Charm: A charming network coordinate system

Çağatay Demiralp

December 5, 2007

Abstract

We present a new network coordinate system, Charm, embedding round-trip time latencies into the Euclidean space. Charm is based on a particle system model with two spatial force interactions: attraction and repulsion. The update algorithm (Charming) that Charm uses is simple and lightweight: For each latency sample, the algorithm either brings the communicating nodes closer (attraction) or moves them farther (repulsion). We evaluate Charm by comparing with Vivaldi algorithm using simulations. Preliminary results suggests that Charm is fast and stable, and has a good converging rate.

1 Introduction

Methods that allow hosts to predict round-trip times (RTT) to other hosts in a distributed network can be useful in reducing communication overhead and increasing the utility of the network resources (i.e., bandwidth). There have been several schemes proposed for embedding the measured latencies into a metric space [1] in order to create synthetic coordinates. Unfortunately most of these schemes do not produce stable and accurate coordinates under live network conditions. There are two reasons for this under-performance. First, RTT measurements are prone to random noise which makes modeling the variations between among RTT measurements very difficult. The second reason is more fundamental: While most of the existing network coordinate (NC) systems rely on metric embedding techniques, RTT measurements may not satisfy triangular inequality due to routing policies. This situation exacerbates with the presence of random noise. One of the approaches proposed to mitigate these problems is to use filters on the histories of measurements and coordinates [5, 3].

Charm should be seen as an attempt to develop a simple and stable NC system. Our algorithm considers two basic force interactions between hosts. At each latency sample, the algorithm either attracts the communicating nodes to each other or repels them from each other. In the following sections, we first give an overview of related work (section 2) and move on to discussing the

[1] A metric space is a pair of a point set X and a distance function d defined over X such that for all $x_i, x_j, x_k \in X$, $d(x_i, x_j) = d(x_j, x_i)$ (symmetry), $d(x_i, x_j) = 0 \leftrightarrow x_i = x_j$ (positive definiteness) and $d(x_i, x_k) \leq d(x_i, x_j) + d(x_j, x_k)$ (triangle inequality), are all true.

details of Charm system (section 3). We report the results from our simulations in section 4. And finally, we offer our conclusions and point some future work directions in section 5.

2 Previous Work

Our method can be considered a simulation-based technique in a sense that we assume a model of a physical system and its equilibrium as the energy minimizing state. There have been several such methods proposed. For example, Vivaldi [2] is based on a simulation of springs, which we use as the baseline algorithm in our experiments to evaluate Charm. Similarly, Big-Bang Simulation [9] uses particle explosion interactions to determine network coordinates. At the other end of the spectrum of NC research, there are landmark-based systems. In landmark-based systems, distance between two nodes are computed indirectly via predetermined, fixed nodes. For example, GNP determines the coordinates of a node by measuring its latency to a fixed set of landmarks and then solving a nonlinear minimization problem iteratively [7]. To address some of the computational issues (i.e, being expensive) related to GNP, Virtual Landmarks [6] and ICS [10] propose using PCA for linear approximation to the minimization problem. One of the potential drawbacks of using fixed landmarks is that it is not clear what happens in case of failure of these fixed nodes. In addition, there is a need to optimize the number and spatial configuration of the landmarks to avoid unbalanced workload. Concerned with these issues, PCoord [11] and PIC [1] use a hybrid approach, where landmarks are used only for bootstrapping; After bootstrapping, nodes adjust their coordinates with respect to the coordinates of peers. In a somewhat different approach, Lighthouse [8] assumes that shape of the network is a manifold and uses local coordinates on locally flat patches to create a global coordinate system. Not all NC systems use RTT measurements as their "distance metric". For example, landmark-based IDMaps [4] use dissimilarities between the IP addresses of the hosts as the metric to be predicted. Note that all the methods we discussed here embed nodes into a Euclidean coordinate space.

3 Charm

Charm is an NC coordinate system where nodes are modeled as particles with spatial force interactions: attraction and repulsion. Charm uses the same single algorithm, Charming, for continuously updating node coordinates both centralized and distributed settings. The algorithm corrects a constant fraction of the error between the measured and the current distance by either repelling or attracting the nodes. Algorithm 1 shows Charming algorithm which takes new latency measurement ℓ_{ij} and the node j coordinates x_j as parameters and computes the new x_i and x_j. There is only one constant (a correction fraction) c, which we set to 0.1 for all the experiments that we show results for.

Algorithm 1

CHARMING(ℓ_{ij}, x_j)

1. $e = \ell_{ij} - ||x_i - x_j||_2$ {difference between the current and measured distance}
2. $d = \frac{1}{2} \times e \times c$ {length of repulsion/attraction}
3. $x_i = x_i + d \times \frac{x_j - x_i}{||x_j - x_i||_2}$ {repel/attract x_i}
4. $x_j = x_j + d \times \frac{x_i - x_j}{||x_i - x_j||_2}$ {repel/attract x_j}

Since we compare Charming algorithm to both centralized and adaptive Vivaldi algorithms we give them here for completeness (see Algorithms 3 and 2). Further details for these algorithms can be found in [2]. The main difference between Charming and Vivaldi is that Charming adjusts the node coordinates bi-directionally. Note that mass-spring model used by Vivaldi is a particle system as well. We set the adaptive Vivaldi constants c_e and c_c to 0.1 and 0.25 throughout our comparisons.

Algorithm 2

CENTRALIZED_VIVALDI(ℓ, x)

1. **while** $error(\ell, x) > \epsilon$ **do**
2. **for all** i **do**
3. $F = \vec{0}$
4. **for all** j **do**
5. $e = \ell_{ij} - ||x_i - x_j||_2$
6. $F = F + e \times \frac{x_i - x_j}{||x_i - x_j||_2}$
7. $x_i = x_i + t \times \frac{F}{||F||_2}$
8. **end for**
9. **end for**
10. **end while**

Algorithm 3

VIVALDI(ℓ_{ij}, x_i, x_j, e_j)

1. $w = \frac{e_i}{e_i + e_j}$
2. $e_s = \frac{|||x_i - x_j||_2 - \ell_{ij}}{\ell_{ij}}$
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. $x_i = x_i + \delta \times (\ell_{ij} - ||x_i - x_j||_2) \times \frac{x_j - x_i}{||x_j - x_i||_2}$ {adjust x_i}

4 Experiments

We evaluate the performance of Charming using about 300 nodes with RTT measurements sampled on a 2D regular grid. The spatial configuration of the nodes according to the latencies is shown in Figure 2a (a 3D view is given for better view of the layout). Initial values for coordinates are determined by sampling from a multivariate normal distribution ($X \sim N(\mu, \Sigma)$ where $\mu = [0 \quad 0]^T$ and $\Sigma = diag([1 \quad 1]^T)$. In our experiments, we consider three cases: centralized embedding, new node accommodation, and distributed embedding. For each of the three cases Charming performs the best.

Figure 1 shows the evolution of the nodes for the centralized and adaptive Vivaldi (where δ constant changes adaptively), and Charming algorithms for a centralized coordinate update scenario. Figure 2b show the change of mean square error for each algorithm.

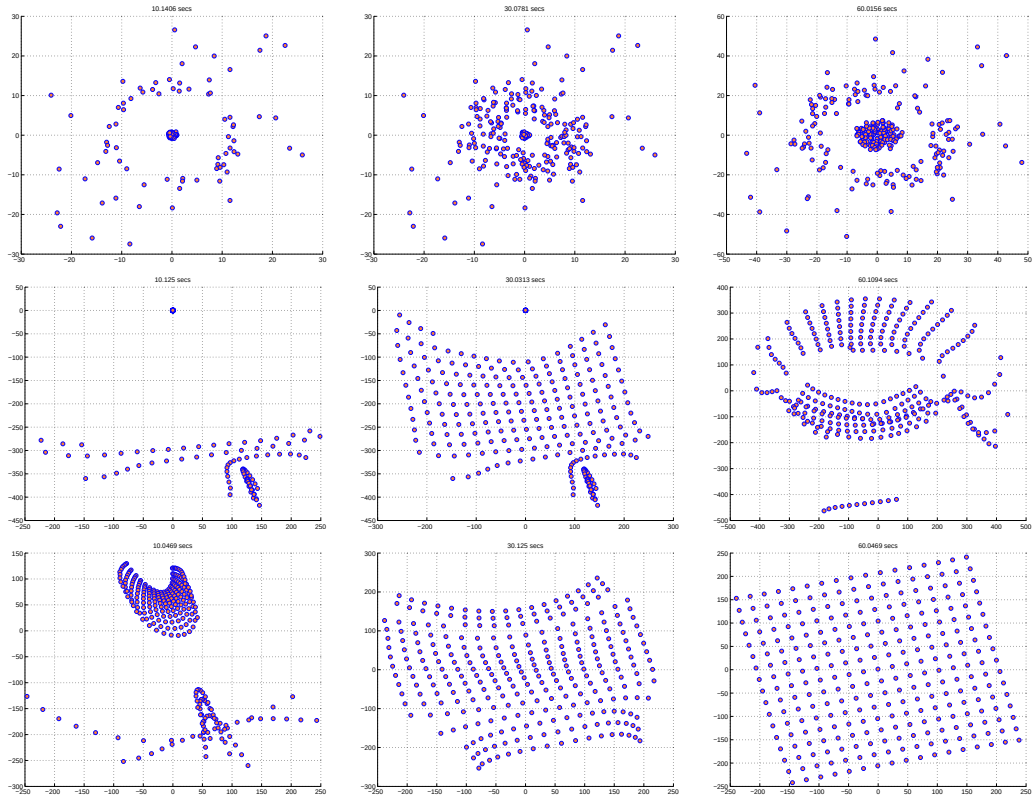
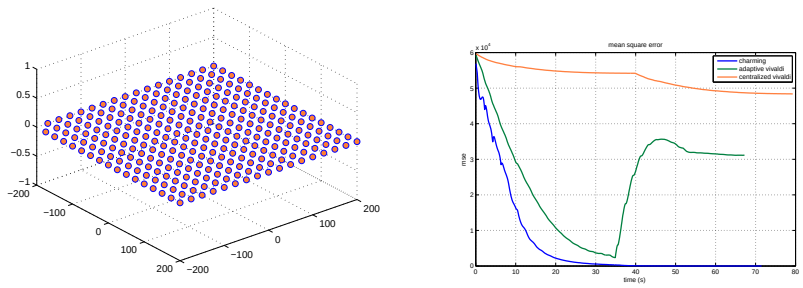


Figure 1: Evolution of the nodes (for the toy network layout see Figure 2a when centralized Vivaldi (1st row), adaptive Vivaldi (2nd row) and Charming (3rd row) algorithms are used in a centralized fashion. See Figure 2b for convergence and mean square error rates.



(a) (b)

Figure 2: (a) shows the layout of our toy network where latencies (RTTs) are sampled on a regular grid for simulation purposes. There are about 300 nodes in the network. (b) Mean square error values from the simulation (shown in Figure 1 comparing the centralized Vivaldi, adaptive Vivaldi, and Charming algorithms for a centralized NC computation. Notice the oscillatory behavior of adaptive Vivaldi algorithm.

In Figure 4, we compare how well the adaptive Vivaldi and Charming algorithms accommodate the new nodes entering the network. The orange-colored nodes in Figure 4a represent the new nodes (coordinates of which are initialized from $N(\mu, \Sigma)$) entering to a network with stabilized coordinates (shown as blue-colored nodes in the same figure). As seen Figures 4and 4b, Charming converges and stabilizes quick in this case as well. As the last case, we compare Charming and adaptive Vivaldi in a simulation where nodes measure latencies to every other node randomly (see Figure 4). While Charming still performs significantly better than Vivaldi, Vivaldi performed better than it did in the previous settings. This suggests that oscillations in Vivaldi could be due to localized measurements.

5 Conclusions and Future Work

We presented Charm, a new network coordinate system using spatial force interactions of attraction and repulsion to embed measured latencies into a Euclidean space. The initial results show that the update algorithm (Charming) of this system converges faster and it is more stable than Vivaldi algorithm. However, without further evaluation on real and simulated network environments it would be premature to claim that these would scale to large networks and real world domains. In this study, we confirm that Vivaldi is very sensitive to parameter setting, initial conditions and prone to oscillations. The fact that adaptive Vivaldi worked better with random measurements from the complete set of the nodes could be important. This might suggest that oscillations in Vivaldi could be due to localized measurements, which is the most likely situation, for example, in the Internet. Combining Vivaldi with a landmark-based system (i.e.,

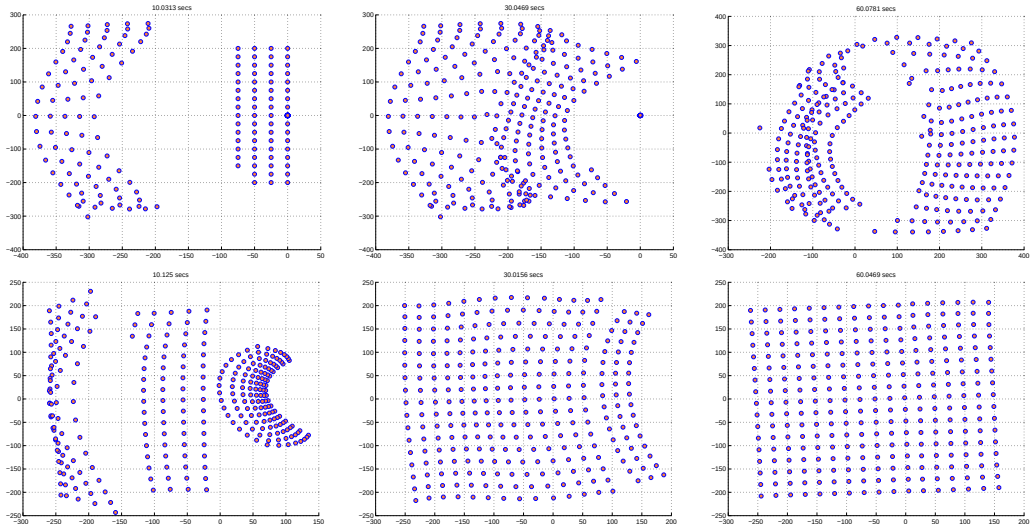


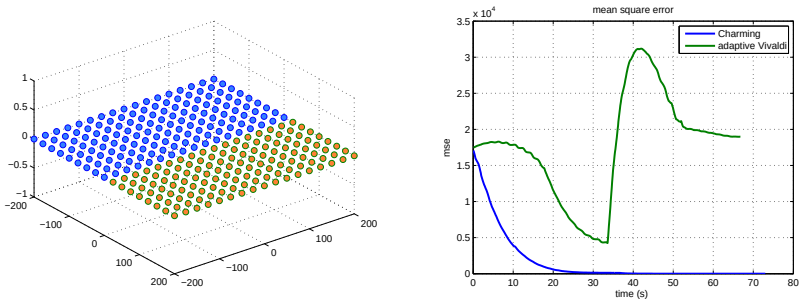
Figure 3: Evolution of nodes after new nodes enter the system (for the configuration of the stable and entering nodes see Figure 4a) when adaptive Vivaldi (1st row) and Charming (2nd row) algorithms are used. See Figure 4b for convergence and mean square error rates.

GNP) could mitigate the problems related to oscillations. In the future, we would like to explore the behavior of Charming in distributed settings further. Since it will be easy to modify existing simulation-based NC algorithms to apply Charming-like interactions, we would like to experiment with hybrid algorithms similar to the one shown in Algorithm 4.

Algorithm 4

CHARMING_VIVALDI(ℓ_{ij}, x_i, x_j, e_j)

1. $w = \frac{e_i}{e_i+e_j}$
2. $e_s = \frac{|||x_i-x_j||_2-\ell_{ij}}{\ell_{ij}}$
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. $x_i = x_i + \delta \times (\ell_{ij} - ||x_i - x_j||_2) \times \frac{x_j-x_i}{||x_j-x_i||_2}$ {repel/attract x_i}
6. $x_j = x_j + \delta \times (\ell_{ij} - ||x_i - x_j||_2) \times \frac{x_i-x_j}{||x_i-x_j||_2}$ {repel/attract x_j}



(a) (b)

Figure 4: (a) shows the stable nodes (in blue) and the nodes that are about to join the network (in orange). Latencies (RTTs) for the new nodes are sampled on a regular grid extended from the stable grid embedding of the existing nodes for simulation purposes. There are about 150 existing and 150 new nodes in the network. (b) Mean square error values from the simulation (shown in Figure 4 comparing adaptive Vivaldi and Charming algorithms in how well the new nodes are accommodated. Notice again the oscillatory behavior of adaptive Vivaldi algorithm.

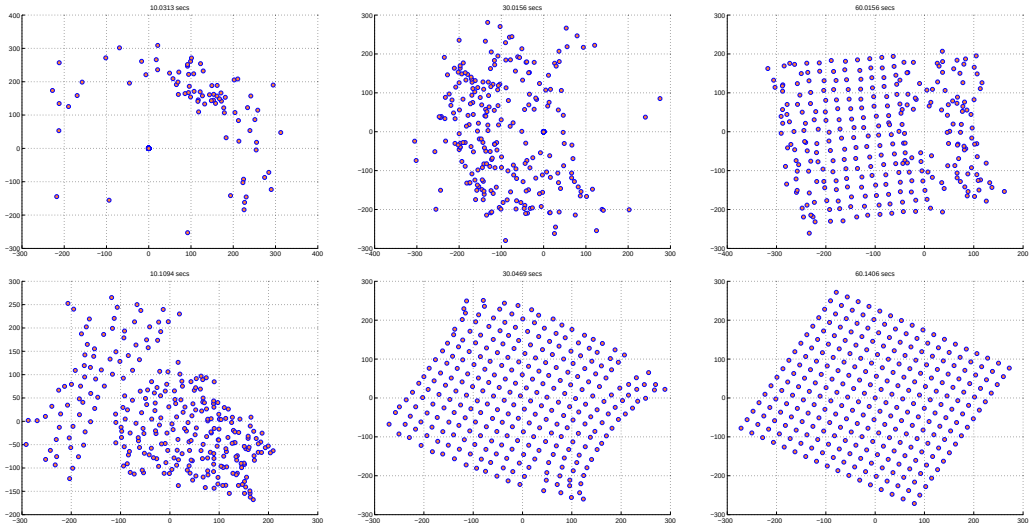


Figure 5: Evolution of the nodes (see the network layout in Figure 2a when the adaptive Vivaldi (1st row) and Charming (2nd row) algorithms are used in a distributed setting. See Figure 4 for convergence and mean square error rates.

References

[1] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. Pic: Practical internet coordinates for distance estimation. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 178–187, Washington, DC, USA, 2004. IEEE Computer Society.

[2] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the ACM SIGCOMM '04 Conference*, Portland, Oregon, August 2004.

[3] C. de Launois, S. Uhlig, and O. Bonaventure. A stable and distributed network coordinate system. Technical report, Universite Catholique de Louvain, December 2004.

[4] Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. Idmaps: a global internet host distance estimation service. *IEEE/ACM Trans. Netw.*, 9(5):525–540, 2001.

[5] Jonathan Ledlie, Peter Pietzuch, and Margo Seltzer. Stable and accurate network coordinates. In *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, page 74, Washington, DC, USA, 2006. IEEE Computer Society.

[6] Hyuk Lim, Jennifer C. Hou, and Chong-Ho Choi. Constructing internet coordinate system based on delay measurement. In *IMC '03: Proceedings*

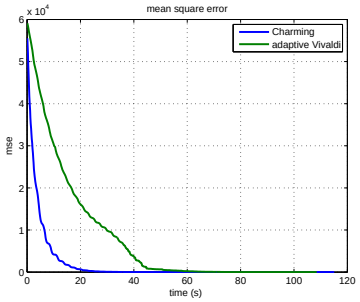


Figure 6: Mean square error values from the simulation (shown in Figure 4 comparing adaptive Vivaldi and Charming algorithms for a distributed NC computation. Note that, although it is still significantly slower than Charming, adaptive Vivaldi converges much faster than it did in the centralized computation. This result with the randomized nature of our distributed network simulation suggests that oscillations caused by Vivaldi could be due to localized measurements. This might also explain the relatively poor performance of Vivaldi in the Internet where most of the measurements are likely to come from a close neighbor instead of a random node of the complete network.

of the 3rd ACM SIGCOMM conference on Internet measurement, pages 129–142, New York, NY, USA, 2003. ACM Press.

[7] T. S. Eugene Ng and Hui Zhang. Predicting internet network distance with coordinates-based approaches. In *INFOCOM*, 2002.

[8] Marcelo Pias, Jon Crowcroft, Steve R. Wilbur, Tim Harris, and Saleem N. Bhatti. Lighthouses for scalable distributed location. In *IPTPS*, pages 278–291, 2003.

[9] Yuval Shavitt and Tomer Tankel. Big-bang simulation for embedding network distances in euclidean space. *IEEE/ACM Trans. Netw.*, 12(6):993–1006, 2004.

[10] Liying Tang and Mark Crovella. Virtual landmarks for the internet. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 143–152, New York, NY, USA, 2003. ACM Press.

[11] Li wei Lehman and Steven Lerman. Pcoord: Network position estimation using peer-to-peer measurements. In *NCA '04: Proceedings of the Network Computing and Applications, Third IEEE International Symposium on (NCA'04)*, pages 15–24, Washington, DC, USA, 2004. IEEE Computer Society.