Resolving the Noxious Effect of Churn on Internet Coordinate Systems

Bamba Gueye*, Guy Leduc

University of Liege, Belgium
{cabgueye,guy.leduc}@ulg.ac.be

Abstract. Internet Coordinate Systems (ICS) provide easy and practical latency predictions in the Internet. However, peer dynamics (*i.e*, churn), which is an inherent property of peer-to-peer (P2P) systems, affects the accuracy of such systems. This paper addresses the problem of churn in an ICS without landmarks, like Vivaldi. We propose a framework to assess the robustness of such an ICS in the presence of churn, and evaluate two models for handling churn. The key idea is to reactively recover lost neighbours, either by picking new nodes at random, or by selecting a new one among the node's two-hop neighbours, while maintaining high reliability and low communication overhead. We then show by simulations that our models mitigate the impact of churn, and lead to a good accuracy compared to an instance of an ICS running without churn.
Keywords: ICS, Node Churn, Accuracy, Clustering.

1 Introduction

Nowadays, a new class of large-scale globally-distributed network services and applications such as distributed overlay network multicast, content addressable overlay networks, and peer-to-peer file sharing (*e.g.*, Gnutella BitTorrent, etc.) have emerged. To achieve network topology-awareness, most, if not all, of these overlays rely on the notion of proximity, usually defined in terms of network delays or round-trip times (RTTs), for optimal neighbor selection during overlay construction and maintenance.

It is important for the new applications presented above to limit the resources consumption and particularly the number of on-demand measurements. In such a context, Internet Coordinate Systems (ICS) [1,2,3] have been proposed to allow hosts to estimate delays without performing direct measurements and thus reduce the consumption of network resources. The key idea of an ICS is to model the Internet as a geometric space and characterize any node in the Internet by a position (i.e., a *coordinate*) in this space. The network distance between any two nodes is then predicted as the geometric distance between their coordinates. Explicit measurements are, therefore, not required any longer.

Generally, an ICS follows a three step procedure. The first step is neighbor selection. In this step, each node in the system chooses a constant number of neighbors. In the second step, each node measures the delays to its neighbors. After collecting the delay

* This work has been partially supported by the EU under projects FP6-FET ANA (FP6-IST-27489)

measurement, all the nodes use an optimization algorithm to compute the coordinates based on these delays.

In a large-scale P2P system, peer dynamics (i.e., churn) is a prevalent phenomenon, which makes maintenance a challenging task [4]. Almost every distributed system has to deal with churn: *i.e.*, the continuous process of node arrival and departure due to various reasons, *e.g.*, link outage, graceful leaves, failure, etc. In an ICS, in the presence of churn, nodes often did not have time to settle into a stable position before they exited the system. In such case, some nodes will update their coordinates according to neighbors that have not stabilized their coordinates, leading to skewed coordinates [5,6]. As consequence, they had a deleterious effect not only on themselves, but on the overall system convergence. Since churn is inherent to P2P system, it is mandatory to build an ICS that should predict latency with accuracy under churn situation. The remainder of this paper is dedicated to determining whether an ICS can be built so that it continues to perform well under churn.

We study how to reduce the harmful effect of churn on Vivaldi by intelligently remplacing reactively node's neighbors which have left the system. In so doing, we considered two potential solutions to the problem of sustaining a coordinate system under high churn rates. The first one, the *Random Replacement* (RR) replaces a failed neighbor with a randomly chosen node. The second strategy, the *Two-hop Neighbors Replacement* (TNR), considers the two-hop neighbors as a preference list, and thus picks randomly in this list the set of nodes that will be used to replace the failed ones. Note that, the set of two-hop neighbors is formed by the union of the direct neighbors, *i.e.*, the set of peer nodes that are used as neighbors in the ICS for the purpose of coordinate computation, and the neighbors' neighbors. We then provide a comparison of the performance of a range of different node selection strategies in three real-word traces. One of our contribution is to show that Vivaldi can in fact handle churn following an approach that reactively recovers lost neighbors. Our second contribution is an examination of churn in a Self-Organized network, according to a Two-Tier approach of Vivaldi, where nodes cluster themselves based on their network distance [7].

Our main results can be summarized as follows: (*i*) Coordinate systems that experience churn have trouble converging; (*ii*) The strategies of node replacement perform well compared to the case where no recovery mechanism is settled; (*iii*) The Two-tier approach, where we apply the TNR and the RR techniques for addressing churn, is more accurate under high node churn rate compared to a flat Vivaldi.

In this paper, we begin by giving an overview on Vivaldi and a few proposed works that do provide a mechanism for handling churn in Vivaldi, but do not rely on the "original" Vivaldi (Sec 2). Then, we describe different churn scenarios and analyze the behaviour of the RR and TNR strategies in the presence of churn (Sec. 3). We also test as case study the RR and the TNR strategies in a Two-Tier Vivaldi environment (Sec. 4). We finally conclude this paper by reminding its main contributions (Sec. 5).

2 Background on Vivaldi

Vivaldi [2] is probably the most successful Internet coordinates system that has been proposed so far. It does not require any fixed network infrastructure and makes no dis-

tinctions between nodes. In fact, Vivaldi [2] is based on a simulation of springs, where the position of the nodes that minimizes the potential energy of the spring also minimizes the embedding error. A Vivaldi node collects distance information towards other nodes (its neighbors) and computes its new coordinates with the collected measurements (sample). The sample used by each node A is based on measurement to a node, B, its coordinates x_B and the estimated error e_B being reported by B. A relative error of this sample is then computed with respect to d_{AB} and $\hat{d}_{AB}$. Note that d_{AB} and $\hat{d}_{AB}$ represent respectively the measured distance and the estimated distance. The node then computes the sample weight, balancing local and remote error. The local (resp. remote) error represents node A's (resp. node B's) confidence in its own coordinate. Thus, the coordinates are updated by moving a small step toward the perfect position that best reflects the RTT measured. The Vivaldi algorithm quickly converges towards a solution when latencies satisfy the triangle inequality. Vivaldi considers a few possible coordinate spaces that might better capture the underlying structure of the Internet Euclidean spaces, spherical coordinates, etc. For the present study, we use a 2D Euclidean space and each node computes its coordinates by doing measurements with 32 neighbors.

It is worth noticing that Dabek et *al.* [2] have only studied Vivaldi under stable environment. In such case, nodes will not leave the system after their join. This assumption is not realistic due to the prevalence of peer dynamics in P2P systems. Therefore, no recovery mechanism is proposed in [2] when nodes have lost their neighbors. To overcome this limitation, Ledlie et *al.* in [5] propose a simple technique by considering a *"full Vivaldi embedding"* that runs over Azureus [8] (now called *Vuze*) which is a popular clients for BitTorrent, a file sharing protocol. Note that, in Azureus metadata are stored in a Distributed Hash Table (DHT) which enables Vivaldi to choose node's neighbors. In this approach, *i.e.*, full Vivaldi embedding, Vivaldi nodes have no dedicated set of neighbors as designed in [2]. Therefore, the information necessary for a coordinate update is piggybacked in the other application level messages, such as routing table heartbeats [5]. In such case, nodes have no control over the selection of which nodes we gossiped with, and nodes communicate only with a limited set of nodes that was much smaller than the number of nodes with which it actually communicated.

In the same way, Chen et *al.* in [6] propose also *Myth* a Landmark-based NCS which used partially a full Vivaldi embedding as in [5] running on Bamboo DHT. Myth has a initial coordinates prediction scheme that is used before the Vivaldi algorithm. This scheme like GNP [1] uses the nodes already in the system as landmark to compute its coordinates. Requiring that some nodes be designated as landmarks may be a burden on symmetric systems (such as P2P systems). To obtain a neighbor, a given node randomly generates a global unique identifier and queries the DHT for it, and then the DHT returns the node that owns this identifier which will be used as neighbor. Assuming that nodes did not have a fixed set of neighbors, as in [5,6], is an easy way to let down the problem of how a new neighbor will take the place of the left one. It is worth noticing that a fixed set of neighbors is very important because nodes could expect regular exchanges for updating their coordinates. In contrast to previous works, our approaches for handling churn is not based on a structured P2P overlay network or DHT. As a consequence, we will rely on the "original" Vivaldi, where each node has a fixed set of neighbors.

3 Handling node churn in Vivaldi

In this section we formally model peer churn, describe and evaluate our two replacement strategies for reducing the harmful effect of churn on Vivaldi.

3.1 Churn scenario: Node Arrival and Departure Rates

Churn can be modelled by two kinds of peer-level characterization. Firstly, the session length distribution, which is one of the most basic properties of churn, captures how long peers remain in the system each time they appear. Secondly, the downtime can be defined as the interval between the moment a peer departs and its next arrival. The characterization of churn has been relatively well addressed in the literature. One issue is whether a good mathematical distribution exists to model network churn. In fact, previous works [9,10,11] have shown that both distributions in typical P2P systems are exponential, though other studies claim that the distribution of session length follows a Pareto distribution [5,12,13]. In contrast, the results in [14] show that the distribution of session lengths does not exactly follow the exponential distribution or the long-tailed Pareto distribution across different P2P systems, (e.g., Kad, Gnutella, BitTorrent). Thus, there is still no clear answer on how to model the peer behavior appropriately. Nevertheless, as some studies have shown that session lengths are either exponential or Pareto, we model churn in our simulations by testing both distributions. Note that, when the session length is modelled as a Pareto distribution we have considered that nodes sleep for a random period with uniform distribution and rejoin the system as a newcomer. In contrast, when the session length follows an exponential distribution, the downtime is modelled also as exponential [9,10,11].

We study the set of strategies described above and we believe that they are all relevant in practice. Since this paper focuses more on handling churn in network coordinate systems, finding a good distribution that fits well churn in P2P system is left for future work.

Modeling peer churn: In the previous section we showed that the session length and the downtime can be modelled by different kind of distributions. We concentrate primarily on the use of the quantile function for the formulation of distributions. In fact, the quantile function can be used as the basis for a range of approaches to the construct of models of populations. After the probability density function and the cumulative density function, the *Quantile Function*, QF, denoted by $QF(p)$, provides a third way of defining a distribution. By definition, the QF of a probability distribution is the inverse of its cumulative distribution function. Formally, we have

$$x_p = \text{the value of } x \text{ for which Probability}(X \leq x_p) = p$$

For instance, if $\mathcal{F}$ is a probability distribution function, the QF may be used to "construct" a random variable having $\mathcal{F}$ as its distribution function. This fact serves as the basis of a method of simulating the churn from an arbitrary distribution with the aid of a random number generator. In the following, we present a detailed system model based on the above observation.

In our simulations, the individual peers have different arrival rates for the join/leave events. Theses events can be scheduled as follows according to a fixed distribution. The quantile function for the exponential distribution at time t can be computed as:

$$QF(p) = -\frac{\ln(1-p)}{\lambda} \qquad \text{for } 0 \leq p < 1 \tag{1}$$

where λ is the parameter of the distribution.

To model the peers behavior following a Pareto distribution, the quantile function is obtained by:

$$QF(p) = \frac{\beta}{(1-p)^{\frac{1}{\alpha}}} \qquad \text{for } 0 \leq p < 1 \tag{2}$$

where α is a shape parameter that determines how skewed the distribution is, and β is a location parameter that determines where the distribution starts.

Finally, the quantile function of the uniform distribution, which defines an equal probability over a given range, is expressed as follows:

$$QF(p) = (x_{min} + x_{max} \times p) \times mean \qquad \text{for } 0 \leq p < 1. \tag{3}$$

3.2 Approaches for handling churn

We focus only on agnostic strategies, *i.e.*, where we ignore past uptime or availability of individual node because we do not explicitly try to minimize churn, but rather to deal with its presence. Therefore, we study two set of strategies that we believe are both relevant in practice: (*i*) the *Random Replacement* (RR) where each node replaces a failed neighbor reactively with a uniform-random available node; (*ii*) the *Two-Hop Neighbors Replacement* (TNR) where each node replaces lost neighbors by one of its neighbors' neighbors (*i.e.*, node's *Two-hop neighbors*). It should be noted that the list of neighbors' neighbors can be obtained in the network by simply piggybacking the information in the messages exchanged by the ICS system.

We allow our selection algorithm to react immediately after each change in node's neighbors state. We feed the sequence of events into the P2psim simulator [15] following the different distributions of churn characteristics described in Sec. 3.1. Events are nodes joins and failures. The obtained results are shown in the next section.

3.3 Experimental results

In this section we present the results of an extensive simulation study of Vivaldi under churn using the P2Psim discrete-event simulator [15]. Each Vivaldi node has 32 neighbors and results are obtained for a 2-dimensional Euclidean space.

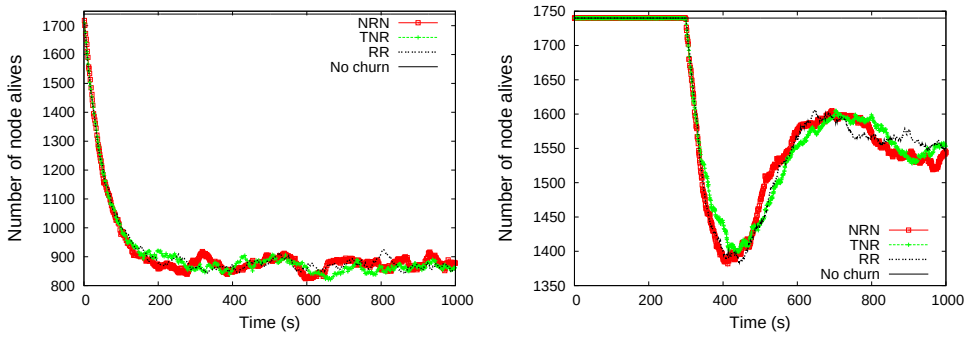
We performed a set of simulations using three datasets: the P2psim data (1740 nodes) [15], the Meridian data (2500 nodes) [16], and the PlanetLab data which we collected using *ping* measurements between 180 PlanetLab nodes [17]. Note that, the King and Meridian data sets are obtained following the *King* measurement technique [18] which is similar to ping in the sense that it estimates the latency between arbitrary end hosts by using recursive DNS queries. Based on these delay matrices, we study through

the basic *Absolute Estimation Error* (AEE) and *Relative Estimation Error* (REE) metrics the accuracy of Vivaldi under churn. For a given link between two nodes A and B we have the following definitions:

$$AEE(AB) = |EST(A,B) - RTT(A,B)|$$
$$REE(AB) = \frac{AEE(AB)}{RTT(A,B)}$$

where $RTT(X,Y)$ is the measured RTT between the nodes X and Y, and $EST(X,Y)$ is the estimated RTT obtained with the coordinates of the nodes X and Y.

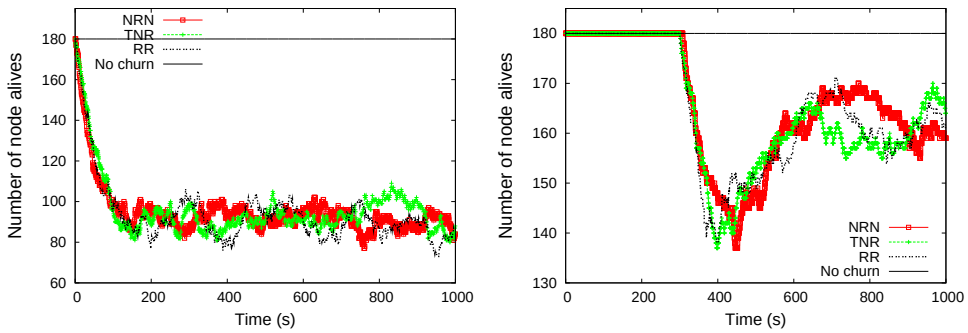
During our simulations, in some cases, churn is temporary as the departed peers may rejoin the system; churn can also be permanent as peers may depart the system forever. In particular, we set the Pareto distribution parameters in Eq. 2 as: $\alpha = 1.03$ and $\beta = 300s$. The choice of the value of α is guided by Wang et *al.* in [19] using traces of *PPLive*, which is a popular P2P live streaming system. They have shown that the node's stay duration of PPLive is well approximated by a Pareto distribution of $\alpha = 1.03$. The value of β represents the time when the churn starts in the system. After a node leaves the network, it sleeps (*i.e.*, downtime) during a time which is uniformly distributed between $0.1 \times mean$ and $1.9 \times mean$ (see Eq. 3), where $mean = 100s$, and rejoins the system (if the downtime is not beyond the simulation time) as a newcomer to stay another Pareto distribution. In the second approach of characterizing churn, events for each node is exponentially distributed with a mean of $100s$ (Eq. 1). Indeed, peers within the network are assigned exponentially distributed session lengths. When a peer reaches the end of its session length, it leaves the network and waits an exponentially distributed time for another potential join. It should be noted that the choice of mean session time is consistent with past studies of P2P networks [20].



(a) Exponential distribution (b) Pareto distribution

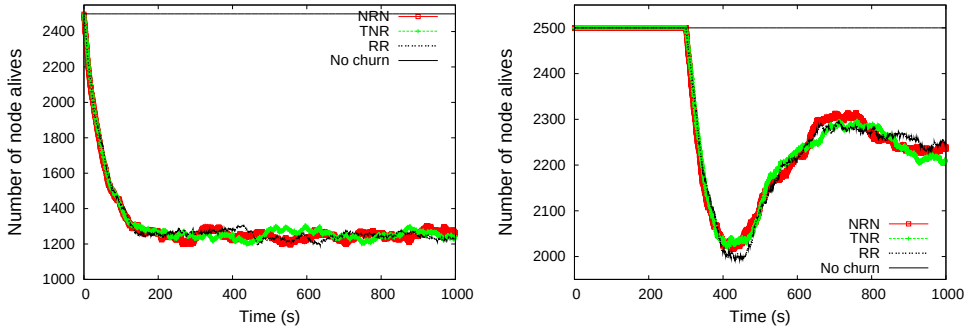
Fig. 1. King dataset: nodes alive as function of time.

Figures 1, 2, and 3 illustrate the general behavior of nodes for the King, PlanetLab, and Meridian data sets during our simulations. Note that *No Recovery Nodes* (NRN) means an instance of Vivaldi under churn without replacement of lost nodes. When



(a) Exponential distribution (b) Pareto distribution

Fig. 2. PlanetLab dataset: nodes alive as function of time.



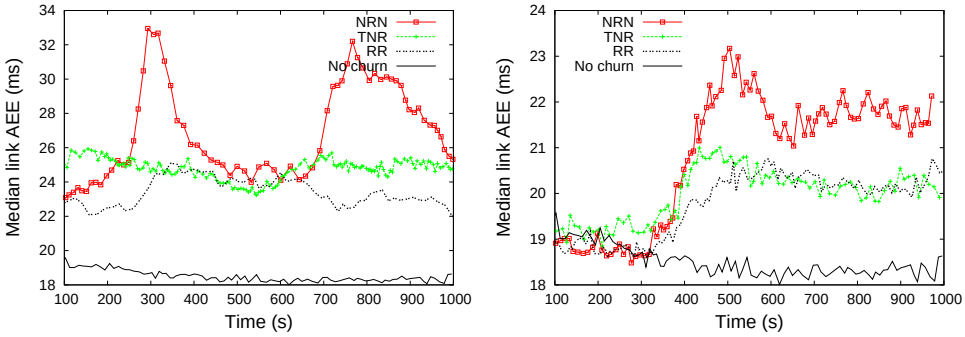
(a) Exponential distribution (b) Pareto distribution

Fig. 3. Meridian dataset: nodes alive as function of time.

the churn is intensive it remains at least 80% of the nodes in the system following the Pareto distribution, whereas for the exponential distribution the average of available nodes is roughly 52%. In summary, churn is more intensive following the exponential distribution.

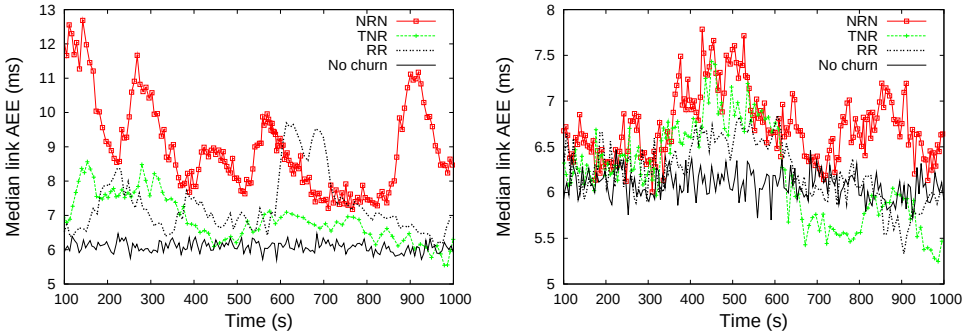
We ran Vivaldi on King, Meridian, and PlanetLab respectively and recorded the coordinates of the nodes every 10 ticks as well the corresponding time to this tick in order to plot the evolution of coordinates as function of time. Note that a tick represents an update coordinates once. Furthermore, every 10 ticks we compute the percentile AEE and ERR overall links and we considered the 5th, 10th, 25th, 50th, 75th, 90th, and 95th percentile error. Each simulation runs for $1000s$ of simulated time. Unless otherwise noted, all figures are for simulations done in the churn environment.

Figures 4, 5, and 6 illustrate the general behavior of Vivaldi under different churn scenarios as function of time according to King, PlanetLab and Meridian data set. As expected, the curve labelled NRN (No Recovery Nodes) in Figures 4, 6, 5 which shows an instance of Vivaldi under churn without replacement of lost nodes, has always the



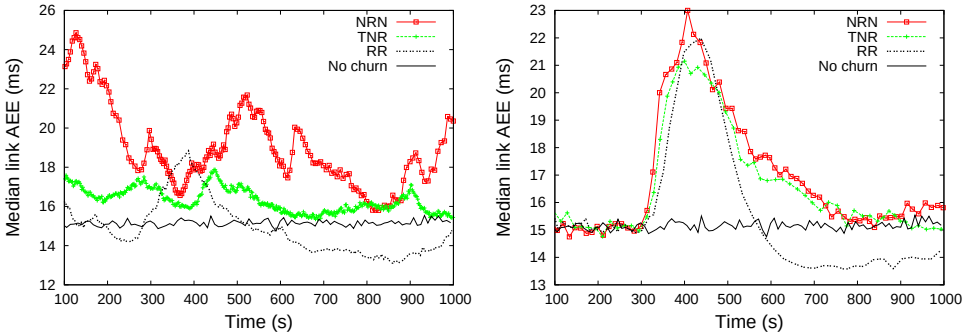
(a) Churn following Exponential distribution (b) Churn following Pareto distribution

Fig. 4. King dataset: Median absolute error as function of time.



(a) Churn following Exponential distribution (b) Churn following Pareto distribution

Fig. 5. PlanetLab dataset: Median absolute error as function of time.



(a) Churn following Exponential distribution (b) Churn following Pareto distribution

Fig. 6. Meridian dataset: Median absolute error as function of time.

worst accuracy with respect to AEE metric: the performance of Vivaldi degrades in node churn scenario and Vivaldi never converges to a steady state. The main observation one can notice is that it exists a benefit of doing neighbors recovery according to our strategies. For instance, Figure 4(b) shows the AEE as a function of time in the case where the session length follows a Pareto distribution. It illustrates that the AEE increases suddenly (roughly at $300s$), as soon as the churn starts in the network. As consequence, one can see the abrupt jump of the NRN curve as well the TNR and RR curves but in less proportion. The same trend is observed in Fig. 5(b) and Fig. 6(b).

We can see also that the accuracy when the churn follows a Pareto distribution is better than the accuracy obtained with an exponential distribution. This is due to the fact that the churn intensive workload is more important in the exponential distribution (Fig.1(a) and Fig. 3(a)). Nevertheless, with our replacement strategies, TNR and RR, one can see in Figure 5(a), 6(a) that the obtained accuracy if quite similar to a stable Vivaldi and then the RR strategy outperforms the stable Vivaldi.

Additionally the RR strategy performs better than the TNR strategy with respect to the Meridian dataset (Fig. 6). Nevertheless, for other datasets the difference is less important. It is worth noticing that in the RR strategy a node can select out of all nodes available in the system to replace its failed neighbors whereas in the TNR strategy a node chooses only out of node's two-neighborhood which is in fact less important. It should be noted that in terms of communication overhead the cost of retrieving the list of all nodes that are in the system is more important compared to the TNR strategy.

By lack of space we have shown only the median AEE. Note that we found similar trend for the other percentiles and for the REE metric.

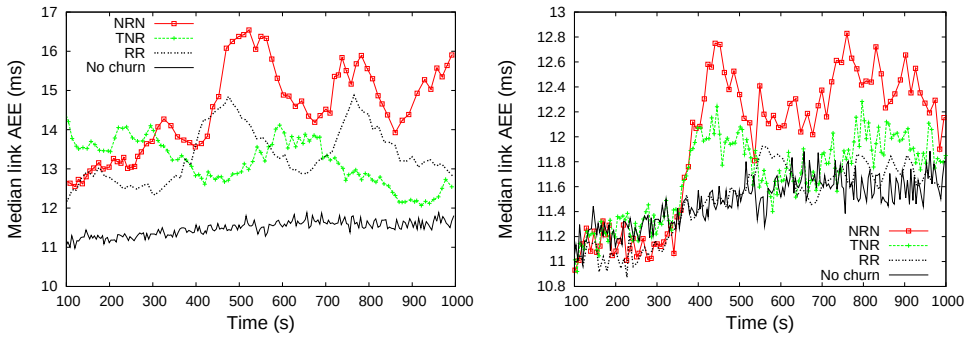
4 Case study: Handling node churn in a Two-Tier architecture

Internet latencies, due to routing policies or path inflation [21], do sometimes violate the triangle inequalities which must hold in a metric space. Such Triangle Inequality Violations (TIV) could be a major barrier for the accuracy of Internet coordinate systems. Kaafar et *al.* have shown that longer edges cause more severe TIVs, and thus proposed a Two-Tier architecture opposed to a flat structure of Vivaldi, based on the clustering of nodes [22]. In fact, coordinates computed at the lower (resp. higher) level of clusters are called local coordinates (resp. global coordinates). Within their cluster, nodes use more accurate local coordinates to predict intra-cluster distances, and keep using global coordinates when predicting longer distances towards nodes belonging to foreign clusters. In this section, according to this Two-tier approach and the self-organizing clustering scheme proposed in [7], we test the strategies studied, in Sec. 3, in a peer dynamics system.

To construct clusters in a self-organizing fashion, each node relies on coordinates provided by their two-hop neighbors, but also on measurements towards a potential existing cluster. For instance, if a node has 32 neighbors in order to estimate its coordinates, its Two-hop neighbors will be formed by at most 1024 nodes. Therefore, nodes do not need global knowledge of nodes in the network, nor distances between these nodes, nor a common landmark/anchor infrastructure. In general, the clustering forming phase can be described as follows: each time a node joins a network, it gets the list

of cluster heads from its Two-hop neighbors set and verifies if the measurement towards the cluster heads satisfies the cluster diameter. If the constraint diameter is satisfied, the node joins the cluster owned by these cluster head. Nevertheless if none of the distances to existing cluster heads satisfies the clustering criterion, the node starts the clustering algorithm on the basis of the coordinates of its Two-hop neighbors set [7]. For more details about the clustering algorithm we suggest the reader to refer to [7]. Next, we describe how we set up the clusters we experimented with to illustrate our results on the Two-tier Vivaldi approach.

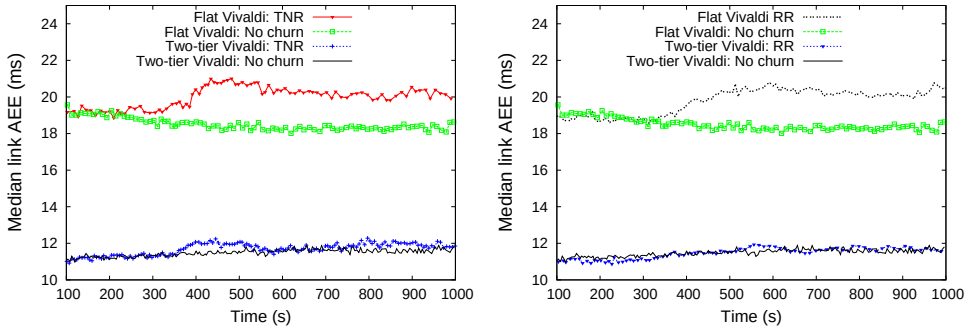
For handling churn in a Two-Tier architecture, we have first based our clusters recognition on the coordinates as observed by running a flat Vivaldi over the P2psim data. Our second step has then consisted of using the algorithm proposed in [7] to self organize nodes into clusters. Following this cluster selection method, we run an extensive simulation either without churn, or under node churn scenario without recovery mechanism, or churn recovery with random replacement (RR), or churn recovery with Two-hop neighbors replacement (TNR). Note that, if a node belongs to a given cluster, it takes halfot its neighbors inside its own cluster and the remaining out of the available nodes in the network. We use the absolute error as our main performance indicator. Again, we compute the AEE over all links to represent the accuracy of the overall system. Nevertheless, if two nodes belong to the same cluster, we used their local coordinates to compute the AEE. Otherwise, we consider their global coordinates in order to estimate the AEE.



(a) Two-tier: exponential distribution (b) Two-tier: Pareto distribution

Fig. 7. King dataset: Median absolute error as function of time.

Figures 7(a) and 7(b) represent the median Absolute Estimation Error (AEE) belonging to our Two-tier Vivaldi according to the P2psim data. We see that the same trend is observed with respect to an instance of a “flat Vivaldi” (Fig. 4). In other words, the churn NRN still has the worst accuracy compared to an instance of Vivaldi where we replace nodes leaving the system. Furthermore, Fig. 8 clearly illustrates that the AEE computed based on Two-tier architecture are much less than errors as computed using the flat Vivaldi. More generally, improvements inside these clusters is explained



(a) Two-hop Neighbors Replacement (TNR) (b) Random Replacement (RR)

Fig. 8. King dataset: Comparison of median absolute error between Two-tier and Flat Vivaldi.

by the fact that intra cluster nodes, when computing their local coordinates select only close by nodes as their neighbors. This constraints the node-to-neighbors edge lengths and thus reduces the selection of severe TIVs likelihood.

5 Conclusion

We have shown that the performance of Vivaldi degrades in churn scenario. We designed and evaluated two strategies for reducing the harmful effect of churn in Vivaldi. These strategies were deployed under different churn distribution characteristics. The experimental results according to three data sets show that the Random Replacement (RR) and the Two-Hop Neighborhood Replacement (TNR) improve the precision of Vivaldi under churn environment. The RR outperforms the TNR with respect to churn following an exponential distribution. Nevertheless, in the case of a Pareto distribution, the gap noticed between both strategies is much smaller.

Although we focused on Vivaldi for experimentations, the proposed strategies are independent of the embedding protocol used. Our proposed approach would then be general enough to be applied in the context of coordinates computed by other Internet coordinate system than Vivaldi.

We have also considered churn situations in a Self-Organized network, with respect to a Two-tier approach of Vivaldi, where we applied our Random Replacement and the Two-Hop Neighborhood Replacement strategies. The Two-tier approach is more accurate under high node churn rate compared to a flat Vivaldi. Our findings show that the performance of the Two-tier Vivaldi exceed that of flat Vivaldi a lot in churn scenario as well as in a scenario without churn. Our future work would then consist on the deployment of the Two-tier approach on Internet (*e.g.*, PlanetLab).

Even though this paper does not address the problem of comparing the Pareto and the exponential distribution, we note that the obtained accuracy, when the session length is modelled as a Pareto distribution, is lower than when it follows an exponential distribution. We are pursuing further study for more general conclusion, considering different values for the parameters of both distributions.

References

1. T. S. E. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proc. IEEE INFOCOM*, New York, NY, USA, June 2002.
2. F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proc. ACM SIGCOMM*, Portland, OR, USA, Aug. 2004.
3. B. Donnet, B. Gueye, and M. A. Kaafar, "A survey on network coordinates systems, design, and security," *IEEE Communications Surveys & Tutorials*, To appear.
4. P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing churn in distributed systems," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 147–158, 2006.
5. J. Ledlie, P. Gardner, and M. I. Seltzer, "Network coordinates in the wild," in *Proc NSDI*, Cambridge, apr 2007.
6. Y. Chen, G. Zhao, A. Li, B. Deng, and X. Li, "Myth: An accurate and scalable network coordinate system under high node churn rate," in *IEEE International Conference on Networks ICON'*, Adelaide, Australia, Nov. 2007.
7. F. Cantin, B. Gueye, M. A. Kaafar, and G. Leduc, "A self-organized clustering scheme for overlay networks," in *Lectures Notes in Computer Science 5343*, Dec. 2008, pp. 59–70.
8. "Azureus bittorent client," http://azureus.sourceforge.net/index.php.
9. D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Analysis of the evolution of peer-topeer systems," in *Principles of Distributed Computing*, 2002, pp. 233–242.
10. S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a dht," in *USENIX Conference*, 2004.
11. J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil, "A performance vs. cost framework for evaluating dht design tradeoffs under churn.," in *INFOCOM*, 2005, pp. 225–236.
12. F. Bustamante and Y. Qiao, "Friendships that last: Peer lifespan and its role in p2p protocols," *Web Content Caching and Distribution*, pp. 233–246, 2004.
13. J. Liang, R. Kumar, and K. W. Ross, "The kazaa overlay: A measurement study," *Computer Networks Journal (Elsevier)*, 2005.
14. D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *IMC*, Rio de Janeriro, Brazil, 2006, pp. 189–202.
15. *A simulator for peer-to-peer protocols*, `http://www.pdos.lcs.mit.edu/p2psim/index.html`.
16. B. Wong, A. Slivkins, and E. Sirer, "Meridian: A lightweight network location service without virtual coordinates," in *Proc. the ACM SIGCOMM*, aug 2005.
17. *PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services*, 2002, `http://www.planet-lab.org`.
18. K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating latency between arbitrary Internet end hosts," in *Proc the SIGCOMM Internet Measurement Workshop*, Marseille, France, Nov. 2002.
19. F. Wang, Y. Q. Xiong, and J. C. Liu, "mtreebone: A hybrid tree/mesh overlay for application-layer live video multicast," in *ICDCS*, June 2007.
20. S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," vol. 9, pp. 170–184, Aug. 2003.
21. H. Zheng, E. K. Lua, M. Pias, and T. G. Griffin, "Internet routing policies and round-trip-times," in *Proc. the Passive and Active Measurement Workshop – PAM*, Boston, MA, USA, Mar. 2005, Lecture Notes in Computer Science (LNCS) 3431.
22. M. A. Kaafar, B. Gueye, F. Cantin, G. Leduc, and L. Mathy, "Towards a two-tier internet coordinate system to mitigate the impact of triangle inequality violations," in *Proc. IFIP Networking Conference*, Singapore, May 2008, LNCS 4982, pp. 397–408.