Scalable Route Selection for IPv6 Multihomed Sites

Cédric de Launois *, Steve Uhlig **, and Olivier Bonaventure

Department of Computing Science and Engineering
Université catholique de Louvain
{*delaunois*, *suh*, *bonaventure*}@info.ucl.ac.be

Abstract. IPv6 multihoming with multiple prefixes increases the number of paths available between multihomed sites. Selecting the path with the lowest delay is important for many interactive and real-time applications. We propose in this paper to use a network coordinate system as a scalable an efficient way to help hosts in IPv6 multihomed sites in selecting the best source and destination IPv6 prefixes. Relying on RTT measurements from the RIPE NCC data set, our experiments show that, using synthetic coordinates, all paths with really bad delays can be avoided. Moreover, we are able to select paths with a delay at most 20% worse than the lowest delay for more than 85% of the pairs of multihomed sites. A second contribution is SVivaldi, an improved version of the Vivaldi distributed algorithm for computing synthetic coordinates. We show that SVivaldi produces more accurate coordinates and is able to stablilize Vivaldi's coordinates.

Key words: IPv6 multihoming, route selection, network coordinates.

1 Introduction

Today, the Internet connects more than 18000 *Autonomous Systems* (AS), operated by many different technical administrations. The large majority are *stub* ASes, i.e. autonomous systems that do not allow external domains to use their infrastructure, except to reach them. Less than 20% of autonomous systems provide transit services to other ASes [1]. They are called *transit* ASes. The Border Gateway Protocol (BGP) is used to distribute routing announcements among routers that interconnect ASes.

Internet connectivity takes a strategic importance for a growing number of companies. Therefore, many ISPs and corporate networks wish to be connected through at least two providers to the Internet, primarily to enhance their reliability in the event of a failure in a provider network, but also to increase their network performances such as network latency. Nowadays, at least 60% of stub

* Supported by a grant from FRIA (Fonds pour la Formation à la recherche dans l'Industrie et dans l'Agriculture, Belgium). This work is also partially supported by the Walloon Government within the WIST TOTEM project.
** Supported by the FNRS (Fonds National de la Recherche Scientifique, Belgium).

domains are multihomed to two or more providers [1], and it is expected that IPv6 sites will continue to be multihomed. In order to preserve the size of the BGP routing tables in the Internet, every IPv6 multihoming solution is required to allow route aggregation at the level of their providers [2]. Most IPv6 multihoming mechanisms proposed at the IETF rely on the utilization of several IPv6 provider-aggregatable prefixes per site, instead of a single provider-independent prefix, see [3, 4] and the references therein.

It has been shown that the use of multiple provider-aggregatable prefixes increases the number of paths available between multihomed sites [5]. When two ASes have respectively m and n providers, the total number of paths is typically $m \times n$. Among the available paths, some of them offer lower delays than the path selected by BGP using traditional IPv4 multihoming [5]. Since the number of providers today ranges from 2 to more than 20 [5], the number of paths available between two stub ASes is often higher than or equal to 4. The end-to-end delay is a major component of the performance of a path since it reflects its length and bandwidth. Choosing paths that offer low delays is an important traffic engineering goal [6–8]. One approach to identify the path with the lowest delay is to probe all available paths. However, measuring adequately the delay of a path typically requires many probes, distributed over time [9]. For example, RON [10] uses at least 10 probes to estimate the latency of a path, with 14 seconds in average between two consecutive probes. Besides the high measurement overhead imposed, the cost of probing all available paths to identify the one with the lowest delay will probably outweigh the benefit of an intelligent choice. Suppose that each multihomed site has about 2.5 providers on average. If each multihomed site has a significant amount of traffic towards 1000 other multihomed sites, then the average number of paths to probe is $1000 \times (2.5)^2 = 6250$ for each site.

In this paper, we propose to use a network coordinate system in order to select, for each source and destination pair, the IPv6 addresses to be used that will lead to a low delay path. Our goal is to avoid all paths with really bad delays, and to use the path with the lowest delay as much as possible. In section 3, we explain how synthetic coordinates can help hosts in IPv6 multihomed sites to select their best paths towards a destination node. We then present and improve a distributed algorithm that computes these coordinates in a scalable way. Next, in section 4, we evaluate the quality of the routes selected using a coordinate system. Finally, the related work is presented in section 5.

2 Background on IPv6 Multihoming

This section provides some background on how sites are expected to multihome in IPv6. Figure 1 illustrates a standard IPv6 multihomed site. Suppose two Internet Service Providers, AS 10 and AS 20, provide connectivity to the multihomed site AS 65001. Each provider assigns to AS 65001 a site prefix, respectively 2001:10:1::/48 and 2001:20:1::/48. The two prefixes are advertised by the site exit routers RA and RB to every host inside AS 65001. These prefixes are used

to derive one IPv6 address per provider for each host interface. In this architecture, AS 65001 advertises prefix 2001:10:1::/48 only to AS 10, and AS 10 only announces its own IPv6 aggregate 2001:10::/32 to the global Internet. This solution is expected to be used by stub ASes. Large transit ASes are not concerned by this solution since they will receive provider-independent IPv6 prefixes. Consequently, in this study, we focus on stub ASes and maybe small transit ASes as well, such as broadband access providers, enterprises or campus networks.

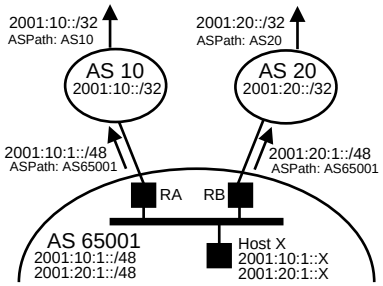

Fig. 1. IPv6 Multihoming

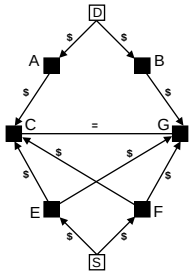

Fig. 2. Topology

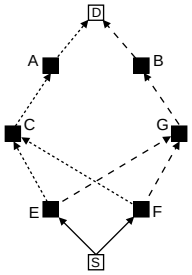

Fig. 3. IPv6 path tree

Figure 2 shows an AS-level interdomain topology with shared-cost peerings and customer-provider relationships. An arrow labelled with "$" from AS x to AS y means that x is a customer of y. A link labelled with "=" means that the ASes have a shared-cost peering relationship [11]. Each AS prefers routes received from a customer over routes received from a peer. In this figure, both S and D are dual-homed ASes. Suppose that both AS D and AS S use IPv6 multihoming with multiple PA prefixes. Each host in S has two IPv6 addresses. When selecting the source address of a packet to be sent, the host in S could in theory pick any of its two addresses. However, for security reasons, IPv6 providers must refuse to convey packets with source addresses outside their address range [3, 4]. For example, E refuses to forward a packet with a source address belonging to F. Using traditional IPv4 multihoming, two BGP routes towards D (e.g. $SECAD$ and $SFCAD$) are advertised by E and F to S. In an IPv6 multihoming scenario, since both S and D have two prefixes, S can reach D via A or B depending on which destination prefix is used, and via E or F depending on which source prefix is used. So, S has a total of four paths to reach D : $SECAD$, $SEGBD$, $SFCAD$ and $SFGBD$, see figure 3.

3 The use of Network Coordinate Systems to Identify paths with Lower Delays

Synthetic coordinate systems have been originally developed to allow an Internet host to predict the round-trip delays to other hosts, without having to contact

them first [12–15]. Each host computes its synthetic coordinates in some coordinate space such that the distance between the synthetic coordinates of two hosts predicts the RTT between them. For example, if two hosts have coordinates x and y respectively in the coordinate space, the distance $\|x - y\|$ is a predictor of the RTT between them.

IPv6 hosts currently arbitrarily choose between two or more global-scope IPv6 addresses. The source address selection algorithm specifies that, for a given destination address, the source address with the longest matching prefix must be selected [16]. We propose that hosts in IPv6 multihomed sites base their source and destination IPv6 address selection on the predicted delays provided by synthetic coordinates. The selection of those addresses is made once at the start of a flow (e.g. TCP connexion) between two hosts. When the addresses are selected, they do not change during the connexion to avoid packet re-ordering.

Typically, hosts whithin a single IPv6 prefix should have the same coordinates. The coordinates of a prefix should thus be a good estimate of the coordinates of any host within this prefix. We propose that the DNS server of a site computes the coordinates for the few prefixes assigned to the site, and publishes them in the Domain Name System. These coordinates can be advertised using a new DNS resource record. In practice, this resource record can be associated directly with the domain name of a host, so that the coordinates and the domain name of a host can be provided together in a single DNS response message.

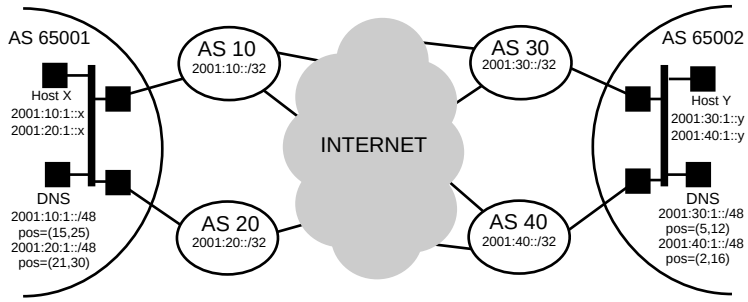

Fig. 4. Both dual-homed IPv6 sites have computed the coordinates associated to their prefixes. Those coordinates are published using the DNS.

Figure 4 illustrates two dual-homed IPv6 sites. The DNS server in AS 65001 has computed coordinates $(15, 25)$ and $(21, 30)$ for its prefixes 2001:10:1::/48 and 2001:20:1::/48 respectively. Note that we could use other coordinate spaces than the 2-dimensional euclidean one. In order to predict the lowest delay path towards a host Y in AS 65002, a host X in AS 65001 first issues a DNS request for the addresses and coordinates of Y. For example in figure 4, host X will receive coordinates $(5, 12)$ and $(2, 16)$ associated with addresses 2001:30:1::y and 2001:40:1::y respectively. Next, host X uses those coordinates together with the

coordinates of its own addresses to predict the RTT for all possible couples of (source, destination) addresses. In the example, the path with the lowest delay is obtained by using addresses 2001:10:1::x and 2001:40:1::y, with an estimated RTT of $\|(15,25)-(2,16)\| = 15.8$ ms. The best path can thus be predicted using a single DNS request, instead of performing four series of delay measurements, each involving several probes. Moreover, the coordinates do not change frequently. They can be cached in the DNS resolvers, further reducing the cost associated to the prediction of the best path. In the previous example, each host computes by itself the addresses to use. Another option is that the DNS server makes the choice in behalf of the host by removing bad addresses from the DNS response message. In this case, no modification is required for the hosts.

Any algorithm like GNP, [12] NPS, [13], Vivaldi [15] or Lighthouse [14] can be used to assign synthetic coordinates to hosts. In this paper, we evaluate the use of the Vivaldi decentralized network coordinate system, because it has the advantages of being simpler and fully decentralized.

3.1 Computing Stable Synthetic Coordinates

Vivaldi is a light-weight, adaptative and fully distributed algorithm for computing synthetic coordinates for Internet nodes. It does not require any fixed infrastructure and can compute coordinates for itself after collecting latency information from only a few other nodes. This number of nodes is constant and does not depend on the total number of nodes.

Alg. 1. The adaptative Vivaldi algorithm

// Node j has been measured to be rtt ms away, has coordinate x_j,
// and an error estimate of e_j.
// Our own coordinates and error estimate are x_i and e_i.
// The constants c_e and c_c are tuning parameters.
1: Vivaldi(rtt, x_j, e_j)
2: $w = e_i/(e_i + e_j)$
3: $e_s = |\|x_i - x_j\| - rtt| / rtt$
4: $e_i = e_s \times c_e \times w + e_i \times (1 - c_e \times w)$
5: $\delta = c_c \times w$
6: $x_i = x_i + \delta \times (rtt - \|x_i - x_j\|) \times \frac{(x_i - x_j)}{\|x_i - x_j\|}$

Alg. 1 shows pseudocode for the Vivaldi algorithm, as described in [15]. The Vivaldi procedure uses each RTT sample to update its coordinates. The weight w of a sample is based on the ratio between the local and the remote error estimates (line 2). The algorithm tracks the local relative error by using a weighted moving average (lines 3 and 4). The coordinates are updated by moving a small step towards the position that best reflects the RTT measured. The size of the modification depends on the weight of the sample, and on the difference between the measured and the predicted RTTs (lines 5 and 6).

The Vivaldi algorithm quickly converges towards a solution when latencies satisfy the triangle inequality, which states that the distance directly beween two nodes a and c must be less than or equal to the distance along any path going

through an intermediate node b [15]. Unfortunately, in the Internet, the latencies sometimes violate this inequality, due for example to policy-based routing with BGP. In such cases, the nodes never converge towards stable coordinates.

In order to illustrate this phenomenon, we simulate the Vivaldi algorithm using a subset of the full matrix of inter-host Internet RTTs. The data set involves 58 active test boxes from the RIPE NCC Test Traffic Measurements Service. The RIPE measurement configuration is described in [17]. The test boxes are scattered over Europe and a few are located in the US, Australia, New Zealand and Japan. Each test box is equipped with a GPS clock so that the one-way delays between each pair of boxes can be measured accurately (within 10μs). Every day, more than 2000 probes are performed for each test box pair. The interval between two consecutive probes is randomized according to a Poisson distribution, as recommended in [9]. The measurements done also include packet losses, path information, bandwidth and delay variation. They are regularly used by ISPs and in the litterature. In this paper, we only use the RTT computed as the sum of the two one-way delays, between every pair of test boxes. In this simulation 2-dimensional Euclidean coordinates are used for the sake of simplicity.

Figure 5 shows the evolution of the coordinates chosen by Vivaldi for the RIPE nodes. The figure shows that all the nodes constantly update their coordinates, making the whole system to shift slowly.

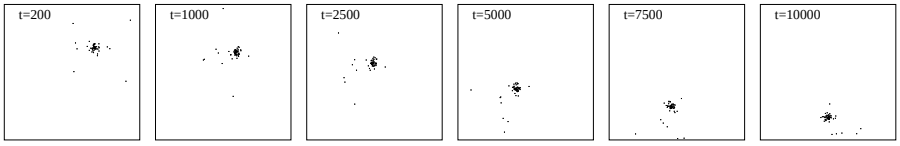

Fig. 5. The evolution of the Vivaldi coordinates of the RIPE nodes.

This phenomenon also occurs with a small system composed of three nodes that violate the triangle inequality, see figure 6. The algorithm that computes synthetic coordinates should minimize either $L1 = \sum_i \sum_j |rtt_{ij} - \|x_i - x_j\||$ or $L2 = \sum_i \sum_j (rtt_{ij} - \|x_i - x_j\|)^2$ [15]. The optimal analytical solution for the illustrated example is reached when nodes a, b and c are aligned, and when the distances (a, b), (b, c) and (a, c) are respectively 5/3, 8/3 and 13/3. In this case, L1 equals 4 and L2 equals 8/3. Figure 7 shows the evolution against time of errors L1 and L2, together with the evolution of the local error estimated by node a. We can observe that the Vivaldi algorithm quickly finds coordinates that minimize error L1, but fails to produce coordinates that minimize error L2. In figure 7, node a tries to lower its local error, and prevents the system to find the optimum solution. Worse, Vivaldi fails to provide stable coordinates. An animation of the system, similar to figure 5 shows that these small oscillations make the coordinates raise indefinitely [18]. In figure 7, error L2 reflects these

oscillations. Such behaviour is unacceptable in our case since this would require constantly updating the DNS with new coordinates.

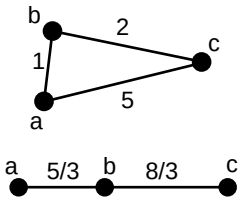

Fig. 6. The RTTs between nodes a, b and c violate the triangle inequality (top). The optimal analytical solution is reached when the nodes are aligned and separated by the distances illustrated (bottom).

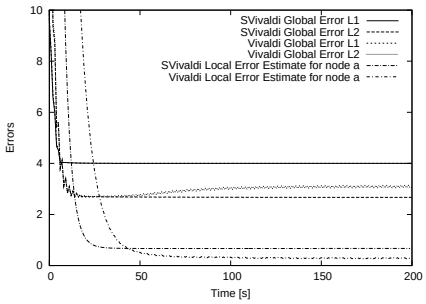

Fig. 7. Evolution against time of the global errors L1 and L2, together with the local error estimation of node a.

In this paper, we propose two modifications to the original Vivaldi algorithm. The first is to improve the local error estimate, such that each node can compute a better estimation of the accuracy of its coordinates. The second is to introduce a new factor to prevent the system from oscillating indefinitely. The new algorithm is presented in Alg. 2.

Alg. 2. The SVivaldi algorithm

1: SVivaldi(rtt_{ij}, x_j, e_j)
2: $w = e_i/(e_i + e_j)$
3: $Neigh_i = Neigh_i \cup \{j\}$
4: $Rtt_i = Rtt_i \cup \{rtt_{ij}\}$
5: $e_i = \frac{\sum_{j \in Neigh_i} |\|x_i - x_j\| - rtt_{ij}|}{\#Neigh_i} \times c_e + e_i \times (1 - c_e)$
6: $loss = c_l + (1 - c_l) * loss$
7: $\delta = c_c \times w \times (1 - loss)$
8: $x_i = x_i + \delta \times (rtt - \|x_i - x_j\|) \times \frac{(x_i - x_j)}{\|x_i - x_j\|}$

Our first modification is to use a more accurate local error estimator. Each node maintains an estimate of the accuracy of its coordinates. The Vivaldi algorithm uses a moving average of recent relative errors. Figure 7 has shown that, in presence of triangle inequalities, this local error estimate can oscillate and prevent the system from finding a good solution. As a more accurate local error estimator we propose the mean of the absolute differences between the predicted RTT for neighbor i and the actual RTT measured for this neighbor, for any neighbor i (Alg.2 line 5). This estimator only requires that a node retains the last RTT measured for each of its neighbors (Alg.2 lines 3 and 4). We will show that this estimator allows to find better solutions. In particular, it allows the system illustrated in figure 6 to find the optimum solution.

While a more accurate local error estimator improves the solution, it still does not prevent the system from oscillating. The Vivaldi algorithm simulates a physical spring between each pair of nodes (i, j) with a rest length set to the measured rtt_{ij}. When triangle inequalities exist, some of those springs can oscillate indefinitely around their rest length.

Our second modification to Vivaldi is to introduce a *loss* factor to allow these springs to progressively rest in a local minimum (Alg. 2 line 6). A constant factor $c_l < 1$ is used to control the quantity of energy that is lost at each oscillation. The value c_l is a tradeoff beween accurate coordinates and short convergence times. A high value for c_l quickly stabibilizes the coordinates but prevents the system from converging to accurate coordinates. Low values for c_l allow the system to find accurate coordinates at the price of a longer convergence period. Our experiments show that a value of $c_l = 0.02$ is a good compromise and we use it in this paper. The *loss* factor (Alg. 2 line 6) is set to zero when the node starts converging, and gradually grows to 1. Whenever a significant change is observed for the measured RTT with a neighbor, the *loss* factor can be reset to zero in order to let the node move again. Our new version of the Vivaldi algorithm in Alg. 2 is called SVivaldi throughout this paper.

3.2 Evaluation of SVivaldi

We now evaluate the stability and the accuracy of the coordinates provided by SVivaldi, and compare them with Vivaldi. The evaluation is performed by simulating the SVivaldi and Vivaldi algorithms. The simulations still rely on the full matrix of Internet RTTs between 59 nodes of the RIPE data set. Vivaldi and SVivaldi use a subset of these RTTs, but the full matrix is needed to evaluate the accuracy of predictions made by those coordinates. For the sake of simplicity, 2-dimensional Euclidean coordinates are used in order to compare the behaviours of SVivaldi and Vivaldi algorithms. Each node takes measurements every second from another node, chosen uniformly at random in a given set of 20 nodes.

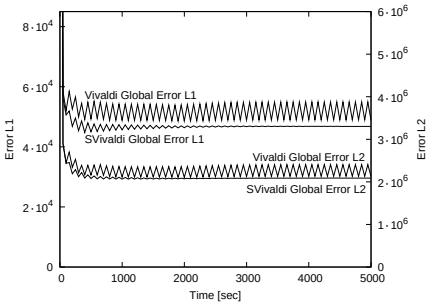

Fig. 8. Evolution against time of the global errors L1 and L2 for 2D coordinates chosen by Vivaldi and SVivaldi.

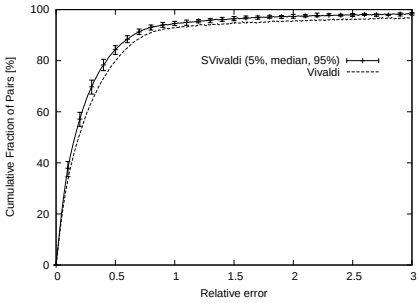

Fig. 9. Cumulative distribution of prediction errors for 2D coordinates chosen by Vivaldi and SVivaldi.

Figure 8 shows the evolution against time of the global errors L1 and L2 for the coordinates chosen by SVivaldi and Vivaldi. The figure shows that using the Vivaldi algorithm, nodes constantly update their coordinates, making both global errors L1 and L2 oscillate around a constant value. Instead, the *loss* factor of our improved SVivaldi algorithm allows the nodes to converge to stable coordinates. In figure 8 both errors L1 and L2 for SVivaldi progessively stop oscillating until completely stabilizing. At this point, the coordinates are no longer updated. As figure 9 confirms, SVivaldi also produces slightly more accurate coordinates.

Figure 9 compares the cumulative distribution of prediction errors for the 2-dimensional Euclidean coordinates chosen by Vivaldi and SVivaldi for the RIPE data set. The coordinates provided by the algorithms depend on the neighbors each node chooses. The figure shows the median distribution among 100 simulations with different sets of neighbors for each node. The 5^{th} and 95^{th} percentiles are also indicated for SVivaldi. The percentiles for the Vivaldi algorithm are similar but are not shown for graphical reasons. Figure 9 shows that the SVivaldi algorithm produces more accurate coordinates than the Vivaldi algorithm. The improvement is due to its improved local error estimator.

We have also verified that SVivaldi preserves Vivaldi's ability to cope well with large numbers of newly-joined nodes with inconsistent coordinates, but due to space limitations, we cannot report this evaluation here.

4 Evaluation of the Quality of the Route Selection

We evaluate in this section how the use of the SVivaldi coordinate system can help in the selection of the lowest delay path between two multihomed IPv6 sites. In this section, the SVivaldi and Vivaldi algorithms use Euclidean coordinates augmented with a height [15]. A packet sent from one node to another must first travel the source node's height, then travel in the Euclidean space, then travel the destination node's height. [15] showed that so called height vectors perform better than both 2D and 3D Euclidean coordinates.

IPv6 multihoming with multiple prefixes is not currently deployed in the Internet. In order to simulate IPv6 multihoming, we follow a similar methodology to the one used in [19, 20]. We emulate a multihoming scenario by selecting a few RIPE nodes in the same metropolitan area, and use them collectively as a stand-in for a multihomed network. This method actually models IPv6 multihoming where the provider-dependent prefixes advertised by the virtual site are aggregated by its providers. A total of 13 multihomed sites are emulated by this method, a number similar to the study of Akella et al. on multihoming [20]. In our study, 10 sites are dual-homed, 1 is 3-homed, 1 is 4-homed and a last one has 8 providers. One multihomed site is located in the US, one in Japan, and the others in Europe. Unfortunately, since the BGP routing tables of the RIPE test boxes are not available, we cannot compare the SVivaldi path selection directly with the BGP path selection. However, it has been shown in [21] that BGP path lengths are not correlated with their performance [21].

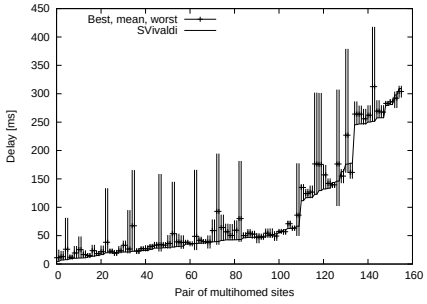

Fig. 10. The delay of the path chosen by SVivaldi for each pair of multihomed sites, in the RIPE data set.

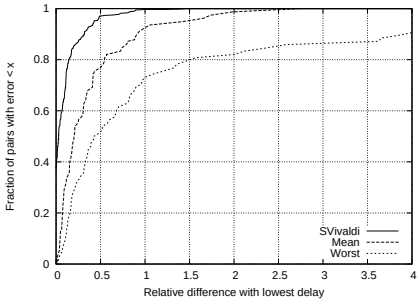

Fig. 11. The cumulative distribution of the relative difference between the delay of the best path and the delay of the path selected by SVivaldi.

Figure 10 shows the delay of the path chosen by SVivaldi for each pair of multihomed sites, sorted by increasing delay. The bars indicate the delay of the best, mean and worst path. We can see that the delay of the worst paths can sometimes be several times larger than than the delay of the best path. In this data set, SVivaldi never selects those really bad paths. For the large majority of multihomed pairs, SVivaldi even manages to select almost the best path. When SVivaldi does not select the best paths, the difference between the delay of the path selected and the best delay is not that large.

Figure 11 shows the relative difference between the path with the lowest delay and the path selected by different path selection algorithms. The figure shows $f(x)$, the fraction of pairs of multihomed sites where a relative difference lower than x is observed. We see that SVivaldi finds the absolute best path in about 40% of the time, and selects a path with a delay at most 20% worse than the best delay for more than 85% of the pairs of multihomed sites. It should be noted that in IDMaps [22], a path selection is considered correct as long as the delay of the selected path is within a factor of 2 times the delay of the best path. Following this criteria, SVivaldi practically never selects a wrong path. Figure 11 confirms that SVivaldi succesfully manages to avoid all really bad paths, i.e. paths where the delay is more than twice the best delay. According to figure 11, these bad paths are not unusual. For example, the delay of the worst path is more than twice the delay of the best path for about 25% of IPv6 multihomed sites pairs. Finally, we can observe again that, besides producing stable coordinates, the use of SVivaldi also produces slightly lower relative errors than the use of Vivaldi.

5 Related Work

Other solutions have been proposed to perform intelligent route selection. A Resilient Overlay Network (RON) [10] is an application-layer overlay on top of the existing Internet., that aims at detecting and recovering from path outages

and periods of degraded performance. RON nodes regularly monitor the quality of paths to each other, and use this information to dynamically select between the direct path and an indirect path via other RON nodes. However, it has been shown that the use of overlays is not necessary to achieve good end-to-end resilience and performance [20]. Our utilization of coordinate systems to select the routes relies on the existing BGP routes, and does not circumvent the BGP's policy-driven routing. Moreover, the use of coordinates is more scalable and does not impose the costs associated to overlays.

The problem of the address selection in IPv6 multihomed sites is addressed by the NAROS approach [23], where IPv6 multihomed hosts inquire a NAROS server in order to select the pair of IPv6 addresses to be used between end hosts. By selecting addresses, a NAROS server roughly select routes. It can thereby provide features like traffic engineering and fault tolerance. By using coordinates together with the NAROS approach, complex path selections involving load-balancing, policy requirements and delay optimization could be developed.

Several vendors also enable route selection [6–8]. However, these solutions rely on actively probing popular nodes. In an IPv6 multihoming environment, these solutions will have to cope with the multiplication of paths available, and their pressure imposed on the infrastructure will increase. Moreover, those solutions cannot help for prefixes for which they do not have active measurements.

The King method [24] is a tool that predicts Internet RTTs between arbitrary end-hosts by using recursive DNS queries. Unlike the use of coordinates, King would need to probe all available paths in order to identify the one with the lowest delay.

6 Conclusion

With IPv6, the use of multiple prefixes increases the number of paths available to a multihomed site. Selecting the path with the lowest delay is important for many interactive and real-time applications.

Our first contribution is to propose the use of a network coordinate system as an efficient and scalable way to help IPv6 hosts in selecting the best source and destination IPv6 prefixes. Our observations have shown that the use of synthetic coordinates is a scalable way to select good paths and to avoid all really bad paths, i.e. paths where the delay is more than twice the best delay. Our experiments with the RIPE data set have shown that we are able to select paths with a delay at most 20% worse than the lowest delay for more than 85% of the pairs of multihomed sites.

Our second contribution is SVivaldi, an improved version of the Vivaldi distributed algorithm for computing synthetic coordinates. Vivaldi suffers from coordinate stability problems. We have shown that adding a loss factor to the Vivaldi algorithm stabilizes the coordinates. Moreover, our observations have shown that our proposed local error predictor allows the nodes to compute more accurate coordinates. SVivaldi is completely distributed and requires no infras-

tructure. It only requires that a multihomed site maintains its coordinates by measuring its RTTs with a small number of other multihomed sites.

Acknowledgments

We thank the RIPE NCC for providing the Test Traffic Measurements Service.

References

1. Agarwal, S., Chuah, C.N., Katz, R.H.: OPCA: Robust interdomain policy routing and traffic control. In: Proceedings OPENARCH. (2003)
2. Atkinson, R., Floyd, S.: IAB concerns & recommendations regarding internet research & evolution. Internet Draft, IAB (2004) <draft-iab-research-funding-03.txt>, work in progress.
3. Huitema, C., Draves, R., Bagnulo, M.: Host-centric IPv6 multihoming. Internet Draft (2004) <draft-huitema-multi6-hosts-03.txt>, work in progress.
4. Huston, G.: Architectural approaches to multi-homing for IPv6. Internet Draft, IETF (2004) <draft-ietf-multi6-architecture-02.txt>, work in progress.
5. de Launois, C., Quoitin, B., Bonaventure, O.: Leveraging Network Performances with IPv6 Multihoming and Multiple Provider-Dependent Aggregatable Prefixes. In: 3rd International Workshop on QoS in Multiservice IP Networks QoS-IP 2005, LNCS 2856, pp.118-179, Catania, Italy (2005)
6. Allen, D.: NPN: Multihoming and route optimization: Finding the best way home. Network Magazine (2002)
7. Bartlett, J.: Optimizing multi-homed connections. Business Communications Review **32** (2002)
8. Cisco Systems, Inc.: Cisco IOS Optimized Edge Routing. `http://www.cisco.com/warp/public/732/Tech/routing/oer/` (2004)
9. Almes, G., Kalidindi, S., Zekauskas, M.: A round-trip delay metric for IPPM. RFC 2681, IETF (1999)
10. Andersen, D., Balakrishnan, H., Kaashoek, F., Morris, R.: Resilient overlay networks. In: Proceedings of ACM SOSP'01. (2001)
11. Gao, L.: On inferring autonomous system relationships in the internet. IEEE/ACM Transactions on Networking **vol. 9, no. 6** (2001)
12. Ng, T.S.E., Zhang, H.: Predicting internet network distance with coordinates-based approaches. In: Proceedings of IEEE INFOCOM'02, New York, USA (2002)
13. Ng, T.S.E., Zhang, H.: A network positioning system for the internet. In: Proceedings of USENIX Conference. (2004)
14. Pias, M., Crowcroft, J., Wilbur, S., Bhatti, S., Harris, T.: Lighthouses for scalable distributed location. In: Proceedings of IPTPS'03. (2003)
15. Dabek, F., Kaashoek, F., Morris, R.: Vivaldi: A decentralized network coordinate system. In: Proceedings of ACM SIGCOMM'04, Portland, Oregon, USA (2004)
16. Draves, R.: Default address selection for internet protocol version 6 (IPv6). RFC 3484, IETF (2003)
17. F.Georgatos, et al.: Providing Active Measurements as a Regular Service for ISP's. In: Proceedings of PAM'01, Amsterdam (2001) `http://www.ripe.net/ttm`.
18. de Launois, C. `http://www.info.ucl.ac.be/people/delaunoi/svivaldi/` (November 2004)

19. Akella, A., et al.: A measurement-based analysis of multihoming. In: Proceedings ACM SIGCOMM'03. (2003)
20. Akella, A., et al.: A comparison of overlay routing and multihoming route control. In: Proceedings ACM SIGCOMM'04. (2004)
21. Huffaker, B., Fomenkov, M., Plummer, D., Moore, D., Claffy, K.: Distance Metrics in the Internet. In: Proc. of IEEE International Telecommunications Symposium (ITS). (2002)
22. Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: IDMaps: A global internet host distance estimation service. In: Proceedings of IEEE/ACM Transactions on Networking. (2001)
23. de Launois, C., Bonaventure, O., Lobelle, M.: The NAROS approach for IPv6 multihoming with traffic engineering. In: Proceedings QoFIS 2003, LNCS 2811, pp. 112-121. (2003)
24. Gummadi, K.P., Saroiu, S., Gribble, S.D.: King: Estimating Latency between Arbitrary Internet End Hosts. In: Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002), Marseille, France (2002)