracy without any prior knowledge, this paper judges the reference nodes are in the same AS with the target node or not through the metric distance between the target node and the source, and weights them by the Gaussian kernel function. Then the source node can computes a new coordinate to estimates the latency between the target node and itself more precisely. The experiments show this mechanism can reduce the $Stress$ criterion of the entire system by 19.38% without any additional communication overhead, its computation cost is also very small at the same time.

Keywords: *Network Coordinate System, Multidimentional Scaling, Optimization.*

1. Introduction

Many Internet applications, for instances, content distribution networks[1, 2], electronic sports[3] and cloud computing[4], are all latency-sensitive. For those kinds of applications, a very important issue is how to find the closest service providers. The traditional nearest server selection method requires nodes ping or traceroute each other. But this simple method has pool scalability since the communication cost of measurement is $O(n^2)$, when the network scale is large enough, node can hardly update its latency information in real time.To reduce the communication overhead with an acceptable estimation accuracy loss, a new kind of latency estimation model named Network Coordinate System was presented in recent years[5]. NCS defines a metric space, and assigns each node a coordinate as its virtual location in the space through measuring the latencies between a small set of reference nodes and itself. Thus NCS can estimate the latency between any two nodes by computing their metric distance instead of measuring it directly. The measurement communication overhead of NCS is only $O(n)$.

This paper presents a novel distributed NCS enhancing algorithm named drwMDS for estimating latency between any two nodes. Unlike the dwMDS algorithm which was used in Wireless Sensor Networks[6, 7], our method weights each reference node by its metric distance to the target node rather than the real latency to the source node, and through which it refines the coordinate of the source node, so the latency between the source and the target node can be estimated by the refined coordinate with higher accuracy.

2. The Proposed Method

As mentioned above, all the NCSs estimate the latency between two nodes by their corresponding coordinates. By embedding N nodes into a metric space and associating each node i with a D-dimensional vector $C_i = [\ c_{i,1} \quad c_{i,2} \quad \cdots \quad c_{i,D}\ ]$ as its coordinate, we can represent

Received July 3, 2013; Revised September 5, 2013; Accepted October 2, 2013

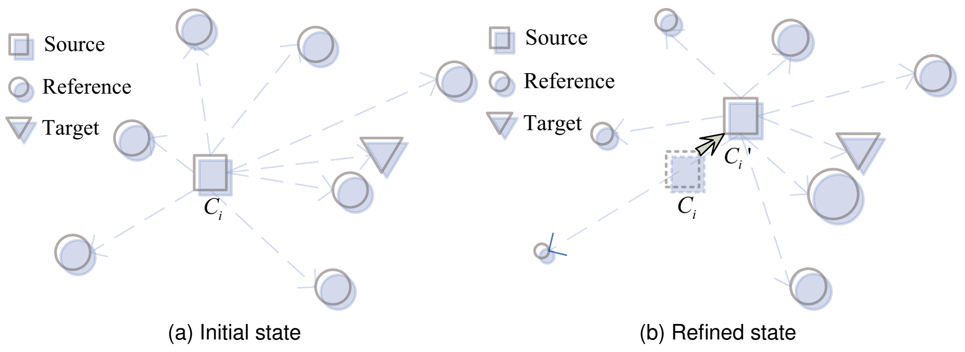

Figure 1. The refinement process of drwMDS.

the metric distance between node i and j as:

$$\hat{d}_{i,j} = \|C_i - C_j\| \tag{1}$$

The $\|\cdot\|$ operation computes the norm which is defined in the metric space. In generally, it represents the L_2 norm while the Euclid space is defined as the metric space. The latency between node i and j is denoted as $d_{i,j}$, through which we can build a Mean Square Error function e:

$$e = \sum_{i=1}^{N}\sum_{j=1}^{N}\left(d_{i,j} - \hat{d}_{i,j}\right)^2 \tag{2}$$

How to minimize function e is the key issue of NCS construction. Various optimization algorithms are used to compute node's coordinate such as (stochastic) gradient descent algorithm[8] and SMACOF[9].

An important hypothesis of NCS is that nodes can be embedded into the metric space ideally. Taking the high nonlinearity of the Internet into consider, this hypothesis is difficult to be satisfied. [10, 11] proves that the Internet latency space is heterogeneous due to the weak linear latency-distance correlation in moderately and poorly connected Internet regions. According to this phenomenon, many hierarchical NCS models are presented[12, 13]. This kind of mechanism divides the whole metric space into multiple layers, for example, local layer and global layer, and maintains different coordinates in each layer, hence the intra-cluster and inter-cluster latencies can be estimated by local coordinates and global coordinates respectively. The hierarchical model can achieve street-level accuracy while the landmarks are dense enough[14]. But, to assign a node into a specific cluster, some prior knowledge about the Internet, either the Internet topology information[12] or a fixed set of landmarks with precise locations are necessary[13, 14], thus none of the previous work can work without any prior knowledge of Internet.

3. Research Method

3.1. The Basic Architecture of drwMDS

In this section, we discuss the full distributed architecture of drwMDS. As what has been discussed, selecting those nodes which are in the same cluster with the target as references can improve the estimation accuracy significantly. This paper evaluates how likely that node A is in the same cluster with node B based on their metric distance: the closer in the metric space they are, the higher the likelihood is, thus Internet topology information is not required in drwMDS. The basic idea of drwMDS is shown in Fig.2.

In our approach, the latency estimation process is splited into a sup and a sub phase. In the sup-phase, each node maintains a global coordinate and updates it periodically. The sub-phase is a refinement process. In this phase, a node, we refer it as node i, estimates the latency

$d_{i,j}$ between another node j and itself by weighting each reference node k with a non-negative value $w_{i,k}$ and calculating a new coordinate C'_i which satisfies:

$$C'_i = \arg\min_{C_i} e_i \tag{3}$$

e_i represents the local error function of node i, it is defined as:

$$e_i = \sum_{k \in R(i)} w_{i,k}(d_{i,k} - \|C_i - C_k\|_2)^2 \tag{4}$$

Where $R(i)$ is the references set of node i and $w_{i,k}$ is the weight of node k for node i which is generated by a specific algorithm. Then the latency $d_{i,j}$ can be estimated by $\|C'_i - C_j\|_2$. The whole process is shown in Algorithm 1.

Algorithm 1 The Architecture of drwMDS Algorithm.

```
Require: $d_{i,k}$: the round trip time from node $i$ to reference node $k$; $C_k$: the coordinate of node $k$;
    $j$: the target node for estimating; $C_j$: the coordinate of node $j$.
Ensure: $\hat{d}_{i,j}$: the estimation of $d_{i,j}$.
1: Updating $C_i$ periodically;
2: if $require\_estimation(d_{i,j})$ then
3:     for each $k$ in $R(i)$ do
4:         $w_{i,k} = Weighting(C_j, k, C_k)$; //Weighting reference node $k$ with a non-negative value;
5:     end for
6:     $C'_i = \arg\min_{C_i} e_i$; //Refining $C'_i$ by equation(3);
7:     $\hat{d}_{i,j} = \|C_i{}' - C_j\|_2$; //Estimating $d_{i,j}$;
8:     return $\hat{d}_{i,j}$;
9: end if
```

3.2. Global-layer construction

The global-layer construction algorithm embeds nodes into a uniform metric space. We build the global-layer in the sup-phase by the widely used Vivaldi algorithm for its simplicity and decentralization[8]. The whole process of Vivaldi is as Algorithm2:

Algorithm 2 Vivaldi: The Global-layer Construction Algorithm.

```
Require: $d_{i,j}$: the round trip time from node $i$ to node $j$; $C_j$: the coordinate of $j$; $e_j$: the local error
    of $j$; $c_c$: the iterative step factor.
Ensure: $C_i$: the coordinate of $i$;
1: while get $(d_{i,j}, C_j, e_j)$ do
2:     $v_{i,j} = e_i/(e_i + e_j)$;
3:     $e_s = abs(d_{i,j} - \|C_i - C_j\|_2)/d_{i,j}$;
4:     $e_i = e_s c_e v_{i,j} + e_i(1 - c_e v_{i,j})$;
5:     $C_i = C_i + c_c v_{i,j}\left(d_{i,j} - \|C_i - C_j\|_2\right)(C_i - C_j)/\|C_i - C_j\|_2$;
6:     return $C_i$;
7: end while
```

3.3. Iterative refinement process

In this section we discuss our iterative refinement algorithm of drwMDS which is based on a multidimensional scaling algorithm, SMACOF. Essentially, the refinement process attempts to generate a non-incremental iterative sequence to approximate a local optimum value of C'_i in equation(3). By denoting $C_i^{(t)}$ as the estimation of C_i in the t-th iterative time, and $e_i^{(t)}$ as the corresponding local error function, we define a temporary function $e_i^{(t)}(X)'$:

$$e_i^{(t)}(X)' = \sum_{k\in R(i)} w_{i,k}\left[d_{i,k}^2 - 2d_{i,k}\frac{(X-C_k)\left(C_i^{(t)}-C_k\right)^T}{\left\|C_i^{(t)}-C_k\right\|_2} + \|X-C_k\|_2^2\right] \tag{5}$$

Obviously there must be a vector $C_i^{(t+1)}$ which satisfies:

$$e_i^{(t)}\left(C_i^{(t+1)}\right)' \le e_i^{(t)}\left(C_i^{(t)}\right)' = e_i^{(t)} \tag{6}$$

According to the Cauchy-Schwarz inequality:

$$\sum_{a=1}^{m} p_a q_a \le \left(\sum_{a=1}^{m} p_a^2\right)^{1/2}\left(\sum_{a=1}^{m} q_a^2\right)^{1/2} \tag{7}$$

We can get that for each $k \in R(i)$ it has:

$$\left\|C_i^{(t+1)}-C_k\right\|_2 \ge \frac{\left(C_i^{(t+1)}-C_k\right)\left(C_i^{(t)}-C_k\right)^T}{\left\|C_i^{(t)}-C_k\right\|_2} \tag{8}$$

Equality of inequation(8) occurs if and only if $C_i^{(t)} = C_i^{(t+1)}$. Thus we can derive:

$$e_i^{(t+1)} \le e_i^{(t)}\left(C_i^{(t+1)}\right)' \tag{9}$$

By minimizing $e_i^{(t)}(X)'$, e.g. letting the Jacobian matrix of $e_i^{(t)}(X)'$ equal to the zero vector:

$$\frac{de_i^{(t)}(X)'}{dX} = \mathbf{0} \tag{10}$$

We can get the iterative function finally:

$$C_i^{(t+1)} = \frac{\sum\limits_{k\in R(i)} w_{i,k}\left[C_k + \frac{d_{i,k}\left(C_i^{(t)}-C_k\right)}{\left\|C_i^{(t)}-C_k\right\|_2}\right]}{\sum\limits_{k\in R(i)} w_{i,k}} \tag{11}$$

Algorithm 3 shows this refinement process in detail.

3.4. Weighting algorithm

In this paper, we weighted references by the Gaussian kernel function since its domain of definition is $(-\infty, +\infty)$ so it can weight those references which maybe far away from the target node. We make the target node j be the centroid of the kernel, the weight of reference node k $w_{*,k}$ can be calculated by the Gaussian kernel weighting function:

$$w_{*,k} = \exp\left[-\frac{(C_k-C_j)(C_k-C_j)^T}{2\sigma_{*,j}^2}\right] \tag{12}$$

Where $\sigma_{*,j}$ represents the bandwidth of the Gaussian kernel. A constant kernel bandwidth factor can not fit all the nodes because of the inhomogeneity of the Internet host distribution. Thus we compute the kernel bandwidth adaptively by importing an adjustment factor λ:

$$\sigma_{i,j} = \lambda\frac{\sum\limits_{k\in R(i)} \hat{d}_{j,k}}{|R(i)|} \tag{13}$$

Obviously, the drwMDS will degrade into Hariri algorithm[9] while $\lambda \to +\infty$.

Algorithm 3 The Refinement Algorithm.

```
Require: $R(i)$: the reference set of node $i$; $d_{i,k}$: the round trip time from node $i$ to each reference
    node $k$; $w_{i,k}$: the weight value of node $k$; $C_k$: the coordinate of node $k$; $C_i$: the global-layer
    coordinate of node $i$; $\epsilon$: the terminal condition of iteration; $t_{max}$: the maximum iterative times.
Ensure: $C_i'$: the refined coordinate of $i$.
 1: $t = 0$;
 2: ${C_i}^{(t)} = C_i$;
 3: while $\left\|{C_i}' - C_i\right\|_2 > \epsilon$ do
 4:     $t = t + 1$;
 5:     $Calculating\ {C_i}^{(t)}\ by\ equation(11)$;
 6:     $C_i' = {C_i}^{(t)}$;
 7:     if $t > t_{max}$ then
 8:         break;
 9:     end if
10: end while
```

4. Results and Discussion

To evaluate the accuracy and convergence of the proposed approach, we compare the dr-wMDS with the Vivaldi algorithm. All the experiments are performed on the PlanetLab dataset[15] which collect the real latencies among 226 PlanetLab nodes. We define the 2-D Euclid space as our metric space, and set the default size of the reference set equal to 60. We also set the convergence threshold ϵ equal to 0.01 ms for all the experiments.

4.1. Performance evaluation coefficients

At first we introduce a series of evaluation criteria including Stress, Relative Error and Relative Rank Loss to evaluate the estimation accuracy.

4.1.1. Stress:

Summing all the mean square error between node pairs yields a badness-of-fit coefficient Stress for the entire NCS representation:

$$Stress = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(d_{i,j} - \hat{d}_{i,j}\right)^2}{\sum_{i=1}^{N}\sum_{j=1}^{N} d_{i,j}^2}} \tag{14}$$

The scaling factor $\sum_{i=1}^{N}\sum_{j=1}^{N} d_{i,j}^2$ removes the network scale dependency of Stress.

4.1.2. Relative Error:

The Relative Error $RE_{i,j}$ between node i and j is defined as:

$$RE_{i,j} = abs\left(\frac{d_{i,j} - \hat{d}_{i,j}}{d_{i,j}}\right) \times 100\% \tag{15}$$

A derived indicator is the Ninetieth Percentile Relative Error. $NPRE$ guarantees 90% of the latency estimations have lower RE values than it.

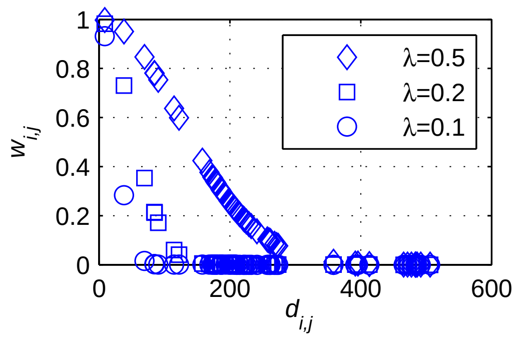

Figure 2. The weights of references

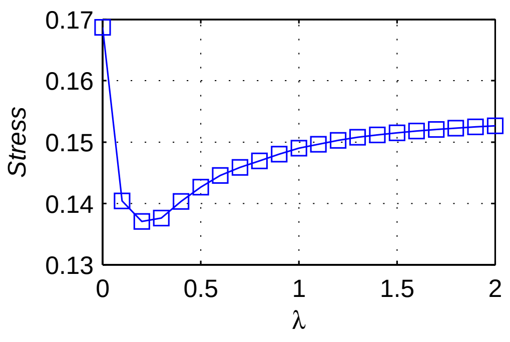

Figure 3. The influence of λ on $Stress$

4.1.3. Relative Rank Loss:

It is important to keep the relative ranking of distances since many applications need to order other nodes by their latency to the host. Thus we define the Relative Rank Loss RRL_i for node i to evaluate the rank disordering:

$$RRL_i = \frac{\left|\left\{(j,k)\left|\forall j,k : d_{i,j} > d_{i,k}\ and\ \hat{d}_{i,j} \leq \hat{d}_{i,k}\right.\right\}\right|}{\left|\left\{(j,k)\left|\forall j,k\right.\right\}\right|} \times 100\% \tag{16}$$

Be similar with $NPRE$, we define another derived indicator Ninetieth Percentile Relative Rank Loss ($NPRRL$) which guarantees 90% of the nodes have lower relative rank loss than it.

4.2. Impact of Parameters

4.2.1. The bandwidth adjustment factor λ

At first we describe the impact of the kernel's bandwidth. From equation(13) we can see that there is a strict linear relationship between adjustment factor λ and bandwidth σ.

As shown in Fig.3, an excessive λ usually leads to a too wide bandwidth, so that some references which are not in the same cluster with the target node will be weighted with high values. If λ is too small, the bandwidth will turn to be so narrow that only a few references can obtain appropriate weights while others can hardly affect the calculation because of their negligible weights. As a result, the influence of the measurement noise will be amplified due to the inadequacy of references.

Fig.4 shows the influence of λ. From this figure we can see that the value of λ affects the estimation accuracy significantly. The highest accuracy occurs while the value range of λ is [0.2,0.3]: the $Stress$ is equal to 0.1370 and 0.1376 while we set λ to be 0.2 and 0.3 respectively, so we suggest 0.2 is an appropriate value of λ. Fig.4 also shows the negative influence of an inappropriate λ value. Both insufficient and excessive value of λ lead to the decrease of accuracy: the $Stress$ is 0.1404 while we set λ to be 0.1, it will also grow up to 0.1527 gradually along with the increasing of λ up to 2.

4.2.2. The maximum iterative times t_{max}

There is a strong relationship between t_{max} and the convergence speed. A smaller t_{max} leads to an acceleration on convergence speed along with some accuracy loss and vice versa.

Fig.5 shows the cumulative distribution function (CDF) of the iterative times while ϵ=0.01 without t_{max} setting. We can see that drwMDS converges slowly: more than 50% calculation processes experience no less than 170 iterative times. Accordingly, the mean value of iterative times is equal to 464.7. To reduce the calculating load, an appropriate t_{max} is necessary.

We also plot the relationship between t_{max} and $Stress$ in Fig.6. From this figure we can see clearly that $Stress$ drops by 19.38% from 0.1687 to 0.1360 only after 5 iterative steps.

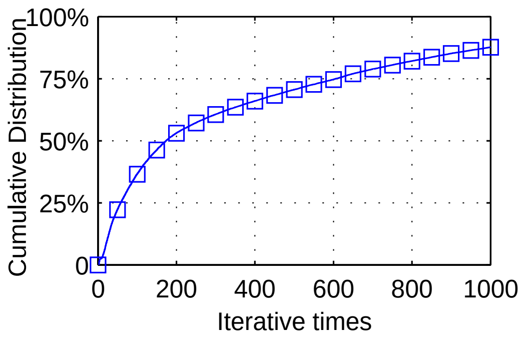

Figure 4. The CDF of the iterative times

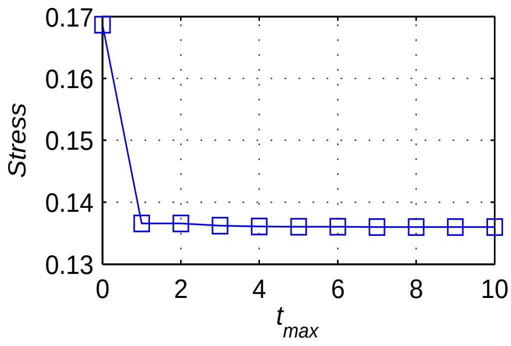

Figure 5. The influence of t_{max} on $Stress$

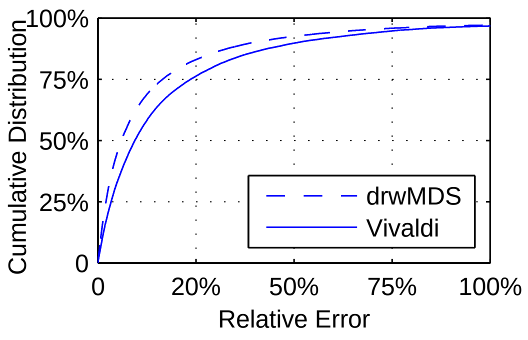

Figure 6. The CDF of RE

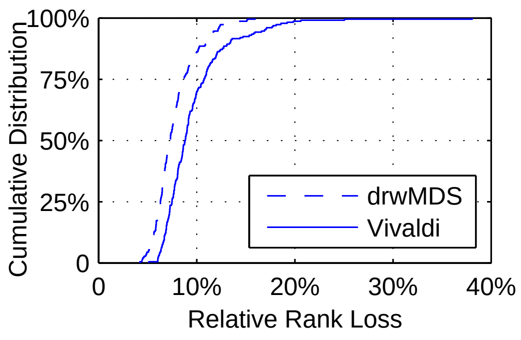

Figure 7. The CDF of RRL

Furthermore, no meaningful improvement for $Stress$ is found while t_{max} increases from 5 to 10. It indicates that 5 is an appropriate value of t_{max}. Through setting t_{max} equal to 5 we reduce the average iterative times fall by 98.93% to 4.96.

4.3. Performance analysis

We evaluate the performance of our approach by RE and RRL and compare it with the Vivaldi algorithm. As discussed above, we set λ=0.2, ϵ=0.01 and t_{max}=5. By this time, the $Stress$ of drwMDS is 0.1360 which is far better than that of Vivaldi.

For the entire network, the cumulative distribution of RE is shown in Fig.7. It is clearly that drwMDS can improve the estimation accuracy significantly. In more detail, there is 78.82% latency estimations in drwMDS have no more than 20% relative error which is 7.94% higher than Vivaldi; Similarly, the $NPRE$ value of drwMDS is only 39.64% while that of Vivaldi is up to 51.05%.

Now we examine the RRL of the entire network. Fig.8 indicates that drwMDS can keep the latency rank far more precisely than Vivaldi. In practical terms, there are 194 nodes, e.g. 85.84% of all the nodes which RRL values are less than 10% in drwMDS, while the corresponding node number of Vivaldi is only 158, the percentage of these nodes is only 69.91%. Furthermore, the $NPRRL$ of drwMDS is only 10.95%, which is far less than Vivaldi's 13.34%.

5. Conclusion

In this paper, we propose a novel latency estimation mechanism called drwMDS for P2P systems and other distributed latency-sensitive applications. In drwMDS, the reference nodes is weighted by the metric distance to a specific target node rather than the latency to the source node, and through which we can refine the coordinate of the source node for the target node and achieve higher estimation accuracy. Comparing with previous works,the main advantage of drwMDS is, it can model the clusting characteristic of the Internet without any additional measurement without any prior knowledge of the topology information of Internet for the first time. Our method is also fully distributed, its computational cost can be almost ignored, too. Experimental results show that our approach can improve the latency estimation accuracy significantly.

Acknowledgement

This work was supported by The National Key Scientific and Technological Special Project of China (2011ZX03002-002-03) and the Key Project on the Integration of Industry, Education and Research of Guangdong Province (Grant No. 2012B091000054)

References

[1] M. Szymaniak, D. Presotto, G. Pierre, and M. van Steen, “Practical large-scale latency estimation,” *Computer Networks*, vol. 52, no. 7, pp. 1343–1364, 2008.

[2] J. Ledlie, P. Gardner, and M. Seltzer, “Network coordinates in the wild,” in *Proceedings of the 4th NSDI*, vol. 7, 2007, pp. 299–311.

[3] G. Armitage and A. Heyde, “Reed: Optimizing first person shooter game server discovery using network coordinates,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 8, no. 2, p. 20, 2012.

[4] L. Wei, Z. Jian, W. Chunzhi, and X. Hui, “Kalman filter localization algorithm based on sdstwr ranging,” *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 3, pp. 1436–1448, 2013.

[5] Z.-F. Wang, M. Chen, C.-Y. Xing, B. Li, and X.-F. Qiu, “Research on the modeling of the Internet delay space,” *Tongxin Xuebao/Journal on Communications*, vol. 33, no. 7, pp. 164–176, 2012.

[6] J. A. Costa, N. Patwari, and A. O. Hero III, “Distributed weighted-multidimensional scaling for node localization in sensor networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 1, pp. 39–64, 2006.

[7] W. Jiang, J. Zhang, J. Li, and H. Hu, “An improved resource query and location algorithm based on cloud computing,” *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 10, pp. 6166–6172, 2013.

[8] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, “Vivaldi: A decentralized network coordinate system,” in *Proceedings of ACM SIGCOMM 2004*, vol. 34, no. 4. ACM, 2004, pp. 15–26.

[9] N. Hariri, B. Hariri, and S. Shirmohammadi, “A Distributed Measurement Scheme for Internet Latency Estimation,” *Instrumentation and Measurement, IEEE Transactions on*, vol. 60, no. 5, pp. 1594–1603, 2011.

[10] D. Li, J. Chen, C. Guo, Y. Liu, J. Zhang, Z. Zhang, and Y. Zhang, “Ip-geolocation mapping for moderately connected internet regions,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 2, pp. 381 –391, 2013.

[11] Z. feng Wang, M. Chen, C. you Xing, J. Feng, X. lin Wei, and H. li Bai, “Multi-manifold model of the internet delay space,” *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 211 – 218, 2013.

[12] Z. Chen, Y. Chen, Y. Zhu, C. Ding, B. Deng, and X. Li, “Tarantula: Towards an Accurate Network Coordinate System by Handling Major Portion of TIVs,” in *Proceedings of GLOBECOM 2011, 2011 IEEE*. IEEE, 2011, pp. 1–6.

[13] C.-Y. Xing and M. Chen, “A hierarchical network distance prediction mechanism,” *Jisuanji Xuebao/Chinese Journal of Computers*, vol. 33, no. 2, pp. 356–364, 2010.

[14] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang, “Towards street-level client-independent IP geolocation,” in *Proceedings of the 8th NSDI*, 2011, pp. 27–36.

[15] “PlanetLab Data Set.” [Online]. Available: http://www.eecs.harvard.edu/ syrah/nc/sim/pings.4hr.stamp.gz