Finding Routing Shortcuts using an Internet Coordinate System

François Cantin* and Guy Leduc

University of Liège, Belgium
Research Unit in Networking
{francois.cantin,guy.leduc}@ulg.ac.be

Abstract. Overlay routing is a promising way to improve the quality of service in the Internet but its main drawback is scalability: measuring the characteristics of the paths, exchanging the measurement results between the nodes and computing the best routes in the full mesh overlay network generally imply a high consumption of resources. In this paper, we design the basis of a lightweight self-organising one-hop overlay routing mechanism improving the latencies: we define criteria that rely on the information provided by an Internet Coordinate System (ICS) in order to provide a small set of potential one-hop shortcuts for any given path in the network with a small measurement cost. Our best criterion does not guarantee to find the best shortcut for any given path in a network but, even in networks with hundreds or thousands of nodes, it will restrict the search for potential shortcuts to about one or two percent of the total number of nodes.

Keywords: Networking, Overlay routing, Routing shortcuts, Internet coordinate system, Vivaldi.

1 Introduction

Nowadays lots of real-time applications are used in the Internet: voice over IP, online video games, etc. Such applications generally need some QoS (Quality of Service) guarantees and, in particular, low delays between communicating nodes in order to perform correctly. However, since the Internet was not developed with QoS guarantees in mind, the default route between two nodes is not guided by QoS constraints and, in many cases, it is possible to find nodes C that are shortcuts in terms of delays:

$$RTT(A,B) > RTT(A,C) + RTT(C,B) \tag{1}$$

where $RTT(X,Y)$ denotes the RTT (Round Trip Time) between the nodes X and Y, i.e., the time necessary to travel in the network from X to Y and go back from Y to X.

* F. Cantin is a Research Fellow of the Belgian Fund for Research in Industry and Agriculture (FRIA). This work has also been partially supported by the EU under project FP7-Fire ECODE

Our objective is to find such shortcuts for any given path in a network in order to obtain smaller delays: if C is a shortcut for the path AB, we intend to use C as relay instead of sending the data directly from A to B. To know exactly which node C is the best shortcut for a given path AB, we must check if the inequality (1) is true for each node C. To be able to do that for any given path AB, it is necessary to measure the RTT of each path in the network: if there are n nodes in the network, we have to measure permanently the RTT of $O(n^2)$ paths. That measurement traffic can be considerable and its impact on the data traffic can be significant. A solution to avoid a too large resource consumption due to measurements is to use estimations of the RTTs instead of measuring them.

We propose to use an Internet Coordinate System (ICS) [10], namely Vivaldi [2], to estimate RTTs in a scalable manner (i.e., without too much measurement overhead) and use this knowledge to find shortcuts in the network. Using the estimations provided by an ICS to detect shortcuts leads to a major problem: by definition, a shortcut situation is a violation of the triangle inequality that disappears after its (imperfect) embedding in a metric feature space [5]. Since there are no shortcuts in the feature space, equation (1) can never be satisfied when estimated RTTs are used instead of real RTTs. To circumvent this problem we propose some detection criteria that combine estimated and measured RTTs.

The remainder of this paper is organized as follows. In section 2, we review existing work related to our study. In section 3, we briefly introduce the concept of an ICS and, in particular, of the ICS named Vivaldi. We also discuss the problem an ICS faces in the Internet due to triangle inequality violations. In section 4 we give two simple shortcuts detection criteria and present the results obtained with these criteria. In section 5 we combine the two simple criteria into one hybrid criterion to get the advantages of both. We finally conclude and discuss future work in section 6.

2 Related work

Overlay routing is attractive because deploying an overlay requires only the cooperation of the nodes participating in the overlay. It has already been proposed to use overlay routing to improve the performance (and the reliability) of a network: it has been observed in [14] that indirect routing can significantly improve the performance for many paths and, on the basis of these observations, Detour [13] and RON [1] were proposed. The idea of RON is to build a fully connected mesh between the nodes participating in the overlay and to monitor all the paths. If a direct path between two nodes is broken or if it has poor performance it proposes to use relay nodes to reach the destination.

Even if overlay networks seem to be an easy way to improve the performance of the Internet, they are not often used by applications. The main problem is the scalability. Indeed, to obtain the results proposed by RON, it is necessary to measure all the paths and distribute measurements results among the overlay

nodes in order to apply a routing algorithm. These operations become costly if a large number of nodes are members of the overlay.

A first approach to improve the scalability of overlay networks consists in eliminating reduntant overlay paths. For example, Nakao et al. [8, 9] proposed to do that by using topological information (AS-level topology, etc.).

Another solution consists in reducing the communication overhead generated by the exchanges of the measurements results between all the nodes. In RON this overhead is $O(n^2)$ where n is the number of nodes in the overlay. Sontag et al. [15] proposed a protocol allowing each node to find an optimal one-hop path to any other node with a communication overhead $O(n^{1.5})$.

The last way to improve the scalability of overlay routing is to reduce the measurement overhead. Since the objective is to change traffic routes, we suppose that measurements must be done quite frequently to have accurate information about the state of the network. To circumvent this problem Gummadi et al. [3] proposed to route through random relay nodes instead of doing measurements and they observed that it is sufficient to ensure reliability. However, Sontag et al. [15] observed that it is not sufficient to find good alternative paths considering particular metrics like latency. Recently, Lumezanu et al. [7] proposed to use an ICS to detect one-hop shortcuts in a network. Since these situations cannot be reproduced by the estimations provided by the ICS, their idea consists in using estimation errors to find paths that are edges of one-hop shortcuts. We have also explored that solution at the beginning of our work but we gave up because our observations showed that the impact of one-hop shortcuts on an ICS vary according to which shortcut's edges are measured by the ICS (to compute the estimations) and according to the existence of other shortcuts in the network. We have investigated other ways [4, 6] to detect one-hop shortcuts by observing the behaviour of the ICS over time instead of computing the estimation errors at a given time. The results obtained are satisfactory but these detection methods are quite heavy to deploy: they need constant collection of data and the processing of these data has a cost. Since the detection of one-hop shortcuts by analysing the behaviour of an ICS seems difficult we propose to use the estimations provided by the ICS to estimate whether or not some node C is a shortcut for a given path AB.

3 Internet Coordinate Systems

Internet Coordinate Systems (ICS) have been developed to predict end-to-end network latency between all pairs of hosts in a population without extensive measurements. Several prediction algorithms have been proposed, e.g., [10, 2]. In such systems, a coordinate vector (or coordinates in short) in a metric space is computed for each node by doing measurements between this node and a few other nodes called neighbors or reference points (also called landmarks). When each node has its own coordinates, the RTT for a given pair of nodes can be estimated by computing the distance in the metric space between their coordinates.

3.1 Vivaldi

Vivaldi [2] is a popular fully distributed ICS that does not require a particular network infrastructure. In Vivaldi, each node computes its own coordinates by doing measurements with a few (typically 32) other nodes called its neighbors. If m denotes the number of neighbors chosen by each node, the measurement cost of Vivaldi in a network containing n nodes is $O(n \times m)$. Since m is a small value compared to n, this is better than the measurement cost $O(n^2)$ obtained if all paths must be measured.

Multiple coordinate spaces have been proposed: multi-dimentional Euclidean spaces, spherical coordinates, etc. For this work, we use a 10-dimensional Euclidean space where each node computes its coordinates by doing measurements with 32 neighbors. Following the results of [2] this will ensure us quite reliable coordinates.

3.2 Triangle inequality violations

Consider three nodes A, B and C with $RTT(A,B) = 50ms$, $RTT(A,C) = 10ms$ and $RTT(C,B) = 20ms$. There is a TIV (Triangle Inequality Violation) between these nodes because $RTT(A,C) + RTT(C,B) < RTT(A,B)$. By definition, if a node C violates the triangle inequality with a path AB, it means that C is a one-hop shortcut for the path AB. Studies [17] have shown that TIVs are common in the Internet and this is a real problem for an ICS because the triangle inequality must hold in metric spaces. Consequently, TIV situations will inevitably generate estimation errors. Indeed, to try and represent a TIV situation in a metric space, the ICS will have to under-estimate and/or over-estimate the RTT of some paths. It becomes a problem for us if we want to use the estimations to find one-hop shortcuts, because it is impossible to find three nodes such that

$$EST(A,B) > EST(A,C) + EST(C,B) \tag{2}$$

where EST(X,Y) is the estimated RTT between the nodes X and Y.

4 Two basic one-hop shortcut detection criteria

In the previous section, we have seen that using only the estimations provided by an ICS to find the one-hop shortcuts in a network is impossible. So, we have to combine estimations with measurements in order to obtain a shortcut detection criterion. In addition to the estimated RTT of each path in the network, we consider that we can obtain the following measurement results. First, if we look for a shortcut for the path AB, we assume that $RTT(A,B)$ can be measured. Secondly, we assume that we can obtain Vivaldi's measurement results between the nodes and their neighbors.

Given these data, for a given path AB, we want to find criteria that provide a set of C nodes that are probably one-hop shortcuts for that path. As such criteria can provide a potentially large set of nodes, we need also a way to rank the C nodes in order to find the best shortcuts as fast as possible.

4.1 Detection criteria definitions

In [7], Lumezanu et al. stated that, if a node C violates the triangle inequality with a path AB, $EST(A, B)$ is an under-estimation of $RTT(A, B)$. We observed this too: generally, more than 80% of the paths for which there exists at least one (significant) shortcut are under-estimated by the ICS. The following criteria are based on that observation. Indeed, if the estimation of the alternative path is reliable and if the path AB is significantly under-estimated, we will restore the TIV by replacing $EST(A, B)$ by $RTT(A, B)$ in (2).

Estimation Detection Criterion (EDC) is our first criterion. To decide if a node C is a shortcut for a path AB, this criterion compares the measured RTT of the direct path between A and B and the estimated RTT of the alternative path using C as relay. Formally, a node C is considered as a shortcut for the path AB if

$$RTT(A, B) > EST(A, C) + EST(C, B) \tag{3}$$

The potential problem with that criterion is that it uses the values of the estimations provided by the ICS as if there were no estimation error. However, we know that there are estimation errors and, in particular, that these errors cannot be avoided if node C is a shortcut for the path AB. So, using the exact values of the estimated RTTs to find shortcuts is not necessarily a good idea.

Approximation Detection Criterion (ADC) is our second criterion. For a path AB and a node C, we define C_A (resp. C_B) as C's nearest node among A's (resp. B's) Vivaldi neighbors according to the estimated RTTs. Since A and C_A (resp. B and C_B) are neighbors, we assume that $RTT(A, C_A)$ (resp. $RTT(B, C_B)$) is known and can be used by the criterion to approximate the RTT of the alternative path: a node C is considered as a shortcut for the path AB if,

$$RTT(A, B) > RTT(A, C_A) + RTT(C_B, B) \tag{4}$$

4.2 Ranking of the detected C nodes

We have two criteria which, for a given path AB, are able to return a set of C nodes that are probably shortcuts for that path. The problem with such criteria is that they do not provide a set of nodes containing only the best shortcuts: they provide a possibly large set of nodes containing nodes that are important shortcuts, nodes that are less important shortcuts and even nodes that are not shortcuts (detection errors). So, we need a way to rank the C nodes of a set in order to find quickly and easily the best shortcuts in that set. Since we want to find the node C providing the smallest RTT for a path between A and B, we will rank the C nodes by order of provided gain. For a path AB, the *absolute gain* (G_a) and the *relative gain* (G_r) provided by a node C are

$$G_a = RTT(A, B) - (RTT(A, C) + RTT(C, B)) \qquad G_r = \frac{G_a}{RTT(A, B)}$$

If C is a shortcut for the path AB, then G_a and G_r will have positive values and the most interesting shortcut is the one that provides the highest value for these parameters. However, we cannot compute G_a and G_r for all C nodes. Indeed, generally, we do not know the real RTT of the alternative path that uses node C: we only have Vivaldi's estimations for that path. As we have used an estimation/approximation for the RTT of the alternative path in the shortcut detection criteria, we will also use that estimation/approximation in the ranking criteria. The values used to rank the C nodes of a set will be denoted *estimated absolute gain* (EG_a) and *estimated relative gain* (EG_r). The definitions of these values depend on the shortcut detection criterion used to obtain the set of C nodes:

$$EG_a = RTT(A,B) - (VAL(A,C) + VAL(C,B)) \qquad EG_r = \frac{EG_a}{RTT(A,B)} \quad (5)$$

where $VAL(X,Y)$ is the value used by the detection criterion to estimate the RTT of the path XY.

For a path AB, we will rank the C nodes of the set selected by a shortcut detection criterion in decreasing order of their estimated gain. If the nodes with the highest estimated gains are also those with the highest (real) gains then we will find the nodes providing the most interesting shortcuts in the top of the ranking.

4.3 Performance evaluation

To model Internet latency, we used three delay matrices containing results of measurements performed in real networks : two publicly available data sets, the P2PSim data (1740 nodes) [11] and the Meridian data (2500 nodes) [16], and a data set we obtained by doing measurements between 180 nodes on Planetlab [12]. In these matrices, the percentage of paths for which there exists at least one shortcut is respectively 86%, 97% and 67%. Since a shortcut is not necessary useful[1], we define an *interesting shortcut* as a shortcut that provides at least an absolute gain of $10\,ms$ and a relative gain of 10%. The percentage of paths for which there exists at least an interesting shortcut in our matrices is respectively 43%, 83% and 16%. So, searching shortcuts in the networks modelled by these matrices can provide an improvement in terms of delays for many paths.

We have simulated the behaviour of Vivaldi on these three networks by using the P2PSim [11] discrete-event simulator. Each node has computed its coordinates in a 10-dimensional Euclidean space by doing measurements with 32 neighbors. Then, we simply applied our detection criteria using the estimated delay matrices computed with the coordinates obtained at the end of the simulations of Vivaldi. We will now evaluate the quality of the sets of detected nodes provided by our criteria.

[1] For example, for a path AB such that $RTT(A,B) = 100\,ms$, a node C such that $RTT(A,C) + RTT(C,B) = 99\,ms$ is a shortcut that provides an absolute gain of $1\,ms$ and a relative gain of 1%. Since using C as relay for sending data from A to B will add an additional forwarding delay, such shortcuts are useless in practice.

Shortcut detection To evaluate the performance of our detection criteria, we first use the classical true positive rate and false positive rate indicators. For a path AB, a good shortcut detection criterion must detect a node C as a shortcut if it is a shortcut for the path AB (i.e., if it is a positive) and must reject a node C if it is not a shortcut for the path AB (i.e., if it is a negative). The percentage of positives detected as shortcuts is the *true positive rate* (TPR) and the percentage of negatives detected as shortcuts is the *false positive rate* (FPR). We also define the *interesting true positive rate* (ITPR) as the percentage of interesting shortcuts detected as shortcuts by the criterion. A good detection criterion must provide a high (I)TPR and a low FPR.

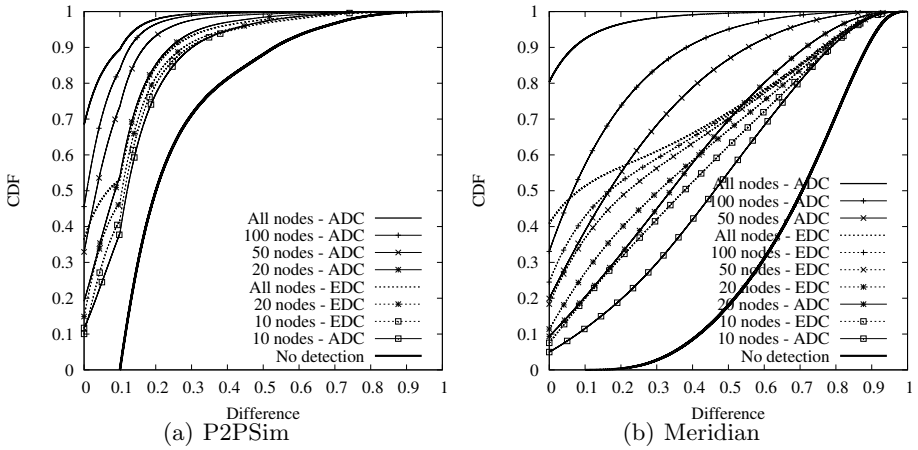
	EDC			ADC		
	TPR	ITPR	FPR	TPR	ITPR	FPR
P2PSim	53%	83%	2%	65%	84%	9%
Meridian	54%	64%	9%	70%	76%	25%
Planetlab	37%	75%	1%	60%	81%	5%

Table 1. EDC and ADC shortcut detection results

The true positive rates and false positive rates obtained with our criteria are given in table 1. We see that the percentage of interesting shortcuts detected as shortcuts (ITPR) is good in most of the cases for both criteria. Furthermore, the percentage of non-shortcuts detected as shortcuts (FPR) is generally quite low. So, these results are satisfactory and, considering these results, EDC seems to perform better than ADC: although ADC is always able to detect slightly more shortcuts than EDC, it also gives more false positives.

Detection of the best shortcuts Being able to detect lots of the shortcuts in a network is one thing, but what matters most is to detect the most interesting shortcuts (those that provide the most important gain). Considering only the paths for which there exists at least one interesting shortcut, the percentage of paths for which the most interesting shortcut is detected in the matrices P2PSim, Meridian and Planetlab is respectively 36%, 41% and 49% with the EDC criterion and 68%, 80% and 70% with the ADC criterion.

Regarding those results ADC seems to be a better criterion than EDC. Indeed, EDC is able to find the best shortcut for 40% of the paths (on average) while the ADC is able to find the best shortcut for more than 70% of the paths in each matrix. However, we must perhaps moderate our conclusion. Firstly because ADC returns large sets of C nodes (including a non-negligible number of false positives) compared to EDC. At the limit, a criterion that detects as shortcut all C nodes will obviously detect the best shortcut for each path but is



(a) P2PSim (b) Meridian

Fig. 1. Comparison of EDC and ADC: Difference of G_r between the best shortcut and the best detected shortcut.

completely useless. So, if we choose to use ADC, we absolutely need a criterion[2] to rank the C nodes of a set in order to keep only a subset of the nodes. Moreover, EDC can give better results than we think. Indeed, even if a criterion cannot find the best shortcut for a path, it may be able to find another shortcut that provides almost the same gain. We will investigate that during the evaluation of the quality of the ranking of the C nodes.

Ranking of the detected nodes For a given matrix and a given detection criterion, we proposed in section 4.2 to rank the C nodes of each path on the basis of the EG_r they provide. We will now evaluate if this gives a ranking with the C nodes providing the best G_r in the first positions. For this evaluation, we only consider the paths for which there exists at least one interesting shortcut.

To do this evaluation, we consider for each path that only the first k C nodes of the ranking are detected by the criterion (for several values of the parameter k). For these subsets of C nodes we compute the difference between the G_r provided by the best existing shortcut and the G_r provided by the best shortcut in the subset. We thus obtain one value for each path and we build the CDF (Cumulative Distribution Function) of these values. The CDFs obtained with different values of the parameter k are given in figure 1.

The graphs named "no detection" in figures 1(a) and 1(b) give the CDF of the G_r provided by the best existing shortcut for each path of the matrix. Indeed, if there is no detection criterion applied, there is no shortcut detected and the difference between the G_r provided by the best existing shortcut and the

[2] Such criterion can also be useful for EDC because, even if the sets of C nodes are generally smaller than those returned by ADC, they can contain tens or hundreds of nodes.

G_r provided by the best detected shortcut is the G_r provided by the best existing shortcut. By applying one shortcut detection criterion, we will detect some shortcuts and, thus, reduce that difference for some paths. Since the computed difference is smaller for more paths, the CDF will rise faster on the graphs.

The graphs named "all nodes - XDC" in the subfigures of figure 1 give the CDF computed by considering all the nodes selected by the shortcut detection criterion XDC (ADC or EDC). This is equivalent to using $k = \infty$. These are the best results that the given detection criterion applied on the given matrix can provide. We can see that ADC still gives better results than EDC considering those graphs. Indeed, with ADC, there are only a small part of the paths for which the difference between the G_r provided by the best existing shortcut and the G_r provided by the best detected shortcut is bigger than 0.2. That means that, for a small part of the paths AB, it is still possible to find another shortcut C that would improve by more than 20% of $RTT(A, B)$ the gain provided by the alternative path proposed by our shortcut detection criterion. This difference is generally bigger with EDC.

Let us see what is the situation if we keep only the first nodes of the rankings. The graphs named "k nodes - XDC" in the subfigures of figure 1 give the CDF computed by considering as detected only the first k nodes of the rankings obtained by using the shortcut detection criterion XDC (ADC or EDC). The first thing we see is that even if we take only a few nodes in the ranking (e.g., 10 nodes), we obtain already a good improvement compared to the situation whithout shortcut detection. We also see that ADC gives better results than EDC only if we keep a sufficient number of nodes: more than 50 nodes for Meridian, more than 20 nodes for P2PSim and more than 5 nodes for Planetlab[3]. Moreover, if we keep a sufficient number of nodes (100 nodes for Meridian, 20 nodes for P2PSim and 10 nodes for Planetlab), we obtain a result with ADC that is better than what we can obtain by considering all the nodes with EDC. The number of nodes to keep to obtain good results may seem important for Meridian but it represents only 4% of the total number of nodes.

Given those results we can conclude that, with ADC, when considering only 5% of the total number of nodes in each matrix (that represents 125 nodes for Meridian, 87 nodes for P2PSim and 9 nodes for Planetlab), we are able to provide a significant improvement of the RTT for lots of paths for which there exists at least one interesting shortcut.

5 Hybrid one-hop shortcut detection criterion

It is possible to obtain better results than those obtained with ADC. Indeed, if it is impossible to find some A's (resp. B's) Vivaldi neighbors near C, the approximation of $RTT(A, C)$ (resp. $RTT(C, B)$) by $RTT(A, C_A)$ (resp. $RTT(C_B, B)$)

[3] Since there are only 180 nodes in the Planetlab matrix, the sets of C nodes returned by the criteria are quite small and keeping all the detected nodes is not really a problem. So, the quality of the ranking is less important for that matrix and we will not show the graphs here.

can be very bad. In such case, using the EDC criterion can provide more reliable detection results even if there are estimation errors. So, we define a *Hybrid Detection Criterion* (HDC) by combining our two basic criteria in order to exploit their advantages.

5.1 Criterion definition

Formally, let C_A (resp. C_B) be C's nearest node among A's (resp. B's) Vivaldi neighbors according to the estimated RTTs. We define

$$VAL(A,C) = \begin{cases} RTT(A,C_A) \text{ if } EST(C_A,C) < threshold \\ EST(A,C) \quad \text{otherwise} \end{cases}$$

$$VAL(C,B) = \begin{cases} RTT(C_B,B) \text{ if } EST(C_B,C) < threshold \\ EST(C,B) \quad \text{otherwise} \end{cases}$$

where *threshold* is a value used to decide if C_A (resp. C_B) is sufficiently near C to obtain a quite good approximation of $RTT(A,C)$ (resp. $RTT(C,B)$) by using $RTT(A,C_A)$ (resp. $RTT(C_B,B)$). To test the HDC criterion, we choose *threshold* equal to 10% of $RTT(A,B)$ when we search a shortcut for the path AB. Fine tuning this parameter is part of our future work. Using these definitions, a node C is considered as a shortcut for the path AB if

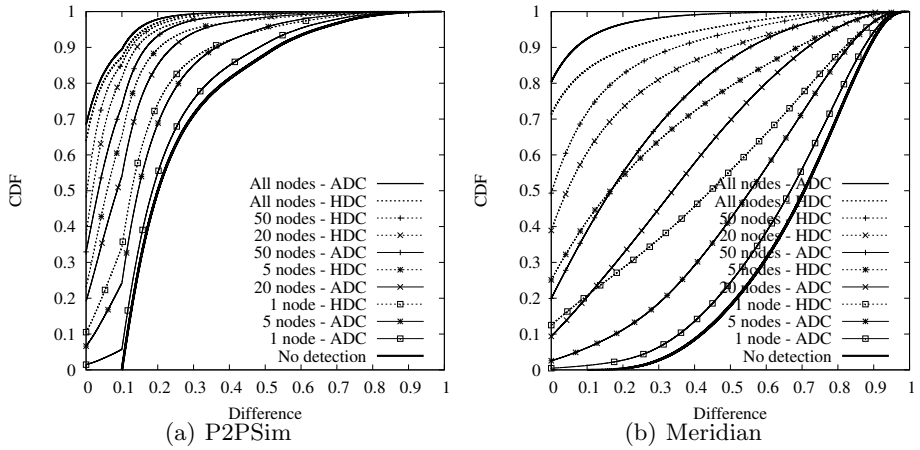
$$RTT(A,B) > VAL(A,C) + VAL(C,B) \tag{6}$$

To rank the C nodes of the set provided by this criterion for a given path AB, we proceed as described in section 4.2.

5.2 Performance evaluation

For the evaluation of HDC, we consider only the paths for which there exists at least one interesting shortcut. Let us begin with the detection of the best shortcuts. The percentages of paths for which the most interesting shortcut is detected with the HDC in the matrices P2PSim, Meridian and Planetlab are respectively 64%, 71% and 74%. These results are better than those obtained with EDC but are worse than those obtained with ADC. So, HDC misses interesting shortcuts that ADC is able to find. Moreover, in figure 2, considering all the nodes detected by the criterion, we can see that ADC is potentially able to provide a better improvement of the latencies in the network than HDC.

However, considering the quality of the ranking of the C nodes, we see in figure 2 (graphs named "k nodes - XDC") that HDC performs a lot better than ADC. Indeed, with HDC, even if we take only the first node of the ranking, we obtain already a significant improvement compared to the situation without shortcut detection. Furthermore, if we only consider the first 5 nodes of a ranking, we obtain better results than if we consider the first 50 nodes of the rankings with ADC. So, with HDC, we can provide a substantial improvement of the RTT for lots of paths by considering only about 2% of the total number of nodes (that represents only 50 nodes for Meridian, 34 nodes for P2PSim and 3 nodes for Planetlab).



(a) P2PSim (b) Meridian

Fig. 2. Comparison of ADC and HDC: Difference of G_r between the best shortcut and the best detected shortcut.

6 Conclusion and future work

In this article, we showed that, for any given path AB, using only the RTT of that path and the information available in an ICS, it is possible to select a small set of nodes containing very likely an interesting one-hop shortcut (but not necessarily the best one) when shortcuts exist for that path. We obtained the best results with our shortcut detection criterion called HDC. With that criterion we are able to limit the number of potential shortcuts for any given path AB to about one or two percent of the total number of nodes in the network. So, to improve significantly the latency between A and B, we will only have to do measurements between A, B and these few candidate nodes to know if they are really shortcuts and which of them is the best shortcut. These results are encouraging and it is probably possible to tune the HDC threshold to obtain even better results.

Our future work will consist in designing a distributed self-organised one-hop routing mechanism based on the encouraging observations reported so far. The distributed and self-organised aspects of the mechanism are natural since the ICS exhibits these properties and a node A can search shortcuts between itself and a node B simply by collecting the coordinates of the other nodes and some more information about B. We will also have to consider problems like churn (arrival and departure of nodes), incentives (motivate a node to participate as relay [7]), etc. Compared to a solution like RON [1] the traffic generated by the measurements will be significantly reduced with our approach. With RON, since each node has to do RTT measurements with all the other nodes this traffic is $O(n^2)$. With our approach, each node measures only with its m neighbors and this traffic is $O(n \times m)$ (with $m \ll n$). Our approach will also reduce the communication overhead generated by the exchanges of the measurements

results. With RON, each node has to send its measurement results to each other nodes. So, the number of messages is $O(n^2)$ and the size of each message is $O(n)$. Consequently, the traffic generated is $O(n^3)$. With our approach this traffic is only $O(n^2)$: each node will have to send a message to each other nodes (n^2 messages) but each message contains only one coordinate vector. Finally, our approach will be less efficient than RON in terms of quality of the shortcuts proposed but it will generate less traffic in the network.

References

1. D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. *SIGOPS Oper. Syst. Rev.*, 35(5):131–145, 2001.
2. F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proc. of SIGCOMM*, Portland, OR, USA, August 2004.
3. K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall. Improving the reliability of internet paths with one-hop source routing. In *Proceedings of OSDI*, Berkeley, CA, USA, 2004. USENIX Association.
4. M. Kaafar, F. Cantin, B. Gueye, and G. Leduc. Detecting triangle inequality violations for internet coordinate systems. In *Proc. of Future Networks 2009 workshop*, Dresden, Germany, June 2009.
5. J. Ledlie, P. Gardner, and M. I. Seltzer. Network coordinates in the wild. In *Proc of NSDI*, Cambridge, UK, April 2007.
6. Y. Liao, M. Kaafar, B. Gueye, F. Cantin, P. Geurts, and G. Leduc. Detecting triangle inequality violations in internet coordinate systems by supervised learning - work in progress. In *Proc. of Networking 2009*, Aachen, Germany, May 2009.
7. C. Lumezanu, R. Baden, D. Levin, N. Spring, and B. Bhattacharjee. Symbiotic relationships in internet routing overlays. In *Proceedings of NSDI*, pages 467–480, Berkeley, CA, USA, 2009. USENIX Association.
8. A. Nakao, L. Peterson, and A. Bavier. A routing underlay for overlay networks. In *Proceedings of SIGCOMM*, pages 11–18, New York, NY, USA, 2003. ACM.
9. A. Nakao, L. Peterson, and A. Bavier. Scalable routing overlay networks. *SIGOPS Oper. Syst. Rev.*, 40(1):49–61, 2006.
10. T. S. E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proc. of INFOCOM*, New York, NY, USA, June 2002.
11. *A simulator for peer-to-peer protocols.* `http://www.pdos.lcs.mit.edu/p2psim/index.html`.
12. *PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services.* `http://www.planet-lab.org`.
13. S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: Informed internet routing and transport. *IEEE Micro*, 19(1):50–59, 1999.
14. S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of internet path selection. *SIGCOMM Comput. Commun. Rev.*, 29(4):289–299, 1999.
15. D. Sontag, Y. Zhang, A. Phanishayee, D. G. Andersen, and K. D. Scaling all-pairs overlay routing. In *Proceedings of CoNEXT*, Rome, Italy, December 2009.
16. B. Wong, A. Slivkins, and E. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *Proc. of the ACM SIGCOMM*, August 2005.
17. H. Zheng, E. K. Lua, M. Pias, and T. G. Griffin. Internet routing policies and round-trip-times. In *Proc. of PAM*, LNCS 3431, Boston, MA, USA, March 2005.